

# **Real Fingerprint Detection System (RFDS) Based on Image Quality Measures and Six Classifiers**

**Abdurahman Ibrahim Mriheel**

Submitted to the  
Institute of Graduate Studies and Research  
in partial fulfillment of the requirements for the degree of

Master of Science  
in  
Computer Engineering

Eastern Mediterranean University  
February 2020  
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

---

Prof. Dr. Ali Hakan Ulusoy  
Director

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

---

Prof. Dr. Işık Aybay  
Chair, Department of Computer  
Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

---

Assoc. Prof. Dr. Alexander Chefranov  
Supervisor

---

Examining Committee

1. Assoc. Prof. Dr. Alexander Chefranov

2. Assoc. Prof. Dr. Önsen Toygar

3. Asst. Prof. Dr. Mehtap Köse Ulukök

## ABSTRACT

Fingerprint detection in biometrics is an important field of study in our new modern world, many forensic departments around the world use fingerprints as the key to detect criminals and bring them justice. To improve the accuracy of fingerprint detection system we implemented a Real Fingerprint Detection System (RFDS) that has high performance level of detecting real and fake fingerprint images. In this thesis, we present an RFDS system based on image quality measures (IQM's) to detect real fingerprint images and fake fingerprint images. We performed different RFDS experiments with 25, 10, 15, and 5 IQM's; they showed sufficient quality of real and fake fingerprint images detection. We compared our RFDS using 25, 15, 10, and 5 IQM's with RFDS that has used 25 IQMs. Based on the comparison in this thesis we can conclude that the best result from all these RFDS is the one with 25 IQMs, because it's HTER score is the minimum one with 0.3%, and the worst RFDS is the one with the 15 IQMs which has the maximum HTER score with 14.8%.

**Keywords:** Biometrics, Image Quality Measure, Real and Fake Fingerprint Image, Classifier.

# ÖZ

Biyometride parmak izi tespiti, yeni modern dünyamızda önemli bir çalışma alanıdır, dünyadaki birçok adli departman, suçluları tespit etmek ve adalet getirmek için parmak izi kullanmaktadır. Parmak izi algılama sisteminin doğruluğunu artırmak için, gerçek ve sahte parmak izi görüntülerini algılamada yüksek performans seviyesine sahip bir Gerçek Parmak İzi Algılama sistemi (RFDS) uyguladık. Bu tezde, gerçek parmak izi görüntülerini ve sahte parmak izi görüntülerini tespit etmek için görüntü kalitesi ölçümlerine (IQM'ler) dayanan bir RFDS sistemi sunuyoruz. 25, 15, 10 ve 5 IQM'lerle farklı RFDS deneyleri gerçekleştirdik; yeterli kalitede gerçek ve sahte parmak izi görüntüleri algılama özelliği gösterdiler. 25, 15, 10 ve 5 IQM kullanarak RFDS'imizi 25 IQM kullanan RFDS ile karşılaştırdık. Bu tezdeki karşılaştırmaya dayanarak, tüm bu RFDS'lerden en iyi sonucun 25 IQM'ye sahip olduğu sonucuna varabiliriz, çünkü HTER puanı % 0.3 ile en düşük olanıdır ve en kötü RFDS ise %14.8 HTER puanı ile 15 IQM'ye sahip olanıdır.

**Anahtar Kelimeler:** Biyometri, Görüntü Kalitesi Ölçüsü, Gerçek Ve Sahte Parmak İzi Görüntüsü, Sınıflandırıcı.

# **DEDECATION**

To My Family

## **ACKNOWLEDGMENT**

I would like to thank my supervisor (Assoc. Prof. Dr. Alexander Chefranov) who gave me the golden opportunity to do this wonderful project on the topic (Real Fingerprint Detection System (RFDS) Based on Image Quality Measures and Six Classifiers), which also helped me in doing a lot of research and I came to know about so many new things, I am thankful to him.

Secondly, I would like also to express my special thanks of gratitude to my parents and friends who helped me in finalizing this project within the limited time frame

# TABLE OF CONTENTS

|                                                                                                                                  |      |
|----------------------------------------------------------------------------------------------------------------------------------|------|
| ABSTRACT .....                                                                                                                   | iii  |
| ÖZ .....                                                                                                                         | iv   |
| DEDICATION .....                                                                                                                 | v    |
| ACKNOWLEDGMENT .....                                                                                                             | vi   |
| LIST OF TABLES .....                                                                                                             | xi   |
| LIST OF FIGURES.....                                                                                                             | xii  |
| LIST OF ABBREVIATIONS .....                                                                                                      | xvii |
| 1 INTRODUCTION.....                                                                                                              | 1    |
| 1.1 Definition of Common Biometric Terms [3] .....                                                                               | 2    |
| 1.2 Fingerprint Detection Systems .....                                                                                          | 3    |
| 1.3 Key Terms Used in Fingerprint Images Analysis .....                                                                          | 3    |
| 1.4 Summary .....                                                                                                                | 4    |
| 2 ANALYSIS F KNOWN RFDS AND PROBLEM DEFINITION .....                                                                             | 5    |
| 2.1 Analysis of Known RFDS.....                                                                                                  | 5    |
| 2.1.1 Image Quality Assessment for Fake Biometric Detection: Application to<br>Iris, Fingerprint, And Face Recognition [6] ..... | 5    |
| 2.1.2 Full Reference (FR) Image Quality Measures Used In [6] .....                                                               | 6    |
| 2.1.3 Classifiers Used in [6] and Our RFDS .....                                                                                 | 12   |
| 2.1.3.1 RFDS Experiments In [6].....                                                                                             | 15   |
| 2.1.4 Fingerprint Liveness Detection Based on Quality Measures [27].....                                                         | 17   |
| 2.1.4.1 RFDS Experiments In [27].....                                                                                            | 17   |
| 2.1.5 A High-Performance Fingerprint Liveness Detection Method Based on<br>Quality Related Features [28].....                    | 18   |

|                                                                                                                                            |    |
|--------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1.5.1 RFDS Experiments In [28].....                                                                                                      | 19 |
| 2.1.6 Fingerprint Identification in Biometric Security Systems. International<br>Journal of Computer and Electrical Engineering [29] ..... | 20 |
| 2.1.6.1 RFDS Experiments In [29].....                                                                                                      | 21 |
| 2.1.7 Fingerprint Spoof Detection Using Quality Features [30] .....                                                                        | 21 |
| 2.1.7.1 RFDS Experiments In [30].....                                                                                                      | 21 |
| 2.2 Problem Definition .....                                                                                                               | 22 |
| 2.3 Summary .....                                                                                                                          | 23 |
| 3 DESIGN AND IMPLEMENTATION OF RFDS.....                                                                                                   | 24 |
| 3.1 Design of RFDS .....                                                                                                                   | 24 |
| 3.2 Implementation of RFDS .....                                                                                                           | 25 |
| 3.2.1 Implementation and Testing of Gaussian Filter .....                                                                                  | 25 |
| 3.2.2 Implementation and Testing of Feature Extraction .....                                                                               | 28 |
| 3.2.2.1 Implementation and Testing of MSE (2.1) .....                                                                                      | 29 |
| 3.2.2.2 Implementation and Testing of PSNR (2.2).....                                                                                      | 30 |
| 3.2.2.3 Implementation and Testing of SNR (2.3).....                                                                                       | 31 |
| 3.2.2.4 Implementation and Testing of SC (2.4).....                                                                                        | 32 |
| 3.2.2.5 Implementation and Testing of MD (2.5) .....                                                                                       | 33 |
| 3.2.2.6 Implementation and Testing of AD (2.6).....                                                                                        | 34 |
| 3.2.2.7 Implementation and Testing of NAE (2.7) .....                                                                                      | 35 |
| 3.2.2.8 Implementation and Testing of RAMD (2.8).....                                                                                      | 36 |
| 3.2.2.9 Implementation and Testing of NCC (2.9) .....                                                                                      | 37 |
| 3.2.2.10 Implementation and Testing of TED (2.10).....                                                                                     | 38 |
| 3.2.2.11 Implementation and Testing of TCD (2.11).....                                                                                     | 39 |
| 3.2.2.12 Implementation and Testing of Spectral Phase Error (SPE).....                                                                     | 41 |

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| 3.2.2.13 Implementation and Testing of SME (2.13) .....                              | 43 |
| 3.2.2.14 Implementation and Testing of Gradient Phase Error (GPE) .....              | 47 |
| 3.2.2.15 Implementation and Testing of Gradient Magnitude Error (GME)<br>.....       | 50 |
| 3.2.2.16 Implementation and Testing of Mean Angle Similarity (MAS). 52               |    |
| 3.2.2.17 Implementation and Testing of Mean Angle Similarity (MAMS)<br>.....         | 53 |
| 3.2.2.18 Implementation and testing of Laplacian Mean Squared Error<br>(LMSE) .....  | 54 |
| 3.2.3 Implementation of Training Structure .....                                     | 58 |
| 3.2.4 Implementation of The Classification Process .....                             | 62 |
| 3.3 Development of A Method of Defining the Best IQMs.....                           | 67 |
| 3.3.1 Implementation and Testing of The Method of Defining the Best IQMs             | 69 |
| 3.3.1.1 Calculation of Defining the Best 5 IQMs .....                                | 70 |
| 3.4 Summary .....                                                                    | 74 |
| 4 EXPERIMENTS ON RFDS .....                                                          | 75 |
| 4.1 Experimental Setup .....                                                         | 75 |
| 4.2 Database Used .....                                                              | 75 |
| 4.3 Classification Methods Used.....                                                 | 76 |
| 4.4 Code Parts for Experiments Conducting .....                                      | 76 |
| 4.5 Experimental Results.....                                                        | 77 |
| 4.5.1 Experiments on the RFDS Training and Classification with 15 IQMs ....          | 77 |
| 4.5.2 Experiments on The Rfds Training and Classification with Best 10 IQMs<br>..... | 78 |

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| 4.5.3 Experiments on The RFDS Training and Classification with Best 5 IQMs ..... | 80  |
| 4.5.4 Experiments on The RFDS Training and Classification With 25 IQMs..         | 81  |
| 4.5.5 Comparision of the RFDS Performance With 5, 10, 15, And 25 Iqms Used ..... | 82  |
| 4.6 Summary .....                                                                | 83  |
| 5 CONCLUSION .....                                                               | 84  |
| REFERENCES.....                                                                  | 85  |
| APPENDICES.....                                                                  | 90  |
| Appendix A: Feature Extraction Code: .....                                       | 91  |
| Appendix B: Code for Ranking IQM's .....                                         | 116 |
| Appendix C: Screenshots and Tables of Our Rfds Results.....                      | 119 |

## LIST OF TABLES

|                                                                                  |    |
|----------------------------------------------------------------------------------|----|
| Table 1: RFDS [6] results using three different type of fingerprint sensors..... | 16 |
| Table 2: The results for each sensor [27]. .....                                 | 18 |
| Table 3: Performance result of the two databases used in [28].....               | 19 |
| Table 4: Training results with 15 IQMs. ....                                     | 77 |
| Table 5: Classification results with 15 IQMs.....                                | 78 |
| Table 6: Training results with best 10 IQMs. ....                                | 79 |
| Table 7: Classification results with best 10 IQMs. ....                          | 79 |
| Table 8: Training results with best 5 IQMs. ....                                 | 80 |
| Table 9: Classification results with best 5 IQMs. ....                           | 81 |
| Table 10: Training results with 25 IQMs.....                                     | 81 |
| Table 11: Classification results with 25 IQM's. ....                             | 82 |
| Table 12: Result of comparison HTER (%) between our RFDS and RFDS [6].....       | 83 |

## LIST OF FIGURES

|                                                                                  |    |
|----------------------------------------------------------------------------------|----|
| Figure 1: Fingerprints ridge ending (a) and ridge bifurcation (b) [5]. .....     | 4  |
| Figure 2: The boundary between two classes (Hyper Plane( $\hat{W}$ )). .....     | 14 |
| Figure 3: 'S' shaped logistic function. ....                                     | 14 |
| Figure 4: A new cell (fingerprint image) with unknown category to classify. .... | 15 |
| Figure 5: Average classification error (ACE) results in [30]. ....               | 22 |
| Figure 6: RFDS training and detection structure. ....                            | 25 |
| Figure 7: Gaussian filter code and the Gaussian filter obtained from it. ....    | 26 |
| Figure 8: Result of the original image with Gaussian filter applied to it. ....  | 28 |
| Figure 9: MATLAB result of MSE calculation. ....                                 | 30 |
| Figure 10: MATLAB result of PSNR calculation. ....                               | 31 |
| Figure 11: MATLAB result of SNR calculation. ....                                | 32 |
| Figure 12: MATLAB result of SC calculation. ....                                 | 33 |
| Figure 13: MATLAB result of MD calculation. ....                                 | 34 |
| Figure 14: MATLAB result of AD calculation. ....                                 | 35 |
| Figure 15: MATLAB result of NAE calculation. ....                                | 36 |
| Figure 16: MATLAB result of RAMD calculation with R=7. ....                      | 37 |
| Figure 17: MATLAB result of NCC calculation. ....                                | 38 |
| Figure 18: Matlab result for calculating TED of fingerprint image. ....          | 39 |
| Figure 19: MATLAB result of TCD calculation. ....                                | 40 |
| Figure 20: MATLAB result of SPE calculation. ....                                | 43 |
| Figure 21: MATLAB result of SME calculation. ....                                | 47 |
| Figure 22: MATLAB result of GPE calculation. ....                                | 49 |
| Figure 23: MATLAB result of GME calculation. ....                                | 52 |

|                                                                                                                                                                                         |     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 24: MATLAB result of MAS calculation.....                                                                                                                                        | 53  |
| Figure 25: MATLAB result of MAMS calculation. ....                                                                                                                                      | 54  |
| Figure 26: Matlab result of LMSE calculation. ....                                                                                                                                      | 56  |
| Figure 27: Table which has all the fingerprint image results of IQMS we used. ....                                                                                                      | 59  |
| Figure 28: The arrow shows a button used to call the classification learner app in MATLAB.....                                                                                          | 60  |
| Figure 29: Start new session to import results. ....                                                                                                                                    | 60  |
| Figure 30: Selection of our dataset, predictors and response in MATLAB.....                                                                                                             | 61  |
| Figure 31: Selection of the classifier for our table imported and training result. ....                                                                                                 | 62  |
| Figure 32: Exporting training model of 100 real and fake fingerprint images. ....                                                                                                       | 64  |
| Figure 33: After exporting the training model, we can use it to make prediction by the code provided in the command window highlighted with a rectangle.....                            | 64  |
| Figure 34: The highlighted rectangle shows the code used to classify images according to our exported training model.....                                                               | 65  |
| Figure 35: Results of 10 real and fake images classified correctly using the classification code.....                                                                                   | 66  |
| Figure 36: Fingerprint images used in our expirments, (A) fake fingerprint image from the file” U01_F_Fc_Li_01.Bmp”, (B) real fingerprint image form the file” U01_O_Fc_Li_01.Bmp”..... | 67  |
| Figure 37: Numerical axis for MSE (A) And MD (B), showing MSE of fake (real) fingerprint images. And the respective averages calculated by (3.16) and (3.17)....                        | 67  |
| Figure 38: The list of the best IQMs (in the red rectangle).....                                                                                                                        | 73  |
| Figure 39: List of our best 15,10,5 IQMs used in our RFDS expirments.....                                                                                                               | 74  |
| Figure 40: QDA training result of 100 fingerprint images using 15 IQMs. ....                                                                                                            | 120 |
| Figure 41: Linear SVM training result of 100 fingerprint images using 15 IQMs..                                                                                                         | 120 |

|                                                                                               |     |
|-----------------------------------------------------------------------------------------------|-----|
| Figure 42: Quadratic SVM training result of 100 fingerprint images using 15 IQMs.             | 121 |
| Figure 43: Fine KNN training result of 100 fingerprint images using 15 IQMs. ....             | 121 |
| Figure 44: Logistic Regression training result of 100 fingerprint images using 15 IQMs. ....  | 122 |
| Figure 45: LDA classification result of 1000 fingerprint images using 15 IQMs. ...            | 123 |
| Figure 46: QDA classification result of 1000 fingerprint images using 15 IQMs...              | 123 |
| Figure 47: Linear SVM classification result of 1000 fingerprint images using 15 IQMs.         | 124 |
| Figure 48: Quadratic SVM classification result of 1000 fingerprint images using 15 IQMs. .... | 124 |
| Figure 49: Fine KNN classification result of 1000 fingerprint images using 15 IQMs.           | 125 |
| Figure 50: Logistic Regression classification of 1000 fingerprint images using 15 IQMs. ....  | 125 |
| Figure 51: LDA training result of 100 fingerprint images using 10 IQMs. ....                  | 126 |
| Figure 52: QDA training result of 100 fingerprint images using 10 IQMs. ....                  | 126 |
| Figure 53: Linear SVM training result of 100 fingerprint images using 10 IQMs..               | 127 |
| Figure 54: Quadratic SVM training result of 100 fingerprint images using 10 IQMs.             | 127 |
| Figure 55: Fine KNN training result of 100 fingerprint images using 10 IQMs. ....             | 128 |
| Figure 56: Logistic Regression training result of 100 fingerprint images using 10 IQMs. ....  | 128 |
| Figure 57: LDA Classification result of 1000 fingerprint images using 10 IQMs. .              | 129 |

|                                                                                                   |     |
|---------------------------------------------------------------------------------------------------|-----|
| Figure 58: QDA Classification Result of 1000 Fingerprint Images Using 10 IQMs.                    | 129 |
| Figure 59: Linear SVM classification result of 1000 fingerprint images using 10 IQMs.             | 130 |
| Figure 60: Quadratic SVM classification result of 1000 fingerprint images using 10 IQMs.          | 130 |
| Figure 61: Fine KNN classification result of 1000 fingerprint images using 10 IQMs.               | 131 |
| Figure 62: Logistic Regression classification result of 1000 fingerprint images using 10 IQMs.    | 131 |
| Figure 63: LDA training result of 100 fingerprint images using 5 IQMs.                            | 132 |
| Figure 64: QDA training result of 100 fingerprint images using 5 IQMs.                            | 132 |
| Figure 65: Linear SVM training result of 100 fingerprint images using 5 IQMs.                     | 133 |
| Figure 66: Quadratic SVM training result of 100 fingerprint images using 5 IQMs.                  | 133 |
| Figure 67: Fine KNN training result of 100 fingerprint images using 5 IQMs.                       | 134 |
| Figure 68: Logistic Regression training result displayed for 100 fingerprint images using 5 IQMs. | 134 |
| Figure 69: LDA classification result of 1000 fingerprint images using 5 IQMs.                     | 135 |
| Figure 70: QDA Classification of 1000 Fingerprint Images Using 5 IQMs.                            | 135 |
| Figure 71: Linear SVM LDA classification result of 1000 fingerprint images using 5 IQMs.          | 136 |
| Figure 72: Quadaratic SVM classification result of 1000 fingerprint images using 5 IQMs.          | 136 |

|                                                                                                        |     |
|--------------------------------------------------------------------------------------------------------|-----|
| Figure 73: Fine KNN classification result of 1000 fingerprint images using 5 IQMs.<br>.....            | 137 |
| Figure 74: Logistic Regression classification result of 1000 fingerprint images using 5<br>IQMs. ....  | 137 |
| Figure 75: LDA training result of 100 fingerprint images using 25 IQMs.....                            | 138 |
| Figure 76: QDA Training Result of 100 Fingerprint Images Using 25 IQMs.....                            | 138 |
| Figure 77: Linear SVM training result of 100 fingerprint images using 25 IQMs..                        | 139 |
| Figure 78: Quadaratic SVM training of 100 fingerprint images using 25 IQMs.....                        | 139 |
| Figure 79: Fine KNN training result of 100 fingerprint images using 25 IQMs. ....                      | 140 |
| Figure 80: Logistic Regression training result of 100 fingerprint images using 25<br>IQMs. ....        | 140 |
| Figure 81: LDA classification result of 1000 fingerprint images using 25 IQMs..                        | 141 |
| Figure 82: QDA classification result of 1000 fingerprint images using 25 IQMs...                       | 141 |
| Figure 83: Linear SVM classification result of 1000 fingerprint images using 25 IQMs.<br>.....         | 142 |
| Figure 84: Quadaratic SVM classification result of 1000 fingerprint images using 25<br>IQMs. ....      | 142 |
| Figure 85: Fine KNN classification result of 1000 fingerprint images using 25 IQMs.<br>.....           | 143 |
| Figure 86: Logistic Regression classification result of 1000 fingerprint images using<br>25 IQMs. .... | 143 |

## LIST OF ABBREVIATIONS

|        |                                         |
|--------|-----------------------------------------|
| ACE    | Average Classification Error            |
| AD     | Average Difference                      |
| FAR    | False Accept Rate                       |
| FFR    | False Fake Rate                         |
| FFR    | False Fake Rate                         |
| FLR    | False Living Rate                       |
| FR     | Full Reference                          |
| FR-IQA | Full Reference Image Quality Assessment |
| GME    | Gradient Magnitude Error                |
| GPE    | Gradient Phase Error                    |
| HTER   | Half Total Error Rate                   |
| IQM    | Image Quality Measures                  |
| LDA    | Linear Discriminant Analysis            |
| LMSE   | Laplacian Mse                           |
| MAS    | Mean Angle Similarity                   |
| MD     | Maximum Difference                      |
| MSE    | Mean Squared Error                      |
| NAE    | Normalized Absolute Error               |
| NCC    | Normalized Cross Correlation            |
| NR     | No Reference                            |
| NR-IQA | No Reference Image Quality Assessment   |
| PSNR   | Peak Signal To Noise Ratio              |

|      |                                   |
|------|-----------------------------------|
| QDA  | Quadratic Discriminant Analysis   |
| RAMD | R-Averaged Md                     |
| RFDS | Real Fingerprint Detection System |
| SC   | Structural Content                |
| SME  | Spectral Magnitude Error          |
| SNR  | Signal To Noise Ratio             |
| SPE  | Spectral Phase Error              |
| TCD  | Total Corner Difference           |
| TED  | Total Edge Difference             |

# Chapter 1

## INTRODUCTION

Biometric recognition system is a mixture of hardware and software means used to differentiate and identify individuals [1]. Biometric recognition system obtains data such as fingerprint image of an individual, and then applies comparison to the image against the stored fingerprint images in the database. If the input fingerprint image matches one of the available images in the database, the system will accept it as **Real** image, otherwise it will be considered as **Fake** and will be rejected by the system. A biometric recognition system offers a natural and consistent solution to the problems of individual identification [1].

We can categorize the biometric methods used by biometric systems into two classes [2]:

1. Physiological-based methods use face, fingerprint, voice, retinal, DNA analysis, and measure the physiological characteristics of a person.
2. Behavior-based methods use signature, key stroke analysis and measure behavioral characteristics.

Biometric recognition systems can work in two modes:

- **Identification Mode:** The system recognizes a person searching a large database of registered ones for a match.

- **Verification Mode:** The system confirms a person's claimed identity from his previous registered pattern.

## 1.1 Definition of Common Biometric Terms [3]

- **Enrollment or Registration:** The individual's biometric data are obtained, processed and saved as a template for ongoing usage in a biometric system; this is known as enrollment, or registration process. The template will be used for identification or verification in further stages.

- **Biometric Data:** Any data related to the physiological and behavioral characteristics of an individual is known as biometric data, which can also be referred as raw biometric data.

- **Presentation:** In this procedure, the individual presents his or her biometric data to the hardware that is used to gather data. For example, placing your finger on a biometrical sensor.

- **Feature Extraction:** The procedure of finding and encoding unique characteristics from biometric data in order to produce a template is known as feature extraction. Feature extraction takes place during enrollment and verification stage.

- **Image Quality Measures (IQM's):** Mathematical measures applied to identify the quality of the image such as its color accuracy, noise, and sharpness, the results we get from the iqms assist us to identify or verify if the image is real or fake [4].

▪ **Classification:** A saved template will be compared with other templates during identification or verification stage. Based on the comparison we can classify if the individual is verified or not.

## **1.2 Fingerprint Detection Systems**

Fingerprint is unique for every individual; they are mainly used for identifying individuals. Real fingerprint detection systems (RFDS) are one of the most known biometric systems used in many applications such as border control and forensic department. RFDS are systems that identify or verify any fingerprint image fed to the system as real or fake image. Rfds systems are described in [6], [18], [19], [20], and [21]. RFDS performance is very critical, any failure can cause harm to individuals and governments. For example, detecting the innocent individual's fingerprint as the criminal, or passing the border control of a country with another individual's fingerprint, and system will detect it as true and make you pass the border control. These kinds of problems shouldn't occur as they will cause injustice and security problems to individuals and the governments. Based on this major problem, we chose RFDS as our main area of study in this thesis. We will show different RFDS algorithms and implement the best of them, in which we will try to reduce the amount of wrong fingerprint detection as minimum as possible.

## **1.3 Key Terms Used in Fingerprint Images Analysis**

1. Ridge ending: the point, at which the ridge terminates, shown in figure 1, a.
2. Ridge bifurcation: when one ridge splits in-to two or more ridges, shown in figure 1, b.
3. Minutiae: major features of a fingerprint like ridge endings and ridge bifurcations.

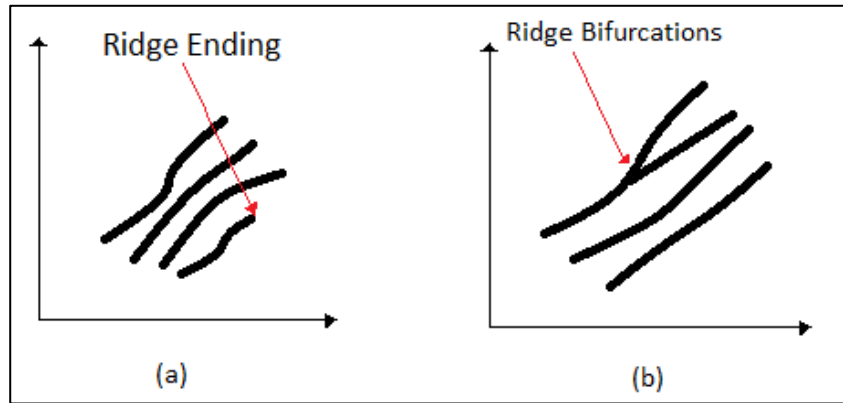


Figure 1: Fingerprints ridge ending (a) and ridge bifurcation (b) [5].

## 1.4 Summary

In this thesis, we implement a real fingerprint detection system (RFDS) using feature extraction and classification algorithm to classify the registered or enrolled fingerprint image. In Chapter 2, we review known fingerprint detection systems and define the problems. In Chapter 3, we show the general diagram of the RFDS, its implementation, classification algorithm used, and defining the best 10 and 5 IQMs. In chapter 4, we describe the experiments conducted on RFDS, steps taken for applying these experiments, and the results obtained. Chapter 5 contains the conclusion of the thesis and future work.

## Chapter 2

### ANALYSIS F KNOWN RFDS AND PROBLEM

#### DEFINITION

In this chapter, we analyze some known rfds and list their experimental results. We also list the problems to be analyzed and tested in this thesis.

#### 2.1 Analysis of Known RFDS

##### 2.1.1 Image Quality Assessment for Fake Biometric Detection: Application to Iris, Fingerprint, And Face Recognition [6]

In RFDS, they input images and extract their features using 25 IQMs; then, these images will be classified on **Real/ Fake**. The details of how RFDS works are as follows [6]:

1. Training model should be created by feeding in to the system with good amount of real and fake biometric images of users. The images fed to system for creating the training model are known as the training data. The explanation of how the training model is created is in Chapter 3, section 3.2.3. Once the training model is created, it is exported to the system making the system ready to predict between real and fake images based on the training model.
2. Image enrolment, input some (Real or Fake) gray scale image to the system for identification.
3. Features of the input image will be extracted by the system using the IQM's.

Classification of the input image, the system will classify if the fingerprint image is real or fake image based on the training model exported to it. Steps of the classification implementation are in 3.2.4. Image quality measures in [6].

In RFDS [6], 25 image quality measures have been used split into two parts: 21 of the measures are full reference image quality measurements (image quality measure that requires a reference of similar biometric image type to determine the quality level of the image fed to the system), and 4 are no reference image quality measures (they don't require a reference of similar biometric image type to determine the quality level of an image). Full reference image quality measures to extract the quality features of the original image and enhanced image (original image with adjusted noise is known as enhanced image); no reference image quality measures evaluate the state of the original image inputted to the system. Next we describe the 25 image quality measures used.

### 2.1.2 Full Reference (FR) Image Quality Measures Used In [6]

**1. Mean Squared Error (MSE):** It calculates the sum of squared difference (error) between the original image (I) and the enhanced image ( $\bar{I}$ ). The mean squared error formula (2.1) [7].

$$MSE(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (I_{i,j} - \bar{I}_{i,j})^2. \quad (2.1)$$

Where **m**, **n** is the number of columns and rows original image has.

**2. Peak Signal-To-Noise Ratio (PSNR):** It calculates the ratio between the signal power and distortion noise. The peak signal to noise ratio formula (2.2) [8,9].

$$PSNR(I, \bar{I}) = 10 \log_{10} \left( \frac{\text{Max}(I^2)}{MSE(I, \bar{I})} \right). \quad (2.2)$$

**3. Signal-To-Noise Ratio (SNR):** Calculates the ratio of power in the input signal to the ratio of the noise power, SNR can be defined as the ratio of wanted information to unwanted. Signal-To-Noise Ratio formula (2.3) [10].

$$\text{SNR} (I, \bar{I}) = 10 \log_{10} \left( \frac{\sum_{i=1}^N \sum_{j=1}^M (I_{i,j})^2}{N \cdot M \cdot \text{MSE}(I, \bar{I})} \right). \quad (2.3)$$

**4. Structural Content (SC):** Calculates the sum of the original image (I) pixel values squared divided by the sum of the enhanced image pixel values ( $\bar{I}$ ) squared. Structural content formula (2.4) [11].

$$\text{SC} (I, \bar{I}) = \frac{\sum_{i=1}^N \sum_{j=1}^M (I_{i,j})^2}{\sum_{i=1}^N \sum_{j=1}^M (\bar{I}_{i,j})^2}. \quad (2.4)$$

**5. Maximum Difference (MD):** Calculates absolute maximum difference of the original image (I) pixel values subtracted with enhanced image ( $\bar{I}$ ) pixel values. Maximum difference formula (2.5) [11].

$$\text{MD} (I, \bar{I}) = \text{Max} |I_{i,j} - \bar{I}_{i,j}|. \quad (2.5)$$

**6. Average Difference (AD):** Calculates sum of difference of original image (I) subtracted with enhanced image ( $\bar{I}$ ) divided with total pixels amount the image has. The average difference formula (2.6) [11].

$$\text{AD} (I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (I_{i,j} - \bar{I}_{i,j}). \quad (2.6)$$

**7. Normalized Absolute Error (NAE):** Calculates sum of absolute difference of the original image (I) subtracted with the enhanced image ( $\bar{I}$ ) and divided with sum of absolute pixel values of the original. The normalized absolute error formula (2.7) [11].

$$\text{NAE} (I, \bar{I}) = \frac{\sum_{i=1}^N \sum_{j=1}^M |I_{i,j} - \bar{I}_{i,j}|}{\sum_{i=1}^N \sum_{j=1}^M |I_{i,j}|}. \quad (2.7)$$

**8. R-Averaged Maximum Difference (RAMD):** Calculates the maximum difference summation of **R** between the original image (I) and the enhanced image ( $\bar{I}$ ) averaged by **R** value. The R-averaged maximum difference formula [7] (2.8).

$$\text{RAMD} (I, \bar{I}, r) = \frac{1}{R} \sum_{r=1}^R \max_r |I_{i,j} - \bar{I}_{i,j}|. \quad (2.8)$$

Where the value of **R** is the number of times we iterate and select the maximum difference between the pixel of the original and enhanced images.

**9. Normalized Cross Correlation (NCC):** Calculates the sum of multiplying the original image (I) and the enhanced image ( $\bar{I}$ ), divided by the sum squared of the original image (I). the normalized cross correlation formula (2.9) [11].

$$\text{NCC} (I, \bar{I}) = \frac{\sum_{i=1}^N \sum_{j=1}^M (I_{i,j} \cdot \bar{I}_{i,j})}{\sum_{i=1}^N \sum_{j=1}^M (I_{i,j})^2}. \quad (2.9)$$

**10. Total Edge Difference (TED):** Calculates the total sum of absolute value of edge difference between the original image (I) pixel values and the enhanced image ( $\bar{I}$ ) pixel values, averaged by total number of pixels in the image. Total edge difference formula (2.10) [12].

$$\text{TED} (I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M |I_{Ei,j} - \bar{I}_{Ei,j}|. \quad (2.10)$$

Where  $I_{Ei,j}$  is total number of corners in the original image (**I**),  $\bar{I}_{Ei,j}$  is total number of corners in the enhanced image ( $\bar{I}$ ).

**11. Total Corner Difference (TCD):** Calculates the total sum of absolute value of corner difference between the original image (I) pixel values and the enhanced image ( $\bar{I}$ ) pixel values, and then averaged by the maximum number of corners in the image.

The total corner difference formula (2.11) [12]

$$\text{TCD} (I, \bar{I}) = \frac{|N_{cr} - \bar{N}_{cr}|}{\text{Max} (N_{cr}, \bar{N}_{cr})}. \quad (2.11)$$

Where NCR is total number of corners in the original image (**I**),  $\bar{N}_{cr}$  is total number of corners in the enhanced image ( $\bar{I}$ ).

**12. Spectral Phase Error (SPE):** Calculates the sum of phase error between the original image (I) and the enhanced image ( $\bar{I}$ ) squared in the fourier transform and divided by the total number of pixels in the image. Fourier transform transforms original/enhanced image to the fourier domain, where each point represents a particular frequency contained in the image [13]. The spectral phase error formula (2.12) [14].

$$\text{SPE}(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M |\text{Arg}(F_{i,j}) - \arg(\bar{F}_{i,j})|^2. \quad (2.12)$$

Where  $\arg(f_{i,j})$  is the phase of the original image, and  $\arg(\bar{F}_{i,j})$  is the phase of the enhanced image.

**13. Spectral Magnitude Error (SME):** Calculates the total sum of absolute value of the difference between the original image (I) and the enhanced image ( $\bar{I}$ ) in the fourier transform averaged by the total number of pixels in the image. The spectral magnitude error formula (2.13) [14].

$$\text{SME}(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (|F_{i,j}| - |\bar{F}_{i,j}|)^2. \quad (2.13)$$

Where  $(|f_i, j| - |\bar{f}_i, j|)$  is the absolute value of difference between two vectors of the original image (I) and enhanced image( $\bar{I}$ ) in the fourier transform.

**14. Gradient Phase Error (GPE):** Calculates the sum of phase error between the original image (I) subtracted from the original image ( $\bar{I}$ ) squared in the gradient transform and divided by the total number of pixels in the image. Gradient phase error formula (2.14) [15].

$$\text{GPE}(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M |\text{Arg}(G_{i,j}) - \text{Arg}(\bar{G}_{i,j})|^2. \quad (2.14)$$

Where  $arg(g_{I,j})$  is the gradient phase of the original image (**i**),  $arg(\bar{G}_{I,j})$  is the gradient phase of the enhanced image (**I**).

**15. Gradient Magnitude Error (GME):** Calculates the total sum of absolute value of the difference between the gradient of the original image (I) and the gradient of the enhanced image (**I**) averaged by the total number of pixels in the image. Gradient magnitude error formula (2.15) [15]:

$$GME(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (|G_{i,j}| - |\bar{G}_{i,j}|)^2. \quad (2.15)$$

Where **gi, j** is the gradient of the original image, and '**gi, j**' is the gradient of the enhanced image.

**16. Mean Angle Similarity (MAS):** The mean angle that calculates the similarity between the original image (I) and the enhanced image (I). The mean angle similarity formula (2.16) [7]:

$$MAS(I, \bar{I}) = 1 - \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (\alpha_{i,j}). \quad (2.16)$$

Where  $\alpha_{i,j}$  the angle of the original/enhanced image.

**17. Mean Angle Magnitude Similarity (MAMS):** The mean angle calculates the magnitude similarity between the original image (I) and the enhanced image (I). The mean angle magnitude similarity formula (2.17) [7]:

$$MAMS(I, \bar{I}) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (1 - [1 - \alpha_{i,j}][1 - \frac{||I_{i,j} - \bar{I}_{i,j}||}{255}]). \quad (2.17)$$

**18. Laplacian Mean Squared Error (LMSE):** Calculates the sum of ratio between the original image(I) and the enhanced image (I) to the original image squared,  $h(ii, j) = ii + 1, j + ii - 1, j + ii, j + 1 + ii, j - 1 - 4ii$ . The laplacian mean squared error formula (2.18) [11]:

$$\text{LMSE} (I, \bar{I}) = \frac{\sum_{i=1}^{N-1} \sum_{j=2}^{M-1} (h(I_{i,j}) - h(\bar{I}_{i,j}))^2}{\sum_{i=1}^{N-1} \sum_{j=2}^{M-1} h(I_{i,j})^2} . \quad (2.18)$$

**19. Structural Similarity Measures (SSIM):** It's a quality measure when one single image is contrasted and the other image is still with its original quality, SSIM is an upgrade widespread index.

(Available in [16] and implemented in [17]).

**20. Visual Information Fidelity (VIF):** Considers that the real images are on scenes defined as natural and according to this the type of properties should be the same.

(Available in [18] and implemented in [17]).

**21. Reduced Reference Entropy Difference (RRED):** It's a quality measure that extracts some local information of a given sample by processing as a wavelet. Speculation of the sample can't be seen in samples in nature.

(Available in [19] and implemented in [17]).

**22. High Low Frequency Index (HLFI):** Concerned with sharpness and functions by the estimation of the power between the low and up frequency actions of a fourier spectrum.

(Available in [20] and implemented in [17]).

**23. Jpeg Quality Index (JQI):** Evaluation of the distorted image qualities by a known closed artificial initiated comparing algorithm at a decreased number of bit rate like jpeg.

(Available in [21] and implemented in [17]).

**24. Blind Image Quality Index Measures (BIQI):** The basic idea of this measure is that clear original images present some common properties if calculated accurately, divergence of the uniformity in the given statistics in nature is able to estimate the given image quality. BIQI is used for the train image in the past.  
(Available in [22] and implemented in [17]).

**25. Natural Image Quality Evaluator (NIQE):** This quality measure evaluates the blind image quality feature when extracting the features of statistics associated to many alterations producing quality information.  
(Available in [23] and implemented in [17]).

### **2.1.3 Classifiers Used in [6] and Our RFDS**

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are used in [6] to classify fingerprint images if they real or fake.

- **Linear Discriminant Analysis (LDA):** a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier or, more commonly, for dimensionality reduction before later classification [24].

- **Quadratic Discriminant Analysis (QDA):** QDA is closely related to linear Discriminant Analysis (LDA), where it is assumed that the measurements are normally distributed. Unlike LDA however, in QDA there is no assumption that the covariance (the measure of how much two or more random variables differ) of each of the classes is identical. To estimate the parameters required in quadratic discrimination more computation and data is required than in the case of linear discrimination [25].

▪ **Linear Support Vector Machine (LSVM)**: use 3 or 4 support vectors to obtain the optimum boundary between features. For example, we will use 3 support vectors s1, s2, and s3.

$$\hat{s}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \hat{s}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}, \text{ And } \hat{s}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}.$$

Find 3 parameters  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  based on the following 3 linear equations:

$$A_1 \hat{s}_1 \cdot \hat{s}_1 + A_2 \hat{s}_2 \cdot \hat{s}_1 + A_3 \hat{s}_3 \cdot \hat{s}_1 = -1 \text{ (-negative class)}$$

$$A_1 \hat{s}_1 \cdot \hat{s}_2 + A_2 \hat{s}_2 \cdot \hat{s}_2 + A_3 \hat{s}_3 \cdot \hat{s}_2 = -1 \text{ (-negative class)}$$

$$A_1 \hat{s}_1 \cdot \hat{s}_3 + A_2 \hat{s}_2 \cdot \hat{s}_3 + A_3 \hat{s}_3 \cdot \hat{s}_3 = +1 \text{ (+positive class)}$$

Substitute the values for  $\hat{s}_1$ ,  $\hat{s}_2$ , and  $\hat{s}_3$  in the above equations, after simplification of the equations above we get:

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

Simplifying the above 3 simultaneous equations we get:

$$\alpha_1 = \alpha_2 = -3.25 \text{ and } \alpha_3 = 3.5.$$

Now we will get the boundary using these three parameters. The boundary between two classes is called hyper plane( $\hat{w}$ ), it discriminates the positive class from the negative class.

$$\hat{w} = \sum_i \alpha_i \hat{s}_i .$$

$$\hat{W} = (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}.$$

The vectors are augmented with a bias; we can equate the entry in  $\hat{w}$  as the hyper plane with an offset  $b$ , therefore the separating hyper plane equation, x and y points on co-ordinate axis (x1, y1) (x2, y2).

$$Y = wX + b. \quad (2.19)$$

$W = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  and offset  $b = -3$ . This is the expected decision boundary for the linear svm.

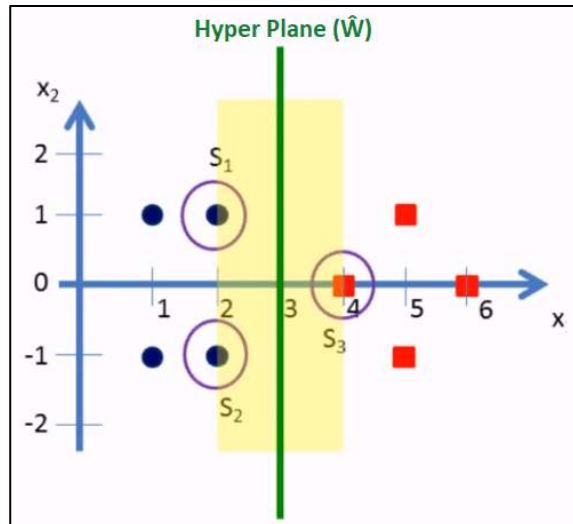


Figure 2: The boundary between two classes (Hyper Plane( $\hat{W}$ )).

▪ **Logistic Regression:** Logistic regression predicts whether something is **True** or **False**, instead of predicting something continuous like size. Logistic regression doesn't fit a line to the data, instead it fits an "s" shaped "logistic function", where the "s" curve goes from 0 to 1, which means the curve tells us the probability that a fingerprint is **Fake** or not based on its **IQM** result.

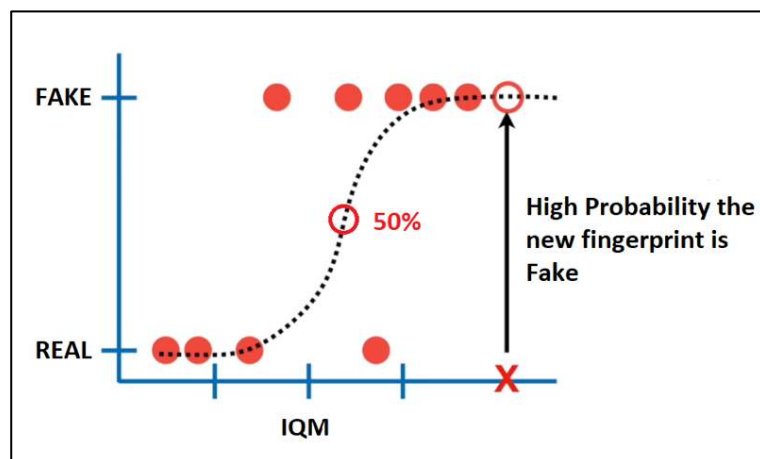


Figure 3: 'S' shaped logistic function.

If the probability a fingerprint is fake is above 50% then we classify it as fake, otherwise we will classify it as real.

▪ **Fine KNN (K-Nearest Neighbors):** Fine KNN is a simple way to classify data, if you already had a lot of data that classify the fingerprint, we could use fine KNN to decide which type of class the new fingerprint image(New Cell) belongs to, to be more clear here is an example:

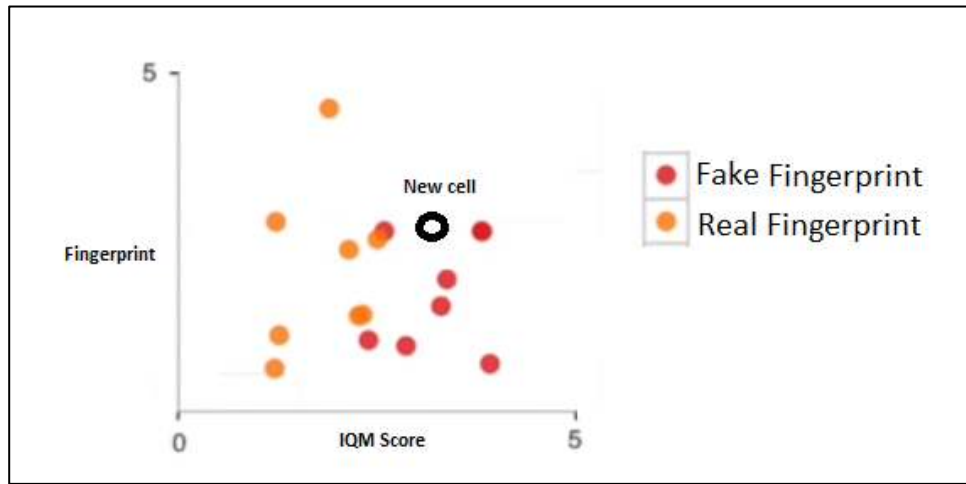


Figure 4: A new cell (fingerprint image) with unknown category to classify.

In figure 4, we added a new cell with unknown category, and we want to classify this new cell. We want to figure out what cell it's most similar to it, then we are going to call it that type of cell. In fine KNN, the new cell will be classified by looking at its nearest neighbors. If "k" in "K-Nearest Neighbors" is equal to 1, then we only use the nearest neighbor to define the category, and in our example the category is red for the new cell because the nearest neighbor is also red.

### 2.1.3.1 RFDS Experiments In [6]

Experiment was implemented on a system with 64-bit operating system, Windows 7 PC, 3.4 GHz processor, RAM of 16GB, and MATLAB r2012b. LivDet09 [26] was the

database used in the fingerprint detection experiment. LivDet09 [26] has above 18,000 samples obtained from more than 100 different fingers.

Three sensors used in [6] are Biometrika, Crossmatch, and Identix. The overall error rate using biometrika sensor was 12.8%, crossmatch 10.7 %, and identix 1.2%, all shown in Table 1.

Table 1: RFDS [6] results using three different type of fingerprint sensors.

|           | <b>Comparative Results: Fingerprints-LivDet09</b> |            |             |                   |            |             |                |            |             |
|-----------|---------------------------------------------------|------------|-------------|-------------------|------------|-------------|----------------|------------|-------------|
|           | <b>Biometrika</b>                                 |            |             | <b>Crossmatch</b> |            |             | <b>Identix</b> |            |             |
|           | <b>FFR</b>                                        | <b>FGR</b> | <b>HTER</b> | <b>FFR</b>        | <b>FGR</b> | <b>HTER</b> | <b>FFR</b>     | <b>FGR</b> | <b>HTER</b> |
| IQA-Based | 14.0                                              | 11.6       | 12.8        | 8.6               | 12.8       | 10.7        | 1.1            | 1.4        | 1.2         |

Description of the measures used in Table 1:

False Fake Rate (FFR) refers to the number of real images that were considered as fake in the system.

$$FFR = \frac{FFR}{TOTAL\ NUMBER\ OF\ REAL\ IMAGES} . \quad (2.20)$$

False genuine rate (FGR) refers to number of fake images that were considered as real in the system:

$$FGR = \frac{FGR}{TOTAL\ NUMBER\ OF\ FAKE\ IMAGES} . \quad (2.21)$$

Half Total Error (HTER) refers to the number of real images that were considered as fake in the system summed with the number of fake images that were considered as real in the system divided by 2,

$$HTER = \frac{FGR + FFR}{2} . \quad (2.22)$$

#### **2.1.4 Fingerprint Liveness Detection Based on Quality Measures [27]**

In [27], the RFDS uses image quality measures for a Software-Based solution in fingerprint image detection. The main purpose of [27] is to counter the direct attacks implemented against the fingerprint recognition systems.

Direct attacking methods present artificial fingerprints to the sensors so that the sensor will identify them as real fingerprints inserted to the system, and thus access is granted. To prevent direct attacking occurrence, a new live detection system was proposed in this paper with 3 types of image quality measures, ridge strength, ridge continuity, and ridge clarity. The new live detection system proposed in [27] includes the steps that are explained below:

**Input:** Once the fingerprint image gets inserted into the system, it will be immediately segmented (Image is Easier to Analyze) in the background.

**Feature Extraction:** Image quality measures will be extracted, the image quality measures with the best performance only will be selected from the all the available ones. After selecting the best image quality measures, we reach the final stage, classification of the fingerprint image. Linear discriminant analysis (LDA) (described in subsection 2.1.1) is used in [27] to classify whether the inserted fingerprint image is Real or Fake.

##### **2.1.4.1 RFDS Experiments In [27]**

Database used in [27] is provided in the fingerprint liveness detection competition, livdet 2009, containing over 18,000 real and fake samples of fingerprint images generated with different materials and captured with different sensors. Three sensors were used in [27]: biometrika, crossmatch, and identix.

The proposed new liveness detection system result was on average (7.58% hter) in identifying real or fake sample of fingerprint images. The results of [27] for each sensor are shown in Table 2.

Table 2: The results for each sensor [27].

|            | <b>Fgr(%)</b> | <b>Ffr(%)</b> | <b>Hter(%)</b> |
|------------|---------------|---------------|----------------|
| Bionetrika | 2.12          | 1.54          | 1.83           |
| Crossmatch | 12.48         | 12.32         | 12.40          |
| Identix    | 6.40          | 10.67         | 8.53           |
| Average    | 7.00          | 8.17          | 7.58           |

#### **2.1.5 A High-Performance Fingerprint Liveness Detection Method Based on Quality Related Features [28]**

A software-based fingerprint liveness detection approach using image quality measures is proposed in [28]. The system was tested by using two different databases and five different sensors.

The proposed system starts with inserting a fingerprint image and, the type of sensor used for getting the fingerprint image. After inserting the sensor type and the fingerprint image, segmentation will be applied to the fingerprint image, and then three different types of image quality measures will be extracted from the fingerprint image: ridge strength, ridge continuity, and ridge clarity.

After applying extraction to the fingerprint image and obtaining the results of every single image quality measurement, selection of the ones with the best results from the image quality measures will be done, and the rest will be excluded for the classification

of the image stage. Classification is done using LDA (defined in 2.1.1) to classify if the fingerprint image is real or fake.

### 2.1.5.1 RFDS Experiments In [28]

In [28], two databases were used: LivDet DB and ATVS DB. ATVS database contains three different datasets of real and fake fingerprints captured each by three different sensors: Biometrika, Precise, and thermal Yubee. LivDet database contains also three different datasets of real and fake fingerprints captured each captured by three different sensors, Namely, Biometrika, Crossmatch, and Identix.

Experiments results in [28] with both ATVS DB and LivDet show overall around 10% of error in identifying real and fake fingerprint images in the liveness detection system. The results for both databases with five different sensors are listed in Table 3,  $FGR_1$ ,  $FFR_1$ , and  $HTER_1$  REFER to the ATVS db, and  $FGR_2$ ,  $FFR_2$ , and  $HTER_2$  refer to the LivDet.

Table 3: Performance result of the two databases used in [28].

|                    | Performance Results In % |               |                 |      |
|--------------------|--------------------------|---------------|-----------------|------|
|                    | $FGR_1/FGR_2$            | $FFR_1/FFR_2$ | $HTER_1/HTER_2$ | Hter |
| <b>Biomet.Ld</b>   | 3.1/9.8                  | 71.8/21.5     | 37.4/15.6       | 26.5 |
| <b>Crossmatch</b>  | 8.8/16.7                 | 20.8/6.8      | 14.8/11.7       | 13.2 |
| <b>Identix</b>     | 4.8/8.0                  | 5.0/9.1       | 4.9/8.5         | 6.7  |
| <b>Biomet.Atvs</b> | 9.4/0.4                  | 1.5/11.8      | 5.5/6.1         | 5.8  |
| <b>Precise</b>     | 3.1/0.4                  | 13.7/0.4      | 8.4/0.4         | 4.4  |
| <b>Yubee</b>       | 2.7/1.6                  | 6.3/13.0      | 4.5/7.3         | 5.9  |
| <b>Average</b>     | 5.3/6.1                  | 19.8/10.4     | 12.5/8.2        | 10.4 |

### **2.1.6 Fingerprint Identification in Biometric Security Systems. International Journal of Computer and Electrical Engineering [29]**

In [29], two software based matching algorithms (algorithms that match fingerprints against a database of known and unknown fingerprint) are discussed. The first algorithm is developed at the Hong Kong Baptist University; the second matching algorithm is developed at the michigan state university in usa. Two of the algorithms were downloaded and tested with a common database.

The Hong Kong Baptist University fingerprint identification system uses a traditional minutia matching algorithm, while the Michigan State University uses filter-based algorithm in their system.

The Hong Kong baptist university fingerprint identification system is based on three steps:

**1. Preprocessing:** The inserted fingerprint image is enhanced using fourier transform (conversion of the image from space domain to frequency domain) and histogram equalization (modify the intensity and contrast of the fingerprint image), after that binarization is applied to the image which will change it from pixel image to a binary image using threshold for each block in the image.

**2. Minutiae Extraction:** The extraction is done by thinning the inserted fingerprint image and marking any minutiae that exists in the image.

**3. Post-Processing:** In this stage, all false minutiae (major features of a fingerprint such as ridge ending and bifurcation) are removed.

Michigan State University method uses filter-bank based matcher: the proposed filter-bank based algorithm uses more than one filter to capture details in a fingerprint as a

compact fixed length feature vector. The fingerprint matching is based on the euclidean distance between the two corresponding vectors.

#### **2.1.6.1 RFDS Experiments In [29]**

The database used in [29] is NIST special database 4 which is publicly available; it contains 8-bit gray scale image fingerprints distributed for development and testing of fingerprint identification systems.

For 25 pairs of fingerprints with high quality inserted into the software using filter-based algorithm, the results are as follows: 8% FAR and 4% FFR.

In [29], they concluded that both of the algorithms are equal in performance and none of them can be considered as better than the other in general sense.

#### **2.1.7 Fingerprint Spoof Detection Using Quality Features [30]**

In [30], three stages are considered for fake fingerprint detection:

**1. Feature Extraction:** The inserted fingerprint image is divided into blocks and the features are extracted from each block. There is a total of 7 features which will be get extracted from the input fingerprints image.

**2. Features Selection:** The features with optimal performance will be selected; the ones that are not effective, will be excluded.

**3. Classification:** After feature extraction and selecting the best extracted features, classification of the fingerprint image is implemented to identify if the inserted fingerprint image is real or fake.

##### **2.1.7.1 RFDS Experiments In [30]**

The database used in [30] is provided in the fingerprint liveness detection competition LivDet 2009 [26] with three data sets of real and fake fingerprint images all captured with different kind of optical sensors: Biometrika, Identix, and Crossmatch.

Experiment results in [30] are considered overall as a good RFDS to use, the results in Figure 5 show that if all features are used at once without excluding any one of them, better results will be obtained.

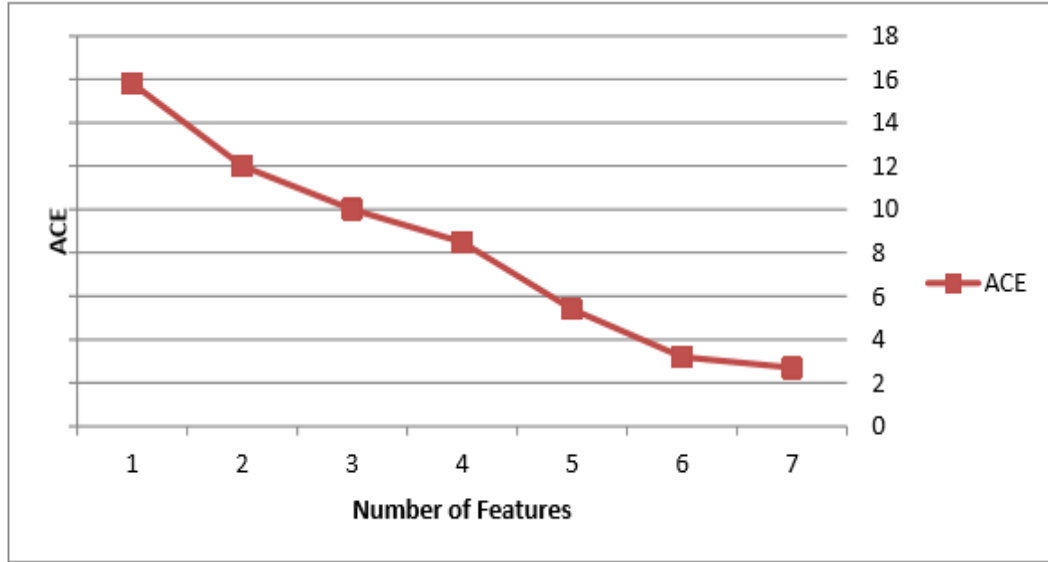


Figure 5: Average classification error (ACE) results in [30].

## 2.2 Problem Definition

We have observed [6], [18], [19], [20], and [30], but we will mainly compare our RFDS to be developed with RFDS [6]. The list of problems considered in this thesis is:

1. Implement and test a real fingerprint detection system (RFDS) similar to [6].
2. Repeat experiments on RFDS implemented as in [6].
3. Compare our RFDS with [6].
4. Find the best 5, 10, and 15 iqms in our RFDS.
5. Modifying thesis [31] the distinction from our RFDS and thesis [31] is that we used 10 more IQMs and 6 classifiers. Thesis [31] was implemented for face detection while ours is for fingerprints.

## 2.3 Summary

In this chapter, we have done a literature review from the reading and analysis of [6], [27], [28], [29], and [30]. We conclude that some of the presented RFDS use similar image quality features and classification algorithms and some use different. The experiment results on these systems were gathered using different databases, all these results obtained still didn't reach 100% accuracy of detecting fingerprint spoof attacks. The features to be built in our RFDS is to find the best 5, 10 and 15 IQM's automatically calculating the FFR, FAR, and HTER of the RFDS. We define our thesis problem as: implementation and testing of real fingerprint image detection system (RFDS).

## Chapter 3

### DESIGN AND IMPLEMENTATION OF RFDS

In this chapter, we show the design of our RFDS, implementation of the feature extraction in our system as well as the classification process. We implemented the Gaussian filter as explained in subsection 3.2.1, feature extraction stage for the fingerprint images as explained in subsection 3.2.2 using 25 image quality measures. The training structure was implemented and explained in subsection 3.2.3, classification process in subsection 3.2.4, method of defining the best IQMs in subsection 3.2.5, and the method of defining the best IQMs was implemented and tested as described in subsection 3.2.

#### 3.1 Design of RFDS

Figure 6 shows the design of our RFDS. Basically, our system contains two structures:

1. Training Structure (as shown at the top of figure 6)
2. Detection Structure (as shown at the top of figure 6).

The top of Figure 6 shows our RFDS training structure where (I) refers to the original fingerprint image and (I') refers to the enhanced image. Images will be input in our RFDS training structure. In block 1, each single image will be filtered using gaussian filter (see subsection 3.2.1) with 3\*3 kernel to reduce the noise of the image. By filtering the input images (I), we get the enhanced fingerprint image (I'). In block 2, we will apply feature extraction process to both of the fingerprint images by extracting image quality measures (see subsection 3.2.2), we then create a table and list the results

of the fingerprint images and their IQM. In block 3, we import the table with all the results to classification learner application, which is a user-friendly application that we use in MATLAB software to create our training model (block number 4) and classify images (see subsection 3.2.3). From this we move to the classification stage where all the fingerprint images will be classified as either real or fake images.

The bottom of Figure 6 shows our RFDS detection structure. All input images will be classified on real or fake depending on the trained model created in the “training structure” shown in figure 6.

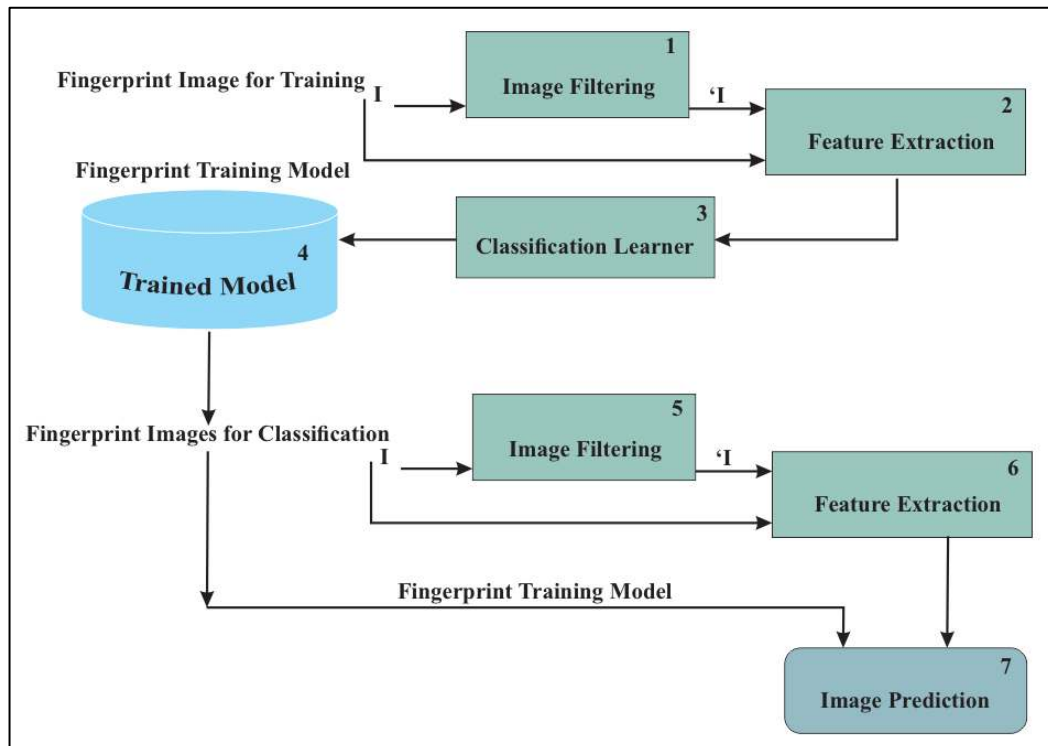


Figure 6: RFDS training and detection structure.

## 3.2 Implementation of RFDS

### 3.2.1 Implementation and Testing of Gaussian Filter

Gaussian filter is a filter whose impulse response is a Gaussian function (or an approximation to it). Gaussian filters have the properties of no overshoot to a step

function input while minimizing the rise and fall time [32]. For our RFDS implementation and testing, we used MATLAB 2015 in this thesis. For implementation of Gaussian filter, we used a ready MATLAB function “Gaussian” to enhance and reduce the noise of the image. All we had to do was to call the function by typing “Gaussian” which is shown below in Figure 7, and highlighted with number 1. Then, we set the value of the Gaussian kernel as “3x3” highlighted and numbered 2 in the screenshot, and set the standard deviation as “0.6” (0.6 was selected just as an example) highlighted and numbered 3 in Figure 7. By these three values, we set our RFDS system that gives us a Gaussian filter with the values shown in number 4 of the screenshot below.

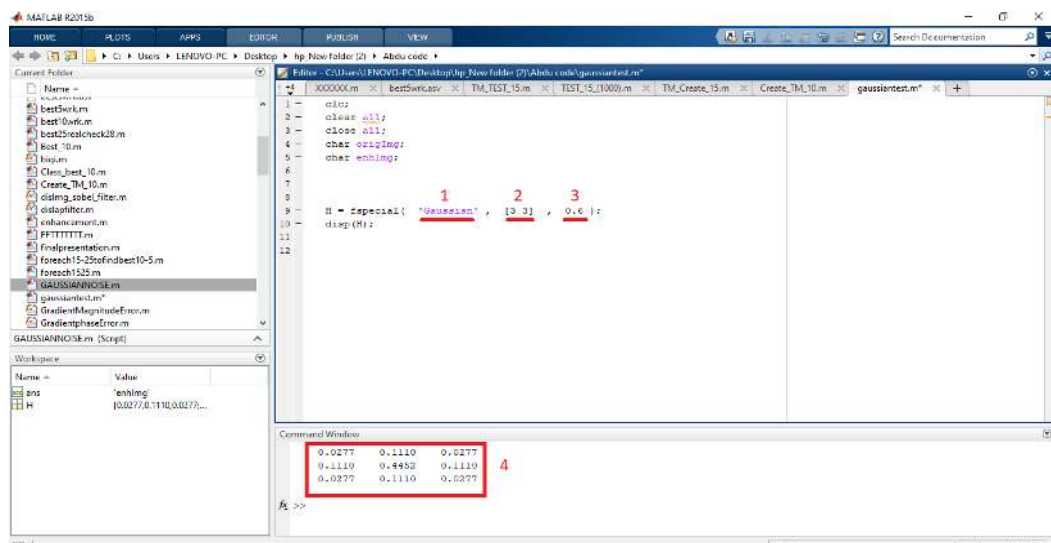


Figure 7: Gaussian filter code and the Gaussian filter obtained from it.

Example of how Gaussian filter is implemented using a 4\*4 matrix image:

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

| 3x3 Gaussian Kernel |        |        |
|---------------------|--------|--------|
| 0.0277              | 0.1110 | 0.0277 |
| 0.1110              | 0.4452 | 0.1110 |
| 0.277               | 0.1110 | 0.0277 |

Let's select the pixel in 2nd row 3rd column of the original image and apply Gaussian filter to it:

|   |   |   |    |
|---|---|---|----|
| 0 | 1 | 7 | 3  |
| 4 | 2 | 8 | 10 |
| 0 | 2 | 1 | 6  |
| 6 | 3 | 5 | 4  |

 $*$ 

|        |        |        |
|--------|--------|--------|
| 0.0277 | 0.111  | 0.0277 |
| 0.1110 | 0.4452 | 0.111  |
| 0.0277 | 0.111  | 0.0277 |

 $=$ 

|        |        |        |
|--------|--------|--------|
| 0.0277 | 0.777  | 0.0831 |
| 0.222  | 3.5616 | 1.11   |
| 0.0554 | 0.111  | 0.1662 |

Add the values in the vector:  $0.0277 + 0.777 + 0.0831 + 0.222 + 3.5616 + 1.11 + 0.0554 + 0.111 + 0.1662 = 6.114$

Final result of the original image after Gaussian filter as.

|       |       |       |       |
|-------|-------|-------|-------|
| 0.610 | 1.776 | 4.780 | 3.444 |
| 2.085 | 2.777 | 6.114 | 6.560 |
| 1.470 | 2.193 | 3.302 | 4.696 |
| 3.059 | 2.806 | 3.335 | 3.029 |

To check if our manual calculation of the Gaussian filter applied to the original image is correct or not, we ran our MATLAB code and took a screenshot of the result shown below in figure 8, the result of the original image with Gaussian filter applied to it is the same as (3.1).

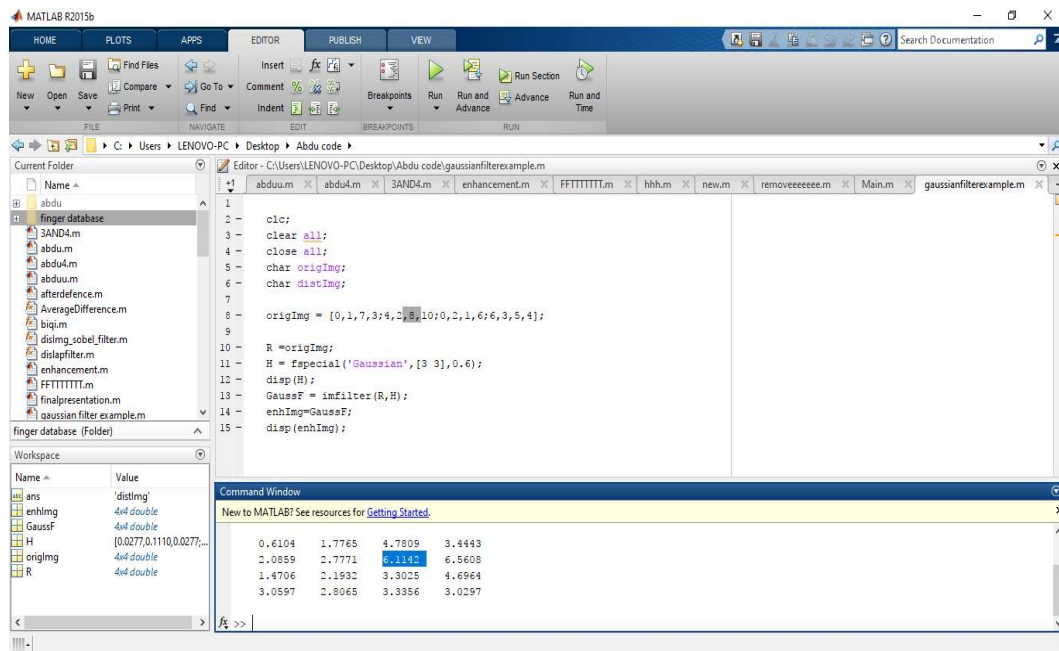


Figure 8: Result of the original image with Gaussian filter applied to it.

### 3.2.2 Implementation and Testing of Feature Extraction

To test the implementation of our RFDS, we use 4x4-sized (for ease of calculation) two gray scale images, one named original image (I) with matrix elements: (0, 1, 7, 3),

(4, 2, 8, 10), (0, 2, 1, 6), (6, 3, 5, 4) and the second one named enhanced image ('I') (original image with Gaussian filter applied to it) with matrix elements: (2, 5, 10, 4), (0, 1, 5, 1), (3, 6, 2, 3), (1, 3, 10, 0).

| Original Image (I) |   |   |    |
|--------------------|---|---|----|
| 0                  | 1 | 7 | 3  |
| 4                  | 2 | 8 | 10 |
| 0                  | 2 | 1 | 6  |
| 6                  | 3 | 5 | 4  |

| Enhanced Image ('I') |   |    |   |
|----------------------|---|----|---|
| 2                    | 5 | 10 | 4 |
| 0                    | 1 | 5  | 1 |
| 3                    | 6 | 2  | 3 |
| 1                    | 3 | 10 | 0 |

For each of the 25 IQMs we will do the following (our MATLAB code is provided in Appendix A and the IQM function in Appendix B).

For 4x4-sized of the enhanced and original image, we will take screenshots of every IQM result in MATLAB.

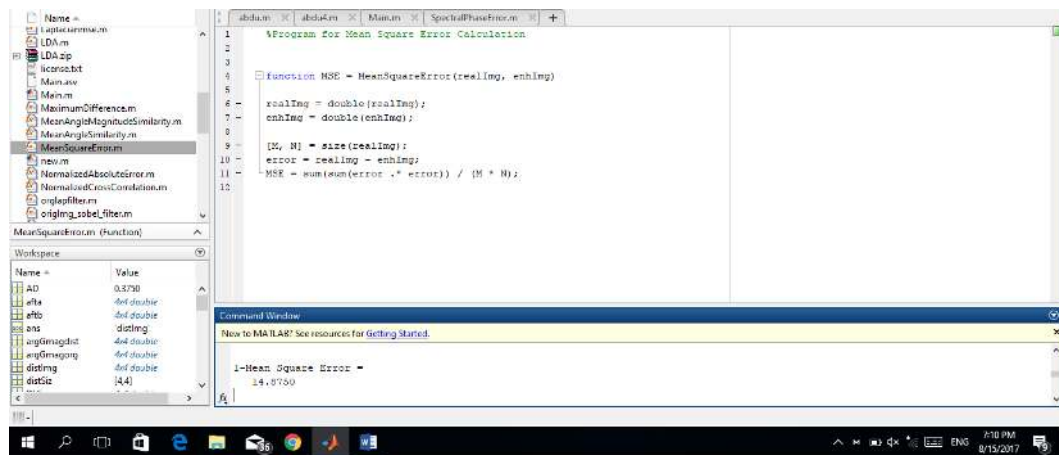
### 3.2.2.1 Implementation and Testing of MSE (2.1)

Explanation of MSE code given in (Appendix A-1):

Line 1: MSE function has two inputs: `rel_img` refers to the real image, and `enhcd_img` to the enhanced image. Line 4: `[m n]` is defined where `m` is the number of rows and `n` is the number of columns the real image has. Line 5: difference between real and enhanced images is calculated. Line 6: using (2.1), MSE is calculated.

Testing example of MSE calculation for I and 'I':

$$\text{MSE} = 1/16 * ( (0-2)^2 + (1-5)^2 + (7-10)^2 + (3-4)^2 + (4-0)^2 + (2-1)^2 + (8-5)^2 + (10-1)^2 + (0-3)^2 + (2-6)^2 + (1-2)^2 + (6-3)^2 + (6-1)^2 + (3-3)^2 + (5-10)^2 + (4-0)^2 ) = 1/16*(238) = 14.875 \quad (3.2)$$



### 3.2.2.2 Implementation and Testing of PSNR (2.2)

The result of PSNR calculation shown in Figure 10 is equal to our manually calculated result (3.3).

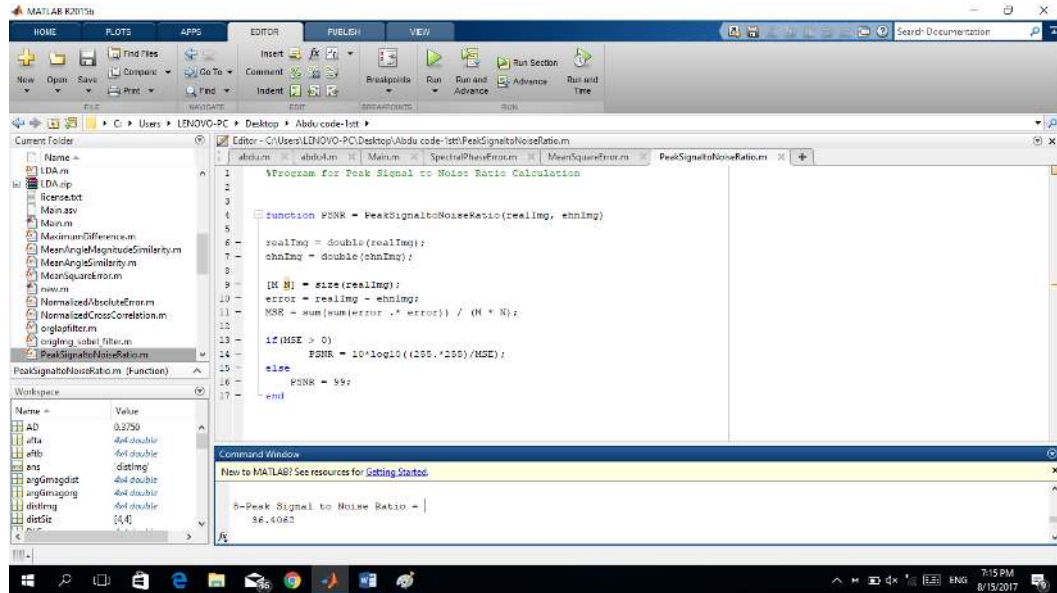


Figure 10: MATLAB result of PSNR calculation.

### 3.2.2.3 Implementation and Testing of SNR (2.3)

Explanation of the SNR code (Appendix A-3):

Line 1: SNR function with two inputs: rel\_img and enhcd\_img refer to the real and enhanced image, respectively. Line 4: [m n] is defined, where m and n are the number of rows and columns of the real image. Line 5: difference between real image and enhanced image is calculated. Line 6: using (2.1), MSE is calculated., Line 7: using (2.3), SNR (2.3) is calculated. Line 8: we set SNR to 99 if MSE is zero.

**MSE = 14.875** (obtained from Figure 9)

SNR Calculation:

$$\text{SNR} = 10 \log_{10} (0^2 + 1^2 + 7^2 + 3^2 + 4^2 + 2^2 + 8^2 + 10^2 + 0^2 + 2^2 + 1^2 + 6^2 + 6^2 + 3^2 + 5^2 + 4^2) / 4 * 4 * 14.875$$

$$\text{SNR} = 10 \log_{10} 370/238 = 10 \log_{10} 1.55 = 1.91 \quad (3.4)$$

The result of SNR calculation shown in Figure 11, is equal to our manually calculated result (3.4).

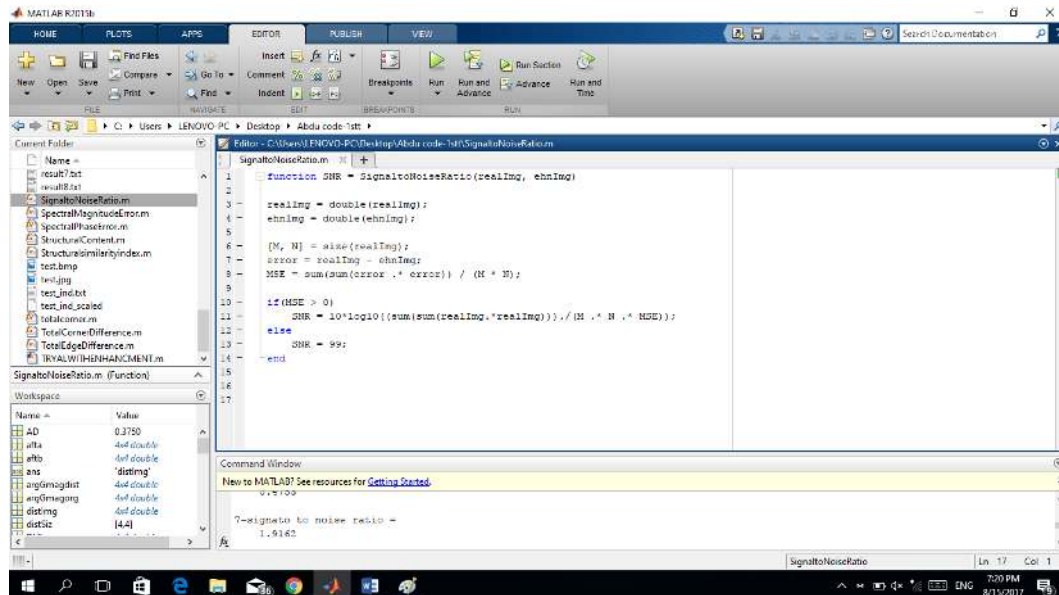


Figure 11: MATLAB result of SNR calculation.

#### 3.2.2.4 Implementation and Testing of SC (2.4)

Explanation of the SC code (Appendix A-4):

Line 1: SC function with two inputs `rel_img` and `enhcd_img` refers to the real and enhanced image. Line 4: using (2.4), SC is calculated.

$$\begin{aligned} \mathbf{SC} = & (0^2 + 1^2 + 7^2 + 3^2 + 4^2 + 2^2 + 8^2 + 10^2 + 0^2 + 2^2 + 1^2 + 6^2 + 6^2 \\ & + 3^2 + 5^2 + 4^2) / (2^2 + 5^2 + 10^2 + 4^2 + 0^2 + 1^2 + 5^2 + 1^2 + 3^2 + 6^2 \\ & + 2^2 + 3^2 + 1^2 + 3^2 + 10^2 + 0^2) = 370/340 = 1.0882. \end{aligned} \quad (3.5)$$

The result of SC calculation shown in Figure 12, is equal to our manually calculated result (3.5).

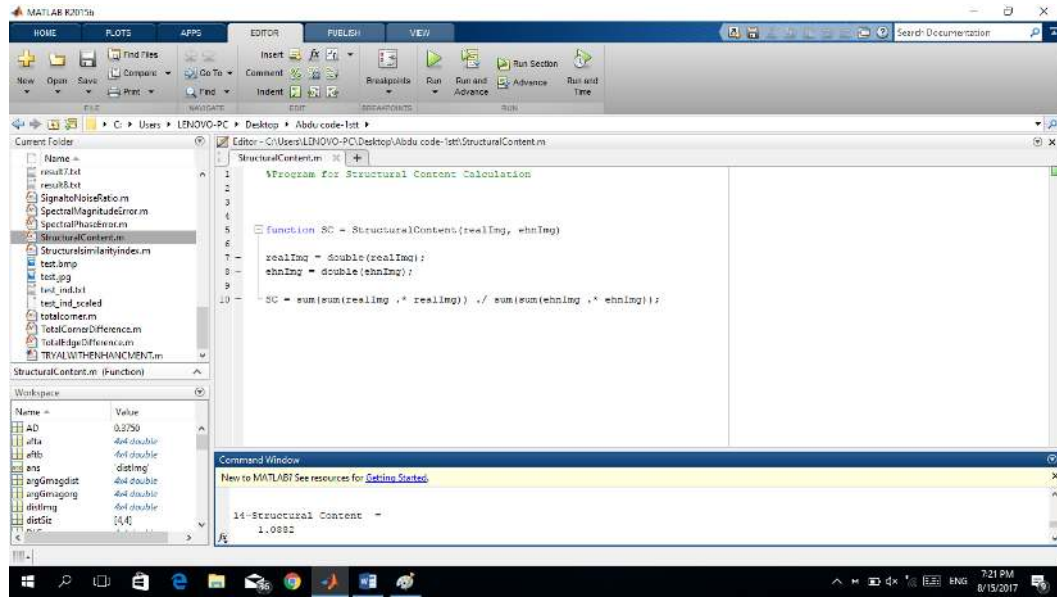


Figure 12: MATLAB result of SC calculation.

### 3.2.2.5 Implementation and Testing of MD (2.5)

Explanation of MD code (Appendix A-5):

Line 1: MD function with two inputs: `rel_img` and `ehcd_img` refer to the real and enhanced image. Line 4: the difference between the real image and the enhanced image will be calculated. Line 5: using (2.5), MD will be calculated.

$$\begin{aligned}
 P1 &= 0 - 2 = -2 & P9 &= 0 - 3 = -3 \\
 P2 &= 1 - 5 = -4 & P10 &= 2 - 6 = -4 \\
 P3 &= 7 - 10 = -3 & P11 &= 1 - 2 = -1 \\
 P4 &= 3 - 4 = -1 & P12 &= 6 - 3 = 3 \\
 P5 &= 4 - 0 = 4 & P13 &= 6 - 1 = 5 \\
 P6 &= 2 - 1 = 1 & P14 &= 3 - 3 = 0 \\
 P7 &= 8 - 5 = 3 & P15 &= 5 - 10 = -5 \\
 P8 &= 10 - 1 = 9 & P16 &= 4 - 0 = 4
 \end{aligned}
 \tag{3.6}$$

$$Md = 9$$

The result of MD calculation shown in Figure 13 is equal to our manually calculated result (3.6).

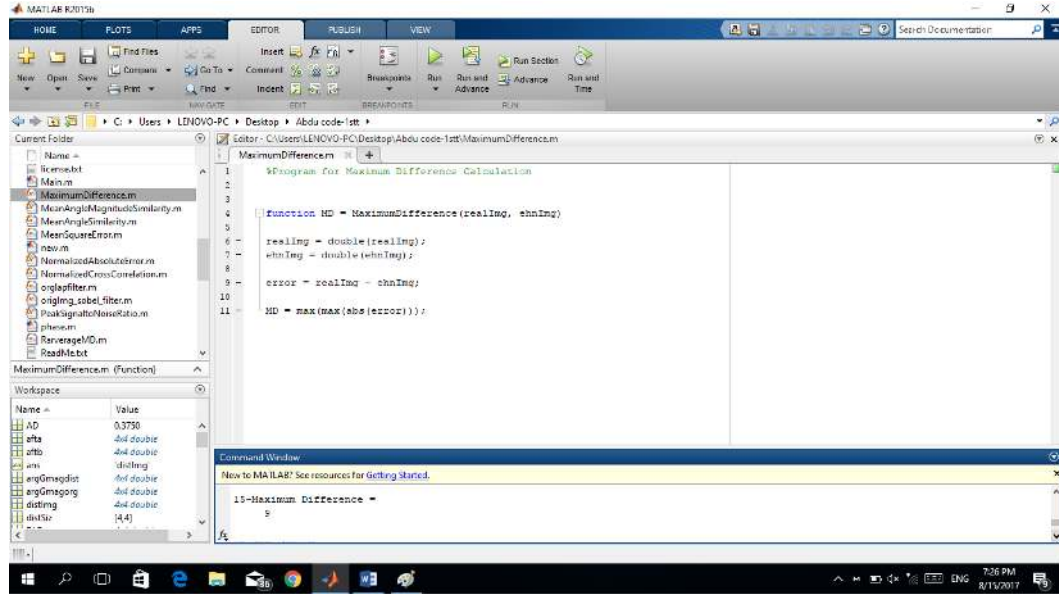


Figure 13: MATLAB result of MD calculation.

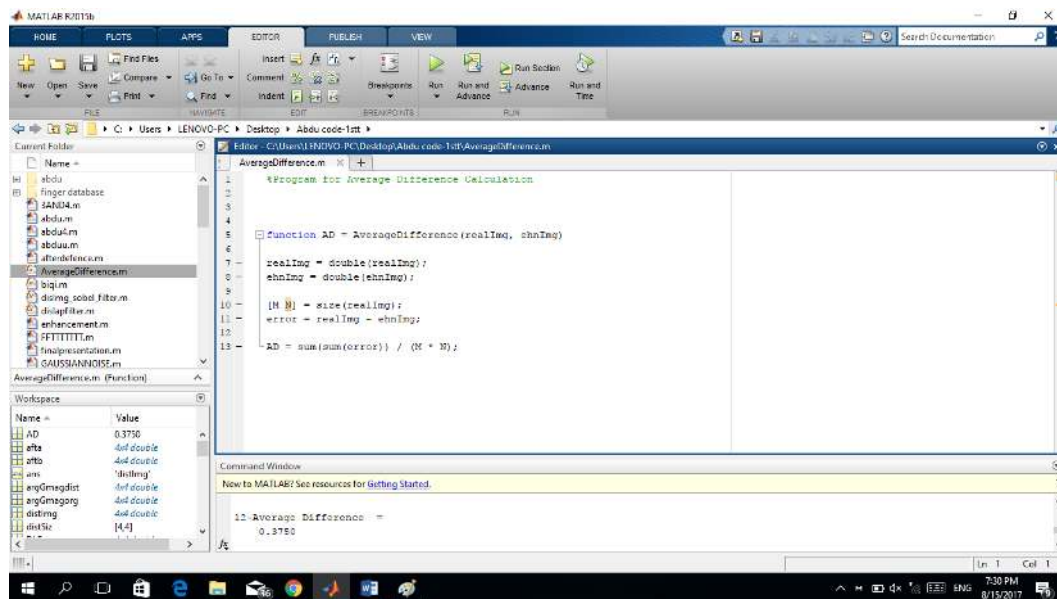
### 3.2.2.6 Implementation and Testing of AD (2.6)

Explanation of the AD code (Appendix A-6):

Line 1: AD function with two inputs: `rel_img` and `enhcd_img` refer to the real and enhanced image, line 4: `[m n]` is defined, where `m` and `n` is the number of rows and columns of the real image. 5: difference between real image and enhanced image will be calculated. Line 6: using (2.6), `ad` will be calculated.

$$\begin{aligned}
 \mathbf{AD} &= 1/16((0-2) + (1-5) + (7-10) + (3-4) + (4-0) + (2-1) + (8-5) + (10-1) + (0-3) + (2-6) \\
 &+ (1-2) + (6-3) + (6-1) + (3-3) + (5-10) + (4-0)) = 1/16(-2-4-3-1+4+1+3+9-3-4- \\
 &1+3+5+0-5+4) = 6/16 = 0.375
 \end{aligned} \tag{3.7}$$

The result of `ad` calculation shown in Figure 14 is equal to our manually calculated result (3.7).



### 3.2.2.7 Implementation and Testing of NAE (2.7)

Line 1: NAE function with two inputs: `rel_img` and `enhcd_img` refer to the real and enhanced image. Line 4: the difference between the real and enhanced image will be calculated. Line 5: using (2.7), NAE will be calculated.

$$\mathbf{Nae} = [2+4+3+1+4+1+3+9+3+4+1+3+5+0+5+4] = 52/62 = 0.8387 \quad (3.8)$$

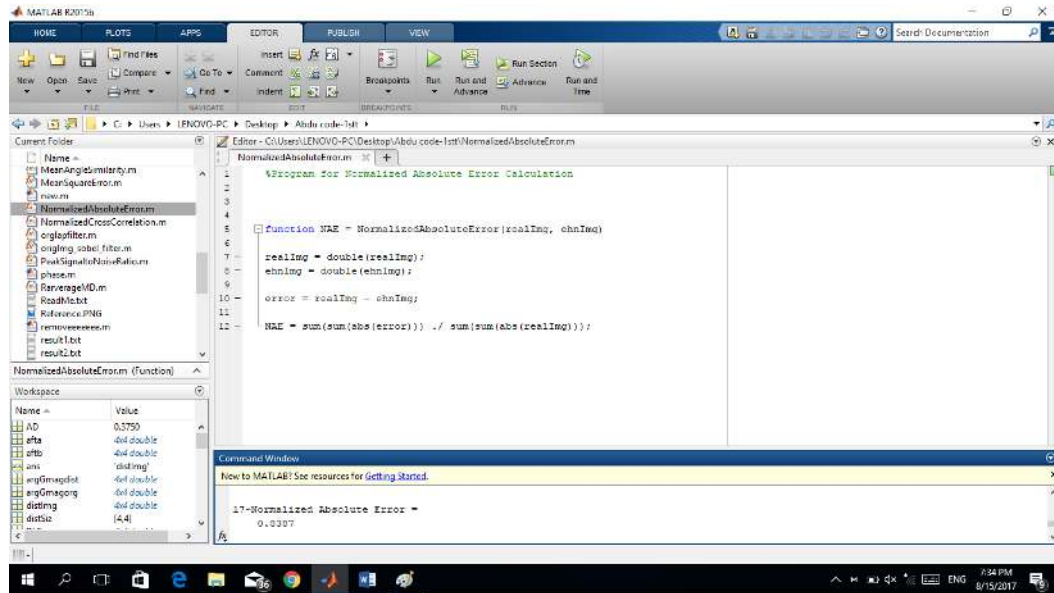


Figure 15: MATLAB result of NAE calculation.

### 3.2.2.8 Implementation and Testing of RAMD (2.8)

Explanation of the RAMD code (Appendix A-8):

Line 1: RAMD function with two inputs: `rel_img` and `ehncd_img` refer to the real and enhanced image. Line 4: the difference between the real and enhanced image is calculated. Line 5: the variable 'error' will have the absolute value of error. Line 6: we convert matrix to vector, then we order the numbers by descending order (using `flipud` a ready MATALAB function) without having repeated numbers, all numbers should be unique. Line 7: select top ten from the vector. Line 8: using (2.8), RAMD is calculated.

$R = 7$  (we select the maximum 7 differences between the image  $I$  and ' $I$  and divide over 7, in the main code  $r=10$ , but for the testing case we set  $R=7$  because if we set  $R=10$  the 8<sup>th</sup>, 9<sup>th</sup>, and 10<sup>th</sup> values will be equal to zero).

$$\begin{aligned}
P1 &= 0 - 2 = -2 & P9 &= 0 - 3 = -3 \\
P2 &= 1 - 5 = -4 & P10 &= 2 - 6 = -4 \\
P3 &= 7 - 10 = -3 & P11 &= 1 - 2 = -1 \\
P4 &= 3 - 4 = -1 & P12 &= 6 - 3 = 3 \\
P5 &= 4 - 0 = 4 & P13 &= 6 - 1 = 5 \\
P6 &= 2 - 1 = 1 & P14 &= 3 - 3 = 0 \\
P7 &= 8 - 5 = 3 & P15 &= 5 - 10 = -5 \\
P8 &= 10 - 1 = 9 & P16 &= 4 - 0 = 4 \\
\mathbf{RAMD} &= 1/7 | 9 + 5 + 4 + 3 + 2 + 1 | = 24/7 = 3.428
\end{aligned} \tag{3.9}$$

The result of RAMD calculation shown in Figure 16 is equal to our manually calculated result (3.9).

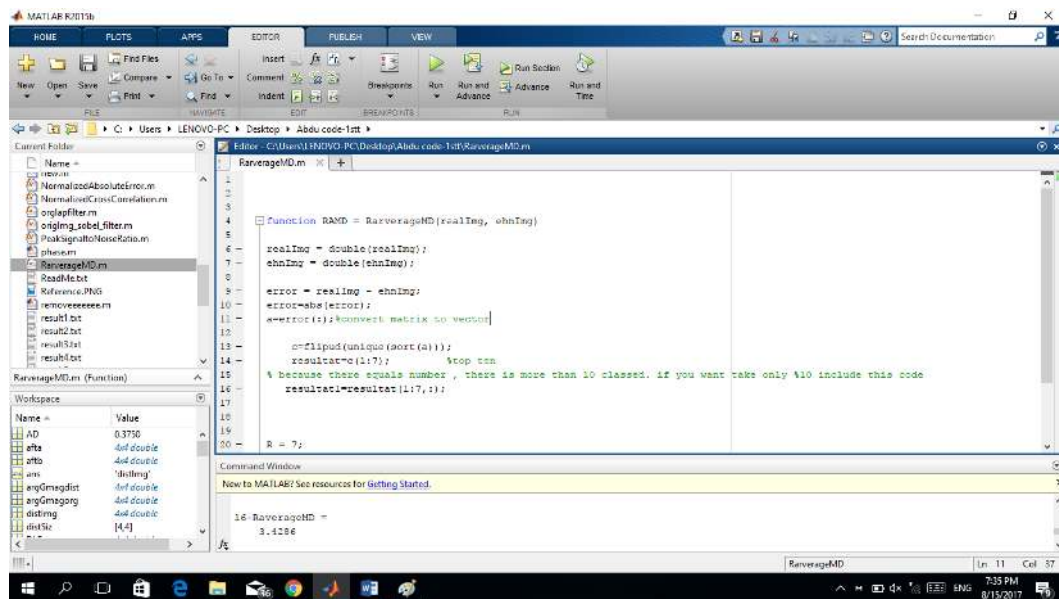


Figure 16: MATLAB result of RAMD calculation with R=7.

### 3.2.2.9 Implementation and Testing of NCC (2.9)

Explanation of NCC code (Appendix A-9):

Line 1: NCC function with two inputs: rel\_img and ehncd\_img refer to the real and enhanced image. Line 4: using (2.9) NCC will be calculated.

$$\begin{aligned}
NCC = & ( (0*2) + (1*5) + (7*10) + (3*4) + (4*0) + (2*1) + (8*5) + (10*1) + (0*3) + \\
& (2*6) + (1*2) + (6*3) + (6*1) + (3*3) + (5*10) + (4*0) / (0^2 + 1^2 + 7^2 + 3^2 + 4^2 \\
& + 2^2 + 8^2 + 10^2 + 0^2 + 2^2 + 1^2 + 6^2 + 6^2 + 3^2 + 5^2 + 4^2) = \\
NCC = & 236 / 370 = 0.637
\end{aligned} \tag{3.10}$$

The result of NCC calculation shown in Figure 17 is equal to our manually calculated result (3.10).

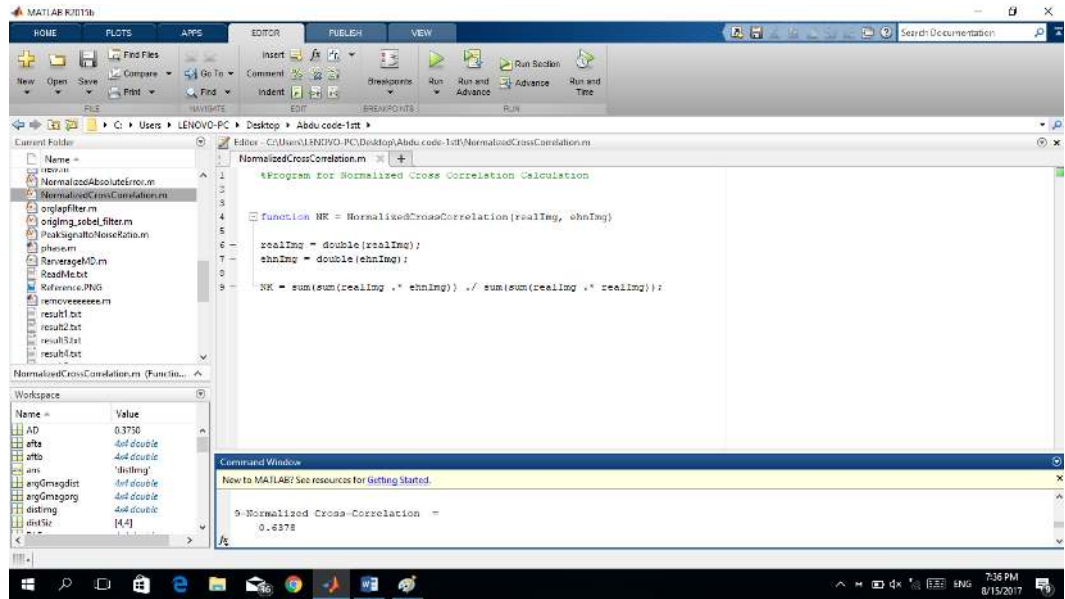


Figure 17: MATLAB result of NCC calculation.

### 3.2.2.10 Implementation and Testing of TED (2.10)

Explanation of TED code (Appendix A10):

Line 1: TED function with two inputs: `rel_img` and `enhcd_img` refer to the real and enhanced image. In Line 2, ready MATLAB function (`edge`) is used to detect the edges of the real image. In Line 3, all edges that were detected by the MATLAB function will be displayed. In Line 5 and 6 the same process is done as in lines 2 and 3 but this time for the enhanced image. Line 8: `[m n]` is defined, where `m` is the number of rows and `n` is the number of columns the real image has. Line 9: difference between real

image and enhanced image will be calculated and saved in 'eror' variable. Line 10: using the formula provided in (2.10) TED will be obtained.

We didn't apply manual calculation to test TED because in our code we use ready MATLAB function "edge (image)" to automatically find the image total edge difference of any image inserted to the RFDS, so we feed a fingerprint image to our system and took a screenshot of the result shown below in figure 18.

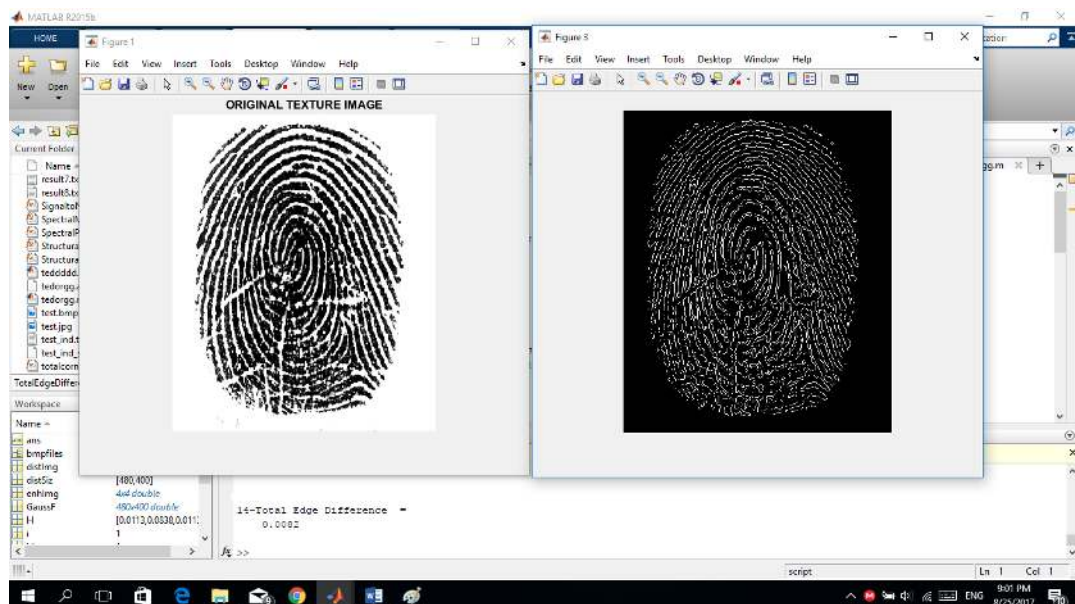


Figure 18: Matlab result for calculating TED of fingerprint image.

### 3.2.2.11 Implementation and Testing of TCD (2.11)

Explanation of the TCD code (each line number corresponds to its code in Appendix A-15):

Line 1: TCD function with two inputs `rel_img` which refers to the real image and `enhcd_img` refers to the enhanced image, line 4 we use a ready MATLAB function called "detectharrisfeatures" to detect corners of the real image, Line 5: we plot the corners in the real image using a MATLAB function called "plot", Line 6: we will get the sum of corners of the real image using ready MATLAB function called "length",

Line 7 we will detect corners of the enhanced image, Line 8: we plot the corners of the enhanced image same as we did to the real image, Line 9: we will get the sum of corners in of the enhanced image using the same function we applied to the real image, Line 10: we will set the value of variable “max1” either as the number of corners of the real or enhanced image depending on which one has th maximum number of corners detected, Line 11: using the formula provided in (2.11) TCD will be calculated.

**Ncr** = Number of Corners in The Original Image.

**‘Ncr** = Number of Corners in The Enhanced Image.

**Ncr**= 2496

**‘Ncr**= 2400

$$\mathbf{Tcd} = |2496-2400| / 2496 = 0.0385 \quad (3.11)$$

The result of TCD calculation shown in Figure 19 is equal to (3.11).

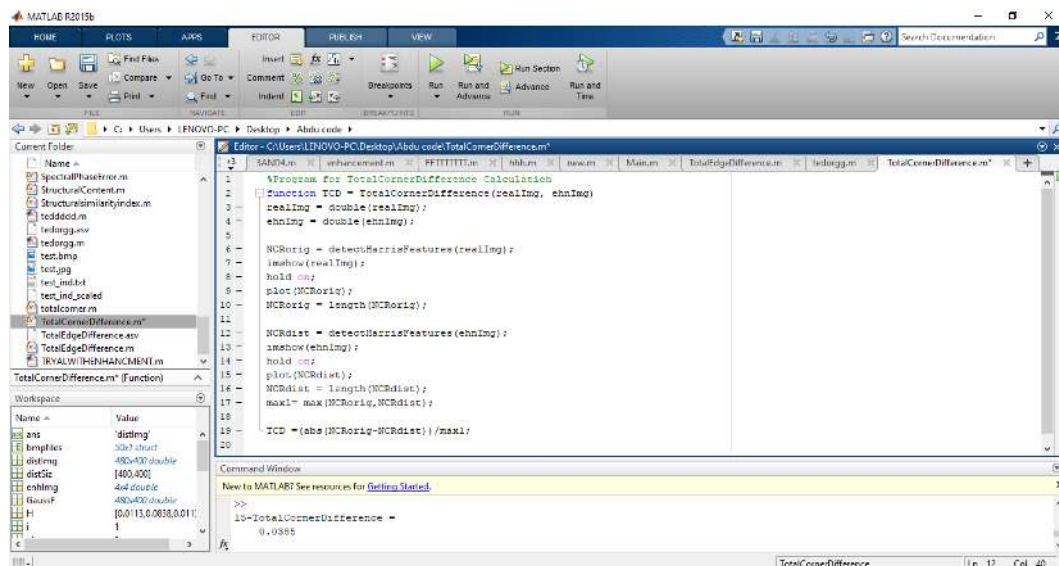


Figure 19: MATLAB result of TCD calculation.

### 3.2.2.12 Implementation and Testing of Spectral Phase Error (SPE)

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

Explanation of the SPE code (each line number corresponds to its code in Appendix B-14):

Line 1: SPE function with two inputs `rel_img` which refers to the original image and `enhcd_img` refers to the enhanced image, Line 4: we use a ready MATLAB function “fft” to transform the original and enhanced image to the frequency domain, Line 5: `[m n]` where `m` is amount of row the real image has and `n` is amount of columns the real image has; Line 6: difference between real image and enhanced image will be calculated, Line 7: using the formula provided in (2.12) `spe` will be calculated.

| FFT Original Image |         |       |        |
|--------------------|---------|-------|--------|
| 62 +0i             | -11+15i | 0+0i  | -11-   |
| 2-6i               | 1+3i    | 10+4  | -13+7i |
| -22+0i             | -5-3i   | -8+0i | -5+3i  |
| 2+6i               | -13-7i  | 10-4i | 1-3i   |

| Distorted Image |   |    |   |
|-----------------|---|----|---|
| 2               | 5 | 10 | 4 |
| 0               | 1 | 5  | 1 |
| 3               | 6 | 2  | 3 |
| 1               | 3 | 10 | 0 |

| FFT Distorted Image |        |        |            |
|---------------------|--------|--------|------------|
| 56+0i               | -21-7i | 10+0i  | -<br>21+   |
| 7+7i                | -6-2i  | 7+5i   | -12-<br>6i |
| 14+0i               | 7-1i   | -12+0i | 7+<br>1i   |
| 7-7i                | -12+6i | 7-5i   | -<br>6+2   |

| Argument (Original Image) |         |          |          |
|---------------------------|---------|----------|----------|
| 0                         | 2.2035  | 0        | - 2.2035 |
| -1.2490                   | 1.2490  | 0.3805   | 2.6477   |
| 3.1416                    | -2.6012 | 3.1416   | 2.6012   |
| 1.2490                    | -2.6477 | - 0.3805 | -1.2490  |

| Argument (Enhanced Image) |         |         |         |
|---------------------------|---------|---------|---------|
| 0                         | -2.8198 | 0       | 2.8198  |
| 0.7854                    | -2.8198 | 0.6202  | -2.6779 |
| 0                         | -0.1419 | 3.1416  | 0.1419  |
| -0.7854                   | 2.6779  | -0.6202 | 2.8198  |

| Original – Enhanced |          |         |         |
|---------------------|----------|---------|---------|
| 0                   | -0.6163? | 0       | -0.6163 |
| 0.4636              | -1.5708  | -0.2397 | -0.0303 |
| 3.1416              | 2.4593   | 0       | 2.4593  |
| 0.4636              | -0.0303  | -0.2397 | -1.5708 |

$$\text{SPE} = 1/16 (0^2 + 0.6163^2 + 0^2 + 0.6163^2 + 0.4636^2 + 1.5708^2 + 0.2397^2 + 0.0303^2 + 3.1416^2 + 2.4593^2 + 0^2 + 2.4593^2 + 0.4636^2 + 0.0303^2 + 0.2397^2 + 1.5708^2) =$$

$$\text{SPE} = 28.20703848/16 = 1.7629 \quad (3.12)$$

The SPE result calculation shown in Figure 20 is equal to (3.14).

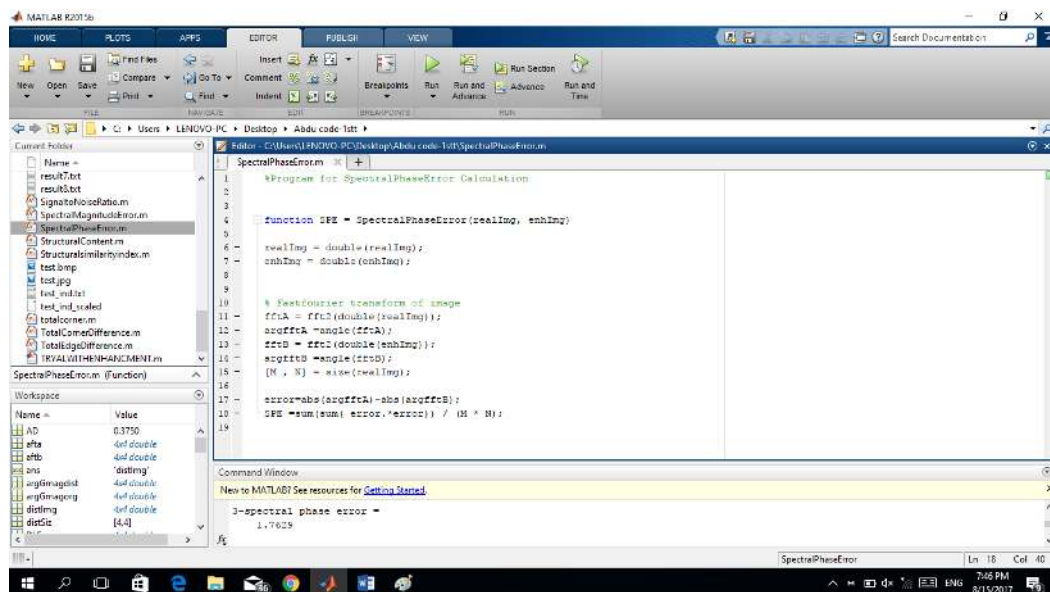


Figure 20: MATLAB result of SPE calculation.

### 3.2.2.13 Implementation and Testing of SME (2.13)

Explanation of the SME code (each line number corresponds to its code in Appendix A13):

Line 1: SME function has two inputs `rel_img` which refers to the original image and `enhcd_img` refers to the enhanced image, Line 4: we use a ready matlab function “**fft**” to transform the real image to the frequency domain, Line 5: shows the real part of the original image Line 6: shows the imaginary part of the original image Line 7: we take the SQRT of the real part summed with imaginary part of the original image, Line 8: shows the real part of the enhanced image, Line 9: shows the imaginary part of the enhanced image Line 10: we take the sqrt of the real part summed with imaginary part of the enhanced image Line 11: using the formula provided in (2.13) SME will be calculated.

Fourier Transform [13]:

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

| FFT Original Image |         |       |         |
|--------------------|---------|-------|---------|
| 62+0i              | -11+15i | 0+0i  | -11-15i |
| 2-6i               | 1+3i    | 10+4  | -13+7i  |
| -22+0i             | -5-3i   | -8+0i | -5+3i   |
| 2+6i               | -13-7i  | 10-4i | 1-3i    |

| Distorted Image |   |    |   |
|-----------------|---|----|---|
| 2               | 5 | 10 | 4 |
| 0               | 1 | 5  | 1 |
| 3               | 6 | 2  | 3 |
| 1               | 3 | 10 | 0 |

| FFT Distorted Image |        |        |        |
|---------------------|--------|--------|--------|
| 56+0i               | -21-7i | 10+0i  | -21+7i |
| 7+7i                | -6-2i  | 7+5i   | -12-6i |
| 14+0i               | 7-1i   | -12+0i | 7+1i   |
| 7-7i                | -12+6i | 7-5i   | -6+2i  |

FFT (Original Image) Gradient:

$$\text{Sqrt } 62^2 + 0^2 = 62.$$

$$\text{Sqrt } -11^2 + 15^2 = \text{Sqrt } 121+225 = \text{Sqrt } 346 = 18.6011.$$

| Gradient (Original Image) |         |         |         |
|---------------------------|---------|---------|---------|
| 62                        | 18.6011 | 0       | 18.6011 |
| 6.3246                    | 3.1623  | 10.7703 | 14.7648 |
| 22                        | 5.8310  | 8       | 5.8310  |
| 6.3246                    | 14.7648 | 10.7703 | 3.1623  |

| Gradient (Distorted Image) |         |     |         |
|----------------------------|---------|-----|---------|
| 56                         | 22.1359 | 10  | 22.1359 |
| 9.8                        | 6.3246  | 8.6 | 13.4164 |
| 14                         | 7.0711  | 12  | 7.0711  |
| 9.8                        | 13.4164 | 8.6 | 6.3246  |

| Original – Distorted |        |       |        |
|----------------------|--------|-------|--------|
| 6                    | -3.534 | -10   | -3.534 |
| -3.574               | -3.162 | 2.168 | 1.348  |
| 8                    | -1.24  | -4    | -1.24  |
| -3.574               | 1.348  | 2.168 | -3.162 |

$$\text{SME} = (6^2 - 3.534^2 - 10^2 - 3.534^2 - 3.574^2 - 3.162^2 + 2.168^2 + 1.348^2 + 8^2 - 1.24^2 - 4^2 - 1.24^2 - 3.574^2 + 1.348^2 + 2.168^2 - 3.162^2) / 16$$

$$\text{SME} = 295.795364 / 16 = 18.9 \quad (3.13)$$

The result of SME calculation shown in Figure 21 is equal to our manual calculation result (3.13).

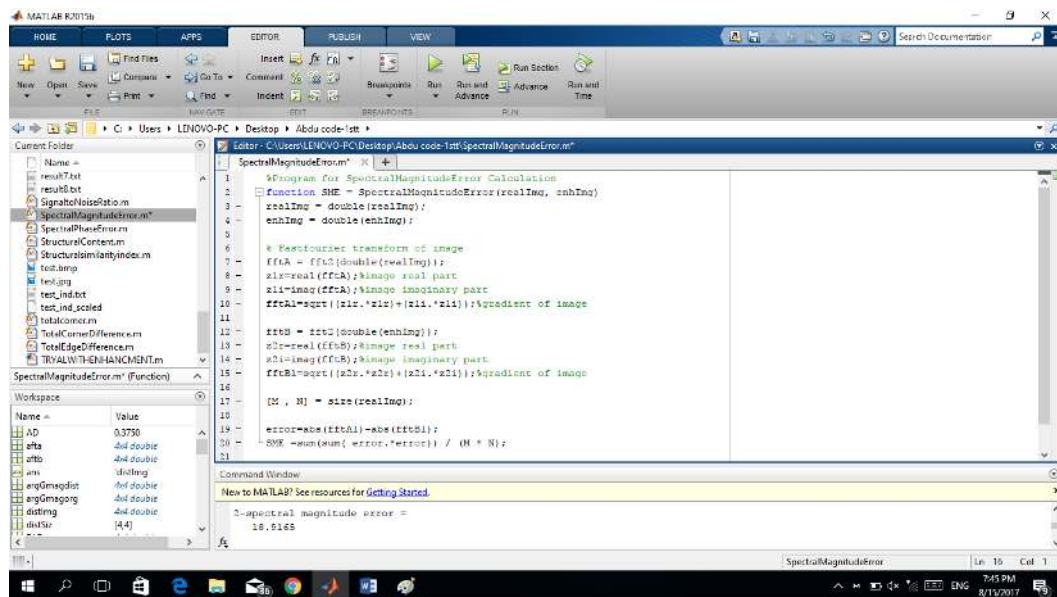


Figure 21: MATLAB result of SME calculation.

### 3.2.2.14 Implementation and Testing of Gradient Phase Error (GPE)

Explanation of the GPE code (each line number corresponds to its code in Appendix A-14):

Line 1: GPE function has two inputs `rel_img` which refers to the real image and `enhcd_img` refers to the enhanced image, Line 4: the gradient 2 of the original image will be calculated, Line 5: transfer gradient values of the original image to complex number, Line 6: angle of complex of the original image will be calculated, Line 7: the gradient of the enhanced image will be calculated, Line 8: transfer gradient values of the enhanced image to complex, Line 9: angle of complex of the enhanced image will be calculated Line 10:  $[m \ n]$  where  $m$  is amount of row the real image has and  $n$  is amount of columns the real image has; Line 11: difference between real image and enhanced image will be calculated and saved in 'error' variable, Line 12: using the formula provided in (2.14) GPE will be calculated.

Gradient of The Original Image:

| Fx1 |      |     |    |
|-----|------|-----|----|
| 1   | 3.5  | 1   | -4 |
| -2  | 2    | 4   | 2  |
| 2   | 0.5  | 2   | 5  |
| -3  | -0.5 | 0.5 | -1 |

| Fy1 |     |      |     |
|-----|-----|------|-----|
| 4   | 1   | 1    | 7   |
| 0   | 0.5 | -3   | 1.5 |
| 1   | 0.5 | -1.5 | -3  |
| 6   | 1   | 4    | -2  |

| Z =     |            |          |          |
|---------|------------|----------|----------|
| $1+4i$  | $3.5+1i$   | $1+1i$   | $-4+7i$  |
| $-2+0i$ | $2+0.5i$   | $4-3i$   | $2+1.5i$ |
| $2+1i$  | $0.5+0.5i$ | $2-1.5i$ | $5-3i$   |
| $-3+6i$ | $-0.5+1i$  | $0.5+4i$ | $-1-2i$  |

| F =      |            |                         |           |
|----------|------------|-------------------------|-----------|
| $3-2i$   | $4-4i$     | $-0.5-5i$               | $-6-3i$   |
| $1+0.5i$ | $2.5+0.5i$ | $0-4i$                  | $-4-0.5i$ |
| $3+0.5i$ | $-0.5+1$   | -                       | $1-0.5i$  |
| $2-2i$   | $4-3i$     | $1.5+2.5i$<br>$-1.5+8i$ | $-10-3i$  |

| Arg Original (Angle(Z)) |        |         |         |
|-------------------------|--------|---------|---------|
| 1.3258                  | 0.2783 | 0.7854  | 2.0899  |
| 3.1416                  | 0.2450 | -0.6435 | 0.6435  |
| 0.4636                  | 0.7854 | -0.6435 | -0.5404 |
| 2.0344                  | 2.0344 | 1.4464  | -2.0344 |

| Arg Distorted (Angle(F)) |         |         |         |
|--------------------------|---------|---------|---------|
| -0.5880                  | -0.7854 | -1.6705 | -2.6779 |
| 0.4636                   | 0.1974  | -1.5708 | -3.0172 |
| 0.1651                   | 2.0344  | 2.1112  | -0.4636 |
| -0.7854                  | -0.5880 | 1.7561  | -2.8501 |

| Arg Original – Arg Distorted |         |         |         |
|------------------------------|---------|---------|---------|
| 0.7378                       | -0.5071 | -0.8851 | -0.5880 |
| 2.6779                       | 0.0476  | -0.9273 | -2.3737 |
| 0.2985                       | -1.2490 | -1.4677 | 0.0768  |
| 1.2490                       | 1.4464  | -0.3097 | -0.8157 |

$$\text{GPE} = 1/16 (0.7378^2 - 0.5071^2 - 0.8851^2 - 0.5880^2 + 2.6779^2 + 0.0476^2 - 0.9273^2 - 2.3737^2 + 0.2985^2 - 1.2490^2 - 1.4677^2 + 0.0768^2 + 1.2490^2 + 1.4464^2 - 0.3097^2 - 0.8157^2) =$$

$$\text{GPE} = (11.46483746 + 12.50639178) / 16 = 1.48 \quad (3.14)$$

The result of GPE calculation shown in figure 22 is equal to our manual calculation result (3.14).

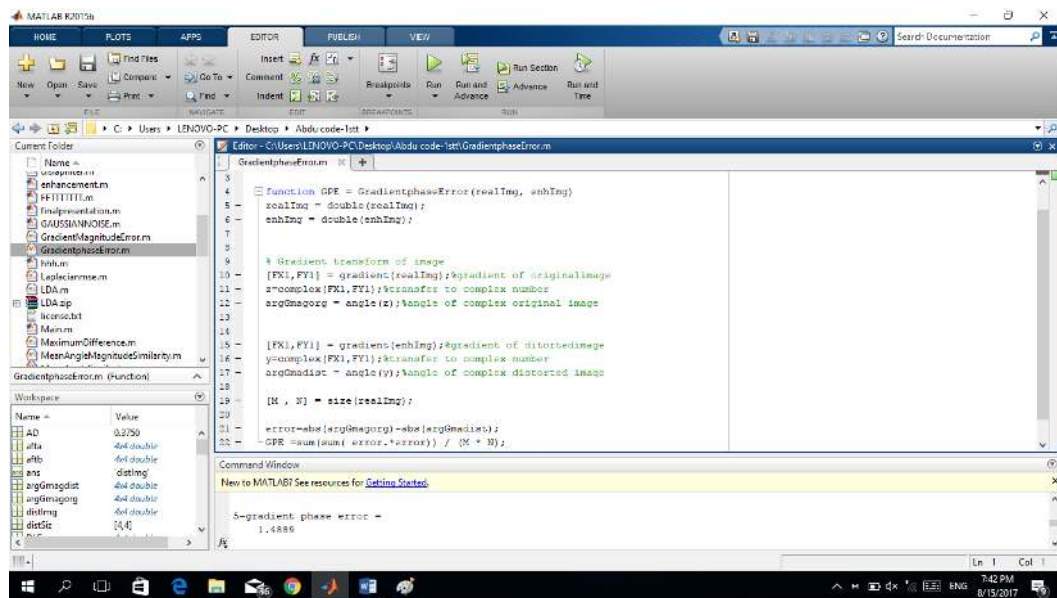


Figure 22: MATLAB result of GPE calculation.

### 3.2.2.15 Implementation and Testing of Gradient Magnitude Error (GME)

Explanation of the GME code (each line number corresponds to its code in Appendix A-15):

Line 1: GME function with two inputs `rel_img` which refers to the real image and `enhcd_img` refers to the enhanced image, line 4: using a ready MATLAB function the gradient (shows directional change in the intensity or color in an image) of the real image will be calculated, then the gradient magnitude is calculated and saved in “`gamgorg`” variable, same steps will be applied for the enhanced image, Line 5: `[m n]` where `m` is amount of row the real image has and `n` is amount of columns the real image has; Line 6: difference between real image and enhanced image will be calculated and saved in “`eror`” variable, Line 7: using the formula provided in (2.15) GME will be calculated.

For testing our `gme` function used in our RFDS we will calculate manually the GME of one pixel only, which is the matrix element with value equal to (2).

| Original Image (I) |   |   |    |
|--------------------|---|---|----|
| 0                  | 1 | 7 | 3  |
| 4                  | 2 | 8 | 10 |
| 0                  | 2 | 1 | 6  |
| 6                  | 3 | 5 | 4  |

Gradient of Pixel (2,2)

$$F_{x1} = (Df/Dx) = (8 - 4) / 2$$

$$4 / 2 = 2$$

$$F_{y1} = (Df/Dx) = (2 - 1) / 2$$

$$1 / 2 = 0.5$$

$$\text{Sqrt } 2^2 + 0.5^2 = \text{Sqrt } 4.25 = 2.061$$

| Fx1 |      |     |    |
|-----|------|-----|----|
| 1   | 3.5  | 1   | -4 |
| -2  | 2    | 4   | 2  |
| 2   | 0.5  | 2   | 5  |
| -3  | -0.5 | 0.5 | -1 |

| Fy1 |     |      |     |
|-----|-----|------|-----|
| 4   | 1   | 1    | 7   |
| 0   | 0.5 | -3   | 1.5 |
| 1   | 0.5 | -1.5 | -3  |
| 6   | 1   | 4    | -2  |

| Original Image I |       |       |       |
|------------------|-------|-------|-------|
| 4.123            | 3.640 | 1.414 | 8.062 |
| 2                | 2.061 | 5     | 2.5   |
| 2.236            | 0.707 | 2.5   | 5.831 |
| 6.708            | 1.118 | 4.031 | 2.236 |

| Distorted Image 'I |       |       |        |
|--------------------|-------|-------|--------|
| 3.605              | 5.656 | 5.024 | 6.708  |
| 1.118              | 2.549 | 4     | 4.031  |
| 3.041              | 1.118 | 2.915 | 1.118  |
| 2.828              | 5.408 | 8.139 | 10.440 |

| Original – Distorted |        |        |        |
|----------------------|--------|--------|--------|
| 0.517                | -2.016 | -3     | 1.354  |
| 0.882                | -0.488 | 1      | -1.531 |
| -0.805               | -0.41  | -0.415 | 4.712  |
| 3.879                | -4.29  | -4.1   | -8.2   |

$$1/16 (0.517^2 + -2.016^2 + -3.610^2 + 1.354^2 + 0.882^2 + -0.488^2 + 1^2 + -1.531^2 + -0.805^2 + -0.41^2 + -0.415^2 + 4.712^2 + 3.879^2 + -4.29^2 + -4.1^2 + -8.2^2)$$

$$\begin{aligned} \text{GME} &= 1/16 (0.2673 - 4.0643 -13.0321+ 1.8333 + 0.7779 - 0.2381 + 1 - 2.3440 - \\ &0.6480 - 0.1681 -0.1722 + 22.2029 + 15.0466 -18.4041 -16.8100 + -67.2400) \\ \text{GME} &= 164.2489/16 = 10.27 \end{aligned} \quad (3.15)$$

The result of GME calculation shown in figure 23 is equal to our manual calculation result (3.15).

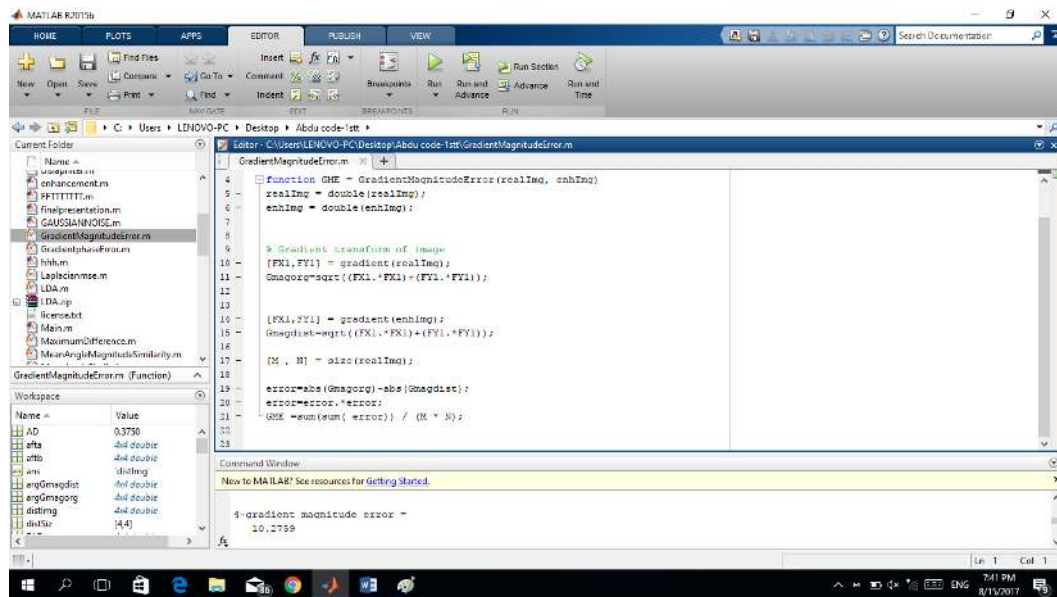


Figure 23: MATLAB result of GME calculation.

### 3.2.2.16 Implementation and Testing of Mean Angle Similarity (MAS)

Explanation of the MAS code (each line number corresponds to its code in Appendix B-16):

Line 1: MAS function with two inputs `rel_img` which refers to the original image and `enhcd_img` refers to the enhanced image, line 4: we use a ready MATLAB function “dot” to get the scalar dot of the real image and enhanced image, we also use another ready MATLAB function “acosd” which returns the inverse cosine of the element of the real; Line 5 : we take the sum of the variable “alfa2” ; Line 6: using the formula provided in (2.16) MAS will be calculated.

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

| Distorted Image |   |    |   |
|-----------------|---|----|---|
| 2               | 5 | 10 | 4 |
| 0               | 1 | 5  | 1 |
| 3               | 6 | 2  | 3 |
| 1               | 3 | 10 | 0 |

Alfa2 (Original Image, Distorted Image) = 0.9877, 0.9425, 0.6506, 0.9178

Angel2= Sum (Alfa2) = 3.4985

MAS= 1-(3.4985/16) = 0.7813 (3.16)

The result of MAS calculation shown in Figure 24 is equal to (3.16).

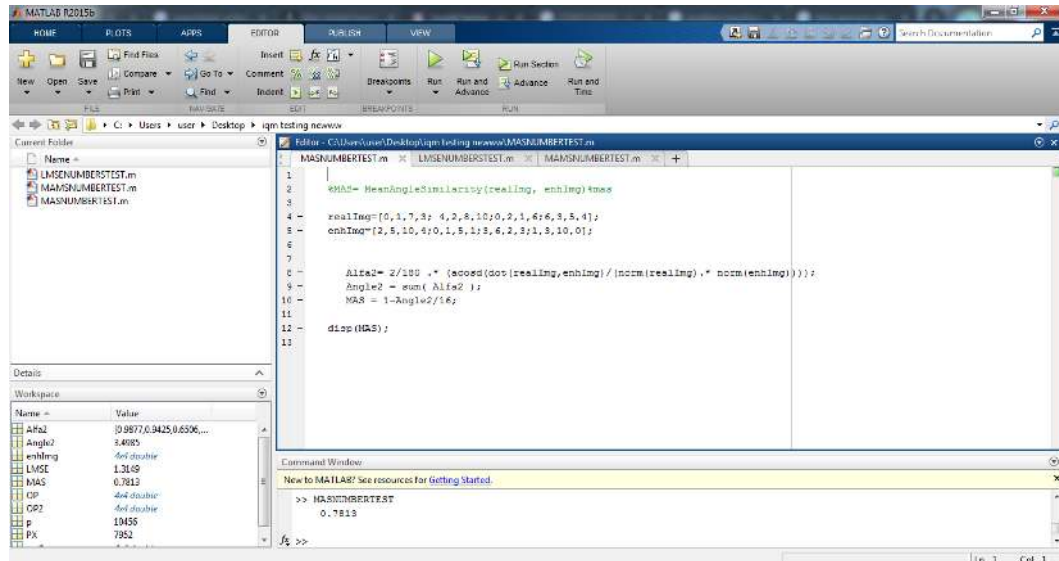


Figure 24: MATLAB result of MAS calculation.

### 3.2.2.17 Implementation and Testing of Mean Angle Similarity (MAMS)

Explanation of the MAMS code (each line number corresponds to its code in Appendix A-17):

Line 1: MAMS function with two inputs rel\_img which refers to the original image and enhcd\_img refers to the enhanced image, Line 4: we use a ready MATLAB

function “dot” to get the scalar dot of the real image and enhanced image, we also use another ready MATLAB function “acosd” which returns the inverse cosine of the element of the real; Line 5 :” ; Line 6 : using the formula provided in (2.17) MAMS will be calculated and assigned to variable “x”; Line 7: we take the sum of the variable “x; Line 8: we divide the sum and assign the result to variable ”MAMS”.

Alfa (Original Image, Distorted Image) = 0.9877, 0.9425, 0.6506, 0.9178

$X = 1 - (1 - \text{Alfa}) * (1 - (\text{Original Image} - \text{Distorted Image} / 255)) = 0.9883, 0.9452, 0.6667, 0.9215$

**SUM** = Sum(X) = 3.5216

**MAMS** = (3.5216/16) = 0.2201 (3.17)

The result of MAMS calculation shown in Figure 25 is equal to (3.17).

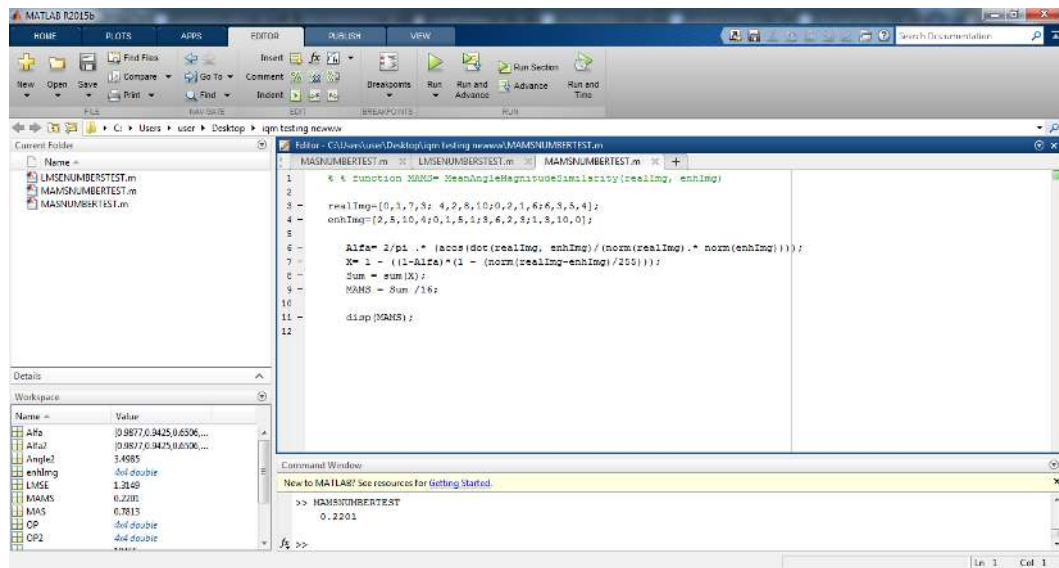


Figure 25: MATLAB result of MAMS calculation.

### 3.2.2.18 Implementation and testing of Laplacian Mean Squared Error (LMSE)

Explanation of the LMSE code (each line number corresponds to its code in Appendix A-18):

Line 1: LMSE function with two inputs: `rel_img` which refers to the real image, and `enhcd_img` refers to the enhanced image; line 4: `[m n]` where `m` is amount of row the real image has and `n` is amount of columns the real image has; Line 5& Line 6: we use a ready MATLAB function “`del2`” to calculate the second derivavtive of the real image and distorted image; Line 7: using the formula provided in (2.18) LMSE will be calculate.

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

| Del2 Original Image |    |     |     |
|---------------------|----|-----|-----|
| -6                  | 2  | -37 | -49 |
| 12                  | 7  | -12 | -27 |
| -2                  | -2 | 17  | 17  |
| 41                  | 8  | 27  | 4   |

| Original Image |   |   |    |
|----------------|---|---|----|
| 0              | 1 | 7 | 3  |
| 4              | 2 | 8 | 10 |
| 0              | 2 | 1 | 6  |
| 6              | 3 | 5 | 4  |

| Del2 Original Image |    |     |     |
|---------------------|----|-----|-----|
| -6                  | 2  | -37 | -49 |
| 12                  | 7  | -12 | -27 |
| -2                  | -2 | 17  | 17  |
| 41                  | 8  | 27  | 4   |

| Distorted Image |   |    |   |
|-----------------|---|----|---|
| 2               | 5 | 10 | 4 |
| 0               | 1 | 5  | 1 |
| 3               | 6 | 2  | 3 |
| 1               | 3 | 10 | 0 |

| Del2 Distorted Image |     |     |     |
|----------------------|-----|-----|-----|
| 30                   | 28  | -18 | -9  |
| 19                   | 12  | -6  | -14 |
| -24                  | -15 | 16  | 12  |
| 12                   | -20 | 3   | -54 |

| Del2 Original Image - Del2 Distorted |     |     |     |
|--------------------------------------|-----|-----|-----|
| -36                                  | -26 | -19 | -40 |
| -7                                   | -5  | -6  | -13 |
| 22                                   | 13  | 1   | 5   |
| 29                                   | 28  | 24  | 58  |

$$\text{sum}(\text{sum}(\text{op}-\text{op2}).^2)= 10456$$

$$\text{sum}(\text{sum}(\text{op}.^2))= 7952$$

$$\text{LMSE} = 10456/7952 = 1.3149. \quad (3.18)$$

The Result of LMSE calculation shown in Figure 26 is equal to (3.18).

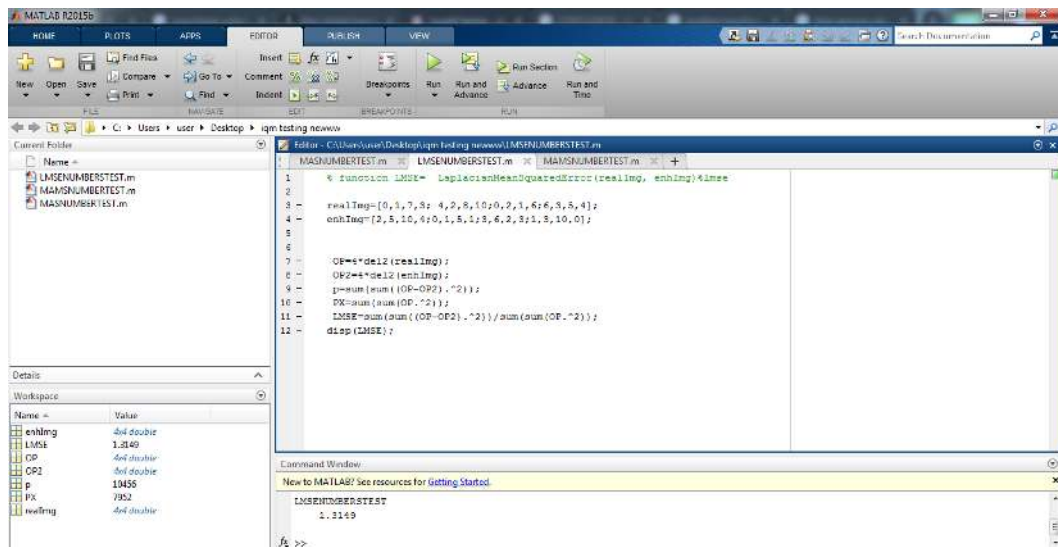


Figure 26: Matlab result of LMSE calculation.

For the rest of the IQMs (SSIM, VIF, RRED, HLFI, JQI, BIQI, AND NIQE) we implemented the following:

1. Downloaded ready implemented code from:

“live.ece.utexas.edu/research/quality/index\_algorithms.htm” Texas University [17].

We added the function files of every IQM to our thesis code folder. Then in our main.m file we call all the IQM functions. Input and output we get from these functions:

## 2. VIF

**Input:** (1) Img1: the reference image as a matrix. (2) Img2: the distorted image (Order Is Important).

**Output:** (1) VIF the visual information fidelity measure between the two images.

## 3. NIQE

**Input:** A test image loaded in an array.

**Output:** A quality score of the image. higher value represents a lower quality.

## 4. JPEG

**Input:** A test 8bits/pixel grayscale image loaded in a 2-d array.

**Output:** A quality score of the image. the score typically has a value. between 1 and 10 (10 represents the best quality, 1 the worst).

## 5. BIQI

**Input:** A test 8bits/pixel grayscale image loaded in A 2-D array.

**Output:** A quality score of the image. the score typically has a value. between 1 and 10 (10 represents the best quality, 1 the worst).

## 6. RRED

**Input:** A test 8bits/pixel grayscale image loaded in a 2-d array.

**Output:** Estimated entropy at different locations and the local spatial/temporal ('spatial' and 'temporal' refer to the spatial and temporal scaled entropy information for different blocks in the subband).

### **3.2.3 Implementation of Training Structure**

MATLAB 2015 provides a classification learner application that can be used to import tables from the work space. The application is very simple to use. It automatically extracts predictors (our IQM's) and observations (real or fake image), also has more than one classification algorithm such as LDA and QDA which are used for classifying the extracted samples of an image.

In order to use classification learner application, the results have to be arranged in a table and displayed in the workspace. The results displayed in workspace will be all imported to the classification learner application, then we select the table which has all the fingerprint image results of IQM's, after that we select the response (Real/Fake) and predictors (Quality Measures). The steps of implementing our classifier using the classification learner application are shown in Figures 27-31.

The screenshot shows the MATLAB R2016a interface. The Command Window displays a table with 15 columns: MSE, SNR, PSNR, NR, AD, SC, MD, RAND, NAE, TED, SME, SPE, QME, and GPE. The table contains 100 rows of data, representing results for 100 fingerprint images. The first two rows are highlighted in blue.

| MSE        | SNR    | PSNR   | NR      | AD         | SC     | MD      | RAND    | NAE      | TED       | SME    | SPE       | QME        | GPE     |
|------------|--------|--------|---------|------------|--------|---------|---------|----------|-----------|--------|-----------|------------|---------|
| 0.5267     |        |        |         |            |        |         |         |          |           |        |           |            |         |
| 0.0285     |        |        |         |            |        |         |         |          |           |        |           |            |         |
| 0.00092134 | 29.666 | 78.487 | 0.99406 | 0.0009757  | 1.0109 | 0.24721 | 0.2189  | 0.014769 | 0.0055885 | 165.73 | 0.0023344 | 0.00084169 | 0.52559 |
| 0.0018056  | 25.909 | 75.565 | 0.98719 | 0.00097497 | 1.0236 | 0.30818 | 0.27437 | 0.030249 | 0.010141  | 336.39 | 0.001637  | 0.0017335  | 0.66537 |
| 0.00092134 | 29.666 | 78.487 | 0.99406 | 0.0009757  | 1.0109 | 0.24721 | 0.2189  | 0.014769 | 0.0055885 | 165.73 | 0.0023344 | 0.00084169 | 0.52559 |
| 0.0018056  | 25.909 | 75.565 | 0.98719 | 0.00097497 | 1.0236 | 0.30818 | 0.27437 | 0.030249 | 0.010141  | 336.39 | 0.001637  | 0.0017335  | 0.66537 |
| 0.00077608 | 30.079 | 79.232 | 0.99434 | 0.00097585 | 1.0104 | 0.23563 | 0.22512 | 0.014612 | 0.0055781 | 137.54 | 0.0021246 | 0.00071072 | 0.4144  |
| 0.0010434  | 29.154 | 77.946 | 0.99401 | 0.00097608 | 1.0109 | 0.24728 | 0.22956 | 0.01595  | 0.0075729 | 189.12 | 0.002363  | 0.0010104  | 0.55784 |
| 0.00095809 | 28.114 | 78.317 | 0.99082 | 0.00097309 | 1.0171 | 0.24147 | 0.21345 | 0.024174 | 0.0056979 | 172.61 | 0.0021583 | 0.00085074 | 0.48851 |
| 0.00077608 | 30.079 | 79.232 | 0.99434 | 0.00097585 | 1.0104 | 0.23563 | 0.22512 | 0.014612 | 0.0055781 | 137.54 | 0.0021246 | 0.00071072 | 0.4144  |
| 0.0010909  | 28.682 | 77.753 | 0.99338 | 0.00097561 | 1.012  | 0.23743 | 0.21528 | 0.019604 | 0.0093906 | 198.53 | 0.002211  | 0.0010525  | 0.63225 |
| 0.0011389  | 28.352 | 77.566 | 0.99255 | 0.00097496 | 1.0136 | 0.22575 | 0.21819 | 0.020312 | 0.0087188 | 207.1  | 0.002274  | 0.0010683  | 0.54563 |
| 0.00061564 | 28.48  | 80.238 | 0.99175 | 0.0007184  | 1.0153 | 0.21268 | 0.18518 | 0.0269   | 0.0063333 | 111.15 | 0.0057324 | 0.00052643 | 0.38322 |
| 0.00065386 | 28.631 | 79.976 | 0.99049 | 0.00087616 | 1.018  | 0.20167 | 0.17304 | 0.026395 | 0.0064844 | 116.55 | 0.0036697 | 0.00057279 | 0.38598 |
| 0.00061564 | 28.48  | 80.238 | 0.99175 | 0.0007184  | 1.0153 | 0.21268 | 0.18518 | 0.0269   | 0.0063333 | 111.15 | 0.0057324 | 0.00052643 | 0.38322 |
| 0.00065386 | 28.631 | 79.976 | 0.99049 | 0.00087616 | 1.018  | 0.20167 | 0.17304 | 0.026395 | 0.0064844 | 116.55 | 0.0036697 | 0.00057279 | 0.38598 |
| 0.00083715 | 28.138 | 78.903 | 0.99067 | 0.00088737 | 1.0174 | 0.2066  | 0.18413 | 0.028246 | 0.0064427 | 151.76 | 0.0046852 | 0.00073305 | 0.49141 |
| 0.00086932 | 26.75  | 78.739 | 0.98739 | 0.00083905 | 1.0237 | 0.20167 | 0.18093 | 0.035568 | 0.0069479 | 158.29 | 0.0047787 | 0.00075073 | 0.37509 |
| 0.00058409 | 29.329 | 80.466 | 0.99208 | 0.00087389 | 1.0149 | 0.20167 | 0.17253 | 0.022648 | 0.0047917 | 102.95 | 0.0036381 | 0.00052062 | 0.44672 |
| 0.00042173 | 29.188 | 81.881 | 0.9917  | 0.00079205 | 1.0156 | 0.20167 | 0.17447 | 0.025268 | 0.0046875 | 73.786 | 0.0084337 | 0.00036497 | 0.36521 |

Figure 27: Table which has all the fingerprint image results of IQMS we used.

Figure 27 shows the table created in the workspace with 100 fingerprint images, 50 fake and 50 real images. To create table to display our RFDS results we had to set a variable for every single IQM, each variable contains results calculated for one single IQM, then we used a ready MATLAB function called “table” for creating tables with our IQM variables in it. The table is imported to the classification learner application. Table is not completely shown in figure 27 due to screenshot limitation.

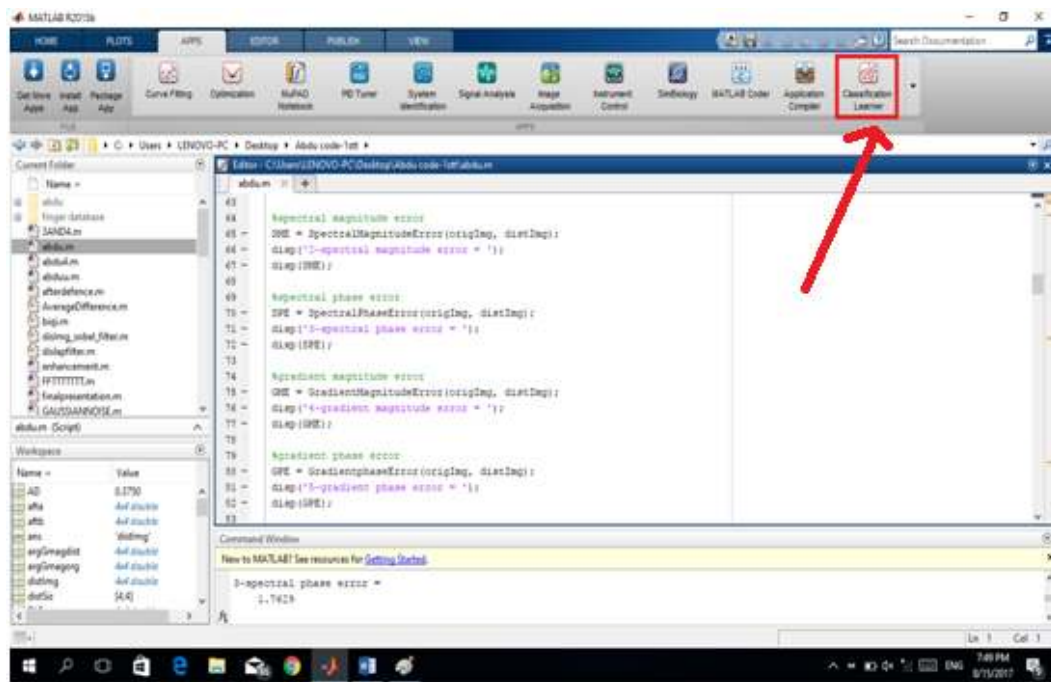


Figure 28: The arrow shows a button used to call the classification learner app in MATLAB.

Figure 28 shows all the apps provided in MATLAB 2015; to access the classification learner app in MATLAB click on apps, select classification learner app.

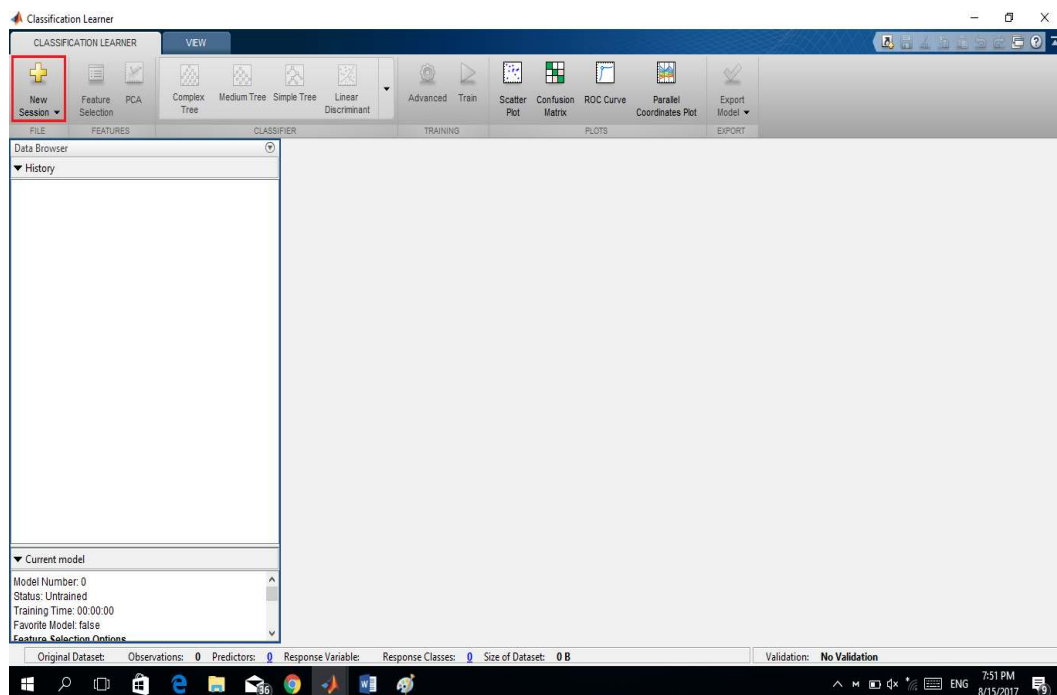


Figure 29: Start new session to import results.

Figure 29 shows classification learner first page, click new session to import all datasets from the recent work space.

**Step 1**  
Select dataset from MATLAB workspace.

**Step 2**  
Select predictors and response.

| Name  | Type   | Range                | Import as |
|-------|--------|----------------------|-----------|
| USERS | cell   | 2 unique             | Response  |
| MSE   | double | 0.000276086 .. 0.... | Predictor |
| SNR   | double | 24.0263 .. 34.9059   | Predictor |
| PSNR  | double | 75.7656 .. 83.7204   | Predictor |
| NK    | double | 0.980054 .. 0.997... | Predictor |
| AD    | double | 0.000811909 .. 0.... | Predictor |
| SC    | double | 1.00396 .. 1.03727   | Predictor |
| MD    | double | 0.20167 .. 0.302508  | Predictor |
| RAMD  | double | 0.151429 .. 0.260... | Predictor |
| NAE   | double | 0.00785197 .. 0.0... | Predictor |
| TED   | double | 0.00480729 .. 0.0... | Predictor |
| SME   | double | 42.59 .. 319.175     | Predictor |
| SPE   | double | 0.00148655 .. 0.0... | Predictor |
| GME   | double | 0.000284421 .. 0.... | Predictor |
| GPE   | double | 0.276111 .. 0.788... | Predictor |
| TCD   | double | 0.000606796 .. 0.... | Predictor |

**Step 3**  
Define validation method.

☒ **Cross Validation**  
Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.

Cross-validation folds: 5 folds

☐ **Holdout Validation**  
Recommended for large data sets.

Percent held out: 25%

☐ **No Validation**  
No protection against overfitting.

[Read about validation](#)

Start Session Cancel

Figure 30: Selection of our dataset, predictors and response in MATLAB.

Figure 30 shows the imported datasets from the works space which includes table t, the one we created, response (Real/Fake), and the predictors which is the image quality measures.

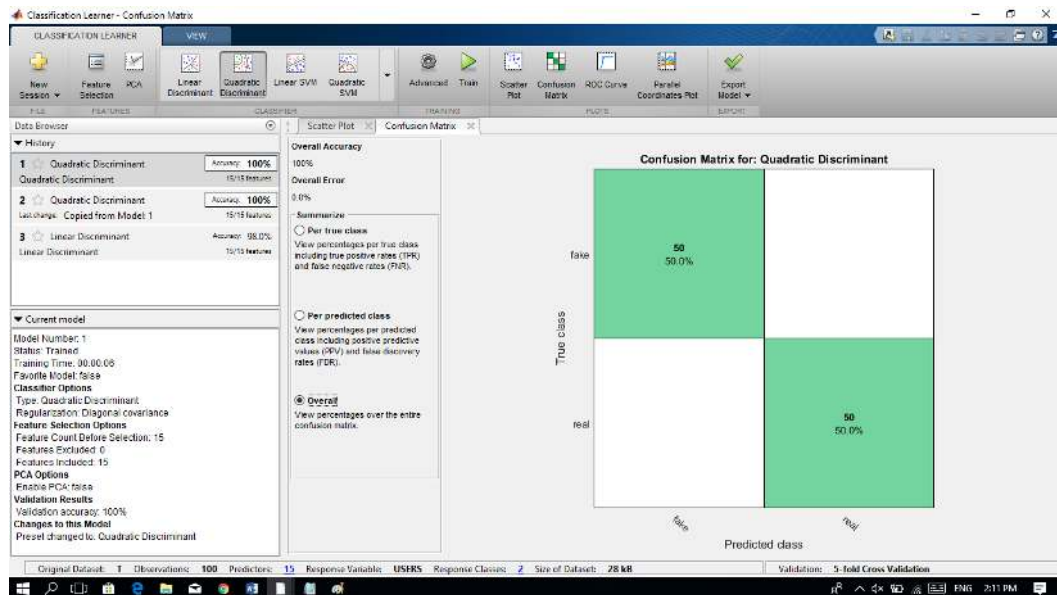


Figure 31: Selection of the classifier for our table imported and training result.

Figure 31 shows the classifiers that can be selected on the left-side of the screenshot, in our case we selected LDA classifier shown on the left corner of the figure, we then clicked train button to start the classification process for the fingerprint images which were selected for training. The right-side of Figure 29 shows the result of training 50 real and 50 fake fingerprint images, the training result had classified them all correctly. This training result obtained is considered as our training model, Figure 30 shows how to use and export the training model.

### 3.2.4 Implementation of The Classification Process

The training model we got from training on 100 fingerprint images is required for classification process. We input 1000 images from the fingerprint database for RFDS to classify the images on real and fake. There are four stages to implement classification process.

1. Export model of the classifier to the workspace to classify fingerprint images. Figure 30 shows how to use and export the training model

2. To classify the fingerprint images using the training model obtained from the classification learner app, we need to call a ready MATLAB function `"yfit = trainedclassifier.predictfcn(t)"` and insert it after the table creation code, because this function will predict the fingerprint images if they are real or fake using a table with all the IQM results listed in it, in our case the function will predict table 'T' results real or fake based on the training model we obtained.

3. 1000 fingerprint images were used for classification. (Appendix C2) shows results of the 1000 images classification.

4. Run and classify.

The steps that show how the classification process works are shown in Figures 32-35: Click on the rectangle highlighted on the top right corner of Figure 32 to export the training model of 100 real and fake fingerprint training images, we export this model to our work space for the classification process. We enter then random images of fingerprints from different users and let our training model classify if the images FED to the RFDS are real or fake. For exporting the model click on the top right button "export model" and select the first option export model", by this, the training model will be exported to the workspace.

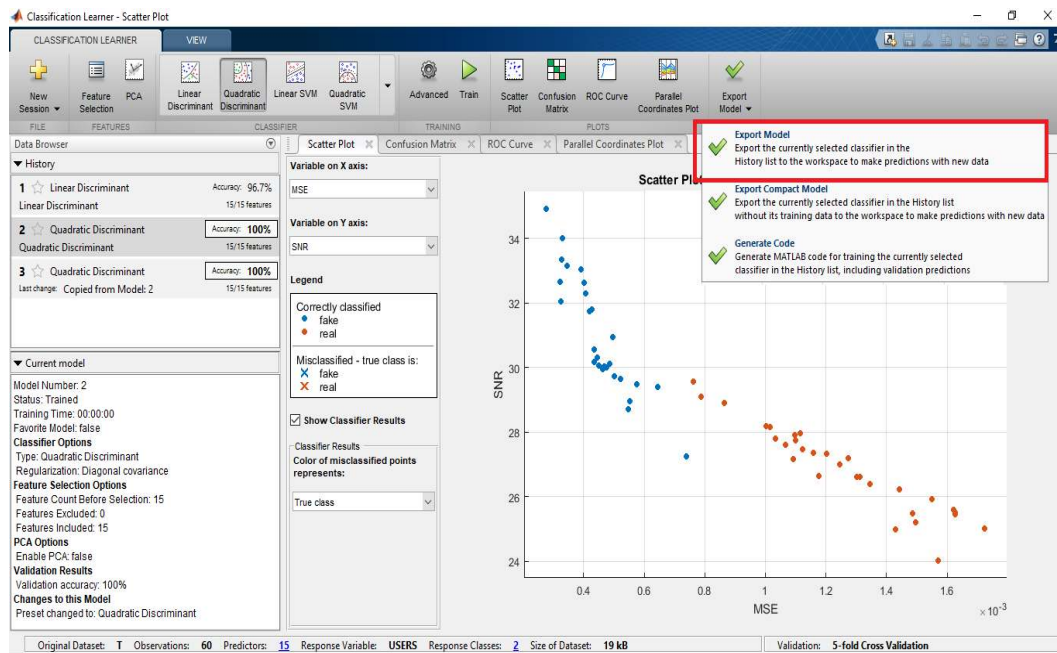


Figure 32: Exporting training model of 100 real and fake fingerprint images.

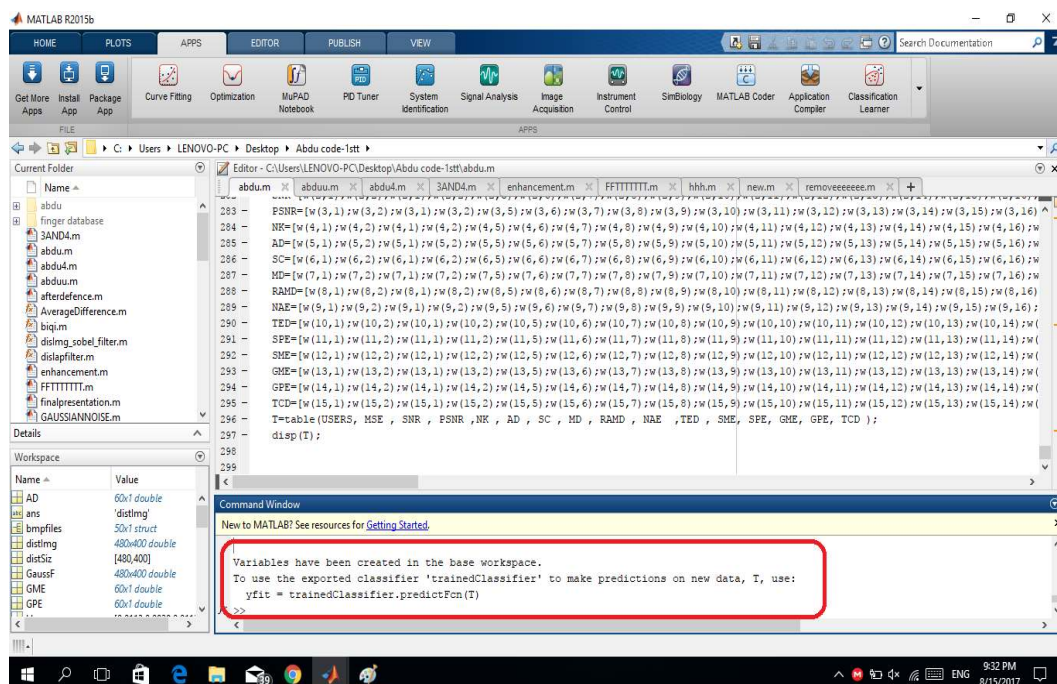


Figure 33: After exporting the training model, we can use it to make prediction by the code provided in the command window highlighted with a rectangle.

Figure 33, shows that our training model has been exported to the works space, as it can be seen clearly in the command window. If this highlighted rectangle message

doesn't appear in the command window, it means, the training model wasn't exported successfully.

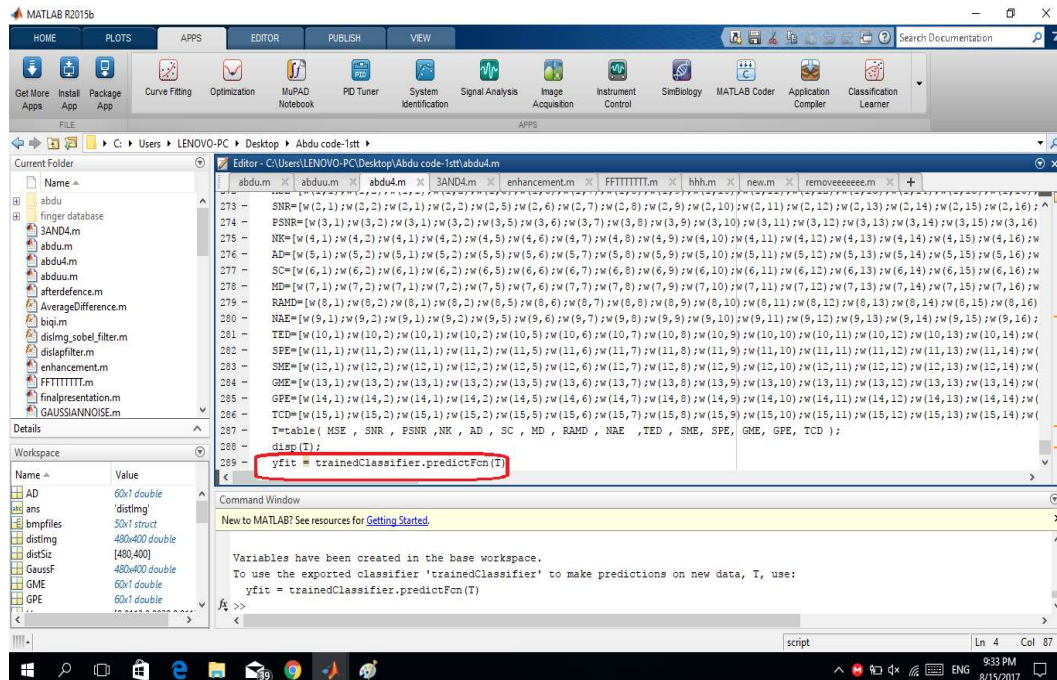


Figure 34: The highlighted rectangle shows the code used to classify images according to our exported training model.

Figure 34, Shows the code used for classifying the images “yfit = trainedclassifier.predictfcn(t)”. As we mentioned before this code has to be added after creating a table with all the results of the IQMS listed in it, so that we can classify the fingerprint images as real or fake. In our case we created a table named ‘T’ with all the results saved, then we added the function “yfit = trainedclassifier.predictfcn(t)” in our code to classify images results from the table ‘T’ according to our exported training model.

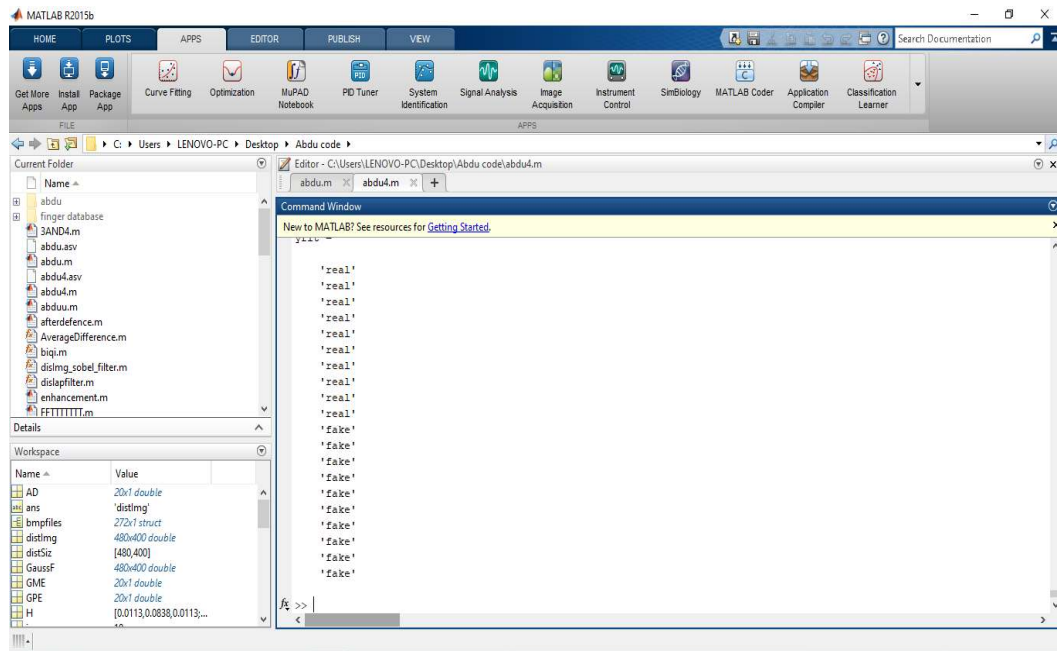


Figure 35: Results of 10 real and fake images classified correctly using the classification code.

Figure 35 shows the results obtained from fingerprint images after running the classification code (Appendix A). In this run we had 10 real images and 10 fake all were classified correctly. The fingerprint images used were obtained from fingerprint database atvs-ffp db [33]. Figure 36 shows a screenshot of one real and fake fingerprint image sample we used in our experiments. Real and fake finger print images were read from directory” c:\users\lenovo-pc\desktop\hp\_new folder (2)\abdu code\finger database\abdu\1biometrika\1m real”. Appendix A. Line 3. Shows the code we used for reading the directory of the fingerprint images.

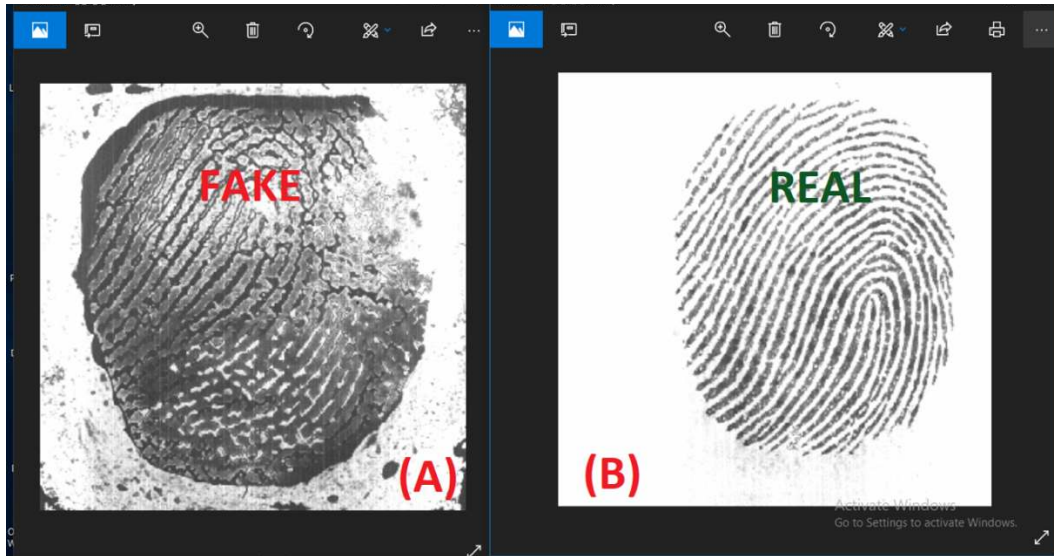


Figure 36: Fingerprint images used in our experiments, (A) fake fingerprint image from the file "U01\_F\_Fc\_Li\_01.Bmp", (B) real fingerprint image from the file "U01\_O\_Fc\_Li\_01.Bmp".

### 3.3 Development of A Method of Defining the Best IQMs

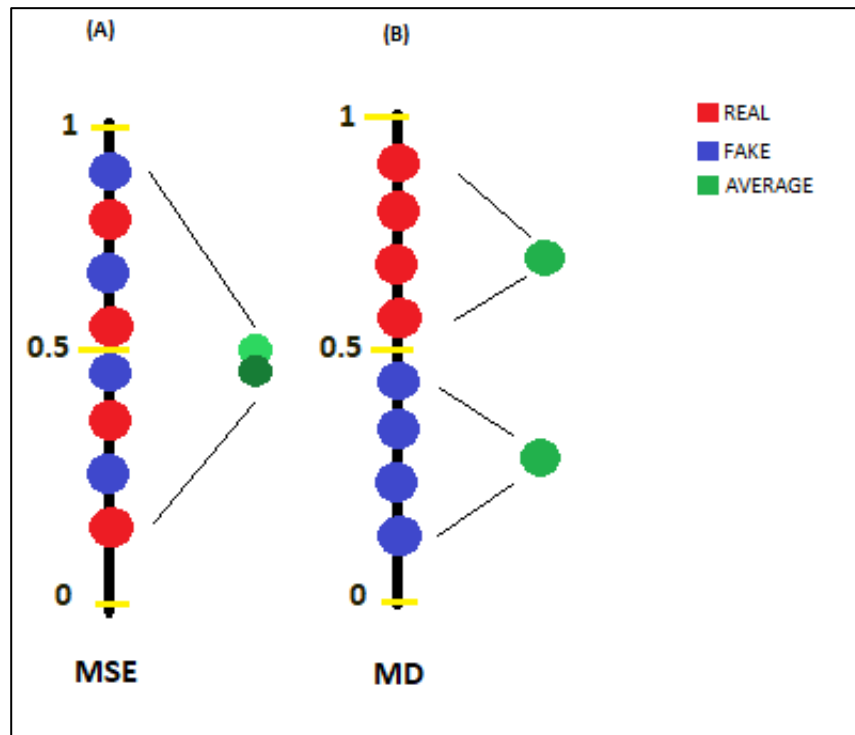


Figure 37: Numerical axis for MSE (A) And MD (B), showing MSE of fake (real) fingerprint images. And the respective averages calculated by (3.16) and (3.17).

Figure 37, shows a drawing that will help you to understand our basic idea behind finding the best 5 and 10 IQMs ,as you can see in Figure 34 there are two graphs (A) for MSE and (B) for MD, both of the graphs have red dots which represent the values of MSE and MD in real fingerprint images, the blue dots which represent the fake fingerprint image values of MSE and MD, and the green dots represent the average of MSE and MD in real fingerprint images (red dots) and fake fingerprint images (blue dots) calculated by (3.16) and (3.17) . Figure 34 has two scenarios: let's start with (A), the MSE red points are close to the blue points which means that the real and fake points in MSE are very close to each other and makes it difficult for us to distinguish between both of the points, in (A) the average point of the real points of MSE and the average point of the fake points of MSE are very close, in other words, it means that the real and fake points are very near to each other and it's hard to differentiate between them. In (B) the MD red and blue points are well separated from each other, so that means the real points and fake points can be well distinguished from each other, we can also see that in (B) the average points for the real and fake images are far away from each other, so (B) is far better than (A) in distinguishing real and fake fingerprint images, from these two cases we can say that MD is a better IQM than MSE. The IQM which will have the highest absolute value of the difference between the green dots representing averages will be set as the best in the list of the IQM sorted by the difference (3.18) in the descending order.

1. To find the best 5 and 10 IQM of our RFDS we need to apply the following steps for every measurement.

Find the IQM average of real fingerprint images ( $\bar{x}_r$ ).

$$\bar{x}_r = \frac{\sum_{i=1}^N x_i^r}{Nr} \quad (3.16)$$

Where  $x_i^r$  is the value of IQM in one single real fingerprint image, NR is number of real fingerprint images FED to the system.

2. Find the IQM average of fake fingerprint images ( $\bar{x}f$ ).

$$\bar{x}f = \frac{\sum_{j=1}^N x_j^f}{Nf} \quad (3.17)$$

Where  $x_i^r$  is the value of iqm in one single fake fingerprint image, nf is number of fake fingerprint images fed to the system.

3. Find the difference between real ( $\bar{x}r$ ) and fake fingerprint images ( $\bar{x}f$ ).

$$\Delta X = |\bar{x}r - \bar{x}f| \quad (3.18)$$

Apply these three steps for every single IQM, then sort them from maximum to minimum difference value which is found in step 3. The best IQM will be the one with the maximum distance value.  $\bar{x}r$ .

$$\text{Sort} = \max(\Delta Xr \dots \Delta Xn)$$

Where n is the number of iqm used in the system

### 3.3.1 Implementation and Testing of The Method of Defining the Best IQMs

The code for the defining the best IQMs was implemented using MATLAB. The complete code is shown in Appendix B.

Explanation of defining the best IQMS code (Appendix B-1):

Line 1: We set the results of the IQM from 1-500 as real values. Line 2: we set the results of the IQM from 501-1000 as fake values. Line 3: we take the average of the real values of the IQM. Lines 4 and 5: display the average of the real values of the IQMs. Line 6: we take the average of the fake values of the IQM. Lines 7 and 8: display the average of the fake values of the IQM; line 9: displays the difference of the

averages of real values and fake values of the IQMs. Line 10: we take the absolute value of the difference calculated. Line 11: we display the result.

Explanation of ranking the best IQMs code (Appendix B-2):

Line 1: We list the names of all IQMs in one array. Line 2: we list the difference of the averages of real and fake values of every IQM in one array. Line 3: we create a table with “title” and “best 15” as its fields. Line 4: we sort the table rows in descending order.

### **3.3.1.1 Calculation of Defining the Best 5 IQMs**

#### **1. For MSE**

Real fingerprint image values of the MSE: 0.0017764, 0.0016535, 0.0016386, 0.00163, 0.0016254

Fake fingerprint image values of the MSE: 0.0015887, 0.0015395, 0.0015314, 0.0014846, 0.0014052

Average of the real fingerprint image values of MSE:  $(0.0017764 + 0.0016535 + 0.0016386 + 0.00163 + 0.0016254) / 5 = 0.0017$

Average of the fake fingerprint image values of MSE:  $(0.0015887 + 0.0015395 + 0.0015314 + 0.0014846 + 0.0014052) / 5 = 0.0015$

Difference between the average of real fingerprint image values of MSE and average of the fake fingerprint image values of MSE:  $|0.0017 - 0.0015| = 0.0001549$

#### **2. For SNR**

Real fingerprint image values of the SNR: 32.608, 31.179, 30.719, 30.149, 30.149

Fake fingerprint image values of the SNR: 29.958, 29.674, 29.664, 29.584, 29.674

Average of the real fingerprint image values of SNR:  $(32.608 + 31.179 + 30.719 + 30.149 + 30.149) / 5 = 30.9608$

Average of the fake fingerprint image values of SNR:  $(29.958 + 29.674 + 29.664 + 29.584 + 29.674) / 5 = 29.7108$

Difference between the average of real fingerprint image values of SNR and average of the fake fingerprint image values of SNR:  $|30.9608 - 29.7108| = 1.2500$

### **3. For PSNR**

Real fingerprint image values of the PSNR: 81.171, 81.106, 80.41, 79.938, 79.541

Fake fingerprint image values of the PSNR: 79.142, 79.142, 78.689, 78.668, 78.522

Average of the real fingerprint image values of PSNR:  $(81.171 + 81.106 + 80.41 + 79.938 + 79.541) / 5 = 80.4332$

Average of the fake fingerprint image values of PSNR:  $(79.142 + 79.142 + 78.689 + 78.668 + 78.522) / 5 = 78.8326$

Difference between the average of real fingerprint image values of psnr and average of the fake fingerprint image values of PSNR:  $|80.4332 - 78.8326| = 1.6006$

### **4. For NK**

Real fingerprint image values of the NK: 0.99731, 0.99604, 0.99542, 0.99542, 0.9948

Fake fingerprint image values of the NK: 0.99453, 0.99447, 0.99417, 0.99409, 0.994

Average of the real fingerprint image values of NK:  $(0.99731 + 0.99604 + 0.99542 + 0.99542 + 0.9948) / 5 = 0.9958$

Average of the fake values of NK:  $(0.99453 + 0.99447 + 0.99417 + 0.99409 + 0.994) / 5 = 0.9943$

Difference between the average of real fingerprint image values of NK and average of the fake fingerprint image values of NK:  $|0.9958 - 0.9943| = 0.0015$

### **5. For AD**

Real fingerprint image values of the AD: 0.00097608, 0.00097608, 0.00097608, 0.00097608, 0.197608

Fake fingerprint image values of the AD: 0.297608, 0.07608, 0.208, 0.1608, 0.397608

Average of the real fingerprint image values of AD:  $(0.00097608 + 0.00097608 + 0.00097608 + 0.00097608 + 0.197608) / 5 = 0.0403$

Average of the fake fingerprint image values of AD:  $(0.297608 + 0.07608 + 0.208 + 0.1608 + 0.397608) / 5 = 0.2280$

Difference between the average of real fingerprint image values of ad and average of the fake fingerprint image values of AD:  $|0.0403 - 0.2280| = 0.1877$

The best IQM is the one with the fake fingerprint image values. The rank of the best 5 IQMs will be:

SNR     1.6006

SNR     1.25

AD       0.18772

NK       0.001546

MSE      0.0001549

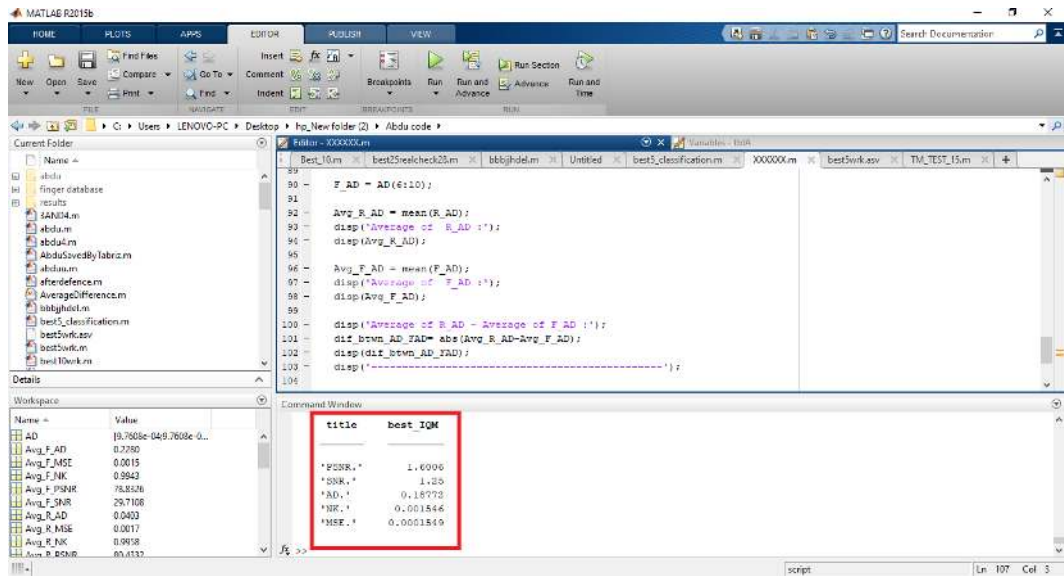


Figure 38: The list of the best IQMs (in the red rectangle).

Figure 38 shows a screenshot with the list of the best IQMs. Using MATLAB, we inserted the same values used for every IQM in the manual calculation above and applied the same steps of calculation, which generated expected results at the end. For our RFDS expirment we used the same method explained above to define our best 15, 10, and 5 IQMs. Figure 38 shows a screenshot of our best 15, 10, and 5 IQMs generated by our code which is available in Appendix B. The MATLAB screenshot in Figure 39 shows our IQMs listed as follows: for **best 10 IQMs**: SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, and SC, and the first five of them are the **Best 5 IQMs**.

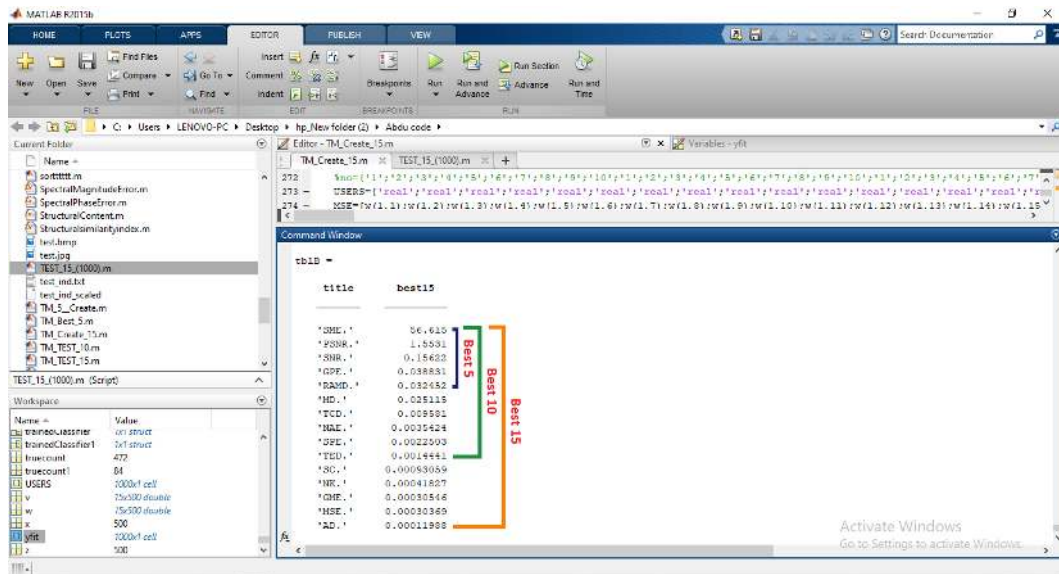


Figure 39: List of our best 15,10,5 IQMs used in our RFDS expirments.

### 3.4 Summary

In this chapter we have presented the implementation of our RFGS, including Gaussian Filter, Feature Extraction, Measurement of Training Structure, Classification Process and the best IQMs using MATLAB r2015b as shown in (Appendices A- C).

## Chapter 4

### EXPERIMENTS ON RFDS

#### 4.1 Experimental Setup

The experiment was implemented on a PC with Windows 10 pro, 64-bit operating system, CPU Intel Core I7 2.50 Ghz, 8 Gb Ram, and MATLAB r2015b.

#### 4.2 Database Used

ATVS fingerprint database [33] contains two datasets of fingerprint images with bmp file format:

**1. Dataset 1:** fake samples were captured from gummy fingers generated with the cooperation of the user. It contains fingerprint samples of the middle and index finger of both hands of 17 users, which means in total 68 ( $17 \times 4 = 68$ ) different fingers. Four samples of each fingerprint (Fake and Real) were captured in one acquisition session with the use of three sensors, so the dataset consists of  $68 \times 4 \times 3 = 816$  real image samples and as many fake images.

The three types of sensors are used in the dataset:

- Biometrika Sensor.
- Thermal Sensor.
- Flat Capacitive Sensor.

**2. Dataset 2:** ds\_withoutcooperation fake samples were captured from gummy fingers generated with the cooperation of the user. It contains fingerprint samples of the

middle and index fingers of both hands of 16 users, which means in total 64 (16x4) different fingers. Four samples of each fingerprint (Fake and Real) were captured in one acquisition session with the use of three sensors, this way the dataset comprises 768 ( $64 \times 4 \times 3 = 816$ ) real image samples and as many fake images.

In all the experiments we ran on the RFDS we used 50 real and 50 fake images for training stage, while in the classification stage we used 500 original fingerprint images and 500 fake fingerprint images a total of 1000 fingerprint images. Three different types of experiments on the RFDS were conducted: RFDS training and classification with 25 IQMs, RFDS training and classification with best 15 IQMs, RFDS training and classification with the best 10 IQMs, RFDS training and classification with the best 5 IQMs.

### **4.3 Classification Methods Used**

- LDA (Linear Discriminant Analysis)
- QDA (Quadratic Discriminant Analysis)
- Linear SVM, Quadratic SVM
- Logistic Regression, and fine KNN.

### **4.4 Code Parts for Experiments Conducting**

The experiment code is divided into three parts:

- Input the Fingerprint Image and Filtering Code.
- Feature Extraction Code.
- Classification Process.

In the first part of the code (Appendix A, lines 1-29), we input the fingerprint images and convert them from RGB to grayscale image. Then, we convert the image to double,

and apply a 3\*3 Gaussian filter to it; we resize both the original and fake fingerprint image to an equal size.

In the second part of the code (Appendix A, Lines 30-80), 25 image quality measures are implemented and each of them has a function for calculation, using the two inputs, real and enhanced fingerprint image. Each image quality measure function provided in the code was successfully tested. For training, 50 real fingerprint images and 50 fake fingerprint images with their 25 image quality measures calculated will be gathered all in a vector and arranged in a table to be able to import them to the classification learner application for training process.

In the third part which is the classification process, we use a user-friendly MATLAB application called “Classification Learner Application” to classify fingerprint images to real and fake images using six classifiers.

## 4.5 Experimental Results

### 4.5.1 Experiments on the RFDS Training and Classification with 15 IQMs

Table 4: Training results with 15 IQMs.

| <b>15 IQMs</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, SC, TED, NK, MSE, GME, AND AD; ATV Database, 50 Real And 50 Fake Fingerprints. |     |     |      |
|------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----|------|
| Classifier/Sample No.                                                                                                                          | FFR | FGR | HTER |
| Lda/1000 Images                                                                                                                                | 4%  | 30% | 17%  |
| Qda/100 Images                                                                                                                                 | 4%  | 24% | 14%  |
| Linear Svm/100 Images                                                                                                                          | 2%  | 2%  | 2%   |
| Quadratic Svm/100 Images                                                                                                                       | 0%  | 0%  | 0%   |
| Logistic Regression/100 Images                                                                                                                 | 0%  | 0%  | 0%   |
| Fine Knn/100 Images                                                                                                                            | 0%  | 0%  | 0%   |

The results listed in Table 4, are generated from ATV database. There were 50 real and 50 fake fingerprint images input to the system to train, the detailed results of the 100 samples of fingerprints using 15 IQMs and 6 classifiers are shown in Appendix C-1, Figures 38-43.

Table 5: Classification results with 15 IQMs.

| <b>15 IQMs</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, SC, TED, NK, MSE, GME, AND AD; ATV Database, 500 Real And 500 Fake Fingerprints |       |       |       |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-------|-------|-------|
| Classifier/Sample No.                                                                                                                           | FFR   | FGR   | HTER  |
| LDA/1000 Images                                                                                                                                 | 5.4%  | 21.6% | 13.5% |
| QDA/1000 Images                                                                                                                                 | 12.4% | 17.2% | 14.8% |
| Linear SVM/1000 Images                                                                                                                          | 8.4%  | 7.6%  | 8%    |
| Quadratic SVM/1000 Images                                                                                                                       | 1.6%  | 10%   | 5.8%  |
| Logistic Regression/1000 Images                                                                                                                 | 8%    | 9%    | 8.5%  |
| Fine KNN/1000 Images                                                                                                                            | 2.6%  | 13.6% | 8.1%  |

Table 5, shows the classification results with 15 IQMs for 500 real and 500 fake fingerprint images inputted to the system to classify, the detailed results of the 1000 samples of fingerprints for classification are shown in Appendix C-2, Figures 44–49.

#### 4.5.2 Experiments on The Rfds Training and Classification with Best 10 IQMs

Table 6: Training results with best 10 IQMs.

| <b>Best 10 IQMs</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, AND SC |     |     |      |
|-----------------------------------------------------------------------------|-----|-----|------|
| Classifier/Sample No.                                                       | FFR | FGR | HTER |
| LDA/100 Images                                                              | 4%  | 24% | 14%  |
| QDA/100 Images                                                              | 4%  | 12% | 8%   |
| Linear SVM/100 Images                                                       | 2%  | 0%  | 1%   |
| Quadratic SVM/100 Images                                                    | 0%  | 0%  | 0%   |
| Logistic Regression/100 Images                                              | 0%  | 0%  | 0%   |
| Fine KNN/100 Images                                                         | 0%  | 0%  | 0%   |

The results listed in Table 6, are generated with the use of 10 IQMs, there were 50 real and 50 fake fingerprint images inputted to the system to train, the detailed results of the 100 samples of fingerprints using 6 classifiers are shown in Appendix C-3, Figures 50-55.

Table 7: Classification results with best 10 IQMs.

| <b>Best 10 IQMs:</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, AND SC |       |       |       |
|------------------------------------------------------------------------------|-------|-------|-------|
| Classifier/Sample No.                                                        | FFR   | FGR   | HTER  |
| LDA/1000 Images                                                              | 9%    | 18.8% | 13.9% |
| QDA/1000 Images                                                              | 13.4% | 13.6% | 13.5% |
| Linear SVM/1000 Images                                                       | 8.8%  | 7.4%  | 8.1%  |
| Quadratic SVM/1000 Images                                                    | 3%    | 9.4%  | 6.2%  |
| Logistic Regression/1000 Images                                              | 14.6% | 12.2% | 13.4% |
| Fine KNN/1000 Images                                                         | 3.8%  | 13.2% | 8.5%  |

Table 7, shows the classification results with 10 IQMs for 500 real and 500 fake fingerprint images inputted to the system to classify. The results of the 1000 samples of fingerprint classification are shown in Appendix C-4, Figures 56–61.

#### 4.5.3 Experiments on The RFDS Training and Classification with Best 5 IQMs

Table 8: Training results with best 5 IQMs.

| <b>Best 5 IQMs:</b><br>SME, PSNR, SNR, GPE, AND RAMD |            |            |             |
|------------------------------------------------------|------------|------------|-------------|
| <b>Classifier/Sample No.</b>                         | <b>FFR</b> | <b>FGR</b> | <b>HTER</b> |
| LDA/100 Images                                       | 4%         | 16%        | 10%         |
| QDA/100 Images                                       | 4%         | 6%         | 5%          |
| Linear SVM/100 Images                                | 2%         | 0%         | 1%          |
| Quadratic SVM/100 Images                             | 2%         | 0%         | 1%          |
| Logistic Regression/100 Images                       | 0%         | 0%         | 0%          |
| Fine KNN/100 Images                                  | 0%         | 0%         | 0%          |

The results listed in Table 8, are obtained with the use of 5 IQMs, there were 50 real and 50 fake fingerprint images inputted to the system to train, the detailed results of the 100 samples of fingerprint using 6 classifiers are shown in Appendix C-5, Figures 62-67.

Table 9: Classification results with best 5 IQMs.

| <b>Best 5 Iqms:</b><br>SME, PSNR, SNR, GPE, AND RAMD |            |            |             |
|------------------------------------------------------|------------|------------|-------------|
| Classifier/Sample No.                                | <b>FFR</b> | <b>FGR</b> | <b>HTER</b> |
| LDA/1000 Tested Fingerprint Images                   | 10.4%      | 14.6%      | 12.5%       |
| QDA/1000 Images                                      | 13.8%      | 9.2%       | 11.5%       |
| Linear SVM/1000 Images                               | 8.8%       | 6.4%       | 7.6%        |
| Quadratic SVM1000 Images                             | 7.8%       | 6.2%       | 7%          |
| Logistic Regression/1000 Ima+Ges                     | 6.2%       | 6.8%       | 6.5%        |
| Fine KNN/1000 Images                                 | 6.4%       | 8.6%       | 7.5%        |

Table 9, shows the classification results with 5 IQMs for 500 real and 500 fake fingerprint images inputted to the system to classify, the detailed results of the 1000 samples of fingerprints tested for classification are shown in Appendix C-6, Figures 68-73. 4.3.4.

#### 4.5.4 Experiments on The RFDS Training and Classification With 25 IQMs

Table 10: Training results with 25 IQMs.

| <b>25 IQMs</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, SC, TED, NK, MSE, GME, AD, LMSE, MAS, MAMS, HLFREQIQ, BIQI, SSIM, RRED, JPEG, VIF, AND NIQE |            |            |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------|-------------|
| Classifier/Sample No.                                                                                                                                       | <b>FFR</b> | <b>FGR</b> | <b>HTER</b> |
| LDA/100 Images                                                                                                                                              | 4%         | 20%        | 12%         |
| QDA/100 Images                                                                                                                                              | 2%         | 0%         | 1%          |
| Linear SVM/100 Images                                                                                                                                       | 0%         | 2%         | 1%          |
| Quadratic SVM/100 Images                                                                                                                                    | 0%         | 0%         | 0%          |
| Logistic Regression/100 Images                                                                                                                              | 0%         | 0%         | 0%          |
| Fine KNN/100 Images                                                                                                                                         | 0%         | 0%         | 0%          |

The results listed in Table 10 are generated from ATV database for 50 real and 50 fake fingerprint images inputted to the system to train. The detailed results of the 100 samples of fingerprints using 25 IQMs using 6 classifiers are shown in Appendix C-7, Figures 74 -78.

Table 11: Classification results with 25 IQM's.

| <b>25 IQMs</b><br>SME, PSNR, SNR, GPE, RAMD, MD, TCD, NAE, SPE, SC, TED, NK, MSE, GME, AD, LMSE, MAS, MAMS, HLFREQIQ, BIQI, SSIM, RRED, JPEG, VIF, AND NIQE |            |            |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------|-------------|
| Classifier/Sample No.                                                                                                                                       | <b>FFR</b> | <b>FGR</b> | <b>HTER</b> |
| LDA/1000 Images                                                                                                                                             | 5.4%       | 0%         | 2.7%        |
| QDA/1000 Images                                                                                                                                             | 19%        | 0%         | 9.5%        |
| Linear SVM/1000 Images                                                                                                                                      | 2.6%       | 0%         | 1.3%        |
| Quadratic SVM/1000 Images                                                                                                                                   | 0.6%       | 0%         | 0.3%        |
| Logistic Regression/1000 Images                                                                                                                             | 1%         | 0%         | 0.5%        |
| Fine KNN/1000 Images                                                                                                                                        | 1.2%       | 0%         | 0.6%        |

Table 11, shows the classification results with 25 IQM's for 500 real and 500 fake fingerprint images inputted to the system to classify. The detailed results of the 1000 samples of fingerprints for classification are shown in Appendix C-8, Figures 79-84.

#### 4.5.5 Comparision of the RFDS Performance With 5, 10, 15, And 25 Iqms Used

Table 12: Result of comparison HTER (%) between our RFDS and RFDS [6].

| Result of Comparison For 5, 10, 15 And 25 IQMs |         |       |       |      |      |        |
|------------------------------------------------|---------|-------|-------|------|------|--------|
| Classifiers                                    |         |       |       |      |      |        |
| RFDS                                           | FINEKNN | QDA   | LDA   | QSVM | LSVM | LOGREG |
| 25                                             | N/A     | 12.8% | N/A   | N/A  | N/A  | N/A    |
| 25 IQMs                                        | 0.6%    | 9.5%  | 2.7%  | 0.3% | 1.3% | 0.5%   |
| 15 IQMs                                        | 8.1%    | 14.8% | 13.5% | 5.8% | 8%   | 8.5%   |
| 10 IQMs                                        | 8.5%    | 13.5% | 13.9% | 6.2% | 8.1% | 13.4%  |
| 5 IQMs                                         | 7.5%    | 11.5% | 12.5% | 7%   | 7.6% | 6.5%   |

Table 11, shows the comparison between RFDS performances with the usage of 25, 15, 10, 5 IQMs. We can clearly see that performance of our RFDS with 25 IQMs has better performance than the rest of the RFDS. Table 11, also shows that 10 and 15 IQMs RFDS are very similar to each other, their hter results are very close not like the RFDS with 5 IQMs where the result of its HTER has minimum 12% gap from the other RFDS. From Table 11, we can say that the best performance from the RFDS in the list is the one with 25 IQMs, because it's HTER score is the minimum one with 0.3%, and the worst RFDS is the one with the 5 IQMs which has the maximum HTER score with 29.7%.

#### 4.6 Summary

We show the experiment setup of our RFDS, we also compared four of our RFDS using 5, 10, 15, and 25 IQMs with RFDS [6] which has used 25 IQMS. Our RFDS experiment was implemented using atv database [33], 100 fingerprint images were used for training process, and 1000 fingerprints were used for detection process. With the exclusion of our RFDS that used 5 IQMs, we can summarize that with less IQMs than RFDS [6], our system still managed to give us better results, listed all in table 11.

## Chapter 5

### CONCLUSION

In our thesis we analyzed similar to the “State-of-The-Art” RFDS systems; some of them used different image quality measures (IQMs), and also different types of classification process. The results of these RFDS systems in detecting fake fingerprint images still had errors and were not 100% accurate. These existing errors of detecting fingerprint image made us interested in implementing a RFDS system with high performance and fewer errors. In our studies we explained step by step procedure of how to implement our RFDS, implementation of Gaussian filter to the input image, implementation of the IQMs, implementation of the classification process, and implementation of the method of defining the best IQMs. All these implementations were done using MATLAB r2015b. We also show our experimental setup used for investigation of our RFDS. We compared our RFDS (using 5,10, 15, and 25 IQMs) with state of the art RFDS (using 25 IQMs). Our best RFDS hter result is 0.3% using 25 IQMs, and state oft he art RFDS hter was 12.8% using also the same 25 IQMs, which indicates that our RFDS performance is overall much better than the state of the art” RFDS even though in some experiments we used a smaller number of IQMs in our system compared to the state of the art RFDS.

## REFERENCES

- [1] Omran, S. S., & Salih, M. A. (2014). Design and Implementation of Multi-Model Biometric Identification System. *International Journal of Computer Applications*, 99(15), 14-21.
- [2] Mudholkar, S. S., Shende, P. M., & Sarode, M. V. (2012). Biometrics authentication technique for intrusion detection systems using fingerprint recognition. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 2(1), 57-65.
- [3] Srivastava, H. (2013). A comparison based study on biometrics for human recognition. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 15(1), 22-29.
- [4] Image Quality. (n.d.). In Wikipedia. Retrieved October 28, 2019, from [https://en.wikipedia.org/wiki/Image\\_quality#image\\_quality\\_metrics](https://en.wikipedia.org/wiki/Image_quality#image_quality_metrics).
- [5] Naim, N. F., Yassin, A. I. M., Zamri, W. M. A. W., & Sarnin, S. S. (2011, March). Mysql Database for storage of fingerprint data. In *2011 UkSim 13th International Conference on Computer Modelling and Simulation* (pp. 293-298). IEEE.
- [6] Galbally, J., Marcel, S., & Fierrez, J. (2013). Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition. *IEEE transactions on image processing*, 23(2), 710-724.

- [7] I. Avcibaş, I., Sankur, B., & Sayood, K. (2002). Statistical evaluation of image quality measures. *Journal of Electronic imaging*, 11(2), 206-223.
  
- [8] Huynh-Thu, Q., & Ghanbari, M. (2008). Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, 44(13), 800-801.
  
- [9] Peak signal-to-noise ratio. In Wikipedia. Retrieved October 7, 2019, from [https://en.wikipedia.org/wiki/Peak\\_signal-to-noise\\_ratio](https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio).
  
- [10] Yao, S., Lin, W., Ong, E., & Lu, Z. (2005, September). Contrast signal-to-noise ratio for image quality assessment. In *IEEE International Conference on Image Processing 2005* (Vol. 1, pp. I-397). IEEE.
  
- [11] Eskicioglu, A. M., & Fisher, P. S. (1995). Image quality measures and their performance. *IEEE Transactions on communications*, 43(12), 2959-2965.
  
- [12] Martini, M. G., Hewage, C. T., & Villarini, B. (2012). Image quality assessment based on edge preservation. *Signal Processing: Image Communication*, 27(8), 875-882.
  
- [13] Fourier Transform. (n.d.). In American Psychological Association. Retrieved October 27, 2019, from <https://blog.apastyle.org/apastyle/urls/>.
  
- [14] Nill, N. B., & Bouzas, B. (1992). Objective image quality measure derived from digital image power spectra. *Optical engineering*, 31(4), 813-826.

- [15] Liu, A., Lin, W., & Narwaria, M. (2011). Image quality assessment based on gradient similarity. *IEEE Transactions on Image Processing*, 21(4), 1500-1512.
- [16] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 600-612.
- [17] IQM Code. (n.d.). In Laboratory for Image & Video Engineering. Retrieved October 12, 2019, from <http://live.ece.utexas.edu/research/quality/index.htm>.
- [18] Sheikh, H. R., & Bovik, A. C. (2006). Image information and visual quality. *IEEE Transactions on image processing*, 15(2), 430-444.
- [19] Soundararajan, R., & Bovik, A. C. (2011). RRED indices: Reduced reference entropic differencing for image quality assessment. *IEEE Transactions on Image Processing*, 21(2), 517-526.
- [20] Zhu, X., & Milanfar, P. (2009, July). A no-reference sharpness metric sensitive to blur and noise. In *2009 International Workshop on Quality of Multimedia Experience* (pp. 64-69). IEEE.
- [21] Wang, Z., Sheikh, H. R., & Bovik, A. C. (2002, September). No-reference perceptual quality assessment of JPEG compressed images. In *Proceedings. International Conference on Image Processing* (Vol. 1, pp. I-I). IEEE.

- [22] Moorthy, A. K., & Bovik, A. C. (2010). A two-step framework for constructing blind image quality indices. *IEEE Signal processing letters*, 17(5), 513-516.
- [23] Mittal, A., Soundararajan, R., & Bovik, A. C. (2012). Making a “completely blind” image quality analyzer. *IEEE Signal Processing Letters*, 20(3), 209-212.
- [24] Linear discriminant analysis. In Wikipedia. Retrieved October 10, 2019, from [https://en.wikipedia.org/wiki/Linear\\_discriminant\\_analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis).
- [25] Quadratic classifier. In Wikipedia. Retrieved October 5, 2019, from [https://en.wikipedia.org/wiki/Quadratic\\_classifier](https://en.wikipedia.org/wiki/Quadratic_classifier).
- [26] LivDet Databases. (n.d.). In Liveness Detection Competition. Retrieved October 20, 2019, from <http://livdet.org/registration.php>
- [27] Galbally, J., Alonso-Fernandez, F., Fierrez, J., & Ortega-Garcia, J. (2009, September). Fingerprint liveness detection based on quality measures. In *2009 First IEEE International Conference on Biometrics, Identity and Security (BIDS)* (pp. 1-8). IEEE.
- [28] Galbally, J., Alonso-Fernandez, F., Fierrez, J., & Ortega-Garcia, J. (2012). A high performance fingerprint liveness detection method based on quality related features. *Future Generation Computer Systems*, 28(1), 311-321.

- [29] Lourde, M., & Khosla, D. (2010). Fingerprint Identification in Biometric Security Systems. *International Journal of Computer and Electrical Engineering*, 2(5), 852.
- [30] Arunalatha, G., & Ezhilarasan, M. (2015). Fingerprint spoof detection using quality features. *Int. J. Secur. Its Appl*, 9(10), 83-94.
- [31] Mohammed, M. O. (2017). *Parametric Real Face Images Detection System (RFIDS) Using Multiple Classifiers* (Master's thesis, Eastern Mediterranean University (EMU)-Doğu Akdeniz Üniversitesi (DAÜ)).
- [32] Gaussian Filter. (n.d.). In Wikipedia. Retrieved October 14, 2019, from [https://en.wikipedia.org/wiki/Gaussian\\_filter](https://en.wikipedia.org/wiki/Gaussian_filter).
- [ 33] DESCRIPTION OF ATVS-FFp DB. (n.d.). In Biometrics Ideal Test. Retrieved October 1, 2019, from <http://biometrics.idealtest.org/dbDetailForUser.do?id=11>

## **APPENDICES**

## **Appendix A: Feature Extraction Code:**

### **1. MSE Function**

```
Function Iqm_Mse = Mean_Sqr_Error(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

[M N] = Size(Rel_Img);

Error = Rel_Img - Enhcd_Img;

Iqm_Mse = Sum(Sum(Error .* Error)) / (M * N);
```

### **2. PSNR Function**

```
Function Iqm_Psnr = Pk_Signalt_Nse_Ratio(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

[M N] = Size(Rel_Img);

Error = Rel_Img - Enhcd_Img;

Iqm_Mse = Sum(Sum(Error .* Error)) / (M * N);

If(Iqm_Mse > 0)

    Iqm_Psnr = 10*Log10((255.*255)/Iqm_Mse);

Else

    End
```

### 3. SNR Function

```
Function Iqm_Snr = Signal_Nse_Ratio(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

[M, N] = Size(Rel_Img);

Error = Rel_Img - Enhcd_Img;

Iqm_Mse = Sum(Sum(Error .* Error)) / (M * N); If(Iqm_Mse > 0)

Iqm_Snr = 10*Log10((Sum(Sum(Rel_Img.*Rel_Img)))/(M .* N .* Iqm_Mse)); Else

Iqm_Snr = 99; End
```

### 4. SC Function

```
Function Iqm_Sc = Struct_Cntnt(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

Iqm_Sc = Sum(Sum(Rel_Img .* Rel_Img)) ./ Sum(Sum(Enhcd_Img .* Enhcd_Img));
```

### 5. MD Function

```
Function Iqm_Md = Max_Diff(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

Error = Rel_Img - Enhcd_Img;

Iqm_Md = Max(Max(Abs(Error)));
```

## 6.AD Function

```
Function Ad = Avrge_Diff(Rel_Img, Enhcd_Img)
```

```
Rel_Img = Double(Rel_Img);
```

```
Enhcd_Img = Double(Enhcd_Img);
```

```
[M N] = Size(Rel_Img);
```

```
Error = Rel_Img - Enhcd_Img;
```

```
Iqm_Ad = Sum(Sum(Error)) / (M * N);
```

## 7. NAE Function

```
Function Iqm_Nae = Normalized_Abs_Error(Rel_Img, Enhcd_Img)
```

```
Rel_Img = Double(Rel_Img);
```

```
Enhcd_Img = Double(Enhcd_Img);
```

```
Error = Rel_Img - Enhcd_Img;
```

```
Iqm_Nae = Sum(Sum(Abs(Error))) / Sum(Sum(Abs(Rel_Img)));
```

## 8. RAMD Function

```
Function Iqm_Ramd = Ravg_Md(Rel_Img, Enhcd_Img)
```

```
Rel_Img = Double(Rel_Img);
```

```
Enhcd_Img = Double(Enhcd_Img);
```

```
Error = Rel_Img - Enhcd_Img;
```

```
Error=Abs(Error);
```

```
A1=Error(:);
```

```

B=Flipud(Unique(Sort(A1)));
Resultatt=B(1:10);
Resultat_1=Resultatt(1:10,:);
R = 10;
Iqm_Ramd = Sum((Abs(Resultat_1)))/R;

```

## 9. NCC Function

```

Function Iqm_Ncc = Normalized_Crs_Correlation(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);
Enhcd_Img = Double(Enhcd_Img);
Iqm_Ncc = Sum(Sum(Rel_Img .* Enhcd_Img)) ./ Sum(Sum(Rel_Img .* Rel_Img));

```

## 10. TED Function

```

Function Iqm_Ted = Total_Edg_Dif(Rel_Img, Enhcd_Img)

Rel_Img=Edge(Rel_Img,'Sobel');
Figure, Imshow(Rel_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img=Edge(Enhcd_Img,'Sobel');
Figure, Imshow(Rel_Img)

Enhcd_Img = Double(Enhcd_Img);

[M, N] = Size(Rel_Img);

Error = Rel_Img - Enhcd_Img;

Iqm_Ted = Sum(Sum(Abs(Error))) / (M * N);

```

## 11.TCD Function

```
Function Iqm_Tcd = Total_Crner_Diff(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

Ncr_Orig = Detectharrisfeatures(Rel_Img);

Imshow(Rel_Img);

Hold On;

Plot(Ncr_Orig);

Ncr_Orig = Length(Ncr_Orig);

Ncr_Dist = Detectharrisfeatures(Enhcd_Img); Imshow(Enhcd_Img); Hold On;

Plot(Ncr_Dist);

Ncr_Dist = Length(Ncr_Dist);

Max_1= Max(Ncr_Orig,Ncr_Dist);

Iqm_Tcd =(Abs(Ncr_Orig-Ncr_Dist))/Max_1;
```

## 12. SPE Function

```
Function Iqm_Spe = Spectral_Ph_Error(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

Fft_A = Fft2(Double(Rel_Img));

Argfft_A =Angle(Fft_A);

Fft_B = Fft2(Double(Enhcd_Img));

Argfft_B =Angle(Fft_B);
```

```

[M , N] = Size(Rel_Img);

Error=Abs(Argfft_A)-Abs(Argfft_B);

Iqm_Spe =Sum(Sum( Error.*Error)) / (M * N);

```

### 13. SME Function

```

Function Iqm_Sme = Spectral_Mg_Error(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

Fft_A = Fft2(Double(Rel_Img));

Z1rr=Real(Fft_A);

Z1ii=Imag(Fft_A);

Fft_A1=Sqrt((Z1rr.*Z1rr)+(Z1ii.*Z1ii));

Fft_B = Fft2(Double(Enhcd_Img));

Z2rr=Real(Fft_B);

Z2ii=Imag(Fft_B);

Fft_B1=Sqrt((Z2rr.*Z2rr)+(Z2ii.*Z2ii));

[M , N] = Size(Rel_Img); Error=Abs(Fft_A1)-Abs(Fft_B1);

Iqm_Sme =Sum(Sum( Error.*Error)) / (M * N);

```

### 14. GPE Function

```

Function Iqm_Gpe = Gradient_Ph_Error(Rel_Img, Enhcd_Img)

Rel_Img = Double(Rel_Img);

Enhcd_Img = Double(Enhcd_Img);

```

```

[Fx1,Fy1] = Gradient(Rel_Img);
Zz=Complex(Fx1,Fy1);
Arggmagorig = Angle(Zz);
[Fx1,Fy1] = Gradient(Enhcd_Img);
Yy=Complex(Fx1,Fy1);
Arggma_Dist = Angle(Yy);
[M , N] = Size(Rel_Img);
Error=Abs(Arggmagorig)-Abs(Arggma_Dist);
Iqm_Gpe =Sum(Sum( Error.*Error)) / (M * N)

```

## 15. GME Function

```

Function Iqm_Gme = Gradient_Mg_Error(Rel_Img, Enhcd_Img)
Rel_Img = Double(Rel_Img);
Enhcd_Img = Double(Enhcd_Img);
[Fx1,Fy1] = Gradient(Rel_Img);
Gmgorig=Sqrt((Fx1.*Fx1)+(Fy1.*Fy1)); [Fx1,Fy1] = Gradient(Enhcd_Img);
Gmgdis=Sqrt((Fx1.*Fx1)+(Fy1.*Fy1));
[M , N] = Size(Rel_Img);
Error=Abs(Gmgorig)-Abs(Gmgdis);
Error=Error.*Error;
Iqm_Gme =Sum(Sum( Error)) / (M * N)

```

## 16. MAS Function

```
Function Mas= Meananglesimilarity(Realimg, Enhimg)%Mas  
[M N]= Size(Realimg);  
Alfa2= 2/180 .* (Acosd(Dot(Realimg,Enhimg)/(Norm(Realimg).* Norm(Enhimg))));  
Angle2 = Sum( Alfa2 );  
Mas = 1-Angle2/(M*N);  
End
```

## 17. MAMS Function

```
Function Mams= Meananglemagnitudesimilarity(Realimg, Enhimg)  
[M N]= Size(Realimg);  
Alfa= 2/Pi .* (Acos(Dot(Realimg, Enhimg)/(Norm(Realimg).* Norm(Enhimg))));  
X= 1 - ((1-Alfa)*(1 - (Norm(Realimg-Enhimg)/255)));  
Sum = Sum(X);  
Mams = Sum / (M*N);  
End
```

## 18. LMSE Function

```
Function Lmse= Laplacianmeansquarederror(Realimg, Enhimg)%Lmse  
[M N]= Size(Realimg);  
Op=4*Del2(Realimg);  
Op2=4*Del2(Enhimg);
```

```
Lmse=Sum(Sum((Op-Op2).^2))/Sum(Sum(Op.^2));
```

```
End
```

## **19. SSIM Function**

```
Function Ssim= Structuralsimilarityindex(Realimg, Ehnimg)
```

```
Img1=Double(Realimg);
```

```
Img2=Double(Ehnimg);
```

```
K = [0.01 0.03];
```

```
Window = Fspecial('Gaussian', 11, 1.5);
```

```
L = 255;
```

```
Ssim = Ssim_Index(Img1, Img2, K, Window, L);
```

## **20. VIF Function**

```
Function Vif=Vifvec(Imorg,Imdist);
```

```
% -----Copyright Notice Starts With This Line-----
```

```
% Copyright (C) 2005 The University Of Texas At Austin
```

```
% All Rights Reserved.
```

```
% Permission Is Hereby Granted, Without Written Agreement And Without License
```

```
Or Royalty Fees, To Use, Copy,
```

```
% Modify, And Distribute This Code (The Source Files) And Its Documentation For
```

```
% Any Purpose, Provided That The Copyright Notice In Its Entirety Appear In All
```

```
Copies Of This Code, And The
```

% Original Source Of This Code, Laboratory For Image And Video Engineering (Live,  
[Http://Live.Ece.Utexas.Edu](http://live.ece.utexas.edu))

% At The University Of Texas At Austin (Ut Austin,  
[Http://Www.Utexas.Edu](http://www.utexas.edu)), Is Acknowledged In Any Publication That Reports  
 Research Using This Code. The Research  
 % Is To Be Cited In The Bibliography As:

% H. R. Sheikh And A. C. Bovik, "Image Information And Visual Quality", Ieee  
 Transactions On  
 % Image Processing, (To Appear).

% In No Event Shall The University Of Texas At Austin Be Liable To Any Party For  
 Direct, Indirect, Special, Incidental,  
 % Or Consequential Damages Arising Out Of The Use Of This Database And Its  
 Documentation, Even If The University Of Texas  
 % At Austin Has Been Advised Of The Possibility Of Such Damage.

% The University Of Texas At Austin Specifically Disclaims Any Warranties,  
 Including, But Not Limited To, The Implied  
 % Warranties Of Merchantability And Fitness For A Particular Purpose. The Database  
 Provided Hereunder Is On An "As Is" Basis,  
 % And The University Of Texas At Austin Has No Obligation To Provide  
 Maintenance, Support, Updates, Enhancements, Or Modifications.

% -----Copyright Notice Ends With This Line-----

% This Is An Implementation Of The Algorithm For Calculating The  
 % Visual Information Fidelity (Vif) Measure (May Also Be Known As The Sheikh  
 %-Bovik Index) Between Two Images. Please Refer  
 % To The Following Paper:

```

%H. R. Sheikh And A. C. Bovik "Image Information And Visual Quality"
%Ieee Transactios On Image Processing, In Publication, May 2005.
%Download Manuscript Draft From Http://Live.Ece.Utexas.Edu In The
%Publications Link
%This Implementation Is Slightly Differnet From The One Used To Report
%Results In The Paper Above. The Modification Have To Do With Using More
%Subbands Than Those Used In The Paper, Better Handling Of Image Boundaries,
%And A Window That Automatically Resizes Itself Based On The Scale.
%Report Bugfixes And Comments To Hamid.Sheikh@Ieee.Org
%-----
% Prerequisites: The Steerable Pyramid Toolbox. Available At
% Http://Www.Cns.Nyu.Edu/~Lcv/Software.Html
%Input : (1) Img1: The Reference Image
%      (2) Img2: The Distorted Image (Order Is Important)
%Output: (1) Vif Te Visual Information Fidelity Measure Between The Two Images
%Default Usage:
%  Given 2 Test Images Img1 And Img2, Whose Dynamic Range Is 0-255
%  Vif = Vifvec(Img1, Img2);
%Advanced Usage:
%  Users May Want To Modify The Parameters In The Code.
%  (1) Modify Sigma_Nsq To Find Tune For Your Image Dataset.
%  (2) Mxm Is The Block Size That Denotes The Size Of A Vector Used In The
%  Gsm Model.
%  (3) Subbands Included In The Computation
%=====

```

```

M=3;

Subbands=[4 7 10 13 16 19 22 25];

Sigma_Nsq=0.4;

% Do Wavelet Decomposition. This Requires The Steerable Pyramid. You Can
% Use Your Own Wavelet As Long As The Cell Arrays Org And Dist Contain
% Corresponding Subbands From The Reference And The Distorted Images
% Respectively.

[Pyr,Pind] = Buildspyr(Imorg, 4, 'Sp5filters', 'Reflect1'); % Compute Transform
Org=Ind2wtree(Pyr,Pind); % Convert To Cell Array
[Pyr,Pind] = Buildspyr(Imdist, 4, 'Sp5filters', 'Reflect1');
Dist=Ind2wtree(Pyr,Pind);

% Calculate The Parameters Of The Distortion Channel
[G_All,Vv_All]=Vifsub_Est_M(Org,Dist,Subbands,M);

% Calculate The Parameters Of The Reference Image
[Ssarr, Larr, Cuarr]=Refparams_Vecgsm(Org,Subbands,M);

% Reorder Subbands. This Is Needed Since The Outputs Of The Above Functions
% Are Not In The Same Order

Vvtemp=Cell(1,Max(Subbands));

Ggtemp=Vvtemp;

For(Kk=1:Length(Subbands))

Vvtemp{Subbands(Kk)}=Vv_All{Kk};

Ggtemp{Subbands(Kk)}=G_All{Kk};

End

% Compute Reference And Distorted Image Information From Each Subband

For I=1:Length(Subbands)

```

```

Sub=Subbands(I);

G=Ggtemp{Sub};

Vv=Vvtemp{Sub};

Ss=Ssarr{Sub};

Lambda = Larr(Sub,:),

Cu=Cuarr{Sub};

% How Many Eigenvalues To Sum Over. Default Is All.

Neigvals=Length(Lambda);

% Compute The Size Of The Window Used In The Distortion Channel Estimation,

And Use It To Calculate The Offset From Subband Borders

% We Do This To Avoid All Coefficients That May Suffer From Boundary

% Effects

Lev=Ceil((Sub-1)/6);

Winsize=2^Lev+1; Offset=(Winsize-1)/2;

Offset=Ceil(Offset/M);

% Select Only Valid Portion Of The Output.

G=G(Offset+1:End-Offset,Offset+1:End-Offset);

Vv=Vv(Offset+1:End-Offset,Offset+1:End-Offset);

Ss=Ss(Offset+1:End-Offset,Offset+1:End-Offset);

%Vif

Temp1=0; Temp2=0;

For J=1:Length(Lambda)

Temp1=Temp1+Sum(Sum((Log2(1+G.*G.*Ss.*Lambda(J)./(Vv+Sigma_Nsq))))); %

Distorted Image Information For The I'th Subband

```

```

Temp2=Temp2+Sum(Sum((Log2(1+Ss.*Lambda(J)./(Sigma_Nsq))))); % Reference

Image Information

End

Num(I)=Temp1;

Den(I)=Temp2;

End

% Compute Vif

Vif=Sum(Num)./Sum(Den);

```

## 21. RRED Function

```

Function Rred=Rred_Out(Realimg, Enhimg)

% Total = 5;

Original=Realimg;

Fake= Enhimg;

Rredoutput= Extract_Info(Original, Fake);

[X Y] = Size(Rredoutput);

Sumoutput=0;

For I=1 :X

For J=1:Y

Sumoutput = Sumoutput + Rredoutput(I,J);

End

End

Rred=Norm(Sumoutput);

% End

```

## 22.HLFI Function

```
Function Hlfreqiq= High_Low_Freq_Index(Realimg)

[R,C]=Size(Realimg);

N = C;

Imgfft =Fft(Fft2(Realimg));

Colhalf = (Round(N / 2));

Freqsel = 0.15;

Freqcol = Round(Freqsel * N);

Lowfreqcolhalf = (Round(Freqcol / 2.0));

Fftres = Imgfft; %R# Np.Fft.Fft2(Image)

Fftmag = Abs(Fftres);

Totalenergy = Sum(Sum(Fftmag));

%# Print(Totalenergy)

Lowidx = Colhalf - Lowfreqcolhalf;

Hiidx = Colhalf + Lowfreqcolhalf;

Lowfreqmag = Fftmag(:, Lowidx:Hiidx);

Lowfreqmagtotal = Sum(Sum(Lowfreqmag));

Fftmag(:, Lowidx:Hiidx) = 0;

Highfreqmagtotal =Sum( Sum(Fftmag));

Hlfreqiq = Abs(Lowfreqmagtotal - Highfreqmagtotal) /(Totalenergy);

End
```

## 23. JQI Function

Function [Score B A Z] = Jpeg\_Quality\_Score(Img)

%=====

%Copyright (C) 2002 The University Of Texas At Austin

%All Rights Reserved.

%This Program Is Free Software; You Can Redistribute It And/Or Modify

%It Under The Terms Of The Gnu General Public License As Published By

RFDS training and detection structure %(At Your Option) Any Later Version.

%This Program Is Distributed In The Hope That It Will Be Useful,

%But Without Any Warranty; Without Even The Implied Warranty Of

%Merchantability Or Fitness For A Particular Purpose. See The

%Gnu General Public License For More Details.

%The Gnu Public License Is Available In The File License, Or You

%Can Write To The Free Software Foundation, Inc., 59 Temple Place -

%Suite 330, Boston, Ma 02111-1307, Usa, Or You Can Find It On The

%World Wide Web At [Http://Www.Fsf.Org](http://www.fsf.org).

% Author : Zhou Wang

% Version : 1.0

% Modified: Anush Krishna Moorthy

% Version: 1.Buqi

%The Authors Are With The Laboratory For Image And Video Engineering

%(Live), Department Of Electrical And Computer Engineering, The

%University Of Texas At Austin, Austin, Tx.

%

```

%Kindly Report Any Suggestions Or Corrections To Zhouwang@Ieee.Org

%=====

%This Is An Implementation Of The Algorithm For Calculating The Quality
%Score Of Jpeg Compressed Images Proposed By Zhou Wang, Hamid R. Sheikh
%And Alan C. Bovik. Please Refer To The Paper: Zhou Wang, Hamid R. Sheikh
%And Alan C. Bovik, "No-Reference Perceptual Quality Assessment Of Jpeg
%Compressed Images," Submitted To Ieee International Conference On Image
%Processing, Sept. 2002.

%You Can Change This Program As You Like And Use It Anywhere, But Please
%Refer To Its Original Source (Cite Our Paper And Our Web Page At
%Http://Anchovy.Ece.Utexas.Edu/~Zwang/Research/Nr_Jpeg_Quality/Index.Html)

%

%Input : A Test 8bits/Pixel Grayscale Image Loaded In A 2-D Array
%Output: A Quality Score Of The Image. The Score Typically Has A Value
%      Between 1 And 10 (10 Represents The Best Quality, 1 The Worst).

%Usage:

%1. Load The Image, For Example

% Image = imread('Testimage.Jpg');

%2. Call This Function To Calculate The Quality Score:

% Quality_Score = Jpeg_Quality_Score(Image)

%=====

If (Nargin > 1)

Score = -1;

Return;

End

```

```

[M N] = Size(Img);

If (M < 16 | N < 16)

Score = -2;

Return;

End

X = Double(Img);

% Feature Extraction:

% 1. Horizontal Features

D_H = X(:, 2:N) - X(:, 1:(N-1));

B_H = Mean2(Abs(D_H(:, 8:8:8*(Floor(N/8)-1))));

A_H = (8*Mean2(Abs(D_H)) - B_H)/7;

Sig_H = Sign(D_H);

Left_Sig = Sig_H(:, 1:(N-2));

Right_Sig = Sig_H(:, 2:(N-1));

Z_H = Mean2((Left_Sig.*Right_Sig)<0);

% 2. Vertical Features

D_V = X(2:M, :) - X(1:(M-1), :);

B_V = Mean2(Abs(D_V(8:8:8*(Floor(M/8)-1), :)));

A_V = (8*Mean2(Abs(D_V)) - B_V)/7;

Sig_V = Sign(D_V);

Up_Sig = Sig_V(1:(M-2), :);

Down_Sig = Sig_V(2:(M-1), :);

Z_V = Mean2((Up_Sig.*Down_Sig)<0);

% 3. Combined Features

B = (B_H + B_V)/2;

```

$A = (A\_H + A\_V)/2;$

$Z = (Z\_H + Z\_V)/2;$

% Quality Prediction

Alpha = -927.4240; Beta = 850.8986; Gamma1 = 235.4451 ; Gamma2 =  
128.7548; Gamma3 = -341.4790;

Score = Abs(Alpha +  
Beta\*(B.^(Gamma1/10000))\*(A.^(Gamma2/10000))\*(Z.^(Gamma3/10000)));

## 24. BIQI Function

Function [Score B A Z] = Blinimagequalityindex(Rimg)

%=====

%Copyright (C) 2002 The University Of Texas At Austin

%All Rights Reserved.

%This Program Is Free Software; You Can Redistribute It And/Or Modify

%It Under The Terms Of The Gnu General Public License As Published By

%The Free Software Foundation; Either Version 2 Of The License, Or

%(At Your Option) Any Later Version.

%

%This Program Is Distributed In The Hope That It Will Be Useful,

%But Without Any Warranty; Without Even The Implied Warranty Of

%Merchantability Or Fitness For A Particular Purpose. See The

%Gnu General Public License For More Details.

%

%The Gnu Public License Is Available In The File License, Or You

%Can Write To The Free Software Foundation, Inc., 59 Temple Place -  
 %Suite 330, Boston, Ma 02111-1307, Usa, Or You Can Find It On The  
 %World Wide Web At [Http://Www.Fsf.Org](http://www.fsf.org).  
 %  
 % Author : Zhou Wang  
 % Version : 1.0  
 % Modified: Anush Krishna Moorthy  
 % Version: 1.Buqi  
 %  
 %The Authors Are With The Laboratory For Image And Video Engineering  
 %(Live), Department Of Electrical And Computer Engineering, The  
 %University Of Texas At Austin, Austin, Tx.  
 %  
 %Kindly Report Any Suggestions Or Corrections To [Zhouwang@Ieee.Org](mailto:Zhouwang@ieee.org)  
 %  
 %=====

%This Is An Implementation Of The Algorithm For Calculating The Quality  
 %Score Of Jpeg Compressed Images Proposed By Zhou Wang, Hamid R. Sheikh  
 %And Alan C. Bovik. Please Refer To The Paper: Zhou Wang, Hamid R. Sheikh  
 %And Alan C. Bovik, "No-Reference Perceptual Quality Assessment Of Jpeg  
 %Compressed Images," Submitted To Ieee International Conference On Image  
 %Processing, Sept. 2002.  
 %  
 %You Can Change This Program As You Like And Use It Anywhere, But Please  
 %Refer To Its Original Source (Cite Our Paper And Our Web Page At

```

%Http://Anchovy.Ece.Utexas.Edu/~Zwang/Research/Nr_Jpeg_Quality/Index.Html)

%Input : A Test 8bits/Pixel Grayscale Image Loaded In A 2-D Array

%Output: A Quality Score Of The Image. The Score Typically Has A Value

%      Between 1 And 10 (10 Represents The Best Quality, 1 The Worst).

%Usage:

%1. Load The Image, For Example

% Image = imread('Testimage.Jpg');

%2. Call This Function To Calculate The Quality Score:

% Quality_Score = Blinimagequalityindex(Image)

%=====

If (Nargin > 1)

Score = -1;

Return;

End

[M N] = Size(Rimg);

If (M < 16 | N < 16)

Score = -2;

Return;

End

X = Double(Rimg);

% Feature Extraction:

% 1. Horizontal Features

D_H = X(:, 2:N) - X(:, 1:(N-1));

B_H = Mean2(Abs(D_H(:, 8:8:8*(Floor(N/8)-1))));

A_H = (8*Mean2(Abs(D_H)) - B_H)/7;

```

```

Sig_H = Sign(D_H);

Left_Sig = Sig_H(:, 1:(N-2));

Right_Sig = Sig_H(:, 2:(N-1));

Z_H = Mean2((Left_Sig.*Right_Sig)<0);

% 2. Vertical Features

D_V = X(2:M, :) - X(1:(M-1), :);

B_V = Mean2(Abs(D_V(8:8:8*(Floor(M/8)-1), :)));

A_V = (8*Mean2(Abs(D_V)) - B_V)/7;

Sig_V = Sign(D_V);

Up_Sig = Sig_V(1:(M-2), :);

Down_Sig = Sig_V(2:(M-1), :);

Z_V = Mean2((Up_Sig.*Down_Sig)<0);

% 3. Combined Features

B = (B_H + B_V)/2;

A = (A_H + A_V)/2;

Z = (Z_H + Z_V)/2;

% Quality Prediction

Alpha = -927.4240; Beta = 850.8986; Gamma1 = 235.4451 ; Gamma2 =
128.7548; Gamma3 = -341.4790;

Score = Abs(Alpha +
Beta*(B.^(Gamma1/10000))*(A.^(Gamma2/10000))*(Z.^(Gamma3/
10000)));

NIQE Function

Function Quality = Computequality(Im,Blocksize_row,Blocksize_col,...
Blockrow_overlap,Blockcol_overlap,Mu_Prisparam,Cov_Prisparam)

```

```

% Input

% Im - Image Whose Quality Needs To Be Computed

% Blocksize row - Height Of The Blocks In To Which Image Is Divided

% Blocksize col - Width Of The Blocks In To Which Image Is Divided

% Blockrowoverlap - Amount Of Vertical Overlap Between Blocks

% Blockcoloverlap - Amount Of Horizontal Overlap Between Blocks

% Mu_Prisparam - Mean Of Multivariate Gaussian Model

% Cov_Prisparam - Covariance Of Multivariate Gaussian Model

% For Good Performance, It Is Advisable To Use Make The Multivariate Gaussian
Model

% Using Same Size Patches As The Distorted Image Is Divided In To

% Output

%Quality    - Quality Of The Input Distorted Image

% Example Call

%Quality = Computequality(Im,96,96,0,0,Mu_Prisparam,Cov_Prisparam)

% -----

%Number Of Features

% 18 Features At Each Scale

Featnum    = 18;

%-----

%Compute Features

If(Size(Im,3)==3)

Im          = Rgb2gray(Im);

End

Im          = Double(Im);

```

```

[Row Col]    = Size(Im);

Block_Rownum = Floor(Row/Blocksizerow);

Block_Colnum = Floor(Col/Blocksizecol);

Im          = Im(1:Block_Rownum*Blocksizerow,1:Block_Colnum*Blocksizecol);

[Row Col]    = Size(Im);

Block_Rownum = Floor(Row/Blocksizerow);

Block_Colnum = Floor(Col/Blocksizecol);

Im = Im(1:Block_Rownum*Blocksizerow, .
1:Block_Colnum*Blocksizecol);

Window       = Fspecial('Gaussian',7,7/6);

Window       = Window/Sum(Sum(Window));

Scalenum     = 2;

Warning('Off')

Feat = [];

For Itr_Scale = 1:Scalenum

Mu = Imfilter(Im,Window,'Replicate');

Mu_Sq = Mu.*Mu;

Sigma = Sqrt(Abs(Imfilter(Im.*Im,Window,'Replicate') - Mu_Sq));

Strctdis     = (Im-Mu)./(Sigma+1);

Feat_Scale = Blkproc(Strctdis,[Blocksizerow/Itr_Scale Blocksizecol/Itr_Scale],
[Blockrowoverlap/Itr_Scale Blockcoloverlap/Itr_Scale], ...
@Computefeature);

Feat_Scale   = Reshape(Feat_Scale,[Featnum ....
Size(Feat_Scale,1)*Size(Feat_Scale,2)/Featnum]);

Feat_Scale   = Feat_Scale';

```

```

If(Itr_Scale == 1)

Sharpness = Blkproc(Sigma,[Blocksizerow Blocksizecol], ...
[Blockrowoverlap Blockcoloverlap],@Computemean);

Sharpness = Sharpness(:);

End

Feat = [Feat Feat_Scale];

Im =Imresize(Im,0.5);

End

% Fit A Mvg Model To Distorted Patch Features

Distparam = Feat;

Mu_Distparam = Nanmean(Distparam);

Cov_Distparam = Nancov(Distparam);

% Compute Quality

Invcov_Param = Pinv((Cov_Prisparam+Cov_Distparam)/2);

Quality = Sqrt((Mu_Prisparam-Mu_Distparam)* ...
Invcov_Param*(Mu_Prisparam-Mu_Distparam)');

```

## Appendix B: Code for Ranking IQM's

### 1. Mean Squared Error

```
R_Mse = Mse(1:500);  
F_Mse = Mse(501:1000);  
Avg_R_Mse = Mean(R_Mse);  
Disp('Average Of R_Mse :');  
Disp(Avg_R_Mse);  
Avg_F_Mse = Mean(F_Mse);  
Disp('Average Of F_Mse :');  
Disp(Avg_F_Mse);  
Disp('Average Of R_Mse - Average Of F_Mse :');  
Dif_Btwn_Mse_Fmse= Abs(Avg_R_Mse-Avg_F_Mse);  
Disp(Dif_Btwn_Mse_Fmse);  
Disp('-----');
```

Note: Apply The Same Code For Every Iqm

### 2. Ranking the Best 15 IQMs

```
Title={'Mse.'; 'Snr.'; 'Psnr.'; 'Nk.'; 'Ad.'; 'Sc.'; 'Md.'; 'Ramd.'; 'Nae.'; 'Ted.'; 'Spe.'; 'Sme.';  
'Gme.'; 'Gpe.'; 'Tcd.'};  
Best15=[Dif_Btwn_Mse_Fmse;Dif_Btwn_Snr_Fsnr;Dif_Btwn_Psnr_Fpsnr;Dif_Btwn_Nk_Fnk;Dif_Btwn_Ad_Fad;Dif_Btwn_Sc_Fsc;Dif_Btwn_Md_Fmd;Dif_Btwn_Ramd_Framd;Dif_Btwn_Nae_Fnae;Dif_Btwn_Ted_Fted;Dif_Btwn_Spe_Fspe;Dif_Btwn_Sme_Fsme;Dif_Btwn_Gme_Fgme;Dif_Btwn_Gpe_Fgpe;Dif_Btwn_Tcd_Ftcd];  
Tbla = Table(Title,Best15)
```

```
Disp('-----Best 15 Iqm Rank (Classification)-----');
Tblb = Sortrows(Tbla,{'Best15'},{'Descend'})
```

### 3. Calculating FGR, FFR, And HTER Of The RFDS

```
Yfit = Trainedclassifier1.Predictfcn(T)
```

```
Truecount=0;
```

```
Fakecount=0;
```

```
Truecount1=0;
```

```
Fakecount1=0;
```

```
%Real Images Count
```

```
Z=500;
```

```
For B=1:Z;
```

```
If Isequal(Yfit{B},'Real')
```

```
Truecount=Truecount+1;
```

```
Else Isequal(Yfit{B},'Fake')
```

```
Fakecount=Fakecount+1;
```

```
End
```

```
End
```

```
%500 Fake Images Count
```

```
M=1000;
```

```
For Q=501:M;
```

```
If Isequal(Yfit{Q},'Real')
```

```
Truecount1=Truecount1+1;
```

```
Else Isequal(Yfit{Q},'Fake')
```

```
Fakecount1=Fakecount1+1;
```

```

End

End

Disp('500 Real Fingerprint Images Result :');

Rr= ['Number Of Real Images Predict As Real = ',Num2str(Truecount)];

Rf= ['Number Of Real Images Predict As Fake = ',Num2str(Fakecount)];

Disp(Rr);

Disp(Rf);

Disp('500 Fake Fingerprint Images Result :');

Rr1= ['Number Of Fake Images Predict As Fake = ',Num2str(Fakecount1)];

Rf1= ['Number Of Fake Images Predict As Real = ',Num2str(Truecount1)];

Disp(Rr1);

Disp(Rf1);

Fgr=['Fals Genuine Rate(Fgr) = ',Num2str(Truecount1/500)];

Ffr=['Fals Fake Rate(Ffr) = ',Num2str(Fakecount/500)];

Hter=['Half          Total          Error          Rate          (Hter)          =

',Num2str(((Truecount1/500)+(Fakecount/500))/2)];

Disp(Fgr);

Disp(Ffr);

Disp(Hter);

```

## Appendix C: Screenshots and Tables of Our Rfids Results

### 1. Table and Screenshot of Training Results For 100 Fingerprint Images Using 15 IQMs

The following are the screenshots providing results of a dataset with 50 real and fake fingerprint images, with 15 image quality measures calculated for each fingerprint image. Table d.1 shows the result of 15 image quality measures of 50 real fingerprint images and 50 fake fingerprint images used for training. 100 fingerprint images with 15 IQMs results using 6 classifiers.

#### 1. LDA

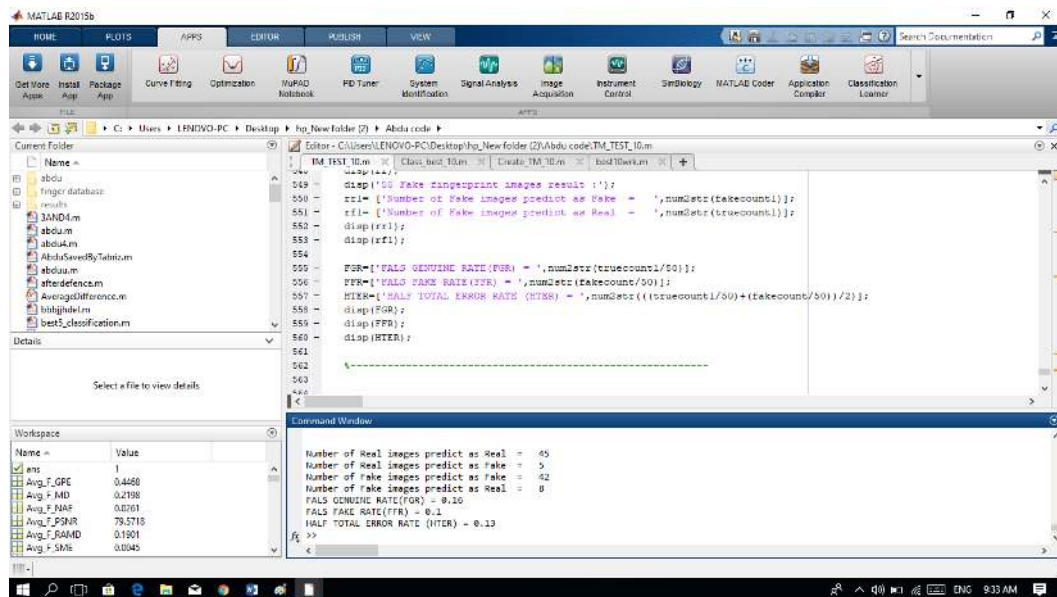


Figure 38: LDA training result of 100 fingerprint images using 15 IQMs.

## 2. QDA

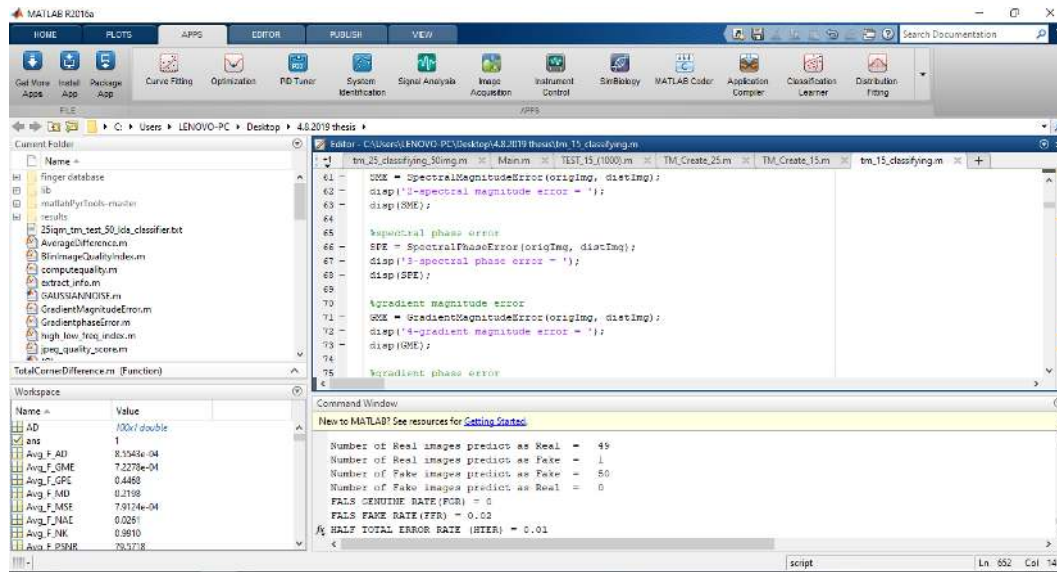


Figure 40: QDA training result of 100 fingerprint images using 15 IQMs.

## 3. Linear SVM

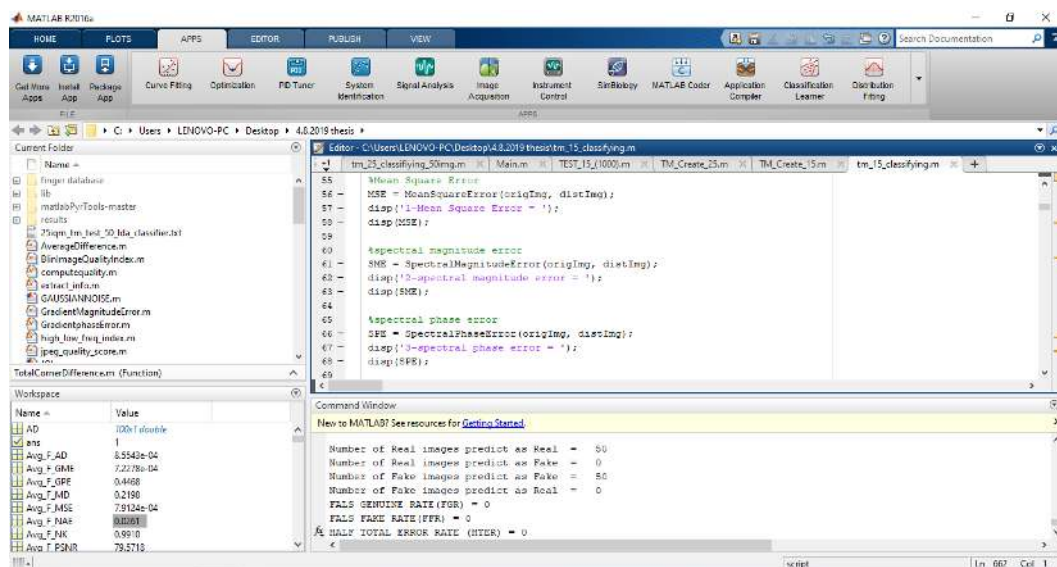


Figure 41: Linear SVM training result of 100 fingerprint images using 15 IQMs.

## 4. Quadratic SVM

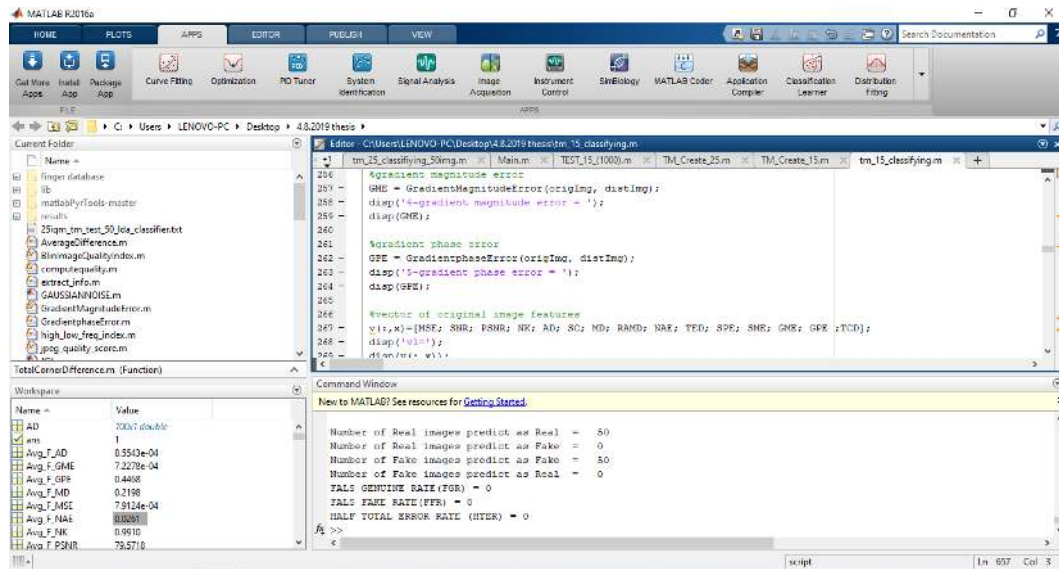


Figure 42: Quadratic SVM training result of 100 fingerprint images using 15 IQMs.

## 5. Fine KNN

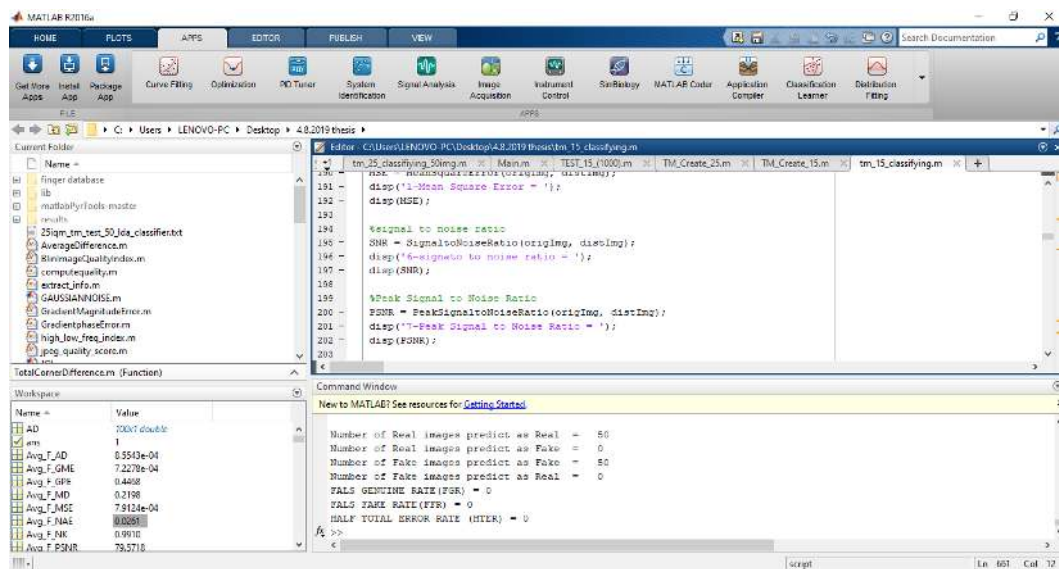


Figure 43: Fine KNN training result of 100 fingerprint images using 15 IQMs.

## 6. Logistic Regression

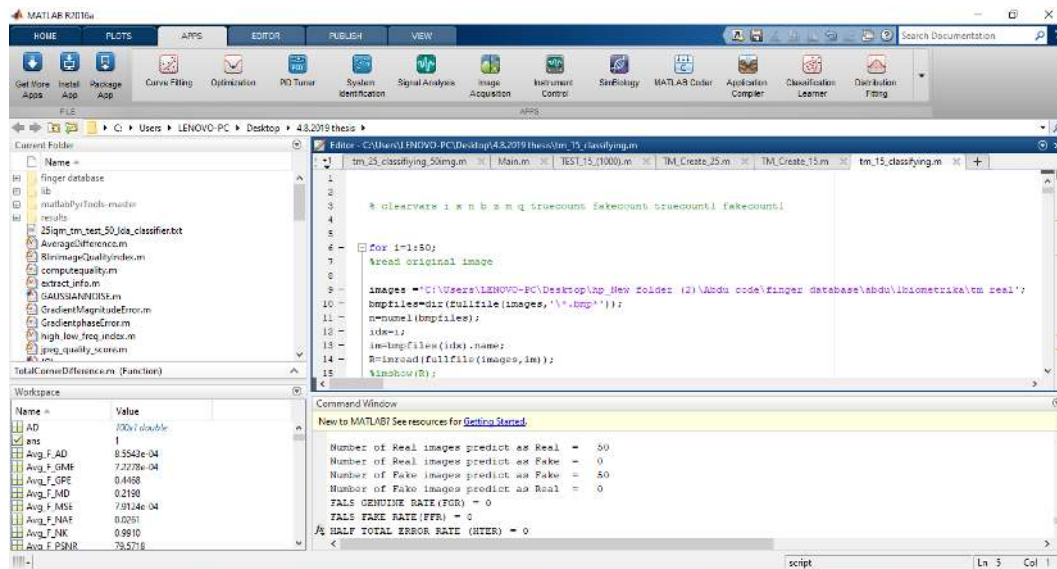


Figure 44: Logistic Regression training result of 100 fingerprint images using 15 IQMs.

## 2. Table and Screenshot of Classification Results For 1000 Fingerprint Images Using 15 IQMs

1000 fingerprint images with 15 IQMs results using 6 classifiers.

### 1. LDA

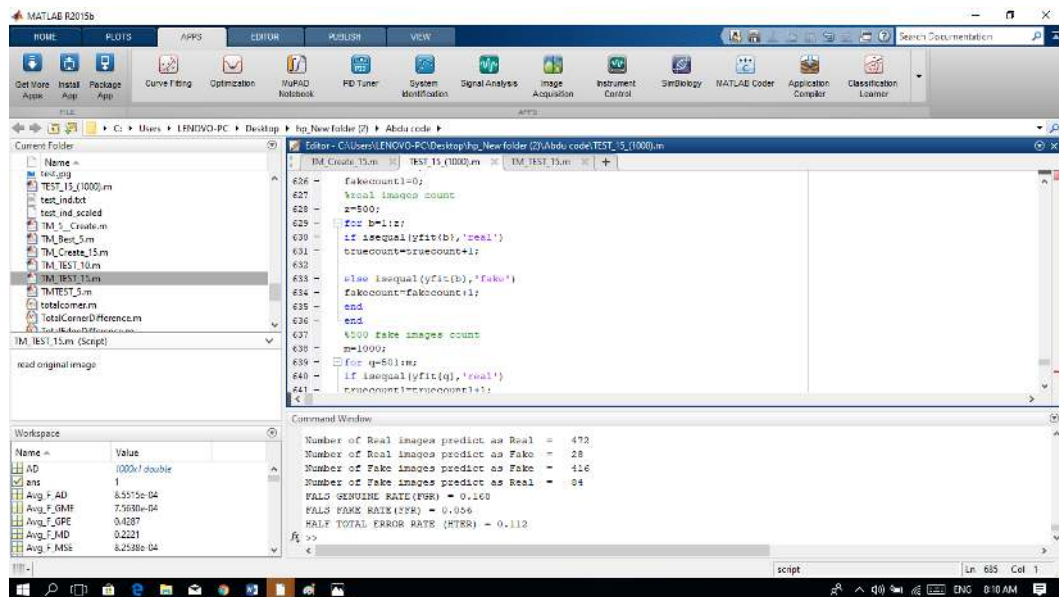


Figure 45: LDA classification result of 1000 fingerprint images using 15 IQMs.

### 2. QDA

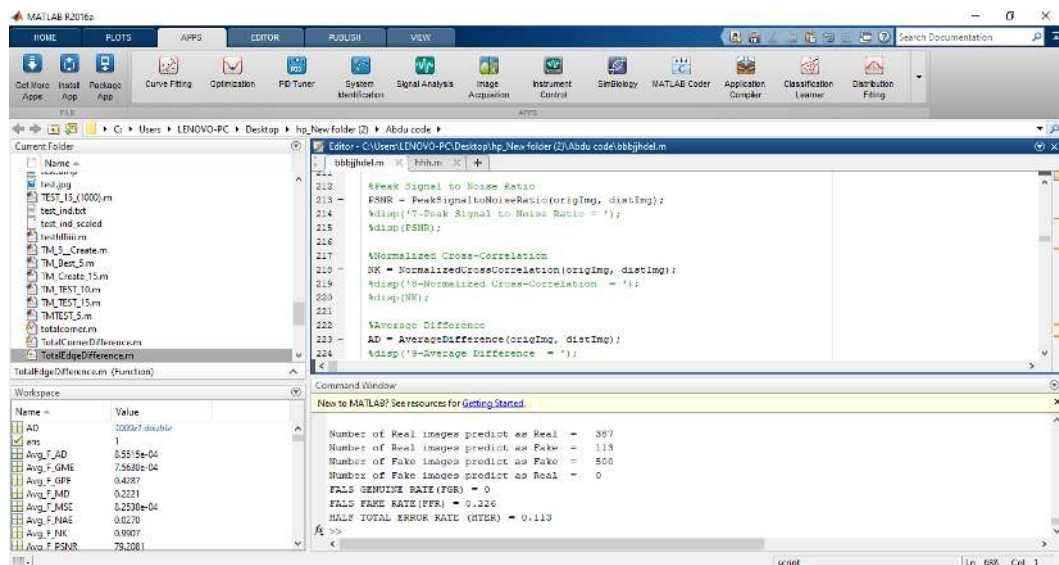


Figure 46: QDA classification result of 1000 fingerprint images using 15 IQMs.

### 3. Linear SVM

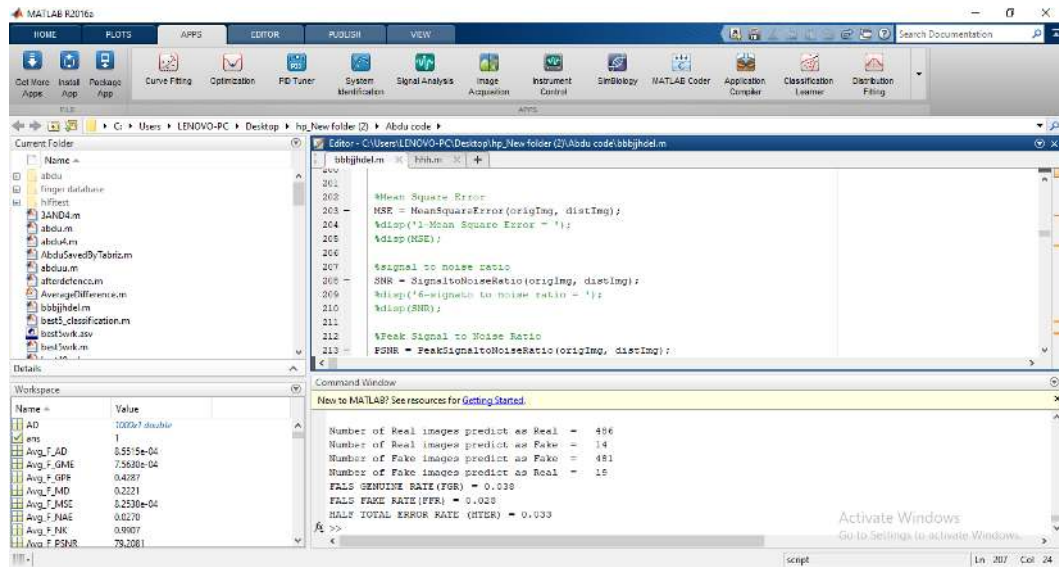


Figure 47: Linear SVM classification result of 1000 fingerprint images using 15 IQMs.

### 4. Quadratic SVM

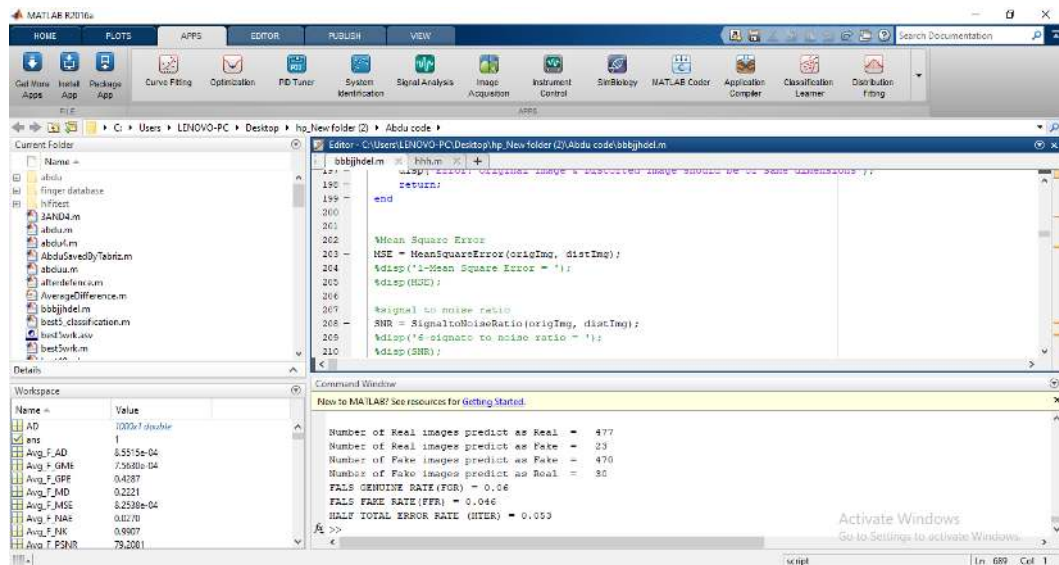


Figure 48: Quadratic SVM classification result of 1000 fingerprint images using 15 IQMs.

## 5. Fine KNN

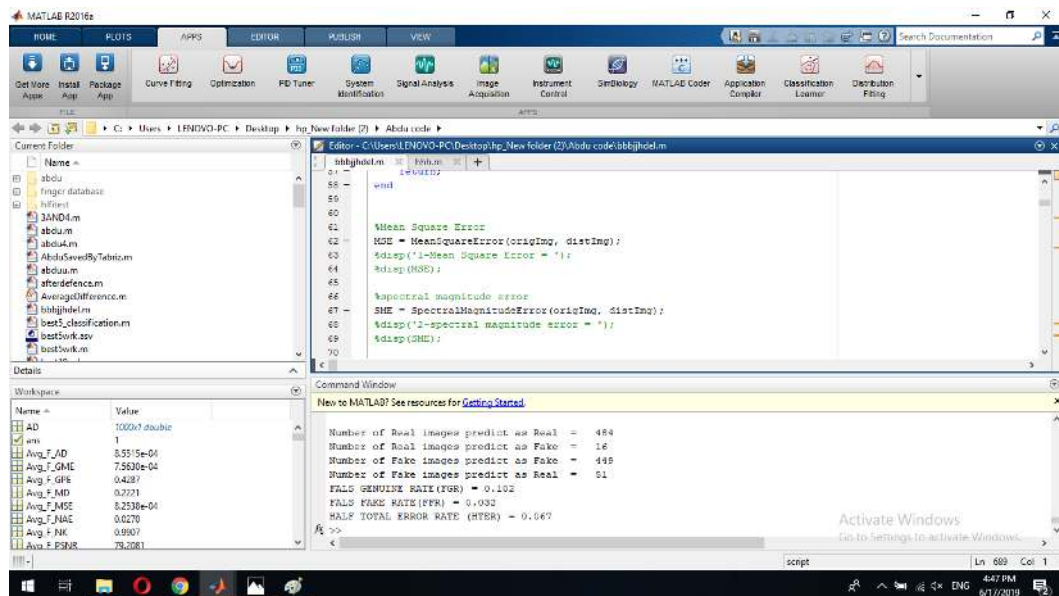


Figure 49: Fine KNN classification result of 1000 fingerprint images using 15 IQMs.

## 6. Logistic Regression

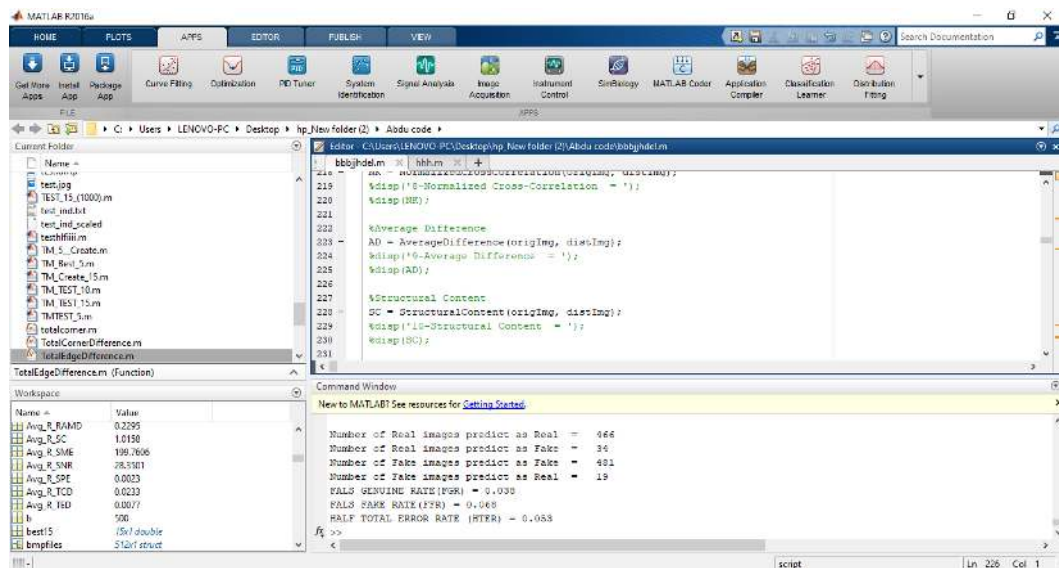


Figure 50: Logistic Regression classification of 1000 fingerprint images using 15 IQMs.

### 3. Table and Screenshot of Training Results For 100 Fingerprint Images Using 10 IQMs

100 fingerprint images with 10 IQMs result using 6 classifiers.

#### 1. LDA

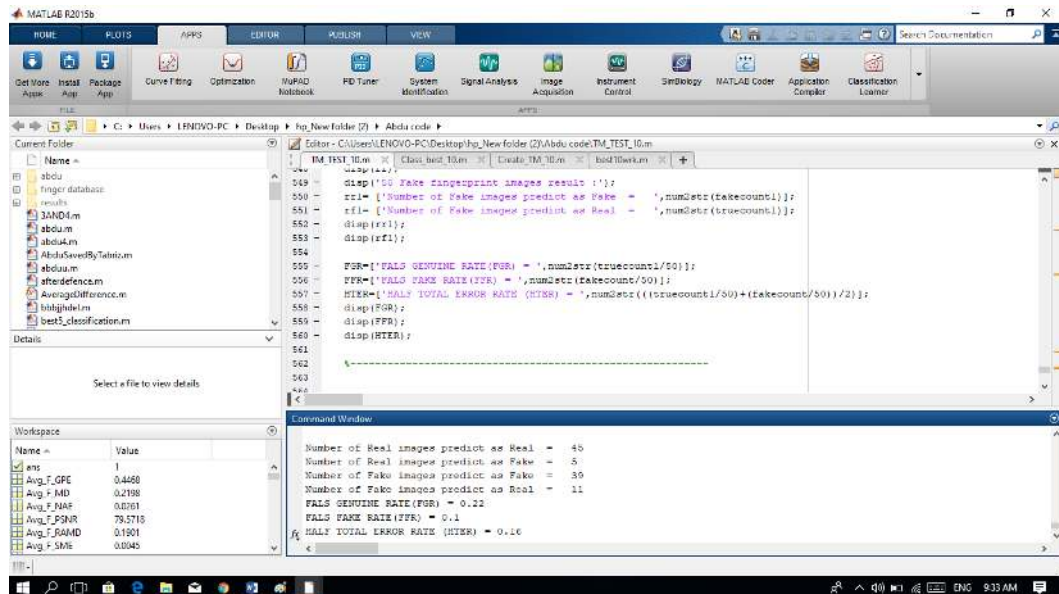


Figure 51: LDA training result of 100 fingerprint images using 10 IQMs.

#### 2. QDA

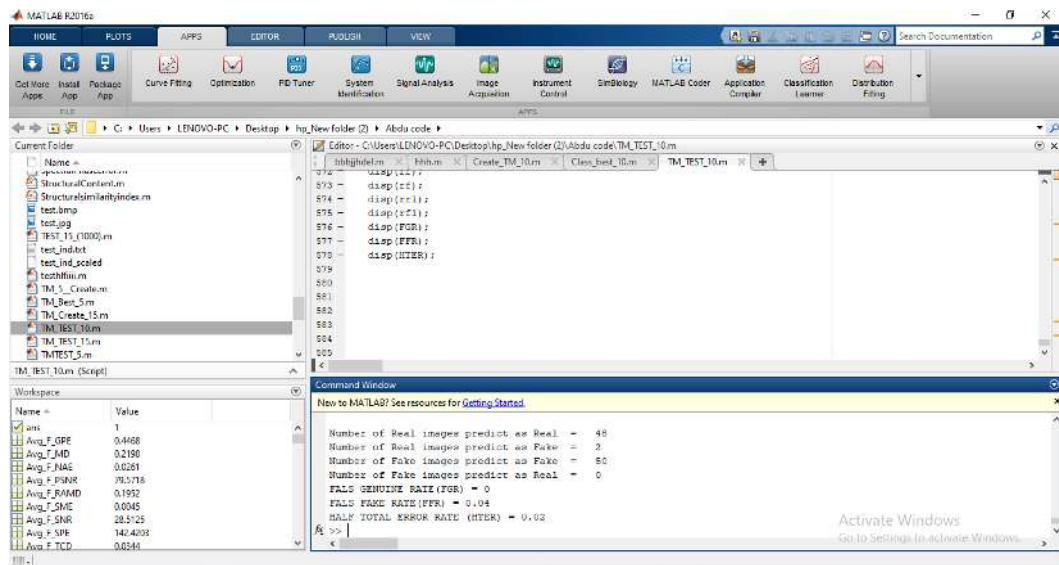


Figure 52: QDA training result of 100 fingerprint images using 10 IQMs.

### 3. Linear SVM

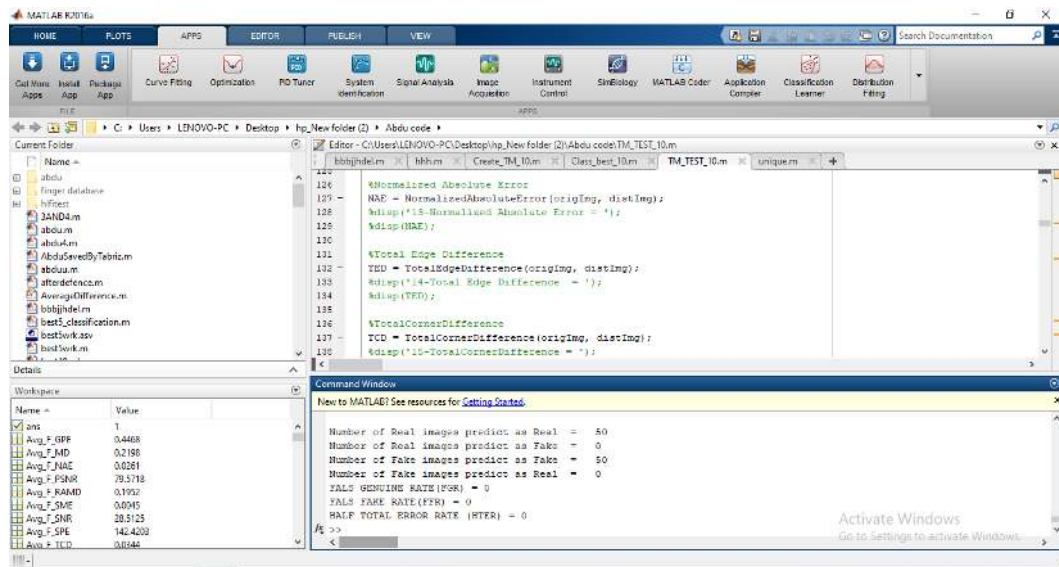


Figure 53: Linear SVM training result of 100 fingerprint images using 10 IQMs.

### 4. Quadratic SVM

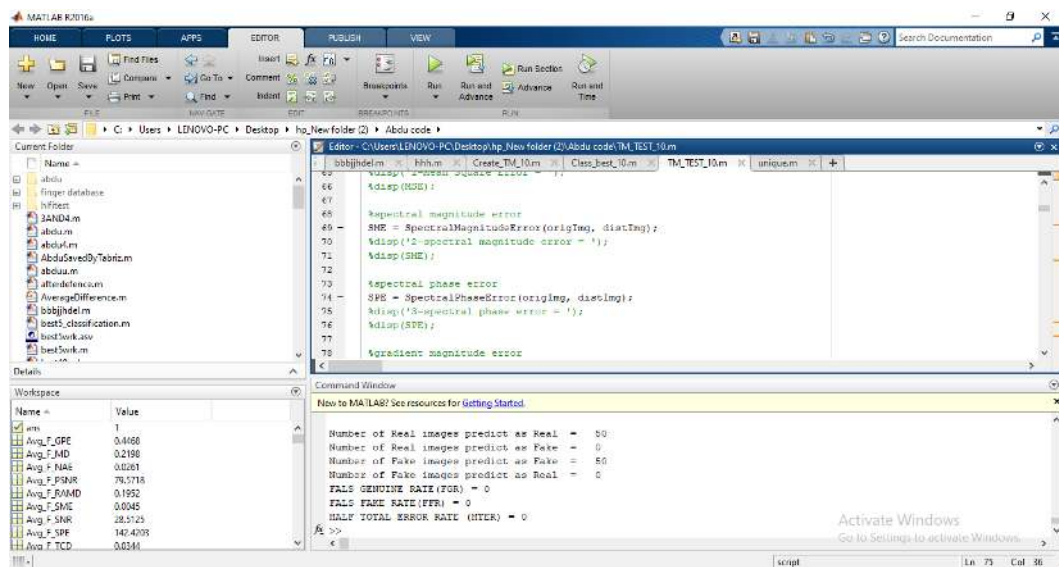


Figure 54: Quadratic SVM training result of 100 fingerprint images using 10 IQMs.

## 5. Fine KNN

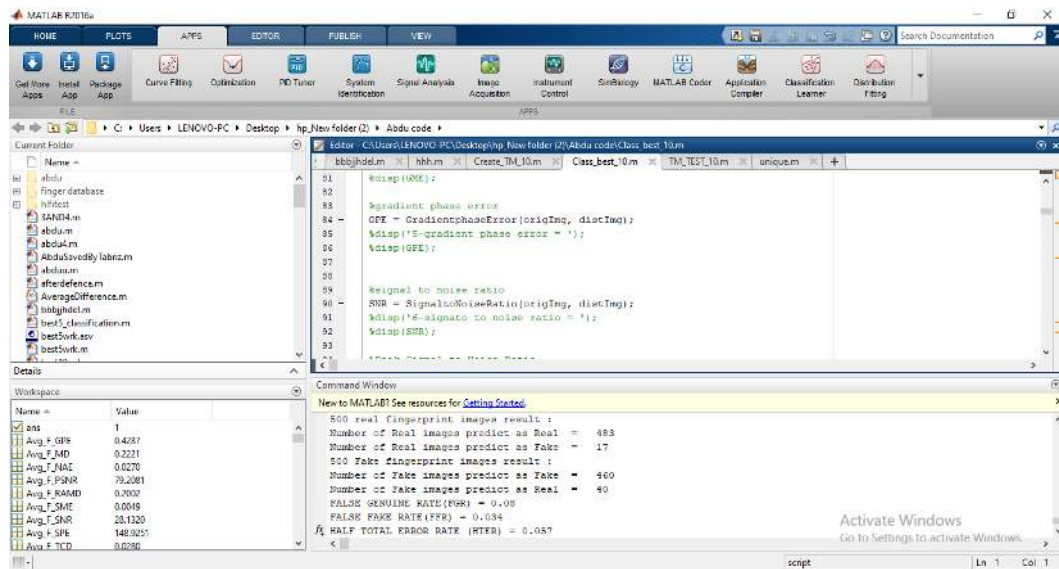


Figure 55: Fine KNN training result of 100 fingerprint images using 10 IQMs.

## 6. Logistic Regression

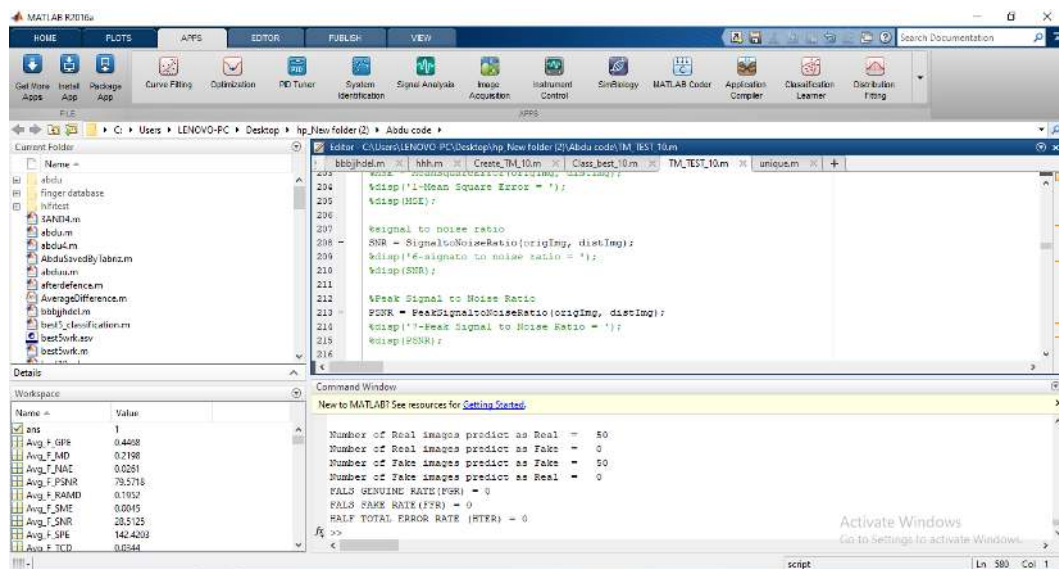
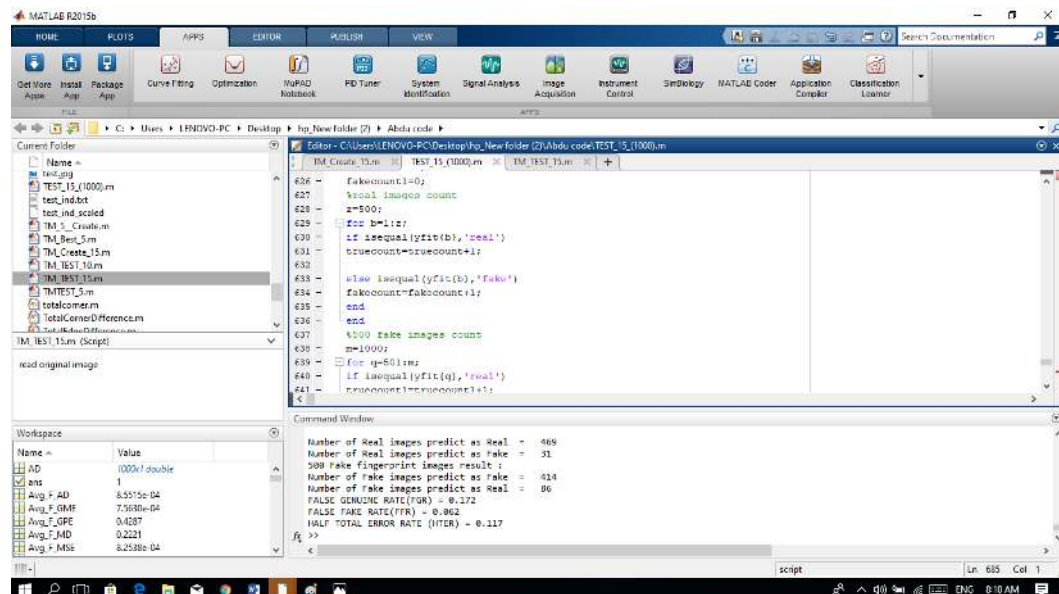


Figure 56: Logistic Regression training result of 100 fingerprint images using 10 IQMs.

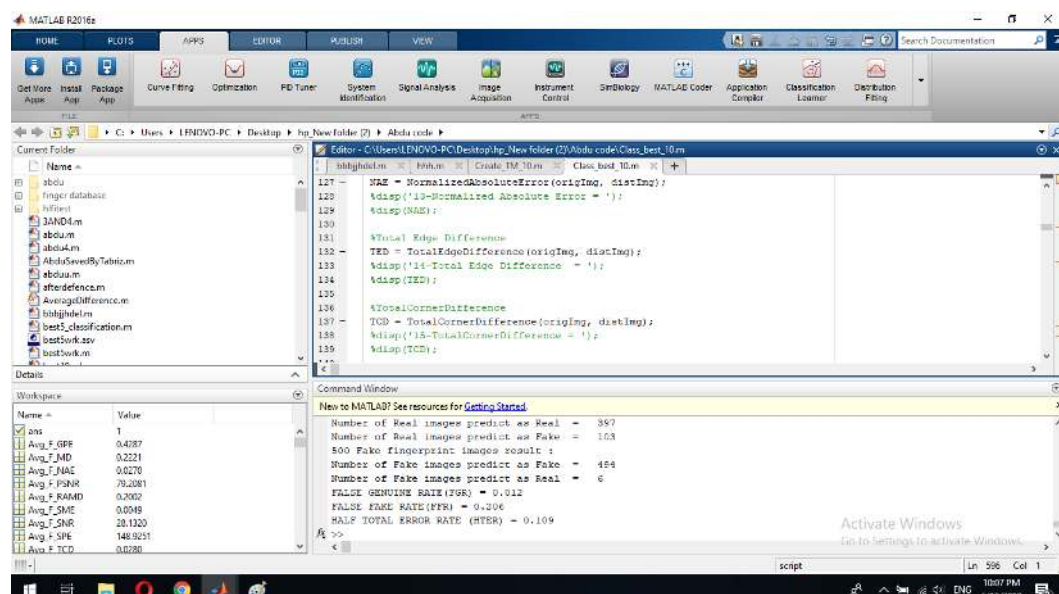
## 4. Table and Screenshot of Classification Results For 1000 Fingerprint Images Using 10 IQMs

1000 Fingerprint images with 10 IQMs result using 6 classifiers:

### 1. LDA



### 2. QDA



### 3. Linear SVM

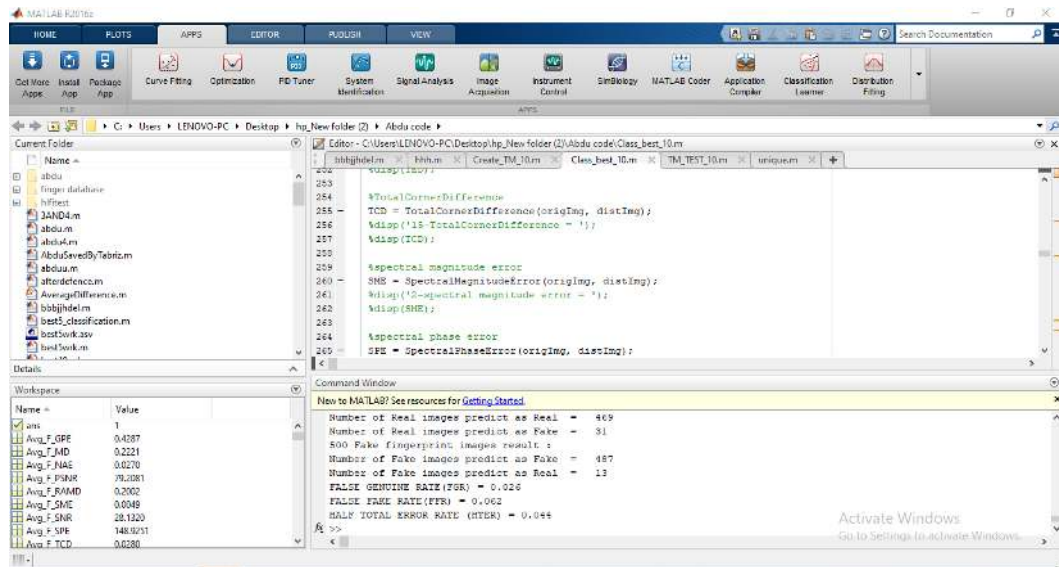


Figure 59: Linear SVM classification result of 1000 fingerprint images using 10 IQMs.

### 4. Quadratic SVM

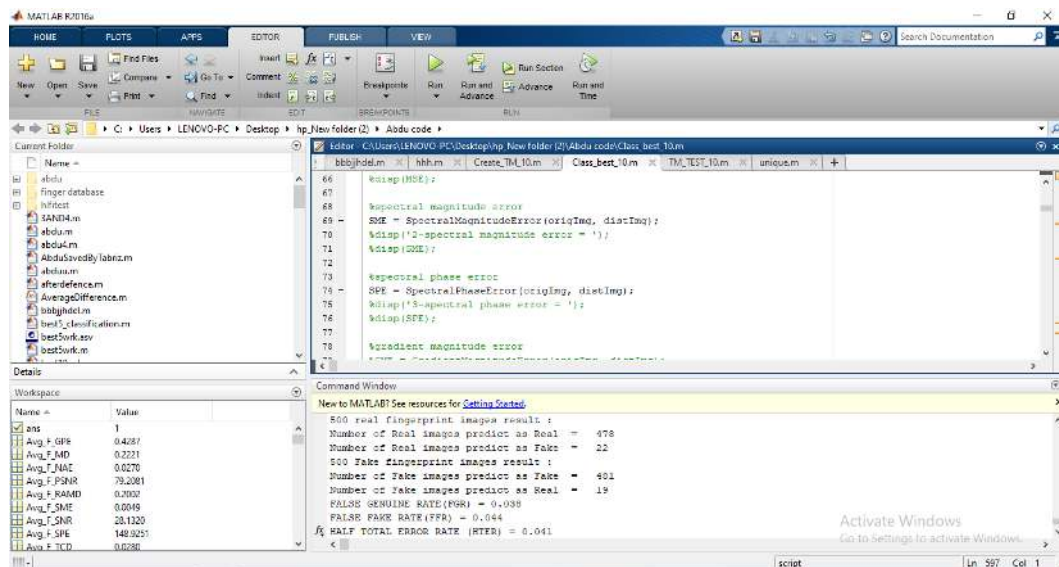


Figure 60: Quadratic SVM classification result of 1000 fingerprint images using 10 IQMs.

## 5. Fine KNN

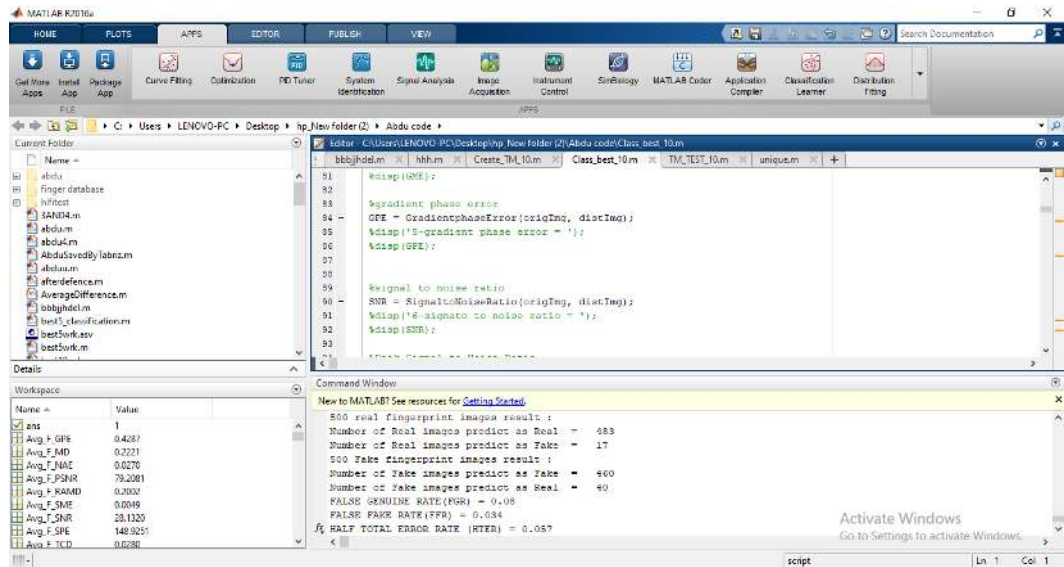


Figure 61: Fine KNN classification result of 1000 fingerprint images using 10 IQMs.

## 6. Logistic Regression

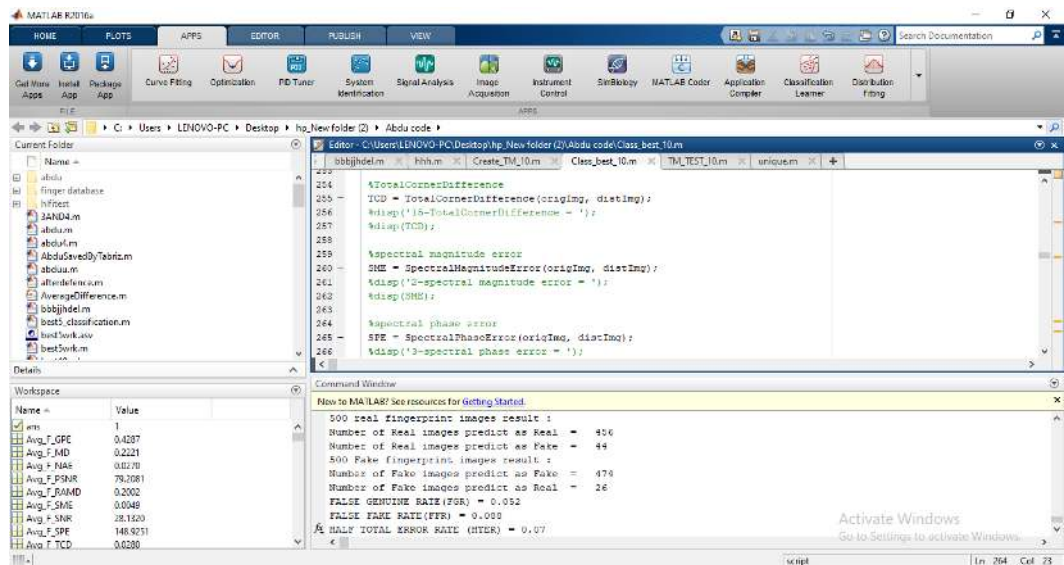


Figure 62: Logistic Regression classification result of 1000 fingerprint images using 10 IQMs.

## 5. Table and Screenshot of Training Results For 100 Fingerprint Images Using 5 IQMs

100 fingerprint images with 5 IQMs result using 6 classifiers:

### 1. LDA

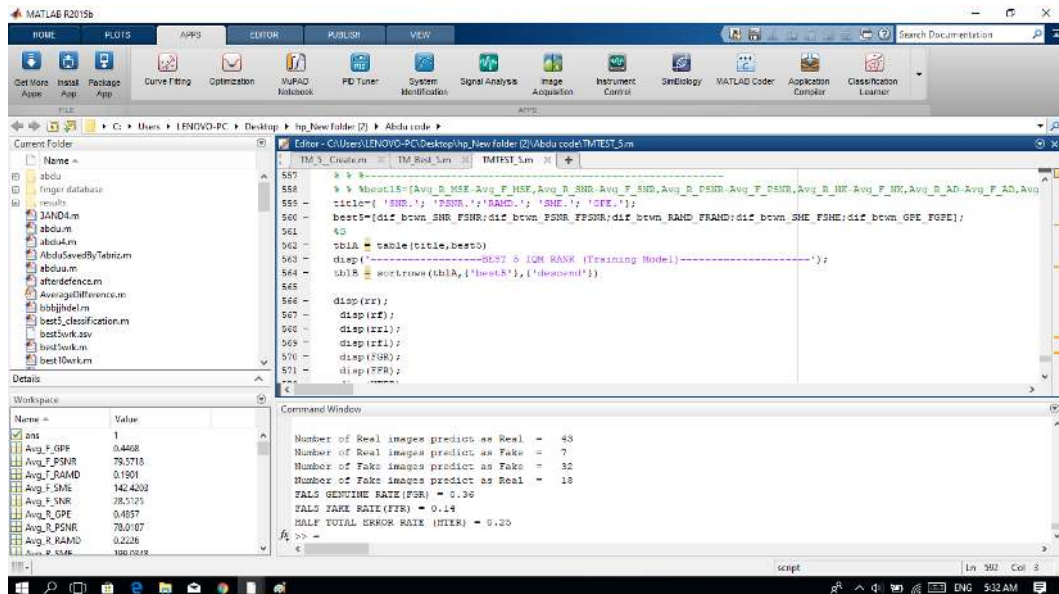


Figure 63: LDA training result of 100 fingerprint images using 5 IQMs.

### 2. QDA

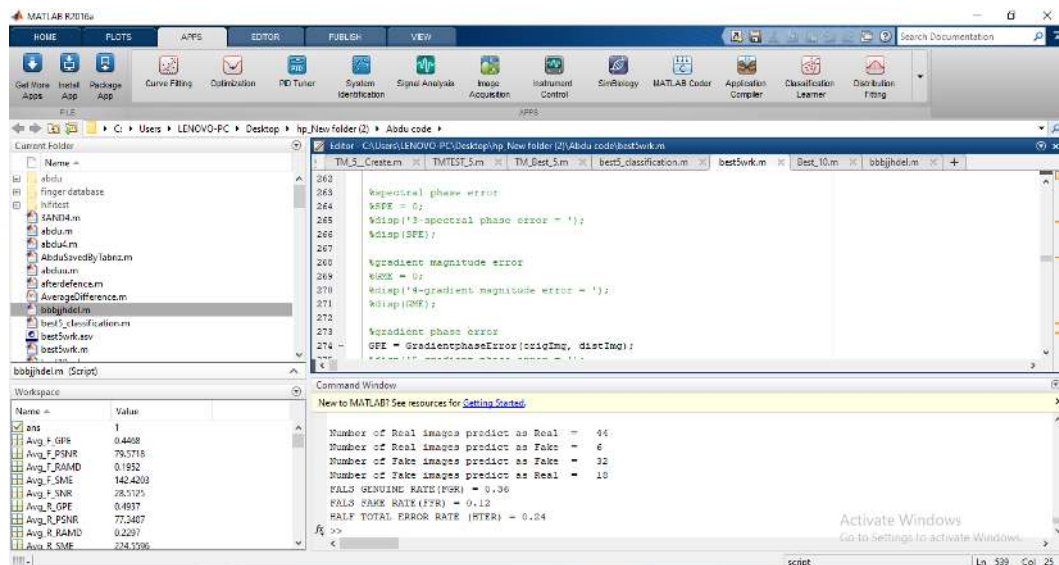


Figure 64: QDA training result of 100 fingerprint images using 5 IQMs.

### 3. Linear SVM

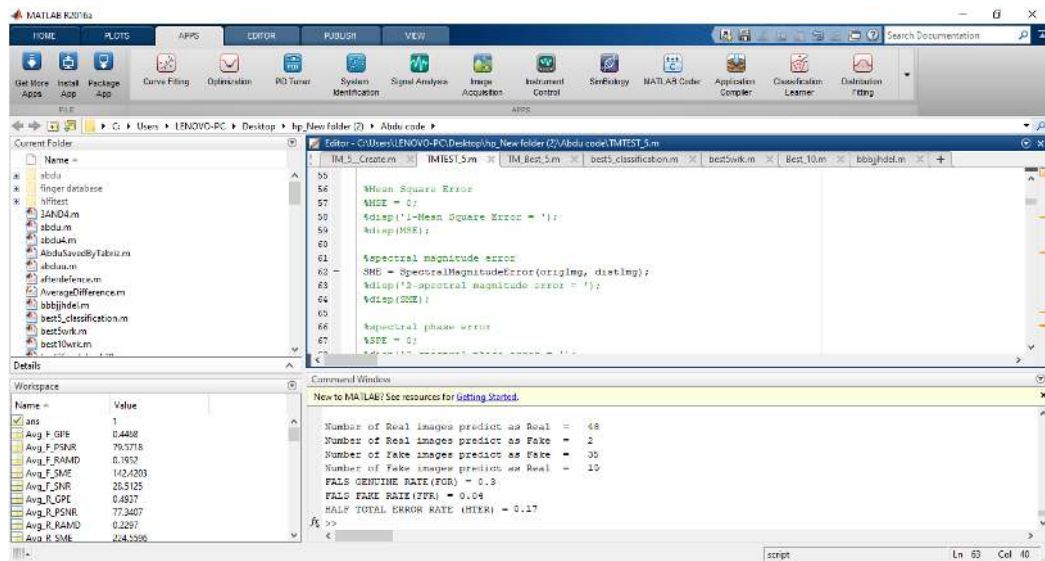


Figure 65: Linear SVM training result of 100 fingerprint images using 5 IQMs.

### 4. Quadratic SVM

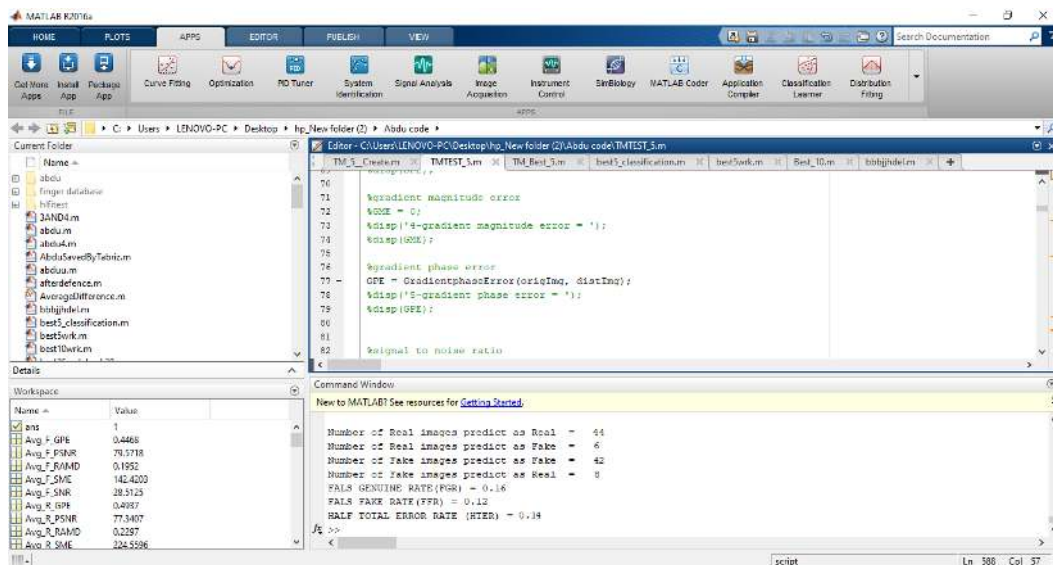


Figure 66: Quadratic SVM training result of 100 fingerprint images using 5 IQMs.

## 5. Fine KNN

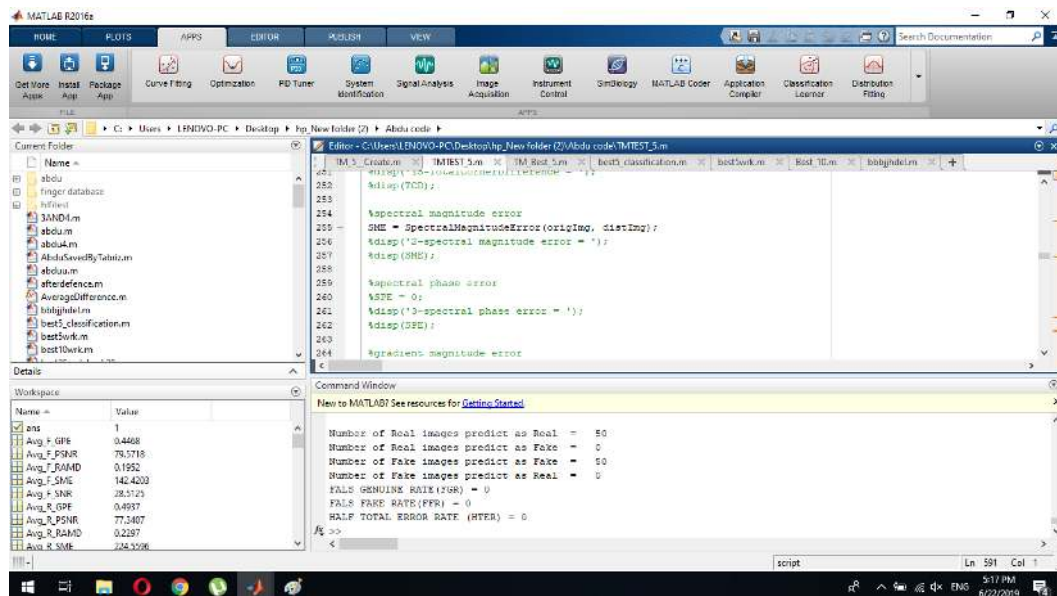


Figure 67: Fine KNN training result of 100 fingerprint images using 5 IQMs.

## 6. Logisctic Regression

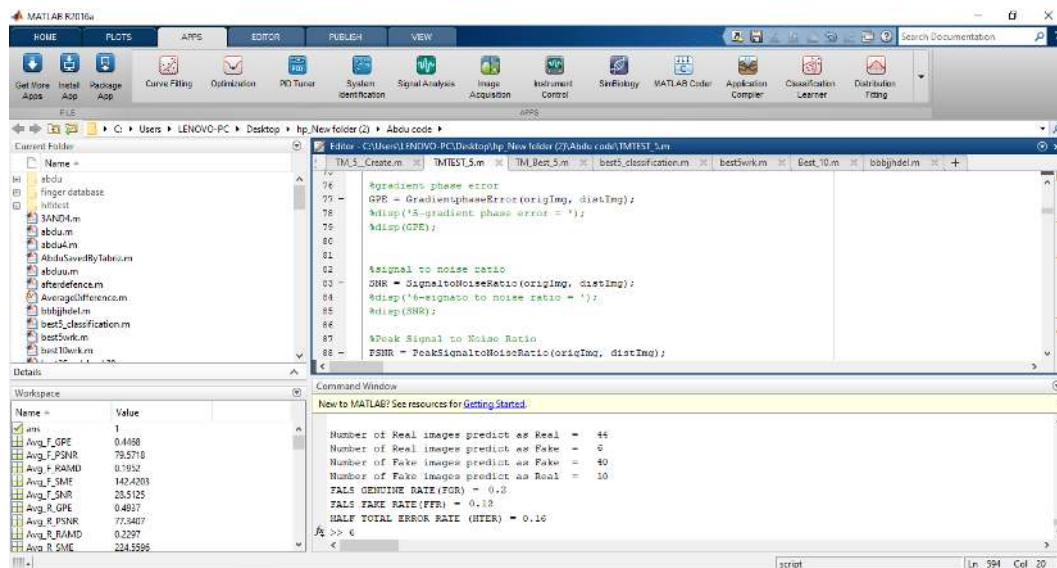


Figure 68: Logistic Regression training result displayed for 100 fingerprint images using 5 IQMs.

## 6. Screenshot of Classification Results For 1000 Fingerprint Images

### Using 5 IQMs:

1000 fingerprint images with 5 IQMs result using 6 classifiers:

#### 1. LDA

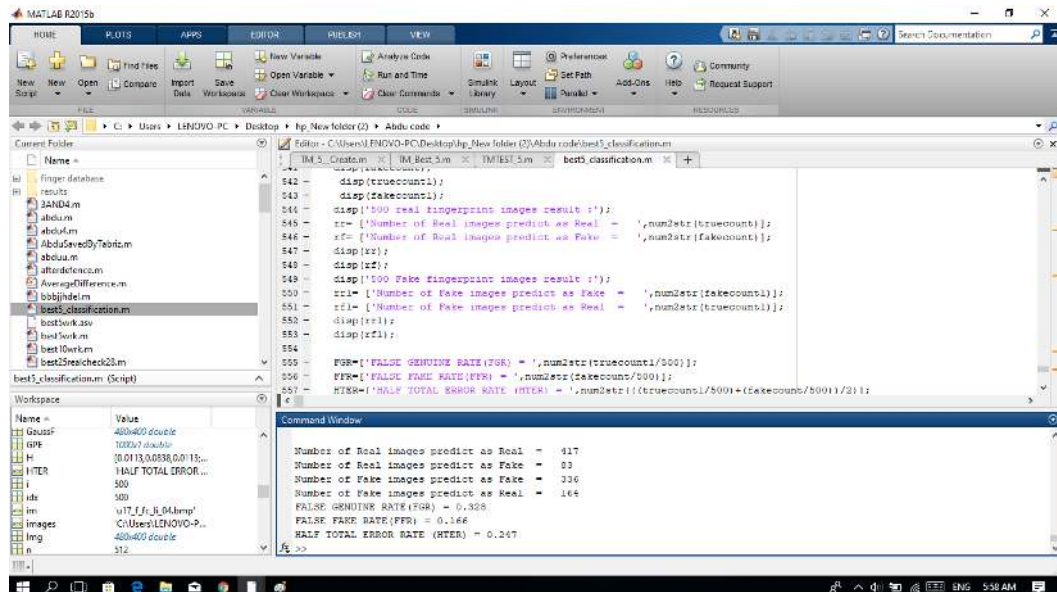


Figure 69: LDA classification result of 1000 fingerprint images using 5 IQMs.

#### 2. QDA

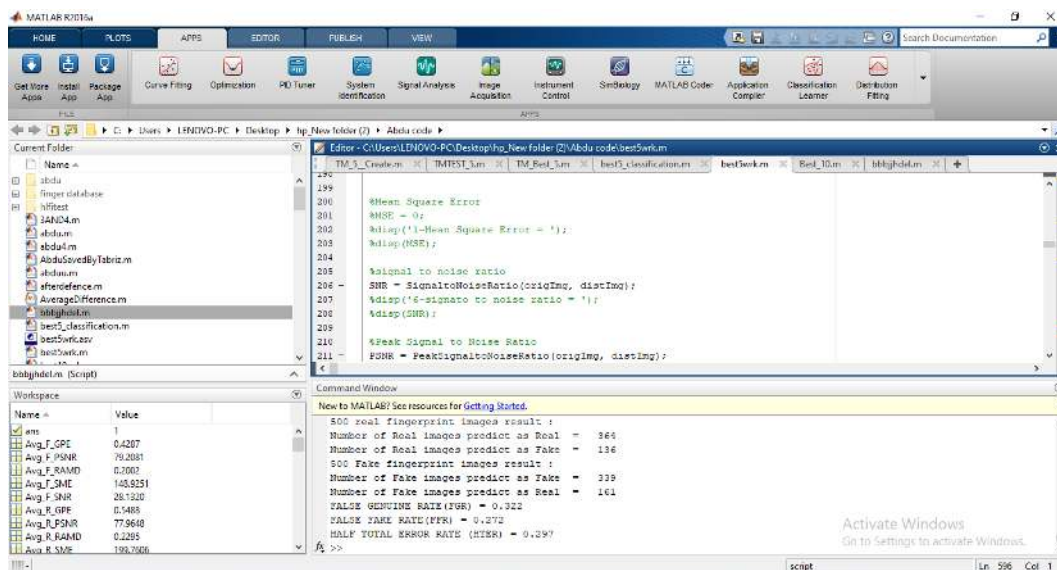


Figure 70: QDA Classification of 1000 Fingerprint Images Using 5 IQMs.

### 3. Linear SVM

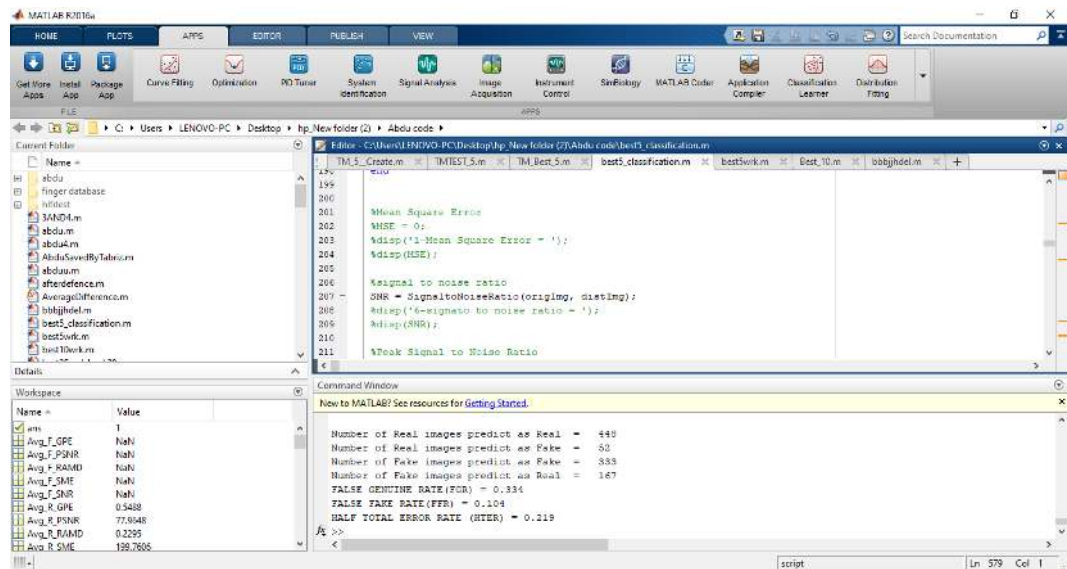


Figure 71: Linear SVM LDA classification result of 1000 fingerprint images using 5 IQMs.

### 4. Quadaratic SVM

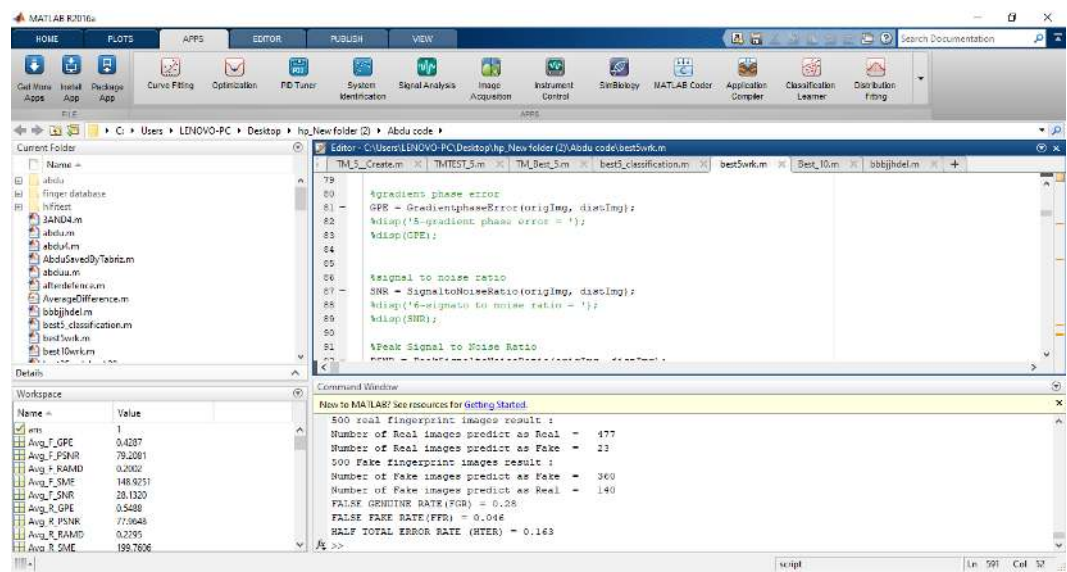


Figure 72: Quadaratic SVM classification result of 1000 fingerprint images using 5 IQMs.

## 5. Fine KNN

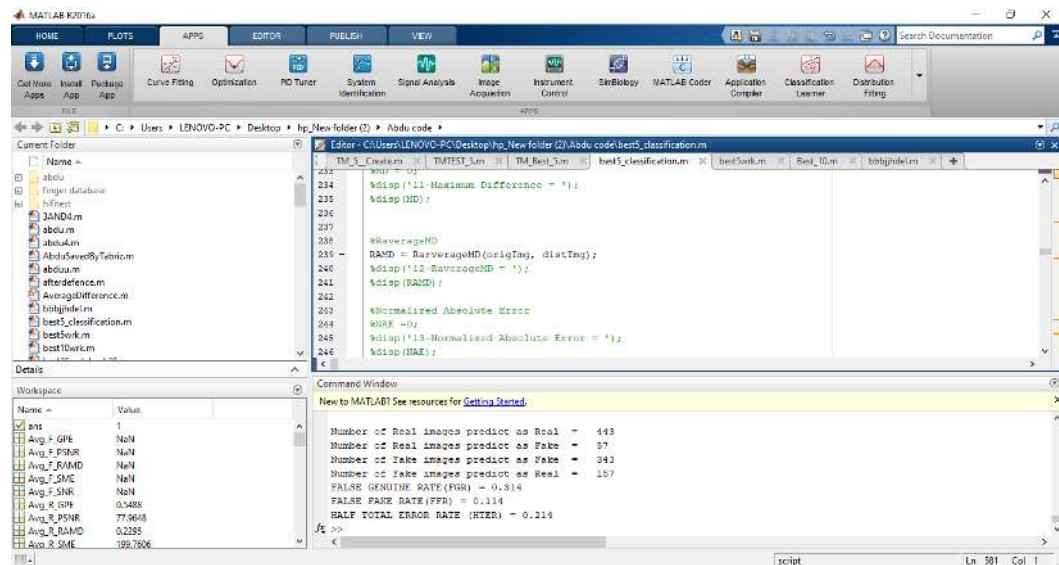


Figure 73: Fine KNN classification result of 1000 fingerprint images using 5 IQMs.

## 6. Logistic Regression

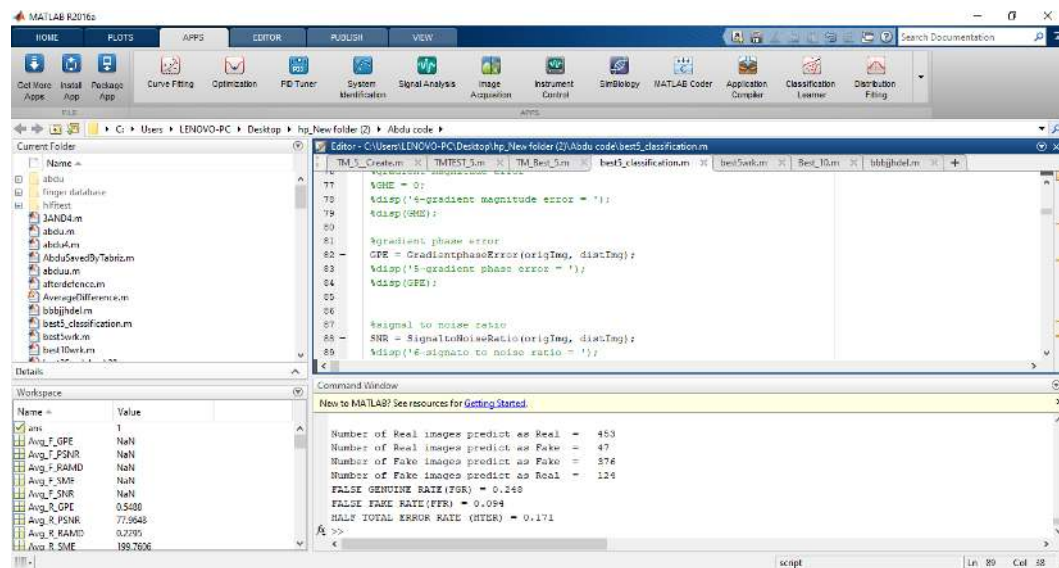


Figure 74: Logistic Regression classification result of 1000 fingerprint images using 5 IQMs.

## 7. Table and Screenshot of Training Results For 100 Fingerprint Images Using 25 IQMs:

100 Fingerprint Images With 25 IQMs Result Using 6 Classifiers:

### 1. LDA

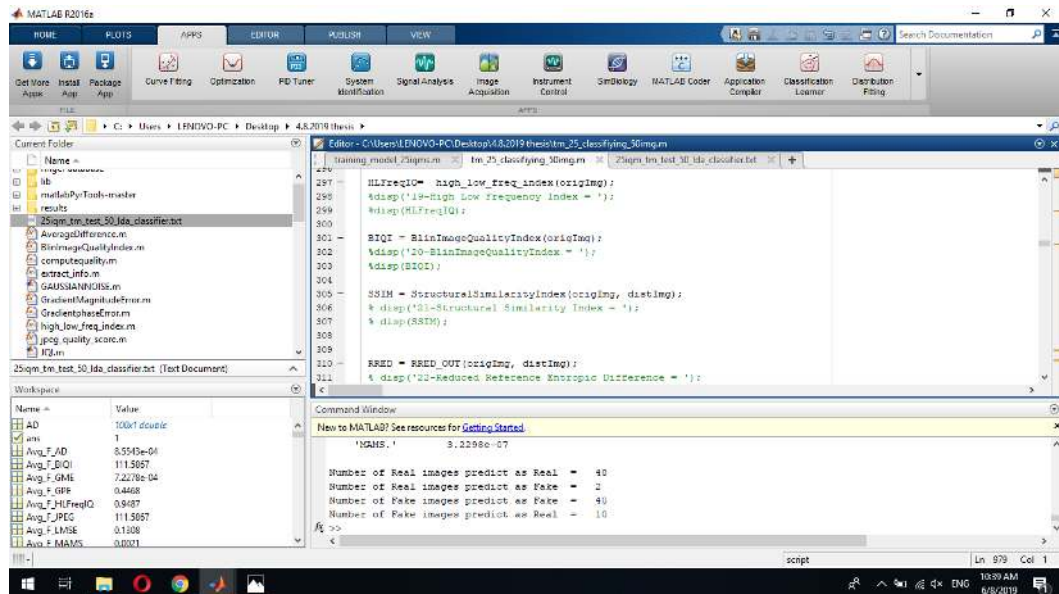


Figure 75: LDA training result of 100 fingerprint images using 25 IQMs.

### 2. QDA

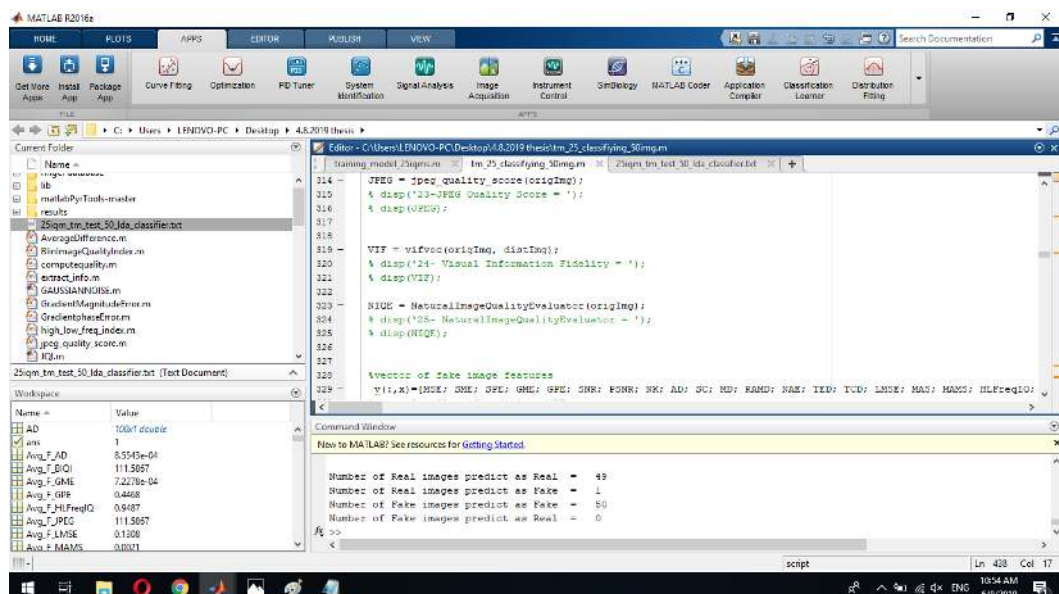


Figure 76: QDA Training Result of 100 Fingerprint Images Using 25 IQMs.

### 3. Linear SVM

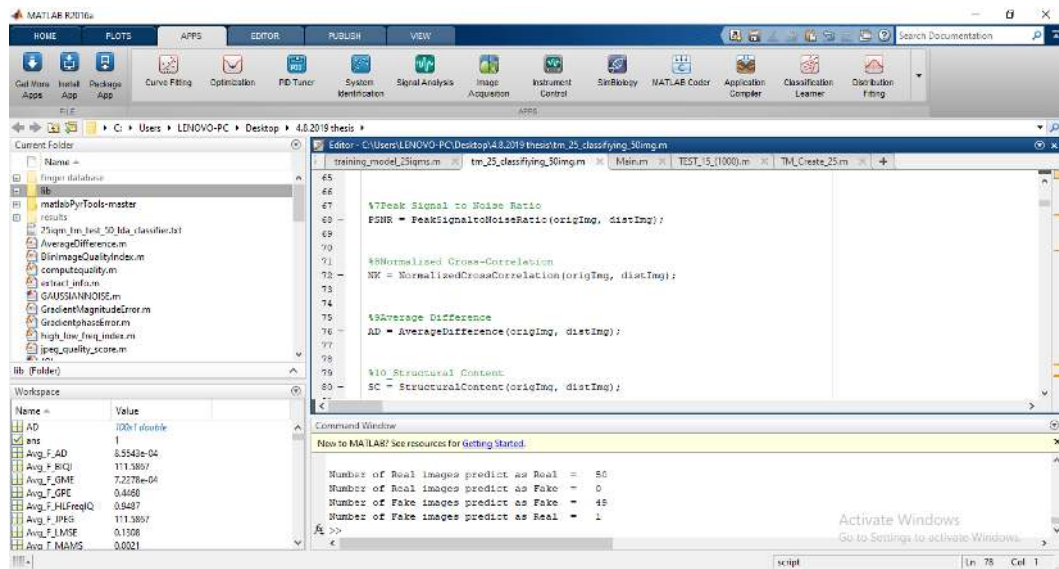


Figure 77: Linear SVM training result of 100 fingerprint images using 25 IQMs.

### 4. Quadaratic SVM

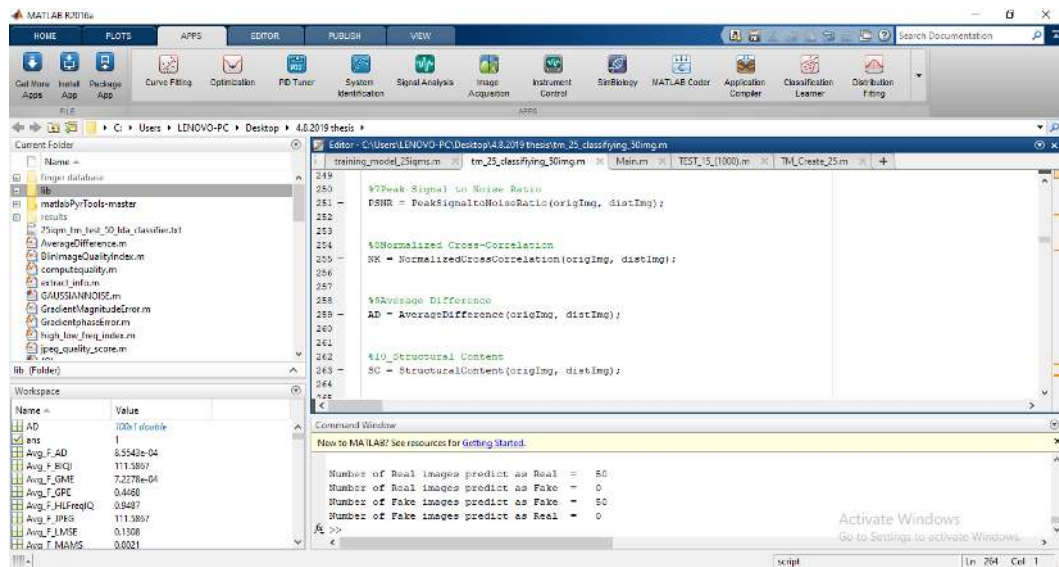


Figure 78: Quadaratic SVM training of 100 fingerprint images using 25 IQMs.

## 5. Fine KNN

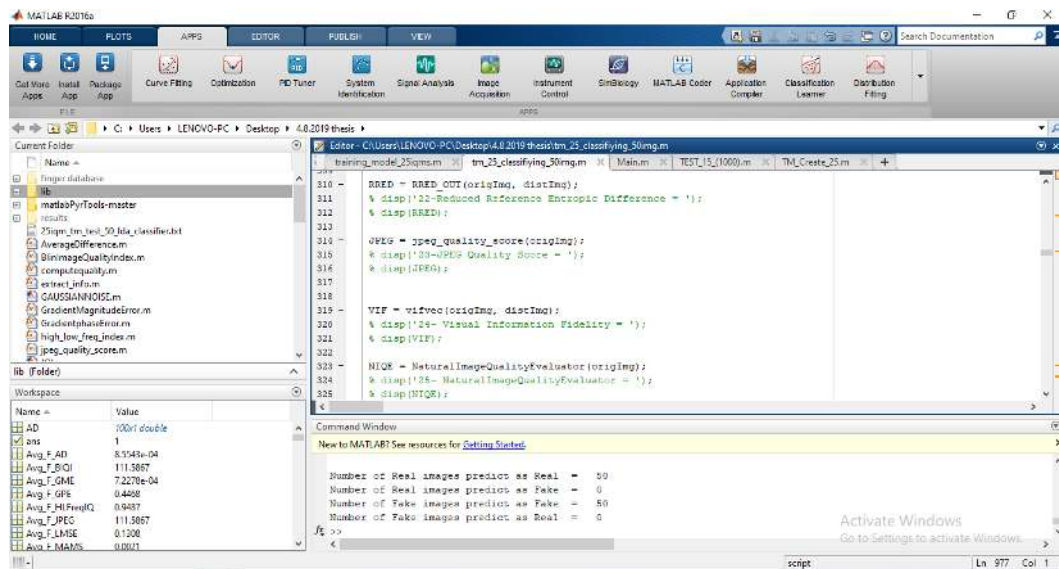


Figure 79: Fine KNN training result of 100 fingerprint images using 25 IQMs.

## 6. Logistic Regression

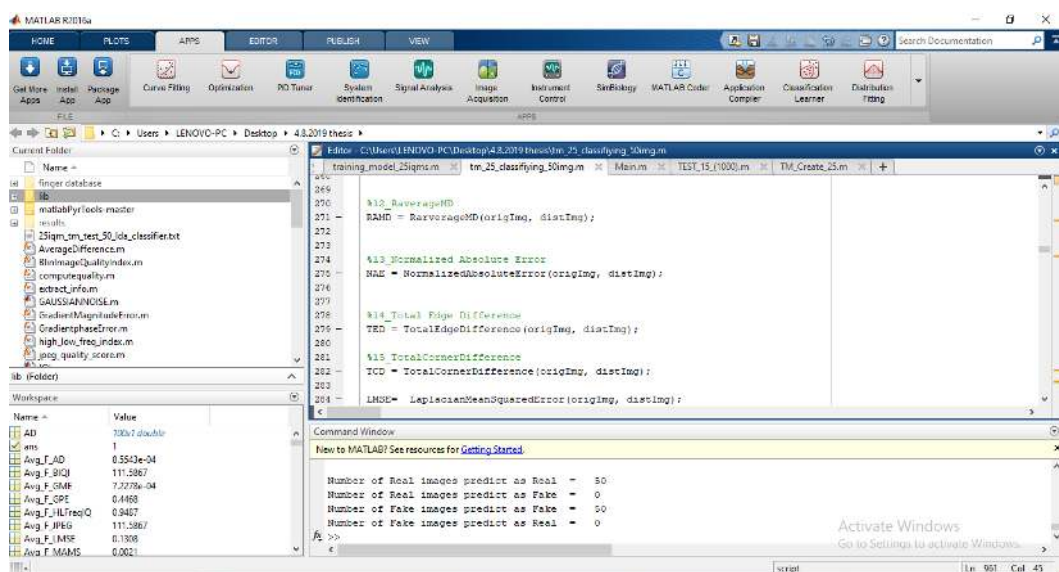


Figure 80: Logistic Regression training result of 100 fingerprint images using 25 IQMs.

## 8. Screenshot of Classification Results For 1000 Fingerprint Images Using All 25 IQMs

1000 Fingerprint images with 25 IQMs result using 6 classifiers:

### 1. LDA

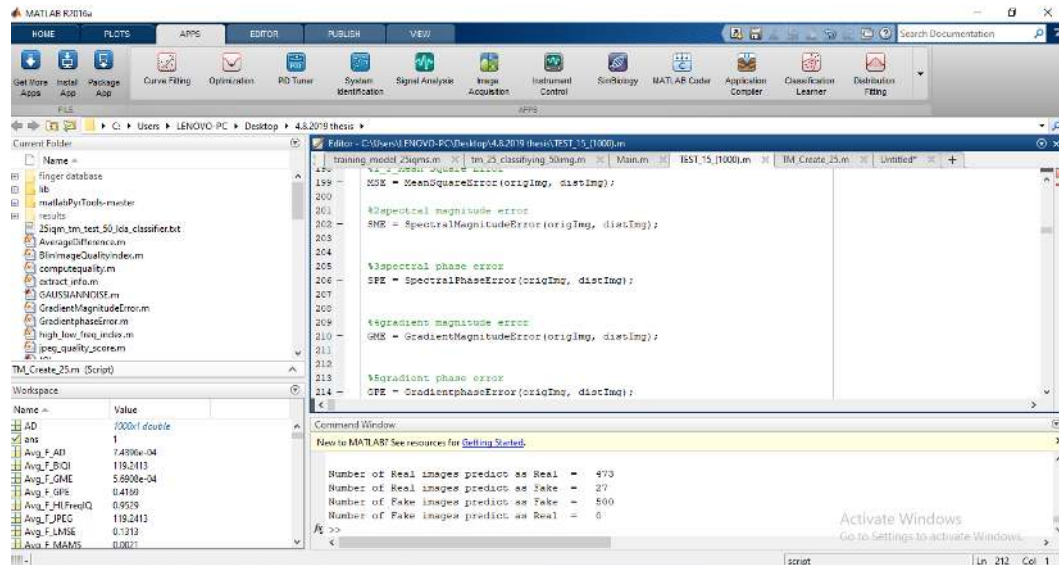


Figure 81: LDA classification result of 1000 fingerprint images using 25 IQMs.

### 2. QDA

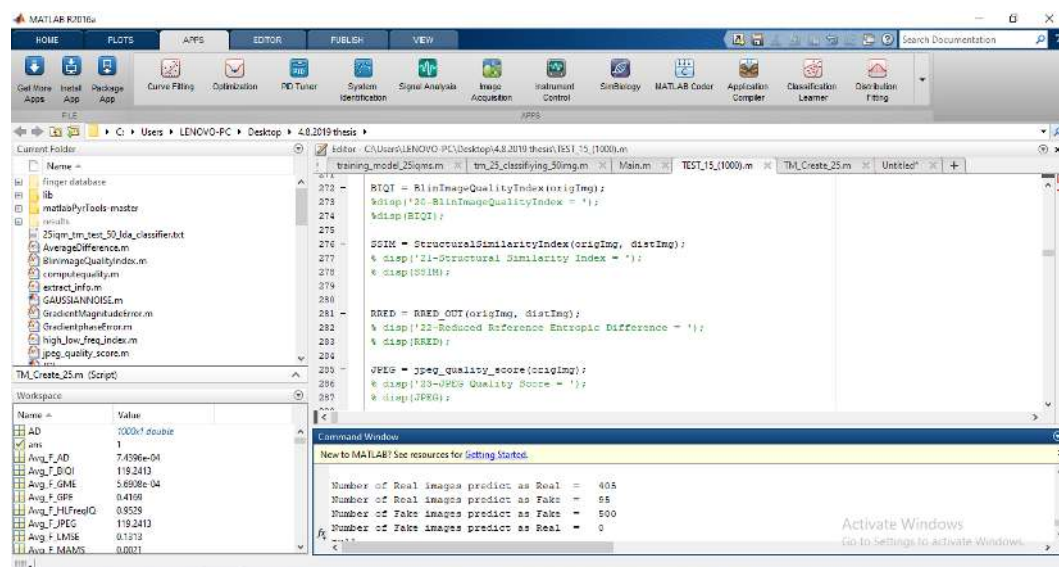


Figure 82: QDA classification result of 1000 fingerprint images using 25 IQMs.

### 3. Linear SVM

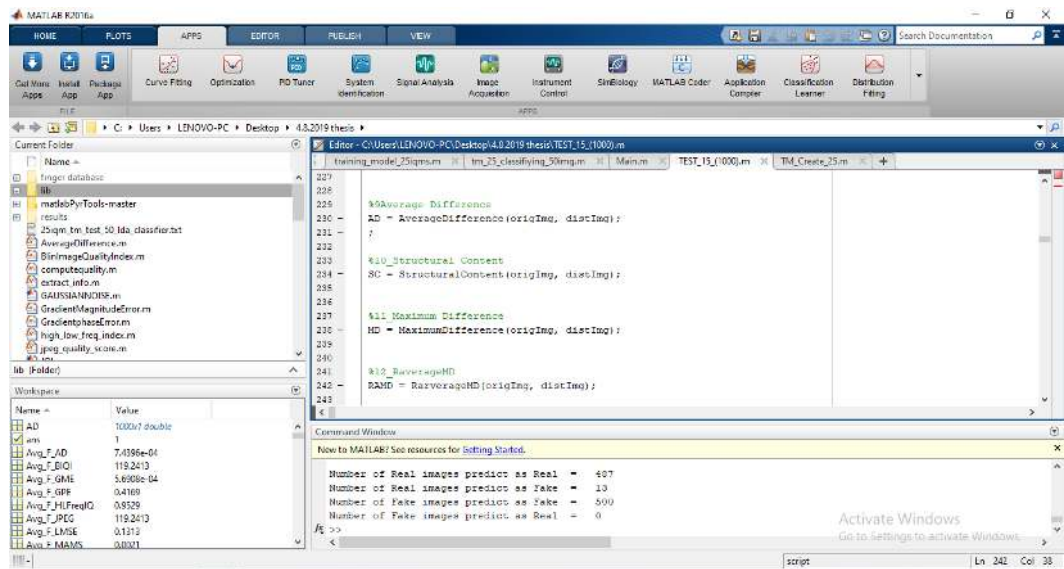


Figure 83: Linear SVM classification result of 1000 fingerprint images using 25 IQMs.

### 4. Quadaratic SVM

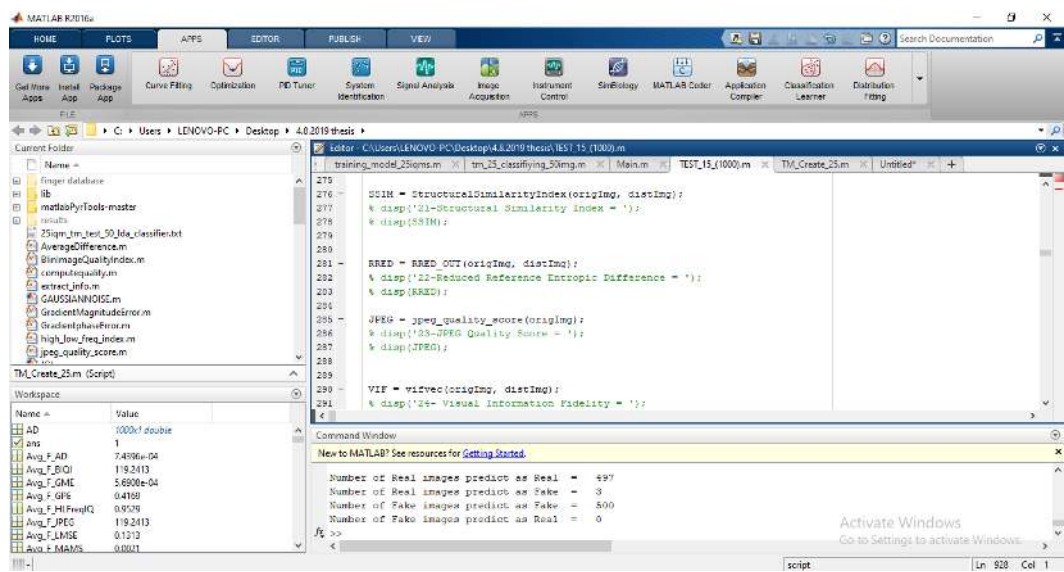


Figure 84: Quadaratic SVM classification result of 1000 fingerprint images using 25 IQMs.

## 5. Fine KNN

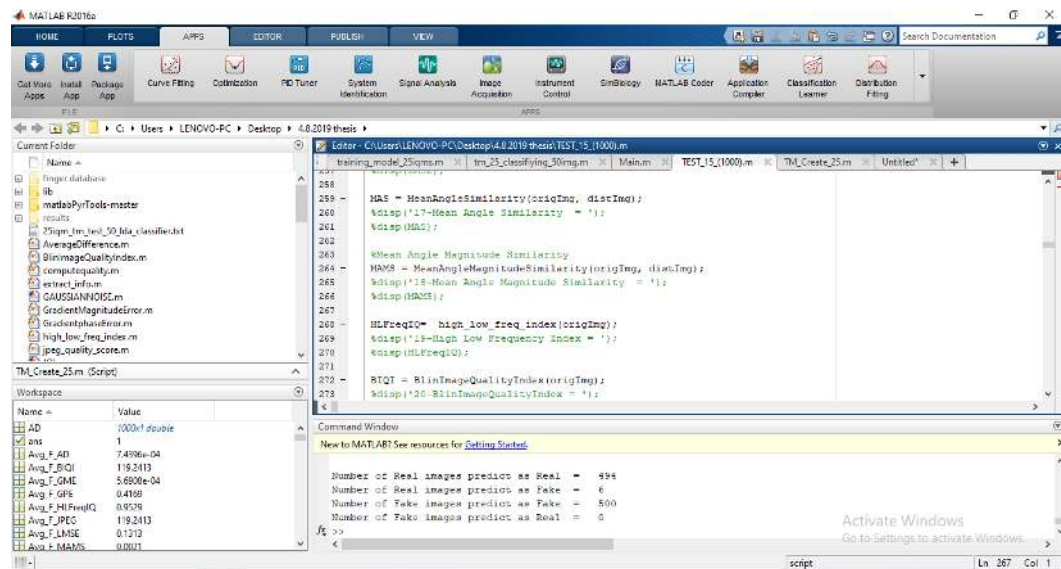


Figure 85: Fine KNN classification result of 1000 fingerprint images using 25 IQMs.

## 6. Logistic Regression

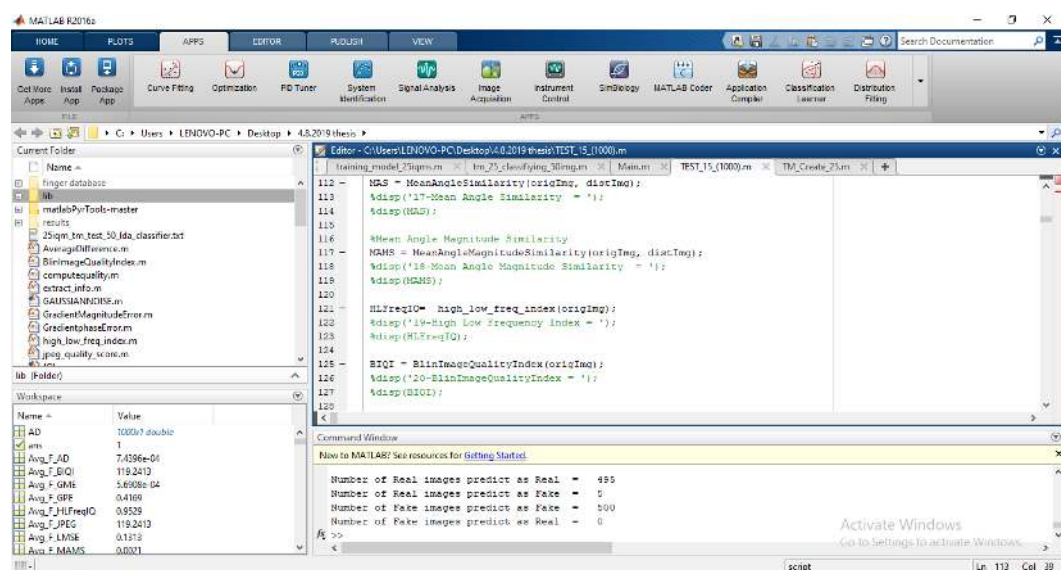


Figure 86: Logistic Regression classification result of 1000 fingerprint images using 25 IQMs.