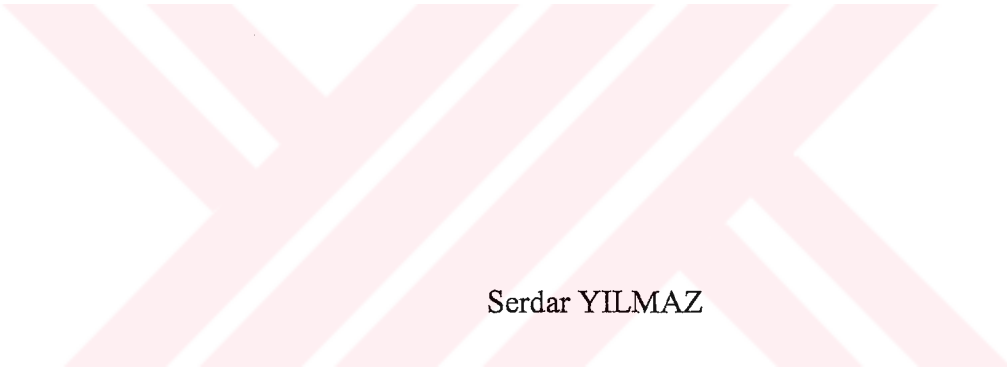


**THE EFFECT of MULTIPLE EXPRESSIONS AND SIZE  
REDUCTION ON PERFORMANCE of HUMAN FACE  
CLASSIFIER**

by



Serdar YILMAZ

September 2003

136 737

**THE EFFECT of MULTIPLE EXPRESSIONS AND SIZE  
REDUCTION ON PERFORMANCE of HUMAN FACE  
CLASSIFIER**

by

Serdar YILMAZ

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science


in

Electronics Engineering

September 2003  
Istanbul, Turkey

136 737

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

  
Prof. Dr. Muhammed KÖKSAL

Head of Department

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

  
Asst. Prof. Dr. Metin ARTIKLAR

Supervisor

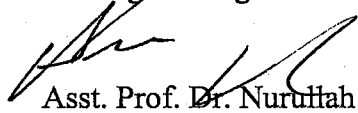
Examining Committee Members

Asst. Prof. Dr. Metin ARTIKLAR

Assoc. Prof. Dr. Erkan İMAL

Asst. Prof. Dr. Atakan KURT

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

  
Asst. Prof. Dr. Nurullah ARSLAN

Director

Date

# **THE EFFECT of MULTIPLE EXPRESSIONS AND SIZE REDUCTION ON PERFORMANCE of HUMAN FACE CLASSIFIER**

Serdar YILMAZ

M. S. Thesis - Electronics Engineering  
September 2003

Supervisor: Asst. Prof. Dr. Metin ARTIKLAR

## **ABSTRACT**

This thesis demonstrates effectiveness of different facial expressions for a template matching based face recognition system. We used the well known nearest neighbor approach in our study. 8 different expressions of each individual are utilized to study the effect of various facial expressions on the recognition performance of the system. Simulations results are reported on CNL database that consists of more than 600 individuals each having 10 different images.

**Keywords:** Face recognition; Template matching; Similarity measures;

# İNSAN YÜZÜ TANIMADA ÇOKLU YÜZ İFADELERİNİN VE BOYUT AZALTMANIN PERFORMANSA OLAN ETKİLERİ

Serdar YILMAZ

Yüksek Lisans Tezi – Elektronik Mühendisliği  
Eylül 2003

Tez Yöneticisi: Yrd.Doç.Dr. Metin ARTIKLAR

## ÖZ

Bu tezde farklı yüz ifadelerinin kalıp (şablon) eşleme yöntemiyle yüz tanımadaki etkinlikleri gösterilmektedir. Çalışmamızda yaygın yüz tanıma yöntemlerinden biri olan Nearest Neighbor yaklaşımını kullandık. Tanımlama performansı çalışmalarımız için 8 farklı resimle ifade edilen kişilerin değişik yüz ifadeleri kullanıldı. Simülasyon sonuçlarımız herbiri 10 farklı resimle ifade edilen CNL veri bankasındaki resimler kullanılarak rapor edilmiştir.

**Anahtar Kelimeler:** Yüz tanıma, Kalıp (Şablon) Eşleme, Benzerlik Ölçüsü

To my parents



## ACKNOWLEDGEMENT

I express sincere appreciation to Asist. Prof. Dr. Metin ARTIKLAR for his guidance and insight throughout the research.

Thanks go to the other faculty members and faculty chairman for their valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but in no sense the least, I am thankful to all colleagues and friends who made my stay at the university a memorable and valuable experience.

## TABLE OF CONTENTS

ABSTRACT .....	III
ÖZ.....	IV
DEDICATION .....	V
ACKNOWLEDGMENT .....	VI
TABLE OF CONTENTS .....	VIII
LIST OF TABLES .....	XI
LIST OF FIGURES .....	XII
LIST OF SYMSBOLS AND ABBREVIATIONS .....	XIV
CHAPTER 1 .....	1
INTRODUCTION.....	1
CHAPTER 2 .....	3
BACKGROUND of FACE RECOGNITION.....	3
2.1 GEOMETRICAL FEATURES MATCHING METHODS .....	3
2.2 TEMPLATE MATCHING METHODS .....	4
CHAPTER 3 .....	6
DATABASE.....	6
CHAPTER 4.....	9
GOALS AND OBJECTIVE.....	9
4.1 COMPARISON of METRICS .....	9
4.2 MULTIPLE IMAGE REPRESENTATION .....	10
4.3 REDUCTION SIZE AND TIME EFFECTS .....	10
4.4 COMPUTER CONFIGURATION .....	10
CHAPTER 5 .....	11
ALGORITHMS.....	11
5.1 FORMAT of THE EXPERIMENTS.....	11
5.2 EUCLIDIAN DISTANCE vs. CITY BLOCK.....	12



CHAPTER 6 .....	15
EFFECT OF FACIAL EXPRESSIONS .....	15
6.1 EXPERIMENTAL RESULT on FACIAL EXPRESSIONS .....	15
6.1.1 Case 1 .....	15
6.1.2 Case 2 .....	17
6.1.3 Case 3 .....	20
6.1.4 Case 4 .....	23
CHAPTER 7 .....	26
REDUCED SIZE AND TIME EFFECT .....	26
7.1 CASE 1: ONE EXPRESSION IN THE MEMORY PER INDIVIDUAL .....	28
7.2 CASE 2: TWO EXPRESSIONS IN THE MEMORY PER INDIVIDUAL .....	31
7.3 CASE 3: THREE EXPRESSIONS IN THE MEMORY PER INDIVIDUAL .....	33
7.4 CASE 2: FOUR EXPRESSIONS IN THE MEMORY PER INDIVIDUAL .....	35
CHAPTER 8 .....	37
SUMMARY AND FUTURE WORK .....	37
8.1 SUMMARY .....	37
8.2 FUTURE WORK .....	38
APPENDIX .....	39
MATLAB FUNCTIONS .....	39
A.1 COMPARISON of METRICS .....	39
A.1.1 CB, ED and L0.5 Norm Comparison Function .....	39
A.1.2 Result of Comparision Function .....	41
A.1.3 Summary of Result Function .....	42
A.2 ONE MEMORY IMAGE SIMULATION FUNCTIONS (E.D.) .....	43
A.2.1 One Memory Main Function .....	43
A.2.2 Result of Main Function .....	44
A.2.3 Summary of Result Function .....	46
A.3 TWO MEMORY IMAGES SIMULATION FUNCTIONS (E.D.) .....	47
A.3.1 Two Memory Main Function .....	47
A.3.2 Result of Main Function for Two Memory Images .....	49

A.3.3 Summary of Result Function For Two Memory Images.....	51
A.4 THREE MEMORY IMAGES SIMULATION FUNCTIONS (E.D.).....	52
A.4.1 Three Memory Main Function .....	52
A.4.2 Result of Main Function For Three Memory Images.....	53
A.4.3 Summary of Result Function for Three Memory Images.....	56
A.5 FOUR MEMORY IMAGES SIMULATION FUNCTIONS (E.D.).....	57
A.5.1 Four Memory Main Function .....	57
A.5.2 Result of Main Function For Four Memory Images .....	59
A.5.3 Summary of Result Function For Four Memory Images .....	62
REFERENCES .....	64



## LIST OF TABLES

### TABLE

7.1 Reduced Row and Column Size of Image Expressions .....27



## LIST OF FIGURES

### FIGURE

Figure 2.1 Some Feature Points Used to Represent a Face:.....	4
Figure 3.1 The distribution of ethnic-backgrounds of 1200 individuals in the CNL database.....	6
Figure 3.2 Memory images of person 1 in the database. . . . .	7
Figure 3.3 Memory images of person 2 in the database. . . . .	7
Figure 4.1 Computer configuration and software program.....	10
Figure 5.1 Comparison of metrics using blank test expression.....	13
Figure 5.2 Comparison of metrics using arbitrary test expression.....	13
Figure 5.3 Comparison between processing time of an input image for $L_{0.5}$ , $L_1$ and $L_2$ metrics.....	14
Figure 6.1 Recognition performances for including one expression in the memory for the cases of (a) blank test image and (b) arbitrary test image.....	16
Figure 6.2 Recognition performances for including two expressions in the memory when tested against blank test image set.....	17
Figure 6.3 Recognition performances for including two expressions in the memory when tested against arbitrary test image set.....	18
Figure 6.4 Contributions of individual expressions in the case of 2 expressions per individual tested against (a) blank test images set (b) arbitrary test images set.....	20
Figure 6.5 Recognition performances for case of including three expressions in the memory when tested against (a) blank test image set (b) arbitrary test image set.....	21
Figure 6.6 Contributions of individual expressions in the case of 3 expressions per individual tested against (a) blank test images set (b) arbitrary test images set.....	22

Figure 6.7 Recognition performances for case of including four expressions in the memory when tested against (a) blank test image set (b) arbitrary test image set.....	23
Figure 6.8 Contributions of individual expressions in the case of 4 expressions per individual tested against (a) blank test images set (b) arbitrary test images set.....	24
Figure 7.1 Original and Reduced Size Classification Performances for One Memory Expression vs. Blank Type Test Expression.....	29
Figure 7.2 Original and Reduced Size Classification Performances for One Memory Expression vs. Arbitrary Type Test Expression.....	29
Figure 7.3 Original and Reduced Size Classification Time for One Memory Expressions as Second.....	30
Figure 7.4 Original and Reduced Size Classification Performances for Two Memory Expression vs. Blank Type Test Expression.....	31
Figure 7.5 Original and Reduced Size Classification Performances for Two Memory Expression vs. Arbitrary Type Test Expression.....	32
Figure 7.6 Original and Reduced Size Classification Times for Two Memory Expression.....	32
Figure 7.7 Original and Reduced Size Classification Performances for Three Memory Expression vs. Blank Type Test Expression.....	33
Figure 7.8 Original and Reduced Size Classification Performances for Three Memory Expression vs. Arbitrary Type Test Expression.....	34
Figure 7.9 Original and Reduced Size Classification Times for Three Memory Expressions.....	34
Figure 7.10 Original and Reduced Size Classification Performances for Four Memory Expression vs. Blank Type Test Expression.....	35
Figure 7.11 Original and Reduced Size Classification Performances for Four Memory Expression vs. Arbitrary Type Test Expression. ....	36
Figure 7.12 Original and Reduced Size Classification Times for Four Memory Expressions.....	36

## LIST OF SYMSBOLS AND ABBREVIATIONS

### SYMBOL/ABBREVIATION

CNNL	Computation and Neural Network Laboratory
E.D.	Euclidean Distance
C.B.	City Block



## CHAPTER 1

### INTRODUCTION

Biometrics is a field in which humans are being recognized by their biological features. This field has been recently drawing a lot of attention because of its applications to vision and security systems.

Identification systems based on biological features have been used in our daily life for quite some time. For example; fingerprint analysis is being utilized by law enforcement agencies in order to identify the criminals, systems based on retina scan are also successfully implemented for identifying individuals. Recently, DNA samples taken from human body are being accepted as legitimate evidence in the court rooms (Anil K. 2000).

The reason that faces recognition became very popular is that; it does not need a sophisticated system in order to obtain people's face image. For example, for a fingerprint recognition process; person has to put his/her finger onto a scanner voluntarily. For iris recognition, a high quality picture of the subject eyes has to be obtained. A typical face recognition system, on the other hand, does not need this kind of sophisticated systems (Anil K. 2000).

In order to state what face recognition is, there are two fundamental concepts to be explained in advance: A face database and a face recognition algorithm.

Face database is, in its simplest term, collection of human face images. In general images in the database can come from different size orientation and illumination backgrounds. Subjects in the database can be represented by one or multiple images.

A face recognition algorithm is a method which is applied to an input face image in order to find the similarity between the input and database images. Typically a predetermined similarity measure is used (like Euclidian or city block distance) in order to perform this task.

After explaining the two fundamental concepts in face recognition, we are now ready to face recognition problem: Given a face recognition algorithm, a face recognition algorithm takes an input image and (possibly by comparing it to all the images in the database) decides whether the input face image belongs to the database or not.

There are so many difficulties associated with a typical face recognition system. First, face images contained in the database can differ in size and orientation. Secondly, the changes in the light illumination of the environment in which the images are obtained can cause problems for the recognition system. Thirdly, different hair styles make-up existence or non-existence of facial hair (mustache and beard) can affect the face recognition system in a significant way. In fact, it is shown that a difference between two images of a same person with different facial expressions can be sometimes larger than the difference between two images of the two different individuals with similar facial expressions (Belhumeur, 1997).



## CHAPTER 2

### BACKGROUND of FACE RECOGNITION

Face recognition has been subject of an extensive research in recent years and many algorithms have been proposed in order to identify a face. These methods can roughly be classified into four main areas: geometrical features base methods, template matching methods, statistical methods, and neural network based methods (also known as learning based methods).

#### 2.1 GEOMETRICAL FEATURES MATCHING METHODS

Geometrical features methods are based on extracting relative size and distances of important facial features and forming a feature vector based on these values in order to identify an individual. This was the first method applied to face recognition. In 1964, Goldstein used this method and extracted manually identified feature points from a face in order to identify that individual. Later, Kanade extended the idea and developed first automated face recognition system, i.e. the facial features were extracted automatically by the computer in this system.

There are two main points of this approach which need to be given careful attention:

Selection of a set of features which can properly characterize a face.

A reliable and fast algorithm method that can accurately and quickly extract this set of features by computers.

Kaya and Kobayashi (1972) used Euclidean distances between manually identified points in the images to characterize faces. When they were deciding on which kind of features they were going to use, they considered some factors such as given below (R. Brunelli 1992);

- estimation must be as easy as possible;
- light dependencies must be as small as possible;
- facial expressions dependencies must be as small as possible;

- information content must be as high as possible.

Examples of facial features are distances and angles between geometric points such as eyes, eyebrows, mouth, nose and chin. These features differ from one to another person and can be represent that person.

Following figure shows a pictorial representation of some of these features on a face image (R. Brunelli 1992).

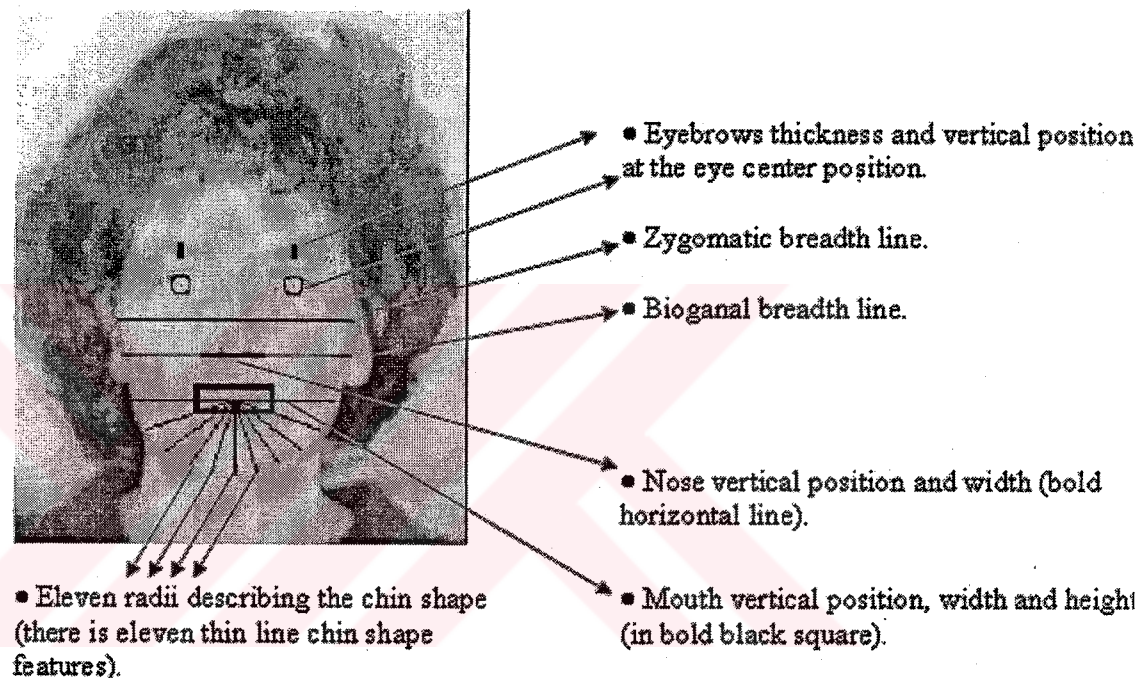


Figure 2.1 Some Feature Points Used to Represent a Face: 22 geometrical features extracted in this face (R. Brunelli 1992).

Kanade used a method called integral projection in order to extract these facial features. In integral projection methods, we first generate a vector that represents each row by the summation of all pixels resulting in that row (per row). After that we apply the same process to obtain a column vector representing each column in the image by one value which is equal to summation of the pixel values in that particular column.

Kanade showed integral projection vectors can be used to identify the position of some important facial features like eyes and nose, in a face image (R. Brunelli 1992).

## 2.2 TEMPLATE MATCHING METHODS

Template matching is a useful operation in pattern recognition which is used to determine the similarity between two data (signals, vectors, strings or 2D picture

pattern) of the same type. Matching algorithm calculates or measures similarities and dissimilarities between training database and test database using templates matching. Template matching could be applied to whole picture or could be applied to some parts of the picture (Anil K. 2000). Generally templates are chosen from high similarity part of the pictures or it is chosen from high dissimilarity parts of pictures, from another way low similarities could be template as well (Anil K. 2000). Template patterns are accepted as reference patterns or reference sets from training data. A test picture (or not recognized pictures) is identified by using template patterns. Therefore, it is necessary to calculate pattern similarities or dissimilarities between training patterns and test pictures (R.Brunelli 1993).

In general this is done by comparing the test pattern to the template pattern in all different size and orientation levels. Note that it is also important to use a suitable metric in this comparison in order to get an accurate value of measurements. This process is also known as cross-correlating the training pattern with the test pattern (R.Brunelli 1993) (R.Brunelli 1997).

The template matching procedure can be applied to a signal, vector, string, two dimensional or multiple dimensional data. In face recognition two dimensional templates are used most popularly.

For face recognition, a whole image can be used to form a template or one or more distinctive parts, like eyes and nose, can be used as templates. In later case the number of templates can be more than one.

Compared to geometrical features based method, template matching methods are in general less time consuming and easy to implement (since we do not need to extract so many features for each face). It is also shown that (R. Brunelli 1992) the template matching algorithms perform better in identification compared to geometrical feature based ones.

## CHAPTER 3

### DATABASE

We have conducted all of our simulations on CNNL face database. A brief explanation of the database is given next. Detailed discussion of the construction of the database can be found in [Artiklar, 2002].

CNNL face database consists of over 1200 individuals between ages of 15 to 60 years old. Subjects come from different ethnic backgrounds such as Caucasian-American, African-American, Asian, Indian, Middle Eastern, etc. Out of 1200 individuals, 733 are men and 467 are women. Figure 1 shows the distribution of ethnic backgrounds over 1200 individuals in our database.

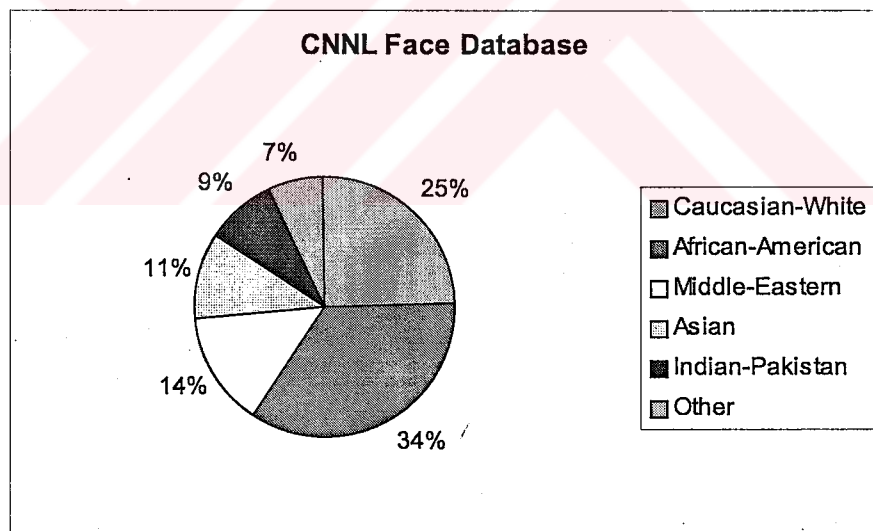


Figure 3.1 The distribution of ethnic-backgrounds of 1200 individuals in the CNNL database.

The images were collected in a laboratory setting so as to minimize the amount of pre-processing that is necessary in order to reduce the effect of tilt rotation, translation, scaling, etc.

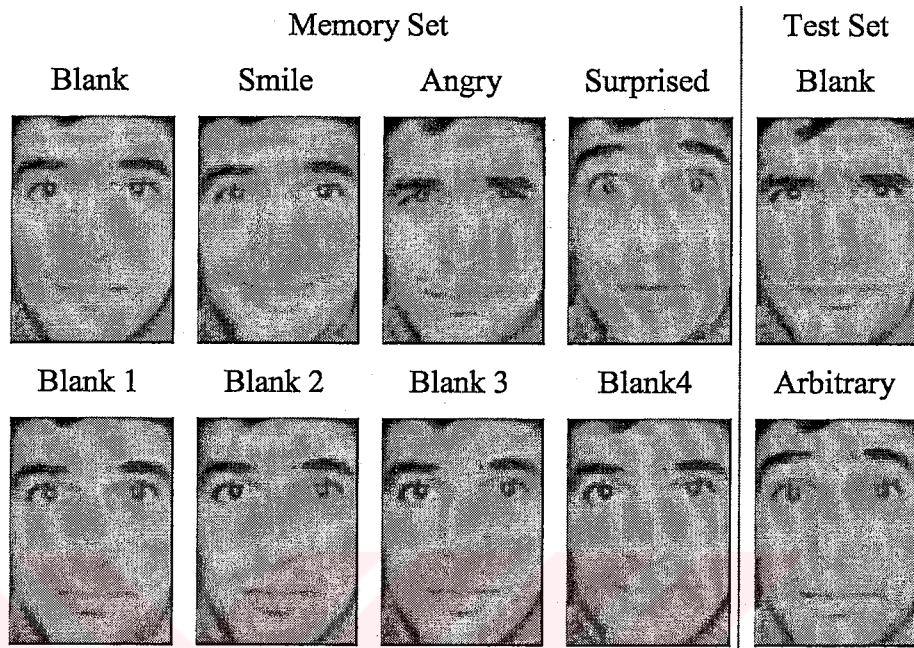


Figure 3.2 Memory images of person 1 in the database

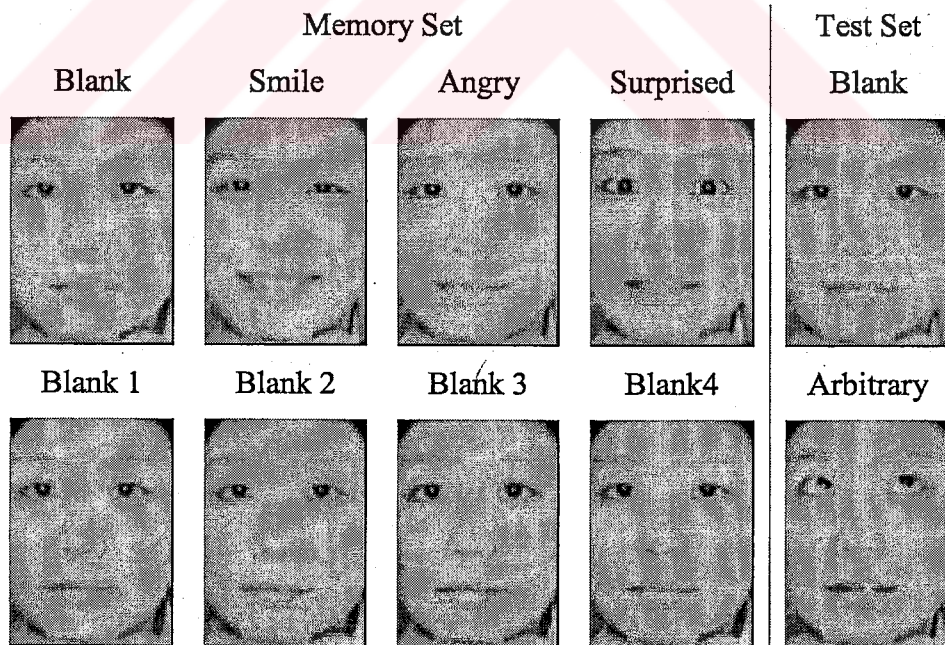


Figure 3.3 Memory images of person 2 in the database.

Each individual is represented by 8 expressions in the database: Blank, smile, angry, surprise, blank1, blank2, blank3, and blank4. Two additional expressions, blank and arbitrary, are obtained separately to generate additional test sets. For the arbitrary test expression, the subject is asked to give an expression that makes it difficult for computer to recognize the person. Both database and test sets are snapped as gray scale images at dimension of 115 x 82. Figure 3.2 and 3.3 show these images of an individual from our database (Artiklar 2002).



## CHAPTER 4

### GOALS and OBJECTIVE

This chapter presents the goals of this thesis. We start by mentioning that our aim in this thesis is not to develop a new face recognition algorithm but rather to perform experiments regarding relevance of different facial expressions on recognition performance of a typical classification method.

We will achieve our goal by implementing simple yet powerful recognition algorithm known as Nearest Neighbor algorithm. Although simple, Nearest Neighbor algorithm can achieve close to optimal solution (bounden by Bayesian from above and twice as worse as Bayesian from below) in its extremes (Duda 2000).

Our objectives, from this point on, will be divided in 3 parts. In the first part; we will compare the effect of 3 different distance measures on the performance of our face recognition algorithm. In the second part, the effect of including one expression versus multiple expressions into memory and their effect on performance of the algorithm will be investigated. In the third part, we will reduce the size of database pictures gradually and measure the system's performance (T. Sim 2000).

#### 4.1 COMPARISON of METRICS

It is well known that different metrics affect the performance of a system in different ways depending on the type of application being applied. In this thesis, we investigate the effect of 3 distance measures on the classification performance of a face recognition system, namely Euclidean, City Block and  $L_{0.5}$  norm.

The reason for experimenting on Euclidian and City Block is due to their widespread use in image processing community. We have added  $L_{0.5}$  to our list because some papers mentioned this metric.

## 4.2 MULTIPLE IMAGE REPRESENTATION

In the second part of our thesis, we investigate the effect of expression or expressions on recognition performance of the Nearest Neighbor algorithm. To achieve this objective, we first include one expression per individual into the memory and measure the system's performance against test sets. We then increase the number of expressions per individuals to two, three and four in all possible combinations. We will run extensive simulations as to demonstrate which combination in each case (in the cases of one, two, three, four expressions in memory) is best suited too represent an individual. In addition to that, the performance of the Nearest Neighbor algorithm will be watched against the increase in the number of images per individual in the memory.

## 4.3 REDUCTION SIZE AND TIME EFFECTS

As mentioned earlier, Nearest Neighbor algorithm has good recognition performance over face images nevertheless classification time increases by the size of data linearly. Therefore, representing an individual with more than one image will result in longer classification time.

It is expected that as the size of face images decreases the performance of the recognition system should deteriorate accordingly. However it is also known that images taken in real environment contain a certain percentage of noise. Hence reducing the size of an image, in this case, would correspond to eliminating some portion of noise and might even increase the performance of the system. In order to experiment on this subject, we will gradually decrease the size of the database pictures and look at the affect of this process on the classification performance (T. Sim 2000).

## 4.4 COMPUTER CONFIGURATION

These experiments will be performed on AMD XP of the 2.1 GHz computer. Computer system has 512 MB RAM and 7200 rpm hard drive. Our simulations are run using Matlab R13 software.

CPU	Speed	Ram	Operating System	Matlab Edition
AMD XP	2.1 G Hz	512 MB	Windows 2000	Matlab R13

Figure 4.1 Computer configuration and software program.



## CHAPTER 5

### ALGORITHMS

Our goal in this study is not to develop a new face recognition algorithm but to study characteristics of human face through different facial expressions. Hence, we will use well known nearest neighbor algorithm in classification tests. One thing to be determined, though, is as to choose which metric needs to be used when implementing the nearest neighbor.  $L_2$  norm which also known as Euclidian distance is the one that most popularly used among image processing community because of its convenient analytical properties. However, several researchers reported that beside the benefits, Euclidian distance tends to overemphasize the extremes in the image and suggested the use of  $L_1$  norm instead.

In the remainder of this thesis we will first investigate the most appropriate distance measure among some of the well-known metrics for the face recognition applications and then study the effect of different facial expressions on the recognition performance of a face identification system.

#### 5.1 FORMAT of THE EXPERIMENTS

We have conducted extensive simulations on CNL database for our study. All images are retained in their original form and no dimensionality reduction or preprocessing is applied prior to application of algorithms. (Note that images in our database were already taken under controlled environment to eliminate such preprocessing). The format of reporting simulation results is described below.

We have partitioned database into sets of 200, 400, and 600 images that are selected randomly from the database to form a memory set. Depending on the experiment, one, two, three or four images per person are included in the memory. The blank and arbitrary expressions of those individuals in the test image dataset are used to form test sets.

When reporting a result, we have repeated each experiments 5 times (where in each case images in the memory set are selected randomly) and averaged the results. Performance of algorithms is always measured based on the best matching memory image for an input test image

## 5.2 EUCLIDIAN DISTANCE vs. CITY BLOCK

In this section, we will investigate on 3 different metrics that can be used in face recognition systems: Euclidian distance ( $L_2$  norm), City Block distance ( $L_1$  norm) and  $L_{0.5}$  norm. Note that these metrics are by no means the only ones used in image processing community, yet the first two are quite popular among the researchers. The reason for adding the  $L_{0.5}$  norm into our list is because it has been reported that for two-dimensional images  $L_{0.5}$  has outperformed the previous two with a clear margin by Sim at all (T. Sim 2000).

All 3 distance measures mentioned above are known as  $L_p$  similarity measures in the literature and are defined as follows.

Let image  $\mathbf{I}^i$  represent a 2 dimensional matrix of size  $n \times m$  where  $\mathbf{I}^i \in \mathbf{Z}^{n \times m}$  for gray level images and superscript  $i$  refers to index of the image in the database. Let also  $I(x,y)$  denote the gray level pixel value of the image at row  $x$  and column  $y$ . Then, the distance between two images  $\mathbf{I}^1$  and  $\mathbf{I}^2$  is defined in terms of  $L_p$  similarity measure as:

$$L_p(\mathbf{I}^1 - \mathbf{I}^2) = \left( \sum_{x=1}^n \sum_{y=1}^m |I^1(x,y) - I^2(x,y)|^p \right)^{\frac{1}{p}}$$

Below, we demonstrate simulation results in order to compare these three metrics. For the simulations, we have included only one image per person in the memory set which was the blank expression of subjects. This memory set is then tested against blank and arbitrary test expressions as to measure the classification performance of the system.

Figure 5.1 demonstrates the performance of the recognition algorithm while 200, 400 and 600 people were included in the memory (one blank expression per person) and being tested against blank test expressions. From the figure, we could see that for the case of 200, Euclidian distance measure has performed 98.4% accuracy which is slightly better than the other two, whereas for 400 case, all 3 metrics seem to be

performing around 97%. For the case of 600 images though,  $L_{0.5}$  distance performs little better than the other two with 96.9%.

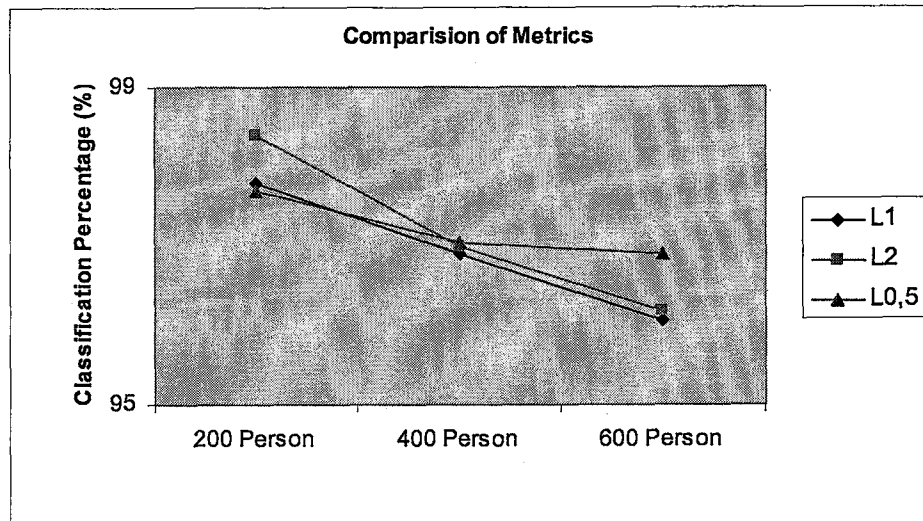


Figure 5.1 Comparison of metrics using blank test expression.

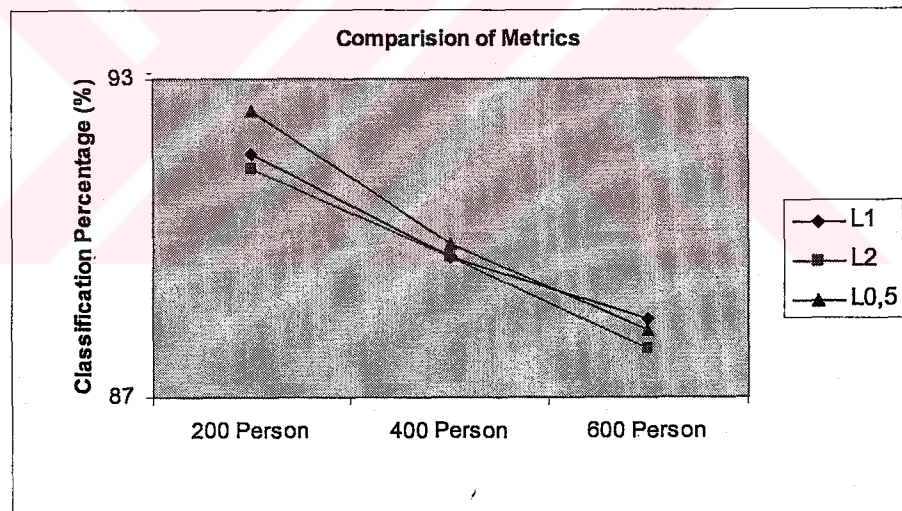


Figure 5.2 Comparison of metrics using arbitrary test expression

Figure 5.2 demonstrates the performance of the similar cases against arbitrary test images. Here, we see that for 200 images in the memory,  $L_{0.5}$  performs 92.4%, which is better than the other two with around 1% margin. For the case of 400, all three perform equally well and for 600 images case, City Block distance slightly outperforms the other two with 88.5%.

The next figure compares the time that it takes for each metric to process a test image. As it can be seen from the Figure 5.3, the processing time of identifying a test

image against a database of 200 individuals for the case of City Block metric is 0.22 sec. This value is 0.37 sec for the remaining two metrics. For databases of 400 and 600 images, we again see the similar trend.

The following comments can be made after analysis of below figures: In terms of recognition performance, none of the 3 metrics clearly outperformed the other two in all cases. On the other hand, by looking at the time requirement, we could say that for applications that need faster retrieval time, City Block distance seems to be a better choice since it performs very close to others in most cases but requires a lot less time to process a test image. However, note that Euclidian distance has analytical properties that make it suitable for the cases where algorithm developments require extensive math. Due to this reason, this metric is quite popularly used.

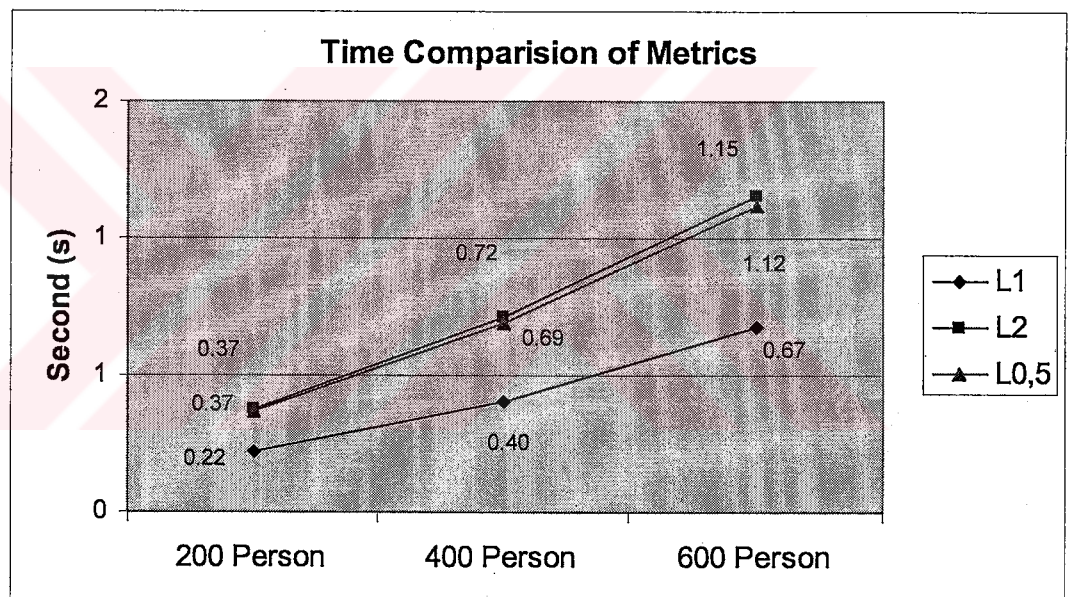


Figure 5.3 Comparison between processing time of an input image for  $L_{0.5}$ ,  $L_1$  and  $L_2$  metrics

For the remaining of this thesis, we will investigate which facial expressions are best suited in order to represent a human face. Since our goal is to compare one expression with the others, the choice of the metric used in distance calculations will not affect the outcome. Hence, in our case, we have decided to use Euclidian distance due to its analytical properties in the remaining simulations.

## CHAPTER 6

### EFFECT of FACIAL EXPRESSIONS

In this chapter, we investigate the role of facial expressions in representation of human face. In particular, we look for which expressions are better suited for classification of an individual. In order to do so, we have conducted the following set of experiments: We have first included only one expression in the memory and performed classification experiments against blank and arbitrary test images. This case is called case 1. In this, our aim was to determine which particular expression was the most effective in representation.

In the second case, we included two expressions per individual into the memory and repeated the simulation. This is referred as case 2. Here, our goal was to understand which combination would be the most appropriate in representation of a human face.

Next, we increased the number of images representing each subject to 3 and 4, which will be referred as case 3 and 4 respectively and performed similar expressions in order to see the effect of multiple expressions on the system performance.

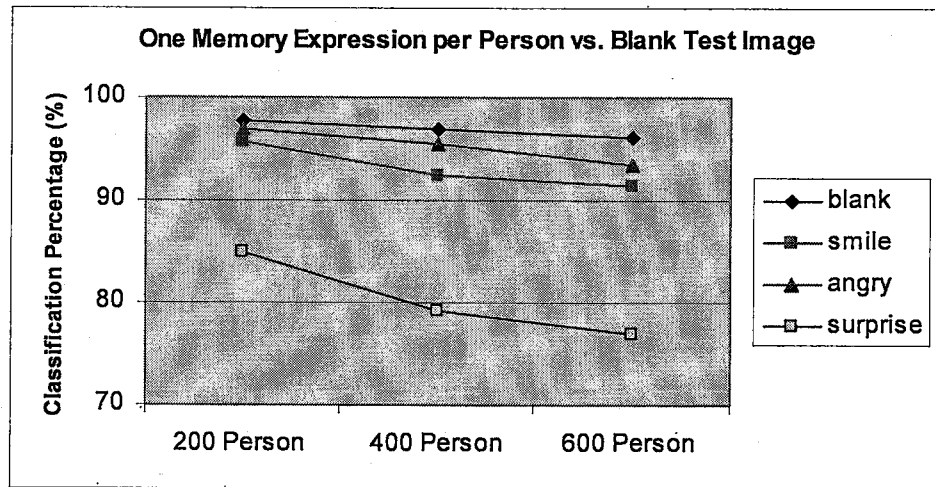
#### 6.1 EXPERIMENTAL RESULT on FACIAL EXPRESSIONS

##### 6.1.1 Case 1

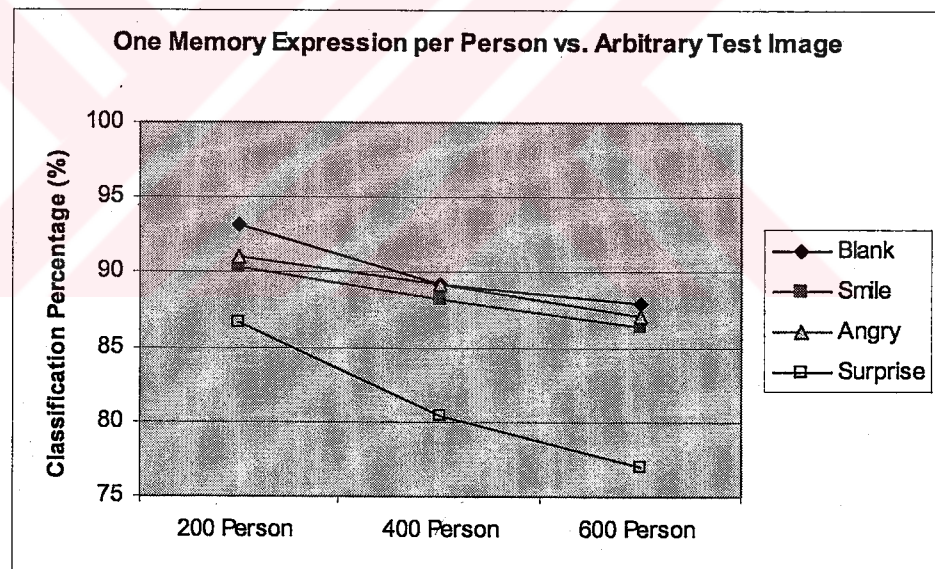
In this section we demonstrate, through simulation results, which expression (or expressions if used together) is the most useful in identifying humans. In order to answer this question, we first included only one expression of each individual in the memory. These were blank, smile, angry and surprise expressions. We have then measured the performance of the algorithm on blank and arbitrary test images. Simulation results for both test image cases are given below.

In the below simulations, the Figure 6.1(a) shows classification performance results for the case of blank test expressions and the Figure 6.1(b) demonstrates results for the arbitrary test expressions. For the blank test images case, the blank memory

expression performs the best as compare to remaining three with 97.8% recognition rate when 200 individuals are used to form the memory set. Angry expression gives a performance of 96.9% and is the second best. The smile and surprise expressions come in the third and fourth places respectively.



(a)



(b)

Figure 6.1 Recognition performances for including one expression in the memory for the cases of (a) blank test image and (b) arbitrary test image

For the case of arbitrary test images, we see a similar pattern. Blank expression performs the highest with 93.2%, which is followed by angry expression performing at 91.0%. The remaining two expressions of smile and surprise give recognition rates of 90.3% and 86.6% and come in third and fourth places respectively.

Note that during the collection of database, subjects are asked to give an unusual expression when snapshot is taken for the arbitrary test image. Hence this expression is quite different from the other expressions. However, the fact that the blank expression is the best performing expression in the case of both test images clearly indicates that this expression is the most suitable one in the identification of a human face.

The fact that angry expression comes as the second best is to be expected because subjects tend to exaggerate the smile expression when they are asked to pose since it is easier to imitate. This causes large amount of changes around mouth area of the face and in turn makes it little harder to recognize him/her. Angry poses, on the other hand, cause changes around eyebrows/eye region of the face and are minor when compared to those in the smile case. It also worth mentioning that during the generation of the database, subjects generally commented on difficulty of posing angry expression when they are not in that mood naturally.

### 6.1.2 Case 2

For this case, we have included 2 expressions per individual in the memory. By doing so, we wanted to find which combination of expressions are the most effective in recognition and also to see if there is a significant performance increase as compared to case 1.

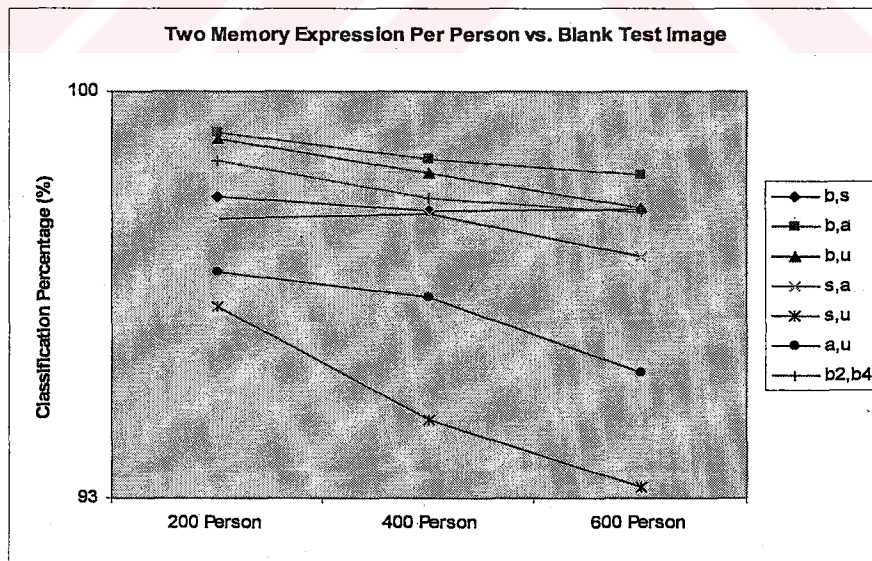


Figure 6.2 Recognition performances for including two expressions in the memory when tested against blank test image set.

Figure 6.2 shows simulation results for the following combination of expressions: blank-smile (b-s), blank-angry (b-a), blank-surprise (b-u), smile-angry (s-

a), smile-surprise (s-u), angry-surprise (a-u), and blank-blank (b2-b4). Note that for the last case, we have selected 2 blank expressions among blank1, blank2, blank3, and blank4 expressions of a subject.

The Figure 6.2 shows performance results when the system is tested against blank test images. From the figure, we see that blank-angry combination is outperforming the rest with 99.3% when 200 people are included in the memory. The same combination consistently performed in the first place when number of people in the memory is increased to 400, and 600. This was expected due to the reasons explained in case 1. Surprisingly enough, blank-surprise combination seems to be performing the second best after the blank angry combination. Unfortunately, we could not bring a clear explanation to this because our expectation was toward having either blank-smile or blank2-blank4 combination to be in this place. We have been conducting more simulations to better understand this case. Blank2-blank4 and blank-smile combinations performed in the third and fourth places with 98.8% and 98.2% recognition rates for the case of 200 subjects in the memory. Simulation results for arbitrary test images of this case are given in the Figure 6.3 below.

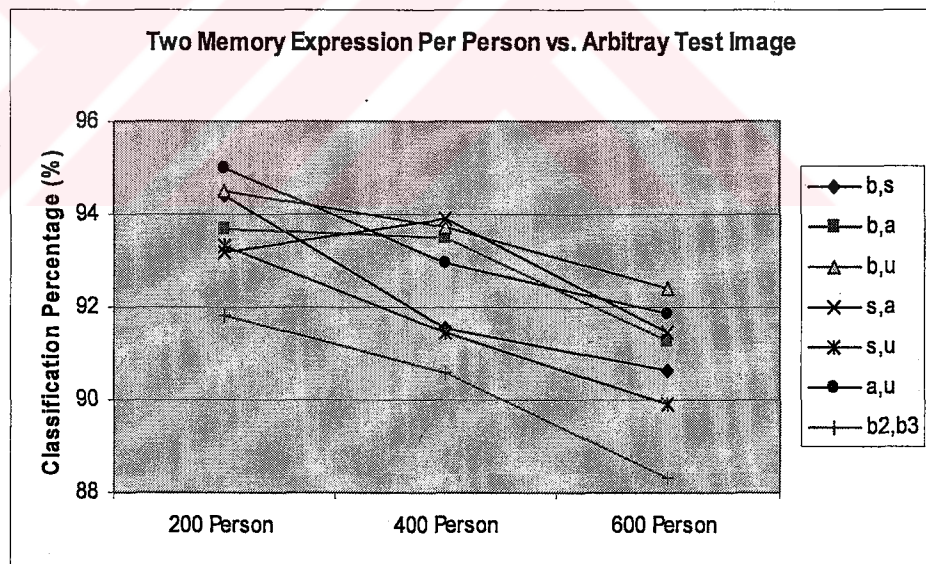


Figure 6.3 Recognition performances for including two expressions in the memory when tested against arbitrary test image set.

For the arbitrary test images, the combinations of angry-surprise, blank-surprise, and blank-smile are ranked in top three places by 95%, 94.5% and 94.0% classification rates respectively for the case of 200 people. For 400, the best 3 performers are smile-angry, blank-surprise and blank-angry combinations with 93.9%, 93.7% and 93.5% respectively. As for the case of 600, blank-surprise, angry-surprise and smile-angry



combinations gave the highest recognition rates with 92.4%, 91.8 and 91.4% respectively.

By looking at the best performing combinations for both test image cases, we see that blank expression is the main contributor when system is tested against blank test images whereas surprise expression (although not as a main contributor) seems to be quite effective for the case of arbitrary test set.

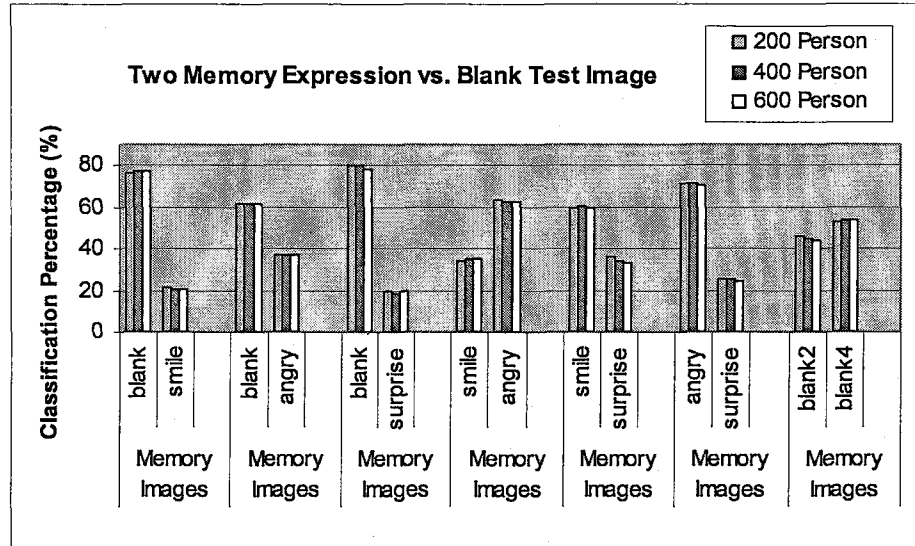
In the case of blank test images, the overall performance of the system is increased by 1.5% when number of images representing each individual is increased to two (i.e. as compared to case 1). This increase is 1.8% when arbitrary test images are used to test the system. Considering that the processing time is doubled when an additional expression is included in the memory, the amount of performance increase does not seem to be significant. This might be of importance for the type of applications that require faster retrieval time and could tolerate small amount of decrease in performance.

In order to look at contributions of individual of expressions to the system performance in all cases, we have plotted below figures.

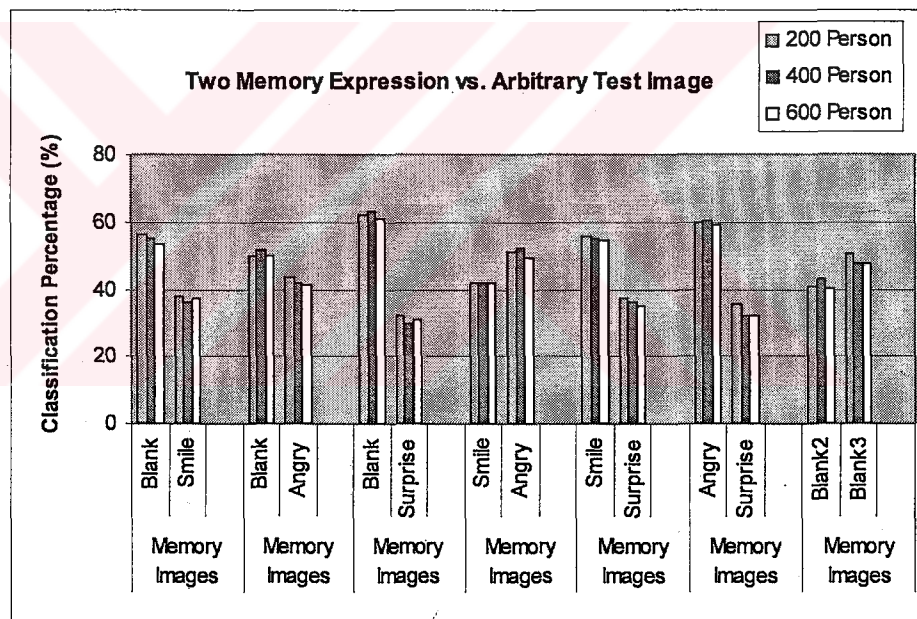
The below figure summarizes the contribution of individual expressions to the performance of the system when 200, 400 and 600 people are included in the memory. For example the first three columns with different colors in Figure 6.4(a) represent the contribution of blank expressions to the system performance for 200, 400, 600 cases when blank-smile combination is used to form the memory set and the next 3 columns show the similar contribution made by the smile expression etc.

Note how contributions of individual expressions are close to be evenly distributed for the case of arbitrary test images whereas in the case of blank test images blank and angry expressions contribute more to the system performance than the other expressions associated with them.

In the Figure 6.4(a), the distribution between blank and angry expressions in the third and fourth columns is in favor of the argument we have made earlier. Here, the first one gets around 60% hit whereas the second one gets around 38%. This difference is clearly a lot more for the blank-smile and the blank-surprise cases. This, in turn, implies how close these two expressions are to each other. Also, note how the percentage is close to being equal for the blank2-blank4 case where the rates are around 45% and 55% respectively.



(a)

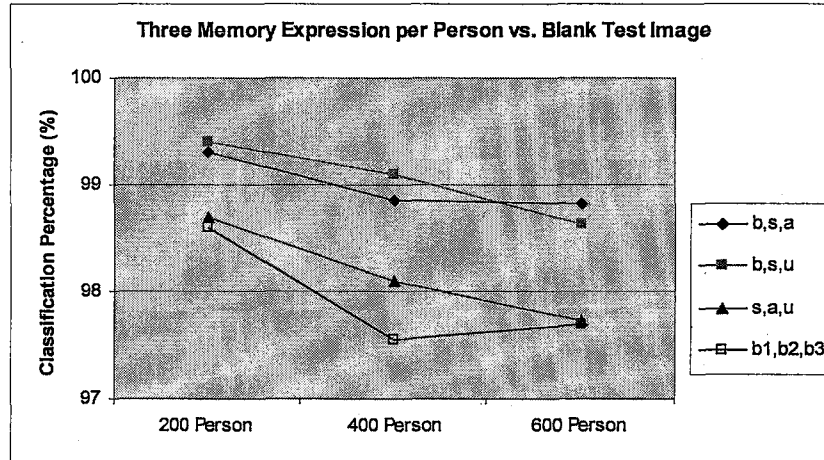


(b)

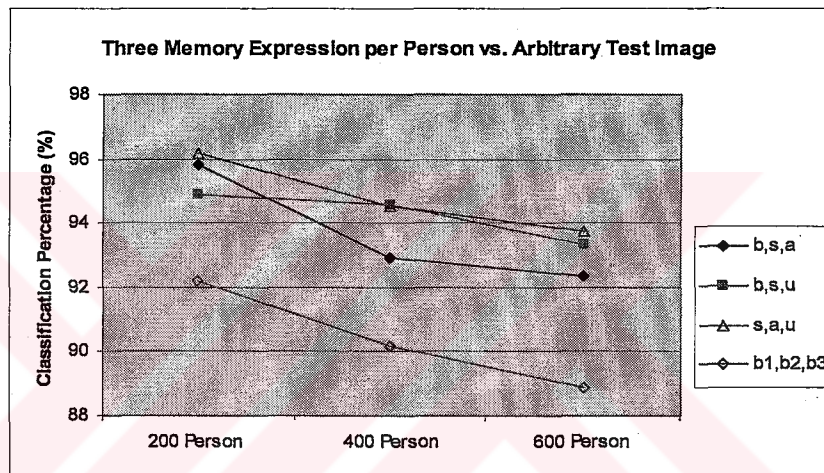
Figure 6.4 Contributions of individual expressions in the case of 2 expressions per individual tested against (a) blank test images set (b) arbitrary test images set.

6.1.3 Case 3

Simulation results for the case of including 3 expressions per person are given below: Here, the possible combinations are blank-smile-angry (b-s-a), blank-smile-surprise (b-s-u), smile-angry-surprise (s-a-u) and blank1-blank2-blank3 (b1-b2-b3).



(a)



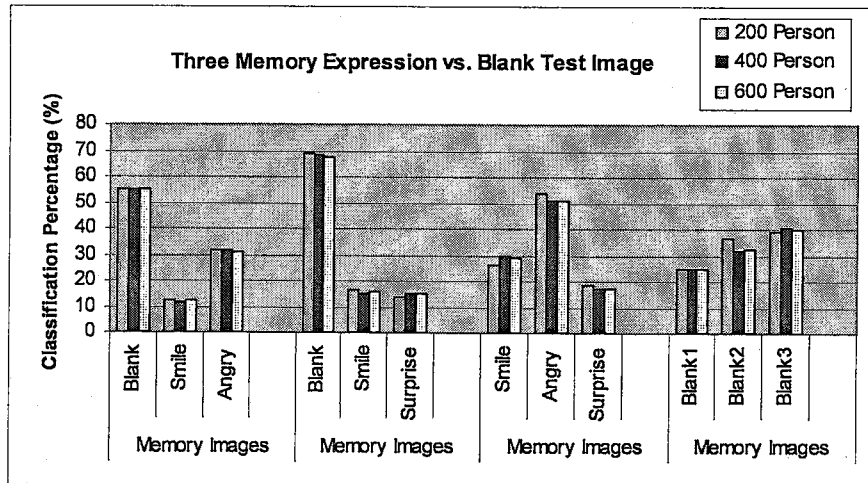
(b)

Figure 6.5 Recognition performances for case of including three expressions in the memory when tested against (a) blank test image set (b) arbitrary test image set.

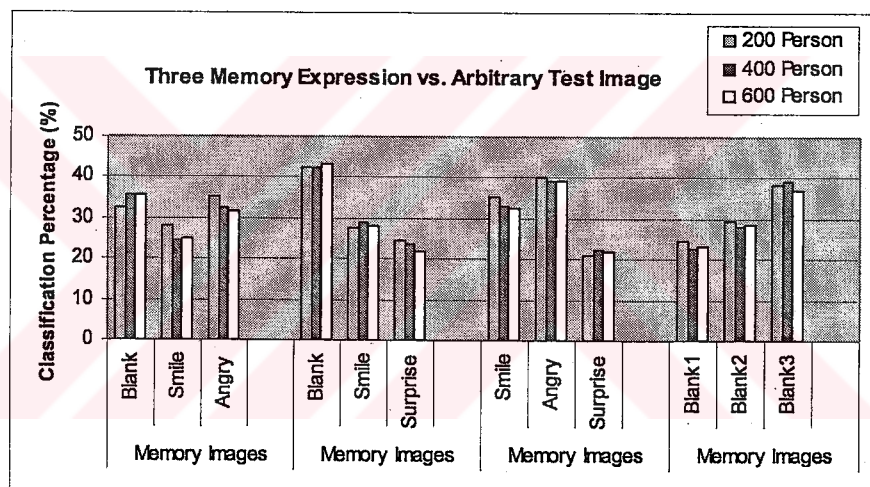
Figure 6.5(a) indicates that, for the case of blank test images, b-s-u combination performs better for 200 and 400 memory sizes with 99.4% and 99.1%. For 600 memory case, though, b-s-a combination outperforms the remaining three with 98.8%. Note how the all-blank (b1-b2-b3) combination is always the least performing. This is because having multiple images of same individual with similar expressions causes distances to be in a close range and hence makes the recognition more difficult. We will have more to say on this when we go to case 4.

In the case of arbitrary test images, smile-angry-surprise and blank-smile-angry combinations perform the best with 96.2% and 95.8% respectively for the case of 200 people. When number of individuals is increased to 400 and then 600 in the memory, blank-smile-surprise and smile-angry-surprise expressions perform in the top two

places. Again, blank1-blank2-blank3 combination performs lowest due to the reason explain above.



(a)



(b)

Figure 6.6 Contributions of individual expressions in the case of 3 expressions per individual tested against (a) blank test images set (b) arbitrary test images set.

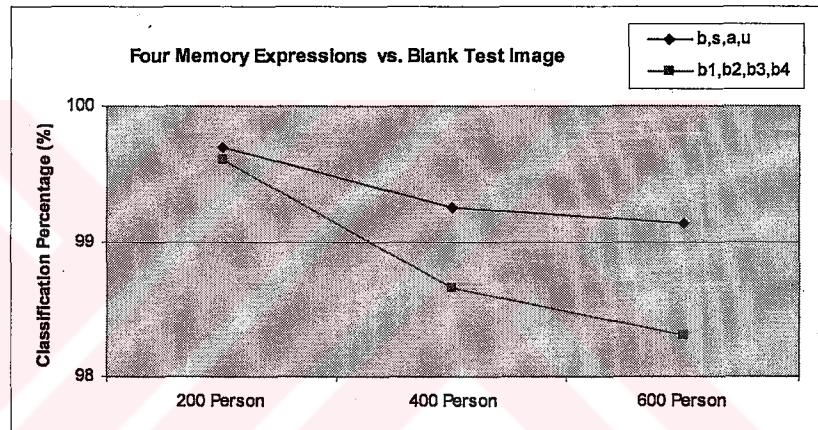
The figure above shows the individual contributions of each expression to the system performance for both blank and arbitrary test image cases.

For the case of blank test images, Figure 6.6(a) shows that when blank-smile-angry expressions are included in the memory, blank and angry expressions are contributing to the performance more with around 55% and 30% whereas the contribution of the smile is around 10%. In the next 3 columns when blank-smile-surprise expressions are used, blank expressions contributes to the overall performance with around 70% whereas the contributions of smile and surprise expressions stays

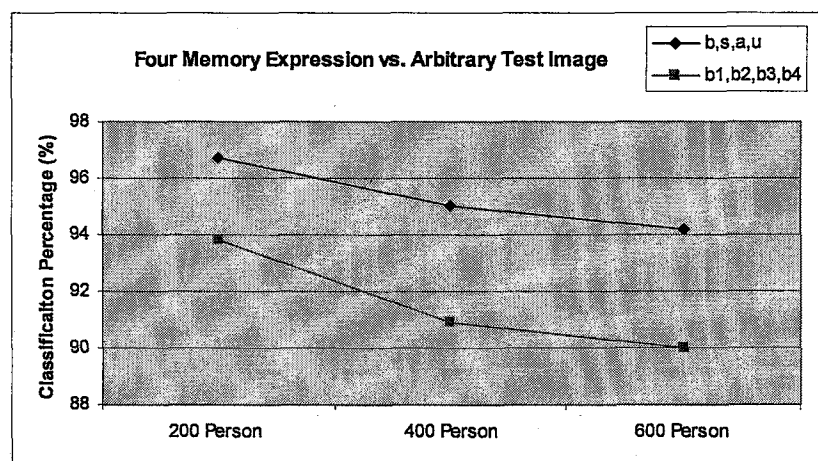
around 15%. Note also how the individual contributions of blank1, blank2, and blank3 are close to each other, which is in the range of 25% to 40% as compared to the other cases. In the case of arbitrary test images, shown in Figure 6.6(b), the contributions are distributed more evenly as expected.

#### 6.1.4 Case 4

In the final case of our simulations, we have included 4 images per person in the memory and performed the classification experiments. Here, we have only experimented on two combinations which are blank-smile-angry-surprise (b-s-a-u) and blank1-blank2-blank3-blank4 (b1-b2-b3-b4) cases. Figure below shows the results of this simulation for both blank and arbitrary test image cases.



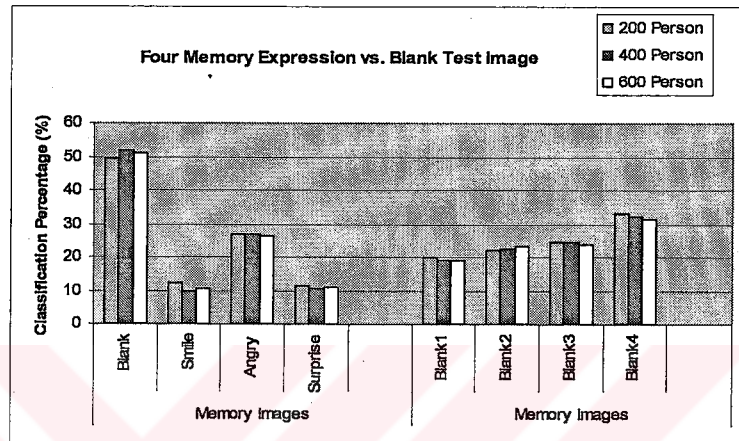
(a)



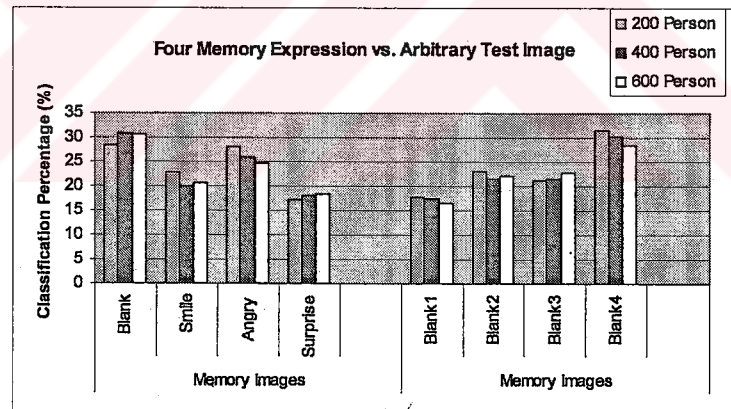
(b)

Figure 6.7 Recognition performances for case of including four expressions in the memory when tested against (a) blank test image set (b) arbitrary test image set.

In the Figure 6.7(a), b-s-a-u combination performs with 99.7% as compare to 99.6% classification in b1-b2-b3-b4 combination when 200 people are used to form a memory set. The performance difference between these two cases increases as number of individuals in memory increases to 400 and 600. In the case of 600, b-s-a-u combination performs with 99.1% whereas the performance of b1-b2-b3-b4 drops to 98.3%. The similar argument is there when system is tested against arbitrary test images as well (see Figure 6.7(b)).



(a)



(b)

Figure 6.8. Contributions of individual expressions in the case of 4 expressions per individual tested against (a) blank test images set (b) arbitrary test images set

It is clear from these and previous set of simulations that when subjects are represented with multiple images in the memory, recognition performance of the system is increased if different facial expressions of individuals are used. This again supports the fact given (Artiklar, 2003), i.e. for face images; the difference between same expressions of different individuals can be less than the difference between different expressions of same individuals. By including more images of the same type

expression, we cause distance measures to be closer when an input image is introduced to the system, which in turn makes the selection of the winner difficult.

Finally the individual contributions of each expression to the system performance are given in the Figure 6.8.

Figure 6.8 indicates that when b-a-s-u combination is used to form a memory set, surprise and smile expressions are the least effective when blank test images are considered and all four expressions are closely effective when arbitrary test images are concerned. As for similar expression case, contributions of individual expressions are close to equal as it was in the previous cases.



## CHAPTER 7

### REDUCED SIZE AND TIME EFFECT

In this section, we investigate the relation between reduced size images and their effect on classification performance. The underlying goal in performing this study is to reveal the fact that whether a linear reduction in the size of a face database pictures would correspond to a linear decrease in the performance of a recognition algorithm as well. If that is not the case, we will then search to find an optimal image size as to compensate for both high classification performance and low recognition time together.

Changing size of images is done by the help of process known as *interpolation*. In another word, interpolation is the process by which we estimate an image value at a location in between image pixels when its size is altered. For example, if one resizes an image so it contains less pixels than it did originally, the interpolation method obtains values of pixels in the new image for the additional pixels through interpolation. There are so many interpolation methods, among them; *bicubic*, *bilinear* and *nearest neighbor* are the most popularly used ones.

After some initial experiments, we chose bilinear interpolation method in order to be used in reducing the size of face images. Detailed explanation of bilinear method can be found in any image processing book. Below, we give a brief explanation of this method.

The interpolation methods generally work in a similar way. To determine a value of an interpolated pixel, you find the point in the input image that the output pixel corresponds. You then assign a value to the output pixel by computing a weighted average of some set of pixels around of that point. The weightings are based on the distance each pixel is from that point. For bilinear interpolation, the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood. If  $(x' y')$  denote the coordinates of a point in the transformed (or reduced) image, and  $v(x' y')$  denote the gray level assigned to it. "For bilinear interpolation, the assigned gray level pixel value is given by;



$$v(x', y') = ax' + by' + cx'y' + d$$

where the four coefficients are determined from the four equations in four unknowns that can be written using the four nearest neighbors of point  $(x' y')$ " (R. C. Gonzales 2002).

We run simulations of the reduced size experiment for four different cases. In the first case, only one expression per person is included in memory. We then introduced two, three and four expressions into the memory for the other cases.

For each of these cases, we reduce size of database images along with the test sets in the amounts of 50%, 25%, 10%, 9%, 8%, 7%, 6%, 5%, and 4% of the original size and then measure the performance of the system for each case against the test sets (i.e.  $82 \times 115 \times 50\% = 41 \times 58$ ).

Differently from previous chapter, we only used one combination of expressions for each case. This is because of the fact that our goal is not to compare different combinations but rather look at the relationship between image size and systems performance. Therefore, we decided on blank type memory expression for the case of one memory expression per individual. For two memory expressions per individual, blank-angry combination is selected. For three memory expressions per individual, blank-smile-angry combination and for four memory expressions, blank-smile-angry-surprise expression combinations are selected.

Reducing Size (%)	Column x Row (pixel)
100	82 x 115
50	41 x 58
25	21 x 29
10	8 x 12
9	7 x 10
8	7 x 9
7	6 x 8
6	5 x 7
5	4 x 6
4	3 x 5

Table 7.1.Reduced Row and Column Size of Image Expression

Table 7.1 shows various reduction values applied to the original image in percent and corresponding pixel by pixel representations. Note that the reduction values are not uniformly distributed, i.e. we jump from 100% to 50% then to 25%. The remaining reduction size though are very closely distributed such as 10%, 9%, 8%, 7%, 6%, 5% and 4%. This is because the performance of the system does not change in significant amount for the initial reduction sizes and hence we have used large difference steps between them. On the other hand, when it comes to 10% and lower reduction sizes, there is sharp performance deterioration. In order to demonstrate this, we have chosen smaller step sizes for these cases.

The format of the running simulations are as follows. We have partitioned the database into randomly selected sets of 200, 400 and 600 individuals. Memory sets are formed by including one, two, three or four expressions of each subject into the memory depending on the case. We have, then, applied all the reduction sizes given in the table 7.1 to the memory set and tested it against blank and arbitrary test sets. Of course, test sets were also subjected to the same reduction in size as memory images. In order to report the results, we run each of the above simulation 5 times where memory sets of each run were formed randomly. We have, then, averaged the results for reporting.

### **7.1 CASE 1: ONE EXPRESSION IN THE MEMORY PER INDIVIDUAL**

In the case of one expression per individual, we have selected blank expression of each subject to be included in the memory. The system is, then, tested for both blank and arbitrary test sets for all the reduction sizes given in Table 7.1.

Figure 7.1 demonstrates performance of the algorithm for this case. It stays close to 97% until the size is reduced to 8% of the original image size. From this point on, we experience sharp performance decrease.

The largest change is from 5% to 4% which corresponds to around 10 percent deterioration. For instance the change is from 93.2% to 87.0% for 200 people. For 400 people, it is from 90.3% to 81.1% and for 600 people performance is decreased from 88.2% to 80.0%.

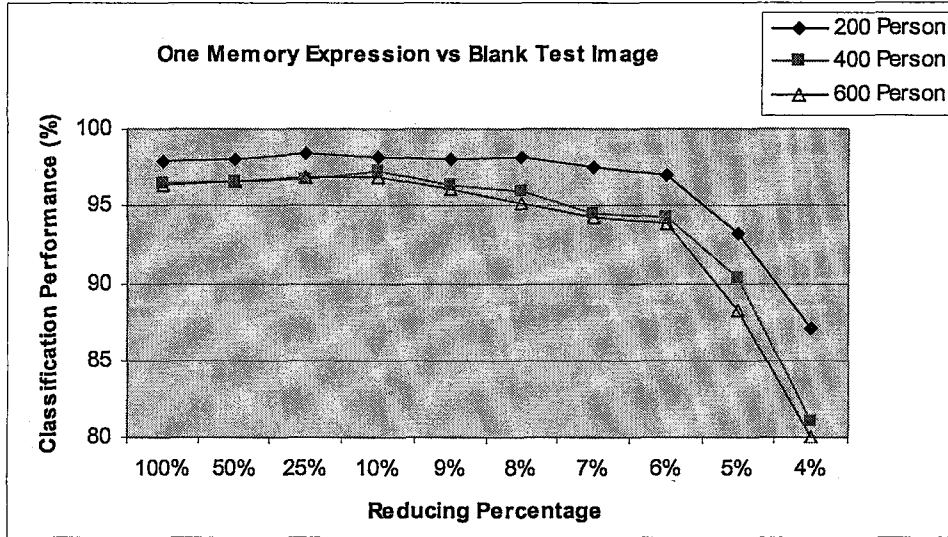


Figure 7.1 Original and Reduced Size Classification Performances for One Memory Expression vs. Blank Type Test Expression.

Figure 7.2 shows performance against the arbitrary test image set for the same cases. There is no significant change in performance until the size is reduced to 8% of original. Generally performance stays around 90% for 200, 400 and 600 people.

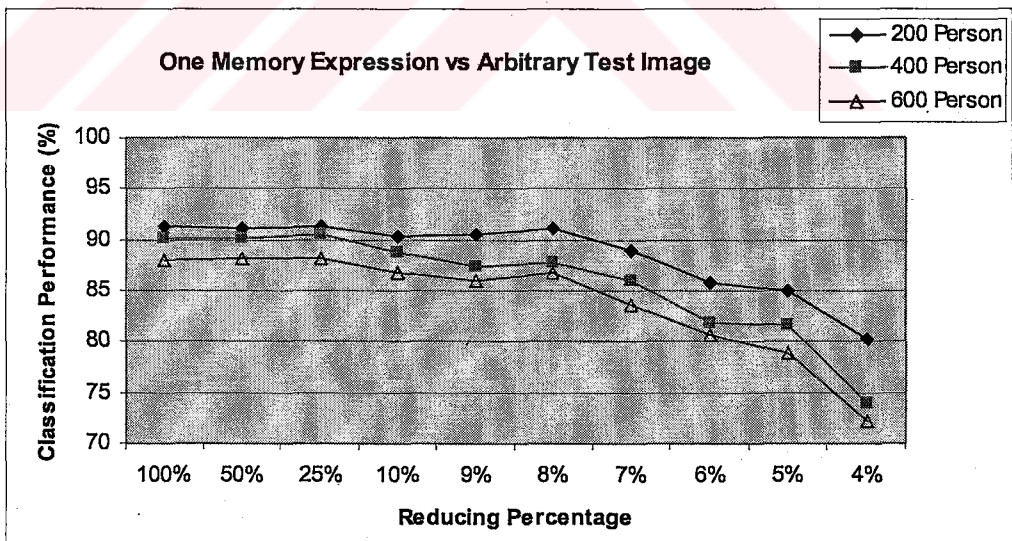


Figure 7.2 Original and Reduced Size Classification Performances for One Memory Expression vs. Arbitrary Type Test Expression.

Significant fall starts from 7% of original size by around 2 percent deterioration for all cases. But noteworthy decrease is at 4% of original size by 11-17 percent deteriorations. The similar argument is true for all 3 memory sizes in Figure 7.2.

According to these simulation results, both blank and arbitrary type test expression performances do not change from 100% to 10% of original size. Performances start to fall down at 7% of original size significantly. Notice, there is a slight increase in performance until 7% of original size in some cases. This is because of the noise embedded to face images due to imperfections in the environment and in the image acquisition set-up. The size reduction can be considered as a low-pass filtering operation which averages out some of these noises and causes the performance to increase.

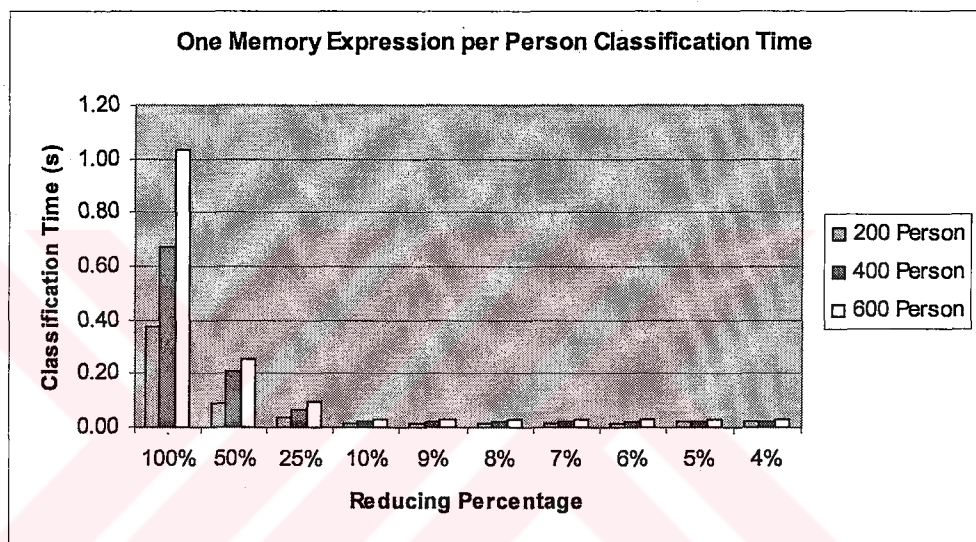


Figure 7.3 Original and Reduced Size Classification Time for One Memory Expressions in Second.

The change in the performance is smoother as the size of the images varies in the case of blank type test images when compared to that in the arbitrary case.

Figure 7.3 shows the time that it takes to classify a test image under all specified reduced image sets. For example the first 3 columns show that the classification time of a test image, when 200 individuals are included in the memory, is 0.38 second. These values are 0.68 and 1.04 second when the number of subjects in the memory is increased to 400 and 600 individuals. Notice that as the amount of reduction applied to memory images increases, the difference between classification times of a test image in all three memory size cases approach to each other. This is because number of pixels becomes close to each other after 10% of original size (Table 7.1). In another word, after %10 or lower reduction of the original size, the number of pixels representing an image became very small and does not make too much of difference in terms of a PC

with gigahertz of processing speed to process them. Finally after the 10% and lower reduction of the original size, classification times become 0.02 second for 200 and 400 individuals, 0.03 second for 600 individuals.

## 7.2 CASE 2: TWO EXPRESSIONS IN THE MEMORY PER INDIVIDUAL

This section demonstrates simulation results for the case when two expressions, blank and angry, are used to form the memory set.

Figure 7.4 demonstrates performance of algorithm for blank type test expression as the size of the images varies. From the figure, system performance is around the 99% when the original size images are used in the memory. Until 10% of original size, there is no significant change in performance. Performances start to decrease at the 7% of original size around 1 percent deterioration. Dramatic performance loss is at 4% of original size by around 10 percent deterioration for all three size memory cases.

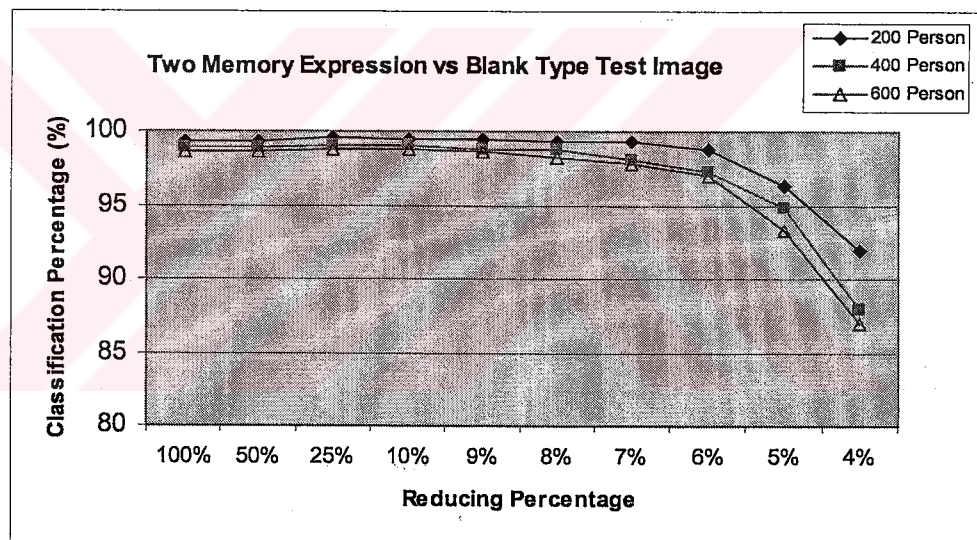


Figure 7.4 Original and Reduced Size Classification Performances for Two Memory Expression vs. Blank Type Test Expression.

Figure 7.5 shows the performance of arbitrary test set for the cases of same reduced image sets. The performance of the system is around 93% for all three memory sizes when the original size images are used. From 100% of original size to 10% of original size there is no significant change in performance. Between 10% and 8%, we see a small fluctuation in performance yet this value is not very large. Starting from 7% of original size there is a significant decrease in classification performance by 1.5 percent deterioration. As in the case of blank type test images the largest performance loss is at the 4% of original size. At this point, performance decreases around 10 percent

as compared to that in original size case. We see a similar trend in terms of change in performance for all three memory size cases.

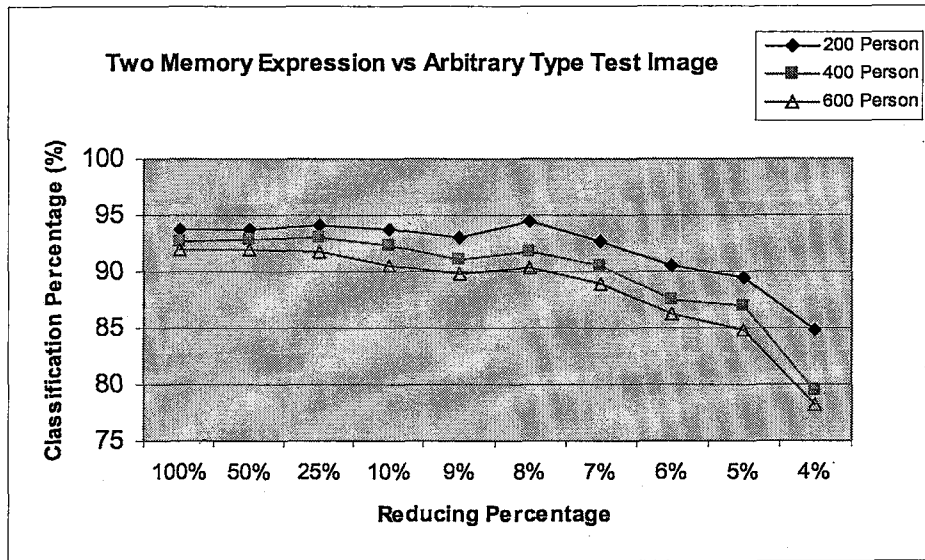


Figure 7.5 Original and Reduced Size Classification Performances for Two Memory Expression vs. Arbitrary Type Test Expression.

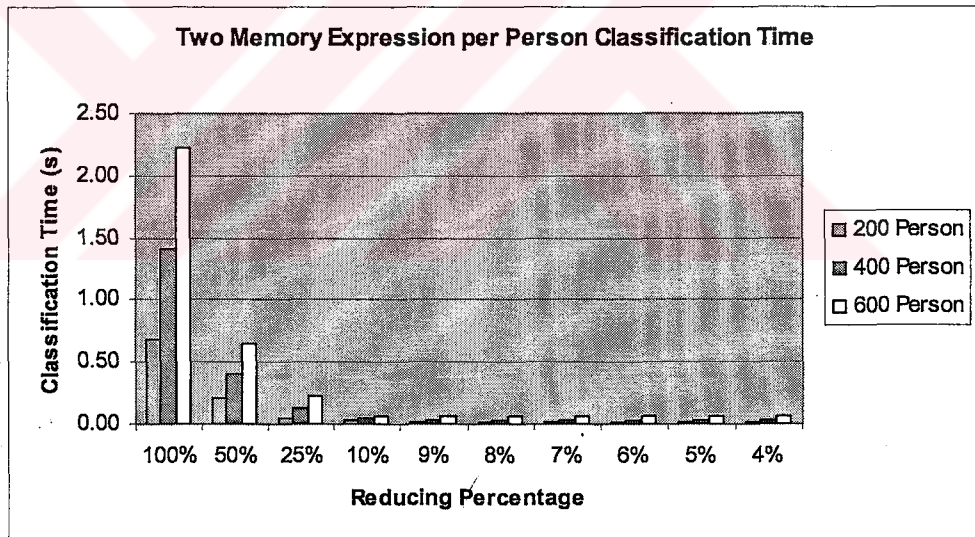


Figure 7.6 Original and Reduced Size Classification Times for Two Memory Expression.

Figure 7.6 show the timing requirements of this case. The classification time of a test image for the case of original size pictures is 0.68 second when measured against 200 individuals in the memory in Figure 7.6. These values are 1.42 and 2.22 second when the number of subjects in the memory is increased to 400 and 600 individuals. Classification time becomes almost constant with 10% of original size. 50% of original size classification times become approximately 4 times faster. At the 25% of original

size, classification times become approximately 11 times faster for all three size memory cases. After the 10% and lower reduction of original size classification times become approximately thirty times faster. Finally for 200 individuals, 10% and lower reduction of the original size classification time is 0.02 second. For 400 individuals, classification time is 0.04 second and for 600 individuals, classification time is 0.06 second.

By looking at the Figures 7.6, it seems that reducing the database images to 10% of their original size give both satisfactory classification performance and fast retrieval time.

### 7.3 CASE 3: THREE EXPRESSIONS IN THE MEMORY PER INDIVIDUAL

In the case of three expressions used per individual, we have selected blank-angry-smile expressions combination of each subject to be included in the memory. The system is, then, tested for both blank and arbitrary test sets.

Figure 7.7 demonstrates performance of the algorithm for this case. Results are similar as in the previous blank type memory cases. There is no significant change in performance up to 10% of the original size. Sharp performance decreases are seen from 7% to 4% of original image sizes. 4% of original size has the largest performance loss. Classification performances decrease around 10 percent at the 4% of original size for all three memory sizes.

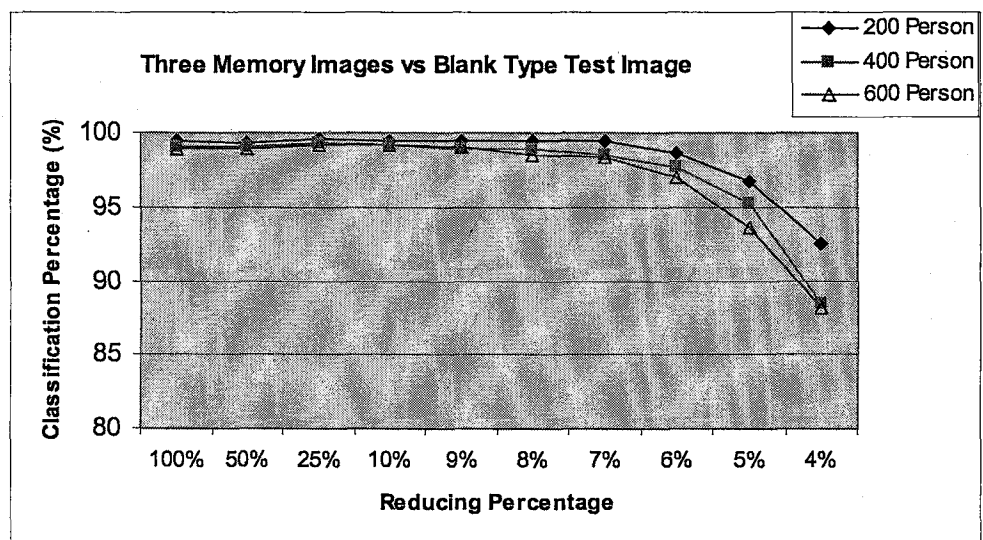


Figure 7.7 Original and Reduced Size Classification Performances for Three Memory Expression vs. Blank Type Test Expression.

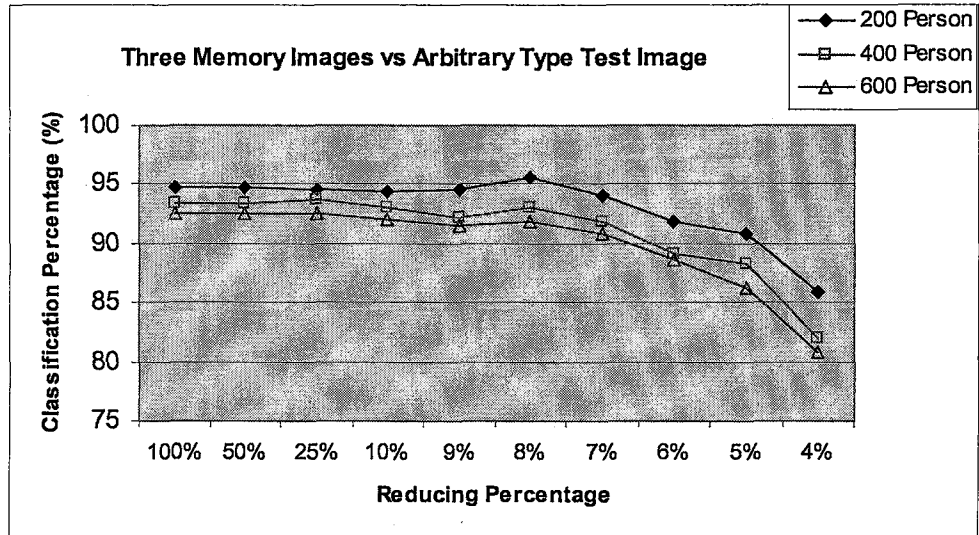


Figure 7.8 Original and Reduced Size Classification Performances for Three Memory Expression vs. Arbitrary Type Test Expression.

Figure 7.8 shows similar classification results for arbitrary type test expression. There is no change performance significant until the 9% of original size. Starting from 7% of original size, we start to observe performance deterioration. 4% of original sizes decrease the performances by around 10 percent deterioration for all three memory size.

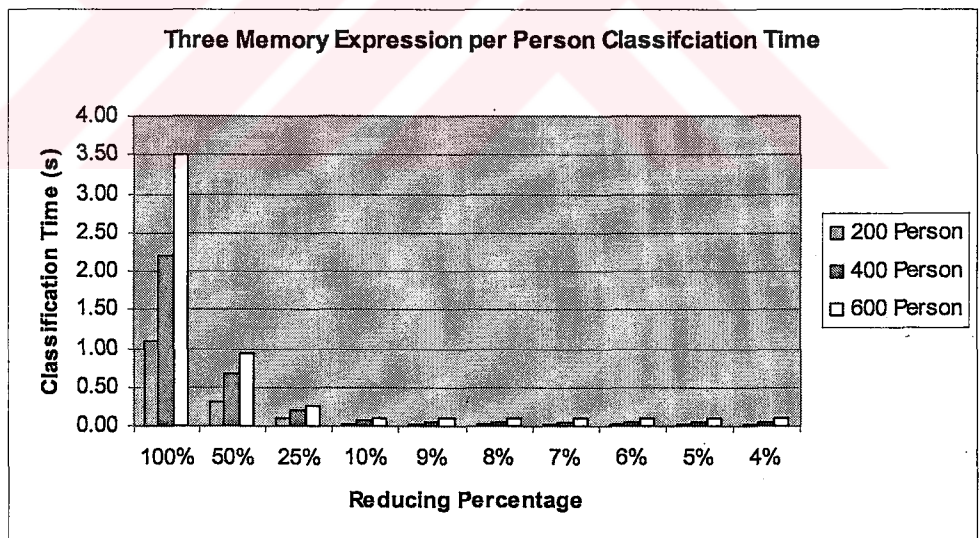


Figure 7.9 Original and Reduced Size Classification Times for Three Memory Expressions.

Figure 7.9 represents classification time requirement of this case. The above figure is very close to what we had in previous 2 cases. The classification times are 1.1, 2.2 and 3.5 seconds when the original size images are used to form memory sets of 200,



400 and 600 individuals. When image sizes are reduced to 10% of their original size or lower, classification times become thirty times faster approximately.

#### 7.4 CASE 2: FOUR EXPRESSIONS IN THE MEMORY PER INDIVIDUAL

We performed similar experiments by including 4 expressions in the memory which were blank, smile, angry and surprise expressions combination.

Figures 7.10 and 7.11 shows classification performance of this case for blank and arbitrary test sets and Figure 7.12 shows the time that it takes for an input image to be classified for specified reduced memory sets. By looking at the figures, we can conclude that an additional expression into the memory does not change the overall behavior of the algorithm in terms of performance. An obvious difference is seen in terms of classification time given in Figure 7.12 when compared to previous cases. Because of the additional expression in the memory, it takes longer for the system to classify the input test image. Classification times are 1.45, 3.05 and 4.82 seconds when original size images are used to form memory sets of 200, 400 and 600 individuals. Eventhough these classification times are four times more when one memory images used in memory but under the 10% of original size; classification times become 0.04, 0.08 and 0.14 seconds for all three size memory cases.

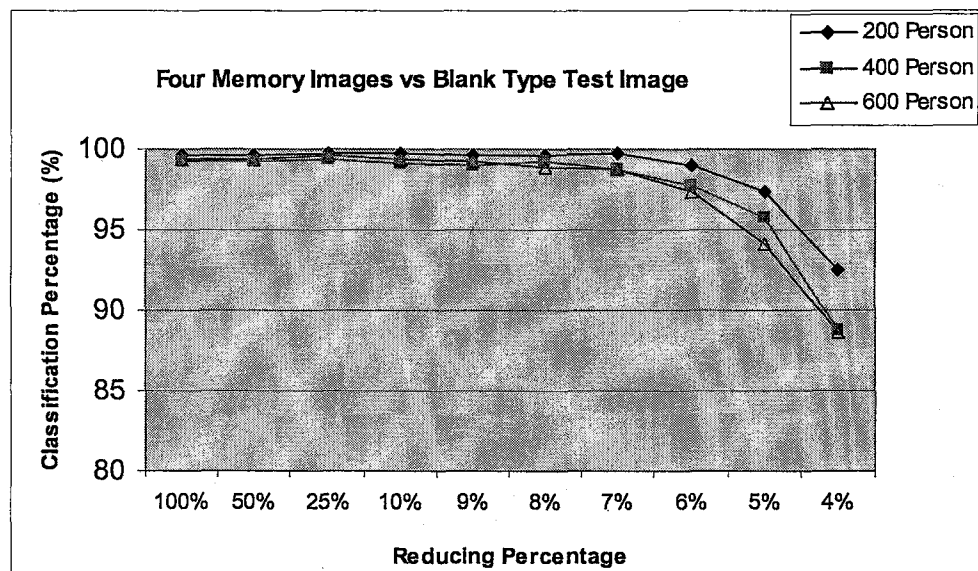


Figure 7.10 Original and Reduced Size Classification Performances for Four Memory Expression vs. Blank Type Test Expression.

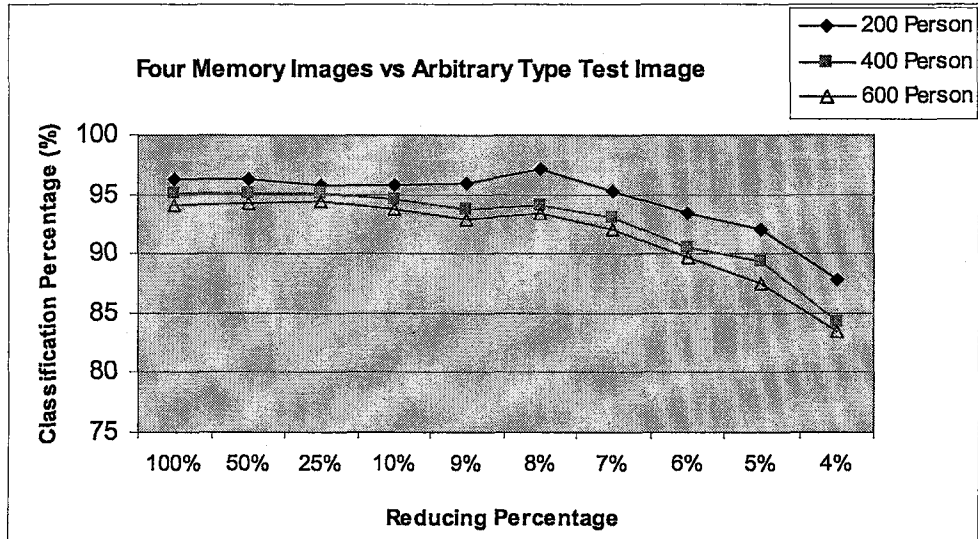


Figure 7.11 Original and Reduced Size Classification Performances for Four Memory Expression vs. Arbitrary Type Test Expression.

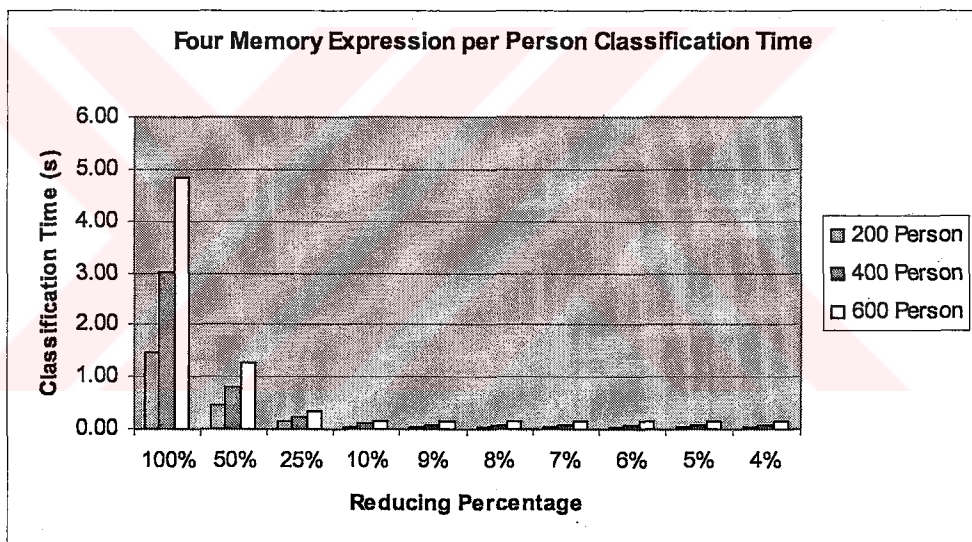


Figure 7.12 Original and Reduced Size Classification Times for Four Memory Expressions.

## CHAPTER 8

### SUMMARY AND FUTURE WORK

#### 8.1 SUMMARY

The objective of this thesis was to study the effect of expressions on performance of a face recognition system and to demonstrate the relation between size of images retrieval time of the system. In order to achieve these goals, we have demonstrated the following facts:

$L_2$  and  $L_{0.5}$  norms take almost same time to process an input image and their effect on the classification performance is also very close. Hence one does not yield a better distance computation than the other. The  $L_1$  norm, on the other hand, needs almost as half of the time as compared to other two in order to calculate a distance between an input and a database image. The effect of the  $L_1$  norm on the performance is a little worse than the other two in most cases but the difference is very small. This feature of  $L_1$  norm would make it very appropriate for real time applications.

Second, in the representation of human face, we have found that blank and angry expressions are the most effective. These two performed the highest when only one expression is used to describe a face. We have also showed that using multiple images increases the performance of the system. However, the increase was not as significant as we expected. When represented with multiple images, blank, angry, and surprise expressions were proved to be the most effective. The surprise expression was mainly a contributor for arbitrary test images. We have showed that representing a face by different facial expressions improves the performance of the system when compared to that of using similar expressions.

Third, we showed that reducing the size of database images from their original size does not necessarily yields a worse classification performance. In fact, reducing the size of images as much as 10% of their original still yielded a very close performance compared to that in the full size. The retrieval time however was improved as much as thirty times.

## **8.2 FUTURE WORK**

Multiple expressions of an individual can be averaged to get an approximate image and can be used in identification task. This approach can help to lower the retrieval time of an input image since it has to only be compared to one average image of each individual in the database rather than multiple expressions of each. However, the classification performance of the system can deteriorate.

Another improvement to this study is to consider normalization of face images. In our study, we have used database images in their original form. However there were variations in term translation, rotation and light illumination among database images due to imperfect conditions during the collection of this database. Artiklar has showed that normalizing images in terms of slight translation and rotation can improve classification performance of recognition algorithm (Artiklar, 2002). Further improvement can be obtained by considering normalization procedure that can take care of some of the illumination changes embedded in database images.

# APPENDIX

## MATLAB FUNCTIONS

### A.1 COMPARISON of METRICS

#### A.1.1 CB, ED and L0.5 Norm Comparison Function

```

%Metrics: Compare test database and memory database by a Metrics.
%[index,winner]=MetoneM(metric,iType,pnumber,Mtype,Ttype,HT)

%OUTPUTS:
%index : index number reading pictures from database
%winner: this array PNUMBERx10 array include 10 winners and distances from memory database for
%every test images.

% INPUTS:
%pnumber : numbers of picture are going to read from the memorydatabase and testdatabase.
%lengthpic : this is the length of the only one picture matrix.
%pertypeM-pertypeT : this is the type of the picture is going to read from test and memory location.
%metric: this value select a metrics;
%CB=City Block
%ED=Euclidean Distance
%NM=L 0.5 norm

function [index,winner]=MetoneM(metric,iType,pnumber,Mtype,Ttype,HT)

lengthpic=9430;
% chosing index number random or normal type
if iType=='n'
    index=1:pnumber;
else iType=='r'
    index1=su(pnumber);
    index=sort(index1);
end
matmemory=RMreader(pnumber,lengthpic,Mtype,index); % 100 pictures are reading and as a length 9430 in
memory database
winner=zeros(pnumber*2,10);

for perftestpic=1:pnumber
    % ===== read test images =====
    % read files from the directory and load in mfilename pointer.
    tname=sprintf('C:\images\test\image%d%s.raw', index(:,perftestpic), Ttype);
    fpmem=fopen(tname,'r');

    % check to error fp pointer
    if (fpmem<0)
        error('Files or a file could not read on this path')
        exit(0);
    end
end

```

```

% reading to memory files to imgmem (1x9430) row matrix.
[testing,countmem]=fread(fpmem,[1 inf]);

% writing to whole row matrix into the test-files matrix.

fclose(fpmem);

%=====

perfdistance=0;
if metric=='CB'
    %City Block: measuring the distance test image and memory images
    for imempic=1:pnumber

        perfabs=abs(testing-matmemory(imempic,:));
        [sp,sf]=size(perfabs);
        perfint=sum(perfabs);
        perfdistance(:, imempic)=perfint;

    end
end
if metric=='ED'
    %Euclidean Distance: measuring the distance test image and memory images
    for imempic=1:pnumber

        perfabs=(testing-matmemory(imempic,:)).^2;
        perfabs=sqrt(perfabs);
        [sp,sf]=size(perfabs);
        perfint=sum(perfabs);
        perfdistance(:, imempic)=perfint;

    end
end
if metric=='NM'
    %L0.5 norm: measuring the distance test image and memory images
    for imempic=1:pnumber

        perfabs=sqrt(abs(testing-matmemory(imempic,:)));
        perfabs=(perfabs).^2;
        [sp,sf]=size(perfabs);
        perfint=sum(perfabs);
        perfdistance(:, imempic)=perfint;

    end
end

%finding first 10 winner
for i=1:10
    [mini indice]=min(perfdistance);
    win1(1,i)=indice;
    win2(1,i)=mini;
    perfdistance(:,indice)=9999999;
end

%writing first 10 winner in winner matrix
winner((perftestpic*2)-1,:)=win1(1,:);
winner((perftestpic*2),:)=win2(1,:);
perftestpic
end
clc
% writing to performance result excel files
filename1XLS=sprintf('MetriconeMperform-%s-%s-%d-%d.xls',Mtype,Ttype,pnumber,HT);
fid=fopen(filename1XLS,'w');
fprintf(fid,'DATABASE INDEX NUMBER AND Metrics Block DISTANCES \n\n');

```

```

if iType == 'n'
    fprintf(fid,'database index number selected normally')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end

if iType == 'r'
    fprintf(fid,'database index number selected randomly')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end

fclose(fid);

```

### A.1.2 Result of Comparison Function

%MetoneMR: This function use outputs of MetoneM function. Calculate  
%classification,misclassification,classification time and Classification  
%Percentage.

```
[claspictures,misclaspictures,time,PERcentage]=MetoneMR(iType,pnumber,Mtype,Ttype,HT);
```

%OUTPUTS:

%claspictures : Classified pictures index values.

%misclaspictures : Misclassified pictures index values.

%time : Classification time.

%PERcentage : Classification Percentage.

```
function [claspictures,misclaspictures,time,PERcentage]=MetoneMR(iType,pnumber,Mtype,Ttype,HT);
```

```
t = clock;
```

```
[index,winner]=MetoneM(metric,iType,pnumber,Mtype,Ttype,HT);
```

```
time=etime(clock,t);
```

```
COMindex=1:pnumber;
```

```
winnerR=zeros(1,pnumber);
```

```
winnerR=winnerR';
```

```
for i=0:pnumber-1
```

```
    winnerR(i+1,:)=winner(2*i+1,1);
```

```
end
```

```
% finding misclassification pictures and their index number
```

```
claspictures=0;
```

```
misclaspictures=0;
```

```
Mismatrix=0;
```

```
for j=1:pnumber
```

```
    if winnerR(j)==COMindex(j)
```

```
        claspictures=claspictures+1;
```

```
    else winnerR(j)~=COMindex(j)
```

```
        misclaspictures=misclaspictures+1;
```

```
        Mismatrix(j,:)=1;
```

```
    end
```

```
end
```

```
% calculation of total mismatched picture number
```

```
totalmismatched=sum(Mismatrix);
```

```
mismatchindex=find(Mismatrix);
```

```
mismatched=winnerR(mismatchindex);
```

```
%calculation of Percentage of The Performance
```

```

PERcentage=100*(pnumber-totalmismatched)/pnumber;

% writing to result excel files
filenameXLS=sprintf('MetoneMresult-%s-%s-%d-%d.xls',Mtype,Ttype,pnumber,HT);

fid=fopen(filenameXLS,'w');

fprintf(fid,'Classified pictures number %3.2f \n',claspictures);
fprintf(fid,'Matching Performance Time is %4.3f second \n',time);
fprintf(fid,'Matching Performance Percentage is %3.2f percent \n',PERcentage);

fprintf(fid,'Misclassified Pictures Index\n');
fprintf(fid,'%3.2f\n',mismatchindex);

fprintf(fid,'Mismatched Pictures \n');
fprintf(fid,'%3.2f\n',mismatched);

%index for random pictures number
indres=1:pnumber;
indres(2,:)=index;
indres=indres';

if iType == 'r'
fprintf(fid,'Database Picture Index \n');
for i=1:pnumber
    fprintf(fid,'\n');
    fprintf(fid,'%3.3f',indres(i,:));
end
end

fclose(fid);

```

### A.1.3 Summary of Result Function

%MetoneMTestSUMM : This funtion run simulation 5 times and calculate average %value for this simulation results, than writes average results in excel file.  
 %MetoneMTestSUMM(iType,pnumber,Mtype,Ttype,TR);

```

function MetoneMTestSUMM(iType,pnumber,Mtype,Ttype,TR);
for HT=1:TR
    [claspictures,misclaspictures,time,PERcentage]=MetoneMR(iType,pnumber,Mtype,Ttype,HT);
    b(HT,:)=claspictures;
    d(HT,:)=misclaspictures;
    Ttime(HT,:)=time;
    PERcent(HT,:)=PERcentage;
end
Class=sum(b)/TR;
misClass=sum(d)/TR;
TTime=sum(Ttime)/TR;
PerCent=sum(PERcent)/TR;

fileNameXLS=sprintf('MetoneMSummaryRes-%s-%s-%d.xls',Mtype,Ttype,pnumber);
fid=fopen(fileNameXLS,'w');

fprintf(fid,'Summary of the %d Times Face Recognition Simulation \n\n',TR);

fprintf(fid,' Average of The Classified Pictures Number is %3.2f \n',Class);
fprintf(fid,' Average of The MIS-Classified Pictures Number is %3.2f \n',misClass);
fprintf(fid,' Average Classification Time is %3.2f \n',TTime);
fprintf(fid,' Average Classification Percentage is %3.2f \n',PerCent);

fclose(fid);

```



## A.2 ONE MEMORY IMAGE SIMULATION FUNCTIONS (E.D.)

### A.2.1 One Memory Main Function

```

%EDoneMRedSi: This function reads one memory image and test image and
%than compare them.
%[index,winner]=EDoneMRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method)
%OUTPUTS:
%index :index number reading pictures from database.
%winner: this array PNUMBERx10 array include 10 winners and distances from memory database for
%every test images.
% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%Mtype,Ttype : This is the type of the picture is going to read from test and memory location.
%HT         : This value show to program "how many times this simulation will run for different index database."
%reduceP    : Shows to reduce percentage for row and column.
%method     : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

function [index,winner]=EDoneMRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method)

lengthpic=9430;

% chosing index number random or normal type
if iType=='n'
    index=1:pnumber;
else iType=='r'
    indexname = sprintf('C:\\MATLAB6p5\\work\\randomDATA\\indexfile-%d-%d.data',HT,pnumber);
    index=textread(indexname,'%f');
    index=index';
end

matmemory=RMreducesize(pnumber, lengthpic, Mtype, index,reduceP,method); % 100 pictures are reading and as a
length 9430 in memory database

winner=zeros(pnumber*2,10);

for perfstestpic=1:pnumber

    % ===== read test images =====
    % read files from the directory and load in mfilename pointer.
    tname=sprintf('C:\\images\\test\\image%d%s.raw', index(:,perfstestpic), Ttype);
    fpmem=fopen(tname,'r');

    % check to error fp pointer
    if (fpmem<0)
        error('Files or a file could not read on this path')
        exit(0);
    end

    % reading to memory files to imgmem (1x9430) row matrix.
    [testing,countmem]=fread(fpmem,[1 inf]);
    %
    %
    if reduceP ~= 100;
        testing=resizeres(testing,reduceP,method);
    else reduceP == 100;

```

```

testing=testing;
end
%=====
%=====
% writing to whole row matrix into the test-files matrix.

fclose(fpmem);
%=====

perfdistance=0;

%measuring the distance test image and memory images
for imempic=1:pnumber

    perfabs=(testing-matmemory(imempic,:)).^2;
    perfabs=sqrt(perfabs);
    [sp,sf]=size(perfabs);
    %this line cancelled to average of distance
    %perfint=sum(perfabs)/sf;
    perfint=sum(perfabs);
    perfdistance(:, imempic)=perfint;

end
%finding first 10 winner
for i=1:10
    [mini indice]=min(perfdistance);
    win1(1,i)=indice;
    win2(1,i)=mini;
    perfdistance(:,indice)=9999999;
end

%writing first 10 winner in winner matrix
winner((perftestpic*2)-1,:)=win1(1,:);
winner((perftestpic*2),:)=win2(1,:);
perftestpic
end
clc
% writing to performance result excel files
filename1XLS=sprintf('EDoneMRedSiperform-%s-%s-%d-%d-%d.xls',Mtype,Ttype,pnumber,reduceP,HT);
fid=fopen(filename1XLS,'w');
fprintf(fid,'DATABASE INDEX NUMBER AND City Block DISTANCES \n\n');
if iType == 'n'
    fprintf(fid,'database index number selected normally')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end
end

if iType == 'r'
    fprintf(fid,'database index number selected randomly')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end
end

fclose(fid);

```

## A.2.2 Result of Main Function

%EDoneMRRedSi : This function use outputs of the EDoneMRedSi function.  
 %Takes the winner matrix from EDoneMRedSi functions. Calculate  
 %Classification, Misclassification, Classification Time and Classification

```

%Percentage.
%[claspictures,misclaspictures,time,PERcentage]=EDoneMRRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method);
%OUTPUTS:
%claspictures : gives the classification number based on index number.
%misclaspictures : this outputs gives misclassification index.
%time : gives classificaiton time.
%PERcentage : calculate the classification performance percentage.

%INPUTS:
%iType : To define index type to read database images.
%pnumber : Numbers of picture are going to read from the memorydatabase and testdatabase.
%Mtype,Ttype : This is the type of the picture is going to read from test and memory location.
%HT : This value show to program "how many times this simulation will run for different index database."
%reduceP : Shows to reduce percentage for row and column.
%method : Shows to interpratation method like ,'bilinear','bicubic','nearest'.

function
[claspictures,misclaspictures,time,PERcentage]=EDoneMRRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method);

t = clock;
[index,winner]=EDoneMRRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method);
time=etime(clock,t);

COMindex=1:pnumber;

winnerR=zeros(1,pnumber);
winnerR=winnerR';

for i=0:pnumber-1
    winnerR(i+1,:)=winner(2*i+1,1);
end

% finding misclassification pictures and their index number
claspictures=0;
misclaspictures=0;
Mismatrix=0;
for j=1:pnumber
    if winnerR(j)==COMindex(j)
        claspictures=claspictures+1;
    else winnerR(j)~=COMindex(j)
        misclaspictures=misclaspictures+1;
        Mismatrix(j,:)=1;
    end
end

% calculation of total mismatched picture number
totalmismatched=sum(Mismatrix);
mismatchindex=find(Mismatrix);
mismatched=winnerR(mismatchindex);

%calculation of Percentage of The Performance

PERcentage=100*(pnumber-totalmismatched)/pnumber;

% writing to result excel files
filenameXLS=sprintf('EDoneMRRedSireult-%s-%s-%d-%d-%d.xls',Mtype,Ttype,pnumber,reduceP,HT);

fid=fopen(filenameXLS,'w');

fprintf(fid,'Classified pictures number %3.2f \n',claspictures);
fprintf(fid,'Matching Performance Time is %4.3f second \n',time);
fprintf(fid,'Matching Performance Percentage is %3.2f percent \n',PERcentage);

fprintf(fid,'Misclassified Pictures Index\n');
fprintf(fid,'%3.2f\n',mismatchindex);

```

```

fprintf(fid,'Mismatched Pictures \n');
fprintf(fid,'%3.2f\n',mismatched);

%index for random pictures number
indres=1:pnumber;
indres(2,:)=index;
indres=indres';

if iType == 'r'
fprintf(fid,'Database Picture Index \n');
for i=1:pnumber
    fprintf(fid,'\n');
    fprintf(fid,'%3.3ft',indres(i,:));
end
end

fclose(fid);

```

### A.2.3 Summary of Result Function

```

%EDoneMTestSUMMRedSi : Run 5 times EDoneMRRedSi function and takes the outputs of EDoneMRRedSi
%and calculate Reduced Size Row and Column,Classified and Mis-Classified Pictures
%Number,Time and Classification Percentage, and finally writes all results
%in excel files.
%EDoneMTestSUMMRedSi(iType,pnumber,Mtype,Ttype,TR,reduceP,method);

%INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%Mtype,Ttype : This is the type of the picture is going to read from test and memory location.
%TR         : This value show to program "how many times this simulation will run for different index database."
%reduceP    : Shows to reduce percentage for row and column.
%method     : Shows to interpratation method like 'bilinear','bicubic','nearest'.

function EDoneMTestSUMMRedSi(iType,pnumber,Mtype,Ttype,TR,reduceP,method);

for HT=1:TR

[claspictures,misclaspictures,time,PERcentage]=EDoneMRRedSi(iType,pnumber,Mtype,Ttype,HT,reduceP,method);
    b(HT,:)=claspictures;
    d(HT,:)=misclaspictures;
    Ttime(HT,:)=time;
    PERcent(HT,:)=PERcentage;
end
Class=sum(b)/TR;
misClass=sum(d)/TR;
TTime=sum(Ttime)/TR;
PerCent=sum(PERcent)/TR;

rrow=82;
rcolumn=115;
rproW=round((rrow*reduceP)/100);
rpcolumN=round((rcolumn*reduceP)/100);

fileNameXLS=sprintf('EDoneMRRedSiSummaryRes-%s-%s-%d-%s-%d.xls',Mtype,Ttype,reduceP,method,pnumber);
fid=fopen(fileNameXLS,'w');

fprintf(fid,'Summary of the %d Times Face Recognition Simulation \n\n',TR);

fprintf(fid,' Reduce Percentage of 82x115 is %3.2f \n',reduceP);
fprintf(fid,' Reduce Pixel Values of Row Column %3.2f \n',rproW,rpcolumN);
fprintf(fid,' Average of The Classified Pictures Number is %3.2f \n',Class);
fprintf(fid,' Average of The MIS-Classified Pictures Number is %3.2f \n',misClass);
fprintf(fid,' Average Classification Time is %3.2f \n',TTime);
fprintf(fid,' Average Classification Percentage is %3.2f \n',PerCent);

```

```
fclose(fid);
```

### A.3 TWO MEMORY IMAGES SIMULATION FUNCTIONS (E.D.)

#### A.3.1 Two Memory Main Function

```
%EDtwoMRedSi: This function reads two memory image and one test image and
%than compare them.
%[index,winner]=EDtwoMRedSi(iType,pnumber,M1type,M2type,Ttype,HT,reduceP,method);

%OUTPUTS:
%index :index number reading pictures from database.
%winner: this array PNUMBERx10 array include 10 winners and distances from memory database for
%every test images.
% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%Ttype      : Test database picture type.
%HT         : This value show to program "how many times this simulation will run for different index database."
%reduceP    : Shows to reduce percentage for row and column.
%method     : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

function [index,winner]=EDtwoMRedSi(iType,pnumber,M1type,M2type,Ttype,HT,reduceP,method);

lengthpic=9430;

% chosing index number random or normal type
if iType=='n'
    index=1:pnumber;
else iType=='r'
    indexname = sprintf('C:\\MATLAB6p5\\work\\randomDATA\\indexfile-%d-%d.data',HT,pnumber);
    index=textread(indexname,'%f');
    index=round(index);
end

matmemory1=RMreducesize(pnumber, lengthpic, M1type, index,reduceP,method); % 100 pictures are reading and as
a length 9430 in memory database
matmemory2=RMreducesize(pnumber, lengthpic, M2type, index,reduceP,method); % 100 pictures are reading and as
a length 9430 in memory database
matmemory=[matmemory1;matmemory2];

winner=zeros(pnumber*2,10);

for perftestpic=1:pnumber

    % =====read test images =====

    % read files from the directory and load in mfilename pointer.
    tname=sprintf('C:\\images\\test\\image%d%s.raw', index(:,perftestpic), Ttype);
    fpmem=fopen(tname,'r');

    % check to error fp pointer
    if (fpmem<0)
        error('Files or a file could not read on this path')
        exit(0);
    end

    % reading to memory files to imgmem (1x9430) row matrix.
    [testing,countmem]=fread(fpmem,[1 inf]);
    %=====
```

```

%=====
if reduceP ~= 100;
    testing=resizeres(testing,reduceP,method);
else reduceP == 100;
    testing=testing;
end
%=====
% writing to whole row matrix into the test-files matrix.
fclose(fpmem);
%=====

perfdistance=0;

%measuring the distance test image and memory images
for imempic=1:pnumber*2

    perfabs=(testing-matmemory(imempic,:)).^2;
    perfabs=sqrt(perfabs);
    [sp,sf]=size(perfabs);
    %this line cancelled to average of distance
    %perfint=sum(perfabs)/sf;
    perfint=sum(perfabs);
    perfdistance(:, imempic)=perfint;

end
%finding first 10 winner
for i=1:10
    [mini indice]=min(perfdistance);
    win1(1,i)=indice;
    win2(1,i)=mini;
    perfdistance(:,indice)=9999999;
end

%writing first 10 winner in winner matrix
winner((perftestpic*2)-1,:)=win1(1,:);
winner((perftestpic*2),:)=win2(1,:);
perftestpic
end
clc
% writing to performance result excel files
filename1XLS=sprintf('EDtwoMRedSiperform-%s-%s-%s-%d-%d-
%d.xls',M1type,M2type,Ttype,pnumber,reduceP,HT);
fid=fopen(filename1XLS,'w');
fprintf(fid,'DATABASE INDEX NUMBER AND City Block DISTANCES \n\n');
if iType == 'n'
    fprintf(fid,'database index number selected normally')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2ft',winner(i,:));
    end
end
end

if iType == 'r'
    fprintf(fid,'database index number selected randomly')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2ft',winner(i,:));
    end
end
end

fclose(fid);

```

### A.3.2 Result of Main Function for Two Memory Images

```

%EDwoMRRedSi : This function use outputs of the EDtwoMRedSi function.
%Takes the winner matrix from EDoneMRedSi functions. Calculate
%Classification, Misclassification, Classification Time and Classification
%Percentage.
%[Tclas, TMisClas, time, TFiMemory, TSeMemory, PERcentage]=EDtwoMRRedSi(iType, pnumber, M1type, M2type, Ttype, HT, reduceP, method);

%OUTPUTS:
%Tclas : gives the classification number based on index number.
%TMisClas : this outputs gives misclassification index.
%time : gives classification time.
%TFiMemory : Total first memory winner results.
%TSeMemory : Total second memory winner results.
%PERcentage : calculate the classification performance percentage.

% INPUTS:
%iType : To define index type to read database images.
%pnumber : Numbers of picture are going to read from the memory database and test database.
%M1type : This is the first type memory picture.
%M2type : This is the second type memory picture.
%Ttype : Test database picture type.
%HT : This value show to program "how many times this simulation will run for different index database."
%reduceP : Shows to reduce percentage for row and column.
%method : Shows to interpretation method like ;'bilinear', 'bicubic', 'nearest'.

function
[Tclas, TMisClas, time, TFiMemory, TSeMemory, PERcentage]=EDtwoMRRedSi(iType, pnumber, M1type, M2type, Ttype, HT, reduceP, method);

t = clock;
[index, winner]=EDtwoMRedSi(iType, pnumber, M1type, M2type, Ttype, HT, reduceP, method);
time=etime(clock, t);

COMindex=1:pnumber;

winnerR=zeros(1, pnumber);
winnerR=winnerR';

for i=0:pnumber-1
    winnerR(i+1,:)=winner(2*i+1, 1);
end

% finding misclassification pictures and their index number
claspictures=0;
misclaspictures=0;
FiMemory=0;
SMemory=0;
Mismatrix=0;
for j=1:pnumber
    if winnerR(j)==COMindex(j)
        claspictures=claspictures+1;
        FiMemory(j,:)=1;
    elseif (winnerR(j)-pnumber)==COMindex(j)
        claspictures=claspictures+1;
        SMemory(j,:)=1;
    else winnerR(j)~=COMindex(j)
        misclaspictures=misclaspictures+1;
        Mismatrix(j,:)=1;
    end
end
% calculation of total matched picture number
Tclas=claspictures;

% calculation of total mismatched picture number

```

```

totalmismatched=sum(Mismatrix);
TMisClas=misclaspictures;
mismatchindex=find(Mismatrix);
mismatched=index(mismatchindex);

%calculation of second memory set
totalmatchFiMemory=sum(FiMemory);
TFiMemory=totalmatchFiMemory;
matchFiMemoryIndex=find(FiMemory);
matchedFiMemory=index(matchFiMemoryIndex);

%calculation of second memory set
totalmatchSeMemory=sum(SMemory);
TSeMemory=totalmatchSeMemory;
matchSeMemoryIndex=find(SMemory);
matchedSeMemory=index(matchSeMemoryIndex);

%calculation of Percentage of The Performance
PERcentage=100*(pnumber-totalmismatched)/pnumber;

% writing to result excel files
filenameXLS=sprintf('EDtwoMRRRedSi-%s-%s-%s-%d-%d-%d.xls',M1type,M2type,Ttype,pnumber,reduceP,HT);

fid=fopen(filenameXLS,'w');

fprintf(fid,'Matching Performance Time is %4.3f second \n',time);
fprintf(fid,'Matching Performance Percentage is %3.2f percent \n',PERcentage);
fprintf(fid,'Total match performance from First Memory (M1) is %3.2f \n',totalmatchFiMemory);
fprintf(fid,'Total match performance from Second Memory (M2) is %3.2f \n',totalmatchSeMemory);
%=====
%=====
fprintf(fid,'Misclassified Pictures Index\n');
fprintf(fid,'%3.2f\n',mismatchindex);

fprintf(fid,'Mismatched Pictures \n');
fprintf(fid,'%3.2f\n',mismatched);
%=====
%=====
fprintf(fid,'Classified Pictures Index Number From First Memory Set\n');
fprintf(fid,'%3.3f\n',matchFiMemoryIndex);

fprintf(fid,'Classified Pictures From First Memory Set\n');
fprintf(fid,'%3.3f\n',matchedFiMemory);
%=====
%=====
fprintf(fid,'Classified Pictures Index Number From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchSeMemoryIndex);

fprintf(fid,'Classified Pictures From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchedSeMemory);
%=====
%=====
%index for random pictures number
indres=1:pnumber;
indres(2,:)=index;
indres=indres';

if iType == 'r'
    fprintf(fid,'Database Picture Index \n');
    for i=1:pnumber
        fprintf(fid,'\n');
        fprintf(fid,'%3.3ft',indres(i,:));
    end
end

fclose(fid);

```



### A.3.3 Summary of Result Function For Two Memory Images

%EDtwoMTestSUMMRedSi : Run 5 times EDoneMRRedSi function and takes the outputs of EDoneMRRedSi and calculate Reduced Size Row and Column, Classified and Mis-Classified Pictures  
%Number, Time and Classification Percentage, and finally writes all results  
%in excel files.

% INPUTS:

%iType : To define index type to read database images.  
%pnumber : Numbers of picture are going to read from the memorydatabase and testdatabase.  
%M1type : This is the first type memory picture.  
%M2type : This is the second type memory picture.  
%Ttype : Test database picture type.  
%TR : This value show to program "how many times this simulation will run for different index database."  
%reduceP : Shows to reduce percentage for row and column.  
%method : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

```
function EDTwoMTestSUMMRedSi(iType,pnumber,M1type,M2type,Ttype,TR,reduceP,method);
```

```
for HT=1:TR
```

```
[Tclas, TMisClas, time, TFiMemory, TSeMemory, PERcentage]=EDTwoMRRedSi(iType,pnumber,M1type,M2type,Ttype,HT,reduceP,method);
```

```
    b(HT,:)=Tclas;  
    d(HT,:)=TMisClas;  
    de(HT,:)=TFiMemory;  
    f(HT,:)=TSeMemory;  
    Ttime(HT,:)=time;  
    PERcent(HT,:)=PERcentage;
```

```
end
```

```
Class=sum(b)/TR;  
misClass=sum(d)/TR;  
FirstMemoryClas=sum(de)/TR;  
SecMemoryClas=sum(f)/TR;  
TTime=sum(Ttime)/TR;  
PerCent=sum(PERcent)/TR;
```

```
PercentFirst=(FirstMemoryClas*100)/pnumber;  
PercentSecond=(SecMemoryClas*100)/pnumber;  
rrow=82;  
rcolumn=115;  
rproW=round((rrow*reduceP)/100);  
rpcolumn=round((rcolumn*reduceP)/100);
```

```
fileNameXLS=sprintf('EDtwoMRedSiSummaryRes-%s-%s-%s-%d-%s-%d.xls',M1type,M2type,Ttype,reduceP,method,pnumber);  
fid=fopen(fileNameXLS,'w');
```

```
fprintf(fid,'Summary of the %d Times Face Recognition Simulation \n\n',TR);
```

```
fprintf(fid,' Reduce Percentage of 82x115 is %3.2f \n',reduceP);  
fprintf(fid,' Reduce Pixel Values of Row Column %3.2f \n',rproW,rpcolumn);
```

```
fprintf(fid,' Average of The Classified Pictures Number is %3.2f \n',Class);  
fprintf(fid,' Average of The MIS-Classified Pictures Number is %3.2f \n',misClass);
```

```
fprintf(fid,' Average of The Classified Pictures Number from First Memory %3.2f \n',FirstMemoryClas);  
fprintf(fid,' Average of The Classified Pictures Number from First Memory as Percentage %3.2f \n',PercentFirst);
```

```
fprintf(fid,' Average of The Classified Pictures Number from Second Memory %3.2f \n',SecMemoryClas);  
fprintf(fid,' Average of The Classified Pictures Number from Second Memory as Percentage %3.2f \n',PercentSecond);
```

```
fprintf(fid,' Average Classification Time is %3.2f \n',TTime);
fprintf(fid,' Average Classification Percentage is %3.2f \n',PerCent);

fclose(fid);
```

## A.4 THREE MEMORY IMAGES SIMULATION FUNCTIONS (E.D.)

### A.4.1 Three Memory Main Function

```
%EDthreeMRedSi: This function reads three memory image and one test image and
%than compare them.
%[index,winner]=EDthreeMRedSi(iType,pnumber,M1type,M2type,M3type,Ttype,HT,reduceP,method);

%OUTPUTS:
%index :index number reading pictures from database.
%winner: this array PNUMBERx10 array include 10 winners and distances from memory database for
%every test images.
% INPUTS:

%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%M3type     : This is the third type memory picture.
%Ttype     : Test database picture type.
%HT         : This value show to program "how many times this simulation will run for different index database."
%reduceP   : Shows to reduce percentage for row and column.
%method     : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

function [index,winner]=EDthreeMRedSi(iType,pnumber,M1type,M2type,M3type,Ttype,HT,reduceP,method);

lengthpic=9430;

% chosing index number random or normal type
if iType=='n'
    index=1:pnumber;
else iType=='r'
    indexname = sprintf('C:\\MATLAB6p5\\work\\randomDATA\\indexfile-%d-%d.data',HT,pnumber);
    index=textread(indexname,'%f');
    index=index';
end

matmemory1=RMreducesize(pnumber, lengthpic, M1type, index,reduceP,method);
matmemory2=RMreducesize(pnumber, lengthpic, M2type, index,reduceP,method);
matmemory3=RMreducesize(pnumber, lengthpic, M3type, index,reduceP,method);
matmemory=[matmemory1;matmemory2;matmemory3];

winner=zeros(pnumber*2,10);

for perftestpic=1:pnumber

    % read test images *****

    % read files from the directory and load in mfilename pointer.
    tname=sprintf('C:\\images\\test\\image%d\\%s.raw', index(:,perftestpic), Ttype);
    fpmem=fopen(tname,'r');

    % check to error fp pointer
    if (fpmem<0)
        error('Files or a file could not read on this path')
        exit(0);
    end

    % reading to memory files to imgmem (1x9430) row matrix.
    [testing,countmem]=fread(fpmem,[1 inf]);
    %
```

```

%=====
if reduceP ~= 100;
    testing=resizers(testing,reduceP,method);
else reduceP == 100;
    testing=testing;
end
%=====
%=====
% writing to whole row matrix into the test-files matrix.

fclose(fpmem);
%=====

perfdistance=0;

%measuring the distance test image and memory images
for imempic=1:pnumber*3

    perfabs=(testing-matmemory(imempic,:)).^2;
    perfabs=sqrt(perfabs);
    [sp,sf]=size(perfabs);
    %this line cancelled to average of distance
    %perfint=sum(perfabs)/sf;
    perfint=sum(perfabs);
    perfdistance(:, imempic)=perfint;

end
%finding first 10 winner
for i=1:10
    [mini indice]=min(perfdistance);
    win1(1,i)=indice;
    win2(1,i)=mini;
    perfdistance(:,indice)=9999999;
end

%writing first 10 winner in winner matrix
winner((perftestpic*2)-1,:)=win1(1,:);
winner((perftestpic*2),:)=win2(1,:);
perftestpic
end
clc
% writing to performance result excel files
filename1XLS=sprintf('EDthreeMRedSiperform-%s-%s-%s-%s-%d-%d-
%d.xls',M1type,M2type,M3type,Ttype,pnumber,reduceP,HT);
fid=fopen(filename1XLS,'w');
fprintf(fid,'DATABASE INDEX NUMBER AND City Block DISTANCES \n\n');
if iType == 'n'
    fprintf(fid,'database index number selected normally')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2ft',winner(i,:));
    end
end
end

if iType == 'r'
    fprintf(fid,'database index number selected randomly')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2ft',winner(i,:));
    end
end
end

fclose(fid);

```

#### A.4.2 Result of Main Function For Three Memory Images

%EDthreeMRRedSi : This function use outputs of the EDthreeMRRedSi function.

```

%Takes the winner matrix from EDthreeMRRedSifunctions. Calculate
%Classification, Misclassification, Classification Time, First, Second, Third Memory Image Classification Results and
Total Classification Percentage.
%[claspictures,misclaspictures,time,FiMemory,SeMemory,ThMemory,PERcentage]=EDthreeMRRedSi(iType,pnumber,
M1type,M2type,M3type,Ttype,HT,reduceP,method);

%OUTPUTS:
%claspictures : Classified picture number.
%misclaspictures : misclassified pictures number.
%time : Classification Time
%FiMemory : Classified Pictures Result from First Memory Images.
%SeMemory : Classified Pictures Result from Second Memory Images.
%ThMemory : Classified Pictures Result from Third Memory Images.
%PERcentage : Total Classification Performance Percentage.

% INPUTS:
%iType : To define index type to read database images.
%pnumber : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type : This is the first type memory picture.
%M2type : This is the second type memory picture.
%M3type : This is the third type memory picture.
%Ttype : Test database picture type.
%HT : This value show to program "how many times this simulation will run for different index database."
%reduceP : Shows to reduce percentage for row and column.
%method : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

function
[claspictures,misclaspictures,time,FiMemory,SeMemory,ThMemory,PERcentage]=EDthreeMRRedSi(iType,pnumber,
M1type,M2type,M3type,Ttype,HT,reduceP,method);

t = clock;
[index,winner]=EDthreeMRRedSi(iType,pnumber,M1type,M2type,M3type,Ttype,HT,reduceP,method);
time=etime(clock,t);

COMindex=1:pnumber;

winnerR=zeros(1,pnumber);
winnerR=winnerR';

for i=0:pnumber-1
    winnerR(i+1,:)=winner(2*i+1,1);
end

% finding misclassification pictures and their index number
claspictures=0;
misclaspictures=0;
Fimem=0;
Smem=0;
Tmem=0;
Mismatrix=0;
for j=1:pnumber
    if winnerR(j)==COMindex(j)
        claspictures=claspictures+1;
        Fimem(j,:)=1;
    elseif (winnerR(j)-pnumber)==COMindex(j)
        claspictures=claspictures+1;
        Smem(j,:)=1;
    elseif (winnerR(j)-2*pnumber)==COMindex(j)
        claspictures=claspictures+1;
        Tmem(j,:)=1;
    else winnerR(j)~=COMindex(j)
        misclaspictures=misclaspictures+1;
        Mismatrix(j,:)=1;
    end
end
end
% calculation of total matched picture

```

```

totalmatched=claspictures;

% calculation of total mismatched picture number
totalmismatched=misclaspictures;
mismatchindex=find(Mismatrix);
mismatched=index(mismatchindex);

%Calculation of First Memory
totalmatchFiMemory=sum(Fimem);
FiMemory=totalmatchFiMemory;
matchFiMemoryIndex=find(Fimem);
matchedFiMemory=index(matchFiMemoryIndex);
percentFimem=(totalmatchFiMemory*100)/pnumber;

%calculation of second memory set
totalmatchSeMemory=sum(Smem);
SeMemory=totalmatchSeMemory;
matchSeMemoryIndex=find(Smem);
matchedSeMemory=index(matchSeMemoryIndex);
percentSemem=(totalmatchSeMemory*100)/pnumber;

%calculation of third memory set
totalmatchThMemory=sum(Tmem);
ThMemory=totalmatchThMemory;
matchThMemoryIndex=find(Tmem);
matchedThMemory=index(matchThMemoryIndex);
percentThmem=(totalmatchThMemory*100)/pnumber;

%calculation of Percentage of The Performance

PERcentage=100*(pnumber-totalmismatched)/pnumber;

% writing to result excel files
filenameXLS=sprintf('EDThreeMRRedSi-%s-%s-%s-%s-%d-%d-
%d.xls',M1type,M2type,M3type,Ttype,pnumber,reduceP,HT);

fid=fopen(filenameXLS,'w');

fprintf(fid,'Total Classified Pictures is %3.2f \n',totalmatched);
fprintf(fid,'Total Mis-Classified Pictures is %3.2f \n',totalmismatched);
fprintf(fid,'Matching Performance Time is %4.3f second \n',time);
fprintf(fid,'Matching Performance Percentage is %3.2f percent \n',PERcentage);
fprintf(fid,'Total First Memory Classified Pictures is %3.2f \n',totalmatchFiMemory);
fprintf(fid,'Total Second Memory Classified Pictures is %3.2f \n',totalmatchSeMemory);
fprintf(fid,'Total Third Memory Classified Pictures is %3.2f \n',totalmatchThMemory);

%=====
fprintf(fid,'Classified Pictures Indexing Number From First Memory Set\n');
fprintf(fid,'%3.3f\n',matchFiMemoryIndex);

fprintf(fid,'Classified Pictures From First Memory Set as Percentage \n');
fprintf(fid,'%3.3f\n',matchedFiMemory);
%=====
fprintf(fid,'Classified Pictures Indexing Number From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchSeMemoryIndex);

fprintf(fid,'Classified Pictures From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchedSeMemory);
%=====
fprintf(fid,'Classified Pictures Indexing Number From Third Memory Set\n');
fprintf(fid,'%3.3f\n',matchThMemoryIndex);

fprintf(fid,'Classified Pictures From Third Memory Set\n');
fprintf(fid,'%3.3f\n',matchedThMemory);
%=====

fprintf(fid,'Misclassified Pictures Indexing number\n');

```

```

fprintf(fid,'%3.2f\n',mismatchindex);

fprintf(fid,'Mismatched Pictures \n');
fprintf(fid,'%3.2f\n',mismatched);

%index for random pictures number
indres=1:pnumber;
indres(2,:)=index;
indres=indres';

if iType == 'r'
fprintf(fid,'Database Picture Index \n');
for i=1:pnumber
    fprintf(fid,'\n');
    fprintf(fid,'%3.3f\t',indres(i,:));
end
end

fclose(fid);

```

### A.4.3 Summary of Result Function for Three Memory Images

%EDthreeMTestSUMMRRedSi : Run 5 times EDoneMRRedSi function and takes outputs of the EDthreeMRRedSi function calculate Classification, Misclassification, Classification Time, First, Second, Third Memory Image Classification Results and Total Classification Percentage and finally writes all results in excel files.

```

% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%M3type     : This is the third type memory picture.
%Ttype      : Test database picture type.
%TR         : This value show to program "how many times this simulation will run for different index database."
%reduceP    : Shows to reduce percentage for row and column.
%method     : Shows to interpretation method like ;'bilinear','bicubic','nearest'.

function EDthreeMTestSUMMRRedSi(iType,pnumber,M1type,M2type,M3type,Ttype,TR,reduceP,method);

for HT=1:TR

[claspictures,misclaspictures,time,FiMemory,SeMemory,ThMemory,PERcentage]=EDthreeMRRedSi(iType,pnumber,
M1type,M2type,M3type,Ttype,HT,reduceP,method);
    b(HT,:)=claspictures;
    d(HT,:)=misclaspictures;
    de(HT,:)=FiMemory;
    f(HT,:)=SeMemory;
    g(HT,:)=ThMemory;
    Ttime(HT,:)=time;
    PERcent(HT,:)=PERcentage;
end
Class=sum(b)/TR;
misClass=sum(d)/TR;

FirstMemoryClas=sum(de)/TR;
PercentFirst=(FirstMemoryClas*100)/pnumber;
SecMemoryClas=sum(f)/TR;
PercentSecond=(SecMemoryClas*100)/pnumber;
ThirdMemoryClass=sum(g)/TR;
PercentThird=(ThirdMemoryClass*100)/pnumber;

TTime=sum(Ttime)/TR;
PerCent=sum(PERcent)/TR;

rrow=82;
rcolumn=115;

```

```

rproW=round((rrow*reduceP)/100);
rpcolumn=round((rcolumn*reduceP)/100);

fileNameXLS=sprintf('EDthreeMRedSiSummaryRes-%s-%s-%s-%s-%d-%s-
%d.xls',M1type,M2type,M3type,Ttype,reduceP,method,pnumber);
fid=fopen(fileNameXLS,'w');

fprintf(fid,'Summary of the %d Times Face Recognition Simulation \n\n',TR);

fprintf(fid,' Reduce Percentage of 82x115 is %3.2f \n',reduceP);
fprintf(fid,' Reduce Pixel Values of Row Column %3.2f \n',rproW,rpcolumn);

fprintf(fid,' Average of The Classified Pictures Number is %3.2f \n',Class);
fprintf(fid,' Average of The MIS-Classified Pictures Number is %3.2f \n',misClass);

fprintf(fid,' Average of The Classified Pictures Number from First Memory %3.2f \n',FirstMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from First Memory as Percentage %3.2f \n',PercentFirst);

fprintf(fid,' Average of The Classified Pictures Number from Second Memory %3.2f \n',SecMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from Second Memory as Percentage %3.2f
\n',PercentSecond);

fprintf(fid,' Average of The Classified Pictures Number from Third Memory %3.2f \n',ThirdMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from Third Memory as Percentage %3.2f \n',PercentThird);

fprintf(fid,' Average Classification Time is %3.2f \n',TTime);
fprintf(fid,' Average Classification Percentage is %3.2f \n',PerCent);

fclose(fid);

```

## A.5 FOUR MEMORY IMAGES SIMULATION FUNCTIONS (E.D.)

### A.5.1 Four Memory Main Function

```

%EDfourMRedSi: This function reads four memory image and one test image and
%than compare them.
%[index, winner]=EDfourMRedSi(iType,pnumber,M1type,M2type,M3type,Ttype,HT,reduceP,method);

%OUTPUTS:
%index :index number reading pictures from database.
%winner: this array PNUMBERx10 array include 10 winners and distances from memory database for
%every test images.

% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%M3type     : This is the third type memory picture.
%M4type     : This is the fourth type memory picture.
%Ttype     : Test database picture type.
%HT        : This value show to program "how many times this simulation will run for different index database."
%reduceP   : Shows to reduce percentage for row and column.
%method    : Shows to interpretation method like ;'bilinear','bicubic','nearest'.

function
[index, winner]=EDfourMRedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,HT,reduceP,method);

lengthpic=9430;

% chosing index number random or normal type
if iType=='n'
    index=1:pnumber;

```

```

else iType=='r'
    indexname = sprintf('C:\MATLAB6p5\work\randomDATA\indexfile-%d-%d.data',HT,pnumber);
    index=txtread(indexname,'%f');
    index=index';
end

matmemory1=RMreducesize(pnumber, lengthpic, M1type, index,reduceP,method);
matmemory2=RMreducesize(pnumber, lengthpic, M2type, index,reduceP,method);
matmemory3=RMreducesize(pnumber, lengthpic, M3type, index,reduceP,method);
matmemory4=RMreducesize(pnumber, lengthpic, M4type, index,reduceP,method);
matmemory=[matmemory1;matmemory2;matmemory3;matmemory4];

winner=zeros(pnumber*2,10);

for perftestpic=1:pnumber

    % read test images *****

    % read files from the directory and load in mfilename pointer.
    tname=sprintf('C:\images\test\image%d%s.raw', index(:,perftestpic), Ttype);
    fpmem=fopen(tname,'r');

    % check to error fp pointer
    if (fpmem<0)
        error('Files or a file could not read on this path')
        exit(0);
    end

    % reading to memory files to imgmem (1x9430) row matrix.
    [testing,countmem]=fread(fpmem,[1 inf]);
    %=====
    %=====
    if reduceP ~= 100;
        testing=resizeres(testing,reduceP,method);
    else reduceP == 100;
        testing=testing;
    end
    %=====
    %=====
    % writing to whole row matrix into the test-files matrix.

    fclose(fpmem);
    %*****

    perfdistance=0;

    %measuring the distance test image and memory images
    for imempic=1:pnumber*4

        perfabs=(testing-matmemory(imempic,:)).^2;
        perfabs=sqrt(perfabs);
        [sp,sf]=size(perfabs);
        %this line cancelled to average of distance
        %perfint=sum(perfabs)/sf;
        perfint=sum(perfabs);
        perfdistance(:, imempic)=perfint;

    end

    %finding first 10 winner
    for i=1:10
        [mini indice]=min(perfdistance);
        win1(1,i)=indice;
        win2(1,i)=mini;
        perfdistance(:,indice)=9999999;
    end

    %writing first 10 winner in winner matrix

```



```

winner((perftestpic*2)-1,:)=win1(1,:);
winner((perftestpic*2),:)=win2(1,:);
perftestpic
end
clc
% writing to performance result excel files
filename1XLS=sprintf('EDfourMRRedSiperform-%s-%s-%s-%s-%s-%d-%d-
%d.xls',M1type,M2type,M3type,M4type,Ttype,pnumber,reduceP,HT);
fid=fopen(filename1XLS,'w');
fprintf(fid,'DATABASE INDEX NUMBER AND City Block DISTANCES \n\n');
if iType == 'n'
    fprintf(fid,'database index number selected normally')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end
end

if iType == 'r'
    fprintf(fid,'database index number selected randomly')
    for i=1:pnumber*2
        fprintf(fid,'\n');
        fprintf(fid,'%3.2f\t',winner(i,:));
    end
end
end

fclose(fid);

```

## A.5.2 Result of Main Function For Four Memory Images

```

%EDfourMRRedSi : This function use outputs of the EDfourMRRedSi function.
%Takes the winner matrix from EDfourMRRedSi functions. Calculate
%Classification, Misclassification, Classification Time, First, Second, Third, Fourth Memory Image Classification
Results and Total Classification
%Percentage.

%[claspictures,misclaspictures,time,TFiMemory,TSeMemory,TThMemory,TFouMemory,PERcentage]=EDfourMR
RedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,HT,reduceP,method);

% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%M3type     : This is the third type memory picture.
%M4type     : This is the fourth type memory picture.
%Ttype     : Test database picture type.
%HT        : This value show to program "how many times this simulation will run for different index database."
%reduceP   : Shows to reduce percentage for row and column.
%method    : Shows to interpratation method like ;'bilinear','bicubic','nearest'.

function
[claspictures,misclaspictures,time,TFiMemory,TSeMemory,TThMemory,TFouMemory,PERcentage]=EDfourMRRe
dSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,HT,reduceP,method);

t = clock;
[index,winner]=EDfourMRRedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,HT,reduceP,method);
time=etime(clock,t);

COMindex=1:pnumber;
winnerR=zeros(1,pnumber);
winnerR=winnerR';

for i=0:pnumber-1

```

```

winnerR(i+1,:)=winner(2*i+1,1);
end

% finding misclassification pictures and their index number
claspictures=0;
misclaspictures=0;
Fimem=0;
Smem=0;
Tmem=0;
Foumem=0;
Mismatrix=0;
for j=1:pnumber
    if winnerR(j)==COMindex(j)
        claspictures=claspictures+1;
        Fimem(j,:)=1;
    elseif (winnerR(j)-pnumber)==COMindex(j)
        claspictures=claspictures+1;
        Smem(j,:)=1;
    elseif (winnerR(j)-2*pnumber)==COMindex(j)
        claspictures=claspictures+1;
        Tmem(j,:)=1;
    elseif (winnerR(j)-3*pnumber)==COMindex(j)
        claspictures=claspictures+1;
        Foumem(j,:)=1;
    else winnerR(j)~=COMindex(j)
        misclaspictures=misclaspictures+1;
        Mismatrix(j,:)=1;
    end
end
end
% calculation of total mismatched picture number GENERAL RESULTS
totalmismatched=sum(Mismatrix);
mismatchindex=find(Mismatrix);
mismatched=index(mismatchindex);

%calculation of first memory set
TFiMemory=sum(Fimem);
matchFiMemoryIndex=find(Fimem);
matchedFiMemory=index(matchFiMemoryIndex);
percentFimem=(TFiMemory*100)/pnumber;

%calculation of second memory set
TSeMemory=sum(Smem);
matchSeMemoryIndex=find(Smem);
matchedSeMemory=index(matchSeMemoryIndex);
percentSemem=(TSeMemory*100)/pnumber;

%calculation of third memory set
TThMemory=sum(Tmem);
matchThMemoryIndex=find(Tmem);
matchedThMemory=index(matchThMemoryIndex);
percentThmem=(TThMemory*100)/pnumber;

%calculation of fourth memory set
TFouMemory=sum(Foumem);
matchFouMemoryIndex=find(Foumem);
matchedFouMemory=index(matchFouMemoryIndex);
percentFoumem=(TFouMemory*100)/pnumber;

%calculation of Percentage of The Performance
PERcentage=100*(pnumber-totalmismatched)/pnumber;

% writing to result excel files
filenameXLS=sprintf('EDfourMRRedSi-%s-%s-%s-%s-%s-%d-%d-
%d.xls',M1type,M2type,M3type,M4type,Ttype,pnumber,reduceP,HT);

fid=fopen(filenameXLS,'w');

```

```

fprintf(fid,'Matching Performance Time is %4.3f second \n',time);
fprintf(fid,'Matching Performance Percentage is %3.2f percent \n',PERcentage);

fprintf(fid,'Total First Memory Classified Pictures is %3.2f \n',TFiMemory);
fprintf(fid,'Total Second Memory Classified Pictures is %3.2f \n',TSeMemory);
fprintf(fid,'Total Third Memory Classified Pictures is %3.2f \n',TThMemory);
fprintf(fid,'Total Fourth Memory Classified Pictures is %3.2f \n',TFouMemory);

%=====
%*****
fprintf(fid,'Classified Pictures From First Memory Set as Percentage \n');
fprintf(fid,'%3.3f\n',percentFimem);
%=====
fprintf(fid,'Classified Pictures From Second Memory Set as Percentage \n');
fprintf(fid,'%3.3f\n',percentSemem);
%=====
fprintf(fid,'Classified Pictures From Third Memory Set as Percentage \n');
fprintf(fid,'%3.3f\n',percentThmem);
%=====
fprintf(fid,'Classified Pictures From Fourth Memory Set as Percentage \n');
fprintf(fid,'%3.3f\n',percentFoumem);
%*****
%=====
fprintf(fid,'Misclassified Pictures Index\n');
fprintf(fid,'%3.2f\n',mismatchindex);

fprintf(fid,'Mismatched Pictures \n');
fprintf(fid,'%3.2f\n',mismatched);

%=====
fprintf(fid,'Classified Pictures Index Number From First Memory Set\n');
fprintf(fid,'%3.3f\n',matchFiMemoryIndex);

fprintf(fid,'Classified Pictures From First Memory Set\n');
fprintf(fid,'%3.3f\n',matchedFiMemory);
%=====
fprintf(fid,'Classified Pictures Index Number From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchSeMemoryIndex);

fprintf(fid,'Classified Pictures From Second Memory Set\n');
fprintf(fid,'%3.3f\n',matchedSeMemory);
%=====
fprintf(fid,'Classified Pictures Index Number From Third Memory Set\n');
fprintf(fid,'%3.3f\n',matchThMemoryIndex);

fprintf(fid,'Classified Pictures From Third Memory Set\n');
fprintf(fid,'%3.3f\n',matchedThMemory);
%=====
%=====
fprintf(fid,'Classified Pictures Index Number From Fourth Memory Set\n');
fprintf(fid,'%3.3f\n',matchFouMemoryIndex);

fprintf(fid,'Classified Pictures From Fourth Memory Set\n');
fprintf(fid,'%3.3f\n',matchedFouMemory);
%=====

%index for random pictures number
indres=1:pnumber;
indres(2,:)=index;
indres=indres';

if iType == 'r'
fprintf(fid,'Database Picture Index \n');
for i=1:pnumber
fprintf(fid,'\n');

```

```

    fprintf(fid,'%3.3f\t',indres(i,:));
end
end

fclose(fid);

```

### A.5.3 Summary of Result Function For Four Memory Images

```

%EDthreeMTestSUMMRedSi : Run 5 times EDoneMRRedSi function and takes outputs of the EDfourMRRedSi
function calculate Classification, Misclassification, Classification Time, First,Second,Third,Fourth Memory Image
Classification Results and Total Classification Percentage and finally writes all results in excel files.
%EDfourMTestSUMMRedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,TR,reduceP,method);

% INPUTS:
%iType      : To define index type to read database images.
%pnumber    : Numbers of picture are going to read from the memorydatabase and testdatabase.
%M1type     : This is the first type memory picture.
%M2type     : This is the second type memory picture.
%M3type     : This is the third type memory picture.
%M4type     : This is the fourth type memory picture.
%Ttype      : Test database picture type.
%HT         : This value show to program "how many times this simulation will run for different index database."
%reduceP    : Shows to reduce percentage for row and column.
%method     : Shows to interpretation method like 'bilinear','bicubic','nearest'.
function EDfourMTestSUMMRedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,TR,reduceP,method);

for HT=1:TR
[claspictures,misclaspictures,time,TFiMemory,TSeMemory,TThMemory,TFouMemory,PERcentage]=EDfourMRRedSi(iType,pnumber,M1type,M2type,M3type,M4type,Ttype,HT,reduceP,method);
    b(HT,:)=claspictures;
    d(HT,:)=misclaspictures;
    de(HT,:)=TFiMemory;
    f(HT,:)=TSeMemory;
    g(HT,:)=TThMemory;
    h(HT,:)=TFouMemory;
    Ttime(HT,:)=time;
    PERcent(HT,:)=PERcentage;
end
Class=sum(b)/TR;
misClass=sum(d)/TR;

FirstMemoryClas=sum(de)/TR;
PercentFirst=(FirstMemoryClas*100)/pnumber;

SecMemoryClas=sum(f)/TR;
PercentSecond=(SecMemoryClas*100)/pnumber;

ThirdMemoryClass=sum(g)/TR;
PercentThird=(ThirdMemoryClass*100)/pnumber;

FourthMemoryClass=sum(h)/TR;
PercentFourth=(FourthMemoryClass*100)/pnumber;

TTime=sum(Ttime)/TR;

PerCent=sum(PERcent)/TR;

fileNameXLS=sprintf('EDFourthMSummaryResRedSi-%s-%s-%s-%s-%s-%d-%s-%d.xls',M1type,M2type,M3type,M4type,Ttype,reduceP,method,pnumber);
fid=fopen(fileNameXLS,'w');

fprintf(fid,'Summary of the %d Times Face Recognition Simulation \n\n',TR);

fprintf(fid,' Average of The Classified Pictures Number is %3.2f \n',Class);
fprintf(fid,' Average of The MIS-Classified Pictures Number is %3.2f \n',misClass);

```

```
fprintf(fid,' Average of The Classified Pictures Number from First Memory %3.2f \n',FirstMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from First Memory as Percentage %3.2f \n',PercentFirst);

fprintf(fid,' Average of The Classified Pictures Number from Second Memory %3.2f \n',SecMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from Second Memory as Percentage %3.2f
\n',PercentSecond);

fprintf(fid,' Average of The Classified Pictures Number from Third Memory %3.2f \n',ThirdMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from Third Memory as Percentage %3.2f \n',PercentThird);

fprintf(fid,' Average of The Classified Pictures Number from Fourth Memory %3.2f \n',FourthMemoryClass);
fprintf(fid,' Average of The Classified Pictures Number from Fourth Memory as Percentage %3.2f
\n',PercentFourth);

fprintf(fid,' Average Classification Time is %3.2f \n',TTime);
fprintf(fid,' Average Classification Percentage is %3.2f \n',PerCent);

fclose(fid);
```



## REFERENCES

- Anil K. Jain, Fellow, IEEE, Robert P.W. Duin, and Jianchang Mao, "Statistical Pattern Recognition: A Review", *IEEE Transactions On Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, January 2000
- Artiklar, M., Mu, X., Hassoun, M., and Watta, P., "Local Voting Networks for Human Face Recognition", *International Joint Conference on Neural Networks (IJCNN'03)*, Portland, Oregon, July 20-24, 2003
- Artiklar, M., "Capacity Analysis of Voting Networks with Application to Human Face Recognition", Ph.D. Thesis, Wayne State University, Detroit, MI, 2002
- Artiklar, M., Hassoun, M., and Watta, P., "Application of a Post-Processing Algorithm for Improved Human Face Recognition", *International Joint Conference on Neural Networks (IJCNN99)*, CD ROM Proceedings paper number JCNN2166, Washington D.C., July 1999
- Belhumeur, P., Hespanha, J., and Kriegman, D., "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No 7, July 1997.
- Duda Richard O. Peter E. Hart, David G. Stork "Pattern Classification" ISBN: 0471056693 Wiley-Interscience; 2nd edition , October 2000.
- John Daugman, Associate Editor, PAMI, "Face Gesture Recognition: Overview" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.19, No 7, July 1997.
- Lades, M., Vorbruggen, J., C., Buhmann, J., Lange, J., Malsburg, C.v.d., Wurtz, R., P., Konen, W., "Distortion Invariant Object Recognition in Dynamic Link Architecture", *IEEE Trans. On Computation*, 42, 1993, pp 300-311.
- Maxim A. Grudin, "On Internal Representations in Face Recognition Systems", *Pergamon, Pattern Recognition*, 33, 1161-1177, 2000
- Phillips, P., J., Wechsler, H., Huang, J., Rauss, P., J., "The FERET Database and Evaluation Procedure for Face Recognition Algorithms", *Image and Vision Computing*, 16 (1998), 295-306

- R. Brunelli, T. Poggio, "Face Recognition Through Geometrical Features", In *Proceedings of ECCV '92*, pages 792-800. S. Margherita Ligure, 1992.
- R. Brunelli, T. Poggio, "Face Recognition: Features Versus Templates", *IEEE Trans. Pattern. Anal. Mach. Intell* 15 1042-1052, 1993.
- R. Brunelli, T. Poggio, "Template Matching: Matched Spatial Filters and Beyond", *IRST Tech. Rep. 9510-03. Pattern Recognition* Vol. 30, No. 5, pp. 751-768, 1997
- R. C. Gonzales and R. E. Woods, "Digital Image Processing", *Second Edition*, Prentice Hall, New Jersey, Book 2002.
- Robert Schalkoff, "Pattern Recognition Statistical, Structural and Neural Approaches" Book 1992.
- Samaria, F. S., Harter, A. C., "Parameterization of a Stochastic Model for Human Face Identification", *Proceedings of the 2nd IEEE workshop on Applications of Computer Vision*, Sarasota, Florida, 1994.
- Sirovich, L., Kirby, M., "Low Dimensional Procedure for Characterization of Human Faces", *J. Opt. Cos. Amer.*, Vol 4, pp 519-524, 1987.
- T. Sim, R. Sukthankar, M. Mullin, and S. Baluja. "Memory-based face recognition for visitor identification.", *Proceedings of 4th International Conf. on Face and Gesture Recognition*, pages 214--220, Grenoble, France, March 2000.
- Terence Sim, Rahul Sukthankar, Matthew Mullin and Shumeet Baluja, "Memory-Based Face Recognition for Visitor Identification". *Proceedings of IEEE Face and Gesture*, 2000.
- Turk, M., Pentland, A., "Eigenfaces for Recognition", *J. Cognitive Neuroscience*, 3(1), 71-86, 1991.
- Wiskott, L., Fellous, J. M., Kruger, N., and Von Der Malsburg, C., "Face Recognition by Elastic Bunch Graph Matching", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 775-779, 1997.