THESIS APPROVAL

The abstract and thesis of Dimitriy A. Labunsky for the Master of Science in Electrical and Computer Engineering were presented November 6, 2009, and accepted by the thesis committee and the department.

COMMITTEE APPROVALS:

Marek Perkowski (Adviser)

Doug Hall

Christof Teuscher

ABSTRACT

An abstract of the thesis of Dimitriy A. Labunsky for the Masters of Science in Electrical and Computer Engineering presented November 06, 2009.

Title: Facial Emotion Recognition System Based on Principle Component Analysis and Neural Networks.

In order for autonomous systems to learn through interaction, they must have the ability to collect data about their surroundings. Computer vision research has resulted in many viable approaches being presented for face detection, face recognition and face tracking. In contrast, few approaches have been presented for emotion recognition. Until computer systems have the ability to recognize emotion, realistic human-computer interaction cannot take place.

A system for human emotion recognition, through the analysis of facial expressions is presented. The system consists of two phases. The first phase preprocesses the image by implementing principle component analysis. This results in a compressed representation of image data. The output of the first phase is used as the input into the second phase. The second phase consists of an artificial neural

network which is trained to identify emotions based on facial expressions. The output of the emotion recognition system consists of seven probabilities corresponding to each of the seven select emotions: angry, disgusted, fearful, happy, neutral, sad and surprised.

The presented system performs well in a controlled environment. It has the ability to be implemented in real-time and requires very few computational resources to function. This system can be easily combined with the eigenface face recognition system, resulting in a multimodal system which is able of both identifying an individual and recognizing their emotion.

FACIAL EMOTION RECOGNITION SYSTEM BASED ON PRINCIPLE COMPONENT ANALYSIS AND NEURAL NETWORKS

by

DIMITRIY A. LABUNSKY

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL AND COMPUTER ENGINEERING

Portland State University
2009

Acknowledgments

I would like thank to my advisor Marek Perkowski for all of his help and for providing me with the motivation and resources necessary for building D.I.M, the humanoid robot. This has been the largest and most complex single project I have ever undertaken. It would not have been possible without his help.

I would also like to thank my family for supporting and motivating me and providing me with an ideal environment for carrying out my studies. Because of my family, I was able to dedicate my whole self to my studies without unnecessary distractions. Special thank goes to my dad Aleksey, my mom Anna, my sister Lubov and my brother Sergey.

Finally I would like to thank Larry Caminiti for providing me with feedback on earlier versions of the thesis. He identified areas which were not particularly clearly written. He also provided many useful suggestions which were implemented in later versions of the thesis. Along with him, I'd like to express my thanks to Veronica Caminiti for her persistent encouragement and motivation.

Table of Contents

List of Tables

List of Figures

Glossary

ANN – *Artificial Neural Network.*
A computational model that mimics the behavior of a biological network of neurons and has the ability to learn and use what it learned to generalize.

DCT – *Discrete Cosine Transform.*

Mathematical approach to expressing a finite set of data points as a sum of cosine waves at different frequencies.

ERS - *Emotion Recognition System.*
Emotion Recognition System developed to improve both HCI and in particularly HRI. This system is based on PCA and ANN.

FPGA – *Field Programmable Gate Array.*
A semiconductor device that is composed of large amount of logic cells which can be programmed by either the manufacturer or the customer. These blocks are reprogrammable.

GUI – *Graphical User Interface.*
This is an interface for users which is based on visual elements such as buttons, menus and various other visual elements that the user can use to control or view a program.

HCI – *Human Computer Interaction.*
The study of interaction between humans and computers.

HRI – *Human Robot Interaction.*
The study of interaction between humans and robots. This is a subset of HCI.

MSE – *Mean Square Error.*
A mathematical measure of what the difference is between a desired value and an actual value.

PCA – *Principle Component Analysis.*
This is a mathematical approach to simplify complex multivariable problems by reducing dimensionality.

Chapter 1 - Introduction

Section 1.1 – Problem Description

Today there is a new trend in robot technology. Apart from the common applications of robots in the industry and military, humanoid robots are beginning to attracting attention. Humanoid robotics has primarily been a tool for advanced studies in such research fields as psychology, mechanical engineering, electrical engineering and computer science. For the first time, humanoid robots are being considered for applications where they would interact directly with humans. While some humanoid robot applications are already being validated in practice, others are still on the drawing board. The market for humanoid robots designed for entertainment, has caught on successfully. Currently Japan has taken the lead in humanoid robotics research and has applied them in real-world applications, unlike any other country. The most common application for humanoid robots, apart from entertainment, is applications such as receptionists and

Figure 1.1 – Humanoid Robot.

tour guides, or helpers at such locations as large international airports. With technology being so commonly found in our everyday lives, the overall social attitude towards technology is improving. This together with ever increasing advancements in technology is allowing for humanoid robots to find applications in the real-word.

In order for humanoid robots to effectively serve humans and interact with humans of various technical backgrounds, they must posses as many humanlike qualities as possible. It shouldn’t be difficult for a non-technical person to interact with a humanoid robot. If humanoid robots are used for such applications as keeping the elderly company, or helping them to remember to take their medications on time, it must not be expected that interaction with robots will be difficult task. It cannot be expected that the elderly people will devote a great deal of their time to studying how to interact with the robots. Humanoid robots need to be designed in such a way that they can interact with humans at a level that would seem natural to humans. Given that emotion plays such an important factor in human communication, the humanoid robots must be designed with the ability to both interpret and express emotions. The ability to interpret and express emotions becomes an especially important factor when the humanoid robot is designed to interact with humans on a regular basis.

Therefore the requirement that humanoid robots be able to interact with humans at a realistic level by both understanding and expressing emotions, introduces a demand for a computational system that will enable the robot the ability to be emotional. Such a system will allow humanoid robots to be far more suitable for applications where interaction with humans is required.

Section 1.2 – Area of Focus

One of the more practical sensory inputs that the robot can obtain in order to interact realistically with a human, is information about the person's emotional state. Currently there are methods for such image processing tasks as object detection, object recognition and tracking. Most of these areas have been heavily researched and viable solutions have been presented [3], [22] and [27]. However, far less research has been conducted in emotion recognition. If a humanoid robot obtains the ability to communicate with knowledge about a person's emotion, far more realistic and meaningful interaction can take place. It is therefore the aim of this research to develop an emotion recognition system. There are a number of approaches that can be taken to achieve this. These approaches will be further explored in the background chapter. The approach that has been selected for this thesis is emotion recognition based only on facial expression. No other

factors such as tone of voice, or body posture shall be taken into consideration. A still image of the face will be used to extract its emotional state.

Section 1.3 – Requirements

The approach presented in this study must take into consideration a number of important factors in order for this approach to be a useful and feasible solution. This approach has to be considerably accurate, and has to have the potential to be implemented in real-time. The real-time requirement forces a number of other constraints on the solution. In order for the presented approach to be considered for real-time execution, it has to be relatively fast and simple to calculate the solution. Also real-time application requires that the code be kept as small as possible, with preferably minimal specialized co-processor support. More on real-time execution will be discussed in later chapters.

The emotion recognition system should be able to recognize a selected set of universally recognized facial emotion expressions described by Ekman's work [11]. The six universally recognized emotion states presented by Ekman are: Anger, Disgust, Fear,

Happiness, Sadness and Surprise. The neutral state of the face will also be explored as part of the other six expressions. Therefore a system that can recognize all seven emotional expressions from still images will be explored.

Section 1.4 – Thesis Structure

The overall thesis structure is composed of two components. The first component discusses the development of an emotion recognition system which can be implemented in the study of behavioral robotics. The second component of the thesis discusses behavioral robotics and introduces a humanoid robot which was built as part of this research.

The following is a chapter-by-chapter summary of the overall structure of the thesis. The *Background* chapter introduces previous research which provided the foundation used to develop the emotion recognition system presented in this thesis.

The *Principle Component Analysis* chapter explains why PCA was selected for data compression and how it is implemented for emotion recognition. The mathematics necessary to perform principle component analysis are presented. The principle component analysis is first introduced with a simple example. Then the more advanced subject of PCA application to image compression for utilization in emotion recognition is discussed.

The chapter on *Artificial Neural Networks* introduces the general concept of what is neural network is and what it does. The chapter then discusses why the neural network approach was selected for emotion recognition. This is then followed by a detailed discussion of how a neural network was designed and trained for the purpose of emotion recognition.

Having introduced the two main functional blocks of the emotion recognition system (PCA and ANN), the *Emotion Recognition System* chapter discusses how the two blocks function together to recognize emotion. This chapter's main objective is to introduce the emotion recognition system (ERS) as a whole. The systems block diagram is presented and explained as well as the user interface for the system.

The *ERS Performance Analysis* chapter discusses how the emotion recognition system performed on its cross-validation data set. This chapter further compares the performance of the ERS presented in this thesis, to the performance of similar other emotion recognition systems. Discussion on how the emotion recognition system was evaluated is presented.

Having presented the emotion recognition system as a whole and evaluated its performance, the *Emotion Robotics* chapter is introduced. This chapter discusses why there exists a need for emotion

recognition. Various research projects dealing with humanoid robots that are able to interpret and express emotions are presented here.

The *ERS Hardware and Software* chapter introduces a humanoid robot that was constructed as part of the thesis research. The robots design approach and its hardware and software are discussed. Possible future approaches for implementing ERS in real-time are presented.

Finally the *Conclusion* chapter summarizes what the thesis research achieved, how it compares to similar research projects and how it can be improved in the future.

Chapter 2 - Background

Section 2.1 – Contributions of Shared Ideas

This section describes how various ideas and concepts have lead to the development of the emotion recognition system described in this thesis. Starting with the mathematical invention of principle component analysis, various developments in computational approaches contributed ideas that together allowed the emotion recognition system to be developed. The following is a summary of these ideas.

2.1.1 – Principle Component Analysis

Principle Component Analysis was invented by Karl Pearson in 1906 [21].

This was a statistical approach which reduces multivariable data by identifying variance in the dataset and representing the data in terms of this variance. This is achieved by building a covariance matrix and finding the eigenvectors and eigenvalues of the covariance matrix. The covariance matrix identifies the variance of the dataset. The eigenvector with the largest corresponding eigenvalue identifies the greatest variance in the dataset. Each eigenvector contributes to identifying variance in the dataset and is referred to as the principle component. This process allowed the interpretation of complex datasets and became a popular tool in many fields of research.

2.1.2 – Image Compression based on PCA

A method for compressing images of faces was developed in 1987 by Sirovich and Kirby [25], which was based on principle component analysis. They devised a method for representing an image as a set of weights. Their approach allowed a set of face images to be compressed by representing each image as a set of weights. This set of weights was much smaller in size compared to the original image. The set of weights was obtained by projecting the image onto a set of eigenpictures. The term *eigenpicture* is a term they used to describe an image obtained after calculating which coordinate system would be best used to compress an image. This is the set of orthogonal images

obtained from projecting each image onto the eigenvectors obtained from the covariance matrix. The term orthogonal images, refers to the images identified by the principle components. Since principle components are orthogonal to each other, the images are said to be orthogonal images. Therefore, their method allows a set of images to be compressed into a set of weights using the eigenpictures. These images can then be uncompressed or reconstructed by combining a weighted sum of eigenpictures. Each weight represents a particular eigenpictures involvement in representing the image. Although there is some information loss that is introduced due to this process, it is still a very efficient method of representing face image data in a compressed form. The compression and decompression process is also very simple and computationally not very intensive.

2.1.3 – Eigenface based Face Recognition

The process of using eigenfaces for face identification developed by Pentland and Truk [22] of MIT in 1991 was heavily based on the work of Sirovich and Kirby. They used Sirovich and Kirby’s approach to face compression to represent faces of known and unknown individuals as sets of weights. Eigenpictures or eigenfaces, as Pentland and Turk referred to them could be used to encode a set of known faces as weights. This approach to storing known face images in a compressed

from is very efficient. When a new unknown face is introduced, it is projected onto the eigenfaces, generating a set of weights. The distance between the set of weights for the unknown face and the weights of all the known faces is compared. The distance between the unknown and known face weights is obtained by either finding the Euclidean distance (Equation 2.1) or the Mahalanobis distance (Equation 2.2). If the distance is a very large value, above some predefined threshold value, the unknown input image can be labeled as a non-face image. If the distance is below this predefined threshold, a comparison is made to find between which set of known weights and unknown weights the distance is minimized. Having found the shortest distance between the known and unknown weights, the identity of the unknown face image can be speculated with a high degree of probability. Their approach is beneficial from a number of standpoints. First of all, the known face images are stored in a very compact form. This allows data to be retained about a very large set of known individuals, while requiring very little storage resources. Secondly, in order to identify an individual, very little computational resources are required. Essentially a set of matrices need to be multiplied.

Given $X = (a_1, a_2, a_3, ..., a_n)$ and $Y = (b_1, b_2, b_3, ..., b_n)$

$$Dist(X,Y) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Equation 2.1 – Euclidean Distance Formula.

C = Covariance Matrix

$$Dist(X,Y) = \sqrt{(x - y)C^{-1}(x - y)}$$

Equation 2.2 – Mahalanobis Distance Formula.

2.1.4 – Emotion Recognition System

The emotion recognition system presented in this thesis is based on the work of Pentland and Turk and their face recognition approach. Their approach was selected for a number of reasons. The overall goal is to design a system which would be able to obtain a wealth of important information about a person. Obtaining such information as gender, identity and emotion would be very beneficial in improving human-computer interaction. It would be useful if all of this information could be obtained using the same approach. Therefore, using PCA to obtain all three of these important pieces of information would result in a system which would be able to perform very complex tasks, using a relatively simple unified approach. Since much work has been done in the area of eigenface based face recognition, it was selected to build upon this approach, rather than creating a new and unrelated one. This would lead to a unified system that would be able to obtain much information with little complexly.

The PCA based approach Pentland and Turk use for image compression was selected to be implemented in the emotion recognition system presented in this thesis. This compression technique allows images to be represented with a vector of weights which represents the contribution of each eigenface in representing a particular image. Eigenfaces are essentially images that are obtained from the PCA process which identifies data that is common to all images in the training set. Each image in the training set can be reconstructed (uncompressed) by multiplying a set of eigenfaces by the corresponding weights obtained from the compression process. The approach of Pentland and Turk use these weights (compressed form of an image) to compare unknown images to a database of known images. This comparison is achieved by applying either the Euclidean or Mahalanobis distance formulas.

Unlike Pentland and Turk's approach used to determine the distance between known and unknown face images, the system presented in this thesis will utilize a neural network to achieve this. Rather than depending on functionally limited Euclidean or Mahalanobis distance formulas to determine the distance, the method presented in this thesis will rely on a different approach. The approach selected in this thesis will rely on a neural network to learn what the best method to distinguish images will be. The PCA based compression

method will be used to compress new images generating a set of weights. These weights will then be used as input into the neural network. The neural network will be trained to recognize 7 distinct universal facial expressions.

The neural network essentially learns where each emotional expression is located in the face-space. The face-space is a region within the image space where all images of faces are located. This region is defined by converting all the training images into vectors. These vectors are treated as coordinates defining single points in a multi-dimensional space. The entire space defined by these vectors is the image space. The region where only the points representing faces get plotted in the image space is the face-space. Figure 2.1 compares the similarities and differences of the two approaches (Pentland vs. Thesis Approach).

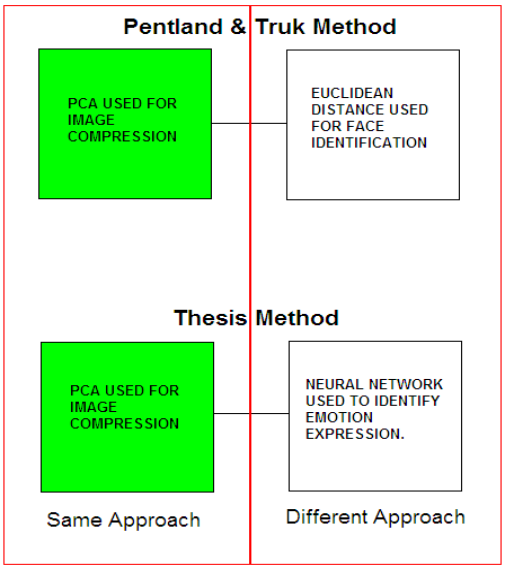


Figure 2.1 – Pentland vs. Thesis Approach.

It is therefore desirable that the identity, emotion and gender, all be obtained by combining these approaches into one unified system. Pentland and Turk's method is able to identify faces, while the method

presented here will be able to distinguish various emotions. Perhaps further research in this area will result in a method for distinguishing gender based on PCA. The completion of all three of these systems would provide humanoid robots with an opportunity to interact with humans more realistically.

Section 2.2 – Universality of Human Emotions

Before the emotion recognition system presented in this thesis is explored in more detail, some basic facts need to be established about the expression of emotions.

Dr. Paul Ekman [11] is one of the leading researchers in the area of facial expressions. His research has brought about the Facial Action Coding System. This system describes which Action Units are required to produce a particular facial movement. Action units are groups of muscles which when activated together, result in a particular movement of facial features. This includes actions such as an eyebrow being raised or lowered or wrinkling of the nose. An example of Action Units is show in figure 2.2. This research became a large contribution to facial expression

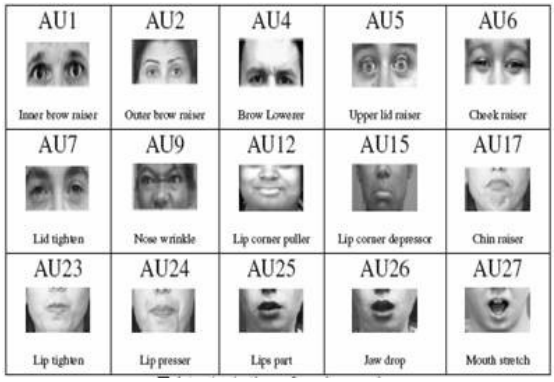


Figure 2.2 – Facial Action Coding Units.

research. In particular, it was a large contribution to psychology research, computer science and was very useful in the field of computer animation.

The work of Ekman, and of those prior to him that studied facial expressions have confirmed that generally all literate cultures express emotions very similarly. Ekman concludes that "*Members of different cultures show the same facial expressions when experiencing the same emotion unless culture-specific display rules interfere."* These studies have mainly been conducted amongst literate cultures. His work also concludes that for the literate cultures, on which this research has been carried out, all facial expressions of emotions can be characterized into one of six active expressions. These facial expressions are anger, disgust, happiness, sadness or distress, fear and surprise. Further research findings suggest that at birth an infant has all of its facial muscles fully developed that are required to express the six basic facial expressions. Studies conducted in 1914 through 1970 confirm that a person's facial expression may be used to accurately distinguish between pleasant and unpleasant emotional states. However it was also confirmed that an individual can manipulate their facial expressions to conceal their actual emotional state. Although Ekman and some of his predecessors have contributed

a great amount to studying facial expressions, this area is still under-researched.

Ekman presented six active emotions which have universal facial experessions associated with them. It will be these six active emotions plus the neutral expression (lack of excitation), for which a recognition system will be developed. By basing the system on these six universal expressions, the recognition system can also be regarded as a universal emotion recognition system.

Section 2.3 – Related Research Overview

A number of different approaches have been proposed for solving the emotion recognition problem. The emotion recognition problem can be defined as the task of collecting information about a person and then interpreting this information in order to determine their emotion. The discussion here mainly focuses on using the facial expression depicted in an image to determine the emotion of a particular person. Methods that require other information about the person along with the facial expression are also briefly introduced. As a reminder, the approach presented in this thesis requires that only a still image of a face expressing an emotion be used for emotion interpretation. While some of the proposed methods are developed

with real-time execution in mind, others are strictly for experimental data collection and analysis purposes.

2.3.1 – Point Analysis Approach

One of the more popular approaches [13, 17] requires identifying particular regions or points on the face and then using these as the basis from which information can be obtained about the emotion expression. Once these points are identified, a number of distance analysis algorithms can be applied to find the distance between the points of interest. These distances are used to conclude which emotion the particular configuration of points represents. The distances between various points are usually used as input into a neural network. The neural network is trained on these distances to learn how to distinguish different emotions.

A similar method where points of interest are found and then information is deduced from the distance ratios of these points is also used in one popular 3-D modeling approach for emotion recognition [7, 10]. In this approach a 3-D mesh is fitted onto the face based on predefined points. The distance ratios along the vectors that define this mesh are used to produce a 3-D

Figure 2.3 – Face Mesh Model.

rendering of the facial expression. The relationship between all the points of interest is expressed as a set of vectors. These are then used as the input to train a neural network for interpretation.

A common problem with the point analysis approaches is that it is very difficult to accurately obtain these points of interest from the face. Most of these points that are used to calculate the distances are located on smooth portions of the face, with no distinct marks or edges to use as indicators as to where a point should be located. Given that almost every face will look different and in most cases extremely different, it is difficult to select a suitable set of points for this type of analysis. Often times these points are manually placed on a set of training images and then the algorithm proceeds from there. When the images are preprocessed in such a manner, promising results have been achieved. The 3-D mesh mask used in the 3-D modeling approach can be seen in figure 2.3, where the points of interest and the vectors joining these points are depicted.

2.3.2 – Compression Based Approach

Some of the other methods [15] require using discrete cosine transform (DCT) to simplify the image data before patterns are recognized. This compressed data is

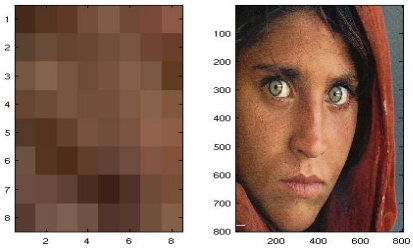


Figure 2.4 – Discrete Cosine Transform.

then used by the neural networks to learn various patters revealed by the simplified data. This approach is very similar to using PCA to simplify the data before using it as input into the neural network. Both methods compress the image data before the neural network uses the compressed image data to identify the emotion expression associated with the image. Any other compression methods would also fall into this category.

2.3.3 – Multimodal Approach

The multimodal approach [4, 5] makes the claim that face images alone are not sufficient to identify emotion. These types of approaches usually require using a number of sources to recognize emotion. For instance, emotion recognition can be based on the facial expression and information about the voice, the gestures made by the arms or posture of the body. This approach requires a number of sources of data to make a conclusion about the emotion of an individual. This method requires a considerable amount of computational resources and is fairly complex. Application of such systems in real-time is debatable, as most of these approaches require a very controlled environment to function properly. If the voice is to be used as a supportive indicator to a particular emotion, then a very large variation of voices and various properties associated with voices

needs to be taken into consideration. A definite distinction needs to be made between male and female voices. Also the emotion expression needs to be perfectly coordinated with a particular voice in order for the two to be supportive of each other. The same conclusions can be made about using a particular physical gesture to support a particular facial expression. Therefore this approach is unlikely to be very successful in real-time applications.

Section 2.4 – Two Approaches to Image Analysis

There are generally two popular methods for analyzing facial expressions. The first approach is the static approach, also known as the target approach. This approach obtains its information from still images. This approach obtains all the information that it needs in order to identify a particular expression from a still image. The second approach is dynamic in nature and is known as the gesture approach. This approach obtains data from a short sequence of images of the expression appearing on the face. This approach has shown more promising results but is more difficult to obtain and process. The exact starting point when the emotion expression begins and when it ends need to be known. This is difficult to obtain in a real-time environment. This method is really only effective in a very controlled environment. The target method of using a still image is simpler to implement and

can be used in a real-time environment. However the target method encompasses less useful information that can be used in identifying a particular expression.

Chapter 3 - Principle Component Analysis

Section 3.1 – Objective

The emotion recognition system (ERS) is composed of two functional blocks, the PCA block and the ANN block. The ANN block is responsible for learning how to recognize various emotions. The purpose of the PCA block is to prepare the input data for the ANN block. It is not very practical for the neural network to implement the entire image as its input. The neural network should only be provided with the essential data that encodes a particular facial expression.

The burden of learning various facial expressions on the neural network can be reduced by providing the neural network with only the essential data necessary to distinguish one expression from another. It

is an easier task for the neural network to learn from data samples that contain only relevant information.

Principle component analysis can be used to eliminate redundancy of image data, thus reducing its complexity and size. The ability of principle component analysis to considerably reduce data size and complexity makes PCA a very good candidate as a preprocessing agent for the neural network. As an example if an image of the face region is 300 x 240 pixels then a total of 72,000 pixels need to be processed. By applying the PCA process to the image, the data is reduced from 72,000 pixels (8-bits per pixel) to 127 weights (64-bits per weight). The application of PCA on the input data in this case results in about a 70% reduction. This is a considerable reduction allowing the neural network to train on only the essential data that is required to identify various expressions of emotion. Due to the ability of the PCA process to compress data efficiently, PCA has been selected as the preprocessing step for the neural network.

Section 3.2 – PCA Introduction

This section will introduce principle component analysis with a simple example. Starting with a simple example allows a good understanding of the subject to be discussed, before more complex applications of PCA are introduced. Understanding what each step in the PCA process

is trying to achieve will make it easier to understand how PCA can be applied to image processing.

1) Data for Principle Component Analysis

Table 3.1 below summarizes the data selected to be used for the PCA example. The 2-dimensional data was generated in such a way that the two variables (X and Y) have a positive correlation. Positive correlation means that if the overall trend of variable X is increasing, then Y also increases. The positive correlation of the dataset can be seen in figure 3.1. The data sample is intentionally kept very small to minimize complexity of performing principle component analysis on the dataset.

Variable X	Variable Y
1	38
3	33
6	30
10	35
9	38
12	32
20	48
22	47
19	45
16	50
25	53
26	52
18	55
16	58
33	62
30	65
29	67
27	62
20	68
31	70
Average X 18.65	**Average Y 50.4**

Table 3.1 - Dataset for PCA Example

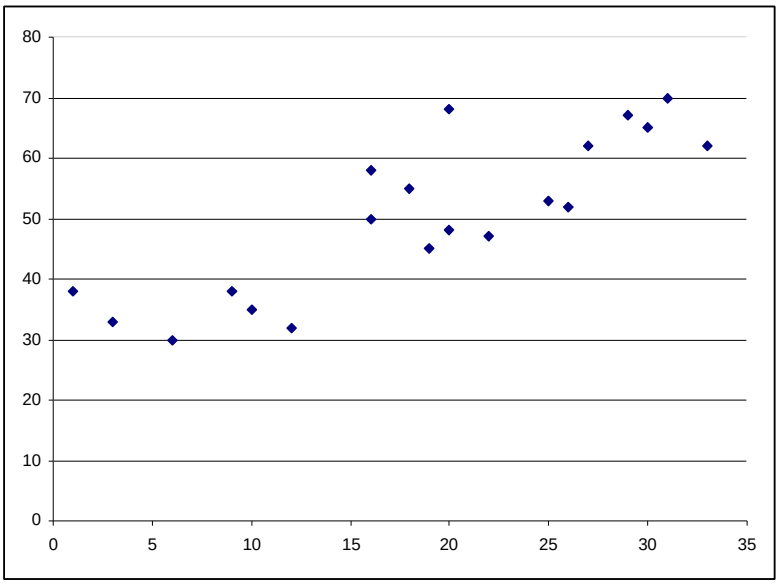


Figure 3.1 – Plot of Dataset for PCA Example.

2) Covariance Matrix Calculation

Having established the dataset, a covariance matrix can now be computed. A covariance matrix will identify the variation amongst all the variables in the dataset. The question that a covariance matrix addresses is how much the variables vary, from the mean, with respect to each other. Equation 3.1 is used to compute the covariance matrix.

$$COV(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})$$

Equation 3.1 – Covariance formula

A covariance matrix can be constructed by applying equation 3.1 to all pairs of the variables involved. In this case there are only two variables involved, X and Y. All possible combinations of variables X and Y are (X,X), (X,Y), (Y,X) and (Y,Y). In general if N variables exist in the dataset, the resulting covariance matrix will be NxN. In this example, the use of two variables results in a 2x2 matrix. The structure for a 2x2 variable matrix can be seen in figure 3.2.

X-Average(X)	Y-Average(Y)

$$\begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{cov}(y,y) \end{bmatrix}$$

Figure 3.2 – Covariance Matrix for Two Variable Dataset.

0.35	-5.4
-2.65	-0.4
6.35	2.6
7.35	1.6
-0.65	4.6
-2.65	7.6
14.35	11.6
11.35	14.6
10.35	16.6
8.35	11.6
1.35	17.6
12.35	19.6

Table 3.2 – Dataset with Average Subtracted.

Table 3.2 is the same as table 3.1, except that each data member has the average subtracted from it. That is, all X variables have average X value subtracted form them and all Y variables have the average Y values subtracted from them. The data in table 3.2 is essentially the calculation of $(X - \overline{X})$ and $(Y - \overline{Y})$ for equation 3.1.

The rest of the calculations of the covariance matrix can now be carried out. Table 3.3 summarizes the rest of the calculations for every X and Y variable pair. Each column in table 3.3 indicates the order in which the mean-centered values from table 3.2 were multiplied together. Table 3.2 and table 3.3 are essentially various components of equation 3.1 being calculated one step at a time.

3)

	(X,X)	(X,Y)	(Y,X)	(Y,Y)
	311.5225	218.86	218.86	153.76
	244.9225	272.31	272.31	302.76
	160.0225	258.06	258.06	416.16
	74.8225	133.21	133.21	237.16
	93.1225	119.66	119.66	153.76
	44.2225	122.36	122.36	338.56
	1.8225	-3.24	-3.24	5.76
	11.2225	-11.39	-11.39	11.56
	0.1225	-1.89	-1.89	29.16
	7.0225	1.06	1.06	0.16
	40.3225	16.51	16.51	6.76
	54.0225	11.76	11.76	2.56
	0.4225	-2.99	-2.99	21.16
	7.0225	-20.14	-20.14	57.76
	205.9225	166.46	166.46	134.56
	128.8225	165.71	165.71	213.16
	107.1225	171.81	171.81	275.56
	69.7225	96.86	96.86	134.56
	1.8225	23.76	23.76	309.76
	152.5225	242.06	242.06	384.16
SUM:	**1716.55**	**1980.8**	**1980.8**	**3188.8**
SUM/N:	**90.34474**	**104.2526**	**104.2526**	**167.8316**

Table 3.3 – Calculations for Each Set of Variables.

Understanding the Covariance Matrix

Having calculated the values for the covariance matrix in table 3.3, the 2x2 matrix can now be populated. The covariance matrix for the selected dataset is shown in figure 3.3.

$$\begin{bmatrix} [x,x] = (90.34) & [x,y] = (104.25) \\ [y,x] = (104.25) & [y,y] = (167.83) \end{bmatrix}$$

Figure 3.3 – Covariance Matrix of Dataset.

Inspecting the matrix in figure 3.3, it can be seen that (X, Y) and (Y, X) values are identical. From the covariance formula, it can be concluded that the product between $(X - \overline{X})(Y - \overline{Y})$ produces the same results as the product between $(Y - \overline{Y})(X - \overline{X})$. The diagonal of the matrix corresponds to the variance of individual variables. The covariance of one variable with itself results in the variance of that variable. The variance of a dataset provides information on how the data is spread with regard to the mean. The formula for variance is shown in equation 3.2.

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n - 1}$$

Equation 3.2 – Variance Formula.

Returning back to the subject of the covariance matrix, the data that is of interest are the covariance values of different variable pairs. In this case, it is the (X, Y) and (Y, X) values. These indicate the correlation

that exists between the two variables in the dataset. If this value which is identical for both pairs is positive, then there exists a positive correlation between X and Y. That is, X increases with Y. If the covariance value is negative, then this suggests a negative correlation. As X increases, Y decreases. If the covariance value is zero, then this indicates that there is no correlation between the two variables. In this example the covariance value is positive, suggesting that there exists a positive correlation between the two variables. This was expected since the dataset was generated in such a way that the two variables have a positive correlation.

The conclusions which can be deduced from the covariance matrix do not seem very impressive when dealing with two dimensions. However, when the dataset is composed of hundreds or even thousands of variables, this technique for gaining insight into correlations which might exist between variables is very useful. In systems where a large number of variables are involved, such analysis allows conclusions to be formed about which variables affect the system the most and which variables have little overall affect.

4) Derivation of Eigenvectors and Eigenvalues

For a square matrix A, the value λ is the eigenvalue for a non-zero vector v, which satisfies equation 3. 3. The non-zero vector associated

with the eigenvalue is the eigenvector. The eigenvalues of matrix A can be obtained by solving the characteristic polynomial of A, given in equation 3.4. In this example, A is the covariance matrix, I is the identity matrix, λ is the eigenvalue and v is the eigenvector corresponding to matrix A. Using equation 3.4 to solve for λ results in the eigenvalues of the covariance matrix.

$$Av = \lambda v$$

Equation 3.3 – Eigenvector to eigenvalue relationship.

$$\det(A - \lambda I)v = 0$$

Equation 3.4 – Characteristic polynomial of matrix A.

Calculating Eigenvalues:

$$\begin{bmatrix} 90.34 & 104.25 \\ 104.25 & 167.83 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 90.34 - \lambda & 104.25 \\ 104.25 & 167.83 - \lambda \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 90.34 - \lambda & 104.25 \\ 104.25 & 167.83 - \lambda \end{bmatrix} \right)$$

$$= (90.34 - \lambda)(167.83 - \lambda) - (104.25)(104.25)$$
$$= (90.34 - \lambda)(167.83 - \lambda) - 10868.1$$
$$= \lambda^2 - 258.17\lambda + 4293.66$$

$$\lambda_1 = 17.869$$
$$\lambda_2 = 240.31$$

Equation x – Characteristic Polynomial of Matrix A.

From the set of steps above, λ_1 and λ_2 are derived. These values correspond to eigenvalue 1 and eigenvalue 2 of the covariance matrix calculated in earlier steps.

Having calculated the eigenvalues of the covariance matrix, it is now possible to calculate the corresponding eigenvectors. The eigenvectors and eigenvalues will be used to identify data patterns in the original dataset obtained from the covariance matrix.

It is important to mention that some of the mathematical approaches presented here are only applicable to small matrices under 4x4. The technique presented below is applicable to small matrices. This technique allows the eigenvectors of a small matrix to be found by following a few outlined steps.

Calculating Eigenvectors for 2x2 Matrices:

Let us define matrix: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ If c is non-zero, then the eigenvectors for such a matrix can be defined as:

- $eigenvector_1 = \begin{vmatrix} \lambda_1 - d \\ c \end{vmatrix}$
- $eigenvector_2 = \begin{vmatrix} \lambda_2 - d \\ c \end{vmatrix}$

Using this approach, the eigenvectors for the covariance matrix are calculated.

Calculating Eigenvector for λ_1:

$$A = \begin{bmatrix} 90.34 & 104.25 \\ 104.25 & 167.83 \end{bmatrix}$$

$$\lambda_1 = 17.869$$

First Eigenvector:

$$V_1 = \begin{bmatrix} \lambda_1 - d \\ c \end{bmatrix}$$

$$V_1 = \begin{bmatrix} 17.869 - 167.83 \\ 104.25 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} -149.961 \\ 104.25 \end{bmatrix}$$

After Normalizing Eigenvector:

$$d = \sqrt{a^2 + b^2}$$

$$d = \sqrt{-149.961^2 + 104.25^2} = 182.637$$

$$\begin{bmatrix} \frac{-149.961}{d} \\ \frac{104.25}{d} \end{bmatrix} = \begin{bmatrix} -0.82108 \\ 0.57080 \end{bmatrix}$$

Normalized Eigenvector for λ_1:

$$v_1 = \begin{bmatrix} -0.82108 \\ 0.57080 \end{bmatrix}$$

Calculating Eigenvector for λ_2:

$$\lambda_2 = 240.31$$

Second Eigenvector:

$$V_2 = \begin{bmatrix} \lambda_1 - d \\ c \end{bmatrix}$$

$$V_2 = \begin{bmatrix} 240.31 - 167.83 \\ 104.25 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} 72.48 \\ 104.25 \end{bmatrix}$$

After Normalizing Eigenvector:

$$d = \sqrt{a^2 + b^2}$$

$$d = \sqrt{72.48^2 + 104.25^2} = 126.97$$

$$\begin{bmatrix} \frac{72.48}{d} \\ \frac{104.25}{d} \end{bmatrix} = \begin{bmatrix} 0.5708 \\ 0.8210 \end{bmatrix}$$

Normalized Eigenvector for λ_2:

$$v_2 = \begin{bmatrix} 0.5708 \\ 0.8210 \end{bmatrix}$$

5) Eigenvectors and Eigenvalues of the Covariance Matrix

Having calculated the eigenvectors and eigenvalues of the covariance matrix, important patterns can now be identified in the original data. In particularly the eigenvalues and eigenvectors can be used to identify variance in the dataset. Figure 3.5 shows how the eigenvectors identify the variance in the normalized dataset. That is, the data with the mean subtracted from each variable. In figure 3.5 the individual points represented with diamonds are the normalized data points. The line that goes through all the data points (major trend) is the main principle component. The main principle component is the eigenvector with the largest corresponding eigenvalue. The line that is perpendicular (minor trend) to the main principle component is the secondary principle component. The secondary principle component is the second eigenvector corresponding with the second largest eigenvalue. The lines were obtained using the eigenvectors.

$$\begin{bmatrix} 90.34 & 104.25 \\ 104.25 & 167.83 \end{bmatrix}$$

(a) – Covariance Matrix

$$v_1 = \begin{bmatrix} -0.82108 \\ 0.57081 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0.57081 \\ 0.82108 \end{bmatrix}$$

(b) - Eigenvectors

$\lambda_1 = 17.869$

$\lambda 2 = 240.31$

(c) - Eigenvalues

Figure 3.4 – Summary of Calculated Data.

$$Y_{MAJOR} = \left(\frac{0.82}{0.57}\right) X ,$$

$$Y_{MINOR} = \left(\frac{-0.57}{0.82}\right) X$$

Therefore having found the eigenvectors and eigenvalues, the eigenvectors are sorted by their eigenvalues. This results in a set of principle components ordered by how much variance each component represents. The two principle components have identified patterns in the dataset. Each principle component represents a certain portion of the entire data. Since the size of the covariance matrix was 2x2, there are only two principle components defining the trend of the data. Having obtained the principle components, the data can now be compressed.

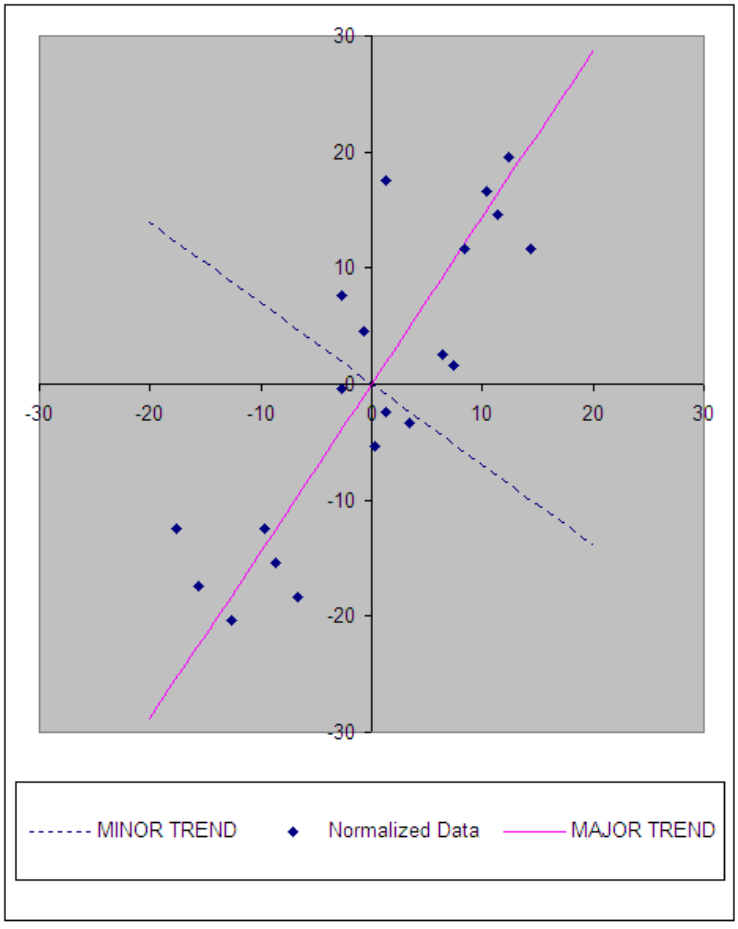


Figure 3.5 – Eigenvectors Identifying Data Variance.

6) Relating to PCA for Image Compression

Having established the fundamental steps necessary to obtain the principle components of a particular dataset, the discussion turns to how this process can be applied to images for the purpose of compression. Here only the theoretical discussion will take place on how the steps in the above example can be applied to a more complex dataset. The actual application with the mathematical background will be provided later in this chapter. This section serves to link the concepts provided in the simple example above with the more complex application to be discussed later in this chapter.

The steps presented in the above example can be applied to image compression. An image can be converted into a vector by concatenating either the rows or the columns of the image together. A large set of these image vectors can be used to create a matrix, where each column in the matrix is a vector corresponding to an image. The image matrix can be used to calculate the covariance matrix. The eigenvectors and eigenvalues can then be found from the covariance matrix. The eigenvectors are then sorted by their corresponding eigenvalues. These are essentially the principle components as they identify the patterns in the dataset.

If this particular set of images were of faces, the data patterns which would be identified by the principle components would be the space of all the faces. Each principle component would contribute a certain amount to representing the space of faces. This space of faces can be referred to as the face-space. The conversion of each image into a vector converts the data from being interpreted as a 2-dimensional dataset into a N^2-dimensional dataset for an NxN image. Therefore the principle components would identify this *face-space* in N^2-dimensional space. The N^2-dimensional space is the entire space that can be represented by an NxN image converted into a N^2 vector. The face-space is a sub-region within the entire N^2 image space. The principle components identify this sub-region that has been labeled as the face-space. Each principle component represents a certain portion of the entire face-space. All these portions together represent the entire face-space.

By multiplying each principle component by the entire dataset, various portions identified by the principle components can be extracted. Such an operation would result in a set of vectors which would represent the underlining data corresponding to each principle component. These extracted data vectors contain face related data. Since the eigenvectors identified this face related data, these vectors can be referred to as eigenfaces. Therefore when each principle

component is multiplied by the dataset, the result is a set of eigenfaces. The eigenface is nothing more than the data identified by the eigenvector (principle component). Figure 3.6 shows some eigenfaces calculated from a database of face images.

Furthermore, it has been established that the sum of all the data identified by the principle components defines the entire face-space. This would indicate that the sum of all the eigenfaces would encompass all the face data contained in the face-space. Therefore any face within the face-space can be represented by some combination of the eigenfaces. If a new image of a face is introduced and converted into a vector, it too would be located in the face-space since it shares similar properties to other faces. This would mean that this new face images can also be represented by a specific sum of eigenfaces. This means that any face can be represented as a weighted sum of the eigenfaces. Each weight represents the particular contribution of each eigenface to representing a particular face image. It follows that any face image can be represented by the same amount of weights as there are principle components. This means that a large vector representing an image can be represented with a small set of weights. The representation of the large image vector as a small set of weights results in compression.

Once a set of eigenfaces has been calculated, they can be applied to any image of a face to represent that face in a more compressed form as a weighted sum of eigenfaces. Each image can then be represented as a set of weights. A set of weights can then be used to reconstruct the original image by multiplying the eigenfaces by the corresponding weights and summing the result. It is these weights that are the goal of applying principle component analysis to images.

Figure 3.6 – Example of Eigenfaces.

Section 3.3 – PCA Discussion

Defining the Face-Space

Each N^2-dimensional image vector represents a single point in an N^2-dimensional space. Since the image vector is of a face, each N^2-dimensional point is representative of that particular face. If a set of M face image vectors are plotted in the N^2-dimensional space, the points will all cluster in very close proximity of each other due to their shared characteristics. To reduce very tight clustering of all the points the average image can be subtracted from each face. The average image is essentially the vector which results from obtaining the average of all the image vectors in the set M. Each point defined by a face vector will be different from other face points. However faces are generally similar and share many common features. By removing the average of all the faces, it is possible to remove all that is common between the faces leaving only the differences. Plotting only the differences of each face, results in a better distribution of the face data. This allows the points or faces to be distinguished easily. If the points or faces are close together, they are difficult to distinguish. The space that is defined by these image points is only a very small portion of the entire N^2-dimensional space. This cluster of face vectors essentially defines a sub-space in the entire N^2-dimensional space where all face images will be mapped to. Any new face image vector plotted in the N^2-D space will be mapped to this sub-space unique to face images. Therefore by plotting M face images, a face-space is defined. By making M large, the

face-space is made more defined. This space is very important because this is the region of interest. The rest of the space outside of the face-space is of no use for the purpose of emotion recognition.

Recognizing Faces in the Face-Space

The importance of the face-space becomes evident when a new unknown image is introduced. This new unknown image can be of anything. However it needs to be the same size as the images that were used to define the face-space. This image is then projected onto the face-space. An average image of all the known faces is also projected onto the face-space. The distance between the two images is calculated. If the distance of the unknown image to the average face image is beyond some set threshold, the unknown image may be regarded as a non-face image. That is, the image is of something other then a face. By simply checking the distance of an unknown image to a known face image, it may conclude whether the unknown image is of a face or not. The classification between face or non-face image is important and useful. This same approach can be used to not only identify whether an image is of a face or not, but also to identify the face itself. The application of this process to face identification is where this approach has gained popularity.

At the most basic level an image can be characterized into face and non-face images. But the ability to take an image and recognize it as being that of a previously defined face makes this process incredibly useful and powerful. A similar approach is taken to perform this recognition process. Images are obtained of known people. Each known person is defined by a set of images, all of which are slightly different. These differences might be clothes they are wearing, the way they style their hair or having glasses on or off. The orientation of the face might be varied to some degree as well. All groups of images collected for every known person are averaged together. Each averaged set of images for every person is referred to as a class. A class is nothing more than a region within the face-space that belongs to a single person. There may be a single class or a very large set of classes. Each class is projected onto the face-space. The process of projecting each class onto the faces-space results in as set of weights being generated. Each class will be represented by this set of weights. When a new unknown image is introduced, it may first be used to check whether it falls into the face-space at all. When the input image is determined to be an image of a

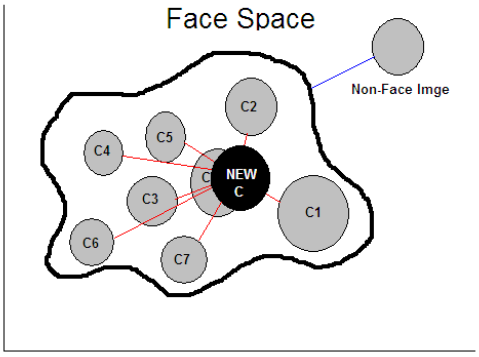


Figure 3.7 – Face Classes within Face-Space.

face, it can be identified. Identification of the new images is achieved by projecting the new unknown image onto the face-space. This results in a set of weights. These weights are then compared to the weights of each class. The distances between the input image and all the classes are calculated. The shortest distance between the input image and one of the classes indicates that a particular class is likely to be the identity of the image. This concept is depicted in figure 3.7.

Difference between Recognizing Faces and Expressions

There is an important difference between recognizing faces in the face-space and recognizing emotions. One face, or point in the face-space will be considerably different from another face. This is due to faces generally being different in appearance. Therefore there exists a distinguishable distance between each class. One class can be distinguished from another class with the approach discussed earlier. However the distance between one expression and another expression for the same face is far more difficult to distinguish in the face-space. A single face can express many different emotion expressions all of which differ very little from on another. Some of these expressions will differ more than others. For example sad and neutral are very similar in appearance, while neutral and surprised are considerably different from each other. But even with these differences it is very difficult to distinguish such minute facial differences when these are represented

as points in the face-space. For this reason a new approach must be selected to distinguishing points in the face-space which differ very little from each other.

Neural Network Approach

Finding the distances between image classes and input images is usually based on formulas like Euclidean distance. However applying a formula such as this to very tightly plotted, highly dimensional data has demonstrated poor performance. Therefore a different approach to identifying various classes will be selected. This approach will depend on a neural network learning how to recognize various emotion expression classes within the face-space. The neural network approach has demonstrated far better performance. The function of principle component analysis will serve only as a technique for image compression. The compressed images will be used as the input for the neural network.

Section 3.4 – PCA Application

This section will discuss how principle component analysis can be applied to images for the purpose of compressing them, thus allowing them to be effectively utilized by the neural networks. The PCA compression process results in an image being represented by a set of weights. These weights are used by the neural network as the

input for both the training phase and the execution phase. The following set of steps describes how PCA is applied to image compression which results in a set of weights used by the neural network.

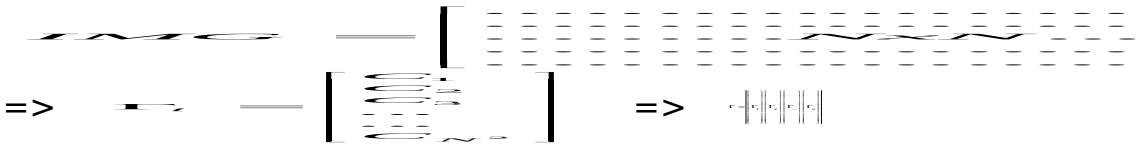
The steps presented below are similar to those described in the *Introduction* section presented earlier in this chapter. The only real difference is that the steps presented below are specific to image compression, whereas the approach described above is a generic approach to principle component analysis.

STEP 1 – Obtaining Grayscale Image Dataset:

Each grayscale image I is treated as an NxN data matrix with each data element consisting of an 8-bit pixel intensity value. It is not a requirement that the image be square. All the images in the dataset must be equal in size. Also, all the faces in the images must be relatively equal in size and orientation.

STEP 2 – Converting Images into Vectors:

A Γ_i vector is created by resizing each NxN matrix I into an N^2x1 vector. This is achieved by concatenating all the rows or columns of matrix I to form the N^2x1 vector. A total of M such vectors are combined to form the columns of the new matrix Γ. There are a total of M images that are used to form the Γ matrix. The dimensions of the Γ matrix are N^2xM.



a) Original Image **b)** Resized to N^2x1 **c)** M vectors combined to form Γ

Figure 3.8 – PCA Image Matrix Manipulation.

STEP 3 – Creating an Average Vector:

At this point all the images are represented as columns in the newly constructed Γ matrix. The M images contain information that is common to all of them. It is necessary that non-redundant data for each image vector (Γ column) is retained, while all unnecessary redundancy amongst the images is removed. This is achieved by obtaining the average image vector Ψ, which is the mean of Γ. This average vector Ψ is obtained by summing all of the columns of Γ and dividing this sum by M. The number of total images being analyzed (Γ columns) is M. Equation 3.6 is used to obtain the average face image.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

Equation 3.6 – Average Face Formula.

STEP 4 – Subtracting Average from All Vectors:

When the average vector Ψ is obtained, the original Γ matrix is modified to remove the redundant data patters as described above. The Ψ vector is used to modify the original Γ matrix removing the redundant data patters. This is achieved by subtracting Ψ from every Γ column using equation 3.7. Applying this process to the Γ matrix results in a new matrix Φ, where each column represents the non-redundant image data not shared amongst other columns. Subtracting the mean from every image also functions to spread the data points out in the N^2 image space.

$$\Phi_i = \Gamma_i - \Psi$$

Equation 3.7 – Normalizing Image Vectors.

STEP 5 – Generating the Covariance Matrix:

The newly constructed matrix Φ is used to generate the covariance matrix. The covariance matrix represents how much the dimensions vary from the mean with respect to each other. This matrix will expose the relationship that exists amongst all the data. Covariance is always

measured between two dimensions. As in the example before, equation 3.8 is used to calculate covariance between two dimensions.

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})$$

Equation 3.8 – Covariance Matrix Formula.

The covariance will expose the relationship between two vectors. If the resultant value is greater than zero, then the relationship between the two mean centered vectors is such that they both increase together (positive correlation). If the resultant value is negative, then there exists an inverse relationship between the two vectors such that one vector increases while the other decreases (negative correlation). In the case the resultant is zero, the two dimensions are independent (no correlation). Since matrix Φ is a mean centered Γ matrix, the covariance matrix can be generated using equation 3.9, resulting in a covariance matrix C with N^2xN^2 dimensions.

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T$$

Equation 3.9 – Modified Covariance Matrix

STEP 6 – Calculating Eigenvector and Eigenvalues:

The covariance matrix is used to calculate the eigenvectors and eigenvalues. The eigenvectors and eigenvalues will identify data variance obtained from the covariance matrix. If matrix A is defined as $A = [\Phi_1, \Phi_2, \ldots, \Phi_M]$, then the relationship in equation 3.10 can be derived.

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T$$

Equation 3.10 – Equation Redefinition.

Applying equation 3.10 results in an N^2xN^2 covariance matrix C, which is too large to be used effectively to find the eigenvectors and eigenvalues. To find the eigenvectors and eigenvalues of the covariance matrix C, the following matrix is defined: $L=A^TA$.

$$L=A^TA \quad (MxM)$$

Equation 3.11 – New Matrix Definition.

The L matrix is used instead to find eigenvectors and eigenvalues. This results in the M eigenvectors and eigenvalues that are of interest to us.

1. $L = A^T A$	-- Smaller MxM matrix.
2. $AV_i = \mu_i V_i$	-- Formal definition.
3. $(A^T A)V_i = \mu_i V_i$	-- Replace A with $L=A^T A$.
4. $A(A^T A)\ V_i = A(\mu_i V_i)$	-- Multiply both sides by A.
5. $(AA^T)\ AV_i = \mu_i(AV_i)$	-- Rearrange Equation. Note: $C=AA^T$
6. $(C)\ AV_i = \mu_i(AV_i)$	-- Replace AA^T with its definition C.
7. $(C)\ U = \mu_i(U)$	-- Replace $AV_i = U$.
8. $CU = \mu_i U$	-- Where $U = AV_i$.

This is possible because of the following relationship that exists between the C matrix and the L matrix:

The following conclusion can be deduced from the above steps: AA^T and A^TA have the same eigenvalues. The eigenvectors are related by factor $U = AV_i$. The eigenvectors of AA^T are equal to those of A^TA related by factor AV_i. Applying this technique it is possible to find the eigenvectors and eigenvalues for the very large covariance matrix AA^T by first solving the smaller A^TA matrix and multiplying the eigenvectors by A. The eigenvectors are usually normalized in which case the factor will not play a role.

STEP 7 – Identifying Principle Components:

The eigenvectors are sorted by their corresponding eigenvalues. The larger the eigenvalue associated with a particular eigenvector, the more variance this eigenvector represents. The eigenvectors with their associated eigenvalues are the principle components. Each eigenvector will be orthogonal to all other eigenvectors. The process of identifying eigenvectors results in the definition of a new coordinate system. This new system describes the data in terms of variance amongst dimensions.

STEP 8 – Calculating Eigenfaces:

After the eigenvectors have been sorted, they are multiplied by the mean-centered images Φ_I to generate eigenfaces. Eigenfaces are images which the eigenvectors (principle components) identify. Each

eigenface will represent a particular component of the entire face-space. Equation 3.12 is used to calculate the eigenfaces. This calculation results in M eigenfaces (vectors) of size N^2.

$$\mu_i = \sum_{k=1}^{M} V_{ik}\Phi_k \quad \text{For i = 1, 2, . . ., M}$$

STEP 9 – Eigenface Discussion:

This concludes the set of steps that must be performed to obtain the eigenfaces. The benefit of this approach is that the eigenfaces only need to be calculated once. All future input images can be converted into weight vectors (compressed) by being projected onto the already calculated eigenfaces. Although the process to calculate the eigenfaces is somewhat time consuming, it must be done only once.

Step 10 – Image Compression:

Each new image can be converted into a set of weights (compressed) by being projecting onto the eigenfaces. This is accomplished by implementing equation 3.13. This processes results in M weights represented by vector ω.

$$\omega_k = \mu_k^T(\Gamma_{new} - \Psi) \quad \text{For k = 1 to M}$$

Equation 3.13 – Eigenface Equation

Chapter 4 - Artificial Neural Networks

Section 4.1 – Introduction to Neural Networks

Neural networks are computational models which are biologically inspired. In particular, neural nets attempt to model how the human brain works. At a very fundamental and simplified level the human brain contains a large amount of neurons. These neurons are interconnected with each other in various ways. Each neuron can be simplified to an input output transaction model. The neuron, represented in figure 4.1 (Top) is composed of three main components. These three

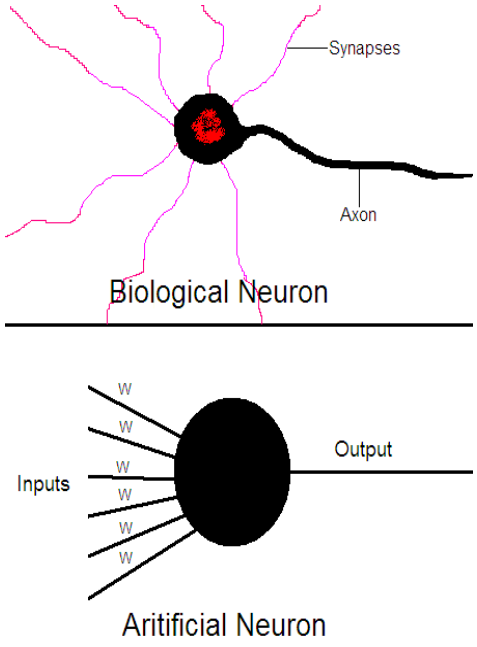


Figure 4.1 – Biological and Artificial Neuron.

components are the Nucleus (the dark round object), the Synapses and the Axon. The neuron functions by receiving electrochemical signals on the Synapses and when a certain threshold is reached, an electrochemical signal is fired on the Axon. Each Synapses signal leading into the neuron nucleus is the axon output of some other neuron. This network of neurons ranging in the billons is all interconnected in some complex manner. Stimuli throughout our lifetime train this network by breaking some Synapses connections or creating new ones. Also the triggering function that causes the Axon to fire is adapted. Again, at a very simplified level, this model represents how neurons in our brains allow us to learn.

Artificial neural networks try to simulate this exact phenomenon. An artificial network can be described in the same manner as a biological one. A neuron, also sometimes called a processing element is composed of a set of inputs and an output just like its biological counterpart. Each input has a weight associated with it. It is these weights that are adapted during the training

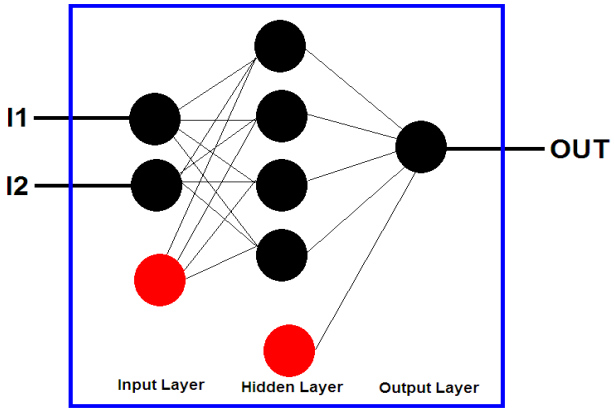


Figure 4.2 – F.F Neural Network Model with B.P.

process. At a very fundamental level the neuron would get its input data from the weighted input signals. A predefined threshold value would be used as a triggering mechanism for the output. However neurons use various more advanced approaches to selecting when the output should fire. These functions are called activations functions. And there exists a variety of them to select from. Some activation functions are better than others at particular learning tasks.

As the term network indicates, a neural network is composed of many neurons. These neurons are interconnected in different ways for different type of networks. Figure 4.2 depicts what a typical fully connected feed-forward network with back-propagation is structured like. A network of this type is composed of an input layer, a hidden layer and an output layer. The input layer interacts with the input stimuli directly. The hidden layer may be composed of multiple layers and is where the majority of the learning takes place. The output layer interacts with the outside world. Fully connected indicates that every neuron in layer N is connected to every other neuron in layer N+1. Even though every neuron in layer N is connected to every other neuron in layer N+1, it doesn’t mean that every weighted connection will be active. More on this will be said later. A feed-forward network indicates that all data flows from input to output. There are no bi-directional connections.

Once a desired neural network is constructed for a particular purpose, it must then be trained. During the training process the weights which are associated with the neuron interconnections are derived. The goal is to present a large training set to the neural network during the training period and have the weights adapt to this training set. After the training period a new and previously unseen input is presented of similar type to the neural network. The goal is to have the neural network make generalized decisions with input which has never been experienced. Training can take on two forms, reinforced or supervised learning and unsupervised learning. Supervised learning depends on the correct output response being provided for each input in the learning set. A popular algorithm that is often used with this approach is back-propagation. This approach measures the error rate at each neuron after an input has been presented to the network and an output has been observed. Starting with the output layer, the presented output is compared to the expected output and the error is used to modify the weights. This process propagates backwards to each neuron modifying the weights to minimize the difference between actual and expected. Along with different learning algorithms, there are also different approaches as to where to train the network. If the training takes place outside of the environment where it is going to be utilized, it is referred to as offline

learning. Where as training that takes place in the environment where this network will later be utilized, it is considered online learning. Neither of these approaches is necessarily better then the other. Selection of everything ranging from number of neurons per layer, to learning style all depends on the application of the neural network. There is no one particular approach that works best for all applications.

Once the training is complete, the network is ready to be utilized in the environment it was intended for. If the network is capable of generalizing input data similar to what it used at the input set for training, then the network has learned successfully and may be used. If it fails to generalize properly, it may need to be modified and retrained until desired performance is attained.

Section 4.2 – Neural Network Approach

Having described how a neural network functions and what it is capable of achieving, it is important to establish why the use of a neural network was selected. Why use something like a neural network rather than a known algorithm like Euclidean or Mahalanobis distance to identify emotions? For the purpose of identifying faces, as was implemented by Pentland and Turk it might be suitable to use these distance algorithms. Each face within the face-space is considerably distant from one another. They are distant from one another because

each face is rather different in appearance. Due to this, it is possible to use the algorithmic approach to distinguish the faces. However to distinguish emotion, it is far more difficult to use the algorithmic approach successfully. This is because emotion expressions are not as spread out within the face-space as faces of different individuals. For example, the same face can express all seven emotional expressions. Some of the emotion expressions are considerably similar in appearance. For instance the neutral and sad expressions look considerably alike. This means that they would occupy relatively the same area within the face-space. The same applies to the expressions for surprised and fearful. The happy expression in its more subtle form will be located closer to either neutral or sad. Because these expressions are confined to such a relatively small area, the task of distinguishing them becomes a challenging one. The attempt to use Euclidean distance resulted in successful recognition rates ranging in the 50% range. This indicated that a simple formulaic approach was not a viable one for the purpose of emotion expression recognition. For this reason neural networks were considered for this particular purpose.

By using a neural network the limitations that are introduced by a formula are eliminated. The neural network learns the best approach to identifying specific emotions. By training the network on a large

enough dataset, it is possible to learn a very efficient way of characterizing emotion expressions. The emotion expressions are within a very confined expression space located in the face-space. The only input provided to the neural network is the weights of each image. These weights essentially are the projection of the input image onto the eigenfaces. What the neural network is forced to learn is the how the relative location of each input image is associated with a particular emotion expression. This approach has shown far more promising results than simply applying the Euclidean distance formula.

Section 4.3 – Designing Neural Network for ERS

The neural network is designed for ERS in such a way that it takes as input a set of weights and outputs the probability for one of seven emotions. The input weights are the projection of the input image onto the eigenfaces. There are 127 eigenfaces which result from the use of 127 training images to define the face-space. All 127 eigenfaces are used to define the weight vector, thus resulting in most accuracy.

The type of neural network selected to function as the learning agent for the ERS is a feed-forward neural network with back-propagation. The input layer consists of 127 inputs, the single hidden layer consisted of 300 neurons and the output layer consisted of 7 neurons. The number of neurons in the input layer was dictated by the

size of the weight vector. The output layer was constrained by the size of the output, or in this case the number of emotions being detected. The number of neurons in the hidden layer was obtained through trial and error. There are really no particular rules that govern the number of neurons that should be used in the hidden layer. If too few neurons are used, the neural network is unable to learn successfully. If too many neurons are used then a phenomenon called over-fitting may occur.

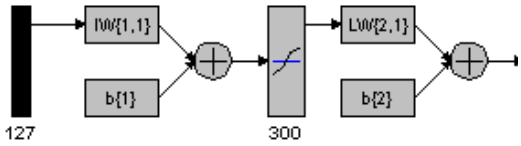


Figure 4.3 – ERS Neural Network Configuration.

Over-fitting occurs when the neural network begins to memorize its training data rather than trying to learn the rules that define it. A neural network which over-fits its dataset will function with a very high success rate on the training data. Usually it will be able to learn the training data exactly. However the neural network is no longer capable of generalizing. This essentially eliminates the desired functionality of a neural network. We do not wish to memorize a training set, instead we wish to learn from it and then have the ability to generalize on new input presented in the future. For this reason the hidden layer was not adjusted to contain the maximum number of neurons. A network with 300 hidden layer neurons seemed to learn well, while containing the ability to generalize on new previously unseen inputs.

Section 4.4 –Neural Network Training Statistics

Once the structure of the neural network for the ERS was established, it was then trained on a training set consisting of 127 images. The training set consisted of both male and female images expressing all seven emotions. Table 4.1 is a summary of the training set that was implemented in the training of the neural network. This training set is considerably small and a larger training set would have undoubtedly resulted in better overall neural network performance.

		Percent
Male:	49	**39%**
Female:	78	**61%**
Angry:	19	15.0%
Disgusted:	16	12.6%
Fearful:	16	12.6%
Happy:	26	20.5%
Neutral:	16	12.6%
Sad;	19	15.0%
Surprised:	15	11.8%
Total:	127	100%

Table 4.1 – Training Data Statistics.

Something to notice from table 4.1 is that a far greater percent of females were present than males. As mentioned before, female subjects have a particular effect on emotion expression. Females in general will modify the shape of their eye brows and wear makeup, changing the way a particular expression is conveyed. Also looking at the number of images per emotion, a few interesting deductions can be made. First of all, the happy expression performed the best out of all the emotions. In this table we can see that the happy expression was the largest percent of the entire training set. This could support

the claim that increasing the training set even by a slight amount, increases the performance by a considerable factor.

The neural network was trained using the back-propagation algorithm. This algorithm depends on knowing the correct output for every input that is presented to the neural network. Typically the neural network will be initialized with random weights before training is initiated. When the training starts, a new set of weights is presented to the neural network to be identified. The neural network will present some output. The first number of iterations of the training algorithm the output will be incorrect. The back-propagation will take the output presented and compare it to the expected output that was provided to it. This is usually referred to as the target output. The algorithm then determines the difference between the (actual – target) and use this data to modify the weights to minimize the difference to be zero. Starting at the output layer, this approach is back propagated to the hidden layer and then to the input layer. There is a whole mathematical proof of how back-propagation works. This is however beyond the scope of this thesis. At the fundamental level back-propagation can be summarized as it is briefly described above. The goal is to keep on applying the back-propagation rule until the weights have been modified to the point where the difference between actual and target is either zero, or very close to that.

Although the neural network's weights are adapted by back-propagation, another higher order training algorithm needs to maintain the rate at which the weights are adapted and what the termination criteria is going to be. For the training of this neural network, MSE (mean squared error) was selected as the termination criteria. The way to determine when to terminate the training process is by reaching a certain termination criteria. In this case, the termination criterion is going to be performance. Performance will be measured by how close

$$MSE = \frac{1}{M} \sum_{i=1}^{M} \left[TARGET - ACTUAL \right]^2$$

Equation 4.1 – Mean Squared Error Equation.

the network output is in comparison to the target output. In other words, when the difference between desired and actual output becomes very small the training should terminate. The equation that is going to be used to calculate this will be equation 4.1. The goal value that was set for MSE was zero. Given the training sample size and the complexity of the problem, MSE = 0 was not expected to be reached. But if the network is learning successfully the MSE value should at least become very close to that, on the magnitude of $1e^{-8}$. If the termination criterion is never reached, a backup termination criterion is also provided. In this case it is the number of epochs that the back-

propagation algorithm executes. From trail and error the number of epochs to execute was set to 30,000. Figure 4.5 below show the MSE of the neural network as it trained for 30,000 epochs.

The other parameters used in the training of the neural network are summarized in table 4.2. The transfer function used for each neuron is the tan-sigmoid function and it can be seen in figure 4.4. The tan-sigmoid transfer function performs particularly well in pattern recognition applications. The TRAINSCG and LEARNGDM are all training functions that control how rapid and to what extent the neural network weights are modified.

Training Parameter	Setting	Description
Training Function:	TRAINSCG	Scaled Conjugate Gradient
Adaptation Function:	LEARNGDM	Gradient descent with momentum weight/bias learning
Performance Function:	MSE	Mean Squared Error
Transfer Function:	TANSIG	Tan-Sigmoid Transfer Function

Table 4.2 – Neural Network Training Parameters.

Scaled Conjugate Gradient

It is often difficult to decide which particular learning algorithm to use with which class of problems. While some learning algorithms are particularly well suited for function fitting, others are good for pattern recognition. The Scaled Conjugate Gradient (SCG) training algorithm was

Acronym	
LM	Levenberg-Marquardt
BFG	BFGS Quasi-Newton
RP	Resilient Backpropagation
SCG	Scaled Conjugate Gradient
CGB	Conjugate Gradient with Powell/Beale Restarts
CGF	Fletcher-Powell Conjugate Gradient
CGP	Polak-Ribiere Conjugate Gradient
OSS	One Step Secant
GDX	Variable Learning Rate Backpropagation

Table 4.3 – Summary of Training Function.

selected as the training algorithm for the ERS neural network. This selection was based on the results obtained from a study which conducted extensive comparisons made between a large set of training algorithms (table 4.3). The comparison results were presented in the MATLAB Neural Network Toolbox documentation [9]. The study presented compares various training functions to both pattern recognition problems and function fitting problems. The comparison was made to determine which training algorithm performed the fastest. Three differently structured feed-forward neural networks were trained on three separate pattern recognition problems. The three problems used for performance measurements, the structures of the neural networks and the error goals are all presented in table 4.4. These three separate tests were executed and their test results are summarized in table 4.5. In all three tests, the RP algorithm performed best. The overall top performance is achieved by RP and SCG algorithms. This study further claims that RP performs worse than SCG when error is reduced. Also SCG tends to perform better with larger networks. Given that the network structure selected for the ERS is large, SCG was selected as the training algorithm for the neural network implemented in the ERS. The other training parameters such

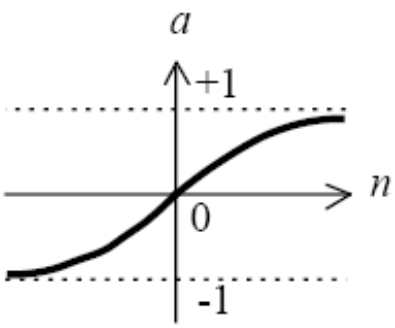
Problem	Structure	Error Goal
PARITY	3-10-10-1	0.001
CANCER	9-5-5-2	0.012
DIABETES	8-15-15-2	0.05

Table 4.4 - Summary test configurations.

as the adaptation function, performance function and transfer function were left at their default settings associated with the SCG training function in the MATLAB environment.

ALGORITHM	MEAN TIME	TEST
RP	3.73	PARITY
SCG	4.09	PARITY
CGP	5.13	PARITY
CGB	80.27	CANCER
RP	83.41	CANCER
SCG	86.41	CANCER
RP	323.9	DIABETES
SCG	390.53	DIABETES
CGB	394.67	DIABETES

Table 4.5 – Test Results Summary.



$$a = tansig(n)$$

Figure 4.4 – Tan-Sigmoid Transfer Function

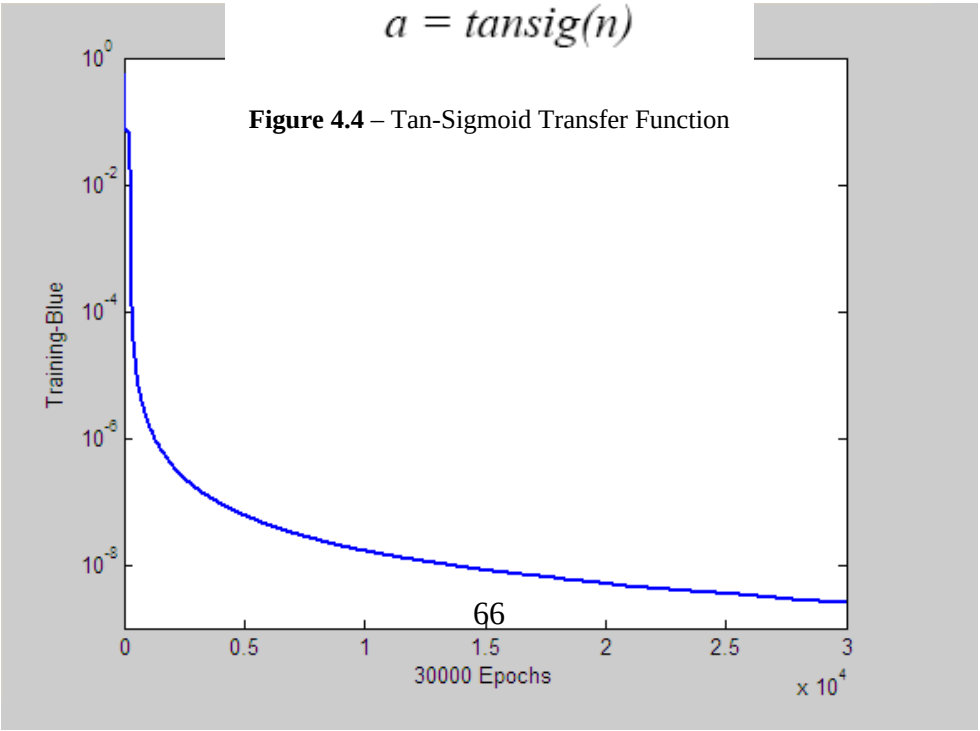


Figure 4.5 – Neural Network Training Performance.

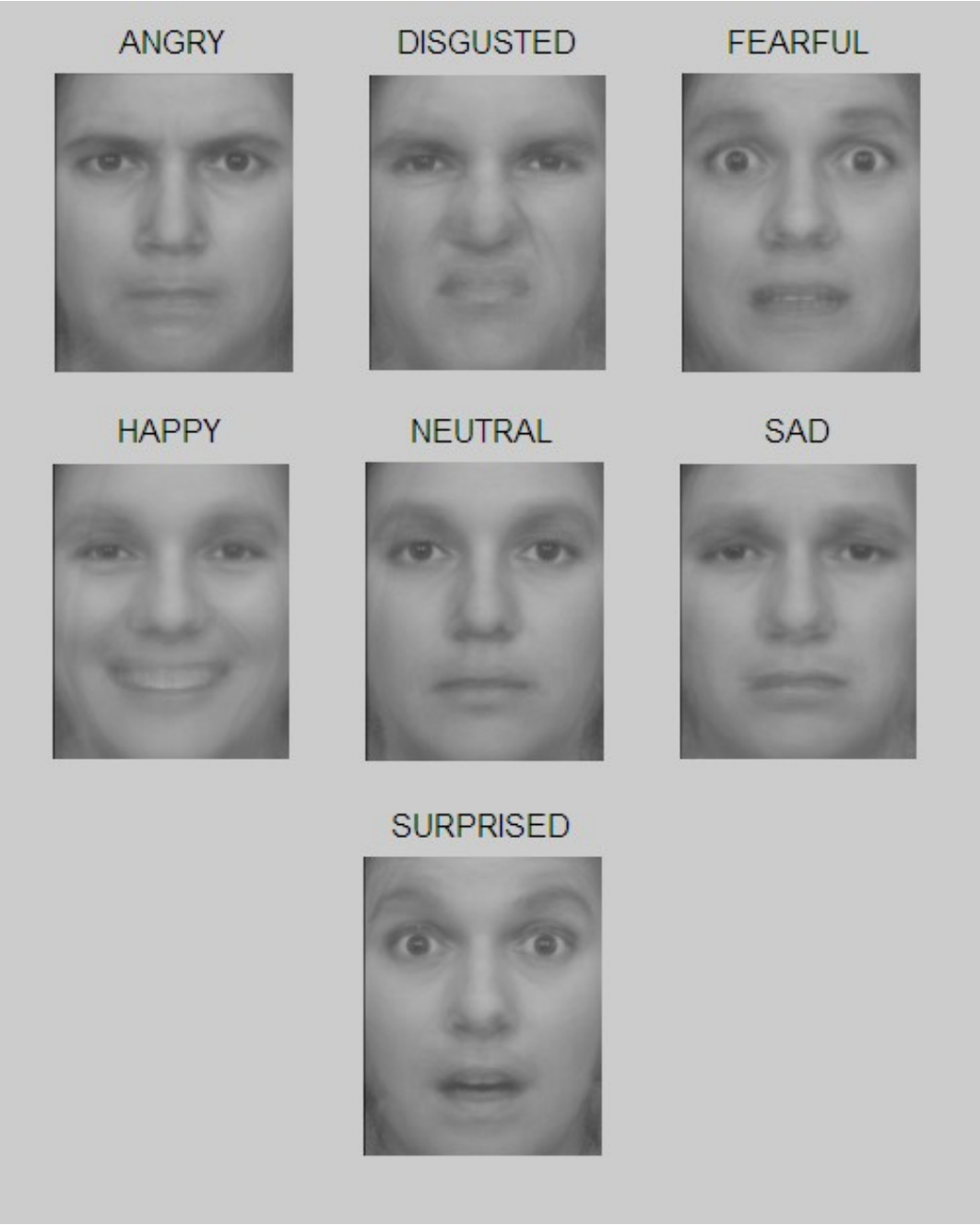


Figure 4.6 – Seven Average Emotions.

Chapter 5 - Emotion Recognition System

Section 5.1 – ERS Overview

The emotion recognition system was designed to recognize all seven emotions: Anger, Disgust, Fear, Happiness, Neutrality, Sadness and Surprise. The system was implemented in MATLAB in order to focus more on system operation principles rather than implementing various mathematical procedures. The system is designed to load a face with an unknown facial expression and output a vector of size 7x1. Each of the vectors seven values corresponds to a particular emotion in the order listed above. The values associated with each emotion are the probabilities that a particular emotion is detected in the input image. The output values are normalized to range from 0% to 100%. The output of a perfect match to a particular emotion will be a 100% for one of the seven emotions and 0% elsewhere. Typical outputs consist of 100% or a very high percent for one of the seven emotions and lower values for the other six emotions. The emotion detection process is as simple as loading in an image with an unknown expression and inspecting the output for the highest probability. The emotion with the highest probability suggests that the unknown input face is expressing that particular expression.

Section 5.2 – ERS Structure

The system consists of two main blocks which are the PCA block and the ANN block. The PCA block takes an image as input and outputs the weights of the image. The weights are obtained by projecting the input image onto the eigenfaces which results in 127 weights. There are 127 weights because that is the size of M, or the number of images in the training set used to define the face-space. This number can be reduced to leave only the main principle components, but it has been selected to use all of the principle components as this would result in the most accuracy.

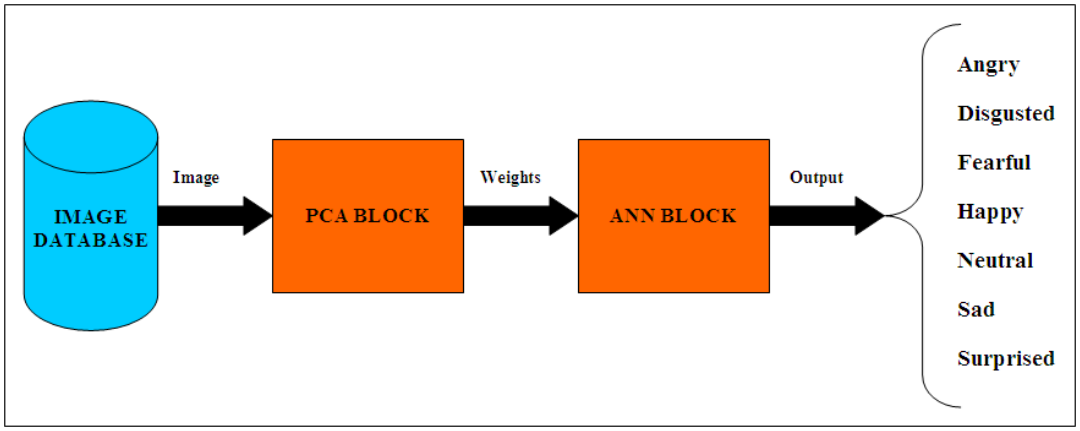


Figure 5.1 – ERS Structure Diagram.

These 127 weights essentially describe where a face image expressing a particular emotion resides in the face-space. In other words, the face-space is defined by a set of training images including all emotion

expressions. The entire N^2 space can be seen as the superset defined by all of the images of NxN in dimension. The face-space is a kind of subset defining the region within the superset where images containing faces would be located. This subset or face-space is then further divided into even smaller sets where specific emotion expressions reside.

Having projected the input image onto the face-space resulting in a set of weights, the weights are passed onto the second block. The second block is the ANN block and it is responsible for the actual recognition of the expression from the weights provided. The ANN is trained on a set of weights to associate a particular weight type with a particular emotion output. What the ANN is essentially taught to do is learn where specific emotion expressions are located in the face-space from the weights that are generated from the input image being projected onto the eigenfaces. It learns to associate a particular expression location within the face-space with a particular emotion. After training is complete, the ANN is provided new unseen sets of weights and the neural network must generalize to provide an answer. The neural network must interpret the new set of values and attempt to find the best match based on previous weights it had seen during the training. Once an output is produced, the output is displayed as the probability that the input image is likely to be one of seven emotions.

In the ideal case, the output would be a 100% for one of the emotions and 0% for everything else. This is rarely the case. Usually the output will report that each emotion is detected with some low probability. The goal is to have a large difference between the actual emotion expression probability and that of others. As will be mentioned later, it is often the case where presence of one or more emotions is strongly detected in an image. There are cases where the face within the image will look as though it is displaying more than one particular emotion. In such a case, it is possible to get high probabilities on the output for more than one emotion expression. Sometimes the emotion assigned to an image is overtaken by some other similar emotion present in the facial expression.

If this system was redesigned for real-time execution, another block would need to be added which would be responsible for preprocessing of images before they are fed into the PCA block. This block would be responsible for such tasks as obtaining an image and making sure it is the correct size and that it is an image of a face. It would also be responsible for making sure that the face region captured is consistent and similar to the images used to train the neural network.

Section 5.3 – ERS User Interface

The ERS interface was designed in MATLAB and requires that the user manually point to the image that is going to function as the input image. The user must also set a number of Boolean flags which control how the program is executed. There are two methods in which the program can be executed. The first method is used when the program is executed for the very first time. This will instruct the ERS to load all the images in the training set and perform all the principle component analysis computation tasks such as finding the covariance matrix, the eigenvectors and corresponding eigenvalues. It also calculates the eigenfaces and some other necessary data structures. These calculations only need to be performed once during the first runtime. These data structures will be saved to a file after they have been computed for later reuse.

The second method of running the program bypasses the generation of all the data structures and instead depends on these structures to already be present on file. The program just loads the already generated data structures and continues execution from there. This method is much faster than the previous method that requires the generation and backup of all the data structures. The second method of ERS execution should be manually selected after the first execution has completed.

The output is presented in both graphical and text format. The graphical output consist of the original input image, images of seven average expressions obtained from the training set and a normalized [0,100] bar graph displayed for each emotion. The seven average emotion images are used to graphically portray each of the seven regions within the face-space. If each emotion region within the face-space is to be graphically visualized, it would resemble each of the seven images presented on the GUI. A GUI with a perfect 100% accurate detection is presented below in figure 5.2. Figure 5.3 and figure 5.4 present the ERS GUI with mixed result detection and a failed detection respectfully.

The text output consists of the normalized output values represented by the bar graphs. It is difficult to sometimes tell what the specific probability of a certain emotion is graphically. The text output can be used to obtain the exact values. This is particularly necessary in instances where the probability of more than one emotion is very high.

Figure 5.2 – ERS User Interface with 100% Accurate Detection.

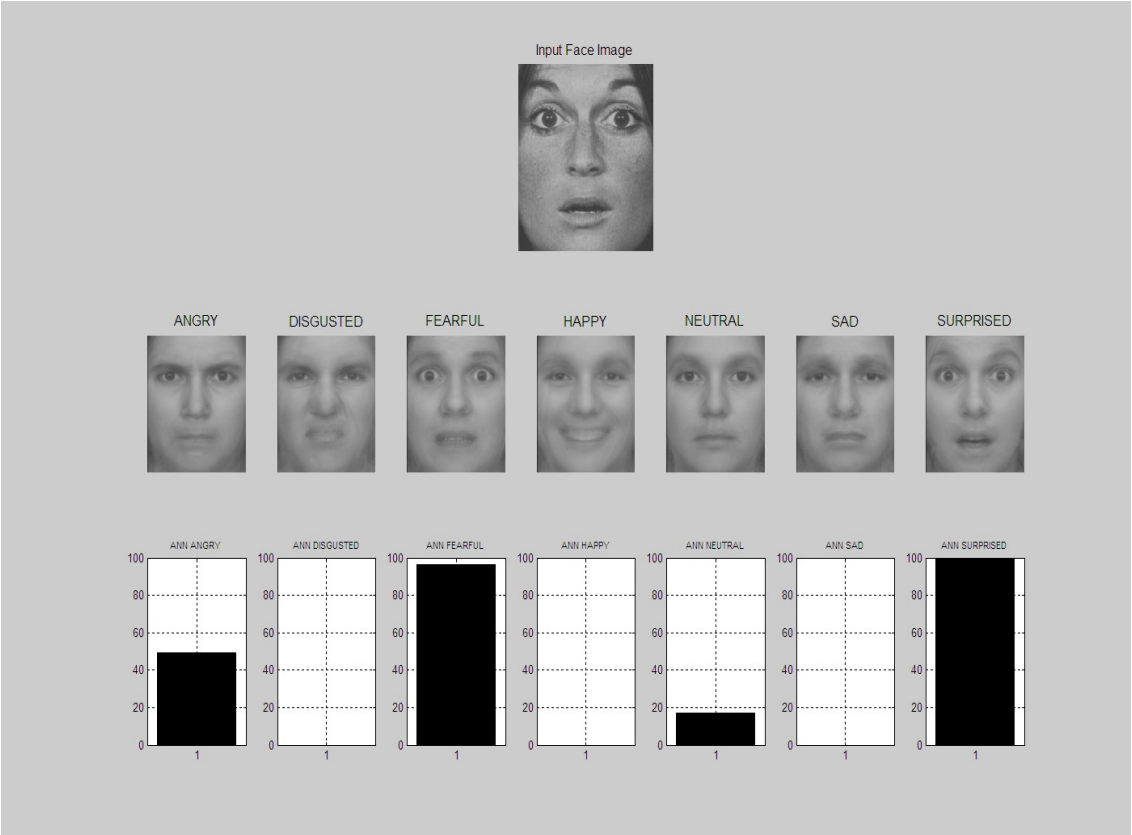


Figure 5.3 – ERS User Interface with Mixed Percentage Detection.

Figure 5.4 – ERS User Interface with Failed Detection.

Chapter 6 - ERS Performance Analysis

Section 6.1 – Introduction

After all components of ERS were completed, trained and all the data was collected, the systems overall performance was analyzed. A cross-validation (CV) set representative (but separate) of the training set of images was used to validate the systems performance. In cases of poor performance, the neural network was re-trained using different training algorithms and different neural network structures to obtain a desirable performance rate. After a few iterations of tuning and retraining the neural network, desirable results were obtained. The following sections describe the details of the test setup and the performance results obtained.

Section 6.2 – Test Configuration

The Emotion Recognition Systems performance was verified using a cross-validation test set of 52 images. These were selected to be representative of the training set. The cross-validation set contains both male and female subjects. No data from the training set were later used for cross-validation. The cross-validation set was visually inspected to make certain that all emotions were expressed in a manner similar to that of the training set. Images that did not meet this requirement were flagged as unacceptable and were not used. The CV set represents 40.94% of the training set. This is believed to be a

fair size to be representative of the training set. Because obtaining a large set of images for emotion recognition was difficult, each emotion could not be tested with the same amount of images. Table 6.1 displays the cross-validation statistics about the gender distribution of each emotion. There were more male images than female images in the CV set. It is important to analyze the gender distribution because there are factors about the way different genders express emotions that needs to be taken into consideration (discussed earlier).

	Male	Percent	Female	Percent
Angry:	4	57%	3	43%
Disgusted:	4	57%	3	43%
Fearful:	6	67%	3	33%
Happy:	7	64%	4	36%
Neutral:	4	57%	3	43%
Sad:	4	67%	2	33%
Surprised:	3	60%	2	40%
Total:	28		17	

Table 6.1 – Male and Female Representation of CV Set.

As with the training image set all the images in the cross-validation set were manually inspected and classified by their emotions. One emotion was assigned to each image in the CV set. The performance was analyzed through a manual inspection of input image and compared to the probabilistic output of the ERS.

Each image was represented by a weights vector consisting of 127 weights. All 127 weights were used as the input into the neural network. The output of the ERS system consisted of seven outputs, one per emotion, arranged in the same top to bottom order as in table 6.1.

Section 6.3 – Test Results

The following data presented in table 6.2(a) – 6.2(b) are the results of the cross-validation image set executed on ERS. The image ID field is the name of the image file that was used. A total of 52 images are presented in this table. The Gender field is used to compare what effect gender had on the performance of ERS. As mentioned before, there are differences in how a male and female expression might be perceived by ERS. The Emotion field indicates what emotion was assigned to each image. This was achieved by visually inspecting each image and assigning an emotion to it that best represented it. The Status field indicates how well the assigned image emotion matches the ERS output. A *PASS* indicates that the ERS output matched the emotion which was assigned to the image. A *FAIL* in the status field indicates that the image failed to be properly recognized by the ERS. Some images that passed are labeled as *PASS*, while others are labeled either *P-NEUT*, *P-SURP* or *P-FEAR*. These labels indicate that although the exact emotion was not recognized by the ERS, the image presented and the emotion recognized by the ERS was very close, and thus is considered a successful detection.

Image ID	Gender	Emotion	Status
1	F	Angry	PASS
2	F	Disgusted	PASS
3	F	Happy	PASS
4	F	Neutral	PASS
5	F	Sad	P-NEUT
6	F	Surprised	P-NEUT
7	F	Angry	F-Disg.
8	F	Disgusted	PASS
9	F	Fearful	PASS
10	F	Happy	PASS
11	F	Neutral	PASS
12	F	Sad	F-Ne-Fe
13	F	Surprised	PASS
14	M	Angry	F-Fear
15	M	Disgusted	P-NEUT
16	M	Fearful	PASS
17	M	Happy	PASS
18	M	Happy	P-NEUT
19	M	Neutral	PASS
20	M	Sad	P-NEUT
21	M	Surprised	P-FEAR
22	M	Angry	PASS
23	M	Disgusted	F-HAP
24	M	Fearful	P-SURP
25	M	Happy	PASS
26	M	Happy	PASS

Table 6.2 (a) – ERS Data Results Part 1.

Image ID	Gender	Emoton	Status
27	M	Sad	FAIL
28	M	Neutral	PASS
29	M	Surprised	PASS
30	M	Fearful	P-SURP
31	M	Sad	P-NEUT
32	M	Neutral	PASS
33	M	Surprised	PASS
34	M	Angry	PASS
35	M	Disgusted	PASS
36	M	Fearful	P-NEUT
37	M	Happy	PASS
38	F	Fearful	P-SURP
39	F	Disgusted	PASS
40	F	Fearful	F-SAD
41	F	Angry	PASS
42	F	Happy	PASS
43	F	Happy	PASS
44	F	Neutral	FAIL
45	M	Angry	PASS
46	M	Disgusted	PASS
47	M	Fearful	PASS
48	M	Happy	PASS
49	M	Happy	PASS
50	M	Sad	FAIL
51	M	Neutral	FAIL
52	M	Fearful	PASS

Table 6.2 (b) – ERS Data Results Part 2.

The reason why instances where fearful were misrecognized for surprised and surprised for fearful is because these two facial expressions are very difficult to distinguish. Even visually inspecting the images, it would be difficult to select which one expresses fear and which one surprise. The two images in figure 6.1 below show how

Figure 6.1 – Comparison of Fearful and Surprised Expressions.

similar a surprised expression and a fearful one appear.

The two images above look very similar. To the left is the fearful expression while to the right is the surprised expression. Both have widened eyes and parted lips. The fearful expression seems to depict a

more extreme widening of the eyes than the surprised expression. Also the eyebrows of the surprised expression seem to be risen higher than that of the fearful expression. In this particular comparison the fearful expression is also exposing clinched teeth, while the surprised one is not. However, the main features of the face that would be used to compare the two expressions are very similar. Given that the training is done with 127 training images containing both male and female subjects, it becomes very difficult to distinguish such fine detail. For this reason fearful detections and surprised detections are treated the same (interchangeably). Even with these very indistinct differences between the two expressions, the system was able to correctly identify 9 out of 13 surprised and fearful expressions with only one fearful expression being identified incorrectly as sad.

Another interesting observation that should be mentioned is that although only one expression was assigned per image, it is often the case that the image portrays more than one emotion expression. It often becomes difficult to determine which expression is the dominate one. The image displayed in figure 6.2

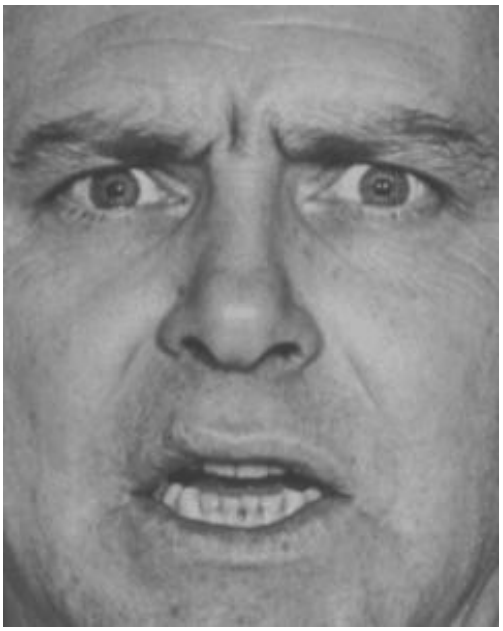


Figure 6.2 – Example of Mixed Expressions.

is an example of this. This image is of an angry expression. It was falsely identified as fearful. The ERS produced the following output results for this image: 100% Fearful, 97% Angry, 77% Neutral, 73% Sad and 22% Surprised. This is an interesting image to try and characterize visually. It can be labeled as angry, but at the same time it could equally be labeled fearful. Looking at the eyes of the individual in figure 6.2 there seems to be a hint of fear depicted in them. Perhaps the approach of assigning only a single emotion per image is insufficient for characterizing emotion expressions. Perhaps each image can be labeled with a set of suitable emotion expressions that best define it. The purpose of the ERS would be to indicate which emotional expressions are the most dominate, rather than trying to assign one single emotion expression per image.

One current issue with the ERS at this point is its ability to recognize the neutral emotion. Although its positive recognition rates are 71.4%, its false positive rates were too high at about 15.4%. It was inconclusive as to what exactly caused the probability of neutrality to be presented as the most probable amongst other emotions. For this reason, the neutral emotion is ignored. It is believed that with a larger

	PASS	Total	Percent
Angry:	5	7	71.4%
Disgusted:	6	7	85.7%
Fearful:	8	9	88.9%
Happy:	11	11	100.0%
Neutral:	5	7	71.4%
Sad;	3	6	50.0%
Surprised:	5	5	100.0%
Total:	38	45	84.44%

Table 6.3 – ERS System Performance Summary.

training set and perhaps more neurons in the neural network, this issue can be resolved. However given the lack of training data these further investigations cannot be carried out. Having more neurons in the neural network trained on the same dataset has not resolved the issue. Therefore increasing the training dataset along with the number of neurons at the same time might be the solution.

The ERS system performed well particularly in cases where it was able to distinguish between fearful and surprised expressions. Table 6.3 summarizes the emotion recognition system's performance. Table 6.4 provides a comparison between the number of images used for training versus the number of images used for cross-validation and the performance obtained. The happy expression was recognized best by the ERS with a recognition rate of 100%. Fearful and surprised expressions performed second best, however there were cases where fearful was confused with surprised and surprised with fearful. The ERS performance is followed by the disgusted expression and then by the angry expression. It would be naturally expected that fearful and surprised expressions would be easiest to learn for the neural network and recognized, as they

	Used For Training	Used For CV	Success Rate
Angry:	19	7	71.4%
Disgusted:	16	7	85.7%
Fearful:	16	9	88.9%
Happy:	26	11	100.0%
Neutral:	16	7	71.4%
Sad:	19	6	50.0%
Surprised:	15	5	100.0%
Total:	127	52	

Table 6.4 – Training Data Used vs. Success Rate.

express the most change in the face. After this, it would be expected that angry would be the next expression to be most easily recognized, however this isn't the case. Perhaps the reason why happiness is so easy to recognize for the ERS is because in the training set this emotion is expressed very similarly as opposed to other emotions. That is, the expression for happy, varies very little from individual to individual, while other expressions vary to a degree between individuals. This would lead to the images being plotted in a very dense region within the face-space. All the weights that would be used as the input into the neural network would be very similar. This would result in a set of weights representing the happy expression which would be very easy for the neural network to learn as they would be very similar.

A possible reason why the neutral expression was difficult to learn for the neural network might be because a neutral expression lacks any dramatic changes in the face. This leaves a set of different faces in appearance with no particular strong pattern to distinguish neutral expressions from other expressions. The pattern that would be detected for each neutral face would be the features of the face itself. These patterns would vary and would not provide data that could be used to identify the presence of neutrality. This would explain why the neutral emotion was so difficult to detect. It would also explain why

there were so many false positives. This was due to the neural networks treating the identity of an individual as the data pattern that indicated the presence of neutrality. The neural network after being trained would detect a similar face and conclude that it was seeing a neutral face. If the neural network learned that the other emotion expressions were very probable, it would be able to reject this false positive detection. However due to lack of training samples it was not able to learn such behavior.

Section 6.4 – Comparison to Other Research

This section will introduce a different approach to the emotion recognition problem. The approach used for comparison will be based on Discrete Cosine Transform (DCT) used for image compression and neural networks for image identification. The DCT approach has been selected for comparison purposes because it is most similar to the approach of this thesis. The performance of this approach will be compared to the performance the ERS presented in this thesis. The cons and pros of implementing each approach will be evaluated.

6.4.1 – DCT and Neural Network Based Approach

The study by Kharat and Dudul [15] presents two approaches to emotion recognition. The first approach is based on implementing a

Multilayer Perceptron Neural Network (MLP), while the second approach is based on implementing a Generalized Feed-Forward Neural Network. The input into both of the networks is a vector consisting of features extracted by Discrete Cosine Transform (DCT) and physical parameters. The physical parameters are obtained from the image and are energy, entropy, variance, standard deviation, contrast, homogeneity and correlation. The combined data of DCT and physical parameters creates a 71x1 input vector for the neural network. The input vector is composed of 64 parameters obtained from DCT and 7 physical parameters.

The Neural Networks are trained to recognize all seven universally recognized emotion expressions. The database selected for training and cross-validation is the well known Japanese Female Emotion Expression database. This database consists of 219 images. There are 10 individuals in this database each expressing all seven of the emotions, with 3 to 4 examples of each emotion. The division of the image database that is established for training and cross-validation is 90% dedicated towards training and 10% dedicated towards cross-validation.

The following describes the particular details of each selected approach and the results that were obtained. The following parameters were determined to function best for each of the neural networks. The

following parameters for each neural network were derived through extensive experimentation.

Neural Network Configuration

MLP Setup:

- **Number of Inputs:** 1
- **Number of Hidden Layers:** 1
- **Number of Processing Elements in Hidden Layer:** 10
- **Transfer Function:** tanh
- **Training Rule:** Momentum
- **Number of Epochs:** 5000

GFFNN Setup:

- **Number of Inputs:** 1
- **Number of Hidden Layers:** 1
- **Number of Processing Elements in Hidden Layer:** 19
- **Transfer Function:** tanh
- **Training Rule:** Momentum
- **Number of Epochs:** 5000

Neural Network Performance Analysis

MLP Results:

Output/ Desired	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Angry	**3**	0	0	0	0	0	0
Disgust	0	**4**	0	0	0	0	0
Fear	0	0	**3**	0	0	0	0
Happy	0	0	0	**1**	0	0	0
Neutral	0	0	0	0	**5**	0	0
Sad	0	0	0	0	0	**2**	0
Surprise	0	0	0	0	0	0	**3**

Table 6.5 – Confusion Matrix for MLP NN.

GFFNN Results:

Output/ Desired	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Angry	**5**	0	0	0	0	0	0
Disgust	0	**4**	0	0	0	0	0
Fear	0	0	**3**	0	0	0	0
Happy	0	0	0	**4**	0	0	0
Neutral	0	0	0	0	**3**	0	0
Sad	0	0	0	0	0	**1**	0
Surprise	0	0	0	0	0	0	**1**

Table 6.6 – Confusion Matrix for GFF NN.

Conclusions

The test results suggest that both the neural networks performed flawlessly on the cross-validation set. In both cases the size of the cross-validation set was

Neural Network Model	MSE		% Classification Accuracy		Time elapsed per epoch per exemplar	N/P
	Train	CV	Train	CV		
MLP	0.0028	0.0358	99.47	100	0.1117 mSec.	0.2371
GFFNN	0.0048	0.0831	97.42	100	0.0496 mSec.	0.1253

Table 6.7 – Summary of Results for Both Neural Networks.

21 images representing 7 emotions. All of these images were correctly recognized by both the MLP and the GFFNN, 100% of the time. The confusion matrix for the MLP can be found in table 6.5, while the confusion matrix for the GFFNN is seen in table 6.6. Table 6.7 summarizes the results for both of the neural networks for both the training set and the cross-validation set. Better results were obtained on the cross-validation set than on the training set. Since both neural networks performed with 100% accuracy on the cross-validation set, the final evaluation was based on the performance on training set. In this case, the MLP performed better than the GFFNN network. The MLP performed with 99.47% accuracy, while the GFFNN performed with 97.42% accuracy. The final conclusion which is made is that the MLP performs better than the GFFNN by a small margin.

6.4.2 – Result Comparison and Discussion

The performance of the Emotion Recognition System (ERS) presented in this thesis can be summarized with a confusion matrix which is presented in table 6.8. The results of the ERS will be compared to the results presented in the above study in a number of ways. The overall system performance on both the training set and the cross-validation set of both systems will be compared. Since both approaches are designed for real-time applications, the ability to apply a particular

approach in real-time will be assessed. Given that the training data and data used for cross-validation plays an important role in overall system performance, the two databases used for system performance evaluations will be analyzed.

First of all, the actual results of all the confusion tables can be compared. The performance of the ERS on the cross-validation (CV) set is 84.44%. This performance was obtained with the exceptions that were defined in this chapter. This performance is worse than that of either the MLP or the GFFNN which both executed at 100% on the CV set. The performance of the ERS on the training set is 100%, which is better than the 99.47% and 97.42% that were obtained for the MLP and GFFNN networks respectfully. Comparing just the performances of each approach, the MLP and GFFNN approach performs better than the ERS approach.

Inspecting both approaches from the real-time application perspective, the following must be taken into consideration: the computational effort required to achieve the emotion identification, the size and complexity of the code. The MLP and GFFNN approach requires that for the detection of every emotion expression, the following parameters be calculated: DCT, energy, entropy, variance, standard deviation, contrast, homogeneity and correlation. Since the input into the neural networks consists of these parameters, after the

neural networks have been trained, they continue to require all these parameters for each detection instance. Calculating all these parameters for each image detection in real-time becomes a costly affair. In comparison, the ERS requires that a somewhat costly set of calculations be made only once during the training process. During the training process the covariance matrix needs to be calculated, the eigenvectors and eigenvalues must be found and the eigenfaces must be generated. Once the eigenfaces are generated, they are saved and are reused. The next time that detection is required, the ERS system must simply project the new image onto the eigenfaces resulting in a compressed representation of the image. This compressed representation is then used as input into the neural network. This entire process requires a small number of multiplications as compared to all the parameters that must be found for the other presented approach. From a real-time application perspective, the ERS system would be more suitable. Also as mentioned in the first chapter of this thesis, the aim of using PCA is to be able to combine the face identification approach with the emotion recognition approach and hopefully in the future with the gender identification approach to create a three in one system. For this reason, the ERS approach is preferable.

Lastly the two databases used for neural network training and cross-validation can be compared. The database used to train the MLP and GFFNN consists of 219 images of only females of the Japanese nationality. Each emotion is expressed multiple times with slight variations. The cross-validation set consisted of 10% or 21 images. In comparison the database used for training and validating the ERS consisted of 127 (training) and 52 (cross-validation) images consisting of both males and females. The database also had a mixture of white subjects and Asian decent subjects. In some instances a particular emotion expression was represented with more than one image, however for the most part, one image was presented per every emotion for each subject. The database used for evaluating the ERS is more realistic since it is less constrained. Also, unless each of the ten individuals was selected to be the cross-validation set one at a time, only one individual would have been used for validation. The cross-validation set used for the ERS was a mixture of 7 unique individuals. It can therefore be concluded that the dataset used for training and validating the ERS was more realistic even though it was smaller in size.

PREDICTED \ ACTUAL	Angry	Disgusted	Fearful	Happy	Neutral	Sad	Surprised
Angry:	5 (71.4%)	0	0	0	1 (14.3%)	0	0
Disgusted:	1 (14.3%)	6 (85.7)	0	0	0	0	0
Fearful:	1 (14.3%)	0	8 (88.9%)	0	1 (14.3%)	2 (33.3%)	0
Happy:	0	1 (14.3%)	0	11 (100%)	0	0	0
Neutral:	0	0	0	0	5 (71.4)	1 (16.7%)	0
Sad:	0	0	1 (11.1%)	0	0	3 (50.0%)	0
Surprised:	0	0	0	0	0	0	5 (100%)
Total:	7	7	9	11	7	6	5

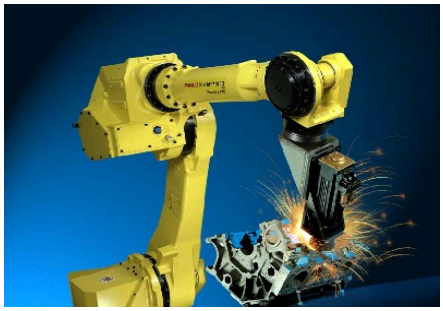
Table 6.8 – Cross-Validation Results Confusion Matrix.

Although the MLP and GFFNN approach has demonstrated better performance on the cross-validation set, it has weaknesses in other areas that make it less suitable for real-time applications than the ERS. A better performance comparison would be made if both of these approaches used the same database.

Chapter 7 - Emotion Robotics

Section 7.1 – Emotion Based Robotics

Popular applications for robots have primarily been in manufacturing, hazardous waste removal and various military applications. These applications require that the robot do exactly what it is indented to do. In manufacturing the robot is usually given exact and precise steps that it must take to complete certain desired tasks. If the robot fails to follow these predefined steps, serious problems can arise. The ability to follow exact steps becomes even more important when the robot is utilized in military or



(A) – Industrial Welding Robot Arm.

(b) – Military MQ-1B Predator UAV.

Figure 7.1 – Popular Robot Applications.

hazardous waste removal applications. In military applications a robot that is assigned the task of searching and destroying a particular target must be very precise. Failure to follow precise steps in these situations will result in innocent human deaths. The same can be concluded about robots who fail to perform hazardous waste removal tasks as instructed. All of these applications require that the robot perform exactly as it is instructed to do.

Robots designed for realistic interaction with humans typically should not function in a predicable manner as the robots described above. Humanoid robots designed to interact with humans generally are designed to mimic human appearance and behavior. Robots which can exhibit both human appearance and behavior are able to achieve realistic human-robot interaction. Humans are emotional creatures that exhibit emotion and are also able to understand the emotion of others.

Figure 7.2 – David Hanson's Humanoid Robot.

Emotion and emotional expressions play a vital role in human communication. It is therefore the goal of humanoid robot designers that robots exhibit and interpret emotions. Robots capable of performing such tasks typically do not function in a predictable step-

by-step fashion. Emotion based robots are able to react to various interpreted emotions in different ways based on their own emotional states. It is complex human-like robot behavior such as this that is the goal of many ongoing research projects today. A number of these research projects today have been able to achieve robot systems that are able to both interpret and express emotions. However there are still few robots today that are humanoid in appearance and have the abilities to both interpret and express emotions. Some of the well known emotion based robotics research projects are presented in the following sections.

Section 7.2 – Kismet Robot

Overview

Kismet is one of the more popular robots that interprets and expresses emotions found in United States. This robot was designed at Massachusetts Institute of technology by Cynthia Breazeal as her doctoral research project. Kismet is an autonomous sociable robot. The purpose of this project is to research human-robot communication. In many regards, Kismet is designed to be similar to an infant. The long term goal of this project is for Kismet to

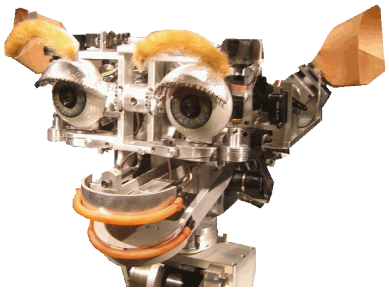


Figure 7.3 – MIT's Kismet Robot.

be able to learn from interacting with humans as an infant would. Kismet is able to modify its facial expressions, posture and voice in response to auditory and visual stimulations from the caregiver. Kismet's behavior can be altered by modifying the way in which the caregiver addresses him. The behavior achieved is similar to that of an infant. Yelling or showing signs of anger towards Kismet will make him react by lowering his head, ears and gazing down. This is one example of how Kismet expresses emotion in response to interpreting the emotion of the caregiver. The importance of this project is that the robot can interpret common human emotion expressions and respond in a manner that humans can understand. Projects such as this one are responsible for laying the groundwork in human-robot interaction.

System Block Overview

This fairly simple robot requires a considerable amount of processing power to function. Figure 7.4 is a simplified block diagram of how Kismet functions from an input-output perspective. The low-level feature extraction block is responsible for filtering out only input information that is useful for extracting behavioral information. The attention system functions by asserting the robots

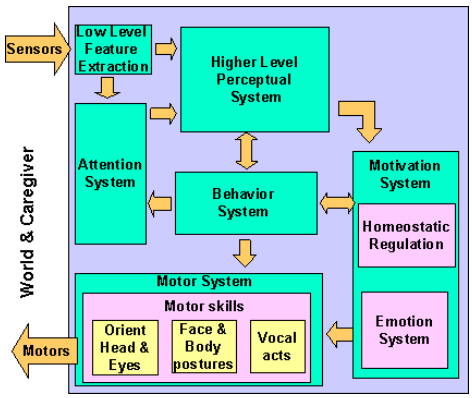


Figure 7.4 – Functionality Block Diagram.

focus on stimuli that is of importance at a particular time during the interaction. The higher-level perceptual system can be viewed as a wrapper block. It analyzes the low-level features obtained from the attention-system and determines whether certain states are met. It collects all the low-level stimuli and attempts to recognize high-level events. The motivation system is composed of two parts, the drives part and emotions part. These two blocks correspond to the homeostatic regulations block and emotion system block respectively. The drives are the various needs that Kismet might have. Comparing these drives or needs to that of an infant, they would compare to such needs as the need to be fed or the need to rest. In the same way Kismet has predefined needs. These are states that either deplete, or build-up with time. They are associated with various behaviors which restore the needs to appropriate levels. The behavior block is responsible for determining which emotion should be utilized to restore the needs of the robot. These selected emotions are then fed into the motor systems block, which then drives the motors and the auditory block to express a particular selected emotion. In such a way Kismet is always requiring some sort of attention and is satisfied through human interaction.

Hardware and Software

Kismet’s hardware and software is designed to perform both image processing and auditory processing, all in real-time. The high-level perception, motivation, behavior and motor skill systems are all controlled by four Motorola 68332 microprocessors executing L, which is multi-threaded lisp developed at MIT. All visual processing, the attention system and neck and eye control is achieved with nine 400MHz PCs networked together running QNX, a real-time operating system based on UNIX OS. Speech recognition is processed on a 500MHz PC running Linux OS. Speech analysis is performed on a dual 450 MHz PC running window NT.

Kismet’s vision is acquired using four color CCD cameras. Two wide view cameras are mounted in the center of kismet’s head. These are responsible for distance calculations and primary high-level tracking. The second pair of cameras is located in the eyes of Kismet and these are responsible for higher resolution processing task such as eye detection.

The robots auditory input is collected through a wireless microphone that is worn by the caregiver. Speech processing and recognition is achieved through software developed by the MIT’s Spoken Language Systems Group. All processing is performed on the two windows machines described above.

The robots ability to speak is achieved through a software package called DECtalk v4.5. This vocalization package is unique in that it is capable of simulating a human's articulacy tract. Various parameters can be adjusted to obtain different voices with different characteristics.

Kismet is able to express various emotions with a head design which has 15 Degrees of freedom. This allows Kismet to make such motions as moving the ears and moving each eyelid independently of each other. Kismet can also move the eyebrows in a number of different ways. The lips are designed to move in a way that allows Kismet to smile or frown. Finally the lower jaw can open and close the mouth of Kismet. Together all these movements allow Kismet to express a wide range of expressions.

Section 7.3 – WE-4RII Robot

Overview

The robot presented in this section is less commonly known. The robots name is WE-4RII and it stands for **W**aseda **E**ye No. **4** **R**efined **II**. This robot was constructed in Takanishi's laboratory at Waseda University in Japan. It is by far more advanced than the Kismet robot. The robots design consists of a head,

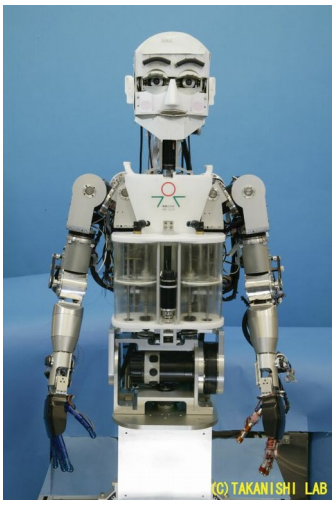


Figure 7.5 – WE-4RII Robot.

torso and arms as can be seen in figure 7.5. The design of this robot took into consideration the physical constraints of a human body. The robots ability to turn and reach is similar to that of a human. This humanoid robot was designed for the purpose of researching realistic communication between humans and robots. The robot is able to express seven emotions: anger, disgust, fear, happiness, neutrality, sadness and surprise. Unlike Kismet this robot uses its entire body to express emotions. This robot also possesses far more input senses than Kismet.

System Block Overview

The robot has an interesting approach to modeling various behaviors. Figure 7.6 shows a block diagram of the robots various functional units and how they interact. The selected approach is based on a needs model. In a way, this model is similar to that used with Kismet. The robot has certain needs which can be met by taking certain actions. The robots needs are divided into a number of categories. These categories are appetite, need for security and need for exploration. The appetite need results from tracking how active the robot is. If the robot is too active, it will run out of energy and will need to calm down to regain it. The need for security is demonstrated when the robot is exposed to a strong or dangerous stimuli for a long duration of time.

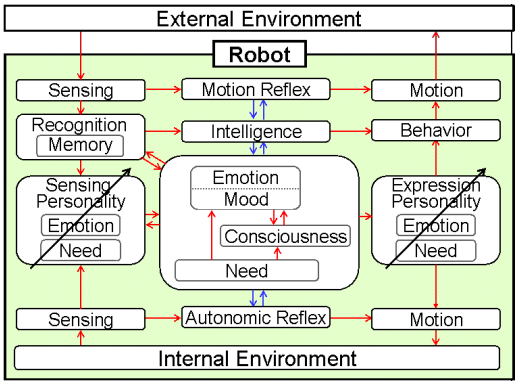


Figure 7.6 – Functional Block Diagram.

When the robot is exposed to a dangerous or strong stimuli for an extended period, it will exhibit a defensive behavior. The last need is the need for exploration. When the robot encounters a new object or environment, it will express an exploratory behavior. The robot will attempt to explore the new object or environment until its curiosity is satisfied. As with all of these needs, the robot will continue to exhibit a particular behavior until the robots needs are satisfied.

Hardware and Software

This robot has 59 degrees of freedom compared to Kismet's 15 degrees of freedom. The robot has a large collection of advanced sensory inputs. The sensory inputs are composed of visual, auditory, cutaneous (touch), and olfactory (smell) inputs. All these senses allow the robot to interact with its surroundings in a realistic manner.

The eyes of the robot consist of a specially designed eye unit. This integrates all of the eye components into a very small space allowing the head to contain other important components. The eye unit is designed with two eyelids that can blink in 0.3 seconds, which is at

the same rate as a human. The robots control of the eyeballs is designed to also be like that of a human. The eyes change directions at a rate which would be expected from those of a human

The robots neck, shoulders and arms are also designed to be like that of a human. This particular design allows the robot to have the ability to move the shoulders. This action is particularly useful for

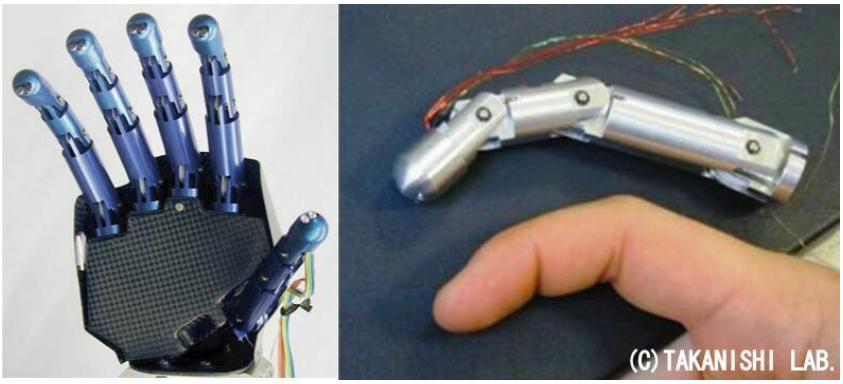


Figure 7.7 – The hands of the robot.

expression of emotions. All body movements are achieved with the same angular velocity as that of a human.

The hands of the robot are particularly advanced. The robot's hands were developed by a separate research team at the ARTS Lab in Italy. This particular design of the hands allows the hand to grasp objects of various shapes effectively. Just like a human hand, three separate segments of each finger firmly enclose an object providing reliable grasping functionality. The hands are full of sensory inputs as well. The hands have contact sensors, temperature sensors and force

sensors. This allows the robot to use its hands to explore its surroundings and gather data about its environment.

The robot expresses its emotions using its entire body and facial features. The robot also has the ability to change the color of its cheeks to pale or red. The eyebrows and mouth are especially flexible which allows various unique expressions to be conveyed on the face.

This particular robot design is unique in that the robot is fitted with a pair of artificial lungs. The lungs constantly pump air in and out. Not only does this create the sensation that the robot is alive and breathing, it also allows the robot to take samples of the air. The robot has a gas sensor which allows it to achieve a basic sense of smell. This robot can smell such gasses as alcohol, ammonia and cigarettes.

As this robot has a very large assortment of sensors and functionality, all of it will not be covered here. This sections primary purpose is to familiarize the reader with various ongoing projects in emotion based robotics. More details on this robot can be obtained at Takanishi’s homepage where the robot is described in great detail.

Chapter 8 - ERS Hardware and Software

Section 8.1 – Introduction

This chapter will describe a humanoid robot which was designed and built for such an application as Emotion Recognition. This chapter will also focus on real-time implementation of the emotion recognition system described in this thesis. The physical design and design considerations for real-time implementation will be explored. Possible

applications for such a robot will also be discussed. Finally, coding approaches for emotion recognition in real-time will be covered.

Section 8.2 – Design Overview

The humanoid robot nicknamed D.I.M, which stands for Digital Intelligence Machine, was constructed over duration of over a year to serve a number of purposes. The primary purpose of the robot is to serve as an example of where an emotion recognition system might be implemented.

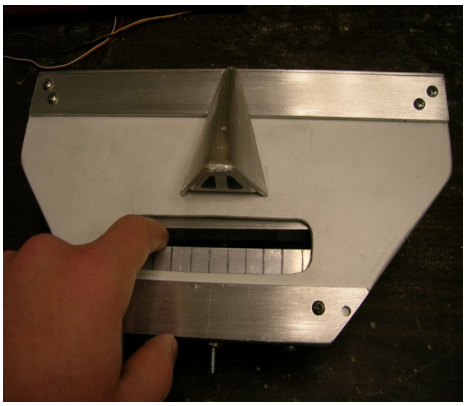


Figure 8.1 – DIM Head Design.

Building a life size humanoid robot gives valuable insight into what is required to construct a real-time system with many interconnecting components which are all required to function together flawlessly. Essentially the robot was constructed to give experience in constructing a platform where the emotion recognition system might be utilized. Secondly the robot was constructed to function as an advanced platform for various robotic research projects at Portland State University. This robot was designed to be very durable so that it has a long life span as a robot research platform. DIM will be PSU's first full-scale humanoid robot on which advanced research in the area of

artificial intelligence, voice recognition, computer vision and pattern recognition can be conducted. The robot may also be used as a foundation for research in feedback and control and various interdisciplinary projects between the mechanical engineering, computer science and electrical and computer engineering departments.

Also perhaps this robot platform will function as a steppingstone to more advanced robotics research at Portland State. The robot was designed with expandability in mind, allowing this robot to change the scope of its functionality by simply reprogramming some of its components. New components can also be easily added making this robot especially suitable for project reusability.

Section 8.3 – Design Specifications

Since this robot was designed with reusability in mind, durability was a factor which had high priority. This robot is designed to be as light as possible while being as durable and sturdy as possible. To achieve this, the robot was designed using aluminum for its properties of being light and strong. All connections are made with bolts some of which have locknuts on them. The main body of the robot is covered with light steel which is attached with rivets to the aluminum frame. Throughout

the design, great care was put into the physical appearance of the robot keeping it as humanoid in appearance as possible.

8.3.1 – Physical Body Design

The overall robot structure is comprised of a stationary torso with a moving head. The neck is made up of a stepper motor which turns the head left or right at a given pace. The head has a number of moving components. The head has a mouth with a moving top jaw which is controlled by a heavy duty servo. This was implemented for realistic human robot verbal interaction. On top of the robot head is a pair of USB cameras that serve as the robots eyes. The USB cameras are of average quality. Higher quality image capture may be obtained by either replacing these cameras with a more expensive, higher quality set of cameras, or by installing a separate new set of high quality cameras. Each eye has the ability to move left and right as well as up and down. The eyes can move in these directions independent of each other if it is so desired. This functionality is obtained by the use of pan-tilt kits which hold a pair of servo's together allowing each to move in a particular degree of freedom. Each eye is also equipped with a

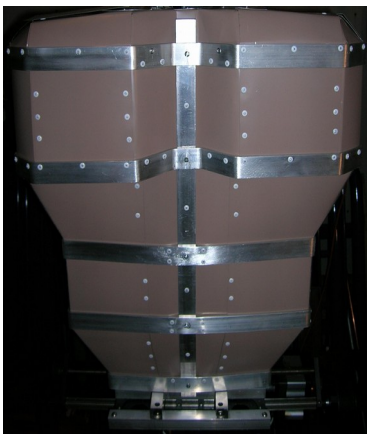


Figure 8.2 – DIM Body Design.

functional eyelid and eyebrow. The purpose for this is to give the robot a more human-like appearance. Also these components can be used to research behavioral robotics where emotion expression is an important factor. All of the robots hardware is located on the back of the robot. Some of the more vital hardware components are in a protective aluminum cage to prevent accidental damage to them.

8.3.2 – FPGA Development Board

The hardware consists of a number of modules that can be configured to suit various research needs. Most of the heavy processing will be executed on external systems either running in parallel or running on a single dedicated machine. However the robot itself is equipped with hardware to perform some of the desired computation onboard the robot, as opposed to externally. The robots hardware consists of an ALTERA CYCLONE DE2 FPGA development board. This board has input/output ports to handle a wide variety of input/output functionality ranging from USB to VGA. The board also comes with SDRAM, SRAM and Flash memory. It also has an array of toggle switches which are ideal for

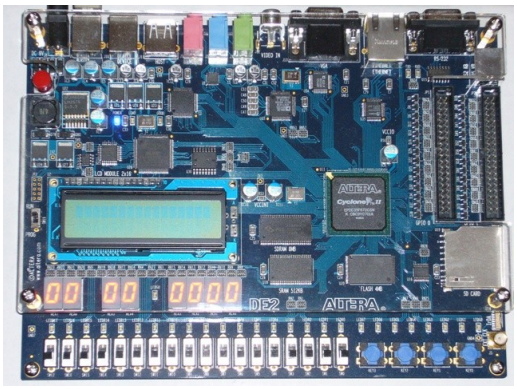


Figure 8.3 – Altera FPGA Development Board.

quick reconfigurations that might be necessary during run-time. This particular Cyclone FPGA is particularly useful because it can support two types of soft-core microcontrollers. The first microcontroller is a very simple 8-bit microcontroller and the other is a 32-bit microcontroller. These can be used to provide specialized onboard processing or function as co-processors for a specific task that needs to be optimized. In general the FPGA can be reconfigured to meet a wide range of needs making the FPGA development board the most important hardware component.

8.3.3 – ASC16 Servo Controller

Apart from the onboard FPGA development board, which is currently used to control the stepper motor, there is also an ASC16 servo controller board. This board has a small ARM processor and the board's sole purpose is to control all of the robots servos. This board can control up to 16 independent servos. These modules receive serial commands via RS-232. Multiple ASC16 boards can be chained together to operate in parallel. This board is currently configured to receive all of its

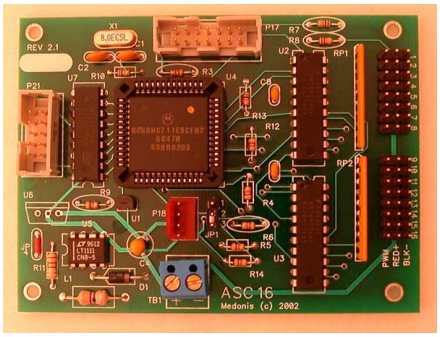


Figure 8.4 – AS16 Servo Controller Board.

commands from a host PC. A USB to serial converter is used to send commands to the board.

8.3.4 – Robots Power Supply

The robot is also equipped with two separate power supplies. One power supply is a PC type power supply producing power for the ASC16 and will also produce power for most future modules that might need to be added on. This power supply is 300W and produces +3.3V at 14A, +5V at 30A, +12V at 11A, +5V at 2A, -5V at 0.3A and -12V at 0.8A. The second power supply (COSEL R100U-24) is a heavy duty supply utilized by the stepper motor controller, producing +24V at 4.5A.

8.3.5 – Stepper Motor Controller

The 5-Phase UPS503 stepper motor controller is configured to drive the stepper motor in the neck of the robot. The controller receives the step clock and direction signals from the FPGA board. This signal is sent across a parallel 40-bit wide ribbon to an intermediate board that splits the signal and voltage level-shifts the output to meet the stepper motor controller input specifications. The stepper motor step clock is sent by the FPGA to the intermediate board and finally to the stepper motor controller, the entire time the FPGA board has power. The

direction signals are transmitted from a host PC via RS-232, same as with ASC16 board. The FPGA board has a simple module written in Verilog to handle the RS-232 transactions.

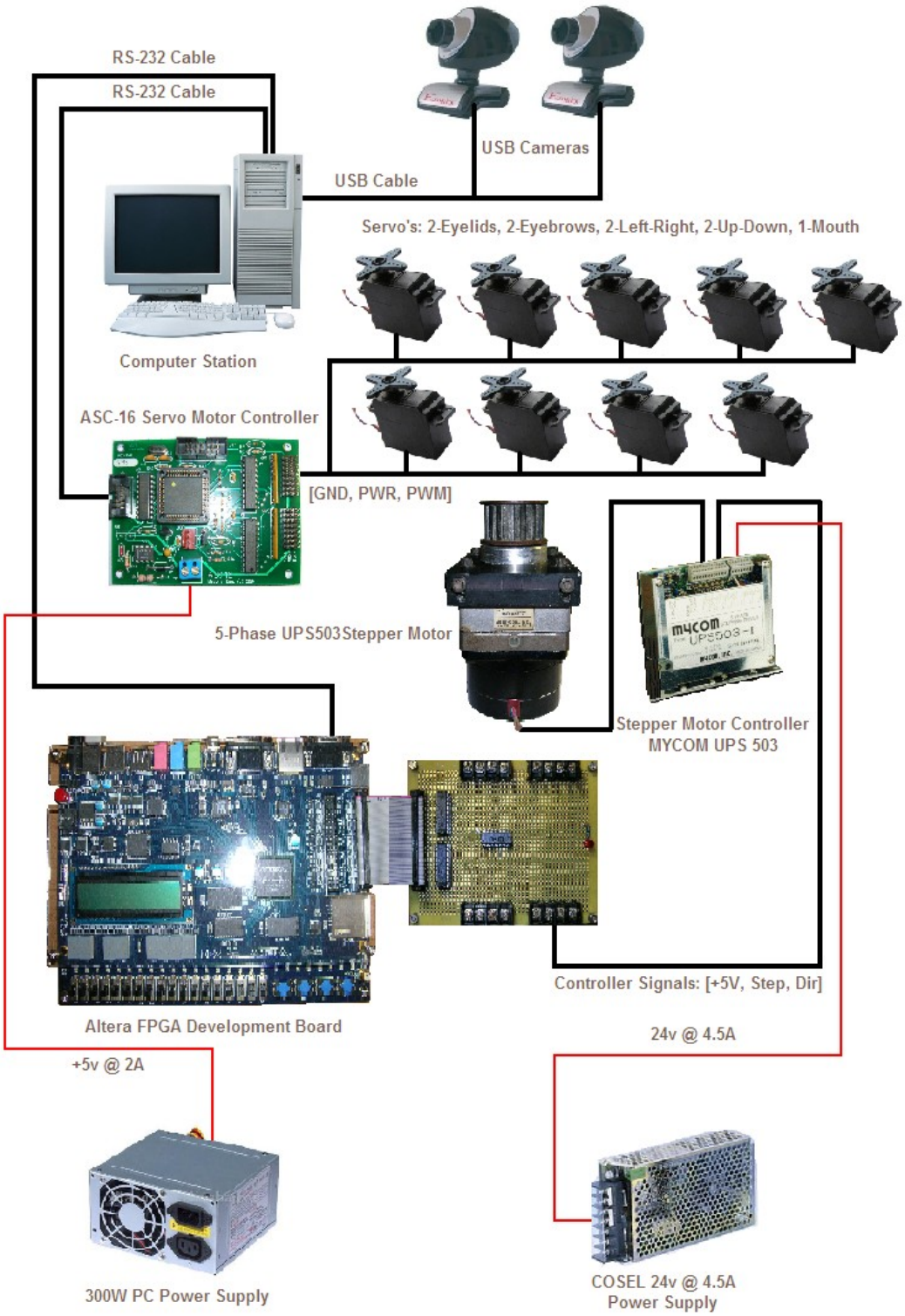


Figure 8.5 – High-Level DIM Schematic.

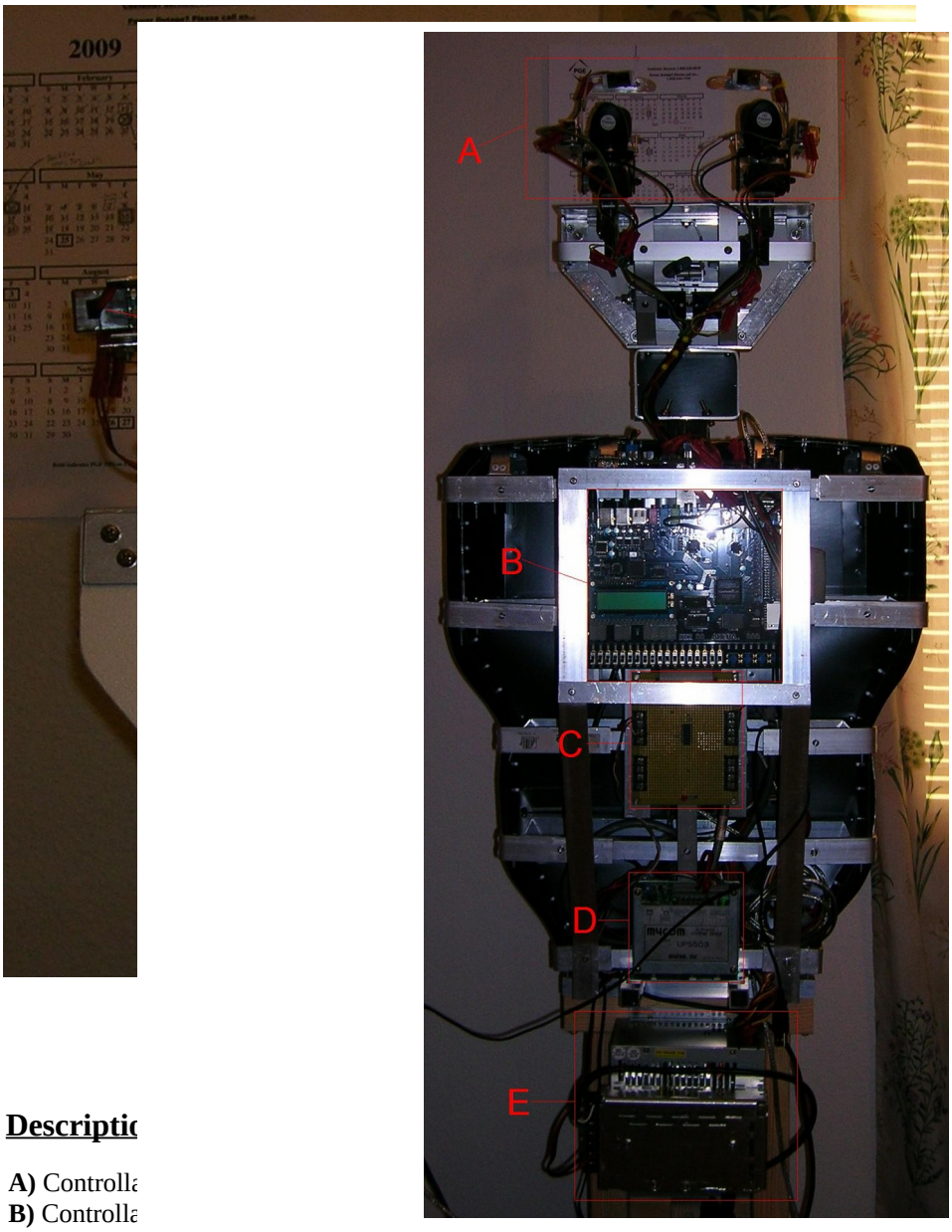


Figure 8.7 – Back View of Robots Hardware.

Description

A) USB Cameras Adapted to Function as Robot Eyes.
B) Altera FPGA Board Used to Drive Stepper Motor Controller.
C) FPGA Board to Stepper Motor Interface Card.
D) Stepper Motor Controller.
E) Power Supply System.

Figure 8.8 – Front and Side Views of Robot.

Section 8.4 – Current Functionality

Currently the robot has been programmed with the ability to use its cameras to detect a human face and track it. The tracking is done by adjusting the servo's attached to the pan-tilts described in the previous section. This operation is performed in real-time with a tracking rate sufficient to track the face of an individual moving their head at a rate that would be expected when conversing with a human being. The purpose of the tracking is not to track the face at incredible speeds, but rather to provide minor adjustments to the range of sight so that the human in front of the robot is not lost due to small changes in position. Face detection is achieved through a process developed by Paul Viola [27], which at a very fundamental level consists of template matching. This approach to face detection is very fast and is perhaps one of the better approaches currently available for real-time applications. All coding is implemented in C++ using the OpenCV graphics library developed by Intel. The OpenCV graphics library provides functionality for a majority of standard graphic operations.

Section 8.5 – Real-time Coding Considerations

The Emotion Recognition System has been developed in MATLAB as a proof of concept, rather than a model that can be implemented without change to produce real-time results. The images loaded by the system

for the purpose of emotion recognition were all hand cropped and produced in a very controlled environment. This is not an option if the ERS is to obtain input images in real-time. Factors such as head rotation, lighting and distance of the individual from the camera will all have an impact on the input image. All these factors will have a negative impact on principle component analysis producing accurate results. Since PCA is essentially a pattern recognition approach, the pattern of the input image must be similar to a particular predefined pattern. If the input image is modified by one of the factors mentioned above, the pattern extracted through the process of PCA is than modified and successful matching of image classes becomes very difficult.

Before live captured images can be passed into the emotion recognition system, they must be pre-processed. All images that are passed into ERS must have the same dimensions. They must also represent the same geometric region of the face. Even though an image might be the same size, it can represent different regions of the face. One image might be of the entire head, while the other might be a close up of the face. Capturing the correct geometric region and having the correct dimensions of the image are two of the more important issues associated with obtaining an image that can be used for emotion recognition.

Most of the issues described above can be alleviated by defining the face-space and training the neural network using a very large and diverse set of images. This set needs to have a wide variation of head rotation along with a wider range of facial expressions per emotion. There are a number of ways to represent each emotion, and this more complex emotion representation needs to be incorporated in the training set. By training the ERS on a more diverse training set, the ERS will learn to recognize a broader range of input images. This would simplify the preprocessing phase and put more of the burden on the ERS.

The process of obtaining the image itself can be achieved with the OpenCV face detection software. The face will be detected in real-time and can be easily extracted from the image by defining an ROI or region of interest. It may then be rescaled to meet the dimension requirements of the ERS, and histogram equalized to ensure proper image intensity distribution. The same approach used to detect a face image can also be used to detect facial features such as eyes and mouth. A simple distance vector can

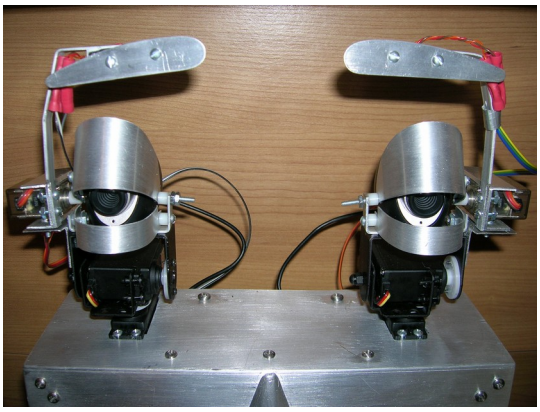


Figure 8.9 – DIM Eye Design.

be constructed measuring the distance between the eyes and the nose to properly crop the obtained image. Once the measurements between the eyes and the nose is obtained, a rectangle region of interest can be fitted over this point guaranteeing similar cropping amongst all images used as input for the ERS.

Once the images are preprocessed to meet the requirements of ERS, the true reason why this particular approach of using PCA and a neural network becomes evident. This system requires a large image training set for both defining the face-space and training the neural network. Also the neural network training and creating the input dataset requires a considerable time. However, once the data is collected and the training is complete, very little computational resources are necessary to take an input image and produce an output. The majority of the data required to compute the output can be loaded at the initialization phase of the ERS and just used. At initialization a considerable amount of data needs to be loaded. All the images must be loaded and reshaped to form vectors. The average needs to be calculated and subtracted from every image. The principle components must be computed, which is a relatively computationally expensive task. Each image must be projected onto the eigenfaces creating a set of weights. These weights must then be collected and used to train the neural network. All this must be done during the

training phase. Once the training is complete, all relevant data must be saved. The eigenfaces are saved because they define the face-space obtained from the image training set. This data will be necessary for future calculations. It does not need to be recalculated again and therefore may be saved and just reused. The average calculated over the image set of size M will also not need to be recalculated and can be saved and reused. Lastly after the neural network has been trained to a desirable precision, the neural network weights that are obtained from the training process are also saved. To recap, only the eigenfaces, average image and neural network weights need to be saved. This saved data is then loaded at the ERS initialization process and used to calculate the output of new input images. Not only can all this data that requires a lot of resources to compute be reused, but the way in which it is reused is also quit efficient.

When a new image is introduced to the ERS after preprocessing, the image must be reshaped into a vector. This requires looking through all the rows or columns of the image and concatenating them together to form a vector. This vector is then mean centered by having the average vector calculated earlier and restored from a file subtracted from it. Once the vector is mean centered, it must then be projected onto the eigen-space by being multiplied by a weighted sum of eigenfaces. This produces a weights vector, which is fed into the

neural network. The neural network uses its pre-calculated weights to determine the appropriate output. So essentially the ERS can produce an output presented any preprocessed input with a constant number of multiplications. This number of multiplications is a reasonable amount given the task that the ERS is expected to perform. Also given that most mainstream processors are considerably good at performing multiplications, the benefits of using this approach to emotion recognition become self-evident. Therefore after the training phase of the ERS, the actual code required to implement it is reduced. Also, a large amount of data can be reused making it a very efficient system particularly for embedded systems. Anther note to mention here is that unlike the recognition of faces, that constantly needs to redefine its face database, emotions don’t change. Once the set of given emotions are learned, they will never need to be redefined.

Chapter 9 - Conclusion

This thesis aimed at developing an emotion recognition system which had the potential of being implemented in real-time with minimum modifications. The purpose of such a system is to improve human-robot interaction. Giving a robot the ability to interpret emotions is the first step in achieving a realistic and meaningful communication with humans.

A system for emotion recognition was developed in the MATLAB environment. The emotion recognition system (ERS) was developed to detect the presence of all seven universally recognized facial

expressions: anger, disgust, fear, happiness, neutrality, sadness and surprise. This system was realized implementing principle component analysis which was used for data compression and a neural network used for pattern recognition. The emotion recognition system performed well (84.44% success rate) given the small set of data available for training. In comparison it also performed well against a similar emotion recognition system based on Discrete Cosine Transform and a neural network.

A humanoid robot platform was also developed as part of the research. This humanoid robot can be implemented for validating a real-time version of this system. Further, the humanoid robot has the potential of being the platform for many other robotics research projects.

The presented emotion recognition system can further be improved by using a larger training set which would have made the system less sensitive to variations of each expression. It would have also given the neural network more training data with which it could improve its performance. This particular approach to emotion recognition is also particularly useful for its ability to be implemented in real-time, requiring very little computational resources. The system can also easily be combined with the eigenface face recognition approach, resulting in a system capable of both recognizing emotions

and identifying people. The research work conducted for this thesis presents a viable approach to emotion recognition. Although this system still requires further improvements to be implemented in real-time, the overall functionality of the emotion recognition system is favorable.

References

[1] Y. Andreu, R. A. Mollineda ,"The Role of Face Parts in Gender Recognition", In: Campilho, A., Kamel, M.S. (eds.) ICIAR 2008. LNCS, vol. 5112, pp. 945-954. Springer, Heidelburg (2008).

[2] M. S. Bartlett, G. Littlewort, C. Lainscsek, I. Fasel and J. Movellan. (2004). "Machine learning methods for fully automatic recognition of facial expressions and facial actions.", In IEEE International Conference on Systems, Man & Cybernetics, The Hague, Netherlands, October 2004. p. 592-597

[3] G. Bradski. "Computer Vision Face Tracking For Use in a Perceptual User Interface", Intel Technology Journal, 2 (1998).

[4] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C. M. Lee, A. Kazemzadeh, S. Lee, U. Neumann, S. Narayanan, "Analysis of Emotion Recognition using Facial Expressions, Speech and Multimodal Information", Proceedings of ACM 6th International Conference on Multimodal Interfaces (ICMI 2004), ISBN: 1-58113-890-3, State College, PA, 2004.

[5] G. Caridakis, K. Kostas, S. Kollias. "User and context adaptive neural networks for emotion recognition", Neurocomputing 71 (2008) 2553-2562.

[6] M. Castrillon-Santana, O. Deniz-Suarez, L. Anton-Canalis, J. Lorenzo-Navarro, "Face and Facial Feature Detection Evaluation", Institute of Intelligent Systems and Numerical Applications in Engineering.

[7] M. Chaumont, B. Beaumesnil. "Robust and Real-Time 3D-Face Model Extraction", in IEEE International Conference on Image Processing, ICIP'2005, pp. 461-464, Sept. 2005.

[8] R. Cowie, E. Douglas-Cowi, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, J.G. Taylor. "Emotion Recognition in Human-Computer Interaction." IEEE Signal Processing Magazine, 2001.

[9] H**.** Demuth, M. Beale, Martin Hagan, "Neural Network Toolbox**.** User Guide.", v6 ed.U.S.A.: The. Math Works Inc., 2008, pp. 34-51.

[10] F. Dornaika, F. Davoine, "Simultaneous Facial Action Tracking and Expression Recognition in the Presence of Head Motion" International Journal of Computer Vision, Vol. 76, pp. 257-281, March 2008.

[11] P. Ekman, H. Oster, "Facial Expressions of Emotion", Annual Review of Psychology. 1979. 20, 527-554.

[12] S. S. Ge, H. A. Samani, Y. H. Janus, C. C. Hang, "Active Affective Facial Analysis For Human-Robot Interaction", Proceedings of the 17th IEEE International Symposium on Robot

and Human Interactive Communication, Technische Universitat Munchen, Munich, Germany, August 1-3, 2008.

[13] C. Huang, Y. Huang, “Facial Expression Recognition Using Model-Based Feature Extraction and Action Parameters Classification” Journal of visual communications and representations, vol. 8, no. 3, September, pp. 278-290, 1997.

[14] S. V. Ioannou, A. T. Raouzaiou, V. A. Tzouvaras, T. P. Mailis, K. C. Karpouzis, S. D. Kollias. “Emotion Recognition through Facial Expression Analysis based on a Neurofuzzy Network.” Neural Networks 18 (2005) 423-435.

[15] G. U. Kharat, S. V. Dudul. “Neural Network Classifier for Human Emotion Recognition from Facial Expressions Using Discrete Cosine Transform.” IEEE computer society, 2008.

[16] K. Kim, “Face Recognition using Principle Component Analysis”, DCS, University of Maryland, College Park, USA 2003.

[17] S. Kumano, K. Otsuka, J. Yamato, E. Maeda, Y. Sato, “Pose-Invariant Facial Expression Recognition Using Variable-Intensity Templates”, International Journal of Computer Vision, Vol. 83, No. 2, pp. 178-194, June 2009.

[18] J. J. Lien, T. Kanade, J. F. Cohn, C. Li, ”Detection Tracking, and Classification of Action Units in Facial Expression”, Journal of Robotics and Autonomous Systems, Vol. 31, No. 3, pp. 131-146, May, 2000.

[19] M. J. Lyons, J. Budynek, S. Akamatsu, ”Automatic Classification of Single Facial Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 12, December 1999.

[20] S. Park, D. Kim, ”Spontaneous Facial Expression Classification with Facial Motion Vectors”, Proceedings of the 8th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2008)(2008.9.17), 2008.

[21] K. Pearson, “On Lines and Planes of Closest Fit to Systems of Points in Space”, Philosophical Magazine, Vol. 2, pp. 559-572, 1901.

[22] M. Turk and A. Pentland, “Eigenfaces for Recognition”, Journal of Cognitive Neuroscience, vol. 3, no.1, pp. 71-86, 1991.

[23] D. PISSARENKO, “Eigenface-based facial recognition”, February 13, 2003.URL: http://openbio.sourceforge.net/resources/eigenfaces/eigenfaces-html/facesOptions.html. (URL Accessed on July 17, 2009).

[24] J. Ren, M. Rahman, N. Kehtarnavaz, and L. Estevez, ”Real-time head pose estimation on mobile platforms”, Proceedings of WMSCI Conference on Systemics, Cybernetics, and Informatics, Orlando, July 2009.

[25] L. Sirovich, M. Kirby. “Low-dimensional procedure for the characterization of human faces.”, Journal of the optical society of America, 4(3) 519-524, 1987.

[26] L. I. Smith. “A tutorial on principal components analysis”, February 2002. URL: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. (URL accessed on Jun 05, 2009).

[27] P. Viola, M. Jones, “Robust Real-Time Face Detection”, International Journal of Computer Vision 57(2), pp. 137-154, 2004.

Appendix A - ERS Code

The following is the **E**motion **R**ecognition **S**ystem code written in the MATLAB environment. Each code module is presented separately.

EigenFace.m

```
%Dimitriy Labunsky
%Portland State University
%06/27/2009

%Note: Some PCA Calculation code is based on code written by Santiago Serrano
```

```
%prep environment
clear all;
clc;


fprintf('+=========================================+');
fprintf('\n|    #####    ####### # #   # #   ##### |');
fprintf('\n|   #       #      #   # #   # #     |');
fprintf('\n|  ##### *** ####   ###### #      ####   |');
fprintf('\n|  #       #     #   # #   # #       |');
fprintf('\n| #####    #     #   #   # #  ######   |');
fprintf('\n+=========================================+');

fprintf('\n\t\tDimitriy Labunsky (C)2009\n');


%Determine whether to use stored data or generate new data.
%--Do not Generate Data, use old  = 0.
%--Generate Data and Save to File = 1.
RUN_LOAD_DATA = 0;

%Run only once if data matrices have not been generated
if RUN_LOAD_DATA
   load_data;    %Generate New Data.
end


%#########################################################################
%############################ Determine Emotion ############################
%######## Angry, Disgusted, Fearful, Happy, Neutral, Sad, Surprised ########
%#########################################################################


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load Cross Validation Image and Determine it's size.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1);

%TRAINING SET
%img_path = 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Database\TrainingSet\
Full Set\(1).jpg'; % 1 -> 127 @ 100% Rec.
%CV SET
img_path = 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Database\
CrossValidation\(14).jpg';     % 1 -> 29:53 CV Set

cv_image = imread(img_path);
[height width depth] = size(cv_image);
```

```
subplot(3,7,3:5);
imshow(cv_image);
title('Input Face Image','fontsize',12);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LOAD STORED IMAGE DATA FROM FILE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Load Full Image Set.
load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\FULL_SET.mat'
FULL_SET

%Load Stored Emotion Matrix of 7 Average Emotions from File.
load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\GAMMA.mat'
GAMMA

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display the Seven Average Emotions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Display the Seven Average Emotions
EMOTION = {'ANGRY' 'DISGUSTED' 'FEARFUL' 'HAPPY' 'NEUTRAL' 'SAD' 'SURPRISED'};

for i=1:size(GAMMA,2)
   subplot(3,7,i+7);
   E = reshape(uint8(GAMMA(:,i)), height, width, depth);
   imshow(E);
   title(EMOTION(i),'fontsize',12);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform PCA calculation on Full Image Set - Define Face Space.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M = 127;  %Matrix Set. (127 Images in Full Image Set)

u   = []; %Eigen Faces
PSI = []; %Average Image = PHI


[PSI, u] = run_pca(M, FULL_SET);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine Distance from Input Image to Each Emotion using Neural Network
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

net_output  = [];

net_output = get_dist(u, cv_image, PSI);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display Neural Net Emotion Prabability Output Graphs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ANN_EMOTION = {'ANN ANGRY' 'ANN DISGUSTED' 'ANN FEARFUL' 'ANN HAPPY' 'ANN
NEUTRAL' 'ANN SAD' 'ANN SURPRISED'};

%Display Emotion Bar Graphs for both Euc. and Mah. Distances.
for i=1:7
   subplot(3,7,i+14);
   bar([net_output(i)]);
   title(ANN_EMOTION(i),'fontsize',8);
   ylim([0 100]);
   grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display Neural Network Emotion Probability Output
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\n\n*****************************\n');
fprintf('Neural Network Output: \n');
fprintf('*****************************\n');

fprintf('\nAngry----: %f'  ,net_output(1));
fprintf('\nDisgusted: %f'  ,net_output(2));
fprintf('\nFearful--: %f'  ,net_output(3));
fprintf('\nHappy----: %f'  ,net_output(4));
fprintf('\nNeutral--: %f'  ,net_output(5));
fprintf('\nSad------: %f'  ,net_output(6));
fprintf('\nSurprised: %f\n',net_output(7));

fprintf('\n*****************************\n');


fprintf('\nDone!!!\n');

%---end
```

run_pca.m

```
function [PSI, u] = run_pca(M, GAMMA)
% Perform PCA Calculation.
% Input:
% M ----- Number of Images in Training Set.
% GAMMA - Raw Image Set Matrix.


%Generate new data or use pre-generated data.
% Generate New Data: 1
% User Saved Data; 0
GEN_NEW_DATA = 0;

if GEN_NEW_DATA

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
   % Generating Mean Normalize Matrix PHI
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

   fprintf('\n\nGenerating Mean Centered Image Matrix... ');

   PHI = []; %Mean Centered Image Matrix
   PSI = []; %Mean Image

   PSI = uint8(mean(GAMMA')');  %Calculate Mean Image

   %Generate Mean Centered Matrix of Images
   for i=1:M
      PHI = [PHI double(GAMMA(:,i)) - double(PSI)];
   end

   PHI = uint8(PHI);

   fprintf('Done!!!\n');

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
   % Calculate Eigen Vecors and Eigen Values
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
   %Vec = Eigen Vectors as columns.
   %Val = Eigen Values on the Diagonal.

   fprintf('\nGenerating Covariance Matrix and Finding Eigen Values/Vectors... ');

   L=[];

   A = double(PHI);  %Mean Normalize Matrix PHI
```

```
L = A'*A; %Find smaller (MxM) Covariance Matirx

Vec = [];
Val = [];

[Vec Val] = eig(L); % Calculate Eigen Vectors and Values.

fprintf('Done!!!\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Eliminating Evec's that have corresponding Evals=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

fprintf('\nEliminate eVectors with Corresponding Zero eValues... ');

eVec = [];
eVal = [];

%Eliminate EVectors with Corresponding EValues < 0.0001
for i=1:size(Val,2)
   if(Val(i, i) > 0.0001)
      eVec = [eVec Vec(:,i)]; % MxM
      eVal = [eVal Val(i,i)]; % 1XM
   end;
end

fprintf('Done!!!\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Sorting Eigen vectors by Eigen value order
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

fprintf('\nSorting Eigen vectors by Eigen value order... ');

values  = [];
vectors = [];

D = Val;
V = Vec;

dvec = diag(D);
NV = zeros(size(V));
[dvec, index_dv] = sort(dvec);

%-------flipud
%   X = 1 4     becomes  3 6
%       2 5              2 5
%       3 6              1 4
```

```
    index_dv = flipud(index_dv);

    for i = 1:size(D,1)
        ND(i,i) = D(index_dv(i), index_dv(i));
        NV(:,i) = V(:,index_dv(i));
    end;

    vectors = NV;
    lambda  = ND;

    fprintf('Done!!!\n');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    % Normalize Eigen Vectors
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    % V = [3;2] Length(V) = sqrt(3^2 + 2^2)

    %%for i=1:size(vectors,2)
    %%    vectors(:,i) = vectors(:,i) ./ norm(vectors(:,i));
    %%end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    % Generate Eigen Faces
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

    fprintf('\nGenerating Eigen Faces... ');

    u = []; %Eigenface

    for i=1:size(vectors,2)
        u =[ u (A*vectors(:,i))./sqrt(lambda(i,i))]; %Replaced A with GAMMA for Diff Efaces. (N^2 x 1)
    end

    fprintf('Done!!!\n');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    % Normalize Eigen Faces
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    % V = [3;2] Length(V) = sqrt(3^2 + 2^2)

     for i=1:size(u,2)
         Len = sqrt(sum(u(:,i).^2));
         u(:,i) = u(:,i) ./Len;
     end
```

```
   save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\PSI.mat' PSI
   save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\u.mat' u
   save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\A.mat' A
   save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\lambda.mat' lambda
   save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\L.mat' L
else
   fprintf('\nLoading Saved PCA Data...');

   load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\PSI.mat' PSI
   load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\u.mat' u
   load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\A.mat' A
   load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\lambda.mat' lambda
   load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\L.mat' L

end
```

get_dist.m

```
function [output] = get_dist(u, image, PSI)
% Calculate Probability of Input Image Emotion Using Neural Network.

% u      = [] - Eigen Faces
% image  = [] - Input Image
% PSI    = [] - Average Image


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reshape Image Into Vector and Mean Center Input Image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[height width depth] = size(image);

img = reshape(image, height*width*depth, 1);
img = double(img) - double(PSI);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate Weights by Mapping Input Image to Face Space
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

input_vector = [];

for i=1:size(u,2)
  input_vector = [input_vector; dot(u(:,i)', img')];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% LOAD [127 x 300 x 7] NEURAL NETWORK & SIMULATE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

load 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\NN\Networks\MLP.mat';

output = sim(MLP_127_300_7, input_vector);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% FILTER OUT NEGATIVE VALUES AND NORMALIZE OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

[max_val index] = max(output);

for i=1:size(output)
   if(output(i) <= 0.01)
      output(i) = 0;
   end
   output(i) = (output(i) / max_val) * 100;
end
```

load_data.m

```
function load_data()
%Load All the Data either from Matrix or Database.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE FULL IMAGE SET & EMOTION MATIRX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\nLOADING DATA---START:\n');

   FULL_SET = [];  %full image set
   GAMMA    = [];  %7 emotion averages

   path = 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Database\TrainingSet\';

   FULL_SIZE      = 127;
   ANGER_SIZE     = 19;
   DISGUSTED_SIZE = 16;
   FEARFUL_SIZE   = 17;
   HAPPY_SIZE     = 26;
   NEUTRAL_SIZE   = 16;
```

```
SAD_SIZE      = 19;
SURPRISED_SIZE = 14;

FILE_EXT = '.jpg';

%-----------------------------------------------

folder = 'Full Set\';
        FULL_SET   = []; %%Full Set

        location = strcat(path, folder);

        fprintf('\nFULL SET:');
        for i=1:FULL_SIZE
   file = strcat('(', int2str(i));
   file = strcat(file, ')');
   file = strcat(file, FILE_EXT);
   filename = strcat(location, file);
   fprintf('\n\t\tLoading: %s ', file);
   image = imread(filename);
   [height width depth] = size(image);
   X = reshape(image, height*width*depth, 1);
   FULL_SET = [FULL_SET X];
        end

%-----------------------------------------------

        folder = 'Angry\';
        ANGRY      = []; %%Angry Images
        M_ANGRY    = []; %%Mean Angry

        location = strcat(path, folder);

        fprintf('\nANGRY:');
        for i=1:ANGER_SIZE
   file = strcat('(', int2str(i));
   file = strcat(file, ')');
   file = strcat(file, FILE_EXT);
   filename = strcat(location, file);
   fprintf('\n\t\tLoading: %s ', file);
   image = imread(filename);
   [height width depth] = size(image);
   X = reshape(image, height*width*depth, 1);
   ANGRY = [ANGRY X];
        end

        M_ANGRY = mean(ANGRY')';

        GAMMA = [GAMMA M_ANGRY];

        %-----------------------------------------------
```

```
        folder = 'Disgusted\';
        DISGUSTED   = [];
        M_DISGUSTED = [];

        location = strcat(path, folder);

        fprintf('\nDISGUSTED:');
        for i=1:DISGUSTED_SIZE
    file = strcat('(', int2str(i));
    file = strcat(file, ')');
    file = strcat(file, FILE_EXT);
    filename = strcat(location, file);
    fprintf('\n\t\tLoading: %s ', file);
    eval('image = imread(filename);');
    [height width depth] = size(image);
    X = reshape(image, height*width*depth, 1);
    DISGUSTED = [DISGUSTED X];
        end

        M_DISGUSTED = mean(DISGUSTED')';
GAMMA = [GAMMA M_DISGUSTED];

        %-----------------------------------------------

        folder = 'Fearful\';
        FEARFUL     = [];
        M_FEARFUL   = [];

        location = strcat(path, folder);

        fprintf('\nFEARFUL:');
        for i=1:FEARFUL_SIZE
    file = strcat('(', int2str(i));
    file = strcat(file, ')');
    file = strcat(file, FILE_EXT);
    filename = strcat(location, file);
    fprintf('\n\t\tLoading: %s ', file);
    eval('image = imread(filename);');
    [height width depth] = size(image);
    X = reshape(image, height*width*depth, 1);
    FEARFUL = [FEARFUL X];
        end

        M_FEARFUL = mean(FEARFUL')';

        GAMMA = [GAMMA M_FEARFUL];

        %-----------------------------------------------

        folder = 'Happy\';
        HAPPY       = [];
        M_HAPPY     = [];
```

```
    location = strcat(path, folder);

    fprintf('\nHAPPY:');
    for i=1:HAPPY_SIZE
file = strcat('(', int2str(i));
file = strcat(file, ')');
file = strcat(file, FILE_EXT);
filename = strcat(location, file);
fprintf('\n\t\tLoading: %s ', file);
eval('image = imread(filename);');
[height width depth] = size(image);
X = reshape(image, height*width*depth, 1);
HAPPY = [HAPPY X];
    end

    M_HAPPY = mean(HAPPY')';

    GAMMA = [GAMMA M_HAPPY];

    %-----------------------------------------------

    folder = 'Neutral\';
    NEUTRAL     = [];
    M_NEUTRAL   = [];

    location = strcat(path, folder);

    fprintf('\nNEUTRAL:');
    for i=1:NEUTRAL_SIZE
file = strcat('(', int2str(i));
file = strcat(file, ')');
file = strcat(file, FILE_EXT);
filename = strcat(location, file);
fprintf('\n\t\tLoading: %s ', file);
eval('image = imread(filename);');
[height width depth] = size(image);
X = reshape(image, height*width*depth, 1);
NEUTRAL = [NEUTRAL X];
    end

    M_NEUTRAL = mean(NEUTRAL')';

    GAMMA = [GAMMA M_NEUTRAL];

    %-----------------------------------------------

    folder = 'Sad\';
    SAD         = [];
    M_SAD       = [];

    location = strcat(path, folder);
```

```
        fprintf('\nSAD:');
        for i=1:SAD_SIZE
    file = strcat('(', int2str(i));
    file = strcat(file, ')');
    file = strcat(file, FILE_EXT);
    filename = strcat(location, file);
    fprintf('\n\t\tLoading: %s ', file);
    eval('image = imread(filename);');
    [height width depth] = size(image);
    X = reshape(image, height*width*depth, 1);
    SAD = [SAD X];
        end

        M_SAD = mean(SAD')';

        GAMMA = [GAMMA M_SAD];

        %-----------------------------------------------

        folder = 'Surprised\';
        SURPRISED   = [];
        M_SURPRISED = [];

        location = strcat(path, folder);

        fprintf('\nSURPRISED:');
        for i=1:SURPRISED_SIZE
    file = strcat('(', int2str(i));
    file = strcat(file, ')');
    file = strcat(file, FILE_EXT);
    filename = strcat(location, file);
    fprintf('\n\t\tLoading: %s ', file);
    eval('image = imread(filename);');
    [height width depth] = size(image);
    X = reshape(image, height*width*depth, 1);
    SURPRISED = [SURPRISED X];
        end

        M_SURPRISED = mean(SURPRISED')';

        GAMMA = [GAMMA M_SURPRISED];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STORE IMAGE DATA FOR FUTURE USE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Complete Image Set
```

```
save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\FULL_SET.mat'
FULL_SET

%Seven Averaged Emotions Set
save 'C:\Documents and Settings\Administrator\Desktop\Project DIM\ERS\Matrix\GAMMA.mat'
GAMMA


fprintf('\n\nLOADING DATA---END:\n');
```