الاستهلال

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

قال تعالى :

﴿ نَرْفَعُ دَرَجَاتٍ مَّن نَّشَاءُ ۗ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴾

سورة يوسف - الآية (٧٦)

DEDICATION

We are very grateful to Almighty Allah for helping us through this long journey, May He continues to bless, help and guide us to the right path.

We dedicate this project to all those that helped us toward this success, specially our parents, our families, our teachers and our colleagues.

And do not forget also those who departed from our world Of teachers,

***Dr Abdul Rasool Al-Zubaidi** , **Abdul Rahman Al-Taj** And our colleague*

Anas Fathi

Thank you all…

ACKNOWLEDGEMENTS

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this project. A special thanks to our final year project supervisor **Dr. Ala Eldin Abdullah Awouda,** who help us, stimulating suggestions and encouragement, help gratitude to coordinate our project and writing this report.

ABSTRACT

An application for face recognition and tracking system on video streams from surveillance cameras in public or commercial places is discussed in this thesis. Human surveillance through the fixed camera is useless and not suitable for security , to resolve this problem was suggested design face recognition and tracking system associated with database to track and monitor user face to improve security method and reduce security errors and facilitate monitoring. This research is divided to two parts: software implementation use matlab to capture the image from webcam and in hardware implementation webcam moves with the direction of the facial movement detected by servo motor. A successful simulation as well. The problem that is encountered is after start of tracking code after recognition, any face inside the video is tracked.

المستخلص

في هذه البحث تم تصميم نظام التعرف على الوجه وتتبعها على دفق الفيديو من كاميرات المراقبة في الأماكن العامة أو التجارية او غيرها. المراقبة البشرية من خلال الكاميرا الثابتة هي عديمة الفائدة وغير مناسبة للأمن، لحل هذه المشكلة تم اقتراح نظام التعرف على الوجه وتتبعها المرتبطة بقاعدة البيانات لتتبع ورصد وجه الناس ، لتحسين طريقة الأمن والحد من الأخطاء الأمنية وتسهيل الرصد. وينقسم هذا البحث إلى جزأين: في البرمجيات يتم استخدام برنامج الماتلاب لالتقاط الصورة من "كاميرا ويب" وفي تنفيذ العتاد تتحرك الكاميرا مع اتجاه حركة الوجه المكتشف بواسطة أجهزة سيرفو موتور . تم اجراء محاكاة ناجحة ، ولكن المشكلة التي تمت مواجهتها هي بعد بدء كود التتبع بعد التعرف على الوجه، يتم تتبع أي وجه داخل الفيديو.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

T_i	-	The training set of face images
Ψ	-	Average face of the set
Φi	-	difference with the average
λ_k	-	Vector
u_n	-	Vector
ω_k	-	weight of the kth
Ω^T	-	Weights form a vector
Θ	-	threshold

ABBREVIATIONS

AC	Alternating current
AREF	Analog reference
ATM	Automated teller machine
AVR	Automatic control to regulate the voltage
BV	Brightness Value
CNC	Computer Numerical Control
CV	Computer vision
DC	Direct current
DIP	Digital Image Processing
DN	Digital Number
FTC	Facial Treat Code
FTDI	Future Technology Devices International
GND	Ground
GPIO	General - Purpose Input/Output
GUI	Graphical User Interface
IC	Integrated Circuit
IDE	Integrated Development Environment
IP	Internet protocol
LBP	Local binary patterns
LCD	Liquid crystal display
LDA	Local Density Approximation
LED	Light Emitting Diode
LTE	Long Term Evolution
Matlab	Matrix Laboratory
OS	Operating system

PC	Personal computer
PCA	Principal Component Analysis
PM	Permanent magnet
PTZ	Pan -Tilt- Zoom Cameras
PWM	Pulse width modulation
RAM	Random access memory
Rx	Receiver
SD	Secure Digital
SMQT	Successive Mean Quantization Transform
SNOW	Sparse Network of Winnows
SVM	Support vector machines
Tx	Transmitter
USB	Universal Serial Bus
VR	Variable reluctance
Webcam	Web camera
Wi-Fi	Wireless fidelity

CHAPTER ONE
INTRODUCTION

Chapter One

Introduction

1.1 Preface

A Face recognition and tracking system is a Technique capable of identifying or verifying a person from a digital image or a video frame from a video source And follow it. One of the ways to do this is by comparing selected facial features from the image and a face database. It is typically used in security systems and can be compared to other biometrics such as finger print or eye iris recognition systems. Recently, it has also become popular as a commercial identification and marketing tool.

Some face recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. it focuses on the same identifier that humans use primarily to distinguish one person from another: their “faces”. One of its main goals is the understanding of the complex human visual system and the knowledge of how humans represent faces in order to discriminate different identities with high accuracy. Among the different biometric techniques, it does not require the cooperation of the test subject to work. Properly designed systems installed in airports, multiplexes, and other public places can identify individuals among the crowd, without passers-by even being aware of the system. Other biometrics like fingerprints, iris scans, and speech recognition cannot perform this kind of mass identification.[1].

1.2 Problem Statement

Human surveillance through the fixed camera is useless and not suitable for security, especially in crowded and public area and cannot distinguish all faces accurately**.**

1.3 Proposed Solution

Design face recognition and tracking system associated with database to track and monitor user face.

1.4 Aim and Objectives

The aim of this project is to design Face Recognition and Tracking system using matlab code and arduino.

The objectives are:

- To improve security method by using face recognition and tracking system and reduce security errors and facilitate monitoring.
- To design and implement the software and hardware of the proposed system.

1.5 Research scope

This Thesis focuses on recognition and tracking face of human in a continuous video transfer. This system used in security. It can be applied to Airports, borders, customs, elections, conferences, Automated teller machines "ATM" and others.

1.6 Thesis organization

Chapter Two is a theoretical background and related works in face recognition and tracking.

Chapter Three describes steps of hardware design the tracking process and view face recognition in a software platform.

Chapter Four discusses the results of simulation and implementation for the project.

Chapter Five explain the conclusion and the future ideas that can be performed.

CHAPTER TWO

LITERATURE REVIEW

Chapter Two
Literature Review

2.1 Overview

This Chapter is about different hardware and software that are used in this system. It also describes the algorithms that are used in recognizing and tracking the face, and contains a number of studies have been conducted on this technique in the past. In this section, a number of these studies are reviewed.

2.2 Related works

Baykara and Das have proposed a biometric system for human face detection and recognition is put into practice. This system works real time and successfully carries out face recognition, detection and tracking. In this way, the system has the feature of being integrated into different practices. Face detection and tracking system, is put into practice in a personal automation. Here, the aim is to carry out personnel’s entry and exit automatically and safely. In this system practiced basing on Principal Component Analysis "PCA", in the future studies, various optimization algorithms can be used in order to increase system performance [2].

Park and others have proposed a novel Coaxial-Concentric camera system that can capture and track high resolution face images at any distance in the range of 6 to 12 meters for face recognition. The Coaxial-Concentric camera configuration provides a large operating distance to track moving persons and recognize them with high accuracy. We have introduced a linear prediction model and a pan and tilt motion velocity control method for robust tracking. The face recognition results show the effectiveness of the proposed system for fully automatic subject tracking and identification at a distance of up to 12 meters. They plan to extend the operating distance beyond 12 meters by using either a high definition static camera or multiple Pan-Tilt-Zoom Cameras "PTZ" cameras to employ multistage zooming process [3].

Cai and others have proposed in which an off-line face detector, an online tracker and an online recognizer are efficiently combined. Boosting is applied as the classifier in detector and tracker, and the features are Local binary patterns "LBP" and Haar respectively. Considering the good performance of Canonical Correlation Analysis in pose-invariant face recognition, we incorporate it into our online recognizer, combined with an online classifier LASVM. The superior performance in challenging sequences proves the robustness of our framework [4].

Viraktamath and others have proposed Prototype system for automatic face detection and tracking. The test results show that the detection method used in the paper can accurately detect and trace human face in real time. in paper shows the intersection of Image processing and embedded systems, by using open CV " Computer vision" and Arduino real time implementation is possible. Future Work: Along with face detection, face recognition may also be implemented [5].

2.3 Digital Image Processing (DIP)

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

The influence and impact of digital images on modern society is tremendous, and image processing is now a critical component in science and technology. The rapid progress in computerized medical image reconstruction, and the associated developments in analysis methods and computer-aided diagnosis, has propelled medical imaging into one of the most important sub-fields in scientific imaging [6].

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output. The most common example is Adobe Photoshop. It is one of the widely used application for processing digital images.

A digital remotely sensed image is typically composed of picture elements (pixels) located at the intersection of each row i and column j in each K bands of imagery. Associated with each pixel is a number known as Digital Number (DN) or Brightness Value (BV), which depicts the average radiance of a relatively small area within a scene. A smaller number indicates low average radiance from the area and the high number is an indicator of high radiant properties of the area.

The field of digital image processing refers to processing digital images by means of a digital computer, the depiction of a digital image presented in figure 2-1. Once computer has visual information in appropriate format, computer can analyze it, which is called image analysis [7].

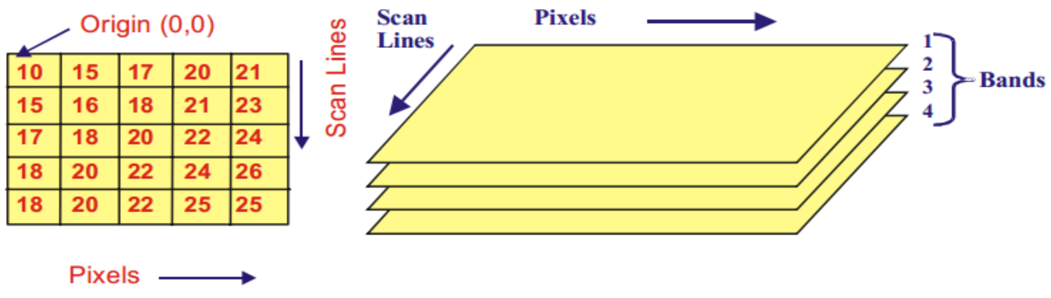


Figure 2-1: Depiction of a digital image.

Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means. In particular, digital image processing is the only practical technology for:

- Classification
- Feature extraction
- Multi-scale signal analysis
- recognition
- Projection

Some techniques which are used in digital image processing include:

- Anisotropic diffusion
- Hidden Markov models
- Image editing
- Image restoration
- Independent component analysis
- Linear filtering
- Neural networks

- Partial differential equations
- Pixilation
- Principal components analysis
- Self-organizing maps
- Wavelets

2.4 Face Recognition Algorithm

Face recognition is the process of identifying or verifying a person from a digital image or a video frame from a video source.

There are many algorithms used in face recognition such as:

- Local binary pattern
- Facial treat code
- Eigenface

2.4.1 Local Binary Pattern (LBP)

His relative new approach was introduced in 1996 by Ojala et al. [8]. With LBP it is possible to describe the texture and shape of a digital image. This is done by dividing an image into several small regions from which the features are extracted. These features consist of binary patterns that describe the surroundings of pixels in the regions. The obtained features from the regions are concatenated into a single feature histogram, which forms a representation of the image. Images can then be compared by measuring the similarity (distance) between their histograms. According to several studies [9], [10], face recognition using the LBP method provides very good results, both in terms of speed and discrimination performance. Because of the way the texture and shape of images is described, the method seems to be quite robust against face images with different facial expressions, different lightening conditions, image rotation and aging of persons [11].

2.4.2 Facial Treat Code (FTC)

Facial Trait Code is used to represent a face image. it was proposed by Lee et al. in 2008 [12]. The idea is to mark a local patch from face image to several classes, and use different coded local patches to represent a face. A local patch, so called facial trait, will be projected by PCA and LDA to reduce the dimensionality, and extract the feature for clustering. The features which are clustered together will be seen as same class.

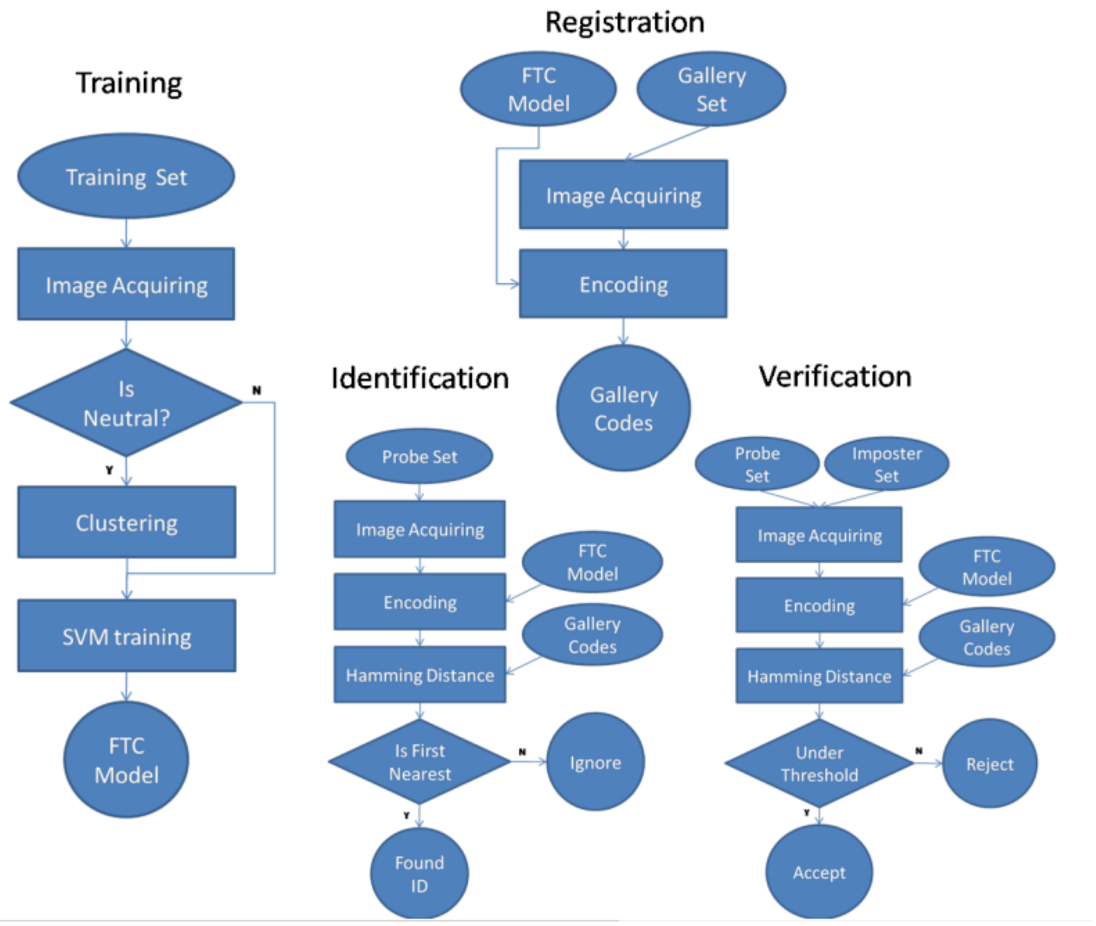


Figure 2-2: FTC working flow charts

The work flow was illustrated as figure 2-2 and the objects are defined as follow:

• Training Set: Containing neutral and non-neutral images.

• Gallery Set: Containing registration images.

• Probe Set: Testing images whose ids are in gallery set.

• Imposter Set: Testing images whose ids are not in gallery set.

• FTC Model: Containing SVM and clustering models.

Then the actions are defined as follow

• Image Acquiring: Reading images and scaling with bilinear filters, and then normalizing with XmeanYstd or Adaptive Histeq.

• Clustering: Using PCA and Local Density Approximation "LDA" for dimension reduction first, and then Clustering training ids for each defined patch. If the patches have not defined yet, we do.

clustering for all possible patches, and then find the best “discriminated” patches.

• Support vector machines "SVM" training: For each patch, using all training images as data and clusters as label to train with SVM.

• Encoding: For each patch, applying dimension reduction, and then predict the cluster the patch belongs to by SVM.

• Hamming D: Finding bit difference between two codes as the distance.

2.4.3 EigenFace

The recognition process is done by Eigen face algorithm. This approach was first developed by Sirovich and Kirby (1987) and used later by Matthew Turk and Alex Pentland [13].

A dataset, such as a digital image, consists of a large number of inter-related variables. Using Principal Component Analysis, the dimensionality of the dataset is reduced while retaining as much as variation in the dataset as possible. The datasets are transformed to a new set of uncorrelated variables called the principal components. These principal components are ordered in such a way that the first few retain most of the variation present in all of the original variables.

PCA method is applied in face recognition by discriminating the image data into several classes. There will be a lot of noise in the image caused by differing lighting conditions, pose and so on. Despite these noises, there are patterns that can be observed in the image such as the presence of eyes, mouth or nose in a face and the relative distances between these objects. PCA's aim is to extract these patterns or the features from the image.

In the domain of face recognition, the principal components (features) are referred to as Eigen faces. The original image can be reconstructed by summing up all the Eigen faces in the right proportion by adding more weights to the real features of the face. Certain Eigen faces that do not contribute to the important face features are omitted. This is necessary because of performance issues while doing large computations. This idea is applied in our approach where Eigen faces are prepared from a set of training images and stored first. Then Eigen face is prepared for the input test image and compare with the training images. The matching image is the image having similar weights in the test database.

Sirovich and Kirby showed that a collection of face images can be approximately reconstructed by storing a small collection of weights for each face and a small set of standard pictures [14].

Using Principal Component Analysis on a set of human face images, a set of Eigen faces can be generated. Eigen faces are a set of eigenvectors used mostly in human face recognition. Eigenvectors are a set of features that characterize the variation between face images. These eigenvectors are derived from the covariance matrix of the probability distribution of the high-dimensional vector space of faces of human beings. The main idea here is to use only the best Eigen faces that account for the major variance within the set of face images. By using lesser Eigen faces, computational efficiency and speed is achieved. The Eigen faces are the basis vectors of the Eigen face decomposition.

Below are the steps of face recognition process:

- A training set of same resolution digital images is initially prepared.
- The images are stored as a matrix with each row corresponding to an image.
- Each image is represented as a vector with (r X c) elements where "r" and "c" are the number of rows and the number of columns respectively.
- An average image is calculated from the individual training set images.
- For each image, the deviation from the average image is then calculated and stored.
- The Eigen vectors and Eigen values are then calculated. These represent the directions in which the training set images differ from the average image.
- A new image is then subtracted from the average image and projected into Eigen face space.
- This is compared with the projection vectors of training faces and the matching image is determined.

A face image is represented by a two dimensional N by N array of intensity values or a vector of dimension N2. If there is an image of size 128 by 128, then that can be said as a vector of dimension 16384. Or, this is equivalent to one point in a 16384-dimensional space. A group of images then maps to a collection of points in this image space. The training images chosen are all of same dimensions. We need to find the vectors that best represent the distribution of face images within this image space and these vectors that define the sub-space of face images are termed as face space. Each vector of length N2 represents an image of dimension N by N and is a linear combination of the original face images. These vectors are termed as Eigen faces because these are the vectors of the covariance matrix corresponding to the original face images and they have face-like appearance.

Step 1: Prepare the test data

Choose "M" training face images and prepare the training set images T_i. The training set of face images are represented as T_1, T_2, $T_3 ... T_M$.

Step 2: Calculate the average of the matrix

Average face of the set $\Psi = 1/M (T_1+T_2+T_3+ ... +T_M)$. [1] (3.1)

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \mathrm{T}_n \qquad (3.2)$$

Ψ : Average face

M: Training face images

Step 3: Subtract the average

For each face, the difference with the average is $\Phi i = Ti - \Psi$ (3.3)

Step 4: Calculate the covariance matrix

These vectors are then subjected to PCA which seeks a set of M orthonormal vectors un and their associated Eigen values λ_k that best represent the distribution of the data. The vectors u_n and λ_k are the eigenvectors and Eigen values, respectively of the covariance matrix

$$\mathrm{C} = \frac{1}{\mathrm{M}}\sum_{\mathrm{n}=1}^{\mathrm{M}} \Phi_{\mathrm{n}} \Phi_{\mathrm{n}}^{\mathrm{T}} \qquad (3.4)$$

$$C = AA^T \text{ where the matrix } A = [\Phi_1, \Phi_2, \Phi_3, \Phi_{4\ldots}, \Phi_M] \qquad (3.5)$$

C: covariance matrix

Step 5: Calculate the eigenvectors and Eigen values of the covariance matrix.

Since the covariance matrix C is of size N^2 by N^2, determining N^2 will be a huge task for typical image sizes. So the alternative is to determine the eigenvectors by solving a smaller M by M matrix and taking linear combinations of the resulting vectors. This would reduce the calculation from the order of the number of pixels in the images N^2 to the order of the images in the training set M.

Step 6: Select the principal components.

Usually, we will use only a subset of *M* Eigen faces, the *M'* Eigen faces with the largest Eigen values. Eigen faces with low Eigen values are omitted, as they explain only a small part of characteristic features of the faces. This completes the training phase of the face recognition.

Step 7: Face recognition – classifying the face

The next task is the face recognition. The test image T test is projected into face space by the following operation:

$$\omega_k = u_k^T(T_{test} - \psi) \text{where } k = 1,2 \ldots . M \qquad (3.6)$$

Weights form a vector $\Omega^T = (\omega_1, \omega_2, \omega_3 \ldots \omega_M)$ describing the contribution of each Eigen face in representing the input face image.

Whichever face class provides a minimum of Euclidean distance of $\varepsilon_k = ||\Omega - \Omega_k||$, where Ω_k is vector defining *k*th face class, is the matching face image.

A threshold of θ defines the maximum allowable distance from the face space beyond which the face is considered as “unknown”. [15]

This algorithm was used in this system to face recognition.

2.5 Face Detection and Tracking Algorithm

Face detection is the process of identifying the human faces in a digital image or a video frame from a video source.

There are many algorithms used in face detection and tracking such as:

- AdaBoost algorithm
- SMQT features and SNOW classifier
- Viola Jones algorithm

2.5.1 AdaBoost Algorithm

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and incorrect rules.

The AdaBoost algorithm was the first practical boosting algorithm, and one of the most widely used and studied, with applications in numerous field. Using boosting algorithm to train a classifier which is capable of processing images rapidly while having high detection rates. AdaBoost is a learning algorithm which produces a strong classifier by choosing visual features in a family of simple classifiers and combining them linearly [16].

Although AdaBoost is more resistant to over fitting than many machine learning algorithms, it is repeatedly sensitive to noisy data and outliers. AdaBoost is called adaptive because it uses multiple iterations to generate a single composite strong learner. AdaBoost creates the strong learner (a classifier that is well-correlated to the true classifier) by iteratively adding weak learners (a classifier that is only slightly correlated to the true classifier). Throughout each round of training, a new weak learner is added to the group and a weighting vector is adjusted to focus on examples that were misclassified in preceding rounds. The outcome is a classifier that has higher accuracy than the weak learners ‘classifiers [17].

2.5.2 SMQT features and SNOW classifier

This method consists of two phase. The primary phase is face luminance. The operation of this phase is being performed to get pixel information of an image and further implemented to detection purpose.

The second phase is detection. In this phase, local SMQT features are used as feature extraction for object detection. The features were found to be able to cope with illumination and sensor variation in object detection. The split up SNOW is proposed to speed up the standard SNOW classifier.

The split up SNOW classifier requires just training of one classifier network which can be arbitrarily divided into several weaker classifiers in cascade. All weak classifier uses the result from previous weaker classifiers which makes it computationally efficient [18].

2.5.3 Viola Jones Algorithm

The Viola–Jones Algorithm is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection then tracking it. [19]

The Voila Jones algorithm is given as follow:

- In voila zone algorithm the detection is done by the Feature extraction and feature evaluation Rectangular features are used, with a new image representation their calculation is very fast.

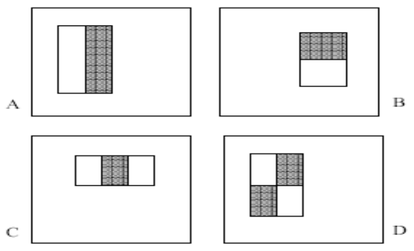


Figure 2-3: Rectangular features

- They are easy to calculate.
- The white areas are subtracted from the black ones.
- A special representation of the sample called the integral image makes feature extraction faster.
- Features are extracted from sub windows of a sample image.

The base size for a sub window is 24 by 24 pixels.

- Each of the four feature types are scaled and shifted across all possible combinations.
- A real face may result in multiple nearby detections
- Post process detected sub windows to combine overlapping detections into a single detection [20].

The algorithm has four stages:

- Haar Feature Selection
- Creating an Integral Image
- Adaboost Training
- Cascading Classifiers

The features sought by the detection framework universally involve the sums of image pixels within rectangular areas. As such, they bear some resemblance to Haar basis functions, which have been used previously in the realm of image-based object detection.[3] However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. The figure on the right illustrates the four different types of features used in the framework. The value of any given feature is the sum of the pixels within clear rectangles subtracted from the sum of the pixels within shaded rectangles. Rectangular features of this sort are primitive when compared to alternatives such as steerable filters. Although they are sensitive to vertical and horizontal features, their feedback is considerably coarser.

- Haar Features

All human faces share some similar properties. These regularities may be matched using Haar Features.

A few properties common to human faces:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

Composition of properties forming matchable facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

The four features matched by this algorithm are then sought in the image of a face (shown at left).

- Rectangle features:

- Value = Σ (pixels in black area) - Σ (pixels in white area)
- Three types: two-, three-, four-rectangles, Viola & Jones used two-rectangle features
- For example: the difference in brightness between the white &black rectangles over a specific area
- Each feature is related to a special location in the sub-window

- Summed area table

An image representation called the integral image evaluates rectangular features in *constant* time, which gives them a considerable speed advantage over more sophisticated alternative features. Because each feature's rectangular area is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in nine.

- Learning algorithm

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of M = 162,336 possible features, and it would be prohibitively expensive to evaluate them all when testing an image. Thus, the object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers

$$h(x) = \mathrm{sign}\left(\sum_{j=1}^{m} \alpha_j h_j(x)\right)$$

Each weak classifier is a threshold function based on the featuref_j

$$h_j(x) = \begin{cases} -s_j & \text{if} \quad f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

The threshold value θ_j and the polarity $s_j \in \pm 1$ are determined in the training, as well as the coefficients.α_j

Here a simplified version of the learning algorithm is reported

Input: Set of N positive and negative training images with their labels (x^i, y^i). If image i is a face $y^i = 1$, if not $y^i = -1$.

A. Initialization: assign a weight $w_1^i = \frac{1}{N}$ to each image i.
B. For each feature f_j with $j = 1, \dots, M$
    1. Renormalize the weights such that they sum to one.
    2. Apply the feature to each image in the training set, then find the optimal threshold and polarity θ_j, s_j that minimizes the weighted classification error. That is

$$\theta_j, s_j = \mathrm{argmin}_{\theta_j, s_j} \sum_{i=1}^{N} w^i_j \varepsilon^i_j \quad \text{where} \quad \varepsilon^i_j = \begin{cases} 0 & \text{if}\, y^i = h_i(x^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases}$$

    3. Assign a weight α_j to h_i that is inversely proportional to the error rate. In this way best classifiers are considered more.

4. The weights for the next iteration, i.e. w^{i}_{j+1} , are reduced for the images i that were correctly classified.

C. Set the final classifier to

$$h(x)=\text{sign}\left(\sum_{j=1}^{m} \alpha_j h_j(x)\right)$$

- Cascade architecture

- On average only 0.01% of all sub-windows are positive (faces)
- Equal computation time is spent on all sub-windows
- Must spend most time only on potentially positive sub-windows.
- A simple 2-feature classifier can achieve almost 100% detection rate with 50% FP rate.
- That classifier can act as a 1st layer of a series to filter out most negative windows
- 2nd layer with 10 features can tackle “harder” negative-windows which survived the 1st layer, and so on…
- A cascade of gradually more complex classifiers achieves even better detection rates. The evaluation of the strong classifiers generated by the learning process can be done quickly, but it isn’t fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and continue on searching the next sub-window. The cascade therefore has the form of a degenerate

tree. In the case of faces, the first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%.[6] The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated.

In cascading, each stage consists of a strong classifier. So all the features are grouped into several stages where each stage has certain number of features. The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

A simple framework for cascade training is given below:

- f = the maximum acceptable false positive rate per layer.
- d = the minimum acceptable detection rate per layer.
- Ftarget = target overall false positive rate.
- P = set of positive examples.
- N = set of negative examples.

The cascade architecture has interesting implications for the performance of the individual classifiers. Because the activation of each classifier depends entirely on the behavior of its predecessor, the false positive rate for an entire cascade is:

$$\text{F}=\prod_{i=1}^{K} f_i$$

Similarly, the detection rate is:

$$D=\prod_{i=1}^{K} d_i$$

Thus, to match the false positive rates typically achieved by other detectors, each classifier can get away with having surprisingly poor performance. For example, for a 32-stage cascade to achieve a false positive rate of 10−6, each classifier need only achieve a false positive rate of about 65%. At the same time, however, each classifier needs to be exceptionally capable if it is to achieve adequate detection rates. For example, to achieve a detection rate of about 90%, each classifier in the aforementioned cascade needs to achieve a detection rate of approximately 99.7%

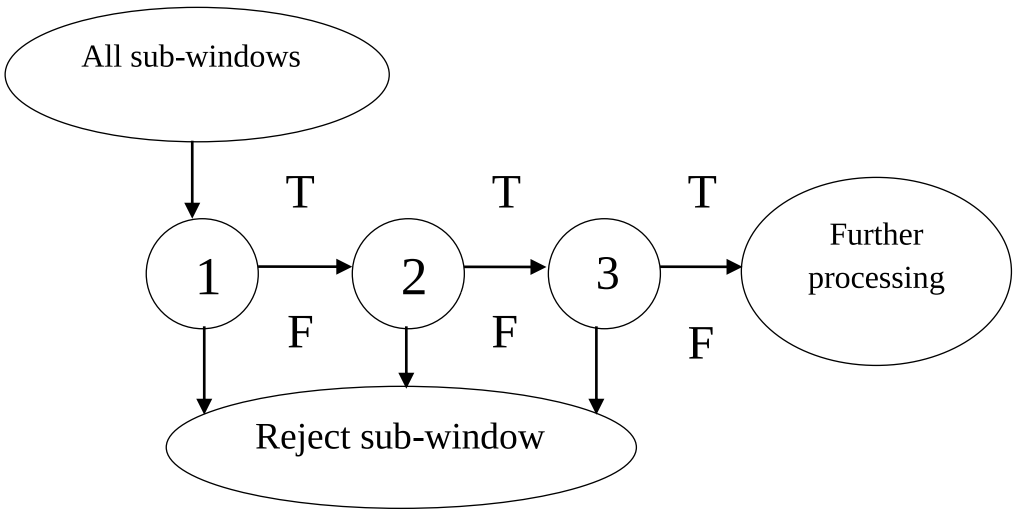


Figure 2-4: A Cascade of Classifiers

This algorithm was used in this system to face Detection then tracking.

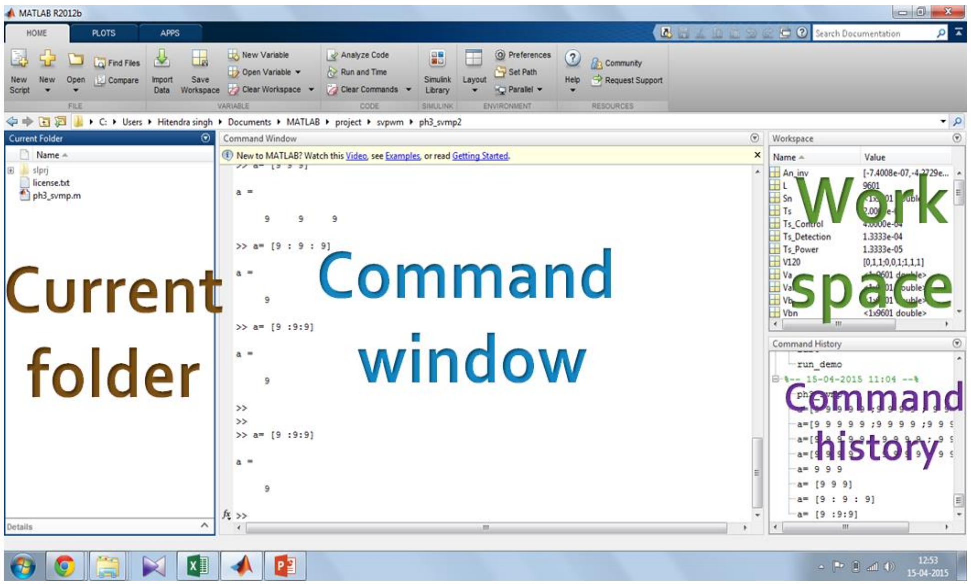
2.6 Software Environment

In this section the software that used in this system was mention.

2.6.1 Matlab environment

Millions of engineers and scientists worldwide use Matlab to analyze and design the systems and products transforming our world. Matlab is in automobile active safety systems, interplanetary spacecraft, and health monitoring devices, smart power grids, and Long Term Evolution "LTE" cellular networks. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more. The Matlab platform is optimized for solving engineering and scientific problems. The matrix-based Matlab language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data [21].

A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain. Matlab code can be integrated with other languages, enabling to deploy algorithms and applications within web, enterprise, and production systems. Information about digital image processing using Matlab and about programming graphics and Graphical User Interface "GUIs" with Matlab can be found, the matlab functions can also be used for detection and recognition. This project, have been implemented using Matlab programming environment version R2014. Although we cannot guarantee that, all functions will work properly in older versions of Matlab [22].



```
a =

     9     9     9

>> a= [9 : 9 : 9]

a =

     9

>> a= [9 :9:9]

a =

     9

>>
>>
>> a= [9 :9:9]

a =

     9

>>
```

Figure 2-5: Matlab environment

2.6.2 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, Automatic control to regulate the voltage "AVR" Studio and the newer Atmel Studio. [46][47][48]

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU tool chain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware [23].

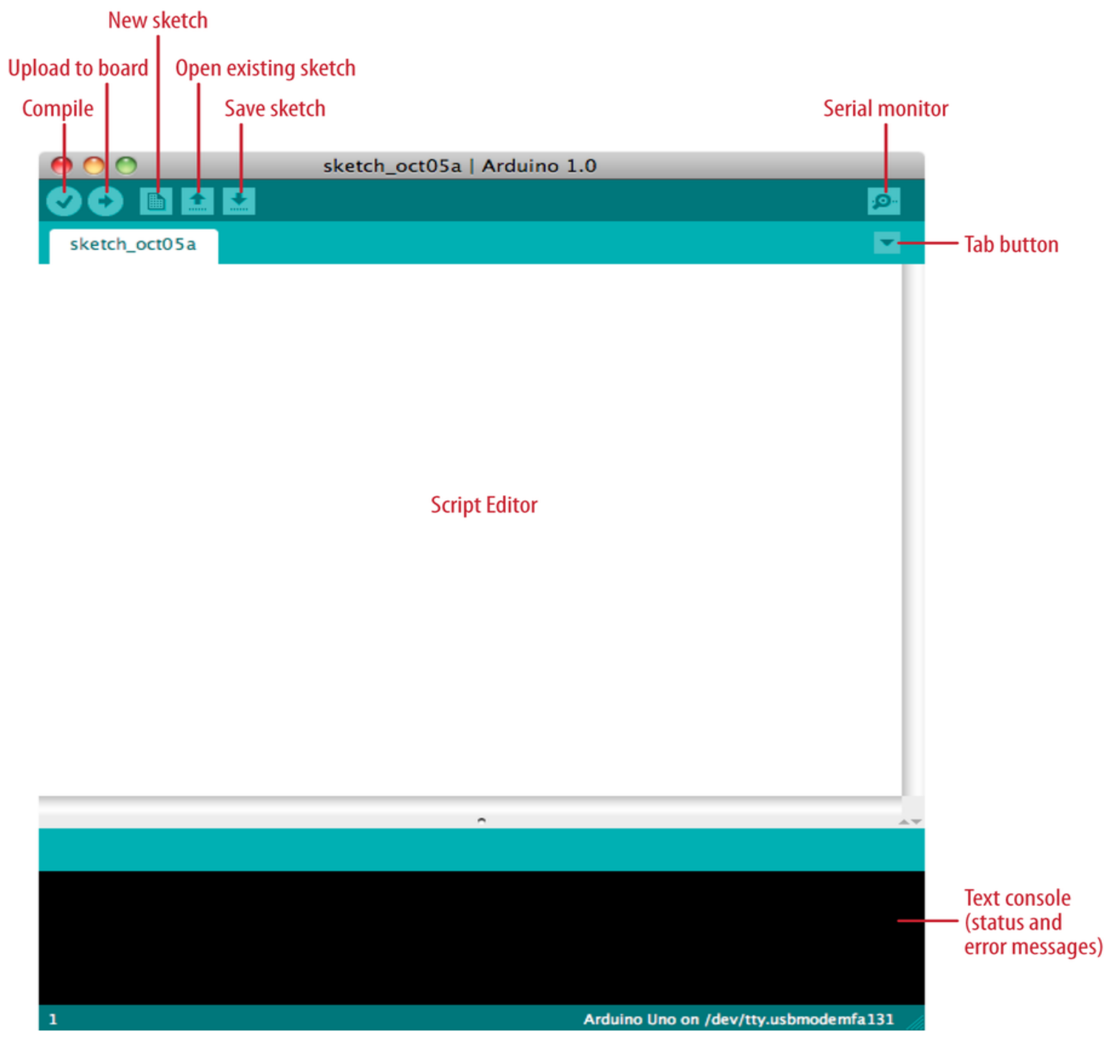


Figure 2-6: Arduino IDE Software

2.7 Control Unit

The process of tracking requires a hardware unit to control the motors movement.

Many types of controllers can be used to perform this operation such as:

- Microcontroller
- Raspberry Pi
- Arduino kit

2.7.1 Microcontroller

A microcontroller is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, peripherals, automobiles and household appliances for computer systems. Due to that, another name for a microcontroller is "embedded controller." Some embedded systems are more sophisticated, while others have minimal requirements for memory and programming length and a low software complexity. Input and output devices include solenoids, Liquid crystal display "LCD", relays, switches and sensors for data like humidity, temperature or light level, amongst others.

There are several different kinds of programmable microcontrollers at Future Electronics. We stock many of the most common types categorized by several parameters including Bits, Flash size, RAM size, number of input/output lines, packaging type, supply voltage and speed. Our parametric filters will allow you to refine your search results according to the required specifications.

Programmable microcontrollers contain general purpose input/output pins. The number of these pins varies depending on the microcontroller. They can be configured to an input or an output state by software. When configured to an input state, these pins can be used to read external signals or sensors. When they are configured to the output state, they can drive external devices like Light Emitting Diode "LED" displays and motors.

Programmable microcontrollers are designed to be used for embedded applications, unlike microprocessors that can be found in PCs. Microcontrollers are used in automatically controlled devices including power tools, toys, implantable medical devices, office machines, engine control systems, appliances, remote controls and other types of embedded systems. [24]

2.7.2 Raspberry Pi

A Raspberry Pi is a credit card-sized computer originally designed for education, it's board contains Broadcom based ARM Processor, Graphics Chip, Random access memory "RAM", General - Purpose Input/Output "GPIO" and other connectors for external devices. The operating procedure of Raspberry Pi is very similar as compared to PC and requires additional hardware like Keyboard, Mouse, Display Unit, Power Supply, Secure Digital "SD" Card with Operating system "OS" Installed (Acting like Hard Disk) for operation. Raspberry Pi also facilitates Universal Serial Bus "USB" ports, Ethernet for Internet/Network-Peer to Peer Connectivity.

Like any other computer, where Operating system acts as backbone for operation. Raspberry Pi, facilitates open source operating system’s based on Linux. [25]

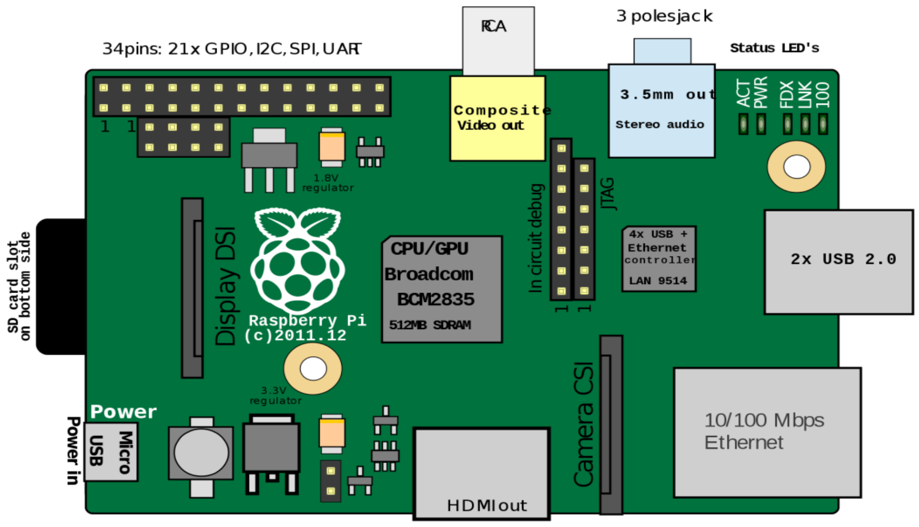


Figure 2-7: Raspberry Pi

2.7.3 Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

Arduino makes several different boards, each with different capabilities. In addition, part of being open source hardware means that others can modify and produce derivatives of Arduino boards that provide even more form factors and functionality. If you're not sure which one is right for your project, check this guide for some helpful hints. Here are a few options that are well-suited to someone new to the world of Arduino: [26]

- Arduino Uno

The Uno is a great choice for your first Arduino. It's got everything you need to get started, and nothing you don't. It has 14 digital input/output pins (of which 6 can be used as Pulse width modulation "PWM" outputs), 6 analog inputs, a USB connection, a power jack, a reset button and more. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a Alternating current "AC"-to- Direct current"DC" adapter or battery to get started.

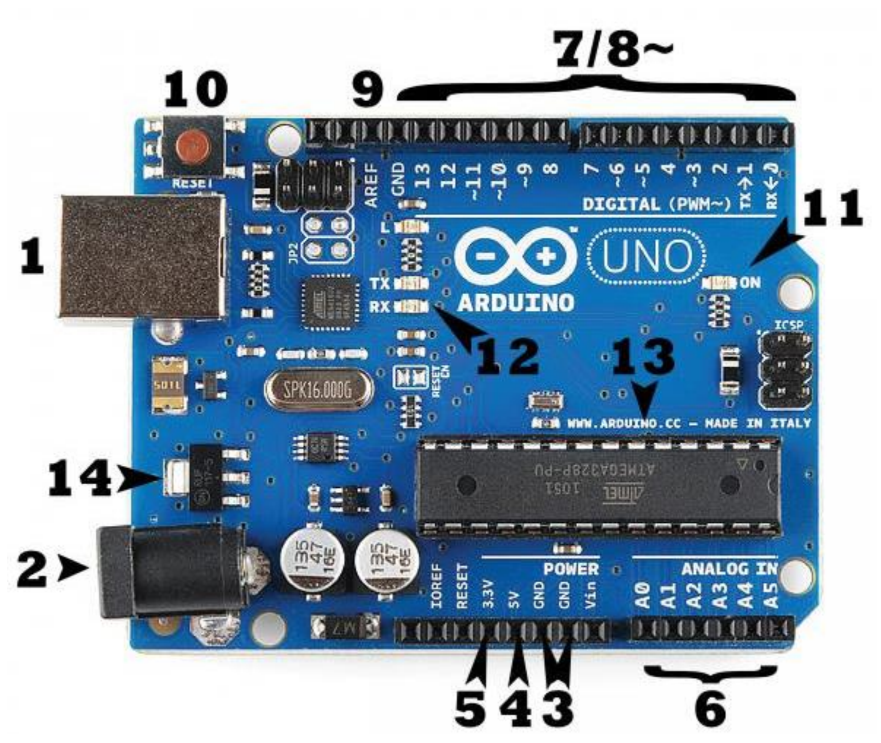


Figure 2-8: Arduino Uno

- Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2)

- Pins (5V, 3.3V, Ground "GND", Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

GND (3): Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog (6): The area of pins under the ‘Analog IN’ label (A0 through A5 on the UNO) are Analog IN pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins

- Reset Button

Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn’t repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn’t usually fix any problems.

- Power LED Indicator

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’ (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong. Time to re-check your circuit!

- Transmitter "Tx" Receiver "Rx" LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

- Main Integrated Circuit "IC"

The black thing with all the metal legs is an IC, or Integrated Circuit (13). Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

- Voltage Regulator

The voltage regulator (14) is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino anything greater than 20 volts [26].

This type of Arduino was used in this system to control servo motor.

- LilyPad Arduino

LilyPad is a wearable e-textile technology developed by Leah Buechley and cooperatively designed by Leah and SparkFun. Each LilyPad was creatively designed with large connecting pads and a flat back to allow them to be sewn into clothing with conductive thread. The LilyPad also has its own family of input, output, power, and sensor boards that are also built specifically for e-textiles. They're even washable! [26].

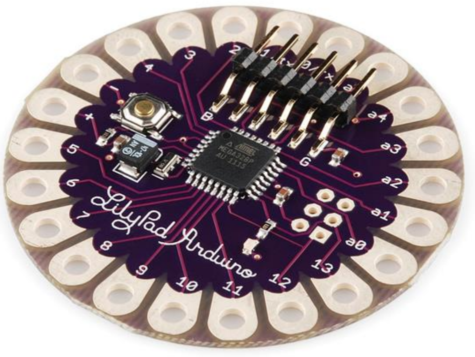


Figure 2-9: LilyPad Arduino

- RedBoard Arduino

The RedBoard can be programmed over a USB Mini-B cable using the Arduino IDE. It'll work on Windows 8 without having to change your security settings (we used signed drivers, unlike the UNO). It's more stable due to the USB/Future Technology Devices International "FTDI" chip we used, plus it's completely flat on the back, making it easier to embed in your projects. Just plug in the board, select "Arduino UNO" from the board menu and you're ready to upload code. You can power the RedBoard over USB or through the barrel jack. The on-board power regulator can handle anything from 7 to 15VDC [26].

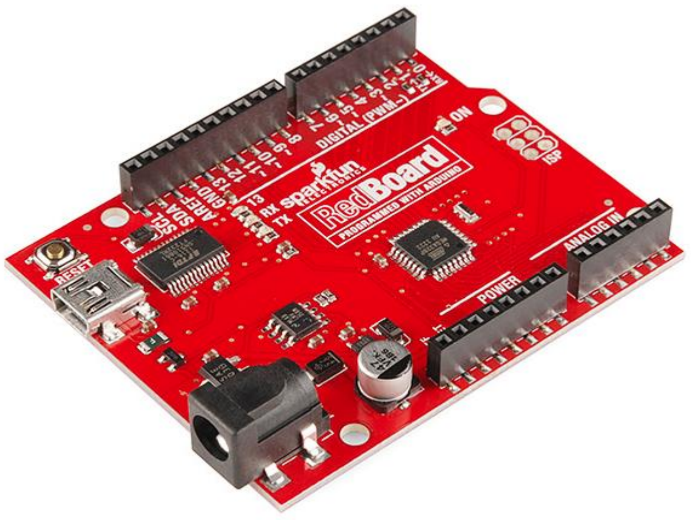


Figure 2-10: RedBoard Arduino

- Arduino Mega

The Arduino Mega is like the UNO's big brother. It has lots (54!) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The large number of pins make this board very handy for projects that require a bunch of digital inputs or outputs (like lots of LEDs or buttons) [26].

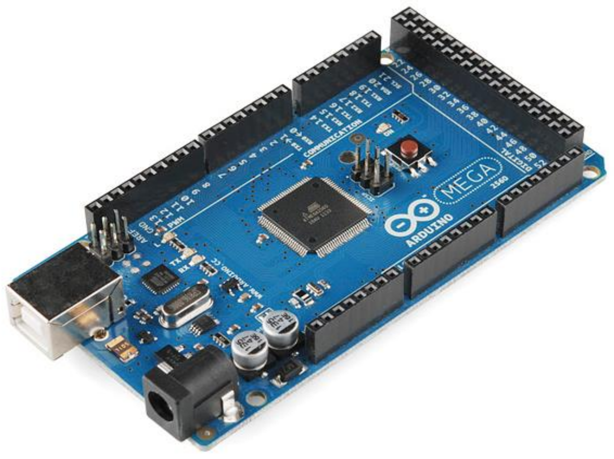


Figure 2-11: Arduino Mega

- Arduino Leonardo

The Leonardo is Arduino's first development board to use one microcontroller with built-in USB. This means that it can be cheaper and simpler. Also, because the board is handling USB directly, code libraries are available which allow the board to emulate a computer keyboard, mouse, and more! [26].

Figure 2-12: Arduino Leonardo

2.8 Motors

Motors are the power transmission workhorse of industry, converting electrical energy into mechanical movement. There many type of motor some of them are listed below.

2.8.1 DC Motor

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances.

There are three types of electrical connections between the stator and rotor possible for DC electric motors: series, shunt/parallel and compound (various blends of series and shunt/parallel) and each has unique speed/torque characteristics appropriate for different loading.[27]

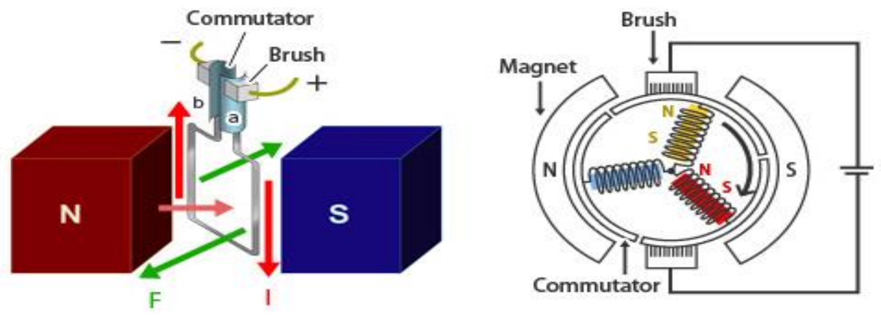


Figure 2-13: DC Motor

2.8.2 AC Motor

An AC motor is an electric motor driven by an alternating current (AC). The AC motor commonly consists of two basic parts, an outside stationary stator having coils supplied with alternating current to produce a rotating magnetic field, and an inside rotor attached to the output shaft producing a second rotating magnetic field. The rotor magnetic field may be produced by permanent magnets, reluctance saliency, or DC or AC electrical windings. Less commonly, linear AC motors operate on similar principles as rotating motors but have their stationary and moving parts arranged in a straight line configuration, producing linear motion instead of rotation.

The simple AC Motor contains a coil of wire and two fixed magnets surrounding a shaft. When an electric (AC) charge is applied to the coil of wire, it becomes an electromagnet, generating a magnetic field. Simply described, when the magnets interact, the shaft and the coil of wires begin to rotate, operating the motor.

The AC Motor comes in three different types known as Induction, Synchronous, and Industrial. These AC Motor types are determined by the rotor design used in the construction. Anaheim Automation carries all three types in its product line. [28]

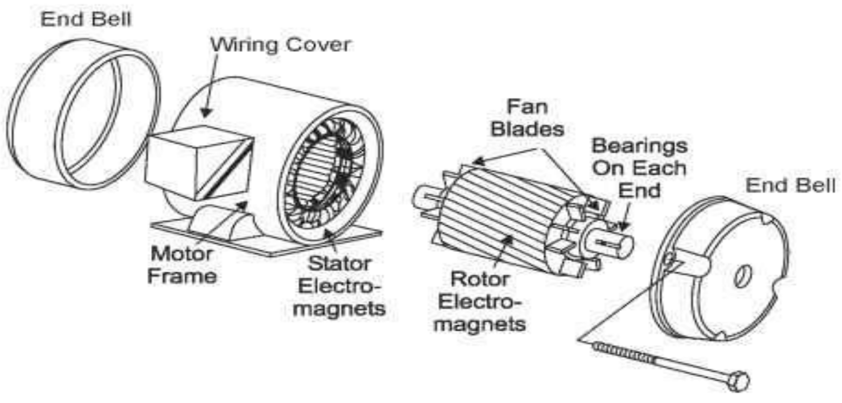


Figure 2-14: AC Motor

2.8.3 Stepper Motor

A stepper motor or step motor or stepping motor is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

There are three main types of stepper motors: Permanent magnet stepper ,Hybrid synchronous stepper and Variable reluctance stepper. Permanent magnet motors use a permanent magnet (PM) in the rotor and operate on the attraction or repulsion between the rotor PM and the stator electromagnets. Variable reluctance (VR) motors have a plain iron rotor and operate based on the principle that minimum reluctance occurs with minimum gap, hence the rotor points are attracted toward the stator magnet poles.[29]

There are two basic winding arrangements for the electromagnetic coils in a two phase stepper motor: bipolar and unipolar.

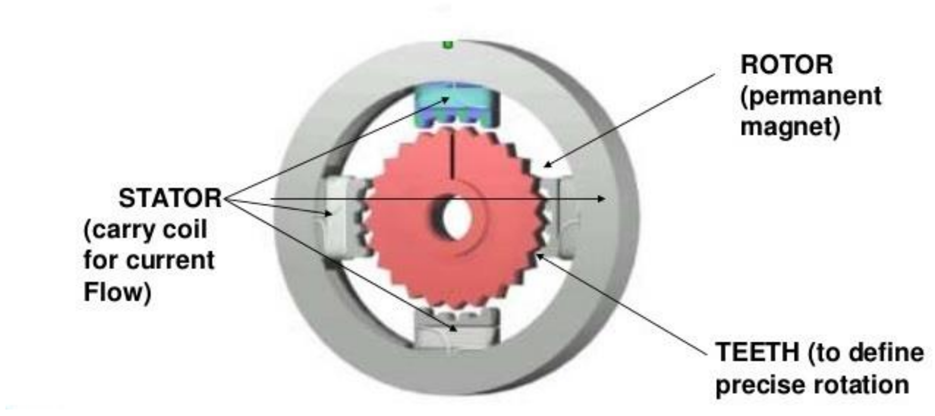

ROTOR (permanent magnet)
STATOR (carry coil for current Flow)
TEETH (to define precise rotation

Figure 2-15: Stepper Motor

2.8.4 Servo Motor

servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are used in applications such as robotics, Computer Numerical Control "CNC" machinery or automated manufacturing. The first servomotors were developed with synchro’s as their encoders. Modern servomotors use rotary encoders, either absolute or incremental. Absolute encoders can determine their position at power-on, but are more complicated and expensive. Incremental encoders are simpler, cheaper and work at faster speeds. Incremental systems, like stepper motors, often combine their inherent ability to measure intervals of rotation with a simple zero-position sensor to set their position at start-up. Most modern servomotors are designed and supplied around a dedicated controller module from the same manufacturer. Controllers may also be developed around microcontrollers in order to reduce cost for large-volume applications.

In this project it was used to move the camera with the direction of facial movement [30].

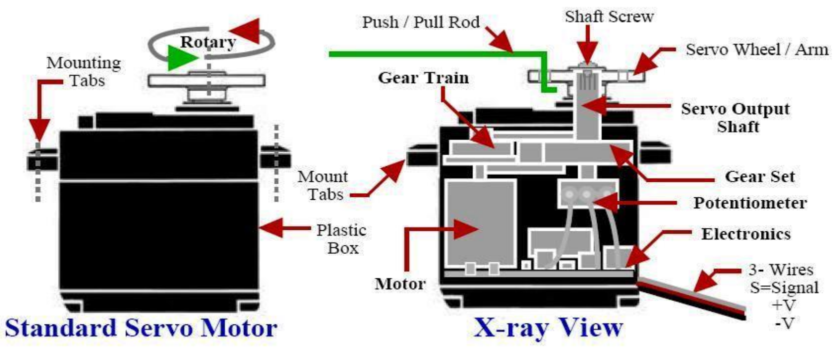


Figure 2-16: Servo Motor

2.9 Webcam

A webcam is a video camera that feeds or streams its image in real time to or through a computer to a computer network. When "captured" by the computer, the video stream may be saved, viewed or sent on to other networks via systems such as the internet, and emailed as an attachment. When sent to a remote location, the video stream may be saved, viewed or on sent there. Unlike an Internet protocol "IP" camera (which connects using Ethernet or Wireless fidelity "Wi-Fi"), a webcam is generally connected by a USB cable, or similar cable, or built into computer hardware, such as laptops. The term "webcam" (a clipped compound) may also be used in its original sense of a video camera connected to the Web continuously for an indefinite time, rather than for a particular session, generally providing a view for anyone who visits its web page Over the Internet. Some of them, for example, those used as online traffic cameras, are expensive, rugged professional video cameras.

First developed in 1991, a webcam was pointed at the Trojan Room coffee pot in the Cambridge University Computer Science Department (initially operating over a local network instead of the web). The camera was finally switched off on August 22, 2001. The final image captured by the camera can still be viewed at its homepage. In 2004, the oldest webcam still operating was FogCam at San Francisco State University, which had been running continuously since 1994 [31].

Figure 2-17: Webcam

CHAPTER THREE

METHODOLOGY

Chapter Three

Methodology

3.1 Overview

The system consists two parts: software implementation and hardware implementation. In software implementation matlab will be used to capture the image from webcam and compare it with the saved image in database. In hardware implementation webcam moves with the direction of the facial movement detected by servo motor, which was attached to the Arduino, and Arduino is linked to the Matlab in PC. System Block diagram as shown in the figure 3-1.

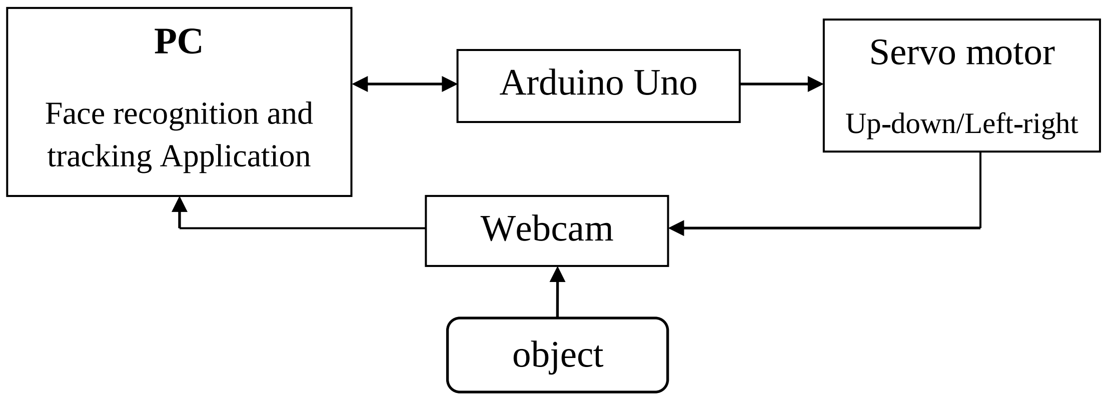


Figure 3-1: System Block diagram

3.2 Circuit Diagram

System contains on Matlab, 2 servo motors, Arduino uno and webcam. Figure 3-2 below show how system component connects with each other.

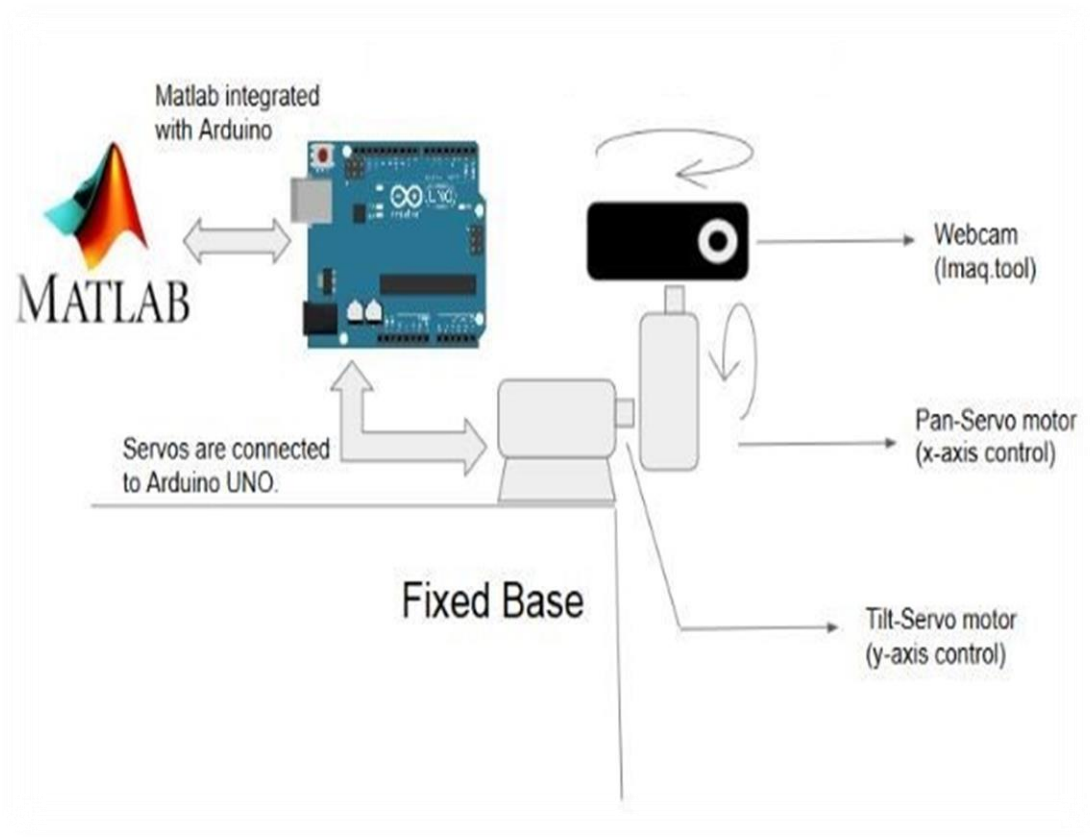

Matlab integrated with Arduino
MATLAB
Webcam (Imaq.tool)
Pan-Servo motor (x-axis control)
Servos are connected to Arduino UNO.
Fixed Base
Tilt-Servo motor (y-axis control)

Fig 3-2: Face recognition and tracking component block diagram

3.3 Principle of Operation

Graphical menu which contain choices was created. Simply clicking through mouse on specified choice can perform the desired action as shown in figure 3-3 below.

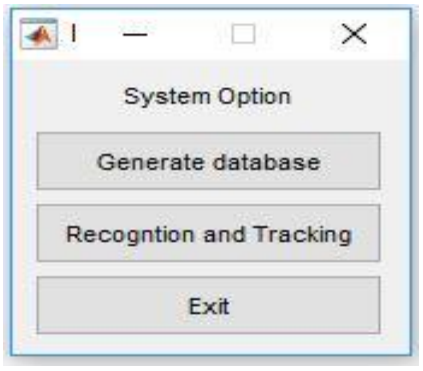


Figure 3-3: System Option

3.3.1 Database generation

Database was created by simply taking face photos through camera, renaming and saving face images of .jpg format in a particular folder then comparing with the face to recognize with database faces. The number of faces that saves in database is equal to M * times, where M is the no. of person entered by user and times is used for increasing the accuracy i.e. let times be 5, so 5 face images per person. The flow chart for database generation is given below in figure 3-4.

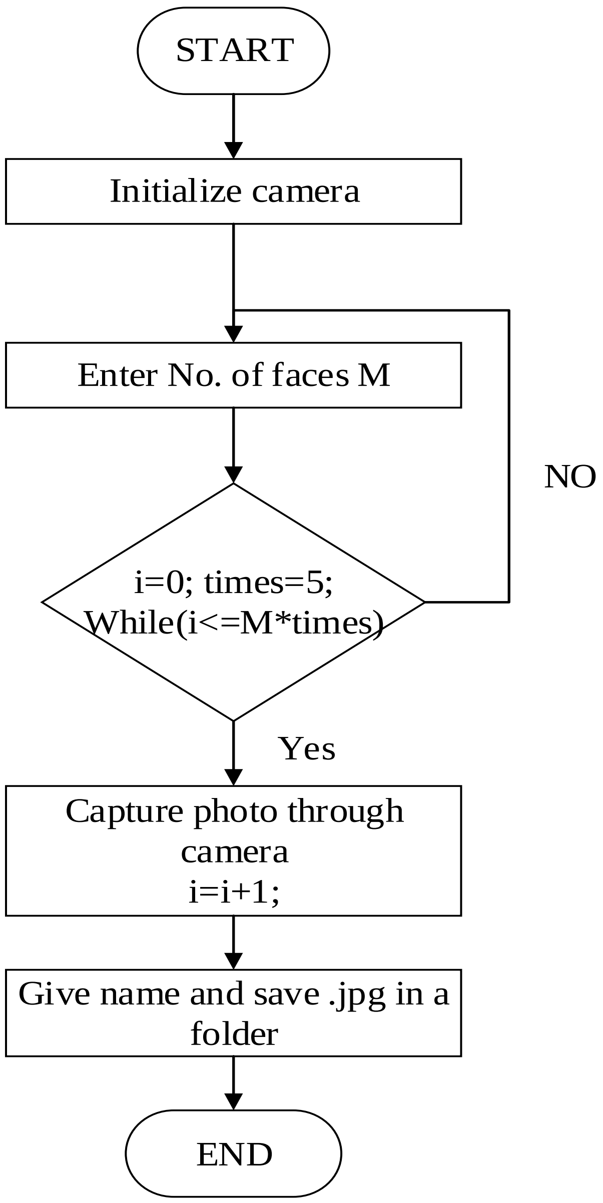


Figure 3-4: Flow chart for generating database

3.3.2 Face Recognition

The recognition process is done by Eigen face algorithm which comparing picture that taken from webcam with picture that saved in database. The flow chart for recognition process is given below in figure 3-5.

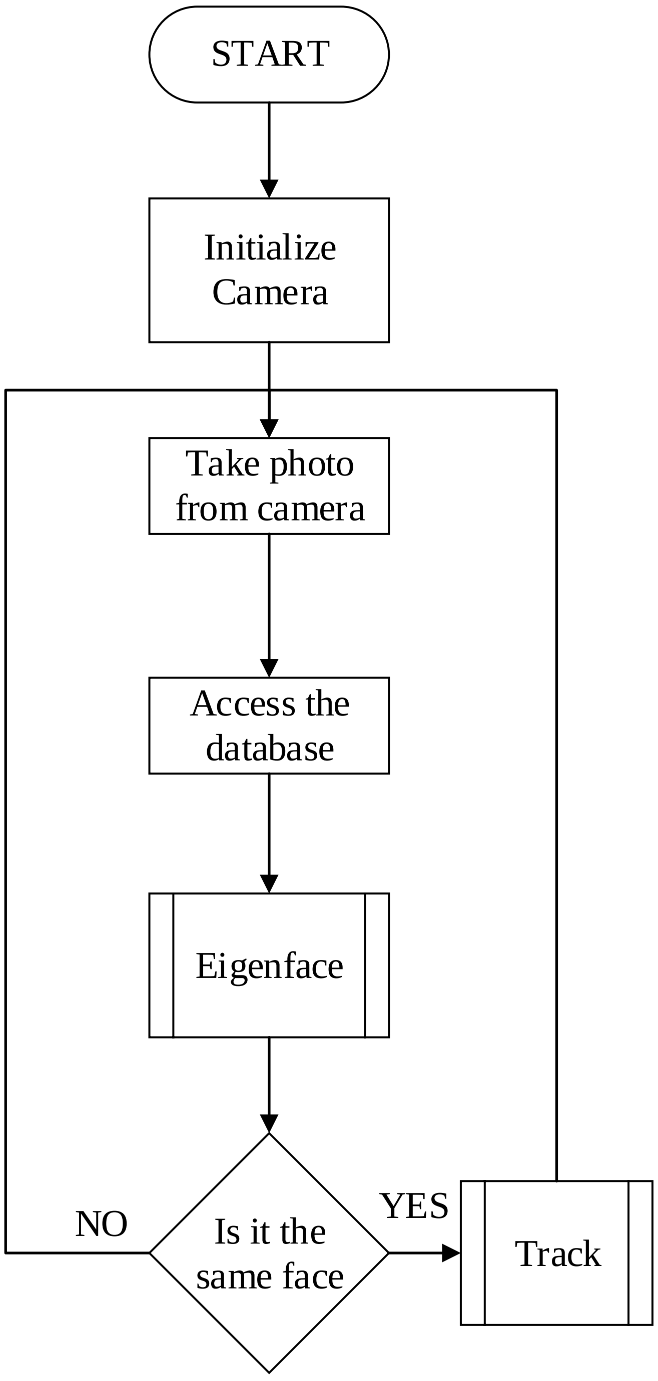


Figure 3-5: Flow chart for recognition process

The Eigenface algorithm was explained in detail in the previous chapter. In this section we show flow chart of Face Recognition using Eigenface in figure 3-6 and figure 3-7.

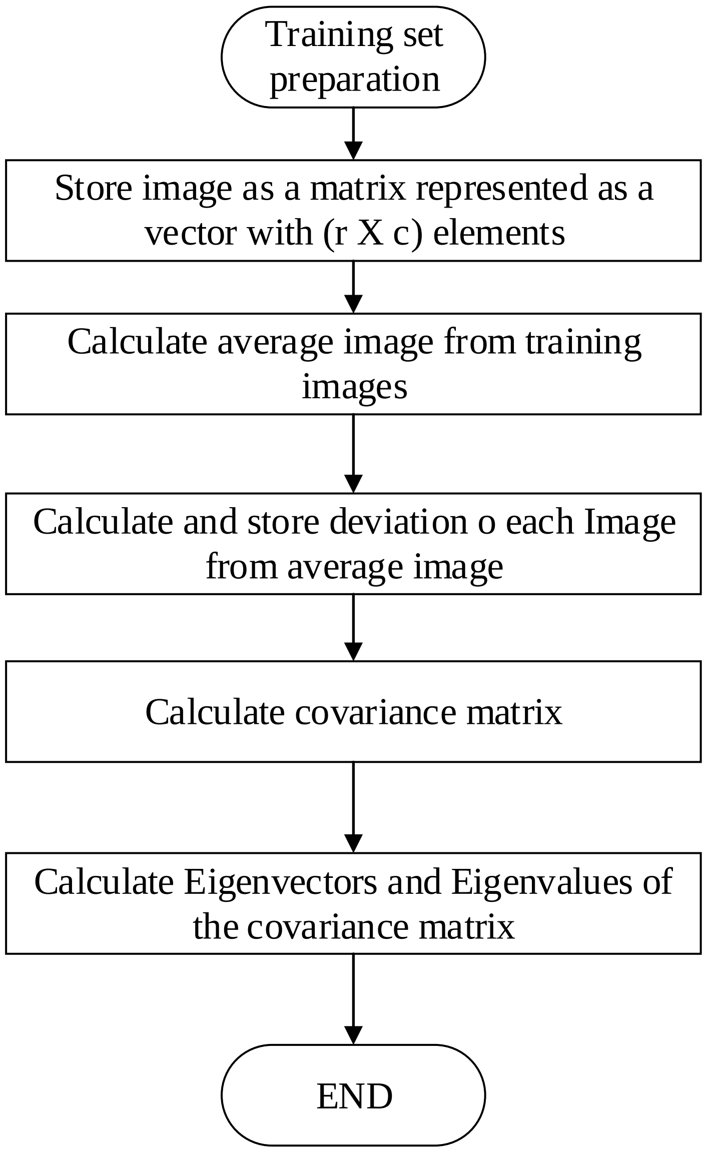

Training set preparation
Store image as a matrix represented as a vector with (r X c) elements
Calculate average image from training images
Calculate and store deviation o each Image from average image
Calculate covariance matrix
Calculate Eigenvectors and Eigenvalues of the covariance matrix
END

Figure 3-6: Train set preparation

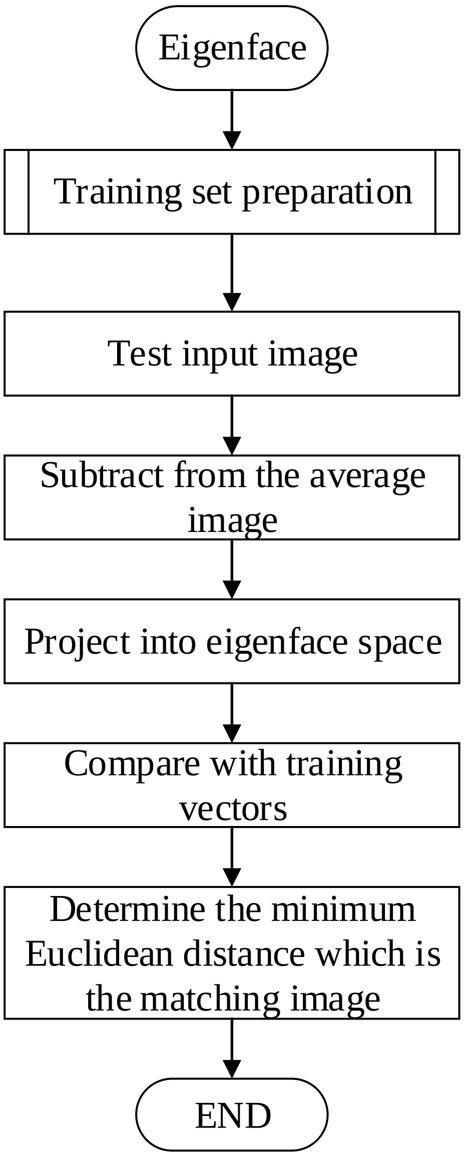

Eigenface
Training set preparation
Test input image
Subtract from the average image
Project into eigenface space
Compare with training vectors
Determine the minimum Euclidean distance which is the matching image
END

Figure 3-7: Face recognition stage

3.3.3 Object Tracking

The face tracking is done using Viola Jones algorithm. It was explained in the previous chapter. Code in Matlab detects a face from every frame of the live video stream and inserts a bounding box around the Region of Interest, which is a face in this case (by detecting some haar features present in the human faces).

The set of frames with bounding boxes make up the addition of a bounding box around the face in live video. While adding a bounding box, the coordinates of centroid of the bounding box was calculated also. These coordinates are sent as a string to the arduino UNO microcontroller, from Matlab.

Figure 3-7 below show flow chart for tracking process.

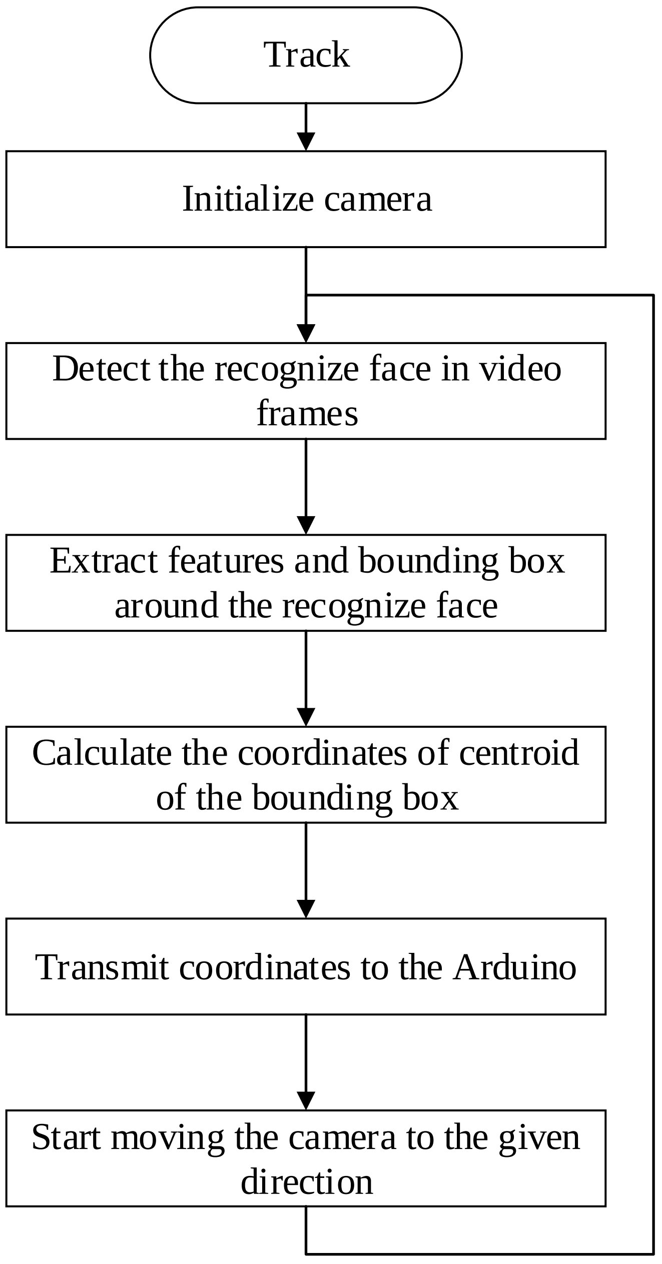

Track
Initialize camera
Detect the recognize face in video frames
Extract features and bounding box around the recognize face
Calculate the coordinates of centroid of the bounding box
Transmit coordinates to the Arduino
Start moving the camera to the given direction

Figure 3-8: Flow chart for tracking process

3.4 Arduino Connection

The coordinates that came from matlab are processed according to the code written on arduino IDE for the movement of motors. During processing, the arduino gets the positions of PAN and TILT servo motors. Then, arduino checks if the centroid coordinates lie in the center region of the screen. We are trying to move the camera in such a way that the centroid lies at the center of the frame. (The pan and tilt servos are given to the digital pins 9 and 10 respectively).

For this reason, the frame is divided into left and right halves and also top and bottom halves. If the centroid falls in the left half, the camera is panned right and if it falls in the right half, camera is panned left and the same with the top and bottom halves and tilting.

An image is divided into sections using its parameters such as length and width. These parameters can be determined using size command of MATLAB. Following is the logic behind this algorithm.

centx=bbox(1) + (bbox(3)/2)

centy=bbox(2) - (bbox(4)/2)

The resolution of image is 320x240.

The centroid of the object is computed using the horizontal and vertical mean of the object. Bwlabel is a MATLAB function used to extract the features from a binary image generated after calibration. This function generates a 2-D matrix. Components of this matrix is used for determining the centroid of the detected object. The Matlab send the coordinates in form of ASCII code to the Arduino via the serial, which converts it to PWM to servo motor. Communication of object detection algorithm with Arduino board is done through serial data transfer. To access the serial port of a computer using MATLAB, couple of lines of coding should be done. MATLAB function for serial port access is: -

Arduino=serial('COM3','BaudRate',9600)

Here, COM3 is a serial communication port on which Arduino board is connected. Moreover, a program has to be booted on ATmega 328 using Arduino IDE. This program makes serial communication port as an input port for the Arduino board.

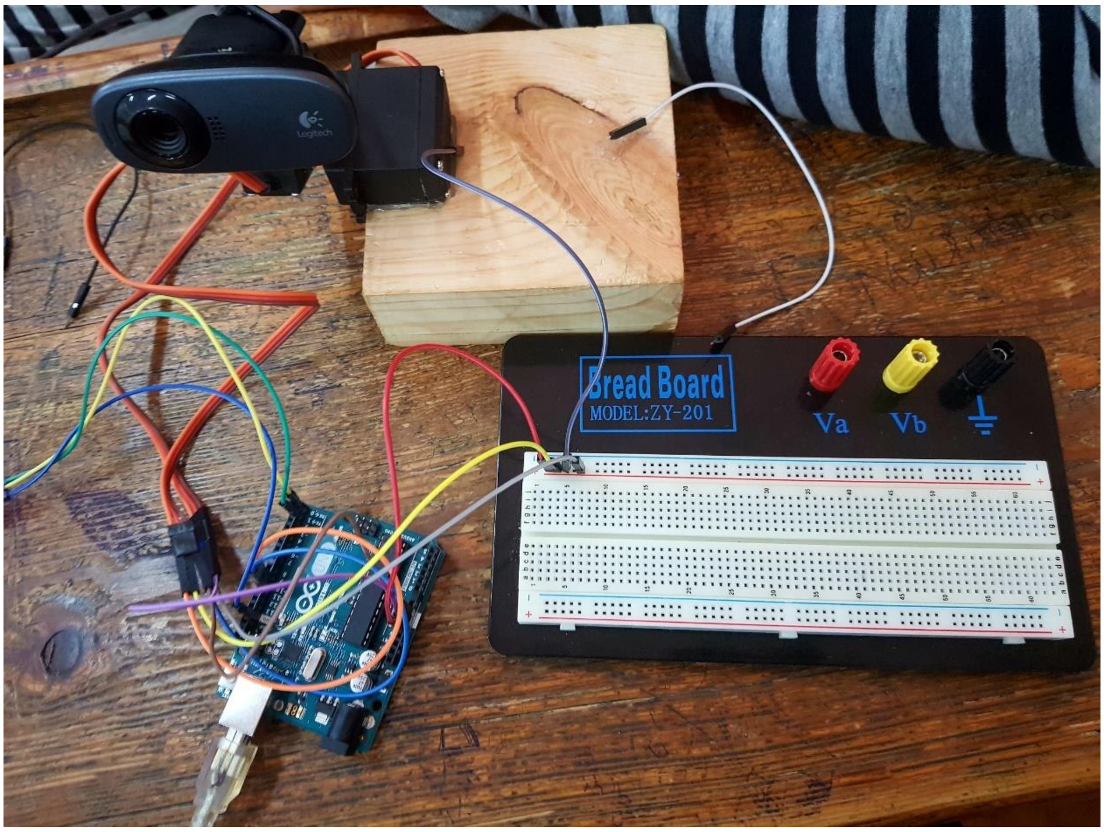


Figure 3-9: Servo motor connecting with Arduino

CHAPTER FOUR
RESULT AND DISCUSSION

Chapter Four

Result and Discussion

4.1 Overview

In the previous chapter, part of the methodology used to implement face recogntion and tracking system have been discussed. The work discussed in the previous chapter comprises the steps from take a picture for face step to tracking it step.

In this chapter, the results that obtained from the system are described and discussed and the performance of system is evaluated.

4.2 Result of Generate Database

Database was Generated Successfuly by taking a image of one person for 5 times in all directions as shown in the figure 4-1.

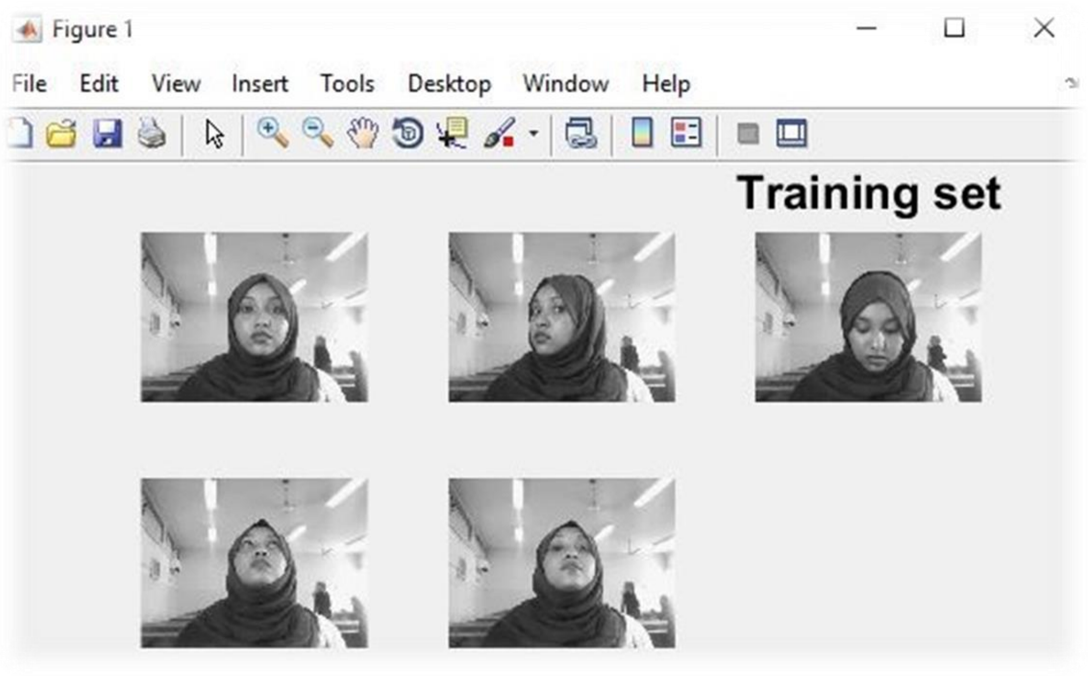


Figure 4-1: Training set

The mean image for the respected set of images is given as shown in the figure 4-2:

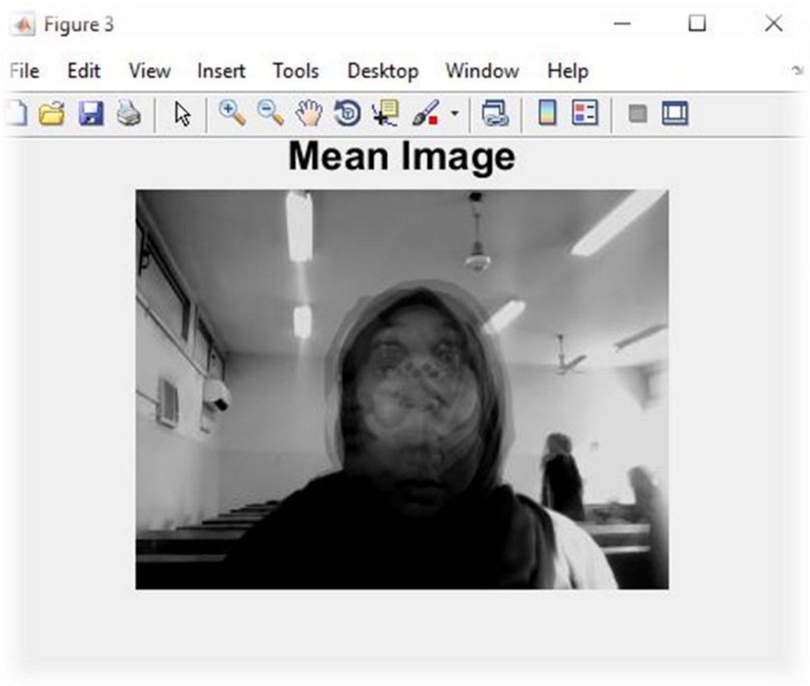


Figure 4-2: Mean Image

4.3 Result of Face Recognition

The Eigenfaces for the given set of image with respect to evaluated mean image is given as shown in figure 4-3 .

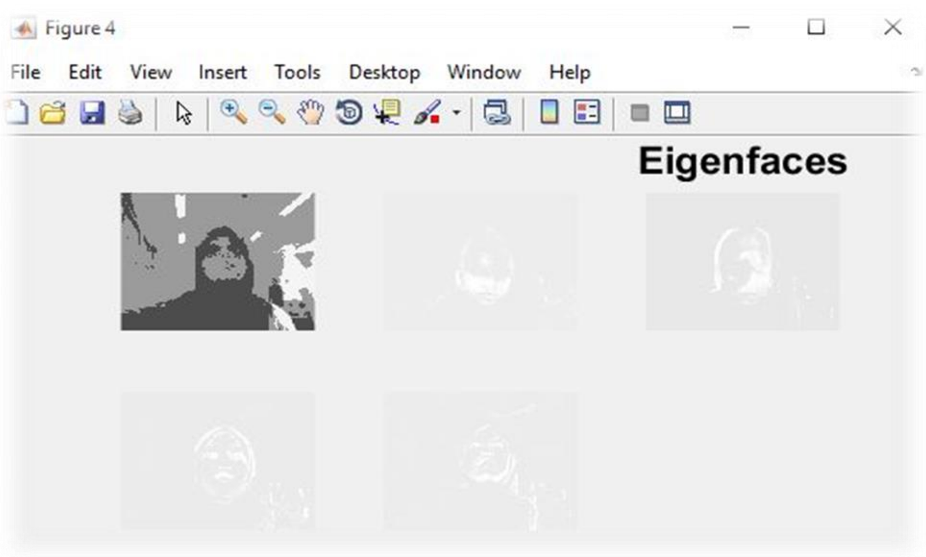


Figure 4-3: Eigenfaces

The same person in database was taken by webcam for recognition as shown in the figure 4-4.

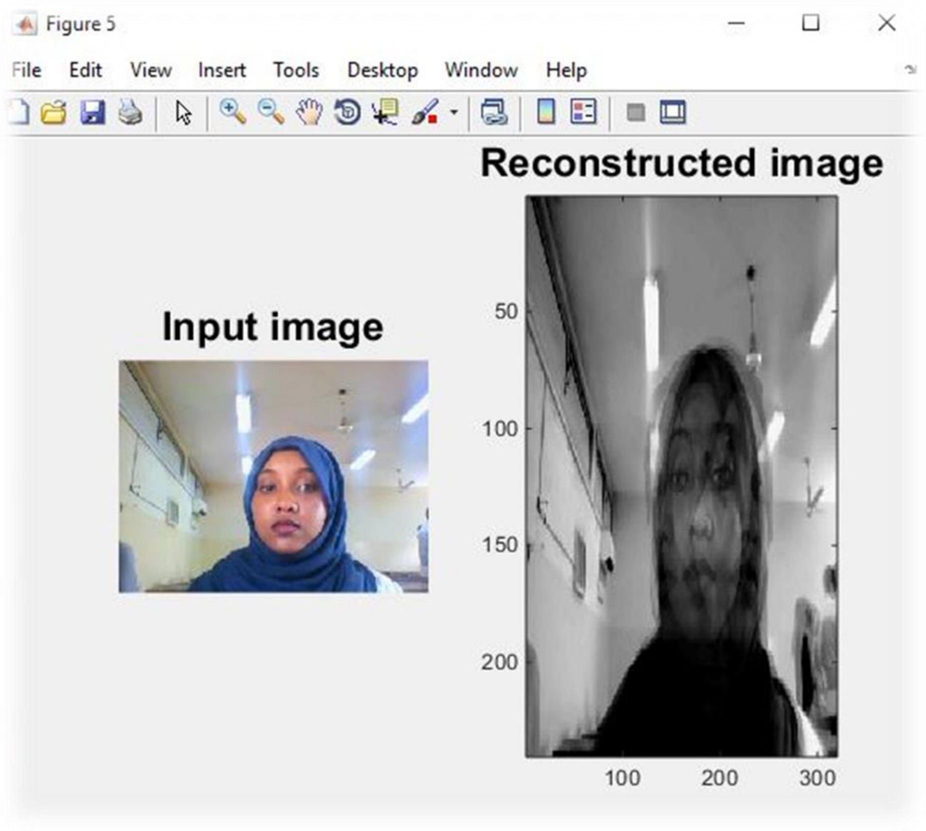
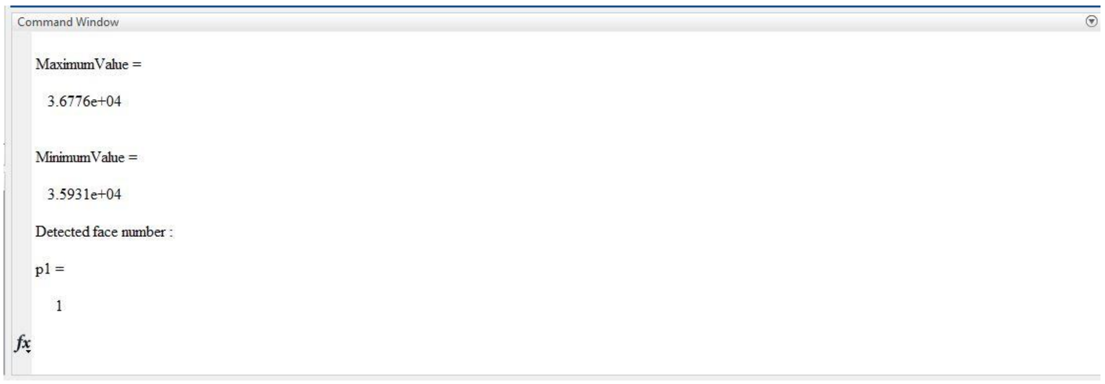


Figure 4-4: Input image

result is given as shown in the figure 4-5:



```
Command Window

MaximumValue =

   3.6776e+04


MinimumValue =

   3.5931e+04

Detected face number :

p1 =

     1

fx
```

Figure 4-5: recognition result

Maximum value and minimum value explain Euclidean distance.

MaximumValue=max(Euclidean distance).

MinimumValue=min(Euclidean distance).

If is the same person in database the maximum value and minimum value will be <= 3.8e+04.

Else the maximum value and minimum value will be > 3.8e+04.

Face class provides the best description for the input image by minimizing the Euclidean distance. The input face is consider to belong to a class if Euclidean distance is bellow an established threshold θε.

Figure 4-6 show another person's picture was taken by a webcam to recognize:

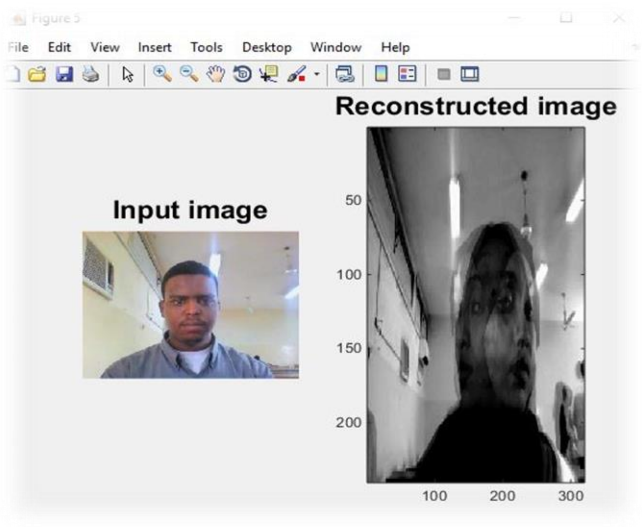
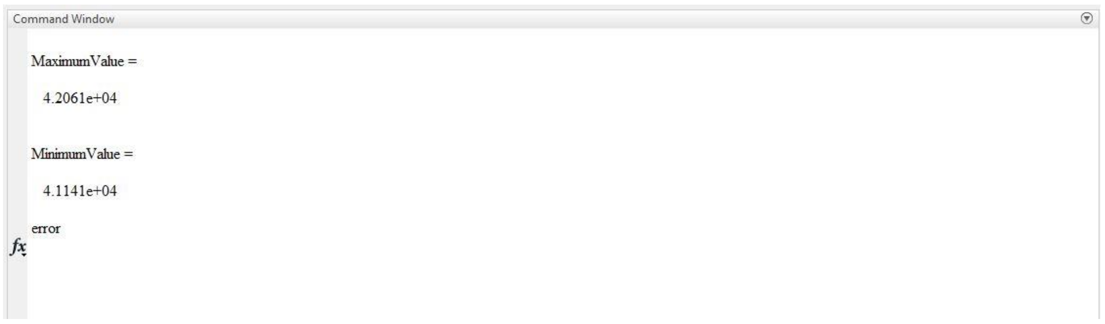


Figure 4-6: Input image

result is given as follow:



```
MaximumValue =

  4.2061e+04

MinimumValue =

  4.1141e+04

error
```

Figure 4-7: recognition result

Successful test result of Face Recognition process using eigenface method. But there were number of challenges and problems we encountered, some of them were:

1- When the program is turned off and restarted face recognition process cannot access the database.

2- In case of similar facial features, the recognition process cannot discriminate.

3- In the case of low light we also had problems.

4.4 Result of Face Tracking

Figure 4-8 show face tracking by using Viola Jones Algorithm.

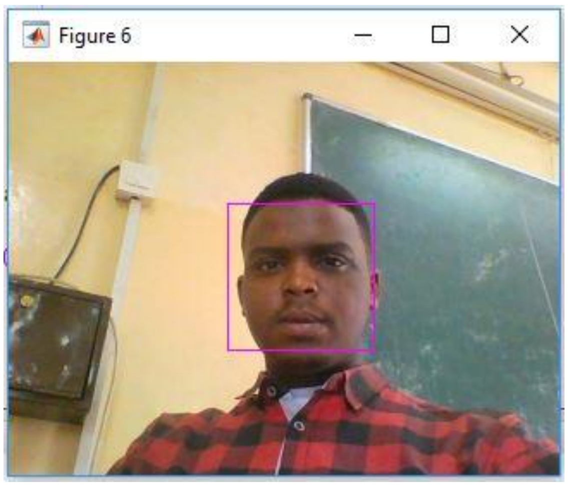


Figure 4-8: Face Tracking

After face recognition process, the face tracking was successfully tested. Some of the problems we encountered are:

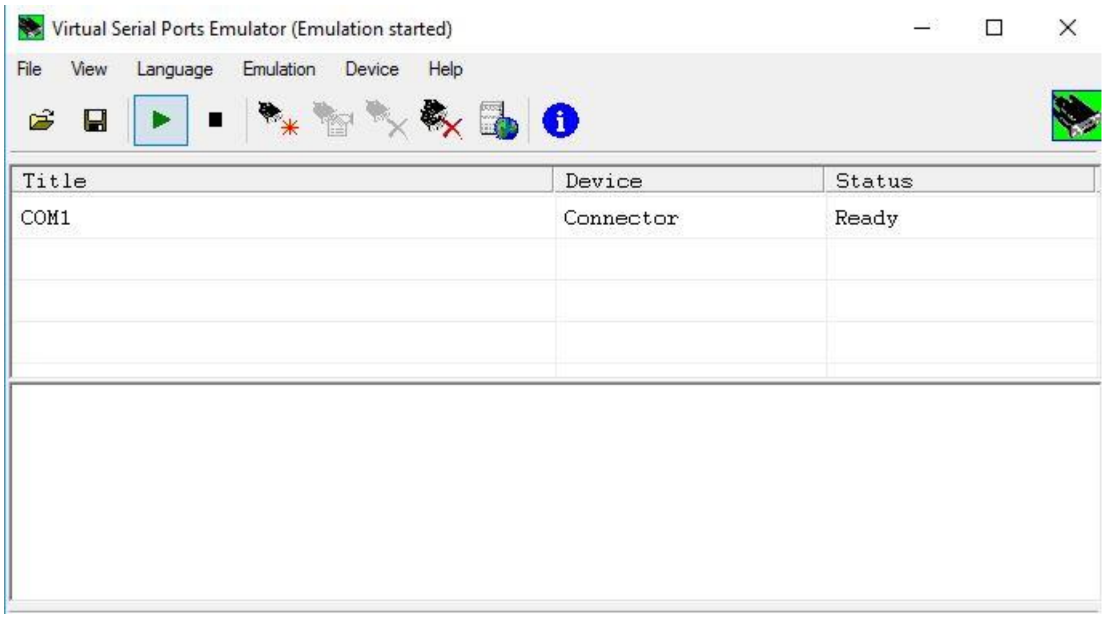
1- After recognition , any face inside the video is tracked

2- Delay in tracking.

3- In low light we also had problems.

4.5 Result of simulation

MATLAB has been linked to the simulation by "virtual serial ports emulator" as shown in figure 4-9.



Title	Device	Status
COM1	Connector	Ready

Figure 4-9: Virtual Serial Ports Emulator

Figure 4-8 show simulation of system hardware.

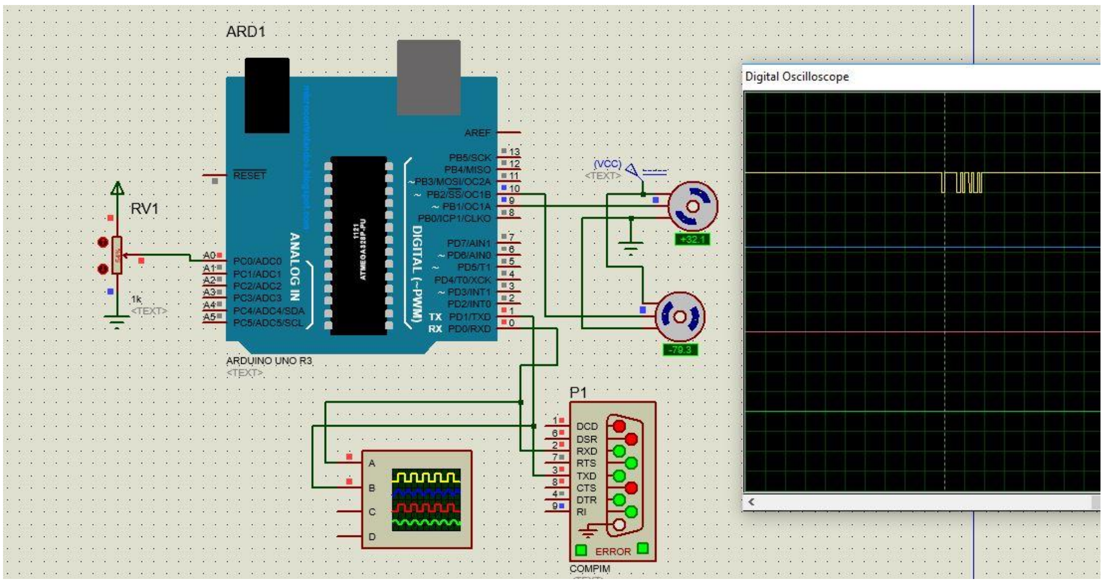


Figure 4-10: System simulation

MATLAB has been linked to the simulation successfully by D9 and the servo motor was successfully moved.

The problem we encountered is movement of motor is slow.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

Chapter Five

Conclusion and Recommendation

5.1 Conclusion

5.2 Recommendation

5.1 Conclusion

In this research a successful simulation as well of face recognition and tracking system has been done. The proposed system is composed of a complete hardware prototype and user friendly Software application.

Face recognition using eigenface algorithm approach is definitely robust, simple, and easy and fast to implement compared to other algorithms, and we implement face tracking in the Matlab by using Viola jones Algorithm. This method is verified and the limitations of the scheme are observed through testing and debugging our codes. And then, limited by Matlab performance. Compared with other popular tracking algorithms such as optical flow, we found the Viola jones algorithm is more suitable for real-time face tracking since they require less CPU resource and costs shorter time.

5.2 Recommendation

Further work still need

- Improve the proposed system and making it more efficient and more secure.
- Solve tracking any face problem by combining the recognition code and tracking code together.
- Use three-dimensional face recognition. This technique uses 3D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin. One advantage of 3D face recognition is that it is not affected by changes in lighting like other techniques. It can also

identify a face from a range of viewing angles, including a profile view.

- Use Skin texture analysis. The addition of skin texture analysis, performance in recognizing faces can increase 20 to 25 percent.
- Use Thermal cameras, by this procedure the cameras will only detect the shape of the head and it will ignore the subject accessories such as glasses, hats, or make up.

REFERENCES

References

[1] Bonsor, K. "How Facial Recognition Systems Work". Retrieved 2008-06-02 R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates.

[2] Muhammet Baykara and Resul Das "Real time face recognition and tracking system" | Software Engineering Department, Firat University Elazig , Turkey "2013.

[3] Unsang Park, Hyun-Cheol Choi, Anil K. Jain, Seong-Whan Lee "Face Tracking and Recognition at a Distance: A Coaxial and Concentric PTZ Camera System" | Department of Computer Science and Engineering, Sogang University, Seoul, Korea "2013.

[4] Zhaowei Cai, Longyin Wen, Dong Cao, Zhen Lei, Dong Yi and Stan Z. Li " Person-Specific Face Tracking with Online Recognition" | Center for Biometrics and Security Research & National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences "2013.

[5] S.V. Viraktamath, Mukund Katti, Aditya Khatawkar and Pavan Kulkarni " Face Detection and Tracking using OpenCV" Department of Electronics and Communication Engineering, SDM College of Engineering & Technology, Dharwad, Karnataka, INDIA "2013.

[6] R. P. Archana, "Medical Application of Image Segmentation with Intensity Inhomogeneities," Int. J. Sci. Eng. Res 2014.

[7] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," *Satell.* Remote Sens. GIS Appl. Agric. Meteorol., vol. 24, pp. 1–30, 2002.

[8] T. Ojala, M. Pietikainen and D. Harwood, “A comparative study of texture measures with classification based on feature distributions” Pattern Recognition vol. 29, 1996.

[9] T. Ahonen, A. Hadid and M. Pietikainen, “Face description with Local Binary Patterns”, Application to Face Recognition. Machine Vision Group, University of Oulu, Finland, 2006.

[10] T. Ahonen, A. Hadid, M. Pietikainen and T. M aenpaa. “Face recognition based on the appearance of local regions”, In Proceedings of the 17th International Conference on Pattern Recognition, 2004.

[11] R. Gottumukkal and V.K. Asari, “An Improved Face Recognition Technique Based on Modular PCA Approach” Pattern Recognition Letters, vol. 25, pp. 429- 436, Mar. 2004.

[12] P.-H. Lee, G.-S. Hsu, T. Chen, and Y.-P. Hung. Facial trait code and its application to face recognition. In ISVC ’08: Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II, pages 317–328, Berlin, Heidelberg, 2008.

[13] L. Sirovich; M. Kirby (1987). "Low-dimensional procedure for the characterization of human faces" . Journal of the Optical Society of America .

[14] M. Kirby; L. Sirovich (1990). "Application of the Karhunen-Loeve procedure for the characterization of human faces".

[15] Face Recognition using Eigenfaces and Neural Networks "Mohamed Rizon, Muhammad Firdaus Hashim, Puteh Saad, Sazali Yaacob, Mohd Rozailan Mamat, Ali Yeon Md Shakaff, Abdul Rahman Saad, Hazri Desa and M. Karthigayan " |School of Mechatronics and Engineering School of Computer and Communication Engineering -

Kolej Universiti Kejuruteraan Utara Malaysia, Jalan Kangar-Arau 02600 Jejawi, Perlis, Malaysia' 2006.

[16] R. Meir and G. R¨atsch. An introduction to boosting and Leveraging S. Mendelson and A. J. Smola Ed., Advanced Lectures on Machine Learning, Springer-Verlag Berlin Heidelberg, pages 118–183, 2003.

[17] Kégl, Balázs (20 December 2013). "The return of AdaBoost.MH: multi-class Hamming trees".

[18] K Somashekari , Puttamadappa C2 & DN Chandrappa2, face detection by SMQT features and snow classifier using color information , K Somashekar et al. / International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462 Vol. 3 No. 2 Feb 2011.

[19] Viola, Jones: Robust Real-time Object Detection, IJCV 2001 See pages 1,3.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I– 511. IEEE, 2001.

[21] Goering, Richard (4 October 2004). "Matlab edges closer to electronic design automation world.

[22] MATLAB Programming Language". Altius Directory. Retrieved 17 December 2010.

[23] Programming Arduino Getting Started with Sketches". McGraw-Hill. Nov 8, 2011. Retrieved 2013-03-28.

[24] Edwards, Robert (1987). "Optimizing the Zilog Z8 Forth Microcontroller for Rapid Prototyping" (PDF). Martin Marietta: 3. Retrieved 9 December 2012.

[25] Pritchard, Stephen (1 March 2012). "Raspberry Pi: A BBC Micro for today's generation". ITPRO. Retrieved 15 March 2012.

[26] Arduino. Arduino introduction. 2014 [cited 2015 21 April]; Available from: https://www.arduino.cc/en/guide/introduction.And https://www.arduino.cc/en/Main/Products

[27] Herman, Stephen. Industrial Motor Control. 6th ed. Delmar, Cengage Learning, 2010. Page 251.

[28] Electromechanical Dynamics, Part 1 John Wiley and Sons, Inc. 1968.

[29] Liptak, Bela G. (2005). Instrument Engineers' Handbook: Process Control and Optimization. CRC Press.

[30] Suk-Hwan Suh; Seong Kyoon Kang; Dae-Hyuk Chung; Ian Stroud (22 August 2008). Theory and Design of CNC Systems. Springer Science & Business Media.

[31] Jonathan Knoder (9 May 2013). "1080p, 2.0 Mega Pixels? Understanding Webcam Technical Terms". Top Ten Reviews. Retrieved 29 July 2015.

APPENDIXES

Appendix A: Matlab code

Option and database code

```
imaqreset;
clear all;
close all;
clc;
i=1;
global M;   %input no. of faces
global face_id
times=5;   %no. of pics capture for an individual
N=4;   %default no. of faces
vid = videoinput('winvideo',1,'YUY2_320x240');
while (1==1)
    choice=menu('System Option',...
        'Generate database',...
        'Recogntion and Tracking',...
        'Exit');

    if (choice==1)
        choice1=menu('Face Recognition',...
            'Enter no. of faces',...
            'Exit');
        if (choice1==1)
            M=input('Enter : ');
            preview(vid);
            while(i<((M*times)+1))
                choice2=menu('Face Recognition',...
                    'Capture');

                if(choice2==1)
                    g=getsnapshot(vid);

                    rgbImage=ycbcr2rgb(g);
                    str=strcat(int2str(i),'.jpg');
                    fullImageFileName =
fullfile('C:\Users\User\Desktop\graduation
project\test2\database',str);
                    imwrite(rgbImage,fullImageFileName);


                    grayImage=rgb2gray(rgbImage);
                    Dir_name=fullfile(pwd,str);
                    imwrite(grayImage,Dir_name);
                    i=(i+1);
                end
            end
            closepreview(vid);
        end

        if (choice1==2)
            clear choice1;
        end
    end
    if(choice==2)
        if(isempty(M)==1)
            default=N*times;
            face_id=recognize_face_cam(default);
        else
            faces=M*times;
            face_id=recognize_face_cam(faces);
        end
    end
    if (choice==3)
        close all;
        return;
    end
end
stop(vid);
```

Face recognition code

```
function p1 = recognize_face_cam(M)
imaqreset;

close all
clc

vid = videoinput('winvideo',2,'YUY2_320x240');

um=100;
ustd=80;
person_no=0;
times=5;

S=[];
figure(1);
for i=1:M
    str=strcat(int2str(i),'.jpg'
    eval('img=imread(str);');

    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    if i==3
        title('Training set','fontsize',18)
    end
    drawnow;
    [irow icol]=size(img);

    temp=reshape(img',irow*icol,1);
    S=[S temp];

end
```

```
for i=1:size(S,2)
    temp=double(S(:,i));
    m=mean(temp);
    st=std(temp);
    S(:,i)=(temp-m)*ustd/st+um;
end

figure(2);
for i=1:M
    str=strcat(int2str(i),'.jpg');
    img=reshape(S(:,i),icol,irow);
    img=img';
    eval('imwrite(img,str)');
    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    drawnow;
    if i==3
        title('Normalized Training Set','fontsize',18)
    end
end


m=mean(S,2
tmimg=uint8(m);
img=reshape(tmimg,icol,irow);

img=img'

figure(3);
imshow(img);
title('Mean Image','fontsize',18)


dbx=[];   for i=1:M
    temp=double(S(:,i));
    dbx=[dbx temp];
end


A=dbx';
L=A*A';

[vv dd]=eig(L);

v=[];
d=[];
for i=1:size(vv,2)
    if(dd(i,i)>1e-4)
        v=[v vv(:,i)];
        d=[d dd(i,i)];
    end
end


[B index]=sort(d);
ind=zeros(size(index));
```

```
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
    dtemp(i)=B(len+1-i);
    ind(i)=len+1-index(i);
    vtemp(:,ind(i))=v(:,i);
end
d=dtemp;
v=vtemp;



for i=1:size(v,2)

    kk=v(:,i);
    temp=sqrt(sum(kk.^2));
    v(:,i)=v(:,i)./temp;
end


u=[];
for i=1:size(v,2)
    temp=sqrt(d(i));
    u=[u (dbx*v(:,i))./temp];
end


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



figure(4);
for i=1:size(u,2)
    img=reshape(u(:,i),icol,irow);
    img=img';
    img=histeq(img,255);
    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    drawnow;
    if i==3
        title('Eigenfaces','fontsize',18)
    end
end



omega = [];
for h=1:size(dbx,2)
    WW=[];
    for i=1:size(u,2)
        t = u(:,i)';
        WeightOfImage = dot(t,dbx(:,h)');
        WW = [WW; WeightOfImage];
    end
    omega = [omega WW];
end



preview(vid);
choice=1;
if(choice==1)
    g=getsnapshot(vid);
end
rgbImage=ycbcr2rgb(g);
imwrite(rgbImage,'camshot.jpg');
closepreview(vid);
InputImage = imread('camshot.jpg');
figure(5)
subplot(1,2,1)
imshow(InputImage); colormap('gray');title('Input
image','fontsize',18)
input_img=rgb2gray(InputImage);

InImage=reshape(double(input_img)',irow*icol,1);
temp=InImage;
me=mean(temp);
st=std(temp);
temp=(temp-me)*ustd/st+um;
NormImage = temp;
Difference = temp-m;

p = [];
aa=size(u,2);
for i = 1:aa
    pare = dot(NormImage,u(:,i));
    p = [p; pare];
end
ReshapedImage = m + u(:,1:aa)*p;

ReshapedImage = reshape(ReshapedImage,icol,irow);
ReshapedImage = ReshapedImage';

subplot(1,2,2)
imagesc(ReshapedImage); colormap('gray');
title('Reconstructed image','fontsize',18)

InImWeight = [];
for i=1:size(u,2)
    t = u(:,i)';
    WeightOfInputImage = dot(t,Difference');
    InImWeight = [InImWeight; WeightOfInputImage];
end

ll = 1:M;
figure(68)
subplot(1,2,1)
```

```
stem(ll,InImWeight)
title('Weight of Input Face','fontsize',14)

e=[];
for i=1:size(omega,2)
    q = omega(:,i);
    DiffWeight = InImWeight-q;
    mag = norm(DiffWeight);
    e = [e mag];
end

kk = 1:size(e,2);
subplot(1,2,2)
stem(kk,e)
title('Eucledian distance of input image','fontsize',14)

MaximumValue=max(e)
MinimumValue=min(e)

Min_id=find(e==min(e));
person_no=Min_id/times;
p1=(round(person_no));
if(person_no<p1 && MinimumValue < 3.8e+04)
    p1=(p1-1);
    display('Detected face number :')
    display(p1)
end
if(person_no>p1 && MinimumValue < 3.8e+04)
    p1=(p1+1);
    display('Detected face number1 :')
    display(p1)
recog_trck()
end
if(MinimumValue > 3.8e+04)
    display('error')
end
if(person_no==p1 && MinimumValue < 3.8e+04)
    display('Detected face number :')
    display(p1)
    recog_trck()
end
stop(vid);
end
```

Face Tracking code

```
    function [std_f,mean_f] = recog_trck()
clear all
clc
answer=1;
arduino=serial('COM4','BaudRate',9600);
fopen(arduino);
faceDetector = vision.CascadeObjectDetector();
obj =imaq.VideoDevice('winvideo', 2, 'I420_320x240','ROI', [1
1 320 240]);
```

```
set(obj,'ReturnedColorSpace', 'rgb');
figure('menubar','none','tag','webcam');
wait=0;
while (wait<6000);
    wait=wait+1;
    frame=step(obj);
    bbox=step(faceDetector,frame);
    wait;
    if(~isempty(bbox))
        bbox;
        centx=bbox(1) + (bbox(3)/2) ;
        centy=bbox(2) - (bbox(4)/2) ;
        c1=(centx);
        c2=(centy);
        c1;
        c2;
        fprintf(arduino,'%s',char(centx));
        fprintf(arduino,'%s',char(centy));
    end
    boxInserter  =
vision.ShapeInserter('BorderColor','Custom',...
    'CustomBorderColor',[255 0 255]);
videoOut = step(boxInserter, frame,bbox);
    imshow(videoOut,'border','tight');
    f=findobj('tag','webcam');
    if (isempty(f));
        [hueChannel,~,~] = rgb2hsv(frame);
rectangle('Position',bbox(1,:),'LineWidth',2,'EdgeColor',[1 1
0]);
hold off
noseDetector = vision.CascadeObjectDetector('Nose');
faceImage    = imcrop(frame,bbox);
noseBBox     = step(noseDetector,faceImage);
noseBBox(1:1) = noseBBox(1:1) + bbox(1:1);
videoInfo    = info(obj);
ROI=get(obj,'ROI');
VideoSize = [ROI(3) ROI(4)];
tracker = vision.HistogramBasedTracker;
initializeObject(tracker, hueChannel, bbox);
time=0;
while (time<600);
    time=time+1;
    frame = step(obj);
    time;
    [hueChannel,~,~] = rgb2hsv(frame);
    bbox = step(tracker, hueChannel);
    pause (.2);
end
time;
release(obj);
release(videoPlayer);
        close(gcf)
        break
    end
    pause(0.05);
end
```

```
fclose(arduino);
release(obj);
end
```

Appendix B : Arduino code

```
#include <Servo.h>

// Title:   Auto Pan-Tilt Servo/Cam Control

// Subject: This Sketch receives X,Y coordinates from srial then

//          moves the camera to center of those coordinates.

#define  servomaxx   180   // max degree servo horizontal (x) can turn

#define  servomaxy   180   // max degree servo vertical (y) can turn

#define  screenmaxx   320   // max screen horizontal (x)resolution

#define  screenmaxy   240    // max screen vertical (y) resolution

#define  servocenterx   90  // center po#define  of x servo

#define  servocentery   90  // center po#define  of y servo

#define  servopinx   9   // digital pin for servo x

#define  servopiny   10  // digital servo for pin y

#define  baudrate 9600  // com port speed. Must match your C++ setting

#define distancex 1  // x servo rotation steps

#define distancey 2  // y servo rotation steps

int valx = 0;       // store x data from serial port

int valy = 0;       // store y data from serial port

int posx = 0;

int posy = 0;

int incx = 10;  // significant increments of horizontal (x) camera movement

int incy = 10;  // significant increments of vertical (y) camera movement

Servo servox;

Servo servoy;
```

```
short MSB = 0;  // to build  2 byte integer from serial in byte

short LSB = 0;  // to build  2 byte integer from serial in byte

int   MSBLSB = 0;  //to build  2 byte integer from serial in byte

void setup() {

  Serial.begin(baudrate);        // connect to the serial port

  Serial.println("Starting Cam-servo Face tracker");

 pinMode(servopinx, OUTPUT);   // declare the LED's pin as output

  pinMode(servopiny, OUTPUT);   // declare the LED's pin as output

  servoy.attach(servopiny);

  servox.attach(servopinx);

  // center servos

  servox.write(servocenterx);

  delay(200);

  servoy.write(servocentery);

  delay(200);

}

void loop () {

  while (Serial.available() <= 0); // wait for incoming serial data

  if (Serial.available() >= 4)  // wait for 4 bytes.

  {

    // get X axis 2-byte integer from serial

    //MSB = Serial.read();

    //delay(5);

    LSB = Serial.read();
    //MSBLSB=word(MSB, LSB);

    valx = int(LSB);

    delay(5);

    // get Y axis 2-byte integer from serial

    //MSB = Serial.read();

    //delay(5);

    LSB = Serial.read();

    //MSBLSB=word(MSB, LSB);

    valy = int(LSB);

    delay(5);

    // read last servos positions

    posx = servox.read();

    posy = servoy.read();

    //Find out if the X component of the face is to the left of the middle of the screen.

    if (valx < (screenmaxy / 2 - incx)) {

      if ( posx >= 5/*incx*/ ) posx -= distancex; //Update the pan position variable to move the
servo to the left.

    }

    //Find out if the X component of the face is to the right of the middle of the screen.

    else if (valx > screenmaxy / 2 + incx) {

      if (posx <= 175/*servomaxx-incx*/) posx += distancex; //Update the pan position
variable to move the servo to the right.

    }

    //Find out if the Y component of the face is below the middle of the screen.

    if (valy < (screenmaxx / 2 - incy)) {
      if (posy >= 5)posy += distancey; //If it is below the middle of the screen, update the tilt
position variable to lower the tilt servo.

    }

    //Find out if the Y component of the face is above the middle of the screen.

    else if (valy > (screenmaxx / 2 + incy)) {

      if (posy <= 175)posy -= distancey; //Update the tilt position variable to raise the tilt
servo.

    }

    // Servos will rotate accordingly

    servox.write(posx);

    servoy.write(posy);

  }

}
```