Palestine Polytechnic University

College of Engineering

Raspberry Pi and old personal computer's based face detection and recognition system.

Prepared By

Ameer O. Herbawi Mohammad J. Teeti Sajed Y. Hmeed

Supervisor: Eng. Ayman Wazwaz

Submitted to the College of Engineering in fulfillment of the requirements for the Bachelor degree in Communication and Electronic Engineering

Hebron, May 2017

Palestine Polytechnic University

Collage of Engineering

Department of Electrical Engineering

Project title:

Raspberry Pi and old personal computers based face detection and recognition system.

Team:

Ameer O. Herbawi Mohammad J. Teeti Sajed Y. Hmeed

By the guidance of our supervisor, and by the acceptance of all members in the testing committee, this project is delivered to the Electrical Engineering Department in the College of Engineering, to be as a fulfillment of the requirement of the department for the degree of Bachelor’s in communication and electronic engineering.

Supervisor signature

The head of department signature

Dedication

This work is dedicated to our parents, who taught us that the best kind of knowledge to have is that which is learned for its own sake. Who also taught us that even the largest task can be accomplished if it is done one step at a time.

Acknowledgment

We give to the Palestine polytechnic university, special thanks to the Communication and Electronics Engineering teachers in college of engineering who guided us for the past years, without their support, knowledge and patience, this work would never be done.

A special thanks for our supervisor Engineer Ayman Wazwaz. His sage advice, insightful criticisms, and patient encouragement aided the writing of this thesis in innumerable ways.

We also thanks Engineer Mutaz Jawadeh for the help he aided during our implementation and to the 2012 communications and electronic engineering students who helped us in our testing phases.

Abstract

This project aims to deploy a network. This network consists of group of computers connected together with a microcomputer to which a camera is connected. The system will take images of people, analyze, detect and recognize the human faces using image processing algorithms, thus such a system can serve as a security system in public gathering (e.g. Malls, Universities, airports ...etc.). The system will be able to detect and recognize a human face in different situations and scenarios.

This project will implement “Boosted Cascade of Simple Features algorithm” to detect human faces. “Local Binary Pattern algorithm” to recognize these faces. Raspberry Pi will be the main component for its remarkable capabilities among with the raspberry pi camera for image capturing; all needed programmers are written in python 2 and 3.

The project goals were achieved, an improvement of about 80 % in recognition processing time was gained due to the multiple server approach, load balancing were also achieved to send images over the connected servers.

الملخص

يهدف هذا المشروع لبناء نظامٍ متكامل من الأجهزة الحاسوبية القديمة والحاسوب المصغر راسبيري باي، إضافة لربطه مع كمرا لالتقاط الصور، هذا النظام سيلتقط صوراً ويحدد الأشخاص ويحلل هذا الصور ويتعرف على هؤلاء الأشخاص باستخدام خوارزميات تحديد الوجوه والتعرف عليها، يمكن استخدام هذا النظام كنظام حماية لتحديد الوجوه والتعرف عليها في الأماكن العامة مثل المولات والجامعات والتجمعات السكانية الكبيرة.

سيستخدم هذا النظام خوارزمية "Boosted Cascade of Simple Features" لتحديد الوجوه في الصورة وخوارزمية "Local Binary Pattern" للتعرف على هذه الوجوه، راسبيري باي سيكون المتحكم الأساسي في هذا المشروع، لقدراته المهولة وصغر حجمه. الكاميرا المستخدمة خاصة بمتحكم الراسبيري باي، ولغة البرمجة المعتمدة هي بايثون 2 وبايثون 3.

اهداف المشروع تم تحقيقها ووصلت نسبة النجاح الى 80 % في تحسين سرعة التعرف على الوجوه بفضل استخدام ثلاثة أجهزة مختلفة لتقسيم حمل الصور القادم من الراسبيري باي، إضافةً لتحيق مبدأ توزيع الاحمال عند ارسال الصور للأجهزة الحاسوبية مما يخفف الضغط على الأجهزة المتصلة.

List of Figures

List of Tables

List of Abbreviations

RP	Raspberry Pi
PC	Personal Computer
GPU	Graphics processing unit
CCTV	Closed-circuit television camera
FLD	Fisher Liner Discriminant
SVM	Support Vector Machine
LCD	Liquid crystal displays
PCA	Principle Component analysis

LDA	Linear Discriminant analysis
LBP	Local Binary Pattern
LBPH	Local Binary Pattern Histograms
CAT 5 E	Category 5 Ethernet
Wi-Fi	Wireless Fidelity
LAN	Local Area Network
IEEE	Institute of Electrical and Electronics Engineers
OSI	Open Systems Interconnection
TCP/IP	Transmission Control Protocol/Internet Protocol
RAM	Random Access Memory
PIC	Peripheral Interface Controller
SD Card	Secure Digital card
FOSS	Free and Open Source Software
YCbCr	Luminance; Chroma: Blue; Chroma: Red
CSI	Camera Serial Interface
SoC	System on Chip
OS	Operating System
YML	Why A Markup Language (XML generating language by Volker Birk)
GB	Gigabyte
MAC	Medium Access Control
apt	Advance Package Tool
CV	Computer Vision
I/O	Input/output
GUI	Graphical User Interface
DB	Database
IDE	Integrated Development Environment

Table of Contents

Chapter Three: System Design

CHAPTER ONE

Introduction

Chapter One

Introduction

1.1 Motivation

Digital graveyard has become one of the greatest challenges in this fast growing technical world; according to Daily-mail [1] it’s a terrible waste of space and money, since all the old devices-*we will consider personal computers for the sake of the desired application*-and electronics are thrown away for trivial reasons, such as the availability of a new device or components, thus this does not mean that the old devices are useless, in fact, they are not.

As electronics and communications engineers, we started searching for ways to benefit from the existed technologies to expand the resources of the new technologies instead of tolerating the big cost of creating new matching resources.

Viewing the resources available with the microcomputers, they are tremendous, but, for very complex tasks, microcontrollers would take a long and unnecessary processing time to accomplish its tasks or applications.

1.2 Objectives:

The goal of the project is to build a system that can detect and recognize faces of people using image-processing techniques. Practically this idea can be implemented in large gathering areas (Malls, Airports, and Universities) to provide security.

The main objectives of this project summarized as follows:

1) Benefiting from the unused computers in corporation with microcomputers.
2) Expanding the desired microcomputer capabilities.
3) Implement a face detection and recognition algorithm to run over the microcomputer to notice the effect of combining the computers with microcontrollers all together.

1.3 Overview:

Eventually, every one of us is going to get rid of old computer and throw it away, if a larger scope is consider such as municipalities and universities they do an annual upgrade for the computers that they own. That is a huge aspect of the problem, what if these computers could have been used and embed into a specific application using their quit well resources.

In addition, when observing the somehow restricted capabilities of Raspberry pi -which can be considered as a second problem - and how it could benefited from old computers, while evolving the capabilities that raspberry pi have. By conducting a proposal that combines these two aspects. an inspiration to build a system that consists of raspberry pi and old computers in order to perform a complex application over them.

This project is likely to be a good way to re-use the existing PC along with modern microcomputers and solve the mentioned problems, and it will create a new integrated system, which will save Money, Effort and Time, which are much appreciated in technical field.

1.4 Methodology:

The block diagram in **Figure 1.1** consists of a camera that will capture and forward frames to the Raspberry pi [2], the Raspberry pi then will detect and crop the faces in this frame and transmit the resulted face over Ethernet to one of the available PC's. Meanwhile the PC's are going to recognize the received faces and display the unknown picture on one of the servers and notify the mobile phone (android).

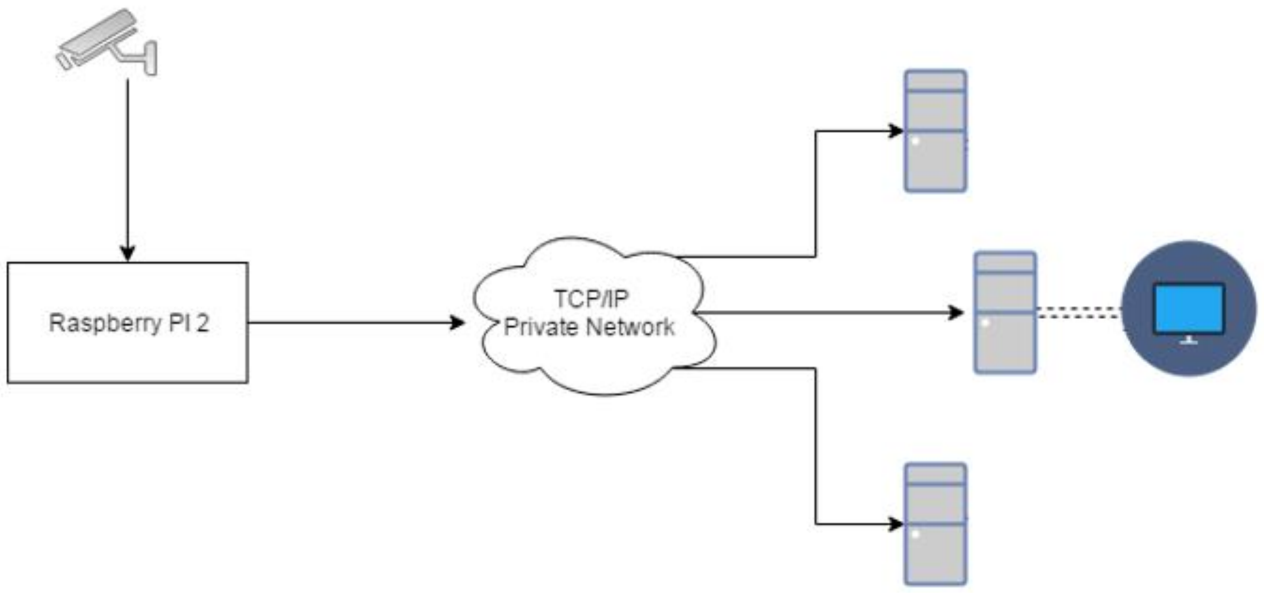


Figure 1.1: General System Block Diagram.

The system works as follows:

1) A camera is connected to the raspberry pi will stream live video to the raspberry pi.
2) By implementing detection algorithm on the raspberry pi, faces of the people should be detected, and cropped so they are ready for transmission to the connected old personal computers.

3) The connection between the raspberry pi and the computers is a TCP/IP based, either as Ethernet or as a wireless connection.
4) A specific computer will receive the cropped face image and run it through a face recognizer algorithm to recognize the face from the database.
5) If the face is unknown, and the system cannot recognize it, a notification message will appear over the computer screen along with the mobile device notification (android application) showing the stranger face.

1.5 Related works

1) Raspberry pi Based Human Face Detection [3], in this paper a face detection system using raspberry pi was developed. The system that they built was Python based, they implemented real time face detection, and were able to detect faces from a specific images, object detection also was considered here, using a variety of data bases to test their result they were able to detect a very foggy and ambiguous faces, as far as we can consider that this is a great work, but the authors didn’t approach face recognition algorithms due to complexity of the recognition process, and since the recognition process will need a more powerful resources to accomplish an excellent result.

2) Graphical processing unit (GPU): Application for Closed Circuit Television Camera (CCTV) systems [4], the authors were able to switch from the CCTV graphical processor to a PC GPU, in this we noticed the trend toward using a PC GPU for its quit well capabilities, a capability better than the one in the CCTV system, they embedded the security cameras into the PC GPU, and we could notice that the PC GPU is an excellent resource to use when generating and displaying the typical content of a CCTV system, image composition and content decoding were not a problem in this case, they were done in an accurate way. What was noticeable that they suggested at the end of their paper saying “if we need a system capable of displaying more source we can increase the power of the PC components, and when we reach a point where it is not possible, we can add a second PC, making this a cheap and scalable solution”, we can refer to this as a good point regarding the re-use of old computers.

3) Accelerating Real-time Face Detection on a Raspberry Pi Telepresence Robot[5], this paper is focusing on accelerating only the face detection process over the Raspberry pi, the authors were able to achieve a good results, thus they had to take the CPU of the Raspberry pi to its limits, and they also needed a little bit more RAM, they implemented a software to accelerate the detection of faces over the raspberry pi, which was better than their detection algorithm in Telepresence robot, they had showed that combining the Raspberry Pi into another microcomputer (Telepresence robot here) could improve the overall system performance.

4) Face Detection, [6] attempted to detect faces in a digital image using various techniques such as skin color segmentation, morphological processing, template matching, Fisher linear discriminant (FLD), Eigen face decomposition, and support vector machines (SVM). It determined that the more complex classifiers did not work as well as expected due to the lack of large databases for training. Reasonable results were obtained with color segmentation, template matching at multiple scales, and clustering of correlation peaks.

5) Reusing personal computer devices - good or bad for the environment? [7]: The energy saving potential of reusing / reselling personal computer (PC) devices was evaluated relative to the choice of buying new. Contrary to the common belief of reuse leading to energy savings, with the advent of more efficient laptops and liquid crystal displays (LCD), reuse of an old personal computer device can lead to relative energy expenditure. We found that in certain scenarios this expenditure could be as large as 300% of the lifecycle energy inventory for the new device. As a result, it is essential to assess the reuse of personal computer devices more critically, incorporating the different factors that influence the analysis.

1.6 Equipment:

1) Raspberry pi 2 version B Microcomputer.
2) Raspberry Pi Camera Module Version 1.3.
3) TP-Link 4-Port Router with Wireless adapter.
4) Ethernet Cables.
5) Group of Unused (Old) computers as servers.
6) Android Mobile phone.

1.7 Estimated Costs:

The following **table 1.1** estimate the total cost for the project

Table 1.1: Estimated Total cost for the project

Component	**Quantity**	**Estimated Total Cost**
Raspberry Pi	1	80$
Camera Module	1	25$
Router	1	30$
Ethernet cables	4	10$
Computers	3	450$
Mobile Phone	1	100$
Total estimated cost for the project		695$

1.8 Timing Diagram:

Table 1.2: First Semester Timing Diagram

Week Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Searching for Topic	■	■														
Collecting Information			■	■	■											
Analyzing						■	■									
Documentation								■	■	■	■	■	■	■	■	■

Table 1.3: Second Semester Timing Diagram

Week Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Implementation	■	■	■	■	■	■	■	■	■	■	■	■	■			
Testing									■	■	■	■	■	■	■	
Documentation							■	■	■	■	■	■	■	■	■	■

CHAPTER TWO

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

This chapter outlines the basic concept behind face detection, recognition algorithms, in addition to the primary aspect of the cropping, transmission and processing of the picture. In this chapter, the theory regarding the algorithms will be fairly described. Moreover, a mathematical overview will be represented when needed.

The first section in this chapter investigates the different types of face detection algorithms, whilst there are many detection algorithms major focusing will be given to static face detection and detection in color images, the previous algorithms will be generally introduced, in addition to the mathematical background of each algorithm when needed. While viewing this section a graphical example will be added to support a better understanding of each algorithm.

Face recognition is the second section in this chapter, in this section an introduction to a three famous face recognition algorithms, Eigenfaces, Fisherfaces, and Local binary patterns histograms will be given, in addition to some historical background whenever its needed to support a better understanding of how each algorithm evolved within time. Major focusing will be given to mathematical background.

The last section in this chapter "System backbone", this name was chosen since this section will provide an overview for the whole backbone of the system. Network part will be explained, the various transmission mediums are discussed, the desired transmission protocol will be discussed, and the theory behind the microcomputers in both hardware – software terms is explained, the hardware components rather than the microcomputer such as the camera, server and mobile phone are also considered.

2.2 Face Detection:

2.2.1 Introduction

Face detection is a technology that has been developed and used to detect human faces for several applications and purposes. In this section derivation and explanation of the three most popular face detection algorithm is provided.

Given an image, the goal of face detection is to determine whether there are any faces in the image and if present, Identify and locate human faces in an image regardless of their position, scale, orientation, pose, and illumination. This process is hard because of Pose Variation, Beards, Glasses, Facial Expression, Image Orientation, Lighting, Noise and Resolution.

In the following sections, a deeper look is given to clarify the different algorithms.

2.2.2 Face Detection using a static or mono color background

This technique is a very simple one that uses image with a mono-color or static predefined background, removing the background will always give the Face area. Simple, yet not efficient because it does not work with different backgrounds, which is the normal case.

2.2.3 Face Detection in color images

In such algorithms the skin color is a key factor for detection process, there are two primary methodologies, first one detection in color images using skin tone color model and detection by a boosted cascade of simple features, these two algorithms are described in the following sections. [8]

2.2.3.1 Detection in Color Images using skin-tone color model

The method is to detect skin regions over the entire image, and then generate face candidates based on the spatial arrangement of these skin patches. The algorithm constructs eye, mouth, and boundary maps for verifying each face candidate. The algorithm provides two modules:

1) Face localization to detect face candidates.
2) Facial features to verify the detected face.

Figure 2.1 represents a graphical clarification for the previous two modules, a color image is presented to the algorithm, then a lighting compression is applied to minimize the external effects, then the model determines the candidates, which are the features of the face, finally, the second model is applied to verify that it’s a face.

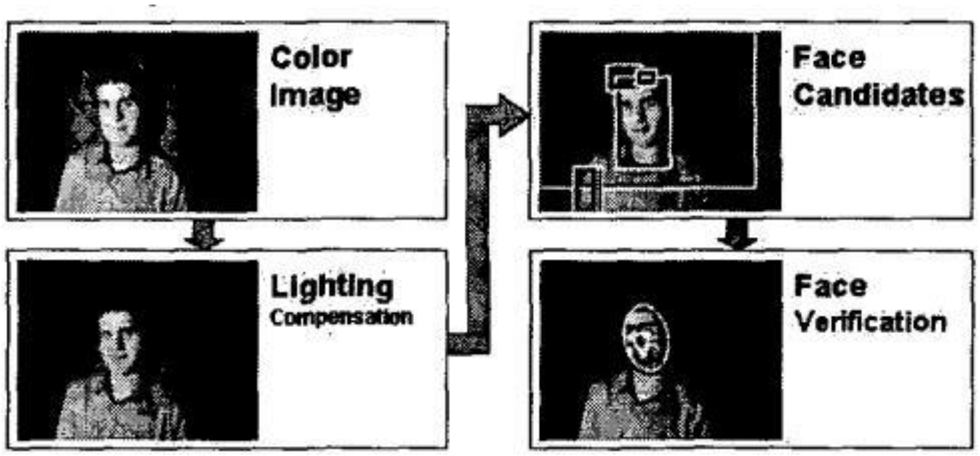


Figure 2.1: The two models for Face detection in color images.

The Luminance; Chroma: Blue; Chroma: Red (YCbCr) color space is used to model skin color and this color space is non-linearly transformed to make the skin cluster independent of luma (non-linear gamma-corrected luminance 0<ɣ <1) **figure 2.2** shows the skin detection.

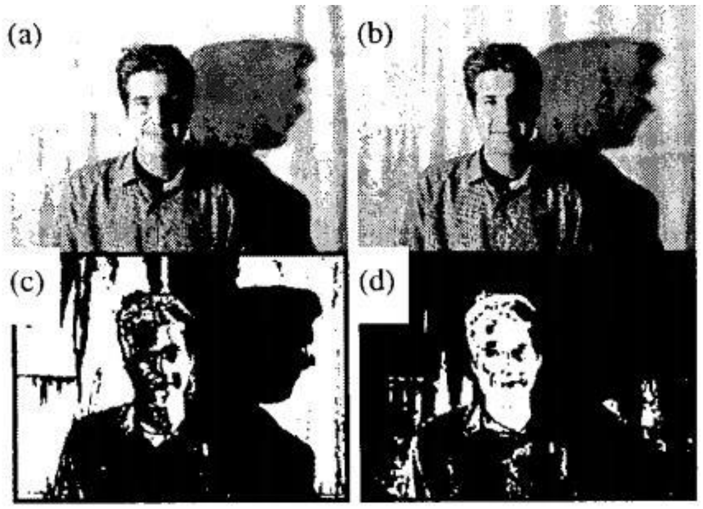


Figure 2.2: Skin Detection **A**) Yellow biased face image.

B) a lighting compensated image.

C) skin of face in A.

D) skin of face in B.

The algorithm uses the facial feature (eyes and nose) as the most important features for detection. It was found that high values of Cb color (blue difference component in the Y`CbCr space) and Low Values of Cr color (red difference component in the Y`CbCr space) are around the eyes and so the algorithm estimates an Eye Map by using the value of Cb, Inverse of Cr and the ratio of Cb/Cr. **Figure 2.3** estimates and locates the eyes.

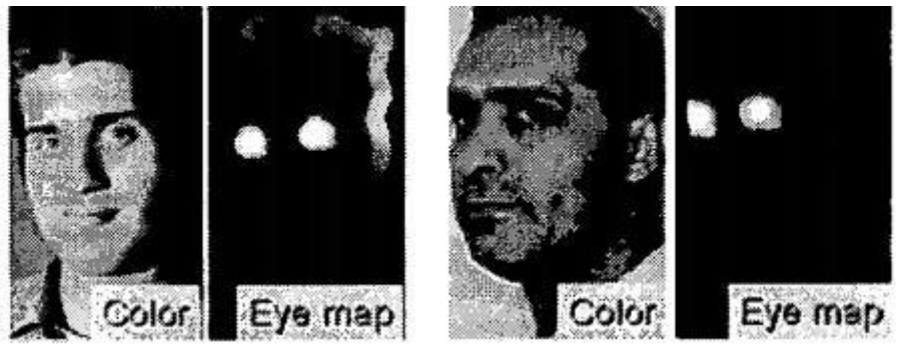


Figure 2.3: Locating and identifying the eye location for a two different face pose, and color.

Since the mouth has more red component than blue component then Cr > Cb and also it was noticed that there is a relatively low response in Cr/Cb and high response in Cr^2.hence the difference between Cr^2 and Cr/Cb can determine the mouth region as shown in **figure 2.4**.

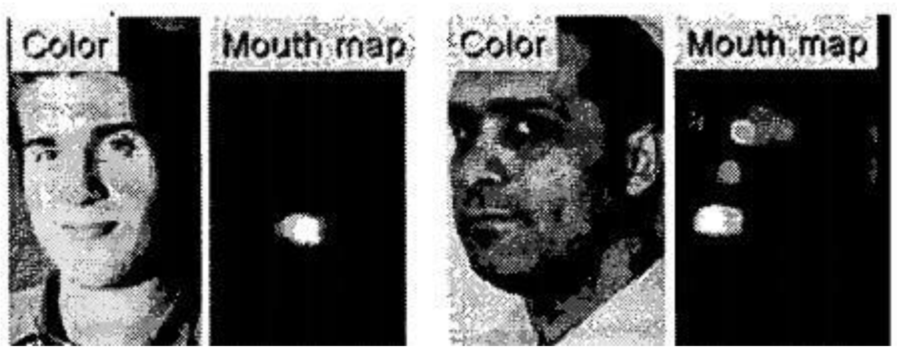


Figure 2.4: Construction of the mouth maps for two subjects.

Then, verifying eyes and mouth by using:

1) Luma variations of eye and mouth blobs.
2) Geometry and orientation constraints of eyes-mouth triangles.
3) The presence of a face boundary around eyes-mouth triangles.

2.2.3.2 Detection by a boosted cascade of simple features [9]

This algorithm provides a very rapid object detection depending on three main key contributions, which are:

1) The concept of "Integral Images", which introduces a new way of image representation that makes the detector capable of computing the features more quickly.
2) AdaBoost based learning algorithm, which can select few but very critical features for calculation from a huge number of features, and so exclude unwanted features that appear in the background.

3) A method to cascade the calculation of complex features. The Cascading method will allow the system to spend more time in processing the important features while discarding unwanted background features rapidly.

In the approach a group of reminiscent features is used to help detect a human face, Image Integral does the features evaluation.

The first approach that algorithm takes is the image integral.

<u>Image Integral</u>

An Image representation in which the integral can be computed from image using a few number of operations per pixel and then the features are evaluated using Harr-filter (group of basis that are used to represent a portion of an image as follows:

Let P(x,y) is the value of a pixel in an image ,then in image integral the value of a pixel is the sum of all pixels above and to the left of (x1,y1), **Figure 2.5** represents an integral system.

$P(x1, y1) = \sum_X \sum_Y p(x, y)$ Equation (2.1)

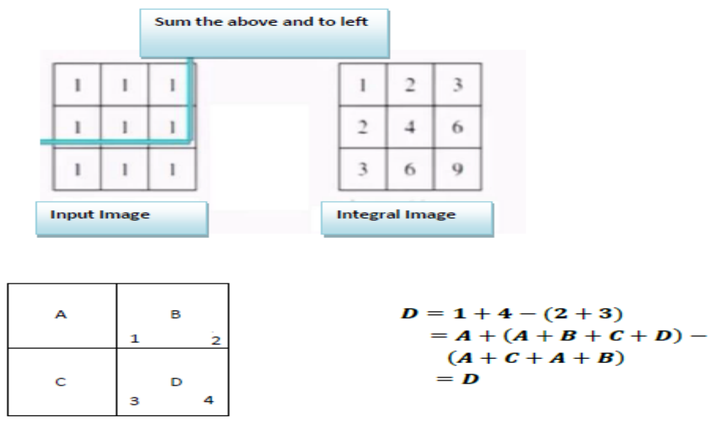


Figure 2.5: Example of integrating an image.

Applying the features to the image, subtracting the pixel values under white region from pixel values under black region, and the result is a single value.

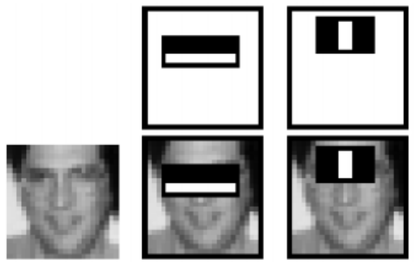


Figure 2.6: Harr Features used to compare with the calculated features after integral image representation.

For example, **Figure 2.7** shows a nose feature, applying the feature all over the image will give a lower value everywhere except for the nose region, then the highest value will be obtained there.

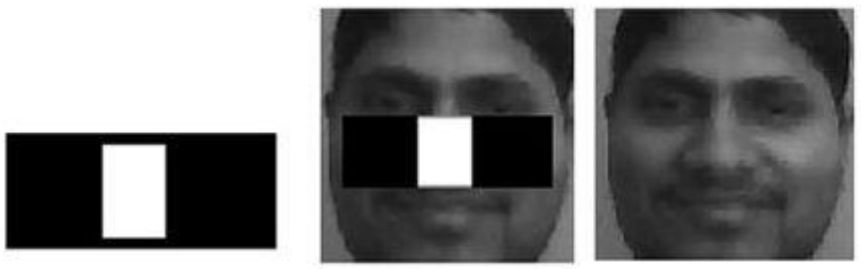


Figure 2.7: Nose Harr-Like Feature.

Because of the large number of Harr-like features in a specific sub window in the image, the calculation will be tedious and time consuming, so the AdaBoost technique is used to reduce the number of the features by excluding the (non-facial features) which are not important and carry no information.

The second approach that the algorithm takes to improve the detection and remove the unneeded features is the AdaBoost

The AdaBoost

A learning algorithm that helps finding the most critical and best features among all features. After eliminating the irrelevant features, a weighted combination of all relevant features is used to decide whether the given window has a face or not, each feature is called a weak classifier and AdaBoost algorithm build a strong classifier as a linear combination of the weak classifiers.

$$F(x) = a_1 f(x) + a_2 f(x) + a_3 f(x) + \cdots \qquad \text{Equation (2.2)}$$

Applying the strong AdaBoost classifiers to all given sub windows in an image is also time consuming and may limit the real time process of detection, the algorithm solves the problem by involving the Cascading method.

The last feature added to the algorithm to improve detection speed is the cascading.

Cascading:

In cascading technique, the strong classifiers are divided into stages each contains a strong classifier. Each stage is used to determine whether the given sub window is a face or not, if the sub window fails in any of the stages then it's immediately discarded as non-face and another sub window is processed, by that method time can be preserved to test for faces in the windows that are most likely to have face.

2.3 Face Recognition:

2.3.1 Historical background

Since the early of 50’s, facial recognition had gain its important due to the severe attacks on buildings, airports in different European countries, it was imminent to develop a system to help the law in identifying and recognizing the suspect, and match it among a library of pre-known suspect faces. Therefore, face recognition importance appeared since it doesn’t require interaction between the system and the suspect, this system simply uses the image to compare and match it with a pre-known faces and if the face appeared to be known, an action can be taken after. This technology was monopolized for the government until it was commercially available in 1990s. [10]

2.3.2 Introduction:

Face recognition systems are the new trend in the world of biometric security, we will introduce some of the most important and recent face recognition algorithms as we will look into the core of each algorithm, a description will be provided in addition to the mathematical background when needed.

In the following sections, we will research into the (Open Source Computer Vision Library) OpenCV Libraries, OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

In this section we discuss Eigenfaces, Fisherfaces and local binary pattern histogram algorithms, each of the above algorithms is supported by the OpenCV library, however we will choose one of these algorithms to work on considering many factors: efficiency, stability and complexity are a primary interest.

2.3.3 Eigenfaces

Eigenfaces is the first algorithm that is introduce and discussed here, it is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. Eigenfaces refers to an appearance-based approach of face recognition that seeks to capture the variation in a collection of face images and use these variations information to compare images of individual faces in a feature-based manner. Specifically, the Eigenfaces are the principal of a distribution of faces, or equivalently, the eigenvectors of the covariance of the set of face images, where an image with pixels is considered a point (or vector) in -dimensional space. Eigenfaces is still often considered as a baseline comparison method to demonstrate the minimum expected performance of such a system [14]**.**

Eigenfaces is a non-zero vector that does not change its direction when the linear transformation is applied to it.

2.3.4 Fisherfaces

Fisherfaces algorithm is a second recognition algorithm that is discussed in this section, in the last section, Eigenfaces algorithm was introduced, which was implemented based on Principle components analysis (PCA), a linear combination of features that maximizes the total variance in data. While this is clearly a powerful way to represent data, it does not consider any classes and so a lot of discriminative information may be lost when throwing some components away. This can yield bad results, especially when it comes to classification. [15]

Fisherfaces is similar to Eigenface but with improvement in better classification of different classes image. The training set can be classified to deal with different people and different facial expressions. Better accuracy can be achieved in facial expression than Eigen face approach. Besides, Fisherfaces removes the first three principal components, which are responsible for light intensity changes; it is more invariant to light intensity.

2.3.5 Local Binary Patterns histograms

Eigenfaces and Fisherfaces take a somehow holistic approach to face recognition. The data is being treated as a high-dimensional image space, which is not the best approach we can take since high-dimensional space always requires more attention that is mathematical in addition for its huge size, which leads to high storage consumption. [16]

Therefore, a lower-dimensional subspace is identified, where useful information is preserved. The Eigenfaces maximize the total scatter, which can cause some trouble if the variance is generated by external source as mentioned before. Moreover, some of the components with a maximum variance over all class aren’t necessarily useful for classification.

The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighbor. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. it will end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels it will end up with $2^{\wedge 8}$ possible combinations, called Local Binary Patterns or sometimes referred to as LBP codes.

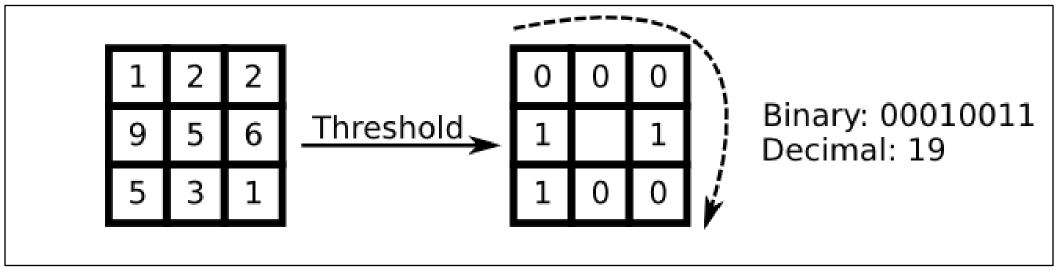


Figure 2.8: This figure illustrate how local binary patterns are generated.

2.3.5.1 Mathematical Background

Local binary patterns are much more efficient than the previous two algorithms; the following steps will provide an insight to how this algorithm works.

1. A more formal description of the LBP operator can be given as

$$\mathrm{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c) \qquad \text{Equation (2.3)}$$

   With (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the neighbor pixel

2. s is the sign function which is defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \qquad \text{Equation (2.4)}$$

   This description enables to capture very fine details in images.

3. The idea is to align an arbitrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods as figure 2.9 shows.

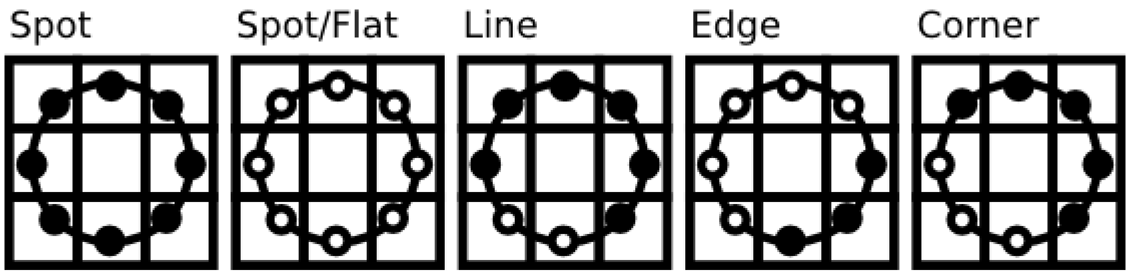


Figure 2.9: This figure illustrates the arbitrary number of neighbors.

4. For a given Point (x_c, y_c) the position of the neighbor $(x_p, y_p), p \in P$ can be calculated by:

$$x_p = x_c + R\cos\left(\frac{2\pi p}{P}\right)$$
$$y_p = y_c - R\sin\left(\frac{2\pi p}{P}\right)$$

Equation (2.5)

   Where R is the radius of the circle and P is the number of sample points.

5. The operator is an extension to the original LBP codes, so it is sometimes called *Extended LBP* (also referred to as *Circular LBP*). If a point coordination's on the circle does not correspond to image coordinates, the point is interpolated as follows.

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

Equation (2.6)

6. Incorporate the spatial information in the face recognition model. By dividing the LBP image into m local regions and extract a histogram from each. The spatially enhanced feature vector is then obtained by concatenating the local histograms these histograms are called Local Binary Patterns Histograms.

2.4 System backbone

This section will generally introduce the basic component of the system, networking side in addition to the raspberry pi and its camera module, the server operating system, and the mobile phone and its application, a description for each component will be stated with the appropriate configuration to be used.

In the first section the network is discussed, what sort of communication methods is used, network size, and the transmission medium. Second section will talk about raspberry pi and its camera module, a features list will be provided, and a description for the raspberry pi itself along with its camera module. Next, a discussion for the server and the raspberry pi operating system is stated among the preferable choices to be made.

2.4.1 Network infrastructure

The network configuration is an important aspect, among the various communication methods such as wired, wireless communication, which can be further divided to Ethernet lines, coaxial cables, fiber line as for the wired side, and to wireless fidelity (Wi-Fi) or the Bluetooth connection in the wireless communication.

Ethernet was a preferred due to its simplicity regarding implementing and using among the system in addition to the more than excellent data rate of 100 Mb/s for Cat5E. [17]

Ethernet is a family of computer networking technologies commonly used in local area networks (LANs) and metropolitan area networks (MANs. It was commercially introduced in 1980 and first standardized in 1983 as IEEE 802.3, and has since been refined to support higher bit rates and longer link distances.

The original 10BASE5 Ethernet uses coaxial cable as a shared medium, while the newer Ethernet variants use twisted pair and fiber optic links in conjunction with hubs or switches. Over the course of its history, Ethernet data transfer rates have been increased from the original 2.94 megabits per second (Mbit/s) to the latest 100 gigabits per second (Gb/s).

The Ethernet cables used in this project are Cat5E, which could serve up to 100Mbit/s transmission rates in stable networks.

The second component in our network is the router that is going to connect the different servers to the raspberry pi, and connect the android phone via wireless connection, a four-port router will be appropriate for this implementation.

2.4.1.1 TCP/IP Protocol

TCP/IP is the basic communication language or protocol of the Internet.TCP/IP is a two-layer program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination.

Each computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they will be reassembled at the destination.

TCP/IP uses the client/server model of communication, in which a computer user i.e. client requests a service (such as requesting a Web page) from another computer i.e. server in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point or host computer in the network to another point or host computer. [18]

2.4.2 Microcomputers

A microcomputer is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of flash is often included on chip, as well as typically small amount of RAM. Microcomputers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications [19].

2.4.2.1 Classifications: [20]

1) Microcontroller: it includes several types such as PIC, Arduino etc. It has an embedded system consists of RAM, Input/output Pins and an oscillator that operates from 16-80 MHz
2) Microcomputer: Includes several types, such as Raspberry Pi, Beagle Bone. It consists of internal clock, Input/output Pins and operates at frequency from 700MHz to 1GHz.

Table 2.1 provide a comparison between raspberry pi an Arduino.

Table 2.1: Comparison between Raspberry Pi (B) & Arduino

Model	**Arduino**	**Raspberry Pi V2 Model B**
Data Rate	16-20 MHz	700-1000 MHz
Random Access Memory	2 KB	512 MB
Read Only Memory	32 KB	External SD Card
USB Port	-	4
Ethernet	-	10/100
Programming Language	Set of C/C++	Any language supported by OS

2.4.2.2 Raspberry Pi Microcomputer

The Raspberry pi shown in **figure 2.10** is a single-board computer developed in the UK by the Raspberry Pi Foundation. The Raspberry Pi is a credit-card sized computer that plugs into the TV and a keyboard. It’s a capable little PC which can be used for many of the things that a desktop PC does, like spreadsheets, word-processing and games. The design does not include a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage. This board is intended to run Linux kernel based operating systems [21].

Figure 2.10: Raspberry pi 2 Model B Microcomputer.

2.4.2.3 Software background [22]

The main module of the system is the microcomputer. In order to work it needs an operating system. The operating system is the backbone, which will run the microcomputer, in addition, a programing language is needed to give the microcomputer the ability to process, send, and receive data. The programming language used within the raspberry pi called "Python".

The operating systems for the raspberry pi are various. Linux, Raspbian, and Windows 10 are the most popular operating systems; a flexible operating system like Raspbian will be used.

The most popular programming language among Raspberry Pi is Python. It is one of those rare languages, which is both simple and powerful. It is an easy to learn language. It has efficient high-level data structures and a simple but effective approach to object oriented programming. Python's elegant syntax and dynamicity, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python has many features such as:

1) Simple: Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudocode nature of Python is one of its greatest strengths. It allows you to concentrate on the solution of the problem rather than the language itself.

2) Easy to Learn: Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

3) Free and Open Source: Python is an example of a FOSS (Free and Open Source Software). In simple terms, anyone can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FOSS is based on the concept of a community, which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

4) Portable: Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. Python programs can work on any of these platforms without requiring any changes at all. Python can be used on Linux, Windows, FreeBSD, and Macintosh.

5) Extensive Libraries: The Python Standard Library is huge indeed. It can help to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, graphical user interfaces (GUI).

2.4.3 Camera Module

The Raspberry Pi camera module is used to take high-definition video, as well as still photographs. There are many examples online of people using it for time-lapse, slow motion and other video cleverness. The module has a five megapixel fixed-focus camera that supports 1080/30, 720/60 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. The camera works with all models of Raspberry Pi 1 and 2. [23]

2.4.4 Servers

A server is a computer program or a machine that waits for requests from other machines or software (clients) and responds to them. The purpose of a server is to share data or hardware and software resources among clients. This architecture called the client–server model. The clients may run on the same computer or may connect to the server over a network. Typical computing servers are database servers, file servers, mail servers, print servers, web servers, and application servers.

The term server is used quite broadly in information technology. In theory, any computerized process that shares a resource to one or more client processes is a server. So, while the existence of files on a machine does not classify it as a server, if it uses some mechanism to share these files then it can be a file server. Similarly, web server software can be run on any capable computer, and so a laptop or personal computer can fulfill the role of a web server, the purpose of a server is to share data. A server computer can serve its own computer programs as well.

Servers that will be used must be at least dual core processer, with two giga bytes of RAM, this specification are the minimum required in order for effective processing time since these servers will process the images from the raspberry pi and recognize faces.

2.4.5 Android smart phone and application

In this configuration an android phone will be used as a second notification device, this android smart phone will be part of the system as it will be programmed to receive and view the faces

The android application is Java based, which can be written with the help of android integrated development environment (IDE); meanwhile the server will trigger a TCP/IP connection based on a code to transmit the faces to the android application.

2.5 Conclusion

After viewing the different types of face detection algorithms, face recognition algorithms and the system backbone, an initial decision is to work with the following configurations:

1) Face detection algorithm: Detection by a boosted cascade of simple features.
2) Face recognition algorithm: Local Binary Patterns Histograms.
3) Raspberry Pi version 2 model B.
4) Raspberry Pi Camera Module 2.
5) Three servers.
6) Android smart phone

These initial setups will be used to develop the next chapter, where we classify each component task and how the whole system communicates and process the image in between.

CHAPTER THREE

System Design

Chapter Three

System Design

3.1 Introduction

In the previous chapter, the system was viewed in mathematical terms, a description was given for the different system components. Thus, each component was treated independently. If the system were to be dealt with as a set of sub systems such as the camera, raspberry pi and the server, the view will be different, since each sub system will process the input information to a suitable output form for the next stage. This chapter introduces the functionality of the system component integrated all together.

In this chapter major focusing will be given to the way the components interact and generate the suitable form of output for the next stage, this will ease a better understanding of system functionality away from the complicated mathematical formulas.

The First section in this chapter discusses the interaction between camera and the Raspberry Pi, and how they work. Block diagram and a flow chart are provided in order to understand how the frame that the camera captures will be delivered to the raspberry pi; the specifications for the captured images and the duration between different images will be provided.

From there, the next section will proceed with how the raspberry pi processes the frame and apply the detection algorithm, a block diagram in addition to the flow chart for this sub system and the algorithm will be combined to have a clear overview for this part.

The processing regarding the server will be discussed in the third section, how each server will process the images and apply the recognition algorithm, a block diagram in addition to the flow chart are provided, explanation to the way that the servers process the images and then display the faces into a different display is also provided.

Concluding with the last section, general block diagram and the general system flow chart, in addition to some theoretical scenarios in which specific configurations are considered in order to notice how different configuration regarding hardware – software may affect the system capabilities, processing time and efficiency.

3.2 The Camera and Raspberry pi

In this section, an explanation regarding the camera and communication with the raspberry pi is given, the way that the images are captured and how they are transmitted to the raspberry pi is also discussed.

Initially, the camera is connected to the raspberry pi via a special serial port dedicated for the camera on the raspberry pi module. What follows that this camera must be programmed and configured over the raspberry pi to run properly, in the configuration terminal many parameters can be set, such as resolution, frame rate, and delay between captured images.

The minimum image resolution allowed is 64 x 64 pixel, and the maximum camera resolution is 2592 x 1944 pixel with a maximum frame rate of 32, the transparency of the captured image and the brightness - which is an important factor when it comes to detect the faces in the next step - are also configurable parameters. [24]

The camera will stream live video to the raspberry pi, in order for the detection algorithm to work with this stream.

3.2.1 Block Diagram

Figure 3.1 shows how the Raspberry pi camera module will be connected to the serial port on the raspberry pi microcomputer, this schematic illustrates the following:

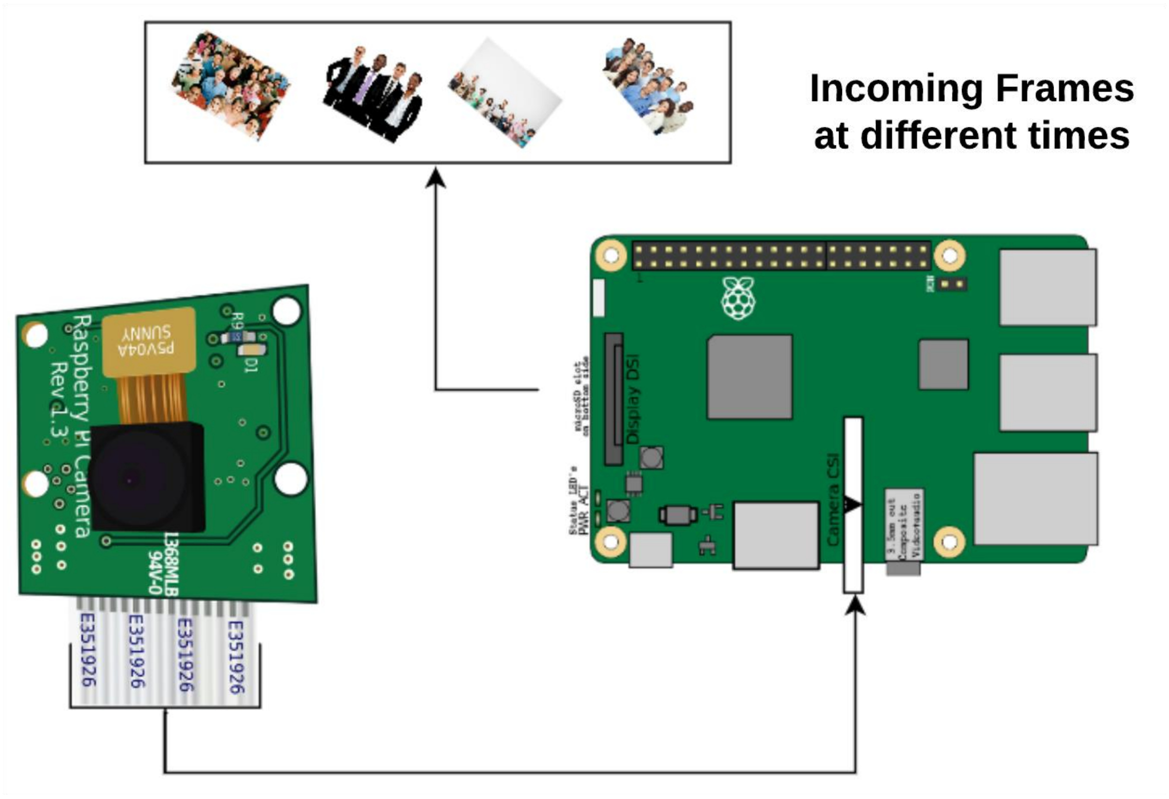

Incoming Frames at different times
Raspberry Pi Camera Rev 1.3
Display DSI
Camera CSI

Figure 3.1: Raspberry Pi Microcomputer and Camera Module in Schematic view.

1) Programming the camera - after physically connecting it to the raspberry pi bored – to initialize.
2) A Python code will configure the camera to stream video.
3) Raspberry pi will read each frame in this video and apply the detection on each frame.
4) Whenever a face present in the frame, the raspberry pi will detect and crop this face and save it on a folder on raspberry pi.

After the camera initialization and as the first frame arrives, the raspberry pi will start processing the incoming frames. Each frame will run through face detection algorithm, and faces, if existed will be detected and cropped.

At the end of this stage, multiple captured faces will be stored on the SD card placed within the Raspberry pi; these faces were taken every time the camera detected them in the frames.

3.2.2 Flow Chart

Figure 3.2 shows the flow chart for the camera and raspberry pi sub system, first the camera is initialized with the frame rate, resolution, and other configurations. Then a live streaming video starts to transmit directly to the raspberry pi, this will take place forever until supervisor terminate the camera for any reasons.

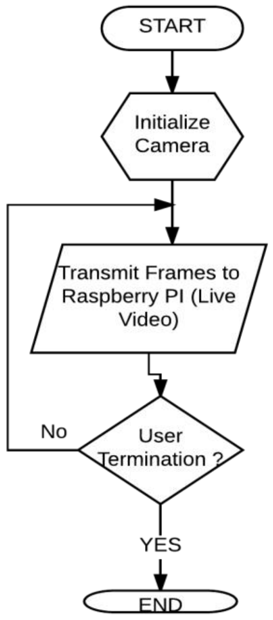


Figure 3.2: Raspberry Pi and Camera Flow chart.

3.3 The Raspberry pi and servers.

In this section the incoming frames in the raspberry pi memory are processed with the detection algorithm, the raspberry pi applies the algorithm to each frame, one at a time.

<u>**Step A**</u>: After the Raspberry pi detects the faces, it is taken into consideration that for a better recognition stage, it is better for the algorithm to work with the face image only. To do so a cropping procedure is done, so that each frame that has many faces will be marked with a rectangular block, and then each block is cropped to form group of faces that will be transmitted to the servers.

<u>**Step B**</u>: Since there are three servers available for the recognition process, each face will be transmitted to a server at once. For example, if the captured picture has six faces in it, the raspberry pi shall detect and crop them and send two faces to each server, this will make sure a better load balancing when it comes to large crowded frames.

In order for images to be transmitted, a session between the raspberry pi and the server must be created, for the three servers, three sessions are needed; to do so, socket programming which involves opening a session from the Raspberry pi to the servers is needed.

Programming the sockets into the raspberry pi will be the first stage, to associate a session to each server, and then transmit the available cropped faces to the servers, face after face over each session to achieve load-balancing criteria.

3.3.1 Block Diagram

Camera and raspberry pi block diagram in **figure 3.3** shows that the camera started to stream live frames to the raspberry pi. This stream of frames is handled frame by frame on the raspberry pi each frame is processed by looking for the faces in it, after a face is detected the raspberry pi will crop this face and write it on the SD card to be ready for transmission to the connected servers.

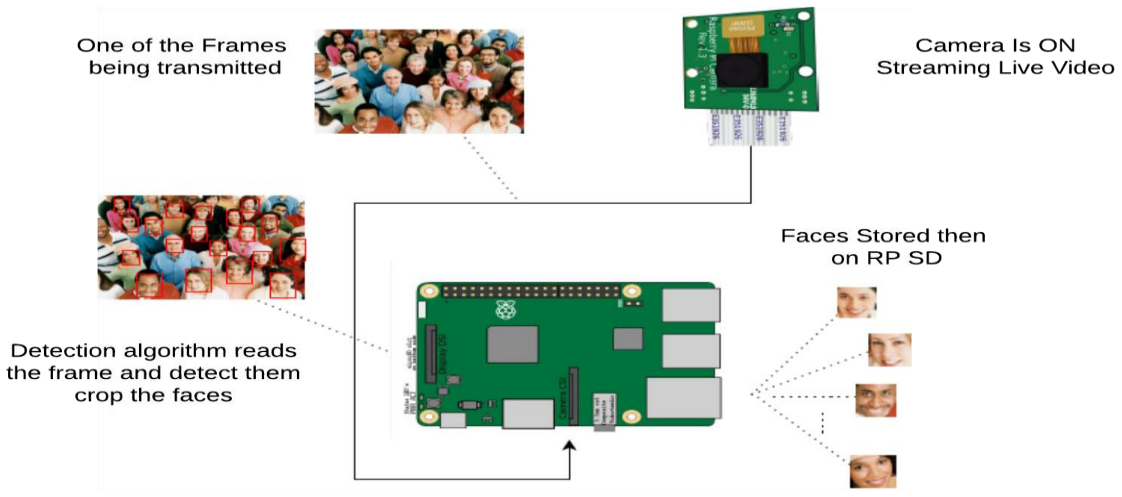

One of the Frames being transmitted
Camera Is ON Streaming Live Video
Faces Stored then on RP SD
Detection algorithm reads the frame and detect them crop the faces

Figure 3.3: Block Diagram illustrating Raspberry Pi input and output.

3.3.2 Flow Charts

Flow chart in **figure3.4** represents how the raspberry pi starts handling the incoming frames, the Face detection algorithm is then applied, and if a face is detected the output will be the same image with rectangle around the face. Applying the cropping code will result in cropped faces, that will be transmitted to each server one at a time, till the faces are done, if a new frame is available, the process is repeated.

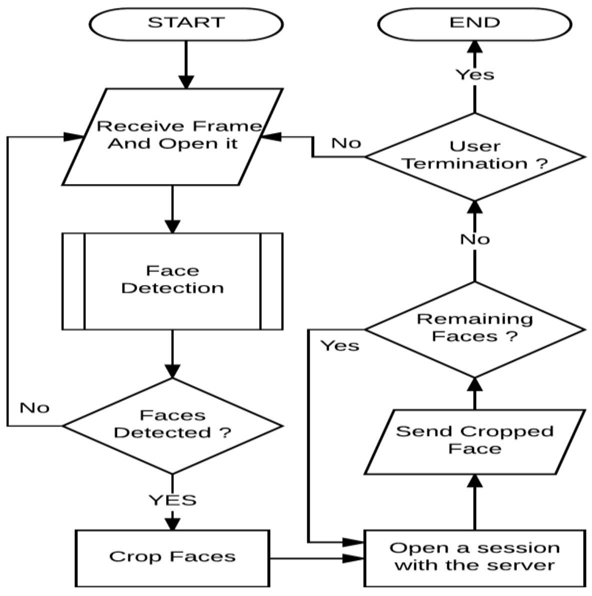


Figure 3.4: Flow chart for face detection and cropping phase.

The following Flow chart in **figure 3.5** illustrates face detection by a boosted cascade of simple features.

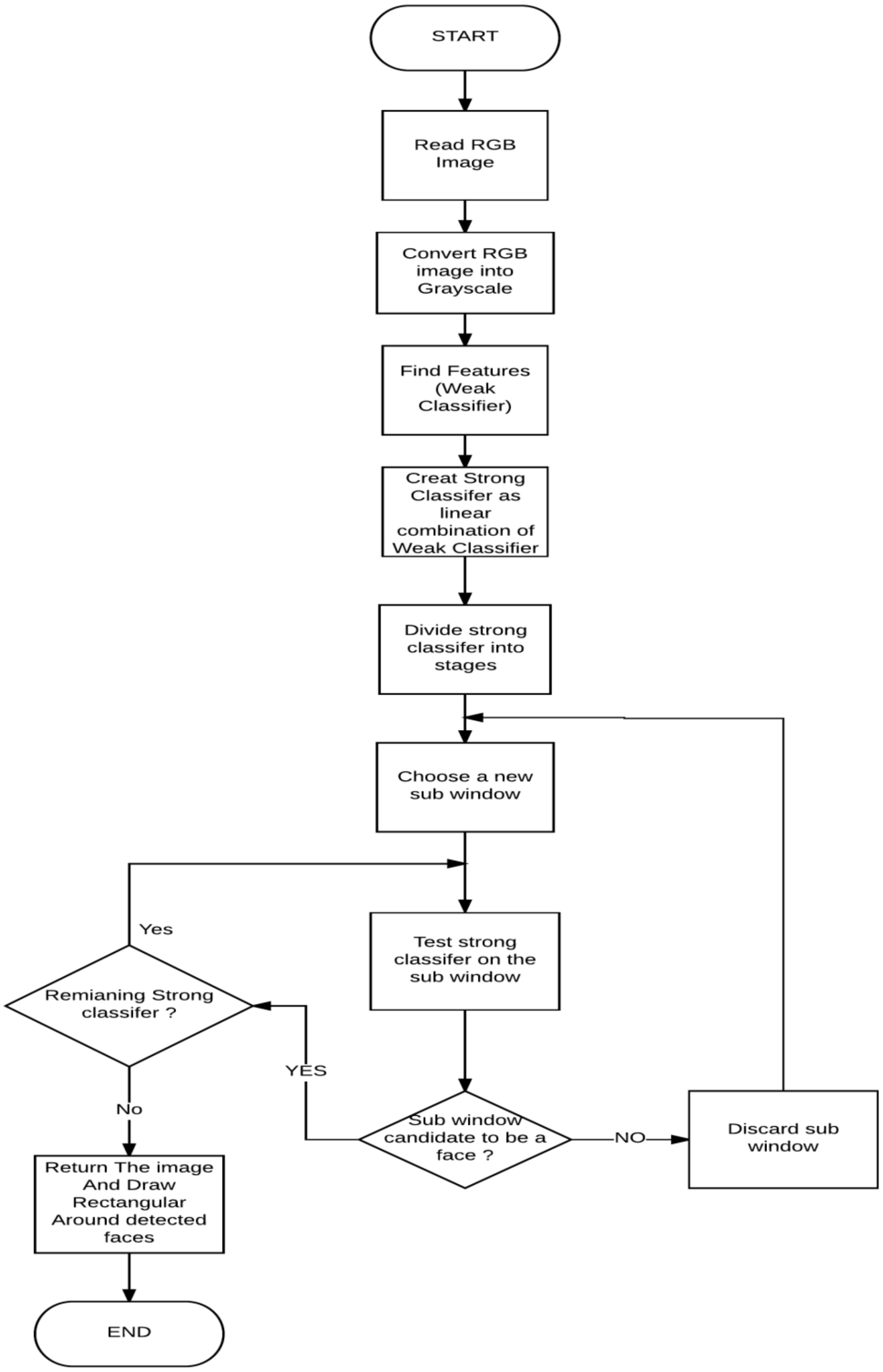

START
Read RGB Image
Convert RGB image into Grayscale
Find Features (Weak Classifier)
Creat Strong Classifer as linear combination of Weak Classifier
Divide strong classifer into stages
Choose a new sub window
Test strong classifer on the sub window
Sub window candidate to be a face ?
NO
Discard sub window
YES
Remianing Strong classifer ?
Yes
No
Return The image And Draw Rectangular Around detected faces
END

Figure 3.5: Flow chart illustrating face detection by a boosted cascade of simple features.

3.4 Servers Subsystem (Recognition stage)

In this section the final process, recognition is described, how the servers receive the faces and start recognizing them, and how each server handles the received faces according to the recognition algorithm “Local Binary Pattern”.

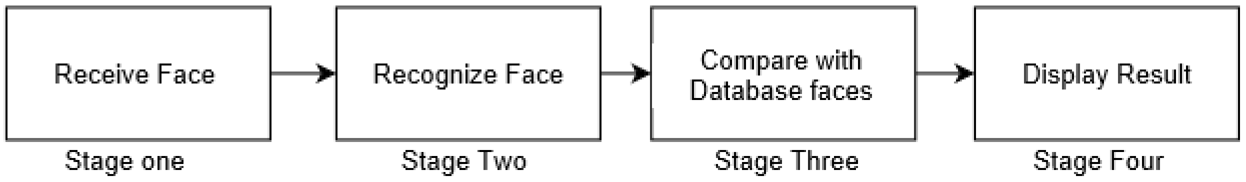


Figure 3.6: Functional Block Diagram for face recognition.

First stage in **figure 3.6**, the server will receive the face throughout the socket that were opened in the previous detection stage, then apply the recognition algorithm, since the faces were cropped in the last stage, it will be much easier to recognize a face among the faces stored in the database.

The second stage, the face will be processed by the Local Binary Pattern Histogram, the output of this stage is the local binary pattern histogram which will be compared later with the local binary pattern histogram stored in the data base.

In the third stage, a comparison between the resulted face and the database faces set will be accomplished through the LBP algorithm. **The last stage** will display the result on the supervisor devices, and will notify the mobile phone if the supervisor is currently away.

The database consist of names and images, with each name linked to a set of images, and a unique ID. Initially nine images for each face, but since LBPH is used, the number of training set images will heavily depend on the testing results in chapter five.

The database will be unified for all of the servers, so all the servers will have a direct connection to the database. Initially the database will be placed in three servers, thus one of the servers will have a primary database that if it has to be updated, the rest of the databases will be updated.

3.4.1 Block Diagram

The block diagram in **figure 3.7** illustrates how to achieve the final process. The servers will receive the images one by one, and apply the recognition algorithm, the result is viewed in the dotted line were the database has three faces only, and since six faces were received, the screen is viewing the server's result if known; the name is displayed, if not a flag will appear over each unknown face.

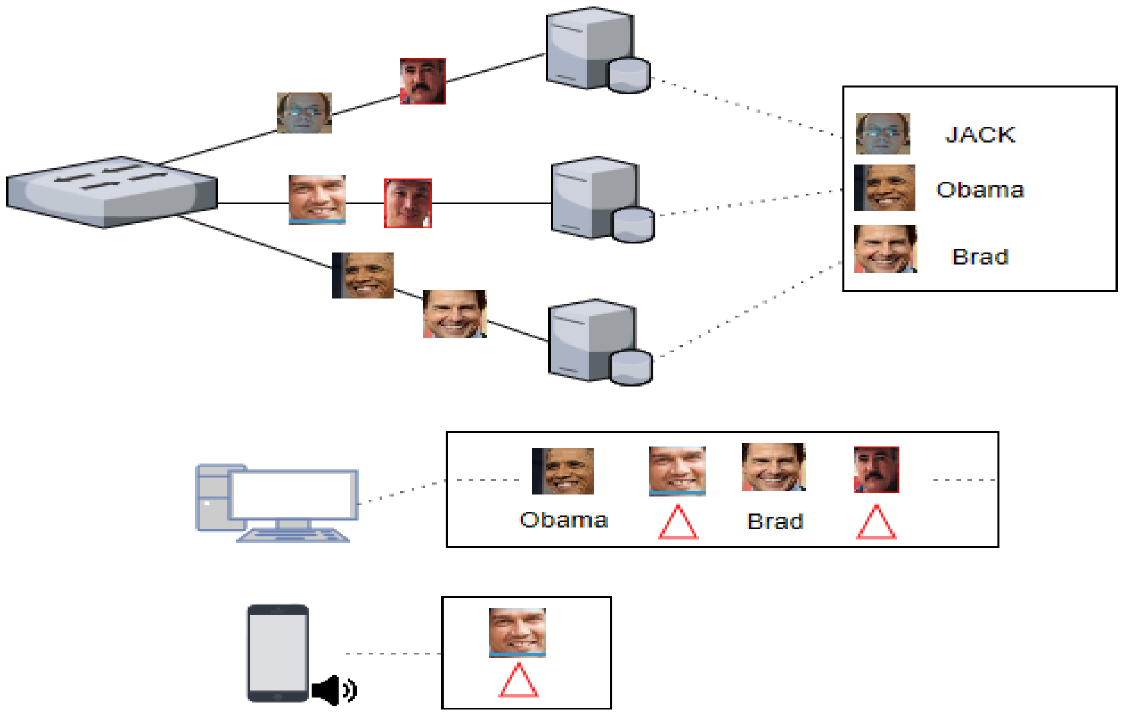


Figure 3.7: Block Diagram illustrating final recognition stage.

Notifying the mobile with the unknown faces only, in addition to notification sound to attract the supervisor attention.

3.4.2 Flow charts

Flow Chart in **figure 3.8** starts when the server receives the faces and apply the recognition process until all the faces are either recognized or flagged unknown.

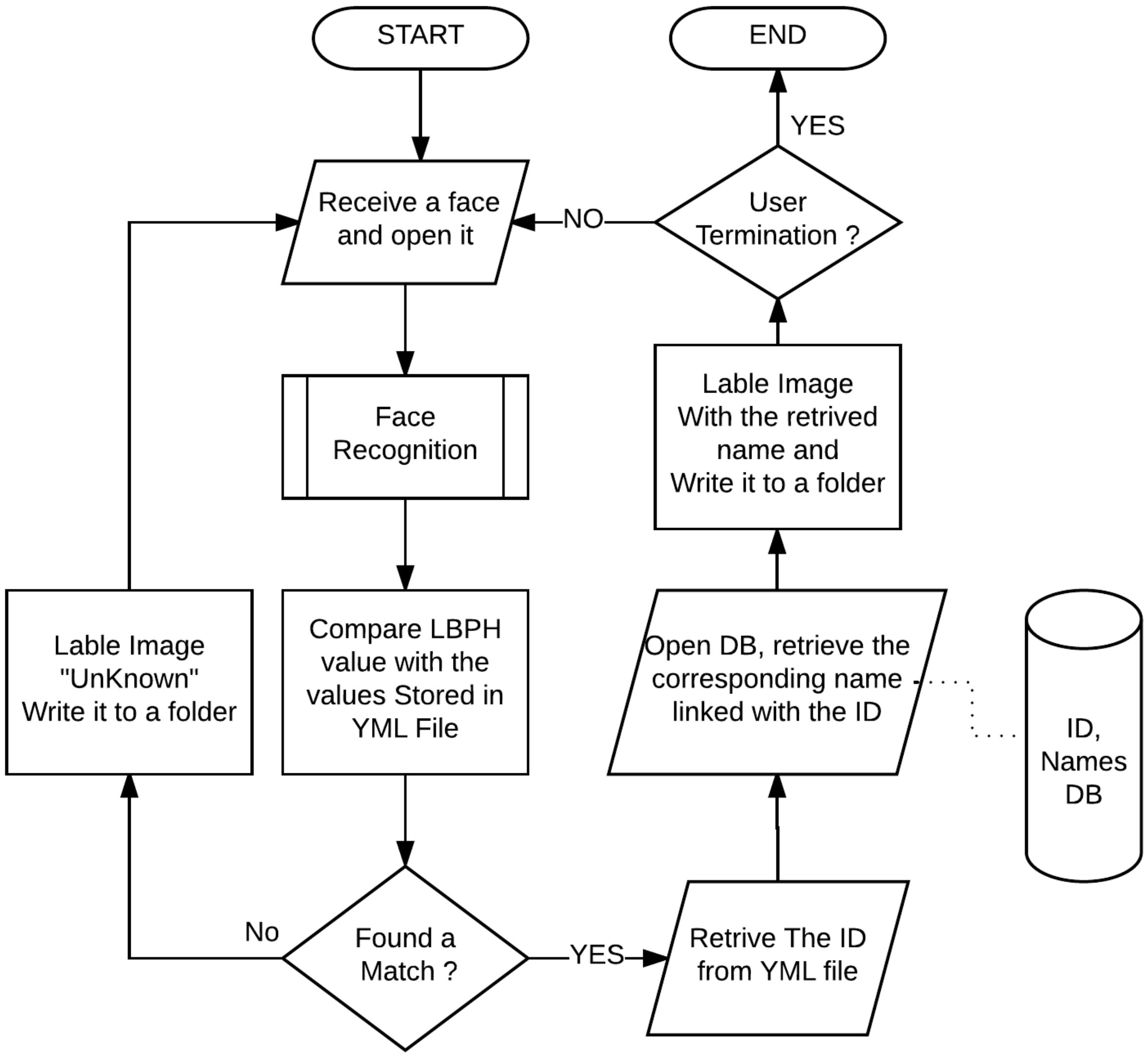

START
END
YES
Receive a face and open it
NO
User Termination ?
Face Recognition
Lable Image With the retrived name and Write it to a folder
Lable Image "UnKnown" Write it to a folder
Compare LBPH value with the values Stored in YML File
Open DB, retrieve the corresponding name linked with the ID
ID, Names DB
No
Found a Match ?
YES
Retrive The ID from YML file

Figure 3.8: Servers flow chart.

The flow chart in **figure 3.9** clarifies the local binary pattern histogram flow. Initially divide the image into blocks and apply the histogram calculation which involves selecting the central pixel in the block, and declare its intensity with a number between 0 and 254 then declare the eight surrounding pixel in the same manner, any pixel with the same or higher intensity is labeled with 1, otherwise it's a zero.

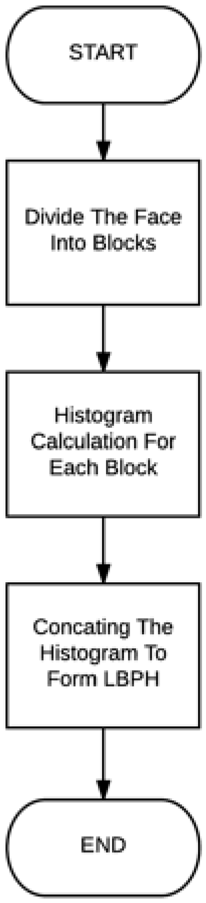

START
Divide The Face Into Blocks
Histogram Calculation For Each Block
Concating The Histogram To Form LBPH
END

Figure 3.9: Face Recognition Algorithm flow chart

As soon as deciding for the blocks intensity is over, they are grouped into one weighted value, which called local binary pattern histograms (LBPH), this block represent the unique features of each image, which will be compared with the LBPH of the database images i.e. YML file.

3.5 Servers Subsystem (Training stage)

The servers are responsible for another task, which is to create the data set images and train them to the LBP algorithm to obtain the YML file, which contains all the calculated LBPH for each image in the dataset images.

Whenever a new face comes, the recognition process mentioned in **figure 3.9** takes place, but before that stage is applicable, the database (DB) must be created, in addition to the YML file described above, therefore dataset images for each user must be collected and linked to a user ID and name.

What follows that these trained images must be processed with the LBP algorithm and create a corresponding YML file that the algorithm can read and retrieve the weighted values when needed.

3.5.1 Flow Charts

The first flow chart **figure 3.10** shows the creation of data set images, this step occurs each time the supervisor wishes to add known persons to the database.

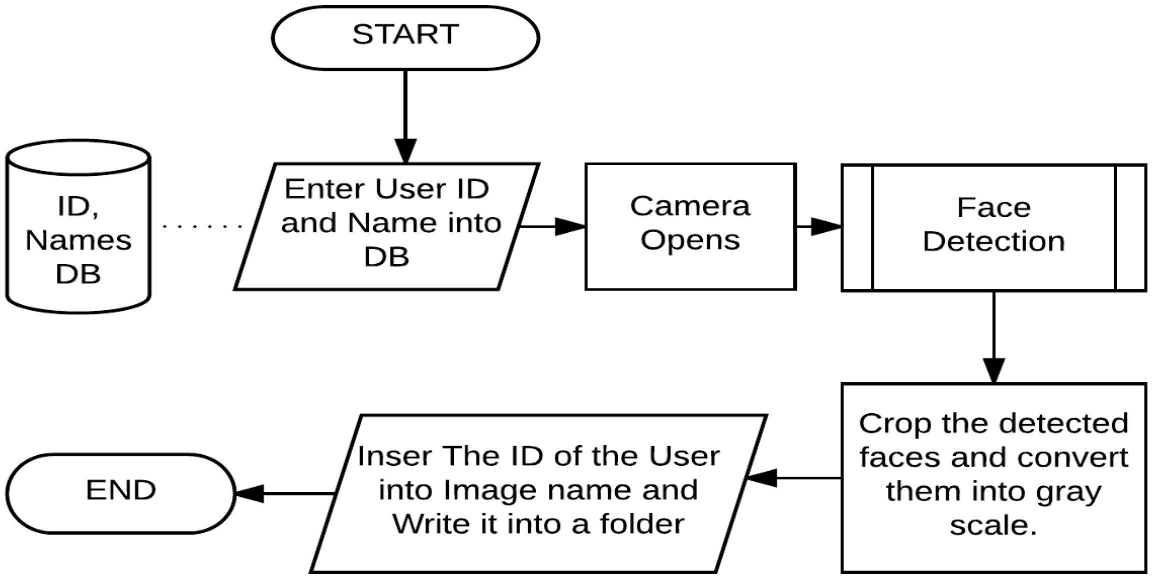

START
ID, Names DB
Enter User ID and Name into DB
Camera Opens
Face Detection
Crop the detected faces and convert them into gray scale.
Inser The ID of the User into Image name and Write it into a folder
END

Figure 3.10: Flow chart of dataset creation and inserting names and ID to DB.

The second flow chart **figure 3.11** in this section shows the training process, which takes its main part using the LBP algorithm discussed in chapter two.

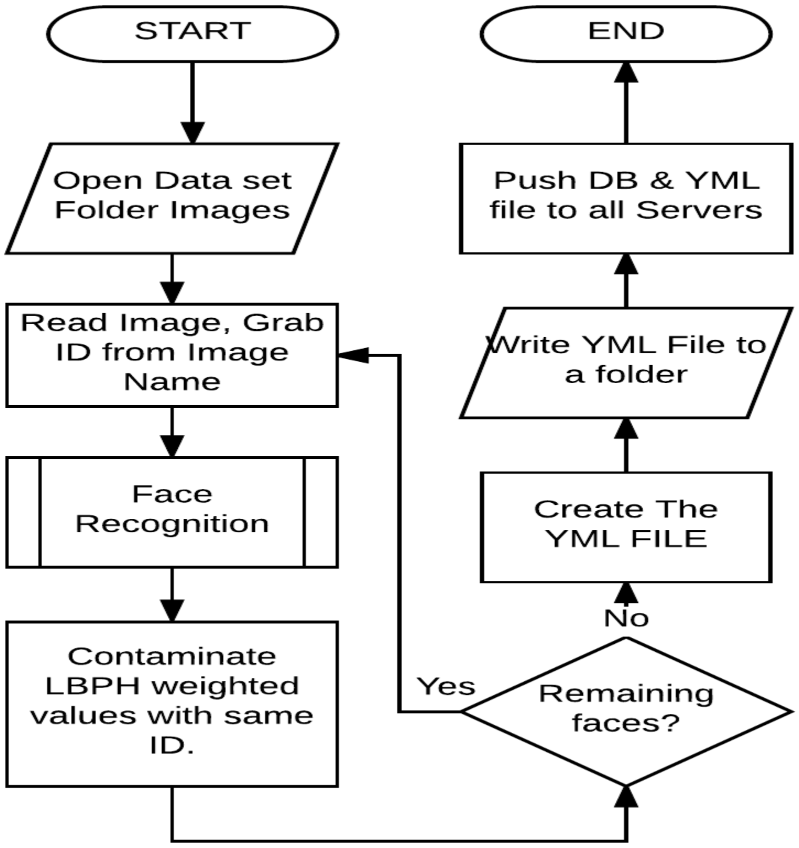

START
Open Data set Folder Images
Read Image, Grab ID from Image Name
Face Recognition
Contaminate LBPH weighted values with same ID.
Yes
Remaining faces?
No
Create The YML FILE
Write YML File to a folder
Push DB & YML file to all Servers
END

Figure 3.11: Flow chart of training procedure and pushing DB and YML to servers.

3.6 General System Illustration

In this section, theoretical and expected implementation is covered, to provide an insight about the following two chapters, where implementations and testing take place.

The general system illustration will provide a sufficient overview for the component functionality. Giving a major focusing for the server part where explanation to how the server -supervisor computer and mobile communicate and transfer data.

Beginning with the general block diagram of the system, then discussing the general flowchart, where the general system block diagram consists of the previous subsystems block diagrams, combining these subsystems into one block diagram will result in a better understanding for the functionality of this system.

3.6.1 General Block Diagram

The general system block diagram is represented in **figure 3.12**

1) Camera stream video to the raspberry pi.
2) The detection algorithm then receives this stream, loads a frame, detects the faces in it – if there is any – and returns a cropped faces ready for transmission to the servers.
3) Every face is then transmitted to the server after the raspberry pi opens a session with the server to stream the faces.
4) The transmission continues until all the faces are successfully delivered to the servers.
5) The server starts receiving the image through the socket (Session) that were opened, and starts the recognition process.
6) In the background, the recognition algorithm will run to calculate the LBPH for the faces in the database, this process happens at initial system run, and whenever a new face is added to the main database, i.e. training new persons to the system.
7) Calculation for the LBPH for each received image and compare the result with the database LBPH.
8) If a match occurs, the database is opened and the corresponding name to that matched ID is retrieved
9) The supervisor will be monitoring the main server screen where each known face will be shown with the name of the person and each unknown face is flagged with a mark.
10) In case that the supervisor is away from the system, the unknown face is sent to the supervisor smart phone.

The previous steps summarize the system response and processing to each input frame, it can be continuously repeated if a new frame contains faces.

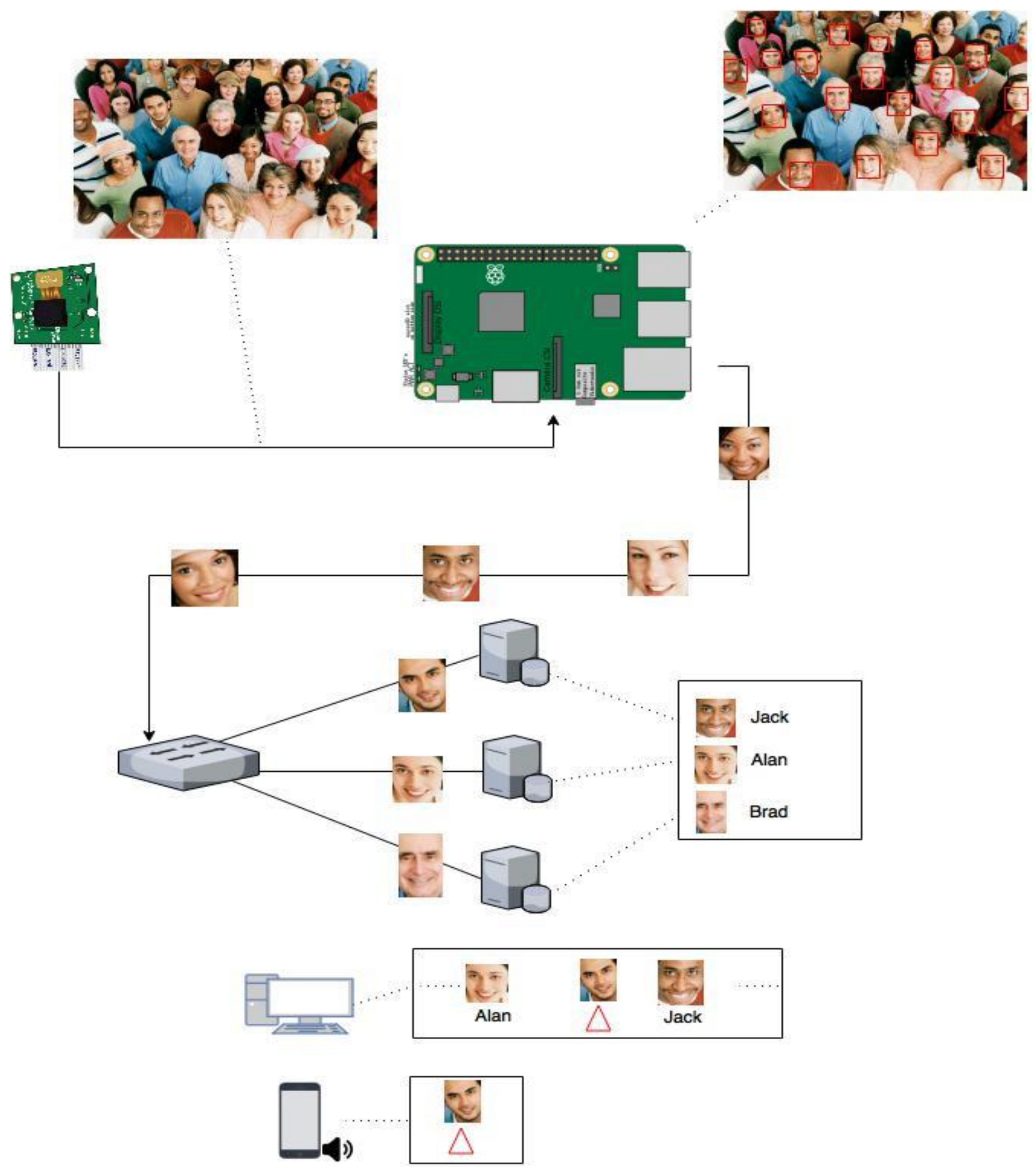


Figure 3.12: General System Block Diagram

3.5.2 General Flowchart

The general flow chart for the system **figure 3.13** describes the initializing process for the camera, and the processing regarding face detection and recognition over one server.

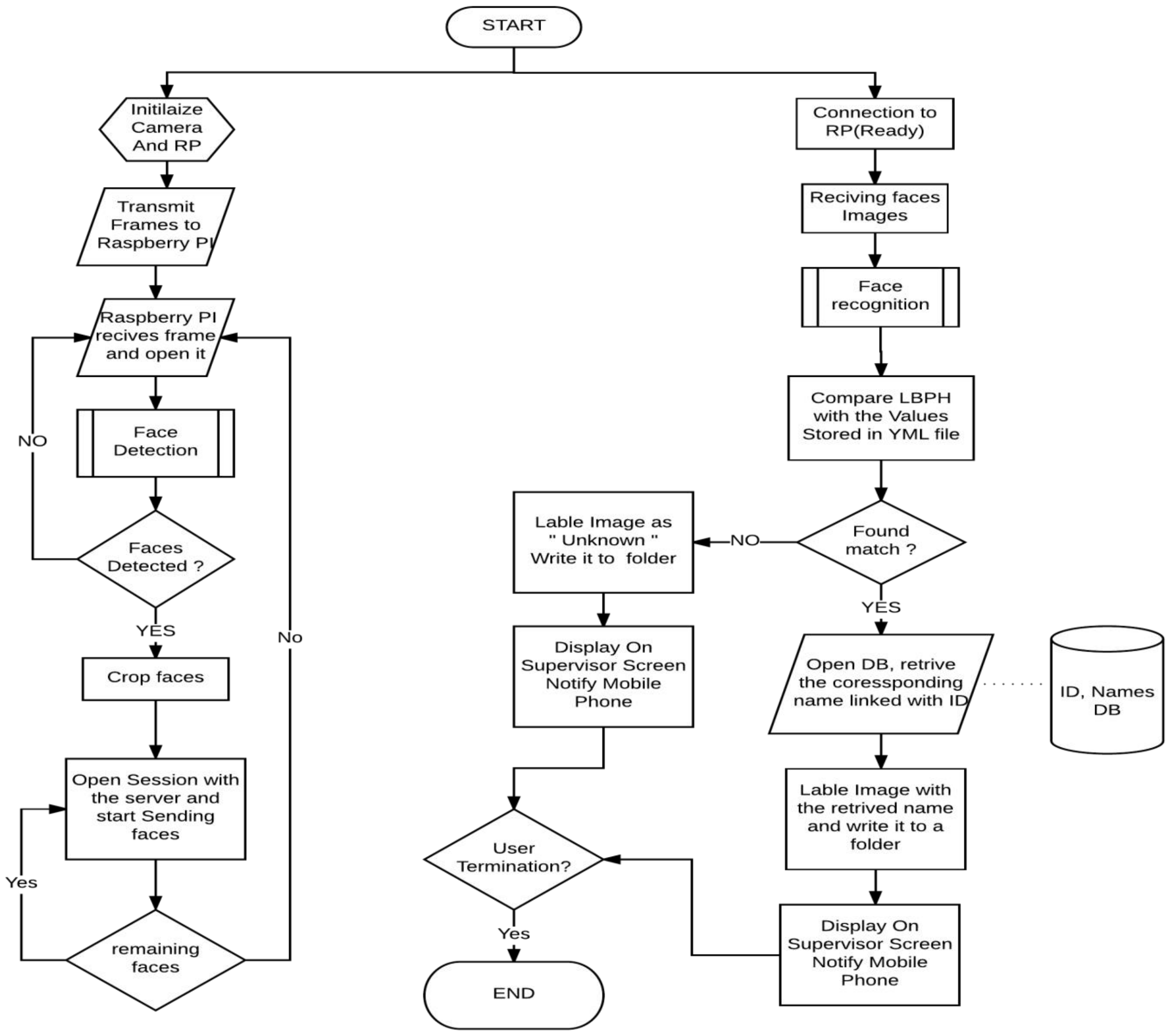

START
Initilaize Camera And RP
Transmit Frames to Raspberry PI
Raspberry PI recives frame and open it
Face Detection
Faces Detected ?
NO
YES
Crop faces
Open Session with the server and start Sending faces
remaining faces
Yes
No
Connection to RP(Ready)
Reciving faces Images
Face recognition
Compare LBPH with the Values Stored in YML file
Found match ?
NO
YES
Lable Image as " Unknown " Write it to folder
Display On Supervisor Screen Notify Mobile Phone
Open DB, retrive the coressponding name linked with ID
ID, Names DB
Lable Image with the retrived name and write it to a folder
Display On Supervisor Screen Notify Mobile Phone
User Termination?
Yes
END

Figure 3.13: General System Flow Chart

The following steps summarize the flowchart:

1) The system is turned on, at the same moment that the camera is initializing, the servers will start.
2) Camera starts streaming and sending frames to the raspberry pi.
3) The raspberry pi reads the incoming frame and detects the faces in it, if no faces appear the raspberry pi will load and process another frame.
4) If a face is detected, the raspberry pi will crop the face and prepare it for the next stage.
5) A session is opened with the servers so the server starts receiving the faces.
6) Each time a face is received; the server applies the recognition algorithm, and compare the result of LBPH with the LBPH from the YML file.
7) If match occurs, the server opens the database and retrieves the corresponding name with the ID.
8) Label the image with the retrieved name, and display it on the supervisor screen.
9) If the algorithm fails to find a match, the image is labeled as Unknown, and viewed on the supervisors screen and sent to his smart phone.

This chapter concludes the theoretical background; the hardware and software implementations will be discussed in the next chapter.

CHAPTER FOUR

System Implementation

Chapter Four

System Implementation

4.1 Introduction

In this chapter, all the real work is done; this chapter contains all the system implementation regarding the system setup whether in hardware connections or in software programming.

The following content is supposed to give the reader a clear explanation and full understanding of the steps that converted the system design (Chapter 3) to a working system.

The chapter will start with the illustration of the system components and the method of connecting and relating all parts together. Then it will investigate each part and clarify its requirements and tasks. This chapter will conclude all the theoretical aspects in chapter three and implement them in the real world.

4.2 General Flow Diagram of System Implementation

The implantation in this system took five main stages, in each stage different steps were taken, **figure 4.1** contains the main five stages, and the rest of this chapter will explain each stage separately.

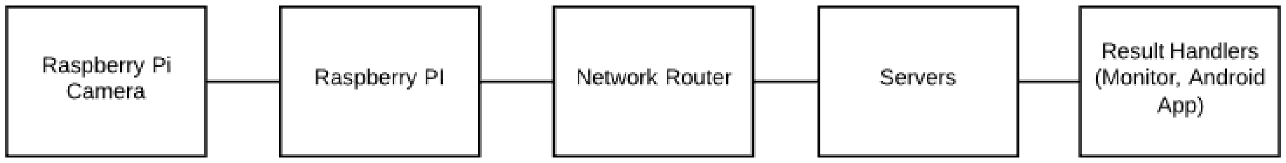


Figure 4.1: General Flow Diagram of System Implementation.

As shown in the above figure, the system implementation will start with the raspberry pi camera, the connection to the raspberry pi, and then enabling, configuring and running are covered.

Raspberry pi, the main component in this project, most of the job is done here, from installing the operating system, and connecting the main components, and programming the detecting-cropping algorithm to TCP/IP server establishment to send images.

However, the server stage has a huge role too, since the load-balancing concept would not be achieved without servers. This stage contains operating system installation, programming the TCP/IP client to receive faces, programming the datasets creator, LBP trainer, recognition, and programming the SQL database to save the known user ID’s and names, this is all part of servers duties that are covered.

The in-between stage (Network Router) was an easy, simple phase were connecting the servers physically, and programming the router to bind devices MAC to a specific IP took place.

Final stage, Result handlers, this stage where all the system results are viewed from, it contains the supervisor monitoring either by the screen or on his mobile phone.

4.3 Raspberry Pi Camera

This section covers two things, a general information about the camera module, and how it is connected, then information about the camera programming code, were it was enabled and configured.

4.3.1 Connecting the Camera

The system starts by connecting the Raspberry Pi camera module with the raspberry pi microcomputer through a 15-pins ribbon cable, the cable connects between the fast camera Serial Interface bus and the system-on-chip processor (SoC), the camera is connected to the raspberry pi in **figure 4.2**.

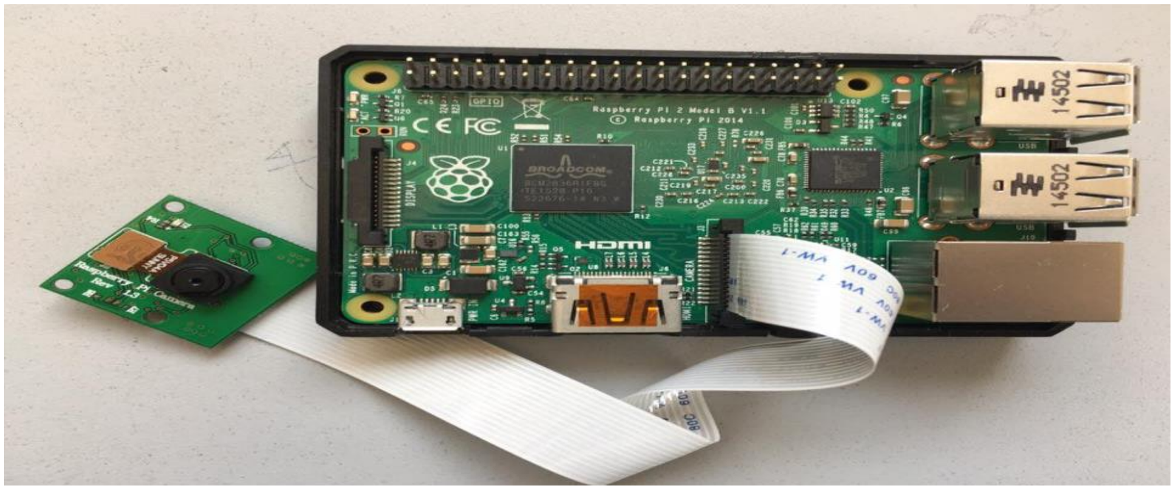


Figure 4.2: Connecting raspberry pi camera to the Raspberry pi

4.3.2 Camera Setup and Configuration

This stage assumes that Raspbian Operating System (OS) is already Installed on Raspberry pi Microcomputer, this is done through later stage (section 4.4). In this section the camera setup and configuration are done, they are shown in appendix A.

4.4 Raspberry Pi Microcomputer:

The raspberry pi -as mentioned in the previous chapters- is a major node in the system because it is the part in which streaming, face detection, face cropping and image transfer processes take place. After unpacking raspberry pi 2, it is connected to the power and to all the needed peripherals (monitor, keyboard and mouse, and Ethernet cable)

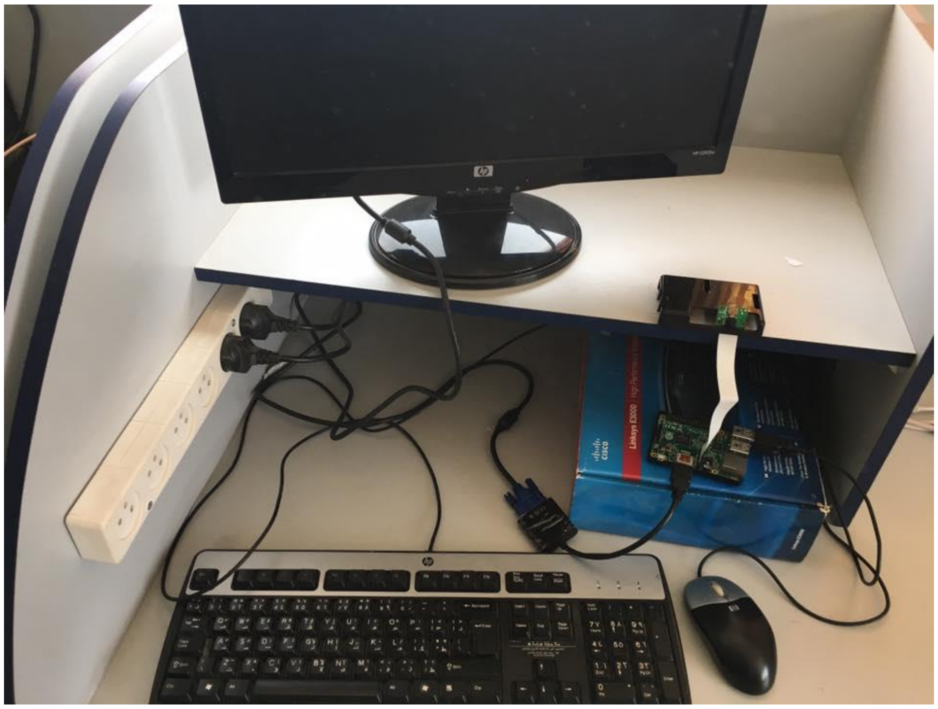


Figure 4.3: Raspberry pi with mouse, keyboard, camera, and monitor connected.

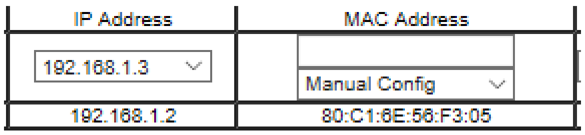
Raspberry Pi operates on Raspbian OS, it is the Linux OS distribution that was built for Raspberry Pi microcomputer, and the installation of this system is shown in appendix B.

Since the raspberry pi is now up and running, then both face detection and cropping are programmed into the raspberry pi, the necessary programing codes for face detection and cropping were written in Python. Mainly face detection is accomplished by using the function (detectMultiScale(image[, scaleFactor[, minNeighbors[, minSize[, maxSize]]]])), on each frame, this function will detect each face in the frame and pass it to the cropping stage, where the function ImWrite(imagePath,image[y:y+h,x:x+w]) will write the cropped dimensions y to y+h and x to x+w which represent the face coordination's on the RP SD.

Note: Raspberry Pi need OpenCV library in order to detect faces, the installation of this library is discussed in appendix C.

4.5 Network Setup

Connecting the raspberry pi and server to the router was an easy step, the configuration needed to be done at this point was binding each IP address to a mac address in binding table in the router **figure 4.4** shows how to bind a specific MAC to IP.



IP Address	MAC Address
192.168.1.3	Manual Config
192.168.1.2	80:C1:6E:56:F3:05

Figure 4.4: Router binding table

TCP/IP connection was programmed in python on the Raspberry pi for the transfer of detected faces; a socket is created and opened in a round-robin fashion to transmit toward the three servers. Another socket was created on the server side to transfer unrecognized faces wirelessly to the Android application on the supervisor phone .the IP address of the phone is also static.

As far for the raspberry pi, the listening state was written in python to initiate the server and waits for the server (clients) to connect, as long as the raspberry pi is up, it will keep listening for new incoming clients and bind with them to transmit a new face from the camera.

4.6 Servers:

The Servers side mainly consists of three laptops as the recognition servers, which will handle the face recognition and training process as well as managing system outcomes and results. The following sub sections will show OS installing, preparing the servers, implementations of python codes that facilitate TCP/IP connection, creation of data sets, training, and recognition.

4.6.1 Installing Operating System

At the time of project implementation started, Ubuntu latest version was 16.04, this is the version that has been installed on the three servers, appendix D contains all the needed information to install the operating system properly.

4.6.2 Preparing the Servers with OpenCV

After Installing Ubuntu and login into Desktop, the system is now ready to install all updates, upgrades, and libraries that supports OpenCV, Appendix C shows the servers preparations and OpenCV installations.

4.6.3 Programming codes

After the system is ready on the server side with the correct programs and environment, all required programs that will do the job was written in python.

A. The dataset creator program was written in order for the supervisor to be able to add new people to the database including their names and IDs. Each user ID is attached to his image name for the next process where the trainer will read this ID and assign it with the calculated weight of LBPH to that specific user (full code provided in appendix E)

B. The Trainer program was written, this program calculates the LBPH for each known person that was added in the previous step, and stores the result in an YML file in order to be used by the recognition program. The OpenCV function that facilitate the training process is recognizer.train(FaceImage,ImageID).

Where this function reads each image and links it to the user ID and then runs LBPH on it and stores the resulted weight, the complete code is on appendix F.

C. Then the Recognition Program was written, this program calculates the Local Binary Pattern Histogram for all the received images and compare each one with the stored patterns from the YML file, detects the unknown person and sends it to the Result Handlers Stage. The main function that is used in recognition is id, conf = recognizer.predict(Image) which returns a weight for the incoming images and compare it with the YML file see (appendix G) for complete code.

D. Client Program was built; the client receives the image from the Raspberry pi and store it on the servers. The socket is created with the server IP address using socket.connect (host,port) and a request to connect and retrieve images is done using socket.recv(ByteNumber), see (appendix H) for the complete code.

E. Server Program was built: the server program will send images from RP to the connected servers. (see appendix I)

F. Detection program: this program will detect the faces from the camera frames and save them on SD card. (see appendix J)

Now, servers are connected together, the servers specifications are (two devices of octa core processors, 8GB RAM, and the third one is quad-core processors, 4 GB RAM), servers are connected using local area network and configured as section 4.5 pointed, then, system is up, and testing this system is the next chapter topic.

4.7 Result Handlers

The results can be observed in the system using either, supervisor main server screen or the android application.

4.7.1 Monitoring using supervisor Screen

In this approach, there will be a main server from which the unknown images are going to be displayed on its screen. The images are sent from the two servers unknown folders –where the unknown faces are - to the main server that shows all unknown faces on the screen including images from its unknown folder.

4.7.2 Monitoring using Android Application

One of the important features of this system was adding mobility allowing the supervisor to move, while still having the target area under supervision. That feature was achieved by building an android application that will be monitoring the system and launch a notification each time unknown is detected. Android studio was used to build the application.

The application has three activities, the first is the main activity where all stored images are viewed in a list and the first connection to the system is established. The second activity is the activity where new received images are viewed after clicking the notification bar and the third activity is where the clicked photo is viewed with its information, the following figures shows the system icon with the three different windows within this application.

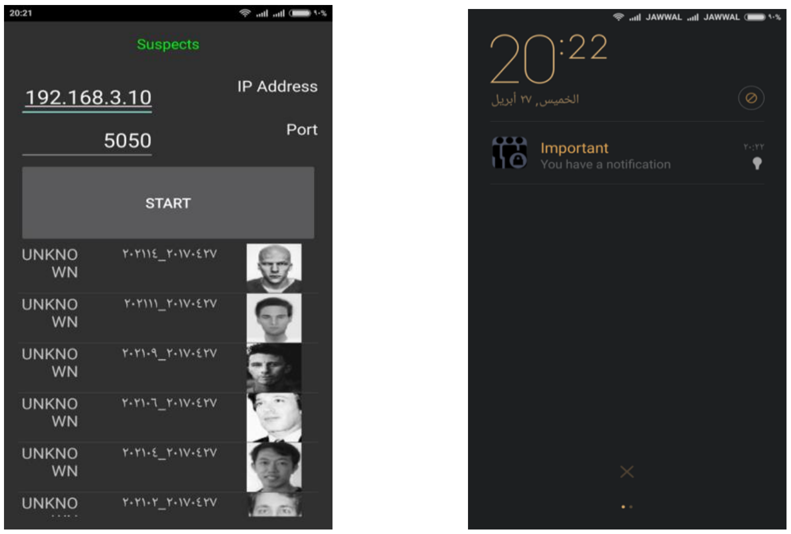


Figure 4.5: Android application.

The application will start a main thread that constantly listens to the channel defined for the system. Once any server detects a new unknown face, the server will have the way open to send the image to the phone.

To do so the server will launch a program build in java that will open a socket with the phone IP address for the communication, the image is then received and stored in a database on the mobile storage and a new notification is displayed for the user.

In next chapter testing phase for the system to obtain an optimum values for a better detection and recognition parameters is done.

CHAPTER FIVE

Testing and Analyzing Data

Chapter Five

Testing and Analyzing Data

5.1 Introduction

After system is properly configured and all the needed programs are written, this chapter will go through many tests that were accomplished over the system; these tests consist of four main parts, which are:

1. Testing detection parameters.
2. Testing faces transmission time from raspberry pi to servers.
3. Testing recognition-processing time.
4. Testing recognition parameters.

In detection parameters test, the main function that was mentioned earlier in chapter four regarding the detection which is (detectMultiScale(image[, scaleFactor[, minNeighbors[, minSize[, maxSize]]]])) will be tested in terms of Scale Factor (SF), Minimum neighbors (MN) and window size.

In testing the faces transmission time from raspberry pi to servers, the main aim is to notice how the servers can handle the incoming images rates as if the recognition side has one server or three servers in addition to a comparison between the ideal delay and actual run time delay.

Finishing this chapter with recognition parameters tests. The first test here is dedicated to find the recognition-processing time with one server and three servers working to recognize the faces, followed with a recognition accuracy test in terms of correctly recognized persons.

5.2 Detection Parameters Tests

In this section, the first test was about the first component of the system i.e. the Camera. The test will determine several parameters related to the camera, these parameters were used in the face detection algorithm to help detect people faces with the least percentage of error.

The test investigates three main parameters of the captured frame, these parameters are:

1. **Windows Size** :the size of the rectangle drawn around the detected area and its divided into :
    - Minimum Size
    - Maximum Size

Window size will decide for the minimum detection distance for after which, no detection is valid.

2. **Scale Factor (SF)**: Parameter specifies how much the image size is reduced at each image scale.

3. **Minimum Neighbors (MN)**: The minimum number of applied windows that each candidate rectangle should contain.

The test will run with the following conditions:

1) The camera will take a 100 image (faces and wrong faces are counted equally) on each run.
2) Random movement will be in front of the camera, some people are closer and some are further away.
3) On each camera run, one of the previous parameters (Window size, SF, MN) was under test to observe the number of wrongly detected faces.
4) Window size will decide for the distance, the first window size (20*20 | 300*300) is capable of 4 meters, second window (50*50 | 250*250) can detect 3.7 meters while the last window (80*80 | 200*200) detects 3 meters away.

The test result are in the tables from **5.1 – 5.9**

Table 5.1: Detection parameters test (SF 1.3, MN 6)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	21	36	27	28%
50*50	**250*250**	24	40	35	33%
80*80	**200*200**	52	38	43	44.33%

Table 5.2: Detection parameters test (SF 1.2, MN 6)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	13	20	23	18.66%
50*50	**250*250**	26	19	19	21.33%
80*80	**200*200**	16	14	23	17.66%

Table 5.3: Detection parameters test (SF 1.1, MN 6)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	31	33	20	28%
50*50	**250*250**	12	15	12	13%
80*80	**200*200**	31	39	39	36.33%

The three tables (**5.1**, **5.2**, and **5.3**) shows the percentage of error in detection when the MN is fixed and the SF is varying respectively, each test is repeated three times, with three window size ranges in each.

After observing these tables it can be said that when SF is 1.3 the percentage of error is relatively high, when the SF is reduced to be 1.2 the percentage of error decreases. However ,when SF is reduced to 1.1 there was an increment in the error at (20×20-300×300) window size as well as an increment of error at (80×80-200×200) ,but it's clear that the error decreased when the window size (50×50-250×250) is used .

This implies that the window of size range ((50×50-250×250) gives a small error percentage when SF is 1.1, yet those results are obtained when the MN is fixed.

In the following tables (**5.4-5.6**) the same conditions applies, but a different MN was chosen.

Table 5.4: Detection parameters test (SF 1.3, MN 5)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	29	22	17	22.67%
50*50	**250*250**	24	27	22	24.33%
80*80	**200*200**	67	58	70	65%

Table 5.5: Detection parameters test (SF 1.2, MN 5)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	63	19	25	35.67%
50*50	**250*250**	24	17	21	20.67%
80*80	**200*200**	36	40	46	40.67%

Table 5.6 Detection parameters test (SF 1.1, MN 5)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	15	17	14	15.33%
50*50	**250*250**	21	8	17	15.33%
80*80	**200*200**	49	31	40	40%

After decreasing MN to five, previous tables shows the results for each SF value .The error is fluctuating between 20% to 65% in the two tables (**5.4, 5.5**) which is considered a relatively high error percentage.

When the SF decreased to 1.1 ,the error dramatically decreases to about 15% for window size ranges (20×20-300×300) and (50×50-250×250) .at the same time ,high percentage (about 40%) still exists for window size range (80×80-200×200).

It is worthy to mention that the above results were taken in the daylight condition in a normal sunny day, different light conditions will cause different results especially for the far faces.

The third set of tables was obtained after decreasing MN to four:

Table 5.7: Detection parameters test (SF 1.3, MN 4)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	37	30	20	29%
50*50	**250*250**	20	16	35	23.67%
80*80	**200*200**	52	54	61	55.67%

Table 5.8: Detection parameters test (SF 1.2, MN 4)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	16	12	4	10.66%
50*50	**250*250**	16	18	23	19%
80*80	**200*200**	54	46	43	47%

Table 5.9: Detection parameters test (SF 1.1, MN 4)

Minimum Size	**Maximum Size**	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**False percentage %**
20*20	**300*300**	29	29	28	28.66%
50*50	**250*250**	29	27	36	30.66%
80*80	**200*200**	57	51	65	57%

The last set of table’s shows an improvement in detection when SF 1.2 and MN is four; a 10 % error was obtained here.

Figure 5.1 represents the best window size in terms of the minimum detection error, the (20*20 | 300*300) for different values of SF and MN.

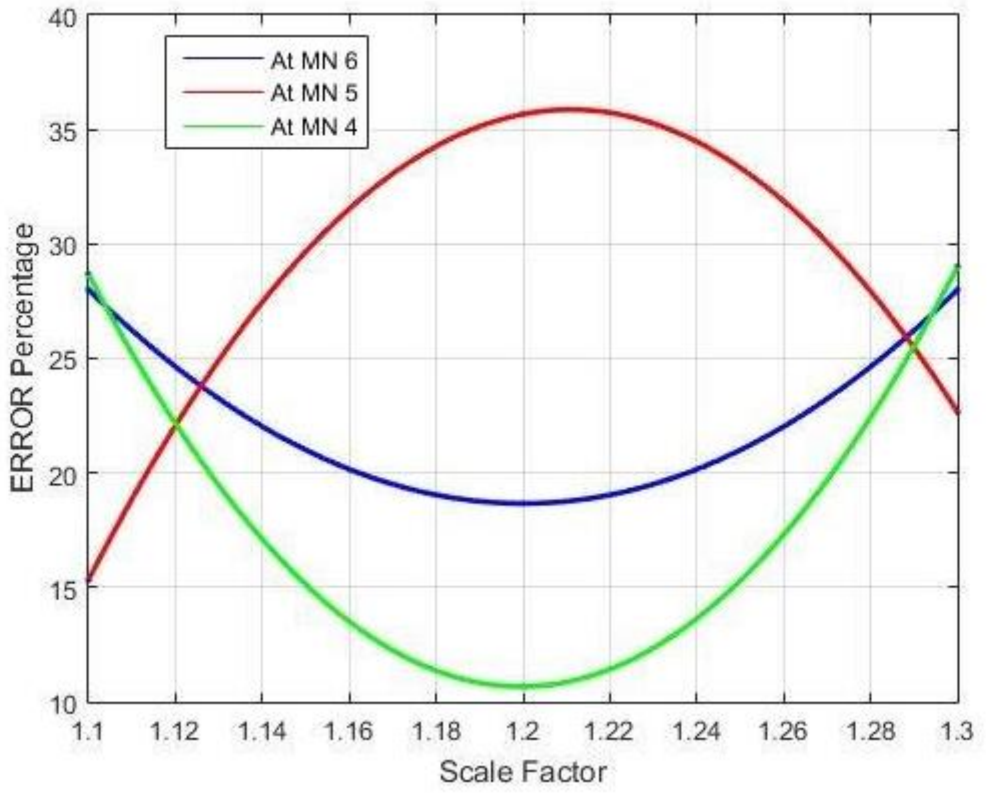


Figure 5.1: Error percentage versus scale factor for different minimum neighbors' values.

Using Matlab fitting tool for the window size (20*20 | 300*300) to calculate the number of false detections per 100 sample, the following equations were derived

Number of false detections $|_{\text{MN=4}} = (\mathbf{1817}) \times \boldsymbol{SF^2} - (\mathbf{4358}) \times \boldsymbol{SF} + \mathbf{2625}$ Equation (5.1)

Number of false detections $|_{\text{MN=5}} = (\mathbf{-1667}) \times \boldsymbol{SF^2} + (\mathbf{4037}) \times \boldsymbol{SF} - \mathbf{2408}$ Equation (5.2)

Number of false detections $|_{\text{MN=6}} = (\mathbf{933.3}) \times \boldsymbol{SF^2} - (\mathbf{2240}) \times \boldsymbol{SF} + \mathbf{1363}$ Equation (5.3)

To test the equations as an example the number of false detections at MN = 4 and SF = 1.2 using equation 5.2 the theoretical value was calculated to be 11.88, while the actual value was found to be 10.66.

5.2.1 Detection Parameters Tests

Previous results were conducted with three reading each time, the following tests works with window size of (20*20 | 300*300) but this time the SF were varied from 1.1 - 1.3 with a 0.02 step each time.

Table 5.10 provide the SF readings versus number of false detections for MN 4.

Table 5.10: Detection parameters test for varying SF for MN of 4.

Scale Factor	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**Average**	**Error Percentage %**
1.1	13	10	24	15.66	15.67
1.12	11	7	14	10.66	10.67
1.14	10	8	7	8.33	8.33
1.16	7	10	8	8.33	8.33
1.18	9	6	10	8.33	8.33
1.2	8	12	8	9.33	9.33
1.23	10	11	10	10.33	10.33
1.25	10	12	12	11.33	11.33
1.27	11	13	16	13.33	13.33
1.3	13	16	16	15	15

The following figure (figure 5.1.2) represent the previous values, the values were plotted and then fitted to a different polynomial order (four, five and six).

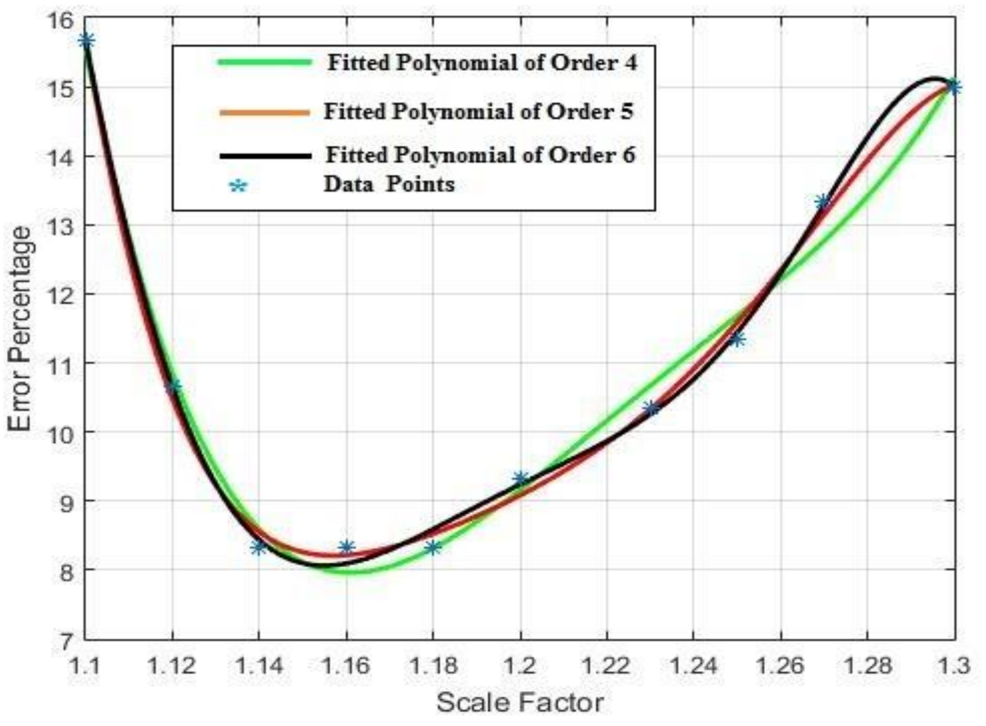


Figure 5.1.1: Error percentage versus scale factor for MN 4 and different polynomial fit.

Table 5.11 provide the SF readings versus number of false detections for MN 5.

Table 5.11: Detection parameters test for varying SF for MN of 5.

Scale Factor	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**Average**	**Error Percentage %**
1.1	13	9	11	11	11
1.12	10	4	6	10	10
1.14	11	12	6	9.66667	9.67
1.16	10	8	12	10	10
1.18	11	10	13	11.3333	11.3
1.2	11	14	12	12.333	12.3
1.23	12	15	13	13.33	13.3
1.25	16	16	14	15.33	15.3
1.27	18	18	16	17.33	17.3
1.3	20	20	19	19.66	19.6

The following figure (figure 5.1.3) represent the previous values, the values were plotted and then fitted to a different polynomial order (four, five and six).

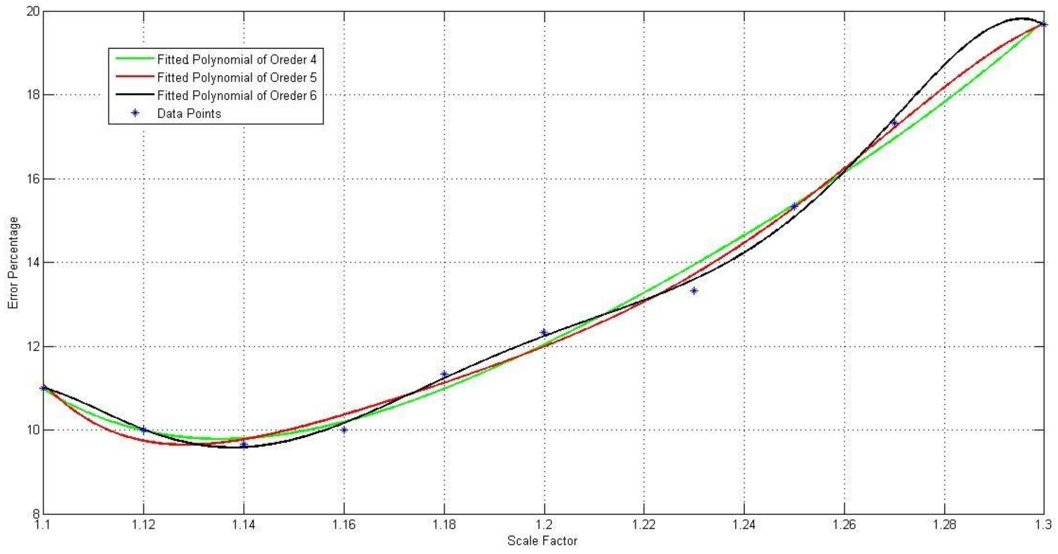


Figure 5.1.2: Error percentage versus scale factor for MN 5 and different polynomial fit.

Table 5.12 provide the SF readings versus number of false detections for MN 6.

Table 5.12: Detection parameters test for varying SF for MN of 6.

Scale Factor	**Number of false (1)**	**Number of false (2)**	**Number of false (3)**	**Average**	**Error Percentage %**
1.1	16	15	2	11	11
1.12	17	14	15	15.33	15.3
1.14	18	15	16	16.33	16.33
1.16	19	17	20	18.6	18.6
1.18	20	20	18	19.33	19.3
1.2	20	21	21	20.66	20.6
1.23	22	20	21	21	21
1.25	19	22	18	19.66	19.6
1.27	18	20	18	18.66	18.6
1.3	16	20	17	17.66	17.6

The following figure (figure 5.1.4) represent the previous values, the values were plotted and then fitted to a different polynomial order (four, five and six).

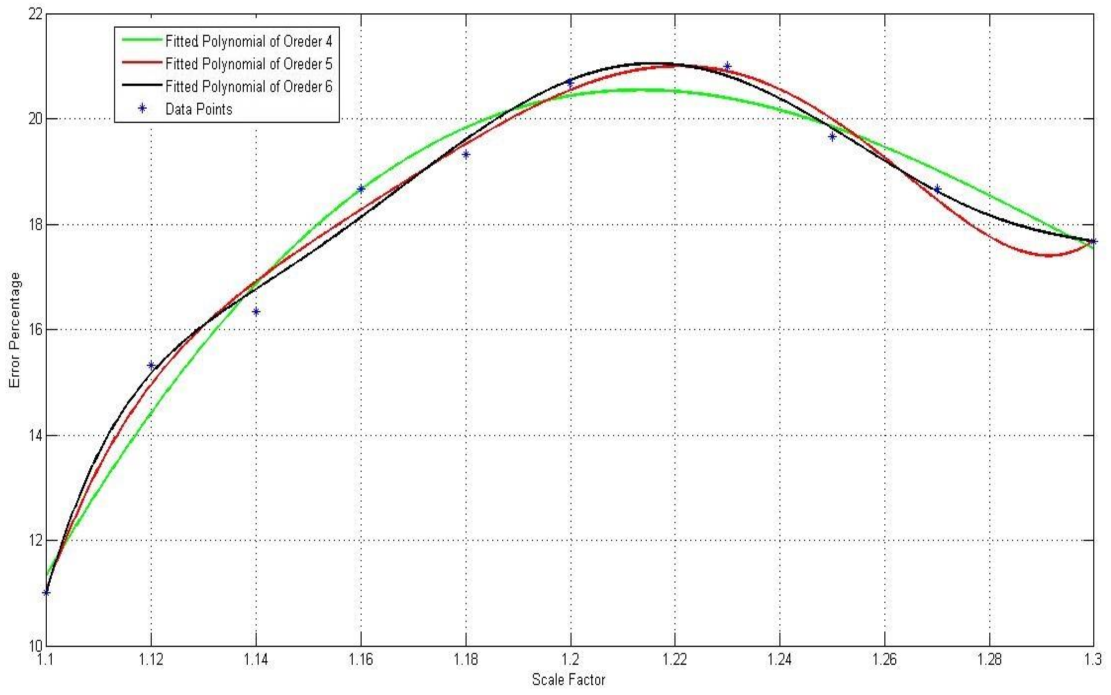


Figure 5.1.3: Error percentage versus scale factor for MN 6 and different polynomial fit.

5.3 Testing Faces Transmission Time from Raspberry Pi to Servers.

In this section, a test was applied to determine the speed of the file transfer between raspberry pi and the servers using socket programming, the number of transferred files is determined and then a relationship is obtained between number of receiving servers and number of files transferred.

The following tables clarify the results obtained, note that the time is a conventional term that corresponds to speed, in the tables below there are two main readings of the transmission time, which are:

1) With delay.
2) Without delay.

Table 5.13 contains data when one server is connected and receiving.

Table 5.13: Transmission Time towards one server

Number of images sent	Transmission time without delay (seconds)	Transmission time with delay (seconds)
10	0.042	0.84
50	0.225	5.06
100	0.4	14.24
200	0.868	26.51
500	2.4	65.16
750	3.43	80.93
1000	4.56	107.285

Table 5.14 contains data when three servers are connected and receiving.

Table 5.14: Transmission Time towards three servers.

Number of images sent	Transmission time without delay (seconds)	Transmission time with delay (seconds)
10	0.0723	0.84
50	0.2029	4.96
100	0.3266	10.26
200	0.8699	21.56
500	1.434	54.04
750	1.94	80
1000	2.21	105

As shown above the transmission time increases slightly with images below 200, and then it starts to increase notably after 500 images.

The tables also show the difference in results between one and three servers, in the three servers setup all the transmission time readings were clearly less than the one server setup, it can be said that load balancing is achieved when the number of servers increased.

However ,the time gets large in the two cases when the files number exceeds 750 , (96.94) on one server and (93.68) on three servers , which implies that the number of servers limits the speed of file transfer .New servers should be add to reduce the transmission time after 750 files and so re-achieve the load balancing again .

The results with delay are the practical results, which were taken for analysis, because they represent the actual processing time .They were fitted with Matlab and curves were drawn.

After analyzing the results obtained above using Matlab, **figure 5.2** represents the images transmission time towards one server and three servers.

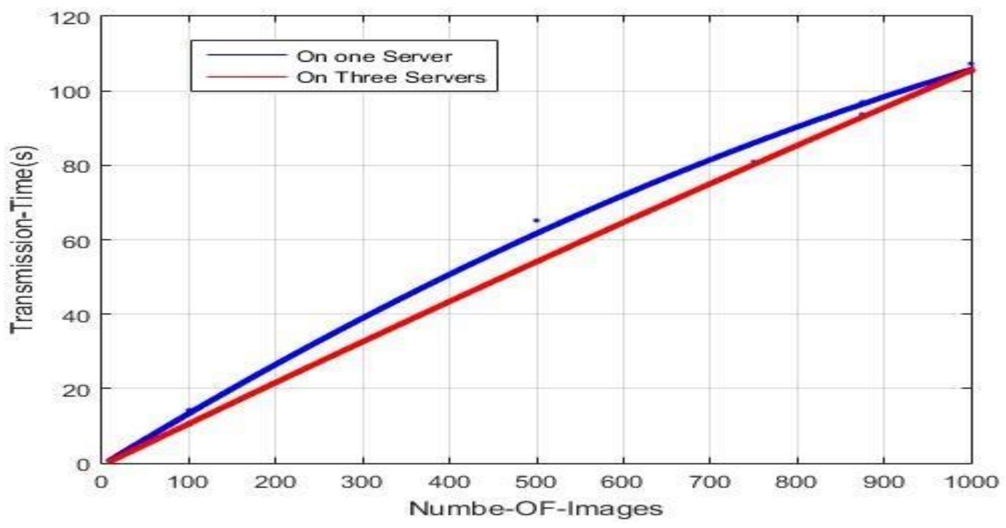


Figure 5.2: Transmission time towards servers.

After observing figure 5.2, it is clear that when the image number exceeds 750 images, the required transmission time increases. The curve represents polynomial equations of second order and the equations were approximated using Matlab fitting tool to the following formulas:

<u>For one server</u>

$$\boldsymbol{T = \left(-3.633 \times 10^{-5}\right) \times N^2 + (0.1426) \times N - 0.5623} \qquad \text{Equation (5.1)}$$

<u>For three server</u>

$$\boldsymbol{T = \left(-6.643 \times 10^{-5}\right) \times N^2 + (0.1127) \times N - 0.6345} \qquad \text{Equation (5.2)}$$

Where T represents the required time in seconds and N represents the number of images to be sent, applying equation 5.1 on 875 images as an example gives $T|_{875(\mathrm{ONE})}$ = 96.39 seconds, and when applying equation 5.5 on 875 images gives $T|_{875(\mathrm{Three})}$ = 92.89 seconds.

The test was then repeated to check the error percentage, the actual readings for sending 875 image for one server is 96.94 and for three servers 93.68 the error is then calculated using:

$$\text{Error}_{(\text{ONE})} = \left|\frac{Experimental\ Value - Theoretical\ Value}{Theoretical\ Value}\right| * 100\% = \frac{96.94-96.39}{96.39} * 100\% = 0.57\%$$

$$\text{Error}_{(\text{THREE})} = \left|\frac{Experimental\ Value - Theoretical\ Value}{Theoretical\ Value}\right| * 100\% = \frac{93.68-92.89}{92.89} * 100\% = 0.85\%$$

5.4 Testing recognition-processing time.

This section contains an essential test; the system efficiency comes when the number of servers' increases, this section investigates that by testing the required time to recognize varying number of images for one server, comparing these results with three servers to notice the effects of multi-servers approach.

Processing Time is the time required to process all images as a function of images number and number of servers.

The following points were taken into consideration while conducting the test:

1) The recognition database contained one hundred names.
2) Each person has varying number of faces, some users have three images, and others have twenty-five images.
3) The test images group consist of 1000 images, varying in size and persons.

The tables 5.15 – 5.16 show the results:

Table 5.15: Processing time on one server

Images number	Processing time (seconds)
10	3.73
100	42.29
500	202
1000	413.11

Table 5.16: Processing time on three servers

Images number	Processing time (seconds)
10	0.81
100	14.41
500	96.97
1000	126.6

Figure 5.3 was generated by Matlab to visualize the effect of adding two extra servers.

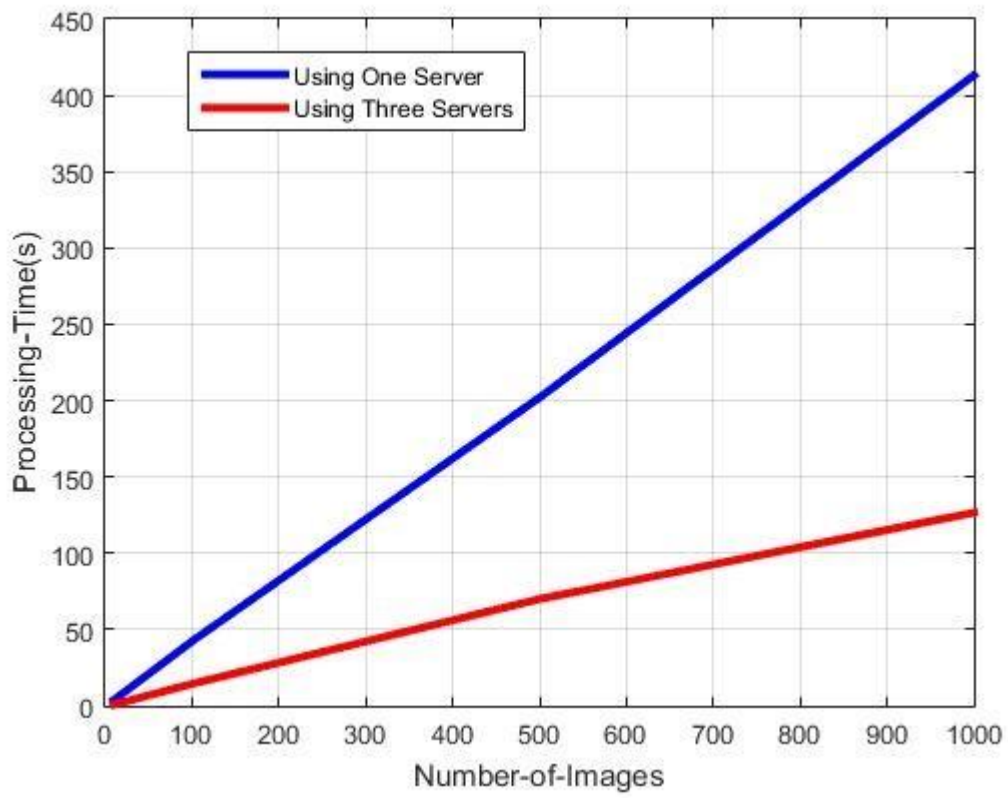


Figure 5.3: Recognition processing time (one server vs. three servers)

The curves were fitted with Matlab fitting tool; the following equations were derived for the required time to process the faces.

For one server

$$\boldsymbol{Tp = 0.4122 \times N - 0.611} \qquad \text{Equation (5.3)}$$

For three servers

$$\boldsymbol{Tp = 0.1271 \times N + 1.794} \qquad \text{Equation (5.4)}$$

Where Tp represents the required time in seconds to process N images, applying equation 5.3 on 250 faces as an example gives $Tp|_{250(ONE)}$ = 102.43 seconds, and when applying equation 5.4 on 250 faces gives $Tp|_{250(Three)}$ = 33.56 seconds.

The test then was repeated to check the error percentage, the actual readings for processing 250 faces for one server is 100.34 and for three servers 34.71 the error is then calculated using:

Error $_{\text{(ONE)}} = \left| \frac{Experimental\,Value - Theoretical\,Value}{Theoretical\,Value} \right| * 100\% = \frac{100.34-102.43}{102.43} * 100\% = 2.04\%$

Error $_{\text{(THREE)}} = \left| \frac{Experimental\,Value - Theoretical\,Value}{Theoretical\,Value} \right| * 100\% = \frac{34.71-33.56}{33.56} * 100\% = 3.39\%$

By inspecting the above tables and figure, we can say that increasing number of servers has a huge effect on processing time, to be precis when 500 images are processed on one then three servers the time reduces by the following factor$\frac{Time\ of\ three\ servers}{Time\ of\ \ one\ server}$, which yields 48% improvement when to servers were added.

5.5 Testing recognition parameters.

In this section a different tests were taken to improve the recognition accuracy which is the hardest task to do, recognition is a very complex issue, there are many parameters that could decide for the recognition accuracy. The number of images in the training set, the quality of these images, the specifications for the training set camera (laptop webcam), the detection camera (raspberry pi camera), and lighting, these among others play an essential rule in recognition stage.

The following conditions were taken into consideration while conducting the test:

1) The person under test must be trained to the recognizer prior to the recognition if we wish him to be a known person in the system.
2) While the person is being trained, various number of images for him was taken, these numbers are the number of one known person faces in the database, 10, 25, 50, 75 and 150.
3) The person under the test (whether add to the database or not) must walk towards the detection camera not side to side when test starts.
4) Since the camera specifications play an essential rule, tests with the raspberry pi camera as the trainer and the detection camera were done, another test was accomplished when the raspberry pi camera was the detection camera, and the laptop webcam was the training camera.
5) The number of images under test is 40 images, 10 images for each known person and 10 images for a stranger.

Note: Positive: Correctly recognized.

False Positive: Wrongly recognized.

Negative: Unknown person (could not be recognized)

5.5.1 Detection using raspberry pi camera - training using laptop webcam

The following table (5.17) contains the results obtained when different cameras were used for detection and training.

Table 5.17: Recognition accuracy for different cameras.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	18	4	18	45%
25	17	2	21	42.5%
50	17	6	17	42.5%
75	21	2	17	52.5%
150	23	1	16	57.5%

The chart in **figure 5.4** visualizes the results in table 5.17.

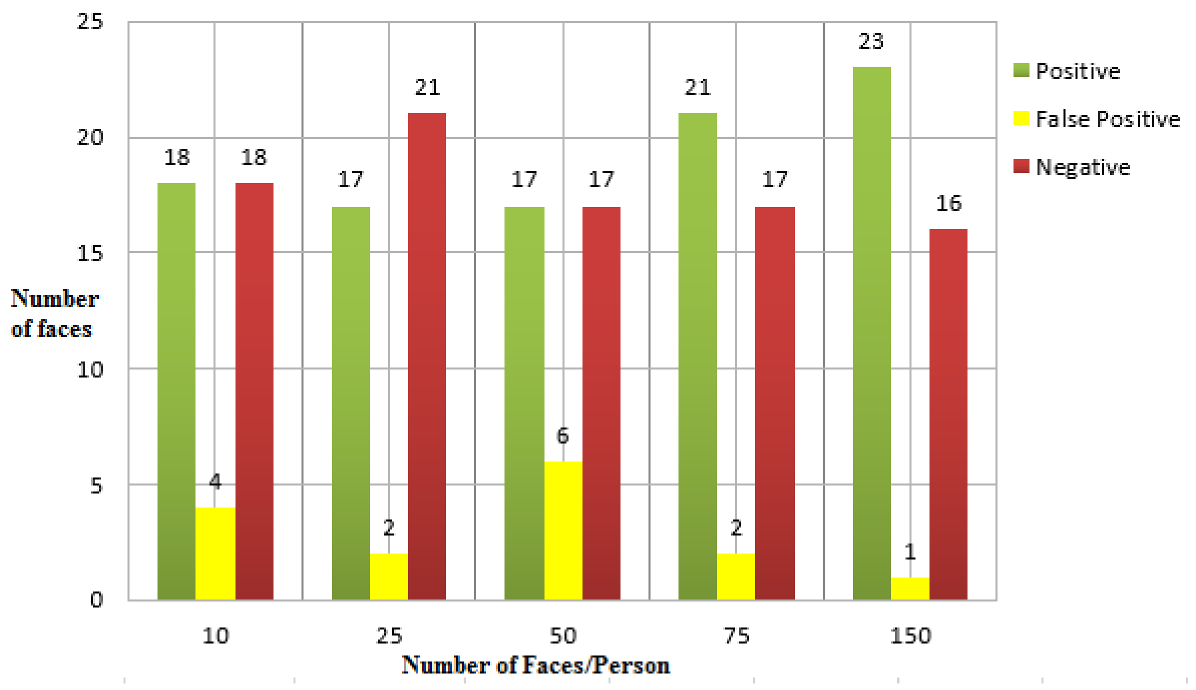


Figure 5.4: Chart representing the number of faces vs the number of faces per person in the dataset when two different cameras were used for training and detection.

These success rates are extremely low here; mainly due to two main reasons:

1) Lighting condition.
2) Two different cameras specifications.

The following table (Table 5.18) contains a test result for the different cameras, but this time an equalizer was programmed to balance the lighting of the images, this equalizer was implemented into detection and training.

Table 5.18: Recognition accuracy for different cameras with equalization implemented.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	28	0	12	70%
25	32	0	9	80%
50	33	0	7	82.5%
75	32	0	8	80%
150	33	0	7	82.5%

The chart in **figure 5.5** visualizes the results in table 5.18.

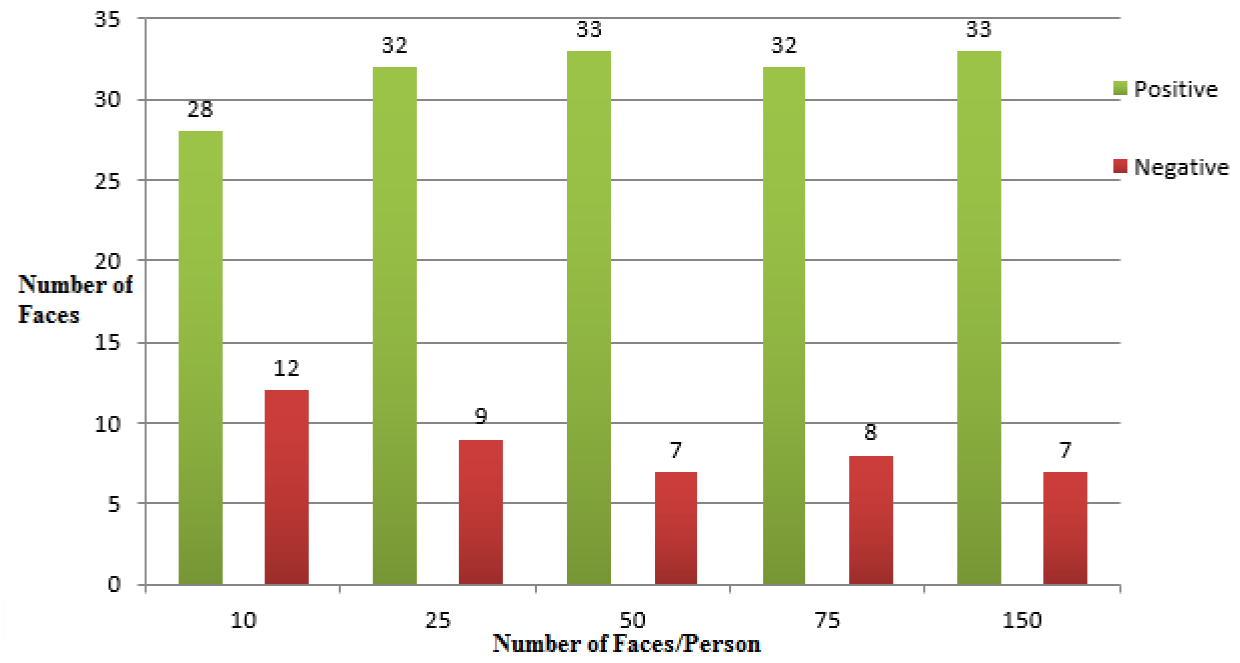


Figure 5.5: Chart representing number of faces vs number of faces per person in the dataset when two different cameras were used for training and detection with equalization implemented.

After implementing the equalizer to mitigate the lighting problem, it can be said that the results have improved, precisely for 50 faces per person the success rate went from 42.5 % to 82.5% i.e. an improvement of 51.51 %.

5.5.2 Detection and training using laptop webcam

Obviously, the next step is to try the same camera for detection and training while equalizing was implemented, this test yields the results shown in table 5.19

Table 5.19: Recognition accuracy for the same camera with equalization implemented.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	33	0	7	82.5%
25	34	1	5	85%
50	34	0	6	85%
75	35	0	5	87.5%
150	35	0	5	87.5%

The chart in **figure 5.6** visualizes the results in table 5.19.

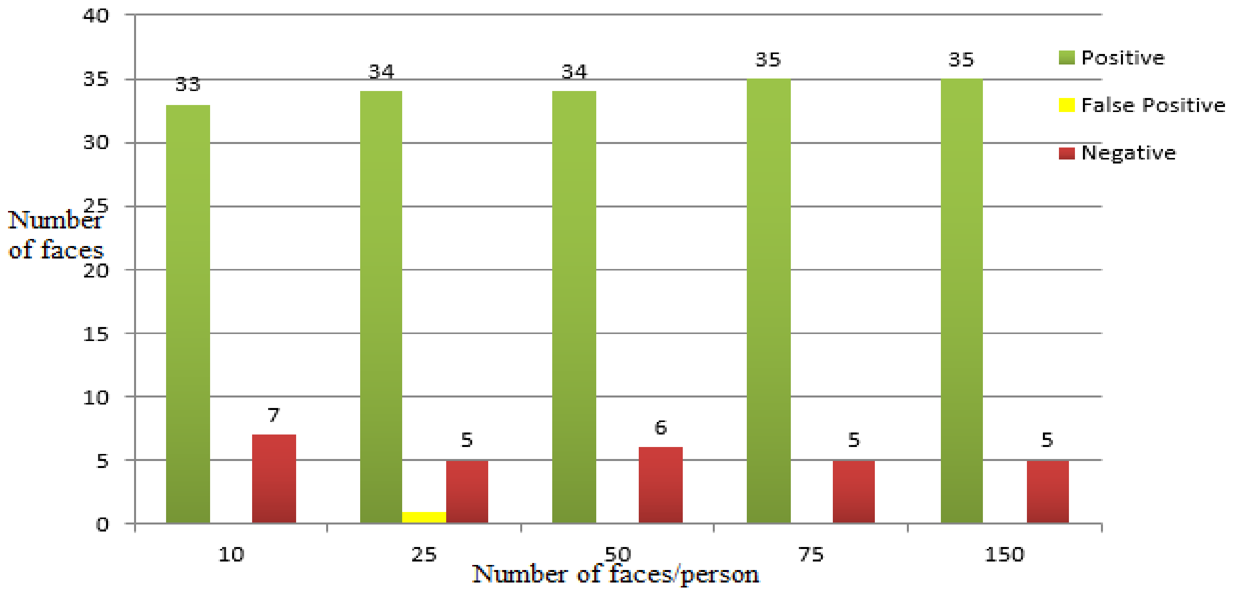


Figure 5.6: Chart representing number of faces vs number of faces per person in the dataset when the same camera was used for training and detection with equalization implemented.

Again, another improvement gained due to unifying the cameras and if the process was to be improved even more, manipulating a threshold value called recognizer confidence (see appendix G line 38) which was used to separate the known and the unknown values for any test images. If the value changed to 60, the following results in table 5.20 are obtained.

Table 5.20: Recognition accuracy for the same camera with equalization implemented and confidence of 60.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	34	0	6	85%
25	35	0	5	87.5%
50	35	0	5	87.5%
75	35	0	5	87.5%
150	36	0	4	90%

The chart in **figure 5.7** visualizes the results in table 5.20.

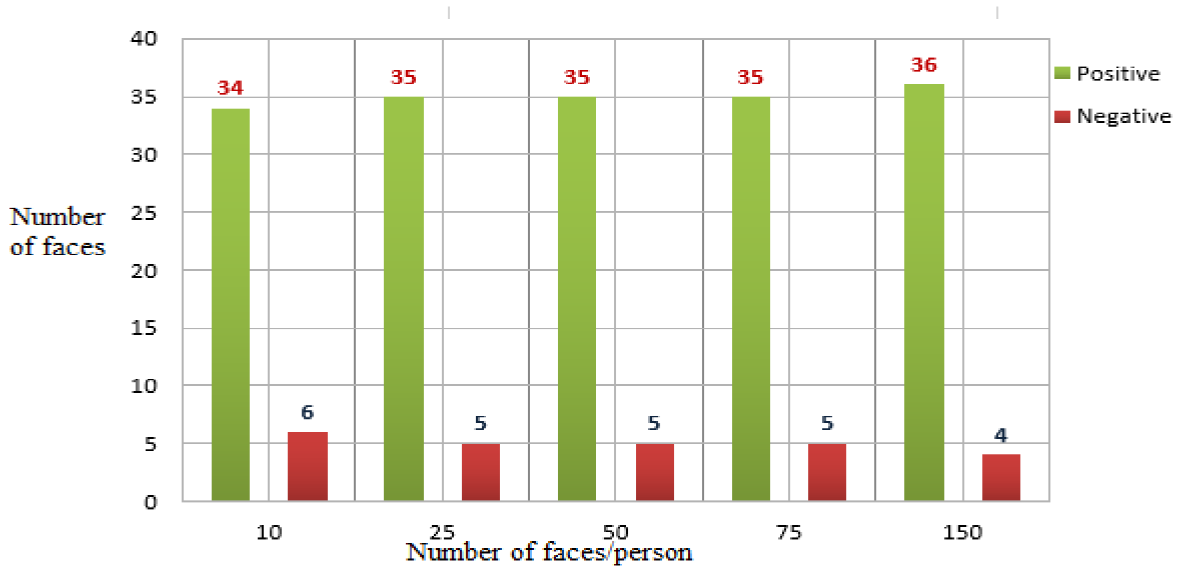


Figure 5.7: Chart representing number of faces vs number of faces per person in the dataset while the same camera was used for training and detection with equalization implemented with confidence of 60.

Since a test for the laptop webcam was done for training and detection, another test was implemented to see the specifications of the raspberry pi camera in the next section.

5.5.3 Detection and training using raspberry pi camera

In this section, raspberry pi camera capabilities were tested, applying same conditions mentioned in earlier, thus the test was implemented with equalization since it obviously provides a reasonable result.

The following table (table 5.21) contains the result obtained when the raspberry pi camera was used for detection and training.

Table 5.21: Recognition accuracy for the same camera with equalization implemented.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	29	1	10	72.5%
25	30	0	10	75%
50	30	0	10	75%
75	30	0	10	75%
150	30	0	10	75%

The chart in **figure 5.8** visualize the results in table 5.21.

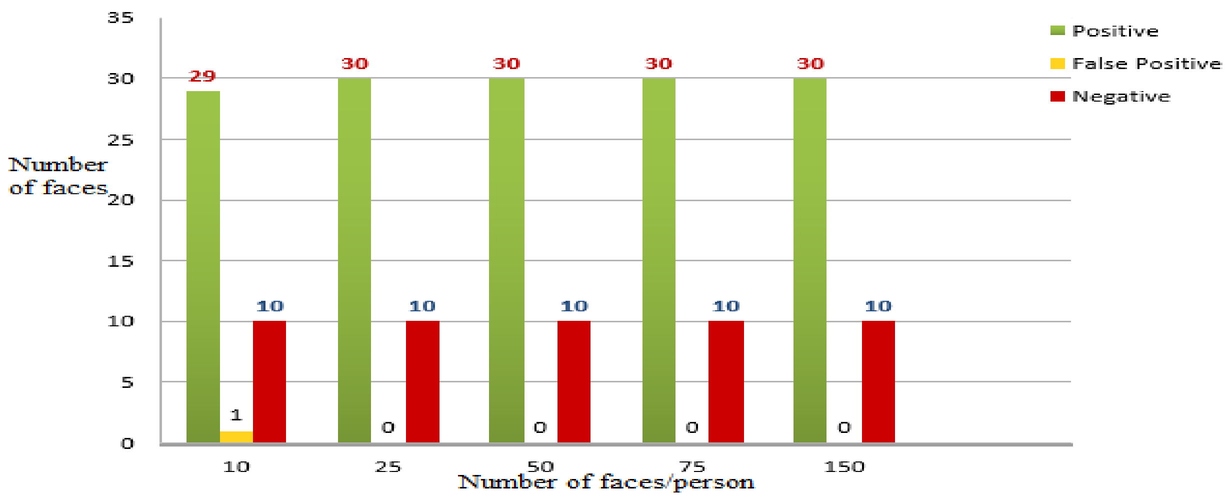


Figure 5.8: Chart representing number of faces vs number of faces per person in the dataset while the same camera was used for training and detection with equalization implemented.

The results are reasonable for the raspberry pi camera in comparison to the laptop camera since the laptop webcam has a higher image resolution.

To improve the obtained results it is possible to manipulate the confidence value, by setting the confidence to 60 yielded the following results.

Table 5.22: Recognition accuracy for the same camera with equalization implemented and confidence of 60.

Number of faces/person	Positive	False positive	Negative	Success percentage
10	29	0	11	72.5%
25	32	0	8	80%
50	33	0	7	82.5%
75	34	0	6	85%
150	36	0	4	90%

The chart in **figure 5.9** visualizes the results in table 5.22.

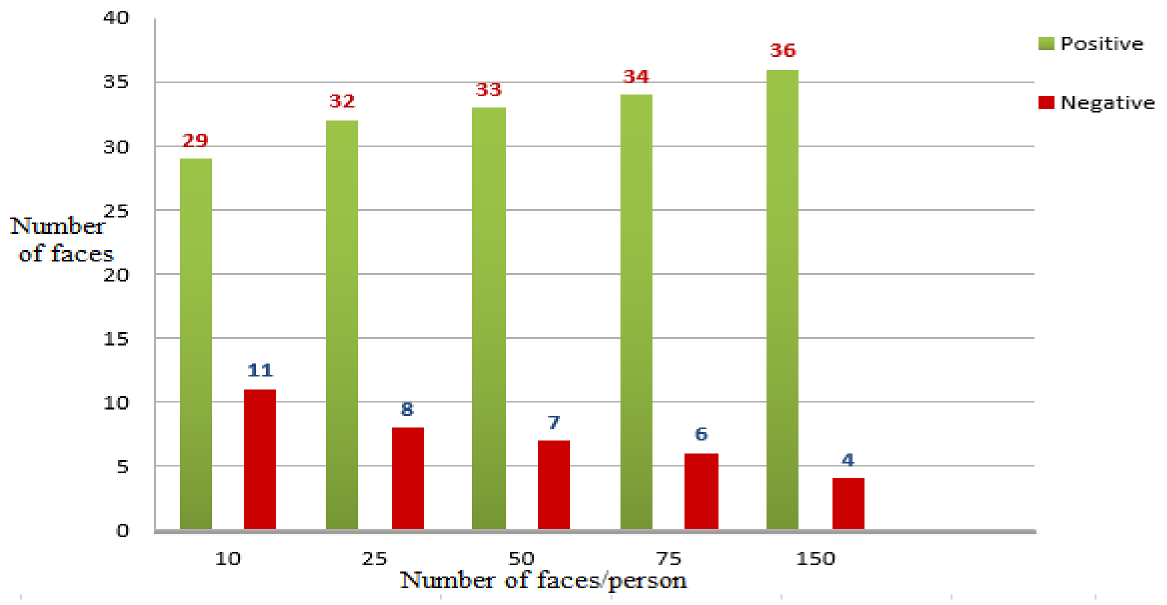


Figure 5.9: Chart representing number of faces vs number of faces per person in the dataset when the same camera was used for training and detection with equalization implemented and confidence of 60.

Better results were gained, that is due to the more restricted confidence value.

5.6 Conclusion

The entire tests that were needed is done, on each section the needed equations and graphs are provided to support a better understanding of the system functionality.

It can be said that the general equations and figures that have been derived are applicable in certain conditions and in a specific environment, applying the system in any other environment will require re-doing the tests to obtain the suitable parameters for the new environment, however the system is functional for this testing environment with acceptable error margins.

In the following chapter, the testing result is discussed and analyzed in addition to the conclusion regarding the objectives that were chosen in chapter one, ending with some future work suggestions.

CHAPTER SIX

Results and Future Work

Chapter Six

Results and Future Work

6.1 Introduction

This chapter concludes the project, in this chapter all the previous results are discussed, followed by a conclusion for the best parameters and conditions that is needed to properly run the system, ending with the suggested future work that will definitely improve the system capabilities and values.

6.2 Results

This section discuss the result obtained from each test in chapter five.

6.2.1 Detection parameters results

After finishing the detection parameters test we can state the following results to be chosen whenever the system is operating, keeping in mind the conditions stated earlier which are:

1. The camera should be placed in front of the person under test, meaning that the person should be approaching the camera; otherwise, unexpected results will be obtained
2. The lighting problem is very complex, but for a better performance, the area of interest should be always lighted.

The test started with detection using raspberry pi camera, training using laptop webcam, the test yielded bad results, and these bad results were mainly due to lighting conditions and different camera specifications.

This problem was partially solved by implementing an equalizer to mitigate the lighting conditions, but still, the different cameras specifications caused a bit too high error rates, to proof that unifying the cameras would help to reduce these errors, a third test was conducted using same camera for training and detection, a better results were obtained then.

Under the previous conditions among the other mentioned in chapter five, the following conclusions were stated in order for the detection to work with the minimum false detection. These values of window size, scale factor, minimum neighbor, equalizer and camera type are:

1. Minimum window size: 20*20
2. Maximum window size: 300*300
3. Scale factor: 1.2
4. Minimum neighbors: 4
5. Implement the equalization.
6. Detection and training camera must be of the same specifications.

These are not ideal values nor assumptions, they could be easily changed if the camera location changed, or the lighting in the room also changed, as mentioned earlier, these values resulted in about 10% false detections only.

To find the number of false detections when using the (20*20 | 300*300) window size the following equations apply

- **Number of false detections** $_{|MN=6}$ $= (1817) \times SF^2 - (4358) \times SF + 2625$
- **Number of false detections** $_{|MN=5}$ $= (-1667) \times SF^2 + (4037) \times SF - 2408$
- **Number of false detections** $_{|MN=6}$ $= (933.3) \times SF^2 - (2240) \times SF + 1363$

Minimum neighbors (MN), Scale factor (SF), the result is the number of false detection out of 100 image.

6.2.2 Transmission time results

As for the transmission time results the system was able to perform load-balancing criteria whenever a new server is added, it was obvious that increasing the number of servers will reduce the needed time to transmit a group of images.

The main problem that took effort to be solved here is how to facilitate the TCP/IP connection using python to achieve the load balancing criteria, and a great result were obtained when testing took place.

The decrease in the transmission time when sending 500 images for example is about 83% when three server were used instead of only one.

The equation that relates the required transmission time and the number of images are:

<u>For one server</u>

- $\boldsymbol{T = (-3.633e-05) \times N^2 + (0.1426) \times N - 0.5623}$

<u>For three servers</u>

- $\boldsymbol{T = (-6.643e-06) \times N^2 + (0.1127) \times N - 0.6345}$

Transmission time (T), Number of images to be transmitted (N).

6.2.3 Recognition processing time

The processing recognition time was extremely useful test; it proved that the increase in the servers number will result in a faster recognition time, this is essential for implementing the project in the real life.

The main problem in this test was the resources shortage, only three available servers, thus a quite well results were obtained, the processing time is reduced by about 48% when three servers were used instead of only one server.

The equations that relate the processing time and the number of images are

<u>For one server</u>

- $\boldsymbol{Tp = 0.4122 \times N - 0.611}$

<u>For Three servers</u>

- $\boldsymbol{Tp = 0.1271 \times N + 1.794}$

Processing time (Tp), Number of processed faces (N)

6.2.4 Recognition parameters test

This test pointed out to the best setup to be used to obtain a better recognition, when talking about the number of faces per person. While keeping in mind that recognition process is a very complex task that requires an almost perfect dataset, i.e. the better the dataset, the better the recognition accuracy.

Another important result from this stage is that in order to get a better recognition accuracy the same camera is needed for both training and detection.

In addition, the lighting problem was solved by implementing an equalizer here in both training and detection phases to ensure a better results.

The desired configuration depends mainly on the setup, in this case, and by using two different cameras while equalization implemented the best number of faces to be used is 50 faces per person, this will yield 82.5 % successful recognition.

6.3 Future Work

After finishing the required objectives in this project, the final suggestions and recommendations for this project are:

1. To improve the system efficiency, more than one camera must to be connected to the raspberry pi computer to increase the coverage area.
2. More research can be done to DetectMultiscale function to relate the minimum factor as a third variable into the detection parameters equation.
3. Multiple raspberry pi – multiple server configuration to provide more efficient system to cover bigger areas.
4. Implement Wi-Fi connection between raspberry pi and the servers either by implementing the new raspberry pi version 3 or by adding Wi-Fi adapter to the existing raspberry pi 2.
5. Improve the detection further more by implementing a Bilateral Filter to smooth out small variations in detected faces.
6. Implement face alignment on both training and detected faces to allow the system to work with the persons even though they are not walking towards the camera.
7. Adding more than three servers to improve the recognition processing time.

8. To improve detection distance, a different harr cascaded classifier needs to be used, specifically building a new cascaded classifier that contains facial images smaller than 20*20 to allow the system to detect their faces.
9. Implement the system at night; this will require using night vision camera.
10. A lighting sensor can be add to manipulate the detected faces equalization by measuring the current light intensity and upon that value a specific equalization is done.
11. Instead of saving the detected faces on the raspberry pi and then sending them, the TCP/IP server python code can be modified to directly send the captured images to the connected clients.

References

[1] Jay Akbar, where your computer goes to die. Available at http://www.dailymail.co.uk/news/article-3049457/Where-computer-goes-die-Shocking-pictures-toxic-electronic-graveyards-Africa-West-dumps-old-PCs-laptops-microwaves-fridges-phones.html (Last Visit Date: 14/12/2016).

[2, 23, 24] Raspberry Pi Official Web site, Available at https://www.raspberrypi.org (Last Visit Date: 14/12/2016).

[3] Swathi. V, Steven Fernandes, Raspberry Pi Based Human Face Detection, published in International of advanced research in computer and communication engineering Vol. 4, Issue 9, September 2015

[4] Víctor Bautista Saiz; Fernan Gallego, GPU: Application for CCTV systems, Published in: Security Technology (ICCST), 2014 International Carnahan Conference.

[5] Krit Janard; Worawan Marurngsith, Accelerating Real-time Face Detection on a Raspberry Pi Telepresence Robot, publish in: Fifth international conference on Innovative Computing Technology 20-22 May 2015

[6] Ping Hsin Lee, Vivek Srinivasan, and Arvind Sundararajan. Face Detection, Final Year Project, Stanford university 2014

[7] Sahni, S. et al. “Reusing personal computer devices - good or bad for the environment?” Sustainable Systems and Technology (ISSST), 2010 IEEE International Symposium on 6-1-2010

[8] Rein-Lien Hsu’, Mohamed Abdel-Mottaleb, and Ani1 K. Jain’ Dept. of Computer Science & Engineering, Michigan State University, Electrical and Computer Engineering Dept., University of Miami.

[9] Rapid Object Detection using a Boosted Cascade of Simple Features, P. Viola Mitsubishi Electr. Res. Labs. Published in: Computer Vision and Pattern Recognition, 2001.

[10] Carson Reynolds and Rosalind W. Picard, Designing for Affective Interactions, MIT Media Laboratory

[11] T. Kohonen, Self-organization and Associative Memory, Springer-Verlag, Berlin, 1989.

[12] J.Shermina, “Face recognition system using multilinear PCA and locality preserving projection,” Proc. 2011 IEEE GCC Conference and Exhibition, pp. 283-286, 2011.

[13] Mohd Noah A. Rahman, Armanadurni Abd Rahman, Afzaal H. Seyal and Nursuziana Kamarudin , Facial Recognition Using Eigenfaces, published in Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference

[14] Eigenfaces from OpenCV documentation from http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#eigenfaces (last visited 14/12/2016)

[15] Fisherfaces from OpenCV documentation from http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#fisherface, (last visited 14/12/2016)

[16] Local Binary Pattern Histograms from OpenCV documentation http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms (last visited 14/12/2016)

[17] Ethernet features from IEEE official website http://www.ieee802.org/3/

[18] TCP/IP protocol suite from http://spectrum.ieee.org/computing/networks/osi-the-internet-that-wasnt

[19] R. R. Ambadas and R. P. Chaudhari, "PIC Microcontroller Universal Board", International Journal of Innovative Technology and Exploring Engineering, Vol.3, Issue.7, December 2013, p.1

[20] Tarun Agarwal, Microcontrollers type and applications, from https://www.elprocus.com/microcontrollers-types-and-applications/ (last visited 14/12/2016)

[21] Raspberry Pi microcontroller, overview, types, software, available at http://www.element14.com/community/docs/DOC-42993/1/raspberry-pi-single-board-computer (last visited 14/12/2016)

[22] Features of Python. https://www.ibiblio.org/swaroopch/byteofpython/read/features-of-python.html. (last visited 14/12/2016)