A PROJECT REPORT

ON

FACE RECOGNITION USING PYTHON

SVS GROUP OF INSTITUTIONS
INTEGRATED CAMPUS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF

DIPLOMA
IN
COMPUTER ENGINEERING

SUBMITTED

BY

T.SOUMYA SRI 21419-CS-039
K.HARISH 21419-CS-018
MD.MAHIMUDA 21419-CS-057
D.GANESH. 21419-CS-056

UNDER THE GUIDANCE OF

K.MOUNIKA

DEPARTMENT OF COMPUTER ENGINEERING

SVS GROUP OF INSTITUTIONS

Hasanparthy, Bheemaram, Hanamkonda, 506015.

TELANGANA

(2020-2023)

A project report

On

“FACE RECOGNITION USING PYTHON”

A Project Work submitted in Partial Fulfillment of the requirements for

DIPLOMA

In

COMPUTER ENGINEERING

Submitted

To

Department

Under the Guidance of

CH. PRIYANKA

DEPARTMENT OF COMPUTER ENGINEERING

SVS GROUP OF INSTITUTIONS

Hasanparthy, Bheemaram, Hanamkonda, 506015.

Telangana

(2020-2023)

SVS GROUP OF INSTITUTIONS

COMPUTER ENGINEERING

SVS GROUP OF INSTITUTIONS
INTEGRATED CAMPUS

CERTIFICATE

This is to certify that this project work entitled **" FACE RECOGNITION USING PYTHON "** is a bonafide work of , T.SOUMYA SRI (21419-CS-039) , K.HARISH (21419-CS-018) ,MD.MAHIMUDA(21419-CS-057) , D.GANESH(22419-CS-056) of final year D.C.M.E 2020-2023. Along with his brachmates submitted in partial fulfillment of the requirements for the award of diploma in Computer Engineering of Telangana State Board Of Technical Education and Training during academic year 2020-2023.

PROJECT GUIDE

K .MOUNIKA

HEAD OF DEPARTMENT

CH. PRIYANKA

PRINCIPAL

A. RANJITH KUMAR

EXTERNAL

CONTENTS

ABSTRACT

In colleges, universities, organizations, schools, and offices, taking attendance is one of the most important tasks that must be done on a daily basis. The majority of the time, it is done manually, such as by calling by name or by roll number. The main goal of this project is to create a Face Recognition-based attendance system that will turn this manual process into an automated one. This project meets the requirements for bringing modernization to the way attendance is handled, as well as the criteria for time management. This device is installed in the classroom, where and student's information, such as name, roll number, class, sec, and photographs, is trained. The images are extracted using Open CV. Before the start of the corresponding class, the student can approach the machine, which will begin taking pictures and comparing them to the qualified dataset. Logitech C270 web camera and NVIDIA Jetson Nano Developer kit were used in this project as the camera and processing board. The image is processed as follows: first, faces are identified using a Haarcascade classifier, then faces are recognized using the LBPH (Local Binary Pattern Histogram) Algorithm, histogram data is checked against an established dataset, and the device automatically labels attendance. An Excel sheet is developed, and it is updated every hour with the information from the respective class instructor.

Keywords: Face Detection, Face Recognition, HaarCascade classifier, NVIDIA Jetson Nano

LIST OF FIGURES:

LIST OF TABLES:

CHAPTER-1
INTRODUCTION

1.1 Project Objective:

Attendance is prime important for both the teacher and student of an educational organization. So it is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room.

Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. So an automatic attendance system can solve all above problems.

There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique and RFID system. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking.

This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions. In addition, the students have to register in the database to be recognized. The enrolment can be done on the spot through the userfriendly interface.

1.2 Background:

Face recognition is crucial in daily life in order to identify family, friends or someone we are familiar with. We might not perceive that several steps have actually taken in order to identify human faces. Human intelligence allows us to receive information and interpret the information in the recognition process. We receive information through the image projected into our eyes, by specifically retina in the form of light. Light is a form of electromagnetic waves which are radiated from a source onto an object and projected to human vision. Robinson-Riegler, G., & Robinson-Riegler, B. (2008) mentioned that after visual processing done by the human visual system, we actually classify shape, size, contour and the texture of the object in order to analyze the information. The analyzed information will be compared to other representations of objects or face that exist in our memory to recognize. In fact, it is a hard challenge to build an automated system to have the same capability as a human to recognize faces. However, we need large memory to recognize different faces, for example, in the Universities, there are a lot of students with different race and gender, it is impossible to remember every face of the individual without making mistakes. In order to overcome human limitations, computers with almost limitless memory, high processing speed and power are used in face recognition systems.

The human face is a unique representation of individual identity. Thus, face recognition is defined as a biometric method in which identification of an individual is performed by comparing real-time capture image with stored images in the database of that person (Margaret Rouse, 2012).

Nowadays, face recognition system is prevalent due to its simplicity and awesome performance. For instance, airport protection systems and FBI use face recognition for criminal investigations by tracking suspects, missing children and drug activities (Robert Silk, 2017). Apart from that, Facebook which is a popular social networking website implement face recognition to allow the users to tag their friends in the photo for entertainment purposes (Sidney Fussell, 2018). Furthermore, Intel Company allows the users to use face recognition to get access to their online account (Reichert, C., 2017). Apple allows the users to unlock their mobile phone, iPhone X by using face recognition (deAgonia, M., 2017).

The work on face recognition began in 1960. Woody Bledsoe, Helen Chan Wolf and Charles Bisson had introduced a system which required the administrator to locate eyes, ears, nose and mouth from images. The distance and ratios between the located features and the common reference points are then calculated and compared. The studies are further enhanced by Goldstein, Harmon, and Lesk in 1970 by using other features such as hair colour and lip thickness to automate the recognition. In 1988, Kirby and Sirovich first suggested principle component analysis (PCA) to solve face recognition problem. Many studies on face recognition were then conducted continuously until today (Ashley DuVal, 2012).

1.3 Problem Statement:

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as calling student names or checking respective identification cards. There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names,

attendance sheet is passed around the classroom during the lecture sessions. The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class. Thus, face recognition attendance system is proposed in order to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance. Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

The paper proposed by Zhao, W et al. (2003) has listed the difficulties of facial identification. One of the difficulties of facial identification is the identification between known and unknown images. In addition, paper proposed by Pooja G.R et al. (2010) found out that the training process for face recognition student attendance system is slow and time-consuming. In addition, the paper proposed by Priyanka Wagh et al. (2015) mentioned that different lighting and head poses are often the problems that could degrade the performance of face recognition based student attendance system.

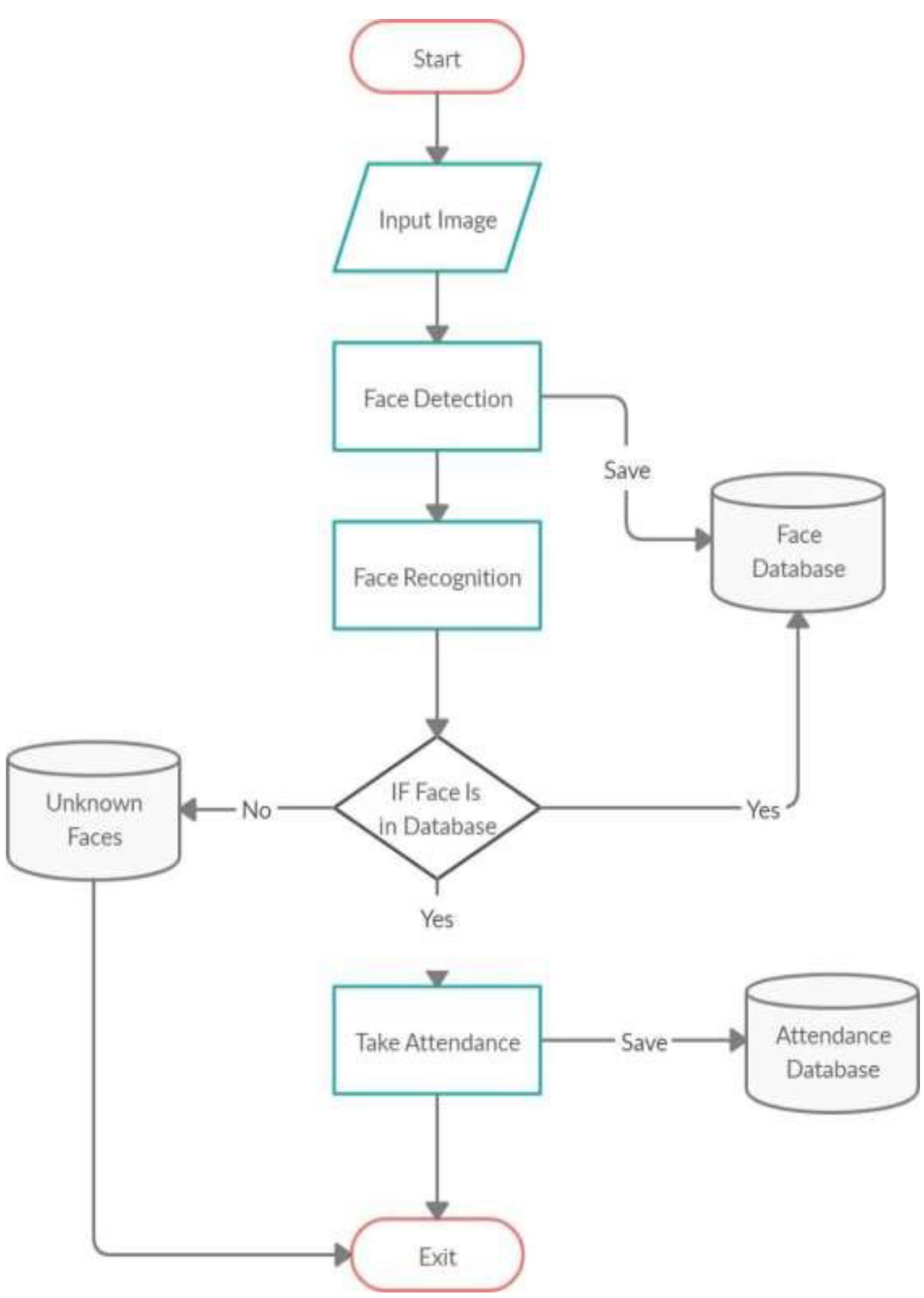
Hence, there is a need to develop a real time operating student attendance system which means the identification process must be done within defined time constraints to prevent omission. The extracted features from facial images which represent the identity of the students have to be consistent towards a change in background, illumination, pose and expression. High accuracy and fast computation time will be the evaluation points of the performance.

1.4 **Aims and Objectives:**

The objective of this project is to develop face recognition attendance system. Expected achievements in order to fulfill the objectives are:

- To detect the face segment from the video frame.
- To extract the useful features from the face detected.
- To classify the features in order to recognize the face detected.
- To record the attendance of the identified student.

1.5 **Flow chart**

- 

1.6 Scope of the project:

We are setting up to design a system comprising of two modules. The first module (face detector) is a mobile component, which is basically a camera application that captures student faces and stores them in a file using computer vision face detection algorithms and face extraction techniques. The second module is a desktop application that does face recognition of the captured images (faces) in the file, marks the students register and then stores the results in a database for future analysis.

CHAPTER 2- LITERATURE REVIEW

2.1 Student Attendance System:

Arun Katara et al. (2017) mentioned disadvantages of RFID (Radio Frequency Identification) card system, fingerprint system and iris recognition system. RFID card system is implemented due to its simplicity. However, the user tends to help their friends to check in as long as they have their friend's ID card. The fingerprint system is indeed effective but not efficient because it takes time for the verification process so the user has to line up and perform the verification one by one. However for face recognition, the human face is always exposed and contain less information compared to iris. Iris recognition system which contains more detail might invade the privacy of the user. Voice recognition is available, but it is less accurate compared to other methods. Hence, face recognition system is suggested to be implemented in the student attendance system.

System Type	Advantage	Disadvantages
RFID card system	Simple	Fraudulent usage
Fingerprint system	Accurate	Time-consuming
Voice recognition system		Less accurate compared to Others
Iris recognition system	Accurate	Privacy Invasion

Table 2.1: Advantages & Disadvantages of Different Biometric System

2.2 Digital Image Processing:

Digital Image Processing is the processing of images which are digital in nature by a digital computer. Digital image processing techniques are motivated by three major applications mainly:

- Improvement of pictorial information for human perception
- Image processing for autonomous machine application

- Efficient storage and transmission.

2.3 Image Representation in a Digital Computer:

An image is a 2-Dimensional light intensity function

$$\mathbf{f}\ (\mathbf{x}\ ,\mathbf{y}\ ) = \mathbf{r}\ (\mathbf{x}\ ,\mathbf{y}\ ) \times \mathbf{i}\ (\mathbf{x}\ ,\mathbf{y}\ ) \text{ - (2.0)}$$

Where, r (x, y) is the reflectivity of the surface of the corresponding image point. i (x,y) Represents the intensity of the incident light. A digital image f(x, y) is discretized both in spatial co-ordinates by grids and in brightness by quantization. Effectively, the image can be represented as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point. These elements are referred to as pixels or pels. Typically following image processing applications, the image size which is used is $\mathbf{2\ 5\ 6\ \times 2\ 5\ 6}$, elements, $\mathbf{6\ 4\ 0\ \times 4\ 8\ 0}$ pels or $\mathbf{1\ 0\ 2\ 4\ \times 1\ 0\ 2\ 4}$ pixels. Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for colored images (because of the three color planes Red, Green and Blue each at 8 bits)[.

2.4 Steps in Digital Image Processing:

Digital image processing involves the following basic tasks:

- Image Acquisition - An imaging sensor and the capability to digitize the signal produced by the sensor.
- Preprocessing – Enhances the image quality, filtering, contrast enhancement etc.
- Segmentation – Partitions an input image into constituent parts of objects.

- Description/feature Selection – extracts the description of image objects suitable for further computer processing.
- Recognition and Interpretation – Assigning a label to the object based on the information provided by its descriptor. Interpretation assigns meaning to a set of labelled objects.
- Knowledge Base – This helps for efficient processing as well as inter module cooperation.

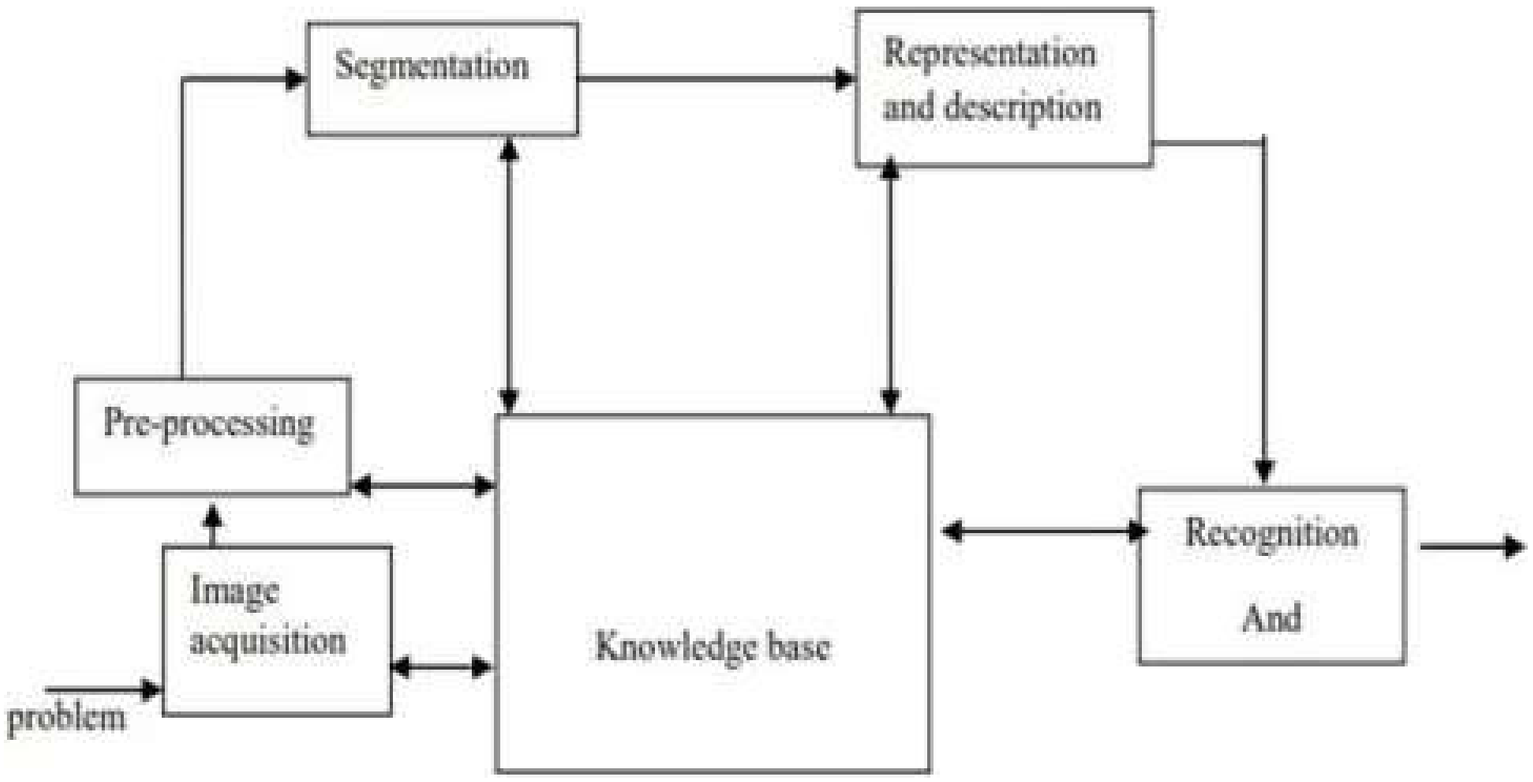


Figure 2.1 : A diagram showing the steps in digital image processing

2.5 Definition of Terms and History:

Face Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content[5].

2.5.1 Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a three dimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images[6].

Face Recognition is therefore simply the task of identifying an already detected face as a known or unknown face and in more advanced cases telling exactly whose face it is[7].

Difference between Face Detection and Face Recognition

Face detection answers the question, Where is the face? It identifies an object as a “face” and locates it in the input image. Face Recognition on the other hand answers the question who is this? Or whose face is it? It decides if the detected face is someone .It can therefore be seen that face detections output (the detected face) is the input to the face recognizer and the face Recognition’s output is the final decision i.e. face known or face unknown.

Face Detection

A face Detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. Face detection can be performed based on several cues: skin color (for faces in color images and videos, motion (for faces in videos), facial/head shape, facial appearance or a combination of these parameters. Most face detection algorithms are appearance based without using other cues. An input image is scanned at all possible locations and scales by a sub window. Face detection is posed as classifying the pattern in the sub window either as a face or a non-face. The face/nonface classifier is learned from face and non-face training examples using statistical learning methods[9]. Most modern algorithms are based on the Viola Jones object detection framework, which is based on Haar Cascades.

Face Detection Method	Advantages	Disadvantages
Viola Jones Algorithm	1. High detection Speed. 2. High Accuracy.	1. Long Training Time. 2.Limited Head Pose. 3.Not able to detect dark faces.
Local Binary Pattern Histogram	1.Simple computation. 2.High tolerance against the monotonic illumination changes.	1.Only used for binary and grey images. 2.Overall performance is inaccurate compared to Viola-Jones Algorithm.
Ada Boost Algorithm	Need not to have any prior knowledge about face structure.	The result highly depends on the training data and affected by weak classifiers.
SMQT Features and SNOW Classifier Method	1. Capable to deal with lighting problem in object detection. 2. Efficient in computation.	The region contain very similar to grey value regions will be misidentified as face.
Neural-Network	High accuracy only if large size of image were trained.	1. Detection process is slow and computation is complex. 2. Overall performance is weaker than Viola-Jones algorithm.

Table 2.2: Advantages & Disadvantages of Face Detection Methods

Viola-Jones algorithm which was introduced by P. Viola, M. J. Jones (2001) is the most popular algorithm to localize the face segment from static images or video frame. Basically the concept of Viola-Jones algorithm consists of four parts. The first part is known as Haar feature, second part is where integral image is created, followed by implementation of Adaboost on the third part and lastly cascading process.

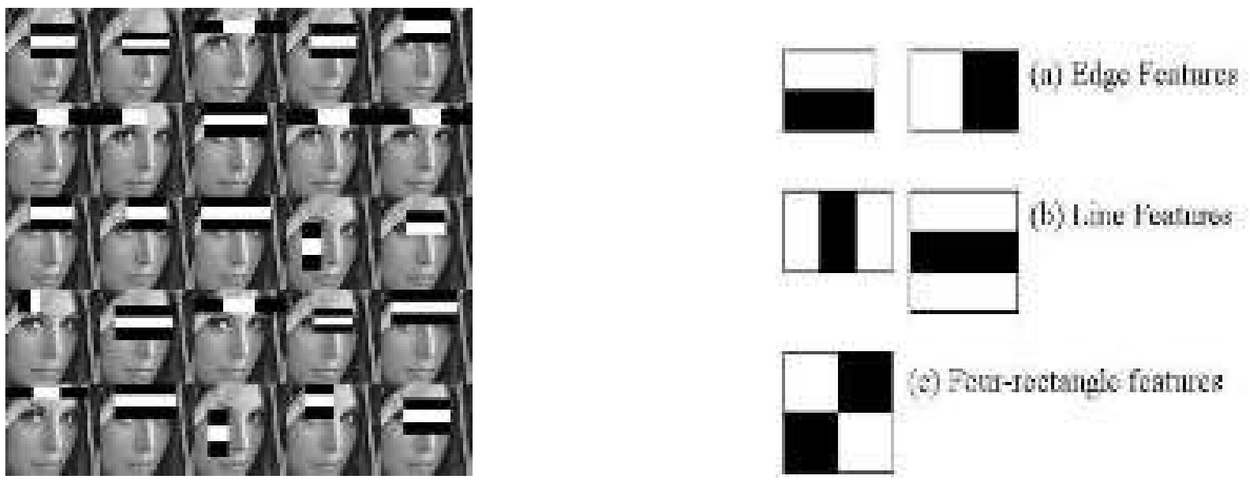


Figure 2.2: Haar Feature

Viola-Jones algorithm analyses a given image using Haar features consisting of multiple rectangles (Mekha Joseph et al., 2016).

In the fig shows several types of Haar features. The features perform as window function mapping onto the image. A single value result, which representing each feature can be computed by subtracting the sum of the white rectangle(s) from the sum of the black rectangle(s).

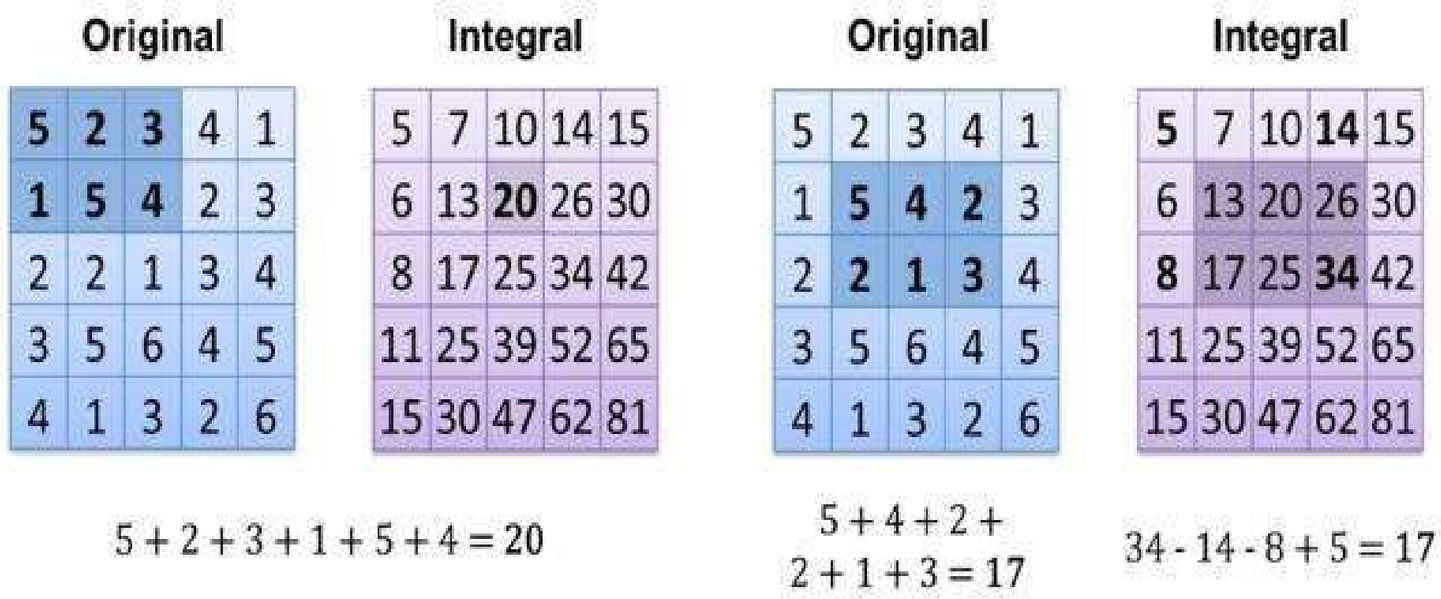


Figure 2.3: Integral of Image

The value of integrating image in a specific location is the sum of pixels on the left and the top of the respective location. In order to illustrate clearly, the value of the integral image at location 1 is the sum of the pixels in rectangle A. The values of integral image at the rest of the locations are cumulative. For instance, the value at location 2 is summation of A and B, (A + B), at location 3 is summation of A and C, (A + C), and at location 4 is summation of all the regions, (A + B + C + D). Therefore, the sum within the D region can be computed with only addition and subtraction of diagonal at location 4 + 1 − (2 + 3) to eliminate rectangles A, B and C.

2.5.2 Local Binary Pattern Histogram

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector.

LBPH algorithm work step by step:

LBPH algorithm work in 5 steps.

1. **Parameters:** the LBPH uses 4 parameters:
    - **Radius:** the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to
      1.
    - **Neighbors:** the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
    - **Grid X:** the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
    - **Grid Y:** the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. **Training the Algorithm:** First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.
3. **Applying the LBP operation:** The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

The image below shows this procedure:

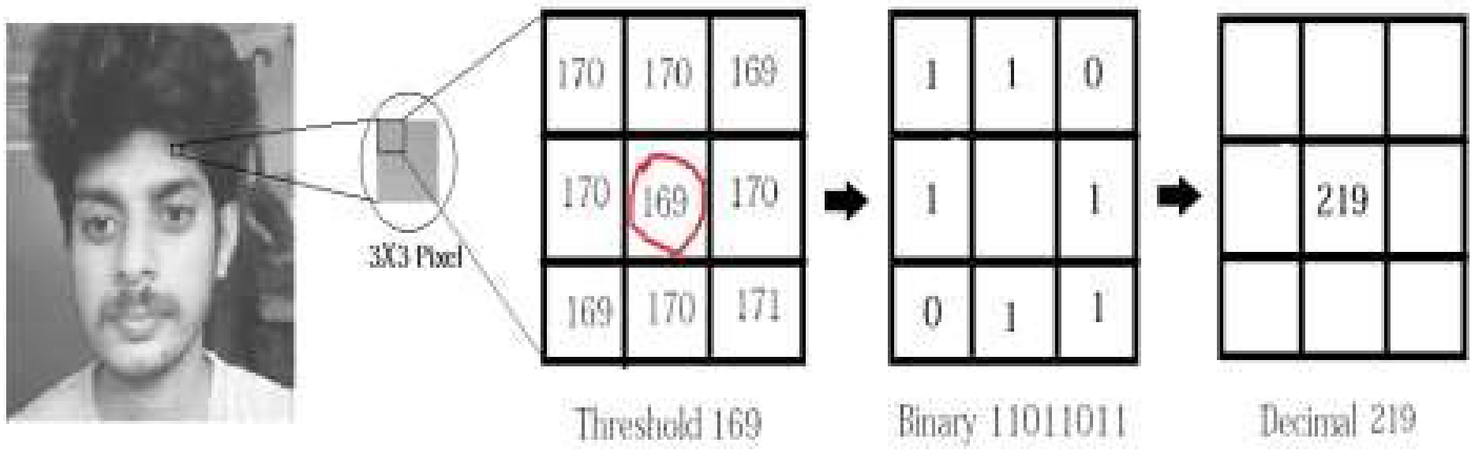


Figure 2.4: LBP Operation

Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).

- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

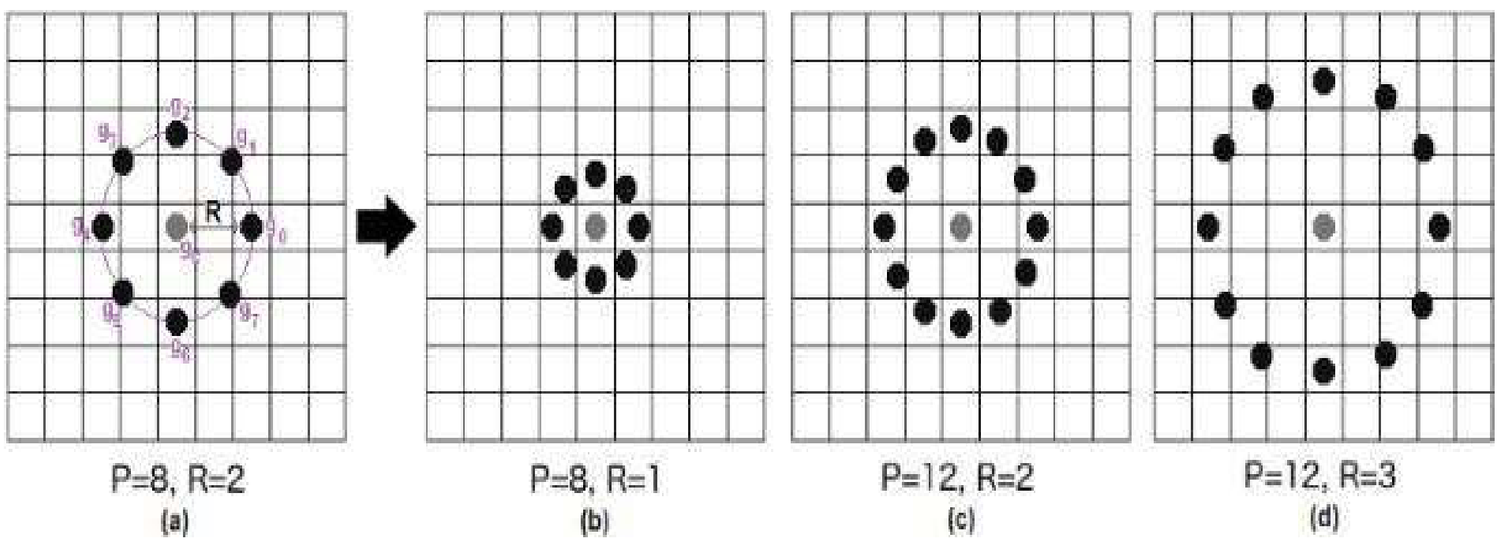


Figure 2.5: The LBP operation Radius Change

It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. **Extracting the Histograms:** Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image:

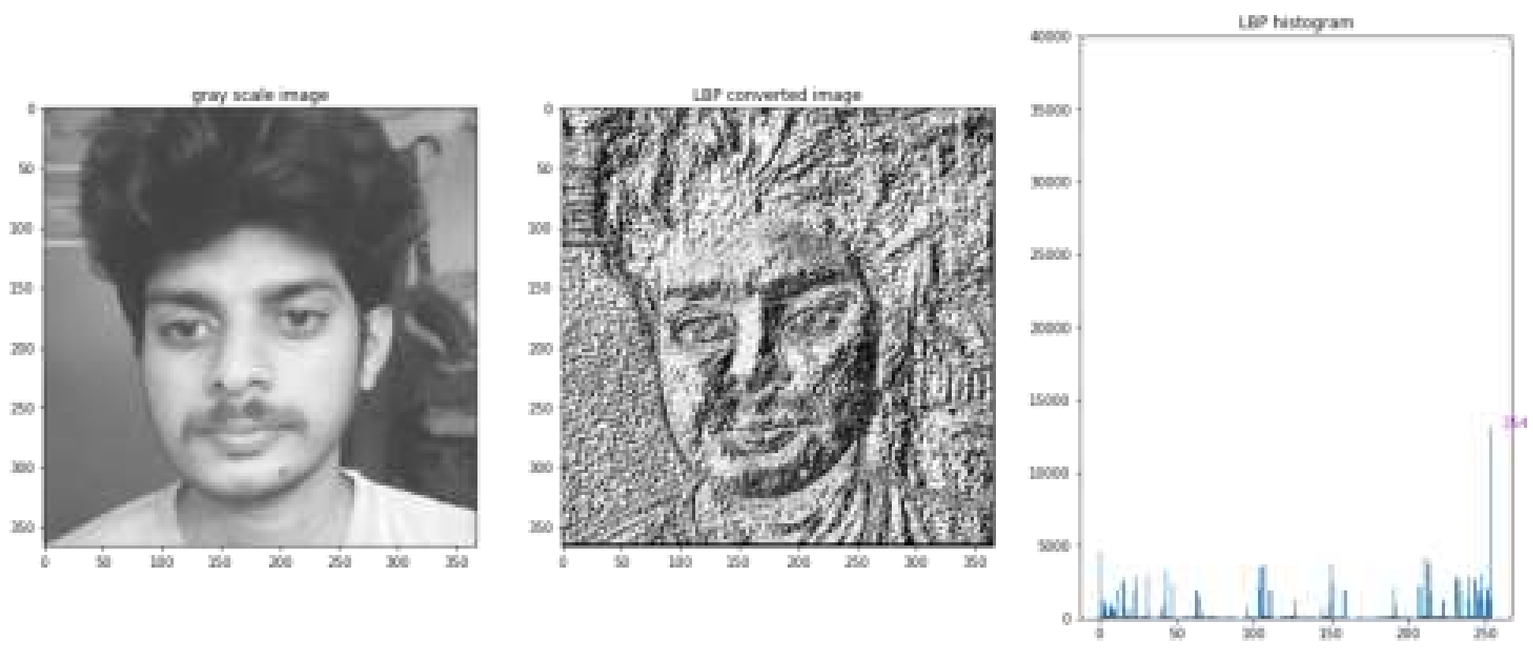


Figure 2.6: Extracting The Histogram

Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

5. **Performing the face recognition:** In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. In this example, we can use the **Euclidean distance** (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

CHAPTER-3 MODAL IMPLEMENTATION AND ANALYSI

3.1 INTRODUCTION:

Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin color and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).After taking the picture the system will compare the equality of the pictures in its database and give the most related result.

We will use NVIDIA Jetson Nano Developer kit, Logitech C270 HD Webcam, open CV platform and will do the coding in python language.

3.2 Modal Implementation:

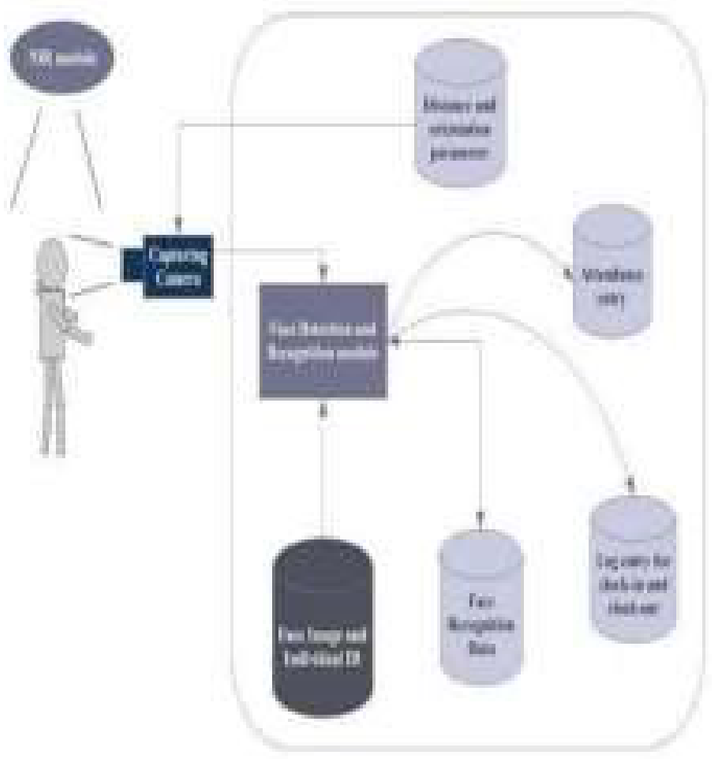


Figure 3.1: Model Implement

The main components used in the implementation approach are open source computer vision library (OpenCV). One of OpenCV's goals is to provide a simpleto-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. OpenCV library contains over 500 functions that span many areas in vision. The primary technology behind Face recognition is OpenCV. The user stands in front of the camera keeping a minimum distance of 50cm and his image is taken as an input. The frontal face is extracted from the image then converted to gray scale and stored. The Principal component Analysis (PCA) algorithm is performed on the images and the eigen values are stored in an xml file. When a user requests for recognition the frontal face is extracted from the captured video frame through the camera. The eigen value is re-calculated for the test face and it is matched with the stored data for the closest neighbour.

3.3 Design Requirements:

We used some tools to build the system. Without the help of these tools it would not be possible to make it done. Here we will discuss about the most important one.

3.3.1 Software Implementation:

1. **OpenCV:** We used OpenCV 3 dependency for python 3. OpenCV is library where there are lots of image processing functions are available. This is very useful library for image processing. Even one can get expected outcome without writing a single code. The library is cross-platform and free for use under the open-source BSD license. Example of some supported functions are given bellow:

- **Derivation**: Gradient/Laplacian computing, contours delimitation
- **Hough transforms:** lines, segments, circles, and geometrical shapes detection
- **Histograms**: computing, equalization, and object localization with back projection algorithm
- **Segmentation**: thresholding, distance transform, foreground/background detection, watershed segmentation
- **Filtering**: linear and nonlinear filters, morphological operations
- **Cascade detectors**: detection of face, eye, car plates
- **Interest points**: detection and matching
- **Video processing:** optical flow, background subtraction, camshaft (object tracking)
- **Photography**: panoramas realization, high definition imaging (HDR), image inpainting

So it was very important to install OpenCV. But installing OpenCV 3 is a complex process. How we did it is given below:

```
#!/bin/bash
#Usage : sudo bash./installopencv.bash
echo OpenCV 3.1.0 Raspbian Jessie auto install script - Thomas Cyria
echo ==========================================================
FILE="/tmp/out.tmp"
GREP="/bin/grep"
if [ "$(id -u)" != "0" ]; then
   echo "This script must be run as root" 1>&2
   exit 1
fi
echo installing core dependencies ...
apt-get -y install cmake pythonl-dev pythonl.4-dev python3-numpy gcc build-essential cmake-curses-gui
apt-get -y install pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlibig-dbg zlibig
zlibig-dev pngtools libtiff5-dev libtiff5 libtiffxx5c3 libtiff-tools libeigen3-dev
echo installing helper apps ...
apt-get -y libav-tools
#apt-get -y ffmpeg libavcodec56 libavformat56
apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs libavcodec-dev libavformat-dev libgstreamer0.10-0-dbg
libgstreamer0.10-0 libgstreamer0.10-dev libxine2-ffmpeg libxine2-dev libxine2-bin libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev libpython3.4 libgtk2.0-dev
echo Removing OpenCV 3.0.0 source...
git clone --branch 3.1.0 --depth 1 https://github.com/Itseez/opencv.git
cd opencv
mkdir release
cd release
echo Preparing compilation, may take a long while...
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=$(python3 -c "import sys; print(sys.prefix)") -D PYTHON_EXECUTABLE=$(which python3) ..
echo Compiling Open CV 3.1.0, may take 2 to 24 hours
make -j4
echo Compilation Ok, installing...
make install
cd ../..
rm -r opencv
echo Completed !
echo You now can use OpenCV 3.1.0 in both Python 2 and Python 3 !
```

Fig 3.2: Installing OpenCV

We copied this script and place it on a directory on our raspberry pi and saved it.

Then through terminal we made this script executable and then ran it.

```
Sudo chmod 755 /myfile/pi/installopencv.bash
sudo /myfile/pi/installopencv.bash
```

these are the command line we used.

2. **Python IDE:** There are lots of IDEs for python. Some of them are PyCharm, Thonny, Ninja, Spyder etc. Ninja and Spyder both are very excellent and free but we used Spyder as it feature- rich than ninja.

3. Spyder is a little bit heavier than ninja but still much lighter than PyCharm. You can run them in pi and get GUI on your PC

```
1. sudo apt-get isntall     spyder
```

through ssh-Y. We installed Spyder through the command line below.

3.3.2 Hardware Implementation:

Figure 3.3 Jetson Board

3.3.2.1 NVIDIA Jetson Nano Developer kit:

NVIDIA® Jetson Nano™ Developer Kit is a small, powerful computer lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts.

It’s simpler than ever to get started! Just insert a microSD card with the system image, boot the developer kit, and begin using the same NVIDIA JetPack SDK used across the entire NVIDIA Jetson™ family of products. JetPack is compatible with NVIDIA’s world-leading AI platform for training and deploying AI software, reducing complexity and effort for developers.

Specifications:

Video Decode	4Kp60 | 2x 4Kp30 | 8x 1080p30 | 18x 720p30 (H.264/H.265)
Connectivity	Gigabit Ethernet, 802.11ac wireless†
Mechanical	100 mm x 80 mm x 29 mm

Table 3.1 Specifications of Jetson Nano Developer kit

The developer kit uses a microSD card as boot device and for main storage. It's important to have a card that's fast and large enough for your projects; the minimum requirement is a 32GB UHS-1 card.

So we used 64Gb microSD card.

GPU	128-core NVIDIA Maxwell™
CPU	Quad-core ARM® A57 @ 1.43 GHz
Memory	2 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (Card not included)
Video Encode	4Kp30 | 4x 1080p30 | 9x 720p30 (H.264/H.265)

Camera	1x MIPI CSI-2 connector
Display	HDMI
USB	1x USB 3.0 Type A,2x USB 2.0 Type A, USB 2.0 Micro-B
Others	40-pin header (GPIO, I2C, I2S, SPI, UART) 12-pin header (Power and related signals, UART) 4-pin Fan header†

Before utilizing it, we have to configure our NVIDIA Jetson Nano Board for Computer Vision and Deep Learning with TensorFlow, Keras, TensorRT, and OpenCV.

The NVIDIA Jetson Nano packs 472GFLOPS of computational horsepower. While it is a very capable machine, configuring it is not easy to configure.

Step #1: Flash NVIDIA's Jetson Nano Developer Kit .img to a microSD for Jetson Nano

In this step, we will download NVIDIA's Jetpack 4.2 Ubuntu-based OS image and flash it to a microSD. You will need the microSD flashed and ready to go to follow along with the next steps. So ensure that you download the "Jetson Nano Developer Kit SD Card image" as shown in the following screenshot:

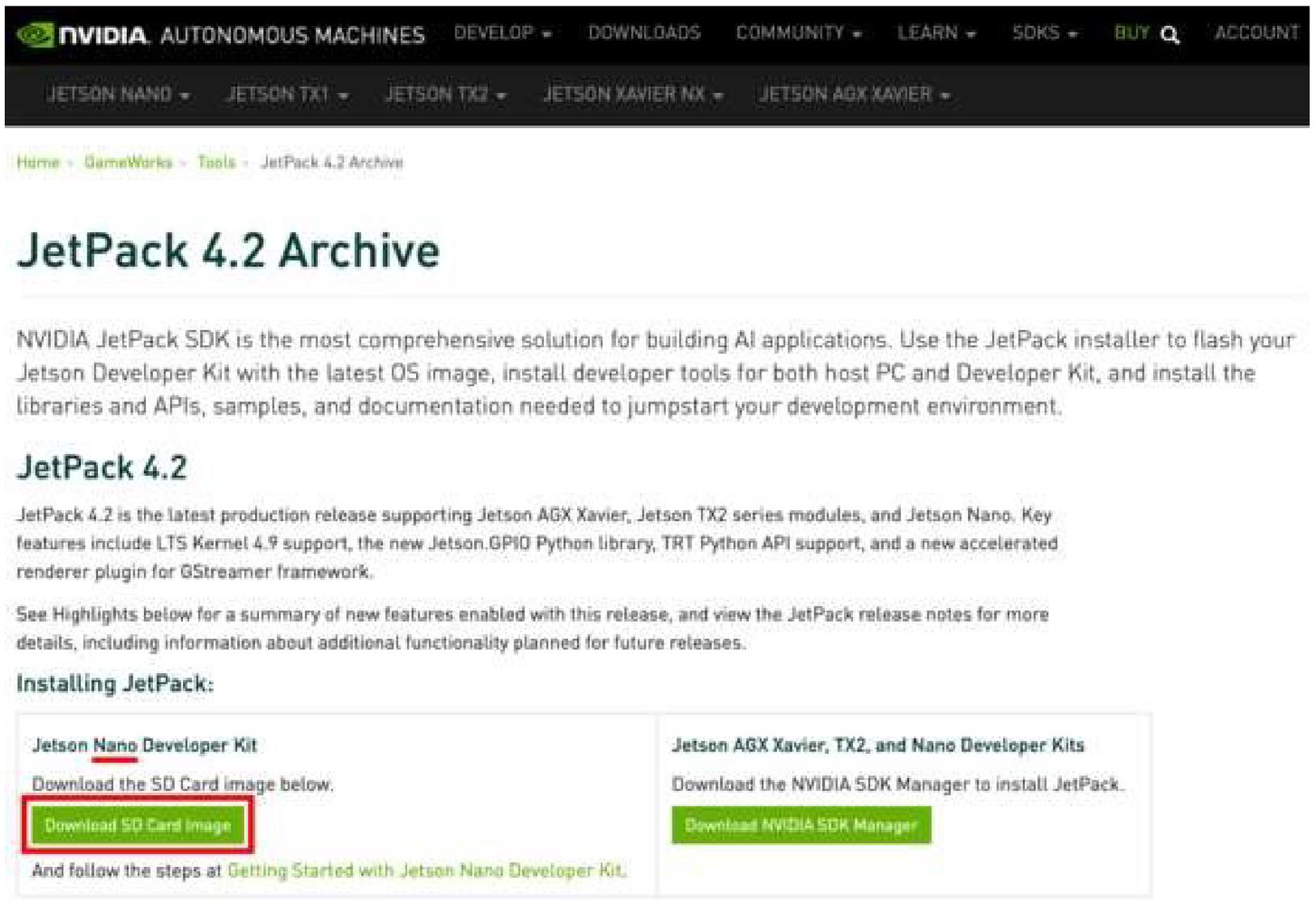


Figure 3.4**:** The first step to configure your NVIDIA Jetson Nano for computer vision and deep learning is to download the **Jetpack SD card image** While your Nano SD image is downloading, go ahead and download and install **balenaEtcher**, a disk image flashing tool:

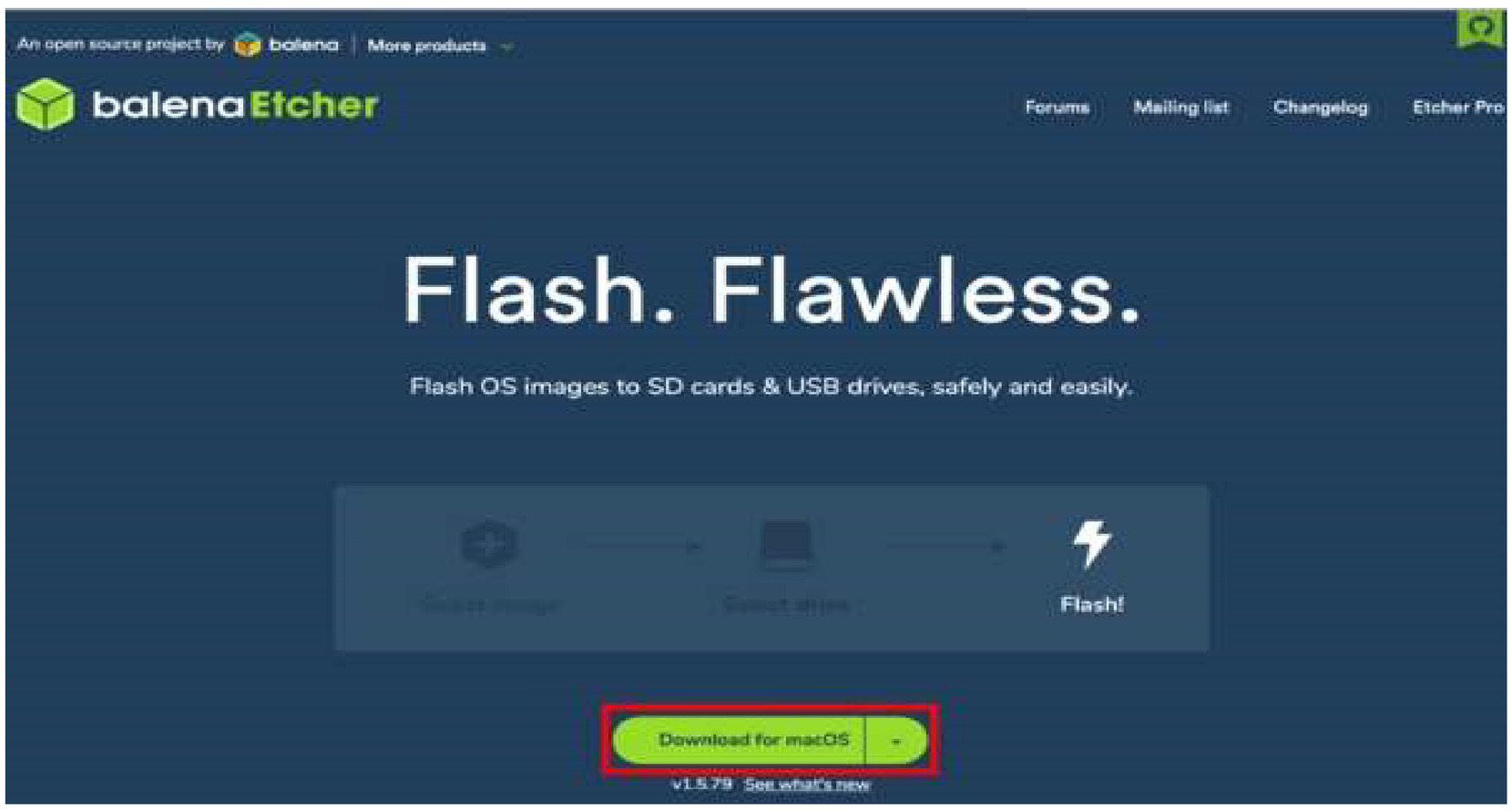


Figure 3.5**:** Download and install balenaEtcher for your OS. You will use it to flash your Nano image to a microSD card.

Once both (1) your Nano Jetpack image is downloaded, and (2) balenaEtcher is installed, you are ready to flash the image to a microSD.

Insert the microSD into the card reader, and then plug the card reader into a USB port on your computer. From there, fire up balenaEtcher and proceed to flash.

Figure 3.6: Flashing NVIDIA's Jetpack image to a microSD card with balenaEtcher is one of the first steps for configuring your Nano for computer vision and deep learning.

When flashing has successfully completed, you are ready to move on to **Step #2**.

Step #2: Boot your Jetson Nano with the microSD and connect to a network ● Insert your microSD into your Jetson Nano as shown in **Figure 4:**

Figure 3.7: To insert your Jetpack-flashed microSD after it has been flashed, find the microSD slot as shown by the red circle in the image. Insert your microSD until it clicks into place.

From there, connect your screen, keyboard, mouse, and network interface. Finally, apply power. Insert the power plug of your power adapter into your Jetson Nano (use the J48 jumper if you are using a 20W barrel plug supply).

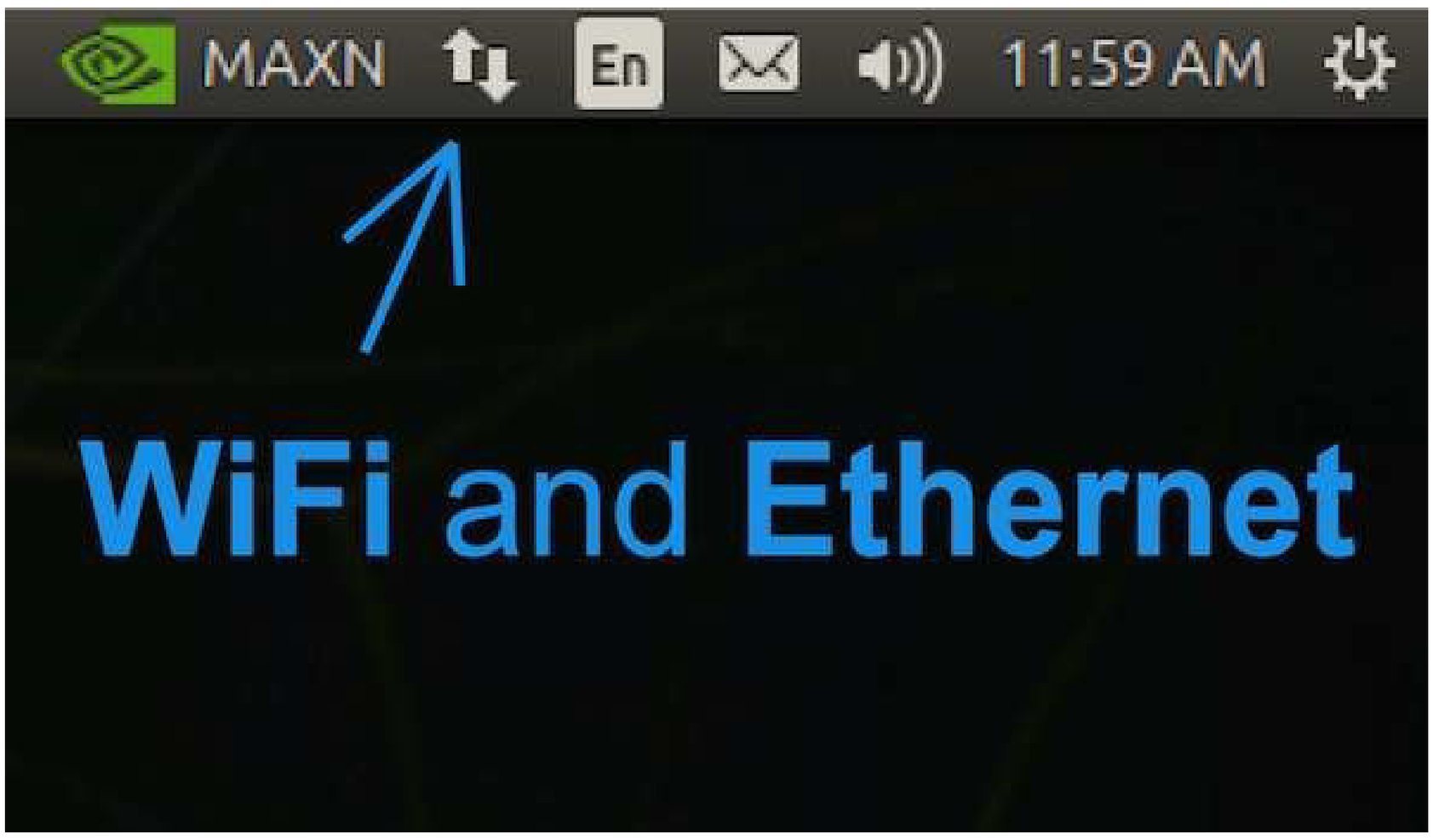


Figure 3.8: Use the icon near the top right corner of your screen to configure networking settings on your NVIDIA Jetson Nano. You will need internet access to download and install computer vision and deep learning software.

Once you see your NVIDIA + Ubuntu 18.04 desktop, you should configure your wired or wireless network settings as needed using the icon in the menubar as shown in **Figure 5.** When you have confirmed that you have internet access on your NVIDIA Jetson Nano, you can move on to the next step.

Step #3: Open a terminal or start an SSH session

In this step we will do one of the following:

1. **Option 1:** Open a terminal on the Nano desktop, and assume that you'll perform all steps from here forward using the keyboard and mouse connected to your Nano

2. **Option 2:** Initiate an SSH connection from a different computer so that we can remotely configure our NVIDIA Jetson Nano for computer vision and deep learning

Both options are equally good.

Option 1: Use the terminal on your Nano desktop

For **Option 1**, open up the application launcher, and select the terminal app. You may wish to right click it in the left menu and lock it to the launcher, since you will likely use it often.

You may now continue to **Step #4** while keeping the terminal open to enter commands.

Option 2: Initiate an SSH remote session

For **Option 2**, you must first determine the username and IP address of your Jetson Nano. On your Nano, fire up a terminal from the application launcher, and enter the following commands at the prompt:

```
$
whoami
nvidia    $
ifconfig
en0: flags=8863 mtu 1500
 options=400
ether 8c:85:90:4f:b4:41
inet6 fe80::14d6:a9f6:15f8:401%en0 prefixlen 64 secured scopeid 0x8
inet6 2600:100f:b0de:1c32:4f6:6dc0:6b95:12 prefixlen 64 autoconf secured
```

```
inet6 2600:100f:b0de:1c32:a7:4e69:5322:7173 prefixlen 64 autoconf
temporary inet 192.168.1.4 netmask 0xffffff00 broadcast 192.168.1.255 nd6
options=201
media: autoselect
status: active
```

Grab your IP address. Then, on a ***separate*** computer, such as your laptop/desktop, initiate an SSH connection as follows:

```
$ ssh nvidia@192.168.1.4
```

Notice how I've entered the username and IP address of the Jetson Nano in my command to remotely connect.

Step #4: Update your system and remove programs to save space

In this step, we will remove programs we don't need and update our system. First, let's set our Nano to use maximum power capacity:

```
$ sudo nvpmodel -m 0
$ sudo jetson_clocks
```

The `nvpmodel` command handles two power options for your Jetson Nano: (1) 5W is mode `1` and (2) 10W is mode `0`. The default is the higher wattage mode, but it is always best to force the mode before running the `jetson_clocks` command.

After you have set your Nano for maximum power, go ahead and remove LibreOffice — it consumes lots of space, and we won't need it for computer vision and deep learning:

```
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
```

From there, let's go ahead and update system level packages:

$ sudo apt-get update && sudo apt-get upgrade

In the next step, we'll begin installing software.

Step #5: Install OpenCV system-level dependencies and other development dependencies

Let's now install OpenCV dependecies on our system beginning with tools needed to build and compile OpenCV with parallelism:

$ sudo apt-get install build-essential pkg-config

$ sudo apt-get install libtbb2 libtbb-dev

Next, we'll install a handful of codecs and image libraries:

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev

$ sudo apt-get install libxvidcore-dev libavresample-dev

$ sudo apt-get install libtiff-dev libjpeg-dev libpng-dev

And then we'll install a selection of GUI libraries:

$ sudo apt-get install python-tk libgtk-3-dev

$ sudo apt-get install libcanberra-gtk-module libcanberra-gtk3-module

Lastly, we'll install Video4Linux (V4L) so that we can work with USB webcams and install a library for FireWire cameras:

$ sudo apt-get install libv4l-dev libdc1394-22-dev

Step #6: Set up Python virtual environments on your Jetson Nano

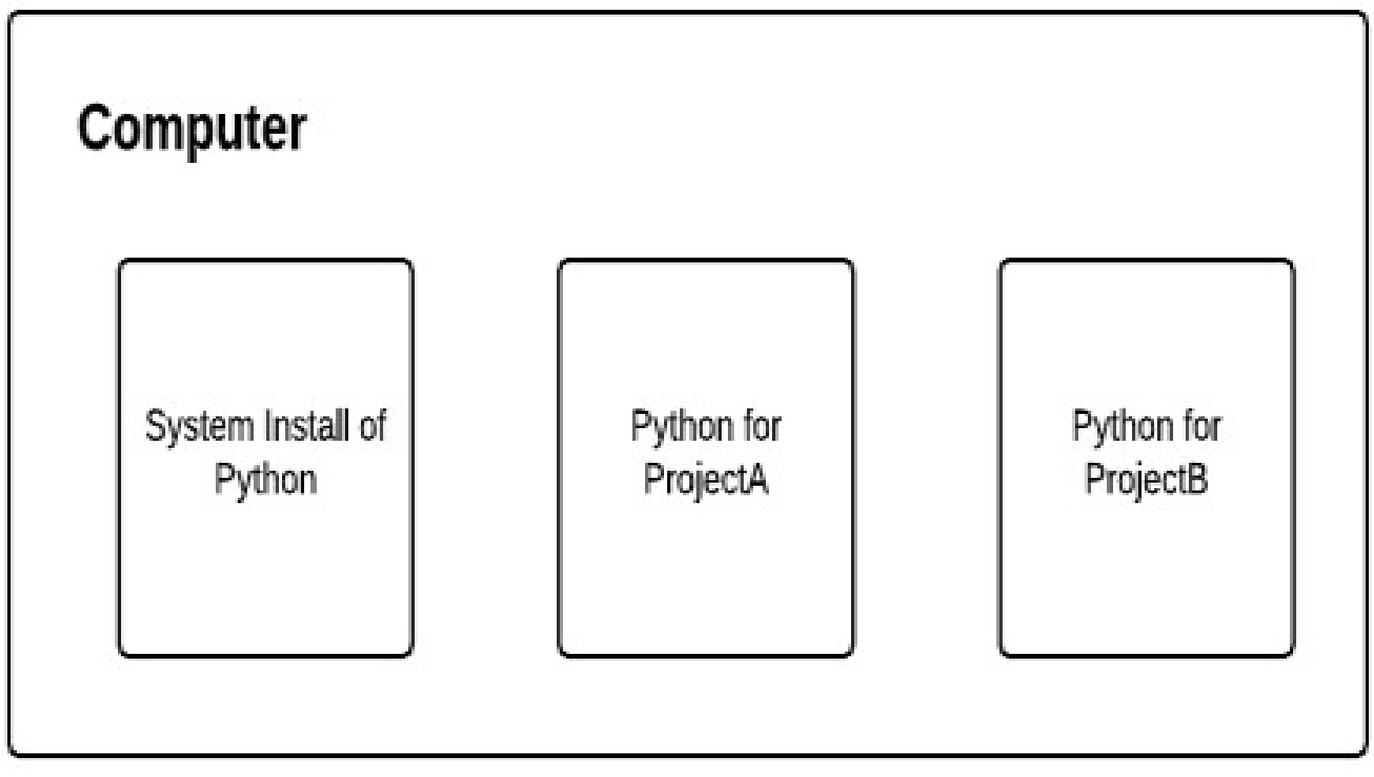


Figure 3.9: Each Python virtual environment you create on your NVIDIA Jetson Nano is separate and independent from the others.

I can't stress this enough: Python virtual environments are a best practice when both developing and deploying Python software projects.

Virtual environments allow for isolated installs of different Python packages. When you use them, you could have one version of a Python library in one environment and another version in a separate, sequestered environment.

In the remainder of this tutorial, we'll create *one* such virtual environment; however, you can create *multiple* environments for your needs after you complete this

Step#6. Be sure to read the RealPython guide on virtual environments if you aren't familiar with them.

First, we'll install the de facto Python package management tool, pip:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
$ rm get-pip.py
```

And then we'll install my favorite tools for managing virtual environments, `virtualenv` and `virtualenvwrapper`:

```
$ sudo pip install virtualenv virtualenvwrapper
```

The `virtualenvwrapper` tool is not fully installed until you add information to your bash profile. Go ahead and open up your `~/.bashrc` with the `nano` ediitor:

```
$ nano ~/.bashrc
```

And then insert the following at the bottom of the file:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

Save and exit the file using the keyboard shortcuts shown at the bottom of the `nano` editor, and then load the bash profile to finish the `virtualenvwrapper` installation:

```
$ source ~/.bashrc
```

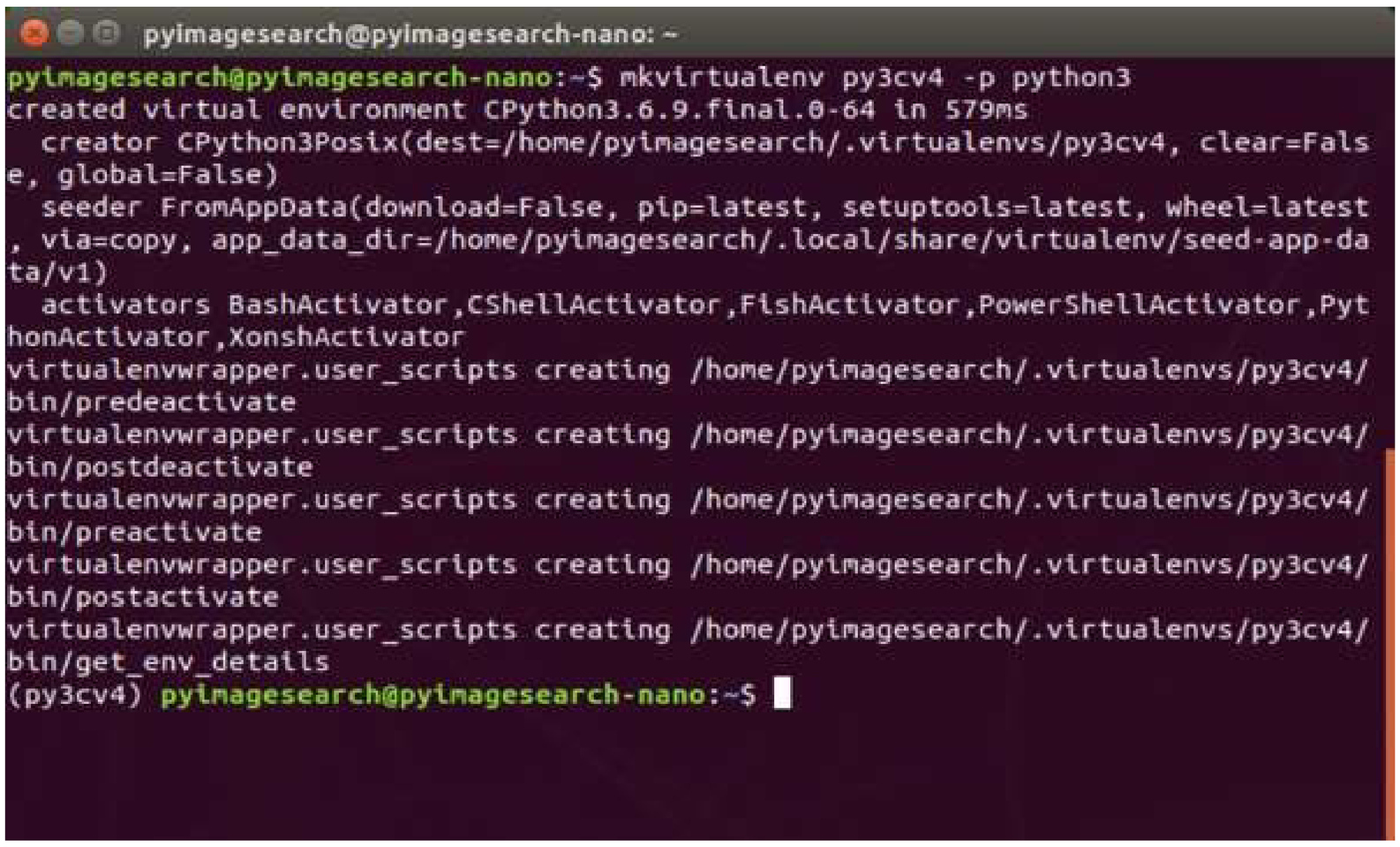


Figure 3.10: Terminal output from the `virtualenvwrapper` setup installation indicates that there are no errors. We now have a virtual environment management system in place so we can create computer vision and deep learning virtual environments on our NVIDIA Jetson Nano.

So long as you don't encounter any error messages, both `virtualenv` and `virtualenvwrapper` are now ready for you to create and destroy virtual environments as needed in **Step #7**.

Step #9: Create your 'py3cv4' virtual environment

This step is dead simple once you've installed `virtualenv` and `virtualenvwrapper` in the previous step. The `virtualenvwrapper` tool provides the following commands to work with virtual environments:

- `mkvirtualenv`

  : Create a Python virtual environment

- `lsvirtualenv`

: List virtual environments installed on your system

- rmvirtualenv

  : Remove a virtual environment

- workon

  : Activate a Python virtual environment

- deactivate

  : Exits the virtual environment taking you back to your system environment

Assuming **Step #6** went smoothly, let's **create a Python virtual environment on our Nano:**

$ mkvirtualenv py3cv4 -p python3

I've named the virtual environment py3cv4 indicating that we will use Python 3 and OpenCV 4. You can name yours whatever you'd like depending on your project and software needs or even your own creativity.When your environment is ready, your bash prompt will be preceded by (py3cv4). If your prompt is not preceded by the name of your virtual environment name, at any time you can use the workon command as follows:

$ workon py3cv4

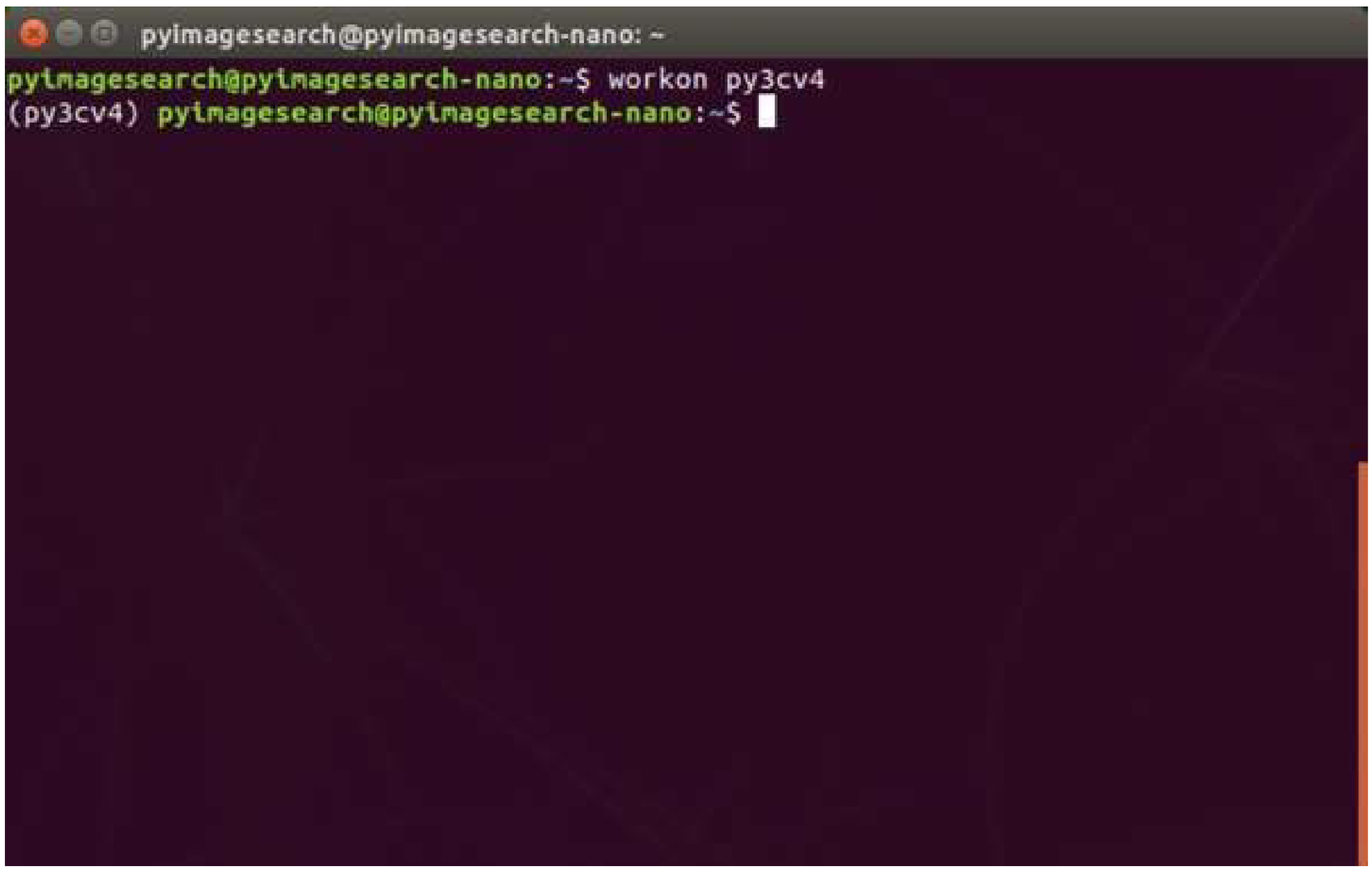


Figure 3.11: Ensure that your bash prompt begins with your virtual environment name for the remainder of this tutorial on configuring your NVIDIA Jetson Nano for deep learning and computer vision.

For the remaining steps , you must be “in” the `py3cv4` virtual environment.

3.3.2.2 Webcam:

Figure 3.12 Web Camera

Specifications:

- Logitech C270 Web Camera (960-000694) supports for NVIDIA jetson nano developer kit.
- The C270 HD Webcam gives you sharp, smooth conference calls (720p/30fps) in a widescreen format. Automatic light correction shows you in lifelike, natural colors.
- Which is suitable to use with the NVIDIA Jetson Nano and NVIDIA Jetson Xavier NX Development Kits.

3.4 Experimental Results:

The step of the experiments process are given below:

Face Detection:

Start capturing images through web camera of the client side: Begin:

- Pre-process the captured image and extract face image
- calculate the eigen value of the captured face image and compared with eigen values of existing faces in the database.
- If eigen value does not matched with existing ones,save the new face image information to the face database (xml file).
- If eigen value matched with existing one then recognition step will done.

End

Face Recognition:

Using PCA algorithm the following steps would be followed in for face recognition:

Begin:

- Find the face information of matched face image in from the database.
- update the log table with corresponding face image and system time that makes completion of attendance for an individua students.

End

This section presents the results of the experiment conducted to capture the face into a grey scale image of 50x50 pixels.

Test data	Expected Result	Observed Result	Pass/ Fail
OpenCAM_CB()	Connects with the installed camera and starts playing.	Camera started.	pass
LoadHaar Classifier()	Loads the HaarClassifier Cascade files for frontal face	Gets ready for Extraction.	Pass
ExtractFace()	Initiates the Paul-Viola Face extracting Frame work.	Face extracted	Pass
Learn()	Start the PCA Algorithm	Updates the facedata. xml	Pass
Recognize()	It compares the input face with the saved faces.	Nearest face	Pass

Table 3.2 Experimental Results-1

Here is our data set sample.

Figure 3.13 : Dataset sample

Face Orientations	Detection Rate	Recognition Rate
0º (Frontal face)	98.7 %	95%
18º	80.0 %	78%
54º	59.2 %	58%
72º	0.00 %	0.00%
90º(Profile face)	0.00 %	0.00%

Table 3.3 Experimentaal Results-2

We performed a set of experiments to demonstrate the efficiency of the proposed method. 30 different images of 10 persons are used in training set. Figure 3 shows a sample binary image detected by the ExtractFace() function using Paul-Viola Face extracting Frame work detection method.

CHAPTER 4 -CODE IMPLEMENTATION

4.1 Code Implementation:

All our code is written in Python language. First here is our project directory structure and files.

```
FRASJN
    |__[Attendance]
    |__[ImagesUnknown]
    |__[StudentDetails]
    |__[TrainingImage]
    |__[TrainingImage]
    |__main.py
    |__automail.py
    |__CaptureImage.py
    |__check_camera.py
    |_haarcascade_frontalface_default.xml
    |_____recognize.py
    |        requirements.txt
```

All those file in the project directory.

```
Note: The names inside square brackets [“folder name”] indicate it is a folder.
[Attendance]              => It contains all the attendance sheets saved after taking attendance.
[ImagesUnknown]   => Unknown images are placed inside this folder to avoid false positives.
[StudentDetails]       => Here we place Studentdetails.csv file to use while recognizing faces. [Trainingimage]      => After capture dataset of a student, all his/her images are stored here.
```

4.1.1 main.py

All the work will be done here, Detect the face ,recognize the faces and take attendance.

```
import os  # accessing the os functions
import check_camera
import Capture_Image
import Train_Image
import Recognize


# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program


print("\t**********************************************************")
    print("\t***** Face Recognition Attendance System using jetson
nano*****")

print("\t**********************************************************")


# creating the user main menu function

def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")

    while True:
        try:
            choice = int(input("Enter Choice: "))

            if choice == 1:
                checkCamera()
                break
            elif choice == 2:
    CaptureFaces()
    break
elif choice == 3:
    Trainimages()
    break
elif choice == 4:
    RecognizeFaces()
    break
elif choice == 5:
    os.system("py automail.py")
    break
    mainMenu()
elif choice == 6:
```

```
import os  # accessing the os functions
import check_camera
import Capture_Image
import Train_Image
import Recognize


# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program


print("\t**********************************************************")
    print("\t***** Face Recognition Attendance System using jetson
nano*****")

print("\t**********************************************************")


# creating the user main menu function

def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")

    while True:
        try:
            choice = int(input("Enter Choice: "))

            if choice == 1:
                checkCamera()
                break
            elif choice == 2:
    CaptureFaces()
    break
elif choice == 3:
    Trainimages()
    break
elif choice == 4:
    RecognizeFaces()
    break
elif choice == 5:
    os.system("py automail.py")
    break
    mainMenu()
elif choice == 6:
```

```
import os  # accessing the os functions
import check_camera
import Capture_Image
import Train_Image
import Recognize


# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program


print("\t**********************************************************")
    print("\t***** Face Recognition Attendance System using jetson
nano*****")

print("\t**********************************************************")


# creating the user main menu function

def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")

    while True:
        try:
            choice = int(input("Enter Choice: "))

            if choice == 1:
                checkCamera()
                break
elif choice == 2:
    CaptureFaces()
    break
elif choice == 3:
    Trainimages()
    break
elif choice == 4:
    RecognizeFaces()
    break
elif choice == 5:
    os.system("py automail.py")
    break
    mainMenu()
elif choice == 6:
    print("Thank You")
```

```
import os  # accessing the os functions
import check_camera
import Capture_Image
import Train_Image
import Recognize


# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program


print("\t**********************************************************")
    print("\t***** Face Recognition Attendance System using jetson
nano*****")

print("\t**********************************************************")


# creating the user main menu function

def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")

    while True:
        try:
            choice = int(input("Enter Choice: "))

            if choice == 1:
                checkCamera()
                break
elif choice == 2:
    CaptureFaces()
    break
elif choice == 3:
    Trainimages()
    break
elif choice == 4:
    RecognizeFaces()
    break
elif choice == 5:
    os.system("py automail.py")
    break
    mainMenu()
elif choice == 6:
    print("Thank You")
```

```
                break
            else:
                print("Invalid Choice. Enter 1-6")
                mainMenu()
        except ValueError:
            print("Invalid Choice. Enter 1-6\n Try Again")


# -----------------------------------------------------
# calling the camera test function from check camera.py file

def checkCamera():
    check_camera.camer()
    key = input("Press Enter to return main menu")
    mainMenu()


# ---------------------------------------------------------
# calling the take image function form capture image.py file

def CaptureFaces():
    Capture_Image.takeImages()
    key = input("Press Enter to return main menu")
    mainMenu()


# ---------------------------------------------------------------
# calling the train images from train_images.py file

def Trainimages():
    Train_Image.TrainImages()
    key = input("Press Enter to return main menu")
    mainMenu()


# -------------------------------------------------------------------
# calling the recognize_attendance from recognize.py file

def RecognizeFaces():
    Recognize.recognize_attendence()
    key = input("Press Enter to return main menu")
    mainMenu()


# ---------------main driver ------------------
mainMenu()
```

4.1.2 automail.py

In this project we add an extra feature called auto mail. It can automatically sent the attendance file to specific mail. Auto mail code given below.

```
import yagmail
import os
import datetime
date = datetime.date.today().strftime("%B %d, %Y")
path = 'Attendance'
os.chdir(path)
files = sorted(os.listdir(os.getcwd()), key=os.path.getmtime)
newest = files[-1]
filename = newest
sub = "Attendance Report for " + str(date)
# mail information
yag = yagmail.SMTP(user = "SenderEmailAddress@domain.com", password =
"Password of that Account should be inserted here")

# sent the mail
yag.send(
    to="Receiver email adress",
    subject= sub, # email subject
    contents= "None",  # email body
    attachments= filename  # file attached
)
print("Email Sent!")
```

4.1.3 Capture_Image.py

This capture_image.py will collect the data set of a student and add his/her name in tha StudentsDetails.csv

```
import csv

import cv2
import os


# counting the numbers


def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False


# Take image function

def takeImages():


    Id = input("Enter Your Id: ")
    name = input("Enter Your Name: ")

    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0

        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5,
minSize=(30,30),flags = cv2.CASCADE_SCALE_IMAGE)
            for(x,y,w,h) in faces:
                cv2.rectangle(img, (x, y), (x+w, y+h), (10, 159, 255), 2)
    #incrementing sample number
    sampleNum = sampleNum+1
    #saving the captured face in the dataset folder TrainingImage
    cv2.imwrite("TrainingImage" + os.sep +name + "."+Id + '.' +
                str(sampleNum) + ".jpg", gray[y:y+h, x:x+w])
    #display the frame
    cv2.imshow('frame', img)
#wait for 100 miliseconds
if cv2.waitKey(100) & 0xFF == ord('q'):
```

```
                break
            # break if the sample number is more than 100
            elif sampleNum > 100:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id + " Name : " + name
        row = [Id, name]
        with open("StudentDetails"+os.sep+"StudentDetails.csv", 'a+') as
csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
    else:
        if(is_number(Id)):
            print("Enter Alphabetical Name")
        if(name.isalpha()):
            print("Enter Numeric ID")
```

4.1.4 checkcamera.py

This checkcamra.py will check weather the camera is correctly connected or not, if connected whether the face is detecting or not.

```
def camer():
    import cv2
    # Load the cascade
    face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    # To capture video from webcam.
    cap = cv2.VideoCapture(0)
    cnt = 0
    while True:
        # Read the frame
        _, img = cap.read()
        # Convert to grayscale
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # Detect the faces
        faces = face_cascade.detectMultiScale(gray, 1.3, 5, minSize=(30,
30),flags = cv2.CASCADE_SCALE_IMAGE)
        # Draw the rectangle around each face
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (10,159,255), 2)
        # Display
        cv2.imwrite("Avinash"+str(cnt)+".jpg",gray)
        cv2.imshow('Webcam Check', img)
        cnt += 1
        # Stop if escape key is pressed
        if (cv2.waitKey(1) & 0xFF == ord('q')) or cnt==100:
            break
# Release the VideoCapture object
cap.release()
cv2.destroyAllWindows()
```

4.1.5 Train_Image.py

All the images in the Training Image folder will be accessed here and a model is created by using this trainimage.py file.

```
import os
import time
import cv2
import numpy as np
from PIL import Image
from threading import Thread
# -------------- image labesl ----------------------
def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    # print(imagePaths)
    # create empth face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the
images
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids
# ---------- train images function --------------
def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, Id = getImagesAndLabels("TrainingImage")
    Thread(target = recognizer.train(faces, np.array(Id))).start()
    # Below line is optional for a visual counter effect
    Thread(target = counter_img("TrainingImage")).start()
    recognizer.save("TrainingImageLabel"+os.sep+"Trainner.yml")
    print("All Images")
# Optional, adds a counter for images trained (You can remove it)
def counter_img(path):
    imgcounter = 1
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    for imagePath in imagePaths:
        print(str(imgcounter) + " Images Trained", end="\r")
        time.sleep(0.008)
        imgcounter += 1
    return
```

4.1.6 Recognize.py

When this Recognize.py file is executed, camera will be opened and it will recognize all the students present in this Students.csv file and those who are present it will mark attendance automatically and save in Attendance folder with date and time.

```
import datetime
import os
import time
import cv2
import pandas as pd
#-------------------------
def recognize_attendence():
    recognizer = cv2.face.LBPHFaceRecognizer_create()  #
cv2.createLBPHFaceRecognizer()
    recognizer.read("TrainingImageLabel"+os.sep+"Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    df = pd.read_csv("StudentDetails"+os.sep+"StudentDetails.csv")
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', 'Name', 'Date', 'Time']
    attendance = pd.DataFrame(columns=col_names)
    # Initialize and start realtime video capture
    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cam.set(3, 640)  # set video width
    cam.set(4, 480)  # set video height
    # Define min window size to be recognized as a face
    minW = 0.1 * cam.get(3)
    minH = 0.1 * cam.get(4)
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5,minSize =
(int(minW), int(minH)),flags = cv2.CASCADE_SCALE_IMAGE)
        for(x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x+w, y+h), (10, 159, 255), 2)
            Id, conf = recognizer.predict(gray[y:y+h, x:x+w])
            if conf < 100:
                aa = df.loc[df['Id'] == Id]['Name'].values
                confstr = "  {0}%".format(round(100 - conf))
                tt = str(Id)+"-"+aa
            else:
                Id = '  Unknown  '
                tt = str(Id)
                confstr = "  {0}%".format(round(100 - conf))
            if (100-conf) > 67:
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-
%d')
                timeStamp =
datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa = str(aa)[2:-2]
                attendance.loc[len(attendance)] = [Id, aa, date, timeStamp]
```

```
tt = str(tt)[2:-2]
if(100-conf) > 67:
    tt = tt + " [Pass]"
    cv2.putText(im,str(tt),(x+5,y-5),font,1,(255,255,255),2)
else:
    cv2.putText(im,str(tt),(x+5,y-5),font,1,(255,255,255),2)
if (100-conf) > 67:
    cv2.putText(im,str(confstr),(x+5,y+h-5),font,1,(0,255,0),1)
elif (100-conf) > 50:
    cv2.putText(im,str(confstr),(x+5,y+h-5),font,1,(0,255,255),1)
                else:
                    cv2.putText(im,str(confstr),(x+5,y+h-5),font,1,(0,0,255),1)
            attendance = attendance.drop_duplicates(subset=['Id'], keep='first')
            cv2.imshow('Attendance', im)
            if (cv2.waitKey(1) == ord('q')):
                break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour, Minute, Second = timeStamp.split(":")
    fileName = "Attendance"+os.sep+"Attendance_"+date+"_"+Hour+"-"+Minute+"-
"+Second+".csv"
    attendance.to_csv(fileName, index=False)
    print("Attendance Successful")
    cam.release()
    cv2.destroyAllWindows()
```

4.1.7 requirements.txt

This file consists all the required files to be install before executing the codes.

```
pip install opencv-contrib-python
pip  install  numpy  pip  install
pandas  pip  install  Pillow  pip
install pytest-shutil
pip install python-csv pip
install yagmail
```

We can make use of the above commands or we can run a simple command with thw requirements.txt file

```
pip install -r requirements.txt
```

The text file consists:

opencv-contrib-pyth
on numpy pandas
Pillow pytest-shutil
python-csv yagmail

4.2 Sample Images:

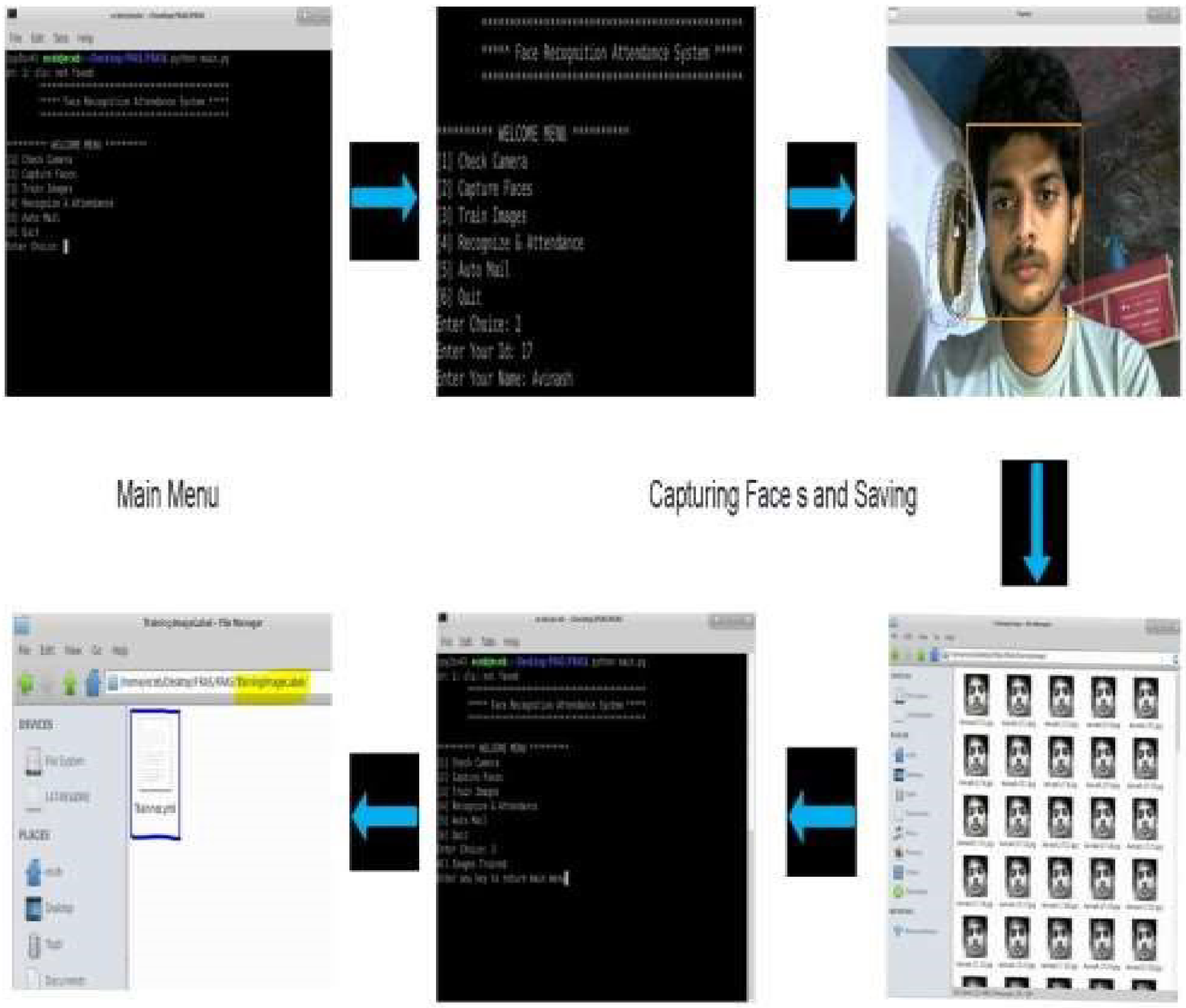


Figure 4.1 :Output Images

CHAPTER5-PERFORMANCE ANALYSIS

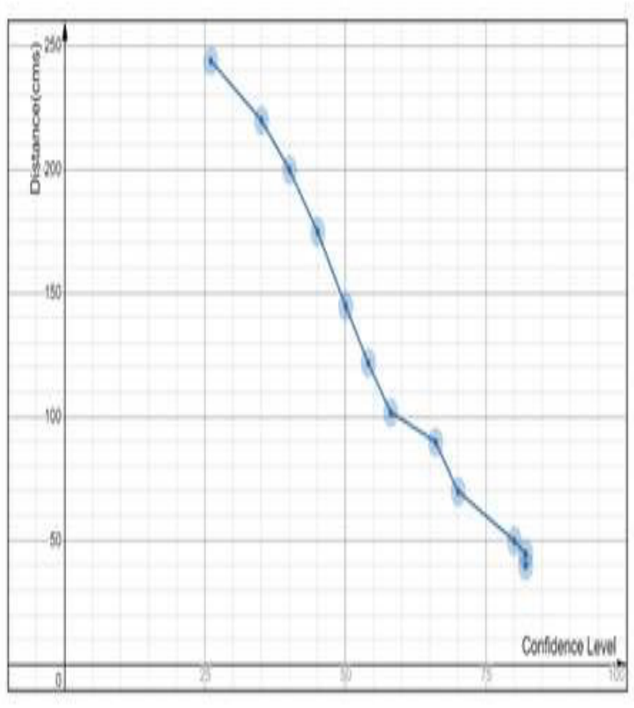
5.1 Introduction:

We conducted a series of experiments to illustrate the system performance under different situations. By carrying out those tests, we were able to get the graph shown above (Distance vs Confidence Level). We may deduce from the graph that when the face is closer to the camera, the confidence level is higher, and vice versa. Therefore, by keeping a threshold for confidence level, we can mark attendance to the person according to the threshold.

5.2 Analysis:

Here we consider one constant parameter intensity of light . we performed different experiments on different distance and different angles. we observed the confidence level at the different positions by gradually increasing the distance .we plotted the graph using the x and y coordinates by considering the x values as the confidence level or accuracy rate. and y values as the distance (cms).

5.3 Flow Chart:



x	y
26	244
35	220
40	200
45	175
50	145
54	122
58	102
66	90
70	70
80	50
82	45
82	40

x = (100 - Confidence)
y = Distance (cms)

Figure 5.1 Flow Chart

CONCLUSION

Face recognition systems are part of facial image processing applications and their significance as a research area are increasing recently. Implementations of system are crime prevention, video surveillance, person verification, and similar security activities. The face recognition system implementation can be part of Universities. Face Recognition Based Attendance System has been envisioned for the purpose of reducing the errors that occur in the traditional (manual) attendance taking system. The aim is to automate and make a system that is useful to the organization such as an institute. The efficient and accurate method of attendance in the office environment that can replace the old manual methods. This method is secure enough, reliable and available for use. Proposed algorithm is capable of detect multiple faces, and performance of system has acceptable good result.

REFERENCES

[1]. *A brief history of Facial Recognition*, NEC, New Zealand,26 May 2020.[Online]. Available:.

https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facialrecognition/

[2]. *Face detection,TechTarget Network*, Corinne Bernstein, Feb, 2020.[Online]. Available: https://searchenterpriseai.techtarget.com/definition/face-detection

[3]. Paul Viola and Michael Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features*. Accepted Conference on Computer Vision and Pattern Re cognition, 2001.

[4]. *Face Detection with Haar Cascade*,Towards Data Science-727f68dafd08,Girija Shankar Behera, India, Dec 24, 2020.[Online]. Available:https://towardsdatascience.com/face-detectionwith-haar-cascade-727f68dafd08

[5]. *Face Recognition: Understanding LBPH Algorithm,*Towards Data Science90ec258c3d6b,Kelvin Salton do Prado, Nov 11, 2017.[Online]. Available

:https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

[6]. *What is Facial Recognition and how sinister is it*, Theguardian, IanSample, July, 2019. [Online]. Available: https://www.theguardian.com/technology/2019/jul/29/what-is-facialrecognition-and-how-sinister-is-it

[7].Kushsairy Kadir , Mohd Khairi Kamaruddin, Haidawati Nasir, Sairul I Safie, Zulkifli Abdul Kadir Bakti,"A comparative study between LBP and Haar-like features for Face Detection using OpenCV", *4th International Conference on Engineering Technology and Technopreneuship (ICE2T)*, DOI:10.1109/ICE2T.2014.7006273, 12 January 2015.

[8].Senthamizh Selvi.R,D.Sivakumar, Sandhya.J.S , Siva Sowmiya.S, Ramya.S , Kanaga Suba Raja.S,"Face Recognition Using Haar - Cascade Classifier for Criminal Identification", *International Journal of Recent Technology and Engineering(IJRTE)*, vol.7, issn:2277-3878, , issue-6S5, April 2019.

[9]. Robinson-Riegler, G., & Robinson-Riegler, B. (2008). Cognitive psychology: applying the science of the mind. Boston, *Pearson/Allyn and Bacon*..

[10]. Margaret Rouse, *What is facial recognition? - Definition from WhatIs.com,* 2012. [online] Available at: http://whatis.techtarget.com/definition/facial-recognition

[11]. Robert Silk, *Biometrics: Facial recognition tech coming to an airport near you: Travel Weekly,* 2017. [online] Available at: https://www.travelweekly.com/Travel-News/Airline-News/Biometrics-Facial-recognition-tech-coming-airport-near-you

[12]. Sidney Fussell, *NEWS Facebook's New Face Recognition Features: What We Do (and Don't) Know*, 2018. [online] Available at: https://gizmodo.com/facebooks-new-face-recognition-features-what-we-do-an-1823359911

[13]. Reichert, C. *Intel demos 5G facial-recognition payment technology | ZDNet*, 2017. [online] ZDNet. Available at: https://www.zdnet.com/article/intel-demos-5g-facial-recognition-paymenttechnology/#:~:text=Such%20%22pay%20via%20face%20identification,and%20artificial%20intelligence%20(AI). [Accessed 25 Mar. 2018].

[14]. Mayank Kumar Rusia, Dushyant Kumar Singh, Mohd. Aquib Ansari, “Human Face Identification using LBP and Haar-like Features for Real Time Attendance Monitoring”, *2019 Fifth International Conference on Image Information Processing (ICIIP)* ,Shimla, India, DOI: 10.1109/ICIIP47207.2019.8985867 10 February 2020.