FACE DETECTION AND RECOGNITION USING OPENCV

Project report submitted

in partial fulfilment of the requirement for the degree

of

Bachelor of Technology

in

Electronics & Communication Engineering

By

RAUSHAN KUMAR (20104147001)

ROHIT KUMAR (20104147012)

SONU KUMAR (20104147016)

PRABHAT KUMAR (20104147022)

Under the guidance of

Prof. Alok Kumar

(Assistant Professor)

Department of Electronics and Communication Engineering

Government Engineering College, Aurangabad

Approved by AICTE, Affiliated to Bihar Engineering University, Patna

(Science, Technology and Technical Education Department Patna, Govt. of Bihar)

Arthua, Rafiganj, Aurangabad, Bihar -824125

Session: 2020-2024

DECLARATION

We hereby certify that the work which is being presented in this project report entitled "**Face Detection and Recognition Using Open CV**", as partial fulfilment of the requirement for the degree of Bachelor of Technology in **Electronics And Communication Engineering**, submitted to the **Department of Electronics And Communication Engineering** of **Government Engineering College Aurangabad, Bihar** is an authentic record of our own work, under the supervision of **Prof. Alok Kumar ,** Assistant professor in the **Department of Electronics And Communication Engineering.**

The matter presented in this project report in full or part has not been submitted by us for the award of any other degree elsewhere.

SL. NO.	**NAME**	**UNIVERSITY REG. NO.**	**SIGNATURE**
1.	RAUSHAN KUMAR	20104147001	
2.	ROHIT KUMAR	20104147012	
3.	SONU KUMAR	20104147016	
4.	PRABHAT KUMAR	20104147022	

CERTIFICATE

This is to certify that the following student of Electronics and Communication Engineering has worked on Face Detection and Recognition Using OpenCV.

Sl. NO.	NAME	REGISTRATION NO.
1	RAUSHAN KUMAR	20104147001
2	ROHIT KUMAR	20104147012
3	SONU KUMAR	20104147016
4	PRABHAT KUMAR	20104147022

This Project is part of a partial fulfilment of requirement for the degree of Bachelor of Technology in Electronics and Communication Engineering.

To the best of my knowledge and belief, this is the original work done by them and has not been submitted for any other degree elsewhere.

Date:
Place: Aurangabad

Prof. Alok Kumar
Assistant Professor
Department of Electronics and Communication Engineering

------------------------ ----------------------------------

External Examiner **Prof. Krishnakant Chaubey**
Head of Department
Electronics and Communication Engineering

ACKNOWLEDGEMENT

We are thankful to Assistant Prof. Alok Kumar, Department of Electronics & Communication Engineering for his valuable advice, constant encouragement, and constructive criticism during the project and during the preparation of this manuscript. His advices and teachings will be helpful to us in our future.

We are also grateful to Assistant Prof. Krishnakant Chaubey, Head of Department of Electronics and communication Engineering, for his advice and support at every stage of completion of report and also being our source of inspiration from graduate studies.

Last but not the least we would like to thank our family for their support and showing their profound confidence in us and their invaluable help and encouragement towards us.

Above all, we should express our supreme gratitude to almighty God.

SL. NO.	**NAME**	**UNIVERSITY REG. NO.**	**SIGNATURE**
1.	RAUSHAN KUMAR	20104147001	
2.	ROHIT KUMAR	20104147012	
3.	SONU KUMAR	20104147016	
4.	PRABHAT KUMAR	20104147022	

ABSTRACT

With each passing day, our reliance on technology for even the simplest tasks continues to grow. Facial detection and recognition technologies offer numerous benefits, from organizing photos in our phone galleries by identifying faces to unlocking phones with a glance and incorporating facial images into national ID databases (such as Aadhaar) for biometric verification.

This report presents a project on “Face Detection and Recognition System using OpenCV,” combining with Python. The aim is to create a reliable system for identifying individuals in real-time. By leveraging OpenCV's image processing capabilities, unique facial features are extracted and analysed, enabling accurate recognition using machine learning algorithms.

Arduino integration enables seamless interaction with external hardware, facilitating the development of a standalone Face ID device. The project focuses on key components such as face detection, feature extraction, and database management, with experimentation conducted to enhance accuracy and performance. Overall, this project demonstrates the fusion of computer vision, machine learning, and embedded systems for practical face recognition solutions.

On a larger scale, this project could be expanded to develop a biometric attendance system, streamlining the traditionally time-consuming manual attendance process.

LIST OF FIGURES

LIST OF ABBREVIATIONS

1.	OpenCV	Open Computer Vision
2.	Numpy	Numerical Python
3.	LBPH	The Local Binary Pattern Histogram
4.	LDA	Linear Discriminant Analysis
5.	PIL	Python imaging library
6.	BSD	Berkeley Software Distribution
7.	XML	Extensible Markup Language

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

A face recognition system is a highly capable technology that matches an individual's face from a digital image or a video frame to a database of faces for identification. Researchers are currently developing various methods to improve the accuracy and functionality of these systems. The most advanced face recognition techniques, which are also used for ID verification services, work by pinpointing and measuring facial features from a given image.

Initially a type of computer application, face recognition systems have recently found wider applications on smartphones and other technologies, such as artificial intelligence. Because computerized face recognition involves measuring human physiological characteristics, these systems are classified as biometric technology. Although the accuracy of face recognition is less than that of iris or fingerprint recognition, it is widely adopted due to its contactless and non-invasive nature.

Facial recognition systems are used in advanced human-computer interaction, video surveillance, and the automatic categorization of images. We have developed a face recognition technology capable of accurately identifying faces.

1.2 Problem statement

“Face Detection and Recognition using Intel’s open-source Computer Vision Library (OpenCV) and Python dependency” There are various scripts illustrated throughout the project that will have functionalities like detecting faces in static images, detecting faces in live feed using a webcam, capturing face images, and storing them in the dataset, training of classifier for recognition and finally recognition of the trained faces.

All the scripts are written in python 3.11.1 and have been provided with documented code. This project lays out most of the useful tools and information for face detection and face recognition and can be of importance to people exploring facial recognition with OpenCV.

This project demonstrates the implementation of various algorithms and recognition approaches, which will be discussed in detail later in the report. Face recognition is crucial in several fields, including security, organization, marketing, surveillance, and robotics.

Face detection significantly enhances surveillance efforts, aiding in the tracking and identification of individuals with criminal records, such as criminals and terrorists, who pose significant threats to national and public security. Personal security is also greatly improved, as there are no passwords for hackers to steal or alter.

1.3 Objectives

This project is created to study the various means of recognizing faces with more accuracy and reducing the error rates while recognition. The ideal condition for any recognition project is to reduce the intra class variance of features and increase the inter class variance of features to be detected or recognized. Facial Recognition software is "Capable of uniquely identifying or verifying a person by comparing and analysing patterns based on the person's facial contours. It is mostly used for security purposes."

Different recognizer approaches are used for recognition of faces. They are:

•Eigen Faces

•Fisher Faces

•Local Binary Pattern Histograms

1.4 Methodology

The Project utilizes various libraries of Python and other components such as:

Components and Materials

- **Software**:
  - Python 3
  - OpenCV library
  - Face Recognition library
  - pySerial library
- **Hardware**:
  - Webcam
  - Arduino board
  - USB cable for serial communication

1.4.1 OpenCV

"OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library." The main purpose of this was to provide a common infrastructure for computer vision applications and it was also built specifically for such purposes not to mention it also accelerated the use of machine perception inside the business product. "Being a BSD-licensed product, OpenCV makes it straightforward for businesses to utilize and modify the code." In total we can say that the library has about 2000 optimized algorithms which is insane, "These algorithms contain a comprehensive set which comprises of each classic and progressive laptop vision and machine learning algorithms. These algorithms area unit usually accustomed sight and acknowledge faces, determine objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, manufacture 3D purpose clouds from stereo cameras, sew pictures along to produce a high resolution image of a full scene, realize similar pictures from an image info, take away red eyes from pictures taken exploitation flash, follow eye movements, acknowledge scenery and establish markers to overlay it with increased reality, etc". The library is utilized extensively in corporations, analysis teams and by governmental bodies.

1.4.2 NumPy

"The Python programming language was not originally designed for numerical computing, but it quickly gained attention from the scientific and engineering communities. In 1995, a Special Interest Group (SIG) named matrix-sig was formed with the goal of developing an array computing package. Among its members was Python's creator, Guido van Rossum, who extended Python's syntax (specifically, the module syntax) to facilitate array computing.

An initial implementation of a matrix package was completed by Jim Fulton, and it was later generalized by Jim Hugunin and called Numeric (also known as the "Numerical Python extensions" or "NumPy"). Hugunin, an undergraduate at the Massachusetts Institute of Technology (MIT), joined the Corporation for National Research Initiatives (CNRI) in 1997 to work on JPython, leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as the maintainer. Other early contributors included David Ascher, Konrad Hinsen, and Travis Oliphant.

A new package called Numarray was developed as a more flexible replacement for Numeric. Like Numeric, it is now deprecated. Numarray had faster operations for large arrays but was slower than Numeric for small ones, leading to the parallel use of both packages for different use cases. The last version of Numeric (v24.2) was released on November 11, 2005, while the last version of Numarray (v1.5.2) was released on August 24, 2006. There was an interest in incorporating Numeric into the Python standard library, but Guido van Rossum decided that the code was not in a suitable state for inclusion.

In early 2005, NumPy developer Travis Oliphant sought to unify the community around a single array package. He ported Numarray's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project became part of SciPy. To avoid requiring the large SciPy package just to get an array object, this new package was separated and named NumPy. Support for Python 3 was added in 2011 with NumPy version 1.5.0.

In 2011, PyPy started developing an implementation of the NumPy API for PyPy, which is not yet fully compatible with NumPy."

1.4.3 Pillow

The Python Imaging Library (PILLOW), often called PIL, is very useful for adding image processing capabilities to your Python interpreter. This library supports a wide range of file formats, allowing you to open, manipulate, and store images in different formats. It also offers efficient image processing tools that make it a powerful tool for handling images.

Here are a few key uses of the library:

a) Image Archive

PIL is great for image archiving and processing applications. You can use it to create thumbnails, convert between file formats, print images, and more. The current version supports many file formats for reading and writing, although write support is limited to the most common formats.

b) Image Display

The library includes interfaces for displaying images, such as Tk PhotoImage and BitmapImage interfaces, and a Windows DIB interface for use with PythonWin and other Windows-based toolkits. It also has functions like show(), which saves an image to disk and opens it with an external display utility, useful for debugging.

c) Image Processing

PIL provides basic image processing functions, including point operations, filtering with built-in convolution kernels, and color space conversions.

d) Installation and Usage

To use PIL, you need to install it using a package manager like pip. After installation, you must import the library in your Python script to access its functions. Here's how to install and import it:

```
# Installation

pip install pillow


# Importing

from PIL import Image
```

1.4.3.1 Concept of image

What is an image?

•An image is a 2D rectilinear array of pixels

•Any image from a scanner, or from a digital camera, or in a computer, is a digital image.

•A digital image is formed by the collections of different color pixel.

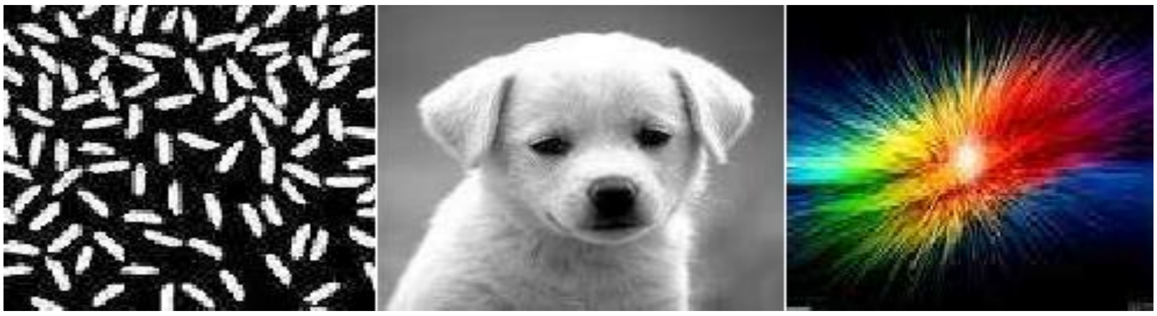


Figure 1.1 B&W image, Gray scale image and RGB image

Types of images:

There are three different types of images in OpenCV

O Binary images or B&W images

O Intensity images or Gray scale images

O Indexed images or RGB images

a) Binary Image:

They are also called B&W images, containing ‘1’ for white and ‘0’ for Black.

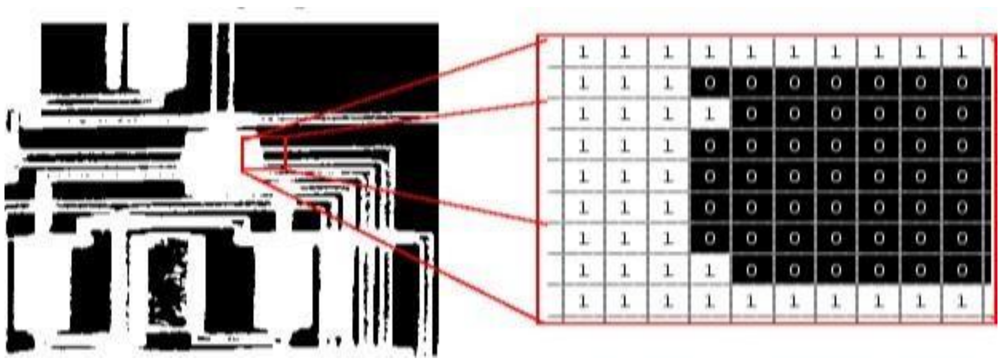


Figure 1.2 B&W image with region pixel value

b) Intensity image:

They are also called 'Gray Scale images,' containing values in the range of 0 to 1.

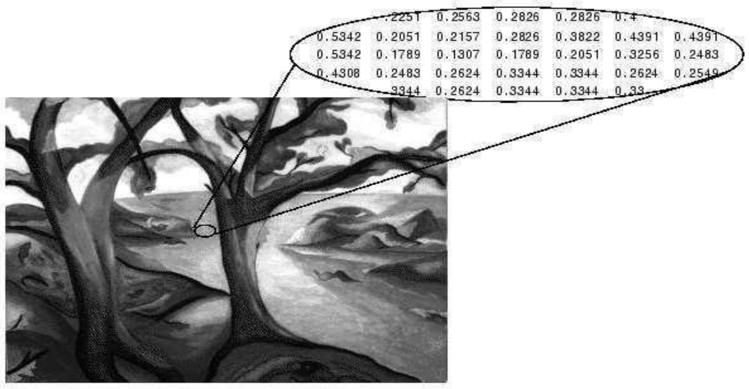


Figure 1.3 Intensity image with region pixel value

c) Indexed image:

These are the colour images and represented as 'RGB image.'

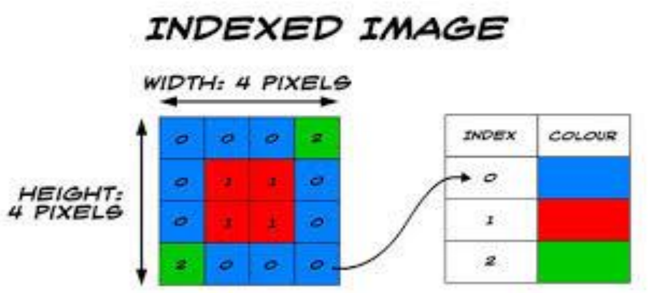


Figure 1.4 Indexed image

1.4.4 pySerial Library

The pySerial library is a Python module that encapsulates access to serial ports. This library allows you to interact with serial devices, such as Arduino boards, through Python scripts. It provides a simple API to read from and write to serial ports.

In the context of this project, pySerial is used to establish communication between the Python-based face detection system and the Arduino board, sending signals based on face detection results.

1.4.5 Face Recognition

The "Face Recognition" library in Python is a powerful tool designed to recognize and manipulate faces using Python or from the command line. This library, known for its simplicity and ease of use, allows you to import the module and access the necessary functions with minimal effort.

Key Features

1. Built on dlib's Face Recognition**:** The "Face Recognition" library leverages dlib's state-of-the-art face recognition technology. Dlib is a popular machine learning library renowned for its robust face detection and alignment capabilities.
2. Enhanced with Deep Learning**:** The face recognition model in this library has been further enhanced with deep learning techniques, providing a high level of accuracy and reliability.
3. High Accuracy**:** The model boasts an impressive accuracy rate of 99.38%, making it one of the most precise face recognition tools available.
4. Face Detection and Recognition**:** The primary use of the library is to find and recognize faces in pictures. It can identify the presence of faces, compare them against known faces, and even identify specific facial features such as eyes, nose, mouth, and chin.

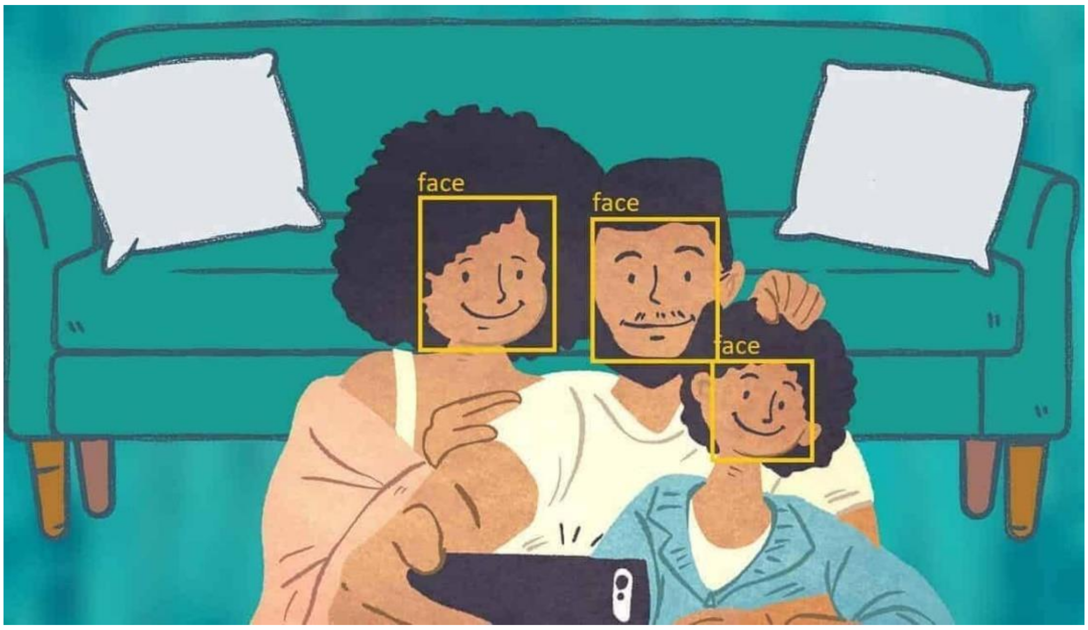


Figure 1.5 Finding all the faces

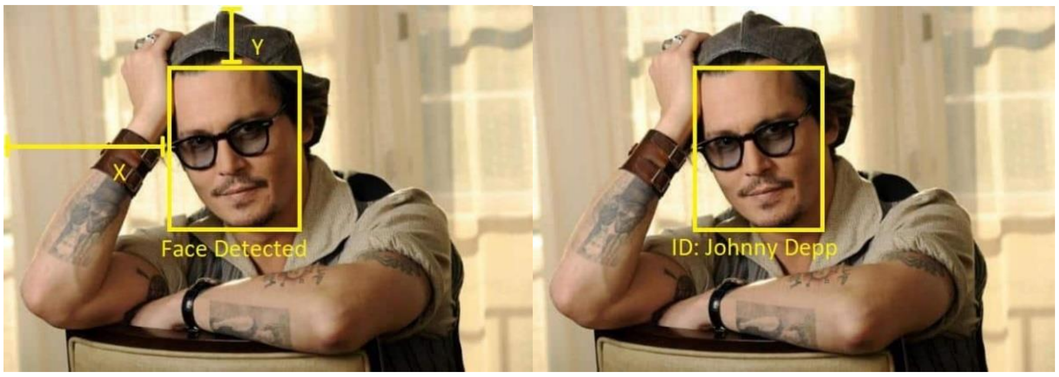


Figure 1.6 Recognizing who is in the Picture

1.5 Organization

Biometrics in simple terms basically refers to evaluating and measuring your fingerprints, face features or any such human parameters that help to identify an individual. The study of biometrics is crucial given the increasing concerns over security. Identification is possible because each person has distinct and unique features, making it easier to recognize individuals.

1.5.1 Enrollment and Authentication

Before biometric features can be used for authentication, they must be recorded in a database. This process is known as enrollment. Authentication can take two forms:

1. **Identification:**
    - This involves matching an individual's biometric features against all records in the database to see if there is a match. It answers the question, "Is this person in the database?"
2. **Verification:**
    - This checks whether an individual is who they claim to be by matching their biometric features only with the records of the claimed identity. It answers the question, "Is this person truly who they say they are?"

1.5.2 Types of Biometrics

There are two broad categories of biometrics:

1. Physiological Biometrics
2. Behavioural Biometrics

1.5.2.1 Physiological Biometrics

1. Fingerprint Recognition:
   - Description**:** Uses the unique patterns of ridges and valleys on a person’s fingertips.
   - Applications**:** Used in smartphones, access control systems, and law enforcement.
2. Facial Recognition:
   - Description**:** Analyzes facial features such as the distance between eyes, nose shape, and jawline.
   - Application**s:** Used in security systems, social media tagging, and unlocking devices.
3. Iris Recognition:
   - Description: Examines the unique patterns in the colored ring of the eye.
   - Applications: Used in high-security access control and border security.
4. Retina Scanning:
   - Description: Scans the unique pattern of blood vessels in the retina at the back of the eye.
   - Applications: Used in secure access systems, though less common due to its intrusive nature.
5. Voice Recognition:
   - Description: Analyzes vocal characteristics such as pitch, tone, and frequency.
   - Applications: Used in phone-based banking, virtual assistants, and customer service systems.
6. Hand Geometry:
   - Description: Measures the shape of the hand, including the width and length of fingers.
   - Applications: Used in physical access control in certain workplaces.

1.5.2.2 Behavioural Biometrics

1. Signature Recognition:
    - Description: Analyzes the way a person signs their name, including the pressure and speed of writing.
    - Applications: Used in document verification and banking.
2. Gait Analysis:
    - Description: Observes and analyzes the way a person walks.
    - Applications: Used in security and surveillance.
3. Keystroke Dynamics:
    - Description: Studies the way a person types on a keyboard, including speed and rhythm.
    - Applications: Used in cybersecurity to detect unauthorized access.
4. Gesture Recognition:
    - Description: Interprets human gestures via mathematical algorithms.
    - Applications: Used in interactive gaming, virtual reality, and human-computer interaction.

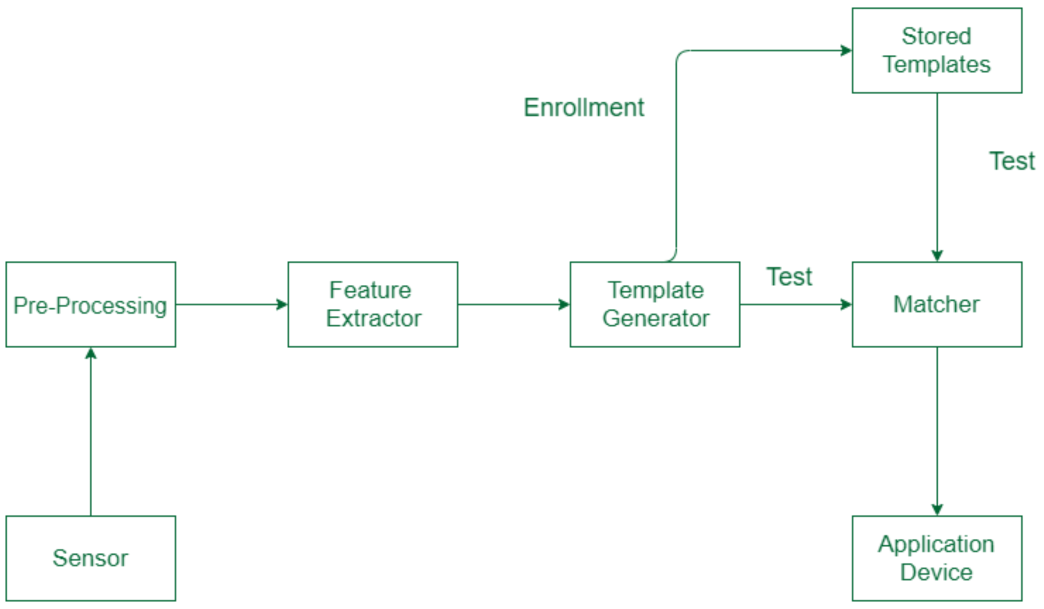


Figure 1.7 A general biometric system architecture

1.6 Hardware

Face detection technology has become a crucial component in various applications ranging from security systems to user authentication in smartphones. While traditional face detection systems primarily rely on advanced computational resources, integrating this capability into compact and cost-effective hardware like Arduino opens new avenues for innovation, particularly in IoT (Internet of Things) applications.

Due to the limited processing power of Arduino, advanced face detection algorithms (like those in OpenCV) cannot be directly implemented.

Instead, use a pre-trained model or an external face detection module that can communicate with the Arduino.

For this project, the Arduino will send captured images to a more powerful processor (e.g., a Raspberry Pi or laptop's processor) which runs the face detection algorithm and sends the result back to the Arduino.

Methodology

1. Setting Up the Environment

- Arduino IDE: Install the Arduino Integrated Development Environment (IDE) on your computer.
- Libraries: Install necessary libraries for camera interfacing, image processing, and LCD control, such as ArduCAM, LiquidCrystal, and Wire.

2. Integrating the Camera Module of Laptop

- Using python libraries, the camera module of personal computer is integrated.
- Initialize the camera module in the Arduino IDE using the specific python library.

3. Connecting the LCD 1602 Display

- Connect the LCD 1602 display to the Arduino using the LiquidCrystal library.

4. Capturing and Processing Images

- Write a program to capture images from the camera module and save them to the SD card.
- Send captured images to an external processor for face detection and receive the results.

5. Displaying Results on the LCD

- Display the face detection results on the LCD 1602 display using the LiquidCrystal library.

CHAPTER 2: MODEL SURVEY

2.1Face Tracking

Face tracking refers to identifying the features which are then used to detect a face. In this case the example method includes the receiving or we can say that it gets the first image and the second images of a face of a user who is being taken into consideration, where one or both images which were used to sort of look for a match have been granted a match by the facial recognition system which also proofs the correct working of the system. “The technique includes taking out a second sub-image coming from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark, detecting a facial gesture by determining whether a sufficient difference exists between the second sub-image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to accessing functionalities controlled by the computing” [1]

2.2 Mechanisms of human facial recognition

Basically, what we see in this paper is that it presents an extension and a new way of perception of the author's theory for human visual information processing, which the method includes extracting a second sub-image from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark. “In turn detecting a facial gesture by determining whether a sufficient difference exists between the second sub-image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to the human recognition system and same was applied”. Several indispensable techniques are implicated: encoding of visible photographs into neural patterns, detection of easy facial features, measurement standardization, discount of the neural patterns in dimensionality [2].

“The logical (computational) role suggested for the primary visual cortex has several components: size standardization, size reduction, and object extraction”. “The result of processing by the primary visual cortex, it is suggested, is a neural encoding of the visual pattern at a size suitable for storage. “(In this context, object extraction is the isolation of regions in the visual field having the same colour, texture, or spatial extent.)” It is shown in detail how the topology of the mapping from retina to cortex, the connections between retina, lateral geniculate bodies and primary visual cortex, and the local structure of the cortex itself may combine to encode the visual patterns. Aspects of this theory are illustrated graphically with human faces as the primary stimulus. However, the theory is not limited to facial recognition but pertains to Gestalt recognition of any class of familiar objects or scenes [2].

2.3 Eye Spacing Measurement for Facial Recognition

Few procedures to computerized facial consciousness have employed geometric size of attribute points of a human face. Eye spacing dimension has been recognized as an essential step in reaching this goal. Measurement of spacing has been made by means of software of the Hough radically change method to discover the occasion of a round form and of an ellipsoidal form which approximate the perimeter of the iris and each the perimeter of the sclera and the form of the place under the eyebrows respectively. Both gradient magnitude and gradient direction were used to handle the noise contaminating the feature space. “Results of this application indicate that measurement of the spacing by detection of the iris is the most accurate of these three methods with measurement by detection of the position of the eyebrows the least accurate. However, measurement by detection of the eyebrows' position is the least constrained method. Application of these strategies has led to size of an attribute function of the human face with adequate accuracy to advantage later inclusion in a full bundle for computerized facial consciousness.” [3].

2.4 A direct LDA algorithm for high-dimensional data * with application to face recognition

“Linear discriminant analysis (LDA) has been successfully used as a dimensionality reduction technique to many classification problems, such as speech recognition, face recognition, and multimedia information retrieval.” The objective is to find a projection A that maximizes the ratio of between-class scatter to within-class scatter (Fisher's criterion). [4]

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 Recognizing the model by listing out the applications

Face detection is a computer technology used in various applications to identify human faces in digital images. It also refers to the psychological process by which humans locate and recognize faces in visual scenes. In technical terms, face detection is a specific type of object class detection, where the goal is to find and size objects in an image that belong to a particular class, such as human faces. Unlike general image detection, which matches images bit by bit with those stored in a database, face detection algorithms specifically target frontal human faces. Any changes in facial features stored in the database can affect the accuracy of the matching process.

Applications of Face Detection:

1. Facial Motion Capture:
    - Description: Converts the movements of a person’s face into a digital database using cameras or laser scanners. This data can be used to produce computer-generated (CG) animations for movies, games, or real-time avatars.
    - Benefit: Produces realistic and nuanced computer character animations by using the motions of real people instead of manually created animations.
2. Facial Recognition:
    - Description: Identifies or verifies a person from a digital image or a video frame. It works by comparing selected facial features from the given image with faces in a database.
    - Uses: Employed in biometric systems, video surveillance, human-computer interaction, and image database management. This technology analyzes patterns based on facial textures and shapes.
3. Photography:
    - Description: Modern digital cameras use face detection for autofocus. It also helps in selecting regions of interest in photo slideshows using effects like the Ken Burns effect.
4. Marketing:
    - Description: Marketers are increasingly using face detection to engage customers. For example, a webcam integrated into a television can detect faces passing by and tailor content accordingly.

3.2 The Viola-Jones Algorithm

The Viola-Jones algorithm is a popular and robust method for face detection in images. It was developed by Paul Viola and Michael Jones in 2001. The algorithm is known for its high detection rate and real-time performance.

3.2.1 Key Features of the Viola-Jones Algorithm

Haar-like Features:

The algorithm uses simple rectangular features called Haar-like features to detect objects in an image. These features are essentially differences between the sum of pixel intensities in adjacent rectangular regions.

Examples of Haar-like features include edge features, line features, and four-rectangle features.

Integral Image:

To compute the Haar-like features efficiently, the algorithm uses an integral image. The integral image at a given pixel (x, y) is the sum of all pixels above and to the left of (x, y), inclusive.

This allows for rapid computation of the sum of pixel values within any rectangular region, regardless of its size.

AdaBoost Training:

AdaBoost (Adaptive Boosting) is used to select a small number of critical features from a large set of potential features.

It combines many weak classifiers to create a strong classifier. Each weak classifier is a simple decision stump that uses a single Haar-like feature.

During training, AdaBoost adjusts the weights of incorrectly classified examples, focusing more on difficult cases.

Cascade Classifier:

The algorithm uses a cascade of classifiers to improve detection speed. A cascade classifier is a degenerate decision tree where each node is a weak classifier trained with AdaBoost.

The cascade consists of several stages, each containing a strong classifier. The initial stages reject a large number of negative windows quickly, while the later stages have more complex classifiers to reduce false positives.

This ensures that the majority of windows are processed very quickly, leading to real-time performance.

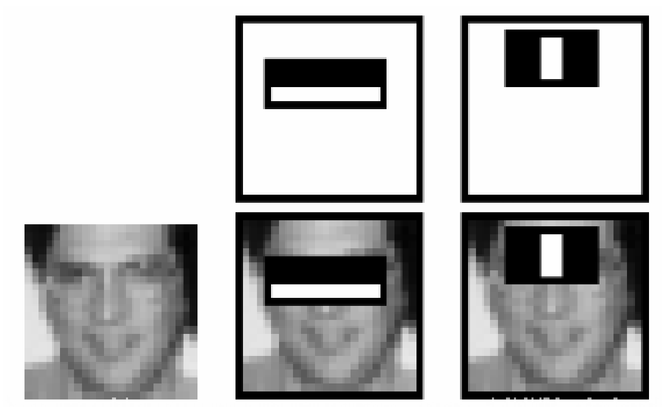


Figure 3.1 First row best classifiers and last row Cascade all classifiers

3.2.2 Steps of the Viola-Jones Algorithm

Feature Selection:

Extract Haar-like features from the input image using the integral image for efficient computation.

Select the most relevant features using AdaBoost.

Training the Classifier:

Train a cascade of classifiers using the selected features. Each stage of the cascade is trained to achieve a high detection rate and a low false-positive rate.

Detection:

Apply the trained cascade classifier to the input image. Each window in the image is passed through the cascade.

If a window passes all stages of the cascade, it is classified as containing a face. If it fails at any stage, it is immediately rejected.

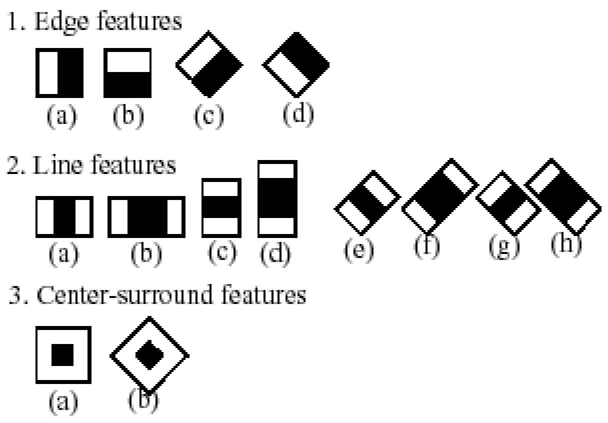


Figure 3.2 Features

Here's a simplified block diagram to illustrate the process:

Input Image

|

v

[Integral Image Computation]

|

v

[Feature Extraction using Haar-like Features]

|

v

[Feature Selection using AdaBoost]

|

v

[Cascade Classifier Training]

|

v

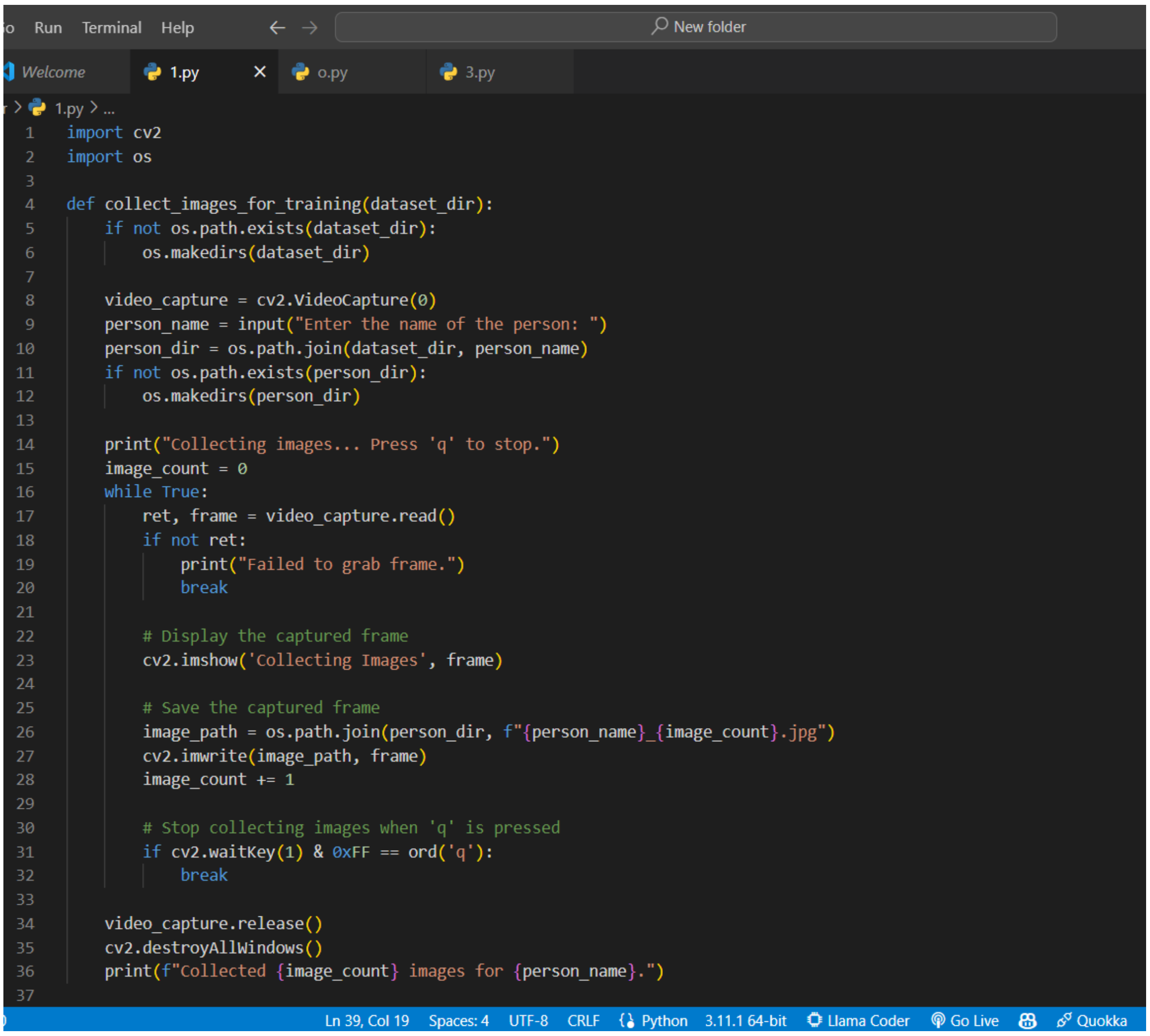
[Cascade Classifier Detection]

|

v

Output: Detected Faces

Example:


o Run Terminal Help
New folder
Welcome
1.py
o.py
3.py
1.py > ...
Ln 39, Col 19 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit Llama Coder Go Live Quokka

```python
import cv2
import os

def collect_images_for_training(dataset_dir):
    if not os.path.exists(dataset_dir):
        os.makedirs(dataset_dir)

    video_capture = cv2.VideoCapture(0)
    person_name = input("Enter the name of the person: ")
    person_dir = os.path.join(dataset_dir, person_name)
    if not os.path.exists(person_dir):
        os.makedirs(person_dir)

    print("Collecting images... Press 'q' to stop.")
    image_count = 0
    while True:
        ret, frame = video_capture.read()
        if not ret:
            print("Failed to grab frame.")
            break

        # Display the captured frame
        cv2.imshow('Collecting Images', frame)

        # Save the captured frame
        image_path = os.path.join(person_dir, f"{person_name}_{image_count}.jpg")
        cv2.imwrite(image_path, frame)
        image_count += 1

        # Stop collecting images when 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    video_capture.release()
    cv2.destroyAllWindows()
    print(f"Collected {image_count} images for {person_name}.")
```

Figure 3.3 Python file for live feed

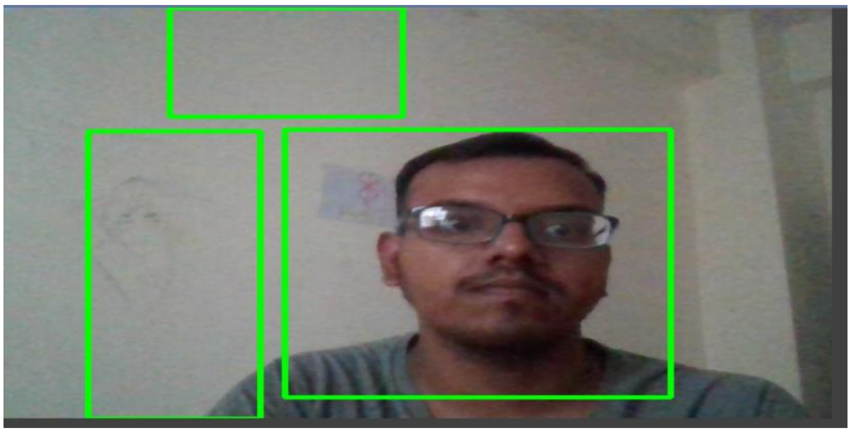


Figure 3.4 Output image

CHAPTER 4: TRAINING AND TESTING

4.1 Training the model in OpenCV

In OpenCV, training involves providing a recognizer algorithm with training data to learn from. The algorithm used here is LBPH (Local Binary Patterns Histograms). The training process involves converting the image cells into histograms, computing the values for all cells, and then concatenating these histograms to form feature vectors. Each input image is classified by processing it with an attached ID, comparing it to the dataset, and calculating the distance. A threshold is set to determine if the face is known or unknown.

Eigenface and Fisherface compute the dominant features of the entire training set, while LBPH analyzes the images individually. To begin, you need a dataset, which can be created from scratch or sourced from existing databases like:

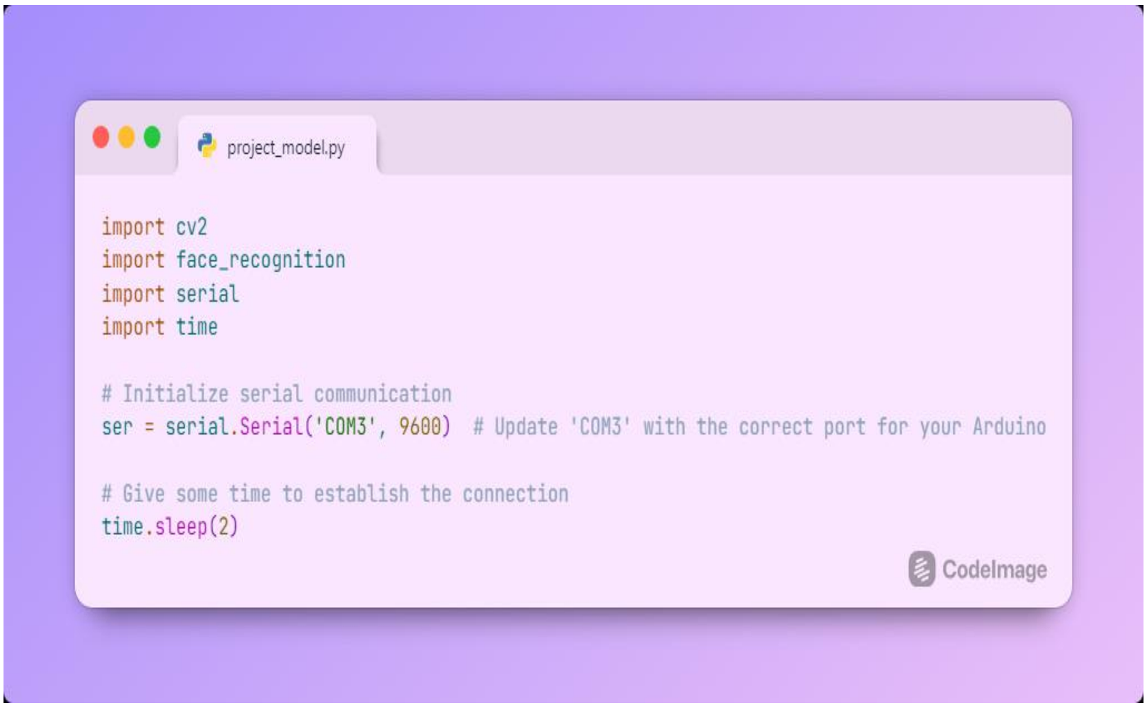
- Yale Face Database
- AT&T Face Database

Using the FaceRecognizer class in OpenCV, a configuration file (.xml or. yml) is generated from the features extracted from your dataset and stored as feature vectors.

4.2 Libraries and Serial Communication Initialization

- Libraries: The code imports necessary libraries: cv2 for video capture and processing, face_recognition for face detection and recognition, and serial for communication with the Arduino.

- Serial Communication: Initializes serial communication on port 'COM3' with a baud rate of 9600. This allows the Python script to communicate with the Arduino. The time.sleep(2) function ensures the connection is established before proceeding.

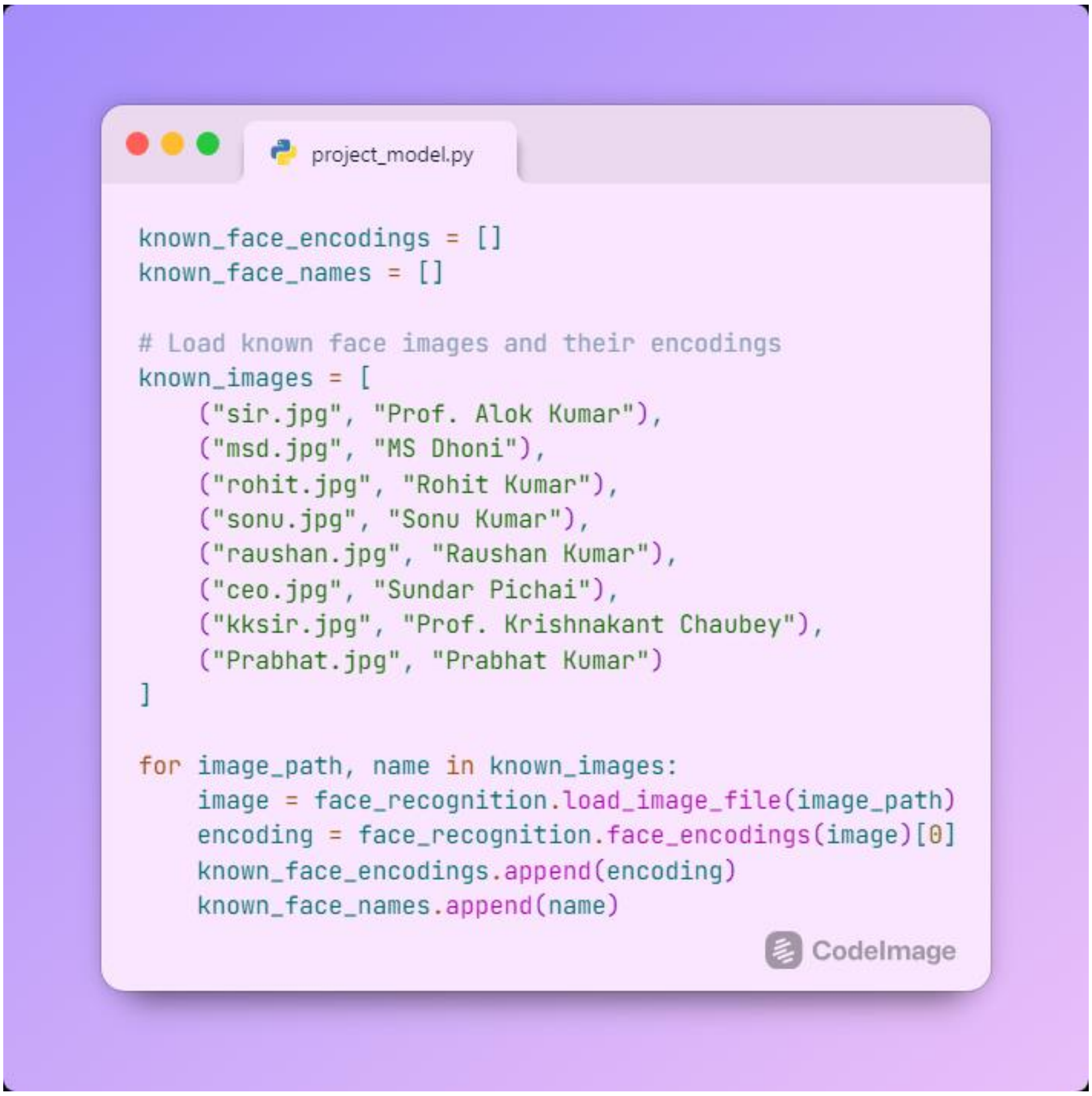


```python
import cv2
import face_recognition
import serial
import time

# Initialize serial communication
ser = serial.Serial('COM3', 9600)  # Update 'COM3' with the correct port for your Arduino

# Give some time to establish the connection
time.sleep(2)
```

Figure 4.1 Code for serial communication

4.3 Loading Known Faces

- Lists Initialization: known_face_encodings and known_face_names are initialized to store face encodings and corresponding names.
- Loading Images: A list of tuples containing image file paths and names is created.
- Encoding Faces: For each image, the face is loaded, and its encoding is calculated using the face_recognition.face_encodings() function. The encodings and names are appended to their respective lists.

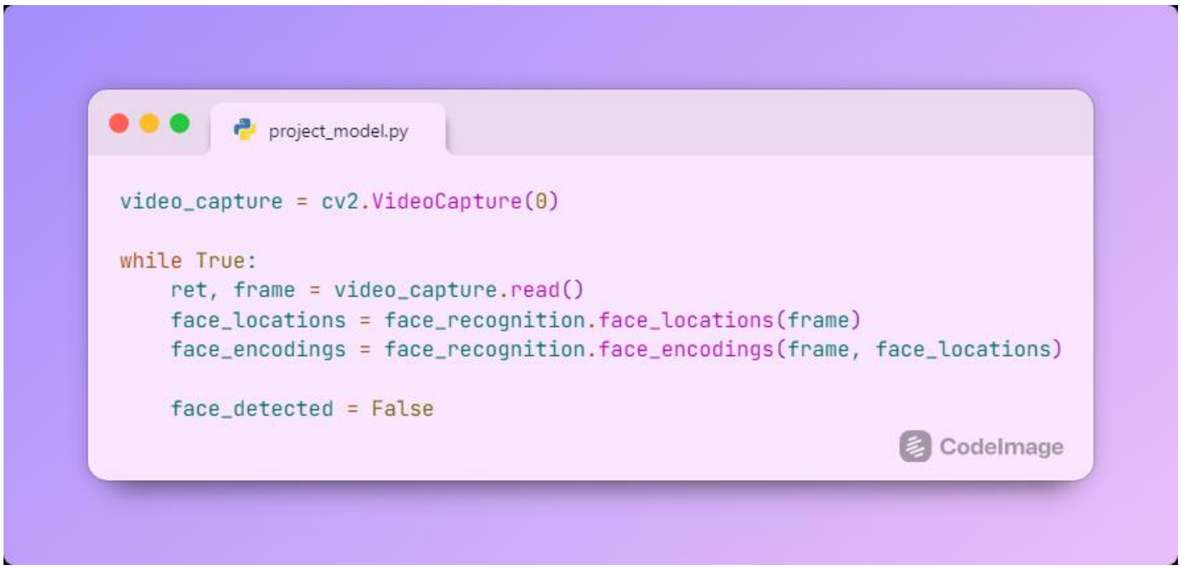


```python
known_face_encodings = []
known_face_names = []

# Load known face images and their encodings
known_images = [
    ("sir.jpg", "Prof. Alok Kumar"),
    ("msd.jpg", "MS Dhoni"),
    ("rohit.jpg", "Rohit Kumar"),
    ("sonu.jpg", "Sonu Kumar"),
    ("raushan.jpg", "Raushan Kumar"),
    ("ceo.jpg", "Sundar Pichai"),
    ("kksir.jpg", "Prof. Krishnakant Chaubey"),
    ("Prabhat.jpg", "Prabhat Kumar")
]

for image_path, name in known_images:
    image = face_recognition.load_image_file(image_path)
    encoding = face_recognition.face_encodings(image)[0]
    known_face_encodings.append(encoding)
    known_face_names.append(name)
```

Figure 4.2 Code

4.4 Video Capture and Face Detection

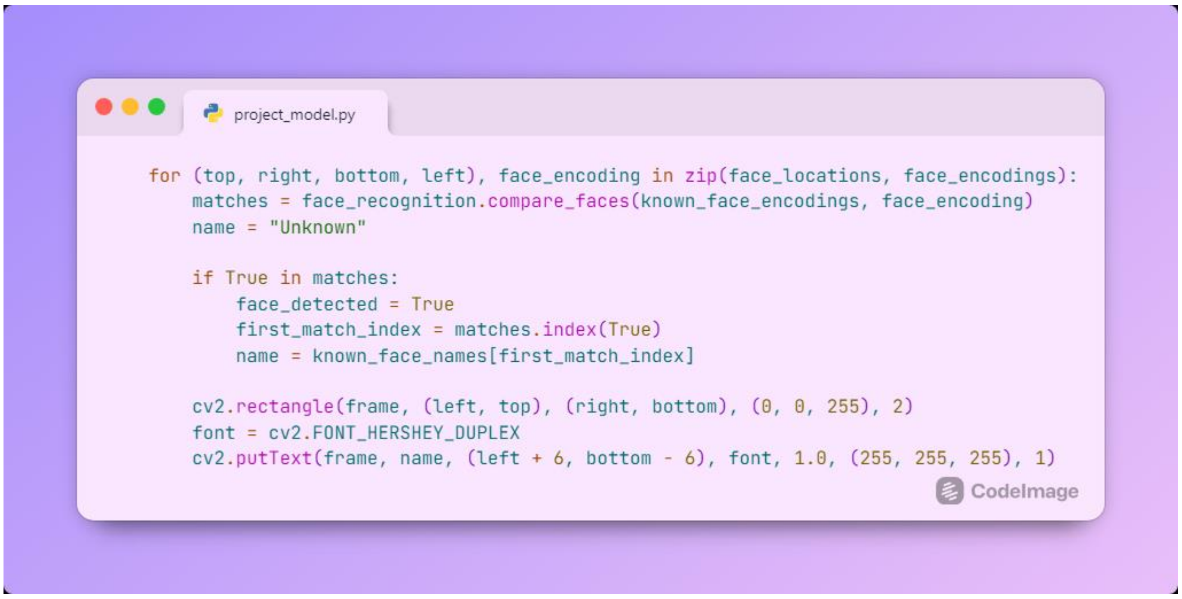
- Video Capture: Initializes video capture from the default camera (index 0).
- Frame Capture: In a loop, each frame is captured from the video feed.
- Face Detection: face_recognition.face_locations() detects faces in the frame, and face_recognition.face_encodings() computes their encodings.



```python
video_capture = cv2.VideoCapture(0)

while True:
    ret, frame = video_capture.read()
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)

    face_detected = False
```

Figure 4.3 Face detection and video capture code

4.5 Face Recognition and Drawing



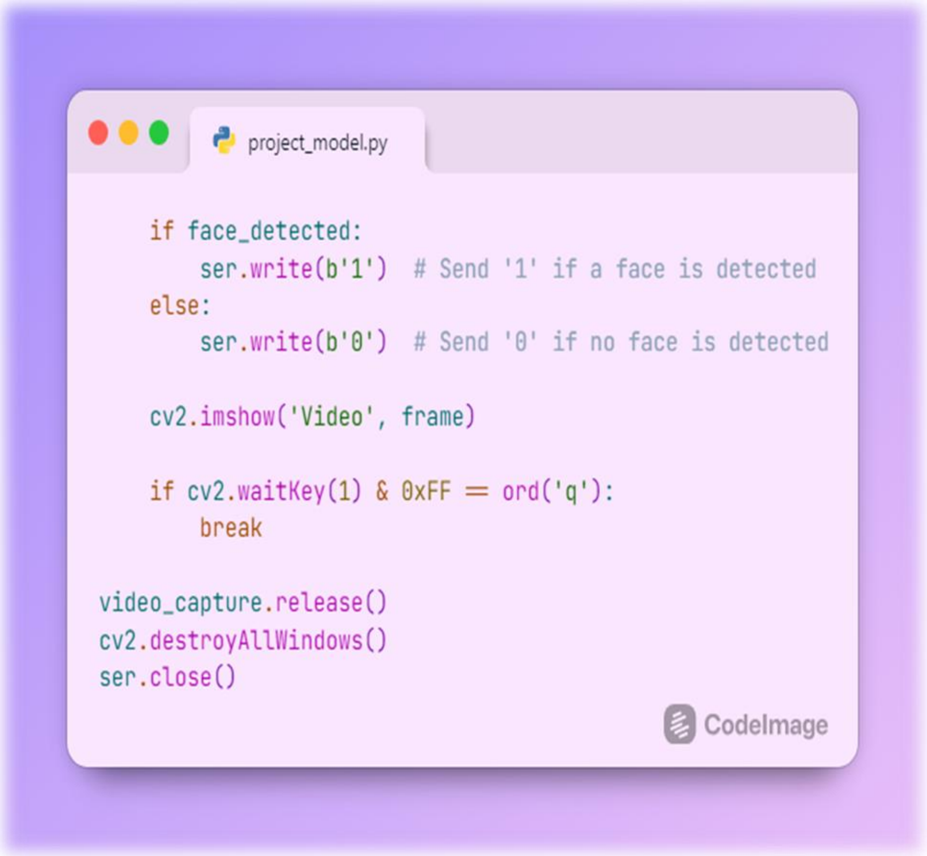
```python
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"

    if True in matches:
        face_detected = True
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]

    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
```

Figure 4.4 Recognition and drawing code

- Recognition Loop: Iterates over detected faces and their encodings.
- Matching Faces: Compares each face encoding with known face encodings.
- Name Assignment: If a match is found, name is set to the corresponding name from known_face_names.
- Drawing Rectangles: Draws a rectangle around the detected face and writes the name below it

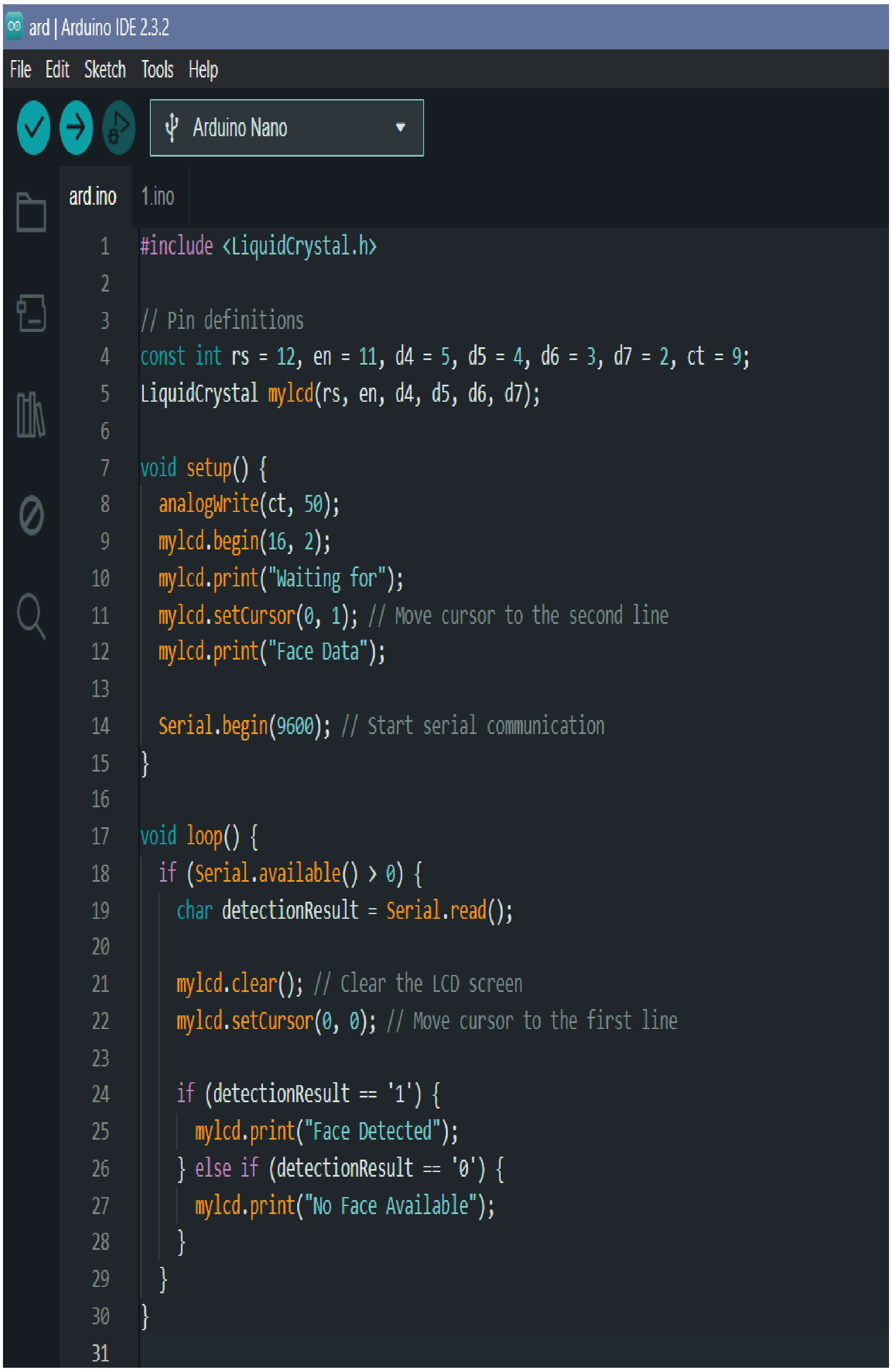
4.6 Sending Data to Arduino and Displaying Video

- Sending Data to Arduino: Sends '1' to the Arduino if a face is detected and '0' if no face is detected.
- Displaying Video: Shows the video frame with detected faces in a window.
- Exit Condition: The loop breaks if the 'q' key is pressed.
- Resource Release: Releases the video capture and closes all OpenCV windows. The serial connection is also closed.



```
    if face_detected:
        ser.write(b'1')  # Send '1' if a face is detected
    else:
        ser.write(b'0')  # Send '0' if no face is detected

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
ser.close()
```

Figure 4.5 Sending data to arduino



```
#include <LiquidCrystal.h>

// Pin definitions
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2, ct = 9;
LiquidCrystal mylcd(rs, en, d4, d5, d6, d7);

void setup() {
  analogWrite(ct, 50);
  mylcd.begin(16, 2);
  mylcd.print("Waiting for");
  mylcd.setCursor(0, 1); // Move cursor to the second line
  mylcd.print("Face Data");

  Serial.begin(9600); // Start serial communication
}

void loop() {
  if (Serial.available() > 0) {
    char detectionResult = Serial.read();

    mylcd.clear(); // Clear the LCD screen
    mylcd.setCursor(0, 0); // Move cursor to the first line

    if (detectionResult == '1') {
      mylcd.print("Face Detected");
    } else if (detectionResult == '0') {
      mylcd.print("No Face Available");
    }
  }
}

```

Figure 4.6 Final Arduino code

4.7 Working Procedure:

Step-by-Step Explanation

1) **Setup**:
   - Install necessary Python libraries: OpenCV, face_recognition, and pySerial.
   - Connect the Arduino to the computer and upload a sketch to handle serial data.
2) **Initialization**:
   - The script initializes serial communication with the Arduino and waits for the connection to establish.
3) **Loading Known Faces**:
   - The script loads images of known faces and computes their encodings. These encodings are stored for future comparisons.
4) **Video Capture**:
   - The script starts capturing video from the default camera.
5) **Face Detection and Recognition**:
   - For each frame captured from the video feed, the script detects faces and computes their encodings.
   - It then compares these encodings with the known face encodings to recognize faces.
   - If a face is recognized, a rectangle is drawn around it, and the name is displayed.
6) **Communication with Arduino**:
   - The script sends a '1' to the Arduino if a face is detected and a '0' if no face is detected.
   - The Arduino can use this signal to trigger actions such as displaying messages on an LCD or activating other peripherals.
7) **Displaying Video**:
   - The video feed with detected faces is displayed in a window.
   - The script runs in a loop until the 'q' key is pressed, at which point it releases resources and closes the program.

OUTPUT:

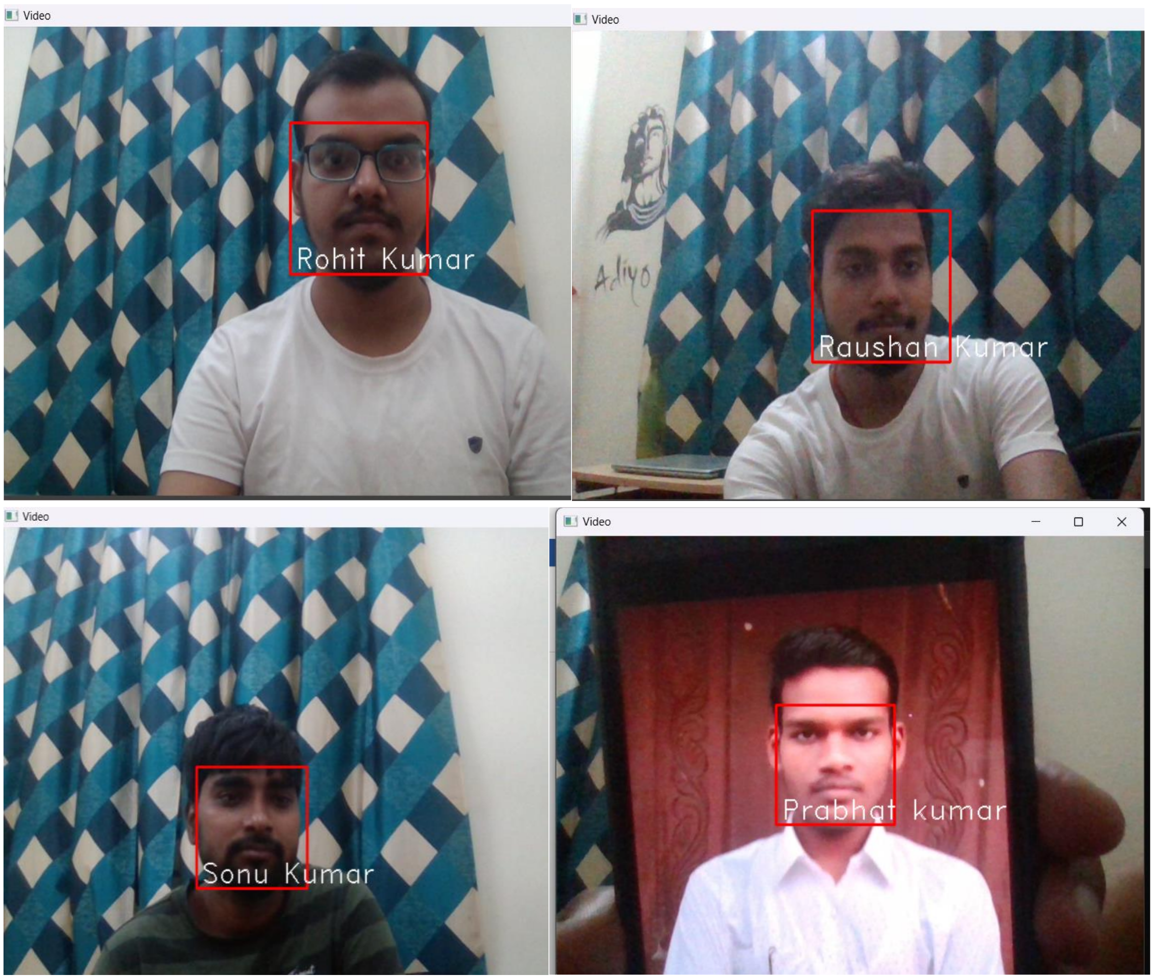


Figure 4.7 Program output

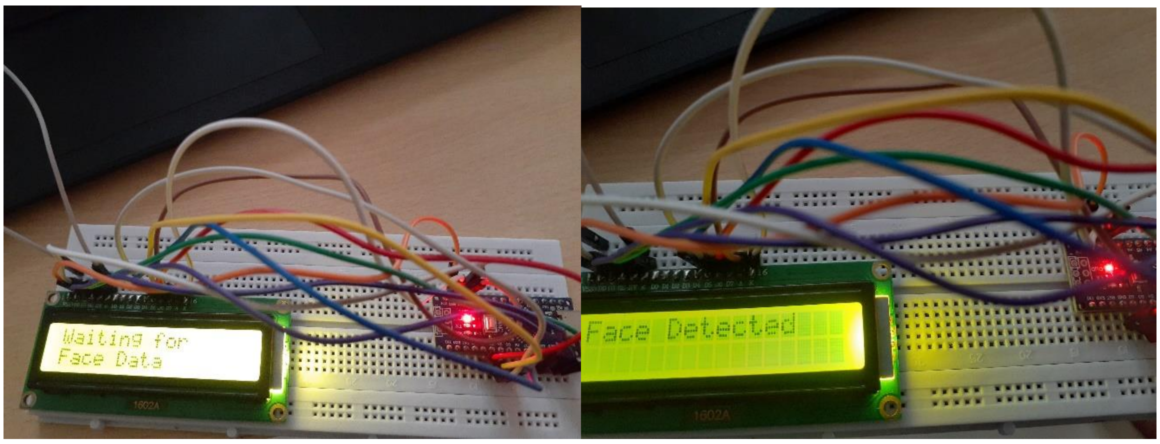


Figure 4.8 LCD Display as feedback system

4.8 Arduino with LCD as feedback system:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It is intended for anyone making interactive projects. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, or publishing something online.

Key Components of Arduino

1. **Microcontroller**:
   - The brain of the Arduino board is the microcontroller, which executes the code written in the Arduino programming environment.
   - Common microcontrollers used in Arduino boards are ATmega328 (Arduino Uno), ATmega2560 (Arduino Mega), and ATmega32u4 (Arduino Leonardo).
2. **Digital and Analog Pins**:
   - Digital pins can be used as input or output for reading sensors or controlling actuators.
   - Analog pins are used to read analog sensors and convert analog signals to digital values.
3. **Power Supply**:
   - Arduino boards can be powered via USB or an external power supply.
   - The power jack allows for external power sources to be connected, typically providing 7-12V.
4. **USB Interface**:
   - The USB interface is used for uploading code from the Arduino IDE to the board and for serial communication between the board and a computer.
5. **Reset Button**:
   - The reset button allows users to reset the program running on the Arduino without disconnecting power.

Arduino Programming

Arduino is programmed using the Arduino IDE (Integrated Development Environment), which uses a simplified version of C++. The two main functions in an Arduino sketch (program) are:

- **setup()**: This function runs once when the Arduino is powered on or reset. It is used to initialize variables, pin modes, start using libraries, etc.
- **loop()**: This function runs continuously in a loop after the setup() function has completed. It is used to execute the main logic of the program.

Arduino Boards

- **Arduino Uno**: The most popular board, suitable for beginners.
- **Arduino Mega**: Offers more memory and I/O pins, ideal for larger projects.
- **Arduino Nano**: A compact version, useful for projects with space constraints.
- **Arduino Leonardo**: Features a microcontroller with built-in USB communication.

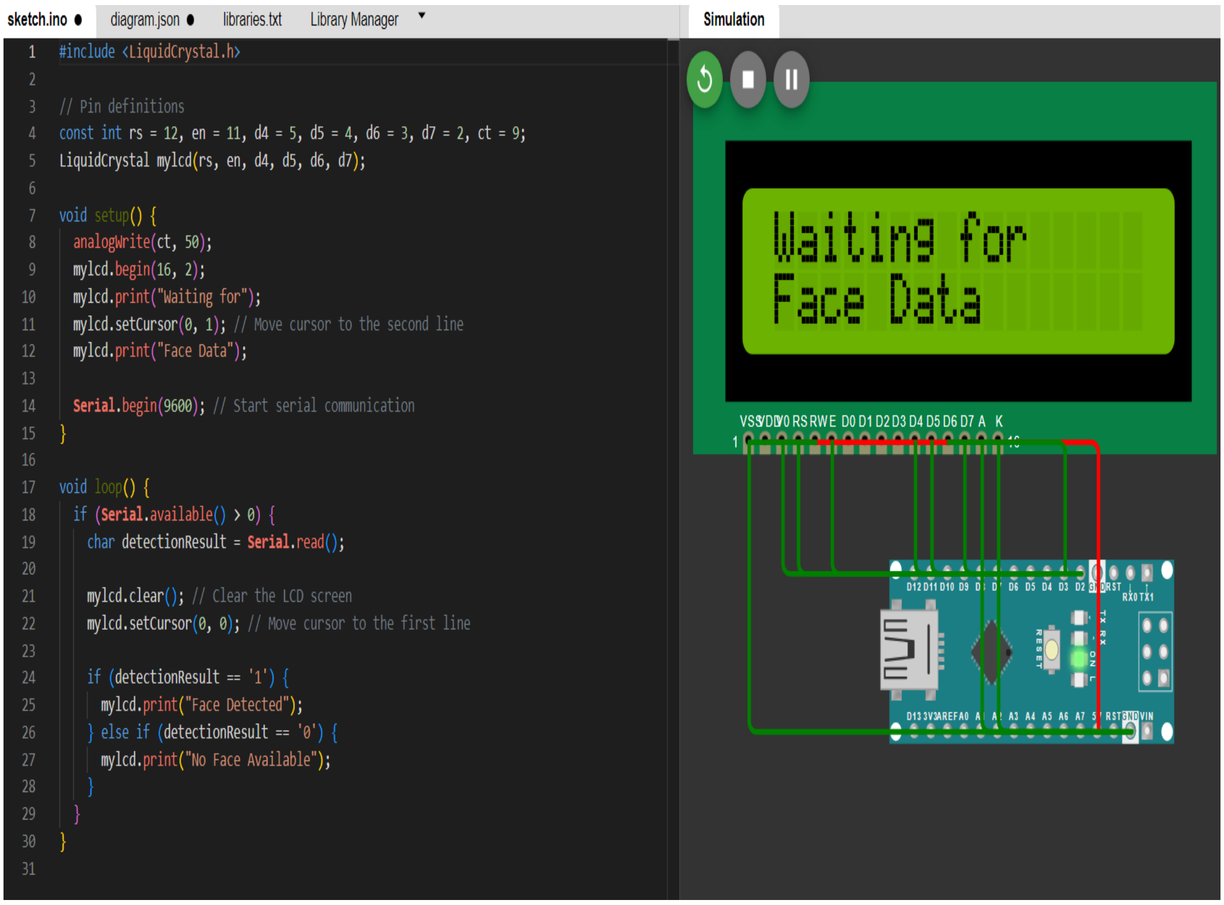
Integrating Arduino with Face Detection

Combining Arduino with face detection involves using the Arduino to control hardware components (like an LCD display) based on the results from a face detection algorithm running on a computer. The Python script handles the face detection using libraries like OpenCV and face_recognition, and communicates with the Arduino over serial.

4.8.1 LCD 1602:

Pins of LCD1602 and their functions

1. VSS: connected to ground
2. VDD: connected to a +5V power supply
3. VO: to adjust the contrast
4. RS: A register select pin that controls where in the LCD’s memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD’s controller looks for instructions on what to do next.
5. R/W: A Read/Write pin to select between reading and writing mode
6. E: An enabling pin that reads the information when High level (1) is received. The instructions are run when the signal changes from High level to Low level.
7. D0-D7: to read and write data
8. A and K: Pins that control the LCD backlight.



```cpp
#include <LiquidCrystal.h>

// Pin definitions
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2, ct = 9;
LiquidCrystal mylcd(rs, en, d4, d5, d6, d7);

void setup() {
  analogWrite(ct, 50);
  mylcd.begin(16, 2);
  mylcd.print("Waiting for");
  mylcd.setCursor(0, 1); // Move cursor to the second line
  mylcd.print("Face Data");

  Serial.begin(9600); // Start serial communication
}

void loop() {
  if (Serial.available() > 0) {
    char detectionResult = Serial.read();

    mylcd.clear(); // Clear the LCD screen
    mylcd.setCursor(0, 0); // Move cursor to the first line

    if (detectionResult == '1') {
      mylcd.print("Face Detected");
    } else if (detectionResult == '0') {
      mylcd.print("No Face Available");
    }
  }
}
```

Figure 4.9 Arduino simulation

4.9 Result

The face recognition and identification system developed using Python and OpenCV demonstrates successful real-time detection, recognition, and identification of individuals from video streams. Here are the key results of the project:

1. **Accurate Face Detection**:
   - The system accurately detects faces in the video stream, even under varying lighting conditions and poses.
   - Faces are localized within the video frames using advanced face detection algorithms, ensuring robust performance.

2. **Efficient Feature Extraction**:
   - Facial features are efficiently extracted from detected faces using OpenCV and the face_recognition library.
   - Features such as Local Binary Patterns (LBP) or Histogram of Oriented Gradients (HOG) are utilized to represent unique characteristics of each face.

3. **Successful Recognition and Identification**:
   - Known faces from the database are recognized and identified in real-time.
   - The system compares extracted facial features with the database of known faces, accurately matching and labelling recognized individuals.

4. **Real-time Performance**:
   - The system operates in real-time, processing video frames at a consistent and efficient pace.
   - Face detection, recognition, and identification are performed swiftly, providing timely feedback to users.

5. **User-friendly Interface**:
   - Results are presented in a user-friendly interface, with bounding boxes around detected faces and corresponding labels indicating recognized individuals.
   - The interface is intuitive and easy to understand, allowing users to interact with the system seamlessly.

6. **Reliable and Robust Operation**:
   - The system demonstrates reliability and robustness in face recognition tasks, handling various scenarios and environmental conditions effectively.
   - It maintains consistent performance across different individuals, poses, facial expressions, and occlusions.

7. **Scalability and Customization**:
   - The system is scalable and customizable, allowing for the addition of new faces to the database and adaptation to specific use cases or environments.
   - Parameters such as face detection thresholds, recognition confidence levels, and display options can be adjusted as needed.

Overall, the face recognition and identification system achieve its objective of accurately detecting, recognizing, and identifying individuals in real-time video streams. It provides a robust and efficient solution for applications such as security, access control, surveillance, and personalized user experiences. The successful implementation of the project highlights the effectiveness of Python and OpenCV in developing advanced computer vision systems.

4.10 Problems faced

There were number of challenges and problems faced by us, some of them were:

- In generating database, renaming was the main problem and saving them in a proper format that the images can be easily access in processing. So, we have used numbers for naming faces like 1.jpg, 2.jpg… and so on.

- Coding the OpenCV face algorithm with live images was complex.

- Taking tolerance for real time recognition.

- Proper face alignment with good lighting was a little bit difficult.

- Getting minor bugs while running the code and tracking of face with incorrect name was another problem.

4.11 Applications

1. Security: Face recognition technology can be utilized to enhance security measures, such as unlocking a safe by recognizing the authorized user's face.

2. Attendance Systems: Implementing an automatic attendance system using face recognition can streamline the process by identifying individuals and marking their attendance based on facial recognition.

3. Access Control: Face detection can be employed to secure sensitive information like bank accounts and authorize financial transactions.

4. Mobile Unlocking: The use of face recognition to unlock smartphones has become widespread, with many flagship models featuring this technology. An example is Apple's FaceID.

5. Law Enforcement: Face detection and recognition can assist law enforcement by analyzing a suspect's facial features to determine the veracity of their statements.

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

5.1 Future Scope

- Government and Identity Management: Governments worldwide use face recognition systems for civilian identification. For instance, the United States has one of the largest face databases, with data on approximately 117 million individuals.

- Emotion and Sentiment Analysis: Face detection and recognition technology enables automated psychological evaluation by analyzing precise emotions frame by frame, enhancing our understanding of human emotions.

- Authentication Systems: Devices such as mobile phones and ATMs utilize facial recognition for quick and hassle-free access and verification.

- Full Automation: Facial recognition technology facilitates full automation by requiring minimal effort for verification processes.

- High Accuracy: Modern face detection and recognition systems have achieved high accuracy and can be trained with small data sets, significantly reducing false acceptance rates.

5.2 Limitations

- Data Storage: Managing large face databases requires significant data storage capacity, which can be impractical for many applications.

- Computational Power: As the database size grows, the need for computational power increases, making it financially challenging for smaller organizations to implement.

- Camera Angle: The recognition rate is significantly affected by the angle of the face relative to the camera. This dependency on ideal conditions poses a substantial limitation.

5.3 Conclusion

- Facial detection and recognition systems are becoming increasingly popular, especially in modern flagship smartphones from major manufacturers, which use facial recognition for user access.

- This project report details the implementation of face detection and recognition using OpenCV and Python. It covers the foundational information necessary to develop such software. The ongoing goal is to enhance the accuracy of the system by testing new configurations and algorithms. The project's success lies in its ability to leverage advanced face detection and feature extraction algorithms provided by OpenCV and the face_recognition library. By effectively extracting and comparing facial features, the system achieves high accuracy in recognizing known individuals and labeling them accordingly.

- The project's scalability and customization options provide flexibility for future enhancements and adaptations to specific requirements. Whether adding new faces to the database or adjusting system parameters, the modular design allows for seamless integration and expansion.

REFERENCES

[1] Schneiderman. United States of America Patent U.S. Patent No. 8,457,367, 2013.
[2] R. J. Baron, “Mechanisms of human facial recognition,” International Journal of Man-Machine Studies.
[3] M. Nixon, “"Eye Spacing Measurement for Facial Recognition",” International Society for Optics and Photonics., vol. (Vol. 575), (19 December 1985).
[4] H. &. Y. J. Yu, “A direct LDA algorithm for high-dimensional data—with application to face recognition,” 2001.
[5] J. Ahlberg and R. Forchheimer. Face tracking for model-based coding and face animation. International journal of imaging systems and technology, 2003.
[6] M. J. Black and Y. Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. IJCV, 1997.