# NAME : NIRMA ABRO

**TEACHER:DR.YASIR ARFAT MALKANI (PYTHON PROJECT REPORT)**

# TOPIC: FACE RECOGNITION SYSTEM USING PYTHON

**DEPARTMENT OF COMPUTER SCIENCE**

# UNIVERSITY OF SINDH JAMSHORO

**( Face Recognition Project Report )**

# Table of Contents

1. **Project Overall Summary (Abstract)**
2. **Introduction**
3. **Material and Methods**
   * **Hardware Components**
   * **Software Components and Libraries 4)Methodology**
   * **Loading Known Faces and Names**
   * **Video Capture and Processing**
   * **Face Detection and Recognition**
   * **Display and Labeling**
   * **Key Functions and Methods**
4. **Complete Source Code with output**
5. **Flowchart**
6. **Conclusion**
7. **References / Bibliography**

# Project Overall Summary (Abstract)

This report presents the development and implementation of a real-time face recognition system utilizing computer vision techniques. The system is designed to detect and identify known individuals in live video streams, offering applications in security, surveillance, and user experience enhancement.

The project employs the Python programming language along with the “OpenCV “and “face\_recognition libraries “ to create a functional face recognition . The system is trained on a collection of known faces, including prominent personalities such as "DR\_YASIR\_ARAFAT\_MALKANI," "QUAID\_E\_AZAM," "DR.HIRA\_FATIMAH," and

"MS\_NIRMA\_ABRO." These known faces are encoded to generate reference encodings for comparison.

**“The heart of the system lies in its ability to process real-time video frames captured from a webcam. Upon capturing each frame”**, the system detects facial features, computes face encodings, and compares them against the reference encodings of known individuals. In cases of positive matches, the system labels the detected face with the corresponding name and draws a bounding rectangle around it. If no match is found, the system classifies the face as **"Unknown."**

During development, we faced challenges like handling different lighting and camera angles, and dealing with image quality issues. The installation of the face recognition library (dlib) was a major roadblock at first, causing face recognition problems.

However, after seeking help online, sharing errors with the community, and persistent troubleshooting, we managed to solve the library installation issue. This effort led to a successful enhancement of our face recognition system..

The face recognition system not only showcases practical implementation but also highlights the significance of real-time capabilities in diverse fields. It enables efficient identification and authentication, contributes to personalized user experiences, and holds potential in retail, security, and interactive installations.

In conclusion, this report provides an in-depth exploration of the methodology, and considerations involved in developing a real-time face recognition system. By bridging theoretical concepts with practical application, the project underscores the promising possibilities of facial recognition technology in modern technological landscapes.

# Introduction

Face recognition technology has gained prominence in various applications, including security, authentication, and identification. This project aims to demonstrate the practical implementation of a face recognition system that operates in real-time and can accurately identify known individuals from a live video stream.

# Material and Methods

## Hardware Components

To develop and test the face recognition system, the following hardware components were utilized:

**Webcam or Video Capture Device**: A webcam or similar video capture device was used to capture live video frames. This input serves as the source for face detection and recognition.

## 1.1 Software Components and Libraries

The software components and libraries used in this project include:

**Python Programming Language**: Python serves as the core programming language for implementing the face recognition system and related tasks.

**OpenCV Library**: OpenCV provides a comprehensive set of tools for image and video processing. It is used to capture video frames, convert color spaces, and draw visual elements.

**face\_recognition Library**: The face\_recognition library simplifies face detection and recognition tasks by providing a high-level interface to deep neural networks.

**Visual Studio Code**: Visual Studio Code, a widely used integrated development environment (IDE), was utilized for writing, debugging, and managing the project's source code.

4.Methodology

The implemented methodology for real-time face recognition involves a step-by-step process that seamlessly combines face detection, encoding, comparison, and visualization. Below I have breakdown of how each key component contributes to the overall functionality:

### Loading Known Faces and Names

A list of known individuals' names and their respective images was assembled. These images served as reference points for face recognition.

### Encoding Known Faces

Using the face\_recognition library, the known face images were processed to generate numerical encodings. These encodings represented the unique features of each face.

### Video Capture and Preprocessing

The OpenCV library's module was utilized to access the webcam feed.

**VideoCapture**

Captured frames were converted from the BGR color space to the RGB color space using

the function for consistent processing.

**cvtColor**

### Real-Time Face Recognition Loop

The system entered a loop to process each frame of the video feed in real time.

### Face Detection and Encoding

The

library's

function identified face

locations within each frame. Face encodings were computed for each detected face

**face\_locations**

**face\_recognition**

using the function.

**face\_encodings**

### Comparing Face Encodings

Computed face encodings were compared against the known face encodings using the function. This step determined if the detected face matched any of

**compare\_faces**

the known individuals.

### Labeling and Visualization

In case of a match, the detected face was assigned the corresponding name. OpenCV's

and

**putText**

**rectangle**

face and display the name.

functions were employed to draw a rectangle around the

### Displaying Processed Frame

The processed frame, indicating recognized faces, was displayed in real time using

OpenCV's function.

**imshow**

### Termination

The loop continued until the user pressed the "z" key. Upon termination, the video capture was released, and OpenCV windows were closed.

The described methodology facilitated the development of a functional real-time face recognition system, demonstrating the integration of face detection, encoding, comparison, and visualization techniques to achieve the project's objectives.

# Key Functions and Methods

In this section, i will delve into how the key functions and methods outlined earlier work together cohesively to achieve the face recognition task.

### Loading Known Faces and Names: The

face\_recognition.load\_image\_file

function is used to load known face images from files. These images are processed and

list, while the corresponding names are stored in

stored in the

the known\_face\_names list.

**Video Capture and Processing:** The c

read

v2.VideoCapture

known\_face\_images

frames from the webcam. The

class captures live video method retrieves each frame, and the

function converts the color space from BGR to RGB for compatibility with the face recognition library.

cv2.cvtColor

**Face Detection and Recognition:** The face\_recognition.face\_locations function detects faces in the RGB video frame. The resulting bounding box coordinates (**face\_locations)** are used by subsequent functions.

**Display and Labeling:**

The face\_recognition.face\_encodings function computes facial encodings for each

detected face, generating a list of face\_encodings.

The function compares each computed

face\_recognition.compare\_faces

face encoding with the known face encodings to determine potential matches.

The

cv2.rectangle

face\_locations.

function draws rectangles around detected faces using

The function overlays recognized names onto the video frame based on

cv2.putText

the results of face comparison.

Together, these key functions and methods collaboratively execute the face recognition process:

1. Known faces are loaded and encoded.
2. Live video frames are captured, converted, and processed.
3. Faces are detected, encoded, and compared with known faces.
4. Recognized faces are visually labeled and displayed in real time.

# Complete Source Code

import cv2

import face\_recognition

# Load the known faces and their names

known\_face\_names = ["DR\_YASIR\_ARFAT\_MALKANI", "QUAID\_E\_AZAM", "DR.HIRA\_FATIMAH","MS\_NIRMA\_ABRO"]

known\_face\_images = [face\_recognition.load\_image\_file("SIR\_YASIR.jpg"), face\_recognition.load\_image\_file("QUAID\_E\_AZAM.jpg"), face\_recognition.load\_image\_file("MAM\_HIRA.jpg"), face\_recognition.load\_image\_file("NIRMA\_ABRO.jpeg")]

# Encode the known faces

known\_face\_encodings = [face\_recognition.face\_encodings(image)[0] for image in known\_face\_images]

video\_cap = cv2.VideoCapture(0)

while True:

ret, video\_data = video\_cap.read()

col = cv2.cvtColor(video\_data, cv2.COLOR\_BGR2RGB)

# Find all face locations and encodings in the current frame face\_locations = face\_recognition.face\_locations(col)

face\_encodings = face\_recognition.face\_encodings(col, face\_locations)

for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):

# Compare the face encoding of the detected face with the known face encodings

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

# Check if any face matches with the known faces for i in range(len(matches)):

if matches[i]:

name = known\_face\_names[i] break

# Draw a rectangle and display the name of the face cv2.rectangle(video\_data, (left, top), (right, bottom), (0, 255, 0), 2) cv2.putText(video\_data, name, (left, top - 10), cv2.FONT\_HERSHEY\_SIMPLEX,

0.7, (0, 255, 0), 2)

cv2.imshow("video\_live", video\_data)

if cv2.waitKey(10) == ord("z"): break

video\_cap.release() cv2.destroyAllWindows()

# OUTPUT :

1. **FLOWCHART:**

**Start**

**Convert to RGB**

**Capture Video Frame**

**Detect Faces**

**Generate Encodings**

**Draw Rectangles around Faces**

**Compare with Known Faces**

**Exit (User Presses "z" Key)**

**Loop Back (Continue Processing Frames)**

**Display Identified Name**

**End**

# Conclusion

The developed face recognition system showcases the application of advanced computer vision techniques to real-time video processing. The project effectively demonstrates the identification of known faces, drawing bounding boxes, and labeling individuals based on detected faces. This system can find potential application in security systems, user authentication, and various other domains where face recognition is required.

# Bibliography

* 1. Youtube Channel video (WSCube )
  2. OpenCV tutorial (ProgrammingKnowledge )
  3. OpenCV tutorial GeeksforGeek
  4. ChatGPT
  5. Google

# THE END