**PBR VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE**

**A**

**MINI PROJECT**

**ON**

**FACE RECOGNITION SYSTEM**

****

**AIM**: To develop a mini project on face recognition system via voice

**DESCRIPTION:**

Face recognition systems are technologies that identify or verify individuals by analyzing and comparing patterns based on their facial features. These systems typically involve the following steps:

**Detection:** The system identifies and locates human faces within an image or video frame. This step involves algorithms that can detect facial features like eyes, nose, mouth, etc.

**Alignment:** Once faces are detected, the system aligns them to a standard position and scale, often referred to as normalization. This step ensures that facial features are consistently positioned for accurate comparison.

**Feature Extraction:** The system extracts unique features from the face, which could include distances between eyes, the shape of the nose, the curve of the lips, etc. These features are then transformed into a mathematical representation, often called a face template or face signature.

**Matching:** The extracted features are compared with a database of known faces. This comparison can be done using various algorithms, such as Euclidean distance, neural networks, or deep learning models. The system determines the similarity between the extracted features and those stored in the database.

**Recognition/Verification:** Based on the similarity score, the system makes a decision about whether the face matches any of the faces in the database. In identification mode, it tries to determine the identity of the person by comparing the input face with all faces in the database. In verification mode, it confirms whether the input face matches a specific identity claimed by the user.

**Decision:** Finally, the system produces an output, which could be the identity of the person if recognized, or a rejection if the face doesn't match any known identities or if the similarity score is below a certain threshold.

**Multi-Modal Biometrics:** Face recognition systems can be integrated with other biometric modalities such as fingerprint recognition, iris recognition, or voice recognition to enhance accuracy and reliability, especially in scenarios where face recognition alone might be insufficient or unreliable.

**Real-Time Processing:** Many face recognition systems are designed for real-time processing, allowing them to quickly analyze and match faces in live video streams. This capability is crucial for applications like surveillance, access control, and monitoring public spaces.

**Robustness to Variability:** Face recognition systems must be robust to variations in facial appearance due to factors such as changes in lighting conditions, facial expressions, head pose, aging, and occlusions (e.g., wearing glasses or a hat). Advanced algorithms and techniques, such as deep learning and data augmentation, help improve robustness to such variations.

**Privacy and Ethical Considerations:** The widespread adoption of face recognition systems has raised concerns about privacy infringement and potential misuse. Issues such as unauthorized surveillance, profiling, and the collection of sensitive personal data have sparked debates about the ethical implications of deploying these technologies.

**Bias and Fairness:** Face recognition algorithms can exhibit biases, leading to inaccuracies and unfair treatment, particularly for certain demographic groups. Bias can arise from imbalanced training data, algorithmic design choices, or environmental factors. Addressing bias and ensuring fairness in face recognition systems is an ongoing challenge for researchers and practitioners.

**Regulatory and Legal Frameworks:** Governments and regulatory bodies are increasingly scrutinizing the use of face recognition technology, leading to the development of regulations and legal frameworks to govern its deployment. These regulations may dictate how face recognition can be used, data retention policies, consent requirements, and measures to ensure accountability and transparency.

**Advancements in Deep Learning:** Deep learning techniques, particularly convolutional neural networks (CNNs), have significantly advanced the state-of-the-art in face recognition. Deep learning models can automatically learn hierarchical representations of facial features, leading to improved accuracy and robustness compared to traditional methods.

**Challenges in Uncontrolled Environments:** Face recognition systems face significant challenges in uncontrolled environments, such as outdoor surveillance or public spaces, where factors like varying lighting conditions, occlusions, and diverse demographics can impact performance.

Face recognition systems are widely used in various applications, including security and surveillance, access control, law enforcement, and personal device authentication. However, they also raise concerns regarding privacy, surveillance, and potential biases in the algorithms used for detection and recognition.

**METHODOLOGY:**

Here we have used several functions which helps our program to run successfully.Some of the methods and their description are given below:

**cv2.VideoCapture()**: This function initializes the video capture from a camera, in this case, it captures video from the default camera (index 0).

**cv2.cvtColor()**: Converts an image from one color space to another. Here, it converts the captured frame from BGR (Blue-Green-Red) color space to RGB (Red-Green-Blue) color space.

**face\_recognition.face\_locations()**: Detects the locations of faces in the frame.

**face\_recognition.face\_encodings()**: Generates encodings (vectors) for the faces detected in the frame.

**face\_recognition.compare\_faces()**: Compares face encodings to determine if they match with known faces.

**cv2.rectangle()**: Draws a rectangle on the frame to highlight the recognized face.

**cv2.putText()**: Writes text on the frame, used here to display the name of the recognized person.

**pyttsx3.init()**: Initializes the text-to-speech engine.

**engine.say()**: Adds a speech utterance to the engine's queue.

**engine.runAndWait()**: Blocks until all currently queued speech is finished playing.

**sr.Recognizer()**: Initializes a recognizer for speech recognition.

**recognizer.adjust\_for\_ambient\_noise()**: Adjusts the recognizer for ambient noise levels.

**recognizer.listen()**: Listens for audio input from the microphone.

**recognizer.recognize\_google()**: Performs speech recognition using the Google Speech Recognition API.

**cv2.imshow()**: Displays an image or video frame in a window.

**cv2.waitKey()**: Waits for a specific time for a key event. Used here to break the loop if the 'q' key is pressed.

**video\_cap.release()**: Releases the video capture resource.

**cv2.destroyAllWindows()**: Destroys all created windows.

**SOURCE CODE:**

import cv2

import face\_recognition

import pyttsx3

import speech\_recognition as sr

# Load known face images and encode them

known\_faces = []

known\_names = []

person\_details = {} # Store details about recognized persons

# Define person\_data with details of recognized persons

person\_data = {

"virat-kohli": {"age": "32", "occupation": "Cricketer", "location": "India"},

"ali": {"age": "19", "occupation": "student", "location": "kavali"},

"sai": {"age": "21", "occupation": "student", "location": "kavali"},

"saaketh": {"age": "19", "occupation": "student", "location": "kavali"},

"uday": {"age": "19", "occupation": "student", "location": "kavali"}

# Add more details for other recognized persons as needed

}

# Load and encode known faces

known\_faces\_paths = [

"virat-kohli.png",

"sai.jpg",

"ali.jpg",

"saaketh.jpg",

"uday.jpg"

]

#Add more file paths as needed

for image\_path in known\_faces\_paths:

known\_image = face\_recognition.load\_image\_file(image\_path)

face\_encoding = face\_recognition.face\_encodings(known\_image)[0]

known\_faces.append(face\_encoding)

known\_names.append(image\_path.split(".")[0])

# Initialize speech engine

engine = pyttsx3.init()

# Initialize speech recognizer

recognizer = sr.Recognizer()

# Start video capture

video\_cap = cv2.VideoCapture(0)

# Initialize a flag to keep track of whether a face has been recognized or not

face\_recognized\_flag = False

while True:

ret, frame = video\_cap.read()

# Convert the frame to RGB for face\_recognition

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Find all face locations and encodings in the current frame

face\_locations = face\_recognition.face\_locations(rgb\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)

# Flag to determine if any face is recognized

face\_recognized = False

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

# Compare face encoding with the known face encodings

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

name = "Unknown"

# Check if the face is recognized

if any(matches):

face\_recognized = True

# If face recognized and response not given yet

if not face\_recognized\_flag:

face\_recognized\_flag = True

for i, match in enumerate(matches):

if match:

name = known\_names[i]

if name not in person\_details:

# If person details are available

if name in person\_data:

person\_details[name] = person\_data[name]

else:

# If person details are not available, provide a default response

person\_details[name] = {"age": "Unknown", "occupation": "Unknown", "location": "Unknown"}

break

# Draw rectangle around the face with green color

cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

# Put the name of the recognized person at the center of the rectangle

# Calculate the size of the text

text\_width, text\_height = cv2.getTextSize(name, cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, 1)[0]

# Calculate the position to center the text

text\_x = left + (right - left - text\_width) // 2

text\_y = bottom + 20 # Adjust as needed

cv2.putText(frame, name, (text\_x, text\_y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (0, 255, 0), 1)

# Speak out the name and details

engine.say("Hello, {}. Nice to meet you. Your age is {}. Your occupation is {}. Your location is {}.".format(name, person\_details[name]["age"], person\_details[name]["occupation"], person\_details[name]["location"]))

engine.runAndWait()

# If no face recognized, reset the flag

if not face\_recognized:

face\_recognized\_flag = False

engine.say("You are not in my database, but still nice to meet you.")

engine.runAndWait()

# Perform speech recognition

with sr.Microphone() as source:

print("Listening...")

recognizer.adjust\_for\_ambient\_noise(source, duration=0.5)

audio = recognizer.listen(source)

try:

# Recognize speech using Google Speech Recognition

user\_input = recognizer.recognize\_google(audio).lower()

print("User input:", user\_input)

engine.say("You said: " + user\_input)

engine.runAndWait()

except sr.UnknownValueError:

print("Google Speech Recognition could not understand audio")

except sr.RequestError as e:

print("Could not request results from Google Speech Recognition service; {0}".format(e))

# Display the frame

cv2.imshow("Video Live", frame)

# Break the loop if 'q' key is pressed

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

break

# Release video capture

video\_cap.release()

cv2.destroyAllWindows()

**OUTPUT:**

****

After recognising the face,it shares the details of a specific person via voice. For example;

Hello, virat kohli. Nice to meet you. Your age is 32. Your occupation is cricketer. Your location is India

Later it also takes the input voice from the user that is if user says any thing in voice then it responds the same message through voice.For example;

Listening…..

User said: Hey! This is Shaik Ali

You said:Hey! This is Shaik Ali

**RESULT:** We have successfully completed the mini project on face recognition system

via voice.