LIVE FACE DETECTION

Introduction

Face detection is a computer vision technique used to identify and locate human faces in digital images or video streams. In this project, Open CV(Open source Computer Visions) is used along with a Haar Cascade to detect faces in real time using a webcam.

OpenCV provides pre-trained classifiers and efficient image-processing functions that make real-time face detection possible with minimal computational cost.

System Components

The face detection system consists of the following components:

1. Webcam (input device)
2. OpenCV library
3. Haar Cascade face detection model
4. Python programming language
5. Display window (output)

OpenCV Library

OpenCV is an open-source library mainly used for:

- Image processing
- Video analysis
- Object detection
- Face detection and recognition

It supports multiple programming languages such as Python, C++, and Java. In this project, Python open (cv2) is used.

Haar Cascade Classifier

The Haar Cascade Classifier is a machine learning–based approach for object detection. It is trained using:

- Positive images (with faces)
- Negative images (without faces)

OpenCV provides a pre-trained XML file:

`haarcascade_frontalface_default.xml`

This file contains the data needed to detect frontal human faces.

The classifier works by:

- Extracting Haar features
- Using a cascade of classifiers
- Rejecting non-face regions quickly
- Detecting face regions efficiently

Coding

```
import cv2

import os

import numpy as np

from datetime import datetime



# ---------- PATHS ----------
```

```
KNOWN_DIR = "known_faces"

CAPTURE_DIR = "captured"

ATTENDANCE_FILE = "attendance.csv"



os.makedirs(CAPTURE_DIR, exist_ok=True)



# ---------- LOAD HAAR CASCADE----------

face_cascade = cv2.CascadeClassifier(

   cv2.data.haarcascades + "haarcascade_frontalface_default.xml"

)



# ---------- CREATE RECOGNIZER ----------

recognizer = cv2.face.LBPHFaceRecognizer_create()



faces=[]

labels=[]

names={}



label_id = 0



# ---------- LOAD KNOWN FACES ----------

for file in os.listdir(KNOWN_DIR):

   img_path = os.path.join(KNOWN_DIR, file)

   name = os.path.splitext(file)[0]



   img = cv2.imread(img_path)
```

```
    if img is None:
        continue

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    detected_faces = face_cascade.detectMultiScale(gray, 1.2, 5)

    for (x, y, w, h) in detected_faces:
        face = gray[y:y+h, x:x+w]
        face = cv2.resize(face, (200, 200))
        faces.append(face)
        labels.append(label_id)
        names[label_id] = name
        label_id += 1

if len(faces) == 0:
    print("No face found in known_faces folder")
    exit()

recognizer.train(faces, np.array(labels))
print("Known faces trained successfully")

# ---------- ATTENDANCE FUNCTION ----------
def mark_attendance(name):
    if not os.path.exists(ATTENDANCE_FILE):
        with open(ATTENDANCE_FILE, "w") as f:
            f.write("Name,Date,Time\n")
```

```
    now = datetime.now()
    with open(ATTENDANCE_FILE, "a") as f:
f.write(f"{name},{now.strftime('%Y-%m-%d')},{now.strftime('%H:%M:%S')}\n")

# ---------- START WEBCAM ----------
cap = cv2.VideoCapture(0)

recognized_once = False

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    detected_faces = face_cascade.detectMultiScale(gray,1.2,5)

    for (x, y, w, h) in detected_faces:
        face = gray[y:y+h, x:x+w]
        face = cv2.resize(face, (200, 200))

        label, confidence = recognizer.predict(face)

        if confidence < 65:
```

```
            name = names[label]
            text = f"✔ {name}"
            color = (0, 255, 0)

            if not recognized_once:
                #  CAPTURE PHOTO
                cv2.imwrite(os.path.join(CAPTURE_DIR, f"{name}.jpg"), frame)

                #  ATTENDANCE
                mark_attendance(name)

                recognized_once = True

        else:
            text = "Unknown"
            color = (0, 0, 255)

        cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
        cv2.putText(frame, text, (x, y-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)

    cv2.imshow("Face Recognition Attendance", frame)

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

```
cap.release()

cv2.destroyAllWindows()
```

Conclusion

This face detection system efficiently demonstrates how computer vision techniques can be applied in real time using OpenCV. By combining webcam input, grayscale conversion, and Haar Cascade classification, the system detects faces accurately and highlights them on screen while maintaining efficient resource usage.