Tutorial Report: Face Detection and Recognition using MTCNN and OpenCV

1. Introduction

This project demonstrates a complete pipeline for **face detection and recognition** using a webcam feed. It utilizes **MTCNN** for robust face detection and **OpenCV's LBPH** (Local Binary Patterns Histograms) algorithm for face recognition. Detected faces are stored, recognized in real time, and logged into a **SQLite database**.

---

2. System Overview

The system comprises three main stages:

1. **Face Capture** – Captures face images via webcam and stores them in a user-specific folder.
2. **Model Training** – Trains a face recognizer using the captured dataset.
3. **Real-time Recognition** – Detects and recognizes faces in webcam feed and logs metadata into an SQLite database.

---

3. Requirements

- Python 3.x
- OpenCV (opencv-contrib-python) – for LBPH face recognition
- mtcnn – for face detection
- numpy
- sqlite3 – standard library for database logging

Installation

pip install opencv-contrib-python mtcnn numpy

---

4. Code & Explanation

4.1 Imports

```
import cv2
import os
import numpy as np
from mtcnn import MTCNN
import sqlite3
from datetime import datetime
```

4.2 Capture User Face Images

```
def capture_faces(user_name):
    detector = MTCNN()
    cap = cv2.VideoCapture(0)
    save_path = f"dataset/{user_name}"
    os.makedirs(save_path, exist_ok=True)
    count = 0
    print("Press 's' to start, 'e' to end...")

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

```
        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):
            faces = detector.detect_faces(frame)
            for face in faces:
                x, y, w, h = face['box']
                face_img = frame[y:y+h, x:x+w]
                gray_face = cv2.cvtColor(face_img, cv2.COLOR_BGR2GR
                cv2.imwrite(f"{save_path}/{count}.jpg", gray_face)
                count += 1
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0
        elif key == ord('e'):
            break

        cv2.imshow("Capturing Faces", frame)
```

Explanation:
Captures face images using webcam and MTCNN. Press 's' to start saving, 'e' to exit. Images are stored as grayscale in dataset/{user_name}.

4.3 Train Face Recognizer

```
def train_recognizer():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    faces = []
    labels = []
    label_map = {}
    current_id = 0

    for user_name in os.listdir("dataset"):
        user_path = os.path.join("dataset", user_name)
        label_map[current_id] = user_name
        for img_name in os.listdir(user_path):
            img_path = os.path.join(user_path, img_name)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            faces.append(img)
            labels.append(current_id)
        current_id += 1
```

```
recognizer.train(faces, np.array(labels))
recognizer.save("face_model.yml")
np.save("label_map.npy", label_map)
```

Explanation:
Trains an LBPH face recognizer from the captured images and saves the model and label mapping for recognition.

4.4 Real-Time Detection and Recognition

```
def recognize_faces():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read("face_model.yml")
    label_map = np.load("label_map.npy", allow_pickle=True).item()
    detector = MTCNN()
    conn = sqlite3.connect("face_detection.db")
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS detections (timest

    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
```

```
faces = detector.detect_faces(frame)
for face in faces:
    x, y, w, h = face['box']
    face_img = frame[y:y+h, x:x+w]
    gray_face = cv2.cvtColor(face_img, cv2.COLOR_BGR2GRAY)
    try:
        gray_face = cv2.resize(gray_face, (100, 100))
        label, confidence = recognizer.predict(gray_face)
        if confidence < 70:
            name = label_map[label]
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S
            cursor.execute("INSERT INTO detections VALUES (?, ?)",
            conn.commit()
            cv2.putText(frame, f"{name} ({confidence:.1f})", (x, y
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0)
        else:
```

Explanation:
Performs real-time face detection and recognition. Detected names are logged into face_detection.db. Confidence threshold filters out uncertain predictions.

5. Output Files

File	Description
dataset/{user_name}/	Captured face images
face_model.yml	Trained LBPH face model
label_map.npy	Label mapping for recognition
face_detection.db	SQLite DB of recognized faces and timestamps

---

6. Usage Summary

1. Capture Faces

capture_faces("john")

2. Train Recognizer

train_recognizer()

3. Run Recognition

recognize_faces()

---

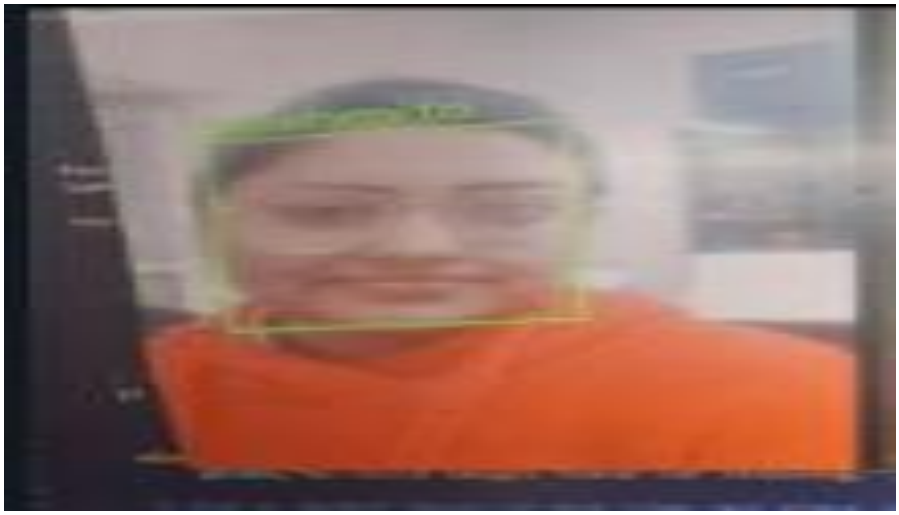
7. Notes

- Capture at least **20–30** images per user for better accuracy.
- Ensure good lighting during image capture.
- MTCNN may struggle in poor lighting or with extreme angles.

---

8. Possible Enhancements

Feature	How to Add
Email alert on unknown face	Integrate smtplib
UI to manage users	Use Tkinter or PyQt
REST API for results	Use Flask
Cloud sync for logs	Integrate with Firebase or AWS S3



```
Predicted Label: 3, Confidence: 55.245768270479O5
Detected: yashmita at 2025-01-30 16:24:39 with confidence 55.2
```