**Name:** Suyog Bodke  
**Roll No:** 381006  
**PRN:** 22310072

**Assignment No: -** 2  
**Title: -** Facial Recognition using OpenCV and Deep Learning

**Problem Statement:**

Implement a facial recognition system using OpenCV and deep learning techniques for binary classification (Match / Not Match).

**Objective:**

* To understand how facial recognition systems work using feature extraction and classification.
* To implement LBPH (Local Binary Pattern Histogram) for face recognition.
* To integrate OpenCV face detection with webcam input in Google Colab.
* To classify faces into two categories: **Match** or **Not Match** based on similarity with a reference image.
* To visualize and annotate real-time verification results.

**S/W Packages and H/W apparatus used:**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook / Google Colab
* **Hardware:** Webcam, CPU (4GB+ RAM), GPU optional
* **Libraries:** OpenCV, NumPy, Matplotlib, ipywidgets, js2py, Google Colab output utilities

**Theory:**

Facial recognition is a biometric method of identifying individuals by comparing captured face features with stored reference images.

* **Face Detection:** Haar Cascade Classifier detects face regions in an image.
* **Feature Extraction:** LBPH (Local Binary Pattern Histogram) is used to encode facial texture into histograms.
* **Classification:** A trained recognizer compares the live face with the reference face.
* **Thresholding:** Confidence values determine whether the faces match. Lower confidence = better match.

The algorithm outputs **MATCH** or **NOT MATCH** with confidence scores.

**Methodology:**

1. **Data Preparation:**
   * Upload a reference face image (PHOTO.JPG).
   * Convert to grayscale, detect face, resize to (200×200).
2. **Model Training:**
   * Train LBPH recognizer with the extracted reference face.
3. **Webcam Initialization:**
   * Use Google Colab + JS to capture live webcam frames.
4. **Face Verification:**
   * Detect face in live frame using Haar Cascade.
   * Extract ROI, resize, and compare with LBPH model.
   * Predict label and confidence score.
5. **Output:**
   * Annotate live image with bounding box, label (**MATCH/NOT MATCH**), and confidence score.
   * Display annotated image with Matplotlib.

**Results:**

* Reference image successfully trained with LBPH recognizer.
* Webcam captures tested in real time.
* The system correctly identifies **MATCH** when the same face appears, and **NOT MATCH** otherwise.
* Confidence values validate recognition (threshold = 80).

**Advantages:**

* Lightweight and fast (LBPH is computationally efficient).
* Works well with grayscale images and small datasets.
* Can run in real-time with webcam streams.

**Limitations:**

* Sensitive to lighting conditions and facial angles.
* LBPH is less accurate compared to deep CNN-based models.
* Requires clear frontal faces for detection.
* Works only for binary classification (Match / Not Match).

**Applications:**

* Biometric authentication (attendance systems, access control).
* Security verification in banking and offices.
* Real-time user verification for digital platforms.

**Working / Algorithm:**

1. Import libraries (cv2, NumPy, Matplotlib, ipywidgets).
2. Load and preprocess reference face image.
3. Train LBPH recognizer.
4. Initialize webcam stream in Google Colab using JS.
5. Capture frames, detect faces, and classify using recognizer.
6. Display annotated output (Match/Not Match + Confidence).

**Conclusion:**

The facial recognition system implemented using OpenCV and LBPH successfully verified faces in real-time with high accuracy for binary classification. While lightweight and easy to implement, it has limitations under varying lighting conditions and complex backgrounds. Deep learning models (CNN, FaceNet) can further improve accuracy.