Face Recognition System Documentation

Overview

This Python script implements a real-time face recognition system using OpenCV. It trains a face recognizer on a dataset of facial images and then uses the trained model to identify faces from a webcam feed, granting access to authorized personnel.

Dependencies

```
import cv2          # OpenCV library for computer vision tasks
import numpy as np  # Numerical computing library
import os           # Operating system interface for file operations
```

Key Components

1. Face Detection Classifier

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_defaul
```

- Uses Haar Cascade classifier for detecting frontal faces in images
- Pre-trained model provided by OpenCV

2. Face Recognizer

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

- Implements Local Binary Patterns Histograms (LBPH) face recognizer
- Robust against lighting changes and computationally efficient

Dataset Structure

```
dataset/
  Person1_Name/
      image1.jpg
      image2.jpg
      ...
  Person2_Name/
      image1.jpg
      ...
  ...
```

Training Process

1. Data Collection

```
dataset_path = "dataset"
faces, labels = [], []
```

```
label_to_name = {}
label_count = 0
```

- Initializes empty lists for storing face data and labels
- Creates mapping between numerical labels and person names

2. Image Processing Loop

```
for person_name in os.listdir(dataset_path):
    person_path = os.path.join(dataset_path, person_name)
    # Process each image in person's directory
    for image_name in os.listdir(person_path):
        img = cv2.imread(img_path)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces_detected = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5
```

- Converts images to grayscale
- Detects faces using Haar Cascade
- Parameters: `scaleFactor=1.1` (image scale reduction), `minNeighbors=5` (detection sensitivity)

3. Face Extraction and Resizing

```
face_resized = cv2.resize(gray[y:y+h, x:x+w], (200, 200))
faces.append(face_resized)
labels.append(label_count)
```

- Extracts detected face region
- Resizes to consistent 200×200 pixels for uniform training
- Appends to training data with corresponding label

4. Model Training

```
faces = np.array(faces, dtype=np.uint8)
labels = np.array(labels, dtype=np.int32)
recognizer.train(faces, labels)
```

- Converts data to NumPy arrays with appropriate data types
- Trains the LBPH recognizer on the collected faces

Real-Time Recognition

1. Video Capture Setup

```
cap = cv2.VideoCapture(0)  # Initialize webcam (device index 0)
AUTHORIZED_PERSON = "Rajendra Chimala"
CONFIDENCE_THRESHOLD = 75  # Threshold for recognition confidence
```

2. Main Recognition Loop

```
while True:
    ret, frame = cap.read()  # Capture frame from webcam
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces_detected = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors
```

- Continuously captures frames from webcam
- Converts to grayscale and detects faces in each frame

3. Face Prediction and Recognition

```
for (x, y, w, h) in faces_detected:
    face_resized = cv2.resize(gray_frame[y:y+h, x:x+w], (200, 200))
    label, confidence = recognizer.predict(face_resized)
    name = label_to_name.get(label, "Unknown")
```

- Processes each detected face
- Uses trained model to predict identity and confidence level
- Retrieves name from label mapping (defaults to "Unknown")

4. Access Control Logic

```
if confidence < CONFIDENCE_THRESHOLD:
    color = (0, 255, 0)  # Green - Authorized
    if name == AUTHORIZED_PERSON:
        print("Access Granted")
else:
    color = (0, 0, 255)  # Red - Unauthorized or low confidence
```

- Grants access if confidence is below threshold AND person is authorized
- Visual feedback through colored bounding boxes

5. Display and Annotation

```
cv2.putText(frame, f"[{name} ({int(confidence)})]", (x, y-10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)
cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
cv2.imshow("Face Recognition", cv2.resize(frame,(400,400)))
```

- Annotates frame with recognition results and confidence score
- Displays resized frame for better viewing

Exit Condition

```
if cv2.waitKey(1) & 0xFF == ord(' '):  # Press spacebar to exit
    break
```

Cleanup

```
cap.release()           # Release webcam resource
cv2.destroyAllWindows() # Close all OpenCV windows
```

Configuration Parameters

- `CONFIDENCE_THRESHOLD = 75`: Lower values mean more strict recognition
- `scaleFactor=1.1`: Controls image scaling for detection
- `minNeighbors=5`: Higher values reduce false positives but may miss faces
- Frame size: 400×400 pixels for display

Usage

1. Create dataset directory with subdirectories for each person
2. Place multiple facial images in each person's directory
3. Run the script to train the model
4. Webcam feed will open with real-time recognition
5. Press spacebar to exit

Notes

- Ensure good lighting conditions for better recognition accuracy
- Use frontal face images with minimal obstructions for training
- Higher quality training images yield better recognition results
- System performance may vary based on hardware capabilities