Introduction

Face recognition technology has gained significant popularity in security, authentication, and automated systems. This project implements a Face Recognition and Attendance System using Python, OpenCV, and Tkinter. The system captures and stores face images, trains a machine learning(classification) model for recognition, and marks attendance in an Excel file.

Objective

The objective of this project is to develop an automated Face Recognition and Attendance System that accurately detects and recognizes individuals for seamless attendance tracking. Using OpenCV, Tkinter, and Pandas, it eliminates manual effort by capturing face images, training a recognition model, and marking attendance in an Excel file while preventing duplicates. The system enhances accuracy, security, and efficiency, making it a scalable solution for workplaces and educational institutions.

Key Features

- **User Registration:** Captures student details and assigns a unique ID.
- **Dataset Generation:** Captures and stores face images.
- **Model Training:** Trains a LBPH-based classification model.
- **Face Detection & Recognition:** Detects faces and recognizes registered users.
- **Attendance Marking:** Records attendance with date and time.

Tech-Stack

- **OpenCV –** Image processing and face recognition
- **Tkinter –** GUI design
- **Pandas –** Attendance management (Excel handling)
- **Pillow –** Image handling

Features

1. User Interface (Tkinter GUI)

Graphical User Interface (GUI) is developed using Tkinter. It allows users to:

1. Enter personal details (Name, Age, Address)
2. Capture and store face images
3. Train a face recognition model
4. Detect faces and mark attendance

2. Dataset Generation

To train the model, face images are collected using OpenCV and stored in a dataset.

Execution Flow:

- The system captures the user's face using a webcam.
- The face is cropped using a Haar Cascade classifier.
- 300 images per user are saved in a folder (data/).
- Each image is labelled with a unique ID for training.

3. Face Recognition Model Training

The system uses **LBPH (Local Binary Patterns Histograms) a classification algorithm in OpenCV** for face recognition.

Process:

1. Reads images from the dataset.
2. Converts them into grayscale.
3. Extracts features and trains the LBPH model.
4. Saves the trained model as "classifier.xml".

4. Face Detection and Attendance Marking

When a person is detected, the system:

1. **Detects Faces:** Uses a Haar Cascade classifier.
2. **Recognizes the Face:** Compares the detected face with the trained model.
3. **Displays Name & Confidence:** If recognized, their name and confidence score appear.
4. **Marks Attendance:** If a recognized face is detected, the system:
    - Records the **User ID, Name, Date, and Time** in an **Excel file (attendance.xlsx)**.
    - Prevents duplicate attendance on the same day.

Code Overview

2. generate_dataset():

```python
def generate_dataset():
    global user_id_counter

    name, age, address = t1.get(), t2.get(), t3.get()
    if not (name and age and address):
        messagebox.showinfo("Result", "Please provide complete details of the user")
        return

    # Assign an ID and store user data
    user_data[user_id_counter] = name

    face_classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

    def face_cropped(img):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray, 1.3, 5)

        if len(faces) == 0:
            return None

        for (x, y, w, h) in faces:
            cropped_face = img[y:y+h, x:x+w]
            return cropped_face

    cap = cv2.VideoCapture(0)
    img_id = 0

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: Failed to capture image")
            break

        cropped_face = face_cropped(frame)

        if cropped_face is not None:
            img_id += 1
            face = cv2.resize(cropped_face, (200, 200))
            face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

            file_name_path = f"data/user.{user_id_counter}.{img_id}.jpg"
            cv2.imwrite(file_name_path, face)

            cv2.imshow("Cropped Face", face)

        if cv2.waitKey(1) == 13 or img_id == 200:
            break

    cap.release()
    cv2.destroyAllWindows()

    messagebox.showinfo("Result", f"Dataset Generated for {name} (ID: {user_id_counter})!")
    user_id_counter += 1
```

- This function captures 200 images of a user's face using a webcam.

- It uses a Haar cascade classifier to detect and crop faces.
- The images are converted to grayscale and saved in the data folder.
- Each image is labeled with a user ID and image number.

3. train_classifier():

```
def train_classifier():
    data_dir = "data"
    path = [os.path.join(data_dir, f) for f in os.listdir(data_dir)]
    faces, ids = [], []

    for image in path:
        img = Image.open(image).convert("L")
        imageNp = np.array(img, "uint8")
        id = int(os.path.split(image)[1].split(".")[1])
        faces.append(imageNp)
        ids.append(id)

    ids = np.array(ids)

    clf = cv2.face.LBPHFaceRecognizer_create()
    clf.train(faces, ids)
    clf.write("classifier.xml")

    messagebox.showinfo("Result", "Training Completed!")
```

- It reads face images from the data folder.
- The LBPH (Local Binary Pattern Histogram) face recognizer is trained using these images.
- User IDs are extracted from image filenames for labeling.
- The trained model is saved as classifier.xml for face recognition.

4. detect_face():

```
def detect_face():
    faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    clf = cv2.face.LBPHFaceRecognizer_create()
    clf.read("classifier.xml")

    attendance_file = "attendance.xlsx"
    if not os.path.exists(attendance_file):
        df = pd.DataFrame(columns=["ID", "Name", "Time"])
        df.to_excel(attendance_file, index=False)

    def mark_attendance(user_id, name):
        df = pd.read_excel(attendance_file)
        today = datetime.now().date()
        if not ((df["ID"] == user_id) & (pd.to_datetime(df["Time"]).dt.date == today)).any():
            new_entry = pd.DataFrame([[user_id, name, datetime.now().strftime("%Y-%m-%d %H:%M:%S")]],
                                     columns=["ID", "Name", "Time"])
            df = pd.concat([df, new_entry], ignore_index=True)
            df.to_excel(attendance_file, index=False)
```

- This function uses the webcam to detect and recognize faces using the trained model.
- Detected faces are labeled with names and confidence scores.
- Recognized users have their attendance marked using the mark_attendance() function.
- Unknown faces are labeled as "UNKNOWN" on the screen.

5. mark_attendance():

```
def mark_attendance(user_id, name):
    df = pd.read_excel(attendance_file)
    today = datetime.now().date()
    if not ((df["ID"] == user_id) & (pd.to_datetime(df["Time"]).dt.date == today)).any():
        new_entry = pd.DataFrame([[user_id, name, datetime.now().strftime("%Y-%m-%d %H:%M:%S")]],
                                 columns=["ID", "Name", "Time"])
        df = pd.concat([df, new_entry], ignore_index=True)
        df.to_excel(attendance_file, index=False)

def draw_boundary(img, classifier, scaleFactor, minNeighbors, color, clf):
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    features = classifier.detectMultiScale(gray_img, scaleFactor, minNeighbors)

    for (x, y, w, h) in features:
        cv2.rectangle(img, (x, y), (x+w, y+h), color, 2)
        id, pred = clf.predict(gray_img[y:y+h, x:x+w])
        confidence = 100 - pred

        if confidence > 30:
            name = user_data.get(id, "Unknown")
            cv2.putText(img, f"{name} ({confidence:.2f}%)", (x, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 1, cv2.LINE_AA)
            if name != "Unknown":
                mark_attendance(id, name)
        else:
            cv2.putText(img, "UNKNOWN", (x, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 1, cv2.LINE_AA)

    return img

video_capture = cv2.VideoCapture(0)
while True:
    ret, img = video_capture.read()
    if not ret:
        break
    img = draw_boundary(img, faceCascade, 1.1, 10, (255, 255, 255), clf)
    cv2.imshow("Face Detection", img)
    if cv2.waitKey(1) == 13:
        break
video_capture.release()
cv2.destroyAllWindows()
```

- It logs user attendance in an attendance.xlsx file using Pandas.
- The function records the user ID, name, and current timestamp.
- It checks for duplicate entries to prevent marking attendance multiple times a day.
- If no file exists, a new one is created with appropriate columns.

Advantages

I. **Effortless Attendance:** Traditional methods, such as manual registers or online forms, often consume valuable lecture time. In contrast, this system enables students to mark their attendance instantly through facial recognition, eliminating delays.

II. **Convenience for Educators:** Teachers no longer need to manage attendance sheets or manually update records. Once a student's face is recognized, the system automatically logs their presence for the day, streamlining the process.

III. **Enhanced Functionality:** With attendance data stored in an Excel sheet, various insights can be easily retrieved, such as individual attendance records, identifying students with less than 75% attendance, and analyzing average attendance per session.

Disadvantages

I. **Large Dataset Requirement:** To achieve high accuracy, multiple facial images from different angles are needed for each student, leading to significant storage consumption.

II. **Camera Dependency:** Recognition accuracy heavily relies on the camera's resolution. Lower-quality cameras may struggle with identification, while high-resolution cameras increase costs.

III. **Computational Complexity:** Advanced models, such as CNNs, may be needed for improved detection, demanding high processing power and longer training times.

IV. **Lighting and Environmental Factors:** Variations in lighting conditions, shadows, or background changes can affect recognition performance, leading to misidentification or failed detections.

Future Enhancements

- Improve accuracy using Deep Learning (CNN models).
- Increase accuracy by more efficient code and CNN models.
- Store data in a database instead of an Excel file.

Conclusion

This project effectively brings to life an automated Face Recognition Attendance System powered by Python, seamlessly integrating real-time detection, recognition, and attendance marking. By leveraging advanced computer vision techniques, it ensures a smooth and hassle-free experience, eliminating the inefficiencies of traditional methods. With its ability to enhance both accuracy and security, this system holds immense potential for deployment in educational institutions, corporate environments, and various organizations. Its implementation not only simplifies attendance tracking but also lays the foundation for a more efficient, reliable, and technology-driven future in identity verification and record management.