Face Recognition Attendance System - Documentation

Overview

This Python-based Face Recognition Attendance System automatically records user attendance using facial recognition through a webcam. The system captures face images, encodes them, and matches them in real-time to mark attendance into a CSV file.

Technologies Used

- **Python 3.x**
- **OpenCV** (cv2) - For image and video processing
- **face_recognition** - For detecting and recognizing faces
- **pickle** - For saving and loading encoded face data
- **pandas** - For managing attendance records in CSV format
- **datetime** - For recording current date and time

Library Installation (including dlib)

To install the required libraries and set up dlib, follow the steps:

1. Install CMake (for building dlib)

```
pip install cmake
```

2. Install dlib

Ensure CMake is installed. Then:

```
pip install dlib
```

If this fails, install dlib from a prebuilt binary or manually compile it using CMake and Visual Studio (on Windows).

3. Install remaining dependencies

```
pip install opencv-python face_recognition pandas
```

Folder Structure

```
project_folder/
│
├── dataset/            # Stores user images
│   └── <user_name>/      # Folder per user with 20 face images
├── encodings.pickle      # Serialized face data
├── attendance.csv        # CSV log of attendance
└── face_attendance.py      # Main Python file
```

How It Works

Step 1: Register User

Function: register_user(name)

Creates a folder dataset/<name>

Captures 20 images from webcam

Stores them in the folder

Step 2: Encode Faces

Function: encode_faces()

Reads all images from dataset

Converts them to RGB and encodes them (128-d vectors)

Saves data to encodings.pickle

Step 3: Mark Attendance

Function: mark_attendance(name)

Loads/creates attendance.csv

Adds a new row only if the user hasn't been marked for today

Step 4: Real-Time Face Recognition

Function: recognize_faces()

Opens webcam

Detects and encodes faces in live frames

Matches with encodings.pickle

If matched, displays name and calls mark_attendance

4. Required Package Installation

1. Install Python

Make sure Python is installed (preferably Python 3.8 - 3.10). You can download it from:

https://www.python.org/downloads/

2. Install Required Python Libraries

You can install all necessary Python libraries using pip:

```
bash
CopyEdit
pip install opencv-python
pip install face_recognition
pip install numpy
pip install pandas
```

3. Install CMake

CMake is required to build dlib (a core part of the face_recognition library)

Download and install from:
https://cmake.org/download/

4. Install Visual C++ Build Tools (Compiler for dlib)

face_recognition and dlib require Visual C++ to compile.

Download the Microsoft Visual C++ Build Tools from:
https://visualstudio.microsoft.com/visual-cpp-build-tools/

OR install the latest Microsoft Visual C++ Redistributable:
https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist

5. Install dlib (using .whl for Windows)

You may get errors installing `dlib` directly via pip, so use a precompiled `.whl` file:

Visit the following GitHub repository for `.whl` files:
https://github.com/ZMahmud22/Dlib_Windows_Python3.x

Choose the correct `.whl` file for your Python version and Windows architecture. Example for Python 3.10 (64-bit)

```
bash
CopyEdit
pip install dlib- 19.22.99- cp310- cp310- win_amd64.whl
```

Usage Instructions

1. Register a User

Uncomment the line in `__main__`:

```
register_user("YourName")
```

Run the script to capture 20 images.

2. Encode the Faces

```
encode_faces()
```

Run this to generate `encodings.pickle`.

3. Start Face Recognition & Mark Attendance

```
recognize_faces()
```

This will:

- Show webcam feed
- Recognize known users
- Mark attendance in `attendance.csv`

Press **ESC** key to stop.

Output

- Recognized faces get a bounding box and name displayed on-screen.
- Attendance is saved in:

```
Name,Date,Time
Pravallika,26-05-2025,14:34:12
```

Notes

- Use good lighting and clean camera lens for better results
- Images must clearly show the face

Each user should have at least 20 varied-angle face images for accurate recognition.

Future Enhancements

Connect to Firebase or database instead of CSV

Add GUI interface for easier use

Optimize performance using threading