Face Recognition System: Documentation

Overview

This project consists of two Python scripts that together implement a simple face recognition system. The system:

1. Captures and stores face data in a structured format.
2. Uses a K-Nearest Neighbors (KNN) algorithm to classify and recognize faces in real-time.

Key Features

- **Face Data Capture Script**: Captures images from a webcam, detects faces using Haar cascades, and stores them in NumPy arrays for training. **(generating_training_data.py)**
- **Face Recognition Script**: Uses stored training data to recognize faces in real-time webcam feed and displays the predicted name along with a bounding box.**(BuildingFaceClassifier.py)**

Script 1: Generating_training_data.py

Purpose

Captures images from the webcam, detects faces, and saves them in a NumPy array format for training purposes.

How It Works

1. **Initialize Camera**: Start capturing video feed from the webcam.
2. **Detect Faces**: Use Haar cascades to detect faces in real-time.
3. **Extract Largest Face**: Identify and extract the largest face in the frame.
4. **Save Face Data**: Flatten and resize the extracted face region and store it in a NumPy array.
5. **Repeat for Multiple Faces**: Collect data for different individuals to build a dataset.

Code Walkthrough

- **Face Detection**: Uses the `haarcascade_frontalface_alt.xml` model for detecting faces.
- **Face Extraction**: Extracts and resizes the face to a standard size (100x100 pixels).
- **Data Saving**: Stores the processed data into `./data/` directory as `.npy` files, named after the user.

Usage Instructions

1. Run the script.
2. Enter the name of the person whose face data is being captured.
3. Use the webcam feed to capture multiple images of the person's face.
4. Press `q` to quit the script.

Script 2: BuildingFaceClassifier.py

Purpose

Classifies and recognizes faces in real-time using stored training data and the KNN algorithm.

How It Works

1. **Load Training Data**: Loads all `.npy` files from the `./data/` directory.
2. **Prepare Dataset**: Combines face data and corresponding labels into a training set.
3. **Initialize Webcam**: Captures live video feed.
4. **Detect Faces**: Detects faces in each frame using Haar cascades.
5. **Recognize Faces**: Classifies the detected faces using the KNN algorithm.
6. **Display Results**: Draws a bounding box around the face and displays the recognized name.

Code Walkthrough

- **Data Loading**: Reads face data and assigns unique labels to each person.
- **KNN Algorithm**: A custom implementation of the KNN classification algorithm.
- **Real-Time Recognition**: Predicts the label of each detected face and maps it to the person’s name.

Usage Instructions

1. Run the script.
2. Ensure that the `./data/` directory contains `.npy` files from the first script.
3. Use the webcam feed to detect and recognize faces.

4. Press `q` to quit the script.

Requirements

Libraries

- `OpenCV` for face detection and video capture.
- `NumPy` for data manipulation.

Files

- `haarcascade_frontalface_alt.xml`: Pre-trained Haar cascade model for face detection.

Prerequisites

Install required libraries:
pip install opencv-python numpy

1.
2. Place `haarcascade_frontalface_alt.xml` in the working directory.
3. Create a `./data/` folder to store training data.

Key Concepts

Haar Cascades

Haar cascades are pre-trained classifiers used for object detection. They work by detecting features such as edges, lines, and rectangles in images.

K-Nearest Neighbors (KNN)

KNN is a simple and effective classification algorithm that works by finding the `k` nearest neighbors of a test point in the feature space and assigning the most common label among them.

Data Preparation

The face data is flattened into a 1D array and stored with corresponding labels. This ensures compatibility with the KNN algorithm.

Notes and Tips

1. **Data Collection**
    - Ensure good lighting while capturing face data.
    - Capture images from different angles to improve recognition accuracy.
2. **Performance Optimization**
    - Experiment with the `k` value in the KNN algorithm for better results.
    - Use a GPU-accelerated version of OpenCV for faster processing.
3. **Troubleshooting**
    - If no faces are detected, verify the `haarcascade_frontalface_alt.xml` path.
    - Ensure the `./data/` directory contains valid `.npy` files for recognition.
