Build a Facial recognition system

Building a facial recognition system involves several steps, including data collection, face detection, feature extraction, and face recognition. We'll use Python and some popular libraries like OpenCV, Dlib, and face_recognition to implement this. Here's a detailed guide:

Step 1: Install Necessary Libraries

First, install the required libraries. You can install them using pip:

sh

Copy code

pip install opencv-python dlib face_recognition

Step 2: Data Collection

Collect images of the individuals you want to recognize. Ensure you have a sufficient number of images for each person.

Step 3: Face Detection and Feature Extraction

We'll use the face_recognition library to detect faces and extract features.

Step 4: Face Recognition

We'll recognize faces by comparing the extracted features with those of known individuals.

Here's a step-by-step implementation:

Step 1: Import Libraries

python

Copy code

import cv2

import face_recognition

import os

import numpy as np

Step 2: Load Images and Encode Faces

Create a directory structure where each subdirectory contains images of a single person. For example:

Copy code

known_faces/

```
    person1/
        img1.jpg
        img2.jpg
    person2/
        img1.jpg
        img2.jpg
python
Copy code
def load_images_from_folder(folder):
    images = []
    labels = []
    for subdir in os.listdir(folder):
        subdir_path = os.path.join(folder, subdir)
        if os.path.isdir(subdir_path):
            for filename in os.listdir(subdir_path):
                img_path = os.path.join(subdir_path, filename)
                img = cv2.imread(img_path)
                if img is not None:
                    images.append(img)
                    labels.append(subdir)
    return images, labels

known_faces_dir = 'known_faces'
known_images, known_labels = load_images_from_folder(known_faces_dir)

# Encode faces
known_face_encodings = []
known_face_names = []
for img, label in zip(known_images, known_labels):
```

```
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    encodings = face_recognition.face_encodings(img_rgb)
    if encodings:
        known_face_encodings.append(encodings[0])
        known_face_names.append(label)
```

Step 3: Recognize Faces in a Video Stream

python

Copy code

```
def recognize_faces_in_video(known_face_encodings, known_face_names):
    video_capture = cv2.VideoCapture(0)

    while True:
        ret, frame = video_capture.read()
        if not ret:
            break

        # Convert the frame to RGB
        rgb_frame = frame[:, :, ::-1]

        # Find all face locations and face encodings in the frame
        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

        # Loop through each face in this frame
        for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown"

            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
```

```
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            # Draw a box around the face
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
            cv2.putText(frame, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255, 255), 1)

        # Display the resulting image
        cv2.imshow('Video', frame)

        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    video_capture.release()
    cv2.destroyAllWindows()

recognize_faces_in_video(known_face_encodings, known_face_names)
```

Full Implementation

Here's the complete script:

python

Copy code

```
import cv2
import face_recognition
import os
import numpy as np
```

```
def load_images_from_folder(folder):
    images = []
    labels = []
    for subdir in os.listdir(folder):
        subdir_path = os.path.join(folder, subdir)
        if os.path.isdir(subdir_path):
            for filename in os.listdir(subdir_path):
                img_path = os.path.join(subdir_path, filename)
                img = cv2.imread(img_path)
                if img is not None:
                    images.append(img)
                    labels.append(subdir)
    return images, labels

known_faces_dir = 'known_faces'
known_images, known_labels = load_images_from_folder(known_faces_dir)

# Encode faces
known_face_encodings = []
known_face_names = []
for img, label in zip(known_images, known_labels):
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    encodings = face_recognition.face_encodings(img_rgb)
    if encodings:
        known_face_encodings.append(encodings[0])
        known_face_names.append(label)

def recognize_faces_in_video(known_face_encodings, known_face_names):
    video_capture = cv2.VideoCapture(0)
```

```
while True:
    ret, frame = video_capture.read()
    if not ret:
        break

    # Convert the frame to RGB
    rgb_frame = frame[:, :, ::-1]

    # Find all face locations and face encodings in the frame
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    # Loop through each face in this frame
    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.putText(frame, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255,
255), 1)

    # Display the resulting image
```

```
        cv2.imshow('Video', frame)

        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    video_capture.release()
    cv2.destroyAllWindows()
```

recognize_faces_in_video(known_face_encodings, known_face_names)

This script captures video from the webcam, detects faces, extracts their features, and recognizes them based on previously encoded faces. Adjust the known_faces directory structure and content