import cv2

import face\_recognition

import os

# Load the known faces and their names from the faces directory

known\_faces = []

known\_names = []

faces\_dir = "faces" # Directory containing the face images

for name in os.listdir(faces\_dir):

known\_names.append(name.split('.')[0]) # Extract name from file name (assuming file name is in the format 'name.jpg')

face\_image = face\_recognition.load\_image\_file(os.path.join(faces\_dir, name))

face\_encoding = face\_recognition.face\_encodings(face\_image)[0]

known\_faces.append(face\_encoding)

# Initialize the camera

cap = cv2.VideoCapture(0) # Use camera index 0 (default) for the built-in camera

while True:

ret, frame = cap.read() # Capture frame from camera

if not ret:

print("Failed to capture frame from camera")

break

# Convert the frame from BGR (Blue-Green-Red ) to RGB format

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Detect faces in the frame

face\_locations = face\_recognition.face\_locations(rgb\_frame) #chứa tọa độ hộp giới hạn (trên, phải, dưới, trái) của các khuôn mặt được phát hiện trong khung. Các tọa độ này xác định vùng hình chữ nhật xung quanh mỗi khuôn mặt được phát hiện.

face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations) #chứa mã hóa khuôn mặt của các khuôn mặt được phát hiện trong khung. Mã hóa khuôn mặt là biểu diễn véc tơ của các đặc điểm khuôn mặt có thể được sử dụng để nhận dạng khuôn mặt.

# Loop through detected faces

for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):

# So sánh mã hóa khuôn mặt được phát hiện với mã hóa khuôn mặt đã biết

matches = face\_recognition.compare\_faces(known\_faces, face\_encoding)

name = "Unknown" # Default name if face is not recognized

# Calculate face recognition percentage

face\_distances = face\_recognition.face\_distance(known\_faces, face\_encoding)

#Dòng mã này sử dụng thư viện để tính toán khoảng cách Euclide giữa mã hóa khuôn mặt được phát hiện và mã hóa khuôn mặt đã biết. Hàm lấy danh sách mã hóa khuôn mặt đã biết và mã hóa của khuôn mặt được phát hiện làm đầu vào, đồng thời trả về danh sách các giá trị số biểu thị khoảng cách giữa khuôn mặt được phát hiện và từng khuôn mặt đã biết.

min\_distance = min(face\_distances)

#Dòng mã này tính toán khoảng cách tối thiểu từ danh sách khoảng cách khuôn mặt, đại diện cho kết quả khớp gần nhất với khuôn mặt được phát hiện trong số các khuôn mặt đã biết.

percentage = (1 - min\_distance) \* 100

#Dòng mã này chuyển đổi khoảng cách tối thiểu thành giá trị phần trăm, trong đó phần trăm cao hơn cho biết khuôn mặt được phát hiện và khuôn mặt đã biết gần giống nhau hơn. Tỷ lệ phần trăm được tính bằng (1 - khoảng cách tối thiểu) nhân với 100 để chuyển đổi thành tỷ lệ phần trăm.

if any(matches):

# Gán tên của khuôn mặt đã biết với tỷ lệ phần trăm cao nhất

best\_match\_index = face\_distances.argmin()

if percentage >= 50:

name = known\_names[best\_match\_index]

else:

name = f"Maybe + {known\_names[best\_match\_index]}"

# Draw a rectangle around the detected face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

# Display the name and percentage of the detected face

cv2.putText(frame, f"{name} ({percentage:.2f}%)", (left, top - 20), cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (0, 255, 0), 2)

# Display the frame with detected faces

cv2.imshow('Face Recognition', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Release the camera and close all OpenCV windows

cap.release()

cv2.destroyAllWindows()