import cv2

import mediapipe as mp

import serial

import time

# Initialize serial connection to ESP32 on COM8 at 115200 baud

try:

ser = serial.Serial("COM8", 115200, timeout=1)

print("[INFO] Serial connected to COM8")

except serial.SerialException as e:

print(f"[ERROR] Could not open COM8: {e}")

ser = None

# Initialize MediaPipe Hands module

mp\_hands = mp.solutions.hands

hands = mp\_hands.Hands(static\_image\_mode=False, max\_num\_hands=1)

def detect\_gesture(hand\_landmarks):

thumb\_tip = hand\_landmarks.landmark[mp\_hands.HandLandmark.THUMB\_TIP]

thumb\_ip = hand\_landmarks.landmark[mp\_hands.HandLandmark.THUMB\_IP]

wrist = hand\_landmarks.landmark[mp\_hands.HandLandmark.WRIST]

# Thumb up: thumb tip above wrist and thumb tip close horizontally to thumb IP joint

if thumb\_tip.y < wrist.y and abs(thumb\_tip.x - thumb\_ip.x) < 0.05:

return "A" # Thumb up

# Thumb down: thumb tip below wrist and thumb tip close horizontally to thumb IP joint

elif thumb\_tip.y > wrist.y and abs(thumb\_tip.x - thumb\_ip.x) < 0.05:

return "B" # Thumb down

else:

return None

# Open webcam

cap = cv2.VideoCapture(0)

print("[INFO] Starting continuous thumb gesture detection and serial sending...")

try:

while cap.isOpened():

ret, frame = cap.read()

if not ret:

print("[ERROR] Failed to read from webcam")

break

frame = cv2.flip(frame, 1) # Mirror image for natural interaction

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

results = hands.process(rgb\_frame)

gesture = None

if results.multi\_hand\_landmarks:

for hand\_landmarks in results.multi\_hand\_landmarks:

gesture = detect\_gesture(hand\_landmarks)

if gesture:

break # Only consider first detected hand

if gesture:

gesture\_name = "Thumbs Up" if gesture == "A" else "Thumbs Down"

print(f"[DETECTED] {gesture\_name} ({gesture})")

if ser:

try:

ser.write(gesture.encode())

print(f"[SENT] Sent '{gesture}' successfully")

except Exception as e:

print(f"[ERROR] Failed to send '{gesture}': {e}")

else:

print("[WARN] Serial not connected; cannot send data")

time.sleep(0.5) # Wait 0.5s before next send (adjust if needed)

else:

# No gesture detected, short delay to reduce CPU usage

time.sleep(0.1)

cv2.imshow("Thumb Gesture Detection", frame)

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

break

except KeyboardInterrupt:

print("[INFO] Exiting on user interrupt")

finally:

cap.release()

if ser:

ser.close()

print("[INFO] Program terminated")