**HAND GESTURE ESTIMATION**

import cv2

import mediapipe as mp

import numpy as np

# Initialize MediaPipe Hands

mp\_hands = mp.solutions.hands

mp\_drawing = mp.solutions.drawing\_utils

mp\_drawing\_styles = mp.solutions.drawing\_styles

# Gesture recognition function

def recognize\_gesture(hand\_landmarks):

    """

    Recognize hand gestures based on landmark positions

    """

    # Fingertip landmarks

    fingertips = [

        mp\_hands.HandLandmark.THUMB\_TIP,

        mp\_hands.HandLandmark.INDEX\_FINGER\_TIP,

        mp\_hands.HandLandmark.MIDDLE\_FINGER\_TIP,

        mp\_hands.HandLandmark.RING\_FINGER\_TIP,

        mp\_hands.HandLandmark.PINKY\_TIP

    ]

    # Get y-coordinates of fingertips

    tips\_y = [hand\_landmarks.landmark[tip].y for tip in fingertips]

    # Thumb x-coordinate

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

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

    # Index finger base and tip

    index\_tip\_y = hand\_landmarks.landmark[mp\_hands.HandLandmark.INDEX\_FINGER\_TIP].y

    index\_base\_y = hand\_landmarks.landmark[mp\_hands.HandLandmark.INDEX\_FINGER\_MCP].y

    # Check different gesture patterns

    if all(tips\_y[i] < tips\_y[i+1] for i in range(len(tips\_y)-1)):

        return "THUMBS UP"

    if all(tips\_y[1:]):  # All fingertips down

        return "CLOSED FIST"

    # Peace sign (index and middle finger extended)

    if (tips\_y[0] > tips\_y[1] and  # Thumb down

        tips\_y[1] < tips\_y[2] and  # Index up

        tips\_y[2] < tips\_y[3]):    # Middle up

        return "PEACE"

    return "UNKNOWN"

# Main hand tracking and gesture recognition

def main():

    # Open webcam

    cap = cv2.VideoCapture(0)

    # Initialize MediaPipe Hands

    with mp\_hands.Hands(

        model\_complexity=0,

        min\_detection\_confidence=0.5,

        min\_tracking\_confidence=0.5) as hands:

        while cap.isOpened():

            # Read frame

            success, frame = cap.read()

            if not success:

                break

            # Flip the frame horizontally for a later selfie-view display

            frame = cv2.flip(frame, 1)

            # Convert the BGR image to RGB

            frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

            # Process the frame and find hands

            results = hands.process(frame\_rgb)

            # Draw hand annotations and recognize gestures

            if results.multi\_hand\_landmarks:

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

                    # Draw hand landmarks

                    mp\_drawing.draw\_landmarks(

                        frame,

                        hand\_landmarks,

                        mp\_hands.HAND\_CONNECTIONS,

                        mp\_drawing\_styles.get\_default\_hand\_landmarks\_style(),

                        mp\_drawing\_styles.get\_default\_hand\_connections\_style())

                    # Recognize and display gesture

                    gesture = recognize\_gesture(hand\_landmarks)

                    cv2.putText(frame, gesture, (10, 50),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 1,

                                (0, 255, 0), 2)

            # Display the frame

            cv2.imshow('Hand Gesture Recognition', frame)

            # Exit on 'q' key press

            if cv2.waitKey(5) & 0xFF == ord('q'):

                break

    # Release resources

    cap.release()

    cv2.destroyAllWindows()

# Run the main function

if \_\_name\_\_ == '\_\_main\_\_':

    main()