import cv2

import mediapipe as mp

import pyautogui

mp\_drawing = mp.solutions.drawing\_utils

mp\_holistic = mp.solutions.holistic

mp\_hands = mp.solutions.hands

# Info and Initializing Webcam a

cap = cv2.VideoCapture(0)

tipIds = [4, 8, 12, 16, 20] # Marking of the objects

game\_started = 1

charac\_pos = [0,1,0]

index\_pos = 1

fixedx = None

fixedy = None

rec = None

with mp\_hands.Hands(min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5) as hands:

    with mp\_holistic.Holistic(min\_detection\_confidence=0.5,min\_tracking\_confidence=0.5) as holistic:

        # Capturing Frames and Processing the Image

        while True: # infinite loop created for infinite frames

            success, frame = cap.read()

            #adjust

            frame = cv2.flip(frame, 1)

            frame = cv2.resize(frame, (440,330))

            height, width, channel = frame.shape

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

            results\_holistic = holistic.process(img)

            results\_hands = hands.process(img)

            img = cv2.cvtColor(img,cv2.COLOR\_RGB2BGR)

            width\_hf = int(width/2)

            height\_hf = int(height/2)

            # Detecting Shoulder/Body Landmarks

            if results\_holistic.pose\_landmarks:

                right\_x = int(results\_holistic.pose\_landmarks.landmark[mp\_holistic.PoseLandmark.LEFT\_SHOULDER].x \* width)-7

                right\_y = int(results\_holistic.pose\_landmarks.landmark[mp\_holistic.PoseLandmark.LEFT\_SHOULDER].y \* height)

                # cv2.circle(img, (right\_x, right\_y), 5, (0, 0, 0), 2)

                left\_x = int(results\_holistic.pose\_landmarks.landmark[mp\_holistic.PoseLandmark.RIGHT\_SHOULDER].x \* width)+7

                left\_y = int(results\_holistic.pose\_landmarks.landmark[mp\_holistic.PoseLandmark.RIGHT\_SHOULDER].y \* height)

                # cv2.circle(img, (left\_x, left\_y), 5, (0, 0, 0), 2)

                # cv2.line(img, (left\_x,left\_y), (right\_x,right\_y), (255, 0, 255), 2)

                mid\_x = left\_x + int(abs(right\_x - left\_x) / 2)

                mid\_y = int(abs(right\_y + left\_y) / 2)

                # cv2.circle(img, (mid\_x, mid\_y), 2, (255, 255, 0), 2)

                if rec != None:

                    # Sideways movement command

                    if right\_x < width\_hf and index\_pos > 0 and charac\_pos[index\_pos-1] == 0:

                        charac\_pos[index\_pos] = 0

                        charac\_pos[index\_pos-1] = 1

                        pyautogui.press('left')

                        index\_pos -= 1

                        print("Left key")

                        print(charac\_pos)

                    if left\_x > width\_hf and index\_pos < 2 and charac\_pos[index\_pos+1] == 0:

                        print("Right key")

                        charac\_pos[index\_pos] = 0

                        charac\_pos[index\_pos+1] = 1

                        pyautogui.press('right')

                        index\_pos += 1

                        print(charac\_pos)

                    # resetting the position of the subway surfer

                    if right\_x > width\_hf and left\_x < width\_hf and index\_pos == 0:

                        charac\_pos[index\_pos] = 0

                        charac\_pos[index\_pos +1] = 1

                        index\_pos += 1

                        pyautogui.press('right')

                        print(charac\_pos)

                        print('left to center')

                    if right\_x > width\_hf and left\_x < width\_hf and index\_pos == 2:

                        charac\_pos[index\_pos] = 0

                        charac\_pos[index\_pos -1] = 1

                        index\_pos -= 1

                        pyautogui.press('left')

                        print('right to center')

                        print(charac\_pos)

            hand\_cor\_list\_right = []

            hand\_cor\_list\_left = []

            hand\_type1 = None

            hand\_type2 = None

            fingers\_right = []

            fingers\_left = []

            # Detection of both hands and extracting both hand landmarks (to start)

            try:

                hand\_type1 = results\_hands.multi\_handedness[0].classification[0].label

                hand\_type2 = results\_hands.multi\_handedness[1].classification[0].label

                for hand\_no, hand\_landmarks in enumerate(results\_hands.multi\_hand\_landmarks):

                    if hand\_no == 0:

                        if hand\_type1 == 'Left':

                            for id, lm in enumerate(hand\_landmarks.landmark):

                                cx, cy = int(lm.x \* width), int(lm.y \* height)

                                # cv2.circle(img, (cx,cy),2, (100,255,100),2)

                                hand\_cor\_list\_left.append([id,cx,cy])

                        elif hand\_type1 == 'Right':

                            for id, lm in enumerate(hand\_landmarks.landmark):

                                cx, cy = int(lm.x \* width), int(lm.y \* height)

                                # cv2.circle(img, (cx,cy),2, (100,255,100),2)

                                hand\_cor\_list\_right.append([id,cx,cy])

                    if hand\_no == 1:

                        if hand\_type2 == 'Left':

                            for id, lm in enumerate(hand\_landmarks.landmark):

                                cx, cy = int(lm.x \* width), int(lm.y \* height)

                                # cv2.circle(img, (cx,cy),2, (100,255,100),2)

                                hand\_cor\_list\_left.append([id,cx,cy])

                        elif hand\_type2 == 'Right':

                            for id, lm in enumerate(hand\_landmarks.landmark):

                                cx, cy = int(lm.x \* width), int(lm.y \* height)

                                # cv2.circle(img, (cx,cy),2, (100,255,100),2)

                                hand\_cor\_list\_right.append([id,cx,cy])

                if hand\_cor\_list\_right != []:

                    # Right Hand Thumb open Detection

                    if hand\_cor\_list\_right[tipIds[0]][1] < hand\_cor\_list\_right[tipIds[0] - 1][1]:

                        fingers\_right.append(1)

                    else:

                        fingers\_right.append(0)

                    # Right Hand 4 Fingers open Detection

                    for id in range(1, 5):

                        if hand\_cor\_list\_right[tipIds[id]][2] < hand\_cor\_list\_right[tipIds[id] - 2][2]:

                            fingers\_right.append(1)

                        else:

                            fingers\_right.append(0)

                    totalFingers\_right = fingers\_right.count(1)

                if hand\_cor\_list\_left != []:

                    # Left hand Thumb open detection

                    if hand\_cor\_list\_left[tipIds[0]][1] > hand\_cor\_list\_left[tipIds[0] - 1][1]:

                        fingers\_left.append(1)

                    else:

                        fingers\_left.append(0)

                    # Left Hand 4 Fingers open Detection

                    for id in range(1, 5):

                        if hand\_cor\_list\_left[tipIds[id]][2] < hand\_cor\_list\_left[tipIds[id] - 2][2]:

                            fingers\_left.append(1)

                        else:

                            fingers\_left.append(0)

                    totalFingers\_left = fingers\_left.count(1)

            except:

                pass

            # Command to Start the game

            if fingers\_right.count(1) == 2 and fingers\_left.count(1) == 2 and fingers\_right[1] == 1 and fingers\_right[2] == 1 and fingers\_left[1] == 1 and fingers\_left[1] == 1:

                fixedx = left\_x + int(abs(right\_x - left\_x) / 2)

                fixedy = int(abs(right\_y + left\_y) / 2)

                rec = 35

                pyautogui.press('space')

            # Up and Down command

            if fixedy is not None:

                if (mid\_y- fixedy) <= -24:

                    pyautogui.press('up')

                    print('jump')

                elif (mid\_y - fixedy) >= 40:

                    pyautogui.press('down')

                    print('down')

            center\_arrow = 10

            cv2.circle(img,(width\_hf,height\_hf),2,(0,255,255),2)

            cv2.line(img,(width\_hf,height\_hf -center\_arrow),(width\_hf,height\_hf+center\_arrow),(0,255,0),2)

            cv2.line(img,(width\_hf -center\_arrow,height\_hf),(width\_hf+center\_arrow,height\_hf),(0,255,0),2)

            # Lines to be crossed to detect up and down movement

            # if rec is not None:

            #     cv2.line(img, (0, fixedy), (width, fixedy), (0, 0, 0), 2)

            #     cv2.line(img, (0, fixedy - 24), (width, fixedy - 24), (0, 0, 0), 2)

            #     cv2.line(img, (0, fixedy + rec), (width, fixedy + rec), (0, 0, 0), 2)

            cv2.imshow('Subway Surfers',img)

            # Updation Of Index Position

            cv2.waitKey(1)