Project Synopsis

Nithish Murugavenkatesh
5/9/2022

Project CHLD2MI

Contour Based Hand Landmark Detection To Mouse Input

Overview

As humans, We have always felt the need to interface the real world with the Virtual Environment. What better way to do that than controlling your cursor using your hands? And the best thing is that it doesn't require any external sensor. Only webcam input. That Is what I have created

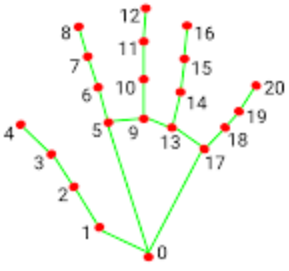
Goals

1. **To Make the Hand Detection module 80% Accurate:**
2. **To translate Real-world XYZ Values to an XY Cartesian plane on which mouse data is derived from Hand Landmark Data:**

Specifications

The Hand Detection and Landmark Allocation:

The Python Program uses a Tensorflow ML (Machine Learning) model that allocates 21 Landmarks to a hand in a webcam input.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

The Landmark that's XY Co-ordinate I require is the HandLandmark.INDEX_FINGER_TIP

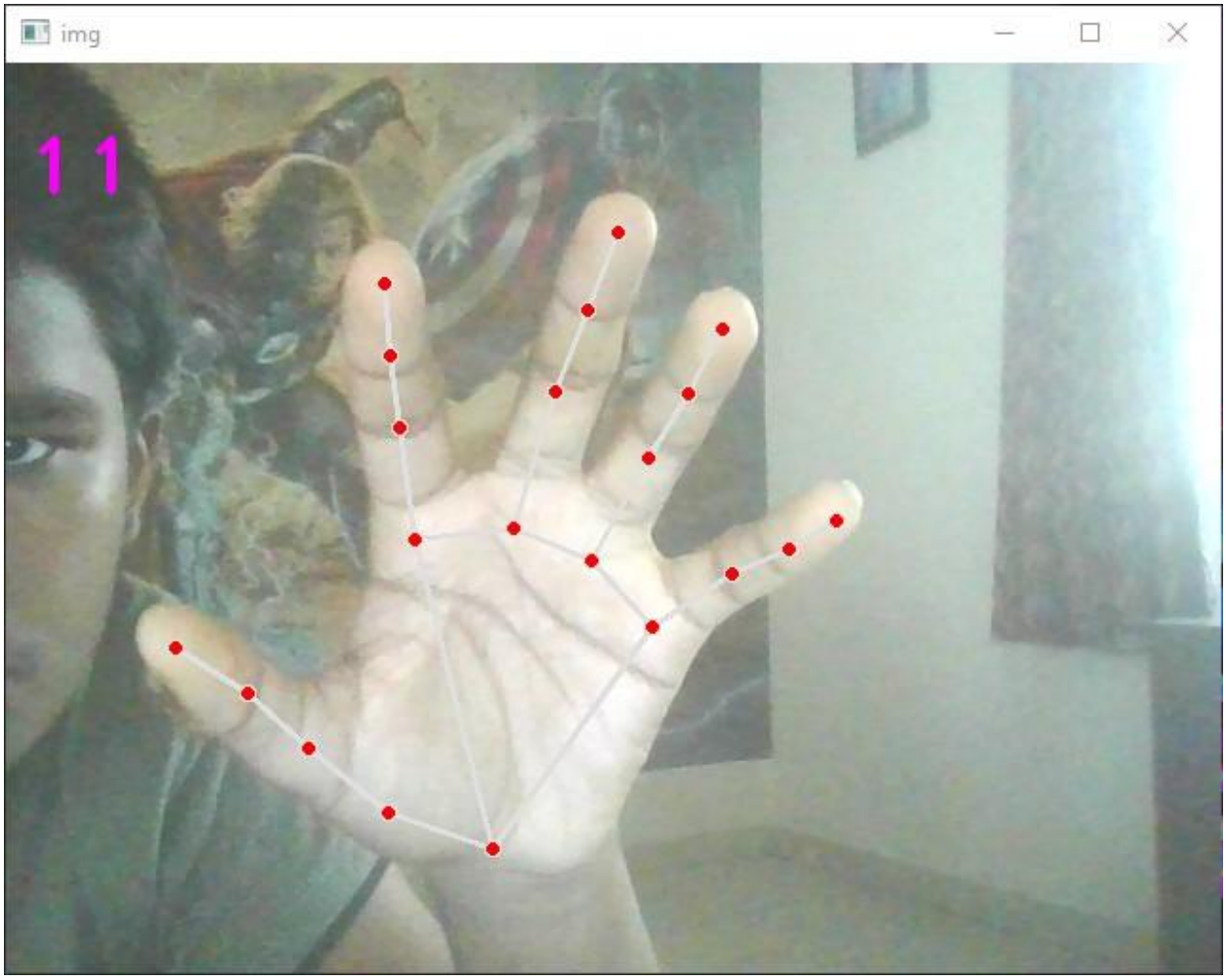
This is then translated from default 3Dimensional Coordinates to normalized Pixel Coordinates.

Transferring Normalized Pixel Coordinates to 1366x768 Coordinates

The Normal Screen Reso in OpenCV is 640x480. To upgrade it to 1366x768, First, I Cross-Divide them and return their values as X and Y.

Then I multiply the respective Normalized pixel Coordinates by the X and Y values respectively.

Finally, It inputted towards a sys Module known as mouse that moves the cursor in the respective XY Values



Milestones

1. Successfully implemented said Module!

2. Feels Flexible and Has many UseCase Scenarios

Videos/Photos:

https://drive.google.com/file/d/1XTbjcqlulVuDEffKPI9k_0fHpro3Xus7/view?usp=sharing

```
import cv2
import mediapipe as mp
import time
import mouse
cap = cv2.VideoCapture(0)

mpHands = mp.solutions.hands
hands = mpHands.Hands(
                min_detection_confidence=0.8,
                min_tracking_confidence=0.5)
mpDraw = mp.solutions.drawing_utils

pTime = 0
cTime = 0
while True:

    success, img = cap.read()
    img = cv2.flip(img, 1)
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    mp_drawing = mp.solutions.drawing_utils
    mp_hands = mp.solutions.hands
```

```
#print(results.multi_hand_landmarks)
if results.multi_hand_landmarks:
    for handLms in results.multi_hand_landmarks:
            for id, lm in enumerate(handLms.landmark):
        h, w, c = img.shape
        cx, cy = int(lm.x *w), int(lm.y*h)
        cv2.circle(img, (cx,cy), 3, (255,0,255), cv2.FILLED)
      mpDraw.draw_landmarks(img, handLms, mpHands.HAND_CONNECTIONS)
if results.multi_hand_landmarks != None:
    imageHeight, imageWidth, _ = img.shape
    for handLandmarks in results.multi_hand_landmarks:
      for point in mp_hands.HandLandmark:
        normalizedLandmark = handLandmarks.landmark[point]
        pixelCoordinatesLandmark = mp_drawing._normalized_to_pixel_coordinates(normalizedLandmark.x,
                                                    normalizedLandmark.y, imageWidth,
                                                    imageHeight)
        point = str(point)
        if point == 'HandLandmark.INDEX_FINGER_TIP':
          try:
            indexfingertip_x = pixelCoordinatesLandmark[0]
            indexfingertip_y = pixelCoordinatesLandmark[1]

            x, y = 1366 / 480, 768 / 640
            screen_pos = [indexfingertip_x * x, indexfingertip_y * y]
```

```
                mouse.move(screen_pos[0], screen_pos[1])
            except:
                pass
    cTime = time.time()
    fps = 1/(cTime-pTime)
    pTime = cTime
    cv2.putText(img,str(int(fps)), (10,70), cv2.FONT_HERSHEY_PLAIN, 3, (255,0,255), 3)


    cv2.imshow("img", img)
    cv2.waitKey(1)
```