

HAND LANDMARK DETECTION AND TRACKING WITH MEDIAPIPE

Before of starting to explain the code, let's talk about MediaPipe, this library was crated for Goggle for the recognizer face and tracking of objects in images and videos, some of the applications more important of this library is the detection of landmarks in hands and tracking of the same, without doubt a tool of great help for projects in persons with disabilities auditive and of the speech.

For this project, we will identify the landmarks and tracking of hands in real time with the purpose of knowing the basic function of this tool, and we will have the knowledge to create different projects related to this in the future.

We will use two libraries, the first OpenCV that will be in charge of the video in real time and MediaPipe, that will be in charge of the related with the hands.

We import these two libraries, and we will be ready to start.

```
import cv2
import mediapipe as mp
```

We will initialize the `.solutions` module of the library MediaPipe, with the `.hands` function, what we will do in this initialization is bring the necessary functions to the code, the `.hands` function is that we will help us with the detection and tracking of the hands.

```
mp_hands = mp.solutions.hands
```

Now we will configure the parameters for the detection, to do this we will use the `.Hands()` function, there is not that confuse this function with the initialization function because they are not the same, `.hands` only does the initialization and `.Hands()` configures the parameters.

The parameters that we will use to `.Hands()` are:

- `static_image_mode`: this boolean parameter is responsible for differentiating between an image and a video, if it is True (value by default), it will be taken as we are going to

use an image to the detection and in case of it is False, it will be taken as we are going to use a video with motion.

- `max_min_hands`: this parameter has two possible variables, 1 and 2; if we use 1 it will detect only one hand, opposite case if we will use 2 it will detect two hands.
- `min_detection_confidence` and `min_tracking_confidence`: these two float parameters are responsible for setting the percentage of confidence for the detection and tracking, respectively.
- `model_complexity`: here we set the complexity of the model, if it is 0 it will be faster, but less accurate and if it is 2 it will be more accurate, but also slower.

```
hands = mp_hands.Hands(static_image_mode=True,
                        max_num_hands=2, min_detection_confidence=0.5,
                        min_tracking_confidence=0.5, model_complexity=0)
```

Once this is done, we will initialize the `.drawing_utils` function to draw the landmarks in the hands.

```
mp_drawing = mp.solutions.drawing_utils
```

After initializing both the hand detector and the drawing functions, we will initialize our video capture using the `.videoCapture()` function, and thus start to work in the algorithm completely.

```
Cap = cv2.VideoCapture(0)
```

We will pass to create our while loop in which we will do the configurations necessary to the correct work of the library MediaPipe, first we will read our frames through of the `.read()` function. Having this we will do a mirror effect with the `.flip()` function, this is because the camera will go according to our movements and not the other way around, this function will be use in case of the camera that we are using to need it. In case of using `.flip()` we will give it as parameter the variable on which we will apply the effect and as second parameter the flip Code that is an integer number corresponding to the effect that we need, these are the different values that it can have:

- 0: flip the image horizontally.
- 1: flip the image vertically.
- -1: flip both horizontally and vertically.

Finally, we will configure the change of the color scale from BGR to RGB, this is because the tools of MediaPipe work on the RGB scale, this change we will do through the `.cvtColor()` function, the parameters that we will use are:

- The variable to which we will apply the change.
- The code of the modification, in this case it will be `cv2.COLOR_BGR2RGB`.

The part of the code with the realized configurations will be written of the next way:

```
while True:
    Ret, Frames = Cap.read()
    Frames = cv2.flip(Frames, 1)
    Frames_rgb = cv2.cvtColor(Frames, cv2.COLOR_BGR2RGB)
```

Now we will store the data of the frames in a variable, using the `.process()` function, and as parameter the variable that contains the change from BGR to RGB.

This function is in charge of storing all the information of the frames for later return it and use it in an adequate way in the code.

```
Results = hands.process(Frames_rgb)
```

We will create the configurations corresponding to the circles and lines that will be displayed in the hands when it will be detected, to both configurates we will use the same function that is `.DrawingSpec()`. First, we will start with the configurations of the circles, to which we will give it three parameters:

- The color that in which to display the circles (BGR scale).
- The thickness of the circles.
- The radius of the circles.

And for the lines, we will use two:

- The color of the lines (BGR scale).
- The thickness of the lines.

```
Circles_color=mp_drawing.DrawingSpec(color=(255,0,0),thickness=4, circle_radius=2)
Lines_color=mp_drawing.DrawingSpec(color=(0,0,255),thickness=3)
```

Now, we will pass to create an **if** conditional, in this conditional we will verify that our **Result** variable, that contains the detection data, together with the **.multi_hands_landmarks** function, that contains the three dimensional coordinates of the hand, it has information that returns us, once we have this conditional and having data that deliver, we will pass to create a **for** loop with which we will draw the landmarks or landmarks in the hands, we will use a variable with the name **Landmarks** that will be iterating in **Results.multi_hands_landmarks**.


Inside of this `for` loop we will use the `mp_drawing` variable, which will contain the drawing tools previously initialized, together with the `.draw_landmarks()` function, to which we will give it the following parameters:

- The variable in which we will draw the landmarks.
- The variable containing the list of detected landmarks, in this case it will be `Landmarks`.
- We will use `mp_hands.HAND_CONNECTIONS`, that is the list of connections between the landmarks, this will be the lines displayed in the hands.
- The variable with the configuration of the color and shape of the circles previously configured.
- The variable with the configuration of the color and shape of the lines previously configured.

Having all this, the explained lines of code would look as follows:


[illegible]

Once this is done, we could display the video capture with the frames and the hands with the landmarks and connecting lines, to do this we will use the `.show()` function, and as parameters we will give the name that the window will have and the function that we will display.




```
cv2.imshow("LANDMARKS", Frames)
```

We indicate that, with the letter “q” or “Q”, we will leave from the program this through the `.waitKey(1)` function and an if conditional.



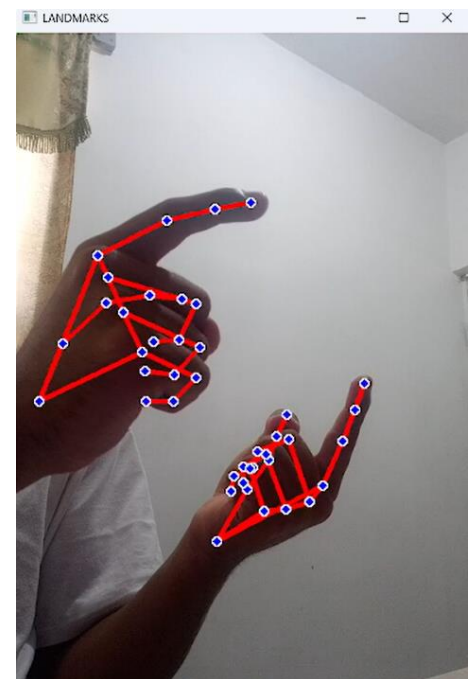
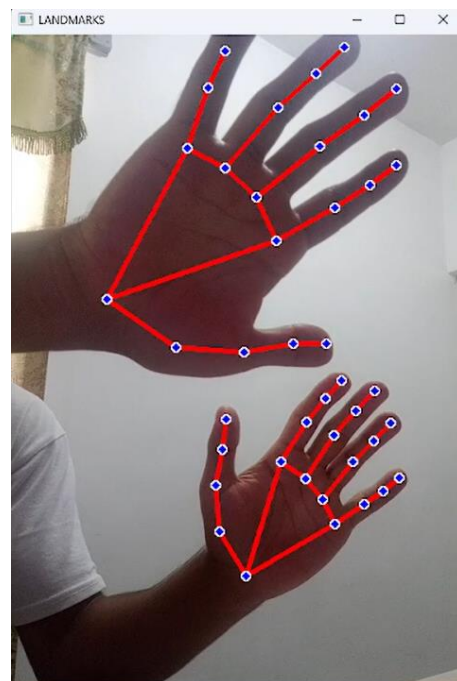
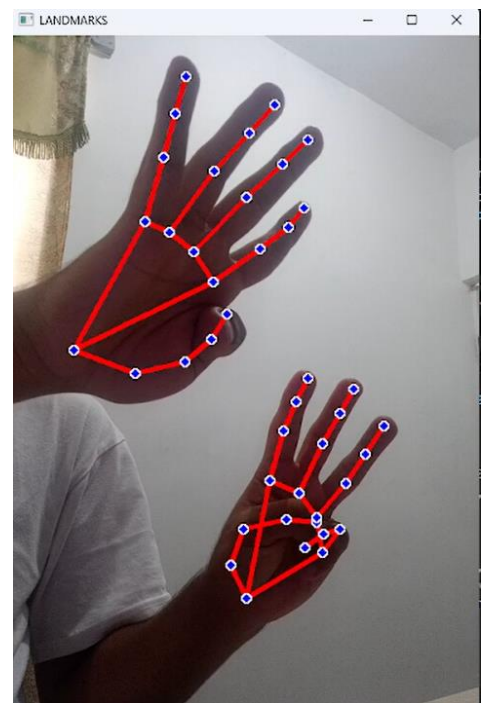
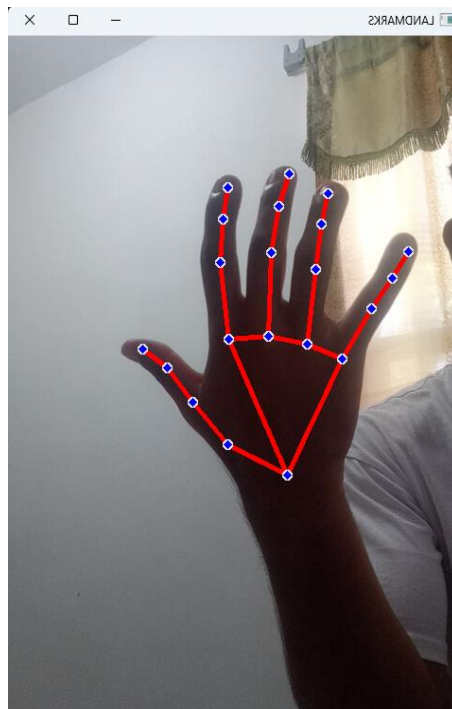
```
t = cv2.waitKey(1)
if t == ord('q') or t == ord("Q"):
    break
```

Finally, we will release our video capture with the `.release()` function and our window with `.destroyAllWindows()`.



```
Cap.release()
cv2.destroyAllWindows()
```

RESULTS



If you got to this part, I thank you for the time that you gave me and I hope that I have helped to you with your doubts and concerns of learning more, this is just a small fragment of explain if you want to continue learning and improve you can visit these pages that will help you in the topic of artificial vision and Python.

- 1.- <https://omes-va.com/>
- 2.- <https://www.pythonpool.com/>
- 3.- <https://hetpro-store.com/TUTORIALES/>
- 4.- <https://learnopencv.com/>

Wisdom is the principal thing; therefore get wisdom: yea, with all thou hast gotten get understanding. Proverbs 4:7