Makerspace Documentation

-By Nikhil Giri

Abstract

Makerspace is a novel initiative taken by the Technology Robotix Society, IIT Kharagpur to promote the culture of working on robotics-related projects among the students of IIT Kharagpur. This Documentation consists of my works for the Makerspace 2021, as part of the Motion Capture and transfer using CV.
I have done 2 Projects with different types of game control using Hand Detection via OpenCV and Mediapipe.

- *Snakes Game (Made in Pygame) control via detection of the direction in which the hand is pointing.*
- *Online Multiplayer game (SmashKarts) control via detection of whether certain fingers are opened or closed.*

Table of Contents

List of figures

Introduction to Problem Statement

The problem statement was to build a system that would capture the motion of the user and create a 3D model imitating the motion which can be used later for various purposes such as gaming and creating virtual avatars.

Software requirements

The project requires running complex image processing along with game and keyboard control. So, a Discrete GPU is recommended otherwise the camera output might feel laggy and give a bad user experience. Also, a webcam is required.

Along with these software requirements are:

- Python (3.7 or greater)
- OpenCV- Python
- Mediapipe (An open-source framework by Google)
- Pygame
- Pynput

Also, some IDE such as Pycharm or VScode is recommended.

Demonstration Images

Figure 1: Hand Detection and Highlighting the 21 Hand landmarks

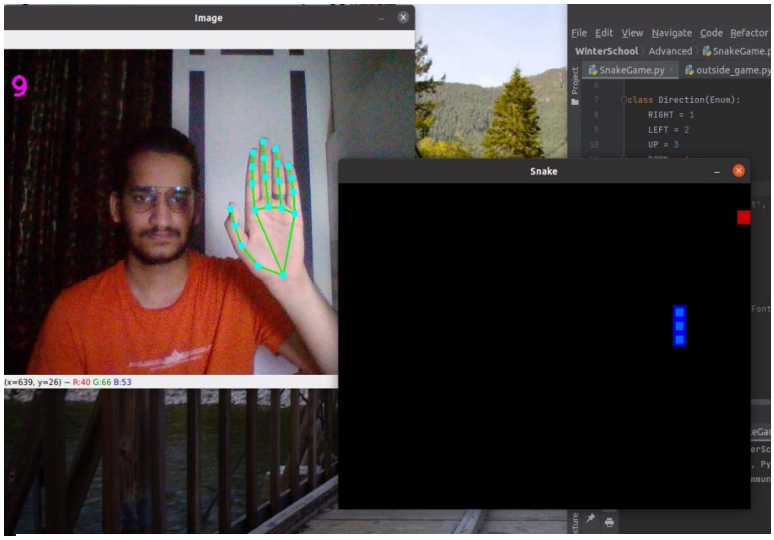


Figure 2: User Input via the direction the Hand is pointing--> Up

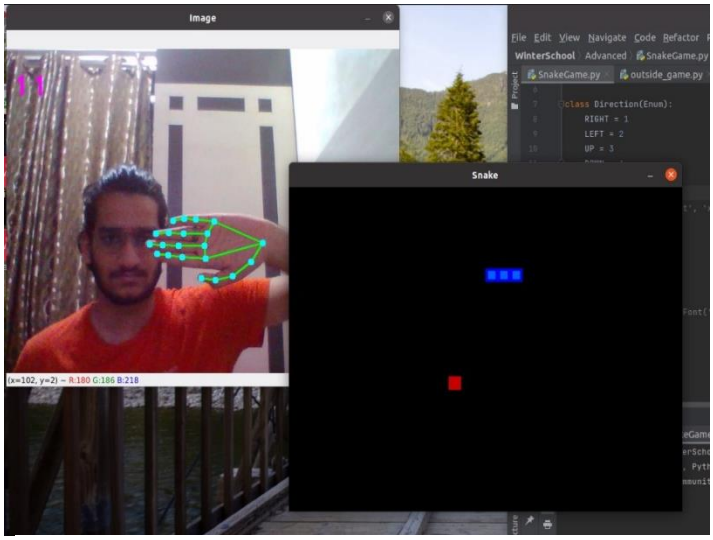


Figure 3: User Input via the direction the Hand is pointing--> Left

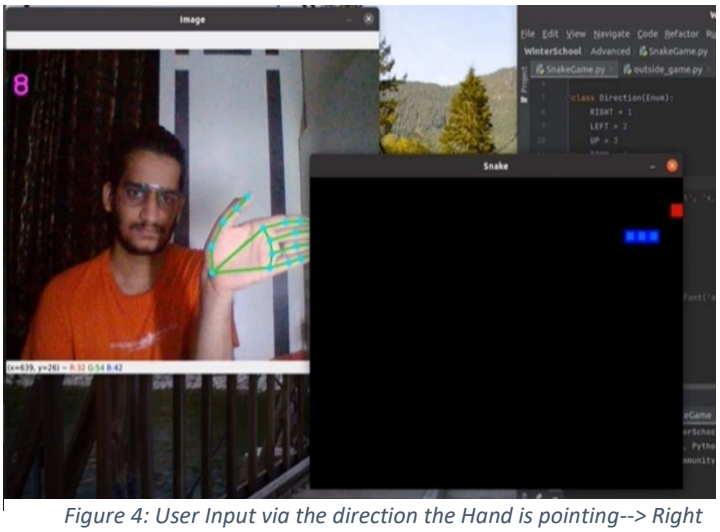


Figure 4: User Input via the direction the Hand is pointing--> Right

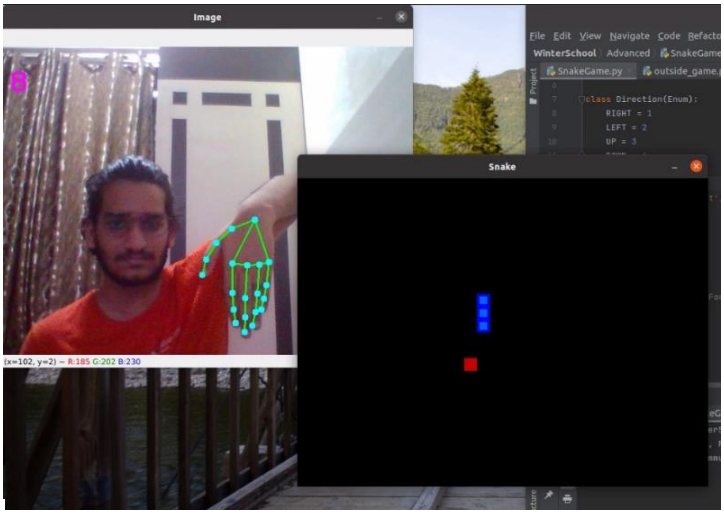


Figure 5: User Input via the direction the Hand is pointing--> Down

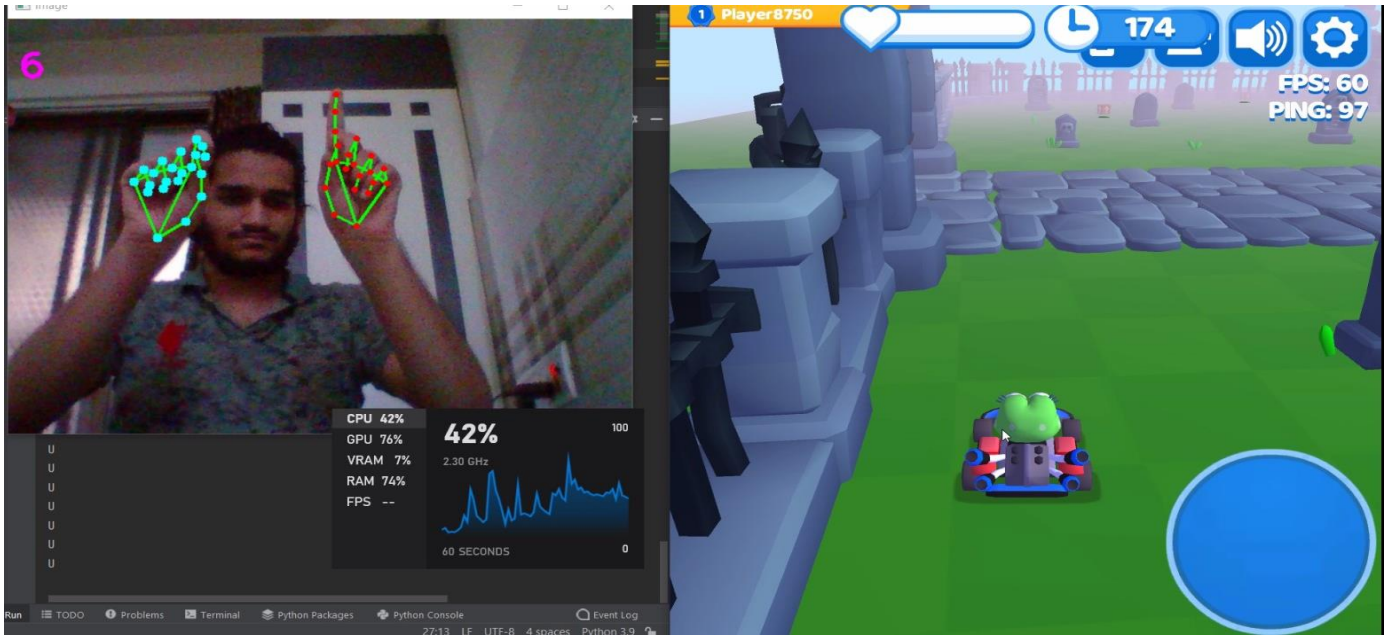


Figure 6: User Input via Checking if Certain Finger is Open or Close--> Accelerate (Right Index open)

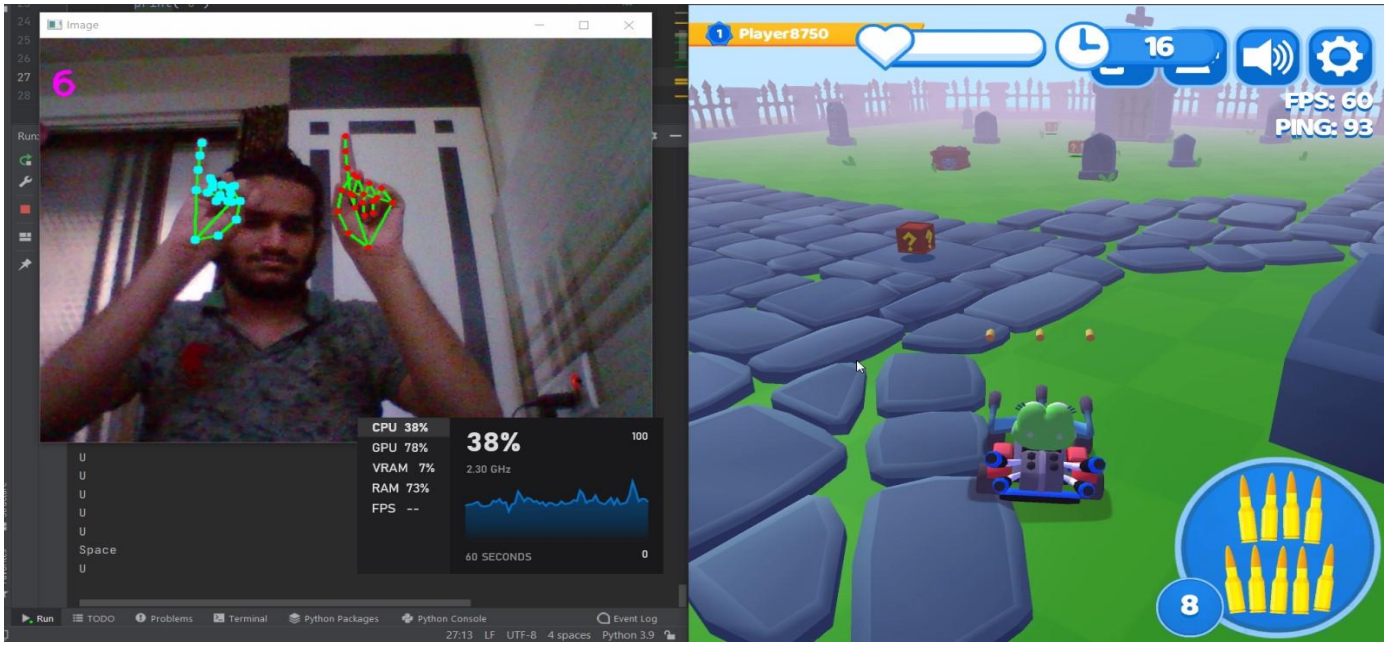


Figure 7: User Input via Checking if Certain Finger is Open or Close--> Power up (Left Little finger open)

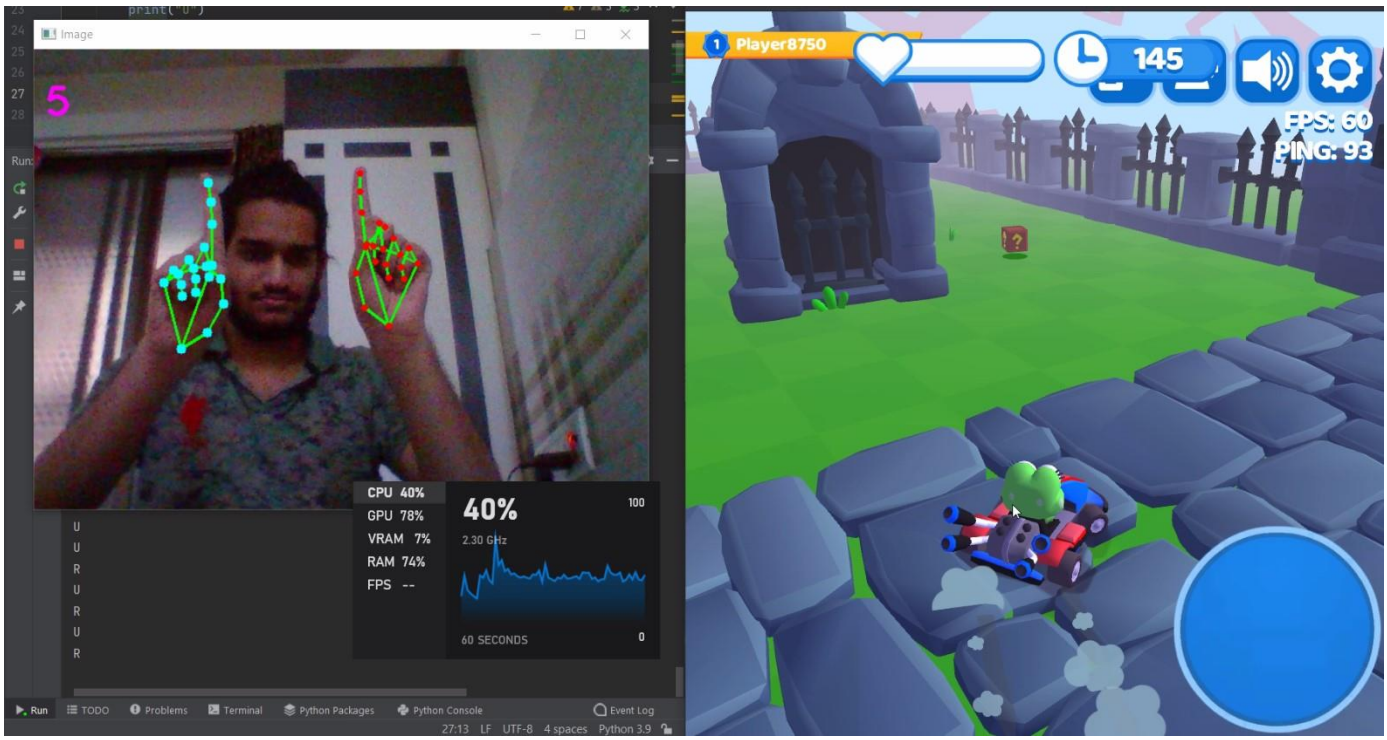
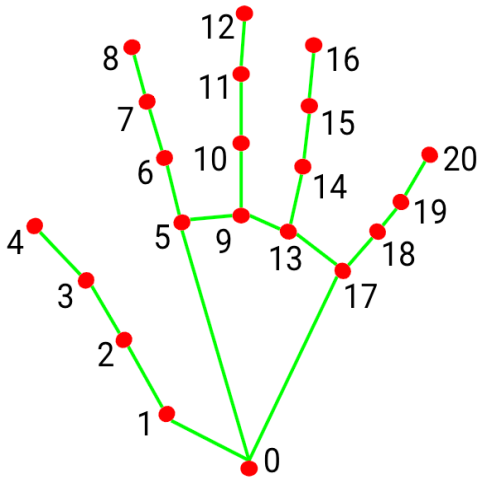


Figure 8: User Input via Checking if Certain Finger is Open or Close--> Right turn (Right and Left Index open)

Theory

The programs work such that it takes input from the webcam using OpenCV and then process it using the Mediapipe package to detect hands in the frame and then identify or predict the position of all 21 landmarks for each hand. This then returns the position of these *landmarks for each hand detected in a 3D coordinate system. Using this X, Y, and Z positions we try to estimate different poses of the hand and if these estimation and detection are equal that means that the hand is at the estimated position* and we can carry out an action. This pose can be of any manner from the angle between fingers to the direction of the finger to either the fingers are open or closed. *Using these poses we can give the user option to give different inputs to the program and thus play a game or carry out different functions of the program.*



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

Figure 9: Hand Landmarks

At this stage, we have the main task of detecting user inputs and we now want to carry out these actions into a game. We can either
(a) *Create a game using python or any programming language.*
(b) *We can also play a commercially available game.*

(A)Now, ideally in a modern high-performance machine if we make a game both tasks i.e. image processing and game loops should work in a single python file but as some of the users might have some restrictions we introduce the third component into the game that is simulating the keyboard presses via Pynput package. We make a script containing our image processing code and instead of controlling the game we make the required keyboard presses. Along we this script we also run our game implementation (Pygame in this

project). This reduces the load on the processor and we can smoothly play the game.

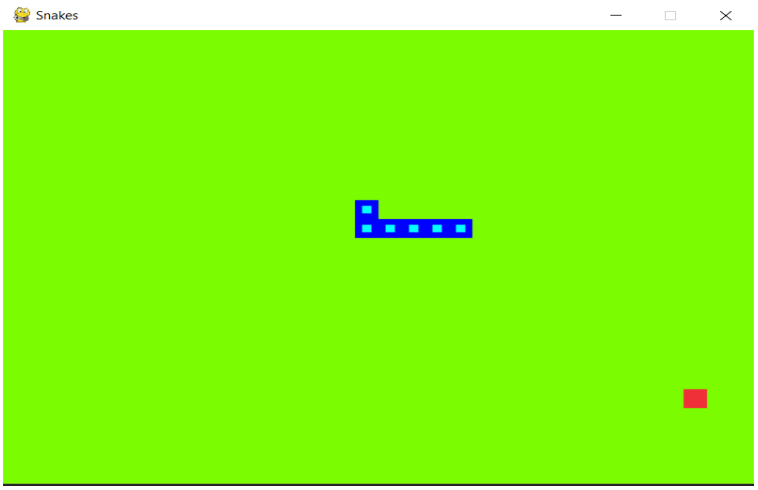


Figure 10: Snakes Game (Created in pygame)

(B)When we are trying to control a game not made by us, we don't have the access to the source code to make the changes required to play the game via Hand Detection. So, we use the same method as we used in part (A) i.e. we run the game (SmashKarts) and simultaneously run our image processing script to simulate the key presses as per the inputs given by the user. In these cases, the program had to be very efficient as the game was to be played at the same speed as normally played by any user.

Figure 11: SmashKarts Game Interface

Along with these two variations, I also explored different types of detection methods. As mentioned before that I used hand detection to control games. Even while implementing hand detection, I have used different approaches. They are: -

- *Detection of the direction the hand is pointing to determine the input by the user (For snakes)*

  In this method, I created a function in my Hand tracking module called ***direction_hand*** which takes on certain parameters. In these parameters 'thres' is the only one that might need to be tweaked a little if the user got

smaller hands than average. It checks the distance between two hand landmarks and compares the vertical or horizontal distance between them to thres(hold) parameter and returns the direction assuming ind_2 to be the base and ind_1 to be the pointer.

```
#### The function to detect the direction the hand is pointing to ####
def direction_hand(self, img, thres, ind_1, ind_2, flag):
    img = self.findHands(img)
    lmList = self.findPosition(img)
    if len(lmList) != 0:
        x = (lmList[ind_1][1] - lmList[ind_2][1])
        y = (lmList[ind_1][2] - lmList[ind_2][2])
        # print(x, y)

        if y > thres:
            return 'd'

        elif y < -thres:
            return 'u'

        elif x > thres:
            return 'l'

        elif x < -thres:
            return 'r'

    return flag
```

Figure 12: The direction_hand function in HandDetectionModule

- *Detecting that if a certain finger is open or close to determine the user input (For smashkarts)*

  In this method, I created a function in my Hand tracking module called ***left_or_right*** and ***num_fingers*** which takes on certain parameters. The left_or_right function determines which detected hand is left or right and returns the hand number (0,1 if valid; 2 if invalid). Then in the num_fingers we check if a specific finger of a specific hand is open or closed. The finger and hand are determined by the parameter of the function. You can play with it and test different fingers. I used just a few fingers which seemed suitable to me to control the game. You can easily simplify or complexify the inputs as per your needs.

```
#### The function to detect and return the hand no for left and right hand ####
def left_or_right(self, img):
    img = self.findHands(img, draw=False)
    lmList = self.findPosition(img, draw=False)
    if len(lmList) != 0:
        num_hands = len(self.results.multi_handedness)
        if num_hands != 2:
            # print(num_hands)
            # print("I got removed Here")
            return False, 2, 2
        for id, classification in enumerate(self.results.multi_handedness):
            # print("I got here !!!!!!!!!!")
            if classification.classification[id].label == "Left":
                l = classification.classification[id].index
                return True, l, 1 - l
            elif classification.classification[id].label == "Right":
                r = classification.classification[id].index
                return True, 1 - r, r
            else:
                print("What the hell just happened....heh?")
                return False, 2, 2
    return False, 2, 2
```

Figure 13: The left_or_right function

```
#### The function to detect whether a finger is open or closed ####
# tip_num => 8, 12, 16, 20
def num_fingers(self, img, hand_flag, tip_num):

    flag, left, right = self.left_or_right(img)
    if flag:
        if hand_flag == "l":
            hand_index = left
        elif hand_flag == "r":
            hand_index = right
        else:
            print("Please check the hand_flag Parameter")
            return False
        lmList = self.findPosition(img, draw=False, handNo=hand_index)

        if lmList[tip_num][2] < lmList[tip_num - 2][2]:
            return True

    return False
```

Figure 14 : The num_fingers function

Conclusion

The entire project was very fun to do as I got to explore a lot of new concepts and learn lots of new things. Along with these, it was interesting to come up with my own solutions for various problems and debug the code. Though the image processing requires great hardware to work seamlessly it is a great alternative to standard gameplay and fun to interact with.

References

- https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html
- https://google.github.io/mediapipe/solutions/solutions.html
- https://pynput.readthedocs.io/en/latest/
- https://www.pygame.org/docs/
- GitHub Repository containing project files →https://github.com/NikhilGiri29/Makerspace--Motion-Capture-using-OpenCV
- Snakes Game Youtube video → https://youtu.be/6TaWWQhuGfI
- SmashKart Game Youtube video → https://youtu.be/mYAa3zG2blQ