

Practical-5

Student Name: Gauri Prabhakar

UID: 18BCS6201

Branch: 18AITAIML-2

Section/Group: B

Semester: 7

Date of Performance: 5th October, 2021

Subject Name: Computer Vision Lab

Subject Code: CSF - 432

1. Aim/Overview of the practical:

To implement handtracking using mediapipe in python and OpenCV.

2. Task to be done:

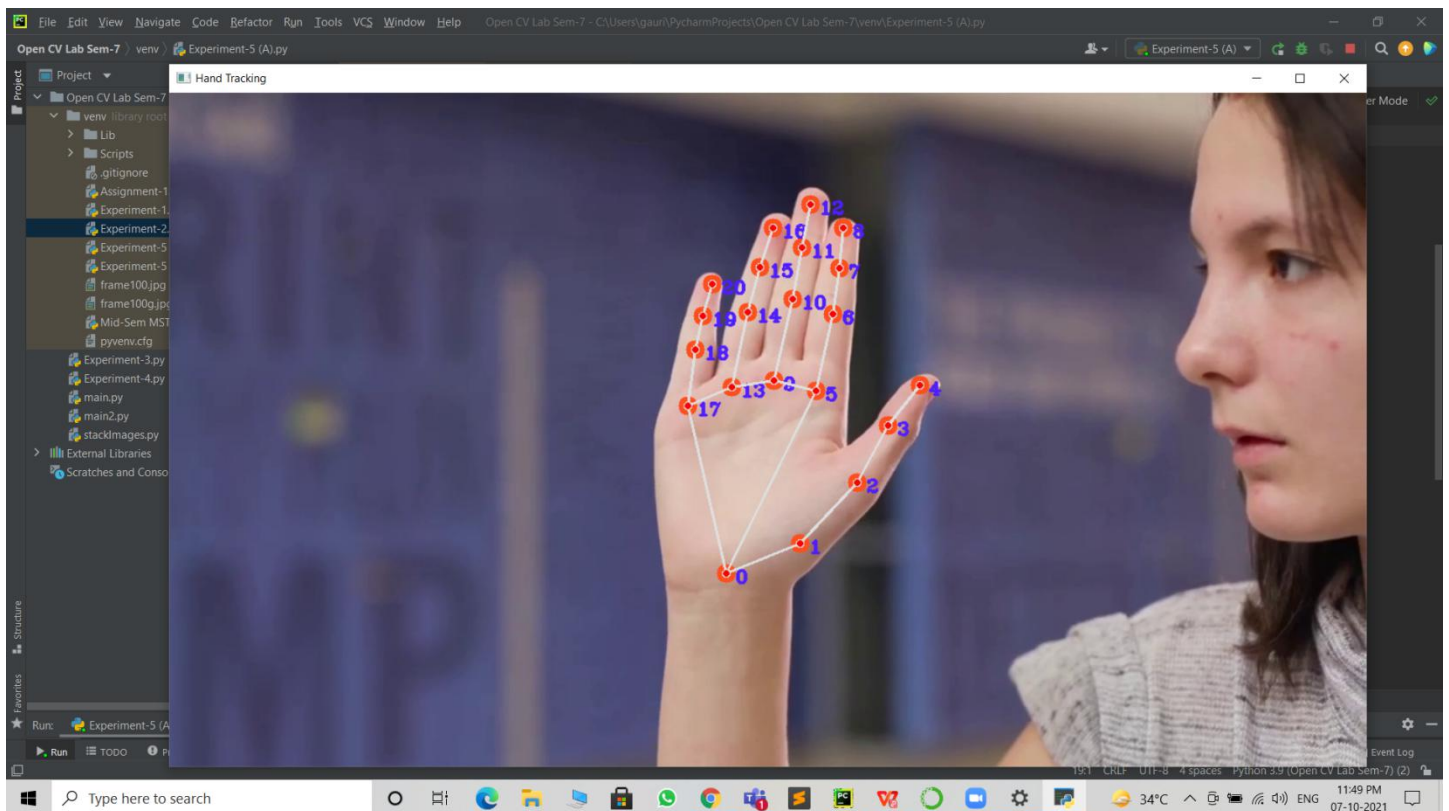
To implement handtracking using mediapipe in python and OpenCV and the explanation.

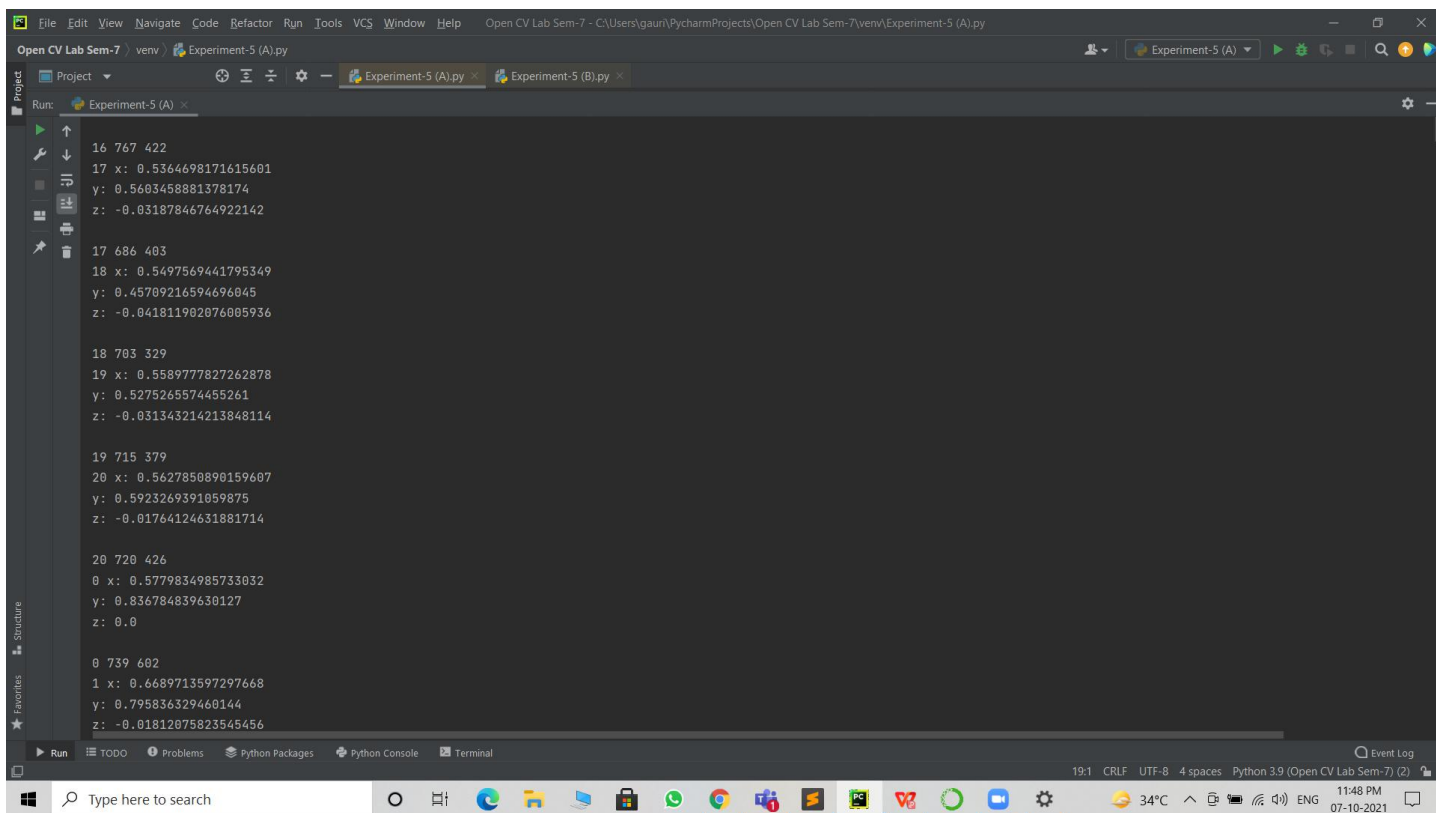
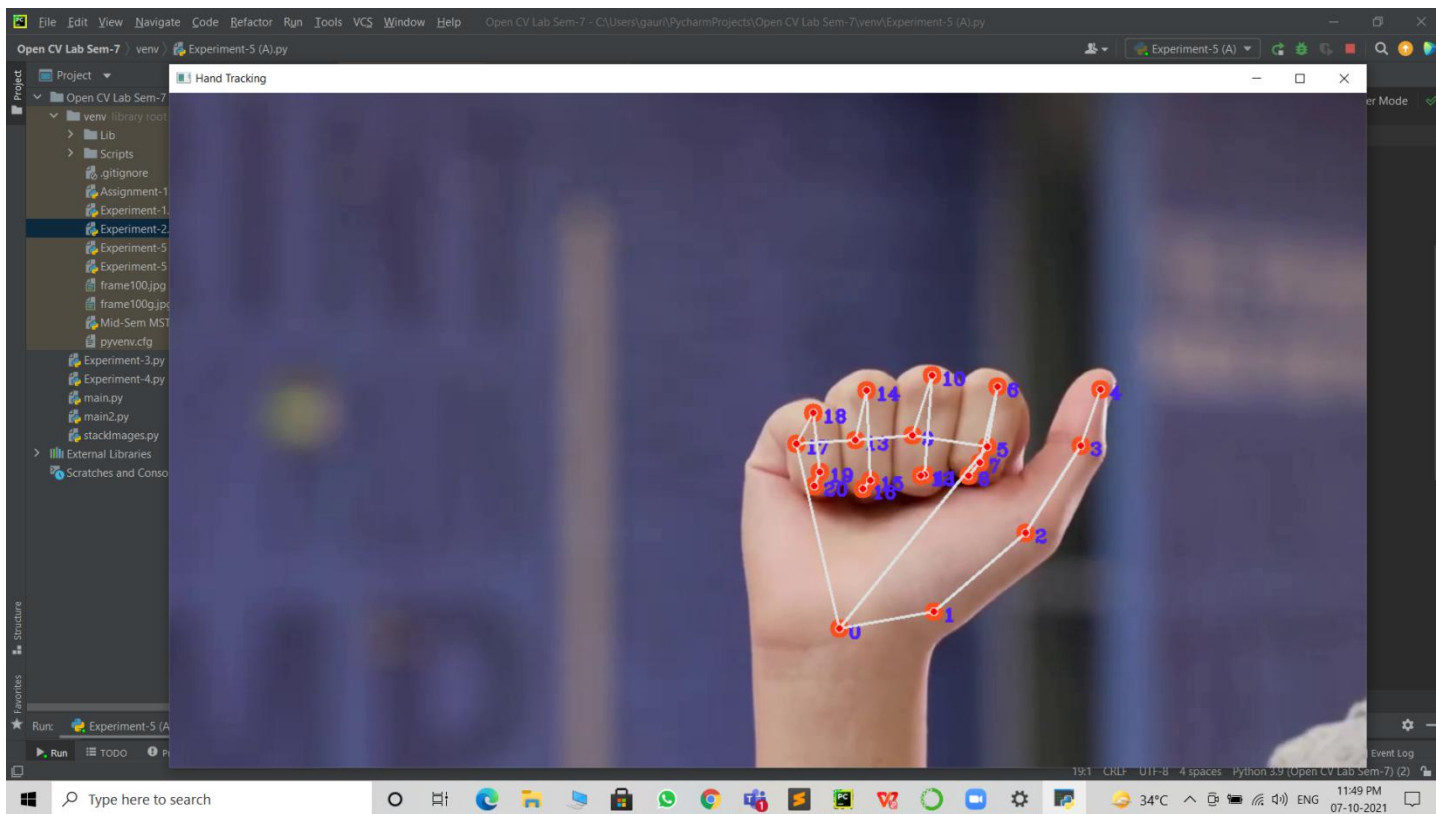
3. Steps to be followed:

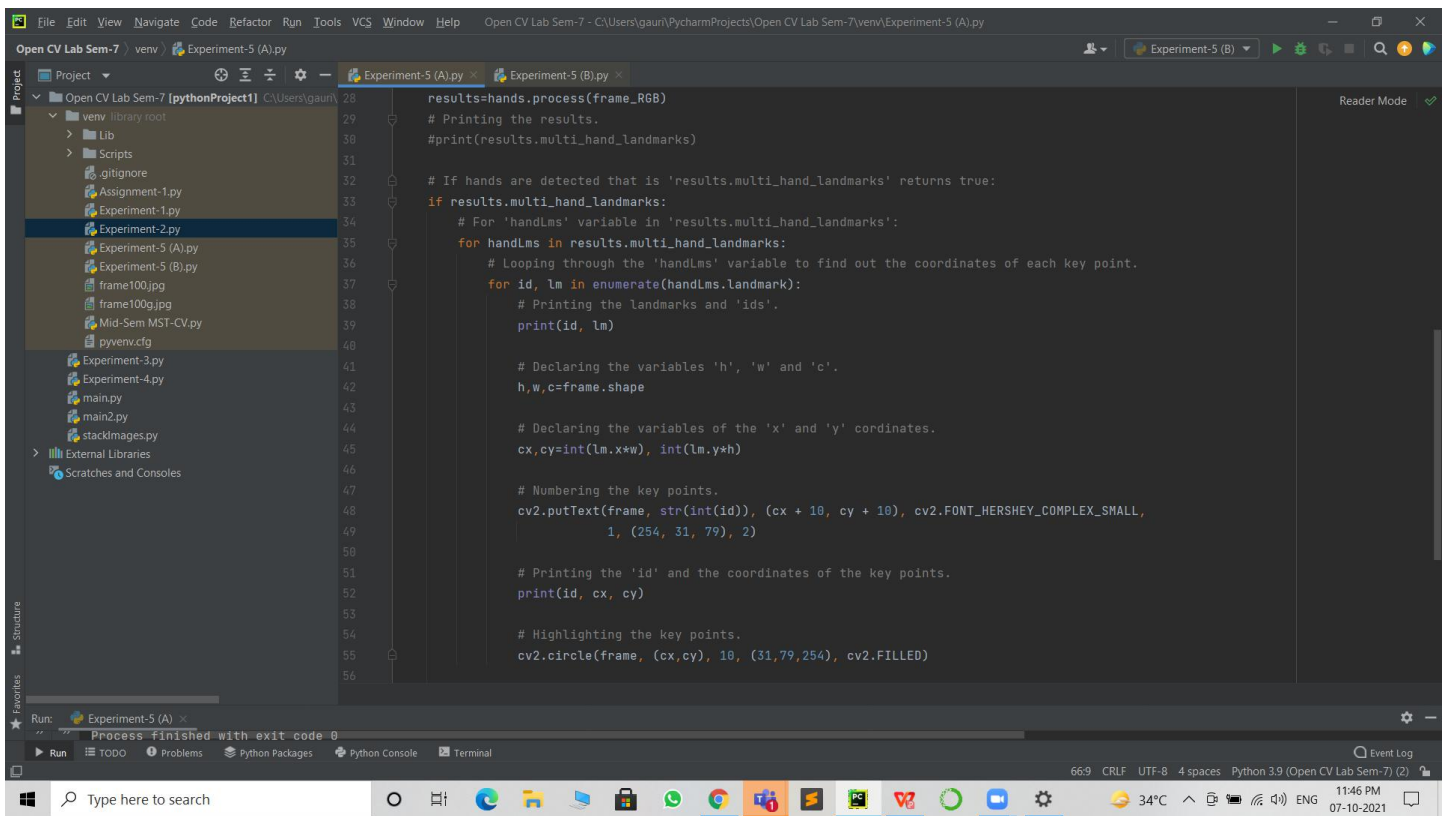
1. Importing necessary modules.
2. Creating a variable to store the video using the '.VideoCapture()' function.
3. Importing 'mp.solutions.hands' to a variable 'mpsh'.
4. Declaring the object 'hands' from 'mp.solutions'.
5. Importing 'mp.solutions.hands' to a variable 'mpsdu'.
6. We will use 'drawing_utils' to draw the key points.
7. Capturing the video frame by frame using the '.read()' method.
8. Reading the frames and converting them to RGB.
9. Detecting hands in the frame using the function 'hands.process()'.
10. Printing the results.
11. If hands are detected that is 'results.multi_hand_landmarks' returns true:
12. For 'handLms' variable in 'results.multi_hand_landmarks':
13. Looping through the 'handLms' variable to find out the coordinates of each key point.

14. Printing the landmarks and 'ids'.
15. Declaring the variables 'h', 'w' and 'c'.
16. Declaring the variables of the 'x' and 'y' coordinates.
17. Numbering the key points.
18. Printing the 'id' and the coordinates of the key points.
19. Highlighting the key points.
20. Connecting the key points using the function 'mpsdu.draw_landmarks()'.
`mpsdu.draw_landmarks()`
21. Rendering the video with effective Handtracking to the console by using the function '`.imshow()`'.
22. Setting up '`.waitkey()`' to wait for a specific time until any key is pressed and break the loop.
23. Implementing Hand Tracking using Live footage from the Webcam.

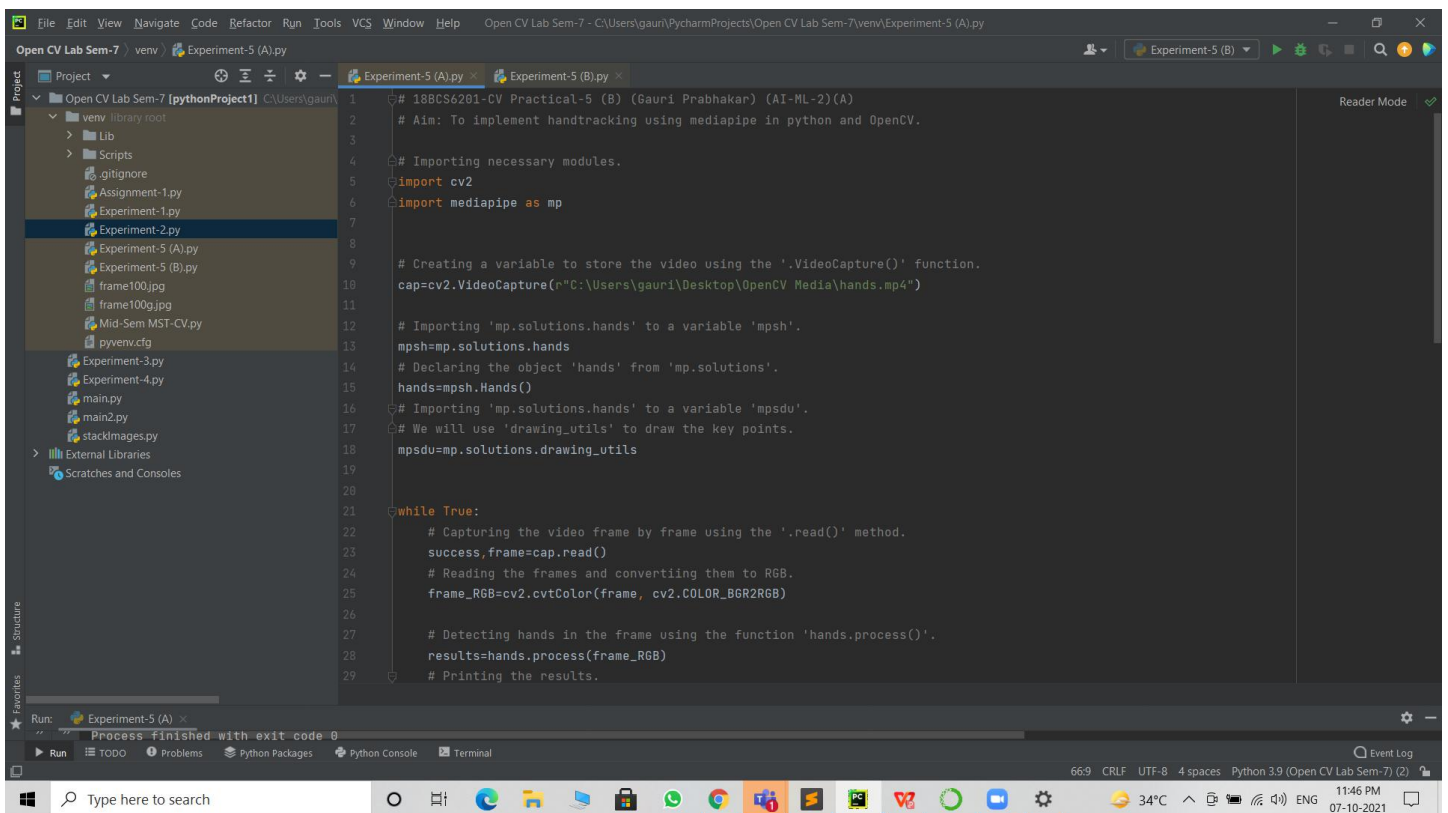
4. Result/Output/Writing Summary:



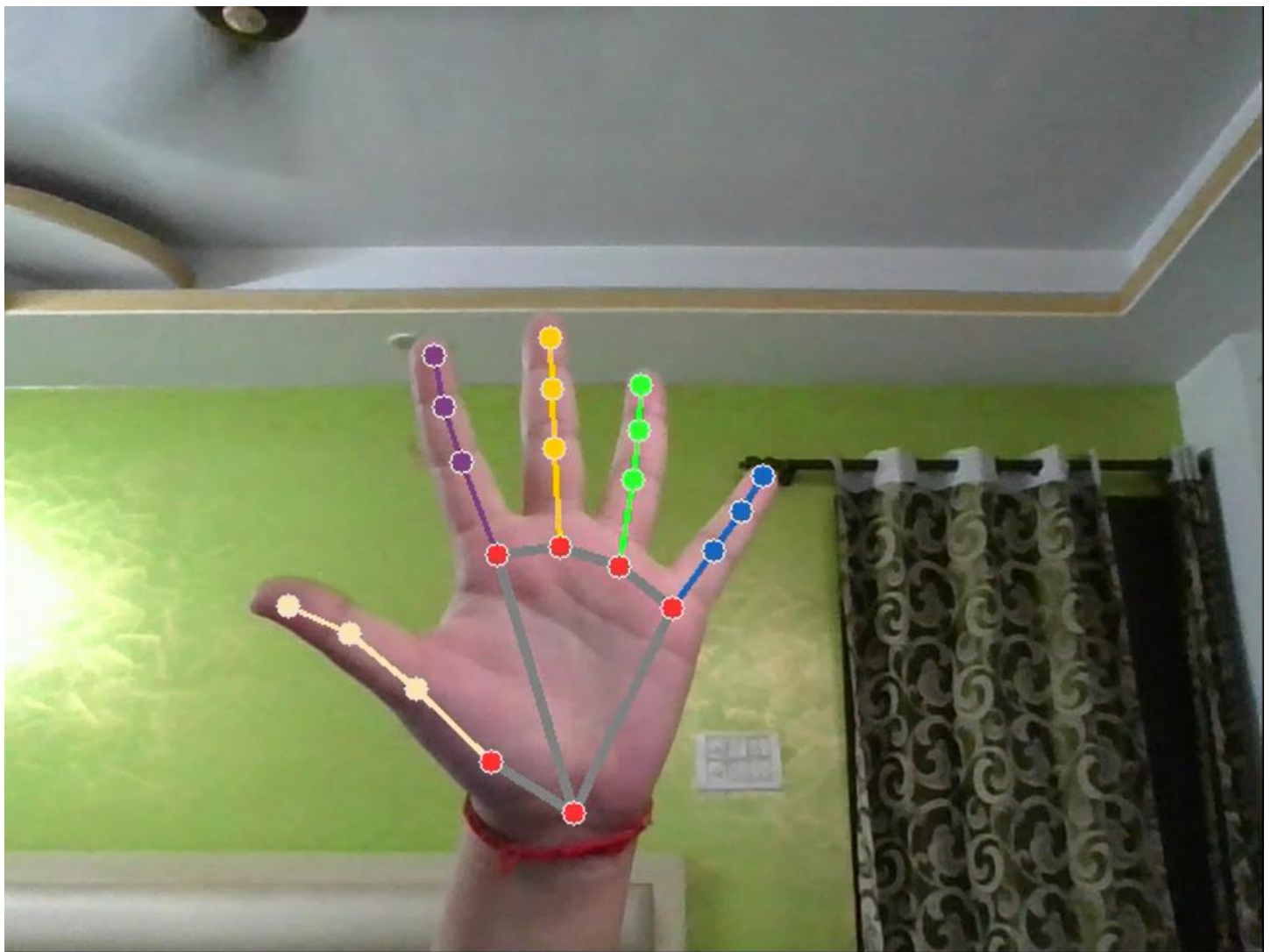
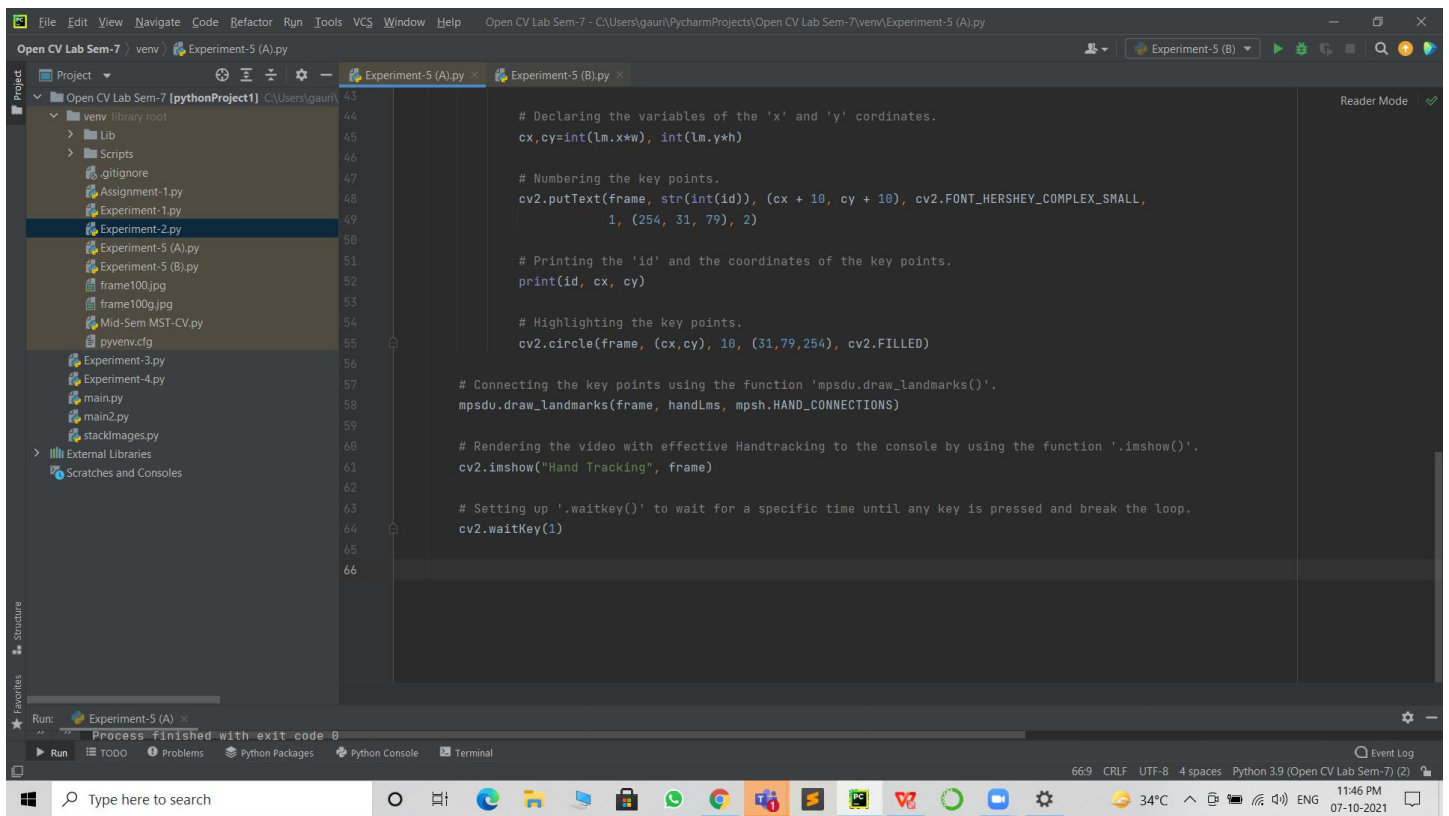




```
28 results=hands.process(frame_RGB)
29 # Printing the results.
30 #print(results.multi_hand_landmarks)
31
32 # If hands are detected that is 'results.multi_hand_landmarks' returns true:
33 if results.multi_hand_landmarks:
34     # For 'handLms' variable in 'results.multi_hand_landmarks':
35     for handLms in results.multi_hand_landmarks:
36         # Looping through the 'handLms' variable to find out the coordinates of each key point.
37         for id, lm in enumerate(handLms.landmark):
38             # Printing the landmarks and 'ids'.
39             print(id, lm)
40
41             # Declaring the variables 'h', 'w' and 'c'.
42             h,w,c=frame.shape
43
44             # Declaring the variables of the 'x' and 'y' coordinates.
45             cx,cy=int(lm.x*w), int(lm.y*h)
46
47             # Numbering the key points.
48             cv2.putText(frame, str(int(id)), (cx + 10, cy + 10), cv2.FONT_HERSHEY_COMPLEX_SMALL,
49                        1, (254, 31, 79), 2)
50
51             # Printing the 'id' and the coordinates of the key points.
52             print(id, cx, cy)
53
54             # Highlighting the key points.
55             cv2.circle(frame, (cx,cy), 10, (31,79,254), cv2.FILLED)
56
```



```
1 # 18BCS6201-CV Practical-5 (B) (Gauri Prabhakar) (AI-ML-2) (A)
2 # Aim: To implement handtracking using mediapipe in python and OpenCV.
3
4 # Importing necessary modules.
5 import cv2
6 import mediapipe as mp
7
8 # Creating a variable to store the video using the '.VideoCapture()' function.
9 cap=cv2.VideoCapture("C:\Users\gaurn\Desktop\OpenCV Media\hands.mp4")
10
11 # Importing 'mp.solutions.hands' to a variable 'mpsh'.
12 mps=mp.solutions.hands
13 # Declaring the object 'hands' from 'mp.solutions'.
14 hands=mpsh.Hands()
15
16 # Importing 'mp.solutions.hands' to a variable 'mpsdu'.
17 # We will use 'drawing_utils' to draw the key points.
18 mpsdu=mp.solutions.drawing_utils
19
20
21 while True:
22     # Capturing the video frame by frame using the '.read()' method.
23     success,frame=cap.read()
24     # Reading the frames and converting them to RGB.
25     frame_RGB=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
26
27     # Detecting hands in the frame using the function 'hands.process()'.
28     results=hands.process(frame_RGB)
29     # Printing the results.
```

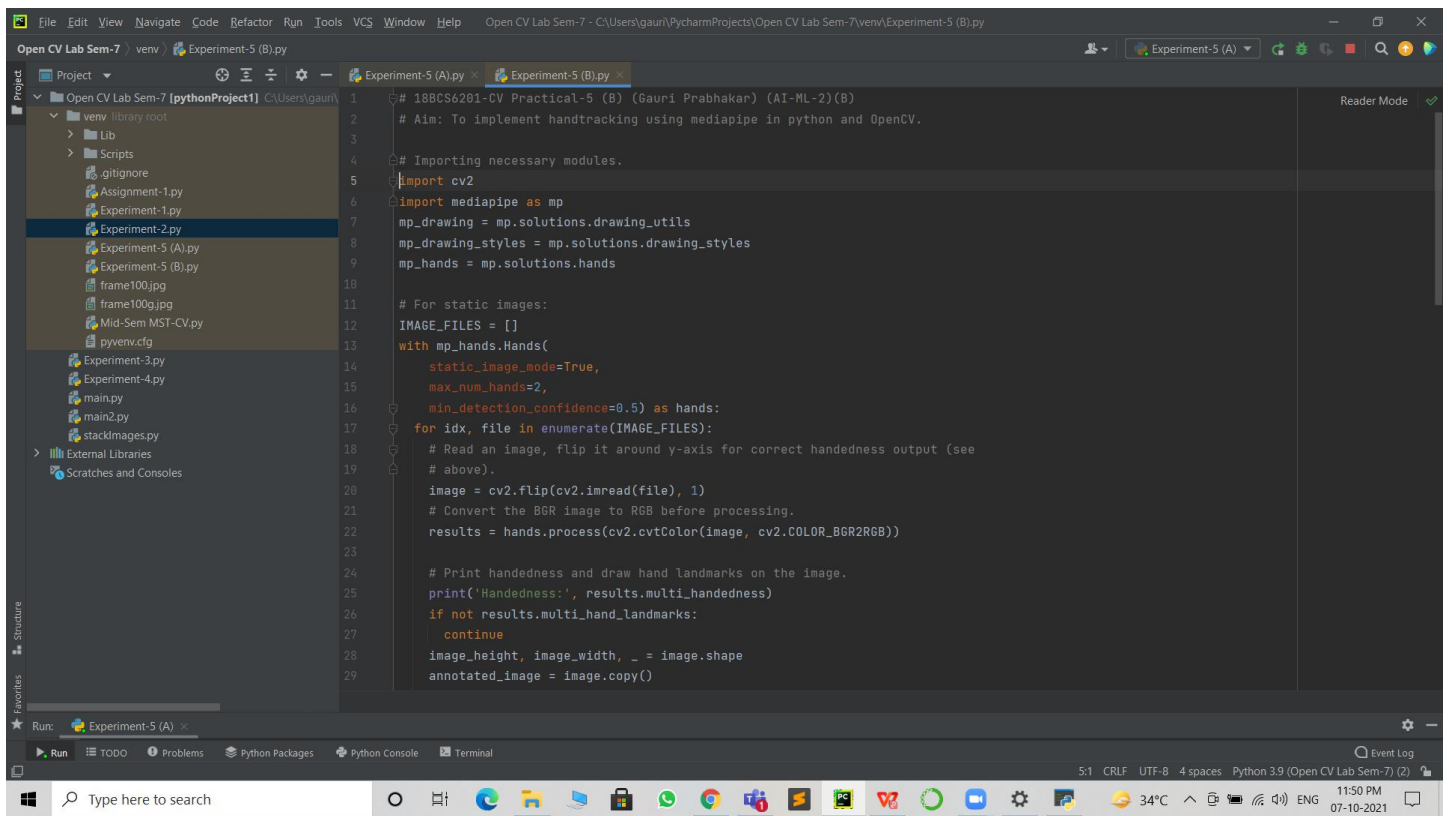
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Open CV Lab Sem-7 - C:\Users\gaurn\PycharmProjects\Open CV Lab Sem-7\venv\Experiment-5 (B).py
Experiment-5 (B).py
# For webcam input:
cap = cv2.VideoCapture(0)
with mp_hands.Hands(
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue

        # Flip the image horizontally for a later selfie-view display, and convert
        # the BGR image to RGB.
        image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
        # To improve performance, optionally mark the image as not writeable to
        # pass by reference.
        image.flags.writeable = False
        results = hands.process(image)

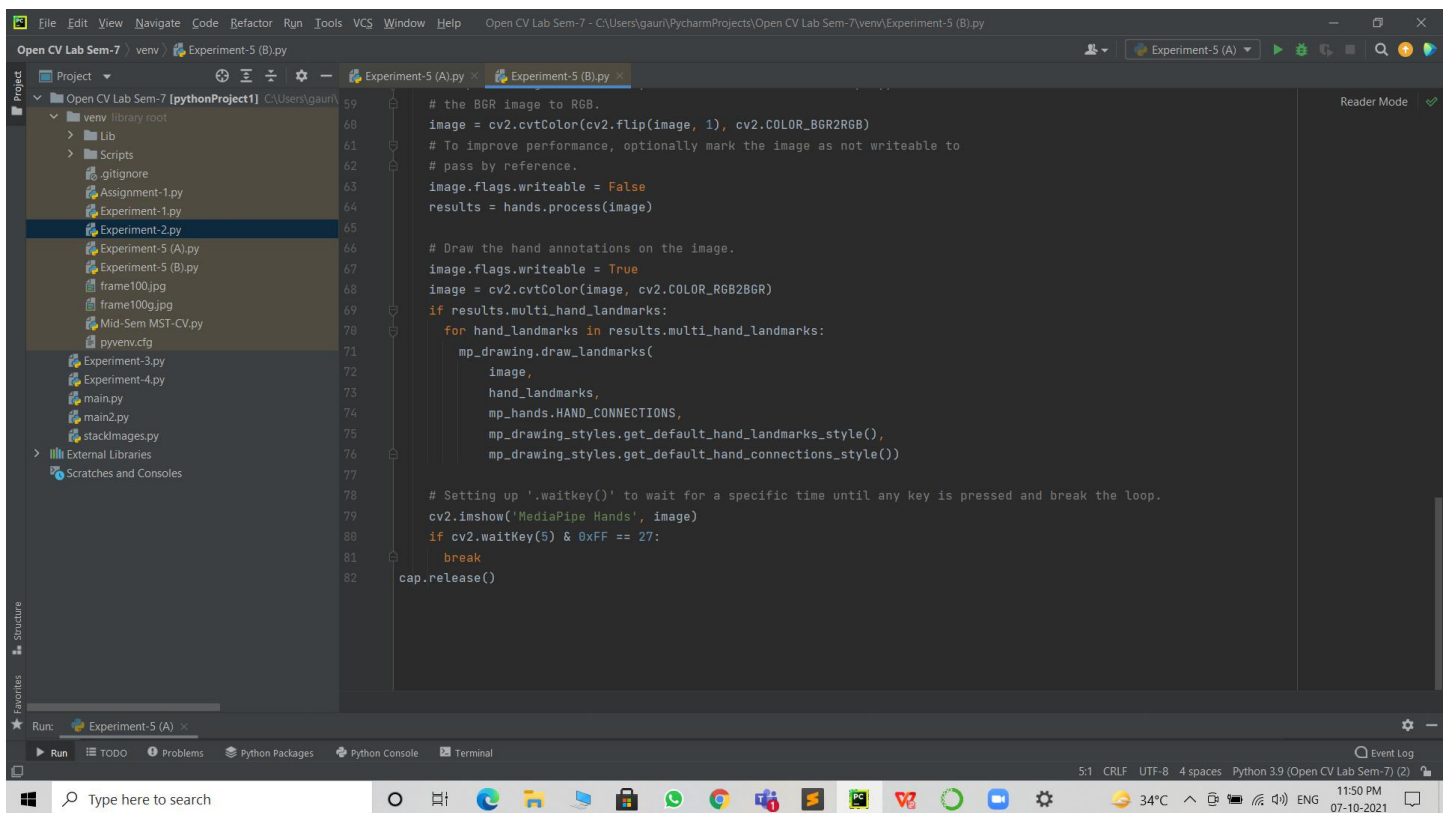
        # Draw the hand annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    image,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS,
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Open CV Lab Sem-7 - C:\Users\gaurn\PycharmProjects\Open CV Lab Sem-7\venv\Experiment-5 (B).py
Experiment-5 (B).py
image_height, image_width, _ = image.shape
annotated_image = image.copy()
for hand_landmarks in results.multi_hand_landmarks:
    print('hand_landmarks:', hand_landmarks)
    print(
        f'Index finger tip coordinates: ({
        f'{hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x * image_width}, '
        f'{hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y * image_height})'
    )
    mp_drawing.draw_landmarks(
        annotated_image,
        hand_landmarks,
        mp_hands.HAND_CONNECTIONS,
        mp_drawing_styles.get_default_hand_landmarks_style(),
        mp_drawing_styles.get_default_hand_connections_style())
cv2.imwrite(
    r'C:\Users\gaurn\Desktop\OpenCV Media' + str(idx) + '.png', cv2.flip(annotated_image, 1))

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_hands.Hands(
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue
```



```
1 # 18BCS6201-CV Practical-5 (B) (Gauri Prabhakar) (AI-ML-2)(B)
2 # Aim: To implement handtracking using mediapipe in python and OpenCV.
3
4 # Importing necessary modules.
5 import cv2
6 import mediapipe as mp
7 mp_drawing = mp.solutions.drawing_utils
8 mp_drawing_styles = mp.solutions.drawing_styles
9 mp_hands = mp.solutions.hands
10
11 # For static images:
12 IMAGE_FILES = []
13 with mp_hands.Hands(
14     static_image_mode=True,
15     max_num_hands=2,
16     min_detection_confidence=0.5) as hands:
17     for idx, file in enumerate(IMAGE_FILES):
18         # Read an image, flip it around y-axis for correct handedness output (see
19         # above).
20         image = cv2.flip(cv2.imread(file), 1)
21         # Convert the BGR image to RGB before processing.
22         results = hands.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
23
24         # Print handedness and draw hand landmarks on the image.
25         print('Handedness:', results.multi_handedness)
26         if not results.multi_hand_landmarks:
27             continue
28         image_height, image_width, _ = image.shape
29         annotated_image = image.copy()
```



```
59
60 # the BGR image to RGB.
61 image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
62 # To improve performance, optionally mark the image as not writeable to
63 # pass by reference.
64 image.flags.writeable = False
65 results = hands.process(image)
66
67 # Draw the hand annotations on the image.
68 image.flags.writeable = True
69 image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
70 if results.multi_hand_landmarks:
71     for hand_landmarks in results.multi_hand_landmarks:
72         mp_drawing.draw_landmarks(
73             image,
74             hand_landmarks,
75             mp_hands.HAND_CONNECTIONS,
76             mp_drawing_styles.get_default_hand_landmarks_style(),
77             mp_drawing_styles.get_default_hand_connections_style())
78
79 # Setting up 'waitKey()' to wait for a specific time until any key is pressed and break the loop.
80 cv2.imshow('MediaPipe Hands', image)
81 if cv2.waitKey(5) & 0xFF == 27:
82     break
83 cap.release()
```

5. Learning outcomes (What I have learnt):

- Open CV modules.
- The mediapipe library.
- Detect hands using the mediapipe library.
- Hand tracking a saved video.
- Hand tracking live video from the webcam.
- Highlighting key points.
- Labeling/numbering the key points.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			