Mini project On

AIR MOUSE

Under ASME, IIT Roorkee chapter

Indian Institute of Technology

Roorkee Uttarakhand

Made By

Sumit Kumar

Mentors

Srushti Borate
Aditya Gupta

AIR-Mouse

Task:

The task is to build a computer-vision(python) based mouse in which you perform basic mouse function (cursor hovering, scroll up\down, Right click, Left clicks) through your hand gestures.

Requirements:

For this project we will use python version 3.8.0 as most of the required library works with python 3.8.0,

In libraries

1) Open CV (4.6.0.66) for Video capture and image processing,
2) Mediapipe (0.8.10) for hand detection,
3) NumPy (1.23.1) for mathematical calculations,
4) Autopy (3.0.1) for smooth hovering of cursor,
5) Pyautogui (0.9.50) for Right\Left click Scroll up\Down &
6) Time for stopping the code to smooth transition is used

Source Code:

```
import cv2
import mediapipe as mp
import numpy as np
import autopy
import pyautogui as pug
import time


mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(min_detection_confidence=0.8, min_tracking_confidence=0.5)

# Defining the Global Constant
width_cam, Height_cam = 640, 430
smoothening = 10
# finding the Width, Height of screen
width_scr, height_scr = autopy.screen.size()
```

```python
frameR= 100
plocX, plocY = 0, 0
clocX, clocY = 0, 0
# definig the array of index of all finger tips
tipId = [4, 8, 12, 16, 20]

# Function to findposition of co-ordinates and depth of input frame
def findPosition(img):
    X_lst = []
    Y_lst = []
    land_mkslst = []
    if results.multi_hand_landmarks:
        for id, land_mks in enumerate(results.multi_hand_landmarks[0].landmark):
            height,width,depth = img.shape
            # print(height, width, depth)
            cx, cy = int(land_mks.x*width), int(land_mks.y*height)
            X_lst.append(cx)
            Y_lst.append(cy)
            land_mkslst.append([id, cx, cy])
            cv2.circle(img, (cx, cy), 5, (240, 22, 51), cv2.FILLED)
        xmin, xmax = min(X_lst), max(X_lst)
        ymin, ymax = min(Y_lst), max(Y_lst)
        bbox = xmin, ymin, xmax, ymax
        cv2.rectangle(img, (bbox[0],bbox[1]),
                              (bbox[2]+20, bbox[3]+20), (213, 235, 21), 2)
    return land_mkslst

# returns the length between two points in list of list of landmarks
def findDistance(pos1, pos2, land_mkslst):

    x1, y1 = land_mkslst[pos1][1], land_mkslst[pos1][2]
    x2, y2 = land_mkslst[pos2][1], land_mkslst[pos2][2]
    length = pow(abs((pow((x2-x1),2)-pow((y2-y1),2))), 0.5)
    return length


# returns a list of 1's or 0's for all five fingers
def fingersUp(land_mkslst):
    global tipId
    fingers = []
    # if Thunmb is up returns 0 if down returns 1
    try:
        if land_mkslst[tipId[0]][1] > land_mkslst[tipId[0]-1][1]:
             fingers.append(1)
        else:
            fingers.append(0)
    except:
        pass

    # if any of four fingers are up it retruns 1 otherwiae returns 0 to that finger index
    for i in range(1,5):
        try:
            if land_mkslst[tipId[i]][2] < land_mkslst[tipId[i]-1][2]:
                fingers.append(1)
            else:
                fingers.append(0)
        except:
            break
    return fingers


############################################### CODE INITIALIZATION
##################################################


cap = cv2.VideoCapture(0)

while(1):
    _, img = cap.read()
    #converting frame BGR to RGB
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    #Making image not changable for processing ease
    img.flags.writeable=False

    results = hands.process(img)

    img.flags.writeable=True

    img = cv2.cvtColor(img,cv2.COLOR_RGB2BGR)

    # Drawing landmarks in input frame
    if results.multi_hand_landmarks:
        for num, h_landmks in enumerate(results.multi_hand_landmarks):
            mp_drawing.draw_landmarks(img,h_landmks,mp_hands.HAND_CONNECTIONS,
                                            mp_drawing.DrawingSpec(color=(245, 218,
86),thickness=2,circle_radius=3),
                                            mp_drawing.DrawingSpec(color=(245, 218,
86),thickness=2,circle_radius=3))


    land_mkslst = findPosition(img)
    fingers = fingersUp(land_mkslst)
    # print(fingers)

    # taking x and y co-ordinate Index and Middel finger
    if len(land_mkslst)!=0:
        x1, y1 = land_mkslst[8][1:]
        x2, y2 = land_mkslst[12][1:]

    # move cursor if index finger is up and thumb is down
    try:
        if fingers[1]==1 and fingers[2]==0 and fingers[0]==1:
            x3 = np.interp(x1, (frameR, width_cam - frameR), (0, width_scr))
            y3 = np.interp(y1, (frameR, Height_cam - frameR), (0, height_scr))
            clocX = plocX + (x3 - plocX)/smoothening
            clocY = plocY + (y3 - plocY)/smoothening
            # print(clocX)
            # print(width_scr-clocX)
            autopy.mouse.move(width_scr-clocX, clocY)
            plocX, plocY = clocX, clocY
    except:
        pass

    # Right click if Index finger and middle finger is up and distance between them is less than 15
    try:
        if fingers[1]==1 and fingers[2]==1:
            if findDistance(8, 12, land_mkslst) < 15:
                pug.click(button='right')
    except:
        pass

    # Left click if Thumb is up and distance between Thumb and Index finger is less than 15
    try:
        if fingers[0]==0:
            if findDistance(4, 8, land_mkslst) < 15:
                pug.click(button='left')
    except:
        pass

    # Scroll Down on fist gesture
    try:
        if(fingers[0]==1 and fingers[1]==0 and fingers[2]==0 and fingers[3]==0 and fingers[4]==0):
            pug.scroll(-100)
            time.sleep(0.5)

    except:
        pass

    # Scroll Up if Middle Ring & Pinky finger is up
    try:
        if(fingers[0]==1 and fingers[1]==0 and fingers[2]==1 and fingers[3]==1 and fingers[4]==1):
            pug.scroll(100)
            time.sleep(0.5)
    except:
        pass

    # Code-block for terminating the code
    try:
        if fingers[0]==1 and fingers[1]==1 and fingers[2]==1 and fingers[3]==1 and fingers[4]==0:
            break
    except:
        pass

    cv2.imshow('Air-Mouse', cv2.flip(img, 1))
    cv2.waitKey(1)


cap.release()
cv2.destroyAllWindows()
```

Description and Implementation of code:

Mediapipe object creation: -

```
# initializing the mediapipe library and creating Drawing and hands objects
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(min_detection_confidence=0.8, min_tracking_confidence=0.5)
```

This is the initialization of the mediapipe library we are creating the mp_hands and mp_drawing objects from the mediapipes hands and drawing_utils method respectively, now using mp_hands creating another object hands in which the minimum detection confidence is 80% and minimum tracking confidence is 50%

Global Constant/Variable: -

```
# Defining the Global Constant
width_cam, Height_cam = 640, 440
smoothening = 7
# finding the Width, Height of screen
width_scr, height_scr = autopy.screen.size()
```

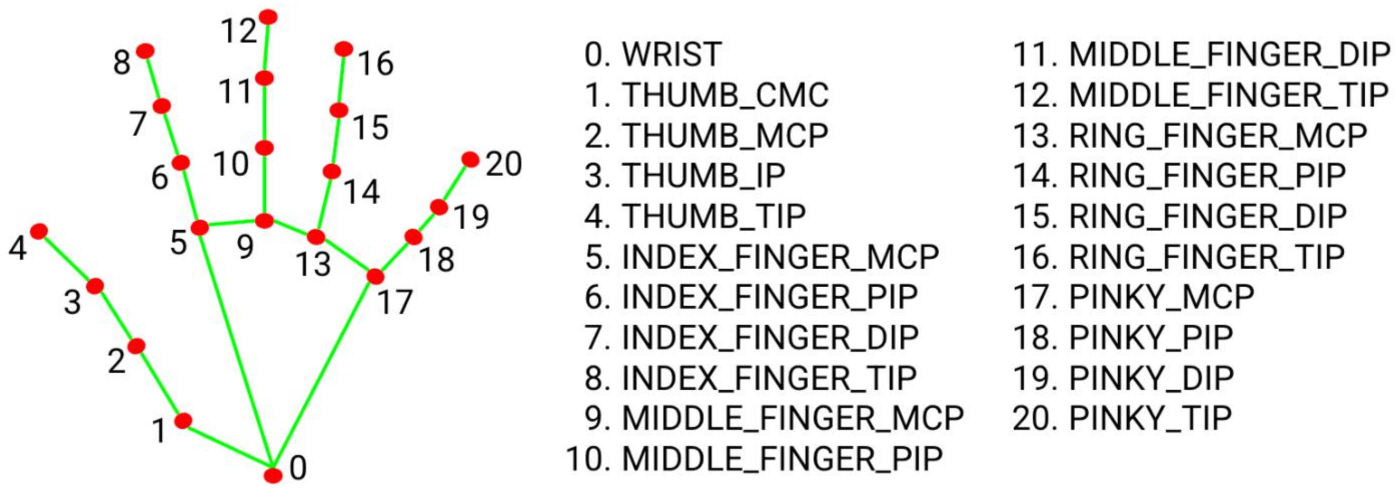
```
frameR= 100
plocX, plocY = 0, 0
clocX, clocY = 0, 0
# definig the array of index of all finger
tips tipId = [4, 8, 12, 16, 20]
```

Here we define some global constants/variables also using autopy.screen.size method to get the size(height*width) of the screen.

FindPosition Function: -

```
# Function to findposition of co-ordinates and depth of input frame
def findPosition(img):
    X_lst = [] Y_lst
    = [] land_mkslst
    = []
    if results.multi_hand_landmarks:
        for id, land_mks in enumerate(results.multi_hand_landmarks[0].landmark):
            height,width,depth = img.shape
            cx, cy = int(land_mks.x*width), int(land_mks.y*height)
            X_lst.append(cx)
            Y_lst.append(cy)
            land_mkslst.append([id, cx, cy])
            cv2.circle(img, (cx, cy), 5, (240, 22, 51), cv2.FILLED)
        xmin, xmax = min(X_lst), max(X_lst)
        ymin, ymax = min(Y_lst), max(Y_lst)
        bbox = xmin, ymin, xmax, ymax
        cv2.rectangle(img, (bbox[0],bbox[1]),
                          (bbox[2]+20, bbox[3]+20), (213, 235, 21), 2)
    return land_mkslst
```

Above code defines a function which takes numpy frame matrix as a input and using mediapipe image processing which is done only for single left hand which returns a 2d array of value tipid , X co-ordinate and Y co-ordinate as land marklist also findposition function draws a rectangle around the hand and circles in X, Y co-ordinate of the landmarks as shown below.



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

Fingers up Function: -

```
# returns a list of 1's or 0's for all five
fingers def fingersUp(land_mkslst):
    global  tipId
    fingers = []
    # if Thunmb is up returns 0 if down returns 1
    try:
        if land_mkslst[tipId[0]][1] > land_mkslst[tipId[0]-
             1][1]: fingers.append(1)
        else:
            fingers.append(0)
    except:
        pass

    # if any of four fingers are up it retruns 1 otherwiae returns 0 to that finger index
    for i in range(1,5):
        try:
            if land_mkslst[tipId[i]][2] < land_mkslst[tipId[i]-1][2]:
                fingers.append(1)
            else:
                fingers.append(0)
        except:
            break
    return fingers
```

fingersUp function returns a 1d array of 0's or 1's for thumb to each fingers respectively if any finger is up, it returns 1 else 0 (opposite for thumb as its up position is in x-axis so for uniformity 1 if thumb is down and 0 when thumb is up) E.g., finger is down for that index of finger, starting index 0 for thumb to index 4 to pinkie finger

FindDistance Function: -

```
# returns the length between two points in list of list of
landmarks def findDistance(pos1, pos2, land_mkslst):

    x1, y1 = land_mkslst[pos1][1], land_mkslst[pos1][2]
    x2, y2 = land_mkslst[pos2][1], land_mkslst[pos2][2]
    length = pow(abs((pow((x2-x1),2)-pow((y2-y1),2))),
    0.5) return length
```

The above function returns the distance between two points in the list of land marks as its attributes are index of the tip of finger(pos1), index of the tip of other finger(pos2) and land mark list(land_mkslst)

Code Initialization: -

```
################################# CODE INITIALIZATION #################################


cap = cv2.VideoCapture(0)

while(1):
    _, img = cap.read()
    #converting frame BGR to RGB
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    #Making image not changable for processing
    ease img.flags.writeable=False

    results = hands.process(img)

    img.flags.writeable=True

    img = cv2.cvtColor(img,cv2.COLOR_RGB2BGR)

    # Drawing landmarks in input frame
    if results.multi_hand_landmarks:
        for num, h_landmks in enumerate(results.multi_hand_landmarks):
            mp_drawing.draw_landmarks(img,h_landmks,mp_hands.HAND_CONNECTIONS,
                                                mp_drawing.DrawingSpec(color=(245, 218,
86),thickness=2,circle_radius=3),
                                                mp_drawing.DrawingSpec(color=(245, 218,
86),thickness=2,circle_radius=3))


    land_mkslst = findPosition(img)
    fingers = fingersUp(land_mkslst)
```

Code captures the video from cam 0 than we retrieve each frame of video and apply basic image processing like changing its format from BGR to RGB, than applying hands.process method to detect hand(can detect multi hand) if hand is detected than drawing the lines and circles in the detected hand finally invoking findposition to get the list of landmarks if hand is detected in frame(img)

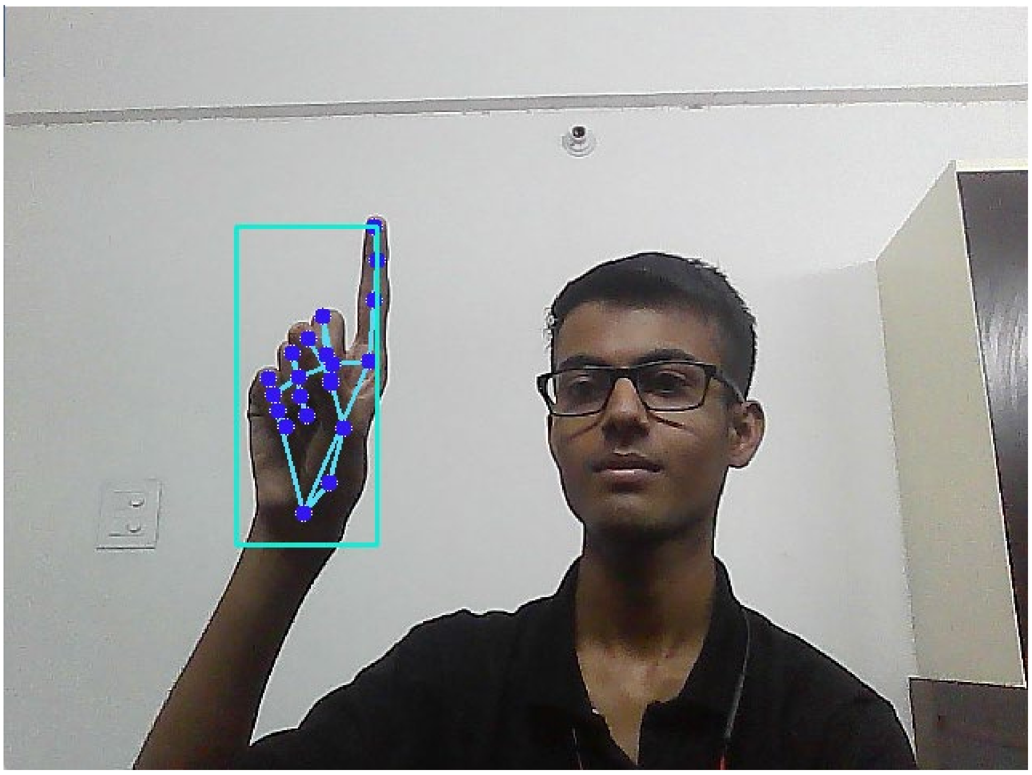
Secondly through fingersup getting the list of 0’s or 1’s according to whether the finger/thumb is up or down in frame and all this is in while loop so that it can iterate for each frame

Code block for Mousehover: -

```
# taking x and y co-ordinate Index and Middel
    finger if len(land_mkslst)!=0:
        x1, y1 = land_mkslst[8][1:]
        x2, y2 = land_mkslst[12][1:]

    # move cursor if index finger is up and thumb is down
    try:
        if fingers[1]==1 and fingers[2]==0 and fingers[0]==1:
            x3 = np.interp(x1, (frameR, width_cam - frameR), (0, width_scr))
            y3 = np.interp(y1, (frameR, Height_cam - frameR), (0,
            height_scr)) clocX = plocX + (x3 - plocX)/smoothening
            clocY = plocY + (y3 - plocY)/smoothening
            autopy.mouse.move(width_scr-clocX, clocY)
            plocX, plocY = clocX, clocY
    except:
        pass
```

This code-block when executed takes the X and Y co-ordinates of index and middle finger, it interpolates the new X, Y co-ordinates with respect to the height\width of input camera video/frame to the height/width of screen after which the values is decreased by some amount for smooth movement and added to the previous X, Y co-ordinate of the cursor the respective values are then sent to autopy.mouse.move method move the cursor.

Mouse movement gesture



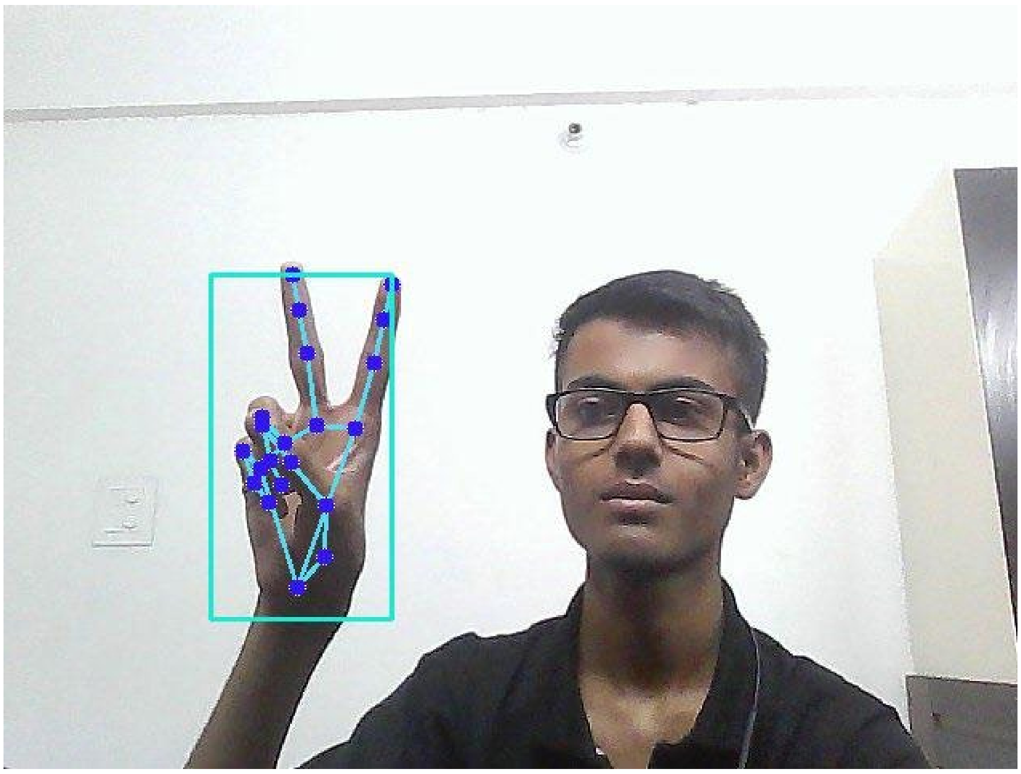
Code block for Right click: -

```
# Right click if Index finger and middle finger is up and distance between them is 
less than 15
    try:
        if fingers[1]==1 and fingers[2]==1:
            if findDistance(8, 12, land_mkslst) < 15:
                pug.click(button='right')
    except:
        pass
```

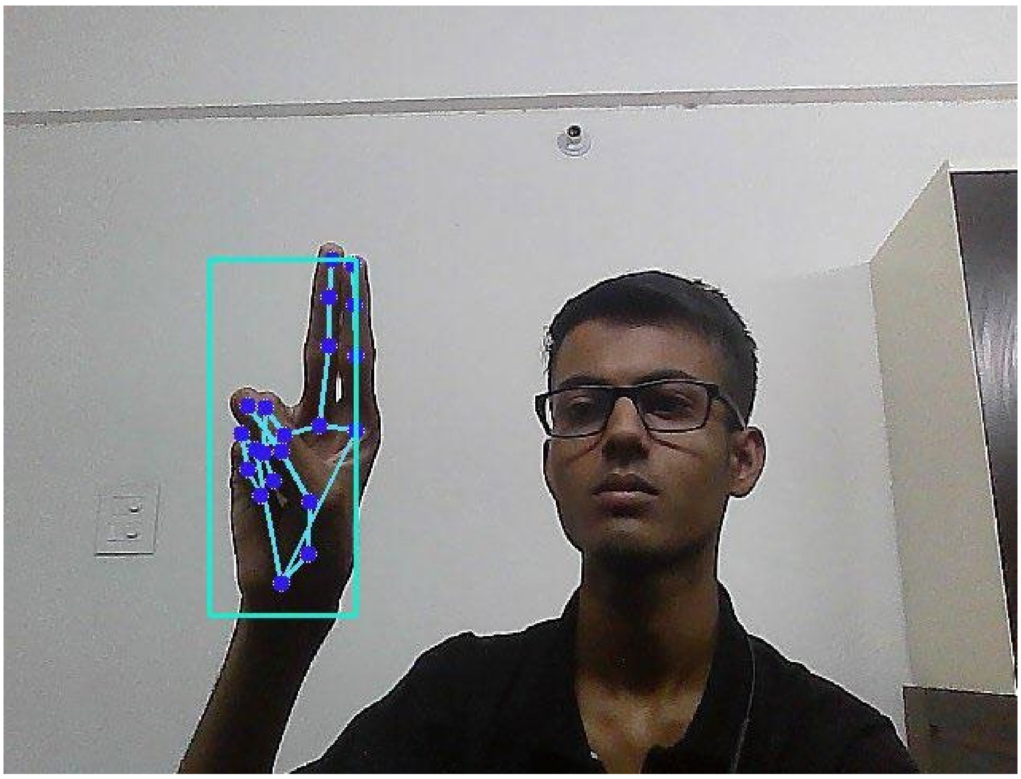
The above code is use to right click when left hands middle and index finger is up and distance between their tip is less than 15 pixel in screen (through webcam)

Right click Implementation is shown in images

Distance between the fingers is more than 15px so no Right click initiated:



Distance between fingers is less than is less than 15px so Right click function initiates:



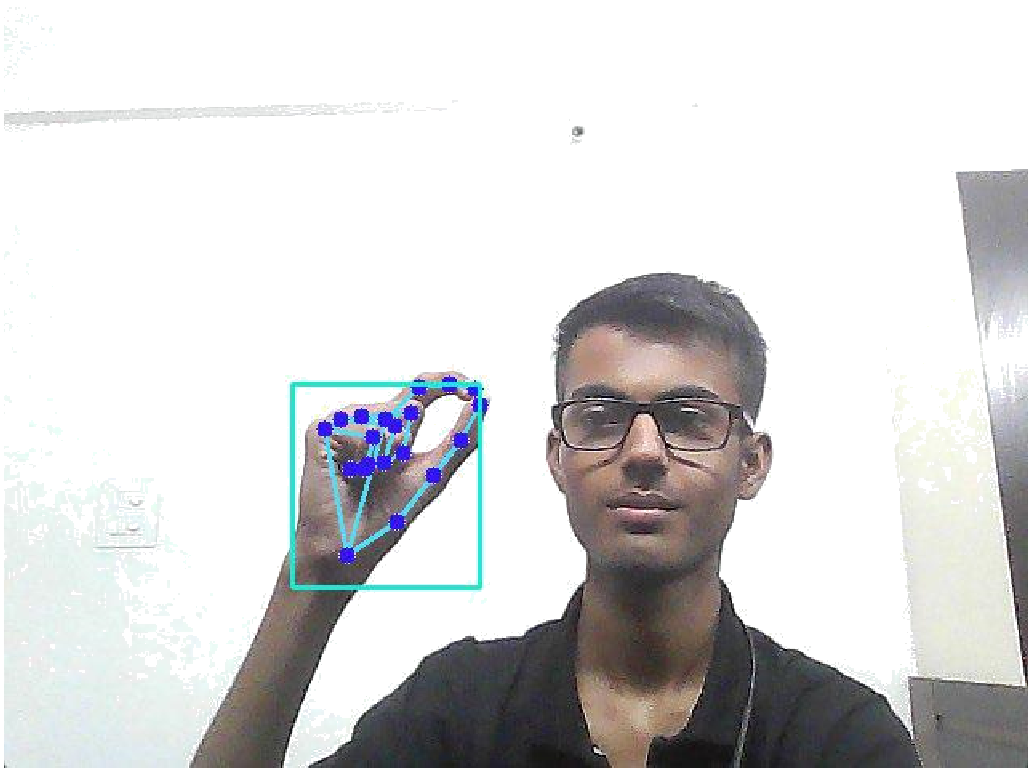
Code block for Left click: -

```
# Left click if Thumb is up and distance between Thumb and Index finger is less than
    15 try:
        if fingers[0]==0:
            if findDistance(4, 8, land_mkslst) < 15:
                pug.click(button='left')
    except:
        pass
```

The above code is use to left click when left hands thumb and index finger is up and distance between their tip is less than 15 pixel in screen (through webcam)

Left click implementation is shown in images below

The distance between index and thumb is less than 15px show left click function initiated

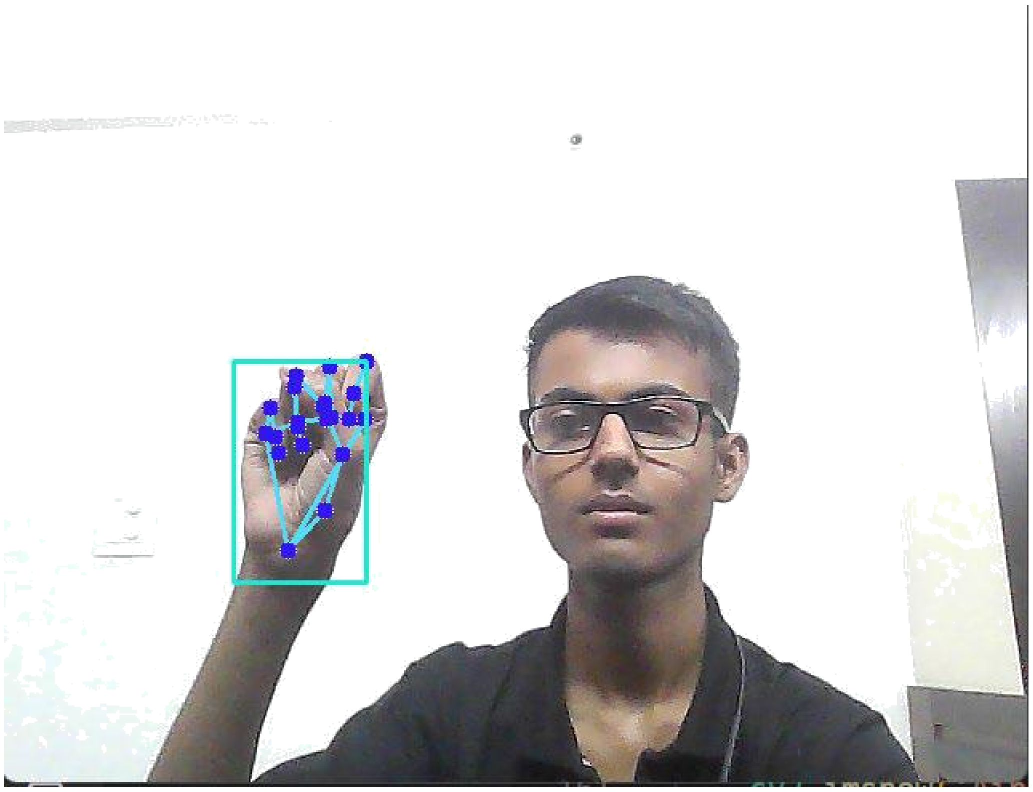


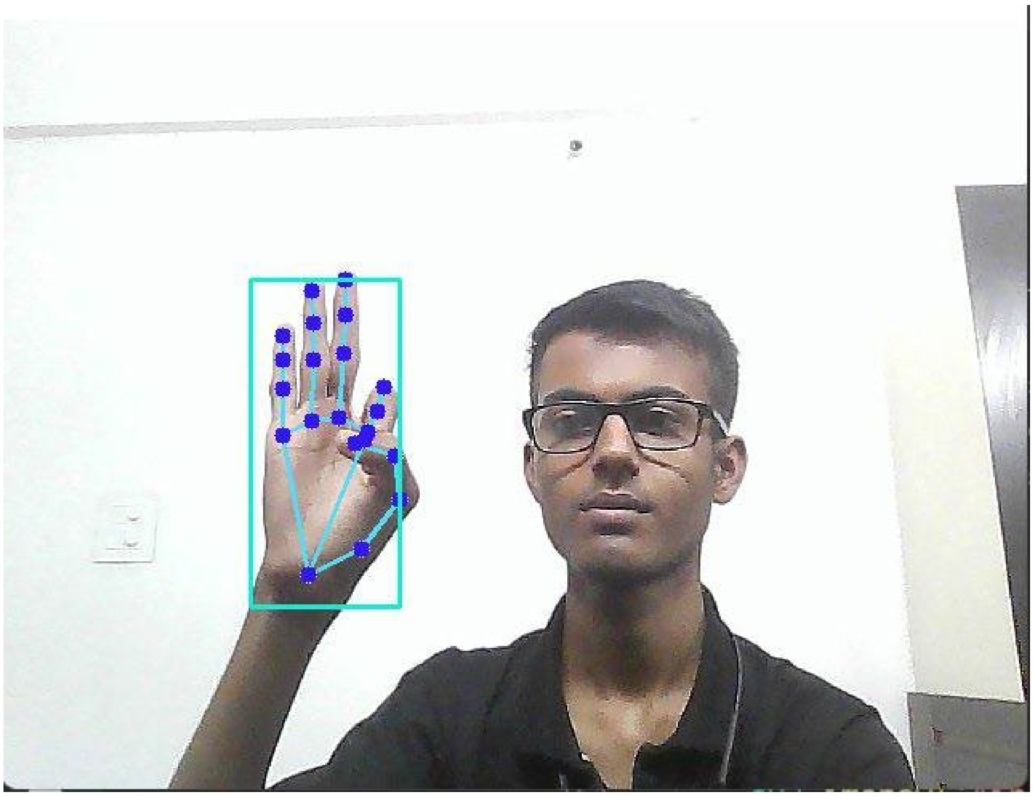
Code block for Scroll Down\Up: -

```
 # Scroll Down on fist
    gesture try:
        if(fingers[0]==1 and fingers[1]==0 and fingers[2]==0 and fingers[3]==0 and
fingers[4]==0):
            pug.scroll(-100)
            time.sleep(0.5)

    except:
        pass

    # Scroll Up if Middle Ring & Pinky finger is
    up try:
        if(fingers[0]==1 and fingers[1]==0 and fingers[2]==1 and fingers[3]==1 and
fingers[4]==1):
            pug.scroll(100)
            time.sleep(0.5)
    except:
        pass
```

The above code executes when there is a fist gesture for scroll down and scroll up when three fingers are up and index fingers and thumb is down

Scroll Down gesture


Scroll Up gesture

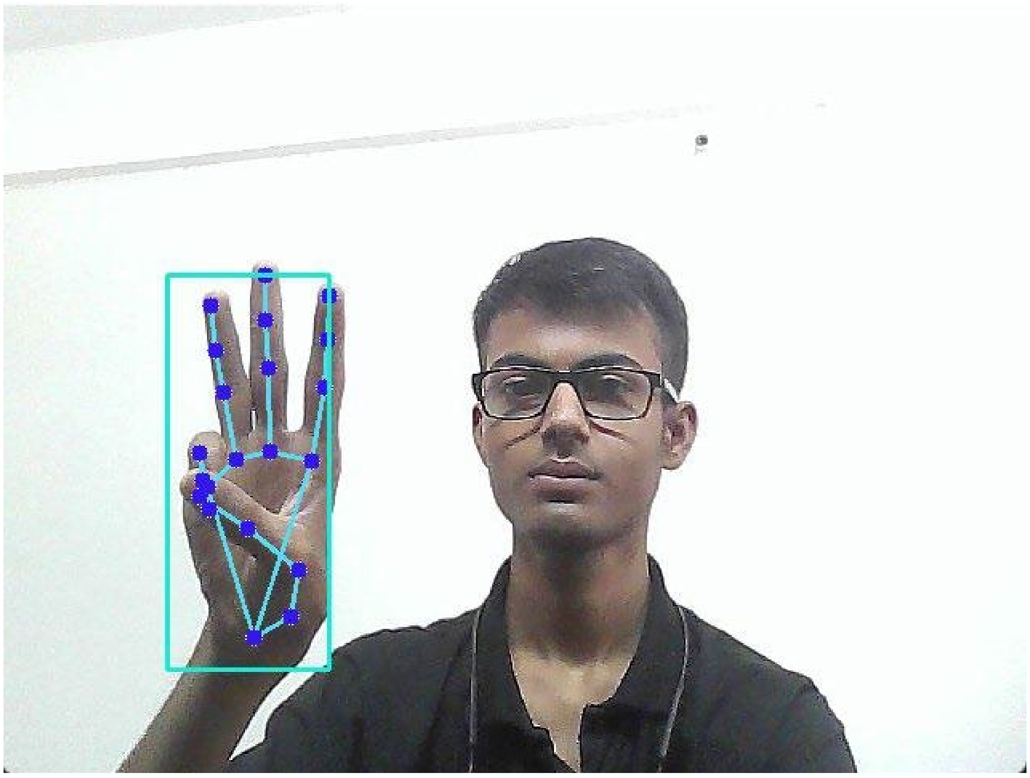


Code block for termination of code: -

```
 # Code-block for terminating the
    code try:
        if fingers[0]==1 and fingers[1]==0 and fingers[2]==1 and fingers[3]==0 and
fingers[4]==0:
            break
    except:
        pass

    cv2.imshow('Air-Mouse', cv2.flip(img, 1))
    cv2.waitKey(1)


cap.release()
cv2.destroyAllWindows()
```

the above code executes when Index, Middle and Ring fingers are up and rest are down when this gesture is made it simply terminates loop after which the video capture is released and all windows are destroyed

Termination gesture



Applications: -

The AI virtual mouse system is useful for many applications, it can be used to reduce the space for using the physical mouse, and it can be used in situations where we cannot use the physical mouse. The system eliminates the usage of devices, and it improves the human-computer interaction.

Major applications:

(i) Amidst the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to control the PC mouse functions without using the physical mouse.

(ii) The system can be used to control robots and automation systems without the usage of devices.

(iii) AI virtual mouse can be used to play virtual reality- and augmented reality-based games without the wireless or wired mouse devices.

(iv) Persons with problems in their hands can use this system to control the mouse functions in the computer.

(v) In designing and architecture, the proposed system can be used for designing virtually for prototyping.

Challenges faced: -

- Finding the corresponding module versions for all libraries which work with each other was tedious.
- It was difficult to set up the IDE because the modules were throwing an installation error.
- When using PyCharm IDE, the OpenCV library was not imported, so I used VS Code IDE instead.

- Smoothening of mouse cursor movement

Future Work: -

The proposed AI virtual mouse has some limitations such as small decrease in accuracy of the right click mouse function and also the model has some difficulties in executing clicking and dragging to select the text. These are some of the limitations of the proposed AI virtual mouse system, and these limitations will be overcome in future works.

References: -

1. For mathematics of code: https://www.youtube.com/watch?v=8tng9RsbXoU

2. For basic structure of code: https://github.com/HxnDev/Virtual-Mouse-using-OpenCV
3. For digital Image Processing: https://www.geeksforgeeks.org/digital-image-processing-basics/

4. For Mediapipe background processing: https://blog.gofynd.com/mediapipe-with-custom-tflite-model-d3ea0427b3c1

For in-depth learning about respective libraries:

- Mediapipe: https://google.github.io/mediapipe/solutions/hands
- Opencv: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- Autopy: https://github.com/autopilot-rs/autopy
- Pyautogui: https://pyautogui.readthedocs.io/en/latest/
- Numpy: https://numpy.org/doc/stable/