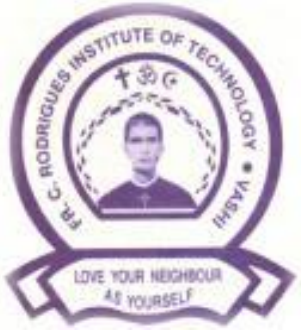
A Summer Project Report on

"AI Virtual Mouse Controller"

By

Joel Dsilva (5019116)
Aryan Koul (5019129)
Yash Pratapwar (5019148)
Abhinav Gajakosh (5019166)



Department of Information Technology
Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai – 400703

University of Mumbai
2020-2021

ABSTRACT

This project promotes an approach for the Human Computer Interaction (HCI) where cursor movement can be controlled using a real-time camera, it is an alternative to the current methods including manual input of buttons or changing the positions of a physical computer mouse. Instead, it utilizes a camera and computer vision technology to control various mouse events and is capable of performing every task that the physical computer mouse can.

The proposed system uses nothing more than a low-resolution webcam that acts as a sensor and it is able to track the users' hand, using mediapipe library's hand landmarks. The system will be implemented using the python, OpenCV, mediapipe and pyautogui. The hand gesture is the most effortless and natural way of communication. The output of the camera will be displayed on the monitor. Shape and position information about the gesture will be gathered using detection of hand landmarks.

INDEX

LIST OF FIGURES

INTRODUCTION

1.1 Background

A mouse, in computing terms is a pointing device that detects two-dimensional movements relative to a surface. This movement is converted into the movement of a pointer on a display that allows to control the Graphical User Interface (GUI) on a computer platform. There are a lot of different types of mouse that have already existed in the modern days technology, there's the mechanical mouse that determines the movements by a hard rubber ball that rolls around as the mouse is moved. Years later, the optical mouse was introduced that replace the hard rubber ball to a LED sensor to detects table top movement and then sends off the information to the computer for processing. On the year 2004, the laser mouse was then introduced to improve the accuracy movement with the slightest hand movement, it overcome the limitations of the optical mouse which is the difficulties to track high-gloss surfaces. However, no matter how accurate can it be, there are still limitations exist within the mouse itself in both physical and technical terms. For example, a computer mouse is a consumable hardware device as it requires replacement in the long run, either the mouse buttons were degraded that causes inappropriate clicks, or the whole mouse was no longer detected by the computer itself. Despite the limitations, the computer technology still continues to grow, so does the importance of the human computer interactions. Ever since the introduction of a mobile device that can be interact with touch screen technology, the world is starting to demand the same technology to be applied on every technological devices, this includes the desktop system. However, even though the touch screen technology for the desktop system is already exists, the price can be very steep. Therefore, a virtual human computer interaction device that replaces the physical mouse or keyboard by using a webcam or any other image capturing devices can be an alternative way for the touch screen. This device which is the webcam will be constantly utilized by software that monitors the gestures given by the user in order to process it and translate to motion of pointes, as similar to a physical mouse.

1.2 Motivation

It is fair to say that the Virtual Mouse will soon to be substituting the traditional physical mouse in the near future, as people are aiming towards the lifestyle where that every technological device can be controlled and interacted remotely without using any peripheral devices such as the remote, keyboards, etc. it doesn't just provides convenience, but it's cost effective as well.

1.2.1. Convenient: It is known in order to interact with the computer system, users are required to use an actual physical mouse, which also requires a certain area of surface to operate, not to mention that it suffers from cable length limitations. Virtual Mouse requires none of it, as it only a webcam to allow image capturing of user's hand position in order to determine the position of the pointers that the user want it to be. For example, the user will be able to remotely control and interact the computer system by just facing the webcam or any other image capturing devices and moving your fingers, thus eliminating the need to manually move the physical mouse, while able to interact with the computer system from few feet away.

1.2.2. Cost Effective: A quality physical mouse is normally cost from the range of INR 300 – INR 40,000 depending on their functionality and features. Since the Virtual Mouse requires only a webcam, a physical mouse are no longer required, thus eliminating the need to purchase one, as a single webcam is sufficient enough to allow users to interact with the computer system through it, while some other portable computer system such as the laptop, are already supplied with a built-in webcam, could simply utilize the Virtual Mouse software without having any concerns about purchasing any external peripheral devices.

1.2.3. Purpose: The Virtual Mouse application is expected to replace the current methods of utilizing a physical computer mouse where the mouse inputs and positions are done manually. This application offers a more effortless way to interact with the computer system, where every task can be done by gestures. Furthermore, the Virtual Mouse application could assist the motor-impaired users where he/she could interact with the computer system by just showing the correct combination of colors to the webcam.

1.3 Problem Definition

To design virtual mouse which detects hand gestures patterns instead of physical mouse. The camera is positioned such that it recognizes the moment of finger tips and performs the operations of mouse. The utilization of virtual mouse appears in space saving situations or in movement situation.

1.4 Scope

Virtual Mouse that will soon to be introduced to replace the physical computer mouse to promote convenience while still able to accurately interact and control the computer system. To do that, the software requires to be fast enough to capture and process every image, in order to successfully track the user's gesture. Therefore, this project will develop a software application with the aid of the latest software coding technique and the open-source computer vision library also known as the OpenCV. The scope of the project is as below:

- Real time application.
- User friendly application.
- Removes the requirement of having a physical mouse.

1.5 Proposed System Features

Using the current system even-though there are a number of quick access methods available for the hand and mouse gesture for the laptops, using our project we could make use of the laptop or web-cam and by recognizing the hand gesture we could control mouse and perform basic operations like mouse pointer controlling, select and deselect using left click, and a quick access feature for file transfer between the systems connected via network LAN cable. The project done is a “Zero Cost” hand recognition system for laptops, which uses simple algorithms to determine the hand, hand movements and by assigning an action for each movement. But we have mainly concentrated on the mouse pointing and clicking. The system we are implementing which is been written in python code be much more responsive and is easily implemented since python is a simple language and is platform independent with a flexibility and is portable which is desirable in creating a program which is focused in such an aim for creating a Virtual Mouse and Hand Recognition system. The system be much more extendable by defining actions for the hand movement for doing a specific action. It could be further modified to any further extent by implementing such actions for the set of hand gestures, the scope is restricted by your imagination.

1.6 Objectives

By this project we are aiming in creating a feasible hand recognition software for laptops and PCs with a web-cam support. The project covers as a hand recognition tool which could be used to move the mouse pointer, perform simple operations like clicking and scrolling.

1.7 Issues /Limitations

Although the virtual mouse offers a great replacement to the present physical mouse in computers, it has certain limitations as well. It cannot provide high precision performance, it cannot be used on PCs with no camera Input. It has less accuracy in clicking with respect to actual physical mouse (can be Improved with further development)

LITERATURE SURVEY

2.1 Survey of existing system

Track Ball:

The user rolls the ball with the thumb, fingers, or the palm of the hand to move a cursor.
Large tracker balls are common on CAD workstations for easy precision.
Before the advent of the touchpad, small trackballs were common on portable computers.

Disadvantages:
Usually not as accurate as a mouse.
Ball mechanism of trackballs requires more frequent cleaning than a mouse.
Not very user friendly.

Mechanical Mouse:

A single ball that could rotate in any direction.
As part of the hardware package of the Xerox Alto computer.
Detection of the motion of the ball was light based with the help of chopper wheels.

Disadvantages:
Cannot provide high precision performance.
Has specific surface requirements to operate.
Needs more desk space when compared with a trackball.

Touchpad:

It is a pointing device featuring a tactile sensor.
A specialized surface that can translate the motion and position of a user's fingers to a relative position on the operating system that is made output to the screen.

Disadvantages:
Moist, sweaty or calloused fingers can disrupt the signals picked up by the sensors.
Takes practice and skill to control the position of the cursor using the touchpad
Hard to use in cramped condition as there is limited space to move

SYSTEM DESIGN

3.1 Architectural Diagram/ block diagram

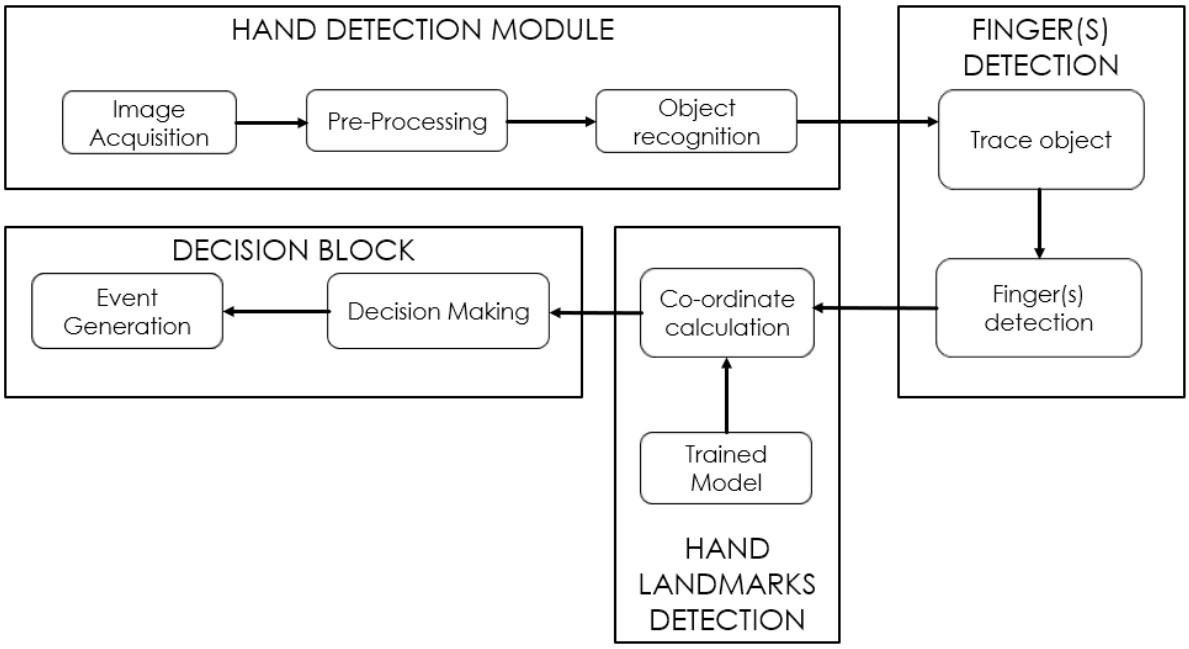


Figure 3.1.1: Architectural block diagram

- **Working of Hand Detection Module:**
  - Image Acquisition: The camera starts collecting images.
  - Pre-processing: The hand tracking module detects whether hand is visible in the image or not.
  - Object Recognition: The acquired images are fed to the model, which determines the hand area in the image and creates a box around it.

- **Working of Finger(s) Detection block:**
  - Trace Object: The model determines the region where the hand is located.
  - Finger(s) detection: The model then detects the tips of finger(s) visible.

- **Working of Hand Landmarks Detection block:**
    - Feature calculation: The models determines the co-ordinates the finger tips.

- **Working of decision block:**
    - Decision making: Based on the co-ordinates calculated, the programs determines the position of the cursor and moves the cursor to that position.
    - Event generation: According to the hand gestures that are detected, the program will execute the respective functions of the mouse.

- **Libraries**
    - **cv2:** OpenCV-Python is a library of Python bindings designed to solve computer vision problems.
    - **mediapipe:** MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.
    - **pyautogui:** Pyautogui is a library that allows you to control the mouse and keyboard to do various things. It is a cross-platform GUI automation Python module for human beings.

3.2 Algorithm and Process Design Implementation

Step 1 – Take image as input from a camera.

Step 2 – Detect the object in the image and if it is a hand create a Region Of Interest(ROI) around it.

Step 3 – Detect the fingers from the ROI.

Step 4 – The model will calculate the hand landmarks and based on it will understand the hand gesture made.

Step 5 – Based on the hand gesture detected, the program will execute certain function the mouse.

SYSTEM REQUIREMENT

4.1 Hardware Requirement

The following describes the hardware needed in order to execute and develop the Virtual Mouse application:

- **Computer Desktop or Laptop**
  The computer desktop or a laptop will be utilized to run the visual software in order to display what webcam had captured. A notebook which is a small, lightweight and inexpensive laptop computer is proposed to increase mobility.
  System will be using

Processor	:	intel i3
Main Memory	:	4GB RAM
Hard Disk	:	256 GB
Display	:	HD Monitor

- **Webcam**
  Webcam is utilized for image processing, the webcam will continuously taking image in order for the program to process the image and find pixel position.

4.2. Software Requirement

The following describes the software needed in-order to develop the Virtual Mouse application:

- **Python Language:** The coding technique on developing the Virtual Mouse application will be the Python with the aid of the integrated development environment (IDE) that are used for developing computer programs, known as the Microsoft Visual Studio.

- **OpenCV Library:** OpenCV are also included in the making of this program. OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. OpenCV have the utility that can read image pixels value, it also have the ability to create real time eye tracking and blink detection.

- **mediapipe**: MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.

- **time:** The Python time module provides many ways of representing time in code, such as objects, numbers, and strings.

- **math:** This module provides access to the mathematical functions defined by the C standard.

- **numpy:** NumPy is a library for the Python programming language, adding support for

large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **autopy:** AutoPy is a simple, cross-platform GUI automation toolkit for Python. It includes functions for controlling the keyboard and mouse, finding colors and bitmaps on-screen, and displaying alerts -- all in a cross-platform, efficient, and simple manner.

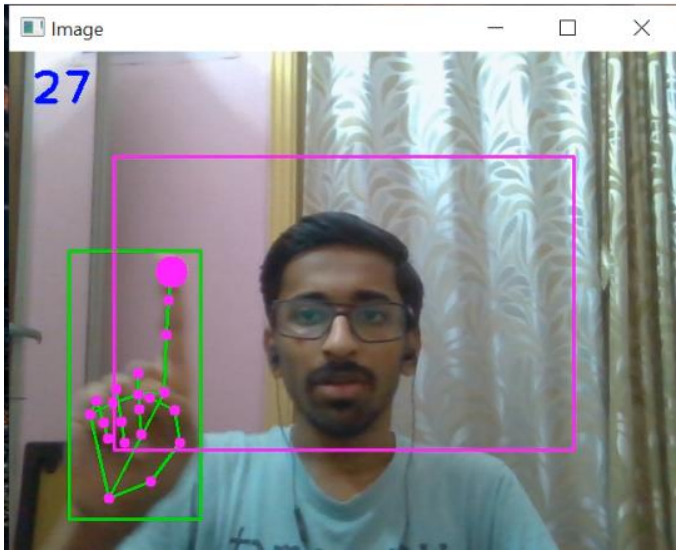
- **pyautogui:** Pyautogui is a library that allows you to control the mouse and keyboard to do various things. It is a cross-platform GUI automation Python module for human beings.

Software will be using:

OS	:	Window 10 64-bit
Language	:	Python
Tool Used	:	OpenCV

IMPLEMENTATION DETAILS

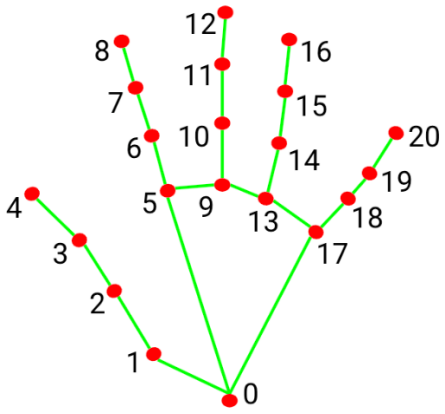
5.1 User Interface



5.1.1: User Interface

- The program instantly detects the hand and hand landmarks.
- We can make different gestures allotted to certain mouse function ko execute the respective mouse function, such as left-click, right-click, scroll up and down.

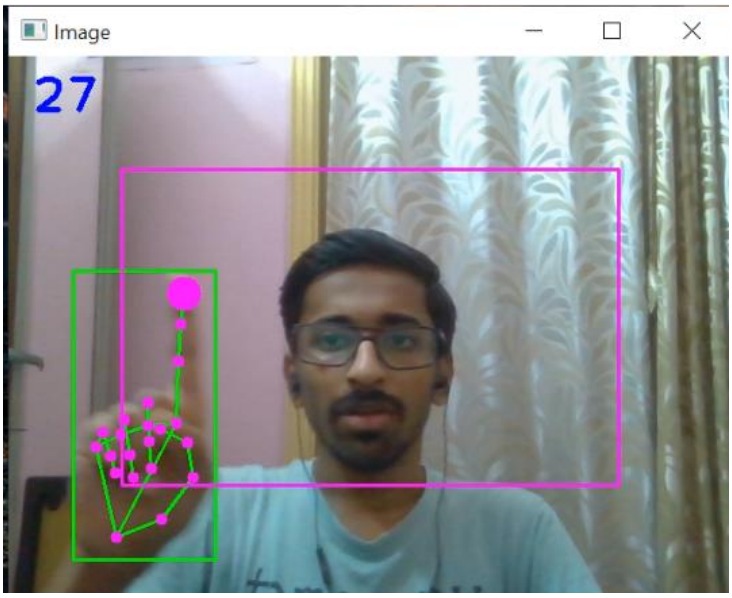
Following are the hand landmarks:



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

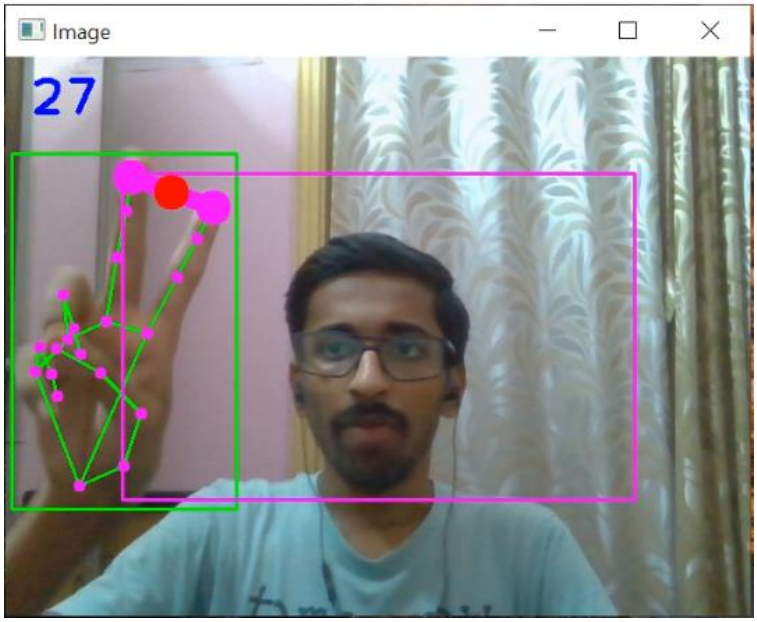
EXPERIMENTAL RESULTS

- **Free movement of mouse:** When only index finger is visible



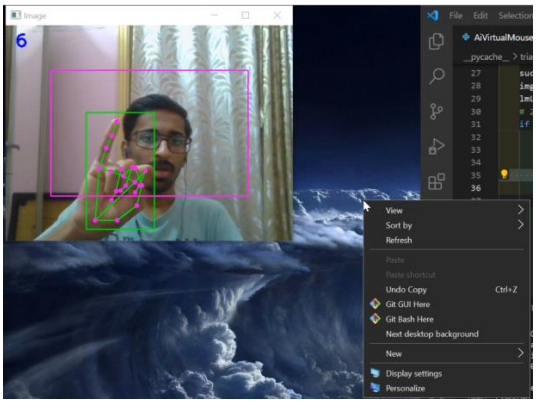
6.1: Free movement of cursor

- **Left-click:** When index and middle fingers are visible, they are moved closer together in a scissor-like fashion and when they are close enough, left-click will be executed.



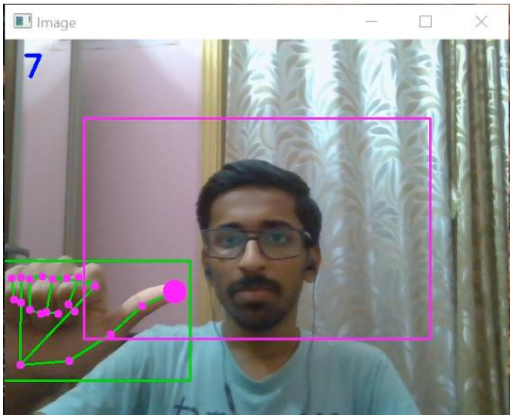
6.2: Left-click

- **Right-click:** When only little finger is visible right click is executed.



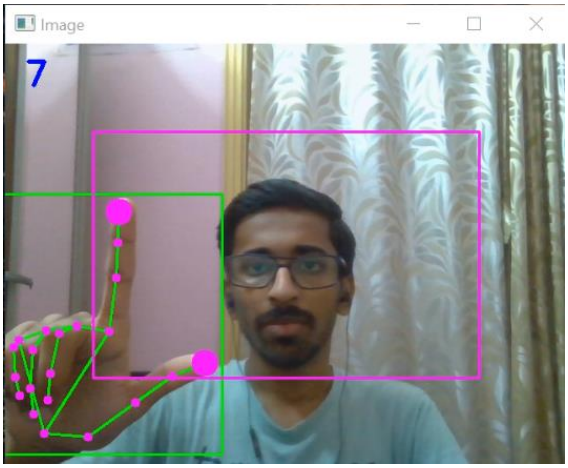
6.3: Right-click

- **Scroll-up:** When only thumb is shown, it will scroll up.



6.4: Scroll-up

- **Scroll-down:** When both index finger and thumb are shown, it will scroll down.



6.5: Scroll-down

CONCLUSION/FUTURE SCOPE

7.1 Conclusion

It is believed that gestures are the easiest way of interaction with anyone. So then why not apply it to the machines that we are using. The motive of this project was to make the machine more interactive and responsive towards human behaviour. The sole aim of this paper was to make a technology that is affordable and portable with any standard operating system.

The proposed system is used to control the pointer of the mouse by detecting the human hand and moving the pointer in the direction in the human hand respectively. The system control simple function of the mouse such as left-clicking, right-clicking, and cursor movement.

7.1 Future Scope

This project can easily replace the traditional mouse system that has been in existence for decades, with the use of this algorithm the user can control the cursor without the use of any other hardware device this is done using a hand gesture recognition library with input from a web-cam.

As there can be innumerable hand gestures made with single hand, developing this project further the hand detection technique can be used to make virtual keyboards which will type the alphabet with respect to hand gestures. This technology could also replace use of remote control in electronic devices.

Appendix: Code Sample

- **Hand Tracking Module**

```
import cv2
import mediapipe as mp
import time
import math
import numpy as np


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        #print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    def findPosition(self, img, handNo=0, draw=True):
        xList = []
        yList = []
```

```
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                #print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                #print(id, cx, cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)

        #Fingers
        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)
```

```
        # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True, r=15, t=13):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(1)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        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("Image", img)
```

```
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```

- **AiVirtualMouse.py**

```
import cv2
import numpy as np
import HandTrackingModule as htm
import time
import autopy
import pyautogui

##########################
wCam, hCam = 640, 480
frameR = 100  # Frame Reduction
smoothening = 7
##########################

pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0

cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.handDetector(maxHands=1)
```

```
wScr, hScr = autopy.screen.size()
#print(wScr, hScr)

while True:
    # 1. Find hand Landmarks
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)
    # 2. Get the tip of the index and middle fingers
    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]
        x0, y0 = lmList[4][1:]
        x4, y4 = lmList[20][1:]
        #print(x1, y1, x2, y2)

        # 3. Check which fingers are up
        fingers = detector.fingersUp()
        # print(fingers)
        cv2.rectangle(img, (frameR, frameR), (wCam-frameR, hCam-frameR),
                      (255, 0, 255), 2)  # for top and bottom reach
        # 4. Only Index Finger : Moving Mode
        if fingers[1] == 1 and fingers[2] == 0:
            # 5. Convert Coordinates
            x3 = np.interp(x1, (frameR, wCam-frameR), (0, wScr))
            y3 = np.interp(y1, (frameR, hCam-frameR), (0, hScr))
            # 6. Smoothen Values
```

```
    clocX = plocX + (x3 - plocX) / smoothening
    clocY = plocY + (y3 - plocY) / smoothening

    # 7. Move Mouse
    autopy.mouse.move(wScr-clocX, clocY)
    cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
    plocX, plocY = clocX, clocY

# 8. Both Index and middle fingers are up : Clicking Mode
if fingers[1] == 1 and fingers[2] == 1:
    # 9. Find distance between fingers
    length, img, lineInfo = detector.findDistance(8, 12, img)
    # print(length)
    # 10. Click mouse if distance short
    if length < 30:
        cv2.circle(img, (lineInfo[4], lineInfo[5]),
        15, (0, 255, 0), cv2.FILLED)
        pyautogui.click(button='left')

    # # scroll up
    # if length > 45:
    #     pyautogui.scroll(-10)

#scroll down
if fingers[0] == 1 and fingers[1] == 0 and fingers[4] == 0 and fingers[2] == 0:
    cv2.circle(img, (x0, y0), 15, (255, 0, 255), cv2.FILLED)
    # cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
```

```
        pyautogui.scroll(15)

    #right click
    if fingers[1] == 0 and fingers[4] == 1 and fingers[0] == 0:
        pyautogui.click(button='right')

    #scroll up
    if fingers[0] == 1 and fingers[1] == 1:
        cv2.circle(img, (x0, y0), 15, (255, 0, 255), cv2.FILLED)
        pyautogui.scroll(-15)

# 11. Frame Rate
cTime = time.time()
fps = 1/(cTime-pTime)
pTime = cTime
cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,
            (255, 0, 0), 3)
# 12. Display
cv2.imshow("Image", img)
cv2.waitKey(1)
```

REFERENCES

[1] www.stackoverflow.com

[2] www.tutorialspoint.com

[3] www.researchgate.net

[4] www.Quora.com

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. We consider ourself privileged to express gratitude and respect towards all those who have guided us through the completion of this project.

We convey thanks to all our teachers of Information Technology Department for providing encouragement, constant support and guidance which was of a great help to complete this project successfully.