# Table of Contents

# Chapter 1

# 1. Introduction

**Virtual Mouse:** Through this project we try to improve our knowledge of Python by practicing our existing skills in some real world application. We decided to make a system that can use hand/fingers motions as input and can enable one to control cursor without actually using a physical mouse or touchpad.

In the digital information time, daily life is inseparable with human-computer interface (HCI). Human computer interaction has a long history to become more intuitive. For human being, hand gesture of different kind is one of the most intuitive and common communication.

Human Computer Interaction with a personal computer today is not just limited to keyboard and mouse interaction. Interaction between humans comes from different sensory modes like gesture, speech, facial and body expressions.

However, vision-based hand gesture recognition is still a challenging problem. In this project an embedded virtual mouse system by using hand gesture recognition is proposed.

This project was built keeping in mind about latest advancements in technology i.e AI and image processing and it would also help people to use their systems more freely and easily.

# Chapter 2
# 2. LITERATURE SURVEY

These are the following research papers we have read to decide our project and these papers include the previous work done on this topic.

**1.** J. Univers. Comput. Sci. 14 (19), 3127-3141, 2008The following paper introduces the work conducted to create a relative virtual mouse based on the interpretation of head movements and face gesture through a low cost camera and the optical flow of the images.

Link to the following research paper is:

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=virtual+mouse&oq=virtual+mou#d=gs_qabs&u=%23p%3DpUv7xaNVEFYJ

**2.** DS Suresh, IV Bhavanavol 1, 23-32, 2014 Here we present a novel approach for Human Computer Interaction (HCI) where, we control cursor movement using a realtime camera and color pointers. A color pointer has been used for the object recognition and tracking.

Link to the following research paper is:

https://scholar.google.com/scholar?hl=en&as_sdt=0,5&qsp=3&q=virtual+mouse+using+color&qst=ib#d=gs_qabs&u=%23p%3DsAF7G3_Zm7MJ

## 2.1. Research Approach

In B tech third year we decided to make a project through which we can do some real- world task. We decided to make a project which will solve a real-world problem.

After some brainstorming, we landed up on making a virtual mouse. We developed an AI-based approach for controlling the mouse movement using python and openCV with real-time camera that detects hand landmarks, tracks gesture patterns instead of a physical mouse. This project will also boost our existing knowledge of python programming language and will also enable us to actually solve areal-world problem.

We started to search for the things which can help us and after lot of searching on Google we discovered two modules named as openCV and mediapipe.

OpenCV- (For image processing and drawing) Mediapipe- (For Hand Tracking)
Autopy- (For controlling the mouse movement and click)

These are promising module with easy to use methods and a huge community support.

Our Idea for the project was very simple. It is a simple cursor controlling system which takes hand gestures as input. To achieve our goal we used python, studied openCV and mediapipe documentation and also watched several youtube videos to understand their functioning.

Github: After making the project we also decided to make a github repository of our project.

Here is the link of our repository: https://github.com/tusharchauhantc14/Virtual-Mouse

# Chapter 3

## 2. Problem Description

The virtual mouse has been tried to make so many times. The virtual mouse which were made previously had some limitations. Through this project we are trying to remove those limitations.

Two types of virtual mouse has already been made. One of them uses the finger caps which are tracked by the web cam of our computer. Those finger caps contains different types of color lights which are detected by the webcam.

And another one which was made uses the robotic hand to control the mouse. So through our project we want to remove the use of any type of external object or device.

We are trying to make a virtual mouse which doesn't need any type of assistance from any hardware. So that it can work totally on the software basis.

Our virtual mouse will detect the hands of the person and work according to them. We have used python language to make this project.

# Chapter 4
# 4. Purpose of The Project

The purpose of any project is to benefit the society in an effective way. The purpose of building this project is to solve real world problem and expand our python knowledge. This project will help people to easily control and operate there devices.

The main objective of the proposed system is to perform computer mouse cursor functions and scroll function using a web camera or a built-in camera in the computer instead of using a traditional mouse device. Hand gesture and hand tip detection by using computer vision is used as a HCI [1] with the computer.

## Aims

1. Expand our Python knowledge.
2. Build some real-world project.

## Outcome

A virtual mouse system which responds to hand/fingers gestures applied using python.

## Initial Research

Main research in our project was to find a suitable python library that is both compatible and easy to use and learn.

After searching through various libraries we selected openCV and mediapipe, autopy libraries .

# Chapter 5

# 5. PROJECT DESCRIPTION

As a virtual mouse, this project provides user to control the mouse of their system just by giving hand gestures. The system will automatically detect the hand gestures and user will be able to control the mouse of the system on their will.

## 5.1. TOOLS AND TECHNOLOGIS USED

### 5.1.1. PYTHON

**Python** is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

#### 5.1.1.1 OpenCv

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. In this tutorial, we explain how you can use OpenCV in your applications.

#### 5.1.1.2. Mediapipe

Media Pipe is a framework for building multimodal(e.g. video, audio or any time series data),cross-platform (i.e. Android, IOS, web, edge devices) applied ML pipelines. Mediapipe also facilitates the

deployment of machine learning technology into demos and applications on a wide variety of different hardware platforms.

### 5.1.1.3. Autopy

AutoPy is a simple, cross-platform GUI automation library for Python. It includes functions for controlling the keyboard and mouse, finding colors and bitmaps on-screen, and displaying alerts.

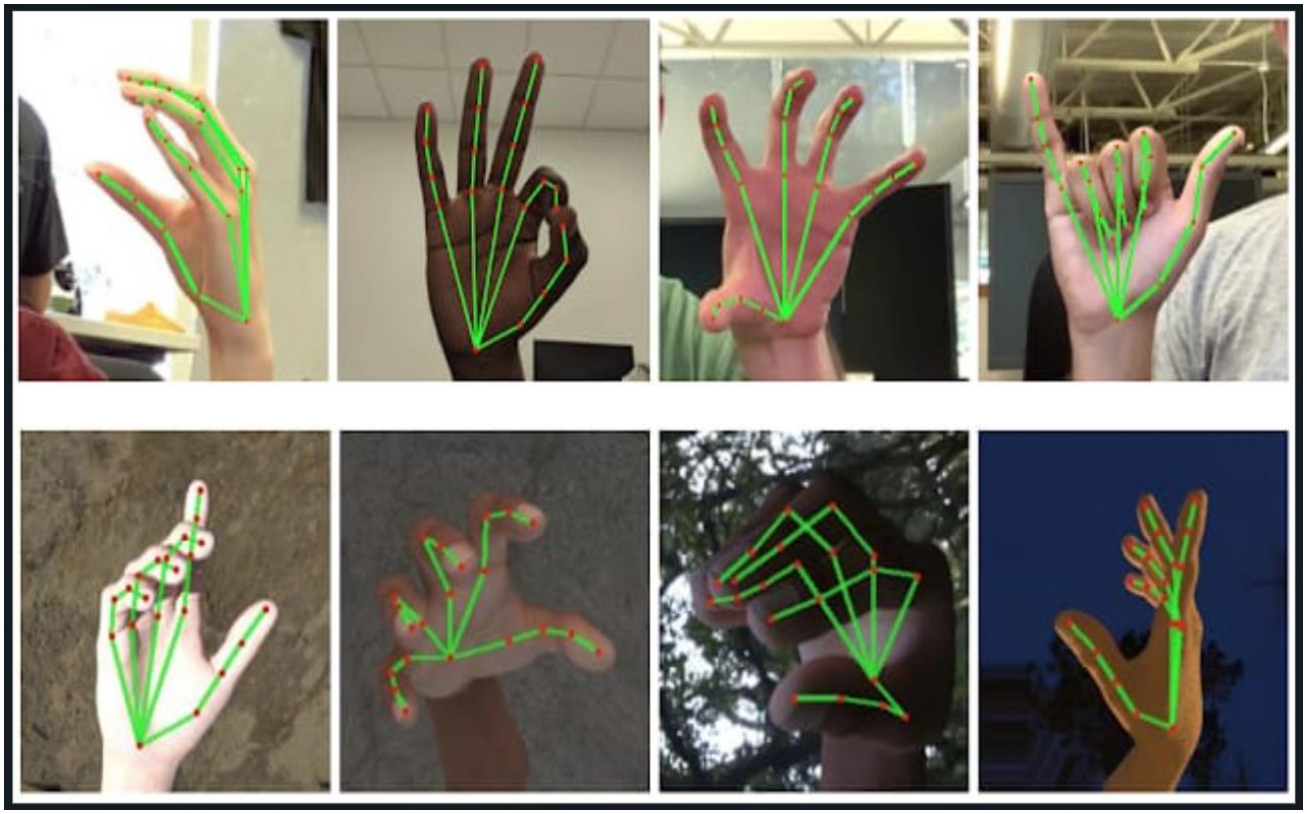
Currently supported on macOS, Windows, and X11 with the XTest extension.

## 5.2. Workflow of the Model

With the development technologies in the areas of augmented reality and devices that we use in our daily life, these devices are becoming compact in the form of Bluetooth or wireless technologies. This proposes an AI virtual mouse system that makes use of the hand gestures and hand tip detection for performing mouse functions in the computer using computer vision.

With the use of the AI virtual mouse system, we can track the fingertip of the hand gesture by using a built-in camera or web camera and perform the mouse cursor operations and scrolling function and also move the cursor with it.

Virtual mouse detects the hands by giving the dots to different positions in the hand. The model is trained according to that.



## 5.3. Methodology

The various functions and conditions used in the system are explained in the flowchart of the real-time AI virtual mouse

The Camera Used in the AI Virtual Mouse System. The proposed AI virtual mouse system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library OpenCV, the video capture object is created and the web camera will start capturing video

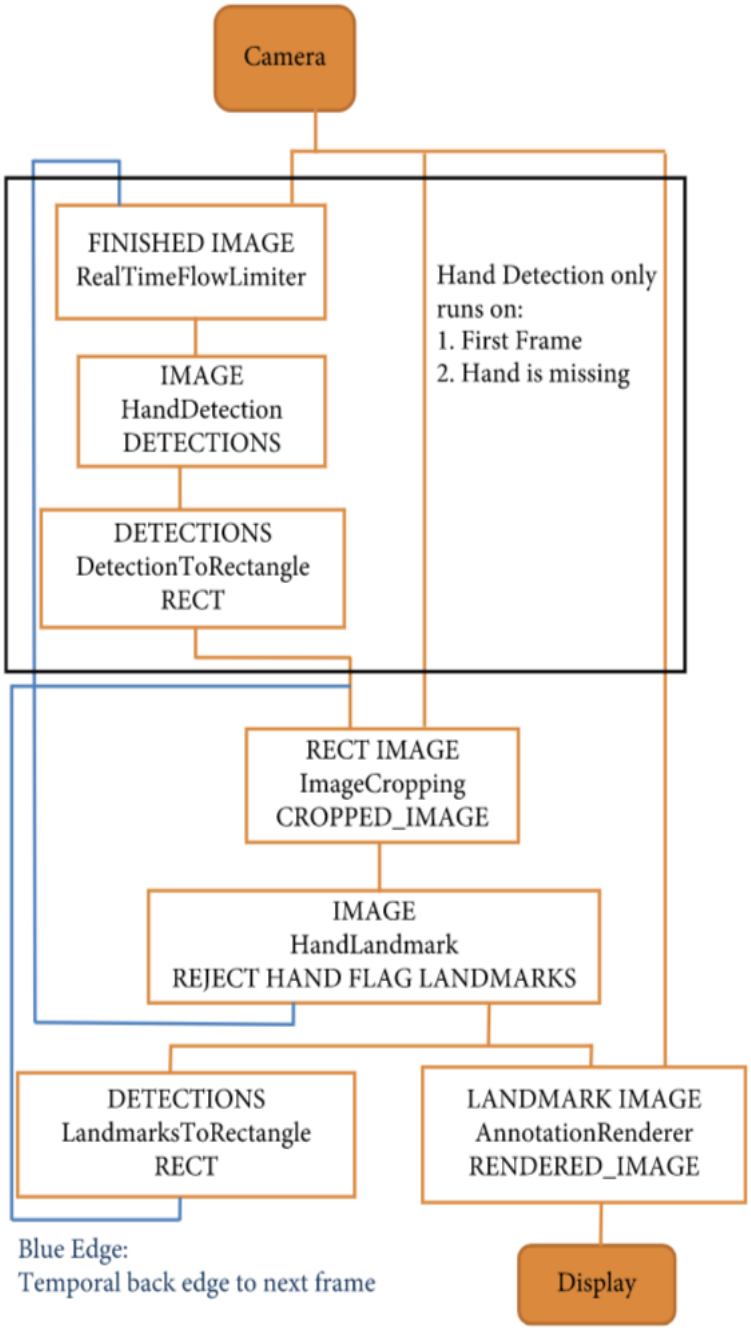
Capturing the Video and Processing. The AI virtual mouse system uses the webcam where each frame is captured till the termination of the program. $e video frames are processed from BGR to RGB color space to find the hands in the video frame by frame.

(Virtual Screen Matching) Rectangular Region for Moving through the Window. The AI virtual mouse system makes use of the transformational algorithm, and it converts the co-ordinates of fingertip from the webcam screen to the computer window full screen for controlling the mouse

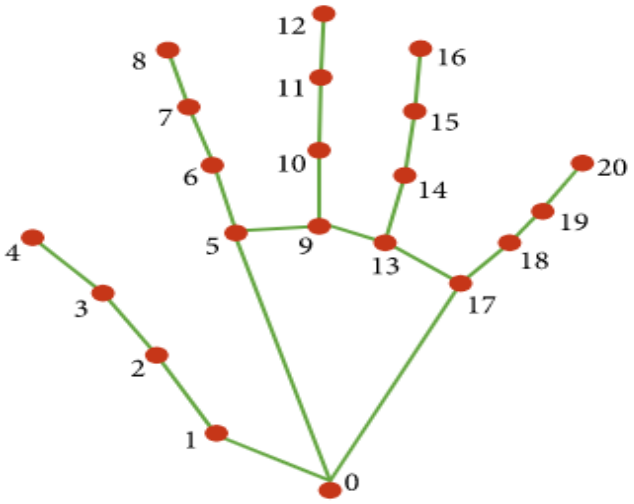
Detecting Which Finger Is Up and Performing the Particular Mouse Function. In this stage, we are detecting which finger is up using the tip Id of the respective finger that we found using the MediaPipe and the respective co-ordinates of the fingers that are up, and according to that, the particular mouse function is performed.

For the Mouse Cursor Moving around the Computer Window. If the index finger is up with tip Id = 1 or both the index finger with tip Id - 1 and the middle finger with tip Id = 2 are up, the mouse cursor is made to move around the window of the computer using the AutoPy package of Python.

## 5.4. Hand detection using mediapipe



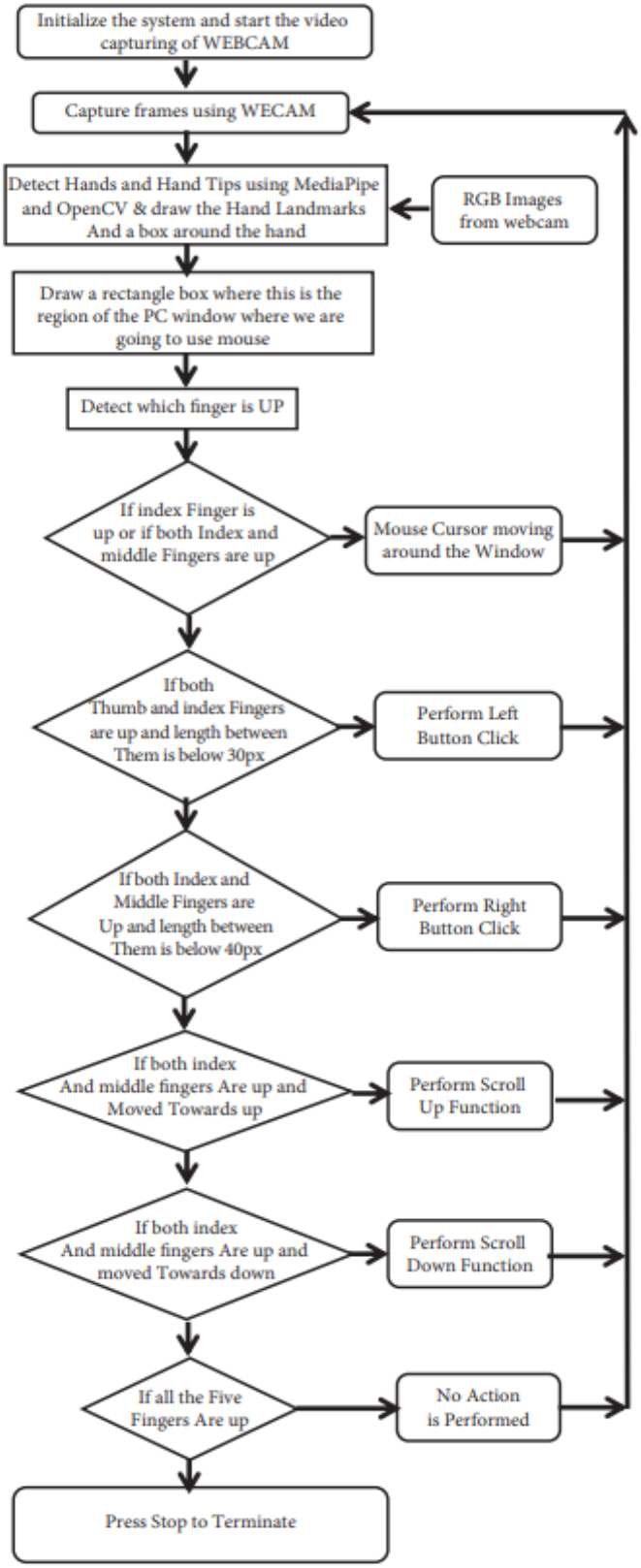

Mediapipe hand recognition graph




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

Co-ordinates or land marks in the hand

## 5.5.Architecture of the system




**Flowchart of the real-time AI Virtual Mouse**

## 5.6.System Requirements

### Hardware Requirements :-

1.Monitor
2.RAM /memory required: 512 mb.
3. Hard disk capacity: 200mb.
4. Web Cam

### Software Requirements :-

1-Operating System: Windows XP or above version.
2-IDE: Pycharm/ Intellij Idea
3- Processor: 32 bit or more.

### Language used-:

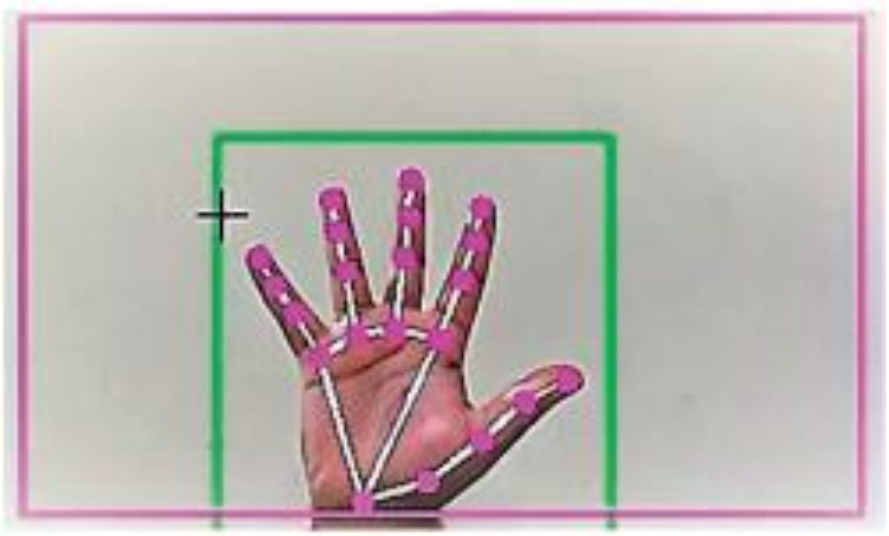
- Python

# Chapter 6

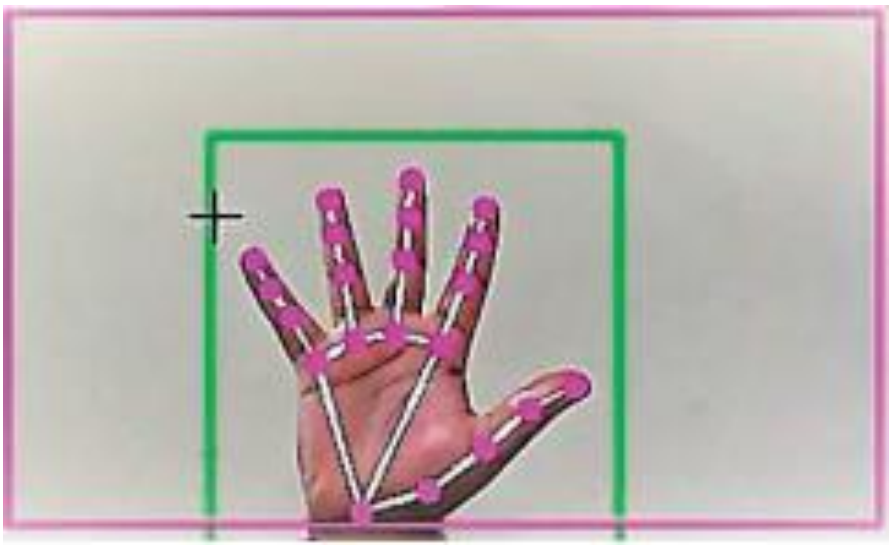
## 6. User Interface and Result

### 6.1. Screenshots of user interface

**1.**Captures video using the webcam (Computer Vision)



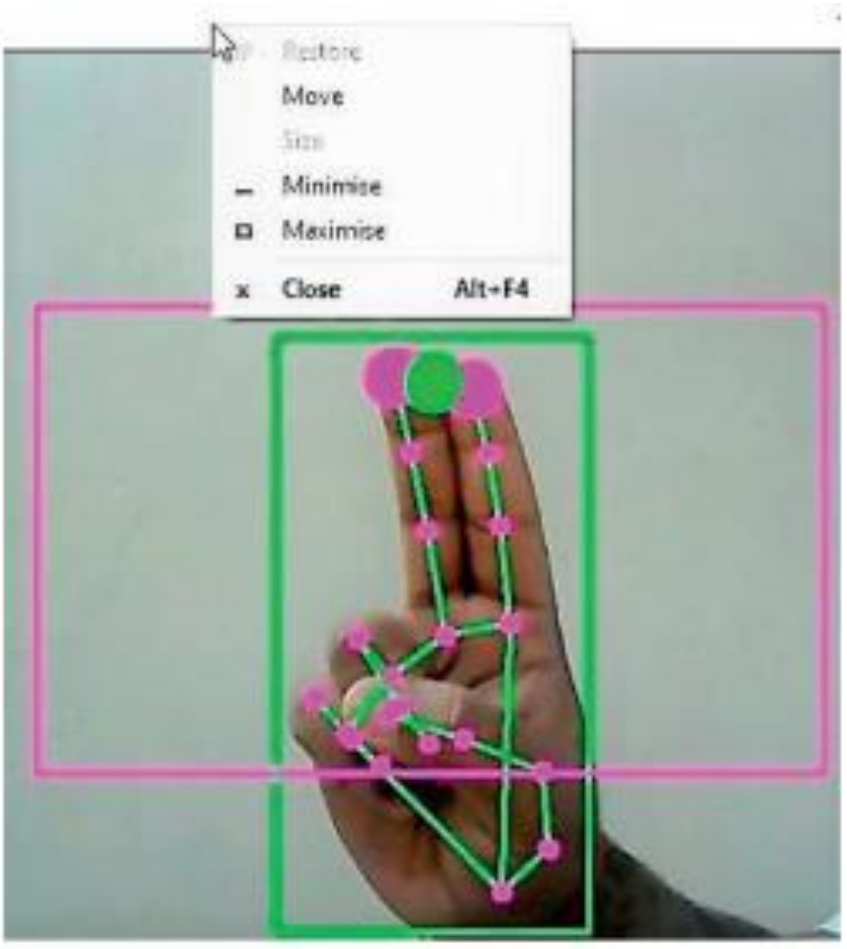
**2.**Rectangular box for the area of the computer screeen where we can move the cursor.



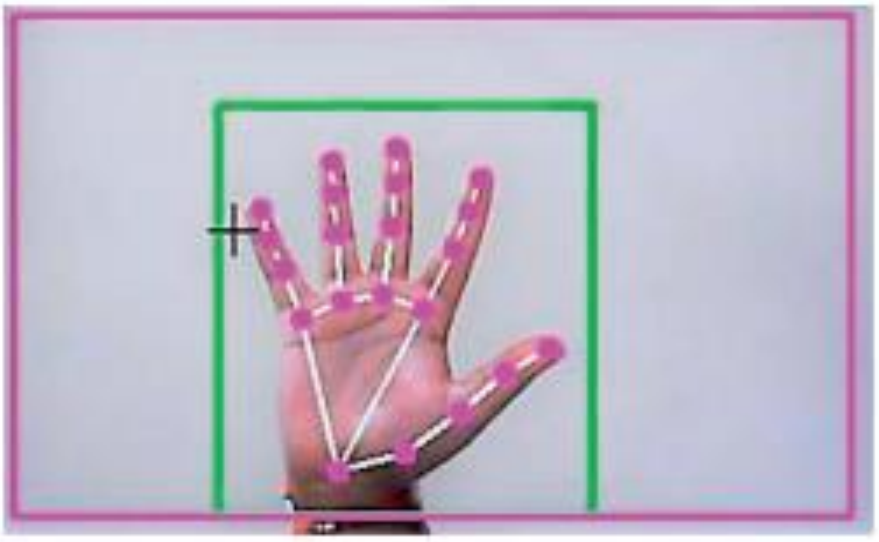
**3.**Detection of which finger is up.

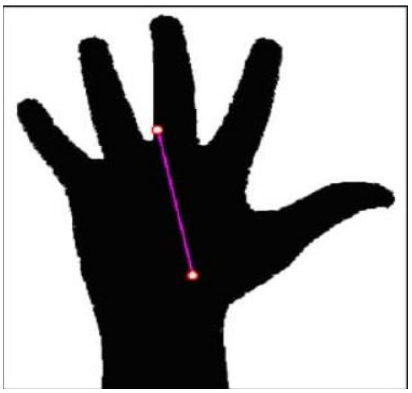
**4.**Mouse cursor moving around the computer window.

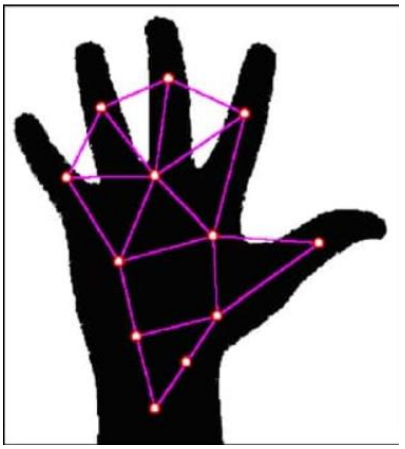
**5.**Gesture for the computer to perform right button click.

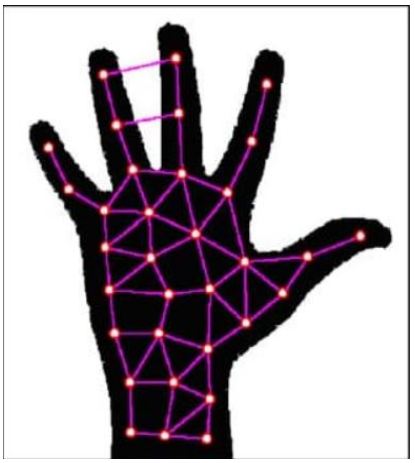


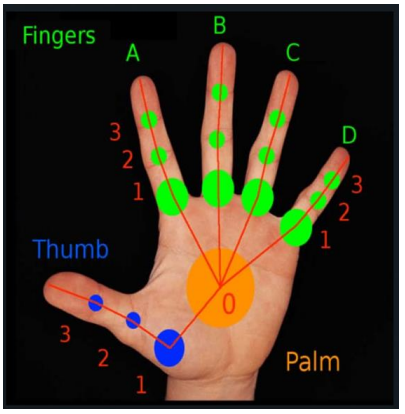

**6.** Gesture for the computer to perform no action.










Fingers
A
B
C
D
3
2
1
3
2
1
Thumb
0
3
2
1
Palm

# Chapter 7

# 7. Conclusion and Future Work

## 7.1. Conclusion

The main objective of the AI virtual mouse system is to control the mouse cursor functions by using the hand gestures instead of using a physical mouse. The proposed system can be achieved by using a webcam or a built-in camera which detects the hand gestures and hand tip and processes these frames to perform the particular mouse functions.

From the results of the model, we can come to a conclusion that the proposed AI virtual mouse system has performed very well and has a greater accuracy compared to the existing models and also the model overcomes most of the limitations of the existing systems. Since the proposed model has greater accuracy, the AI virtual mouse can be used for real-world applications, and also, it can be used to reduce the spread of COVID-19, since the proposed mouse system can be used virtually using hand gestures without using the traditional physical mouse.

The model has some limitations such as small decrease in accuracy in right click mouse function and some difficulties in clicking and dragging to select the text. Hence, we will work next to overcome these limitations by improving the finger tip detection algorithm to produce more accurate results.

## 7.2. 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. $ese are some of the limitations of the proposed AI virtual mouse system, and these limitations will be overcome in our future work.

Furthermore, the proposed method can be developed to handle the keyboard functionalities along with the mouse functionalities virtually which is another future scope of Human-Computer Interaction (HCI).

| Existing models | Accuracy (%) |
|---|---|
| Virtual mouse system using RGB-D images and fingertip detection [16] | 96.13 |
| Palm and finger recognition based [17] | 78 |
| Hand gesture-based virtual mouse [18] | 78 |
| The proposed AI virtual mouse system | 99 |

**Comparison with existig models.**

# References

1] J. Katona, “A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications,” Applied Sciences, vol. 11, no. 6, p. 2646, 2021.

[2] D. L. Quam, “Gesture recognition with a DataGlove,” IEEE Conference on Aerospace and Electronics, vol. 2, pp. 755–760, 1990.

[3] D.-H. Liou, D. Lee, and C.-C. Hsieh, “A real time hand gesture recognition system using motion history image,” in Proceedings of the 2010 2nd International Conference on Signal Processing Systems, July 2010.

[4] S. U. Dudhane, “Cursor control system using hand gesture recognition,” IJARCCE, vol. 2, no. 5, 2013.

[5] K. P. Vinay, “Cursor control using hand gestures,” International Journal of Critical Accounting, vol. 0975–8887, 2016.

[6] L. Thomas, “Virtual mouse using hand gesture,” International Research Journal of Engineering and Technology (IRJET, vol. 5, no. 4, 2018.

# Appendix

## 1.Code for Virtual mouse

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

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

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:
     # Step1: Find the landmarks
     success, img = cap.read()
     img = detector.findHands(img)
     lmList, bbox = detector.findPosition(img)

```

```
        # Step2: Get the tip of the index and middle finger
        if len(lmList) != 0:
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]

            # Step3: Check which fingers are up
            fingers = detector.fingersUp()
            cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
                          (255, 0, 255), 2)

            # Step4: Only Index Finger: Moving Mode
            if fingers[1] == 1 and fingers[2] == 0:

                # Step5: Convert the coordinates
                x3 = np.interp(x1, (frameR, wCam-frameR), (0, wScr))
                y3 = np.interp(y1, (frameR, hCam-frameR), (0, hScr))

                # Step6: Smooth Values
                clocX = plocX + (x3 - plocX) / smoothening
                clocY = plocY + (y3 - plocY) / smoothening

```

```
                # Step7: Move Mouse
                autopy.mouse.move(wScr - clocX, clocY)
                cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
                plocX, plocY = clocX, clocY

            # Step8: Both Index and middle are up: Clicking Mode
            if fingers[1] == 1 and fingers[2] == 1:

                # Step9: Find distance between fingers
                length, img, lineInfo = detector.findDistance(8, 12, img)

                # Step10: Click mouse if distance short
                if length < 40:
                    cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)
                    autopy.mouse.click()

        # Step11: Frame rate
        cTime = time.time()
        fps = 1/(cTime-pTime)
        pTime = cTime
        cv2.putText(img, str(int(fps)), (28, 58), cv2.FONT_HERSHEY_PLAIN, 3, (255, 8, 8), 3)

        # Step12: Display
        cv2.imshow("Image", img)
        cv2.waitKey(1)
```

# 2. Code for hand tracking

```python
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:
```

```python
            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=3):
        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()
```