**VIRTUAL MOUSE**

**A Project report**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

# Submitted by -

Rudresh Raj – 210160223036

# Contents -

1. **Project Description**
2. **Scope of the Project**
3. **Use – Case Diagram**
4. **API Documentation**
5. **Code Documentation**
6. **Coding Conventions**
7. **Conclusion**
8. **Citations**

**PROJECT DESCRIPTION**

## **Project Objectives–**

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 .Currently, wireless mouse or a Bluetooth mouse still uses devices and is not free of devices completely since it uses a battery for power and a dongle to connect it to the PC. In the proposed AI virtual mouse system, this limitation can be overcome by employing webcam or a built-in camera for capturing of hand gestures and hand tip detection using computer vision. The algorithm used in the system makes use of the machine learning algorithm. Based on the hand gestures, the computer can be controlled virtually and can perform dragging, clicking, scrolling functions, and computer cursor function without the use of the physical mouse.

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 with the computer. 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.

Python programming language is used for developing the AI virtual mouse system, and also, OpenCV which is the library for computer vision is used in the AI virtual mouse system. In the proposed AI virtual mouse system, the model makes use of the MediaPipe package for the tracking of the hands and for tracking of the tip of the hands, and also, Pynput, Autopy, and PyAutoGUI packages were used for moving around the window screen of the computer for performing functions such as click, and scrolling functions. The results of the proposed model showed very high accuracy level, and the proposed model can work very well in real-world application with the use of a CPU without the use of a GPU.

**Problem Statement-**

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.

The proposed AI virtual mouse system can be used to overcome problems in the real world such as situations where there is no space to use a physical mouse and also for the persons who have problems in their hands and are not able to control a physical mouse. Also, in situations like COVID-19 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 overcome these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera.

**Aims & Objectives-**

The objective of this project is to develop a Virtual Mouse.

Expected achievements :

* To design to operate with the help of a webcam.
* To convert hand gesture/motion into mouse input that will be set to a particular screen position.
* Two Types of Mouse functions are achieved :

1. Simple Click
2. Scrolling

**SCOPE OF THE PROJECT**

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.

#### **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)The proposed model has a greater accuracy of 99% which is far greater than the that of other proposed models for virtual mouse, and it has many applications.

(ii)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.

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

(iv)2D and 3D images can be drawn using the AI virtual system using the hand gestures

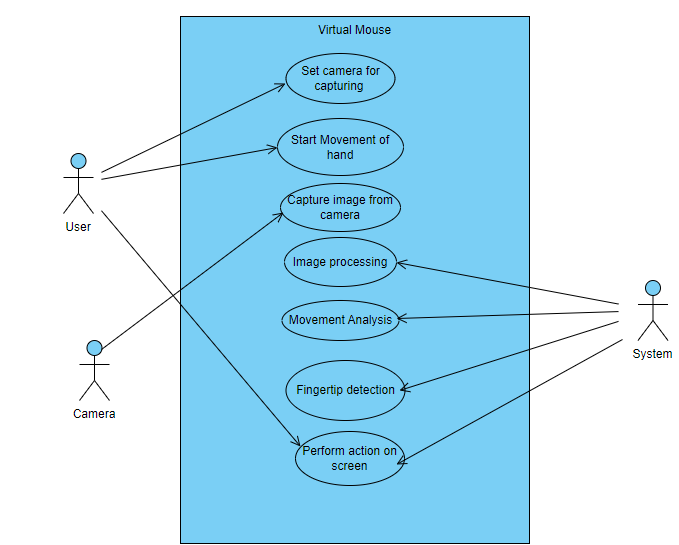
(v)AI virtual mouse can be used to play virtual reality- and augmented reality-based games without the wireless or wired mouse devices.

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

(vii)In the field of robotics, the proposed system like HCI can be used for controlling robots.

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

**USE CASE DIAGRAM**

****

**PSUDOCODE:**

Initializing the system and start the video capturing of WEBCAM

Capture frame using WEBCAM

Detect Hands and hand tips using MediaPipe and OpenCV and draw the Hand Landmarks and coloured circles over the :

THUMB : landmark 4 (Orchid4 RGB CODE : (139,71,137))

INDEX FINGER : landmark 8 (YELLOW RGB CODE : (0, 255, 255))

RING FINGER : landmark 16 (RED RGB CODE : (205,55,0))

LITTLE FINGER : landmark 20 (GREEN RGB CODE : (0,255,0))

MOVE THE MOUSE CURSER WITH THE INDEX FINGER.

DETECT THE INDEX FINGER ON THE SCREEN.

DETECT IF THUMB AND INDEX FINGER IS CLOSER THAN 40 pixels

PERFORM A CLICK.

PERFORM UP SCROLLING.

DETECT IF THUMB AND RING FINGER IS CLOSER THAN 70 pixels

PERFORM DOWN SCROLLING.

DETECT IF THUMB AND LITTLE FINGER IS CLOSER THAN 70 pixels

**API DOCUMENTATTION**

**Brief introduction about Python and it’s installation instructions -**

Python is a popular high-level programming language that is widely used for web development, data analysis, artificial intelligence, scientific computing, and many other applications. It is known for its simple syntax, ease of learning, and vast collection of libraries and frameworks.

Python can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install Python on a Windows computer:

1. Visit the official Python website at python.org.
2. Click on the "Downloads" link and select the appropriate version of Python for the operating system. For example, in Windows 10, we should download the latest version of Python 3.X.
3. Run the installer and follow the prompts to install Python
4. During the installation, we will be prompted to add Python to the system PATH. We should be sure to select this option for easily access Python from the command line.

Once the installation is complete, open a command prompt and type "python" to verify that Python is installed correctly. We should see the Python interpreter start up and display its version information.

**Packages required for the project-**

* **Open CV Library:**

OpenCV is used in the making of this program. OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision like image and video processing . OpenCV have the utility that can read image pixels value, it also have the ability to create real time eye tracking, blink detection, face recognition and augmented reality. It supports various programming languages, including Python, C++, and Java.

* **NumPy** :

NumPy is a Python library that provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It is widely used for numerical computation, data analysis, and scientific computing.

* **Mediapipe :**

MediaPipe is a cross-platform pipeline framework to build custom machine learning solutions for live and streaming media. Using MediaPipe, such a perception pipeline can be built as a graph of modular components. MediaPipe was built for machine learning (ML) teams and software developers who implement production-ready ML applications, or students and researchers who publish code and prototypes as part of their research work.

* **Pyautogui :**

PyAutoGUI is a Python library that provides a simple and cross-platform way to automate GUI tasks. It is a popular library that is widely used for tasks such as GUI testing, automating repetitive tasks, and generating human-like input. PyAutoGUI is a very powerful library that can simulate mouse and keyboard actions, take screenshots, and perform various other GUI-related tasks.

PyAutoGUI has a simple and easy-to-use interface that makes it a popular choice for automating GUI tasks. It provides functions for controlling the mouse, keyboard, and other input devices. PyAutoGUI can simulate mouse clicks, double-clicks, and drags. It can also perform keyboard input, such as typing text, pressing keys, and sending hotkeys. Additionally, PyAutoGUI can take screenshots of the screen, locate and identify GUI elements, and control the window focus.

PyAutoGUI is a cross-platform library that works on Windows, Mac, and Linux. It can be used with various programming languages that have a Python API, **such** as Java, Ruby, and C++. PyAutoGUI is also compatible with Python 2 and 3.

In the Virtual Mouse project, PyAutoGUI is used to control the mouse cursor and perform mouse-related actions. For example, PyAutoGUI is used to simulate mouse clicks, drag and drop operations, and scrolling. PyAutoGUI provides a reliable and easy-to-use interface for controlling the mouse cursor and performing GUI-related tasks.

Overall, PyAutoGUI is a powerful and versatile Python library that provides a simple and cross-platform way to automate GUI tasks. It is a valuable tool for anyone who needs to automate repetitive GUI tasks, generate human-like input, or perform GUI testing.

**Code Documentation:**

COMPLETE CODE : VIRTUAL MOUSE - The virtual mouse code tracks the position of the hand using computer vision techniques and maps it to the position of the mouse cursor, enabling the user to control the mouse without a physical device WITH TWO FUNCTIONS – click and scrolling.

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

*# to know the size of the computer screen*

screen\_width, screen\_height = pyautogui.size()

index\_y = 0

while True:

    \_, frame = cap.read()

*# because the screen was originally opposite, this flips it back*

    frame = cv2.flip(frame, 1)

    frame\_height, frame\_width, \_ = frame.shape

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

        for hand in hands:

*# <---- for seeing the landmarks on the frame*

            drawing\_utils.draw\_landmarks(frame, hand)

            landmarks = hand.landmark

            for id, landmark in enumerate(landmarks):

                x = int(landmark.x\*frame\_width)

                y = int(landmark.y\*frame\_height)

*# print(x, y)*

*# INDEX FINGER*

                if id == 8:  *# as position of index finger is 8*

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(0, 255, 255))

*# this makes a yellow circle on the index finger only*

                    index\_x = screen\_width/frame\_width\*x

                    index\_y = screen\_height/frame\_height\*y

*# <--- this helps the curser move in the whole computer window*

                    pyautogui.moveTo(index\_x, index\_y)

*# pyautogui.moveTo(x, y) # but the curser only moves in the frame not the whole window like we want <--- so just above we wrote code for moving the curser on whole window*

*# MAKING THE THUMB SEPERATE TOO SO THAT WE CAN MAKE A GESTURE AND CREATE A CLICKING OPTION*

*# CLICK FUNCTION*

*# Thumb and Index finger comes close for click*

*# THUMB*

                if id == 4:  *# as position of Thumb is 4*

                    cv2.circle(img=frame, center=(x, y), radius=10, color=(139,71,137))

*# this makes a yellow circle on the index finger only*

                    thumb\_x = screen\_width/frame\_width\*x

                    thumb\_y = screen\_height/frame\_height\*y

                    print('outside', abs(index\_y - thumb\_y))

*# adding a click function when the thumb and the index finger comes closer than 40 pixels*

                    if abs(index\_y - thumb\_y) < 50:

                        print('click')

                        pyautogui.click() *# click function*

                        pyautogui.sleep(1)

*# SCROll FUNCTION*

*# SCROLLING UP*

*# Little finger and thumb comes close for scrolling up*

*# Little Finger*

                if id == 20:  *# as position of Little finger is 8*

                    cv2.circle(img=frame, center=(x, y),radius=10, color=(0,255,0))

*# this makes a yellow circle on the index finger only*

                    Little\_x = screen\_width/frame\_width\*x

                    Little\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - Little\_y) < 70:

                        print('SCROLLING UP')

                        pyautogui.scroll(20) *# Scroll function*

*# SCROLLING DOWN*

*# THE THUMB AND RING FINGER COMES CLOSE FOR SCROLLING DOWN*

*#RING FINGER*

                if id == 16:  *# as position of Ring Finger is 16*

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(205,55,0))

*# this makes a yellow circle on the index finger only*

                    ring\_x = screen\_width/frame\_width\*x

                    ring\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - ring\_y) < 70:

                        print('SCROLLING DOWN')

                        pyautogui.scroll(-20) *# Scroll function*

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

Explaining the code in 5 STEPS :

STEP 1: OPENING THE CAMERA

import cv2

cap = cv2.VideoCapture(0)

while True:

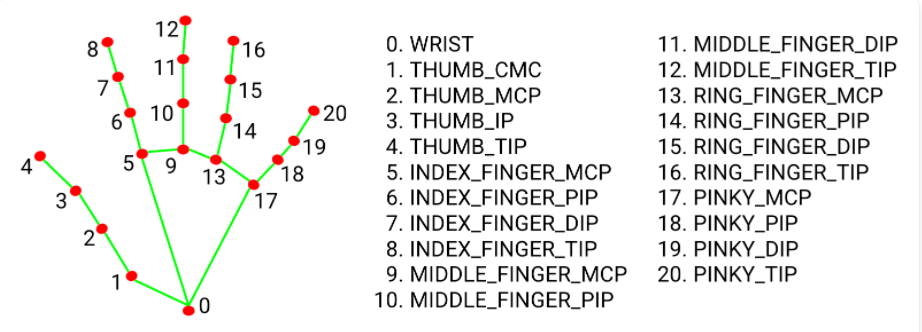
    \_, frame = cap.read()

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

STEP 2 : TO DETECT THE HAND IN THE CAMERA

For this we use the library “MediaPipe”, and we mark out the hand landmarks



import cv2

import mediapipe as mp

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

while True:

    \_, frame = cap.read()

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    print(hands)

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

## **for showing the hand with all the Landmarks Of the Hand**

import cv2

import mediapipe as mp

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

while True:

    \_, frame = cap.read()

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

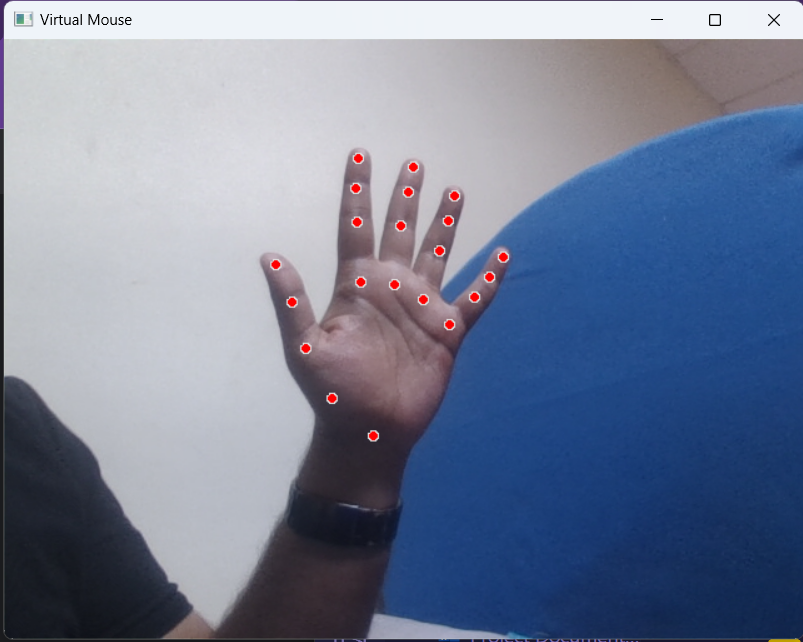
        for hands in hands:

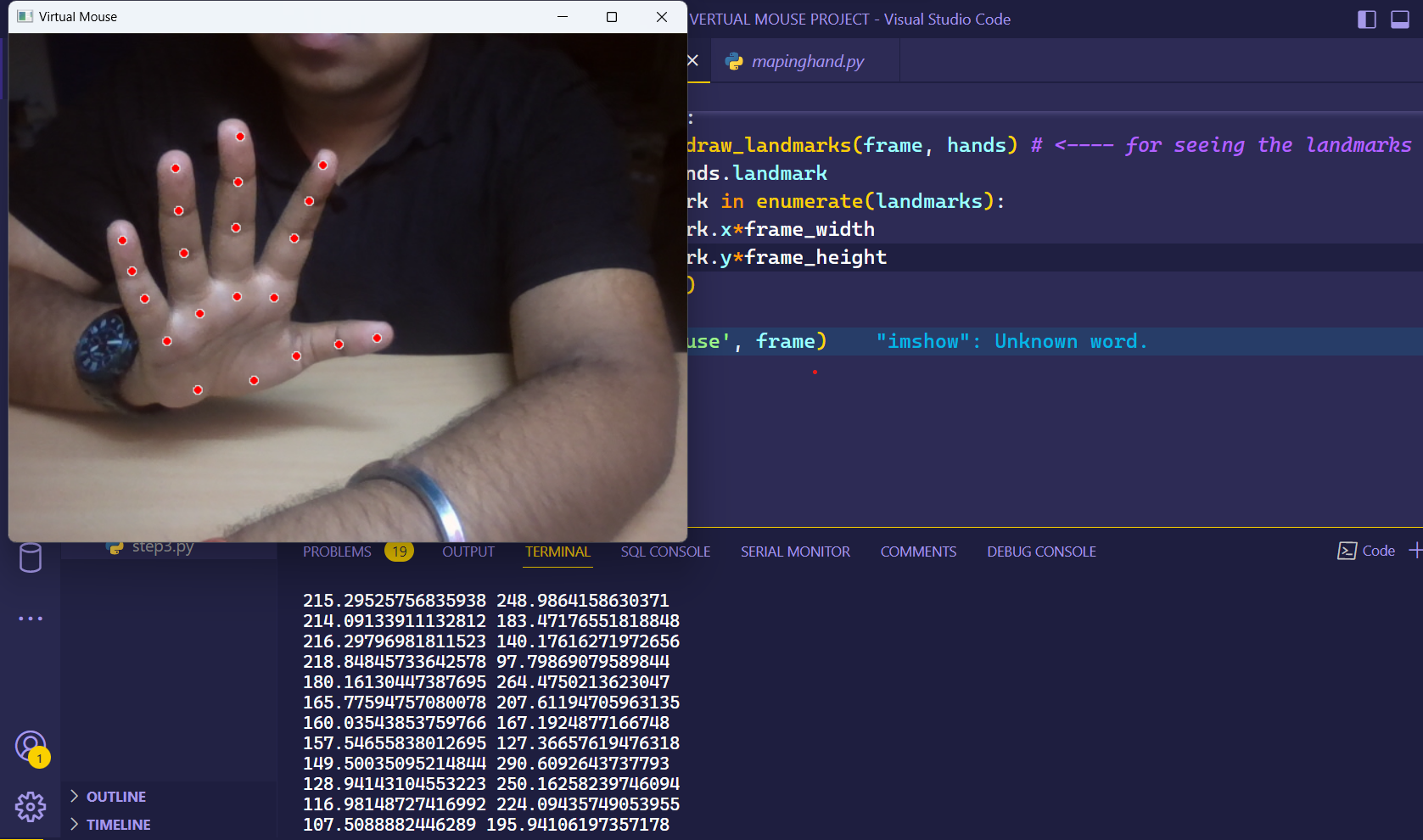
            drawing\_utils.draw\_landmarks(frame, hands) *# <---- for seeing the landmarks on the frame*

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

MAPPING THE HAND :-





STEP 3 : DETECTING THE INDEX FINGER INDIVIDUALLY

import cv2

import mediapipe as mp

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

while True:

    \_, frame = cap.read()

    frame = cv2.flip(frame, 1) *# because the screen was originally opposite, this flips it back*

    frame\_height, frame\_width, \_ = frame.shape

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

        for hands in hands:

            drawing\_utils.draw\_landmarks(frame, hands) *# <---- for seeing the landmarks on the frame*

            landmarks = hands.landmark

            for id, landmark in enumerate(landmarks):

                x = int(landmark.x\*frame\_width)

                y = int(landmark.y\*frame\_height)

                print(x, y)

                if id == 8: *# as position of index finger is 8*

                    cv2.circle(img=frame, center=(x,y), radius=10, color=(0, 255, 255))

*# this makes a yellow circle on the index finger only*

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

STEP 4 : MOVING THE MOUSE POINTER USING INDEX FINGER

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

screen\_width, screen\_height = pyautogui.size() *# to know the size of the computer screen*

while True:

    \_, frame = cap.read()

    frame = cv2.flip(frame, 1) *# because the screen was originally opposite, this flips it back*

    frame\_height, frame\_width, \_ = frame.shape

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

        for hands in hands:

            drawing\_utils.draw\_landmarks(frame, hands) *# <---- for seeing the landmarks on the frame*

            landmarks = hands.landmark

            for id, landmark in enumerate(landmarks):

                x = int(landmark.x\*frame\_width)

                y = int(landmark.y\*frame\_height)

                print(x, y)

                if id == 8: *# as position of index finger is 8*

                    cv2.circle(img=frame, center=(x,y), radius=10, color=(0, 255, 255))

*# this makes a yellow circle on the index finger only*

                    index\_x = screen\_width/frame\_width\*x

                    index\_y = screen\_height/frame\_height\*y

                    pyautogui.moveTo(index\_x, index\_y) *# <--- this helps the curser move in the whole computer window*

*# pyautogui.moveTo(x, y) # but the curser only moves in the frame not the whole window like we want <--- so just above we wrote code for moving the curser on whole window*

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

STEP 5 : INTRODUCING CLICK FEATURE

* When the index finger and thumb comes closer than 50 pixels the mouse clicks.
* We separated both the thumb and the index figure from the Hand Landmark, we also marked index finger with yellow circle and thumb with purple(Orchid4) circle, so that it is differentiable from one another.

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

# **to know the size of the computer screen**

screen\_width, screen\_height = pyautogui.size()

index\_y = 0

while True:

    \_, frame = cap.read()

    # because the screen was originally opposite, this flips it back

    frame = cv2.flip(frame, 1)

    frame\_height, frame\_width, \_ = frame.shape

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

        for hand in hands:

            # <---- for seeing the landmarks on the frame

            drawing\_utils.draw\_landmarks(frame, hand)

            landmarks = hand.landmark

            for id, landmark in enumerate(landmarks):

                x = int(landmark.x\*frame\_width)

                y = int(landmark.y\*frame\_height)

                # print(x, y)

                if id == 8:  # as position of index finger is 8

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(0, 255, 255))

                    # this makes a yellow circle on the index finger only

                    index\_x = screen\_width/frame\_width\*x

                    index\_y = screen\_height/frame\_height\*y

                    # <--- this helps the curser move in the whole computer window

                    pyautogui.moveTo(index\_x, index\_y)

                    # pyautogui.moveTo(x, y) # but the curser only moves in the frame not the whole window like we want <--- so just above we wrote code for moving the curser on whole window

                    # MAKING THE THUMB SEPERATE TOO SO THAT WE CAN MAKE A GESTURE AND CREATE A CLICKING OPTION

                if id == 4:  # as position of Thumb is 4

                    cv2.circle(img=frame, center=(x, y), radius=10, color=(139,71,137))

                    # this makes a yellow circle on the index finger only

                    thumb\_x = screen\_width/frame\_width\*x

                    thumb\_y = screen\_height/frame\_height\*y

                    print('outside', abs(index\_y - thumb\_y))

                    # adding a click function when the thumb and the index finger comes closer than 40 pixels

                    if abs(index\_y - thumb\_y) < 40:

                        print('click')

                        pyautogui.click() # click function

                        pyautogui.sleep(1)

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

STEP 6 : INTRODUCING THE SCROLL FEATURE

THIS WILL INCLUDE TWO FEATURES

              1. SCROLLING UP

              2. SCROLLING DOWN

1. SCROLLING UP ---> IF THUMB AND RING FINGER COMES CLOSER THAN 70 pixels screen scrolls up

            WE SEPARATE THE RING FINGER TOO WITH BLUE COLOR (RGB CODE : 0,255,0)

#SCROll FUNCTION

 #SCROLLING UP

 #Little finger and thumb comes close for scrolling up

                    # Little Finger

                if id == 20:  # as position of Little finger is 8

                    cv2.circle(img=frame, center=(x, y),radius=10, color=(0,255,0))

                    # this makes a yellow circle on the index finger only

                    Little\_x = screen\_width/frame\_width\*x

                    Little\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - Little\_y) < 70:

                        print('SCROLLING UP')

                        pyautogui.scroll(20) # Scroll function

2. SCROLLING DOWN ---> IF THUMB AND LITTLE FINGER COMES CLOSER THAN 70 pixels screen scrolls DOWN

            WE SEPERATE THE LITTLE FINGER TOO WITH GREEN COLOR RGB CODE : (RGB CODE : 205,55,0)

    #SCROLLING DOWN

    #THE THUMB AND RING FINGER COMES CLOSE FOR SCROLLING DOWN

                    #RING FINGER

                if id == 16:  # as position of Ring Finger is 16

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(205,55,0))

                    # this makes a yellow circle on the index finger only

                    ring\_x = screen\_width/frame\_width\*x

                    ring\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - ring\_y) < 70:

                        print('SCROLLING DOWN')

                        pyautogui.scroll(-20) # Scroll function

COMPLETE FINAL WORKING CODE :

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands()

drawing\_utils = mp.solutions.drawing\_utils

*# to know the size of the computer screen*

screen\_width, screen\_height = pyautogui.size()

index\_y = 0

while True:

    \_, frame = cap.read()

*# because the screen was originally opposite, this flips it back*

    frame = cv2.flip(frame, 1)

    frame\_height, frame\_width, \_ = frame.shape

    rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    output = hand\_detector.process(rgb\_frame)

    hands = output.multi\_hand\_landmarks

    if hands:

        for hand in hands:

*# <---- for seeing the landmarks on the frame*

            drawing\_utils.draw\_landmarks(frame, hand)

            landmarks = hand.landmark

            for id, landmark in enumerate(landmarks):

                x = int(landmark.x\*frame\_width)

                y = int(landmark.y\*frame\_height)

*# print(x, y)*

*# INDEX FINGER*

                if id == 8:  *# as position of index finger is 8*

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(0, 255, 255))

*# this makes a yellow circle on the index finger only*

                    index\_x = screen\_width/frame\_width\*x

                    index\_y = screen\_height/frame\_height\*y

*# <--- this helps the curser move in the whole computer window*

                    pyautogui.moveTo(index\_x, index\_y)

*# pyautogui.moveTo(x, y) # but the curser only moves in the frame not the whole window like we want <--- so just above we wrote code for moving the curser on whole window*

*# MAKING THE THUMB SEPERATE TOO SO THAT WE CAN MAKE A GESTURE AND CREATE A CLICKING OPTION*

*# CLICK FUNCTION*

*# Thumb and Index finger comes close for click*

*# THUMB*

                if id == 4:  *# as position of Thumb is 4*

                    cv2.circle(img=frame, center=(x, y), radius=10, color=(139,71,137))

*# this makes a yellow circle on the index finger only*

                    thumb\_x = screen\_width/frame\_width\*x

                    thumb\_y = screen\_height/frame\_height\*y

                    print('outside', abs(index\_y - thumb\_y))

*# adding a click function when the thumb and the index finger comes closer than 40 pixels*

                    if abs(index\_y - thumb\_y) < 50:

                        print('click')

                        pyautogui.click() *# click function*

                        pyautogui.sleep(1)

*# SCROll FUNCTION*

*# SCROLLING UP*

*# Little finger and thumb comes close for scrolling up*

*# Little Finger*

                if id == 20:  *# as position of Little finger is 8*

                    cv2.circle(img=frame, center=(x, y),radius=10, color=(0,255,0))

*# this makes a yellow circle on the index finger only*

                    Little\_x = screen\_width/frame\_width\*x

                    Little\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - Little\_y) < 70:

                        print('SCROLLING UP')

                        pyautogui.scroll(20) *# Scroll function*

*# SCROLLING DOWN*

*# THE THUMB AND RING FINGER COMES CLOSE FOR SCROLLING DOWN*

*#RING FINGER*

                if id == 16:  *# as position of Ring Finger is 16*

                    cv2.circle(img=frame, center=(x, y),

                               radius=10, color=(205,55,0))

*# this makes a yellow circle on the index finger only*

                    ring\_x = screen\_width/frame\_width\*x

                    ring\_y = screen\_height/frame\_height\*y

                    if abs(thumb\_y - ring\_y) < 70:

                        print('SCROLLING DOWN')

                        pyautogui.scroll(-20) *# Scroll function*

    cv2.imshow('Virtual Mouse', frame)

    cv2.waitKey(1)

**Coding Conventions:**

The coding conventions used for the project are as follows:

a. Variable names should be in lowercase and words should be separated by an underscore.

b. Constants should be in uppercase and words should be separated by an underscore.

c. Function names should be in lowercase and words should be separated by an underscore.

d. Indentation should be done using four spaces.

e. Comments should be used to explain the purpose of the code.

**CITATIONS**

1. “Virtual Mouse Implementation using Color Pointer Detection” by D.S. Suresh and I.V. Bhavana.
2. “A virtual mouse interface with a two layered Bayesian network” by S.W. Lee, D. Kang, S. Huh, M.C. Roh.
3. D. Lee, D.-H. Liou, and C.-C. Hsieh, “A real time hand gesture recognition system using motion history image,” in Proceedings of the 2nd International Conference on Signal Processing Systems, July 2010.
4. “Cursor system using hand gesture recognition,” in IJARCCE by S. U. Dudhane
5. Izaman Khan and Adnan Ibraheem Rafiqu, “Survey on Various Gesture Recognition Technologies and Techniques” in International Journal of Computer Applications Volume50 – No.7, July 2012.