Chapter 1

Introduction

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 paper 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. 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. 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.

While using a wireless or a Bluetooth mouse, some devices such as the mouse, the dongle to connect to the PC, and also, a battery to power the mouse to operate are used, but in this paper, the user uses his/her built-in camera or a webcam and uses his/her hand gestures to control the computer mouse operations. In the proposed system, the web camera captures and then processes the frames that have been captured and then recognizes the various hand gestures and hand tip gestures and then performs the particular mouse function.

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 left click, right 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.

One of the simplest, proficient, and significant ways of human communication is through hand gestures that people tend to make use of even unknowingly and a universally accepted language. In this paper, an online hand gesture mouse controlling system is proposed. In our setup of the system, a fixed integrated laptop camera is used to keep the cost at a minimum to implement the system in a cheap way affordable to be used by everyone without the need for any additional equipment that may incur a cost to the system and make it expensive, the camera system helps by capturing snapshots of the user object under a movement that is to be tracked using Red Green Blue [RGB] color space from a fixed distance and converted to HSV scheme for use by the system. Under this project, a productive hand gesture segmentation technique has been recommended based on the pre-processing, background subtraction, and edge detection techniques. Initially, before making use of the system, the color objects or color caps that will be used for tracking should be identified. Also, a threshold on the system should be set accordingly to detect these colored objects and store these thresholds for the system to make use of it while tracking. A range system has been initialized to set the HSV min, max values, and stored for use while tracking and performing operations. This is a significant part of the system as it may fail to detect objects and perform tracking if the range is not set correctly. This system typically performs all operations that could be performed by a traditional mouse pointer. Operations like left-click, right-click, and hand recognition could easily perform other hand gesture operations like dragging a file to

another location. The main aim is to create a cost-free hand recognition software for laptops and PCs with external webcams.

1.1 Problem Description and Overview

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, amidst of 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 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.

1.2 Objective

The main objective of the proposed AI virtual mouse system is to develop an alternative to the regular and traditional mouse system to perform and control the mouse functions, and this can be achieved with the help of a web camera that captures the hand gestures and hand tip and then processes these frames to perform the particular mouse function such as left click, right click, and scrolling function.

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM

The existing system consists of the generic mouse and trackpad system of monitor controlling and the nonavailability of a hand gesture system. The remote accessing of monitor screen using the hand gesture is unavailable. Even-though it is largely trying to implement the scope is simply restricted in the field of virtual mouse. The existing virtual mouse control system consists of the simple mouse operations using the hand recognition system, where we could perform the basic mouse operation like mouse pointer control, left click, right click, drag etc. The further use of the hand recognition is not been made use of. Even-though there are a number of systems which are used for hand recognition, the system they made used is the static hand recognition which is simply recognition of the shape made by hand and by defining an action for each shape made, which is limited to a number of defined actions and a large amount of confusion.

2.2 PROPOSED SYSTEM

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 actions along with an action for the file transfer between connected systems by hand action and the

movements. 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.

2.3 USE OF PROPOSED WORK

This Virtual Mouse Hand Recognition application uses a simple color cap on the finger without the additional requirement of the hardware for the controlling of the cursor using simple gestures and hand control. This is done using vision based hand gesture recognition with inputs from a webcam.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 SOFTWARE SPECIFICATION

The software specification are the specification of the system. It should include both the specification and a definition of the requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide the basis for creating the software requirement specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development.

REQUIREMENTS

- Language : Python
- Platform : OpenCV
- Tool : Deep Leaning toolbox

3.2 HARDWARE SPECIFICATION

The Hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system do not and how it should be implemented.

REQUIREMENTS

- Processor : Intel® core ™ I 3-4030Ucpu@1.90GHZ
- RAM : 4 GB or More
- System type : 64-bit operating system
- Keyboard : Normal or Multimedia

CHAPTER 4

SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM:

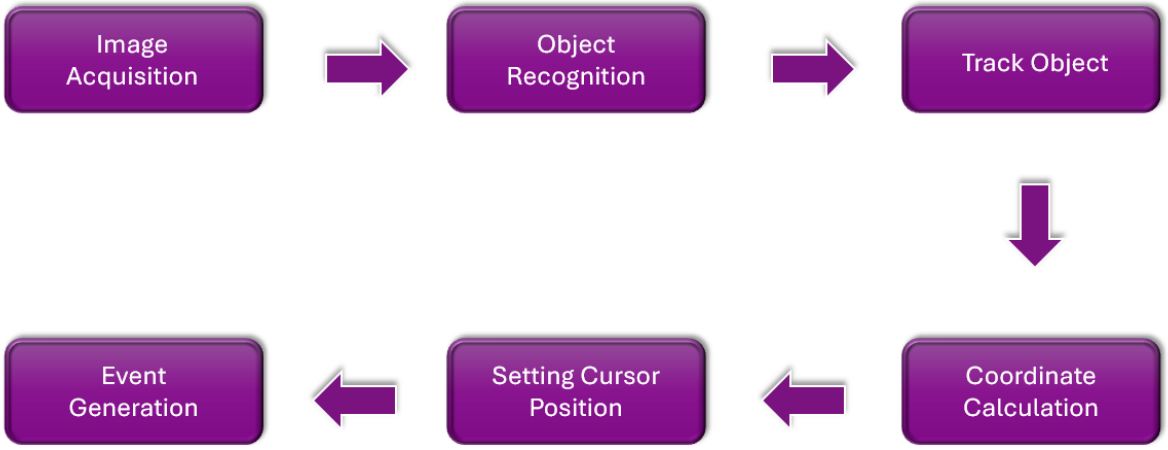

Fig: 4.1 Architecture

4.2 UML Diagrams:

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. It is based on diagrammatic representations of software components.

4.2.1 USE CASE DIAGRAM

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a “system” is something being developed or operated, such as a web site. The “actors” are people or entities operating under defined roles within the system. Use case diagrams are valuable for

visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration. They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

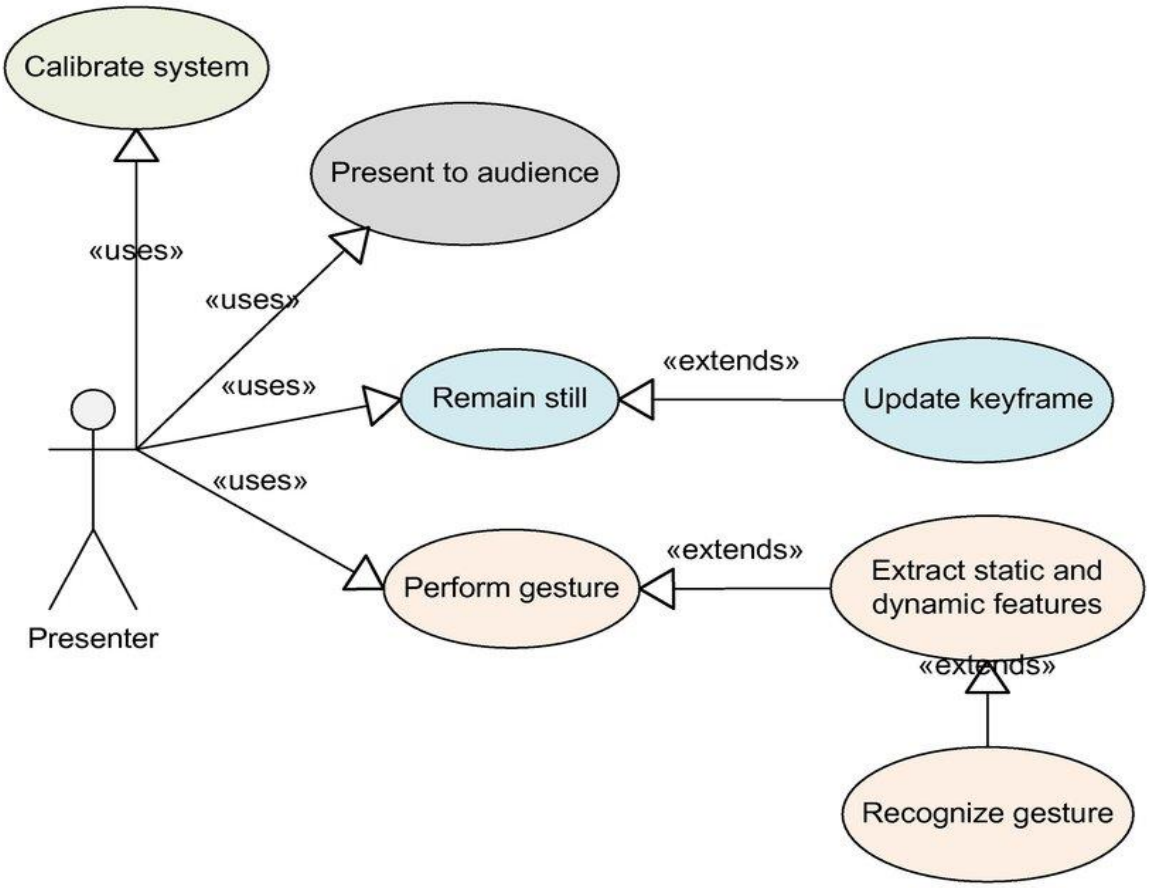

Fig: 4.2.1 Use Case Diagram

4.2.2 CLASS DIAGRAM

A Class diagram is a Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure

of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

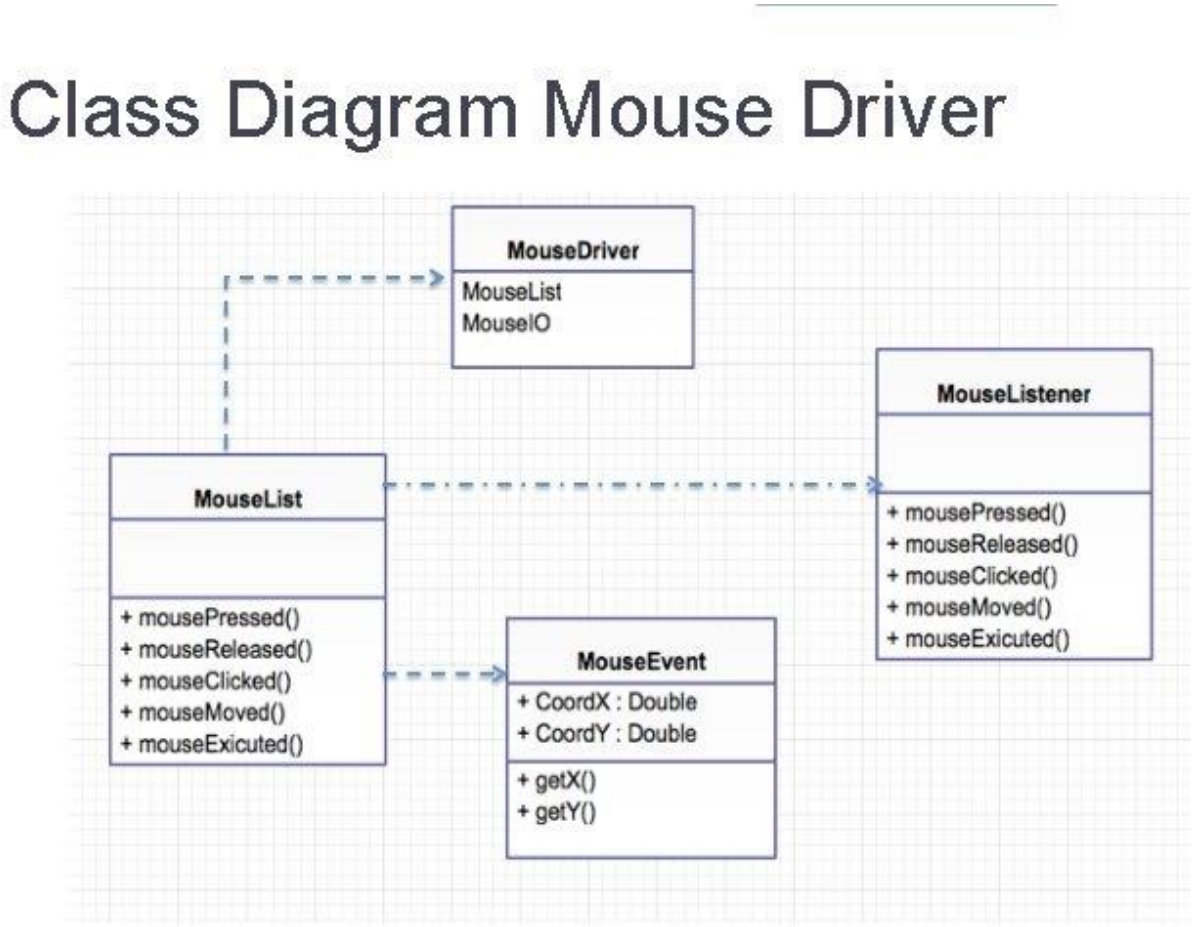

Fig: 4.2.2 Class Diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Algorithm Used for Hand Tracking

For the purpose of detection of hand gestures and hand tracking, the MediaPipe framework is used, and OpenCV library is used for computer vision [7–10]. Algorithm makes use of the machine learning concepts to track and recognize the hand gestures and hand tip.

5.2 MediaPipe

MediaPipe is a framework which is used for applying in a machine learning pipeline, and it is an opensource framework of Google. The MediaPipe framework is useful for cross platform development since the framework is built using the time series data. The MediaPipe framework is multimodal, where this framework can be applied to various audios and videos. The MediaPipe framework is used by the developer for building and analyzing the systems through graphs, and it also been used for developing the systems for the application purpose. The steps involved in the system that uses MediaPipe are carried out in the pipeline configuration. The pipeline created can run in various platforms allowing scalability in mobile and desktops. The MediaPipe framework is based on three fundamental parts; they are performance evaluation, framework for retrieving sensor data, and a collection of components which are called calculators, and they are reusable. A pipeline is a graph which consists of components called calculators, where each calculator is connected by streams in which the packets of data flow through. Developers are able to replace or define custom calculators anywhere in the graph creating their own application. The calculators and streams combined create a data-flow diagram; the graph is created with MediaPipe where each node is a calculator and the nodes are connected by streams.

Single-shot detector model is used for detecting and recognizing a hand or palm in real time. $e single-shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the non maximum suppression works significantly better on small objects such as palms or fists . A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region.

5.3 Open CV

OpenCV is a computer vision library which contains image-processing algorithms for object detection [14]. OpenCV is a library of python programming language, and real-time computer vision applications can be developed by using the computer vision library. The OpenCV library is used in image and video processing and also analysis such as face detection and object detection

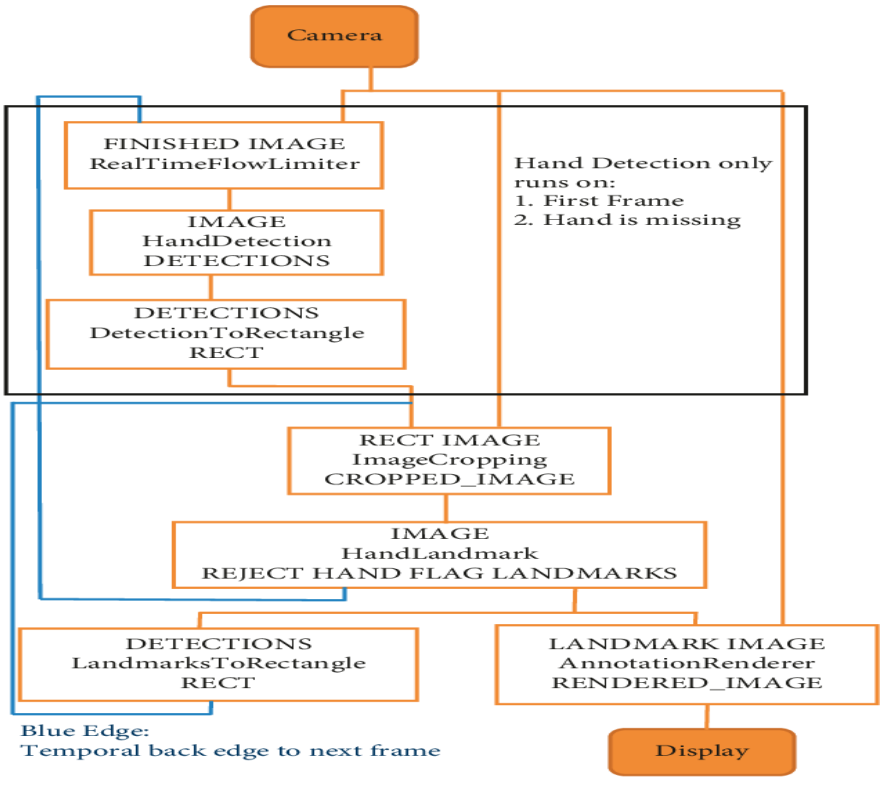

Fig: 5.3 MediaPipe hand recognition graph

CHAPTER 6

Methodology

In the Methodology, the method used in each component of the system will be explained separately. They are following subsections:

6.1 Camera Settings

The runtime operations are managed by the webcam of the connected laptop or desktop. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera.Since we only use a single camera we pass it as ‘0’. We can add additional camera to the system and pass it as 1,2 and so on. After that, you can capture frame-by-frame. But at the end, don’t forget to release the capture. We could also apply color detection techniques to any image by doing simple modifications in the code.

6.2 Capturing frames

The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program. We capture the live feed stream, frame by frame. Then we process each captured frame which is in RGB(default) color space to HSV color space. There are more than 150 color-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV.

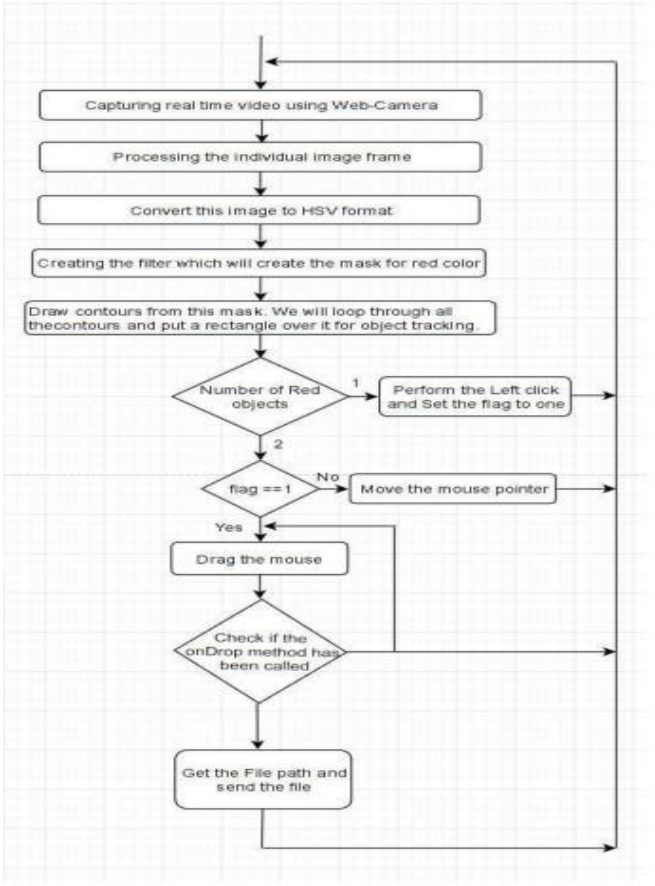

Fig: 6.2 Flow Chart

6.3 Masking Technique

The mask is basically creating some specific region of the image following certain rules. Here we are creating a mask that comprises of an object in red color. After that we perform a bitwise AND operation on the Input image and threshold image, which result in only the red colored objects are highlighted. This result of the AND operation is stored in res. We then display the frame, res and mask on 3 separate windows using imshow() function.

6.4 Display the frame

The imShow() is a function of HighGui and it is required to call the waitKey regulerly. The processing of the event loop of the imshow() function is done by calling waitKey. The function waitKey() waits for key event for a “delay” (here, 5 milliseconds). Windows events like redraw, resizing, input event etc. are processed by HighGui. So we call the waitKey function, even with a 1ms delay.

6.5 Mouse Movement

We have to first calculate the center of both detected red object which we can easily do by taking the average of the bounding boxes maximum and minimum points. now we got 2 co-ordinate from the center of the 2 objects we will find the average of that and we will get the red point shown in the image. We are converting the detected coordinate from camera resolution to the actual screen resolution. After that we set the location as the mouse_position. but to move the mouse pointer it will take time. So we have to wait till the mouse pointer reaches that point. So we started a loop and we are not doing anything there we are just waiting will the current mouse location is same as assigned mouse location. That is for the open gesture.

6.6 Clicking

The next step is to implement the close gesture. The operation is performed by clicking the object and dragging it. It is similar to the open gesture, but the difference is we only have one object here so we only need to calculate the center of it. And that will be placed on the location where we will position our mouse pointer. Instead of mouse release operation we will be performing a mouse press operation.

6.7 Drag

In order to implement the dragging we indroduce a variable ‘pinchflag’. It will be set to 1 if it was clicked earlier. So after clicking whenever we find the open gesture we check if the pinchflag is set to 1. If it is set to one then Drag operation is performed otherwise the mouse move operation is performed.

6.8 DnD Frame

First we create the MyFileDropTarget class. Inside that we have one overridden method, OnDropFiles. This method accepts the x/y position of the mouse along with the file paths that are dropped.

CHAPTER 7

Testing

7.1 OVERVIEW ABOUT TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risk of software implementation. Test techniques include the process executing the program or application with the intent of finding software bugs(errors or other defects), and verifying that the software product is fit for use.

7.2 TYPES OF SOFTWARE TESTING:

7.2.1 White box testing

7.2.2 Black box testing

7.2.3 Unit Testing

7.2.4 Functional Testing

7.2.5 Performance Testing

7.2.6 Integration Testing

7.2.7 Validation Testing

7.2.8 System Testing

7.2.9 Structure Testing

7.2.10 Output Testing

7.2.11 User Acceptance Testing

7.2.1 WHITE BOX TESTING

White Box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white box testing an internal perspective of the system as well as programming skills are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. White box testing can be applied at the unit, integration and system levels of the testing process. Although traditional testers tended to think of white box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration and between subsystems during a system level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

7.2.2 BLACK BOX TESTING

Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing like unit, integration, software, system and acceptance. It is sometimes referred to as specification based testing. Black box testing, also known as behavioural testing is a software testing method in which the internal structure design implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

7.2.3 UNIT TESTING

• Unit testing, also known as Module testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.

• Unit test comprises of the set of tests performed by an individual programmer before integration of the unit into the system.

• Unit test focuses on the smallest unit of software design the software component or module.

• Using component level design, important control paths are tested to uncover errors within the boundary of the module.

• Unit test is white box oriented and the step can be conducted in parallel for multiple components.

7.2.4 FUNCTIONAL TESTING

Functional testing is a type of software testing whereby the system is tested against the functional requirements specifications. Functions or features are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs but rather with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black box testing technique is used in which the internal logic of the system being tested is not known to the tester. Functional testing is normally performed during the levels of system testing and acceptance testing. Typically Functional testing involves the following steps: Identify functions that the software is expected to perform. Create input data based on the function's specification. Determine the output

based on the function's specification. Execute the test case. Compare the actual and expected outputs.

7.2.5 PERFORMANCE TESTING

Performance testing is done to provide stakeholders with information about their application regarding speed, stability and scalability. More importantly, Performance testing uncovers what needs to be improved before the product goes to market. Without performance testing, software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems and poor usability. Performance testing determines the amount of execution tine spent in various parts of the unit, program throughput and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

7.2.6 INTEGRATION TESTING

- It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.
- It takes the unit tested modules and builds a program structure.
- All the modules are combined and tested as a whole.
- Integration of all the components to form the entire system and a overall testing is executed.

7.2.7 VALIDATION TESTING

- Validation test succeeds when the software functions in a manner that can be reasonably by the client.

- Software validation is achieved through a series of black box testing

which confirms to the requirements. The test is designed to uncover interface errors, is also used to demonstrate that software functions are operational, input is properly accepted, output are produced and the integrity of external information is maintained.

7.2.8 SYSTEM TESTING

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. Tests to find the discrepancies between the system and its original objective, current specification and system documentation.

7.2.9 STRUCTURAL TESTING

Structural testing is the type of testing carried out to test the structure of code. It is also known as testing or Glass box testing. This type of testing requires knowledge of the code, so it is mostly done by the developers. It is more concerned with how system does it rather than the functionality of the system. It provides more coverage to the testing. For example, to test certain error message in application, we need to test the trigger condition for it, but there must be many trigger for it. It is possible to miss out one while testing the requirements drafted in SRS. But using this testing, the trigger is most likely to be covered since structural testing aims to cover all the nodes and paths in the structure of code. It is concerned with exercising the internal logic of a program and traversing particular execution paths.

7.2.10 OUTPUT TESTING

• Output of test cases compared with the expected results created during design of test cases.

• Asking the user about the format required by them tests the output generated or displayed by the system under consideration.

• Here, the output format is considered into two was, one is on screen and another one is printed format.

• The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.

• The output comes out as the specified requirements as the user's hard copy.

7.2.11 USER ACCEPTANCE TESTING

o Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.

o The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.

It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

Two set of acceptance test to be run:

1. Those developed by quality assurance group.

2. Those developed by customer.

CHAPTER 8

SCREENSHOTS

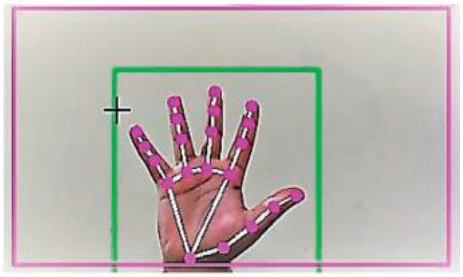

Fig: 8.1 Capturing video using the webcam (computer vision).

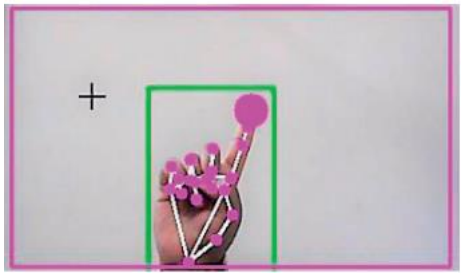

Fig: 8.2 Detection of which finger is up.

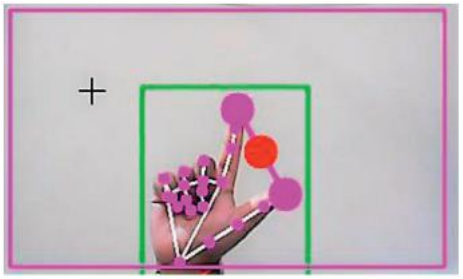

Fig: 8.3 Gesture for the computer to perform left button click.

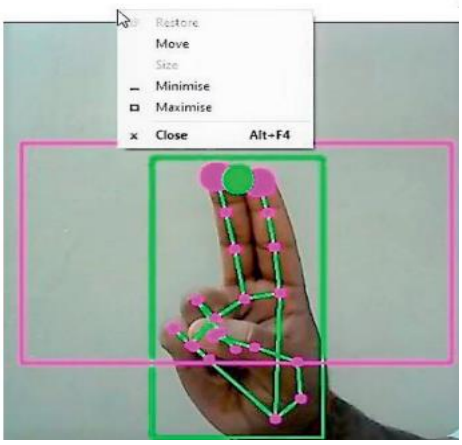

Fig: 8.4 Gesture for the computer to perform right button click.

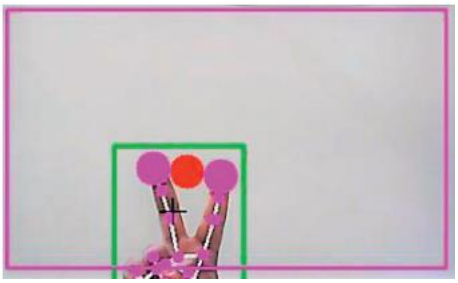

Fig: 8.5 Gesture for the computer to perform scroll down function.

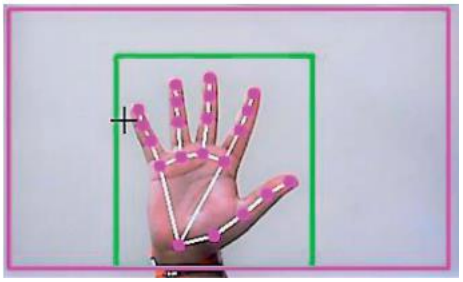

Fig: 8.6 Gesture for the computer to perform no action.

CHAPTER 9

CONCLUSION

Due to accuracy and efficiency plays an important role in making the program as useful as an actual physical mouse, a few techniques had to be implemented. After implanting such type of application there is big replacement of physical mouse i.e., there is no need of any physical mouse. Each & every movement of physical mouse is done with this motion tracking mouse (virtual mouse).

FUTURE SCOPE

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 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).

CHAPTER 10

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.

[7] P. Nandhini, J. Jaya, and J. George, “Computer vision system for food quality evaluation—a review,” in Proceedings of the 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), pp. 85–87, Coimbatore, India, July 2013.

[8] J. Jaya and K. Thanushkodi, “Implementation of certain system for medical image diagnosis,” European Journal of Scientific Research, vol. 53, no. 4, pp. 561–567, 2011.

[9] P. Nandhini and J. Jaya, “Image segmentation for food quality evaluation using computer vision system,” International Journal of Engineering Research and Applications, vol. 4, no. 2, pp. 1–3, 2014.

[10] J. Jaya and K. $anushkodi, “Implementation of classification system for medical images,” European Journal of Scientific Research, vol. 53, no. 4, pp. 561–569, 2011.

[11] J. T. Camillo Lugaresi, “MediaPipe: A Framework for Building Perception Pipelines,” 2019, https://arxiv.org/abs/1906.08172.

CHAPTER 11

APPENDIX

APPENDIX A - SOURCE CODE:

```
# Imports

import cv2

import mediapipe as mp

import pyautogui

import math

from enum import IntEnum

from ctypes import cast, POINTER

from comtypes import CLSCTX_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

from google.protobuf.json_format import MessageToDict

import screen_brightness_control as sbcontrol

pyautogui.FAILSAFE = False

mp_drawing = mp.solutions.drawing_utils

mp_hands = mp.solutions.hands

# Gesture Encodings

class Gest(IntEnum):

    # Binary Encoded

    FIST = 0

    PINKY = 1

    RING = 2
```

```
    MID = 4
    LAST3 = 7
    INDEX = 8
    FIRST2 = 12
    LAST4 = 15
    THUMB = 16
    PALM = 31
        # Extra Mappings
    V_GEST = 33
    TWO_FINGER_CLOSED = 34
    PINCH_MAJOR = 35
    PINCH_MINOR = 36
# Multi-handedness Labels
class HLabel(IntEnum):
    MINOR = 0
    MAJOR = 1
# Convert Mediapipe Landmarks to recognizable Gestures
class HandRecog:
    def __init__(self, hand_label):
        self.finger = 0
        self.ori_gesture = Gest.PALM
        self.prev_gesture = Gest.PALM
        self.frame_count = 0
        self.hand_result = None
```

```
    self.hand_label = hand_label
    def update_hand_result(self, hand_result):
    self.hand_result = hand_result
def get_signed_dist(self, point):
    sign = -1
    if self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y:
        sign = 1
    dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x)**2
    dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y)**2
    dist = math.sqrt(dist)
    return dist*sign
    def get_dist(self, point):
    dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x)**2
    dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y)**2
    dist = math.sqrt(dist)
    return dist
    def get_dz(self,point):
    return abs(self.hand_result.landmark[point[0]].z - self.hand_result.landmark[point[1]].z)
    # Function to find Gesture Encoding using current finger_state.
# Finger_state: 1 if finger is open, else 0
def set_finger_state(self):
    if self.hand_result == None:
        return
```

```
    points = [[8,5,0],[12,9,0],[16,13,0],[20,17,0]]
    self.finger = 0
    self.finger = self.finger | 0 #thumb
    for idx,point in enumerate(points):
      dist = self.get_signed_dist(point[:2])
       dist2 = self.get_signed_dist(point[1:])
              try:
          ratio = round(dist/dist2,1)
       except:
          ratio = round(dist1/0.01,1)

       self.finger = self.finger << 1
       if ratio > 0.5 :
          self.finger = self.finger | 1
# Handling Fluctations due to noise
def get_gesture(self):
    if self.hand_result == None:
       return Gest.PALM
    current_gesture = Gest.PALM
    if self.finger in [Gest.LAST3,Gest.LAST4] and self.get_dist([8,4]) < 0.05:
       if self.hand_label == HLabel.MINOR :
          current_gesture = Gest.PINCH_MINOR
       else:
          current_gesture = Gest.PINCH_MAJOR
```

```
elif Gest.FIRST2 == self.finger :
    point = [[8,12],[5,9]]
    dist1 = self.get_dist(point[0])
    dist2 = self.get_dist(point[1])
    ratio = dist1/dist2
    if ratio > 1.7:
        current_gesture = Gest.V_GEST
    else:
        if self.get_dz([8,12]) < 0.1:
            current_gesture =  Gest.TWO_FINGER_CLOSED
        else:
            current_gesture =  Gest.MID

else:
    current_gesture =  self.finger

if current_gesture == self.prev_gesture:
    self.frame_count += 1
else:
    self.frame_count = 0

self.prev_gesture = current_gesture

if self.frame_count > 4 :
```

```
            self.ori_gesture = current_gesture
        return self.ori_gesture
# Executes commands according to detected gestures
class Controller:
    tx_old = 0
    ty_old = 0
    trial = True
    flag = False
    grabflag = False
    pinchmajorflag = False
    pinchminorflag = False
    pinchstartxcoord = None
    pinchstartycoord = None
    pinchdirectionflag = None
    prevpinchlv = 0
    pinchlv = 0
    framecount = 0
    prev_hand = None
    pinch_threshold = 0.3
        def getpinchylv(hand_result):
        dist = round((Controller.pinchstartycoord - hand_result.landmark[8].y)*10,1)
        return dist

    def getpinchxlv(hand_result):
```

```
    dist = round((hand_result.landmark[8].x - Controller.pinchstartxcoord)*10,1)
    return dist


def changesystembrightness():
    currentBrightnessLv = sbcontrol.get_brightness()/100.0
    currentBrightnessLv += Controller.pinchlv/50.0
    if currentBrightnessLv > 1.0:
        currentBrightnessLv = 1.0
    elif currentBrightnessLv < 0.0:
        currentBrightnessLv = 0.0
    sbcontrol.fade_brightness(int(100*currentBrightnessLv) , start = sbcontrol.get_brightness())
    def changesystemvolume():
    devices = AudioUtilities.GetSpeakers()
    interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
    volume = cast(interface, POINTER(IAudioEndpointVolume))
    currentVolumeLv = volume.GetMasterVolumeLevelScalar()
    currentVolumeLv += Controller.pinchlv/50.0
    if currentVolumeLv > 1.0:
        currentVolumeLv = 1.0
    elif currentVolumeLv < 0.0:
        currentVolumeLv = 0.0
    volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)
    def scrollVertical():
    pyautogui.scroll(120 if Controller.pinchlv>0.0 else -120)
```

```
        def scrollHorizontal():
    pyautogui.keyDown('shift')
    pyautogui.keyDown('ctrl')
    pyautogui.scroll(-120 if Controller.pinchlv>0.0 else 120)
    pyautogui.keyUp('ctrl')
    pyautogui.keyUp('shift')


# Locate Hand to get Cursor Position
# Stabilize cursor by Dampening
def get_position(hand_result):
    point = 9
    position = [hand_result.landmark[point].x ,hand_result.landmark[point].y]
    sx,sy = pyautogui.size()
    x_old,y_old = pyautogui.position()
    x = int(position[0]*sx)
    y = int(position[1]*sy)
    if Controller.prev_hand is None:
        Controller.prev_hand = x,y
    delta_x = x - Controller.prev_hand[0]
    delta_y = y - Controller.prev_hand[1]


    distsq = delta_x**2 + delta_y**2
    ratio = 1
    Controller.prev_hand = [x,y]
```

```
    if distsq <= 25:
        ratio = 0
    elif distsq <= 900:
        ratio = 0.07 * (distsq ** (1/2))
    else:
        ratio = 2.1
    x , y = x_old + delta_x*ratio , y_old + delta_y*ratio
    return (x,y)


def pinch_control_init(hand_result):
    Controller.pinchstartxcoord = hand_result.landmark[8].x
    Controller.pinchstartycoord = hand_result.landmark[8].y
    Controller.pinchlv = 0
    Controller.prevpinchlv = 0
    Controller.framecount = 0
# Hold final position for 5 frames to change status
def pinch_control(hand_result, controlHorizontal, controlVertical):
    if Controller.framecount == 5:
        Controller.framecount = 0
        Controller.pinchlv = Controller.prevpinchlv

        if Controller.pinchdirectionflag == True:
            controlHorizontal() #x
```

```
            elif Controller.pinchdirectionflag == False:
                controlVertical() #y

        lvx =  Controller.getpinchxlv(hand_result)
        lvy =  Controller.getpinchylv(hand_result)

        if abs(lvy) > abs(lvx) and abs(lvy) > Controller.pinch_threshold:
            Controller.pinchdirectionflag = False
            if abs(Controller.prevpinchlv - lvy) < Controller.pinch_threshold:
                Controller.framecount += 1
            else:
                Controller.prevpinchlv = lvy
                Controller.framecount = 0

        elif abs(lvx) > Controller.pinch_threshold:
            Controller.pinchdirectionflag = True
            if abs(Controller.prevpinchlv - lvx) < Controller.pinch_threshold:
                Controller.framecount += 1
            else:
                Controller.prevpinchlv = lvx
                Controller.framecount = 0
    def handle_controls(gesture, hand_result):
        x,y = None,None
        if gesture != Gest.PALM :
```

```
    x,y = Controller.get_position(hand_result)
        # flag reset
if gesture != Gest.FIST and Controller.grabflag:
    Controller.grabflag = False
    pyautogui.mouseUp(button = "left")

if gesture != Gest.PINCH_MAJOR and Controller.pinchmajorflag:
    Controller.pinchmajorflag = False

if gesture != Gest.PINCH_MINOR and Controller.pinchminorflag:
    Controller.pinchminorflag = False
# implementation
if gesture == Gest.V_GEST:
    Controller.flag = True
    pyautogui.moveTo(x, y, duration = 0.1)
elif gesture == Gest.FIST:
    if not Controller.grabflag :
        Controller.grabflag = True
        pyautogui.mouseDown(button = "left")
    pyautogui.moveTo(x, y, duration = 0.1)

elif gesture == Gest.MID and Controller.flag:
    pyautogui.click()
    Controller.flag = False
```

```
        elif gesture == Gest.INDEX and Controller.flag:
            pyautogui.click(button='right')
            Controller.flag = False

        elif gesture == Gest.TWO_FINGER_CLOSED and Controller.flag:
            pyautogui.doubleClick()
            Controller.flag = False

        elif gesture == Gest.PINCH_MINOR:
            if Controller.pinchminorflag == False:
                Controller.pinch_control_init(hand_result)
                Controller.pinchminorflag = True
            Controller.pinch_control(hand_result,Controller.scrollHorizontal, Controller.scrollVertical)

        elif gesture == Gest.PINCH_MAJOR:
            if Controller.pinchmajorflag == False:
                Controller.pinch_control_init(hand_result)
                Controller.pinchmajorflag = True
            Controller.pinch_control(hand_result,Controller.changesystembrightness,
Controller.changesystemvolume)

class GestureController:
    gc_mode = 0
    cap = None
    CAM_HEIGHT = None
```

```
CAM_WIDTH = None

hr_major = None # Right Hand by default

hr_minor = None # Left hand by default

dom_hand = True


def __init__(self):

    GestureController.gc_mode = 1

    GestureController.cap = cv2.VideoCapture(0)

    GestureController.CAM_HEIGHT = GestureController.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)

    GestureController.CAM_WIDTH = GestureController.cap.get(cv2.CAP_PROP_FRAME_WIDTH)


def classify_hands(results):

    left , right = None,None

    try:

        handedness_dict = MessageToDict(results.multi_handedness[0])

        if handedness_dict['classification'][0]['label'] == 'Right':

            right = results.multi_hand_landmarks[0]

        else :

            left = results.multi_hand_landmarks[0]

    except:

        pass


    try:

        handedness_dict = MessageToDict(results.multi_handedness[1])
```

```
            if handedness_dict['classification'][0]['label'] == 'Right':
                right = results.multi_hand_landmarks[1]
            else :
                left = results.multi_hand_landmarks[1]
        except:
            pass
                if GestureController.dom_hand == True:
            GestureController.hr_major = right
            GestureController.hr_minor = left
        else :
            GestureController.hr_major = left
            GestureController.hr_minor = right


    def start(self):


        handmajor = HandRecog(HLabel.MAJOR)
        handminor = HandRecog(HLabel.MINOR)


        with mp_hands.Hands(max_num_hands = 2,min_detection_confidence=0.5,
min_tracking_confidence=0.5) as hands:
            while GestureController.cap.isOpened() and GestureController.gc_mode:
                success, image = GestureController.cap.read()


                if not success:
                    print("Ignoring empty camera frame.")
```

```
        continue
    image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = hands.process(image)

    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    if results.multi_hand_landmarks:
        GestureController.classify_hands(results)
        handmajor.update_hand_result(GestureController.hr_major)
        handminor.update_hand_result(GestureController.hr_minor)

        handmajor.set_finger_state()
        handminor.set_finger_state()
        gest_name = handminor.get_gesture()

        if gest_name == Gest.PINCH_MINOR:
            Controller.handle_controls(gest_name, handminor.hand_result)
        else:
            gest_name = handmajor.get_gesture()
            Controller.handle_controls(gest_name, handmajor.hand_result)

        for hand_landmarks in results.multi_hand_landmarks:
```

```
                    mp_drawing.draw_landmarks(image, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
            else:
                Controller.prev_hand = None
            cv2.imshow('Gesture Controller', image)
            if cv2.waitKey(5) & 0xFF == 13:
                break
        GestureController.cap.release()
        cv2.destroyAllWindows()


# uncomment to run directly
gc1 = GestureController()
gc1.start()
```