GESTURE MATE -GESTURE CONTROLLED VIRTUAL MOUSE

A PROJECT REPORT

Submitted by

SRIRAM V.S
[REGISTER NO:211419104267]

RAGHUL SADAGOBAN S
[REGISTER NO:211419104207]

YAZHINIA KUMARAN I
[REGISTER NO:211419104313]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE,

(An Autonomous Institution, Affiliated to Anna University,Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University,Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"GESTURE MATE-GESTURE CONTROLLED WITH VIRTUAL MOUSE"**is the bonafide work of **"SRIRAMV.S (211419104267), RAGHUL SADAGOBAN (211419104207), YAZHINIA KUMARAN I (211419104313)"**who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA M.E.,Ph.D.	**Mrs.A.KANCHANA M.E**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
	SUPERVISOR
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENTS

We **SRIRAM V.S (211419104267), RAGHUL SADAGOBAN S (211419104207), YAZHINIA KUMARAM I (211419104313)** hereby declare that this project report titled "**GESTURE MATE (GESTURE CONTROLLED VIRTUAL MOUSE)**", under the guidance of **Mrs.A.KANCHANA M.E** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SRIRAM V.S

RAGHUL SADAGOBAN S

YAZHINIA KUMARAN I

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project. We would like express our heartfelt and sincere thanks to our Director **Tmt.C.VIJAYARAJESWARI**, **DR.C.SAKTHI KUMAR,M.E.,Ph.D.** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,Ph.D** for providing us with the necessary facilities for completion of this project. We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course. We thank the HOD of CSE Department, **DR.L.JABASHEELA.,M.E.,Ph.D.** for the support extended throughout the project. We would like to thank my project coordinator **MR.M.MOHAN,M.E(Ph.D.)**, and Project Guide **Mrs.A.KANCHANA M.E** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project

SRIRAM V.S

RAGHUL SADAGOBAN S

YAZHINIA KUMARAN I

<u>TABLE OF CONTENTS</u>

<u>CHAPTER-1</u>

INTRODUCTION

ABSTRACT

The mouse is one of the wonderful inventions of Human-Computer Interaction (HCI) 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 left click, right click, scrolling functions, and computer cursor function without the use of the physical mouse. The algorithm is based on deep learning for detecting the hands. Hence, the proposed system will avoid COVID-19 spread by eliminating the human intervention and dependency of devices to control the computer.

1.1 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.

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

LITERATURE REVIEW

2.1 LITERATURE SURVEY

1. Title: A gesture-based virtual mouse control system for head-mounted displays

   AUTHOR: H. Chen, W. Sheng, Y. Huang, and W. Wang

   YEAR: 2018

   PAPER: IEEE International Conference on Information and Automation (ICIA), 2018, pp. 417-422.

This paper presents a gesture-based virtual mouse control system for head-mounted displays (HMDs) using hand tracking and recognition techniques. The system allows users to control the mouse cursor by moving their hand in the air and performing hand gestures. The proposed system was evaluated through a user study and showed promising results in terms of accuracy and usability.

2. Title:A novel gesture-based virtual mouse control system for virtual reality applications

   Author: Y. Jiao, X. Qiu, Y. Sun, and Y. Wang

   Year: 2018

   Paper:15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)

This paper presents a novel gesture-based virtual mouse control system for virtual reality applications using a depth camera. The system allows users to control the mouse cursor by performing hand gestures, such as pointing and clicking, in front of the camera. The proposed system was evaluated through a user study and showed good performance in terms of accuracy and user satisfaction.

3. Title: Gesture-based virtual mouse control using deep learning for VR applications

   Author: S. Kim, H. Park, and J. Kim

   Year:2020

Paper: IEEE International Conference on Consumer Electronics (ICCE), 2020, pp. 1-2.

This paper presents a gesture-based virtual mouse control system using deep learning techniques for virtual reality applications. The system uses a camera to capture the user's hand movements and applies a deep learning model to recognize hand gestures and control the mouse cursor accordingly. The proposed system was evaluated through a user study and showed good performance in terms of accuracy and user satisfaction.

4. Title: A Comparative Study of Gesture Recognition Techniques for Virtual Mouse Control

Author: S. Saha and S. R. Saha

Year: 2021

Paper: International Journal of Computer Science and Network Security, vol. 21, no. 2, pp. 69-74.

This paper provides a comparative study of different gesture recognition techniques for virtual mouse control. The authors evaluated four different techniques, including template matching, contour-based detection, color-based detection, and deep learning-based detection, and compared their performance in terms of accuracy and efficiency.

5. Title: Design and Implementation of a Gesture-Controlled Virtual Mouse for Disabled People

Author: J. J. Liu and C. H. Huang

Year: 2018

Paper: IEEE Access, vol. 6, pp. 11184-11194.

This paper presents a gesture-controlled virtual mouse system for people with disabilities. The system uses a depth sensor to track hand movements and translate them into mouse movements. The authors evaluated the system with a group of disabled participants and found that it was easy to use and provided a reliable alternative to traditional mouse devices.

6.Title: A Survey of Hand Gesture Recognition Techniques and Applications

Author: N. N. Anusuya and S. B. Suresh

Year: 2016

Paper: International Journal of Advanced Research in Computer and Communication Engineering, vol. 5, no. 6, pp. 303-307.

This paper provides a comprehensive survey of hand gesture recognition techniques and their applications, including virtual mouse control. The authors discuss a range of techniques, from traditional computer vision approaches to more recent deep learning-based methods, and highlight the key challenges and future directions in this field.

Overall, these studies demonstrate the potential of gesture-based virtual mouse control systems for virtual reality applications using different techniques such as hand tracking, depth cameras, and deep learning. However, there is still room for improvement in terms of accuracy, speed, and user experience, and further research is needed to address these challenges.

<u>CHAPTER 3</u>

SYSTEM ANALYSIS

3.1 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.

3.2 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[2]. 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.

<u>CHAPTER-4</u>

REQUIREMENT AND SPECIFICATION

4.1 HARDWARE REQUIREMENTS

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM
- 2gb Graphic card 3.2 Software Specification

4.2 SOFTWARESPECIFICATION

● Language : PYTHON

● IDE : ANACONDA, OPEN CV

CHAPTER-5

SOFTWARE DESCRIPTION

5.1 PYTHON (Programming Language)

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers get interested in Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

5.2 ANACONDA

Anaconda is an open-source software that contains Jupyter, Spyder, etc that are used for large data processing, data analytics, heavy scientific computing.Anaconda works for R and Python programming language. Package versions are managed by the package management system conda.

A virtual environment is a named, isolated, working copy of Python that that maintains its own files, directories, and paths so that you can work with specific versions of libraries or Python itself without affecting other Python projects. Virtual environments make it easy to cleanly separate different projects and avoid problems with different dependencies and version requirements across components.

The conda command is the preferred interface for managing installations and virtual environments with the Anaconda Python distribution. If you have a vanilla Python installation or other Python distribution, see virtual environment.

5.3OPEN CV

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russians, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language**.**

5.4 USE OF VIRTUAL ENVIRONMENT IN PYTHON

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.
This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.
The solution for this problem is to create a virtual environment, a self-contained

directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0.

CHAPTER-6

SYSTEM ARCHITECTURE

SYSTEM ARCHITECTURE

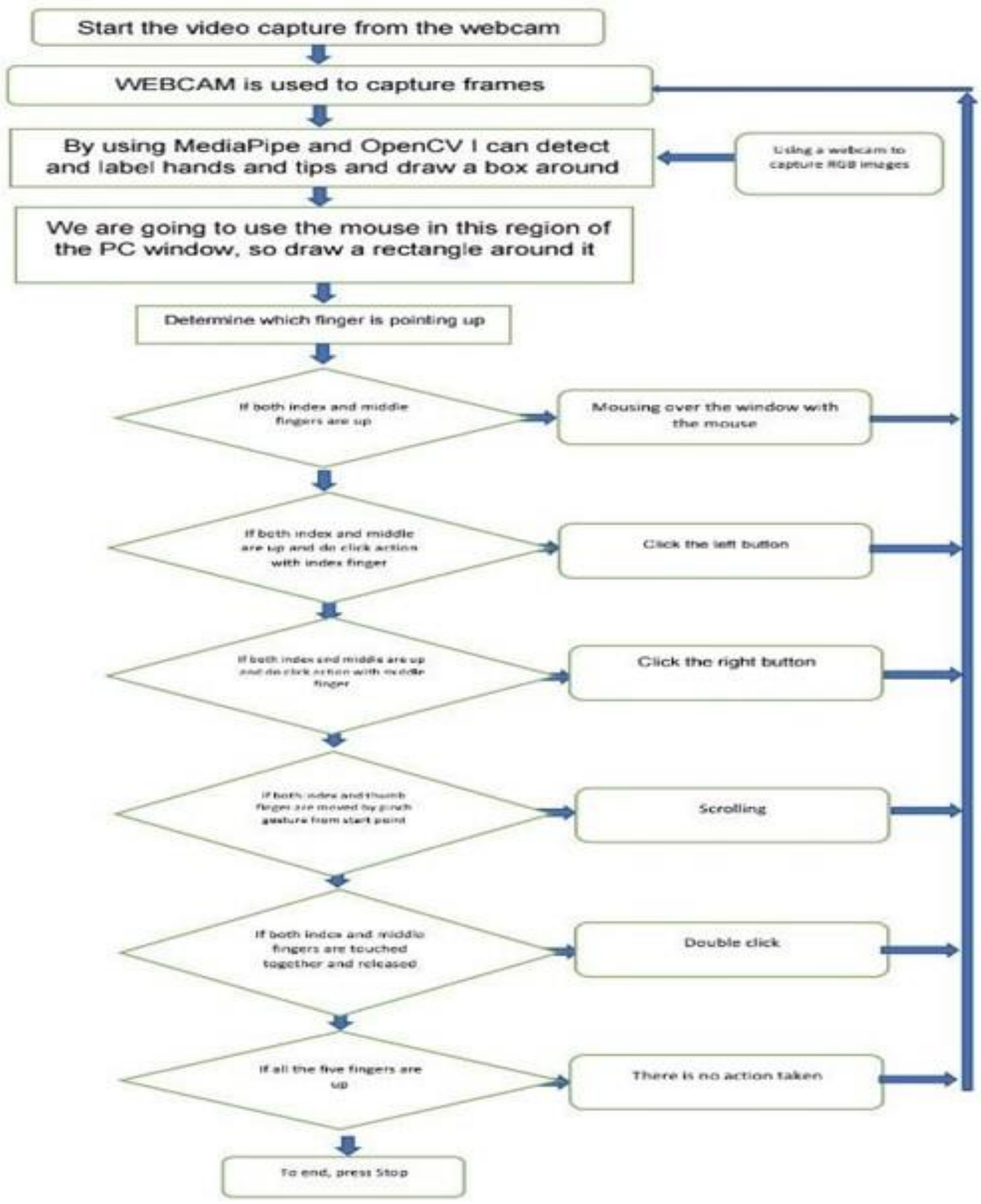

Start the video capture from the webcam
WEBCAM is used to capture frames
By using MediaPipe and OpenCV I can detect and label hands and tips and draw a box around
Using a webcam to capture RGB images
We are going to use the mouse in this region of the PC window, so draw a rectangle around it
Determine which finger is pointing up
If both index and middle fingers are up
Mousing over the window with the mouse
If both index and middle are up and do click action with index finger
Click the left button
If both index and middle are up and do click action with middle finger
Click the right button
If both index and thumb finger are moved by pinch gesture from start point
Scrolling
If both index and middle fingers are touched together and released
Double click
If all the five fingers are up
There is no action taken
To end, press Stop

Fig.3.1 Work Flowchart Diagram

The system architecture for a gesture control virtual mouse typically includes the following components:

Gesture recognition module: This module detects and recognizes hand gestures made by the user, typically using a camera or a depth sensor. The gestures may include movements such as swipes, taps, and clicks.

Hand tracking module: This module tracks the position and movement of the user's hand in real-time. It may use computer vision techniques such as optical flow or feature detection to estimate the hand's location and orientation.

Gesture interpretation module: This module maps the detected gestures to specific mouse actions such as left-click, right-click, scroll, and drag. Machine learning algorithms such as decision trees or neural networks may be used to classify the gestures and perform the mapping.

User interface module: This module provides the user with visual feedback, indicating the position of the virtual mouse cursor on the screen and the actions being performed. It may also include settings for adjusting the sensitivity of the gesture recognition and tracking algorithms.

Integration module: This module integrates the gesture control virtual mouse with the operating system and other applications. It may use standard input/output protocols such as USB or Bluetooth to communicate with the computer and emulate the behavior of a traditional mouse.

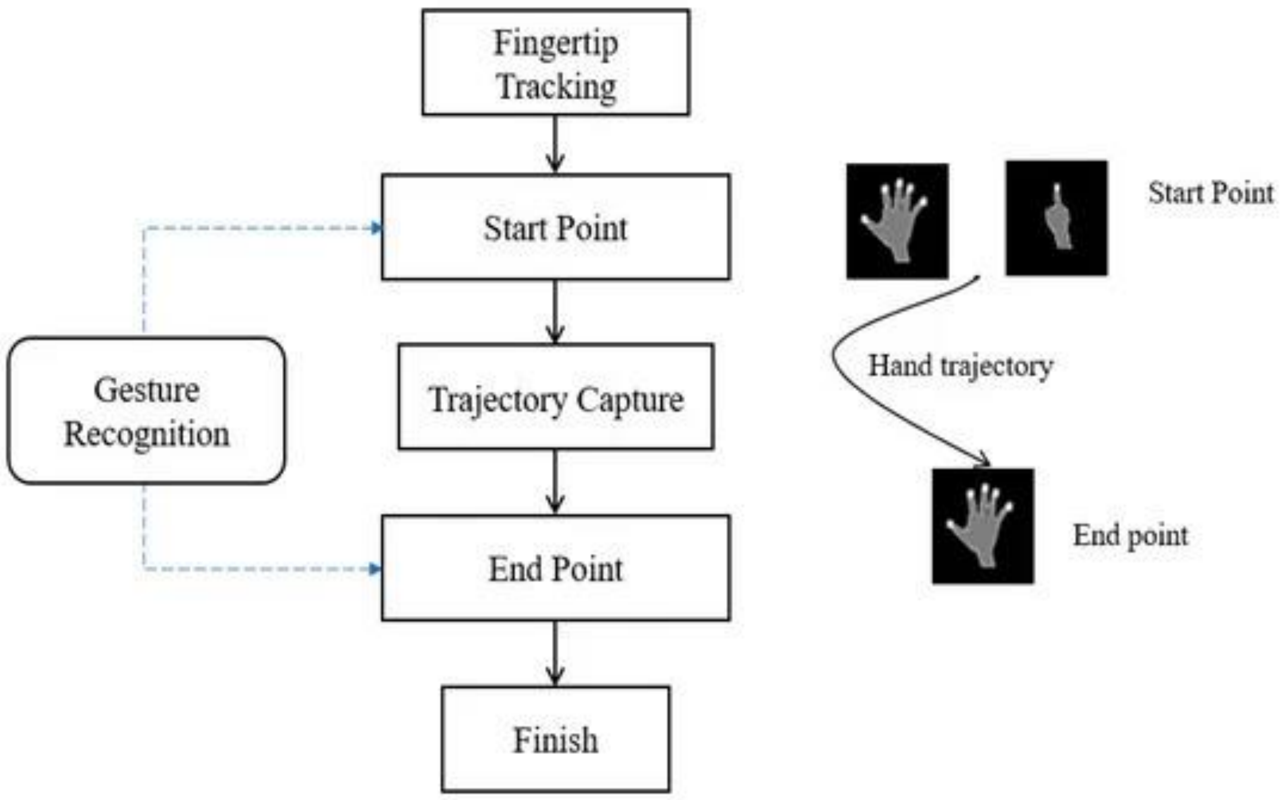
Overall, the architecture of a gesture control virtual mouse system involves the integration of multiple hardware and software components, requiring careful design and implementation to ensure accurate and responsive operation.

.

<u>CHAPTER-7</u>

SYSTEM DESIGN

7.1 UML DIAGRAM



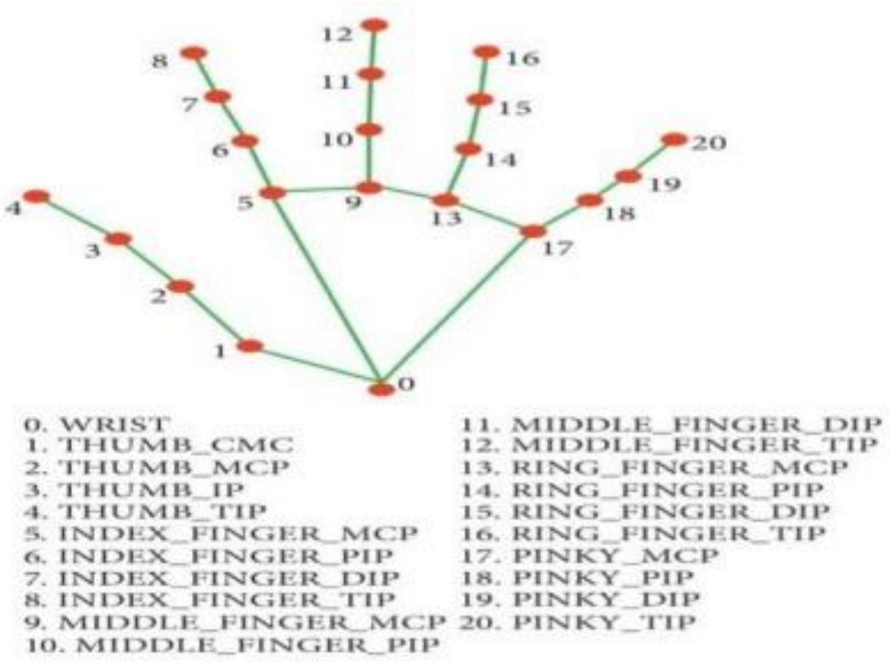
The UML diagram for gesture control virtual mouse, includes several classes, such as a Hand Recognition class and a Controller class. The Hand Recognition class would handle hand recognition and gesture detection using the Media pipe library, while the Controller class would handle mouse movement and click events using the Py Auto GUI library.

The Hand Recognition class would have methods for updating hand results, calculating distances and ratios between hand landmarks, and determining the current hand gesture based on finger position and proximity. The gestures would be defined as an enumeration using the Int Enum class.

The Controller class would have methods for calculating the mouse position based on hand movement and orientation, as well as detecting and responding to different hand gestures for mouse clicks and scrolling. It would also have flags and counters for tracking pinch gestures and click events.

Overall, the UML diagram would show the relationships and dependencies between the classes, including class inheritance, composition, and association. It would provide a visual representation of the code structure and logic for gesture control virtual mouse.

7.2 MODULAR DIAGRAM



Gesture Recognition Module: This module is responsible for recognizing the gestures made by the user. The module uses sensors or cameras to capture the hand movements of the user, and then processes the data to identify the intended gesture.

Mapping Module: This module maps the recognized gesture to a corresponding action. For example, a specific gesture may be mapped to the action of moving the cursor up or down, left or right.

Cursor Control Module: This module takes the actions identified by the mapping module and controls the movement of the cursor accordingly. It may use algorithms such as the Kalman filter to smooth out the cursor movement and improve accuracy.

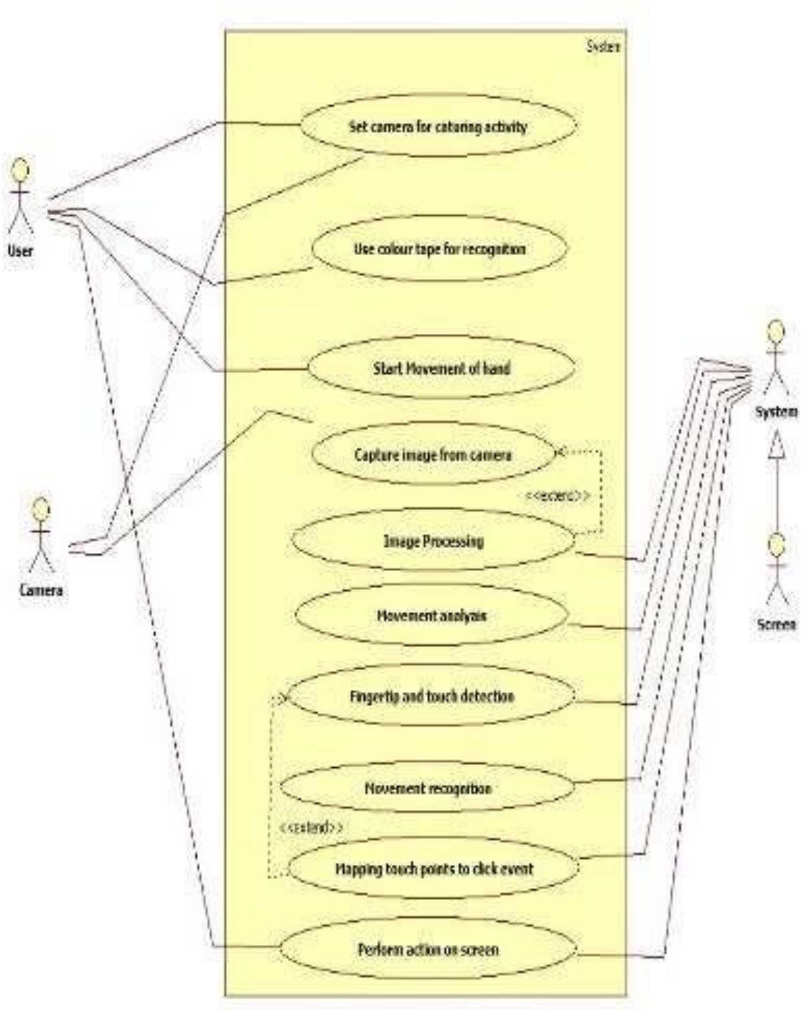
Click Control Module: This module is responsible for detecting the user's intention to click or double-click. It may use a variety of techniques such as analyzing the hand movement speed, time between clicks, or finger positions to determine the intended action.

Feedback Module: This module provides feedback to the user to indicate that their actions have been recognized and executed correctly. It may use visual or auditory cues such as sound effects or cursor changes to provide feedback.

User Interface Module: This module provides a graphical user interface for the user to interact with the system. It may display the current cursor position, provide options for adjusting sensitivity or gesture recognition settings, or allow the user to switch between different modes of operation.

Overall, the gesture-controlled virtual mouse system works by capturing the user's hand movements, recognizing the intended gesture, mapping it to a corresponding action, and then controlling the cursor accordingly. The system provides feedback to the user to indicate that their actions have been recognized and executed correctly.

7.3 USE CASE DIAGRAM


System
Set camera for caturing activity
User
Use colour tape for recognition
Start Movement of hand
Capture image from camera
<<extend>>
Image Processing
Camera
Movement analysis
Fingertip and touch detection
Movement recognition
<<extend>>
Mapping touch points to click event
Perform action on screen
System
Screen

Use Case Name: Mouse controlled using Hand gestures

Primary Actor: User

Secondary Actor: Screen

Use Case Description:

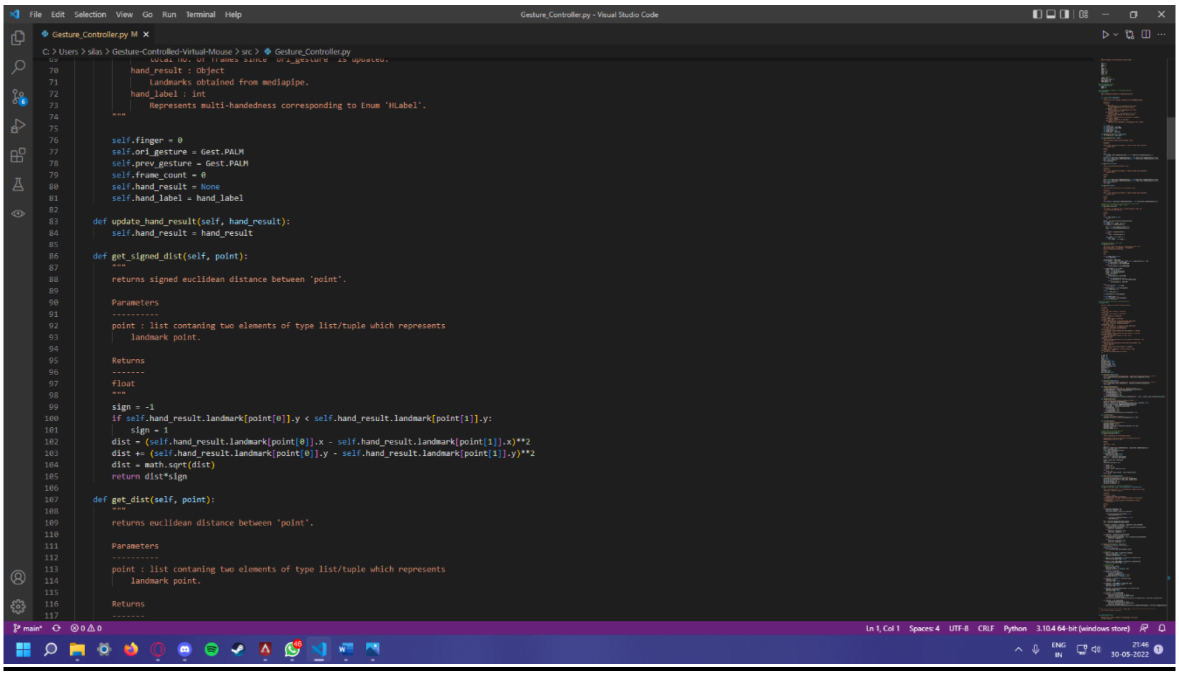
There are four actors in the use case. First actor is user which gives input to the web camera. That inputs are nothing but different hand gestures. Second actor is web camera which captures the inputs from user and gives these frames to the system. Thirs actor is system which performs mapping between users input and different actions (left click, right click, scrolling etc.) which are to be performed on the screen. Fourth actor is screen on which actual mouse movement will take place.

CHAPTER-8

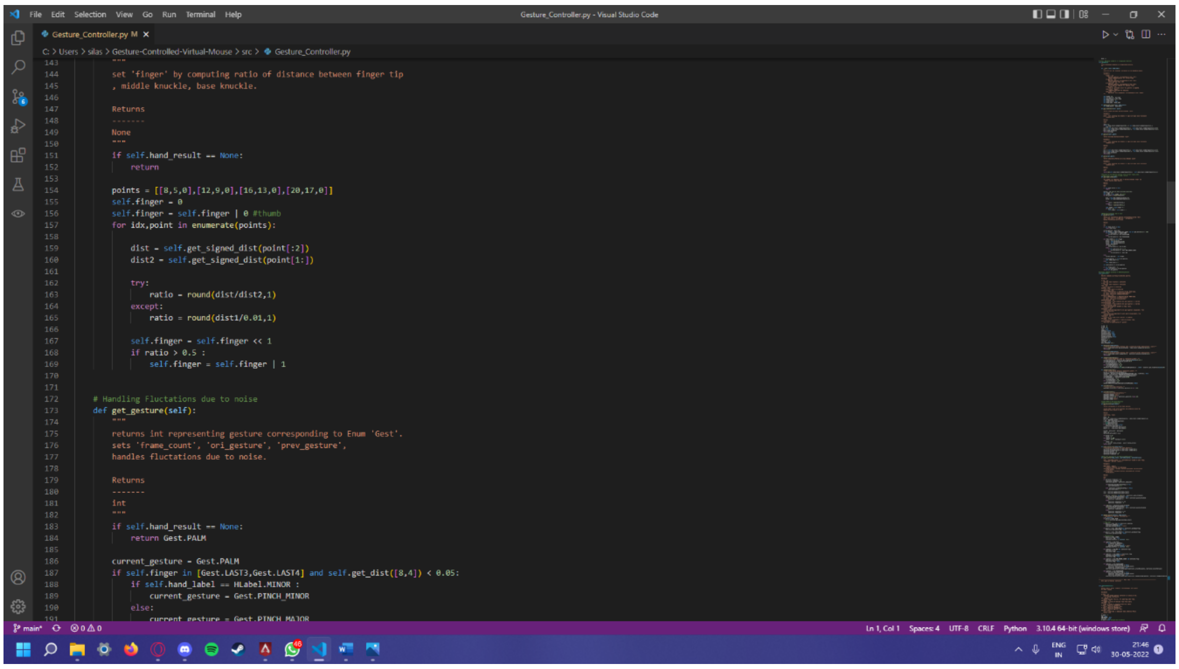
IMPLEMENTATION

8.1 CODING

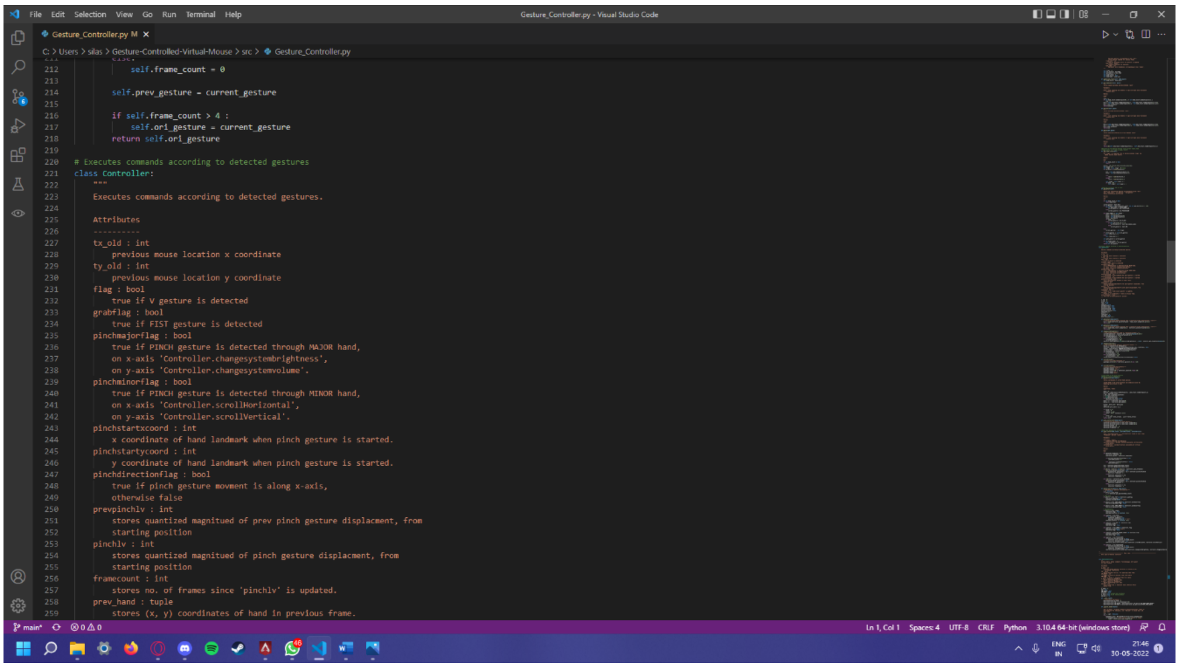
Gesture_Controller.py


File Edit Selection View Go Run Terminal Help
Gesture_Controller.py - Visual Studio Code
Gesture_Controller.py M
C: > Users > silas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py

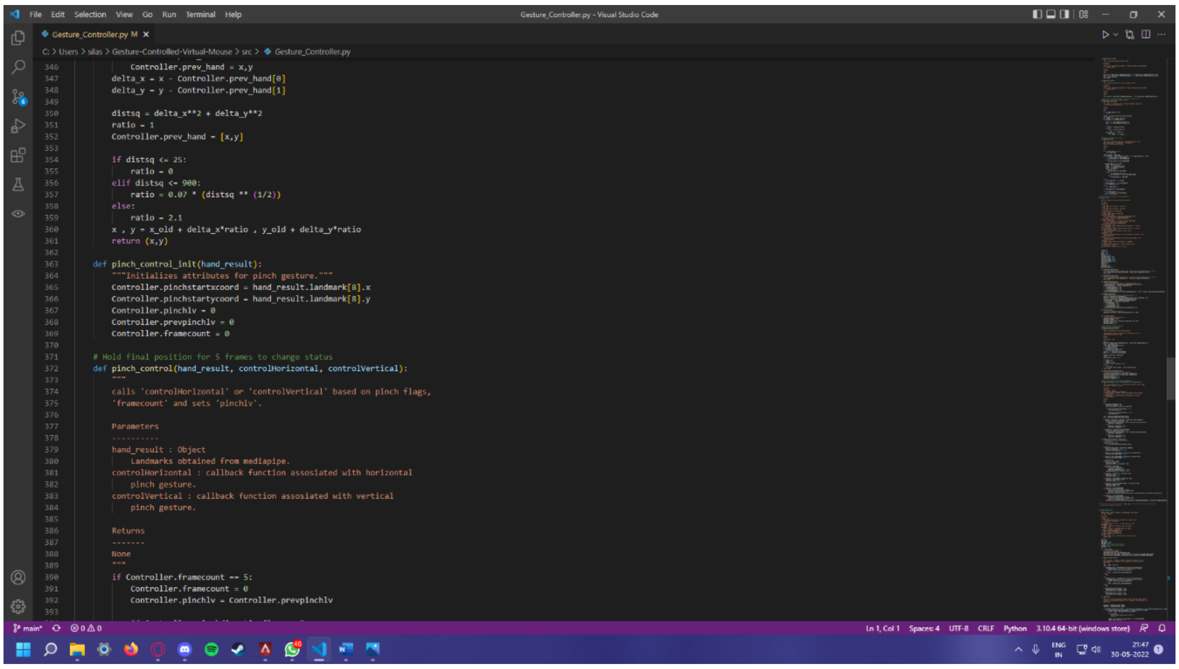
```python
                total no. of frames since 'ori_gesture' is updated.
            hand_result : Object
                Landmarks obtained from mediapipe.
            hand_label : int
                Represents multi-handedness corresponding to Enum 'HLabel'.
        """

        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):
        """
        returns signed euclidean distance between 'point'.

        Parameters
        ----------
        point : list contaning two elements of type list/tuple which represents
            landmark point.

        Returns
        -------
        float
        """
        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):
        """
        returns euclidean distance between 'point'.

        Parameters
        ----------
        point : list contaning two elements of type list/tuple which represents
            landmark point.

        Returns
        -------
```


File Edit Selection View Go Run Terminal Help
Gesture_Controller.py - Visual Studio Code
Gesture_Controller.py M
C: > Users > silas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py

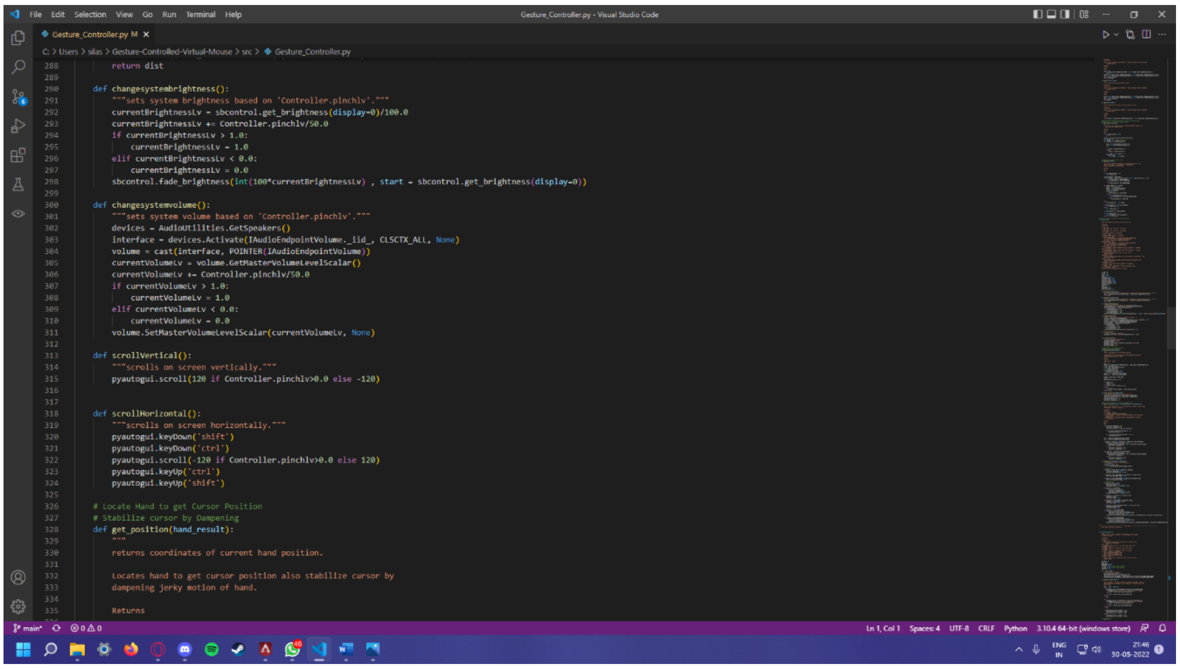
```python
        """
        set 'finger' by computing ratio of distance between finger tip
        , middle knuckle, base knuckle.

        Returns
        -------
        None
        """
        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):
        """
        returns int representing gesture corresponding to Enum 'Gest'.
        sets 'frame_count', 'ori_gesture', 'prev_gesture',
        handles fluctations due to noise.

        Returns
        -------
        int
        """
        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
```



```python
            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:
    """
    Executes commands according to detected gestures.

    Attributes
    ----------
    tx_old : int
        previous mouse location x coordinate
    ty_old : int
        previous mouse location y coordinate
    flag : bool
        true if V gesture is detected
    grabflag : bool
        true if FIST gesture is detected
    pinchmajorflag : bool
        true if PINCH gesture is detected through MAJOR hand,
        on x-axis 'Controller.changesystembrightness',
        on y-axis 'Controller.changesystemvolume'.
    pinchminorflag : bool
        true if PINCH gesture is detected through MINOR hand,
        on x-axis 'Controller.scrollHorizontal',
        on y-axis 'Controller.scrollVertical'.
    pinchstartxcoord : int
        x coordinate of hand landmark when pinch gesture is started.
    pinchstartycoord : int
        y coordinate of hand landmark when pinch gesture is started.
    pinchdirectionflag : bool
        true if pinch gesture movment is along x-axis,
        otherwise false
    prevpinchlv : int
        stores quantized magnitued of prev pinch gesture displacment, from
        starting position
    pinchlv : int
        stores quantized magnitued of pinch gesture displacment, from
        starting position
    framecount : int
        stores no. of frames since 'pinchlv' is updated.
    prev_hand : tuple
        stores (x, y) coordinates of hand in previous frame.
```

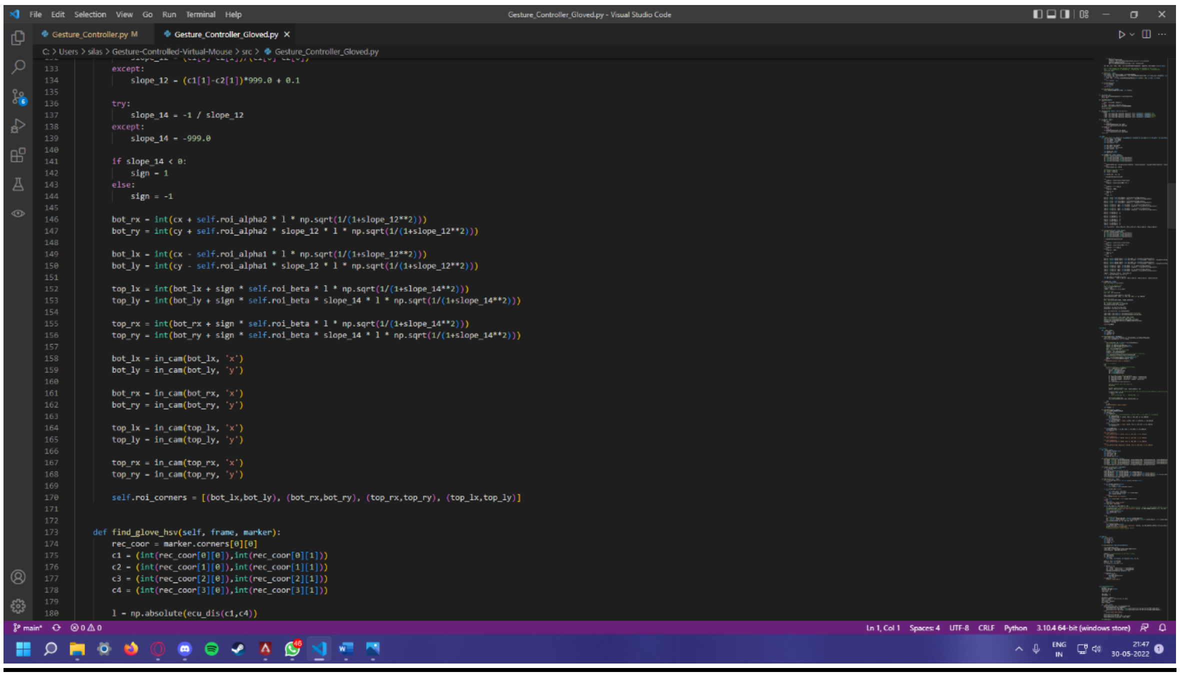


```python
            Controller.prev_hand = x,y
        delta_x = x - Controller.prev_hand[0]
        delta_y = y - Controller.prev_hand[1]

        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):
        """Initializes attributes for pinch gesture."""
        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):
        """
        calls 'controlHorizontal' or 'controlVertical' based on pinch flags,
        'framecount' and sets 'pinchlv'.

        Parameters
        ----------
        hand_result : Object
            Landmarks obtained from mediapipe.
        controlHorizontal : callback function assosiated with horizontal
            pinch gesture.
        controlVertical : callback function assosiated with vertical
            pinch gesture.

        Returns
        -------
        None
        """
        if Controller.framecount == 5:
            Controller.framecount = 0
            Controller.pinchlv = Controller.prevpinchlv

```

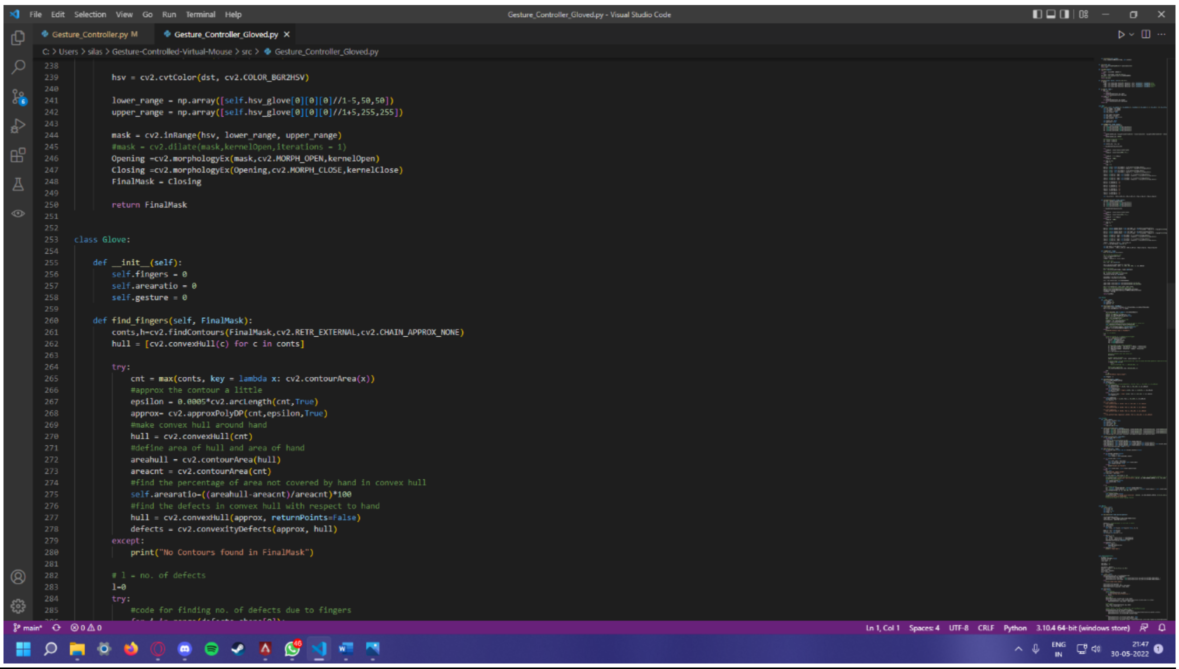


```python
        return dist

    def changesystembrightness():
        """sets system brightness based on 'Controller.pinchlv'."""
        currentBrightnessLv = sbcontrol.get_brightness(display=0)/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(display=0))

    def changesystemvolume():
        """sets system volume based on 'Controller.pinchlv'."""
        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():
        """scrolls on screen vertically."""
        pyautogui.scroll(120 if Controller.pinchlv>0.0 else -120)


    def scrollHorizontal():
        """scrolls on screen horizontally."""
        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):
        """
        returns coordinates of current hand position.

        Locates hand to get cursor position also stabilize cursor by
        dampening jerky motion of hand.

        Returns
```

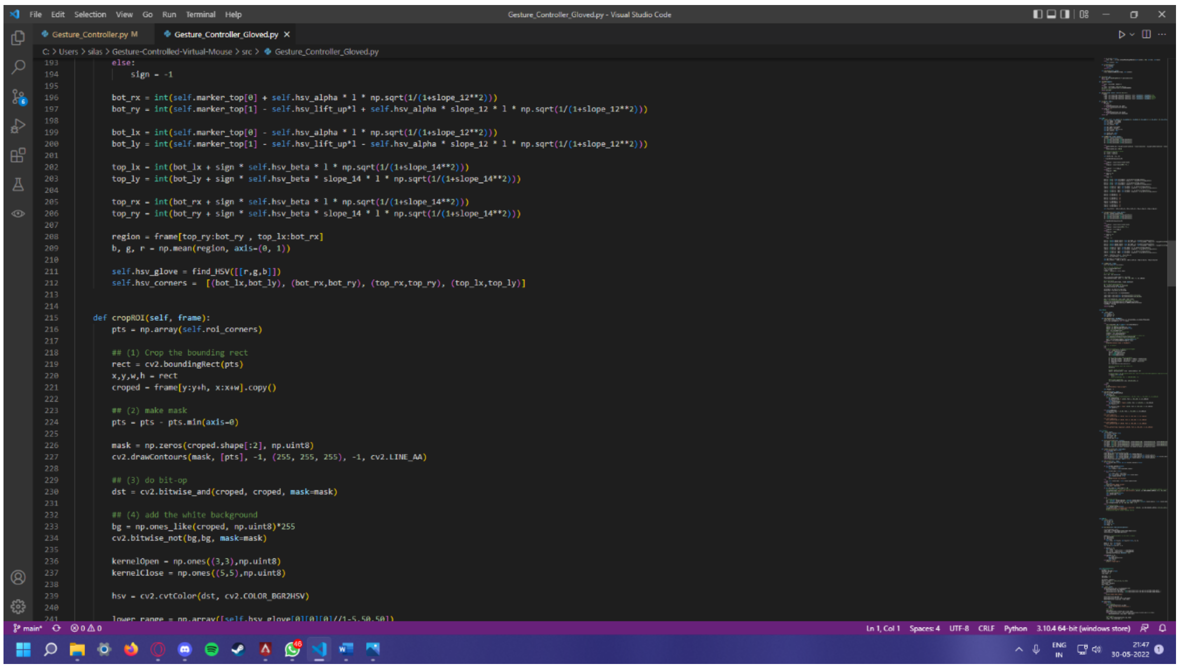
Gesture_Controller_Gloved.py



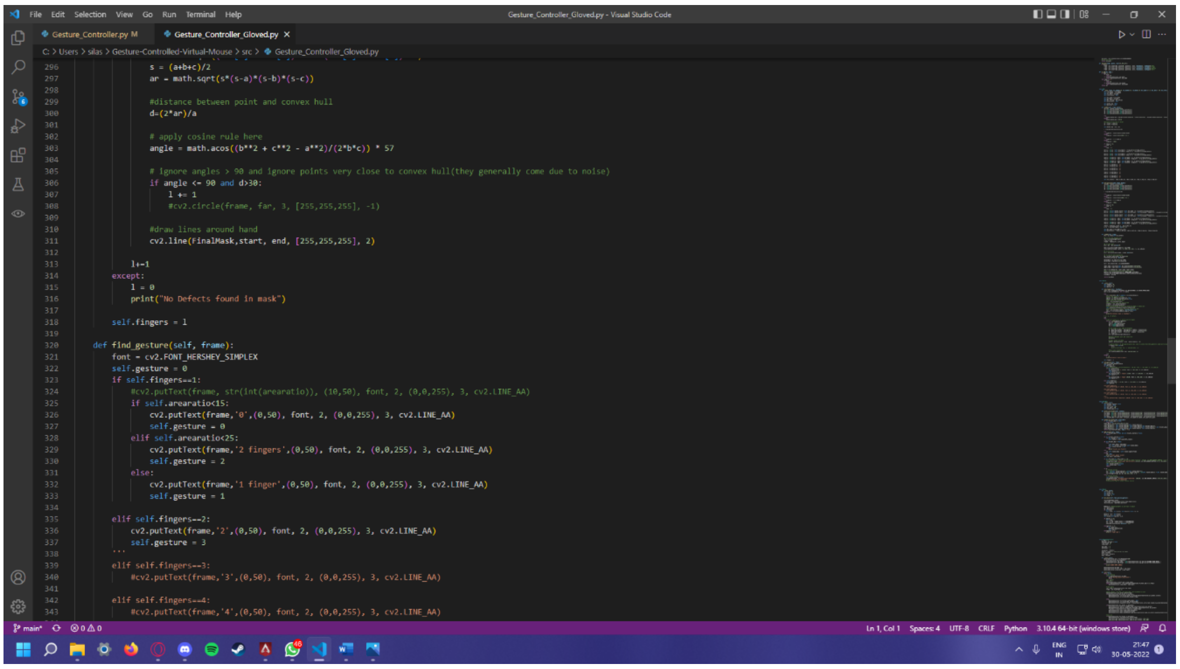
```python
            except:
                slope_12 = (c1[1]-c2[1])*999.0 + 0.1

            try:
                slope_14 = -1 / slope_12
            except:
                slope_14 = -999.0

            if slope_14 < 0:
                sign = 1
            else:
                sign = -1

            bot_rx = int(cx + self.roi_alpha2 * l * np.sqrt(1/(1+slope_12**2)))
            bot_ry = int(cy + self.roi_alpha2 * slope_12 * l * np.sqrt(1/(1+slope_12**2)))

            bot_lx = int(cx - self.roi_alpha1 * l * np.sqrt(1/(1+slope_12**2)))
            bot_ly = int(cy - self.roi_alpha1 * slope_12 * l * np.sqrt(1/(1+slope_12**2)))

            top_lx = int(bot_lx + sign * self.roi_beta * l * np.sqrt(1/(1+slope_14**2)))
            top_ly = int(bot_ly + sign * self.roi_beta * slope_14 * l * np.sqrt(1/(1+slope_14**2)))

            top_rx = int(bot_rx + sign * self.roi_beta * l * np.sqrt(1/(1+slope_14**2)))
            top_ry = int(bot_ry + sign * self.roi_beta * slope_14 * l * np.sqrt(1/(1+slope_14**2)))

            bot_lx = in_cam(bot_lx, 'x')
            bot_ly = in_cam(bot_ly, 'y')

            bot_rx = in_cam(bot_rx, 'x')
            bot_ry = in_cam(bot_ry, 'y')

            top_lx = in_cam(top_lx, 'x')
            top_ly = in_cam(top_ly, 'y')

            top_rx = in_cam(top_rx, 'x')
            top_ry = in_cam(top_ry, 'y')

            self.roi_corners = [(bot_lx,bot_ly), (bot_rx,bot_ry), (top_rx,top_ry), (top_lx,top_ly)]


        def find_glove_hsv(self, frame, marker):
            rec_coor = marker.corners[0][0]
            c1 = (int(rec_coor[0][0]),int(rec_coor[0][1]))
            c2 = (int(rec_coor[1][0]),int(rec_coor[1][1]))
            c3 = (int(rec_coor[2][0]),int(rec_coor[2][1]))
            c4 = (int(rec_coor[3][0]),int(rec_coor[3][1]))

            l = np.absolute(ecu_dis(c1,c4))
```



```
        hsv = cv2.cvtColor(dst, cv2.COLOR_BGR2HSV)

        lower_range = np.array([self.hsv_glove[0][0][0]//1-5,50,50])
        upper_range = np.array([self.hsv_glove[0][0][0]//1+5,255,255])

        mask = cv2.inRange(hsv, lower_range, upper_range)
        #mask = cv2.dilate(mask,kernelOpen,iterations = 1)
        Opening =cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
        Closing =cv2.morphologyEx(Opening,cv2.MORPH_CLOSE,kernelClose)
        FinalMask = Closing

        return FinalMask


class Glove:

    def __init__(self):
        self.fingers = 0
        self.arearatio = 0
        self.gesture = 0

    def find_fingers(self, FinalMask):
        conts,h=cv2.findContours(FinalMask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        hull = [cv2.convexHull(c) for c in conts]

        try:
            cnt = max(conts, key = lambda x: cv2.contourArea(x))
            #approx the contour a little
            epsilon = 0.0005*cv2.arcLength(cnt,True)
            approx= cv2.approxPolyDP(cnt,epsilon,True)
            #make convex hull around hand
            hull = cv2.convexHull(cnt)
            #define area of hull and area of hand
            areahull = cv2.contourArea(hull)
            areacnt = cv2.contourArea(cnt)
            #find the percentage of area not covered by hand in convex hull
            self.arearatio=((areahull-areacnt)/areacnt)*100
            #find the defects in convex hull with respect to hand
            hull = cv2.convexHull(approx, returnPoints=False)
            defects = cv2.convexityDefects(approx, hull)
        except:
            print("No Contours found in FinalMask")

        # l = no. of defects
        l=0
        try:
            #code for finding no. of defects due to fingers
```

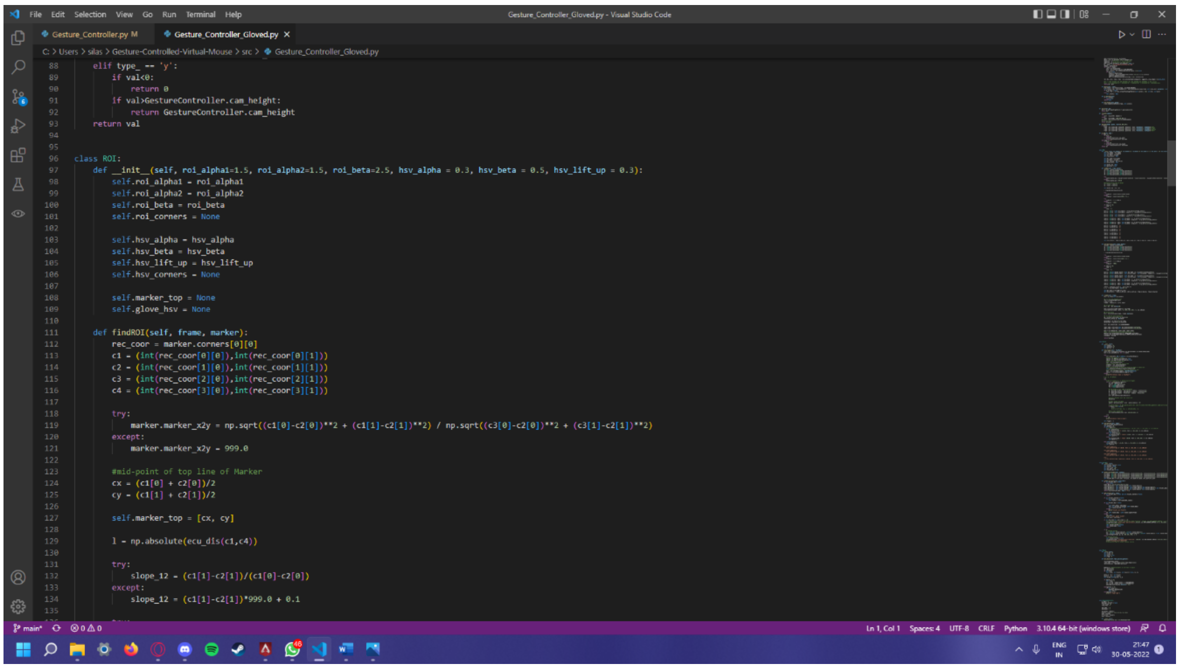


```
        else:
            sign = -1

        bot_rx = int(self.marker_top[0] + self.hsv_alpha * l * np.sqrt(1/(1+slope_12**2)))
        bot_ry = int(self.marker_top[1] - self.hsv_lift_up*l + self.hsv_alpha * slope_12 * l * np.sqrt(1/(1+slope_12**2)))

        bot_lx = int(self.marker_top[0] - self.hsv_alpha * l * np.sqrt(1/(1+slope_12**2)))
        bot_ly = int(self.marker_top[1] - self.hsv_lift_up*l - self.hsv_alpha * slope_12 * l * np.sqrt(1/(1+slope_12**2)))

        top_lx = int(bot_lx + sign * self.hsv_beta * l * np.sqrt(1/(1+slope_14**2)))
        top_ly = int(bot_ly + sign * self.hsv_beta * slope_14 * l * np.sqrt(1/(1+slope_14**2)))

        top_rx = int(bot_rx + sign * self.hsv_beta * l * np.sqrt(1/(1+slope_14**2)))
        top_ry = int(bot_ry + sign * self.hsv_beta * slope_14 * l * np.sqrt(1/(1+slope_14**2)))

        region = frame[top_ry:bot_ry , top_lx:bot_rx]
        b, g, r = np.mean(region, axis=(0, 1))

        self.hsv_glove = find_HSV([[r,g,b]])
        self.hsv_corners =  [(bot_lx,bot_ly), (bot_rx,bot_ry), (top_rx,top_ry), (top_lx,top_ly)]


    def cropROI(self, frame):
        pts = np.array(self.roi_corners)

        ## (1) Crop the bounding rect
        rect = cv2.boundingRect(pts)
        x,y,w,h = rect
        croped = frame[y:y+h, x:x+w].copy()

        ## (2) make mask
        pts = pts - pts.min(axis=0)

        mask = np.zeros(croped.shape[:2], np.uint8)
        cv2.drawContours(mask, [pts], -1, (255, 255, 255), -1, cv2.LINE_AA)

        ## (3) do bit-op
        dst = cv2.bitwise_and(croped, croped, mask=mask)

        ## (4) add the white background
        bg = np.ones_like(croped, np.uint8)*255
        cv2.bitwise_not(bg,bg, mask=mask)

        kernelOpen = np.ones((3,3),np.uint8)
        kernelClose = np.ones((5,5),np.uint8)

        hsv = cv2.cvtColor(dst, cv2.COLOR_BGR2HSV)

        lower_range = np.array([self.hsv_glove[0][0][0]//1-5,50,50])
```


File Edit Selection View Go Run Terminal Help
Gesture_Controller_Gloved.py - Visual Studio Code
Gesture_Controller.py M
Gesture_Controller_Gloved.py
C: > Users > silas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller_Gloved.py

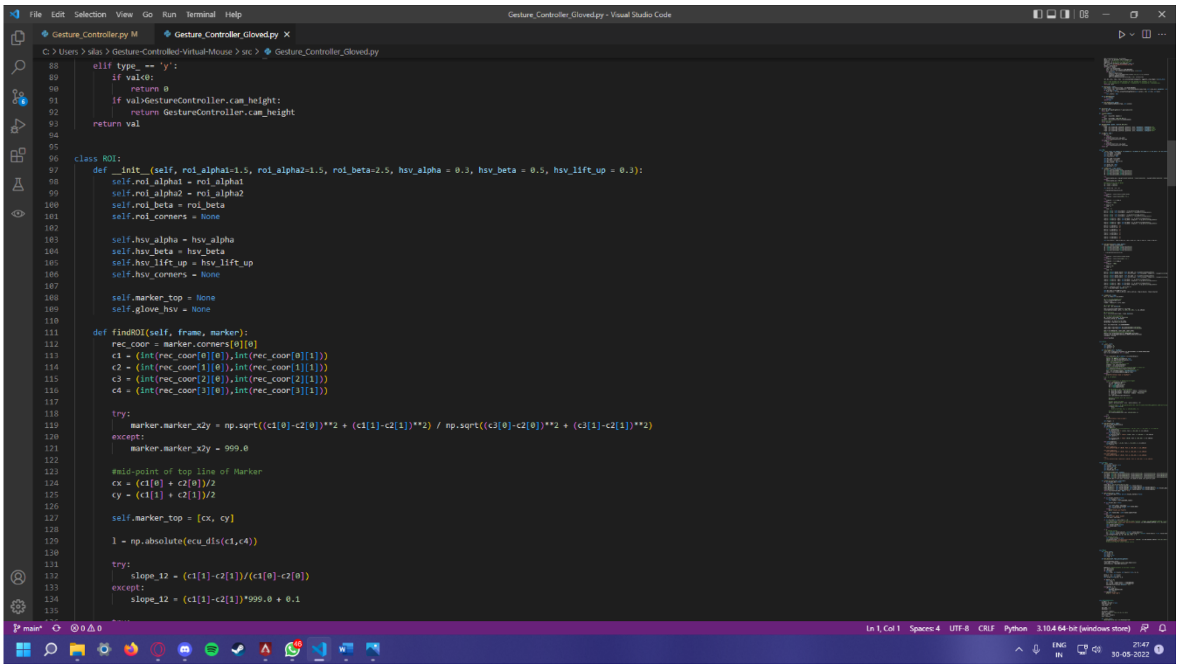
```
                s = (a+b+c)/2
                ar = math.sqrt(s*(s-a)*(s-b)*(s-c))

                #distance between point and convex hull
                d=(2*ar)/a

                # apply cosine rule here
                angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57

                # ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)
                if angle <= 90 and d>30:
                    l += 1
                    #cv2.circle(frame, far, 3, [255,255,255], -1)

                #draw lines around hand
                cv2.line(FinalMask,start, end, [255,255,255], 2)

            l+=1
        except:
            l = 0
            print("No Defects found in mask")

        self.fingers = l

    def find_gesture(self, frame):
        font = cv2.FONT_HERSHEY_SIMPLEX
        self.gesture = 0
        if self.fingers==1:
            #cv2.putText(frame, str(int(arearatio)), (10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
            if self.arearatio<15:
                cv2.putText(frame,'0',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                self.gesture = 0
            elif self.arearatio<25:
                cv2.putText(frame,'2 fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                self.gesture = 2
            else:
                cv2.putText(frame,'1 finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                self.gesture = 1

        elif self.fingers==2:
            cv2.putText(frame,'2',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
            self.gesture = 3
        '''
        elif self.fingers==3:
            #cv2.putText(frame,'3',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

        elif self.fingers==4:
            #cv2.putText(frame,'4',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

main* ⊗ 0 ⚠ 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit (windows store)
ENG IN 21:47 30-05-2022


File Edit Selection View Go Run Terminal Help
Gesture_Controller_Gloved.py - Visual Studio Code
Gesture_Controller.py M
Gesture_Controller_Gloved.py
C: > Users > silas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller_Gloved.py

```
    elif type_ == 'y':
        if val<0:
            return 0
        if val>GestureController.cam_height:
            return GestureController.cam_height
    return val


class ROI:
    def __init__(self, roi_alpha1=1.5, roi_alpha2=1.5, roi_beta=2.5, hsv_alpha = 0.3, hsv_beta = 0.5, hsv_lift_up = 0.3):
        self.roi_alpha1 = roi_alpha1
        self.roi_alpha2 = roi_alpha2
        self.roi_beta = roi_beta
        self.roi_corners = None

        self.hsv_alpha = hsv_alpha
        self.hsv_beta = hsv_beta
        self.hsv_lift_up = hsv_lift_up
        self.hsv_corners = None

        self.marker_top = None
        self.glove_hsv = None

    def findROI(self, frame, marker):
        rec_coor = marker.corners[0][0]
        c1 = (int(rec_coor[0][0]),int(rec_coor[0][1]))
        c2 = (int(rec_coor[1][0]),int(rec_coor[1][1]))
        c3 = (int(rec_coor[2][0]),int(rec_coor[2][1]))
        c4 = (int(rec_coor[3][0]),int(rec_coor[3][1]))

        try:
            marker.marker_x2y = np.sqrt((c1[0]-c2[0])**2 + (c1[1]-c2[1])**2) / np.sqrt((c3[0]-c2[0])**2 + (c3[1]-c2[1])**2)
        except:
            marker.marker_x2y = 999.0

        #mid-point of top line of Marker
        cx = (c1[0] + c2[0])/2
        cy = (c1[1] + c2[1])/2

        self.marker_top = [cx, cy]

        l = np.absolute(ecu_dis(c1,c4))

        try:
            slope_12 = (c1[1]-c2[1])/(c1[0]-c2[0])
        except:
            slope_12 = (c1[1]-c2[1])*999.0 + 0.1
```

main* ⊗ 0 ⚠ 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit (windows store)
ENG IN 21:47 30-05-2022

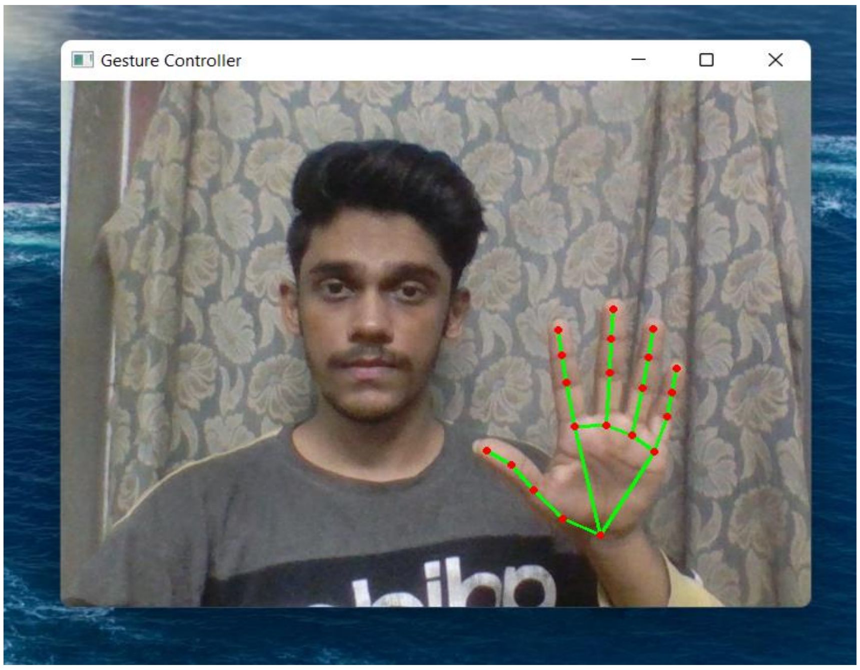


```python
        elif type_ == 'y':
            if val<0:
                return 0
            if val>GestureController.cam_height:
                return GestureController.cam_height
        return val


class ROI:
    def __init__(self, roi_alpha1=1.5, roi_alpha2=1.5, roi_beta=2.5, hsv_alpha = 0.3, hsv_beta = 0.5, hsv_lift_up = 0.3):
        self.roi_alpha1 = roi_alpha1
        self.roi_alpha2 = roi_alpha2
        self.roi_beta = roi_beta
        self.roi_corners = None

        self.hsv_alpha = hsv_alpha
        self.hsv_beta = hsv_beta
        self.hsv_lift_up = hsv_lift_up
        self.hsv_corners = None

        self.marker_top = None
        self.glove_hsv = None

    def findROI(self, frame, marker):
        rec_coor = marker.corners[0][0]
        c1 = (int(rec_coor[0][0]),int(rec_coor[0][1]))
        c2 = (int(rec_coor[1][0]),int(rec_coor[1][1]))
        c3 = (int(rec_coor[2][0]),int(rec_coor[2][1]))
        c4 = (int(rec_coor[3][0]),int(rec_coor[3][1]))

        try:
            marker.marker_x2y = np.sqrt((c1[0]-c2[0])**2 + (c1[1]-c2[1])**2) / np.sqrt((c3[0]-c2[0])**2 + (c3[1]-c2[1])**2)
        except:
            marker.marker_x2y = 999.0

        #mid-point of top line of Marker
        cx = (c1[0] + c2[0])/2
        cy = (c1[1] + c2[1])/2

        self.marker_top = [cx, cy]

        l = np.absolute(ecu_dis(c1,c4))

        try:
            slope_12 = (c1[1]-c2[1])/(c1[0]-c2[0])
        except:
            slope_12 = (c1[1]-c2[1])*999.0 + 0.1
```

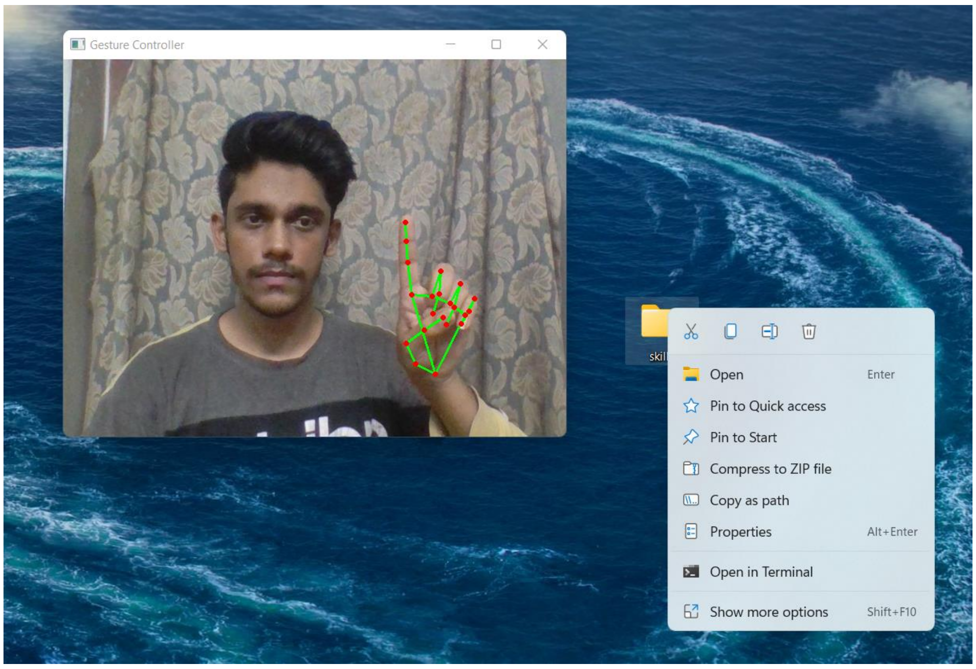
File Edit Selection View Go Run Terminal Help
Gesture_Controller_Gloved.py - Visual Studio Code
Gesture_Controller.py M
Gesture_Controller_Gloved.py
C: > Users > silas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller_Gloved.py
main*
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit (windows store)
21:47
30-05-2022

8.2 SCREENSHOTS OF THE EXECUTION

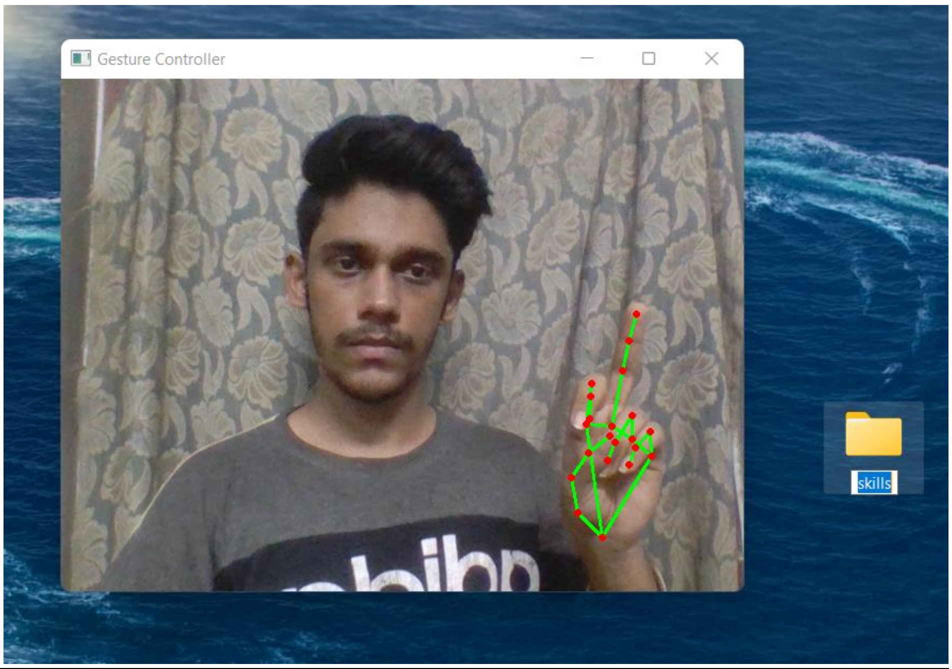
Neutral Gesture :



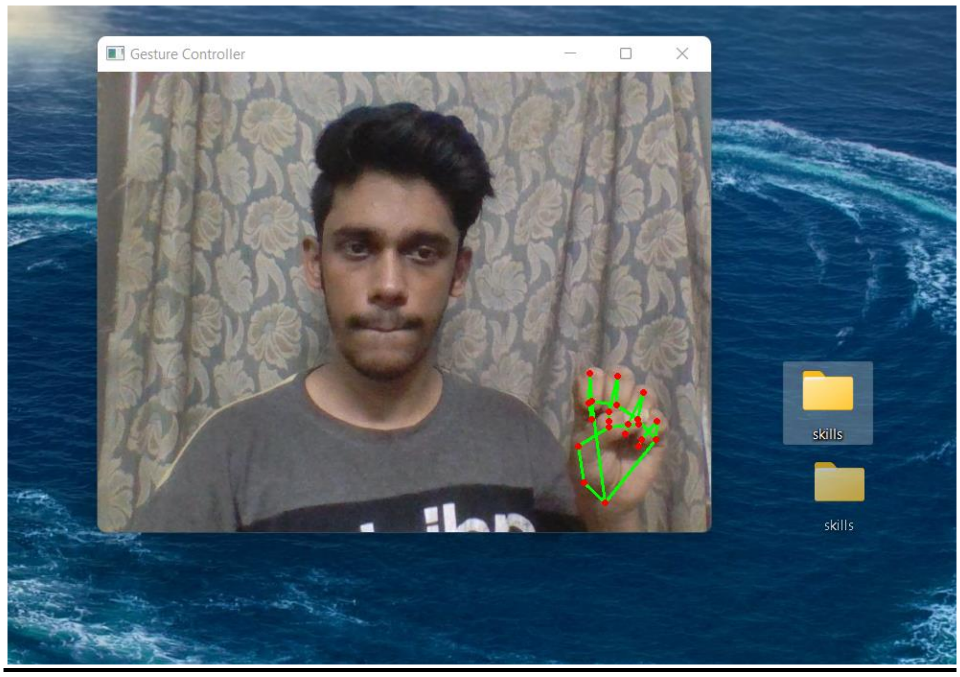
Left Click :



Right Click:



Drag and Drop :



<u>CHAPTER-9</u>

SYSTEM TESTING

SYSTEM TESTING

9.1 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 (i.e. black-box testing). 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 appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. 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 might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code. These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.
2. Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment throughthe use of white-box testing for any interactions of interfaces that are known to the programmer.
3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

9.2 Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration,system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

9.3 Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

9.4 Test cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

9.5 Test design techniques

- Decision table testing
- All-pairs testing
- State transition tables
- Equivalence partitioning
- Boundary value analysis

9.6 Performance testing

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

9.7 Unit testing

In computer programming, **unit testing** is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application.

In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation.

Testing will not catch every error in the program, since it cannot evaluate every execution path in any but the most trivial programs. The same is true for unit testing. Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance). Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behaviour. Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs.

9.8 Software testing

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and one with an outcome of "false". As a result, for every line of code written, programmers often need 3 to 5 lines of test code. This obviously takes time and its investment may not be worth the effort. There are also many problems that cannot easily be tested at all – for example those that are nondeterministic or involve multiple threads. In addition, code for a unit test is likely to be at least as buggy as the code it is testing. Fred Brooks in The Mythical Man-Month quotes: *never take two chronometers to sea. Always take one or three.* Meaning, if two chronometers contradict, how do you know which one is correct. Another challenge related to writing the unit tests is the difficulty of setting up realistic and useful tests. It is necessary to create relevant initial conditions so the part of the application being tested behaves like part of the complete system. If these initial conditions are not set correctly, the test will not be exercising the code in a realistic context, which diminishes the value and accuracy of unit test results. To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time. It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a process is not implemented and ingrained into the team's workflow, the application will evolve out of sync with the unit test suite, increasing false positives and reducing the effectiveness of the test suite.

9.9 Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes *what* the system does.

Functional testing typically involves five steps. The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs

9.10 Testing types

9.10.1 Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

9.10.2 Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

9.10.3 Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

9.10.4 Spike testing

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

9.10.5 Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour. A common example would be experimenting with different methods of load-balancing.

9.11 Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

9.12 Integration testing

Integration testing (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

9.13 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer.

9.14 Structure Testing

It is concerned with exercising the internal logic of a program and traversing particular execution paths.

9.15 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.

9.16 User acceptance Testing

- 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.
- 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.

<u>CHAPTER-10</u>

CONCLUSION

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.

<u>CHAPTER-11</u>

REFERENCES

REFERENCES

[1] Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj," Mouse Control using a Web Camera based on Color Detection",IJCTT,vol.9, Mar 2014.

[2] Angel, Neethu.P.S,"Real Time Static & Dynamic Hand Gesture Recognition", International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013.

[3] Q. Y. Zhang, F. Chen and X. W. Liu, "Hand Gesture Detection and Segmentation Based on Difference Background Image with Complex Background," Proceedings of the 2008 International Conference on Embedded Software and Systems, Sichuan, 29-31 July 2008, pp. 338- 343.

[4] Inside Facebook Reality Labs: Wrist-based interaction for the next computing platform [WWW Document], 2021 Facebook Technol. URL https://tech.fb.com/inside-facebook-realitylabs-wrist-based- inter action-for-the-next computing-platform/ (accessed 3.18.21).

[5] Danckert, J., Goodale, M.A., 2001. Superior performance for visually guided pointing in the lower visual field. Exp. Brain Res. 137, 303–308. https://doi.org/10.1007/s002210000653.

[6] Carlton, B., 2021. Hapt X Launches True-Contact Haptic Gloves For VR And Robotics. VR Scout. URL https://vrscout.com/news/haptx-truecontact-haptic-gloves-vr/ (accessed 3.10.21).

[7] Brenton, H., Gillies, M., Ballin, D., Chatting, D., 2005. D.: The uncanny valley: does it exist, in: In: 19th British HCI Group Annual Conference: Workshop on Human-Animated Character Interaction.

[8] Buckingham, G., Michela kakis, E.E., Cole, J., 2016. Perceiving and acting upon weight illusions in the absence of somatosensory information. J. Neuro physiol. 115, 1946–1953. https://doi.org/10.1152/jn.00587.2015.

[9] J. Katona, "A review of human–computer interaction and virtual reality research fields in cognitive Info Communications," Applied Sciences, vol. 11, no. 6, p. 2646, 2021.View at: Publisher Site | Google Scholar.

[10] D. L. Quam, “Gesture recognition with a Data Glove,” IEEE Conference on Aerospace and Electronics, vol.