

# **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

**Dr.L.JABASHEELA M.E.,Ph.D.**  
**HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NAZARATHPETTAI,  
POONAMALLEE,  
CHENNAI-600 123.

### SIGNATURE

**Mrs.A.KANCHANA M.E**  
**ASSISTANT PROFESSOR**  
**SUPERVISOR**

DEPARTMENT OF CSE,  
PANIMALAR ENGINEERING COLLEGE,  
NAZARATHPETTAI,  
POONAMALLEE,  
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**

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 INTRODUCTION	<b>2</b>
	1.2 OBJECTIVE	<b>3</b>
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
	2.1 LITERATURE SURVEY	
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>8</b>
	3.1 EXISTING SYSTEM	
	3.2 PROPOSED SYSTEM	<b>9</b>

<b>4</b>	<b>REQUIREMENT AND SPECIFICATION</b>	<b>10</b>
	4.1 HARDWARE REQUIREMENTS	11
	4.2 SOFTWARE REQUIREMENTS	11
<b>5</b>	<b>SOFTWARE DESCRIPTION</b>	<b>12</b>
	5.1 PYTHON (Programming Language)	13
	5.2 ANACONDA	14
	5.3 OPEN CV	14
	5.4 USE OF VIRTUAL ENVIRONMENT IN PYTHON	15
<b>6</b>	<b>SYSTEM ARCHITECTURE</b>	<b>16</b>
	6.1 WORK FLOWCHART DIAGRAM	
<b>7</b>	<b>SYSTEM DESIGN</b>	<b>19</b>
	7.1 UML DIAGRAM	
	7.2 MODULAR DIAGRAM	
	7.3 USE CASE DIAGRAM	

<b>8</b>	<b>IMPLEMENTATION</b>	<b>24</b>
	8.1 CODING	
	8.2 SCREENSHOTS OF THE EXECUTION	
 <b>9</b>	 <b>SYSTEM TESTING</b>	 <b>33</b>
	9.1 WHITE BOX TESTING	34
	9.2 BLACK BOX TESTING	34
	9.3 TEST PROCEDURES	34
	9.4 TEST CASES	35
	9.5 TEST DESIGN TECHNIQUES	35
	9.6 PERFORMANCE TESTING	35
	9.7 UNIT TESTING	36
	9.8 SOFTWARE TESTING	36
	9.9 FUNCTIONAL TESTING	36
	9.10 TESTING TYPES	37
	9.10.1 LOAD TESTING	37
	9.10.2 STRESS TESTING	38
	9.10.3 SOAK TESTING	38
	9.10.4 SPIKE TESTING	38
	9.10.5 CONFIGURATION TESTING	38

	9.11 ISOLATION TESTING	39
	9.12 INTEGRATION TESTING	39
	9.13 SYSTEM TESTING	40
	9.14 STRUCTURE TESTING	41
	9.15 OUTPUT TESTING	41
	9.16 USER ACCEPTANCE TESTING	41
<b>10</b>	<b>CONCLUSION</b>	<b>42</b>
<b>11</b>	<b>REFERENCES</b>	<b>44</b>



**CHAPTER-1**  
***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.

**CHAPTER 3**  
***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.

## **CHAPTER-4**

### ***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 SOFTWARE SPECIFICATION**

- 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 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.

## **5.3 OPEN CV**

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. 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 version 1.0 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

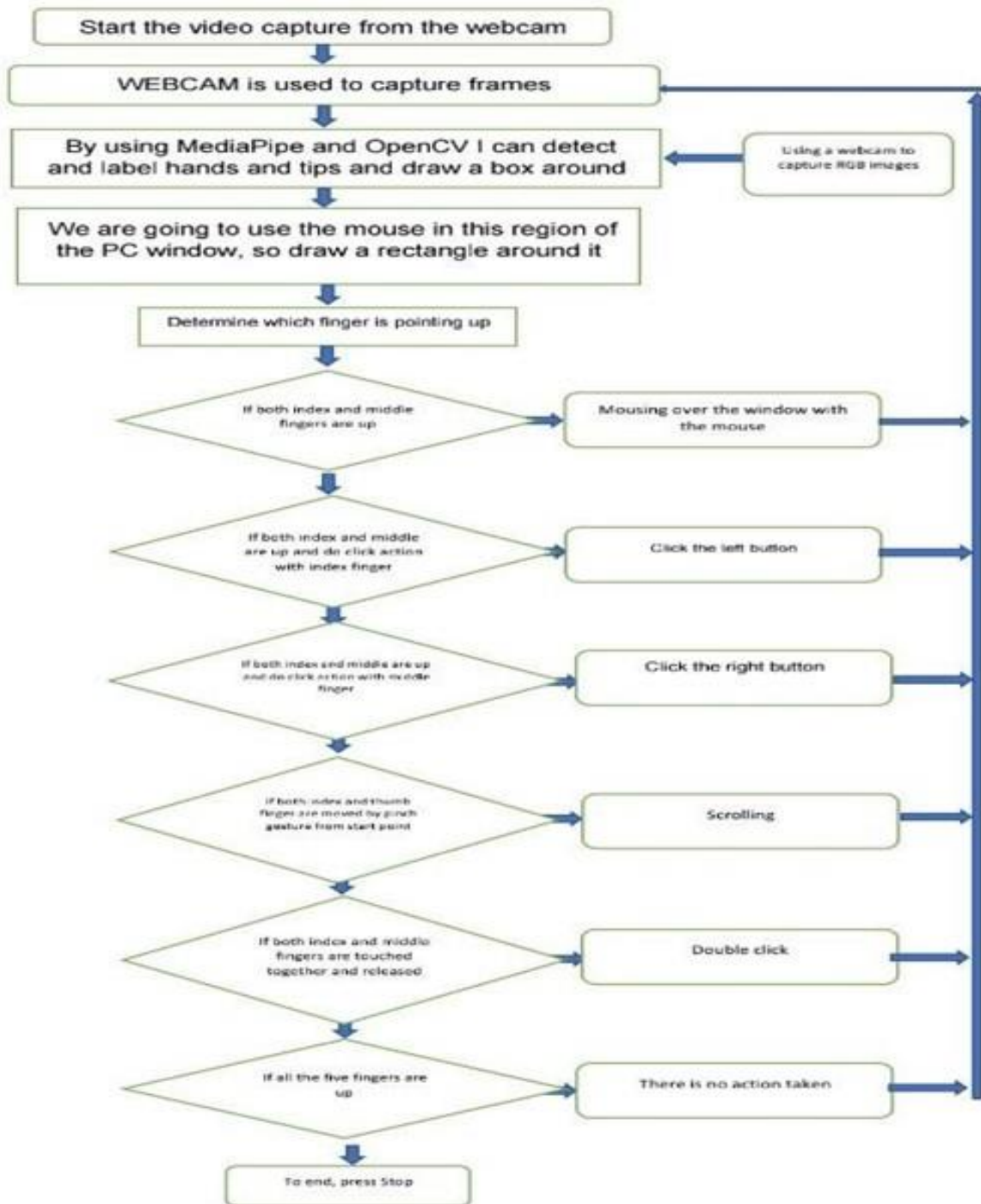
directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

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



**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.

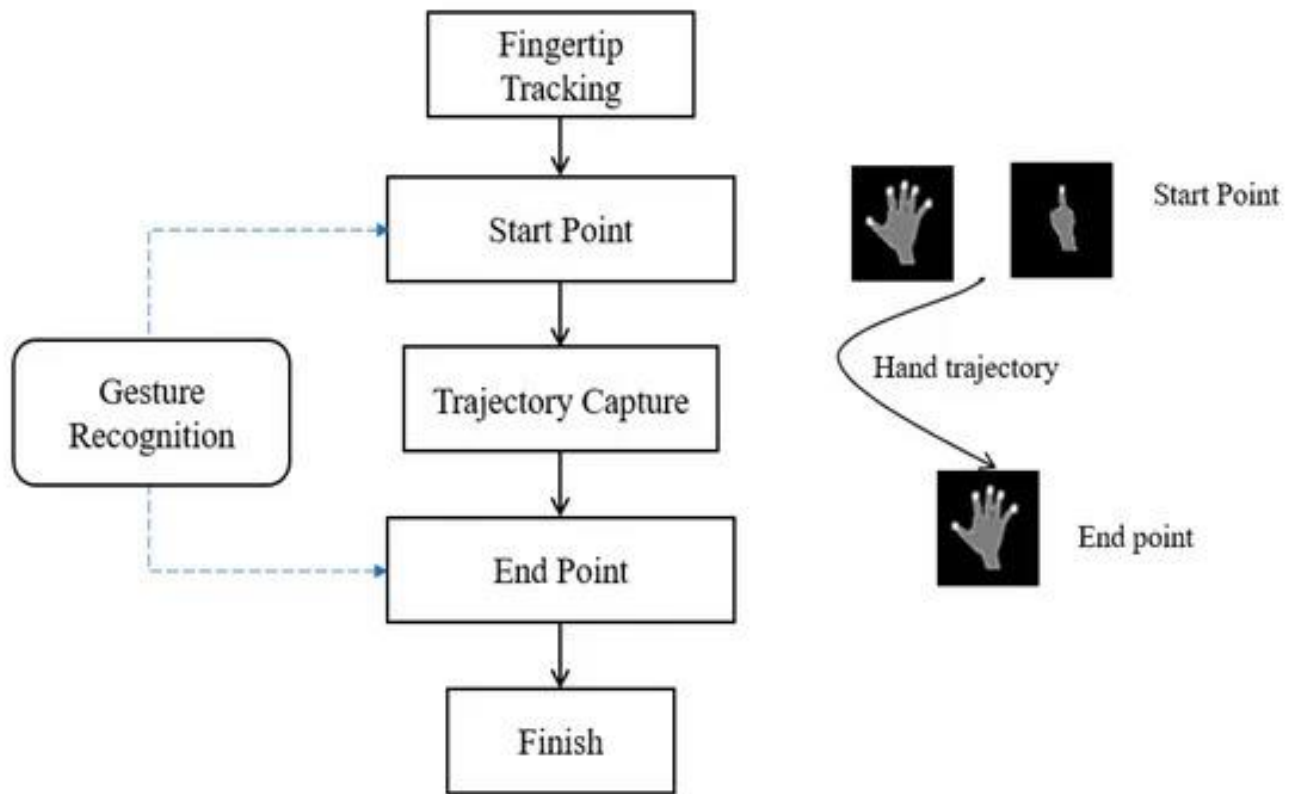
**Integration module:** This module integrates the gesture control virtual mouse with the operating system and other applications. It may use standard input/output protocols such as USB or Bluetooth to communicate with the computer and emulate the behavior of a traditional mouse.

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.

.

**CHAPTER-7**  
***SYSTEM DESIGN***

## 7.1 UML DIAGRAM



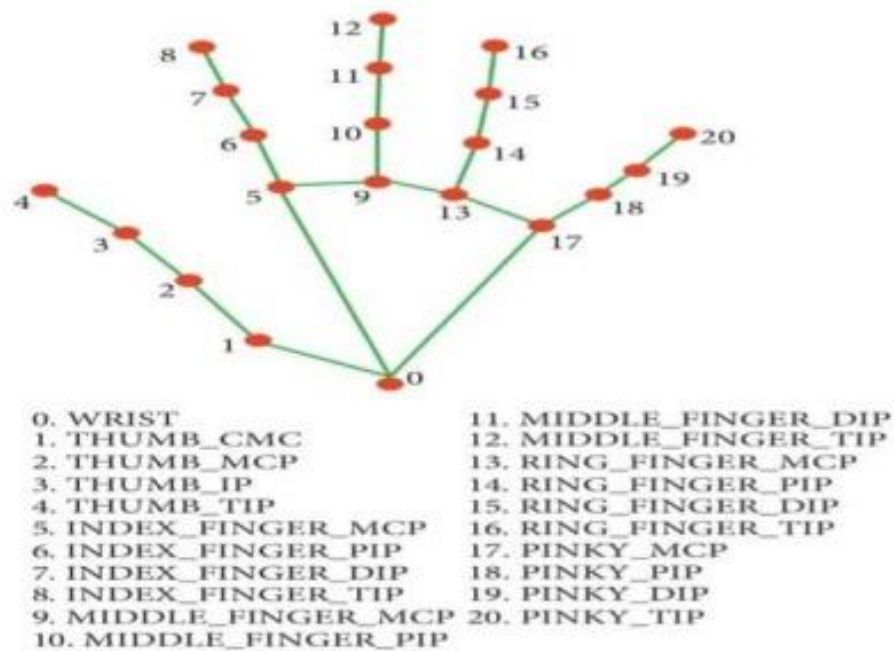
The UML diagram for gesture control virtual mouse, includes several classes, such as a Hand Recognition class and a Controller class. The Hand Recognition class would handle hand recognition and gesture detection using the Media pipe library, while the Controller class would handle mouse movement and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

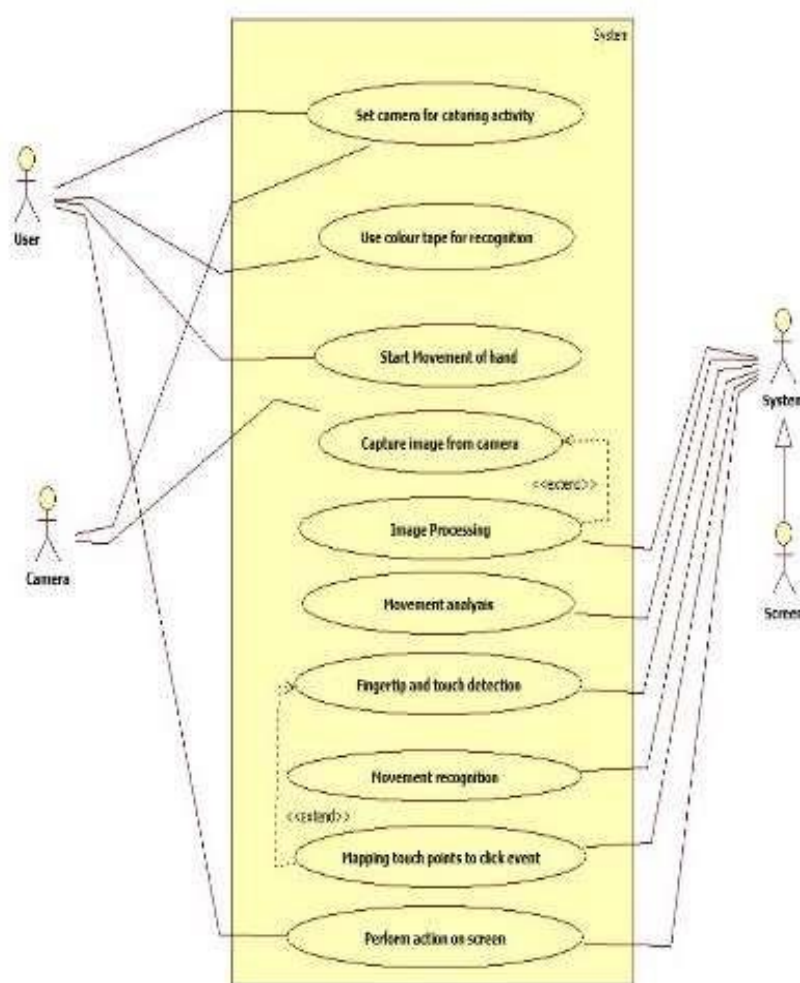
**Click Control Module:** This module is responsible for detecting the user's intention to click or double-click. It may use a variety of techniques such as analyzing the hand movement speed, time between clicks, or finger positions to determine the intended action.

**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



**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

```
Gesture_Controller.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
C:\Users> cd C:\Users> Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py
70
71 landmarks obtained from mediapipe.
72 hand_label : int
73 Represents multi-handedness corresponding to Enum 'HLabel'.
74
75
76 self.finger = 0
77 self_ori_gesture = Gest.PALM
78 self_prev_gesture = Gest.PALM
79 self.frame_count = 0
80 self.hand_result = None
81 self.hand_label = hand_label
82
83 def update_hand_result(self, hand_result):
84     self.hand_result = hand_result
85
86 def get_signed_dist(self, point):
87     """
88     returns signed euclidean distance between 'point'.
89
90     Parameters
91     -----
92     point : list containing two elements of type list/tuple which represents
93     landmark point.
94
95     Returns
96     -----
97     float
98
99     sign = -1
100     if self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y:
101         sign = -1
102     dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x)**2
103     dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y)**2
104     dist = math.sqrt(dist)
105     return dist*sign
106
107 def get_dist(self, point):
108     """
109     returns euclidean distance between 'point'.
110
111     Parameters
112     -----
113     point : list containing two elements of type list/tuple which represents
114     landmark point.
115
116     Returns
117     -----
118     float
119
120     """
121     dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x)**2
122     dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y)**2
123     dist = math.sqrt(dist)
124     return dist
125
126 def get_finger(self):
127     """
128     set 'finger' by computing ratio of distance between finger tip
129     , middle knuckle, base knuckle.
130
131     Returns
132     -----
133     None
134
135     """
136     if self.hand_result == None:
137         return
138
139     points = [[8,5,0],[12,9,0],[10,13,0],[20,17,0]]
140     self.finger = 0
141     self.finger = self.finger | 0 #turn
142     for idx,point in enumerate(points):
143         dist = self.get_signed_dist(point[:2])
144         dist2 = self.get_signed_dist(point[1:])
145
146         try:
147             ratio = round(dist/dist2,1)
148         except:
149             ratio = round(dist/0.01,1)
150
151         self.finger = self.finger << 1
152         if ratio > 0.5 :
153             self.finger = self.finger | 1
154
155     # Handling 'fluctuations due to noise'
156     def get_gesture(self):
157         """
158         returns int representing gesture corresponding to Enum 'Gest'.
159         sets 'frame_count', 'ori_gesture', 'prev_gesture',
160         handles fluctuations due to noise.
161
162         Returns
163         -----
164         int
165
166         """
167         if self.hand_result == None:
168             return Gest.PALM
169
170         current_gesture = Gest.PALM
171         if self.finger in [Gest.LAST3,Gest.LAST4] and self.get_dist([8,4]) < 0.05:
172             if self.hand_label == HLabel.MINOR :
173                 current_gesture = Gest.PINCH_MINOR
174             else:
175                 current_gesture = Gest.PINCH_MAJOR
176
177         return current_gesture
178
179 def main():
180     """
181     Main function to run the program.
182
183     """
184     # Initialize the Gesture_Controller object
185     gc = Gesture_Controller()
186     # Get the hand result from the camera
187     hand_result = get_hand_result()
188     # Update the hand result
189     gc.update_hand_result(hand_result)
190     # Get the finger
191     gc.get_finger()
192     # Get the gesture
193     gesture = gc.get_gesture()
194     # Print the gesture
195     print(gesture)
196
197 if __name__ == '__main__':
198     main()
```

```
Gesture_Controller.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
C:\Users> cd C:\Users> Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py
143
144 set 'finger' by computing ratio of distance between finger tip
145 , middle knuckle, base knuckle.
146
147 Returns
148 -----
149 None
150
151 If self.hand_result == None:
152     return
153
154 points = [[8,5,0],[12,9,0],[10,13,0],[20,17,0]]
155 self.finger = 0
156 self.finger = self.finger | 0 #turn
157 for idx,point in enumerate(points):
158     dist = self.get_signed_dist(point[:2])
159     dist2 = self.get_signed_dist(point[1:])
160
161     try:
162         ratio = round(dist/dist2,1)
163     except:
164         ratio = round(dist/0.01,1)
165
166     self.finger = self.finger << 1
167     if ratio > 0.5 :
168         self.finger = self.finger | 1
169
170 # Handling 'fluctuations due to noise'
171 def get_gesture(self):
172     """
173     returns int representing gesture corresponding to Enum 'Gest'.
174     sets 'frame_count', 'ori_gesture', 'prev_gesture',
175     handles fluctuations due to noise.
176
177     Returns
178     -----
179     int
180
181     """
182     if self.hand_result == None:
183         return Gest.PALM
184
185     current_gesture = Gest.PALM
186     if self.finger in [Gest.LAST3,Gest.LAST4] and self.get_dist([8,4]) < 0.05:
187         if self.hand_label == HLabel.MINOR :
188             current_gesture = Gest.PINCH_MINOR
189         else:
190             current_gesture = Gest.PINCH_MAJOR
191
192     return current_gesture
193
194 def main():
195     """
196     Main function to run the program.
197
198     """
199     # Initialize the Gesture_Controller object
200     gc = Gesture_Controller()
201     # Get the hand result from the camera
202     hand_result = get_hand_result()
203     # Update the hand result
204     gc.update_hand_result(hand_result)
205     # Get the finger
206     gc.get_finger()
207     # Get the gesture
208     gesture = gc.get_gesture()
209     # Print the gesture
210     print(gesture)
211
212 if __name__ == '__main__':
213     main()
```

```

File Edit Selection View Go Run Terminal Help
Gesture_Controller.py - Visual Studio Code

C:\Users> slas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py

212     self.frame_count = 0
213
214     self.prev_gesture = current_gesture
215
216     if self.frame_count > 4:
217         self.on1_gesture = current_gesture
218         return self.on1_gesture
219
220 # Executes commands according to detected gestures
221 class Controller:
222     """
223     Executes commands according to detected gestures.
224
225     Attributes
226     """
227     tx_old : int
228     # previous mouse location x coordinate
229     ty_old : int
230     # previous mouse location y coordinate
231     flag : bool
232     # True if V gesture is detected
233     grabflag : bool
234     # true if FIST gesture is detected
235     pinchmajorflag : bool
236     # true if PINCH gesture is detected through MAJOR hand,
237     # on x-axis 'Controller.changesystembrightness',
238     # on y-axis 'Controller.changesystemvolume'.
239     pinchminorflag : bool
240     # true if PINCH gesture is detected through MINOR hand,
241     # on x-axis 'Controller.scrollHorizontal',
242     # on y-axis 'Controller.scrollVertical'.
243     pinchstartxcoord : int
244     # x coordinate of hand landmark when pinch gesture is started.
245     pinchstartycoord : int
246     # y coordinate of hand landmark when pinch gesture is started.
247     pinchdirectionflag : bool
248     # true if pinch gesture movement is along x-axis,
249     # otherwise false
250     prevpinchlv : int
251     # stores quantized magnitude of prev pinch gesture displacement, from
252     # starting position
253     pinchlv : int
254     # stores quantized magnitude of pinch gesture displacement, from
255     # starting position
256     framecount : int
257     # stores no. of frames since 'pinchlv' is updated.
258     prev_hand : tuple
259     # stores (x, y) coordinates of hand in previous frame.

```

```

File Edit Selection View Go Run Terminal Help
Gesture_Controller.py - Visual Studio Code

C:\Users> slas > Gesture-Controlled-Virtual-Mouse > src > Gesture_Controller.py

346     Controller.prev_hand = x,y
347     delta_x = x - Controller.prev_hand[0]
348     delta_y = y - Controller.prev_hand[1]
349
350     distsq = delta_x**2 + delta_y**2
351     ratio = 1
352     Controller.prev_hand = [x,y]
353
354     if distsq <= 25:
355         ratio = 0
356     elif distsq <= 900:
357         ratio = 0.07 * (distsq ** (1/2))
358     else:
359         ratio = 2.1
360     x, y = x_old + delta_x*ratio, y_old + delta_y*ratio
361     return (x,y)
362
363 def pinch_control_init(hand_result):
364     """Initializes attributes for pinch gesture."""
365     Controller.pinchstartxcoord = hand_result.landmark[8].x
366     Controller.pinchstartycoord = hand_result.landmark[8].y
367     Controller.pinchlv = 0
368     Controller.prevpinchlv = 0
369     Controller.framecount = 0
370
371 # Hold final position for 5 frames to change status
372 def pinch_control(hand_result, controlHorizontal, controlVertical):
373     """
374     calls 'controlHorizontal' or 'controlVertical' based on pinch flag,
375     'framecount' and sets 'pinchlv'.
376
377     Parameters
378     """
379     hand_result : Object
380     landmarks obtained from mediapipe.
381     controlHorizontal : callback function associated with horizontal
382     pinch gesture.
383     controlVertical : callback function associated with vertical
384     pinch gesture.
385
386     Returns
387     """
388     None
389
390     if Controller.framecount == 5:
391         Controller.framecount = 0
392         Controller.pinchlv = Controller.prevpinchlv
393

```

```
File Edit Selection View Go Run Terminal Help
Gesture_Controller - Visual Studio Code

C:\Users> sias> Gesture-Controller-Virtual-Mouse> src> Gesture_Controller

288     return dist
289
290
291     def changesystembrightness():
292         """sets system brightness based on 'Controller.pinchv'."""
293         currentBrightnessv = sbcontrol.get_brightness(display=0)/100.0
294         currentBrightnessv = Controller.pinchv/50.0
295         if currentBrightnessv > 1.0:
296             currentBrightnessv = 1.0
297         elif currentBrightnessv < 0.0:
298             currentBrightnessv = 0.0
299         sbcontrol.fade_brightness(int(100*currentBrightnessv) , start = sbcontrol.get_brightness(display=0))
300
301     def changesystemvolume():
302         """sets system volume based on 'Controller.pinchv'."""
303         devices = AudioUtilities.GetSpeakers()
304         interface = devices.Activate([AudioEndpointVolume._iid_, CLSCTX_ALL, None])
305         volume = cast(interface, POINTER(AudioEndpointVolume))
306         currentVolumev = volume.GetMasterVolumeLevelScalar()
307         currentVolumev = Controller.pinchv/50.0
308         if currentVolumev > 1.0:
309             currentVolumev = 1.0
310         elif currentVolumev < 0.0:
311             currentVolumev = 0.0
312         volume.SetMasterVolumeLevelScalar(currentVolumev, None)
313
314     def scrollVertical():
315         """scrolls on screen vertically."""
316         pyautogui.scroll(120 if Controller.pinchv>0.0 else -120)
317
318     def scrollHorizontal():
319         """scrolls on screen horizontally."""
320         pyautogui.keyDown('shift')
321         pyautogui.keyDown('ctrl')
322         pyautogui.scroll(120 if Controller.pinchv>0.0 else 120)
323         pyautogui.keyUp('ctrl')
324         pyautogui.keyUp('shift')
325
326     # Locate Hand to get Cursor Position
327     # Stabilize cursor by Dampening
328     def get_position(hand_result):
329         """
330         returns coordinates of current hand position.
331
332         locates hand to get cursor position also stabilize cursor by
333         dampening jerky motion of hand.
334
335         Returns
336         """
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
28
```

```

File Edit Selection View Go Run Terminal Help
Gesture_Controller_Glovedpy - Visual Studio Code
C:\Users> sbs> Gesture-Controlled-Virtual-Mouse> src> Gesture_Controller_Glovedpy

238 hsv = cv2.cvtColor(dst, cv2.COLOR_BGR2HSV)
239
240 inner_range = np.array([self.hsv_glove[0][0]/1-5,50,50])
241
242 upper_range = np.array([self.hsv_glove[0][0]/1+5,255,255])
243
244 mask = cv2.inRange(hsv, inner_range, upper_range)
245
246 #mask = cv2.dilate(mask, kernelOpen, iterations = 1)
247 Opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernelOpen)
248 Closing = cv2.morphologyEx(Opening, cv2.MORPH_CLOSE, kernelClose)
249 FinalMask = Closing
250
251 return FinalMask
252
253 class Glove:
254
255     def __init__(self):
256         self.fingers = 0
257         self.arcratio = 0
258         self.gesture = 0
259
260     def find_fingers(self, FinalMask):
261         cnts, hcv2 = cv2.findContours(FinalMask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
262         hull = [cv2.convexHull(c) for c in cnts]
263
264         try:
265             cnt = max(cnts, key = lambda x: cv2.contourArea(x))
266             #approx the contour a little
267             epsilon = 0.0005*cv2.arcLength(cnt, True)
268             approx = cv2.approxPolyDP(cnt, epsilon, True)
269             #mask convex hull around hand
270             hull = cv2.convexHull(cnt)
271             #define area of hull and area of hand
272             areahull = cv2.contourArea(hull)
273             areacnt = cv2.contourArea(cnt)
274             #find the percentage of area not covered by hand in convex hull
275             self.arcratio = (areahull - areacnt) / areacnt * 100
276             #find the defects in convex hull with respect to hand
277             hull = cv2.convexHull(approx, returnPoints=False)
278             defects = cv2.convexityDefects(approx, hull)
279         except:
280             print("No Contours found in FinalMask")
281
282         # l = no. of defects
283         l = 0
284         try:
285             #mode for finding no. of defects due to fingers
286             for i in range(0, len(defects)):

```

```

File Edit Selection View Go Run Terminal Help
Gesture_Controller_Glovedpy - Visual Studio Code
C:\Users> sbs> Gesture-Controlled-Virtual-Mouse> src> Gesture_Controller_Glovedpy

193 else:
194     sign = -1
195
196 bot_rx = int(self.marker_top[0] + self.hsv_alpha * 1 * np.sqrt(1/(1+slope_12**2)))
197 bot_ry = int(self.marker_top[1] - self.hsv_lift*up*1 + self.hsv_alpha * slope_12 * 1 * np.sqrt(1/(1+slope_12**2)))
198
199 bot_lx = int(self.marker_top[0] - self.hsv_alpha * 1 * np.sqrt(1/(1+slope_12**2)))
200 bot_ly = int(self.marker_top[1] - self.hsv_lift*up*1 - self.hsv_alpha * slope_12 * 1 * np.sqrt(1/(1+slope_12**2)))
201
202 top_lx = int(bot_lx + sign * self.hsv_beta * 1 * np.sqrt(1/(1+slope_14**2)))
203 top_ly = int(bot_ly + sign * self.hsv_beta * slope_14 * 1 * np.sqrt(1/(1+slope_14**2)))
204
205 top_rx = int(bot_rx + sign * self.hsv_beta * 1 * np.sqrt(1/(1+slope_14**2)))
206 top_ry = int(bot_ry + sign * self.hsv_beta * slope_14 * 1 * np.sqrt(1/(1+slope_14**2)))
207
208 region = frame[top_ry:bot_ry, top_lx:bot_rx]
209 b, g, r = np.mean(region, axis=(0, 1))
210
211 self.hsv_glove = find_HSV([r,g,b])
212 self.hsv_corners = [(bot_lx,bot_ly), (bot_rx,bot_ry), (top_rx,top_ry), (top_lx,top_ly)]
213
214
215 def cropROI(self, frame):
216     pts = np.array(self.roi_corners)
217
218     ## (1) Crop the bounding rect
219     rect = cv2.boundingRect(pts)
220     x,y,w,h = rect
221     cropped = frame[y:y+h, x:x+w].copy()
222
223     ## (2) make mask
224     pts = pts - pts.min(axis=0)
225
226     mask = np.zeros(cropped.shape[:2], np.uint8)
227     cv2.drawContours(mask, [pts], -1, (255, 255, 255), -1, cv2.LINE_AA)
228
229     ## (3) do bit-op
230     dst = cv2.bitwise_and(cropped, cropped, mask=mask)
231
232     ## (4) add the white background
233     bg = np.ones_like(cropped, np.uint8)*255
234     cv2.bitwise_not(bg, bg, mask=mask)
235
236     kernelOpen = np.ones((3,3), np.uint8)
237     kernelClose = np.ones((5,5), np.uint8)
238
239     hsv = cv2.cvtColor(dst, cv2.COLOR_BGR2HSV)
240
241     inner_range = np.array([self.hsv_glove[0][0]/1-5,50,50])

```



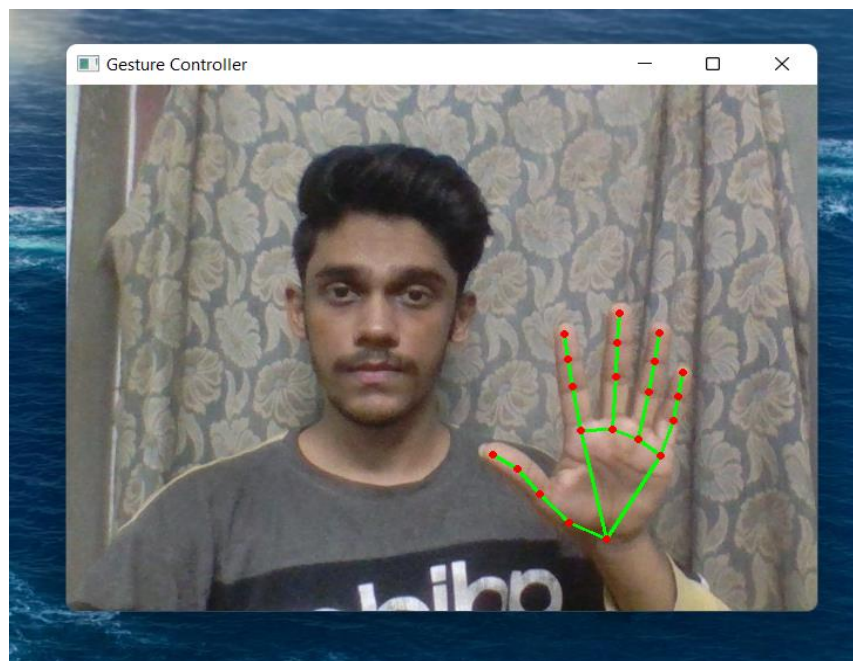
```

88 elif type == 'y':
89     if val<0:
90         return 0
91     if val>GestureController.cam_height:
92         return GestureController.cam_height
93     return val
94
95
96 class ROI:
97     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):
98         self.roi_alpha1 = roi_alpha1
99         self.roi_alpha2 = roi_alpha2
100         self.roi_beta = roi_beta
101         self.roi_corners = None
102
103         self.hsv_alpha = hsv_alpha
104         self.hsv_beta = hsv_beta
105         self.hsv_lift_up = hsv_lift_up
106         self.hsv_corners = None
107
108         self.marker_top = None
109         self.glove_hsv = None
110
111     def findROI(self, frame, marker):
112         rec_coord = marker.corners[0][0]
113         c1 = (int(rec_coord[0]),int(rec_coord[0][1]))
114         c2 = (int(rec_coord[1][0]),int(rec_coord[1][1]))
115         c3 = (int(rec_coord[2][0]),int(rec_coord[2][1]))
116         c4 = (int(rec_coord[3][0]),int(rec_coord[3][1]))
117
118         try:
119             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)
120         except:
121             marker.marker_x2y = 999.0
122
123         #Mid-point of top line of Marker
124         cx = (c1[0] + c2[0])/2
125         cy = (c1[1] + c2[1])/2
126         self.marker_top = [cx, cy]
127
128         l = np.absolute(ecu_dis(c1,c4))
129
130         try:
131             slope_12 = (c1[1]-c2[1])/(c1[0]-c2[0])
132         except:
133             slope_12 = (c1[1]-c2[1])*999.0 + 0.1
134
135

```

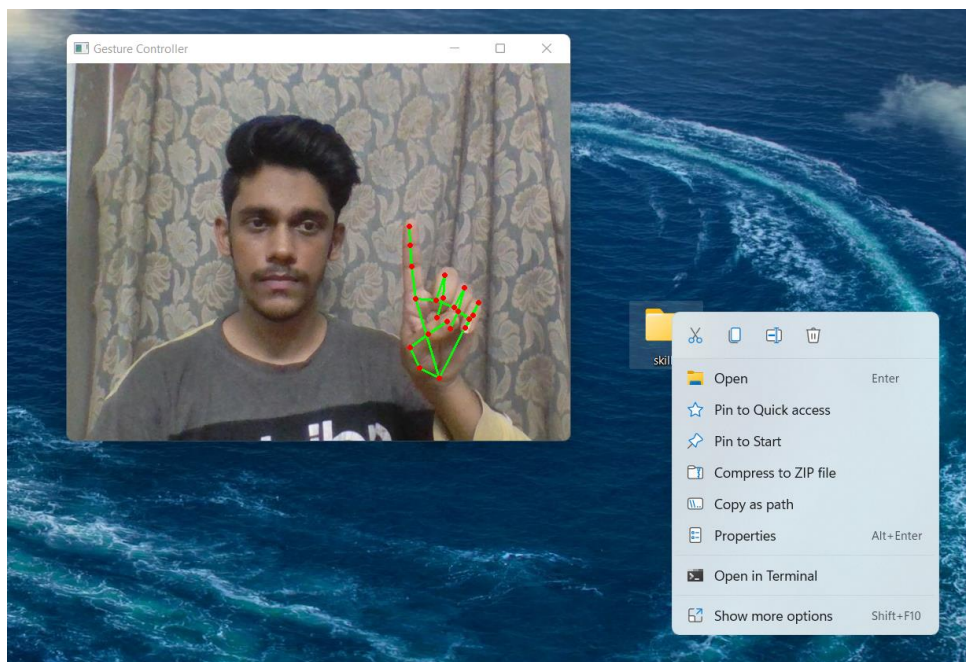
## 8.2 SCREENSHOTS OF THE EXECUTION

Neutral Gesture :

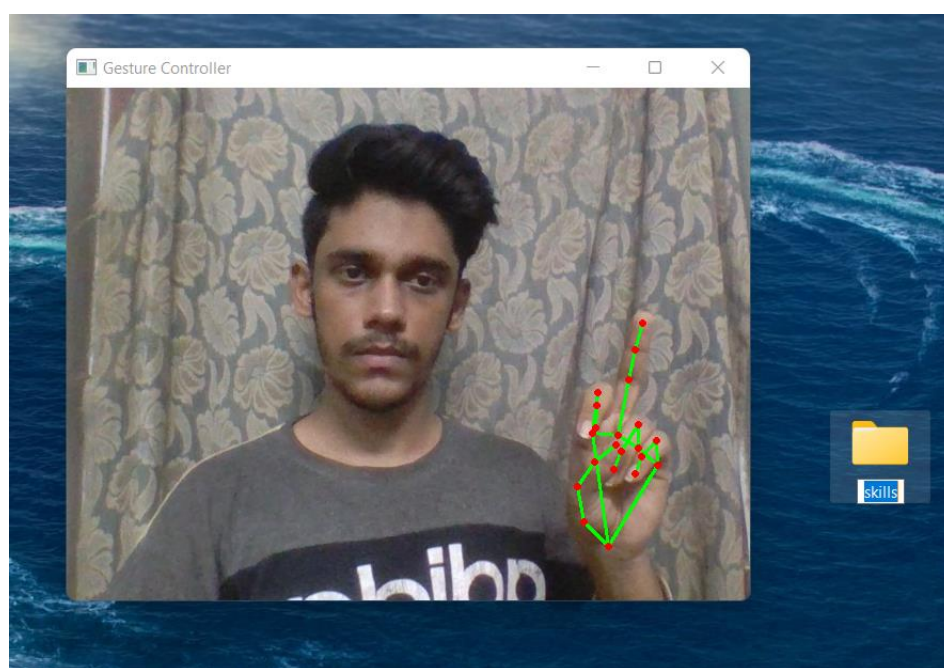




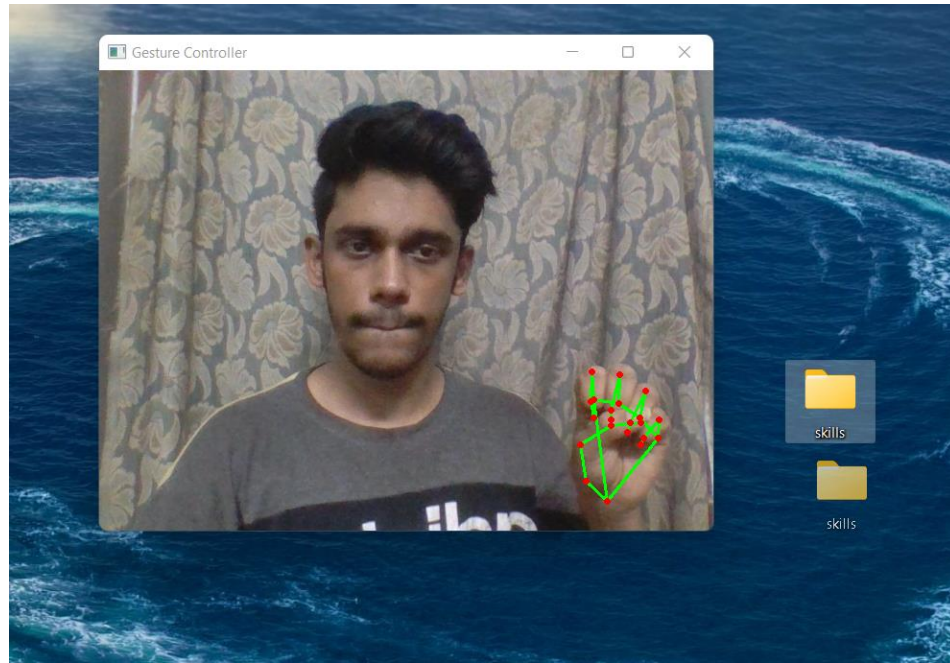
## Left Click :



## Right Click:



## Drag and Drop :





**CHAPTER-9**  
***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 through the 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 handing 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.

**CHAPTER-10**  
**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.

**CHAPTER-11**  
***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/](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.