# AI VIRTUAL MOUSE USING HAND GESTURE

**A MINI PROJECT REPORT**

***Submitted by***


| | |
|---|---|
| **ASHOK A T** | **510420104015** |
| **AJAYDEV M S** | **510420104009** |
| **CHANDRU V** | **510420104020** |
| **AASHIP J** | **510420104001** |


***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***In***

**COMPUTER SCIENCE AND ENGINEERING**


**ARUNAI ENGINEERING COLLEGE**
**TIRUVANNAMALAI- 606603**


**ANNA UNIVERSITY:: CHENNAI 600 025**

**MAY 2023**

## ANNA UNIVERSITY: CHENNAI 600025

## BONAFIDE CERTIFICATE

Certified that this project report **"A NEW METHOD FOR AI VIRTUAL MOUSE USING HAND GESTURE"** is the bonafide work of **"ASHOK A T (510420104015), AJAYDEV M S (510420104009), CHANDRU V (510420104020), AASHIP J (510420104001)"** who carried out the project work under my supervision.

**SIGNATURE OF SUPERVISOR**

**Mr. R. SURESH, M.E.,**

Assistant Professor,

Computer Science and Engineering,

Arunai Engineering College,

Tiruvannamalai-606 603.

**SIGNATURE OF HOD**

**Mrs. V. UMADEVI, M.E.,**

Head of the Department,

Computer Science and Engineering,

Arunai Engineering College,

Tiruvannamalai-606 603.

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# CERTIFICATE OF EVALUATION

**COLLEGE NAME** : 5104-ARUNAI ENGINEERING COLLEGE

**BRANCH** : B.E.COMPUTER SCIENCE AND ENGINEERING

**SEMESTER** : VI (2020-2024)

**SUBJECT CODE &NAME** : CS8611-Mini Project

**DATE OF EXAMINATION :** 03-06-2023

| NAME OF THE STUDENTS | REGISTER NUMBER | TITLE OF THE PROJECT | NAME OF GUIDE WITH DESIGNATION |
|---|---|---|---|
| ASHOK A T<br>AJAYDEV M S<br>CHANDRU V<br>AASHIP J | 510420104015<br>510420104009<br>510420104020<br>510420104001 | A NEW METHOD FOR AI VIRTUAL MOUSE USING HAND GESTURE | Mr. R.SURESH, M.E.,<br>Assistant Professor, Department of Computer Science and Engineering |

The report of the **PROJECT WORK** submitted for the fulfillment of Bachelor of Engineering degree in **COMPUTER SCIENCE ENGINEERING** of Anna University was evaluated and confirmed to be reports of the work done by the above students.

**SIGNATURE OF SUPERVISOR**

**Mr. R SURESH, M.E.,**

Assistant Professor,

Computer Science and Engineering,

Arunai Engineering College,

Tiruvannamalai-606 603.

**SIGNATURE OF HOD**

**Mrs. V. UMADEVI, M.E.,**

Head of the Department,

Computer Science and Engineering,

Arunai Engineering College,

Tiruvannamalai-606 603.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

It is our duty to thank the GOD Almighty and my beloved parents and teachers for their persistent blessing and constant encouragement to reach this level of what I am today.

A project of this nature needs co-operation and support from many for successful completion. In this regard I am fortunate to express my heartfelt thanks to our beloved Founder Chairman **Mr. E.V.VELU**, Chair Person **Mrs. SANKARI VELU** and **Er. E.V.KUMARAN M.E.**, Vice Chairman, for providing necessary facilities with high-class environment throughout the course.

We take the privilege of expressing my sincere thanks to our beloved principal, **Dr.R.RAVICHANRAN Ph.D,** for granting permission to undertake the project and I also express my heartfelt thanks to Registrar **Dr.R.SATHIYASEELAN Ph.D**., for their advice in accomplishing this project work.

We are most grateful to **Mrs V.UMADEVI M.E,** Head of the Department, Department of computer Science and Engineering, who has given me both moral and technical support adding experience to the job we have undertaken.

We thank to our project guide to **Mr R.SURESH M.E**, Assistant Professor, Department of Computer Science and Engineering, for her astonishing guidance and encouragement for the successful completion of this project. They held me to the highest standards of quality and accuracy.

Last but not least I would like to thank my family members, friends, teaching and non-teaching staffs, who has assisted us directly or indirectly throughout this project.

# சுருக்கம்

## AI மெய்நிகர் மவுஸுக்கான புதிய முறை

இந்த திட்டங்கள் மனித கணினி தொடர்பு (HCL)க்கான அணுகுமுறையை ஊக்குவிக்கிறது. மவுஸ் செயல்பாட்டைக் கட்டுப்படுத்த நிகழ்நேர கேமராவைப் பயன்படுத்துகிறோம். எங்கள் முன்மொழியப்பட்ட திட்டம் கை சைகை அடிப்படையிலான அமைப்பில் உள்ளது, இது பயனர்களை கை சைகையைப் பயன்படுத்தி டெஸ்க்டாப் மவுஸ் அசைவுகளைக் கட்டுப்படுத்த அனுமதிக்கிறது. கை சைகை அசைவுகளைக் கண்டறிய, எங்கள் கணினி டெஸ்க்டாப் வெப்கேமைப் பயன்படுத்துகிறது. பாரம்பரிய அல்லது நிலையான சாதனங்களைக் காட்டிலும் எளிய கேமரா அல்லது வெப்கேம் மூலம் மவுஸ் கர்சர் செயல்பாடுகளைக் கட்டுப்படுத்துவதே குறிக்கோள். ஒரு கேமராவை மட்டுமே பயன்படுத்தி, விர்ச்சுவல் மவுஸ் பயனருக்கும் இயந்திரத்திற்கும் இடையே ஒரு உள்கட்டமைப்பை வழங்குகிறது. எந்தவொரு இயந்திர அல்லது இயற்பியல் சாதனங்களும் தேவையில்லாமல் ஒரு இயந்திரத்துடன் தொடர்பு கொள்ள பயனரை இது செயல்படுத்துகிறது, மேலும் மவுஸ் செயல்பாடுகளைக் கட்டுப்படுத்தவும் அனுமதிக்கிறது. திட்டத்தின் டொமைன் AI/ML ஆகும். இந்த திட்டங்களில் நாங்கள் பயன்படுத்திய நிரலாக்க மொழி பைதான். இந்த AI மெய்நிகர் மவுஸ் திட்டம் கணினி பார்வையின் கருத்தை அடிப்படையாகக் கொண்டது.

# ABSTRACT

## A NEW METHOD FOR AI VIRTUAL MOUSE

This projects promotes an approach for the human computer interaction(HCL). Where we use real time camera for controlling the mouse function. Our proposed project is on hand gesture-based system that allows users to control desktop mouse movements using hand gesture. To detect hand gesture movements, our system makes use of a desktop webcam. The goal is to control mouse cursor functions with a simple camera or webcam rather than a traditional or standard devices. Using only a camera, the Virtual Mouse provides an infrastructure between the user and the machine. It enables the user to interact with a machine without the need for any mechanical or physical devices, and even allows to control mouse functions. The domain of the project is AI/ML. The programming language we used in this projects is python. This ai virtual mouse project is based on the concept of computer vision.

# TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF ABBREVATION

| | | |
|---|---|---|
| GUI | - | Graphical User Interface |
| LED | - | Light Emitting Diodes |
| HCI | - | Human Computer Interaction |
| MHI | - | Motion History Images |
| OpenCV | - | Open Computer Vision |
| HSV | - | Hue, Saturation, and Values |
| RGB | - | Red, Green, and Blue |
| NUI | - | Natural User Interface |
| IDE | - | Integrated Development Environment |

# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW

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.

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

Python programming language is used for developing the AI virtual mouse system, and also, OpenCV which is the library for computer vision is used in the AI virtual mouse system. In the proposed AI virtual mouse system, the model makes use of the MediaPipe package for the tracking of the hands and for tracking of the tip of the hands, and also, Pynput, Autopy, and PyAutoGUI packages were used for moving around the window screen of the computer for performing functions such as 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.

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.

## 1.3 PROBLEM DESCRIPTION AND OVERVIEW

The proposed AI virtual mouse system can be used to overcome problems in the real world such as situations where there is no space to use a physical mouse and also for the persons who have problems in their hands and are not able to control a physical mouse. Also during the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome

these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera.

## 1.4 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.

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.

## 1.5 MACHINE LEARNING

Machine learning is an application of artificial intelligence that uses statistical techniques to enable computers to learn and make decisions without being explicitly programmed. It is predicated on the notion that computers can learn from data, spot patterns, and make judgments with little assistance from humans.

It is a subset of Artificial Intelligence. It is the study of making machines more human-like in their behavior and decisions by giving them the ability to learn and develop their own programs. This is done with minimum human intervention, i.e., no explicit programming. The learning process is automated and improved based on the experiences of the machines throughout the process.

There are seven steps of machine learning:

1. Gathering data
2. Preparing that data
3. Choosing a model
4. Training
5. Evaluation
6. Hyperparameter Tuning
7. Prediction

## 1.6 DEEP LEARNING

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier.

At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

To understand deep learning, imagine a toddler whose first word is dog. The toddler learns what a dog is -- and is not -- by pointing to objects and saying the word dog. The parent says, "Yes, that is a dog," or, "No, that is not a dog." As the toddler continues to point to objects, he becomes more aware of the features that all dogs possess. What the toddler does, without knowing it, is clarify a complex abstraction -- the concept of dog -- by building a hierarchy in which each level of abstraction is created with knowledge that was gained from the preceding layer of the hierarchy.

High accuracy: Deep Learning algorithms can achieve state-of-the-art performance in various tasks, such as image recognition and natural language processing. Automated feature engineering: Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.

Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labeled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn’t require labeled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

# CHAPTER 2

## LITERATURE SUREVEY

**1. AUTHORS:** Dinh-Son Tran , Hyung Jeong Yang , Ngoc-Huynh Ho Soo-Hyung kim & Guee Sang Lee.

**DESCRIPTION**: A real-time fingertip-gesture-based interface is still challenging for human–computer inter-actions, due to sensor noise, changing light levels, and the complexity of tracking a fingertip across a variety of subjects.

The hand region of interest and the center of the palm are first extracted using in-depth skeleton-joint information images from a Microsoft Kinect Sensor version 2, and then converted into a binary image.

**2. AUTHORS:** Zhenzhou Wang

**DESCRIPTION**: Segmentation of the color image is challenging because the color information is lost after being projected into three channels of the color space. Many state-of-the-art color image segmentation methods are based on mono chrome segmentation in one channel of the color space. However, the optimal performance of a segmentation method usually could not be achieved in a single color space due to the class.

**3. AUTHORS**: Sugnik Roy Chowdhury, Sumit Pathak, M.D. Anto .

**DESCRIPTION**: Image processing, a division of signal processing, can correspond of an image or a videotape as input and output as an image or different parameters of it. Gesture recognition and shadowing is a kind of image processing process. In recent times, a number of gesture recognition ways have been proposed.

Hand tracking has several operations including motion capture, human-computer interaction and human behaviour analysis. Several types of detectors and detection gloves are used for hand motion detection and tracking. Instead of using more precious detectors simple webcams identify the gesture and track the motion.

**4. AUTHORS:** Manjunath R kounte , E Niveditha , A Sai Sudeshna, Kalaignar Afrose.

**DESCRIPTION**: Hand gesture recognition is very significant for human-computer interaction. Gesture recognition is the procedure of figuring out the gestures through the computer which is made by the user. Till now, many hand gesture recognition technologies have been evolved, but Video-Based Hand Gesture Detection with high efficiency and low computational cost is a very challenging task.

Recently there has been extensive research in the field of hand gesture recognition and video recognition models and has made great progress and achieved high efficient recognition rates in almost all the domains.

Hence to implement a good video based action recognition system that has both high accuracy as that of 3D CNN and having low computational cost as that of 2D CNN, we used a Temporal shift Model (TSM) presented in [8] which is inserted into the 2D CNN back bone. Uni-directional TSM algorithm enables to do real time video recognition with low latency and high temporal modeling on high computation devices.

**5. AUTHORS:** MS Zaharin,N Ibrahim, TMAT

**DESCRIPTION**: There are several steps in image processing. Image acquisition, pre-processing, feature extraction, and classification are the processes used for the detection of human movement based on high-level feature extraction (HLFE), in which HLFE was used for feature extraction in this paper.

This research was conducted to analyse the difference of background subtraction and frame difference methods based on movement of human. Movement of human detected by using feature extraction were centroid image technique used. Furthermore, support vector machine (SVM) was used for classification.

**6. AUTHORS:** Poornima Naik , Kavitha Oza.

**DESCRIPTION**: Here concepts are presented in the form of programs making it quite easy and simple for students to understand. It showcases actual screen shots of the programs from the programming environment to make it more student friendly. Because of the user friendly interface provided in the book a novice learner can also learn Python programming without any difficulty.

**7. AUTHORS:** AP Nemirko , JH Dula.

**DESCRIPTION**: In this paper machine learning methods for automatic classification problems using computational geometry are considered. Classes are defined with convex hulls of points sets in a multidimensional feature space.
Classification algorithms based on the estimation of the proximity of the test point to Class shells are convex considered**.**

**8. AUTHORS:** Alberto Antonietti Alice Geminiani Edoardo Negri Egidio Ugo D'Angelo Claudia Casellato.

**DESCRIPTION:** It is common for animals to use self-generated movements to actively sense the surrounding environment. For instance, rodents rhythmically move their whiskers to explore the space close to their body. The mouse whisker system has become a standard model to study active sensing and sensorimotor integration through feedback loops. In this work, we developed a bioinspired spiking neural network model of the sensorimotor peripheral whisker system, modelling trigeminal ganglion, trigeminal nuclei, facial nuclei, and central pattern generator neuronal populations. This network was embedded in a virtual mouse robot, exploiting the Neurorobotics Platform, a simulation platform offering a virtual environment to develop and test robots driven by brain-inspired controllers. Eventually, the peripheral whisker system was properly connected to an adaptive cerebellar network controller. The whole system was able to drive active whisking with learning capability, matching neural correlates of behaviour experimentally recorded in mice.

**9. AUTHORS:** Mr.Venkateshwar A

**DESCRIPTION:** The technique of interaction between human and computer is evolving since the invention of computer technology. The mouse is one of the invention in HCI (human computer interaction) technology. Though wireless are Bluetooth mouse technology is invented still, that technology is not completely device free. A Bluetooth mouse has the requirement of battery power it requires extra power supply. Presence of extra devices in a mouse increases the difficulty level of more hardware components. The proposed mouse system is outside this limitation. This paper proposes a virtual mouse system using colored hand glove based on HCI using computer vision and hand gestures. Gestures captured with a webcam on processed with color segmentation, detection technique and feature extraction

**10. AUTHORS:** Maniya Chandresh , Patel Pratik , Boda Jaruti

**DESCRIPTION:** The important objective of this paper is to use two of the most important modes of interaction – head and hand to control any Computer Vision algorithms based application running on a computer. Video input stream hand is segmented. The corresponding gesture is recognizing based on the shape and pattern of movement of the hand. Hidden Markov model is used for the head gesture. Pre-processing common to hand and head gesture recognition 1st: Capture a frame from camera. 2nd: Hand and face are detected using Viola Jones algorithm. Classifiers are trained on the images of hand and face to detect them using an artificial neural network. Face detected can be used to figure out the area of the head. Method specific to Head gesture recognition: 1st: Optical flows of the all pixels, calculated with the gradient method, in the extracted head region are treated as values representing a head movement. 2nd: The results of head movements are then used for recognition using finite state automata.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

There are some related works carried out on virtual mouse using hand gesture detection by wearing a glove in the hand and also using color tips in the hands for gesture recognition, but they are no more accurate in mouse functions. There cognition is not so accurate because of wearing gloves; also, the gloves are also not suited for some users, and in some cases, the recognition is not so accurate because of the failure of detection of color tips. Some efforts have been made for camera-based detection of the hand gesture interface. In 1990, Quam introduced an early hardware-based system; in this system, the user should wear a Data Glove The proposed system by Quam although gives results of higher accuracy, but it is difficult to perform some of the gesture controls using the system. Dung-Hua Liou, ChenChiung Hsieh, and David Lee in 2010 proposed a study on "A Real-Time Hand Gesture Recognition System Using Motion History Image." The main limitation of this model is more complicated hand gestures. Monika B. Gandhi, Sneha U. Dudhane, and Ashwini M. Patil in 2013 proposed a study on "Cursor Control System Using Hand Gesture Recognition." In this work, the limitation is stored frames are needed to be processed for hand segmentation and skin pixel detection. Vinay Kr. Pasi, Saurabh Singh, and Pooja Kumari in 2016 proposed "Cursor Control using Hand Gestures" in the IJCA Journal. The system proposes the different bands to perform different functions of the mouse. The limitation is it depends on various colors to perform mouse functions.

## 3.2 PHYSICAL MOUSE

It is known that there are various types of physical computer mouse in the modern technology, the following will discuss about the types and differences about the physical mouse.

### 3.2.1 MECHANICAL MOUSE

It is Known as the trackball mouse that is commonly used in the 1990s, the ball within the mouse are supported by two rotating rollers in order to detect the movement made by the ball itself. One roller detects the forward/backward motion while the other detects the left/right motion. The ball within the mouse are steel made that was covered with a layer of hard rubber, so that the detection are more precise. The common functions included are the left/right buttons and a scroll-wheel. However, due to the constant friction made between the mouse ball and the rollers itself, the mouse are prone to degradation, as overtime usage may cause the rollers to degrade, thus causing it to unable to detect the motion properly, rendering it useless. Furthermore, the switches in the mouse buttons are no different as well, as long term usage may cause the mechanics within to be loosed and will no longer detect any mouse clicks till it was disassembled and repaired. Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

**Figure 3.1 Mechanical mouse, with top cover removed**

The following table describes the advantages and disadvantages of the Mechanical Mouse:

| ADVANTAGES | DISADVANTAGES |
|---|---|
| • Allows the users to control the computer system by moving the mouse.<br>• It Provides precise mouse tracking Movements | • Prone to degradation of the mouse rollers and button switches, causing to be faulty.<br>• Requires a flat surface to operate. |

### 3.2.2. OPTICAL AND LASER MOUSE

A mouse that commonly used in these days, the motions of optical mouse rely on the Light Emitting Diodes (LEDs) to detect movements relative to the underlying surface, while the laser mouse is an optical mouse that uses coherent laser lights. Comparing to its predecessor, which is the mechanical mouse, the optical mouse no longer rely on the rollers to determine its movement, instead it uses an imaging array of photodiodes. The purpose of implementing this is to eliminate the limitations of degradation that plagues the current predecessor, giving it more durability while offers

better resolution and precision. However, there's still some downside, even-though the optical mouse are functional on most opaque diffuse surface, it's unable to detect motions on the polished surface. Furthermore, long term usage without a proper cleaning or maintenance may leads to dust particles trap between the LEDs, which will cause both optical and laser mouse having surface detection difficulties. Other than that, it's still prone to degradation of the button switches, which again will cause the mouse to function improperly unless it was disassembled and repaired.

**Figure 3.2 Optical Mouse, with top cover removed**

The following table describes the advantages and disadvantages of the Optical and Laser Mouse:

| ADVANTAGES | DISADVANTGES |
|---|---|
| • Allows better precision with lesser hand movements.<br>• Longer life-span. | • Prone to button switches degradation.<br>• Does not function properly while on a polished surface. |

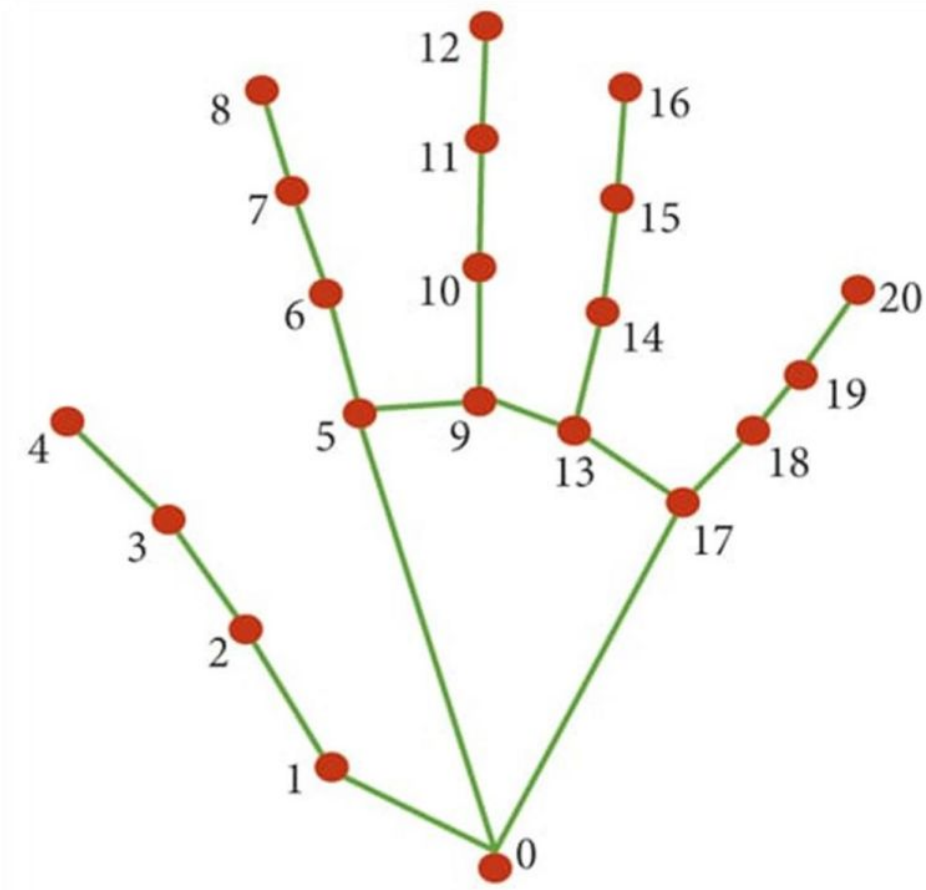
## 3.3 PROPOSED SYSTEM

The various functions and conditions used in the system are explained in the flowchart of the real-time AI virtual mouse system in Figure 4.2. Camera used in the AI Virtual Mouse System. The proposed AI virtual mouse system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library OpenCV, the video capture object is created and the web camera will start capturing video, as shown in Figure 4.2. The web camera captures and passes the frames to the AI virtual system.

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

```
def findHands(self,img,drawTrue):imgRGBcv2.cvtColor(img,cv2.COLOR_BGR2RGB)self.results self.hands.process(imgRGB).
```

Rectangular Region for Moving through the Window. The AI virtual mouse system makes use of the transformational algorithm, and it converts the coordinates of fingertip from the webcam screen to the computer window full screen for controlling the mouse. When the hands are detected and when we find which finger is up for performing the specific mouse function, a rectangular box is drawn with respect to the computer window in the webcam region where we move throughout the window using the mouse cursor.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

**Figure 3.3 Co-ordinates or land marks in the hand**

## 3.4 ADVANTAGES OF PROPOSED SYSTEM

- The Virtual Mouse works as a medium of the user and the machine only using a camera.
- It helps the user to interact with a machine without any mechanical or physical devices and control mouse functions.

## 3.5 ALGORITHM

OpenCV detects the hand and draws an rectangular window around the hand and uses a transformation algorithm that calculates the co-ordinates of the fingertips from the screen capture window to be able in the computer system and controls the pointer of the virtual mouse.

**1.** Initialize the necessary libraries and variables.

**2.** Capture the video stream from the webcam.

**3.** Preprocess the video frames to improve hand detection and segmentation.

**4.** Detect and extract the hand region from the preprocessed frames using a hand detection algorithm (such as a skin color-based approach or deep learning-based methods like CNN or OpenPose).

**5.** Apply gesture recognition techniques to identify different hand gestures. This can be done using machine learning algorithms (such as SVM, Random Forest, or Convolutional Neural Networks) or rule-based approaches.

**6.** Depending on the detected gesture, perform the corresponding mouse action (e.g., moving the cursor, left-click, right-click, scroll).

**7.** If the gesture involves cursor movement, calculate the displacement of the hand position relative to the previous frame.

**8.** Update the cursor position on the screen accordingly.

**9.** If the gesture involves clicking or scrolling, simulate the corresponding mouse event using system functions or libraries (e.g., pyautogui for Python).

**10**. Repeat steps 3 to 9 for each frame in the video stream.

**11**. End the program when the user decides to quit.

## 3.6 SYSTEM REQUIREMENTS

### 3.6.1 HARDWARE REQUIREMENTS

- System : Pentium IV 2.4 GHz.
- Hard Disk : 320 GB.
- Monitor : 15 VGA Color.

### 3.6.2 SOFTWARE REQUIREMENTS

- Operating system : Windows 10.
- Coding Language : Python
- Tool : Anaconda

# CHAPTER 4

# MODULES DESCRIPTION OF PROPOSED METHODOLOGY

## 4.1. METHODOLOGY

The AI virtual mouse framework utilizes the instructive algorithmic rule, and it changes over the co-ordinates of tip from the camera screen to the pc window full screen for the mouse.

## 4.2 FLOWCHART

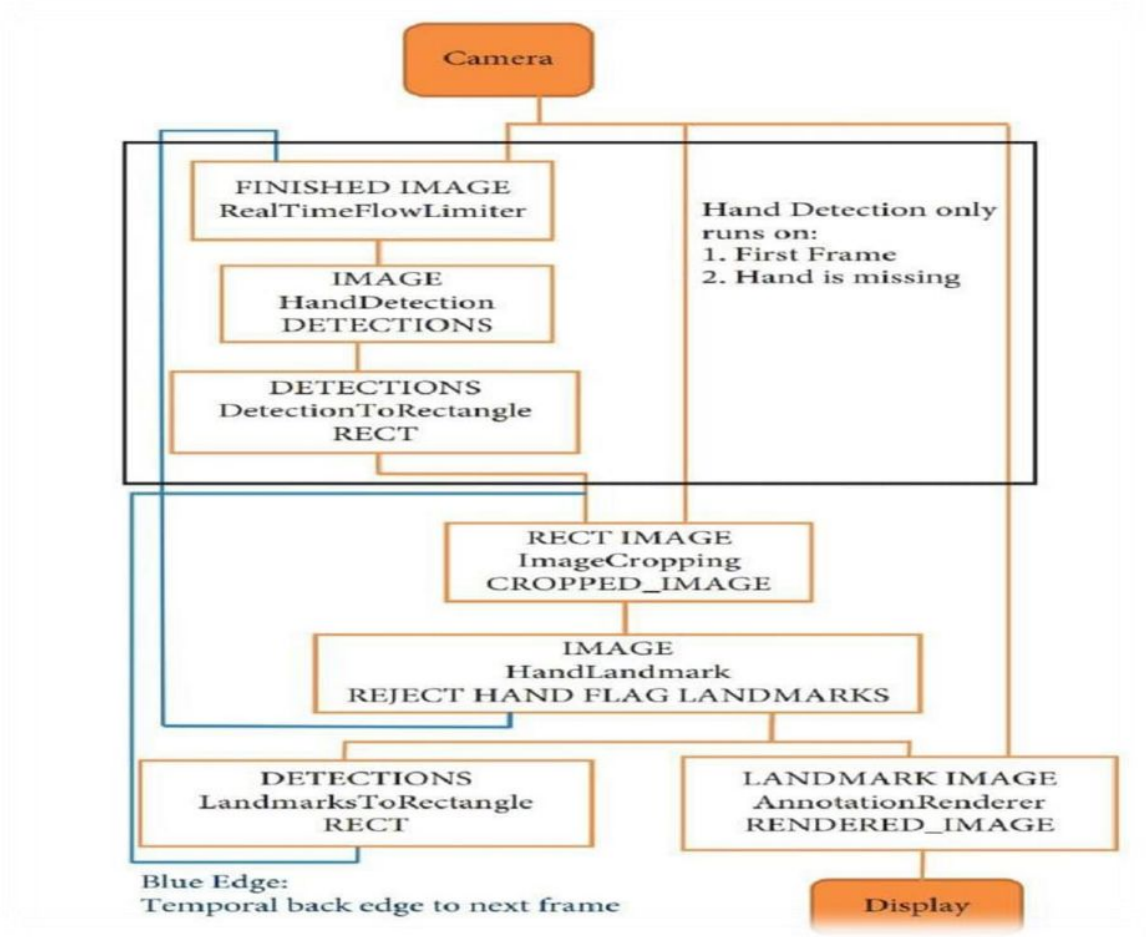



**Figure 4.1 Media Pipe hand recognition graph**

## 4.3 CAMERA USED IN THE VIRTUAL GESTURE MOUSE PROJECT

Open-CV is python vision library that contains Associate in the organized AI virtual mouse system depends upon the edges that are gotten by the camera in Associate in nursing passing PC. Pictures can be conveyed in concealing layered with 3 channels, Grayscale with pixel values fluctuating from 0 (dull) to 255 (white), and twofold portraying dim or white characteristics (0 or 1) specifically.

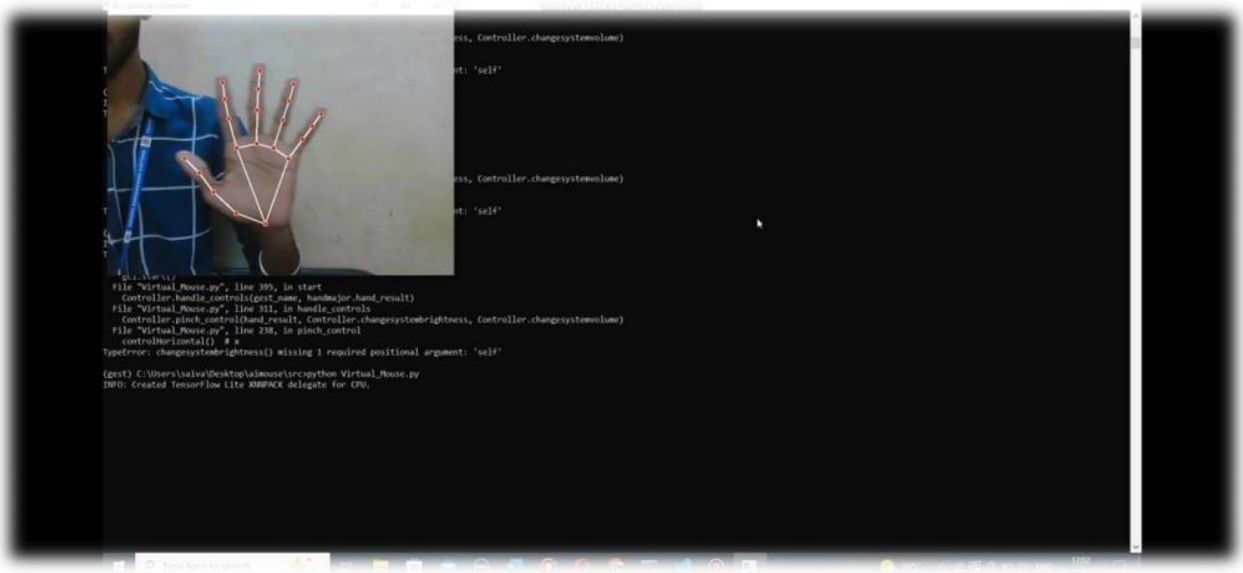



**Figure 4.2 No Movement Cursor**

## 4.4 FOR THE MOUSE TO PERFORM MOVING CURSOR

Hand Gesture Tracking: The system tracks and captures the movements of the user's hand, including the position and orientation of the palm and fingers.

Cursor Movement Gesture Recognition: The system employs gesture recognition algorithms to identify specific hand gestures associated with cursor movement. These gestures can include movements such as pointing, swiping, or waving.

Mapping Gesture to Cursor Movement: Once the cursor movement gesture is recognized, it is mapped to the corresponding cursor movement action. For example, if the user moves their hand to the right, it can be mapped to a cursor movement towards the right side of the screen.

Cursor Position Calculation: The system calculates the new position of the cursor based on the recognized gesture and the current position of the cursor. The speed and precision of the cursor movement can be determined by factors such as the /intensity or distance of the hand gesture.

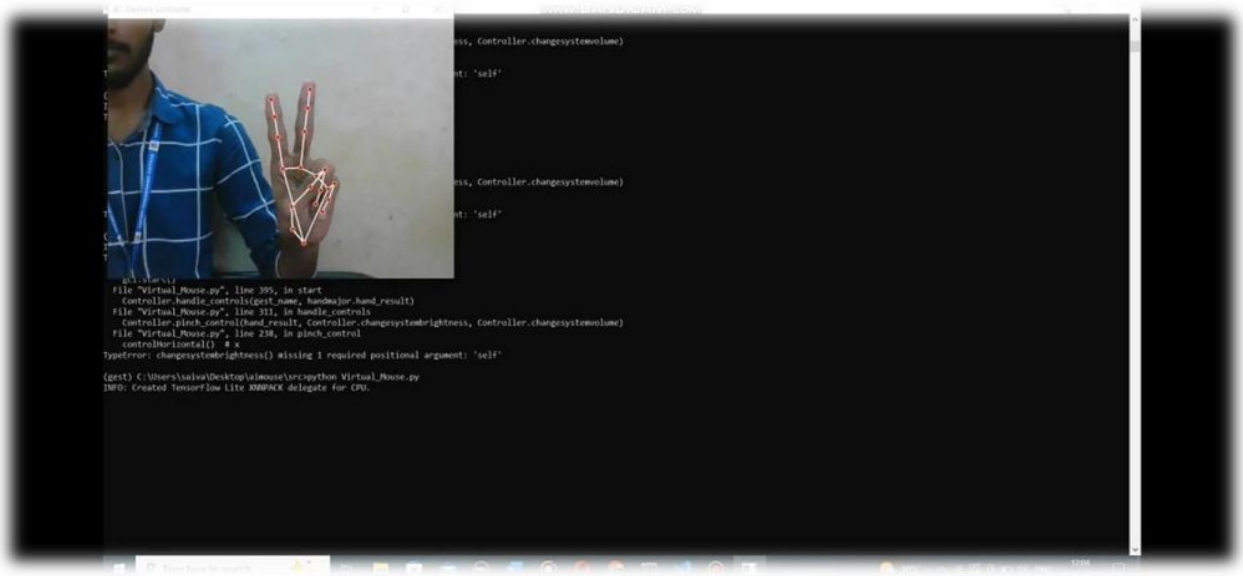



**Figure 4.3 Moving Cursor**

## 4.5 FOR THE MOUSE TO PERFORM LEFT CLICK

Left-Click Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with a left-click action. This gesture can be defined as a distinct hand movement or configuration that represents a left-click, such as tapping a finger or making a specific hand shape.

Mapping Gesture to Left-Click Action: Once the left-click gesture is recognized, it is mapped to the corresponding left-click action. This action instructs the virtual mouse system to simulate a left-click event, similar to pressing the left button on a physical mouse.

Emulating Left-Click Event: When the system detects the left-click gesture, it triggers a left-click event in the virtual mouse system. This event typically involves the activation of a mouse click or tap action at the current cursor position.


**Figure 4.4 Left Click**

## 4.6 FOR THE MOUSE TO PERFORM RIGHT CLICK

Right-Click Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with a right-click action. This gesture can be defined as a distinct hand movement or configuration that represents a right-click, such as tapping a specific finger or making a unique hand shape.

Mapping Gesture to Right-Click Action: Once the right-click gesture is recognized, it is mapped to the corresponding right-click action. This action instructs the virtual mouse system to simulate a right-click event, similar to pressing the right button on a physical mouse.

Emulating Right-Click Event: When the system detects the right-click gesture, it triggers a right-click event in the virtual mouse system. This event typically involves the activation of a context menu or a specific action associated with a right-click input.

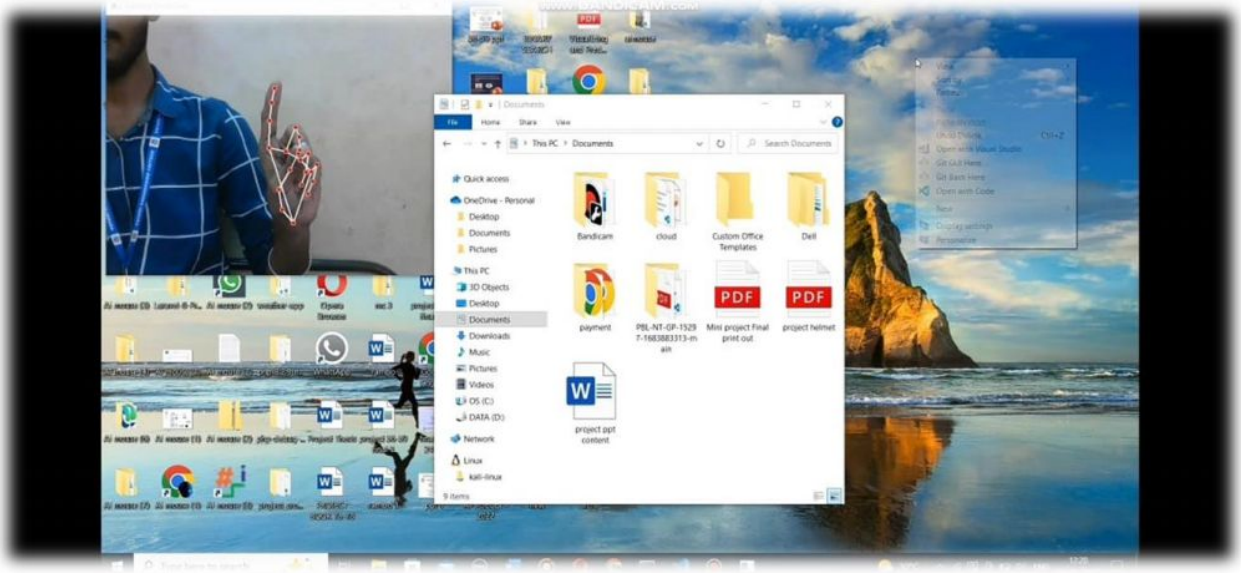



**Figure 4.5 Right Click**

## 4.7 FOR THE MOUSE TO PERFORM DOUBLE CLICK

Double-Click Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with a double-click action. This gesture can be defined as a repeated tapping or a distinct hand movement that represents a double-click action.

Mapping Gesture to Double-Click Action: Once the double-click gesture is recognized, it is mapped to the corresponding double-click action. This action instructs the virtual mouse system to simulate a double-click event, similar to quickly pressing the left button on a physical mouse twice.

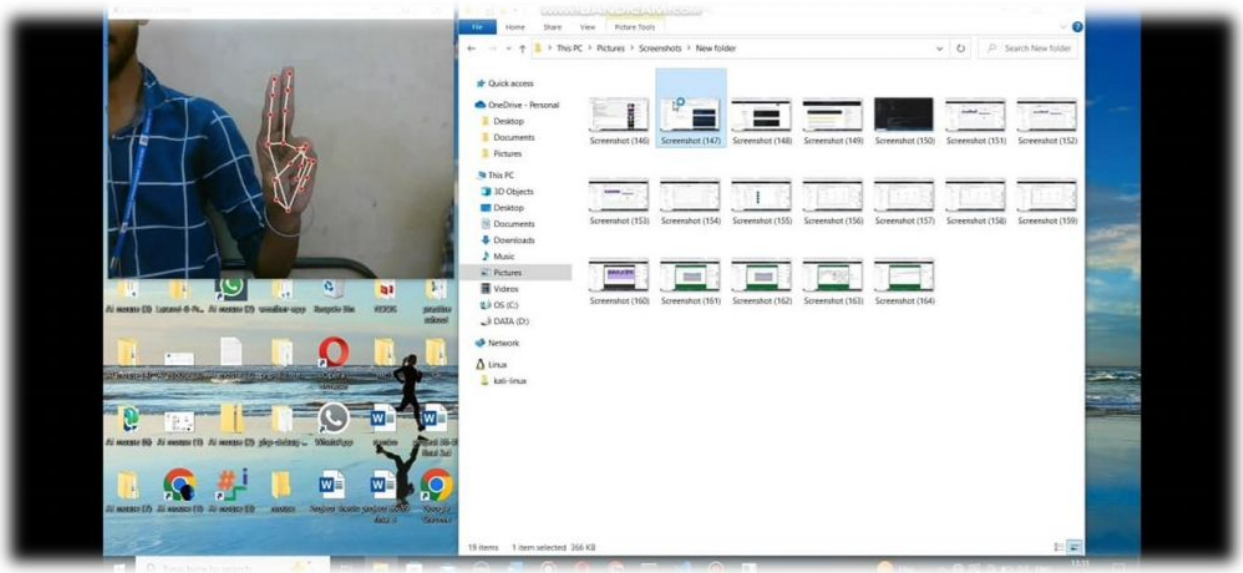


**Figure 4.6 Double Click**

## 4.8 FOR THE MOUSE TO PERFORM DRAG AND DROP

Drag Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with a drag action. This gesture can be defined as a continuous movement of the hand while maintaining contact with a surface or holding a virtual object.

Mapping Gesture to Drag Action: Once the drag gesture is recognized, it is mapped to the corresponding drag action. This action instructs the virtual mouse system to simulate a dragging event, similar to dragging an object with a physical mouse.

Cursor Position Calculation: The system calculates the position of the cursor during the drag gesture, tracking the movement of the hand and translating it into cursor movement.

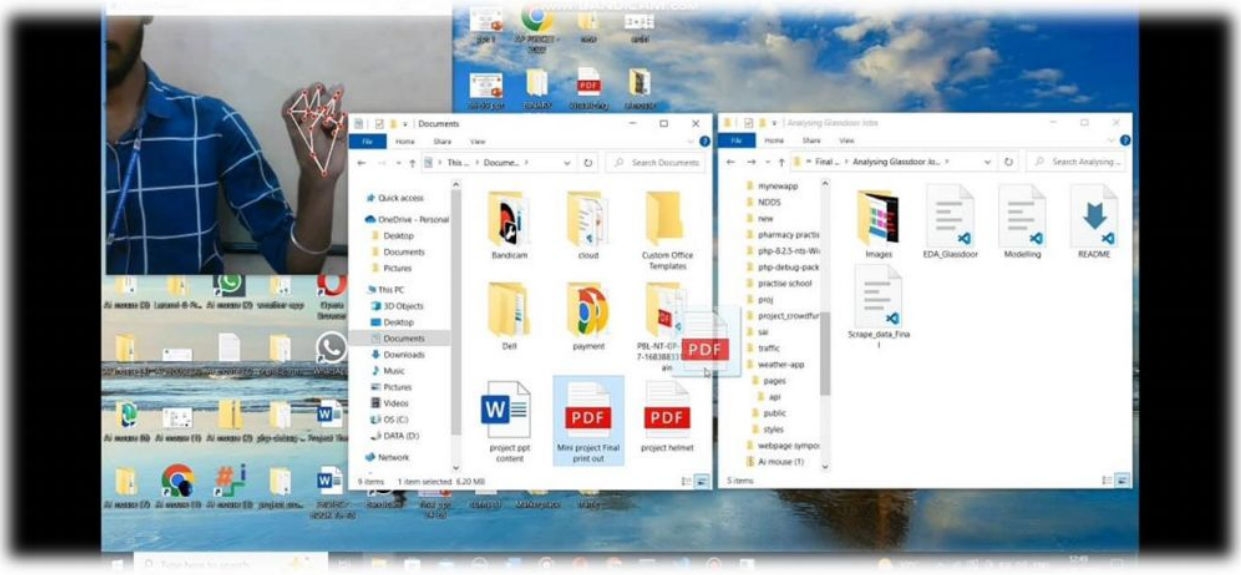


**Figure 4.7 Drag and Drop**

## 4.9 FOR THE MOUSE TO PERFORM MULTIPLE SELECT

Multi-Select Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with multi-select. This gesture can be defined as a gesture that indicates the intention to select multiple objects or elements.

Mapping Gesture to Multi-Select Action: Once the multi-select gesture is recognized, it is mapped to the corresponding multi-select action. This action instructs the virtual mouse system to initiate a multi-select mode, enabling the selection of multiple objects.

Object Selection: The user performs the multi-select gesture while maintaining contact with the surface or holding a virtual object. The system tracks the movement of the hand and identifies the objects or elements that fall within the selection area or gesture.

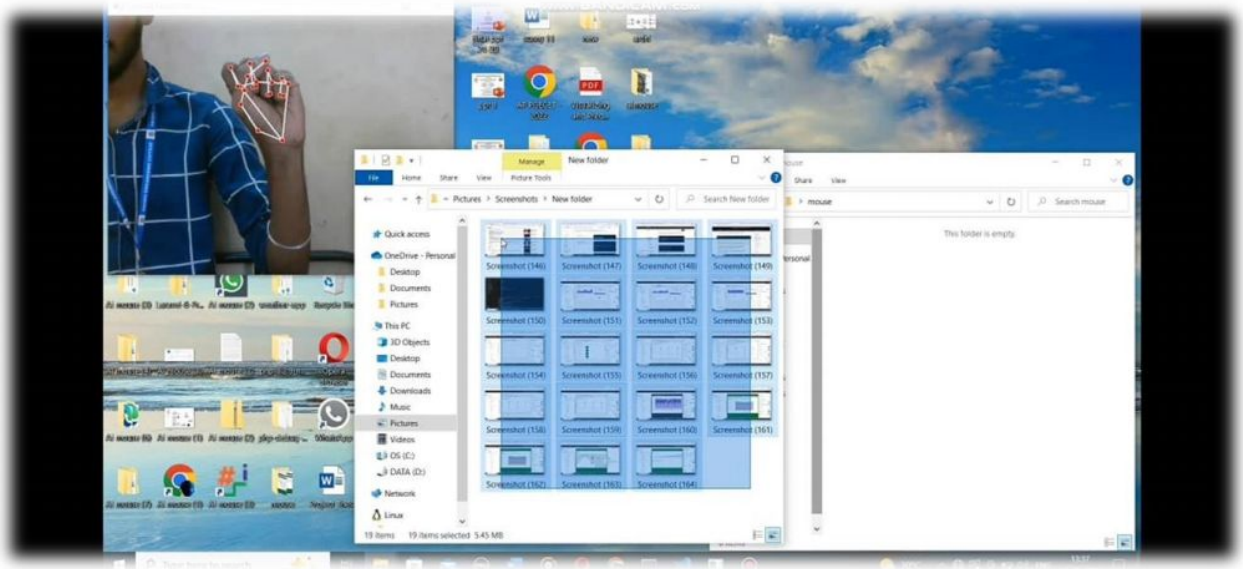


**Figure 4.8 Multiple Select**

## 4.10 FOR THE MOUSE TO PERFORM STOP PROGRAM

Stop Gesture Recognition: The system employs gesture recognition algorithms to identify a specific hand gesture associated with stopping or closing a program. This gesture can be a predefined hand movement or configuration that represents the stop action.

Mapping Gesture to Program Stop Action: Once the stop gesture is recognized, it is mapped to the corresponding program stop action. This action instructs the virtual mouse system to initiate the program termination process.

Program Termination: Upon detecting the stop gesture, the virtual mouse system triggers the appropriate commands to stop the program. This can involve sending a close request or terminating the program's process.

**Figure 4.9 Stop Program**

| Hand tip gesture * | Mouse function performed | Success | Failure | Accuracy (%) |
|---|---|---|---|---|
| Tip ID 1 or both tip IDs 1 and 2 are up | Mouse movement | 100 | 0 | 100 |
| Tip IDs 0 and 1 are up and the distance between the fingers is <30 | **Left button click** | 99 | 1 | 99 |
| Tip IDs 1 and 2 are up and the distance between the fingers is <40 | **Right button click** | 95 | 5 | 95 |
| Tip IDs 1 and 2 are up and the distance between the fingers is >40 and both fingers are moved up the page | Scroll up function | 100 | 0 | 100 |
| Tip IDs 1 and 2 are up and the distancebetween the fingers is >40 and both fingers are moved down the page | Scroll down function | 100 | 0 | 100 |
| All five tip IDs 0, 1, 2, 3, and 4 are up | No action performed | 100 | 0 | 100 |
| Result | | 594 | 6 | 99 |

*Finger tip ID for respective fingers: tip Id 0: thumb finger; tip Id 1: index finger; tip Id 2: middle finger; tip Id 3: ring finger; tip Id 4: little finger.

**Figure 4.10 Experimental Result**

# CHAPTER 5

# IMPLEMENTATION

## 5.1 PLANNING

A thorough planning will be conducted in this phase where the existing systems/product, for this case, physical computer mouse will be reviewed and studied to identify the problems existed, a comparison of problems will be made to compare which problems are more crucial and requires improvement. An outline objective and the scope will be identified in order to provide an alternative solution to the problem.

## 5.2 REQUIREMENT ANALYSIS

The phase that gathers and interpreting the facts, diagnosing problems and recommending improvements to the system. In this phase, the collected problem statements will be extensively studied in order to find a proper solution or at least an improvements to the proposed system. All proposed solutions will be converted into requirements where it will be documented in a requirement specification.

## 5.3 DESIGNING

The requirement specification from the previous phase will be studied and prioritize to determine which requirement are more important where the requirement with the highest priority will be delivered first. After the study, the system design will be prepared as it helps in defining the overall system architecture and specifying the hardware and the software requirements.

## 5.4 BUILDING

The phase where the actual coding implementation takes place. By referring to the inputs from the system design, the system will be developed based on the prioritize requirements. However, due to we're using the agile methodology approach, the developed system will be considered as a prototype system where it will be integrated and tested by the users.

## 5.5 TESTING

The phase where the prototype system going through a series of test. The prototype system will first undergo integration where the features from the previous iteration cycle are added to the latest cycle. After the integration, the prototype system will be thoroughly tested by the users to determine whether they are satisfied with the latest deliverables, the completion of the project depends on whether they've accepted it or otherwise. If the users requires additional features or modification, feedback gathering will be conducted, which resulted in further modification of the requirements and features where it will recorded and documented for the requirement analysis phase on the next iteration.

## 5.6 IMPLEMENTATION ISSUES AND CHALLENGES

Throughout the development of the application, there are several implementation issues occurred. The following describes the issues and challenges that will likely to be encountered throughout the development phase.

The interruptions of salt and pepper noises within the captured frames. Salt and pepper noises occurred when the captured frame contains required HSV values that are too small, but still underwent a series of process even though it's not large enough to be considered an input. To overcome this issue, the unwanted HSV pixels within the frame must first be filtered off, this includes the area of the pixels that are too large and small. With this method, the likelihood of interruptions of similar pixels will reduce greatly.

Performance degradation due to high process load for low-tier system. Since the application is required to undergo several of process to filter, process and execute the mouse functions in real time, the application can be CPU intensive for most of the low-tier system. If the size of the captured frames is too large, the time-taken for the application to process the entire frame are increase drastically. Therefore, to overcome this issue, the application is required to process only the essential part of the frames, and reduces the redundant filtering process that could potentially slow the application down.

The difficulties of calibrating the brightness and the contrast of the frames to get the required HSV values. The intensity of brightness and contrast matters greatly when it comes to acquiring the required color pixels. In order for the application to execute the entire mouse functions provided, all of the required HSV values to execute the specific mouse functions must be satisfied, meaning that the overall HSV values must be satisfied with the brightness and contrast as well. However, the calibration can be somewhat tedious as certain intensity could only satisfy part of the required HSV values, unless the original HSV values were modified to prove otherwise. To overcome this issue, the application must first start up with a calibration phase, which allows the users to choose the color pixels before directing them to the main phase.

## 5.7 ARCHITECTURE

There are following layers in AI virtual mouse

- Interface layer
- Process layer
- Data manipulation layer
- Data layer

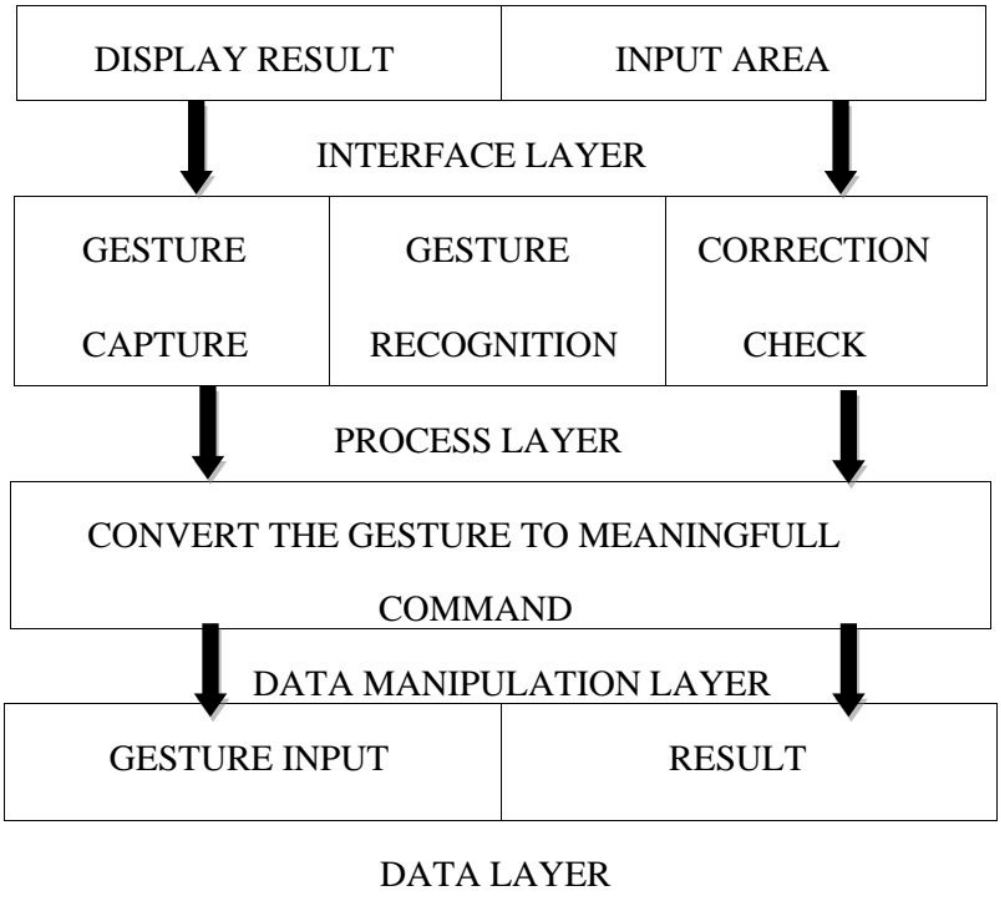



**Figure 5.1 Architecture Diagram**

### 5.7.1 INTERFACE LAYER

The interface layer of a virtual mouse using hand gestures encompasses the crucial components that enable users to control the mouse cursor and execute actions through hand movements. It begins with hand detection and tracking, employing computer vision techniques to locate and monitor the user's hand in the camera feed. Once the hand is identified, gesture recognition algorithms come into play, analyzing the hand movements and classifying them as specific gestures. The recognized gestures are then interpreted, mapping them to corresponding mouse actions like cursor movement, clicking, or scrolling. To ensure accurate control, the system tracks the hand state, detecting actions such as hand open/close or variations in finger positions. The hand movements are mapped to screen coordinates, enabling translation to appropriate cursor movements on the display. Finally, the system emulates mouse behavior programmatically, simulating the desired actions by interacting with the operating system or underlying mouse driver. Through visual cues, feedback, and perhaps audio or haptic responses, users receive confirmation and guidance, enhancing their interaction with the virtual mouse interface.

### 5.7.2 PROCESS LAYER

The process layer of a virtual mouse using hand gestures is responsible for orchestrating the various steps and algorithms involved in capturing, interpreting, and translating hand movements into mouse actions. It acts as a bridge between the input from the hand detection and tracking stage and the output to the interface layer. The process layer involves a series of interconnected operations, including hand detection, gesture recognition, gesture interpretation, coordinate mapping, and mouse emulation. It coordinates the flow of information between these operations, ensuring seamless communication and synchronization. It manages the hand tracking

algorithm's output and passes it to the gesture recognition module, which identifies and classifies the performed hand gestures. The recognized gestures are then interpreted, mapped to specific mouse actions, and transformed into appropriate screen coordinates based on calibration and mapping techniques. Finally, the process layer interacts with the interface layer, translating the interpreted gestures into programmatically emulated mouse actions. It oversees the overall flow of data and commands, ensuring efficient and accurate translation of hand gestures into virtual mouse control.

### 5.7.3 DATA MANIPULATION LAYER

The data manipulation layer of a virtual mouse using hand gestures plays a crucial role in processing and transforming the captured data to facilitate accurate interpretation and control. This layer encompasses various operations and algorithms aimed at manipulating and refining the input data for seamless interaction. It begins by receiving the raw data from the hand detection and tracking module, which includes information about the hand's position, shape, and motion. The layer then performs pre-processing tasks such as noise reduction, filtering, and normalization to enhance the quality and reliability of the data. This ensures that the subsequent steps operate on clean and consistent input. Next, feature extraction techniques may be applied to extract relevant features from the hand data, such as finger positions, hand orientation, or hand velocity. These features serve as input to the gesture recognition module, which employs machine learning or pattern recognition algorithms to classify and identify the performed hand gestures accurately. The data manipulation layer also handles coordinate mapping, converting the hand coordinates to screen coordinates by considering factors like screen resolution, calibration, and mapping parameters. This ensures that the virtual mouse accurately reflects the hand

movements on the screen. Overall, the data manipulation layer processes, refines, and transforms the raw hand data to provide the necessary inputs for gesture recognition, coordinate mapping, and subsequent stages of virtual mouse control, enabling intuitive and responsive interaction.

### 5.7.4 DATA LAYER

The data layer of a virtual mouse using hand gestures is responsible for capturing, storing, and managing the data related to hand movements and gestures. It serves as the foundation for the entire system, ensuring the availability and accessibility of the necessary information. The data layer includes components that capture the input from the camera or sensor, retrieving frames or data points that represent the user's hand position and motion. These raw data points are then processed and stored in a structured format, allowing for efficient retrieval and analysis. The data layer also incorporates techniques for data preprocessing, such as noise reduction, calibration, and normalization, which enhance the quality and consistency of the captured data. Additionally, the data layer may store reference data or models used in gesture recognition, such as labeled gesture examples for training machine learning models. This enables the system to continually improve its gesture recognition accuracy through iterative learning and model updates. Moreover, the data layer may include mechanisms for real-time data streaming or buffering to support smooth and uninterrupted hand tracking and gesture recognition. By providing a robust and well-organized data infrastructure, the data layer ensures that the system can effectively capture, process, and utilize the hand movement and gesture data for accurate control of the virtual mouse.

## 5.8 VIRTUAL MOUSE WORKING

During the process of color recognition, it contains 2 major phases which are the calibration phase and recognition phase. The purpose of the calibration phase is to allow the system to recognize the Hue Saturation Values of the colors chosen by the users, where it will store the values and settings into text documents, which will be used later on during the recognition phase. While on the recognition phase, the system will start to capture frames and search for color input with based on the values that are recorded during the calibration phase. The phases of the virtual mouse is as shown in figure below.

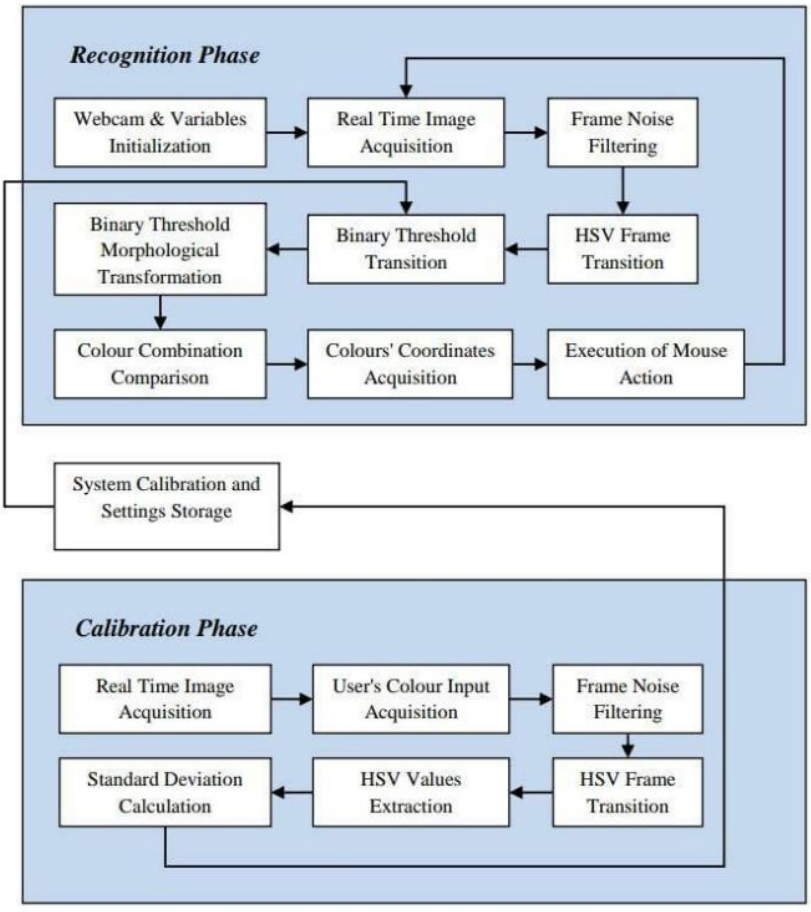



**Figure 5.2 Virtual Mouse Block Diagram**

### 5.8.1 CALIBRATION PHASE

#### a) Real Time Image Acquisition

The program will start of by capturing real-time images via a webcam where it will await for users' color input. The size of the acquired image will be compressed to a reasonable size to reduce the processing loads of processing the pixels within the captured frame.

#### b) User's Color Input Acquisition

The program acquires the frames that consist of input colors submitted by the users, the captured frame will be sent for process where it will undergone a series of transition and calculation to acquire the calibrated HSV values.

#### c) Frame Noise Filtering

Every captured frame consists of noises that will affect the performance and the accuracy of the program, therefore the frame require to be noise free. To do that, filters need to be applied on the captured frames to cancel out the unwanted noise. For the current project, Gaussian filter will be used, which is a common smoothing method to eliminate noise in a frame. This can be done by using GaussianBlur(Input Array src, Output Array dst, Size k size, double sigma X, double sigma Y=0, int border Type =BORDER_DEFAULT ).

#### d) HSV Frame Transition

The captured frame require to be converted from a BGR format to a HSV format. Which can be done by using cvt Color(src, dst, CV_BGR2HSV). Before After the comparison between RGB and HSV frame

### e) HSV Values Extraction

In order to acquire the HSV values, the converted frame require to be split into 3 single different planes, to do that the frame needs to be divided from a multi-channel array into a single channel array, which can be done by using split(const Mat& src, Mat* mvbegin).

### f) Standard Deviation Calculation

To obtain the maximum and the minimum of the HSV values, it requires to gone through the Standard Deviation calculation, a measurement used to quantify the amount of variation / dispersion among other HSV values. Furthermore, to obtain an accurate range of values, three-sigma rule are required in the calculation, so that chances of the captured values have a very high possibility to fall within the three-sigma intervals.

## 5.8.2 RECOGNITION PHASE

### a) Webcam & Variables Initialization

On the early stage of the recognition phase, the program will initialize the required variables which will be used to hold different types of frames and values where each are will be used to carry out certain task. Furthermore, this is the part where the program collects the calibrated HSV values and settings where it will be used later during the transitions of Binary Threshold.

### b) Real Time Image Acquisition

The real time image is captured by using the webcam by using (cv::VideoCapture cap(0);), where every image captured are stored into a frame variable (cv::Mat), which will be flipped and compressed to a reasonable size to reduce process load.

**c) HSV Frame Transition**

The captured frame require to be converted from a BGR format to a HSV format. Which can be done by using cvtColor(src, dst, CV_BGR2HSV).

**d) Binary Threshold Transition**

The converted HSV frame will undergone a range check to check if the HSV values of the converted frame lies between the values of the HSV variables gathered during the calibration phase. The result of the range check will convert the frame into a Binary Threshold, where a part of the frame will set to 255 (1 bit) if the said frame lies within the specified HSV values, the frame will set to 0 ( 0 bit) if otherwise.

**e) Binary Threshold Morphological Transformation**

After the binary threshold is obtained, the frame will undergone a process called Morphological Transformation, which is a structuring operation to eliminate any holes and small object lurking around the foreground. The transformation consist of two morphological operators, known as Erosion and Dilation. The Erosion operator are responsible for eroding the boundaries of the foreground object, decreasing the region of the binary threshold, which is useful for removing small noises.

**f) Color Combination Comparison**

After obtaining results from Morphological Transformation process, the program will calculate the remaining number of objects by highlighting it as blobs, this process requires c v Blob library, which is an add on to Open CV. The results of the calculation will then send for comparison to determine the mouse functions based on the color combinations found within the captured frames.

### g) Colors Coordinates Acquisition

For every object within the binary threshold, the program will highlight the overall shape of the object (cvRenderBlobs(const IplImage *imgLabel, CvBlobs &blobs, IplImage *imgSource, IplImage *imgDest, unsignedshort mode=0x000f, double alpha=1.);), where it will calculate the area of the shape and the coordinates of midpoint of the shapes. The coordinates will be saved and used later in either setting cursor positions, or to calculate the distance between two points to execute various mouse functions based on the result collected.

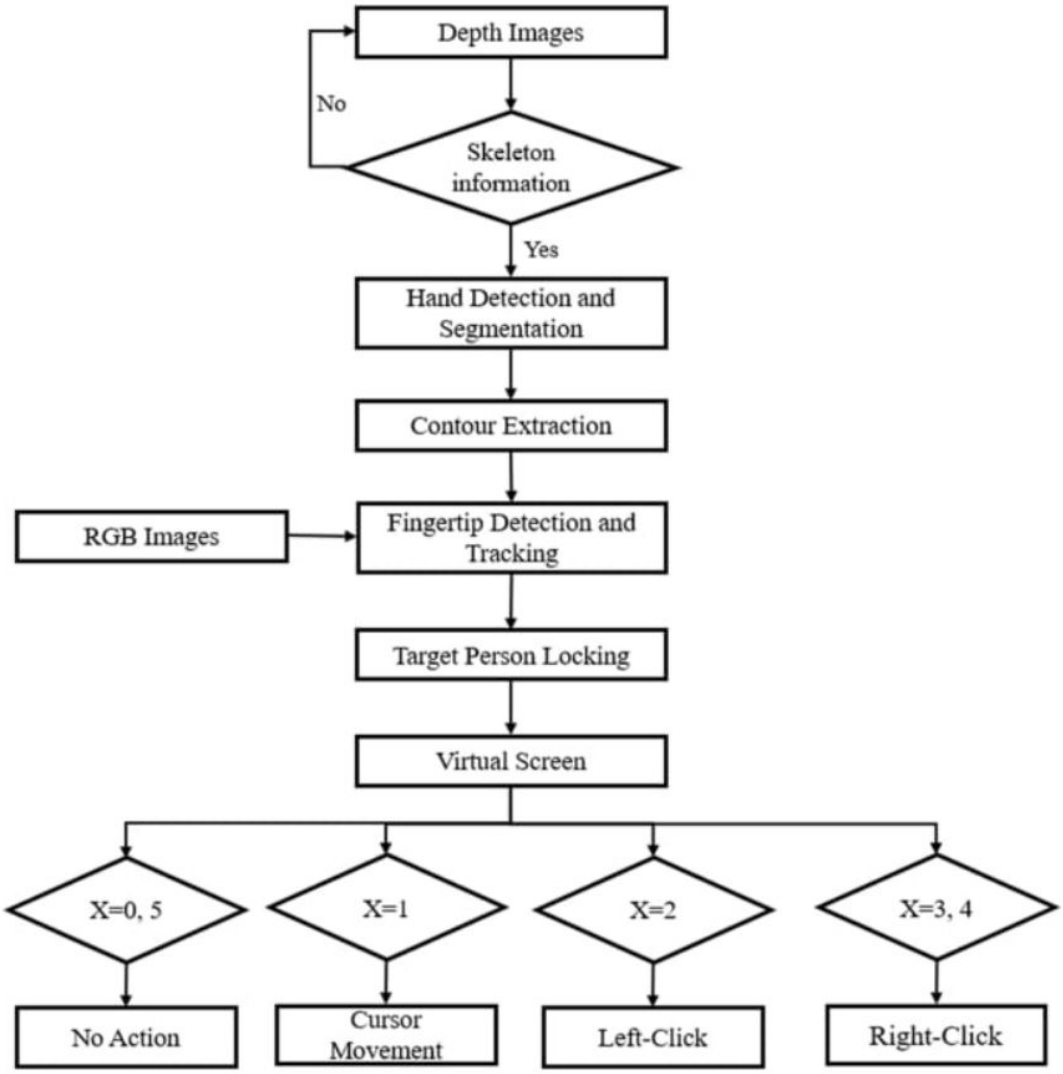



**Figure 5.3 Algorithm Flow Chart**

## 5.9 PSEUDO CODE:

1) Pseudo code algorithm for edge-detection

a) Start

b) Import pyhon OpenCV3- input section

define the normal value for ‘A’‘A’ value to be divided by 500

define the edge algorithm parameters - pictureand intensity

define height X and width Y of an picturedefine the edge

c) Recognizing section

for all height X and width Y pixels in rangeextract pixel values

top and bottom , left and right

top_left and top_rightbottom_left and bottom_rightextract differences

difference I = top minus bottomdifference II = left minusright

extract total diff

total diff. = diff. I + diff. II

total diff. = normal (total diff.) * intensityextract pixels of the image

picture_pix = image [X , Y]extract edge_image

edge_picture [ X , Y] = picture_pix * total diff

d) Output

Display input picture

Display input picture converted togray scaleDisplay edge

e) End

## 5.10 FLOW CHART

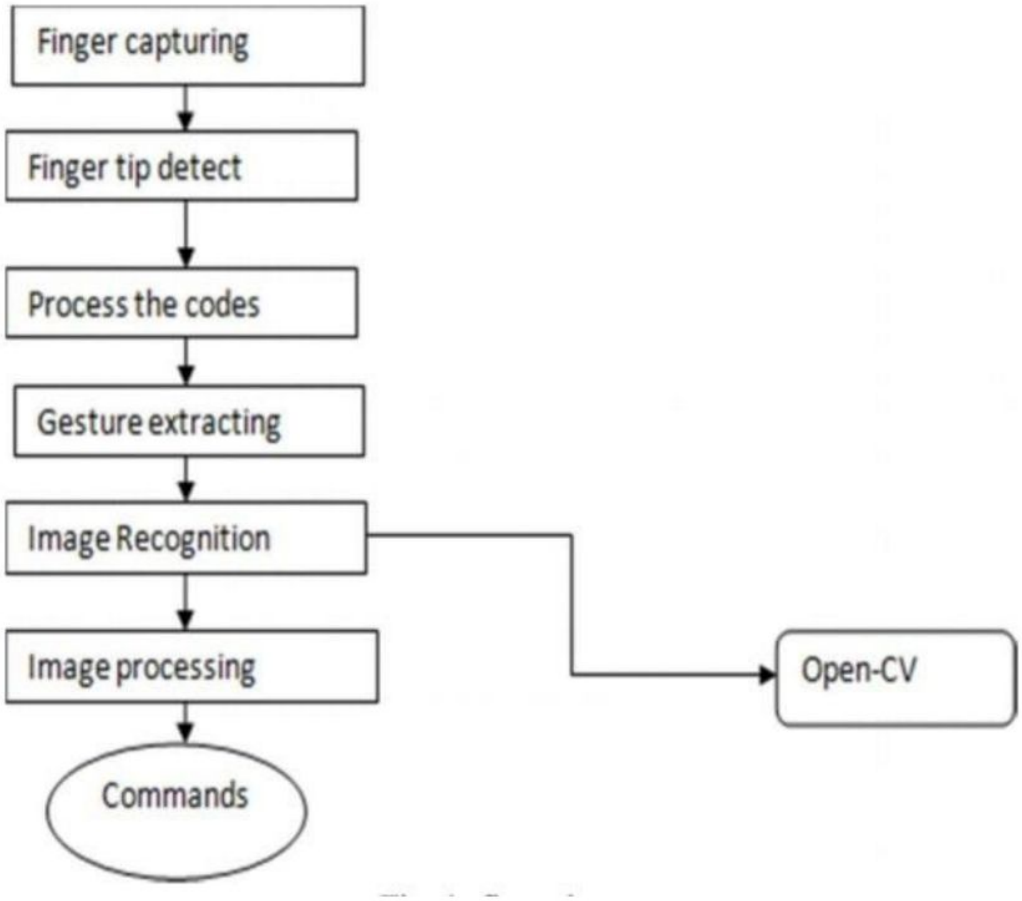



**Figure 5.4 Flowchart of Working Virtual Mouse**

# CHAPTER 6

# SYSTEM DEVELOPEMENT

## 6.1 ANACONDA DESCRIPTION

The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

## 6.2 HISTORY OF ANACONDA

Anaconda was founded in 2012 by Peter Wang and Travis Oliphant out of the need to bring Python into business data analytics, which was rapidly transforming as a result of emerging technology trends. Additionally, the open-source community lacked an entity that could organize and collectivize it to maximize its impact. Since that time, the Python ecosystem has significantly expanded, with Python being the most popular programming language used today. Alongside this expansion, Anaconda has provided value to students learning Python and data science, individual

practitioners, small teams, and enterprise businesses. We aim to meet every user where they are in their data science journey. Anaconda now has over 300 full-time employees based in the United States, Canada, Germany, United Kingdom, Australia, India, and Japan. We are proud to serve over 35 million users worldwide.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other package.

## 6.3 VARIABLES

Anaconda Project sets some environment variables automatically: PROJECT_DIR specifies the location of your project directory. CONDA_ENV_PATH is set to the file system location of the current conda environment. PATH includes the binary directory from the current conda environment.

## 6.4 TOOLS

Tools is a package development process library designed to facilitate packaging Python projects by enhancing the Python standard library distutils (distribution utilities). It includes:

- Python package and module definitions
- Distribution package metadata
- Test hooks and Project installation
- Platform-specific details
- Python 3 support

## 6.5 NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

## 6.6 FUNCTIONS

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments.

## 6.7 LANGUAGES SUPPORT ANACONDA

It is cross-platform (Windows, macOS, Linux [x86/AARCH64/PPC64LE/ s390x]), cross-language (supports Python, R, C/C++, Rust, Go, and more), and ensures package compatibility and environment correctness.

## 6.8 CLASS AND OBJECTS

A class is a code template for creating objects. Objects have member variables and have behaviour associated with them. In python a class is created by the keyword class . An object is created using the constructor of the class. This object will then be called the instance of the class.

## 6.9 KEY FEATURES

- It has more than 1500 Python/R data science packages.
- Anaconda simplifies package management and deployment.
- It has tools to easily collect data from sources using machine learning and AI.
- It creates an environment that is easily manageable for deploying any project.
- Anaconda is the industry standard for developing, testing and training on a single machine.
- It has good community support- you can ask your questions there.
- Download more than 1500 Python/R data science packages.
- Manage libraries, dependencies, and environments with conda.
- Build and train ML and deep learning models with scikit-learn, TensorFlow and Theano.
- Use Dask, NumPy, Pandas and Numba to analyze data scalably and fast.
- Perform visualization with Matplotlib, Bokeh, Datashader, and Holoviews.

## 6.10 THE ANACONDA LANGUAGE

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

## 6.11 ANACONDA WORKING ENVIRONMENT

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks.

## 6.12 GRAPHICAL USER INTERFACE

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® Distribution that allows you to launch applications and manage conda packages, environments, and channels without using command line interface (CLI) commands.

## 6.13 FUNCTIONAL LIBRARY

It includes both the Python and R programming languages,most of the common Python libraries used in science and engineering (including OpenCV, MedisaPipe, NumPy, SciPy, Matplotlib, and pandas), and many commonly used R packages.

### 6.13.1 MEDIAPIPE

Google created the open-source MediaPipe framework to enable the development of cross-platform, real-time computer vision applications. For processing and analyzing video and audio streams, it offers a number of pre-made tools and components, such as object detection, pose estimation, hand tracking, facial recognition, and more.Developers can quickly construct intricate pipelines using MediaPipe that combine numerous algorithms and processes and execute in real-time on a variety of h/w platforms, like CPUs, GPUs, and specialized accelerators like

Google's Edge TPU. Additionally, the framework has interfaces helps us interacting with other well-liked machine learning libraries, including TensorFlow and PyTorch, and it supports several programming languages, like C++, Python, and Java.

For computer vision and ML tasks, MediaPipe is a comprehensive library that offers a many of features. Here are a few of the library's main attributes and features:

1. **Video and Audio Processing:** MediaPipe provides tools for processing and analyzing video and audio streams in real-time. This includes functionalities such as video decoding, filtering, segmentation, and synchronization.

2. **Facial Recognition:** MediaPipe can detect and track facial landmarks, including eyes, nose, mouth, and eyebrows, in real-time. This functionality is useful for applications such as facial recognition, emotion detection, and augmented reality.

3. **Hand Tracking:** MediaPipe can track hand movements in real-time, allowing for hand gesture recognition and interaction with virtual objects.

4. **Object Detection:** MediaPipe can detect and track objects in real-time using machine learning models. This functionality is useful for applications such as augmented reality, robotics, and surveillance.

5. **Pose Estimation:** MediaPipe can estimate the poses of human bodies in real-time, allowing for applications such as fitness tracking, sports analysis, and augmented reality.

For a variety of tasks, such as object detection, position estimation, facial recognition, and more, MediaPipe offers tools for training and deploying machine learning models. All in all, MediaPipe is a potent tool kit that gives programmers the ability to easily create sophisticated real-time computer vision and ML applications.

### 16.13.2 OPENCV

A computer vision and ML software library called OpenCV is available for free download. Its objective is to aid programmers in the development of computer vision applications. Filtering, feature identification, object recognition, tracking, and other processing operations for images and videos are all available through OpenCV.

Python, Java, and MATLAB are just a few of the numerous programming languages that it has bindings for. It is written in C++.Robotics, self-driving cars, AR, medical image analysis, and other fields are just a few of the fields where OpenCV can be employed. A wide range of algorithms and tools are included in the library, making it simple for programmers to build sophisticated computer vision applications.

The steps listed below can be used to broadly classify OpenCV's operation:

1. **Loading and Preprocessing the Image/Video:** OpenCV can load images or videos from a variety of sources such as files, cameras, or network streams. Once the image or video is loaded, it can be preprocessed by applying filters or transforming the image to a different color space, such as converting a color image to grayscale.

2. **Feature Detection and Description:** OpenCV can detect and extract features from an image or video, such as edges, corners, and blobs. These features can

be used to identify objects or track their motion over time. OpenCV also provides algorithms for describing these features, which can be used to match them across multiple frames or images.

3. **Object Detection and Recognition:** OpenCV can be used to detect and recognize objects in an image or video. This can be done using a variety of techniques, such as template matching, Haar cascades, or deep learning-based methods.

4. **Tracking:** OpenCV can track objects in a video stream by estimating their position and motion over time. This can be done using a variety of algorithms, such as optical flow, mean-shift, or Kalman filtering.

5. **Image and Video Output:** Finally, OpenCV can be used to display or save the processed images or videos. This can be done by showing the images in a window, writing the video frames to a file, or streaming the video over a network.

In general, OpenCV offers a large variety of tools and techniques for working with image and video data, making it a potent library for computer vision applications.

# CHAPTER 7

# SOURCE CODE

## 7.1 PYTHON SOURCE CODE

```
import cv2
import mediapipe as mp
import pyautogui
import math
from enum import IntEnum
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
from google.protobuf.json_format import MessageToDict
import screen_brightness_control as sbcontrol

pyautogui.FAILSAFE = False
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

# Gesture Encodings
class Gest(IntEnum):
    # Binary Encoded
    FIST = 0
    PINKY = 1
    RING = 2
    MID = 4
    LAST3 = 7
    INDEX = 8
    FIRST2 = 12
    LAST4 = 15
    THUMB = 16
    PALM = 31
```

```
    # Extra Mappings
    V_GEST = 33
    TWO_FINGER_CLOSED = 34
    PINCH_MAJOR = 35
    PINCH_MINOR = 36

# Multi-handedness Labels
class HLabel(IntEnum):
    MINOR = 0
    MAJOR = 1

# Convert Mediapipe Landmarks to recognizable Gestures
class HandRecog:

    def _init_(self, hand_label):
        self.finger = 0
        self.ori_gesture = Gest.PALM
        self.prev_gesture = Gest.PALM
        self.frame_count = 0
        self.hand_result = None
        self.hand_label = hand_label

    def update_hand_result(self, hand_result):
        self.hand_result = hand_result

    def get_signed_dist(self, point):
        sign = -1
        if self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y:
            sign = 1
        dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
        dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
        dist = math.sqrt(dist)
        return dist * sign

    def get_dist(self, point):
```

```
    dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
    dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
    dist = math.sqrt(dist)
    return dist

def get_dz(self, point):
    return abs(self.hand_result.landmark[point[0]].z - self.hand_result.landmark[point[1]].z)

# Function to find Gesture Encoding using current finger_state.
# Finger_state: 1 if finger is open, else 0
def set_finger_state(self):
    if self.hand_result == None:
        return

    points = [[8, 5, 0], [12, 9, 0], [16, 13, 0], [20, 17, 0]]
    self.finger = 0
    self.finger = self.finger | 0  # thumb
    for idx, point in enumerate(points):

        dist = self.get_signed_dist(point[:2])
        dist2 = self.get_signed_dist(point[1:])

        try:
            ratio = round(dist / dist2, 1)
        except:
            ratio = round(dist / 0.01, 1)

        self.finger = self.finger << 1
        if ratio > 0.5:
            self.finger = self.finger | 1

# Handling Fluctations due to noise
def get_gesture(self):
    if self.hand_result == None:
        return Gest.PALM
```

```
current_gesture = Gest.PALM
if self.finger in [Gest.LAST3, Gest.LAST4] and self.get_dist([8, 4]) < 0.05:
    if self.hand_label == HLabel.MINOR:
        current_gesture = Gest.PINCH_MINOR
    else:
        current_gesture = Gest.PINCH_MAJOR

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

else:
    current_gesture = self.finger

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

self.prev_gesture = current_gesture

if self.frame_count > 4:
    self.ori_gesture = current_gesture
return self.ori_gesture
```

```
# Executes commands according to detected gestures
class Controller:
    tx_old = 0
    ty_old = 0
    trial = True
    flag = False
    grabflag = False
    pinchmajorflag = False
    pinchminorflag = False
    pinchstartxcoord = None
    pinchstartycoord = None
    pinchdirectionflag = None
    prevpinchlv = 0
    pinchlv = 0
    framecount = 0
    prev_hand = None
    pinch_threshold = 0.3

    def getpinchylv(hand_result):
        dist = round((Controller.pinchstartycoord - hand_result.landmark[8].y) * 10, 1)
        return dist

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

    def changesystembrightness(self):
        currentBrightnessLv = sbcontrol.get_brightness() / 100.0
        currentBrightnessLv += Controller.pinchlv / 50.0
        if currentBrightnessLv > 1.0:
            currentBrightnessLv = 1.0
        elif currentBrightnessLv < 0.0:
            currentBrightnessLv = 0.0
        sbcontrol.fade_brightness(int(100 * currentBrightnessLv), start=sbcontrol.get_brightness())
```

```
def changesystemvolume(self):
    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(self):
    pyautogui.scroll(120 if Controller.pinchlv > 0.0 else -120)

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

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

```
    delta_y = y - Controller.prev_hand[1]

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

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

def pinch_control_init(hand_result):
    Controller.pinchstartxcoord = hand_result.landmark[8].x
    Controller.pinchstartycoord = hand_result.landmark[8].y
    Controller.pinchlv = 0
    Controller.prevpinchlv = 0
    Controller.framecount = 0

# Hold final position for 5 frames to change status
def pinch_control(hand_result, controlHorizontal, controlVertical):
    if Controller.framecount == 5:
        Controller.framecount = 0
        Controller.pinchlv = Controller.prevpinchlv

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

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

    lvx = Controller.getpinchxlv(hand_result)
```

```
        lvy = Controller.getpinchylv(hand_result)

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

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

    def handle_controls(gesture, hand_result):
        x, y = None, None
        if gesture != Gest.PALM:
            x, y = Controller.get_position(hand_result)

        # flag reset
        if gesture != Gest.FIST and Controller.grabflag:
            Controller.grabflag = False
            pyautogui.mouseUp(button="left")

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

        if gesture != Gest.PINCH_MINOR and Controller.pinchminorflag:
            Controller.pinchminorflag = False

        # implementation
```

```
        if gesture == Gest.V_GEST:
            Controller.flag = True
            pyautogui.moveTo(x, y, duration=0.1)

        elif gesture == Gest.FIST:
            if not Controller.grabflag:
                Controller.grabflag = True
                pyautogui.mouseDown(button="left")
            pyautogui.moveTo(x, y, duration=0.1)

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

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

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

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

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

```
'''
---------------------------------------- Main Class ----------------------------------------
    Entry point of Gesture Controller
'''


class GestureController:
    gc_mode = 0
    cap = None
    CAM_HEIGHT = None
    CAM_WIDTH = None
    hr_major = None  # Right Hand by default
    hr_minor = None  # Left hand by default
    dom_hand = True

    def _init_(self):
        GestureController.gc_mode = 1
        GestureController.cap = cv2.VideoCapture(0)
        GestureController.CAM_HEIGHT = GestureController.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
        GestureController.CAM_WIDTH = GestureController.cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    def classify_hands(results):
        left, right = None, None
        try:
            handedness_dict = MessageToDict(results.multi_handedness[0])
            if handedness_dict['classification'][0]['label'] == 'Right':
                right = results.multi_hand_landmarks[0]
            else:
                left = results.multi_hand_landmarks[0]
        except:
            pass

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

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

        if GestureController.dom_hand == True:
            GestureController.hr_major = right
            GestureController.hr_minor = left
        else:
            GestureController.hr_major = left
            GestureController.hr_minor = right

    def start(self):

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

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

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

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

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

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

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

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

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


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

# CHAPTER 8

# RESULT AND OUTPUT

## 8.1 RESULT

In the proposed AI virtual mouse system, the concept of advancing the human-computer interaction using computer vision is given. Cross comparison of the testing of the AI virtual mouse system is difficult because only limited numbers of datasets are available. The hand gestures and finger tip detection have been tested in various illumination conditions and also been tested with different distances from the webcam for tracking of the hand gesture and hand tip detection. An experimental test has been conducted to summarize the results. The test was performed 25 times by 4 persons resulting in 600 gestures with manual labelling, and this test.

### 8.1.1 NO MOVEMENT IMAGE


**Figure 8.1 No Movement Hand Gesture**

### 8.1.2 MOVING CURSOR


**Figure 8.2 Movement of Cursor Through Hand Gesture**

### 8.1.3 DOUBLE CLICK IMAGE

**Figure 8.3 Double Click Through Hand Gesture**

Test has been made in different light conditions and at different distances from the screen, and each person tested the AI virtual mouse system 10 times in normal light conditions, 5times in faint light conditions, 5 times in close distance from the webcam, and 5 times in long distance from the webcam, and the experimental results are tabulated, it can be seen that the proposed AI virtual mouse system had achieved an accuracy of about 99%. From this 99% accuracy of the proposed AI virtual mouse system, we come to know that the system has performed well. the accuracy is low for "Right Click" as this is the hardest gesture for the computer to understand. The accuracy for right click is low because the gesture used for performing the particular mouse function is harder. Also, the accuracy is very good and high for all the other gestures. Compared to previous approaches for virtual mouse, our model worked very well with 99% accuracy. The graph of accuracy, the proposed AI virtual mouse model in terms of accuracy.

It is evident that the proposed AI virtual mouse has performed very well in terms of accuracy when compared to the other virtual mouse models. The novelty of the proposed model is that it can perform most of the mouse functions such as left click, right click, scroll up, scroll down, and mouse cursor movement using finger tip detection, and also, the model is helpful in controlling the PC like a physical mouse but in the virtual mode.

# CHAPTER 9

## 9.1 CONCLUSION

The main objective of the AI virtual mouse system is to control the mouse cursor functions by using the hand gestures instead of using a physical mouse. The proposed system can be achieved by using a webcam or a built-in camera which detects the hand gestures and hand tip and processes these frames to perform the particular mouse functions. From the results of the model, we can come to a conclusion that the proposed AI virtual mouse system has performed very well and has a greater accuracy compared to the existing models and also the model overcomes most of the limitations of the existing systems. Since the propose 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.

## 9.2 FUTURE SCOPE

The proposed AI virtual mouse has some limitations such as small decrease in accuracy of the right click mouse function and also the model has some difficulties in executing clicking and dragging to select the text. These are some of the limitations of the proposed AI virtual mouse system, and these limitations will be overcome in our future work. Furthermore, the proposed method can be developed to handle the keyboard functionalities along with the mouse functionalities virtually which is another future scope of Human-Computer Interaction (HCI).

There are several features and improvements needed in order for the program to be more user friendly, accurate, and flexible in various environments. The following describes the improvements and the features required.

**a) Smart Movement:** Due to the current recognition process are limited within 25cm radius, an adaptive zoom in/out functions are required to improve the covered distance, where it can automatically adjust the focus rate based on the distance between the users and the webcam.

**b) Better Accuracy & Performance:** The response time are heavily relying on the hardware of the machine, this includes the processing speed of the processor, the size of the available RAM, and the available features of webcam. Therefore, the program may have better performance when it's running on a decent machine with a webcam that performs better in different types of lightings.

**c) Mobile Application:** In future this web application also able to use on Android devices, where touch screen concept is replaced by hand gestures.

# CHAPTER 10

## REFERENCES


**[1]** 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.

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

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

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

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

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

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

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

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

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

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

**[12]** Google,MP,https://ai.googleblog.com/2019/08/on-devicereal-time-hand-tracking with.html.

**[13]** V. Bazarevsky and G. R. Fan Zhang. On-Device, MediaPipe for Real-Time Hand Tracking.

**[14]** K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Realtime computer vision with openCV,"