# ABSTRACT

The Al Virtual Mouse uses computer vision techniques to track hand movements and translates them into cursor movements on the screen. The system is designed to be intuitive and user-friendly, allowing users to interact with their computer without the need for a physical mouse. The virtual mouse is developed using Python and OpenCV libraries. The project includes the implementation of various image processing algorithms, such as hand segmentation, feature extraction, and classification. Moreover, it is robust to various lighting conditions, backgrounds, and hand sizes. The developed system provides an alternative to conventional mouse devices, particularly for individuals with disabilities or those who prefer a more natural way of interacting with their computers. The target of this project is the invention of something new in the world of technology that helps an individual work without the help of a physical mouse. It will save the user money and time. Real-time images will be continuously collected by the Virtual Mouse color recognition program and put through a number of filters and conversions. When the procedure is finished, the program will use an image processing technique to extract the coordinates for the position of the desired colors from the converted frames. The virtual mouse system is evaluated on various metrics, such as accuracy, speed, and robustness, and compared with existing virtual mouse systems. The trial findings demonstrated a high degree of accuracy 97.37%; the system can operate well in actual scenarios with just one CPU. Following that, it will compare the current color schemes within the frames to a list of color combinations, where various combinations correspond to various mouse operations. If the current color scheme matches, the program will execute a mouse command, which will be converted into a real mouse command on the user's computer.

s

|  |  |  |
| --- | --- | --- |
| **Symbol Name** | **Symbol** | **Description** |
| Class |  | Class represents a collection of similar grouped together. |
| Associations |  | Association represent static relationship between classes. |
| Aggregation |  | Aggregation is a form of association; it aggregates several classes into single one. |
| Usecase | **Usecase** | A usecase is an interaction between the system and the external environment. |
| Actor | **Actor** | Actors are the users of a system. |
| Decision |  | Diamond indicates the decision. |
| Communication |  | It is the communication between various usecases. |
| State | State | It represents the state of a process, each state goes through various flows. |
| Initial State |  | It represents the initial state of the project. |
| Final State |  | It represents the final state of the project. |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **MHI** | Motion History Image |
| **HCL** | Human Computer Interaction |
| **LED** | Light Emitting Diodes |
| **OPEN CV** | Open-Source Computer Vision |
| **RGB** | Red, Green, Blue |
| **HSV** | Hue, saturation and Values |
| **GUI** | Graphical User Interface |
| **IDE** | Integrated Development Environment |
| **CNN** | Convolutional Neural Networks |
| **SVM** | Support Vector Machines |
| **HMM** | Hidden Markov Models |

# INTRODUCTION

## Introduction

In computer jargon, a computer mouse is a directing device that recognizes two-dimensional motions in respect to a surface. This movement is converted into the movement of the cursor on a display in order to manipulate the GUI, or Graphical User Interface, on a computer platform. It's difficult to fathom living in our high-tech day without computers. Another of the greatest innovations ever made by humans is the computer. For people of all ages, using a computer has become a necessity in practically every aspect of daily life. We frequently use computers in daily life to facilitate our job. Consequently, HCI (Human- Computer Interaction) has become a popular area of study. The advent of the laser mouse in 2004 helped to overcome the optical mouse's drawbacks, including its inability to accurately track highly reflective surfaces, by enhancing movement accuracy with even the smallest hand movements . No matter how precise a mouse is, however, there are still physical and technical constraints that must be considered. Since the release of a mobile device with touch screen technology, people have begun to demand that the same technology be used on all other technological devices, including desktop computers. Although touch screen technology for desktop computers already exists, the cost can be prohibitive. In this project, a finger tracking-based virtual mouse application will be designed and implemented using a regular webcam. To implement this, we will be using the object tracking concept of Artificial Intelligence and the OpenCV module of Python. Therefore, an alternative to the touch screen could be a virtual human computer interaction device that uses a webcam or other image capturing devices to replace the actual mouse and keyboard. A software program will continuously use the webcam to track the user's gestures, process them, and translate them into the motion of a pointer, much like physical mouse.

## Review of Physical Mouse

Modern technology has many different physical computer mouse types, and the following discussion will discuss these types and their differences.

#### Mechanical Mouse

A mechanical mouse makes use of the motion of a ball inside it, which is connected to two wheels placed

perpendicular to each other. These wheels are responsible for left/right and up/down movement detection of the ball, and hence send the corresponding motions to cursor on the screen. The mechanical mouse became a nearly universal tool for computer interaction in the 1980s, and continued to be dominant through the 1990s. The mechanical mouse is now largely considered obsolete, replaced by the lightweight and low-cost optical mouse. They are similar in shape and function, but instead of the ball, they rely on optical sensors, which tend to be more reliable. Furthermore, the switches in mouse buttons are not any different either, as continued use may cause the internal mechanics to become loose and render the buttons ineffective until they are disassembled and repaired.



**Figure 1.1** Mechanical Mouse, with Top Cover Removed .

The benefits and drawbacks of the mechanical mouse are listed in the following points Advantages of mechanical mouse:

* + - * Let’s users move the mouse to control the computer system.
      * Offers precise mouse movement tracking.

Disadvantages of mechanical mouse:

* The mouse rollers and button switches are prone to deterioration, which makes them malfunction.
* Needs a flat surface to function.

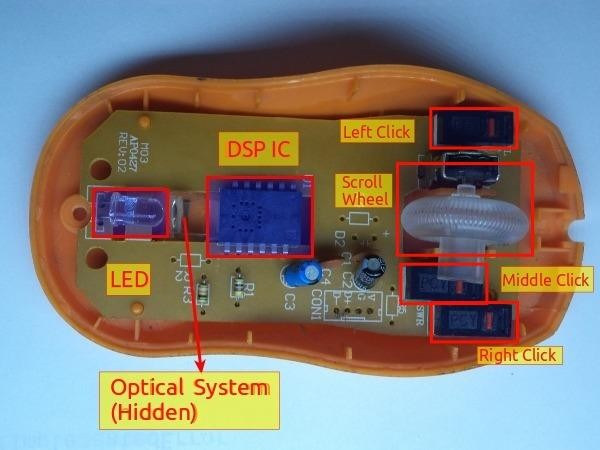
#### Optical Laser and Mouse

The motions of an optical mouse, which is widely used today, rely on LEDs, or light-emitting diodes, to detect movements in relation to the surface underneath them, but a laser mouse is a type of optical mouse that makes use of coherent laser lights. The optical mouse replaces the mechanical mouse, which relied on rollers to control movement, by using an image array of photodiodes to identify movement.

Components of an optical mouse:

The optical system consists mainly of three parts:

* Lens
* Light source
* Sensor.

These three parts are assembled on a custom base plate, i.e. Printed Circuit Boards (PCB), and the clip. The lens is the largest part and is mounted on the base plate of the mouse. In addition, it is still susceptible to button switch degradation, which will again result in improper mouse operation unless it is disassembled and repaired. Additionally, prolonged use without proper cleaning or maintenance can result in dust particles getting trapped between the LEDs, making it difficult for both optical and laser mice to detect surfaces.

**Figure 1.2** Optical Mouse with Top Cover Removed .

The benefits and drawbacks of the optical and laser mice are listed in the following points - Advantages of the Optical and Laser Mouse:

1. Enables more accuracy with fewer hand movements.
2. Extended life span.

Disadvantages of the Optical and Laser Mouse:

* + Susceptible to button switch deterioration.
  + Does not perform as intended when placed on a polished surface.

## 1.2Problem Statement

The proposed AI virtual mouse system can be used to overcome problems in the real world such as situations where there is no space to use a physical mouse and also for the persons who have problems in their hands and are not able to control a physical mouse. Also, amidst of the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera.[6] The following describes the general problem that the current physical mouse suffers:

* Special hardware is required for a physical mouse.
* The performance of a physical mouse changes according on the surroundings, and it is difficult to adjust to varied environments.
* The mouse's operational surroundings are severely constrained.
* Both wired and wireless mice have their own life expectancies.

## Motivation of Virtual Mouse

It is safe to predict that the Virtual the Mouse will soon take the place of the conventional physical in nature mouse in the not-too-distant future, as people strive to live in a world where every technological appliance can be operated and interacted with remotely without the need for any peripheral devices, such as remote controls, keyboards, etc. Not only does it offer ease, but it also saves money.

* + 1. **User-Friendly**

Man-made consciousness has different applications in the present society. It is becoming fundamental for the present time since it can take care of complicated issues with an effective way in various ventures, like

Health care, diversion, finance, schooling, and so forth Computer based intelligence is making our everyday existence more agreeable and quicker. It is well known that users must utilize a physical mouse in order to communicate with the computer system. This mouse requires a specific area of surface to work, in addition to having cable length restrictions. Virtual Mouse doesn't need any of that since all that is needed is a camera to take pictures of the user's hand position and utilize those images to establish where the points should be.

**1.3.2 Cost Effective**

An Al virtual mouse, also known as a software-based mouse or an on-screen mouse, can be cost effective compared to a physical mouse for a few reasons:

1. No need for hardware: A virtual mouse does not require any additional hardware, such as a physical mouse, to function.
2. Accessibility: A virtual mouse can be used by individuals who may have difficulty using a physical mouse due to physical limitations, such as a disability.
3. Compatibility: A virtual mouse is typically compatible with most computer systems, regardless of the hardware or operating system being used.
4. Ease of use: A virtual mouse is typically easy to use and requires minimal training.

## Project Perspective

The scope of this project is to develop a virtual mouse that will be operated without touching any device or screen. In today’s world where we are adjusting our living while being in a pandemic, a touch less mouse controller will be useful to eliminate the risk of spreading infection through touch on public service devices. A virtual mouse will be introduced soon to replace the physical computer mouse in order to promote convenience while still allowing accurate interaction and control of the computer system. The virtual mouse can be used without touching the screen. This project can improve the scope of Human Computer Interaction technology to be explored more.

## Project Goal

This project's objective is to build a Virtual Mouse application that emphasizes a few crucial programming concepts.

The following describes the overall goals of this project:

1. The precision of the suggested AI virtual mouse is about 98%, which is higher than that of other systems that have already been put into use.
2. Amidst the COVID-19 condition, it is not safe to use the devices by touching them because it may result in a hypothetical situation of propagation of the virus by contacting the devices, therefore the proposed AI virtual mouse can be utilized to control the PC mouse operations without using the actual mice
3. Drawings in two dimensions and three dimensions can be produced using the same technique.
4. Virtual reality and augmented reality games can be played with AI without the requirement of wireless or wired mouse devices.

## Impact, Significant and Contribution

The usage of a hardware computer mouse in conjunction with manual mouse inputs and mouse positioning is projected to be replaced by the Virtual Mouse program. With the use of gestures, every job can be completed with this program, making computer use easier. Furthermore, by simply displaying the right combination of colors to the webcam, the Virtual Mouse program enables persons with motor impairments to interact with the computer.

# 2.LITERATURE REVIEW

## Literature Review

There are numerous ways to control the cursor with a hand gesture, but for the longest period of time, Data Glove must be worn. It lessens the effectiveness of the user's and the system's performance. The system's complexity in this method is a major problem.

Adaptive skin color models and a motion history image-based hand moving direction detection technique are implemented in a paper published by Dung-Hua Liou and Chen-Chiung Hsieh. The average accuracy of this project was 94.1%, and processing takes 3.81 milliseconds per frame. The primary problem with paper is that it has trouble recognizing more complex hand gestures when used in a working environment.

This paper mainly applied visional hand gesture identification to the HCI interface, holding control usage, written by Chang-Yi Kao and Chin-Shyurng Fahn. According to experimental findings, the face tracking rate is over 97% under typical circumstances and over 94% when the face has temporal occlusion. The system's execution efficiency is excellent, and we are inspired to market the robot soon. High configuration computers are required for accurate results.

The primary goal of this research was to create a real-time hand gesture detection system based on the skin color model, which was published by Angel et al. Since hand gestures may readily communicate thoughts and activities, employing these different hand forms, when spotted by the gesture recognition system and processed to create related events, have the potential to give a more natural interface to the computer vision system. However, it was unable to function in a complex environment and was only calculable in proper lighting.

A Machine-user interface that performs hand gesture recognition using multimedia techniques and basic computer vision. A paper was published on this topic by Ashwini M. Patil et al. Before utilizing the gesture comparison algorithms, they discovered a significant limitation. From the stored frames, hand segmentation and skin pixels must be completed.

A camera was used to capture hand motions using color detection methods in this project. The utilization of a web camera is the essential component of this technique. Abhik Banerjee and Abhirup Ghosh wrote this paper to cost-effectively construct a virtual human-computer interface device. There were some restrictions on their work, such as the need for a light operating system background and the absence of objects with vivid colors. Computers with a specific high configuration function well.

In this study, which Yimin Zhou et al. reported, where a high-level hand feature extraction approach for real-time gesture detection was provided. The created system has good accuracy in both the extraction of flexional and extensional fingers. However, only computers with high configurations can use this method.

Several color bands were used in this experiment, which was described by Pooja Kumari et al., where various colored bands carried out various tasks. The number of colors is used as the key to control mouse actions. But the system was managed by a number of colors. Instead of using different gestures, the number of colors is used to perform a function. This paper based on a background extraction and contours detection system was proposed by Aashni Haria et al. where they conducted two sets of assessments in order to determine the correctness of their method. In the initial round of evaluations, they made use of settings with a variety of uniformly simple backdrops. For the second assessment, they utilized backdrops that had a number of discrepancies. Ten times were given for each gesture in each setting the numbers' average. The accuracy gained was 85% and 80%, which was calculated as the percentage of times a given gesture was successfully identified. But working with it is incredibly slow. The operation of a cursor control system using hand gestures captured from a webcam through a color detection technique performed in this project which was published by Abhilash SS et al. However, it was limited to a few mouse actions and is inoperable against a static background. A detailed explanation of the algorithms and methodologies for the color detection of a virtual mouse was given in this project by Kollipara Sai Varun et al. In this paper, Open CV (Open-Source Computer Version Library) is primarily used for video capture. The highlight color provided by the user for mouse movement is used in this paper for color detection and mouse movement. This project based can be helpful for presentations as well as for minimizing workspace requirements and the weight of additional hardware. A common way to interact with computers without a mouse device is by using fingertip tracking as a virtual mouse. Kabid Hasan Shibly et al present a novel virtual mouse technique in this paper that makes use of fingertip detection and RGB-D images. The system

captures frames using a webcam or built-in cam and processes the frames to make them track-able and then recognizes different gestures made by users and performs the mouse function. The proposed system eliminates device dependency in order to use a mouse and can be proved beneficial in order to develop HCI technology. The proposed system is implemented in Python programming language using the Computer Vision based library OpenCV and has the potential to replace the typical mouse and remote controller of machines.

The primary goal of the AI simulated mouse device is to replace the need of a hardware mouse with hand gestures for cursor control. This project provided 99% accuracy, which is significantly higher than the other proposal. In which hand gesture recognition movement created a virtual mouse. In this study, hand tracking data were used which was published by B. Nagaraj et al.

Object detection (OD), salient object detection (SOD), and category-specific object detection are just a few of the object detection techniques that Ahmed, Muhammad, et al., explain in great detail (COD). It examines various currently used deep learning methods and evaluates how well they perform in difficult environments. By comparing their performance in terms of output, time required, and comparison to more conventional methods, it illustrates recent research and advancements in this field. Additionally, it discusses the publicly accessible datasets for this task and explains the comparison evaluation metrics. It also provides a brief overview of the drawbacks and the direction that more advancements might be made.

V. Tiwari et al. achieve image classification using the VGC16 pre-trained model. It contrasts the outcomes with various models, including the baseline CNN and three-block VGG model. In order to investigate its impact on accuracy, the paper also included the VGC3 model's data augmentation. The accuracy of the implemented VGC16 model is 98.97%, which is significantly better than the accuracy of the baseline CNN, VGC3, and VGC3+Data augmentation models, which are respectively 55.075%, 74.561%, and 61.404%.

A model with server side and client-side components was proposed by Li Wensheng and colleagues. Server-side: Adaptive online training is used for mouse movement, finger detection using a BP neural network, finger tracking using a mean shift algorithm, and appropriate messages are created and sent to the client. Client-side: responds to the messages, calls the server's API function to get the coordinates of multiple fingertips, and processes the messages appropriately to take control of the application. Because of the different skin tones, the results from adaptive online training are inconsistent.

Data glove is a proposed additional device by Kumar et al. The K-NN classifier is used to classify the gestures based on the data glove's measurements of the hand's current position and the angles between its joints. An IR camera, projector, and laptop system were part of the model proposed by A. Mhetar et al. The idea of a virtual marker is the model's foundation. The interfaced communication for the IR camera is connected to the laptop. The high-end microcontroller used in the virtual marker is configured as a Human Interface Device (HID) for faster response and is connected to an IR camera interface to provide mouse-like functionality. The mouse pointer is moved to the location indicated by the co-ordinates by Teensy after the IR Camera tracks IR sources and sends information about its position. The presence of an infrared camera that meets certain technical specifications is necessary for the model to operate.

A model that incorporates hand pointing gestures along with other hand gestures in 3D space was proposed by S.M.S. Shajideen and V. H. Preetha. To get a top and side view of various hand gestures, two USB cameras that are orthogonal to one another are used. Software for it is MATLAB. The two detectors are trained and selected various image samples for various top and side views for the two separate views. During the training phase, binary patterns are used for each sample's feature generation and image conversation. Then, two cascade detectors that depend on the choice of AdaBoost features were built. Each and every detector monitors and scans the working image during the testing phase, which involves converting the input image to the working image.

Models that are based on color detection and mouse movement based on highlighted colors provided by the user were developed by K.S. Varun et al. It is possible to see a two-figure input that creates two rectangles and an average point from both figures. It will function like the mouse pointer. The mouse pointer in the runtime follows the moving point as it moves. Therefore, using this, mouse movement can be implemented. The position of the predetermined colored caps in the mask that is created for system comprehension determines how the mouse pointers are updated. In order to detect the predetermined colored objects that will aid in mouse movement, the created mask is converted from an RGB background to a black and white image and provided 84% accuracy. If the predetermined colored caps blend in with the background, they won't be seen and no mouse movement will be possible. An all-keyboard and mouse model was suggested by S. R. Chowdhury et al. The Mouse operates using a convex hull process; flaws are recorded or read, and using these flaws, the Mouse's functions are mapped. The convex hull treats the gap between the fingers as a defect because this image recognition process only considers defects and conditional statements, allowing it to be used for a variety of gestures and mapping commands.

A different kind of model was presented by Sai Mahitha G. et al. By putting our fingers in front of the computer's web camera, we can control the mouse cursor in this model. These finger gestures are recorded and managed using a webcam's Color Detection technique. With this system, we can move the system pointer by using our fingers that have colored tapes or caps on them, and actions like dragging files and left-clicking are carried out by making specific finger gestures. Additionally, it handles file transfers between two PCs connected to the same type of network. Only a webcam with low resolution is used by this developed system, acting as a sensor to track the user's hands in two dimensions. The mouse cannot be moved if the predetermined colored caps blend in with the background because they won't be seen and accuracy is 97%.

The virtual mouse method proposed by Tran, DS, et al. uses fingertip detection and RGB-D images. Using detailed skeleton-joint information images from a Microsoft Kinect Sensor version 2, the hand region of interest and the palm's center are first extracted, and they are then converted into a binary image. A border-tracing algorithm is then used to extract and describe the hands' contours. Based on the coordinates of the hand contour, the Kcosine algorithm is used to determine the location of the fingertip. Finally, the mouse cursor is controlled using a virtual screen by mapping the fingertip location to RGB images. Multiple restrictions that are primarily carried over from Microsoft Kinect continue to plague this study.

Two different types of mouse control implementation methods were proposed by V. V. Reddy et al. in their paper. One makes use of color caps, and the other recognizes gestures made with bare hands. It is divided into two categories: "gesture recognition" and "fingertip detection" using colored caps. It entails integrating the video and processing the images through background removal. By ignoring the steady objects and only taking into account the foreground objects, background subtraction helps. Fingertip detection entails finger guessing, circle identification, and color identification. Gesture recognition entails identifying the skin tone, detecting contours, forming convex hulls, and then inferring the gesture. The appropriate mouse operation can be carried out. This model served as the foundation for our research. We have researched the model's background subtraction that will be used in our project. Convex hull is used in this model to recognize gestures; however, convolutional neural networks will replace convex hull in order to improve this model's accuracy.

A novel virtual-mouse method using RGB-D images and fingertip detection techniques was implemented in this project. which was published by Dinh-Son Tran et al. The hand region of interest and the center of the palm are first extracted from depth images provided by the Kinect V2 skeletal tracker and converted to binary images. The hand contours are extracted and described by a border- tracing algorithm. The K-cosine algorithm is used to detect the fingertip location, based on the hand- contour coordinates. Finally, the fingertip location is mapped to RGB images to control the mouse cursor based on a virtual screen and provided 96.13% accuracy. The proposed system works with a single low-cost CPU without the help of a graphics processing unit (GPU), has fast detection in real- time (30 frames per second (fps)), and allows execution on computer screens with many types of resolution. It provides simultaneous fingertip tracking for up to six people and selects the main person to control the mouse cursor, focusing on the right hand.

The system described in this paper by Aabha Waichal et al. uses a Convolutional Neural Network (CNN) model based on hand gesture recognition to control the mouse. A mouse is a pointing tool that facilitates simple human-computer interaction (HCI). It has been investigated to use pre-processing methods like k-cosine and border-tracing, background subtraction, and computing four motion matrices along with image processing methods like 3D convolutional neural network, contour and convex hull area. Using the built-in webcam to record the live feed, this paper proposes an interactive method of controlling the movement of the mouse by hand gesture. In this project, a practical method of controlling a mouse virtually while using a live camera was proposed. They have suggested mouse movements, clicks, scrolling (up and down), and zooming in and out. The strategy involves taking a live feed, taking out the background, and sending it to the CNN model. High accuracy is provided by the CNN model. In complex backgrounds, we can also deliver good results by using background subtraction. CNN model is trained by the dataset. It appears that hand motions taken from a camera employing an HSV color detection technique be utilized to operate the mouse cursor in a paper written by Prof. Monali Shetty et al. Using colored caps or tapes that the computer's webcam tracks, this system enables users to move the system cursor. They can also use various hand gestures to perform mouse actions like left-, right-, and double-clicks. The system is implemented using real-time computer vision in Python and the OpenCV library and provided accuracy 95%. The monitor shows the camera's output.

The idea of a virtual mouse using sixth sense technology has been put forth in this paper by Swati Tiwari et al. because it is highly responsive in real-time applications and uses gestures for interaction.

We looked into hand gesture control for a low-cost, high-performance virtual mouse. For object recognition in this project, they have been used color tapes. By measuring the distance between the thumb and middle finger and the index and middle fingers, respectively, the left and right click events of the mouse have been achieved. When a calibrated pair of cameras is looking down at the hands with the palms facing downward, the system can specifically track the positions of the index finger and middle finger tips and finally provide an accuracy 93%.

A system that uses head and facial movements to control the mouse was proposed by T. Palleja et al. It computes four motion matrices using an algorithm for image processing. The region of interest is used to analyze the ten-frame cumulative image and find the movement. The process takes some time, which slows down how quickly the mouse reacts.

This paper published by Rachit Puri where he presents the maneuver of mouse pointer and performs various mouse operations such as left click, right click, double click, drag etc using gestures recognition technique. The approach is based on calculation of three combined features of hand shape which are compactness, area and radial distance. The algorithm implemented in this paper is divided into seven main steps. The proposed approach is based on detection of number of target colours (region of interest) that triggers the mouse event according to the gesture formed. The implementation has been divided into various steps such as selection of RGB, YCbCr conversion, finding region of interest, storing values and last mouse event and provide 95% accuracy. A gesture will be recognized increases with the percentage of recognition rate.

The only input device needed for the paper which is published by Vijay Kumar Sharma et al. is a webcam. Python and OpenCV are the software programs needed to implement the suggested system. On the system's screen, the output from the camera will be seen so that the user may adjust it further. NumPy, math, and wx will be used as dependencies in Python to construct this system. and mouse. Making the machine more interactive and reactionary to human behaviour was the goal of this work. This paper's only objective was to provide a term that is portable, inexpensive, and compatible with any common operating system. By identifying the hand of human and directing the mouse pointer in that hand's direction, the proposed system operates to control the mouse pointer. The program Control basic mouse actions including left-clicking, dragging, and cursor movement.

The unique method for human computer interaction (hci) presented in this research published by Prachi Agarwal et al. uses a real-time camera to control cursor movement. The software applications required for the suggested device are OpenCV and python, and a webcam will be needed as an input device.

The system's display screen may show the camera's output, and the dependencies for Python are NumPy, math, wx, and mouse. In order to contribute to future vision-based human-machine interaction, they used computer vision and HCI (Human Computer Interaction) in this work. The topic of the proposed article is employing hand gestures to control mouse functionalities. Mouse movement, left- and right-button taps, double taps, and up- and down-scrolling are the primary actions. Users of this system can select any color from a variety of hues. The users may choose any color from the bands of colors that match the backdrops and lighting situations. There are a limited number of color bands defined.This could change depending on the background. For instance, the system will give the user the option to select a color from a variety of hues (Green, Yellow, Red, and Blue) when they first turn it on.

A technique for manually altering the mouse cursor's location without the assistance of an electrical device is suggested in this study by Sankha Sarkar et al. With various hand motions, we may effortlessly perform actions like clicking and dragging objects. The only necessary input device for the suggested system is a webcam. The software will need to be developed using Python and OpenCV. The system's screen will show the webcam's output so that the user may adjust it further. NumPy, Atopy, and Media pipe are the Python requirements that will be utilized to create this system.

The AI virtual mouse system was created by Abhishek R. Shukla using the OpenCV package and the Python programming language. The Media Pipe software is used by the proposed AI virtual mouse system to track hands and perform hand stunts. One of the remarkable inventions of HCI (Human Computer Interaction) is the mouse. As a battery is required for power and a dongle is required to interface the mouse to the PC, even a wireless or Bluetooth mouse utilizes gadgets. They are not entirely device-free as a result. This issue can be resolved by the proposed AI virtual mouse system using a webcam or built-in camera to capture hand motions and recognize hand tips using computer vision. The algorithm for machine learning is incorporated into the system's algorithm. Without a physical mouse, the computer can be handled digitally and can perform left-click, right-click, panning, and computer cursor tasks based on hand motions. The methodology used for hand detection is based on deep learning. A low cost and high-performance virtual mouse was invented by Tasi et al. The proposed system makes use of a number of strategies. To identify the foreground/background area and capture the region-of-interest, motion detection and skin detection methods are used. The connected component labelling algorithm is used to determine an object's centroid. Recognizing the hand area and associated gestures uses the removal on arm and convex hull algorithms.

A computer software that can recognize user inputs like hand gestures or voice commands and replicate mouse movements and clicks in accordance is the end result of building a virtual mouse utilizing algorithms and GUI automation. The execution of the program and the precision of the machine learning algorithm employed will determine the precise outcome.

The virtual mouse ought to be capable of deciphering user inputs with precision and responding with the required mouse movements. The virtual mouse's accuracy will be influenced by a number of variables, such as the quantity and quality of training data, the machine learning method selected, and the interface with GUI automation tools."

The AI virtual Mouse was created by Omkar Shinde et al by using Python programming languages well as OpenCV a computer vision library. Python and the OpenCV computer vision library were both used to construct the AI virtual mouse system. The Media Pipe package, Pynput, Autopy, and PyAutoGUI packages, as well as the suggested AI virtual mouse system's model, are used to track the hands and the tips of the hands for navigating the window screen of the computer and doing actions like scrolling, clicking, and left-clicking. The proposed model can perform incredibly well in real- world applications with just a CPU and no GPU, and the results showed a very high level of accuracy.

In order for the user to travel to the mouse control system and type on the virtual keyboard using a yellow cap on his fingertip, computer vision was used which was written by Dipankar Gupta et al. in this paper. When in mouse control mode, the user can only utilize a variety of fingers to carry out all mouse function. In this research, they describe a revolutionary cross-interactive mouse and keyboard system in which, in place of a traditional keyboard and mouse, they use the device's camera to recognize and track a color.

The research paper which was published by K. Bharath Reddy et al. where they design an AI virtual mouse by using Python programming language, MediaPipe and OpenCV for working with image processing performing computer vision like face detection and object tracking etc. The accuracy of this project is quite high and functional.

The paper published by Roshan Hyalij et al. with a Human Computer Interaction (HCL) approach to controlled the cursor movement by using the device camera. It become more accessible because of using finger detection for instant camera access and user-friendly user interface. The system is used to implement motion tracking mouse, signature input device and application selector.

To introduced virtual mouse using hand gesture and voice assistant a paper was published Khushi Patel et al. The application exhibits good time-based performance based on the suggested algorithm and chosen hand characteristic. It is simpler for the user. The scientific community is still curious about how hand gestures are used to regulate or communicate. It is based on computer vision algorithm and can do any mouse related task.

Gauri et al. in their paper perform the technologies of human-computer interaction connected to biometric identification and tracking. They suggest a position-based head motion detection technique based on the face detection approach that is independent of the precise biometric tracking and identification. It detects eye opening and closing events using the feature classification method. They also create a software system that uses head and eye movement images to control computers. The various mouse events, such as move, click, drag, and so on, are mapped to permutation of head and eye movements. The upper limb impaired who were unable to utilize the conventional mouse and keyboard can use this device.

## Literature Review Summary

**Table 2.1:** Literature Review Summary

|  |  |  |
| --- | --- | --- |
| **`** | **Algorithm** | **Result Finding** |
| Hand Gesture | OpenCV | Adaptive skin color models and motion history image-based hand moving direction detection achieved 94.1% accuracy, but difficulty recognizing  complex hand gestures. |
| Hand  Gesture | MediaPipe, OpenCV | High configuration computers are needed for accurate results in visional hand  gesture identification for HCI interfaces. |
| Hand Gesture | OpenCV, Autopy, MediaPipe NumPy | This research aimed to create a real-time hand gesture detection system based  on the skin color model, which could provide a more natural interface to computer vision systems. |
| Hand  Gesture | MediaPipe, OpenCV | Hand gesture recognition using multimedia and computer vision, but limited by  hand segmentation and skin pixels. |
| Hand  Gesture | MediaPipe, OpenCV | A web camera was used to capture hand motions using color detection methods. |
| Hand  Gesture | Open CV | High-level hand feature extraction approach for real-time gesture detection. |
| Hand  Gesture | MediaPipe, OpenCV | The number of colors is used to control mouse actions and perform a function. |

|  |  |  |
| --- | --- | --- |
| Hand Gesture | HAND TRACKING, FINGER COUNTER, GESTURE VOLUME  CONTROL | Conducted two sets of assessments to determine the accuracy of their background extraction and contours detection system, with an average accuracy of 85% and 80%, respectively. |
| Hand  Gesture | Open CV | Cursor control system using hand gestures, limited to mouse actions and static  background. |
| Hand Gesture | Open CV | Open CV is used to detect virtual mouse color and mouse movement. A detailed  explanation of the algorithms and methodologies for the color detection of a virtual mouse was given. |
| Hand Gesture | Open CV | A novel virtual mouse technique that uses fingertip detection and RGB-D  images to capture frames and recognize gestures. It eliminates device dependency and can be used to develop HCI technology. |
| Hand  Gesture | Open CV | AI simulated mouse device provides 99% accuracy for cursor control. |
| Hand Gesture | Open CV | Object detection techniques, salient object detection, and category-specific  object detection are discussed and compared to more conventional methods, with comparison evaluation metrics and drawbacks. |
| Hand Gesture | CNN, SVM, HMM | VGC16 pre-trained model achieved 98.97% accuracy, significantly better than baseline CNN, VGC3, and VGC3+Data augmentation models. |
| Hand Gesture | CNN | Adaptive online training is used for mouse movement, finger detection, and finger tracking, but results are inconsistent due to skin tones. |
| Hand Gesture | SVM | Data glove, K-NN classifier, IR camera, projector, laptop system, virtual marker, high-end microcontroller, mouse pointer, and infrared camera are necessary for model to operate. IR camera tracks IR sources and sends  information about its position. |
| Hand  Gesture | CNN | Two USB cameras are used to get a top and side view of hand gestures, and  MATLAB software is used to train and select image samples. |
| Hand Gesture | CNN | Models based on color detection and mouse movement based on highlighted colors provided by the user can be implemented. The position of predetermined colored caps in the mask determines how the mouse pointers are updated, with  84% accuracy. |
| Hand  Gesture | Convex hull | all-keyboard and mouse model that uses a convex hull process to map functions. |
| Hand Gesture | Convex hull | A webcam with low resolution to track the user's hands in two dimensions, allowing them to control the mouse cursor and handle file transfers accuracy is  97%. |

|  |  |  |
| --- | --- | --- |
| Hand Gesture | Machine learning algorithms | The virtual mouse method uses fingertip detection and RGB-D images to control the mouse cursor. However, multiple restrictions remain from Microsoft Kinect. |
| Hand Gesture | CNN, SVM, HMM | Two mouse control implementation methods, one using color caps and one using bare hands. Background subtraction helps integrate video and process images, while fingertip detection involves finger guessing, circle identification, and color identification. Convolutional neural networks will replace convex hull to  improve accuracy. |
| Hand Gesture | CNN, SVM, HMM RNN, | The K-cosine algorithm detects fingertip location and maps it to RGB images to control the mouse cursor, providing 96.13% accuracy and simultaneous fingertip tracking for up to six people. |
| Hand Gesture | CNN, SVM, HMM | This paper proposes an interactive method of controlling a mouse by hand gesture using a Convolutional Neural Network (CNN) model based on hand gesture recognition. The strategy involves taking a live feed, taking out the  background, and sending it to the CNN model. |
| Hand  Gesture | SVM | Hand motions taken from a camera are used to operate the mouse cursor, with  accuracy of 95%, using real-time computer vision. |
| Hand  Gesture | Transformational  algorithm | Hand gesture control using color tapes and cameras to accurately track the  positions of the index and middle finger tips and an accuracy 93%. |
| Hand Gesture | Background subtraction algorithm  from OpenCV | System uses head and facial movements to control mouse, taking time to react. |
| Hand Gesture | K-cosine algorithm | A gesture recognition technique based on three features of hand shape: compactness, area and radial distance. The algorithm is divided into seven steps  and provides 95% accuracy, increasing with the recognition rate. |
| Hand Gesture | HSV algorithm | The paper proposes a system that uses a webcam, Python, OpenCV, NumPy,  math, and wax to control the mouse pointer and basic mouse actions. It is portable, inexpensive, and compatible with any operating system. |

## 2.3 Conclusion

Other than the conventional physical mouse interaction, there are many other ways to interact with a computer. Human computer interaction (HCI) techniques have evolved along with modern

programming and technology because they provide countless ways to interact with computers. It is not surprising that in the near future, video cameras that track gestures will take the place of physical mice and be used to create specialized/unique systems that cater to user needs, from gesture movement tracking to colored tracking.

# 3.SYSTEM ANALYSIS

## System Analysis

There are two main steps in the process of color recognition: the calibration phase and the recognition phase. In the calibration phase, which will be utilized later in the recognition phase, the system will be able to identify the Hue Saturation Values of the colors selected by the users. It will save the parameters and settings into text documents for later use. The system will begin to take frames during the recognition phase and look for color input based on the values that have been stored during the calibration process phase. The following figure depicts the stages of the virtual mouse:



***Recognition Phase***

System Calibration and

Settings Storage

***Calibration Phase***

HSV Frame Transition

HSV Values Extraction

Standard Deviation Calculation

Frame Noise Filtering

User's Color Input Acquisition

Real Time Image Acquisition

Execution of Mouse Action

Colors' Coordinates Acquisition

Color Combination Comparison

HSV Frame Transition

Binary Threshold Transition

Binary Threshold Morphological Transformation

Frame Noise

Filtering

Real Time Image

Acquisition

Webcam & Variables

Initialization

**Figure 1.3:** Virtual Mouse Block Diagram

#### Calibration Phase

* + - 1. Acquisition of Real Time Image

The application will begin by using a webcam to capture live photos while it waits for hu man input on color.To lessen the processing demands of processing the pixelin side the

cap tured frame, the size of the obtained image will be compressed to a manageable level.

* + - 1. User's Color Input Acquisition

The application captures frames made up of user-submitted input colors. The captured frame is then transmitted for processing, where it goes through a number of changes and calculations to provide calibrated HSV values.

* + - 1. Frame Noise Filtering

Every taken frame contains noises that will impair the program's efficiency and accuracy, so the frame must be noise-free. Filters must be added to the collected frames to remove the undesirable noise in order to accomplish that. For the current project, the widely used smoothing technique known as the Gaussian filter will be utilized to remove noise from a frame.

* + - 1. Transition of HSV Frame

It is necessary to convert the acquired frame from BGR to HSV format.

* + - 1. Extraction of HSV Values

The transformed frame must be split into three separate planes in order to obtain the HSV values; this can be accomplished by converting the frame from a multi-channel array into a single channel array using splitting.

* + - 1. Calculation of Standard Deviation

The Standard Deviation calculation, a computation used to quantify the amount of variation

/ dispersion among other HSV values, must be performed in order to determine the maximum and lowest of the HSV values. Additionally, three-sigma rules must be used in the calculation to produce an accurate range of values, increasing the likelihood that the collected values will fall inside the three-sigma interval.

#### Recognition Phase

* + - 1. Webcam and variable initialization:

Early in the recognition phase, the software will initialize the necessary variables that will be used to store various frame kinds and value ranges, each of which will be used to complete a specific task. Additionally, during this phase, the program gathers the calibrated HSV values and settings that will be applied later during the Binary Threshold transitions.

* + - 1. Real Time Image Acquisition:

Using (cv: Video Capture cap (0)), the real-time picture is taken using the camera. Each image is placed into a frame variable (cv: Mat), which is then flipped and compressed to a manageable size to lessen process load.

* + - 1. Frame noise Filtering:

The noise in the collected frames will be reduced using Gaussian filters, just like it was done during the calibration step.

* + - 1. HSV frame transition:

It is necessary to change the captured frame's format from BGR to HSV. using cvtColor (src, dst, CV\_BGR2HSV) for example.

* + - 1. Binary Threshold Transition:

A range check will be performed on the converted HSV frame to see if the HSV values fall within the range of the HSV variables collected during the calibration step. The frame will be converted into the binary system Threshold as a consequence of the range check, with a portion of the frame being set to 255 (1 bit) if the frame falls within the given HSV values and to 0 (0 bit) otherwise.

* + - 1. Binary Threshold Morphological Transmission:

After obtaining the binary threshold, the frame will go through a procedure termed morphological conversion, which is a structural operation to get rid of any foreground gaps and tiny objects. Erosion and Dilation are two morphological operators that make up the transformation. In order to remove minor sounds, the Erosion operator works by eroding the foreground object's edges and reducing the area of the binary limit. In terms of dilation, it is the reverse of erosion and raises the binary threshold area, allowing an item that has been eroded to regain its former shape.Color Combination Comparison.The software will determine the remainder of the number of objects by highlighting them as blobs after collecting the results from the morphological transformation process; this procedure

necessitates the use of the cvblob library, an OpenCV add-on. In order to identify how the mouse behaves according to the color configurations identified within the collected frames, the calculation's findings will then be sent for comparison.

* + - 1. Colors’ Coordinates Acquisition:

The program will display the general shape of each object that falls within the binary threshold where it will compute the shape's area and midpoint coordinates. The coordinates will be kept and utilized subsequently to execute different mouse operations based on the data gathered, either in setting cursor positions or in calculating the separation between each of the spots.

* + - 1. Execution of Mouse Action:

Based on the color combinations found in the processed frame, the application will carry out mouse operations. The mouse movements will be carried out in accordance with the coordinates that the software has supplied, and the application will keep acquiring and processing new real-time images up until the users depart it.

The color combinations that cause and carry out mouse actions are described as follows:

|  |  |
| --- | --- |
| **Mouse Function** | **Color Combination** |
| Moving | Initial color (e.g. Blue) |
| Left Click Up | (Un-pinch gesture) First Color ₊ Second Color (e.g. Blue₊ Green) |
| Left Click Down | (pinch gesture) First Color ₊ Second Color  (e.g. Blue ₊ Green) |
| Right Click | First Color ₊ Third Color (e.g. Blue ₊ Red) |

**Table 3.1:** The Overall Color Combination for Specific Mouse Function

## Application Layout

Users must choose from the options in the primary menu since each option leads to a distinct function of the program when the application first launches in a console window. Users will get an error notice and be sent back to the primary menu if they choose the wrong choice.

In order to get the best accuracy and performance possible during the recognition phase, the user can select and calibrate the preferred colors using the second option. In addition, the third option gives the user the ability to change the program's settings, such as the camera options, feedback window size, and other things.

When the first option is selected, the application will launch several processes, initialize the necessary variables, and then start showing several windows that show the binary limit and the HSV track-bars of individual colors, as well as a main window that shows the live-captured frame. Users can make minor adjustments to the HSV track-bars that are provided in order to increase the precision of color detection. Users must, however, have a rudimentary understanding of the HSV color model in order to set the track-bars correctly; otherwise, the recognition process as a whole could go awry. Each track- bar serves the following purposes:

|  |  |  |
| --- | --- | --- |
| **No.** | **Track-Bar Name** | **Purpose** |
| ⅰ | H1 | The upper boundaries of Hue plan |
| ii | H2 | The lower boundaries of Hue plan |
| iii | S1 | The upper boundaries of Saturation plan |
| iv | S2 | The lower boundaries of Saturation  plan |
| v | V1 | The upper boundaries of Value plan |
| vi | V2 | The lower boundaries of value plan |

**Table 3.2:** The Dynamic Track-bars Setup for Changing HSV Values

Two distinct regions, each with a red and black outline, make up the "Live" feedback window, which shows the real-time frame. The area that was highlighted in red denotes the dimension of the real display that users are utilizing, since the coordinates provided inside that area correspond to an accurate depiction on the actual monitor. The other two colors (for example, Green and Red) are only identifiable within the targeted zone, which is the area that was indicated in black.

## Problem and Challenges with Implementation

Several implementation problems arose as the application was being developed. The following lists the problems and difficulties that are likely to be experienced throughout the development phase:

##### The recorded frames contain short bursts of salt and pepper sounds.:

When the requisite HSV values in the recorded frame are too little, yet the frame was nonetheless subjected to a number of operations even though it was too small to be regarded as an input, salt and pepper sounds result. To solve this problem, it is necessary to first filter out

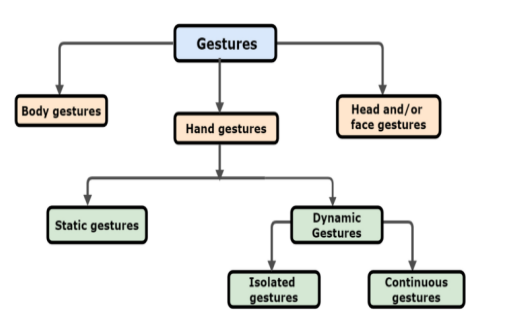
all of the undesirable HSV pixels present in the frame, including the excessively big and small pixel regions. With this approach, there will be far less chance of comparable pixels being interrupted.

##### Low-tier system performance deterioration brought on by heavy process load:

The program might be CPU-intensive for the majority of low-tier systems since it must go through a number of processing steps in order to filter, process, and perform mouse actions in real time. The length of time it takes the program to process a single frame will significantly rise if the size of the collected frames is too huge. By processing only, the crucial portions of the frames, the program may avoid this problem and cut down on redundant filtering, which could otherwise cause the application to lag.

* **The challenges of calibrating the frame’s brightness and constant to obtain the necessary HSV values:** In order to obtain the necessary color pixels, brightness and contrast intensity are crucial factors. All of the needed HSV values for executing the various mouse capabilities must be fulfilled in order for the program to use all of the mouse functions offered, which also means the required overall HSV values for brightness and contrast must be met. However, the calibration process can be laborious since, unless the initial HSV values were altered to show differently, a given intensity may only fulfill a portion of the necessary HSV values. To solve this problem, the program must first launch into a calibration phase that lets users select the color pixels they want before delivering them for the main stage.

The designed system needs to be easily scalable and adaptable to all categories of devices and environments. The input commands through users can be controlled virtually through Dynamic or Sta tic hand gestures. Hand gestures can be classified into two ways such as: static gestures (simple gestures) and dynamic gestures (trajectory-based gestures).



**Figure 1.4:** implementation process

Gesture Acquisition We can acquire the data in raw form for hand gesture

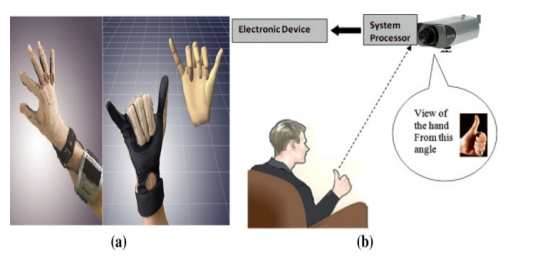
recognition system by using below two methods:

1. Using Sensors

2. Using Vision based cameras

**First approach** uses various types of sensors or other equipments which are attached physically to user’s hand

or arm such that the position, motion and trajectories of the hand and fingers can be captured.



**Figure 2:**first approach sensor

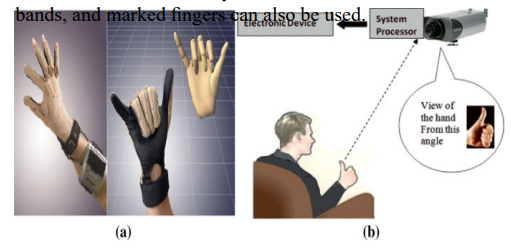
**Various Sensor dependent methods are:**

1. Glove-based: this method detects the position, acceleration, bending of the hand &fingers and degree of freedom of the hand. flex sensors, gyroscope, accelerometer are the examples of glove-based sensors
2. EMG (Electromyography): it measures the electrical pulses through muscles of human and interpret the captured

signal to detect the movements of finger.

1. Wi-Fi and radar based: these methods use the radio waves, broad-beam radar or spectrogram to detect any change in the strength of signal

**Vision-based approaches** are based on acquiring he images/ videos of person’s hand gestures using various cameras. we can use Single cameras,Stereo-camera and multiple camera-based system, light coding technique such as projection of light to get the 3d view of an object using PrimeSense, Microsoft Kinect, Creative Senz-3D or Leap Motion Sensor



**Figure 2.1** second approach sensor

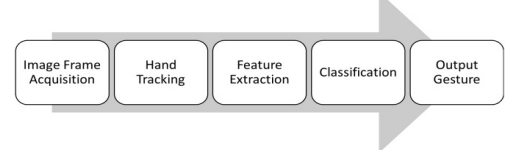
The system captures the images of various movements of an individual’s hand using cameras, which will be and

Then recognized by AI algorithm. The recognition model is first trained with the collection of hand gestures images to

understand recognize different gestures.

After recognition, they are converted into a corresponding movement of mouse, which can be performed on the

computer screen.



It is a priority that gestures should be predicted correctly for meaningful and efficient communication.various

techniques, classifiers, and algorithms are available to recognize gestures in a better way and we have to

choose the best approach so that it enhance human–computer interactions.

**3.4 OBJECTIVE:**

To develop and implement a system through which people can communicate with the computer system solely

using various hand gestures is our objective. It provides an alternative to physical PC mouse and allows an

individual to perform various mouse functions. Also, we can include more features in this application to get

better functionalities from the system which makes it scalable and more efficient than physical mouse.

**3.5** **ALGORITHMS AND TOOLS USED:**

To detect hand and finger tips in the input image or video, Media Pipe and Open cvare used, which are one of

the effective open-source library to perform the CV related tasks. Also, machine learning algorithm is used for

recognizing the hand gestures and capture their movements.

1. **Media Pipe Framework** (open-source, developed by Google): It enables the users to develop real-time

computer vision applications which supports cross-platform, it has preexisting tools and components (like object detection, pose estimation, hand tracking, facial recognition etc.)to process and analyze the input video and audio data.

The various tools offered by MediaPipe framework to train and deploy machine learning models can be executed in real-time environment on various h/w platforms (CPUs, GPUs,) and on specialized accelerators (Edge TPU by Google). Also, it allows the generated models to interact with other machine learning libraries (like TensorFlow and PyTorch)by providing interfaces, and it works with several programming languages, like C++, Python, and Java

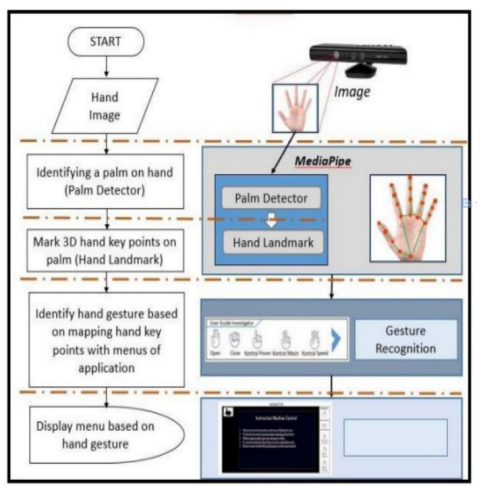
For tracking hand movements, Detection and characterization of a hand or palm(gesture recognition) in real time, the Media Pipe uses Single-shot detector model.

1. **OpenCV:** It is freely available computer vision and Machine Learning software library which helps

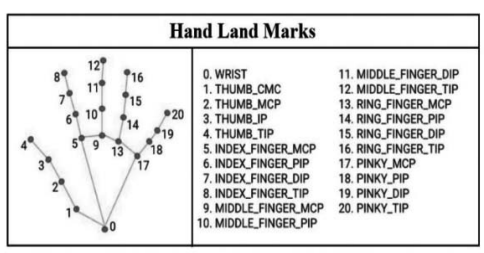
programmers to develop various computer vision applications like image/video Filtering, feature extraction, recognition of objects, tracking of objects. Basically, large variety of tools and techniques are provided by OpenCV to work with images and videos, which makes it a patent library for developing numerous computer vision based applications

**3.6 SYSTEM ARCHITECTURE:** To implement the proposed virtual mouse based on hand gestures, we have to follow the three important steps, which includes:

1. **Tracking of Hand Gestures:** A Blaze Palm model is applied to detect the palm on the full input image of the hand and will return the bounding box with oriented hand.
2. **Extracting the features of Hand regions:** Now, the bounded region of image defined by the palm detector is operated by landmark model of hand which outputs 3D hand key points.
3. **Classifying those extracted features:** In this step, gesture recognizer (or classifier)is applied on the previously calculated keypointswhich generates discrete set of gestures after classifying the configuration.

****

**Figure 3:**system architecture



**Figure 3.1 :**Hand recognition

**4.SOFTWARE ENVIRONMENT**

* 1. **Introduction to Python**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The Python 2 language, i.e., Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it.[32][33] With Python 2's end-of-life, only Python 3.5.x and later are supported. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open- source implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

**SYNTAX AND SEMANTICS**

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation.

Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

**INDENTION**

Main article: Python syntax and semantics § Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end

of the current block. Thus, the program's visual structure accurately represents the program's semantic

structure. This feature is sometimes termed the off-side rule, which some other languages share, but in

most languages, indentation doesn't have any semantic meaning.

**4.2 STATEMENTS AND CONTROL FLOW**

Python's statements include (among others):

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object.

Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound.

Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However, at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

* The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The while statement, which executes a block of code as long as its condition is true.
* The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The raise statement, used to raise a specified exception or re-raise a caught exception.
* The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
* The def statement, which defines a function or method.
* The with statement, from Python 2.5 released in September 2006, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behaviour and replaces a common try/finally idiom.
* The break statement, exits from the loop.
* The continue statement, skips this iteration and continues with the next item.
* The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
* The assert statement, used during debugging to check for conditions that ought to apply.
* The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import <module name> [as <alias>] or from <module name> import \* or from <module name> import <definition 1> [as <alias 1>],

<definition 2> [as <alias 2>],

The print statement was changed to the print () function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

**4.3 EXPRESSIONS**

Some Python expressions are similar to languages such as C and Java, while some are not: Addition, subtraction, and multiplication are the same, but the behaviour of division differs.

There are two types of divisions in Python. They are floor division (or integer division) // and floating point/division. Python also added the \*\* operator for exponentiation.

From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.

From Python 3.8, the syntax: =, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

In Python, == compares by value, versus Java, which compares numeri’s by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals () method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

Python uses the words and, or, not for its Boolean operators rather than the symbolic &&, ||, ! used in Java and C.

Python has a type of expression termed a list comprehension. Python 2.4 extended list comprehensions into a more general expression termed a generator expression.

Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

Conditional expressions in Python are written as x if c else y (different in order of operands from the c? x : y operator common to many other languages).

Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

Python has a "string format" operator %. These functions analogous to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2".

In Python 3 and 2.6+, this was supplemented by the format () method of the str class, e.g. "spam={0} eggs={1}". format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.

**Python has various kinds of string literals**

Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl- influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".

Triple-quoted strings, which begin and end with a series of three single or double quote marks.

They may span multiple lines and function like here documents in shells, Perl and Ruby.

Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

Python has array index and array slicing expressions on lists, denoted as a[key], a[start: stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

List comprehensions vs. for-loops Conditional expressions vs. if blocks

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python.

**4.4 METHODS**

Methods on objects are functions attached to the object's class; the syntax instance. method(argument) is, for normal methods and functions, syntactic sugar for Class. method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

**APPLICATIONS OF PYTHON**

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

**Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

**Easy-to-read** − Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain** − Python's source code is fairly easy-to-maintain.

**A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** − Python provides interfaces to all major commercial databases.

**GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable** − Python provides a better structure and support for large programs than shell scripting.

**Python OOPs Concepts**

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In [Python,](https://www.javatpoint.com/python-tutorial) we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

* Class
* Object
* Method
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

**Class**

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

**Syntax**

**class** ClassName:

<statement-1>

.

.

<statement-N>

**Object**

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute doc , which returns the docstring defined in the function source code.

When we define a class, it needs to create an object to allocate the memory. Consider the following example.

**Method**

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

**Inheritance**

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

it provides the re-usability of the code.

**Polymorphism**

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

**Encapsulation**

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

**Data Abstraction**

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

**Python Class and Objects**

We have already discussed in previous tutorial, a class is a virtual entity and can be seen as a blueprint of an object. The class came into existence when it instantiated. Let's understand it by an example.

Suppose a class is a prototype of a building. A building contains all the details about the floor, rooms, doors, windows, etc. we can make as many buildings as we want, based on these details. Hence, the building can be seen as a class, and we can create as many objects of this class.

On the other hand, the object is the instance of a class. The process of creating an object can be called instantiation.

In this section of the tutorial, we will discuss creating classes and objects in Python. We will also discuss how a class attribute is accessed by using the object.

**Creating classes in Python**

In Python, a class can be created by using the keyword class, followed by the class name. The syntax to create a class is given below.

Syntax

**class** ClassName:

#statement\_suite

In Python, we must notice that each class is associated with a documentation string which can be accessed by using **<class-name>. doc .** A class contains a statement suite including fields, constructor, function, etc. definition.

Consider the following example to create a class **Employee** which contains two fields as Employee id, and name.

The class also contains a function **display (),** which is used to display the information of the **Employee.**

Here, the **self** is used as a reference variable, which refers to the current class object. It is always the first argument in the function definition. However, using **self** is optional in the function call.

**The self-parameter**

The self-parameter refers to the current instance of the class and accesses the class variables. We can use anything instead of self, but it must be the first parameter of any function which belongs to the class.

**Creating an instance of the class**

A class needs to be instantiated if we want to use the class attributes in another class or method.

A class can be instantiated by calling the class using the class name.

The syntax to create the instance of the class is given below.

<object-name> = <class-name>(<arguments>)

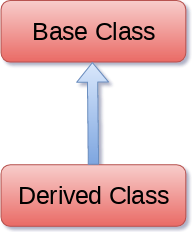
The following example creates the instance of the class Employee defined in the above example.

**Python Inheritance**

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail.

In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Consider the following syntax to inherit a base class into the derived class.



**Figure 3.2:** inheritance

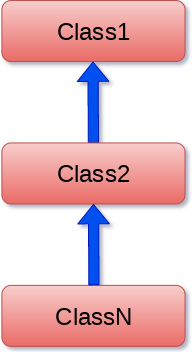
**Syntax**

**class** derived-**class**(base **class**):

<**class**-suite>

**Python Multi-Level inheritance**

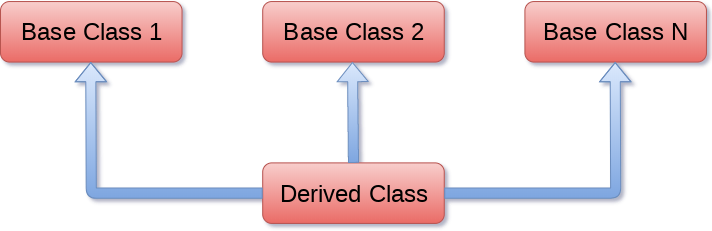
Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.



**Fig 3.3:** Singal inheritance

**Python Multiple inheritance**

Python provides us the flexibility to inherit multiple base classes in the child class.



**Fig3.4** multiple inheritance

**Method Overriding**

We can provide some specific implementation of the parent class method in our child class. When the parent class method is defined in the child class with some specific implementation, then the concept is called method overriding. We may need to perform method overriding in the scenario where the different definition of a parent class method is needed in the child class.

Data abstraction in python

Abstraction is an important aspect of object-oriented programming. In python, we can also perform data hiding by adding the double underscore ( ) as a prefix to the attribute which is hidden. After this, the attribute will not be visible outside of the class through the object.

hidden. After this, the attribute will not be visible outside of the class through the object.

**Abstraction in Python**

Abstraction is used to hide the internal functionality of the function from the users. The users only interact with the basic implementation of the function, but inner working is hidden. User is familiar with that **"what function does"** but they don't know **"how it does."**

In simple words, we all use the smartphone and very much familiar with its functions such as camera, voice-recorder, call-dialing, etc., but we don't know how these operations are happening in the background. Let's take another example - When we use the TV remote to increase the volume. We don't know how pressing a key increases the volume of the TV. We only know to press the "+" button to increase the volume.

That is exactly the abstraction that works in the [object-oriented concept](https://www.javatpoint.com/python-oops-concepts).

**Why Abstraction is Important?**

In Python, an abstraction is used to hide the irrelevant data/class in order to reduce the complexity. It also enhances the application efficiency. Next, we will learn how we can achieve abstraction using the python program.

**Syntax**

from abc **import** ABC

**class** ClassName(ABC):

We import the ABC class from the **abc** module.

**Abstract Base Classes**

An abstract base class is the common application program of the interface for a set of subclasses. It can be used by the third-party, which will provide the implementations such as with plugins. It is also beneficial when we work with the large code-base hard to remember all the classes.

**Working of the Abstract Classes**

Unlike the other high-level language, Python doesn't provide the abstract class itself. We need to import the abc module, which provides the base for defining Abstract Base classes (ABC). TheABC works by decorating methods of the base class as abstract. It registers concrete classes as the

implementation of the abstract base. We use the *@abstractmethod* decorator to define an abstract method or if we don't provide the definition to the method, it automatically becomes the abstract method. Let's understand the following example.

* 1. **INSTALLATION OF PYTHON**

Installing and using Python on Windows 10 is very simple. The installation procedure involves just three steps:

* Download the binaries
* Run the Executable installer
* Add Python to PATH environmental variables

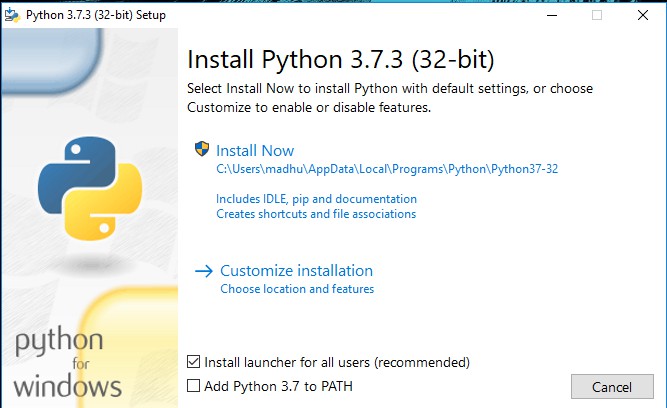
To install Python, you need to download the official Python executable installer. Next, you need to run this installer and complete the installation steps. Finally, you can configure the PATH variable to use python from the command line.

**Step 1**: Download the Python Installer binaries

* Open the official Python website in your web browser. Navigate to the Downloads tab for Windows.
* Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version. Click on the link to download Windows x86 executable installer if you are using a 32-bit installer.
* In case your Windows installation is a 64-bit system, then download Windows x86-64 executable installer.

**Step 2:** Run the Executable Installer

1. Once the installer is downloaded, run the Python installer.
2. Check the Install launcher for all users check box. Further, you may check the Add Python 3.7 to path check box to include the interpreter in the execution path.

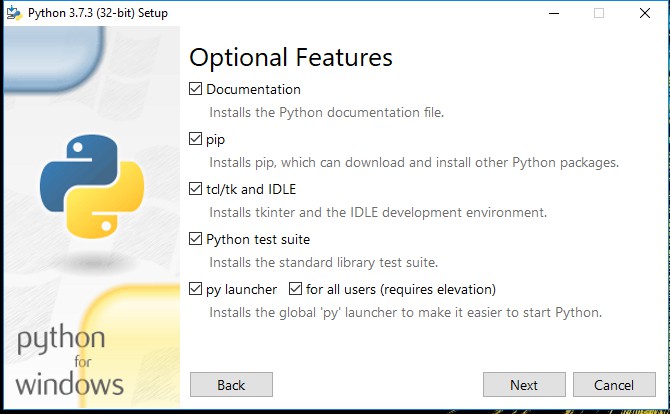


**Installation Python 3.7.3**

**1.Select Customize installation**.

Choose the optional features by checking the following check boxes:

1. Documentation
2. pip
3. tcl/tk and IDLE (to install tkinter and IDLE)
4. Python test suite (to install the standard library test suite of Python)
5. Install the global launcher for `.py` files. This makes it easier to start Python
6. Install for all users.



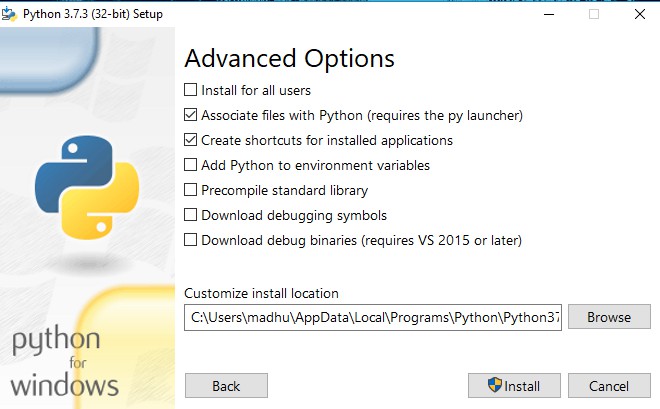
**Fig 3.5:** Optional Features

Click Next.

1. This takes you to Advanced Options available while installing Python. Here, select the Install for all users and Add Python to environment variables check boxes.

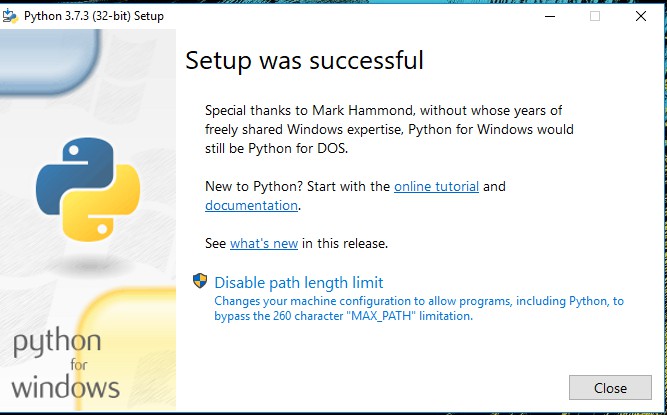
Optionally, you can select the Associate files with Python, Create shortcuts for installed applications and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step.

After selecting the Advanced options, click Install to start installation.



**Fig3.6:** Advanced Options

2.Once the installation is over, you will see a Python Setup Successful window.



**Fig :** Settings Setup

**Step 3:** Add Python to environmental variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the command line. In case you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step. Else, this step is done manually as follows.

In the Start menu, search for “advanced system settings”. Select “View advanced system settings”. In the “System Properties” window, click on the “Advanced” tab and then click on the “Environment Variables” button.

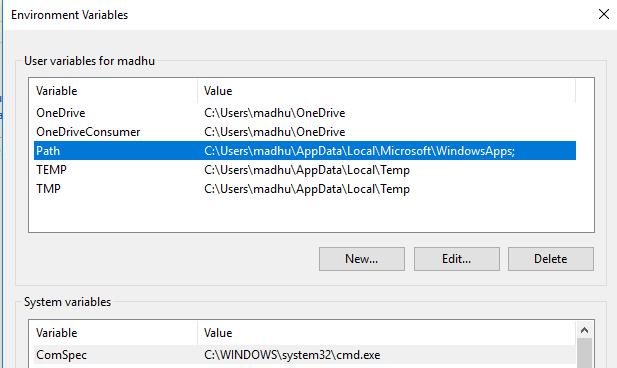
Locate the Python installation directory on your system. If you followed the steps exactly as above, python will be installed in below locations:

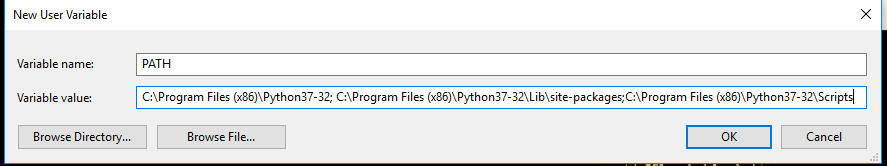
* C:\Program Files (x86)\Python37-32: for 32-bit installation
* C:\Program Files\Python37-32: for 64-bit installation

The folder name may be different from “Python37-32” if you installed a different version.

Look for a folder whose name starts with Python.

Append the following entries to PATH variable as shown below:

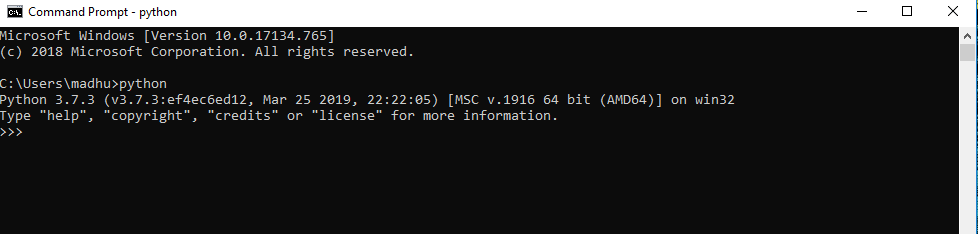




**Environment Settings**

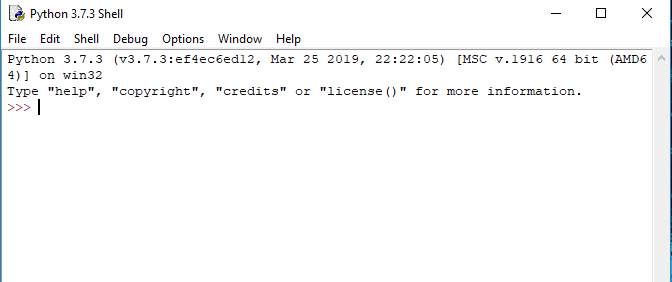
**Step 4:** Verify the Python Installation

You have now successfully installed Python 3.7.3 on Windows 10. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “python”. You can see that Python 3.7.3 is successfully installed.



**Fig.4:** Command Prompt

An alternate way to reach python is to search for “Python” in the start menu and clicking on IDLE (Python 3.7 64-bit). You can start coding in Python using the Integrated Development Environment(IDLE).



**Fig 4.1:** Python Shell Prompt

**USES**

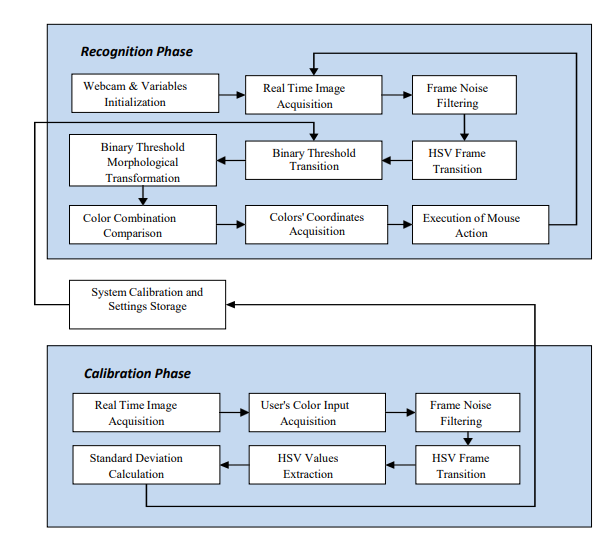
Since 2003, Python has consistently ranked in the top ten most popular programming

languages in the TIOBE Programming Community Index where, as of February 2020, it is the third most popular language (behind Java, and C). It was selected Programming Language of the Year in 2007, 2010, and 2018.

* An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".
* Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify and some smaller entities like ILM and ITA. The social news networking site Reddit is written entirely in Python.
* Python can serve as a scripting language for web applications, e.g., via mod\_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications.
* SQLAlchemy can be used as data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.
* Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.
* Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella.
* GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.
* Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.
* Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.
* Python is used extensively in the information security industry, including in exploit development.
* Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.
* Due to Python's user-friendly conventions and easy-to-understand language, it is commonly used as an intro language into computing sciences with students. This allows students to easily learn computing theories and concepts and then apply them to other programming languages.
* LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

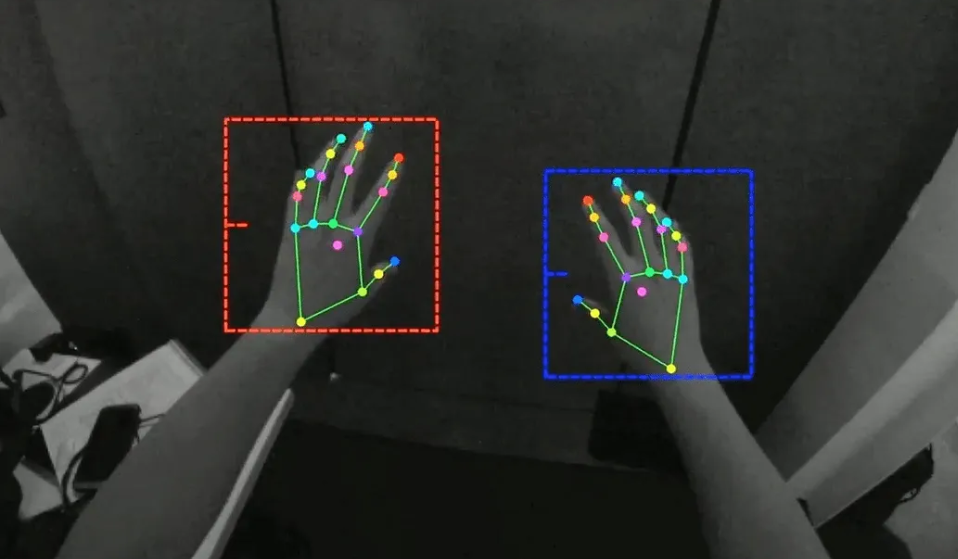
# 5.SYSTEM DESIGN

**5.1. System Architecture**

****

**Fig.4.1**:System Architecture

**5.2 Model Prediction**

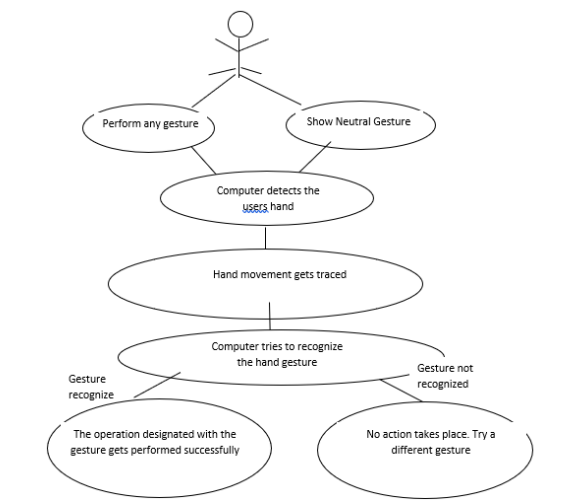
****

**Fig.4.2 :** Model Prediction

#### UML Diagrams

Unified Modeling Language is popular in the market because it is easy to understand. This is part of software engineering. Developer gets better idea about the system.

**5.3.1 Use Case Diagram**

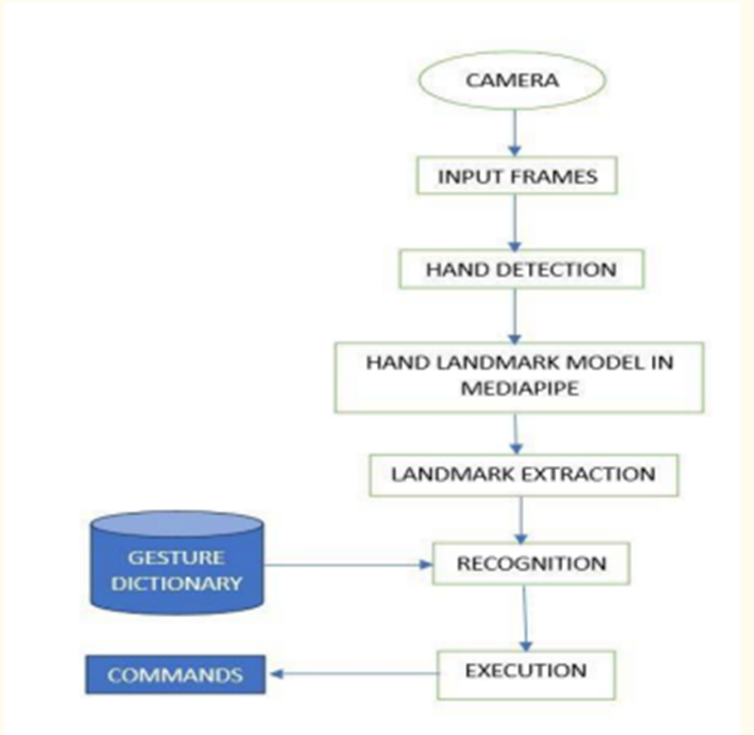
****

**Fig. 4.3:** Activity Diagram

**5.3.2 Data Flow Diagram**

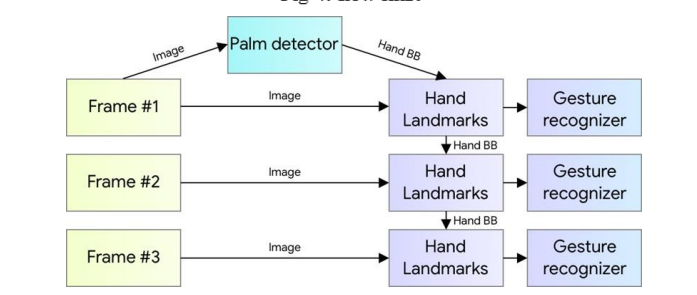


**Fig.4.4 :** Level-0 DFD



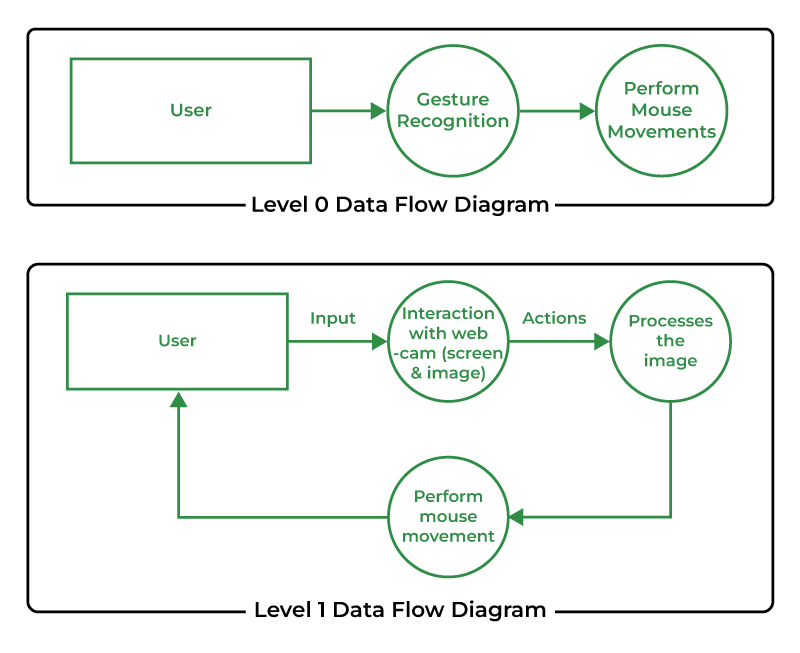
**Fig 4.5:** Level-1 DFD

**5.3.3 Sequence Diagram**

****

**Fig 4.6 :** Sequence Diagram

**5.3.4 State Diagram:**

****

**Fig .4.7:**state diagram

**6.IMPLEMENTATION**

## 6.1. Methodology

Thee various functions and conditions used in the system are explained in the flowchart of the real-time AI virtual mouse system in 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. 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.

## Hardware and Software Requirement

#### Hardware Requirement

The hardware required to run and create the Virtual Mouse program is described below:

**Computer desktop or laptop**

The machine such as a desktop or laptop will be used to run a visual program that will display what the camera captured. To promote mobility, a notebook, which is a tiny, lightweight, and affordable laptop computer, is offered.

System will be using

Processor : Core2Dual

Main Memory : 4GB RAM

Hard Disk : 320GB

Display : 14” Monitor

**Webcam**

The use of a webcam for image processing allows the application to process images and determine the positions of individual pixels.

**6.2.2 Software Requirement**

The following describe the software needed in order to develop the Virtual Mouse application

* Python Language

With the help of the Microsoft Visual Studio integrated development environment (IDE), which is used to create computer programs, the Virtual Mouse application will be coded using the python language. A python library offers many operators, including those for comparisons, logical operations, indirection, bit manipulation, and basic arithmetic.

* Open CV Library

Additionally, OpenCV was used in the development of this software. A collection of programming functions for real-time computer vision is called OpenCV (Open Source Computer Version). OpenCV has a tool that can read picture pixel values and can also make eye movement and blink recognition in real time.

Software will be using

OS : Window 10 Ultimate 64-bit

Language : Python

Tool Used : Open CV and Media Pipe

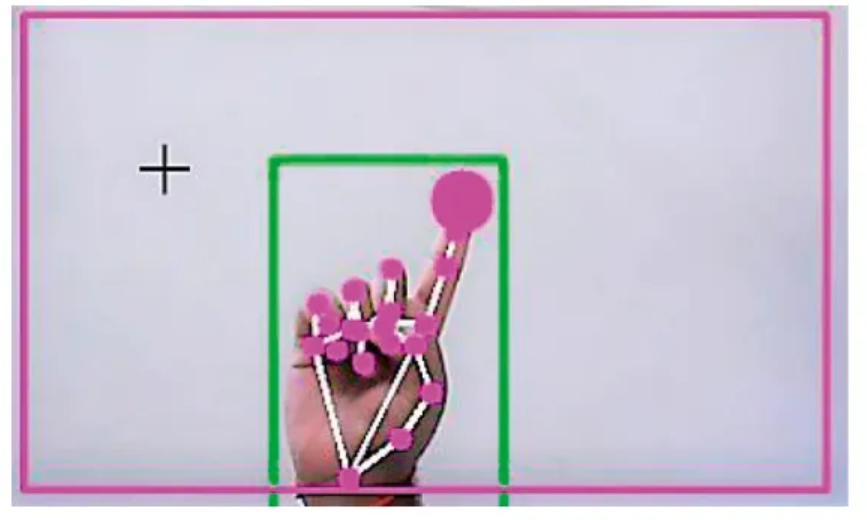
## 6.3Verification Plan

The Virtual Mouse color recognition must be capable of accurately and consistently identifying the majority of colors given by users while having no performance impact on other activities. The recognition outcomes may alter anytime the quality of the obtained frames has changed due to a number of environmental conditions, including surrounds, light, and weather. The following outlines the circumstances that might lead to erroneous detection or any other issue during the recognition phase.

1. The real-time photographs are captured in either a dark or light setting.
2. The real-time photos are captured on a backdrop with color conflicts.

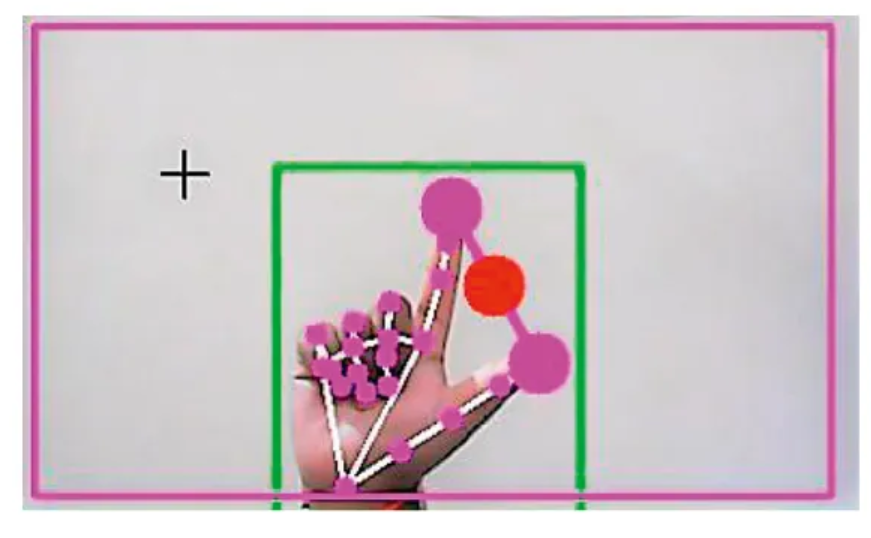
b) Users can engage with the software up close or from a distance.

1. Either a clockwise or counter clockwise rotation is applied to the real-time photographs.
2. the proposed system performs the functions of physical mouse via hand gestures with the help of computer vision techniques which is a alternative way of existing wired and wireless models.
3. Some important hand gestures for performing various mouse functions are :-
4. **Gesture 1: to move the cursor around the computer screen:**
5. (in this case, the mouse cursor will move around the computer screen if the index finger having tip Id-1 is up or the index finger having tip Id -1 as well as the middle finger with tip id-2 are up)

****

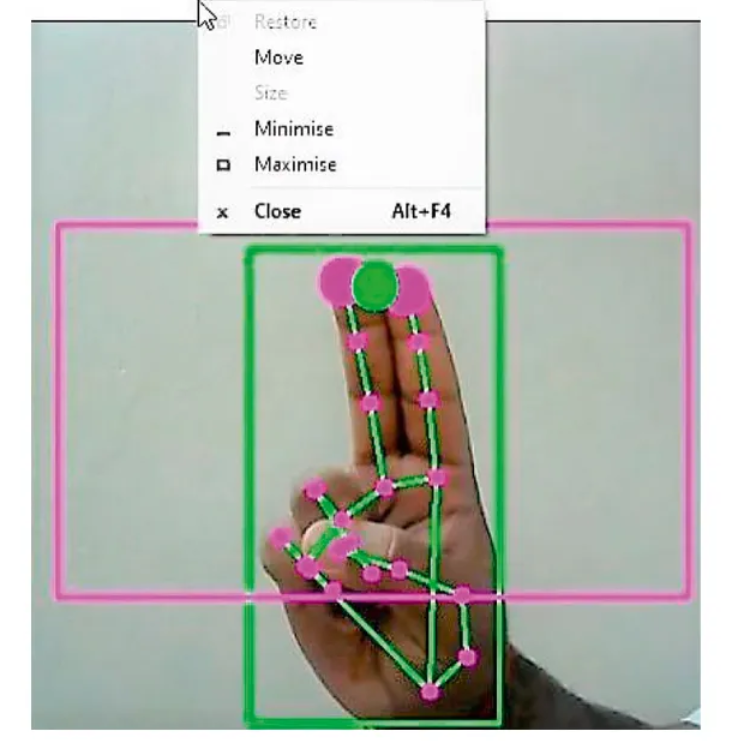
**Fig 4.8:** Cursor seceen

**Gesture 2: for performing the Left Click:** (To perform Left click mouse function on thecomputer screen, the index finger having tip Id-1 as well as the thumb having tip Id-0 should be up and the distance between these two should be less than 30pxcomputer screen)

****

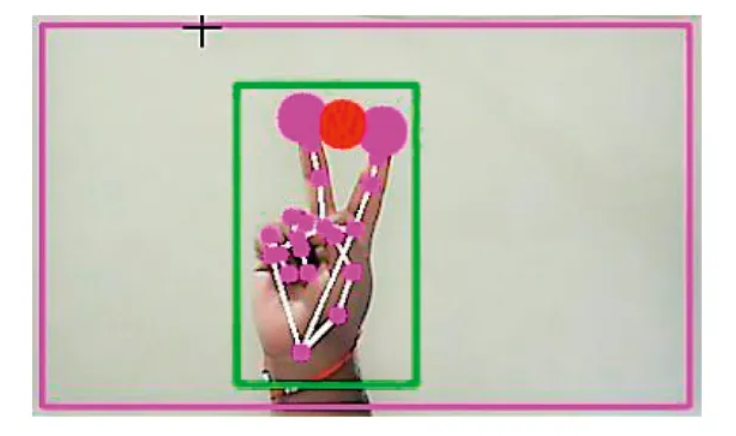
**Fig 4.9:** left click

**Gesture 3: for performing the Right Click :** (To perform Right click mouse function on the computer screen, the index finger having tip Id-1 as well as the middle finger having tip Id-2 should be up and the distance between these two fingers should be less than 40 px)

****

**Fig 4.10:** right click

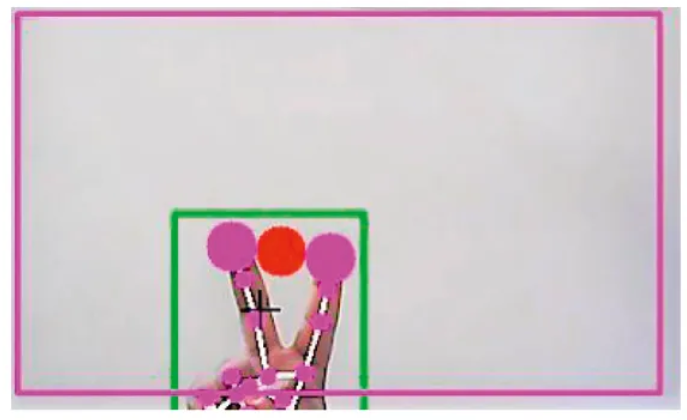
**Gesture 4:to perform scrolling function on computer screen in upward direction:** (To perform scrolling function in upward direction on the computer screen, the index finger having tip Id-1 as well as the middle finger having tip Id-2 should be up and the distance between these two fingers should be greater than 40 px to move up the page)



**Fig 4.11:** scrolling

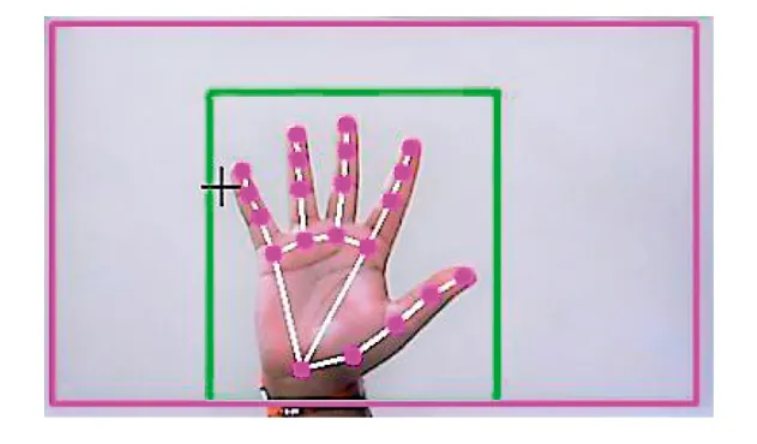
#### 

#### Gesture 5: to perform scrolling function on computer screen in downward direction: (To perform scrolling function in downward direction on the computer screen, the index finger having tip Id-1 as well as the middle finger having tip Id-2 should be up and the distance between these two fingers should be greater than 40 px to move up the pageand theyhave to be moved down the page together)

****

**Fig 4.12:** cursor move

**Gesture 6: Neutral Gesture- for performing no action:** (If all five fingers with tip Id- 0, 1, 2, 3, and 4are up, no function is performed on the screen related to any mouse events)

****

**Fig 4.13 hand mov**

**6.2. 1CODING**

import numpy as np

import cv2

import cv2.aruco as aruco

import os

import glob

import math

import pyautogui

import time

class Marker:

def \_\_init\_\_(self, dict\_type = aruco.DICT\_4X4\_50, thresh\_constant = 1):

self.aruco\_dict = aruco.Dictionary\_get(dict\_type)

self.parameters = aruco.DetectorParameters\_create()

self.parameters.adaptiveThreshConstant = thresh\_constant

self.corners = None # corners of Marker

self.marker\_x2y = 1 # width:height ratio

self.mtx, self.dist = Marker.calibrate()

def calibrate():

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

objp = np.zeros((6\*7,3), np.float32)

objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)

objpoints = [] # 3d point in real world space

imgpoints = [] # 2d points in image plane.

path = os.path.dirname(os.path.abspath(\_\_file\_\_))

p1 = path + r'\calib\_images\checkerboard\\*.jpg'

images = glob.glob(p1)

for fname in images:

img = cv2.imread(fname)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

ret, corners = cv2.findChessboardCorners(gray, (7,6),None)

if ret == True:

objpoints.append(objp)

corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)

imgpoints.append(corners2)

img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)

#mtx = [[534.34144579,0.0,339.15527836],[0.0,534.68425882,233.84359493],[0.0,0.0,1.0]]

#dist = [[-2.88320983e-01, 5.41079685e-02, 1.73501622e-03, -2.61333895e-04, 2.04110465e-01]]

return mtx, dist

def detect(self, frame):

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

self.corners, ids, rejectedImgPoints = aruco.detectMarkers(gray\_frame, self.aruco\_dict, parameters = self.parameters)

if np.all(ids != None):

rvec, tvec ,\_ = aruco.estimatePoseSingleMarkers(self.corners, 0.05, self.mtx, self.dist)

else:

self.corners = None

def is\_detected(self):

if self.corners:

return True

return False

def draw\_marker(self, frame):

aruco.drawDetectedMarkers(frame, self.corners)

def ecu\_dis(p1, p2):

dist = np.sqrt((p1[0]-p2[0])\*\*2 + (p1[1]-p2[1])\*\*2)

return dist

def find\_HSV(samples):

try:

color = np.uint8([ samples ])

except:

color = np.uint8([ [[105,105,50]] ])

hsv\_color = cv2.cvtColor(color,cv2.COLOR\_RGB2HSV)

#print( hsv\_color )

return hsv\_color

def draw\_box(frame, points, color=(0,255,127)):

if points:

frame = cv2.line(frame, points[0], points[1], color, thickness=2, lineType=8) #top

frame = cv2.line(frame, points[1], points[2], color, thickness=2, lineType=8) #right

frame = cv2.line(frame, points[2], points[3], color, thickness=2, lineType=8) #bottom

frame = cv2.line(frame, points[3], points[0], color, thickness=2, lineType=8) #left

def in\_cam(val, type\_):

if type\_ == 'x':

if val<0:

return 0

if val>GestureController.cam\_width:

return GestureController.cam\_width

elif type\_ == 'y':

if val<0:

return 0

if val>GestureController.cam\_height:

return GestureController.cam\_height

return val

class ROI:

def \_\_init\_\_(self, roi\_alpha1=1.5, roi\_alpha2=1.5, roi\_beta=2.5, hsv\_alpha = 0.3, hsv\_beta = 0.5, hsv\_lift\_up = 0.3):

self.roi\_alpha1 = roi\_alpha1

self.roi\_alpha2 = roi\_alpha2

self.roi\_beta = roi\_beta

self.roi\_corners = None

self.hsv\_alpha = hsv\_alpha

self.hsv\_beta = hsv\_beta

self.hsv\_lift\_up = hsv\_lift\_up

self.hsv\_corners = None

self.marker\_top = None

self.glove\_hsv = None

def findROI(self, frame, marker):

rec\_coor = marker.corners[0][0]

c1 = (int(rec\_coor[0][0]),int(rec\_coor[0][1]))

c2 = (int(rec\_coor[1][0]),int(rec\_coor[1][1]))

c3 = (int(rec\_coor[2][0]),int(rec\_coor[2][1]))

c4 = (int(rec\_coor[3][0]),int(rec\_coor[3][1]))

try:

marker.marker\_x2y = np.sqrt((c1[0]-c2[0])\*\*2 + (c1[1]-c2[1])\*\*2) / np.sqrt((c3[0]-c2[0])\*\*2 + (c3[1]-c2[1])\*\*2)

except:

marker.marker\_x2y = 999.0

#mid-point of top line of Marker

cx = (c1[0] + c2[0])/2

cy = (c1[1] + c2[1])/2

self.marker\_top = [cx, cy]

l = np.absolute(ecu\_dis(c1,c4))

try:

slope\_12 = (c1[1]-c2[1])/(c1[0]-c2[0])

except:

slope\_12 = (c1[1]-c2[1])\*999.0 + 0.1

try:

slope\_14 = -1 / slope\_12

except:

slope\_14 = -999.0

if slope\_14 < 0:

sign = 1

else:

sign = -1

bot\_rx = int(cx + self.roi\_alpha2 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ry = int(cy + self.roi\_alpha2 \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_lx = int(cx - self.roi\_alpha1 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ly = int(cy - self.roi\_alpha1 \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

top\_lx = int(bot\_lx + sign \* self.roi\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_ly = int(bot\_ly + sign \* self.roi\_beta \* slope\_14 \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_rx = int(bot\_rx + sign \* self.roi\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_ry = int(bot\_ry + sign \* self.roi\_beta \* slope\_14 \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

bot\_lx = in\_cam(bot\_lx, 'x')

bot\_ly = in\_cam(bot\_ly, 'y')

bot\_rx = in\_cam(bot\_rx, 'x')

bot\_ry = in\_cam(bot\_ry, 'y')

top\_lx = in\_cam(top\_lx, 'x')

top\_ly = in\_cam(top\_ly, 'y')

top\_rx = in\_cam(top\_rx, 'x')

top\_ry = in\_cam(top\_ry, 'y')

self.roi\_corners = [(bot\_lx,bot\_ly), (bot\_rx,bot\_ry), (top\_rx,top\_ry), (top\_lx,top\_ly)]

def find\_glove\_hsv(self, frame, marker):

rec\_coor = marker.corners[0][0]

c1 = (int(rec\_coor[0][0]),int(rec\_coor[0][1]))

c2 = (int(rec\_coor[1][0]),int(rec\_coor[1][1]))

c3 = (int(rec\_coor[2][0]),int(rec\_coor[2][1]))

c4 = (int(rec\_coor[3][0]),int(rec\_coor[3][1]))

l = np.absolute(ecu\_dis(c1,c4))

try:

slope\_12 = (c1[1]-c2[1])/(c1[0]-c2[0])

except:

slope\_12 = (c1[1]-c2[1])\*999.0 + 0.1

try:

slope\_14 = -1 / slope\_12

except:

slope\_14 = -999.0

if slope\_14 < 0:

sign = 1

else:

sign = -1

bot\_rx = int(self.marker\_top[0] + self.hsv\_alpha \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ry = int(self.marker\_top[1] - self.hsv\_lift\_up\*l + self.hsv\_alpha \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_lx = int(self.marker\_top[0] - self.hsv\_alpha \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ly = int(self.marker\_top[1] - self.hsv\_lift\_up\*l - self.hsv\_alpha \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

top\_lx = int(bot\_lx + sign \* self.hsv\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_ly = int(bot\_ly + sign \* self.hsv\_beta \* slope\_14 \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_rx = int(bot\_rx + sign \* self.hsv\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

top\_ry = int(bot\_ry + sign \* self.hsv\_beta \* slope\_14 \* l \* np.sqrt(1/(1+slope\_14\*\*2)))

region = frame[top\_ry:bot\_ry , top\_lx:bot\_rx]

b, g, r = np.mean(region, axis=(0, 1))

self.hsv\_glove = find\_HSV([[r,g,b]])

self.hsv\_corners = [(bot\_lx,bot\_ly), (bot\_rx,bot\_ry), (top\_rx,top\_ry), (top\_lx,top\_ly)]

def cropROI(self, frame):

pts = np.array(self.roi\_corners)

## (1) Crop the bounding rect

rect = cv2.boundingRect(pts)

x,y,w,h = rect

croped = frame[y:y+h, x:x+w].copy()

## (2) make mask

pts = pts - pts.min(axis=0)

mask = np.zeros(croped.shape[:2], np.uint8)

cv2.drawContours(mask, [pts], -1, (255, 255, 255), -1, cv2.LINE\_AA)

## (3) do bit-op

dst = cv2.bitwise\_and(croped, croped, mask=mask)

## (4) add the white background

bg = np.ones\_like(croped, np.uint8)\*255

cv2.bitwise\_not(bg,bg, mask=mask)

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

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

hsv = cv2.cvtColor(dst, cv2.COLOR\_BGR2HSV)

lower\_range = np.array([self.hsv\_glove[0][0][0]//1-5,50,50])

upper\_range = np.array([self.hsv\_glove[0][0][0]//1+5,255,255])

mask = cv2.inRange(hsv, lower\_range, upper\_range)

#mask = cv2.dilate(mask,kernelOpen,iterations = 1)

Opening =cv2.morphologyEx(mask,cv2.MORPH\_OPEN,kernelOpen)

Closing =cv2.morphologyEx(Opening,cv2.MORPH\_CLOSE,kernelClose)

FinalMask = Closing

return FinalMask

class Glove:

def \_\_init\_\_(self):

self.fingers = 0

self.arearatio = 0

self.gesture = 0

def find\_fingers(self, FinalMask):

conts,h=cv2.findContours(FinalMask,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

hull = [cv2.convexHull(c) for c in conts]

try:

cnt = max(conts, key = lambda x: cv2.contourArea(x))

#approx the contour a little

epsilon = 0.0005\*cv2.arcLength(cnt,True)

approx= cv2.approxPolyDP(cnt,epsilon,True)

#make convex hull around hand

hull = cv2.convexHull(cnt)

#define area of hull and area of hand

areahull = cv2.contourArea(hull)

areacnt = cv2.contourArea(cnt)

#find the percentage of area not covered by hand in convex hull

self.arearatio=((areahull-areacnt)/areacnt)\*100

#find the defects in convex hull with respect to hand

hull = cv2.convexHull(approx, returnPoints=False)

defects = cv2.convexityDefects(approx, hull)

except:

print("No Contours found in FinalMask")

# l = no. of defects

l=0

try:

#code for finding no. of defects due to fingers

for i in range(defects.shape[0]):

s,e,f,d = defects[i,0]

start = tuple(approx[s][0])

end = tuple(approx[e][0])

far = tuple(approx[f][0])

# find length of all sides of triangle

a = math.sqrt((end[0] - start[0])\*\*2 + (end[1] - start[1])\*\*2)

b = math. sqrt((far[0] - start [0])\*\*2 + (far[1] - start[1])\*\*2)

c = math.sqrt((end[0] - far[0])\*\*2 + (end[1] - far[1])\*\*2)

s = (a+b+c)/2

ar = math.sqrt(s\*(s-a)\*(s-b)\*(s-c))

#distance between point and convex hull

d=(2\*ar)/a

# apply cosine rule here

angle = math.acos((b\*\*2 + c\*\*2 - a\*\*2)/(2\*b\*c)) \* 57

# ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)

if angle <= 90 and d>30:

l += 1

#cv2.circle(frame, far, 3, [255,255,255], -1)

#draw lines around hand

cv2.line(FinalMask,start, end, [255,255,255], 2)

l+=1

except:

l = 0

print("No Defects found in mask")

self.fingers = l

def find\_gesture(self, frame):

font = cv2.FONT\_HERSHEY\_SIMPLEX

self.gesture = 0

if self.fingers==1:

#cv2.putText(frame, str(int(arearatio)), (10,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

if self.arearatio<15:

cv2.putText(frame,'0',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 0

elif self.arearatio<25:

cv2.putText(frame,'2 fingers',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 2

else:

cv2.putText(frame,'1 finger',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 1

elif self.fingers==2:

cv2.putText(frame,'2',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 3

'''

elif self.fingers==3:

#cv2.putText(frame,'3',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

elif self.fingers==4:

#cv2.putText(frame,'4',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

elif self.fingers==5:

#cv2.putText(frame,'5',(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

else :

# cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

'''

class Tracker:

def \_\_init\_\_(self):

self.tracker\_started = False

self.tracker = None

self.start\_time = 0.0

self.now\_time = 0.0

self.tracker\_bbox = None

def corners\_to\_tracker(self, corners):

csrt\_minX = int( min( [corners[0][0][0][0], corners[0][0][1][0], corners[0][0][2][0], corners[0][0][3][0]] ))

csrt\_maxX = int( max( [corners[0][0][0][0], corners[0][0][1][0], corners[0][0][2][0], corners[0][0][3][0]] ))

csrt\_minY = int( min( [corners[0][0][0][1], corners[0][0][1][1], corners[0][0][2][1], corners[0][0][3][1]] ))

csrt\_maxY = int( max( [corners[0][0][0][1], corners[0][0][1][1], corners[0][0][2][1], corners[0][0][3][1]] ))

self.tracker\_bbox = [csrt\_minX, csrt\_minY, csrt\_maxX-csrt\_minX, csrt\_maxY-csrt\_minY]

def tracker\_to\_corner(self, final\_bbox):

if self.tracker\_bbox == None:

return None

final\_bbox = [[[1,2],[3,4],[5,6],[7,8]]]

final\_bbox[0][0] = [self.tracker\_bbox[0],self.tracker\_bbox[1]]

final\_bbox[0][1] = [self.tracker\_bbox[0]+ self.tracker\_bbox[2],self.tracker\_bbox[1]]

final\_bbox[0][2] = [self.tracker\_bbox[0]+ self.tracker\_bbox[2],self.tracker\_bbox[1] + self.tracker\_bbox[3]]

final\_bbox[0][3] = [self.tracker\_bbox[0],self.tracker\_bbox[1] +self.tracker\_bbox[3]]

return [np.array(final\_bbox, dtype = 'f')]

def CSRT\_tracker(self, frame):

if self.tracker\_bbox == None and self.tracker\_started == False:

return

if self.tracker\_started == False:

if self.tracker == None:

self.tracker = cv2.TrackerCSRT\_create()

if self.tracker\_bbox != None:

try:

self.start\_time = time.time()

ok = self.tracker.init(frame, self.tracker\_bbox)

self.tracker\_started = True

except:

print("tracker.init failed")

try:

ok, self.tracker\_bbox = self.tracker.update(frame)

except:

ok = None

print("tracker.update failed")

self.now\_time = time.time()

if self.now\_time-self.start\_time >= 2.0 :

#cv2.putText(frame, "Please posture your hand correctly", (10,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1,(0,0,255),1)

cv2.putText(frame,'Posture your hand correctly',(10,10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0,0,255),1, c v2.LINE\_AA)

#print("tracking timeout")

self.tracker\_started = False

self.tracker\_bbox = None

return

if ok:

# Tracking success

p1 = (int(self.tracker\_bbox[0]), int(self.tracker\_bbox[1]))

p2 = (int(self.tracker\_bbox[0] + self.tracker\_bbox[2]), int(self.tracker\_bbox[1] + self.tracker\_bbox[3]))

cv2.rectangle(frame, p1, p2, (80, 255, 255), 2, 1)

else :

# Tracking failure

self.tracker\_started = False

cv2.putText(frame, "Tracking failure detected", (100,80), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75,(0,0,255),2)

print("Tracking failure detected")

#reintiallize code to tackle tracking failure

class Mouse:

def \_\_init\_\_(self):

self.tx\_old = 0

self.ty\_old = 0

self.trial = True

self.flag = 0

def move\_mouse(self,frame,position,gesture):

(sx,sy)=pyautogui.size()

(camx,camy) = (frame.shape[:2][0],frame.shape[:2][1])

(mx\_old,my\_old) = pyautogui.position()

Damping = 2 # Hyperparameter we will have to adjust

tx = position[0]

ty = position[1]

if self.trial:

self.trial, self.tx\_old, self.ty\_old = False, tx, ty

delta\_tx = tx - self.tx\_old

delta\_ty = ty - self.ty\_old

self.tx\_old,self.ty\_old = tx,ty

if (gesture == 3):

self.flag = 0

mx = mx\_old + (delta\_tx\*sx) // (camx\*Damping)

my = my\_old + (delta\_ty\*sy) // (camy\*Damping)

pyautogui.moveTo(mx,my, duration = 0.1)

elif(gesture == 0):

if self.flag == 0:

pyautogui.doubleClick()

self.flag = 1

elif(gesture == 1):

print('1 Finger Open')

class GestureController:

gc\_mode = 0

pyautogui.FAILSAFE = False

f\_start\_time = 0

f\_now\_time = 0

cam\_width = 0

cam\_height = 0

aru\_marker = Marker()

hand\_roi = ROI(2.5, 2.5, 6, 0.45, 0.6, 0.4)

glove = Glove()

csrt\_track = Tracker()

mouse = Mouse()

def \_\_init\_\_(self):

GestureController.cap = cv2.VideoCapture(0)

if GestureController.cap.isOpened():

GestureController.cam\_width = int( GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH) )

GestureController.cam\_height = int( GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT) )

else:

print("CANNOT OPEN CAMERA")

GestureController.gc\_mode = 1

GestureController.f\_start\_time = time.time()

GestureController.f\_now\_time = time.time()

def start(self):

while (True):

#mode checking

if not GestureController.gc\_mode:

print('Exiting Gesture Controller')

break

#fps control

fps = 30.0

GestureController.f\_start\_time = time.time()

while (GestureController.f\_now\_time-GestureController.f\_start\_time <= 1.0/fps):

GestureController.f\_now\_time = time.time()

#read camera

ret, frame = GestureController.cap.read()

frame = cv2.flip(frame, 1)

#detect Marker, find ROI, find glove HSV, get FinalMask on glove

GestureController.aru\_marker.detect(frame)

if GestureController.aru\_marker.is\_detected():

GestureController.csrt\_track.corners\_to\_tracker(GestureController.aru\_marker.corners)

GestureController.csrt\_track.CSRT\_tracker(frame)

else:

GestureController.csrt\_track.tracker\_bbox = None

GestureController.csrt\_track.CSRT\_tracker(frame)

GestureController.aru\_marker.corners = GestureController.csrt\_track.tracker\_to\_corner(GestureController.aru\_marker.corners)

if GestureController.aru\_marker.is\_detected():

GestureController.hand\_roi.findROI(frame, GestureController.aru\_marker)

GestureController.hand\_roi.find\_glove\_hsv(frame, GestureController.aru\_marker)

FinalMask = GestureController.hand\_roi.cropROI(frame)

GestureController.glove.find\_fingers(FinalMask)

GestureController.glove.find\_gesture(frame)

GestureController.mouse.move\_mouse(frame,GestureController.hand\_roi.marker\_top,GestureController.glove.gesture)

#draw call

if GestureController.aru\_marker.is\_detected():

GestureController.aru\_marker.draw\_marker(frame)

draw\_box(frame, GestureController.hand\_roi.roi\_corners, (255,0,0))

draw\_box(frame, GestureController.hand\_roi.hsv\_corners, (0,0,250))

cv2.imshow('FinalMask',FinalMask)

#display frame

cv2.imshow('frame',frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# When everything done, release the capture

GestureController.cap.release()

cv2.destroyAllWindows**()**

# 7.SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**7.1.1 Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application

.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

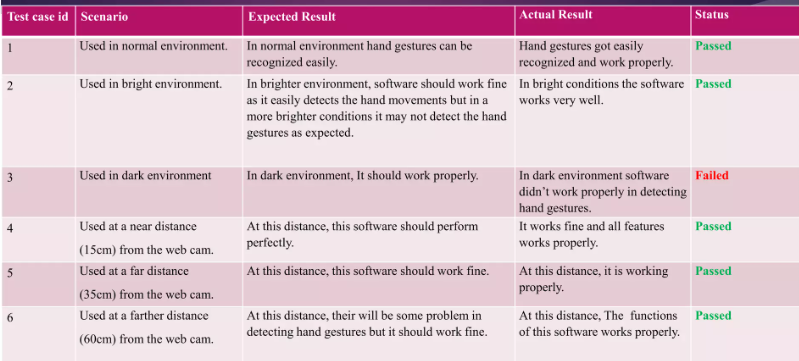
**7.1.2 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**7.1.3 Accepting Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. Acceptance testing for Data Synchronization:

* The Acknowledgements will be received by the auditor after the data is received by the cloud server
* The auditors audit operation is done only when there is a request from user
* The Status of data information on the cloud is viewed only by the cloud server



#### Risk Analysis

pedestrians and any other obstacles. It will also slow down on junctions, and overtake other vehicles, and execute emergency stops when other options are not available. On the hardware side, some issues are occurring due to insufficient compute power. The car is following lanes but some issues are occurring due to limitations of the hardware. The car will execute an emergency stop when the sensor detects any obstacles using the sensor. The backbone of this project is the LAN server because raspberry pi has insufficient computation power. Car drives automatically very well after training using a high-resolution dataset. Car stops whenever it finds a STOP sign or RED signal and if the ultrasonic sensor detects any object it will break and avoid collision.

**8.1Overview**

# 8.RESULTFINDING

Using a virtual mouse, the system's webcam is used for tracking hand gestures, and hand gesture recognition enables users to control the mouse with the help of hand gestures. Gesture recognition is accomplished using computer vision techniques. To collect data from a live video, OpenCV includes a package called video capture. This project can be applied in a variety of real-time applications, such as computer cursor control and smart televisions running Android, among others. The work is so straightforward that it minimizes the use of external hardware in such a way that finger motion in front of a camera will cause the necessary operation on the screen, even though there are tools like mouse and laser remotes for the same purpose. This project demonstrated a real-time hand tracking system that uses markers and a commodity computer with inexpensive cameras. When a calibrated pair of cameras is looking down at the hands with the palms facing downward, the system can specifically track the positions of the index finger and middle finger tips.

## Performance in Various Environments

Evaluations of virtual-mouse systems in the scientific community are still a little rudimentary. A cross- method evaluation is challenging since there is a dearth of available academic research and open-access datasets. The performance provided by the virtual mouse's fingertip detection will be covered first in this section. The performance will then be shown under various lighting, backdrop, and distance-tracking settings. The experimental findings for selecting the primary person to direct the mouse pointer and fingertip tracking with numerous persons are then shown. Finally, we evaluate how well our approach compares to earlier virtual-mouse research. The suggested method is based on detecting the amount of target colors (area of interest) that activates the mouse action based on the gesture generated. Initially, a picture is taken with the hand positioned toward the front of the sensor. The user then chooses the color cap that will be monitored during gesture creation in order to carry out different mouse actions. When taking a picture of the hand, the color of the cap must be chosen from a variety. The notion of enhancing interaction between humans and computers using machine vision is presented in the suggested all virtual mouse system. Cross-validation for the AI simulated mouse system can be problematic due to the low

amount of datasets available. Hand movements and fingertip identification have been verified in a variety of lighting situations, as well as at varied distances from a webcam for tracking and recognition of hand motions and finger tips.

In our project, we can perform various operations by using hand gesture-based cursor movement which depend on the distance between two fingers. Such that

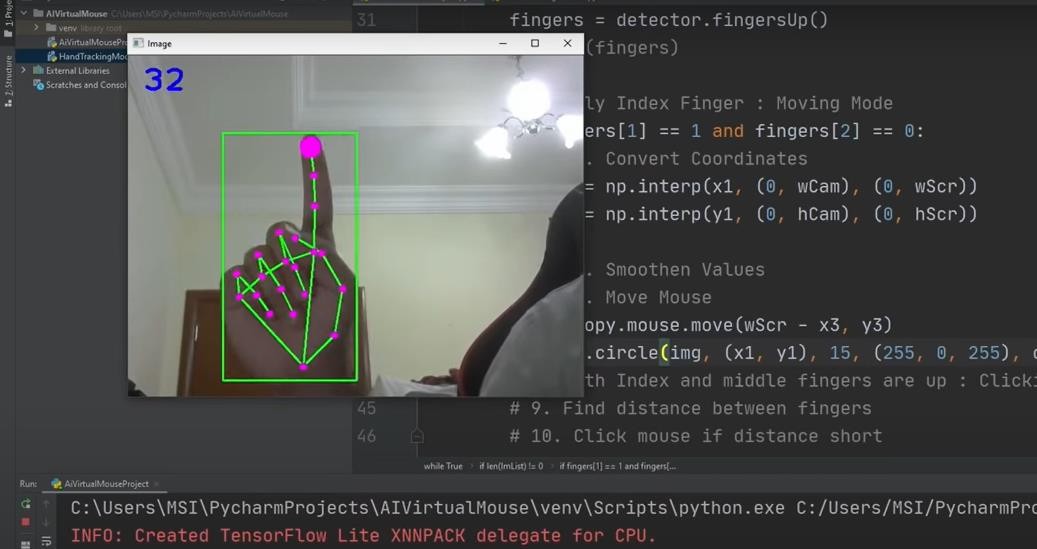
* + 1. we can fix our settings for brightness and sound quality based on the movement of the cursor.
    2. On the other hand, we can open a file or folder by using the hand movement-based cursor.

#### 8.1.3 Moving Environment

When the project runs, the cursor is already in moving mode. Based on the user's hand movements, the cursor will move to perform brightness increasing and decreasing operations. When the distance between two fingers is the greatest, the required one is required, and the cursor is said to be in moving mode. Artificial intelligence virtual mouse system camera.

The frames that have been recorded by a laptop's or PC's camera serve as the foundation for the suggested AI virtual mouse system. The webcam will start recording video when the video collection object is built using the programming language's machine vision package OpenCV. The webcam records and sends footage through the AI Virtual Mouse system.

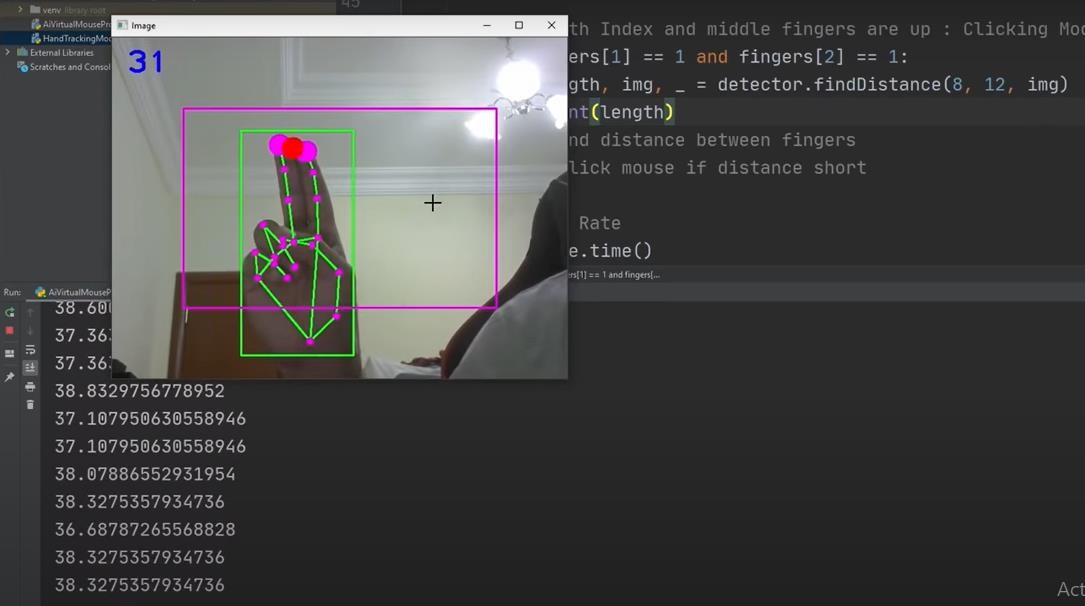
Accuracy: The percentage of accurately classified data samples over all the data is known as accuracy. Accuracy can be calculated by the following equation. [46]Accuracy = (TP+TN)/(TP+FP+TN+FN)



**Figure 5.1:** Mouse Cursor Moving around The Computer Window

#### Open File Environment

In our program, we already specify a minimum distance between two fingers. When the user fulfills this, it will mean the cursor is now in clickable mode. Now the user can open a file or folder. The computer is programmed to activate the left mouse button if the thumb finger with point ID= 0 and the pointer finger with tip ID =1 are both up and the gap among these two fingers is less than 30 px. The AI virtual mouse performed exceptionally well in terms of precision. The suggested model is unique in that it can execute most mouse tasks and mouse cursor movement using fingertip detection, and it is also useful in operating the PC in the virtual mode like a hardware mouse.



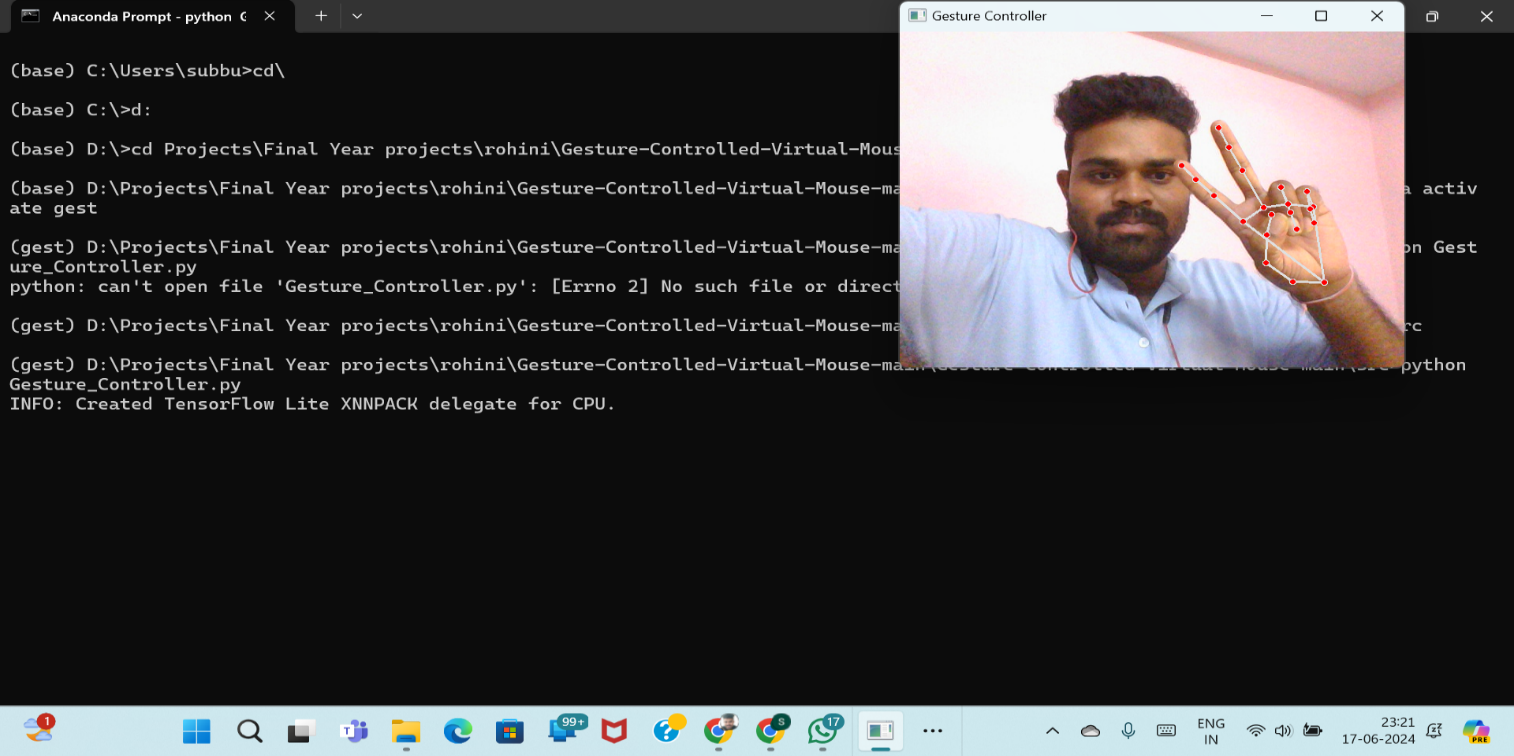
|  |  |  |  |
| --- | --- | --- | --- |
| **AI Virtual Mouse** | **Activity** | **Time** | **Accuracy** |
| Cursor Move | 292/150 | 97.67% |
| Right Click | 289/150 | 95.5% |
| Left Click | 293/150 | 97.67% |
| Double Click | 289/150 | 95.67% |

**Figure 5.2**: Button Click Performance

**Figure 5.2**: Button Click Performance



**Fig 5.3: cursor** move



We have tested our AI Virtual Mouse many times and got a good accuracy which is shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **AI Virtual Mouse** | **Hand Gesture** | **Time** | **Accuracy** |
| Cursor Move | 199/200 | 99.5% |
| Left Click | 195/200 | 97.5% |
| Right Click | 195/200 | 97.5% |
| Double Click | 190/200 | 95% |

Table 5.1: Average accuracy of the defined AI Virtual Mouse.

**Figure 5.3: Graphic** Chart of AI Virtual Mouse Accuracy

100

90

80

70

60

50

40

30

20

10

0

99.5

97.5

97.5

95

Cursor Move Right Click Left Click Double Click

#### 8.2.1 Comparison

Optical mouse illuminates the surface with infrared LED light. While the Laser mouse illuminates the surface with a laser beam.

Optical mouse senses the top of the surface it’s on. Laser mouse senses peaks and valleys in a surface. Optical mouse works better on mouse pads and non-glossy surface. Laser mouse has no such preferred surfaces. [42]

Our Proposed AI Virtual Mouse

98

Hand Gesture Recognication

97

Virtual Mouse Using Object Tracking

95

Virtual Mouse Using Sixth Sense Technology

93

0 10 20 30 40 50 60 70 80 90 100

**Figure 5.4:** Graph for Compression between The Models

## 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. 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 fingertip detection algorithm to produce more accurate results. [43]

# 9.CONCLUSION

## 9.1 Overview

In the Human-Computer Interfaces (HCI) field, where every mouse movement may be done with a fast of your fingertips anywhere, it should come as no surprise that the real mouse will also be overtaken by an immersive non-physical mouse and without regard to the environment, at any moment. In order to replace the common physical mouse without sacrificing precision and efficiency, this project had to design a color recognition program. This program can recognize color movements and combinations and translate them into functional mouse actions. A few strategies had to be used because accuracy and efficiency are crucial factors in making the application as helpful as a real-world mouse.

The primary objective of this method is to lessen and maintain the sensitivity of the cursor by averaging the values of the colors responsible for managing cursor motions based on a set of coordinates movements, as even a small movement could cause unintended cursor movements. In addition, a number of color combinations were developed with the relation of distance computations between the two colors in the combination because a difference in distance can result in a difference in the way the mouse behaves. This implementation's goal is to make it easier to control the application with minimal trouble. As a result, accurate mouse function triggering can be achieved with little trial-and-error.

Moreover, calibrations phase was included to promote effective and versatile color tracking. This enables people to select their preferred colors for various mouse functions as long as the chosen colors don't identical or comparable RGB hues (e.g. blue and sky-blue). Responsive validations were additionally developed, which essentially enables the software to save various sets of HSV levels across various angles to be utilized during the initialization step.

In regards to efficiency and lifestyle, modern technology has made significant progress in improving society's quality of life, as opposed to the other side around. Hence, cultures must not mix while hesitantly adopting outdated technologies. The latest one is accepting revisions at the IA(HONS) Information System Engineering Department of the Institute of Information and Communication Technology (Perak Campus), UTAR 40. Instead, it is advised that individuals accept modifications to lead a lifestyle that is more effective and productive.

## 9.3.Limitation

There are number of ongoing issues in this research that could impede the outcomes of color recognition. The environment aspect when the recognition phase is taking place is one of the issues. The recognition procedure is very sensitive to brightness levels since extreme intensity or blackness may make it impossible to see the targeted colors in the acquired frames.

In addition, distance is another issue that could have an impact on outcomes of color identification. As the current detecting zone can only allow displays of color within limited radius. Any displays of colors beyond this limitation will be viewed as noise and filtered out.

## 9.3 Future Project

The application needs a number of features and enhancements to be more versatile, accurate, and user- friendly in a variety of settings. The required upgrades and functionalities are described as follows:

9.1.1 Intelligent Recognition Algorithm:

An adjustable zoom-in/out function is needed to increase the covered range because the existing recognition mechanism can only detect objects within a 25 cm radius. This function can automatically modify the focus rate dependent on the proximity between both the operator and the video camera.

9.1.2 Increased Efficiency

The hardware of the device, which comprises the processor's processing speed and the amount of the memory space, has a significant impact on the download speed. RAM and the webcam's capabilities. As a result, the program might run more effectively when it's installed on a good computer with a webcam that works well in various lighting conditions.

# References

|  |  |
| --- | --- |
| [1] | Banerjee, A., Ghosh, A., Bharadwaj, K., & Saikia, H. (2014). Mouse control using a web camera  based on color detection. arXiv preprint arXiv:1403.4722. |
| [2] | C.-Chiung Hsieh, D.-Hua Liou, & D. Lee, “A real time hand gesture recognition system using motion history image,” Proc. IEEE Int'l Conf. Signal Processing Systems (ICSPS), 2.  10.1109/ICSPS.2010.5555462, 2010. |
| [3] | C.-Y. Kao, C.-S. Fahn, "A Human-Machine Interaction Technique: Hand Gesture Recognition  Based on Hidden Markov Models with Trajectory of Hand Motion," in Procedia Engineering 15 pp. 3739 – 3743, 2011. |
| [4] | P.S. Neethu, “Real Time Static & Dynamic Hand Gesture Recognition,” International Journal of  Scientific & Engineering Research, Vol. 4, Issue3, March 2013. |
| [5] | A. M. Patil1, S. U. Dudhane1, M. B. Gandhi1,” Cursor Control System Using Hand Gesture  Recognition”, International journal of advanced research in computer and communication engineering, vol. 2, issue: 5, May 2013. |
| [6] | A. Banerjee, A. Ghosh, K. Bharadwaj, “Mouse Control using a Web Camera based on Color  Detection,” IJCTT, vol.9, March 2014. |
| [7] | Y. Zhou, G. Jiang & Y. Lin, “A novel finger and hand pose estimation technique for real-time  hand gesture recognition,” in Pattern Recognition vol. 49, pp. 102-114, January 2016 |
| [8] | P. Kumari, S. Singh & V. Kr. Pasi, "Cursor Control using Hand Gestures," Recent Trends in Future Prospective in Engineering & Management Technology 2016, International Journal of Computer  Applications (0975 – 8887), 2016. |
| [9] | A. Haria, A. Subramanian, N. Asokkumar, S. Poddar & J. S. Nayak, “Hand Gesture Recognition for Human Computer Interaction," 7th International Conference on Advances in Computing & Communications, ICACC-2017, Cochin, India, Procedia Computer Science 115 (2017), pp.367–  374, 22-24 August 2017. |
| [10] | S. S. Abhilash, L. Thomas, N. Wilson & C. Chaithanya, "VIRTUAL MOUSE USING HAND GESTURE," International Research Journal of Engineering and Technology (IRJET), vol. 05  issue: 4, April 2018 |

|  |  |
| --- | --- |
| [11] | Kollipara Sai Varun, I. Puneeth, Dr. T. Prem Jacob ‘’ Virtual Mouse Implementation using Open CV’ ’Guoli Wang, "Optical Mouse Sensor-Based Laser Spot Tracking for HCI Input", Proceedings of the 2015 Chinese Intelligent Systems Conference, vol. 2, pp. 329-340,  2010. |
| [12] | Kabid Hasan Shibly; Smarat kumar Dey; Md. Aminul Islam; Shariar Iftekar Sourav ‘’ Design and Development of Hand Gesture Based Virtual Mouse’’ D.L. Quam, "Gesture recognition with a  Data Glove", vol. 2, pp. 755-760, 1990. |
| [13] | B. Nagaraj, J. Jaya, S. Shankar, P. Ajay ‘’ Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread’’1. J. Katona, “A review of human– computer interaction and virtual reality research fields in cognitive Information  Communications,” Applied Sciences, vol. 11, no. 6, p. 2646, 2021 |
| [14] | Ahmed, Muhammad et al. “Survey and Performance Analysis of Deep Learning Based Object  Detection in Challenging Environments.” Sensors (Basel, Switzerland) vol. 21,15 5116. 28 Jul. 2021, doi:10.3390/s21155116 |
| [15] | V. Tiwari, C. Pandey, A. Dwivedi and V. Yadav, "Image Classification Using Deep Neural Network" 2020 2nd International Conference on Advances in Computing, Communication  Control and Networking (ICACCCN), 2020, pp. 730-733. |
| [16] | Li Wensheng, Deng Chunjian and Lv Yi, "Implementation of virtual mouse based on machine  vision" The International Conference on Apperceiving Computing and Intelligence Analysis Proceeding, 2010, pp. 367-371 |
| [17] | A. Mhetar, B. K. Sriroop, A. G. S. Kavya, R. Nayak, R. Javali and K. V. Suma, "Virtual mouse"  International Conference on Circuits, Communication, Control and Computing, 2014, pp. 69-72. |
| [18] | S. M. S. Shajideen and V. H. Preetha, "Hand Gestures - Virtual Mouse for Human Computer Interaction" International Conference on Smart Systems and Inventive Technology (ICSSIT),  2018, pp. 543-546. |
| [19] | K. S. Varun, I. Puneeth and T. P. Jacob, "Virtual Mouse Implementation using Open CV" 3rd  International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 435-438. |
| [20] | S. R. Chowdhury, S. Pathak and M. D. A. Praveena, "Gesture Recognition Based Virtual Mouse  and Keyboard" 4th International Conference on Trends in Electronics and Informatics (ICOEI), 2020, pp. 585- 589. |

|  |  |
| --- | --- |
| [21] | Sai Mahitha G., Revanth B., Geetha G., Sirisha R. “Hand Gesture Recognition to Implement Virtual Mouse Using Open Source Computer Vision Library: Python” Proceedings of International Conference on Advances in Computer Engineering and Communication Systems.  Learning and Analytics in Intelligent Systems, vol 20. Springer, Singapore, 2021, pp. 435-446. |
| [22] | Tran, DS., Ho, NH., Yang, HJ. et al. “Real-time virtual mouse system using RGB-D images and  fingertip detection” Multimed Tools Appl 80, 10473–10490, 2021. |
| [23] | V. V. Reddy, T. Dhyanchand, G. V. Krishna and S. Maheshwaram, "Virtual Mouse Control Using  Colored Finger Tips and Hand Gesture Recognition" 2020 IEEE-HYDCON, 2020, pp. 1-5 |
| [24] | Dinh-Son Tran & Ngoc-Huynh Ho & Hyung-Jeong Yang & Soo-Hyung Kim & Guee Sang Lee  “Real-time virtual mouse system using RGB-D images and fingertip detection” 2021 |
| [25] | Aabha Waichal, Mauli Gandhi, Amruta Bhanji1, Srushti Bhagwat, Shalaka Deore “Survey paper  on Hand Gesture Recognition Based Virtual Mouse” in 2022 |
| [26] | Prof. Monali Shetty, A. Daniel, Manthan K. Bhaskar, Ofrin P. Lopes “Virtual Mouse Using Object Tracking”. [IEEE 2020 5th International Conference on Communication and Electronics Systems  (ICCES) - COIMBATORE, India (2020.6.10-2020.6.12)] |
| [27] | Swati Tiwari, Siddhi Parkar, Shruti Gharat, Kinjal Patel, Prof. Khalil Pinjari “Virtual mouse  using sixth sense technology” May 2017 [INTERNATIONAL JOURNAL OF ADVANCED](https://www.researchgate.net/journal/INTERNATIONAL-JOURNAL-OF-ADVANCED-RESEARCH-IN-ENGINEERING-TECHNOLOGY-0976-6499) [RESEARCH IN ENGINEERING & TECHNOLOGY](https://www.researchgate.net/journal/INTERNATIONAL-JOURNAL-OF-ADVANCED-RESEARCH-IN-ENGINEERING-TECHNOLOGY-0976-6499) 6(5):564-567 |
| [28] | M. Teixidoa, T. Palleja, M. Tresanchez, D. Font, J. Moreno, A. Fernandez del Viso, C. Rebate,  J. Palacin “Optimization of the virtual mouse Head Mouse to foster its classroom use by children with physical disabilities’’ |
| [29] | Rachit Puri published “Gesture Recognition Based Mouse Events’’ IM\Web, Multimedia & Services\Web Solutions\Web Engine Samsung Research India-Bangalore  Bangalore – 560037, India. |
| [30] | Vijay Kumar Sharma, Vimal Kumar, Md. Iqbal, Sachin Tawara, Vishal Jayaswal  “Virtual Mouse Control Using Hand Class Gesture’’ in December 2020 |
| [31] | Prachi Agarwal, Abhay Varshney, Garvit Bhola, Harsh Gupta, Harsh Beer Singh “Gesture  Controlled Virtual Mouse” |
| [32] | Sarkar, Sankha, and India Sumanta Chatterjee. "A Vision Base Application for Virtual Mouse  Interface Using Hand Gesture” |

|  |  |
| --- | --- |
| [33] | Abhishek R. Shukla “AI Virtual Mouse Using Hand Gesture Recognition” |
| [34] | Tsai, Tsung-Han; Huang, Chih-Chi; Zhang, Kung-Long (2015). [IEEE 2015 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW) - Taipei, Taiwan (2015.6.6- 2015.6.8)] 2015 IEEE International Conference on Consumer Electronics - Taiwan *– “*Embedded virtual mouse system by using hand gesture recognition”. (), 352*–*  *353.* doi:10.1109/ICCE-TW.2015.7216939 |
| [35] | U. Sairam; [Dharani Kumar Reddy Gowra](https://ieeexplore.ieee.org/author/37089418260); [Sai Charan Kopparapu](https://ieeexplore.ieee.org/author/37089415317) “Virtual Mouse using  Machine Learning and GUI Automation” |
| [36] | Omkar Shinde, Kiran Navale, Dipak Kunjir, Akshay More, Prof. Ashwini Taksal “Gesture  Recognition Based Virtual Mouse and Keyboard” |
| [37] | Dipankar Gupta, Emam Hossain, Mohammed Sazzad Hossain, Mohammad Shahadat Hossain,  and Karl Andersson4 “An Interactive Computer System with Gesture-Based Mouse and Keyboard” |
| [38] | K. Bharath Reddy; Md. Fayazuddin; M. John Manohar “Artificial Intelligence Based Virtual  Mouse” |
| [39] | Roshan Hyalij, Siddharth Oturkar, Atharv Kasodekar, Varun Kothawade, Kamlesh Patil,  Prof.Aashutosh kale “VIRTUAL MOUSE APPLICATION” |
| [40] | Khushi Patel, Snehal Solaunde, ShivaniBhong, Prof. Sairabanu Pansare “Virtual Mouse Using  Hand Gesture and Voice Assistant” |
| [41] | Gauri Kulkarni, Babasaheb Khedkar, Animesh Kotwal, Mohit Modi “HEAD TRACKING  VIRTUAL MOUSE” |
| [42] | <https://chat.openai.com/?model=text-davinci-002-render-sha>.  Last Access Date: 15/05/2023 |
| [43] | <https://www.geeksforgeeks.org/difference-between-optical-mouse-and-laser-mouse/>  Last Access Date: 15/05/2023 |
| [44] | <https://ijariie.com/AdminUploadPdf/AI_Virtual_Mouse_ijariie19200.pdf>  Last Access Date: 15/05/2023 |
| [45] | <https://youtube.com/watch?v-vjwzH_2F64g&festure>  Last Access Date: 15/05/2023 |

|  |  |
| --- | --- |
| [46] | Samia, Moythry Manir, et al. "Aspect-based Sentiment Analysis for Bengali Text using Bidirectional Encoder Representations from Transformers (BERT)." International Journal of  Advanced Computer Science and Applications 13.12 (2022). |

MCA Narayana Engineering College : : Gudur Page | 91