A Project Report

on

Virtual Mouse with hand gestures Tracking using Image Processing

Submitted to

The Department of Computer Science and Engineering

In partial fulfillment of the academic requirements of

Jawaharlal Nehru Technological University

For

The award of the degree of

Bachelor of Technology in

Computer Science and Engineering

(2017 – 2021)

By

Yerra Yashua Krupason	17311A05M2
Buddavarapu Teja Swaroop	17311A05M5
Pathkoti Dinesh Kumar	17311A05M3

Under the Guidance of

Mrs. Pallepati Vasavi

Assistant Professor

Sreenidhi Institute of Science and Technology

(An Autonomous Institution)

Yamnampet, Ghatkesar, R.R. District, Hyderabad - 501301

Department of Computer Science and Engineering

Sreenidhi Institute of Science and Technology

CERTIFICATE

This is to certify that this Project-1 report on "Virtual Mouse with hand gestures Tracking using Image Processing", submitted by Yerra Yashua Krupason (17311A05M2), Buddavarapu Teja Swaroop (17311A05M5) and Pathkoti Dinesh Kumar (17311A05M3) in the year 2020 inpartial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as part of their **Project-1 during Fourth Year First Semester**, under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

INTERNAL GUIDE:	PROJECT COORDINATOR:	HEAD OF DEPARTMENT:
Mr. Arun Kumar Assistant Professor	Mrs. Pallepati Vasavi Assistant Professor	Dr. Aruna Varanasi Professor & HOD

External Examiner

Date:-

DECLARATION

We, **Yerra Yashua Krupason (17311A05M2), Buddavarapu Teja Swaroop (17311A05M5) and Pathkoti Dinesh Kumar (17311A05M3),** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR,** studying IVth year 1stsemester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Project-1 work, titled **"Mouse with hand gestures Tracking using Image Processing"** is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **DR. T. CH. SIVA REDDY,** who most ably run the institution and has had the major hand in enabling me to do my project.

I profoundly thank **DR. ARUNA VARANASI**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to mywork.

I would like to thank our Coordinator **MRS.P.VASAVI**& my internal **MR. ARUN KUMAR** for the Project-1 for their constant guidelines, encouragement and support in carrying out my project on time at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased mytask.

Yerra Yashua Krupason 17311A05M2

Buddavarapu Teja Swaroop 17311A05M5

Pathkoti Dinesh Kumar 17311A05M3

Abstract

With the development of technology and knowledge of establishing a connection and process of interaction between human and computer system is evolving since the invention of computer technology. So a specific interactive module like a virtual mouse that makes use of Object Tracking and Gestures that will help us to interact which might be an alternative way for the traditional touch screen technology and the physical mouse. In this project an Object Tracking application that interacts with the system to provide mouse functionalities is created which uses user hand with color bands or tapes. As the mouse is one of the important inventions in computer world, which has improved human machine interactivity. In present scenario, wireless and Bluetooth embedded mouse technology is developed but it is not that ideal to use as this technology is not completely hands free. So, in the proposed model these limitations are removed and interaction is made user friendly as well as cost is also reduced. This model is actually a virtual mouse system using hand gestures with the help of computer vision. Python and OpenCV library is used for real time and computer vision to implement the system. The camera output is displayed on the screen of the monitor. This system allows the user to use the mouse cursor using their hand with color tapes or bands, the webcam tracks and captures the images and these images are analysed perform mouse operations like left-click, right-click, and double click using different hand gestures.

	LIST OF FIGURES		
S.NO	**Fig No**	**Title of Figure**	**Page No**

	LIST OF TABLES		
S.NO	**Table No**	**Title of Table**	**Page No**
1	2.1	Existing System	14
2	5.2	Learning Rate of Cursor Movement	31
3	6.2	List of Test Cases	46

1. Introduction

1.1 Motivation

The mouse is one of the important inventions in computer world which improved human machine interactivity. In present scenario, wireless and Bluetooth embedded mouse technology is developed but it is not that ideal to use as this technology is not completely hands free. In Bluetooth technology the mouse developed requires power such as battery power or connecting devices like dongle.

By adding these extra components to the mouse the cost increases and it is also not user friendly to use. So, in the proposed model these limitations are removed and interaction is made user friendly as well as cost is also reduced. This model is actually a virtual mouse system using hand gestures with the help of computer vision.

1.2 Definition

The primary aim of this project is to improve human and computer interaction by developing an effective and alternative way of controlling the cursor and its various functions such as left click, right click, scroll up, scroll down and selection.

With the help of this model a user can easily interact with the system/computer without the use of an external device. It also allows the user to interact with system from a considerable distance. And because of this improvement the essential hardware required for the computer/laptop is also reduced.

This model makes use of the webcam to capture the video feed, which is used for image processing, and now a days webcam is an essential device/hardware for a laptop/computer and therefore it is not considered as an extra hardware.

1.3 Objective

The objective of this project is to improve the usability and experience of the users by providing a better machine and human interaction. The Image/Video feed from the webcam is used to track and detect gestures and mapping those unique gestures to different mouse events.

Retrieve the image data from the webcam and convert it to a usable format and use it as input. The image must be filtered to identify different colors. Make contours and find the mean positions of those contours. Track those mean positions to recognize the gestures. Map those gestures to different mouse actions and use them.

2. LITERATURE SURVEY

2.1 Existing System

In the existing system[1], a two colored model is used, which uses two distinguished colors and hand gestures to identify and perform respective mouse movements and events. This system performs only basic functionality of mouse i.e, mouse cursor movement, left clicks and right clicks as only two colors are used. This system in [1] uses RBG color coding in which it uses the exact color calibration color code by which we need to use same calibrated colors to get good accuracy. In [1] the background extraction, subtraction etc are done to the image and features are extracted in order to work with functionalities. To perform mouse events and movements an external code is used by which the time complexity is more which indirectly takes time for the system to know the feature the user is expecting and search and perform by the system in [1]. As the mouse events are limited in [1], in order to perform other functionalities we need other devises and these are the limitations of the system.

In [2], the algorithm they used is called convex hull algorithm where lines are drawn through the tips of the fingers and for each gesture made by hands, in the form of geometric shapes like square, triangle etc, a mouse event is mapped to it. But trying to gesture geometric shapes with hands is difficult and also the algorithm gave less accuracy in performing the actions and the continuous movement of fingers also results in strain in fingers.

The Methods used by Authors in [3] gave less accuracy in detecting the colors, especially when the background is not light and pale in color. In [4], authors developed eye ball tracking mouse control, but the limitation of the methods used in this paper [4] is eye blinking can occur randomly so accuracy and performance of the model decreases. The Authors of [5] used pupil movement detection using open source computer vision where the eye movement is converted into voltage and time graph. But for this method in [5] a normal webcam cannot be used. And accuracy in performing the mouse actions by methods of [2-3] gave only 75-80% accuracy.

In [6], authors used a special infra-red camera to achieve the functionality of the mouse. And through this method they were able to achieve three basic functionalities of the mouse i.e. right click, left click and free movement. A monochrome camera with built-in image processing is used in this paper [6] to achieve the functionality of the mouse. But the aim and objective of our project is to achieve a device free mouse control.

In [7], the authors developed a virtual mouse model using opencv. And mapped the gestures with mouse events using pyautogui library. But they developed the model such that it detects either one highlighted color or two highlighted colors. So, it has limited functionalities just like the model in [1]. And further developments of [7] can be made by perfoming actions like opening an application on detecting a unique color or gesture. So this paper gave us an idea to work on the future developments of the above paper. Based on some of the great future development ideas from the above papers and their limitations we came up with a project/model.

2.1.1 Comparing various methods

S.no	Method Used	Algorithms used and Drawbacks	Accuracy
1	Virtual Mouse Using Object Tracking	Hvm,Svm algorithms are used.Time taken to detect is nearly 20 sec	70-75%
2	Gesture Recognition Based Virtual Mouse	Convex Hull algorithm is used.Accuracy is 75-85% Continuous finger movement results in stress in fingures	75-85%
3	Design and Development of Hand Gesture Based Virtual Mouse	Computer Vision is used. Less accuracy and system recognition of colors is poor.	75-80%
4	Virtual Mouse Control by Webcam for the Disabled	Eye ball Tracking using matlab. Eye blinking may occur random	80-90%
5	Cursor Control Using Eye Ball Movement	Pupil movement detected using open source computer vision. The eye movement converted into voltage and time graph, normal webcam can’t be used.	80-85%

Table 2.1 Existing System

2.3 Proposed System

The model/system that we proposed is a cursor control model using image processing, which takes hand gestures with the color tapes and bands as input that are captured from the webcam of the system, is feed as an input which is converted into HSV color format. Then by detecting the centers of the color contours and thereby detecting the gestures, it performs mouse events and mouse movements accordingly. In this we used three standard colors which represents the three fingers of a hand and detect the gestures of the hand. We can calibrate the three colors based on the values of hue, saturation and value accordingly so that the exact color coding is not required i.e, exact colors are not used each time.

In this project a library in Python (PyAutoGui) and OpenCV library from cv2 is used to implement this model for Hand gesture tracking and performing the mouse movements accordingly. This system is easy to use and it is also easy to remember the gestures that are provided to perform different operations. And once the Model is deployed on the system, we can use this version of the mouse all the time and on all apps.

3. SYSTEM ANALYSIS

3.1 Functional Requirements Specifications

After analysing the project objective, we have come up with following modules:

- **Color Calibration:**
  With the help of Computer vision (Opencv2), we recognised three standard colors and converted them into HSV format using cv2.convert() function and calibrated the colors. Users can dynamically adjust hue, saturation, value of the colors.
- **Centroid Detection:**
  From the output of the color calibration module, we calculate a mean point which represents contour of each color. With these centroid points we detect the type of gesture position made by user.
- **Multiple Gesture Recognition:**
  By the position and movement made by the user various gestures are recognised, if any gesture is not recognised by the system model then it considers it as free movement of the mouse cursor with help of only one color.
- **Mouse Events and Movement:**
  Based on the gestures recognised by the model equivalent mouse events are performed with the help of 'Pyautogui', a python library.

3.2 Software Requirements

- 64-bit Operating System: Windows 7 or Higher
- Jupyter Notebook/ Google Colab
- Python version >3.0
- PyAutoGUI
- OpenCV

3.3 Hardware Requirements

- A Good Webcam

- RAM preferably >8GB for faster processing
- GPU
- Memory space >500GB

3.4 Environmental Specifications

- A will lit room with mostly light and pale colors.
- No other objects should be present in front of the webcam (particularly red, yellow blue colored objects) except colors on the user’s hand.

4. SYSTEM DESIGN

4.1 Architecture

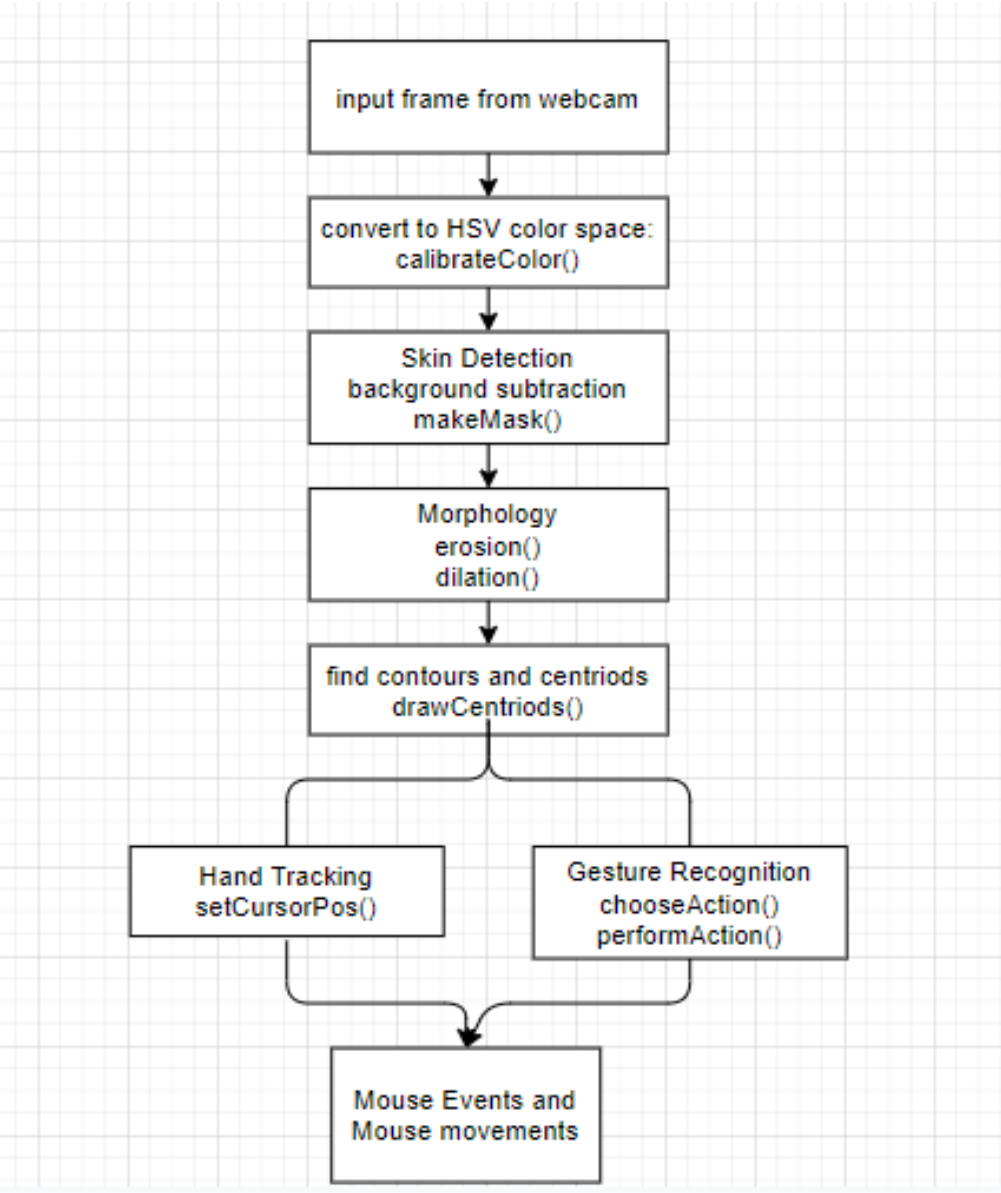


Fig 4.1 : Architecture Diagram

The architecture consists of taking input from webcam and converting into HSV color coding and then performing some operations to remove noise by using background subtraction and applying morphology. Then the contour is extracted and equivalent mouse events are performed.

4.2 UML Diagrams

4.2.1 Use Case Diagram

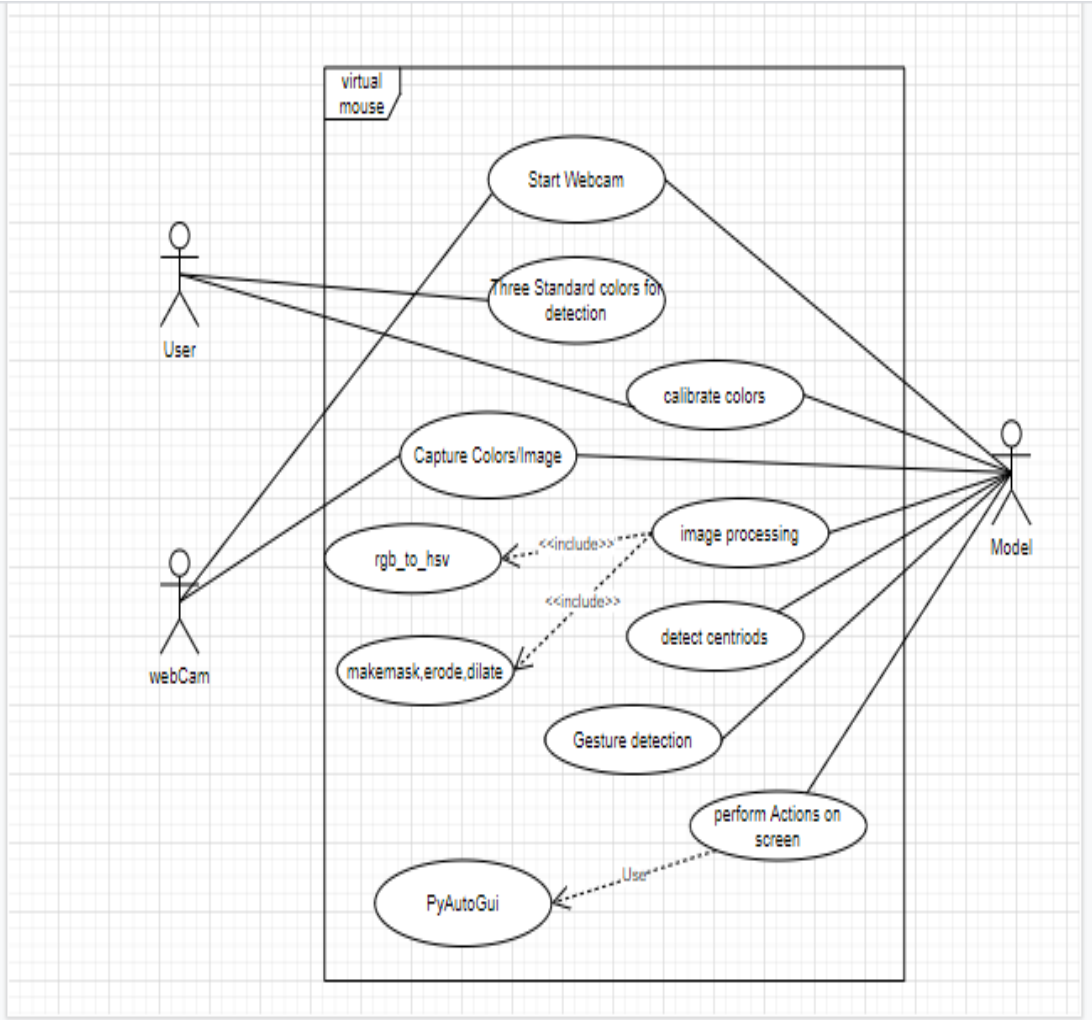


Fig 4.2.1 : Use Case Diagram

In this use case diagram user, webcam and model are the actors of the system. Various operations and functions of each actor is mentioned with their dependency and other relations are shown clearly in the diagram.

4.2.2 Class Diagram

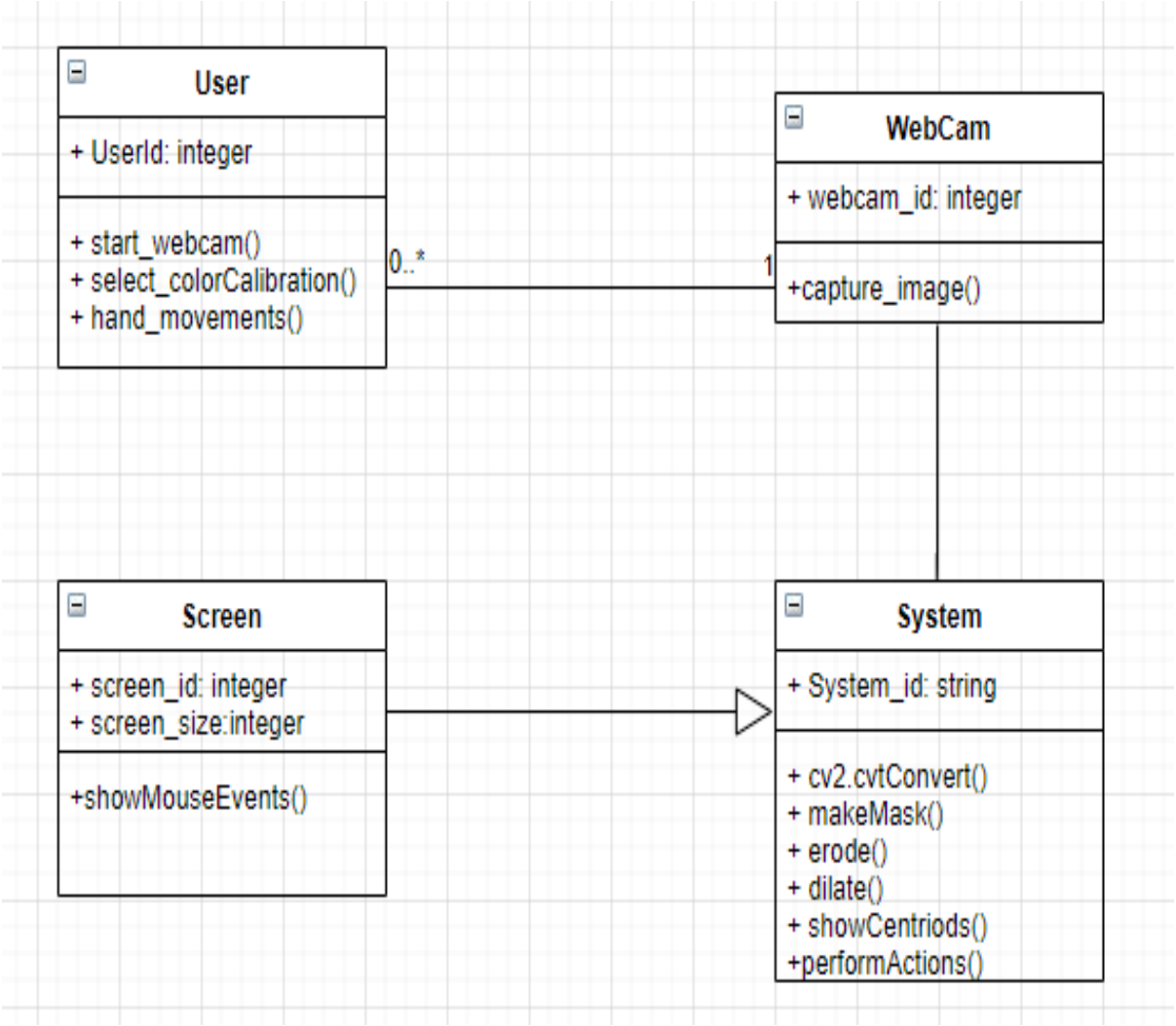


Fig 4.2.2 : Class Diagram

In this class diagram the user, webcam, screen and system are the classes used to represent this virtual mouse. The relationship between them is shown clearly and the functions and methods of each class is explained. Each class has its own attributes to describe the class which holds the variables and model or version description.

4.2.3 Sequence Diagram

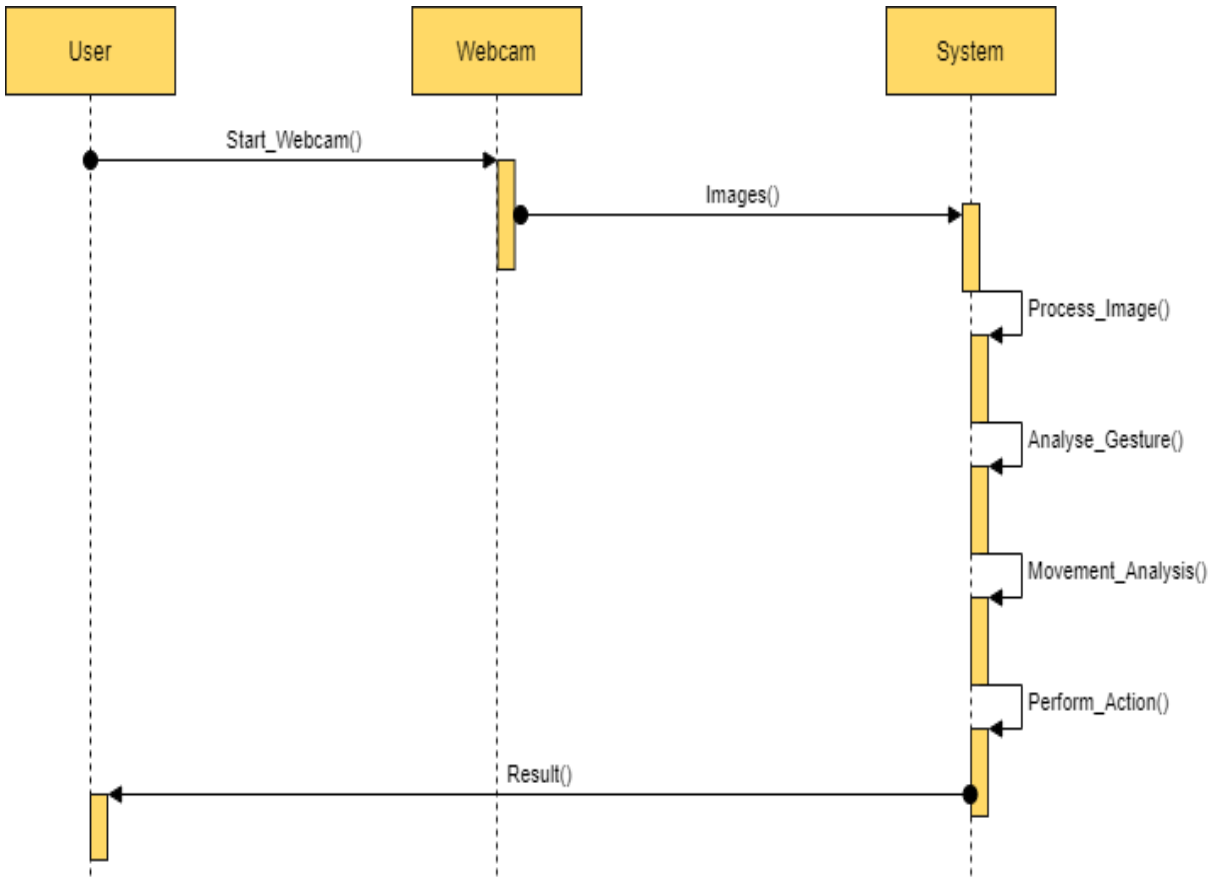


Fig 4.2.3 : Sequence Diagram

In this sequence diagram user, webcam and the system are the three parts where flow of control from one part to another is shown and works done at each stage by that particular system is described. The time and control is shown in the diagram and arrows represent the change in control.

4.2.4 Activity Diagram

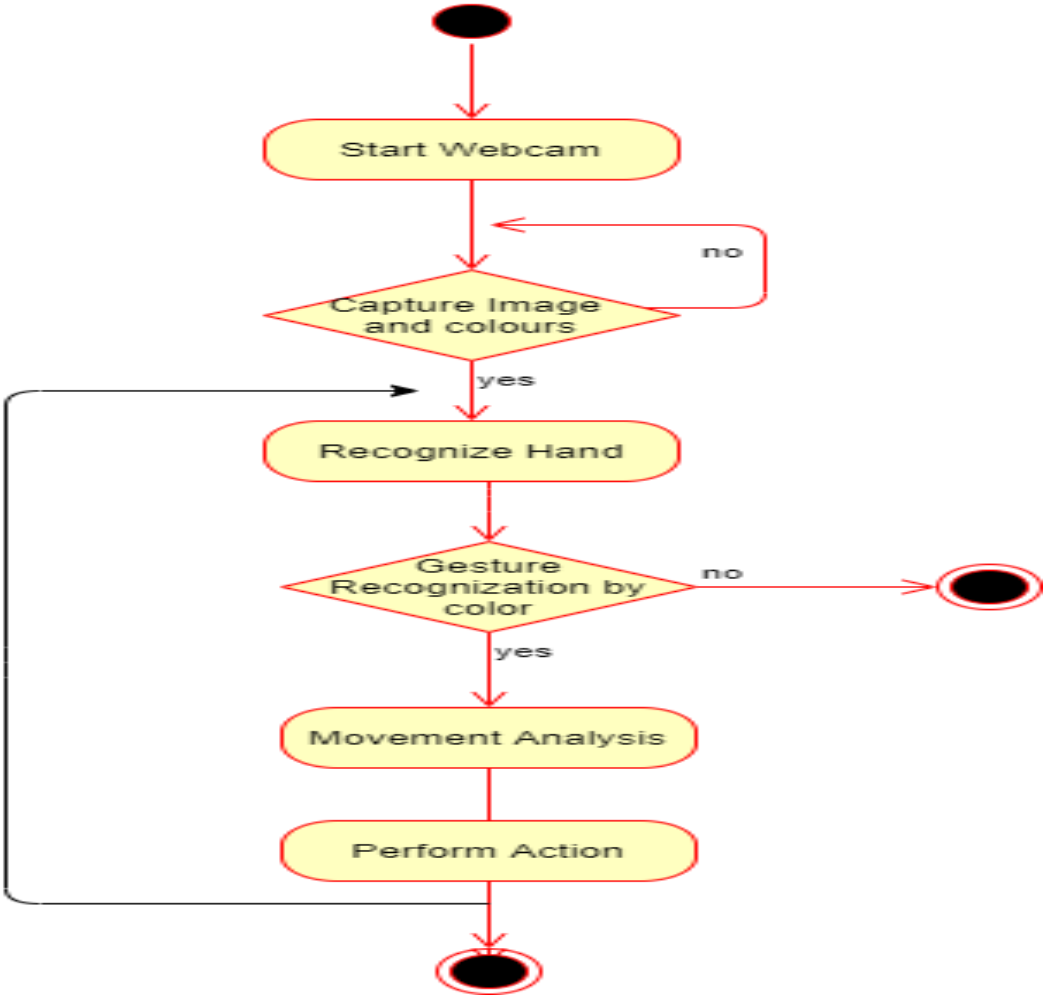
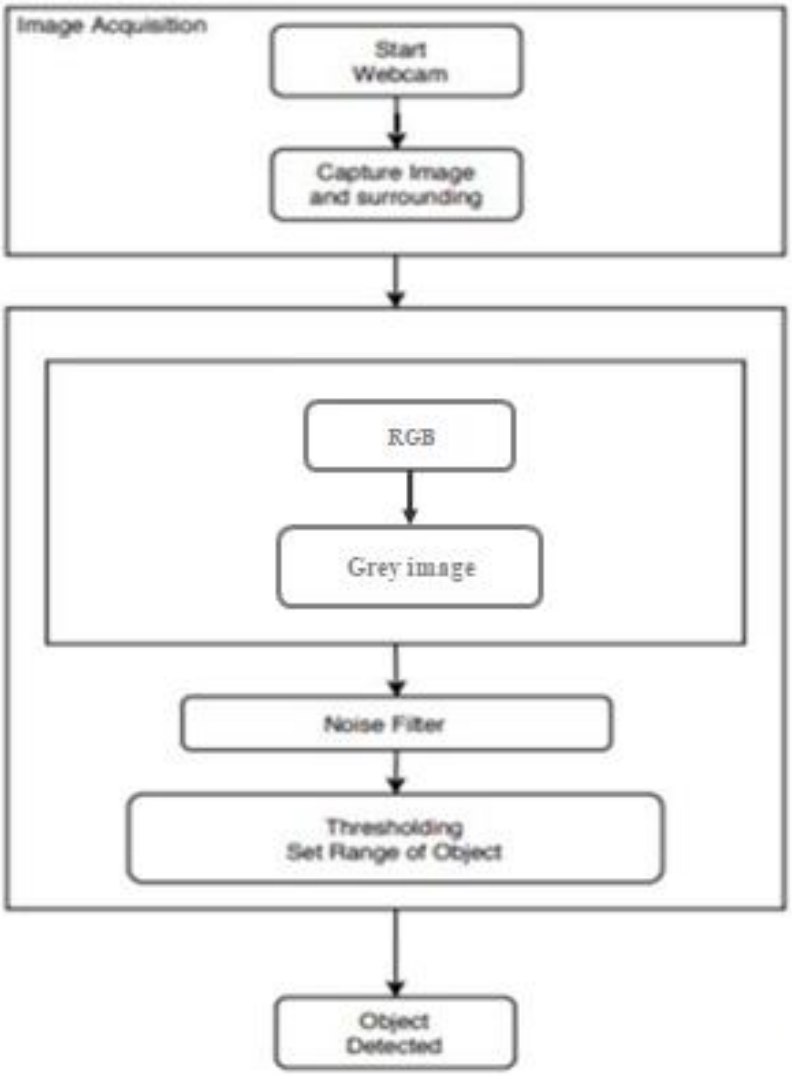


Fig 4.2.4 : Activity Diagram

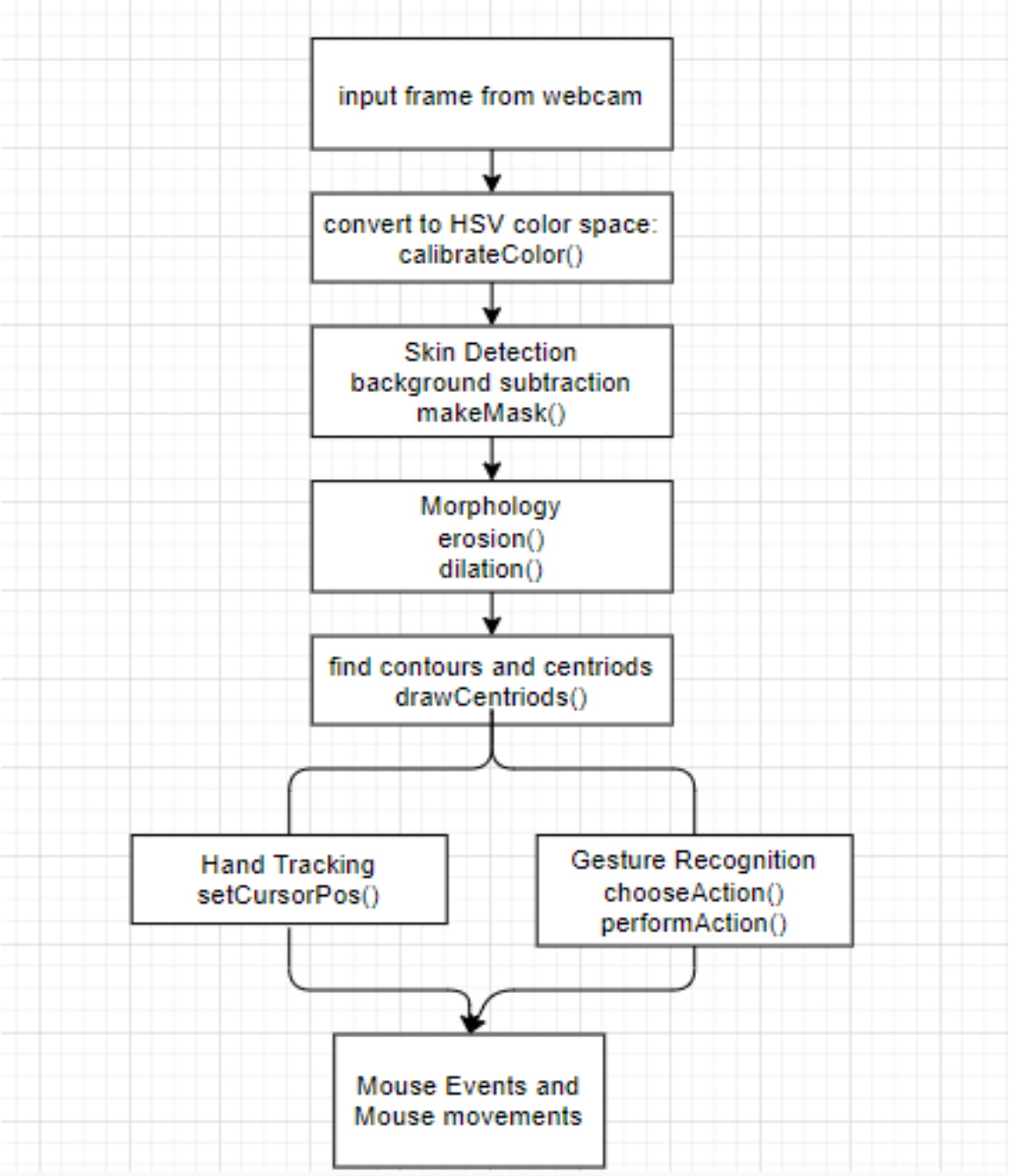
In this diagram the actions and the activities performed by the model is described and it starts from start and flows with the activities done by the model with the decisions and flow for both the options. The flow of model ends at the end of the activity diagram when the user end using the model or the model is interpreted.

4.3 Old vs New Architecture



In this system uses RGB color coding and it is converted into grey scale color coding image which requires lot of time and results in performance of the system. The same color coding need to be used each time so that when we change the calibration of color the system may not work properly.

4.3.2 New System



In this system the RGB color coding is not used and no need to convert the system to grey scale color coding, by which the performance of the system is increased. Dynamic color calibration is done so that there is no need for using same color coding image and thus increasing the accuracy and performance.

5. Implementation

5.1 Methodology

5.1.1 Concept

HSV color format is a color model which is considered to be in a cylindrical shape. It will map the RGB colors space to three Dimensional value space (hue, saturation, and value). Here each color is defined by three values hue, saturation and value.

Hue is considered as a value defining the angle of a particular color if RGB color space is considered as a color circle. Red color has a hue value of 0 degrees, Similarly green color has a hue value of 120 degrees and blue's hue value is considered as 240 degrees. Saturation specifies the intensity of the color or the amount of color used. Saturation value of 100% for a color specifies that the color is in its purest form possible, On contrast color with 0% saturation results in greyscale. Value tells about the brightness of the particular color. Any color with 0% of value is in pure black form whereas any color with 100% value has no black in it.

5.1.2 Implementation of Model

In this model we recorded the image from webcam and converted it from RGB to HSV color format. Then user required to calibrate the three colors i.e, red, yellow, blue in the image which consists of user hand with three colors on fingers individually. In this model we used calibrateColor() function to calibrate the three colors individually by the function. The user can manually calibrate the colors by adjusting the Hue, saturation, value ranges or they can choose default values.

From these calibrated individual images the three fingertips where the color band lies are extracted from user hand, in the video captured by webcam, using the cv2.inRange() function i.e. contours are detected. Noise is removed by applying morphism which contains two steps erosion and dilation. The output images are free from noise and are termed as 'mask', and from these images the centroids are calculated.

As a webcam is not ideal, so the noise of the background is also captured in the image and also the vibrations of the user may also get captured. Due to these vibrations of the user hand the centroids keep varying about a mean position which reduces the efficiency of the model. In order to increase the efficiency and differentiate the actual movements from vibrations, we used an enhanced formula to handle the noise caused by vibrations.

The function chooseAction is used to calculate the three centroids, which is used to decide the functions to be performed by the mouse based on the positions of the three centroids. Based on the action returned by chooseAction() function, we use performAction() function to perform the equivalent mouse events and movements using PyAutoGUI library which can be implemented in python environment. The actions that need to be performed by the system are:

- free cursor movement
- scroll up
- scroll down
- left click
- right click
- drag/select

5.1.3 Algorithm of Model

```
Function distance(center1,center2)
{
        Distance ← square root(center1*center1 + center2*center2)
        return Distance
}


Function makeAMask(hsv_image, ranges_of_color)
{
        Mask ← opencv2.inrange(hsv,color_range)

        // Morphosis
        Eroded ← erode(mask,kernel)
        Dilated ← dilate(eroded,kernel)

        return Dilated
}


Function setCursorPosition(centeriod,previous_pos)
{
        If the cursor movement is voluntary then
                Position = centriod + (learning rate)*(previous_pos - centriod)

        If the cursor movement is because of vibration of hand or involuntary then
                Position = centriod + (learning rate)*(previous_pos - centriod)
```

```
        return Position
}


Function CalibrateColor(color,range)
{
        Opencv2.createHsvTrackBar
        While true
                Frame ← CaputeFrame().read()
                Frame ← Flip_frame(Frame)
                Hsv ← convertColor(Frame,RGB_TO_HSV)
                makeAMask(hsv,color_range)

        return range
}


Function chooseAction(yellow_position, red_center, blue_center)
{
        If distance of yellow ,red and blue centers is less then
                Action ← drag

        Else if distance of red and blue is less then
                Action ← left_click

        Else if distance of red and blue is less then
                Action ← right_click
```

Else if distance of yellow and red is more and difference of red and blue is more then

Action ← scroll_down

Else if difference between red and blue is more then

Action ← scroll_up

Else

Action ← free_movement

return Action

}

5.1.4 Noise rate formula

To deal with the noise caused by unwanted vibrations we use following method- first we compare the new position of the centroid with the previous position. If the difference between them is less than 5 pixels, we considered that it occurred due to noise.

If the difference between them is more then it is done by user voluntarily, so the cusor position is changed accordingly by this function.

- To set cursor position, if the cursor movement is voluntary then

  Position = centriod + (learning rate)*(previous_pos - centriod)

- if the cursor movement is because of vibration of hand or involuntary then

  Position = centriod + (learning rate)*(previous_pos - centriod)

To decide the learning rate, we tried different values ranging from 0 to 1, to decide the optimal learning rate, to get good accuracy and ideal sensitivity, for both voluntary movement and unwanted vibrations. The results of the different learning rates is mentioned below in table 5.2 .

5.1.5 Comparison of various learning rates-

S.No.	Mouse Event	Pixel Difference	Description	Learning Rate	Sensitivity
1	Cursor Movement	< 5	Voluntary Hand Movement	0.7	Ideal
2	Cursor Movement	< 5	Voluntary Hand Movement	0.8	Ideal
3	Cursor Movement	< 5	Voluntary Hand Movement	0.9	High Sensitivity
4	Cursor Movement	< 5	Voluntary Hand Movement	1.0	High Sensitivity
5	Cursor Movement	< 5	Voluntary Hand Movement	0.6	Low Sensitivity
6	Cursor Movement	>5	Vibrations/Noise	0.1	Ideal
7	Cursor Movement	>5	Vibrations/Noise	0.0	High Sensitivity
8	Cursor Movement	>5	Vibrations/Noise	0.2	Low Sensitivity
9	Cursor Movement	>5	Vibrations/Noise	0.3	Low Sensitivity

Table 5.2 Learning rate for Cursor Movement

5.2 Results and discussion

All the results we achieved are satisfactory and all the six basic functionalities are performed using the gesture recognition. And the distance from which the gestures are recognized by the model depends on the quality of the webcam. As laptops have similar kind of webcams, the range of distance from which the gestures can be recognized is tested and the range is from 15cm to 1.2 meters. The model can even perform actions even at the distance of two meters but it also depends upon the background. If the background has light and pale colors then the model can perform well upto 2 meters. Once the colors are detected initially, then we do not require the mouse to control the cursor.

The flow of execution of the model is explained in following steps:

- First, using the calibrateColor() function from the cv2-opencv library, the colors from the video feed are detected. And Hue,Sat,Value track bar is provided to manual adjust the values to detect the colors properly.
- Using cvtColor() function, we converted the colors from RGB format to HSV format.Using Kernal matrix, masking is done.
- And the erode and dilate functions are used to remove the noise and to make the borders smoother.
- After the three colors are detected and contours are formed the centriods of the contours are detected using moments() function. Centriod of the Yellow color is used for the cursor movement on the scree and all three centriods are used for mouse events.
- From the Figures 5.3.1, 5.3.2, 5.3.3, you can see that the three colors are calibrated. From Figure 5.3.4, you can observe that the output screen provides instructions for calibrarting the colors.

- After all the colors are captured, a frame/window opens which shows the video captured by the webcam. When the video is captured by webcam, video is generally flipped/mirrored. So, using a flip function from cv2 library image is flipped to get the original version.
- By pressing the button ‘C’, you can see the centriods as shown in the figure 5.3.6.
- From the sub heading 5.3.4 you can see the different gestures for different mouse events.
- The gestures are recognized by the orientation of different centriods of the colors using chooseAction() function.
- As the final step, performAction() function maps the different gestures to different mouse events and performs those mouse events using pyAutoGUI library. You can see the actions being performed from the Figures 5.3.13 to 5.3.19.

5.3 Results/Screenshots

5.3.1 Calibration of colors

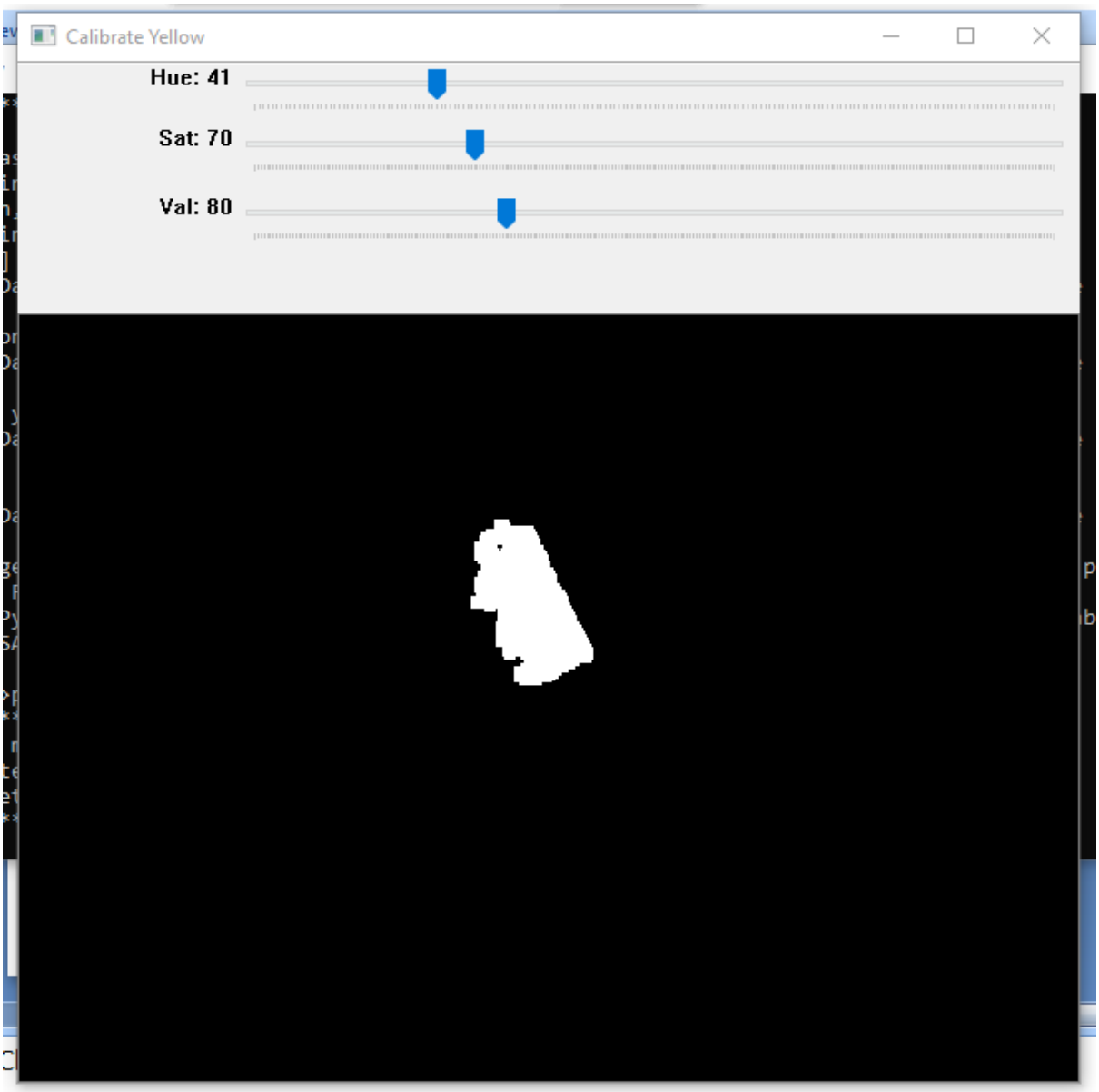


Fig 5.3.1 Calibrating Yellow color

In this we adjust the hue, saturation and value of the Yellow color so that the user need not use the same color coded bands or tapes. By this we can also reduce the noise and other background which may effect the system.

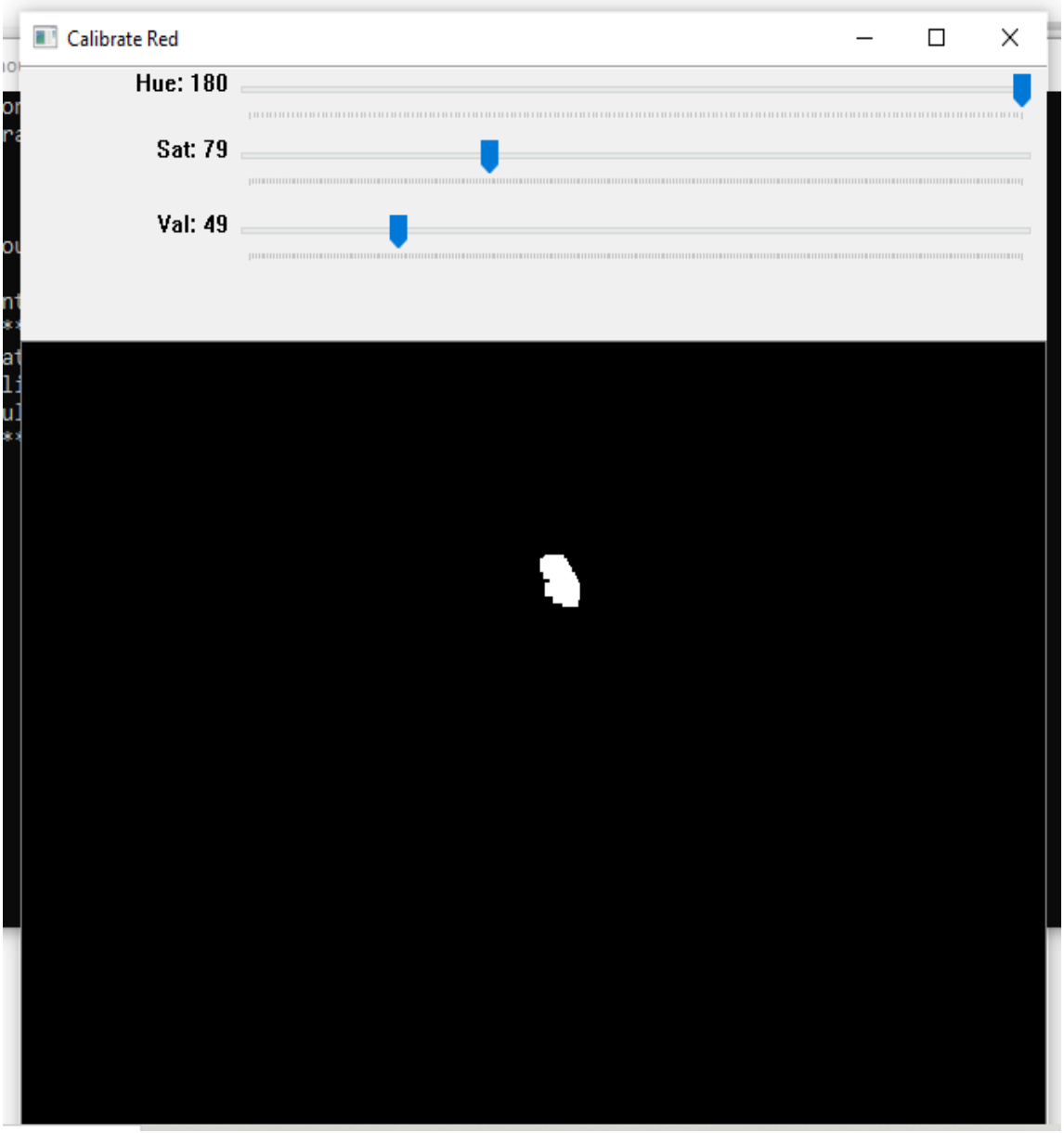


Fig 5.3.2 Calibrating Red Color

In this we adjust the hue, saturation and value of the Red color so that the user need not use the same color coded bands or tapes. By this we can also reduce the noise and other background which may effect the system.

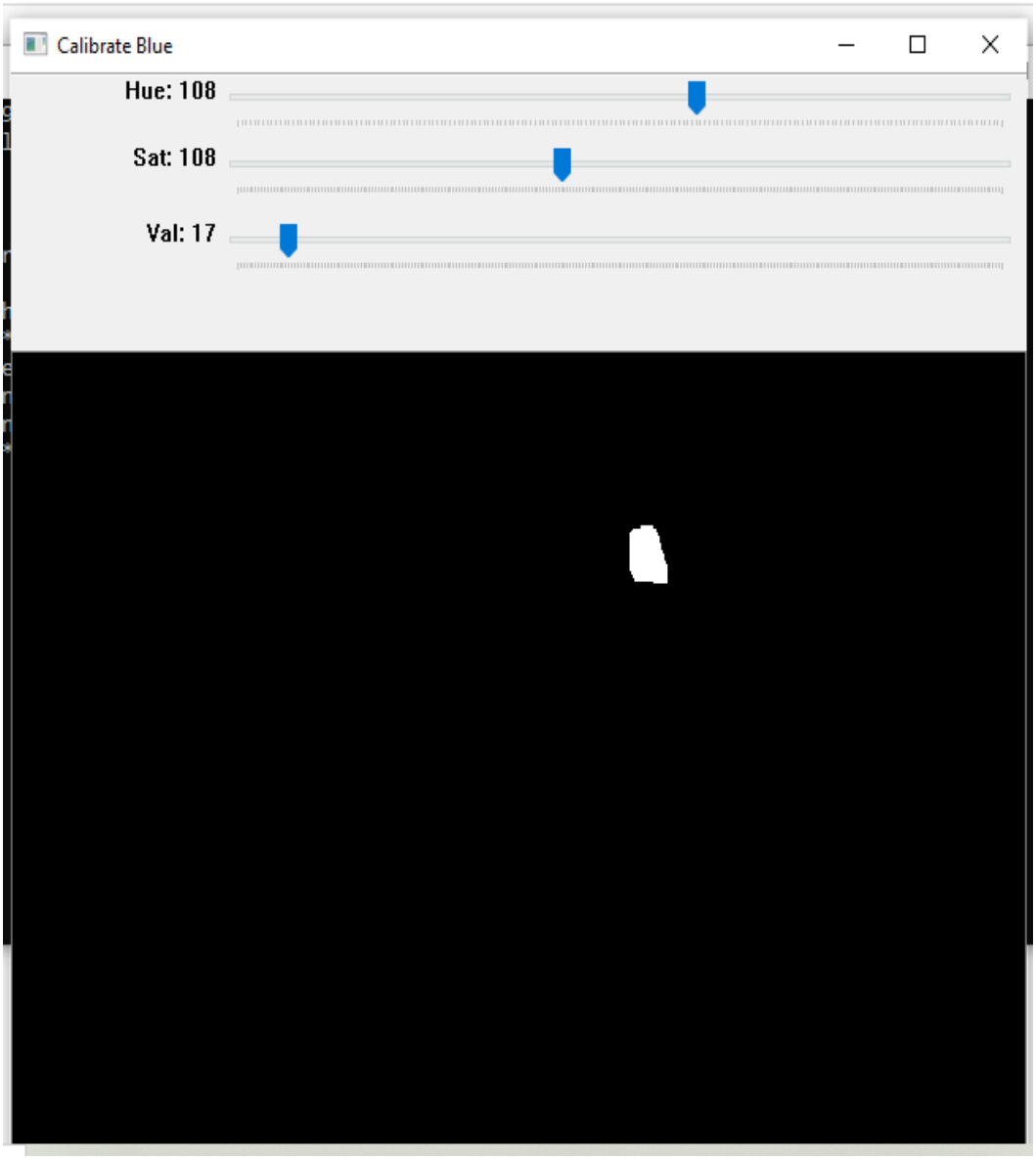


Fig 5.3.3 Calibrating Blue color

In this we adjust the hue, saturation and value of the Blue color so that the user need not use the same color coded bands or tapes. By this we can also reduce the noise and other background which may effect the system.

Output Screen

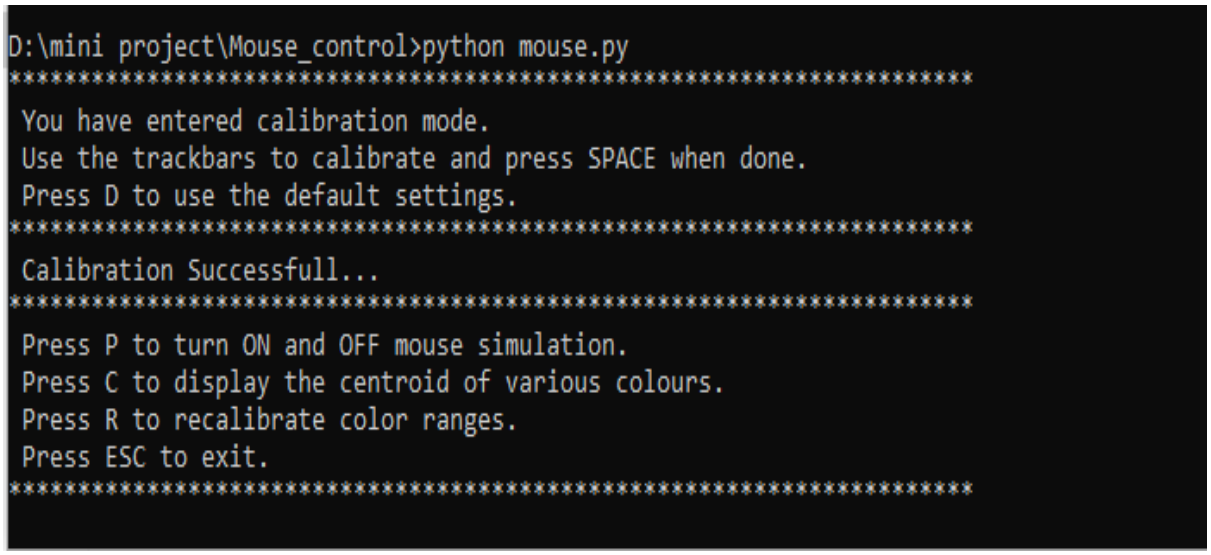


Fig 5.3.4 Calibration Successful

In this output screen shot we can see the calibration is successful and there are various opyions available like P, C and R which performs various operations. We can also skip and press ESC to escape from screen exit.

5.3.2 Capturing the Image.

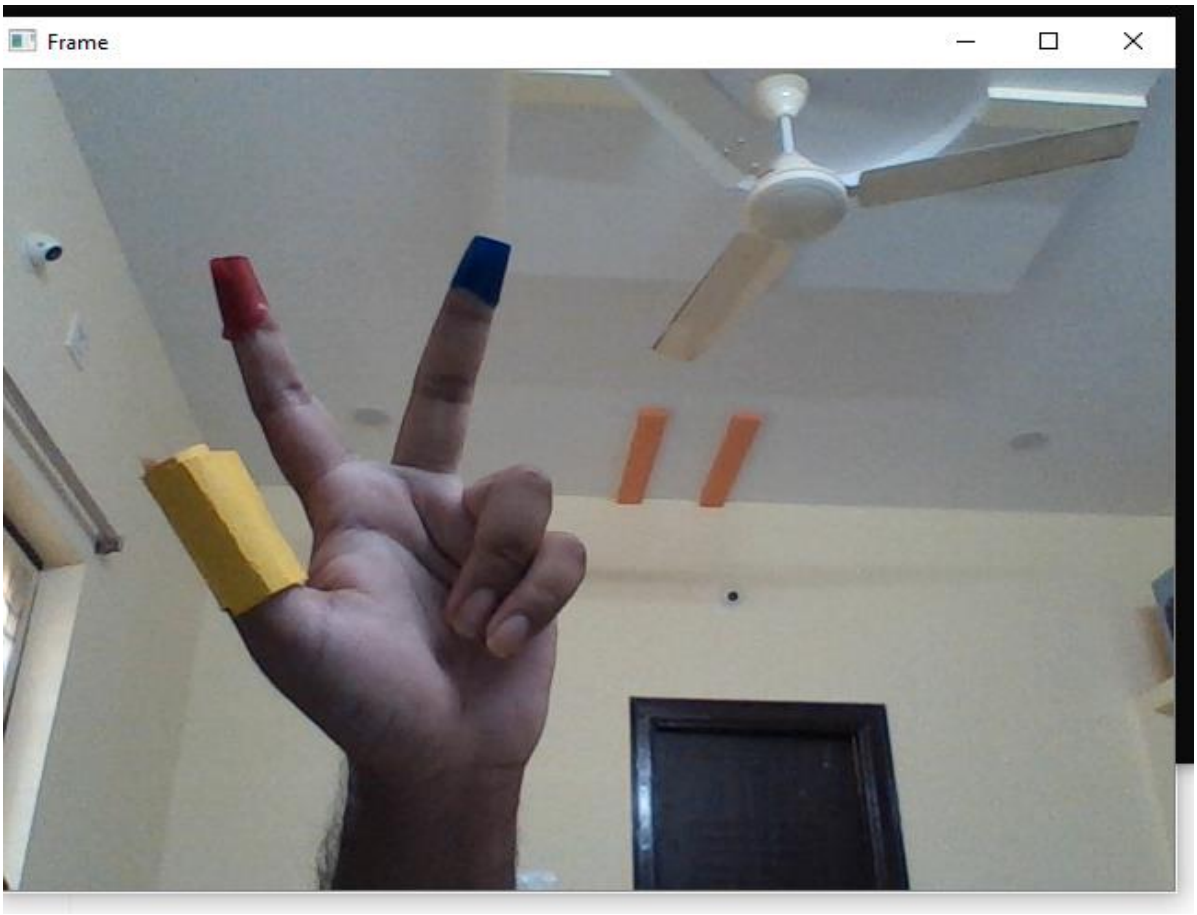


Fig 5.3.5 Capturing Frame

First the image from the webcam is captured which consists of user hand with the color band and tapes with the background. The three tapes yellow, red and blue is used by the user present in the frame.

5.3.3 Centroid Recognition

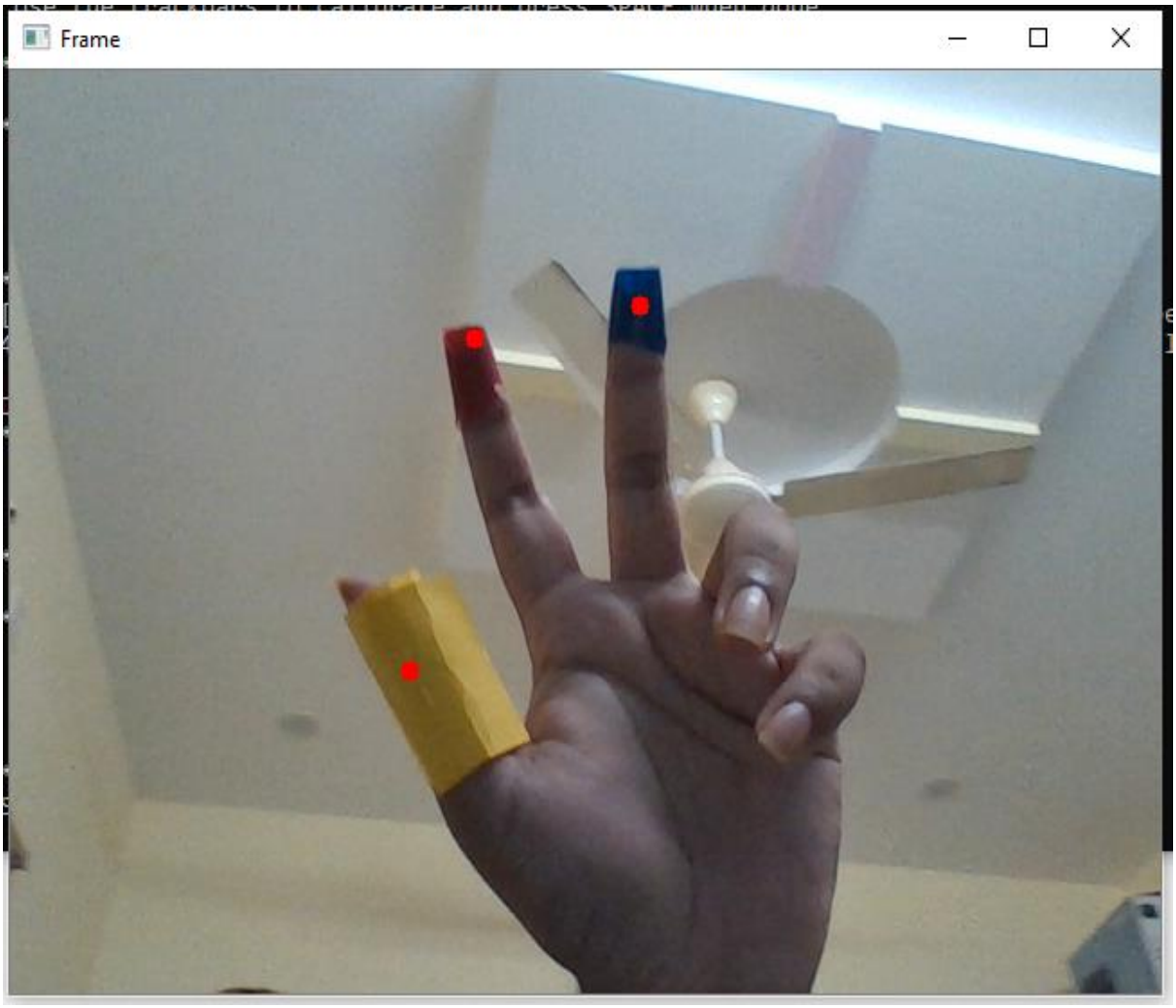


Fig 5.3.6 Centriod Detection

The centroids of the three color tapes are detected which uniquely defines the whole color bands or tapes by which the distance between the three colors are known and gestures are recognized.

5.3.4 Gestures for Actions

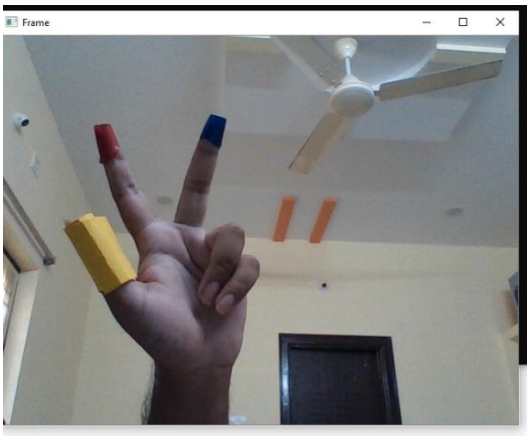


Fig 5.3.7 Free Movement

Fig 5.3.8 Scroll Up

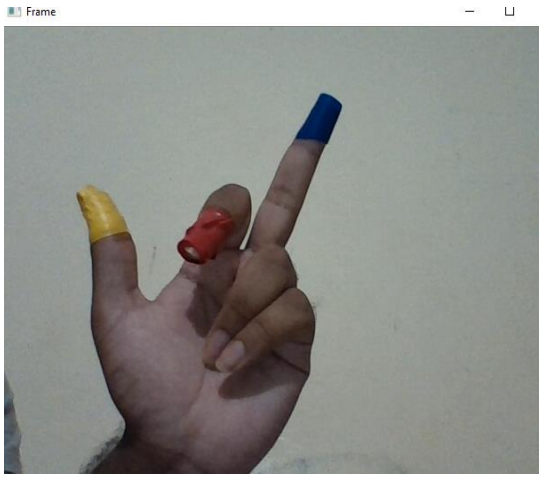


Fig 5.3.9 Scroll Down

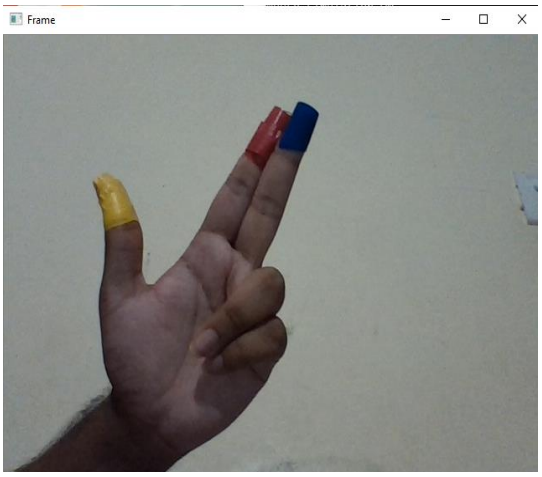


Fig 5.3.10 Left Click

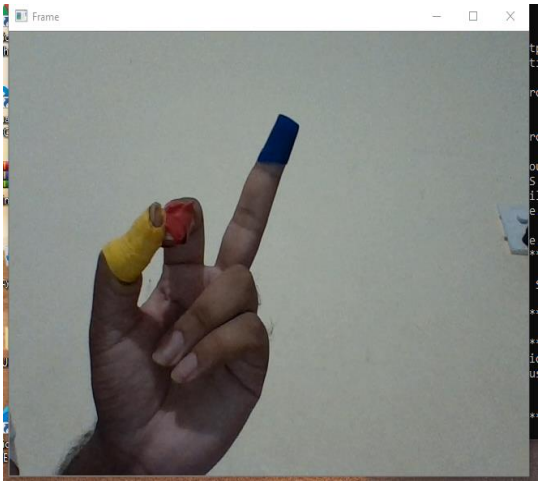


Fig 5.3.11 Right Click

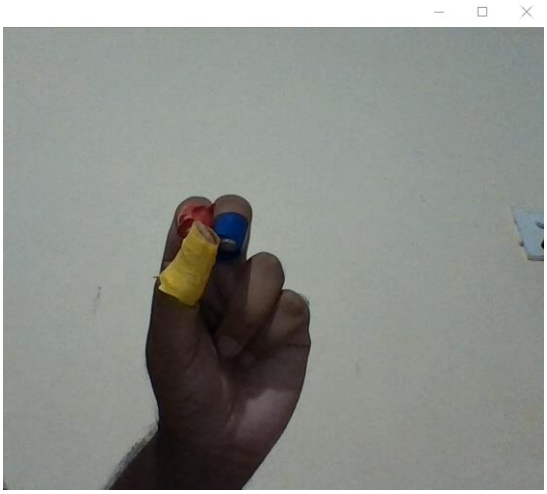


Fig 5.3.12 Select/Drag

Various gesture positions of the hand with three colors of the user by which he can use the variations to uniquely identify the hand gestures to perform actions accordingly.

5.3.5 Mouse Events

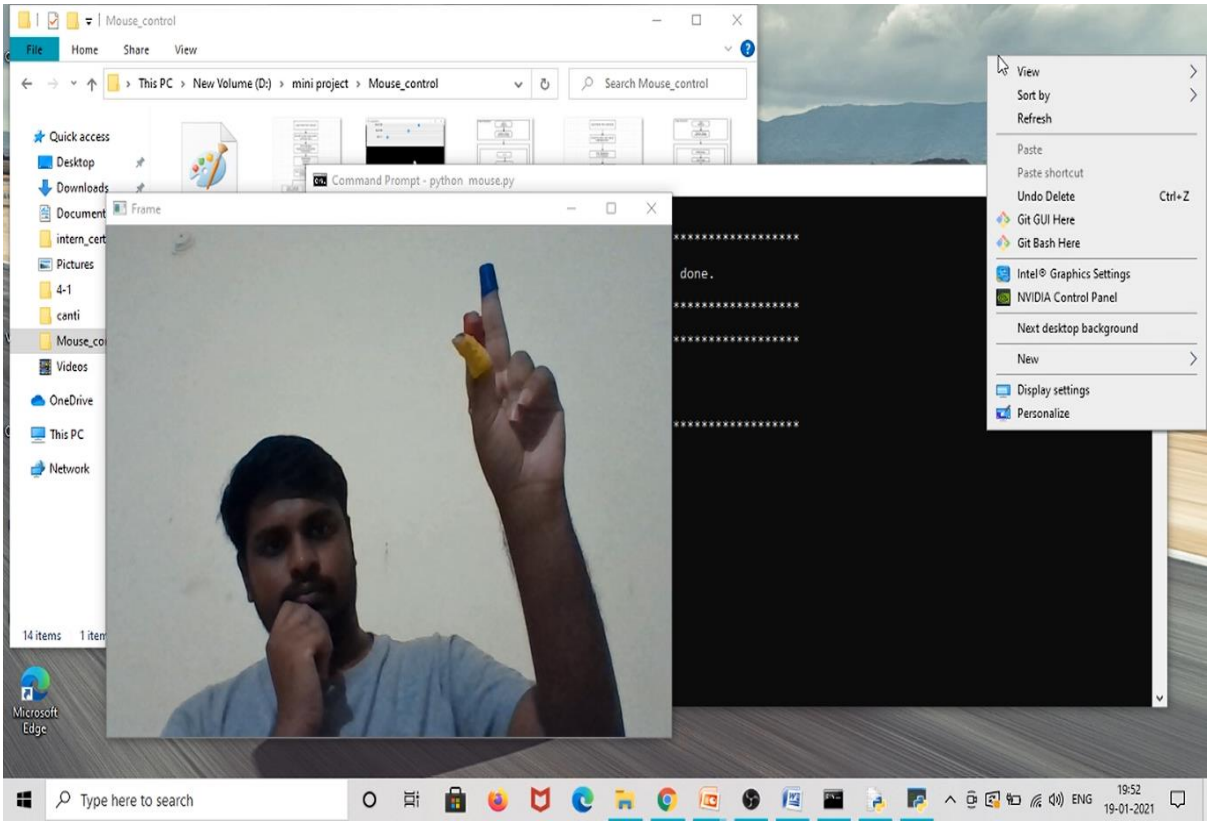


Fig 5.3.13 Right Click Event

The user image showing various hand positions and gestures and related mouse events and movements that are performed by the mouse. These are set accordingly in the code written and can be changed if required.

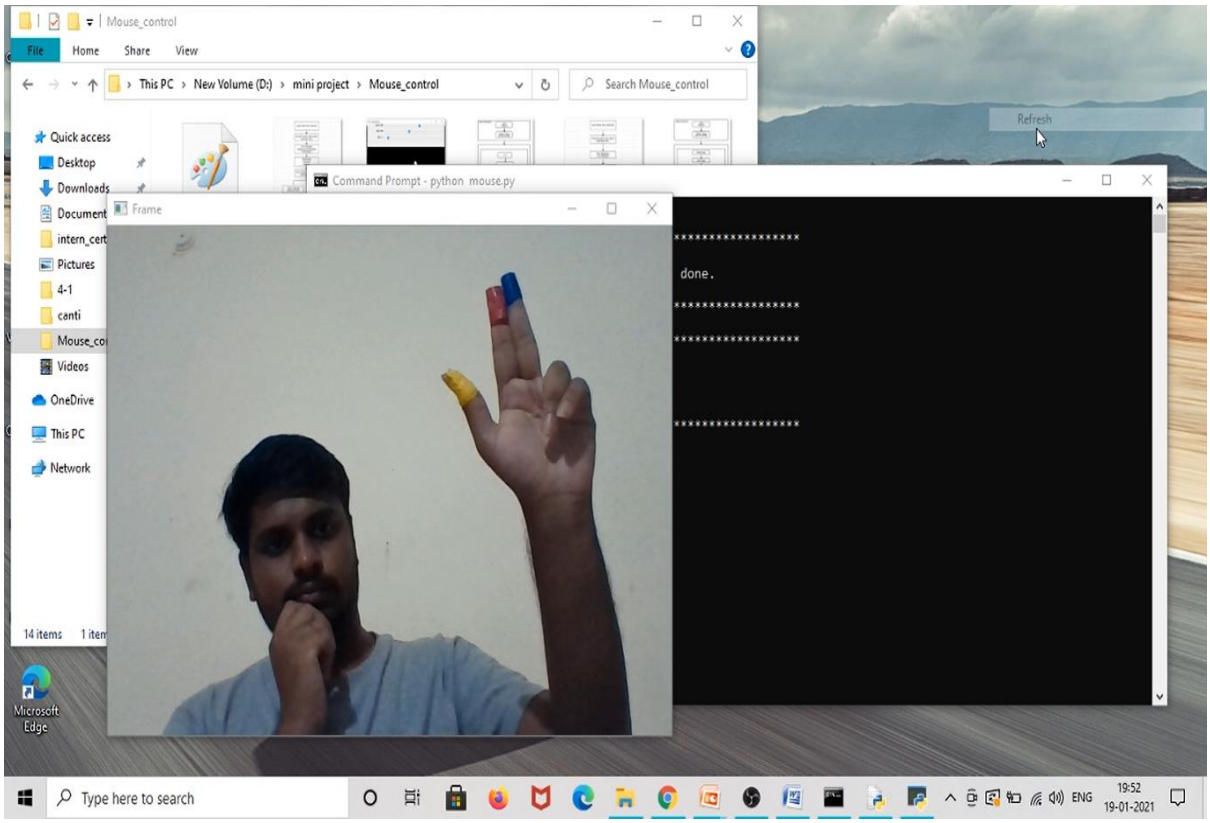


Fig 5.3.12 Left Click Event

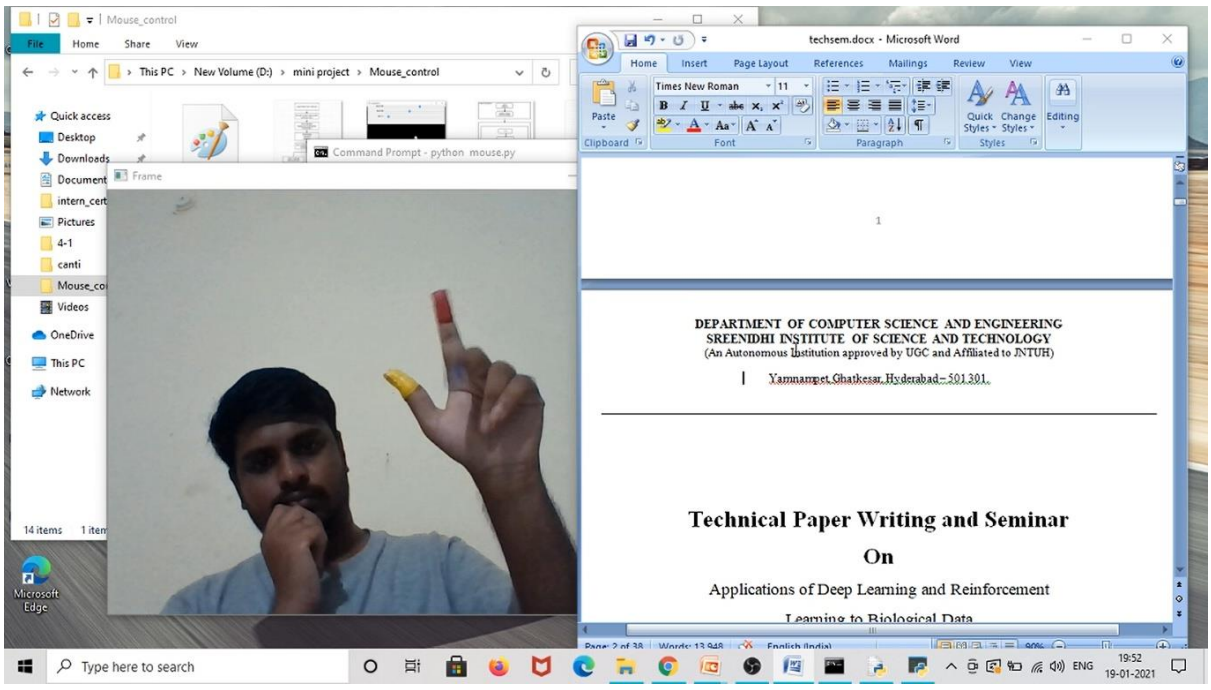


Fig 5.3.15 Before Scroll Up Event

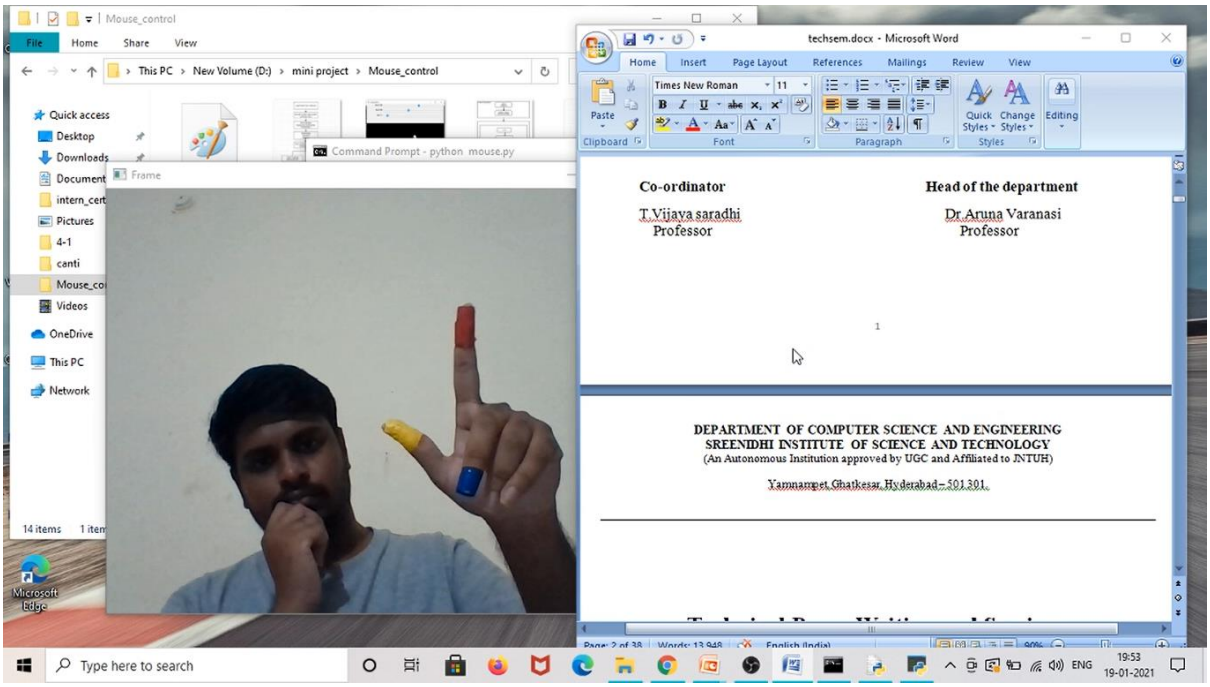


Fig 5.3.16 After Scroll Up Event

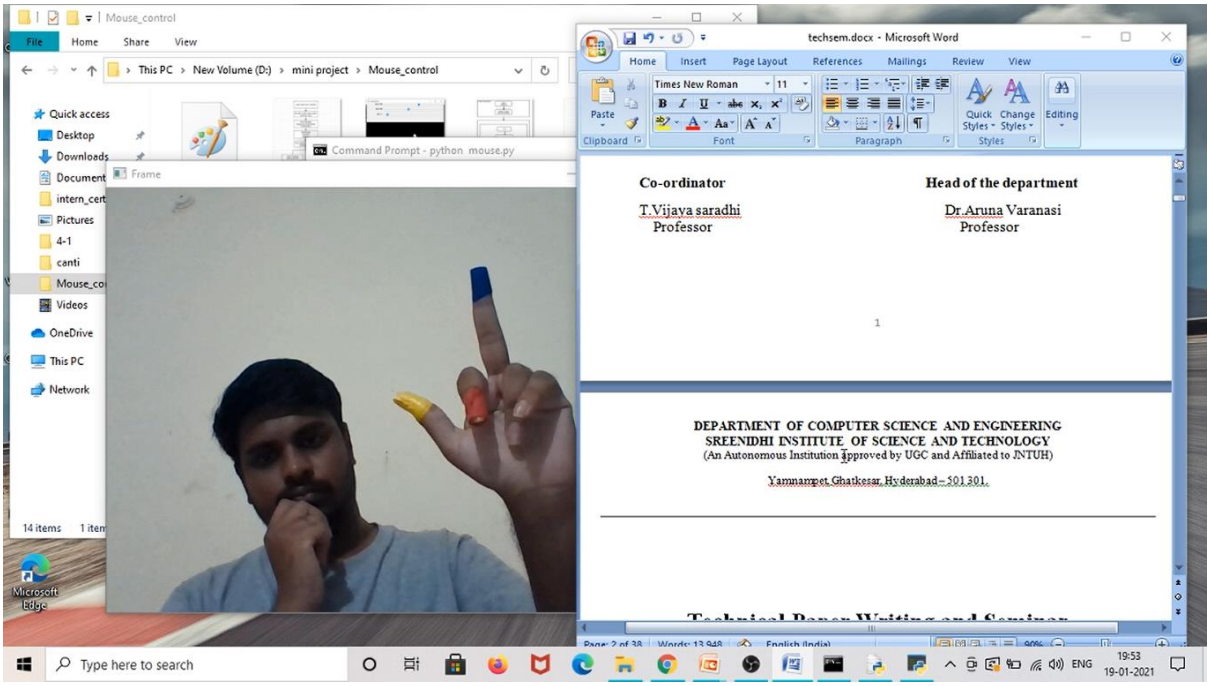


Fig 5.3.17 Before Scroll Down Event

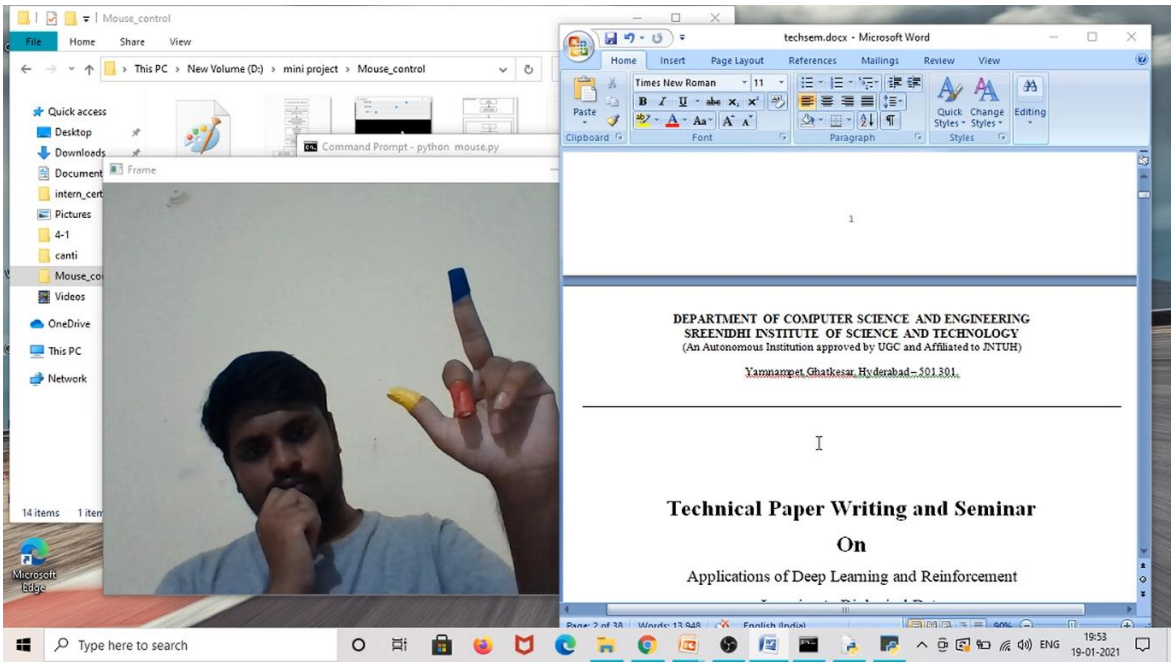


Fig 5.3.18 After Scroll Down Event

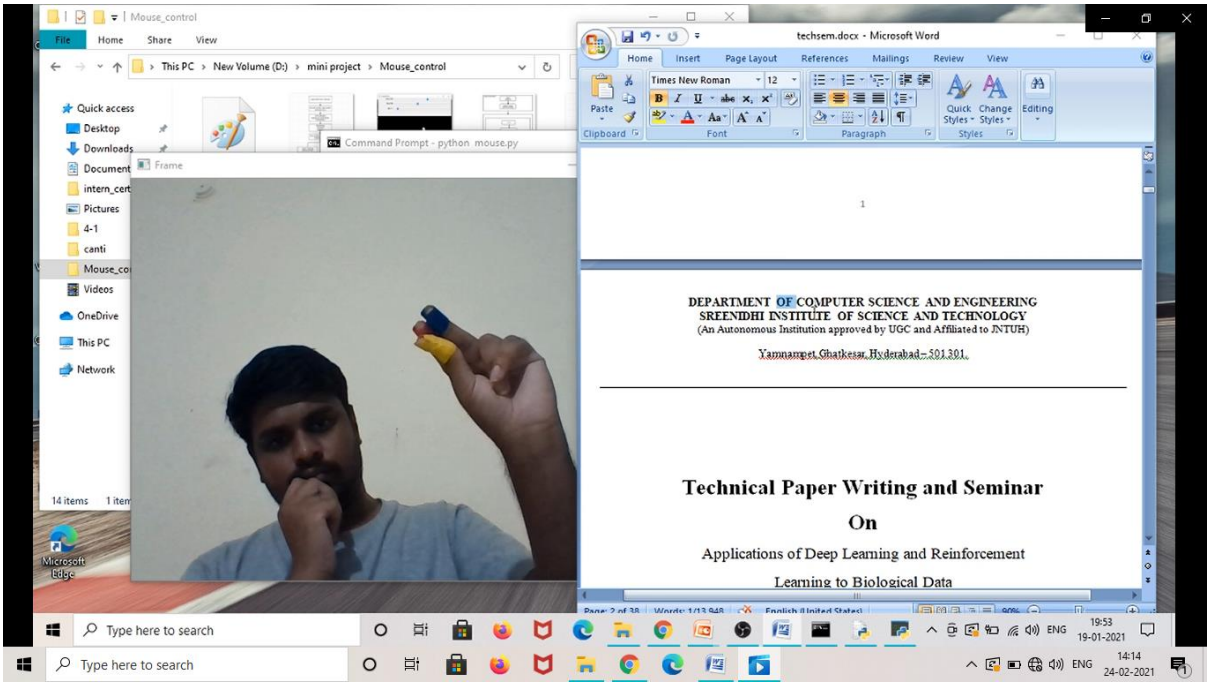


Fig 5.3.19 Drag/Select Event

6. TESTING

6.1 Types of Testing

We have implemented Integration testing and Regression testing methodologies for testing the working of this project. In the Integration testing we tested the code for each module and then integrated them based on the hierarchy to top and tested the final code to verify it passes all of the test cases. As we were doing Integration testing, we identified few bugs in final code and then we modified the code and retested it again and again until it satisfied all test cases.

6.2 List of Test Cases

S.NO	Description	Cursor Status	Test Case Result
1	Test Case with only blue and red colors	No movement of cursor or Action	PASS
2	Test Case with only yellow and red colors	Only Cursor Movement No Actions performed	PASS
3	Test Case with only yellow and blue colors	Only Cursor Movement No Actions performed	PASS
4	Test Case with only red color	No movement of cursor or Action	PASS
5	Test Case with only blue red color	No movement of cursor or Action	PASS
6	Test Case with only yellow color	Only Cursor Movement No Actions performed	PASS
7	Test Case with all the three colors	Free cursor Movements and Actions are performed	PASS

Table 6.2 List of Test Cases

7. CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

Hand gesture tracking mouse control is developed using python language and OpenCv library. Using this model the system can perform all the mouse events and mouse movements without the need of the actual mouse. The system can control the Cursor by recognizing the hand of the user. And for every cursor function or mouse event a unique hand gesture is provided.

This developed model has the potential to replace the actual mouse but because of few constraints or drawbacks it cannot completely replace the mouse. If those drawbacks are tackled then it can completely replace the mouse. There are different tools for gesture recognition. We have used multiple functions from computer vision library and developed a model by tackling few limitations that are present in previous object tracking models.

7.2 Future Work

The problem with this model is that it works well only in a room with good lighting. If the lighting in the room is not good then its performance also decreases. Only because of this reason it cannot completely replace an actual mouse, as people generally use laptops in the outdoor environment as well where the lighting is not good. Therefore further improvements can be made to use it in the bad lighting environments as well. And improvements can also be made such that the three color bands are not required.

It can be further developed with AR technology for more human-machine interaction like in video games etc. More Gestures can be developed where each unique gesture can open certain applications.

8. BIBLIOGRAPHY

[1] M. Shetty, C. A. Daniel, M. K. Bhatkar and O. P. Lopes, "Virtual Mouse Using Object Tracking," 2020 5th International Conference on Communication and Electronics Systems *(ICCES)*, Coimbatore, India, 2020.

[2] S. R. Chowdhury, S. Pathak and M. D. A. Praveena, "Gesture Recognition Based Virtual Mouse and Keyboard," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 585-589.

[3] K. H. Shibly, S. Kumar Dey, M. A. Islam and S. Iftekhar Showrav, "Design and Development of Hand Gesture Based Virtual Mouse," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology *(ICASERT)*, Dhaka, Bangladesh, 2019.

[4] R. S. Baştuğ, B. Yeşilkaya, M. Unay and A. Akan, "Virtual Mouse Control by Webcam for the Disabled," 2018 Medical Technologies National Congress (TIPTEKNO), Magusa, 2018, pp. 1-4.

[5] Khare, S. G. Krishna and S. K. Sanisetty, "Cursor Control Using Eye Ball Movement," 2019 Fifth International Conference on Science Technology Engineering and Mathematics *(ICONSTEM)*, Chennai, India, 2019.

[6] 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, Bangalore, India, 2014.

[7] K. S. Varun, I. Puneeth and T. P. Jacob, "Virtual Mouse Implementation using Open CV," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019.

APPENDIX-A: Python Modules

About PyAutoGui:

PyAutoGUI is an external gui library which supports python scripts. This library has functions which are used to control keyboard events and mouse events. Generally this library is used in the python scripts to automate the keyboard and mouse actions/events. The pyAutoGui API is developed in such a way that it is easy to use by anyone. This pyAutoGui library is supported by or it can be used on macOS, Windows and Linux systems, and compiles/runs on Python 2 and 3 versions.

PyAutoGUI features:
Controlling the mouse events and actions. Automated clicking and typing in various basic applications.
Automated keystroke sending to some applications like in filling a form.
Capture screen in the form of screenshots i.e. the function of print screen button on keyboard,
Open an application in a window, move it, select it, resize it, maximize or minimize it.
Show Dialogue boxes on the screen during the automation process i.e. when the model or program is running for better user interaction.

About Opencv (cv2):

OpenCV or cv2 is a library which can be used along with python language or scripts. This library is developed to solve computer vision problems.

- Python is considered as a programming language initiated and developed by **Guido van Rossum.** And it became very popular in a very short period of time. The main reason for its popularity is its simplicity readability and reusability.

- Python is generally slower than other famous languages like c and c++ but python can be easily converted into c++ using python wrappers. Because of this features it can provide you with the best of both the worlds i.e. faster compilation and easy code writing and readability.
- Opencv has all the essential functions required for image processing. It uses another library called **Numpy,** which is a famous library for all the numerical calculations. The arrays that are produced from opencv functions are converted into numpy arrays for easy numerical calculations.

APPENDIX-B: UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a modeling language developed to universally understand the implementations of the software. It is used to visualize, construct, specify, develop and document the artifacts of a software model or a software system. UML is not just for the developers of the model or software but the general users can also use them to understand the working of the system

It is generally used in developing process of software lifecycles, algorithm implementations, development methods etc. And by developing project reports or software documents using UML, it will be easy in implementing future projects or software as the knowledge about previous projects and the mistakes in previous projects can be easily analyzed from the previous reports as they are developed using UML. In this way refining of the software development process can be done to achieve good results.

UML is not considered as a programming language. But there are some tools which are used to perform reverse and forward engineering. Hence a UML model can be converted into a programming language like java and Code in a programming language can be converted into UML models. The UML is not considered as formal language which is basically intended in proving a theorem. There are few languages available in market for the above stated purpose but it is very difficult to understand those languages.

The conceptual model of Unified modeling language has generally three elements. They are:

- Building blocks
- Rules for those building blocks
- Common mechanisms

The building blocks of uml are:

- diagrams
- Thing
- Relationships

Behavioral Diagrams

UML's five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. It shows how the system behaves and interacts with itself and other entities (users, other systems).

A behavioral diagram shows how the system works 'in motion', that is how the system interacts with external entities and users, how it responds to input or event and what constraints it operates under.

Different behavioral Diagrams are:

- Activity Diagram
- State machine Diagram
- Usecase Diagram
- Sequence Diagram

Structural Diagrams

Structure diagrams depict the static structure of the elements in your system. i.e., how one object relates to another. It shows the things in the system – classes, objects, packages or modules, physical nodes, components, and interfaces.

Different Structural Diagrams are:

- Class Diagram
- Object Diagram
- Deployment Diagram
- Component Diagram

D22_CSE-D.docx

ORIGINALITY REPORT

13%	9%	5%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sreenidhi International School Student Paper	4%
2	Monali Shetty, Christina A. Daniel, Manthan K. Bhatkar, Ofrin P. Lopes. "Virtual Mouse Using Object Tracking", 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020 Publication	2%
3	www.coursehero.com Internet Source	2%
4	Submitted to Engineers Australia Student Paper	1%
5	Submitted to KYUNG HEE UNIVERSITY Student Paper	1%
6	Submitted to University of Witwatersrand Student Paper	1%
7	ieeexplore.ieee.org Internet Source	<1%

www.slideshare.net

8	Internet Source	<1%
9	Brand Fortner, Theodore E. Meyer. "Number by Colors", Springer Science and Business Media LLC, 1997 Publication	<1%
10	Submitted to University of Leeds Student Paper	<1%
11	www.solovatsoft.com Internet Source	<1%
12	www.comp.nus.edu.sg Internet Source	<1%
13	"Proceedings of International Conference on Advances in Computer Engineering and Communication Systems", Springer Science and Business Media LLC, 2021 Publication	<1%
14	Submitted to Whitireia Community Polytechnic Student Paper	<1%
15	idoc.pub Internet Source	<1%

Exclude quotes	On	Exclude matches	< 15 words
Exclude bibliography	On		