**ABSTRACT**

*This project is a mouse simulation system which performs all the functions performed by your mouse corresponding to your hand movements and gestures. Simply speaking, a camera captures your video and depending on your hand gestures, you can move the cursor and perform left click, right click, drag, select and scroll up and down. The predefined gestures make use of only three fingers marked by different colors.*

*You are watching a movie with your friends on a laptop and one of the guys gets a call. You must get off your place to pause the movie. You are giving a presentation on a projector and need to switch between applications. You must move across the whole stage to the podium to use your mouse. How better would it be if you could control your mouse from wherever you were? Well, we have a solution!*

*The project is essentially a program which applies image processing,* *retrieves necessary data* *and implements it to the mouse interface of the computer according to predefined notions. The code is written on Python. It uses of the cross-platform image processing module OpenCV and implements the mouse actions using Python specific library PyAutoGUI. Thus, in addition to a webcam (which almost all laptops are already loaded with) a computer needs to be pre-equipped with the following packages for the project to run such as:*

*1. Python 2.7 interpreter*

*2. OpenCV*

*3. Numpy*

*4. PyAutoGUI*

*Video captures by the webcam is processed and only the three colored finger tips are extracted. Their centers are calculated using method of moments and depending upon their relative positions it is decided that what action is to be performed.*

Table of Contents

[LIST OF TABLES v](#_Toc497663230)

[LIST OF FIGURES vi](#_Toc497663231)

[1. Introduction 1](#_Toc497663232)

[2. Motivation 1](#_Toc497663233)

[2.1 Aim 1](#_Toc497663234)

[2.2 Objective 2](#_Toc497663235)

[3. Methodology 2](#_Toc497663236)

[3.1 Framework Architecture 3](#_Toc497663237)

[3.2 Implementation 3](#_Toc497663238)

[4. Conclusion 3](#_Toc497663239)

[References 4](#_Toc497663241)

LIST OF FIGURES

Figure ‎2.1 Architecture

Figure 2.2 Video capture and cursor movement

# Figure 2.3 Hand gesture guide for mouse control

# Introduction

The project “Mouse control using Hand Gestures” is developed aiming to better the process of human-computer interaction. It aims to provide the user a better understanding of the system and to let them use alternate ways of interacting with the computer for a task.

The task here is to control the mouse even from a distance just by using hand gestures. It uses a program in python and various libraries such as PyAutoGUI, Numpy and image processing module OpenCV to read a video feed which identifies the users’ fingers represented by three different colors and track their movements. It retrieves necessary data and implements it to the mouse interface of the computer according to predefined notions.

The project can be useful for various professional and non-professional presentations. It can also be used at home by users for recreational purposes like while watching movies or playing games.

# Motivation

**2.1 Aim**

The project’s primary aim is to improve the scope of human and computer interaction by developing an effective alternative way of controlling the mouse pointer and its various functions such as left click, right click, scroll up, scroll down and selection. It helps user interact with the computer from a considerable distance without any issue and efficiently without actually touching the mouse. It also decreases the hardware requirement for the interaction by eliminating the necessity of a mouse. All the user needs is a web camera (which is mostly present in all laptops these days) which can record real-time videos.

**2.2 Objective**

The main objectives of the project are as follows: -

a. Obtain input video feed

b. Retrieve useful data from the image to be used as input

c. Filter the image and identify different colors.

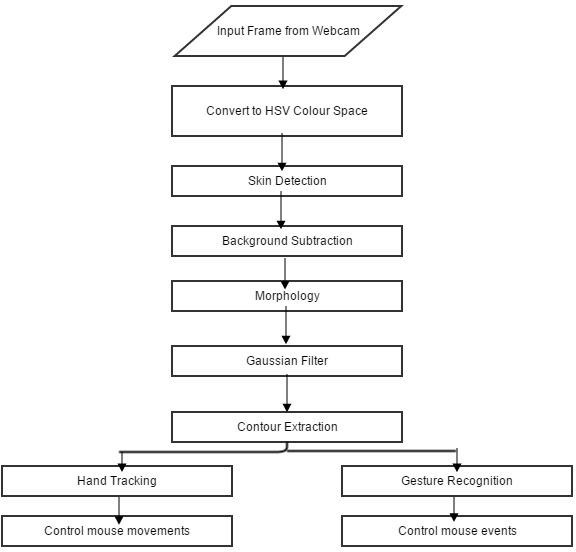
d. Track the movement of colors in the video frame.

e. Implement it to the mouse interface of the computer according to predefined notions for mouse pointer control.

# Methodology

* 1. **Framework Architecture**

The algorithm for the entire system is shown in Figure below.

 Figure 2.1 Architecture

In order to reduce the effects of illumination, the image can be converted to chrominance color space which is less sensitive to illumination changes. The HSV color space was chosen since it was found by to be the best color space for skin detection. The next step would be to use a method that would differentiate selected color pixels from non-color pixels in the image (color detection). Background subtraction was then performed to remove the face and other skin color objects in the background. Morphology Opening operation (erosion followed by dilation) was then applied to efficiently remove noise. A Gaussian filter was applied to smooth the image and give better edge detection. Edge detection was then performed to get the hand contour in the frame. Using the hand contour, the tip of the index finger was found and used for hand tracking and controlling the mouse movements. The contour of the hand was also used for gesture recognition. The system can be broken down in four main components, which are:

1. Color detection
2. Color Contour Extraction
3. Hand Tracking
4. Gesture Recognition
5. Cursor Control

**3.2 Implementation**

The first thing that we do is convert the captured video into HSV format. Now the user gets to calibrate the color ranges for three of his fingers individually. This is done by calling the calibrateColor() function thrice right at the beginning of the program The user has an option to use the default settings as well. Depending on the calibrations, only the three fingertips are extracted from the video, one by one, using the cv2.inRange() function. In order to remove noise in the video feed, we apply a two-step morphism i.e. erosion and dilation. The noise filtered image referred to as mask in the program is then sent for locating the centers.

Location of each of the three centers involves:

* Finding contours in the mask relevant to that color range.
* Discarding contours of irrelevant areas using area filters.
* Finding the largest contour amongst the remaining ones and applying method of moments to find its center.

Then comes the step for defining position of cursor on the screen. The thumb, with yellow color is responsible for position of the cursor. The following techniques have been used in this end:

* Generally, the webcams we use captures video at a resolution of 640x480 pixels.

Suppose this frame was linearly mapped to the 1920x1080 pixel display screen. If we have a right-handed user, he would find it uncomfortable to access the left edge of the screen as compared to the right edge. Also, accessing the bottom portion of the screen would build stress at the wrist.

We realized that instead of mapping the whole video frame to the screen, we could

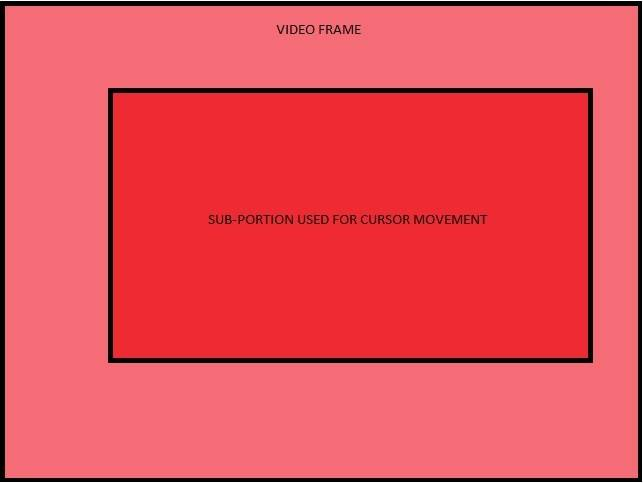


Figure 2.2 Video capture and cursor movement

rather consider a rectangular sub portion more biased towards right (considering right-handed user) and upper parts of the frame in order to improve comfort. This sub portion which measures 480x270 pixels is then linearly mapped to the screen with a scaling factor of 4.

* Due to noise captured by the webcam and vibrations in the hand, the centers keep vibrating around a mean position. On scaling up, these vibrations create a lot of problem with the accuracy of cursor position. To reduce the shakiness in cursor, we make use of differential position allocation for the cursor. We compare the new center with the previous position of the cursor. If difference is less than 5 pixels, it is usually due to noise. Thus, the new cursor position is inclined more towards the previous one. However, a larger difference in previous position and new center is considered as voluntary movement and the new cursor position is set close to the new center. For details, go through the setCursorPosition() function in the code.
* Now the three centers are sent for deciding what action needs to be performed depending on their relative positions. This is done in the chooseAction() function in the code.
* Depending upon its output, the performAction() function carries out either of the following using the PyAutoGUI library:
* free cursor movement
* left click
* right click
* drag/select
* scroll up
* scroll down

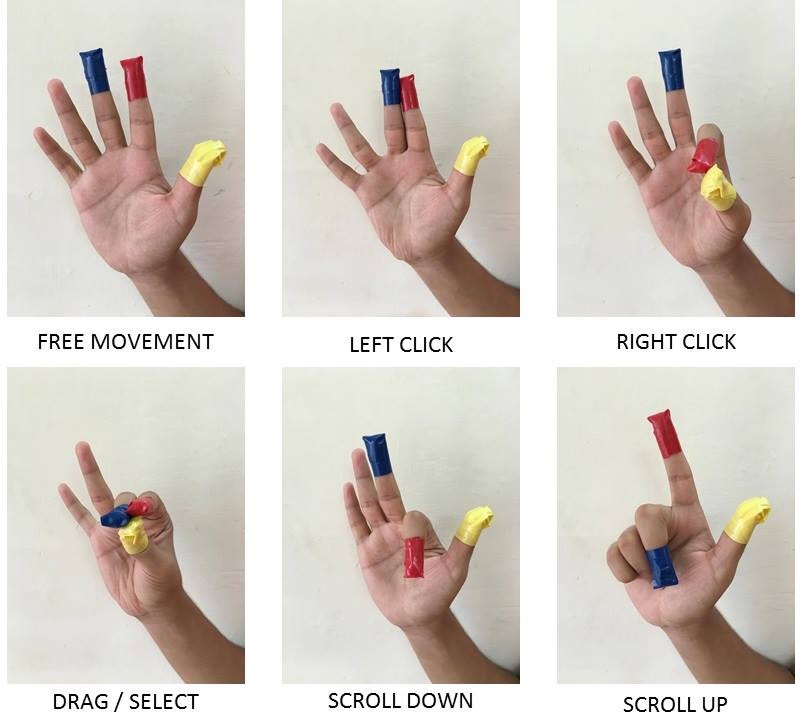


Figure 2.2 Hand gestures and their functions

# PROJECT SPECIFICATIONS

## Software Specifications:

1. 64-bit Operating System: Windows 8 or Higher

2. OpenCV 2.4.9 needs to be installed prior to running.

3. Windows Administrator permissions are needed for some parts of the program to function properly.

## Hardware Specifications:

1. A Webcam

## Environment Specifications:

* 1. A clear white background
  2. There should be no other objects (especially red, blue, yellow colored) in front of the webcam other than on the fingers.

# Conclusion

# The vision based cursor control using hand gesture system was developed in Python language, using the OpenCV library. The system could control the movement of a Cursor by tracking the users’ hand. Cursor functions were performed by using different hand gestures. The system has the potential of being a viable replacement for the computer mouse, however due to the constraints encountered; it cannot completely replace the computer mouse. The major constraint of the system is that it must be operated in a well-lit room. This is the main reason why the system cannot completely replace the computer mouse, since it is very common for computers to be used in outdoor environments with poor lighting condition. The accuracy of the hand gesture recognition could have been improved, if the Template Matching hand gesture recognition method was used with a machine learning classifier. This would have taken a lot longer to implement, but the accuracy of the gesture recognition could have been improved. It was very difficult to control the cursor for precise cursor movements, since the cursor was very unstable. The stability of the cursor control could have been improved if a Kalman filter was incorporated in the design. The Kalman filter also requires a considerable amount of time to implement and due to time constraints, it was not implemented. All the operations which were intended to be performed using various gestures were completed with satisfactory results.

# References

|  |  |
| --- | --- |
| [1] | Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj,” Mouse Control using a Web Camera based on Color Detection”,IJCTT,vol.9, Mar 2014. |
| [2] | Angel, Neethu.P.S,”Real Time Static & Dynamic Hand Gesture Recognition”, International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013. |
| [3-5] | Chen-Chiung Hsieh and Dung-Hua Liou,” A Real Time Hand Gesture Recognition System Using Motion History Image”icsps, 2010. |
| [6] | Ashwini M. Patil1, Sneha U. Dudhane1 , Monika B. Gandhi1 ,” Cursor Control System Using Hand Gesture Recognition”, International journal of advanced research in computer and communication engineering. Vol 2, issue5, may 2013 |
| [7] | Amayeh, Gholamreza, George Bebis, Ali Erol, and Mircea Nicolescu. "Hand-based verification and identification using palm–finger segmentation and fusion."Computer 113, no. 4 (2009). |
| [8] | Angelopoulo, E., Rana Molana, and Kostas Daniilidis. "Multispectral skin color modeling." In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 2, pp. II-635. IEEE, 2001. |