***Research Report***

*A*

***Virtual Mouse***

*submitted*

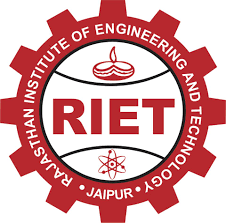
*in partial fulfillment*

*for the award of the degree*

***Bachelor of Technology***

***in Department of Computer Science & Engineering***

**(with specialization in Computer Science & Engineering)**

****

Supervisor: Submitted By:

Mr. Mukesh Choudhary Yuganter Sharma,   
Department of CSE Ajey Jangid,   
 Dayana Singh,

Hariom Singh,

Parul Yadav

**Department of Computer Science & Engineering**

**Rajasthan Institute of Engineering and Technology, Jaipur Rajasthan Technical University, Kota**

# ACKNOWLEDGEMENT

I am over helmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

We pay thank to MR. MUKESH CHOUDHARY who has given guidance and a light to us during the major project.

We acknowledge here out debt to those who contributed significantly to one or more steps.We take full responsibility for any remaining sins of omission and commission.

# INDEX

1. Introduction
2. Motivation
3. Problem Description
4. Issues and Challenges
5. Project Design
6. Capture Video
   1. Calibrate color
   2. Remove noise and define function
   3. find contours and draw centroids
   4. final steps
7. Project Specifications
8. Conclusion
9. Further Work
10. References

1.Introduction

Computer technology has tremendously grown over the past decade and has become a necessary part of everyday live. The primary computer accessory for Human Computer Interaction (HCI) is the mouse. The mouse is not suitable for HCI in some real life situations, such as with Human Robot Interaction (HRI). There have been many researches on alternative methods to the computer mouse for HCI. The most natural and intuitive technique for HCI, that is a viable replacement for the computer mouse is with the use of hand gestures. This project is therefore aimed at investigating and developing a Computer Control (CC) system using hand gestures.

Most laptops today are equipped with webcams, which have recently been used insecurity applications utilizing face recognition. In order to harness the full potential of a webcam, it can be used for vision based CC, which would effectively eliminate the need for a computer mouse or mouse pad. The usefulness of a webcam can also be greatly extended to other HCI application such as a sign language database or motion controller. Over the past decades there have been significant advancements in HCI technologies for gaming purposes, such as the Microsoft Kinect and Nintendo Wii. These gaming technologies provide a more natural and interactive means of playing videogames. Motion controls is the future of gaming and it have tremendously boosted the sales of video games, such as the Nintendo Wii which sold over 50 million consoles within a year of itsrelease.HCI using hand gestures is very intuitive and effective for one to one interaction with computers and it provides a Natural User Interface (NUI). There has been extensive research towards novel devices and techniques for cursor control using hand gestures. Besides HCI, hand gesture recognition is also used in sign language recognition, which makes hand gesture recognition even more significant.

### 2. MOTIVATION

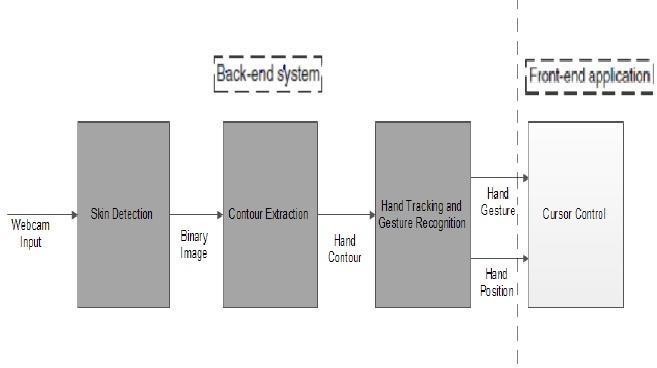
We had chosen this project with an interest of learning the direct interaction of humans with the consumer electronic devices . This takes the user experience to a whole new level. The gesture control technology would reduce our dependence on the age old peripheral devices hence it would reduce the overall complexity of the system. Initially this technology was considered in the field of gaming (like Xbox Kinect), but the application of motion/gesture control technology would be more diverse if we apply it to our other electronics like computers ,televisions, etc., for our day to day purposes like scrolling , selecting, clicking etc.

Our primary objective in doing this project was to build a device inspired from Leapmotion. It is a device which recognizes hand gestures and can be used to virtually control a computer. In short, it provides a virtual screen with which we can interact with the computer. But the required hardware for making a device on these lines was not feasible, in terms of budget and time frame provided. So, we decided to build an introductory software implementation of the device which would eventually act as a virtual mouse.

### 3. PROBLEM DESCRIPTION

There are generally two approaches for hand gesture recognition, which are hardware based, where the user must wear a device, and the other is vision based which uses image processing techniques with inputs from a camera. The proposed system is vision based, which uses image processing techniques and inputs from a computer webcam. Vision based gesture recognition tracking and gesture recognition. The input frame would be captured from the webcam and systems are generally broken down into four stages, skin detection, hand contour extraction, hand the skin region would be detected using skin detection. The hand contour would then be found and used for hand tracking and gesture recognition. Hand tracking would be used to navigate the computer cursor and hand gestures would be used to perform mouse functions such as right click, left click, scroll up and scroll down. The scope of the project would therefore be to design a vision based CC system, which can perform the mouse function previously stated.

In short, the Flowchart for our project will be as follows:



### 4. ISSUESAND CHALLENGES

The first challenge was to correctly detect the hand with a webcam. We needed a Computer

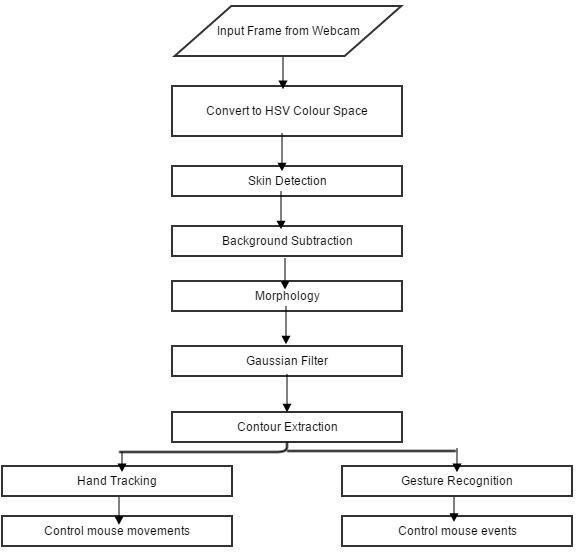
Vision library for this purpose. Many are available but we decided to go ahead with OpenCVas it is the most popular and has been ported to many languages and is supported on many operating systems from Android to Windows. It has a good library collection of standard image processing functions. Then we had to first setup OpenCV]on our IDE(Visual Studio). That was a very tedious task. We also had to learn some basic usage of OpenCV. For which we referred to many tutorials on the web. After learning OpenCV, We had to learn about the skin detection techniques and image processing techniques like Background Subtraction, Image Smoothening, Noise Removal and Reduction.

Now, after detecting the hand correctly and mapping the gestures, we had to learn to use the Windows API in order tune the software with the Metro UI. For learning it, we built some basic applications based on it. So, in short, there was a steep learning curve.

As, high-end cameras and sensors are very costly we decided to go with a simple webcam. So, we decided to optimize our software and its functionality in order the drawback of using a simple webcam. Also, for testing our project we had a primary requirement of having a white background with no visible part except our palm.

## 5.PROJECT DESIGN

In this section the strategies and methods used in the design and development of the vision based CC system will be explained. The algorithm for the entire system is shown in Figure below.



In order to reduce the effects of illumination, the image can be converted to chrominance colour space which is less sensitive to illumination changes. The HSV colour space was chosen since it was found by to be the best colour space for skin detection. The next step would be to use a method that would differentiate skin pixels from non-skin pixels in the image (skin detection). Background subtraction was then performed to remove the face and other skin colour 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, thus in the Methodology the method used in each component of the system will be explained separately

This section is separated into the following subsections:

1.Skin Detection

2. Hand Contour Extraction

3. Hand Tracking

4.Gesture Recognition

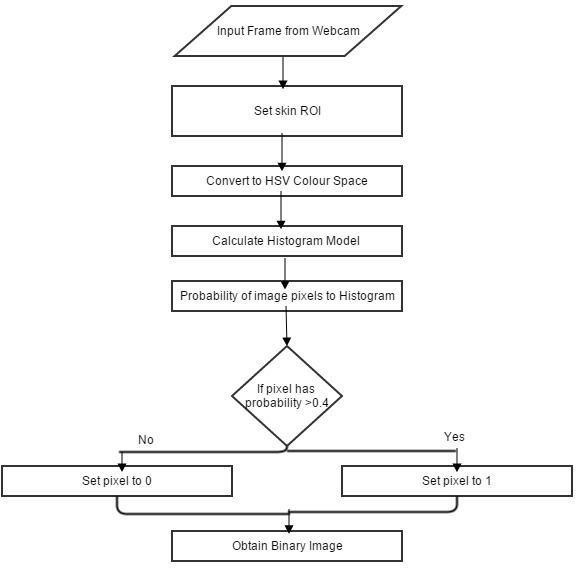
5. Cursor Control

### Skin Detection

Skin detection can be defined as detecting the skin colour pixels in an image. It is a fundamental step a wide range of image processing application such as face detection, hand tracking and hand gesture recognition. Skin detection using colour information has recently gained a lot of attention, since it is computationally effective and provides robust information against scaling, rotation and partial occlusion. Skin detection using colour information can be a challenging task, since skin appearance in images is affected by illumination, camera characteristics, background and ethnicity.

In order to reduce the effects of illumination, the image can be converted to a chrominance colour space, which is less sensitive to illumination changes. A chrominance colour space is one where the intensity information (luminance), is separated from the colour information (chromaticity). In the proposed method**,** the HSV colour space was used with the Histogrambased skin detection method. The HSV colour space has three channels, Hue (H), Saturation(S) and Value (V). The H and S channels hold the colour information, while the V channel holds the intensity information. The input image from the webcam would be in the RGB colour space, thus it would have to be converted to the HSV colour space using the conversion Formulae*.* The Histogram-based skin detection method proposed by uses 32 bins H and S histograms to achieve skin detection. Using a small skin region, the colour of this region is converted to a chrominance colour space. A 32 bin histogram for the region is then found and is used as the histogram model.

Each pixel in the image is then evaluated on how much probability it has to a histogram model. This method is also called Histogram Back Projection. Back projection can be defined as recording how well pixels or patches of pixels fit the distribution of pixels in a histogram model. The result would be a grayscale image (back projected image), where the intensity indicates the likelihood that the pixel is a skin colour pixel. This method is adaptive since the histogram model is obtained from the users ski, under the preset lighting condition.



Algorithm for Skin Detection

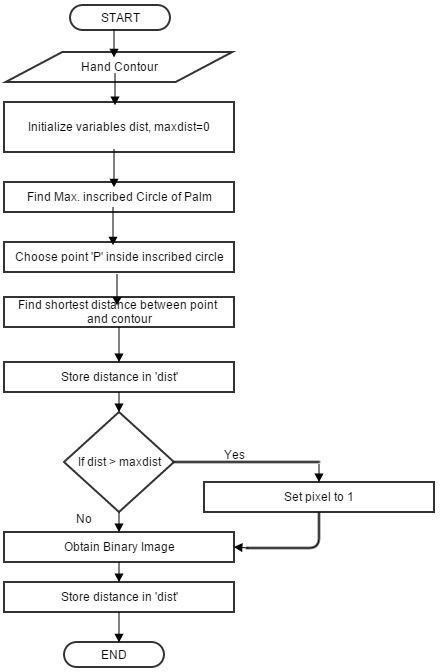
### Hand Contour Extraction

After obtaining the skin segmented binary image, the next step is to perform edge detection to obtain the hand contour in the image. There are several edge detection methods such as, Laplacian edge detection, canny edge detection and border finding The OpenCV[1] function cvFindContours() uses a order finding edge detection method to find the contours in the image. The major advantage of the border finding edge detection method , is that all the contours found in the image is stored in an array. This means that we can analyse each contour in the image individually, to determine the hand contour. The Canny and Laplacian edge detectors are able to find the contours in the image, but do not give us access to each individual contour. For this reason the border finding edge detection method was used in the proposed design.

In the contour extraction process, we are interested in extracting the hand contour so that shape analysis can be done on it to determine the hand gesture. Figure below shows the result when edge detection was applied to the skin segmented binary image. It can be seen that besides the hand contour, there are lots of small contours in the image. These small contours can be considered as noise and must be ignored. The assumption was made that the hand contour is the largest contour thereby ignoring all the noise contours in the image. This assumption can be void, if the face contour is larger than the hand contour. To solve this problem, the face region must be eliminated from the frame. The assumption was made that the hand is the only moving object in the image and the face remains relatively stationary compared to the hand. This means that background subtraction can be applied to remove the stationary pixels in the image, including the face region. This is implemented in the OpenCV[1] function named “BackgroundSubtractorMOG2”.

#### Hand Tracking

The movement of the cursor was controlled by the tip of the index finger. In order to identify the tip of the index finger, the centre of the palm must first be found. The method used for finding the hand centre was adopted from and it has the advantage of being simple and easy to implement. The algorithm for the method is shown in the flow chart of Figure below. The shortest distance between each point inside the inscribed circle to the contour was measured and the point with the largest distance was recorded as the hand centre. The distance between the hand centre and the hand contour was taken as the radius of the hand. The hand centre was calculated for each successive frame and using the hand centre, the tip of the index finger would be identified and used for hand tracking. The method used for identifying the index and the other fingers are described in the following subsection. The results for hand tracking would be demonstrated in Figure in the Results and Analysis section.



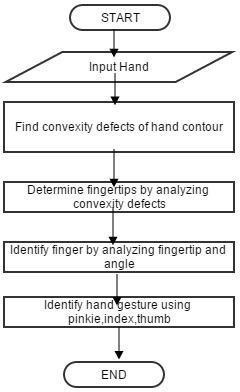
Algorithm for Hand Tracking

### GestureRecognition

The gesture recognition method used in the proposed design is a combination of two methods, the method proposed by Yeo and the method proposed by Balazs. The algorithm for the proposed gesture recognition method is described in the flow chart of Figure below. It can be seen from Figure above that the convexity defects for the hand contour must firstly be calculated. The convexity defects for the hand contour was calculated using the OpenCV inbuilt function

“cvConvexityDefects”. The parameters of the convexity defects (start point, end point and depth point) are stored in a sequence of arrays. After the convexity defects are obtained, there are two main steps for gesture recognition:

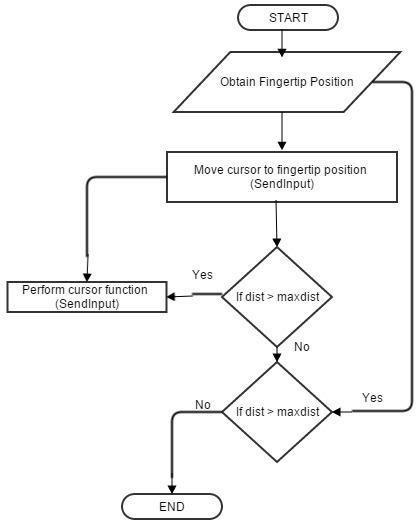
i. Finger Tip Identification ii. Number Of Fingers



### Cursor Control

Once the hand gestures are recognized, it will be a simple matter of mapping different hand gestures to specific mouse functions. It turns out that controlling the computer cursor, in the C/C++ programming language is relatively easy. By including the User.lib library into the program, the “SendInput” function will allow control of the computer cursor. Instructions on how to properly use this function, was obtained from the Microsoft Developers Network MSDN

[2] website. This function is only available for the Windows 2000 Professional operating system or later. This introduces a new limitation on the system, such that it can only be used on newer versions of the Windows operating system. The algorithm for the cursor control is shown in Figure below.



Algorithm for Cursor Control

Following table shows the Operations Performed depending upon the number of fingers detected:

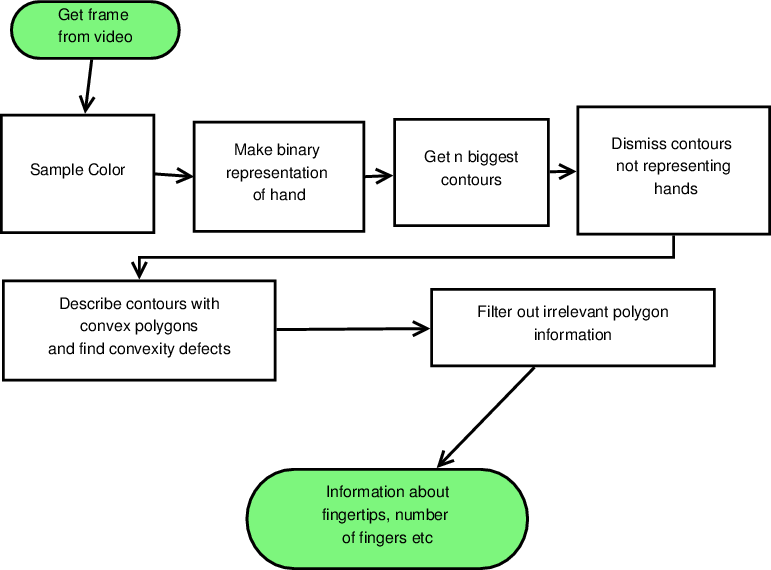
|  |  |
| --- | --- |
| **Number of Fingertips Detected** | **OperationsPerformed** |
| One | Move Cursor |
| Two | Left Click |
| Three | Right Click |
| Four | Start Button |
| Five | My Computer |

Starting with the position of the index fingertip, the cursor is moved to the fingertip position. This is done using the “SendInput” function to control the cursor movement. The next step would be to determine if a hand gesture was performed. If a hand gesture as performed, the

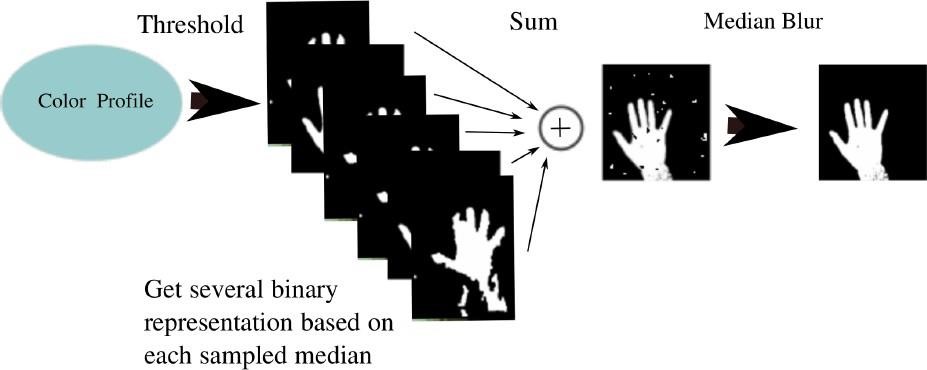
“SendInput” function is again used to control the cursor function. If there is no change in fingertip position, the loop is exited and it would be started again, when a change in fingertip position is detected.

### Summary

Below is a summary flowchart representation of the program:



The hand tracking is based on color recognition. The program is therefore initialized by sampling color from the hand. The hand is then extracted from the background by using a threshold using the sampled color profile. Each color in the profile produces a binary image which in turn are all summed together. A nonlinear median filter is then applied to get a smooth and noise free binary representation of the hand.



# 6. CAPTURE VIDEO

In this first we capture video from the web camera in which we take one frame from capture video that frame is converted in hsv for the color variation

HSV- means Hue-Saturation-Value, where the Hue is the color. Saturation is the greyness , so that a Saturation value near 0 means it is dull or grey looking. And Value is the brightness of the pixel.

In this we use HSV in opencv to filter color from that frame. HSV values of the color which we want to filter out.

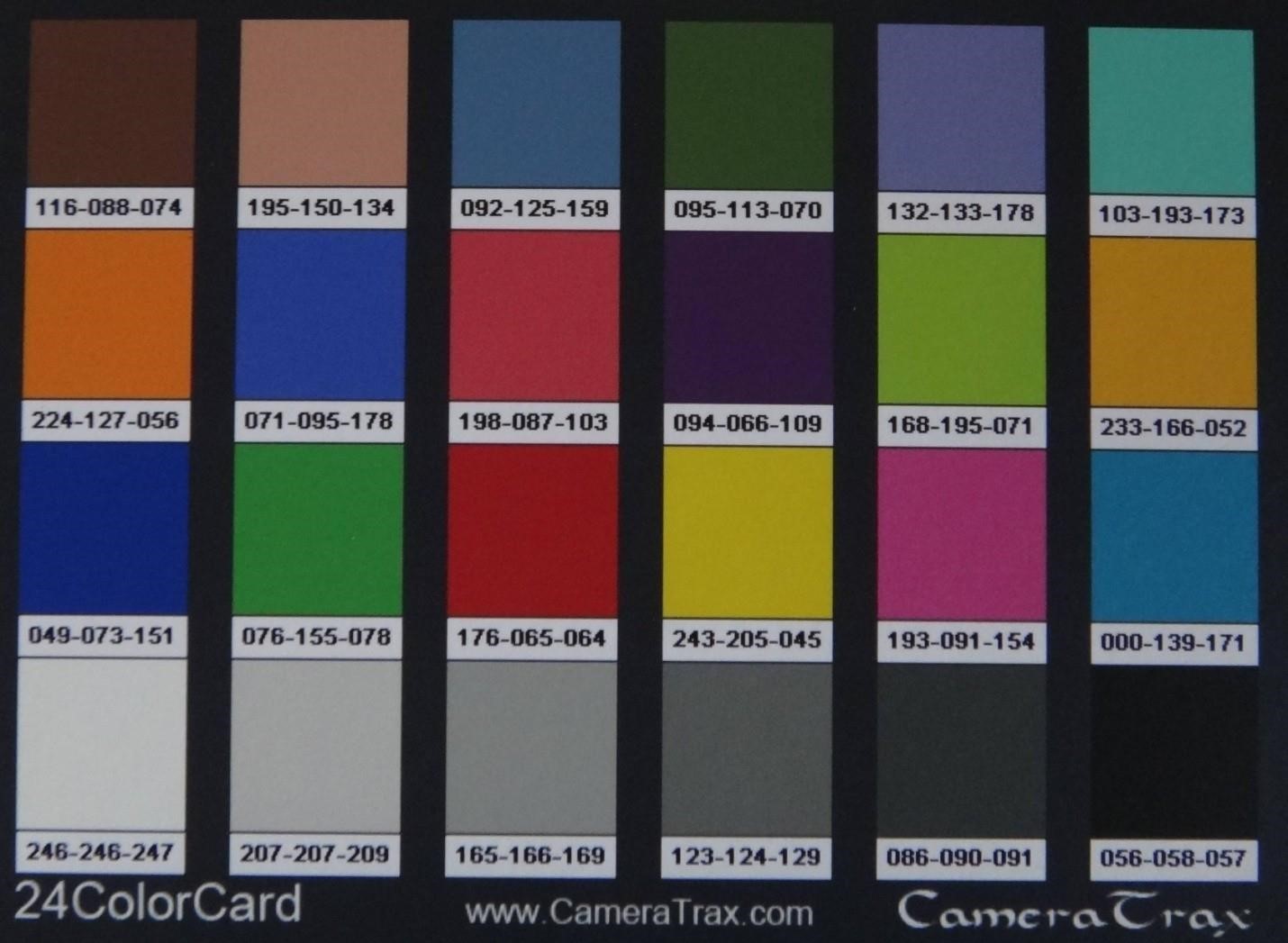
SO if we want to capture video from webcam of laptop we use (0) value inwhile loop or if we want from external webcam the value will be (1) in while loop

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

break

vid.release() cv2.destroyAllWindows()

## 6.1 Calibrate Color-(Calibrate the color range)



**ColorCalibration** is one of the segments that we addressed for a biological discipline. Apart from object detection and segmentation, itrequires color correction for a color-based

analysis.

Now the user gets to calibrate the colour 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.

### 6.2 Remove Noise & Define Functions in the Video Feed(unwanted noise)-

In this we remove the noise that is in background for better color filter

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

-ltered image referred to as mask in the program is then sent for locating the centres.

####  Morphological operations

* In short:A set of operations that process images based on shapes. Morphological operations apply a *structuring element* to an input image and generate an output image.
* The most basic morphological operations are: Erosion and Dilation. They have a wide array of uses, i.e. :
  + Removing noise
  + Isolation of individual elements and joining disparate elements in an image.
  + Finding of intensity bumps or holes in an image

## ->Dilation

* This operations consists of convolving an image A with some kernel ( B), which can have any shape or size, usually a square or circle.
* The kernel B has a defined *anchor point*, usually being the center of the kernel.
* As the kernel B is scanned over the image, we compute the maximal pixel value overlapped by B and replace the image pixel in the anchor point position with that maximal value.As you can deduce, this maximizing operation causes bright regions within an image to "grow" (therefore the name *dilation*).
* The dilatation operation is: dst(x,y)=max(x′,y′):element(x′,y′)≠0src(x+x′,y+y′)



->EROSION-

* This operation is the sister of dilation. It computes a local minimum over the area of given kernel.
* As the kernel B is scanned over the image, we compute the minimal pixel value overlapped by B and replace the image pixel under the anchor point with that minimal value.
* The erosion operation is: dst(x,y)=min(x′,y′):element(x′,y′)≠0src(x+x′,y+y′)

 

# (remove noise and function)-

# cv2.inRange function is used to filter out a particular color from the frame

# The result then undergoes morphosis i.e. erosion and dilation

# Resultant frame is returned as mask

def makeMask(hsv\_frame, color\_Range):

mask = cv2.inRange( hsv\_frame, color\_Range[0],

color\_Range[1])

## # Morphosis next

eroded = cv2.erode( mask, kernel, iterations=1) dilated = cv2.dilate( eroded, kernel, iterations=1)

## 6.3 Find Contours & Draw Centroids

Location of each of the three centres involves:

Finding contours in the mask relevant to that colour range.

Discarding contours of irrelevant areas using area -lters.

Finding the largest contour amongst the remaining ones and applying the method of moments to -find its center

# Contours on the mask are detected.. Only those lying in the previously set area

# range are filtered out and the centroid of the largest of these is drawn and returned

6.4 Final Steps (Set Position, Choose & Perform

## Actions)

Then comes the step for de-ning the position of the cursor on the screen.

The thumb, with yellow colour, is responsible for the 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 -nd 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 realised that instead of mapping the whole video frame to the screen, we could rather consider a rectangular sub-portion more biased towards the 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.

cursor[0] = 4\*(yp[0]-110) cursor[1] = 4\*(yp[1]-120)

Due to noise captured by the webcam and vibrations in the hand, the centres keep vibrating around a mean position. On scaling up, these vibrations create a lot of problem with the accuracy of the cursor position. To reduce the shakiness in the cursor, we make use of dierential position allocation for the cursor. We compare the new centre with the previous position of the cursor. If the dierence 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 dierence in the previous position and the new centre is considered as voluntary movement and the new cursor position is set close to the new centre. For details, go through the setCursorPosition() function in the code.

### Code-

This function takes as input the center of yellow region (yc) and the previous cursor position (pyp).

The new cursor position is calculated in such a way that the mean deviation for desired steady state is reduced def setCursorPos( yc, pyp): yp = np.zeros(2)

if abs(yc[0]-pyp[0])<5 and abs(yc[1]-pyp[1])<5:

yp[0] = yc[0] + .7\*(pyp[0]-yc[0]) yp[1] = yc[1] + .7\*(pyp[1]-yc[1])

else:

yp[0] = yc[0] + .1\*(pyp[0]-yc[0]) yp[1] = yc[1] + .1\*(pyp[1]-yc[1]) returnyp

Now the three centres 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:

PyAutoGUI library:

1. free cursor movement
2. left-click
3. right-click
4. drag/select
5. scroll up
6. scroll down