Design of Intangible Interface for Human Computer Interaction- Hand Gestures Controlled Mouse

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

Bachelor of Technology

In

Industrial Design

By

Alisha Pradhan (111ID0601)

Under the supervision of

Prof. B. B. V. L. Deepak

Department of Industrial Design

National Institute of Technology, Rourkela

Orissa - 769008

April 2015

Declaration

I hereby declare that this thesis is my own work and effort. Throughout this documentation wherever contributions of others are involved, every endeavour was made to acknowledge this clearly with due reference to literature. This work is being submitted for meeting the partial fulfilment for the Degree of **Bachelor of Technology** in **Industrial Design** at **National Institute of Technology, Rourkela** for the academic session 2011 – 2015.

Alisha Pradhan (111ID0601)

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA 769008, INDIA

Certificate of Approval

*This is to certify that the thesis entitled "**Design of hand gestures for mouseless computer control** " being submitted by **Ms. Alisha Pradhan (111ID0601)** in partial fulfillment of the requirement for the award of degree **BACHELOR OF TECHNOLOGY** in **INDUSTRIAL DESIGN** at **National Institute of Technology, Rourkela** is an original work carried out by her under my supervision and guidance. The matter embodied in the thesis has not been submitted to any other university/ Institute for award of any other degree.*

Date: 10/04/2015

Prof. B.B.V.L. Deepak

Assistant Professor

Department of Industrial Design

National Institute of Technology, Rourkela

Acknowledgement

After the completion of my B. Tech. thesis work, words are not enough to express my feelings about all those who helped me to reach my goal. Feeling above this is my indebtedness to The ALMIGHTY for providing me this moment in life.

First and foremost, I take this opportunity to express my deep regards and heartfelt gratitude to my project guide **Prof. B.B.V.L Deepak** for his inspiring guidance, cooperation and timely suggestions in carrying out my project successfully.

I am extremely grateful to **Prof. B.B. Biswal, Head of Industrial Design Department**, for providing all the facilities towards the completion of my B. Tech. thesis work.

I would also like to extend my gratitude to our friends and post graduate students of this department who have always encouraged and supported me in doing my work. I would like to thank all the staff members of **Department of Industrial Design** who have been very cooperative with me. I would also like to thank the authors of various research articles and books that I referred to during the course of the project.

Last but not least, the blend of pleasure & great satisfaction, what I feel is to convey my gratitude & indebtedness to all those who have directly or indirectly contributed to successful completion of the project.

Finally, I wish to thank my beloved **parents** and **friends** for their immense personal and emotional support throughout my work.

Alisha Pradhan

ABSTRACT

The current generation is the touch generation. In this generation, everything works with the touch of a finger, be it our cell phones, computers or any day to day appliance. But the so called 'touch generation' has reached its ultimate state and is soon likely to decline. The generation of 'intangible interfaces' is just at its dawn and is soon going to take over the touch generation. The problem statement for this project is to improve the human computer interaction with the use of intangible interface. The aim is to control all operations of a mouse without the actual use of mouse.This would allow the user to navigate his laptop while sitting far away from it. This is achieved by recognition of hand gestures. A computer vision application is made which captures images of the hand gestures from the laptop's web camera, processes it, and performs the required output.Several algorithms of computer vision including image segmentation, feature extraction have been used. The basic functions of a mouse i.e. left click, right click and double click using hand gestures are performed in this project. The main aim is to improve the interaction of human with the computer i.e. to make it more natural but keeping the cost factor in mind. The purpose is to develop an application that must be easy as well as economic to use.

Contents

Abbreviations

HCI - Human computer interaction

HMI - Human Machine Interaction

CV - Computer vision

HSV - Hue, Saturation, Value

BGR - Blue, Green, Red

List of Figures

List of Tables

1. Introduction

The current generation is the touch generation. In this generation, everything works with the touch of a finger; be it our cell phones, computers or any day to day appliance. But the so called "touch generation" has reached its ultimate state and is soon likely to decline. The generation of “intangible interfaces" is just at its dawn and is soon going to take over the touch generation. The problem statement for this project is to improve the Human Computer Interaction (HCI) with the use of intangible interface.

Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. It is the study of how people design, implement, and use interactive computer systems and how computers affect individuals, organizations, and society. HCI is an interdisciplinary area involving different fields such as computer science (application design and engineering of human interfaces), psychology (the application of theories of cognitive processes and the empirical analysis of user behaviour), sociology and anthropology (interactions between technology, work, and organization) and industrial design (interactive products). Research in HCI primarily deals in the design, implementation, and assessment of new interfaces for improving the interaction between humans and machines.

Hand Gesture Recognition is a natural way of Human Machine Interaction (HMI) and currently is one of the main areas of research. It provides a platform for multimodal interaction. This mode of HMI has the potential to make conventional input devices such as mouse, keyboards and even touch-screens redundant. Gesture Recognition is an important concept in HCI. It involves computer vision and image processing. Computer vision is as important aspect of HCI. It is the science and technology of machines that see and perceive. Computer vision is concerned with building artificial systems that obtain information from images or multi-dimensional data. Image processing is a method to convert an image into digital form and perform some operations on it. It includes importing the image, analyzing and manipulating the image. It is mainly used for visualization, image sharpening and

restoration, image retrieval, pattern measurement, image recognition. However the computing speed in frame processing plays a critical role in this process.

The objectives of HCI are to create usable and safe systems. It mainly aims at:

• Understanding the element that decides how individuals use technology.

• Develop tools and technology for building new systems.

• Making the work environment safe and efficient.

Interaction: The communication between the user and the system. Their interaction framework has four parts: User, Input, System and Output.

Intangible Interfaces: Key components:

Gesture Recognition: Gesture recognition is the mathematical representation of a human motion using a computing device. It enables humans to communicate with the machine (HMI) and interact naturally without any mechanical devices.

Facial recognition: It is a computer application for automatically identifying a person from a digital image or a video frame from a video source. It is typically used in security systems.

Voice recognition: "Voice recognition" can be used for both, speaker recognition (recognizing who is speaking) and speech recognition (recognizing what is being said).

Eye tracking: It is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head. An eye tracker is a device for measuring eye positions and eye movement. Eye trackers are used in research on the visual system, in psychology, in cognitive linguistics and in product design.

The above mentioned features are components of what developers refer to as a perceptual user interface (PUI).

Operating Laptop without mouse:

Presently we can control our laptop without using a mouse or a keyboard by using the Speech Recognition and On Screen Keyboard features present in Windows operating System.

- Use On-Screen Keyboard: This option sets On-Screen Keyboard to run when we log on to Windows. On-Screen Keyboard displays a visual keyboard with all the standard keys.

- Use Speech Recognition: The option allows to control the computer with your voice. With a microphone, you can speak commands that the computer will understand and respond to, as well as dictate text.

This project deals with development of a computer vision based mouse. The aim is to control mouse functioning using hand gestures. It can be used for controlling the mouse cursor and performing functions like left click, right click, double click, and moving the mouse cursor on the screen with the aim of making the human and computer interaction more intuitive. In this project, camera and computer vision technology such as image segmentation and gesture recognition to perform the various mouse functions. The existing technologies use wearable technologies or invasive technologies, thus making the technology costly or difficult to use. Here the only hardware required is the web camera of our laptop. Entire coding is done in open platform Microsoft Visual Studio. Thus the entire project is very economic in nature.

1.1 Background of the work

The present age is the age of touch or commonly called as the "touch screen generation". A touch screen can be a simple push button. The another form of a touch device is when the image of a button switch is displayed on screen, and the device is programmed to be on and off upon touching the image. Since a touch screen can detect coordinate points, it can also function as a mouse (although it does not distinguish between right and left clicks like a mouse).Touch screen devices like smartphones, camcorder, tablet computer etc. can be seen everywhere. Touch screen can also be seen in a car, automotive navigation device. Despite the interactive nature of the touch interfaces, they have their own set of limitations. Intangible user interfaces, new in the field of human computer interaction, are those which require no touch. Intangible user interfaces can make use of gesture recognition, face recognition, voice recognition, eye tracking. Intangible interfaces are highly interactive and possess a great scope in future.

1.2 Motivation

Intangible interfaces are those which require no touch. They are soon going to take over the "touch" devices. Many developments have been seen in this field of Human-Computer Interaction lately. "Mouseless", an invisible computer mouse, provides the familiarity of interaction of a physical mouse without actually requiring a real hardware mouse. The device

eliminates the requirement of having a physical mouse altogether but still provides the intuitive interaction of a physical mouse that users are familiar with. Mouseless consists of an Infrared (IR) laser beam and an Infrared camera. Mouseless depicts the potential of intangible interfaces. These interfaces can make the regular hardware based keyboard and mouse completely obsolete and drive them from the market soon. The vast scope of intangible interfaces was the key motivation for the project.

1.3 Problem Statement

Despite the interactive nature of the touch interfaces, they have their own set of limitations. The screen has to be big enough to enable proper touch of buttons. In direct sunlight, the efficiency is reduced since it becomes difficult to read the screen. Frequent touch of screens for operating, makes the screens dirty. The major disadvantage of touch devices are that one has to be within arm's reach of the device. These devices cannot be operated from a distance. The project aims developing an intangible interface for controlling mouse functions using human hand gestures. It can be used for controlling the mouse cursor and performing functions like left click, right click, double click, zooming in, zooming out, rotation. The entire system has to be economic and easy to use, thus eliminating the difficulties involved in using wearable or invasive technologies.

1.4Objective of Work

The objective of this project is to make the human computer interaction more intuitive but keeping the entire process economic. The aim in each step is to make the interaction more and more natural i.e to reduce the gap between a human mental model and the process of performing the task. The project focuses on mouse control. The aim is to design hand gestures for performing the mouse functions. The course of the work begins with the planning phase involving initial research, literature review and background study of the softwares available and methods of performing hand gesture recognition. The project begins with object detection and object tracking, followed by tracking of human hand, image segmentation and hand gesture recognition.

1) Study and apply open computer vision (CV) library.
2) Study and apply computer vision algorithms for simply gesture recognition.
3) Design and develop a simple hand gesture recognition application.
4) Test and document the results.

2. Literature Review

2.1 Computer Vision

Human vision has its own limitations. But digital technology enables us to venture beyond these limitations. Earlier digital technology provided the data, but the data had to be processed by the humans only. But today, technology has evolved and the systems are becoming more intelligent (similar to the human brain). They can perceive things and process things. Computer vision is a powerful tool which is capable of such things. It emulates human vision and performs actions based on the visual inputs. It has taken artificial intelligence to the next level[1].

Computer vision has its own set of drawbacks. Human vision is three dimensional. But the cameras, which are used in computer vision, capture and process two dimensional images. This leads to loss of information while conversion from 3D to 2D. Despite this, computer vision has emerged as a powerful tool in artificial intelligence.

2.2 Image Processing

In image processing, a digital image is the input. This image is processed for noise reduction, filtering. Example, enhancing a blurred image to get a clear photo. Digital images are also used for identification purpose using image processing.

2.3 Open CV

Open CV stands for open computer vision library. It is used for real time applications and is written in C and C++. It is available for free as open source software. Here the image is stored in a data structure called Mat. Due to this, the image data or fields are easily accessible. These fields include:

- Width of image in pixels.

- Height of image in pixels.
- Depth of image in pixels.
- Number of color per pixel.
- Number of bytes per image row.
- Size of image in bytes.

Open CV performs basic image processing and computer vision algorithms. These include smoothing function to reduce noise, erosion and dilation to remove unwanted elements,floodfill to isolate certain points of the image, canny function for edge detection etc.[1]

2.4 Hand Detection Approaches [2]:

2.4.1 Appearance based approach: Usesfingertip detection for the hand image construction. Finger tips in hands are used as natural determinant of hand posture to reconstruct the image.

2.4.2 Model based approach: It includes techniques like: using of histogram for calculating probability for skin color observation, taking Gaussian distribution for background pixels marking and subtracting the pixels from the new image to acquired gesture image.

2.4.3 Soft computing approach: It includes algorithms like, artificial neural networks, fuzzylogicand genetic algorithm.

2.5 Related Work

Many methods have been developed by several researchers for controlling the mouse movement using gesture recognition. Many of them have been developed around the Hidden Markov Model [3] .The Hidden Markov Model consists of unobserved or hidden states. The state is not directly visible, but output, dependent on the state, is visible. But the computational cost involved in this method is quite high. Pandit et al [4] in his work used wearable gloves from which the hand posture could be extracted.In Erdem&Aykut's method [5], the user simply moves the mouse shaped passive device placed on a surface within the viewing area of the camera to move the cursor on the computer screen. Viola and Jones[6] provide a novel approach to detect an object efficiently using adaboost to interpret the hand motion by palm recognition. Their method provided faster detection

rates. They used a new image representation called “integral image", which facilitated the image features to be computed very quickly. adaboost learning algorithm was used for selecting a smallnumber of critical visual features. They combined the classifiers in a “cascade” which helped in discarding the background region of the image. Thus more computational time can be spend on the object like regions. Park [7] used a simpler method, where the action of clicking of mouse was done by keeping a track of the finger tips. They made use of computer webcam and computer vision technology including image processing. Nayana&Kubakaddi [8] made use of feature extraction for counting the number of fingers displayed. They made use of image processing techniques like pre processing, segmentation followed by feature extraction. Paul et al [6,7], used the motion of the thumb (from a thumbs-up position to a fist) to mark a clicking event thumb. Finger counting and hand orientation approach was used by Dhawan&Honrao[9] for gesture control. The features like centroid, peaks detection, Euclidean distance and thumb detection was used by Panwar andMehra [9]. Dhawan and Honrao presented number of methods for segmenting an image and thresholding with and without background.Finger counting method is used here. Jinda-apiraksa et al. [10] calculated as the ratio of squared perimeter of the shape to its area. If two different hand shapes with same perimeter to area ratio exists, these two different shapes would be classified as same. Thus, it limited the number of gesture pattern that can be classified.

3. Methodology

3.1 Theoretical Analysis

3.1.1 Object detection and object tracking

Color based object detection was studied and performed. . The object here is detected on the basis of its color i.e. HSV values (Hue, Saturation, Value). Suppose we want to detect an object of red color, then we filter the red object from the background depending on its HSV values. This completes object Detection.

Steps of Object Detection:

1. Read image file. It can either be a static image or input from webcam.
2. Convert image from BGR to HSV (hue, saturation, value). This step makes filtering easier. BGR2HSV function is used for this purpose.
3. Filter the color of interest between a Min and Max threshold. The inRange function is used of this purpose.
4. InRange Function: It is used for filtering the color of interest. Parameters required are:

   src-source image i.e input array.

   dst- destination image i.e. output array.

   lowerb - lower bound array.

   upperb - upper bound array.

   dst (I) = lowerb(I)0 ≤ src(I)0 ≤ upperb(I)0
5. Perform Morphological operations. It is used to remove the noise from the image. These operations are performed on binary image. Morphological techniques probe an image with a small shape or template called structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels. It is of two types:

   erode: This function makes white space smaller.

   dilate: This function makes white space larger.

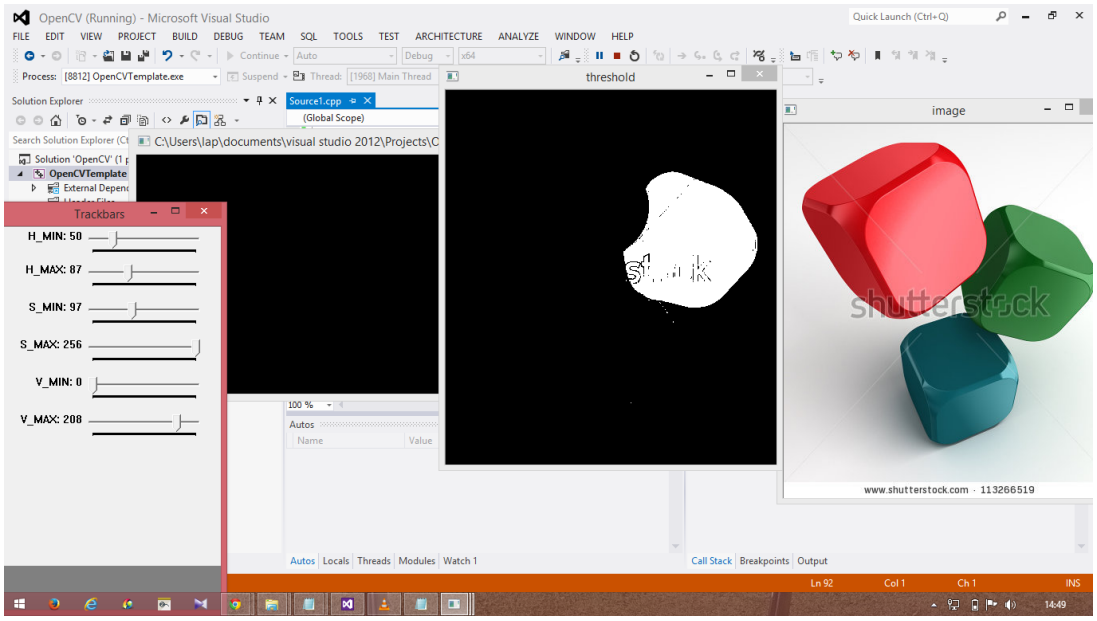


Fig 3.1 Object detection from static image

This is the first step. Here the required object is detected, but a static image is used, i.e. an image which is stored at some place in our computer. Here the input is not taken from a webcam. The following image shows object detection when input is taken from webcam i.e real time object detection. In both cases, the required object is filtered depending on its color and is then detected. The HSV values for filtering a specific colour can be noted from the trackbar and then it can be entered manually in the program of object detection. After completion of object detection, comes object tracking.

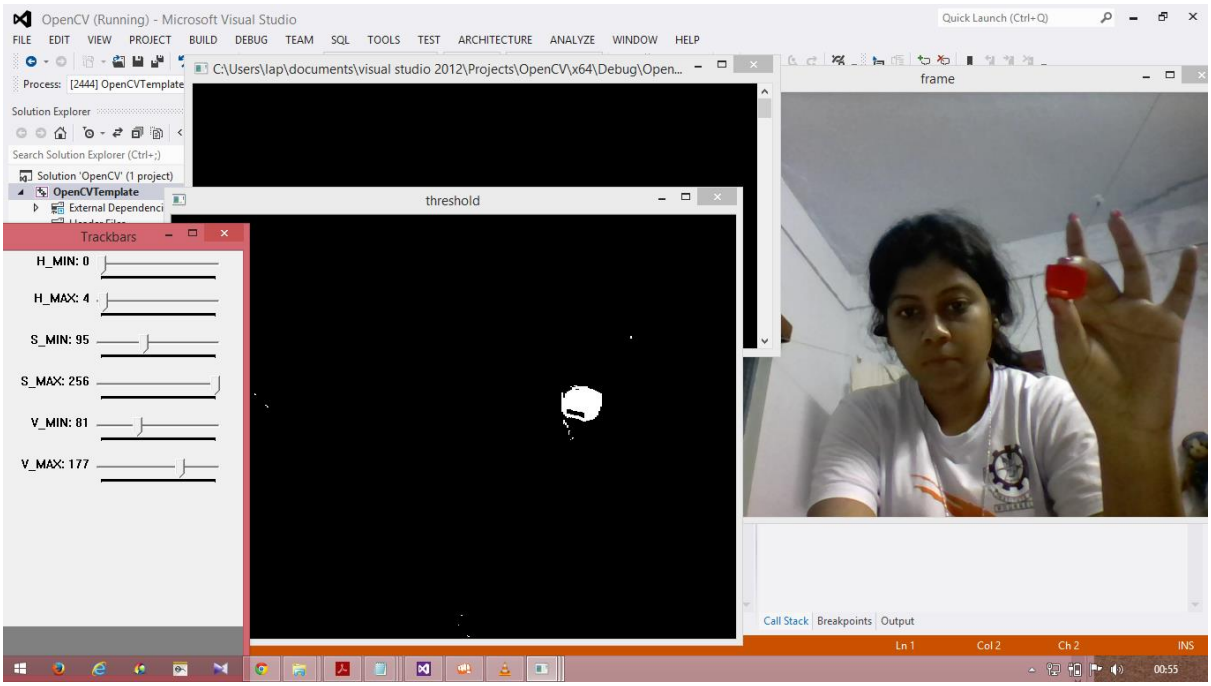


Fig 3.2 Real time coloured object detection

The HSV values for various colours were found using the trackbar. These are mentioned below:

BLUE	GREEN	RED
Hmin =75	Hmin =50	Hmin = 0
Hmax= 169	Hmax= 87	Hmax= 175
Smin=44	Smin=42	Smin=0
Smax=256	Smax=256	Smax=256
Vmin=0	Vmin=0	Vmin=0
Vmax= 256	Vmax= 256	Vmax= 256

Table 3.1 HSV values of colour

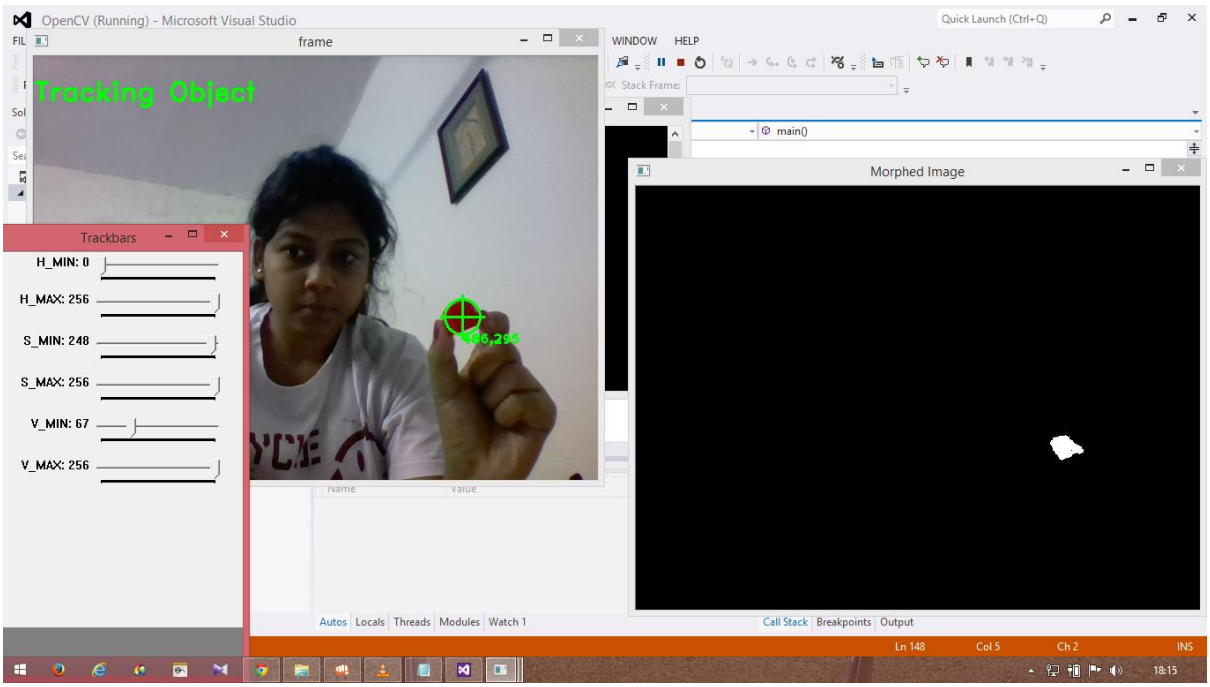


Fig 3.3 Color based real time object detection and tracking.

After completing basics of object detection and object tracking, human hand detection and hand tracking was performed.

Code snippet for object detection and tracking:

```
while(true)
{
boolbsuccess = cap.read(frame);
cvtColor(frame,hsv,CV_BGR2HSV);
inRange(hsv,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),thresh);
erode(thresh, thresh, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
dilate( thresh, thresh, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
Mat temp;
constint MAX_NUM_OBJECTS=50;
constint MIN_OBJECT_AREA = 20*20;
constint MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH/1.5;
thresh.copyTo(temp);
vector< vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPL
E );
doublerefArea = 0;
boolobjectFound = false;
if (hierarchy.size() > 0)
 {
intnumObjects = hierarchy.size();
if(numObjects<MAX_NUM_OBJECTS){
for (int index = 0; index >= 0; index = hierarchy[index][0])
{
Moments moment = moments((cv::Mat)contours[index]);
double area = moment.m00;
if(area>MIN_OBJECT_AREA&&area<MAX_OBJECT_AREA && area>refArea)
{
x = moment.m10/area;
```

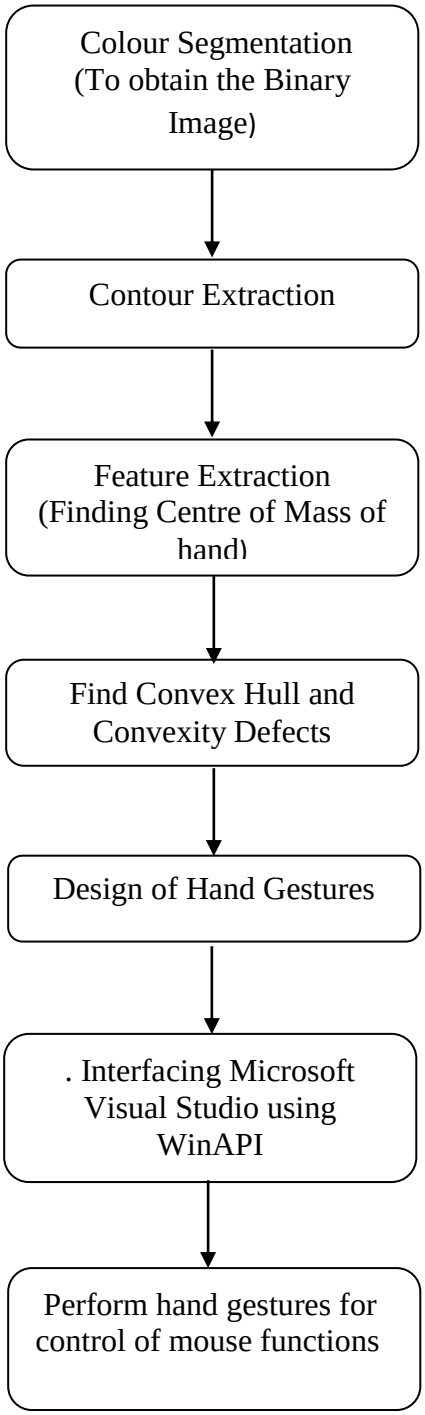
```
                                        y = moment.m01/area;
                                        objectFound = true;
                                        refArea = area;
                                }
elseobjectFound = false;
}
        if(objectFound ==true)
                {
putText(frame,"TrackingObject",Point(0,50),2,1,Scalar(0,255,0),2);
        drawObject(x,y,frame);
                }
}
}
}
```

Human hand detection:

Fig 3.4 Detection of human hand

3.2 Experimental Analysis:

Several algorithms of computer vision including image segmentation, feature extraction have been used. The methodology followed in the project is depicted in the flowchart below.



3.2.1. Image segmentation:

Image segmentation is the division of an image into regions or categories, which is done to identify similar regions in the image. This is very beneficial to the subsequent image analysis. Here color is used for segmentation. The image of the hand is read from the webcam of the laptop as frames. The area of interest being the hand, other pixels of the background need to be isolated. For this,the HSV of the hand are chosen. The HSV values for human hand differ from the HSV values of the background. Thus it can be used to filter the region of our interest. Here, the luminosity and the hand colour of the user play a key role, because the HSV values change accordingly. To accommodate a wide user range, the trackbar system has been used for filtering the hand. The HSV values can be set as per the user requirement by using the trackbars.

After filtering the image frame using the HSV values of hand, we obtain a binary image. In this image the region of the hand appears white and all other regions are black. In order to achieve this, functions available in Open CV library were used.

Algorithm:

1. Capture the hand gesture using the camera of computer.

2. Determine the range of HSV values for skin color of hand. (This particular step differs from person to person and also depends on the luminosity).

3. Convert the image from RGB color space to HSV color space. For this the cvtColor function of OpenCV is used.

4. Convert all pixels falling within the range (filter values) to white.

5. Convert all other pixels to black.

Code snippet for hand detection:

```
while(true)
{
boolbsuccess = cap.read(frame);
cvtColor(frame,hsv,CV_BGR2HSV);
inRange(hsv,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),thresh);
```

```
imshow("threshold",thresh);
                imshow("frame",frame);
waitKey(33);
}
```

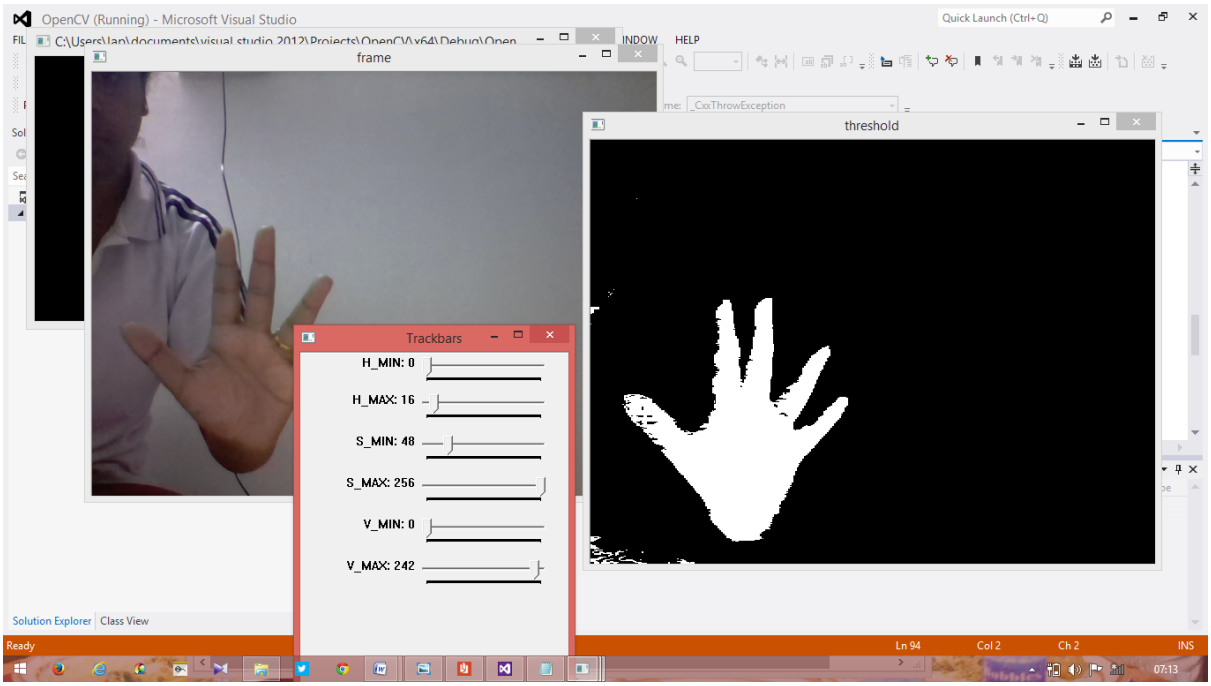


Fig 3.5 Human hand detection

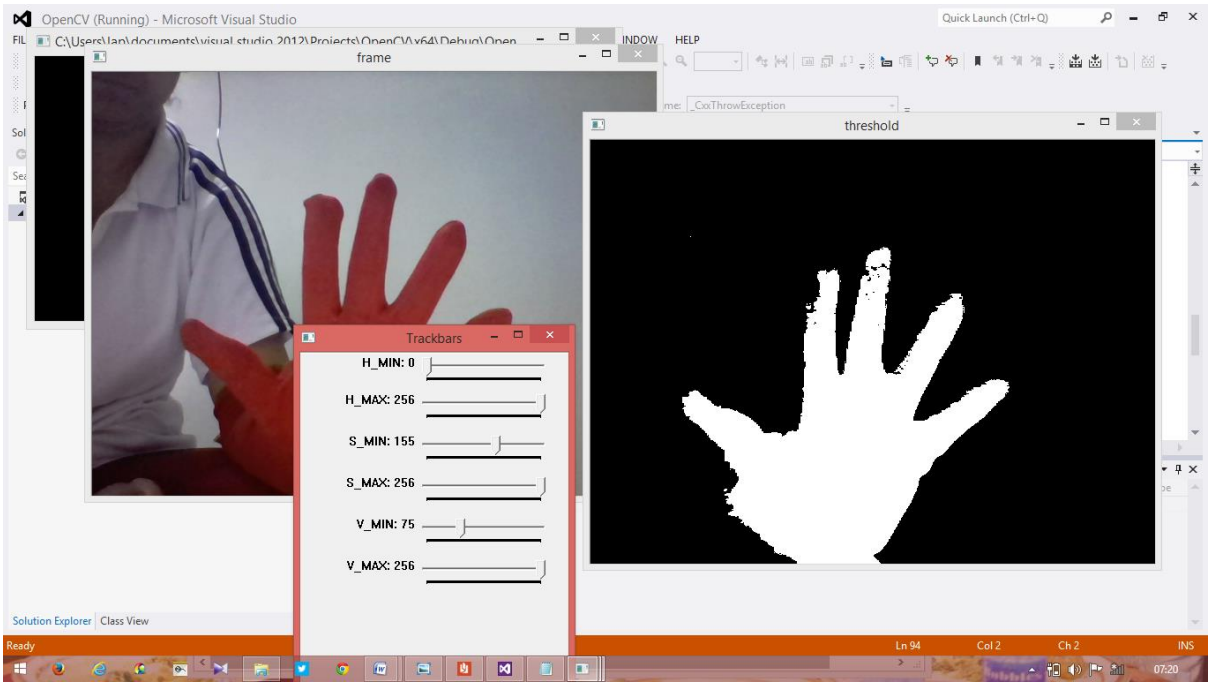


Fig 3.6 Hand detection using red gloves

3.2.2. Contour extraction

Once the binary image is obtained, it is important to extract the boundary information of the region of interest. This is done by drawing the contour around the region. For this, first a blank image is drawn on which the contour of hand is drawn. In the binary image, we find many contours. To fasten the process, define a threshold area, which is approximately equal to the area of human hand. Predefined function of contour area is available in OpenCV for this purpose. Area greater than this area will be drawn on the blank drawing.

Algorithm:

1. Define a threshold area which is approximately equal to the area of hand. Use ContourArea function for calculating the area.
2. Create a blank image.
3. Use FindContours function to find all contours in the binary image.
4. Check if the area of a particular contour is greater than threshold area.
5. If yes, plot the contour on the blank image.

Code snippet for contour extraction:

```
while(true)
{
boolbsuccess = cap.read(frame);
cvtColor(frame,hsv,CV_BGR2HSV);
inRange(hsv,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),thresh);
erode(thresh, thresh, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
dilate( thresh, thresh, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
findContours( thresh, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
vector<vector<Point>>hull( contours.size() );
vector<vector<int>>hullsI(contours.size());
vector<vector<Vec4i>> defects(contours.size());
Point centerP;
```

```
vector<int>vec(0);
drawing = Mat::zeros( thresh.size(), CV_8UC3 );
for(int i = 0;i<contours.size();i++)
{
drawContours( drawing, contours, i, Scalar(255,0,0), 1, 8, vector<Vec4i>(), 0, Point());
}
```

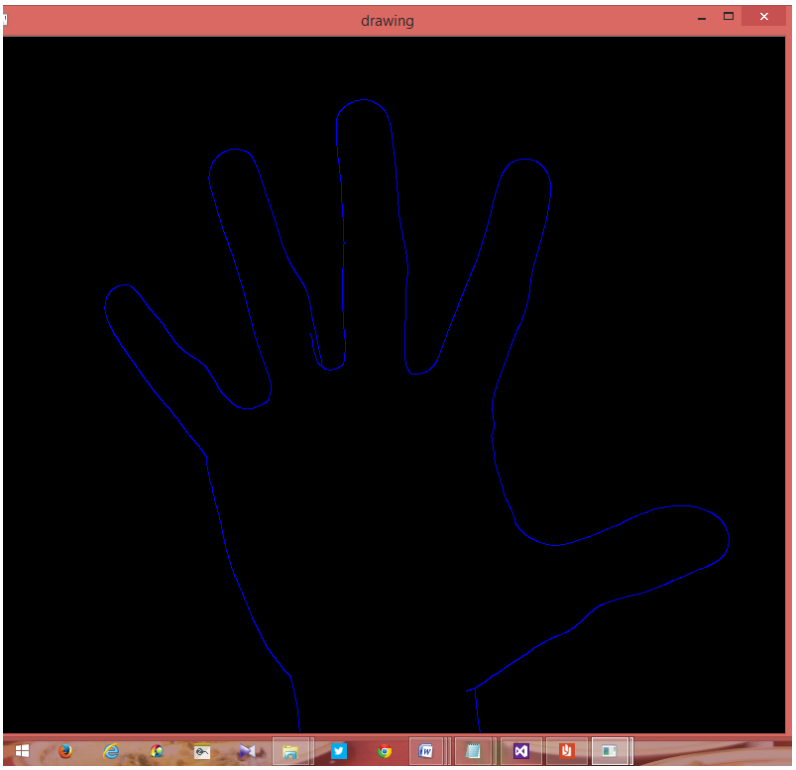


Fig 3.7 Contour of Human hand

3.2.3. Feature extraction:

To find the geometric moments of an M x N image i(x, y) for the moment of order (p + q): the formula is shown below:

$$\sum_{x=0}^{m} \sum_{y=0}^{n} X^p Y^q \text{i(x,y)}$$

m_{00} :The moment of order zero. It is equivalent to the total intensity of the image.

m_{10} : First order moment about X- axis.

m_{01} : First order moment about Y-axis.

The intensity centroidgives the geometric center of the image. It is given by :

$$\bar{X} = \frac{m_{10}}{m_{00}} \qquad \bar{Y} = \frac{m_{01}}{m_{00}}$$

Code snippet for finding center of mass of hand:

```
drawing = Mat::zeros( thresh.size(), CV_8UC3 );
for(int i = 0; i <contours.size(); i++ )
        {
 Moments moment = moments(Mat(contours[i]), true);
                        intcenterX = moment.m10 / moment.m00;
                        intcenterY = moment.m01 / moment.m00;
                        Point centerPoint(centerX, centerY);
                        centerP = centerPoint;
                        circle(drawing, centerPoint, 8, Scalar(255, 0, 0), CV_FILLED);

        }
```

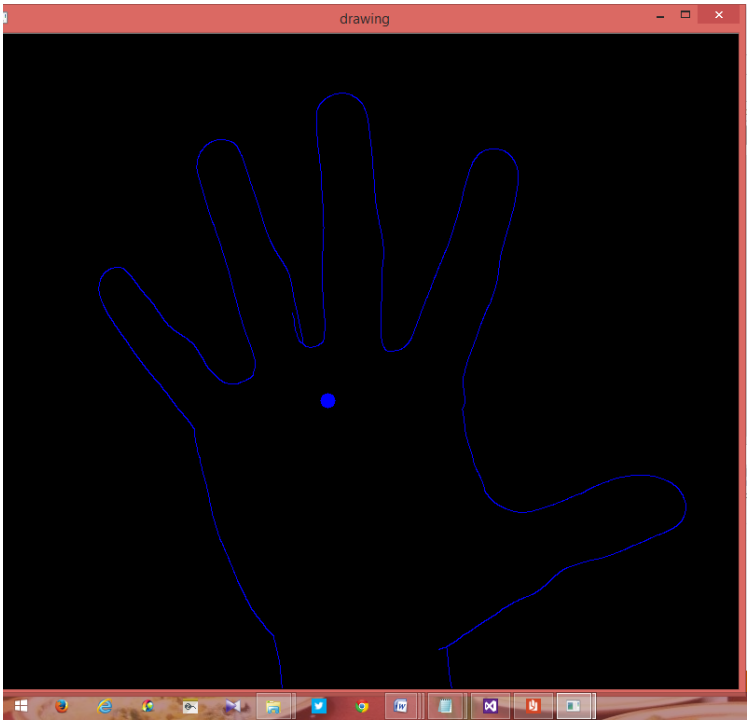


Fig 3.8 Center of mass of hand

The above figure shows the center of mass of hand. It is also called as center of luminance. The blue colour circle denotes the center of mass of the hand.

3.2.4. Finding convex hull and convexity defects

Convex Hull is drawn around the contour of the hand such that all the contour points are within the convex hull. This creates and envelope around the hand contour. It is useful in comprehending the shape of the contour.

Algorithm:

1. The set of contours for which the convex hull is to drawn is given as input.
2. Define the orientation of the convex hull (clockwise or anticlockwise).
3. Draw the hull on the blank drawing.

Convexity Defects :

When the convex hull is drawn around the contour of the hand, it uses minimum points to form the hull to include all contour points inside or on the hull and maintain the property of convexity which causes the formation of defects.

The steps for drawing the convexity defects is the same as that of drawing convex hull. Here the ConvexityDefects( ) function is used instead.

Code snippet for finding convex hull:

```
vector<vector<Point>>hull( contours.size() );
vector<vector<int>>hullsI(contours.size());
vector<vector<Vec4i>> defects(contours.size());
Point centerP;
vector<int>vec(0);
drawing = Mat::zeros( thresh.size(), CV_8UC3 );
    for(int i = 0; i <contours.size(); i++ )
      {
convexHull( Mat(contours[i]), hull[i], false );
    convexHull( Mat(contours[i]), hullsI[i], false);
            if(hullsI[i].size() > 3 )
convexityDefects(contours[i],hullsI[i],defects[i]);
      }
```

```
for(int i = 0;i<contours.size();i++)
    {
drawContours( drawing, contours, i, Scalar(255,0,0), 1, 8, vector<Vec4i>(), 0, Point());
    drawContours( drawing, hull, i, Scalar(0,0,255), 1, 8, vector<Vec4i>(), 0, Point());
    }
```

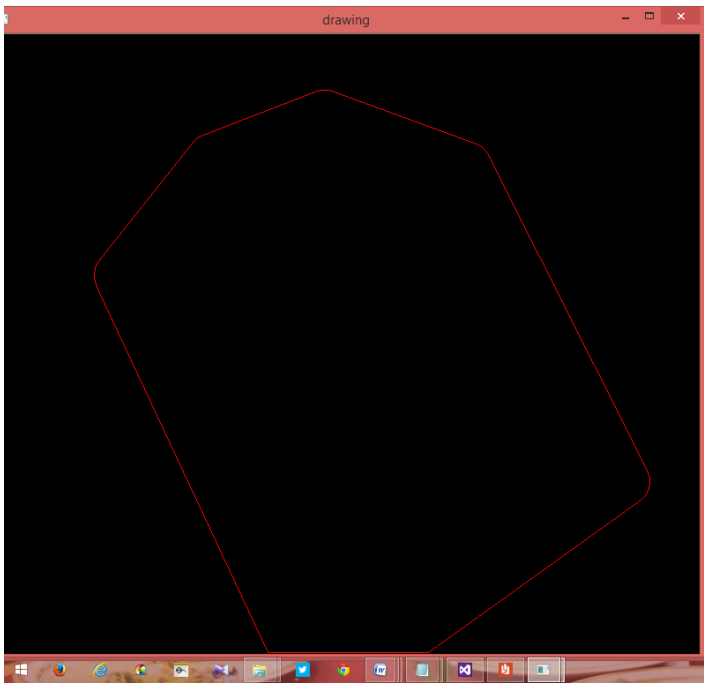


Fig 3.9 Convex hull of hand

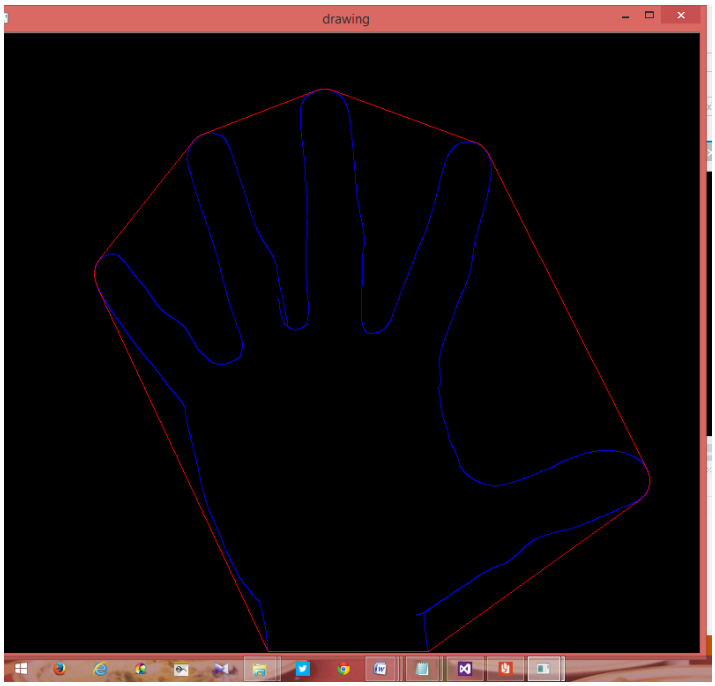


Fig 3.10 Convex hull of enclosing hand contour

3.2.5. Define parameters for design of hand gestures.

1. Point Start: Point of contour where the convexity defect begins.

2. Point End: Point of contour where the convexity defect ends.

3. Point Far: Point within the defect which is farthest from the convex hull.

4. Depth: Distance between the convex hull i.e the outermost points and the farthest points within the contour.

Code snippet for determining gesture parameters:

```
for(int i = 0; i<contours.size(); i++ )
{
size_t count = contours[i].size(); //this is to count the number of contours in your image
if( count <300 )
continue;
vector<Vec4i>::iterator d=defects[i].begin();// this is basically an iterator(which is like a
//loop) which looks for the vector that stores the convexity defects and does it untill the
//end, basically its like a for loop

while( d!=defects[i].end() )
{
Vec4i& v=(*d);
intstartidx=v[0];
Point ptStart( contours[i][startidx] );
intendidx=v[1];
Point ptEnd( contours[i][endidx] );
intfaridx=v[2];
Point ptFar( contours[i][faridx] );
float depth = v[3]/256;
circle( drawing, ptFar,4, Scalar(0,255,0), 2 );
circle( drawing, ptStart,4, Scalar(0,0,255), 2 );
circle( drawing, ptEnd,4, Scalar(255,0,0), 2 );
}
```

}

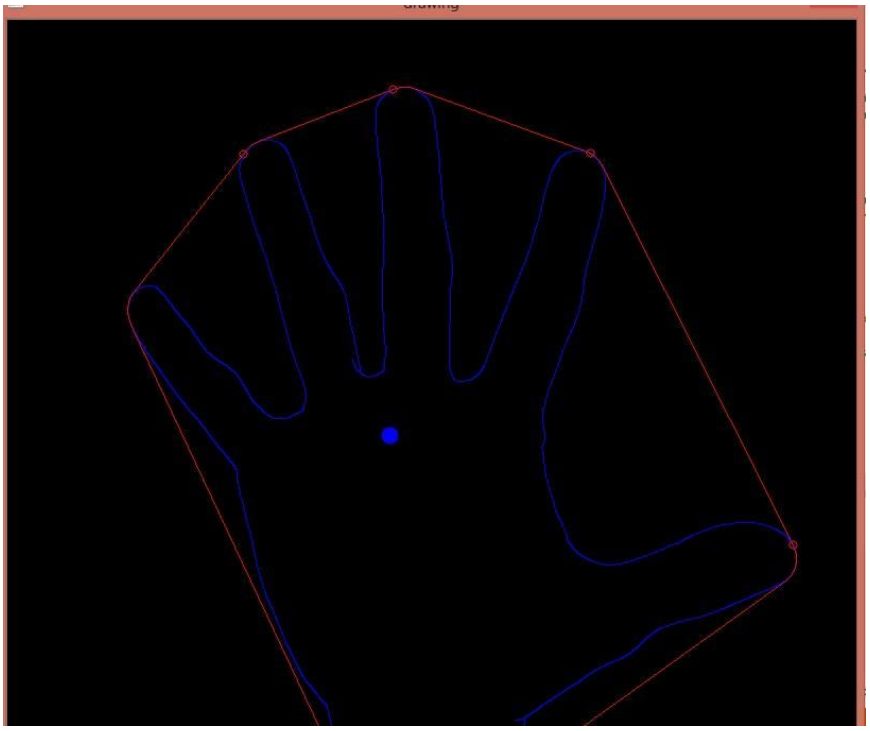


Fig 3.11Point Start

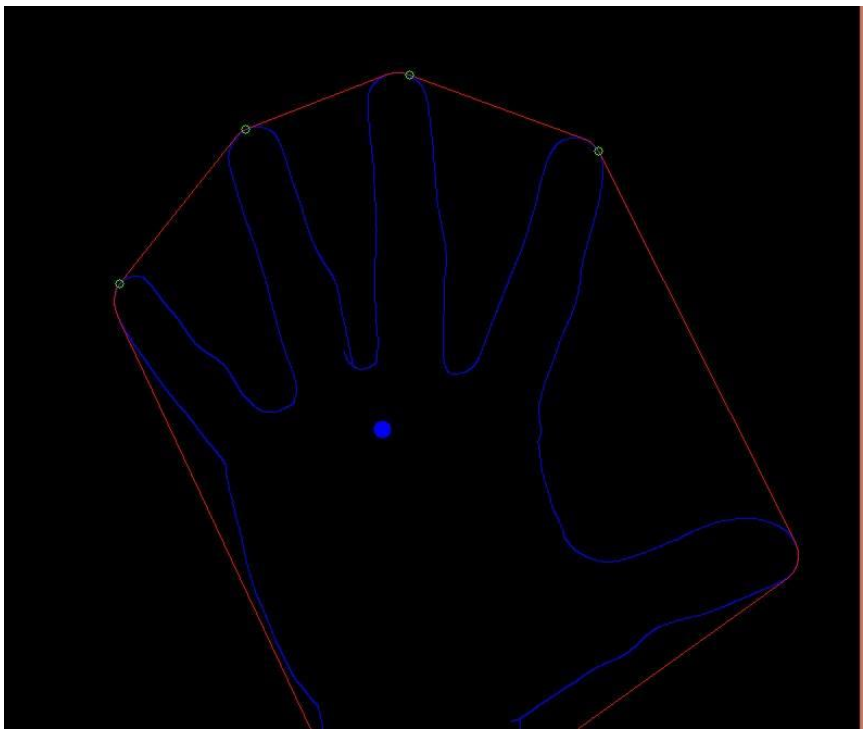


Fig 3.12Point End

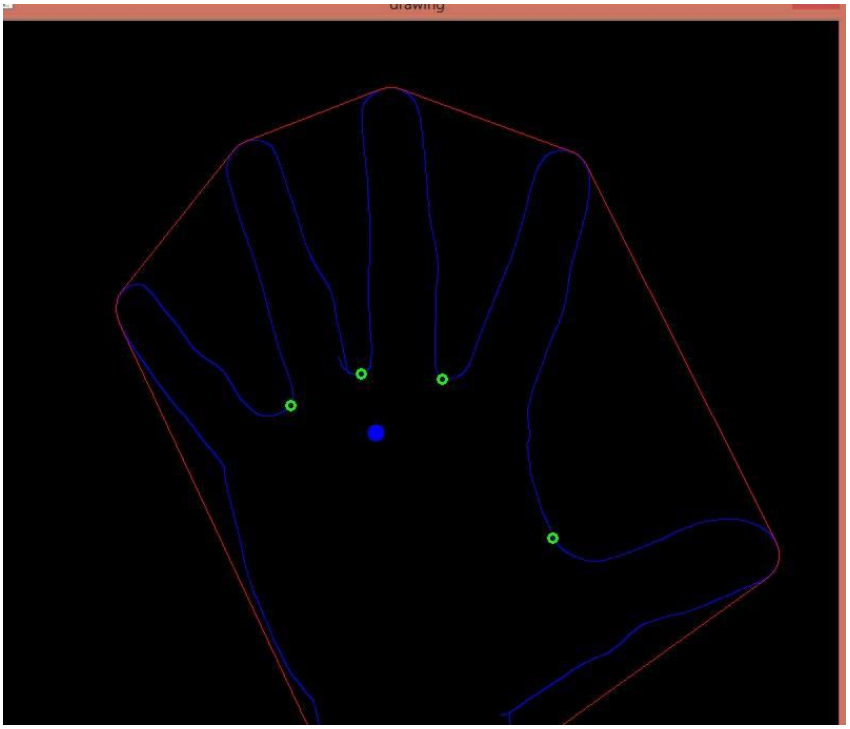


Fig 3.13Point Far

3.2.6. Design of hand gestures.

The number of fingers present in the hand gesture is found it. To find the number of fingers, the convexity defect points are used i.e. the Far points are used. This is preferred to avoid confusion of large number of points that were obtained when points near the hull were used. Besides, the points near the defects are more prominently detected. Hence, here, I concentrated on the Point far for each finger.

No. of fingers = No. of Point far detected +1

i.e if No. of ptFar = 1, No of fingers = 2

if No. of ptFar = 2, No. of fingers = 3and so on.

No. of ptFar in the image can be obtained from the size of the vector.

i.e. if (vec.size ==1), then No. of fingers =2

Compute the angle and distance of ptFar and center:

dist = (ptFar.x - centerP.x) ^2 + (ptFar.y - centerP.y) ^2;

floaty_angle = ptFar.y - centerP.y;

floatx_angle = ptFar.x - centerP.x;

float theta = atan(y_angle/x_angle);

angleFinger = static_cast<int>(theta * 180 / 3.14);

3.2.7. Interfacing Microsoft Visual Studio using WinAPI and perform gesture control

WinAPI stand for windows application programming interface. It contains declarations for all macros that are used by windows programmers.

Win32API is added here. For this we include <windows.h> header file in our code.

3.2.8. Perform hand gestures for control of mouse functions.

As mentioned above, the numbers of fingers are calculated. Depending on the no. of fingers present in the gesture, the respective mouse event is performed.

ifNo. of fingers = 1, perform right click.

if No. of fingers = 2, perform left click.

if No. of fingers = 3, perform double click.

if No. of fingers > 3, move the cursor on the screen.

Code snippet :

```
if (vec.size() == 1)// vec gives the no. of convexity defects, if vec= 1, perform right click
    {
mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0);
    mouse_event(MOUSEEVENTF_RIGHTUP,0,0,0,0);
    }         if(vec.size() == 2) // perform left click          {
    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);  }
    if(vec.size() ==3) // perform right click         {
    mouse_event(MOUSEEVENTF_LEFTDOWN,0,0,0,0);
    mouse_event(MOUSEEVENTF_LEFTUP,0,0,0,0);
    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0); mouse_event(MOUFTUP,
    0, 0, 0, 0);
    }
if(vec.size() >3) // move the cursor on the screen.
{
POINT cursor GetCursorPos(&cursor);
SetCursorPos(x,y);
}
```

4. Results and Discussion

The codes for finding contour, convex hull, convexity defects, and gesture parameters were executed. Following results were obtained.

All results are subjected to the lighting conditions and the relative position of the hand with respect to the webcam. The camera resolution plays an important factor. Since it is the web camera inbuilt in our laptops is used in this project, the performance is quite slow. This is due to lower frame processing speed. To get better results, external camera can be used.

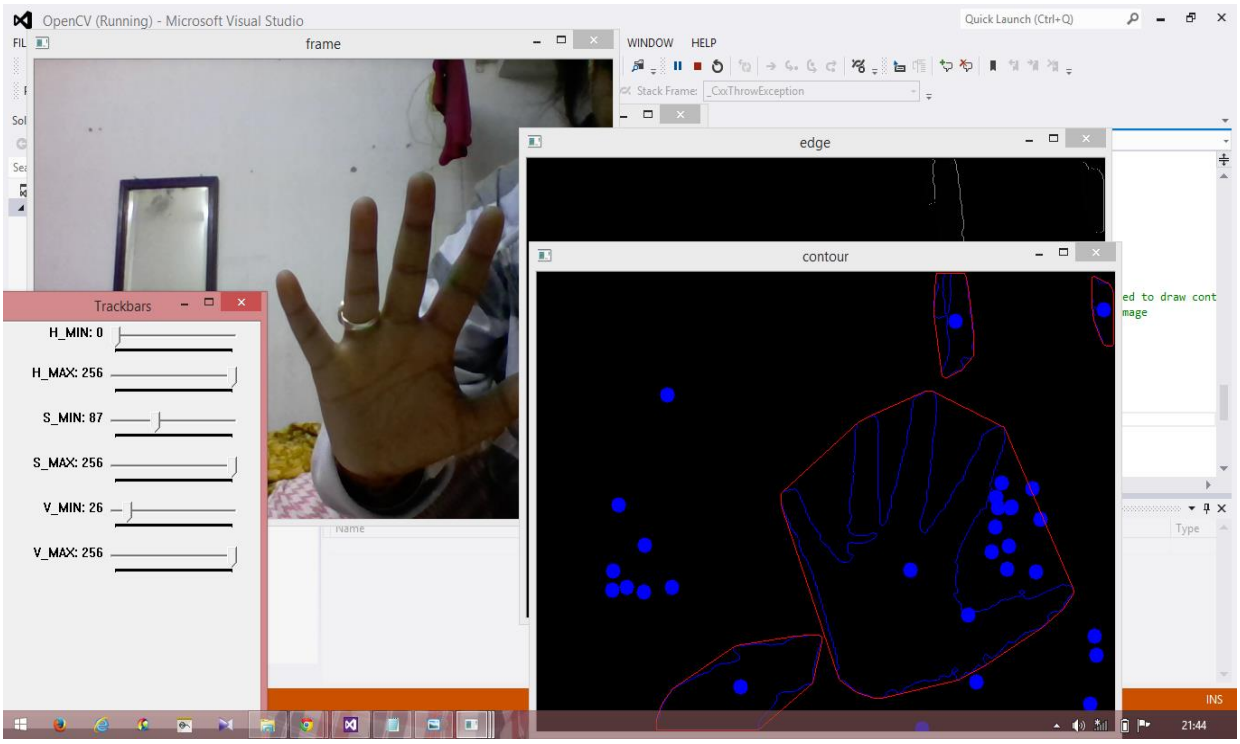


Fig 4.1Convex Hull of hand

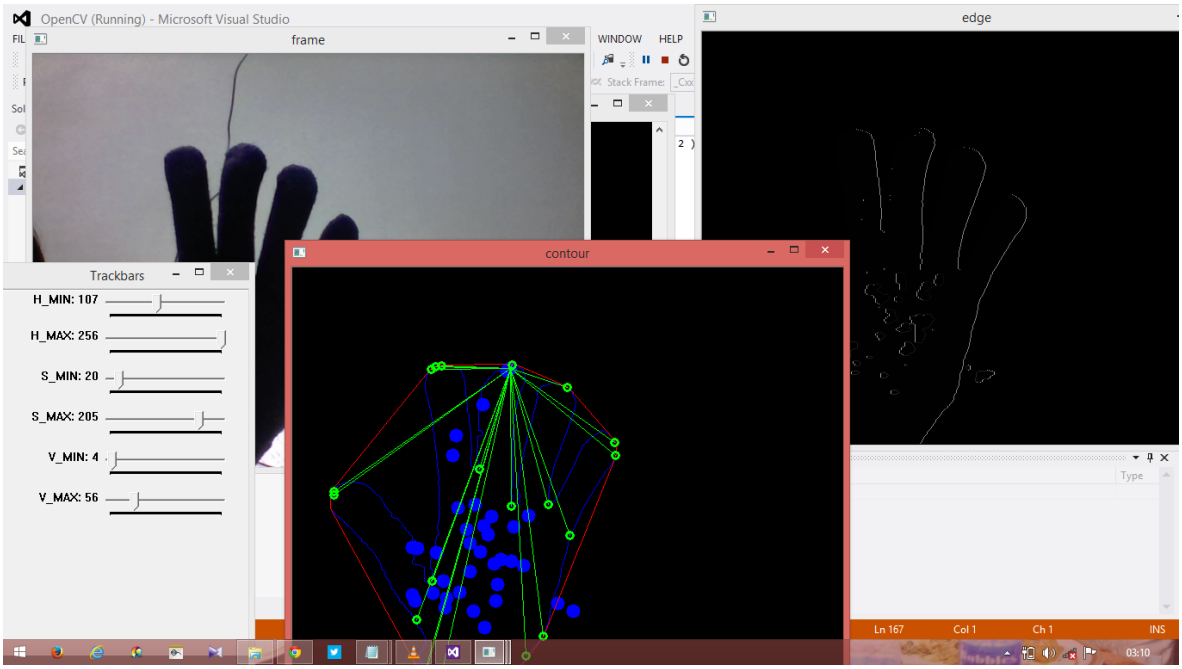


Fig 4.2Gesture parameters

In the above image, the numbers of points detected are too many. To remove unwanted points, define the depth (distance between the convex hull and point far) within a particular range. The points at farthest distance from the convex hull are of interest (ptFar). The depth is calculated from top of convex hull. So we can refine the points by displaying only those points that are beyond certain depth. In this way, only the point far will be displayed. Here points with (depth>50) are displayed.

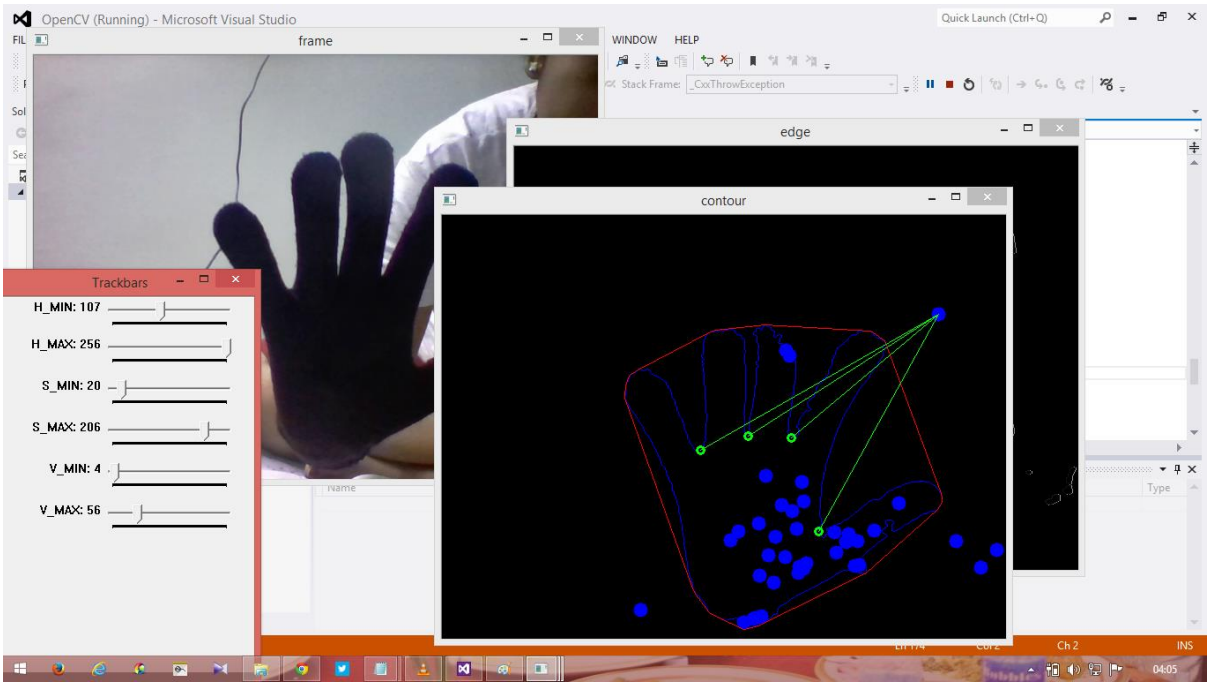


Fig 4.3Deriving Point far of hand

Need of red gloves: In absence of red gloves, the noise in the image was very high and thus multiple points were detected. With the use of red gloves, the image segmentation became easier and faster. So, for better results, red gloves were used.

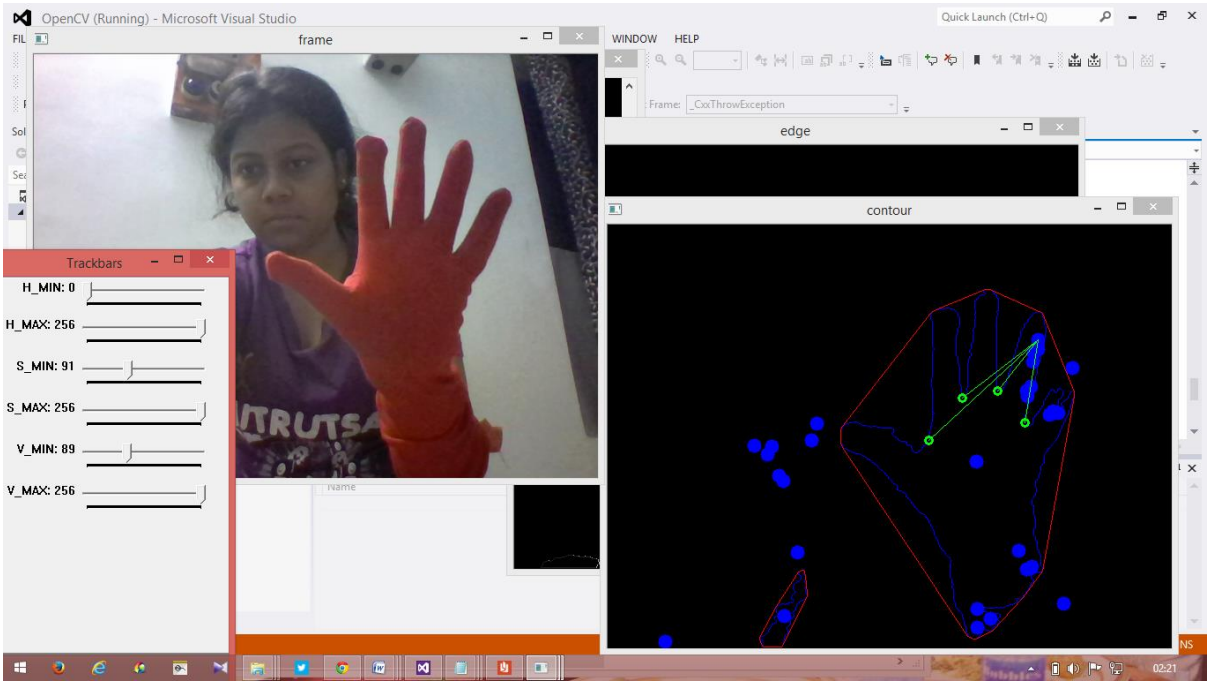


Fig 4.4 Derive point far of hand with red gloves.

Gestures Used:

1. *Right click* :When the no. of fingers is two or when (vec.size ==1) , right click is performed.

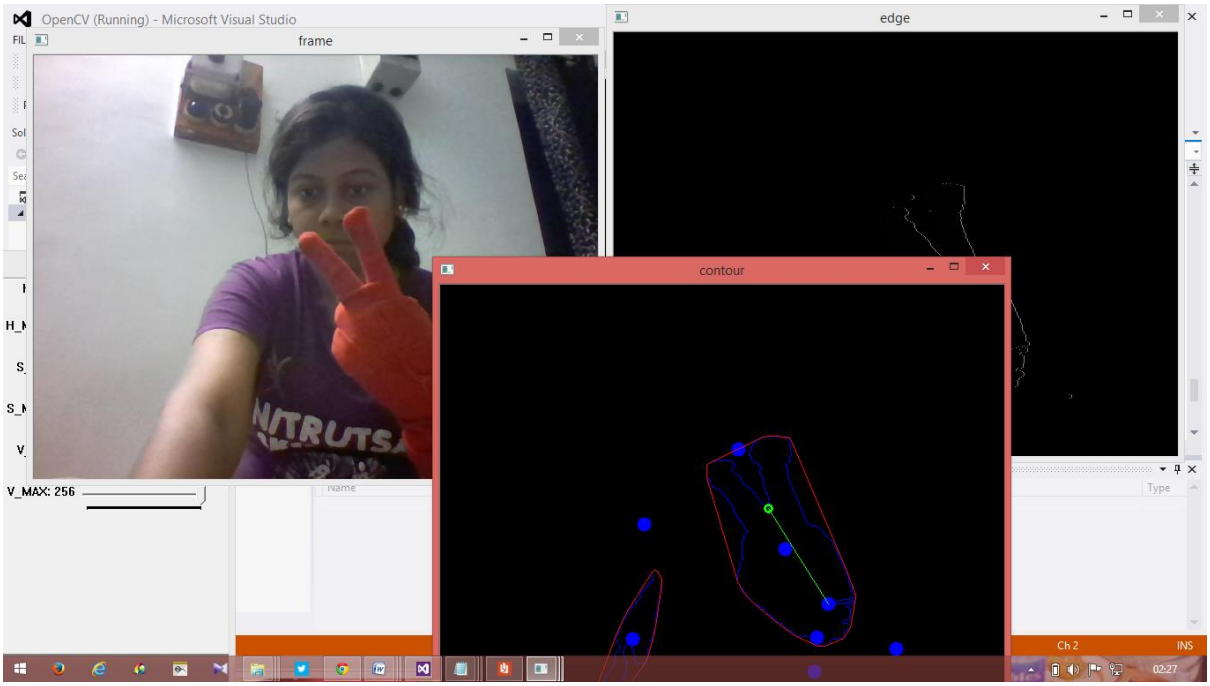


Fig 4.5 Gesture 1 : right click

2. *Left click* :When the no. of fingers is three or when (vec.size ==2) , left click is performed.

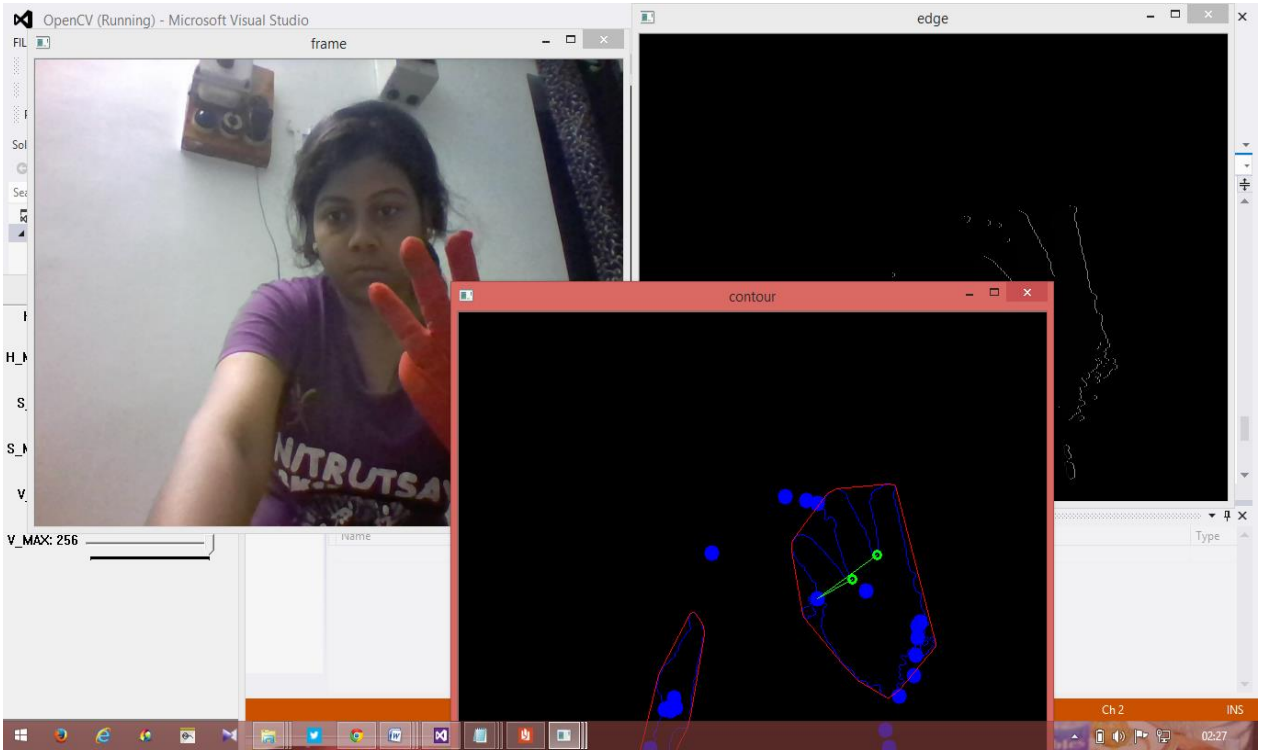


Fig 4.6 Gesture 2: Left Click

3. *Double click* : When the no. of fingers is four or when (vec.size ==3) , double click is performed.

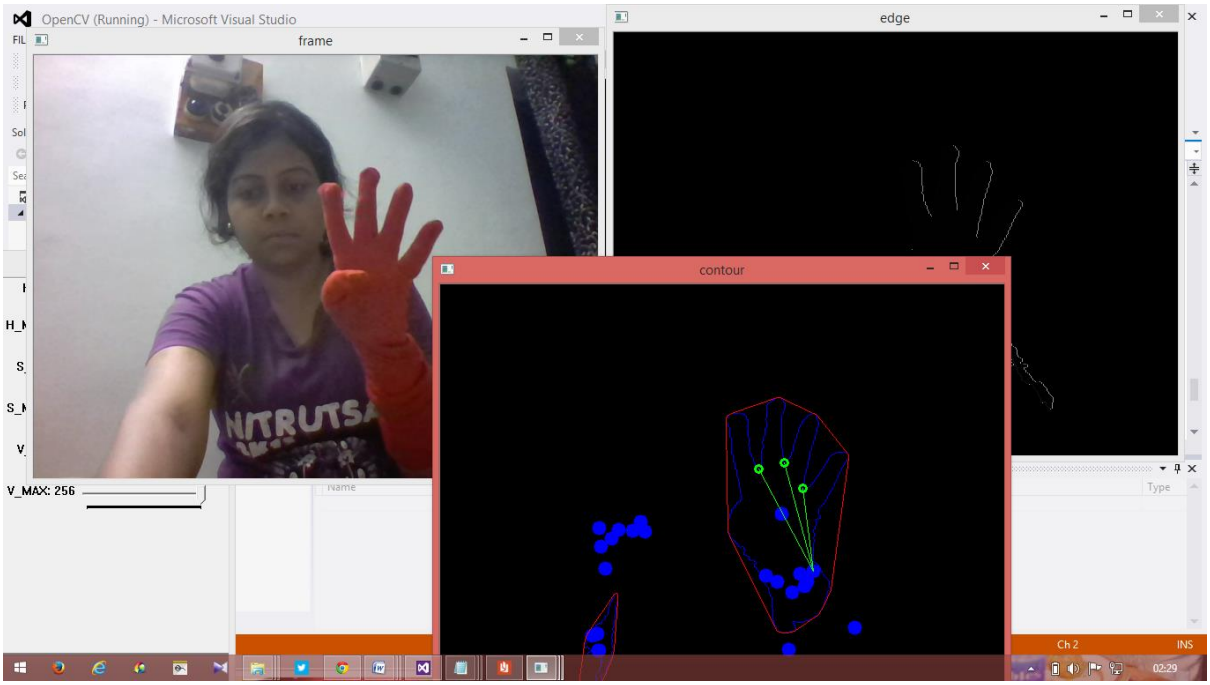


Fig 4.7 Gesture 3 : double click

4. Move cursor on screen: When the no. of fingers is greater than four i.e five or when (vec.size> 3) ,the cursor moves on the screen.

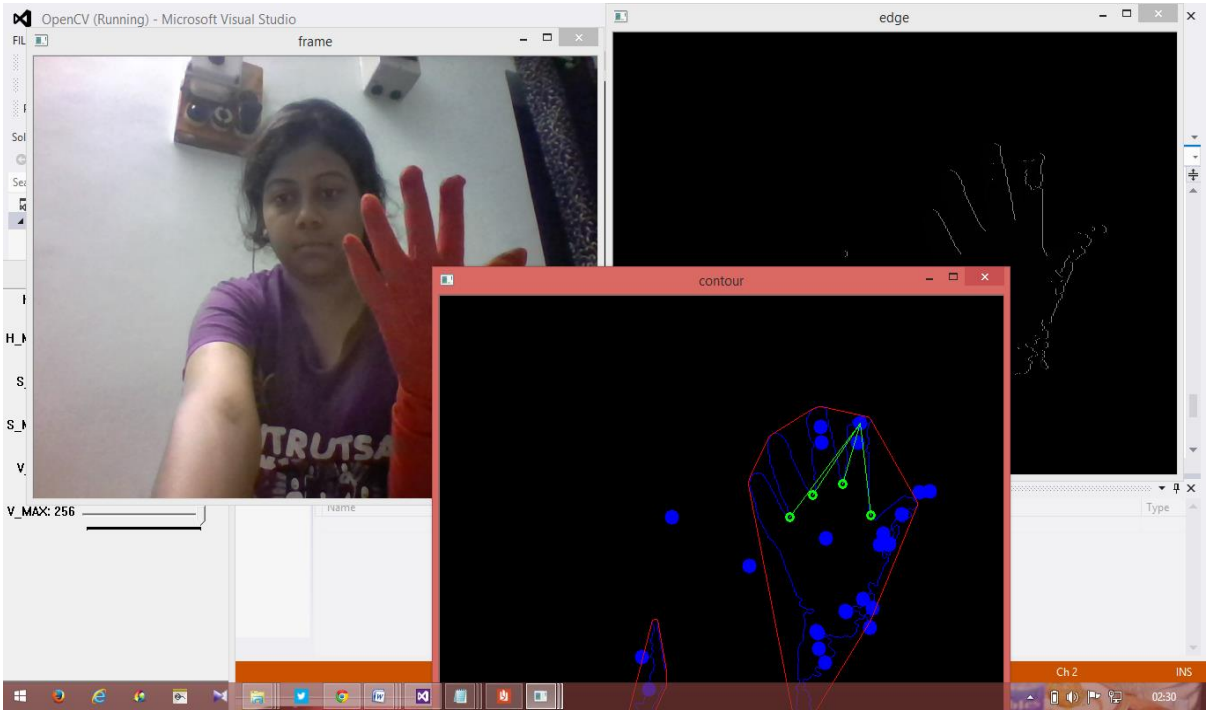


Fig 4.8 Gesture 4 : Move cursor on screen.

5. Scope of Future Work

The gestures that have been used in this project for mouse control are static gestures. The codes can be further improvised for performing dynamic gestures. Dynamic gestures are more interactive in nature.

In this project, color based gesture control is performed. Here red gloves is used which makes segmenting easier. With the use of a more powerful camera, codes can be modified for performing gesture control without color. The user can simply put his hand before the camera and perform the gesture, without the need of any gloves. This will improve user satisfaction and the process will be easier to use.

Due to the poor quality camera in our laptops, the processing is very slow. Besides the detection of the various points i.e. the gesture parameters, depends on the relative position and angle from the camera. With the change in position or angle, the detected points change. The code is to be made more robust so that these problems cannot affect the end results. Use of a better external camera to capture the images will also improve the results to a great extent.

In this project, I have only discussed about performing mouse controls using gesture. Similarly, keyboard functions and many other computer functions can be performed with gestures, only with a slight modification in code.

6. Conclusion

Controlling of mouse functions using gesture control were performed. The application successfully performs left click, right click, double click and moving the mouse cursor on the screen. It takes the hand as input from the webcam in form of continuous frames. Depending on colour, the hand is segmented to get a binary image. The contour of the hand is extracted and is drawn on a blank image. Convex hull of the hand is found out using the convex hull algorithm and is drawn on the blank image. The convexity defects are found and three points are located on the hand contour. These are point start, point end. These are the points where the convexity defects starts and ends respectively. The third point is point far. It is the point which is at the farthest distance from the convex hull. In this project all work has been done using the point far.

The concept of the project possesses a huge prospect in the future. With improvisations in the code, gesture control can be used for performing any activity on the computer like playing games, painting, controlling media player operations etc. From 'touch' generation, we are moving to 'no touch' generation. It is the generation of intangible interfaces. From computer platform to mobile platform, gesture control can be implemented everywhere. The front cameras of the mobile can be used for image input for gesture control. Intangible interfaces are at the dawn of the day and touch interfaces are at the dusk. Soon intangible interfaces will take over the touch interfaces.

References

[1] Learning OpenCV, computer vision in C++ with the open CV library by Adrian Kaehler & Gary Bradski.

[2] A. Chaudhary et al., “Intelligent approaches to interact with machines using hand gesture recognition in a natural way: A survey,” International Journal of Computer Science and Engineering Survey (IJCSES), vol. 2, no. 1, Feb 2011.

[3] Tie Yang & YangshengXu,"Hidden Markov Model for Gesture Recognition",CMU-RI-TR-94 10, The Robotics Institute, Carnegie Mellon University, May 1994.

[4] A. Pandit, D. Dand, S. M. Sabesan, A. Daftery, “A simple wearable hand gesture recognition device using iMEMS,” Soft Computing and Pattern Recognition, 2009. SOCPAR ’09. International Conference, vol. 4, no. 7, pp. 592-597, Dec. 2009.

[5] A. Erdem, E. Yardimci, Y. Atalay, and V. Cetin, “Computer vision based mouse,” A. E. Acoustics, Speech, and Signal Proceedings (ICASS). IEEE International Conference. 2002.

[6] Paul Viola& Michael Jones,"Robust Real-time Object Detection", Second International Workshop on Statistical and Computational Theories of Vision-modeling, Learning, Computing, and Sampling, Vancouver, Canada, July 13, 2001.

[7] H. Park, “A method for controlling mouse movement using a real-time camera,” 2012.

[8] Nayana P B1, SanjeevKubakaddi, "Implentation of Hand Gesture Recognition Technique for HCI Using Open CV" ,International Journal of Recent Development in Engineering and Technology, ISSN 2347 - 6435 (Online) Volume 2, Issue 5, May 2014.

[9] A. Dhawan & V. Honrao Implementation of Hand Detection based Techniques for Human Computer Interaction, International Journal of Computer Applications (0975 – 8887) Volume 72– No.17, June 2013

[10]A. Jinda-apiraksa et al. , ”A Simple Shape-Based Approach to Hand Gesture Recognition”, Proceedings of International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology, Chiang Mai, pp. 851-855, May 2010.