Hand Gesture Recognition and Cursor Control

Technical Report · March 2015

DOI: 10.13140/RG.2.1.3185.0082

3 authors, including:

Shashank Salian
Vivekanand Education Society's Institute of Technology
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Dhiren Serai
Vivekanand Education Society's Institute of Technology
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(Affiliated to University of Mumbai)

Looking Beyond Syllabus Project Report

HAND GESTURE RECOGNITION AND CURSOR CONTROL

Shashank Salian

Pranav Ganorkar

Dhiren Serai

Mentor-: Dr. Nadir Charniya

Computer Engineering Department

And

EXTC Department

Academic Year : 2014 - 15

ACKNOWLEDGEMENT

We are thankful to Harvard’s CS50x course which first introduced us to the device called Leap Motion. This gadget inspired us to take up this project .We are also thankful for the VESIT faculty for giving us this opportunity .This project was possible due to College Lab Support .

Special Thanks:-

- Dr. Nadir Charniya (Mentor LBS project - EXTC Department):-

  His guidance inspired us to improve our project more and more.

- Dr.Nupur Giri (HOD –Computer Engineering Department)

  Her approval helped us to use the college labs even during the vacations.

- Mrs. Priya R.L.(Computer Engineering Department)

  She cleared our various queries at different stages of the project.

INDEX

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.

MOTIVATION

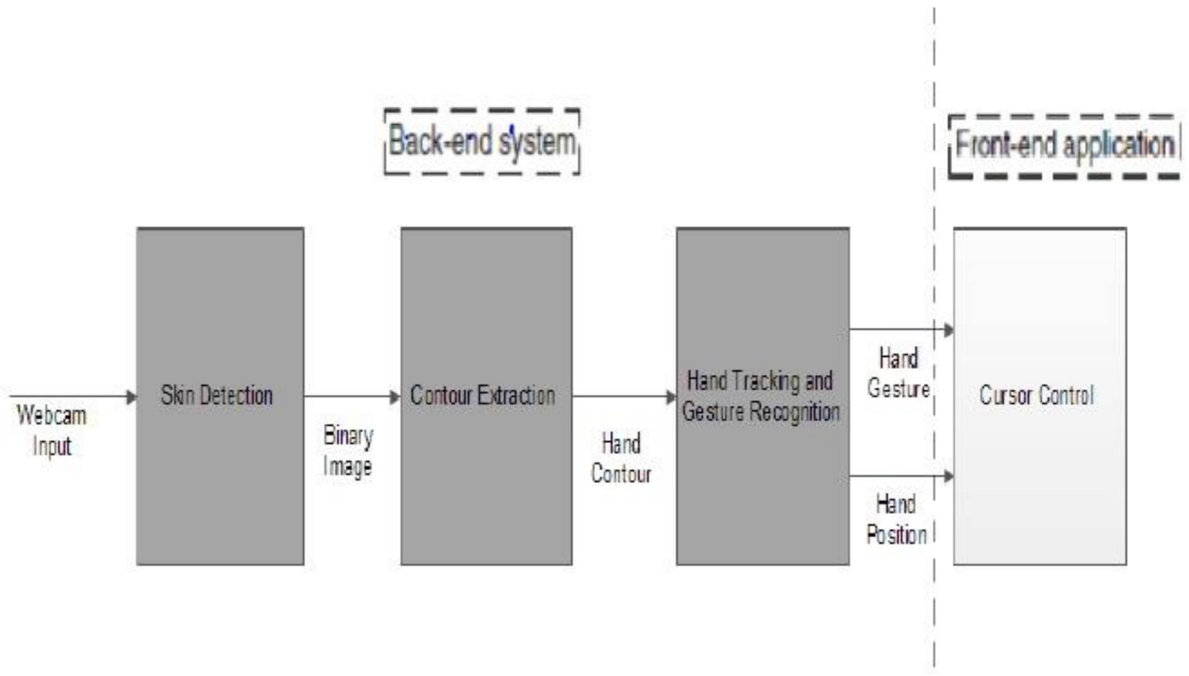
We had chosen this project with an interest of learning the direct interaction of humans with the consumer electronic devices . This takes the user experience to a whole new level. The gesture control technology would reduce our dependence on the age old peripheral devices hence it would reduce the overall 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.

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:



ISSUES AND CHALLENGES

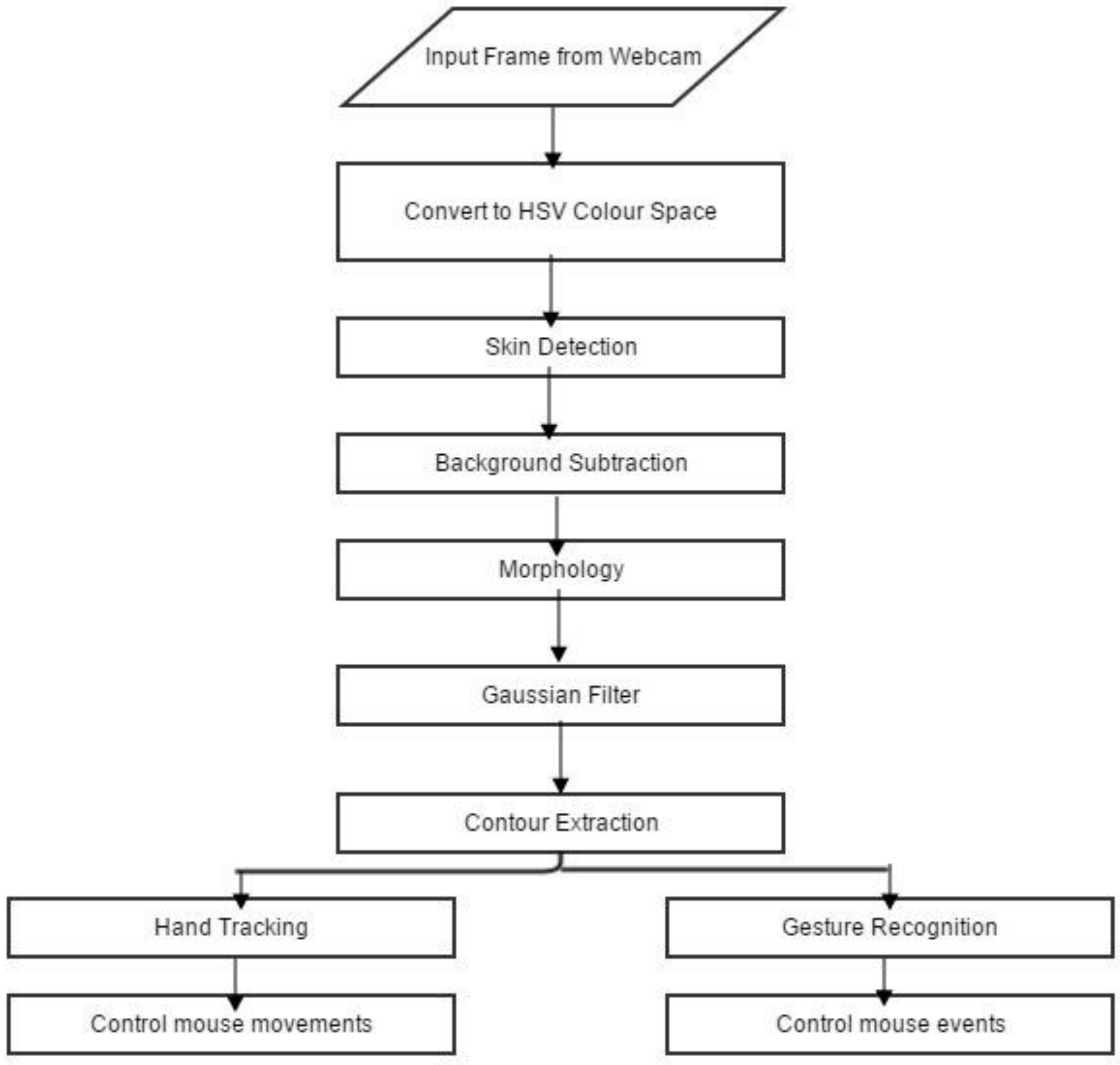
The first challenge was to correctly detect the hand with a webcam. We needed a Computer Vision library for this purpose. Many are available but we decided to go ahead with OpenCV[1] as it is the most popular and has been ported to many languages and is supported on many operating systems from Android to Windows. It has a good library collection of standard image processing functions. Then we had to first setup OpenCV[1] on our IDE(Visual Studio). That was a very tedious task. We also had to learn some basic usage of OpenCV[1]. For which we referred to many tutorials on the web. After learning OpenCV[1], 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.

PROJECT DESCRIPTION

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:

i. Skin Detection

ii. Hand Contour Extraction

iii. Hand Tracking

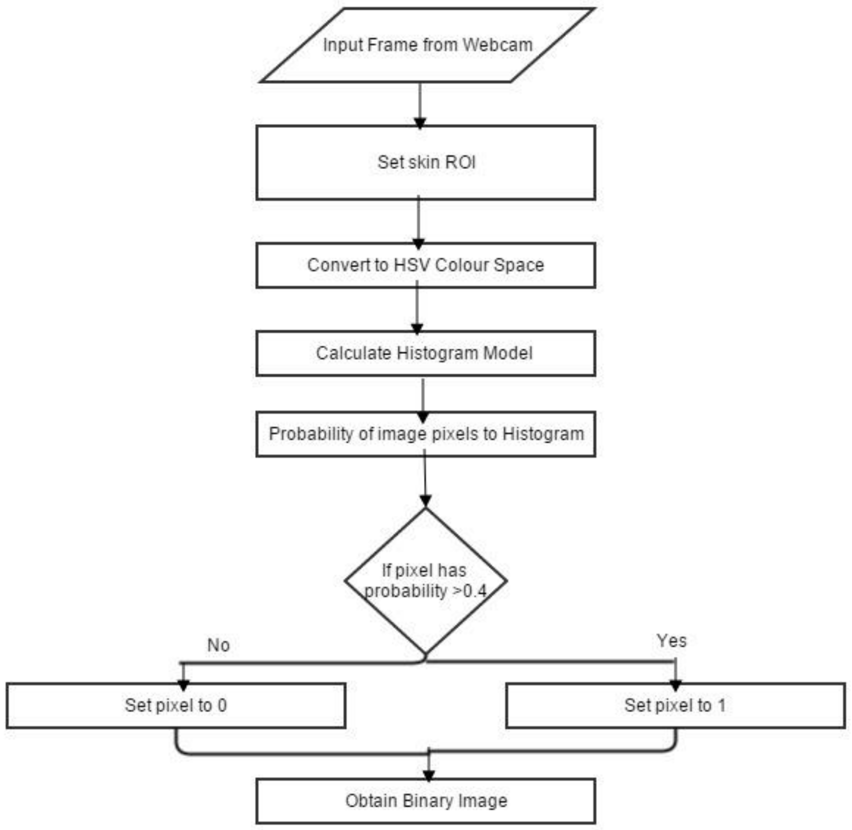
iv. Gesture Recognition

v. 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 Histogram-based 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.


Input Frame from Webcam
Set skin ROI
Convert to HSV Colour Space
Calculate Histogram Model
Probability of image pixels to Histogram
If pixel has probability >0.4
No
Yes
Set pixel to 0
Set pixel to 1
Obtain Binary Image

Algorithm for Skin Detection

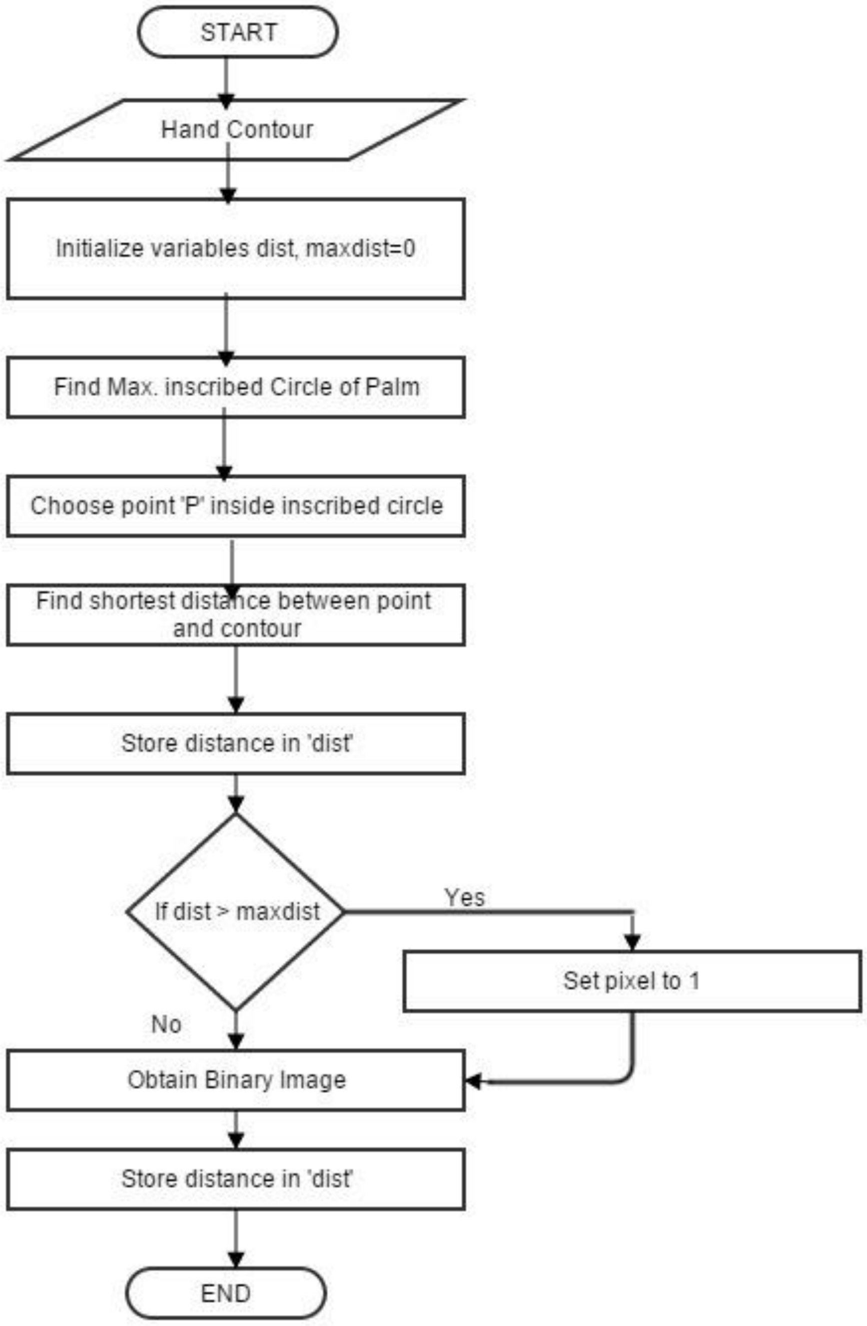
Hand Contour Extraction

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.


START
Hand Contour
Initialize variables dist, maxdist=0
Find Max. inscribed Circle of Palm
Choose point 'P' inside inscribed circle
Find shortest distance between point and contour
Store distance in 'dist'
If dist > maxdist
Yes
Set pixel to 1
No
Obtain Binary Image
Store distance in 'dist'
END

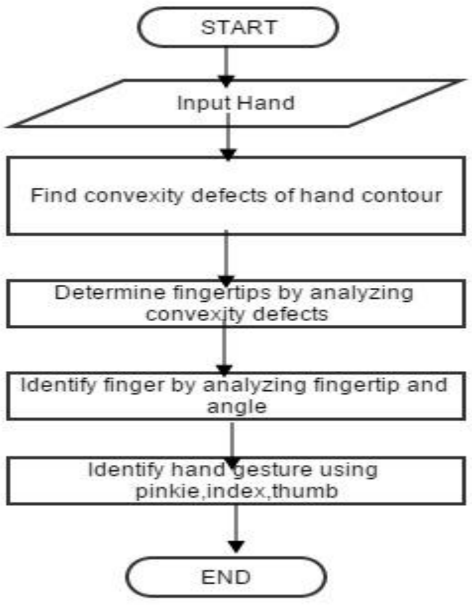
Algorithm for Hand Tracking

Gesture Recognition

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:

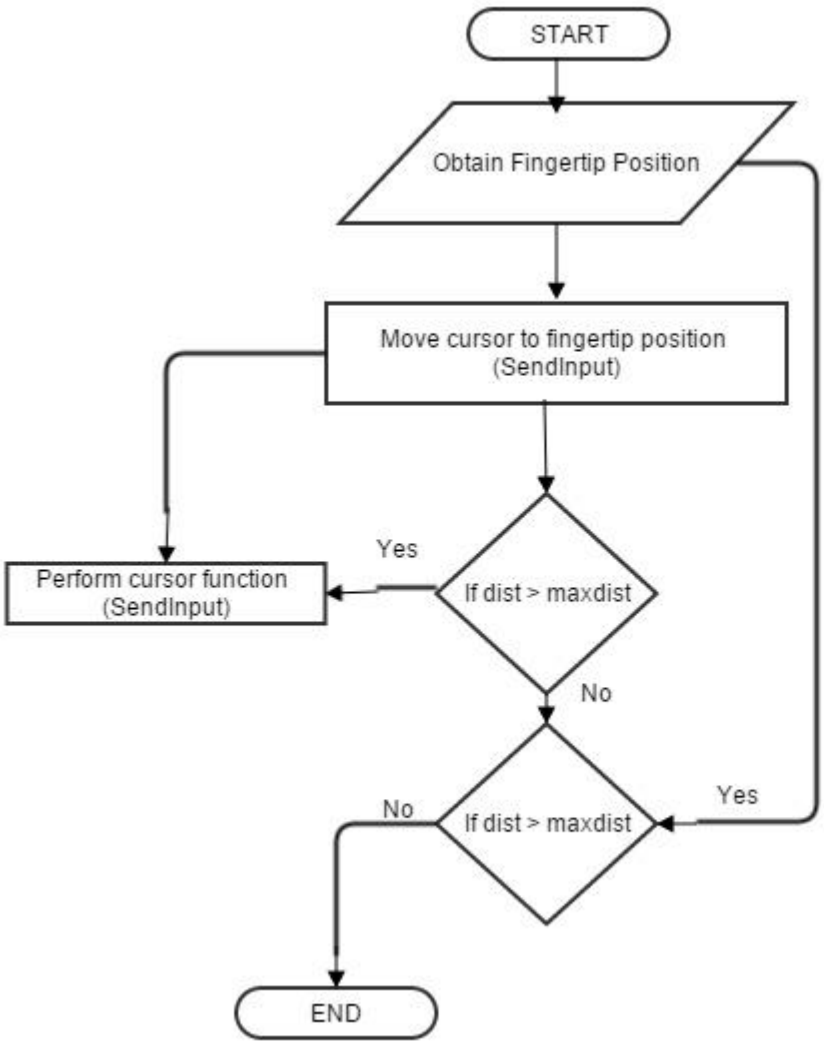
i. Finger Tip Identification

ii. Number Of Fingers


START
Input Hand
Find convexity defects of hand contour
Determine fingertips by analyzing convexity defects
Identify finger by analyzing fingertip and angle
Identify hand gesture using pinkie,index,thumb
END

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.


START
Obtain Fingertip Position
Move cursor to fingertip position (SendInput)
Yes
Perform cursor function (SendInput)
If dist > maxdist
No
No
If dist > maxdist
Yes
END

Algorithm for Cursor Control

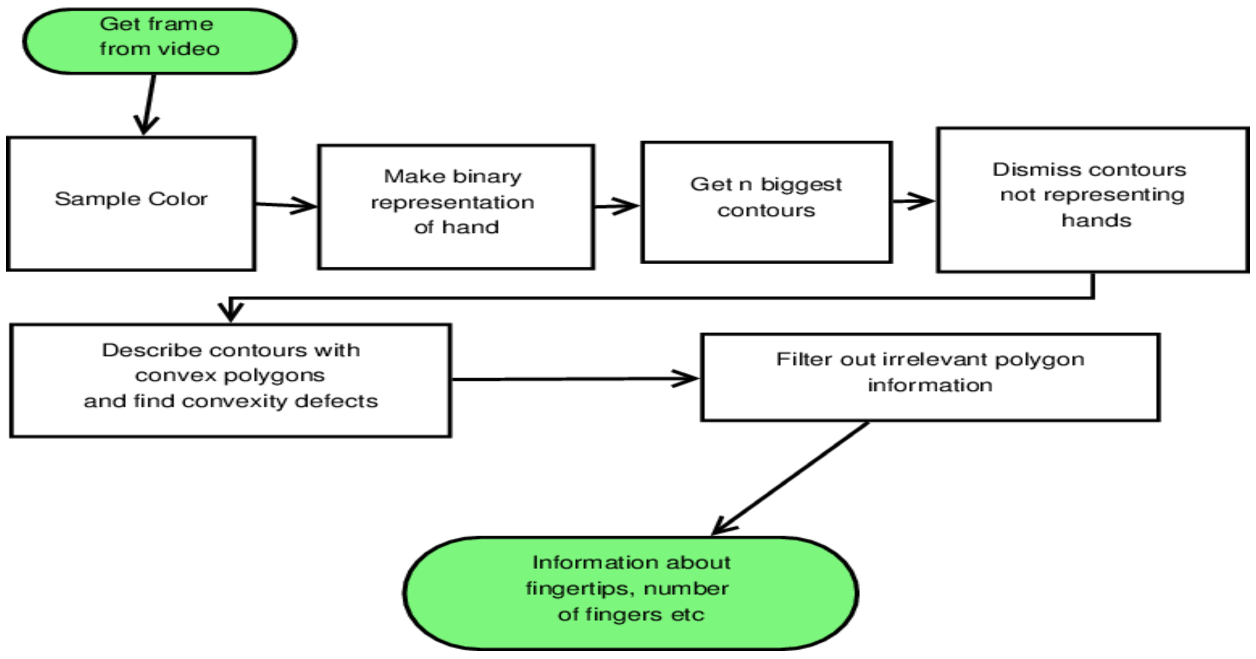
Following table shows the Operations Performed depending upon the number of fingers detected:

Number of Fingertips Detected	**Operations Performed**
One	Move Cursor
Two	Left Click
Three	Right Click
Four	Start Button
Five	My Computer

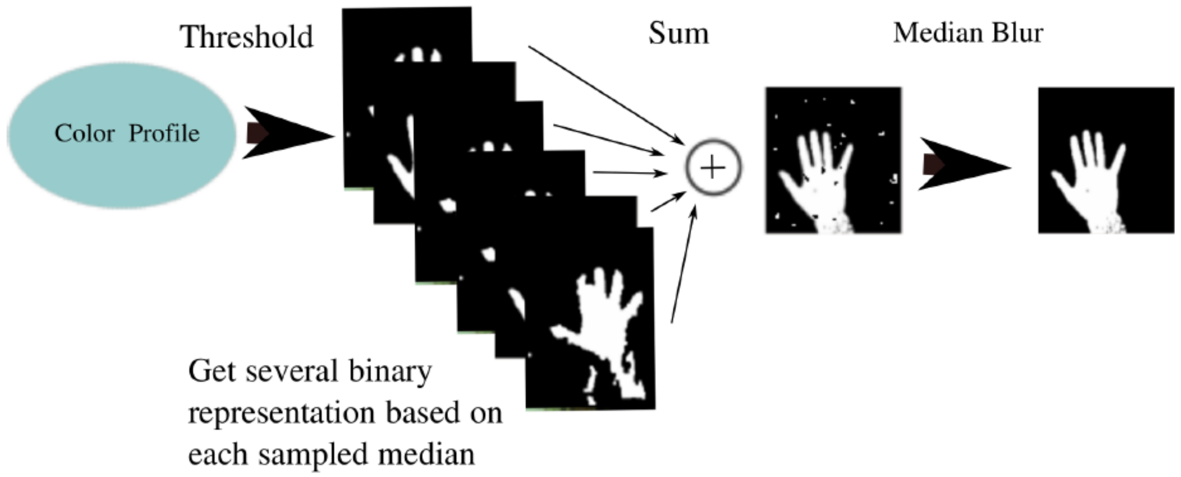
Starting with the position of the index fingertip, the cursor is moved to the fingertip position. This is done using the "SendInput" function to control the cursor movement. The next step would be to determine if a hand gesture was performed. If a hand gesture as performed, the "SendInput" function is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

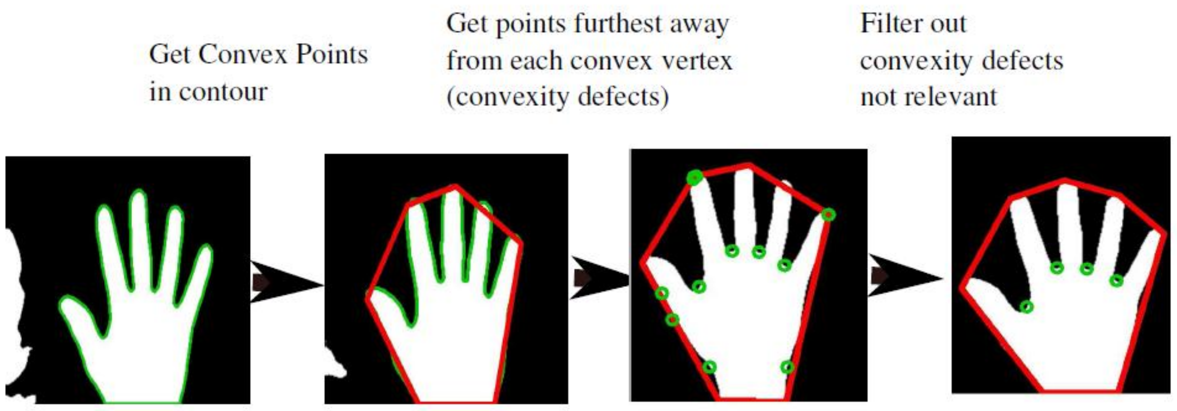
Below is a summary flowchart representation of the program:


Get frame from video
Sample Color
Make binary representation of hand
Get n biggest contours
Dismiss contours not representing hands
Describe contours with convex polygons and find convexity defects
Filter out irrelevant polygon information
Information about fingertips, number of fingers etc

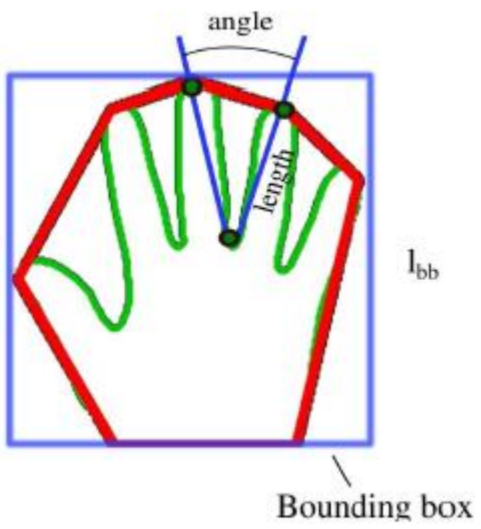
The hand tracking is based on color recognition. The program is therefore initialized by sampling color from the hand. The hand is then extracted from the background by using a threshold using the sampled color profile. Each color in the profile produces a binary image which in turn are all summed together.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


Threshold
Color Profile
Sum
Median Blur
Get several binary representation based on each sampled median

When the binary representation is generated the hand is processed in the following way:


Get Convex Points in contour
Get points furthest away from each convex vertex (convexity defects)
Filter out convexity defects not relevant

The properties determining whether a convexity defect is to be dismissed is the angle between the lines going from the defect to the neighbouring convex polygon vertices.



The defect is dismissed if:

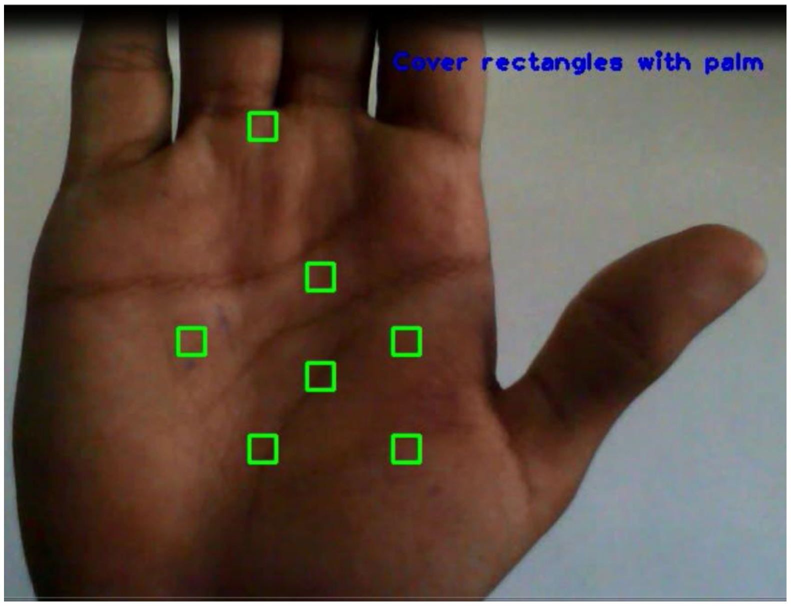
$$\text{Length} < 0.4 l_{bb}$$
$$\text{Angle} > 80^0$$

The analysis results in data that can be of further use in gesture recognition:

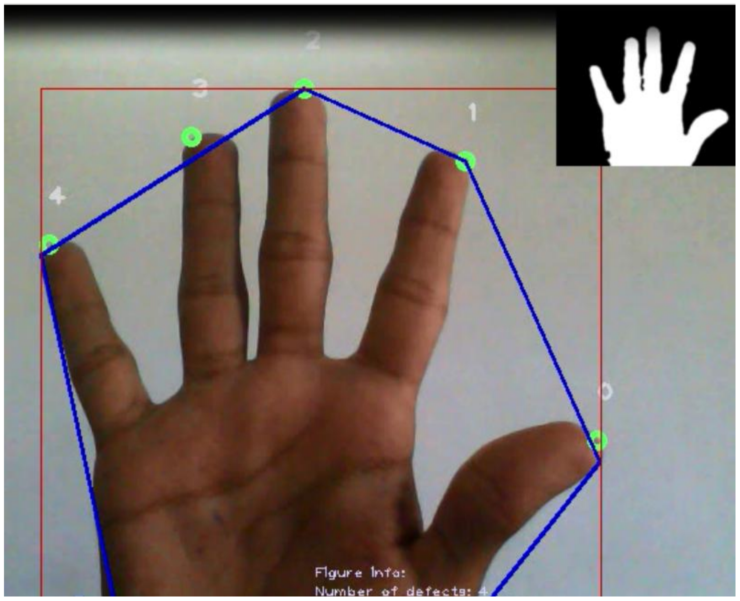
- Fingertip positions
- Number of fingers
- Number of hands
- Area of hands

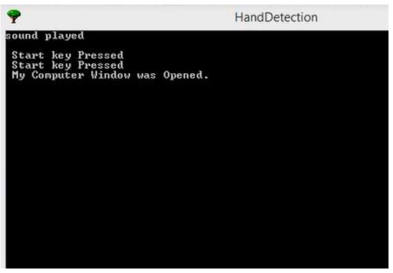
This is how it works:

First, The program asks you to place Palm on the Rectangles:



Now, let’s see how it tracks our palm and detects our fingers:

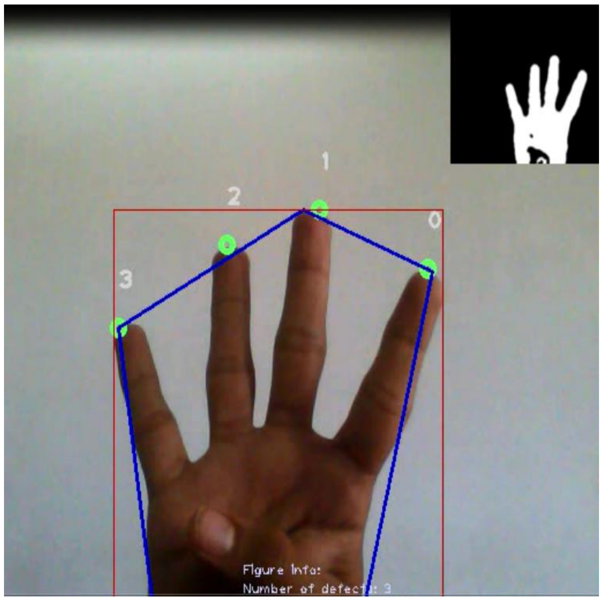


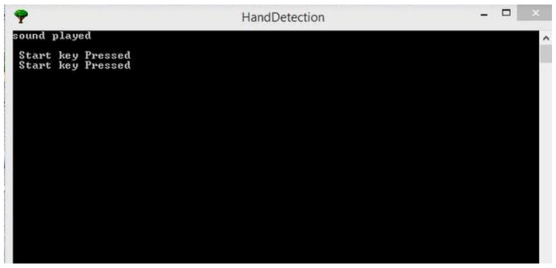


```
sound played

 Start key Pressed
 Start key Pressed
 My Computer Window was Opened.
```

This detects 5 fingertips and hence five fingers.

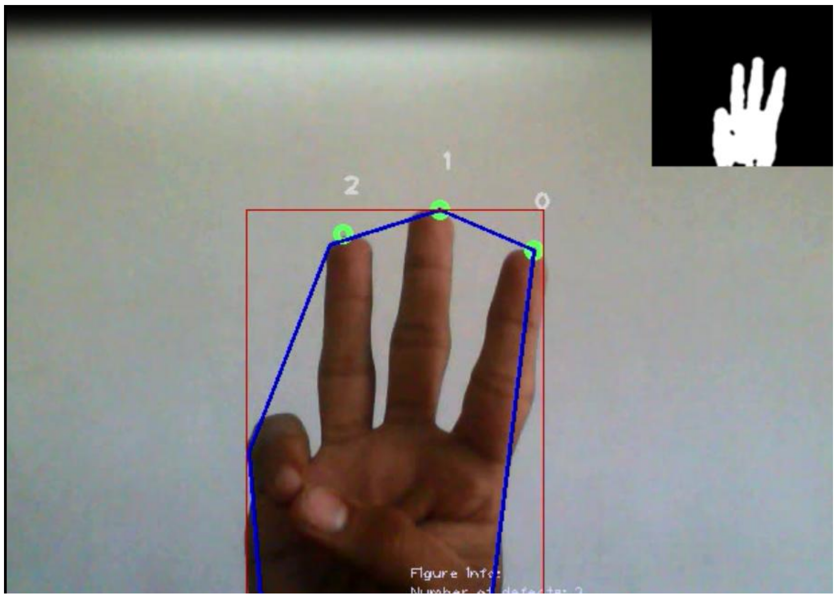


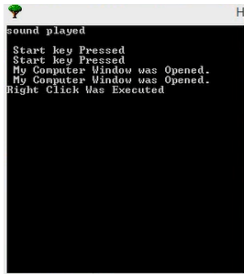


```
sound played

 Start key Pressed
 Start key Pressed
```

This detects 4 fingertips and hence four fingers.

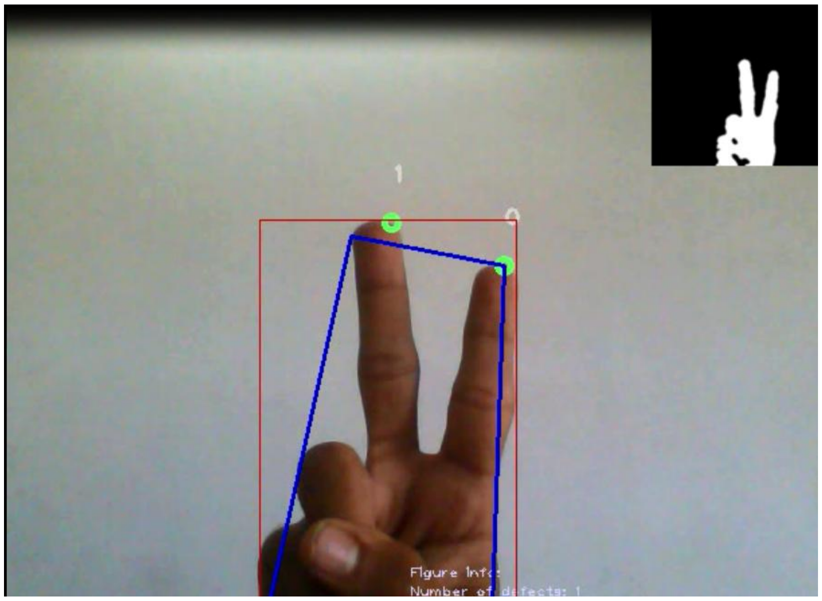


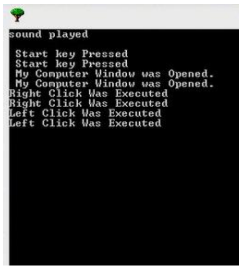


```
sound played

 Start key Pressed
 Start key Pressed
 My Computer Window was Opened.
 My Computer Window was Opened.
Right Click Was Executed
```

This detects 3 fingertips and hence three fingers.

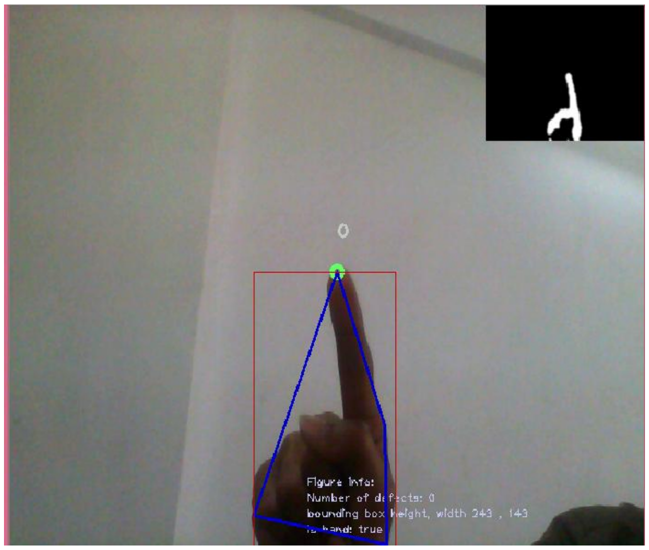




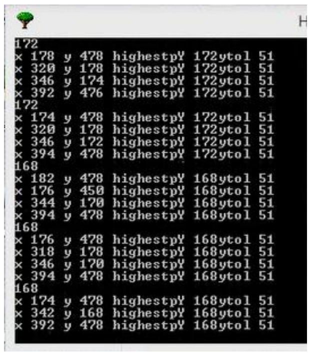
```
sound played

 Start key Pressed
 Start key Pressed
 My Computer Window was Opened.
 My Computer Window was Opened.
Right Click Was Executed
Right Click Was Executed
Left Click Was Executed
Left Click Was Executed
```

This detects 2 fingertips and hence two fingers.



(co-ordinates tracked for cursor movement)



```
172
x 178 y 478 highestpY 172ytol 51
x 320 y 178 highestpY 172ytol 51
x 346 y 174 highestpY 172ytol 51
x 392 y 476 highestpY 172ytol 51
172
x 174 y 478 highestpY 172ytol 51
x 320 y 178 highestpY 172ytol 51
x 346 y 172 highestpY 172ytol 51
x 394 y 478 highestpY 172ytol 51
168
x 182 y 478 highestpY 168ytol 51
x 176 y 450 highestpY 168ytol 51
x 344 y 170 highestpY 168ytol 51
x 394 y 478 highestpY 168ytol 51
168
x 176 y 478 highestpY 168ytol 51
x 318 y 178 highestpY 168ytol 51
x 346 y 170 highestpY 168ytol 51
x 394 y 478 highestpY 168ytol 51
168
x 174 y 478 highestpY 168ytol 51
x 342 y 168 highestpY 168ytol 51
x 392 y 478 highestpY 168ytol 51
```

This detects 1 fingertip and hence one finger.

PROJECT SPECIFICATIONS

Software Specifications:

- 64-bit Operating System: Windows 8 or Higher
- OpenCV 2.4.9 needs to be installed prior to running.
- Windows Administrator permissions are needed for some parts of the program to function properly.

Hardware Specifications:

- A Webcam

Environment Specifications:

- A clear white background
- There should be no other objects (specially skin coloured objects) in front of the webcam other than Palm.

CONCLUSION

The Histogram-based and Explicitly threshold skin detection methods were evaluated and based on the results, the Histogram method was deemed as more accurate. The vision based cursor control using hand gesture system was developed in the C++ language, using the OpenCV library. The system was able to 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 of the operations which were intended to be performed using various gestures were completed with satisfactory results.

FURTHER WORK

We would improve the performance of the software especially hand tracking in the near future. And we also want to decrease the response time of the software for cursor movement so that it can completely be used to replace our conventional mouse. We are also planning to design a hardware implementation for the same inorder to improve accuracy and increase the functionality to various domains such as a gaming controller or as a general purpose computer controller.

Other advanced implementation include the hand gesture recognition stage to use the Template Matching method to distinguish the hand gestures. This method requires the use of a machine learning classifier, which takes a considerably long time to train develop. However, it would have allow the use of lots more hand gestures which in turn would allow the use of more mouse functions such as zoom in and zoom out. Once the classifier is well trained, the accuracy of the Template Matching method is expected to be better than the method used in the proposed design.

Another novel implementation of this technology would to use the computer to train the visually or hearing impaired.

REFERENCES

1) OpenCV Website – www.opencv.org
2) MSDN Microsoft developers network – www.msdn.microsoft.com
3) Code project – www.codeproject.com/Articles/498193/Mouse-Control-via-Webcam
4) Aniket Tatipamula’s Blog - http://anikettatipamula.blogspot.in/2012/02/hand-gesture-using-opencv.html
5) Microsoft Research Paper- http://research.microsoft.com/en-us/um/people/awf/bmvc02/project.pdf