



VIRTUAL MOUSE USING HAND GESTURE RECOGNITION

Final Year Project Report

GROUP 8

Aditya Parikh(471)

Aniket Bharati(455)

Vishwa Desai(458)

VIRTUAL MOUSE USING HAND GESTURE RECOGNITION

SUBMITTED BY

**ADITYA PARIKH (471)
ANIKET BHARATI (455)
VISHWA DESAI (458)**

*Project Report
Submitted
In partial fulfillment for
The award of the degree of*

**BACHELOR OF ENGINEERING
[COMPUTER SCIENCE AND ENGINEERING]**



*Department of Computer Science and Engineering
Faculty of Technology and Engineering,
The Maharaja Sayajirao University of Baroda,
Vadodara
Year 2011-12*



*Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda*

CERTIFICATE

This is to certify that project work entitled
“Virtual Mouse Using Hand Gesture Recognition”
submitted by

ADITYA PARIKH (471)
ANIKET BHARATI (455)
VISHWA DESAI (458)

As per the fulfillment of the award of
DEGREE IN COMPUTER SCIENCE AND ENGINEERING
at the Faculty of Technology and Engineering ,
Maharaja Sayajirao University of Baroda ,
has been carried out
under supervision and guidance .

Guided By :
Mr.P.R.Bhavsar,
Head Of Department,
Dept. of Computer Science & Engineering ,
Faculty of Technology & Engineering,
Maharaja Sayajirao University of Baroda

Head Of Department,
Mr.P.R.Bhavsar,
Head,
Dept. of Computer Science & Engineering,
Faculty of Technology & Engineering,
Maharaja Sayajirao University of Baroda

Signature
(Head of Institute)

Seal of Institute

Acknowledgement

We wish to express our heartfelt appreciation to all those who have contributed to our project , both explicitly and implicitly , without the co-operation of whom , it would not have been possible to complete this project.

We are indebted and thankful to Mr.P.R.Bhavsar , H.O.D for his keen interest , untiring perseverance and unceasing motivation during the course of our project despite of his busy schedule .

We would also like to thank everyone for providing us this platform , which helped us immensely in applying the basic and fundamental knowledge in professional practice.

Last but not the least we would like to thank our family members and friends for their constant encouragement and thorough support through out the duration of our project.

ADITYA PARIKH (471)
ANIKET BHARATI (455)
VISHWA DESAI (458)

1. CONTENT

1. CONTENTS

2. INTRODUCTION

- 2.1. REPORT OVERVIEW
- 2.2. PROJECT SUMMARY
- 2.3. EXISTING SYSTEMS

3. TECHNICAL SPECIFICATIONS

4. DETECTION

- 4.1. CHOICE OF SENSORS
- 4.2. HARDWARE SETUP
- 4.3. CHOICE OF VISUAL DATA FORMAT
- 4.4. COLOR DETECTION METHOD

5. REFINEMENT

- 5.1. REMOVAL OF SKIN PIXELS DETECTED IN FORE-ARM

6. RECOGNITION

- 6.1. CHOICE OF RECOGNITION STRATEGY
- 6.2. SELECTION OF TEST GESTURE SET
- 6.3. ACTUAL GESTURE RECOGNITION METHOD

7. APPLICATION OF SYSTEM OPERATIONS

- 7.1. IMPLEMENTATION OF MOUSE OPERATIONS
- 7.2. IMPLEMENTATION OF EXTRA SYSTEM FUNCTIONS

8. APPLICATION USER INTERFACE

- 8.1. GESTURE SET
- 8.2. WORK FLOW OF APPLICATION
- 8.3. APPLICATION SETUP AND DEMONSTRATION

9. CONCLUSION

- 9.1. PROJECT GOALS
- 9.2. FUTURE PROSPECTS AND IMPROVEMENTS

2. INTRODUCTION

Gesture recognition is one which enables human to interface with the machine and interact naturally without any mechanical devices. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Current field include hand gesture recognition which is useful for processing information from humans which is not conveyed through speech or type. Gesture recognition can be conducted with techniques from computer vision and image processing.

The keyboard and mouse are currently the main interfaces between man and computer. In other areas where 3D information is required, such as computer games, robotics and design, other mechanical devices such as roller-balls, joysticks and data-gloves are used.

Humans communicate mainly by vision and sound, therefore, a man-machine interface would be more intuitive if it made greater use of vision and audio recognition. Another advantage is that the user not only can communicate from a distance, but need have no physical contact with the computer. However, unlike audio commands, a visual system would be preferable in noisy environments or in situations where sound would cause a disturbance. The visual system chosen was the recognition of hand gestures. The amount of computation required to process hand gestures is much greater than that of the mechanical devices, however standard desktop computers are now quick enough to make this project — hand gesture recognition using computer vision — a viable proposition.

2.1 Report Overview

- The system uses the integrated webcam of the laptop or an external USB webcam connected to the computer desktop . The output of the camera is displayed on the monitor . The shape and gesture of the hand is detected using an algorithm .
- The shape information will then be refined using spatial knowledge of the hand.
- The refined shape information will then be compared with a set of predefined training data . The corresponding system operation will then be performed .
- The remaining part of the report describes the application and setup of the system on a windows based user interface .

2.2 Project Summary

In order to detect hand gestures, data about the hand will have to be collected. A decision has to be made as to the nature and source of the data. Two possible technologies to provide this information are:

- A glove with sensors attached that measure the position of the finger joints.
- An optical method.

An optical method has been chosen, since this is more practical (many modern computers come with a camera attached), cost effective and has no moving parts, so is less likely to be damaged through use.

The first step in any recognition system is collection of relevant data. In this case the raw image information will have to be processed to differentiate the skin of the hand (and various markers) from the background.

Once the data has been collected it is then possible to use prior information about the hand to refine the data and remove as much noise as possible. This step is important because as the number of gestures to be distinguished increases the data collected has to be more and more accurate and noise free in order to permit recognition.

The next step will be to take the refined data and determine what gesture it represents. Any recognition system will have to simplify the data to allow calculation in a reasonable amount of time . Obvious ways to simplify the data include translating, rotating and scaling the hand so that it is always presented with the same position, orientation and effective hand-camera distance to the recognition system.

2.3 Existing Systems

Figure shows several of the existing gesture recognition systems along with recognition statistics and method.

Paper	Primary method of recognition	Number of gestures recognized	Background to gesture images	Additional markers required	Number of training images	Accuracy	Frame rate
[Bauer & Hienz, 2000]	Hidden Markov Models	97	General	Multicoloured gloves	7-hours signing	91.7%	--
[Starner, Weaver & Pentland, 1998]	Hidden Markov Models	40	General	No	400 training sentences	97.6%	10
[Bowden & Sarhadi, 2000]	Linear approximation to non-linear point distribution models	26	Blue screen	No	7441 images	--	--
[Davis & Shah, 1994]	Finite state machine / model matching	7	Static	Markers on glove	10 sequences of 200 frames each	≈98%	10

3. TECHNICAL SPECIFICATIONS

- **Operating System :**

The system works best with a Microsoft Windows 7 64-bit operating system which supports execution of applications developed in Microsoft Visual Studio. It also works on a Microsoft Windows 7 32-bit operating system .

- **Application Programming Interface :**

Microsoft Visual Studio 2010 is the programming software used for development and execution of the system . It supports languages included in the .NET framework such as Visual Basic, C++, C# and J#. The system is developed in C# programming language .

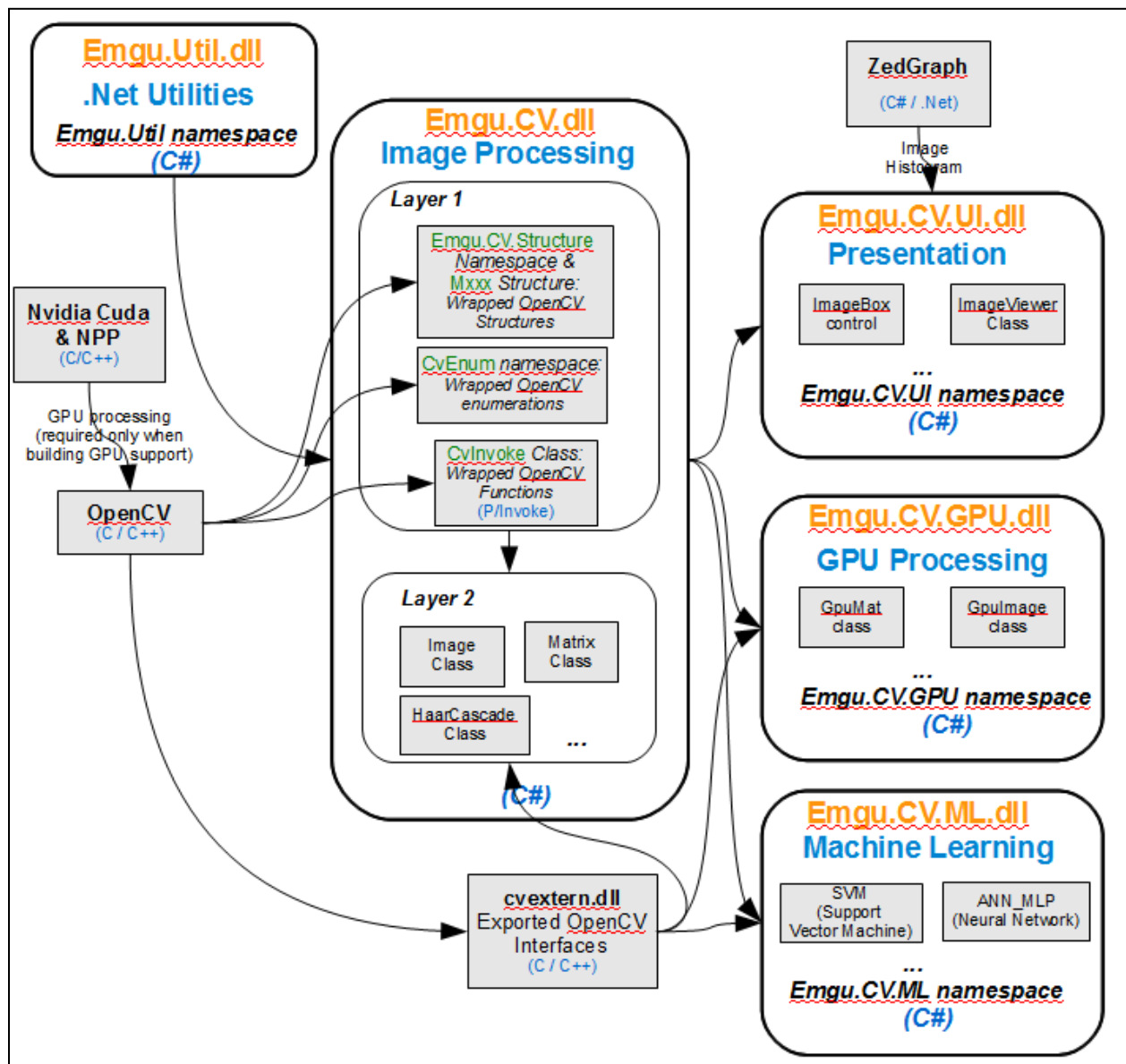
- **Emgu CV Wrapper :**

EmguCV is a cross platform .Net wrapper to the Intel OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc. The wrapper can be compiled in Mono and run on Linux / Mac OS X.

Emgu CV has two layers of wrapper as shown below :

- The basic layer (**layer 1**) contains functions , structure and enumeration mappings which directly reflect those in OpenCV .
- The second layer (**layer 2**) contains classes that mix in advantages from the .NET world.

The diagram below depicts these layer of the EmguCV wrapper .



4. DETECTION

In order to recognize hand gestures it is first necessary to collect information about the hand from raw data provided by any sensors used. This section deals with the selection of suitable sensors and compares various methods of returning only the data that pertains to the hand.

4.1 Choice of sensors

Since the hand is by nature a three dimensional object the first optical data collection method considered was a stereographic multiple camera system. Alternatively, using prior information about the anatomy of the hand it would be possible to garner the same gesture information using either a single camera or multiple two dimensional views provided by several cameras. These three options are considered below:

Stereographic system: The stereographic system would provide pixilated depth information for any point in the fields of view of the cameras. This would provide a great deal of information about the hand. Features that would otherwise be hard to distinguish using a 2D system, such as a finger against a background of skin, would be differentiable since the finger would be closer to camera than the background. However the 3D data would require a great deal of processor time to calculate and reliable real-time stereo algorithms are not easily obtained or implemented.

Multiple two dimensional view system: This system would provide less information than the stereographic system and if the number of cameras used was not great, would also use less processor time. With this system two or more 2D views of the same hand, provided by separate cameras, could be combined after gesture recognition. Although each view would suffer from similar problems to that of the “finger” example above, the combined views of enough cameras would reveal sufficient data to approximate any gesture.

Single camera system: This system would provide considerably less information about the hand. Some features (such as the finger against a background of skin in the example above) would be very hard to distinguish since no depth information would be recoverable.

Essentially only “silhouette” information could be accurately extracted. The silhouette data would be relatively noise free (given a background sufficiently distinguishable from the hand) and would require considerably less processor time to compute than either multiple camera system.

It is possible to detect a large subset of gestures using silhouette information alone and the single camera system is less noisy, expensive and processor hungry. Although the system exhibits more ambiguity than either of the other systems, this disadvantage is more than outweighed by the advantages mentioned above. Therefore, it was decided to use the single camera system.

4.2 Hardware setup

The output of the camera system chosen in Section 3.1 comprises of a 2D array of RGB pixels provided at regular time intervals. In order to detect silhouette information it will be necessary to differentiate skin from background pixels. It is also likely that other markers may be needed to provide extra information about the hand and the marker pixels will also have to be differentiated from the background (and skin pixels). To make this process as achievable as possible it is essential that the hardware setup is chosen carefully.

The various options are discussed below.

Lighting: The task of differentiating the skin pixels from those of the background and markers is made considerably easier by a careful choice of lighting. If the lighting is constant across the view of the camera then the effects of self-shadowing can be reduced to a minimum. The intensity should also be set to provide sufficient light for the CCD in the camera.

However, since this system is intended for a user it would be a disadvantage if special lighting equipment was required. It was decided to attempt to extract the hand and information using standard room lighting. This would permit the system to be used in a non-specialist environment.

Camera orientation: It is important to carefully choose the direction in which the camera points to permit an easy choice of background. The realistic option is to point the camera towards a wall . However the light intensity would be higher and shadowing effects least if the camera was pointed in a direction with source of light behind it.

Background: In order to maximize differentiation it is important that the color of the background differs as much as possible from that of the skin.

4.3 Choice of visual data format

An important trade-off when implementing a computer vision system is to select whether to differentiate objects using color or black and white and, if color, to decide what color space to use (red, green, blue or hue, saturation, luminosity). For the purposes of this project, the detection of skin pixels is required, so the color space chosen should best facilitate this.

Color or black and white: The camera and video card available permitted the detection of color information. Although using intensity alone (black and white) reduces the amount of data to analyze and therefore decreases processor load it also makes differentiating skin and markers from the background much harder (since black and white data exhibits less variation than color data). Therefore it was decided to use color differentiation.

RGB or HSL: The raw data provided by the video card was in the RGB (red, green, blue) format. However, since the detection system relies on changes in color (or hue), it could be an advantage to use HSL (hue, saturation, luminosity) to permit the separation of the hue from luminosity (light level). To test this the maximum and minimum HSL pixel color values of a small test area of skin were manually calculated. These HSL ranges were then used to detect skin pixels in a subsequent frame (detection was indicated by a change of pixel color to white). The test was carried out three times using either hue, saturation or luminosity color ranges to detect the skin pixels. Next, histograms were drawn of the number of skin pixels of each value of hue, saturation and luminosity within the test area.

Histograms were also drawn for an equal sized area of non-skin pixels. The histogram test was repeated using the RGB color space. Hue, when compared with saturation and luminosity, is surprisingly bad at skin differentiation (with the chosen background) and thus HSL shows no significant advantage over RGB.

Moreover, since conversion of the color data from RGB to HSL took considerable processor time it was decided to use RGB.

YCbCr : YCbCr, Y'CbCr, or Y Pb/Cb Pr/Cr, also written as $YC_B C_R$ or $Y' C_B C_R$, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y' is the luma component and C_B and C_R are the blue-difference and red-difference chroma components. Y'(with prime) is distinguished from Y which is luminance, meaning that light intensity is non-linearly encoded using gamma correction.

Y'CbCr is not an absolute color space; rather, it is a *way of encoding* RGB information. The actual color displayed depends on the actual RGB primaries used to display the signal. Therefore a value expressed as Y'CbCr is predictable only if standard RGB primary chromaticity are used.

Conversion from RGB values to the corresponding YCbCr values can be done according to the following formulae :

$$\begin{aligned}
Y' &= \quad + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D) \\
C_B &= 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D) \\
C_R &= 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)
\end{aligned}$$

And back:

$$\begin{aligned}
R &= Y \quad + 1.402 \cdot (C_R - 128) \\
G &= Y - 0.34414 \cdot (C_B - 128) - 0.71414 \cdot (C_R - 128) \\
B &= Y + 1.772 \cdot (C_B - 128)
\end{aligned}$$

4.4 Color Detection Method

- **Binary segmentation algorithm**

There are several image segmentation techniques that we could have used to segment the hand from the background, but the one implemented is comparing the current image to a stored background image of the keyboard. This image is taken when the program first starts running, at a time when the hands are not over the keyboard.

To segment the image, we compared the pixel values between the background image and the current frame image. If the pixel had changed significantly in value then, it is be labeled as 'hand' and set to be white. If the value has not changed significantly, then it can be labeled as 'background' and set to be black.

- **Skin detection algorithm**

Our program hinges on obtaining a quality image segmentation to be able to detect pinch gestures. After much trial and error, we discovered that using just binary image segmentation was not enough.

As a solution, we implemented a second method to analyze the image by using skin detection to detect color values within a certain range. This analysis was done before the binary segmentation, and the results of the two processes were added together to produce a final segmented image, with all background pixels being black, and all hand pixels being white.

This section has described the choice and setup of hardware and detection in order to detect as many of the skin pixels within the frame as possible. The hardware chosen was a single color camera pointing towards a wall surface of a constant color with no special lighting. Detection is performed by scanning the RGB color values of pixels within a preset area of the frame . Calculating the YCbCr values and comparing it with the standard values of skin color , a hull is extracted from the frame and a final contour of the points which represent the hand portion in the frame (largest contour) is obtained .

5. REFINEMENT

Using the methods discussed in the previous section it is possible to detect the majority of the skin pixels in the frame whilst detecting aberrant pixels in the background.

However, some complications were noticed which could reduce the accuracy of recognition at a later stage. These are:

- ***Image distortion:*** If the camera's visual axis is not parallel to the floor plane, a given gesture would appear different depending on the position and yaw of the hand (a given length in one area of the frame would appear longer or shorter in another area of the frame). This is termed projective distortion. Also, if the camera lens is of poor quality then the straight sides of a true square in the frame would appear curved. This is termed radial distortion.
- ***Skin pixels of the arm being detected:*** Any skin pixels above the wrist band will also be detected as skin. It would be preferable if these pixels could be ignored, as they play no part in the gesture. Wearing a long sleeve top helps solve the problem but forearm pixels are still detected between the wrist and the sleeve (which has a tendency to move up and down the arm as different gestures are made, leading to variations in the amount of skin detected).

Image distortion is not handled by this system . It depends on the user to position the webcam properly and using a high definition camera would yield a better result .

5.1 Removal of Skin Pixels detected in Forearm

The Contour of the points of the whole hand including the arm is obtained as explained as in Section 4 . This contour is processed to extract another sub-contour which contains the points only depicting the palm . This is done using the concept of convex hull , convex set and convexity defects .

The points in hand contour are searched for convexity defects . If the defects have the value of convexity depth greater than a threshold value then that defect is considered useful in removing the skin part of fore arm . The height of the palm is decided based on geometric ratio of the forearm to the palm and the defect depth value . Then the points in the main contour are iterated moving clockwise and those that satisfy the conditions are included to a sub-contour which is the required palm contour . Then based on the height of the bounding box of the palm contour we can even determine its centre which is useful in further applications .

6.RECOGNITION

6.1 Choice of recognition strategy

Two methods present themselves by which a given gesture could be recognized from two dimensional “silhouette” information:

- ***Direct method based on geometry:*** Knowing that the hand is made up of bones of fixed width connected by joints which can only flex in certain directions and by limited angles it would be possible to calculate the silhouettes for a large number of hand gestures. Thus, it would be possible to take the silhouette information provided by the detection method and find the most likely gesture that corresponds to it by direct comparison. The advantages of this method are that it would require very little training and would be easy to extend to any number of gestures as required. However, the model for calculating the silhouette for any given gesture would be hard to construct and in order to attain a high degree of accuracy it would be necessary to model the effect of all light sources in the room on the shadows cast on the hand by itself.
- ***Learning method:*** With this method the gesture set to be recognized would be “taught” to the system beforehand. Any given gesture could then be compared with the stored gestures and a match score calculated. The highest scoring gesture could then be displayed if its score was greater than some match quality threshold. The advantage of this system is that no prior information is required about the lighting conditions or the geometry of the hand for the system to work, as this information would be encoded into the system during training. The system would be faster than the above method if the gesture set was kept small. The disadvantage with this system is that each gesture would need to be trained at least once and for any degree of accuracy, several times. The gesture set is also likely to be user specific.

It was decided to proceed with the direct computation based on geometry method for reasons of computation speed and ease of implementation.

6.2 Selection of Test Gesture Set

The gesture set that we decided was based on ease of use for the customer . It is based on simple hand gestures that a person makes in day to day life . We have used gestures that do not have significant differences , which made the use easy but the implementation difficult to program.

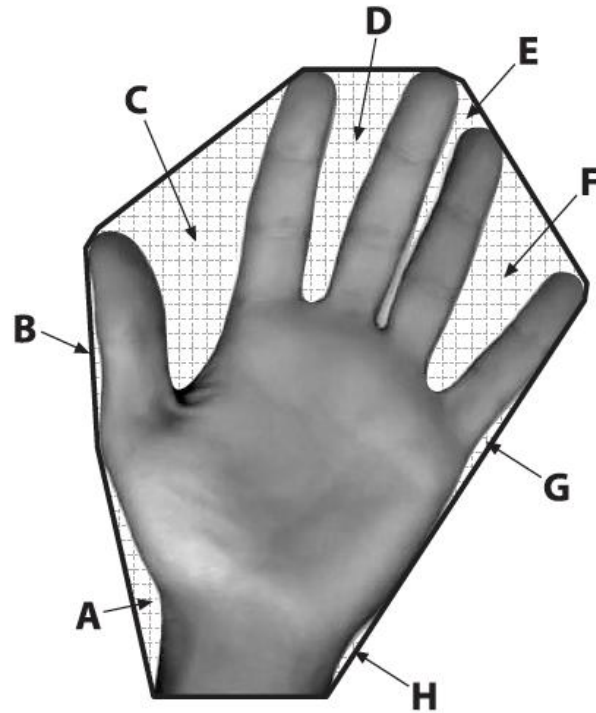
6.3 Actual Gesture Recognition Method

The concepts of convexity defects mentioned in above Section 5 is used for implementation of the geometry based calculations for gesture recognition. The convex hull ,convex set and the convexity defect in defined as below.

Convex hull: for a set of points X in a space S (2- or 3-dimensional space) is the minimal convex set containing X .

Convex Set: consist of pair of points within the object such that every point on the straight line segment that joins them is also within the object.

Convexity Defect: It is the set of points in the convex hull that do not lie in the considered object (i.e. hand).



The regions A to H depict the convexity defects in the convex hull A-B-C-D-E-F-G-H .

The convexity defect is made up of three points :

1. Starting point
2. Ending Point
3. Depth Point

Each of these points depict a tuple of (X,Y) co-ordinates . Then based on the geometric manipulations done on these co-ordinates we can define various gesture patterns and thus we obtain a set of gestures . These co-ordinates mentioned above are actual screen co-ordinates . Considering the algorithm we apply on it to detect gestures , we can detect number of fingers and various other gestures of hand , length of the fingers , position of tip of fingers , position of centre of the palm and various other characteristics which help in implementing the system operations of mouse and special function keys .

7.APPLICATION OF SYSTEM OPERATIONS

7.1 Implementation of Mouse Operations

This section describes how we have implemented the various mouse operations using a hook that is specific to a thread and a hook procedure by using the mouse hook as an example. You can use hooks to monitor certain types of events. You can associate these events with a specific thread or with all the threads in the same desktop as a calling thread.

To set a hook, call the SetWindowsHookEx function from the User32.dll file. This function installs an application-defined hook procedure in the hook chain that is associated with the hook.

To install a global hook, a hook must have a native DLL export to inject itself in another process that requires a valid, consistent function to call into. This behavior requires a DLL export. The .NET Framework does not support DLL exports. Managed code has no concept of a consistent value for a function pointer because these function pointers are proxies that are built dynamically. Low-level hook procedures are called on the thread that installed the hook. Low-level hooks do not require that the hook procedure be implemented in a DLL. There is one disadvantage that except for the WH_KEYBOARD_LL low-level hook and the WH_MOUSE_LL low-level hook, you cannot implement global hooks in the Microsoft .NET Framework.

The following table shows the symbolic constant names, hexadecimal values, and mouse equivalents for the virtual-key codes used by the system.

DESCRIPTION	VALUE
LEFTDOWN	0x00000002
LEFTUP	0x00000004
MIDDLEDOWN	0x00000020
MIDDLEUP	0x00000040
MOVE	0x00000001
ABSOLUTE	0x00008000
RIGHTDOWN	0x00000008
RIGHTUP	0x00000010

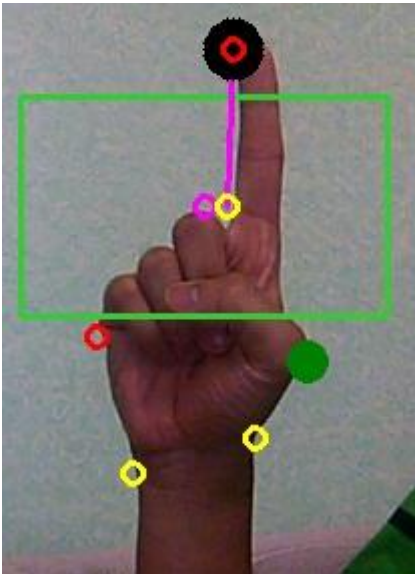
7.2 Implementation of Extra System functions

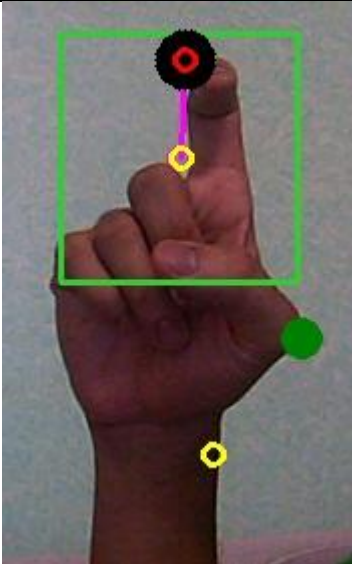
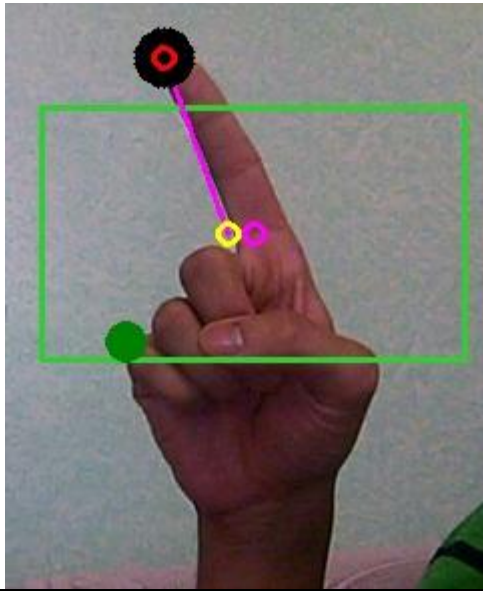
The implementation of special functions like volume up , volume down , scroll vertical , scroll horizontal and others is also done using global constants and their values . The following table shows the symbolic constant names, hexadecimal values, and keyboard equivalents for the virtual-key codes used by the system. The codes are listed in numeric order.

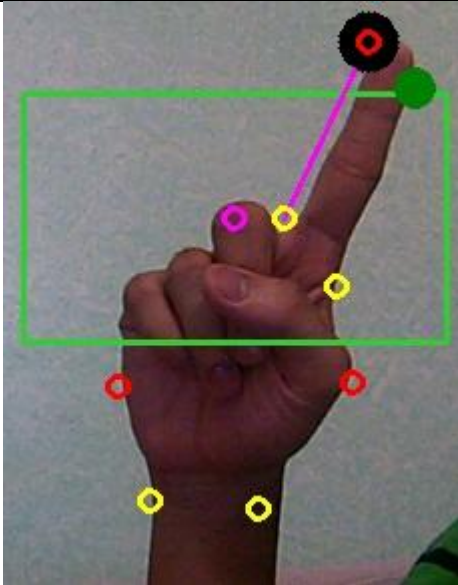
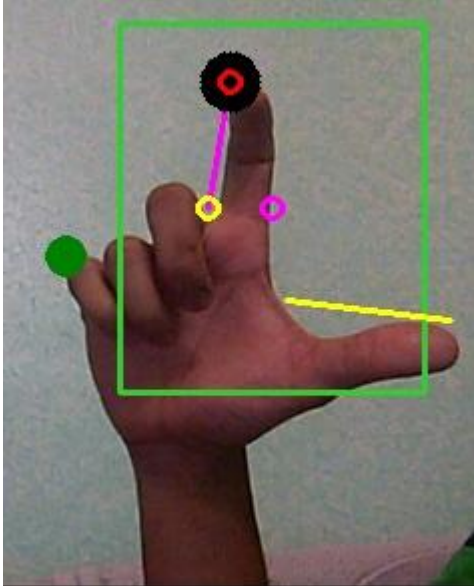
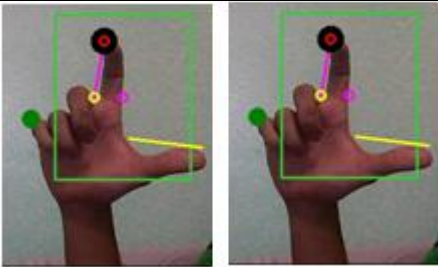
DESCRIPTION	VALUE
VK_VOLUME_UP	0xAF
VK_VOLUME_DOWN	0xAE
VK_SNAPSHOT	0x2C
MOUSEEVENTF_WHEEL	0x800
MOUSEEVENTF_HWHEEL	0x1000
WHEEL_DELTA	Step value (eg. 120)

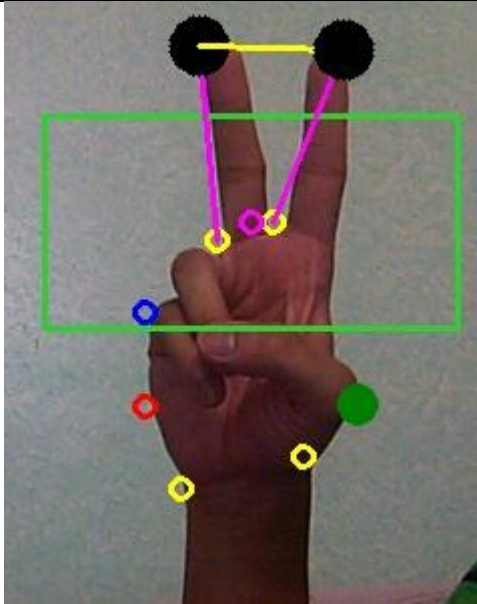
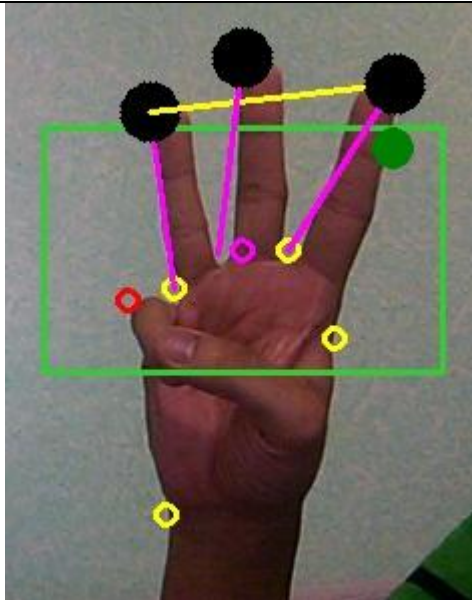
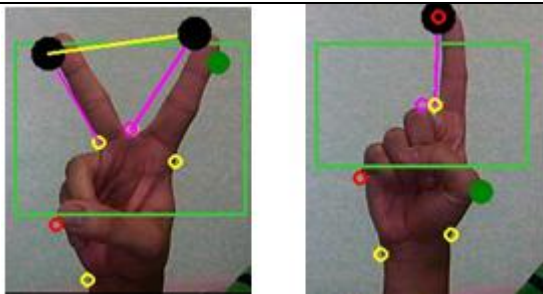
8.APPLICATION USER INTERFACE

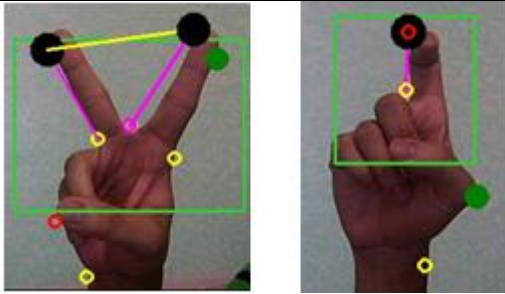
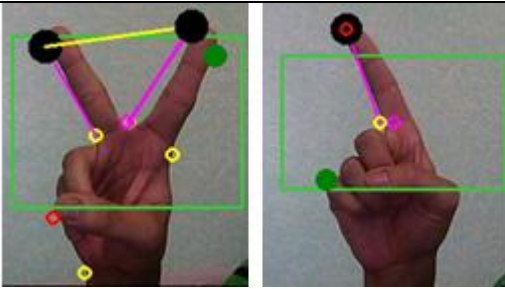
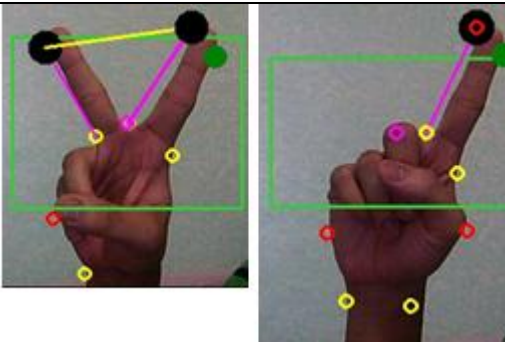
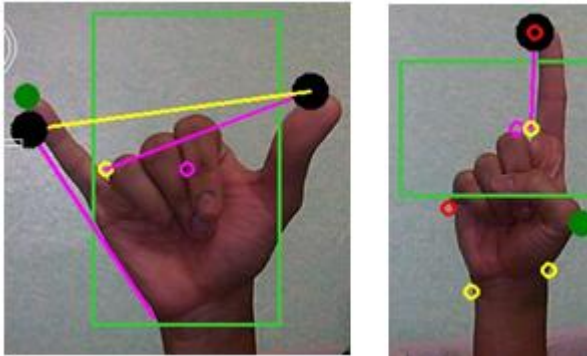
8.1 Gesture Set

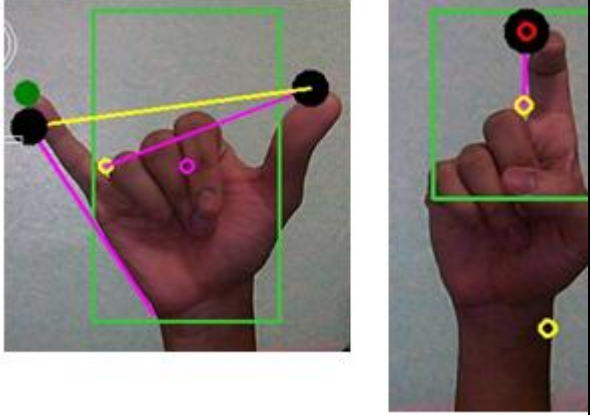
<u>HAND GESTURE SET</u>			
<u>SR.N</u> <u>O</u>	<u>GESTURE(S)</u>	<u>FUNCTIO</u> <u>N</u>	<u>INDIVIDUAL/</u> <u>COMBINATI</u> <u>ON</u>
1.		Move Cursor Up	Individual

2.		Move Cursor Down	Individual
3.		Move Cursor Left	Individual

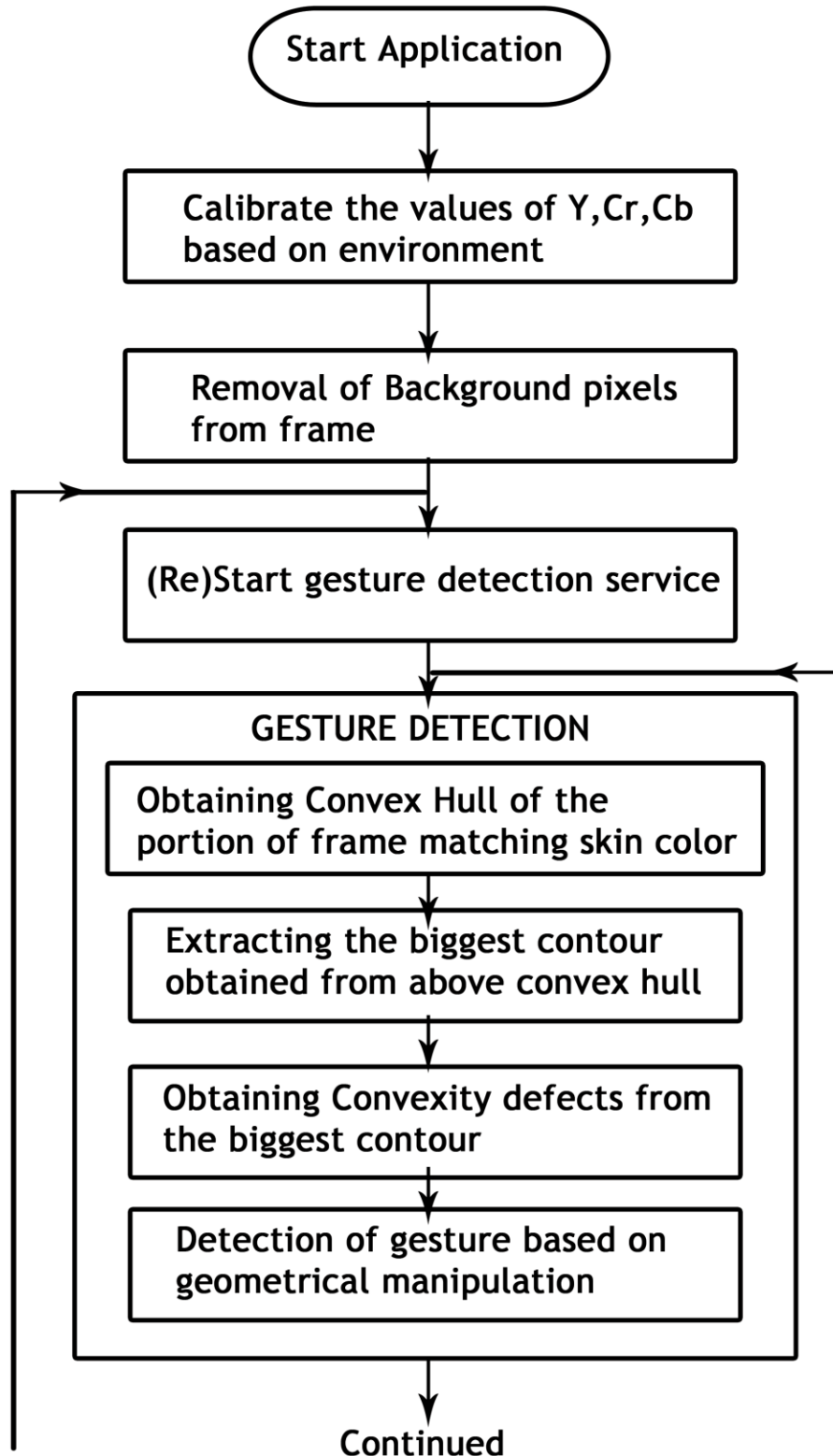
4.		Move Cursor Right	Individual
5.		Left Single Click	Individual
6.		Left Double Click	Combination

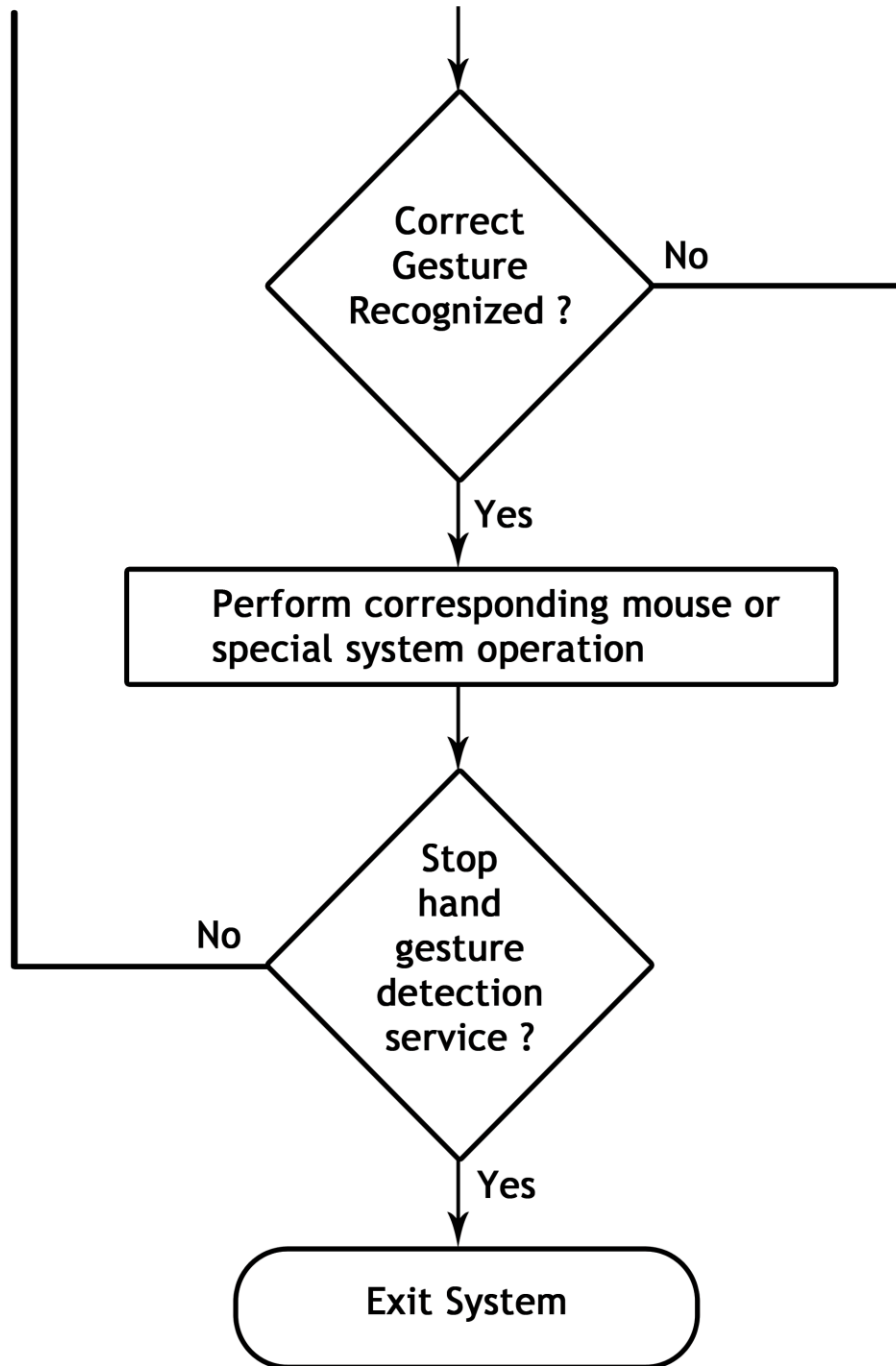
7.		Right Click	Individual
8.		Toggle Drag	Combination
9.		Scroll Up	Combination

10.		Scroll Down	Combination
11.		Scroll Left	Combination
12.		Scroll Right	Combination
13.		Volume Up	Combination

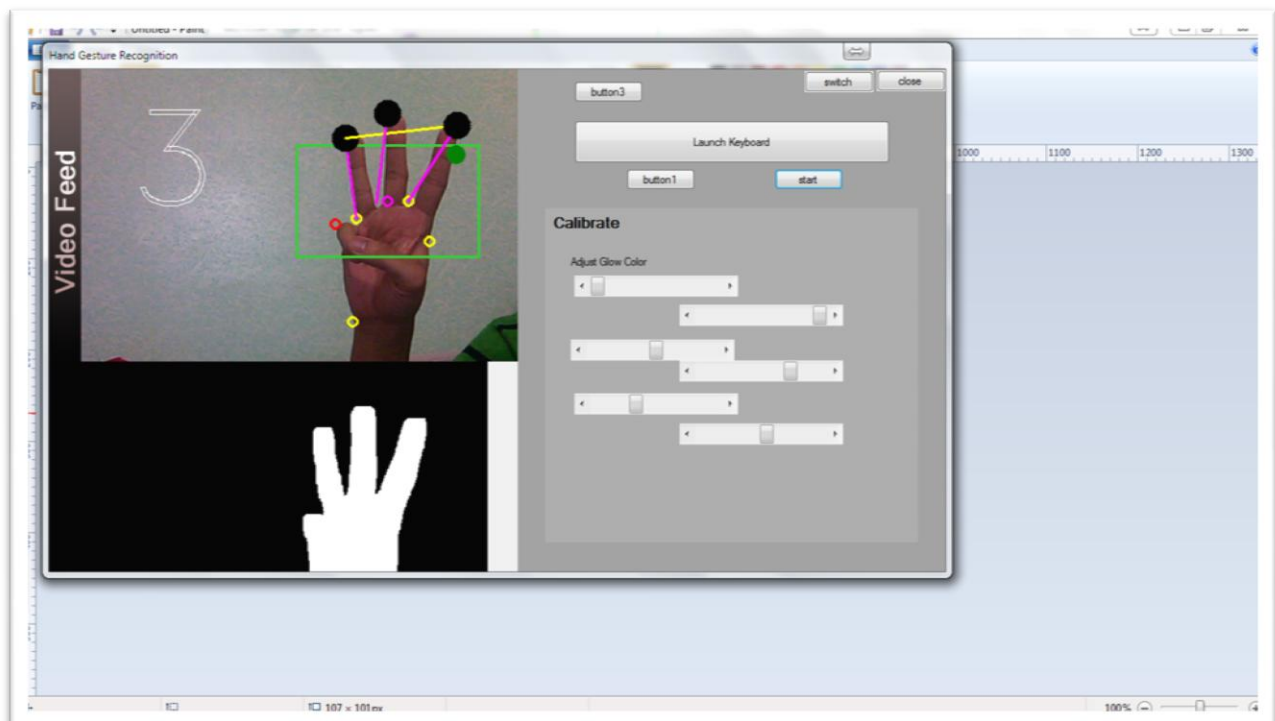
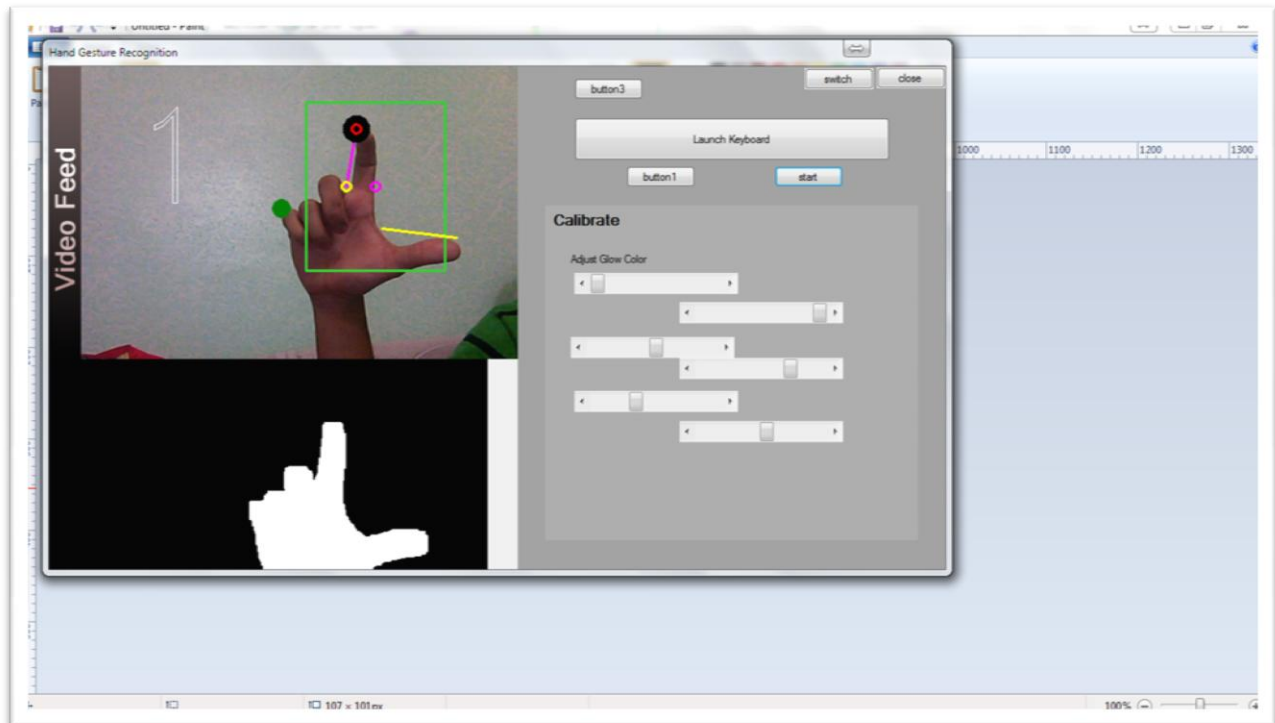
14.		Volume Down	Combination
-----	---	----------------	-------------

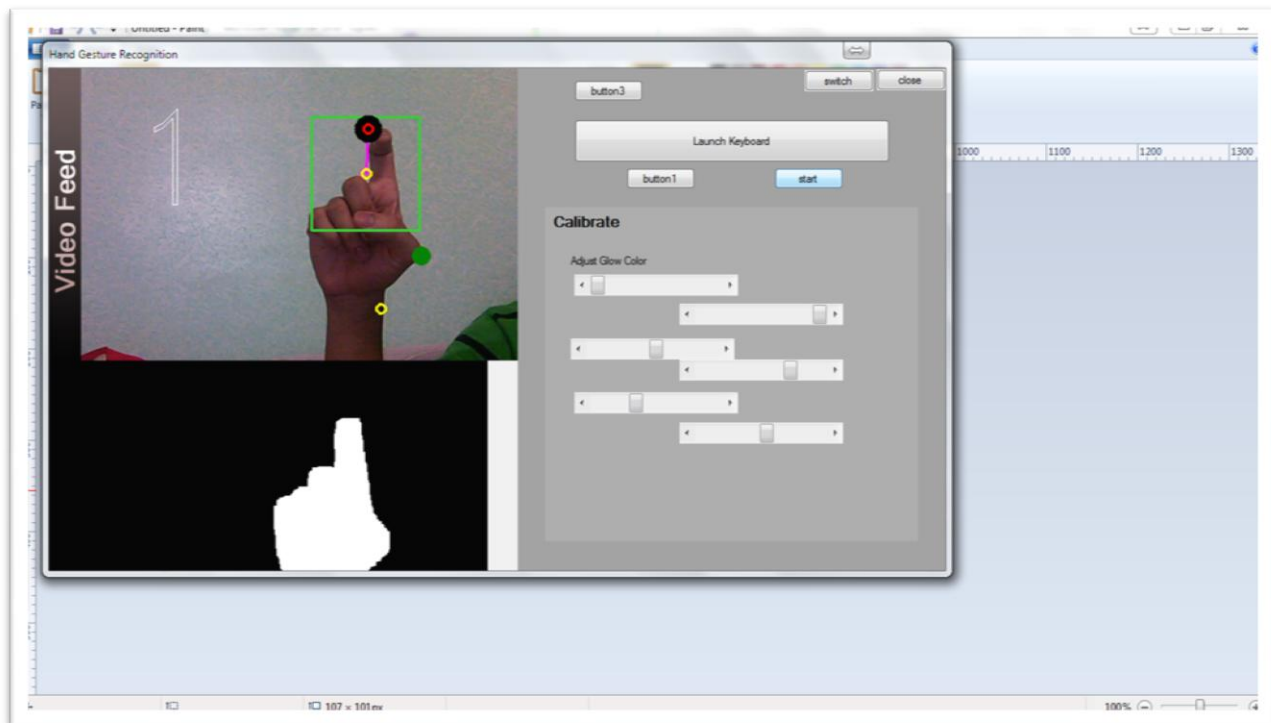
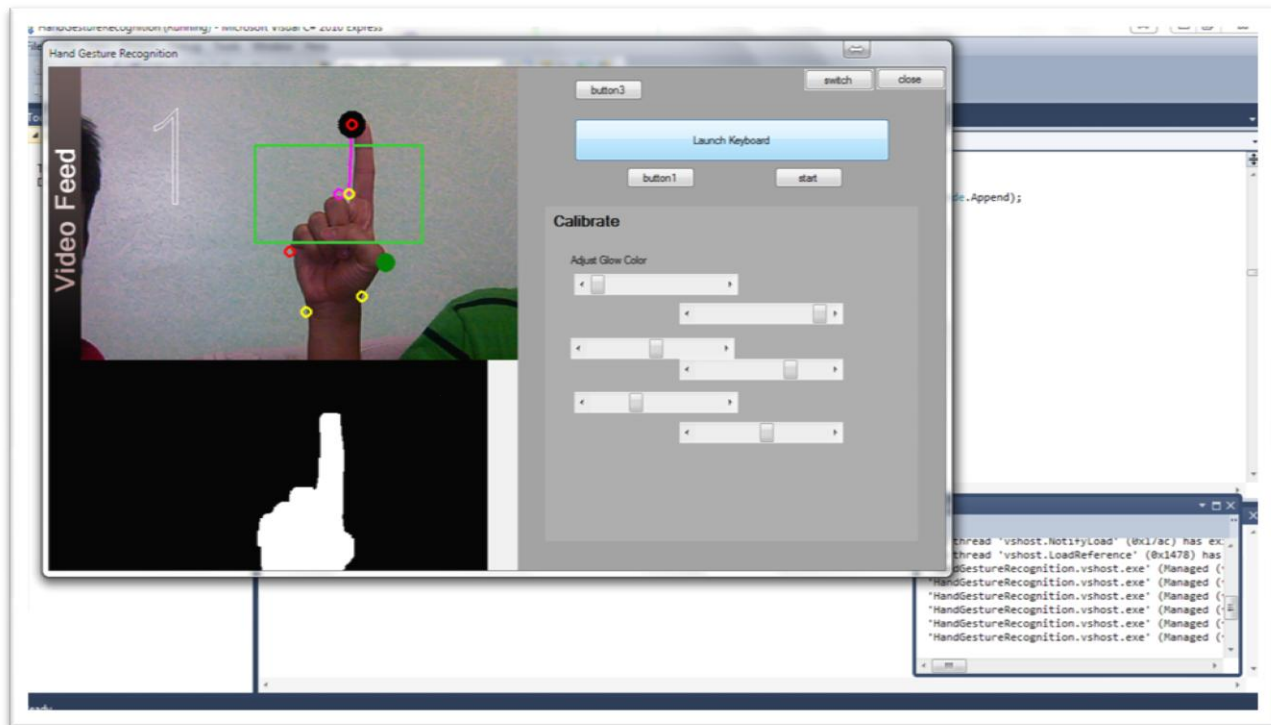
8.2 WorkFlow

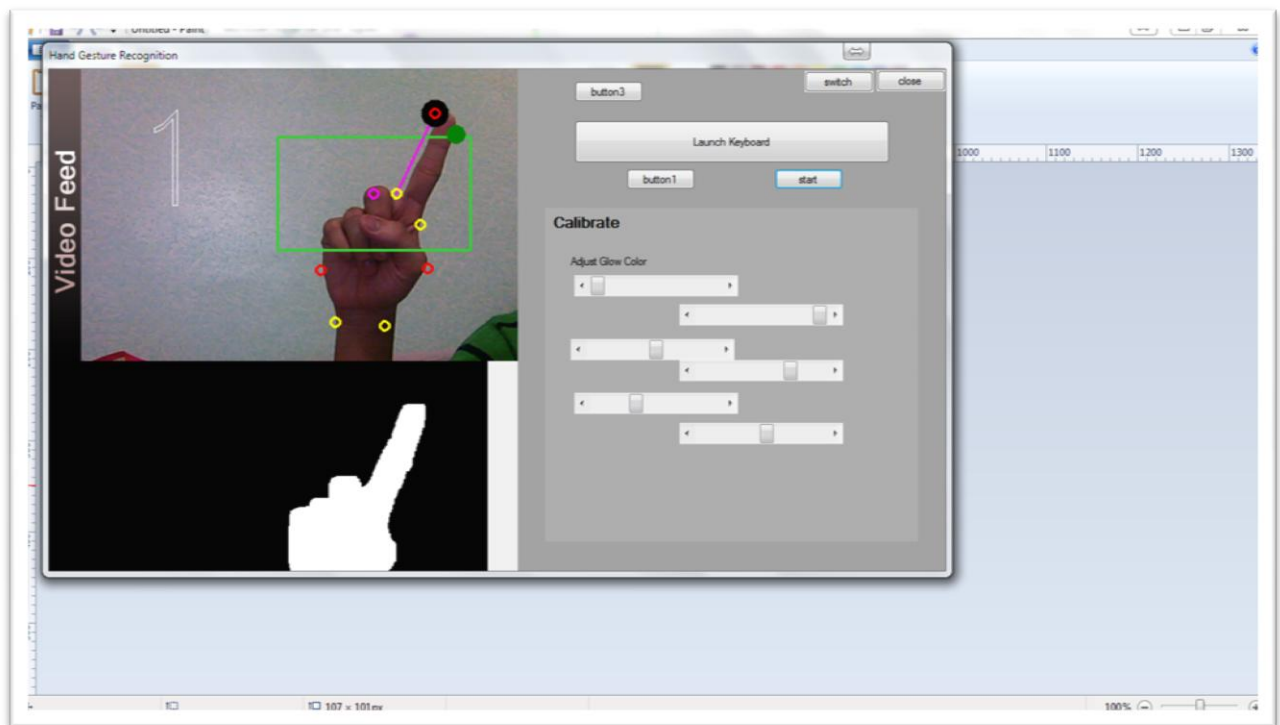
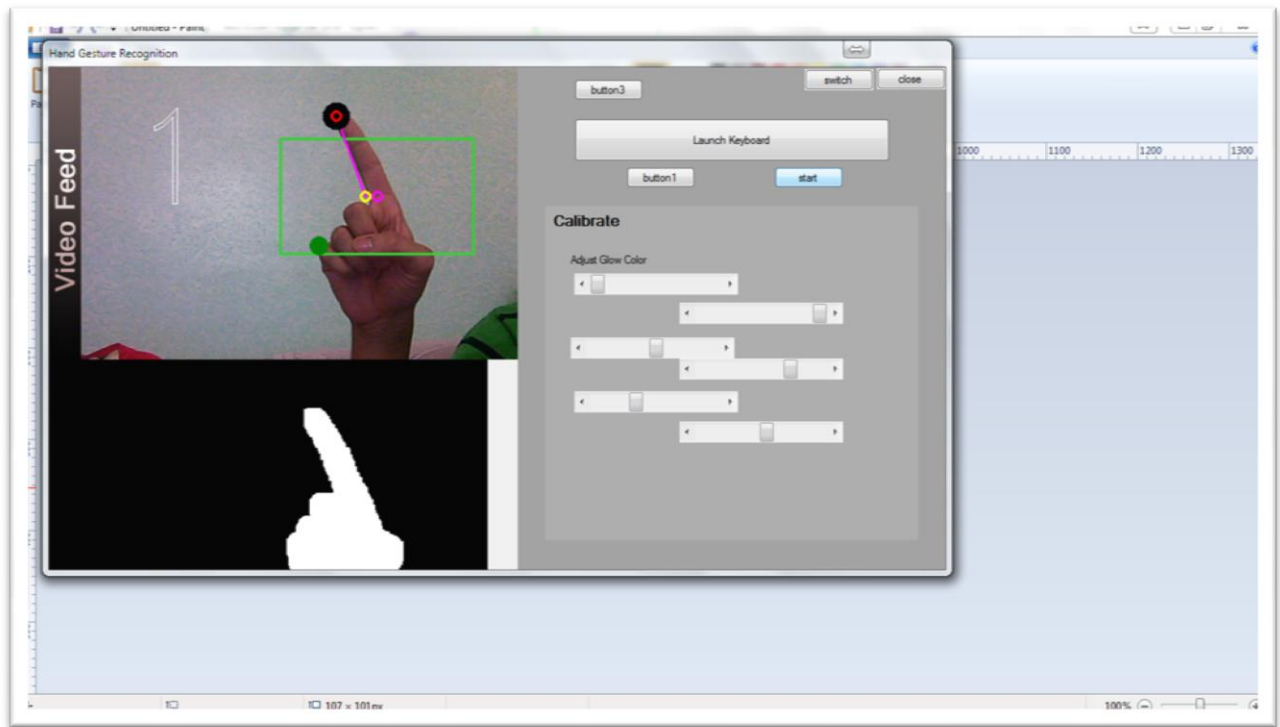


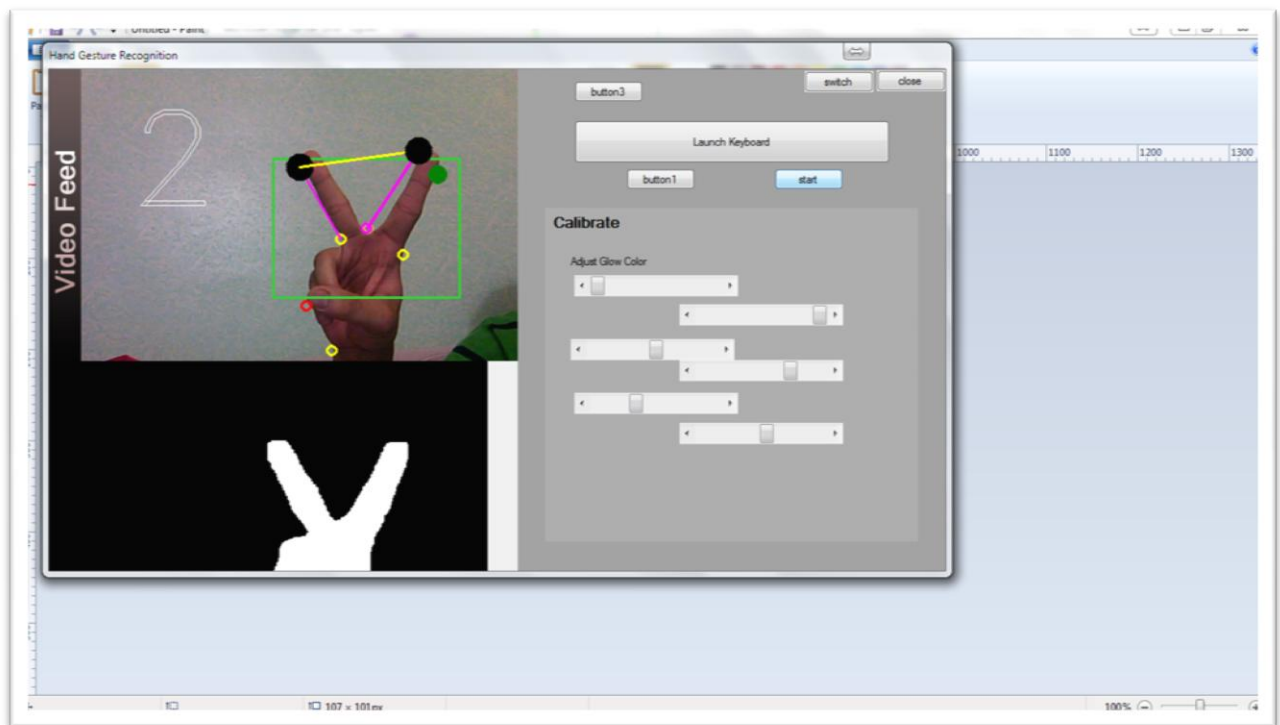
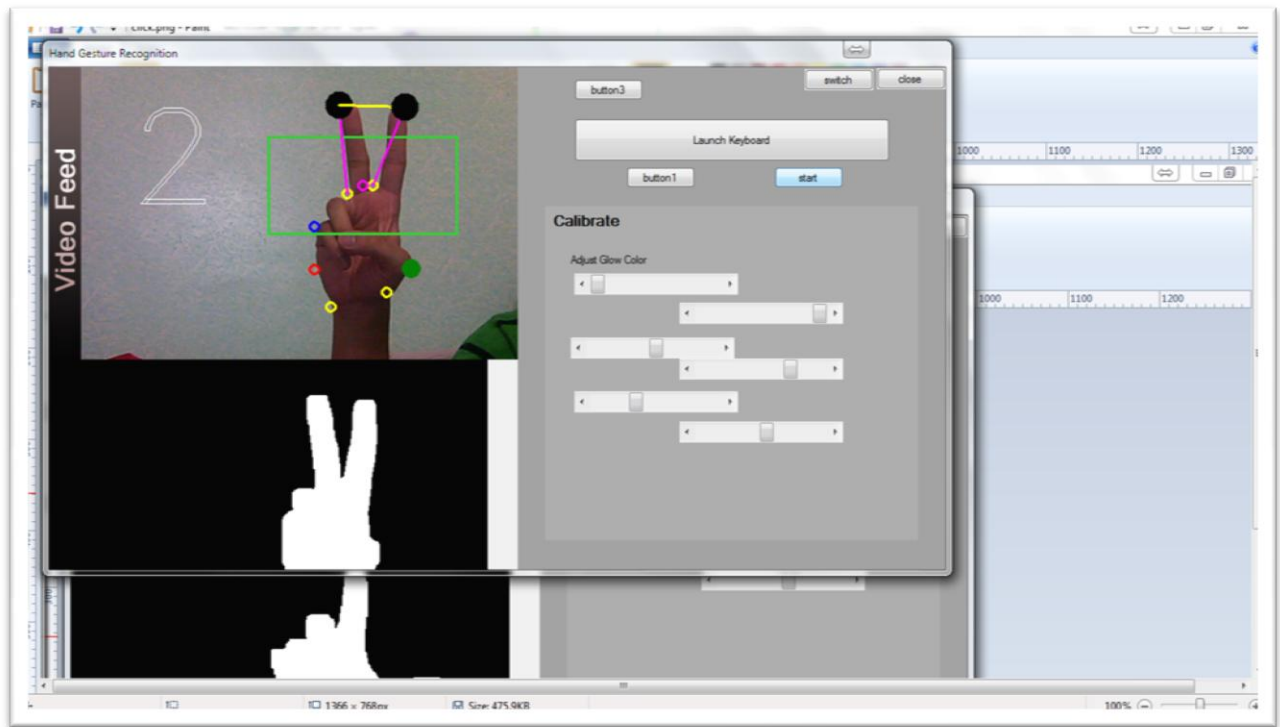


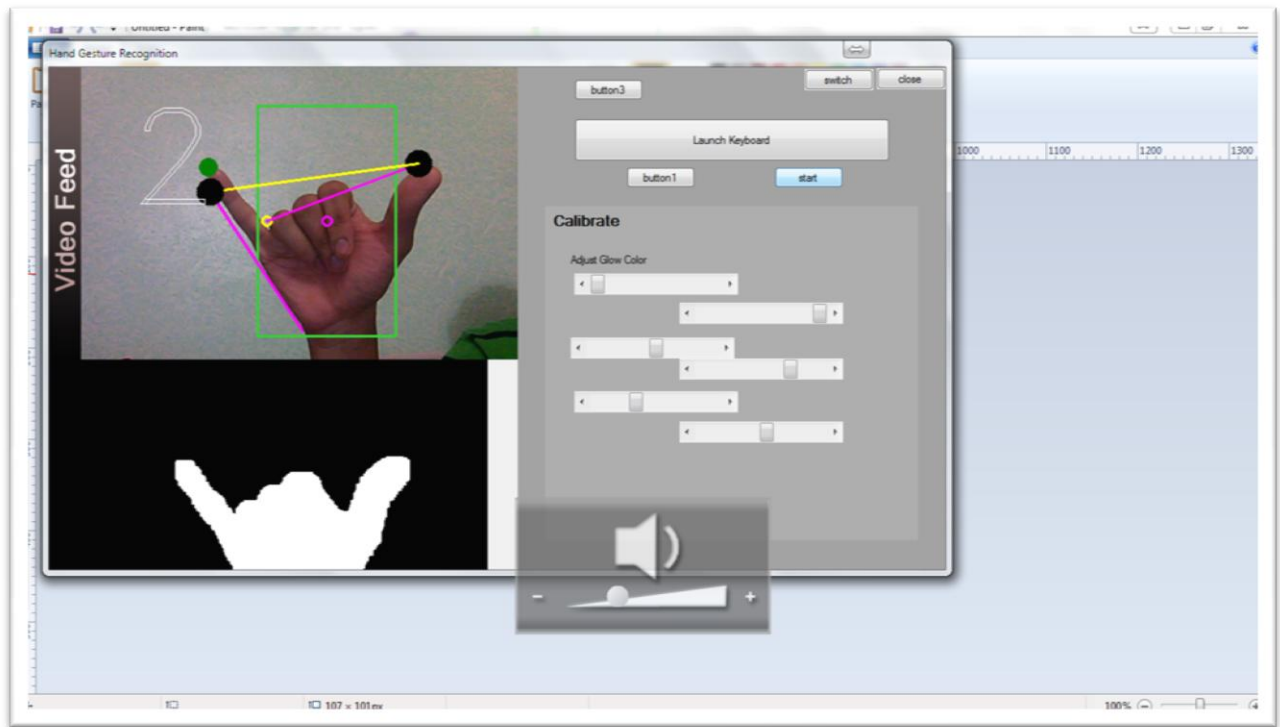
8.3 APPLICATION SETUP AND DEMONSTRATION











9.CONCLUSION

9.1 Project Goals

The goal of this project was to create a system to recognize a set of gestures at a rate of 25fps. It was considered that a modern computer system would allow the project goals to be exceeded. Furthermore, the system performance is amongst the best reported in existing literature.

9.2 Future Prospects and Improvement

- ***Collection of additional gesture information:*** The final system developed recognized gestures using silhouette information alone. Although this was sufficient for the number of trained gestures, the accuracy would doubtless suffer if the number of gestures were increased. In order to remedy this, extra information about the test gesture would have to be gathered, such as edge information.
- ***Increase of the number of recognized gestures:*** For the purposes of a man-machine interface a relatively small set of gestures (≈ 20) would be sufficient and is therefore within the bounds of the final system developed. A system which relies on both training and comparison of all gestures used would not be sufficient for this task. Further work, therefore, could involve the implementation of a gesture recognition system which does not require training.
- ***Two-handed sign language:*** It would be possible to detect the gesture signed by both hands whilst both are in the frame. A method would have to be devised to detect a gesture (or range of gestures) that is represented by a partially occluded hand. This method would be considerably harder to implement.