International Journal of Research Studies in Science, Engineering and Technology [IJRSSET]
Volume 1, Issue 1, April 2014, PP 54-58

Gesture Controlled Human Machine Interface

Chetana Bayas

Biomedical Engineering,
Thadomal Shahani Engineering College
Mumbai, 400050
India (Final Year Student)

Manali Darji

Biomedical Engineering
Thadomal Shahani Engineering College
Mumbai, 400050
India (Final Year Student)

Rishisingh Solanki

Biomedical Engineering
Thadomal Shahani Engineering College
Mumbai, 400050
India (Final Year Student)

Abstract: *This paper presents the manoeuvre of mouse pointer and performs various mouse operations such as left click, right click, drag etc. using gestures recognition technique. A non-contact human-computer interaction system based on gesture recognition is presented using a front facing camera[1][2]. Recognizing gestures is a complex task which involves many aspects such as motion modelling, motion analysis, pattern recognition and machine learning. Keeping all the essential factors in mind software has been developed which recognizes the movement of finger through colour difference and then translates them into mouse controls. The platform used for the above application is OpenCV and Microsoft Visual Studio 2008.*

Keywords: *Computer Vision, Mouse, Thresholding, Centroid, Moments*

1. Introduction

Human–computer interaction (HCI) involves the study, planning, design and uses of the interaction between (users) and computers. Earlier computers punched holes into a piece of card to input data. It was only around the mid-20th century that a new HMI device, based on the typewriter was created, which led to the creation of the 1st computer keyboards. Computers used to be text only, no graphic user interface. It was only when computers became more graphical the need for a mouse arose. When we reached the computer boom, starting in the 1980's, a whole range of new HMI devices began to appear, including a whole assortment of joysticks, graphics tablets, single handed keyboards, a whole range of joy pads and sticks, which are still being used today. Nowadays it seems we are slowly moving away from the more traditional mouse and keyboard interface. In terms of HMI it relates to how the user will interact with a machine and how easy that interaction will be. Although we have had the mouse and keyboard for decades, the future seems to be leading us towards more natural interfaces that are more intuitive to use, and shouldn't impose any restrictions on natural movements.

5 broad categories of HMI-

Acoustic

- Optics
- Bionics
- Motion
- Tactile

Computer Vision is related to HMI, for object detection and tracking which relates to gestures control and forms the basis of the project and will be explored in the further part of paper.

1.1. Color Thresholding

When talking about colours, we have to keep in mind that colour can be realized in different ways. The colour models are based on a physical, physiological and technical aspect.

RGB (Red, Green, Blue) colour model (Fig.1) to see how a RGB model works, we first have to look at the technical aspect of colour. The human eye perceives colour reaction through a mix of red, green and blue signals. It was a thought that each colour can be mixed with only these three primary colours.

GESTURE EXTRACTION

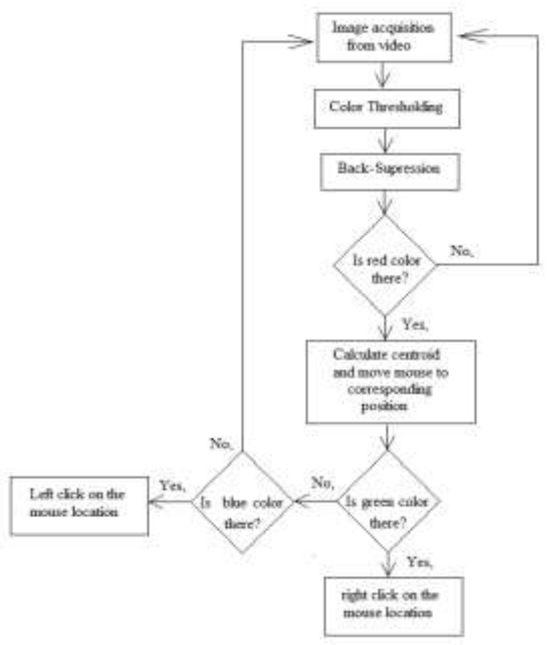
Image acquisition from video
Color Thresholding
Back-Supression
Is red color there?
No,
Yes,
Calculate centroid and move mouse to corresponding position
No,
Is green color there?
No,
Is blue color there?
Yes,
Left click on the mouse location
Yes,
right click on the mouse location

Figure1. *Block diagram*

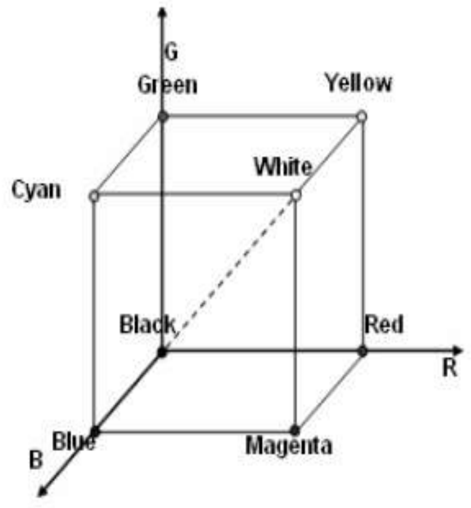

Figure2. *RGB Colour Model*

The HSV colour model, also called HSB (Hue, Saturation, Brightness), defines a colour space in terms of three constituent components:

Hue is the colour type (such as red, magenta, blue, cyan, green or yellow). Hue ranges from 0-360 deg.

Saturation refers to the intensity of specific hue. Saturation ranges are from 0 to 100%. In this work saturation is presenting in range 0-255.

Value refers to the brightness of the colour. Saturation ranges are from 0 to 100%. Value ranges are from 0-100%. In this work saturation and value are presenting in range 0-255.

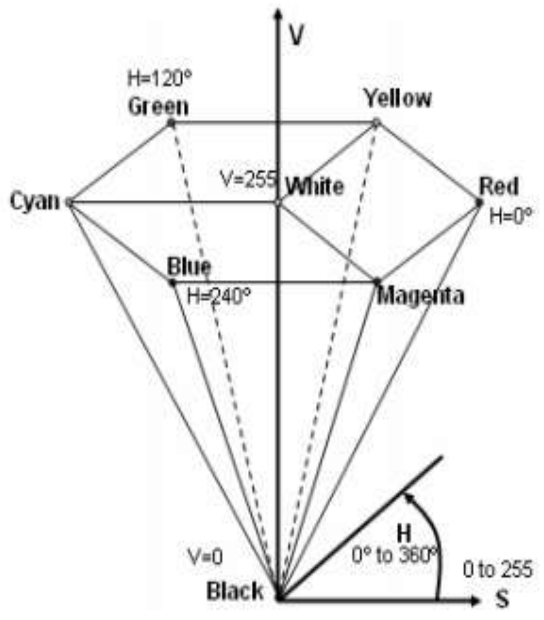

Figure3. *HSV Colour Model.[6]*

For example, the following image shows BLUE COLOR thresholding i.e. distinguishing or separating blue colour from the given image

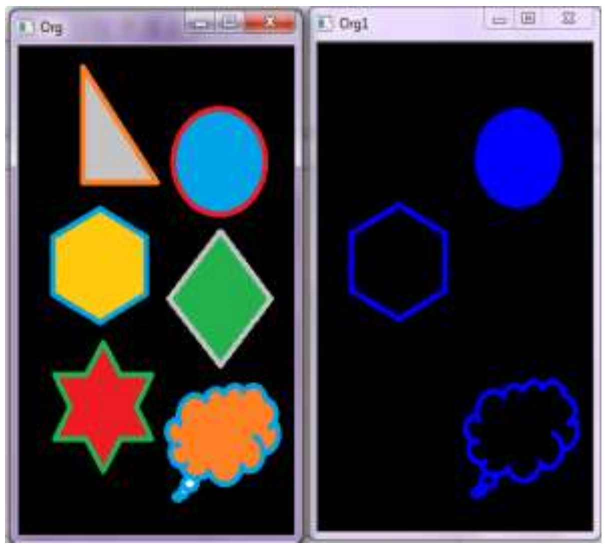

Figure4. *Blue Colour Threshold*

1.2. Calculation of Mouse coordinates using Moments

1.2.1. The math of Moment [5]

In pure math, the nth order moment about the point c is defined as:

$$\mu_n = \int_{-\infty}^{+\infty} (x - c)^n f(x)\, dx \tag{1}$$

This definition holds for a function that has just one independent variable. We're interested in images – they have two dimensions. So we need two independent variables. So the formula becomes:

$$\mu_{m,n} = \int\int (x - c_x)^m (y - c_y)^n f(x,y)\,dy\,dx \quad (2)$$

Here, the f(x, y) is the actual image and is assumed to be continuous. For our purposes, we need a discrete way (think pixels) to describe moments:

$$\mu_{m,n} = \sum_{x=0}^{\infty}\sum_{y=0}^{\infty}(x - c_x)^m (y - c_y)^n f(x,y) \quad (3)$$

The integrals have been replaced by summations. The order of the moment is m + n. Usually, we calculate the moments about (0, 0). So you can simply ignore the constants cx and cy.

Now with the math part out of the way, let's have a look at what you can calculate with this thing.

1.3. Calculating Area

To calculate the area of a binary image, you need to calculate its zeroth moment:

$$\mu_{0,0} = \sum_{x=0}^{w}\sum_{y=0}^{h} x^0 y^0 f(x,y) \quad (4)$$

The x0 and y0 don't have any effect and can be removed.

$$\mu_{0,0} = \sum_{x=0}^{w}\sum_{y=0}^{h} f(x,y) \quad (5)$$

Now, in a binary image, a pixel is either 0 or 1. So for every white pixel, a '1′ is added to the moment, effectively calculating the area of the binary image! Another thing to note is that there is only one zeroth order moment. To calculate the centroid of the binary image you need to calculate two coordinates –

$$centroid = \left(\frac{\mu_{1,0}}{\mu_{0,0}}, \frac{\mu_{0,1}}{\mu_{0,0}}\right) \quad (6)$$

How to get this, Consider the first moment:

$$sum_x = \sum\sum x f(x,y) \quad (7)$$

The two summations are like a for loop. The x coordinate of all white pixels (where f(x, y) = 1) is added up.

Similarly, we can calculate the sum of y coordinates of all white pixels:

$$sum_y = \sum\sum y f(x,y) \quad (8)$$

Now we have the sum of several pixels' x and y coordinates. To get the average, you need to divide each by the number of pixels. The number of pixels is the area of the image – the zeroth moment. So you get:

$$\mu_{0,1} = \frac{sum_y}{\mu_{0,0}} \mu_{1,0} = \frac{sum_x}{\mu_{0,0}} \quad (9)$$

One interesting thing about this technique is that it is not very sensitive to noise. The centroid might move a little bit but not much.

The following figure shows the red line traced by the movement of the box. These lines are joined by two centroids which are calculated at those two points. And thus the red line is obtained.

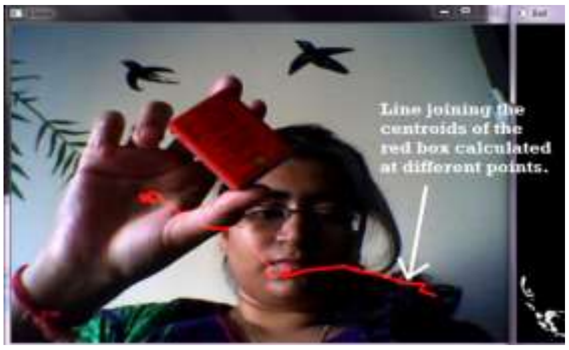
Line joining the centroids of the red box calculated at different points.

Figure5. *Tracing of Centroid.2.3 Interfacing with Microsoft Windows*

1.3.1. Basic Parts

A mouse typically has two buttons: a primary button (usually the left button) and a secondary button (usually the right button). Most mice also include a scroll wheel between the buttons to help us scroll through documents and WebPages more easily.

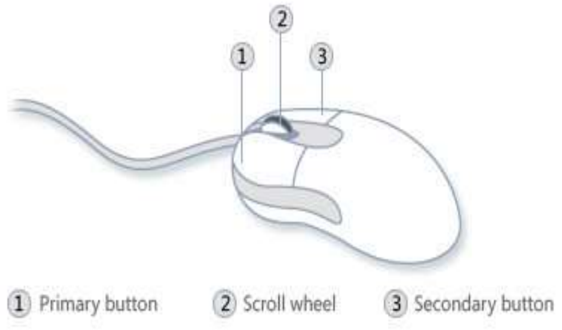

Figure6. *Tracing*

1.3.2. Using the Windows Headers

Winuser.h is part of the Microsoft Visual C++ (VC++) development environment. The tool defines several elements that developers use when they write programs to run on Windows platforms.

1.3.2.1. SetCursorPos

This function moves the cursor to the specified screen coordinates.

Syntax

```
BOOL SetCursorPos(
  int x,
  int y
);
```

X=x*cx/640

y=y×cy/480

cx and cy are screen resolution.

Specifies the new x-coordinate and y-coordinate, in screen coordinates, of the cursor. The centroid coordinates x and y (2.2.3 - Equation (6)) acts as the mouse coordinates.

1.3.2.2. mouse_event function [4]

The mouse_event function synthesizes mouse motion and button clicks.

Syntax

```
VOID WINAPI mouse_event(
  _In_  DWORD dwFlags,
  _In_  DWORD dx,
  _In_  DWORD dy,
  _In_  DWORD dwData,
  _In_  ULONG_PTR dwExtraInfo
);
```

dwFlags [in]

Type: **DWORD**

Controls various aspects of mouse motion and button clicking. This parameter can be certain combinations of the following values.

MOUSEEVENTF_LEFTDOWN - left button is down.

MOUSEEVENTF_LEFTUP - left button is up. **MOUSEEVENTF_MIDDLEDOWN** -middle button is down.

MOUSEEVENTF_MIDDLEUP - middle button is up.

MOUSEEVENTF_RIGHTDOWN - right button is down.

MOUSEEVENTF_RIGHTUP – right button is up.

Table1. *Color and their function*

Color	**Function**	**Role**
Red	Cursor Pointer	Tracking
Green	Click	Single Right Click
Blue	Click	Single Left Click

Eg. ***Mouseeventf_Leftdown*** is set when the left button is first pressed, but not for subsequent motions. Similarly, *Mouseeventf**_Leftup*** is set only when the button is first released.

Thus passing the respective parameters in mouse event function in accordance to the color detected we can use all the basic mouse features.

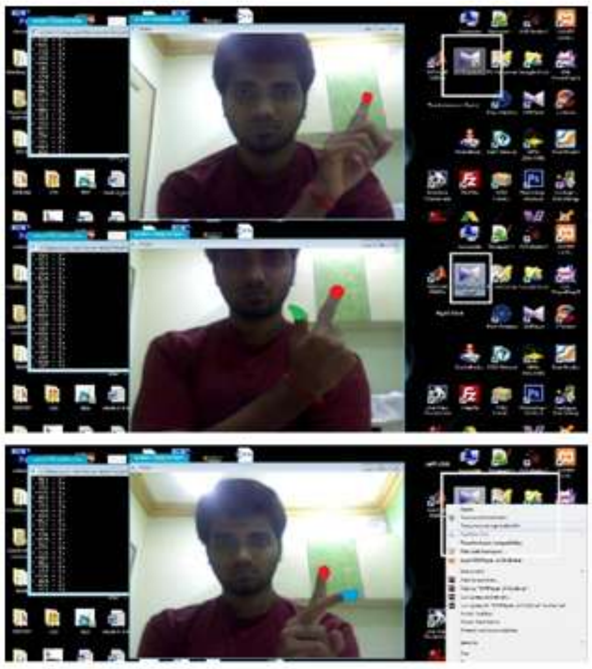

Figure7. *Different color markers and their functions.*

2. Conclusion

We developed a system to control the mouse cursor using a real-time camera. We implemented all mouse tasks such as left and right clicking, double clicking, and scrolling. The system has been implemented in Open CV environment using Microsoft Visual Studio. This system is based on computer vision algorithms and can do all mouse tasks. However, it is difficult to get stable results because of the variety of lighting and skin colours of human races. Most vision algorithms have illumination issues. From the results, we can expect that if the vision algorithms can work in all environments then our system will work more efficiently. This system could be useful in presentations and to reduce work space. In the future, we plan to add more features such as enlarging and shrinking windows, closing window, etc. by using the palm and multiple fingers.

ACKNOWLEDGEMENT

Special Thanks to our project guide and mentor Mr. Dhananjay Theckedath..

REFERENCES

[1] Hojoon Park, “A Method For Controlling The Mouse Movement using a Real Time Camera”, 2008, Brown University, Providence ,RI ,USA, Department of computer science

[2] Jun-ho An; Kwang-Seok Hong; ”Finger gesture-based mobile user interface using a rear-facing camera”, 2011, pp 303-304

[3] http://msdn.microsoft.com/enus/library/aa929734.aspx

[4] http://www.exelisvis.com/docs/MouseEventFunctions.html

[5] http://zoi.utia.cas.cz/files/chapter_moments_color1.pdf

[6] http://coecsl.ece.illinois.edu/ge423/spring05/group8/FinalProject/HSV_writeup.pdf