International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 3, March - 2014

A Novel Approach Towards Optimization Of Vision Based Virtual Mouse Interface

Sandeep Konam[1], Sai Maheswara Reddy[2]
Rajiv Gandhi University of Knowledge Technologies, R.K.Valley, India

***Abstract*—Virtual Interface offers a natural and intelligent mode of interaction with Machines. Hand gesture recognition is more efficient and easier than traditional devices like keyboards and mouses.In this paper, a vision based virtual mouse interfacethat can recognize the pre-defined mouse movements in real time regardless of the context is proposed.Computer vision technology, such as image segmentation and gesture recognition coupled with an intuitive algorithm can control mouse tasks (left and right clicking, double-clicking, holding and scrolling). A fixed set of manual commands anda reasonably structured environment are considered to develop a simple, yet effective procedure for interfacingthat decreases the complexity and boosts the efficiency of present state Human Computer Interaction.The proposed algorithm is invariant to translation, rotation, and scale of the hand. The paper takes off with a brief introduction and few insights into the state of art. Then the proposed methodology is unveiled. The effectiveness of the technique is demonstrated on real imagery and the results are furnished. Finally, opportunities and challenges with a brief note on future research possibilities are highlighted.**

***Keywords*—Centroid, Coloured Pointers, Detection, Mouse Click events, Virtual Interface.**

I. INTRODUCTION

Human computer interaction (HCI) is one of the most important areas of research where researchersare in a continuous quest towards improved interfaces. Of late, smaller and smaller devices are being requested by the users. Most of the present day interfaces need some amount of space so as to use and cannot be used while moving. Though Touch screens have been globally accepted for their efficiency in control applications, they cannot be applied to desktop systems because of cost and other hardware limitations. By applying vision technology, coloured pointers and controlling the mouse by natural hand gestures, the work space required can be reduced. Optimizing Virtual Mouse interface is a challenging problem in its general form. Though the challenge seems to be an age old one, to this day there hasn't been a single version qualitative enough to meet the desire of user.The proposed technique could be attributed as such an attempt to improve the interface in terms of speed, efficiency and reliability.

II. STATE OF THE ART

Most existing approaches involve changing mouse parts such as adding more buttons or changing the position of the tracking ball.

Many researchers in the human computer interaction and robotics fields have tried to control mouse movement using video devices. However, all of them used different methods to make a clicking event. One approach, by Erdem et al, used fingertip tracking to control the motion of the mouse. A click of the mouse button was implemented by defining a screen such that a click occurred when a user's hand passed over the region [1]. Another approach was developed by Chu-Feng Lien [2]. He used only the finger-tips to control the mouse cursor and click. His clicking method was based on image density, and required the user to hold the mouse cursor on the desired spot for a short period of time. Paul et al, used still another method to click. They used the motion of the thumb (from a 'thumbs-up' position to a fist) to mark a clicking event thumb. Movement of the hand while making a special hand sign moved the mouse pointer [3].

To detect the object, salient future detection algorithm such as SIFT, SURF, or ORB can be used. However, this type of complex algorithm might slow down the speed and the system may not be used in real time. Speed and accuracy trade-off needs to be balanced.

III. PROPOSED METHODOLOGY

A coloured pointer has been used for the object recognition and tracking. Left and right click events of the mouse have been achieved by detecting the position of pointers with respective to each other on the image. In the object tracking application one of the main problems is object detection. Based on precise thresholding with defined colour range in the HSV colour space, object can be segmented from the image and can be used for further execution of the mouse click events. Instead of finger tips, a colour pointer has been used to make the object detection easy and fast. To simulate the click events of the mouse the fingers with these colour pointers have been used.

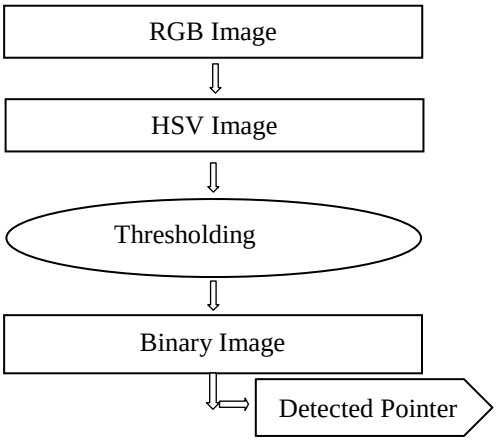
A. The basic algorithm

- Set coloured pointers.
- Detect the pointer using the defined colour information.
- Define the region and the centre of the pointer.
- Track the motion of the pointer.
- Execute the event according to the position of the centre and relative distance between pointers.

IV. PRE-REQUISITES

A. Detection of pointer

Three tracking methods are widely employed for detection of objects from the input image namely Feature based methods, Differential methods and Correlation based methods. Feature based methods extract characteristics such as points, line segments from image sequences and have been proved reliable for detecting and tracking the objects. In the feature based approach, four features namely centroid, dispersion, grey scale distribution, object texture can be used for detecting pointers. Present technique is built upon centroid based detection.



Input image, initially in RGB colour space is converted to HSV colour space for ease and efficiency in identification of colours. Thresholding is then applied on converted image with predefined range of required colour to be segmented. It results in a binary image with concentration of white pixels at the detected pointer.

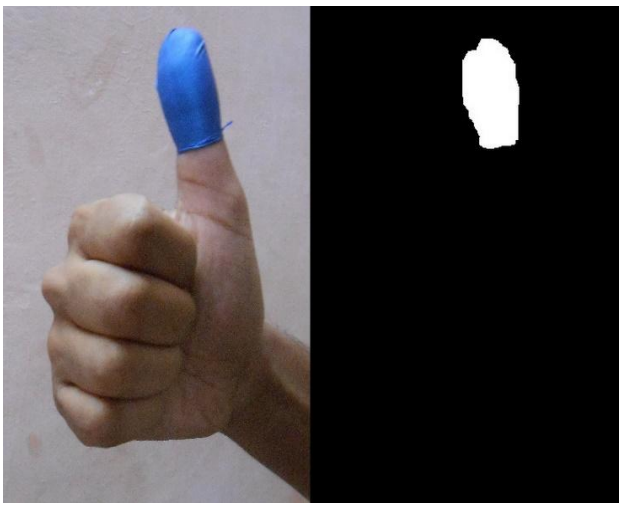


Fig. 1 Detected Pointer from RGB image

B. Detection of Centroid of pointer

After removing all the connected components (objects) other than the pointer using the pointer can be labelled. In other words the region can be detected. Moments can be used to calculate the position of the centre of the object. We have to calculate 1st order axis and the 0^{th} order central moments of the binary image.

0^{th} order central moments of the binary images are equal to the white area of the image in pixels.

- X-coordinate of the position of the centre of the object = 1^{st} order spatial moment around x-axis / 0^{th} order central moment.
- Y-coordinate of the position of the centre of the object = 1^{st} order spatial moment around y-axis / 0^{th} order central moment.

After segmenting the coloured pointer from the image, it is approximated as an ellipse and the centre of the pointer can be calculated for further processing with the following equation.

$$\bar{x} = \frac{\sum_{i=0}^{k} x_i}{k} \quad , \quad \bar{y} = \frac{\sum_{i=0}^{k} y_i}{k}$$

Where x_i and y_i are x and y coordinates of i pixel in the hand region, and k denotes the number of pixels in the region.

Fig. 1Detected Centroid of the pointer

C. Distance between the pointers

Distance between pointers can be calculated from the mathematical formula mentioned below with the two coordinates as the centroids of the detected pointers.

$$\text{Distance D} = \sqrt[2]{(RedX - BlueX)^2 + (RedY - BlueY)^2}$$

Here, RedX and RedY refer to the x and y-coordinates of red colored pointer where as BlueX and BlueY refer to the x and y coordinates of blue coloured pointer. Based on this distance between pointers, click events and scrolling are primarily differentiated and executed. Distance between the pointers coupled with angle can form the basis for distinction between left, right and double click events.

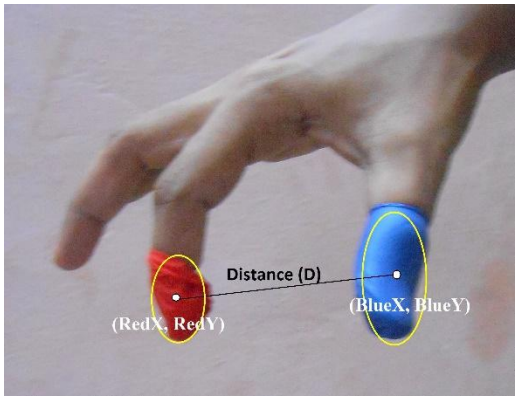


Fig. 3. Measured Distance between the Pointers

In order to know the presence of contact established between two pointers, Distance is essentially calculated.

D. Angle between pointers

Angle between pointers plays a crucial in deciding the type of event to be executed.

$$\theta = \tan^{-1}\left(\frac{\text{RedY} - \text{BlueY}}{\text{RedX} - \text{BlueX}}\right)$$

If $\theta < 0$, then left click performed
$\theta = 0$, then hold operation is performed
$\theta > 0$, then right click performed

Depending on orientation of one pointer with respect to another, mouse click events are performed.

V. MOUSE CONTROLS

Thumb finger is used as a cursor controller to control mouse cursor. It can attributed as a mapping cursor control. It means that the blue coloured pointer supposed to be the thumb finger position maps to a desktop screen position. The proposed solution has very little time delay compared to the weighted speed cursor control. This method's efficiency is proportional to the resolution of camera. In spite, the proposed technique works efficiently on low resolution images.

Depending on the pointer detected in the sequence of image frames, the coordinates of pointer mapped onto screen resolution change and the cursor moves accordingly

A. *Mouse Click events*

Mouse click is one of the most challenging parts for the virtual mouse application. As all we know, the left button of the mouse has a different task for a single or double clicks. To simulate this feature another pointer has been used. When the second pointer is detected in the image, left mouse click event has been executed. Depending on the time that the second pointer is being detected single and the double clicks have been simulated.

1) Left click:

In order to perform left click, red colored pointer needs to be placed on the thumb finger-tip which is already associated with blue band. Angle between the two centroids should be less than 0 radians, which can calculated from above mentioned formula

Fig. 4. Defined gesture for Left Click

Distance between two centroids should be less than sum of Radii of red, blue pointers and additional margin 20. It should be less than the mentioned sum, so as to confirm the existence of contact between the two pointers. 20 is considered as the noise margin, which is directly influenced by illumination and is to be considered only for low resolution imagery.

(D< (RedRadius+BlueRadius+20))&& (RedY>BlueY) &&θ<0

2) Double Click:

Double click can be approximated as the two consecutive left clicks performed with a time delay greater than $1/10^{th}$of a second. The implementation can be corresponded to the above mentioned left click event. The conditions remain the same for both.

3) Right click:

In order to perform right click, red coloured pointer needs to be placed at the bottom end of blue pointer. Angle between the two centroids should be greater than 0 radians, which can calculated from above. As in the case with left click, Distance between two centroids should be less than sum of Radii of red, blue pointers and additional margin 20.

(D<(RedRadius+BlueRadius+20))&& (RedX<BlueX) && θ>0

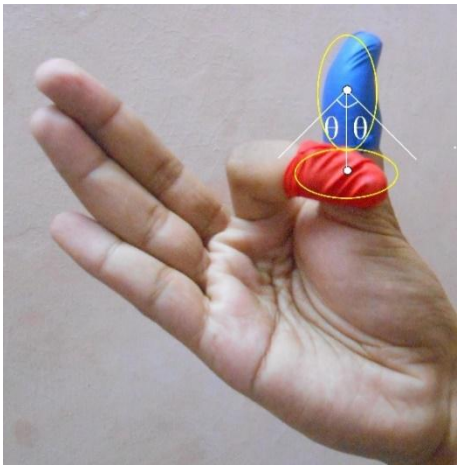


Fig. 5. Defined gesture for Right Click

4) Mouse holding

In order to perform mouse holding, red coloured pointer needs to be placed beside the blue pointer such that they are on the same horizontal axis. Angle between the two centroids should be equal to 0 radians, which can calculated from above. As in the case with the above mentioned clicks, Distance between two centroids should be less than sum of Radii of red, blue pointers and additional margin 20.

(D<RedRadius+BlueRadius+20)&& θ==0

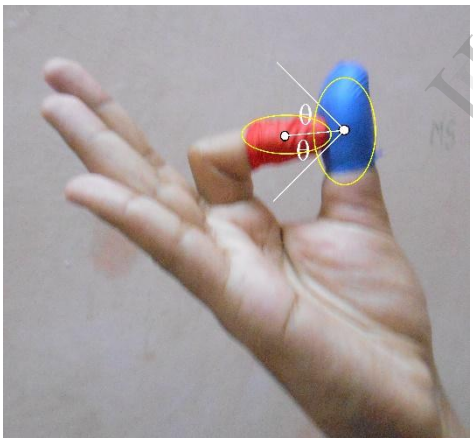


Fig. 6. Defined gesture for Mouse Hold

B. *Scrolling:*

Scrolling is a very useful task with hand gestures. We also implemented this feature in our system. Based on the distance between two centroids, the pointers are ensured that they are not in contact. Depending on the position of pointers with respect to each other, ie. if the blue colored pointer is above the red coloured pointer, scrolling down is executed and if red colored pointer is above the blue colored pointer, scrolling up is executed.

1) Scrolling up:

In order to perform up scrolling, y coordinate of red pointer needs to be greater than y- coordinate of blue pointer. On both sides of Red-X, a margin of 20 pixels is considered for better user interface. The event will be not be executed in case the centroid of one pointer falls out of range ie. 20 pixels. This condition has been considered so as not to mess with the scrolling down condition. The margin of 20 pixels can be increased or decreased based up on user convenience.

Red Y > Blue Y

Blue X – 20 < Red X < Blue X + 20

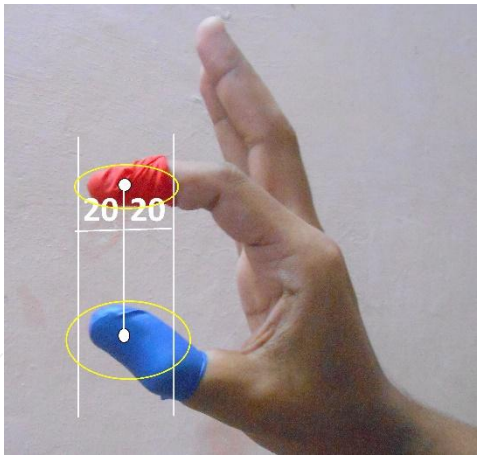


Fig. 7. Defined gesture for Scrolling Up

2) Scrolling Down:

In order to perform down scrolling, y coordinate of red pointer needs to be less than y- coordinate of blue pointer. On both sides of Red-X, a margin of 20 pixels is considered for better user interface. The theory developed for scrolling up applies to scrolling down too.

Red Y < Blue Y

Blue X – 20 < Red X < Blue X + 20

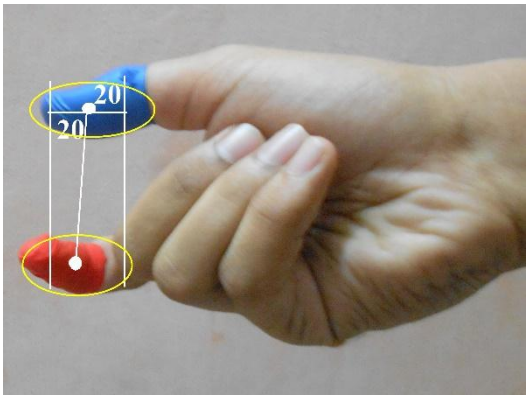


Fig. 8. Defined gesture for Scrolling down

VI. RESULTS

Method followed for evaluating performance is that time for a task, such as clicking, double clicking and scrolling, is noted. Five experiments have been designed to get performance. In the first experiment we placed an icon on the centre of desktop window and put the cursor in the top-left corner. We then measured the time in how long it takes to select the icon. In the second experiment the position of the icon is same and we only measured the time to show the drop down menu on the icon. In the third experiment the position of the icons is same with the second experiment and we measured the time to open the icon. In the fourth experiment, an icon is dragged from one corner to another corner of the screen and the time was measured. In the final experiment we opened The Hindu home page (http://www.thehindu.com) and measured the time until the scroll bar moves from top to bottom. All the experiments have been performed with people new to the interface. Except left and double click events, all the events match to the capabilities of traditional mouse.The results are shown below: (time = sec).

TABLE I
LEFT CLICK: SELECTING THE ICON

	Trial 1	Trial 2	Trial 3	Average	Traditional Mouse
User 1	2.62	2.58	2.68	2.627	1.832
User 2	2.41	2.54	2.48	2.48	1.742
User 3	2.84	2.68	2.62	2.71	2.12
User 4	2.82	2.74	2.76	2.78	1.964

TABLE II
RIGHT CLICK: DROP DOWN MENU

	Trial 1	Trial 2	Trial 3	Average	Traditional Mouse
User 1	0.68	0.64	0.72	0.68	0.54
User 2	0.54	0.62	0.68	0.61	0.48
User 3	0.71	0.67	0.70	0.69	0.62
User 4	0.82	0.78	0.73	0.78	0.59

TABLE III
DOUBLE CLICK: OPENING THE ICON

	Trial 1	Trial 2	Trial 3	Average	Traditional Mouse
User 1	1.48	1.52	1.46	1.49	0.81
User 2	1.39	1.36	1.41	1.39	0.79
User 3	1.52	1.38	1.48	1.46	0.72
User 4	1.56	1.48	1.53	1.52	0.86

TABLE IV
DRAGGING ICON FROM ONE CORNER TO ANOTHER CORNER

	Trial 1	Trial 2	Trial 3	Average	Traditional Mouse
User 1	1.72	1.70	1.73	1.72	1.68
User 2	1.86	1.91	1.95	1.91	1.83
User 3	1.89	1.87	1.79	1.85	1.74
User 4	1.73	1.71	1.74	1.73	1.69

TABLE V
SCROLLING THE HINDU HOME PAGE

	Trial 1	Trial 2	Trial 3	Average	Traditional Mouse
User 1	1.62	1.58	1.56	1.59	1.44
User 2	1.76	1.68	1.72	1.72	1.51
User 3	1.98	1.86	1.92	1.91	1.72
User 4	1.79	1.97	1.86	1.87	1.88

VII. CONCLUSIONS

A vision based virtual mouse interface to control the mouse cursor using a real-time camera was developed that could perform all mouse tasks such as left and right clicking, double clicking and scrolling. Most vision algorithms have illumination issues and the proposed method is no exemption to it. We can expect that if the vision algorithmscan 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, our focus would be extended to develop a virtual mouse interface with fingers, thereby removing the coloured pointers and promoting more interaction with the user. Also more features such as enlarging and shrinking windows, closing window, etc can be added there by making it an integrated version of touch screen and traditional mouse.

However, system has some disadvantages such as: being variant to illumination up to some scale, and movement of the cursor is very sensitive to motion. In the near future, a robust virtual mouse interface overcoming the above said challenges can be developed with little modifications to the existing system.

REFERENCES

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

[2] Chu-Feng Lien, “Portable Vision-Based HCI – A Real-time Hand Mouse System on Handheld Devices”, National Taiwan University, Computer Science and Information Engineering Department.

[3] Hojoon Park, “A Method for Controlling the Mouse Movement using a Real Time Camera”, 2008, Brown University,Providence, RI, USA Department of computer science.

[4] Dix, A. Finlay, J. Abowd, G. D. and Beale, R. Human-Computer Interaction, 3rd ed.Pearson Education Limited, 2004

[5] Colombo, C. Bimbo, A. D. and Valli, A. "Visual Capture and Understanding of Hand Pointing Actions in a 3-D Environment," IEEE Transactions on Systems, Man, and Cybernetics, 33(4), University of Florence,Italy, pp.677-686,August2003.