ISSN 2047-3338

Virtual Touch – Human Computer Interaction at a Distance

Prasanna Dhisale, Puja Firodiya, Priyanka Gawade and Manjiri Mahamuni
Pimpri Chinchwad College of Engineering, Nigdi-44, India
prasannad91@gmail.com, pujafirodiya9@gmail.com, priya.gawade91@gmail.com, manjiri992@gmail.com

***Abstract*– In computing, a Natural User Interface, or NUI, is the common term used by designers and developers of computer interfaces to refer to a user interface that is effectively invisible, or becomes invisible with successive learned interactions, to its users. The word 'Natural' is used because most computer interfaces use artificial control devices whose operations have to be learned. A NUI helps a user to quickly transit from novice to expert. On the same way, we present Virtual Touch, a new vision-based interaction system. Virtual Touch uses computer vision techniques to extend commonly used interaction metaphors, such as multi touch screens, yet removes any need to physically touch the display. The user interacts with a virtual plane that rests in between the user and the display. On this plane, hands and fingers are tracked and sequences of gestures are recognized in a manner similar to a multi touch surface. Many of the other vision and gesture-based human-computer interaction systems presented in the literature have been limited by requirements that users do not leave the frame or do not perform gestures accidentally, as well as by cost or specialized equipment. Virtual Touch does not suffer from these drawbacks. Instead, it is robust, easy to use, builds on a familiar interaction paradigm, and can be implemented using a single camera with off the-shelf equipment such as a webcam enabled laptop. In order to maintain usability and accessibility while minimizing cost, we present a set of basic Virtual Touch guidelines. We have developed two interfaces using these guidelines—one for general computer interaction, and one for searching an image database. We present the workings of this system for power point presentation, Mouse control, 3D Interaction, Gaming, Image viewer and media player control.**

***Index Terms*– Virtual, Blobs, Multi-Touch, HCI and NUI**

I. INTRODUCTION

SINCE the inception of graphical computer systems in April 1981, interaction with graphical computer systems has evolved over the past thirty years to include such interface metaphors as the mouse and keyboard, pen computing, touch, and recently multi touch. These enabling developments have allowed interaction to become more accessible and natural. A recent and notable example is the Apple iPod, which sold more than 3 million units within its first three months. Despite the broad range of technologies in these developments, they all rely on one basic principle: interaction happens on two-dimensional plane. As an *enabling technology, computer vision has great potential to further improve the naturalness of these interaction* metaphors and also to take interaction off of the two-dimensional plane constraint. Indeed, multiple attempts have been made over the past decade to create methods for computer interaction using computer vision, though none of these systems have achieved widespread use.

Computer vision systems that allow for human interaction generally use some sort of interface defined by the gestures the user can perform. These gestures must happen within the field of view and range of some camera device. Many systems utilizing a gesture based interface allow for the gestures to happen at many distances before the camera with largely the same effect. Perhaps the simplest paradigm attempted by some of these systems involves extending the concept of Marking Menus (also known as "pie menus") to computer vision, which only requires a user's hand to be present in a specific region of the cameras view to perform an action. Extensions of these ideas led to two mature vision interaction systems based on localized interaction within certain "hot spots" in the environment. However, the localized interaction does not allow generalized control. Many other systems attempt to use gestures for general computer use.

II. RELATED WORK

There are various techniques have been proposed in the literature to deal with the difficulties in computer vision to control the devices from a distance. Vision based hand gesture recognition is believed to be an effective technique. The system tracks the motion of hand.

The problem of this approach is that it is slow and not accurate on complex background. Moreover the user needs to hold the hand pose still and frontally facing the camera for several seconds to activate an operation. As described in [1] camshift algorithm was used to track the user's hands, but camshift algorithm's processing is not as fast as *BLOB DETECTION* algorithm.

To reliably track the user's hands in a computationally inexpensive way, we use the *BLOB DETECTION* algorithm and require the user to wear a colored band. The use of band could be relaxed via skin-color tracking.

Virtual Touch uses colored marker attached to fingertips and Microsoft Kinect uses depth image sensor to segment people's body, both of their approaches make the hand detection easier. In short, most of the presented interaction techniques of hand gestures interaction have limitations for the purpose of taking self-portraits by using a digital camera. Our proposed interaction technique is application oriented designed especially for self-portrait; a higher speed hand detecting algorithm and a cross motion recognition interface are developed. By using this light-weighted algorithm, it is easier to transfer the algorithm into the camera device.

Our proposal mainly has three contributions. First, we propose *a novel technique that enables user to manipulate digital camera conveniently using hand gesture especially when controlling it from a distant*. Second, we developed a *real-time computer vision algorithm that tracks the hand and fingertip with accuracy and high speed.* Third, a *cross motion interface* used to recognize hand motion direction has been proposed.

III. PROPOSED SYSTEM

Virtual touch implement a motion based control system for advanced Human Computer Interface (HCI). It enables natural human-computer interaction at a distance without requiring the user to adapt his or her behavior in any significant way, or spend time calibrating the system. In this system we use some sort of interface defined by the gestures. These gestures must happen within the field of view and range of camera device. The frontend web-cam will be capturing the motion up to 3-4 feet apart from the Laptop .To interact with the system the user has to wear a band on his/her finger.

A. Computer Vision for General HCI

The *Computer Vision for General HCI (CVHCI)* system allows the user to interact with the computer system from a distance. In this context the Virtual Touch system is used to mimic a touch screen floating in space. The user interacts with the system through various motions. Processing the users interactions happens on two distinct levels— *tracking and gesturing*. Which, when combined, allow for a highly usable system for human-computer interaction.

1) Tracking

To reliably track the user's hands in a computationally inexpensive way, we use the *BLOB DETECTION* algorithm and require the user to wear a colored band. The use of band could be relaxed via skin-color tracking, though we use them primarily because of the low computational overhead of the involved algorithms. Colored band is used with much success in other real-time applications such as MIT's hand tracking system for virtual manipulation of objects.

The *BLOB DETECTION* algorithm allows us to determine not only the X and Y coordinates of each tracked finger, but also the diameter of the tracked area, which we treat as a proxy to the Z-axis using only one camera. The diameter of the finger is noted at the detection point, defining Dtouch, the "depth" location of the Virtual Touch plane. To allow the user to act normally in front of the camera, we use adaptive *color histograms* to provide the finger tracking with robustness to lighting changes and quick movement. Periodically, the histogram of the tracked region is reevaluated and compared with the histogram on which the tracker's back projection is currently based; if there is a large enough discrepancy, the histogram is updated to reflect the color of the tracked object in the current environment.

The histograms we use are in the *HSV* color space since lighting and camera white balance changes cause variation largely in the saturation and value components with less effect on hue. For this reason, we only use the hue histogram in tracking and we threshold the saturation and value channels to try to maximize the presence of the tracked hue in the back projection. When the histogram is updated, the saturation and value threshold values are updated automatically using a binary search of the respective spaces to find the best back projection, the one containing the most positive pixels in the tracked area. This update assists with not only with lighting changes but also with quick movement by ensuring the track is not lost if the hand blurs with fast motion causing a slightly different histogram.

2) Gesturing

The system currently allows for three gestures for interaction with the computer: *mouse movement, mouse clicking, and scrolling*. All three gestures are variations on pointing at the screen. To move the mouse the user moves their pointer finger in front of the camera at a distance outside of the Virtual Touch plane. This can be likened to how a Tablet PC handles mouse movement, change the location of the mouse pointer. Mouse clicking is actually two gestures—moving the pointer finger into the plane to activate mouse down, and removing it to mimic mouse up. In other words, we have a virtual touch screen that allows for click-and-drag type actions. Finally, the scrolling gesture is accomplished by moving both the pointer and middle fingers into the interaction plane and moving them vertically, which is common on multi touch track pads. We have also implemented an *API for applications to plug-in to Virtual touch* and add their own gestures.

B. Processing steps in Virtual Touch

i. Webcam Interfacing
ii. Grabbing Image
iii. Background removal algorithm
iv. Color detection
v. Blurring an image
vi. RGB to HSV Conversion
vii. Color Thresholding

viii. Blob Detection
ix. Operating system Interface
x. Application program Interface(API)

C. System Flow Diagram

Figure 1 shows the flow diagram of the proposed system.

IV. SYSTEM ARCHITECTURAL DIAGRAM

Figure 2 shows the architectural diagram of our proposed system.

A. Blur Filter

Sr. No.	Steps
1.	After getting input from webcam, traverse through entire image array.
2.	Read individual pixel color value.

3.	Split color value into individual R, G, B i.e. 8 bit values.
	
	b = col & 0xff;
	g = (col >> 8) & 0xff;
	r = (col >> 16) & 0xff;

4.	Calculate sum of R,G,B separately, average of surrounding pixels & assign that avg value to it.
	sumR += r;
	sumG += g;
	sumB += b;
	// average of 24 surrounding pixels and center
	r = sumR / 25;
	g = sumG / 25;
	b = sumB / 25;

5.	Repeat above steps for each pixel.
6.	Store new value at same location in output image.

B. RGB to HSV Conversion

Sr. No.	Steps

1.	Find min and max values of RGB.
	rgbMin = Math.min(Math.min(r,g),b);
	rgbMax = Math.max(Math.max(r,g), b);

2.	Compute and normalize values(V) to 1. V = rgbMax; if(V==0) then h = s = 0;

3.	Compute and normalize saturation(S) to 1.
	If V is not 0 then
	s = 255 * (rgbMax-rgbMin)/V;
	if(s==0) then h=0;

4.	Compute hue(H).
	If s is not equal to 0 then,
	if(rgbMax == r) then
	h = 0 + 43*(g-b)/(rgbMax-rgbMin);
	if(rgbMax == g) then
	h = 85 + 43*(b-r)/(rgbMax
	-rgbMin);
	if(rgbMax == b) then
	h = 171 + 43*(r-g)/(rgbMax -rgbMin);
	if(h<0)
	then h = 255+h;

C. Color Thesholding

Sr. No.	Steps
1.	Traverse through entire input image array

2.	Read individual pixel color value (24-bit) and convert it into grayscale.
	b = col & 0xff;
	g = (col >> 8) & 0xff;
	r = (col >> 16) & 0xff;
	gs = (r+g+b) / 3; //gs=Grayscale

3.	Calculate the binary output pixel value (black or white) based on current threshold.
	if(gs < th) {
	pix = 0;
	// pure black color
	}
	else
	{
	pix = 0xFFFFFF;
	// pure white color }

4.	Store the new value at same location in output image

D. Blob Detection

Sr. No.	Steps
1.	Check the first line of the image and find groups of one or more white pixels. These are the blobs on a certain line, called lineblobs.
2.	Number each of these groups.
3.	Repeat this sequence on the next line.
4.	While collecting the lineblobs, check the lineblobs on the line we have checked before this current line and see if these blobs overlap each other.
5.	If we merge these lineblobs as one blob i.e. give the current lineblob the same number or id as the lineblob(s) on the other line. Repeat this for every line and have a collection of blobs.

V. ADVANTAGES OF PROPOSED SYSTEM

i. System is robust.
ii. Easy to use.
iii. It is built on a familiar interaction paradigm.
iv. It can be implemented using a single camera with off-the-shelf equipment such as a frontend webcam-enabled laptop.
v. In combination with gesture recognition, the interaction between human and machine can be greatly simplified.

VI. APPLICATIONS

1. *PowerPoint presentation-*

   Using Virtual Touch we can move slides up-down or shift towards right or left while doing presentation.

2. *Media Player Control-*

   We can control volume, Play, Pause or select other songs using this system.

3. *3D Interaction-*

   We can rotate 3 Dimensional image to see top view side view bottom view etc.

4. *Gaming-*

   We can play games from a distance without touching mouse.

5. *Image viewer-*

   To see next and previous images.

VII. CONCLUSION

We believe that this interface metaphor is one which has nearly limitless applications in offices, labs, households, industry, and on the move. We hope this system will be adopted by others and used to promote efficient and natural methods of human-computer interaction. The issue of natural and comfortable interaction between humans and computers has received much study in recent years. On several occasions vision systems have been proposed in an attempt to create a natural method for interacting with machines while not directly touching them. These systems have primarily been restricted to lab settings, likely due to robustness problems, difficulty of set-up, and cost issues. In contrast, we have focused our efforts on a vision based interaction system that uses standard hardware and extends already well-known interaction metaphors.

REFERENCES

[1]. Daniel R. Schlegel, Albert Y. C. Chen, Caiming Xiong, Jeffrey A. Delmerico, Jason J. Corso, "AirTouch: Interacting With Computer Systems at a Distance", 2010 IEEE.

[2]. L. Bretzner, S. Lenman, and B. Eiderbck. "Computer vision based recognition of hand gestures for human-computer interaction", Technical report, University of Stockholm, 2002.

[3]. R. Wang and J. Popovi, "Real-time hand-tracking with a color glove", ACM Transactions on Graphics, 2009.

[4]. T. Lindeberg (2008/2009). *Scale-Space*, "Scale-space". *Encyclopedia of Computer Science and Engineering (Benjamin Wah, ed), John Wiley and Sons*, IV: 2495–2504. doi:10.1002/9780470050118.ecse609. ISBN 0-470-05011-X.

[5]. K. Mikolajczyk, K. and C. Schmid (2004), "Scale and affine invariant interest point detectors", *International Journal of Computer Vision* 60 (1): pp: 63–86.doi:10.1023/B:VISI.0000027790.02288.f2.

[6]. T. Lindeberg (1998), "Feature detection with automatic scale selection", (abstract page). *International Journal of Computer Vision* 30 (2): pp: 77–116.

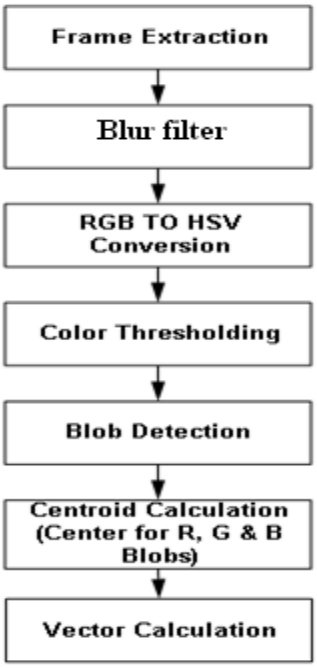

Fig. 1: Proposed System Flow Diagram

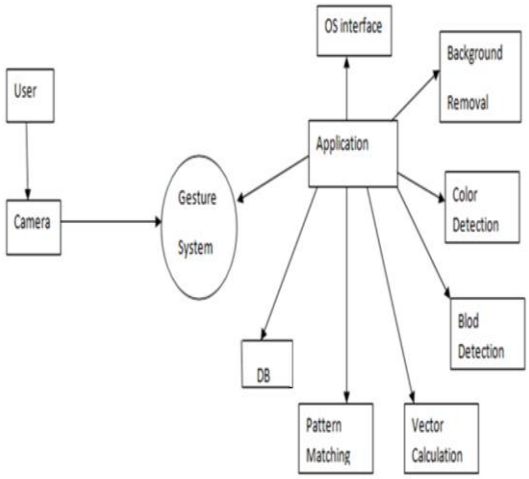

Fig. 2: System Architecture