Gesture Oriented Multi-Disciplinary Gadgets

Neeraj Ajit Abhyankar
Department Of Electronics and Telecommunications
Mumbai, India

Yash Chaudhari
Department Of Electronics and Telecommunications
Mumbai, India

Pratik Haware
Department Of Electronics and Telecommunications
Mumbai, India

Jay Choksi
Department Of Electronics and Telecommunications
Mumbai, India

Prof. Jyoti Dange
Professor, Department Of Electronics and Telecommunications
Mumbai, India

***Abstract*— The need for simplicity in the interface between the human world and the digital world is ever increasing. The ever maturing technology is at the edge of realizing the desired connect between the two, the gesture based interface is very promising for such realization. To control the multi-disciplinary gadgets from the digital world using the gestures and human interactions is very possible. Certainly we can control various devices using the gestures and make the digital world dance on our fingertips. For this purpose a real-time connection between the two worlds will be provided by a real-time camera for gesture recognition and process execution.**

Keywords— gesture; recognition; MATLAB; gadgets; dgital world; hand gesture recognition.

I. INTRODUCTION

The human computer interface technology continues to mature with a increasing pace and in a nick of time, we now like to have smaller and state of the art electronic devices to use with ease and efficiency. Ever growing need for the new technological interfaces is very essential for the interaction between the small electronic gadgets and new features with it. HCI – Human Computing Interaction plays an important role interfacing the digital and the human worlds together using the gesture based object recognition technology. There exist more primitive interfaces but the problem associated is that they cannot be used in motion i.e. they are stationary interfaces. The Touch screens have become very popular and are used worldwide. However, this technology is just not that viable due to its hardware limitations and lacking the cost effectiveness. Relatively using the gesture based object recognition environment we can use the digital devices using simple gestures as we use in daily life to indicate them. In this paper, we put forward a novel approach which uses gesture based recognition system to control multi-disciplinary gadgets. This system can control various gadgets making them very tranquil to use. They involve drawing patterns using gestures, controlling the media devices, making a hand-on call to predefined numbers in the directory or an SOS message, clicking real-time photos etc.

I. RELATED WORKS

There is a vast and wide ranged technological re-search in the field of HCI and the Robotics field for the Human Robot/Digital-devices interfacing. Although, each one of them uses a different approach to define a gesture or an event occurring. An approach by Erdem et al, used finger tip 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, 3]. One other developed by Chu-Feng Lien [4]. It used only the fingertips to control the cursor and click event. The method was developed on image density; it also required the user to hold the mouse cursor on the desired spot for a short period of time. Paul et al, used another method using motion of the thumb (Forming 'thumbs-up' position to all others) to mark an event. A special movement of hand defined the movement of the mouse pointer.

Our project was inspired by a paper of Asanterabi Malima et al. [8]. The team developed a finger counting system to control behaviors of a robot. The other one is based on the works of Soshi Iba, J. Michael Vande Weghe, Christiaan J. J. Paredis, and Pradeep K. Khosla of the Carnegie Mellon University where they used the HMM (Hidden Markov Models) to spot and recognize the gestures captured using Data Glove, where a spotting of Six gestures were made reliably and accurately.

II. SYSTEM FLOW

The image is captured by the camera on real-time basis, this image will be processed for glitches or errors and then processed by the PC Software (in our case MATLAB). Further we resize the image and and scan rows and columns respectively, we define a range of thresholds for the R,G,B colours and convert these pixels to white; all other pixels are converted to black. The process of centroid computation takes place.

After the calculation of the centroid, it is checked gain if the centroid is correctly calculated. If YES, then there takes place a mouse action else if NO, then there is no action which takes place. This process is then repeated in a loop for the continuous movement of the mouse or any other device operation.

This method gives us utmost accuracy and ease of accessibility for device control using gestures.

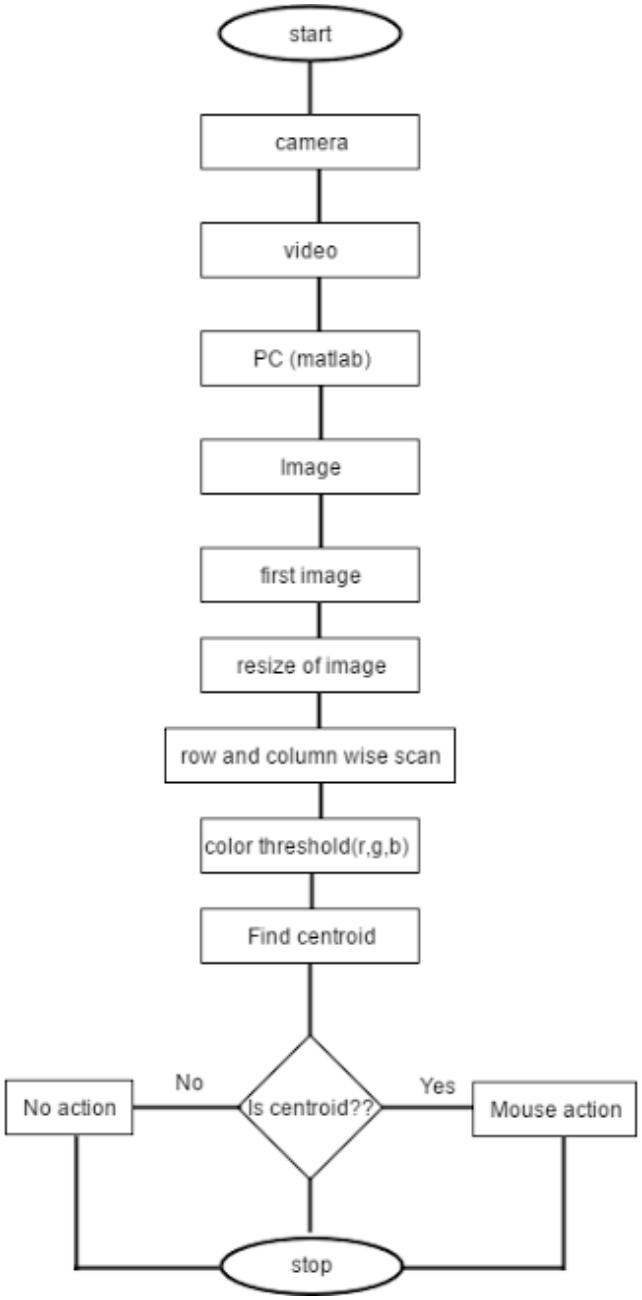

Figure 1. An overview of our hand gesture recognition and gadget Control Flow.

III. HAND GESTURE RECOGNITION

I. *Image Resizing*

First to recognize a hand gesture, we need to resize the input image in order to map camera coordinates to screen coordinates. There are two ways to map from source image to destination image. The first way is to compute the ratio of screen resolution to camera resolution. To determine the x and y coordinates on the screen of a given camera pixel, we use the following equation:

$$x = \frac{x'}{640} \times c_x, \quad y = \frac{y'}{480} \times c_y$$

where (x' , y') is the camera position, (C_x, C_y) is the current screen resolution, and (x, y) is the corresponding screen position of camera position.

The second way is to use the cvResize() function in OpenCV. This function maps a source image to a destination image as smoothly as possible. To accomplish this, the function uses interpolation. Interpolation is a method to add or subtract pixels to an image as necessary to expand or shrink its proportions while introducing minimum distortion. Using this function, we can easily map the position of each input image pixel to a screen position.

In this paper, we used the first method because it is easier to implement, more accurate and preserves more data..

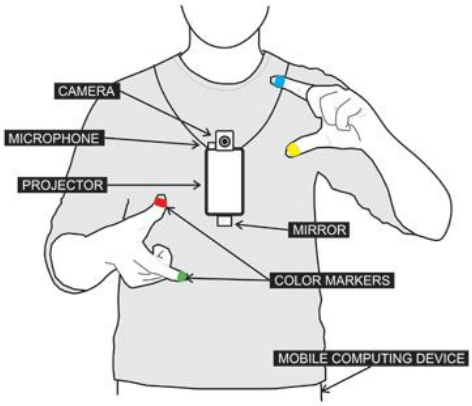

Figure 2. Use of markers and real-time camera .

A. *Segmentation*

Next, we need to separate the hand area from a complex background. It is difficult to detect skin color in natural environments. If we tried to detect skin color in natural environments, we would have to work with a very wide color range because of the variety of illuminations and skin colors and it is possible that other objects whose colors lie in the color range may get detected instead of the user's hand.

Hence, we use colored markers wrapped around on our fingertips for the detection of fingers. Doing this enables us to work in a narrow color range which increases precision.

We use four markers colored blue, green, yellow and red.. To get better results, we converted from RGB color space to YC_bC_r color space, since YC_bC_r is insensitive to color variation. The conversion equation is as follows.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.29900 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500000 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Where Y is luminance, C_b is the blue-difference chroma

component and C_r is the red-difference chroma component. From this information, we detect the marker color by selecting a particular color range from the C_b and C_r values. In this paper, we choose Y, C_r and C_b values for red, blue, yellow and green at their maximum saturation. We also keep the tolerance at a healthy 10% to allow for operations in different illumination levels and with slightly less saturated color markers. (It should be noted that these values were chosen for the convenience of the investigator.) Then we loop over all the image pixels, changing pixels within the marker color range to 0, and all others to 255. Hence, we obtain a binary image. Figure 2 shows the results.

B. Deleting Noise

Using this approach, we cannot get a good estimate of the marker image because of background noise. To get a better estimate of the marker, we need to delete noisy pixels from the image. We use an image morphology algorithm that performs image erosion and image dilation to eliminate noise [1]. Erosion trims down the image area where the colored marker is not present and Dilation expands the area of the Image pixels which are not eroded. Erosion is given by the equation

$$A \ominus B = \{x \mid (B)_x \cap A^c = \phi\}$$

Where A denotes input image and B denotes Structure elements. The Structure element is operated on the Image using a Sliding window and exact matches are marked. Figure 3 shows a graphical representation of the algorithm. Dilation is defined by,

$$A \oplus B = \{x \mid (\hat{B}) \cap A \neq \phi\}$$
$$= \{x \mid [(\hat{B})_x \cap A] \subseteq A\}$$

Where A denotes the input image and B denotes the structure element. The same structure element is operated on the image and if the center pixel is matched, the whole area around that pixel is marked. Figure 4 shows the algorithm. Erosion and Dilation are changed by the shape of B. Thus, B should be selected in advance. We performed erode function with structure of 10x10 square pixels three times and dilate function with 6x6 square pixels structure three times to get a clearer hand. Figure 5 is the result of this algorithm.

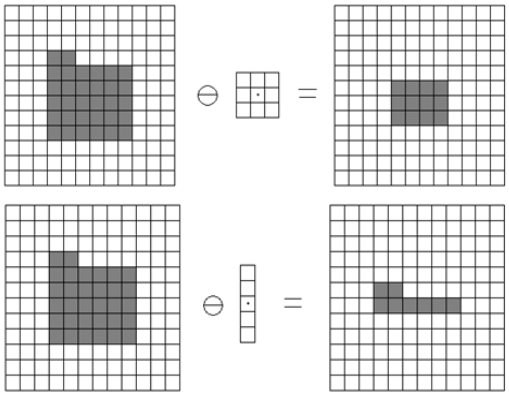

Figure 3. The result of image erosion. The structure element scans the input image. When the structure element matches, the central pixel is kept; when it does not match, all pixels are discarded.

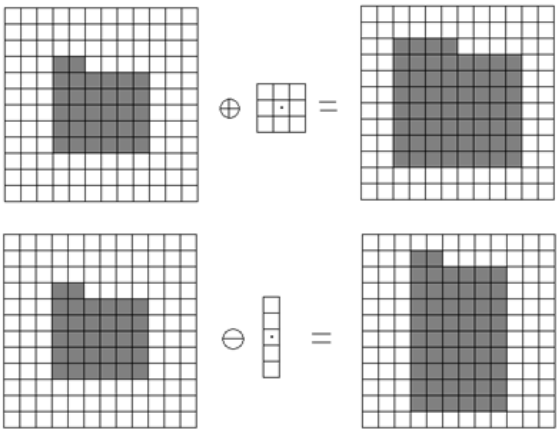

Figure 4. The result of image dilation. The structure element scans the input image. When the center of the structure matches any pixels, the bin of the structure element is filled.

Figure 5. The result of image morphology. Perform erosion and dilation on binary image. (a) Original image. (b) The result of image morphology.

C. *Finding the Center and the Size of Hand*

After segmenting hand region and background, we can calculate the center of the hand with the following equation:

$$\overline{x} = \frac{\sum_{i=0}^{k} x_i}{k} \qquad \overline{y} = \frac{\sum_{i=0}^{k} y_i}{k}$$

Where x_i and y_i are the x and y co-ordinates of the i^{th} pixel in the hand region and k denotes the number of pixels in the region.

After we locate the center of the hand, we compute the radius of the palm region to get hand size. To obtain the size of the hand, we draw a circle increasing the radius of the circle from the center co-ordinate until the circle meets the first black pixel. When the algorithm finds the first black pixel then it returns to the current radius value. This algorithm assumes that when the circle meets the first black pixel, after drawing a larger and larger circle, then the length from the center is the radius of the back of the hand. Thus, the image segmentation is the most significant part because if some of the black pixels are made by shadows and illuminations near the center, then the tracking algorithm will meet earlier than the real background and the size of the hand region becomes smaller than the real hand. This problem breaks this system. Figure 6 shows the results.

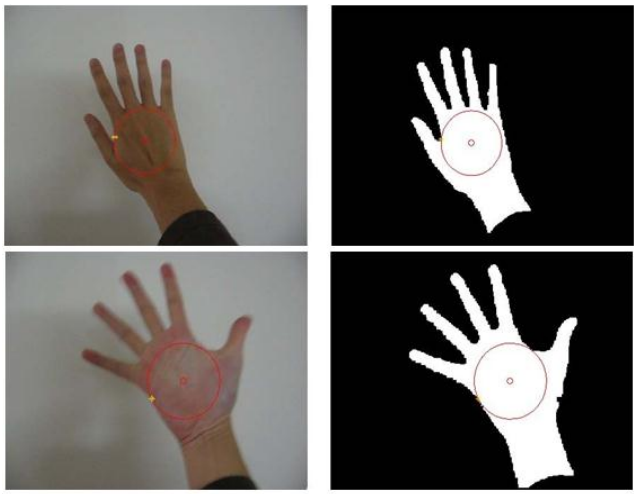

Figure 6. The result of the computed hand size. Drawing a larger and larger circle, we obtained the size of the back of the hand.

D. *Finding Fingertips*

To recognize that a finger is inside of the palm area or not, we used a convex hull algorithm. The convex hull algorithm is used to solve the problem of finding the biggest polygon including all vertices. Using this feature of this algorithm, we can detect finger tips on the hand. We used this algorithm to recognize if a finger is folded or not. To recognize those states, we multiplied 2 times (we got this number through multiple trials) to the hand radius value and check the distance between the center and a pixel which is in convex hull set. If the distance is longer than the radius of the hand, then a finger is spread. In addition, if two or more interesting points existed in the result, then we regarded the longest vertex as the index finger and the hand gesture is clicked when the number of the result vertex is two or more.

The result of convex hull algorithm has a set of vertexes which includes all vertexes. Thus sometimes a vertex is placed near other vertexes. This case occurs on the corner of the finger tip. To solve this problem, we deleted a vertex whose distance is less than 10 pixels when comparing with the next vertex. Finally, we can get one interesting point on each finger. Figure 7 shows the results.

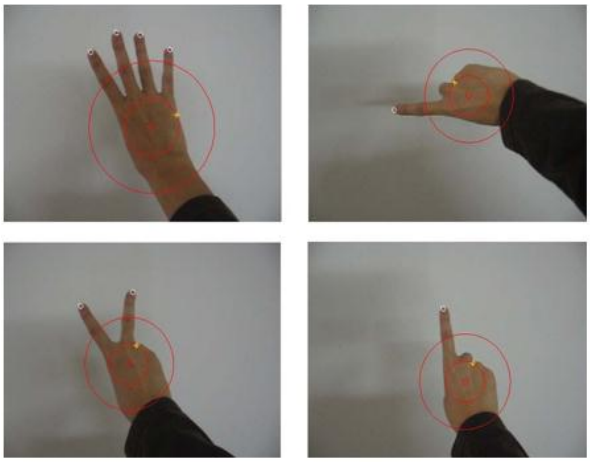

Figure 7. The result of finger tip tracking algorithm. If a tip of a finger is on the outside of the larger red circle, then it can be found by the convex hull algorithm.

IV. CONTROLLING THE MULTI DISCIPLINARY GADGETS

A. *Moving Mouse Cursor*

There will be pre-defined color markers for each specific fingers (for example: Red, Yellow, Green and Blue for Index finger, middle finger, third finger and thumb respectively). The mouse cursor will follow the movements of the index finger.

B. *Clicking the mouse*

If the index finger touches with the thumb finger it will be considered as left click and similarly if the index finger touches the middle finger it will be considered as right click.

C. *Controlling the media player*

In media player, for increasing or decreasing the volume with the index finger we can do upward or downward action. When the middle finger and thumb touches each other it will be considered as ‘play’ and similarly if they are touched twice in succession it will be considered as ‘pause’.

V. EXPERIMENTS AND RESULTS

We first iterated the code and removed bugs to implement various mouse tasks such that left click, right click, double-click, dragging, and scrolling on windows. The tested system is that Core2-Duo T8300, 2GB memory, Microsoft Windows XP, Microsoft Windows 10, Microsoft Life Cam VX-1000 (640x480 resolution, 15fps). It becomes very difficult to compare the speed of cursor by hand gesture and actual mouse. So, instead of comparing with a real mouse, we allowed to use this system to four testers to know

how it can be adapted easily. Our method evaluated the task on the grounds of time taken by each process to get processed. We designed four experiments to get performance. In the first experiment we placed an icon in the middle of the laptop screen and placed the cursor on the top left corner of the screen. We then measured the time the cursor took to reach from its initial position to the icon. In the second experiment we placed the icon in the exact same position before and then measured the time it took to open the drop down list of an icon. In the final experiment we opened the website (www.google.com) from a web browser and measured the time the cursor took from its initial position.

Every tester was new in this system. The clicking task showed similar times with all testers. However, the scrolling result showed the time as unstable. The reason is that the focus of the camera lens works automatically so when the index finger went to the border on the screen then part of the hand colors becomes dim. Eventually, hand segmentation failed because colors could not be on the color range which is in the segmentation part. Thus, we could not get a clear shape of the hand from the video streaming.

VI. PROBLEM DISCUSSION AND SOLUTIONS

The problem arises when the hand shook a lot. We used real time camera so when lighting occurs the position of the hand changes and hence the convex hull algorithm's property of finger point detection also changes. Then the mouse cursor pointer shakes fast. To fix this problem, we added a code that the cursor does not move if the difference of the previous and the current finger tips position is within 5 pixels. This constraint worked well but it makes it difficult to control the mouse cursor sensitively. Another problem arises due to illumination is the segmentation of the hand becomes difficult, since the hand reflects all light sources, the hand color is changed according to the place. If hand shape is not good then our algorithm cannot work well because our algorithm assumes the hand shape is well segmented. In order to determine the approximate radius value of the hand the segmentation or shape of the hand must be perfectly aligned in front of web-camera.

For finding the center of hand, it has a problem to find the center accurately. If the camera showed a hand with wrist, then the center will move a little towards the wrist because the color of the wrist is the same as hand color. Therefore, it causes that algorithm system to fail because if the center is moved down, then the radius of the hand can be smaller than the actual size. Additionally the center will move down and the algorithm used will show us the system fail. And then every time the thumb will be shown outside the circle which is unacceptable.

VII. CONCLUSION

For finding the center of hand, it has a problem to find the center accurately. If the camera showed a hand with wrist, then the center will move a little towards the wrist because the color of the wrist is the same as hand color. Therefore, it causes that algorithm system to fail because if the center is moved down, then the radius of the hand can be smaller than the actual size. Furthermore, the circle will move down and become smaller so when a user crooks his or her hand then the finding finger tips algorithm can also fail because a thumb can be placed outside of the circle every time.

REFERENCES

[1] Computer vision based mouse, A. Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E. Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASS). IEEE International Conference
[2] Virtual mouse vision based interface, Robertson P., Laddaga R., Van Kleek M. 2004.
[3] Vision Based Men-Machine Interaction http://www.ceng.metu.edu.tr/~vbi/
[4] Chu-Feng Lien, Portable Vision-Based HCI - A Real-time Hand Mouse System on Handheld Devices.
[5] Hailing Zhou, Lijun Xie, Xuliang Fang, Visual Mouse: SIFT Detection and PCA Recognition, cisw, pp.263- 266, 2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007), 2007
[6] James M. Rehg, Takeo Kanade, DigitEyes: Vision-Based Hand Tracking for Human-Computer Interactioning, Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects, Austin, Texas, November 1994, pages 16-22.
[7] Gary Bradski, Adrian Kaehler, Learning OpenCV, O'Reilly Media, 2008.
[8] Asanterabi Malima, Erol Ozgur, and Mujdat Cetin, A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control
[9] J. Serra, Image Analysis and Mathematical Morphology, New York: Academic Press, 1983.
[10] K. Homma and E.-I. Takenaka, "An image processing method for feature extraction of space-occupying lesions," journal of Nuclear Medicine 26 (1985): 1472-1477.
[11] www.cs.cmu.edu/~cyberscout/publications/IROS99_GBC.pdf.