Virtual Mouse Using a Webcam

1. INTRODUCTION

Since the computer technology continues to grow up, the importance of human computer interaction is enormously increasing. Nowadays most of the mobile devices are using a touch screen technology. However, this technology is still not cheap enough to be used in desktop systems. Creating a virtual human computer interaction device such as mouse or keyboard using a webcam and computer vision techniques can be an alternative way for the touch screen. In this study, finger tracking based a virtual mouse application has been designed and implemented using a regular webcam. The motivation was to create an object tracking application to interact with the computer, and develop a virtual human computer interaction device.

"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]."

In this study, a color pointer has been used for the object recognition and tracking. Left and the right click events of the mouse have been achieved by detecting the number of pointers on the image.

2. SYSTEM DESCRIPTION

In the object tracking application one of the main problems is object detection. Instead of finger tips, a color pointer has been used to make the object detection easy and fast. A circle blue sticker is used as a color pointer in this study. To simulate the click events of the mouse three fingers with three color pointers has been used.

The basic algorithm is as follows;

* Set a pointer in the image
* Detect the pointer using the defined color information
* Define the region and the center of the pointer and draw a bounding box around it

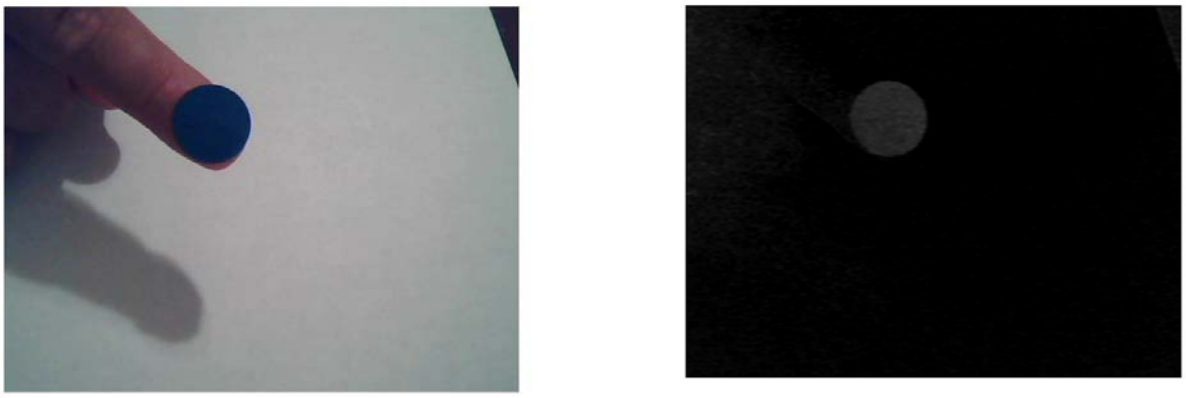
* Track the motion of the pointer
* Move the cursor according to the position of the center of the pointer
* Simulate the single and the double left click and the right click of the mouse

2.1.Color Detection

To detect the blue color of the pointer, MATLAB's built in "*imsubtract*" function has been used. Imsubtract function can be used as;

Z = *imsubtract*(X,Y)

Where; it subtracts each element in array Y from the corresponding element in array X and returns the difference in the corresponding element of the output array Z. X and Y are real, non-sparse numeric arrays of the same size and class, or Y is a double scalar. The array returned, Z, has the same size and class as X unless X is logical, in which case Z is double.



(a) (b)

Figure 1. (a) Input image, (b) after using *imsubtract* and detect the blue color.

2.2 Filtering the Noise

After detecting the blue color in the input image, a median filter has been used to filter out the noise. Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.

As seen in the figure 2 (a) and (b), there wasn't really noise since the illumination was enough when we were getting this sample snapshot.

(a) (b)

Figure 2. (a) Before filtering the image, (b) after filtering the image using median filter.

2.3 Converting the Gray Scale Image to Binary Scale Image

To convert the gray scale image to binary scale image MATLAB's built in *"im2bw"* function has been used. Function can be used as;

BW = *im2bw*(I, level)

Where; it converts the grayscale image I to a binary image. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). Specify level in the range [0,1]. This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 is midway between black and white, regardless of class.

In our study, the threshold 0.15 gave the best result for the large range of illumination change.

(a) (b)

Figure 3. (a) Gray scale image, (b) binary scale image

2.4 Removing All the Small Areas

To get the best accurate number of the object detected in the image, all the areas other than the pointer need to be removed. To do this, we have used MATLAB's "*bwareaopen*" function.

Function can be used as;

BW2 = bwareaopen(BW, P)

Where; it removes from a binary image all connected components (objects) that have fewer than P pixels, producing another binary image, BW2. We set a threshold 300 pixels for this study.

(a) (b)

Figure 4. (a) Before removing the small areas, (b) after removing the small areas (less than 300 pixels)

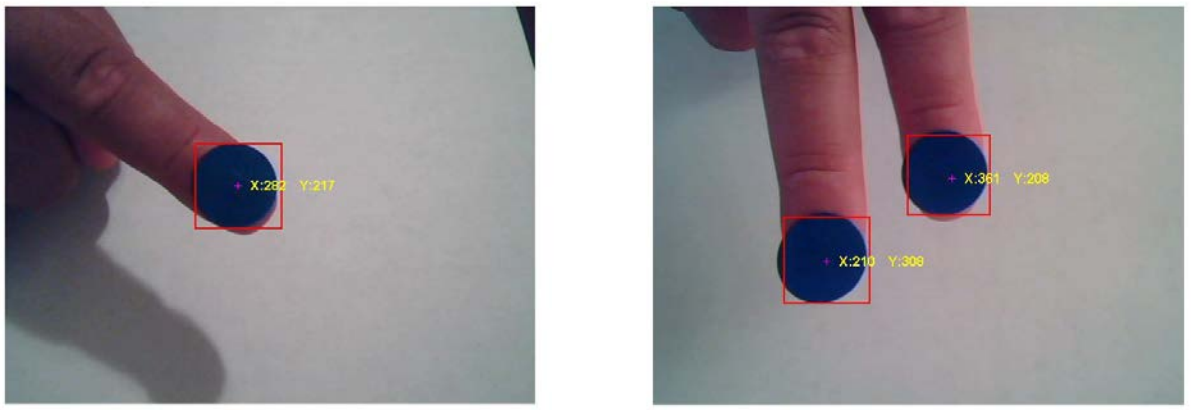
2.5 Find the Center of the Object and Draw a Bounding Box

After removing all the connected components (objects) other than the pointer, using MATLAB's "*bwlabel*" function the pointer can be labeled. In other words the region can be detected. To get the properties of the region such as center point or bounding box etc., MATLAB's built in "*regionprops*" function can be used as;

STATS = *regionprops*(BW, properties)

Where; it measures a set of properties for each connected component (object) in the binary image, BW. The image BW is a logical array; it can have any dimension. In our application

since we need to get the center of the object and the bounding box, we used only these two properties.



(a) (b)

Figure 5. (a) Input image using one pointer, (b) input image using two pointers

2.6 Move the Cursor According to the Position of the Pointer

To move the cursor desired (X,Y) coordinates, MATLAB has a "*set(0,'PointerLocation',[x,y])*" function.

However, MATLAB doesn't have a function for mouse click events. To move the mouse and to simulate the mouse click event "*Java class java.awt.Robot*" which has all these abilities can be used. Since the resolution of the camera is smaller than the resolution of the computer monitor, the (X,Y) coordinates are needed to be scaled. In our application the resolution of the input image was 640x480 and the resolution of the computer monitor was 1280x800. So, we scaled the (X,Y) of the center of the pointer as (4xX-400, 3.33xY-333) to be able to move the cursor along all the window.

2.7 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 I have used another pointer. 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, and the third pointer is being used to simulate the right click of the mouse.

3. CONCLUSION AND FUTURE WORKS

In this study, an object tracking based virtual mouse application has been developed and implemented using a webcam. The system has been implemented in MATLAB environment using MATLAB Image Processing Toolbox. As an object a blue color sticker is used to make the detection easy and fast. Object detection and motion tracking worked very well. Using the pointer moving the cursor and the simulating the mouse click events also worked well. However, system has some disadvantages such as; being invariant to illumination up to some scale, and movement of the cursor is very sensitive to motion. Because of this reason, to control the cursor, pointer cannot be used on the air efficiently.

For the future works, instead of a color pointer directly finger tips can be detected. 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.

References:

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

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 MouseMovement using a Real Time Camera", 2008, Brown University,Providence ,RI ,USA, Department of computer science

4- http://www.mathworks.com/matlabcentral/fileexchange/28757-tracking-red-color-objects-using-matlab

5- http://www.mathworks.com/help/techdoc/

6- http://www.mathworks.com/support/solutions/en/data/1-2X10AT/index.html?solution=1-2X10AT