INTRODUCTION

As the computer technology is grow up, the importance of human computer interaction is rapidly increasing. Most of the mobile devices and laptops are using touch screen technology. But this Technology is still not cheap enough to be used on desktop systems. Creating a virtual human computer interactive module such as mouse or keyboard, can be an alternative way for the touch screen. The motivation is to create an object tracking application to interact with the computer, and develop a virtual human computer interaction device.

A virtual mouse is software that allows users to give mouse inputs to a system without using an actual mouse. To the extreme it can also be called as hardware because it uses an ordinary web camera. A virtual mouse can usually be operated with multiple input devices, which may include an actual mouse or a computer keyboard. Virtual mouse which uses web camera works with the help of different image processing techniques. 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 images. The hand movements of a user are mapped into mouse inputs. A web camera is set to take images continuously. The user must
have a particular color in his hand so that when the web camera takes image it must be visible in the acquired image. This color is detected from the image pixel and the pixel position is mapped into mouse input.

Depending upon the size of the image taken by camera, various scaling techniques are used because the pixel position in the image will not have a correspondence with screen resolution.

In this paper, the mouse cursor movement and click events are controlled using a camera based on color detection technique. Here real time video has been captured using a Web- Camera. The user wears colored tapes to provide information to the system. Individual frames of the video are separately processed. The processing techniques involve an image subtraction algorithm to detect colors. Once the colors are detected, the system performs various operations to track the cursor and performs control actions. No additional hardware is required by the system other than the standard webcam which is provided in every laptop computer.

The purpose of the project is to create a virtual mouse that works with the help of a web camera. In this project a camera continuously takes images of hand movements of a user, which is then mapped into mouse inputs. This means that we can give inputs to computer without having any physical connection with the computer and without having any hardware movements.

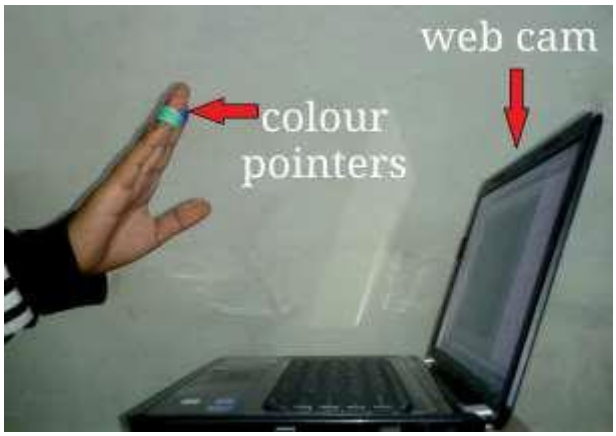
Proposed system

System Approach

1. Capturing real time video using Web-Camera.
2. Finding the region of the image and calculating its centroid.
3. Tracking the mouse pointer using the coordinates obtained from the centroid.
4. Simulating the left click and the right click events of the mouse by assigning different color pointers.

Capturing the Real Time Video

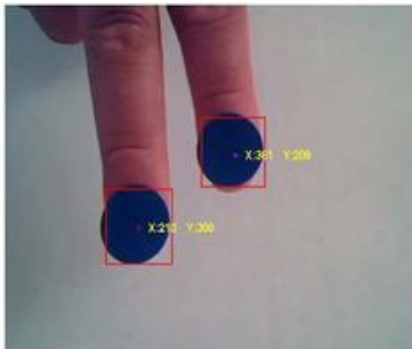
For the system to work we need a sensor to detect the hand movements of the user. The webcam of thecomputer is used as a sensor. The webcam captures the real time video at a fixed frame rate and resolution which is determined by the hardware of the camera. The frame rate and resolution can bechanged in the system if required .Computer Web cam is used to capture the Real Time Video Video is divided into Image frames based on the FPS (Frames per second) of the camera



Finding Centroid of an object and plotting Bounding Box

For the user to control the mouse pointer it is necessary to determine a point whose coordinates can be sent to the cursor. With these coordinates, the system can control the cursor movement. An inbuilt function in is used to find the centroid of the detected region. The output of function is a matrix consisting of the X (horizontal) and Y (vertical) coordinates of the centroid. These coordinates change with time as the object moves across the screen.

- Centroid of the image is detected and a bounding box is drawn around it.
- Its co-ordinates are located and stored in a variable



Tracking the Mouse pointer

Once the coordinates has been determined, the mouse driver is accessed and the coordinates are sent to the cursor. With these coordinates, the cursor places itself in the required position. It is assumed that the object moves continuously, each time a new centroid is determined and for each frame the cursor obtains a new position, thus creating an effect of tracking. So as the user moves his hands across the field of view of the camera, the mouse moves proportionally across the screen. There is no inbuilt function in which can directly access the mouse drivers of the computer. But code supports integration with other languages like C, C++, and JAVA. Since java is a machine independent language so it is preferred over the others. A java object is created and it is linked with the mouse drivers. Based on the detection of other colors along with red the system performs the clicking events of the mouse. These color codes can be customized based on the requirements.

Problems and Drawbacks

Since the system is based on image capture through a webcam, it is dependent on illumination to a certain extent. Furthermore the presence of other colored objects in the background might cause the system to give an erroneous response. Although by configuring the threshold values and other parameters of the system this problem can be reduced but still it is advised that the operating background be light and no bright colored objects be present. The system might run slower on certain computers with low computational capabilities because it involves a lot of complex calculations in a very small amount of time. However a standard pc or laptop has the required computational power for optimum performance of the system. Another fact is that if the resolution of the camera is too high then

Pr-requisites

only software required is

- python IDE
- open CV
- numpy
- wx
- pynput

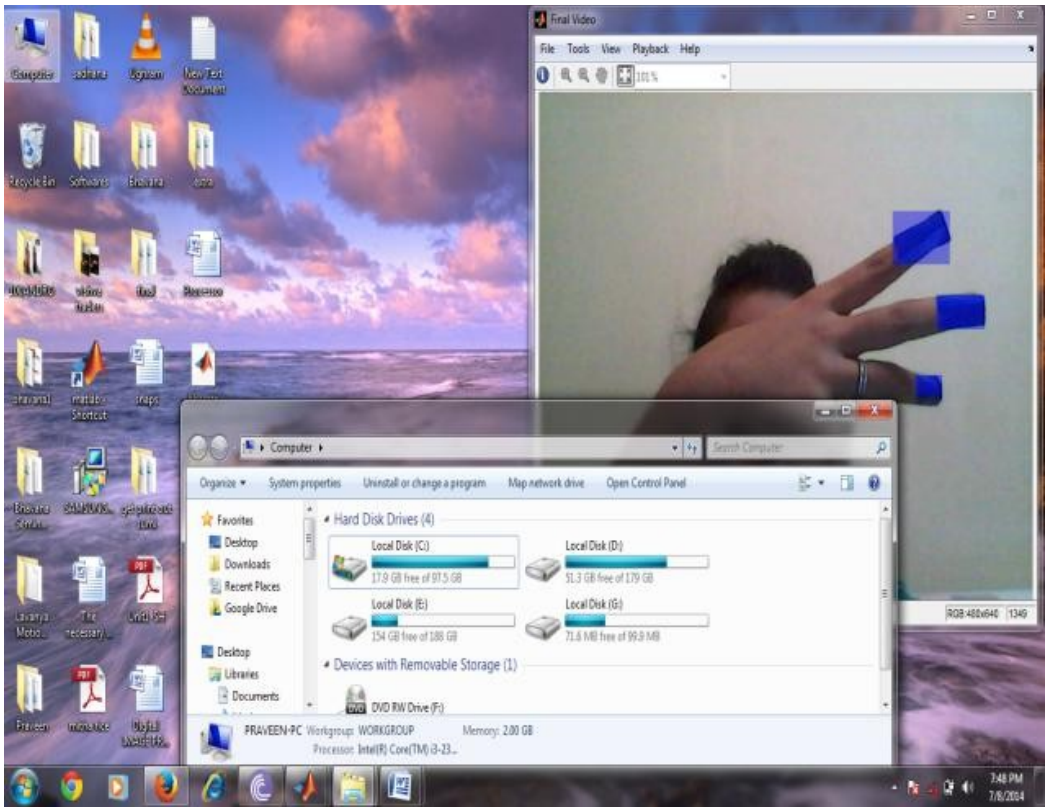
Implementation

CODE:

```
import cv2
#from PIL import Image

import numpy as np
from pynput.mouse import Button, Controller
import wx
mouse=Controller()
app=wx.App(False)
(sx,sy)=wx.GetDisplaySize()
(camx,camy)=(320,240)

lowerBound=np.array([110,50,50])
upperBound=np.array([130,255,255])

cam = cv2.VideoCapture(0)
cam.open(0)
cam.set(3,camx)
cam.set(4,camy)
kernelOpen=np.ones((5,5))
kernelClose=np.ones((20,20))
mLocOld=np.array([0,0])
mouseLoc=np.array([0,0])
DampingFactor=2
pinchFlag=0
openx,openy,openw,openh=(0,0,0,0)
while True:
    ret, img=cam.read()
    img=cv2.resize(img,(340,220))
    cv2.imshow("Shubham's image", img)
    imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    mask=cv2.inRange(imgHSV,lowerBound,upperBound)
    maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
    maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)
    maskFinal=maskClose

    _, conts, h =
cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NO
NE)
    if(len(conts)==2):
        if(pinchFlag==1):
            pinchFlag=0
            mouse.release(Button.left)


        x1,y1,w1,h1=cv2.boundingRect(conts[0])
```

```
        x2,y2,w2,h2=cv2.boundingRect(conts[1])
        cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
        cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
        cx1=x1+w1/2
        cy1=y1+h1/2
        cx2=x2+w2/2
        cy2=y2+h2/2
        cx=(cx1+cx2)/2
        cy=(cy1+cy2)/2
        cv2.line(img, (cx1,cy1),(cx2,cy2),(255,0,0),2)
        cv2.circle(img, (cx,cy),2,(0,255,0),2)
        mouseLoc=mLocOld+((cx,cy)-mLocOld)/DampingFactor
        mouse.position=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy)
        while mouse.position!=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy):
           pass
        mLocOld=mouseLoc
        openx,openy,openw,openh=cv2.boundingRect(np.array([[[x1,y1],[x1+w1,y1+h1],
[x2,y2],[x2+w2,y2+h2]]]))
    elif(len(conts)==1):
        x,y,w,h=cv2.boundingRect(conts[0])
        if(pinchFlag==0):
            if(abs((w*h-openw*openh)*100/(w*h))<30):
                pinchFlag=1
                mouse.press(Button.left)
                openx,openy,openw,openh=(0,0,0,0)
        else:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            cx=x+w/2
            cy=y+h/2
            cv2.circle(img,(cx,cy),(w+h)/4,(0,0,255),2)
            mouseLoc=mLocOld+((cx,cy)-mLocOld)/DampingFactor
            mouse.position=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy)
            while mouse.position!=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy):
                pass
            mLocOld=mouseLoc
            mouse.press(Button.left)
    cv2.imshow("Laptop camera",img)
    if cv2.waitKey(5) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
#cv2.imshow("Image", img)
#img = cv2.imread("xyz.jpg", 0)
```

Screen shot



Applications

- When there is a hardware issue we can use

Conclusion

Many ideas were put forward but they all required physical movement of hardware. Another idea put forward was to use the principle of photoelectric effect. But for that a special hardware is needed and it is not economically feasible. So the final decision is to develop a virtual mouse which uses simple and cheap image processing techniques.

References

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