COMPUTER VISION BASED MOUSE CONTROL USING OBJECT DETECTION AND MARKER MOTION TRACKING

[1]M. Shanmugaraj, [2]K Senthil Vijay, [3]K. Devi, [4]Dr. G. JawaherlalNehru

[1,2,3,4]Assistant Professor, St. Martins Engineering College, Secunderabad, Telangana

ABSTRACT

There have been a lot of developments towards the Humans Computers Interaction (HCI). The importance of computers is increasing constantly. Computer can be used for many purposes. We often used hardware devices such as mouse and keyboard to interact with the computers. In today's world, technologies are evolving day by day. One of the example is the Human-Computer Interface (HCI). Many modules have been developed to help the physical world interact with the digital world. Here, the proposed paper serves to be a new approach for controlling mouse movement using Colored object and marker motion tracking. The project mainly aims at mouse cursor movements and click events based on the object detection and marker identification. The software is developed in Python Language and OpenCV and PyAutoGUI for mouse functions. We have used colored object to perform actions such as movement of mouse and click events. This method mainly focuses on the use of a Web Camera to develop a virtual human computer interaction device in a cost effective manner.

Keywords**:** Color Detection, HCI, PyAutoGUI, Marker Motion Identification, Object Detection

I. INTRODUCTION

The importance of computers is increasing constantly. Computer can be used for many purposes. We often used hardware devices such as mouse and keyboard to interact with the computers. In today"s world, technologies are evolving day by day. One of the example is the Human-Computer Interface (HCI). In a wired mouse there is no scope to extend limit and have to carry it everywhere. In the wireless mouse, one should have a Bluetooth installed in the system with Bluetooth dongle. Touch screen is little expensive for average users. An alternative way can be the creation of HCI device virtually.

In recent years, research efforts seeking to provide more natural, human-centered means of interacting with computer have gained growing interest. There are various opportunities to apply sensor technologies to heightened the user and computer experience. Computer Vision-Based Mouse is a system to control the cursor of our computer without using any physical device even a mouse.

Our system basically used image processing, object detection and marker motion tracking to control the mouse activities such as its movement, left-click, right-click and double click. The user will simply hold the colored object of his choice, place it in the viewing area of the webcam and ROI, and thus controlling the mouse by color detection also the workspace required could be reduced. One tape will be used for controlling the movement while other colored tape can be used for click events. Our project will provide a new experience for HCI.

The rest of the paper is as follows. In Section 2, we present similar works done by the other researcher. In Section 3, we provide the system overview of what this project is about followed by Section 4 where will present system setup and technologies with an overview of the functional block diagram. Section 5 and 6 concludes the paper with results and conclusion respectively.

II. LITERATURE REVIEW

Many researchers have tried to interact with computer through video devices and web camera. Each of them used different approaches to make a mouse cursor move without any hardware. One method by Erdem[1] et al have control of mouse cursor using image segmentation and gesture control. In reference [2] K. Pendke et al used hand gesture technology to interact with the computer. They have used an intuitive method to detect hand gesture. In this approach, EmguCV SDK technology was used instead of more traditional Matlab in order to achieve more accuracy and have a more control over the source code. [3] Abhirup Ghosh et al idea was based on color detection. But the coloured objects in the background may cause a problem like give an erroneous response.

In this paper was inspired by a paper of Ankur Yadav[4] et al where they used web camera and colored tapes to control mouse cursor.But they used Matlab for image processing and mouse action events. Deeksha Verma[5] et al in their journal also used Matlab for image processing. But here we propose a system to use OpenCV library for image processing and Python"s own GUI module PyAutoGUI to control the mouse cursor and its clicking events.

III. SYSTEM OVERVIEW

Computer Vision-Based Mouse is a system to control the cursor of our computer without using any physical device even a mouse. Here we will essentially have a colored object in our hand. The video of the motion of our palm has been captured by the web-camera which acts as a sensor. The colored objects are tracked and using their motion, the cursor of the mouse is controlled. In our work, we have used 4 colors for 4 typical actions of the mouse. Yellow color for mouse cursor movement, green color for left click, orange color for right-click and blue color for double click.

In order for it to work, we will simply use respective colors within the viewing area of the camera. The colored objects should be placed in the Region of Interest(ROI). The video generated by the camera is detected and analyzed using image processing and the computer cursor moves or displays its click events according to color movements, of which all the details are provided below. Mouse handling activities are achieved by PyAutoGUI which is the python GUI module.

IV. METHODS AND TECHNOLOGIES USED

The proposed system is a computer vision application that is based on real time application system. It makes the use of OpenCV for image processing and image acquisition and PyAutoGUI for handling mouse control in order to replace the actual mouse with the colored object. The basic block diagram of computer vision pipeline is shown in figure 1.

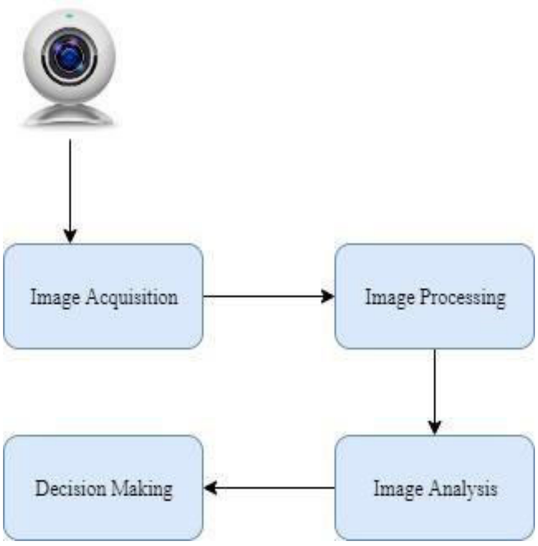


Fig 1 Block Diagram of Computer Vision Pipeline

Each Steps are further divided into smaller steps, based on the application and the project.

A. Image Acquisition

Image or the frame that is detected by the web-cam is acquired as the digital representation of the visual characteristics of the physical world. An image sensor or the web-cam is used to detect and capture the information required to make an image.

B. Image Processing

Image acquired are then processed in the next step. The signals in the acquired images are filtered to remove the noise or any irreverent frequencies. If needed, images are padded and transformed into a different space, so to make them ready for the actual analysis.

C. Image Analysis

The processed image is analyzed to extract useful information. This step involves many important image properties like pattern identification, color recognition, object recognition, feature extraction, motion tracking, and image segmentation.

D. Decision Making

High dimensional data obtained from all the above steps are used to produce meaningful numerical information, which leads to making decisions.

The detailed working of the proposed system is shown in the following flowchart.

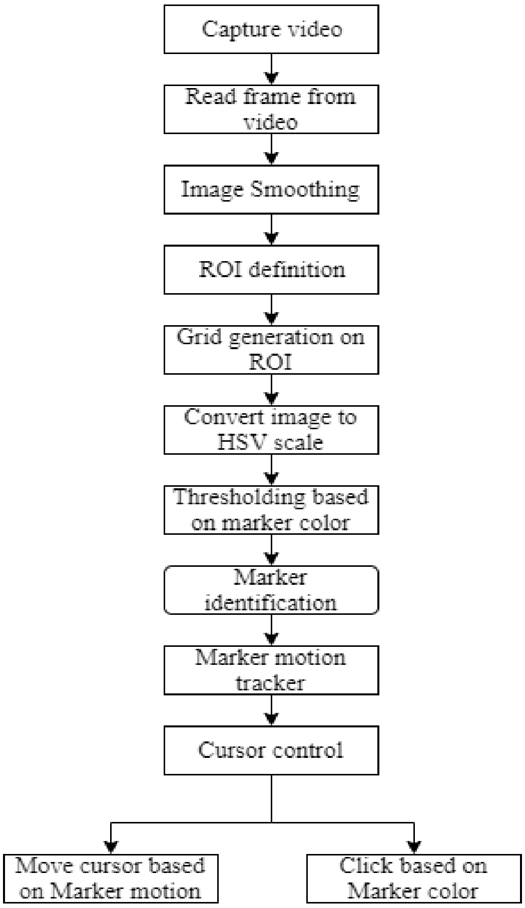


Fig 2 Flowchart of the proposed work

A. Capture video

In the proposed architecture, the input is the colored object which is given to the web-camera. The camera reads the frame of the video. Here, the web-camera is used as a sensor which will help in detecting and tracking the object. Video is a collection of multiple visual images and divided into Frame Rate which ranges from 6 or 8 per second for the old mechanical camera and 120 or more frames per second for a more professional camera.

Video can be captured by OpenCV function:

```
cam = cv2.VideoCapture(index)
```

(1) Where: VideoCapture() is a function of OpenCV which is used to capture the image from the web cam.

Index is a no. of argument or no. of camera that we want to set.cam is output where that function will be stored.

B. Read Frame from the video

After the video is captured by the webcam, frames are read from video by the function „cam.read()‟, where the cam is the output where the video is captured. After the frames are obtained by the OpenCV function, the image that we get is inverted.

That means if we move our object in the right direction, the image of the pointer will move towards the left and vice-versa. It is similar when we stand in front of the mirror where left is detected as right and right is detected as left. To avoid it, we need to vertically flip the Image[].In OpenCV, the image can be conveniently flipped vertically by the function. cv 2.flip(0) .Here 0 is the index because we want to flip our image vertically. The image is in RGB format.

C. Image Smoothing

We need to smooth our image i.e, blur an image. Image Smoothing is useful for removing high frequencies of contents like noise and edges from the image. Image Smoothing is very effective in reducing the Pixelation of an image. Smoothing replaces a pixel value with the average of values of the pixels in its neighborhood. It makes use of Kernel which is a small matric used to apply the smoothing algorithm on the image.

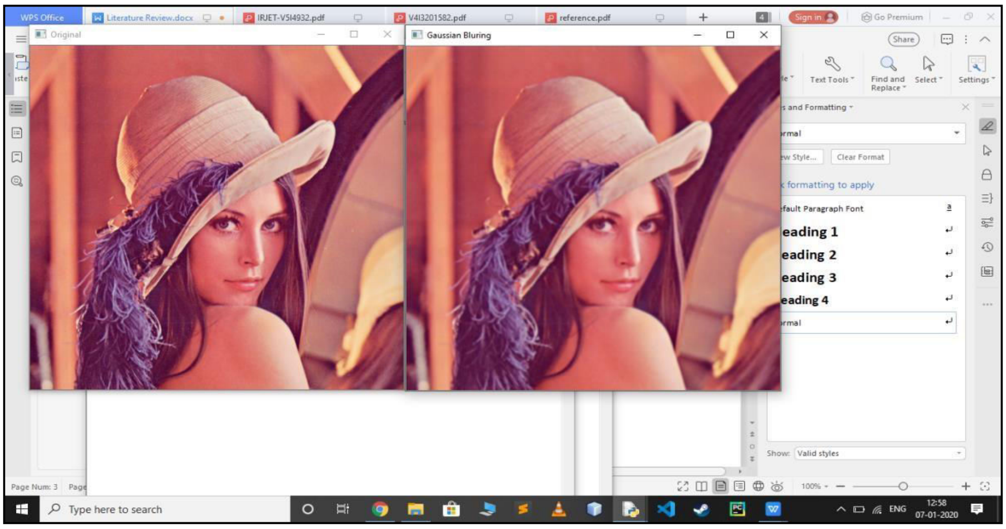


Fig 3 Original vs Blur

D. ROI Definition and Grid Generation

After acquiring the image, applying image smoothing and removing the noise, we need to create a Region of Interest(ROI) and a grid. This will increase the accuracy of the marker identification and tracking. The grid will help in translating the position of the marker to the cursor movement. ROI is where we will perform all the operations of the mouse and where our sensor will detect the marker. We have set the ROI in the top left corner of the screen. The dimension of the grid is 3x3 and the color of the outline is red. After the generation of ROI, We need to convert the image in BGR format to HSV format because it is easier to identify Coloured objects in the HSV space by using the function:
cv2.cvtColor(image, cv2.COLOR_BGR2HSV) .

After converting the image from HSV, we will threshold the image to identify the objects of the specific color range. This technique converts a grayscale image into a binary image, based on some criteria. It is mainly used for object detection.

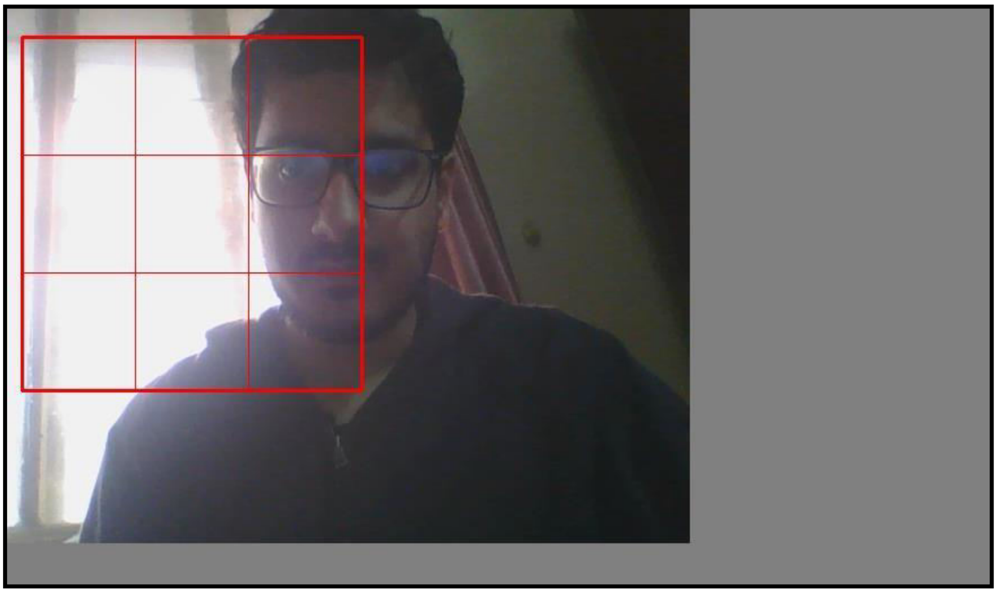


Fig 4 ROI and Grid Generation

Once all the objects of the specific color are separated using color recognition, we will identify the object of our interest. Now that the image is successfully converted into a binary image, we will find the contours of the colored objects from the binary image.

E. Marker Identification

Marker Identification are of three steps:

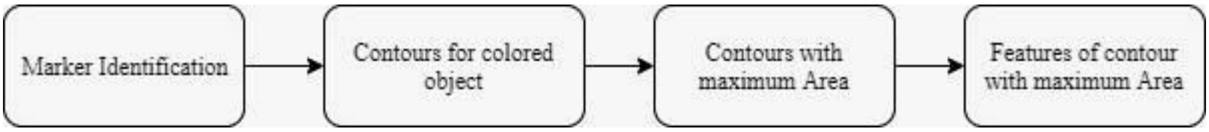

Marker Identification
Contours for colored object
Contours with maximum Area
Features of contour with maximum Area

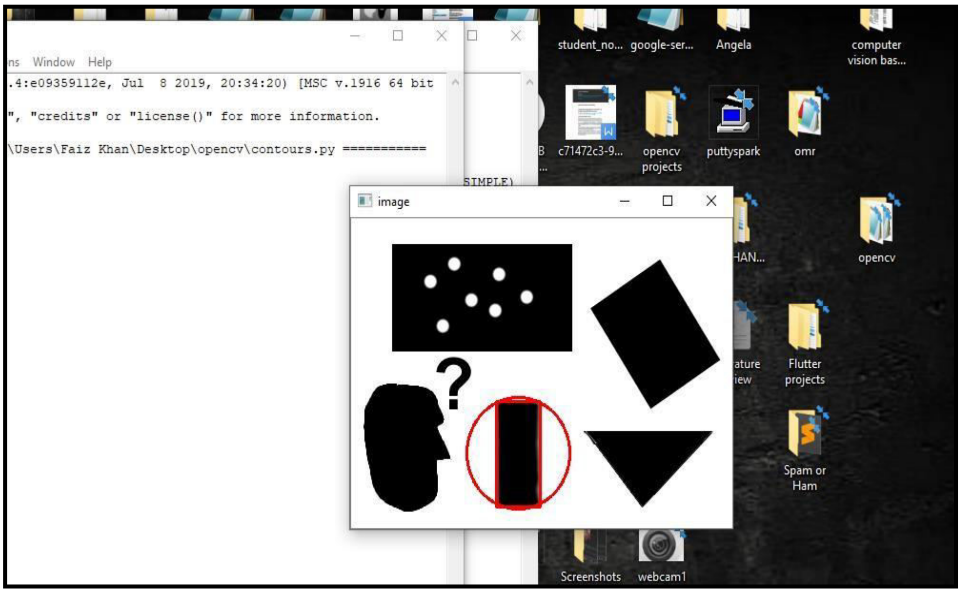
Fig 5 Stages of Marker Identification

Marker Identification identifies the marker from all the object recognized in the image once the markers are identified in each frame.

1) Contours for colored object: Contours is a curve joining all the continuous points and having the same color or intensity. It is a python list of all the contours in the image. Each contour is a numpy array of x,y coordinates of boundary points of the object. We had found the contours of the colored object by cv2.find Contours(image, mode, method). This method outputs a modified image, the contours, and the hierarchy. Hierarchy is the representation of the relationship between contours found from an image.

2) Contours with maximum Area: Contours with maximum Area acts as contours of interest. It identifies as the object. Wehave ignored the smaller contours as they might be due to some error in setting the bounds or some extraneous pixels that do not belong to the specific object.

3) Features of contour with maximum Area: We have used contour features like centroid and bounding rectangle to track the motion of an object. The identified object can be easily tracked from its centroid and the bounding box. The object is bounded with the rectangle and circle as shown below.



```
.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
", "credits" or "license()" for more information.
\Users\Faiz Khan\Desktop\opencv\contours.py ===========
SIMPLE)
```

Fig 6 Contours Identification

F. Marker Motion Tracker

Once the markers in each frame are identified, the motion is tracked in different frames by locating the centroid of the markers in the Region of interest (ROI). We have made use of centroid location relative to the grid. Contour features are useful to find the contour around the markers on the palm. After getting the centroid of the marker, we got the location of the centroid in the grid that we have created. It can fall in any of the 9 regions as shown:

TABLE I

LOCATION OF THE CENTROID IN GRID

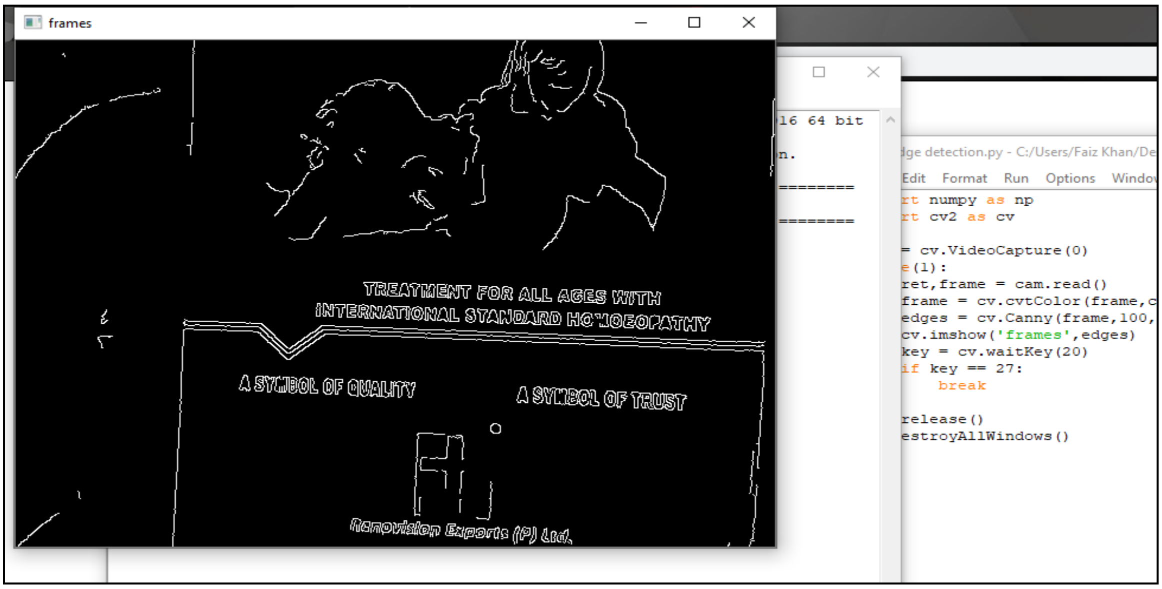
Top-Left
Top-Center
Top-Right
Mid-Left
Mid-Center
Mid-Right
Bottom-Left
Bottom-Center
Bottom-Right

In the above TABLE I, we can locate where the object is from that 9 centroids in the above table. If it is in the Top-Left region, the cursor should move towards the top left part of the screen. If the marker is in the Top-Center region of the grid, the cursor should move toward the top of the screen and so on. If we hold the marker in the central region, the cursor should not move and stay still. This is how we control the cursor, based on the marker motion.

V. RESULT AND EVALUATION

In this paper, we tried to improvise the interaction between humans and the machine by controlling the cursor of the mouse without any physical device. The proposed system tried to control the functions of the mouse pointer by detecting any yellow colored object, right-click by orange color, left click and double click by detecting green and blue colored objects respectively.

A. Edge Detection



frames
TREATMENT FOR ALL AGES WITH
INTERNATIONAL STANDARD HOMOEOPATHY
A SYMBOL OF QUALITY
A SYMBOL OF TRUST
Renovision Exports (P) Ltd.
dge detection.py - C:/Users/Faiz Khan/De
Edit Format Run Options Windo

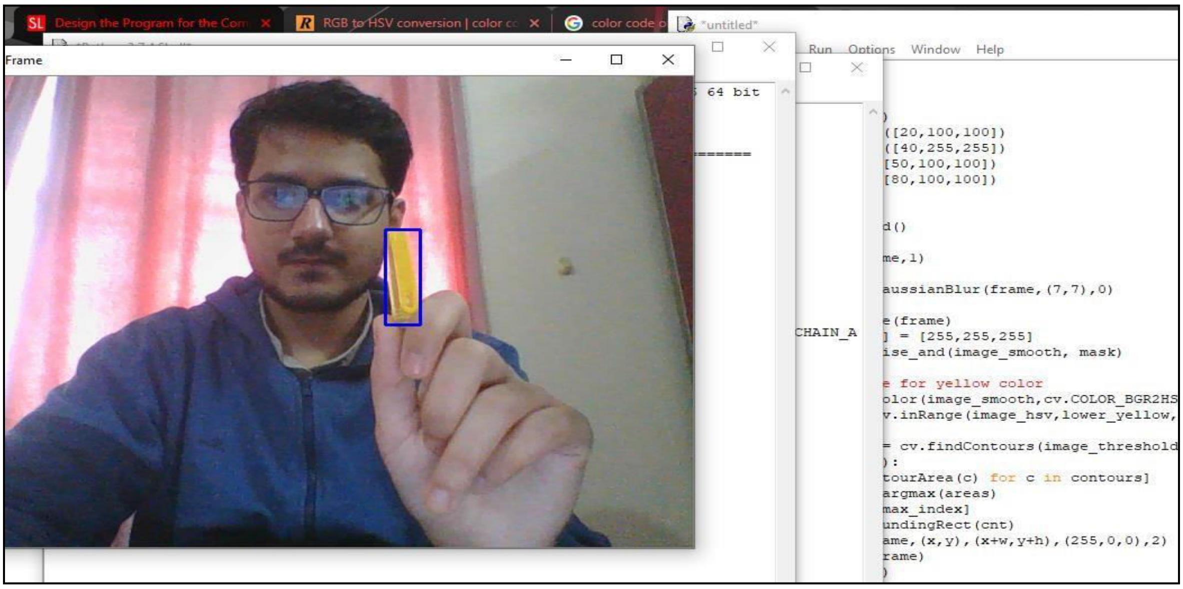
```
16 64 bit

n.

========

========
```

```
rt numpy as np
rt cv2 as cv

= cv.VideoCapture(0)
e(1):
ret,frame = cam.read()
frame = cv.cvtColor(frame,c
edges = cv.Canny(frame,100,
cv.imshow('frames',edges)
key = cv.waitKey(20)
if key == 27:
    break

release()
estroyAllWindows()
```

Fig 7 Canny Edge Detection

Figure 7 displays the edge detection of the object by canny edge detection algorithm which is a very powerful edgedetection algorithm. As we can see, everything is invisible except for the edges in the object.

B. Object Tracking



```
)
([20,100,100])
([40,255,255])
[50,100,100])
[80,100,100])

d()

me,1)

aussianBlur(frame,(7,7),0)

e(frame)
] = [255,255,255]
ise_and(image_smooth, mask)

e for yellow color
olor(image_smooth,cv.COLOR_BGR2HS
v.inRange(image_hsv,lower_yellow,

= cv.findContours(image_threshold
):
tourArea(c) for c in contours]
argmax(areas)
max_index]
undingRect(cnt)
ame,(x,y),(x+w,y+h),(255,0,0),2)
rame)
)
```

Fig 8 Object Tracking

As stated earlier, we have chosen yellow-colored objects for the cursor movement. We have set the required HSV range of the yellow color. The detected object will be enclosed by the blue color bounding rectangle box. But in order to detect the object, it should be placed within the viewing area of the webcam.

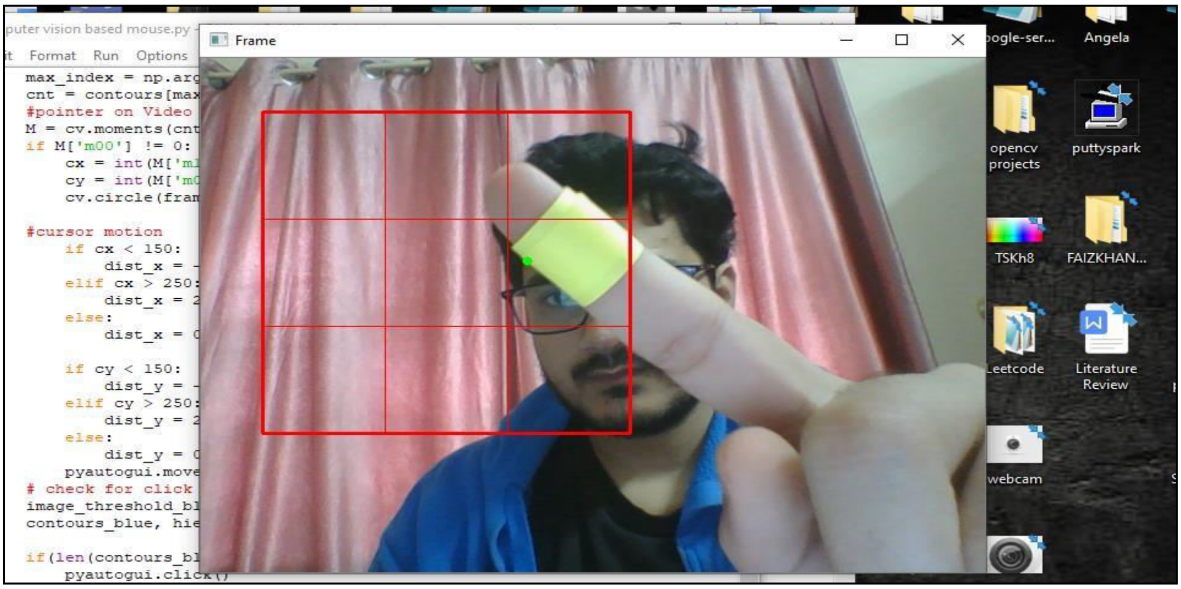
C. Mouse cursor movement and click-events

Instead of Matlab, we have used PyAutoGUI to move the cursor based on marker location in the grid and also click based on the color of our choice(in this case, green for left click, orange for right-click and blue for double click) of the marker. The mouse functions of PyAutoGUI make use of x and y coordinates of the pixels on the screen, just like any image. So, the resolution of a screen which determines how many pixels wide and tall our screen is a significant parameter.

1) pyautogui.size(): This function returns a 2 integer tuple of the screen"s width and height in pixels. Depending on ours screen resolution, the return value may be different. The width and height of the screen can be stored from the above function in variables like width and height, to access them in the programs.

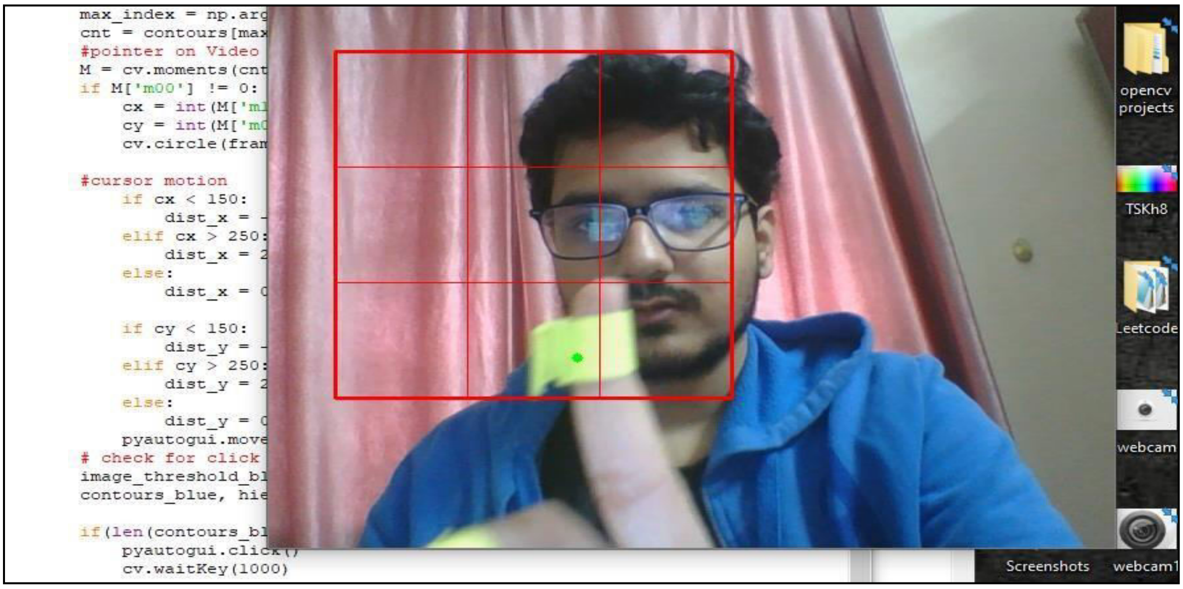
2) pyautogui.moveRel(x, y): This function will instantly move the mouse cursor from its current position to a specified position on the screen. We have made use of this function to move our mouse cursor based on the yellow color. Integer values of the x and y coordinates make up the function"s first and second arguments respectively. Once the coordinates have been determined, the coordinates are sent to the cursor. The cursor places itself in the required position with these coordinates. As
the user moves the colored object across the view of the ROI, the mouse cursor also moves across the screen.

3) pyautogui.click() : The control action of the mouse is performed by this function. The sensor has to detect the specified color for performing the mouse event action. We have set the blue-color for double-click, green-color for left-click and orange-color for right-click.



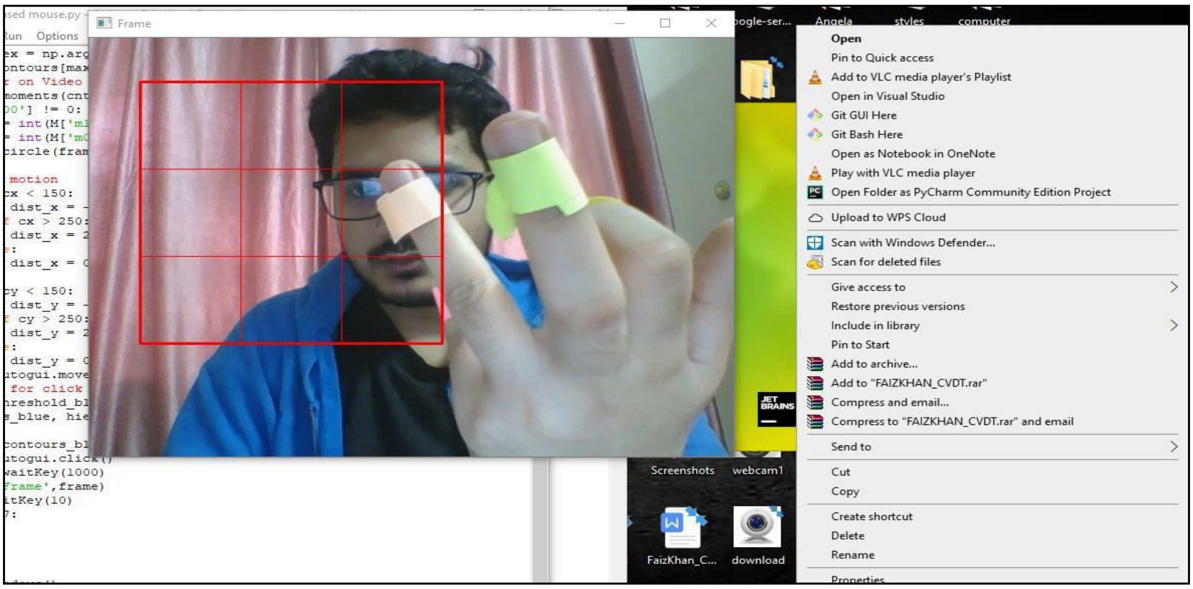
```
max_index = np.arg
cnt = contours[max
#pointer on Video
M = cv.moments(cnt
if M['m00'] != 0:
    cx = int(M['m1
    cy = int(M['m0
    cv.circle(fram

#cursor motion
    if cx < 150:
        dist_x = -
    elif cx > 250:
        dist_x = 2
    else:
        dist_x = 0

    if cy < 150:
        dist_y = -
    elif cy > 250:
        dist_y = 2
    else:
        dist_y = 0
    pyautogui.move
# check for click
image_threshold_bl
contours_blue, hie

if(len(contours_bl
    pyautogui.click()
```

Fig 9 Cursor Movement



```
max_index = np.arg
cnt = contours[max
#pointer on Video
M = cv.moments(cnt
if M['m00'] != 0:
    cx = int(M['m1
    cy = int(M['m0
    cv.circle(fram

#cursor motion
    if cx < 150:
        dist_x = -
    elif cx > 250:
        dist_x = 2
    else:
        dist_x = 0

    if cy < 150:
        dist_y = -
    elif cy > 250:
        dist_y = 2
    else:
        dist_y = 0
    pyautogui.move
# check for click
image_threshold_bl
contours_blue, hie

if(len(contours_bl
    pyautogui.click()
    cv.waitKey(1000)
```

Fig 10 Left Click Event



```
ex = np.arg
ontours[max
r on Video
moments(cnt
00'] != 0:
= int(M['m1
= int(M['m0
circle(fram

 motion
cx < 150:
 dist_x = -
f cx > 250:
 dist_x = 2
e:
 dist_x = 0

cy < 150:
 dist_y = -
f cy > 250:
 dist_y = 2
e:
 dist_y = 0
utogui.move
 for click
hreshold_bl
s_blue, hie

contours_bl
utogui.click()
waitKey(1000)
Frame',frame)
itKey(10)
7:
```

Fig 11 Right Click Event

VI. CONCLUSION

The proposed system architecture will completely change the way people would use the computer system. This project eliminated the need for a mouse or any physical device for cursor control.

The use of object detection, marker motion tracking and PyAutoGUI for the implementation of our proposed work proved to be successful and the movement of the mouse and click events is achieved with the high precision accuracy. This also led to better Humans - Computer Interaction(HCL). The proposed work has many advantages. This technology can be useful for those patients who cannot control their limbs [5]. They can just use any colored objects from their fingertips. It also has wide applications in the modern Gaming, Augmented Reality and Computer Graphics. In the case of computer graphics and gaming, this technology has been applied in modern gaming consoles to create interactive games where a person‟s motions are tracked and interpreted as commands[4]. The main aim is to reduce the use of any physical device and make use of only web cam which is readily available with the laptop. No additional costs are required.

Though this project has many advantages, it does suffer from some drawbacks. If the background image clashed with the specified image, then it can give an erroneous response and may not work properly [3]. So it is advisable to use this technology where the operating background light and does not mix with the color in hand. System may run slower on certain computers of low resolution and computational capabilities. If the camera is high on resolution., then the system might run slow. But problem can be solved by reducing the resolution of the image.

REFERENCES

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

[2] K. Pendke, P.Khuje, S. Narnaware, S. Thool, S. Nimje, “Computer Cursor Control Mechanism by Using Hand Gesture Recognition”, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.3, March- 2015,pg.293-300.

[3] A. Banerjee, A. Ghosh, K. Bharadwaj, H. Saikia, “Mouse Control Using a Web Camera Based on Colour Detection”, International Journal of Computer Trends and Technology (IJCTT) – volume 9 number 1– Mar 2014.

[4] A. Yadav, A. Pandey, A. Singh, A. Kumar, “Computer Mouse Implementation Using Object Detection and Image Processing ”, International Journal of Computer Applications. (IJCA), Volume 69-Number 21, 2013.

[5] D. Verma, et al., “Vision Based Computer Mouse Controlling Using Hand Gestures”, International Journal of Engineering Sciences & Research Technology.(IJESRT), 7(6): June, 2018.

[6] Hojoon Park, “A Method for Controlling the Mouse Movement using a Real Time Camera”, Brown University, Providence,RI, USA, Department of computer science, 2013.

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

[8] Kamran Niyazi, Vikram Kumar, Swapnil Mahe, Swapnil Vyawahare, “Mouse Simulation Using Two Coloured Tapes”, Department of Computer Science, University of Pune, India, International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.2, March 2012.