Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019)
IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-8

Virtual Mouse Implementation using Open CV

[1] Kollipara Sai Varun, [2] I. Puneeth, [3] Dr. T. Prem Jacob

[1][2] – Computer Science Department, UG Student, Sathyabama University, Chennai, India.
saivarunprabas@gmail.com puneeth2011@gmail.com

[3] – Computer Science Department, Professor, Sathyabama University, Chennai, India.
premjac@yahoo.com

***Abstract*—Hand Gesture Recognition plays a key role in human-computer interactions. As we can see that there are so many new Technological advancements happening such as biometric authentication which we can see frequently in our smart phones, similarly hand gesture recognition is a modern way of human-computer interaction i.e., we can control our system by showing our hands in front of webcam and hand gesture recognition can be useful for all kinds of people. Based upon this idea this paper is presented. This paper provides a detailed explanation to the algorithms and methodologies for the color detection and virtual mouse.**

***Index Terms*—Computer Vision, Open CV, Deep Learning, Image Processing.**

I. Introduction

A Computer Mouse is an input device that helps to point and to interact with whatever that is being pointed. There are so many types of mouse in the current trend, there's the mechanical mouse that consists of a single rubber ball which can rotate in any direction and the movement of the pointer is determined by the motion of that rubber ball. Later the mechanical mouse is replaced by the Optical Mouse [1].

Optical Mouse consists of a led sensor to detect the movement of the pointer. Years Later the laser mouse was introduced to improve the accuracy and to overcome the drawbacks of the Optical Mouse. Later as the Technology has been increased drastically wireless mouse was introduced so as to enable hassle free movement of the mouse and to improve the accuracy.

No Matter how much the accuracy of the mouse increases but there will always be limitations of the mouse as the mouse is a hardware input device and there can be some problems like mouse click not functioning properly ad etc., as the mouse is a hardware device like any other physical object even the mouse will have a durability time within which is functional and after its durability time we have to change the mouse.

As the technology increase everything becomes virtualized such as speech recognition. Speech Recognition [4] is used for recognition and translation of the spoken language into text. Thus, Speech Recognition can replace keyboards in the future, Similarly Eye Tracking [5] which is used to control the mouse pointer with the help of our eye. Eye Tracking can replace mouse in the future.

Gestures can be in any form like hand image or pixel image or any human given pose that require less computational difficulty or power for making the devices required for the recognitions to make work. Different techniques are being proposed by the companies for gaining necessary information/data for recognition handmade gestures recognition models [6] [7]. Some models work with special devices such as data glove devices and color caps to develop a complex information about gesture provided by the user/human.

In a Similar way the virtual mouse that we will discuss in this paper is made up of OpenCV, pyautogui. OpenCV is a python library for computer vision which is used for capturing images from webcam. Pyautogui is a python library that is used for specifying keyboard and mouse operations.

Previously when we want to operate and work with system's it's been a really different thing. So once the technology is being developed the things like separate applications and soft wares are created and developed for easy use of system. When the technology had introduced technologies like artificial Intelligence [11] and machine learning [12] they have become booming technologies that help us easily to operate and give commands to systems without any human interactions. So here the model is also a part of these technologies which can help us work with our systems with ease.

The techniques are developed in such a way even machine / System have the intelligence [2] that can identify and vary different thing which are easy for human now became easy for systems to understand the same things with a new way of learning.

Virtual Mouse using Hand gesture recognition [3] allows users to control mouse with the help of hand gestures and system's webcam is used for tracking hand gestures. Computer vision techniques are used for gesture recognition. OpenCV consists of a package called video capture which is used to capture data from a live video.

II. METHOD

A. Open CV

Open CV (Open Source Computer Vision Library) [13] is an open library of python that is mainly aimed at real-time computer-vision. It is also available in C++ and Java. It is an open source machine learning software library. It makes use of Numpy, which is a python library that is used for implementing multi-dimensional arrays and matrices along with high-level mathematical operations on these arrays. OpenCV is mainly used to capture data from a live video hence it is mainly focuses on image processing and video capture. In this paper OpenCV is focused mainly on video capture. OpenCV is also used for applications such as face detection, OCR, Vision-guided robotics surgery, 3D human organ reconstruction, QR code Scanner etc., Using OpenCV we can perform detection of specific objects such as eyes, faces etc., we can also analyze videos such as estimating the motion in the video or subtracting the background from the video, and tracking objects [8] in it.

OpenCV covers the basic data structures such as Scalar, point etc. that are used for building OpenCV applications. OpenCV library is imported into the python using the code 'import cv2'.

B. Anaconda

Anaconda is an open source platform for python and R programming Language which is used specifically for data science, Machine Learning, Data Processing, Predictive Analysis, etc., Anaconda consists of desktop GUI application called Anaconda Navigator and conda prompt which is a command prompt for Anaconda. Anaconda Navigator is used to launch applications and manage conda packages without using command-line commands. Anaconda Prompt is used for launching applications and manage conda packages with the help of command-line commands. The default applications that are available in Anaconda are Jupyter Lab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, and Visual Studio.

Anaconda also has a cloud called Anaconda Cloud. It is a package management service provided by Anaconda where we can access, store and share private and public notebooks, environments and various conda and PyPI packages.

Anaconda Cloud hosts various python packages, environments, notebooks for a variety of applications. With the help of this cloud we can also install python packages using pip or conda.

For operating the models we are using a system that is compatible with GPU's and high powered systems, Since the data which the models are working on requires high processing power that helps for fast and easy usage of models with users.

III. SYSTEM IMPLEMENTATION

The system working conditions are based on Anaconda Environment interface design, with OpenCV, wx, Numpy libraries and some of the sub packages of these libraries. Camera resolution is 1920*1080 and with fps of 40 (Default System Camera).

While the device is ON and when the model is run. The model will open a tab of the camera which takes input from the user. The model is designed for the recognition and further working is done by the commands given to the system and how the user wants the gestures to make recognize. At the same time, the mouse pointer movement will be captured and made operate without any human interaction. The model working and implementation is shown in the figures.

```
while True:
    ret, img=cam.read()
    img=cv2.resize(img,(340,220))

    #convert BGR to HSV
    imgHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    # create the Mask
    mask=cv2.inRange(imgHSV,lowerBound,upperBound)
    #morphology
    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_NONE)

    cv2.drawContours(img,conts,-1,(255,0,0),3)
    for i in range(len(conts)):
        x,y,w,h=cv2.boundingRect(conts[i])
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255), 2)
        cv2.cv.PutText(cv2.cv.fromarray(img), str(i+1),(x,y+h),font,(0,255,255))
    cv2.imshow("maskClose",maskClose)
    cv2.imshow("maskOpen",maskOpen)
    cv2.imshow("mask",mask)
    cv2.imshow("cam",img)
    cv2.waitKey(10)
```

Fig. 1. This is the part of the code that is used for the color detection and highlighting the wanted color.

A. Model Understanding

The main thing we need to identify are the applications the model is going to develop so the development of the mouse movement without using the system mouse, the model is developed using computer vision where the color variation is used for the mouse identification and movement [9]. We can see the fig 1 to identify the color variation / color detection that helps with the main part of our problem.

```
lowerBound=np.array([33,80,40])
upperBound=np.array([102,255,255])
```

Fig. 2. Limits of Green color in HSV color identification (Hue, Saturation, and Value).

```
mask=cv2.inRange(imgHSV,lowerBound,upperBound)
```

Fig. 3. For adding the limits to the mask, so that it can identify the green from the camera and highlights the input from the user.

The model creation depends mostly on the steps that are given in the Fig 2 and 3. There we are identifying the color required for the user by using the range of green by using the color values in RGB image and highlighting them from HSV image and converting them into Black and White and display it in the mask which is system understandable image representation [10].

After recognizing the highlighted color from the Fig 1, it's shown that formation of red boxes around the color.

```
elif(len(conts)==1):   #case for a single rectangle

    if pinchFlag==0:

        pinchFlag=1

        mouse.press(Button.left)    #reducing mutiple clicks

    x,y,w,h=cv2.boundingRect(conts[0])

    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

    cx=int(x+w/2)

    cy=int(y+h/2)
    #plotting a big circle with centre as rectangle's centre
    cv2.circle(img,(cx,cy),int((w+h)/4),(0,0,255),2)

    mouseLoc = mLocOld + ((cx, cy) - mLocOld) / DampingFactor

    mouse.position = (sx - (mouseLoc[0] * sx / camx), mouseLoc[1] * sy / camy)

    mLocOld = mouseLoc
cv2.imshow("cam",img)   #Cam OP

cv2.waitKey(5)
```

Fig. 4. Code for operating if there is only one figure with highlighted color.

```
if(len(conts)==2):  #case for two rectangles

    if pinchFlag==1:

        pinchFlag=0

        mouse.release(Button.left)  #reducing mutiple clicks

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

    x2,y2,w2,h2=cv2.boundingRect(conts[1])
    # Drawing contour rectangle for 1st box
    cv2.rectangle(img, (x1, y1), (x1 + w1, y1 + h1), (255, 0, 0), 2)
    # Drawing contour rectangle for 1st box
    cv2.rectangle(img, (x2, y2), (x2 + w2, y2 + h2), (255, 0, 0), 2)

    cx1, cy1 = int((x1 + w1 / 2)), int((y1 + h1 / 2))
    #Finding middle point of both contours
    cx2, cy2 = int((x2 + w2 / 2)), int((y2 + h2 / 2))

    cv2.line(img,(cx1,cy1),(cx2,cy2),(255,0,0),2)   #Drawing a line between two contours

    cx,cy=int((cx1+cx2)/2),int((cy1+cy2)/2)

    cv2.circle(img,(cx,cy),2,(0,0,255),2)   #plotting a center point between two lines

    mouseLoc=mLocOld+((cx,cy)-mLocOld)/DampingFactor

    mouse.position=(sx-(mouseLoc[0]*sx/camx), mouseLoc[1]*sy/camy)

    mLocOld=mouseLoc
```

Fig. 5. Code for operating if there are two figures with highlighted color.

From the above images we can clarify what and all are the functions and operations that are going on in the model. The detailed description is also presented.

Fig 4 shows the code when a single input is given and the operating that are performed. When a single figure is given there will be moving mouse which is depending on the figure which is not applicable for clicking / selection processes.

So the developed code is again given in the Fig 5 where the code is giving that a point is formed which is used for the movement and without adding / changing the code if we join the two fingers click can be formed by inversing the rectangles into circle that will identification for the system.

IV. Mistakes Made

The common places many people will be making mistakes are they try identifying the color which is a really difficult part that will lead to wrong identification / incorrect Identification. There are some other issues like people forget for checking the worst case scenarios where there are multiple identifications case where the model need to be developed in such a way it can be useful to multiple input's / multiple gestures for the inputs.

Most importantly the model which is developed needs to have a particular color detection if we lose that step or misplaced that we will be having problems. To overcome this we have multiple ways of showing the color variation / detection properly that's by using HSV / Color code, using these methods the system can be developed for a better usage without any problems.

V. Results

The model which we have developed is by using open CV and Numpy. The models which we are developing are color detection and mouse movement based on highlighted color which is given from the user for the movement of the mouse. Let's see how the code will be working which we have implemented for the color detection and the movement based on the rectangles which we have added for the color.

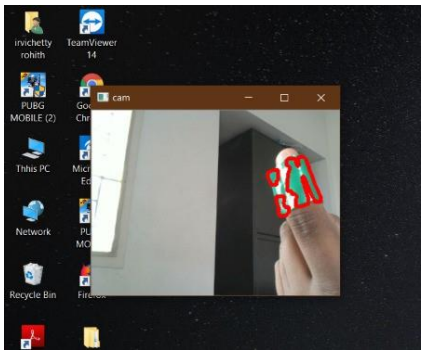

Fig. 4. This is the part of the code that is used for the color detection and highlighting the wanted color.

Here in the fig 4 we are seeing a single figure input provided to the camera feed, what we can observe is that the mask which is created is based on the color Green. This works based on the color detection code that is provided to distinguish the back ground with green. This is a sample representation with a single input from the user. Now let's see how it will work for the different ways of inputs.

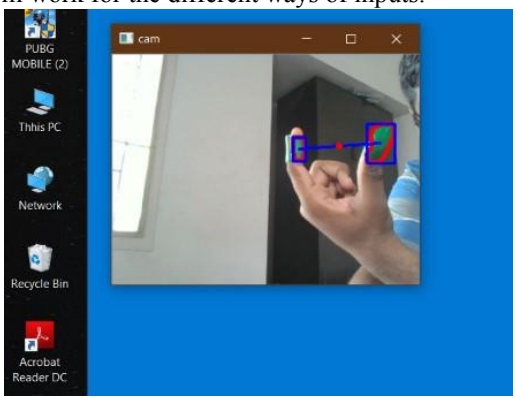

Fig. 5. This is the part of the code where it identifies the multiple inputs from the user that can help for the creation of the mouse pointer movement.

So, here in fig 5 we are seeing a two figure input where it is forming two rectangles and forming an average point from both the figures. That point will be acting like a mouse pointer. Once the point moves the mouse pointer in the runtime also moves along. So using this we can implement the movement of the mouse. The updating of the mouse pointers depends on the position of the green caps in the mask that is created for understanding the system.

The created mask is converted from RGB background to a black and white that will be used for the detection of the green objects that will help for the movement of the mouse.

This is a sample implementation that can be gathered from using this model – movement of apps on the desktop / clicking / Selection of multiple items. Let's see how it will be using the figure – 6 and 7.

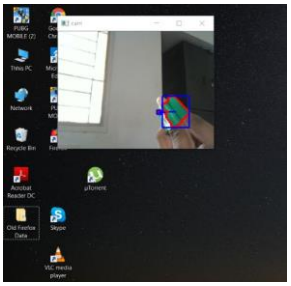
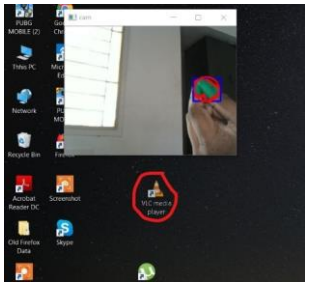

Fig. 6 & 7. In here the system implementation and working where we have selected an application like VLC player and tried moving using our mouse model.

Same way we can observe the things like clicking which is formed form combining both the figures. So the working of this is based on the circle that is formed combining two rectangles that is formed from the detection of the green color in front of the camera.

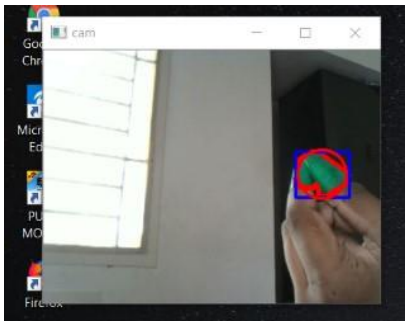

Fig. 8. The figure which that's given here shows the clicking of the image that will be helping for the clicking in case of click.

As the observation from fig 8 we have classified that click is formed and that can be used for the actions like selection / dragging files / many other functions. So this are the results that are obtained from the model development and execution.

VI. Further Development

The development of these techniques and models are really vast. The color detection model can be developed if we want to identify a particular color out of a colored photo. And the mouse movement can be developed in such a way it can act like a real mouse that will help us for using system without even touching the system's keyboard or mouse. The development can be in such a way it can be training on CNN's [14] that will help for a better performed model.

The Models can be developed in different ways by using some latest packages like 'pyauto GUI' [15] that will help us to give commands which will identify an input and perform some function on the system. So if any separate color is detected it can perform special function or if an input from user is detected it will open any specific folder with ease without performing any actions, a simple gesture can do the job.

VII. Conclusion

This model can conclude by using the topics of computer vision like open CV, it can form masks that can variate colors by using color variation techniques and also development of mouse movement by using certain packages like 'mouse' which will be used for the movement of mouse by using the coordinates that are linked to the detected color.

This can provide ease use of systems and many other applications. So the open CV is helping the users with different accessible forms of models that will make ease life.

References

[1] Guoli Wang, (2010). Optical Mouse Sensor-Based Laser Spot Tracking for HCI Input, Proceedings of the 2015 Chinese Intelligent Systems Conference: Volume 2, pp.329-340.

[2] Anna De Liddo, Ágnes Sándor, et.al, (2012). Contested Collective Intelligence: Rationale, Technologies, and a Human-Machine Annotation. Computer Supported Cooperative Work (CSCW) Volume 21, Issue 4–5, pp 417–448.

[3] Rashmi Adatkar, Ronak Joshi, et.al, (2017). Virtual Mouse, Imperial Journal of Interdisciplinary Research (IJIR), Vol-3, Issue-4.

[4] Arul. V. H, Dr. Ramalatha Marimuthu, (2014). A Study on Speech Recognition Technology, Journal of Computing Technologies, Volume 3 Issue 7, pp 2278 – 3814.

[5] Aniwat Juhong, T. Treebupachatsakul, et.al, (2018). Smart eye-tracking system. 2018 International Workshop on Advanced Image Technology (IWAIT).

[6] Guojen Wen, Zhiwei Tong, et.al, (2009), Man machine interaction in machining center. International workshop on intelligent systems and applications. pp 1-4.

[7] S.D. Bharkad, et.al. (2017). international conference on computing methodologies and communication, pp 1151-1155.

[8] Litong Fan, Zhongli Wang, Baigen Cail, et.al (2016). A survey on multiple object tracking algorithm. 2016 IEEE International Conference on Information and Automation (ICIA)

[9] Pritpal Singh, B.B.V.L. Deepak, Tanjot Sethi and Meta Dev Prasad Murthy (2015). Real-Time Object Detection and Tracking Using Color Feature and Motion. International Conference on Communication and Signal Processing.

[10] G. Saravanan, G. Yamuna, S. Nandhini (2016). Real time implementation of RGB to HSV/HSI/HSL and its reverse color space models. 2016 International Conference on Communication and Signal Processing (ICCSP).

[11] Artificial Intelligence [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence

[12] Machine Learning [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[13] Open CV [Online]. Available: https://opencv.org/

[14] Convolution Neural Networks [Online]. Available: http://www.wikipedia.org/wiki/Convolution_neural_networks

[15] Pyauto GUI [Online]. Available: https://pyautogui.readthedocs.io/en/latest/