ELECTROMETHEUS

for

ITSP 2016

Manimouse

Your hand made into mouse

Documentation

Shivam Kajale

- *+91-7738-928993*
- *shivamkajale0@gmail.com*

Manimouse:

You are watching a movie with your friends on a laptop and one of the guys gets a call. Ahh.. you have to get off your place to pause the movie. You are giving a presentation on a projector and need to switch between applications. You have to move across the whole stage to the podium to use your mouse. How better would it be if you could control your mouse from wherever you were? Well, we have a solution! **Manimouse**.

Manimouse is a mouse simulation system which performs all the functions performed by your mouse corresponding to your hand movements and gestures. Simply speaking, a camera captures your video and depending on your hand gestures, you can move the cursor and perform left click, right click, drag, select and scroll up and down. The predefined gestures make use of only three fingers marked by different colours.

Technical Overview:

Manimouse is essentially a program which applies image processing, retrieves necessary data and implements it to the mouse interface of the computer according to predefined notions. The code is written on Python. It uses of the cross platform image processing module OpenCV and implements the mouse actions using Python specific library PyAutoGUI. Thus, in addition to a webcam (which almost all laptops are already loaded with) a computer needs to be pre-equipped with the following packages for Manimouse to run:

- Python interpreter (preferably 2.7)
- OpenCV
- Numpy
- PyAutoGUI

Video captures by the webcam is processed and only the three coloured finger tips are extracted. Their centres are calculated using method of moments and depending upon their relative positions it is decided that what action is to be performed.

Technical Details:

The first thing that we do is convert the captured video into **HSV format**. Now the user gets to calibrate the colour ranges for three of his fingers individually. This is done by calling the calibrateColor() function thrice right at the beginning of the program (It is recommended that you keep the program open while reading this documentation for quickly referring to the code. For the GitHub link click here). The user has an option to use the default settings as well.

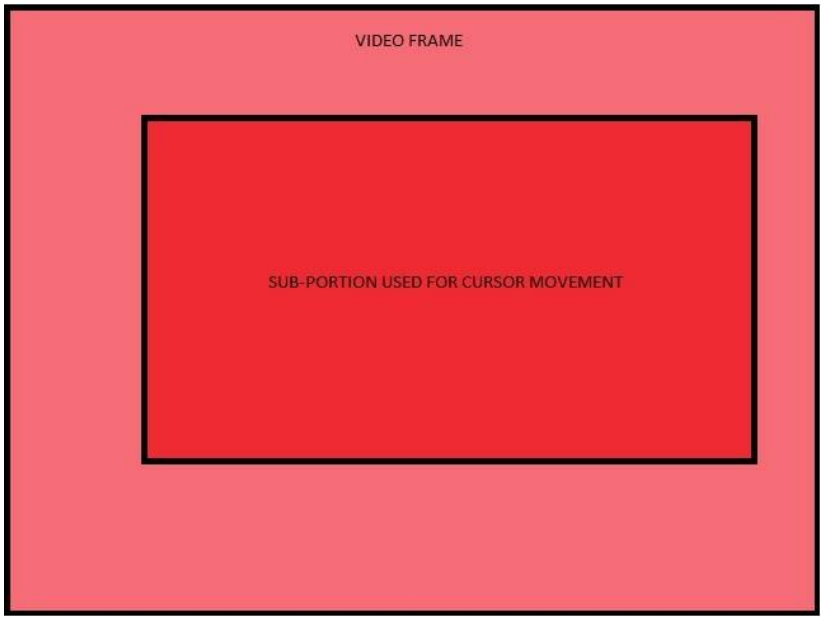
Depending on the calibrations, only the three fingertips are extracted from the video, one by one, using the **cv2.inRange()** function. In order to remove noise in the video feed, we apply a two-step morphism i.e. **erosion** and **dilation**. The noise filtered image referred to as mask in the program is then sent for locating the centres.

Location of each of the three centres involves:

- Finding contours in the mask relevant to that colour range.
- Discarding contours of irrelevant areas using area filters.
- Finding the largest contour amongst the remaining ones and applying method of moments to find its centre.

Then comes the step for defining position of cursor on the screen. The thumb, with yellow colour is responsible for position of the cursor. The following techniques have been used in this end:

- Generally the webcams we use captures video at a resolution of 640x480 pixels. Suppose this frame was linearly mapped to the 1920x1080 pixel display screen. If we have a right-handed user, he would find it uncomfortable to access the left edge of the screen as compared to the right edge. Also accessing the bottom portion of the screen would build stress at the wrist.
  We realised that instead of mapping the whole video frame to the screen, we could rather consider a rectangular sub portion more biased towards right (considering right-handed user) and upper parts of the frame in order to improve comfort. This sub portion which measures 480x270 pixels is then linearly mapped to the screen with a scaling factor of 4.



- Due to noise captured by the webcam and vibrations in the hand, the centres keep vibrating around a mean position. On scaling up, these vibrations create a lot of problem with the accuracy of cursor position. To reduce the shakiness in cursor, we make use of differential position allocation for the cursor. We compare the new centre with the previous position of the cursor. If difference is less than 5 pixels, it is usually due to noise. Thus the new cursor position is inclined more towards the previous one. However, a larger difference in previous position and new centre is considered as voluntary movement and the new cursor position is set close to the new centre. For details, go through the setCursorPosition() function in the code.

Now the three centres are sent for deciding what action needs to be performed depending on their relative positions. This is done in the chooseAction() function in the code. Depending upon its output, the performAction() function carries out either of the following using the PyAutoGUI library:

- free cursor movement
- left click
- right click
- drag/select
- scroll up
- scroll down

FREE MOVEMENT

LEFT CLICK

RIGHT CLICK

DRAG / SELECT

SCROLL DOWN

SCROLL UP

Relevant Links:

Github Repository: https://github.com/shivamkajale/Manimouse

Demonstration Videos:

- Beta Version: https://www.youtube.com/watch?v=oe00gOk0ctM
- Final Version: https://www.youtube.com/watch?v=82M1XkdizmU

References:

- Basic Image Processing: http://www.tutorialspoint.com/dip/index.htm
- OpenCV Python Tutorials: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
- PyAutoGUI: http://www.digitalbookocean.info/etext/12437-automate_the_boring_stuff_with_python_2015.pdf
- http://stackexchange.com/