**README**

* You can change the image path from the main file if you desire.
* Run the program.
* Continue by pressing *“enter”*.
* For question 1, write *“q1”*.
* There are two different answers for question 2. For brute force segmentation algorithm choose “q2.2”. The other’s results are not discussed in the report just the algorithm itself.
* Outputs can be found in txts named as *“segmentation1/2\_1/2\_2.txt”*.

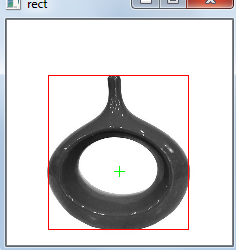
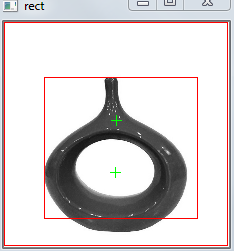
**Note:** Q2.2 algorithm runs slowly. You can see a percentage tracker while running. Be patient! You can always change the resize code in main as you like.

**EE 576 HW4**

**Methodology:**

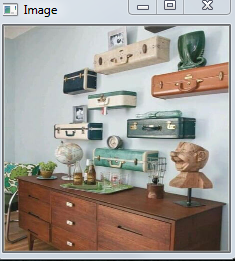
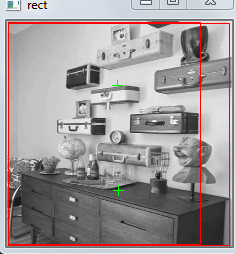
For the first question, I applied grayscale image thresholding to convert my image to a binary image. Then, I used “connectedComponentsWithStats” function of OpenCV to find necessary outputs. Showed tresholded segments both as an image and in the txt file.

For the second question, I used two different approaches. My deduction from the homework explanation was this. Label your image using colors without converting it to a binary image. Then using neighborhood method label them again to find the objects. Finally find the segments, threshold them, output to a txt and show the object with their center and bounding box. But the important part was while doing these tasks write your own labeling and connected component code because OpenCV works with binary images anyway. As a result, this was a vey challenging task with the main part and its required outputs. I did this approach but I also made a second approach which turns the image to a binary image for a color I coded and find segments of that color using OpenCV API. Much more easier and optimized. Repeats this for all colors and show the super-positioned output in the end. First approach coded as “*hsv\_cc2*” and required input to the program is “*q2.2*”.

*Original Image OpenCV Algorithm HSV Algorithm*

For the single object, the only difference between the two algorithms was the recognition of background which can be ignored with a simple code change. (%10 threshold)

**  **

*Original Image OpenCV Algorithm HSV Algorithm*

For the scene, background problem still applies and the absence of wall presence at the bottom of the image forces the background object center upwards, but the little presence of the wall makes bounding box bigger. The other brown object is affected by the presence of same color stuff on it thus center and bounding box are shifted .(%10 threshold)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Threshold %0.1 | Threshold %0.5 | Threshold %1 |
| Total N. of Segments | 182 | 182 | 182 |
| Thresholded Segment N. | 15 | 4 | 1 |
| Size Mean | 814 | 2760 | 9910 |
| Size Variance | 6346084 | 22723582 | 0 |

If threshold gets smaller mean gets small and variance gets bigger. Depending on the picture some key threshold changes can affect the statistics drastically and in a reverse way which can be seen between 0.5 and 0.1. Thus, threshold decision is important.

**References**

1. <http://isl.ee.boun.edu.tr/courses/ee576/lectures/sunum/segPres.pdf>
2. <https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html>
3. <https://docs.opencv.org/master/de/d01/samples_2cpp_2connected_components_8cpp-example.html>
4. EE576 Homework 2