CS484 – Introduction to Computer Vision

Homework-3

Ufuk Palpas

21702958

Part 1

In this section, I preferred to use the MATLAB implementation of superpixels to obtain superpixels and oversegmentation. MATLAB implementation of superpixels is the SLINC0 algorithm. Therefore, I preferred using SLIC0 and obtained superpixels by the SLIC0 algorithm.

As parameters, due to the fact that SLIC0 algorithm determines the best compactness parameter adaptively for each superpixel differently, I didn't select any compactness parameter. However, I determined the number of superpixels as 250. I tested very huge number and a very small number for this parameter as 600 and 50 then changed values by adding/subtracting 50 and I determined **250** as the ideal number.

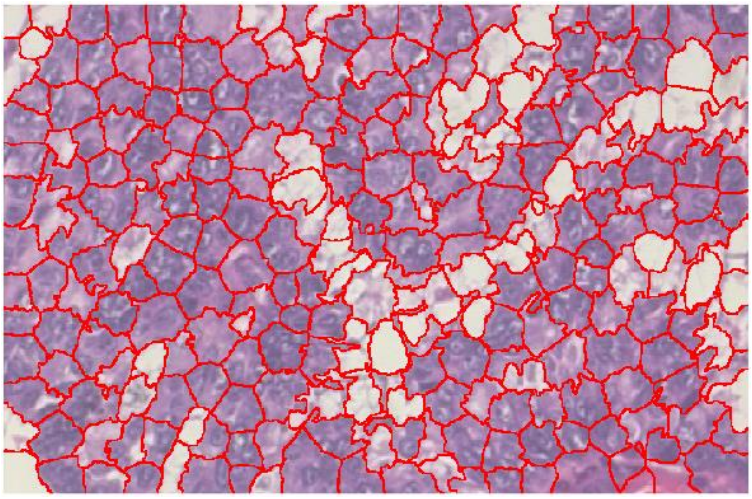

Figure 1: Superpixels of "01.png"

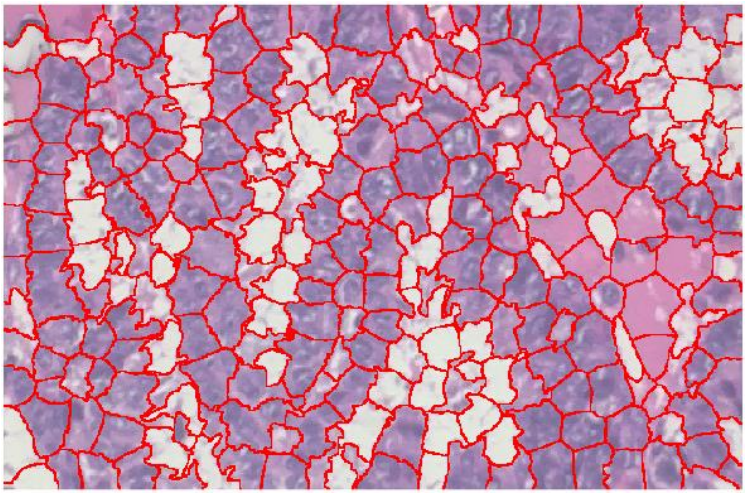

Figure 2: Superpixels of "02.png"

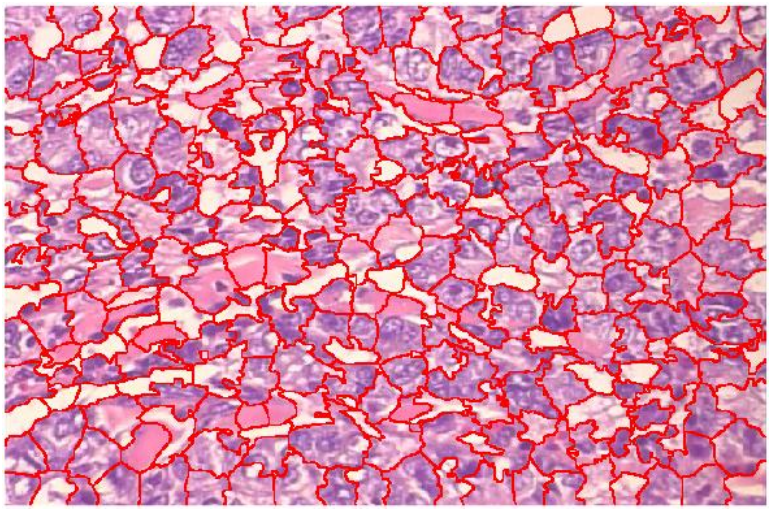

Figure 3: Superpixels of “03.png”

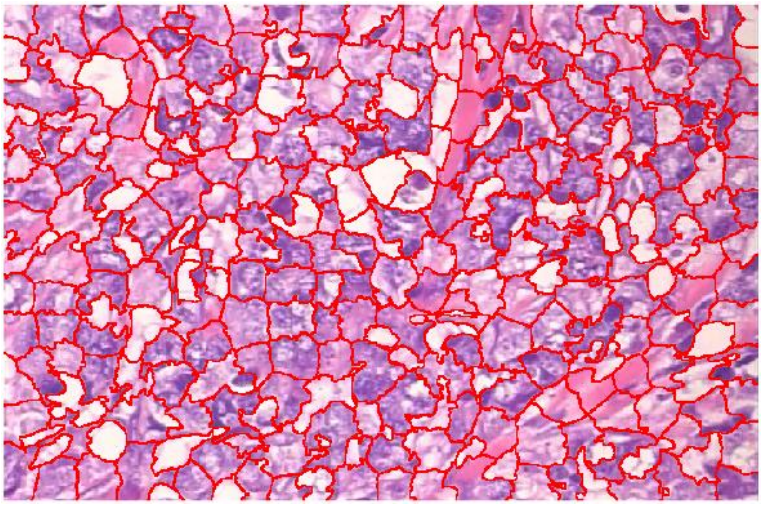

Figure 4: Superpixels of “04.png”

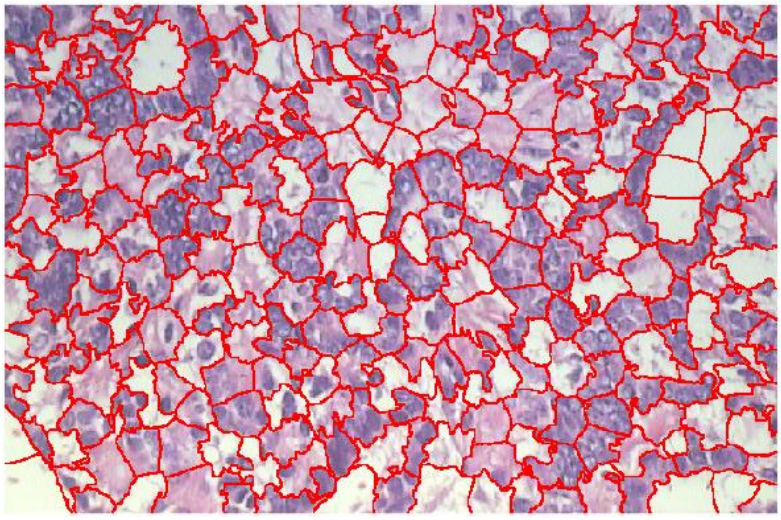

Figure 5: Superpixels of “05.png”

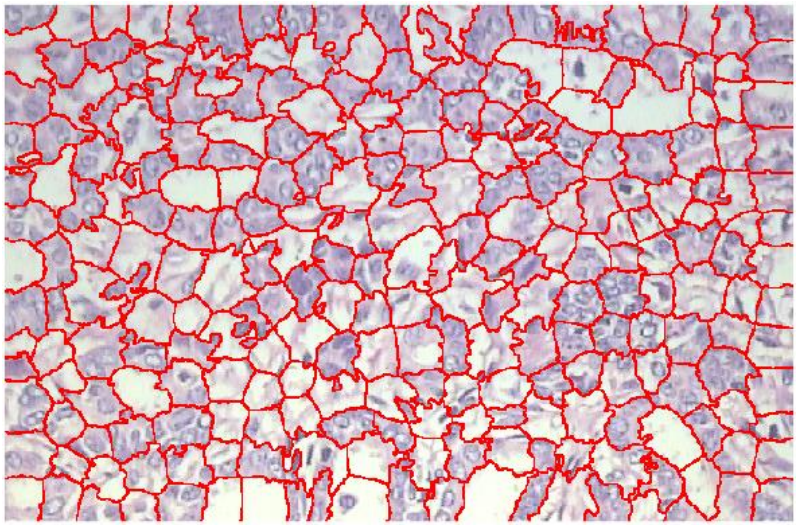

Figure 6: Superpixels of “06.png”

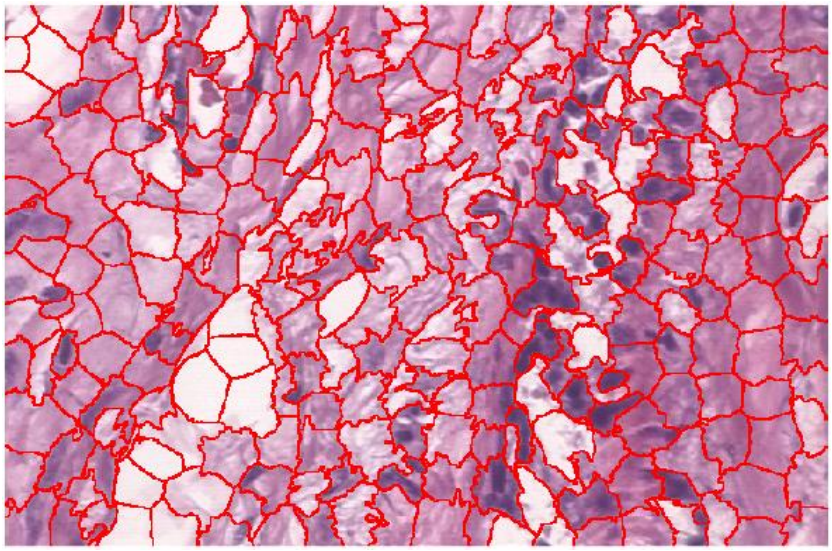

Figure 7: Superpixels of “07.png”

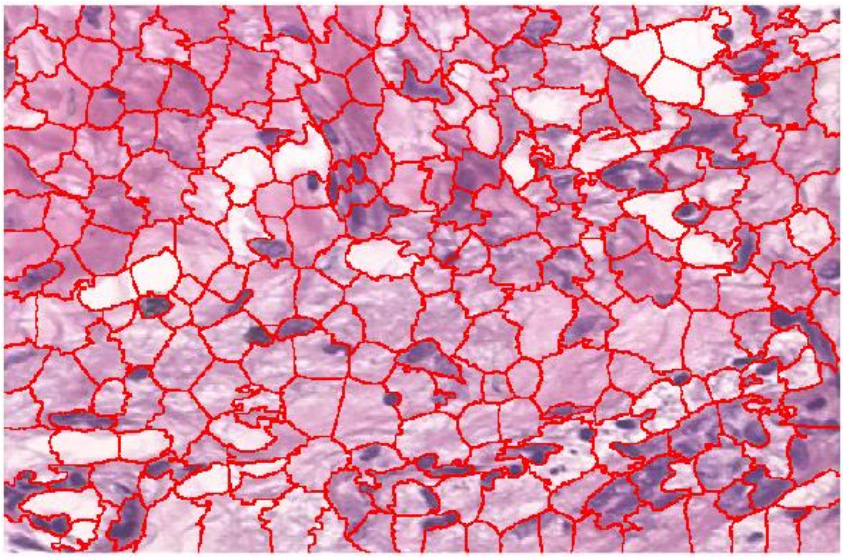

Figure 8: Superpixels of “08.png”

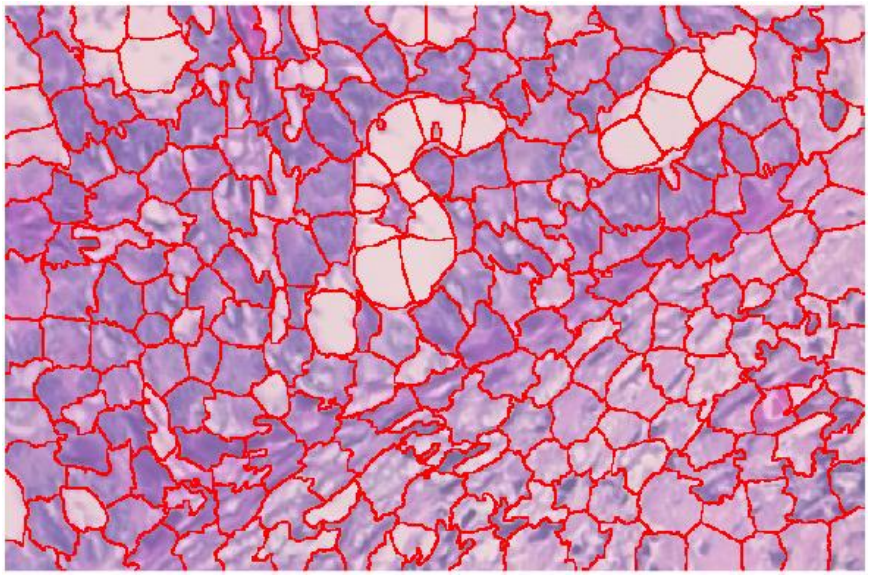

Figure 9: Superpixels of “09.png”

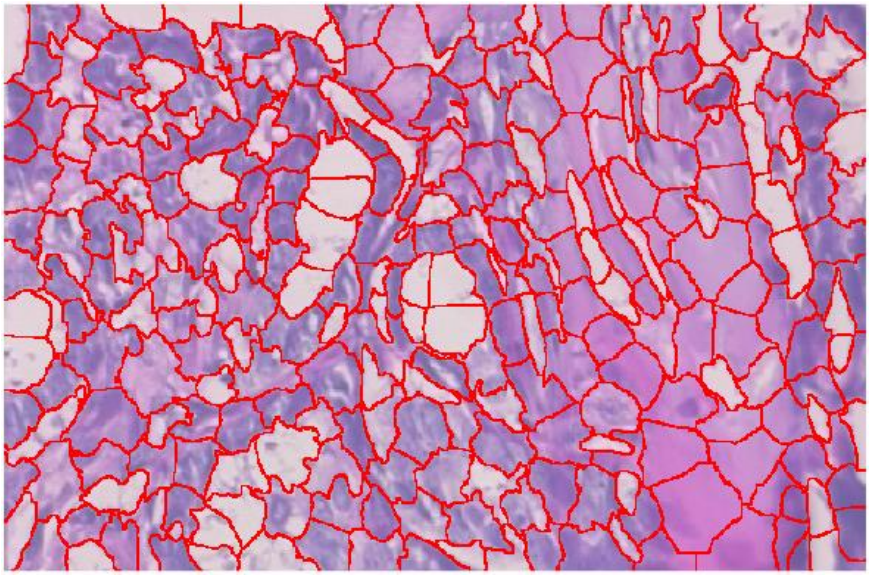

Figure 10: Superpixels of “10.png”

Finding Parameter (Number of Superpixels):

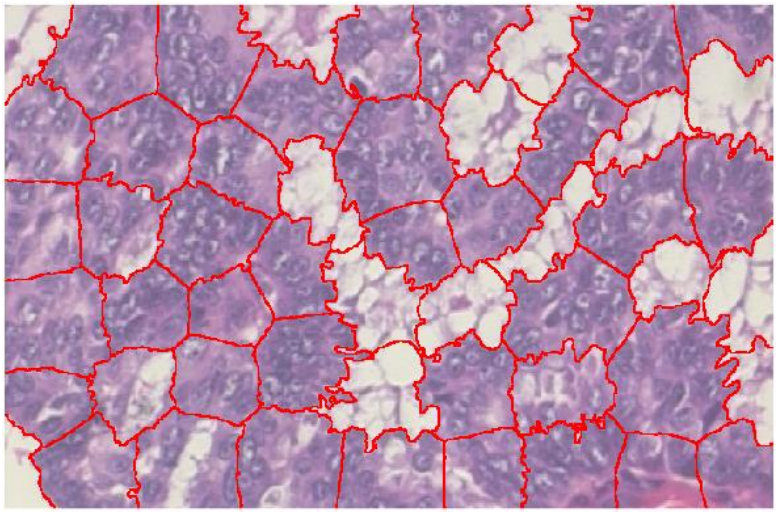

Figure 11: Superpixels of “01.png” with Number of Superpixels = 50

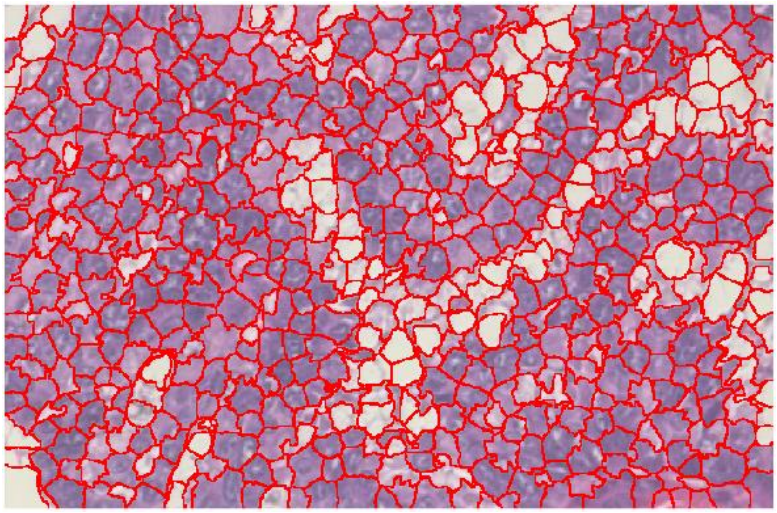

Figure 12: Superpixels of “01.png” with Number of Superpixels = 600

As it can be seen from the Figure 11 and Figure 12, when the number of superpixels increased accuracy of the segmentation increases too however, they become flatter and rectangle likely. However, when we decrease the number of superpixels such as 100 in figure, superpixels become huge which decreases the accuracy. I decided 250 as the number of superpixels because it is a goog middle point and gives good segmentation. The differences between 50, 250 and 600 can be seen above figures.

Part 2

I preferred MATLAB's image processing toolbox for Gabor filter implementation.

I preferred [10 15 20 25] as the wavelengths and [0 45 90 135] as the orientations in degrees and created the Gabor filter for each combination of these two. This gave me the 1x16 Gabor vector. I preferred the suggested default values as the other parameters.

Then I used the imgaborfilt function of MATLAB with the Gabor vectors and get the magnitude of Gabor responses and the phase of the response.

The Gabor filter responses for each image can be shown below (Gabor texture feature examples):

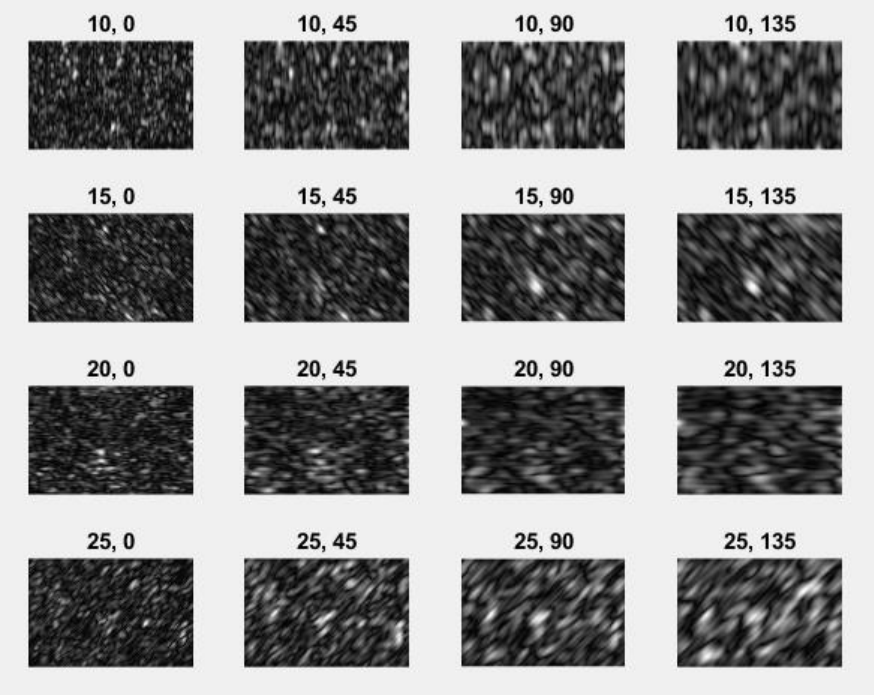

Figure 13: Gabor filter response of "01.png"

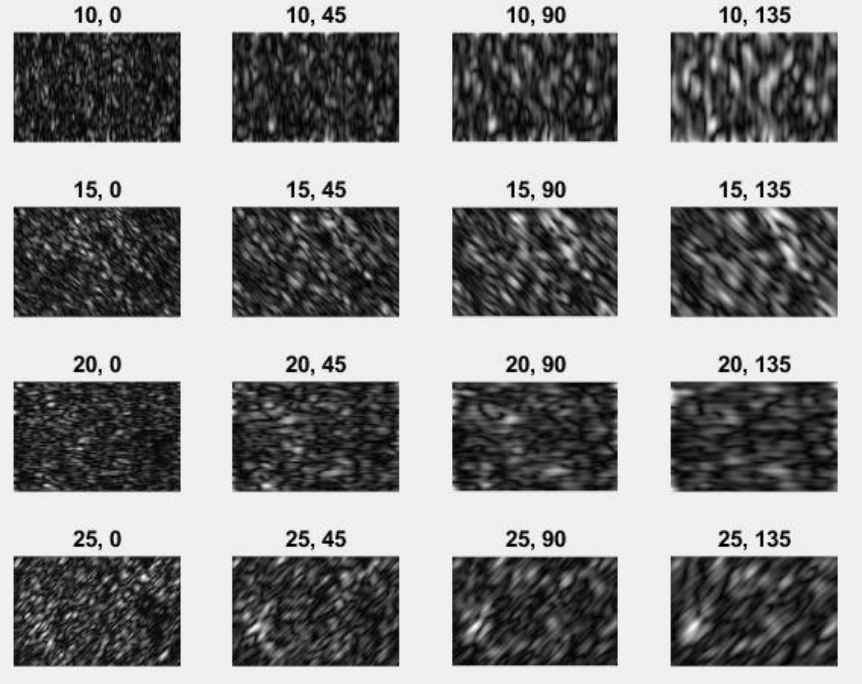

Figure 14: Gabor filter response of “02.png”

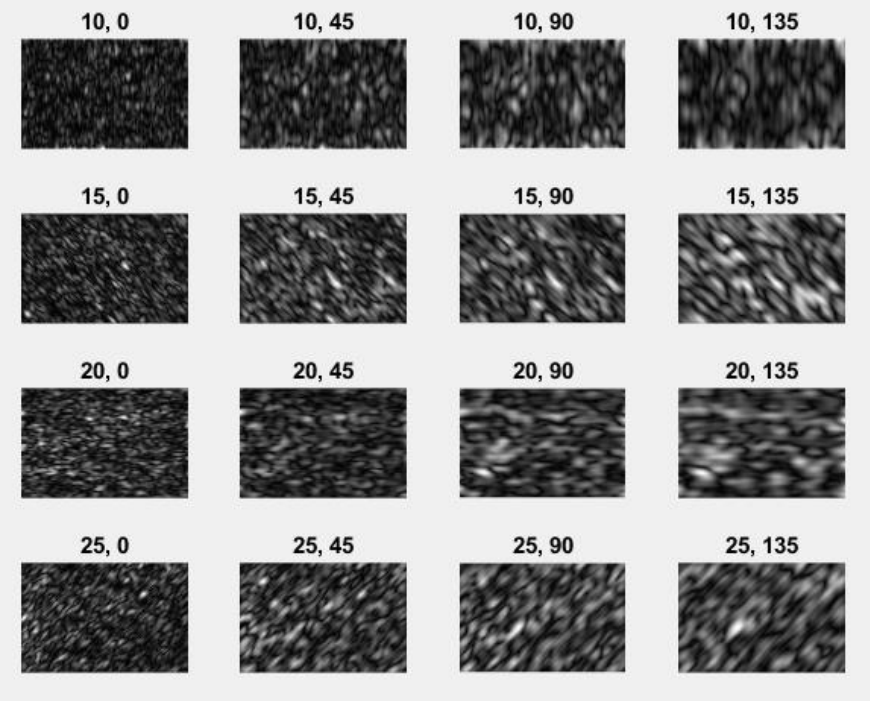

Figure 15: Gabor filter response of “03.png”

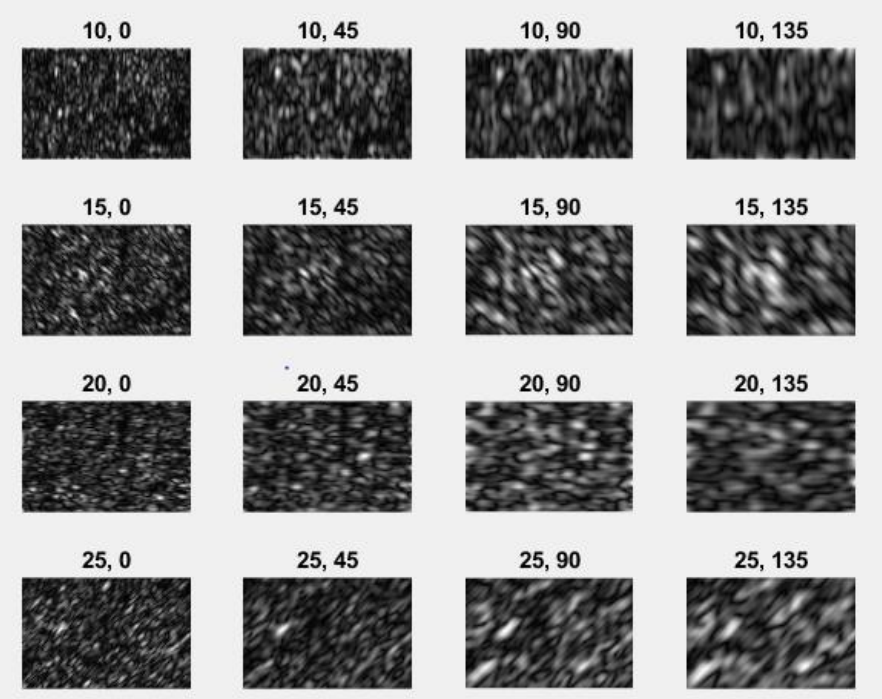

Figure 16: Gabor filter response of “04.png”

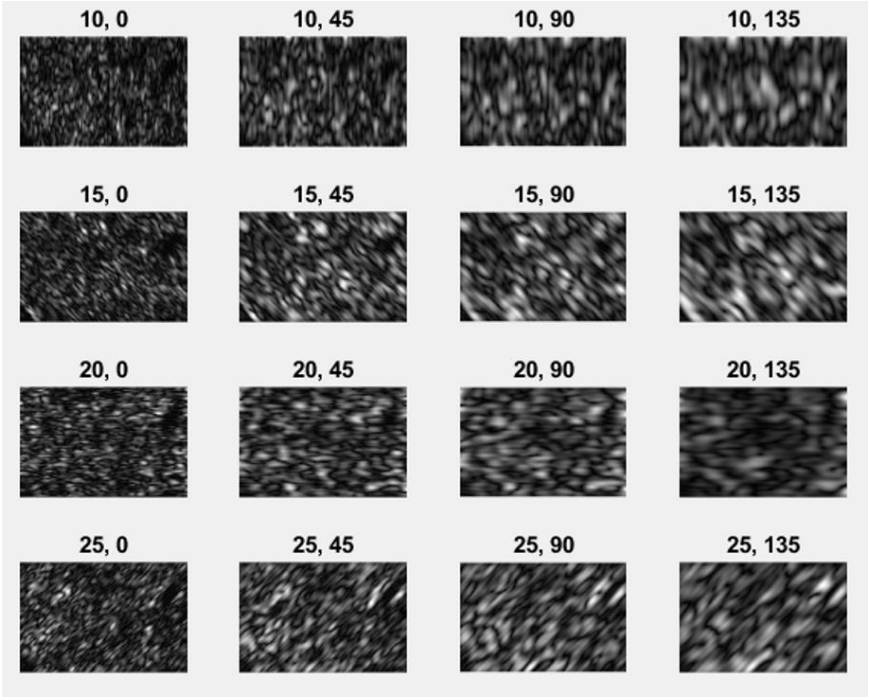

Figure 17: Gabor filter response of “05.png”

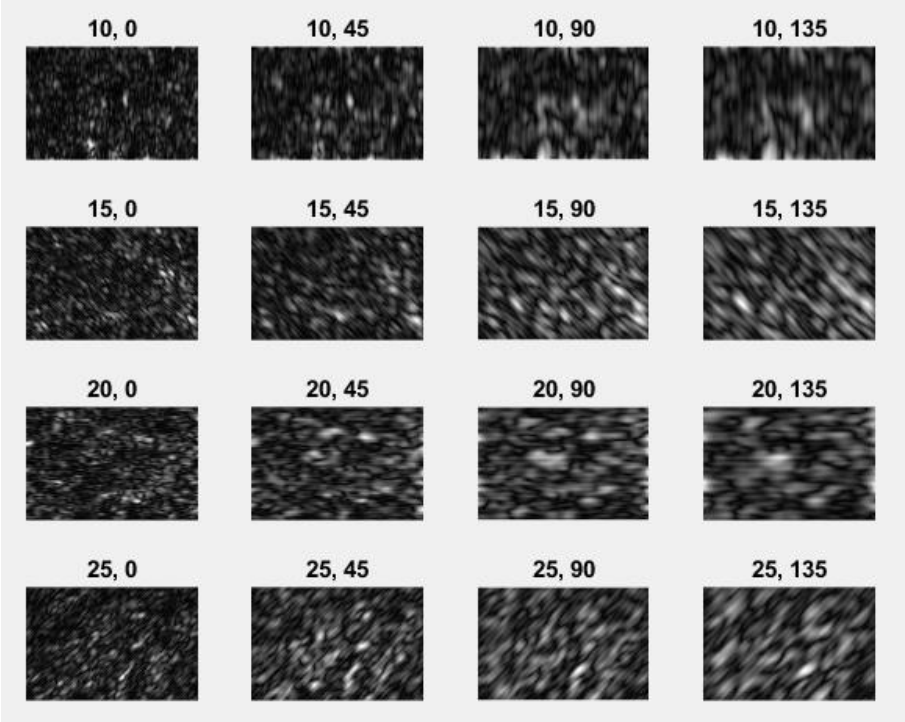

Figure 18: Gabor filter response of “06.png”

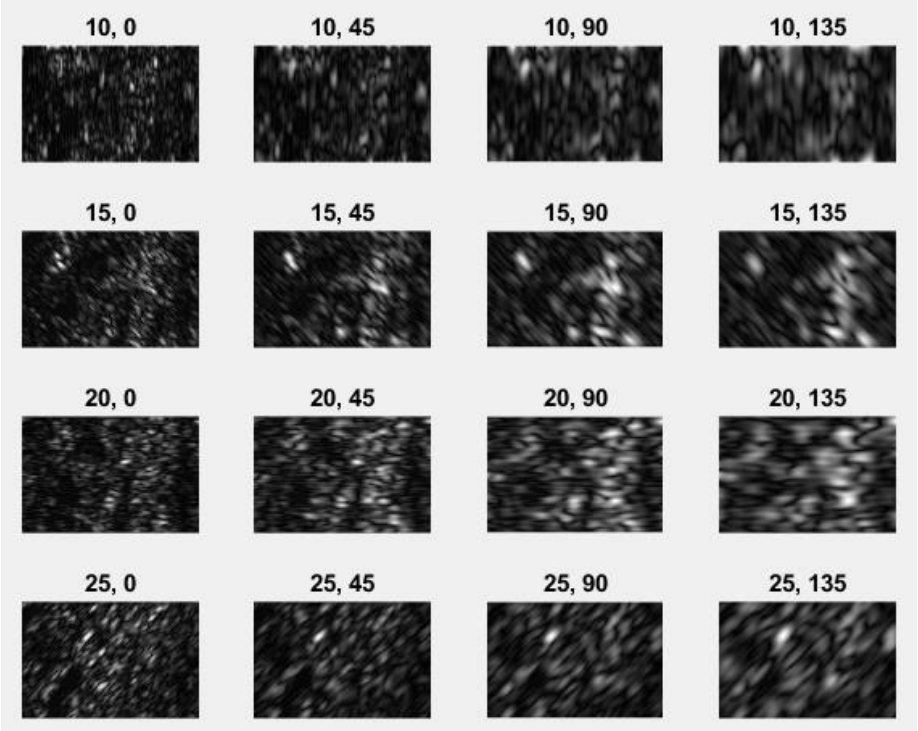

Figure 19: Gabor filter response of “07.png”

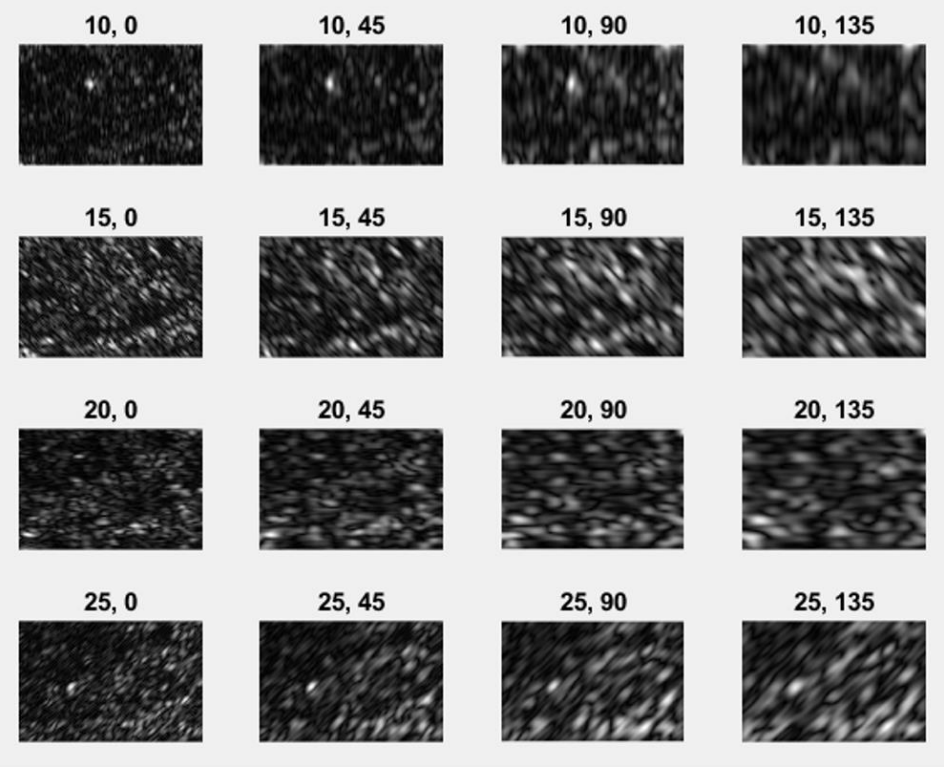

Figure 20: Gabor filter response of “08.png”

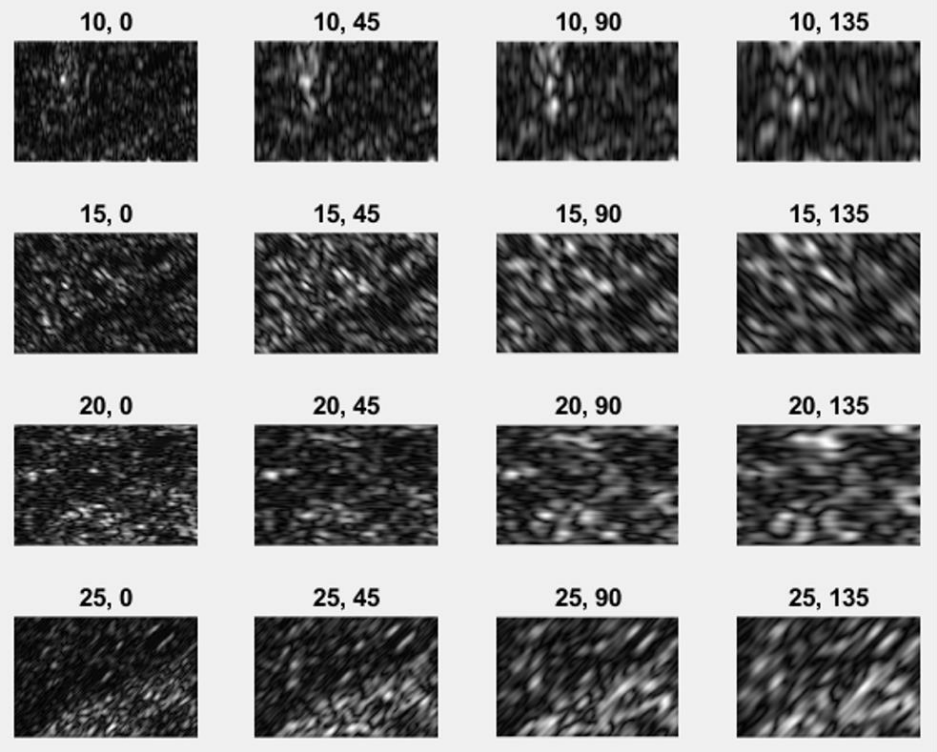

Figure 21: Gabor filter response of “09.png”

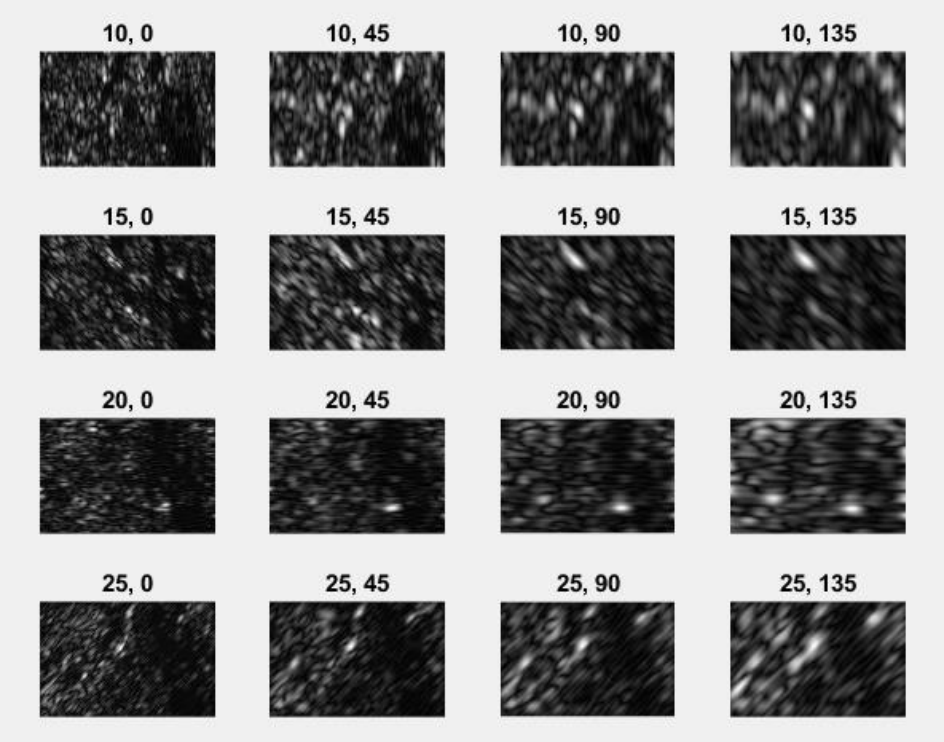

Figure 22: Gabor filter response of “10.png”

Part 3

I preferred k-means clustering in this part. And the images with pseudo color can be seen below. To show result better, both the false color (right) and pseudo color(left) versions are added to each figure. Clustering and segmentation results can be shown below for part 3.

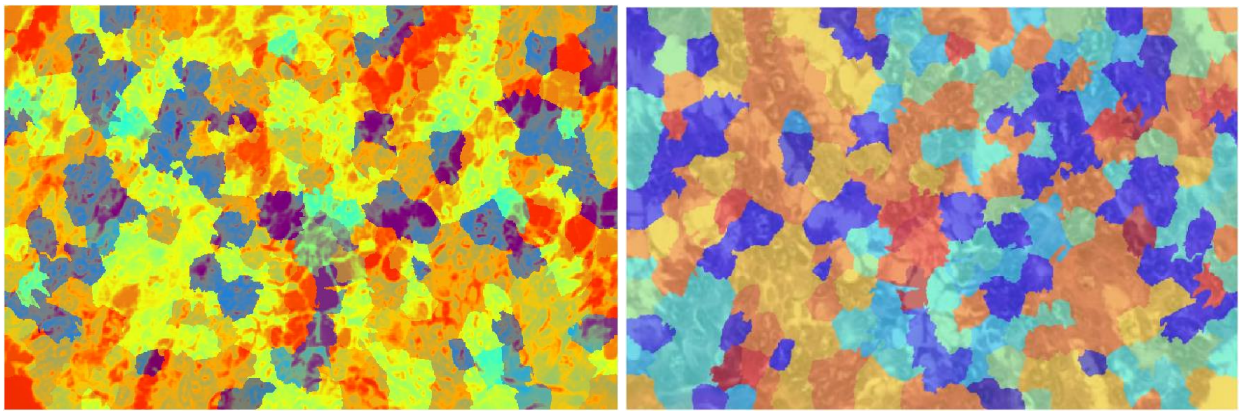

Figure 23: Outputs of “01.png”

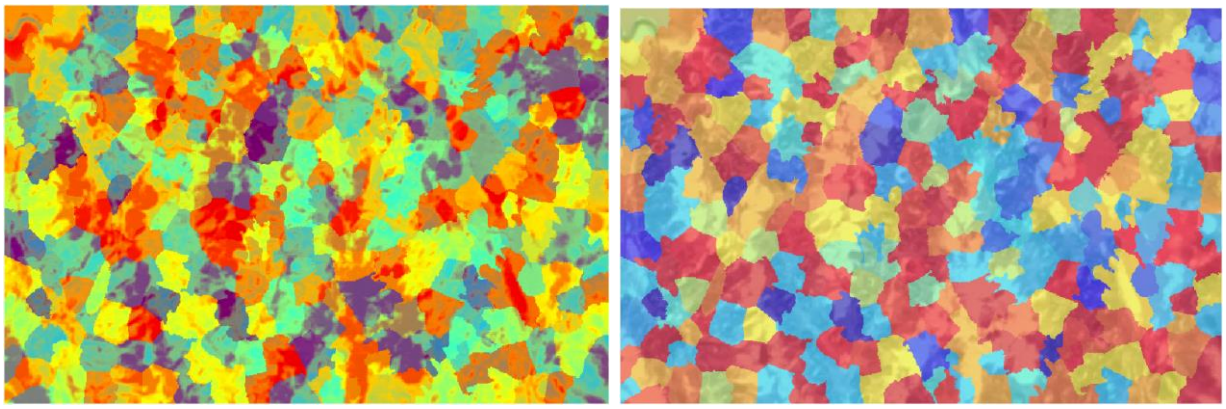

Figure 24: Outputs of “02.png”

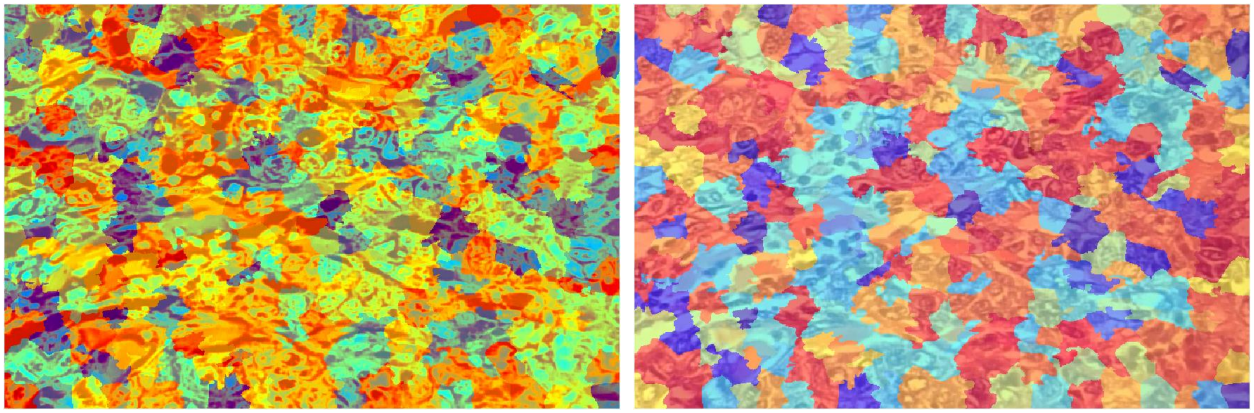

Figure 25: Outputs of “03.png”

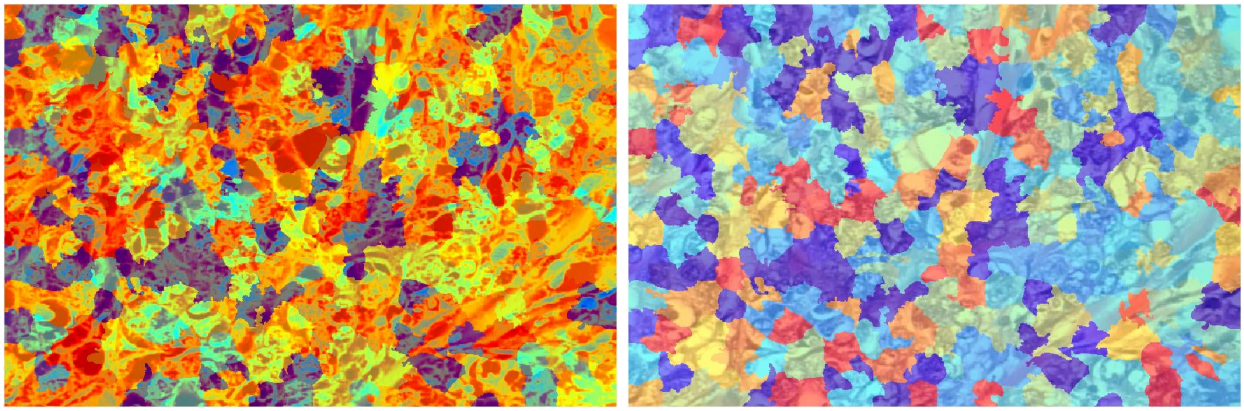

Figure 26: Outputs of “04.png”

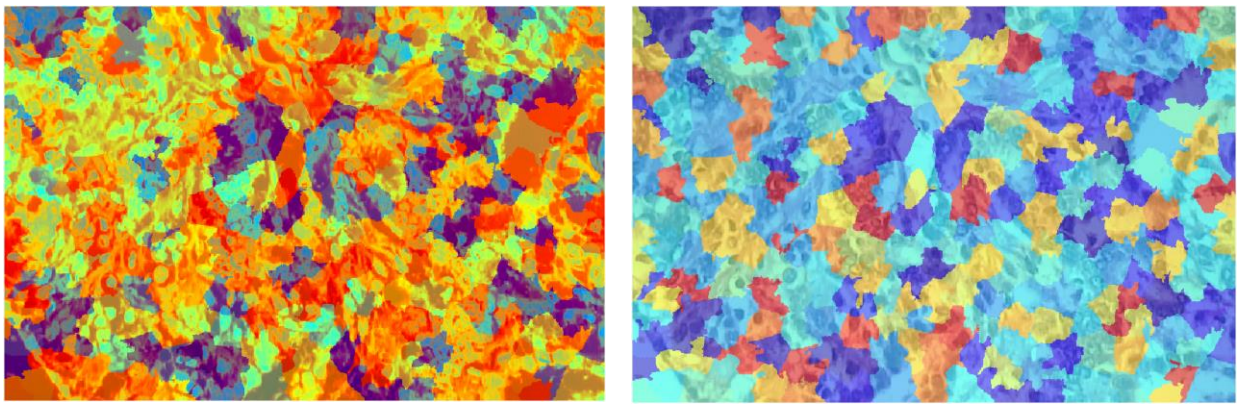

Figure 27: Outputs of “05.png”

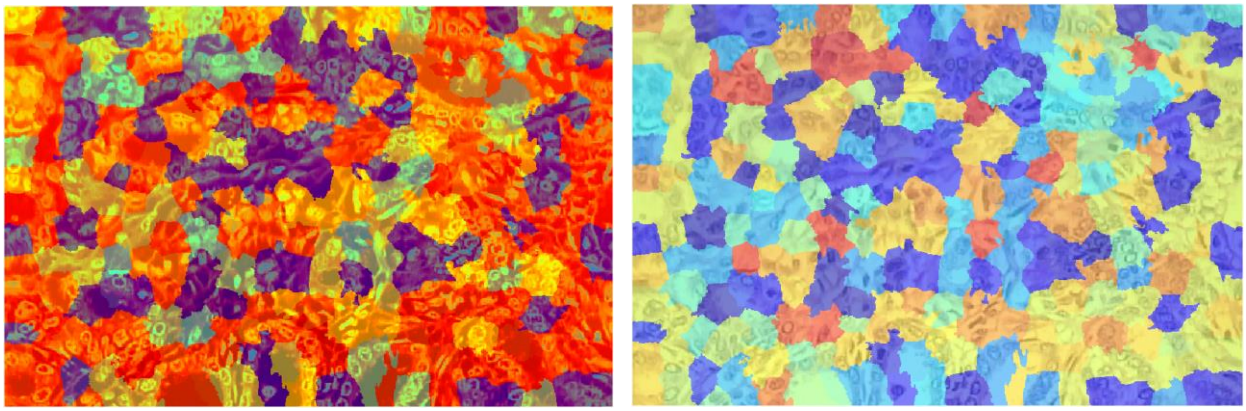

Figure 28: Outputs of “06.png”

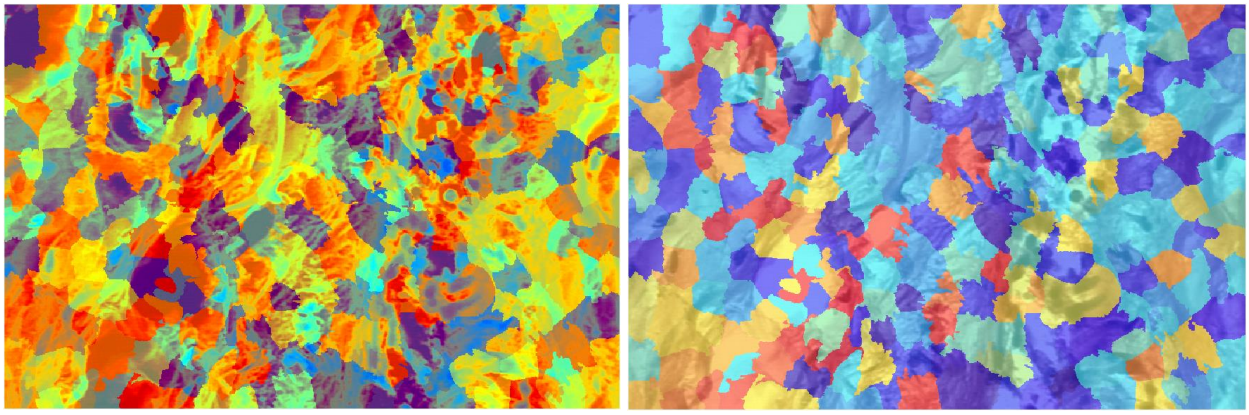

Figure 29: Outputs of “07.png”

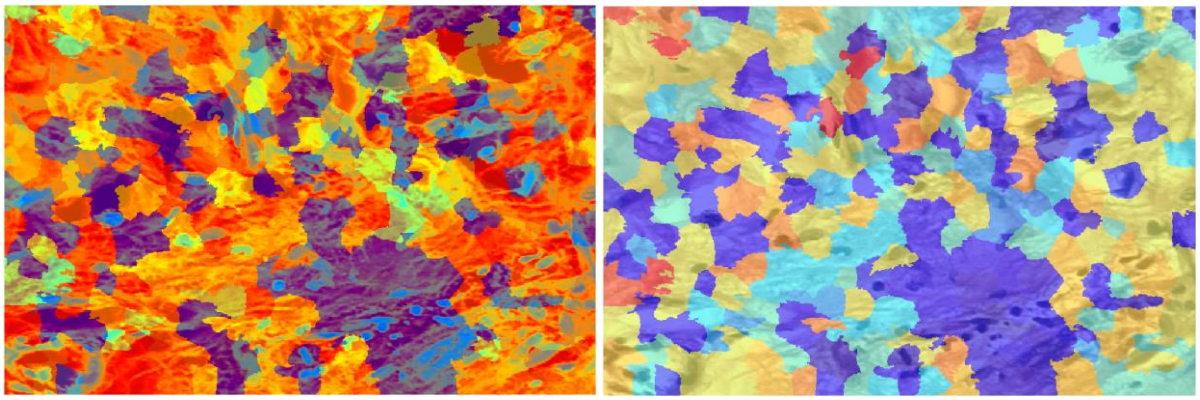

Figure 30: Outputs of “08.png”

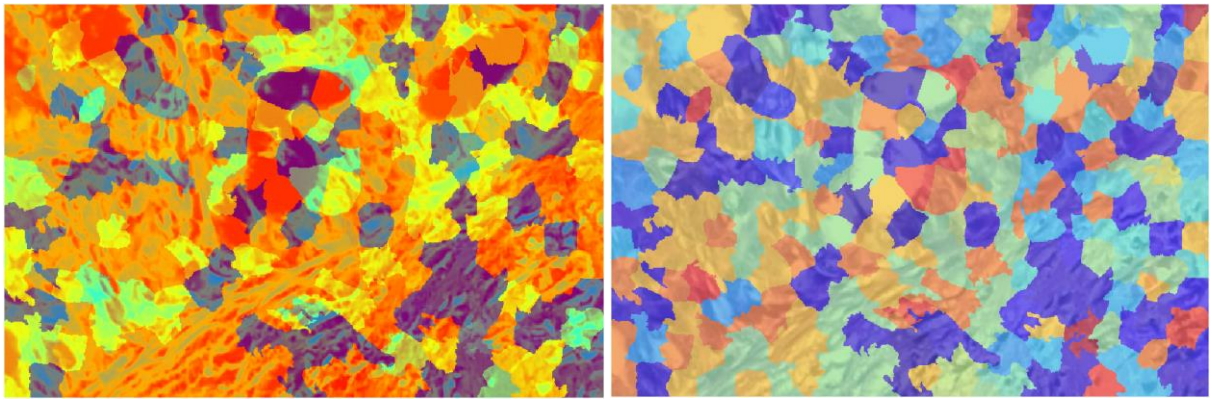

Figure 31: Outputs of “09.png”

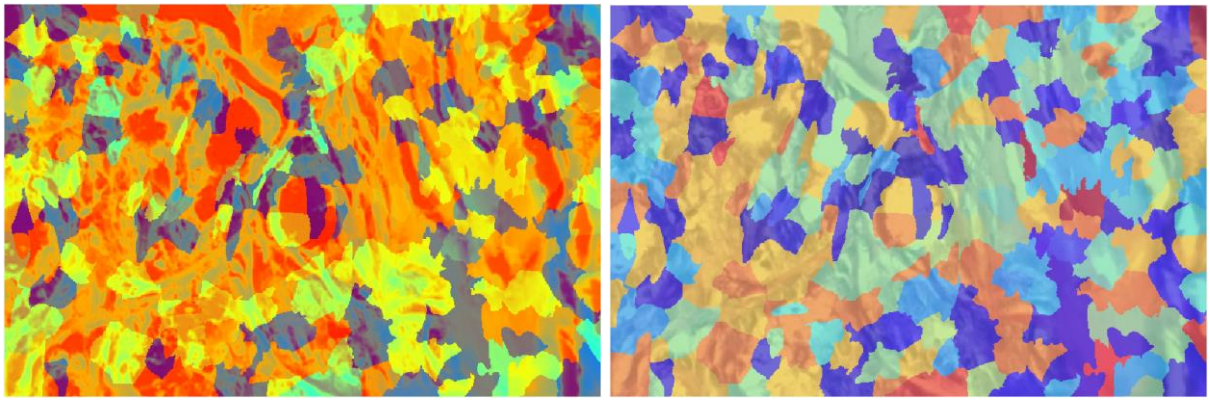

Figure 32: Outputs of “10.png”

Part 4

The outputs of this part are taken by first and second neighbour representations that are integrated to the representations in previous part. To show result better, both the false color (right) and pseudo color(left) versions are added to each figure. Clustering and segmentation results can be shown below for part 4.

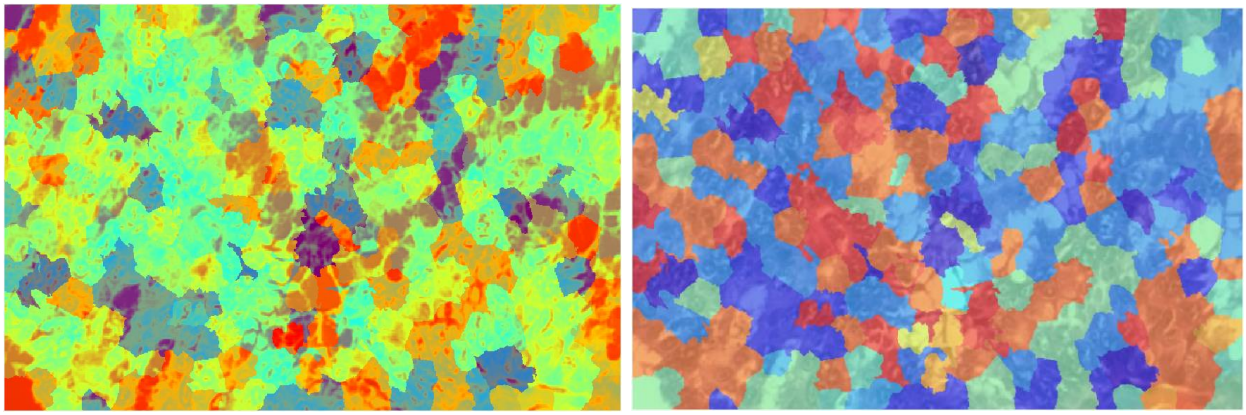

Figure 33: Outputs of “01.png”

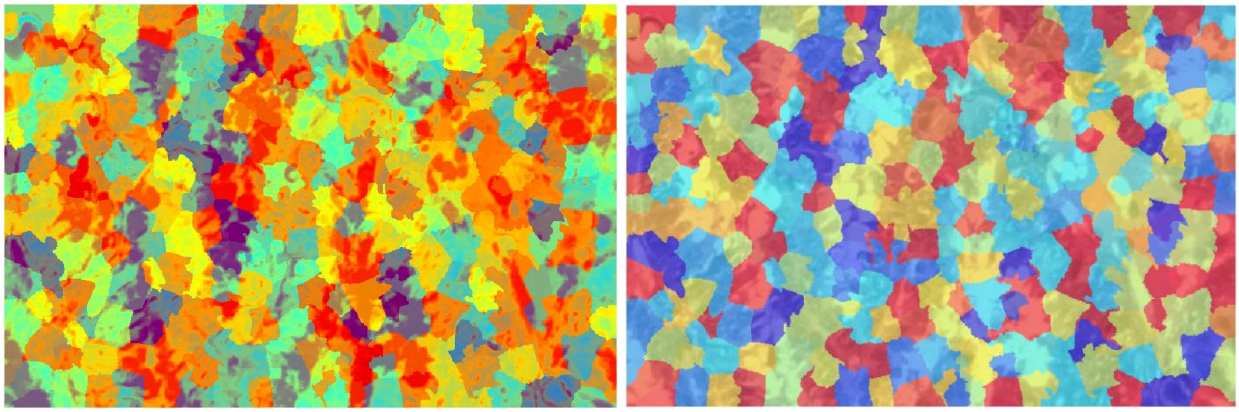

Figure 34: Outputs of “02.png”

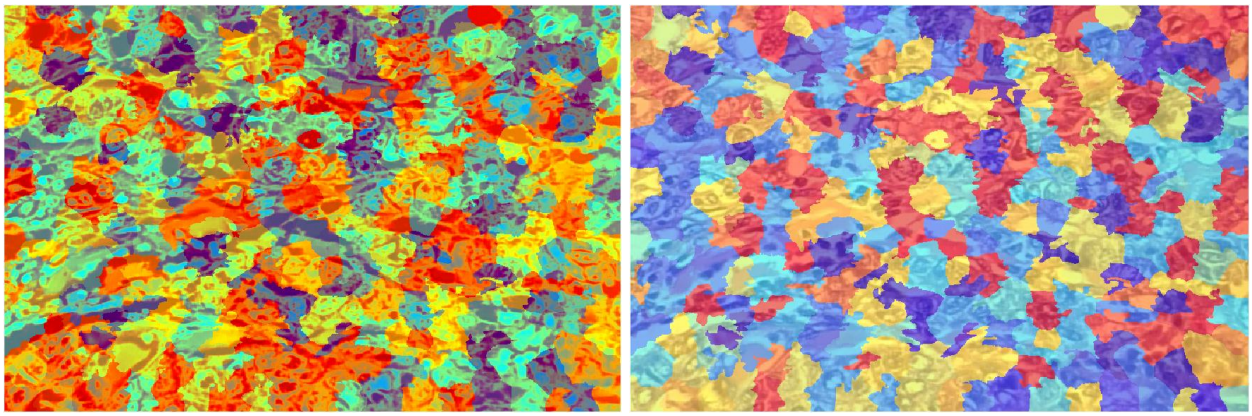

Figure 35: Outputs of “03.png”

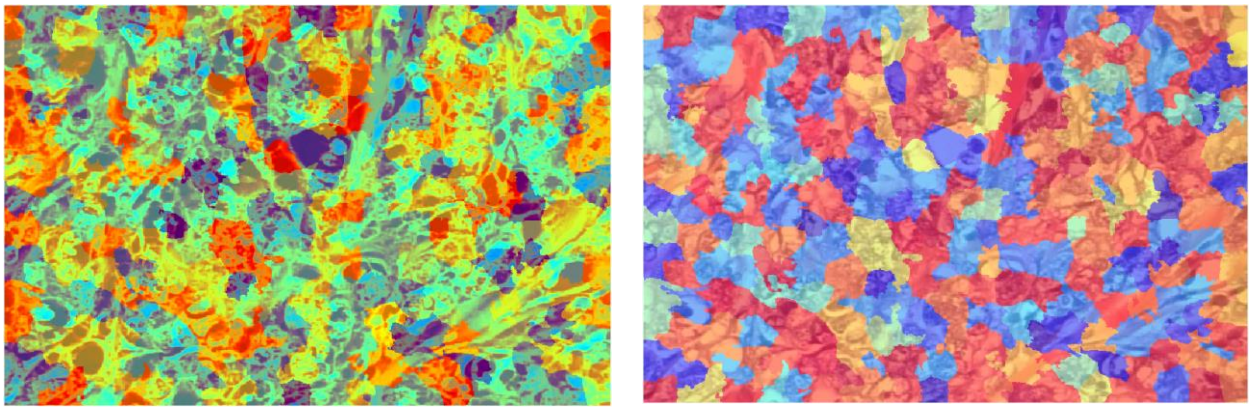

Figure 36: Outputs of “04.png”

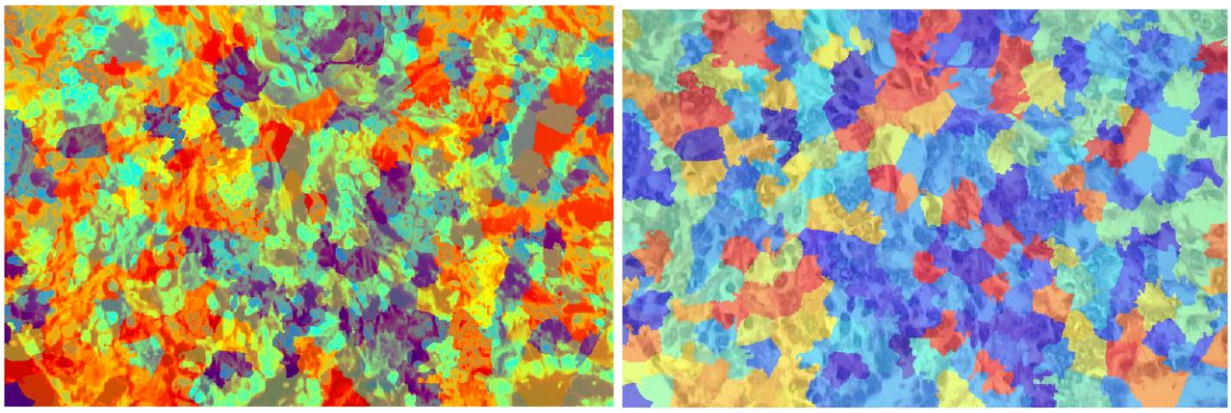

Figure 37: Outputs of “05.png”

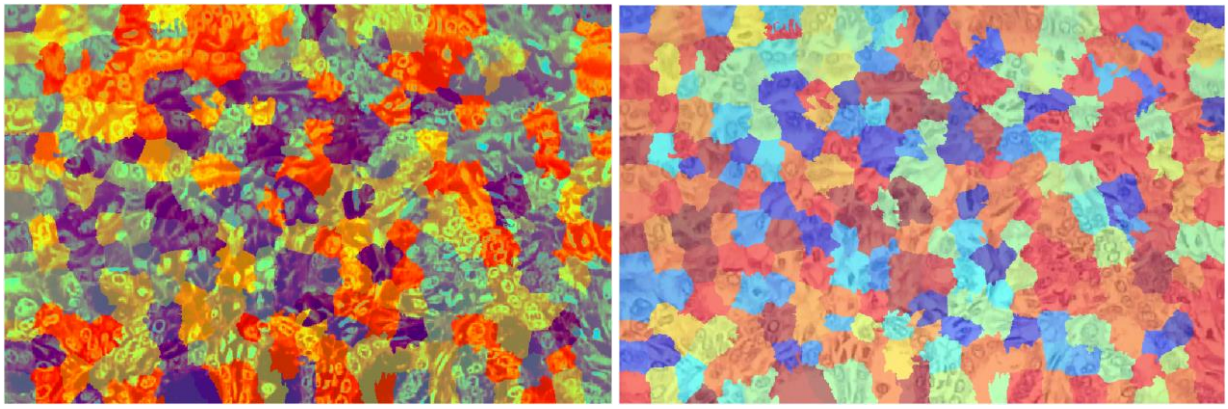

Figure 38: Outputs of “06.png”

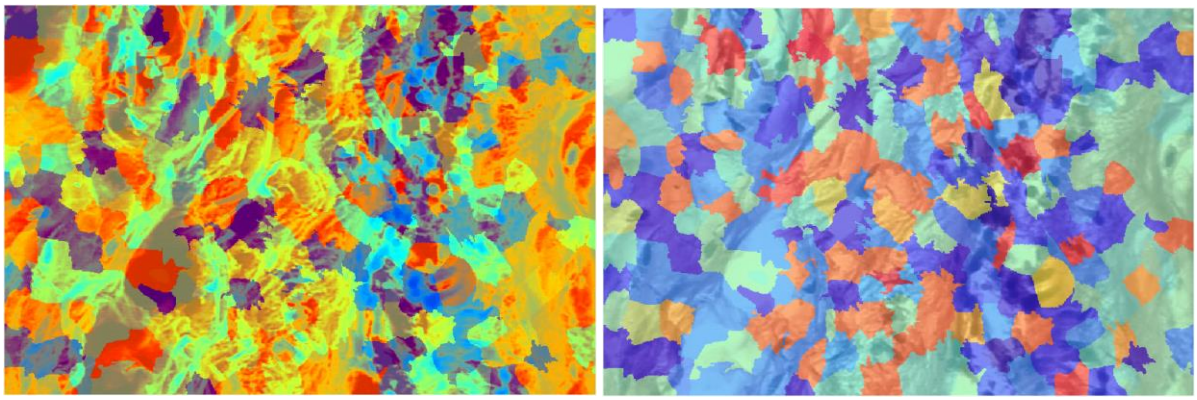

Figure 39: Outputs of “07.png”

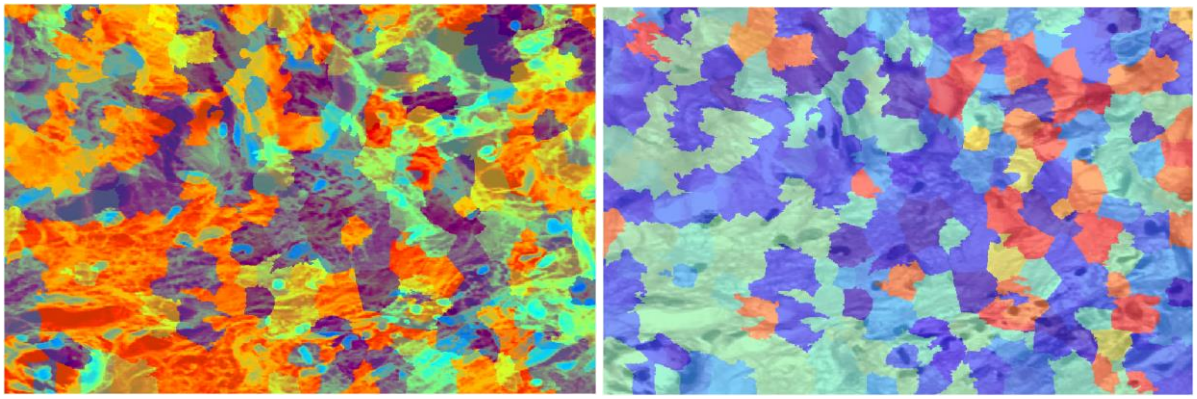

Figure 40: Outputs of “08.png”

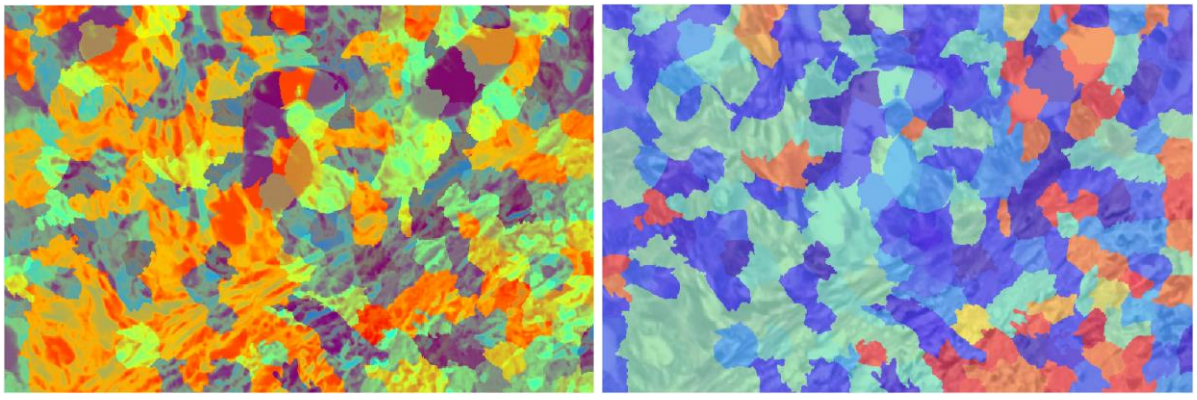

Figure 41: Outputs of “09.png”

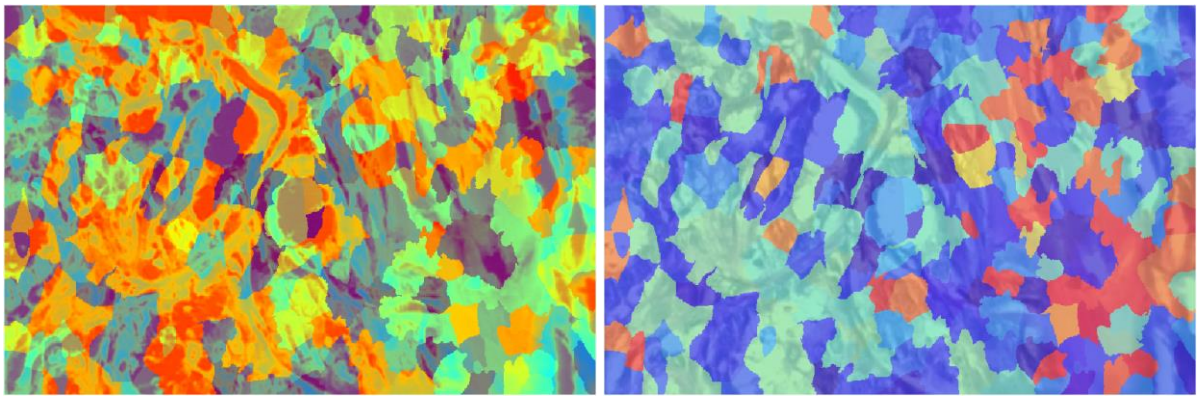

Figure 42: Outputs of “10.png”

Discussion and How I selected the parameters in Part 3/4:

I had to determine a threshold for a superpixel to cite it as a neighbor of another superpixel by checking the distance. For this purpose, I tried 75, 50 and 20. The results of each one applied to “02.png” can be seen below.

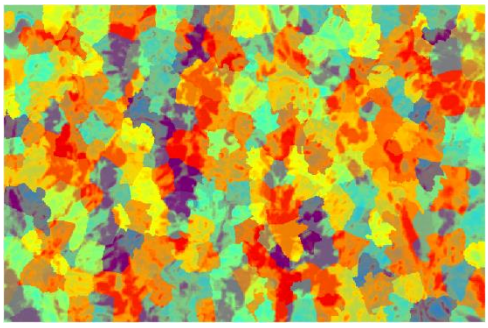

Figure 43: Output of “02.png” with threshold = 75

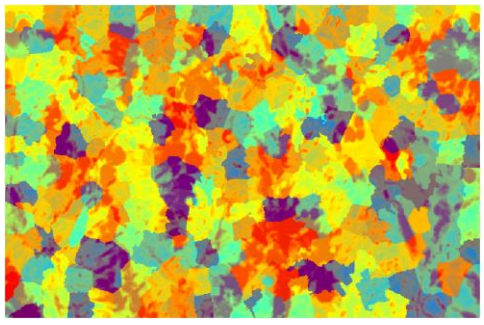

Figure 44: Output of “02.png” with threshold = 50

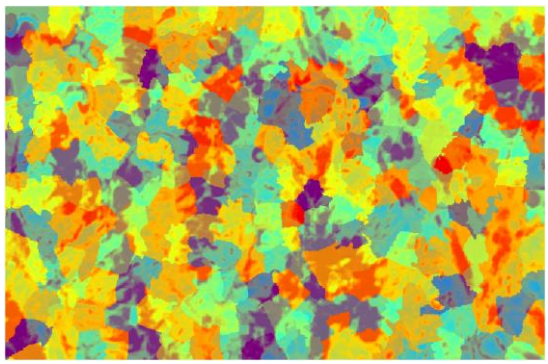

Figure 45: Output of “02.png” with threshold = 20

As it can be seen from the Figure 43, 44 and 45, outputs with threshold 20 and 75 are unstable. While sometimes gives better segmentations, in some other tries it becomes worst. When I tried with 50 the regions becomes more visible than the other values. Therefore I selected 50 as the threshold and the results can be seen in previous part.

I had to wavelengths for the Gabor filter. Wavelength as a parameter means that what should be the size of the texture to get a strong filter response. When I had small tries over different values I decided to use [10 15 20 25] which gives the best result and applied in the previous parts. Wavelength vector also gives unstable results. While in some tries it gives better results in another one it does not satisfy. When I tried with small numbers (smaller than 10) I get a filter response strong in whole image. Therefore, this disturbs the information coming from gabor filter. This problem does not appear on [10 15 20 25]. And thşs values gives the best results (most informative/ obvious) for segmentation.

As another parameter I needed to determine cluster count for k-means algorithm. When I tried this value with huge numbers like 90 -100 the results became noisy as expected. When I tried with small numbers like 5-10 decreased the ratio to differentiation among superpixels. Therefore I tried 20 as this parameter and get the best results. And, applied in all results that can be seen above.

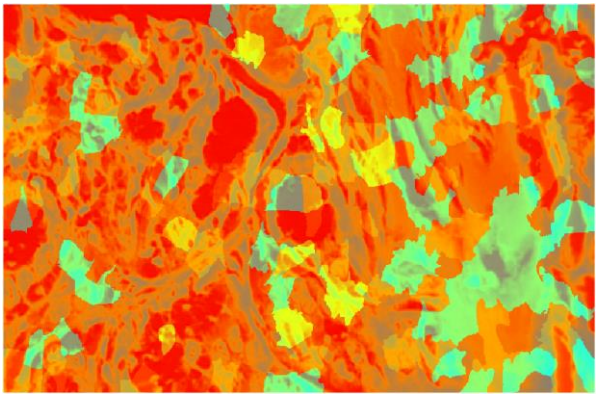

Figure 46: Output of “10.png” with k = 5

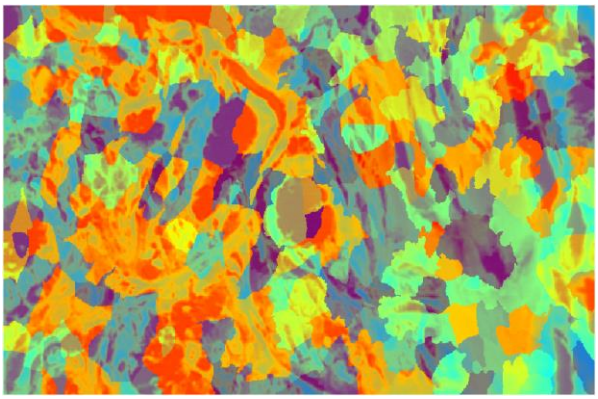

Figure 47: Output of “10.png” with k = 20

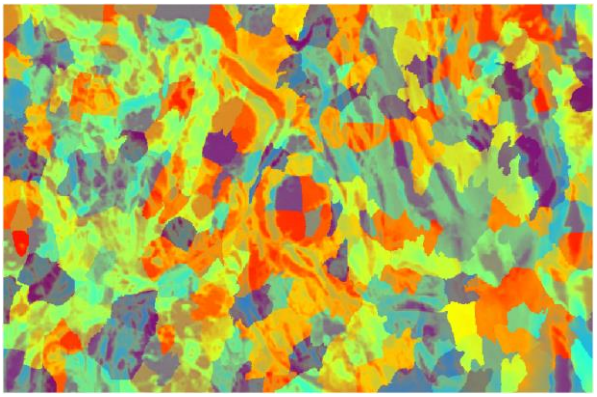

Figure 48: Output of “10.png” with k = 100

As you can see from the Figure 46, 47 and 48 best results are taken while the k=20, and the results are not appropriate for k = 5, k = 100. Especially the problem with k = 5 is obvious.