ECE4655 Project 4 Segmentation

In this project, I wrote a program that performed two different operations on a grayscale image in MATLAB. Grayscale simplified the processing for both segmentation types, and for the binary thresholding in part one, is an ideal way of generating the output. In part two, K-means clustering and connected component analysis, using grayscale reduced the computations to a difference of 2 values instead of having to compute the Euclidean distance. Scaling both processes up to color images would add a few extra calculations to get the distances, but otherwise would not meaningfully increase the time complexity of the program.

The first operation is binary thresholding, for which I used the mean gray level of the image as the initial threshold. The results were straightforward for some of the images, but the noisy one ended up still having the same noise. I did not have any other issues getting thresholding to work since it was just binary on a grayscale image. I implemented the algorithm starting with a chosen threshold value, in this case the mean gray level of the image. Using that, assign each pixel to the high and low group based on if they are brighter or darker than the threshold. Adjust the threshold value according to the equation (m1 + m2)/2 where m1 is the mean of the high pixels and m2 is the mean of the low pixels. Using the new threshold value, repeat these steps until the change in threshold value is below a user defined minimum, in this case 1. I believe all the images I had completed in ten iterations or fewer.

The second operation was k-means clustering. The algorithm starts with picking several k values to serve as center points for each cluster. Then, each pixel is assigned to the nearest center to form the clusters. Since I was processing grayscale only, I used gray level as the distance, making clusters of pixels that fall in small range of values near each center. After all pixels are assigned, update the center values to the mean of the cluster values and reassign the clusters. After either a predefined number of iterations, or the clusters stop changing, stop processing. I set 1000 as the maximum iterations, and total clusters as 16. For the test images, most of them converged after about 30 iterations. When using higher total clusters, many of the low or mid-range clusters would converge to the same set of pixels. After converging, I ran a simple check to remove all duplicate clusters starting from the lowest value ones, or clusters that had no pixels, which seemed to happen at the lowest value cluster and highest value as well. We were supposed to do connected component analysis on each cluster after in order to clean up the smallest components, but I was unable to get that function to work. Using the built-in bwconncom() function worked, but I was not sure how to compute the centroids of each component.

YouTube Link: <https://youtu.be/7NK7TfiTN4o>

Test Images: Fig1001(a) original 

thresholded k-means clustered

figure 1001d

Original



thresholded: 4 intermediate values and the final value

Figure 1001d k-means clustering

Intermediate iterations 5 and 10

Final Iteration



Figure 1026a

Original



Intermediate 1 Intermediate 2 Final threshold



K-means Clustering intermdiate 5th 10th and 15th iteration

Intermediate iterations 20th 25th 30th

35th iteration and final

Figure 1027a Original final thresholding result

Thresholding intermediate levels 1-2

Intermediate levels 3-4

Figure 1027a K-means

Iterations 5 and 10

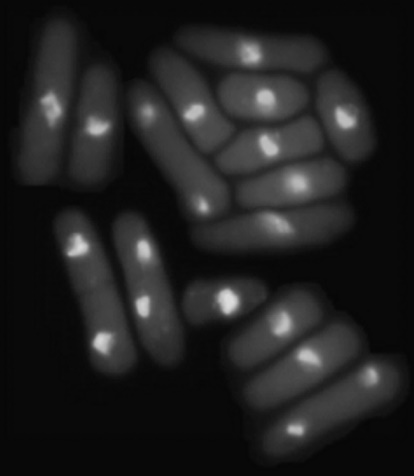
Iterations 15 and 20

Iterations 20 and final

Figure 1043a Original



Threshold iteration 1-3

Final



K-means iterations 5, 10, 15

Iterations 20, 25, 30

Iterations 35, 40, 45

Iterations 50, 55, Final