Project F: Automatic Image Segmentation

Casey Fleck

CSCI 445 Image Processing and Analysis

Due date: Friday April 28th, 2017

Purpose: To segment each of the input grayscale images from the EM stack 'train-volume.tif', to get pixel values as close as possible to the true segmented images from the EM stack 'train-labels.tif'.

Methodology: A grayscale image, f, from 'train-volume.tif' was imported from a MATLAB folder using the function *imread*. In the function ProjectF_07, the input image f, was converted into a black and white image using the function *im2binarize*. The function *imcomplement* was used on image f in order to further filter the black and white logical image. The logical image f was then filtered by using the function *bwlabel* to find all the shapes in the image. These shapes that had an area smaller than a certain value (5000), were turned to black using a *for-loop* (similar to a previous method used during an 'InClass' assignment from chapter 7). The function *bwareaopen* (taken from the MATLAB documentation) was used to further clean the logical image f of noise. The function *imcomplement* was used again, and the new image, BW, was returned through the parameters. The accuracy score was calculated from the formula:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

The F-score was calculated from the formula:

$$F_{1\,Score} = \frac{2TP}{2TP + FP + FN}$$

Where: TP refers to true positive, TN refers to true negative, FP refers to false positive, and FN refers to false negative. The ACC and F scores were calculated against the logical images from 'train-labels.tif'.

Both the ACC and F scores were stored into 2 separate vectors. The above process was repeated 30 times in a *for-loop* for each of the images in 'train-volume.tif'. After the *for-loop* was completed, the average ACC and F score was calculated by summing all the values in the ACC vector and summing all the values in the F vector, and then dividing each sum by 30. The image number, along with ACC scores and F scores were stored in a table using the function *uitable*.

Results & Discussion:

Results: The 30 images from the EM stack 'train-volume.tif' that were processed through the function ProjectF_07 had ACC scores ranging from 0.8505 to 0.8956, and F scores ranging from 0.8983 to 0.9344. The average ACC score was 0.8776 and average F score was 0.9216 (Table 1).

Discussion: The filter created in the function ProjectF_07, displayed some images that had a closer approximations to the true image from 'train-labels.tif' than others, as shown by the ACC and F scores from table 1. Table 1 shows that the function ProjectF_07 seemed to adequately create a logical image that had satisfactory high ACC and F scores. This indicates that the pixel from each computed logical image had very similar pixel values to the ones from the true images in 'train-labels.tif.'

The ACC score computed the ratio of the amount of pixels that were accurately classified to the total amount of comparisons that were made between each pixel in each image. This gave a basic accuracy score on how well the function ProjectF_07 was able to create accurate pixel values in the logical image. However, the ACC score does not show a ratio that covers how many pixels that were not classified correctly. The F score gives the ratio of the weighted average of precision and recall. The F score is important because it shows how close the returned logical image was to the true image. If the image BW had a high F score, then the function ProjectF_07 did a good job at segmenting the gray-scale image and cleaning unnecessary black values.

For the images that had low ACC and F scores, the lines in the original gray-images were not well defined. Either they had lines not dark enough nor light enough for the threshold, therefore being classified with the wrong pixel values. Figure 1 shows image 6, which turned out to have a low ACC of 0.8505 and F score of 0.8983, and image 17, a figure that had a high ACC score of 0.8939 and F score of 0.9344. These images had a large width variation in F and ACC scores. The low scores from image 6 would not be desirable. An example of when having low scores similar to the scores from image 6 would be bad would be if the goal were, for example, to find cancerous spots in images of acute brain slices. An inaccurate returned logical image could be the difference between someone being able to detect and fight brain cancer early on, had the image been accurately filtered. However, the amount of inaccuracy from the function ProjectF_07 might also be the difference of not finding the cancerous cells from the gray-scale image.

The function ProjectF_07 might have addressed the poor accuracy of image 6 by: adjusting the contrast or using the function *imerode* to further detect unnecessary noise. Adjusting the contrast of the gray-scale image could have further defined the necessary lines for the logical image. Applying a function like *imerode* could have separated, or gotten rid of, unnecessary noise from the logical image BW.

Conclusion: The goal of the function ProjectF_07 was to compute a black and white logical image from the images in the EM stack 'train-volume.tif' that had a close approximation to the pixel values to the true logical images from 'train-labels.tif'. In conclusion, segmentation of these specific image included using the function *bwlabel* and *bwareaopen.* Both functions worked fairly well for cleaning the noise of the converted logical image and retrieving the necessary parts of the image in order to get a more accurate ACC and F score. Noticeable improvements needed in the code would be adjusting the contrast of the original gray-scale image to better define necessary lines. In addition, *imerode* could have been used in a more efficient way to better clear the noise from the newly created logical image BW.

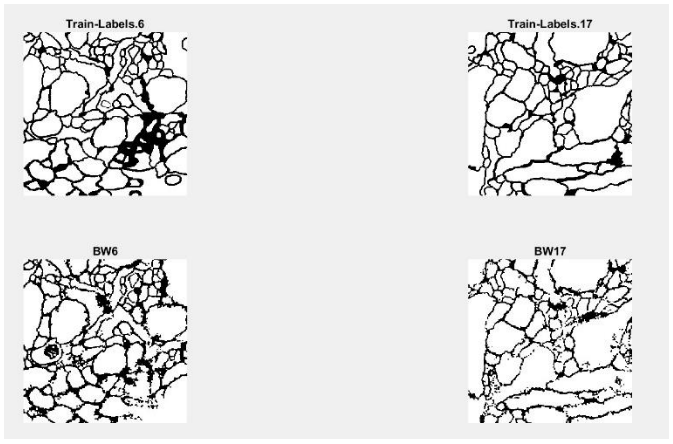
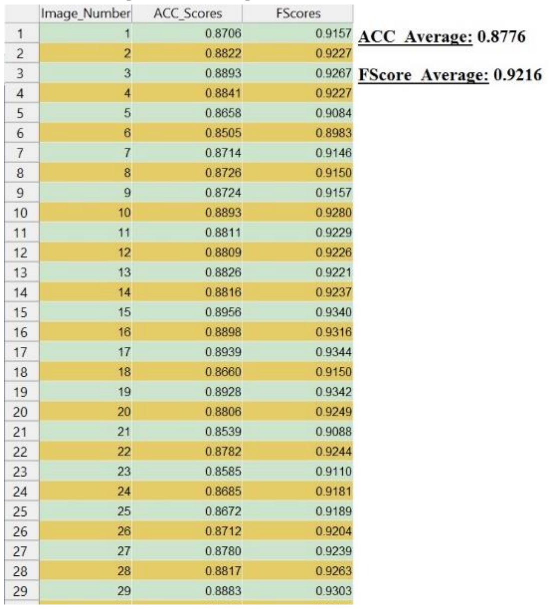


Figure 1: True and segmented images, with high and low F and ACC scores.



	Image_Number	ACC_Scores	FScores
1	1	0.8706	0.9157
2	2	0.8822	0.9227
3	3	0.8893	0.9267
4	4	0.8841	0.9227
5	5	0.8658	0.9084
6	6	0.8505	0.8983
7	7	0.8714	0.9146
8	8	0.8726	0.9150
9	9	0.8724	0.9157
10	10	0.8893	0.9280
11	11	0.8811	0.9229
12	12	0.8809	0.9226
13	13	0.8826	0.9221
14	14	0.8816	0.9237
15	15	0.8956	0.9340
16	16	0.8898	0.9316
17	17	0.8939	0.9344
18	18	0.8660	0.9150
19	19	0.8928	0.9342
20	20	0.8806	0.9249
21	21	0.8539	0.9088
22	22	0.8782	0.9244
23	23	0.8585	0.9110
24	24	0.8685	0.9181
25	25	0.8672	0.9189
26	26	0.8712	0.9204
27	27	0.8780	0.9239
28	28	0.8817	0.9263
29	29	0.8883	0.9303

<u>ACC Average:</u> 0.8776

<u>FScore Average:</u> 0.9216

Table 1: Image segmentation scores for each image used in EM stack 'train-volume.tif'.