**Project Title:  Image Processing Fundamentals**

**Project Number**: Project 8/9

**Course Number**: CEG 7850-01

**Student's Name**: Alex Reigle

**Date Due**: November 29, 2021

**Date Submitted**:  November 24, 2021

**Declaration Statement:**

I hereby declare that this Report and the Matlab codes were written/prepared entirely by me based on my own work, and I have not used any material from another Project at another department/ university/college anywhere else, including Wright State. I also declare that I did not seek or receive assistance from any other person and I did not help any other person to prepare their reports or code.  The report mentions explicitly all sources of information in the reference list. I am aware of the fact that violation of these clauses is regarded as cheating and can result in invalidation of the paper with zero grade. Cheating or attempted cheating or assistance in cheating is reportable to the appropriate authority and may result in the expulsion of the student, in accordance with the University and College Policies.

**Abstract:**

In this project I developed a function to applying thresholding on an image both manually and automatically. This function has the option to return a binary image. I have also implemented built-in MatLab functions to use Otsu’s algorithm for thresholding. I also developed a method of boundary extraction through the use of image dilation and erosion. Finally, I develop a function that can extract the connected pixels in an images extracted boundary.

**Technical Discussion:**

Problem 1 implements a Sobel gradient mask from eqn. 10.14 from the text and figure 10.14(f-g), as follows:

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

This is implemented using the built-in MatLab nlfilter function. Prior to the filtering technique, I apply a Gaussian smoothing function with a standard deviation of 3 and a threshold value of 85, which were both found through trial and error.

Problem 2 recreates example 10.13 from the text. I have written an automatic thresholding function which uses the textbook defined procedure (from p746). This includes a running calculation of the upper and lower local means, based on an initial threshold value. Each iteration resolves for these values until a selected minimum change is not exceeded, this returns the final threshold value. I have left an artifact in the code showing that this is achievable using only built-in MatLab functions. This was also used for results comparison.

Problem 3 utilizes built-in MatLab functions to implement Otsu’s algorithm for optimal thresholding. This is possible in many ways, thus I have selected one method and commented out an alternative method for comparison.

Problem 4 implements a boundary extraction method using erosion and dilation of a 3x3 filter with arbitrary shape. I have implements two methods of boundary extraction. One method is implemented using only image erosion and the intersection, differencing, and complementation. In this method I use the logical element wise functions and and not as well as imerode to achieve this. The other method is implemented using only erosion (imerode) and dilation (imdilate) to extract the boundary by subtracting the former from the latter resulting image.

Problem 5 makes use of the previously developed binary thresholding function and the built-in MatLab function, bwconncomp, to return the various connected pixel regions. Using this and a threshold, found through trial and error, I was able to exactly recreate the values from the textbook example 9.7.

**Discussion of Results:**

Problem 1 consisted of Gaussian smoothing, edge detection using a Sobel gradient and thresholding to convert the image in a binary image. From looking at the histogram of the image (shown below in Fig. 1b), I chose a standard deviation of 3 for the Gaussian smoothing function. From this and some trial and error, I was able to choose at threshold of 85 to produce good results. These results (shown in Fig. 1c) show the primary vein/vessel in the original image (from Fig. 1a). It was noticed that the edges detected were very sensitive to both the smoothing and the threshold selected. Poorly chosen values can result in incomplete edges or lots of noise that isn’t associated with a ‘real’ edge.

Problem 2 could have been completed entirely with built-in MatLab functions. I was able to use these functions as a means to validate the output for my own functions. I auto-calculate a threshold using the procedure described in the book to produce the results shown in Fig 2b. While the thresholding would make the image more easily interpreted by a computer program, it did little to improve the quality of the image. All imperfections that are in the original image could not be removed using this procedure.

Problem 3 is very similar, in outcome to problem 3. However, instead of comparing a change in mean, this method is to implement Otsu’s algorithm. This is done using MatLab built-in functions and can be seen in Fig. 3. I was able to find two ways to calculate the threshold value using Otsu’s method, so I left one commented out as an artifact of my verification process.

Problem 4 included extracting an image boundary through the use of erosion, dilation, and logical relation operations. I also developed a way to extract the boundary of the image using only erosion and dilation. These results can be seen in Fig. 4, where 4b shows the former method and 4c shows the latter. The interesting difference between these two methods is the fineness of the boundary line drawn. Since the image shown in Fig. 4b doesn’t use dilation, its boundary line is thinner and finer.

Problem 5 utilizes many of the concepts used in problems 1-4, where here I extract a boundary from the image (using a slightly different method) and quantify the different boundaries and the connectiveness of them. Through trial and error I was able to find the threshold value to exactly recreate the results of the book, that value being 203. The results of this thresholding is shown in Fig. 5b. The image shown in Fig. 5c shows the boundaries extracted from the original image (Fig. 5a). Table 1 shows that the results of this threshold and boundary extraction results in the exact same results from the textbook example that I was tasked to replicate. This shows that with the selected threshold, I have identified 15 separate impurities for removal.

**Results:**

Chart, histogram

Description automatically generated

Figure 1: These are the results of Problem 1. (a) is the original image. (b) is the histogram resulting from (a). (c) is the edge detection image after binary thresholding is applied with a threshold value of 85.



Figure 2: These are the results of problem 2. (a) is the original image. (b) is the image after auto-thresholding is applied to it.

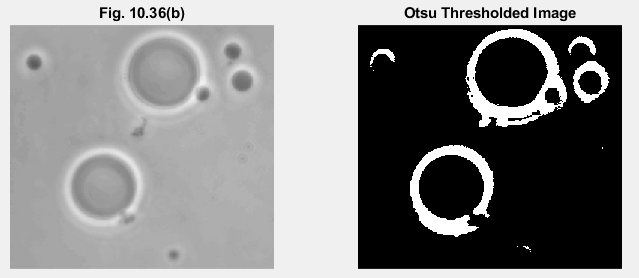


Figure 3: These are the results of problem 3. (a) is the original image. (b) is the image after applying the Otsu threshold algorithm.

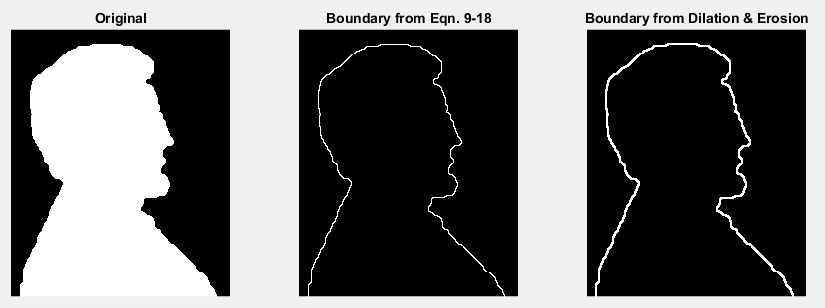


Figure 4: These are the results of problem 4. (a) is the original image. (b) is the boundary extracted using eqn. (9-18) from the textbook. (c) is the boundary extracted using dilation and erosion exclusively.

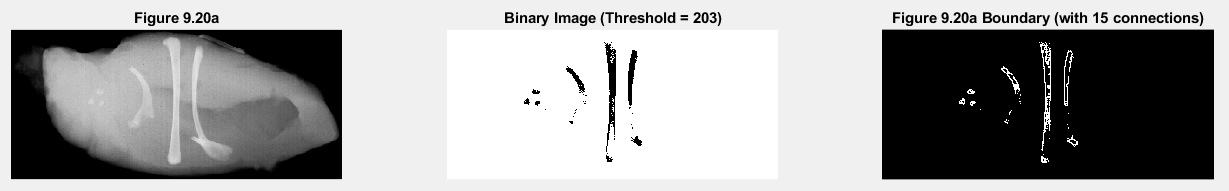


Figure 5: These are the results of problem 5. (a) is the original image. (b) is the resultant of thresholding at a value of 203. The image has been inverted for clarity, as in the text. (c) is the boundary extracted from (b) with 15 objects forming connections.

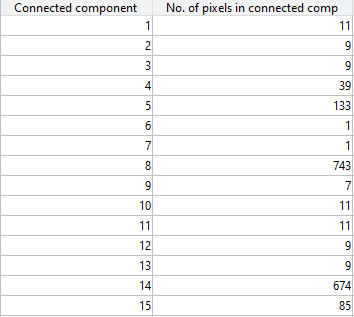


Table 1: This table is the result of problem 5. This shows the number of pixels connected in each distinct object found. These results are identical to that of the textbook.

**Written Programs:**

* main
* extractBoundary
* binaryThreshold
* autoThreshold
* sobelGrating

**Utilized Programs:**

* subplot
* imshow
* cast
* imread
* nlfilter
* title
* length
* ostuthresh
* imbinarize
* num2str
* imerode
* imdilate
* uitable
* zeros
* size
* plot
* bar
* and
* not
* pwd
* bwconncomp
* strel
* max
* mean
* imhist
* imgaussfilt