Connor McCurley

EEE 6512 **Homework 5 Due November 3, 2018** Fall 2018

Part I MATLAB Programming

For this question, we were provided the following image (Fig. 1) and asked to separate the foreground (lake) from the background (rest of the image).

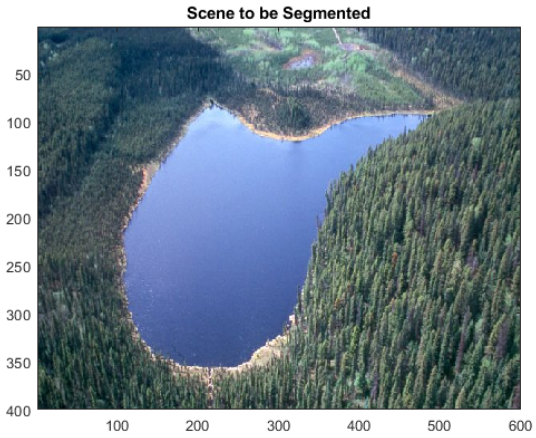

Figure 1: RGB scene to be segmented

To accomplish the segmentation, I first applied Gaussian smoothing to each color channel with $\sigma = 2$. This made sense to do since I didn't want to consider all of the high-frequency "noise" scattered throughout the image, such as in the trees. Since we only cared about the border regions between the foreground and background, I then applied a Sobel edge detector in each color channel to find the first derivative edges. The histograms for Otsu's method only considered these edge pixels. The results of the edge detection are shown in Fig.2

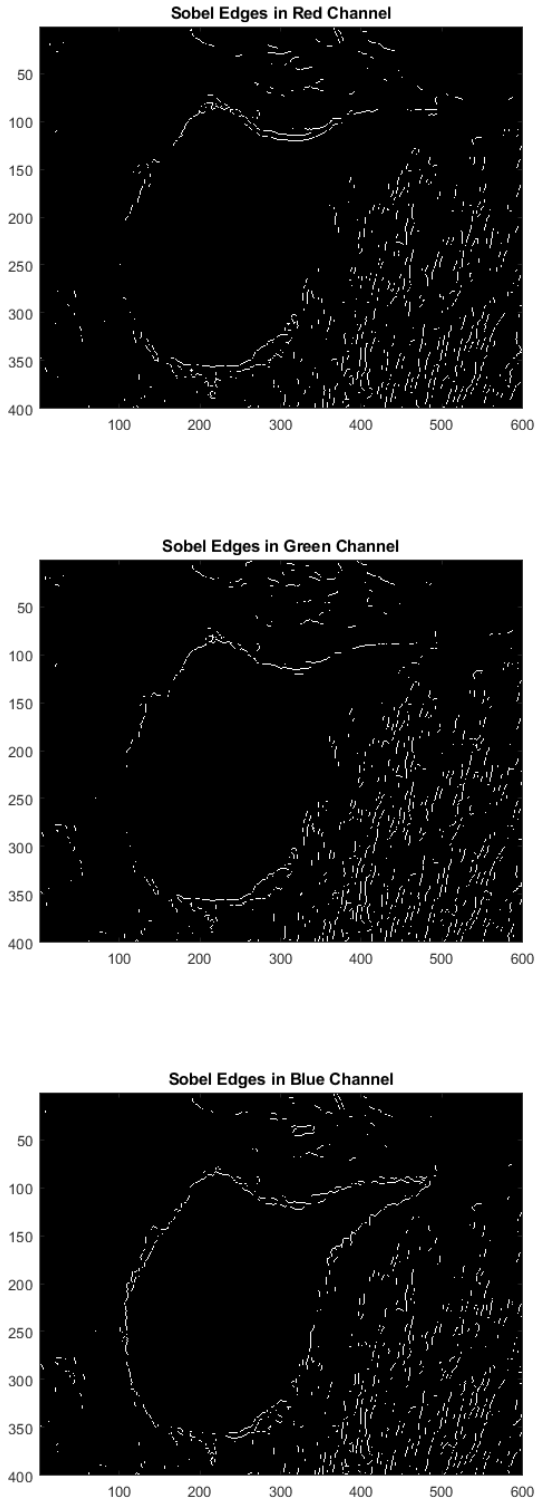

Figure 2: From top to bottom: Sobel edges detected in the red channel, green channel, and blue channel

The normalized histogram was found in each color channel using code written for HW02. The histograms only considered the pixels found from the Sobel edge detection. The normalized histograms of edge pixels

are shown in Fig. 3.

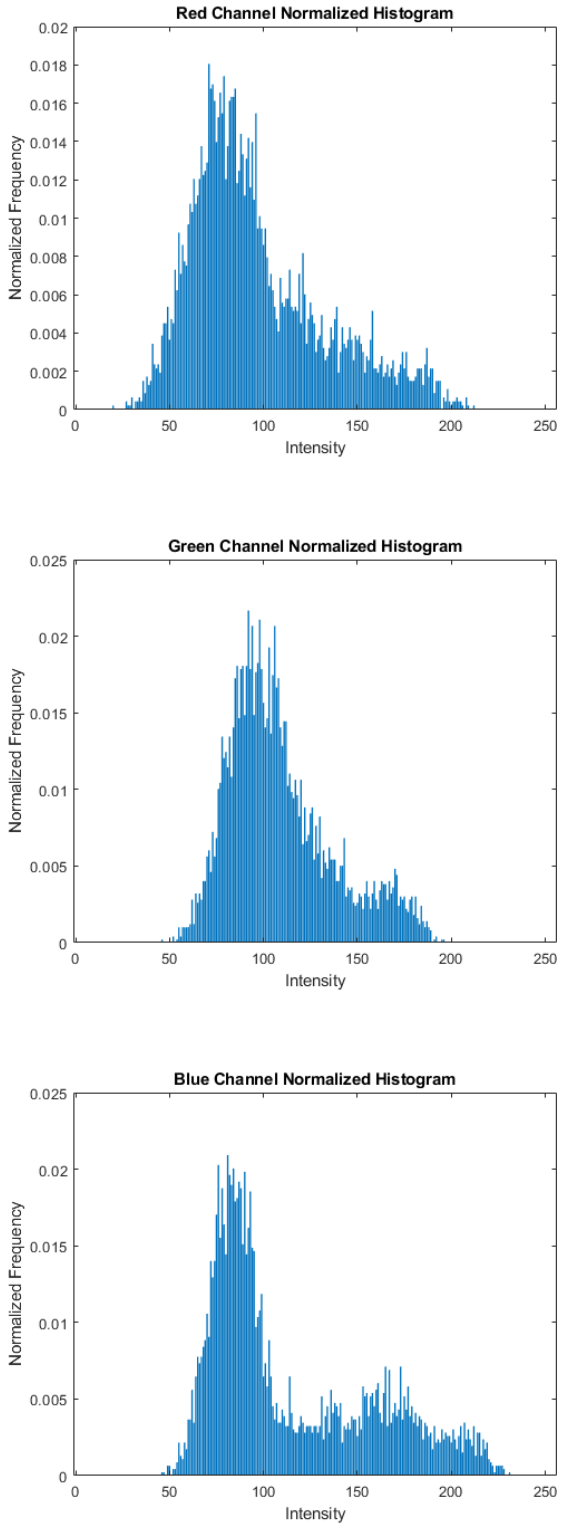

Figure 3: From top to bottom: Normalized histogram of Sobel detected edge pixels detected in the red channel, green channel, and blue channel

It was clear from the normalized histograms that two modes existed in both the green and blue channels. This was promising for segmentation using Otsu's methods since it effectively finds the optimal boundary between two modes in a distribution.

Otsu's method was then performed in each color channel following equations 10-49 through 10-62 in the textbook. The function I wrote, *otsu.m* took in the rgb scene to be segmented, performed Gaussian filtering, Sobel edge detection, and Otsu's method in each channel. It then returned a three channel image containing the binary Otsu segmentation for each color channel. The segmentation results using Otsu's Method for each color channel are shown in Fig. 4.

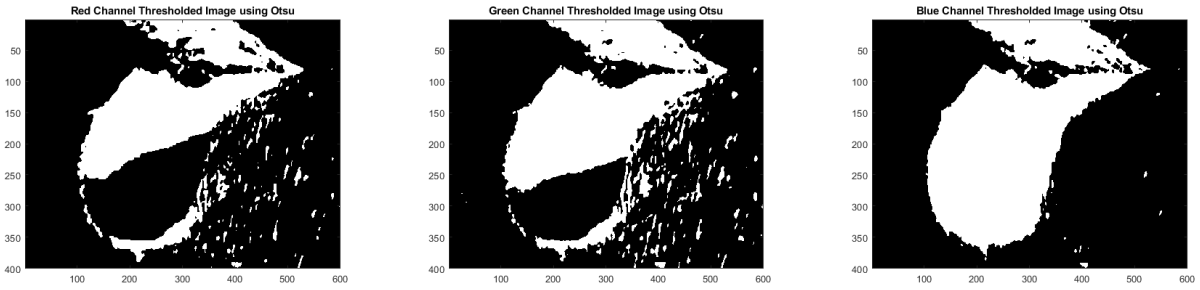

Figure 4: From left to right: Foreground exraction using Otsu's Method in the red channel, green channel, and blue channel

Although it was clear that the blue channel could have been used solely to extract the foreground, we wanted this to be an autonomous process. Therefore, the channels needed to be aggregated to utilize the foreground information contained in each. This was accomplished using a logical OR operation between each channel. The resulting image from the combination is shown in figure 5.

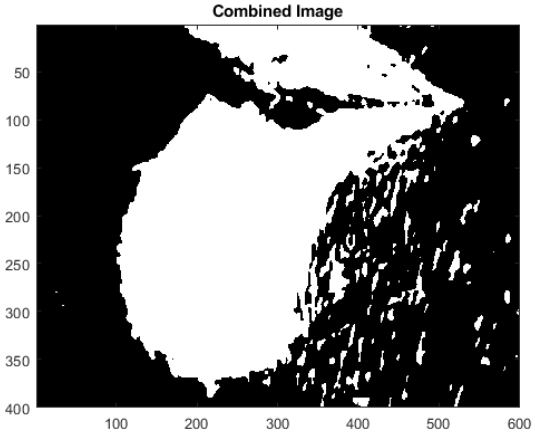

Figure 5: Binary image formed from the logical OR operation between rgb color channels segmented with Otsu's method

The resulting image contained a significant amount of "noise" outside of the foreground. To remove these spots, morphological opening (erosion then dilation) was utilized to rid the image of high areas smaller than a 10x10 disk structuring element (Fig. 6).

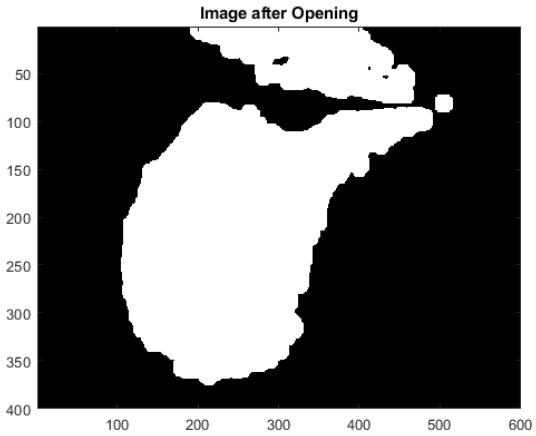

Figure 6: Binary image after opening with a 10x10 disk structuring element

The opening effectively removed the noisy areas outside of the foreground. Next, morphological closing (dilation then erosion) with a 15x15 disk structuring element was performed to close the small holes within the foreground (Fig. 7). This was necessary before boundary extraction so that no false object boundaries would appear within the foreground.

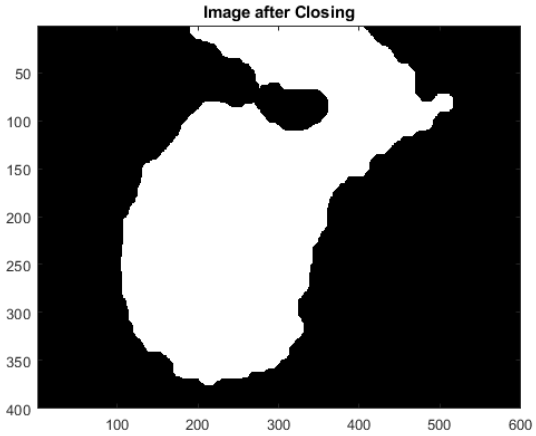

Figure 7: Binary image after closing with a 15x15 disk structuring element

The las step was to extract the boundary of the foreground. This was done by subtracting the binary image eroded with a 3x3 square of ones structuring element from the binary image. The eight neighborhood of each pixel on the boundary was set to 1 for visualization. Results of the boundary extraction are shown in figures 8 and 9.

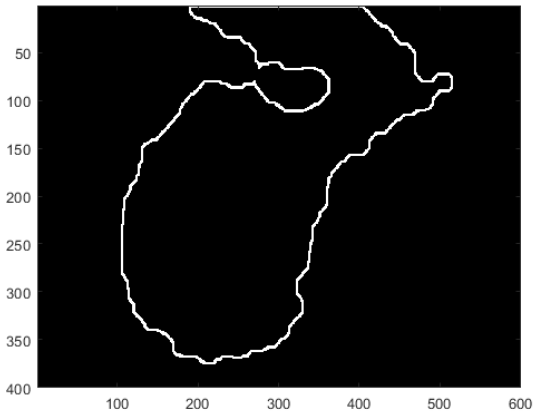

Figure 8: Boundary mask found by subtracting the closed image eroded by a 3x3 square of ones from the closed image. The boundary was widened by setting the eight neighborhood of the boundary pixels to 1's.

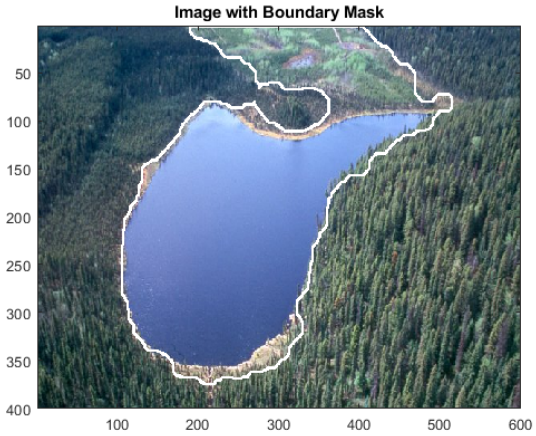

Figure 9: Original scene overlain with the computed boundary mask.

While segmentation results were not perfect due to many factors such as: smoothing parameters, size and shape of the structuring elements, and the general nature of the image, the entire foreground area was discovered using the selected methods.

The functions I wrote which were used for this portion of the assignment are as follows:

1. *mccurleyHW05.m* - Main function which runs the segmentation.
2. *otsu.m* - Applies Gaussian filtering, Sobel edge detection, and Otsu's method in each color channel.
3. *extractContours.m* - Aggregates the three binary images for each channel, performs morphological opening, morphological closing, boundary extraction, and widens the boundary mask.
4. *erodeImage.m* - Erodes a binary image by a given structuring element.

5. *dilateImage.m* - Dilates a binary image by a given structuring element.

6. *plotMask.m* - Plots the original scene overlain with the computed boundary mask.

7. *myHist.m* - Computes the normalized histogram of an 8 bit intensity image.

Part II Extra Credit

This problem gave us two images (Fig.10) and asked to segment out three organs of interest: liver, kidneys, and spleen. To accomplish this, the images were first smoothed by a Gaussian filter with $\sigma = 1.5$.

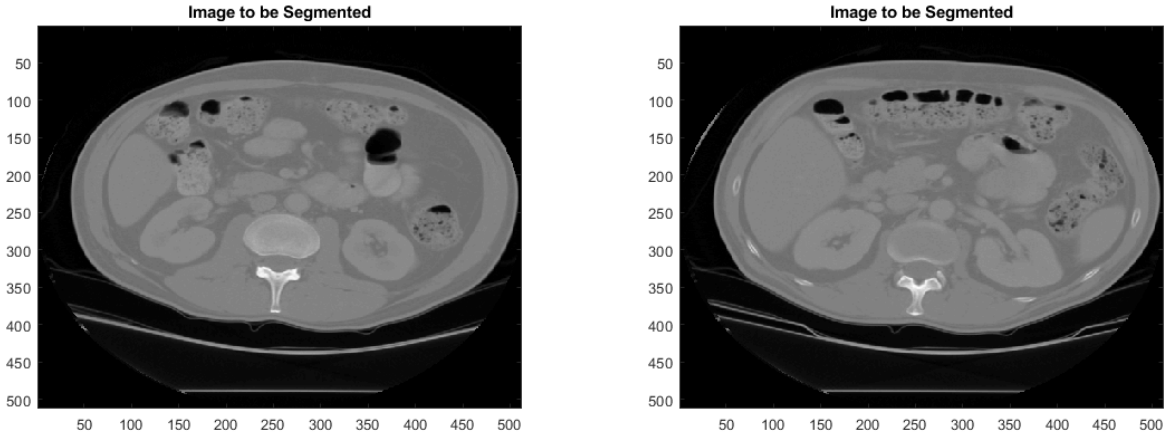

Figure 10: Extra credit images ex1 and ex2

Next, the normalized histograms of the images were computed and Otsu's method with two thresholds was applied (Figs. 11 and 12). It was apparent from both the images and the normalized histograms that two thresholds were needed for organ extraction. This was because most of the background pixels in the images were close to zero intensity, while the remaining image pixels consisted of various lighter gray values. Since we really wanted to extract the lightest gray levels from the image, the larger threshold value between the two calculated was selected for foreground extraction.

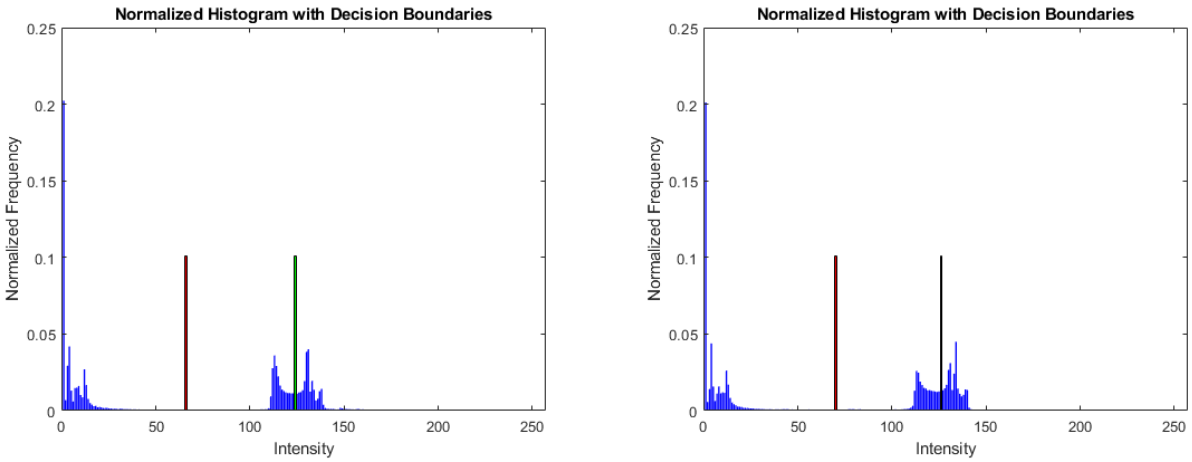

Figure 11: Normalized histograms of ex1 and ex2 with the corresponding thresholds chosen by Otsu's Method

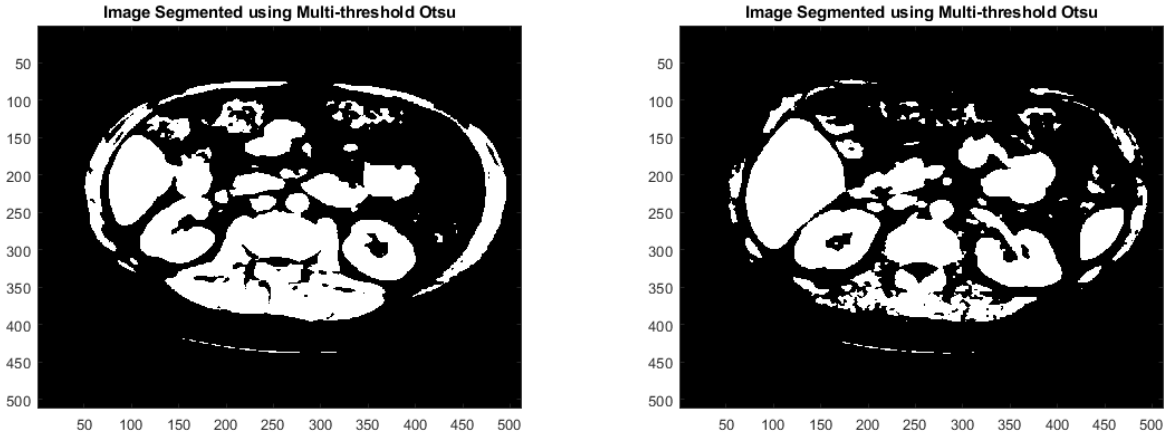

Figure 12: Foreground extraction of ex1 and ex2 with the largest corresponding thresholds chosen by Otsu's Method

After foreground extraction with Otsu's Method, it was apparent that some of the organs were being unnecessarily grouped together. To alleviate this problem, I applied morphological opening to break the narrow segments joining the objects. Opening was performed with a 12x12 disk structuring element. Results of the opening on each image are shown in Fig. 13.

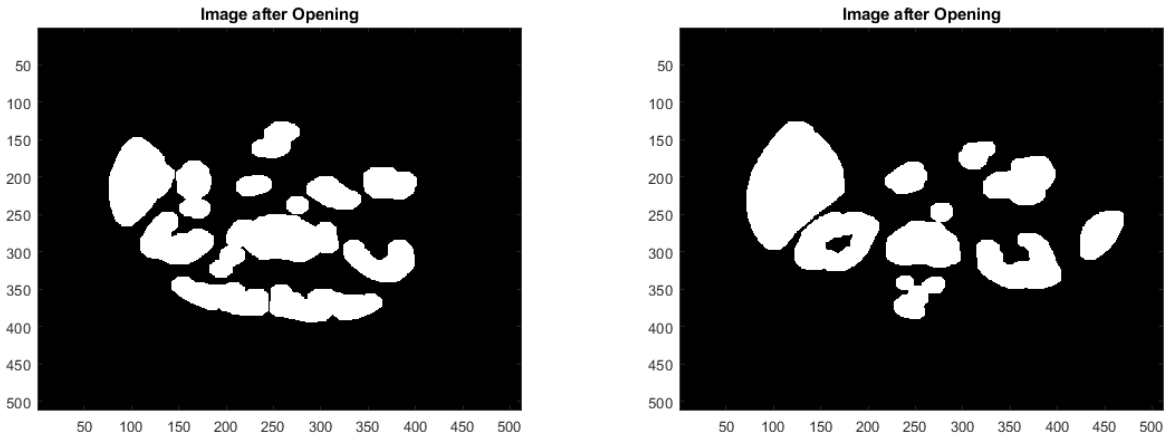

Figure 13: Segmentation of ex1 and ex2 after performing morphological opening with a 12x12 disk structuring element.

In order to rid the image of organs that were not of interest, connected components were extracted. A set of "ground truth" pixels were utilized for this, where a single pixel was chosen in the known region of an organ to act as a starting point for the connected components extraction method discussed in the textbook. As can be seen in figure 14, the liver, kidneys and spleen were all segmented out of the original image (the spleen was not present in the first image). It should be noted that the liver is colored green in the first image and yellow in the second, while both of the kidneys are colored red in the first and green in the second. The spleen is colored red in the second image and was not present in the first.

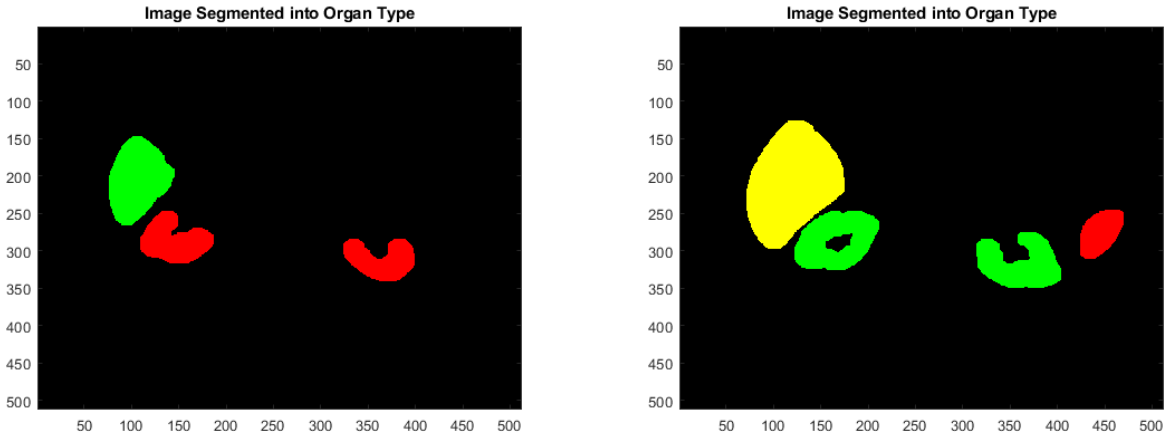

Figure 14: Extracted organs using connected components on segmented images

The functions I wrote which were used for this portion of the assignment are as follows:

1. *mccurleyHW*05.*m* - Main function which runs the organ extraction.
2. *otsuBinary.m* - Filtered image with a Gaussian, ran Otsu's method for two thresholds (three classes), and returned image thresholded by the greater of the two calculated thresholds.

3. *segmentOrgans.m* - Performed morphological opening to break narrow connections between objects, then found the connected components in the image using examples of the three organs of interest.

4. *connectedComp.m* - Performed connected components extraction by starting with a single example point and continually dilating the point with a 3x3 square of ones and intersecting the result with the original image as discussed in chapter 10 of the textbook.