Name: Iyer Krithika Ganesh Sundaram

UNID: u1135255

Email ID: u1135255@utah.edu

Project 1

1. Build a Histogram

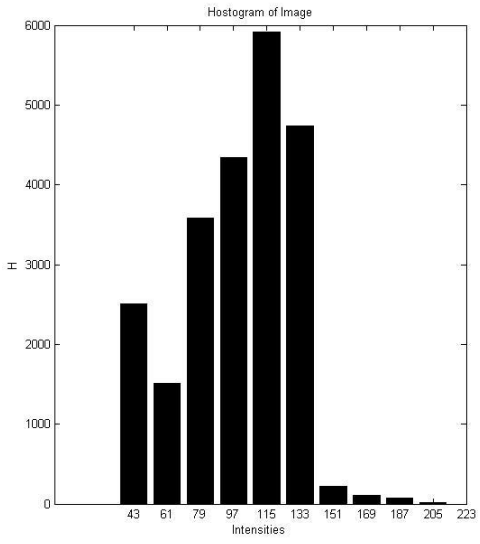
As described in the requirement, a program for plotting histogram has been implemented. The inputs to the program are as follows.

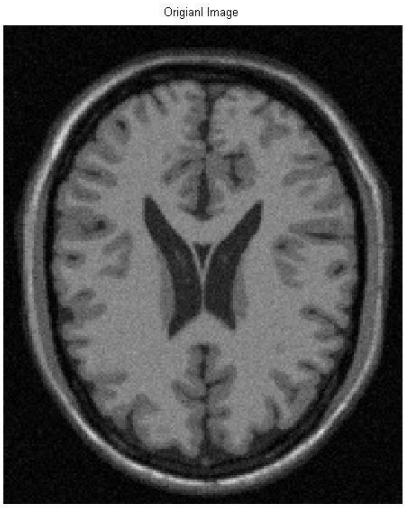
- Input Image
- Number of bins
- Minimum Pixel Intensity
- Maximum Pixel Intensity

The range is calculated by subtracting the minimum pixel intensity from maximum pixel intensity. The number pixels in each bin (bin size) is calculated by diving the range by number of bins. Once the bin size is calculated n bins are created and then the pixels of the image are checked to find out the bin they belong to. And accordingly, the count in the bin is incremented.

Example 1:

Minimum pixel intensity: 43
Maximum pixel intensity: 221
Number of bins: 10

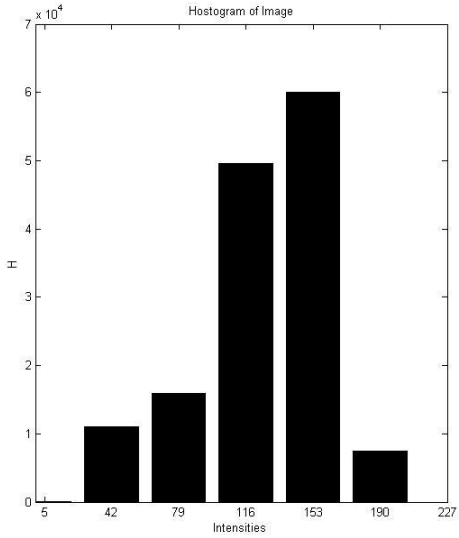


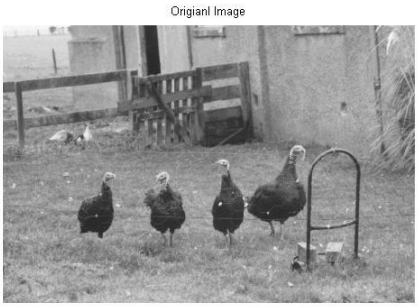


Analysis:
As we can see the input image has very few pixels with high intensity values(white) and majority pixels have low to medium intensities (black to gray). Therefore, the histogram is dense in the middle and sparse towards right.

Example 2:

Minimum pixel intensity: 5
Maximum pixel intensity: 222
Number of bins: 6





Analysis:
This image has very few pixels with low intensities (that are the turkeys). Majority of the pixels comprising of the background are mid-range to high intensities. Hence, we can see the histogram trend increasing from left to right. Also, there are very pixels which have the highest intensities (around 255), therefore sudden drop in the last bin.

CODE: histIm.m

2. Connected Components Analysis

As described in the requirements, the input image is first subjected to threshold and a binary image is obtained. For the binary image, it is assumed that the pixels are part of background or foreground and the connected components in the foreground are to be found. The binary image is given to the flood fill algorithm to label the connected components.
Input to the function:

- Input Image
- Threshold 1
- Threshold 2
- Label Value

Algorithm:

i. Input image is converted to grayscale.
ii. Depending upon the 2 thresholds received the image pixels are labeled as ‘0’ or ‘1’
iii. Flood fill is called for this binary image until all the pixels are labeled. Inputs to the FloodFill function are as follows:

- Binary input image
- Output image initially all zeros (used to mark the output label and used to check if pixel is visited or no)
- Seed value
- Label value

iv. The floodFill function checks the 4 connected neighbors of the seed point and continues to check the consecutive neighbors until all the connected pixels of a component are labelled. A queue is maintained for the neighborhood.

v. The label value is incremented so that each component has a different label value.

Example 1:
Label value: 3
First threshold value "thold1": 110
Second threshold value "thold2": 220

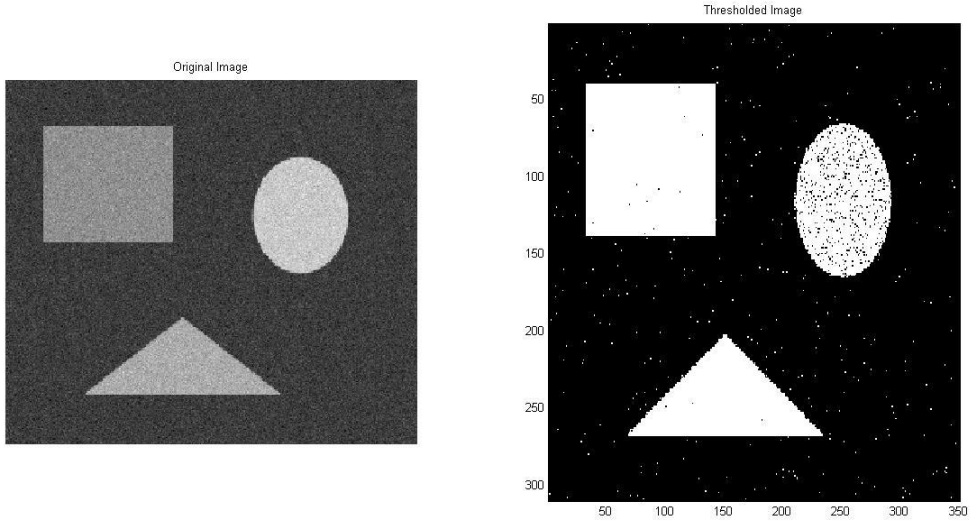


Figure 1

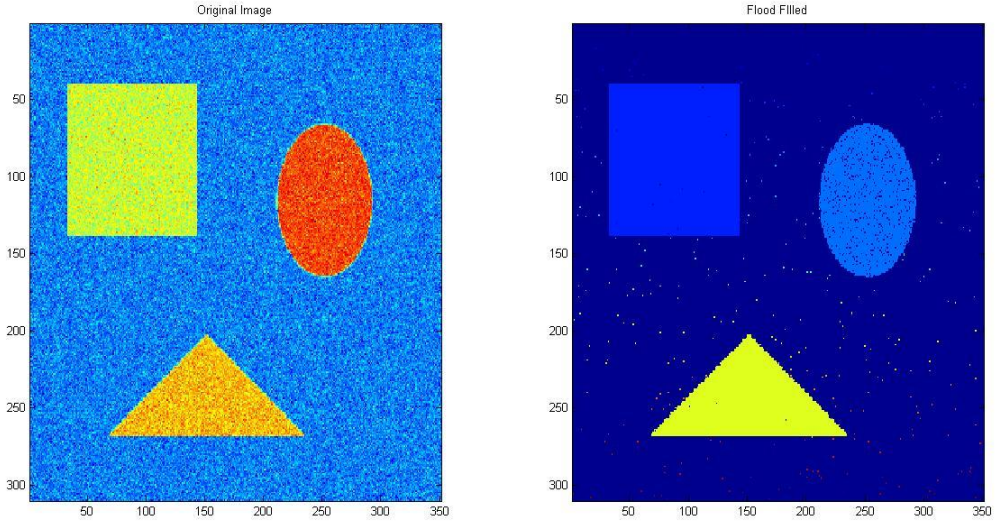


Figure 2

As seen from figure 1, the input image has lots of granular noise, which is partly eliminated using thresholding. When the thresholded image is given to flood fill all the connected components are labeled and shapes are visible clearly. Colours are added to give an idea regarding the different labels assigned.

Example 2:
Label value: 3
First threshold value "thold1": 45
Second threshold value "thold2": 65

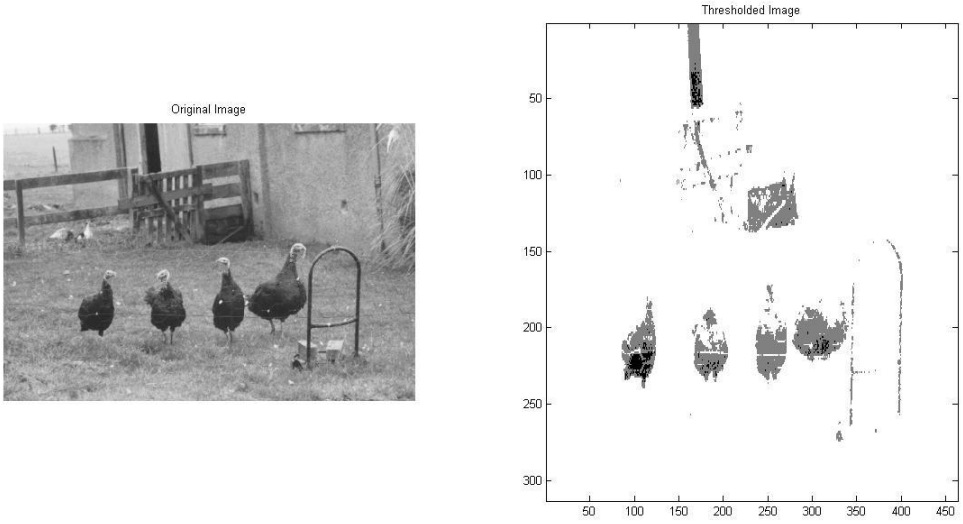


Figure 1

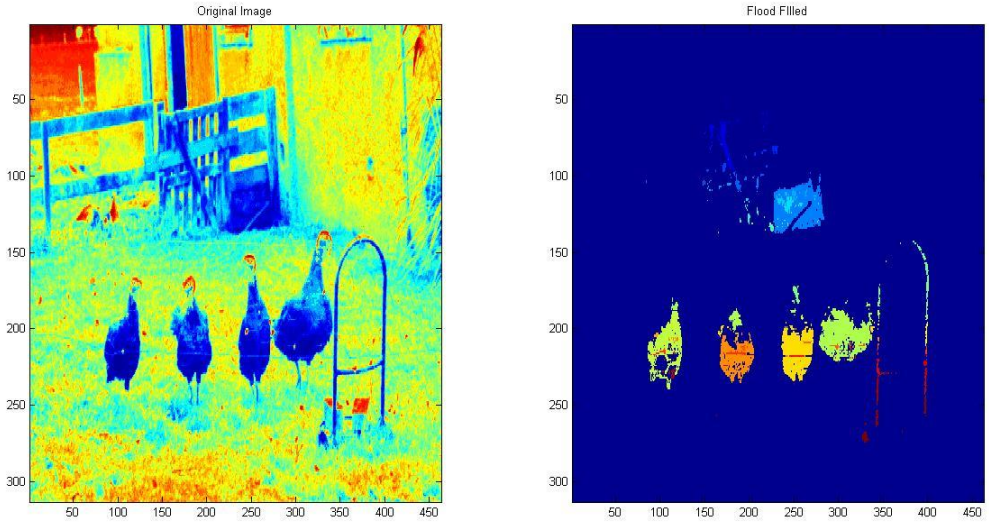


Figure 2

The background pixels are segmented from the foreground and background using thresholding. Some objects in the background have very similar intensities level as turkeys. Hence while thresholding they also appear in the image. The thresholded image is given to the floodfill routine where the components are labelled.

Example3:
Label value: 3
First threshold value "thold1": 110
Second threshold value "thold2" : 220

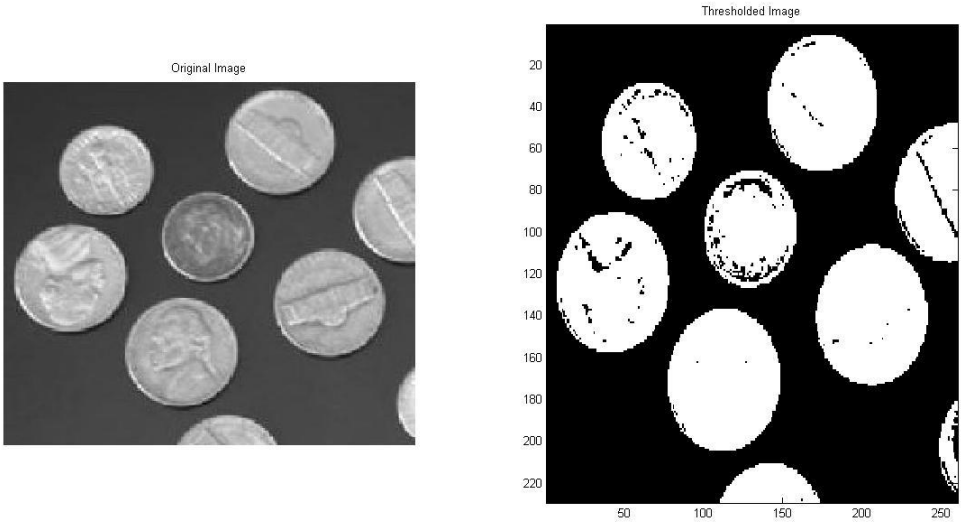


Figure 1

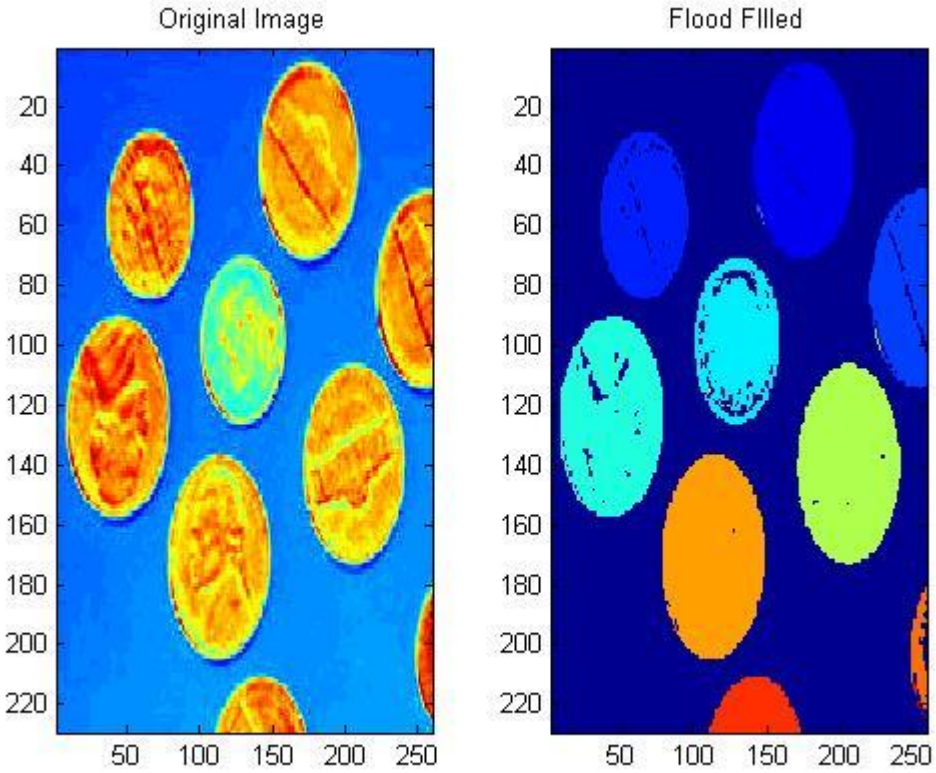


Figure 2

CODE: connectcomp.m and floodfilltp.m

3. Topological Denoising

As described in the requirements the input image is subjected to thresholding and then connected component analysis is done to identify the components. Considering the area/ number of pixels of the components, the threshold for noise is decided. Once we have identified the components which are smaller than the noise threshold, we need to find the all the neighboring labels for that component. The label of the noise component is replaced with the label of the component which shares the maximum boundary with the noise component.

Inputs to the function

- Input image
- Threshold 1
- Threshold 2

Algorithm:

i. Input image is converted to grayscale
ii. The grayscale image is thresholded
iii. The thresholded image is passed to the floodfill function which returns the labelled image
iv. While the floodfill is carried out, a list is maintained which tells us the number of pixels in each component and the seed values (inputs to the floodfill are same as mentioned in the previous section)
v. The list of seed values and number of pixels list is searched for the noise components by filtering using the number of pixels as criterion
vi. The seed values of the noise component and their label is passed to the denoising routine
Inputs to the denoising function:
    - Floodfill Image
    - Output image which is initially equal to the floodfill image
    - Visited image of all zeros
    - Seed values

vii. The denoising routine checks for all the neighbors of the noise component and maintains a list of labels
viii. This list is scanned and the label that occurs maximum number of times is found and that label is applied to the noise component,

Further additions to the system:

i. Before applying topological denoising, we could apply local spatial filter such as median filters to eliminate the noise.
ii. The noise in gray scale images contribute to the high frequency range in the frequency domain. Therefore, to eliminate noise low pass filtering in the frequency domain could be performed first prior to topological denoising.
iii. In this project, the threshold levels are given as inputs, instead Otsu's thresholding could be used for better thresholding.

Example 1:

First threshold value "thold1": 50
Second threshold value "thold2" :100
Label value: 3

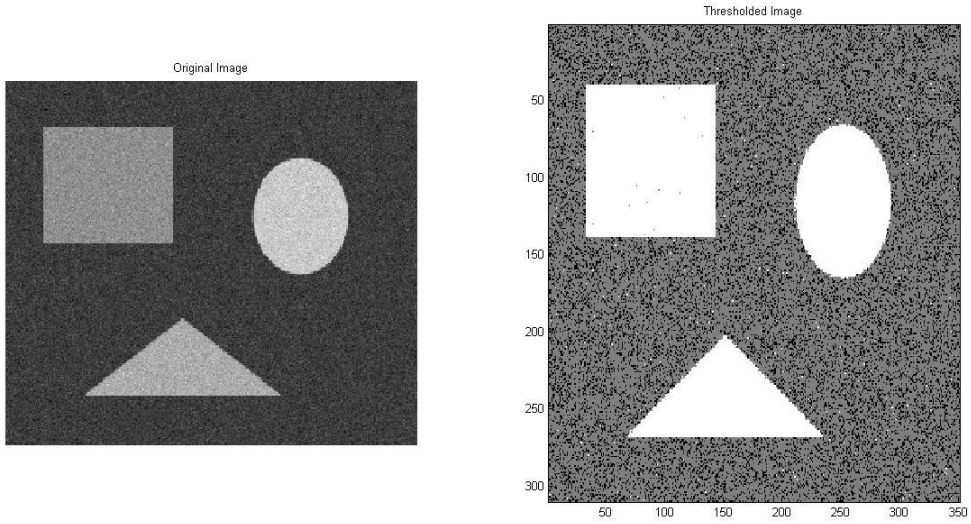


Figure 1

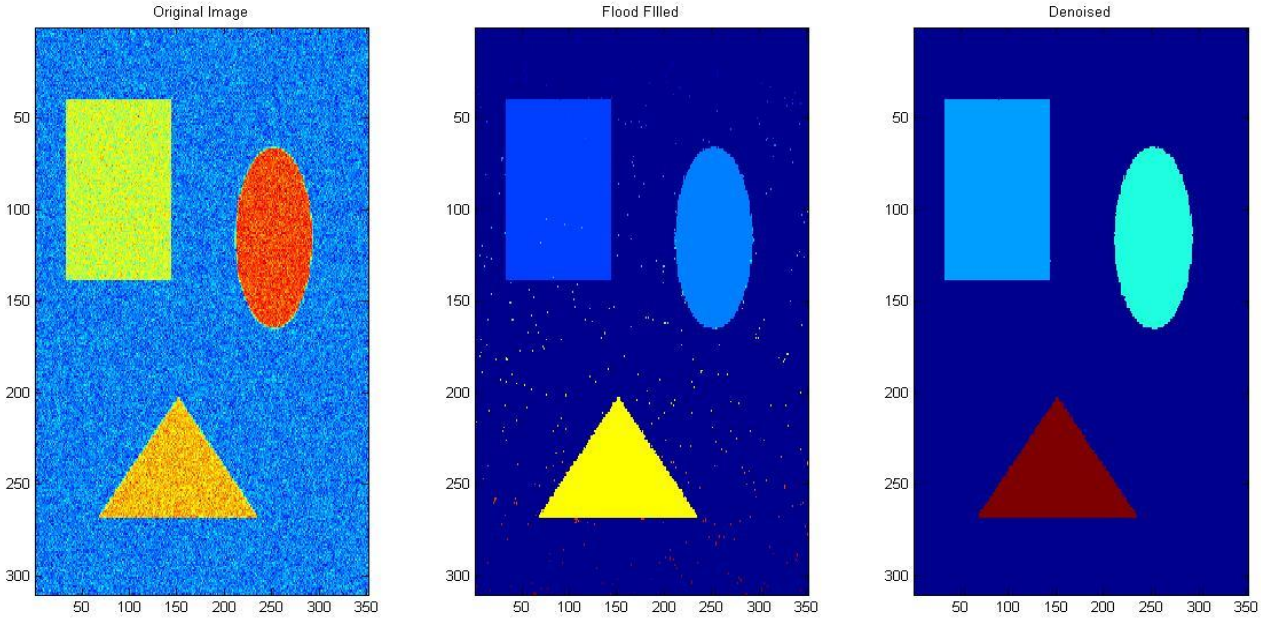


Figure 2

Figure 1 shows the input image and the thresholded image. As seen thresholding has eliminated the noise to a certain extent but not completely. The noise is granular and hence thresholding is not able to eliminate it. To solve this problem, we first perform the connected component analysis using the flood fill algorithm and then with denoising the noise is removed.

Example 2:

First threshold value "thold1": 20
Second threshold value "thold2": 60
Label value: 3

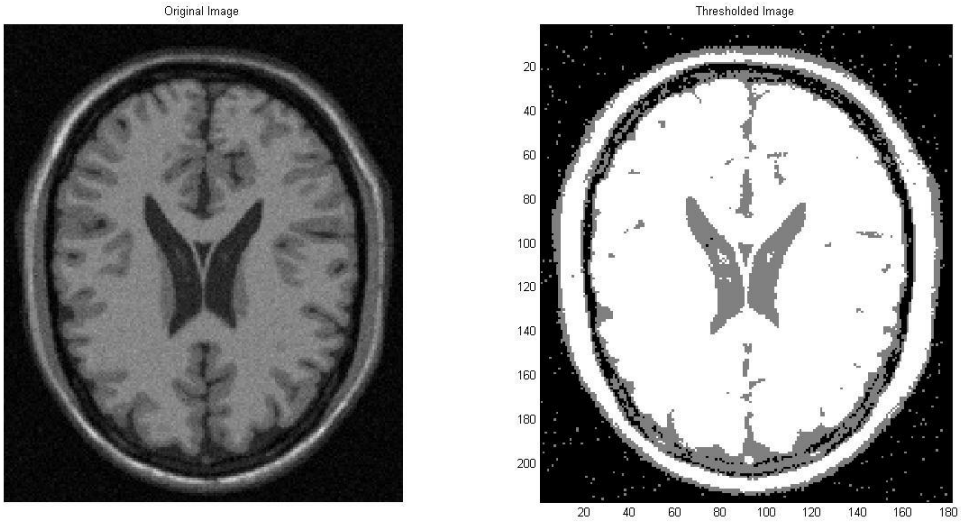


Figure 1

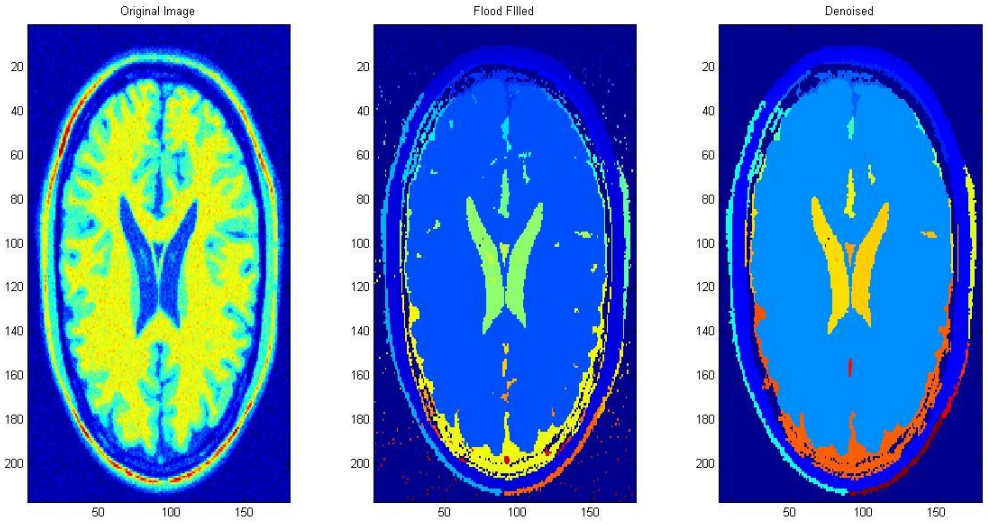


Figure 2

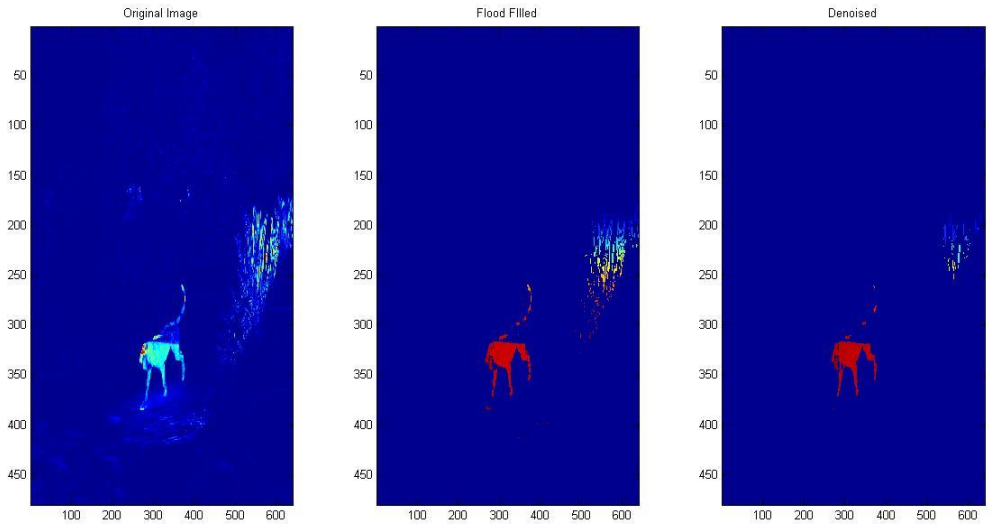
Code : topodenoising.m , floodfilltp.m and denoising.m

4. Motion Detection

Using the functions written above a function to detect motion has been written. In this function, two images are subtracted to find out the moving object. Topological denoising is performed to identify the moving object clearly. In this function 'imabsdiff' is used to find the difference between the two input images. This function is used because the object which is in motion has very close pixel intensity as that of the background hence, taking the absolute difference instead of the normal difference gives us better results.

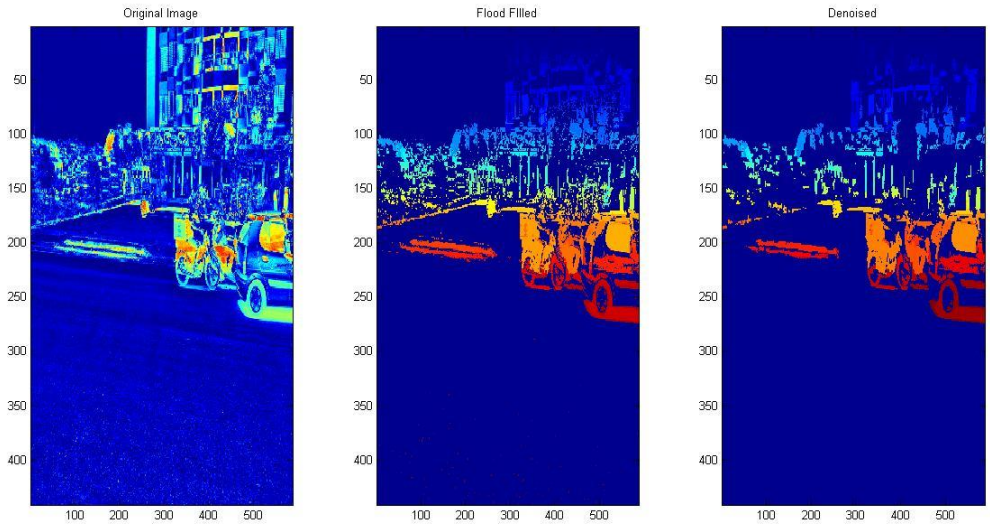
Example 1:

First threshold value "thold": 50
Second threshold value "thold2": 100
Label value: 3



Example 2:

First threshold value "thold": 50
Second threshold value "thold2": 100
Label value: 3



Challenges faced and possible solutions:

a. In the first example, it is difficult to identify the dog distinctly as the pixel intensity of the dog and the background are very close. This has happened as the dog is under the shadow of the tree.
b. Simple subtraction of the two images did not give the correct result, hence absolute values of the two images is taken.
c. Another issue could be lighting condition. As seen in the Example 2, the lighting condition of the two images is different. Hence while taking the absolute values of the images we see that the background (which is the common part in both images) is still present after subtraction. Due to the varied lighting condition, the background pixel intensities are slightly different hence false positives are seen in final denoised image.

Resources:

1. https://in.mathworks.com/help/matlab/ref/imagesc.html
2. Matlab Product Help.