Signal and image processing: Assignment 8

Mariuta Nicolae 30-10-2014

Exercise 1.

- First of all, the threshold has to be chosen so that the objects can be segmented correctly. This can be done manually, but some applications may need automation of the processing and it will be problematic.
- Illumination may not be constant over the entire image, so it is possible that in some areas the background will have intensity below the threshold and that will make wrong segmentation.
- The number of applications for this method is limited, because it requires that the foreground objects to have almost the same intensity, and the background color to be very different. If there are areas where the difference between the intensities of the objects and the background are appropriate, then the shape of the objects will be altered. The method can also be problematic for a blurred image where the difference of intensity between objects and background is hard to distinguish.
- In some areas there it can be a relationship between pixels of the objects but because of the threshold, those will be segmented differently, as it is can be seen in the image 10.1 from the course book where for the less illuminated coin, the edges are segmented but the interior is not contiguous and it has dark areas. If the relationship between pixels could be considered, then the intensity-based segmentation will be more effective
- As a conclusion, in certain applications with constantly illuminated background and objects with almost same color intensities, the method can be very useful, especially that I think it has low computation time so it can be applies very fast

Exercise 2.

Region-based and edge-based segmentation can perform better because they may solve the source of problems for image-based segmentation:

- In case of that illumination is not constant, the difference between the intensities of pixels will be small, so the region-based segmentation will still consider the pixels with different intensity to be within same object, and the edges can still be detected because there it will still be a big difference between regions inside objects and regions outside the objects. If, as in the case of the coins where there is a darker coin, if edges surrounding the coin will be detected properly, then all the inside of the coin will be segmented so the object will be identified properly
- Both methods are using relationships between pixels, not just check if their values as in case of intensity-based segmentation. The region-based segmentation is using difference between pixels intensities in an area and the edge-based segmentation algorithms use integrals which consider all pixel intensities in the image

- In case of region-based and intensity-based segmentations there are more parameters that can be modified to adapt the application to obtain good results in certain conditions: For the region-based segmentation it can be considered: standard deviation over an area, the difference between pixel intensities. In case of edge detection there are several methods with different parameters (standard deviations for Gaussian filters) which can be adapted to the application requirements.

However, both methods also have some disadvantages and can, sometimes work worse than intensity-based segmentation:

- A very blurred image can be problematic for both methods. The values of pixels will vary very slow so region-based method may no longer detect the edge of objects. The edge-detection can also have very bad results because it will not be same effective in finding areas where there is the limit between foreground and background. In this case, image-based segmentation may still locate the objects (not detect the perfect shapes of them) because at some point the pixels will go below the threshold.
- First of all, both methods have longer computation times. Edge-based method is usually using the FFT transform, integrals and matrix multiplications and the region-based segmentation has a lot of comparisons to make. The image-based segmentation is just checking once all the pixels in image and compare them to a certain value, which is a lot faster and easier to do.

Exercise 3.

- For region-growing it is very important where the initial seeds are placed. If they are placed correctly, the results can be better than for the quadtree decomposition because it is better for detecting objects no matter what shape they have, while quadtree decomposition is better for segmenting square-shaped regions
- The downside of region growing is the seeds placing. In case of doing it manually, the results are good but it can be hard to find automatically where to place the initial points.
- There can be used other shapes for quadtree also but however, the shape of partitioning has strong influence on the results
- region growing algorithm is simpler and faster

Exercise 4.

For the coding in Matlab I wrote a code which loads the image, turns it into grey-scale and then applies the edge detection using the *edge* function with different parameters

```
I = imread('concordaerial.png');
I = rgb2gray(I);
J = imnoise(I, 'salt & pepper', 0.01);

Jf = medfilt2(J,[150,150]);

k = fspecial('gaussian',[150,150],15);
Jfg = imfilter(J,k);

BW = edge(I,'sobel');

imshow(BW);
```

Figure 1. Code for loading the concordaerial image and apply edge detection

Before applying the edge detection, I have also written code to apply noise (salt and pepper) to the image and also code for filtering the image (using median and gaussian). I just used image in different stages to apply the edge detection and display it.

Figure 2. Concord aerial image that is used for edge detection and red cross over the area that I zoomed the mostly

Figure 3 Zoomed area in the result for Sobel filter method

In case of applying the Sobel filter for edge detection, on the entire image some streets are a little visible but when I zoomed the area where there is red cross in the grey-scale image, the streed edges and buildings are hard to distinguish.

Figure 4 Zoomed area in the result for the LoG approach (same area with the red sign)

For using the LoG filter the streets edges are more visible and also the edges surrounding the large buildings are also present. In comparison to Sobel approach, the results seem to be improved, as a lot more edges are visible. Moreover, most of the edges form a closed shape so the segmented some typical shapes may be detected as some specific objects.

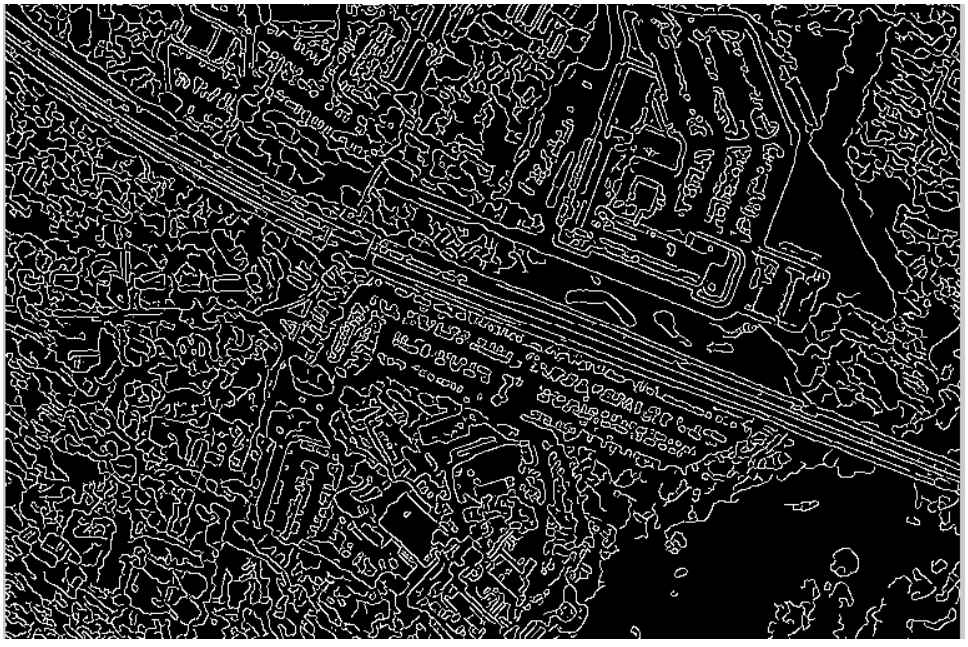

Figure 5 Zoomed area in the result for the Canny filter edge detection (also area of the red sign)

For using the Canny filter, the results are significantly improved. The street between the buildings is even easier to distinguish. The edges surrounding the buildings are also having better shape, less noisy than the edges detected using the LoG filter.

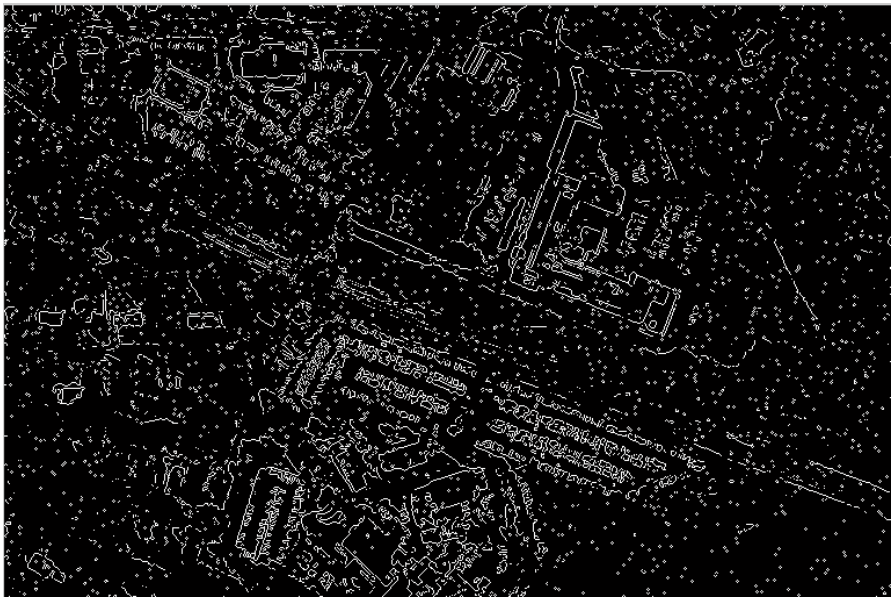

Figure 6. Edge detection using the Sobel filter after applying salt and pepper noise 0.01 to the image

Figure 7 Edge detection using the Sobel filter after applying salt and pepper noise 0.05 to the image

In case of adding salt and pepper noise with value 0.01, after using the Sobel filter some large buildings and structures are still visible but most of the edges have vanished: the street is no longer visible and it seems that there a lot of random points that appeared. For value 0.05 the

building edges are almost vanished and for higher values the result has just some random points, without any visible edges.

Figure 8 Edge detection using the Sobel filter after applying salt and pepper noise 0.01 to the image and then [150, 150] median filter

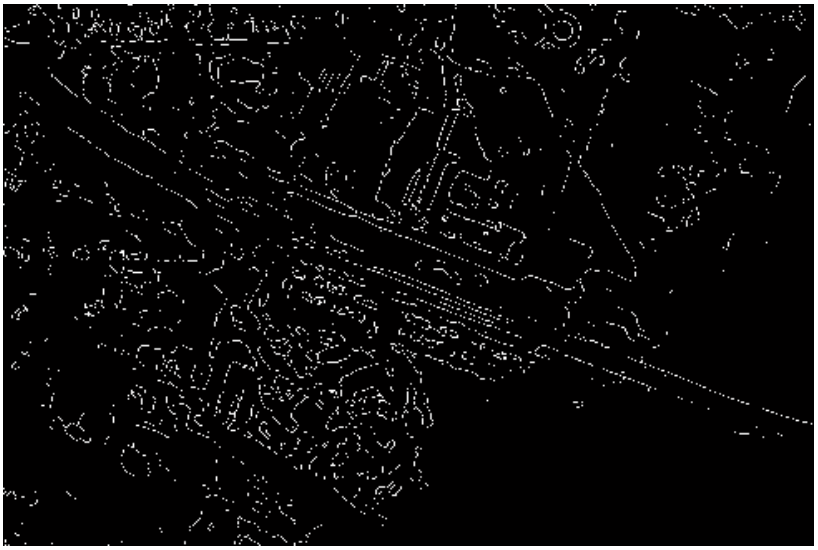

Figure 9 Edge detection using the Sobel filter after applying salt and pepper noise 0.05 to the image and then [10, 10] median filter

I have applied median filter: first with large size and results were very bad, almost all the white pixels are vanished because in a large matrix, but for a smaller window size of filter [10,10] the results are significantly better. Some edges of streets are visible and also edges of the buildings are easier to distinguish.

Figure 10 Edge detection using the Sobel filter after applying salt and pepper noise 0.05 to the image and then [15, 15] gaussian filter with standard deviation 5

For using the Gaussian filter results are worse for Sobel filtering. Most edges are more interrupted. It looks like median filter was a lot more useful in for edge detection using Sobel filtering if image is affected by salt and pepper noise. That happens because median filter is usually very good for removing this type of noise.

Figure 11 Edge detection using the LoG filter after applying salt and pepper noise 0.05 to the image

The LoG filter seemed to be more robust to salt and pepper noise. For 0.05 value of noise (chance that value of a pixel to be changed to 0 or 255) the street and buildings edges are still visible. The edges are almost vanishing for value 0.1 of noise.

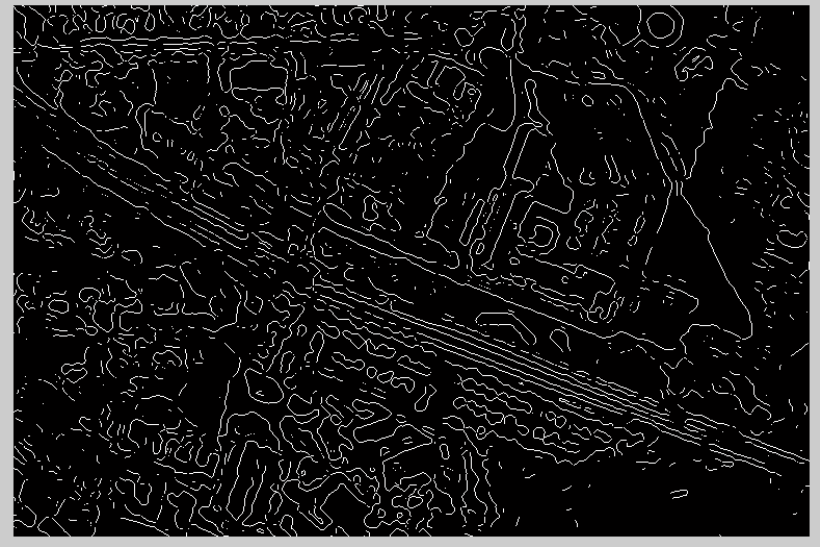

Figure 12 Edge detection using the LoG filter after applying salt and pepper noise 0.05 to the image and then [10, 10] median filter

Figure 13 Edge detection using the LoG filter after applying salt and pepper noise 0.2 to the image and then [10, 10] median filter

Figure 14 Edge detection using the LoG filter after applying salt and pepper noise 0.05 to the image and then [15, 15] gaussian filter with standard deviation 5

For LoG filtering, the median filter also has proved to be the best for detecting edges of image affected by salt and pepper noise. Some edges are still visible for value 0.2 of noise (20% of pixel values changed to 0 or 255). Small size (15x15) for median filter was also the best for improving the result.

Once again, the Gaussian filter was very bad for helping the edge detection, even for small amounts of noise.

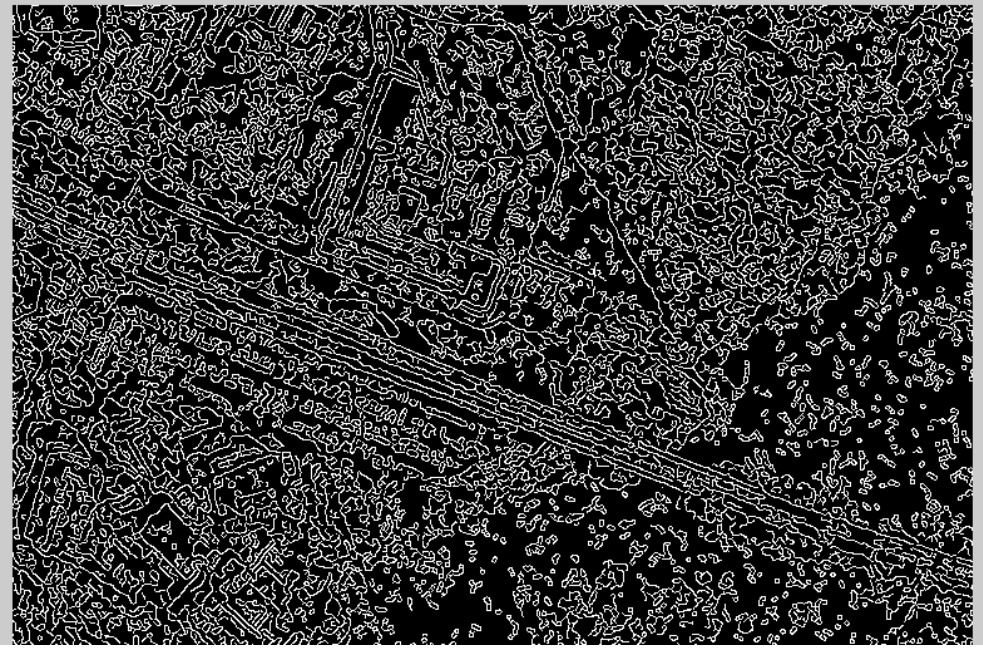

Figure 15 Edge detection using the Canny filter after applying salt and pepper noise 0.05 to the image

In case of Canny filter, the robustness for Salt and Pepper noise is lower than for the LoG approach. Edges of streets are still visible but the edges of buildings have almost disappeared for 0.05 noise.

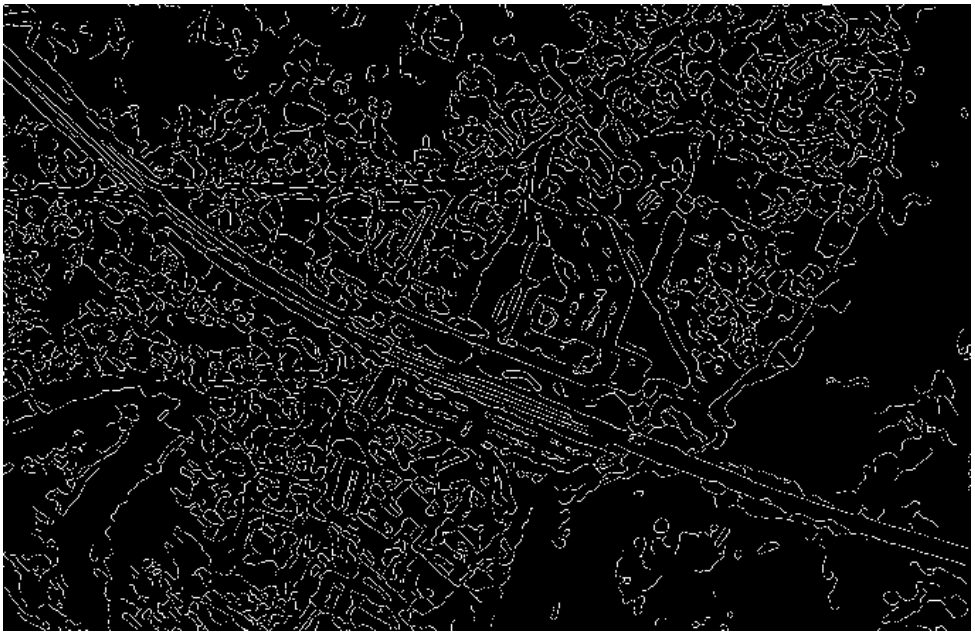

Figure 16 Edge detection using the Canny filter after applying salt and pepper noise 0.05 to the image and then [10, 10] median filter

Figure 17 Edge detection using the Canny filter after applying salt and pepper noise 0.05 to the image and then [10, 10] guassian filter with standard deviation 4

In case of filtering the image before applying the edge detection, in this case the Gaussian filter has proven to be better in obtain results similar to those before making the noise.

As a conclusion, the LoG method seemed to be the most robust to noise and the Sobel filter the worst for edge detection in case of noisy image. Both LoG and Sobel approaches worked better if noisy image was filtered using the median filter, while for the Canny method the Gaussian filter obtained better results. But, for images that are not affected by Salt and Pepper Noise, the Canny method was the best.

Exercise 5.

As for the coding, I have mostly followed the example in snakes test file. At first, the parameters (alpha,betta, gamma,sigma) are set so it will be easier to change them. Then, I have loaded the canoe imag, took the first 2 rgb channels (red and green) and calculated absolute difference between them. That is how I got the best results because the boat and its occupants have red color areas. The trees also have a lot of red pixels so by doing these substraction I have removed the leaves that were appearing in the power image and were altering the results.

In the next stage, the image is blurred by convolving with Gaussian and power image G is calculated. The multiplication by 200,000 is done so the power of the image to be enough high to be possible to detect the edges.

```
alpha = 100;
beta = 17;
gamma = 1200;
sigma = 9.5;
N = 200;

[Im,map] = imread('canoe.tif');
Irgb = ind2rgb(Im,map);

I1 = Irgb(:,:,1);
I2 = Irgb(:,:,2);
I = abs(I2-I1);

k = fspecial('gaussian',[sigma*2,sigma*2],sigma);
Is = imfilter(I,k,'conv');
[Isc,Isr] = gradient(Is);
G = 200000*sigma^2*(Isc.^2+Isr.^2);

% Make an initial curve
t = linspace(0,2*pi,N)';
t = t(1:end-1);
x = [size(I,1)/2+size(I,1)*5/16*cos(t), size(I,2)/2+size(I,2)*5/16*sin(t)];

% Setup the figure
f = figure(1);
clf
subplot(1,2,1)
imagesc(I);
colormap(gray);
title('Original');
subplot(1,2,2)
imagesc(G);
colormap(gray);
title('Image Energy');
set(f,'DoubleBuffer','on')
h1 = -1;
h2 = -1;
for t = 1:3000
  x = snake(x,alpha,beta,G,gamma,15);

  if h1 > -1
    delete(h1);
  end
  if h2 > -1
    delete(h2);
  end
  subplot(1,2,1);
  hold on;
  h1 = plot([x(:,2);x(1,2)],[x(:,1);x(1,1)],'linewidth',sigma);
  hold off;
  subplot(1,2,2);
  hold on;
  h2 = plot([x(:,2);x(1,2)],[x(:,1);x(1,1)],'linewidth',sigma);
  hold off;
  drawnow;
end
```

Figure 48 Code for detecting the canoe edges using the snakes

The curve is then created(circle)

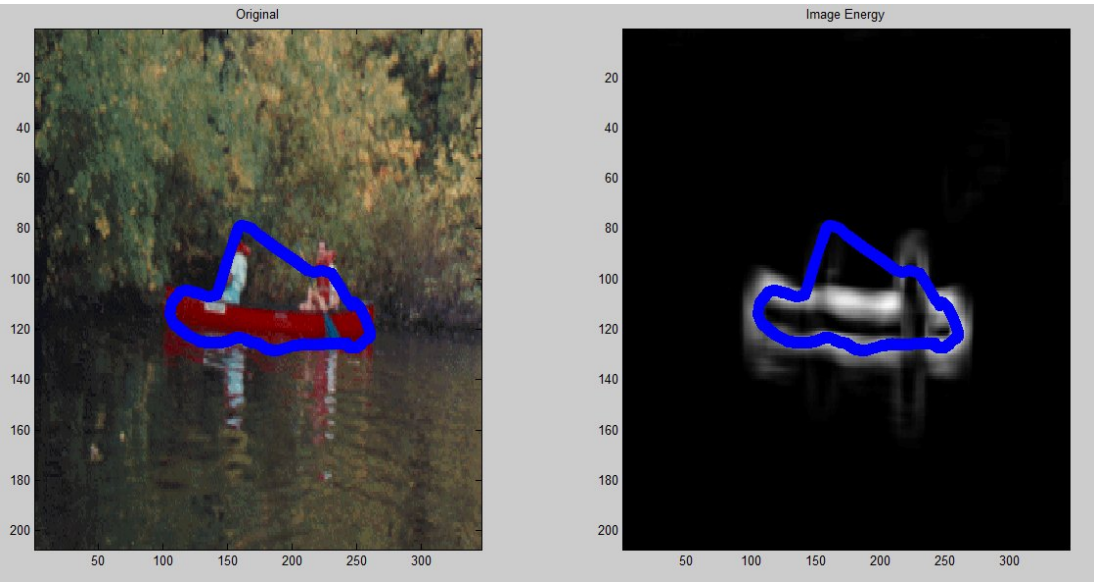

Figure 49 Results for edge detection using the code above

By running the code with exactly the same parameters that are in Fig. 49 the edges of the boat are detected properly and also the men in boat are slightly detected, only problem is that snake does not fall in the space between the 2 men in the boat, but as it was told at the course on Wednesday that is hard to avoid.

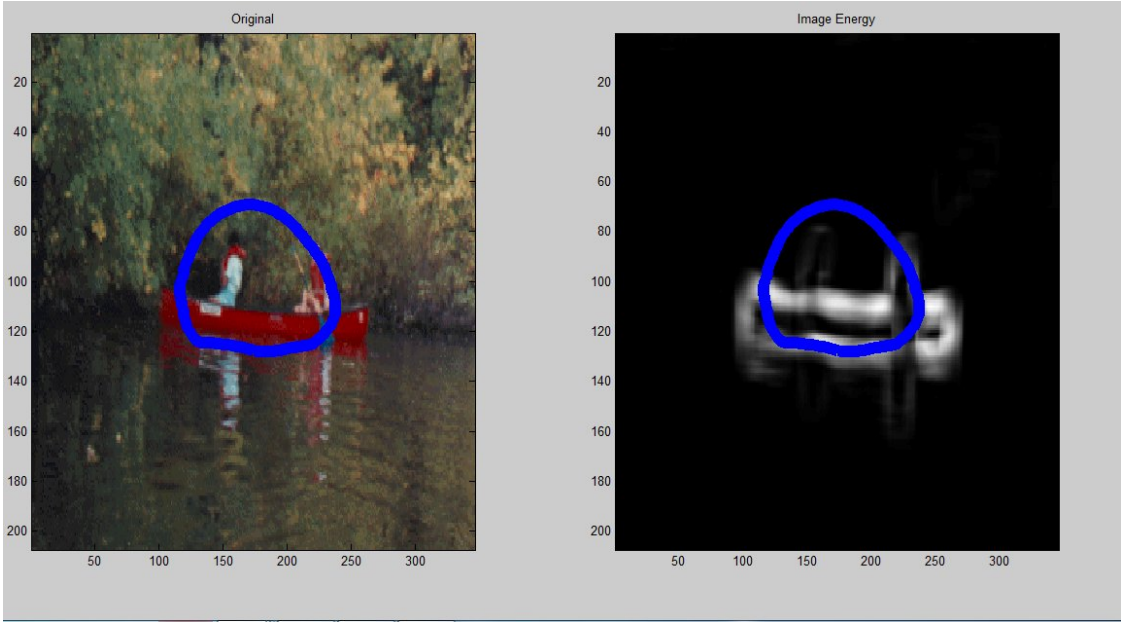

Figure 50 Edge detection when multiply G with 2000 instead of 200000

As a results for reducing the multiplication of the power image, the energy of the edges is no longer strong enough so the snake does no longer respond to the edges.

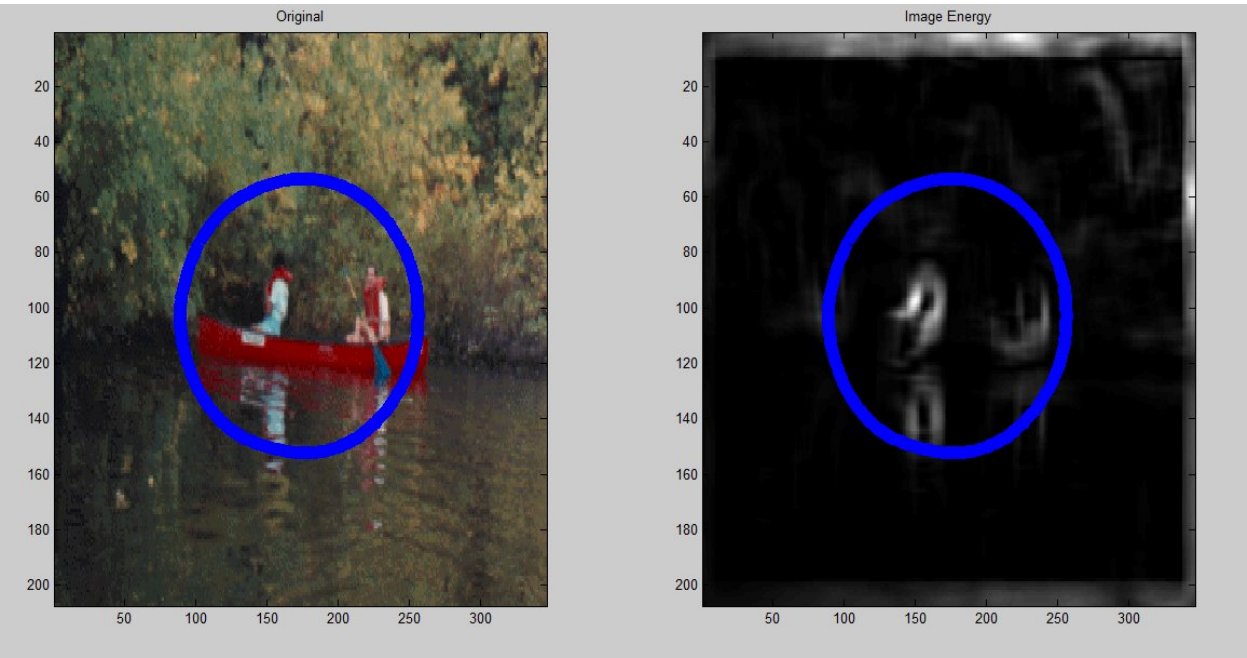

Figure 51 Power image for using the rgb2gray function instead of channel substraction

In the image above I have shown what happens in case I do not use that substraction between red and green channels for obtaining the image to work with. In case of using *rgb2gray* function for making conversion to greyscale then a lot of shapes appear in the background which have a negative impact on the results.

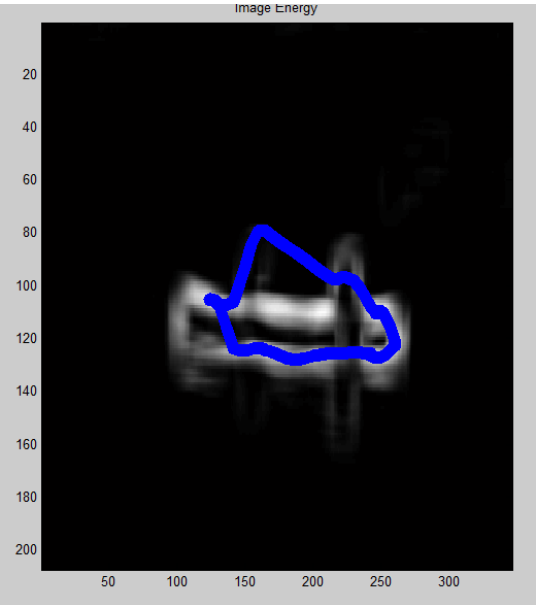

Figure 52 Result for changing beta value to 30

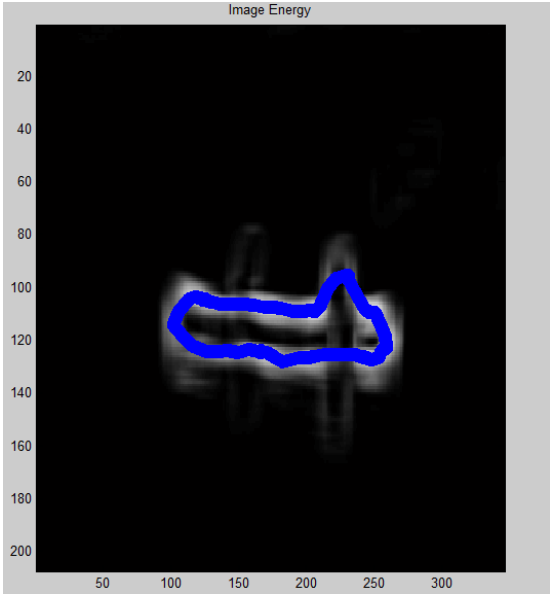

Figure 53 Results for changing sigma value to 10

I have used the parameters in the code after doing many attempts. In case of changing them, results are no longer being so good. For example, at higher values of parameter beta the buttom-left part of the boat edge is deformed because that area is not that clear: there is strong reflection of the boat in the water so it is a bit confusing.

In case of using higher value for parameter sigma(standard deviation of Gaussian filter) the edges are even more blurred so the man on the left side of boat is not having that visible edges. If the image is less blurred, then the reflection of men in the water are also being detected by the snake.