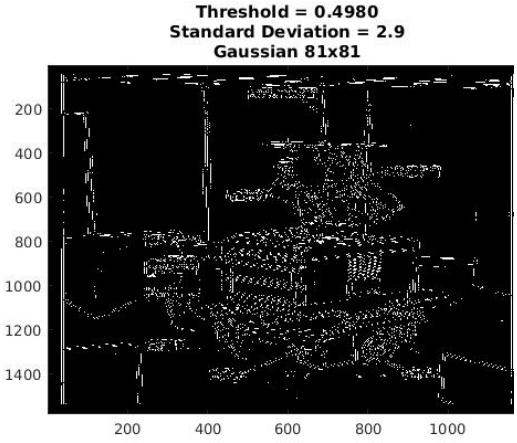
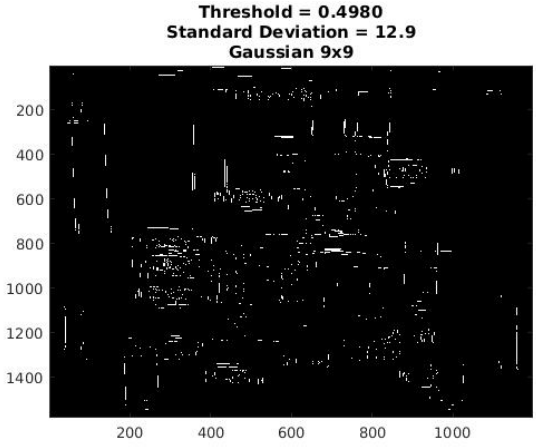
Computational Vision - Assessed Assignment

Antons Lebedjko ID:1702318

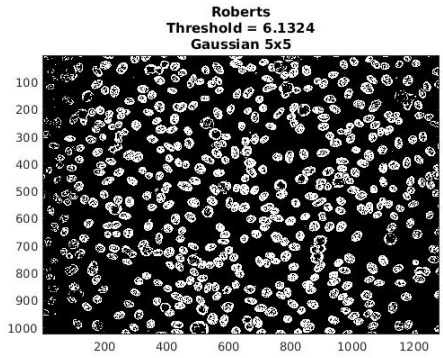
2.1. Laplacian of Gaussian

The creation of the Laplacian of Gaussian filter consists from 2 steps. The first step is to first of all smooth the picture(e.g., using a Gaussian filter) to reduce noise. After the picture is smoothed, we can apply the Laplacian filter which is a derivative filter used to find areas of rapid change(edges) in images. This is called the Laplacian of Gaussian filter. I made a MATLAB function which is called “laplacian_of_gaussian” and it takes 3 parameters such as: 1) Dimensions of gaussian filter (the greater are the gaussian filter dimensions, the more blurred will be the picture), 2) Laplacian filter 3) Standard deviation(tells us how measurements for a group are spread out from the average). The way my function works, is: it creates gaussian filter. After that it convolves gaussian filter with Laplacian to create Laplacian of Gaussian filter. If we would like to apply this Laplacian of Gaussian filter on an image, we need to convolve this image with Laplacian of Gaussian mask, apply zero-crossing after that (to see for which pixel values the intensity changes from negative to positive and vice versa) and finally threshold it (how big the sign change should be). I am using some MATLAB build in functions to do zero crossing and to find the most appropriate threshold which we should use. The function which chooses the most appropriate threshold is “multithresh”. The way “multithresh” works is, it uses the Otsu’s method which assumes that the image to be thresholded contains two classes of pixels or bi-modal histogram (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. MULTITHRESH can accept another argument that decides how many such classes of pixels are needed, so that you can segment the image over multiple levels. After running my code, we get the following results:





From the results above, we can see, that if we choose the standard deviation, which will be to big, we can loose to much information. Same will happen if we will select a threshold which is to big, or Gaussian dimensions which will be to big. We also will not get the best result, if we underestimate the values. But in general, we can see, that Laplacian of Gaussian gives a good results on a shakey image, in case if we select the optimised values.



2.2. Cell Detection

Aim: to test the efficiency of various edge detectors(Roberts, Sobel, first order Gaussian, Laplacian and Laplacian of Gaussian)

Method: 1) Roberts is a first derivative function, so to apply it, what I did, was: I wrote a function that takes 2 parameters, which are the image and the Gaussian filter. My function smooths the image using this Gaussian filter, to reduce a noise(Cell_after_gausian = conv2(image, gaussian_filter, 'same')). After that it convolves the smoothed image with the horizontal and

vertical Roberts masks separately. In the end it takes the absolute values of |Gx|+|Gy|, to get the approximation of the magnitude. To see the results, we need to call this function on an image and after that threshold everything. To find the most appropriate threshold value, I am again using the "multithresh" function, explained in a first part of this report. You can see the result of the cell 9343AM on your left. I am using the same Gaussian dimensions for all of the filters an images, to make a fair experiment.

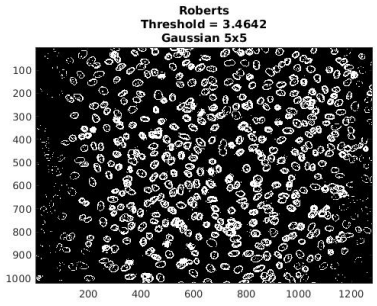
2) Sobel is also a first derivative function. To apply a Sobel, I followed the same principle as for Roberts, except that I am combining the two resulting arrays using Pythagoras theorem.

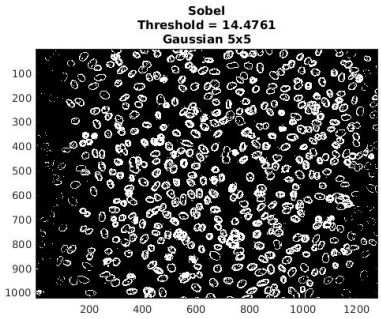
3) First order Gaussian is also a first derivative function. It is again very similar principal as for Robers and Sobel, with the difference that to get a vertical mask, we transpose our horizontal one. In the end I am taking the absolute values of |Gx|+|Gy| (it could be for example a Pythagoras theorem, but I decided to do it with absolute values).

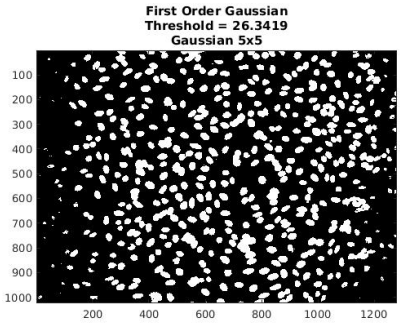
4) Laplacian is a second derivative function. I wrote again the same functions with image and Gaussian filter as a parameters. It smooths the image to reduce noise. After that it convolves the smoothed image with the laplacian mask. And in the end we calculate the absolute value of that image and return it. If we want to see the results, we should call this function, apply a zero crossing because it's a second derivative functions (to see for which pixel values the intensity changes from negative to positive and vice versa) and finally threshold it (how big the sign change should be).

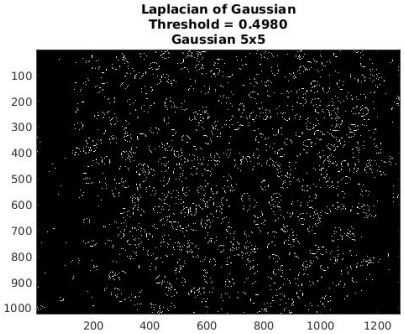
5) For Laplacian of Gaussian I am following exactly the same principle and using the same function as was described in exercise 1. After some experiments which I did manually on Cells images, I have decided to use standard deviation of 2.9, because it was giving the best results visually.

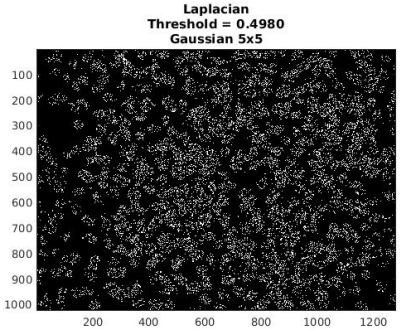
Pictures below after applying each edge detector on Cell_43590AM











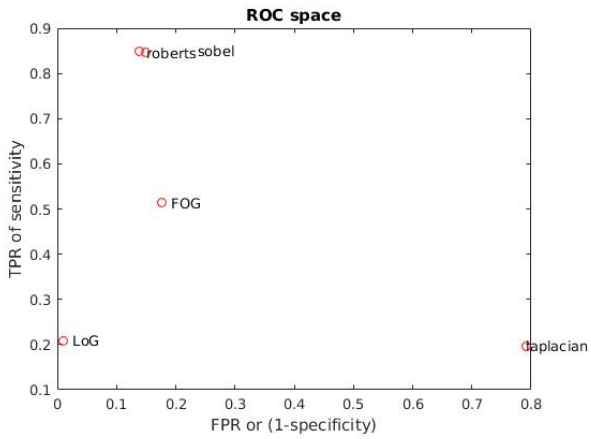
To evaluate the results, I am calculating the Sensitivity(TP/(TP+FN) and Specifity(TN/(TN+FP). I wrote my own function for it which iterates through each pixel of the image after edge detector has been applied and the testing image. It compares all the pixels between each other to calculate the amount of true positives, true negatives, false positives and false negatives to calculate the Sensitivity and Specificity in the end.

Results

Cell_9343AM	Roberts	Sobel	First order Gaussian	Laplacian	Laplacian of Gaussian(sd=2.9)
Threshold	6.1324	27.0098	46.2001	0.4980	0.4980
Gaussian Dimensions	5x5	5x5	5x5	5x5	5x5
Sensitivity	0.8489	0.8551	0.5433	0.2101	0.2140
Specificity	0.8454	0.8484	0.8322	0.9244	0.9849

Cell_10905JL	Roberts	Sobel	First order Gaussian	Laplacian	Laplacian of Gaussian(sd=2.9)
Threshold	7.7301	32.7424	67.6422	0.4980	0.4980
Gaussian Dimensions	5x5	5x5	5x5	5x5	5x5
Sensitivity	0.9376	0.9160	0.5548	0.1814	0.2380
Specificity	0.8305	0.8670	0.8128	0.9116	0.9927

Cell_43590AM	Roberts	Sobel	First order Gaussian	Laplacian	Laplacian of Gaussian(sd=2.9)
Threshold	3.4642	14.4761	26.3419	0.4980	0.4980
Gaussian Dimensions	5x5	5x5	5x5	5x5	5x5
Sensitivity	0.7498	0.7793	0.4443	0.1979	0.1726
Specificity	0.8746	0.8673	0.8230	0.9108	0.9901



If we now calculate the average of Sensitivity and Specificity for each edge detector from 3 images, we can do the ROC space (see the graph on a left).

Conclusion: our experiment showed that the best edge detectors in this case are roberts and sobel (see ROC spase graph on a left). I think that however Laplacian of Gaussian didn't show the best results, but it still could be improved, because it depends on a standard deviation and in this experiment I have chosen this standard deviation manually, so, we cannot guarantee that I have chosen the most appropriate value. Laplacian showed the worst result and First order Gaussian is on a third place. However, even if in this case we got, Roberts and Sobel as the best edge detectors for this case, we cannot say that these are the best edge detectors ever, because it very depends on images.

If you would like to see more pictures or run scripts or function, you can find everything here: https://github.com/antonlebedjko/Edge-Detection. Thank you!