Assignment report

Task1: Laplacian of Gaussian

To process the ‘Shakey’ image with the Laplacian of Gaussian filter, the first step I took was applying the Gaussian filter to remove noise. It uses a convolution (or mask) to scan every pixel in the image, and the weighted average gray-scale value of the pixels in neighborhood (determined by convolution or mask) is used to replace the center pixel’s value. Then, I applied the Laplacian of Gaussian filter to the ‘Shakey’ image, to reach edge detection on gray-scale image. The result shows as below, a 5x5 kernel was used here to approximate the laplacian. The Laplacian of Gaussian (LoG) operator is improved from the Laplacian operator, only one convolution of the image needs to be performed at runtime, because Laplacian operator is more sensitive to noise, it is used to determine whether edge pixels are considered to be bright or dark area of the image. And it is isotropic and can sharpen the boundaries and lines of any direction (without directionality), this is the biggest advantage of Laplacian operator from other algorithms. For Task 1, I wrote it as a function. To run it, the filter(*‘Gaus’*) and the input image(*‘img’*) should be loaded and read firstly in terminal, then the function with these two parameters can be run.

As shown in Fig.1, after applying LOG, a clearly smoothy image with the intensity changes in both horizontal and vertical directions has showed. Also the residual between ‘shakey.gif’ and the denoised result shows that Gaussian filter not only attenuate the noise but also blur the edge of the whole image. However, LoG is still capable of achieving the detection goal.

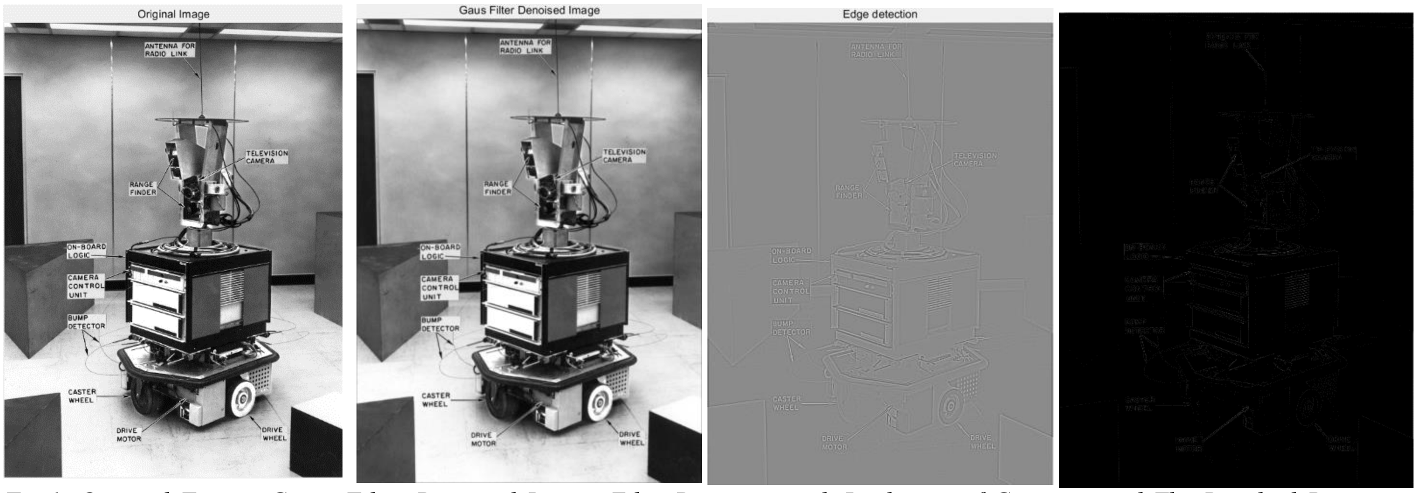


Fig.1. Original Figure, Gauss Filter Denoised Image, Edge Detection with Laplacian of Gaussian and The Residual Between Original Image and Denoised Image

Task2: Cell Detection

The edge of the image refers to the position where the gray-scale value changes sharply. In the process of image formation, due to the difference of brightness, texture, color, shadow and other physical factors, the gray-scale value of the image can change suddenly, thus forming the edge. Edges are quantified by examining the neighborhood of each pixel and quantifying the gray-scale variation, which is equivalent to the directional derivative of a continuous function or the difference of a discrete sequence in calculus.

For Task2, I firstly turned the RGB images to grey images, cause gray-scale image only has one color which is efficient to detect edge. Then I got the ground truth for each image. Next, I applied the following detectors to the images, Gaussian filter used to remove noise. Sobel, Roberts and First Order Gaussian use similar methods which all calculated convolution and magnitude first. Laplacian and Laplacian of Gaussian used zerocrossing based on convolution. All thresholds used here can be seen in Appendix.

1.Roberts: It is a first-order differential operator, From the actual effect of image processing, the calculation is simple and the edge location is accurate, but it is extremely sensitive to noise. It is suitable for image segmentation with obvious edges and less noise. the Roberts operator is a $2 * 2$ operator, which is accurate in edge positioning, so the boundary width of the segmentation result is

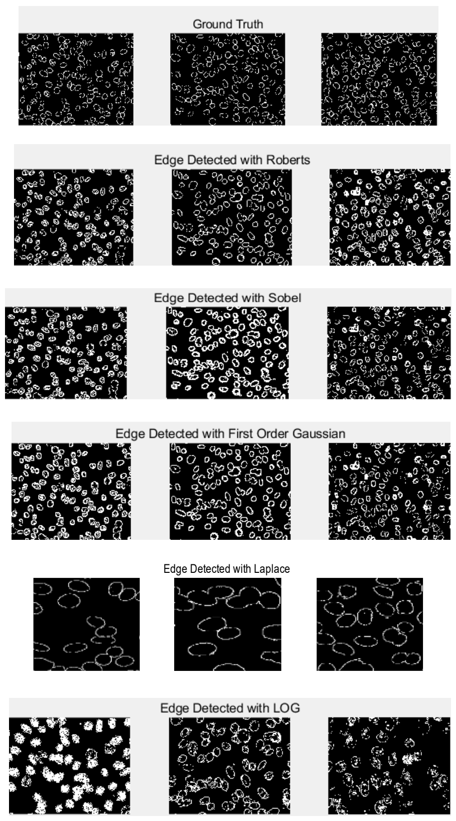

Ground Truth
Edge Detected with Roberts
Edge Detected with Sobel
Edge Detected with First Order Gaussian
Edge Detected with Laplace
Edge Detected with LOG

Fig.2. Edge detection with different operators with a bit enlargement

not so wide.

2.Sobel: The Sobel operator detection method has a good effect on image processing with gray-scale gradient and more noise, but it is not very accurate for edge location. It is a common edge detection method when the precision is not very high.

3.First order Gaussian: The advantage of Gaussian filtering is that it can eliminate Gaussian noise (a type of noise whose probability density function follows a Gaussian distribution (i.e., a normal distribution)). On my opinion, I think the first order Gaussian is a step of Canny method, but it is a horizontal convolution kernel and a vertical convolution kernel, which uses Gaussian blur and filter with a simple filter.

4.Laplacian: The Laplacian operator method is sensitive to noise, and it has no edge orientation information and cannot detect the orientation of edges, so it is rarely used to detect edges. Instead, it is used to determine whether edge pixels are considered as the bright or dark region of the image.

5.Laplacian of Gaussian: As in Task one, it first uses the Gaussian operator to smooth the image, suppress noise, and then uses the Laplacian operator on the smoothed image. The Laplacian gaussian is a second derivative operator that produces a steep zero-crossing at the edge. The Laplacian operator is isotropic and can sharpen the boundary and line in any direction. This is the biggest advantage of Laplacian from other algorithms.

In the case of less noise, the segmentation result is quite good. Sobel operator can process the image with gray gradient and more noise better. Laplacian operator is sensitive to noise, so from the segmentation result, we can see that there are some scattered edge pixels appeared. It can be proved that it is isotropic, that is, independent from the direction of the coordinate axis, and the gradient result does not change after the coordinate axis rotating. It is more accurate to position the edge. And for LoG, it performed better on image 2 than others.

Task3: Advanced Edge Detection

I used Canny edge detection algorithm to process Task3. Canny method implemented using images which have turned into gray-scale images and further converted into binary images based on thresholds. From lectures, this step was not included, but after experiments, I found it is quite useful here and can make this task easier. Binary image aims to set a single threshold, this can separate “interesting” pixels as fronter and leave the rest pixels as background. This further simplifies the gray-scale images, makes the information from images purer and the edge brightness changes are more obvious. And of course, this step is not necessary, but use it properly can make question easier.

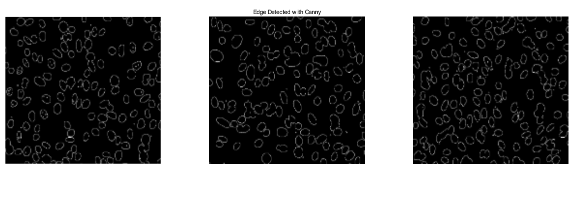
Roughly for Canny, the first step is to use a Gaussian filter to smooth the image, the second step is to calculate the image gradient magnitude and direction, the third step is applying non-maxima suppression, and finally the edge is extracted using double thresholds. For the suppression of non-maxima data, it can also be interpreted as the exclusion of the possibility that non-maxima data is an edge. The larger the element value in the gradient matrix of the image in the neighborhood is, the larger the gradient value of the point is. Combined with the gradient direction of the detection point, the approximate edge information can be located.

Non-maxima suppression has two characteristics:

1.The gradient at the current position is compared with the gradient on both sides of the gradient direction

2.The gradient is perpendicular to the edge

Also, Canny method is not easily disturbed by noise and can detect really weak edges. The advantage is that two different thresholds, T-high and T-low are used to detect strong edges and weak edges respectively, and weak edges are included in the output image only when weak edges and strong edges are connected. The Sobel operator does not make full use of the gradient direction of the edge, and the resulting binary image



is simply processed with a single threshold. but The Canny algorithm is improved based on these two points.

Fig.3 . Edge detection with Canny operator.

By comparing the experimental results, we can find that the Canny method works well on images, it highlights the edge points by using non-maxima suppression which mentioned before. Through the comparison of the results, I found that although the canny operator is easy to misjudge the noise as the boundary, the effect is the best.

Task4: result evaluation

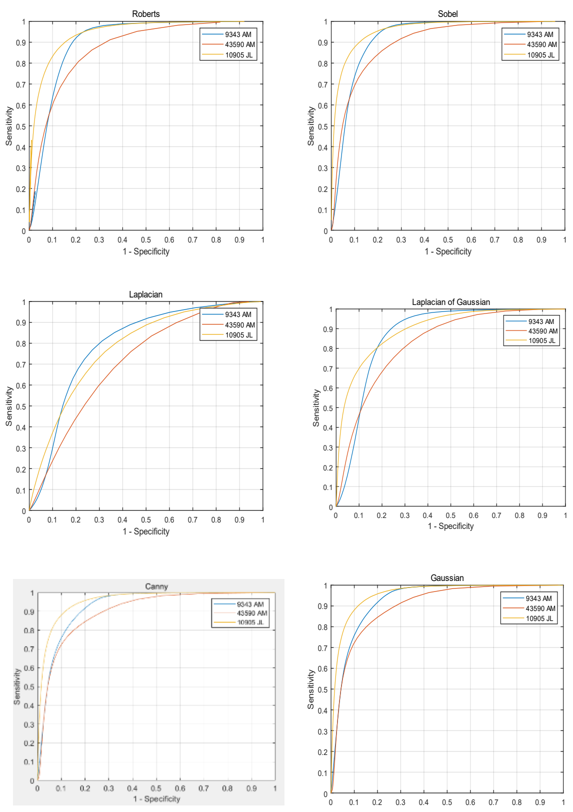


Fig .4, ROC of different edge detectors

The ROC curve (the value of AUC) is a stable indicator that reflects the quality of the model. AUC (Area Under Curve), is defined as the area under the ROC curve and generally ranges between 0.5 and 1. As a numerical value, a classifier with a larger AUC performs better. AUC = 1, a perfect detector; 0.5 < AUC < 1, better than random guessing; AUC = 0.5, The model has no detection value; AUC < 0.5, than random guessing. Roberts operator is a 2x2 operator, which is accurate in edge positioning, so the boundary width of the segmentation result is not as wide as that of Prewitt segmentation. In the case of less noise, the segmentation result is quite good.

As shown in the Fig.4, In this experiment, Roberts operator, Sobel operator, Laplacian operator, LOG operator, Canny operator and morphological gradient method are used as six edge detection algorithms. Each algorithm sets 8 thresholds at a certain interval and performs edge detection on the input . A total of 48 edge detection results are obtained. The candidate edge detection images were obtained by pixel statistics and ROC classification. The Chi-square test parameters between the candidate edge detection graph and the edge detection result atlas of each grade were calculated, and the maximum value was taken as the optimal edge grade, and the corresponding candidate edge detection graph could be used as the "ideal" edge detection reference graph (ground truth). After the synthesis of the edge reference graph, eight different thresholds were set as working points for an edge detection algorithm to be evaluated, and the detection sensitivity(True Positive Rate) and false alarm probability(1-specificity) of the corresponding edge detection results were calculated, and the RoC curve was generated by connecting them. Also considered the Negative Posivie Rate, both TPR and NPR were calculated after the confusion matrix.

Table 1, AUC of Different operators

operators	**Sobel**	**Roberts**	**Laplace**	**LoG**	**Canny**	**fog**
AUC	**0.831**	**0.842**	**0.803**	**0.939**	**0.865**	**0.875**

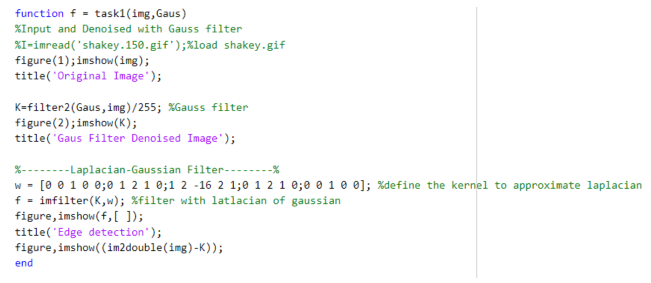
Finally, AUC are calculated to obtain the performance parameters of the edge detection algorithm. AUC indexes of Roberts operator, Sobel operator, Laplacian operator, LOG operator, Canny operator and FoG method are 0.831, 0.842, 0.803, 0.939, 0.825 and 0.875 respectively. It can be seen that among the six edge detection algorithms to be evaluated, LOG operator is outstanding for its strong anti-noise ability, the overall performance is the best, First Order Gaussian operator and Canny are the second and third which have small difference, the Sobel operator and Roberts operator are next, and the Laplacian operator's performance is the worst since its sensitivity towards noise.

A quantitative evaluation method of edge detection algorithm based on ROC curve does not need the direct edge reference image information, and can test the real input image to complete the performance evaluation of different edge detection algorithms.This has a good reference value for selecting the appropriate edge detection operator or even the appropriate correlation threshold for some application environment.

Appendix: CODE

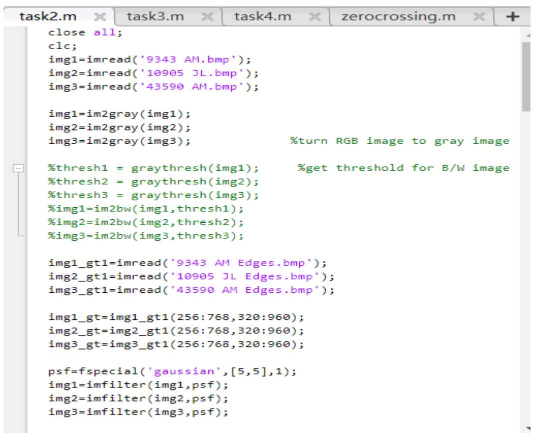
(multiple graphs in one task should read from left-column to right-column)

Task 1:

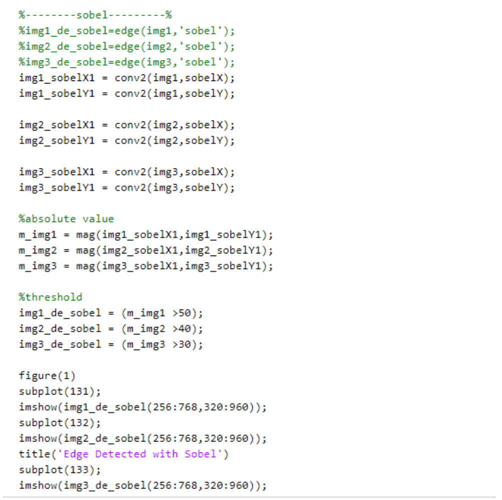


```
function f = task1(img,Gaus)
%Input and Denoised with Gauss filter
%I=imread('shakey.150.gif');%load shakey.gif
figure(1);imshow(img);
title('Original Image');

K=filter2(Gaus,img)/255; %Gauss filter
figure(2);imshow(K);
title('Gaus Filter Denoised Image');

%--------Laplacian-Gaussian Filter--------%
w = [0 0 1 0 0;0 1 2 1 0;1 2 -16 2 1;0 1 2 1 0;0 0 1 0 0]; %define the kernel to approximate laplacian
f = imfilter(K,w); %filter with latlacian of gaussian
figure,imshow(f,[ ]);
title('Edge detection');
figure,imshow((im2double(img)-K));
end
```

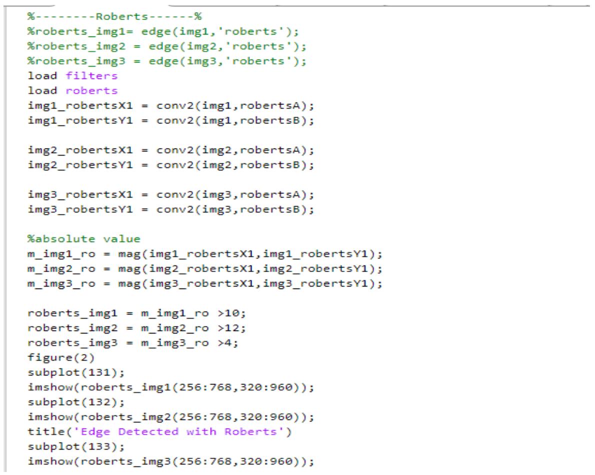
Task 2:



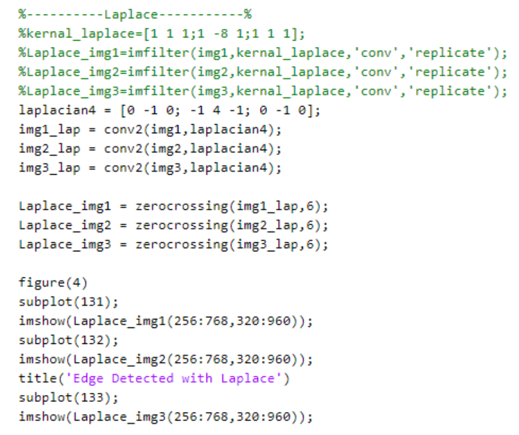
```
    close all;
    clc;
    img1=imread('9343 AM.bmp');
    img2=imread('10905 JL.bmp');
    img3=imread('43590 AM.bmp');

    img1=im2gray(img1);
    img2=im2gray(img2);
    img3=im2gray(img3);                 %turn RGB image to gray image

    %thresh1 = graythresh(img1);      %get threshold for B/W image
    %thresh2 = graythresh(img2);
    %thresh3 = graythresh(img3);
    %img1=im2bw(img1,thresh1);
    %img2=im2bw(img2,thresh2);
    %img3=im2bw(img3,thresh3);

    img1_gt1=imread('9343 AM Edges.bmp');
    img2_gt1=imread('10905 JL Edges.bmp');
    img3_gt1=imread('43590 AM Edges.bmp');

    img1_gt=img1_gt1(256:768,320:960);
    img2_gt=img2_gt1(256:768,320:960);
    img3_gt=img3_gt1(256:768,320:960);

    psf=fspecial('gaussian',[5,5],1);
    img1=imfilter(img1,psf);
    img2=imfilter(img2,psf);
    img3=imfilter(img3,psf);
```



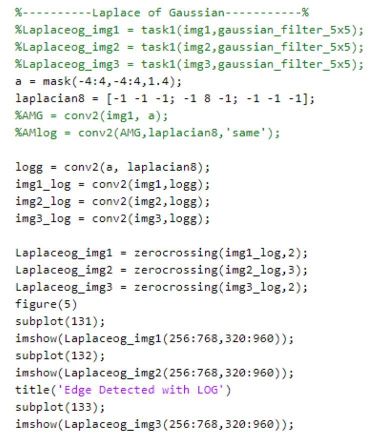
```
%--------sobel---------%
%img1_de_sobel=edge(img1,'sobel');
%img2_de_sobel=edge(img2,'sobel');
%img3_de_sobel=edge(img3,'sobel');
img1_sobelX1 = conv2(img1,sobelX);
img1_sobelY1 = conv2(img1,sobelY);

img2_sobelX1 = conv2(img2,sobelX);
img2_sobelY1 = conv2(img2,sobelY);

img3_sobelX1 = conv2(img3,sobelX);
img3_sobelY1 = conv2(img3,sobelY);

%absolute value
m_img1 = mag(img1_sobelX1,img1_sobelY1);
m_img2 = mag(img2_sobelX1,img2_sobelY1);
m_img3 = mag(img3_sobelX1,img3_sobelY1);

%threshold
img1_de_sobel = (m_img1 >50);
img2_de_sobel = (m_img2 >40);
img3_de_sobel = (m_img3 >30);

figure(1)
subplot(131);
imshow(img1_de_sobel(256:768,320:960));
subplot(132);
imshow(img2_de_sobel(256:768,320:960));
title('Edge Detected with Sobel')
subplot(133);
imshow(img3_de_sobel(256:768,320:960));
```



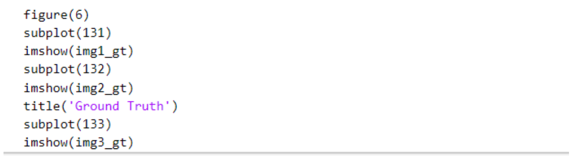
```
%--------Roberts------%
%roberts_img1= edge(img1,'roberts');
%roberts_img2 = edge(img2,'roberts');
%roberts_img3 = edge(img3,'roberts');
load filters
load roberts
img1_robertsX1 = conv2(img1,robertsA);
img1_robertsY1 = conv2(img1,robertsB);

img2_robertsX1 = conv2(img2,robertsA);
img2_robertsY1 = conv2(img2,robertsB);

img3_robertsX1 = conv2(img3,robertsA);
img3_robertsY1 = conv2(img3,robertsB);

%absolute value
m_img1_ro = mag(img1_robertsX1,img1_robertsY1);
m_img2_ro = mag(img2_robertsX1,img2_robertsY1);
m_img3_ro = mag(img3_robertsX1,img3_robertsY1);

roberts_img1 = m_img1_ro >10;
roberts_img2 = m_img2_ro >12;
roberts_img3 = m_img3_ro >4;
figure(2)
subplot(131);
imshow(roberts_img1(256:768,320:960));
subplot(132);
imshow(roberts_img2(256:768,320:960));
title('Edge Detected with Roberts')
subplot(133);
imshow(roberts_img3(256:768,320:960));
```



```
%----------Laplace----------%
%kernal_laplace=[1 1 1;1 -8 1;1 1 1];
%Laplace_img1=imfilter(img1,kernal_laplace,'conv','replicate');
%Laplace_img2=imfilter(img2,kernal_laplace,'conv','replicate');
%Laplace_img3=imfilter(img3,kernal_laplace,'conv','replicate');
laplacian4 = [0 -1 0; -1 4 -1; 0 -1 0];
img1_lap = conv2(img1,laplacian4);
img2_lap = conv2(img2,laplacian4);
img3_lap = conv2(img3,laplacian4);

Laplace_img1 = zerocrossing(img1_lap,6);
Laplace_img2 = zerocrossing(img2_lap,6);
Laplace_img3 = zerocrossing(img3_lap,6);

figure(4)
subplot(131);
imshow(Laplace_img1(256:768,320:960));
subplot(132);
imshow(Laplace_img2(256:768,320:960));
title('Edge Detected with Laplace')
subplot(133);
imshow(Laplace_img3(256:768,320:960));
```

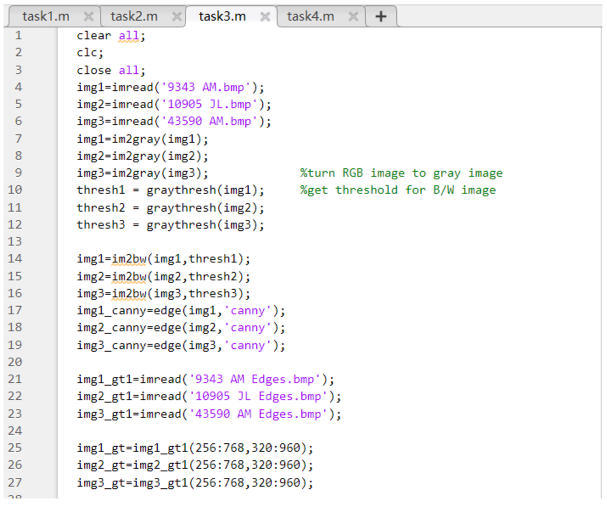


```
%---------Laplace of Gaussian----------%
%Laplaceog_img1 = task1(img1,gaussian_filter_5x5);
%Laplaceog_img2 = task1(img2,gaussian_filter_5x5);
%Laplaceog_img3 = task1(img3,gaussian_filter_5x5);
a = mask(-4:4,-4:4,1.4);
laplacian8 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
%AMG = conv2(img1, a);
%AMlog = conv2(AMG,laplacian8,'same');

logg = conv2(a, laplacian8);
img1_log = conv2(img1,logg);
img2_log = conv2(img2,logg);
img3_log = conv2(img3,logg);

Laplaceog_img1 = zerocrossing(img1_log,2);
Laplaceog_img2 = zerocrossing(img2_log,3);
Laplaceog_img3 = zerocrossing(img3_log,2);
figure(5)
subplot(131);
imshow(Laplaceog_img1(256:768,320:960));
subplot(132);
imshow(Laplaceog_img2(256:768,320:960));
title('Edge Detected with LOG')
subplot(133);
imshow(Laplaceog_img3(256:768,320:960));
```

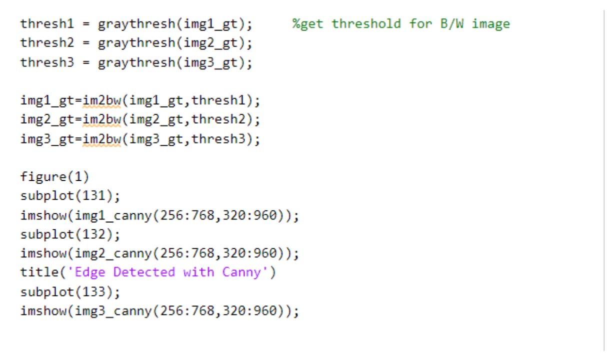


```
figure(6)
subplot(131)
imshow(img1_gt)
subplot(132)
imshow(img2_gt)
title('Ground Truth')
subplot(133)
imshow(img3_gt)
```

Task 3:

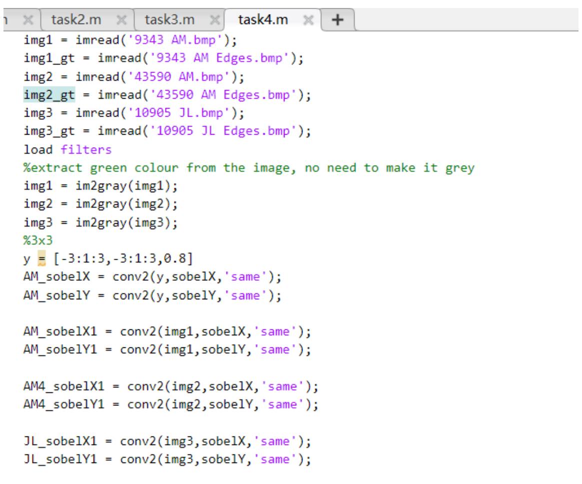


```
clear all;
clc;
close all;
img1=imread('9343 AM.bmp');
img2=imread('10905 JL.bmp');
img3=imread('43590 AM.bmp');
img1=im2gray(img1);
img2=im2gray(img2);
img3=im2gray(img3);              %turn RGB image to gray image
thresh1 = graythresh(img1);      %get threshold for B/W image
thresh2 = graythresh(img2);
thresh3 = graythresh(img3);

img1=im2bw(img1,thresh1);
img2=im2bw(img2,thresh2);
img3=im2bw(img3,thresh3);
img1_canny=edge(img1,'canny');
img2_canny=edge(img2,'canny');
img3_canny=edge(img3,'canny');

img1_gt1=imread('9343 AM Edges.bmp');
img2_gt1=imread('10905 JL Edges.bmp');
img3_gt1=imread('43590 AM Edges.bmp');

img1_gt=img1_gt1(256:768,320:960);
img2_gt=img2_gt1(256:768,320:960);
img3_gt=img3_gt1(256:768,320:960);
```

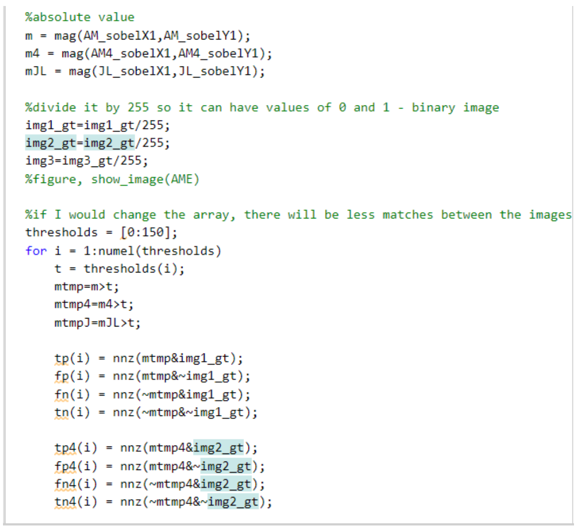


```
thresh1 = graythresh(img1_gt);     %get threshold for B/W image
thresh2 = graythresh(img2_gt);
thresh3 = graythresh(img3_gt);

img1_gt=im2bw(img1_gt,thresh1);
img2_gt=im2bw(img2_gt,thresh2);
img3_gt=im2bw(img3_gt,thresh3);

figure(1)
subplot(131);
imshow(img1_canny(256:768,320:960));
subplot(132);
imshow(img2_canny(256:768,320:960));
title('Edge Detected with Canny')
subplot(133);
imshow(img3_canny(256:768,320:960));
```

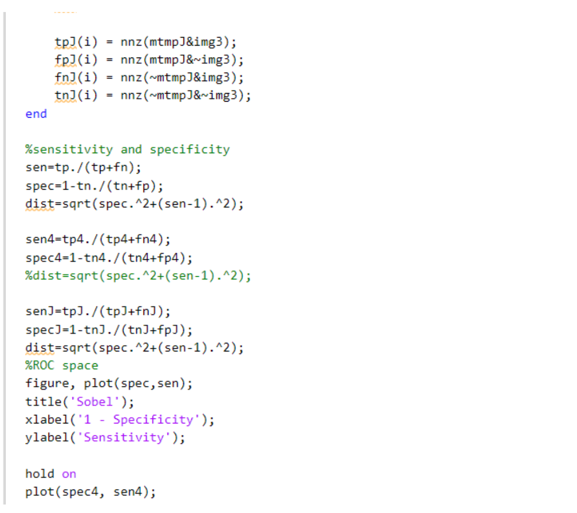
Task 4:



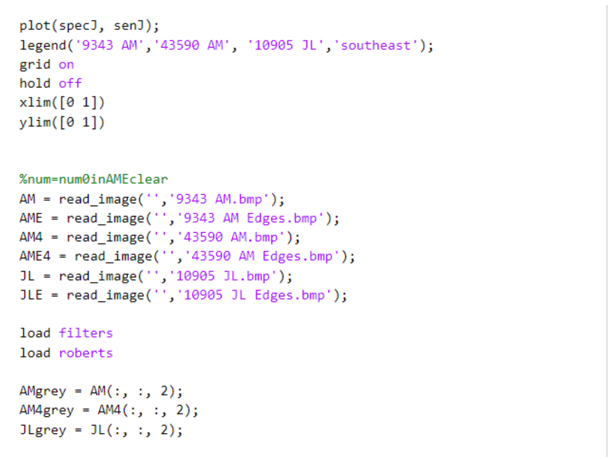
```
img1 = imread('9343 AM.bmp');
img1_gt = imread('9343 AM Edges.bmp');
img2 = imread('43590 AM.bmp');
img2_gt = imread('43590 AM Edges.bmp');
img3 = imread('10905 JL.bmp');
img3_gt = imread('10905 JL Edges.bmp');
load filters
%extract green colour from the image, no need to make it grey
img1 = im2gray(img1);
img2 = im2gray(img2);
img3 = im2gray(img3);
%3x3
y = [-3:1:3,-3:1:3,0.8]
AM_sobelX = conv2(y,sobelX,'same');
AM_sobelY = conv2(y,sobelY,'same');

AM_sobelX1 = conv2(img1,sobelX,'same');
AM_sobelY1 = conv2(img1,sobelY,'same');

AM4_sobelX1 = conv2(img2,sobelX,'same');
AM4_sobelY1 = conv2(img2,sobelY,'same');

JL_sobelX1 = conv2(img3,sobelX,'same');
JL_sobelY1 = conv2(img3,sobelY,'same');
```



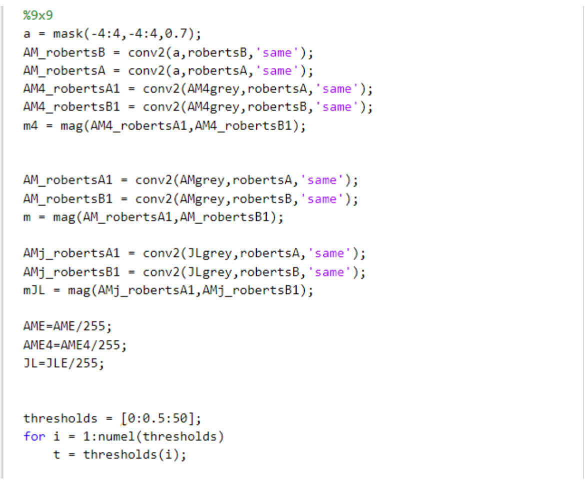
```
%absolute value
m = mag(AM_sobelX1,AM_sobelY1);
m4 = mag(AM4_sobelX1,AM4_sobelY1);
mJL = mag(JL_sobelX1,JL_sobelY1);

%divide it by 255 so it can have values of 0 and 1 - binary image
img1_gt=img1_gt/255;
img2_gt=img2_gt/255;
img3=img3_gt/255;
%figure, show_image(AME)

%if I would change the array, there will be less matches between the images
thresholds = [0:150];
for i = 1:numel(thresholds)
    t = thresholds(i);
    mtmp=m>t;
    mtmp4=m4>t;
    mtmpJ=mJL>t;

    tp(i) = nnz(mtmp&img1_gt);
    fp(i) = nnz(mtmp&~img1_gt);
    fn(i) = nnz(~mtmp&img1_gt);
    tn(i) = nnz(~mtmp&~img1_gt);

    tp4(i) = nnz(mtmp4&img2_gt);
    fp4(i) = nnz(mtmp4&~img2_gt);
    fn4(i) = nnz(~mtmp4&img2_gt);
    tn4(i) = nnz(~mtmp4&~img2_gt);
```



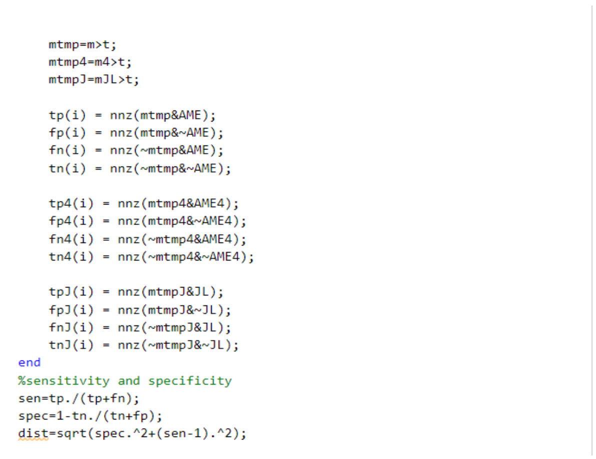
```
    tpJ(i) = nnz(mtmpJ&img3);
    fpJ(i) = nnz(mtmpJ&~img3);
    fnJ(i) = nnz(~mtmpJ&img3);
    tnJ(i) = nnz(~mtmpJ&~img3);
end

%sensitivity and specificity
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);

sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
%ROC space
figure, plot(spec,sen);
title('Sobel');
xlabel('1 - Specificity');
ylabel('Sensitivity');

hold on
plot(spec4, sen4);
```



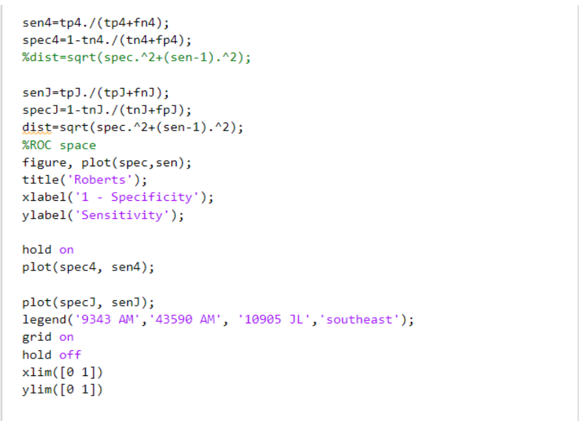
```
plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');
grid on
hold off
xlim([0 1])
ylim([0 1])


%num=num0inAMEclear
AM = read_image('','9343 AM.bmp');
AME = read_image('','9343 AM Edges.bmp');
AM4 = read_image('','43590 AM.bmp');
AME4 = read_image('','43590 AM Edges.bmp');
JL = read_image('','10905 JL.bmp');
JLE = read_image('','10905 JL Edges.bmp');

load filters
load roberts

AMgrey = AM(:, :, 2);
AM4grey = AM4(:, :, 2);
JLgrey = JL(:, :, 2);
```



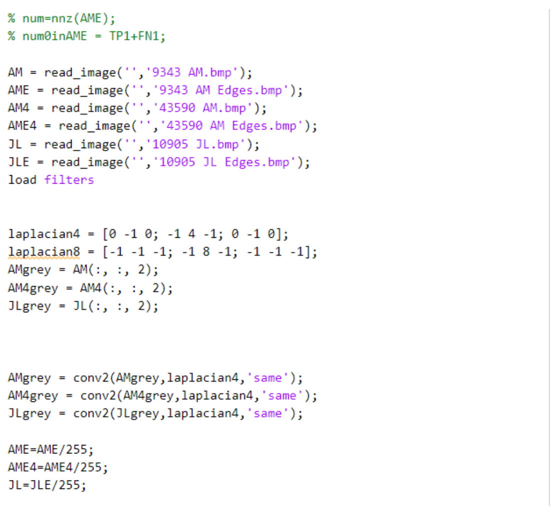
```
%9x9
a = mask(-4:4,-4:4,0.7);
AM_robertsB = conv2(a,robertsB,'same');
AM_robertsA = conv2(a,robertsA,'same');
AM4_robertsA1 = conv2(AM4grey,robertsA,'same');
AM4_robertsB1 = conv2(AM4grey,robertsB,'same');
m4 = mag(AM4_robertsA1,AM4_robertsB1);


AM_robertsA1 = conv2(AMgrey,robertsA,'same');
AM_robertsB1 = conv2(AMgrey,robertsB,'same');
m = mag(AM_robertsA1,AM_robertsB1);

AMj_robertsA1 = conv2(JLgrey,robertsA,'same');
AMj_robertsB1 = conv2(JLgrey,robertsB,'same');
mJL = mag(AMj_robertsA1,AMj_robertsB1);

AME=AME/255;
AME4=AME4/255;
JL=JLE/255;


thresholds = [0:0.5:50];
for i = 1:numel(thresholds)
    t = thresholds(i);
```



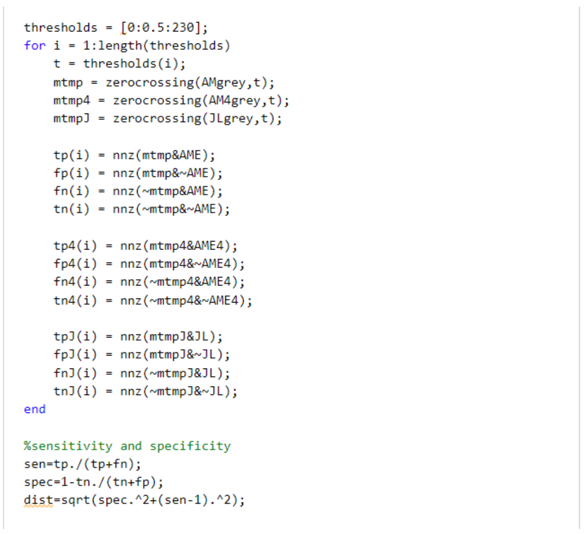
```
    mtmp=m>t;
    mtmp4=m4>t;
    mtmpJ=mJL>t;

    tp(i) = nnz(mtmp&AME);
    fp(i) = nnz(mtmp&~AME);
    fn(i) = nnz(~mtmp&AME);
    tn(i) = nnz(~mtmp&~AME);

    tp4(i) = nnz(mtmp4&AME4);
    fp4(i) = nnz(mtmp4&~AME4);
    fn4(i) = nnz(~mtmp4&AME4);
    tn4(i) = nnz(~mtmp4&~AME4);

    tpJ(i) = nnz(mtmpJ&JL);
    fpJ(i) = nnz(mtmpJ&~JL);
    fnJ(i) = nnz(~mtmpJ&JL);
    tnJ(i) = nnz(~mtmpJ&~JL);
end
%sensitivity and specificity
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);
```



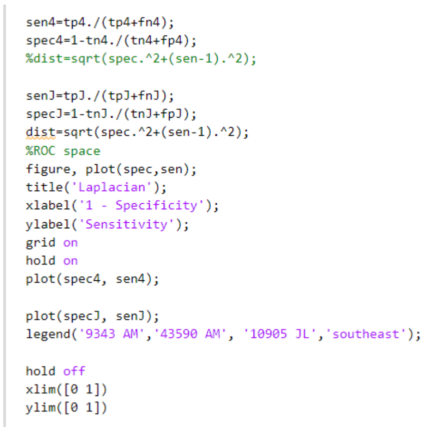
```
sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
%ROC space
figure, plot(spec,sen);
title('Roberts');
xlabel('1 - Specificity');
ylabel('Sensitivity');

hold on
plot(spec4, sen4);

plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');
grid on
hold off
xlim([0 1])
ylim([0 1])
```



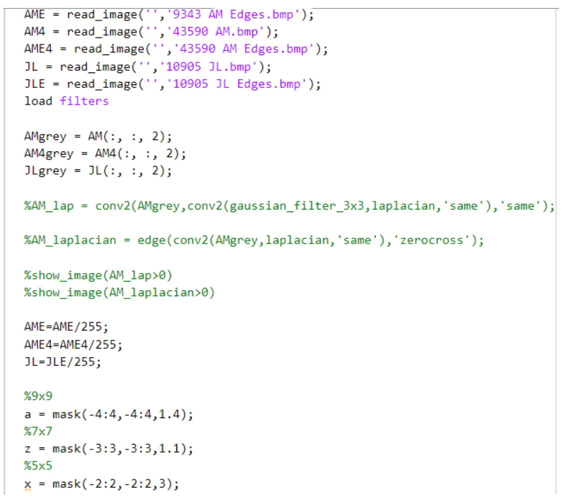
```
% num=nnz(AME);
% num0inAME = TP1+FN1;

AM = read_image('','9343 AM.bmp');
AME = read_image('','9343 AM Edges.bmp');
AM4 = read_image('','43590 AM.bmp');
AME4 = read_image('','43590 AM Edges.bmp');
JL = read_image('','10905 JL.bmp');
JLE = read_image('','10905 JL Edges.bmp');
load filters


laplacian4 = [0 -1 0; -1 4 -1; 0 -1 0];
laplacian8 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
AMgrey = AM(:, :, 2);
AM4grey = AM4(:, :, 2);
JLgrey = JL(:, :, 2);



AMgrey = conv2(AMgrey,laplacian4,'same');
AM4grey = conv2(AM4grey,laplacian4,'same');
JLgrey = conv2(JLgrey,laplacian4,'same');

AME=AME/255;
AME4=AME4/255;
JL=JLE/255;
```



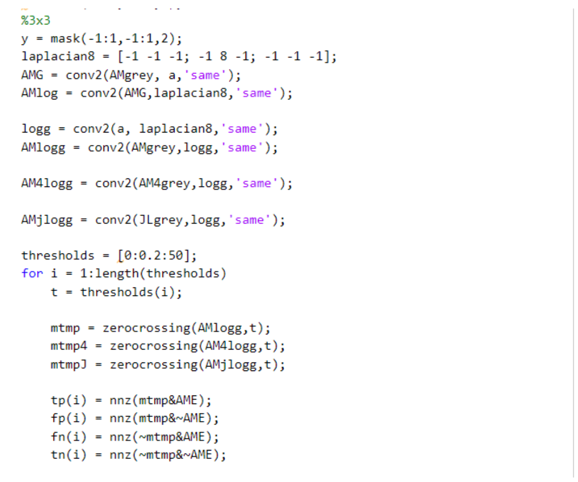
```
thresholds = [0:0.5:230];
for i = 1:length(thresholds)
    t = thresholds(i);
    mtmp = zerocrossing(AMgrey,t);
    mtmp4 = zerocrossing(AM4grey,t);
    mtmpJ = zerocrossing(JLgrey,t);

    tp(i) = nnz(mtmp&AME);
    fp(i) = nnz(mtmp&~AME);
    fn(i) = nnz(~mtmp&AME);
    tn(i) = nnz(~mtmp&~AME);

    tp4(i) = nnz(mtmp4&AME4);
    fp4(i) = nnz(mtmp4&~AME4);
    fn4(i) = nnz(~mtmp4&AME4);
    tn4(i) = nnz(~mtmp4&~AME4);

    tpJ(i) = nnz(mtmpJ&JL);
    fpJ(i) = nnz(mtmpJ&~JL);
    fnJ(i) = nnz(~mtmpJ&JL);
    tnJ(i) = nnz(~mtmpJ&~JL);
end

%sensitivity and specificity
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);
```



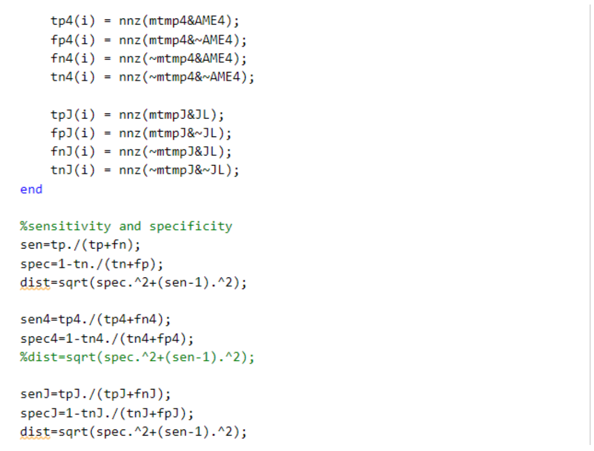
```
sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
%ROC space
figure, plot(spec,sen);
title('Laplacian');
xlabel('1 - Specificity');
ylabel('Sensitivity');
grid on
hold on
plot(spec4, sen4);

plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');

hold off
xlim([0 1])
ylim([0 1])
```



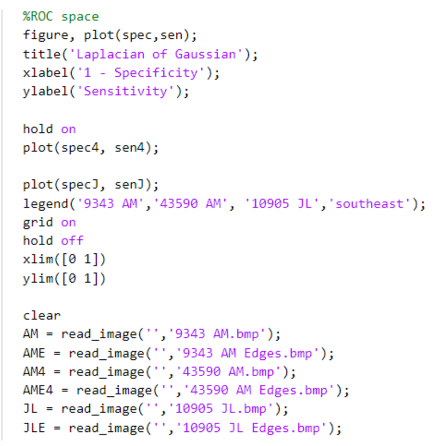
```
AME = read_image('','9343 AM Edges.bmp');
AM4 = read_image('','43590 AM.bmp');
AME4 = read_image('','43590 AM Edges.bmp');
JL = read_image('','10905 JL.bmp');
JLE = read_image('','10905 JL Edges.bmp');
load filters

AMgrey = AM(:, :, 2);
AM4grey = AM4(:, :, 2);
JLgrey = JL(:, :, 2);

%AM_lap = conv2(AMgrey,conv2(gaussian_filter_3x3,laplacian,'same'),'same');

%AM_laplacian = edge(conv2(AMgrey,laplacian,'same'),'zerocross');

%show_image(AM_lap>0)
%show_image(AM_laplacian>0)

AME=AME/255;
AME4=AME4/255;
JL=JLE/255;

%9x9
a = mask(-4:4,-4:4,1.4);
%7x7
z = mask(-3:3,-3:3,1.1);
%5x5
x = mask(-2:2,-2:2,3);
```



```
%3x3
y = mask(-1:1,-1:1,2);
laplacian8 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
AMG = conv2(AMgrey, a,'same');
AMlog = conv2(AMG,laplacian8,'same');

logg = conv2(a, laplacian8,'same');
AMlogg = conv2(AMgrey,logg,'same');

AM4logg = conv2(AM4grey,logg,'same');

AMjlogg = conv2(JLgrey,logg,'same');

thresholds = [0:0.2:50];
for i = 1:length(thresholds)
    t = thresholds(i);

    mtmp = zerocrossing(AMlogg,t);
    mtmp4 = zerocrossing(AM4logg,t);
    mtmpJ = zerocrossing(AMjlogg,t);

    tp(i) = nnz(mtmp&AME);
    fp(i) = nnz(mtmp&~AME);
    fn(i) = nnz(~mtmp&AME);
    tn(i) = nnz(~mtmp&~AME);
```



```
    tp4(i) = nnz(mtmp4&AME4);
    fp4(i) = nnz(mtmp4&~AME4);
    fn4(i) = nnz(~mtmp4&AME4);
    tn4(i) = nnz(~mtmp4&~AME4);

    tpJ(i) = nnz(mtmpJ&JL);
    fpJ(i) = nnz(mtmpJ&~JL);
    fnJ(i) = nnz(~mtmpJ&JL);
    tnJ(i) = nnz(~mtmpJ&~JL);
end

%sensitivity and specificity
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);

sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
```



```
%ROC space
figure, plot(spec,sen);
title('Laplacian of Gaussian');
xlabel('1 - Specificity');
ylabel('Sensitivity');

hold on
plot(spec4, sen4);

plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');
grid on
hold off
xlim([0 1])
ylim([0 1])

clear
AM = read_image('','9343 AM.bmp');
AME = read_image('','9343 AM Edges.bmp');
AM4 = read_image('','43590 AM.bmp');
AME4 = read_image('','43590 AM Edges.bmp');
JL = read_image('','10905 JL.bmp');
JLE = read_image('','10905 JL Edges.bmp');
```

```
load filters

AMgrey = AM(:, :, 2);
AM4grey = AM4(:, :, 2);
JLgrey = JL(:, :, 2);

%5x5
x = mask(-2:2,-2:2,1.4);

AM_Gauss1 = conv2(x,first_order_gaussian_filter_1d_length5,'same');
AM_Gauss2 = conv2(x,first_order_gaussian_filter_1d_length5','same');

AM_Gauss11 = conv2(AMgrey,AM_Gauss1,'same');
AM_Gauss22 = conv2(AMgrey,AM_Gauss2,'same');

AM_Gauss4 = conv2(AM4grey,AM_Gauss1,'same');
AM_Gauss44 = conv2(AM4grey,AM_Gauss2,'same');

AM_GaussJ1 = conv2(JLgrey,AM_Gauss1,'same');
AM_GaussJ2 = conv2(JLgrey,AM_Gauss2,'same');
m=mag(AM_Gauss11,AM_Gauss22);
m4=mag(AM_Gauss4,AM_Gauss44);
mJL=mag(AM_GaussJ1,AM_GaussJ2);
```

```
AME=AME/255;
AME4=AME4/255;
JL=JLE/255;

%show_image(n>1)



thresholds = [0:0.2:50];
for i = 1:numel(thresholds)
    t = thresholds(i);

    mtmp=m>t;
    mtmp4=m4>t;
    mtmpJ=mJL>t;

    tp(i) = nnz(mtmp&AME);
    fp(i) = nnz(mtmp&~AME);
    fn(i) = nnz(~mtmp&AME);
    tn(i) = nnz(~mtmp&~AME);

    tp4(i) = nnz(mtmp4&AME4);
    fp4(i) = nnz(mtmp4&~AME4);
    fn4(i) = nnz(~mtmp4&AME4);
    tn4(i) = nnz(~mtmp4&~AME4);
```

```
    tpJ(i) = nnz(mtmpJ&JL);
    fpJ(i) = nnz(mtmpJ&~JL);
    fnJ(i) = nnz(~mtmpJ&JL);
    tnJ(i) = nnz(~mtmpJ&~JL);
end

%sensitivity and specificity
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);

sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
%ROC space
figure, plot(spec,sen);
title('Gaussian');
xlabel('1 - Specificity');
ylabel('Sensitivity');
```

```
hold on
plot(spec4, sen4);

plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');
grid on
hold off
xlim([0 1])
ylim([0 1])


%hold on
%plot(spec(b),sen(b),'*r')
%[a,b]=min(dist)

img1 = imread('9343 AM.bmp');
img1_gt = imread('9343 AM Edges.bmp');
img2 = imread('43590 AM.bmp');
img2_gt = imread('43590 AM Edges.bmp');
img3 = imread('10905 JL.bmp');
img3_gt = imread('10905 JL Edges.bmp');
```

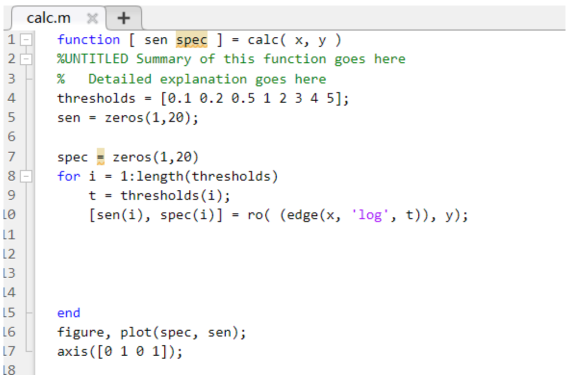
```
img1=im2gray(img1);
img2=im2gray(img2);
img3=im2gray(img3);

img1_canny=edge(img1,'canny');
img2_canny=edge(img2,'canny');
img3_canny=edge(img2,'canny');

thresholds = [0:0.2:50];
for i = 1:numel(thresholds)
    t = thresholds(i);

    mtmp=m>t;
    mtmp4=m4>t;
    mtmpJ=mJL>t;

    tp(i) = nnz(mtmp&img1_gt);
    fp(i) = nnz(mtmp&~img1_gt);
    fn(i) = nnz(~mtmp&img1_gt);
    tn(i) = nnz(~mtmp&~img1_gt);

    tp4(i) = nnz(mtmp4&img2_gt);
    fp4(i) = nnz(mtmp4&~img2_gt);
    fn4(i) = nnz(~mtmp4&img2_gt);
    tn4(i) = nnz(~mtmp4&~img2_gt);
```

```
    tpJ(i) = nnz(mtmpJ&img3_gt);
    fpJ(i) = nnz(mtmpJ&~img3_gt);
    fnJ(i) = nnz(~mtmpJ&img3_gt);
    tnJ(i) = nnz(~mtmpJ&~img3_gt);
end
sen=tp./(tp+fn);
spec=1-tn./(tn+fp);
dist=sqrt(spec.^2+(sen-1).^2);

sen4=tp4./(tp4+fn4);
spec4=1-tn4./(tn4+fp4);
%dist=sqrt(spec.^2+(sen-1).^2);

senJ=tpJ./(tpJ+fnJ);
specJ=1-tnJ./(tnJ+fpJ);
dist=sqrt(spec.^2+(sen-1).^2);
%ROC space
figure, plot(spec,sen);
title('Canny');
xlabel('1 - Specificity');
ylabel('Sensitivity');

hold on
plot(spec4, sen4);
```

```
plot(specJ, senJ);
legend('9343 AM','43590 AM', '10905 JL','southeast');
grid on
hold off
xlim([0 1])
ylim([0 1])
```

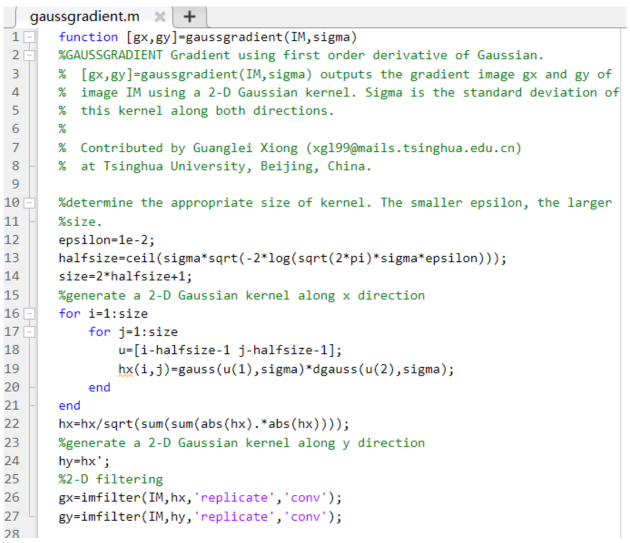
Functions(utils)

Cal.m

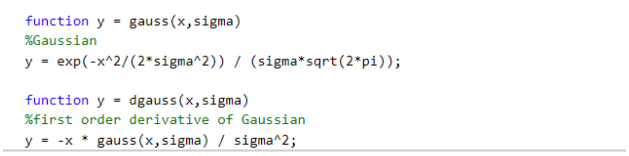


```
function [ sen spec ] = calc( x, y )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
thresholds = [0.1 0.2 0.5 1 2 3 4 5];
sen = zeros(1,20);

spec = zeros(1,20)
for i = 1:length(thresholds)
    t = thresholds(i);
    [sen(i), spec(i)] = ro( (edge(x, 'log', t)), y);





end
figure, plot(spec, sen);
axis([0 1 0 1]);
```

Gaussgradient.m

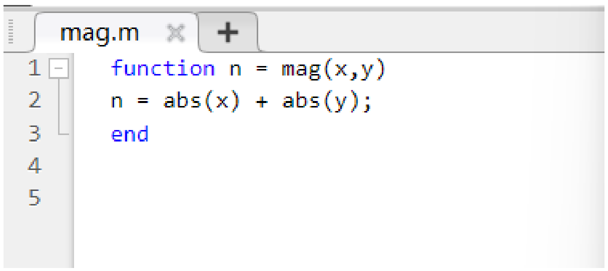


```matlab
function [gx,gy]=gaussgradient(IM,sigma)
%GAUSSGRADIENT Gradient using first order derivative of Gaussian.
%  [gx,gy]=gaussgradient(IM,sigma) outputs the gradient image gx and gy of
%  image IM using a 2-D Gaussian kernel. Sigma is the standard deviation of
%  this kernel along both directions.
%
%  Contributed by Guanglei Xiong (xgl99@mails.tsinghua.edu.cn)
%  at Tsinghua University, Beijing, China.

%determine the appropriate size of kernel. The smaller epsilon, the larger
%size.
epsilon=1e-2;
halfsize=ceil(sigma*sqrt(-2*log(sqrt(2*pi)*sigma*epsilon)));
size=2*halfsize+1;
%generate a 2-D Gaussian kernel along x direction
for i=1:size
    for j=1:size
        u=[i-halfsize-1 j-halfsize-1];
        hx(i,j)=gauss(u(1),sigma)*dgauss(u(2),sigma);
    end
end
hx=hx/sqrt(sum(sum(abs(hx).*abs(hx))));
%generate a 2-D Gaussian kernel along y direction
hy=hx';
%2-D filtering
gx=imfilter(IM,hx,'replicate','conv');
gy=imfilter(IM,hy,'replicate','conv');
```



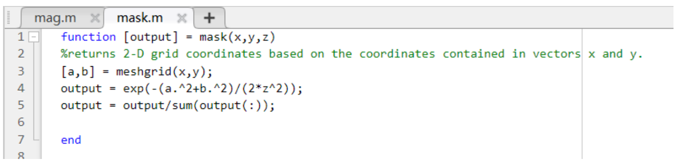
```matlab
function y = gauss(x,sigma)
%Gaussian
y = exp(-x^2/(2*sigma^2)) / (sigma*sqrt(2*pi));

function y = dgauss(x,sigma)
%first order derivative of Gaussian
y = -x * gauss(x,sigma) / sigma^2;
```

Mag.m



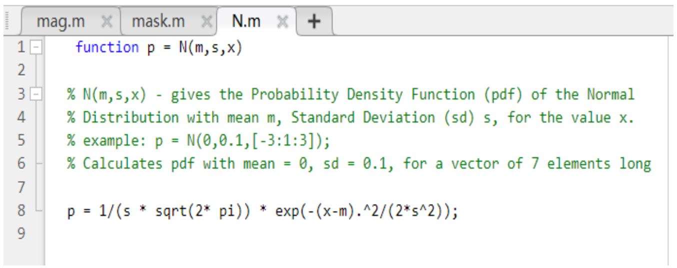
```matlab
function n = mag(x,y)
n = abs(x) + abs(y);
end
```

Mask.m



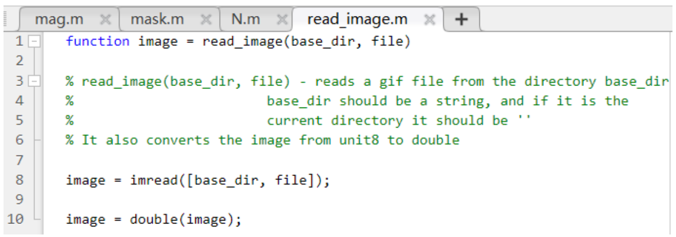
```matlab
function [output] = mask(x,y,z)
%returns 2-D grid coordinates based on the coordinates contained in vectors x and y.
[a,b] = meshgrid(x,y);
output = exp(-(a.^2+b.^2)/(2*z^2));
output = output/sum(output(:));

end
```

N.m



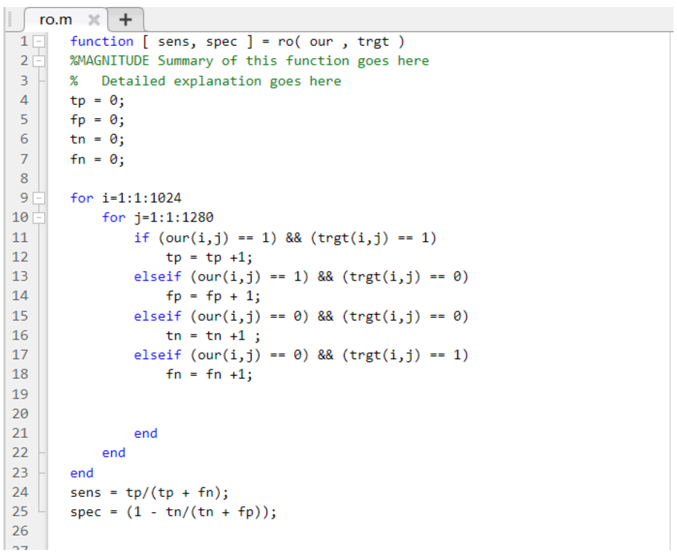
```matlab
function p = N(m,s,x)

% N(m,s,x) - gives the Probability Density Function (pdf) of the Normal
% Distribution with mean m, Standard Deviation (sd) s, for the value x.
% example: p = N(0,0.1,[-3:1:3]);
% Calculates pdf with mean = 0, sd = 0.1, for a vector of 7 elements long

p = 1/(s * sqrt(2* pi)) * exp(-(x-m).^2/(2*s^2));
```

Read_image.m



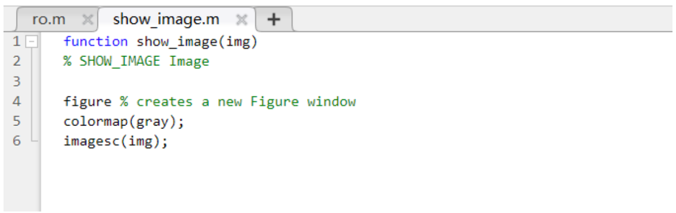
```matlab
function image = read_image(base_dir, file)

% read_image(base_dir, file) - reads a gif file from the directory base_dir
%                          base_dir should be a string, and if it is the
%                          current directory it should be ''
% It also converts the image from unit8 to double

image = imread([base_dir, file]);

image = double(image);
```

Ro.m



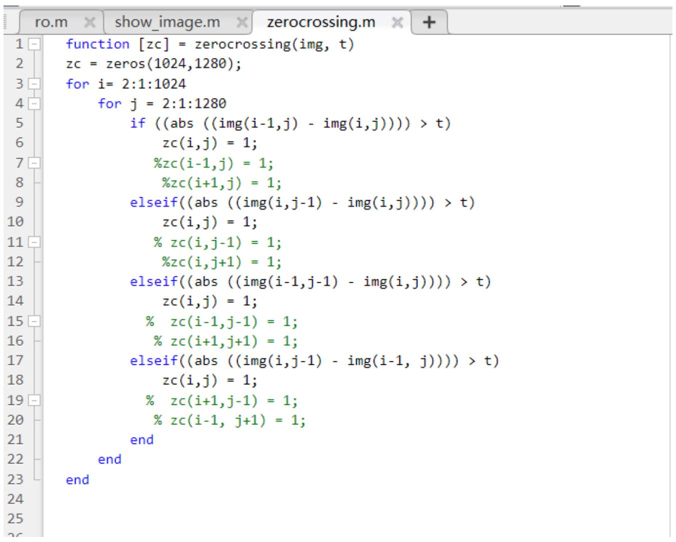
```matlab
function [ sens, spec ] = ro( our , trgt )
%MAGNITUDE Summary of this function goes here
%   Detailed explanation goes here
tp = 0;
fp = 0;
tn = 0;
fn = 0;

for i=1:1:1024
    for j=1:1:1280
        if (our(i,j) == 1) && (trgt(i,j) == 1)
            tp = tp +1;
        elseif (our(i,j) == 1) && (trgt(i,j) == 0)
            fp = fp + 1;
        elseif (our(i,j) == 0) && (trgt(i,j) == 0)
            tn = tn +1 ;
        elseif (our(i,j) == 0) && (trgt(i,j) == 1)
            fn = fn +1;


        end
    end
end
sens = tp/(tp + fn);
spec = (1 - tn/(tn + fp));
```

Show_image.m



```matlab
function show_image(img)
% SHOW_IMAGE Image

figure % creates a new Figure window
colormap(gray);
imagesc(img);
```

Zerocrossing.m



```matlab
function [zc] = zerocrossing(img, t)
zc = zeros(1024,1280);
for i= 2:1:1024
    for j = 2:1:1280
        if ((abs ((img(i-1,j) - img(i,j)))) > t)
            zc(i,j) = 1;
           %zc(i-1,j) = 1;
            %zc(i+1,j) = 1;
        elseif((abs ((img(i,j-1) - img(i,j)))) > t)
            zc(i,j) = 1;
           % zc(i,j-1) = 1;
            %zc(i,j+1) = 1;
        elseif((abs ((img(i-1,j-1) - img(i,j)))) > t)
            zc(i,j) = 1;
          %  zc(i-1,j-1) = 1;
           % zc(i+1,j+1) = 1;
        elseif((abs ((img(i,j-1) - img(i-1, j)))) > t)
            zc(i,j) = 1;
          %  zc(i+1,j-1) = 1;
           % zc(i-1, j+1) = 1;
        end
    end
end
```