

Introduction to Intelligent Systems - Lab 5

Diego Velasco Volkmann Eduardo Faccin Vernier
S2851059 S3012875

October 31, 2015

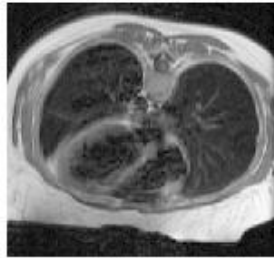
Exercise 1:

To perform this assignment, a convolution function was implemented. It takes an image and a 3x3 kernel as arguments and perform convolution on a padded image with repeated boundary pixel values.

```
1 function resultImage = convolution( originalImage, kernel )
2 %CONVOLUTION takes an image and a 3x3 kernel and applies convolution
3
4     resultImage = zeros(length(originalImage(:,1)),length(originalImage(1,:)));
5     % padding image with boundary pixels
6     paddedImage = padarray (originalImage, [1 1], 'replicate', 'both');
7
8     % 180 degree rotation of kernel in order to perform convolution, not
9     % correlation
10    % K      [0]      [1]      [2]      [3]      [4]      [5]      [6]      [7]      [8]
11    % (x+1,y+1) (x,y+1) (x-1,y+1) (x+1,y) (x,y) (x-1,y) (x+1,y-1) (x,y-1) (x-1,
12    % y-1)
13
14    for x = 2:(length(paddedImage(:,1))-1)
15        for y = 2:(length(paddedImage(1,:))-1)
16            newValue = 0;
17            newValue = newValue + (paddedImage(x+1, y+1)) * kernel (1);
18            newValue = newValue + (paddedImage(x, y+1)) * kernel (2);
19            newValue = newValue + (paddedImage(x-1, y+1)) * kernel (3);
20            newValue = newValue + (paddedImage(x+1, y)) * kernel (4);
21            newValue = newValue + (paddedImage(x, y)) * kernel (5);
22            newValue = newValue + (paddedImage(x-1, y)) * kernel (6);
23            newValue = newValue + (paddedImage(x+1, y-1)) * kernel (7);
24            newValue = newValue + (paddedImage(x, y-1)) * kernel (8);
25            newValue = newValue + (paddedImage(x-1, y-1)) * kernel (9);
26            if (newValue < 0)
27                newValue = 0;
28            end
29            if(newValue>1)
30                newValue = 1;
31            end
32            resultImage(x-1,y-1) = newValue;
33        end
34    end
35 end
```

Each filter was applied in four images: the original chest image, and the chest image with 0.001, 0.01 and 0.05 Gaussian noise added.

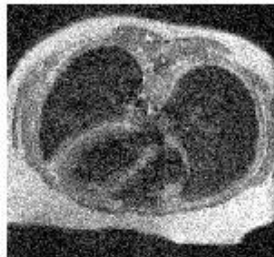
Original



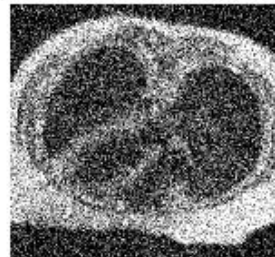
Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance

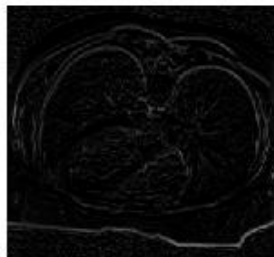


Gaussian Noise with 0.05 variance

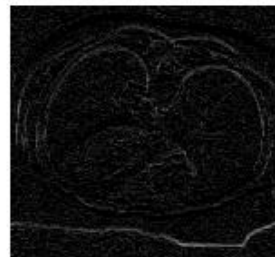


The results of the filtering are displayed below for each filter kernel.

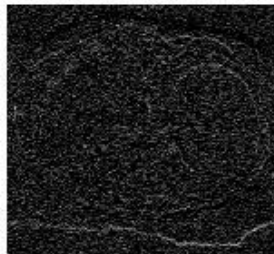
Roberts Filter



Gaussian Noise with 0.001 variance



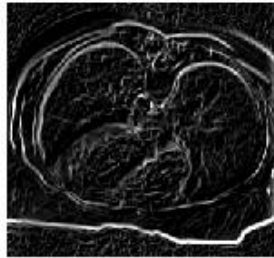
Gaussian Noise with 0.01 variance



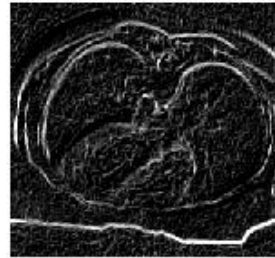
Gaussian Noise with 0.05 variance



Prewitt Filter



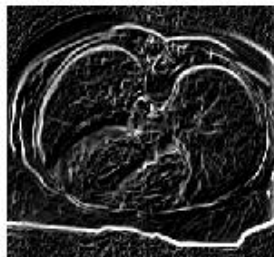
Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance



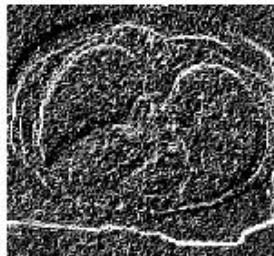
Sobel Filter



Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance



0.05 Gaussian Noise Added

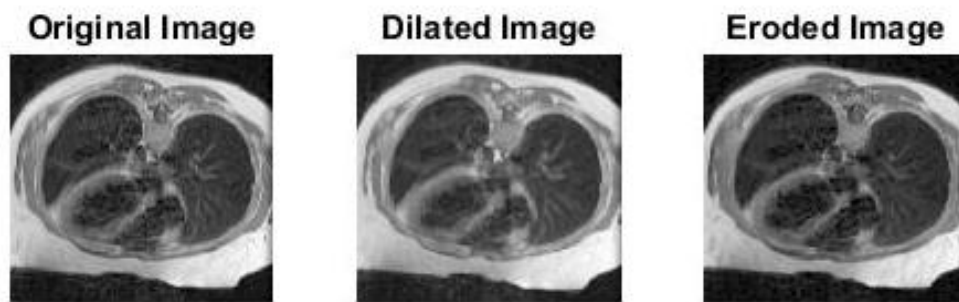


Noise has a big influence in all of these filters considering they are all quite small (3x3 or 2x2). The smaller the filter, the more noise sensitive it is, therefore, as it is visible in the images above, Roberts Cross (2x2) convolutions get very degraded as the noise is increased compared to the other two convolution

kernels. Increasing the size of the filters would make the filtering less noisy, but, as a trade off, would result in a poorer localization of features.

Exercise 2:

First in the assignment, we implement the most basic morphological functions. In a binary image, as seen in class, when applying Erosion, if any pixel inside the structuring elements is has a value of 0 in the neighborhood, the output pixel is also set to 0. But dealing with grey scale images, the value of the output pixel is the minimum value of all the pixels in the input pixel's neighborhood. The same applies to Dilation, but instead of the minimum value, the output pixel is set as the maximum.

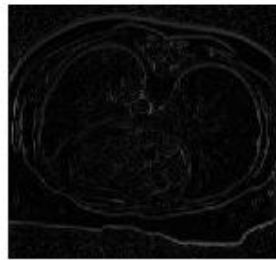


The morphological edge detector used in this assignment is Boundary Extraction. It consists on subtracting from the original A image an eroded version of it with a B Structuring Element.

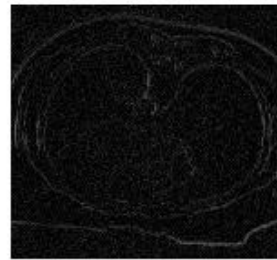
$$\beta(A) = A - (A \ominus B)$$

The results of this technique with different Structuring Elements are displayed below.

Boundary Extraction



Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance**Gaussian Noise with 0.05 variance**

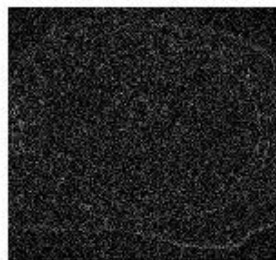
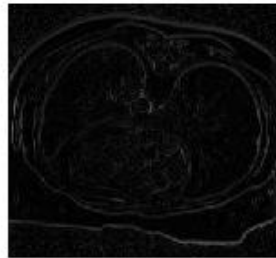
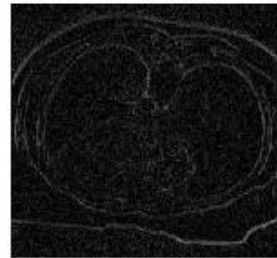


Figure 1: Structuring element = $[0 \ 1 \ 0; 1 \ 1 \ 1; 0 \ 1 \ 0]$

Boundary Extraction



Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance**Gaussian Noise with 0.05 variance**

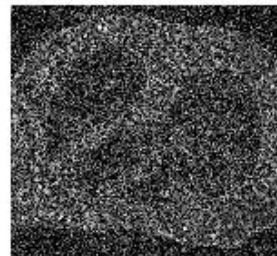
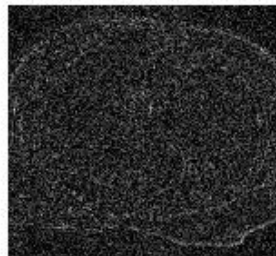


Figure 2: Structuring element = $[1 \ 1 \ 1; 1 \ 1 \ 1; 1 \ 1 \ 1]$

As all previous techniques, this Boundary Extraction is also very noise sensitive. But comparing the results obtained from each structuring element, it is safe to say that the $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ structuring element is slightly more robust to noise addition.

Exercise 3:

Modulating the parameters on the Canny Edge algorithm it is possible to generate a very noise resistant filtering. What our group observed is that increasing the value of the minimum threshold and sigma, most of the noise is discarded and only strong edges are highlighted. That is very clear comparing Figure 3, which has low threshold and sigma, to Figure 6, for example. But this increase in the parameters comes at a cost - edges that are not very strong, but might be important for a diagnoses, for instance, will be disregarded. Increasing the Threshold to very high numbers, as in Figure 7 will also result in edges loss. For noisy images, we've come to the conclusion that high sigma values (as $\sqrt{4}$ and $\sqrt{8}$) with threshold values of $[0.1 \ 0.25]$ are a good trade off between noise resistance and edge strength. Compared to the previous techniques, in this application, Canny seems to be the best by a large margin.

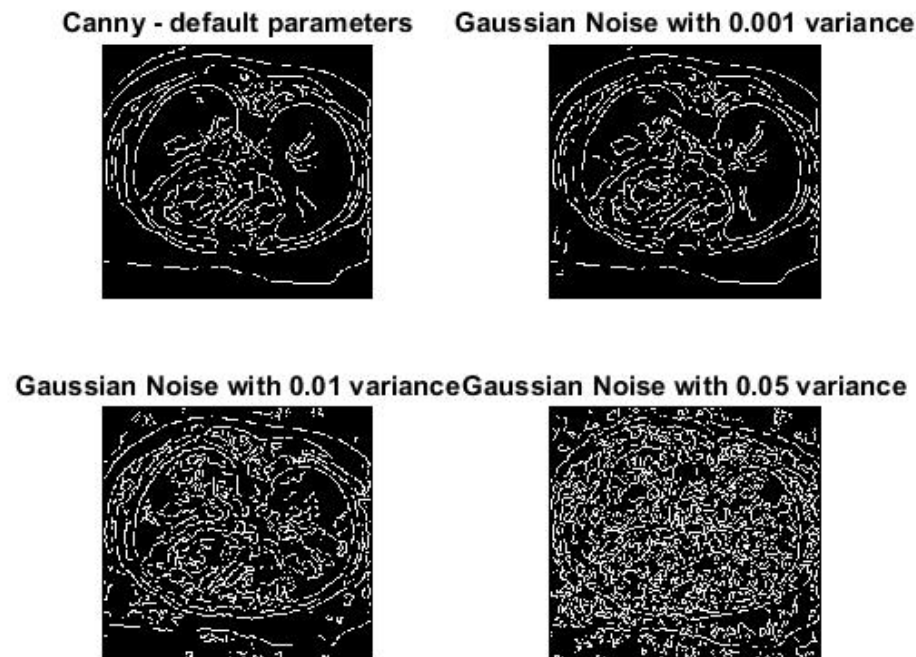
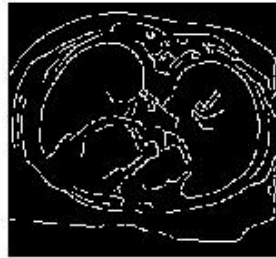


Figure 3: Default values (Threshold chosen automatically by MATLAB and $\sigma = \sqrt{2}$)

Canny - Threshold 0.1 / 0.25 Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance Gaussian Noise with 0.05 variance

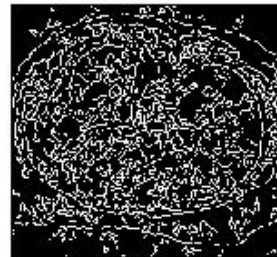


Figure 4: Threshold is set as [0.1 0.25] and sigma is default = $\sqrt{2}$

Threshold 0.1/0.25 - Sigma = $\sqrt{4}$

Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance

Gaussian Noise with 0.05 variance



Figure 5: Threshold is set as [0.1 0.25] and sigma = $\sqrt{4}$

Threshold 0.1/0.25 - Sigma = $\sqrt{8}$ Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance Gaussian Noise with 0.05 variance



Figure 6: Threshold is set as [0.1 0.25] and sigma = $\sqrt{8}$

Canny - Threshold 0.3 / 0.5 Gaussian Noise with 0.001 variance



Gaussian Noise with 0.01 variance Gaussian Noise with 0.05 variance



Figure 7: Threshold is set very high [0.3 0.5] and sigma is default = $\sqrt{2}$

Exercise 4:

Study the paper Contour and boundary detection improved by surround suppression of texture edges by Grigorescu, Petkov, and Westenberg provided on Nestor. This is the predecessor of the methods described in the lecture. The method can be accessed through a web implementation found at:

<http://www.cs.rug.nl/imaging>

Click on Canny edge detector (Canny filter) for image processing and computer vision:, which will lead you to a web application which can perform both the Canny detector and two surround inhibition methods. Select the kanizsa.png image as the target and update the view. Try various settings of surround inhibition, and explain the differences in output. Do the same for popout.png. What is the difference with kanizsa.png? Hint: Set the sigma (around 4 seems to work) and K2 (around 8) parameters high enough to get the expected inhibition effects.

Answer: According to Figure 10, when there's too much surrounding inhibition, popout.png becomes "invisible", unlike kanizsa.png, shown on Figure 11, which survives with a few elements, since popout is composed of several short lines, but kanizsa has a full square outline. The surround inhibition is meant to suppress texture edges while leaving relatively unaffected the contours of objects and region boundaries, therefore removing short and non-continuous edges. The value of sigma changes the edge outline to a more rounded shape, depending on how high sigma is.

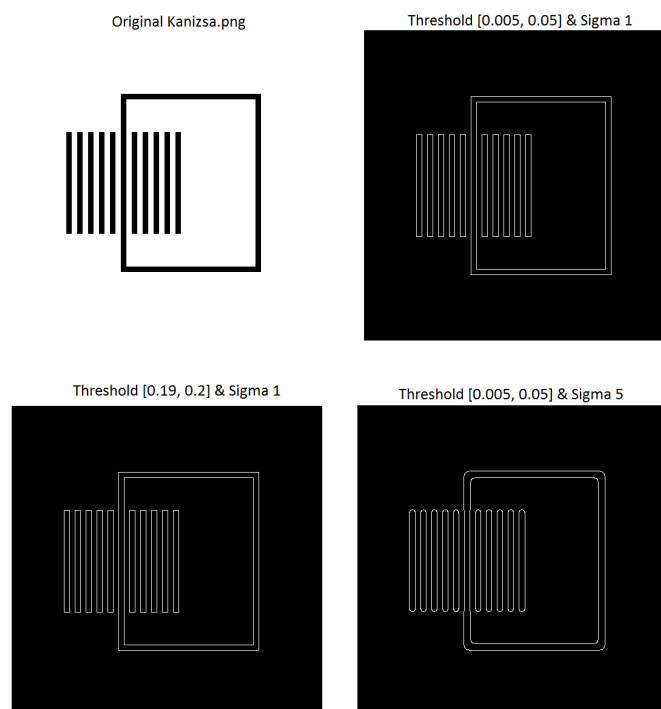


Figure 8: Kanizsa.png tests

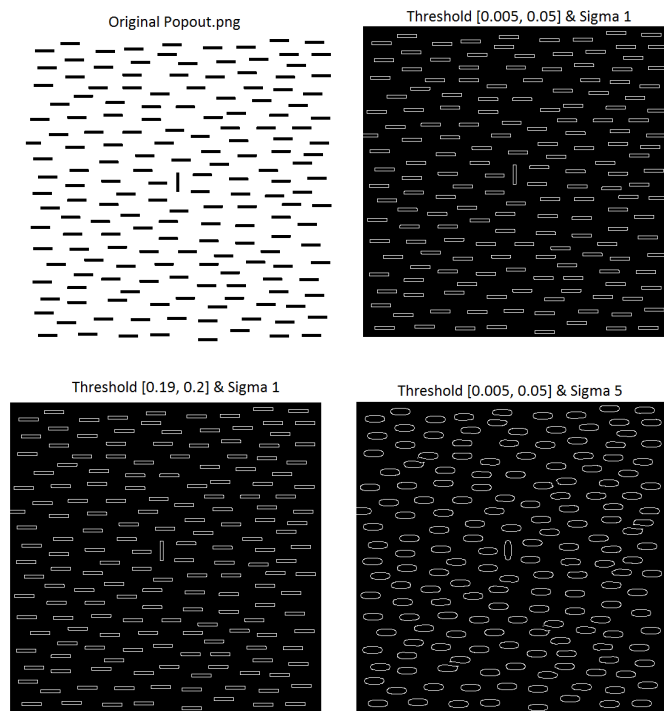


Figure 9: Popout.png tests

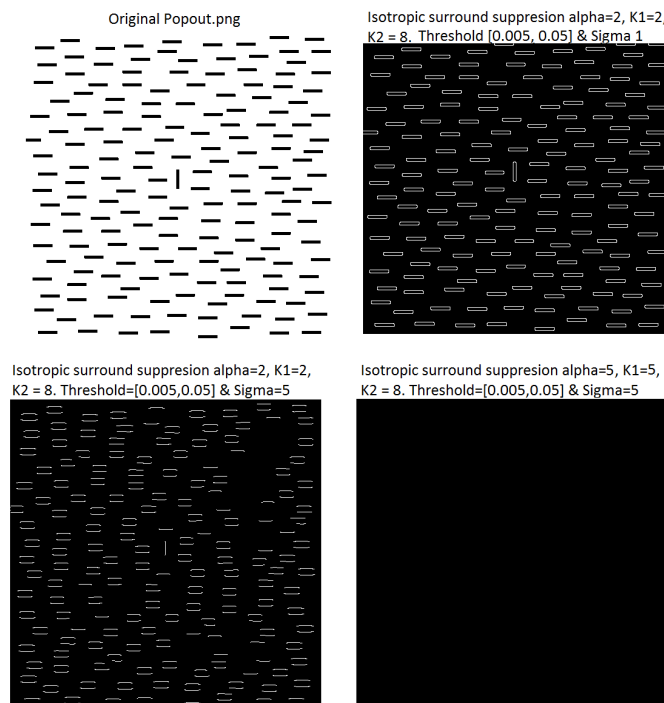


Figure 10: Kanizsa.png tests with surround inhibition

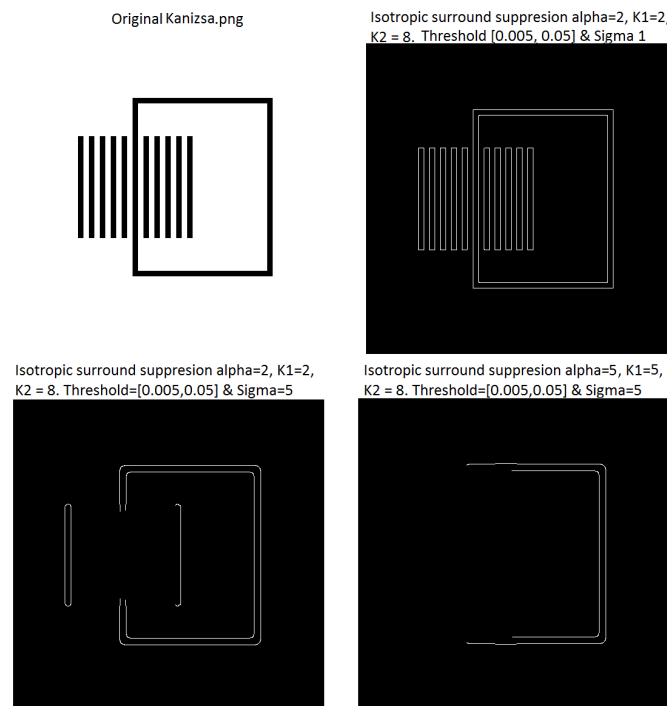


Figure 11: Popout.png tests with surround inhibition

Division of work

Eduardo was responsible for exercise 1 and 2, while Diego was responsible for exercise 3 and 4. Later on, Eduardo deleted Diego's answer on exercise 3, writing a new answer.

Appendix:

Listing 1: lab5q1.m file - function calls and plots using the convolution function shown in the first exercise

```
1 close all;
2
3 I = im2double(imread('chest.pgm'));
4
5 % Convolution Kernels
6 hRbt1 = [0 0 0; 0 -1 0; 0 0 1];
7 hRbt2 = [0 0 0; 0 0 -1; 0 1 0];
8 hPwt1 = [-1 -1 -1; 0 0 0; 1 1 1];
9 hPwt2 = [-1 0 1; -1 0 1; -1 0 1];
10 hSbl1 = [-1 -2 -1; 0 0 0; 1 2 1];
11 hSbl2 = [-1 0 1; -2 0 2; -1 0 1];
12
13 % Images with added noise
14 I0001 = imnoise(I, 'gaussian', 0, 0.001);
15 I001 = imnoise(I, 'gaussian', 0, 0.01);
16 I005 = imnoise(I, 'gaussian', 0, 0.05);
17
18 % Images with different amounts of noise
19 figure;
20 subplot(2, 2, 1);
21 imshow(I);
22 title('Original');
23 subplot(2, 2, 2);
24 imshow(I0001);
25 title('Gaussian Noise with 0.001 variance');
26 subplot(2, 2, 3);
27 imshow(I001);
28 title('Gaussian Noise with 0.01 variance');
29 subplot(2, 2, 4);
30 imshow(I005);
31 title('Gaussian Noise with 0.05 variance');
32
33 % Roberts kernel
34 figure;
35 Rbt1 = convolution(I, hRbt1);
36 Rbt2 = convolution(I, hRbt2);
37 Rbt = sqrt(Rbt1.^2+Rbt2.^2);
38 subplot(2, 2, 1);
39 imshow(Rbt);
40 title('Roberts Filter');
41
42 Rbt1 = convolution(I0001, hRbt1);
43 Rbt2 = convolution(I0001, hRbt2);
44 Rbt = sqrt(Rbt1.^2+Rbt2.^2);
45 subplot(2, 2, 2);
46 imshow(Rbt);
47 title('Gaussian Noise with 0.001 variance');
48
49 Rbt1 = convolution(I001, hRbt1);
50 Rbt2 = convolution(I001, hRbt2);
51 Rbt = sqrt(Rbt1.^2+Rbt2.^2);
52 subplot(2, 2, 3);
53 imshow(Rbt);
54 title('Gaussian Noise with 0.01 variance');
55
56 Rbt1 = convolution(I005, hRbt1);
57 Rbt2 = convolution(I, hRbt2);
58 Rbt = sqrt(Rbt1.^2+Rbt2.^2);
59 subplot(2, 2, 4);
60 imshow(Rbt);
61 title('Gaussian Noise with 0.05 variance');
62
63
64 % Prewitt kernel
65 figure;
66 Pwt1 = convolution(I, hPwt1);
67 Pwt2 = convolution(I, hPwt2);
68 Pwt = sqrt(Pwt1.^2+Pwt2.^2);
69 subplot(2, 2, 1);
70 imshow(Pwt);
71 title('Prewitt Filter');
72
73 Pwt1 = convolution(I0001, hPwt1);
74 Pwt2 = convolution(I0001, hPwt2);
75 Pwt = sqrt(Pwt1.^2+Pwt2.^2);
```

```

76 subplot(2, 2, 2);
77 imshow(Pwt);
78 title('Gaussian Noise with 0.001 variance');
79
80 Pwt1 = convolution(I001, hPwt1);
81 Pwt2 = convolution(I001, hPwt2);
82 Pwt = sqrt(Pwt1.^2+Pwt2.^2);
83 subplot(2, 2, 3);
84 imshow(Pwt);
85 title('Gaussian Noise with 0.01 variance');
86
87 Pwt1 = convolution(I005, hPwt1);
88 Pwt2 = convolution(I001, hPwt2);
89 Pwt = sqrt(Pwt1.^2+Pwt2.^2);
90 subplot(2, 2, 4);
91 imshow(Pwt);
92 title('Gaussian Noise with 0.05 variance');
93
94
95 % Sobel kernel vertical
96 figure;
97 Sb11 = convolution(I, hSb11);
98 Sb12 = convolution(I, hSb12);
99 Sb1 = sqrt(Sb11.^2+Sb12.^2);
100 subplot(2, 2, 1);
101 imshow(Sb1);
102 title('Sobel Filter');
103
104 Sb11 = convolution(I0001, hSb11);
105 Sb12 = convolution(I0001, hSb12);
106 Sb1 = sqrt(Sb11.^2+Sb12.^2);
107 subplot(2, 2, 2);
108 imshow(Sb1);
109 title('Gaussian Noise with 0.001 variance');
110
111 Sb11 = convolution(I001, hSb11);
112 Sb12 = convolution(I001, hSb12);
113 Sb1 = sqrt(Sb11.^2+Sb12.^2);
114 subplot(2, 2, 3);
115 imshow(Sb1);
116 title('Gaussian Noise with 0.01 variance');
117
118 Sb11 = convolution(I005, hSb11);
119 Sb12 = convolution(I005, hSb12);
120 Sb1 = sqrt(Sb11.^2+Sb12.^2);
121 subplot(2, 2, 4);
122 imshow(Sb1);
123 title('0.05 Gaussian Noise Added');

```

Listing 2: lab5q2.m file - function calls and plots from the Boundary Extraction method. Erosion and Dilation functions are listed below.

```

1 close all;
2
3 I = im2double(imread('chest.pgm'));
4 I0001 = imnoise(I, 'gaussian', 0, 0.001);
5 I001 = imnoise(I, 'gaussian', 0, 0.01);
6 I005 = imnoise(I, 'gaussian', 0, 0.05);
7 % Structuring element
8 SE1 = [ 0 1 0; 1 1 1; 0 1 0];
9 SE2 = [ 1 1 1; 1 1 1; 1 1 1];
10
11 dilatedImage = dilation (I, SE1);
12 erodedImage = erosion (I, SE1);
13
14 figure;
15 subplot (1,3,1);
16 imshow(I);
17 title ('Original Image');
18 subplot(1,3,2);
19 imshow(dilatedImage);
20 title('Dilated Image');
21 subplot(1,3,3);
22 imshow(erodedImage);
23 title('Eroded Image');
24
25
26
27 erodedImage0001 = erosion (I0001, SE1);

```

```

28 | erodedImage001 = erosion (I001, SE1);
29 | erodedImage005 = erosion (I005, SE1);
30 |
31 |
32 | figure;
33 | boundaryExtraction = I - erodedImage;
34 | subplot (2,2,1);
35 | imshow(boundaryExtraction);
36 | title('Boundary Extraction');
37 |
38 | boundaryExtraction = I0001 - erodedImage0001;
39 | subplot (2,2,2);
40 | imshow(boundaryExtraction);
41 | title('Gaussian Noise with 0.001 variance');
42 |
43 | boundaryExtraction = I001 - erodedImage001;
44 | subplot (2,2,3);
45 | imshow(boundaryExtraction);
46 | title('Gaussian Noise with 0.01 variance');
47 |
48 |
49 | boundaryExtraction = I005 - erodedImage005;
50 | subplot (2,2,4);
51 | imshow(boundaryExtraction);
52 | title('Gaussian Noise with 0.05 variance');
53 |
54 |
55 | % Using different SE
56 | erodedImage0001 = erosion (I0001, SE2);
57 | erodedImage001 = erosion (I001, SE2);
58 | erodedImage005 = erosion (I005, SE2);
59 |
60 |
61 | figure;
62 | boundaryExtraction = I - erodedImage;
63 | subplot (2,2,1);
64 | imshow(boundaryExtraction);
65 | title('Boundary Extraction');
66 |
67 | boundaryExtraction = I0001 - erodedImage0001;
68 | subplot (2,2,2);
69 | imshow(boundaryExtraction);
70 | title('Gaussian Noise with 0.001 variance');
71 |
72 | boundaryExtraction = I001 - erodedImage001;
73 | subplot (2,2,3);
74 | imshow(boundaryExtraction);
75 | title('Gaussian Noise with 0.01 variance');
76 |
77 |
78 | boundaryExtraction = I005 - erodedImage005;
79 | subplot (2,2,4);
80 | imshow(boundaryExtraction);
81 | title('Gaussian Noise with 0.05 variance');

```

Listing 3: dilation.m file

```

1 | function resultImage = dilation( I, SE )
2 |     resultImage = zeros(length(I(:,1)), length(I(1,:)));
3 |
4 |     for (i = 1:length(I(:,1)))
5 |         for (j = 1:length(I(1,:)))
6 |             [Y X] = meshgrid (-1:1, -1:1);
7 |             X = SE.*(X + i);
8 |             Y = SE.*(Y + j);
9 |
10 |             indexArray(:,1)= X;
11 |             indexArray(:,2)= Y;
12 |
13 |             max = -1;
14 |             for k = 1:3
15 |                 for l = 1:3
16 |                     if ((indexArray(k,l,1)>0 && indexArray(k,l,1) < length (I(:,1))) && ...
17 |                         (indexArray(k,l,2)>0 && indexArray(k,l,2) < length (I(1,:))))
18 |                         if (I(indexArray(k,l,1),indexArray(k,l,2)) > max)
19 |                             max = I(indexArray(k,l,1),indexArray(k,l,2));
20 |                         end
21 |                     end
22 |                 end

```

```

23         end
24         resultImage(i,j) = max;
25     end
26 end
27 end

```

Listing 4: erosion.m file

```

1 function resultImage = erosion( I, SE )
2     resultImage = zeros(length(I(:,1)), length(I(1,:)));
3
4     for(i = 1:length(I(:,1)))
5         for(j = 1:length(I(1,:)))
6             [Y X] = meshgrid (-1:1, -1:1);
7             X = SE.*(X + i);
8             Y = SE.*(Y + j);
9
10            indexArray(:,1)= X;
11            indexArray(:,2)= Y;
12
13            min = 1;
14            for k = 1:3
15                for l = 1:3
16                    if ((indexArray(k,l,1)>0 && indexArray(k,l,1) < length (I(:,1))) && ...
17                       (indexArray(k,l,2)>0 && indexArray(k,l,2) < length (I(1,:))))
18                        if (I(indexArray(k,l,1),indexArray(k,l,2)) < min)
19                            min = I(indexArray(k,l,1),indexArray(k,l,2));
20                        end
21                    end
22                end
23            end
24            resultImage(i,j) = min;
25        end
26    end
27 end

```

Listing 5: lab5q3.m file - function calls and plots using for the Canny Edge technique

```

1 close all;
2
3 I = im2double(imread('chest.pgm'));
4 I0001 = imnoise(I, 'gaussian', 0, 0.001);
5 I001 = imnoise(I, 'gaussian', 0, 0.01);
6 I005 = imnoise(I, 'gaussian', 0, 0.05);
7
8 figure;
9 subplot(2, 2, 1);
10 BW = edge(I, 'Canny');
11 imshow(BW);
12 title('Canny - default parameters');
13
14 subplot(2, 2, 2);
15 BW = edge(I0001, 'Canny');
16 imshow(BW);
17 title('Gaussian Noise with 0.001 variance');
18
19 subplot(2, 2, 3);
20 BW = edge(I001, 'Canny');
21 imshow(BW);
22 title('Gaussian Noise with 0.01 variance');
23
24 subplot(2, 2, 4);
25 BW = edge(I005, 'Canny');
26 imshow(BW);
27 title('Gaussian Noise with 0.05 variance');
28
29 %*****
30 figure;
31 subplot(2, 2, 1);
32 BW = edge(I, 'Canny', [0.1 0.25]);
33 imshow(BW);
34 title('Canny - Threshold 0.1 / 0.25');
35
36 subplot(2, 2, 2);
37 BW = edge(I0001, 'Canny', [0.1 0.25]);
38 imshow(BW);
39 title('Gaussian Noise with 0.001 variance');
40

```

```

41 subplot(2, 2, 3);
42 BW = edge(I001, 'Canny', [0.1 0.25]);
43 imshow(BW);
44 title('Gaussian Noise with 0.01 variance');
45
46 subplot(2, 2, 4);
47 BW = edge(I005, 'Canny', [0.1 0.25]);
48 imshow(BW);
49 title('Gaussian Noise with 0.05 variance');
50
51 %%%%%%%%%
52 figure;
53 subplot(2, 2, 1);
54 BW = edge(I, 'Canny', [0.1 0.25], sqrt(4));
55 imshow(BW);
56 title('Threshold 0.1/0.25 - Sigma = sqrt(4)');
57
58 subplot(2, 2, 2);
59 BW = edge(I0001, 'Canny', [0.1 0.25], sqrt(4));
60 imshow(BW);
61 title('Gaussian Noise with 0.001 variance');
62
63 subplot(2, 2, 3);
64 BW = edge(I001, 'Canny', [0.1 0.25], sqrt(4));
65 imshow(BW);
66 title('Gaussian Noise with 0.01 variance');
67
68 subplot(2, 2, 4);
69 BW = edge(I005, 'Canny', [0.1 0.25], sqrt(4));
70 imshow(BW);
71 title('Gaussian Noise with 0.05 variance');
72 %%%%%%%%%
73 figure;
74 subplot(2, 2, 1);
75 BW = edge(I, 'Canny', [0.1 0.25], sqrt(8));
76 imshow(BW);
77 title('Threshold 0.1/0.25 - Sigma = sqrt(8)');
78
79 subplot(2, 2, 2);
80 BW = edge(I0001, 'Canny', [0.1 0.25], sqrt(8));
81 imshow(BW);
82 title('Gaussian Noise with 0.001 variance');
83
84 subplot(2, 2, 3);
85 BW = edge(I001, 'Canny', [0.1 0.25], sqrt(8));
86 imshow(BW);
87 title('Gaussian Noise with 0.01 variance');
88
89 subplot(2, 2, 4);
90 BW = edge(I005, 'Canny', [0.1 0.25], sqrt(8));
91 imshow(BW);
92 title('Gaussian Noise with 0.05 variance');
93
94 %%%%%%%%%
95 figure;
96 subplot(2, 2, 1);
97 BW = edge(I, 'Canny', [0.3 0.5]);
98 imshow(BW);
99 title('Canny - Threshold 0.3 / 0.5');
100
101 subplot(2, 2, 2);
102 BW = edge(I0001, 'Canny', [0.3 0.5]);
103 imshow(BW);
104 title('Gaussian Noise with 0.001 variance');
105
106 subplot(2, 2, 3);
107 BW = edge(I001, 'Canny', [0.3 0.5]);
108 imshow(BW);
109 title('Gaussian Noise with 0.01 variance');
110
111 subplot(2, 2, 4);
112 BW = edge(I005, 'Canny', [0.3 0.5]);
113 imshow(BW);
114 title('Gaussian Noise with 0.05 variance');

```