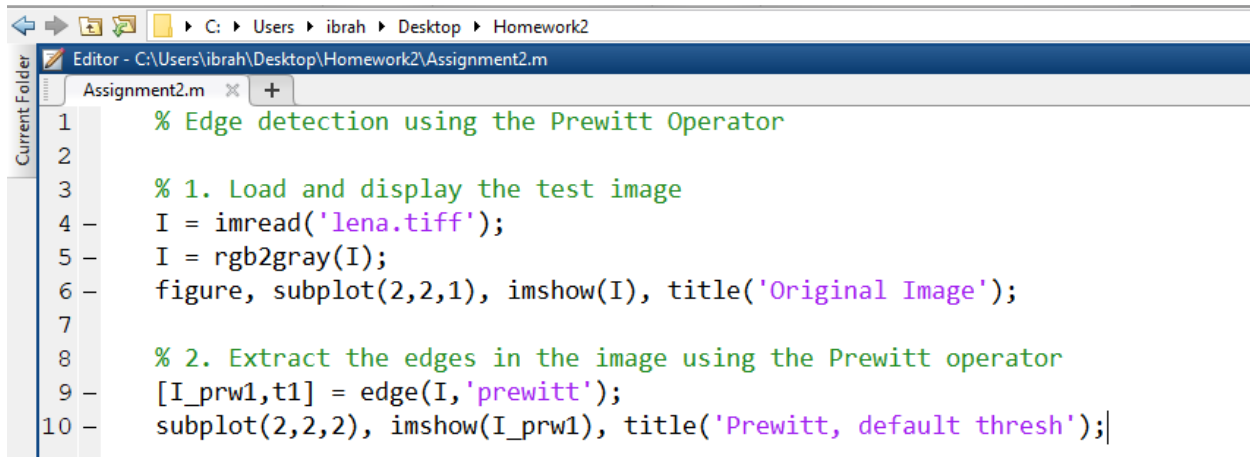


Digital Image Processing

Assignment #2

Edge Detection Using the Prewitt Operator



```
1 % Edge detection using the Prewitt Operator
2
3 % 1. Load and display the test image
4 I = imread('lena.tiff');
5 I = rgb2gray(I);
6 figure, subplot(2,2,1), imshow(I), title('Original Image');
7
8 % 2. Extract the edges in the image using the Prewitt operator
9 [I_prw1,t1] = edge(I,'prewitt');
10 subplot(2,2,2), imshow(I_prw1), title('Prewitt, default thresh');
```

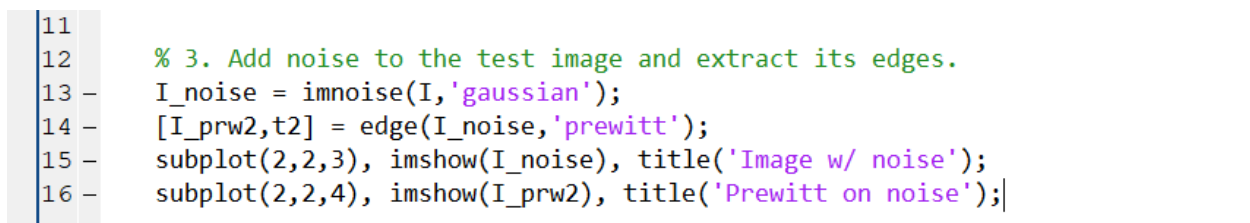
Code Fragment 1: Steps 1 and 2 applied.

Question 1

What does the **t1** variable represent?

Answer 1

The variable **t1** represent the threshold value. ■



```
11
12 % 3. Add noise to the test image and extract its edges.
13 I_noise = imnoise(I,'gaussian');
14 [I_prw2,t2] = edge(I_noise,'prewitt');
15 subplot(2,2,3), imshow(I_noise), title('Image w/ noise');
16 subplot(2,2,4), imshow(I_prw2), title('Prewitt on noise');
```

Code Fragment 2: Step 3 applied.

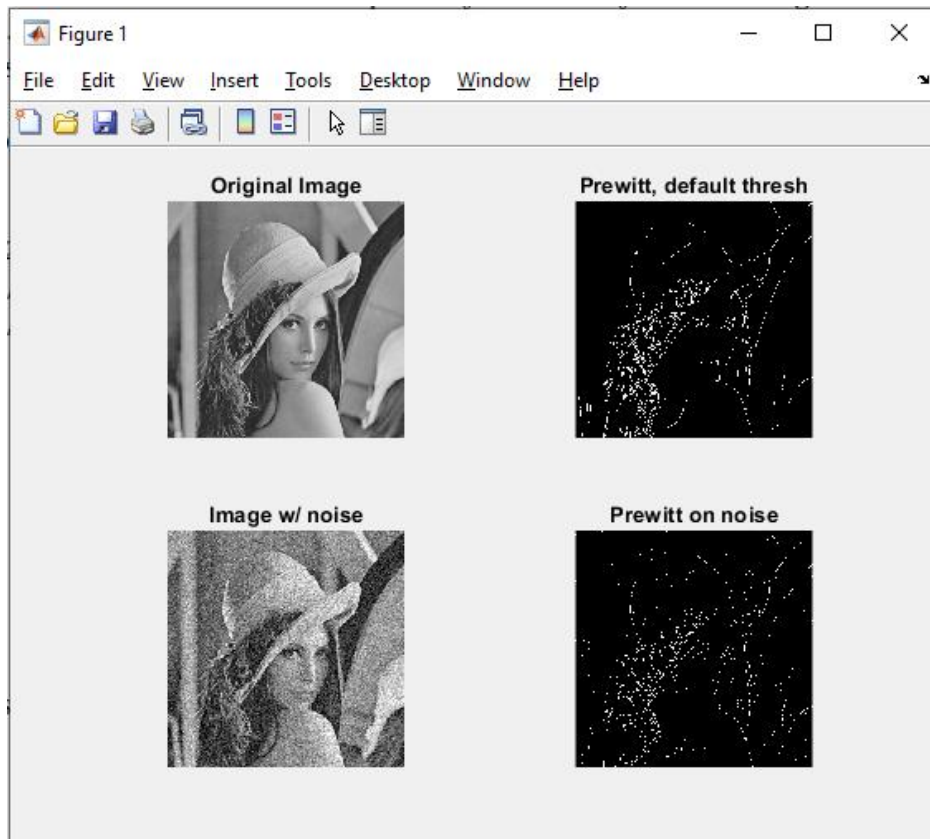


Figure 1: Result of *Edge Detection* using the **Prewitt** operator

Question 2

How did the *Prewitt edge detector* perform in the presence of noise (compared to no noise)?

Answer 2

The *Prewitt edge detection* will be worse, meaning that the edges are not accurate, when the input image has noise. A better edge detection, in other words, edge detection with higher accuracy occurs when the input image does not have noise. ■

Question 3

Did MATLAB use a different threshold value for the noisy image?

Answer 3

Yes, MATLAB uses a different threshold value.

t1 = 0.0796 whereas **t2 = 0.1394** ■

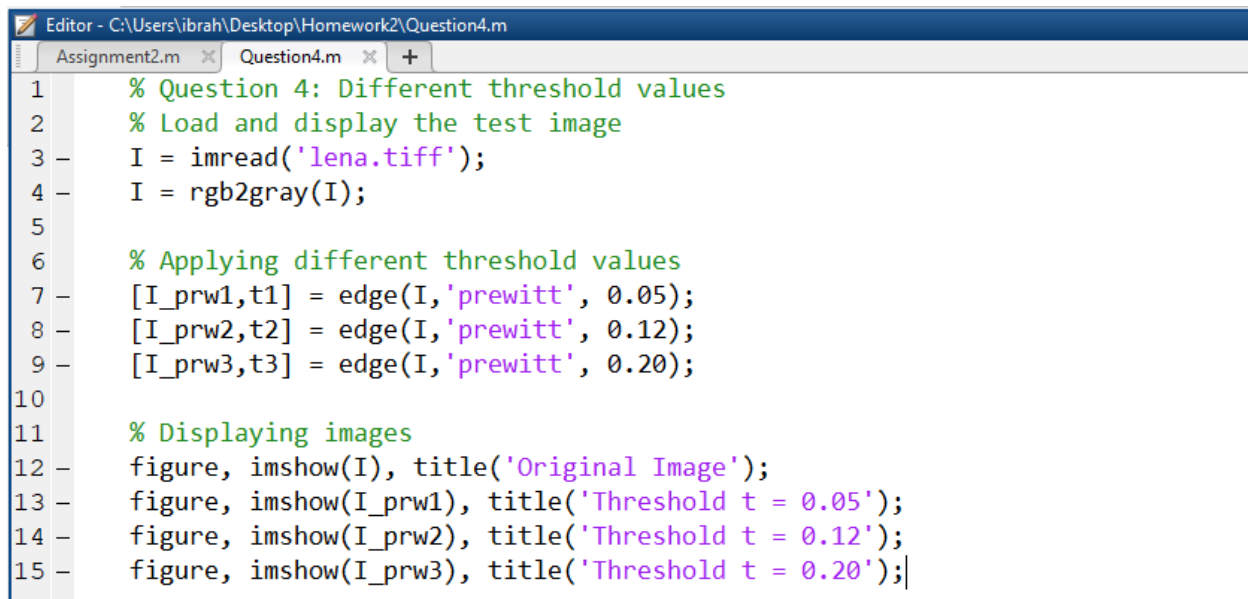
Question 4

Try using different threshold values. Do these different values affect the operator's response to noise? How does the threshold value affect the edges of the object?

Answer 4

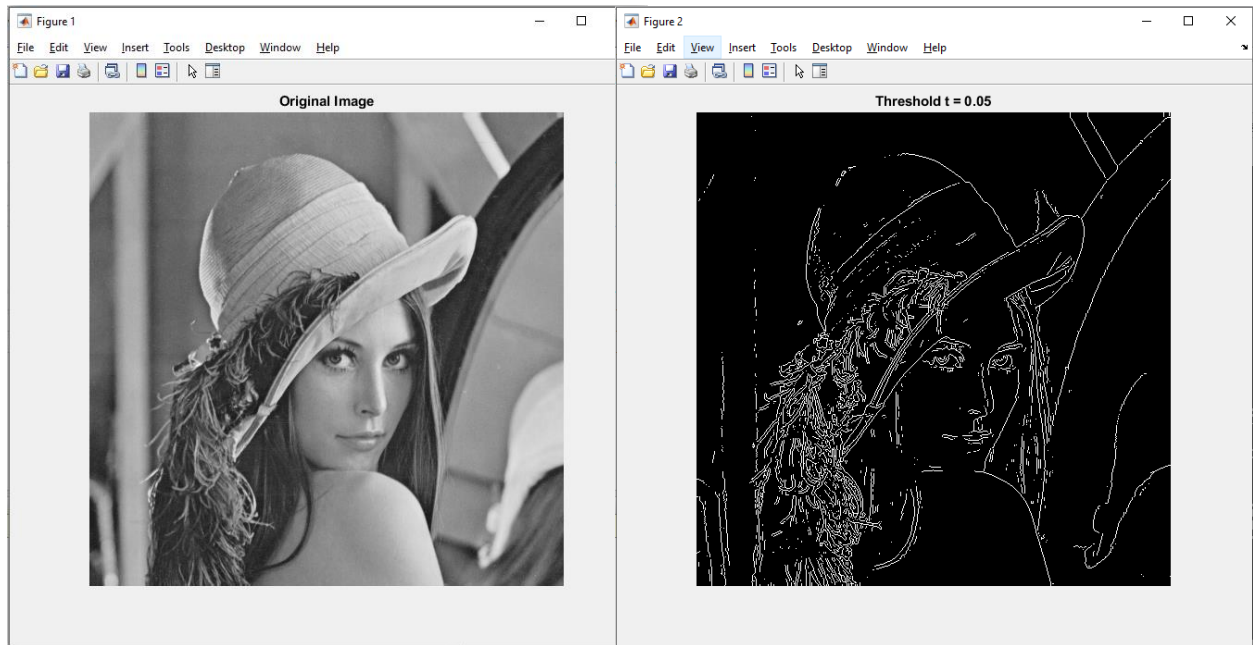
In a new MATLAB file called Question4.m, I have taken original image **lena.tiff** and converted it to its corresponding grayscale image for *Prewitt edge detection* to be work properly. Then, I have performed *Prewitt edge detection* with **edge** function, by introducing a new argument in function called **threshold**. This value will be stored in **t1**, **t2** and **t3** as shown below.

The threshold value will determine how much of the edges will be appearing on the screen. As the value of threshold increases, we get less edges, as it is shown in the Figure 2. ■

The image shows a MATLAB editor window with the title bar 'Editor - C:\Users\ibrah\Desktop\Homework2\Question4.m'. There are two tabs open: 'Assignment2.m' and 'Question4.m'. The code in the 'Question4.m' tab is as follows:

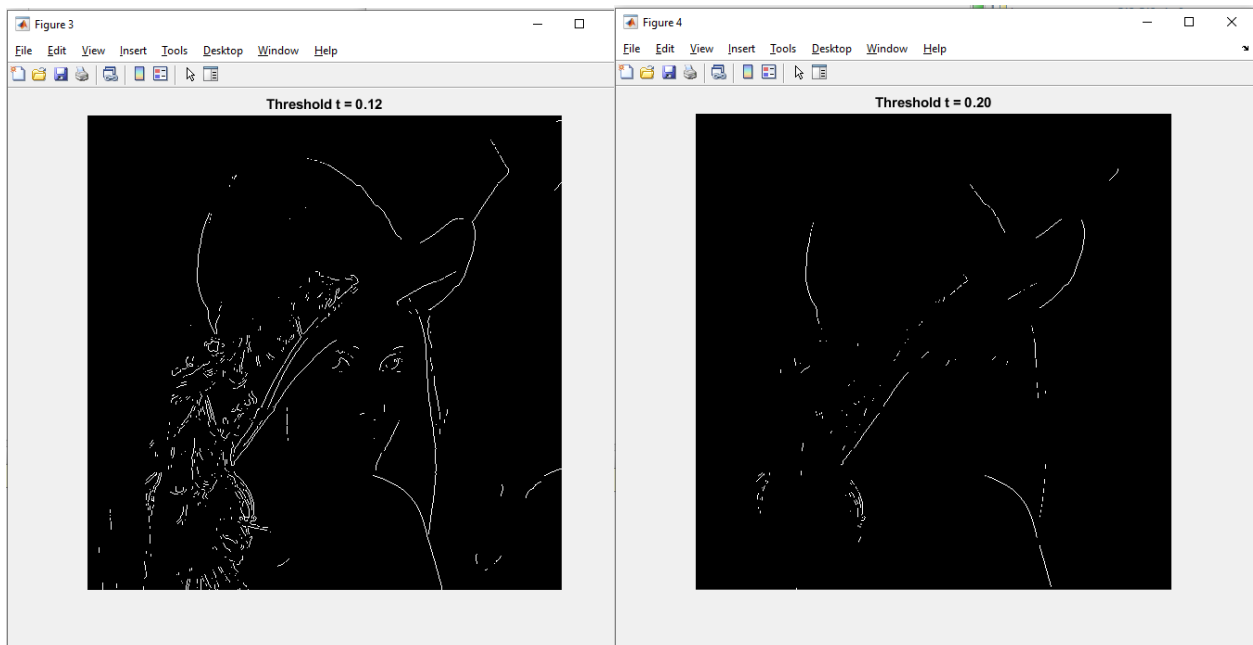
```
1 % Question 4: Different threshold values
2 % Load and display the test image
3 I = imread('lena.tiff');
4 I = rgb2gray(I);
5
6 % Applying different threshold values
7 [I_prw1,t1] = edge(I,'prewitt', 0.05);
8 [I_prw2,t2] = edge(I,'prewitt', 0.12);
9 [I_prw3,t3] = edge(I,'prewitt', 0.20);
10
11 % Displaying images
12 figure, imshow(I), title('Original Image');
13 figure, imshow(I_prw1), title('Threshold t = 0.05');
14 figure, imshow(I_prw2), title('Threshold t = 0.12');
15 figure, imshow(I_prw3), title('Threshold t = 0.20');
```

Code Fragment 3: Different threshold values



(a)

(b)



(c)

(d)

Figure 2: Original Image (a),
Prewitt Edge Detection with $t = 0.05$,
Prewitt Edge Detection with $t = 0.12$, and
Prewitt Edge Detection with $t = 0.20$

Edge Detection Using the Sobel Operator

```
Editor - C:\Users\ibrah\Desktop\Homework2\Sober.m
Sober.m  Assignment2.m  +
1  % Edge detection using the Sober Operator
2
3  I = imread('lena.tiff');
4  I = rgb2gray(I);
5  figure, subplot(2,2,1), imshow(I), title('Original Image');
6
7  % 4. Extract the edges from the test image using the Sobel edge detector.
8  [I_sob1,t1] = edge(I,'sobel');
9  subplot(2,2,2), imshow(I_sob1), title('Sobel, default thresh');
10
11 % 5. Extract the edges from the test image with Gaussian noise
12 % using the Sobel edge detector.
13
14 I_noise = imnoise(I,'gaussian');
15 [I_sob2,t2] = edge(I_noise,'sobel');
16 subplot(2,2,3), imshow(I_noise), title('Image w/ noise');
17 subplot(2,2,4), imshow(I_sob2), title('Sobel on noise');
```

Code Fragment 4: Steps 4 and 5 applied.

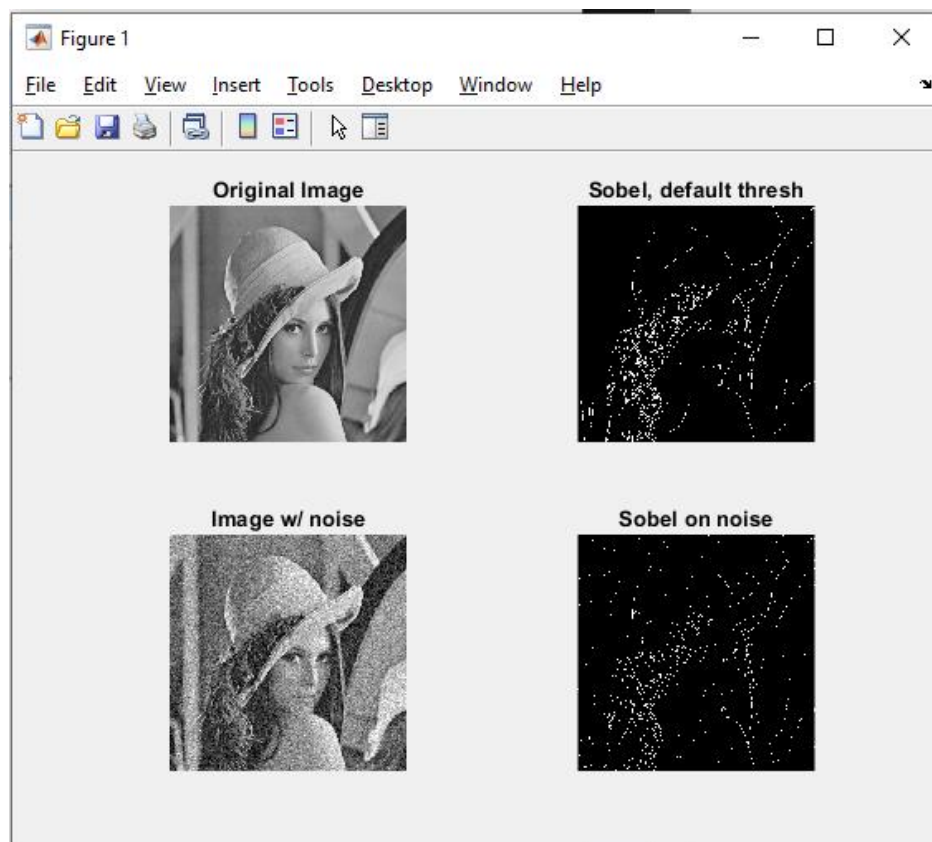


Figure 3: Result of *Edge Detection* using the **Sobel** operator

Question 5

How does the **Sobel** operator compare with the **Prewitt** operator with and without noise?

Answer 5

When the input image does not have noise, **Sobel** operation will give a result with more accurate edge detection, however, when it comes to input image with noise, the output edge detection accuracy is lower. This behavior is also observed in **Prewitt** operator. ■

```
18
19 % 6. Extract the edges from the test image with the Sobel operator
20 % with no thinning.
21
22 - I_sob3 = edge(I,'sobel','nothinning');
23 - figure, subplot(2,1,1), imshow(I_sob1), title('Thinning');
24 - subplot(2,1,2), imshow(I_sob3), title('No Thinning');
```

Code Fragment 5: Step 6 applied.



Figure 4: Thinning vs no Thinning

```

10
11 % 7. Display the horizontal and vertical convolution results
12 % from the Sobel operator.
13
14 - [I_sob4,t,I_sobv,I_sobh] = edge(I,'sobel');
15 - figure, subplot(2,2,1), imshow(I), title('Original Image');
16 - subplot(2,2,2), imshow(I_sob4), title('Complete Sobel');
17 - subplot(2,2,3), imshow(abs(I_sobv),[]), title('Sobel Vertical');
18 - subplot(2,2,4), imshow(abs(I_sobh),[]), title('Sobel Horizontal');

```

Code Fragment 6: Step 7 applied.

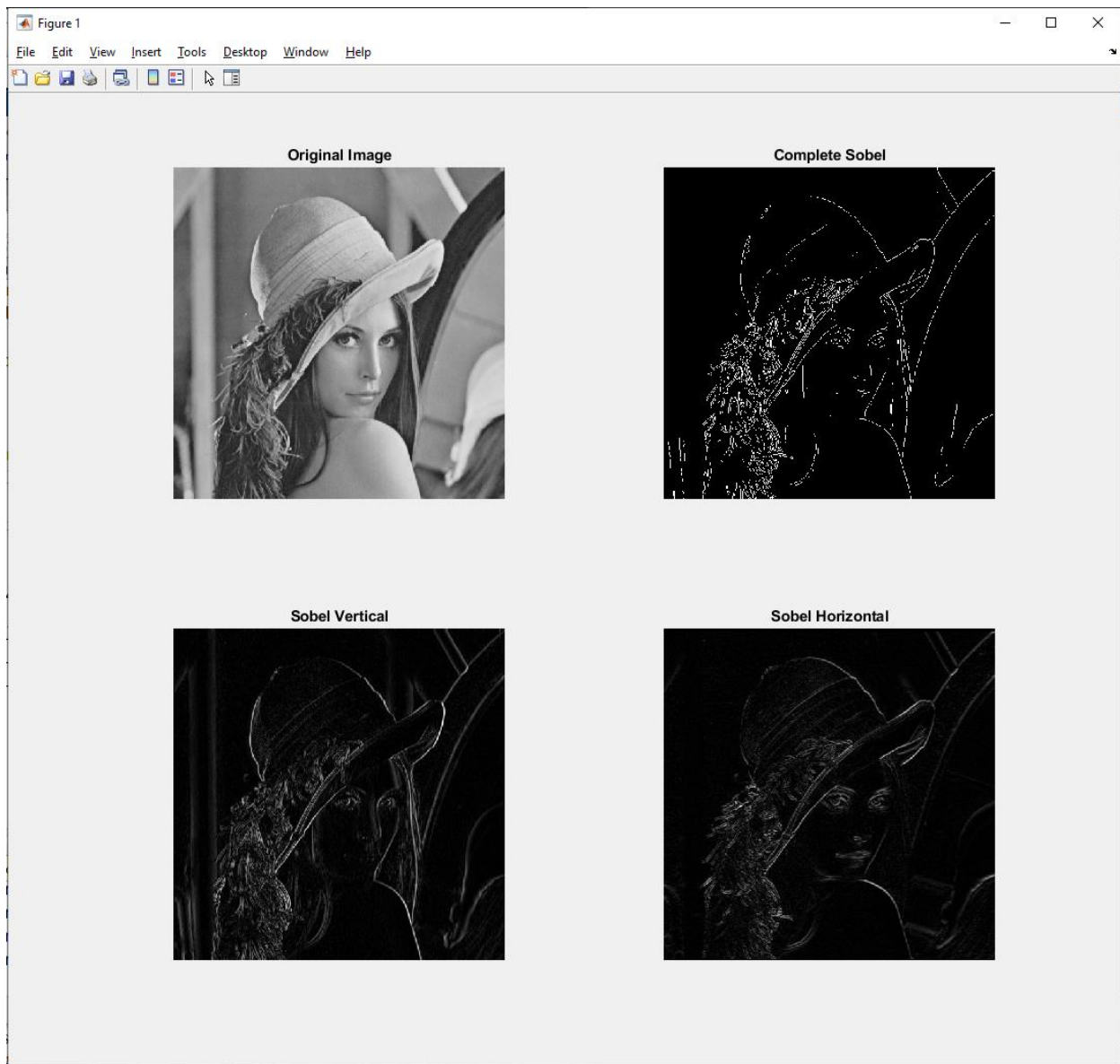


Figure 5: Results of *Horizontal* and *vertical* convolutions from **Sobel** operator

Question 6

Why do we display the absolute value of the vertical and horizontal images? Hint: Inspect the minimum and maximum values of these images.

Answer 6

The **abs** function takes the absolute value of given input and it is needed because of negative values along with positive values returned horizontal and vertical **sobel** function. ■

Question 7

Change the code in step 7 to display thresholded (binarized), not thinned, versions of all images.

Answer 7

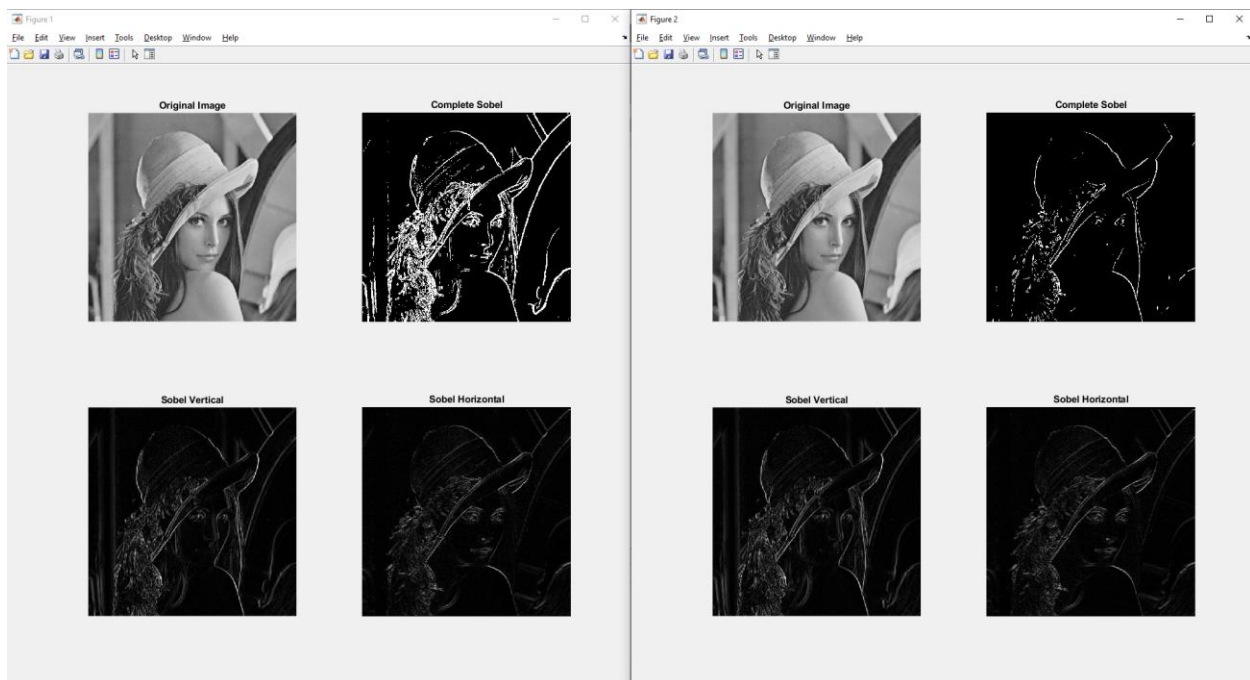


Figure 6: Results of different threshold values for **Sobel** operator and their effect (0.05 on the left, 0.12 on the right)

As it is shown by Figure 6, the edge function returns the vertical and horizontal images before any thresholding takes place. Therefore, two images in the bottom are not affected and identical and only image on upper right corner changes due to threshold value. ■

Edge Detection Using the Roberts Operator

```
Editor - C:\Users\ibrah\Desktop\Homework2\roberts_operator.m
Sober.m Assignment2.m SobelThinning.m roberts_operator.m +
1 % Edge detection using the Roberts Operator
2
3 I = imread('lena.tiff');
4 I = rgb2gray(I);
5 I_noise = imnoise(I,'gaussian');
6
7 % 8. Extract the edges from the original image using the Roberts operator.
8
9 I_rob1 = edge(I,'roberts');
10 figure,subplot(2,2,1), imshow(I), title('Original Image');
11 subplot(2,2,2), imshow(I_rob1), title('Roberts, default thresh');
12
13 % 9. Apply the Roberts operator to a noisy image.
14 [I_rob2,t] = edge(I_noise,'roberts');
15 subplot(2,2,3), imshow(I_noise), title('Image w/ noise');
16 subplot(2,2,4), imshow(I_rob2), title('Roberts on noise');
```

Code Fragment 7: Steps 8 and 9 applied

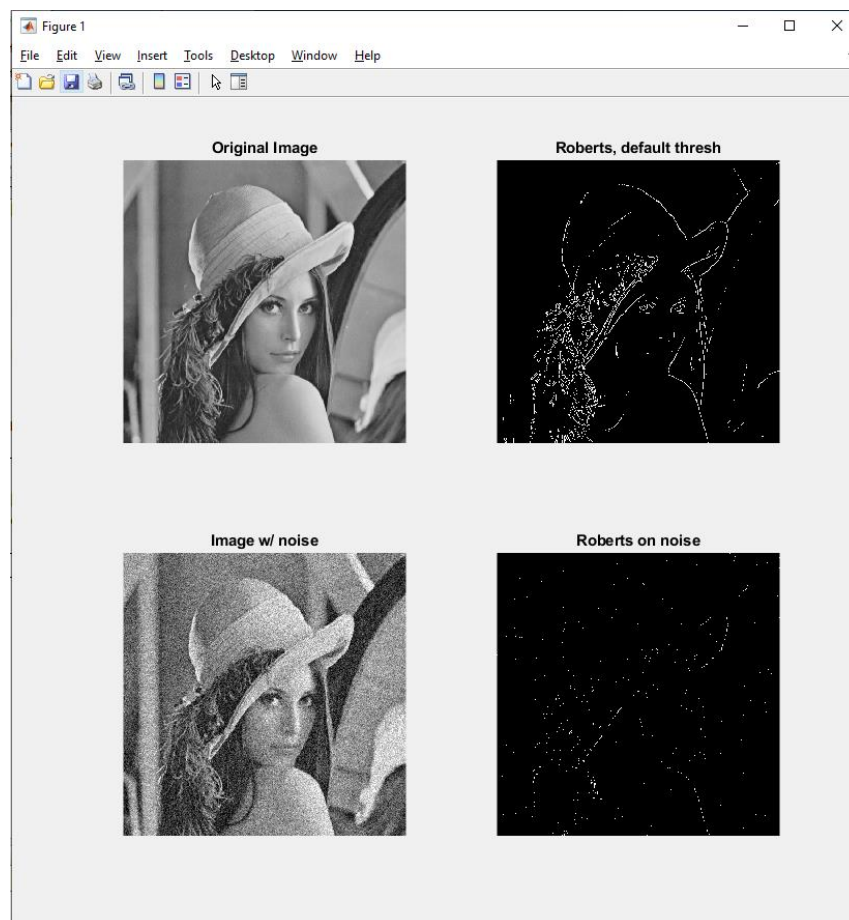


Figure 7: Results of *Edge Detection* using the **Robert** operator

Question 8

Compare the **Roberts** operator with the **Sobel** and **Prewitt** operators. How does it hold up to noise?

Answer 8

Although the behavior of **Roberts** operator is similar to **Sobel** and **Prewitt** operator (higher accuracy of edge detection on non-noised image and lower accuracy of edge detection on noised image), its edge detection is a lot worse when noised image is given as input compared to **Sobel** and **Prewitt** operator. ■

Question 9

If we were to adjust the threshold, would we get better results when filtering the noisy image?

Answer 9

Since the Roberts operator is very sensitive to noise (because of to the small kernel), if we had a lowered noised image, then the results would be better. ■

Question 10

Suggest a method to reduce the noise in the image before performing edge detection.

Answer 10

Some *Noise Removal* techniques can be applied such as *linear filtering* or *averaging Filter* to reduce the noise in the image. ■

Edge Detection with the Laplacian of a Gaussian Operator

```
Editor - C:\Users\ibrah\Desktop\Homework2\laplacian.m
Sober.m Assignment2.m roberts_operator.m laplacian.m +
1 - I = imread('lena.tiff'); % Edge detection with Laplacian of Gaussian
2 - I = rgb2gray(I);
3 - I_noise = imnoise(I,'gaussian');
4
5 % 10. Extract the edges from the original image using the LoG edge detector
6 - I_log1 = edge(I,'log');
7 - figure, subplot(2,2,1), imshow(I), title('Original Image');
8 - subplot(2,2,2), imshow(I_log1), title('LoG, default parameters');
9
10 % 11. Apply the LoG edge detector to the noisy image.
11 - [I_log2,t] = edge(I_noise,'log');
12 - subplot(2,2,3), imshow(I_noise), title('Image w/ noise');
13 - subplot(2,2,4), imshow(I_log2), title('LoG on noise');
```

Code Fragment 8: Steps 10 and 11 applied

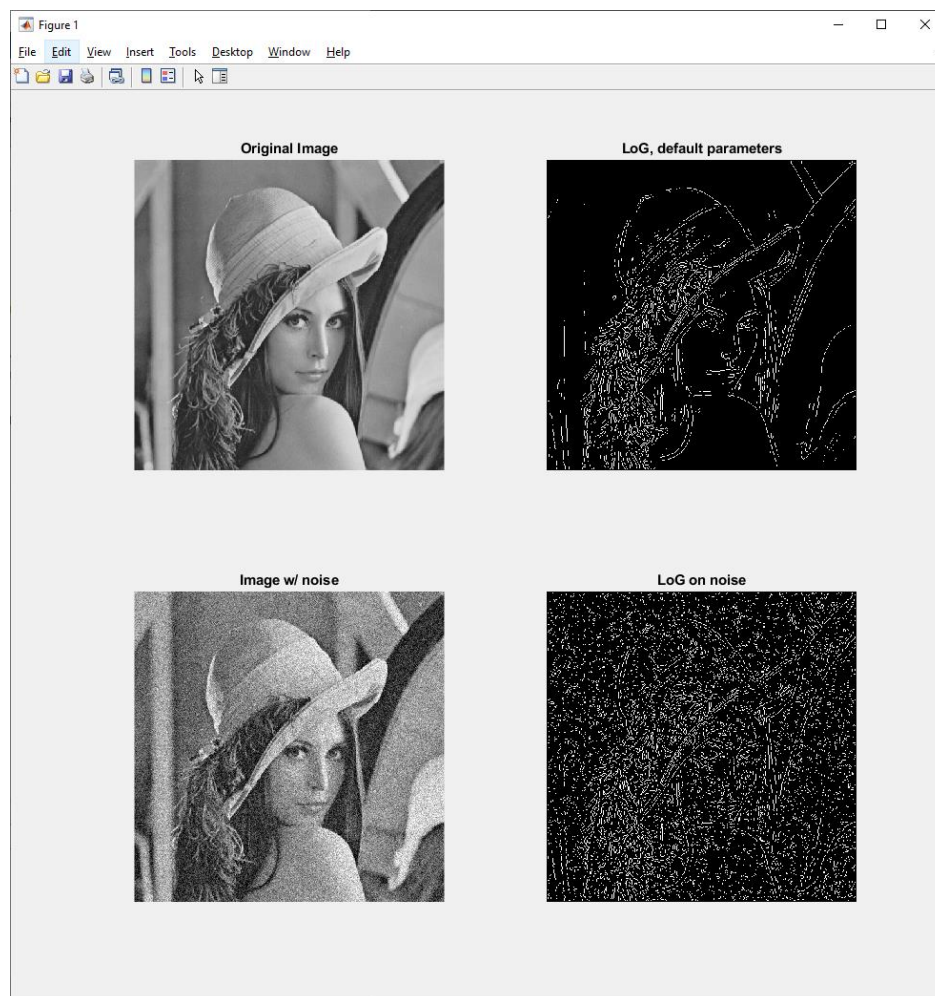


Figure 7: Results of *Edge Detection* using the **Laplacian of a Gaussian** operator

Question 11

By default, the LoG edge detector uses a value of 2 for σ (the standard deviation of the filter). What happens when we increase this value?

Answer 11

```
15 - [I_log3,t] = edge(I_noise,'log', t, 2);
16 - figure, subplot(2,2,1), imshow(I_log3), title('LoG on noise, Sigma = 2');
17
18 - [I_log4,t] = edge(I_noise,'log', t, 2.5);
19 - subplot(2,2,2), imshow(I_log4), title('LoG on noise, Sigma = 2.5');
20
21 - [I_log5,t] = edge(I_noise,'log', t, 3);
22 - subplot(2,2,3), imshow(I_log5), title('LoG on noise, Sigma = 3');
23
24 - [I_log5,t] = edge(I_noise,'log', t, 3.5);
25 - subplot(2,2,4), imshow(I_log5), title('LoG on noise, Sigma = 3.5');
```

Code Fragment 9: Different σ values with LoG Edge Detection
($\sigma = 2$, $\sigma = 2.5$, $\sigma = 3$ and $\sigma = 3.5$)

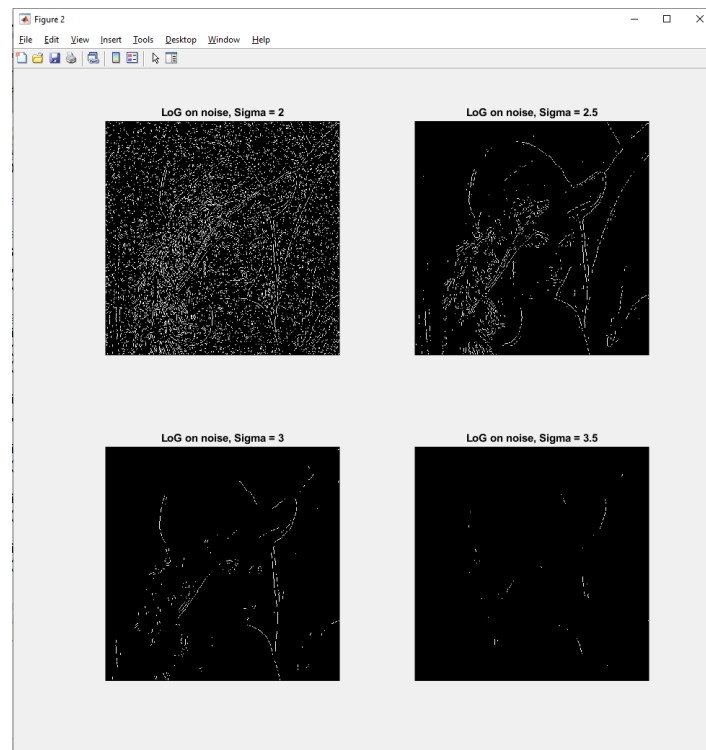


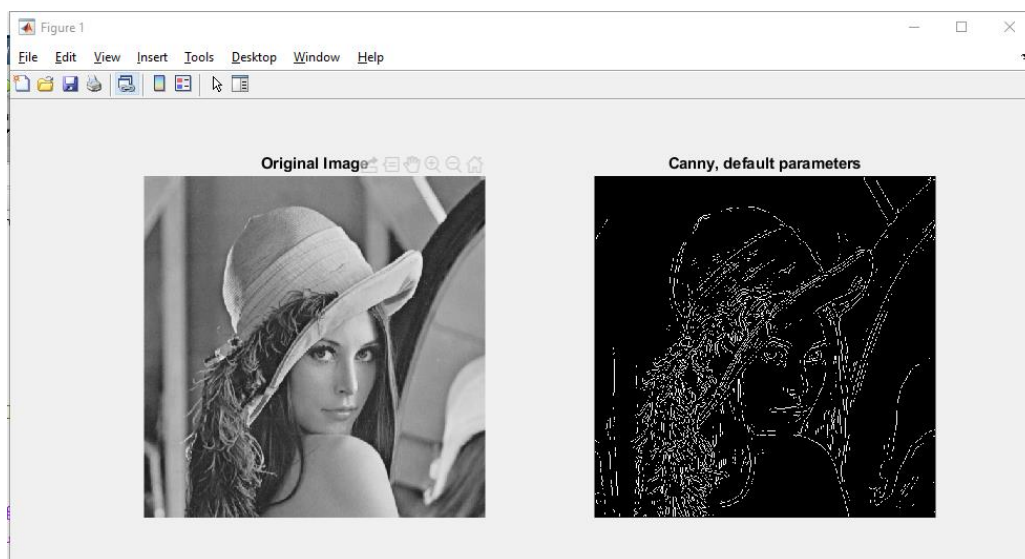
Figure 8: Results of different σ values in LoG Edge Detection

As the values of σ increases, we are getting less edges and after around $\sigma = 4.0$, no edges are appearing on the screen. ■

Edge Detection with the Canny Operator

```
Editor - C:\Users\ibrah\Desktop\Homework2\canny.m
Sober.m Assignment2.m roberts_operator.m laplacian.m canny.m +
1 % Edge detection using the Canny Operator
2
3 I = imread('lena.tiff');
4 I = rgb2gray(I);
5 I_noise = imnoise(I,'gaussian');
6 I_log1 = edge(I,'log');
7
8 % 12. Extract the edges from the original image
9 % using the Canny edge detector.
10
11 I_can1 = edge(I,'canny');
12 figure,subplot(2,2,1), imshow(I), title('Original Image');
13 subplot(2,2,2), imshow(I_log1), title('Canny, default parameters');
14
15 % 13. Apply the filter to the noisy image.
16
17 [I_can2,t] = edge(I_noise,'canny', [], 2.5);
18 subplot(2,2,3), imshow(I_noise), title('Image w/ noise');
19 subplot(2,2,4), imshow(I_can2), title('Canny on noise');
```

Code Fragment 9: Steps 12 and 13 applied



(a)



(b)

Figure 9: Results of *Edge Detection* using the **Canny** operator (a) and (b)

```

21 % 14. Apply the Canny detector on the noisy image where sigma = 2.
22
23 [I_can3,t] = edge(I_noise,'canny', [], 2);
24 figure
25 subplot(1,2,1), imshow(I_can2), title('Canny, default parameters');
26 subplot(1,2,2), imshow(I_can3), title('Canny, sigma = 2');

```

Code Fragment 10: Step 14 applied

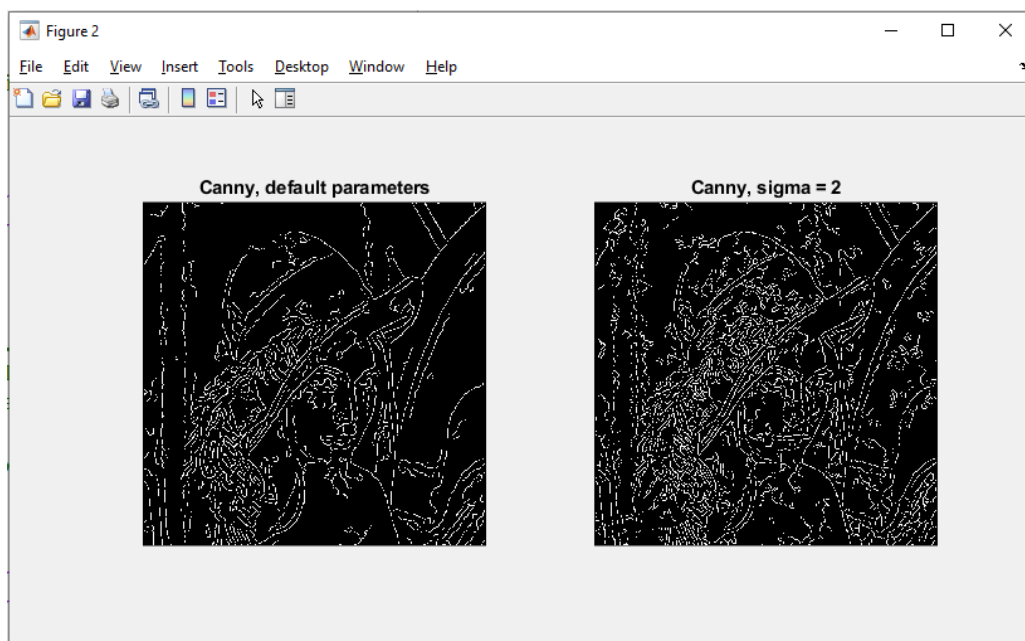


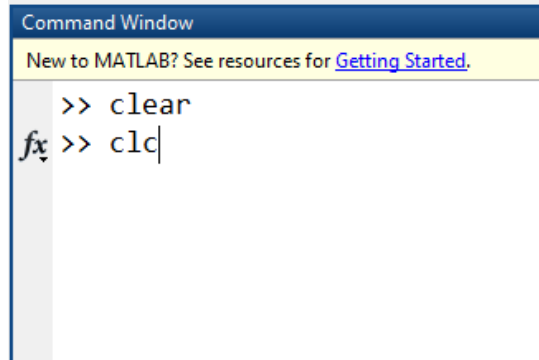
Figure 10: Results of **Canny** operator,
default parameters vs $\sigma = 2$

Question 12

Does increasing the value of sigma give us better results when using the Canny detector on a noisy image?

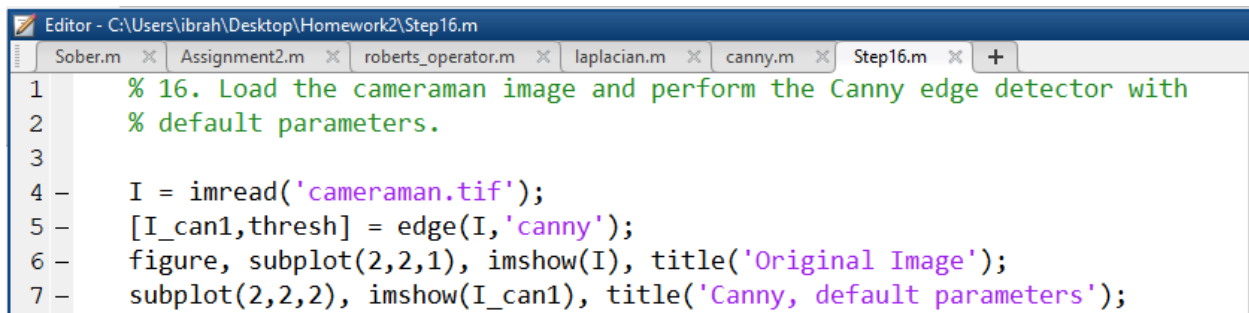
Answer 12

No, increasing the sigma value did not give a better result, it shows more edges, but the clarity of the image has been reduced. ■



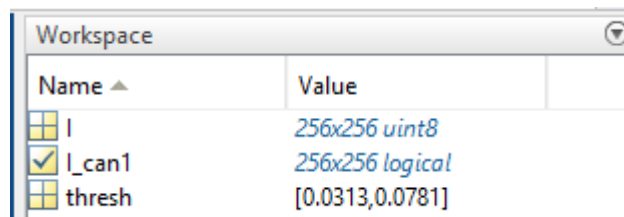
```
Command Window
New to MATLAB? See resources for Getting Started.
>> clear
fx >> clc|
```

Code Fragment 11: Step 15 applied



```
Editor - C:\Users\ibrah\Desktop\Homework2\Step16.m
Sober.m x Assignment2.m x roberts_operator.m x laplacian.m x canny.m x Step16.m x +
1 % 16. Load the cameraman image and perform the Canny edge detector with
2 % default parameters.
3
4 - I = imread('cameraman.tif');
5 - [I_can1,thresh] = edge(I,'canny');
6 - figure, subplot(2,2,1), imshow(I), title('Original Image');
7 - subplot(2,2,2), imshow(I_can1), title('Canny, default parameters');
```

Code Fragment 12: Step 16 applied



Name ▲	Value
I	256x256 uint8
<input checked="" type="checkbox"/> I_can1	256x256 logical
thresh	[0.0313,0.0781]

Code Fragment 13: Step 17 applied


```

Editor - C:\Users\ibrah\Desktop\Homework2\Step16.m
Sober.m x Assignment2.m x roberts_operator.m x laplacian.m x canny.m x Step16.m x +
1 % 16. Load the cameraman image and perform the Canny edge detector with
2 % default parameters.
3
4 I = imread('cameraman.tif');
5 [I_can1,thresh] = edge(I,'canny');
6 figure, subplot(2,2,1), imshow(I), title('Original Image');
7 subplot(2,2,2), imshow(I_can1), title('Canny, default parameters');
8
9 % 18. Use a threshold value higher than the one in variable thresh.
10
11 [I_can2,thresh] = edge(I, 'canny', 0.4);
12 subplot(2,2,3), imshow(I_can2), title('Canny, thresh = 0.4');
13
14 % 19. Use a threshold value lower than the one in variable thresh.
15
16 [I_can2,thresh] = edge(I, 'canny', 0.08);
17 subplot(2,2,4), imshow(I_can2), title('Canny, thresh = 0.08');

```

Code Fragment 13: Steps 18 and 19 applied

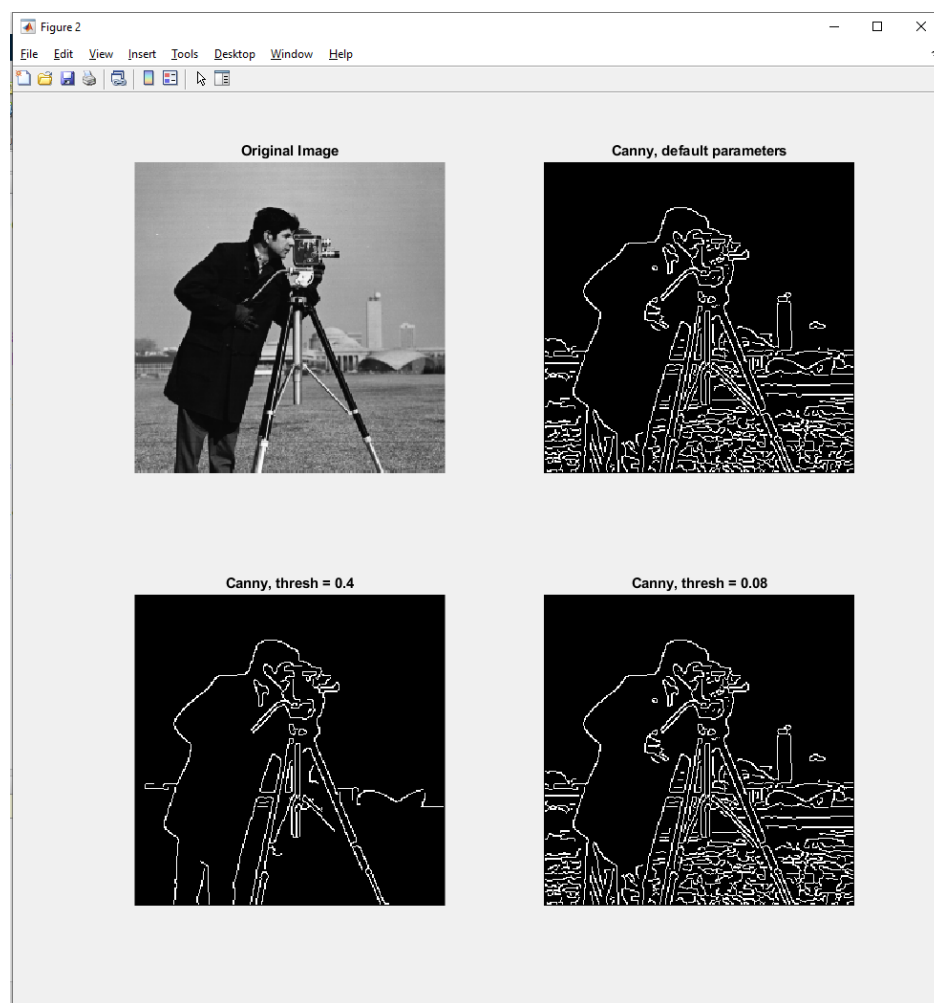


Figure 11: Results of **Canny** operator,
default threshold, threshold = **0.4** and threshold = **0.08**

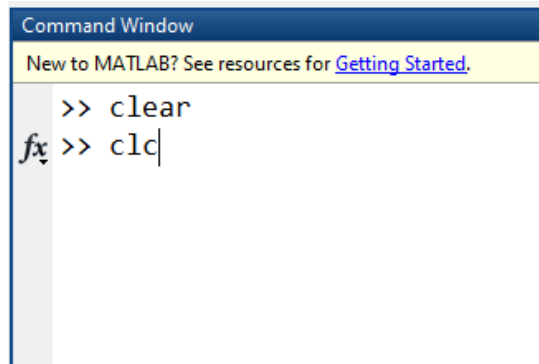
Question 13

How does the sensitivity of the Canny edge detector change when the threshold value is increased?

Answer 13

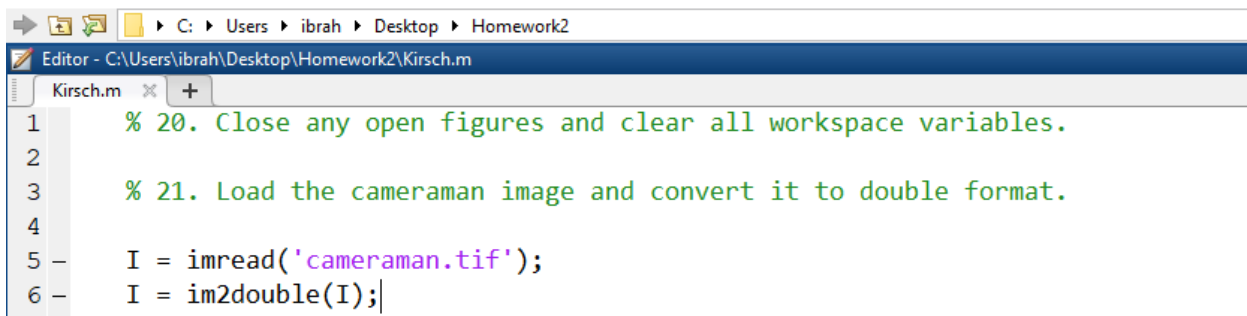
We have lost some of the details. In other words, number of edges decreased. ■

Edge Detection with the Kirsch Operator



```
Command Window
New to MATLAB? See resources for Getting Started.
>> clear
fx >> clc|
```

Code Fragment 14: Step 20 applied



```
Editor - C:\Users\ibrah\Desktop\Homework2\Kirsch.m
Kirsch.m
1 % 20. Close any open figures and clear all workspace variables.
2
3 % 21. Load the cameraman image and convert it to double format.
4
5 - I = imread('cameraman.tif');
6 - I = im2double(I);|
```

Code Fragment 15: Step 21 applied

```

9      % 22. Create the Kirsch masks and store them in a preallocated matrix.
10
11 -   k = zeros(3,3,8);
12 -   k(:,:,1) = [-3 -3 5; -3 0 5; -3 -3 5];
13 -   k(:,:,2) = [-3 5 5; -3 0 5; -3 -3 -3];
14 -   k(:,:,3) = [5 5 5; -3 0 -3; -3 -3 -3];
15 -   k(:,:,4) = [5 5 -3; 5 0 -3; -3 -3 -3];
16 -   k(:,:,5) = [5 -3 -3; 5 0 -3; 5 -3 -3];
17 -   k(:,:,6) = [-3 -3 -3; 5 0 -3; 5 5 -3];
18 -   k(:,:,7) = [-3 -3 -3; -3 0 -3; 5 5 5];
19 -   k(:,:,8) = [-3 -3 -3; -3 0 5; -3 5 5];

```

Code Fragment 16: Step 22 applied

```

20
21      % 23. Convolve each mask with the image using a for loop.
22 -   I_k = zeros(size(I,1), size(I,2), 8);
23 -   for i = 1:8
24 -       I_k(:,:,i) = imfilter(I,k(:,:,i));
25 -   end
26

```

Code Fragment 17: Step 23 applied

```

26
27      % 24. Display the resulting images.
28 -   figure
29 -   for j = 1:8
30 -       subplot(2,4,j), imshow(abs(I_k(:,:,j)),[]), ...
31 -       title(['Kirsch mask ', num2str(j)]);
32 -   end

```

Code Fragment 18: Step 24 applied



Figure 12: Results of **Kirsch** operator

Question 14

Why are we required to display the absolute value of each mask? Hint: Inspect the minimum and maximum values.

Answer 14

Because each and every mask contains minimum of negative values and it is required to take the absolute value for edge detection to work. ■

Question 15

How did we dynamically display the mask number when displaying all eight images?

Answer 15

The 8 images are stored in a data structure called array. For variable **j**, starting from **1** to **8**, we perform operations for each image iteratively, as **Kirsch** masks stored in a **3 x 3 x 8** matrix. ■

```
34 % 25. Find the maximum values.  
35  
36 - I_kir = max(I_k,[],3);  
37 - figure, imshow(I_kir,[]);
```

Code Fragment 18: Step 25 applied

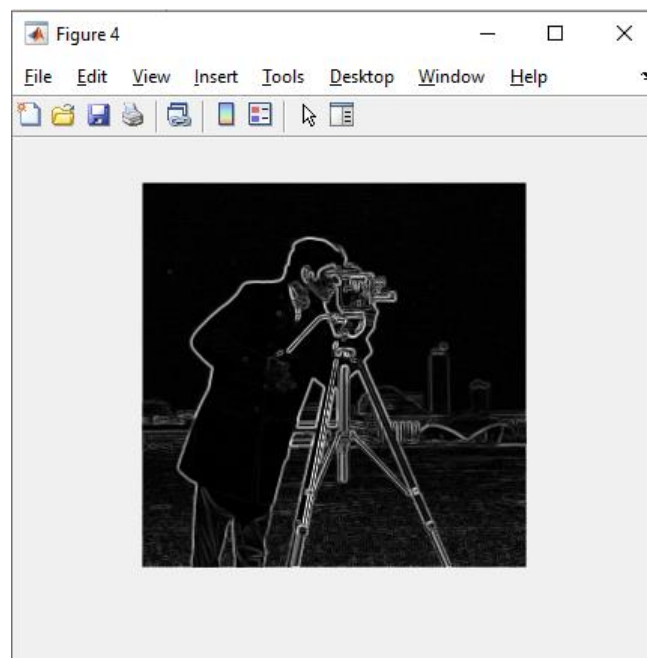


Figure 13: Result of Step 25

Question 16

When calculating the maximum values, what does the last parameter in the max function call mean?

Answer 16

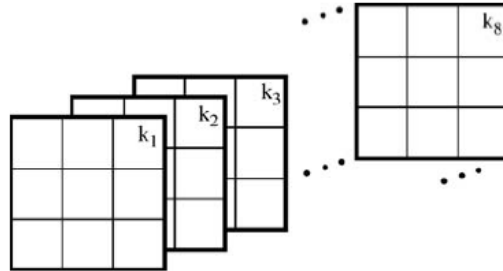


Figure 14: Kirsch masks stored in a $3 \times 3 \times 8$ matrix

Because it is 3x3x8 matrix. ■

Question 17

Why are we required to scale the image when displaying it?

Answer 17

In order to get the image where the edges are accurate proportions, scaling is needed. ■

```
40 % 26. Create a transformation function to map the image to the grayscale
41 % range and perform the transformation.
42
43 - m = 255 / (max(I_kir(:)) - min(I_kir(:)));
44 - I_kir_adj = uint8(m * I_kir);
45 - figure, imshow(I_kir_adj);
```

Code Fragment 19: Step 26 applied

Question 18

Why are we required to scale the image when displaying it?

Answer 18

Because it is already been multiplied by variable m . ■

Question 19

Make a copy of the cameraman image and add Gaussian noise to it. Then perform the Kirsch edge detector on it. Comment on its performance when noise is present.

Answer 19

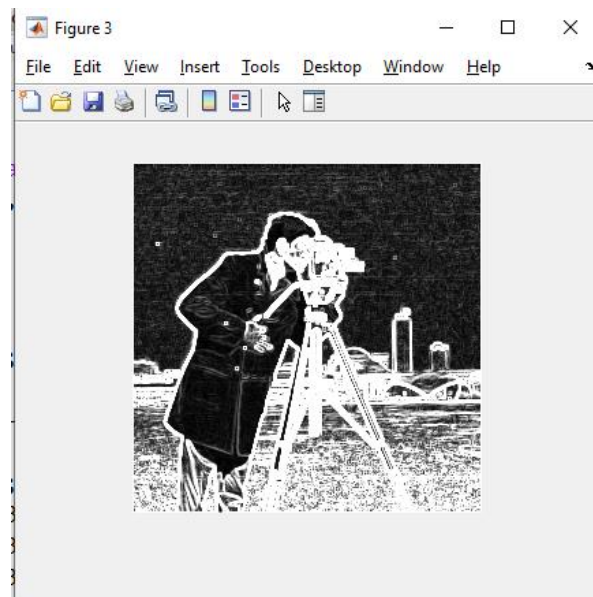


Figure 15: Question 19 image, **cameraman.tif** with **Gaussian** noise on it

With a cameraman image that a Gaussian noise applied onto it, its Kirsch edge detection is actually made a brighter image which some details can be distinguished easily. ■

Edge Detection with the Robinson Operator

```
3 % 27. Generate the Robinson masks.
4
5 - r = zeros(3,3,8);
6 - r(:,:,1) = [-1 0 1; -2 0 2; -1 0 1];
7 - r(:,:,2) = [0 1 2; -1 0 1; -2 -1 0];
8 - r(:,:,3) = [1 2 1; 0 0 0; -1 -2 -1];
9 - r(:,:,4) = [2 1 0; 1 0 -1; 0 -1 -2];
10 - r(:,:,5) = [1 0 -1; 2 0 -2; 1 0 -1];
11 - r(:,:,6) = [0 -1 -2; 1 0 -1; 2 1 0];
12 - r(:,:,7) = [-1 -2 -1; 0 0 0; 1 2 1];
13 - r(:,:,8) = [-2 -1 0; -1 0 1; 0 1 2];
14
15 % 28. Filter the image with the eight Robinson masks and display the output
16 - I_r = zeros(size(I,1), size(I,2), 8);
17 - for i = 1:8
18 -     I_r(:,:,i) = imfilter(I,r(:,:,i));
19 - end
20 - figure
21 - for j = 1:8
22 -     subplot(2,4,j), imshow(abs(I_r(:,:,j)),[]), ...
23 -     title(['Robinson mask ', num2str(j)]);
24 - end
```

Code Fragment 20: Steps 27 and 28 applied

```
26 % 29. Calculate the max of all eight images and display the result.
27 - I_rob = max(I_r,[],3);
28 - figure, imshow(I_rob,[]);
```

Code Fragment 21: Step 29 applied

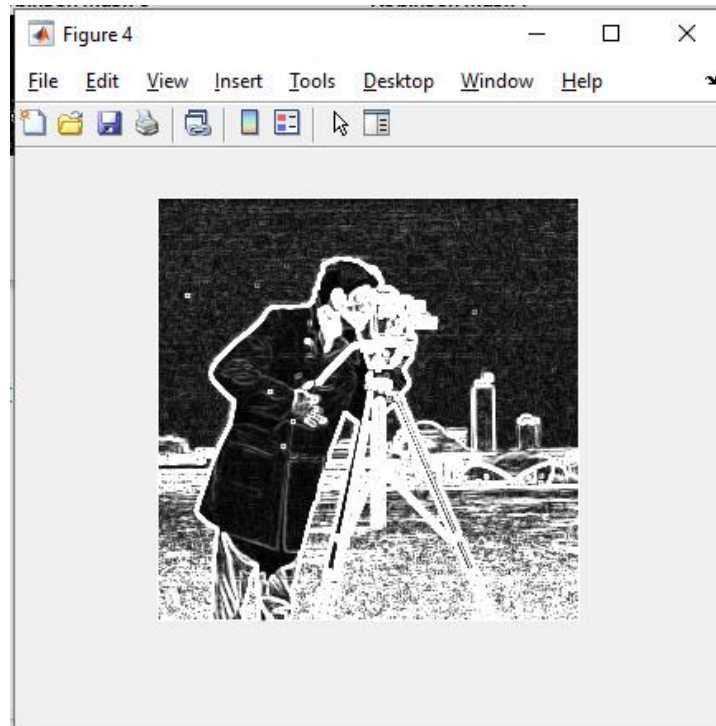


Figure 16: Result of *Robinson Edge Detection*

Question 20

How does the Robinson edge detector compare with the Kirsch detector?

Answer 20

They provide pretty much the same results – at least, from that image, the difference (if any) is not distinguishable. ■