



University of Tabuk
Faculty of Computers and Information Technology
Department of Computer Science
Second Semester 1442
CSC-606 Computer Vision/ Master of Science- Artificial Intelligence
Lab Assignment #2
Due date: 03/04/2021

Student ID: 421010012, 421010021

Student Name: Rawan AlHarbi, Areej Alhowaity

1. imread()

`A = imread(filename)` reads the image from the file specified by `filename`, inferring the format of the file from its contents.

If `filename` is a multi-image file, then `imread` reads the first image in the file.

`A = imread(filename,fmt)` additionally specifies the format of the file with the standard file extension indicated by `fmt`.

If `imread` cannot find a file with the name specified by `filename`, it looks for a file named `filename.fmt`.

`A = imread(___,idx)` reads the specified image or images from a multi-image file. This syntax applies only to GIF, CUR, ICO, TIF, and HDF4 files. You must specify a `filename` input, and you can optionally specify `fmt`.

2. figure()

`figure` creates a new figure window using default property values. The resulting figure is the [current figure](#).

`figure(Name,Value)` modifies properties of the figure using one or more name-value pair arguments. For example, `figure('Color','white')` sets the background color to white.

`f = figure(___)` returns the Figure object. Use `f` to query or modify properties of the figure after it is created.

`figure(f)` makes the figure specified by `f` the current figure and displays it on top of all other figures.

`figure(n)` finds a figure in which the `Number` property is equal to `n`, and makes it the current figure.

3. imshow()

`imshow(I)` displays the grayscale image `I` in a figure. `imshow` uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display.

`imshow(I,[low high])` displays the grayscale image `I`, specifying the display range as a two-element vector, `[low high]`. For more information, see the [DisplayRange](#) parameter.

`imshow(I,[])` displays the grayscale image `I`, scaling the display based on the range of pixel values in `I`. `imshow` uses `[min(I(:)) max(I(:))]` as the display range.

`imshow` displays the minimum value in `I` as black and the maximum value as white.

4. title()

`title(txt)` adds the specified title to the axes or chart returned by the `gca` command.

Reissuing the title command causes the new title to replace the old title.

[example](#)

`title(target,txt)` adds the title to the axes, legend, or chart specified by `target`.

[example](#)

`title(___,Name,Value)` modifies the title appearance using one or more name-value pair arguments. For

example, `'FontSize',12` sets the font size to 12 points. Specify name-value pair arguments after all other input arguments.

Modifying the title appearance is not supported for all types of charts.

[example](#)

`t = title(___)` returns the object used for the title. Use `t` to make future modifications to the title.

5. imnoise()

`J = imnoise(I,'gaussian')` adds zero-mean, Gaussian white noise with variance of 0.01 to grayscale image I.

`J = imnoise(I,'gaussian',m)` adds Gaussian white noise with mean m and variance of 0.01.

`J = imnoise(I,'gaussian',m,var_gauss)` adds Gaussian white noise with mean m and variance var_gauss.

`J = imnoise(I,'localvar',var_local)` adds zero-mean, Gaussian white noise of local variance var_local.

`J = imnoise(I,'localvar',intensity_map,var_local)` adds zero-mean, Gaussian white noise.

`J = imnoise(I,'poisson')` generates Poisson noise from the data instead of adding artificial noise to the data.

`J = imnoise(I,'salt & pepper')` adds salt and pepper noise, with default noise density 0.05.

6. edge()

`E = edge(obj,X,Y)` returns the classification edge for obj with data X and classification Y.

`E = edge(obj,X,Y,Name,Value)` computes the edge with additional options specified by one or more Name,Value pair arguments.

7. filter2()

`Y = filter2(H,X)` applies a finite impulse response filter to a matrix of data X according to coefficients in a matrix H.

8. conv2()

`C = conv2(A,B)` returns the two-dimensional convolution of matrices A and B.

9. sqrt()

`B = sqrt(X)` returns the square root of each element of the array X. For the elements of X that are negative or complex, sqrt(X) produces complex results.

The sqrt function's domain includes negative and complex numbers, which can lead to unexpected results if used unintentionally.

10.imhist()

`[counts,binLocations] = imhist(I)` calculates the histogram for the grayscale image I.

`[counts,binLocations] = imhist(I,n)` specifies the number of bins, n, used to calculate the histogram.

`[counts,binLocations] = imhist(X,map)` calculates the histogram for the indexed image X with colormap map.

`imhist(__)` displays a plot of the histogram. If the input image is an indexed image, then the histogram shows the distribution of pixel values above a colorbar of the colormap map.

11.plot()

`plot(X,Y)` creates a 2-D line plot of the data in Y versus the corresponding values in X.

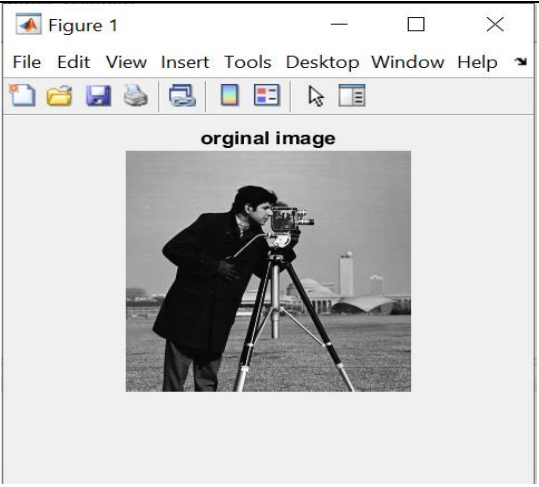
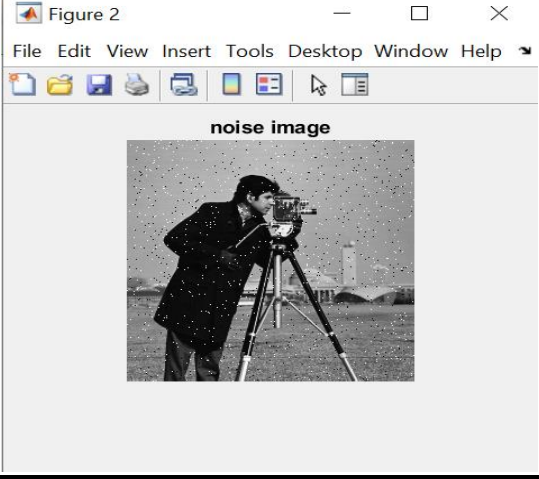
If X and Y are both vectors, then they must have equal length. The plot function plots Y versus X.

```
img=imread('cameraman.tif');
figure(1);
imshow(img);
title('original image');
% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);
imshow(nois);
title('noise image');
% used sobel edge detection
BW1 = edge(nois,'sobel');
figure(3);
imshow(BW1)
title('Sobel Filter');
%%filter noise
ff = filter2(fspecial('average',3),nois)/255;
figure(4);
imshow(ff)
title('filter noisy image');
%edge detection on filter image
e=edge(ff,'sobel')
figure(5);
imshow(e);
title('edge detection on filter image');
% gradient vector
gx=[1 0 -1;2 0 -2;1 0 -1];
Gx=conv2(img,gx);
figure(6);
imshow(Gx);
title('gradient horizontal(X)');
Fy=[1 2 1;0 0 0;-1 -2 -1];
Gy=conv2(img,Fy);
figure(7);
imshow(Gy);
title('gradient vector (Y)');
%Compute gradient magnitude at each pixel
Gm=sqrt(Gx.^2+Gy.^2);
figure(8);
imshow(Gm);
title('gradient magnitude');
%Thresholding and linking
hist=imhist(img)
figure(9);
plot(hist)
IS=Gm>78
figure(10)
imshow(IS)
title('Thresholding and linking');
```

- 1- First, we choose image form MATLAB and read it and add a tittle.
- 2- Second, we apply noise to image and we see the image apply successfully.
- 3- Then we apply edge detection algorithm for the noisy image and show that the image is convert to approximately black and white and the noise is still there.
- 4- Then we use filter to that image and apply edge detection on filter image the noise is disappear successfully.
- 5- Them we Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters we see that the image is change to white and still need more work.
- 6- Then compute gradient vector of magnitude at each pixle we see that the image almost white with a little detail.
- 7- The last Thing we apply thresholding and linking to image we see the image is now is best and all details of image is clear to see and we present a graphical interface.

Assignment #2 - edge detection using gradients

The purpose of this project is to study the effect of noise on edge detection using gradients.

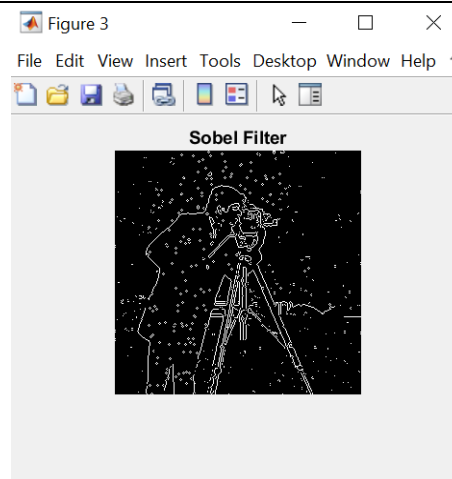
<u>Question</u>	<u>Code</u>	<u>Output</u>
2.1 Use one of the well-known images available in the MATLAB or online and any one you choose.	<pre>img=imread('cameraman.tif'); figure(1); imshow(img); title('original image');</pre>	 A screenshot of a MATLAB window titled 'Figure 1'. The window has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Tools', 'Desktop', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for file operations and viewing. The main area displays a grayscale image of a person operating a camera on a tripod, titled 'original image'.
2.2 Add to read image some noise using of your choice.	<pre>img=imread('cameraman.tif'); figure(1); imshow(img); title('original image'); % Add noise to image nois=imnoise(img,'salt & pepper',0.02); figure(2) imshow(nois) title('noise image')</pre>	 A screenshot of a MATLAB window titled 'Figure 2'. The window has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Tools', 'Desktop', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for file operations and viewing. The main area displays a grayscale image of the same person operating a camera on a tripod, but with significant salt and pepper noise added, titled 'noise image'.

2.3 Then apply the following edge detection algorithm for the noisy image and show its result

```
img=imread('cameraman.tif');
figure(1);imshow(img);title('original image');

% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);imshow(nois);title('noisy image');

% used sobel edge detection
BW1 = edge(nois,'sobel');
figure(3);imshow(BW1)
title('Sobel Filter');
```



2.4 Then try to filter the noisy image by one of the well-known filters then do the same experiment and compare their results

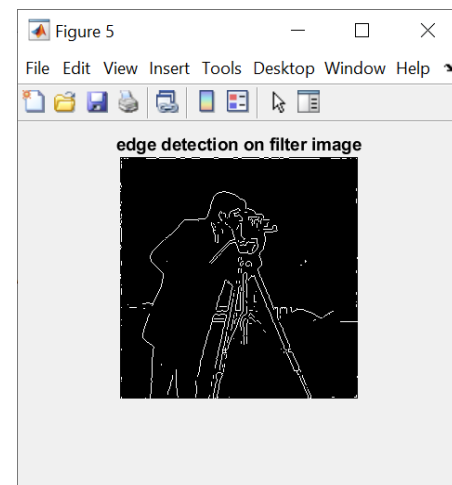
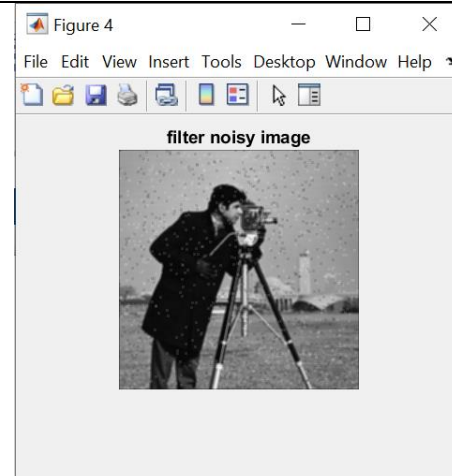
```
img=imread('cameraman.tif');
figure(1);imshow(img);title('original image');

% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);imshow(nois);title('noisy image');

% used sobel edge detection
BW1 = edge(nois,'sobel');
figure(3);imshow(BW1)
title('Sobel Filter');
```

```
% %filter noise
ff =
filter2(fspecial('average',3),nois)/255;
figure(4);
imshow(ff)
title('filter noisy image');
```

```
%edge detection on filter image
e=edge(ff,'sobel')
figure(5);
imshow(e);
title('edge detection on filter image');
```



The compare between last image and this image is that the noise that was in in the last image has successfully disappear! 😊

Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

```
img=imread('cameraman.tif');
figure(1);
imshow(img);
title('orginal image');

% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);
imshow(nois);
title('noise image');

% used sobel edge detaction
BW1 = edge(nois,'sobel');
figure(3);
imshow(BW1)
title('Sobel Filter');

%%filter noise
ff = filter2(fspecial('average',3),nois)/255;
figure(4);
imshow(ff)
title('filter noisy image');

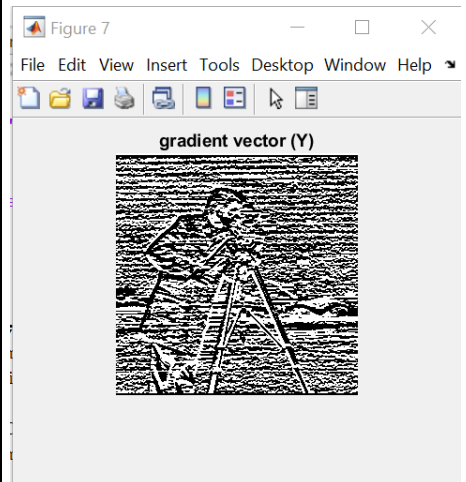
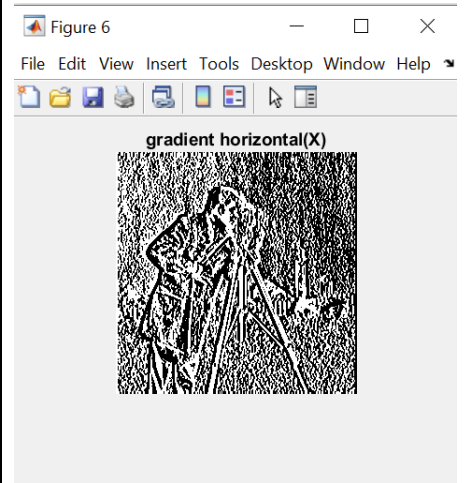
%edge detection on filter image

e=edge(ff,'sobel')
figure(5);
imshow(e);
title('edge detection on filter image');

% gradient vector

gx=[1 0 -1;2 0 -2;1 0 -1];
Gx=conv2(img,gx);
figure(6);
imshow(Gx);
title('gradient horizontal(X)');

Fy=[1 2 1;0 0 0;-1 -2 -1];
Gy=conv2(img,Fy);
figure(7);
imshow(Gy);
title('gradient vector (Y)');
```



Compute gradient magnitude at each pixel

```
img=imread('cameraman.tif');
figure(1);
imshow(img);
title('orginal image');

% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);
imshow(nois);
title('noise image');

% used sobel edge detection
BW1 = edge(nois,'sobel');
figure(3);
imshow(BW1)
title('Sobel Filter');

%% filter noise
ff = filter2(fspecial('average',3),nois)/255;
figure(4);
imshow(ff)
title('filter noisy image');

%edge detection on filter image

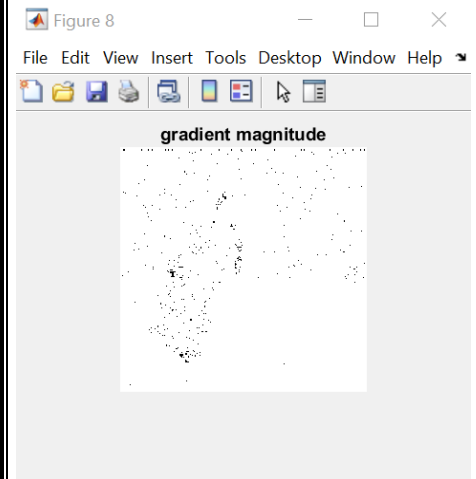
e=edge(ff,'sobel')
figure(5);
imshow(e);
title('edge detection on filter image');

% gradient vector

gx=[1 0 -1;2 0 -2;1 0 -1];
Gx=conv2(img,gx);
figure(6);
imshow(Gx);
title('gradient horizontal(X)');

Fy=[1 2 1;0 0 0;-1 -2 -1];
Gy=conv2(img,Fy);
figure(7);
imshow(Gy);
title('gradient vector (Y)');

%Compute gradient magnitude at each pixel
Gm=sqrt(Gx.^2+Gy.^2);
figure(8);
imshow(Gm);
title('gradient magnitude');
```



Thresholding and linking

```
img=imread('cameraman.tif');
figure(1);
imshow(img);
title('original image');

% Add noise to image
nois=imnoise(img,'salt & pepper',0.02);
figure(2);
imshow(nois);
title('noise image');

% used sobel edge detection
BW1 = edge(nois,'sobel');
figure(3);
imshow(BW1)
title('Sobel Filter');

% %filter noise
ff = filter2(fspecial('average',3),nois)/255;
figure(4);
imshow(ff)
title('filter noisy image');

%edge detection on filter image
e=edge(ff,'sobel')
figure(5);
imshow(e);
title('edge detection on filter image');

% gradient vector
gx=[1 0 -1;2 0 -2;1 0 -1];
Gx=conv2(img,gx);
figure(6);
imshow(Gx);
title('gradient horizontal(X)');

Fy=[1 2 1;0 0 0;-1 -2 -1];
Gy=conv2(img,Fy);
figure(7);
imshow(Gy);
title('gradient vector (Y)');

%Compute gradient magnitude at each
pixel
Gm=sqrt(Gx.^2+Gy.^2);
figure(8);
imshow(Gm);
title('gradient magnitude');

% Thresholding and linking
hist=imhist(img)
figure(9);
plot(hist)

IS=Gm>78
figure(10)
imshow(IS)
title('Thresholding and linking');
```

