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 #3

Student ID: 421010012, 421010021
Student Name: Rawan AlHarbi, Areej Alhowaity

Page 2. Abstract (not to exceed ½ page).

Several algorithms and techniques for images have been developed over the years using domain-specific knowledge to effectively solve especially in digital images when the various types of noise happen. So, Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. In addition, they have many ways that the noise can be introduced into an image, depending on how the image is created.

For example, in my code: The imnoise function:

Can use to add various types of noise to an image and simulate the effects of some of the problems listed. The next part of my code is Image segmentation:

It effectively solves segmentation problems in that specific application area. These applications include medical imaging, automated driving, video surveillance, and machine vision.

Page 3. Technical discussion.
One to two pages (max). This section should include the functions used to create different kinds of matrices and the tools to create, and modify graphic displays

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

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.
6. fspecial()	h = fspecial(type) creates a two-dimensional filter h of the specified type. fspecial('average',hsize) returns an averaging filter h of size hsize. h = fspecial('disk',radius) returns a circular averaging filter (pillbox) within the square matrix of size 2*radius+1.
7. imfilter()	B = imfilter(A,h) filters the multidimensional array A with the multidimensional filter h and returns the result in B. B = imfilter(A,h,options,...) performs multidimensional filtering according to one or more specified options.
8.size()	sz = size(A) returns a row vector whose elements contain the length of the corresponding dimension of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4]. szdim = size(A,dim) returns the length of dimension dim.
9.int32()	Y = int32(X) converts the values in X to type int32
10. plot()	plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.
11. uint8()	Y = uint8(X) converts the values in X to type uint8. Values outside the range [0,2^8-1] map to the nearest endpoint.
12. graythresh	T = graythresh(I) computes a global threshold T from grayscale image I, using Otsu's method [1]. Otsu's method chooses a threshold that minimizes the intraclass variance of the thresholded black and white pixels. The global threshold T can be used with imbinarize to convert a grayscale image to a binary image.
13. imbinarize	BW = imbinarize(I) creates a binary image from 2-D or 3-D grayscale image I by replacing all values above a globally determined threshold with 1s and setting all other values to 0s.

Page 4. Discussion of results. One to two pages (max). A discussion of results should include major findings in terms of the LAB objectives.

Part1: Image filtering:

At beginning, using **Imread** instruction to read the image. Then adding to it noise type:

Salt and pepper by using **Imnoise** instruction.

In an attempt to remove noise using average filter size 3*3 using **fspecial** instruction to generate the filter, then to filter image use two methods:

First method:

Using the direct instruction **Imfilter** which return the filtered image has the same size of original image.

Second Method:

Using set of FOR loops on the x-axis and y-axis where center of average pass on each pixel in the image, then calculate sum of 8 neighbors and center then the result divide by 9 and replace the pixel value with the divide result.

The final filter image in two methods is same where found that the noise less but the blur increased.

Part1: Image Segmentation:

Our main goal was separate the subject from the background. For that using image segmentation.

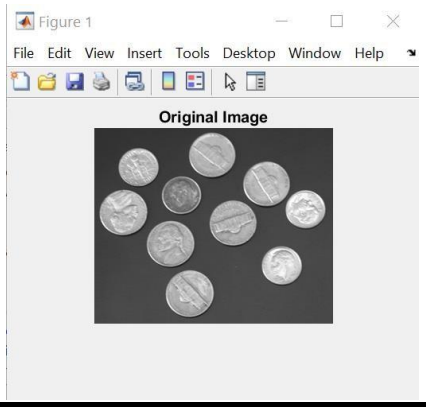
At beginning, using **graythreash** instruction to detect cutting threshold. Then using **Imbinarize** instruction which compare each image pixel if was bigger than cutting threshold replace it by one else replace it by 0.

The result after the previous instruction Image contains a goal represent by ones and background represent by zeros.

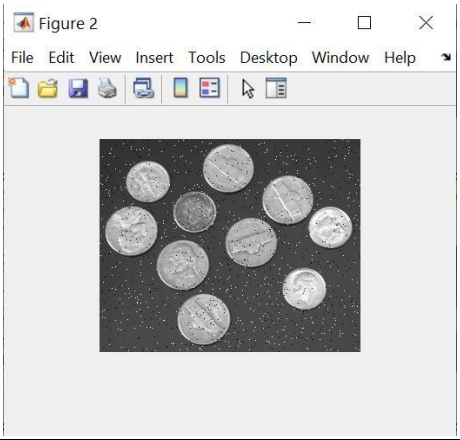
Finally, on my opinion there are many methods to filter image like mean filter that has positive point if compare it with average filter and that is Blur than.

Part 1: Image filtering

Question	Code	Output

1.Read and display the image'coins.png'.	%%Part1: Image filtering:
	%%------------------------
	%%Read Image:
	OrignImage = imread('coins.png');
	%%Display the image:
	imshow(OrignImage);figure(1);
	title('Original Image');
		

2. Add a noise ' salt & pepper' of D=0.02 density to the original image.	%%Part1: Image filtering:
	%%------------------------
	%%Read Image:
	OrignImage = imread('coins.png');
	%%Display the image:
	imshow(OrignImage);figure(1);
	title('Original Image');
	%%Add a noise 'salt & pepper':
	NoiseImage = imnoise(OrignImage,'salt & pepper',0.02);	

3. Display the disturbed image.	%%------------------------
	%%Read Image:
	OrignImage = imread('coins.png');
	%%Display the image:
	imshow(OrignImage);figure(1);
	title('Original Image');
	%%Add a noise 'salt & pepper':
	NoiseImage = imnoise(OrignImage,'salt & pepper',0.02);
	%Display the disturbed image:
	figure(2),imshow(NoiseImage);
		

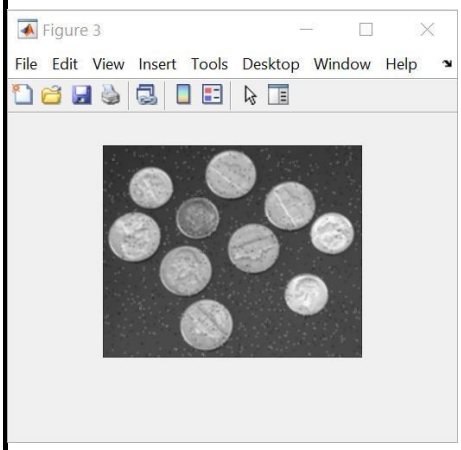
4. Create an average filter of size (3x3).

```
%%Part1: Image filtering:
%%-------------------------
%%Read Image:
OrignImage = imread('coins.png');
%%Display the image:
imshow(OrignImage);figure(1);
title('Original Image');
%%Add a noise 'salt & pepper':
NoiseImage = imnoise(OrignImage,'salt &
pepper',0.02);
%Display the disturbed image:
figure(2),imshow(NoiseImage);
%Create a average filter of size (3x3):
AverageFilter = fspecial('average');
```

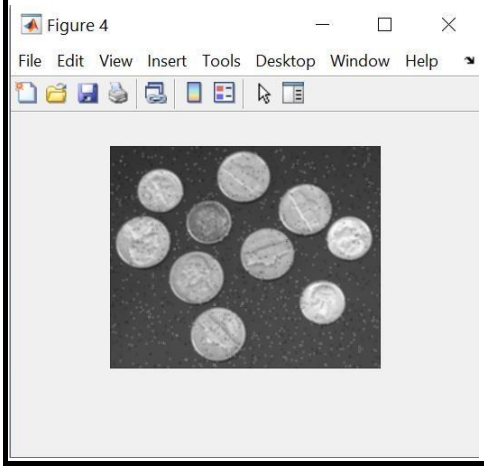
5. Apply this filter to the disturbed image (By using the 2 programmation techniques: FONCTION AND LOOP).

```
%%Part1: Image filtering:
%%-------------------------
%%Read Image:
OrignImage = imread('coins.png');
%%Display the image:
imshow(OrignImage);figure(1);
title('Original Image');
%%Add a noise 'salt & pepper':
NoiseImage = imnoise(OrignImage,'salt &
pepper',0.02);
%Display the disturbed image:
figure(2),imshow(NoiseImage);
%Create a average filter of size (3x3):
AverageFilter = fspecial('average');
%%Filter the image:
%%1-Using function:
Image_Filter=imfilter(NoiseImage,AverageFilter);
figure(3),imshow(Image_Filter);
%%2-Using loop:
[x,y] = size(NoiseImage);
for i = 2:x-1
    for j = 2:y-1
        sum = 0;
        sum=int32(sum);
        for ii = i-1:i+1
            for jj = j-1:j+1
                tmp=NoiseImage(ii,jj);
                tmp=int32(tmp);
                sum = sum + tmp;
                sum=int32(sum);
            end
        end
        Image_Filter(i,j) = ceil(sum/9);
    end
end
%%Display the image:
figure(4),imshow(uint8(Image_Filter));
```

Using function:



Using loop:



Part 2: Image segmentation

1. Determine an appropriate threshold to segment the image into two classes.

```
%%Part1: Image filtering:
%%------------------------
%%Read Image:
OrignImage = imread('coins.png');
%%Display the image:
imshow(OrignImage);figure(1);
title('Original Image');
%%Add a noise 'salt & pepper':
NoiseImage = imnoise(OrignImage,'salt &
pepper',0.02);
%Display the disturbed image:
figure(2),imshow(NoiseImage);
%Create a average filter of size (3x3):
AverageFilter = fspecial('average');
%%Filter the image:
%%1-Using function:
Image_Filter=imfilter(NoiseImage,AverageFilter);
figure(3),imshow(Image_Filter);
%%2-Using loop:
[x,y] = size(NoiseImage);
for i = 2:x-1
    for j = 2:y-1
        sum = 0;
        sum=int32(sum);
        for ii = i-1:i+1
            for jj = j-1:j+1
                tmp=NoiseImage(ii,jj);
                tmp=int32(tmp);
                sum = sum + tmp;
                sum=int32(sum);
            end
        end
        Image_Filter(i,j) = ceil(sum/9);
    end
end
%%Display the image:
figure(4),imshow(uint8(Image_Filter));

%%-----------------------------------------------
-------
%%Part2: Image segmentation:
%%---------------------------
%%Read Image:
OrignImage = imread('coins.png');
%%Select level to segmentation:
level = graythresh(OrignImage);
%%Convert the image to white goal and black back
ground:
BW=imbinarize(OrignImage,level);
%%Display the binary image:
figure(5),imshow(BW);
```