Data Compression Lab

2020-2021

المرحلة الرابعة/ الفصل الثاني

أستاذ المادة

م.م.هديل جبار

Lecture 1: Introuduction to Matlab Language

M-files

There are two types of M-files: script files and function files.

1-Script files: are called M-files because the extension on the file is ".m". Script files are a collection of Matlab commands that when executed are the same as if the commands had been entered from the keyboard. Variables that are used in script files are global variables. A variable that originates in a script file is immediately known in the command window. Variables that originated in the command window are known in the script file.

Creating Script File

To create scripts files, you need to use a text editor. You can open the MATLAB editor in two ways:-

- Using the command prompt
- Using the IDE (Integrated development environment)

If you are using the command prompt, type **edit** in the command prompt. This will open the editor. You can directly type **edit** and then the filename (with .m extension)

```
edit

Or

edit <filename>
```

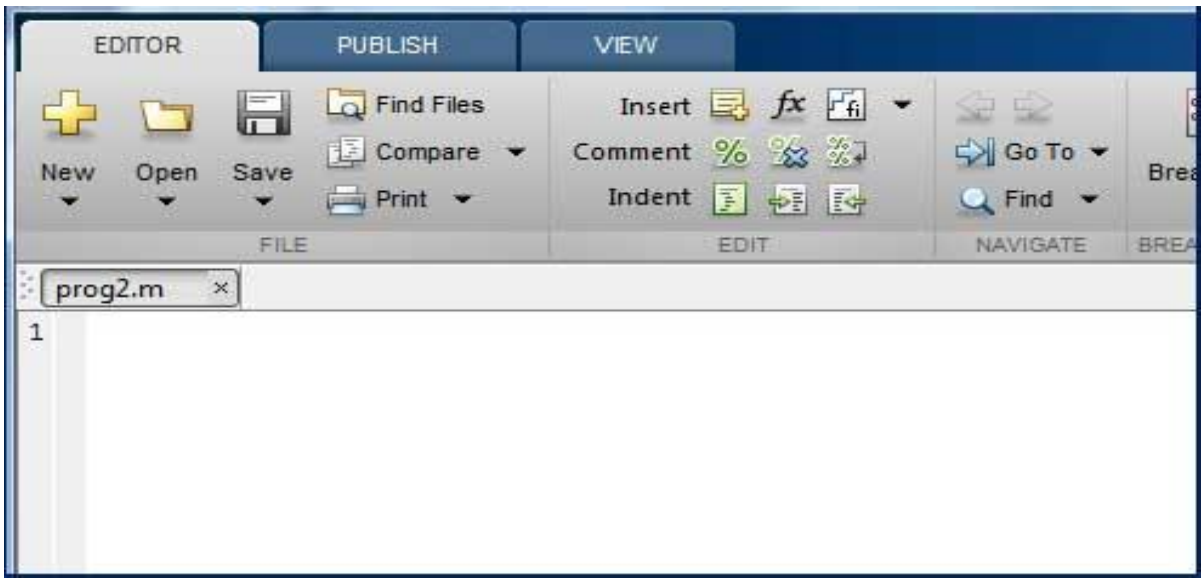
The above command will create the file in default MATLAB directory. If you want to store all program files in a specific folder, then you will have to provide the entire path.

Let us create a folder named progs. Type the following commands at the command prompt (>>) −

```
mkdir progs    % create directory progs under default directory
```

```
chdir progs    % changing the current directory to progs
edit  prog1.m  % creating an m file named prog1.m
```

If you are creating the file for first time, MATLAB prompts you to confirm it. Click Yes.



Alternatively, if you are using the IDE, choose NEW -> Script. This also opens the editor and creates a file named Untitled. You can name and save the file after typing the code.

Example

Create a script file, and type the following code:

```
a = 24 ; b= 6;
c = a + b;
d = c * cosd (a);
e= sqrt (b);
f= exp(d);
```

```
a= 24
b=2
c=26
d=23.7522
e=1.4142
f= 2.0675e+10
```

1-Function files: in Matlab are very much like functions in C/C++. The difference between function files and script files is that variables in function files are local. Any variables that are used within a function file need to be specified within the function or passed in as arguments in the argument list.

- Matlab functions can return values as many as desired.
- The function file should have the same name as the function itself with the ".m" extension added to it.
- Matlab function files can be called directly from the command window, from a script file, or from another function file.

-The syntax for a MATLAB function definition is:

Function [val1, … , valn] = **myfunc** (arg1, … , argk)

Where *(val1 … valn)* are the specified returned values from the function and

(arg1 … argk) are the values sent to the function.

Example

Create a function file, named Find Max.m and type the following code in it:

```
function [ max ] = FindMax ( a , b , c )
max=a;
if (b>max)
   max=b;
end
if (c>max)
   max=c;
end
end
```

Lecture 2: Image Instructions

1-Load image from MATLAB location

```
>> f= imread ('cat.jpg');
```

يستخدم لقراءة الصور بمختلف انواعها ويتم حفظ الصورة على شكل مصفوفه من الارقام بحيث اذا كانت الصورة:-

١-ملونة من نوع RGB سوف تكون المصفوفة ثلاثية الابعاد وكل عنصر قيمته بين (0-255)

٢-ذات التدرج الرمادي Gray Scale ستكون المصفوفة ثنائية الابعاد وكل عنصر قيمته بين (0-255)

٣-اسود وابيض Black White ستكون المصفوفة ثنائية القيم وعناصرها (0,1)

2-Load image from any location on computer

```
>> f= imread ('E: \myimages\cat.jpg');
```

3-Display all images

```
>>imshow (f)
```

```
>>imshow (f), figure, imshow(x);
```

تستخدم لاظهار الصورة في نافذة منفصلة.

4-Image size

```
>> size(f)

ans =

  480   480     3

>> [m n] = size(f);

>> [m n d] = size(f);
```

تستخدم لعرض حجم الصورة حيث أن n=column, m=row

5-Basic information about the image

```
>> whos f
```

Name	Size	Bytes	Class	Attributes
F	480x480x3	691200	uint8	

6- Resize image

```
>> h = imresize(I,2);

>> h = imresize(I,0.5);
```

يستخدم imresize لتغير حجم الصورة (تكبير او تصغير)

```
clc

clear

i= imread('C:\Users\Public\Pictures\Sample  Pictures\Desert.jpg');
```

```
imshow(i);

z=imresize (i,0.5);

figure

imshow(z);
```

Image before resize

Image after resize

Lecture 3: Image Information

>> imfinfo file name

>> imfinfo ‘cat.jpg’

>>imfinfo('C:\Users\Public\Pictures\SamplePictures\ Desert.jpg')

>>imfinfo('https://upload.wikimedia.org/wikipedia/commons/thumb/5/52/Liliumbulbiferumflowertop.jpg/220px-Liliumbulbiferumflowertop.jpg')

يستخدم هذا الايعاز لاستعراض المعلومات الخاصة بالصورة وبالتفصيل وتظهر هذه المعلومات في شاشة ال command window ويجب عدم وضع semicolon (;) في نهاية العبارة لكي تظهر كافة المعلومات.

Example

```
clc
clear
i=imread('C:\Users\Public\Pictures\Sample  Pictures\Desert.jpg');
imshow(i);
info=imfinfo('C:\Users\Public\Pictures\Sample Pictures\Desert.jpg')
```

```
 ans =

Filename: 'C: \Program Files\MATLAB\MATLAB Production Server\R2015a\bin\cat.jpg'

FileModDate: '04-Mar-2018 20:07:55'

FileSize: 11051

Format: 'jpg'

 FormatVersion: ''
```

Width: 480

Height: 480

BitDepth: 24

ColorType: 'truecolor'

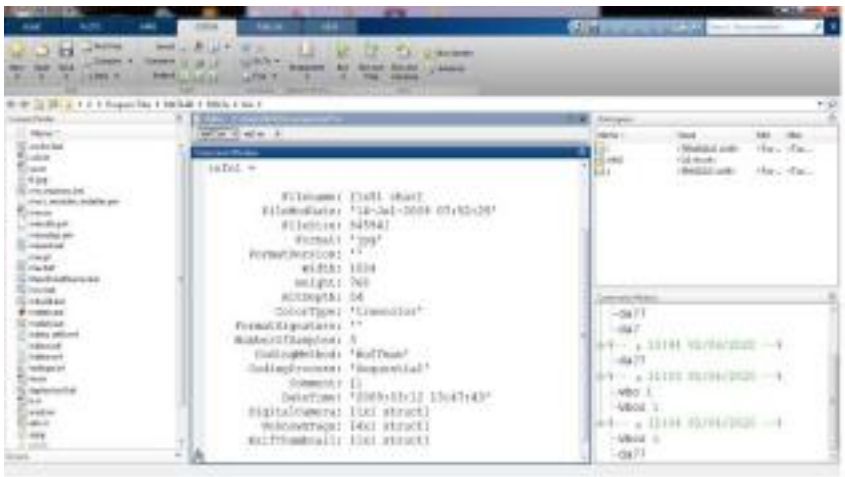
FormatSignature: ''

NumberOfSamples: 3

CodingMethod: 'Huffman'

CodingProcess: 'Sequential'

Comment: {}



Lecture 4: Write Image or Save Image

>> imwrite (f, file name)

>> imwrite (f, 'C:\Users\ Desktop\New folder\cat1 .jpg')

imwrite لحفظ او طباعة الصورة باسم وامتداد معين.

*Save image in different quality (q between 0 and 100)

>> imwrite (f, file name, 'quality', q)

>> imwrite (r2, 'C:\Users\Desktop\Newfolder\1.jpg','quality',12);

Example

```
clc
clear
i= imread('C:\Users\Public\Pictures\Sample pictures\Desert.jpg');
imshow(i)
imwrite(i,'a.jpg','quality',40);
z=imread('a.jpg')
figure
imshow(z)
```

اظهار الصورتين في نفس النافذه:-

```
i= imread('C:\Users\Public\Pictures\Sample
Pictures\Desert.jpg');
imshow(i);
```

```
imwrite(i,'a.JPG','quality',40);
z=imread('a.JPG')
subplot(1,2,1);
imshow(i);
title('picture I');
subplot(1,2,2)
imshow(z);
title('picture Z')
```

نلاحظ هنالك تغيير في الصورة اما اذا كانت قيمة الquality تساوي 90

Lecture 5: Data Classes

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

-The first eight entries in the table are referred to as *numeric* data classes.

-The ninth entry is the *char* class and, as shown, the last entry is referred to as the *logical* data class.

-All *numeric* computations in MATLAB are done using double quantities, so this is also a frequent data class encountered in image processing applications.

Conversion between Image Classes:

1-RGB or color map image to Gray scale image

```
I2 = rgb2gray(I);
```

2-Gray scale image to binary image, replacing all pixels in the input image with intensity greater than level with the value 1 (white) and replacing all other pixels with the value0 (black).

Level is a number between 0 and 1.

```
BW = im2bw(I,level)
>> f1=im2bw(f,0.5);
```

3-Intensity image to double precision (quality).

```
>> I2 = im2double(I);
```

I can be a grayscale intensity image, a true color image, or a binary image.

Example

Read an image in a MATLAB script, calculate the size of the image, and convert it to binary image and save it in quality 40.

```
 I= imread ('E: \myimages\cat.jpg');
[x, y]=size (I);
I2=im2bw (I, 0.6);
imwrite (I2, 'C:\Users\ Desktop\New folder\cat1.jpg','quality',40);
```

-How to split a color image into 3 RGB channels?

```
im=imread('C:\Users\Rore\Desktop\New folder\1 (60).jpg');
r = im(:,:,1);
g = im(:,:,2);
b = im(:,:,3);
% Create an all black channel.
black = zeros(size(im, 1), size(im, 2), 'uint8');
% Create color versions of the individual color channels.
just_red = cat(3, r, black, black);
just_green = cat(3, black, g, black);
just_blue = cat(3, black, black, b);
% Recombine the 3color channels to create the original RGB image
again.
rec = cat(3, r, g, b);
% Display them all.
subplot(3, 3, 2);
imshow(im)
title('Original RGB Image')
imshow(just_red);
title('Red Channel in Red')
subplot(3, 3, 5);
imshow(just_green)
title('Green Channel in Green')
subplot(3, 3, 6);
imshow(just_blue);
title('Blue Channel in Blue')
subplot(3, 3, 8);
imshow(rec);
title('Recombined to Form Original RGB Image Again')

imwrite(just_red,'C:\Users\Rore\Desktop\New folder\1r (60).jpg');
imwrite(just_green,'C:\Users\Rore\Desktop\New folder\1g (60).jpg');
imwrite(just_blue,'C:\Users\Rore\Desktop\New folder\1b (60).jpg');
```

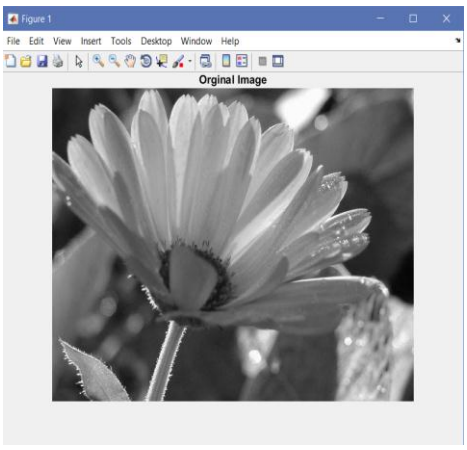
Lecture 6: Frequency Domain

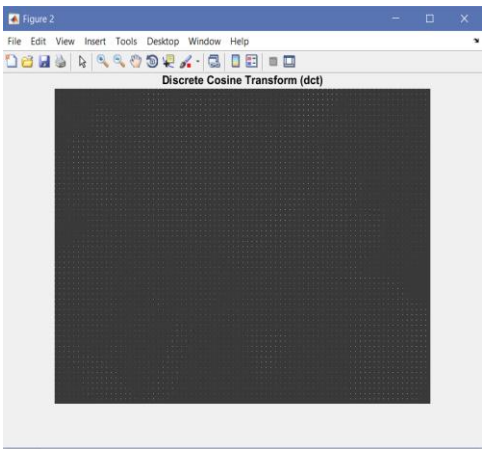
1-Discrete cosine Transform (DCT)

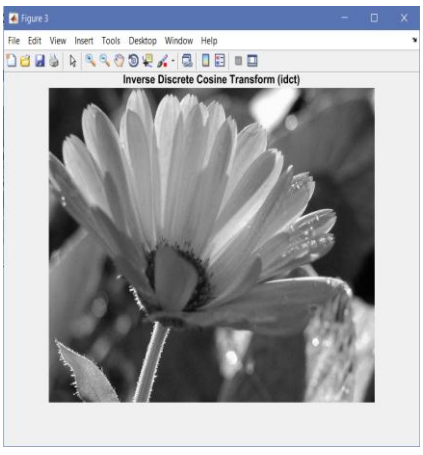
dct2 ⟹ 2-D discrete cosine transform.
idct2 ⟹ 2-D inverse discrete cosine transform.

Example

```
i=imread('image1.jpg');
gray=rgb2gray(i);
figure(1);imshow(gray);title('Orginal Image');

b=blkproc(gray,[8 8],'dct2');
figure(2);
imshow(b,[]);title('Discrete Cosine Transform (dct)');

bb=blkproc(b,[8 8],'idct2');
figure(3);
imshow(bb,[]);title('Inverse Discrete Cosine Transform (idct)');
```





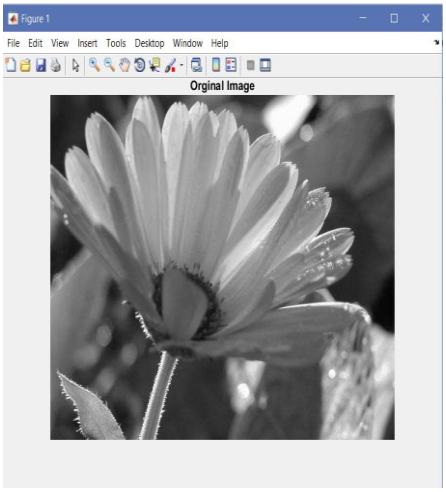


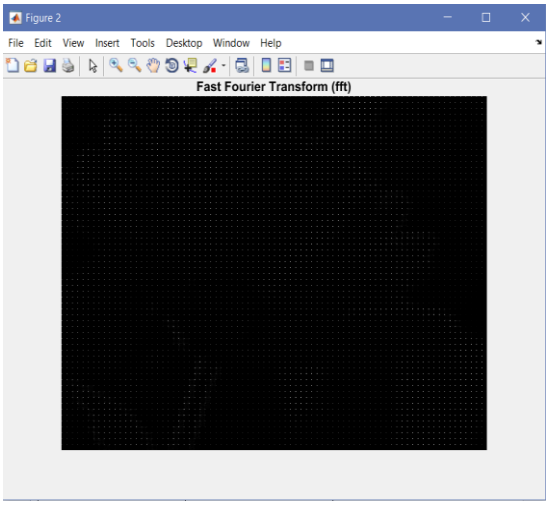
2-Fast Fourier Transform (FFT)

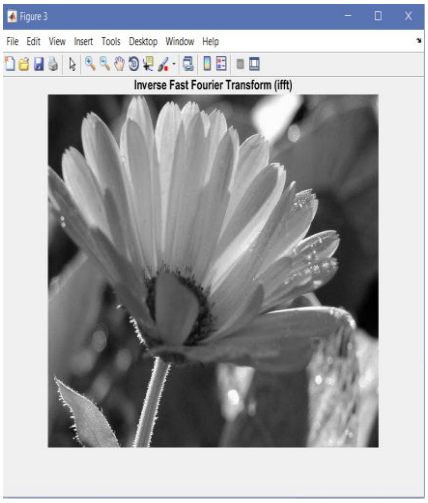
fft2 → 2-D fast fourier transform.
ifft2 → 2-D inverse fast fourier transform.

Example

```
i=imread('image1.jpg');
gray=rgb2gray(i);
figure(1);
imshow(gray);title('Orginal Image');

b=blkproc(gray,[8 8],'fft2');
figure(2);
imshow(abs(b),[]);title('Fast Fourier Transform (fft)');

bb=blkproc(b,[8 8],'ifft2');
figure(3);
imshow(abs(bb),[]);title('Inverse Fast Fourier Transform (ifft)');
```







Lecture 7: Calculate the Compression Ratio

$$\text{compression ratio} = \frac{\text{The number of bytes in the original image}}{\text{file size}}$$

$$\text{The number of bytes in the original image} = \frac{\text{width} \times \text{height} \times \text{bitdepth}}{8}$$

$$\text{The number of bytes in the original image} = \frac{225 \times 225 \times 24}{8}$$

$$\Rightarrow \quad = 151875$$

$$\text{compression ratio} = \frac{151875}{7119} \quad \Longrightarrow \quad = 21.33$$

- The information fields displayed by imfinfo can be captured into a so-called *structure variable* that can be used for subsequent computations.

```
>> k=imfinfo('image2.jpg');
>> image_bytes=k.Width*k.Height*k.BitDepth/8;
>> compressed_bytes=k.FileSize;
>> compression_ratio=image_bytes/compressed_bytes
   compression_ratio =    21.3338
```

Example

```
close all;
clc;
i=imread('image1.jpg');
gray=rgb2gray(i);
figure(1);imshow(gray);title('Orginal Image');
imwrite(gray,'C:\image compression\gray1.jpg')


b=blkproc(gray,[8 8],'dct2');
figure(2);
imshow(b,[]);title('Discrete Cosine Transform (dct)');
imwrite(b,'C:\image compression\bdct2.jpg')


x=imfinfo('gray1.jpg');
y=imfinfo('bdct2.jpg');

imbytesOrginal=(x.Width*x.Height*x.BitDepth)/8;
CRorginal=imbytesOrginal/x.FileSize;

imbytesDct=(y.Width*y.Height*y.BitDepth)/8;
CRDct=imbytesDct/y.FileSize;

CR= CRorginal /CRDct;
```

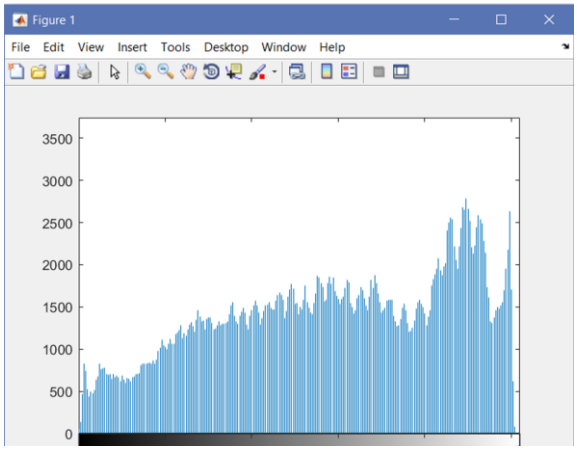
Example

```
i=imread('image1.jpg');
gray=rgb2gray(i);
figure(1);imshow(gray);title('Orginal Image');
imwrite(gray,'C:\image compression\gray1.jpg')


b=blkproc(gray,[8 8],'fft2');
figure(2);imshow(abs(b),[]);title('Fast Fourier Transform (fft)');
imwrite(b,'C:\image compression\bfft2.jpg')
```

```
x=imfinfo('gray1.jpg');
y=imfinfo('bfft2.jpg');

imbytesOrginal=(x.Width*x.Height*x.BitDepth)/8;
CRorginal=imbytesOrginal/x.FileSize;

imbytesFft=(y.Width*y.Height*y.BitDepth)/8;
CRFft=imbytesFft/y.FileSize;

CR=CRorginal/CRFft;
```

Lecture 8: Histogram

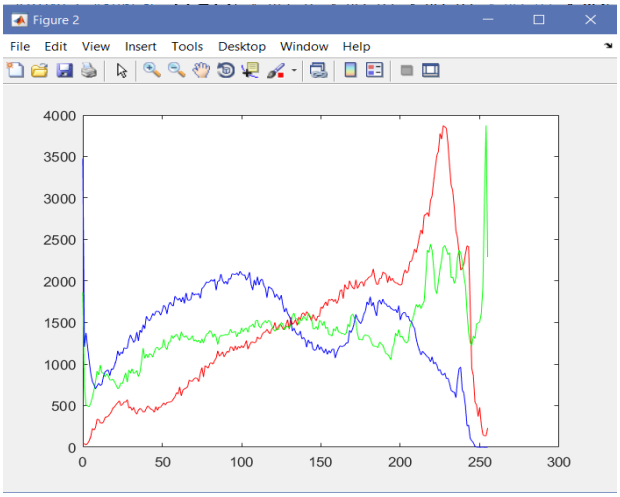
-Gray Image Histogram:




```
i=imread('cat.jpg');
gray=rgb2gray(i);
figure, imhist(gray);
```

-Color Image Histogram:

```
f=imread('cat.png');
R=f(:,:,1);
G=f(:,:,2);
B=f(:,:,3);
[YRed,x]=imhist(R);
[YGreen,x]=imhist(G);
[YBlue,x]=imhist(B);
figure(2);
plot(x,YRed,'Red',x,YGreen,'Green',x,YBlue,'Blue')
```



Measures of image quality:

1. Calculate Meant Signal to Noise Ratio (SNR) for image

```
Im=imread('cat.jpeg');
 imshow(Im)
 Im=im2double(Im);
 m=mean(Im(:));
 s=std(Im(:));
 snr=m/s
```

$$mean = \frac{sum}{count}$$

$$SNR = \frac{mean}{Std}$$

2. Calculate Mean Square Error (MSE) for image

```
im = imread('image4.png');
imshow(im)
ref = im2double(im);
A = imnoise(ref,'salt & pepper', 0.05);
%%A = imnoise(ref,'gaussian', 0.02);
%%A = imnoise(ref,'poisson');
figure(2);imshow(A)

[m ,n]=size(ref);
error=sum(ref(:)-A(:));
MSE=(error^2/(m*n))
```

$$MSE = \frac{Error^2}{m * n}$$

$$Error = \sum (I_{orginal} - I_{noise\ or\ compression})$$

3. Calculate Root Mean Square Error (RMSE) for image

```
RMSE=sqrt(MSE)
```

$$RMSE = \sqrt{MSE}$$

4. Calculate Peak Signal to Noise Ratio (PSNR) for image

```
PSNR=1/MSE
```

$$PSNR = \frac{R^2}{MSE}$$

$R = 1$ *if the input image type is double*

$R = 255$ *if the input image type is unit8*

Lecture 9: Predictive Coding (PC)

function[F,Pred]=PredCoding()

Example

```
clear all;
F1=imread('image1.jpg');
F2=rgb2gray(F1);
F=im2double(F2);
[M,N]=size(F);

  Pred(1,1:N)=F(1,1:N);
  Pred(2:M,1)=F(2:M,1);

for i=2:M
    for j=2:N
      Pred(i,j)=F(i-1,j);
    end
end

for i=1:M
    for j=1:N
      Res(i,j)=F(i,j)-Pred(i,j);
    end
end

for i=1:M
    for j=1:N
      Reconstruct(i,j)=Pred(i,j)+Res(i,j);
    end
end

 figure(1), subplot(2,2,1);imshow(F,[]);title('Original Image');
           subplot(2,2,2);imshow(Pred,[]);title('Predicted Image');
           subplot(2,2,3);imshow(Res,[]);title('Residual Image');
           subplot(2,2,4);imshow(Reconstruct,[]);title('Reconstructed
Image');
end
```

Example

Create three function files, to do the following:

1- Build program for compression image using Predictive Coding
2- Calculate the Compression Ratio.
3- Calculate MSE and PSNR.

Sol:

1. function [F,Pred]=PredCoding()

```
clear all;
F1=imread('image1.jpg');
F2=rgb2gray(F1);
F=im2double(F2);
[M,N]=size(F);

  Pred(1,1:N)=F(1,1:N);
  Pred(2:M,1)=F(2:M,1);

for i=2:M
   for j=2:N
     Pred(i,j)=F(i-1,j);
   end
end

for i=1:M
   for j=1:N
     Res(i,j)=F(i,j)-Pred(i,j);
   end
end

for i=1:M
   for j=1:N
     Reconstruct(i,j)=Pred(i,j)+Res(i,j);
   end
end

 figure(1), subplot(2,2,1);imshow(F,[]);title('Original Image');
          subplot(2,2,2);imshow(Pred,[]);title('Predicted Image');
          subplot(2,2,3);imshow(Res,[]);title('Residual Image');
          subplot(2,2,4);imshow(Reconstruct,[]);title('Reconstructed
Image');
```

```
end
```

2. function [CR] = Calculate_CR1(F,Pred)

```
imwrite(F,'C:\image compression\F1.jpg')
imwrite(Pred,'C:\image compression\Pred2.jpg')

x=imfinfo('F1.jpg');
y=imfinfo('Pred2.jpg');

OrginalBytes=(x.Width*x.Height*x.BitDepth)/8;
CRorginal=OrginalBytes/x.FileSize;

PCBytes=(y.Width*y.Height*y.BitDepth)/8;
CRPC=PCBytes/y.FileSize;

CR=CRorginal/CRPC;
end
```

3. function [MSE,PSNR] = Measures_1(F,Pred)

```
[m ,n]=size(F);
error=sum(F(:)-Pred(:));
MSE=(error^2/(m*n));
PSNR=1/MSE;
end
```

Call the functions in script file:

```
[F,Pred]=PredCoding();
[ MSE,PSNR ] = Measures_1( F,Pred );
[ CR ] = Calculate_CR1( F,Pred);
```

Lecture 10: Quantization

Example

Create two function files, to do the following:

1- Build program for compression image using Quantization.
2- Calculate MSE and PSNR.

Sol:

1.
```
function [F,ImgQuantized] = QD()


F=imread('C:\Users\Rore\Desktop\New folder\1.jpg');
F=rgb2gray(F);
figure(1), imshow(F,[]); title('Original Image');

BSLH=input('Enter quantized level :','s');
B=str2num(BSLH);
Qant. step = 256/B;
ImgQuantized(:,:)=round(F(:,:)/Qant. step);
figure(2), imshow(ImgQuantized,[]); title('Quatized Image');

ImgDeQuantized(:,:)=(ImgQuantized(:,:)*Qant. step);
figure(3), imshow(ImgDeQuantized,[]); title('Dequatized Image');
end
```

2.
```
function [MSE,PSNR] = Measures(F,ImgQuantized)
ref = im2double(F);
ImgQuantized = im2double(ImgQuantized);
[m ,n]=size(F);
error=sum(ref(:)- ImgQuantized (:));
MSE=(error^2/(m*n));
PSNR=1/MSE;
End
```

Call the functions in script file:

```
[F,ImgQuantized] = QD();
[MSE,PSNR] = Measures(F,ImgQuantized);
```

New → Function

Rm.m

```
function [ i ] = Rm()
i=imread('cat.png');
figure(1);imshow(i)
end
```

New → Function

Sm.m

```
function  Sm(i )
g=rgb2gray(i);
d=imresize(g,0.3);
figure(2);imshow(d)
end
```

New → Script

eee.m

```
[ i ] = Rm();
 Sm(i );
```