Image processing

DR :Ayman Soliman

BY : **OMAR NADER SHAMS**

MOHAMED ALAA MOHAMED

ESLAM MAHMOUD ABDELMONEIM

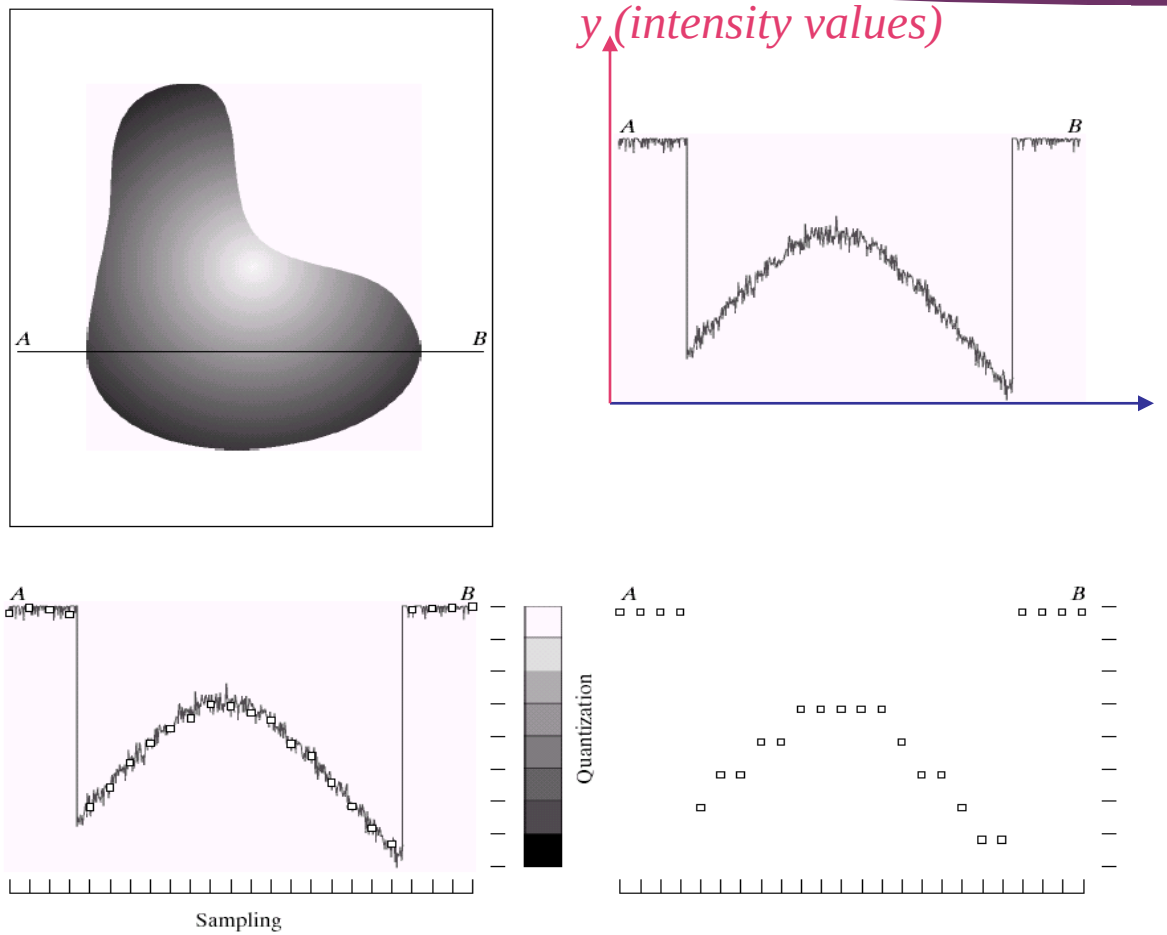
Contents

- Sampling and the Fourier transform of sampled function.
- The Discrete Fourier transform of one variable
- Extension of Functions of two variables
- Some properties of the 2-D DFT and IDFT

Types of Sampling in Digital Image Processing

- There are three types of sampling methods in Image Processing**:**
  - **Ideal Sampling**
    - In ideal sampling (Instantaneous sampling) pulses from the analog signal are sampled. This is an ideal sampling method and **cannot be easily implemented.**
  - **Natural Sampling**
    - Natural sampling is a practical method of sampling in which pulse have finite width equal to T. The result is a sequence of samples that reshape the analog signal**.**
  - **Flat Top Sampling**
    - In comparison to natural sampling, **flat top sampling can be easily obtained**. In this sampling technique, the top of the samples remains constant by using a circuit. This sampling method is commonly used.

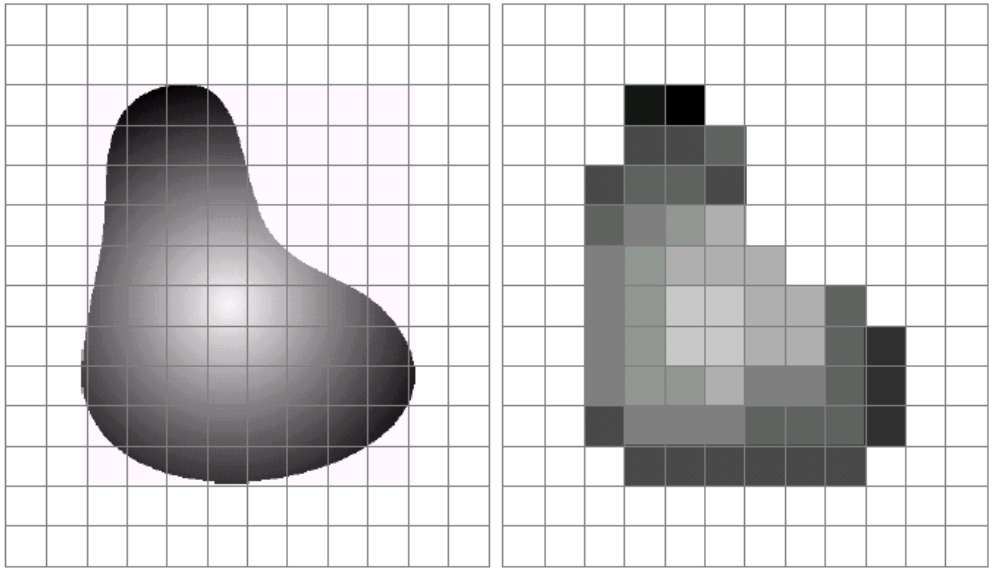
Sampling



a b
c d

- Generating a digital image. (a) Continuous image. (b) A scaling line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization. (c) sampling and quantization. (d) Digital scan line.

Sampling



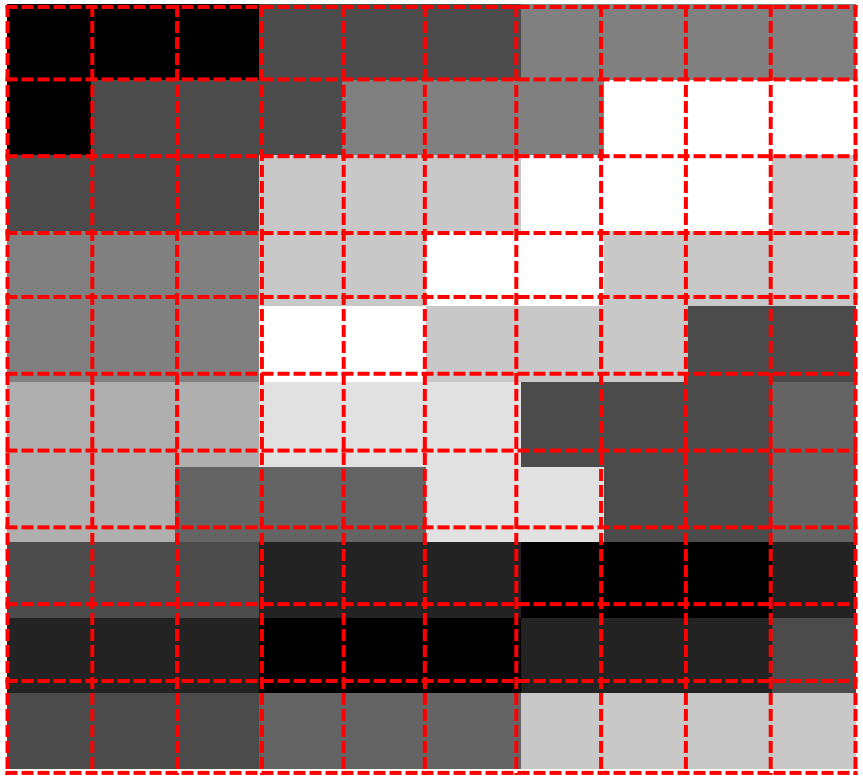
a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

a b

- (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Sampling



0	0	0	75	75	75	128	128	128	128
0	75	75	75	128	128	128	255	255	255
75	75	75	200	200	200	255	255	255	200
128	128	128	200	200	255	255	200	200	200
128	128	128	255	255	200	200	200	75	75
175	175	175	225	225	225	75	75	75	100
175	175	100	100	100	225	225	75	75	100
75	75	75	35	35	35	0	0	0	35
35	35	35	0	0	0	35	35	35	75
75	75	75	100	100	100	200	200	200	200

Sampling

1024

512

256

128

64

32

Sampling

1024

512

256

128

64

32

Sampling

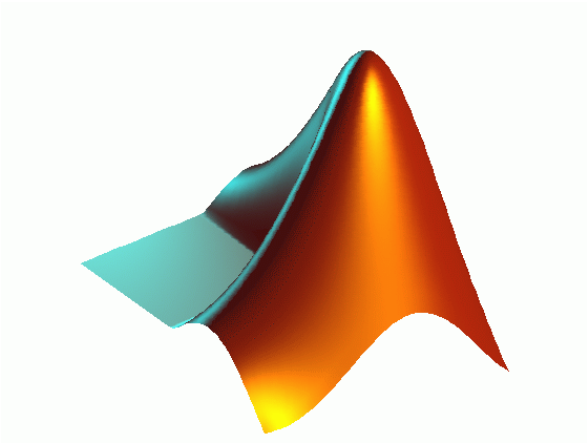
imread() – reading an image with different postfixes

imresize() – resizing an image to any given size

figure – opening a new graphical window

subplot(#of row, # of col, location) – showing different plots/images in one graphical window

imshow() – displaying an image

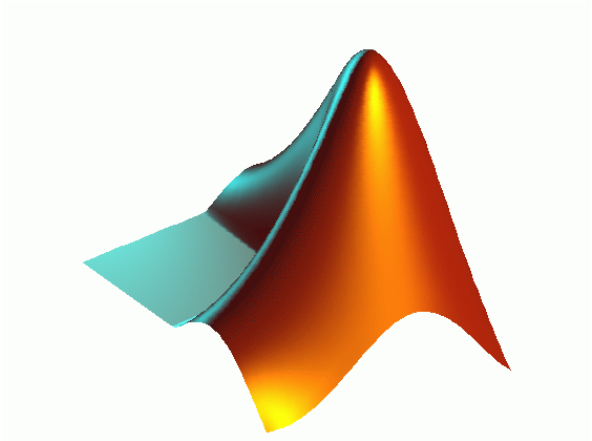


Sampling

generating figures of slide 7

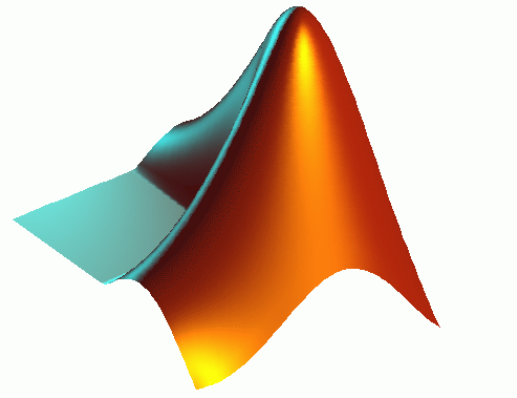
```
im=imread('obelix.jpg');
im=rgb2gray(imread('obelix.jpg'));
im1=imresize(im, [1024 1024]);
im2=imresize(im1, [1024 1024]/2);
im3=imresize(im1, [1024 1024]/4);
im4=imresize(im1, [1024 1024]/8);
im5=imresize(im1, [1024 1024]/16);
im6=imresize(im1, [1024 1024]/32);

figure;imshow(im1)
figure;imshow(im2)
figure;imshow(im3)
figure;imshow(im4)
figure;imshow(im5)
```



Sampling

generating figure of slide 8



```
figure;
subplot(2,3,1);imshow(im1);subplot(2,3,2);imshow(im2)
subplot(2,3,3);imshow(im3);subplot(2,3,4);imshow(im4)
subplot(2,3,5);imshow(im5);subplot(2,3,6);imshow(im6)
```

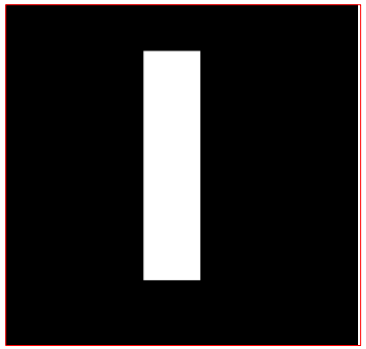
Fourier Transform – What and Why?

- **What is Fourier Transform?**
  - • A function can be described by a summation of waves with different frequency, amplitudes and phases.
- **The importance of Fourier Transform in Imaging?**
  - Signal representations in the frequency domain provide unique information.
  - Certain computations can be performed more efficiently in frequency domain.
  - Certain hardware naturally measures signals in the frequency domain.

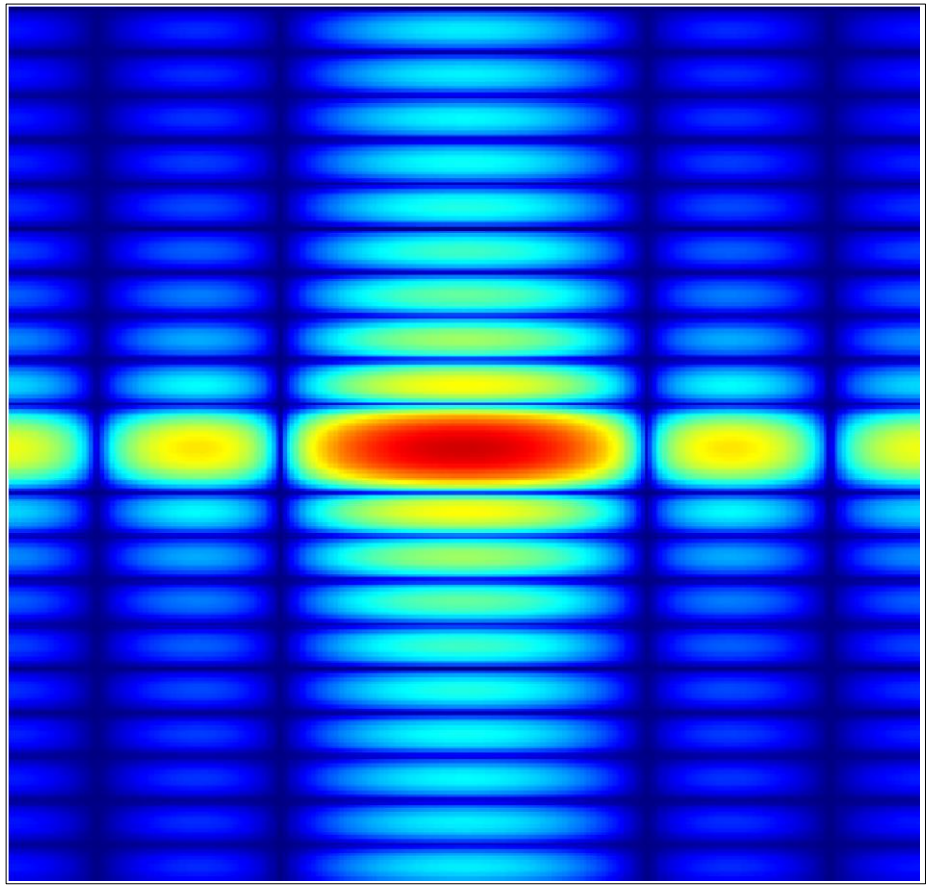
Fourier Transform

- **Why do we convert images to spectrum domain?**
- For exposing image features not visible in spatial domain, eg. periodic interferences
- For achieving more compact image representation (coding), eg. JPEG, JPEG2000
- For designing digital filters
- For fast processing of images, eg. digital filtering of images in spectrum domain

Fourier transform of images



FFT



An Example

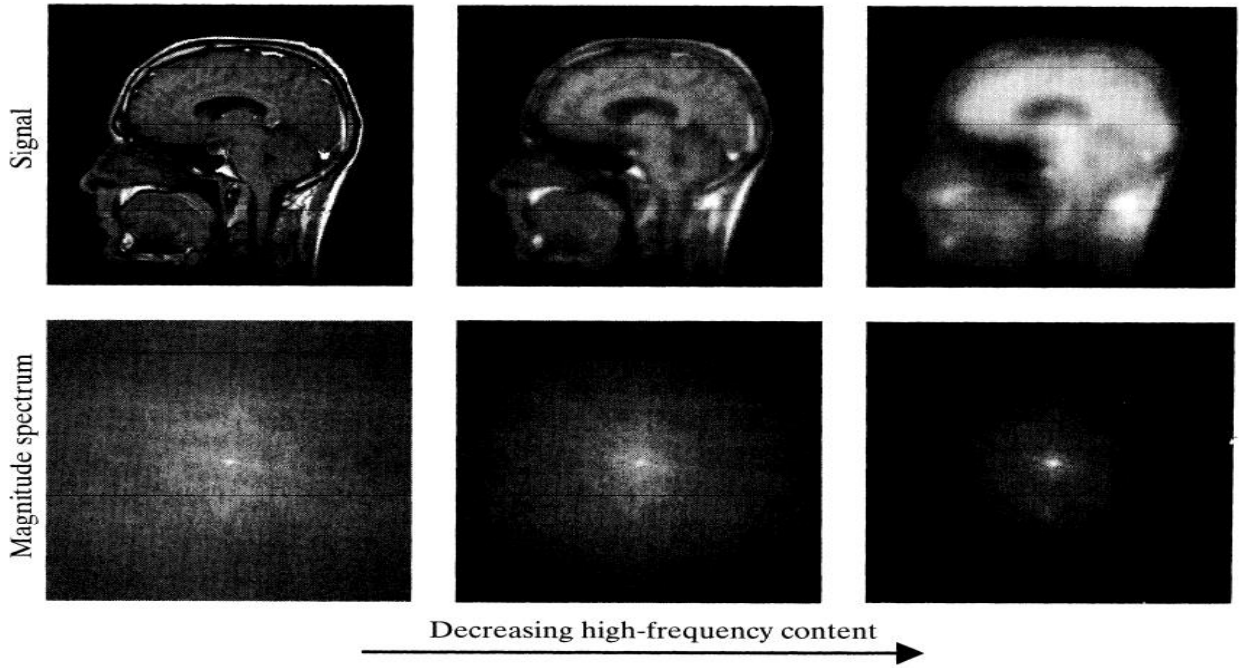
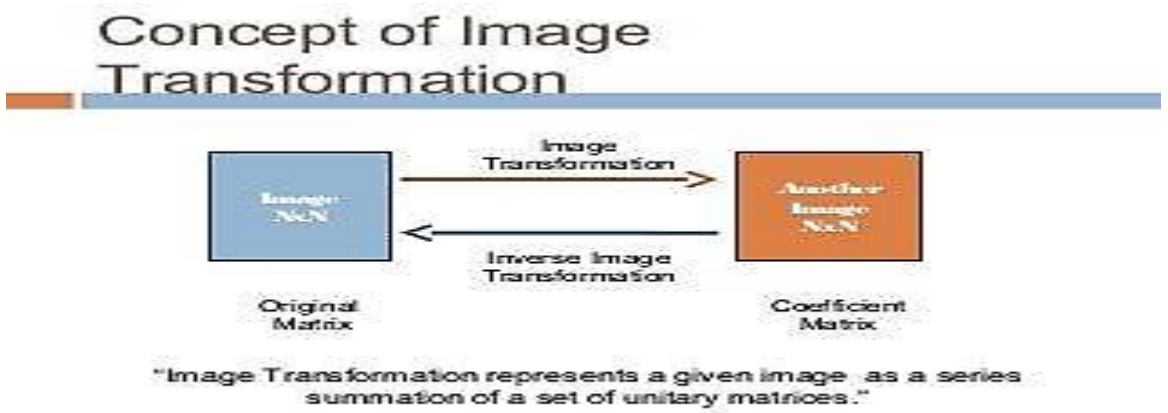

Signal
Magnitude spectrum
Decreasing high-frequency content

Image Transforms

- Image transforms are mathematical tools that help us to convert images from spatial domain to frequency domain.

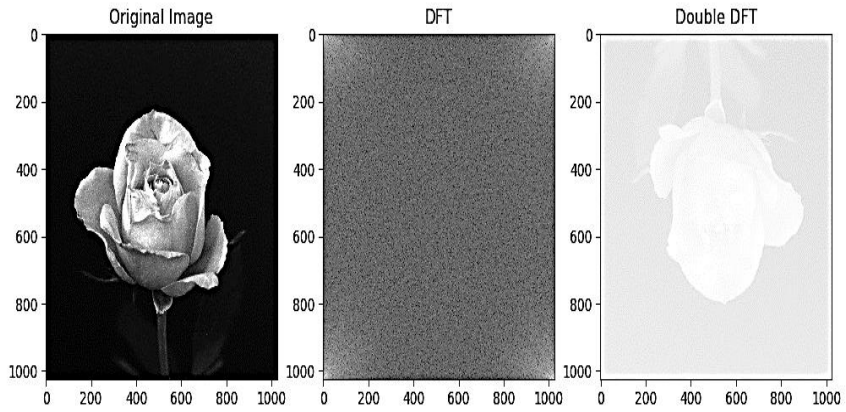


Advantages for transforming images

- It may isolate critical components of image pattern so that they are directly accessible for analysis.
- It may place image data in a more compact form so that it can be stored and transmitted efficiently.
- It is useful for fast computation of 2D convolution and correlation.
- It is reversible where we can have the original image in the initial spatial domain easily.

Discrete Fourier Transform DFT

- DFT is a type of image transforms not the only one.
- The discrete Fourier transform (DFT) is a method for converting a sequence of N complex numbers $x_0, x_1, \ldots, x_{N-1}$ to a new sequence of N complex numbers.



Discrete Fourier Transform DFT

- one-dimensional DFT
  - F(k) = $\sum_{x=0}^{N-1} f(x) e^{(-j2\pi kx/N)}$, where k=0, 1, …, N-1
- Inverse DFT
  - f(x) =$\frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{(j2\pi kx/N)}$, where x=0, 1, …, N-1

Discrete Fourier Transform DFT

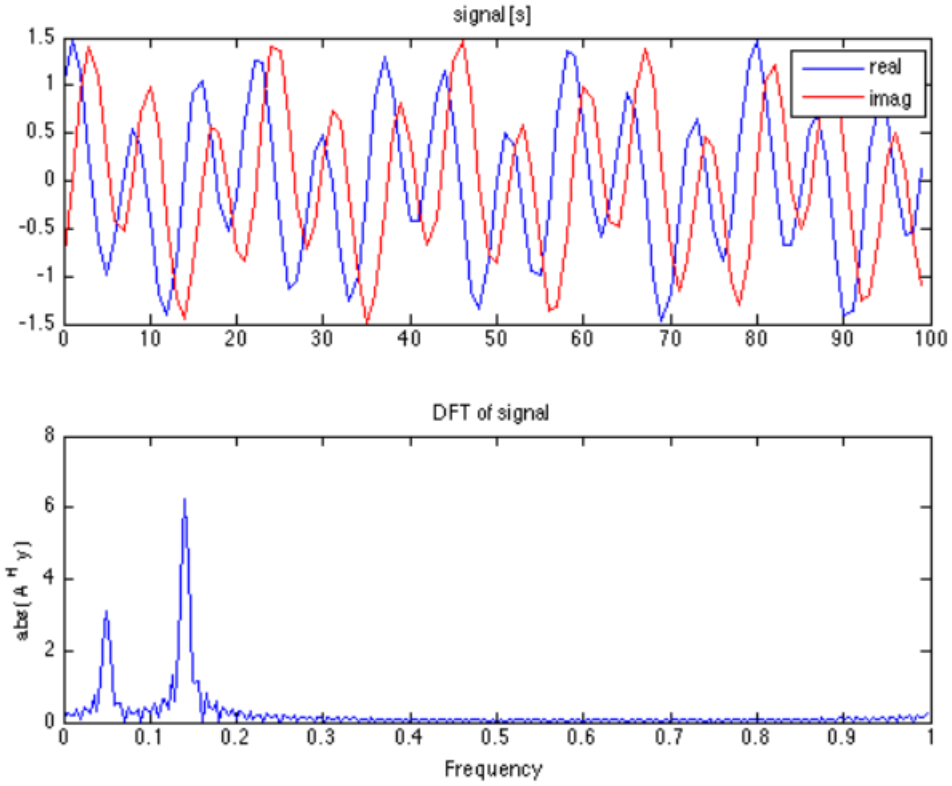
- MATLAB example

```
f = (0:Nfft-1)/Nfft;          % f : frequency axis

figure(1)
clf

subplot(2, 1, 1)
plot(n, real(y), n, imag(y), 'r')
legend('real','imag')
title('signal [s]')

subplot(2, 1, 2)
plot(f, abs( AH(y) ))
title('DFT of signal')
ylabel('abs( A^H y )')
xlabel('Frequency')
```



Discrete Fourier Transform DFT

- Important formulas
  - $e^{(j\pi)} = -1$ - $e^{(-j\pi)} = -1$
  - $e^{(j\pi/2)} = j$ - $e^{(-j\pi/2)} = -j$
  - $e^{(-j2\pi)} = 1$ - $e^{(-j3\pi)} = -1$
  - $e^{(-j3\pi/2)} = j$ - $e^{(-j9\pi/2)} = -j$
  - $e^{(-j\alpha)} = \cos(\alpha) - j\sin(\alpha)$
  - $e^{(j\alpha)} = \cos(\alpha) + j\sin(\alpha)$

Discrete Fourier Transform DFT

- Example: compute DFT of the sequence f(x) = {1,0,0,1}

- Solution

applying: F(k) = $\sum_{x=0}^{N-1} f(x) e^{(-j2\pi kx/N)}$, where k=0, 1, …, N-1

∵ N=4

∴ F(k) = $\sum_{x=0}^{3} f(x) e^{(-j2\pi kx/4)}$

= f (0) $e^{(0)}$ + f (1) $e^{(-j2\pi k/4)}$ + f (2) $e^{(-j\pi k)}$ + f (3) $e^{(-j3\pi k/2)}$

= 1 + 0 + 0 + $e^{(-j3\pi k/2)}$

= 1 + $e^{(-j3\pi k/2)}$

Discrete Fourier Transform DFT

When k=0

$F(0) = 1 + e^{0} = 1 + 1 = 2$

When k=1

$F(1) = 1 + e^{-j3\pi/2} = 1 + j$

When k=2

$F(2) = 1 + e^{-j3\pi} = 1 - 1 = 0$

When k=3

$F(3) = 1 + e^{-j9\pi/2} = 1 - j$

$\therefore F(k) = \{2, 1+j, 0, 1-j\}$

Discrete Fourier Transform DFT

- Apply DFT on an image can be done using one of the following methods:
  - 1D DFT: F(k) = kernel * f(x)
  - 2D DFT: F(k, l) = kernel * f(x,y) * kernel^(Transpose)
- Example

Calculate 4-point DFT for the sequence x(n) = {0,1,2,3} using matrix method

∵ the 4-point DFT in one dimension

$\therefore \mathrm{F(k)} = \mathrm{kernel} * \mathrm{input\ sequence}$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0+1+2+3 \\ 0-j-2+3j \\ 0-1+2-3 \\ 0+j-2-3j \end{bmatrix} = \begin{bmatrix} 6 \\ -2+2j \\ -2 \\ -2-2j \end{bmatrix}$$

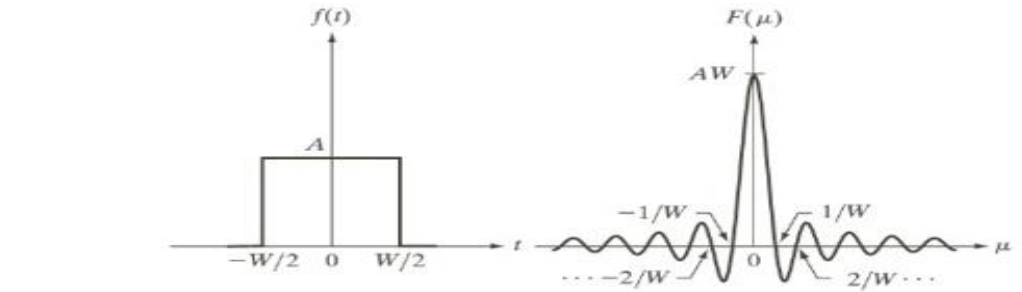
Extension to Two Variables

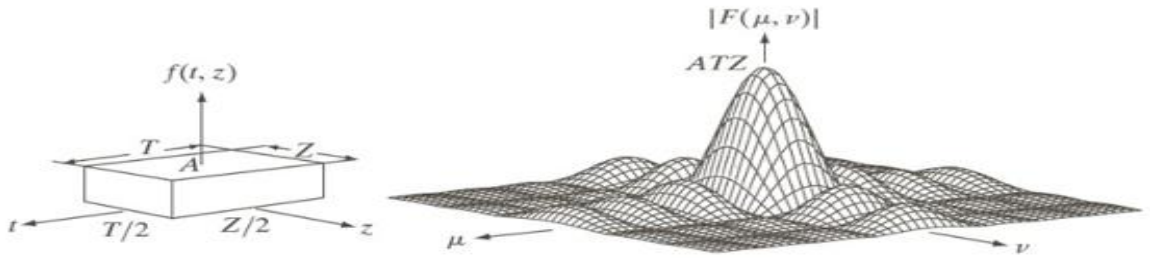
- Functions on Two Variables
  - 2D Discrete Fourier Transform

$$F(\mu, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z)\, e^{-j2\pi(\mu t + vz)}\, dt\, dz$$

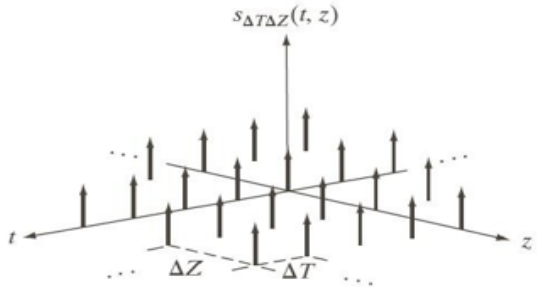
$$f(t, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\mu, v)\, e^{j2\pi(\mu t + vz)}\, d\mu\, dv$$

Example





2D Sampling



2D Impulse Train

$$s[t, z] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta[t - m\Delta T, z - n\Delta Z] \qquad (8)$$

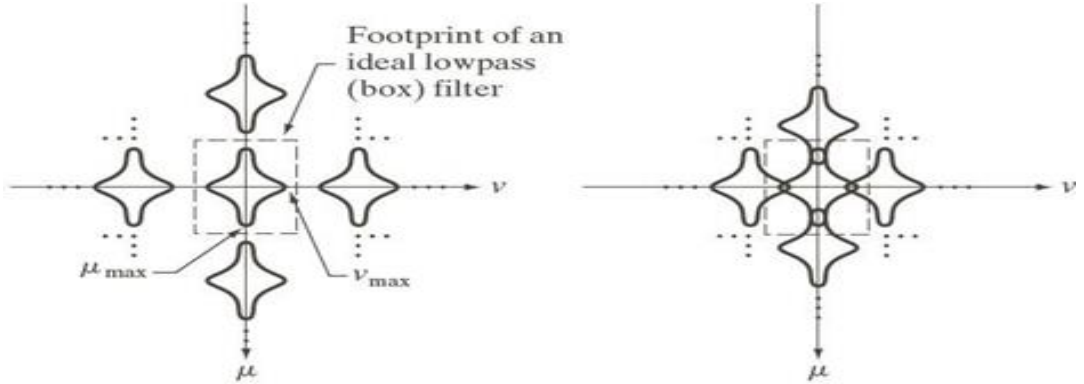
- $f(t, z)\, s[t, z] \Rightarrow$ sampled function.

2D Sampling Theorem

- Aliasing

$$\Delta T < \frac{1}{2\mu_{max}}$$

$$\Delta Z < \frac{1}{2v_{max}}$$



Spatial Aliasing in Images

a b c

FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3×3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

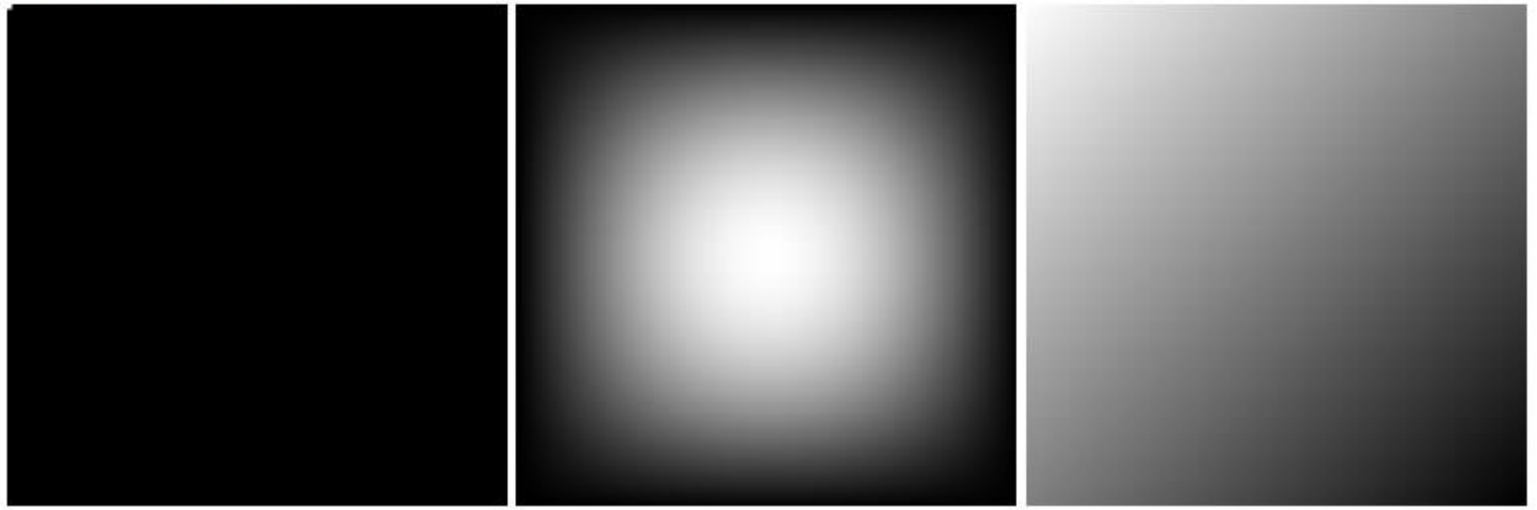
DFT and IDFT properties

- **Shift Property:**
  - As in one dimension, there is a simple relationship that can be derived for shifting an image in one domain or the other. Since both the space and frequency domains are considered periodic for the purposes of the transforms, shifting means rotating around the boundaries. The equations describing this are:

$$f(x,y)e^{j2\pi(u_0x+v_0y)/N} \Leftrightarrow F(u-u_0,v-v_0)$$
$$f(x-x_0,y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0+vy_0)/N}$$

Example

Shift Property

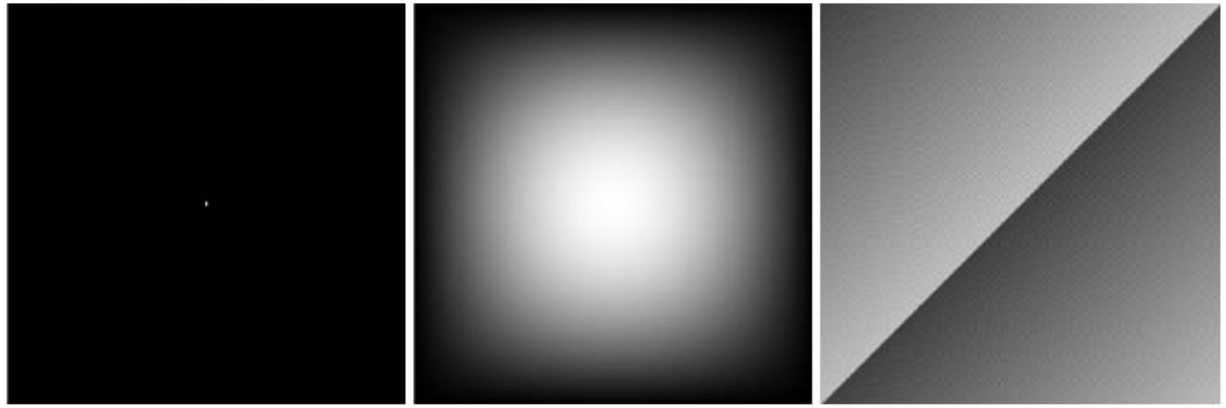


Scale Property

- Just as in one dimension, shrinking in one domain causes expansion in the other for the 2D DFT. This means that as an object grows in an image, the corresponding features in the frequency domain will expand. The equation governing this is:

$$f(ax,by) \leftrightarrow \frac{1}{|ab|} F\left(\frac{u}{a},\frac{v}{b}\right)$$

Scale Property



Rotation Property

- This is a property of the 2D DFT that has no analog in one dimension. Because of the separability of the transform equations, the content in the frequency domain is positioned based on the spatial location of the content in the space domain. This means that rotating the spatial domain contents rotates the frequency domain contents. This can be formally described by the following relationship:

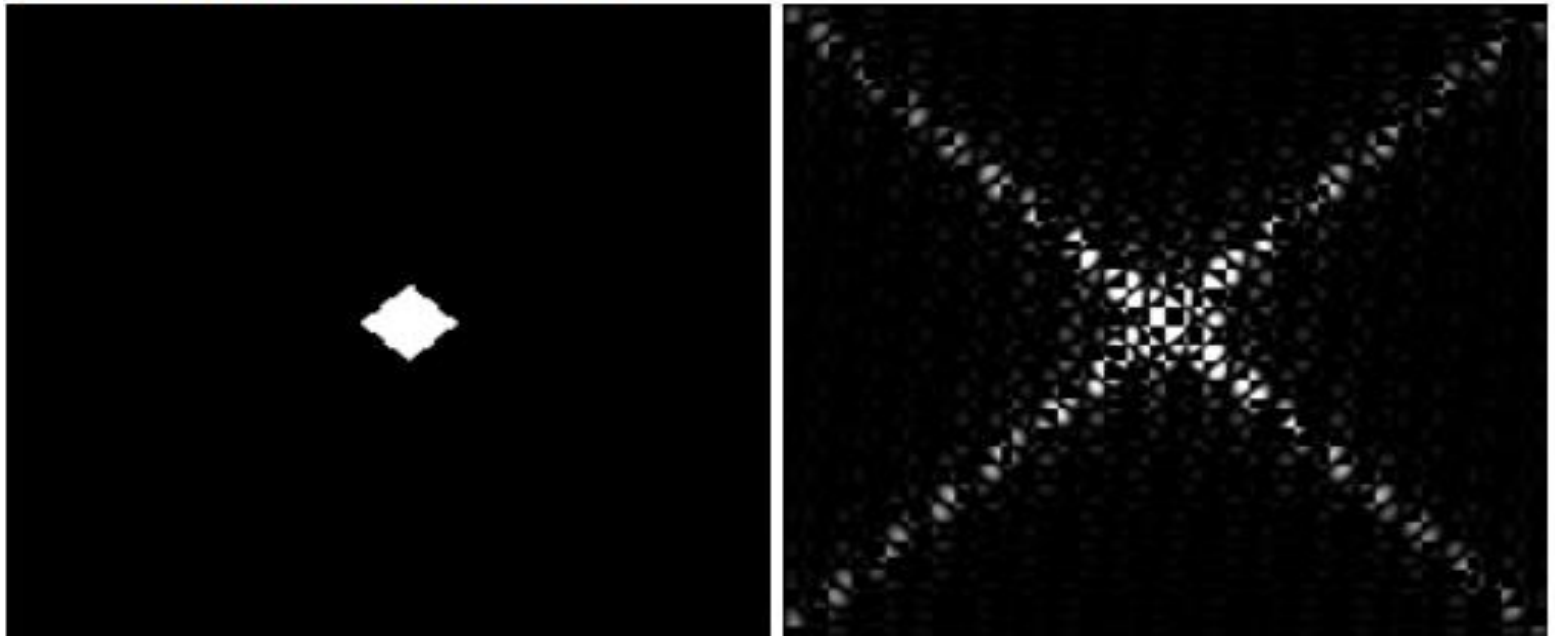
$$\left.\begin{array}{l} f(x,y) \leftrightarrow F(u,v) \\ f(x',y') \leftrightarrow F(u',v') \end{array}\right\}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Example

Rotation Property



MATLAB Code

```
% 2D DFT
function [X] = dft(x, N)
% If we don't specify how many points for the DFT we want,
% then assume it is the length of the sequence
if(nargin == 1)
N = length(x);
end
% If we don't specify the right parameters, spit out an error
if(nargin < 1 || nargin > 2)
error('Not enough parameters');
end
% Check if we're supplying a row vector. If we are,
% the transpose as a column for the matrix multiplication.
dim = size(x,2);
if(dim ~= 1)
x = x.'; % Make sure we don't conjugate
end
% Create indices for DFT matrix
% First create values of K
ind = 0:(N-1);
ind = ind(ones(1,N),:);
% Now create values of N
ind2 = ind.';
% Create DFT matrix
DN = exp(-1j*2*pi*ind.*ind2/N);
% Calculate N-pt DFT
X = DN * x;
X = X.';
% Sweet... no for loops!
end
======
```

```
% 2D IDFT
function [x] = idft(X, N)
% If we don't specify how many points for the DFT we want,
% then assume it is the length of the sequence
if(nargin == 1)
N = length(X);
end
% If we don't specify the right parameters, spit out an error
if(nargin < 1 || nargin > 2)
error('Not enough parameters');
end
% Check if we're supplying a row vector. If we are,
% the transpose as a column for the matrix multiplication.
dim = size(X,2);
if(dim ~= 1)
X = X.'; % Make sure we don't conjugate
end
% Create indices for DFT matrix
% First create values of K
ind = 0:(N-1);
ind = ind(ones(1,N),:);
% Now create values of N
ind2 = ind.';
% Create DFT matrix
DN = exp(-1j*2*pi*ind.*ind2/N);
% Now find its inverse so we can find the inverse DFT
DN = (1/N)*DN';
% Calculate N-pt IDFT
x = DN * X;
x = x.';
% Sweet... no for loops!
end
```

thank you!