DIGITAL IMAGE PROCESSING

UNIT-2

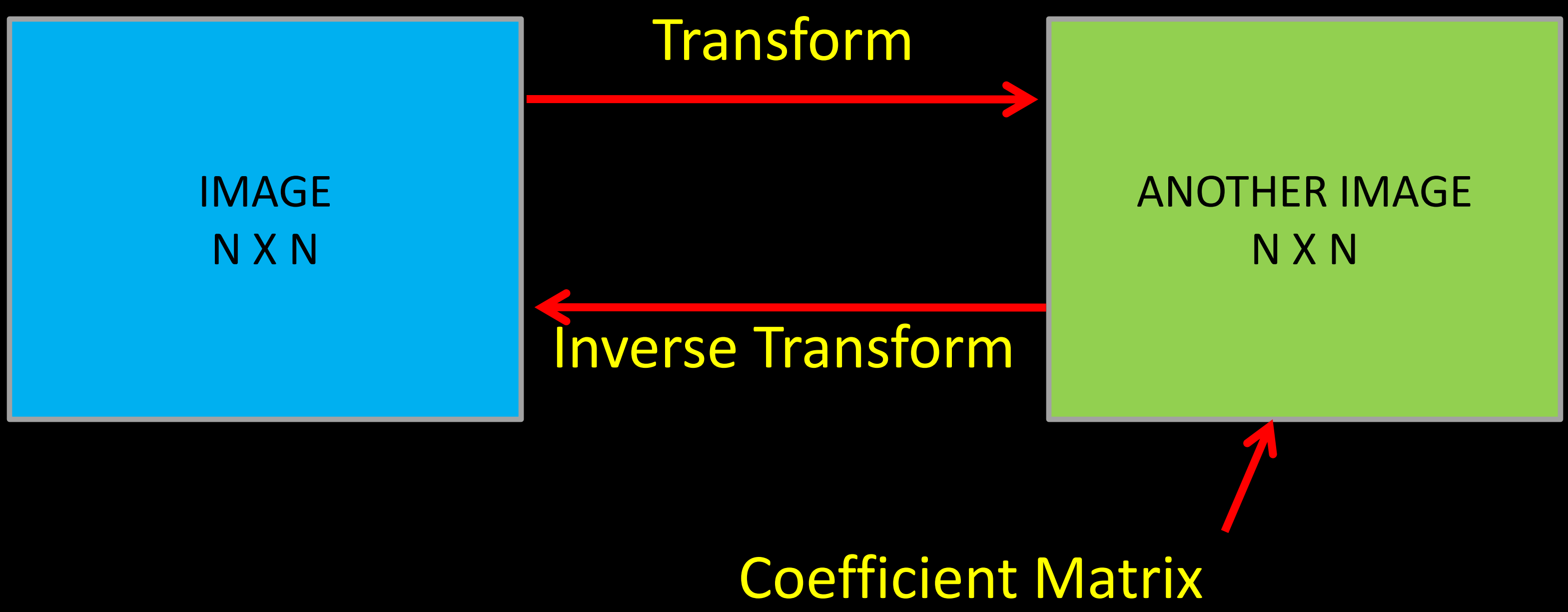
IMAGE TRANSFORMS

by

Paresh Kamble

Introduction

What is Image Transform?



Introduction

- Applying Transform we get another image of same size.
- After applying Inverse Transform we get the original image back.
- Then what is the use of Image Transformation?

Introduction

<u>Applications:</u>

- Preprocessing
  - Filtering
  - Enhancement, etc
- Data Compression
- Feature Extraction
  - Edge Detection
  - Corner detection, etc

Introduction

What does the Image Transform do?

It represents the given image as a series summation of a set of Unitary Matrices.

What is a Unitary Matrix?

A Matrix 'A' is a unitary matrix if

$A^{-1} = A^{*T}$ where A^* is conjugate of A

Unitary Matrix --------> Basis Functions

Introduction

An arbitrary continuous signal x(t) can be represented by a series summation of set of orthogonal basis functions.

The series expansion is given as:

$$x(t) = \sum_{n=0}^{\infty} c_n a_n(t) \qquad \text{infinite series expansion}$$

$$\hat{x}(t) = \sum_{n=0}^{N-1} c_n a_n(t) \qquad \text{finite series expansion}$$

It gives an approximate representation of x(t).

Introduction

In discrete signals

We have a series of 1D sequence of samples represented by:

$\{u(n) \; ; \; 0 \le n \le N-1\}$

We have N number of samples, so we represent this set by a vector ‘u’ of dimension ‘N’.

For transformation:

We pre-multiply this matrix u by unitary matrix of dimension NxN.

$$v = A \,.\, u$$

Where, v -> transformed vector

A -> Transformed matrix

Introduction

After expansion:

$$v(k) = \sum_{n=0}^{N-1} a(k, n)\, u(n) \qquad ; k = 0, 1, \ldots\ldots N-1$$

If vector 'A' is an unitary matrix we can get back our original vector 'u'

So pre-multiplying v by A-1

$u = A^{-1}v = A^{*T}v$

Representing above equation in the form of a series summation,

$$u(n) = \sum_{n=0}^{N-1} a^*(k, n)\, v(k) \qquad ; n = 0, 1, \ldots\ldots N-1$$

Introduction

Expanding matrix a(k, n), it is of the form,

$$a(k, n) = \begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & \ldots a(0,n) \\ a(1,0) & a(1,1) & a(1,2) & \ldots \\ a(2,0) & a(2,1) & a(2,2) & \ldots \\ \ldots & & & \\ a(k,0) & & & a(k,n) \end{bmatrix}$$

$a^*(k, n)$ is the column vector of matrix A^{*T}

These column vectors are usually called the basis vectors of A.

$$u(n) = \sum_{n=0}^{N-1} a^*(k, n)\, v(k) \qquad ; n = 0, 1, \ldots\ldots N-1$$

Where, u(n) is series summation of basis vectors

Introduction

The concept of representing the vector as a series summation of basis vectors can be expanded to 2D signals(image) as well

Consider any image $u(m, n)$; $0 \leq m, n \leq N-1$ of dimension $N \times N$.
Transformation on this image is given by:

$$v(k, l) = \sum_{m, n = 0}^{N-1} a_{k, l}(m, n)\, u(m, n) \qquad 0 \leq k, l \leq N-1$$

Thus, we have N^2 number of unitary matrices.

$$u(m, n) = \sum_{k, l = 0}^{N-1} a^*_{k, l}(m, n)\, v(k, l) \qquad 0 \leq m, n \leq N-1$$

$V = \{v(k, l)\}$ is the transform coefficient.

Introduction

During Inverse Transformation if all coefficients are not considered then an approximate reconstructed image will be generated.

$$\hat{u} = \sum_{k=0}^{P-1} \sum_{l=0}^{Q-1} v(k, l)\, a^*_{k,l}(m, n)$$

So, we are considering only P x Q coefficients instead of N^2.

The sum of squared error will be given by:

$$€^2 = \sum_{m,n=0}^{N-1} \sum [u(m, n) - \hat{u}(m, n)]^2$$

This error will be minimized if the set of basis images $a_{k,l}(m, n)$ is complete.

Computations required : $O(N^4)$

Introduction

To reduce Computational Complexity:

We have to make use of separable unitary transforms

$a_{k,l}(m, n)$ is separable if it can be represented in the form

$$a_{k,l}(m, n) = a_k(m) \,.\, b_l(n) \approx a(k, m)\, b(l, n)$$

where, $\{ a_k(m), k=0, 1, \ldots.N-1\}$

$\{ b_l(n), l = 0, 1, \ldots..N-1\}$

they are set of complete orthogonal set of basis vectors.

If we represent $A \approx \{a(k, m)\}$ & $B \approx \{b(l, n)\}$ in the form of matrices then both should be unitary so that

$AA^{*T} = A^T A^* = I$

If this is correct then we say that the transformation is a separable transformation.

Introduction

Usually we assume both the matrices A & B to be same.
Then the transformation equation changes to

$$v(k, l) = \sum_{m,n=0}^{N-1} \sum a(k, m)\, u(m, n)\, a(l, n)$$

In terms of matrices it is given by:

$V = AUA^T$ where all the matrices are f the order of N x N

By applying inverse transformation we will have he original matrix from the coefficient matrix.
Similarly, inverse transformation can now be written as:

$$u(m, n) = \sum_{k,l=0}^{N-1} \sum a^*(k, m)\, v(k, l)\, a^*(l, n)$$

$U = A^{*T}VA^*$
They are 2D separable transformations.

Introduction

$V = AUA^T$

$V^T = A[AU]^T$

By general matrix arithmetic, we know

if two matrices A & U of order N x N when multiplied requires $O(N^3)$ computations.

Thus it takes $O(2N^3)$ computations over all instead of $O(N^4)$ when separable transform are not considered.

Basis Images

Suppose if a^*_k denotes k^{th} column of A^{*T}.

If the matrix is defined as

$A^*_{k,\,l} = a^*_k \, a^{*T}_l$

where, a^*_l is k^{th} column of A^{*T}

We can have

$$u(m, n) = \sum_{k,\,l=0}^{N-1}\sum v(k,l)\, a^*_{k,l}(m, n)$$

Basis Images

In matrix form,

$$U = \sum_{k,\,l=0}^{N-1}\sum v(k,l)\, A^*_{k,l}$$

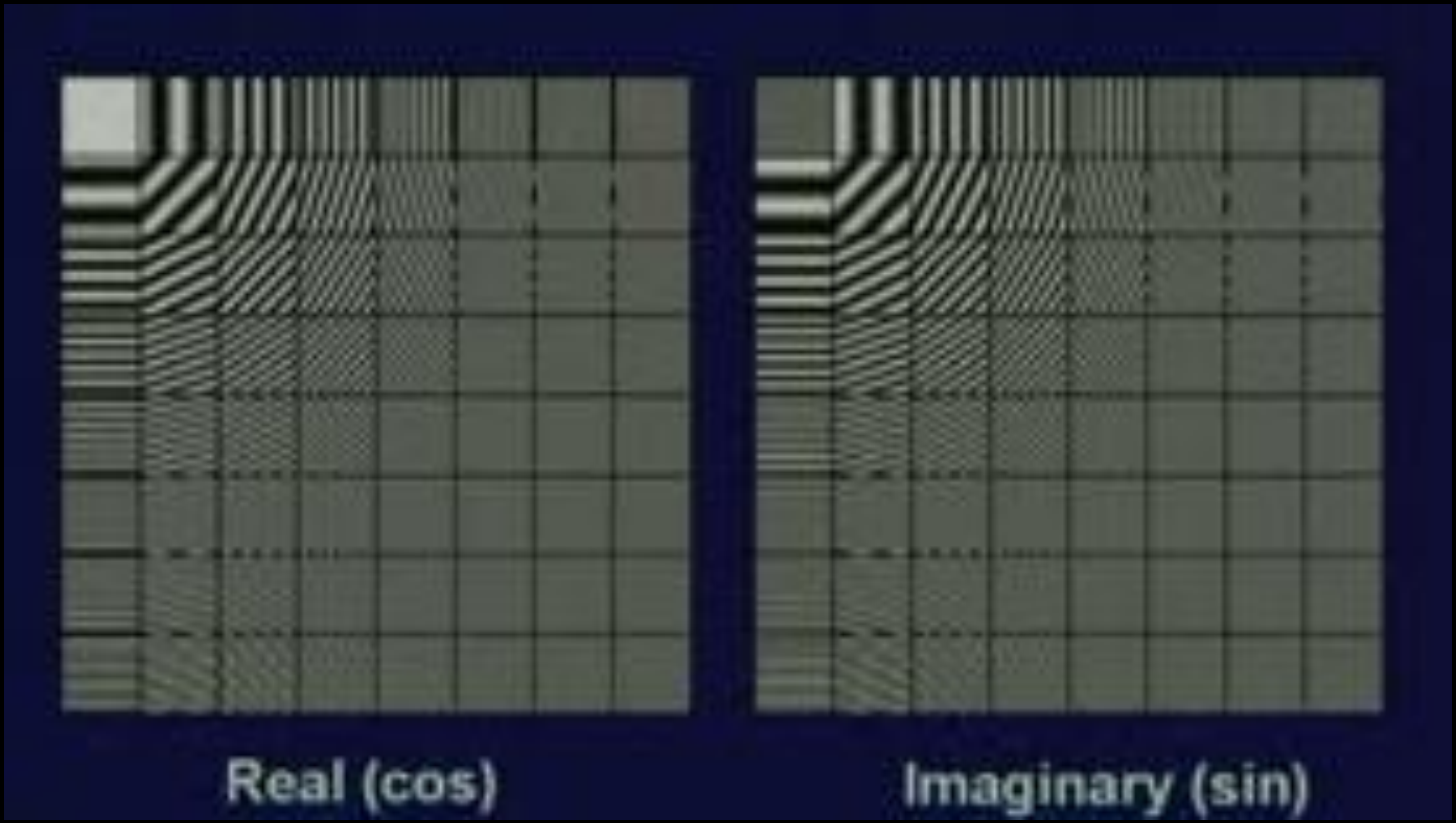
It can be observed that our original image matrix now is represented by a linear combination of N square matrices $A^*_{k,l}$ with each having dimension of N x N.

The matrices $A^*_{k,l}$ is known as the basis images.

The aim of transform is to represent the input image in the form of linear combination of a set of basis images.

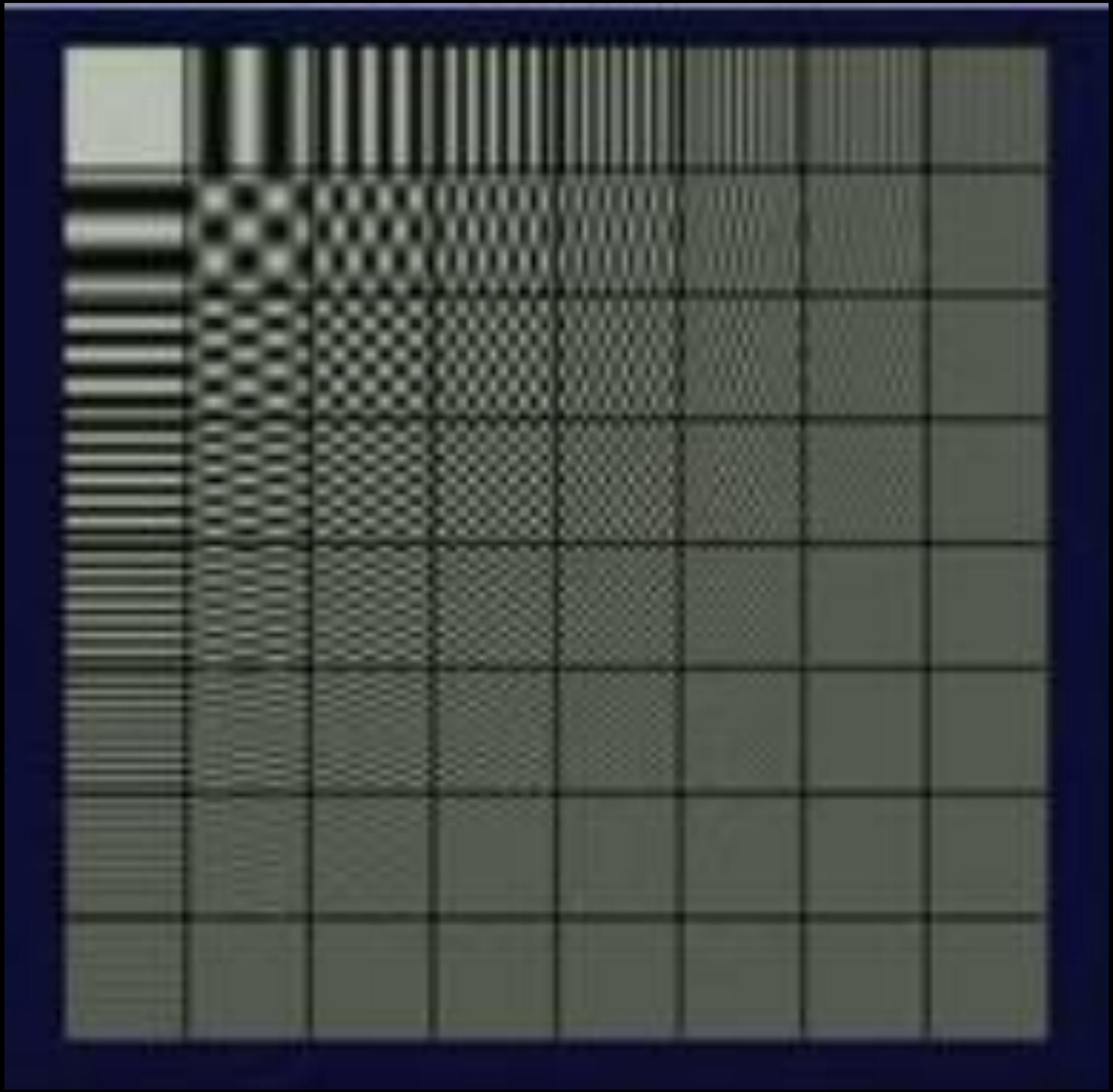
Basis Images

DFT basis function:



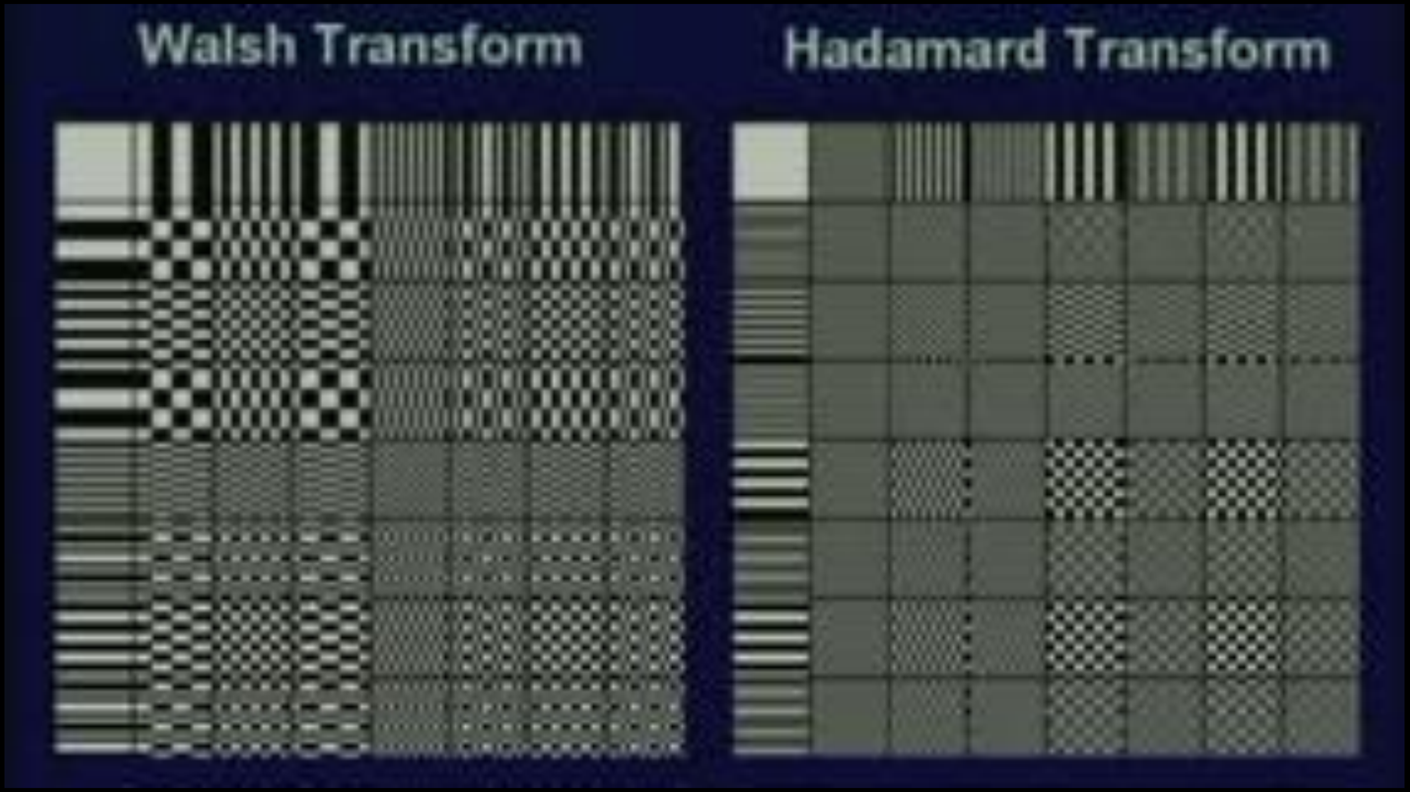
Basis Images

DCT Basis images



Basis Images

Basis images



Basis Images

Ex. 1) $A = (1/\sqrt{2})\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ $\quad U = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

$$V = (½)\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= (½)\begin{bmatrix} 4 & 6 \\ -2 & -2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}$$

Basis Images: $\quad A^*_{k,l} = a^*k \,.\, a^*l$

$A^*_{00} = (½)\begin{bmatrix} 1 \\ 1 \end{bmatrix}[1\ 1] = (½)[1\ 1;\ 1\ 1]$

$A^*_{01} = A^*_{10} = (½)[1\ -1;\ 1\ -1]$

$A^*_{11} = (½)[1\ -1;\ -1\ 1]$

Discrete Fourier Transform

Assume f(x) to be a continuous function of some variable x.

Then FT of the variable f(x) is given by

$$F\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)\, e^{-j2\pi ux}\, dx$$

where, u is the frequency term.

Requirement of the function:

(i) Function must be continuous.

(ii) Function must be integrable.

Similarly, IFT is found by

$$F^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)\, e^{j2\pi ux}\, du$$

But F(u) must be integrable.

Discrete Fourier Transform

F(u) is in general a complex function

It can therefore be broken down into real & imaginary parts.

$F(u) = R(u) + jI(u) \quad \Rightarrow |F(u)|\, e^{j\phi(u)}$

where, $|F(u)| = [R^2(u) + I^2(u)]^{1/2}$ is Fourier spectrum of f(x);

$\phi = \tan^{-1}\{I(u)/R(u)\}$ is the phase angle &

$P(u) = |F(u)|^2 = R^2(u) + I^2(u)$ is the power spectrum.

Discrete Fourier Transform

2D Fourier Transform of a continuous signal f(x, y) is given by:

$$F(u, v) = \int_{-\infty}^{\infty}\int f(x, y)\, e^{-j2\pi(ux + vy)}\, dx\, dy$$

2D IFT is given below:

$$f(x, y) = \int_{-\infty}^{\infty}\int F(u, v)\, e^{j2\pi(ux + vy)}\, dx\, dy$$

Where,

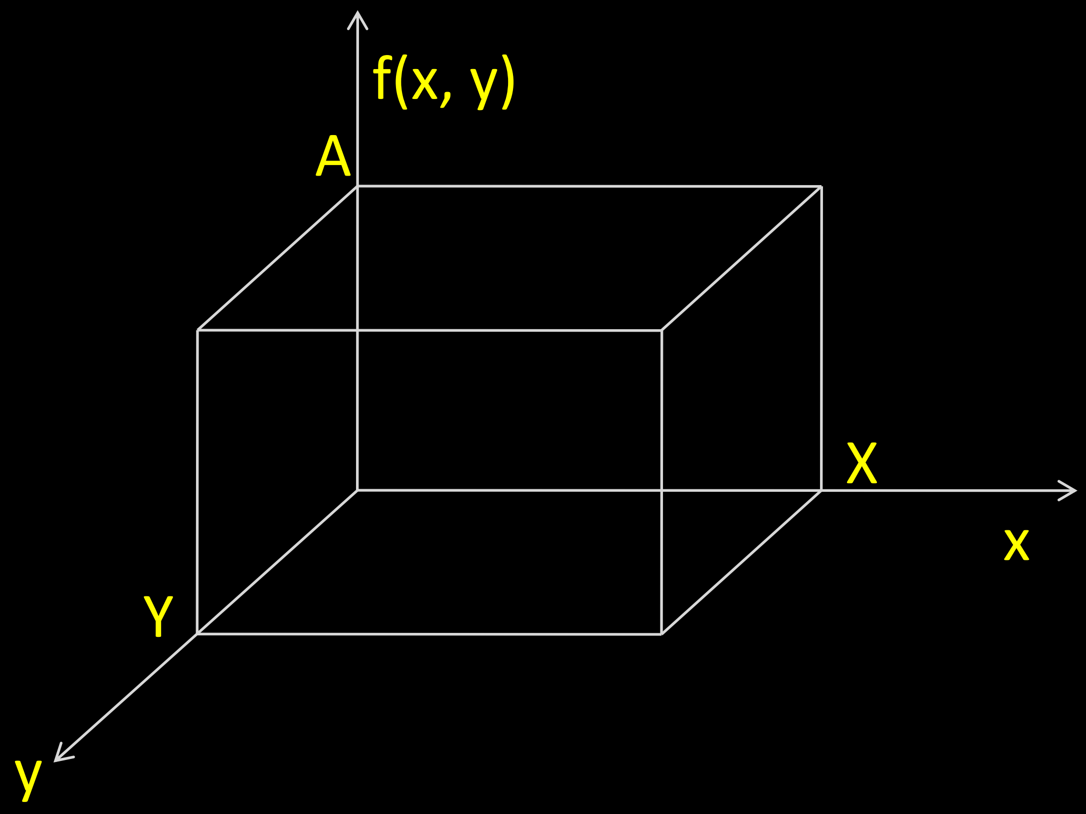
$|F(u, v)| = [R^2(u, v) + I^2(u, v)]^{1/2}$ is Fourier spectrum of f(x, y);

$\phi = \tan^{-1}(I(u, v)/R(u, v))$ is the phase angle &

$P(u) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$ is the power spectrum.

Discrete Fourier Transform

f(x, y) is a 2D signal with magnitude A, then its Fourier Transform is found out.



Discrete Fourier Transform

$$F(u, v) = \int_{-\infty}^{\infty}\int f(x, y)\, e^{-j2\pi(ux + vy)}\, dx\, dy$$

$$= A \int_{0}^{X} e^{-j2\pi ux}\, dx \,.\, \int_{0}^{Y} e^{-j2\pi vy}\, dy$$

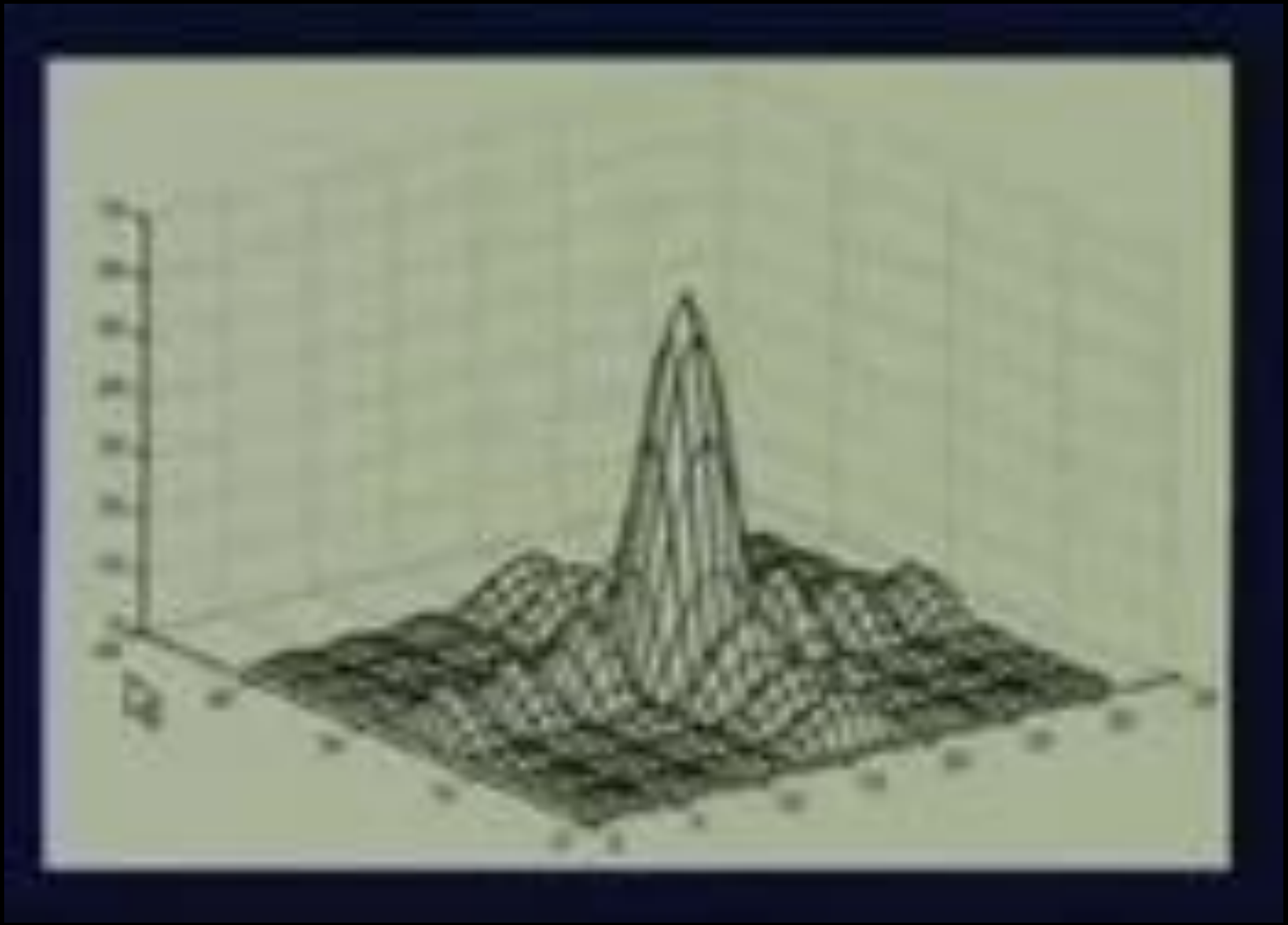
$$= A \left[(e^{-j2\pi ux})/(-j2\pi u)\right]_{0}^{X} . \left[(e^{-j2\pi vy})/(-j2\pi v)\right]_{0}^{Y}$$

$$= AXY \left[(\sin(\pi ux)\, e^{-j2\pi ux})/(\pi ux)\right] \left[(\sin(\pi vy)\, e^{-j2\pi vy})/(\pi vy)\right]$$

Fourier Spectrum of FT:

$$|F(u, v)| = Axy\ |(\sin(\pi ux))/(\pi ux)|\,|(\sin(\pi vy))/(\pi vy)|$$

Discrete Fourier Transform



Discrete Fourier Transform

2D DFT of the function g(x, y) of size M x N is expressed as:

$$F(u, v) = (1/MN) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi((ux/M)+(vy/N))} \quad \text{--------(1)}$$

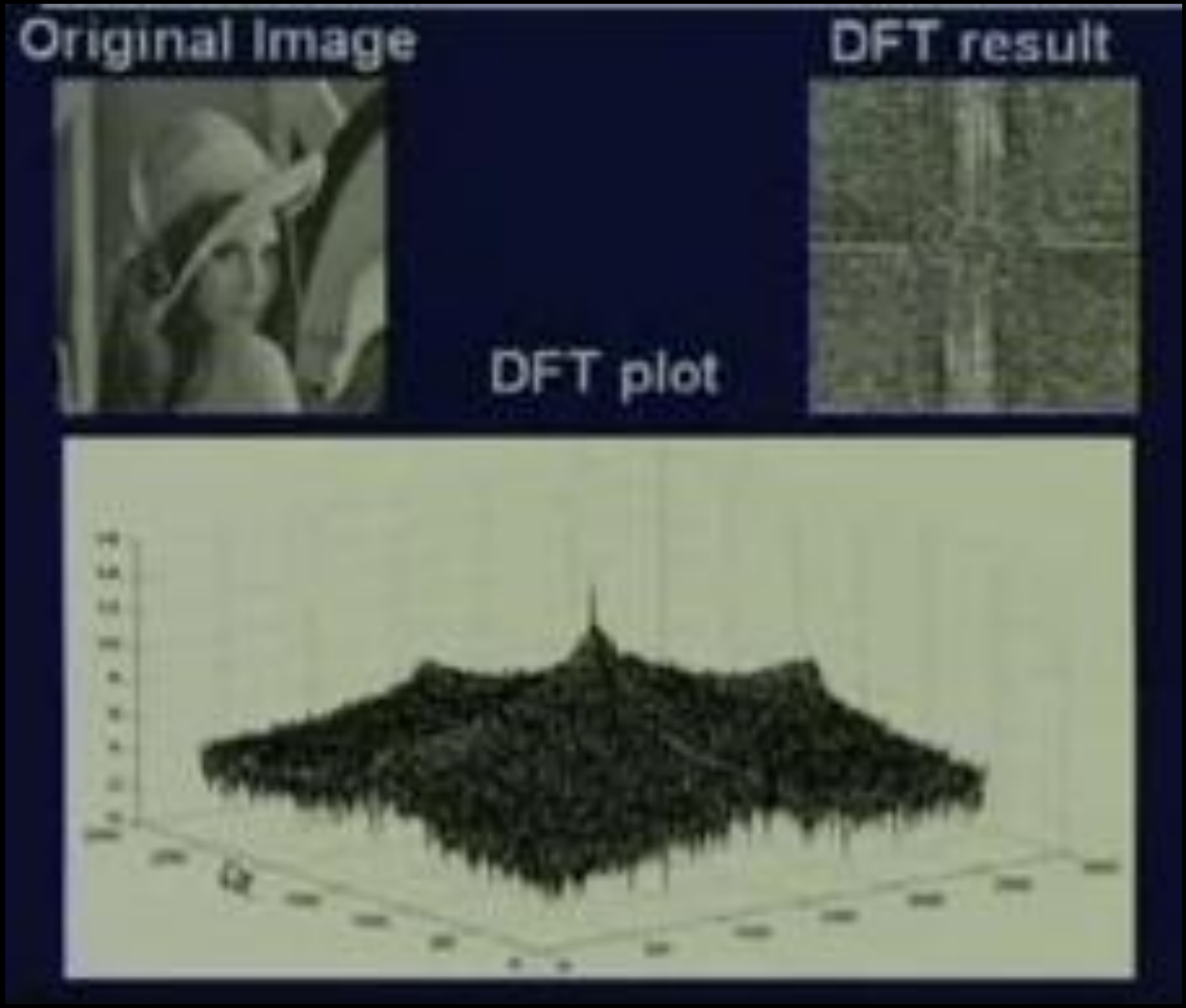
for, u = 0, 1, 2, ……M-1
v = 0, 1, 2, ……N-1

If F(u, v) is given then f(x, y) can be obtained by IDFT

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v)\, e^{j2\pi((ux/M)+(vy/N))} \quad \text{------------(2)}$$

for, x = 0, 1, ………M-1
y = 0, 1, ………N-1

Discrete Fourier Transform



Discrete Fourier Transform

Properties of Fourier Transform:

1) <u>Separability:</u>

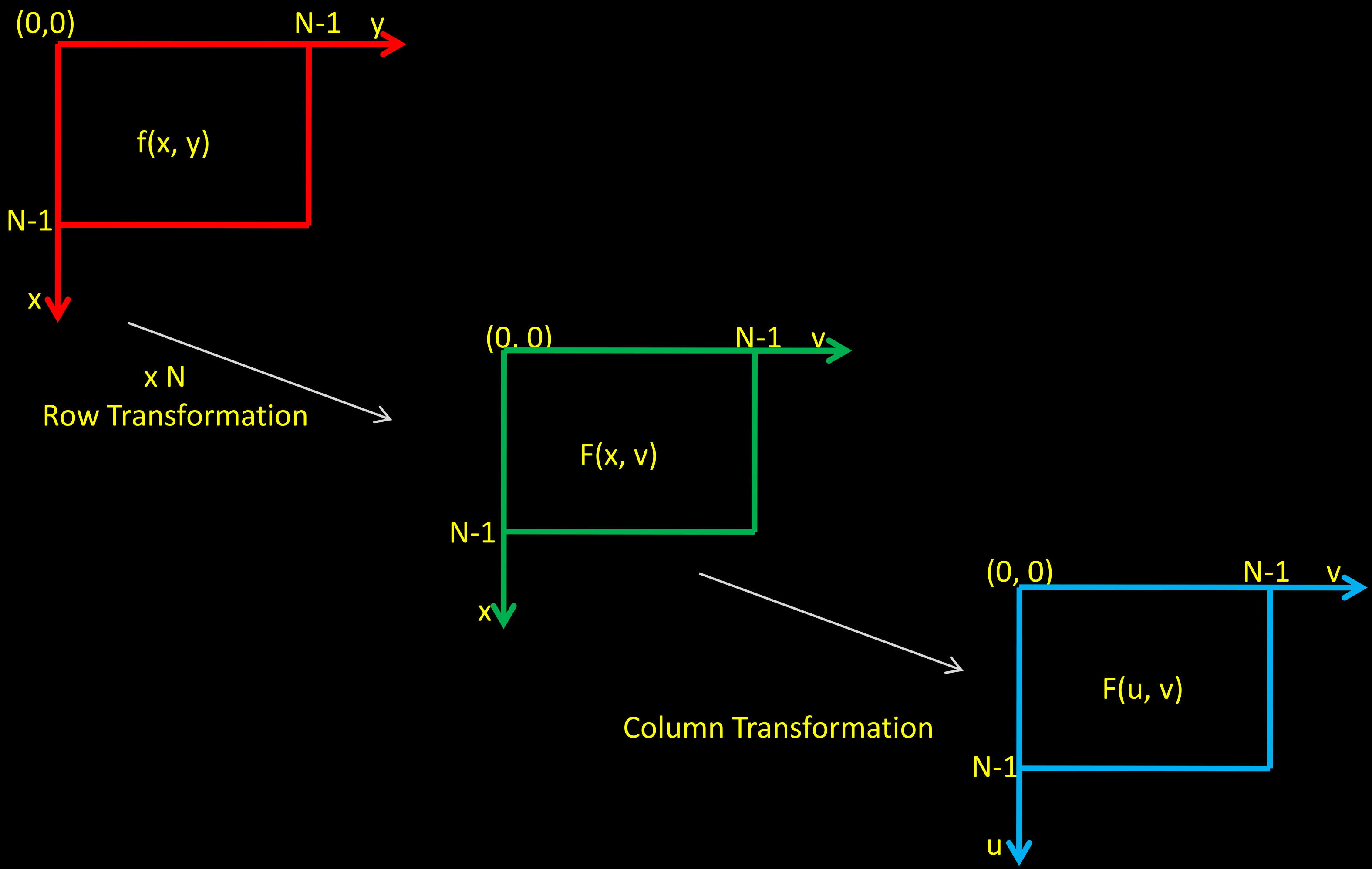
$$F(u, v) = (1/N) \sum_{x=y=0}^{N-1} \sum f(x, y)\, e^{-j(2\pi/N)(ux+vy)} \quad \text{--------(1)}$$

$$= (1/N) \sum_{x=0}^{N-1} e^{-j(2\pi ux/N)} .N.(1/N) \sum_{y=0}^{N-1} f(x, y)\, e^{-j(2\pi vy/N)}$$

$$= (1/N) \sum_{x=0}^{N-1} e^{-j(2\pi ux/N)} .N.F(x, v)$$

$$= (1/N) \sum F(x, v)\, e^{-j(2\pi ux/N)}$$

Discrete Fourier Transform



Discrete Fourier Transform

For Inverse DFT

$$f(x, y) = (1/N) \sum_{u=v=0}^{N-1} \sum F(u, v)\, e^{j(2\pi/N)(ux+vy)}$$

$$= (1/N) \sum_{u=0}^{N-1} e^{j(2\pi ux/N)}\, N.\, (1/N) \sum_{v=0}^{N-1} F(u, v)\, e^{j(2\pi vy/N)}$$

$$= (1/N) \sum_{u=0}^{N-1} N\, f(u, y)\, e^{j(2\pi ux/N)}$$

Column Transformation

Row Transformation

Discrete Fourier Transform

2) <u>Translation Property:</u>

For any image $f(x, y)$ if translated by (x_0, y_0)becomes $f(x-x_0, y-y_0)$

The translated Fourier Transform is given by:

$$F_t(u, v) = (1/N) \Sigma \Sigma f(x-x_0, y-y_0) e^{-j(2\pi/N)(u(x-x_0)+v(y-y_0))}$$

$$= (1/N) \Sigma \Sigma f(x-x_0, y-y_0) e^{-j(2\pi/N)(ux + vy)} e^{j(2\pi/N)(ux_0 + vy_0)}$$

$$= F(u, v) e^{j(2\pi/N)(ux_0 + vy_0)}$$

After Translation, the magnitude does not change, it just includes some additional phase difference.

Similarly, in IDFT, $F(u-u_0, v-v_0) = f(x, y) e^{j(2\pi/N)(u_0x + v_0y)}$

Discrete Fourier Transform

Basics of filtering in Frequency domain:

Frequency domain describes the space defined by values of the FT and the frequency variable (u & v).

Depending on equation (1) we observe the following salient features:

(i) Each term of $G(u, v)$ consists of all values of $g(x, y)$ modified by exponential terms.

(ii) We can establish a general connection between frequency component of FT & spatial characteristics of an image.

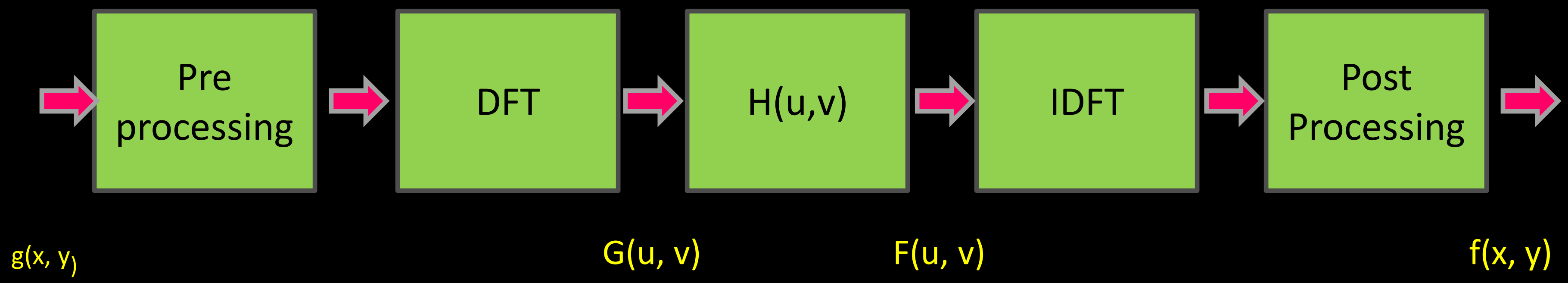
(iii) Slowest varying frequency components ($u=v=0$) corresponds to the average gray level of the image.

Discrete Fourier Transform

(iv) As we move away from origin of transform, LF correspond to slowly varying component of an image.

(v) As we move further away from origin, the higher frequency begin to correspond to faster & faster gray level changes in image. These are the edges of objects & other components of the images characterized by abrupt changes in gray levels (noise).

Filtering in Frequency Domain:



Discrete Fourier Transform

Ex. 2) Compute 2D DFT and amplitude spectrum of the following image

$$U = \begin{matrix} 5 & 3 & 0 & 2 \\ 1 & 7 & 8 & 3 \\ 4 & 2 & 2 & 2 \\ 8 & 5 & 2 & 1 \end{matrix}$$

$$F = \begin{matrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{matrix} = F^T$$

Discrete Fourier Transform

3) Periodicity & Conjugate:

$$F(u, v) = F(u, v+N) = F(u+N, v) = F(u+N, v+N)$$

$$F(u, v) = (1/N) \sum_{x=y=0}^{N-1} \sum f(x, y)\, e^{-j(2\pi/N)(ux+vy)}$$

$$F(u+N, v+N) = (1/N) \sum_{x=y=0}^{N-1} \sum f(x, y)\, e^{-j(2\pi/N)(ux+vy+Nx+Ny)}$$

$$= (1/N) \sum_{x=y=0}^{N-1} \sum f(x, y)\, e^{-j(2\pi/N)(ux+vy)}\, e^{-j(2\pi)(x+y)}$$

$$= (1/N) \sum \sum f(x, y)\, e^{-j(2\pi/N)(ux+vy)} \cdot 1$$

$$= F(u, v)$$

Discrete Fourier Transform

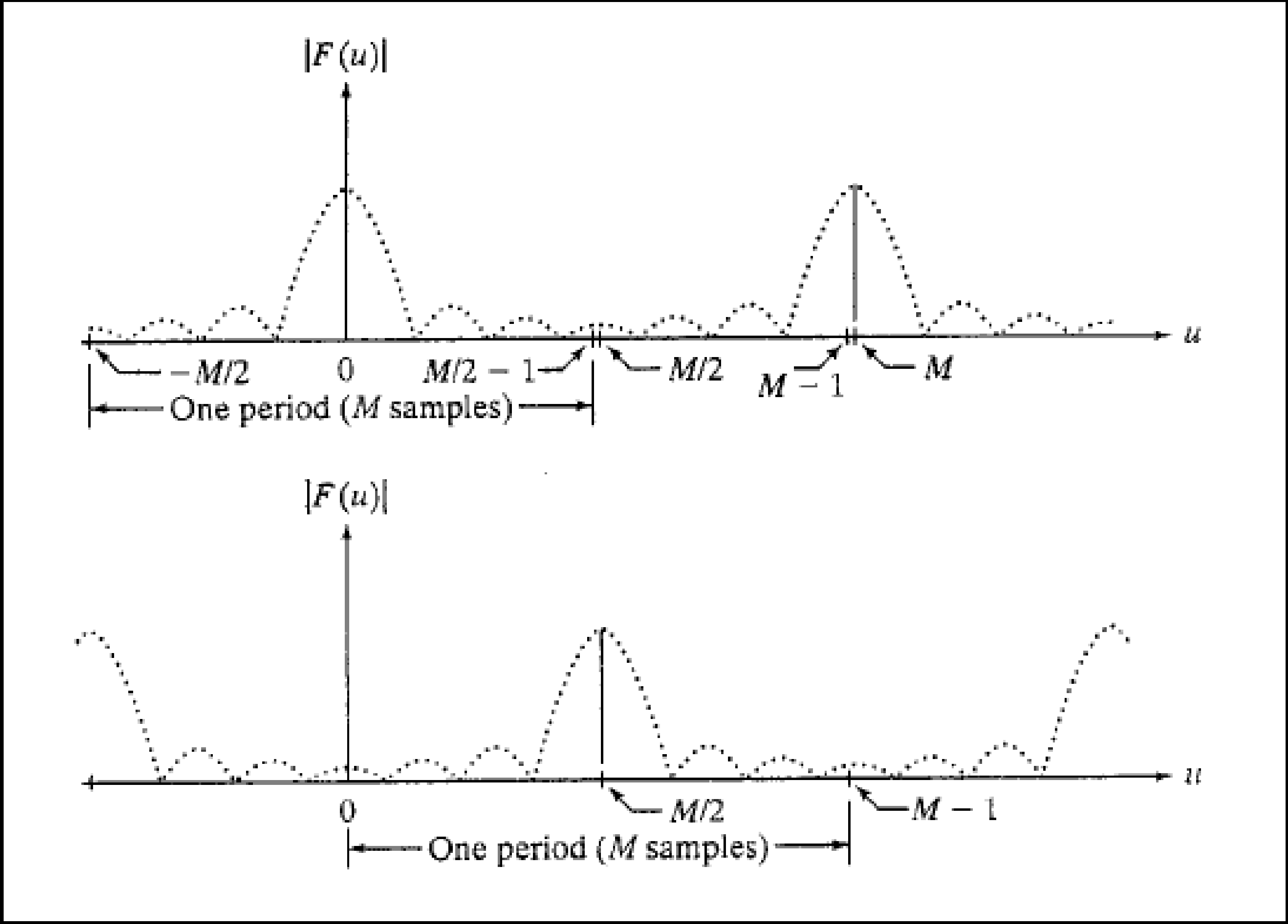
Conjugate:

$f(x, y)$ is real

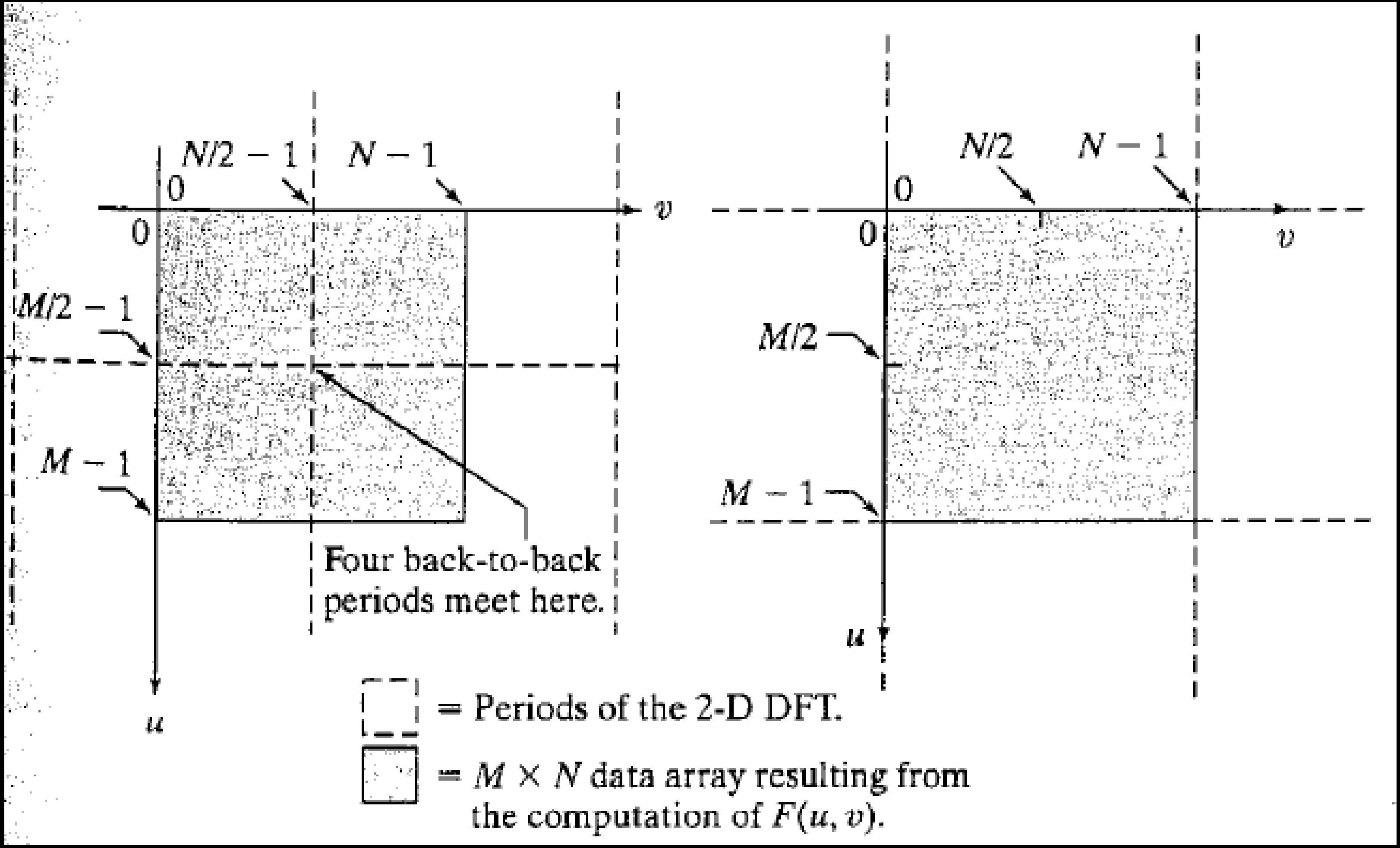
$F(u, v) = F^*(-u, -v)$

$|F(u, v)| = |F(-u, -v)|$

Discrete Fourier Transform



Discrete Fourier Transform



Spectrum obtained by multiplying f(x, y) by $(-1)^{x+y}$

Discrete Fourier Transform

4) Rotation:

$x = r \cos \theta \quad ; \qquad y = r \sin \theta \quad ;$

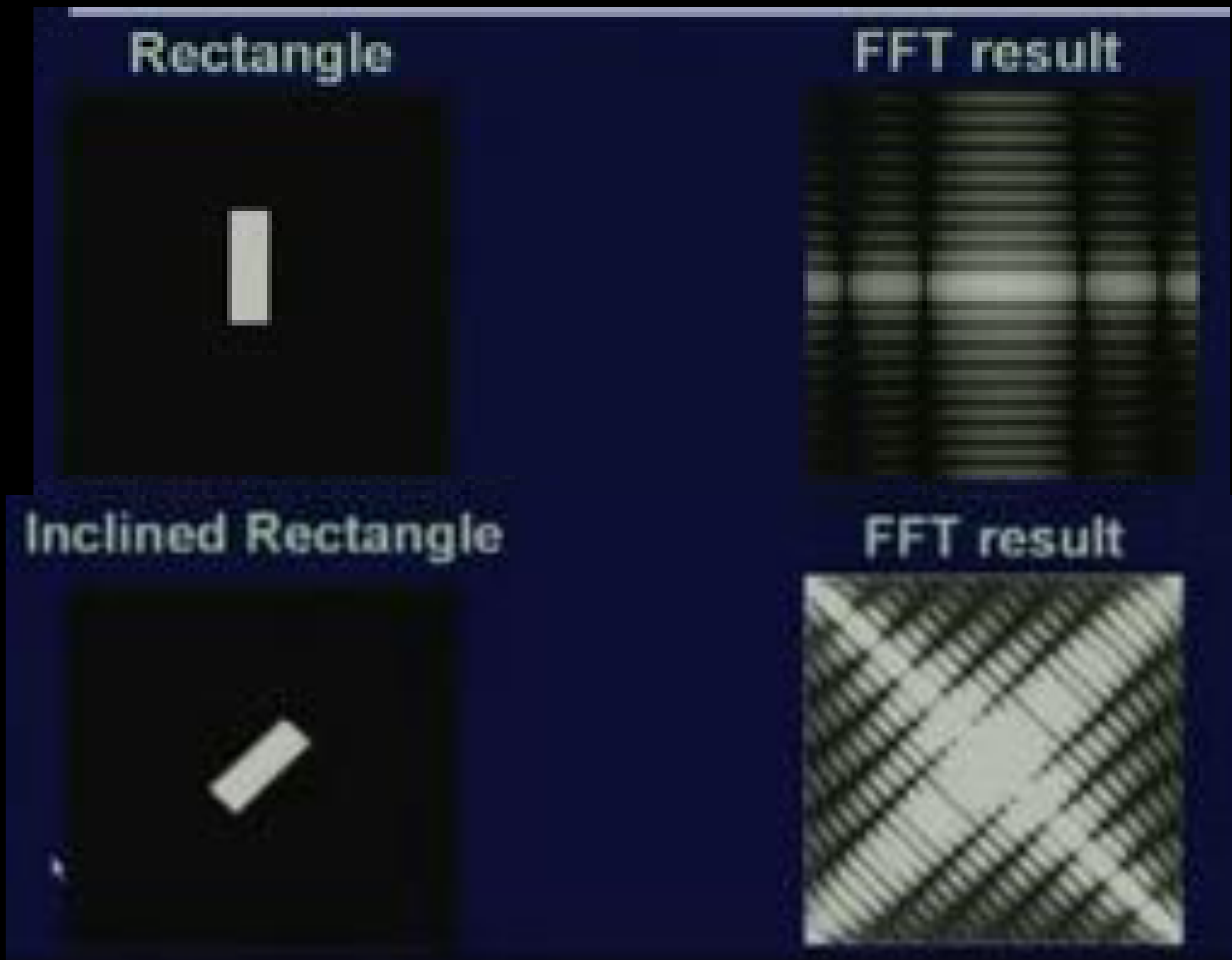
$u = \omega \cos \phi \quad ; \qquad y = \omega \sin \phi \quad ;$

$f(x, y) \Rightarrow f(r, \theta);$

$F(u, v) \Rightarrow F(\omega, \phi);$

$f(r, \theta + \theta_0) \quad \Leftrightarrow \quad F(\omega, \phi + \theta_0);$

Discrete Fourier Transform



Discrete Fourier Transform

5) Distributivity & Scaling:

Distributivity:

$F\{f_1(x, y) + f_2(x, y)\} \quad = \quad F\{f_1(x, y)\} + F\{f_2(x, y)\}$

$F\{f_1(x, y) . f_2(x, y)\} \quad \neq \quad F\{f_1(x, y)\} . F\{f_2(x, y)\}$

True for IDFT also.

Scaling: If a & b are scalar quantities.

$a . f(x, y) \quad \Leftrightarrow \quad a . F(u, v)$

$f(ax, by) \Leftrightarrow (1/|ab|) \; F(u/a , v/b);$

Discrete Fourier Transform

Average:

$$\bar{f}(x, y) = (1/N^2) \sum_{x, y = 0}^{N-1}\sum f(x, y)$$

$$F(0, 0) = (1/N) \sum_{x, y = 0}^{N-1}\sum f(x, y)$$

$$\bar{f}(x, y) = (1/N)\, F(0, 0)$$

Discrete Fourier Transform

6) Convolution:

$f(x) \,.\, g(x) \Leftrightarrow F(u) * G(u)$

$f(x) * g(x) \Leftrightarrow F(u) \,.\, G(u)$

7) Correlation:

$f(x, y) \circ g(x, y) \Leftrightarrow F^*(u, v) \,.\, G(u, v)$

$f^*(x, y) \,.\, g(x, y) \Leftrightarrow F(u, v) \circ G(u, v)$

Discrete Fourier Transform

Few other properties of DFT:

8) It is symmetric.

9) It is a fast Transform.

10) Its computational capacity is given by $N.\log_2 N$.

11) It has very good compaction for the image.

Discrete Cosine Transform

In DCT the forward transformation channel is given by:

$$g(x, y, u, v) = \alpha(u).\alpha(v).\cos[(2x+1)u\pi/2N].\cos[(2y+1)v\pi/2N]$$
$$= f(x, y, u, v)$$

inverse and forward transformation channel are identical.

They are separable as well.

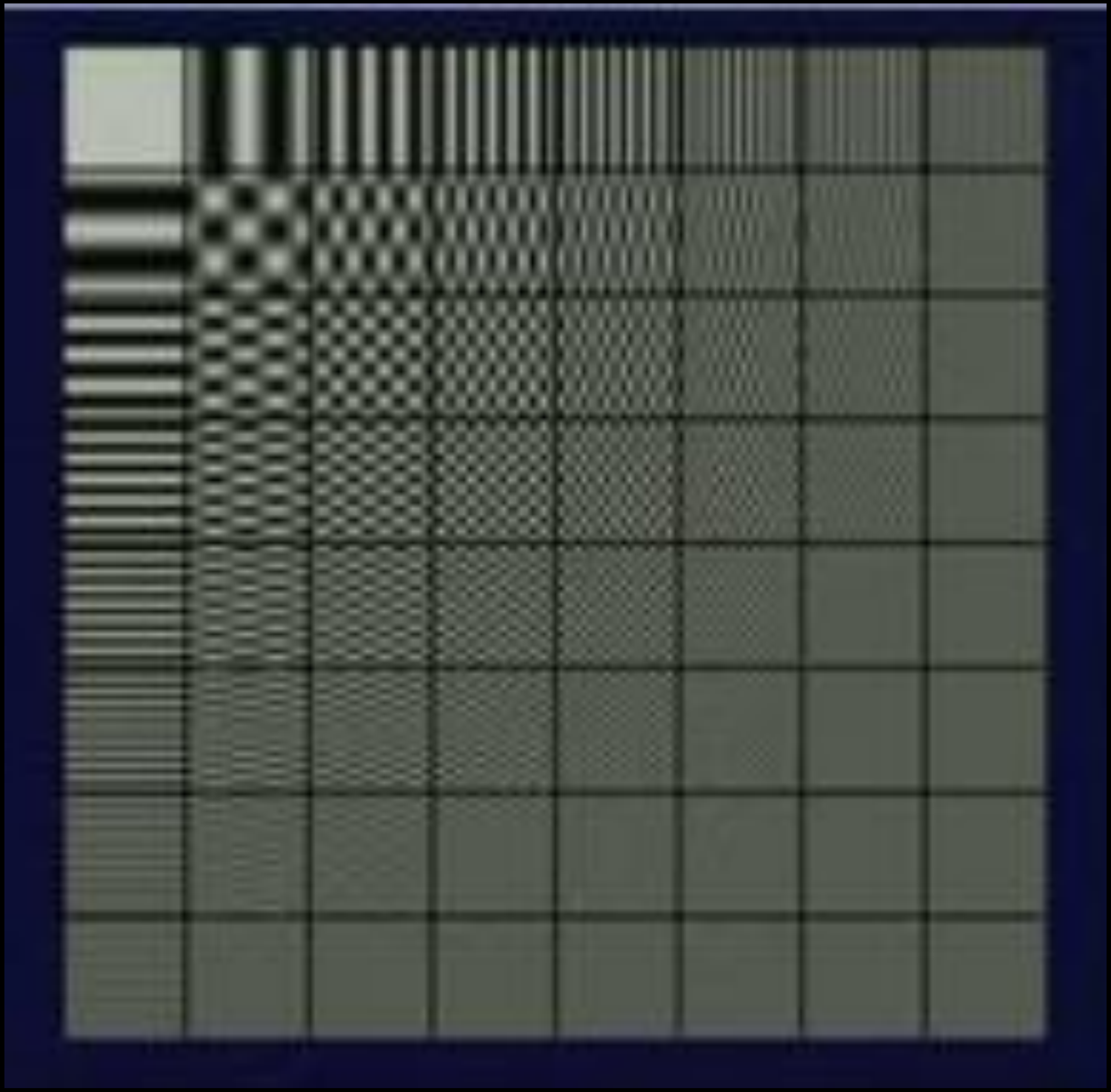
Here, $\alpha(u) = \sqrt{(1/N)} \quad u = 0$

$\sqrt{(2/N)} \quad u = 1, 2, \ldots..N-1;$

Similarly the values of $\alpha(v)$ can be found.

Discrete Cosine Transform

DCT Basis Images



Discrete Cosine Transform

Forward DCT:

$$C(u, v) = \alpha(u).\alpha(v) \sum_{x, y = 0}^{N-1} \{f(x, y) \cos[(2x+1)u\pi/2N]. \cos[(2y+1)v\pi/2N]\}$$

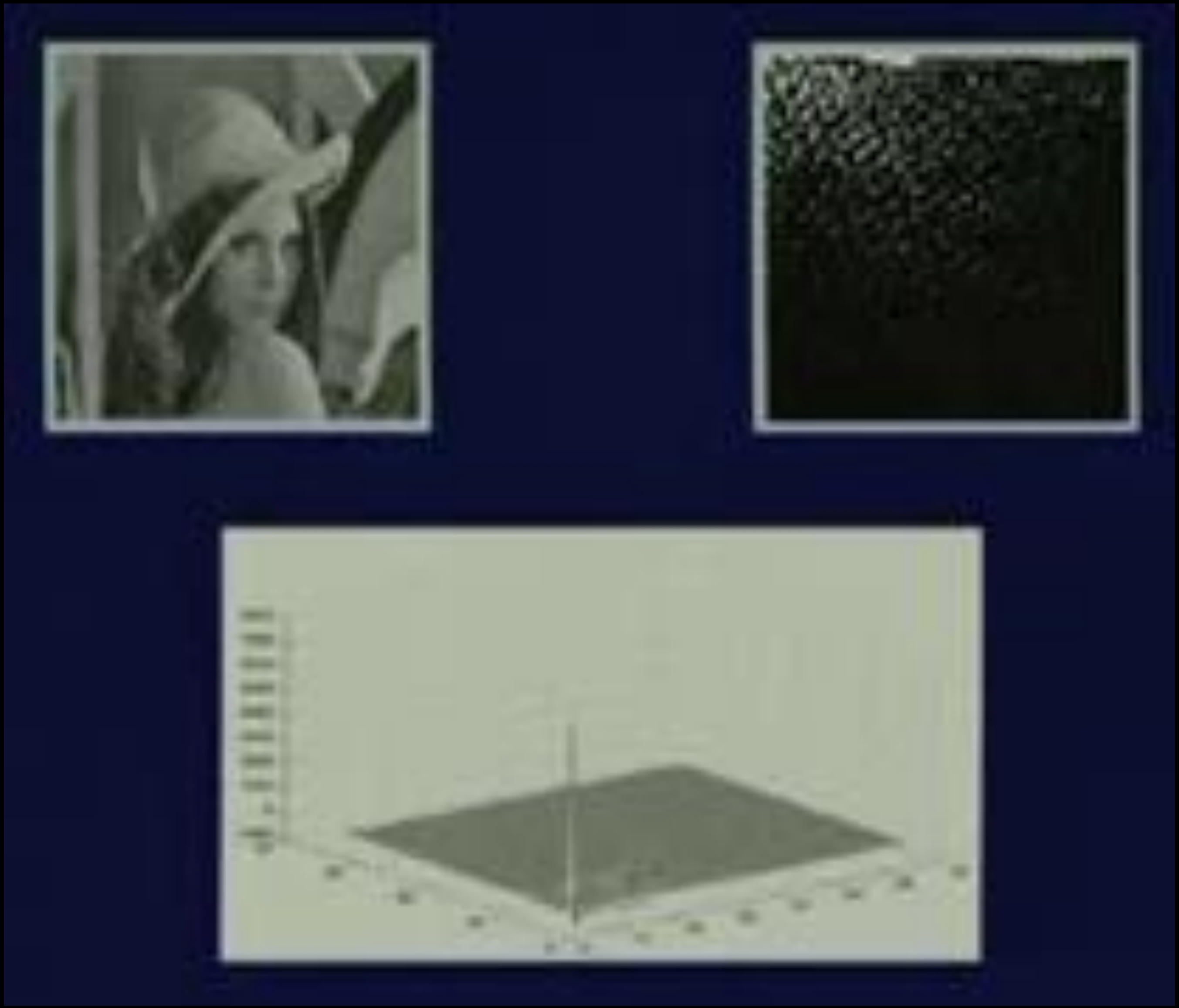
Reverse DCT:

$$f(x, y) = \sum_{u, v = 0}^{N-1} \{\alpha(u).\alpha(v)\, C(u, v) \cos[(2x+1)u\pi/2N]. \cos[(2y+1)v\pi/2N]\}$$

In DCT, energy is concentrated in a small region.

It helps in image compression.

Discrete Cosine Transform



Discrete Cosine Transform

Properties of DCT:

(1) It is real orthogonal.

$$C = C^* \quad \Rightarrow \quad C^{-1} = C^T$$

(2) It is fast transform.

(3) It can be calculated in $N\log_2 N$ operations via N-point transform.

(4) It is good for energy compaction for highly correlated data.

(5) It can be used for designing transform code & wiener filter.

(6) It is near substitute for Hotelling & KL transform.

(7) It is used for JPEG compression.

Discrete Cosine Transform

The NxN cosine transform matrix $C = \{C(k, n)\}$ also called the DCT is defined as

$$C(k, n) = 1/\sqrt{N}; \quad k = 0; \quad 0 \le n \le N-1$$

$$= \sqrt{(2/N)}.\cos((\pi(2n+1)k)/2N); \quad 1 \le k \le N-1; \quad 0 \le n \le N-1$$

Ex. 3) Find the 2D DCT matrix for N = 4.

Discrete Cosine Transform

Sol: C =

k\n	0	1	2	3
0	0.5	0.5	0.5	0.5
1	0.65	0.27	-0.27	-0.65
2	0.5	-0.5	-0.5	0.5
3	0.27	-0.65	0.65	-0.27

Ex. 4) Compute 2D DCT of the following image for N = 4

$$U = \begin{matrix} 5 & 3 & 0 & 2 \\ 1 & 7 & 8 & 3 \\ 4 & 2 & 2 & 2 \\ 8 & 5 & 2 & 1 \end{matrix}$$

V = ?

Discrete Sine Transform

The N x N Discrete Sine Transform is given by matrix $\psi = \{\psi(k, n)\}$

$\psi(k, n) = \sqrt{(2/(N+1))} \sin(\pi\{(k+1)(n+1)/(N+1)\})$, for $0 \le k; \; n \le N - 1$

Properties:

(1) It is symmetrical, real & orthogonal.

$$\psi = \psi^T, \quad \psi = \psi^*, \quad \psi^T = \psi^{-1}.$$

(2) It is faster than DCT.

(3) It is calculated by $N \log_2 N$ via 2(N+1) point FFT.

(4) It has very good energy compression property for images.

(5) It is used for estimating performances in digital image processing problems.

Discrete Sine Transform

For N = 4 the Discrete Sine Transform matrix is given by:

$$\Psi = \begin{array}{c|cccc} k \backslash n & 0 & 1 & 2 & 3 \\ \hline 0 & 0.37 & 0.6 & 0.6 & 0.37 \\ 1 & 0.6 & 0.37 & -0.37 & -0.6 \\ 2 & 0.6 & -0.37 & -0.37 & 0.6 \\ 3 & 0.37 & -0.6 & 0.6 & -0.37 \end{array}$$

WalshTransform

1D Walsh Transform kernel is given by:

$$g(x, u) = (1/N) \prod_{i=0}^{n-1} (-1)^{b_i(x)\, b_{n-1-i}(u)}$$

where, N – no. of samples

n – no. of bits needed to represent x as well as u

$b_k(z)$ – k^{th} bits in binary representation of z.

Thus, Forward Discrete Walsh Transformation is given by:

$$W(u) = (1/N) \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{n-1} (-1)^{b_i(x)\, b_{n-1-i}(u)}$$

WalshTransform

1D Inverse Walsh Transform kernel is given by:

$$h(x, u) = \prod_{i=0}^{n-1} (-1)^{b_i(x)\, b_{n-1-i}(u)}$$

Thus, Inverse Discrete Walsh Transformation is given by:

$$f(x) = \sum_{u=0}^{N-1} W(u) \prod_{i=0}^{n-1} (-1)^{b_i(x)\, b_{n-1-i}(u)}$$

Thus, both the transforms are identical except the factor of (1/N). Same algorithm is used to perform forward and inverse transformation.

WalshTransform

2D signals:

Forward Transformation kernel is given by:

$$g(x, y, u, v) = (1/N) \prod_{i=0}^{N-1} (-1)^{\{b_i(x)\, b_{n-1-i}(u) + b_i(y)\, b_{n-1-i}(v)\}}$$

Inverse Transformation kernel is given by:

$$h(x, y, u, v) = (1/N) \prod_{i=0}^{N-1} (-1)^{\{b_i(x)\, b_{n-1-i}(u) + b_i(y)\, b_{n-1-i}(v)\}}$$

WalshTransform

2D Forward Walsh Transform is given by

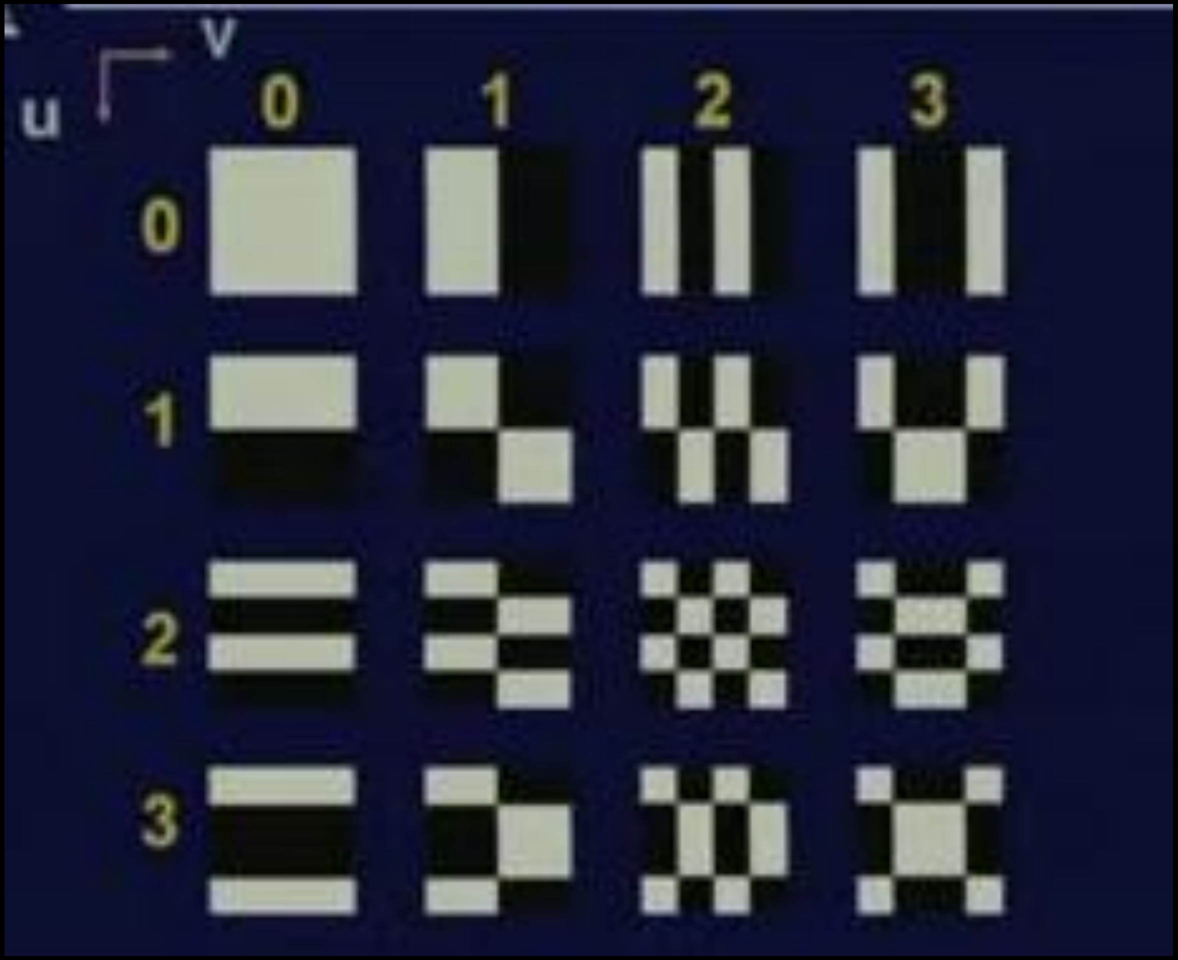
$$W(u, v) = (1/N) \sum_{x,\ y=0}^{N-1}\sum f(x, y) \prod_{i=0}^{N-1} (-1)^{\{b_i(x)\, b_{n-1-i}(u) + b_i(y)\, b_{n-1-i}(v)\}}$$

$$f(x, y) = (1/N) \sum_{u,\ v=0}^{N-1}\sum W(u, v) \prod_{i=0}^{N-1} (-1)^{\{b_i(x)\, b_{n-1-i}(u) + b_i(y)\, b_{n-1-i}(v)\}}$$

Here, the algorithm used for calculating the forward transformation is used for calculating the inverse transformation.

WalshTransform

Basis Function for Walsh Transformation:



WalshTransform

Walsh Transformation applied on image:

WalshTransform

Properties :

(1)The transform is separable and symmetric.

(2)The coefficients near origin have maximum energy and it reduces as we go further away from the origin.

(3)It has energy compaction property but NOT so strong as in DCT.

WalshTransform

Transformation kernel

G = (1/N)

	0	1	2	3	4	5	6	7	Seq
	+	+	+	+	+	+	+	+	0
	+	+	+	+	-	-	-	-	1
	+	+	-	-	+	+	-	-	3
	+	+	-	-	-	-	+	+	2
	+	-	+	-	+	-	+	-	7
	+	-	+	-	-	+	+	-	6
	+	-	-	+	+	-	-	+	4
	+	-	-	+	-	+	+	-	5

WalshTransform

Prob. 2) Find the Walsh Transformation kernel G and then find output for the given image.

$$F = \begin{matrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 1 & 4 & 2 & 3 \\ 1 & 2 & 4 & 3 \end{matrix}$$

For N = 4

$$G = (1/4) \begin{matrix} + & + & + & + \\ + & + & - & - \\ + & - & + & - \\ + & - & - & + \end{matrix}$$

Hadamard Transform

1D Forward Transformation kernel is given by:

$$g(x, u) = (1/N)\,(-1)^{\sum_{i=0}^{n-1} b_i(x)\, b_i(u)}$$

$$H(u) = (1/N) \sum_{x=0}^{N-1} f(x)\,(-1)^{\sum_{i=0}^{n-1} b_i(x)\, b_i(u)}$$

1D Inverse Transformation kernel is given by:

$$h(x, u) = (-1)^{\sum_{i=0}^{n-1} b_i(x)\, b_i(u)}$$

$$f(x) = \sum_{u=0}^{N-1} H(u)\,(-1)^{\sum_{i=0}^{n-1} b_i(x)\, b_i(u)}$$

Hadamard Transform

2D Forward Transformation kernel is given by:

$$g(x, y, u, v) = (1/N)\,(-1)^{\sum_{i=0}^{n-1} \{b_i(x)\, b_i(u) + b_i(y)\, b_i(v)\}}$$

2D Inverse Transformation kernel is given by:

$$h(x, y, u, v) = (1/N)\,(-1)^{\sum_{i=0}^{n-1} \{b_i(x)\, b_i(u) + b_i(y)\, b_i(v)\}}$$

Hadamard Transformation are separable and symmetric.

❑ 2D Hadamard Transformation can be implemented as sequence of 1D Hadamard Transformation.

Hadamard Transform

1D hadamard Transform is given by:

$$g(x, u) = (1/N)\,(-1)^{\sum_{i=0}^{n-1} b_i(x)\, b_i(u)}$$

If the term 1/N is neglected then it forms a Hadamard Matrix.

x ↓ \ u →	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	-	+	-	+	-	+	-
2	+	+	-	-	+	+	-	-
3	+	-	-	+	+	-	-	+
4	+	+	+	+	-	-	-	-
5	+	-	+	-	-	+	-	+
6	+	+	-	-	-	-	+	+
7	+	-	-	+	-	+	+	-

Hadamard Transform

Analyzing Hadamard Matrix further:

If we analyze the number of sign changes along a particular column.

It shows no straight forward relation between the no of sign changes and the values of u.

In DFT & DCT we have such direct relationship between u and the frequency.

If we want similar relation in HM then we need some sort of reordering.

This reordering can be obtained by another transformation.

Hadamard Transform

It is possible to formulate a recursive relation to formulate the Hadamard matrix.

To generate the recursive relation we consider a Hadamard matrix of lowest order of $N = 2$.

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Thus, a Hadamard matrix of order 2N can be obtained from a HM of order N by:

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & H_{-N} \end{bmatrix}$$

Thus, HM of higher order can be formed from a HM of lower dimension.

Hadamard Transform

Kernel of such modified transformation is given by:

$$g(x, u) = (1/N)\,(-1)^{\sum_{i=0}^{n-1} b_i(x)\, p_i(u)}$$

where, $p_0(u) = b_{n-1}(u)$

$p_1(u) = b_{n-1}(u) + b_{n-2}(u)$

$p_2(u) = b_{n-2}(u) + b_{n-3}(u)$

\-   -

\-   -

\-   -

$p_{n-1}(u) = b_1(u) + b_0(u)$

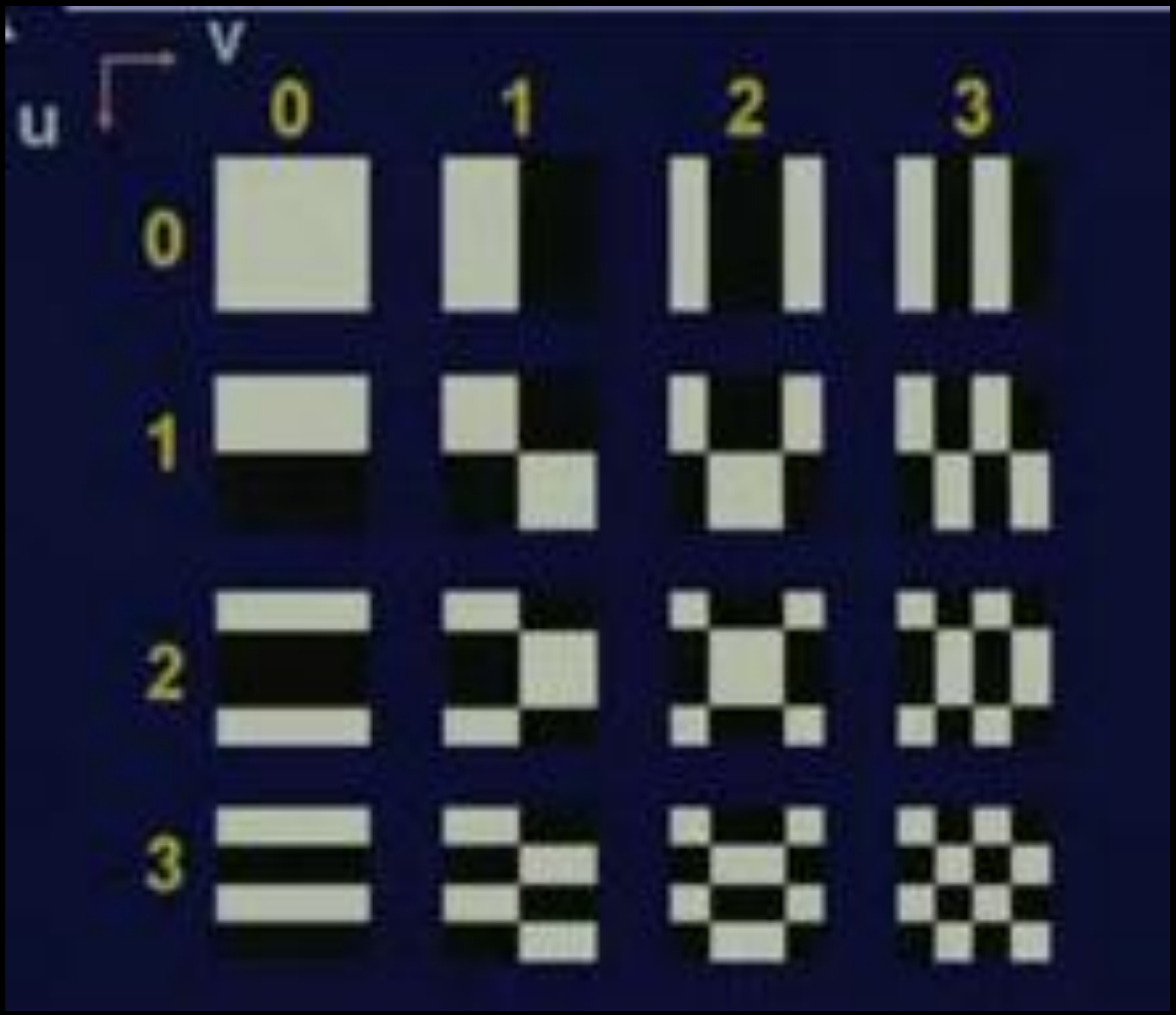
Ordered Hadamard Transform

u →

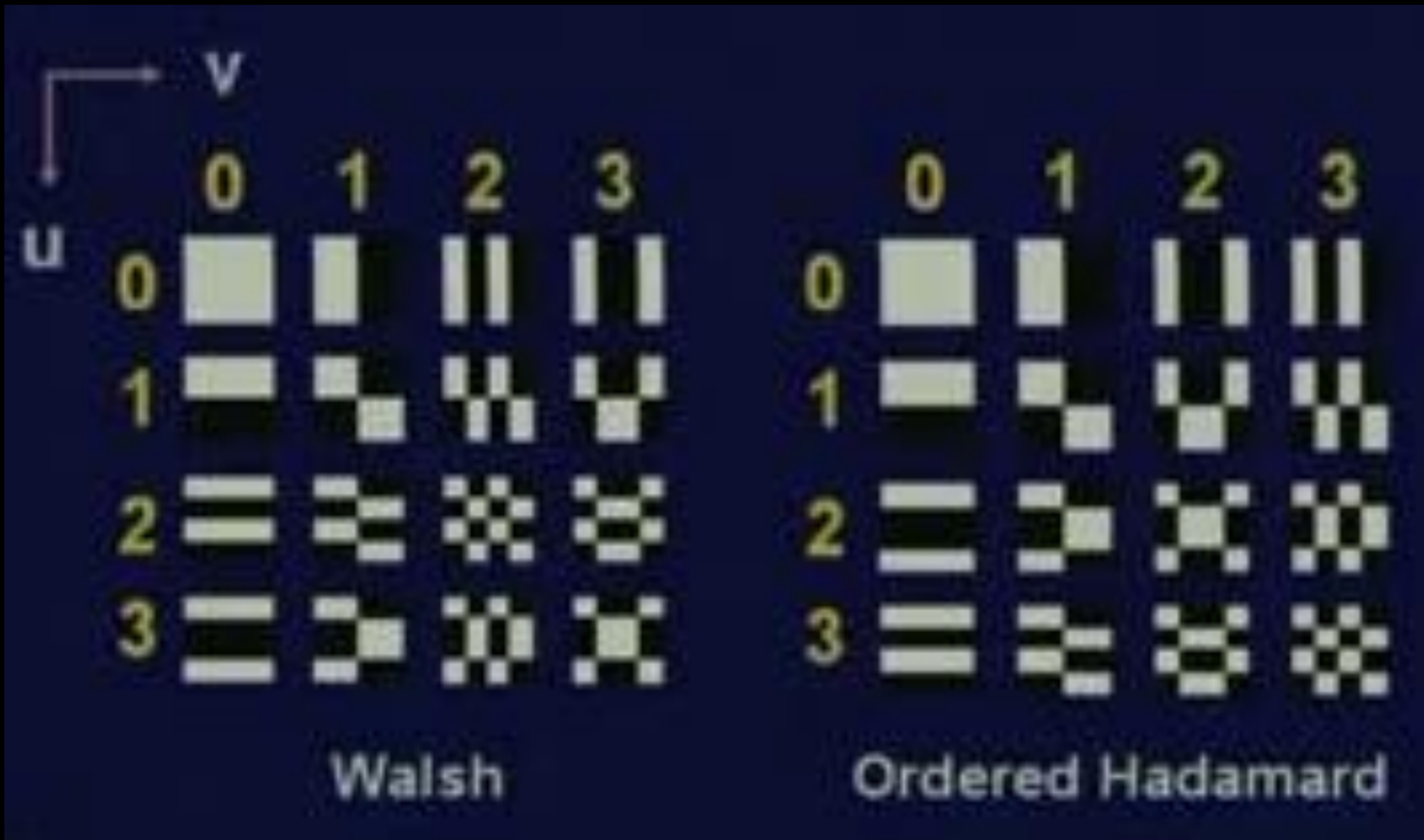
x ↓

x \ u	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-
2	+	+	-	-	-	-	+	+
3	+	+	-	-	+	+	-	-
4	+	-	-	+	+	-	-	+
5	+	-	-	+	-	+	+	-
6	+	-	+	-	-	+	-	+
7	+	-	+	-	+	-	+	-

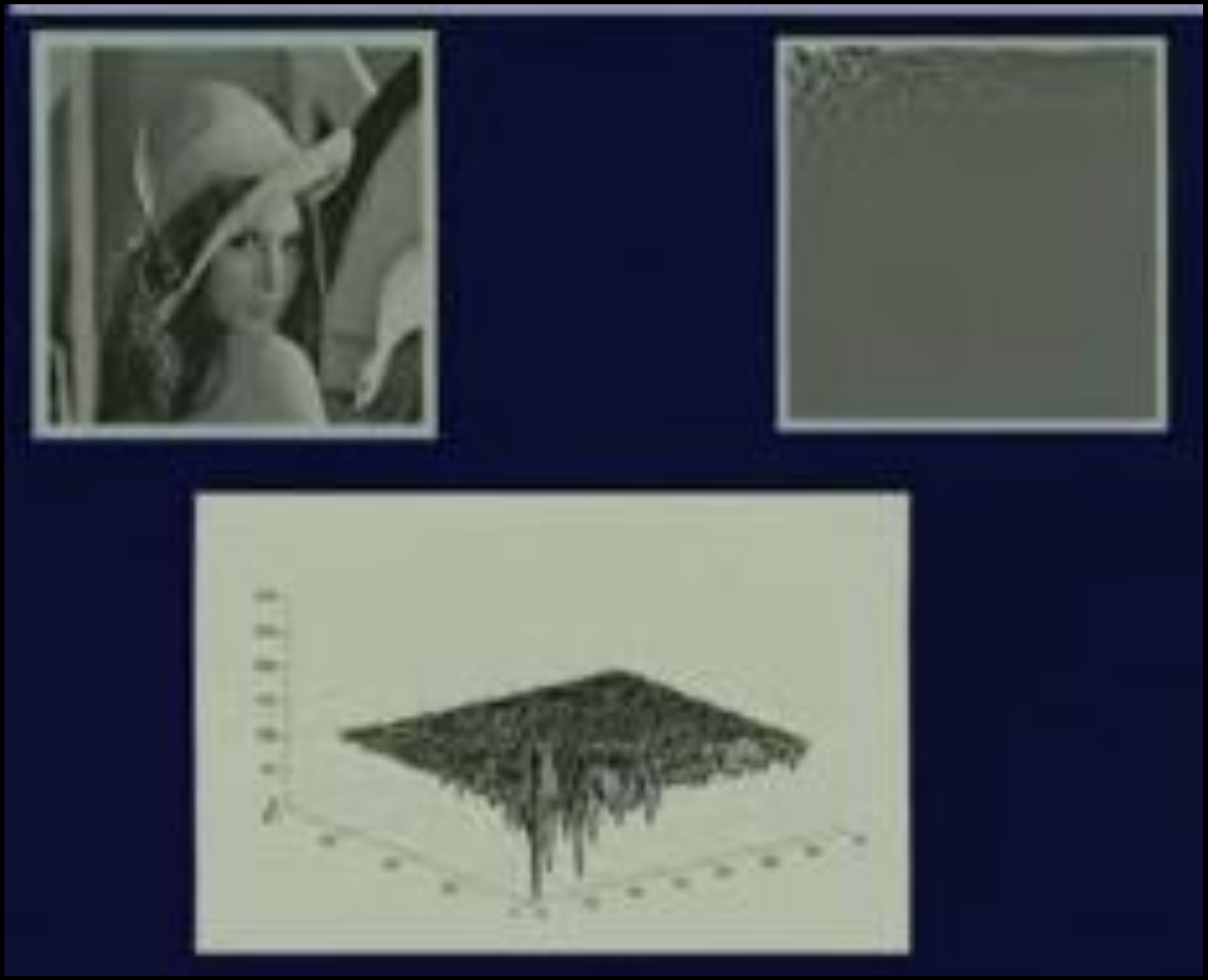
Ordered Hadamard Transform



Ordered Hadamard Transform



Hadamard Transform



Hadamard Transform

Properties of Hadamard transform:

(1) Hadamard Transform H is real, symmetric & orthogonal.

$$H = H^* = H^T = H^{-1}$$

(2) It is fast transform (faster than DCT). The 1D transformation can be implemented in $N\log_2 N$ additions and subtraction. Since HT contain only +/- values, no multiplications are required in the calculations.

(3) It has very good energy compression for highly correlated images.

(4) It is useful in hardware implementation for DFT algorithm, used in image, data compression, filtering and designing of codes.

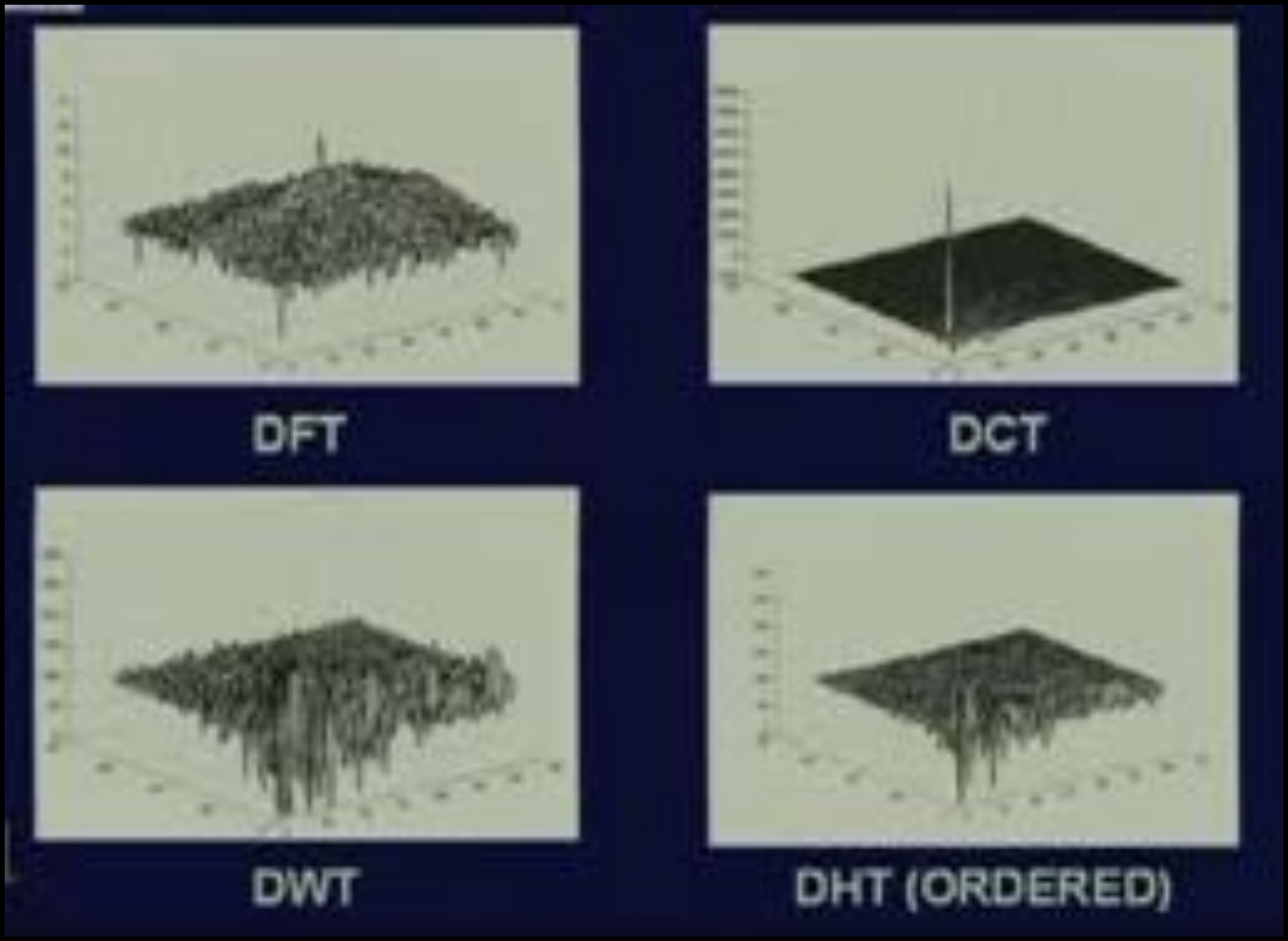
Hadamard Transform

Prob. 3) Derive 4x4 HT matrix and hence find out HT of the following image.

$$G = \begin{matrix} 1 & 2 & 3 & 4 \\ 9 & 7 & 8 & 6 \\ 4 & 5 & 6 & 7 \\ 7 & 6 & 5 & 4 \end{matrix}$$

$N = 4$ Thus $n = N\log_2 N$

Comparision



Haar Transform

The haar function $h_k(x)$ are defined on continuous interval $x \in [0, 1]$ and for $k = 0, 1, \text{-----}N-1$ where $N = 2^n$.

The integer k can be uniquely decomposed as

$$k = 2^p + q - 1$$

where $0 \leq p \leq n-1$, $q = 0,1$ for $p = 0$
& $1 \leq q \leq 2^p$ for $p \neq 0$.

Haar Transform

Prob. 4)