



# **Digital Image Processing**

**Lecture #8**  
**Ming-Sui (Amy) Lee**

# Announcement

## Class Information

- The following schedule

03/23	Lecture 5	05/11	proposal
03/30	Lecture 6 & 7	05/18	Lecture 10
04/06	Lecture 8	05/25	Lecture 11
04/13	RealSense	06/01	Lecture 12
04/20	midterm	06/08	Demo
04/27	RealSense	06/15	Demo
05/04	Lecture 9	06/22	Final Package Due

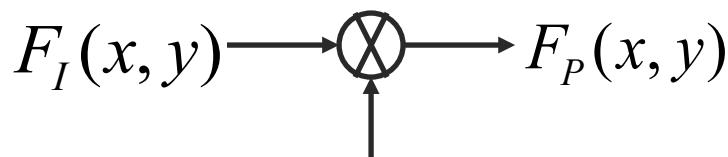
# **Image Sampling & Transforms**

# Image Sampling & Transforms

## ■ Image sampling

### ○ A/D conversion

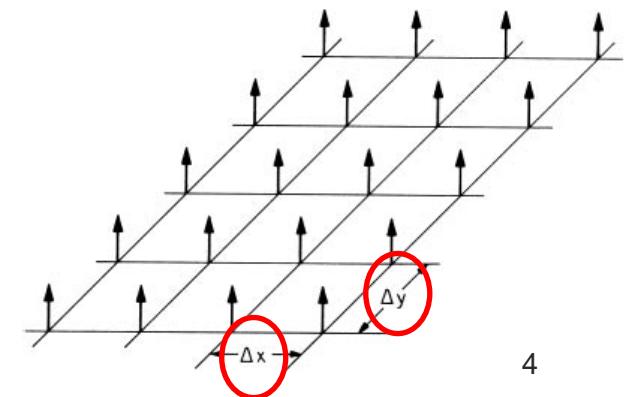
- Usually deals with arrays of numbers obtained by spatially sampling points of a physical image
- $F_I(x, y)$  : continuous, infinite-extent, ideal image field
- $F_P(x, y)$  : the sampled image



transform function

$$S(x, y) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(x - j\Delta x, y - k\Delta y)$$

Dirac delta function



# Image Sampling & Transforms

## ■ Image sampling

- Sampling function  $S(x, y) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(x - j\Delta x, y - k\Delta y)$

■ An infinite array of the Dirac delta functions  
arranged in a grid of spacing,  $(\Delta x, \Delta y)$

- Space domain

$$F_P(x, y) = F_I(x, y)S(x, y) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} F_I(j\Delta x, k\Delta y) \delta(x - j\Delta x, y - k\Delta y)$$

- Transform domain (Fourier transform)

$$\mathcal{F}_P(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_P(x, y) \exp\{-i(\omega_x x + \omega_y y)\} dx dy = \frac{1}{4\pi^2} \mathcal{F}_I(\omega_x, \omega_y) \otimes S(\omega_x, \omega_y)$$

(Convolution Theorem)

where  $S(\omega_x, \omega_y) = \frac{4\pi^2}{\Delta x \Delta y} \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(\omega_x - j\omega_{xs}, \omega_y - k\omega_{ys})$

Sampling frequency

and  $\omega_{xs} = 2\pi / \Delta x$  and  $\omega_{ys} = 2\pi / \Delta y$



# Image Sampling & Transforms

## ■ Image sampling

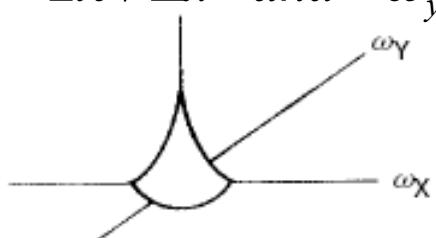
- Transform domain (Fourier transform)

- Assume the spectrum of the ideal image is band-limited

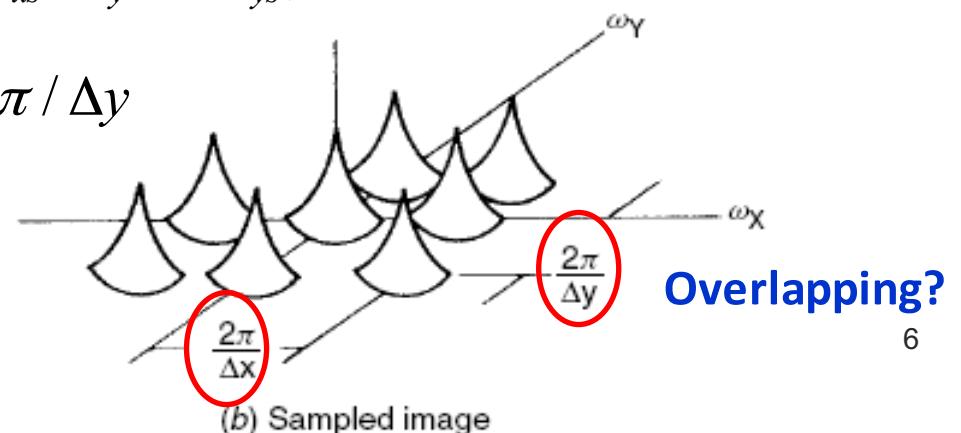
$$\begin{aligned}
 \mathcal{F}_P(\omega_x, \omega_y) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_P(x, y) \exp\{-i(\omega_x x + \omega_y y)\} dx dy = \frac{1}{4\pi^2} \mathcal{F}_I(\omega_x, \omega_y) \otimes S(\omega_x, \omega_y) \\
 &= \frac{1}{\Delta x \Delta y} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{F}_I(\omega_x - \alpha, \omega_y - \beta) \times \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(\alpha - j\omega_{xs}, \beta - k\omega_{ys}) d\alpha d\beta \\
 &= \frac{1}{\Delta x \Delta y} \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \mathcal{F}_I(\omega_x - j\omega_{xs}, \omega_y - k\omega_{ys})
 \end{aligned}$$

(Yellow box highlights the term  $\sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \delta(\alpha - j\omega_{xs}, \beta - k\omega_{ys})$ )

where  $\omega_{xs} = 2\pi / \Delta x$  and  $\omega_{ys} = 2\pi / \Delta y$



(a) Original image



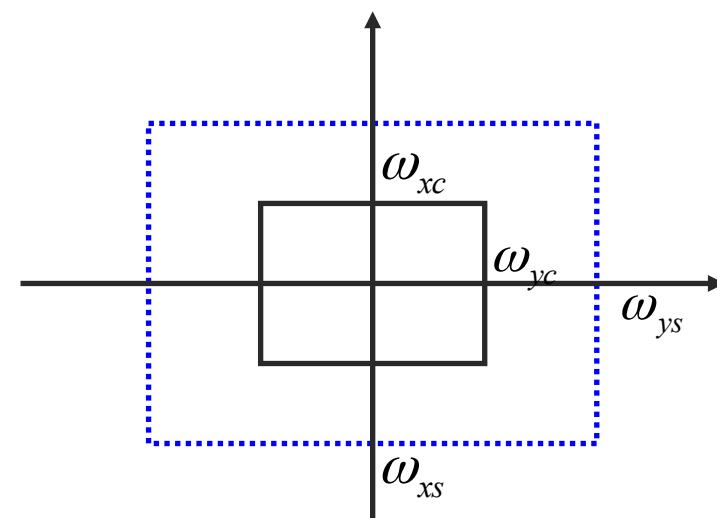
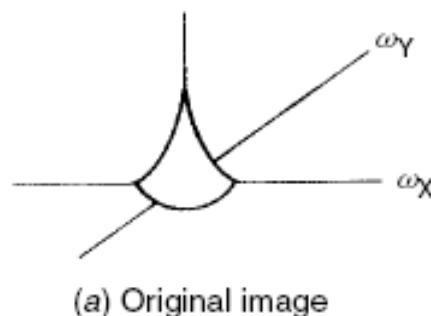
# Image Sampling & Transforms

## ■ Image sampling

### ○ Transform domain (Fourier transform)

- Assume the spectrum of the ideal image is limited and suppose that the cutoff frequency is  $(\omega_{xc}, \omega_{yc})$   
i.e.  $\mathcal{F}_P(\omega_x, \omega_y)$  is non-zero only in the region bounded

by  $|\omega_x| \leq \omega_{xc}$   $|\omega_y| \leq \omega_{yc}$



# [ Image Sampling & Transforms ]

## ■ Image sampling

### ○ Transform domain (Fourier transform)

$$\text{■ If } \omega_{xc} \leq \frac{\omega_{xs}}{2}, \quad \omega_{yc} \leq \frac{\omega_{ys}}{2} \quad ,$$

there is **no overlapping (no aliasing)** between adjacent shifted waveforms of  $\mathcal{F}_P(\omega_x, \omega_y)$

$\omega_{xs}, \omega_{ys}$  : the sampling frequency  
 $\omega_{xc}, \omega_{yc}$  : the cutoff frequency

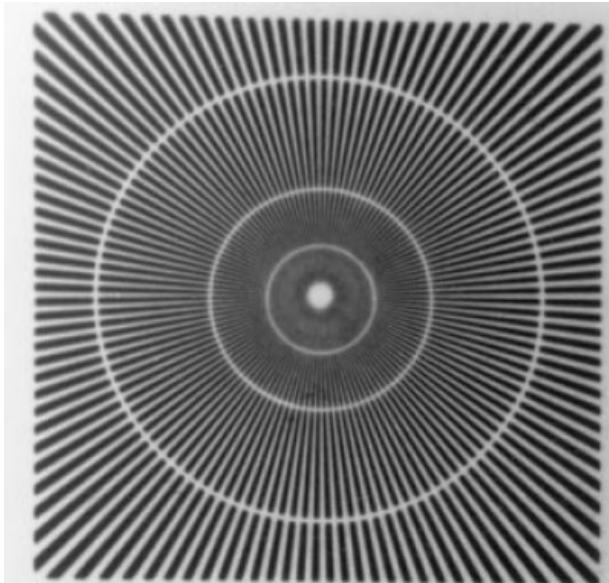
### ○ Sampling Theorem

- To sample a band-limited signal, the sampling period must be **no longer** than one-half of the period of the finest details within the image to avoid aliasing
- Generalization of the 1D **Nyquist Theorem** to the 2D case

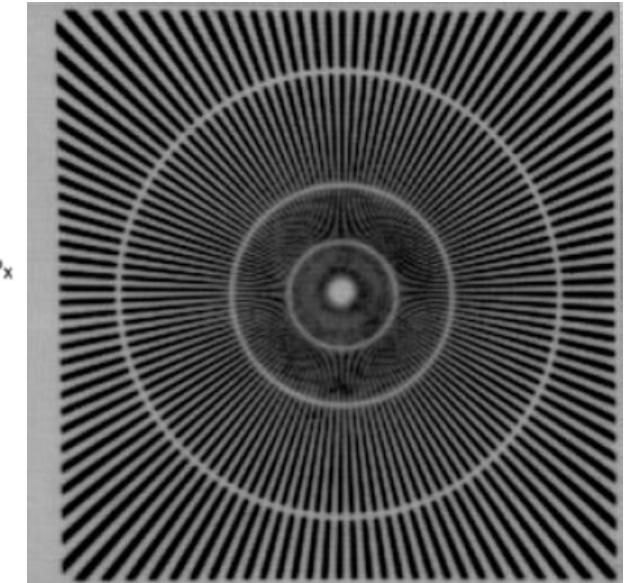
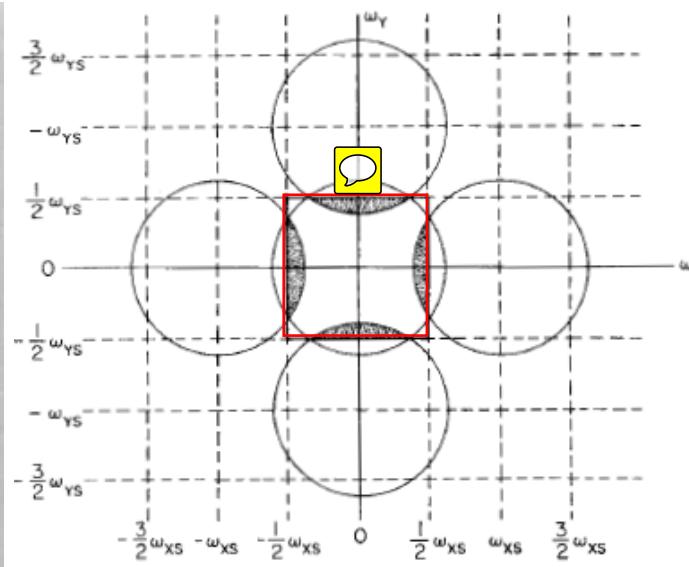
# [Image Sampling & Transforms]

## ■ Sampling Theorem

- To sample a **band-limited** signal, the **sampling period** must be **no longer** than one-half of the period of the finest details within the image to avoid **aliasing**



Original image



Sampled image

# Image Sampling & Transforms

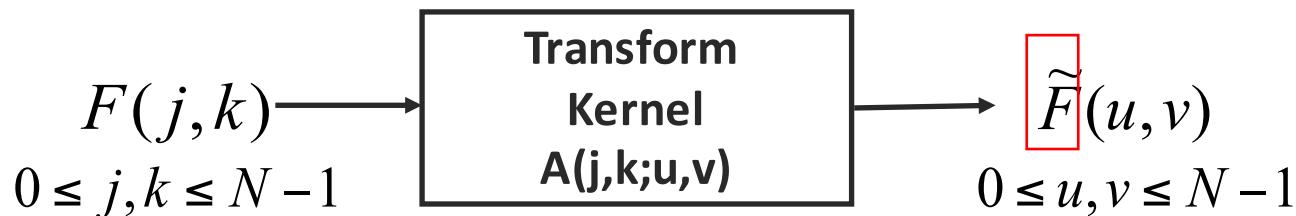
## ■ Image Transform

- Why image transform?
  - Feature extraction
  - Energy compaction
- We are able to concentrate energy distribution over a small number of transform coefficients
  - Can be exploited for compression purpose
    - DCT – Discrete Cosine Transform
    - Block-based transform – 8x8

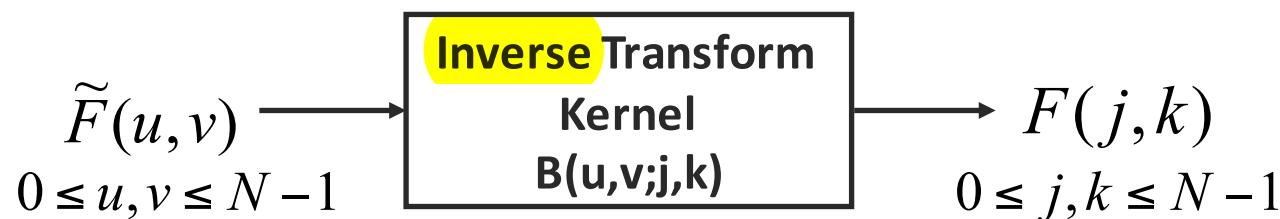
# [Image Sampling & Transforms]

## ■ General 2D transform

- Forward



- Backward



# Image Sampling & Transforms

## ■ 2D **separable** transform

### ○ General 2D transform

#### ■ Forward

$$\tilde{F}(u, v) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} F(j, k) A(j, k; u, v)$$

#### ■ Backward

$$F(j, k) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \tilde{F}(u, v) B(u, v; j, k)$$

#### ■ If A and B are **separable**, we have

$$A(j, k; u, v) = A_X(j, u) A_Y(k, v)$$

$$B(u, v; j, k) = \underline{B_X(u, j)} \underline{B_Y(v, k)}$$

**x-direction**   **y-direction**

# [ Image Sampling & Transforms ]

## ■ 2D separable transform

- The real advantage is the low computational cost of the separable transform

### ■ General 2D transform

$$\tilde{F}(u, v) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} F(j, k) A(j, k; u, v)$$



$O(N^2)$  per  $(u, v)$  &  $O(N^4)$  for all  $(u, v)$

### ■ Separable 2D transform

$$\tilde{F}(u, v) = \sum_{k=0}^{N-1} \left( \sum_{j=0}^{N-1} F(j, k) A_X(j, u) \right) A_Y(k, v)$$

$2N$  per  $(u, v)$  &  $2N^3$  for all  $(u, v)$

# Image Sampling & Transforms

## Unitary Transform

- Consider 1D transform

$$\tilde{F}(u) = \sum_{j=0}^{N-1} F(j) A(j, u)$$

$$\underline{f} = \begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(N-1) \end{bmatrix} \quad \tilde{\underline{f}} = \begin{bmatrix} \tilde{F}(0) \\ \tilde{F}(1) \\ \vdots \\ \tilde{F}(N-1) \end{bmatrix} \quad A = \begin{bmatrix} A(0,0) & \cdots & A(N-1,0) \\ \vdots & \ddots & \vdots \\ A(0,N-1) & \cdots & A(N-1,N-1) \end{bmatrix}$$

- Forward  $\tilde{\underline{f}} = A\underline{f} \Rightarrow B = A^{-1}$
- Backward  $\underline{f} = B\tilde{\underline{f}}$  Gaussian elimination  $\Rightarrow O(N^3)$

# [Image Sampling & Transforms]

## ■ Unitary Transform

- The inverse is easy to compute 

$$A^H A = AA^H = I \Rightarrow A^{-1} = A^H$$

$$A^H = (A^*)^T = (A^T)^*$$

## ■ Karhunen Loeve Transform (KLT)

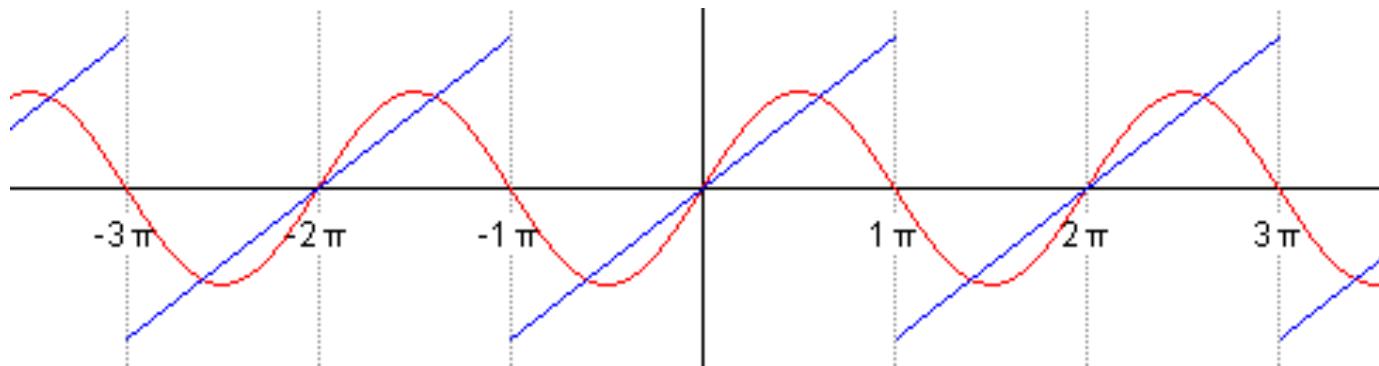
- Linear transform 
- Ideal for energy compaction
- Principal component analysis (PCA)/Singular Value Decomposition (SVD)
- Basis functions  are image dependent
- High computation cost for obtaining basis images

# Image Sampling & Transforms

## ■ Discrete Fourier Transform (DFT)

- Idea

- Any function that periodically repeats itself can be expressed as a sum of sine and cosine of different frequencies.



# Image Sampling & Transforms

## ■ Discrete Fourier Transform (DFT)

- Transform

$$F(u, v) = \frac{1}{n} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j, k) \exp\left\{-\frac{2\pi i(uj + vk)}{n}\right\}$$

- Inverse DFT

$$f(j, k) = \frac{1}{n} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} F(u, v) \exp\left\{\frac{2\pi i(uj + vk)}{n}\right\}$$

$i = \sqrt{-1}$  and  $f(j, k)$  is the input sequence

# Image Sampling & Transforms

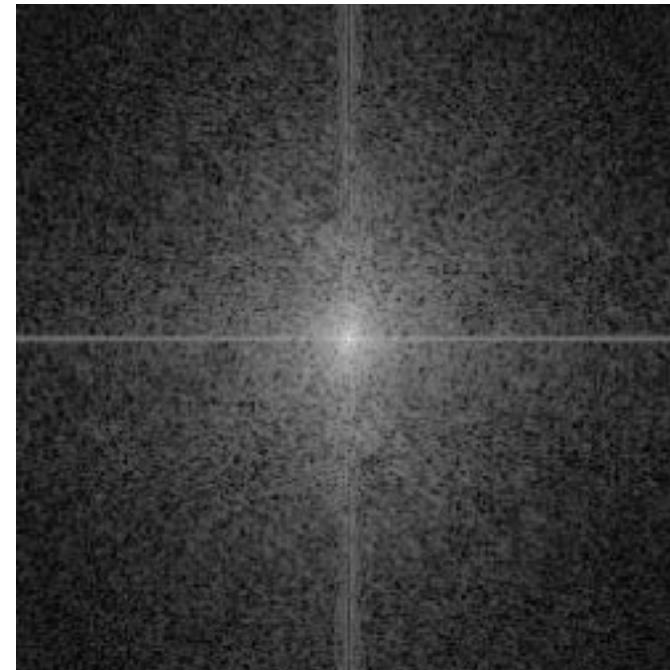
- Discrete Fourier Transform (DFT)
  - Fast Fourier Transform (FFT)
    - $O(n \log n)$
  - Not so popular in image compression
    - performance is not good enough
    - computational cost for complex number is expensive

# Image Sampling & Transforms

- Discrete Fourier Transform (DFT)
  - Example



original



log magnitude

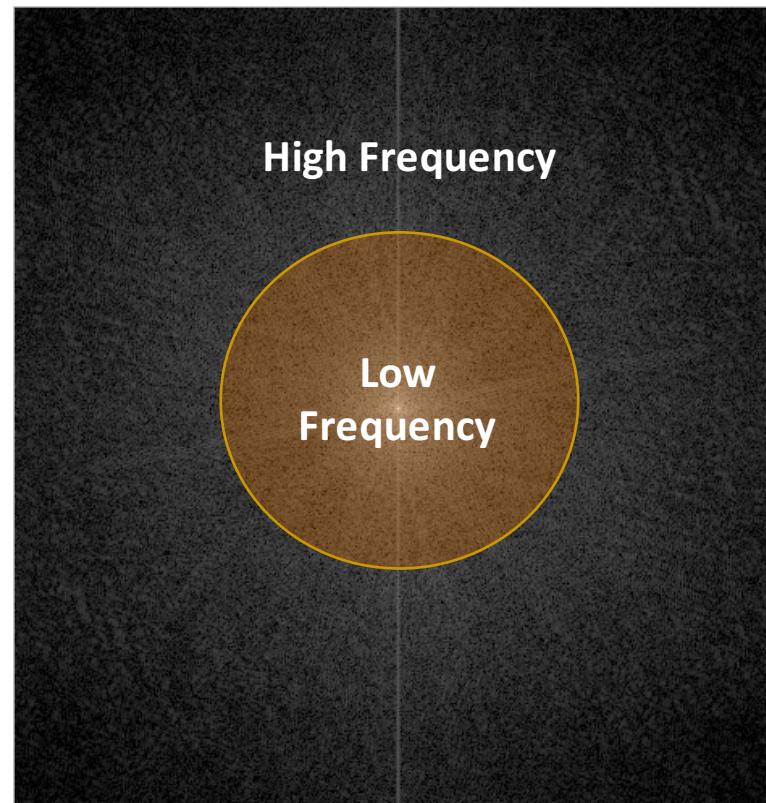
# Image Sampling & Transforms

- Discrete Fourier Transform (DFT)
  - Application – Noise reduction



# Image Sampling & Transforms

- Discrete Fourier Transform (DFT)
  - Application– Low Pass Filter



# Image Sampling & Transforms

- Discrete Fourier Transform (DFT)
  - Application– Low Pass Filter



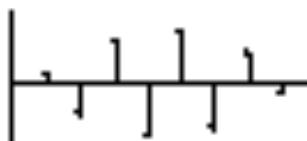
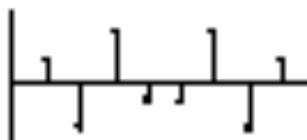
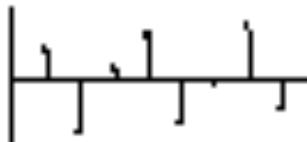
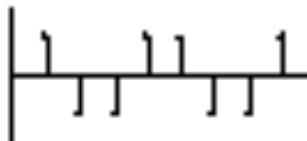
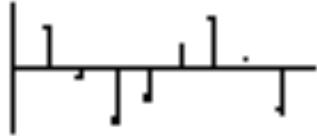
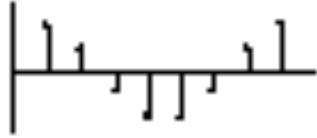
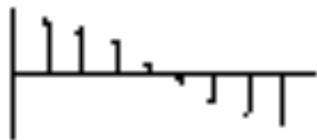
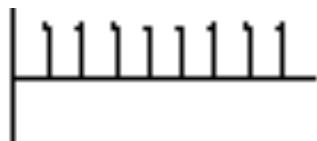
Original image



After low pass filter

# Image Sampling & Transforms

- Discrete Cosine Transform (DCT)
  - 1D DCT basis functions



All basis functions are cosine functions with different frequencies except the first basis function.

The signal is linear combination by these 8 basis functions. The result is the coefficient of linear combinations.

$n=8$

# Image Sampling & Transforms

## ■ Discrete Cosine Transform (DCT)

- **Transform**  $f(j,k) : j \text{ is row, } k \text{ is column}$

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

- **Inverse DCT**

$$f(j,k) = \sum_{v=0}^{n-1} \sum_{u=0}^{n-1} C(u)C(v) F(u,v) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

$$C(w) = \begin{cases} 1/\sqrt{2} & \text{if } w = 0 \\ 1 & \text{if } w = 1, 2, \dots, n-1 \end{cases}$$

# Image Sampling & Transforms

## Discrete Cosine Transform (DCT)

### Transform

$F(0,0)$  is to compute DC, the mean of the image

$$F(u,v) = \frac{c(u)c(v)}{4} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

$$F_{10} = \frac{c_1 c_0}{4} \left[ \cos \frac{\pi}{16} \left( \sum_{i=0}^7 f(0,i) - \sum_{i=0}^7 f(7,i) \right) + \cos \frac{3\pi}{16} \left( \sum_{i=0}^7 f(1,i) - \sum_{i=0}^7 f(6,i) \right) + \right.$$

$$\left. \cos \frac{5\pi}{16} \left( \sum_{i=0}^7 f(2,i) - \sum_{i=0}^7 f(5,i) \right) + \cos \frac{7\pi}{16} \left( \sum_{i=0}^7 f(3,i) - \sum_{i=0}^7 f(4,i) \right) \right]$$

$F_{01} = F(0,1)$

$F_{02} = F(0,2)$

$F_{10} = F(1,0)$

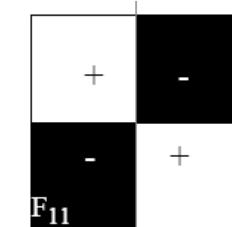
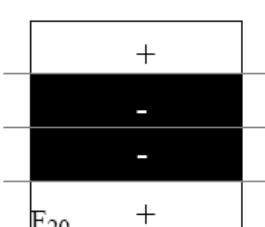
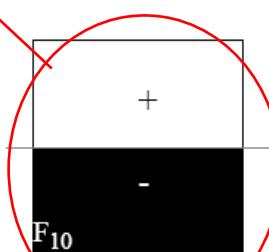
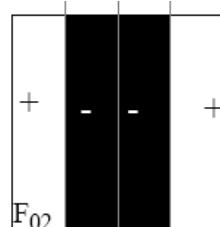
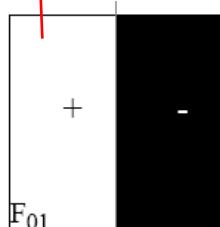
there are

8x8  $F(u,v)$

for 8\*8

basis

functions

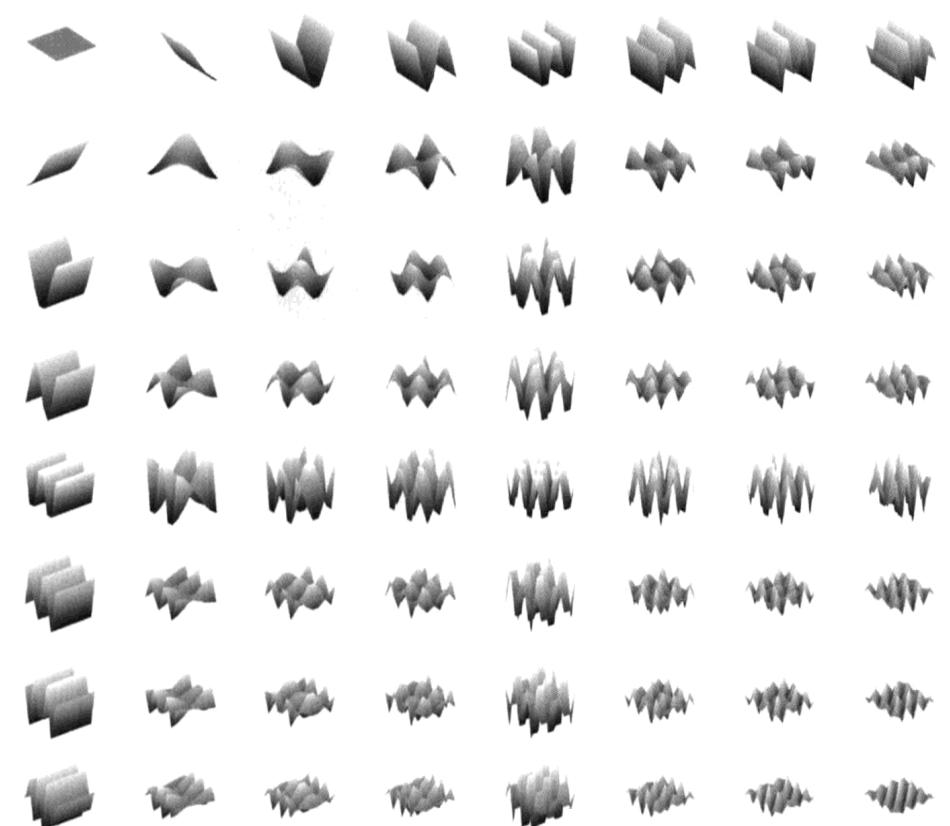
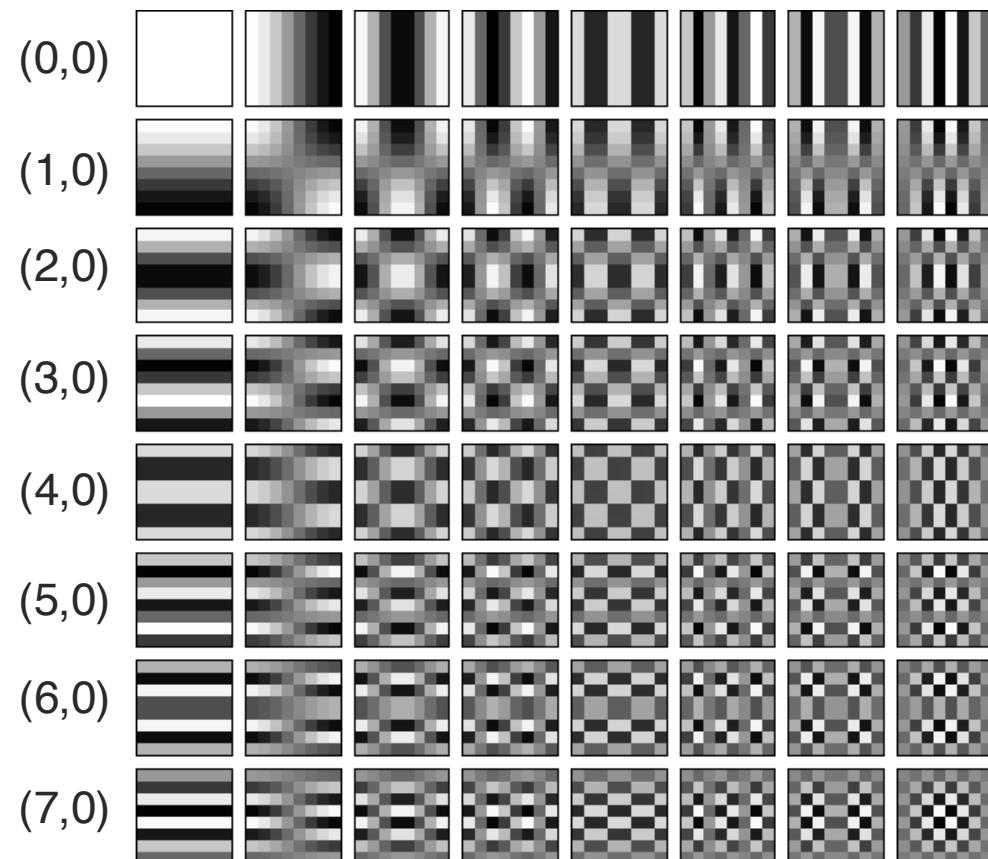


# Image Sampling & Transforms

## ■ 2D DCT basis functions (8x8)

(0,0) for DC value  
the other are AC

(0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 7)



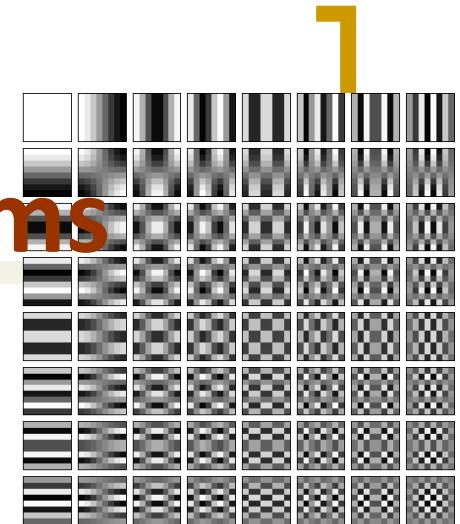
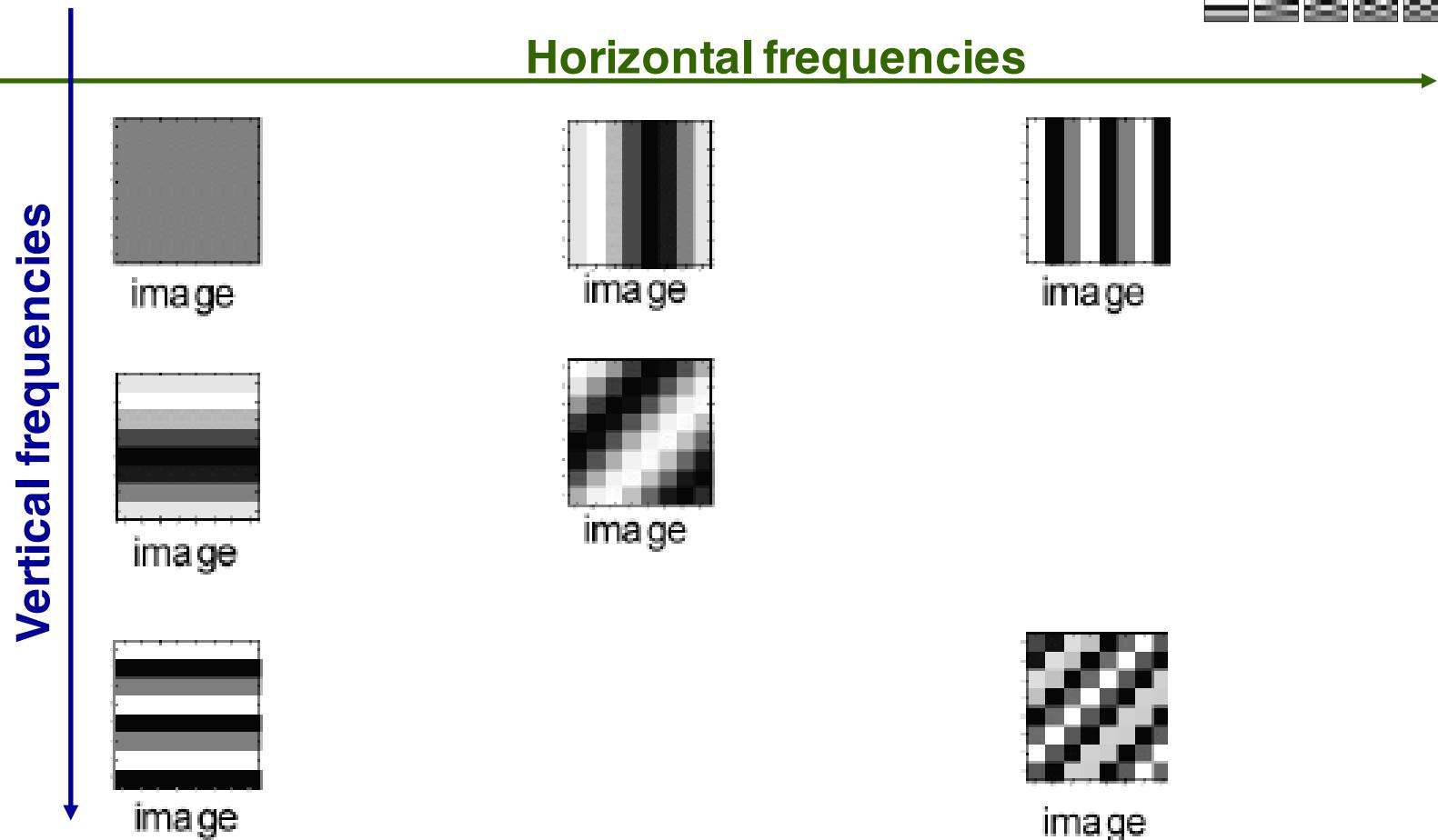
# [Image Sampling & Transforms]

## ■ The Meaning of DCT Coefficients

- Represent the spatial frequency content within an 8x8 image block
- The (0,0) coefficient is called DC coefficient
  - The average of the 64 image pixel values in the block
- The rest of 63 coefficients are called AC coefficients
- Move to the right of the block
  - the energy in higher horizontal frequencies
- Move down the block
  - the energy in higher vertical frequencies

# Image Sampling & Transforms

## ■ Examples of 8x8 DCT Coefficients



# Image Sampling & Transforms

To operate on the compressed image directly without decompression and compression again.  
It save time very much, but sometimes it loses the detail because the limited basis functions.

## ■ Discrete Cosine Transform (DCT)

Vertical-dominant	section 1	$ F_{01}  >  F_{10} $	$F_{01} \leq 0$	$F_{10} > 0$		
				$F_{10} < 0$		
	section 2		$F_{01} \geq 0$	$F_{10} > 0$		
				$F_{10} < 0$		
Horizontal-dominant	section 3	$ F_{01}  <  F_{10} $	$F_{10} \leq 0$	$F_{01} > 0$		
				$F_{01} < 0$		
	section 4		$F_{10} \geq 0$	$F_{01} > 0$		
				$F_{01} < 0$		
Diagonal		$ F_{01}  =  F_{10} $	$F_{01} > 0$	$F_{10} > 0$		
				$F_{10} < 0$		
			$F_{01} < 0$	$F_{10} > 0$		
				$F_{10} < 0$		

# Image Sampling & Transforms

- Discrete Cosine Transform (DCT)
  - Use cosine function as its basis function
  - Performance similar to KLT
  - Fast algorithm available
  - Most popular in image compression
  - JPEG (Joint Photographic Experts Group)
  - The periodicity implied by DCT implies that less blocking artifact will be introduced than DFT

# Image Sampling & Transforms

## Hadamard transform

- Based on Hadamard matrix

- A square array with elements of plus and minus 1's
- Rows and columns are orthogonal
- A normalized Hadamard matrix satisfies  $HH^T = I$
- The general form:

$$H_{2N} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Sign Changes

0
3
1
2

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Sign Changes

0
7
3
4
1
6
2
5
31

# Image Sampling & Transforms

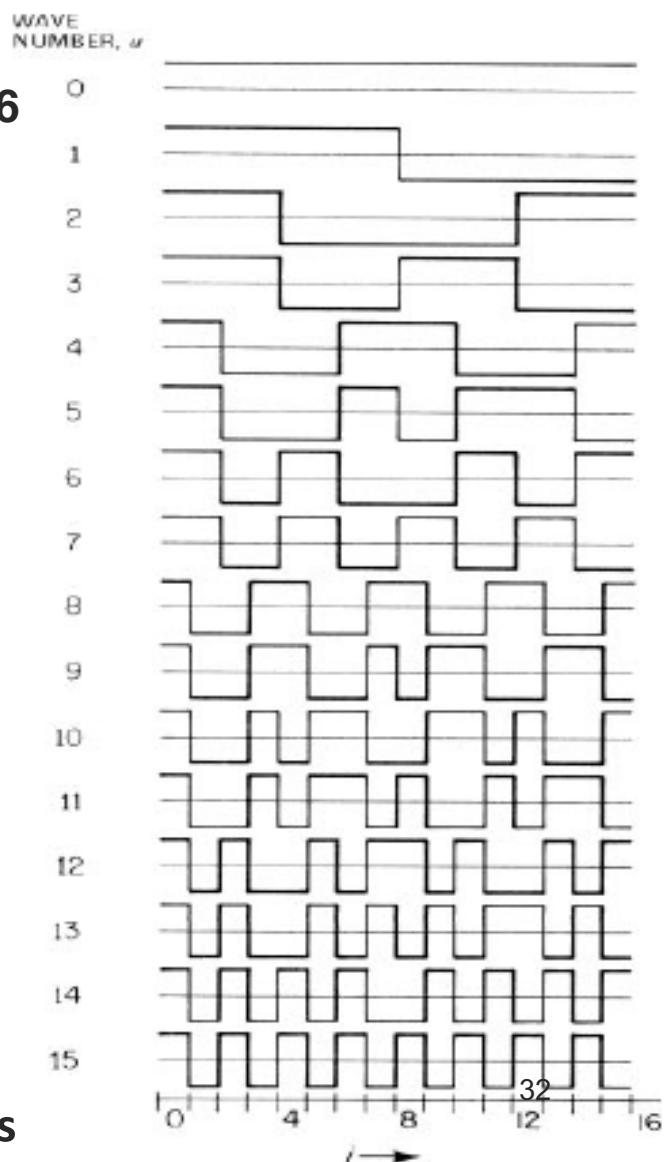
## Hadamard transform

e.g. N=8

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & | & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & | & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & | & 1 & 1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 & | & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & | & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & | & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & | & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & | & -1 & 1 & 1 & -1 \end{bmatrix}$$

Sign Changes
0
7
3
4
1
6
2
5

e.g. N=16



## Applications

- data encryption and **signal processing**
- data compression algorithms
  - HD Photo and MPEG-4 AVC
  - video compression
  - the sum of absolute transformed differences

# Image Sampling & Transforms

## ■ Haar transform

- Derived from the Haar matrix

$$R_N = \begin{bmatrix} V_N \\ W_N \end{bmatrix}_{N \times N}$$

$$V_N = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix}$$

Scaling matrix

$$W_N = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & & & & & & \ddots & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix}$$

Wavelet matrix

e.g.

$$H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

$$H_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & -2 \end{bmatrix}$$

# Image Sampling & Transforms

## ■ Haar transform

- 1st-level Haar transform of a Nx1 vector  $f$

$$f_1 = R_N f = [a_1 \mid d_1]^T \quad \text{where } \begin{matrix} a_1 = V_N f; \\ \text{trend} \end{matrix} \quad \begin{matrix} d_1 = W_N f \\ \text{fluctuation} \end{matrix}$$

global trend    local fluctuation

- 2nd-level Haar transform of a Nx1 vector  $f$

$$f_2 = [a_2 \mid d_2 \mid d_1]^T \quad \text{where } a_2 = V_{N/2} a_1; \quad d_2 = W_{N/2} a_1$$

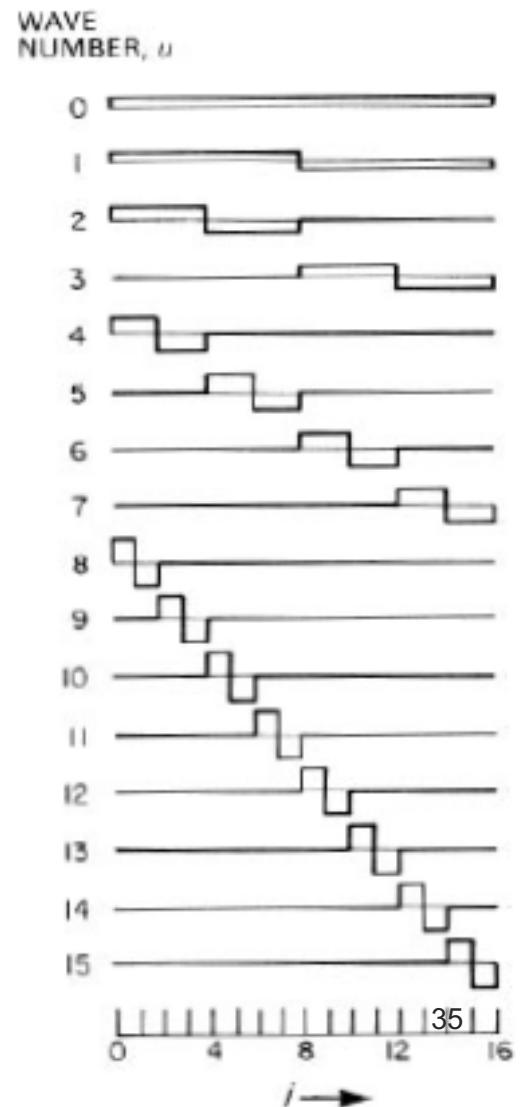
- The process continues until the full transformation

$$\Rightarrow f_n = [a_n \mid d_n \mid d_{n-1} \mid \cdots \mid d_1]^T$$

# Image Sampling & Transforms

## ■ Haar transform wavelet transform

- E.g. basis functions with N=16
- Sample an input data sequence with finer and finer resolution **increasing at the power of 2**
- Provide a transform domain in which type of differential energy is **concentrated in localized regions**

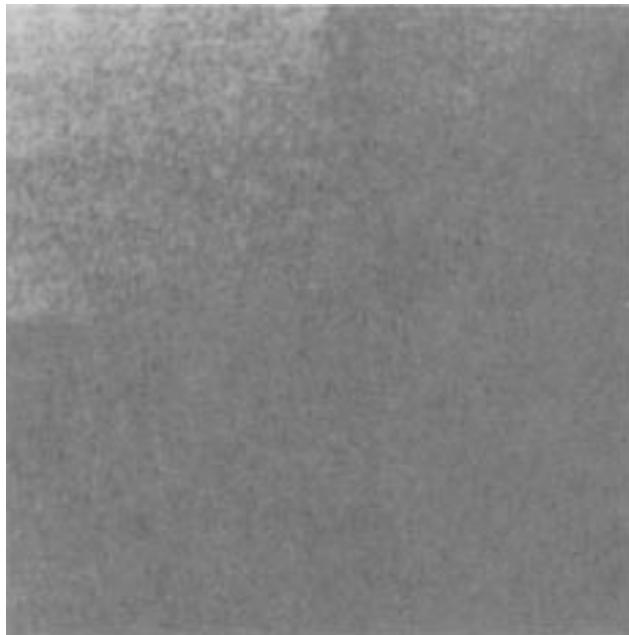


# Image Sampling & Transforms

## ■ Hadamard transform and Haar transform

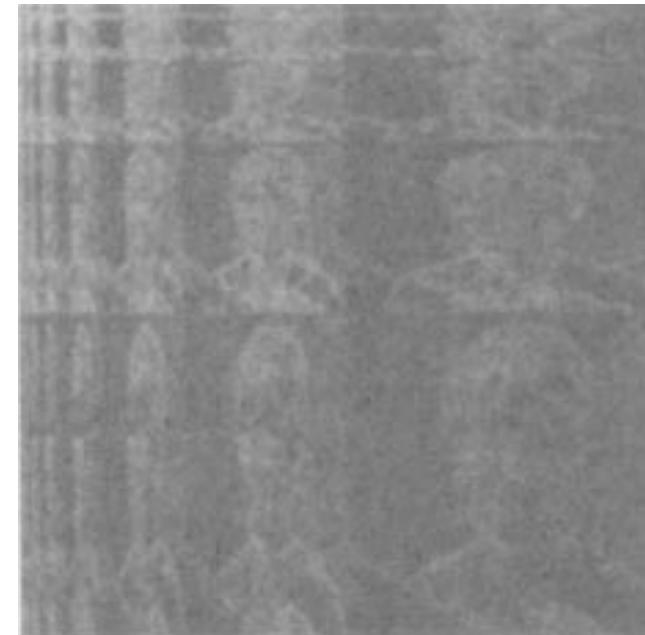
### ○ Example

no spatial domain can be seen



Hadamard transform

sptial domain is still kept

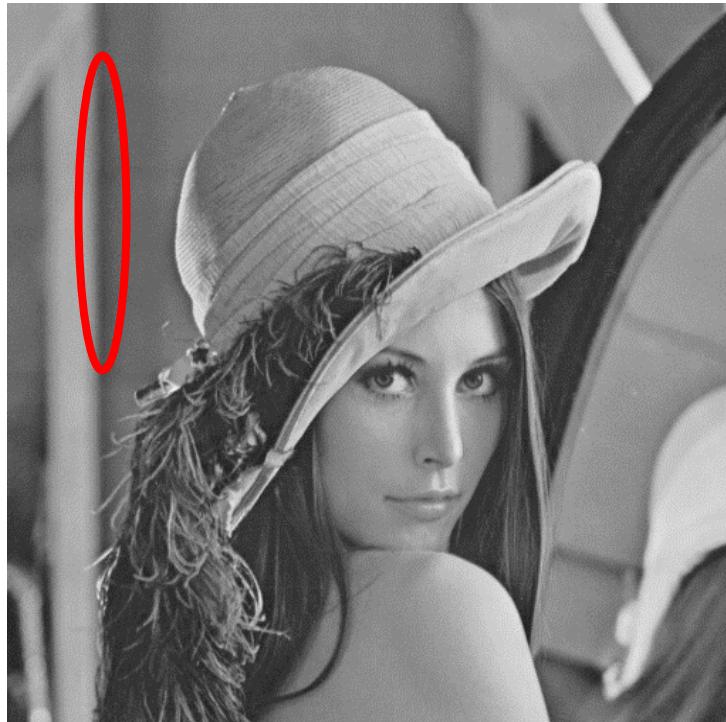


Haar transform

# [Image Sampling & Transforms]

## ■ Hough transform

- An edge is not a line...
- How to detect lines in an image?



# [Image Sampling & Transforms]

## ■ Hough transform

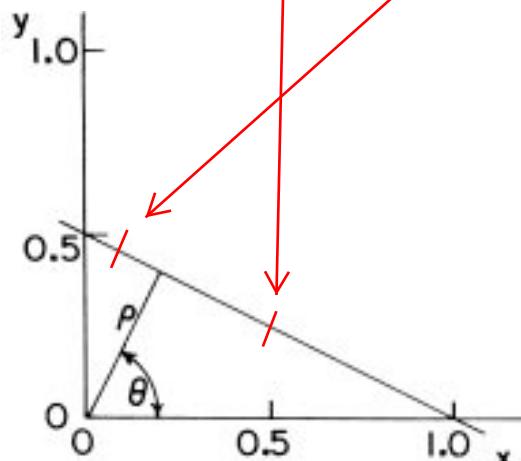
- How to detect lines in an image?
  - Search for the lines at every position with possible orientations
  - Voting scheme: Hough transform
- Performed after edge detection
- Can locate straight lines, circles, parabolas, ellipses, etc
  - As the curve can be specified in a parametric form
- Advantages
  - Tolerate gaps between edges
  - Relatively unaffected by noise
  - Can deal with occlusion

# Image Sampling & Transforms

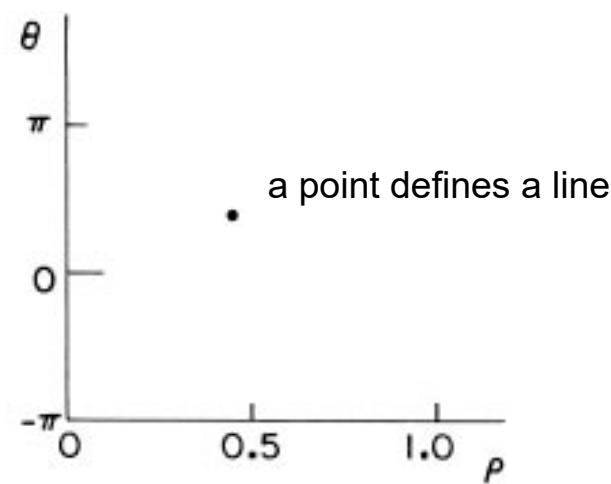
## Hough transform

- A straight line can be represented as  $y = mx + b$ 
  - Vertical lines?  $m$  is infinity, it's not a function of  $y$
- Another representation is

$$\rho = x\cos\theta + y\sin\theta, \quad -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$$



(a) Parametric line

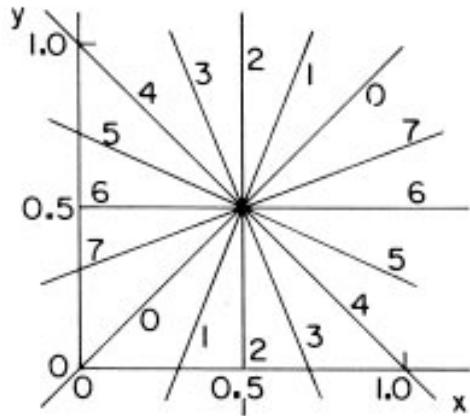


(b) Hough transform

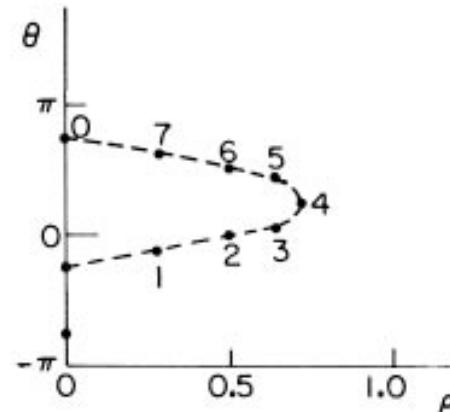
# Image Sampling & Transforms

## Hough transform

(c) Family of lines,  
common point

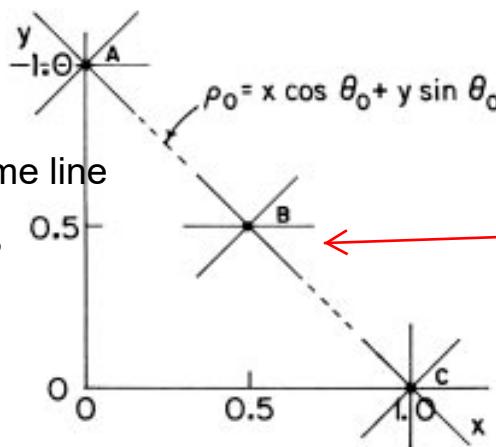


(d) Hough  
transform of (c)

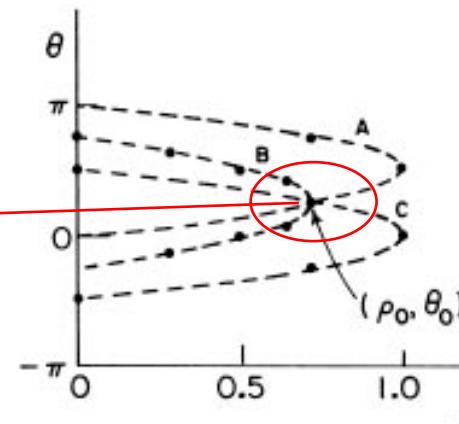


colinear : points are in the same line

(e) Colinear points



(f) Hough  
transform of (e)



all the transform intersects the same point which means they are in the same line.

# [Image Sampling & Transforms]

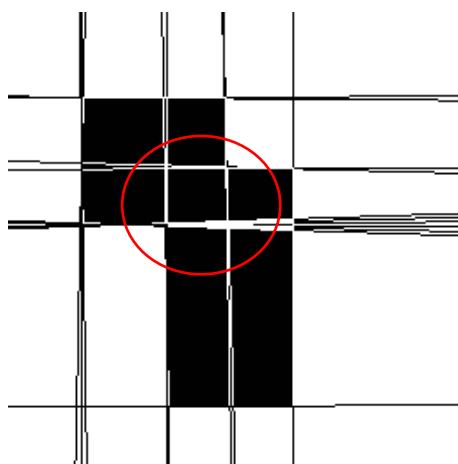
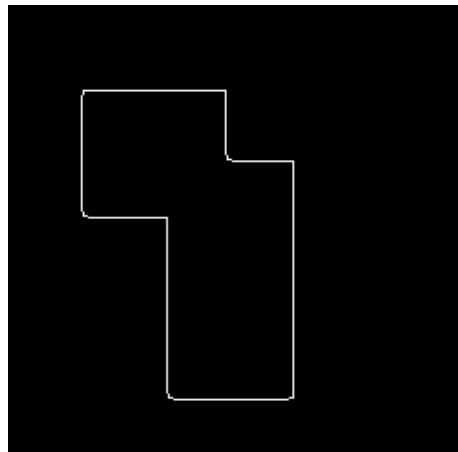
## ■ Hough transform for line detection

- Quantize the Hough Transform space
  - Identify the maximum and minimum values of  $\rho$  and  $\theta$
- Generate an accumulator array  $A(\rho, \theta)$
- For all edge points  $(x_i, y_i)$  in the image
  - Use gradient direction as  $\theta$
  - Compute  $\rho$  from the equation
  - Increment  $A(\rho, \theta)$  by one
- For all cells in  $A(\rho, \theta)$ 
  - Search for the maximum value
  - Calculate the equation of the line

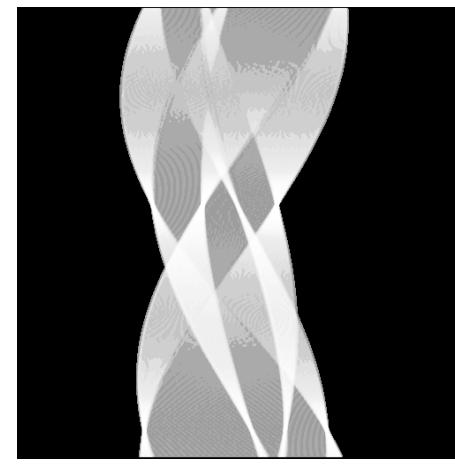
The point in the Hough space is a line, so when the max value is found in A, the  $(r, \theta)$  is the detected line.

# Image Sampling & Transforms

## ■ Hough transform for line detection



Take a relative  
threshold  
and convert  
back to  
image space



H.E.  
histogram equalization