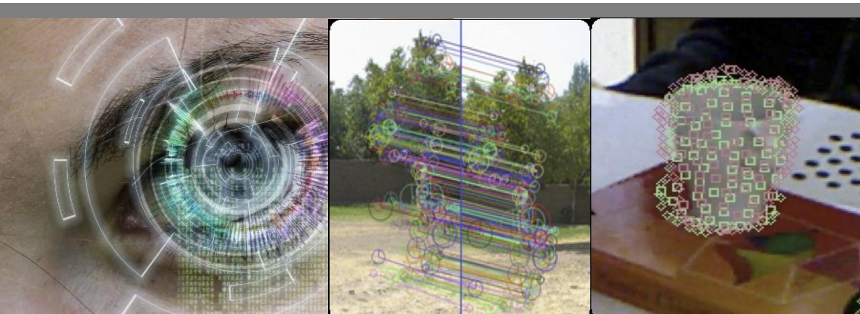
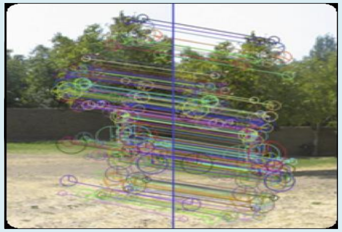
Advanced Multimedia Signal Processing
(#3: Signal Transform/Image Model)

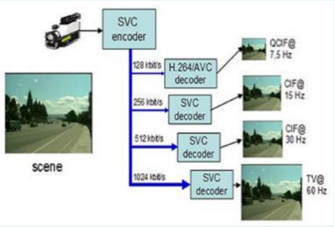


2017. Spring

Prof. Byung-Gyu Kim

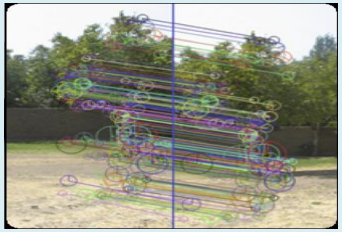
Dept. of IT Engineering, Soomyung Women's University

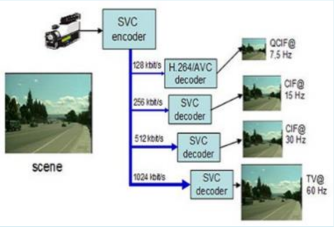




Contents

- Example of Huffman Code
- Signal Transform
- Basic Concept of Image Coding





Contents

- Example of Huffman Code
- Signal Transform
- Basic Concept of Image Coding

❖ Symbol Coding: Huffman Code (Variable Length Code)

- Based on frequency of the symbols (coefficients).
  - High frequency symbols: smaller bits ASAP.
  - Low frequency symbols : larger bits ASAP.
  - Variable (bit) length symbol coding.

▪ Example of Huffman Coding

If we have symbols with probabilities:

a1 (0.1), a2 (0.4), a3 (0.06), a4 (0.1), a5 (0.04), a6 (0.3)

- Ordering the probabilities of the symbols at first.

Huffman Coding (Example) (2)

Original source	
Symbol	Probability
a2	0.4
a6	0.3
a1	0.1
a4	0.1
a3	0.06
a5	0.04

- Create a series of source reductions by ordering the probabilities of the symbols under consideration and combining the lowest probability symbols into single symbol that replace them in the next source reduction. This process is then repeated until a reduced source with two symbols is reached.

Huffman Coding (Example) (3)

Original source		Source Reduction			
Symbol	Probability	1	2	3	4
a2	0.4	0.4	0.4	0.4	0.6
a6	0.3	0.3	0.3	0.3	0.4
a1	0.1	0.1	0.2	0.3	
a4	0.1	0.1	0.1		
a3	0.06	0.1			
a5	0.04				

- To code each reduced source, starting with smallest source and working back to the original source.

VICL
Visual Information Computing Lab

Huffman Coding (Example) (4)

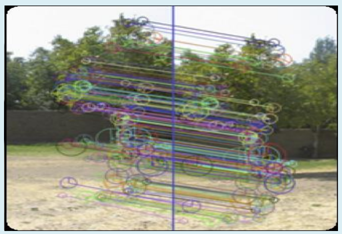
Original source			Source Reduction							
Symbol	Probability		1		2		3		4	
a2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a1	0.1	011	0.1	011	0.2	010	0.3	01		
a4	0.1	0100	0.1	0100	0.1	011				
a3	0.06	01010	0.1	0101						
a5	0.04	01011								

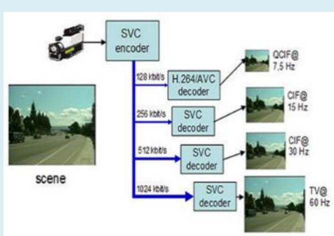
- The final code is appeared at the far left of the table.

- The average length of this code is,

$$L_{avg} = (0.4)(1)+(0.3)(2)+(0.1)(3)+(0.1)(4)+(0.04)(5)+(0.04)(5)$$
$$= 2.2 \text{ bits/pixel}$$

Huffman's procedure creates the optimal code for a set of symbols and probabilities *subject to the constraint* that the symbols be coded one at a time.



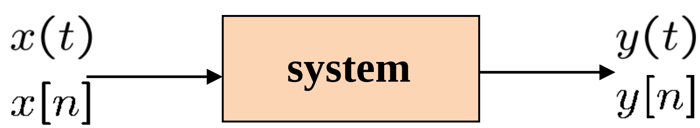


Contents

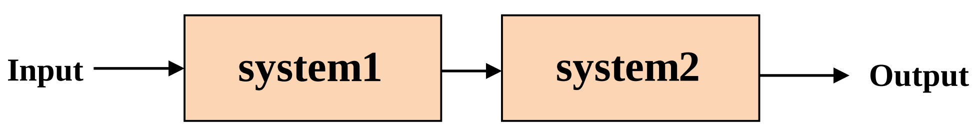
- Example of Huffman Code
- Signal Transform
- Basic Concept of Image Coding

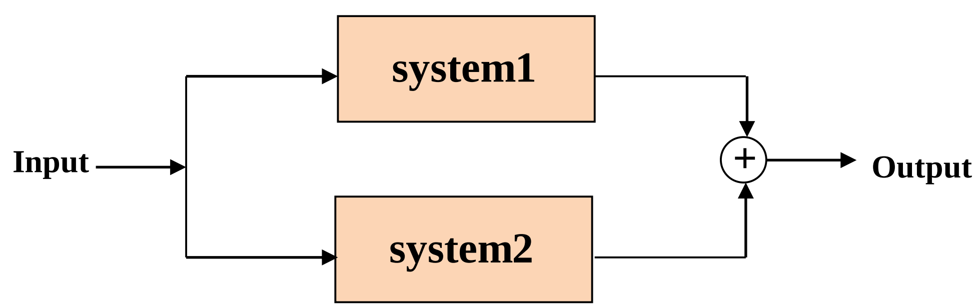
Discrete-time Signal Processing (1): System

- What is <u>system</u>?
  - Any process that results in the transformation of signals.
  - Input signals, output signals, transformation process.
- Continuous-time system/Discrete-time system



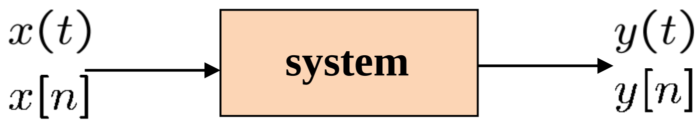
- Interconnectability of systems





Discrete-time Signal Processing (2): Signal Transform

❖ Signal Transform



(signal transform)

- Input : data in **spatial intensity (time) domain**.
- Output: data in **spatial frequency domain**.

❖ Signal Property in Frequency Domain

- Basically, we can observe what frequency components exist in the signal.
- We can analyze the property of frequency of the given signal.

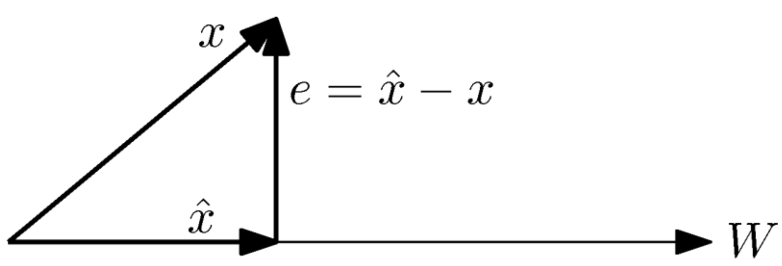
❖ (Linear) Spanning

- the span of *S* may be defined as the set of all finite linear combinations of elements of *S*, which follows from the following definition.

$$\operatorname{span}(S) = \left\{ \sum_{i=1}^{k} \lambda_i v_i \,\middle|\, k \in \mathbb{N}, v_i \in S, \lambda_i \in \mathbf{K} \right\}$$

❖ **Basis function** (vector)

- Linearly independent spanning set
- Each element is not correlated (orthogonal)



❖ Basic structure of Signal Transform

$$G(\theta) = \int f(t)\, d\theta \quad \text{Or}$$

$$G[n] = \sum_k f[k]\ \omega_k$$

where ω_k = the *k*-the basis vector (function).

❖ Concept of Discrete-time Fourier Transform (DFT)

- Fourier Transform (FT)
  - ***Any function that is not periodic*** can be expressed as the integral of sines and/or cosines multiplied by a weighing function

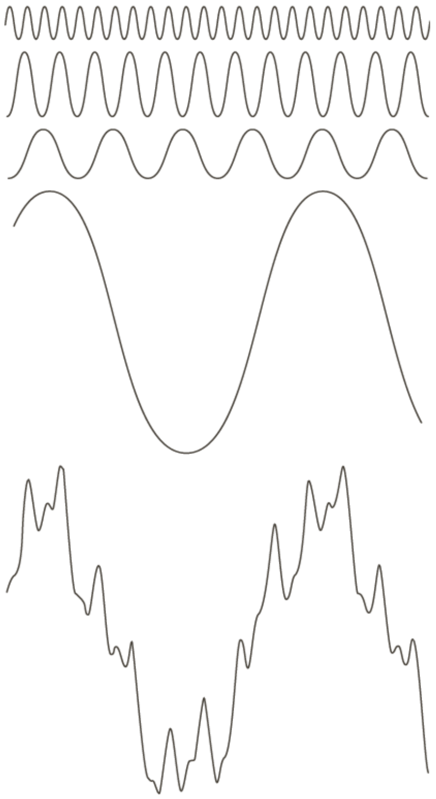
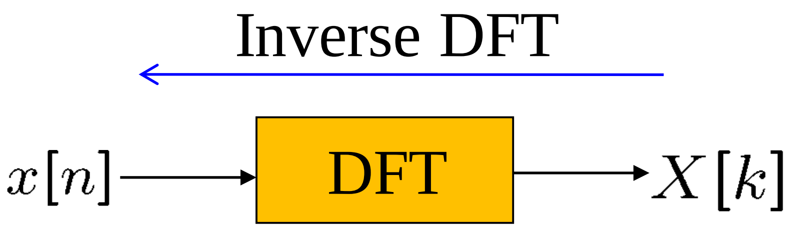


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

- DFT



  - Input signal : $x[n]$ (to be of length N)
  - Output signal: $X[k] = \sum_{n=1}^{N-1} x[n]exp(-\frac{j\pi kn}{N}), \ k = 0, \cdots, N-1.$

- Inverse DFT
  - Input: $X[k]$
  - Output: $x[n]$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]exp(\frac{j2\pi kn}{N}), \ n = 0, \cdots, N-1. \tag{1}$$

VICL Visual Information Computing Lab

❖ DFT

$$j = \sqrt{-1}, \text{ a complex number}$$

$$C = R + jI$$

the conjugate

$$C^* = R - jI$$

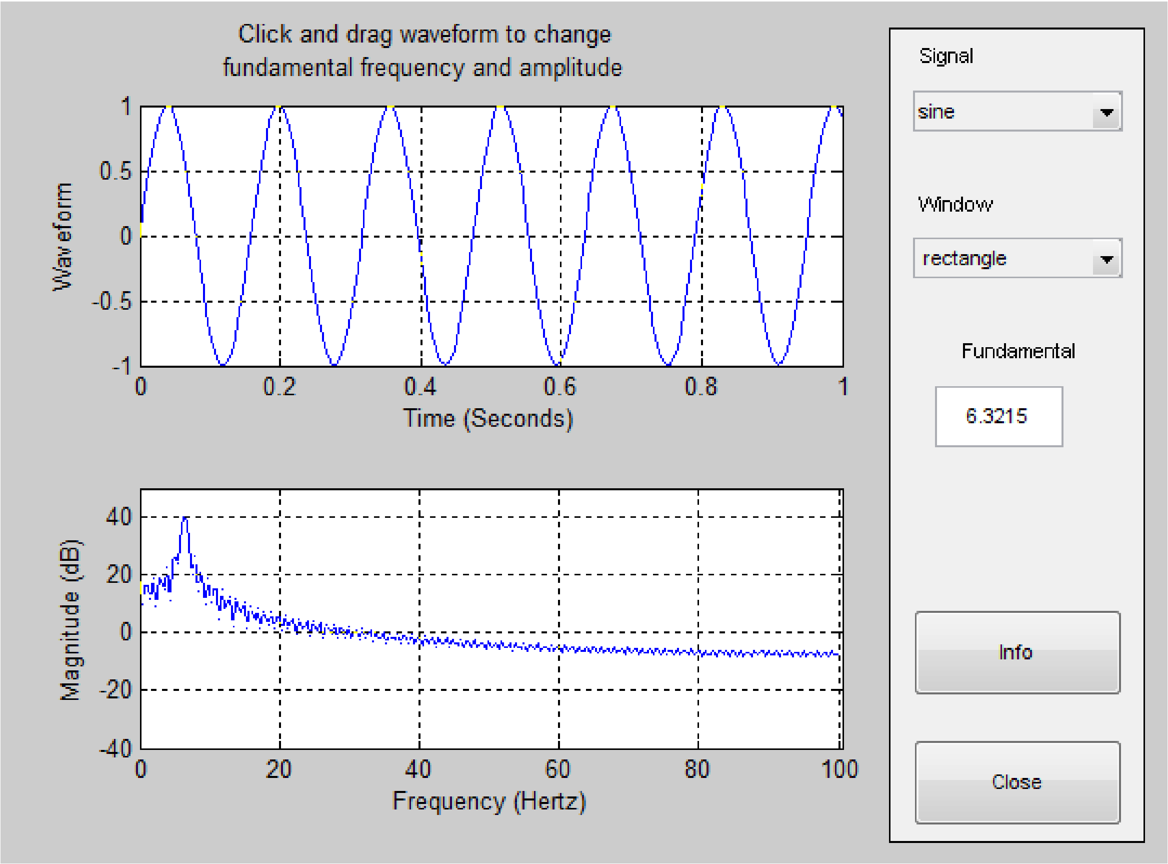
$$|C| = \sqrt{R^2 + I^2} \text{ and } \theta = \arctan(I / R)$$

$$C = |C|(\cos\theta + j\sin\theta)$$

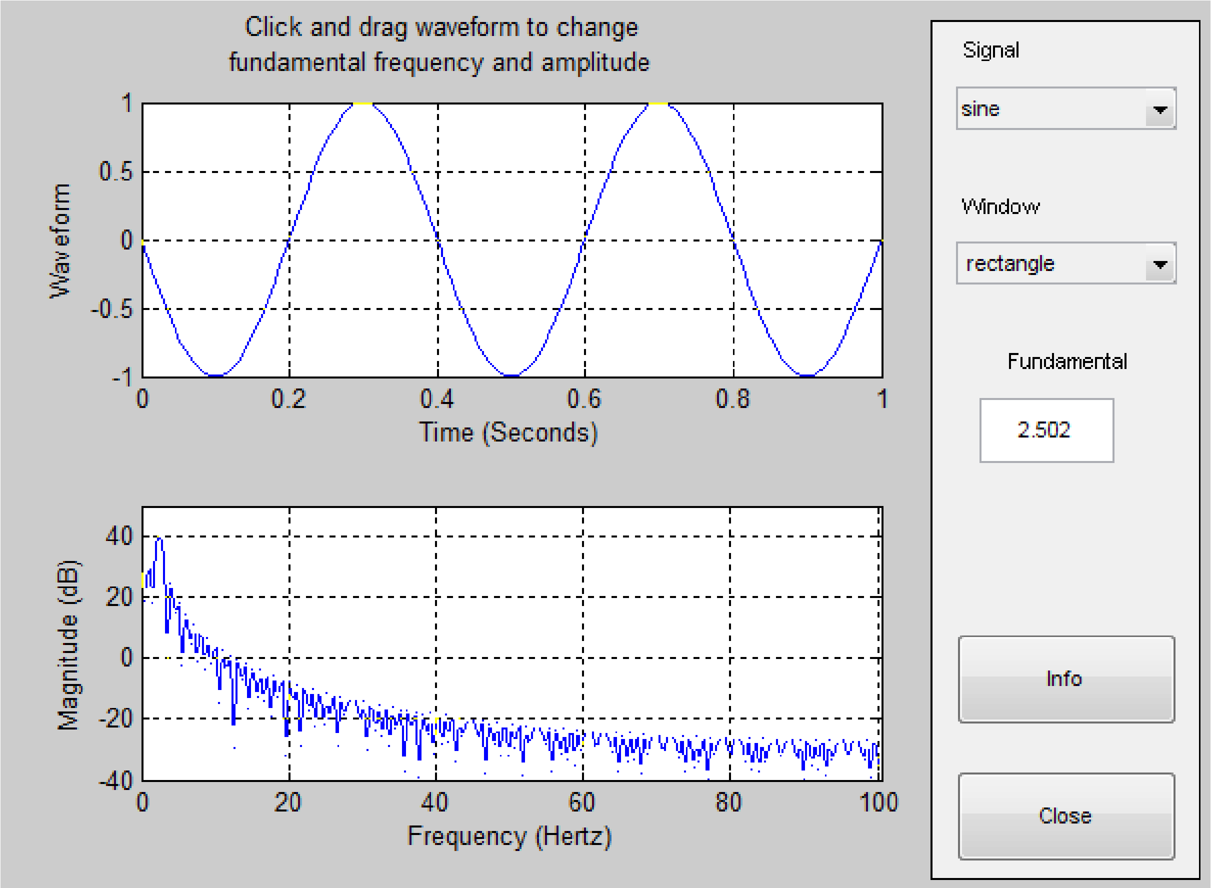
Using Euler's formula,

$$C = |C| e^{j\theta}$$

❖ Implementation of DFT (Ex1) (using Matlab)

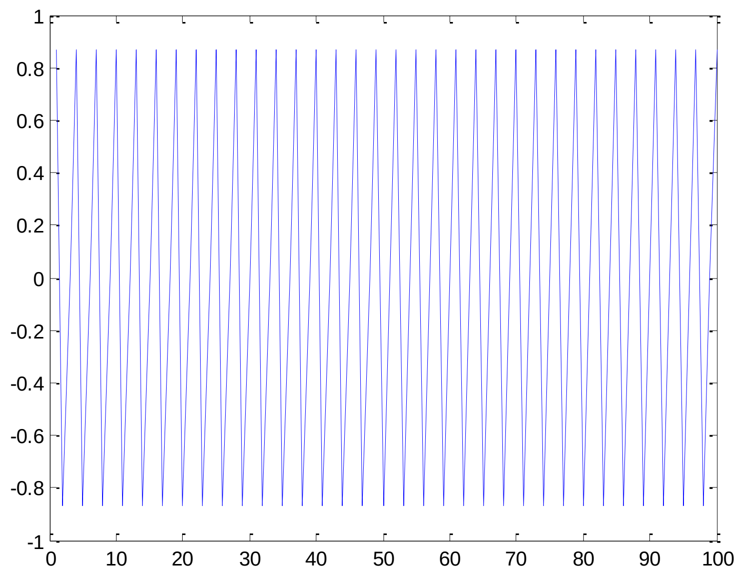


❖ Implementation of DFT (Ex2) (using Matlab)

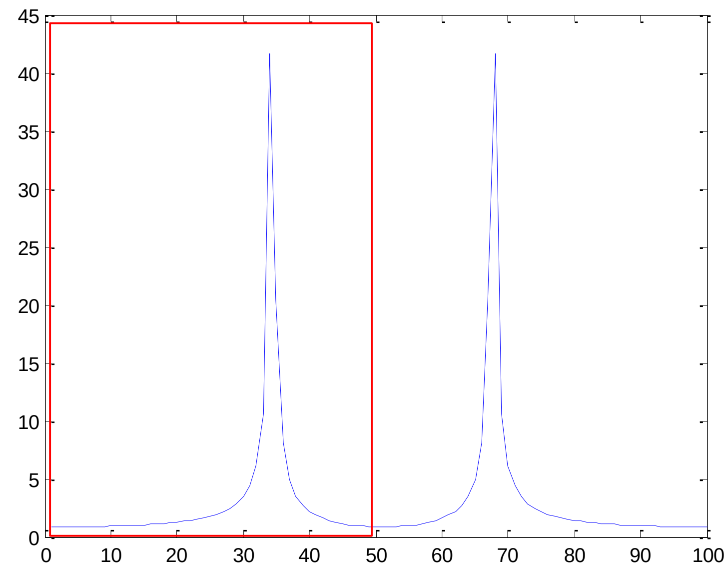


❖ Implementation of DFT (3) (using Matlab)

- X[n] $= \sin(\frac{2\pi f n}{180})$, f=35, n=100.



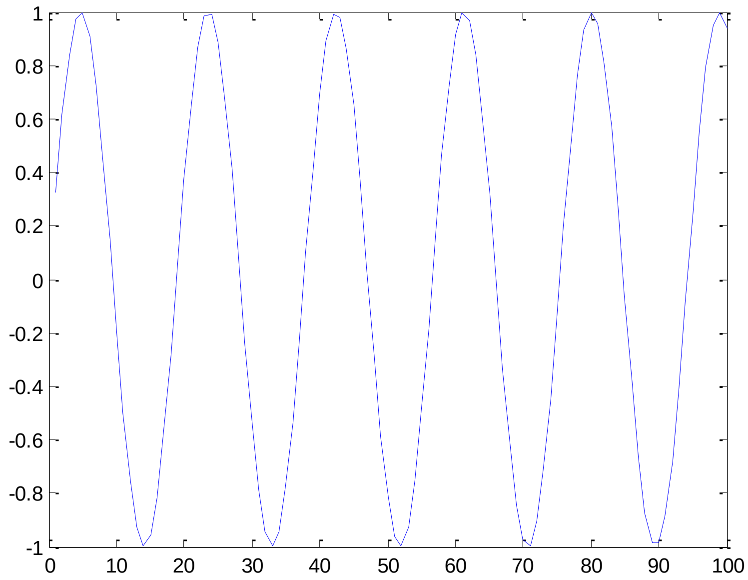
x[n]



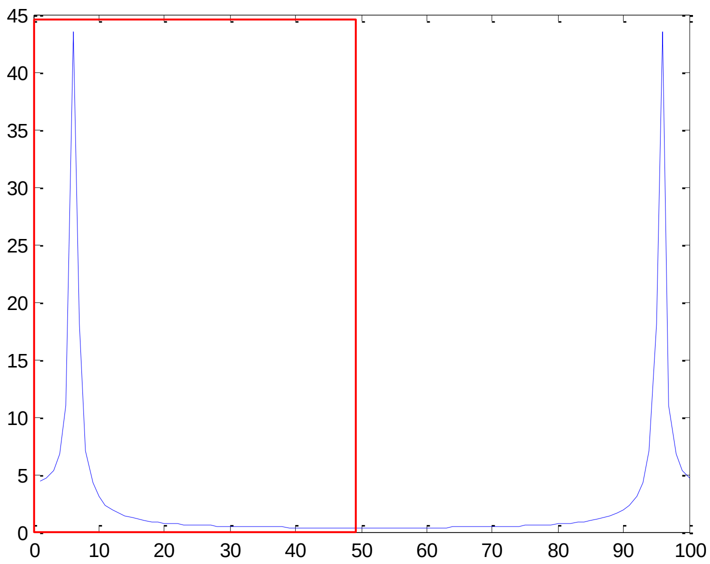
X[k]

❖ Implementation of DFT (Ex4) (using Matlab)

- X[n] = $\sin(\frac{2\pi f n}{180})$, f=6, n=100.



x[n]



X[k]

❖ 2D DFT: Extension of 1D-DFT

- 1-D case: $x[n]$ (length of N)

DFT

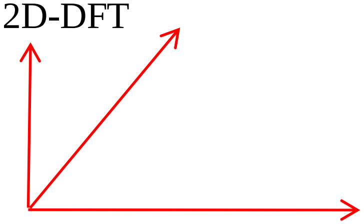
$$X[k] = \sum_{n=1}^{N-1} u[n] exp(-\frac{j\pi kn}{N}), \;\; k = 0, \cdots, N-1.$$

- 2-D case: ***2 axis can be available.....!!!***
  - Input: $x[m,n]$ (size of NxN)
  - Output:

$$X[k,l] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x[m,n] W_N^{km} W_N^{ln}, \; 0 \le k, l \le N-1, \tag{1}$$

where $W_N = exp(-\frac{j2\pi}{N})$

2D-DFT



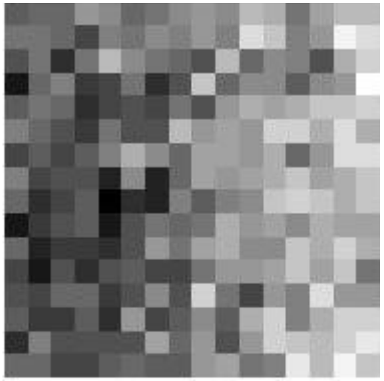
❖ 2D DFT: Extension of 1D-DFT

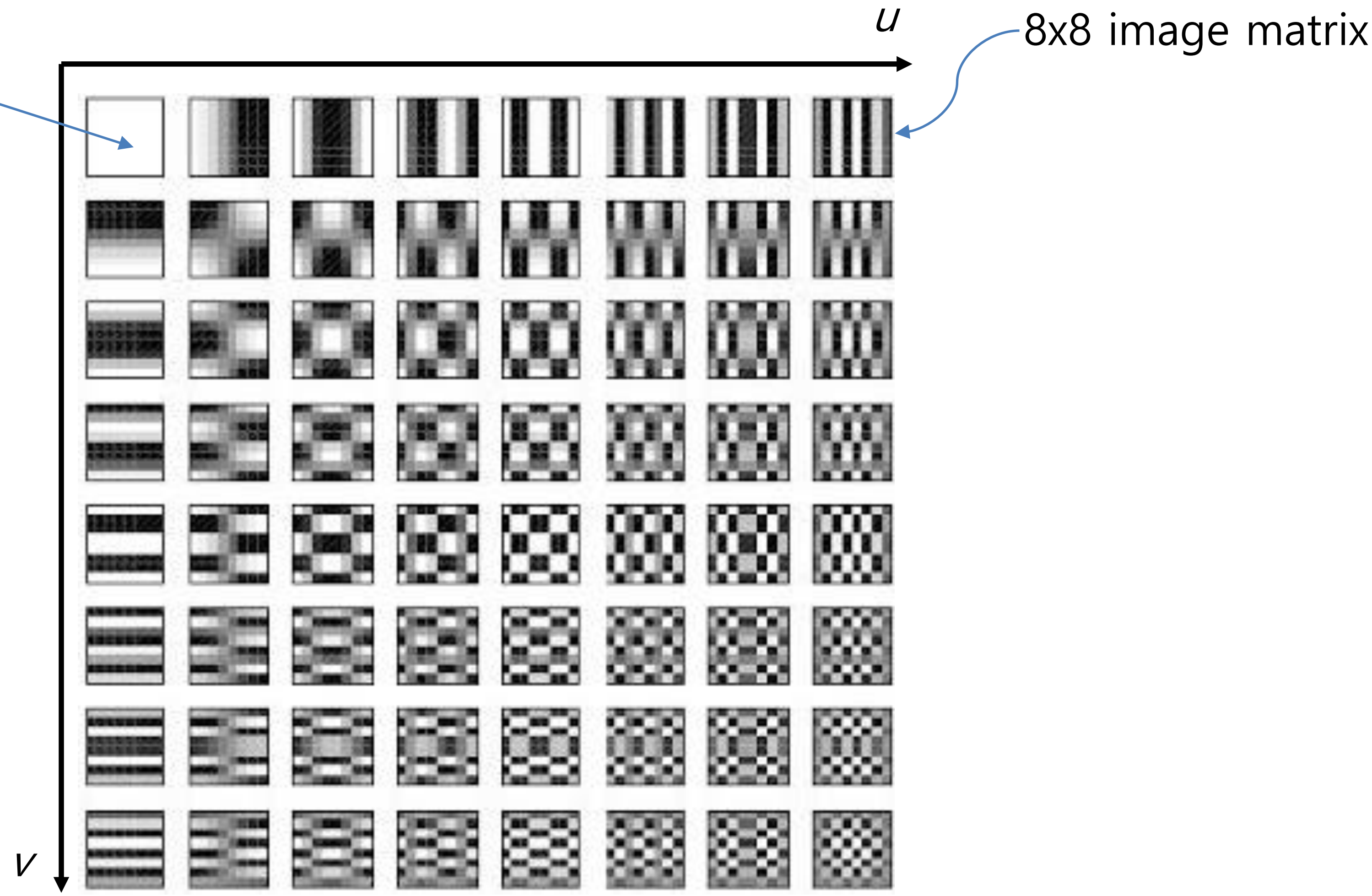
$$V = FXF$$

- F: Fourier transform matrix which contains fourier coeffs..
- X: Input image data matrix

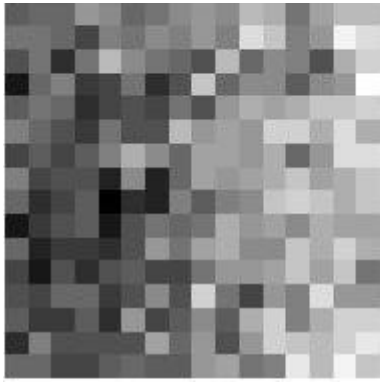
❖ ***Inverse 2D-DFT also can be possible.***

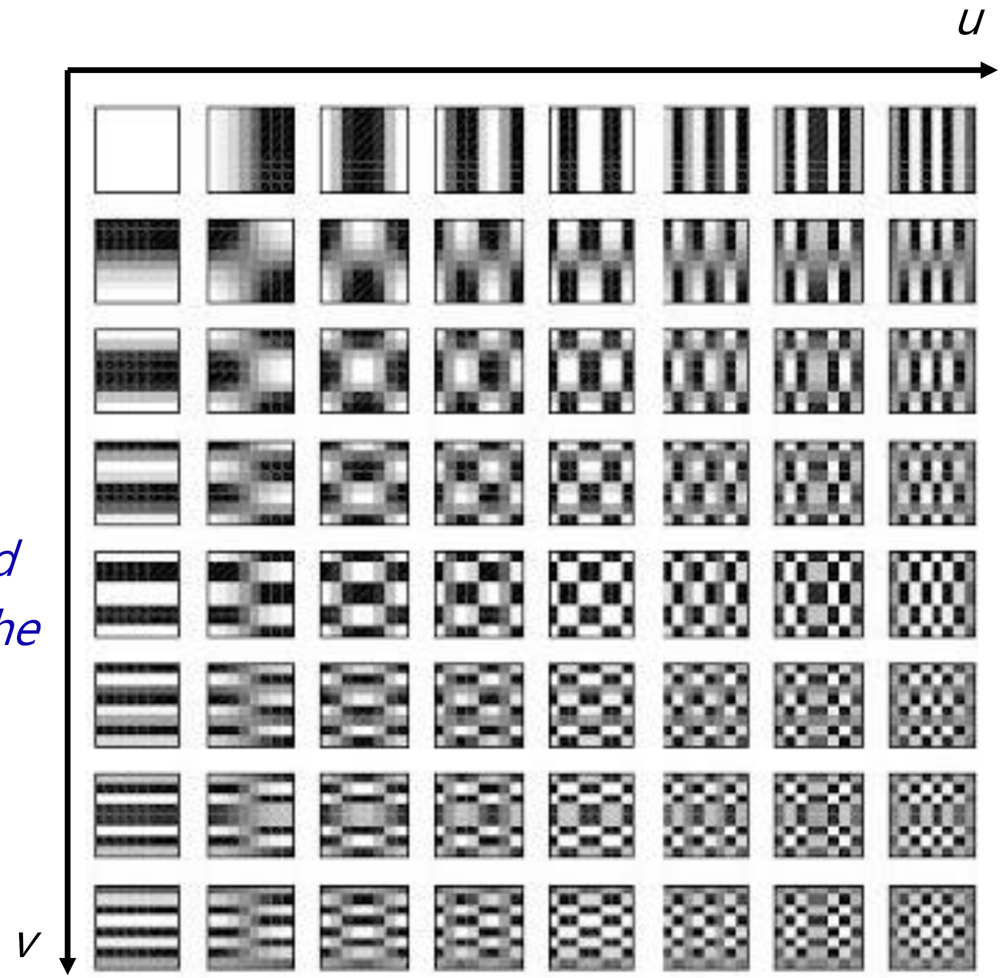
❖ 8x8 DCT basis patterns (functions)





❖ 8x8 DCT basis patterns (functions)





- *All pixels can be presented as the weighted sum of the given basis patterns.*

- ***NxN*** Cosine Transform Matrix ***C*** = {c(*k*, *n*)}

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0,\ 0 \le n \le N-1, \\ \sqrt{\frac{2}{N}} \cos(\frac{\pi(2n+1)k}{2N}), & 1 \le k \le N-1, 0 \le n \le N-1. \end{cases} \quad (1)$$

- 2D-DCT

$$V = CXC^T$$

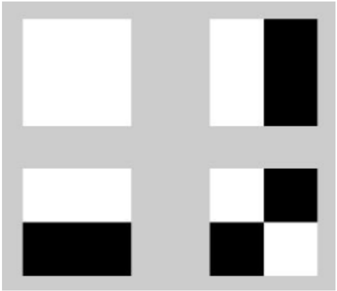
Image matrix

- 2D Inverse DCT

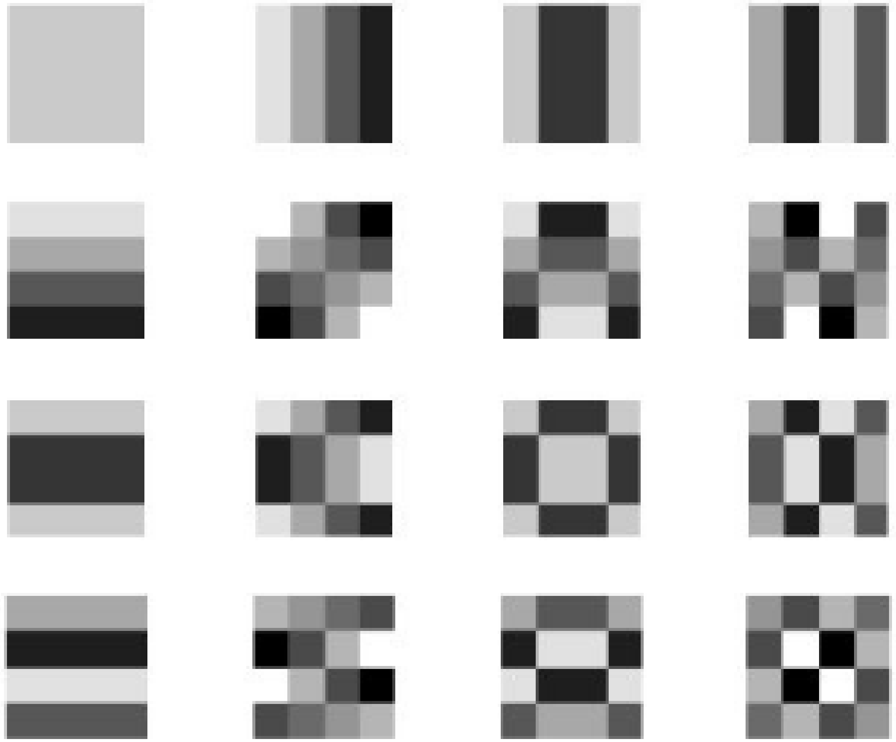
$$X = C^T V C$$

❖ 2 point (2x2) DCT and 4 point (4x4) DCT

2x2 DCT



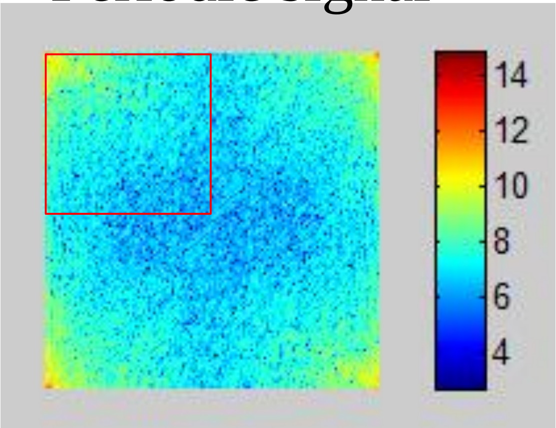
4x4 DCT



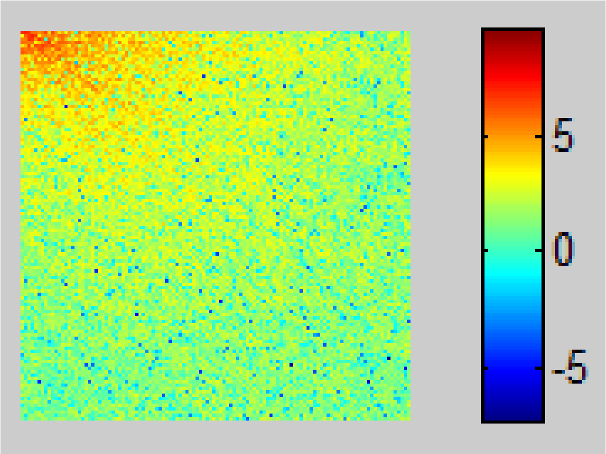
❖ 2D-DCT Example1:

Airplane (128x128)

Periodic signal



❖ 2D-DCT Example2:

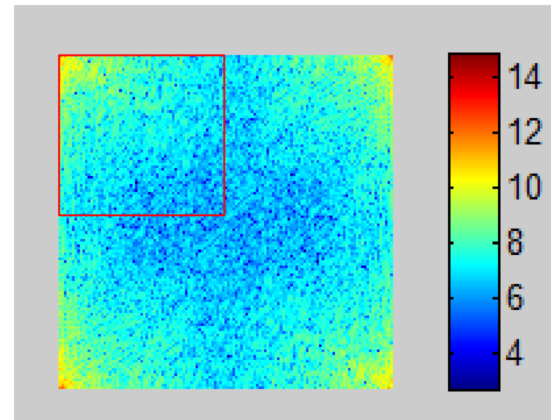


❖ Comparison of Two Transforms (DCT vs. DFT)

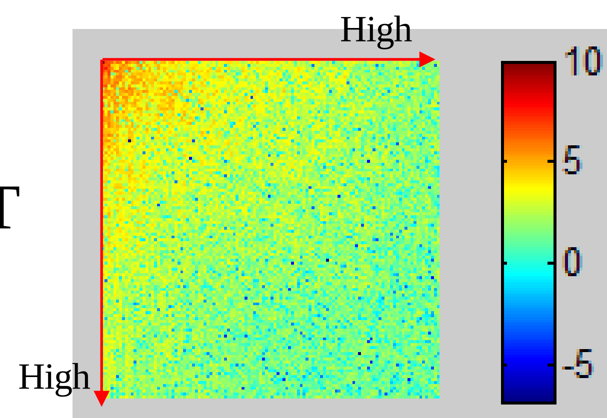
- In terms of energy compaction.
  - DCT wins…!!!!

Airplane (128x128)

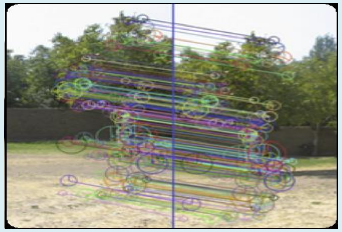
DFT

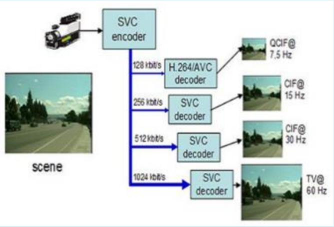


DCT



- Discrete Sine Transform (DST)
  - Basis function: sine function
    - 1D DST
    - 2D DST
- Discrete Hadamard Transform (DHT)
  - Basis vectors are the binary value (± 1).
- Discrete Haar Transform (DHT)
- Discrete KL Transform (KLT)
  - Basis vectors are the orthgonormalized eigenvector of its autocorrelation matrix

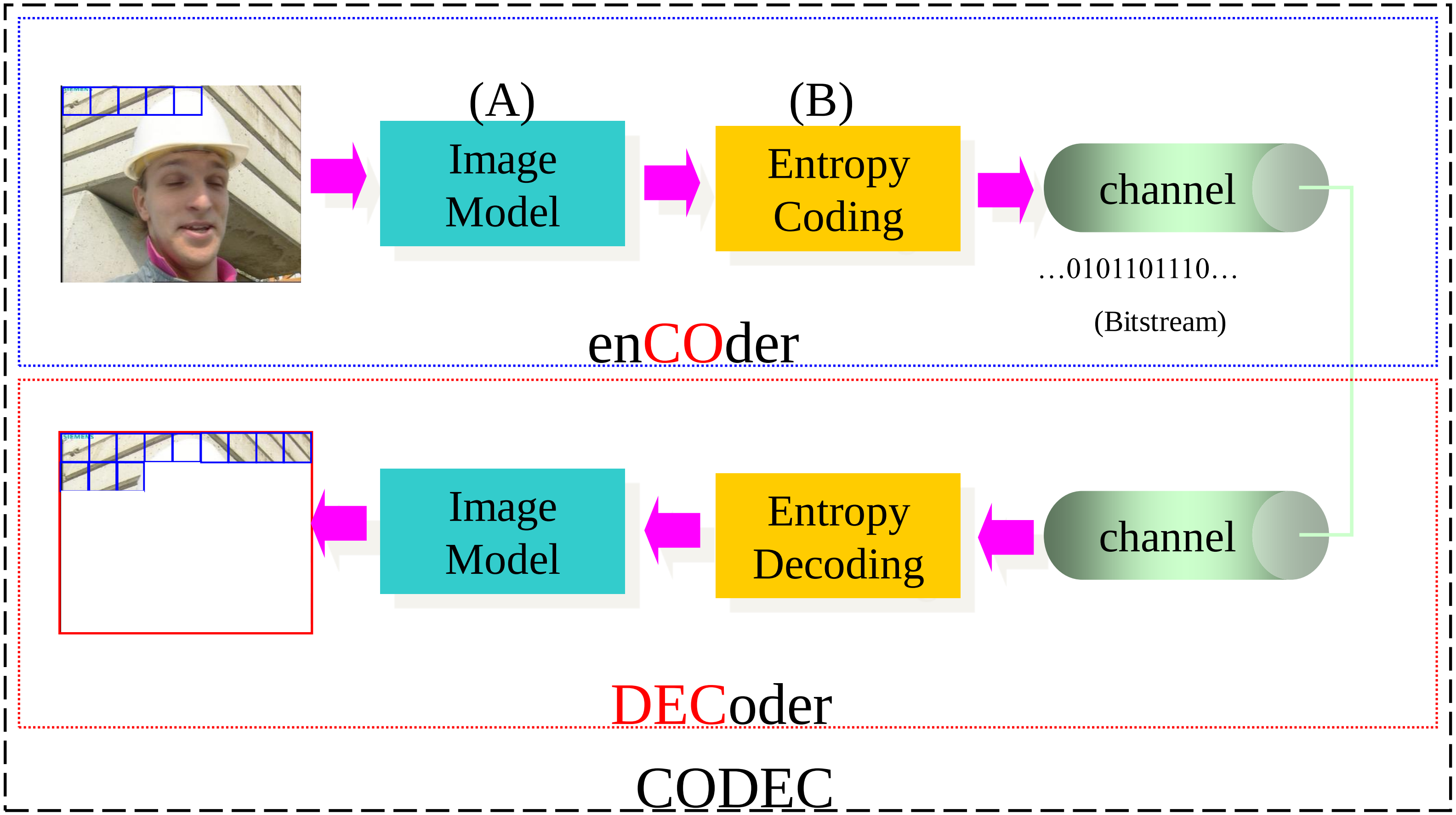




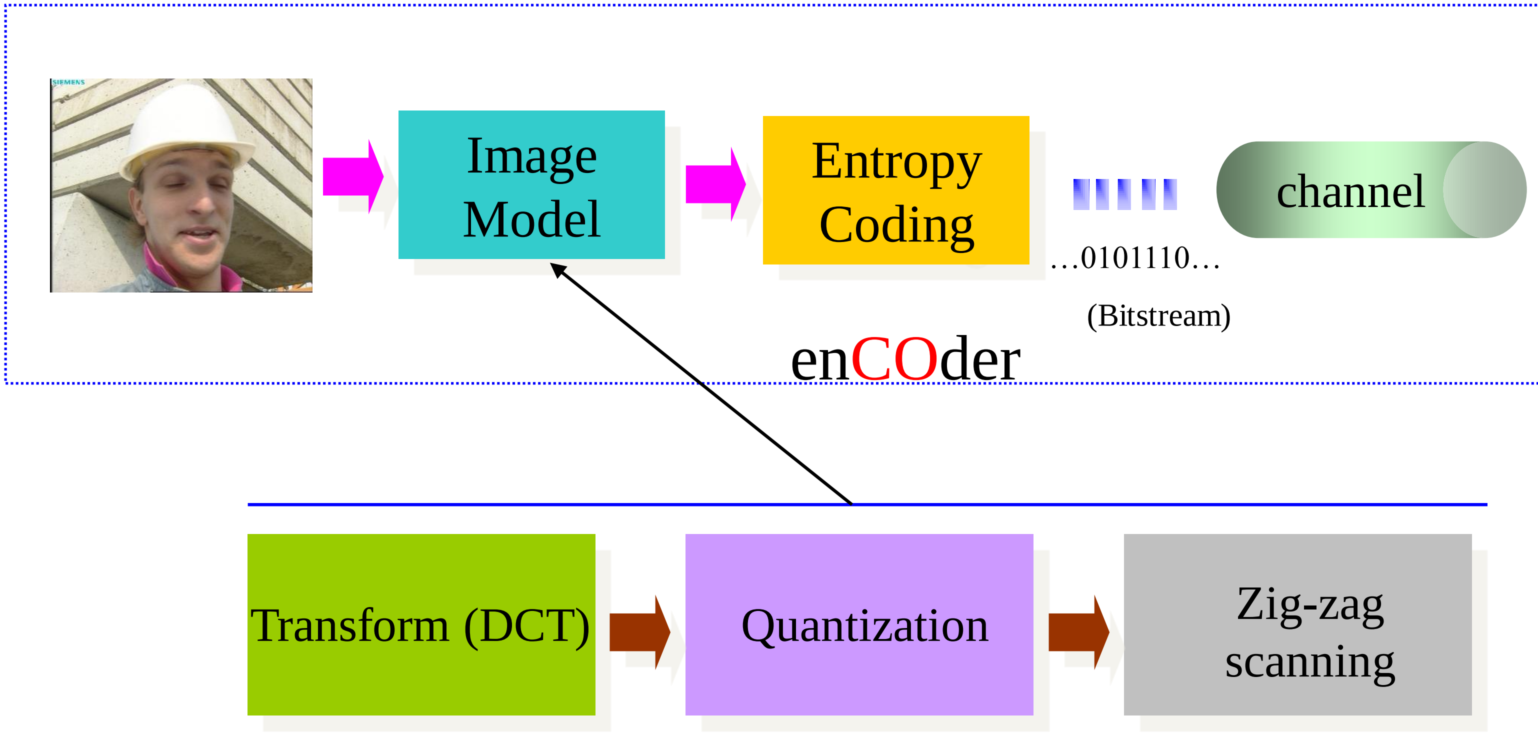
Contents

- Example of Huffman Code
- Signal Transform
- Basic Concept of Image Coding

❖ Image Coding Model: Block unit based processing

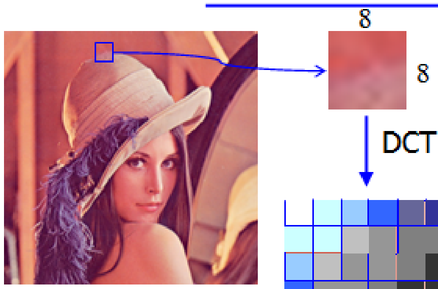


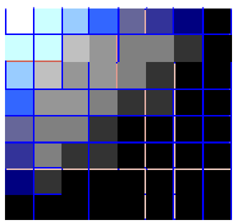
❖ Image Model (A)



❖ The Details of Image Model (A)

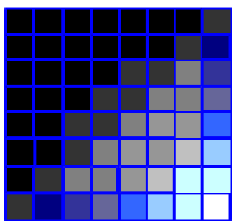
Transform (DCT) → Quantization → Zig-zag scanning





Quantization Matrix

Human visual perception aspect:
The eye is insensitive to "high frequency" image information



Coeff. Scanning Order

Run-length code:
To group the low frequency coefficients in top of vectors.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

84	34	4	0	0	4	0	0
54	0	0	0	0	0	0	0
15	0	14	0	0	0	0	0
0	0	0	0	0	0	0	0
0	7	0	0	0	0	0	0
0	0	0	0	0	5	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

20	5	0	0	0	0	0	0
8	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$$\underbrace{20,5,8,1,0,0,0,0,0 \ldots\ldots,0}_{\text{64 coeffs.(symbols)}}$$

Image Coding (4): Still images

❖ Quantization (Matrix): 8x8 block in JPEG

– Luminance

$$\begin{array}{r c}
 & \text{High freq.} \\
\text{High freq.} & \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}
\end{array}$$

– Chrominance

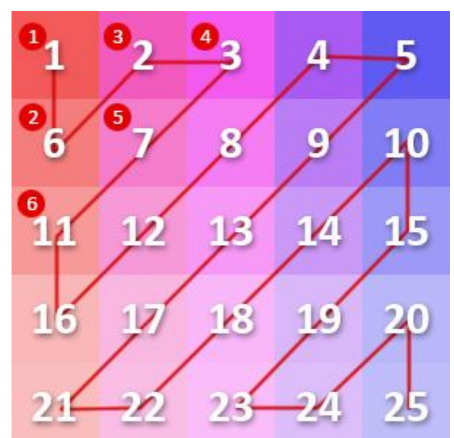
$$\begin{array}{r c}
 & \text{High freq.} \\
\text{High freq.} & \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}
\end{array}$$

Image Coding (4-1): Still images

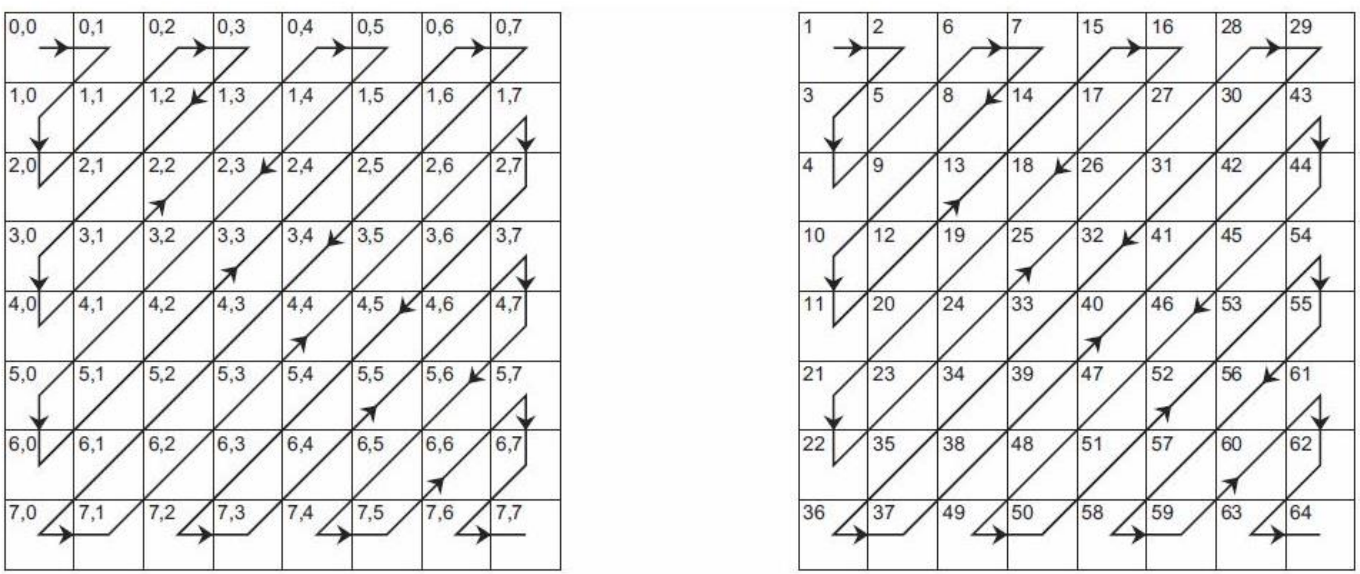
❖ Result example after Quantization

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- How to get symbol group?
  - Progressive direction?
  - Otherwise?



❖ Zig-zag Symbol Scanning



❖ Alternate Zig-zag Pattern

- More suitable for coding of some interlaced picture blocks.
- Added in MPEG-2 video standard.

8

8

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

[the order of symbol scanning]

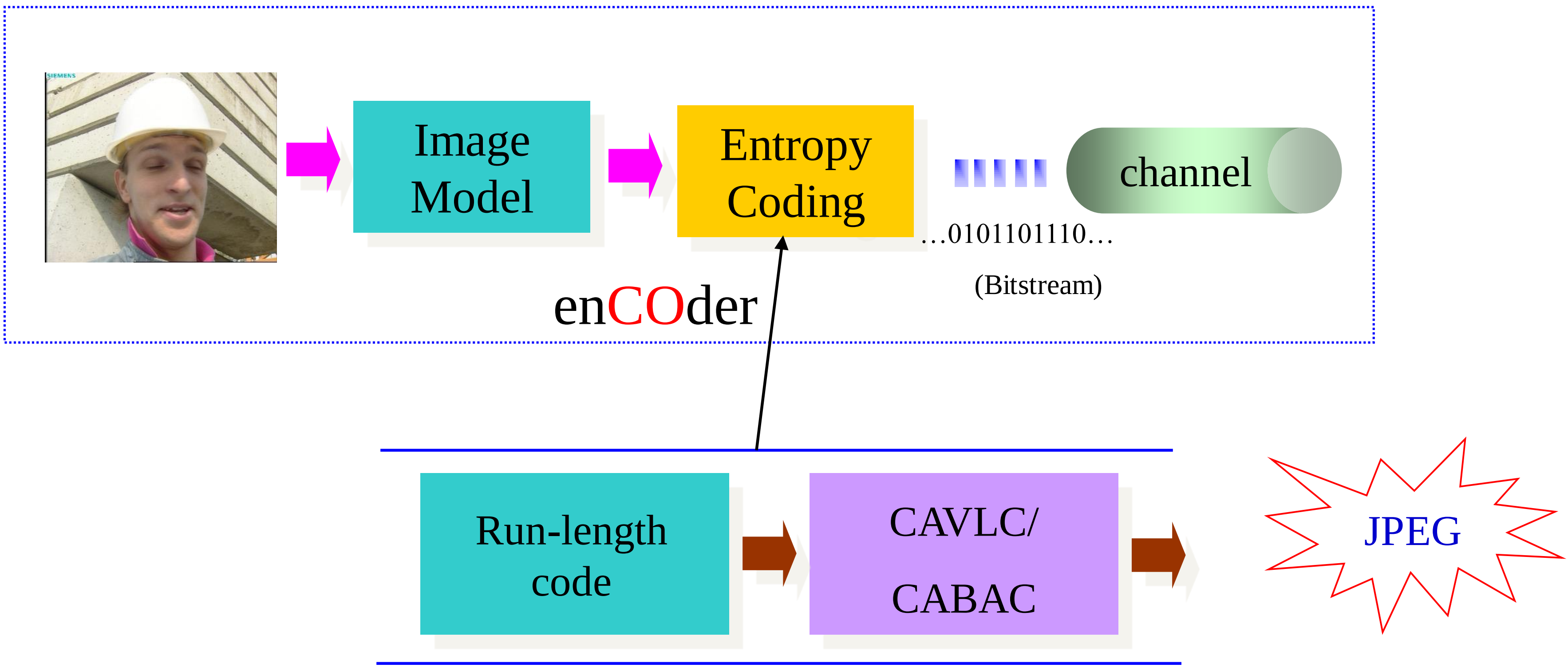
Image Coding (5-2): Still images

❖ Example of Zig-zag Scanning

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Image Coding (6): Still images

❖ Image Model (B)



❖ Run-length Encode (RLE)

- Simple form of data compression in which *runs* of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.

W: white
B: black

WWWWWWWWWBWWWWBBBWWWWWWWWWWBWWWWWWW

↓

9W1B4W3B10W1B7W

$\overbrace{\text{20,5,8,1,0,0,0,0,0 …..,0}}^{\text{64 coeffs.(symbols)}}$ ⇒ (20,1)(5,1)(8,1)(1,1)(0,60)

- CAVLC/CABAC: Symbol bit assignment

(20,1)(5,1)(8,1)(1,1)(0,60) → Huffman Code →

⋮	⋮
(20, 1)	101
(5,1)	10
(8,1)	11
(1,1)	01
⋮	⋮

→ Bit Writing → 0101110…

(Bitstream)

In the next Seminar..?

- Video Coding Model
- Detailed Processes for Video Coding
  - Processing Hierarchy of Video Encoding Scheme
  - Encoding-Decoding Structures (frame unit)
- Color format of the Original Source

Thank you for your attention.!!!

QnA

http://vicl.sookmyung.ac.kr