CS 598KN

Advanced Multimedia Systems Design

Lecture 2 – Video and Basic Compression Concepts

Klara Nahrstedt

Fall 2017

Overview

- Basics of Video Characteristics
- Basic Coding Concepts
- RLE, Huffman Coding, JPEG

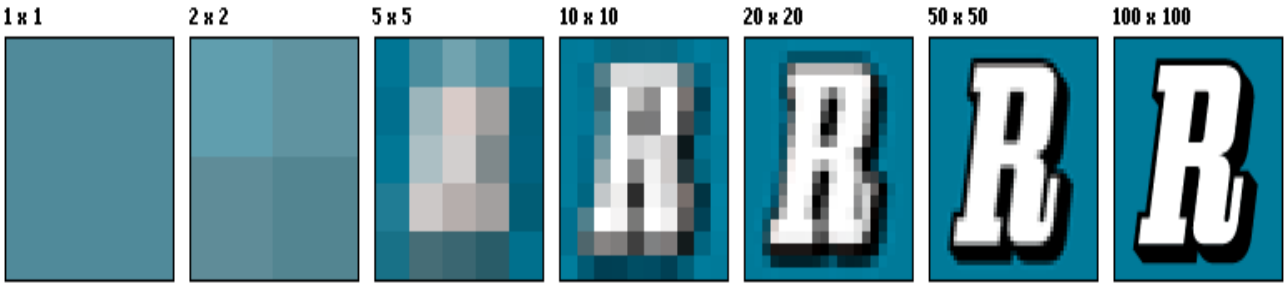
VIDEO CHARACTERISTICS

Video

- Sequence of Digital Images
- Important Video Characteristics
  - Spatial Characteristics
    - Video Image Resolution characterized by Size and Viewing Distance, Brightness, Color
  - Temporal Characteristics
    - Temporal resolution characterized by Video Frame Rate and Flicker avoidance

Visual Perception: Resolution and Brightness

- Spatial Resolution (depends on:)
  - Image size
  - Viewing distance



- Brightness
  - Perception of brightness is higher than perception of color
  - Different perception of primary colors
    - Relative brightness: green:red:blue=59%:30%:11%

- B/W vs. Color

Source: wikipedia

Temporal Resolution

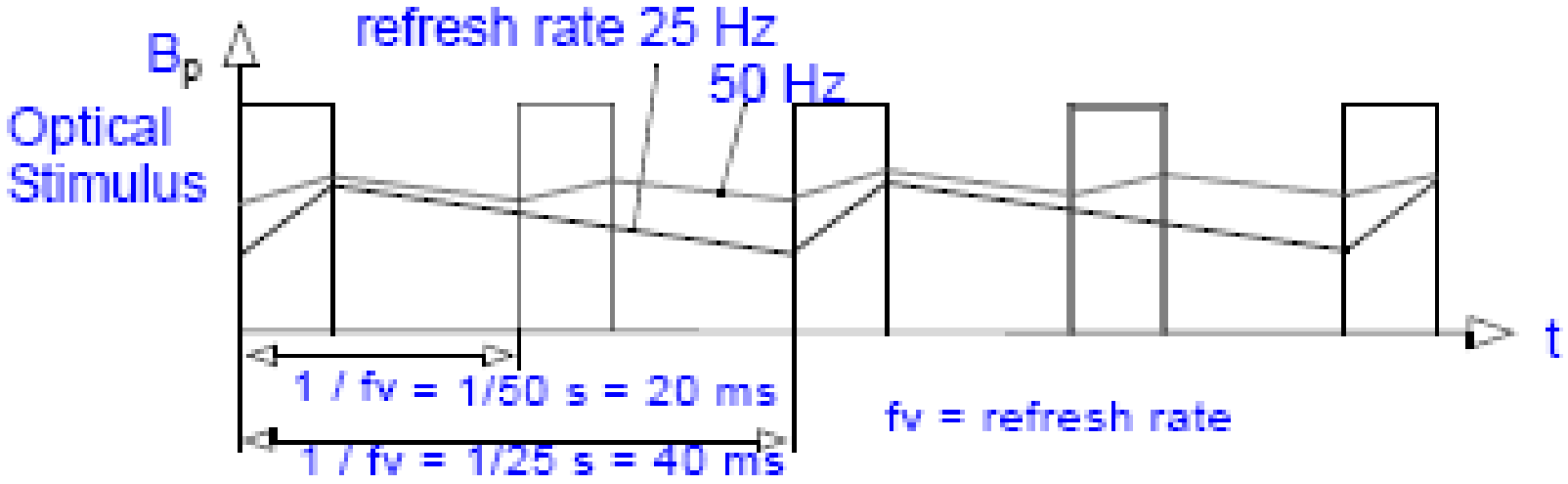
- Flicker
  - Perceived if frame rate or refresh rate of screen too low (<50Hz)
  - Especially in large bright areas
- Higher refresh rate requires
  - Higher scanning frequency
  - Higher bandwidth

Visual Perception Influence

- Viewing distance
- Display ratio (width/height – 4/3 for conventional TV, new TVs have ratio 16/9)
- Number of details still visible
- Intensity (luminance)

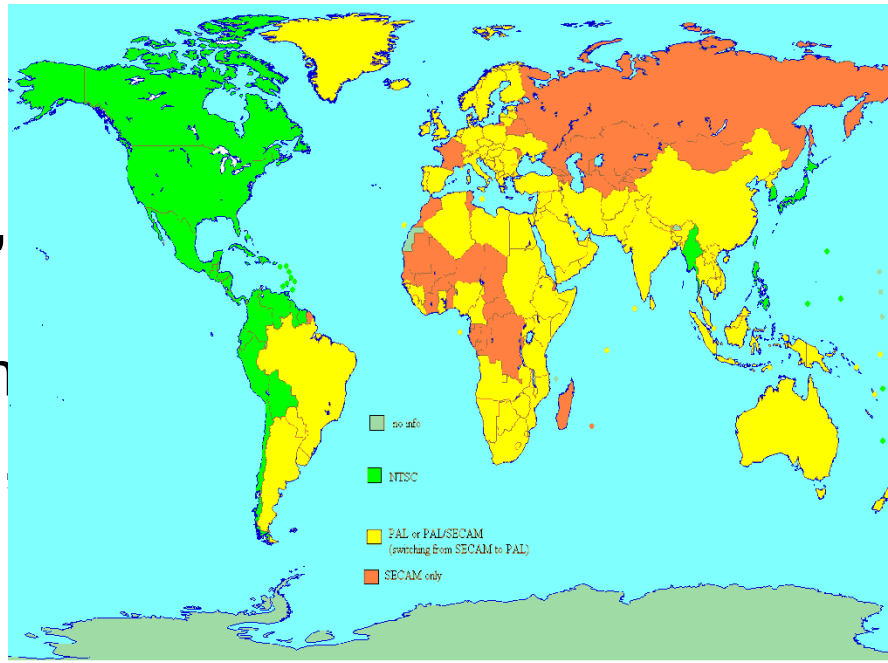
Visual Perception: Temporal Resolution

- *Effects caused by inertia of human eye*
- *Perception of 16 frames/second as continuous sequence*
- *Special Effect: Flicker*



Analog Video (TV)

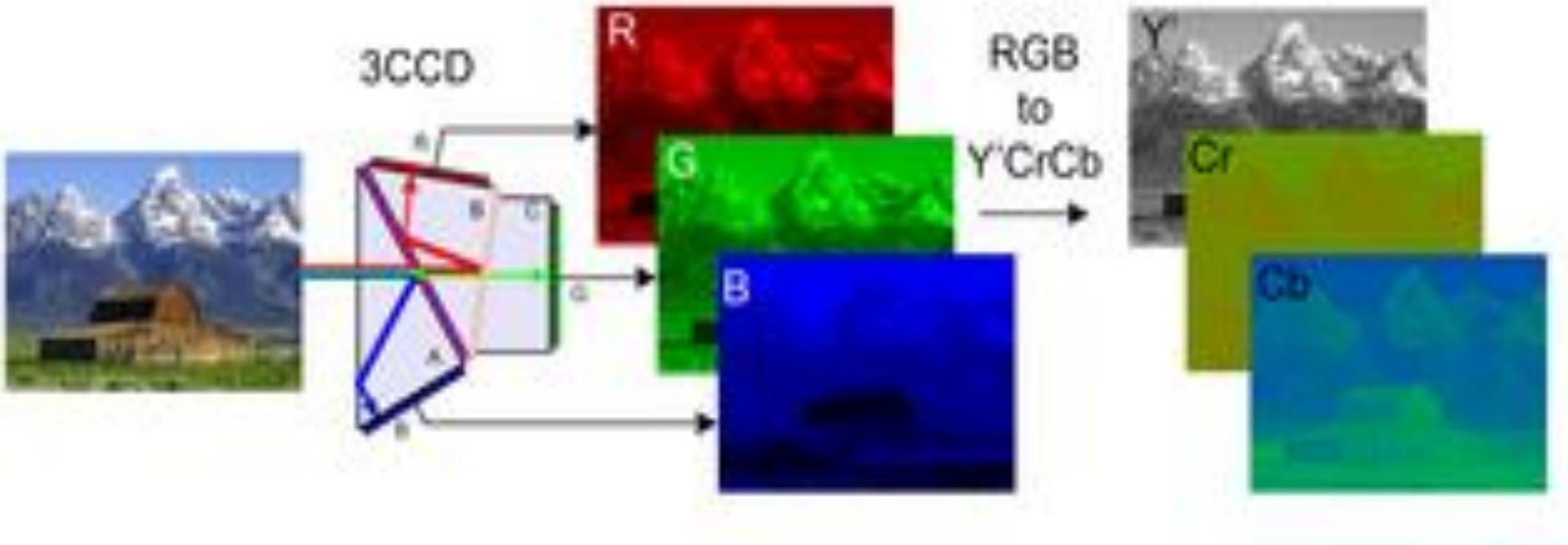
- Production (capture)
  - 2D array of light energy to electrical signals
  - signals must adhere to known, structured formats
- Representation and Transmission
  - popular formats include NTSC, PAL, SECAM, HDTV
- Re-construction
  - CRT technology and raster scanning
  - display issues (refresh rates, temporal resolution)
  - relies on principles of human visual system



Analog Video Representations

- Composite
  - USA NTSC - 6MHz (4.2MHz video), 29.97 fps
- Component
  - Maintain separate signals for color
- **Color spaces**
  - RGB, YUV, YC_RC_B, YIQ (color space in USA)
  - Y component determines **brightness** of the color (luminance or luma)
  - U and V components determine the **color** itself (chroma)

RGB to YCbCr Coversion

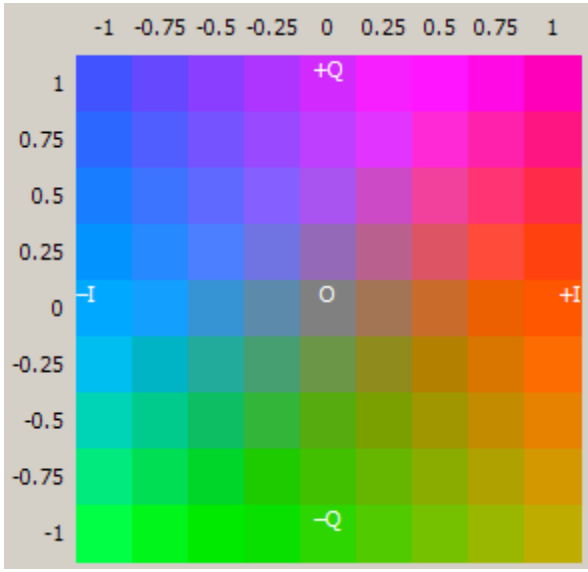


Source: Wikipedia

YIQ

- NTSC standard
- Y – luma information
- I – in-phase (chroma)
- Q quadratre amplitude modulation (chroma)
- YIQ from RGB

$$Y = .299R + .587G + .114B$$
$$I = .74 (R - Y) - .27 (B - Y)$$
$$Q = 0.48 (R - Y) + 0.41 (B - Y)$$



YIQ with Y=0.5

Source: wikipedia

HDTV

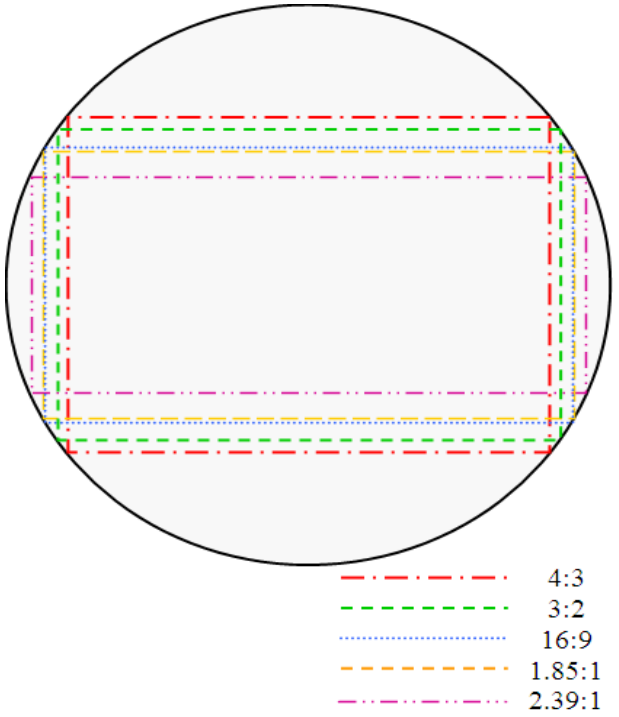
- Digital Television Broadcast (DTB) System
- Twice as many horizontal and vertical columns and lines as traditional TV
- Resolutions:
  - 1920x1080 (1080p) – Standard HDTV
- HDTV Frame rate (refresh rate): 60 times a second or 60Hz (60 frames per second);

UHD (Ultra-High-Definition)

- Resolutions:
  - 3840 pixel x 2160 lines (8.3 megapixel) – 4K UHD
  - 7680 pixels x 4320 lines (33.2 megapixel) – 8K UHD
- UHD Frame rate (refresh rate): 120fps and 240fps

Aspect Ratio and Refresh Rate

- Aspect ratio
  - Conventional TV is 4:3 (1.33)
  - HDTV is 16:9 (2.11)
  - Cinema uses 1.85:1 or 2.35:1
- Frame Rate
  - NTSC is 60Hz interlaced (actually 59.94Hz)
  - PAL/SECAM is 50Hz interlaced
  - Cinema is 24Hz non-interlaced



Source: wikipedia

BASIC CODING CONCEPTS

Data Compression

- Branch of information theory
  - minimize amount of information to be transmitted
- Transform a sequence of characters into a new string of bits
  - same information content
  - length as short as possible

Concepts

- Coding (the code) maps source messages from alphabet (*A*) into code words (*B*)
- Source message (symbol) is basic unit into which a string is partitioned
  - can be a single letter or a string of letters
- EXAMPLE: aa bbb cccc ddddd eeeeee fffffffgggggggg
  - *A* = {a, b, c, d, e, f, g, space}
  - *B* = {0, 1}

Taxonomy of Codes

- Block-block
  - source msgs and code words of fixed length; e.g., ASCII
- Block-variable
  - source message fixed, code words variable; e.g., Huffman coding
- Variable-block
  - source variable, code word fixed; e.g., RLE, LZW
- Variable-variable
  - source variable, code words variable; e.g., Arithmetic

Example of Block-Block

- Coding “aa bbb cccc ddddd eeeeee fffffffgggggggg”
- Requires 120 bits

Symbol	Code word
a	000
b	001
c	010
d	011
e	100
f	101
g	110
space	111

Example of Variable-Variable

- Coding “aa bbb cccc ddddd eeeeee fffffffgggggggg”
- Requires 30 bits
  - don’t forget the spaces

Symbol	Code word
aa	0
bbb	1
cccc	10
ddddd	11
eeeeee	100
fffffff	101
gggggggg	110
space	111

Static Codes

- Mapping is fixed before transmission
  - message represented by same codeword every time it appears in ensemble
  - Huffman coding is an example
- Better for independent sequences
  - probabilities must be known in advance; or values computed from other data sources

Dynamic Codes

- Mapping changes over time
  - also referred to as *adaptive* coding
- Attempts to exploit locality of reference
  - periodic, frequent occurrences of messages
  - dynamic Huffman is an example
- Hybrids?
  - build set of codes, select based on input

Traditional Evaluation Criteria

- Algorithm complexity
  - running time
- Amount of compression
  - redundancy
  - compression ratio
- How to measure?

Measure of Information

- Consider symbols s_i and the probability of occurrence of each symbol $p(s_i)$
- In case of **fixed-length coding** , smallest number of bits per symbol needed is
  - $L \geq log_2(N)$ bits per symbol
  - Example: Message with 5 symbols need 3 bits $(L \geq log_2 5)$

Variable-Length Coding- Entropy

- What is the minimum number of bits per symbol?
- Answer: Shannon's result – theoretical minimum average number of bits per code work is known as Entropy (H)

$$\sum_{i=1}^{n} - p(s_i) \log_2 p(s_i)$$

Entropy Example

- Alphabet = {A, B}
  - p(A) = 0.4; p(B) = 0.6
- Compute Entropy (H)
  - $-0.4*\log_2 0.4 + -0.6*\log_2 0.6 = .97$ bits
- Maximum uncertainty (gives largest H)
  - occurs when all probabilities are equal

Redundancy

- Difference between avg. codeword length (L) and avg. information content (H)
  - If H is constant, then can just use L
- Relative to the optimal value

Compression Ratio

- Compare the average message length and the average codeword length
  - e.g., average L(message) / average L(codeword)
- Example:
  - {aa, bbb, cccc, ddddd, eeeeee, fffffff, gggggggg}
  - average message length is 5
- Relative to the original data

Symmetry

- Symmetric compression
  - requires same time for encoding and decoding
  - used for live mode applications (teleconference)

- Asymmetric compression
  - performed once when enough time is available
  - decompression performed frequently, must be fast
  - used for retrieval mode applications (e.g., an interactive CD-ROM)

Entropy Coding Algorithms (Content Dependent Coding)

- Run-length Encoding (RLE)
  - Replaces sequence of the same consecutive bytes with number of occurrences
  - Number of occurrences is indicated by a special flag (e.g., !)
  - Example:
    - abccccccccdeffffggg (20 Bytes)
    - abc!9def!4ggg (13 bytes)

Variations of RLE (Zero-suppression technique)

- Assumes that only one symbol appears often (blank)
- Replace blank sequence by M-byte and a byte with number of blanks in sequence
  - Example: M3, M4, M14,…
- Some other definitions are possible
  - Example:
    - M4 = 8 blanks, M5 = 16 blanks, M4M5=24 blanks

Huffman Encoding

- Statistical encoding
- To determine Huffman code, it is useful to construct a binary tree
- Leaves are characters to be encoded
- Nodes carry occurrence probabilities of the characters belonging to the subtree

- Example (homework): How does a Huffman code look like for symbols with statistical symbol occurrence probabilities:

  P(A) = 8/20, P(B) = 3/20, P(C) = 7/20, P(D) = 2/20?

Huffman Encoding (Example)

Step 1 : Sort all Symbols according to their probabilities (left to right) from Smallest to largest

these are the leaves of the Huffman tree

P(B) = 0.51

P(C) = 0.09

P(E) = 0.11

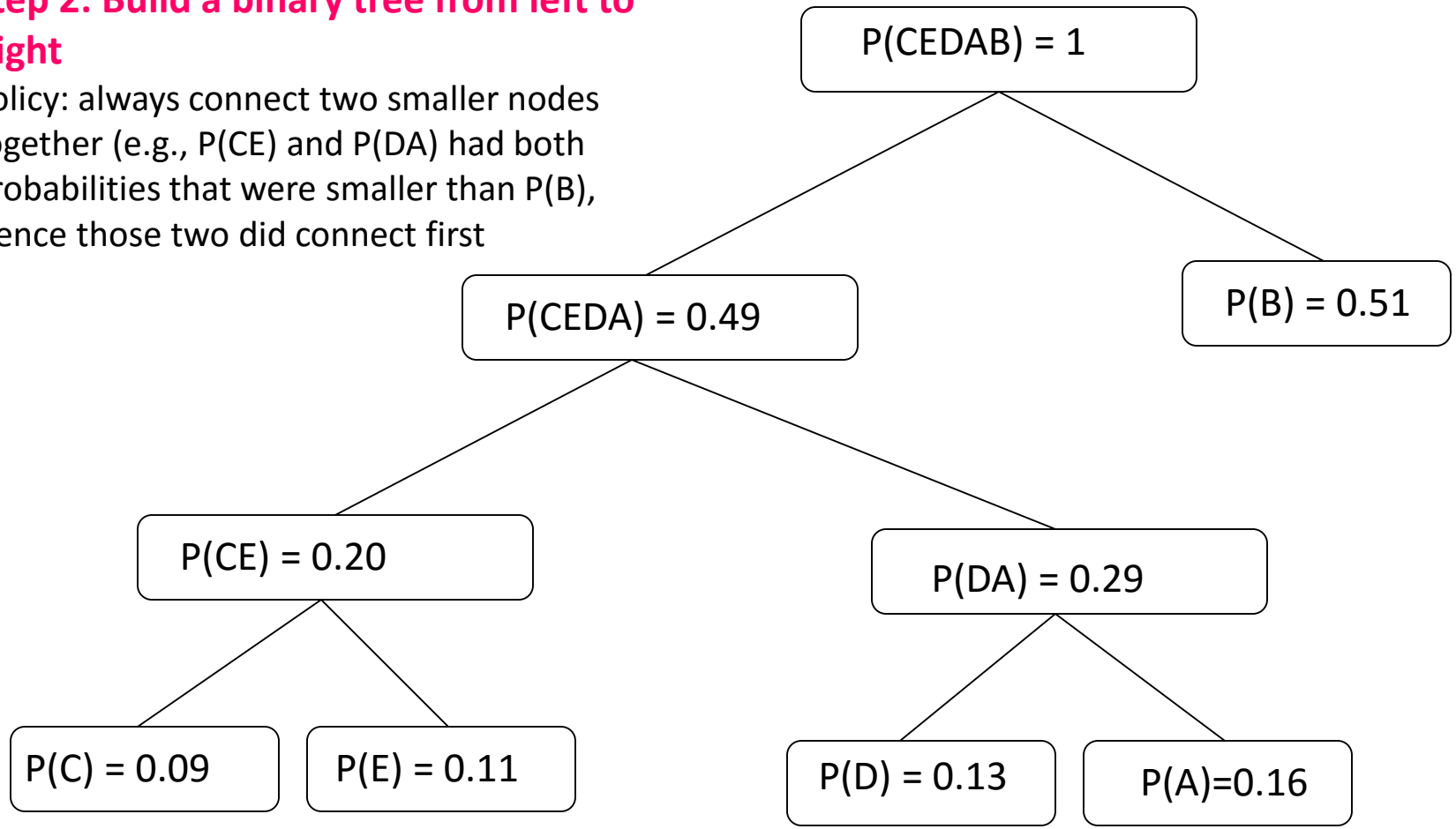
P(D) = 0.13

P(A)=0.16

Huffman Encoding (Example)

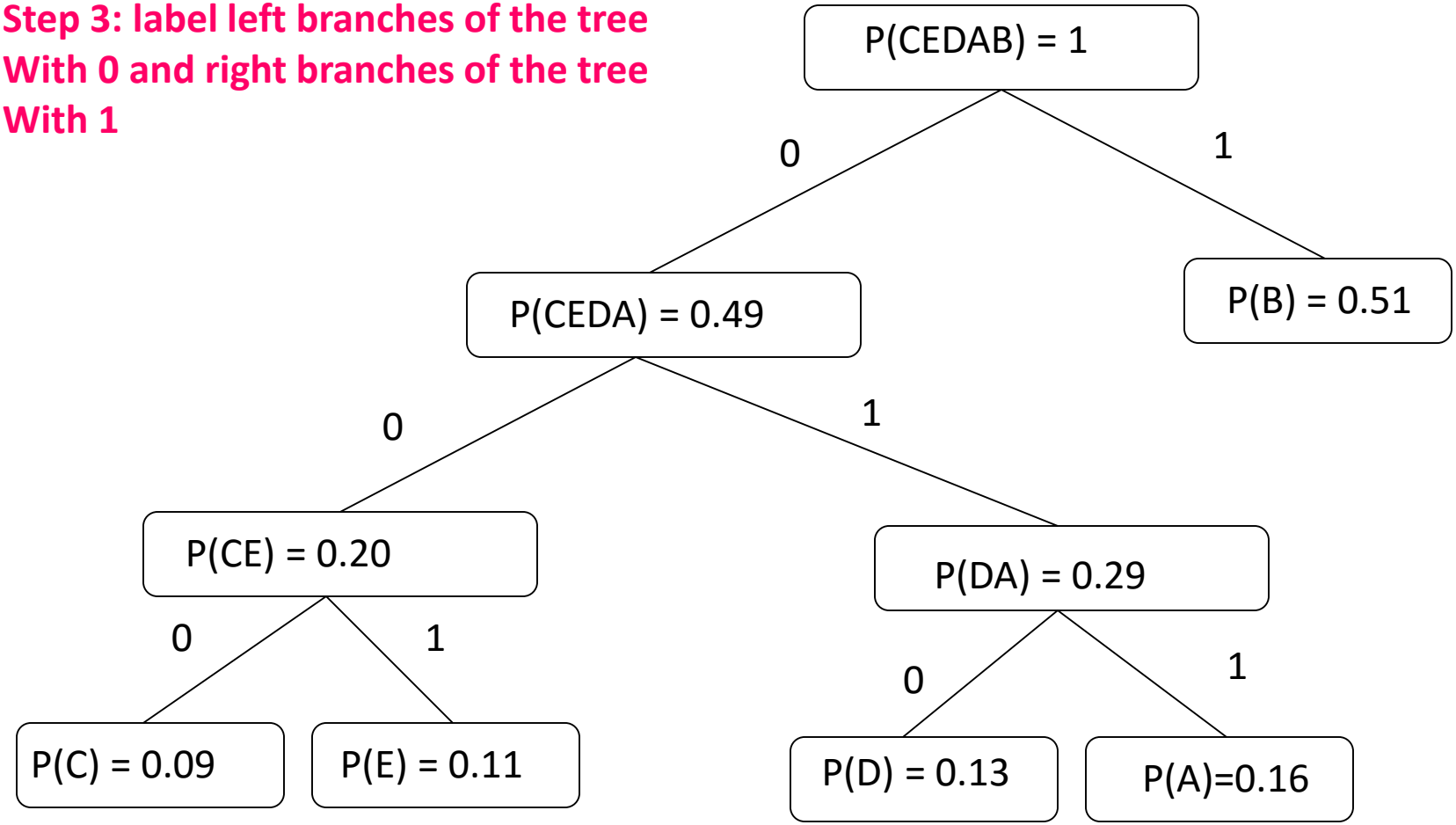
Step 2: Build a binary tree from left to Right

Policy: always connect two smaller nodes together (e.g., P(CE) and P(DA) had both Probabilities that were smaller than P(B), Hence those two did connect first



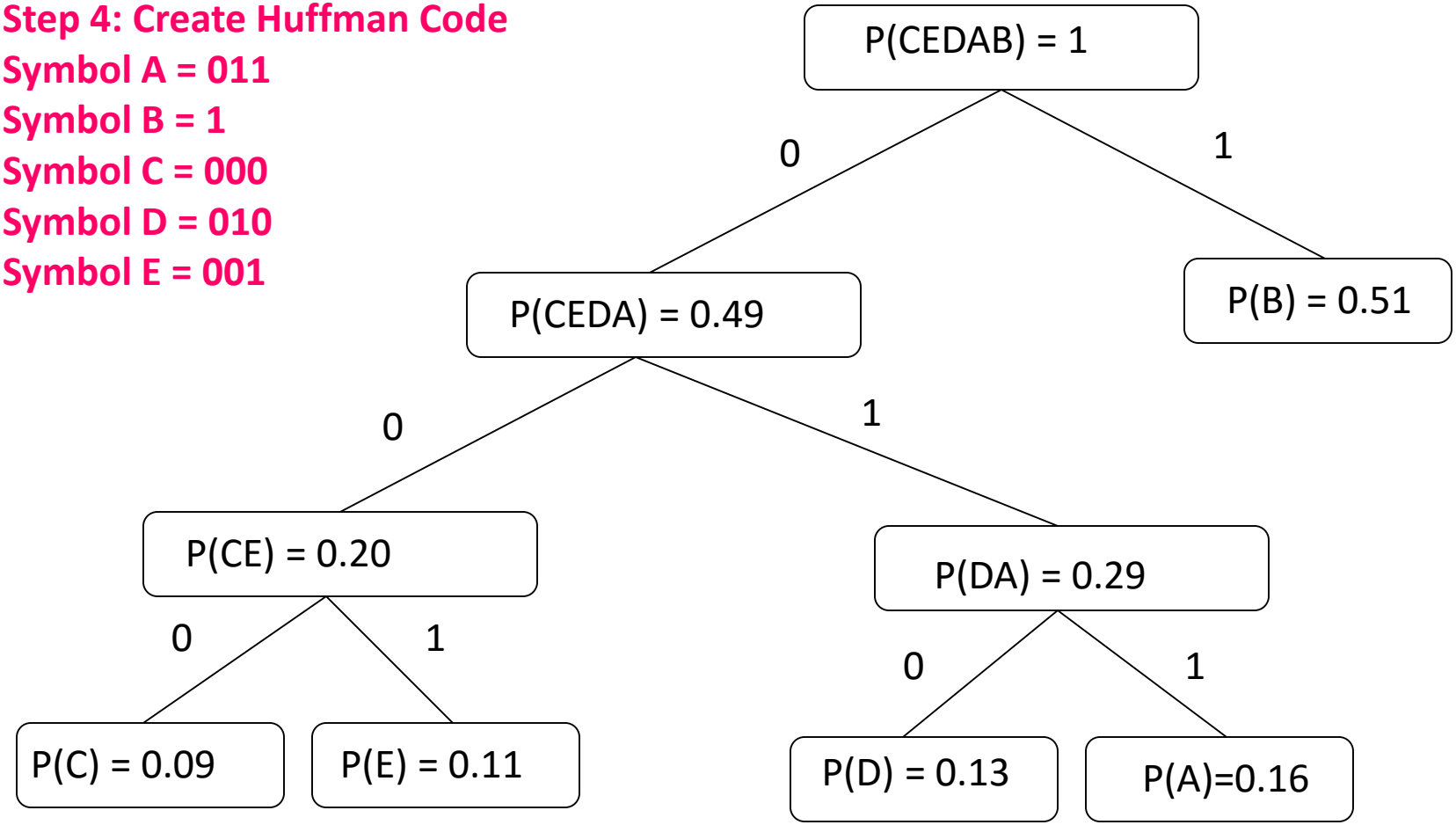
Huffman Encoding (Example)

Step 3: label left branches of the tree With 0 and right branches of the tree With 1



Huffman Encoding (Example)

Step 4: Create Huffman Code
Symbol A = 011
Symbol B = 1
Symbol C = 000
Symbol D = 010
Symbol E = 001

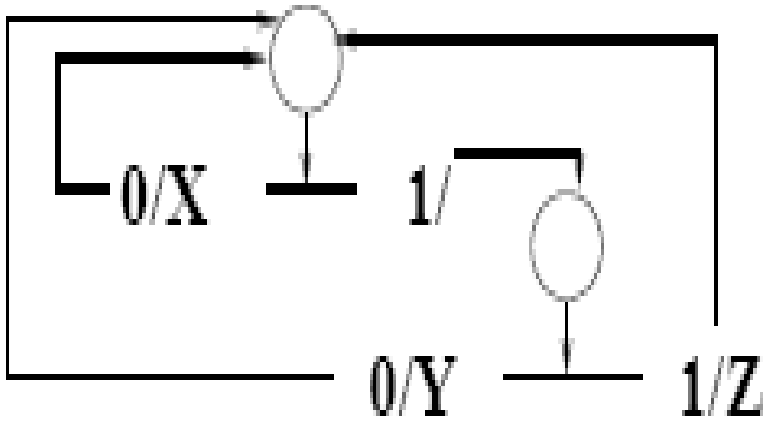


Huffman Decoding

- Assume Huffman Table
- Symbol Code

Symbol	Code
X	0
Y	10
Z	11

Consider encoded bitstream:
000101011001110



What is the decoded string?

Limitations

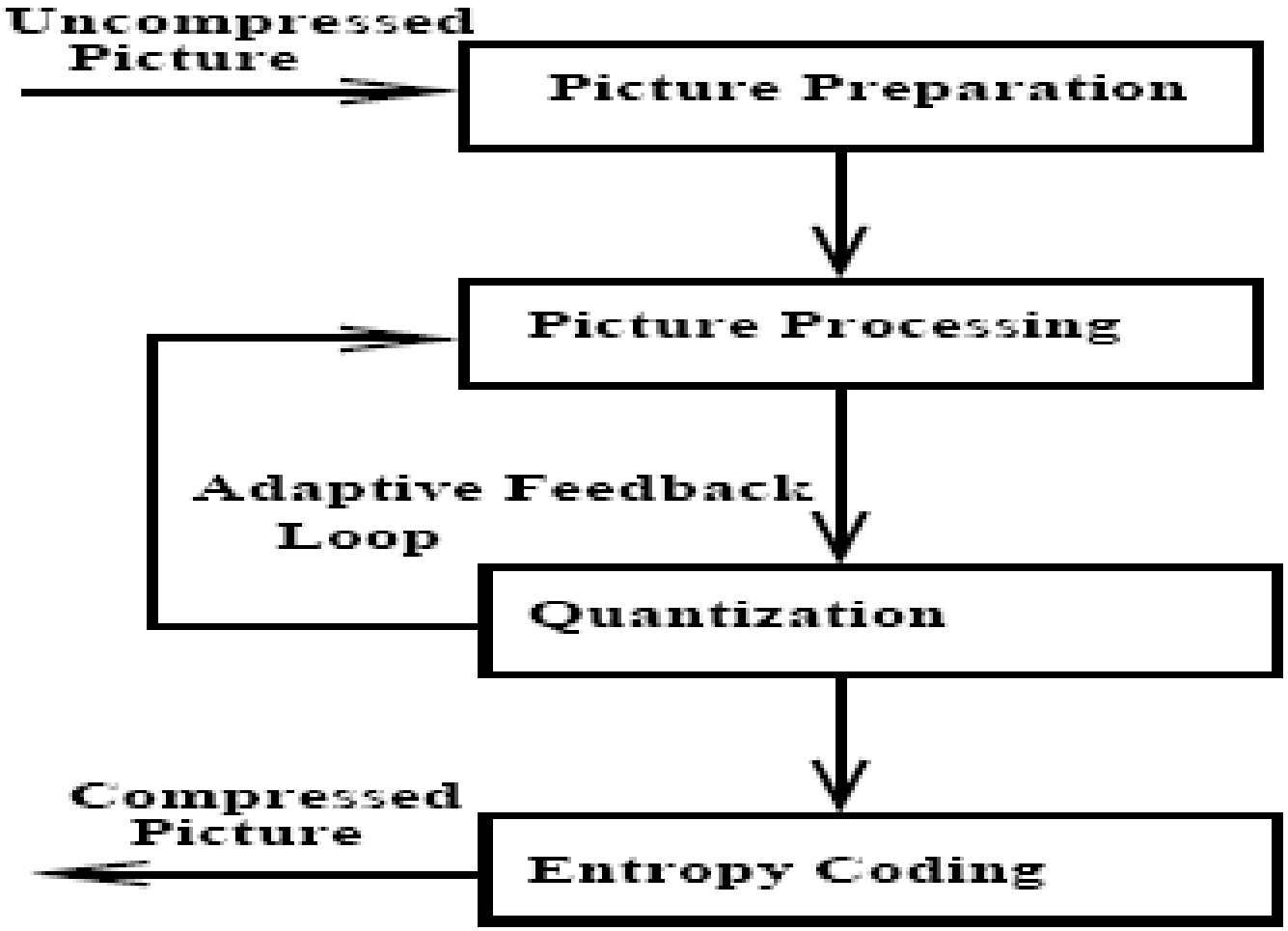
- Diverges from lower limit when probability of a particular symbol becomes high
  - always uses an integral number of bits
- Must send code book with the data
  - lowers overall efficiency
- Must determine frequency distribution
  - must remain stable over the data set

IMAGE COMPRESSION (CONCEPTS USED IN MPEG AND H.26X)

JPEG (Joint Photographic Experts Group)

- Requirements:
  - Very good compression ratio and good quality image
  - Independent of image size
  - Applicable to any image and pixel aspect ratio
  - Applicable to any complexity (with any statistical characteristics)

Hybrid Coding



Picture Preparation

- Generation of appropriate digital representation
- Image division into 8x8 blocks
- Fix number of bits per pixel (first level quantization – mapping from real numbers to bit representation)

Other Compression Steps

- Picture processing (Source Coding)
  - Transformation from time to frequency domain (e.g., use Discrete Cosine Transform)
  - Motion vector computation in video
- Quantization
  - Reduction of precision, e.g., cut least significant bits
  - Quantization matrix, quantization values
- Entropy Coding
  - Huffman Coding + RLE

JPEG Compression

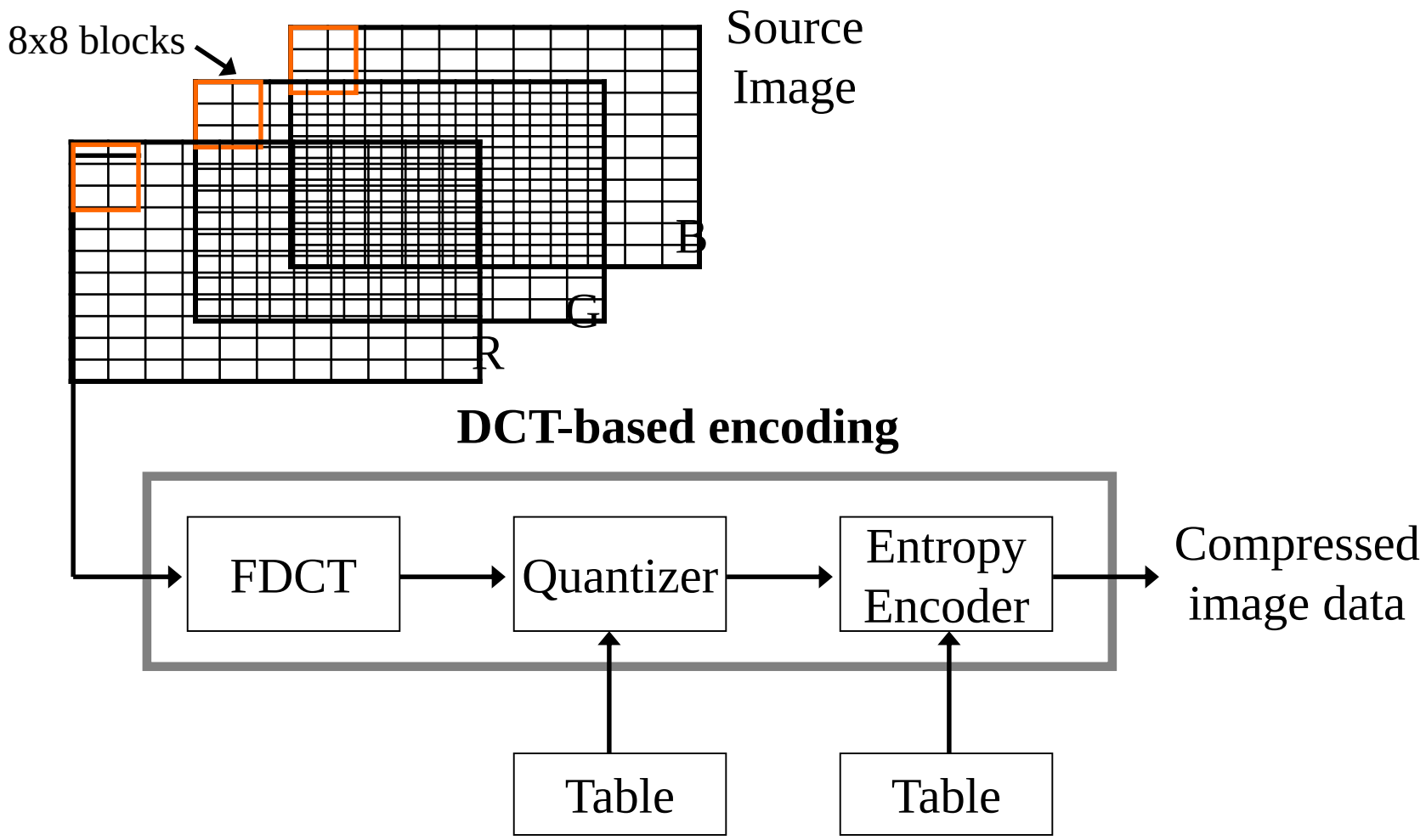
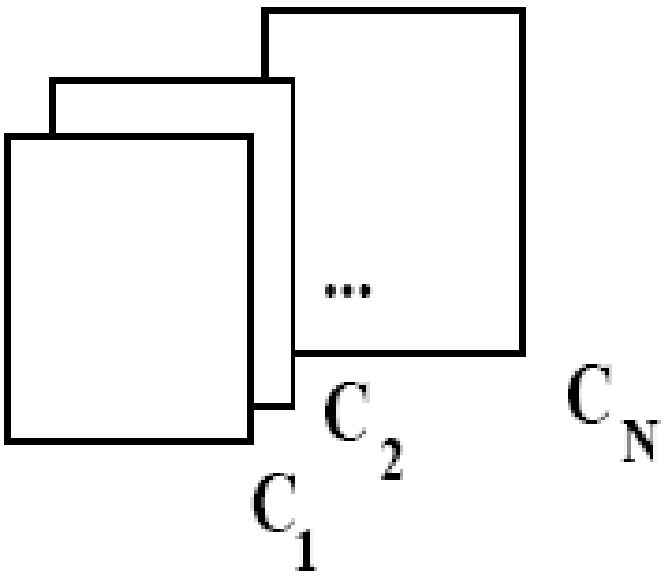
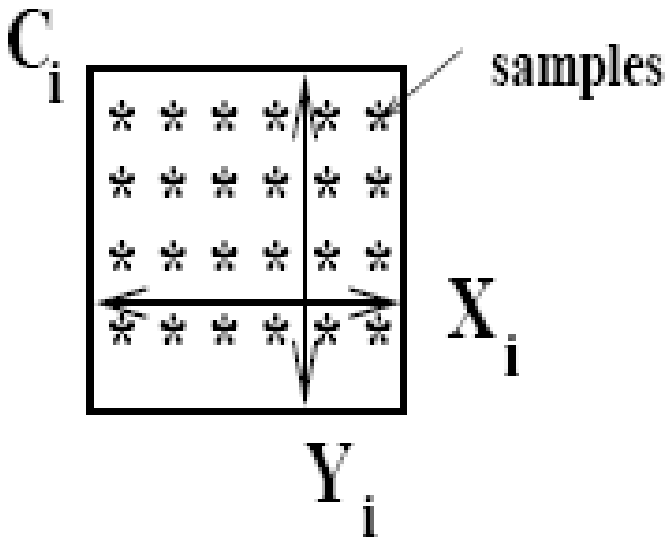


Image Preparation

- The image preparation is NOT BASED on
  - 9-bit YUV encoding
  - Fixed number of lines and columns
  - Mapping of encoded chrominance
- Source image consists of components (C_i) and to each component we assign YUV, RGB or TIQ signals.

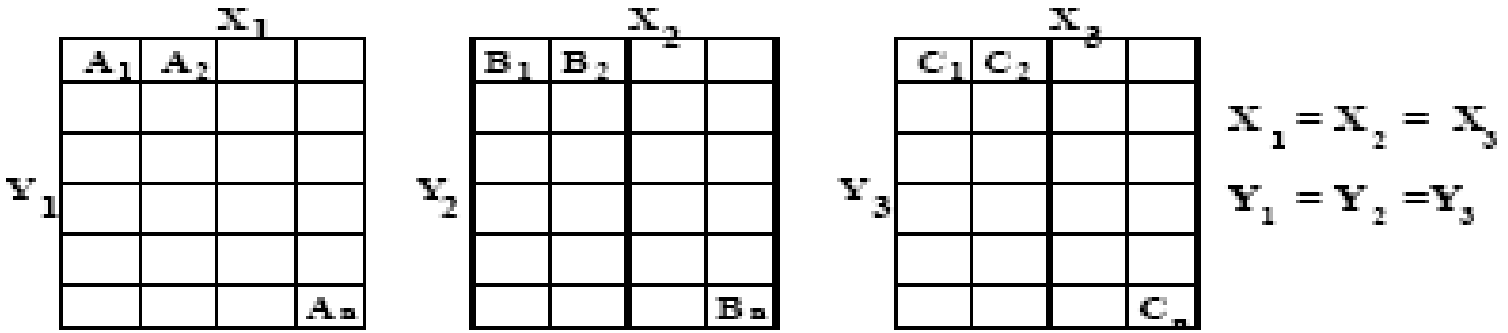
Division of Source Image into Planes



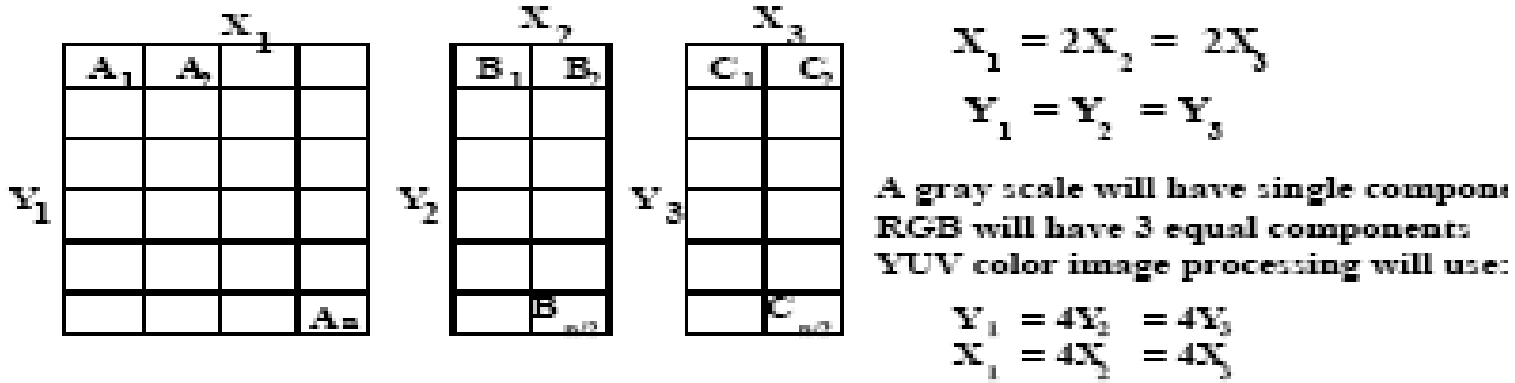


Components and their Resolutions

A./ Components with the same resolution



B./ Components with different resolution



Color Transformation (optional)

- Down-sample chrominance components
  - compress without loss of quality (color space)
  - e.g., YUV 4:2:2 or 4:1:1
- Example: 640 x 480 RGB to YUV 4:1:1
  - Y is 640x480
  - U is 160x120
  - V is 160x120

Image Preparation (Pixel Allocation)

- Each pixel is presented by ‘p’ bits, value is in range of $(0,2^p-1)$
- All pixels of all components within the same image are coded with the same number of bits
- Lossy modes use precision 8 or 12 bits per pixel
- Lossless mode uses precision 2 up to 12 bits per pixel

Image Preparation - Blocks

- Images are divided into data units, called blocks – definition comes from DCT transformation since DCT operates on blocks
- Lossy mode – blocks of 8x8 pixels; lossless mode – data unit 1 pixel

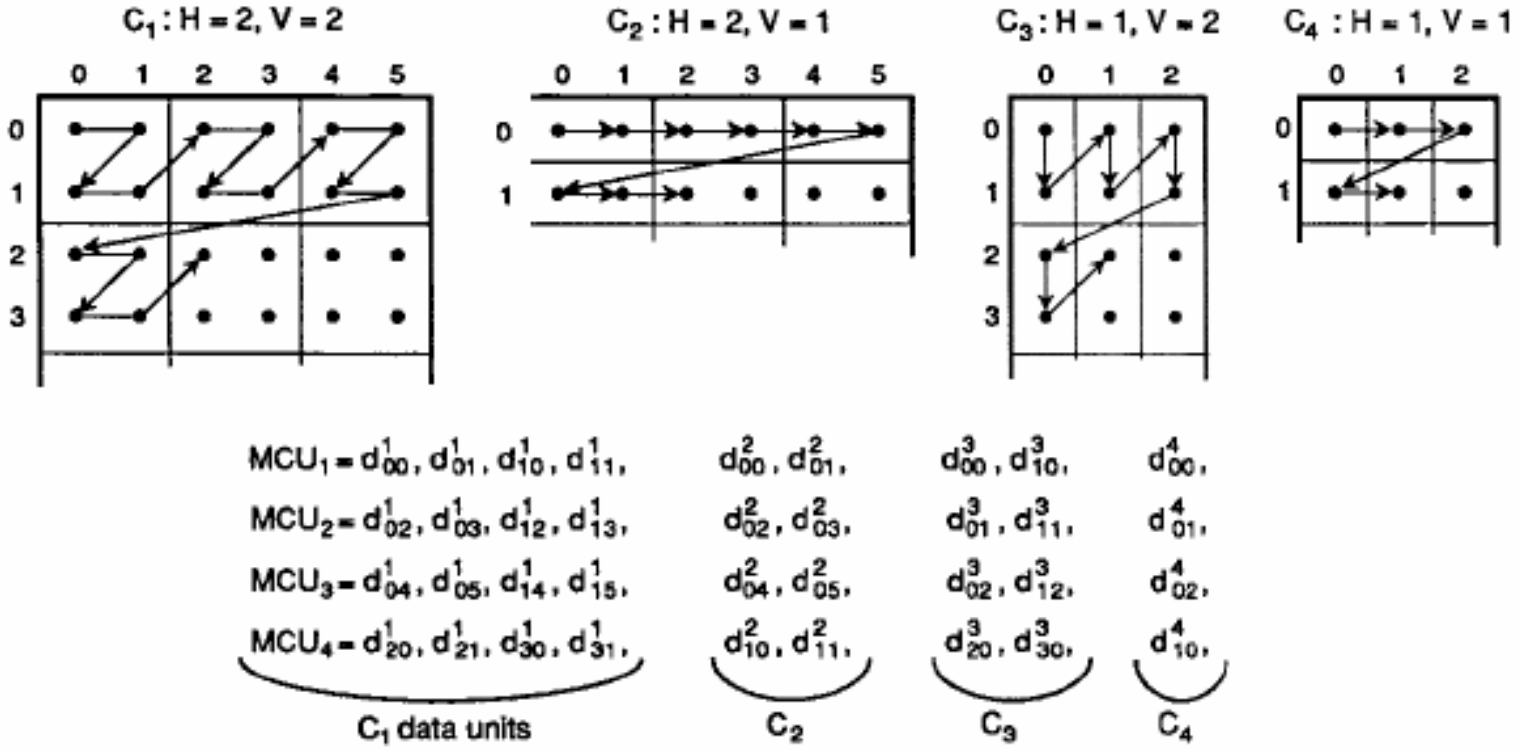
Data Unit Ordering

- *Non-interleaved*: scan from left to right, top to bottom for each color component
- *Interleaved*: compute one “unit” from each color component, then repeat
  - full color pixels after each step of decoding
  - but components may have different resolution

Interleaved Data Ordering

- Interleaved data units of different components are combined into Minimum Coded Units (MCUs)
- If image has the same resolution, then MCU consists of exactly one data unit for each component
- If image has different resolution for each component, reconstruction of MCUs is more complex

Example



[Wallace, 1991]

Image Processing

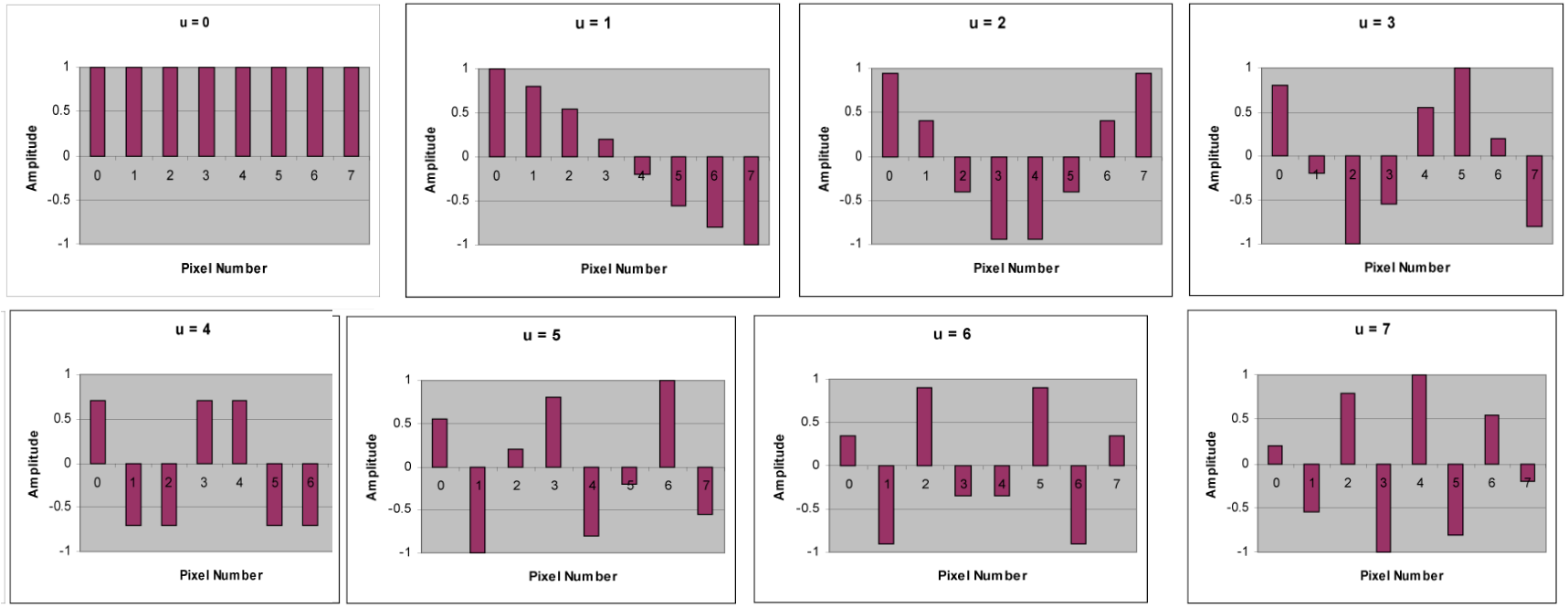
- Shift values $[0, 2^P - 1]$ to $[-2^{P-1}, 2^{P-1} - 1]$
  - e.g. if (P=8), shift [0, 255] to [-127, 127]
  - DCT requires range be centered around 0
- Values in 8x8 pixel blocks are spatial values and there are 64 samples values in each block

Forward DCT

- Convert from spatial to frequency domain
  - convert intensity function into weighted sum of periodic basis (cosine) functions
  - identify bands of spectral information that can be thrown away without loss of quality
- Intensity values in each color plane often change slowly

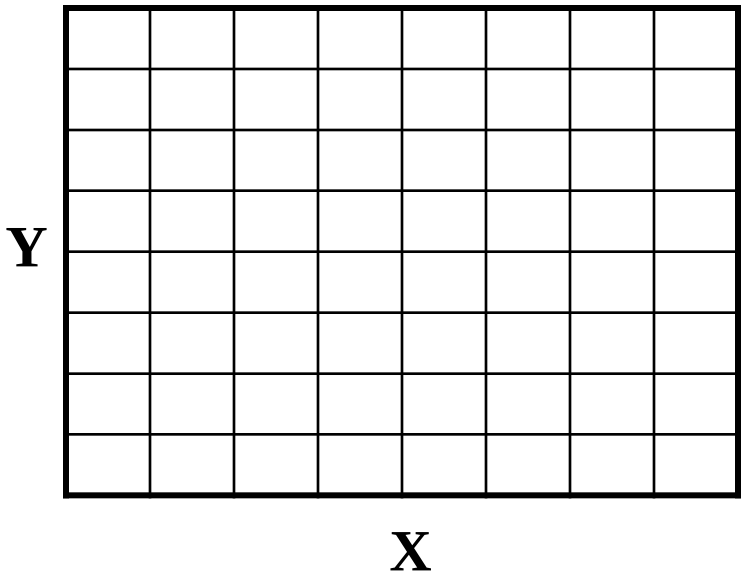
DCT Basic Functions

- Decompose the intensity function into a weighted sum of cosine basis functions



DCT for 2D

- Perform 1D DCT on each row of the block
- Again for each column of 1D coefficients
  - alternatively, transpose the matrix and perform DCT on the rows



Equations for 2D DCT

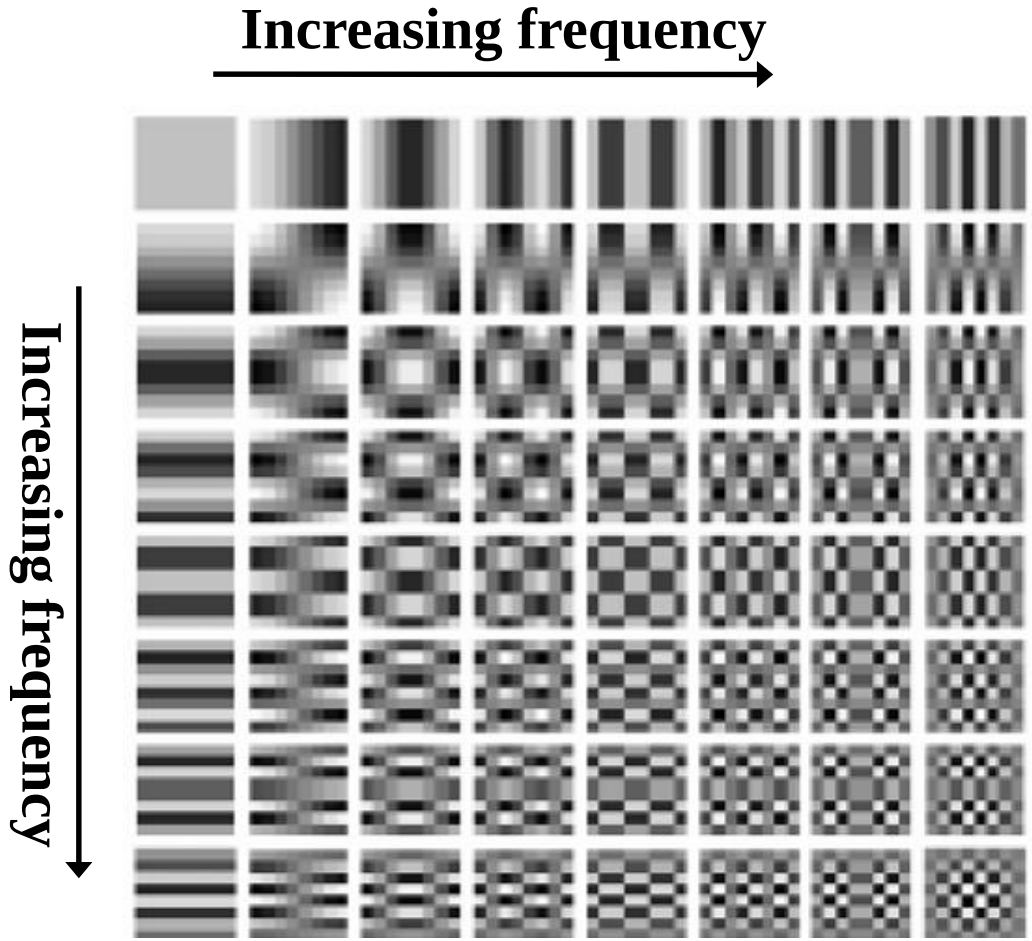
- Forward DCT:

$$F(u,v) = \frac{2}{\sqrt{nm}} C(u)C(v) \sum_{y=0}^{m-1} \sum_{x=0}^{n-1} I(x,y) * \cos\left(\frac{(2x+1)u\pi}{2n}\right) * \cos\left(\frac{(2y+1)v\pi}{2m}\right)$$

- Inverse DCT:

$$I(y,x) = \frac{2}{\sqrt{nm}} \sum_{v=0}^{m-1} \sum_{u=0}^{n-1} F(v,u)C(u)C(v) \cos\left(\frac{(2x+1)u\pi}{2n}\right) * \cos\left(\frac{(2y+1)v\pi}{2m}\right)$$

Visualization of Basis Functions



Coefficient Differentiation

- F(0,0)
  - includes the lowest frequency in both directions
  - is called **DC coefficient**
  - Determines fundamental color of the block
- F(0,1) …. F(7,7)
  - are called **AC coefficients**
  - Their frequency is non-zero in one or both directions

Quantization

- Throw out bits
- Consider example: 101101_2 = 45 (6 bits)
  - We can truncate this string to 4 bits: 1011_2 = 11
  - We can truncate this string to 3 bits: 101_2 = 5 (original value 40) or 110_2 = 6 (original value 48)
- Uniform quantization is achieved by dividing DCT coefficients by N and round the result (e.g., above we used N=4 or N=8)
- In JPEG – use quantization tables
  - Fq(u,v) = F(u,v)/Quv
  - Two quantization tables – one for luminance and one for two chrominance components

De facto Quantization Table

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

Eye becomes less sensitive

Eye becomes less sensitive

Entropy Encoding

- Compress sequence of quantized DC and AC coefficients from quantization step
  - further increase compression, without loss
- Separate DC from AC components
  - DC components change slowly, thus will be encoded using difference encoding

DC Encoding

- DC represents average intensity of a block
  - encode using difference encoding scheme
  - use 3x3 pattern of blocks

- Because difference tends to be near zero, can use less bits in the encoding
  - categorize difference into difference classes
  - send the index of the difference class, followed by bits representing the difference

Difference Coding applied to DC Coefficients

PREDICTOR

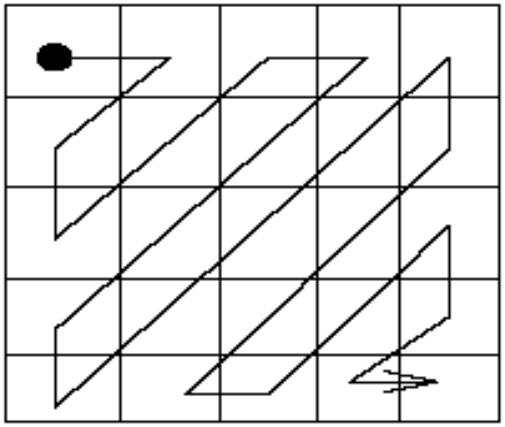
$$Diff_i = DC_i - DC_{i-1} \quad i>0$$

DC_0	DC_1	DC_2
DC_3	DC_4	DC_5
DC_6	DC_7	DC_8

DC_0	$Diff_1$	$Diff_2$
$Diff_3$	$Diff_4$	$Diff_5$
$Diff_6$	$Diff_7$	$Diff_8$

AC Encoding

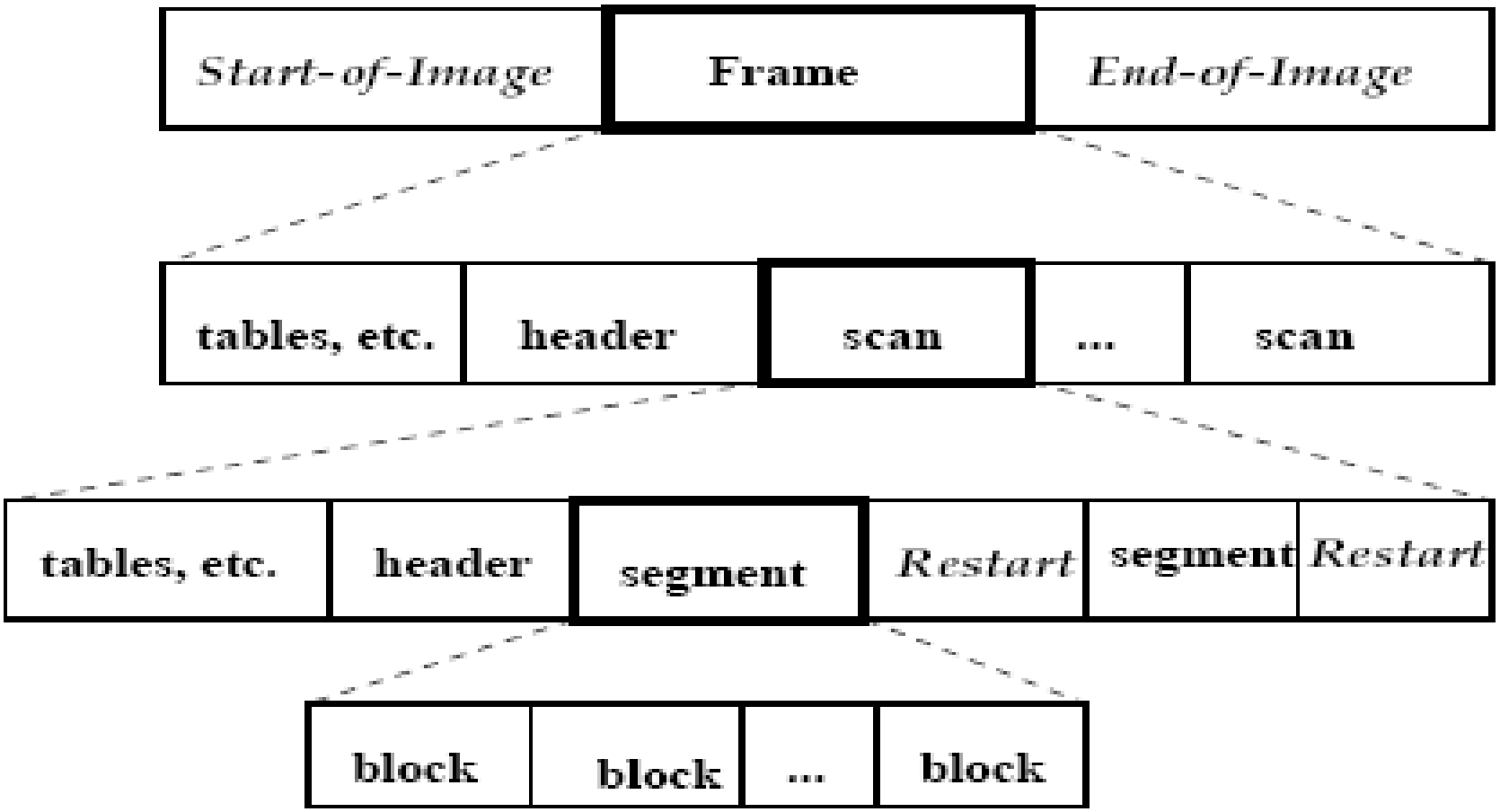
- Use zig-zag ordering of coefficients
  - orders frequency components from low->high
  - produce maximal series of 0s at the end
  - Ordering helps to apply efficiently entropy encoding
- Apply Huffman coding
- Apply RLE on AC zero values



Huffman Encoding

- Sequence of DC difference indices and values along with RLE of AC coefficients
- Apply Huffman encoding to sequence
- Attach appropriate headers
- Finally have the JPEG image!

Interchange Format of JPEG



ADDITIONAL SLIDES

Huffman Example

- Construct the Huffman coding tree (in class)

Symbol (S)	*P*(S)
A	0.25
B	0.30
C	0.12
D	0.15
E	0.18

Characteristics of Solution

Symbol (S)	Code
A	01
B	11
C	100
D	101
E	00

Example Encoding/Decoding

Encode “BEAD”

⇒ 110001101

⇒ Decode “0101100”

Symbol (S)	Code
A	01
B	11
C	100
D	101
E	00

Entropy (Theoretical Limit)

$$H = \sum_{i=1}^{N} - p(s_i) \log_2 p(s_i)$$

= -.25 * $\log_2$ .25 +
-.30 * $\log_2$ .30 +
-.12 * $\log_2$ .12 +
-.15 * $\log_2$ .15 +
-.18 * $\log_2$ .18

H = 2.24 bits

Symbol	*P*(S)	Code
A	0.25	01
B	0.30	11
C	0.12	100
D	0.15	101
E	0.18	00

Average Codeword Length

$$L = \sum_{i=1}^{N} p(s_i)codelength(s_i)$$

= .25(2) +
.30(2) +
.12(3) +
.15(3) +
.18(2)

L = 2.27 bits

Symbol	*P*(S)	Code
A	0.25	01
B	0.30	11
C	0.12	100
D	0.15	101
E	0.18	00

Code Length Relative to Entropy

$$L = \sum_{i=1}^{N} p(s_i) codelength(s_i) \qquad H = \sum_{i=1}^{N} -p(s_i) \log_2 p(s_i)$$

- Huffman reaches entropy limit when all probabilities are negative powers of 2
  - i.e., 1/2; 1/4; 1/8; 1/16; etc.

- H <= Code Length <= H + 1

Example

$H = -.01*\log_2.01 + -.99*\log_2.99$

$= .08$

$L = .01(1) + .99(1)$

$= 1$

Symbol	*P*(S)	Code
A	0.01	1
B	0.99	0

Group Exercise

- Compute Entropy (H)
- Build Huffman tree
- Compute average code length
- Code “BCCADE”

Symbol (S)	*P*(S)
A	0.1
B	0.2
C	0.4
D	0.2
E	0.1

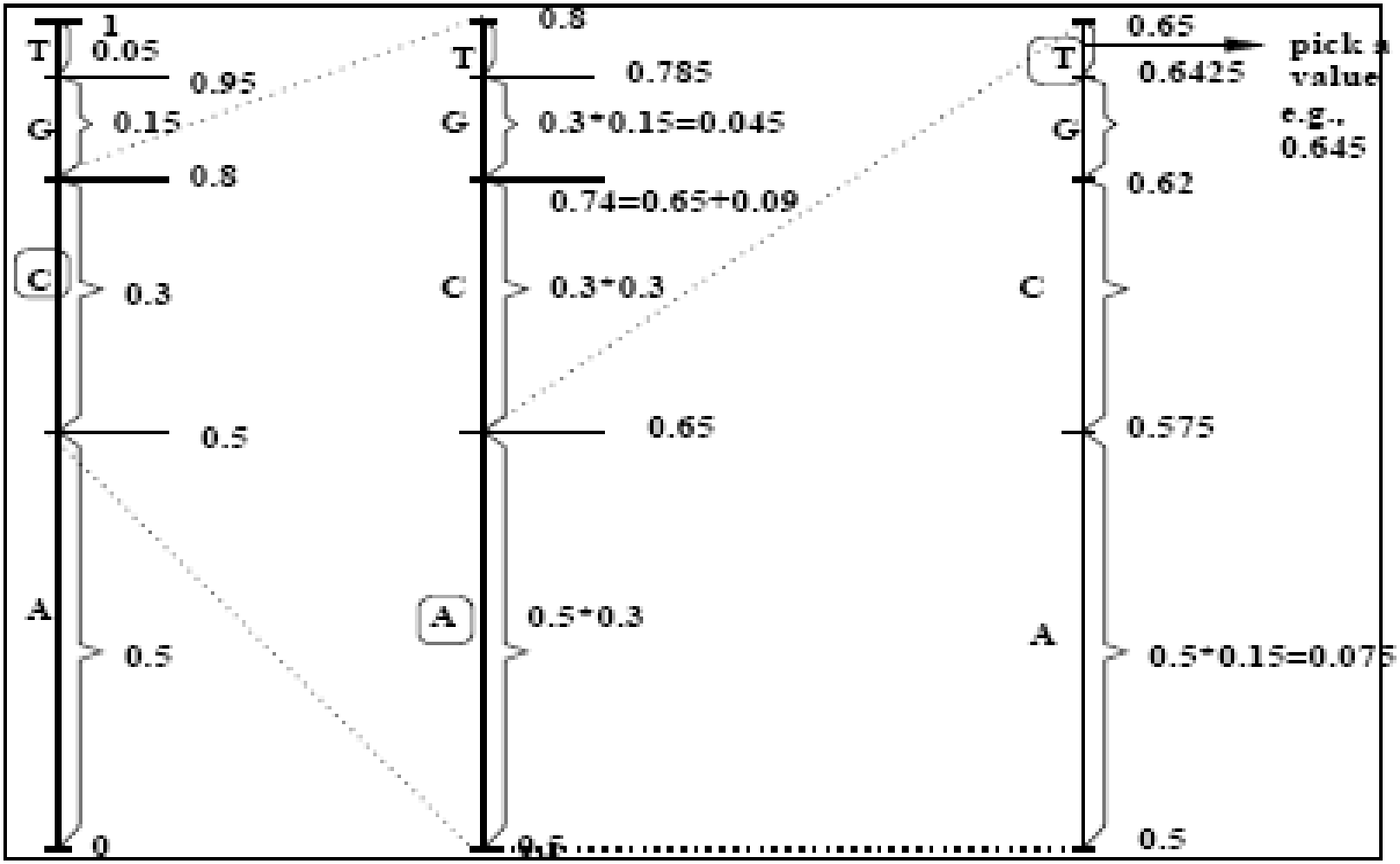
Arithmetic Coding

- Optimal algorithm as Huffman coding wrt compression ratio
- Better algorithm than Huffman wrt transmitted amount of information
  - Huffman – needs to transmit Huffman tables with compressed data
  - Arithmetic – needs to transmit length of encoded string with compressed data

Arithmetic Coding

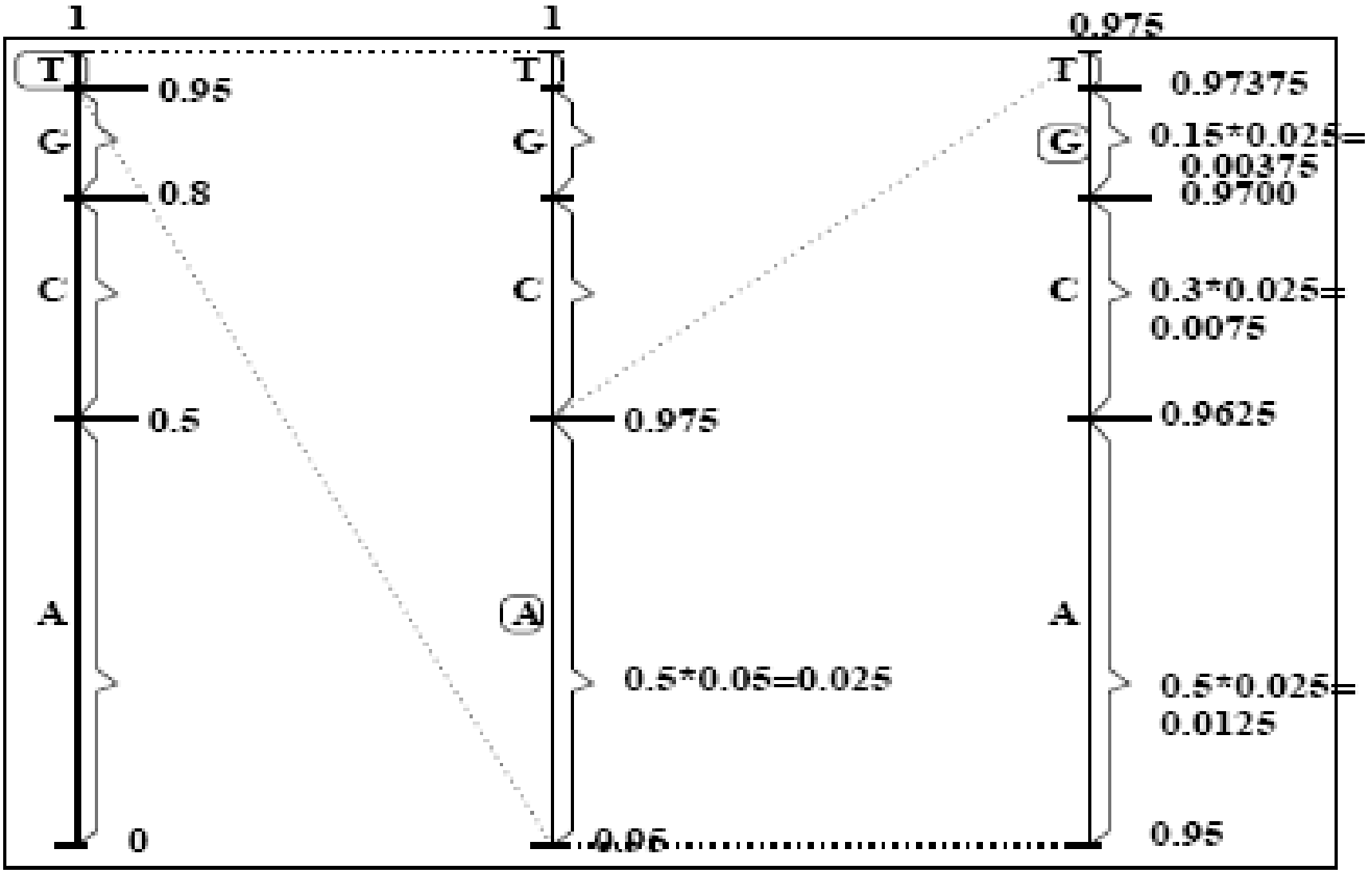
- Each symbol is coded by considering the prior data
- Encoded data must be read from the beginning, there is no random access possible
- Each real number (< 1) is represented as binary fraction
  - $0.5 = 2^{-1}$ (binary fraction = 0.1); $0.25 = 2^{-2}$ (binary fraction = 0.01), 0.625 = 0.5+0.125 (binary fraction = 0.101) ….

P(A)=0.5, P(C) = 0.3, P(G) = 0.15, P(T) = 0.05 => Encode CAT



Send Value: 0.645

Given: P(A)=0.5, P(C) = 0.3, P(G) = 0.15, P(T) = 0.05 => Decode: 0.9715



Decoded Word: TAG