CS 414 – Multimedia Systems Design

Lecture 8 – JPEG Compression (Part 3)

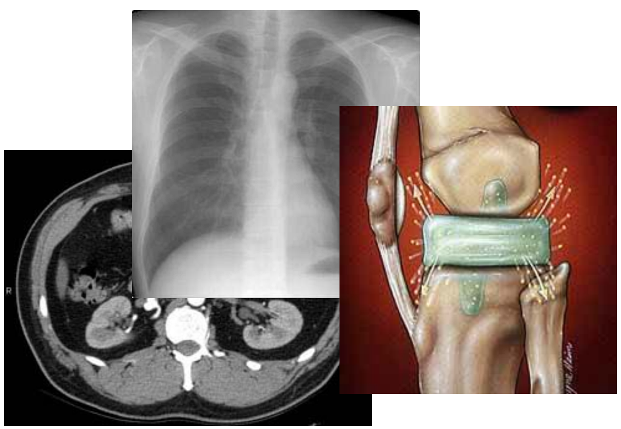
Klara Nahrstedt

Spring 2009

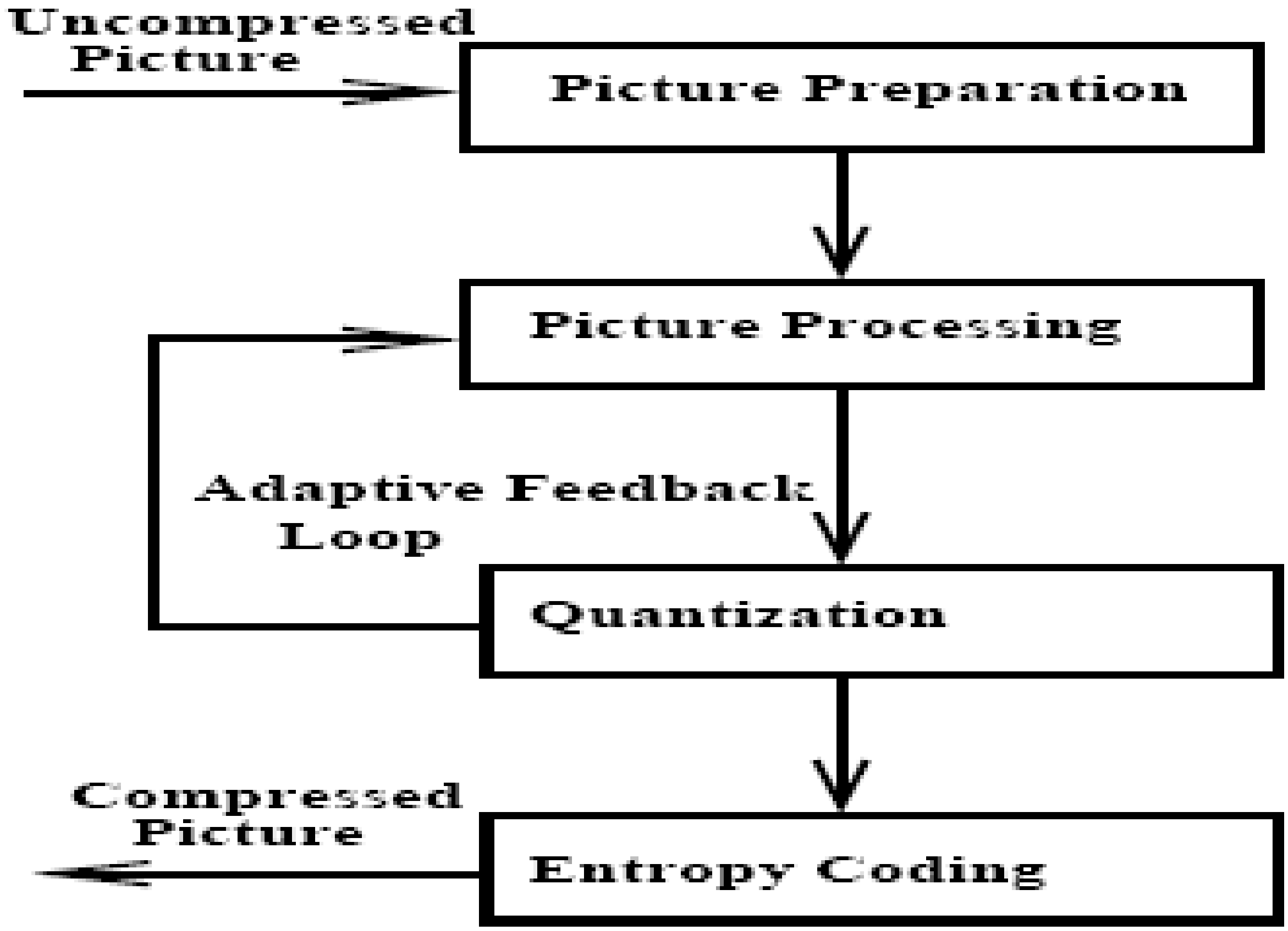
Administrative

- MP1 is posted, deadline Monday, February 9, demonstrations 5-7pm in 0216 SC

Ubiquitous use of digital images

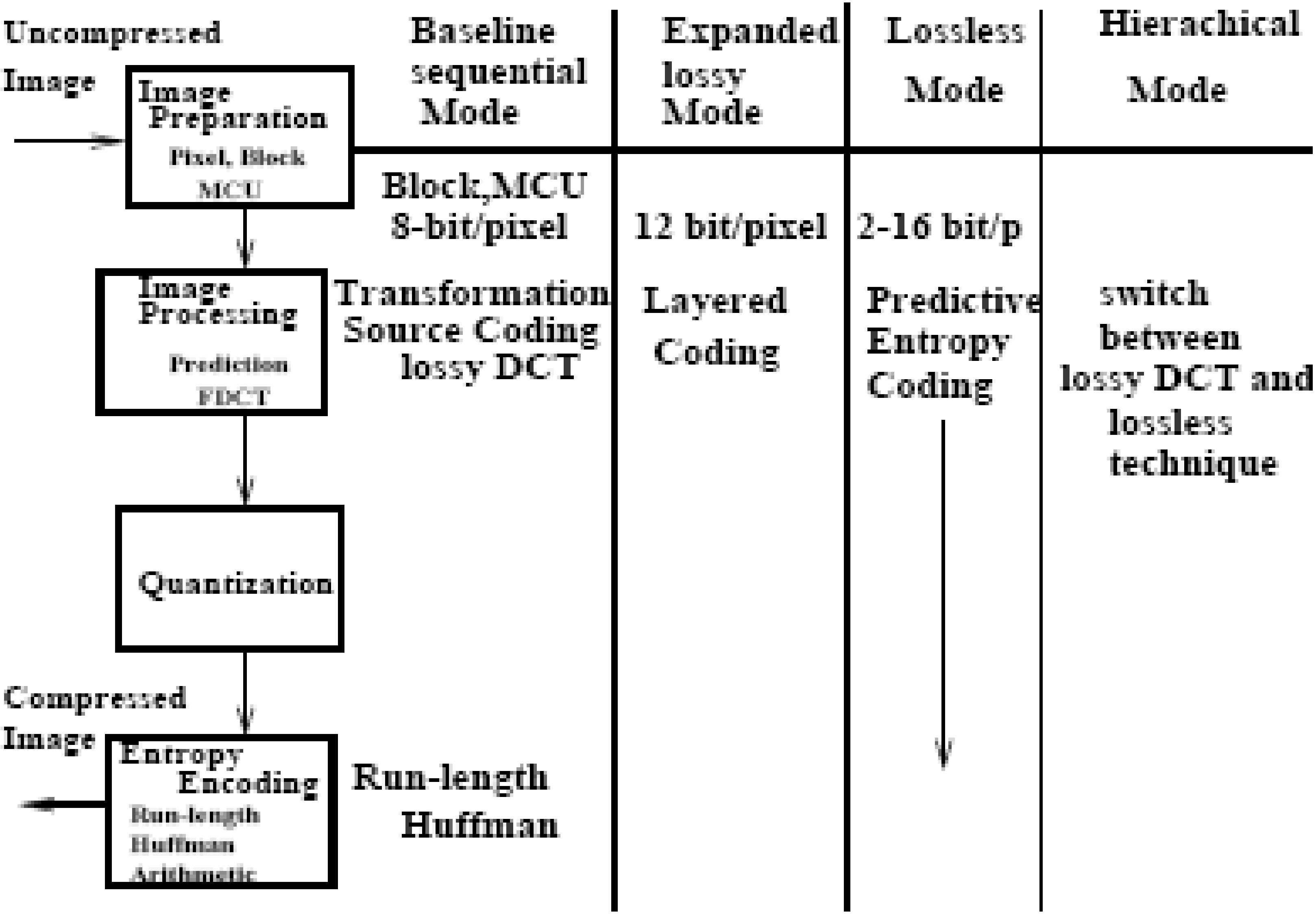


Hybrid Coding



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)


Uncompressed Image
Image Preparation
Pixel, Block
MCU
Image Processing
Prediction
FDCT
Quantization
Compressed Image
Entropy Encoding
Run-length
Huffman
Arithmetic
Baseline sequential Mode
Expanded lossy Mode
Lossless Mode
Hierachical Mode
Block,MCU
8-bit/pixel
12 bit/pixel
2-16 bit/p
Transformation
Source Coding
lossy DCT
Layered Coding
Predictive Entropy Coding
switch between lossy DCT and lossless technique
Run-length
Huffman

JPEG Compression

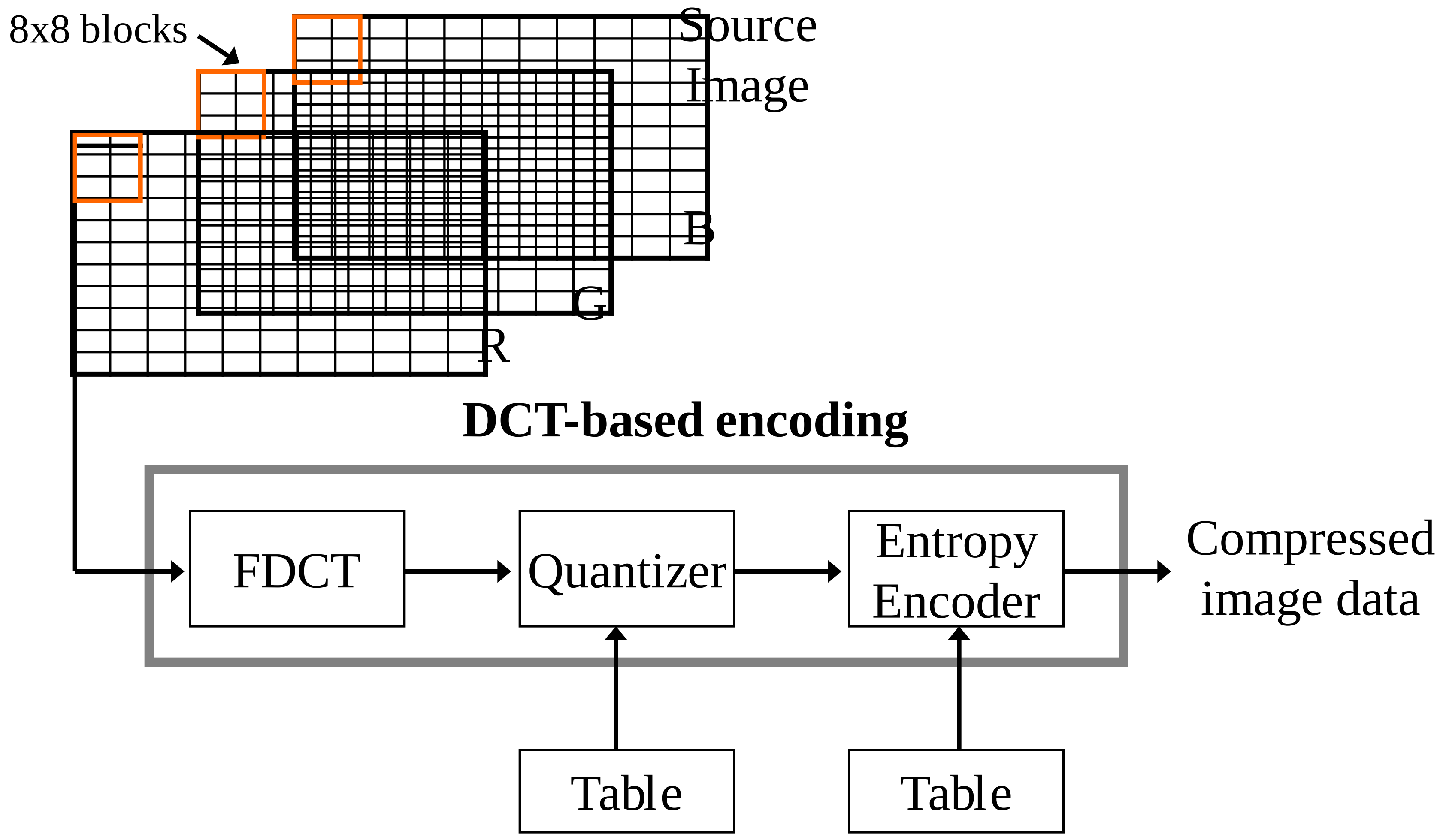
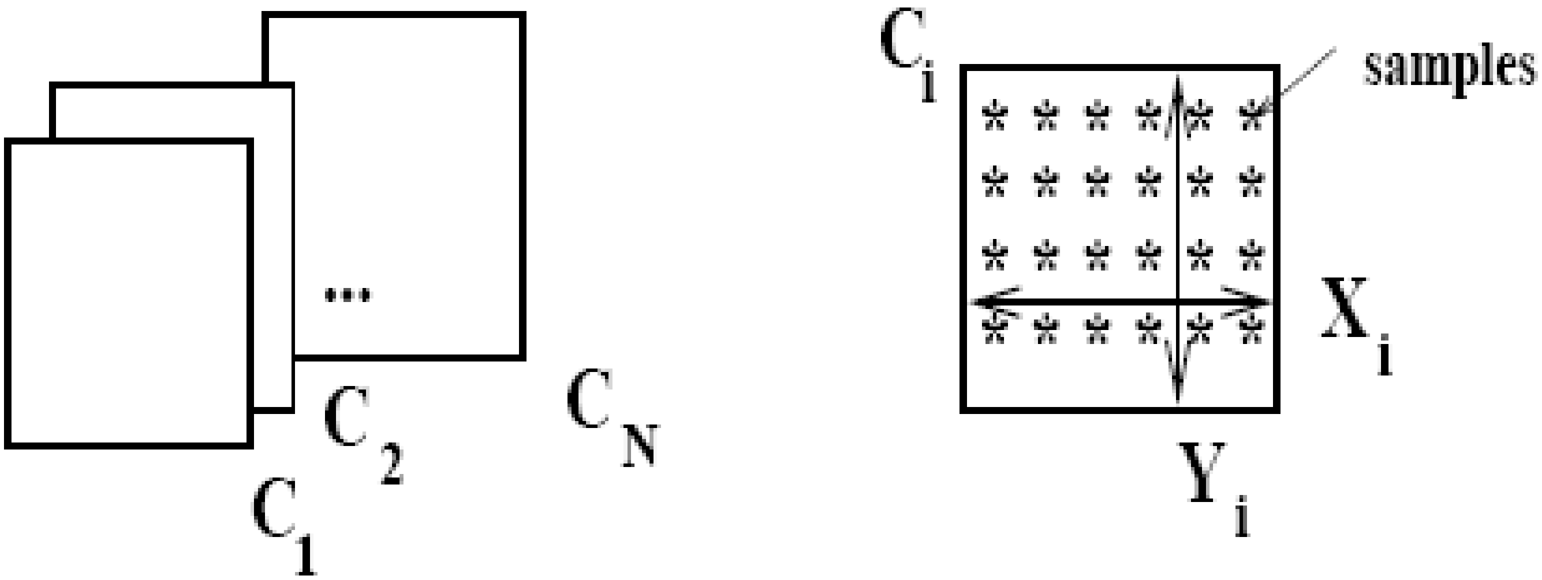


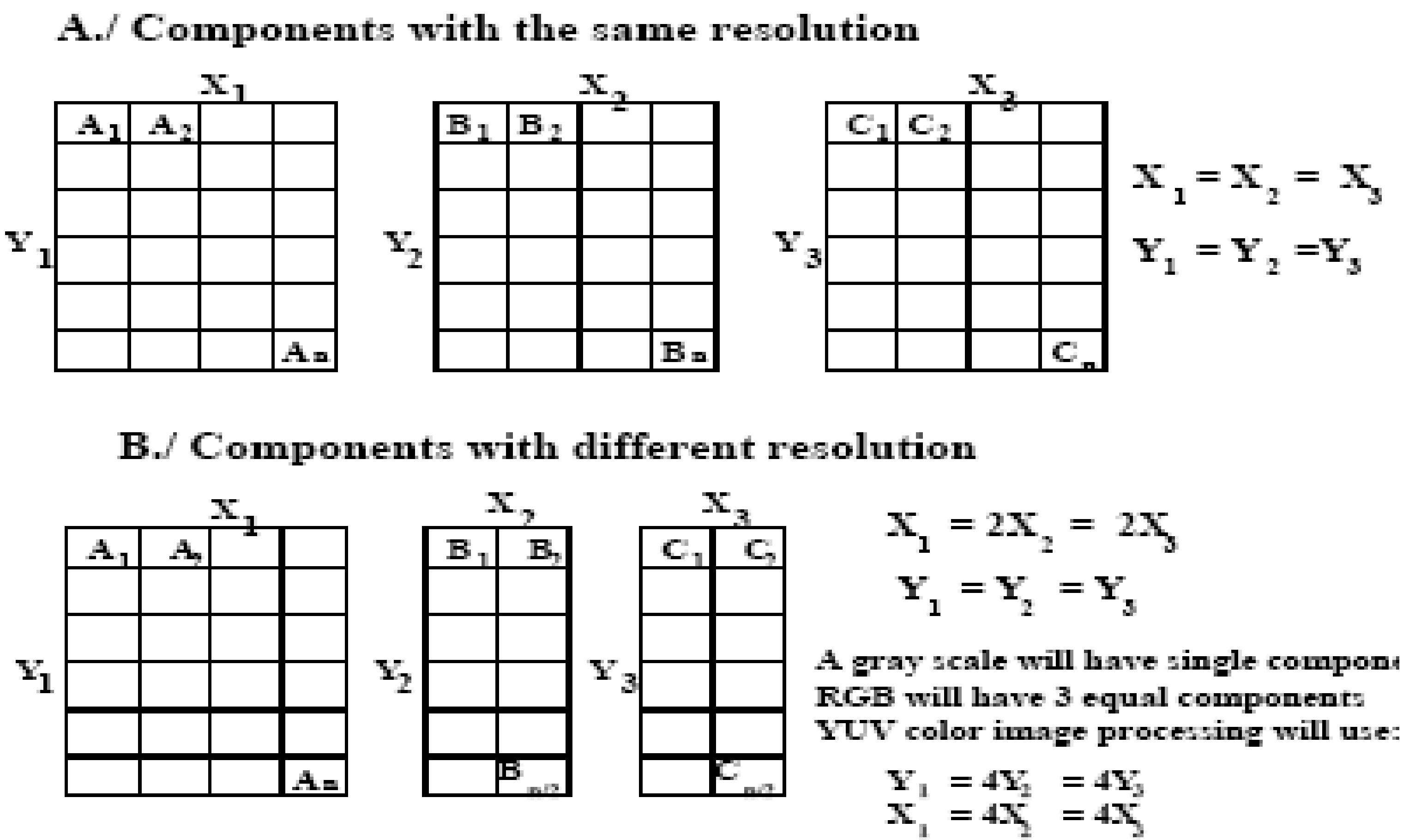
Image Preparation

- 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



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

Dimensions of Compressed Image

- i_{th} color component has dimension (x_i, y_i)
  - maximum dimension value is 2^{16}
  - [X, Y] where $X=\max(x_i)$ and $Y=\max(y_i)$
- Sampling among components must be integral
  - H_i and V_i; must be within range [1, 4]
  - $[H_{max}, V_{max}]$ where $H_{max}=\max(H_i)$ and $V_{max}=\max(V_i)$

- $x_i = X * H_i / H_{max}$
- $y_i = Y * V_i / V_{max}$

Dimensions (Example)

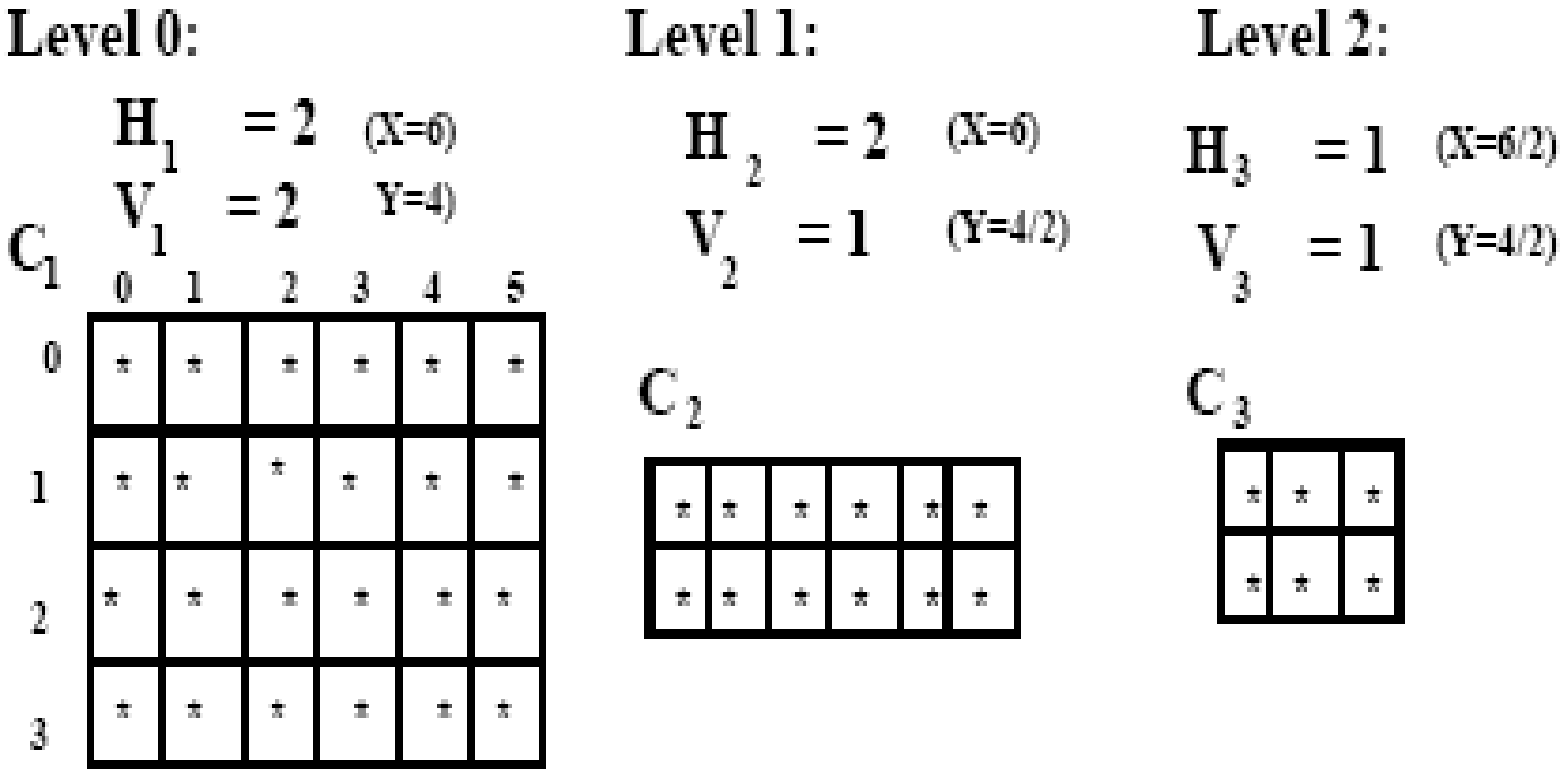


Image Preparation (Pixel Allocation)

- 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

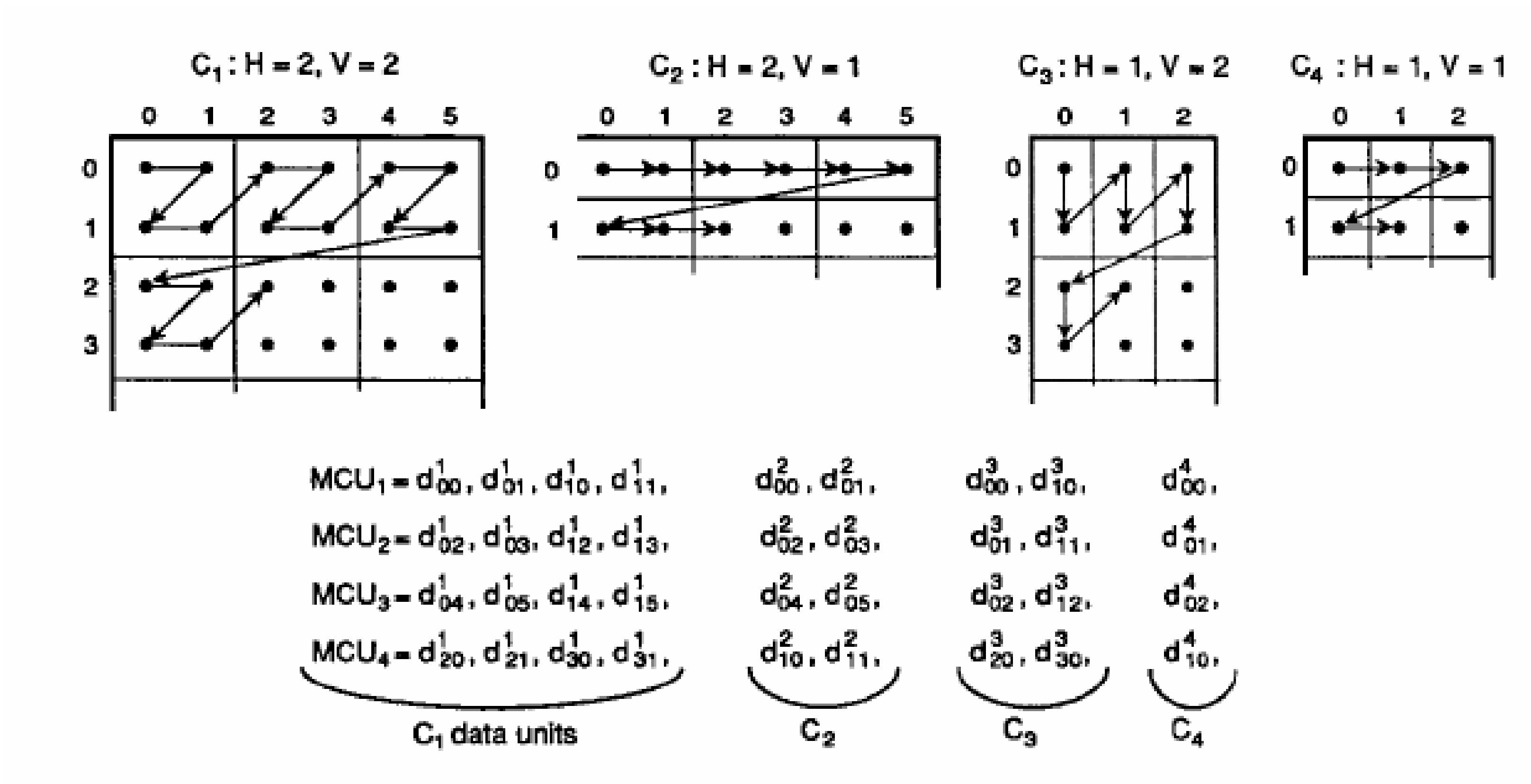
Data Unit Ordering

- *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

Understanding DCT

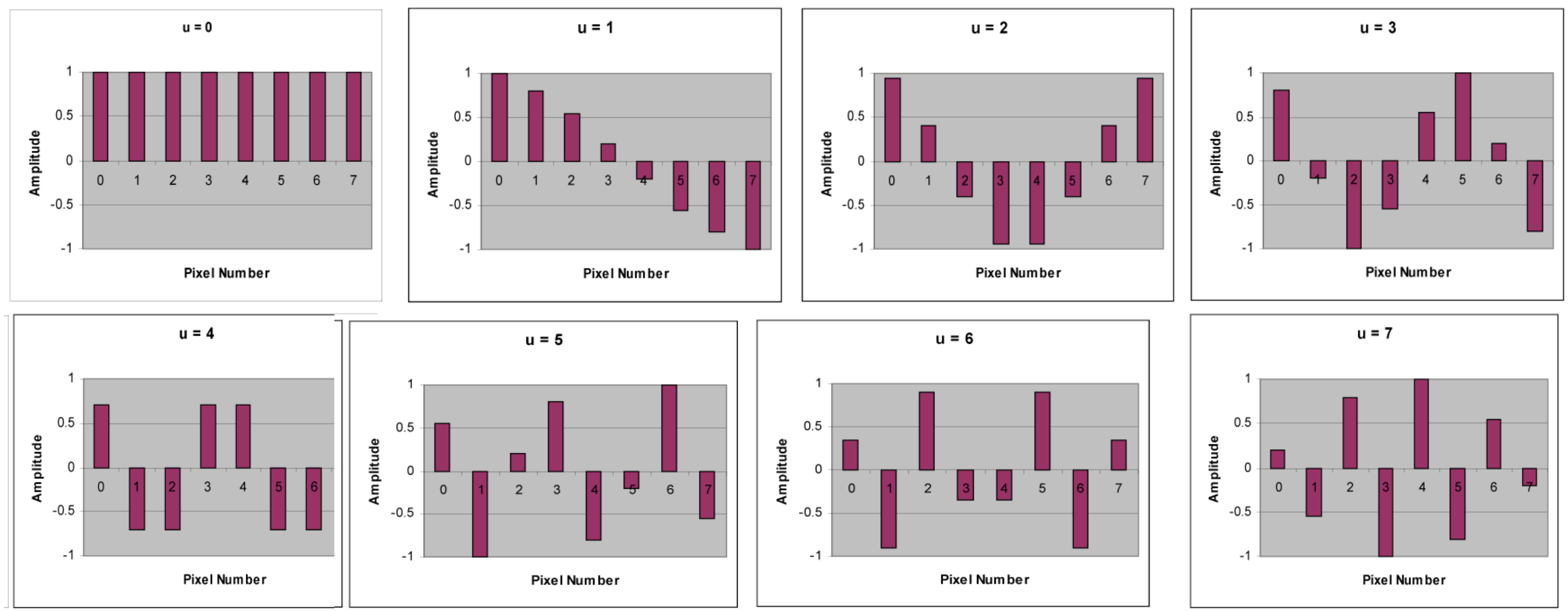
- For example, in R^3, we can write (5, 2, 9) as the sum of a set of basis vectors
  - we know that [(1,0,0), (0,1,0), (0,0,1)] provides one set of basis functions in R^3

$$(5,2,9) = 5*(1,0,0) + 2*(0,1,0) + 9*(0,0,1)$$

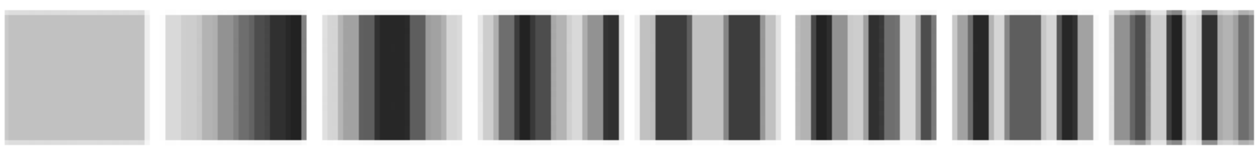
- DCT is same process in function domain

DCT Basic Functions

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



Alternative Visualization



1D Forward DCT

- Given a list of *n* intensity values I(x), where x = 0, …, n-1
- Compute the *n* DCT coefficients:

$$F(u) = \sqrt{\frac{2}{n}} \quad C(u) \sum_{x=0}^{n-1} I(x) \cos \frac{(2x+1)\mu\pi}{2n}, u = 0 \ldots n-1$$

$$\textit{where} \qquad C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \textit{for} \quad u = 0, \\ 1 & \textit{otherwise} \end{cases}$$

1D Inverse DCT

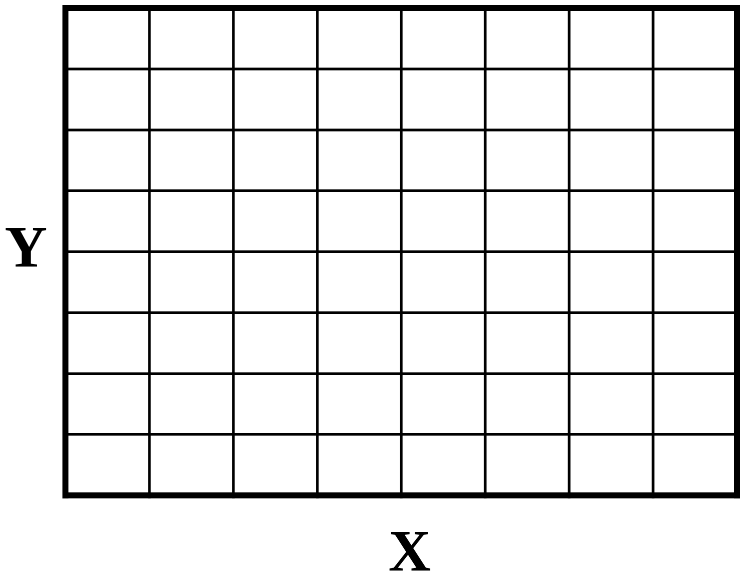
- Given a list of *n* DCT coefficients F(u), where u = 0, …, n-1
- Compute the *n* intensity values:

$$I(x) = \sqrt{\frac{2}{n}} \sum_{u=0}^{n-1} F(u) C(u) \cos \frac{(2x+1)\mu\pi}{2n}, x = 0 \ldots n-1$$

$$where \qquad C(u) = \begin{cases} \frac{1}{\sqrt{2}} & for \quad u = 0, \\ 1 & otherwise \end{cases}$$

Extend DCT from 1D to 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

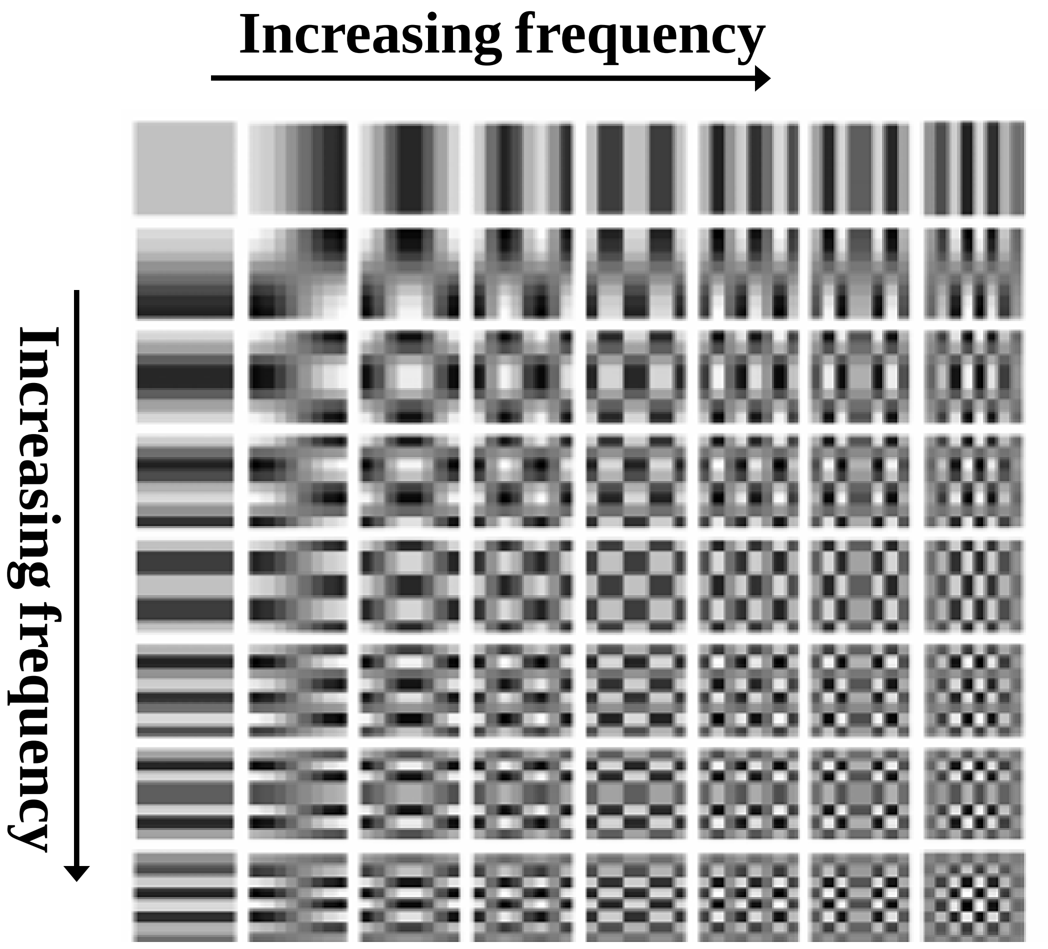
- Forward DCT:

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

- 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

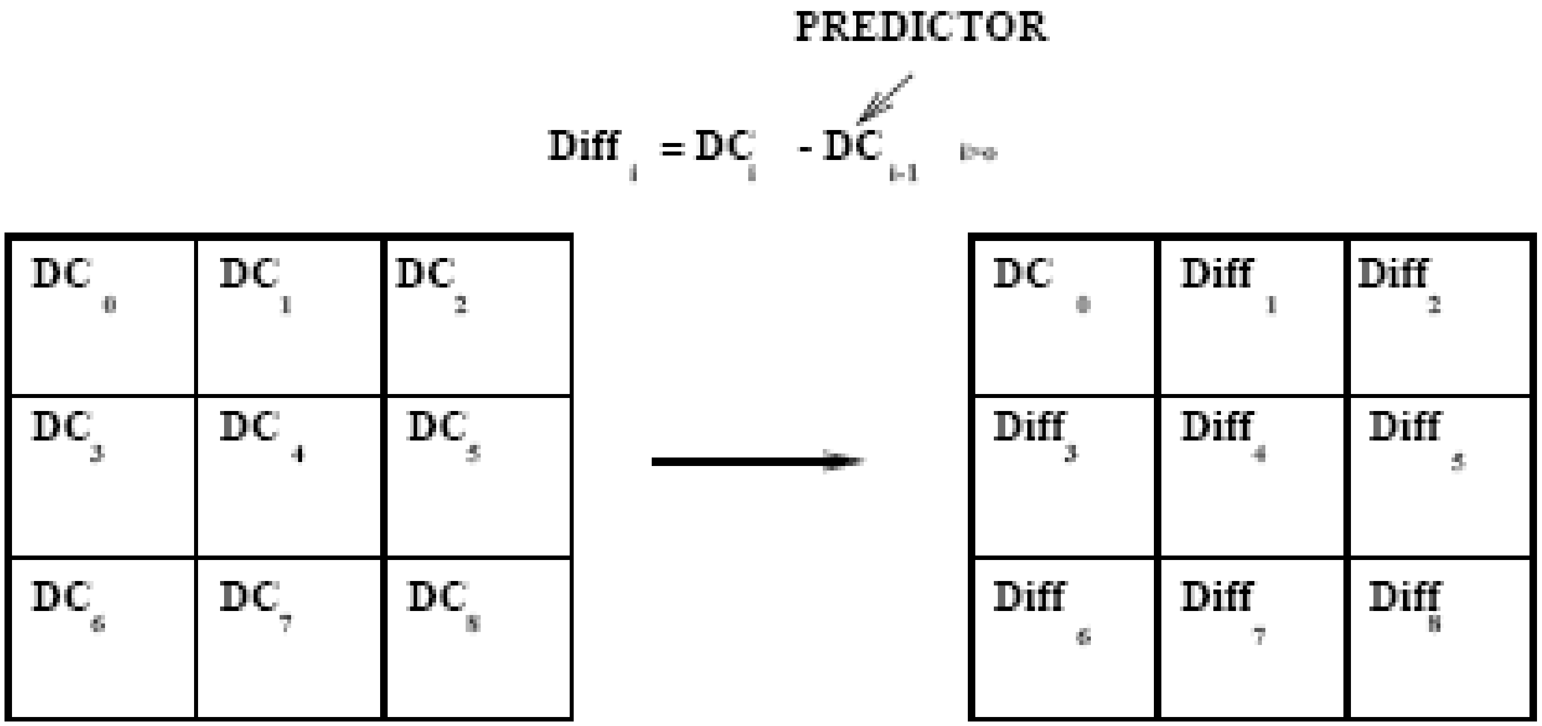
Entropy Encoding

- 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

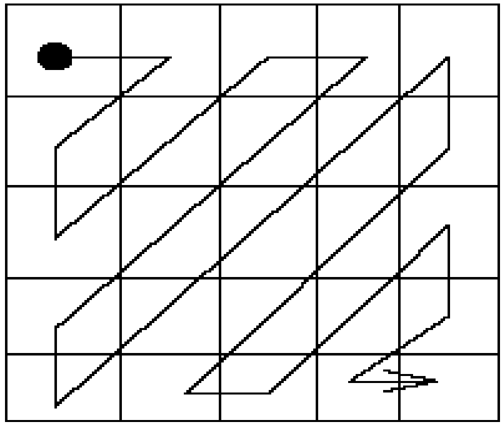
- DC represents average intensity of a block
  - encode using difference encoding scheme
  - use 3x3 pattern of blocks
- Because difference tends to be near zero, can use less bits in the encoding
  - categorize difference into difference classes
  - send the index of the difference class, followed by bits representing the difference

Difference Coding applied to DC Coefficients



AC Encoding

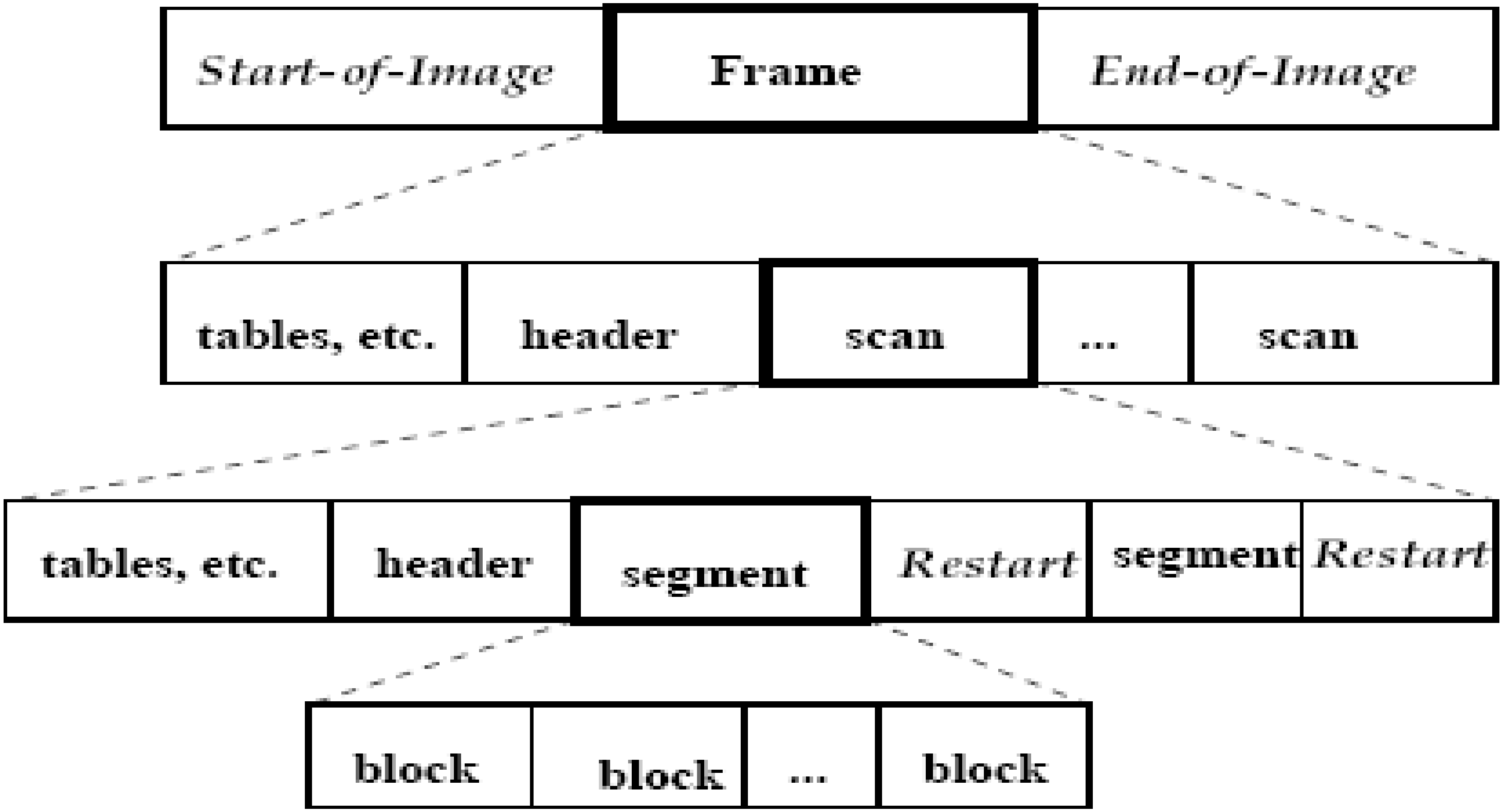
- Use zig-zag ordering of coefficients
  - orders frequency components from low->high
  - produce maximal series of 0s at the end
  - Ordering helps to apply efficiently entropy encoding
- Apply Huffman coding
- Apply RLE on AC zero values



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



Example - One Everyday Photo

2.76M

Example - One Everyday Photo

600K

Example - One Everyday Photo

350K

Example - One Everyday Photo

240K

Example - One Everyday Photo

144K

Example - One Everyday Photo

88K

Discussion

- What types of image content would JPEG work best (worst) for?
- Is image compression solved?
- What’s missing from JPEG?