JPEG

Introduction

- The name JPEG stands for Joint Photographic Expert Group, the name of the committee that created the JPEG standard.
  - www.jpeg.org
- Has published several standards
  - JPEG: lossy coding of still images
    - Based on DCT
  - JPEG2000: scalable coding of still images (from lossy to lossless)
    - Based on wavelet transform

The 1992 JPEG Standard

- Contains several modes:
  - Baseline system (what is commonly known as JPEG!):
    - Can handle gray scale or color images, with 8 bits per color component
  - Extended system: can handle higher precision (12 bit) images, providing progressive streams, etc.
  - Lossless version
- Baseline version
  - Each color component is divided into 8x8 blocks
  - For each 8x8 block, three steps are involved:
    - Block DCT
    - Perceptual-based quantization
    - Variable length coding: Runlength and Huffman coding

Baseline JPEG and Preprocessing

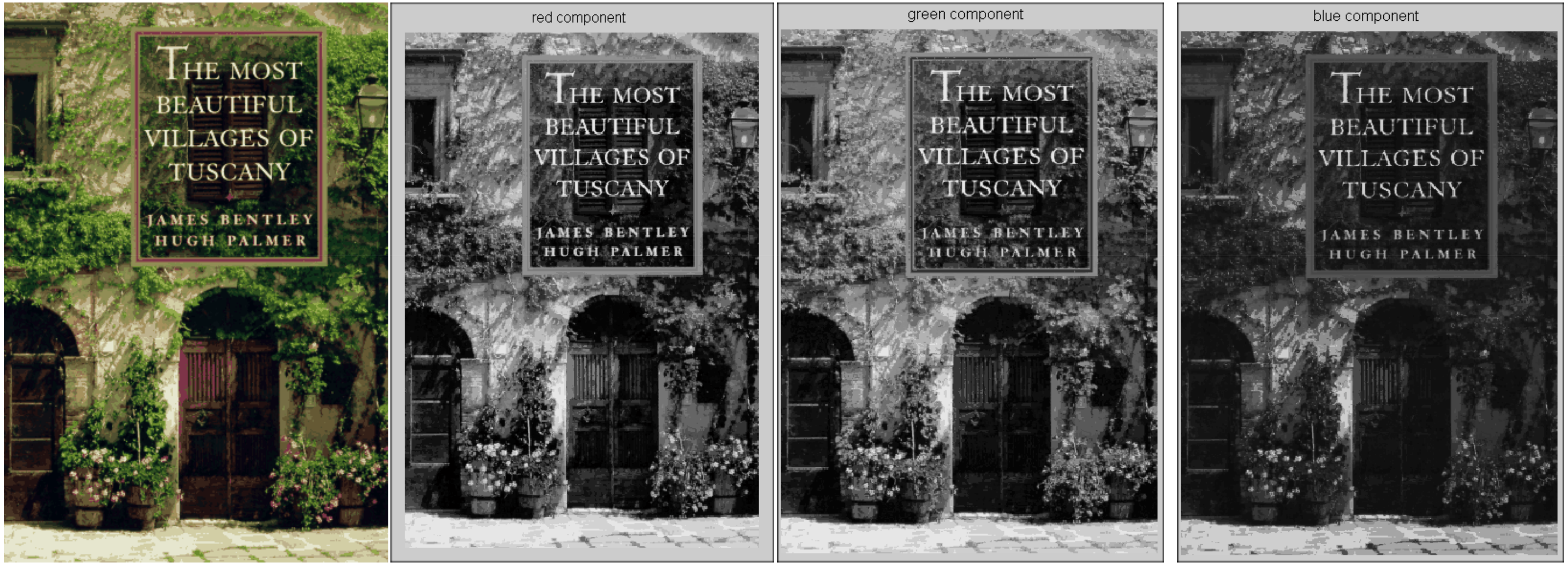
- A DCT-based sequential mode
- Preprocessing: Color transformation (RGB to YCbCr / YUV)

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

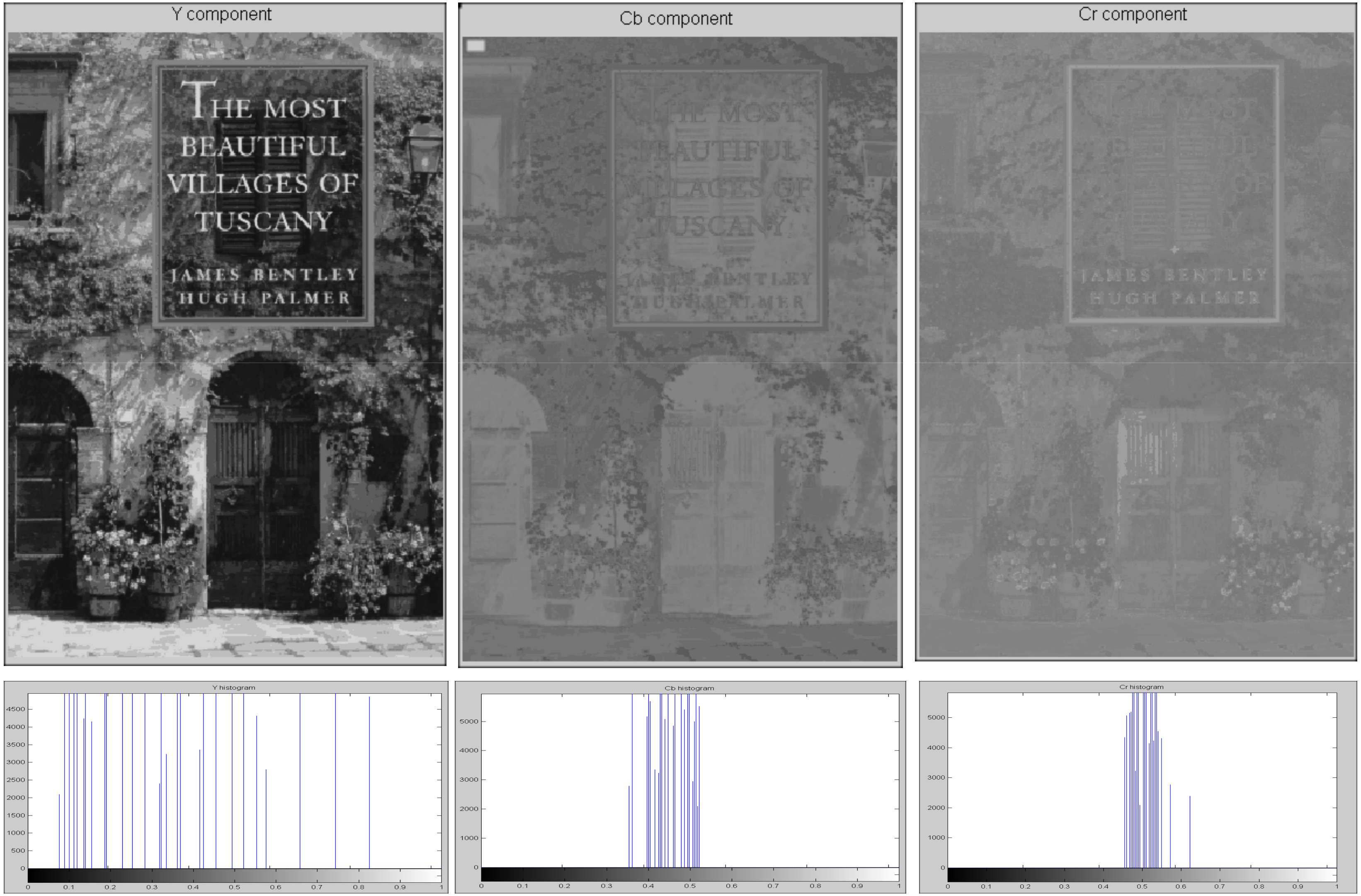
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

Note: Cb ~ Y-B, Cr ~ Y-R, are known as color difference signals.

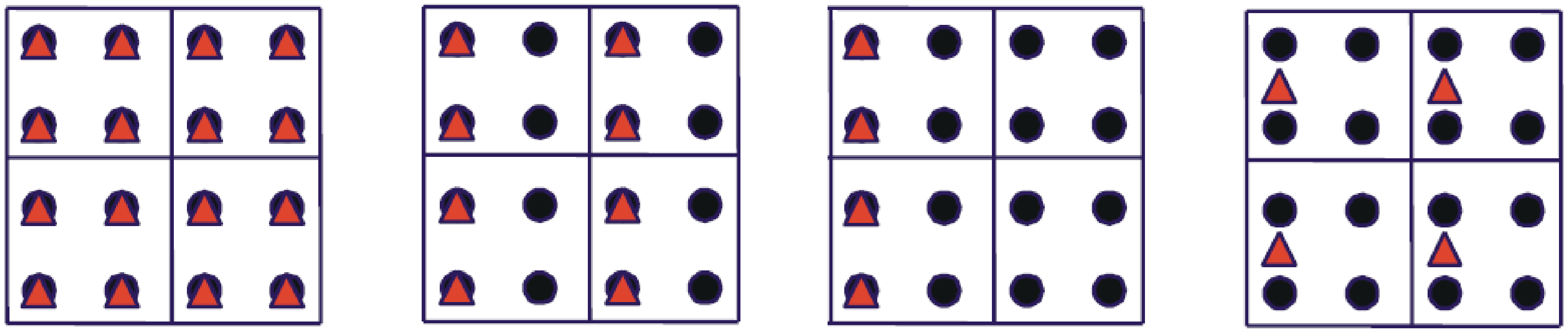
RGB Component



YCbCr Component



Chrominance Subsampling



4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)

4:2:2
For every 2x2 Y Pixels
2 Cb & 2 Cr Pixel
(Subsampling by 2:1
horizontally only)

4:1:1
For every 4x1 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 4:1
horizontally only)

4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both
horizontally and vertically)

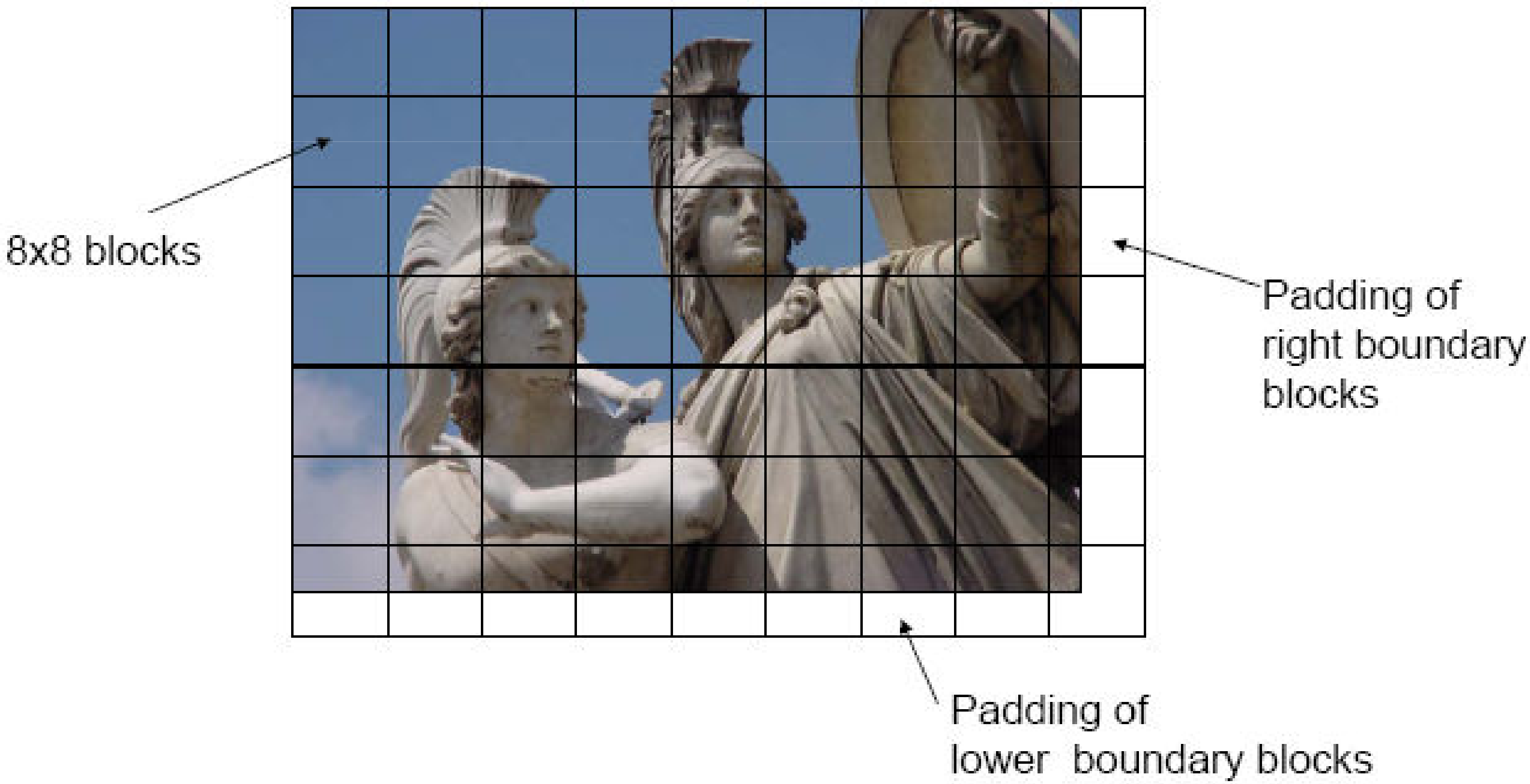
● Y Pixel

▲ Cb and Cr Pixel

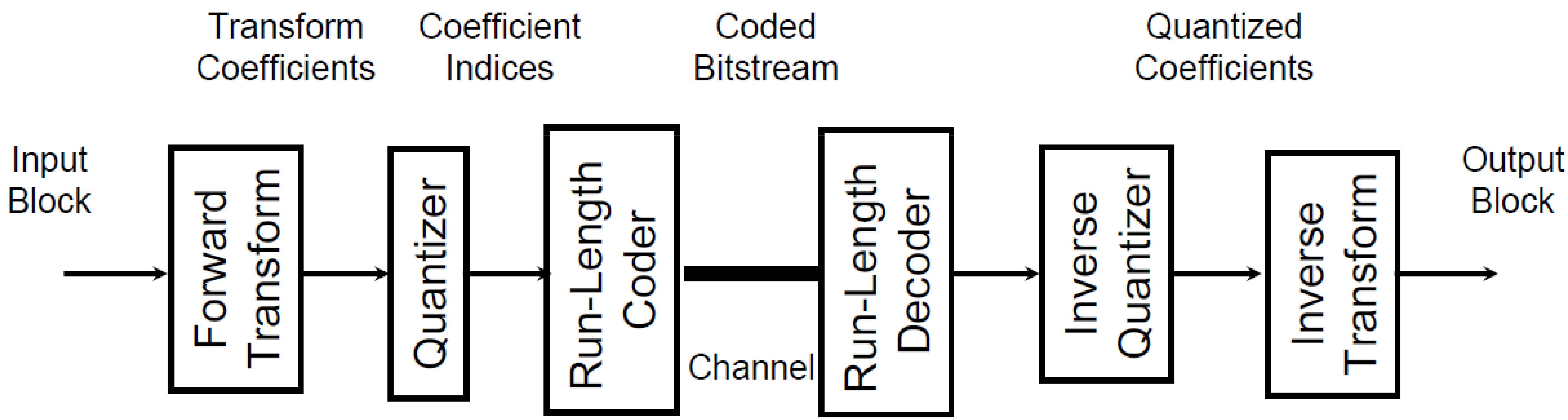
4:2:0 is the most common format

Block Splitting

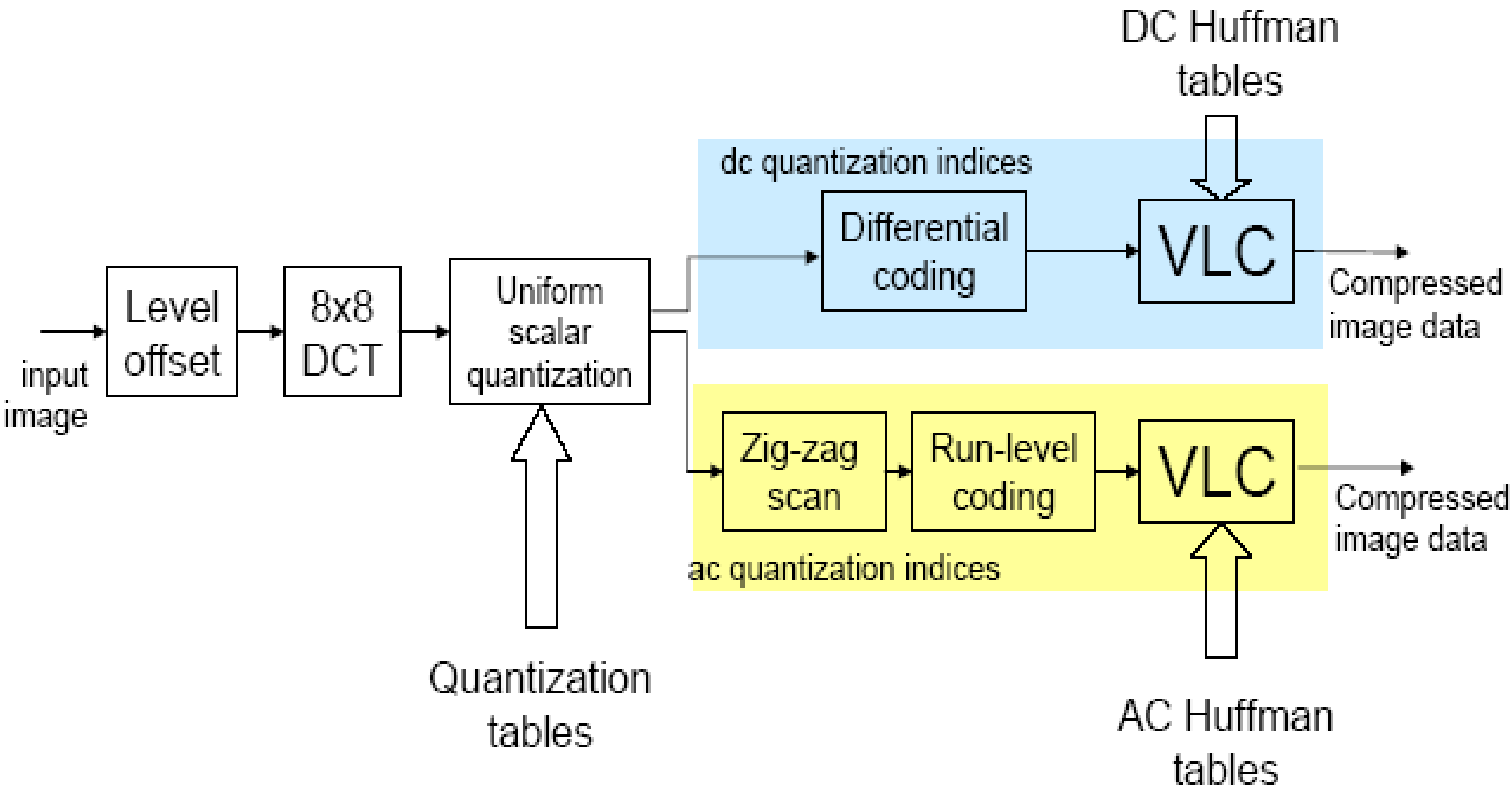
8x8 block size represents a compromise between a block size small enough to minimize storage and processing overheads, but large enough to effectively exploit image redundancy.



Encoder/Decoder Block Diagram



Baseline JPEG Encoder



- Shift to zero-mean by subtracting 128 → [-128, 127]

2D Discrete Cosine Transform

DEFINITION 4.3-1 *(2-D DCT)*
Assume that the data array has finite rectangular support on $[0, N_1 - 1] \times [0, N_2 - 1]$, then the 2-D DCT is given as

$$X_C(k_1, k_2) \triangleq \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 4x(n_1, n_2) \cos \frac{\pi k_1}{2N_1}(2n_1 + 1) \cos \frac{\pi k_2}{2N_2}(2n_2 + 1), \tag{4.3-1}$$

for $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. Otherwise, $X_C(k_1, k_2) \triangleq 0$.

The inverse DCT exists and is given for $(n_1, n_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$ as

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w(k_1) w(k_2) X_C(k_1, k_2) \times \cos \frac{\pi k_1}{2N_1}(2n_1 + 1) \cos \frac{\pi k_2}{2N_2}(2n_2 + 1),$$

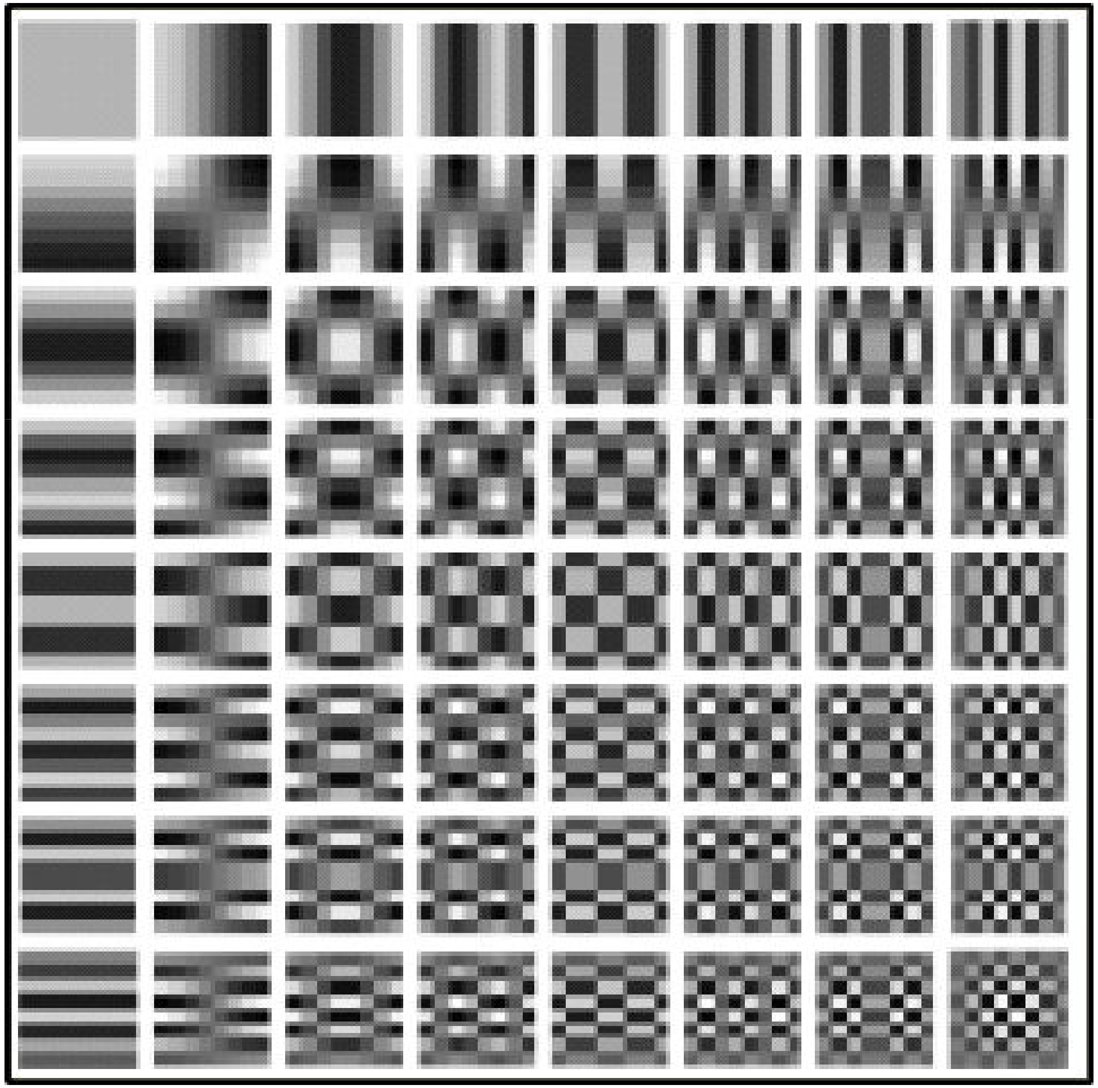
where the weighting function $w(k)$ is given just as in the case of the 1-D DCT by

$$w(k) \triangleq \begin{cases} 1/2, & k = 0 \\ 1, & k \neq 0. \end{cases}$$

Basic Images of 8x8 DCT

Low-low

High-low



Low-high

High-high

DCT on a Real Image Block

```
>>imblock = lena256(128:135,128:135)-128
imblock=
 54   68   71   73   75   73   71   45
   47   52   48   14   20   24   20   -8
   20  -10   -5  -13  -14  -21  -20  -21
  -13  -18  -18  -16  -23  -19  -27  -28
  -24  -22  -22  -26  -24  -33  -30  -23
  -29  -13    3  -24  -10  -42  -41    5
  -16   26   26  -21   12  -31  -40   23
   17   30   50   -5    4   12   10    5
```

```
>>dctblock =dct2(imblock)
dctblock=
 31.0000   51.7034    1.1673  -24.5837  -12.0000  -25.7508   11.9640   23.2873
 113.5766   6.9743  -13.9045   43.2054   -6.0959   35.5931  -13.3692  -13.0005
 195.5804  10.1395   -8.6657   -2.9380  -28.9833   -7.9396    0.8750    9.5585
 35.8733  -24.3038  -15.5776  -20.7924   11.6485  -19.1072   -8.5366  0.5125
 40.7500  -20.5573  -13.6629   17.0615  -14.2500   22.3828   -4.8940  -11.3606
 7.1918  -13.5722   -7.5971  -11.9452   18.2597  -16.2618   -1.4197   -3.5087
 -1.4562  -13.3225   -0.8750    1.3248   10.3817   16.0762    4.4157    1.1041
 -6.7720   -2.8384    4.1187    1.1118   10.5527   -2.7348   -3.2327    1.5799
```

In JPEG, “imblock-128” is done before DCT to shift the mean to zero

Symmetrically extension

- DCT = DFT of a symmetrically extended sequence.

$$y(n) \triangleq x(n) + x(2N-1-n),$$

- Better energy compaction

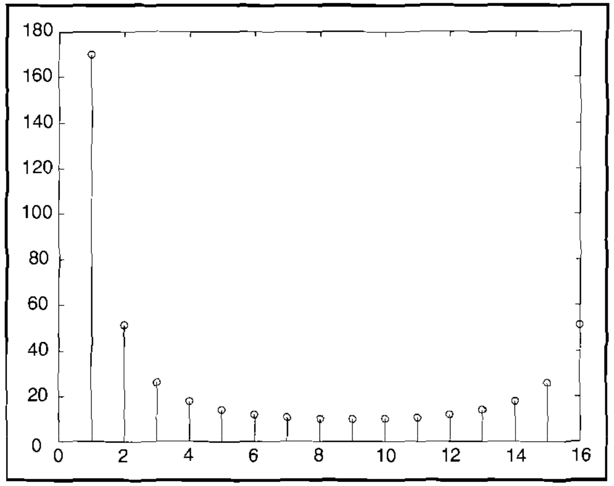


FIGURE 4.10 *MATLAB plot of DFT magnitude* $|X(k)|$

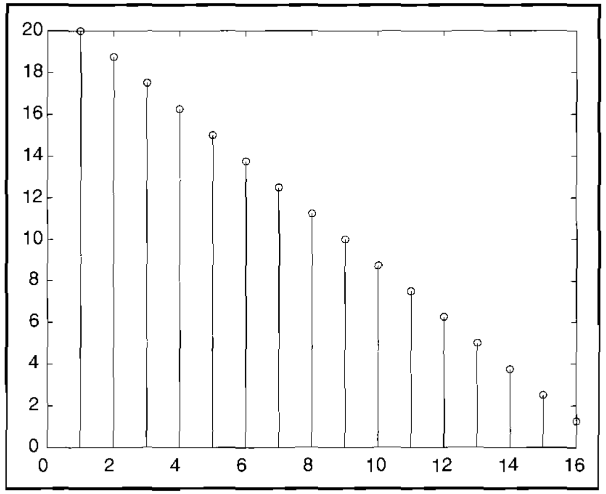


FIGURE 4.9 *MATLAB plot of* $x(n)$ *over its support*

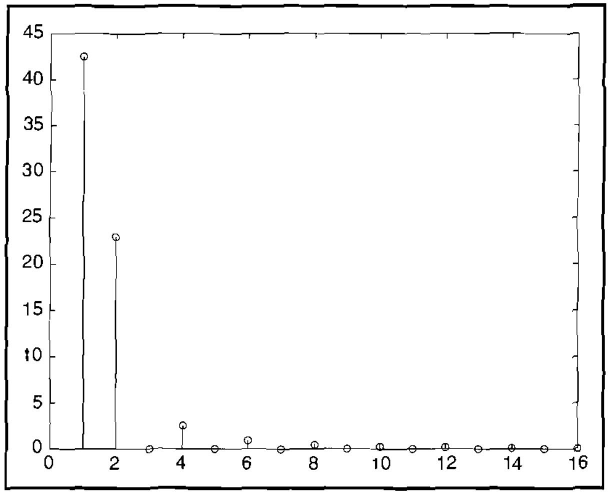


FIGURE 4.11 *MATLAB plot of DCT* $X_C(k)$

Why DCT?

- Involve only real-valued data.
- Symmetric extension property => fewer high frequency coefficients
- Better energy-compaction properties.
- DCT basis vectors approximate the optimal Karhunen-Loeve transforms for signals whose statistics are given by certain limits of Markov processes.

Quantization

- Use uniform quantizer on each coefficient
- Different coefficient is quantized with different step-size (Q):
  - Human eye is more sensitive to low frequency components
  - Low frequency coefficients with a smaller Q
  - High frequency coefficients with a larger Q

DCT + Quantization: Example

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) source image samples

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(b) forward DCT coefficients

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

(c) quantization table

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-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

(d) normalized quantized coefficients

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	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

(e) denormalized quantized coefficients

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(f) reconstructed image samples

Default Quantization Table

- Luminance

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	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

- Chrominance

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

$$B_{j,k} = \text{round}\left(\frac{A_{j,k}}{Q_{j,k}}\right) \quad \text{for } j = 0, 1, 2, \cdots, N_1 - 1; k = 0, 1, 2, \cdots, N_2 - 1$$

Quality levels

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$Q_{30} = \begin{bmatrix} 27 & 18 & 17 & 27 & 40 & 67 & 85 & 102 \\ 20 & 20 & 23 & 32 & 43 & 97 & 100 & 92 \\ 23 & 22 & 27 & 40 & 67 & 95 & 115 & 93 \\ 23 & 28 & 37 & 48 & 85 & 145 & 133 & 103 \\ 30 & 37 & 62 & 93 & 113 & 182 & 172 & 128 \\ 40 & 58 & 92 & 107 & 135 & 173 & 188 & 153 \\ 82 & 107 & 130 & 145 & 172 & 202 & 200 & 168 \end{bmatrix}$$

Entropy Coding

- Huffman coding
- DC coefficients
- Predictive coding
- AC coefficients
- Zigzag scan
- Run-length coding

Huffman Coding

- Idea: more frequent symbols -> shorter codewords
- Prefix code
- Algorithm

Step 1: Arrange the symbol probabilities $p(a_l)$, $l = 1, 2, \ldots, L$, in a decreasing order and consider them as leaf nodes of a tree.

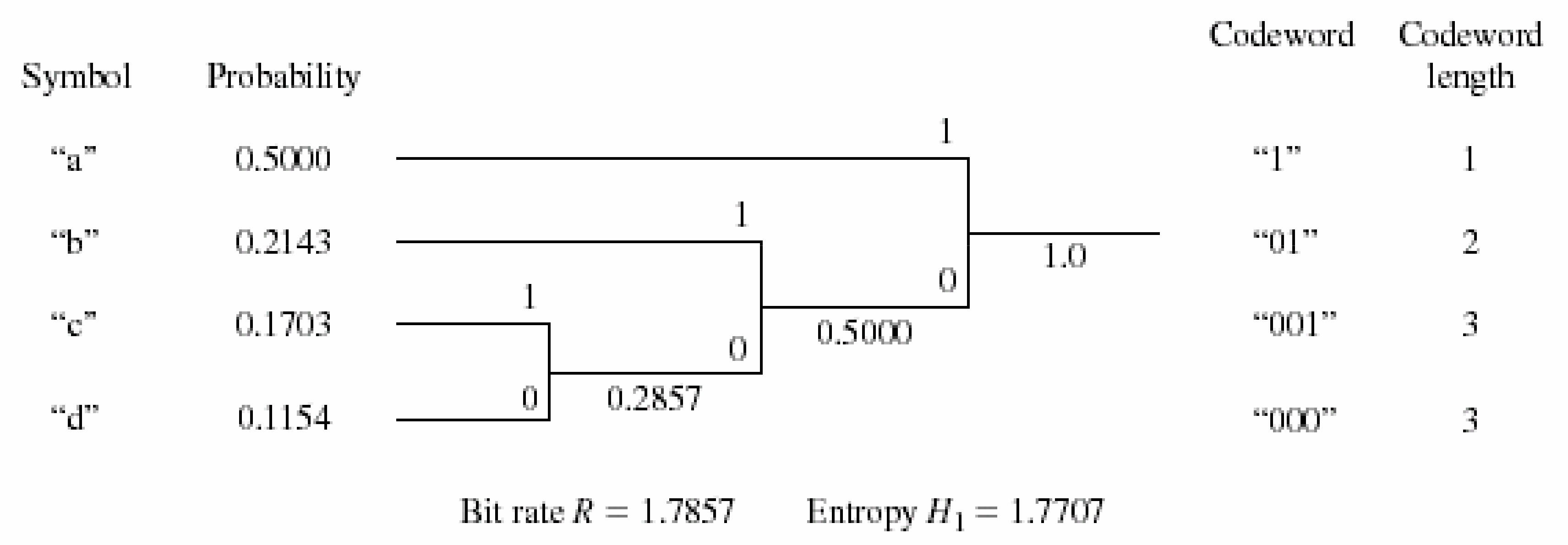
Step 2: While there is more than one node:

(a) Find the two nodes with the smallest probability and arbitrarily assign 1 and 0 to these two nodes.

(b) Merge the two nodes to form a new node whose probability is the sum of the two merged nodes. Go back to Step 1.

Step 3: For each symbol, determine its codeword by tracing the assigned bits from the corresponding leaf node to the top of the tree. The bit at the leaf node is the last bit of the codeword.

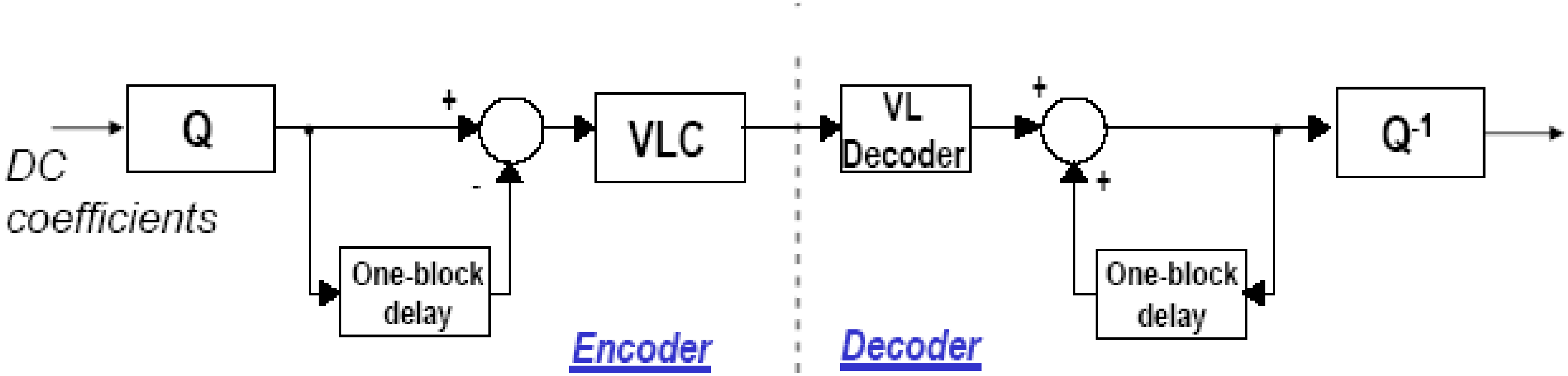
Huffman Coding Example

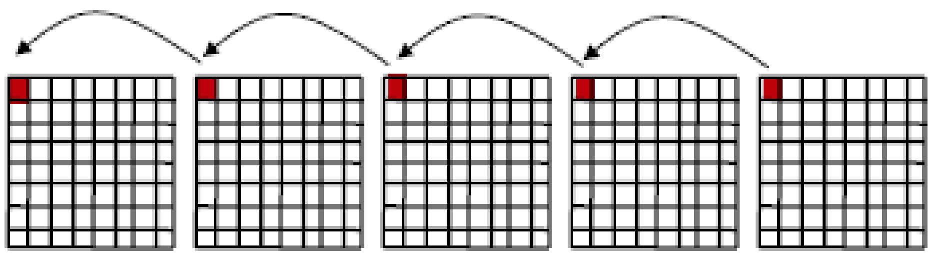


Coding of Quantized DCT Coefficients

- DC coefficient: Predictive coding
  - The DC value of the current block is predicted from that of the previous block, and the error is coded using Huffman coding
- AC Coefficients: Runlength coding
  - Many high frequency AC coefficients are zero after first few low-frequency coefficients
  - Runlength Representation:
    - Ordering coefficients in the zig-zag order
    - Specify how many zeros before a non-zero value
    - Each symbol=(length-of-zero, non-zero-value)
  - Code all possible symbols using Huffman coding
    - More frequently appearing symbols are given shorter codewords
    - For more details on the actual coding table, see Handout (Sec.8.5.3 in [Gonzalez02]
- One can use default Huffman tables or specify its own tables.
- Instead of Huffman coding, arithmetic coding can be used to achieve higher coding efficiency at an added complexity.

Differential coding of DC coefficients





Coding of DC Symbols

- Example:
  - Current quantized DC index: 2
  - Previous block DC index: 4
  - Prediction error: -2
  - The prediction error is coded in two parts:
    - Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code using a Huffman code (JPEG Default DC Code)
      - DC= -2 is in category “2”, with a codeword “100”
    - Which position it is in that category, using a fixed length code, length=category number
      - “-2” is the number 1 (starting from 0) in category 2, with a fixed length code of “10”.
      - The overall codeword is “10010”

JPEG Tables for Coding DC

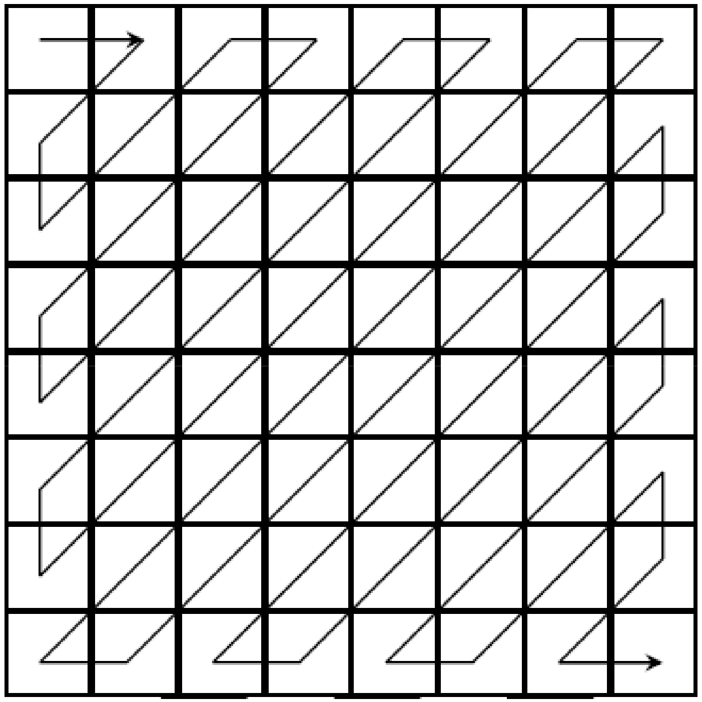
TABLE 8.17
JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, ..., −4, 4, ..., 7	3	3
−15, ..., −8, 8, ..., 15	4	4
−31, ..., −16, 16, ..., 31	5	5
−63, ..., −32, 32, ..., 63	6	6
−127, ..., −64, 64, ..., 127	7	7
−255, ..., −128, 128, ..., 255	8	8
−511, ..., −256, 256, ..., 511	9	9
−1023, ..., −512, 512, ..., 1023	A	A
−2047, ..., −1024, 1024, ..., 2047	B	B
−4095, ..., −2048, 2048, ..., 4095	C	C
−8191, ..., −4096, 4096, ..., 8191	D	D
−16383, ..., −8192, 8192, ..., 16383	E	E
−32767, ..., −16384, 16384, ..., 32767	F	N/A

TABLE 8.18
JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Zigzag Ordering of DCT Coefficients



Zig-Zag ordering: converting a 2D matrix into a 1D array, so that the frequency (horizontal+vertical) increases in this order, and the coefficient variance (average of magnitude square) decreases in this order.

Example of Zigzag ordering

qdct =

2	5	0	-2	0	-1	0	0
9	1	-1	2	0	1	0	0
14	1	-1	0	-1	0	0	0
3	-1	-1	-1	0	0	0	0
2	-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

Run-length symbol representation:
{2,(0,5),(0,9),(0,14),(0,1),(1,-2),(0,-1),(0,1),(0,3),(0,2),(0,-1),(0,-1),(0,2),(1,-1),(2,-1), (0,-1), (4,-1),(0,-1),(0,1),EOB}

EOB: End of block, one of the symbol that is assigned a short Huffman codeword

Coding of AC Coefficients

- Example:
  - First symbol (0,5)
    - The value ‘5’ is represented in two parts:
    - Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code the “(runlength,category)” using a Huffman code (JPEG Default AC Code)
      - AC=5 is in category “3”,
      - Symbol (0,3) has codeword “100”
    - Which position it is in that category, using a fixed length code, length=category number
      - “5” is the number 5 (starting from 0) in category 3, with a fixed length code of “101”.
      - The overall codeword for (0,5) is “100101”
  - Second symbol (0,9)
    - ‘9’ in category ‘4’, (0,4) has codeword ‘1011’,’9’ is number 9 in category 4 with codeword ‘1001’ -> overall codeword for (0,9) is ‘10111001’
  - ETC

JPEG Tables for Coding DC

TABLE 8.17
JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
$-1, 1$	1	1
$-3, -2, 2, 3$	2	2
$-7, \ldots, -4, 4, \ldots, 7$	3	3
$-15, \ldots, -8, 8, \ldots, 15$	4	4
$-31, \ldots, -16, 16, \ldots, 31$	5	5
$-63, \ldots, -32, 32, \ldots, 63$	6	6
$-127, \ldots, -64, 64, \ldots, 127$	7	7
$-255, \ldots, -128, 128, \ldots, 255$	8	8
$-511, \ldots, -256, 256, \ldots, 511$	9	9
$-1023, \ldots, -512, 512, \ldots, 1023$	A	A
$-2047, \ldots, -1024, 1024, \ldots, 2047$	B	B
$-4095, \ldots, -2048, 2048, \ldots, 4095$	C	C
$-8191, \ldots, -4096, 4096, \ldots, 8191$	D	D
$-16383, \ldots, -8192, 8192, \ldots, 16383$	E	E
$-32767, \ldots, -16384, 16384, \ldots, 32767$	F	N/A

TABLE 8.18
JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

JPEG Tables for Coding AC Symboles

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Performance of JPEG

- For color images at 24 bits/pixel (bpp)
  - 0.25-0.5 bpp: moderate to good
  - 0.5-0.75 bpp: good to very good
  - 0.75-1.5 bpp: excellent, sufficient for most applications
  - 1.5-2 bpp: indistinguishable from original
  - From: G. K. Wallace: The JPEG Still picture compression standard, *Communications of ACM*, April 1991.
- For grayscale images at 8 bpp
  - 0.5 bpp: excellent quality

JPEG Performance

487x414 pixels,
Uncompressed, 600471 Bytes,24 bpp
85502 Bytes, 3.39 bpp, CR=7

487x414 pixels
41174 Bytes, 1.63 bpp, CR=14.7

JPEG Performance for B/W images

65536 Bytes
8 bpp

4839 Bytes
0.59 bpp
CR=13.6

3037 Bytes
0.37 bpp
CR=21.6

1818 Bytes
0.22 bpp
CR=36.4

Pros and Cons

- Pros
  - Low complexity
  - Memory efficiency
  - Reasonable coding efficiency
- Cons
  - Single resolution
  - No target bit rate
  - Blocking artifact at low bit rate
  - No lossless capability

References

- Lecture slides by Prof Min Wu
- Lecture slides by Prof Bernd Girod
- Lecture slides by Yao Wang
- G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Trans. Consumer Electronics*, vol. 38, no. 1, pp. xviii-xxxiv, Feb. 1992
- ITU-T Rec. T.81