CS 6351 DATA COMPRESSION

JPEG & MPEG STANDARDS

Instructor: Abdou Youssef

OBJECTIVES OF THIS LECTURE

By the end of this lecture, you will be able to:

- Describe what standards are, why they are useful, and why they do not necessarily hinder progress
- Explain the main steps of the JPEG standard for still images, and the main steps of the MPEG1 and MPEG standards for motion pictures (videos)
- Appreciate how JPEG and MPEG use nothing but the techniques covered in this course, and integrate those techniques in interesting ways
- Describe basic motion tracking techniques, and how motion-compensation is used in MPEG to greatly increase compression performance
- Implement JPEG- and MPEG-like coders/decoders

OUTLINE

- Definition of standards
- Why standards are useful
- JPEG standard
- MPEG Standard
- References

WHAT ARE STANDARDS?

- A standard is an agreed upon way of doing something like
  - making a product
  - managing a process
  - delivering a service
  - supplying materials
- A standard captures the distilled wisdom of
  - researchers/experts/practitioners in the subject matter
  - who know the needs of the organizations they represent
- In compression/communications, standards especially focus on interface, specifying the format/measurements of the IO data and of interacting devices
- While prescriptive, standards need to minimize being prescriptive

MOTIVATION OF STANDARDS

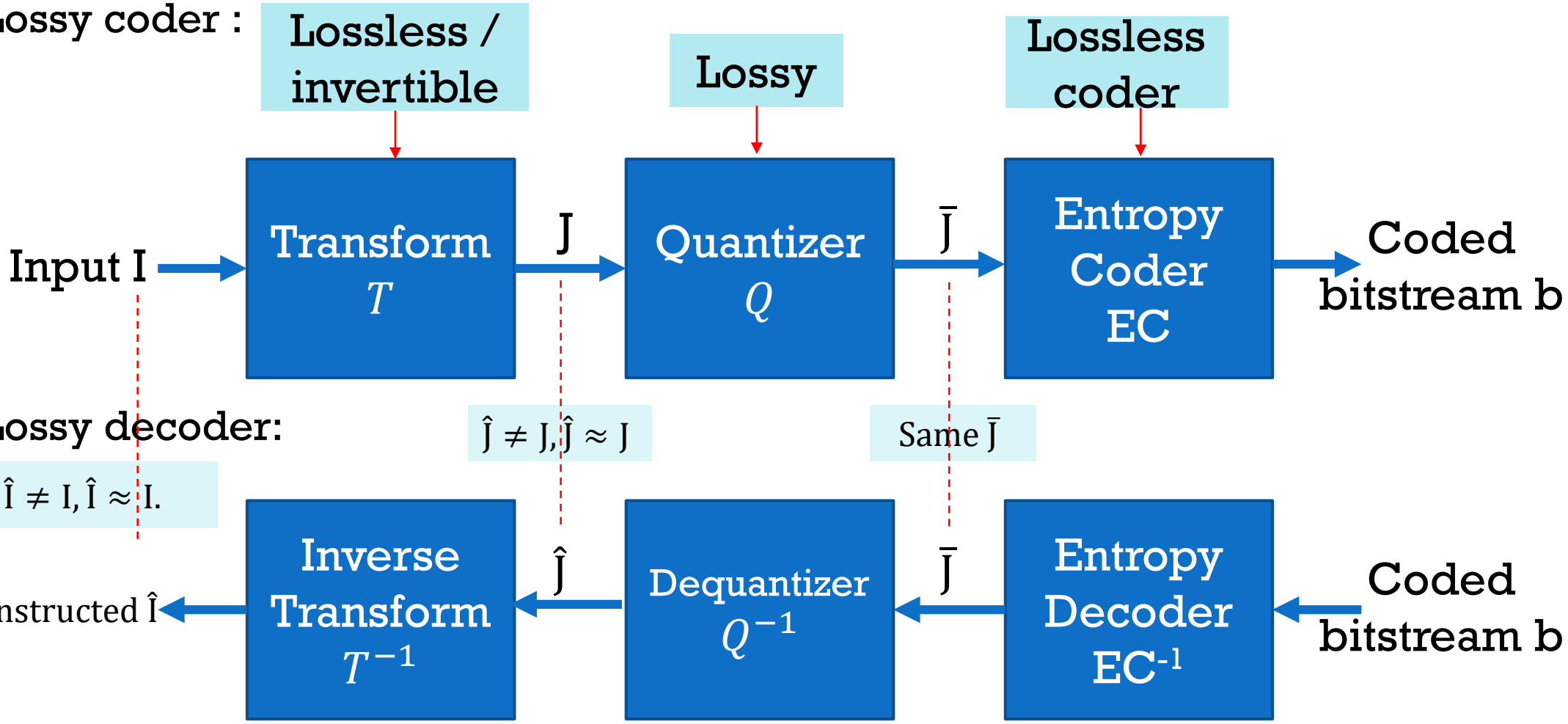
- Why Standards?
  - Compatibility between products by different vendors
    - Interoperability and sharing
      - A JPG image from one camera/phone can be shared and visualized on other devices
    - Flexibility of purchasing from anywhere
      - Can buy a coder from one vendor, and a decoder from another vendor
  - Price reduction
    - When different companies can make a product, you get the usual benefits of competition: better quality, better service, and better prices
    - Standards ensure there will be many buyers of the compliant product, thus vendors can lower their prices and still make a profit
  - Growth: Good for producers, consumers, and the economy

DO STANDARDS LIMIT RESEARCH & INNOVATION?

- Not necessarily
- Standards (at least JPEG & MPEG) are very flexible regarding the encoder design, leaving much room for individual improvement
- The trends are toward even greater flexibility/choices in future generations of the standards
- Another growth area is the development of systems which integrate components that use the standards

GENERAL SCHEME OF LOSSY COMPRESSION

- Lossy coder :



- Lossy decoder:

$\hat{I} \neq I, \hat{I} \approx I.$

We will see how JPEG and MPEG standardize each of those components

IMAGE/VIDEO COMPRESSION STANDARDS (OUTLINE)

- JPEG
  - Baseline JPEG: we will cover it
  - Extended JPEG: we provide a brief coverage of it
  - Lossless JPEG: DPCM + Huffman/Arithmetic (no need to say more)
- MPEG History (H.261 and H.263)
  - MPEG1: subsumed by MPEG2
  - MPEG2: we will cover it
  - No MPEG3: its charge of including HDTV was fulfilled by MPEG2, so it was dropped
  - MPEG4:
    - It allows the option of DCT vs. wavelet
    - Won't cover it, but you will see wavelets/subband coding later in the semester

THE JPEG "TOOLKIT" (1/2)
-- BASELINE JPEG --

- JPEG provides a ``toolkit'' of techniques for compressing continuous-tone, still, color and monochrome images

- Baseline JPEG provides a DCT-based algorithm, and uses run-length encoding and Huffman coding

- Baseline JPEG operates only in sequential mode, and is restricted to 8 bits/pixel input images

THE JPEG "TOOLKIT" (2/2)
-- EXTENDED JPEG --

- Extended JPEG offers several optional enhancements
  - 12-bit/pixel input
  - Progressive transmission
  - Choice between Arithmetic and Huffman coding
  - Adaptive quantization
  - Tiling
  - Still picture interchange file format (SPIFF)
  - Selective refinement

APPLICATIONS OF JPEG

- JPEG is widely used
  - Digital camera
  - Smart phones
  - Computer devices, etc.
- Applications
  - Desktop publishing
  - Photojournalism
  - Medical images
  - General image archiving systems
  - Consumer imaging
  - Graphic arts, etc.

THE BASELINE JPEG ALGORITHM (1/2)

-- SUMMARY --

1. It operates on 8×8 blocks of the input image
2. Mean-normalization (subtract 128 from each pixel)
3. Transform: DCT-transform each block
4. Quantization
   - An 8×8 quantization matrix Q is user-provided
   - Each block is divided by Q (point by point)
   - The terms are then rounded to their nearest integers
   - Remark: Up to 4 quantization matrices per image are allowed (for example, one for luminance, and one for each of the three color components)

THE BASELINE JPEG ALGORITHM (2/2)

-- SUMMARY --

5. Entropy-coding of the DC coefficients (the top left coefficient of each quantized block) using DPCM+Huffman
   - Huffman-encode the DC residuals derived from the difference between each DC and the DC of the preceding block
6. Entropy-coding of the AC (i.e., non-DC) coefficients
   - Zigzag-order the quantized coefficients of each block
   - Record for each nonzero coefficient both its distance (called run) to the preceding nonzero coefficient in the zigzag sequence, and its value (called level)
   - Huffman code the [run,level] terms using one single Huffman table for all the AC's of the image

THE BASELINE JPEG ALGORITHM IN DETAIL (1/15)

-- BLOCK-ORIENTED --

1. It operates on 8×8 blocks of the input image

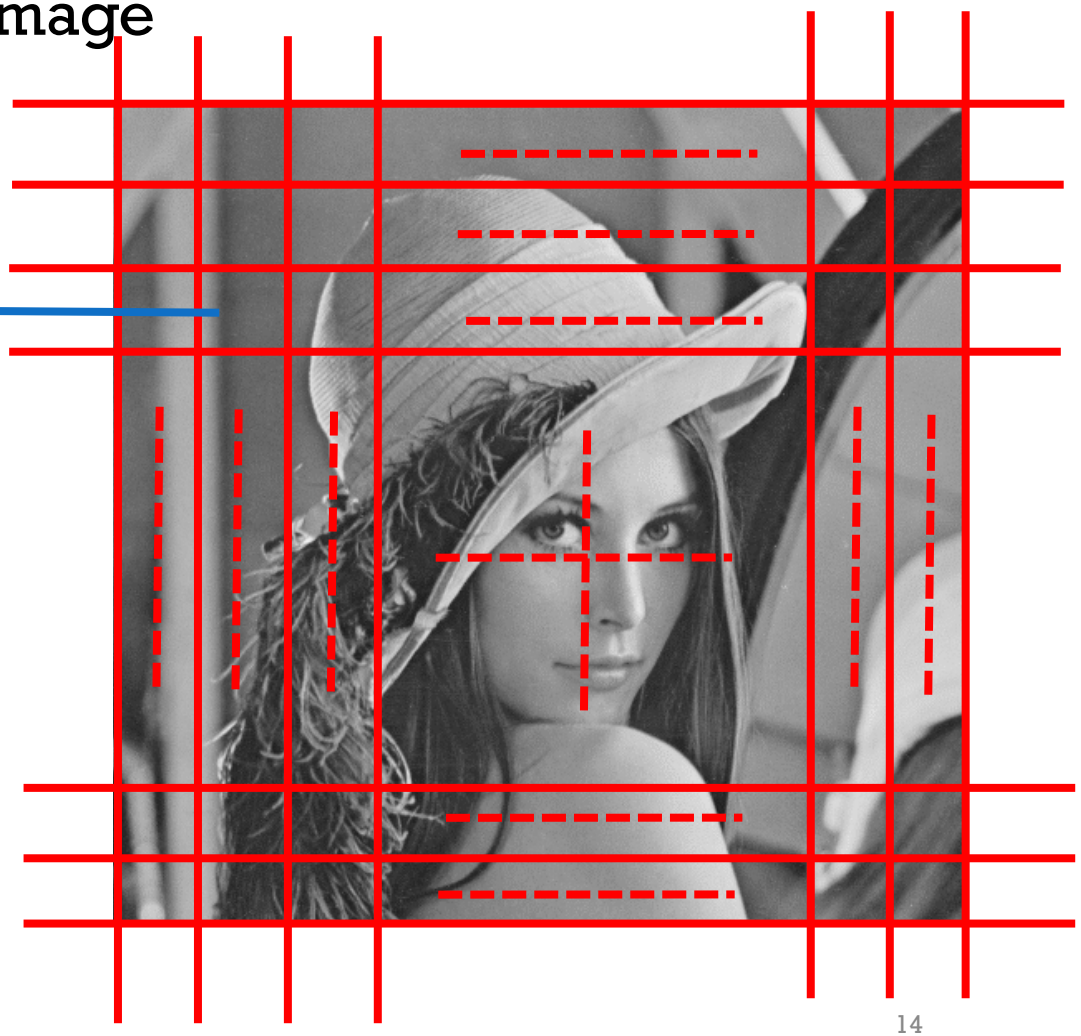
143	147	149	152	156	147	146	149
151	146	143	154	148	144	153	132
147	143	145	149	144	145	128	133
152	145	145	144	146	134	130	137
146	143	142	147	124	127	139	138
139	145	139	127	126	135	139	141
145	137	124	130	138	136	140	144
144	124	136	134	137	139	142	145

Mean-normalize

DCT

Quantize

Entropy-code



THE BASELINE JPEG ALGORITHM IN DETAIL (2/15)

-- MEAN-NORMALIZATION OF BLOCKS --

2. Mean-normalization (subtract 128 from each pixel)

143	147	149	152	156	147	146	149
151	146	143	154	148	144	153	132
147	143	145	149	144	145	128	133
152	145	145	144	146	134	130	137
146	143	142	147	124	127	139	138
139	145	139	127	126	135	139	141
145	137	124	130	138	136	140	144
144	124	136	134	137	139	142	145

15	19	21	24	28	19	18	21
23	18	15	26	20	16	25	4
19	15	17	21	16	17	0	5
24	17	17	16	18	6	2	9
18	15	14	19	-4	-1	11	10
11	17	11	-1	-2	7	11	13
17	9	-4	2	10	8	12	16
16	-4	8	6	9	11	14	17

THE BASELINE JPEG ALGORITHM IN DETAIL (3/15)

-- DCT APPLIED ON EACH MEAN-NORMALIZED BLOCK --

3. Transform: DCT-transform each block

15	19	21	24	28	19	18	21
23	18	15	26	20	16	25	4
19	15	17	21	16	17	0	5
24	17	17	16	18	6	2	9
18	15	14	19	-4	-1	11	10
11	17	11	-1	-2	7	11	13
17	9	-4	2	10	8	12	16
16	-4	8	6	9	11	14	17

DCT

103.4	12.4	6.0	2.1	8.1	5.7	-0.7	-0.4
31.7	11.6	-22.8	-0.7	-0.1	-0.5	-4.7	-1.8
11.0	-21.2	-3.7	8.8	2.3	0.1	-0.2	5.6
0.2	-4.9	10.3	-8.6	-8.9	-2.5	-7.6	-1.4
4.9	-0.4	-1.9	-8.9	6.6	2.6	3.6	3.6
0.7	-0.9	-0.9	4.1	7.2	-15.5	2.8	1.8
-3.1	-1.0	-2.5	-5.5	-5.2	-6.7	10.7	-1.1
-2.9	-0.2	-1.2	-2.7	3.6	2.6	0.4	-6.4

DCT output D

THE BASELINE JPEG ALGORITHM IN DETAIL (4/15)
-- QUANTIZATION --

4. Quantization
   - An 8×8 quantization matrix Q is user-provided
   - Each block D is divided by Q (point by point)
   - The terms are then rounded to their nearest integers (using NINT)

- Example:

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

THE BASELINE JPEG ALGORITHM IN DETAIL (5/15)

-- QUANTIZATION: ILLUSTRATION & EXPLANATION --

103.4	12.4	6.0	2.1	8.1	5.7	-0.7	-0.4
31.7	11.6	-22.8	-0.7	-0.1	-0.5	-4.7	-1.8
11.0	-21.2	-3.7	8.8	2.3	0.1	-0.2	5.6
0.2	-4.9	10.3	-8.6	-8.9	-2.5	-7.6	-1.4
4.9	-0.4	-1.9	-8.9	6.6	2.6	3.6	3.6
0.7	-0.9	-0.9	4.1	7.2	-15.5	2.8	1.8
-3.1	-1.0	-2.5	-5.5	-5.2	-6.7	10.7	-1.1
-2.9	-0.2	-1.2	-2.7	3.6	2.6	0.4	-6.4

./

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**

DCT output D

Quantization matrix Q

NINT(D./Q)

6	**1**	**0**	**0**	**0**	**0**	**0**	**0**
4	**1**	**-2**	**0**	**0**	**0**	**0**	**0**
1	**-2**	**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**

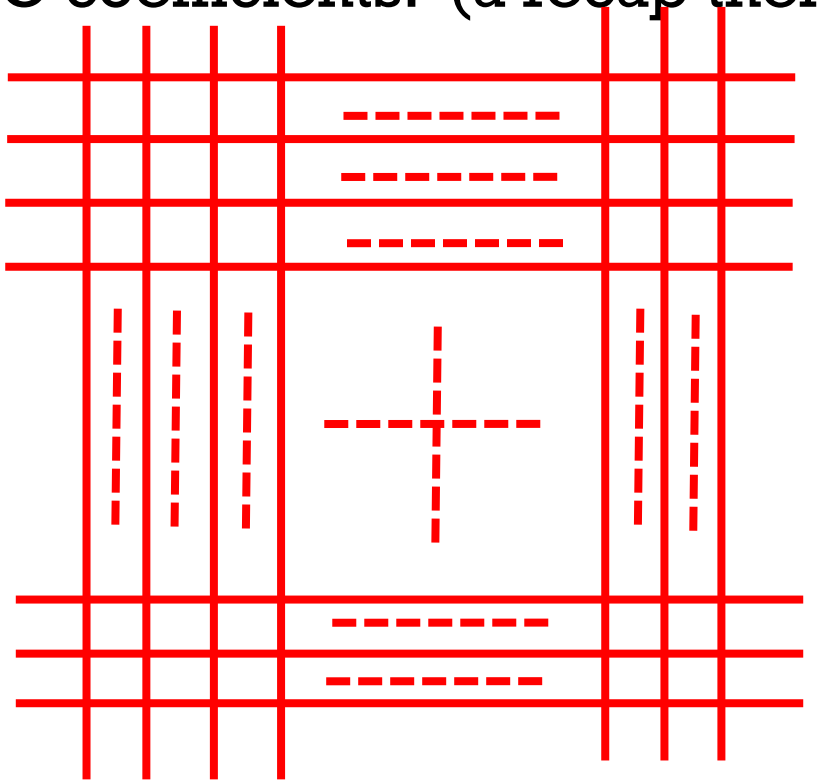
This is called the **DC** term of the block

- Q specifies 64 different quantizers, one per frequency coefficient
- As you move to the bottom right of Q, the numbers tend to get larger. Why?
- A quantizer Q_a where $Q_a(x) = \text{NINT}\left(\frac{x}{a}\right)$ is a uniform quantizer of interval $\text{length} = a$. ???

THE BASELINE JPEG ALGORITHM IN DETAIL (6/15)

-- ENTROPY CODING OF THE DC TERMS --

5. Entropy-coding of the DC coefficients: (a recap then general description)



Mean-normalize+DCT+Quantize all the blocks

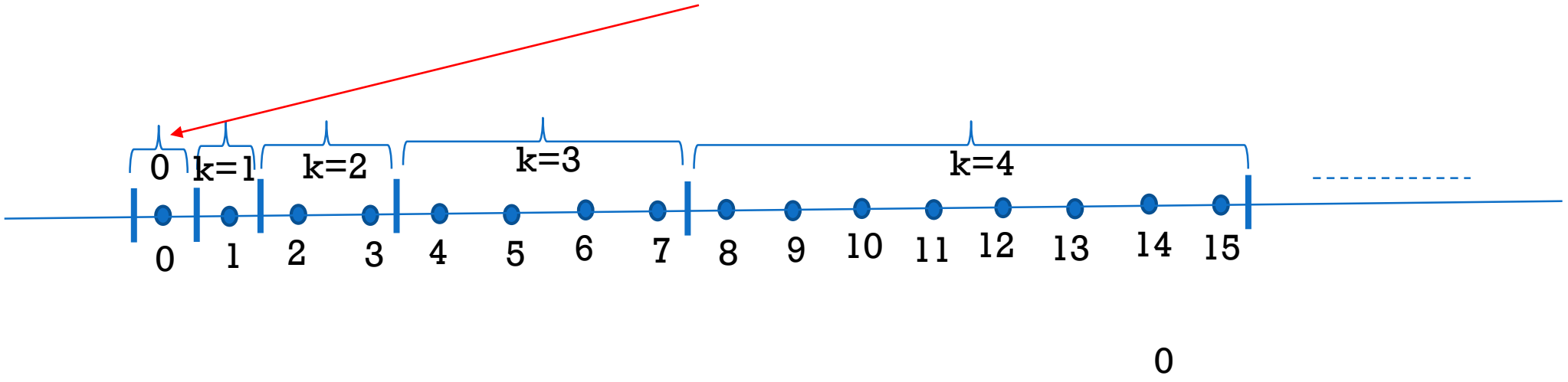
a. Collect all the DC terms, one DC term from each DCTed+quantized block, forming a 1D sequence
b. Apply DPCM on the sequence
c. Apply Huffman coding as explained next

Note: this is the only place in JPEG where any "processing-interactions" between blocks occur

THE BASELINE JPEG ALGORITHM IN DETAIL (7/15)

-- DETAILS OF ENTROPY CODING OF THE DC TERMS (1/2) --

5. Entropy-coding of the DC coefficients:
   a. The DC residuals are in the range [-2047,2047]
   b. Thus, the magnitude of each residual is between 0 and $2047 = 2^{11} - 1$, inclusive.
   c. Divide this range into 12 subranges, or categories, where category k ranges from 2^{k-1} to $2^k - 1$ inclusive.
      - Note that $0 \leq k \leq 11$, and for $k = 0$, category 0 has only the integer 0



THE BASELINE JPEG ALGORITHM IN DETAIL (8/15)

-- DETAILS OF ENTROPY CODING OF THE DC TERMS (2/2) --

5. Entropy-coding of the DC coefficients: Let r be a DC residual.

   d. $r \in [-2047, 2407] \Rightarrow 0 \leq |r| \leq 2047 = 2^{10} + 1023 \Rightarrow |r| = 2^{k-1} + t$, for some k and t,

   - $0 \leq t \leq 2^{k-1} - 1,\ 0 \leq k \leq 11$
   - t can be represented in binary using $k - 1$ bits.

   If $|r| = 0$, take $k = 0$ and $t = 0$;
   If $|r| = 1$, then $k = 1$ and $t = 0$

   e. Therefore, r can be uniquely represented by $s, k,$ and t, where s is the sign of r, and k and t are as above. Represent $r \equiv [k, s, t]$ temporarily

   f. Develop a Huffman code for the 12 categories ($0 \leq k \leq 11$), where every codeword is at most 16 bits long

   Exercise: How would you find the probabilities of the 12 categories, for a given input image?

   g. Encode each DC residual r as a binary string hsm where

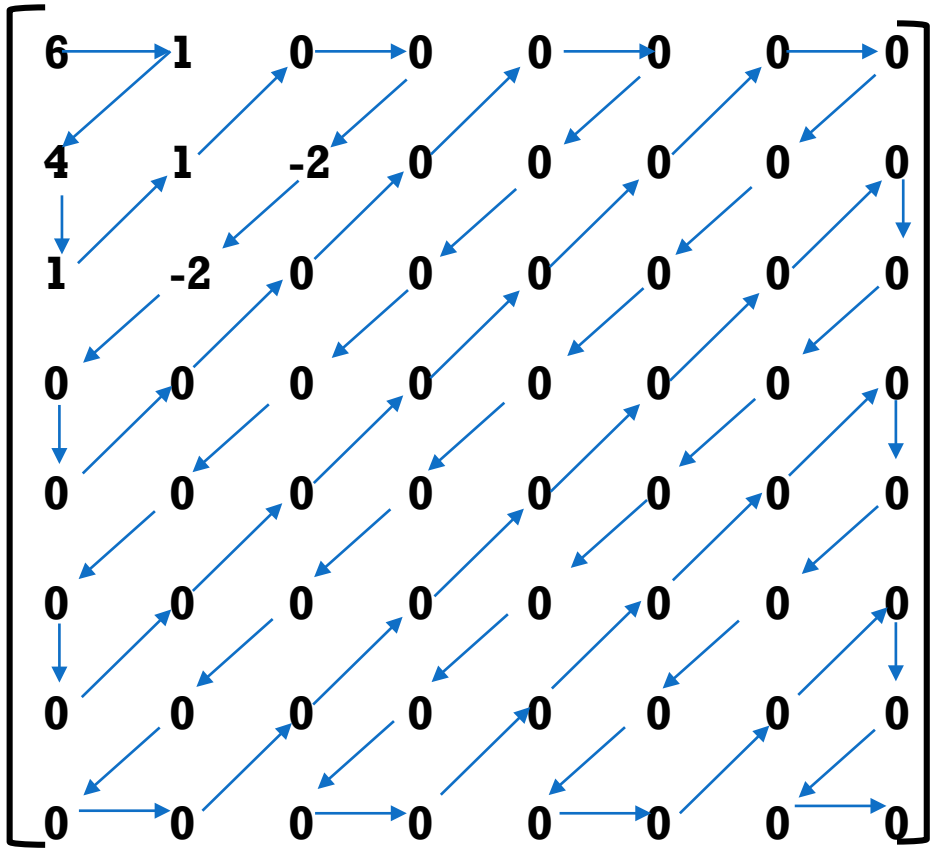
   - h is the Huffman codeword of the residual's category k
   - s= sign of the residual; s=0 if the residual is negative, 1 if positive
   - m= the $(k-1)$-bit binary representation of t

   Exercise: How would you modify the Huffman algorithm to produce a Huffman tree of height at most 16?

THE BASELINE JPEG ALGORITHM IN DETAIL (9/15)

-- ENTROPY CODING OF THE AC TERMS (1/7) --

6. Entropy-code of the AC (i.e., non-DC) coefficients of each block separately
   a. Zigzag ordering of NINT(D./Q)
   b. List the elements in that order : 6 1 4 1 1 0 0 -2 -2 allzeros



THE BASELINE JPEG ALGORITHM IN DETAIL (10/15)

-- ENTROPY CODING OF THE AC TERMS (2/7) --

c. Record for each nonzero AC coefficient both its distance (called run) to the preceding nonzero coefficient in the zigzag sequence, and its value (called level): represent every non-zero AC as [run,level]

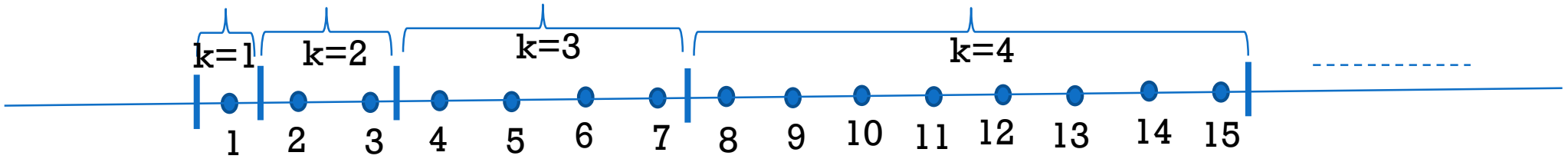
- Example: 6 **1 4 1 1 0 0 -2 -2 allzeros** ← The AC terms
- **[0,1] [0,4] [0,1] [0,1] [2,-2] [0,-2] EOB // EOB= end of block**

b. Huffman code the [run,level] terms using one single Huffman table for all the AC's of the image (explained next)

THE BASELINE JPEG ALGORITHM IN DETAIL (11/15)

-- ENTROPY CODING OF THE AC TERMS (3/7) --

1. All non-zero AC terms are of magnitude $\leq 2^{10} - 1$
2. Let x be a non-zero AC term, $x \equiv [run, level]$, and let $d = run$, i.e., the length of the zero run between x and the previous nonzero AC term.
   - $level$ =value of x. For more convenience, we'll use x instead of *level*
3. $1 \leq |x| \leq 2^{10} - 1$
4. Divide the range from 1 to $2^{10} - 1$, into 10 categories, where category k ranges from 2^{k-1} to $2^k - 1$ inclusive, $1 \leq k \leq 10$,



THE BASELINE JPEG ALGORITHM IN DETAIL (12/15)
-- ENTROPY CODING OF THE AC TERMS (4/7) --

5. Represent the value of x by its sign s and its magnitude $|x|$
   - $s = 0$ if $x < 0$, $s = 1$ if $x > 0$
6. For whatever the value of $|x|$, there is a unique k where $2^{k-1} \leq |x| \leq 2^k - 1$
   - k is the category of x
7. $|x| = 2^{k-1} + t$, where $0 \leq t \leq 2^{k-1} - 1$. So, t can be represented with $k - 1$ bits
8. Therefore, the value of x can be uniquely represented by $s, k,$ and t
9. Combined with its run d, the AC term x can be presented as $[d, k, t, s]$
   - This is intermediate epresentation, especially for d and k

THE BASELINE JPEG ALGORITHM IN DETAIL (13/15)

-- ENTROPY CODING OF THE AC TERMS (5/7) --

We are at $x \equiv [d, k, t, s]$

10. The run-length d is between 0 and 62
11. Express $d = 15p + r$, where $r = 0,1,2, \ldots, 14$ and $p = 0, 1, 2, 3, 4$
12. r can be represented with 4 bits $r_3 r_2 r_1 r_0$, different from 1111 (why?)
13. p can be represented with $11110000_1\ 11110000_2\ \ldots\ 11110000_p$
14. This implies that d is $11110000_1\ 11110000_2\ \ldots\ 11110000_p\ r_3 r_2 r_1 r_0$
15. The category k, being between 1 and 10, can be represented with 4 bits $k_3 k_2 k_1 k_0$
16. Therefore, the $[d, k]$ in the $[d, k, t, s]$ representation of AC term x, is represented as $11110000_1\ 11110000_2\ \ldots\ 11110000_p\ r_3 r_2 r_1 r_0 k_3 k_2 k_1 k_0$
17. This representation of $[d, k]$ can be viewed as a sequence of $p + 1$ bytes

THE BASELINE JPEG ALGORITHM IN DETAIL (14/15)
-- ENTROPY CODING OF THE AC TERMS (6/7) --

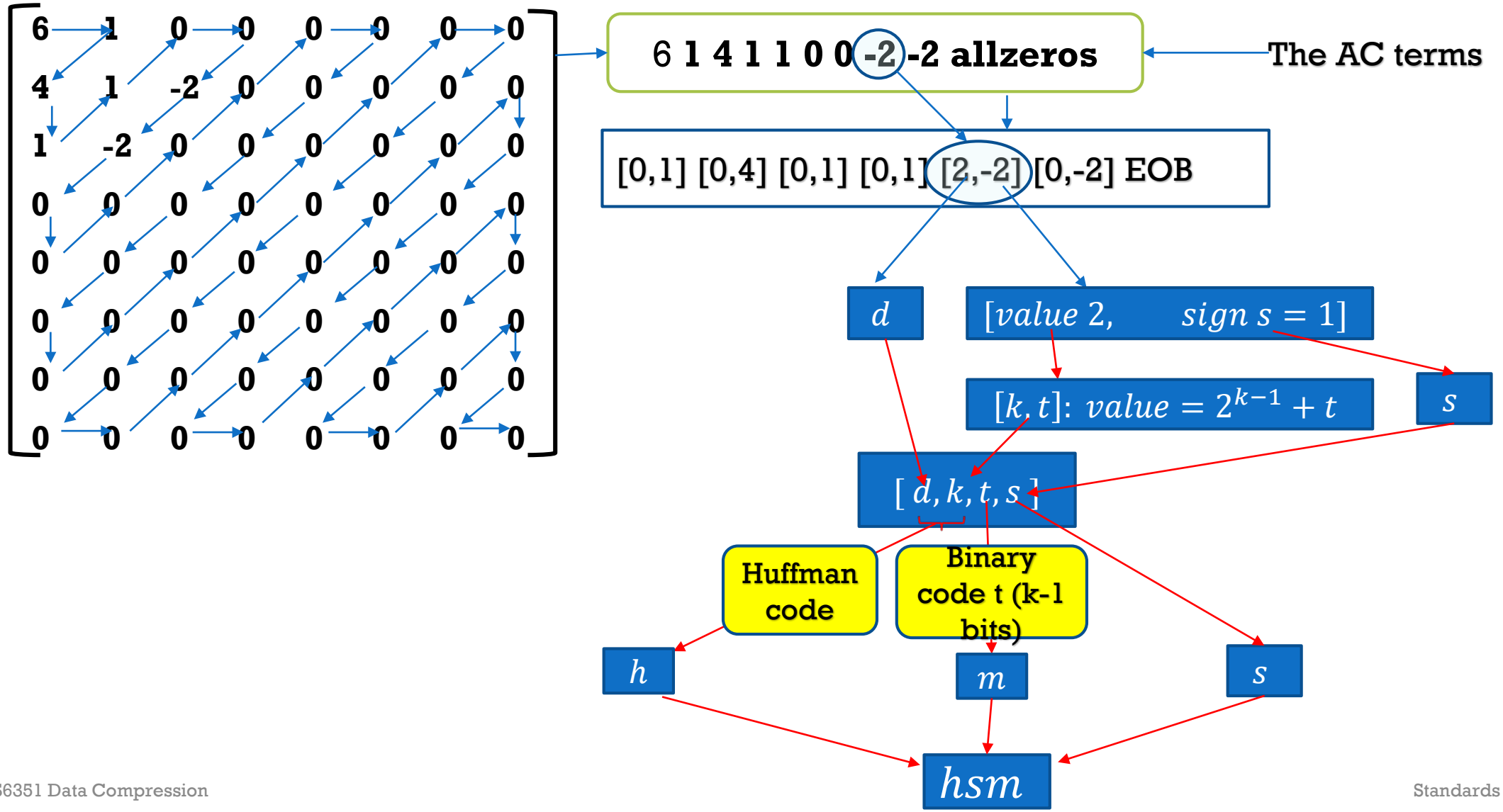
16. Therefore, the $[d, k]$ in the $[d, k, t, s]$ representation of AC term x, is represented as $11110000_1\ 11110000_2\ \ldots\ 1111000_p\ r_3r_2r_1r_0k_3k_2k_1k_0$
17. This representation of $[d, k]$ can be viewed as a sequence of $p + 1$ bytes
18. The last byte $r_3r_2r_1r_0k_3k_2k_1k_0$ represents $15 \times 10 = 150$ values (why?)
19. Add to those the byte 11110000 and the end-of-block (EOB) symbol to signal the end of the nonzero AC terms in a block
20. This results in 152 different symbols needed for the $[d, k]$ representations
21. Build a Huffman table for those 152 symbols, where every codeword is at most 16 bits long
    - The probabilities of those symbols can be derived from the image being coded

THE BASELINE JPEG ALGORITHM IN DETAIL (15/15)

-- ENTROPY CODING OF THE AC TERMS (7/7) --

22. JPEG encodes each quantized non-zero AC term, whose intermediary representation is $[d, k, t, s]$, as $\boldsymbol{hsm}$ where
    - $\boldsymbol{h}$ is the Huffman codeword of $[d, k]$ (just explained on the previous slide)
    - $\boldsymbol{s}$ is the 1-bit sign of the AC term; s=0 if AC term negative, 1 if positive
    - $\boldsymbol{m}$ is the $(k-1)$-bit binary representation of t
23. End of AC coding and the whole JPEG encoding

DIAGRAM SUMMARY OF AC ENCODING



CONTINUING WITH THE EXAMPLE OF THE BLOCK (1/5)

-- EXAMPLE HUFFMAN TABLE FOR LENA --

d=Length of Zero Run	Category k	Code-length	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
…	…	…	…
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
…	…	…	…

d=Length of Zero Run	Category k	Code-length	Codeword
2	1	5	11011
2	2	8	11111000
…	…	…	…
3	1	6	111010
3	2	9	111110111
…	…	…	…
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
…	…	…	…
EOB		4	1010

CONTINUING WITH THE EXAMPLE OF THE BLOCK (2/5)

- The block example in zigzag order:
  - 6 **1 4 1 1 0 0 -2 -2 allzeros**
  - **[0,1] [0,4] [0,1] [0,1] [2,-2] [0,-2] EOB**

The DC value 6 is coded separately along with other DC terms

- The AC terms are coded as follows:
  - Represent the sequence **[0,1] [0,4] [0,1] [0,1] [2,-2] [0,-2] EOB** as a sequence of $[d,k,t,s]'s$ where t for now is in decimal

$[d,x] \rightarrow [d,k,t,s]$ where $|x| = 2^{k-1} + t, \quad s = \text{sign}(x)$

    - [0,1,0,1] [0,3,0,1] [0,1,0,1] [0,1,0,1] [2,2,0,0] [0,2,0,0] EOB
  - Illustrations of $[d,x] \rightarrow [d,k,t,s]$:
    - [0,4] -> [0,3,0,1] because $4 = 2^{3-1} + 0$, so $k = 3, t = 0,$ and $s = 1$ since $4 > 0$
    - [2,-2]->[2,2,0,0] because $|-2| = 2 = 2^{2-1} + 0$, so $k = 2, t = 0,$ and $s = 0$ since -2<0
  - Now as sequence of $[d,k,t,s]'s$ where t is now in $(k-1)$-bit binary

    [0,1,-,1] [0,3,00,1] [0,1,-,1] [0,1,-,1] [2,2,0,0] [0,2,0,0] EOB

CONTINUING WITH THE EXAMPLE OF THE BLOCK (3/5)

- The block example in zigzag order:
  - 6 **1 4 1 1 0 0 -2 -2 allzeros**
  - **[0,1] [0,4] [0,1] [0,1] [2,-2] [0,-2] EOB**
- The AC terms are coded as follows:
  - Sequence: [0,1,-,1] [0,3,00,1] [0,1,-,1] [0,1,-,1] [2,2,0,0] [0,2,0,0] EOB
  - Code([0,1,-,1])is hsm where h=Huffman([0,1])=00, s=1, m=empty, so: 00 1
  - Code([0,3,00,1]) is hsm where h=Huffman([0,3])=100, s=1, m=00, so: 100 1 00
  - Code([2,2,0,0]) is hsm where h=Huffman([2,2])= 11111000, s=0, m=0, so: 11111000 0 0
  - Code([0,2,0,0]) is hsm where h=Huffman([0,2])= 01, s=0, m=0, so: 01 0 0
  - Code(EOB) is 1010
- The final code of the ACs in block: 00 1 100 1 00 00 1 00 1 11111000 0 0 01 0 0 1010

CONTINUING WITH THE EXAMPLE OF THE BLOCK (4/5)

-- EXAMPLE HUFFMAN TABLE FOR LENA --

d=Length of Zero Run	Category k	Code-length	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
…	…	…	…
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
…	…	…	…

d=Length of Zero Run	Category k	Code-length	Codeword
2	1	5	11011
2	2	8	11111000
…	…	…	…
3	1	6	111010
3	2	9	111110111
…	…	…	…
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
…	…	…	…
EOB		4	1010

CONTINUING WITH THE EXAMPLE OF THE BLOCK (5/5)

- The final code of the ACs in block: 00 1 100 1 00 00 1 00 1 11111000 0 0 01 0 0 1010
- Number of bits to code the AC terms: 33 bits
- Bitrate per symbol (out of the 63 AC symbols): $\frac{33}{63} = 0.52$ bits/symbol
  - The denominator 63 is the gross number of all the AC terms in the block
- Compression ratio based on those ACs alone: $\frac{63 \times 8}{33} = 15.27$
  - The numerator is the number of bits in the uncoded 8×8 block at 8 bits per pixel
  - 8×8=64 but we removed one pixel to correspond to the non-counting of the DC term in these simple calculations

JPEG DECODING

1. Entropy-decode the bitstream back to the quantized blocks
2. Dequantize:
   - Multiply each block coefficient by the corresponding coefficient of the quantization matrix Q
3. Apply the inverse DCT transform on each block
4. Denormalize: add 128 to each coefficient

JPEG DECODING EXAMPLE (1/5)

- Take the example block before, and assume we performed the entropy decoding on it, returning it back to quantized form, then multiply by Q

$$\begin{bmatrix} 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} .\times \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantized block **Quantization matrix Q**

JPEG DECODING EXAMPLE (2/5)

- Dequantized Block

$$\begin{bmatrix} 96 & 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 48 & 12 & -28 & 0 & 0 & 0 & 0 & 0 \\ 14 & -26 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dequantized block

JPEG DECODING EXAMPLE (3/5)

- Apply inverse DCT on it, and denormalize (by adding 128), we get:

$$\begin{bmatrix} 143 & 147 & 153 & 157 & 157 & 154 & 149 & 145 \\ 145 & 147 & 151 & 154 & 153 & 149 & 144 & 141 \\ 146 & 147 & 149 & 149 & 146 & 142 & 137 & 134 \\ 146 & 146 & 145 & 142 & 139 & 135 & 132 & 130 \\ 145 & 143 & 140 & 136 & 133 & 131 & 130 & 130 \\ 141 & 138 & 134 & 131 & 130 & 131 & 134 & 135 \\ 136 & 134 & 130 & 128 & 129 & 133 & 139 & 142 \\ 133 & 131 & 127 & 126 & 129 & 135 & 142 & 147 \end{bmatrix}$$

Reconstructed block

JPEG DECODING EXAMPLE (4/5)

- Compute the error (original block – reconstructed block):

$$\begin{bmatrix} 143 & 147 & 149 & 152 & 156 & 147 & 146 & 149 \\ 151 & 146 & 143 & 154 & 148 & 144 & 153 & 132 \\ 147 & 143 & 145 & 149 & 144 & 145 & 128 & 133 \\ 152 & 145 & 145 & 144 & 146 & 134 & 130 & 137 \\ 146 & 143 & 142 & 147 & 124 & 127 & 139 & 138 \\ 139 & 145 & 139 & 127 & 126 & 135 & 139 & 141 \\ 145 & 137 & 124 & 130 & 138 & 136 & 140 & 144 \\ 144 & 124 & 136 & 134 & 137 & 139 & 142 & 145 \end{bmatrix} - \begin{bmatrix} 143 & 147 & 153 & 157 & 157 & 154 & 149 & 145 \\ 145 & 147 & 151 & 154 & 153 & 149 & 144 & 141 \\ 146 & 147 & 149 & 149 & 146 & 142 & 137 & 134 \\ 146 & 146 & 145 & 142 & 139 & 135 & 132 & 130 \\ 145 & 143 & 140 & 136 & 133 & 131 & 130 & 130 \\ 141 & 138 & 134 & 131 & 130 & 131 & 134 & 135 \\ 136 & 134 & 130 & 128 & 129 & 133 & 139 & 142 \\ 133 & 131 & 127 & 126 & 129 & 135 & 142 & 147 \end{bmatrix}$$

Original block **Reconstructed block**

JPEG DECODING EXAMPLE (5/5)

- The resulting error block is:

$$\begin{bmatrix} 0 & 0 & -4 & -5 & -1 & -7 & -3 & 4 \\ 6 & -1 & -8 & 0 & -5 & -5 & 9 & -9 \\ 1 & -4 & -4 & 0 & -2 & 3 & -9 & -1 \\ 6 & -1 & 0 & 2 & 7 & -1 & -2 & 7 \\ 1 & 0 & 2 & 11 & -9 & -4 & 9 & 8 \\ -2 & 7 & 5 & -4 & -4 & 4 & 5 & 6 \\ 9 & 3 & -6 & 2 & 9 & 3 & 1 & 2 \\ 11 & -7 & 9 & 8 & 8 & 4 & 0 & -2 \end{bmatrix}$$

Error block

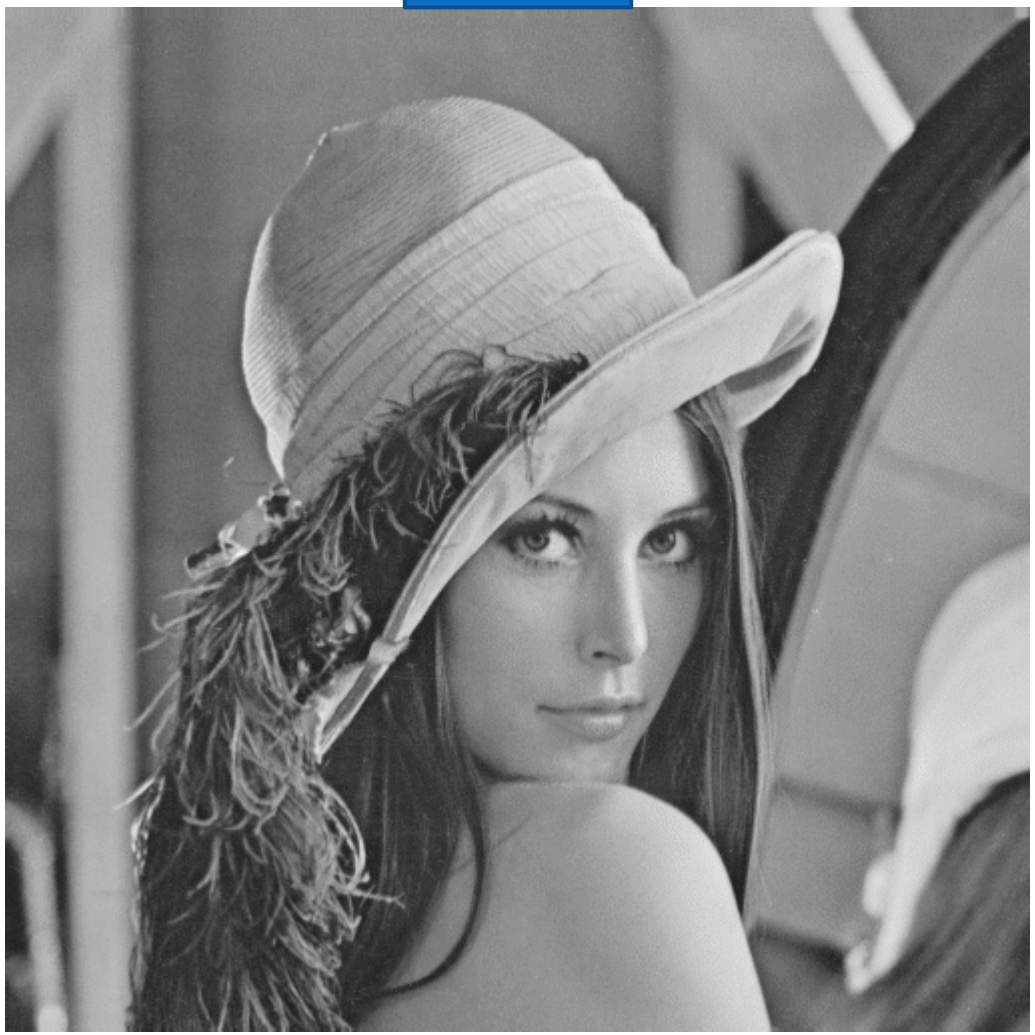
- MSE=5.2, SNR=28.31

VISUAL ILLUSTRATION OF THE PERFORMANCE OF JPEG ON AN ACTUAL IMAGE (1/4)

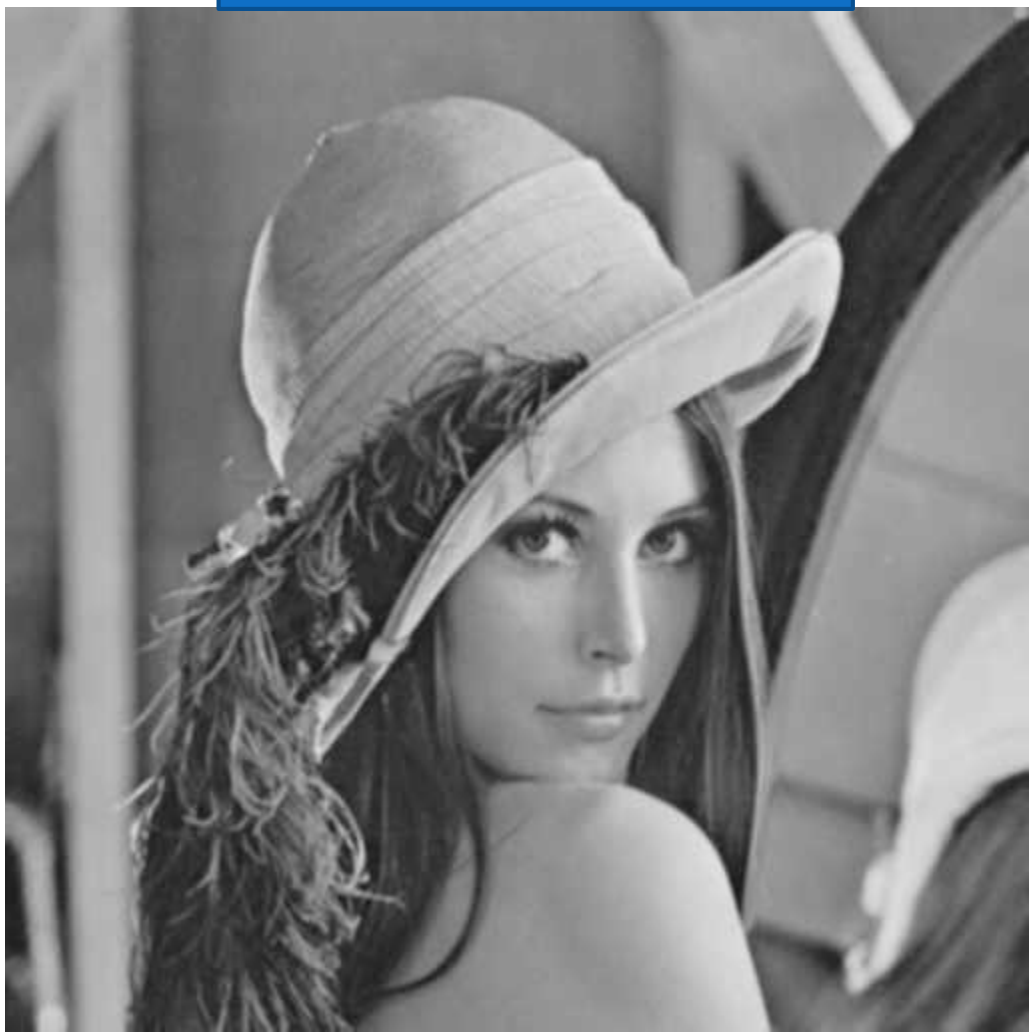
- Original image (Lena):

VISUAL ILLUSTRATION OF THE PERFORMANCE OF JPEG ON AN ACTUAL IMAGE (2/4)

Original

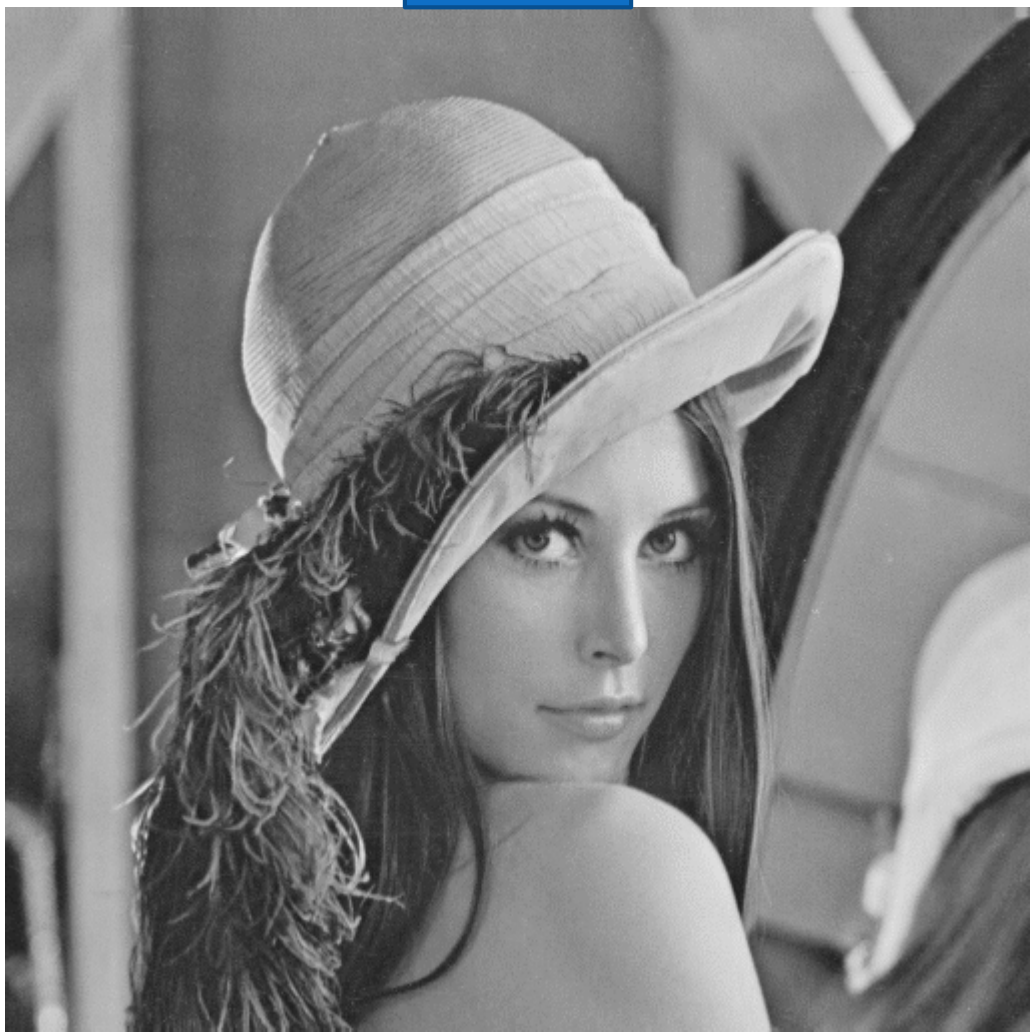


Reconstructed at CR=12

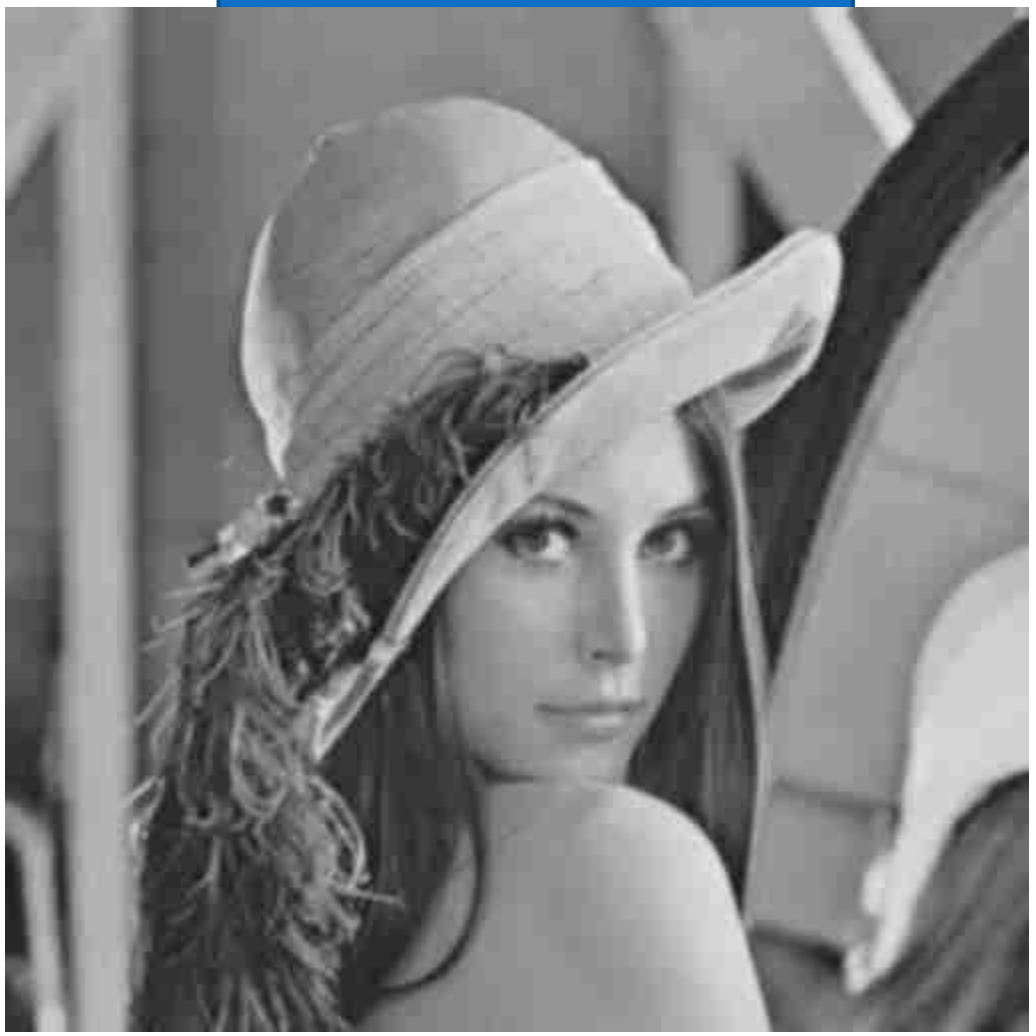


VISUAL ILLUSTRATION OF THE PERFORMANCE OF JPEG ON AN ACTUAL IMAGE (3/4)

Original

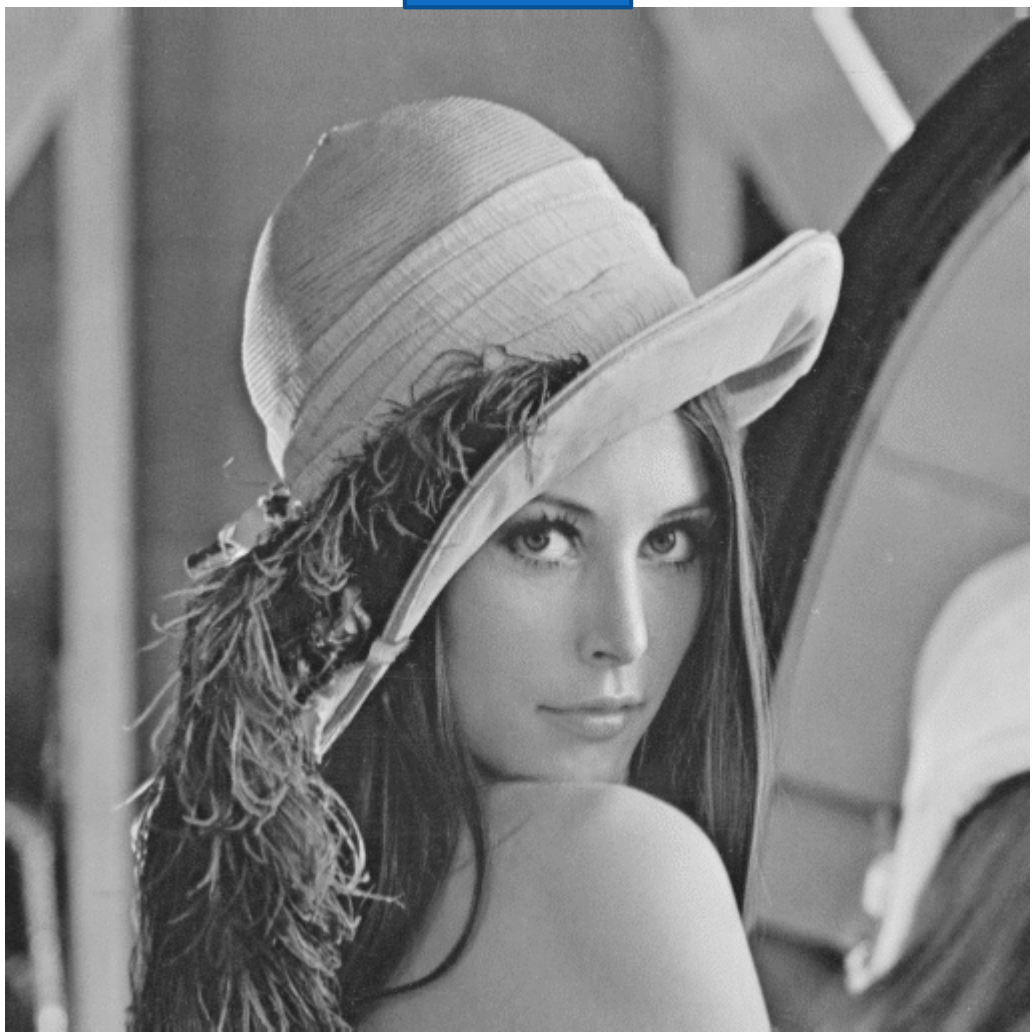


Reconstructed at CR=20



VISUAL ILLUSTRATION OF THE PERFORMANCE OF JPEG ON AN ACTUAL IMAGE (4/4)

Original



Reconstructed at CR=32

OBSERVATIONS ABOUT JPEG PERFORMANCE AT HIGH COMPRESSION RATIOS

- What can you observe at the high compression ratio of 32?
  - Poorer quality, fuzzy
  - What else?
    - Blockiness
  - Why?
    - Because the blocks were processed (compressed) mostly independently from one another, so at high compression ratio, the block boundaries show
- Remedies?
  - Overlapping blocks (possibly, but not in JPEG)
  - DCT on whole image: slow and results in low quality (due to non-localization)
  - A different transform? Yes, wavelets/sub-band coding=JPEG2000

EXTENDED JPEG

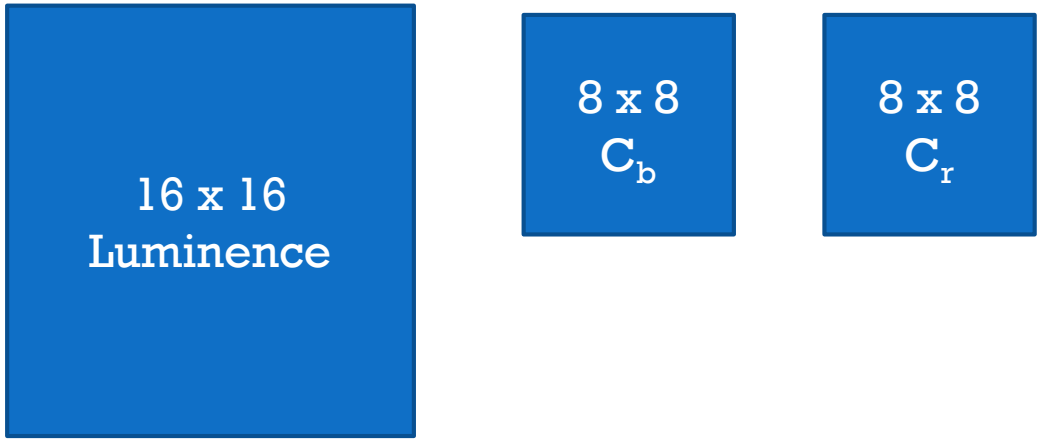
- Extended JPEG allows for several optional enhancements:
  - 12-bit/pixel input
  - Arithmetic coding is allowed as an alternative to Huffman coding
  - Adaptive quantization: Allows 5-bit scale change to the quantization matrix from one block to another
  - SPIFF: A file format that provides for the interchange of compressed image files between different application environments
  - Progressive transmission (PT)
    - Sequential PT: Spectral selection, Successive approximations
    - Hierarchical PT (pyramid encoding)
  - Tiling
  - Selective refinement

MPEG (1 & 2): BASIC CONCEPTS (1/2)

- Video is a sequence of images called frames
- Color
  - Three components: red (R), green (G), and blue (B)
  - For compatibility with non-colored media, the RGB model was converted to an equivalent model --- YC_bC_r
    - Y is the luminance component, which was experimentally determined to be
      $$Y = 0.299R + 0.587G + 0.114B$$
    - $C_b = B - Y$
    - $C_r = R - Y$
  - Y is referred to as *luma*, and C_b & C_r as *chroma*
- Every frame is really 3 images: one Y, one C_b and one C_r

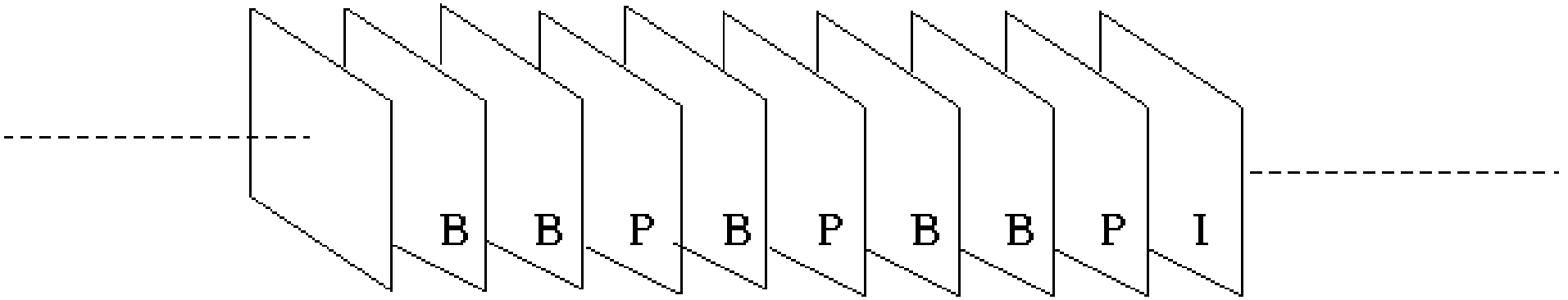
MPEG (1 & 2): BASIC CONCEPTS (2/2)

- Every frame is really 3 images: one Y, one C_b and one C_r
- 8 bits/pixel for each of the three color components
- HVS is less sensitive to color ⇒ the C_b and C_r images are downsampled by 2 in each dimension (every other row and every other column are dropped from C_b and C_r , so they're each ¼ the size of the luma Y)
- Much of MPEG processing is on the basis of a **macroblock**:
  - A 16×16 luminance block with the two 8×8 associated chroma blocks



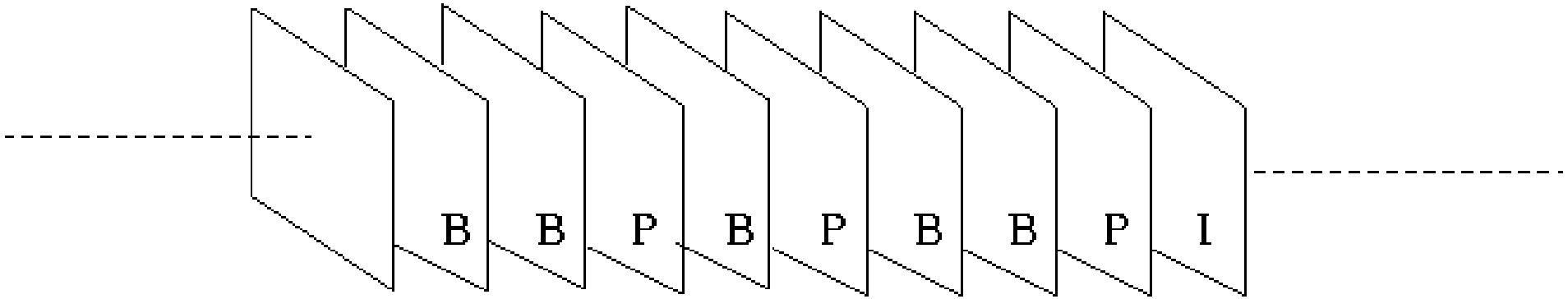
TYPES OF FRAMES IN MPEG (1/4)

- MPEG has 3 types of frames: I, P, and B



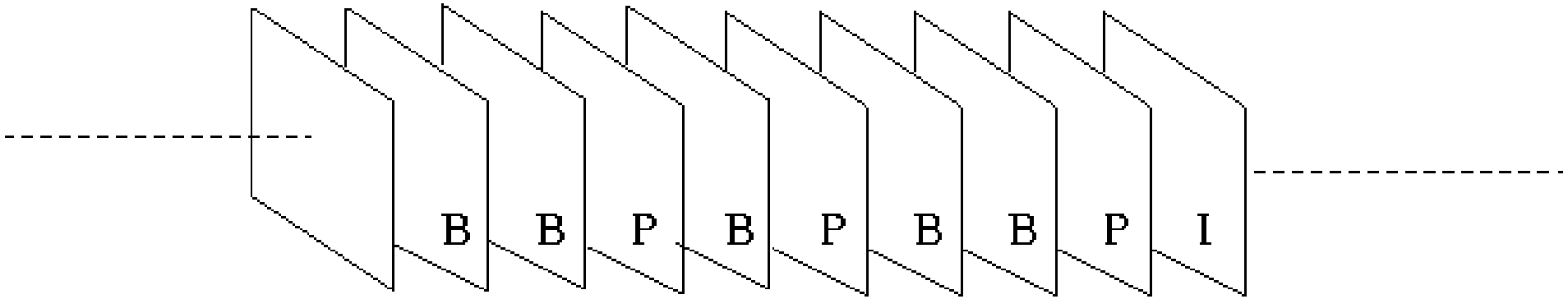
- Each Frame in a video is labeled by I or P or B, explained soon
- The user of MPEG has the responsibility of labeling the frames into I, P or B
- Algorithms for doing "optimal" labeling of the frames have been developed by researchers and implementers
- The relative numbers of I, P and B frames are arbitrary

TYPES OF FRAMES IN MPEG (2/4)



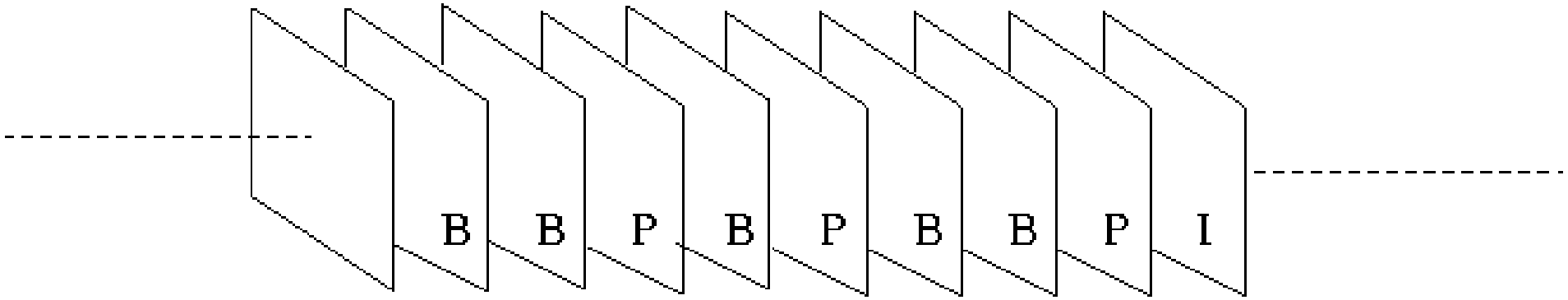
- I frames are strictly *intra-compressed* as in JPEG
  - Intra-compressed means that an I-frame is coded completely independently of any other frame
  - The purpose of I frames is to provide random access points to the video, like for fast forward/rewind
  - An I frame must occur at least once every 132 frames to provide user-acceptable speed of random access to various parts of a video

TYPES OF FRAMES IN MPEG (3/4)



- P frames are motion-compensated forward-predictive-coded frames
  - They are *interframe-compressed, which means:*
    - Residuals between frames are computed and coded
    - Actually, it is residuals between macroblocks (in different frames) which are computed and coded
  - We'll see what "motion-compensated means", and what "forward-predictive" means
  - P frames typically provide more compression than I frames

TYPES OF FRAMES IN MPEG (4/4)



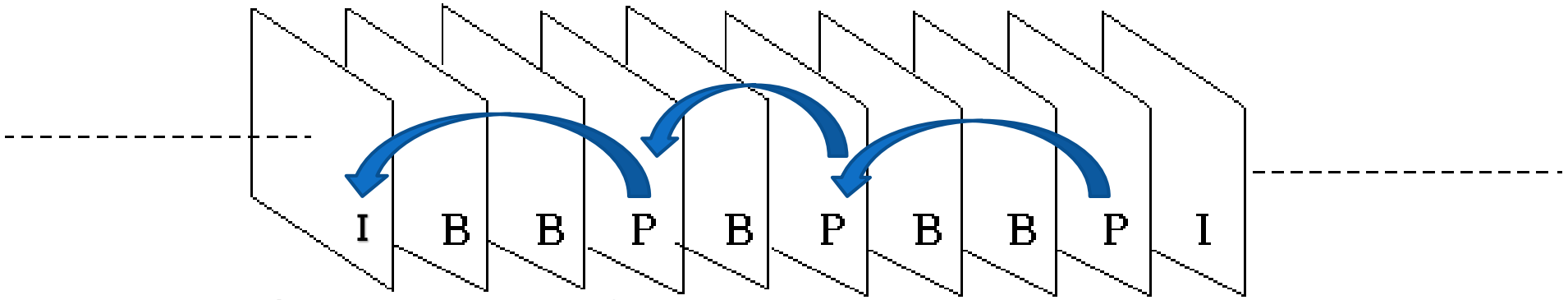
- B frames are motion-compensated **b**idirectionally-predictive-coded frames;
  - They are interframe-compressed
  - We'll see what "bidirectionally-predictive" means
  - They typically provide the most compression

INTERFRAME COMPRESSION OF P/B MACROBLOCKS: A ROUGH IDEA

- Let F be a P/B frame
- For each macroblock M of F, computer a motion-compensated prediction/approximation $M'\ of\ \ M$ (we'll see how later)
  - If the residual $M - M'$ is small enough, compress it in a JPEG-like style
  - Otherwise, intra-compress macroblock M in a JPEG-like style
- Issues to be addressed
  - Method of strict intra-compression
  - Method of compressing residual macroblocks
  - Motion-compensated prediction

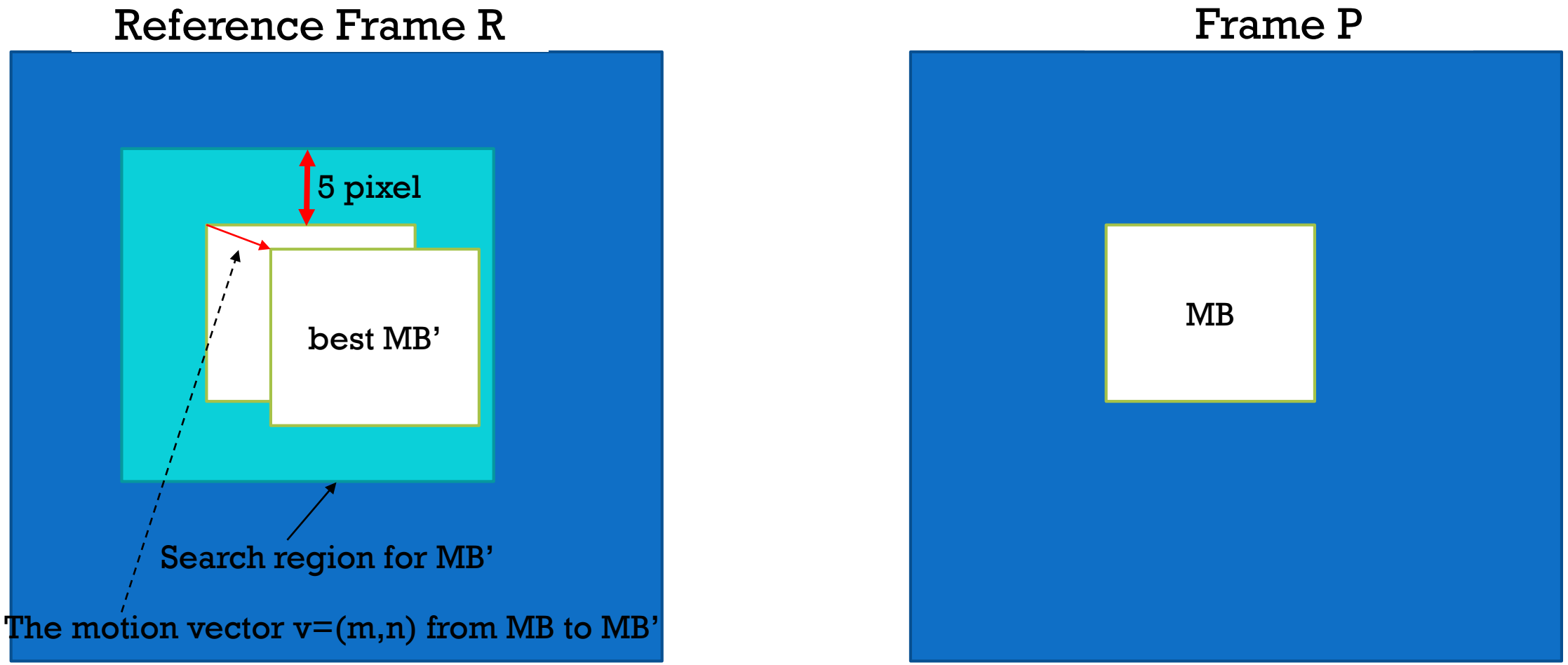
MOTION-ESTIMATION AND PREDICTION (1/4)

-- FOR P FRAMES --



- P will be predicted/approximated from one single *reference frame* R. Where is R?
- R is the **most recent (decoded) I or P frame** before the current P frame
- For each macroblock **MB** of P, find the best-matching macroblock MB′ in the reference frame R
  - One way is to look in R in the vicinity of the same position of **MB** but in R
  - Search exhaustively up to 5-10 pixels away in each direction
  - For each candidate **MB′**, compute MSE(**MB**, **MB′**)
  - From among all candidates, select the **MB′** with least MSE(**MB**, **MB′**)

A WAY FOR SEARCHING FOR BEST MATCH IN A REFERENCE FRAME



- Motion estimation is done per macroblock, using the 16×16 luminance part only
- Motion is assumed **uniform** across all the pixels of a macroblock: ***rigid motion***

MOTION-ESTIMATION AND PREDICTION (2/4)
-- FOR P FRAMES --

- If the MB-to-MB' match is satisfactory, then
  - Treat MB' as the prediction (i.e., approximation) of MB
  - Record the motion (i.e., displacement) vector between the two macroblocks (allowing half-pixel accuracy)
  - Compute and compress the macroblock residual MB-MB' (luma and chroma)
- If the MB-to-MB' match is found to be unsatisfactory, then the macroblock MB is strictly intra-compressed as is done in I frames

MOTION-ESTIMATION AND PREDICTION (3/4)
-- FOR P FRAMES --

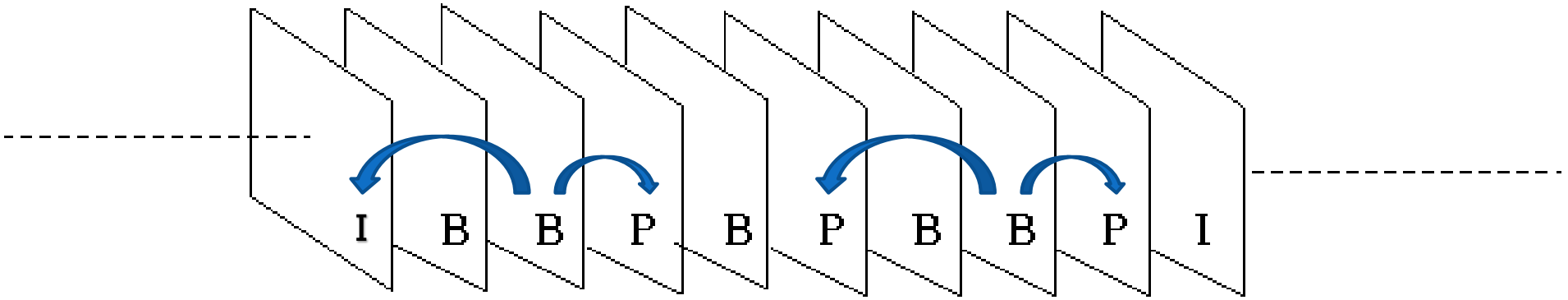
- The motion vectors of all the macroblocks of P exhibit redundancy due to similar (or sometimes identical) motion experienced by many neighboring macroblocks
  - For example, a large rigid body in the frame can cover many macroblocks
  - All those macroblocks will move identically from frame to frame
  - Therefore, in the predictions from the reference frames, the corresponding motion vectors will be identical
  - If the large body is instead nearly-rigid (like a human hand/face), the motion vectors of the corresponding macroblocks are nearly identical
- This redundancy is exploited by coding the consecutive **differential** values of motion vectors (i.e., DPCM)

MOTION-ESTIMATION AND PREDICTION (4/4)
-- FOR P FRAMES --

- Remarks
  1. MPEG does not standardize the decision mechanism for judging whether or not a match between two macroblocks is satisfactory
     - It is up to the user/implementer
  2. MPEG doesn't standardize (nor care) how a matching MB' is found
     - What matters to the decoder is the location of MB', i.e, motion vector
     - So, motion compensation algorithms are up to the implementers
  3. A typical decision mechanism involves computing an error measure between the luminance of the two macroblocks
     - The match is satisfactory if the error is below a certain threshold.
     - Possible error measures include mean-square error (MSE), mean absolute-difference error (MAD), and variance(MB-MB')/variance(MB).

MOTION-ESTIMATION AND PREDICTION (1/2)

-- FOR B FRAMES --



- Consider a B-frame B
- B will be predicted/approximated from TWO reference frames R1 & R2
- R1 is the most recent (decoded) past I/P frame
- R2 is the nearest (decoded) future I/P frame
- For each macroblock MB of B, find the best-matching macroblock MB_1 in R1, and the closest matching macroblock MB_2 in R2

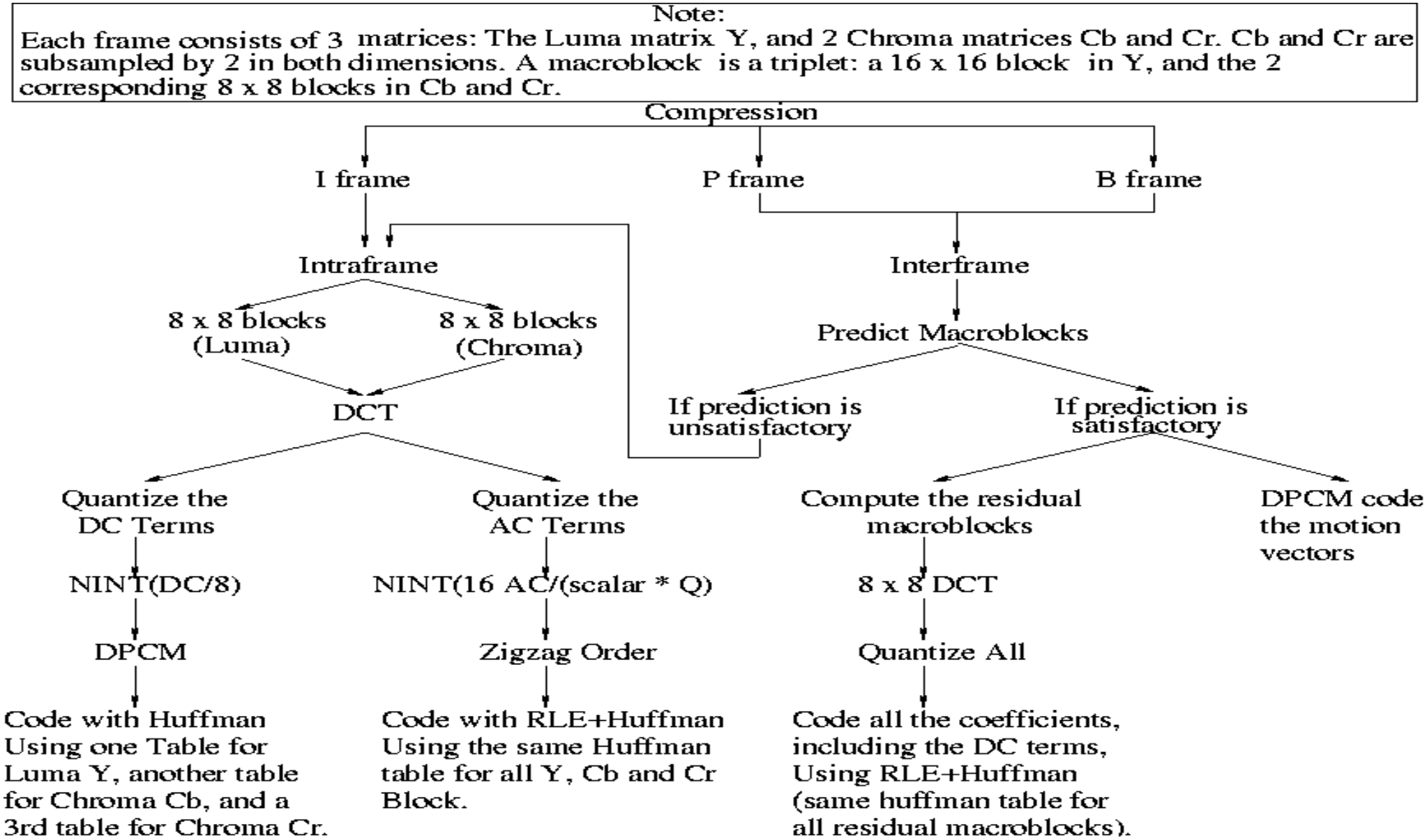
MOTION-ESTIMATION AND PREDICTION (2/2)
-- FOR B FRAMES --

- The predicted macroblock is $\mathrm{PM} = \mathrm{NINT}(\alpha_1 \mathrm{MB}_1 + \alpha_2 \mathrm{MB}_2)$
  - $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = \frac{1}{2}$ if both matches are satisfactory
  - $\alpha_1 = 1$ and $\alpha_2 = 0$ if only MB_1 match is satisfactory
  - $\alpha_1 = 0$ and $\alpha_2 = 1$ if only MB_2 match s satisfactory
  - $\alpha_1 = 0$ and $\alpha_2 = 0$ if neither match is satisfactory

Remark: The prediction mode chosen is 2-bit coded and passed on along with the macroblock header information

- Compute & compress the macroblock residual MB−PM (luma and chroma)
- If neither match is found to be satisfactory, then the macroblock MB is strictly intra-compressed as is done in I frames
- Record and DPCM-encode the motion vector(s) between the MB and MB_1 and/or MB_2 (allowing half-pixel accuracy)

FLOWCHART OF MPEG COMPRESSION



MPEG2 (1/4)

- Higher data rates than MPEG1
- MPEG2 allows for higher quality source images
  - 4:2:0 (chroma subsamples only horizontally)
  - 4:4:4 (no subsampling of chroma)
  - Note: 4:2:2, i.e., (chroma subsamples horizontally and vertically) is what's supported by MPEG1
- MPEG2 allows for finer quantization and for specifying separate quantization tables for luma and chroma
- MPEG2 allows for finer adjustment of quantization scale factor, used in intra compression
- MPEG2 allows for interlaced video

MPEG2 (2/4)

- MPEG2 supports error concealment (of lost macroblocks)
- MPEG2 supports scalable compression
  - SNR-scalability: by sending bands of DCT coefficients
  - spatial-scalability: pixel resolution by down-/up-sampling
  - temporal-scalability: different frame rates by skipping frames

MPEG2 (3/4)

- MPEG2 has a Profile and Level structure
  - Profiles are algorithmic elements included in MPEG2
  - Levels are upper bounds on parameter values
- Profiles (profiles are backward-compatible)
  - Simple: No use of B frames
  - Main: what was described earlier, but no scalability
  - SNR-scalable
  - Spatially scalable
  - High: temporally scalable, higher-quality source data (4:2:0, 4:2:2 and possibly 4:4:4 in the future)

MPEG2 (4/4)

- Levels
  - Low: 352 × 240 frame size
  - Main: 720 × 480 frame size
  - High 1440: 1440 × 1152 frame size
  - Very High: 1920 × 1080 frame size

REFERENCES

- B. pennebaker and J. L. Mitchell, JPEG Still Image Data Compression Standard, Van Nostrand reinhold, New York 1993.
- ISO-11172-2: Generic Coding of moving pictures and associated audio (MPEG-1)
- ISO-13818-2: Generic Coding of moving pictures and associated audio (MPEG-2)

LINKS TO OTHER STANDARDS

- JPEG 2000
- An overview of MPEG 4
- MPEG 2/MPEG 4 Audio Coding: Advanced Audio Coding (AAC)
- MPEG 1 Layer 3 Audio (MP3)
- Digital Audio Compression (AC-3) Standard and its 2001 Revision A

NEXT LECTURE

- Vector quantization