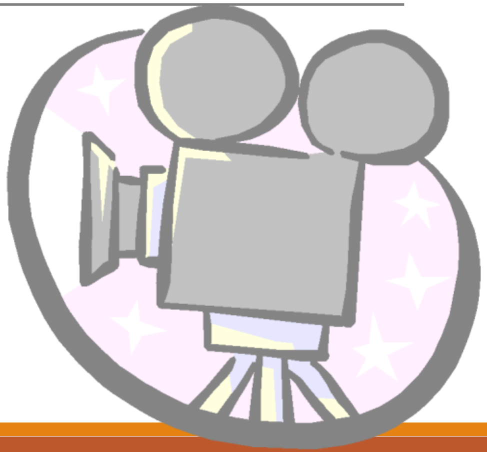
Image Processing

CS-317/CS-341

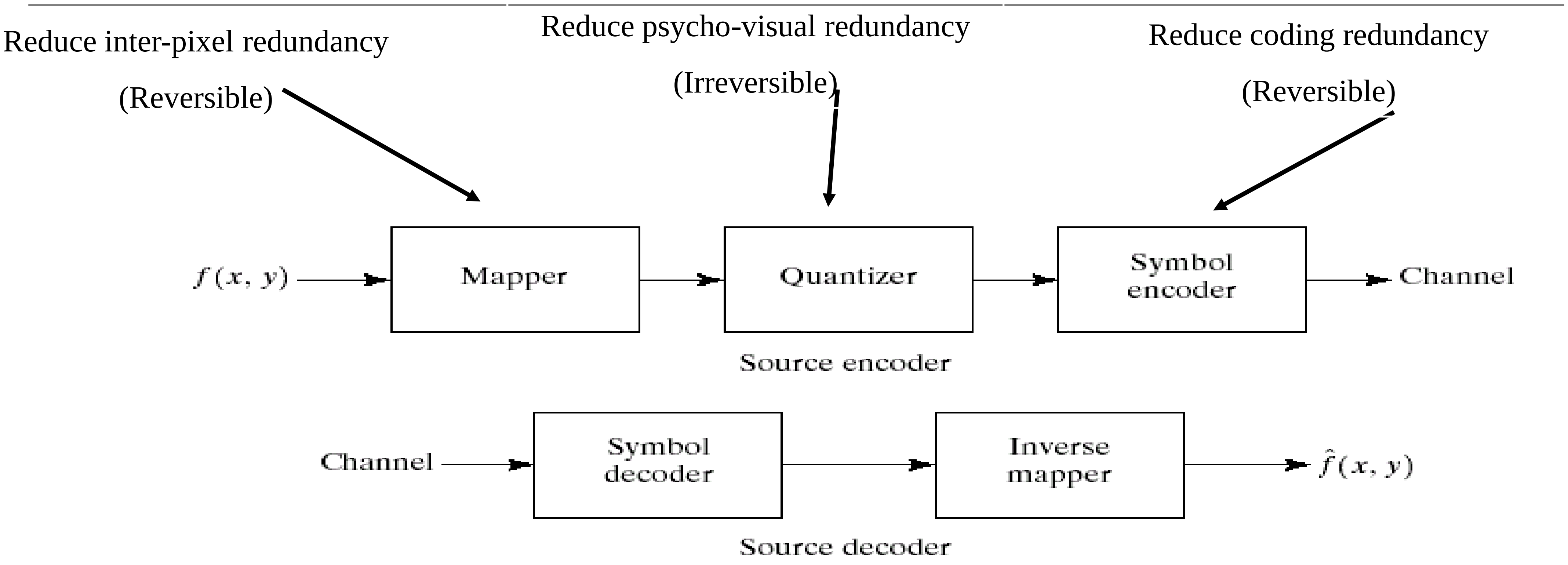


Outline

- Lossy image compression techniques

- JPEG Standard

Image Compression model



Reduce inter-pixel redundancy

(Reversible)

Reduce psycho-visual redundancy

(Irreversible)

Reduce coding redundancy

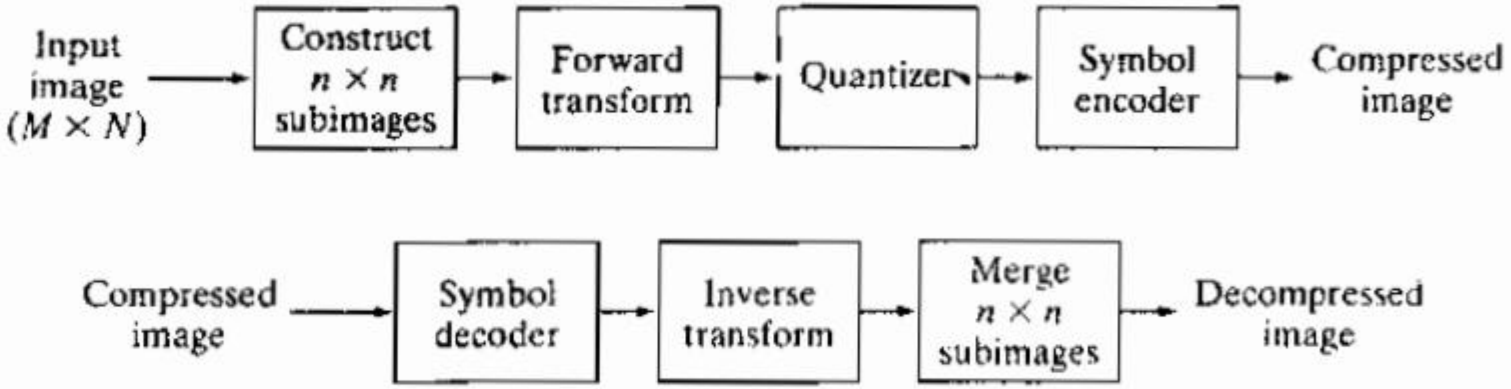
(Reversible)

(a) Source encoder and (b) source decoder model.

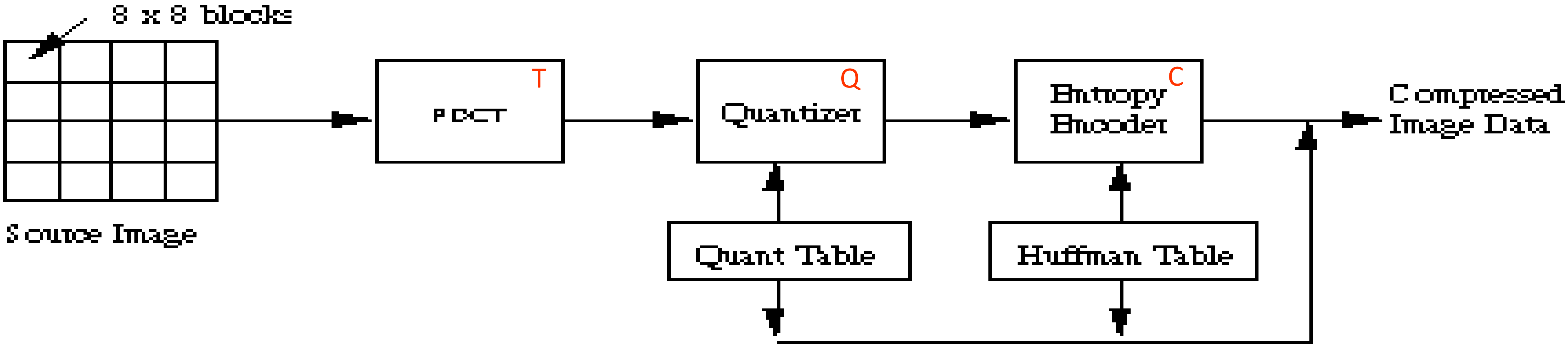
Block Transform Coding

a
b

FIGURE 8.21 A block transform coding system: (a) encoder; (b) decoder.



JPEG Coding Algorithm



Flow-chart diagram of DCT-based coding algorithm specified by Joint Photographic Expert Group (JPEG)

JPEG - Steps

1. Divide image into 8x8 subimages.

For each subimage do:

2. Shift the gray-levels in the range [-128, 127]

3. Apply DCT → 64 coefficients

    1 DC coefficient: F(0,0)

    63 AC coefficients: F(u,v)

4. Quantization

5. Coding

Transform Coding of Images

Why not transform the whole image together?

- Require a large memory to store transform matrix
- It is not a good idea for compression due to spatially varying statistics within an image

Idea of partitioning an image into blocks

- Each block is viewed as a smaller-image and processed independently
- It is not a magic, but a compromise

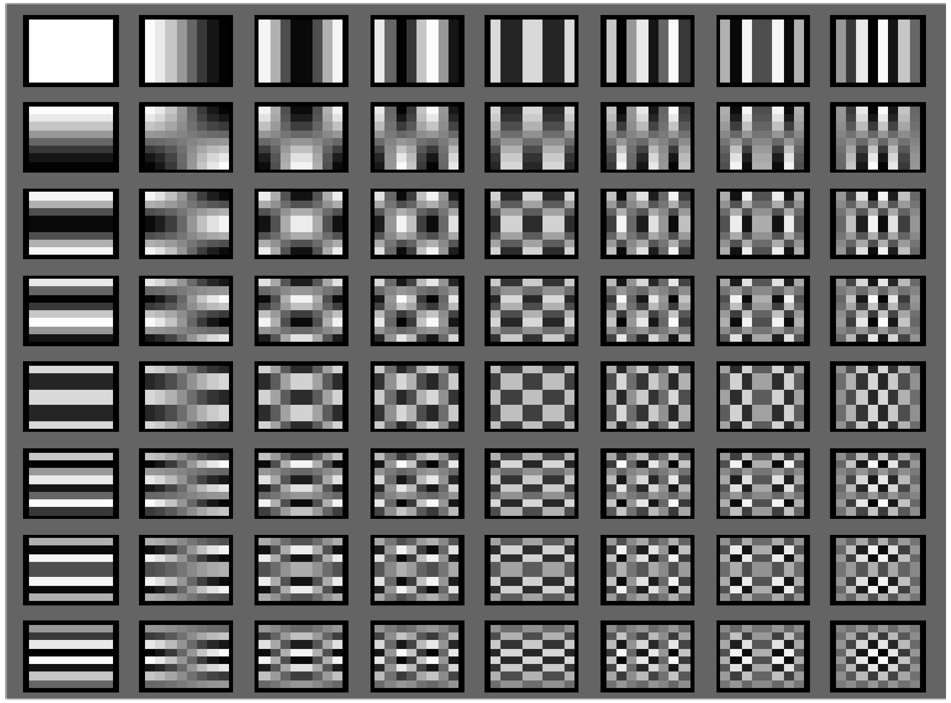
8-by-8 DCT Basis Images

$$\mathbf{A}_{8\times 8} = \begin{bmatrix} a_{11} & \cdots & \cdots & a_{18} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{81} & \cdots & \cdots & a_{88} \end{bmatrix}$$

$$a_{kl} = \begin{cases} {}^{1}\!/\!_{\sqrt{8}}, & k = 1, 1 \le l \le 8 \\ \frac{1}{2}\cos\frac{(2l-1)(k-1)\pi}{16}, & 2 \le k \le 8, 1 \le l \le 8 \end{cases}$$

$$\mathbf{Y} = \sum_{i=1}^{8}\sum_{j=1}^{8} x_{ij}\mathbf{B}_{ij},$$

$$\mathbf{B}_{ij} = \vec{b}_i \vec{b}_j^{\,T}, \vec{b}_i = [a_{i1}, \ldots, a_{i8}]^T$$



Block Processing under MATLAB

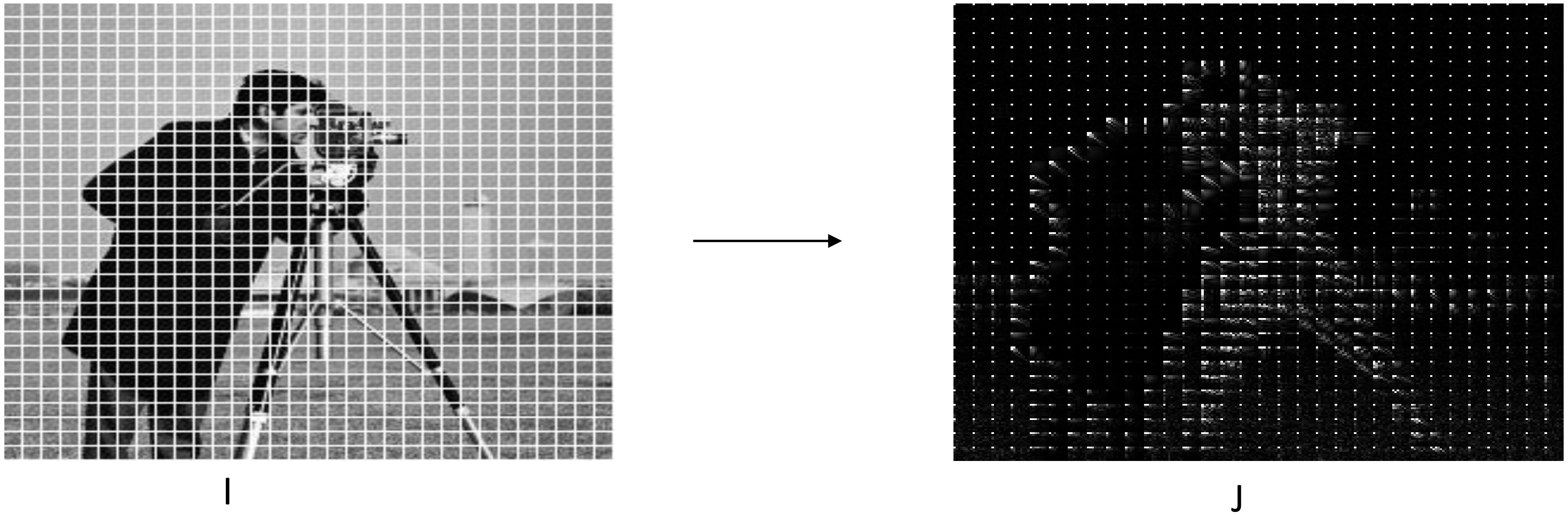
Type “help blkproc” to learn the usage of this function

- B = BLKPROC(A,[M N],FUN) processes the image A by applying the function FUN to each distinct M-by-N block of A, padding A with zeros if necessary.

Example

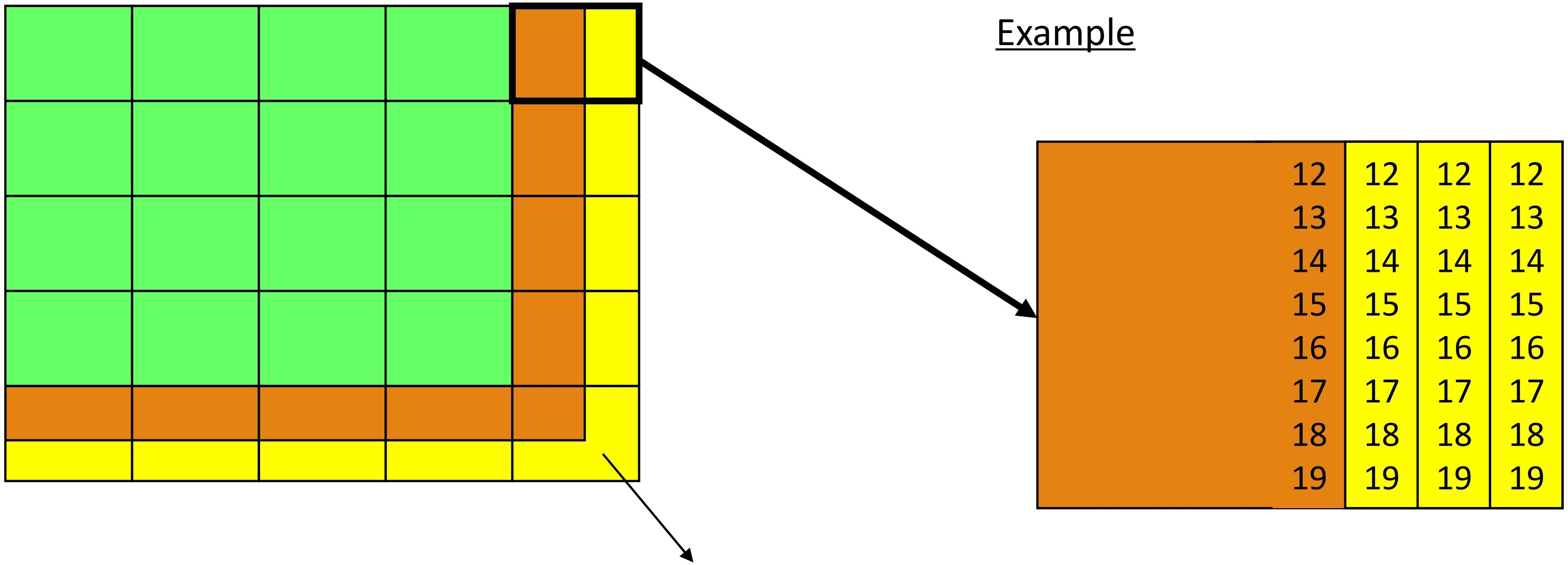
```
I = imread('cameraman.tif');
        fun = @dct2;
        J = blkproc(I,[8 8],fun);
```

Block-based DCT Example



note that white lines are artificially added to the border of each 8-by-8 block to denote that each block is processed independently

Boundary Padding



padded regions

Example

12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19

When the width/height of an image is not the multiple of 8, the boundary is artificially padded with repeated columns/rows to make them multiple of 8

Example

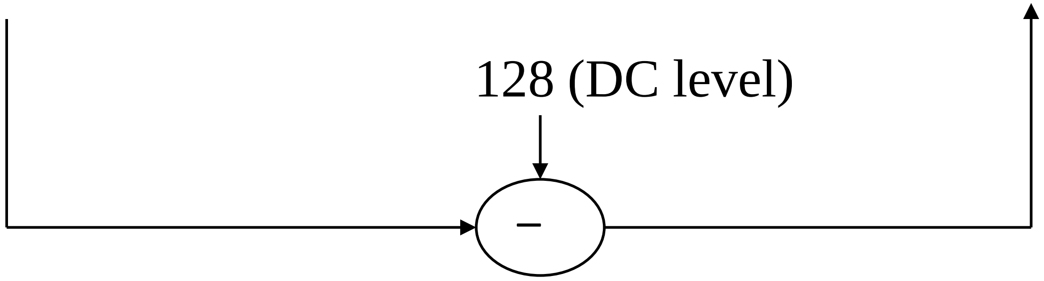
$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

Any 8-by-8 block in an image is processed in a similar fashion

Encoding Stage I: Transform

- Step 1: DC level shifting

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix} \qquad \begin{bmatrix} 55 & 36 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$



Encoding Step 1: Transform (Con’t)

- Step 2: 8-by-8 DCT

$$\begin{tikzcd}
\begin{bmatrix} 55 & 36 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix} \arrow[r] & \boxed{\begin{matrix} 8\times8 \\ \text{DCT} \end{matrix}} \arrow[r] & \begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}
\end{tikzcd}$$

Encoding Stage II: Quantization

<u>Q-table</u> : specifies quantization stepsize (see slide #28)

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

$$f : s_{ij} = \left[\frac{x_{ij}}{Q_{ij}} \right]$$

$$f^{-1} : \hat{x}_{ij} = s_{ij} \cdot Q_{ij}$$

$$1 \le i, j \le 8$$

<u>Notes:</u>

- Q-table can be specified by customer
- Q-table is scaled up/down by a chosen quality factor
- Quantization stepsize Q_{ij} is dependent on the coordinates (i,j) within the 8-by-8 block
- Quantization stepsize Q_{ij} increases from top-left to bottom-right

Encoding Stage II: Quantization (Con’t)

<u>Example</u>

$$x_{ij}$$

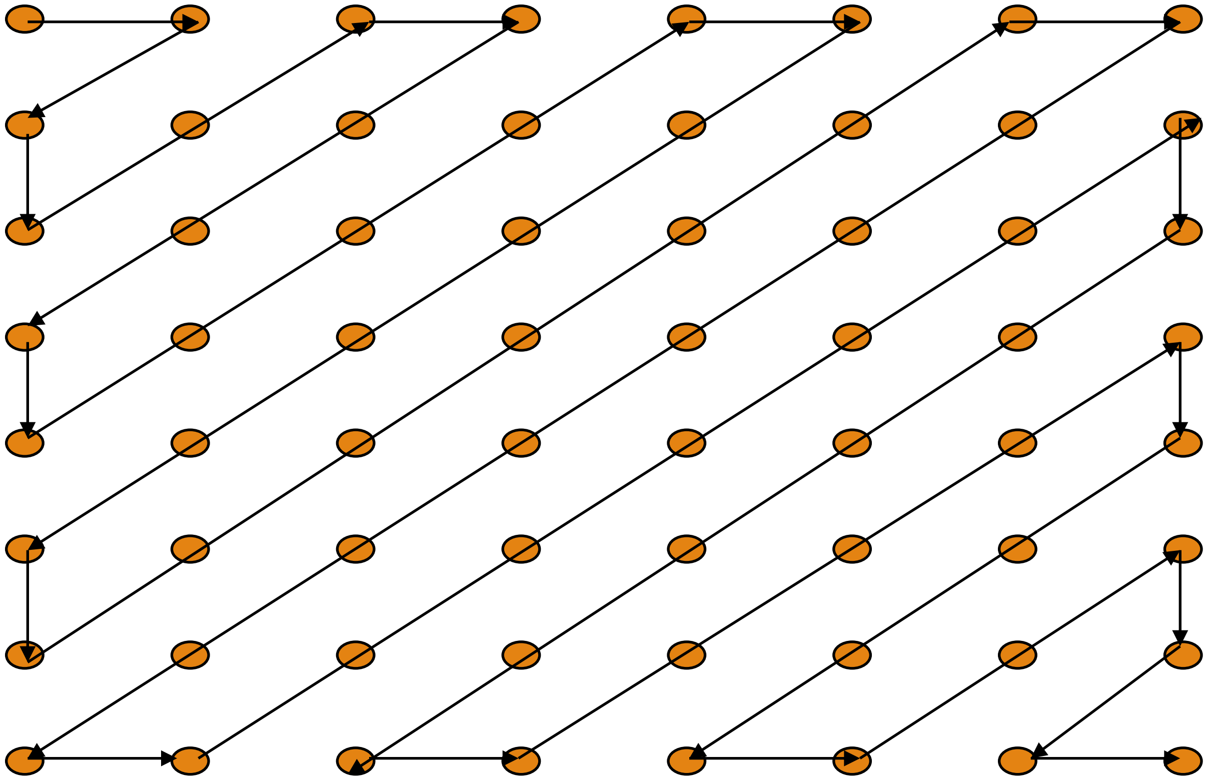
$$\begin{bmatrix}
313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\
-38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\
-20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\
-10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\
-6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\
2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\
4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\
3 & 1 & 0 & -4 & -2 & -1 & 3 & 1
\end{bmatrix}$$

$$s_{ij}$$

$$\begin{bmatrix}
20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\
-3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\
-1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$x_{ij} \xrightarrow{\boxed{f}} s_{ij}$$

Encoding Stage III: Entropy Coding



Zigzag Scan

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

zigzag scan

(20,5,-3,-1,-2,-3,1,1,-1,-1,
0,0,1,2,3,-2,1,1,0,0,0,0,0,
0,1,1,0,1,EOB)

End Of the Block:

All following coefficients are zero

Run-length Coding

(20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

↑ DC coefficient

↑ AC coefficient

- DC coefficient : DPCM coding → encoded bit stream
- AC coefficient : run-length coding (run, level)

(5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

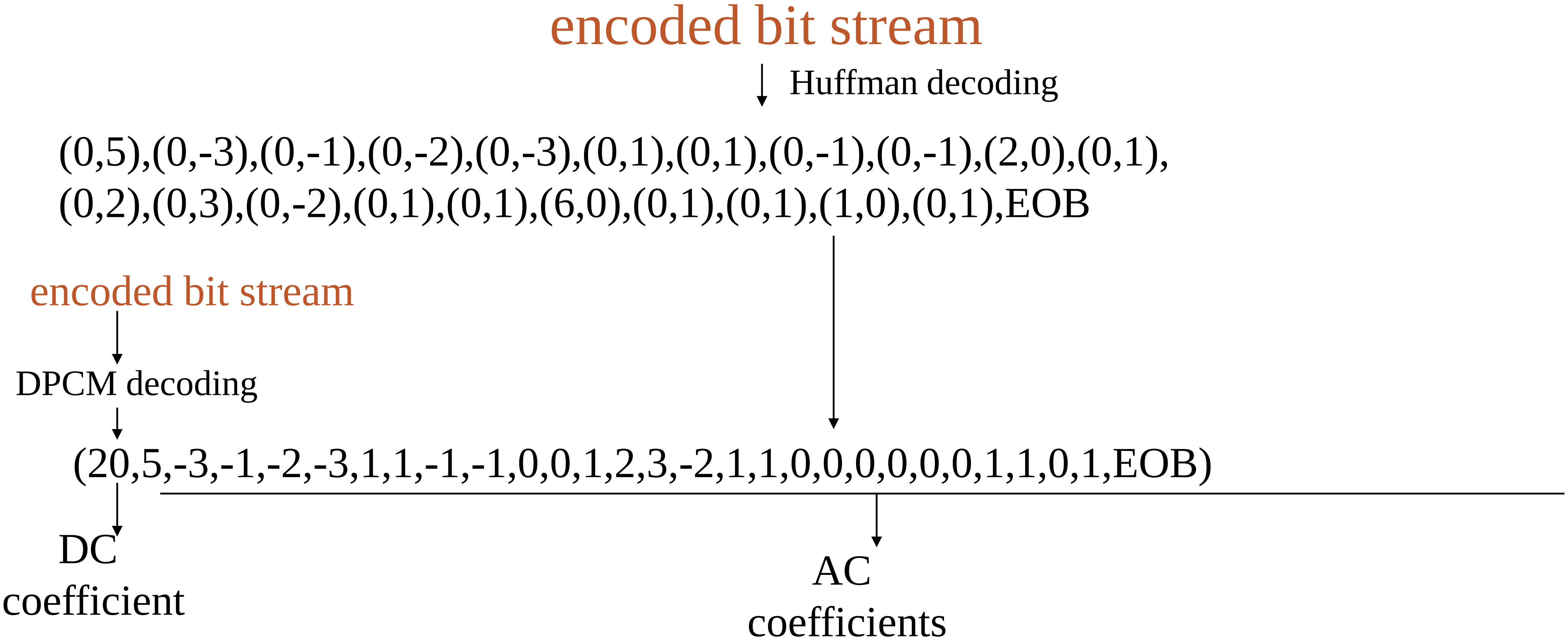
↓

(0,5),(0,-3),(0,-1),(0,-2),(0,-3),(0,1),(0,1),(0,-1),(0,-1),(2,0),(0,1),
(0,2),(0,3),(0,-2),(0,1),(0,1),(6,0),(0,1),(0,1),(1,0),(0,1),EOB

↓ Huffman coding

encoded bit stream

JPEG Decoding Stage I: Entropy Decoding



encoded bit stream

Huffman decoding

(0,5),(0,-3),(0,-1),(0,-2),(0,-3),(0,1),(0,1),(0,-1),(0,-1),(2,0),(0,1),
(0,2),(0,3),(0,-2),(0,1),(0,1),(6,0),(0,1),(0,1),(1,0),(0,1),EOB

encoded bit stream

DPCM decoding

(20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

DC coefficient

AC coefficients

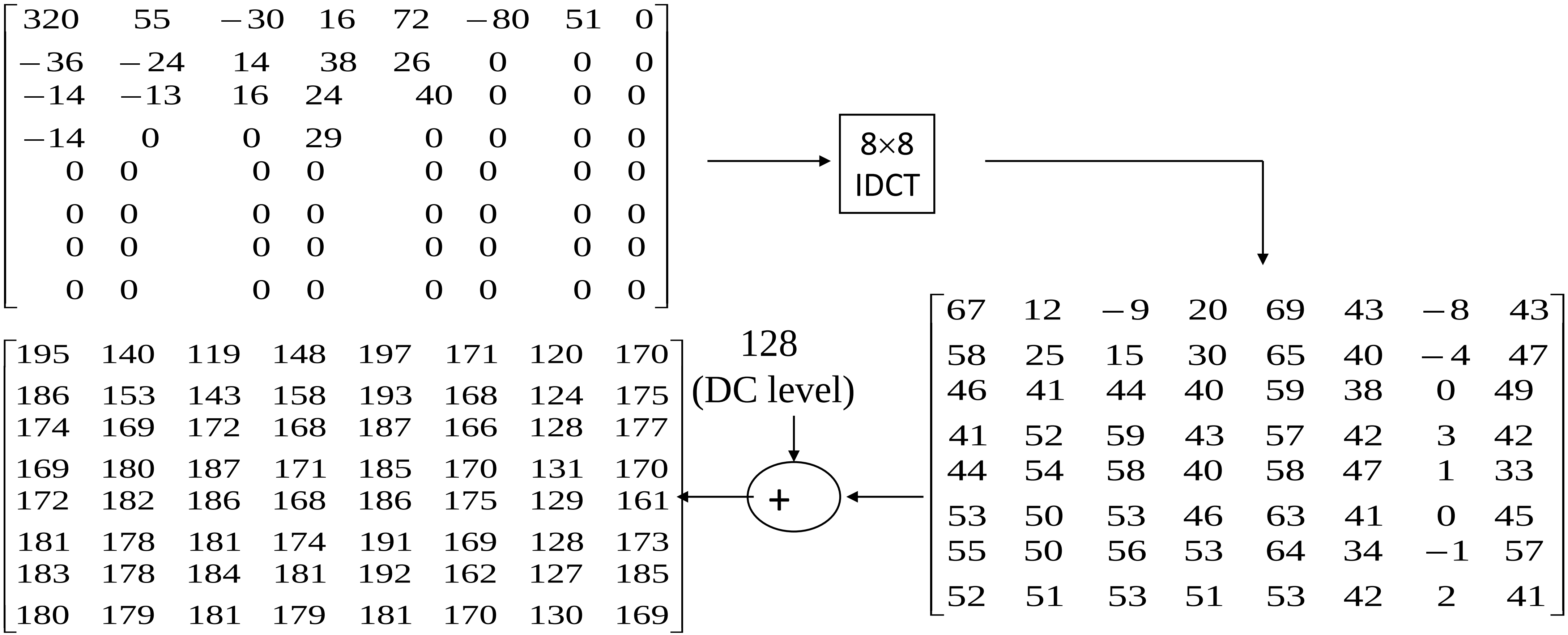
JPEG Decoding Stage II: Inverse Quantization

(20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

$$\Big\downarrow \text{zigzag}$$

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\boxed{f^{-1}}} \begin{bmatrix} 320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 \\ -36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\ -14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\ -14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Decoding Stage III: Inverse Transform



$$\begin{bmatrix} 320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 \\ -36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\ -14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\ -14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

8×8 IDCT

$$\begin{bmatrix} 67 & 12 & -9 & 20 & 69 & 43 & -8 & 43 \\ 58 & 25 & 15 & 30 & 65 & 40 & -4 & 47 \\ 46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\ 41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\ 44 & 54 & 58 & 40 & 58 & 47 & 1 & 33 \\ 53 & 50 & 53 & 46 & 63 & 41 & 0 & 45 \\ 55 & 50 & 56 & 53 & 64 & 34 & -1 & 57 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

128
(DC level)

+

$$\begin{bmatrix} 195 & 140 & 119 & 148 & 197 & 171 & 120 & 170 \\ 186 & 153 & 143 & 158 & 193 & 168 & 124 & 175 \\ 174 & 169 & 172 & 168 & 187 & 166 & 128 & 177 \\ 169 & 180 & 187 & 171 & 185 & 170 & 131 & 170 \\ 172 & 182 & 186 & 168 & 186 & 175 & 129 & 161 \\ 181 & 178 & 181 & 174 & 191 & 169 & 128 & 173 \\ 183 & 178 & 184 & 181 & 192 & 162 & 127 & 185 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

Quantization Noise

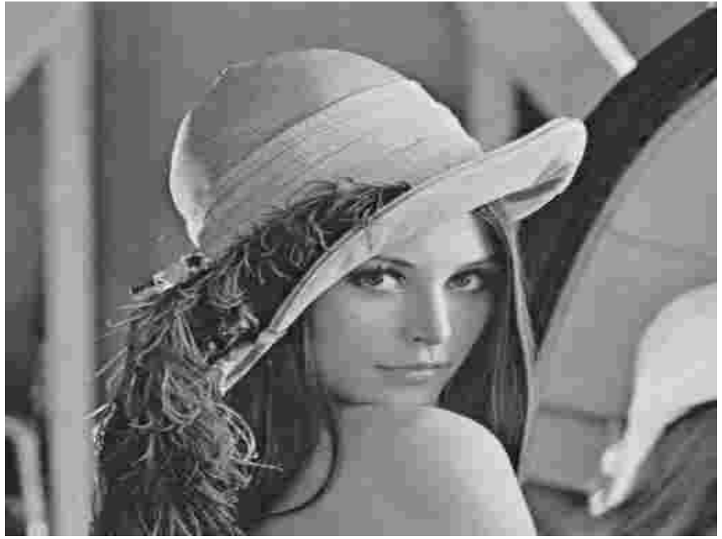
$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix} \quad \begin{bmatrix} 195 & 140 & 119 & 148 & 197 & 171 & 120 & 170 \\ 186 & 153 & 143 & 158 & 193 & 168 & 124 & 175 \\ 174 & 169 & 172 & 168 & 187 & 166 & 128 & 177 \\ 169 & 180 & 187 & 171 & 185 & 170 & 131 & 170 \\ 172 & 182 & 186 & 168 & 186 & 175 & 129 & 161 \\ 181 & 178 & 181 & 174 & 191 & 169 & 128 & 173 \\ 183 & 178 & 184 & 181 & 192 & 162 & 127 & 185 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

$\mathbf{X}$ $\qquad\qquad$ $\hat{\mathbf{X}}$

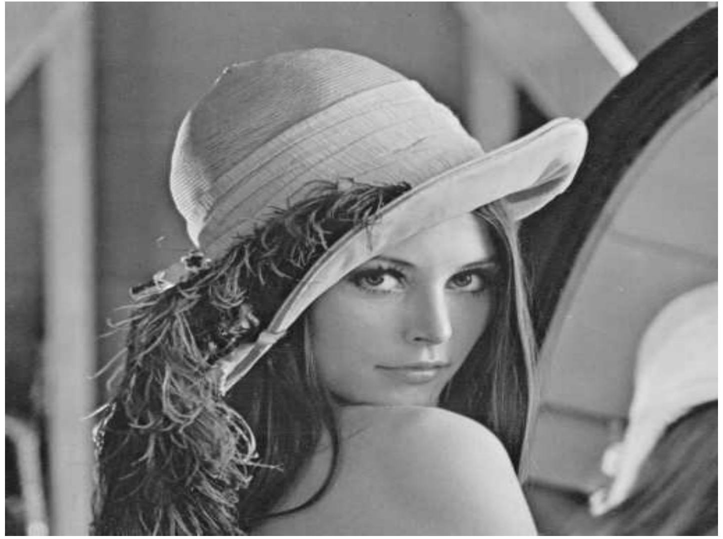
Distortion calculation: $\boxed{\text{MSE}=||\mathbf{X}-\hat{\mathbf{X}}||^2}$

Rate calculation: $\boxed{\text{Rate=length of encoded bit stream/number of pixels (bps)}}$

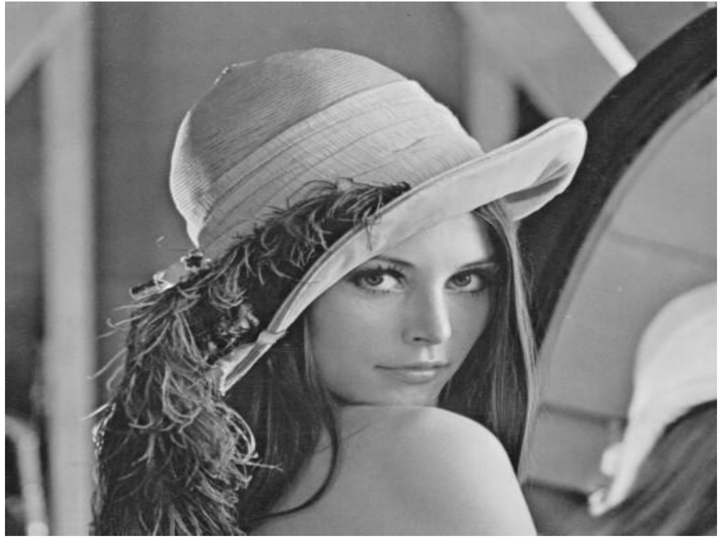
JPEG Examples



10 (8k bytes)



50 (21k bytes)



90 (58k bytes)

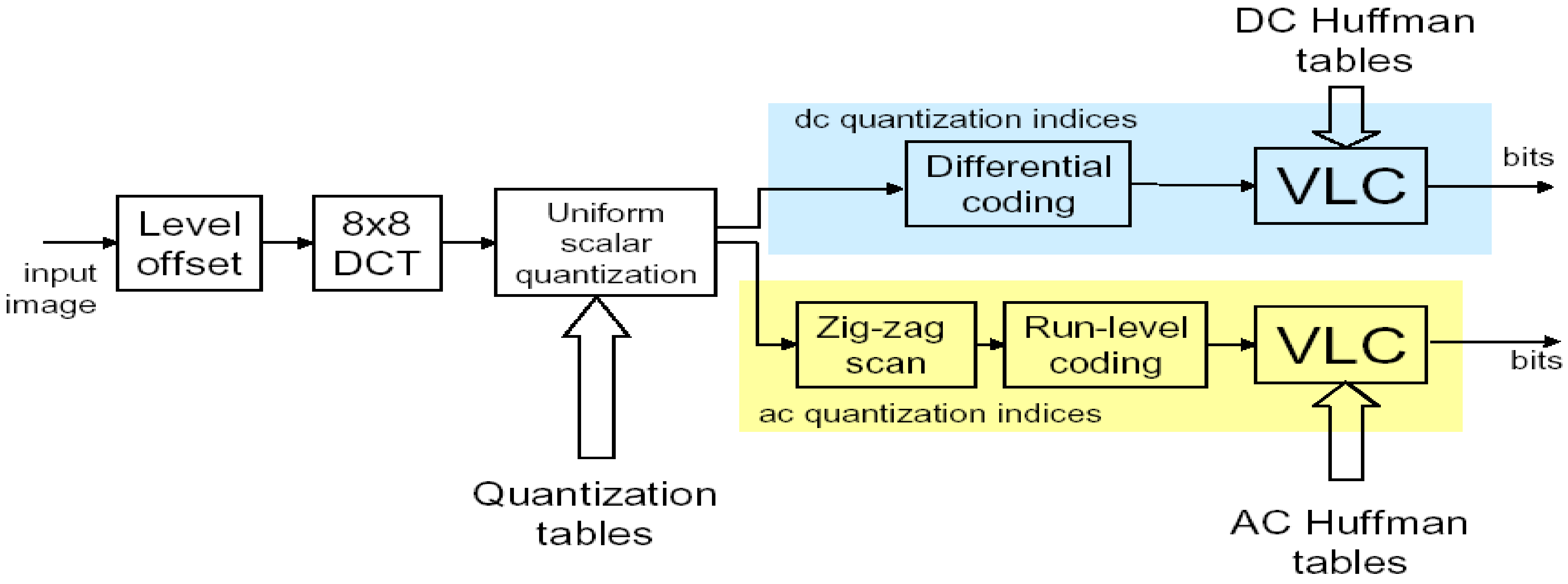
0 worst quality,

highest compression

best quality,

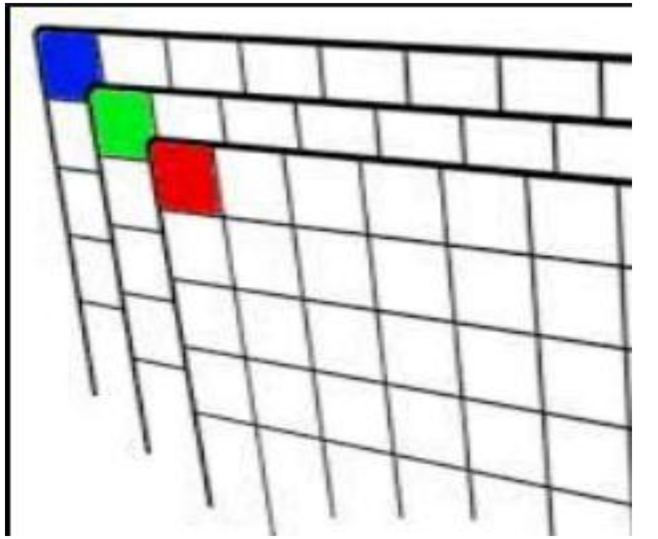
lowest compression 100

JPEG Coding Algorithm Summary



Color image

A **color image** has three values (or channels) per pixel and they measure the intensity and chrominance of light. The actual information stored in the digital **image** data is the brightness information in each spectral band.



Suggested Readings

- **Digital Image Processing by Rafel Gonzalez, Richard Woods, Pearson Education India, 2017.**

- **Fundamental of Digital image processing by A. K Jain, Pearson Education India, 2015.**

Thank you