Digital Image Processing

Image Compression

Relative Data Redundancy

- Let b and b’ denote the number of bits in two representations of the same information, the relative data redundancy R is

$$R = 1-1/C$$

C is called the compression ratio, defined as

$$C = b/b'$$

e.g., C = 10, the corresponding relative data redundancy of the larger representation is 0.9, indicating that 90% of its data is redundant

Why do we need compression?

- Data storage
- Data transmission

How can we implement compression?

- **Coding redundancy**

  Most 2-D intensity arrays contain more bits than are needed to represent the intensities

- **Spatial and temporal redundancy**

  Pixels of most 2-D intensity arrays are correlated spatially and video sequences are temporally correlated

- **Irrelevant information**

  Most 2-D intensity arrays contain information that is ignored by the human visual system

Examples of Redundancy

a b c

FIGURE 8.1 Computer generated 256 × 256 × 8 bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)

Coding Redundancy

TABLE 8.1
Example of variable-length coding.

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

The average number of bits required to represent each pixel is

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)\, p_r(r_k) = 0.25(2) + 0.47(1) + 0.24(3) + 0.03(3) = 1.81 bits$$

$$C = \frac{8}{1.81} \approx 4.42$$

$$R = 1 - 1/4.42 = 0.774$$

Spatial and Temporal Redundancy

1. All 256 intensities are equally probable.
2. The pixels along each line are identical.
3. The intensity of each line was selected randomly.

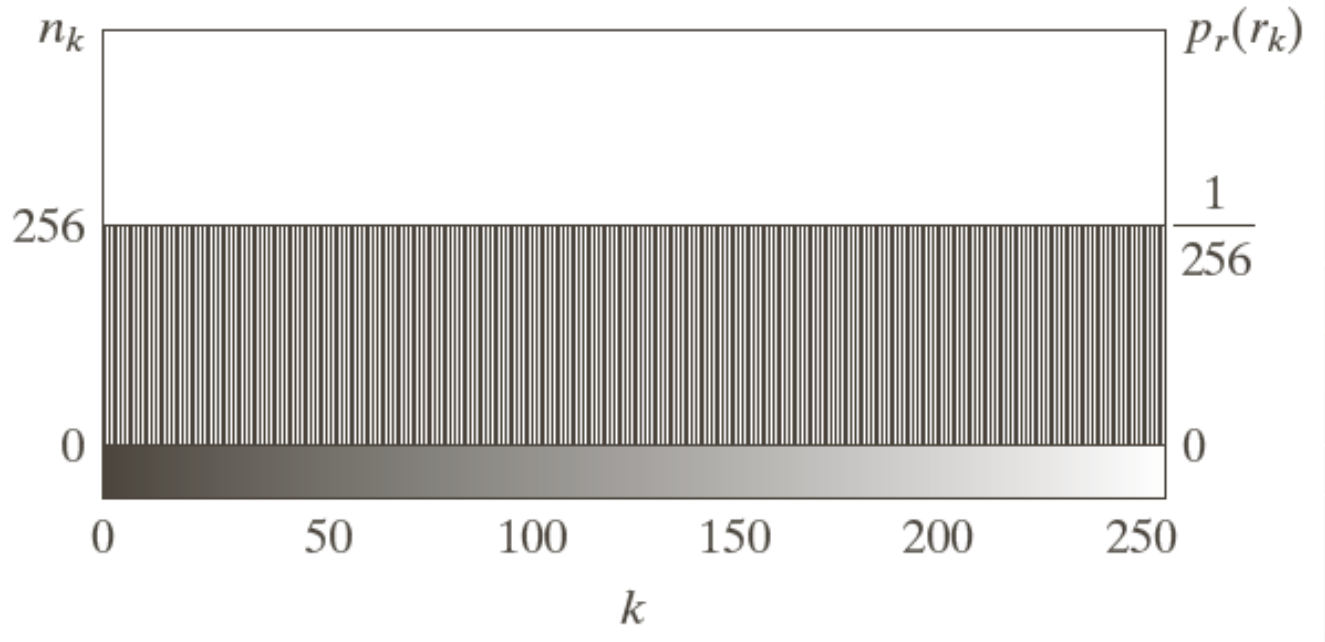


FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).

Spatial and Temporal Redundancy

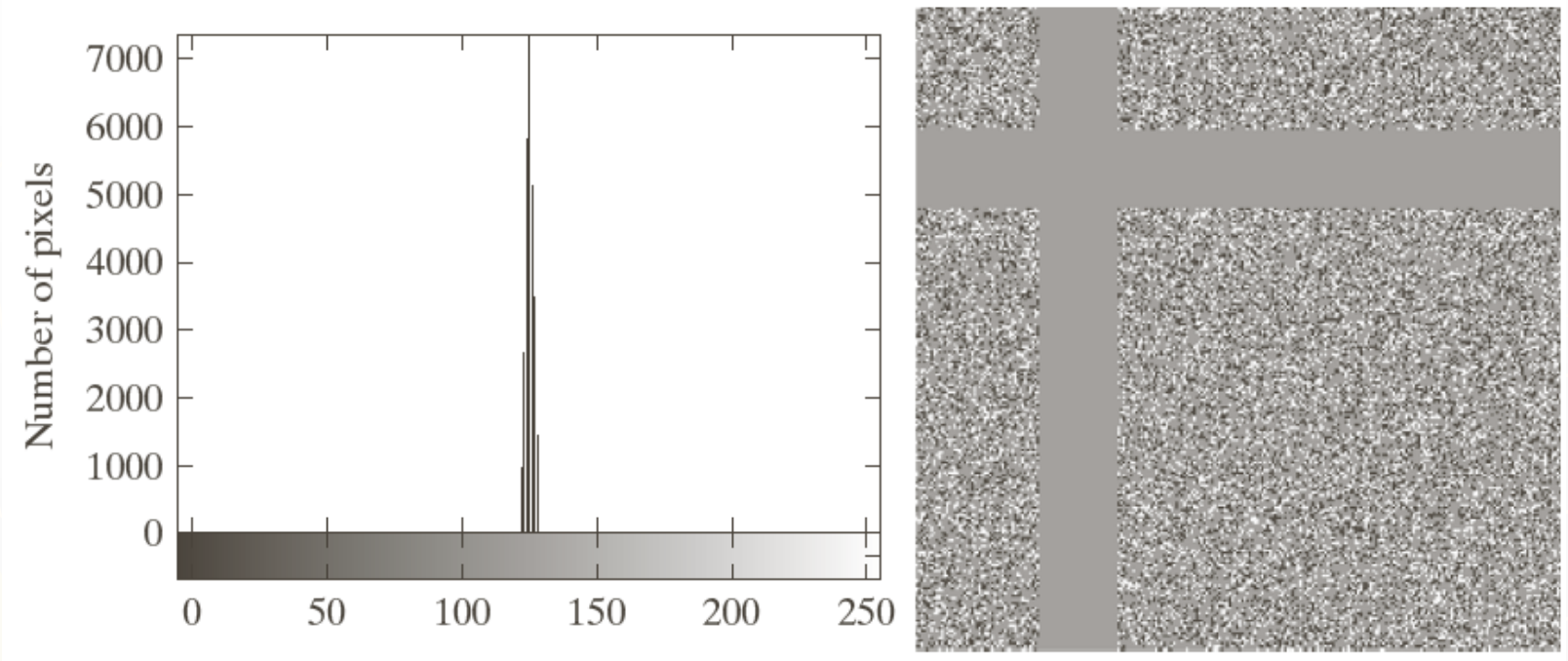
1. All 256 intensities are equally probable.
2. The pixels along each line are identical.
3. The intensity of each line was selected randomly.

Run-length pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity.
Each 256-pixel line of the original representation is replaced by a single 8-bit intensity value and length 256 in the run-length representation.

The compression ratio is

$$\frac{256 \times 256 \times 8}{(256 + 256) \times 8} = 128 : 1$$

Irrelevant Information



a b

FIGURE 8.3
(a) Histogram of the image in Fig. 8.1(c) and (b) a histogram equalized version of the image.

Measuring Image Information

A random event E with probability P(E) is said to contain

$$I(E) = \log \frac{1}{P(E)} = -\log P(E)$$

units of information.

Measuring Image Information

Given a source of statistically independent random events from a discrete set of possible events $\{a_1, a_2, \ ..., \ a_J\}$ with associated probabilities $\{P(a_1), \ P(a_2), \ ..., \ P(a_J)\}$, the average information per source output, called the entropy of the source

$$H = -\sum_{j=1}^{J} P(a_j) \log P(a_j)$$

a_j is called source symbols. Because they are statistically independent, the source called *zero – memory source*.

Measuring Image Information

If an image is considered to be the output of an imaginary zero-memory "intensity source", we can use the histogram of the observed image to estimate the symbol probabilities of the source. The intensity source's entropy becomes

$$\dot{H} = -\sum_{k=0}^{L-1} p_{\mathrm{r}}(r_k) \log p_{\mathrm{r}}(r_k)$$

$p_{\mathrm{r}}(r_k)$ is the normalized histogram.

Measuring Image Information

For the fig.8.1(a),

$$\vdots$$

$$H =$$

$$-[0.25\log_2 0.25 + 0.47\log_2 0.47 + 0.25\log_2 0.25 + 0.03\log_2 0.03]$$

$$\approx 1.6614 \text{ bits/pixel}$$

Fidelity Criteria

Let $f(x, y)$ be an input image and $\hat{f}(x, y)$ be an approximation of $f(x, y)$. The images are of size $M \times N$.

The *root - mean - square error* is

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x, y) - f(x, y) \right]^2 \right]^{1/2}$$

Fidelity Criteria

The *mean - square signal - to - noise ratio* of the output image, denoted SNR_{ms}

$$\text{SNR}_{\text{ms}} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)\right]^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}$$

RMSE = 5.17

RMSE = 15.67

RMSE = 14.17

a b c

FIGURE 8.4 Three approximations of the image in Fig. 8.1(a).

Image Compression Models

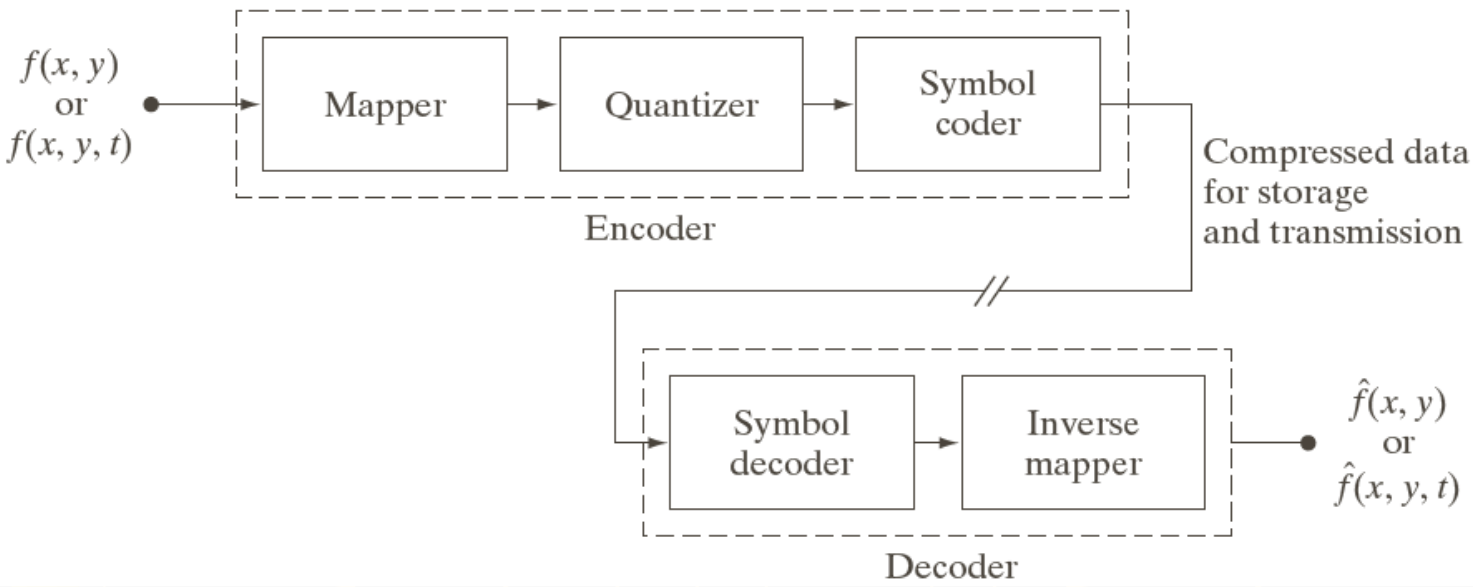


FIGURE 8.5 Functional block diagram of a general image compression system.

Image Compression Standards

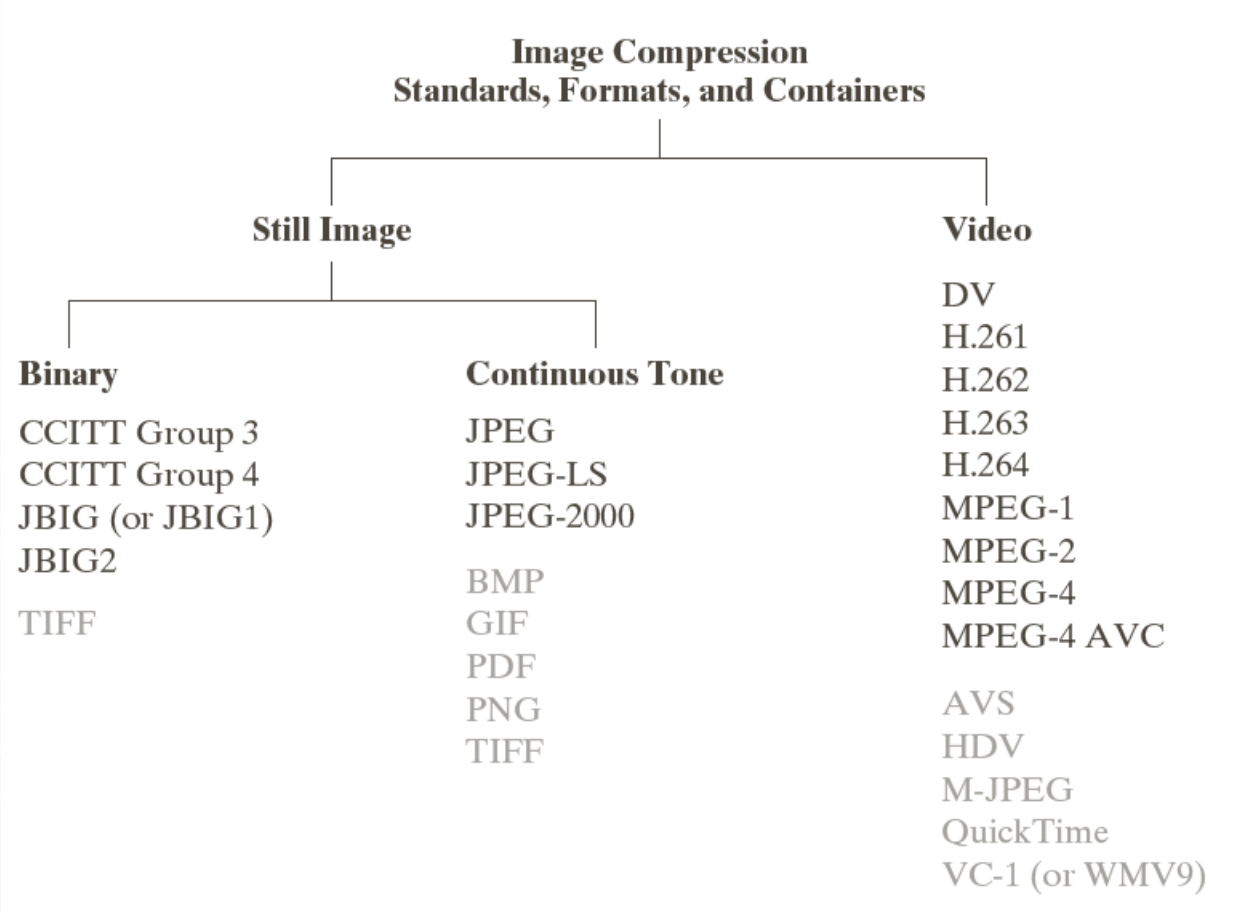


FIGURE 8.6 Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in black; all others are grayed.

Name	Organization	Description
Bi-Level Still Images		
CCITT Group 3	ITU-T	Designed as a facsimile (FAX) method for transmitting binary documents over telephone lines. Supports 1-D and 2-D run-length [8.2.5] and Huffman [8.2.1] coding.
CCITT Group 4	ITU-T	A simplified and streamlined version of the CCITT Group 3 standard supporting 2-D run-length coding only.
JBIG *or* JBIG1	ISO/IEC/ ITU-T	A *Joint Bi-level Image Experts Group* standard for progressive, lossless compression of bi-level images. Continuous-tone images of up to 6 bits/pixel can be coded on a bit-plane basis [8.2.7]. Context sensitive arithmetic coding [8.2.3] is used and an initial low resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary based methods [8.2.6] for text and halftone regions, and Huffman [8.2.1] or arithmetic coding [8.2.3] for other image content. It can be lossy or lossless.
Continuous-Tone Still Images		
JPEG	ISO/IEC/ ITU-T	A *Joint Photographic Experts Group* standard for images of photographic quality. Its lossy *baseline coding system* (most commonly implemented) uses quantized discrete cosine transforms (DCT) on 8×8 image blocks [8.2.8], Huffman [8.2.1], and run-length [8.2.5] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ ITU-T	A lossless to near-lossless standard for continuous tone images based on adaptive prediction [8.2.9], context modeling [8.2.3], and Golomb coding [8.2.2].
JPEG-2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.2.3] and quantized discrete wavelet transforms (DWT) [8.2.10] are used. The compression can be lossy or lossless.

TABLE 8.3 Internationally sanctioned image compression standards. The numbers in brackets refer to sections in this chapter.

(*Continues*)

TABLE 8.3 (*Continued*)

Name	Organization	Description
Video		
DV	IEC	*Digital Video*. A video standard tailored to home and semiprofessional video production applications and equipment—like electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.2.8] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (*integrated services digital network*) lines. It supports non-interlaced 352 × 288 and 176 × 144 resolution images, called CIF (*Common Intermediate Format*) and QCIF (*Quarter CIF*), respectively. A DCT-based compression approach [8.2.8] similar to JPEG is used, with frame-to-frame prediction differencing [8.2.9] to reduce temporal redundancy. A block-based technique is used to compensate for motion between frames.
H.262	ITU-T	See MPEG-2 below.
H.263	ITU-T	An enhanced version of H.261 designed for ordinary telephone modems (i.e., 28.8 Kb/s) with additional resolutions: SQCIF (*Sub-Quarter* CIF 128 × 96), 4CIF (704 × 576), and 16CIF (1408 × 512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, Internet streaming, and television broadcasting. It supports prediction differences within frames [8.2.9], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.2.3].
MPEG-1	ISO/IEC	A *Motion Pictures Expert Group* standard for CD-ROM applications with non-interlaced video at up to 1.5 Mb/s. It is similar to H.261 but frame predictions can be based on the previous frame, next frame, or an interpolation of both. It is supported by almost all computers and DVD players.
MPEG-2	ISO/IEC	An extension of MPEG-1 designed for DVDs with transfer rates to 15 Mb/s. Supports interlaced video and HDTV. It is the most successful video standard to date.
MPEG-4	ISO/IEC	An extension of MPEG-2 that supports variable block sizes and prediction differencing [8.2.9] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 *Advanced Video Coding* (AVC). Identical to H.264 above.

Name	Organization	Description
Continuous-Tone Still Images		
BMP	Microsoft	*Windows Bitmap*. A file format used mainly for simple uncompressed images.
GIF	CompuServe	*Graphic Interchange Format*. A file format that uses lossless LZW coding [8.2.4] for 1- through 8-bit images. It is frequently used to make small animations and short low resolution films for the World Wide Web.
PDF	Adobe Systems	*Portable Document Format*. A format for representing 2-D documents in a device and resolution independent way. It can function as a container for JPEG, JPEG 2000, CCITT, and other compressed images. Some PDF versions have become ISO standards.
PNG	*World Wide Web Consortium* (W3C)	*Portable Network Graphics*. A file format that losslessly compresses full color images with transparency (up to 48 bits/pixel) by coding the difference between each pixel's value and a predicted value based on past pixels [8.2.9].
TIFF	Aldus	*Tagged Image File Format*. A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
Video		
AVS	MII	*Audio-Video Standard*. Similar to H.264 but uses exponential Golomb coding [8.2.2]. Developed in China.
HDV	Company consortium	*High Definition Video*. An extension of DV for HD television that uses MPEG-2 like compression, including temporal redundancy removal by prediction differencing [8.2.9].
M-JPEG	Various companies	*Motion JPEG*. A compression format in which each frame is compressed independently using JPEG.
Quick-Time	Apple Computer	A media container supporting DV, H.261, H.262, H.264, MPEG-1, MPEG-2, MPEG-4, and other video compression formats.
VC-1 WMV9	SMPTE Microsoft	The most used video format on the Internet. Adopted for HD and *Blu-ray* high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.2.8 and 8.2.9] and context dependent variable-length code tables [8.2.1]—but no predictions within frames.

TABLE 8.4 Popular image compression standards, file formats, and containers, not included in Table 8.3.

Some Basic Compression Methods: Huffman Coding

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

FIGURE 8.7 Huffman source reductions.

Some Basic Compression Methods: Huffman Coding

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

FIGURE 8.8 Huffman code assignment procedure.

The average length of this code is

$$L_{avg} = 0.4*1 + 0.3*2 + 0.1*3 + 0.1*4 + 0.06*5 + 0.04*5$$

$$= 2.2 \text{ bits/pixel}$$

LZW (Dictionary coding)

LZW (Lempel-Ziv-Welch) coding, assigns fixed-length code words to variable length sequences of source symbols, but requires no *a priori* knowledge of the probability of the source symbols.

LZW is used in:

- *Tagged Image file format (TIFF)*
- *Graphic interchange format (GIF)*
- *Portable document format (PDF)*

LZW was formulated in 1984

The Algorithm:

- **A codebook or "dictionary" containing the source symbols is constructed.**
- **For 8-bit monochrome images, the first 256 words of the dictionary are assigned to the gray levels 0-255**
- **Remaining part of the dictionary is filled with sequences of the gray levels**

Important features of LZW

1. The dictionary is created while the data are being encoded. So encoding can be done "on the fly"
2. The dictionary is not required to be transmitted. The dictionary will be built up in the decoding
3. If the dictionary "overflows" then we have to reinitialize the dictionary and add a bit to each one of the code words.
4. Choosing a large dictionary size avoids overflow, but spoils compressions

Some Basic Compression Methods: Run-Length Coding

1. Run-length Encoding, or **RLE** is a technique used to **reduce the size of a repeating string of characters**.

2. This repeating string is called a *run*, typically RLE encodes a run of symbols into two bytes , a **count** and a **symbol**.

3. RLE can compress any type of data

4. RLE cannot achieve high compression ratios compared to other compression methods

Some Basic Compression Methods: Run-Length Coding

5. It is easy to implement and is quick to execute.

6. Run-length encoding is supported by most bitmap file formats such as TIFF, BMP and PCX

Some Basic Compression Methods: Run-Length Coding

WWWWWWWWWWWWBWWWWWWWWWWWWBBBW
WWWWWWWWWWWWWWWWWWWWWWWWBWWWW
WWWWWWWWWW

RLE coding:

12W1B12W3B24W1B14W

Some Basic Compression Methods: Symbol-Based Coding

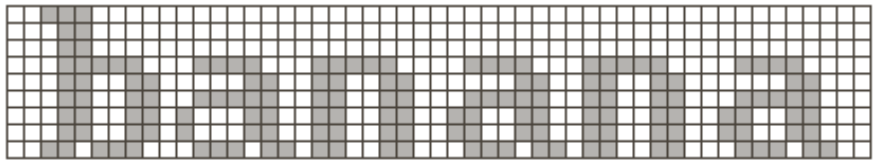
In symbol- or token-based coding, an image is represented as a collection of frequently occurring sub-images, called symbols.

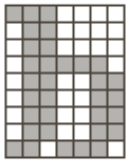
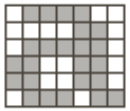
Each symbol is stored in a symbol dictionary

Image is coded as a set of triplets

{(x1,y1,t1), (x2, y2, t2), …}

Some Basic Compression Methods: Symbol-Based Coding



Token	Symbol
0	
1	
2	

Triplet
(0, 2, 0) (3, 10, 1) (3, 18, 2) (3, 26, 1) (3, 34, 2) (3, 42, 1)

a b c

FIGURE 8.17
(a) A bi-level document, (b) symbol dictionary, and (c) the triplets used to locate the symbols in the document.

Some Basic Compression Methods: Bit-Plane Coding

An m-bit gray scale image can be converted into m binary images by bit-plane slicing. These individual images are then encoded using run-length coding.

Code the bitplanes separately, using RLE (flatten each plane row-wise into a 1D array), Golomb coding, or any other lossless compression technique.

- Let I be an image where every pixel value is n-bit long
- Express every pixel in binary using n bits
- Form n binary matrices (called bitplanes), where the i-th matrix consists of the i-th bits of the pixels of I.

Some Basic Compression Methods: Bit-Plane Coding

Example: Let I be the following 2x2 image where the pixels are 3 bits long

101 110
111 011

The corresponding 3 bitplanes are:

1	1	0	1	1	0
1	0	1	1	1	1

However, a small difference in the gray level of adjacent pixels can cause a disruption of the run of zeroes or ones.

Eg: Let us say one pixel has a gray level of 127 and the next pixel has a gray level of 128.

In binary: 127 = 01111111

& 128 = 10000000

Therefore a small change in gray level has decreased the run-lengths in all the bit-planes!

GRAY CODE

1. Gray coded images are free of this problem which affects images which are in binary format.

2. In gray code the representation of adjacent gray levels will differ only in one bit (unlike binary format where all the bits can change.

Let $g_{m-1}.....g_1g_0$ represent the gray code representation of a binary number.

Then:

$$g_i = a_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$

$$g_{m-1} = a_{m-1}$$

In gray code:

127 = 01000000

128 = 11000000

Gray Coding

To convert a binary number $b_1b_2b_3..b_{n-1}b_n$ to its corresponding binary reflected Gray code.

Start at the right with the digit b_n. If the b_{n-1} is 1, replace b_n by $1-b_n$; otherwise, leave it unchanged. Then proceed to b_{n-1} .

Continue up to the first digit b_1, which is kept the same since it is assumed to be a $b_0 = 0$.

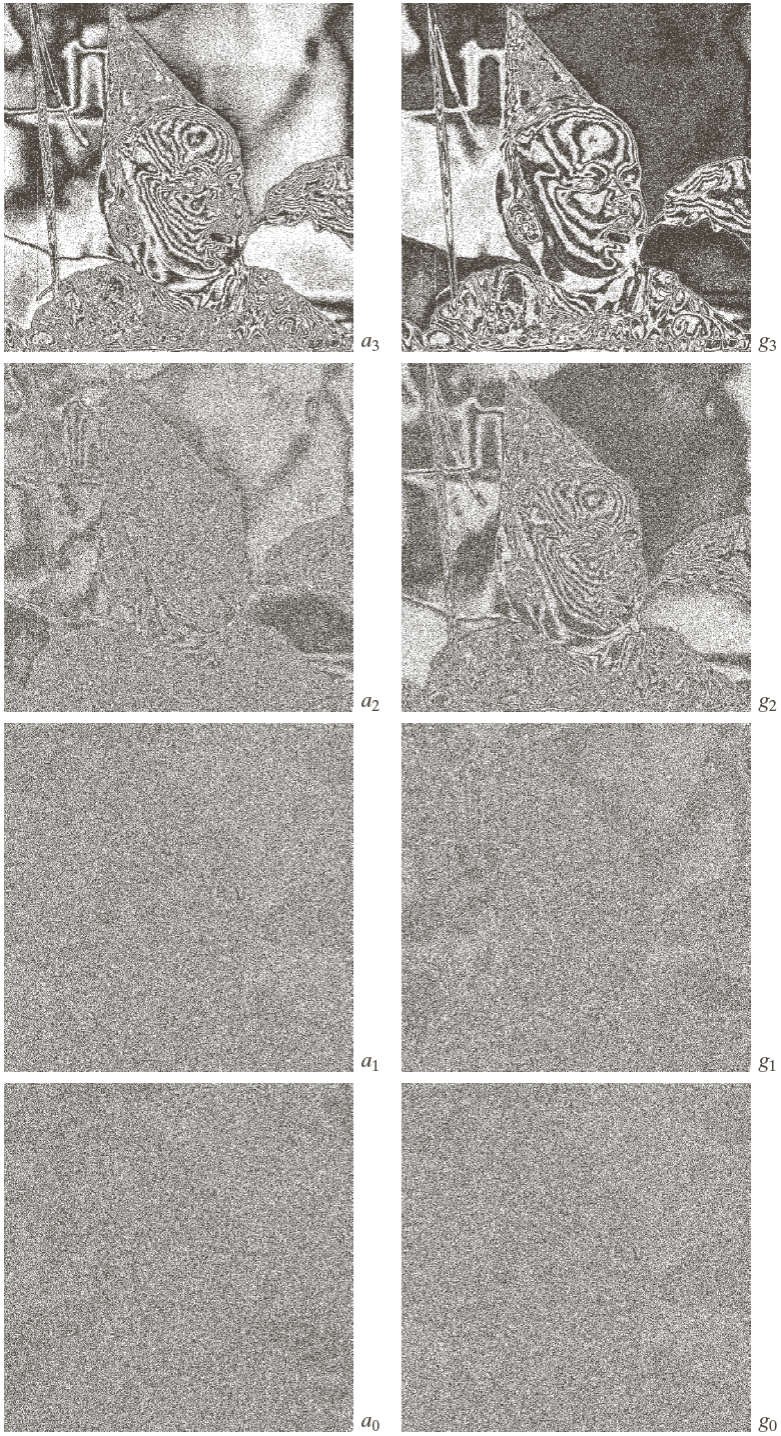
The resulting number is the reflected binary Gray code.

Examples: Gray Coding

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

a b
c d
e f
g h

FIGURE 8.19
(a) A 256-bit monochrome image. (b)–(h) The four most significant binary and Gray-coded bit planes of the image in (a).



a b
c d
e f
g h

FIGURE 8.20
(a)–(h) The four least significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.19(a).

Decoding a gray coded image

The MSB is retained as such,i.e.,

$$a_i = g_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$

$$a_{m-1} = g_{m-1}$$

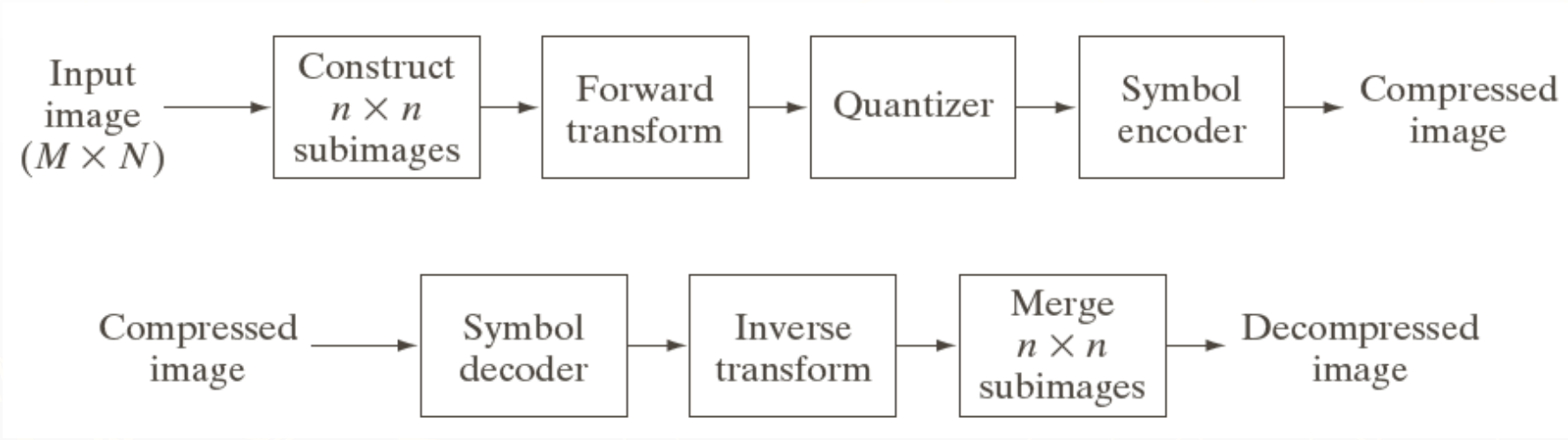
Differential Pulse Code Modulation (DPCM)

- Example:

  AAABBCDDDD encoded as A0001123333

- Change reference symbol if delta becomes too large
- Works better than RLE for many digital images

Block Transform Coding



a
b

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

Block Transform Coding

Consider a subimage of size $n \times n$ whose forward, discrete transform $T(u,v)$ can be expressed in terms of the relation

$$T(u,v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x,y) r(x,y,u,v)$$

for $u, v = 0, 1, 2, \ldots, n\text{-}1$.

Block Transform Coding

Given $T(u,v),\ g(x,y)$ similarly can be obtained using the generalized inverse discrete transform

$$g(x,y) = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)s(x,y,u,v)$$

for $x, y = 0,1,2,...,n\text{-}1.$

Image transform

- Two main types:

-orthogonal transform:

e.g. Walsh-Hdamard transform, DCT

-subband transform:

e.g. Wavelet transform

Orthogonal transform

- Orthogonal matrix **W**

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}$$

➔ C=W. D

- Reducing redundancy
- Isolating frequencies

Block Transform Coding
Walsh-Hadamard transform (WHT)

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

$$H_1 = [1] \qquad H_2 = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Block Transform Coding
Discrete Cosine Transform (DCT)

$$r(x,y,u,v) = s(x,y,u,v)$$

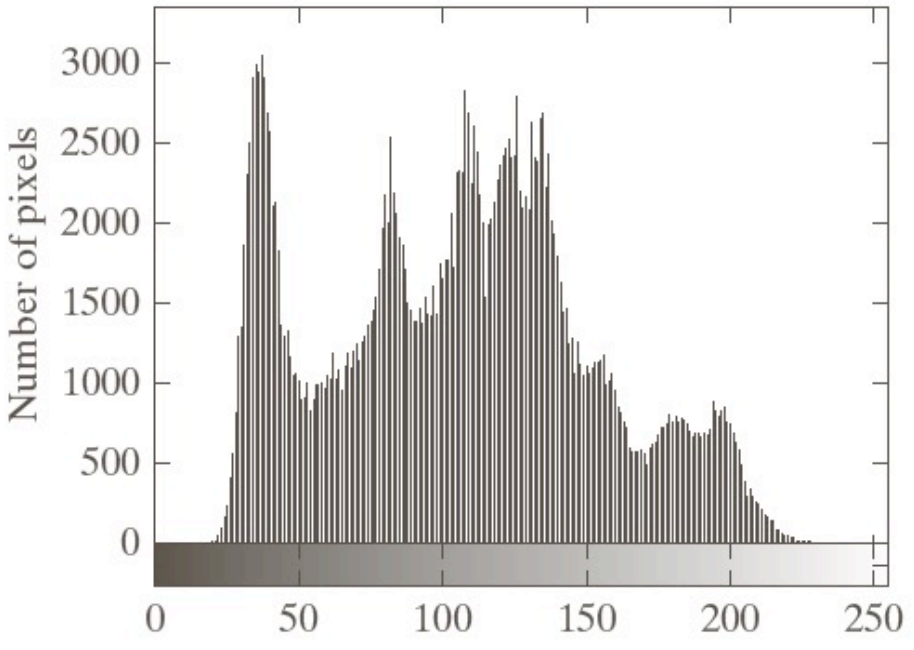
$$= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

$$\text{where } \alpha(u/v) = \begin{cases} \sqrt{\frac{1}{n}} & \text{for } u/v = 0 \\ \sqrt{\frac{2}{n}} & \text{for } u/v = 1,2,\ldots,n-1 \end{cases}$$

Example

a b

FIGURE 8.9 (a) A 512 × 512 8-bit image, and (b) its histogram.



In each case, 50% of the resulting coefficients were truncated and taking the inverse transform of the truncated coefficients arrays.

a b c
d e f

RMSE = 2.32 RMSE = 1.78 RMSE = 1.13

FIGURE 8.24 Approximations of Fig. 8.9(a) using the (a) Fourier, (b) Walsh-Hadamard, and (c) cosine transforms, together with the corresponding scaled error images in (d)–(f).

Subimage Size Selection

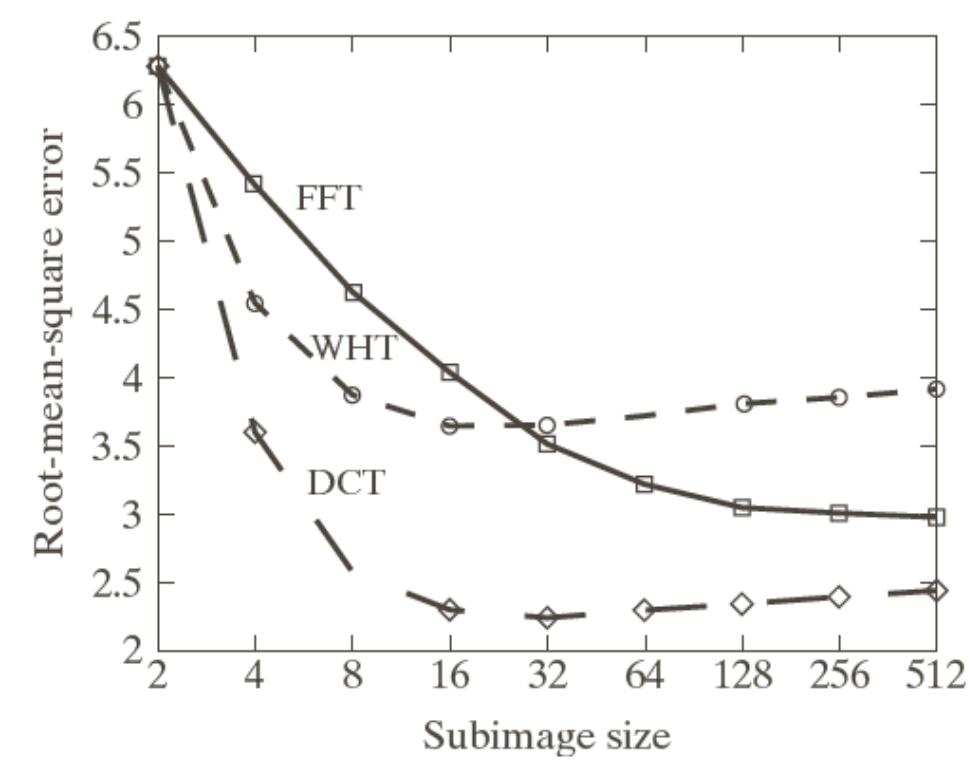
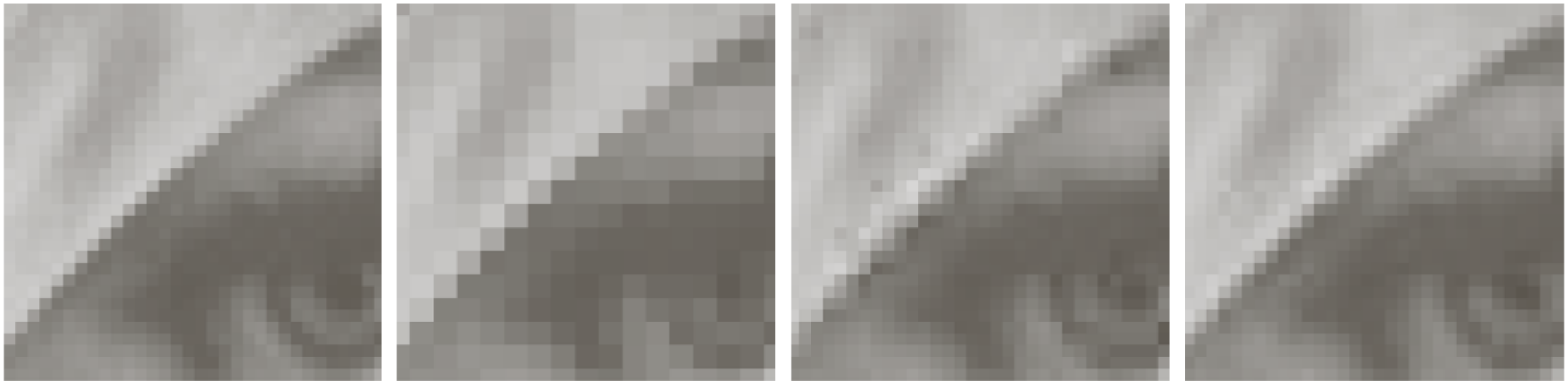


FIGURE 8.26 Reconstruction error versus subimage size.

Subimage Size Selection



a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2×2 subimages, (c) 4×4 subimages, and (d) 8×8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

Bit Allocation

The overall process of truncating, quantizing, and coding the coefficients of a transformed subimage is commonly called **bit allocation**

Zonal coding

The retained coefficients are selected on the basis of maximum variance

Threshold coding

The retained coefficients are selected on the basis of maximum magnitude

RMSE = 4.5

RMSE = 6.5

a b
c d

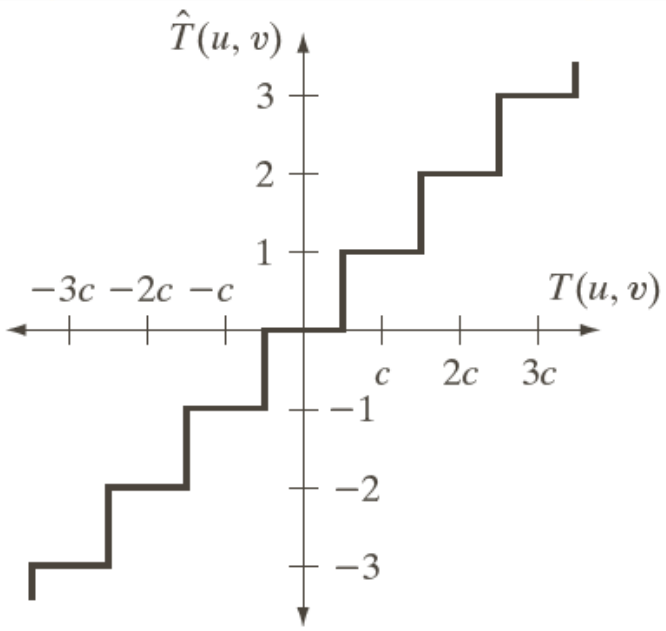
FIGURE 8.28 Approximations of Fig. 8.9(a) using 12.5% of the 8×8 DCT coefficients: (a)—(b) threshold coding results; (c)—(d) zonal coding results. The difference images are scaled by 4.

Threshold Coding

$$\hat{T}(u,v) = round\left[\frac{T(u,v)}{Z(u,v)}\right]$$

$$Z = \begin{bmatrix} Z(0,0) & Z(0,1) & \ldots & Z(0,n-1) \\ Z(1,0) & \ldots & \ldots & \ldots \\ \ldots. & \ldots & \ldots & \ldots \\ Z(n-1,0) & \ldots & \ldots & Z(n-1,n-1) \end{bmatrix}$$

Threshold Coding



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

a b

FIGURE 8.30 (a) A threshold coding quantization curve [see Eq. (8.2-29)]. (b) A typical normalization matrix.

Threshold Coding

FIGURE 8.31 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.30(b): (a) **Z**, (b) 2**Z**, (c) 4**Z**, (d) 8**Z**, (e) 16**Z**, and (f) 32**Z**.

Fact about JPEG Compression

- JPEG stands for Joint Photographic Experts Group
- Used on 24-bit color files.
- Works well on photographic images.
- Although it is a lossy compression technique, it yields an excellent quality image with high compression rates.

Fact about JPEG Compression

- It defines three different coding systems:

  1. a lossy baseline coding system, adequate for most compression applications

  2. an extended coding system for greater compression, higher precision, or progressive reconstruction applications

  3. A lossless independent coding system for reversible compression

Steps in JPEG Compression

1. (Optionally) If the color is represented in RGB mode, translate it to YUV.

2. Divide the file into 8 X 8 blocks.

3. Transform the pixel information from the spatial domain to the frequency domain with the Discrete Cosine Transform.

4. Quantize the resulting values by dividing each coefficient by an integer value and rounding off to the nearest integer.

5. Look at the resulting coefficients in a zigzag order. Do a run-length encoding of the coefficients ordered in this manner. Follow by Huffman coding.

Step 1a: Converting RGB to YUV

- YUV color mode stores color in terms of its luminance (brightness) and chrominance (hue).
- The human eye is less sensitive to chrominance than luminance.
- YUV is not required for JPEG compression, but it gives a better compression rate.

RGB vs. YUV

- It's simple arithmetic to convert RGB to YUV. The formula is based on the relative contributions that red, green, and blue make to the luminance and chrominance factors.

- There are several different formulas in use depending on the target monitor.

For example:

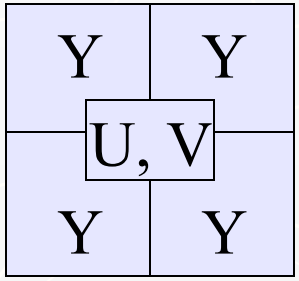
$Y = 0.299 * R + 0.587 * G + 0.114 * B$

$U = -0.1687 * R - 0.3313 * G + 0.5 * B + 128$

$V = 0.5 * R - 0.4187 * G - 0.813 * B + 128$

Step 1b: Downsampling

- The chrominance information can (optionally) be downsampled.
- The notation 4:1:1 means that for each block of four pixels, you have 4 samples of luminance information (Y), and 1 each of the two chrominance components (U and V).



Step 2: Divide into 8 X 8 blocks

- Note that with YUV color, you have 16 pixels of information in each block for the Y component (though only 8 in each direction for the U and V components).
- If the file doesn't divide evenly into 8 X 8 blocks, extra pixels are added to the end and discarded after the compression.
- The values are shifted "left" by subtracting 128.

Discrete Cosine Transform

- The DCT transforms the data from the spatial domain to the frequency domain.
- The spatial domain shows the amplitude of the color as you move through space
- The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.

Step 3: DCT

- The frequency domain is a better representation for the data because it makes it possible for you to separate out – and throw away – information that isn't very important to human perception.
- The human eye is not very sensitive to high frequency changes – especially in photographic images, so the high frequency data can, to some extent, be discarded.

Step 3: DCT

- The color amplitude information can be thought of as a wave (in two dimensions).
- You're decomposing the wave into its component frequencies.
- For the 8 X 8 matrix of color data, you're getting an 8 X 8 matrix of coefficients for the frequency components.

Step 4: Quantize the Coefficients Computed by the DCT

- The DCT is lossless in that the reverse DCT will give you back exactly your initial information (ignoring the rounding error that results from using floating point numbers.)
- The values from the DCT are initially floating-point.
- They are changed to integers by quantization.

Step 4: Quantization

- Quantization involves dividing each coefficient by an integer between 1 and 255 and rounding off.
- The quantization table is chosen to reduce the precision of each coefficient to no more than necessary.
- The quantization table is carried along with the compressed file.

Step 5: Arrange in "zigzag" order

- This is done so that the coefficients are in order of increasing frequency.
- The higher frequency coefficients are more likely to be 0 after quantization.
- This improves the compression of run-length encoding.
- Do run-length encoding and Huffman coding.

EXAMPLE 8.17:
JPEG baseline coding and decoding.

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	0	0	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

EXAMPLE 8.17:
JPEG baseline coding and decoding.

[−26 −3 1 −3 −2 −6 2 −4 1 −4 1 1 5 0 2 0 0 −1 2 0 0 0 0 0 −1 −1 EOB]

Assuming the DC coefficient of the transformed and quantized subimage to its immediate left was -17.

The resulting of DPCM difference is [-26 – (-17)] = -9.

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	0	0	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

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−416	−33	−60	32	48	0	0	0
12	−24	−56	0	0	0	0	0
−42	13	80	−24	−40	0	0	0
−56	17	44	−29	0	0	0	0
18	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

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−70	−64	−61	−64	−69	−66	−58	−50
−72	−73	−61	−39	−30	−40	−54	−59
−68	−78	−58	−9	13	−12	−48	−64
−59	−77	−57	0	22	−13	−51	−60
−54	−75	−64	−23	−13	−44	−63	−56
−52	−71	−72	−54	−54	−71	−71	−54
−45	−59	−70	−68	−67	−67	−61	−50
−35	−47	−61	−66	−60	−48	−44	−44

EXAMPLE 8.17:
JPEG baseline coding and decoding.

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	6	4	3	−2
3	8	4	−4	2	6	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

a b c
d e f

FIGURE 8.32 Two JPEG approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image.

JPEG at 0.125 bpp (enlarged)

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)

JPEG2000 at 0.125 bpp

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)