Introduction to Image Processing

Prof. Alexandre Zaghetto
http://alexandre.zaghetto.com
zaghetto@unb.br

University of Brasília
Department of Computer Science
LISA: Laboratory of Imagens, Signals and Acoustics

Topic 08
Image Coding

1. Fundamentals

• To better understand the need for compact image representations, consider the amount of data required to represent a **two-hour** standard definition (**SD**) television movie using 720 x 480 x 24 bit pixel arrays,

$$30\frac{\text{frames}}{\text{sec}} \times (720 \times 480)\frac{\text{pixels}}{\text{frame}} \times 3\frac{\text{bytes}}{\text{pixel}} = 31{,}104{,}000\ \text{bytes/sec}$$

$$31{,}104{,}000\frac{\text{bytes}}{\text{sec}} \times \left(60^2\right)\frac{\text{sec}}{\text{hr}} \times 2\ \text{hrs} \cong 2.24 \times 10^{11}\ \text{bytes}$$

or 224 GB (gigabytes) of data. Twenty-seven 8.5 GB dual-layer DVDs are needed to store it.

1. Fundamentals

• To put a two-hour movie on a single DVD, each frame must be compressed—on average—by a factor of 26.3.

• The compression must be even higher for high definition (**HD**) television, where image resolutions reach 1920x1080x24 bits/image.

• Web page images, high-resolution digital camera photos, streamed video also are compressed routinely to save storage space and reduce transmission time.

• The term **data compression** refers to the process of reducing the amount of data required to represent a given quantity of information.

• In this definition, **data** and **information** are not the same thing.

1. Fundamentals

• Data are the means by which information is conveyed. Various amounts of data can be used to represent the same amount of information.

• Representations that contain irrelevant or repeated information are said to contain **redundant data**.

• If we let b and b' denote the number of data bits in two representations of the same information, the **relative data redundancy** of the representation with bits is

$$R = 1 - \frac{1}{C}$$

where C, commonly called the **compression ratio**, is defined as

$$C = \frac{b}{b'}$$

1. Fundamentals

• Two-dimensional intensity arrays suffer from three principal types of data redundancies that can be identified and exploited:

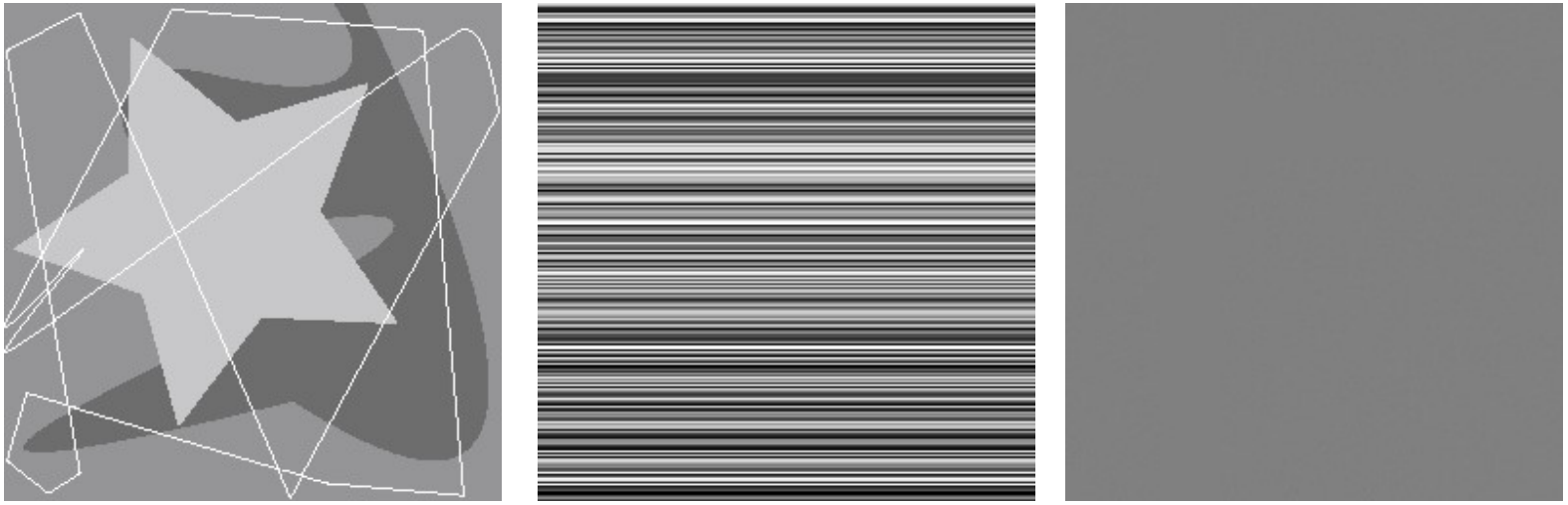
1. **Coding redundancy**: more bits than necessary are used to represent the intensities in 2-D intensity arrays.

2. **Interpixel redundancy**: pixels of most 2-D intensity arrays are correlated spatially (i.e., each pixel is similar to or dependent on neighboring pixels); temporally correlated pixels (i.e., those similar to or dependent on pixels in nearby frames) also replicate information.

3. **Psychovisual redundancy (Irrelevant information):** most 2-D intensity arrays contain information that is ignored by the human visual system.

1. Fundamentals

- Two-dimensional intensity arrays suffer from three principal types of data redundancies that can be identified and exploited.



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.)

1. Fundamentals

- Coding Redundancy

  - Consider the probability mass function defined in "Topic 03 - Intensity Transformation and Spatial Filtering"

$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, \ldots, L-1$$

  - The average number of bits required to represent each pixel is

$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k)\, p_r(r_k)$$

where $l(r_k)$ the number of bits used to represent each value of r_k .

1. Fundamentals

- Coding Redundancy
  - The total number of bits required to represent an image is M x N x L_{avg}.
  - If the intensities are represented using a natural **m-bit fixed-length code** (code 1), then L_{avg} = **m**.
  - The computer-generated image in Fig. 8.1(a) has the intensity distribution shown in the second column of Table 8.1.
  - If a natural 8-bit binary code (denoted as **code 1** in Table 8.1) is used to represent its 4 possible intensities, the average number of bits for **code 1** is 8 bits.

1. Fundamentals

- Coding Redundancy

  - On the other hand, if the scheme designated as **code 2** in Table 8.1 is used

$$L_{avg} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

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

TABLE 8.1 Example of variable-length coding.

1. Fundamentals

- Coding Redundancy

  - The total number of bits needed to represent the entire image is

$$MNL_{\text{avg}} = 256 \times 256 \times 1.81 = 118{,}621$$

  - The resulting **compression rate** and corresponding **relative redundancy** are

$$C = \frac{256 \times 256 \times 8}{118{,}621} = \frac{8}{1.81} \approx 4.42$$

$$R = 1 - \frac{1}{4.42} = 0.774$$

  - Thus **77.4%** of the data in the original 8-bit 2-D intensity array is **redundant**.

1. Fundamentals

- Coding Redundancy

  - The compression achieved by code 2 results from assigning **fewer bits** to the **more probable** intensity values than to the less probable ones.

  - Note that the best fixed-length code that can be used in this example is the natural **2-bit** counting sequence.

  - But the resulting compression is only or 4:1—about 10% less than the 4.42:1 compression of the variable-length code.

  - As the preceding example shows, coding redundancy is present when the codes assigned to a set of events (such as intensity values) do not take full advantage of the probabilities of the events.

1. Fundamentals

- Spatial and Temporal Redundancy

  - Consider the computer-generated collection of constant intensity lines in Fig. 8.1(b).

    1. All 256 intensities are equally probable.

    2. Because the intensity of each line was selected randomly, its pixels are independent of one another in the vertical direction

    3. Because the pixels along each line are identical, they are maximally correlated in the horizontal direction.

1. Fundamentals

- Spatial and Temporal Redundancy

  - Observation 1 tells us that the image in Fig. 8.1(b) cannot be compressed by variable length coding alone.

  - Observations 2 and 3 reveal a significant spatial redundancy that can be eliminated using a sequence of **run-length pairs.**

    - ✓ Each run-length pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity.

    - ✓ For the image in Fig. 8.1(b), the compression rate is

$$C = (256 \times 256 \times 8)/[(256 + 256) \times 8] \text{ or } 128{:}1$$

1. Fundamentals

- Spatial and Temporal Redundancy

  - In most images, pixels are correlated spatially and in time.

  - The **information** carried by a **single pixel** is **small**.

  - Much of its visual contribution is redundant in the sense that it **can be inferred** from its **neighbors**.

  - To **reduce** the **redundancy** associated with spatial and temporal correlation they must be transformed into a more efficient "non-visual" representation.

  - A mapping is said to be **reversible** if the original pixels can be reconstructed without error; otherwise the mapping is said to be **irreversible**.

1. Fundamentals

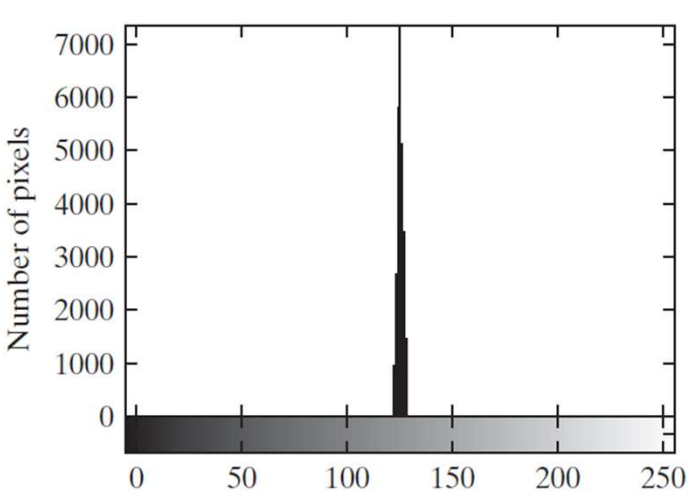
- Psychovisual redundancy (Irrelevant Information)
  - One of the simplest ways to compress a set of data is to remove superfluous data from the set.
  - Information that s **ignored** by the **human visual system** are candidates for **omission**.
  - Thus, the image in Fig. 8.1(c), because it appears to be a homogeneous field of gray, can be represented by its average intensity alone - a single 8-bit value.
  - In this case, the compression rate is

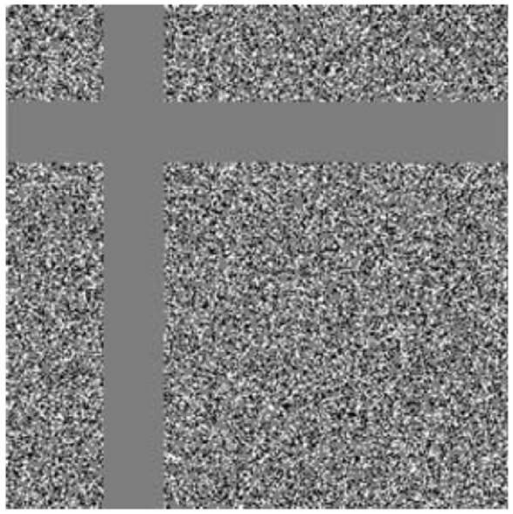
$$C = (256 \times 256 \times 8)/8 \quad \text{or} \quad 65{,}536{:}1$$

1. Fundamentals

- Psychovisual redundancy (Irrelevant Information)

  - However…





a b

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

  - Because its omission results in a **loss** of **quantitative information**, its removal is commonly referred to as **quantization**.

1. Fundamentals

- Measuring Image Information

  - A question that naturally arises is this: How few bits are actually needed to represent the information in an image?

  - **Information theory** provides the mathematical framework to answer this and related questions.

  - 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

  - If *P(E) = 1* (that is, the event always occurs), *I(E) = 0*.

1. Fundamentals

- Measuring Image Information
  - The **base** of the logarithm determines the **unit** used to measure information.
  - If the **base 2** is selected, the unit of information is the **bit**.
  - Note that if $P(E) = ½$, $I(E) = 1$ bit.
  - That is, 1 bit is the amount of information conveyed when one of two possible equally likely events occurs.

1. Fundamentals

- Measuring Image Information

  - Consider o 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, is

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

where a_j are called **source symbols**.

  - Because they are statistically independent, the source itself is called a **zero-memory source**.

1. Fundamentals

- 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.

  - Then the intensity source's entropy becomes

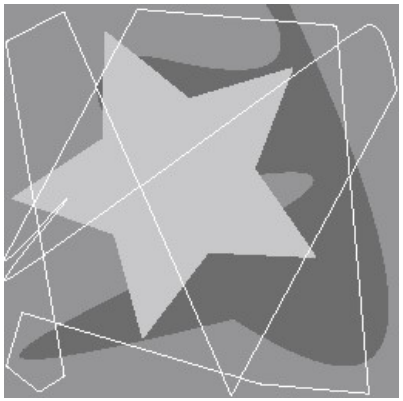
$$\tilde{H} = -\sum_{k=0}^{L-1} p_r(r_k) \log_2 p_r(r_k)$$

  - It is **not possible** to **code** the intensity values of the imaginary source (and thus the sample image) **with fewer than** $\tilde{H}$ bits/pixel.

1. Fundamentals

- Measuring Image Information

  - The entropy of the image in Fig. 8.1(a) can be estimated by substituting the intensity probabilities from Table 8.1 into previous equation:



r_k	$p_r(r_k)$
$r_{87} = 87$	0.25
$r_{128} = 128$	0.47
$r_{186} = 186$	0.25
$r_{255} = 255$	0.03
r_k for $k \neq 87, 128, 186, 255$	0

$$\begin{aligned}\tilde{H} &= -[0.25 \log_2 0.25 + 0.47 \log_2 0.47 + 0.25 \log_2 0.25 + 0.03 \log_2 0.03] \\ &\approx -[0.25(-2) + 0.47(-1.09) + 0.25(-2) + 0.03(-5.06)] \\ &\approx 1.6614 \text{ bits/pixel}\end{aligned}$$

1. Fundamentals

- Measuring Image Information

  - Shannon's first theorem

    - ✓ The variable-length code (**code 2**) in Table 8.1 was able to represent the intensities of the image in Fig. 8.1(a) using only 1.81 bits/pixel.

    - ✓ Although this is higher than the 1.6614 bits/pixel entropy estimate, Shannon's first theorem — also called the **noiseless coding theorem** (Shannon [1948]) — assures us that the image in Fig. 8.1(a) can be represented with as few as 1.6614 bits/pixel.

1. Fundamentals

- Measuring Image Information

  - Fidelity Criteria

    - ✓ The **removal** of **irrelevant information** involves a **loss** of real or **quantitative** image information.

    - ✓ Because information is lost, a means of quantifying the nature of the loss is needed.

    - ✓ Two types of criteria can be used for such an assessment: (1) **objective** fidelity criteria and (2) **subjective** fidelity criteria.

1. Fundamentals

- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ When information loss can be expressed as a mathematical function, it is said to be based on an **objective** fidelity criterion.
      - ○ An example is the **root-mean-square (rms) error** between two images.

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

1. Fundamentals

- Measuring Image Information
  - Fidelity Criteria
    - Another example is the **mean-squared signal-to-noise ratio** (SNR_{ms})**,**

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

1. Fundamentals

- Measuring Image Information
  - Fidelity Criteria
    - But **peak signal-to-noise ratio** (PSNR) is the most commonly objective quality metric used in image processing.

$$\text{PSNR} = 10\log_{10}\frac{MAX^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2}\,\text{dB}$$

1. Fundamentals

- Measuring Image Information

  - Fidelity Criteria

    - ✓ While **objective fidelity** criteria offer a simple and convenient way to evaluate information loss, decompressed images are ultimately viewed by humans.

    - ✓ So, measuring image quality by the **subjective evaluations** of people is often more appropriate.

    - ✓ The evaluations may be made using an absolute rating scale or by means of side-by-side comparisons of $\hat{f}\ (x, y)$ and $f\ (x, y)$.

1. Fundamentals

- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Absolute rating scale

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

TABLE 8.2 Rating scale of the Television Allocations Study Organization. (Frendendall and Behrend.)

1. Fundamentals

- Measuring Image Information

  - Fidelity Criteria

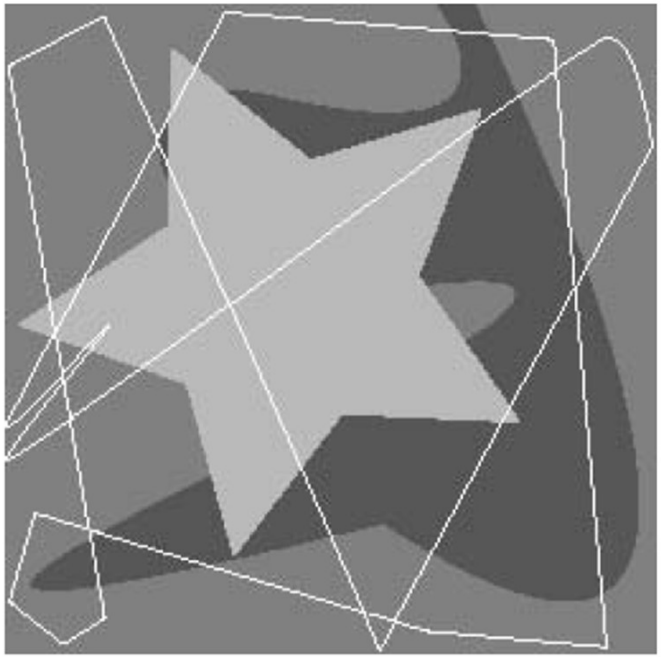
    - ✓ Side-by-side comparisons of $\hat{f}(x,y)$ and $f(x,y)$.

      - Can be done with a scale such as {-3,-2,-1,0, 1,2,3} to represent the subjective evaluations:

        - much worse
        - worse
        - slightly worse
        - the same
        - slightly better
        - better
        - much better

1. Fundamentals

- Measuring Image Information
  - Fidelity Criteria
    - ✓ Example 1 (absolute rating scale)



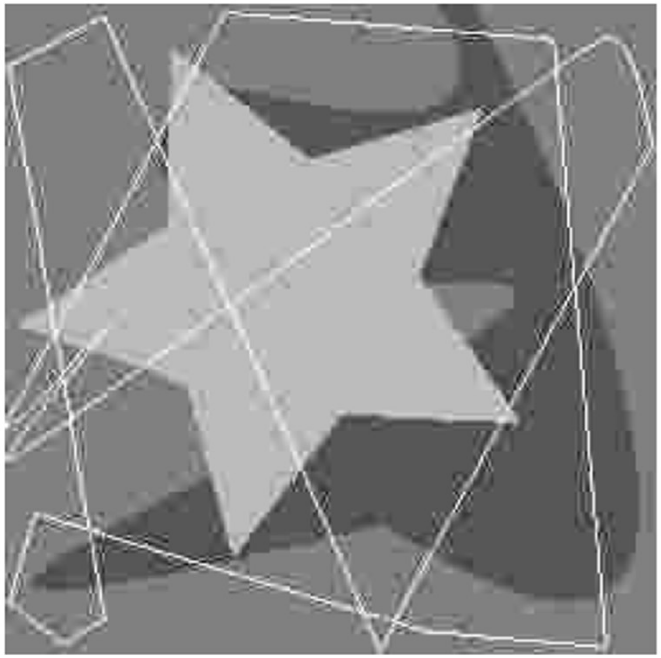
Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

1. Fundamentals

- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 1 (absolute rating scale)

1. Fundamentals

- Measuring Image Information
  - Fidelity Criteria
    - ✓ Example 1 (absolute rating scale)



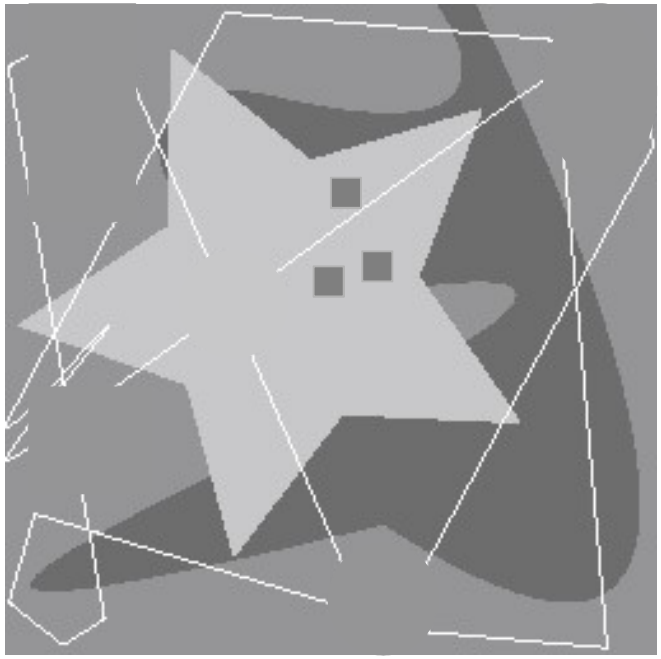
Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

1. Fundamentals

- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 1 (absolute rating scale)

1. Fundamentals

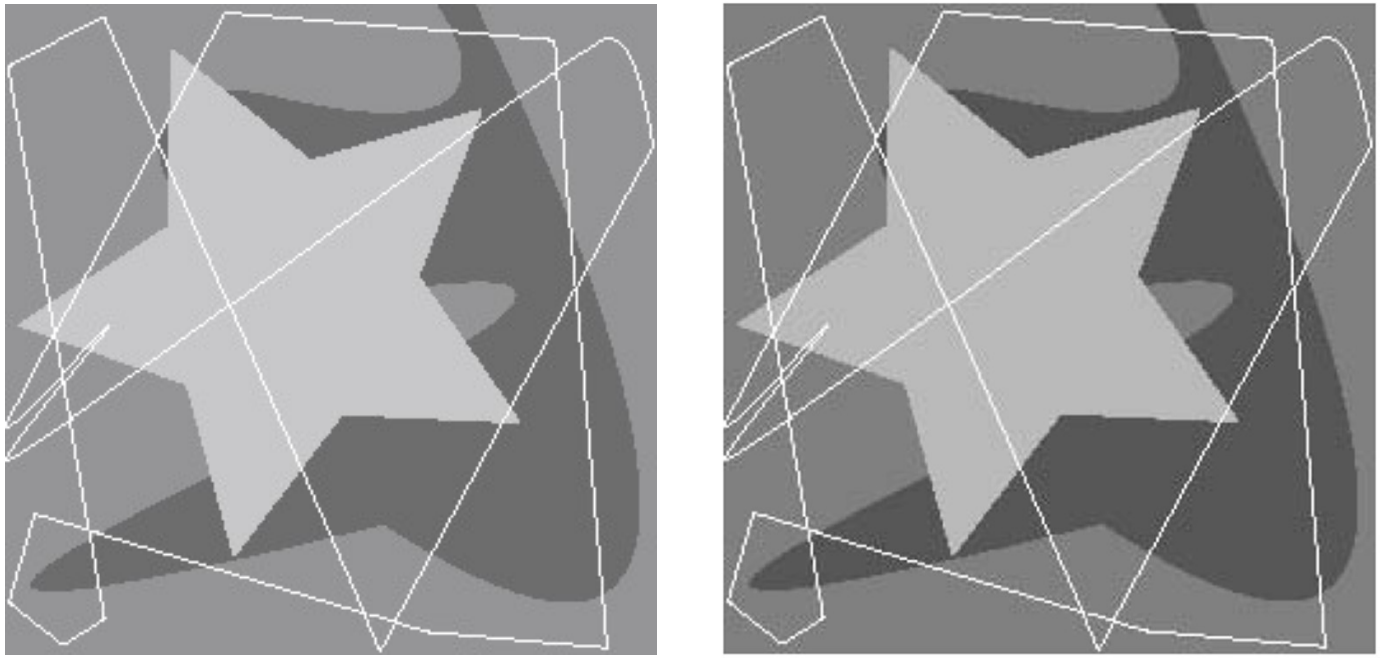
- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 1 (absolute rating scale)



Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

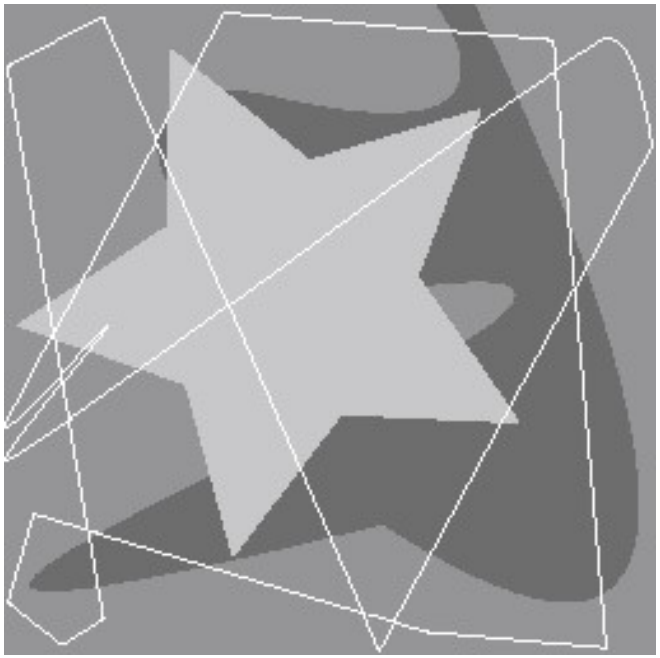
1. Fundamentals

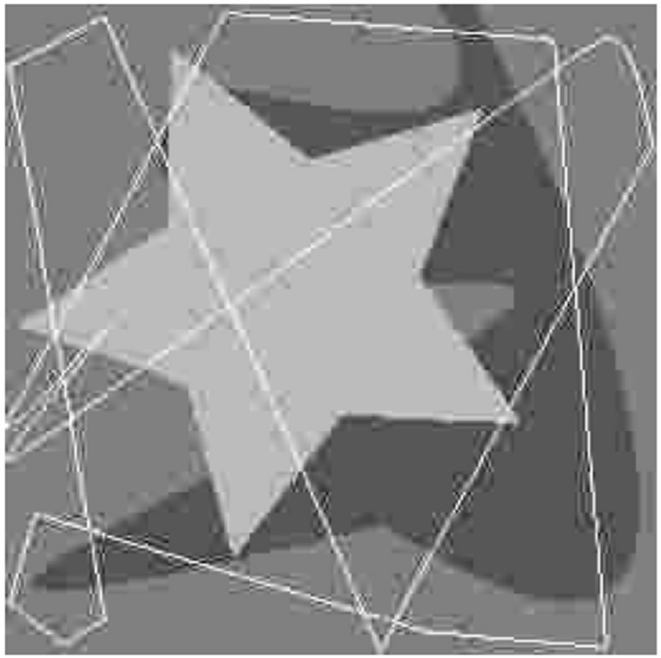
- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 2 (side-by-side)



1. Fundamentals

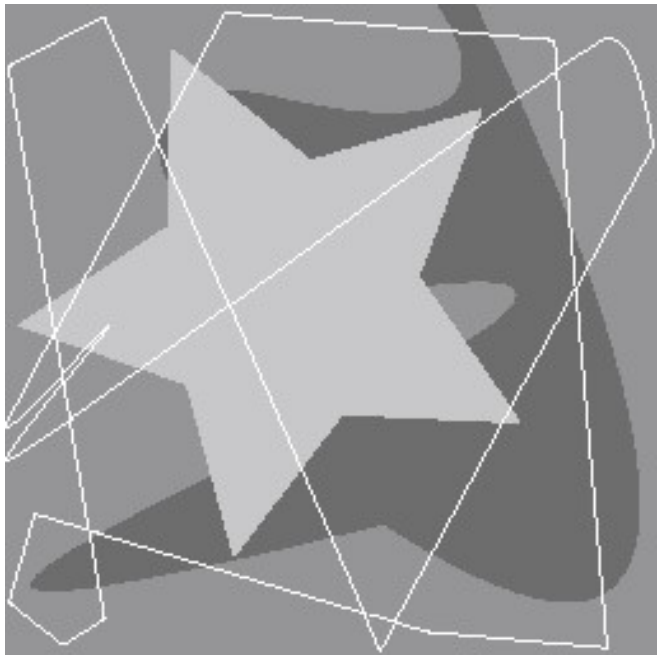
- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 2 (side-by-side)

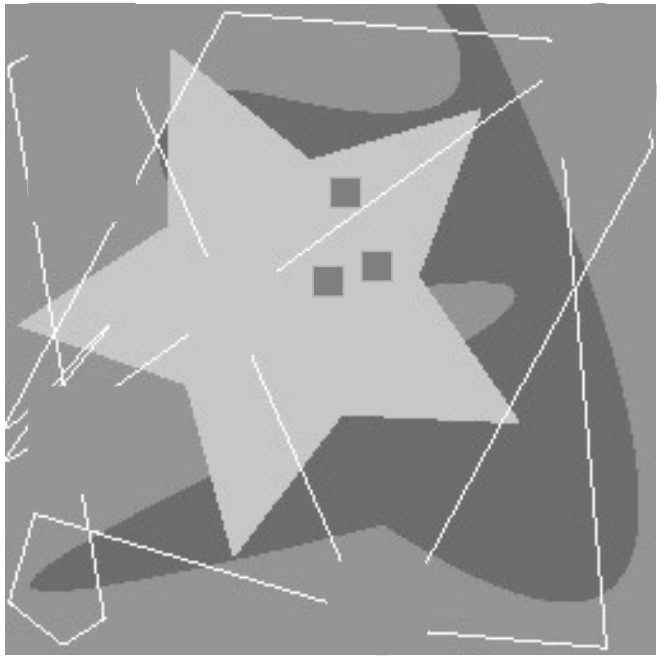




1. Fundamentals

- Measuring Image Information
  - ➢ Fidelity Criteria
    - ✓ Example 2 (side-by-side)





1. Fundamentals

- Image Compression Model

  - An image compression system is composed of two distinct functional components: an **encoder** and a **decoder**.

  - A **codec** is a device or program that is capable of both encoding and decoding.

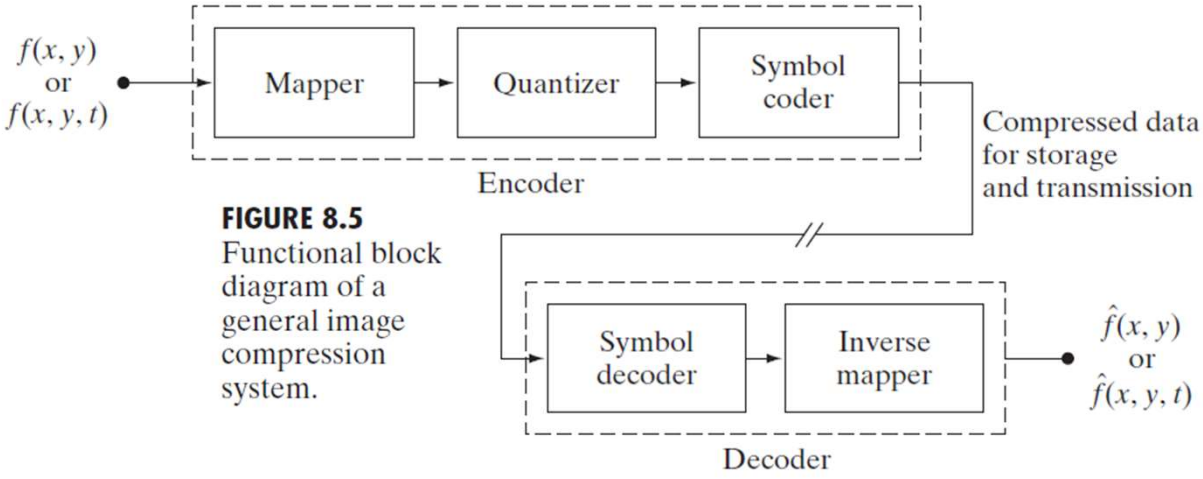


FIGURE 8.5
Functional block diagram of a general image compression system.

1. Fundamentals

- Image Compression Model

  - **Mapper**: transforms *f* into a format designed to reduce spatial and temporal redundancy.

  - **Quantizer**: reduces the accuracy of the mapper's output in accordance with a pre-established fidelity criterion.

    - ✓ The goal is to keep irrelevant information out of the compressed representation.

  - **Symbol coder**: generates a fixed- or variable-length code to represent the quantizer output.

  - Upon its completion, the input image has been processed for the removal of each of the redundancies described before.

1. Fundamentals

- Image Compression Model

  - **Symbol decoder** and **inverse mapper:** they perform, in reverse order, the inverse operations of the encoder's symbol **encoder** and **mapper**.

  - Because **quantization** results in **irreversible** information **loss**, an inverse quantizer block is **not included** in the general **decoder** model.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - **Image file format** is a standard way to organize and store image data.

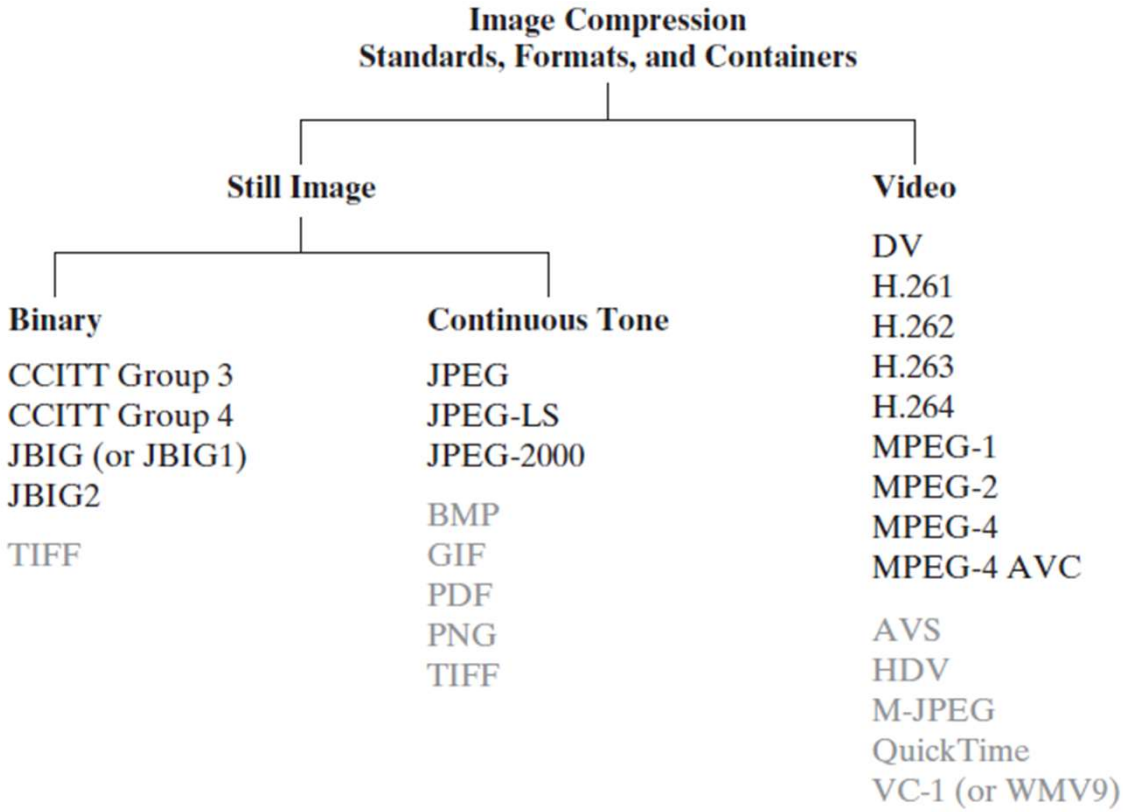
    - ✓ It defines how the data is arranged and the type of compression—if any—that is used

  - **Image container** is similar to a file format but handles multiple types of image data.

  - **Compression standards** define procedures for compressing and decompressing images.

1. Fundamentals

- Image Formats, Containers, and Compression Standards



1. Fundamentals

- Image Formats, Containers, and Compression Standards
  - The entries in black are international standards sanctioned by the
    - **International Standards Organization** (ISO);
    - **International Electrotechnical Commission** (IEC); and/or
    - **International Telecommunications Union** (ITU-T).

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - Bi-level Still Images

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.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - 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.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - Video

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.
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-4 AVC	ISO/IEC	MPEG-4 Part 10 *Advanced Video Coding* (AVC). Identical to H.264 above.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - 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.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

  - Continuous-tone Still Images

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.

2. Some Basic Compression Methods

- Huffman coding

CCITT
JBIG2
JPEG
MPEG-1,2,4
H.261, H.262,
H.263, H.264

- One of the most popular techniques for removing coding redundancy is due to Huffman.
- The **first step** in Huffman's approach is to create a series of source reductions.

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				

2. Some Basic Compression Methods

- Huffman coding

  - The **second step** in Huffman's procedure is to code each reduced source, starting with the smallest source and working back to the original source.

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				

$$L_{\text{avg}} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2 \text{ bits/pixel}$$

  - The entropy of the source is 2.14 *bits/symbol*.

2. Some Basic Compression Methods

- Huffman coding

  - Huffman's procedure creates the **optimal code** for a set of symbols and probabilities subject to the constraint that the symbols be coded one at a time.

  - The code itself is an **instantaneous uniquely decodable block code**.

    - ✓ **Block code**: each source symbol is mapped into a fixed sequence of code symbols.

    - ✓ **Instantaneous**: each **code word**[1] in a string of code symbols can be decoded without referencing succeeding symbols.

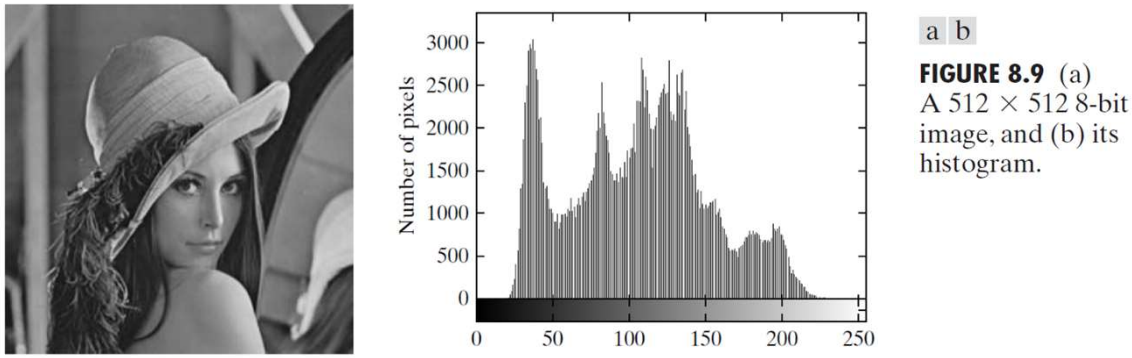
    - ✓ **Uniquely decodable**: any string of code symbols can be decoded in only one way.

[1]Each piece of information or event is assigned a sequence of code symbols, called a code word.

2. Some Basic Compression Methods

- Huffman coding

  - Example:



a b

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

  - Huffman's procedure was used to encode intensities with 7.428 *bits/pixel* - including the Huffman code table that is required to reconstruct the image.

  - The compressed representation exceeds the estimated entropy (7.3838 bits/pixel) only by about 0.6%.

2. Some Basic Compression Methods

- Arithmetic Coding

JBIG1
JBIG2
JPEG-2000
H.264
MPEG-4 AVC

- Unlike the variable-length codes of the previous two sections, arithmetic coding generates **nonblock** codes.

- Example: a five-symbol **message**, $a_1a_2a_3a_3a_4$, from a four-symbol source is coded.

- At the start of the coding process, the **message** is assumed to occupy the entire **half-open interval** [0, 1).

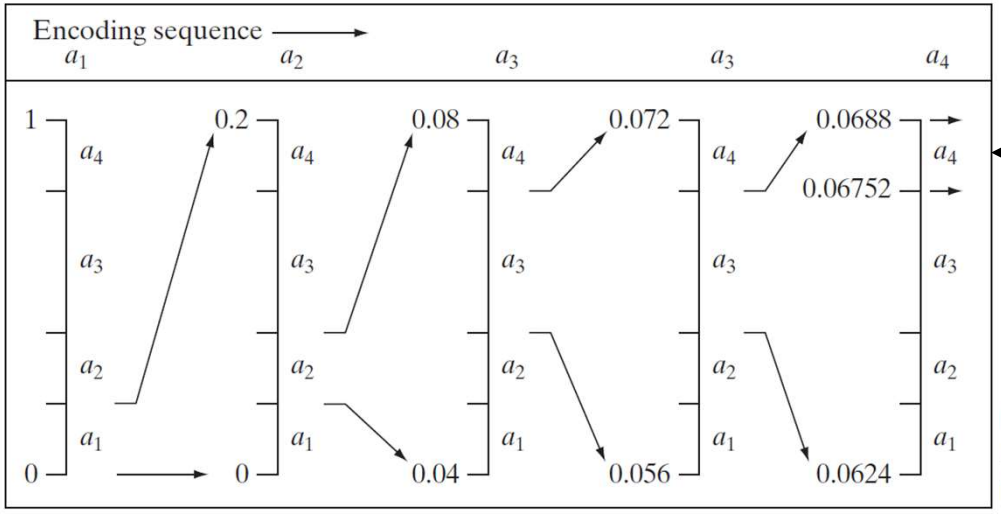
- This interval is subdivided initially into four regions based on the probabilities of each source symbol.

2. Some Basic Compression Methods

- Arithmetic Coding

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

TABLE 8.6
Arithmetic coding example.



$$T = \frac{0.0688 + 0.06752}{2} = 0.0682$$

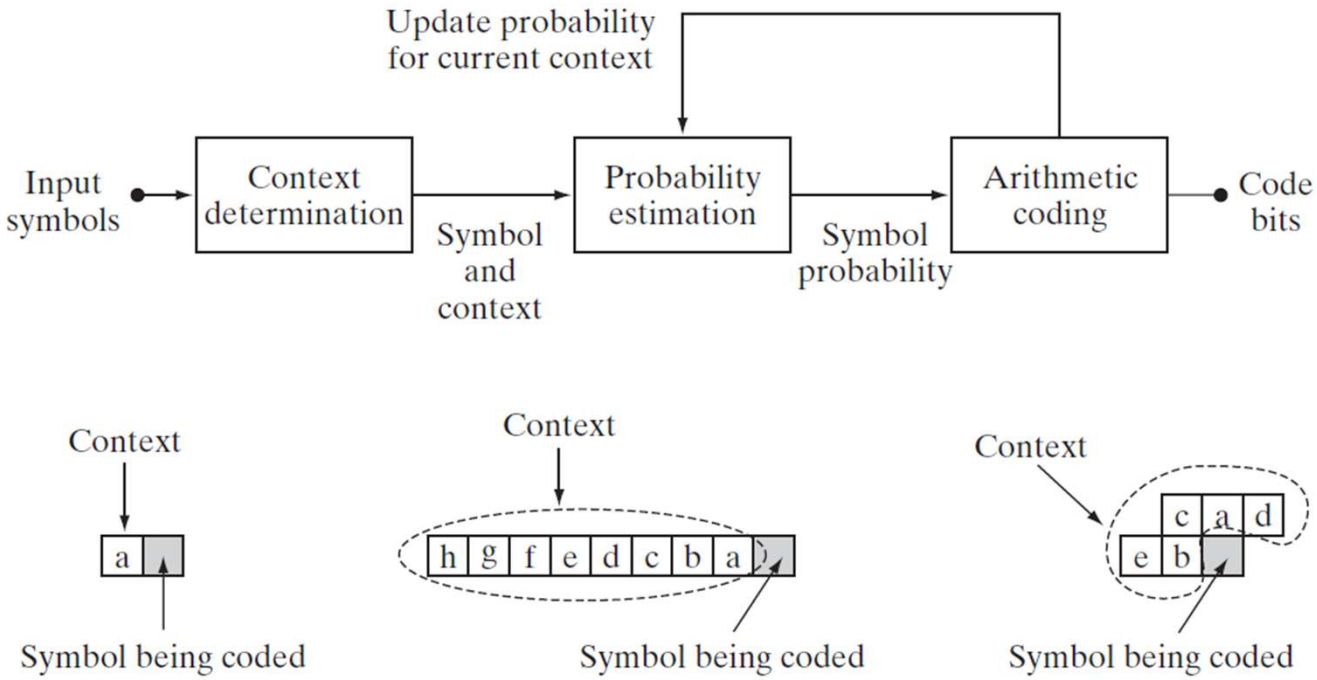
FIGURE 8.12
Arithmetic coding procedure.

2. Some Basic Compression Methods

- Adaptive context dependent probability estimates
  - With accurate input symbol probability models coded, arithmetic coders are near optimal.
  - However, inaccurate probability models can lead to non-optimal results
  - A simple way to improve the accuracy of the probabilities employed is to use an **adaptive**, **context dependent** probability model.

2. Some Basic Compression Methods

- Adaptive context dependent probability estimates



a
b c d

FIGURE 8.13 (a) An adaptive, context-based arithmetic coding approach (often used for binary source symbols). (b)–(d) Three possible context models.

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

GIF
TIFF
PDF

- A key feature of LZW coding is that it requires no a priori knowledge of the probability of occurrence of the symbols to be encoded.
- The **coding dictionary** is created while the data are being encoded.
- Until recently it was protected under a US patent.
- Has been integrated into a variety of mainstream imaging file formats, including GIF, TIFF and PDF.
- The PNG format was created to get around LZW licensing requirements.

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Removes **coding redundancy** and some of the image's **spatial redundancy**.

  - At the onset of the coding process, a codebook or dictionary containing the source symbols to be coded is constructed.

  - For 8-bit monochrome images, the first 256 words of the dictionary are assigned to intensities 0, 1, 2,…, 255.

  - As the encoder sequentially examines image pixels, intensity sequences that are not in the dictionary are placed in algorithmically determined locations.

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

  - 9-bit, 512-word dictionary

Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = ∅
2. C = 39
3. P+C = 39
4. P+C ∈ dictionary:

   ✓ P = P+C = 39

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 39
2. C = 39
3. P+C = 39|39
4. P+C ∉ dictionary:

   - ✓ Output code for P = 39
   - ✓ Add P+C to the dictionary
   - ✓ P = C = 39

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 39
2. C = 126
3. P+C = 39|126
4. P+C ∉ dictionary:

   ✓ Output code for P = 39
   ✓ Add P+C to the dictionary
   ✓ P = C = 126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 126
2. C = 126
3. P+C = 126|126
4. P+C ∉ dictionary:

   - ✓ Output code for P = 126
   - ✓ Add P+C to the dictionary
   - ✓ P = C = 126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 126
2. C = 39
3. P+C = 126|39
4. P+C ∉ dictionary:

   - ✓ Output code for P = 126
   - ✓ Add P+C to the dictionary
   - ✓ P = C = 39

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 39
2. C = 39
3. P+C = 39|39
4. P+C ∈ dictionary:

   ✓ P = P+C = 39|39

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 39|39
2. C = 126
3. P+C = 39|39|126
4. P+C ∉ dictionary:

   - ✓ Output code for P = 39|39
   - ✓ Add P+C to the dictionary
   - ✓ P = C = 126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 126
2. C = 126
3. P+C = 126|126
4. P+C ∈ dictionary:

   ✓ P = P+C = 126|126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

  - Example

P: Previous
C: Current

1. P = 126|126
2. C = 39
3. P+C =126|39|39
4. P+C ∉ dictionary:

   - ✓ Output code for P = 126|126
   - ✓ Add P+C to the dictionary
   - ✓ P = C = 39

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Run-Length Coding

CCITT
JBIG2
JPEG
M-JPEG
MPEG-1,2,4
BMP

- Images with repeating intensities can often be compressed by representing runs of identical intensities as **run-length pairs**.
- The technique, referred to as **run-length encoding** (RLE) became the standard compression approach in facsimile (FAX) coding.
- Compression is achieved by eliminating a simple form of **spatial redundancy**.
- If there are not enough equal intensity runs, compression is not effective and the **file may expand**.

2. Some Basic Compression Methods

- Run-Length Coding

  - Run-length encoding is particularly **effective** when compressing **binary images**, because adjacent pixels are more likely to be identical.

  - In addition, each image row can be represented by a sequence of lengths only.

  - Algorithm:

    1. Let $s \leftarrow 0$.
    2. While there are bits to encode:
       a. Read the next n consecutive bits equal to s
       b. Write n
       c. $s \leftarrow (s + 1) \bmod 2$

2. Some Basic Compression Methods

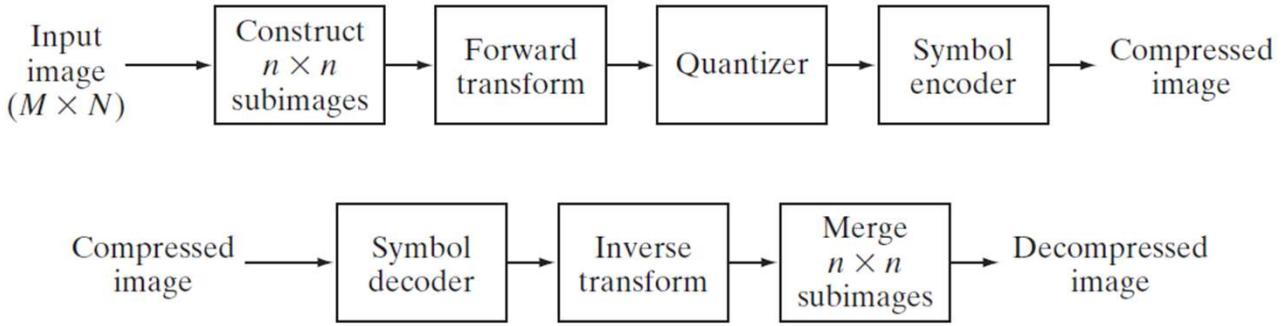
- Block Transform Coding

JPEG
M-JPEG
MPEG-1, 2, 4
H.261, H.262,
H.263, and H.264
DV and HDV
VC-1

- Part of this section was already discussed in "Topic 07 - Image Transforms"
- Here we consider a compression technique that divides an image into **small non-overlapping blocks** and processes the blocks independently.
- A reversible, linear transform is used to map each block into a set of **transform coefficients**, which are then **quantized** and coded.
- For most images, a significant number of the coefficients have **small magnitudes** and can be **coarsely quantized** with little image distortion.

2. Some Basic Compression Methods

- Block Transform Coding



a
b

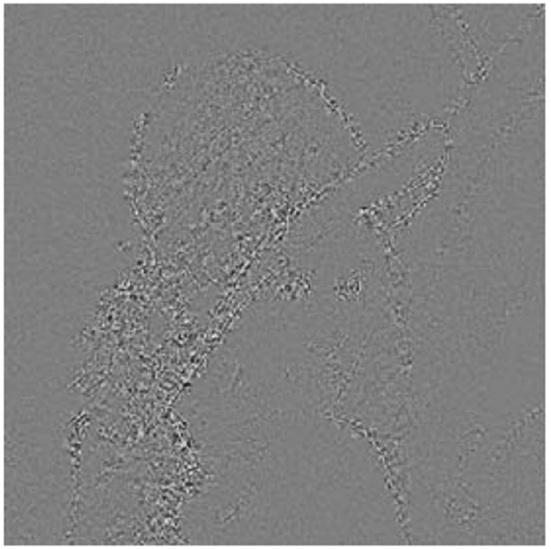
FIGURE 8.21
A block transform coding system:
(a) encoder;
(b) decoder.

2. Some Basic Compression Methods

- Block Transform Coding
  - Bit allocation
    - ✓ The **reconstruction error** is a **function of** the relative importance of the **coefficients** that are **discarded**.
    - ✓ In most transform coding systems, the retained coefficients are selected on the basis of
      - ✓ maximum variance (**zonal coding**); or
      - ✓ maximum magnitude (**threshold coding**).
    - ✓ The overall process of **truncating**, **quantizing**, and **coding** the coefficients of a transformed subimage is commonly called **bit allocation**.

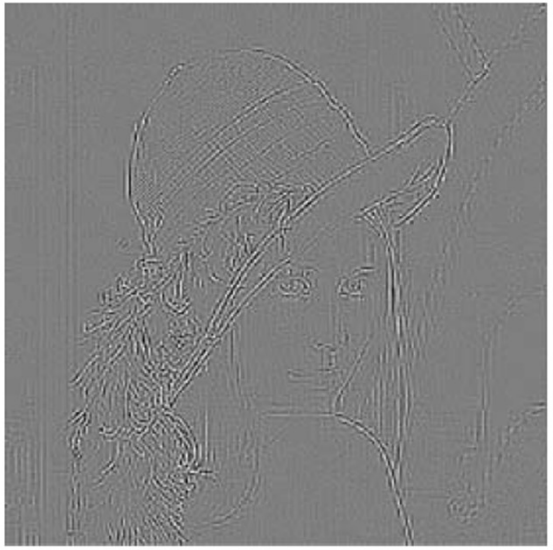
2. Some Basic Compression Methods

- Block Transform Coding
  - Bit allocation
    - ✓ **Threshold coding**: this result was obtained by keeping the eight largest DCT coefficients.



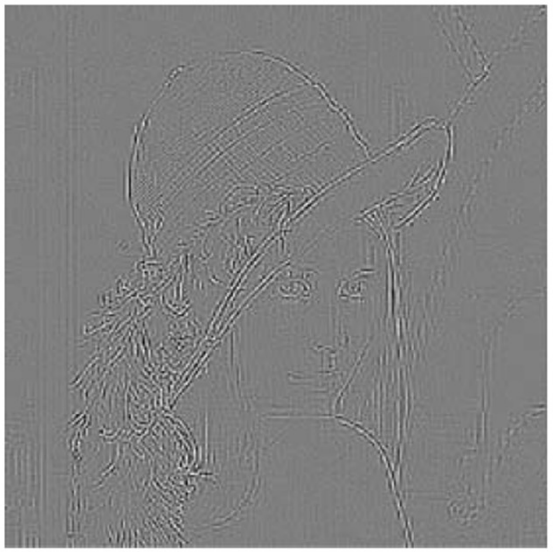
2. Some Basic Compression Methods

- Block Transform Coding
  - Bit allocation
    - ✓ **Zonal coding**: each coefficient was considered a random variable whose distribution could be computed over the ensemble of all subimages.



2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Bit allocation
    - ✓ **Zonal coding**: the 8 distributions of largest variance were located and used to determine the coefficients that were retained.



2. Some Basic Compression Methods

- Block Transform Coding
  - Bit allocation
    - ✓ In both cases **87.5%** of the DCT coefficients of each subimage were discarded

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.

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Threshold coding implementation
    - ✓ The underlying concept is that the transform coefficients of **largest magnitude** make the **most significant contribution**.
    - ✓ There are some ways to threshold a transformed subimage.
    - ✓ We will assume that the threshold can be varied as a **function of the location** of each coefficient within the subimage.
      - ○ In this case, thresholding and quantization can be combined.

2. Some Basic Compression Methods

- Block Transform Coding

  - Threshold coding implementation

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

where $\hat{T}(u,v)$ is a thresholded and quantized approximation of $T(u,v)$ and Z is an element of the transform normalization array

$$\mathbf{Z} = \begin{bmatrix} Z(0,0) & Z(0,1) & \dots & Z(0,n-1) \\ Z(1,0) & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ Z(n-1,0) & Z(n-1,1) & \dots & Z(n-1,n-1) \end{bmatrix}$$

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Threshold coding implementation
    - ✓ The denormalized array, denoted $\dot{T}(u,v)$ is an approximation of $\hat{T}(u,v)$.

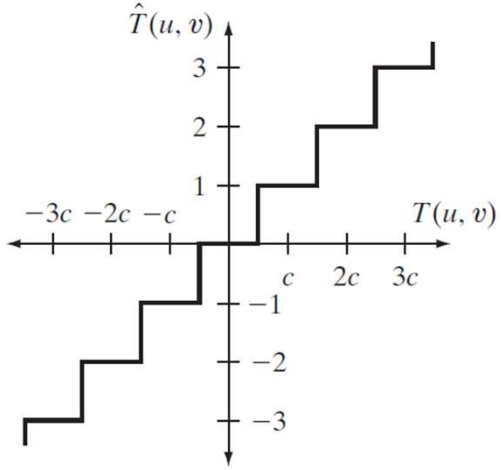
$$\dot{T}(u,v) = \hat{T}(u,v)Z(u,v)$$

    - ✓ The inverse transform of $\dot{T}(u,v)$ yields the decompressed subimage approximation.

2. Some Basic Compression Methods

- Block Transform Coding
  - Threshold coding implementation
    - ✓ The graphics bellow depicts the case in which Z(u, v) is assigned a particular value *c*.
    - ✓ Note that $\hat{T}(u,v)$ assumes an integer value *k* if and only if

$$kc - \frac{c}{2} \leq T(u,v) < kc + \frac{c}{2}$$



2. Some Basic Compression Methods

- Block Transform Coding

  - Threshold coding implementation

    - ✓ If $Z(u,v) > 2T(u,v)$, then $\hat{T}(u,v) = 0$ and the transform coefficient is discarded.

    - ✓ When *k* is represented with a variable-length code that increases in length as the magnitude of increases, the number of bits used to represent T(u, v) is controlled by the value of c.

    - ✓ Thus the elements of Z can be scaled to achieve a variety of compression levels.

2. Some Basic Compression Methods

- Block Transform Coding

  - Threshold coding implementation

    - ✓ The following array has been used extensively in the JPEG standardization.

    - ✓ Each coefficient is quantized according to its psychovisual importance.

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

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Threshold coding implementation

a b c
d e f

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**.

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Threshold coding implementation

a b c
d e f

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**.

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Threshold coding implementation
    - ✓ Quantization matrix: Z, 2Z, 4Z, 8Z, 16Z, 32Z
    - ✓ Compression: 12, 19, 30, 49, 85, 182:1
    - ✓ rms error: 3.83, 4.93, 6.62, 9.35, 13.94, 22.46

2. Some Basic Compression Methods

- Block Transform Coding

  - One of the most popular and comprehensive continuous tone, still frame compression standards is the **JPEG standard**.

  - It defines three different coding systems.

  - In the baseline system, the input and output data precision is limited to 8 bits, whereas the quantized DCT values are restricted to 11 bits.

  - The compression itself is performed in three sequential steps: DCT computation, quantization, and variable-length code assignment.

2. Some Basic Compression Methods

- Block Transform Coding

  - The image is first **subdivided into blocks** of size 8x8, which are processed left to right, top to bottom.

  - As each block is encountered, its 64 pixels are **level-shifted** by subtracting the quantity 2^{k-1}, where 2^k is the maximum number of intensity levels.

  - **The 2-D discrete cosine transform** of the block is then computed, **quantized** and **reordered** to form a 1-D sequence of quantized coefficients.

  - The JPEG coding procedure is designed to take advantage of the **long runs of zeros** that normally result from the reordering

2. Some Basic Compression Methods

- Block Transform Coding

  - The nonzero AC coefficients are coded using a **variable-length** code that defines the coefficient values and number of preceding zeros.

  - The DC coefficient is **difference coded** relative to the DC coefficient of the previous subimage.

  - **Default coding tables** and quantization arrays are provided for both color and monochrome processing.

  - The user is **free to construct** custom tables and/or arrays.

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Original subimage

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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Pixels values intensity-shifted by -128

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ DCT of intensity-shifted array

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

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

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Normalization

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

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

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ The coefficients are reordered using the zig-zag scan

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

−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

[−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]

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Modified Huffman code tables

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

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

TABLE A.4 JPEG default DC code (luminance).

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

TABLE A.5 JPEG default AC code (luminance).

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Modified Huffman code tables

$$-26\ -3\ 1\ -3\ -2\ -6\ 2\ -4\ 1\ -4\ 1\ 1\ 5\ 0\ 2\ 0\ 0\ -1\ 2\ 0\ 0\ 0\ 0\ 0\ -1\ -1\ \text{EOB}$$

$$\downarrow$$

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

    - ✓ The resulting compression ratio is 512/92, or about 5.6:1

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Modified Huffman code tables

      - DC coefficients: compute difference between the current DC coefficient dc_i and that of the previously dc_{i-1} encoded subimage.

      - If $[d_i - d_{i-1}] = [-26 - (-17)]$ or **-9** then the DC difference category is 4.

      - The proper base code for a category 4 is 101 (a 3-bit code).

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
    - ✓ Modified Huffman code tables
      - ○ The total length of a completely encoded category 4 DC coefficient is 7 bits.
      - ○ The remaining k = 4 bits must be generated from the LSBs of the difference value.
      - ○ If the difference is positive, take the *k* LSBs bits; else take the negative difference minus 1.

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
    - ✓ Modified Huffman code tables
      - ○ $-9_{10} = 1\mathbf{0111}_2$
      - ○ -26 (JPEG)➔ $(101: \mathbf{0111} - 1)_2$ ➔ $101{:}0110_2$

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Modified Huffman code tables

      - The nonzero **AC coefficients** of the reordered array are **coded similarly.**

      - The **principal difference** is that each default AC Huffman code word depends also on the number of **zero-valued coefficients** preceding the nonzero coefficient to be coded.

2. Some Basic Compression Methods

- Block Transform Coding
  - ➢ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
    - ✓ Modified Huffman code tables
      - ○ Consider the first AC coefficient -3.
      - ○ It belongs to category 2 and is preceded by no zero-valued coefficients.
      - ○ The base code for a category 2/run 0 is 01.
      - ○ The last 2 bits are generated by the same process used to arrive at the LSBs of the DC difference code.

2. Some Basic Compression Methods

- Block Transform Coding
  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
    - ✓ Modified Huffman code tables
      - -3_{10} ➔ $1\mathbf{01}_2$
      - -3_{10} (JPEG)➔ $01: (\mathbf{01}\text{-}1)_2 = (01{:}00)_2$

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Decoding: denormalization

$$\dot{T}(0,0) = \hat{T}(0,0)Z(0,0) = (-26)(16) = -416$$

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

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Decoding: denormalization

$$\dot{T}(0,0) = \hat{T}(0,0)Z(0,0) = (-26)(16) = -416$$

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

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Decoding: inverse DCT

−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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ Decoding: level shifting

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

2. Some Basic Compression Methods

- Block Transform Coding

  - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

    - ✓ 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

Original

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

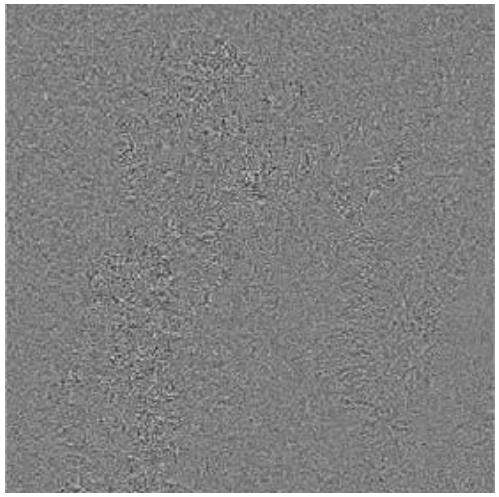
Reconstructed

3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

  - This result was obtained by **dividing** the original image into subimages of **size 8 x 8** using the DCT, **truncating** 50% of the resulting coefficients, and **taking the inverse transform** of the truncated coefficient arrays.



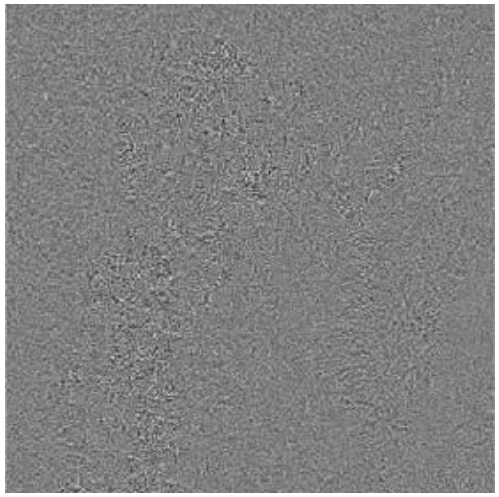


3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

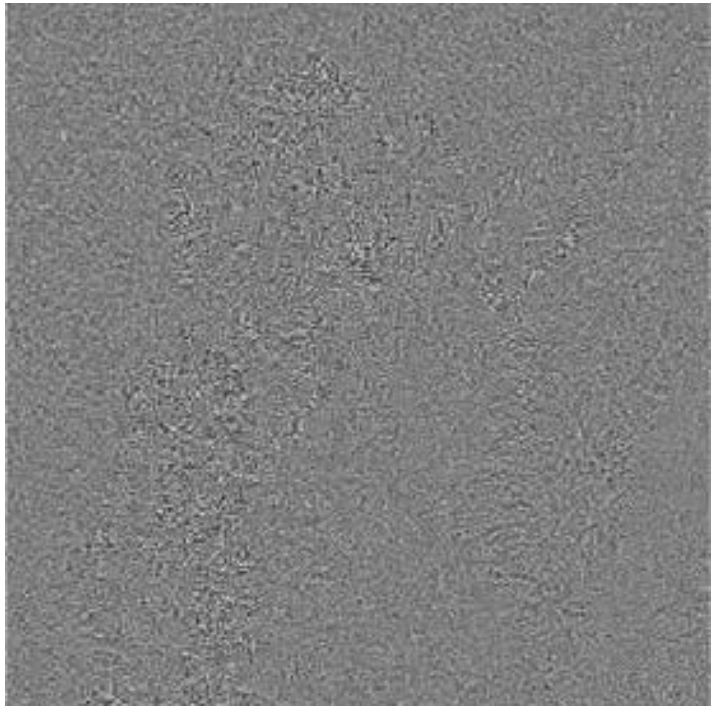
  - The 32 retained coefficients were selected on the basis of maximum magnitude. In this example, the *rms* error was 1.13.





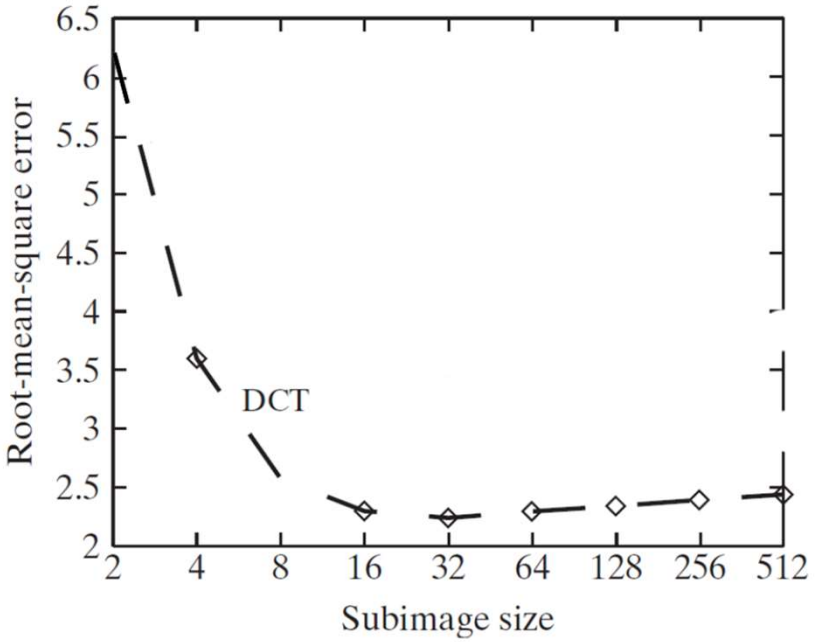
3. Discrete Cosine Transform

- MATLAB: s109Subimage8Size.m



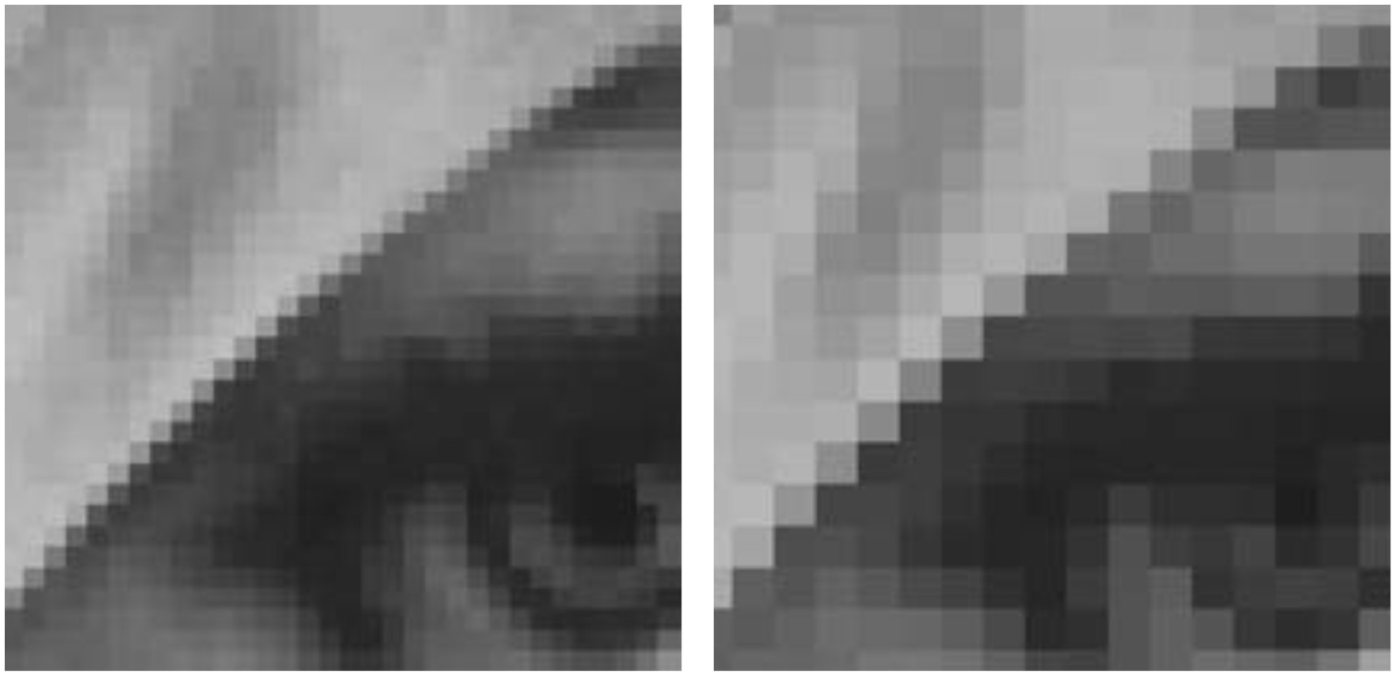
3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
  - Truncating 75% of coefficients



3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

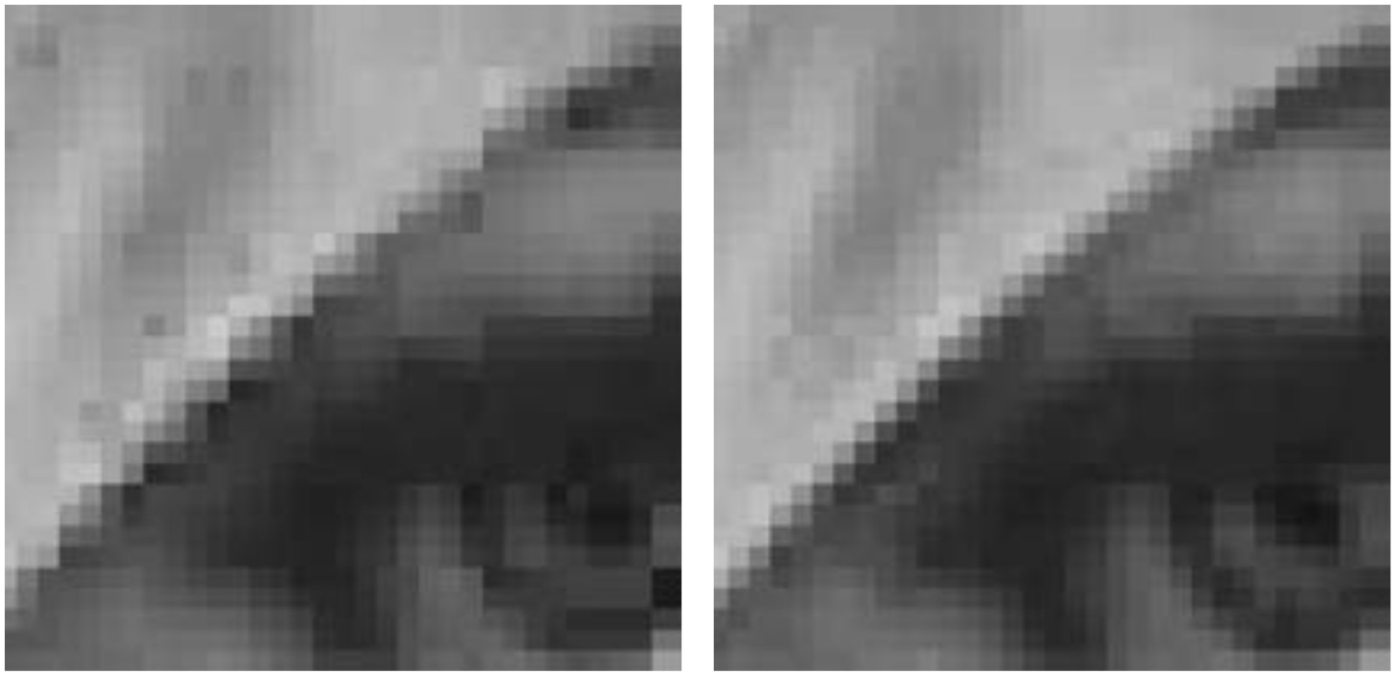


a b c d

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).

3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

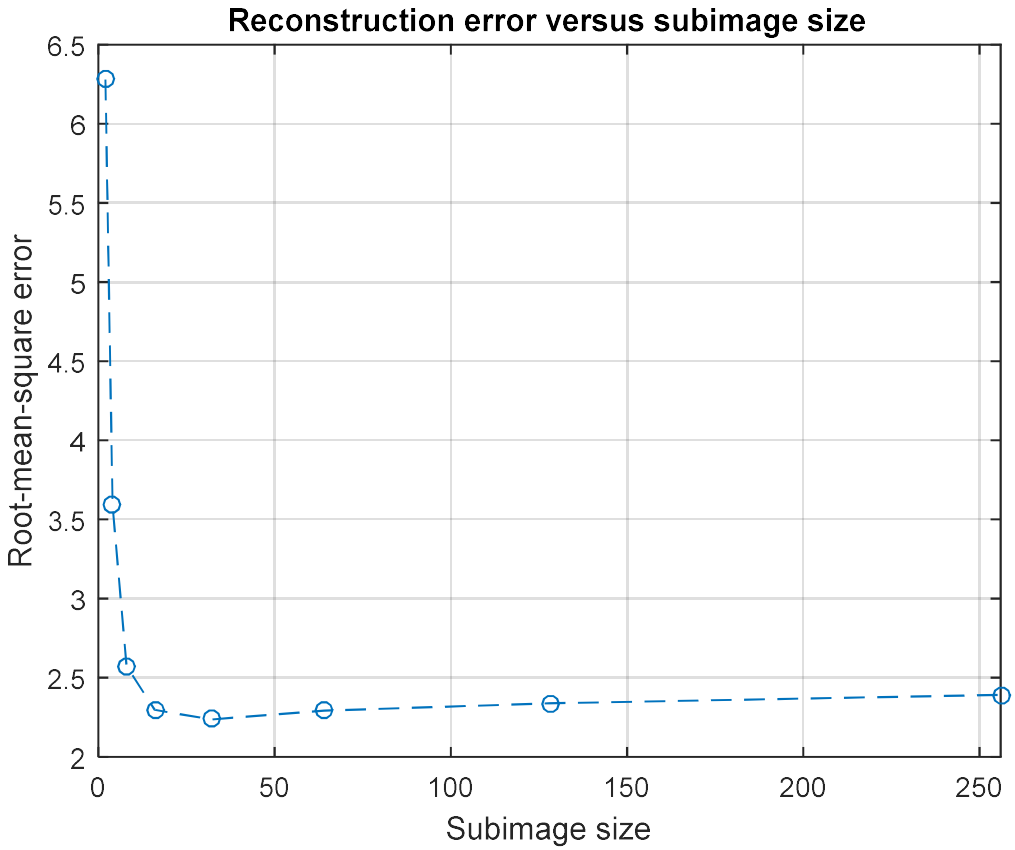


a b c d

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).

3. Discrete Cosine Transform

- MATLAB: s113SubimageNSize.m



3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
  - The **larger** the **size** the **more decorrelation** the transform coding can achieve.
  - However, the **correlation** between pixels becomes **insignificant** when the distance between pixels becomes large, e.g., **20 pixels**.
  - Images are subdivided so that the **correlation between adjacent** subimages is **reduced** to some **acceptable level**.
  - The **computation** of the subimage **transforms** is **simplified** if the block size is an integer power of 2.

3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
  - In adaptive transform coding, a large block **cannot adapt** to **local statistics** well.
  - **Large size** implies a possibly severe effect of **transmission error** in reconstructed images.
  - As will be shown in video coding, transform coding is used together with motion compensated coding, where blocks of 4, 8 and 16 are most often.
  - In general, both the level of compression and computational complexity increase as the subimage size increases.
  - The most popular subimage sizes are 8x8 and 16x16.

4. Further Reading