



PES
UNIVERSITY

CELEBRATING 50 YEARS

DIGITAL IMAGE PROCESSING - 2

Lecture 1: Introduction

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Introduction & Course Overview

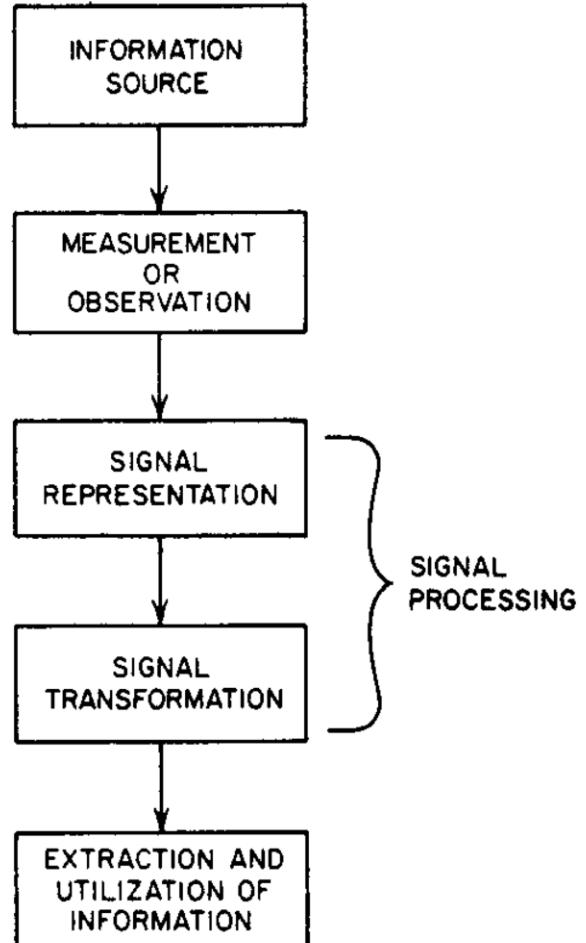
Dr. Shruthi M L J

Department of Electronics & Communication Engineering

“Vision is the most advanced of our senses, so it is not surprising that images play most important role in human perception”



Signal Processing:



Signal Processing Courses

- Speech Processing
- **Digital Image Processing (DIP)**
- Biomedical Signal Processing
- Radar Signal Processing
- Estimation and detection
- Multirate signal processing
- Adaptive Signal Processing
- Pattern Recognition and classification
-
- **DIP is a sub area of Signal processing**

- **Signal Processing**
 - Signals and Systems (S & S)
 - Digital Signal Processing (DSP)
 - Knowledge of “Digital Image Processing” (DIP)

Digital Image Processing

- Digital image processing encompasses processes whose inputs and outputs are images and in addition encompasses processes that extract attributes from images, up to and including the recognition of individual objects

Examples

- Automated analysis of text:
 - Acquiring an image of the area containing text, preprocessing that image, extracting (segmenting) individual characters, describing the characters in a form suitable for computer processing and recognizing those individual characters – **Digital image processing**
 - Making sense of the content of the page- **Image analysis** and **Computer Vision** depending on complexity

Applications of DIP

- Image Sharpening and restoration
- Image Segmentation
- Image Morphology
- Transmission and encoding
- Machine/Robot vision
- Color processing
- Pattern recognition
- Video Processing
- Image/Video Compression
-

Introduction

Course Description

- This course deals with image compression, morphological image processing, image and texture segmentation, wavelet based image processing and object recognition

Course Objectives:

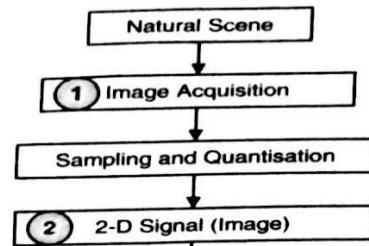
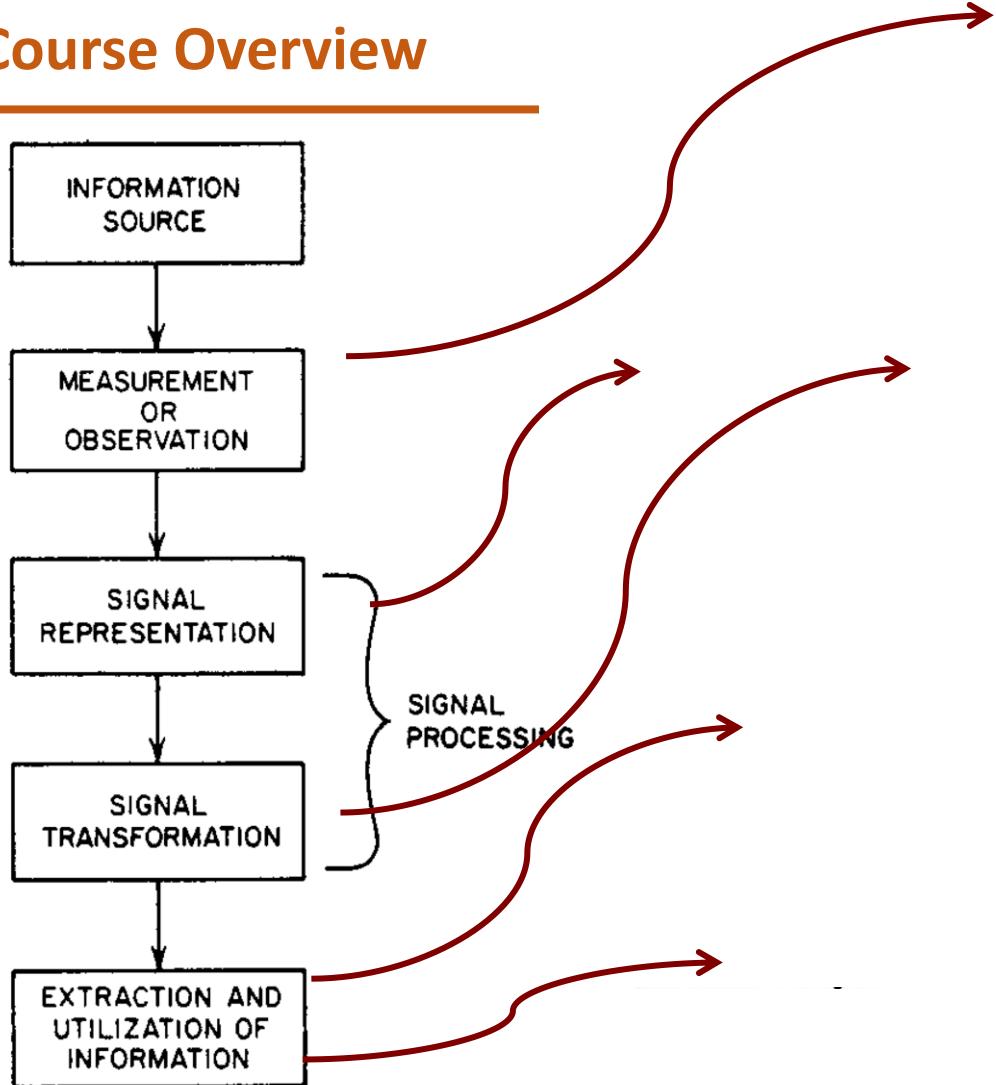
- To Expose students to advanced concepts of image processing
- Investigate current representations and methods in image processing such as wavelets and morphology
- To design and implement algorithms that perform basic image processing operations like filtering of noise and image enhancement
- To design, analyse and implement algorithms for advanced image analysis like image compression, image reconstruction, image segmentation

Course Outcomes

- To compare and use different tools for image analysis in transformed domain (wavelet/Fourier transform)
- To apply the notions learnt in the course to practical image processing problems
- To implement techniques for image enhancement, filtering and compression
- To process and analyze image data
- To learn how higher-level image processing concepts such as edge detection, segmentation, representation can be implemented and used

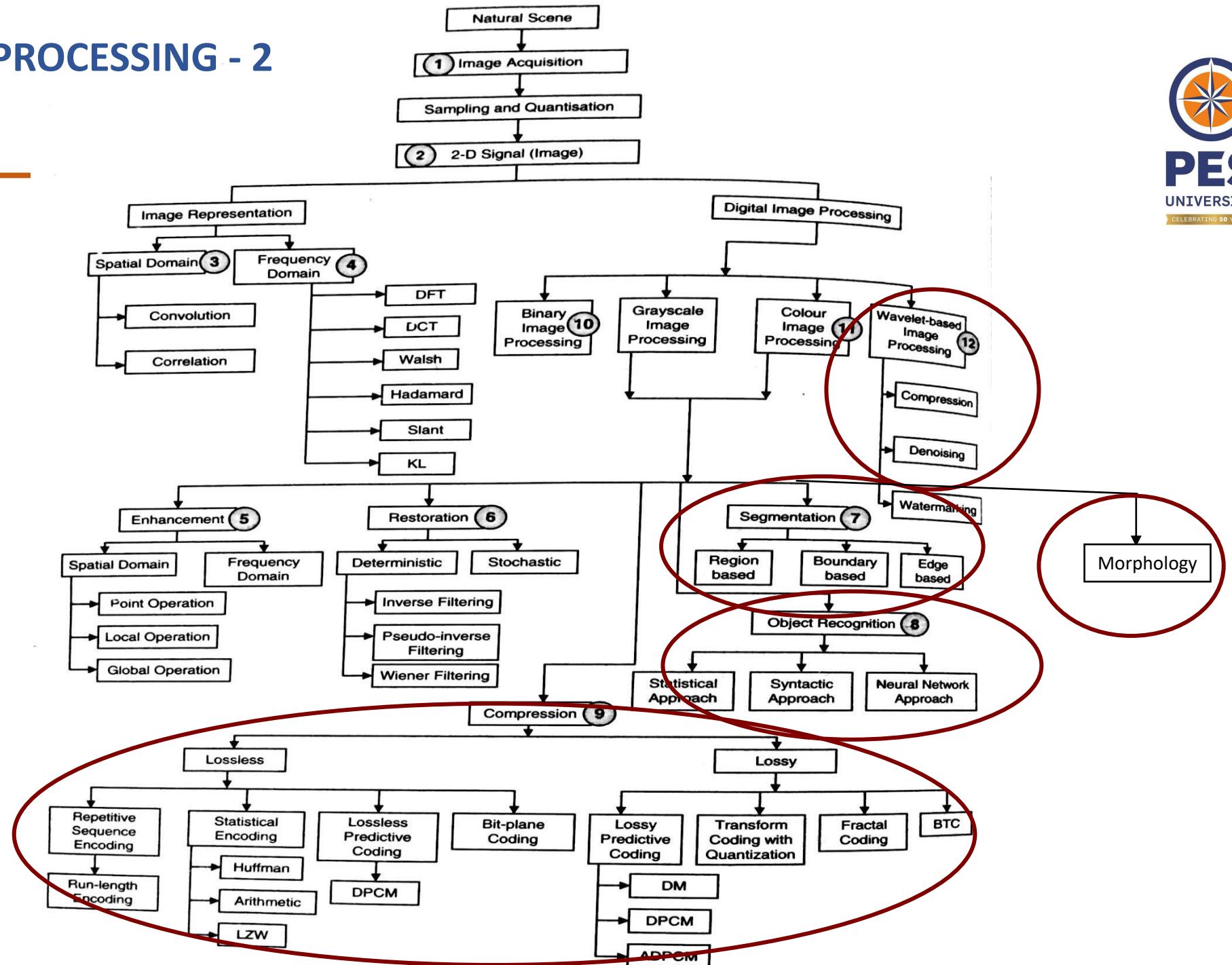
DIGITAL IMAGE PROCESSING - 2

Course Overview



DIGITAL IMAGE PROCESSING - 2

DIP - 2 Overview



Modules:

- **Unit 1: Image Compression**
- **Unit 2: Morphological Image Processing**
- **Unit 3: Image Segmentation**
- **Unit 4: Wavelet based image processing**
- **Unit 5: Object Recognition**

Unit 1: 12 Hrs

Image Compression:

- Need for image compression
- Image Compression models
- Elements of Information Theory
- Error free compression(Lossless Compression)
- Lossy compression
- Image Compression Standards

Unit 2: 12 Hrs

Morphological Image Processing :

- Dilation and Erosion
- Opening and Closing
- The Hit-or-Miss Transformation
- Some Morphological Algorithms
- Extensions to Gray scale Images

Unit 3: 11 Hrs

Image Segmentation:

- Classification of image Segmentation techniques
- Region approach-segmentation
- Clustering techniques
- Segmentation based on thresholding
- Water -shed transformation
- Edge based segmentation
- Classification of edges
- Edge detection and Linking
- Hough Transform
- Shape representation

Unit 4: 10 Hrs

Wavelet based image processing :

- Evolution of wavelet transform
- wavelet transform
- 2D continuous wavelet transform
- Multi resolution analysis
- Examples of wavelets
- Wavelet based image compression

Unit 5: 11 Hrs

Object Recognition :

- Need for an object-recognition System
- Automated Object-recognition Systems
- Patterns and Pattern Class
- Selection of Measurement Parameters
- Relation between Image Processing and Object Recognition
- Approaches to object recognition (Baye's Parametric Classification, Neural Network approach to object recognition, Template-Matching based Object Recognition)
- Applications of Object Recognition

DIGITAL IMAGE PROCESSING - 2

Reference Books



Textbook:

“Digital Image Processing”, R.C.Gonzalez and R.E.Woods, 4th Edition, Pearson, 2018.

Reference Books:

“Digital Image Processing”, S Jayaraman, S Esakkirajan and T

Veerakumar, Mc Graw Hill, 2009.

“Digital Image Compression Techniques”, M. Rabbani and P W Jones,

SPIE Press, 1991.

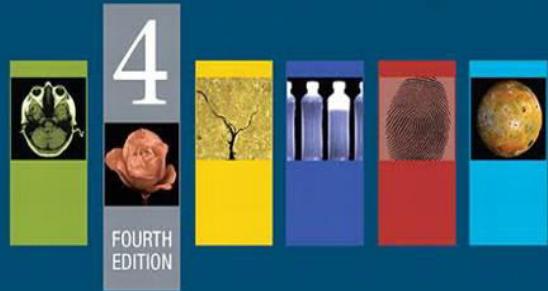
“A Wavelet Tour of Signal Processing”, S Mallat, Second Edition,

Academic Press, 1999.

DIGITAL IMAGE PROCESSING - 2

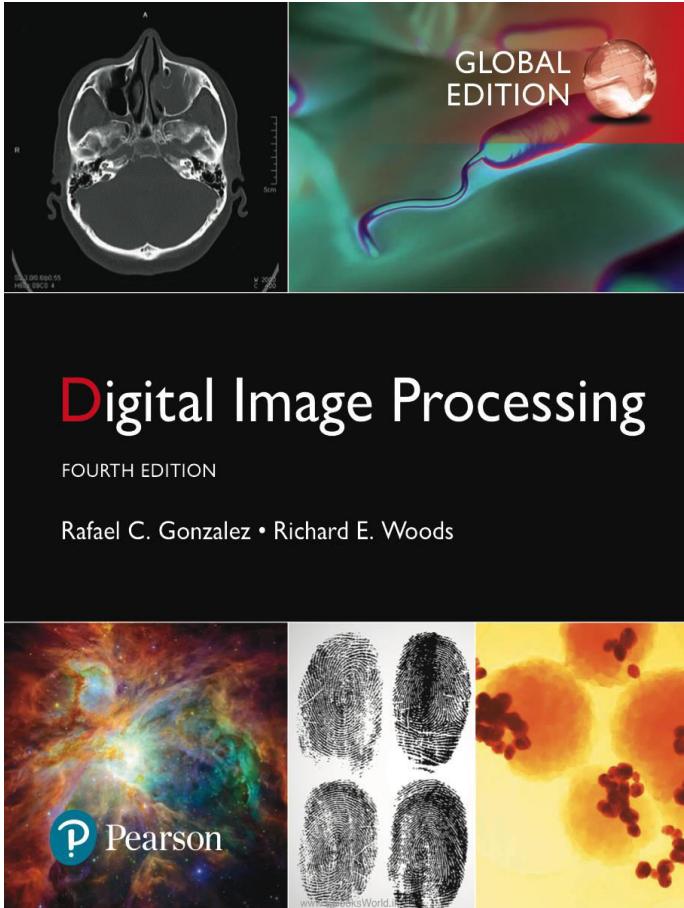
The Text Book

Digital Image Processing



Rafael C. Gonzalez
Richard E. Woods

P Pearson



***“When Something can be
read without effort, great
effort has gone into its
writing”***

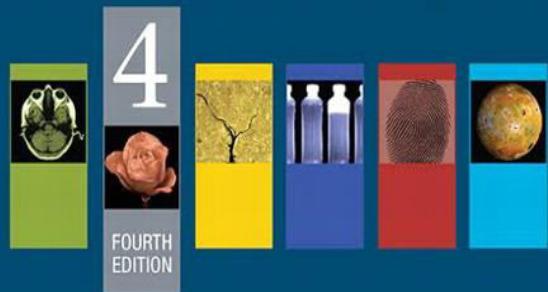
 **Enrique Jardiel Poncela**

DIGITAL IMAGE PROCESSING - 2

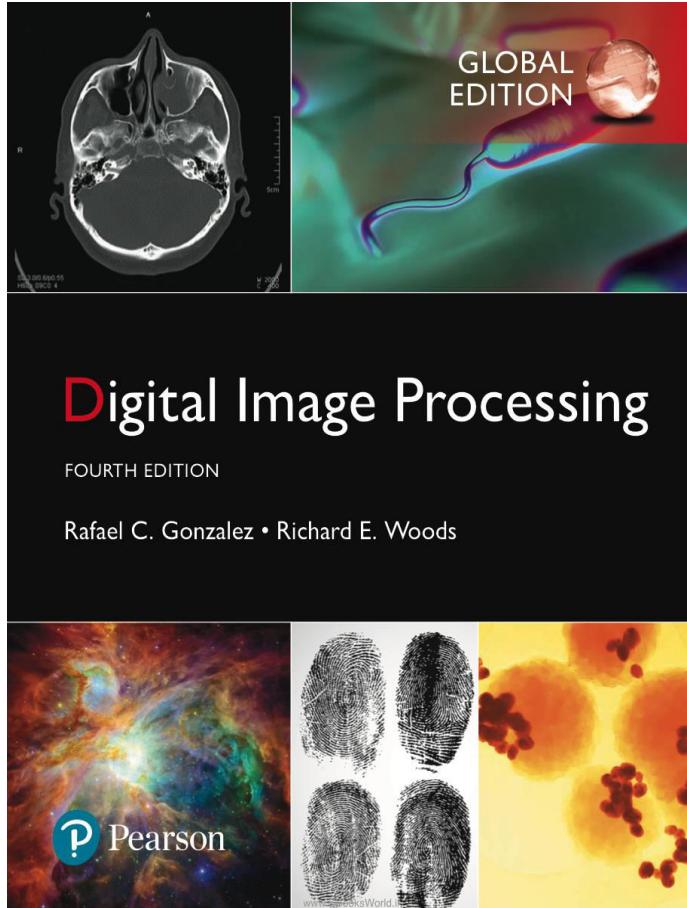
The Course Delivery



Digital Image Processing



Rafael C. Gonzalez
Richard E. Woods



“When Something can be understood without effort, great effort has gone into its teaching”

DIGITAL IMAGE PROCESSING - 2

Course Instructor



- Dr. Shruthi M L J
 - 13 years of teaching and research experience
 - Research Interests:
 - Image /video processing
 - Machine Learning and Artificial Intelligence
 - Signal processing
 - Contact:
 - Email: shruthimlj@pes.edu
 - Chamber: B-Block, B 501
 - Teaching Assistants: TBA

Summary

- Importance of DIP
- Course Overview

Next Session

- Image Processing Basics
- Fundamentals of Image Compression



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu

+91 8147883012



PES
UNIVERSITY
CELEBRATING 50 YEARS

DIGITAL IMAGE PROCESSING - 2

Lecture 2: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Unit 1: Image Compression

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Motivation
- Introduction and overview of course

This Session

- Image Processing Basics
- Introduction to image compression

DIGITAL IMAGE PROCESSING - 2

Course Instructor



Prof. Shruthi M L J

- 13 years of teaching and research experience
- Research Interests:
 - Image /video processing
 - Machine Learning
 - Deep Learning
- Contact:
 - Email: shruthimlj@pes.edu
 - Seating: B-Block, 3rd floor, staff room
- Teaching Assistants: TBA

Course Evaluation Components

ISA-1: 05 Marks

ISA-2: 05 Marks

ISA-3: 05 Marks

ISA-4: 05 Marks

ISA-5: 05 Marks

ISA : 20 Marks (picked from 4 best ISA performances)

Assignments : 10

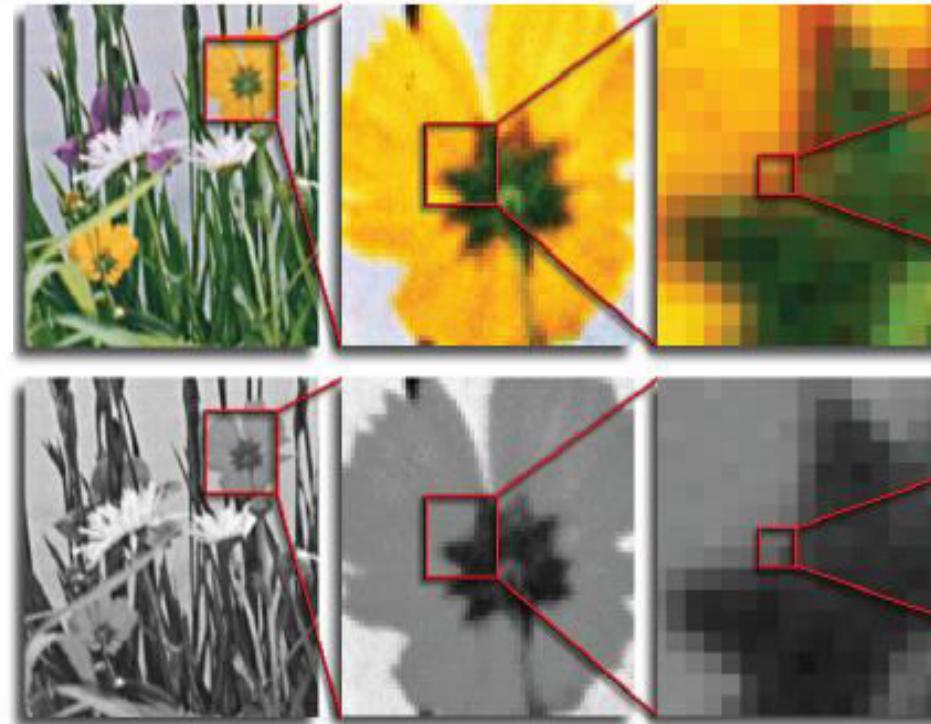
Projects : 20 Marks

ESA: 50 Marks

Digital Image Processing: Basics

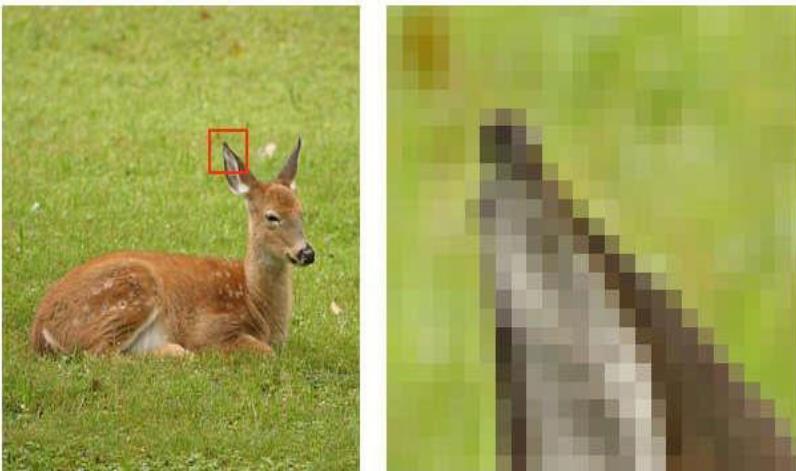
- **Image:** A two-dimensional function $f(x,y)$ where
 - x and y are spatial coordinates.
 - The amplitude of f is called intensity or gray level at the point (x, y)
- **Digital Image:**
- When x, y and amplitude values of f are all finite, discrete quantities
 - Digital image is composed of a finite number of elements, each of which has a particular location and value
- **Digital Image Processing:** Processing of digital images by means of a digital computer

Digital Image



Elements of an Image

- Picture Element
- Pixels
- pels



The smallest square element of a digital image, representing a single color or level of brightness

Image Processing

- Act of converting a captured image from one form to another

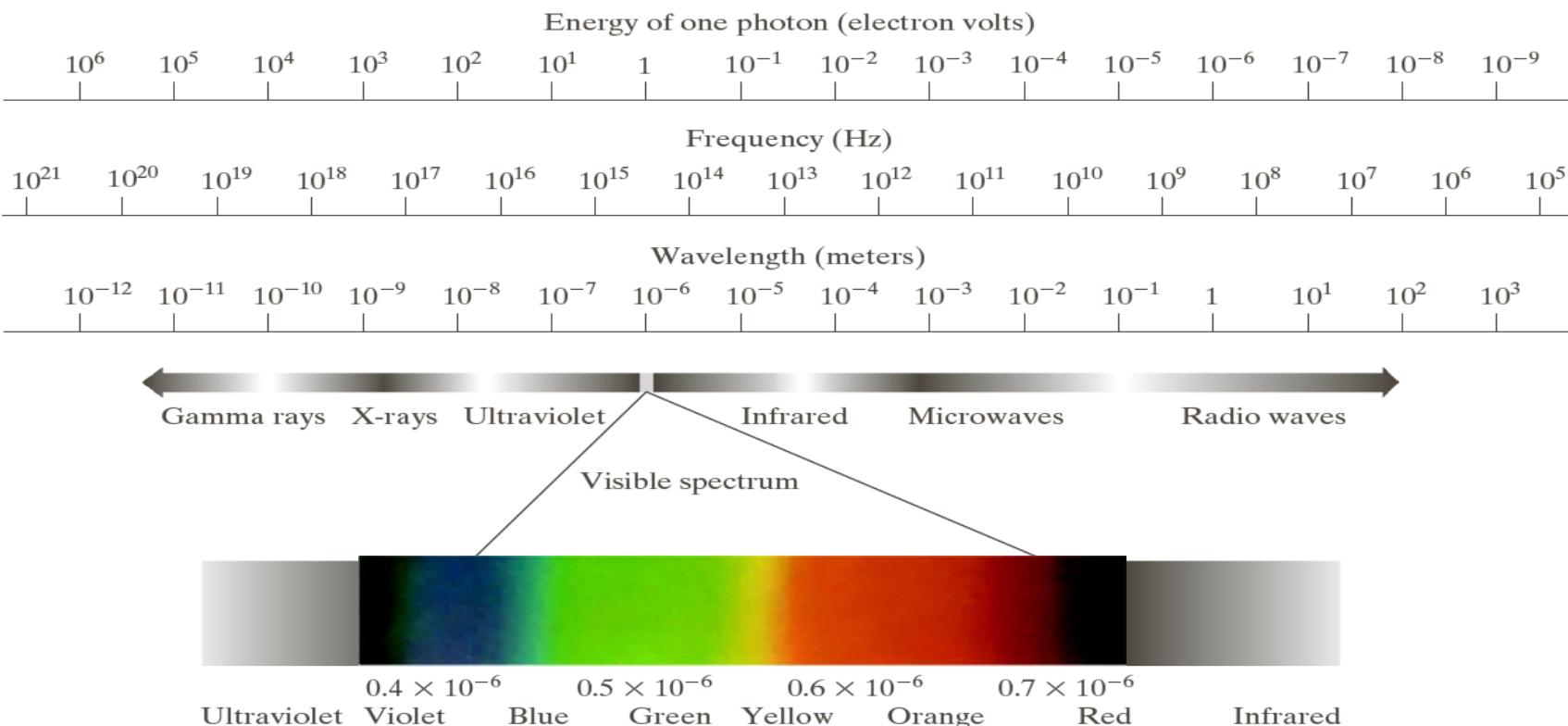


Image Processing / DSP

- DSP: deals with processing of 1-D signals
- IP: deals with visual information that is often in 2 or more dimensions
- Image processing is logical extension of DSP

Computer Vision and Human Visual System

- **Computer Vision:** Machine covers the entire electromagnetic spectrum (gamma to radio waves)



Computer Vision and Human Visual System

- **Human Visual System:** Limited to visual band of EM spectrum
- Imaging machines can operate on images generated by sources that humans are not accustomed to associating with images:
ultrasound, electron microscopy and computer generated images

“Unlike humans, who are limited to the visual band of the electromagnetic spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves”

Image Processing and Computer Graphics

- Very closely related
- Image Processing deals with raster data or bitmaps whereas computer graphics primarily deals with vector data
 - Raster data(bitmap is stored in 2-D matrix form often used to depict real images)
 - Vector images are composed of vectors, which represent mathematical relationships between objects. Vectors are lines or primitive curves used to describe an image

Image Processing and Computer Graphics

- **Computer Graphics:** The algorithms often take numerical data as input and give image as output
- **Image Processing:** Input is often an image
 - **Goal:** To enhance the quality of the image to assist in interpreting it
 - **Result of image processing is often an image or description of an image**
- **Image processing is logical extension of computer graphics**

Image Processing and Computer Graphics

The continuum from image processing to computer vision can be broken up into low-, mid- and high-level processes

Image Processing:

Input: Image

Output: Image

Examples: Noise removal, image sharpening

Image Analysis:

Input: Image

Output: Attributes

Examples: Object recognition, segmentation(partitioning image into regions or objects)

Computer Vision:

Input: Attributes

Output: Understanding

Examples: Scene understanding, autonomous navigation

Input and output are images

Input is image & outputs are attributes extracted form image(edge,contour etc..)

Involves “making sense “ Like cognitive functions associated with human vision

DIGITAL IMAGE PROCESSING - 2

ADIP Overview

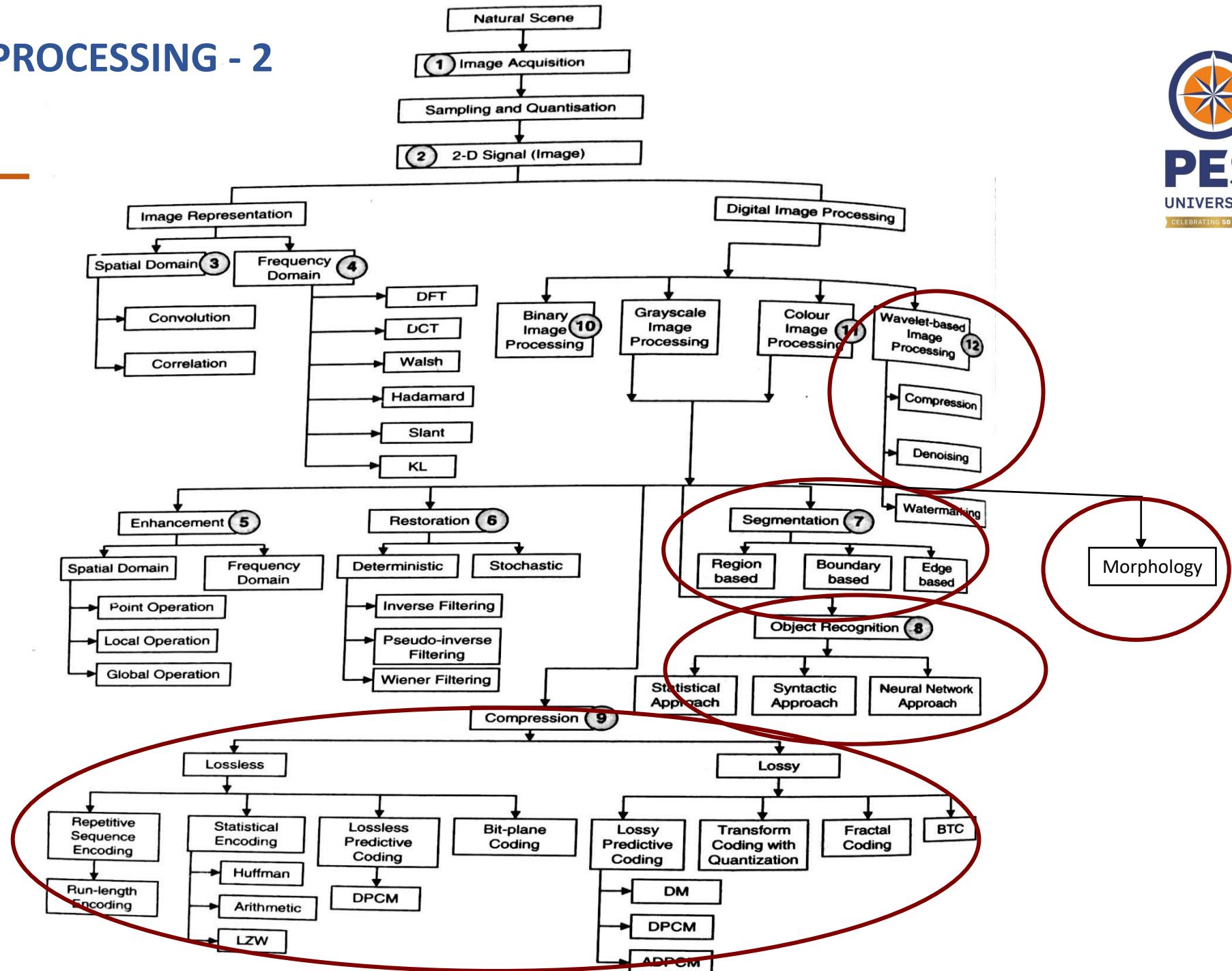


Image Compression

- It is the art and science of reducing the data required to represent an image
- Innumerable number of multimedia data is compressed and decompressed daily (MP4,JPEG,MPEG etc...)
- **One of the most useful and successful technologies in digital image processing**

Summary

- Basic fundamentals of DIP
- Introduction to Image compression

- Image Compression
- Types of redundancies



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 2: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Types of Redundancies

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Image Processing Basics
- Introduction to image compression

This Session

- Image compression:
 - Compression ratio
 - Types of Redundancies

Image Compression

- It is the art and science of reducing the data required to represent an image
- Innumerable number of multimedia data is compressed and decompressed daily (MP4,JPEG,MPEG etc...)
- **One of the most useful and successful technologies in digital image processing**

Fundamentals of image compression

Data Compression:

- The process of reducing the amount of *data* required to represent a given quantity of *information* Here, *data* and *information* are not the same

- Data is 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*.



Fundamentals of image compression

Compression Ratio:

compression ratio is defined as $C = \frac{b}{b'}$

Where b is the number of bits needed to represent an image as a 2-D array of intensity values and b' denotes the number of bits needed in compressed representation of the same information

- *Relative data redundancy*, R , of the representation with b bits is

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

where C , commonly called the *compression ratio*

If $C=10$ then redundancy in representation is

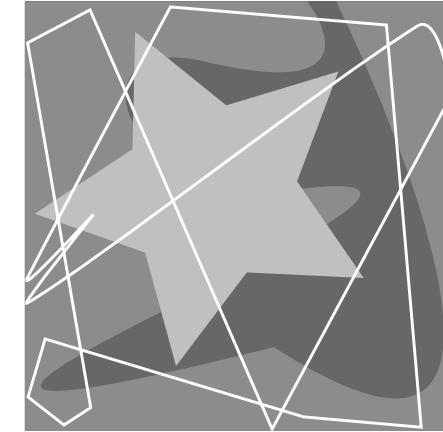
Types of Redundancy

- 2-D intensity arrays suffer from three principal types of data redundancies that can be identified and exploited
 - Coding Redundancy
 - Spatial and temporal redundancy
 - Irrelevant redundancy

Types of Redundancy

- ***Coding redundancy:***

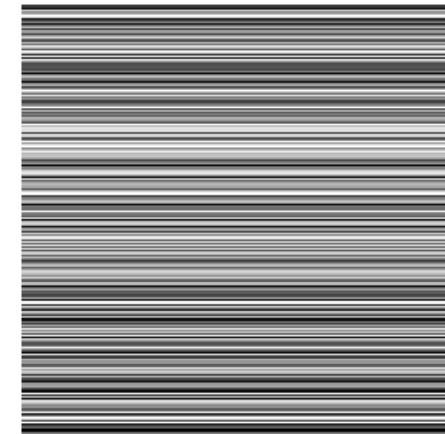
- A code is a system of symbols (letters, numbers, bits, and the like) used to represent a body of information or set of events.
- Each piece of information or event is assigned a sequence of *code symbols*, called a *code word*.
- The number of symbols in each code word is its *length*.
- The 8-bit codes that are used to represent the intensities in most 2-D intensity arrays contain more bits than are needed to represent the intensities.



Computer generated
(256 x 256 x 8)
bit image

Types of Redundancy

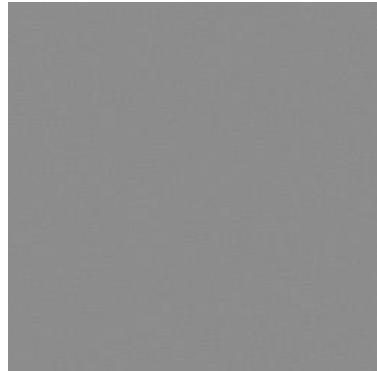
- *Spatial and temporal redundancy :*
 - Because the pixels of most 2-D intensity arrays are correlated spatially (i.e., each pixel is similar to or dependent upon neighboring pixels), information is unnecessarily replicated in the representations of the correlated pixels.
 - In a video sequence, temporally correlated pixels (i.e., those similar to or dependent upon pixels in nearby frames) also duplicate information.



Types of Redundancy

- *Irrelevant information:*

- Most 2-D intensity arrays contain information that is ignored by the human visual system (HVS) and/or extraneous to the intended use of the image.
- It is redundant in the sense that it is not used.



Compression is achieved when one or more redundancy is reduced or eliminated.

Types of Redundancy

Coding Redundancy:

- **Objective:** To find optimal information coding
 - **Assumption:** the intensity values of an image are random quantities
 - Assume that a discrete random variable r_k in the interval $[0, L - 1]$ is used to represent the intensities of an $M \times N$ image, and that each r_k occurs with probability $p_r(r_k)$
- Hence

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

where L is the number of intensity values, and n_k is the number of times that the k_{th} intensity appears in the image

Coding Redundancy Cont..

- If the number of bits used to represent each value of r_k is $l(r_k)$, then the average number of bits required to represent each pixel is

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

- That is, the average length of the code words assigned to the various intensity values is found by summing the products of the number of bits used to represent each intensity and the probability that the intensity occurs.
- Hence the total number of bits required to represent an $M \times N$ image is

Coding Redundancy Cont..

- So with

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

- If the intensities are represented using a *natural m-bit fixed-length code*, the right-hand side of the equation reduces to m bits. That is, $L_{\text{avg}} = m$ when m is substituted for $l(r_k)$
 - The constant m can be taken outside the summation, leaving only the sum of the $p_r(r_k)$ for $0 \leq k \leq L-1$ which equals 1

Summary

- Image compression:
 - Compression ratio
 - Types of Redundancies: Coding Redundancy

- Types of redundancies Cont..



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012



ADVANCED DIGITAL IMAGE PROCESSING

Lecture 1: Introduction

Prof. Shruthi M L J
Department of Electronics &
Communication Engineering

ADVANCED DIGITAL IMAGE PROCESSING

Introduction & Course Overview

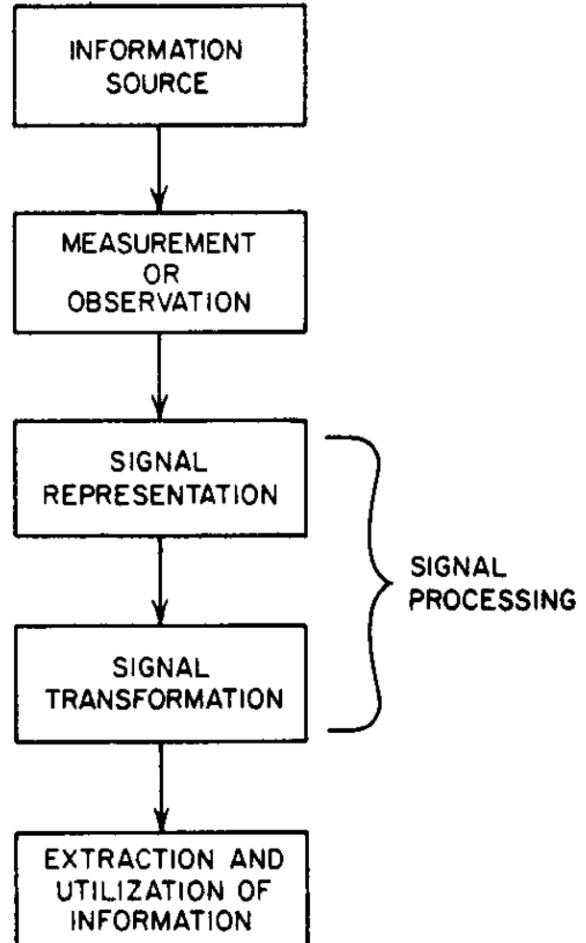
Prof. Shruthi M L J

Department of Electronics & Communication Engineering

“Vision is the most advanced of our senses, so it is not surprising that images play most important role in human perception”



Signal Processing:



ADVANCED DIGITAL IMAGE PROCESSING

Signal Processing Courses



- Speech Processing
- **Digital Image Processing (DIP)**
- Biomedical Signal Processing
- Radar Signal Processing
- Estimation and detection
- Multirate signal processing
- Adaptive Signal Processing
- Pattern Recognition and classification
-
- **DIP is a sub area of Signal processing**

ADVANCED DIGITAL IMAGE PROCESSING

Prerequisites for ADIP



- **Signal Processing**
 - Signals and Systems (S & S)
 - Digital Signal Processing (DSP)
 - Knowledge of “Digital Image Processing” (DIP)

- Digital image processing encompasses processes whose inputs and outputs are images and in addition encompasses processes that extract attributes from images, up to and including the recognition of individual objects

Examples

- Automated analysis of text:
 - Acquiring an image of the area containing text, preprocessing that image, extracting (segmenting) individual characters, describing the characters in a form suitable for computer processing and recognizing those individual characters – **Digital image processing**
 - Making sense of the content of the page- **Image analysis** and **Computer Vision** depending on complexity

Applications of DIP

- Image Sharpening and restoration
- Image Segmentation
- Image Morphology
- Transmission and encoding
- Machine/Robot vision
- Color processing
- Pattern recognition
- Video Processing
- Image/Video Compression
-

Course Description

- This course deals with image compression, morphological image processing, image and texture segmentation, wavelet based image processing and object recognition

Course Objectives:

- To Expose students to advanced concepts of image processing
- Investigate current representations and methods in image processing such as wavelets and morphology
- To design and implement algorithms that perform basic image processing operations like filtering of noise and image enhancement
- To design, analyse and implement algorithms for advanced image analysis like image compression, image reconstruction, image segmentation

ADVANCED DIGITAL IMAGE PROCESSING

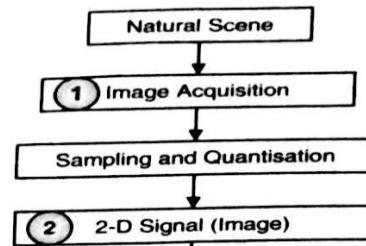
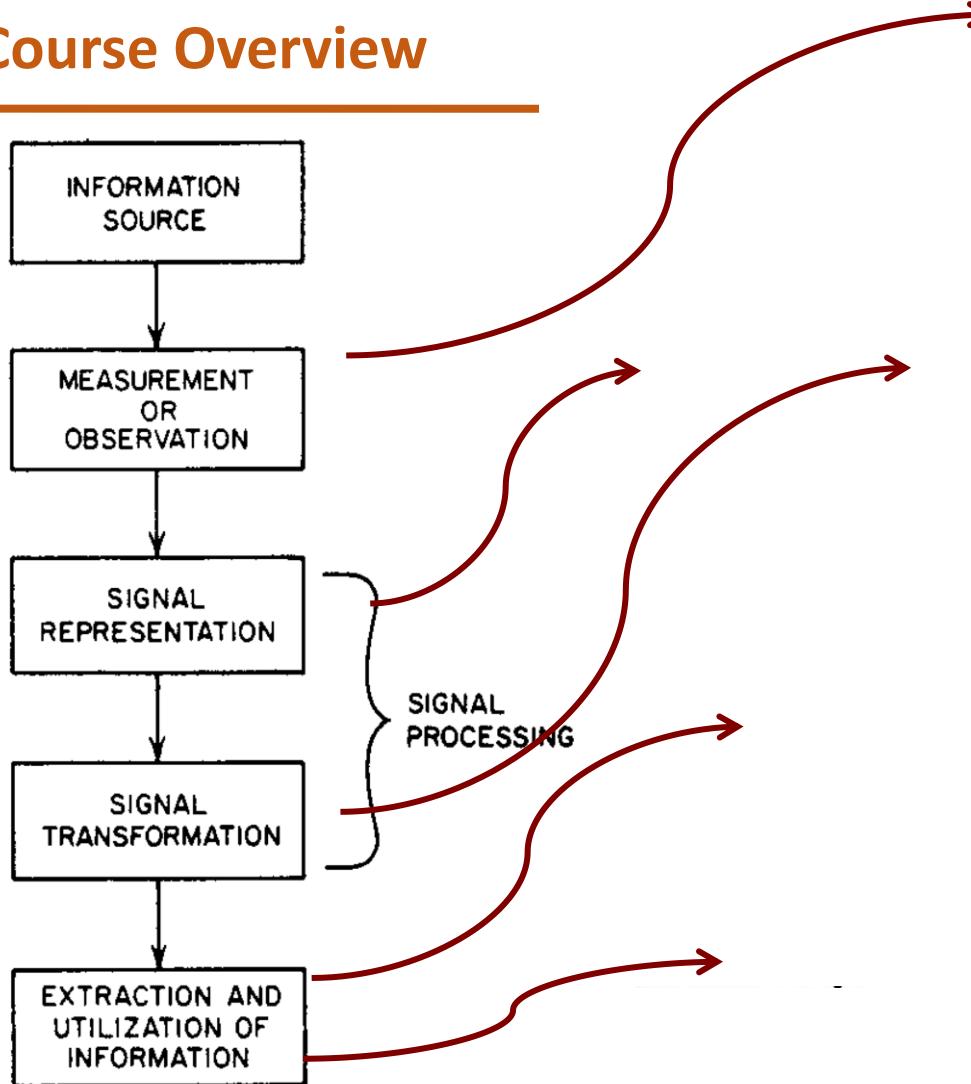
Course Outcomes



- To compare and use different tools for image analysis in transformed domain (wavelet/Fourier transform)
- To apply the notions learnt in the course to practical image processing problems
- To implement techniques for image enhancement, filtering and compression
- To process and analyze image data
- To learn how higher-level image processing concepts such as edge detection, segmentation, representation can be implemented and used

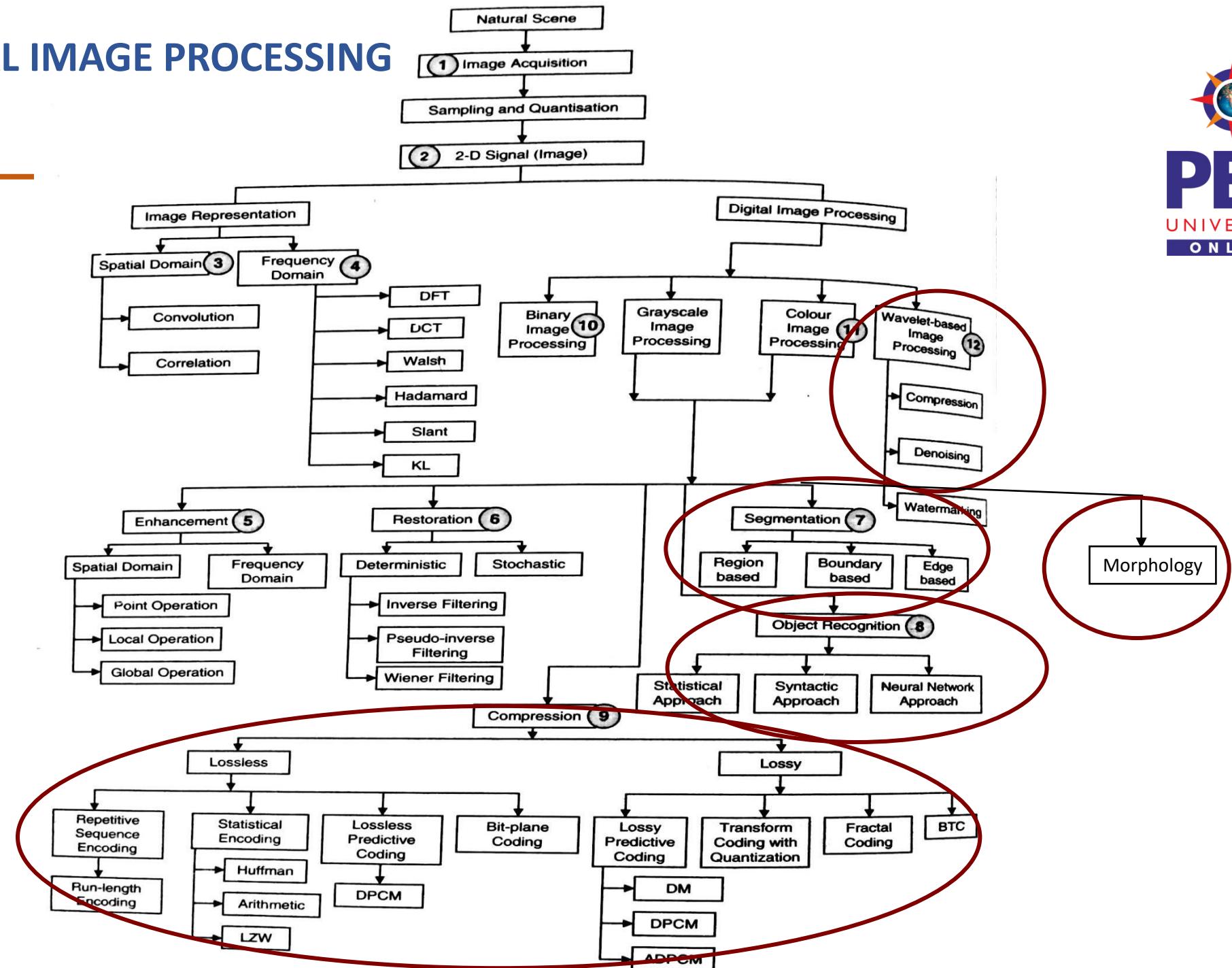
ADVANCED DIGITAL IMAGE PROCESSING

Course Overview



ADVANCED DIGITAL IMAGE PROCESSING

ADIP Overview



ADVANCED DIGITAL IMAGE PROCESSING

Course Overview



Modules:

- **Unit 1: Image Compression**
- **Unit 2: Morphological Image Processing**
- **Unit 3: Image Segmentation**
- **Unit 4: Wavelet based image processing**
- **Unit 5: Object Recognition**

Unit 1: 12 Hrs

Image Compression:

- Need for image compression
- Image Compression models
- Elements of Information Theory
- Error free compression(Lossless Compression)
- Lossy compression
- Image Compression Standards

Unit 2: 12 Hrs

Morphological Image Processing :

- Dilation and Erosion
- Opening and Closing
- The Hit-or-Miss Transformation
- Some Morphological Algorithms
- Extensions to Gray scale Images

Unit 3: 11 Hrs

Image Segmentation:

- Classification of image Segmentation techniques
- Region approach-segmentation
- Clustering techniques
- Segmentation based on thresholding
- Water -shed transformation
- Edge based segmentation
- Classification of edges
- Edge detection and Linking
- Hough Transform
- Shape representation

Unit 4: 10 Hrs

Wavelet based image processing :

- Evolution of wavelet transform
- wavelet transform
- 2D continuous wavelet transform
- Multi resolution analysis
- Examples of wavelets
- Wavelet based image compression

Unit 5: 11 Hrs

Object Recognition :

- Need for an object-recognition System
- Automated Object-recognition Systems
- Patterns and Pattern Class
- Selection of Measurement Parameters
- Relation between Image Processing and Object Recognition
- Approaches to object recognition (Baye's Parametric Classification, Neural Network approach to object recognition, Template-Matching based Object Recognition)
- Applications of Object Recognition

ADVANCED DIGITAL IMAGE PROCESSING

Reference Books



Textbook:

“Digital Image Processing”, R.C.Gonzalez and R.E.Woods, 4th Edition, Pearson, 2018.

Reference Books:

“Digital Image Processing”, S Jayaraman, S Esakkirajan and T

Veerakumar, Mc Graw Hill, 2009.

“Digital Image Compression Techniques”, M. Rabbani and P W Jones,

SPIE Press, 1991.

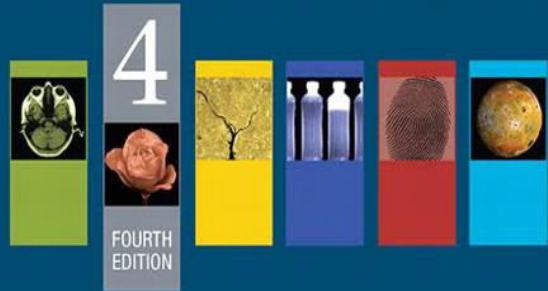
“A Wavelet Tour of Signal Processing”, S Mallat, Second Edition,

Academic Press, 1999.

ADVANCED DIGITAL IMAGE PROCESSING

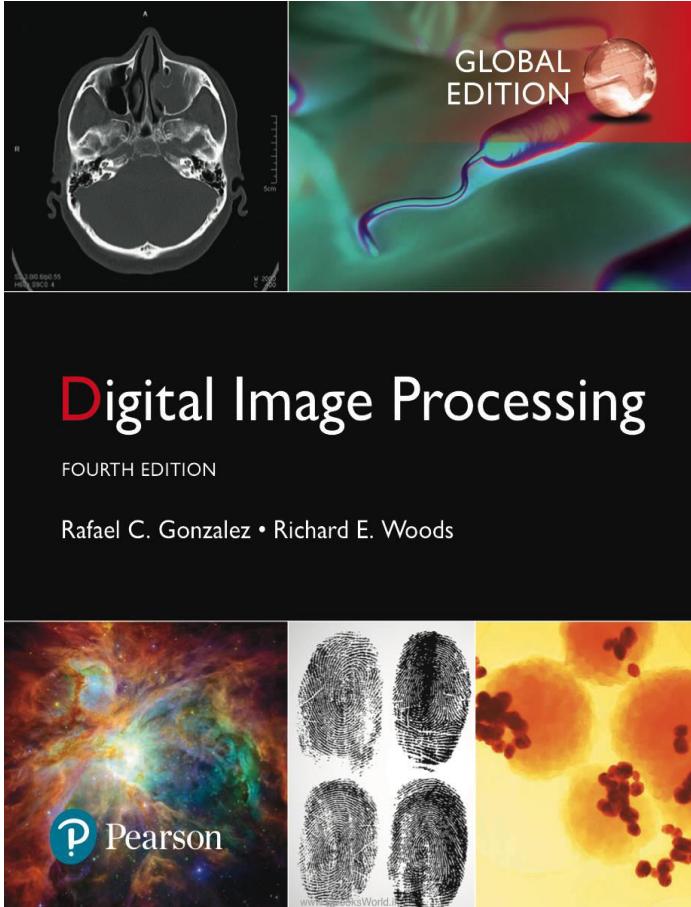
The Text Book

Digital Image Processing



Rafael C. Gonzalez
Richard E. Woods

P Pearson



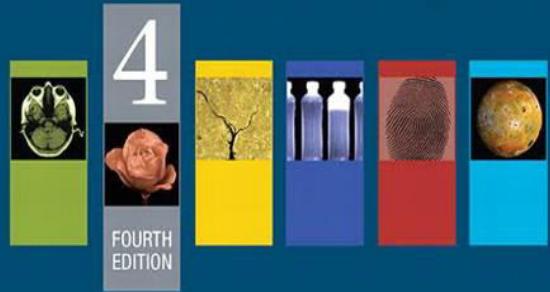
***“When Something can be
read without effort, great
effort has gone into its
writing”***

□ **Enrique Jardiel Poncela**

ADVANCED DIGITAL IMAGE PROCESSING

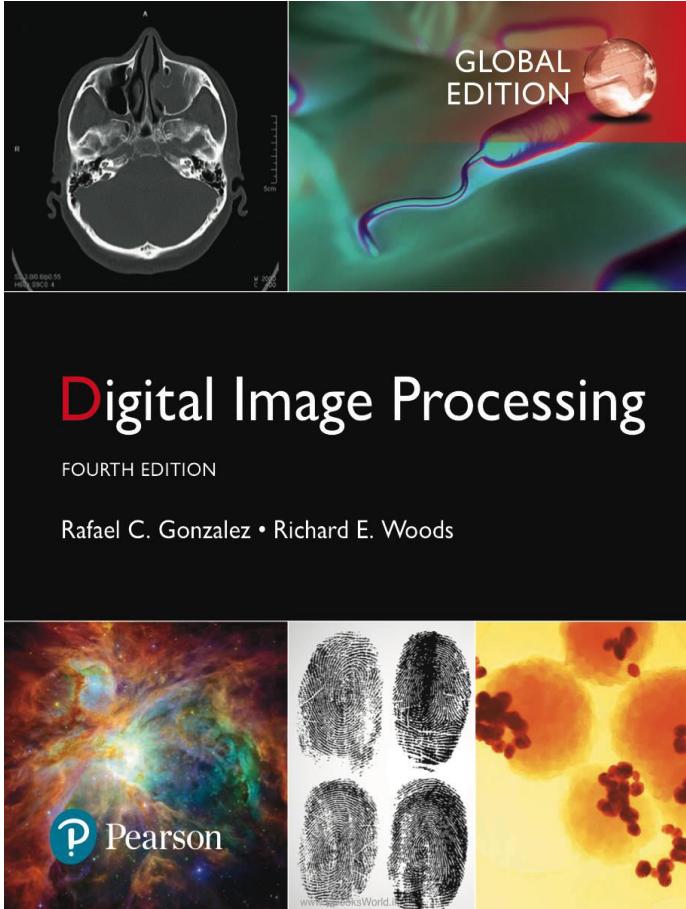
The Course Delivery

Digital Image Processing



Rafael C. Gonzalez
Richard E. Woods

P Pearson



“When Something can be understood without effort, great effort has gone into its teaching”

DIGITAL IMAGE PROCESSING - 2

Course Instructor

Prof. Shruthi M L J

- 13 years of teaching and research experience
- Research Interests:
 - Image /video processing
 - Machine Learning
 - Deep Learning
- Contact:
 - Email: shruthimlj@pes.edu
 - Seating: B-Block, 3rd floor, staff room
- Teaching Assistants: TBA

DIGITAL IMAGE PROCESSING - 2

Course Evaluation Components

ISA-1: 05 Marks

ISA-2: 05 Marks

ISA-3: 05 Marks

ISA-4: 05 Marks

ISA-5: 05 Marks

ISA : 20 Marks (picked from 4 best ISA performances)

Assignments : 10

Projects : 20 Marks

ESA: 50 Marks

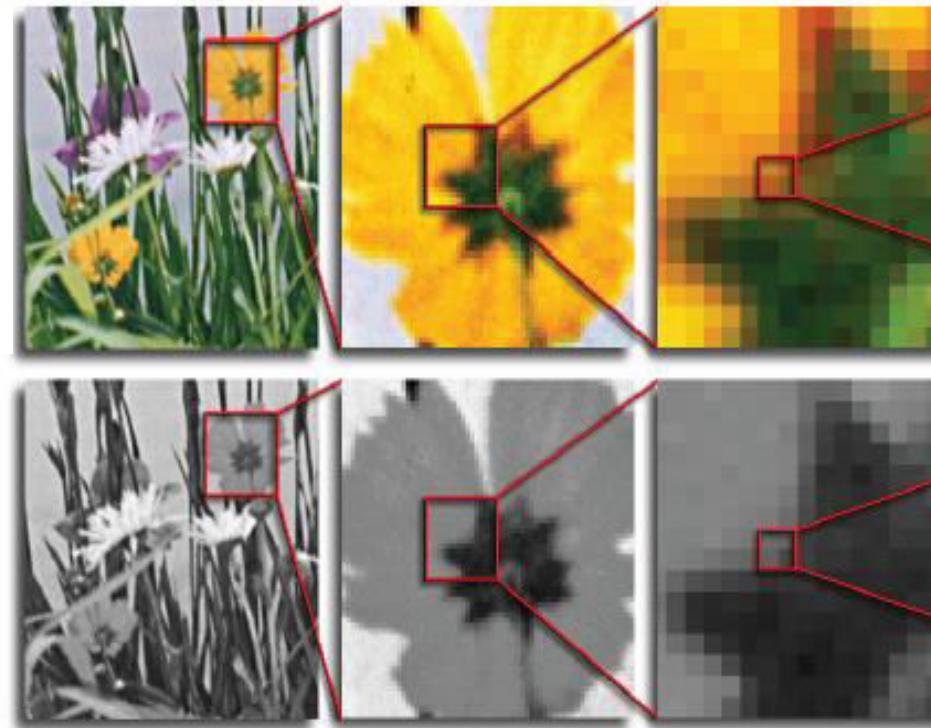
DIGITAL IMAGE PROCESSING - 2

Digital Image Processing: Basics

- **Image:** A two-dimensional function $f(x,y)$ where
 - x and y are spatial coordinates.
 - The amplitude of f is called intensity or gray level at the point (x, y)
- **Digital Image:**
- When x, y and amplitude values of f are all finite, discrete quantities
 - Digital image is composed of a finite number of elements, each of which has a particular location and value
- **Digital Image Processing:** Processing of digital images by means of a digital computer

Digital Image Processing

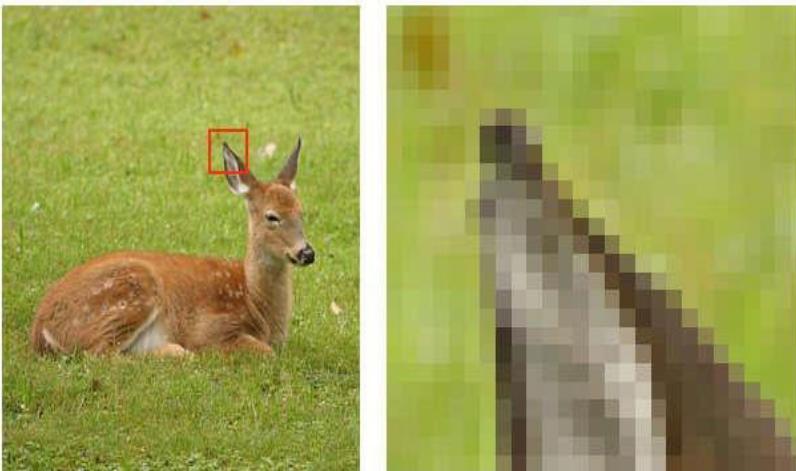
Digital Image



DIGITAL IMAGE PROCESSING - 2

Elements of an Image

- Picture Element
- Pixels
- pels



The smallest square element of a digital image, representing a single color or level of brightness

DIGITAL IMAGE PROCESSING - 2

Image Processing

- Act of converting a captured image from one form to another



DIGITAL IMAGE PROCESSING - 2

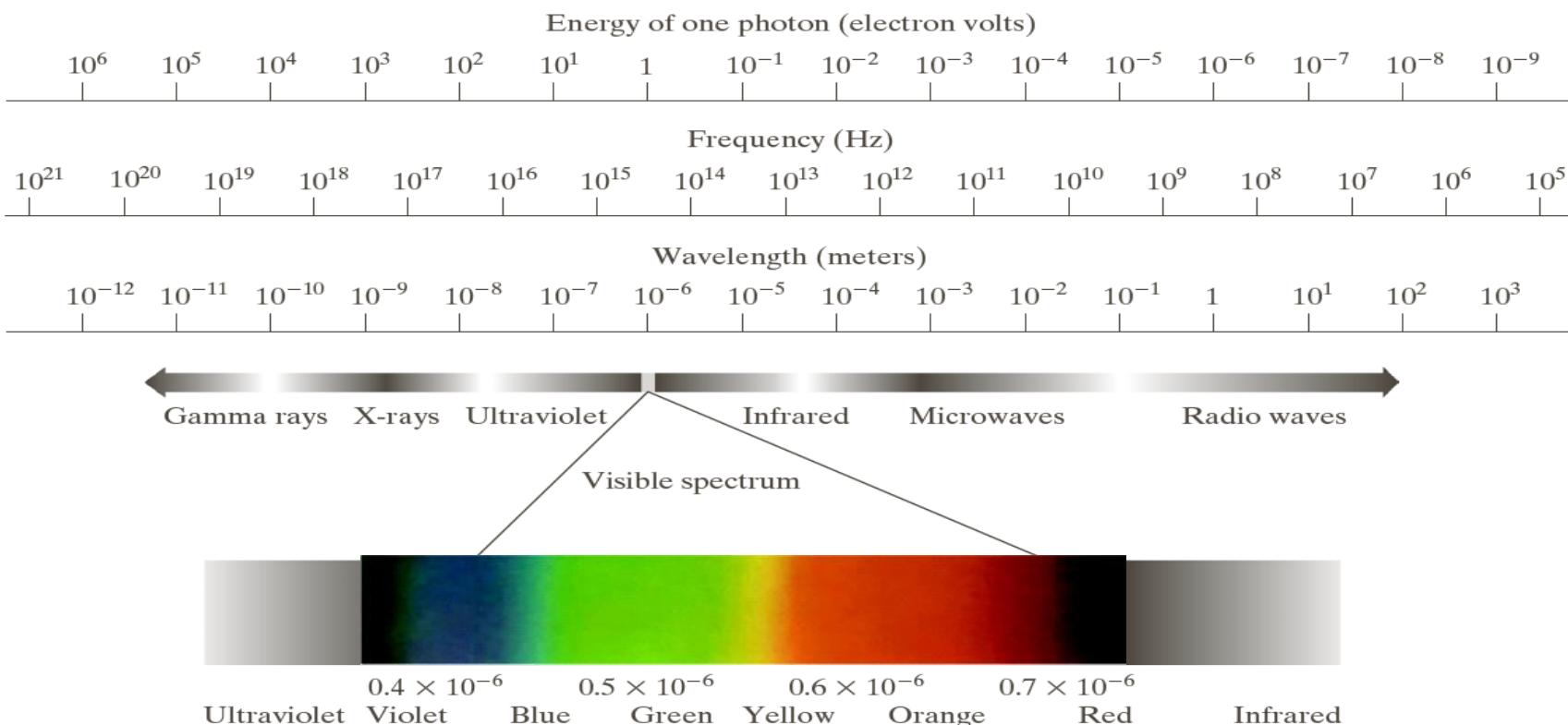
Image Processing / DSP

- DSP: deals with processing of 1-D signals
- IP: deals with visual information that is often in 2 or more dimensions
- Image processing is logical extension of DSP

DIGITAL IMAGE PROCESSING - 2

Computer Vision and Human Visual System

- **Computer Vision:** Machine covers the entire electromagnetic spectrum (gamma to radio waves)



Computer Vision and Human Visual System

- **Human Visual System:** Limited to visual band of EM spectrum
- Imaging machines can operate on images generated by sources that humans are not accustomed to associating with images: ultrasound, electron microscopy and computer generated images

“Unlike humans, who are limited to the visual band of the electromagnetic spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves”

DIGITAL IMAGE PROCESSING - 2

Image Processing and Computer Graphics

- Very closely related
- Image Processing deals with raster data or bitmaps whereas computer graphics primarily deals with vector data
 - Raster data(bitmap is stored in 2-D matrix form often used to depict real images)
 - Vector images are composed of vectors, which represent mathematical relationships between objects. Vectors are lines or primitive curves used to describe an image

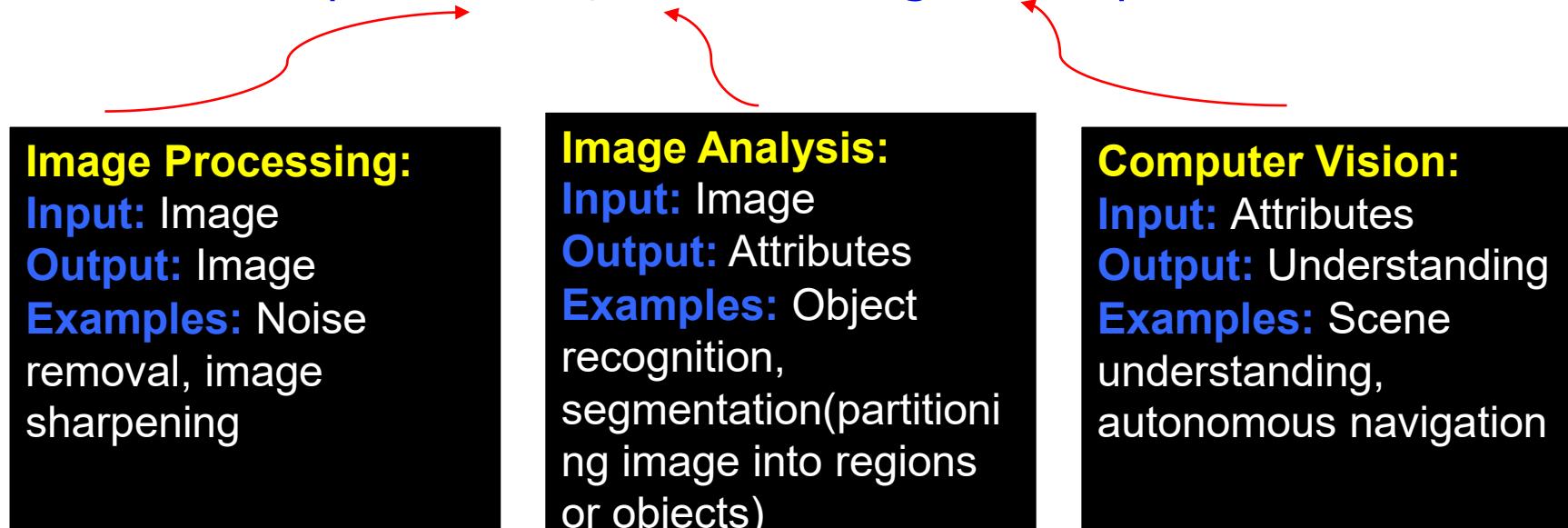
Image Processing and Computer Graphics

- **Computer Graphics:** The algorithms often take numerical data as input and give image as output
- **Image Processing:** Input is often an image
 - **Goal:** To enhance the quality of the image to assist in interpreting it
 - **Result of image processing is often an image or description of an image**
- **Image processing is logical extension of computer graphics**

DIGITAL IMAGE PROCESSING - 2

Image Processing and Computer Graphics

The continuum from image processing to computer vision can be broken up into low-, mid- and high-level processes



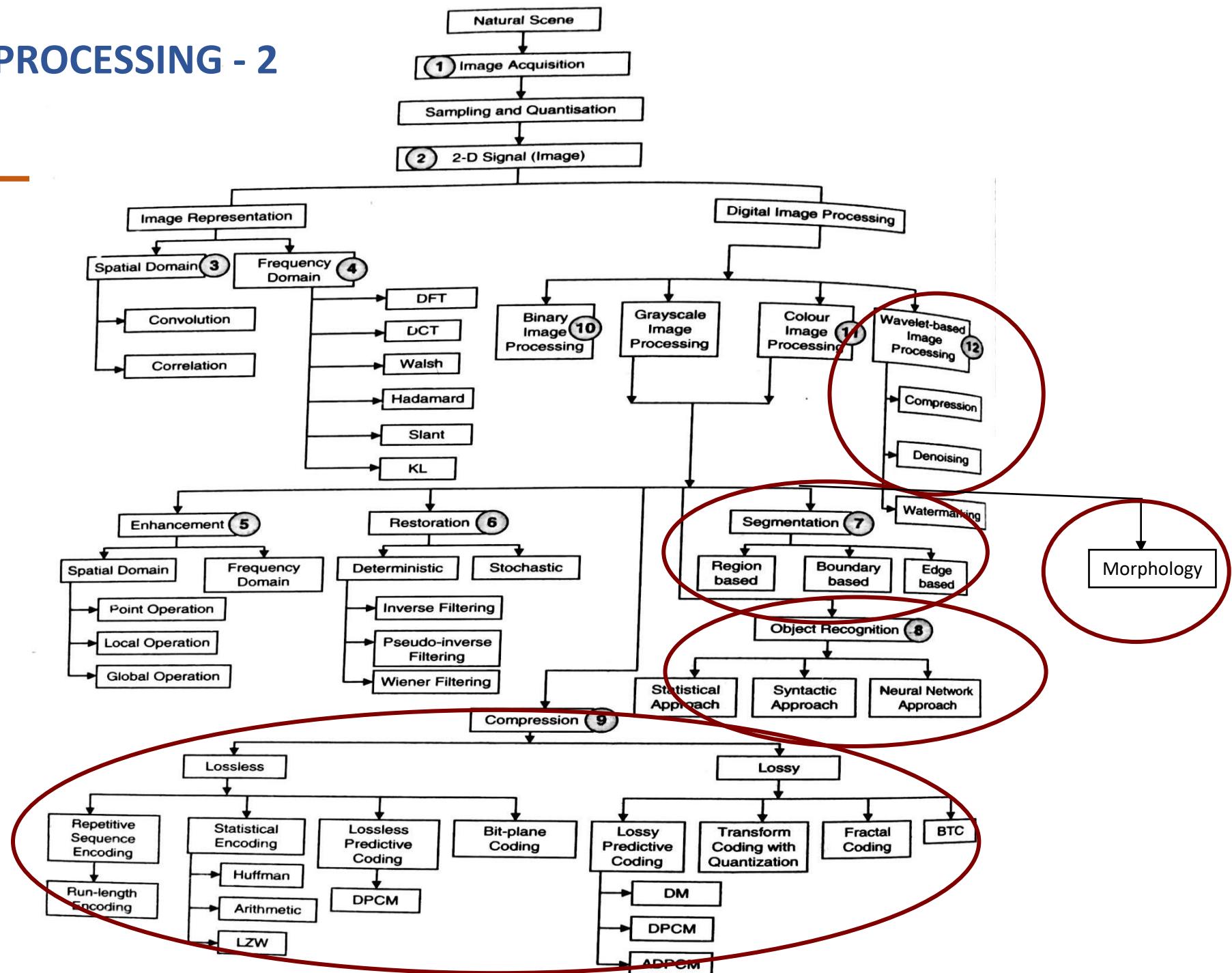
Input and output are images

Input is image & outputs are attributes extracted form image(edge,contour etc..)

Involves “making sense “ Like cognitive functions associated with human vision

DIGITAL IMAGE PROCESSING - 2

ADIP Overview



DIGITAL IMAGE PROCESSING - 2

Image Compression

- It is the art and science of reducing the data required to represent an image
- Innumerable number of multimedia data is compressed and decompressed daily (MP4,JPEG,MPEG etc...)
- **One of the most useful and successful technologies in digital image processing**

DIGITAL IMAGE PROCESSING - 2

Image Compression

- It is the art and science of reducing the data required to represent an image
- Innumerable number of multimedia data is compressed and decompressed daily (MP4,JPEG,MPEG etc...)
- **One of the most useful and successful technologies in digital image processing**

Fundamentals of image compression

Data Compression:

- The process of reducing the amount of *data* required to represent a given quantity of *information* Here, *data* and *information* are not the same
 - Data is 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*.



Fundamentals of image compression

Compression Ratio:

compression ratio is defined as $C = \frac{b}{b'}$

Where b is the number of bits needed to represent an image as a 2-D array of intensity values and b' denotes the number of bits needed in compressed representation of the same information

- *Relative data redundancy*, R , of the representation with b bits is

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

where C , commonly called the *compression ratio*

If $C=10$ then redundancy in representation is

DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

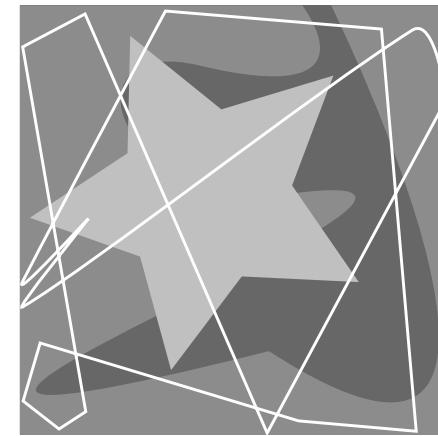
- 2-D intensity arrays suffer from three principal types of data redundancies that can be identified and exploited
 - Coding Redundancy
 - Spatial and temporal redundancy
 - Irrelevant redundancy

DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

- ***Coding redundancy:***

- A code is a system of symbols (letters, numbers, bits, and the like) used to represent a body of information or set of events.
- Each piece of information or event is assigned a sequence of *code symbols*, called a *code word*.
- The number of symbols in each code word is its *length*.
- The 8-bit codes that are used to represent the intensities in most 2-D intensity arrays contain more bits than are needed to represent the intensities.



Computer generated
($256 \times 256 \times 8$)
bit image

DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

- *Spatial and temporal redundancy :*
 - Because the pixels of most 2-D intensity arrays are correlated spatially (i.e., each pixel is similar to or dependent upon neighboring pixels), information is unnecessarily replicated in the representations of the correlated pixels.
 - In a video sequence, temporally correlated pixels (i.e., those similar to or dependent upon pixels in nearby frames) also duplicate information.

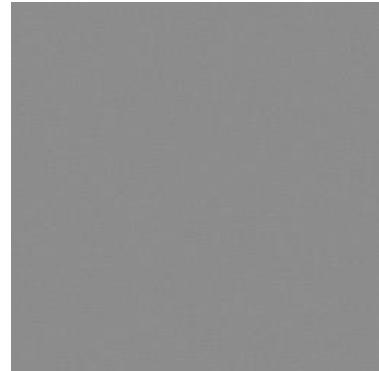


DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

- *Irrelevant information:*

- Most 2-D intensity arrays contain information that is ignored by the human visual system (HVS) and/or extraneous to the intended use of the image.
- It is redundant in the sense that it is not used.



Compression is achieved when one or more redundancy is reduced or eliminated.

Types of Redundancy

Coding Redundancy:

- **Objective:** To find optimal information coding
 - **Assumption:** the intensity values of an image are random quantities
 - Assume that a discrete random variable r_k in the interval $[0, L - 1]$ is used to represent the intensities of an $M \times N$ image, and that each r_k occurs with probability $p_r(r_k)$
- Hence

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

where L is the number of intensity values, and n_k is the number of times that the k_{th} intensity appears in the image

DIGITAL IMAGE PROCESSING - 2

Coding Redundancy Cont..

- If the number of bits used to represent each value of r_k is $l(r_k)$, then the average number of bits required to represent each pixel is

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

- That is, the average length of the code words assigned to the various intensity values is found by summing the products of the number of bits used to represent each intensity and the probability that the intensity occurs.
- Hence the total number of bits required to represent an $M \times N$ image is

Coding Redundancy Cont..

- So with

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

- If the intensities are represented using a *natural m-bit fixed-length code*, the right-hand side of the equation reduces to
- m bits. That is, $L_{\text{avg}} = m$ when m is substituted for $l(r_k)$
 - The constant m can be taken outside the summation, leaving only the sum of the $p_r(r_k)$ for $0 \leq k \leq L-1$ which equals 1

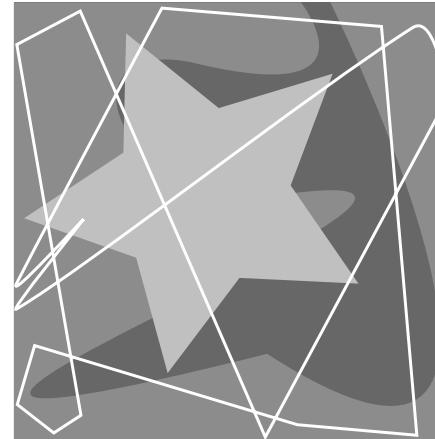
DIGITAL IMAGE PROCESSING - 2

Coding Redundancy Cont..

Coding Redundancy Example:

- Consider the computer generated image
- Consider its intensity distribution as shown in table below

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



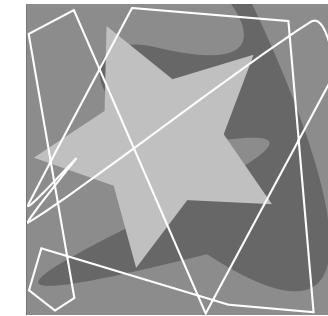
Coding Redundancy Example

Code 1:

- If a natural 8-bit binary code is used to represent its four possible intensities

r_k	$p_r(r_k)$	Code 1
$r_{87} = 87$	0.25	01010111
$r_{128} = 128$	0.47	01010111
$r_{186} = 186$	0.25	01010111
$r_{255} = 255$	0.03	01010111
r_k for $k \neq 87, 128, 186, 255$	0	—

- L_{avg} (the average number of bits for code 1) is 8 bits
(because $I_1(r_k) = 8$ bits for all r_k)

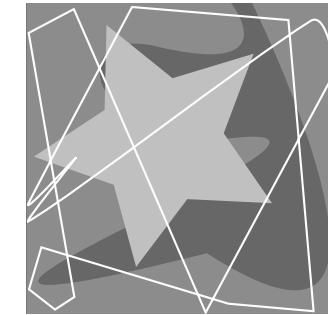


DIGITAL IMAGE PROCESSING - 2

Coding Redundancy Example

Code 2:

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	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0



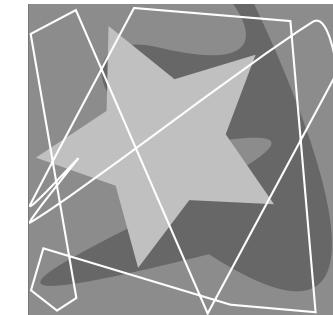
- The average length of the encoded pixels is $L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$
 $L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81$ bits
- The total number of bits needed to represent the entire image is

DIGITAL IMAGE PROCESSING - 2

Coding Redundancy Example

Code 2:

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	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0



- The resulting compression is $C = \frac{256 \times 256 \times 8}{118,621} = \frac{8}{1.81} \approx 4.42$
- And corresponding relative redundancy $R = 1 - \frac{1}{4.42} = 0.774$
- Thus, 77.4% of the data in the original 8-bit 2-D intensity array is redundant

Coding Redundancy Cont...

Analysis:

- The compression achieved by code 2 results from assigning fewer bits to the more probable intensity values than to the less probable ones.
- In the resulting *variable-length code*, r_{128} (the image's most probable intensity) is assigned the 1-bit code word 1 [of length $I_2(128) = 1$], while r_{255} (its least probable occurring intensity) is assigned the 3-bit code word 001 [of length $I_2(255) = 3$]
- The best *fixed-length code* that can be assigned to the intensities of the image is the natural 2-bit counting sequence {00,01,10,11}, but the resulting compression is only 8/2 or 4:1 which is about 10% less than the 4.42:1 compression of the variable-length code

Coding Redundancy Cont...

Analysis:

- *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
- Coding redundancy is almost always present when the intensities of an image are represented using a natural binary code
- The reason is that most images are composed of objects that have a regular and somewhat predictable morphology (shape) and reflectance, and are sampled so the objects being depicted are much larger than the picture elements
- The natural consequence is that for most images, certain intensities are more probable than others (that is, the histograms of most images are not uniform). A natural binary encoding assigns the same number of bits to both the most and least probable values, failing to minimize L_{avg} , and resulting in coding redundancy

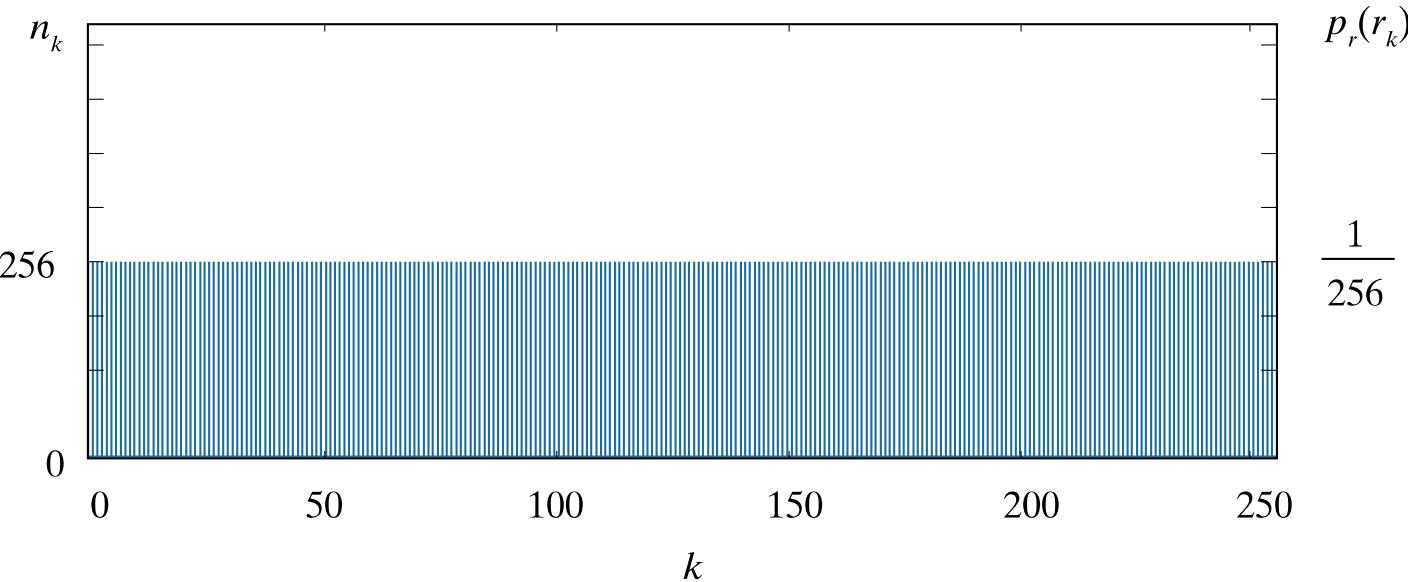
DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

- Coding Redundancy Cont.
- Spatial and temporal redundancy
- Irrelevant redundancy

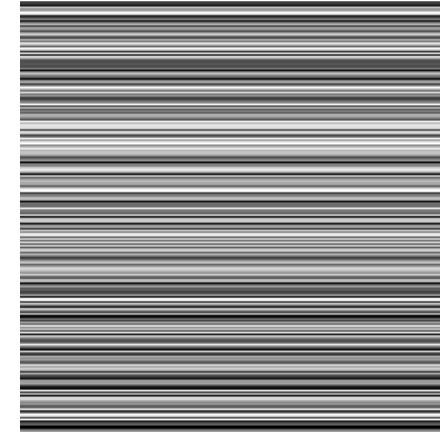
Spatial and Temporal Redundancy

- Consider the computer-generated collection of constant intensity lines
1. All 256 intensities are equally probable. The histogram of the image is uniform.



Spatial and Temporal Redundancy

- The image in Figure (when represented as a conventional 8-bit intensity array) cannot be compressed by variable-length coding alone.
- Unlike the images whose histogram is *not* uniform, a fixed-length 8-bit code in this case minimizes

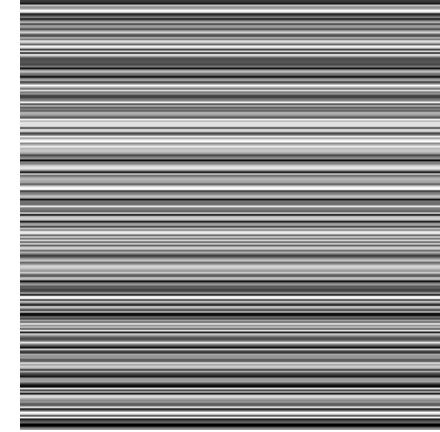


$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

DIGITAL IMAGE PROCESSING - 2

Spatial and Temporal Redundancy

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 (completely dependent on one another) in the horizontal direction.



Observation: This indicates a significant spatial redundancy that can be eliminated by representing the image as a sequence of ***run-length pairs***

- where each **run-length pair** specifies the start of a new intensity and the number of consecutive pixels that have that intensity.

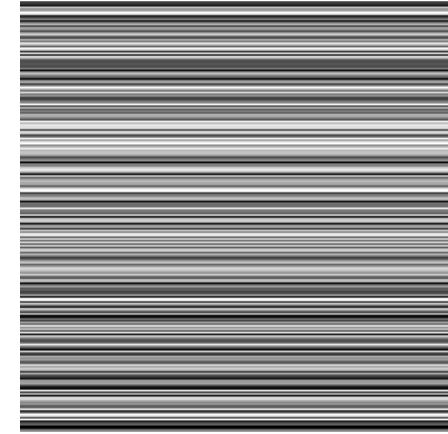
DIGITAL IMAGE PROCESSING - 2

Spatial and Temporal Redundancy

- The compression ratio using a run-length based representation of the original 2-D, 8-bit intensity array would be

$$(256 * 256 * 8) / [(256 + 256) * 8] \text{ or } 128:1.$$

- 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.



Spatial and Temporal Redundancy

- In most images, pixels are correlated spatially (in both x and y) and in time (when the image is part of a video sequence).
- Because most pixel intensities can be predicted reasonably well from neighboring intensities, 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 spatially and temporally correlated pixels, a 2-D intensity array must be transformed into a more efficient but usually “non-visual” representation.

Spatial and Temporal Redundancy

- For example, run-lengths or the differences between adjacent pixels can be used.
- Transformations of this type are called *mappings*.
- A mapping is said to be *reversible* if the pixels of the original 2-D intensity array can be reconstructed without error from the transformed data set; otherwise, the mapping is said to be *irreversible*.

DIGITAL IMAGE PROCESSING - 2

Types of Redundancy

- Coding Redundancy
- Spatial and temporal redundancy
- Irrelevant information

Irrelevant Information

- One of the simplest ways to compress a set of data is to remove superfluous data from the set.
- In the context of digital image compression, information that is ignored by the human visual system, or is extraneous to the intended use of an image, are obvious candidates for omission
- Thus, the computer-generated image in Figure, can be represented by its average intensity alone—a single 8-bit value.
 - because it appears to be a homogeneous field of gray
 - The original $256 * 256 * 8$ bit intensity array is reduced to a single byte



Irrelevant Redundancy

- The resulting compression is

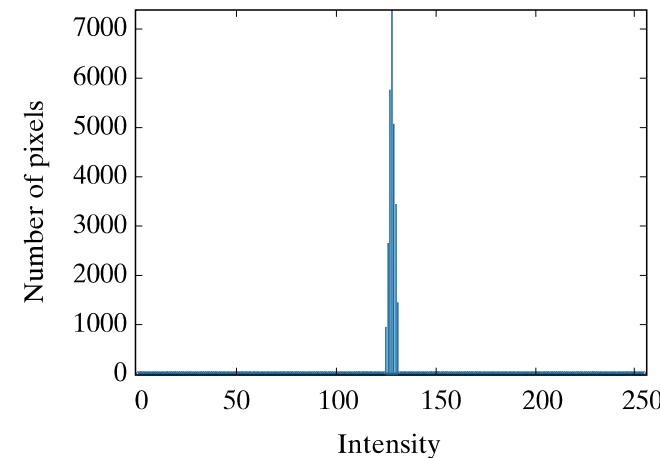
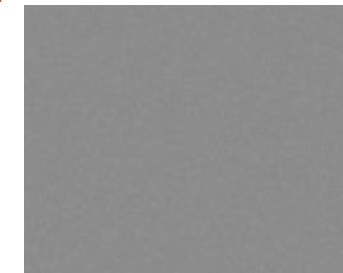
$$(256 * 256 * 8) / 8 \text{ or } 65,536:1.$$

- Of course, the original 256 * 256 * 8 bit image must be recreated to view and/or analyze it, but there would be little or no perceived decrease in reconstructed image quality.

DIGITAL IMAGE PROCESSING - 2

Irrelevant Redundancy

- In the histogram of the image, there are several intensity values (125 through 131) actually present.
- The human visual system (HVS) averages these intensities, perceives only the average value, then ignores the small changes in intensity that are present in this case



Irrelevant Information

- The redundancy examined here is fundamentally different from the first two types of redundancies
- Its elimination is possible because the information itself is not essential for normal visual processing and/or the intended use of the image.
- Because its omission results in a loss of quantitative information, its removal is commonly referred to as *quantization*.
- This terminology is consistent with normal use of the word, which generally means the mapping of a broad range of input values to a limited number of output values. Because information is lost, quantization is an irreversible operation

DIGITAL IMAGE PROCESSING - 2

Measuring Image Information

- **Image Compression:** Goal is to reduce the size of image to its minimum / optimum size **without losing information**
- What is the minimum size actually needed to represent the information in an image?
- Is there a minimum amount of data that is sufficient to describe an image **without losing information?**
- *Information theory* provides the mathematical framework to answer this and related questions.
 - Its fundamental premise is that the generation of information can be modeled as a probabilistic process which can be measured in a manner that agrees with intuition.

Measuring Image Information

- In accordance with this supposition, a random event E with probability $P(E)$ is said to contain $I(E)$ units of information where

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

- If $P(E) = 1$ (that is, the event always occurs), $I(E) = 0$ and no information is attributed to it.
- Because no uncertainty is associated with the event, no information would be transferred by communicating that the event has occurred [it *always* occurs if $P(E) = 1$]
- The base of the logarithm determines the unit used to measure information. If the base m logarithm is used, the measurement is said to be in m -ary units.
- If the base 2 is selected, the unit of information is the *bit*.

DIGITAL IMAGE PROCESSING - 2

Measuring Image Information

- if $P(E) = 1/2$, $I(E) = - \log_2(1/2)$ or 1 bit.
- That is, 1 bit is the amount of information conveyed when one of two possible equally likely events occurs.
- Given a source of statistically independent random events from a discrete set of possible events $\{a_1, a_2, \dots, a_J\}$ with associated probabilities $\{P(a_1), P(a_2), \dots, P(a_J)\}$, the average information per source output, called the *entropy* of the source, is

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

- The a_j in this equation are called *source symbols*. Because they are statistically independent, the source itself is called a *zero-memory source/memoryless source*.

DIGITAL IMAGE PROCESSING - 2

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

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

where variables L , r_k , and $p_r(r)$ are as defined earlier

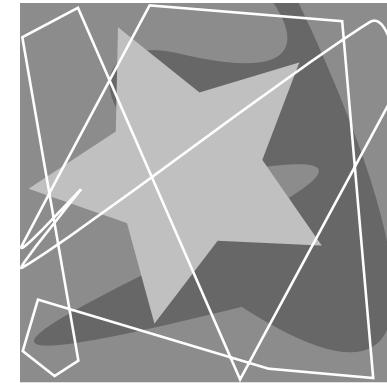
- Because the base 2 logarithm is used, \tilde{H} is the average information per intensity output of the imaginary intensity source in bits.
- It is not possible to code the *intensity values* of the imaginary source (and thus the sample image) with fewer than \tilde{H} bits/ pixel.

DIGITAL IMAGE PROCESSING

Measuring Image Information

- Example:
- Determine the entropy of the image given the probability as in table

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 $k = 87, 128, 186, 255$	0



- Using

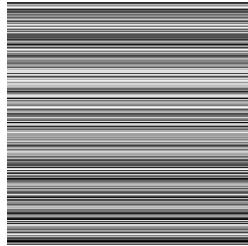
$$\tilde{H} = -\sum_{k=0}^{L-1} p_r(r_k) \log_2 p_r(r_k)$$

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

DIGITAL IMAGE PROCESSING - 2

Measuring Image Information

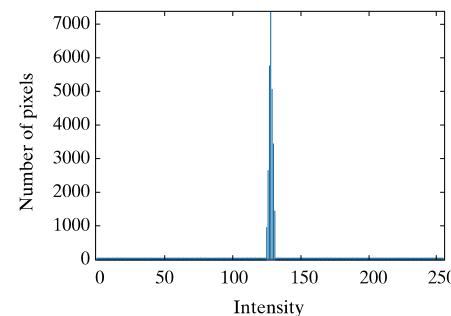
- Similarly entropy of



$$I_s = -(1/256 \log_2 256) \times 256 = 8 \text{ bpp}$$

- And entropy of

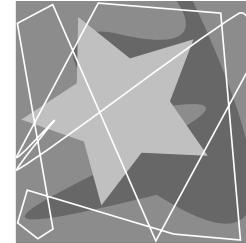
$$I_s = 1.566 \text{ bpp}$$



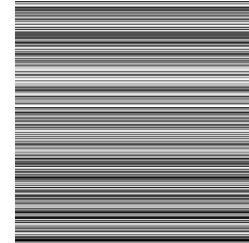
DIGITAL IMAGE PROCESSING

Analysis

- Note that the first image appears to have the most visual information, but has almost the lowest computed entropy— 1.66 bits/pixel.
- The second image has almost five times the entropy of the first image, but appears to have about the same (or less) visual information.
- The third image, which seems to have little or no information, has almost the same entropy as the first image.
- **The obvious conclusion is that the amount of entropy, and thus information in an image, is far from intuitive.**



$H=1.6614$ bpp



$H=8$ bpp



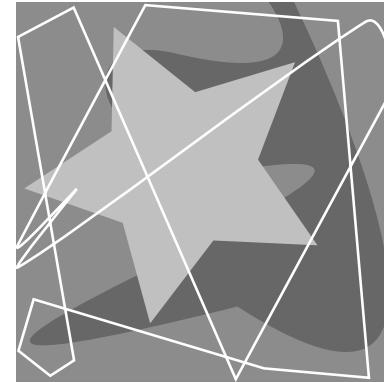
$H=1.566$ bpp

DIGITAL IMAGE PROCESSING - 2

Shannon's First Theorem

Consider the image with VLC

r_k	$p_r(r_k)$	Code 2	$I_2(r_k)$
$r_{87} = 87$	0.25	01	2
$r_{128} = 128$	0.47	1	1
$r_{186} = 186$	0.25	000	3
$r_{255} = 255$	0.03	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	0



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

Entropy, $H=1.6614$ bpp

Shannon's First Theorem

- The variable-length code in the example was able to represent the intensities of the image 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 Figure can be represented with as few as 1.6614 bits/pixel.

Shannon's First Theorem

- Shannon looked at representing groups of consecutive source symbols with a single code word (rather than one code word per source symbol), and showed that

$$\lim_{n \rightarrow \infty} \left[\frac{L_{\text{avg},n}}{n} \right] = H$$

where $L_{\text{avg},n}$ is the average number of code symbols required to represent all n -symbol groups

- This equation tells us that $L_{\text{avg},n}/n$ can be made arbitrarily close to H by encoding infinitely long extensions of the single-symbol source.
- That is, it is possible to represent the output of a zero-memory source with an average of H information units per source symbol.

Measuring Image Information

- A random event E with probability $P(E)$ is said to contain $I(E)$ units of information where

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

- Average Information, Entropy of the source is given by

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

- The amount of entropy, and thus information in an image, is far from intuitive

Shannon's First Theorem

- Shannon looked at representing groups of consecutive source symbols with a single code word (rather than one code word per source symbol), and showed that

$$\lim_{n \rightarrow \infty} \left[\frac{L_{\text{avg},n}}{n} \right] = H$$

where $L_{\text{avg},n}$ is the average number of code symbols required to represent all *n*-symbol groups

- This equation tells us that $L_{\text{avg},n}/n$ can be made arbitrarily close to H by encoding infinitely long extensions of the single-symbol source.
- That is, it is possible to represent the output of a zero-memory source with an average of H information units per source symbol.

DIGITAL IMAGE PROCESSING - 2

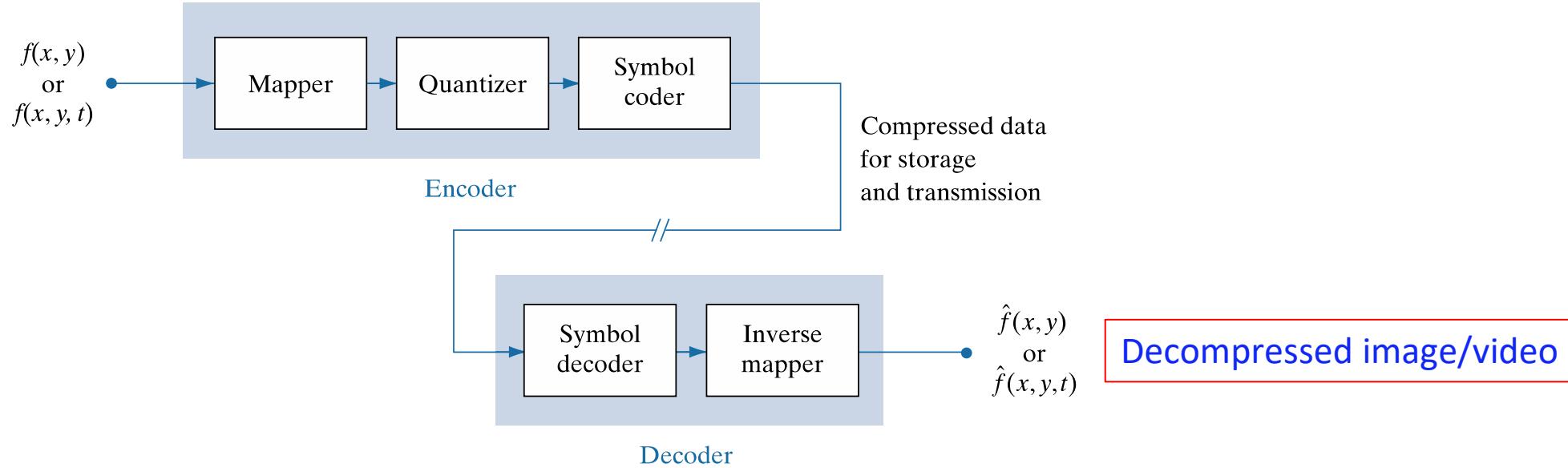
Image Compression Models

- An image compression system is composed of two distinct functional components: an *encoder* and a *decoder*
- The *encoder* performs *compression*, and the decoder performs the complementary operation of *decompression*
- Both operations can be performed in software, as is the case in Web browsers and many commercial image-editing applications, or in a combination of hardware and firmware, as in commercial DVD players
- A *codec* is a device or program that is capable of *both encoding and decoding*

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Functional block diagram of a general image compression system



- Input image $f(x, \dots)$ is fed into the encoder, which creates a compressed representation of the input
- This representation is stored for later use, or transmitted for storage and use at a remote location
- When the compressed representation is presented to its complementary decoder, a reconstructed output image $\hat{f}(x, \dots)$ is generated

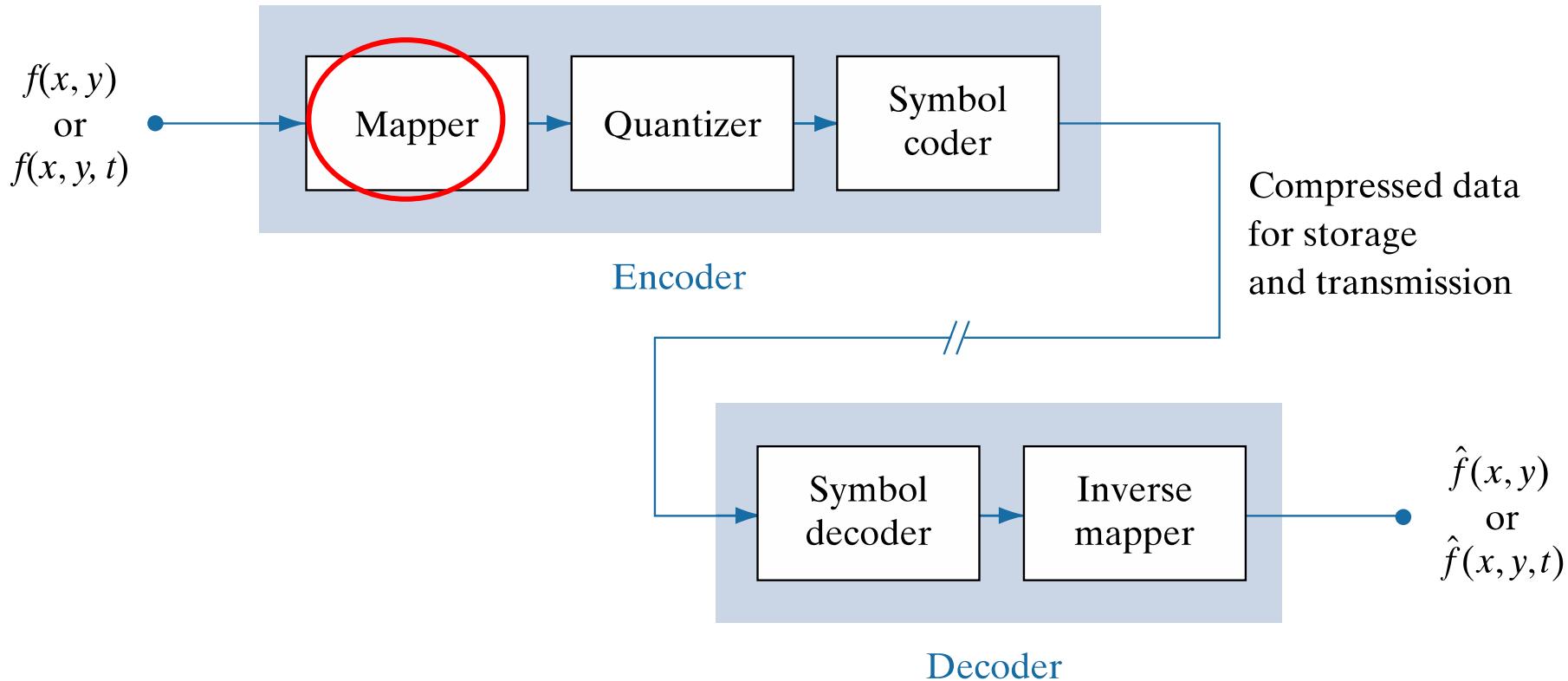
Image Compression Models

- In still-image applications, the encoded input and decoder output are $f(x, y)$ and $\hat{f}(x, y)$, respectively
- In video applications, they are $f(x, y, t)$ and $\hat{f}(x, y, t)$, where the discrete parameter t specifies time
- In general, $f(x, \dots)$ may or may not be an exact replica of $\hat{f}(x, \dots)$
- If it is, the compression system is called *error free, lossless*, or *information preserving*
- If not, the reconstructed output image is distorted, and the compression system is referred to as *lossy*

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Functional block diagram of a general image compression system.



DIGITAL IMAGE PROCESSING - 2

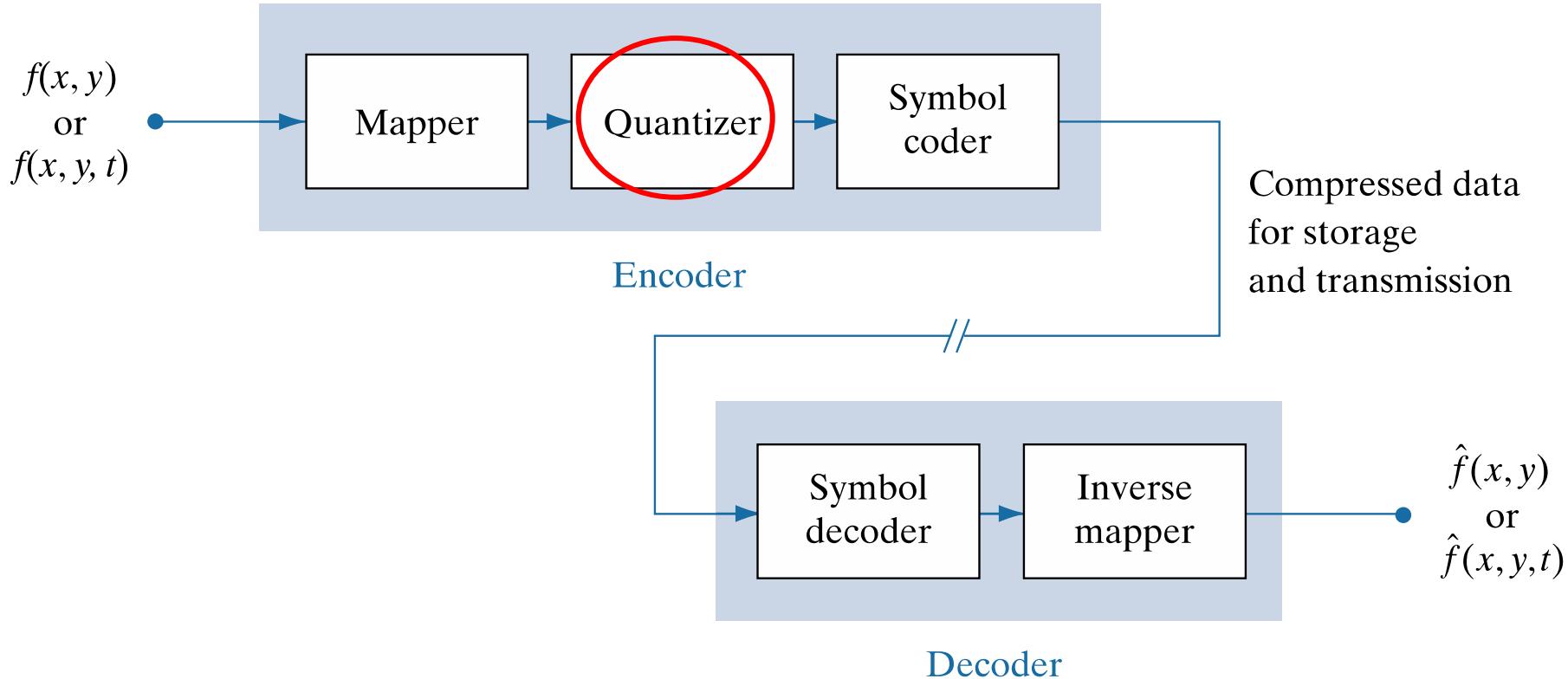
Encoding or Compression Process

- In the first stage of the encoding process, a *mapper* transforms $f(x, \dots)$ into a (usually nonvisual) format designed to reduce spatial and temporal redundancy
- This operation generally is reversible, and may or may not directly reduce the amount of data required to represent the image.
- Run-length coding is an example of a mapping that normally yields compression in the first step of the encoding process.
- The mapping of an image into a set of less correlated transform coefficients is an example of the opposite case (the coefficients must be further processed to achieve compression).
- In video applications, the mapper uses previous (and, in some cases, future) video frames to facilitate the removal of temporal redundancy.

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Functional block diagram of a general image compression system.



DIGITAL IMAGE PROCESSING - 2

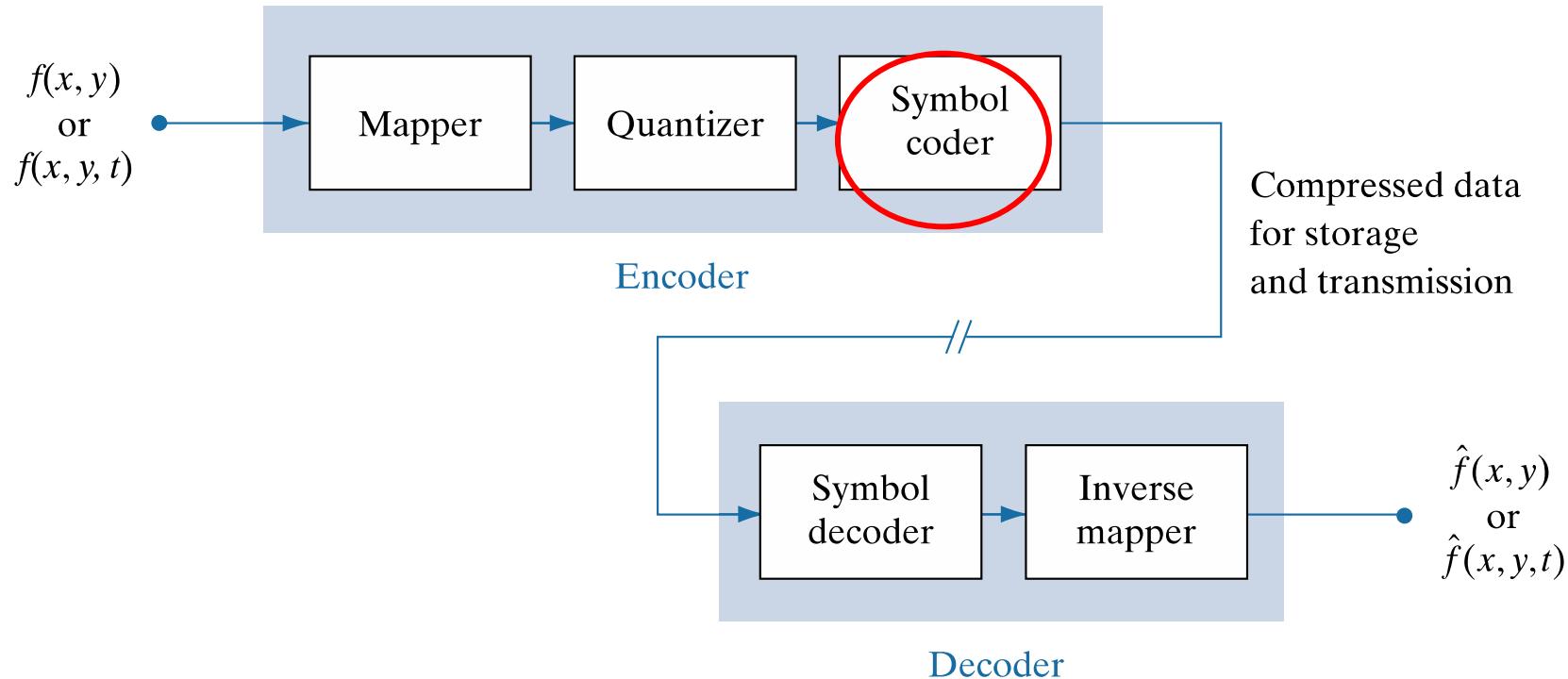
Encoding or Compression Process

- The *quantizer* in the figure 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.
- As noted earlier, this operation is irreversible. It must be omitted when error-free compression is desired.
- In video applications, the *bit rate* (bpp for image) of the encoded output is often measured (in bits/second), and is used to adjust the operation of the quantizer so a predetermined average output rate is maintained.
- Thus, the visual quality of the output can vary from frame to frame as a function of image content.

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Functional block diagram of a general image compression system.



DIGITAL IMAGE PROCESSING - 2

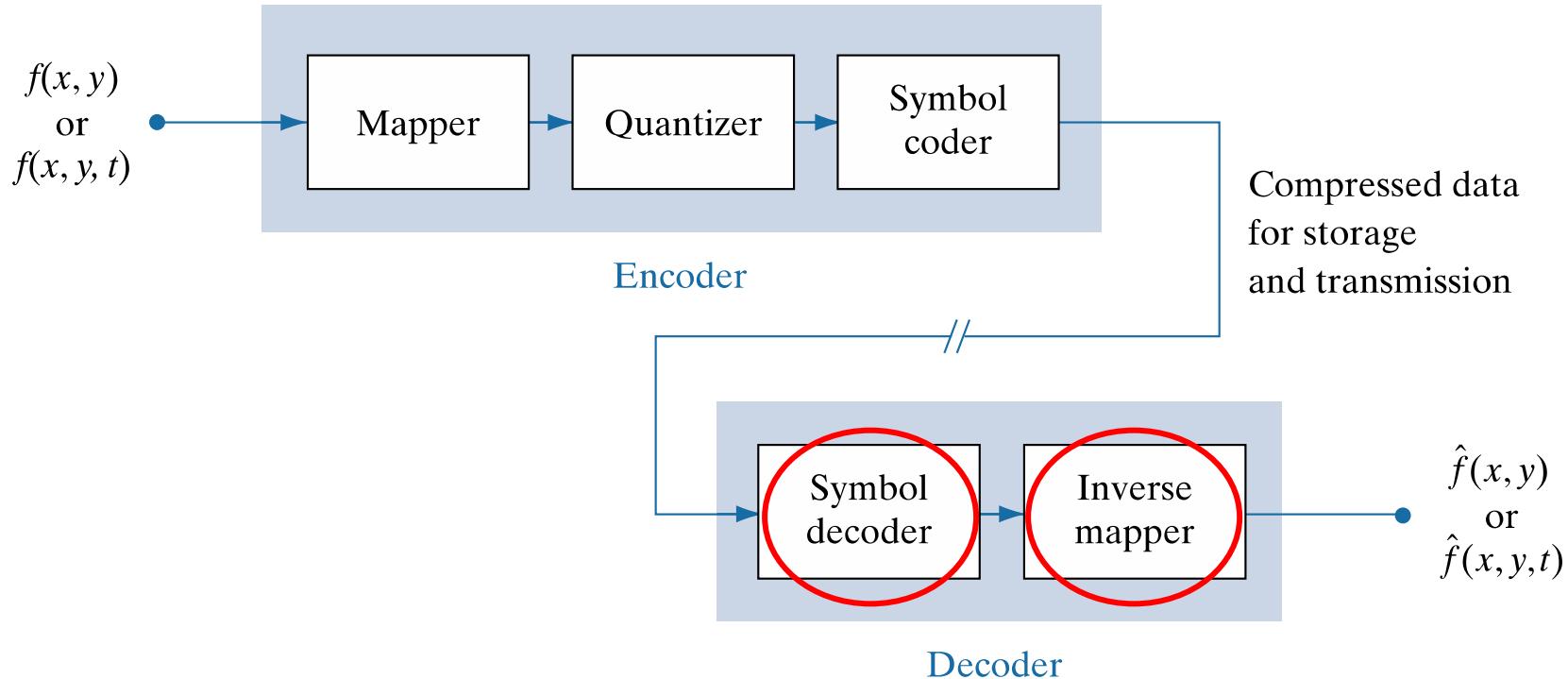
Encoding or Compression Process

- In the third and final stage of the encoding process, the *symbol coder* generates a fixed-length or variable-length code to represent the quantizer output, and maps the output in accordance with the code.
- In many cases a variable-length code is used.
 - The shortest code words are assigned to the most frequently occurring quantizer output values, thus minimizing coding redundancy. This operation is reversible.
 - Upon its completion, the input image has been processed for the removal of each of the three redundancies described earlier.

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Functional block diagram of a general image compression system.



DIGITAL IMAGE PROCESSING - 2

Decoding or Decompression Process

- The decoder contains only two components: a *symbol decoder* and an *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.
- In video applications, decoded output frames are maintained in an internal frame store (not shown) and used to reinsert the temporal redundancy that was removed at the encoder.

DIGITAL IMAGE PROCESSING - 2

Image format, Container and Compression Standards

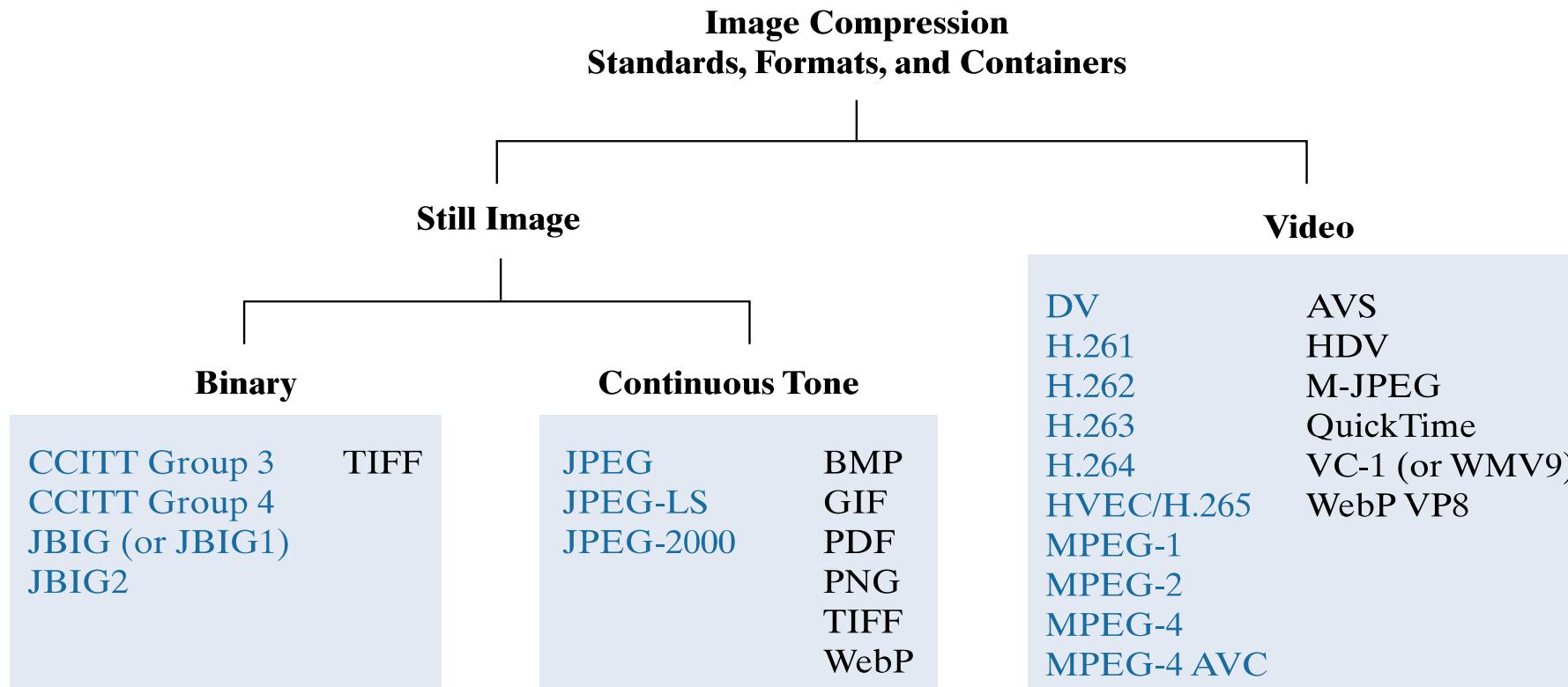
- An *image file format* is a standard way to organize and store image data.
 - It defines how the data is arranged and the type of compression (if any) that is used.
- An *image container* is similar to a file format, but handles multiple types of image data.
- Image *compression standards*, on the other hand, define procedures for compressing and decompressing images
 - that is, for reducing the amount of data needed to represent an image.
 - These standards are the underpinning of the widespread acceptance of image compression technology

DIGITAL IMAGE PROCESSING - 2

Image format, Container and Compression Standards

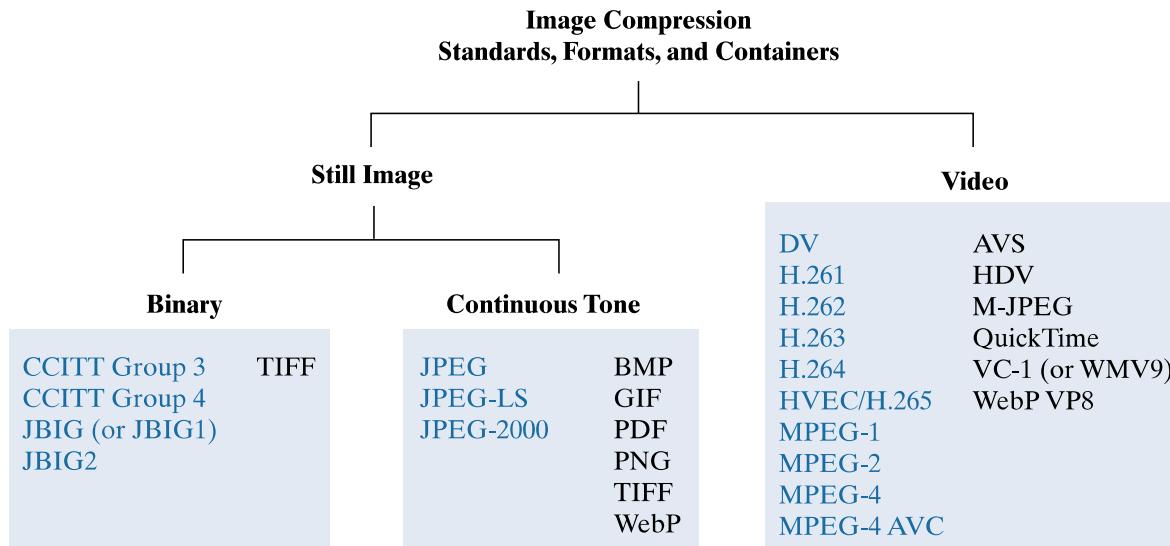
Some popular image compression standards, file formats, and containers.

(Internationally sanctioned entries are shown in blue; all others are in black)



DIGITAL IMAGE PROCESSING - 2

Image format, Container and Compression Standards



- This figure lists the most important image compression standards, file formats, and containers in use today, grouped by the type of image handled.
- The entries in blue are international standards sanctioned by the *International Standards Organization* (ISO), the *International Electrotechnical Commission* (IEC), and/or the *International Telecommunications Union* (ITU-T)—a *United Nations* (UN) organization that was once called the *Consultative Committee of the International Telephone and Telegraph* (CCITT)

Two video compression standards, VC-1 by the *Society of Motion Pictures and Television Engineers* (SMPTE) and AVS by the *Chinese Ministry of Information Industry* (MII), are also included.

Note that they are shown in black, which is used in the Figure to denote entries that are not sanctioned by an international standards organization.

DIGITAL IMAGE PROCESSING - 2

Summary of Image format, Container and Compression Standards

Name	Organization	Description
<i>Bi-Level Still Images</i>		
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 [8.6] and Huffman [8.2] 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 <i>Joint Bi-level Image Experts Group</i> 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 [8.8]. Context-sensitive arithmetic coding [8.4] 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 [8.7] for text and halftone regions, and Huffman [8.2] or arithmetic coding [8.4] for other image content. It can be lossy or lossless.

The focus of these sections is on methods that have proven useful in mainstream binary, continuous-tone still-image, and video compression standards. The standards themselves are used to demonstrate the methods presented.

This table summarizes the standards, formats, and containers listed in Figure earlier. Responsible organizations, targeted applications, and key compression methods are identified.

DIGITAL IMAGE PROCESSING - 2

Summary of Image format, Container and Compression Standards

Continuous-Tone Still Images

JPEG	ISO/IEC/ ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy baseline coding system (most commonly implemented) uses quantized discrete cosine transforms (DCT) on image blocks [8.9], Huffman [8.2], and run-length [8.6] 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 [8.10], context modeling [8.4], and Golomb coding [8.3].
JPEG- 2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.4] and quantized discrete wavelet transforms (DWT) [8.11] are used. The compression can be lossy or lossless.

DIGITAL IMAGE PROCESSING - 2

Summary of Image format, Container and Compression Standards

Internationally sanctioned video compression standards. The numbers in brackets refer to sections in this chapter

Name	Organization	Description
DV	IEC	<i>Digital Video.</i> A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (<i>integrated services digital network</i>) lines. It supports non-interlaced 352×288 and 176×144 resolution images, called CIF (<i>Common Intermediate Format</i>) and QCIF (<i>Quarter CIF</i>), respectively. A DCT-based compression approach [8.9] similar to JPEG is used, with frame-to-frame prediction differencing [8.10] 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 (<i>Sub-Quarter CIF</i> 128×96), 4CIF (704×576) and 16CIF (1408×512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, streaming, and television. It supports prediction differences within frames [8.10], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.4].
H.265 MPEG-H HEVC	ISO/IEC ITU-T	<i>High Efficiency Video Coding</i> (HVEC). An extension of H.264 that includes support for macroblock sizes up to 64×64 and additional intraframe prediction modes, both useful in 4K video applications.
MPEG-1	ISO/IEC	A <i>Motion Pictures Expert Group</i> 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 at up 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 [8.10] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 <i>Advanced Video Coding</i> (AVC). Identical to H.264.

DIGITAL IMAGE PROCESSING - 2

Summary of Image format, Container and Compression Standards

Popular image and video compression standards, file formats, and containers not included in Tables 8.3 and 8.4. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
<i>Continuous-Tone Still Images</i>		
BMP	Microsoft	<i>Windows Bitmap</i> . A file format used mainly for simple uncompressed images.
GIF	CompuServe	<i>Graphic Interchange Format</i> . A file format that uses lossless LZW coding [8.5] for 1- through 8-bit images. It is frequently used to make small animations and short low-resolution films for the Internet.
PDF	Adobe Systems	<i>Portable Document Format</i> . 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)	<i>Portable Network Graphics</i> . 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 [8.10].
TIFF	Aldus	<i>Tagged Image File Format</i> . A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
WebP	Google	<i>WebP</i> supports lossy compression via WebP VP8 intraframe video compression (see below) and lossless compression using spatial prediction [8.10] and a variant of LZW backward referencing [8.5] and Huffman entropy coding [8.2]. Transparency is also supported.
<i>Video</i>		
AVS	MII	<i>Audio-Video Standard</i> . Similar to H.264 but uses exponential Golomb coding [8.3]. Developed in China.
HDV	Company consortium	<i>High Definition Video</i> . An extension of DV for HD television that uses compression similar to MPEG-2, including temporal redundancy removal by prediction differencing [8.10].
M-JPEG	Various companies	<i>Motion JPEG</i> . 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	SMPTE	The most used video format on the Internet. Adopted for HD and <i>Blu-ray</i> high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.9 and 8.10] and context-dependent variable-length code tables [8.2], but no predictions within frames.
WMV9	Microsoft	
WebP VP8	Google	A file format based on block transform coding [8.9] prediction differences within frames and between frames [8.10]. The differences are entropy encoded using an adaptive arithmetic coder [8.4].

Lossless Compression: Huffman Coding

Lossless Coding: Compressing data without loss of information

- One of the most popular techniques for removing coding redundancy is due to **Huffman** (Huffman [1952]).
- When coding the symbols of an information source individually, **Huffman coding** yields the smallest possible number of code symbols per source symbol.
- In terms of Shannon's first theorem, the resulting code is optimal for a fixed value of n , subject to the constraint that the source symbols be coded *one at a time*.
- In practice, the source symbols may be either the intensities of an image or the output of an intensity mapping operation (pixel differences, run lengths, and so on).

Huffman Coding

Steps in the coding technique:

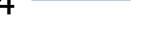
Step1: To create a series of source reductions by ordering the probabilities of the symbols under consideration, then combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a “compound symbol” with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source also are ordered from the most to the least probable.
- This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Original source	
Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

DIGITAL IMAGE PROCESSING - 2

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

Lossless Compression: Huffman Coding

Lossless Coding: Compressing data without loss of information

- One of the most popular techniques for removing coding redundancy is due to **Huffman** (Huffman [1952]).
- When coding the symbols of an information source individually, **Huffman coding** yields the smallest possible number of code symbols per source symbol.
- In practice, the source symbols may be either the intensities of an image or the output of an intensity mapping operation (pixel differences, run lengths, and so on).

Huffman Coding

Steps in the coding technique:

Step1: To create a series of source reductions by ordering the probabilities of the symbols under consideration, then combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a “compound symbol” with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source also are ordered from the most to the least probable.
- This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Original source	
Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

DIGITAL IMAGE PROCESSING - 2

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

Huffman Coding

Steps in the coding technique:

- **Step2:** To code each reduced source, starting with the smallest source and working back to the original source.
 - The minimal length binary code for a two-symbol source, of course, are the symbols 0 and 1.
 - These symbols are assigned to the two symbols on the right. (The assignment is arbitrary; reversing the order of the 0 and 1 would work just as well.)
 - As the reduced source symbol with probability 0.6 was generated by combining two symbols in the reduced source to its left, the 0 used to code it is now assigned to both of these symbols, and a 0 and 1 are arbitrarily appended to each to distinguish them from each other.
 - This operation is then repeated for each reduced source until the original source is reached

DIGITAL IMAGE PROCESSING - 2

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

The average length of this code is

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

The entropy of the source is 2.14 bits/symbol

Analysis: 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.
- After the code has been created, coding and/or error-free decoding is accomplished in a simple lookup table manner.
- The code itself is an **instantaneous uniquely decodable block code**.
 - It is called a **block code** because each source symbol is mapped into a fixed sequence of code symbols.
 - It is **instantaneous** because each code word in a string of code symbols can be decoded without referencing succeeding symbols.
 - It is **uniquely decodable** because any string of code symbols can be decoded in only one way.
- Thus, any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in a left-to-right manner.

DIGITAL IMAGE PROCESSING - 2

Decoding of Huffman Code

For the binary code obtained in previous example, decode the encoded string **010100111100**

Decoding:

Original source		
Symbol	Probability	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

- First valid code word is 01010, which is the code for symbol a_3 .
- The next valid code is ~~$a_3 a_1 a_2 a_2 a_6$~~ 0111100 , which corresponds to symbol a_1 .
- Continuing in this manner reveals the completely decoded message to be $a_3 a_1 a_2 a_2 a_6$

Analysis: Huffman Coding

- When a large number of symbols is to be coded, the construction of an optimal Huffman code is a nontrivial task.
- When source symbol probabilities can be estimated in advance, “near optimal” coding can be achieved with pre-computed Huffman codes.
- Several popular image compression standards, including the JPEG and MPEG standards specify default Huffman coding tables that have been pre-computed based on experimental data.

Arithmetic Coding

- Unlike the variable-length codes, *arithmetic coding* generates nonblock codes.
- In arithmetic coding, which can be traced to the work of Elias (Abramson [1963]), a one-to-one correspondence between source symbols and code words does not exist.
- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.
- The code word itself defines an interval of real numbers between 0 and 1.
- At the start of the coding process, the message is assumed to occupy the entire half-open interval $[0, 1)$

Arithmetic Coding

- As the number of symbols in the message increases, the interval used to represent it becomes smaller, and the number of information units (say, bits) required to represent the interval becomes larger.
- Each symbol of the message reduces the size of the interval in accordance with its probability of occurrence.
- Because the technique does not require, as does Huffman's approach, that each source symbol translate into an integral number of code symbols (that is, that the symbols be coded one at a time), it achieves (but only in theory) the bound established by Shannon's first theorem

DIGITAL IMAGE PROCESSING - 2

Arithmetic Coding Examples

Example 1:

Generate an arithmetic code for sequence $a_1 a_2 a_3 a_3 a_4$ given the following symbols and their probabilities.

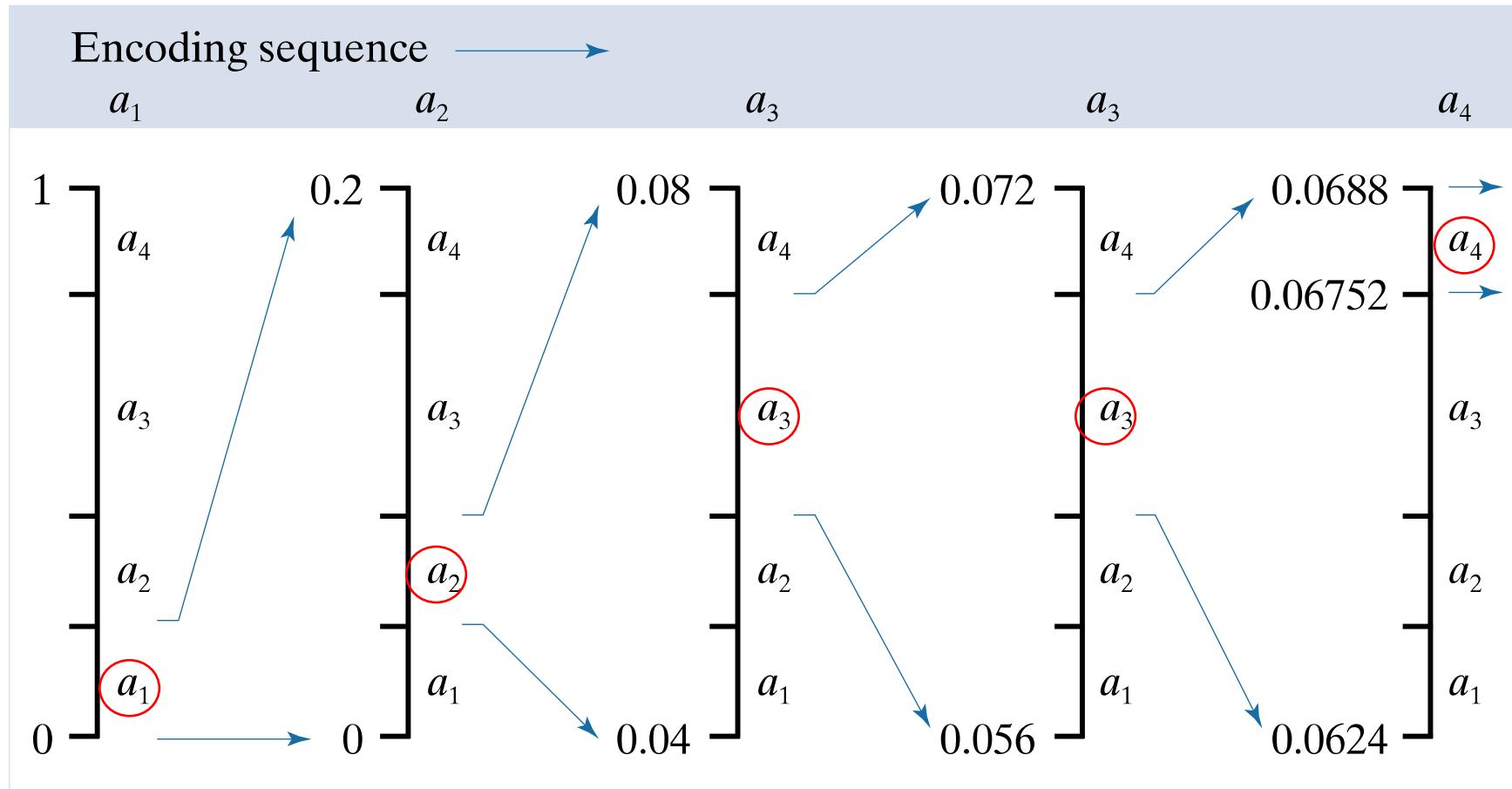
Symbol	Probability	Range
a_1	0.2	[0.0,0.2)
a_2	0.2	[0.2,0.4)
a_3	0.4	[0.4,0.8)
a_4	0.2	[0.8,1)

Soln: $d = \text{Upper bound} - \text{lower bound}$

Upper bound of symbol = lower bound + $d(\text{prob. of that symbol})$

Arithmetic Coding Examples

Coding procedure:



Arithmetic Coding

- Unlike the variable-length codes, *arithmetic coding* generates non block codes.
- In arithmetic coding, which can be traced to the work of Elias (1963), a **one-to-one correspondence between source symbols and code words does not exist.**
- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.
- The code word itself defines an interval of real numbers between 0 and 1.
- At the start of the coding process, the message is assumed to occupy the entire half-open interval $[0, 1)$

DIGITAL IMAGE PROCESSING - 2

Arithmetic Coding Examples

Example 1:

Generate an arithmetic code for sequence $a_1 a_2 a_3 a_3 a_4$ given the following symbols and their probabilities.

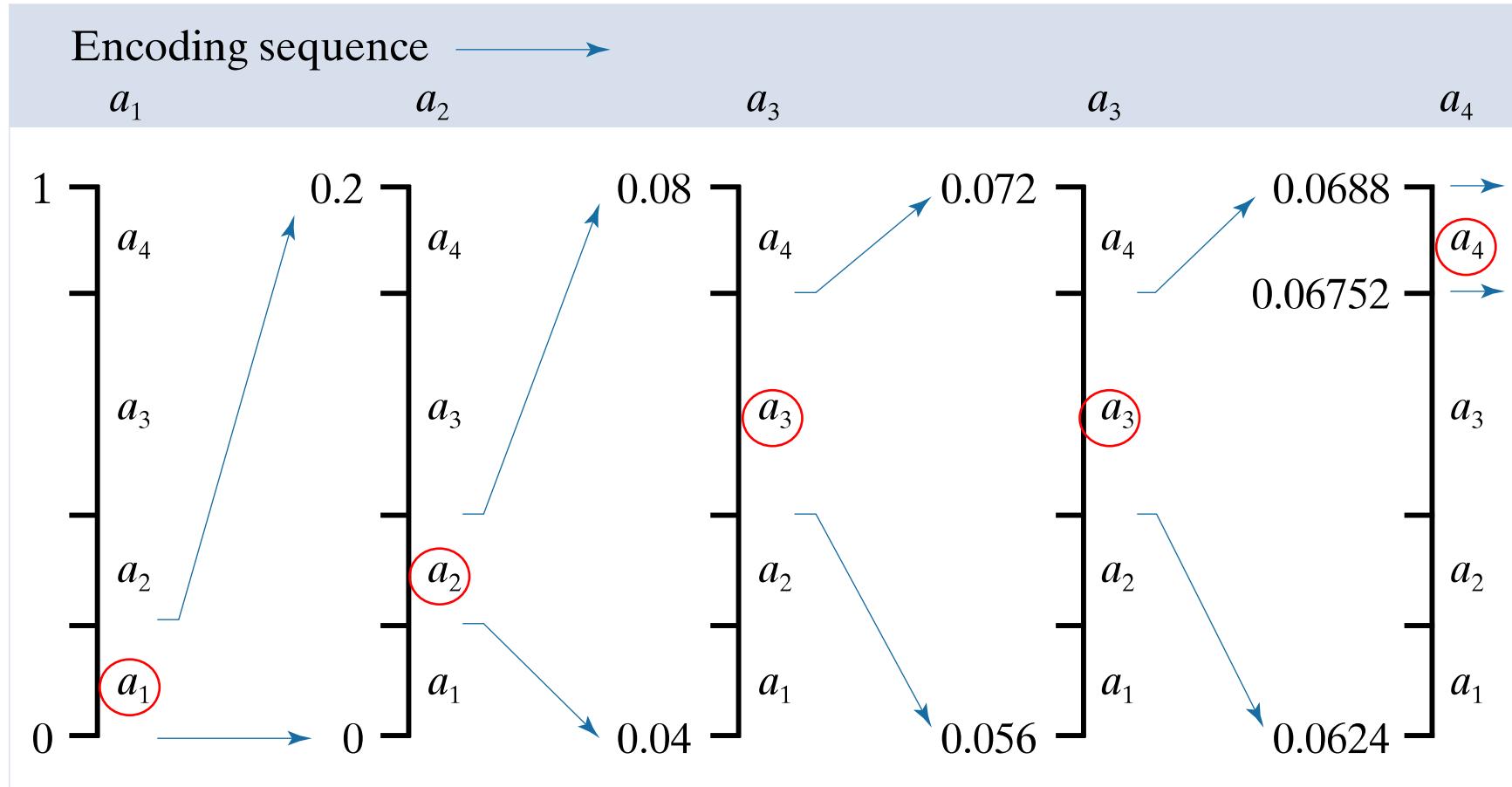
Symbol	Probability	Range
a_1	0.2	[0.0,0.2)
a_2	0.2	[0.2,0.4)
a_3	0.4	[0.4,0.8)
a_4	0.2	[0.8,1)

Soln: $d = \text{Upper bound} - \text{lower bound}$

Upper bound of symbol = lower bound + $d(\text{prob. of that symbol})$

Arithmetic Coding Examples

Coding procedure to code for sequence $a_1 a_2 a_3 a_3 a_4$:



Arithmetic Coding

Arithmetic code for this sequence is:

$0.06756 < \text{Codeword} < 0.0688$

Tag: $(0.06756 + 0.0688) / 2 = 0.0682$

- Any value in this range can be used to represent the message sequence
- **If using 3 decimal digits then 0.068 can be the code**
- Hence 3 decimal digits are used to represent the 5 symbol message
- This translates into 0.6 decimal digits per source symbol(3/5)
- Entropy of the source is 0.58 decimal digits per source symbol
- As the length of the sequence being coded increases, the resulting arithmetic code approaches the bound established by Shannon's first theorem.

DIGITAL IMAGE PROCESSING - 2

Arithmetic Coding Example

Example 2:

Decode the message 0.572 given the following coding model:

Symbol	Probability
C	0.4
E	0.5
.	0.1

Sol: The decoded stream is '**ECE.**'

Drawback of Arithmetic Coding

Two factors cause coding performance to fall short of the bound:

1. The addition of the end-of-message indicator that is needed to separate one message from another
2. The use of finite precision arithmetic.
3. Practical implementations of arithmetic coding address the latter problem by introducing a scaling strategy and a rounding strategy

DIGITAL IMAGE PROCESSING - 2

LZW Coding

- The techniques discussed so far are focused on the removal of coding redundancy.
- We now consider an error-free compression approach that also addresses spatial redundancies in an image.
- The technique, called *Lempel-Ziv- Welch (LZW) coding*, assigns fixed-length code words to variable length sequences of source symbols.
- It is an optimal prefix code
- A key feature of LZW coding is that it requires no apriori knowledge of the probability of occurrence of the symbols to be encoded.
 - Despite the fact that until recently it was protected under a United States patent, LZW compression has been integrated into a variety of main-stream imaging file formats, including GIF, TIFF, and PDF.
 - The PNG format was created to get around LZW licensing requirements.

LZW Coding Technique

- LZW coding is conceptually very simple (Welch [1984])
- 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 at locations 1,2,.....,256
- As the encoder sequentially examines image pixels, intensity sequences that are not in the dictionary are placed in algorithmically determined (e.g., the next unused) locations.
 - Shortest subsequence not encountered previously is placed in pre-determined locations

LZW Coding Technique

- LZW coding is accomplished by passing the source data stream into segments that are the Shortest subsequence not encountered previously

Step 1: Form the subsequences → Subsequence generation which results in 'phrases'

Step 2: Code the phrase by writing the location of prefix(in binary) and value of the last bit

DIGITAL IMAGE PROCESSING - 2

Coding Example of LZW Coding

1. Code the given sequence using LZW coding:

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1.....

Assume binary sequence 0,1 are already stored in memory in numerical position (say 1,2...)

LZW Coding

- *Lempel-Ziv- Welch (LZW) coding* is an error-free/lossless compression approach that also addresses spatial redundancies in an image.
- The technique, assigns fixed-length code words to variable length sequences of source symbols.
- It is an optimal prefix code

LZW Coding Technique

- LZW coding is accomplished by passing the source data stream into segments that are the Shortest subsequence not encountered previously

Step 1: Form the subsequences → Subsequence generation which results in 'phrases'

Step 2: Code the phrase by writing the location of prefix(in binary) and value of the last bit

DIGITAL IMAGE PROCESSING - 2

Coding Example of LZW Coding

1. Code the given sequence using LZW coding:

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1.....

Assume binary sequence 0,1 are already stored in memory in numerical position (say 1,2...)

2. Code the given sequence using LZW coding:

A A B A B B B A B A A B A B B B A B B A B B.....

Assume A and B are stored in locations 1 and 2

A → 0

B → 1

Analysis of LZW Coding

- Clearly, the size of the dictionary is an important system parameter
- If it is too small, the detection of matching intensity level sequences will be less likely
- If it is too large, the size of the code words will adversely affect compression performance
- A unique feature of the LZW coding is that the coding dictionary or code book is created while the data are being encoded.
- Remarkably, an LZW decoder builds an identical decompression dictionary as it simultaneously decodes the encoded data stream

Run-Length Coding (RLE)

- Images with **repeating intensities** along their rows (or columns) can often be compressed by representing runs of identical intensities as *run-length pairs*
 - where each run-length pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity.
- *Run-length encoding* (RLE), was developed in the 1950s and became, along with its 2-D extensions, the standard compression approach in facsimile (FAX) coding.
- Compression is achieved by eliminating a simple form of spatial redundancy—groups of identical intensities.
- When there are few (or no) runs of identical pixels, run-length encoding results in data expansion.

Run-Length Coding (RLE)

- Run-length encoding is particularly effective when compressing binary images.
 - Because there are only two possible intensities (black and white), adjacent pixels are more likely to be identical.
 - In addition, each image row can be represented by a sequence of lengths only, rather than length-intensity pairs.
 - The basic idea is to code each contiguous group (i.e., run) of 0's or 1's encountered in a left-to-right scan of a row by its length *and* to establish a convention for determining the value of the run.
 - The most common conventions are (1) to specify the value of the first run of each row, or (2) to assume that each row begins with a white run, whose run length may in fact be zero.

Run-Length Coding (RLE)

- Although run-length encoding is in itself an effective method of compressing binary images, additional compression can be achieved by variable-length coding the run lengths themselves.
- The black and white run lengths can be coded separately using variable-length codes that are specifically tailored to their own statistics.

Bit Plane Coding

- The previous techniques can be applied to images with more than two intensities by individually processing their bit planes.
- *Bit-plane coding*, is based on the concept of decomposing a multilevel (monochrome or color) image into a series of binary images and compressing each binary image via one of several well-known binary compression methods
- The intensities of an m -bit monochrome image can be represented in the form of the base-2 polynomial

$$a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

Bit Plane Coding

$$a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

- Based on this property, a simple method of decomposing the image into a collection of binary images is to separate the m coefficients of the polynomial into m 1-bit bit planes.
- The lowest-order bit plane (the plane corresponding to the least significant bit) is generated by collecting the a_0 bits of each pixel, while the highest-order bit plane contains the a_{m-1} bits or coefficients.
- In general, each bit plane is constructed by setting its pixels equal to the values of the appropriate bits or polynomial coefficients from each pixel in the original image.

Bit Plane Coding

- The inherent disadvantage of this decomposition approach is that small changes in intensity can have a significant impact on the complexity of the bit planes.
- If a pixel of intensity 127 (01111111) is adjacent to a pixel of intensity 128 (10000000), for instance, every bit plane will contain a corresponding 0 to 1 (or 1 to 0) transition.
- For example, because the most significant bits of the binary codes for 127 and 128 are different, the highest bit plane will contain a zero-valued pixel next to a pixel of value 1, creating a 0 to 1 (or 1 to 0) transition at that point.

Bit Plane Coding

- An alternative decomposition approach (which reduces the effect of small intensity variations) is to first represent the image by an m -bit *Gray code*.
- The m -bit Gray code $g_{m-1} \dots g_2g_1g_0$ that corresponds to the earlier binary polynomial can be computed from

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

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

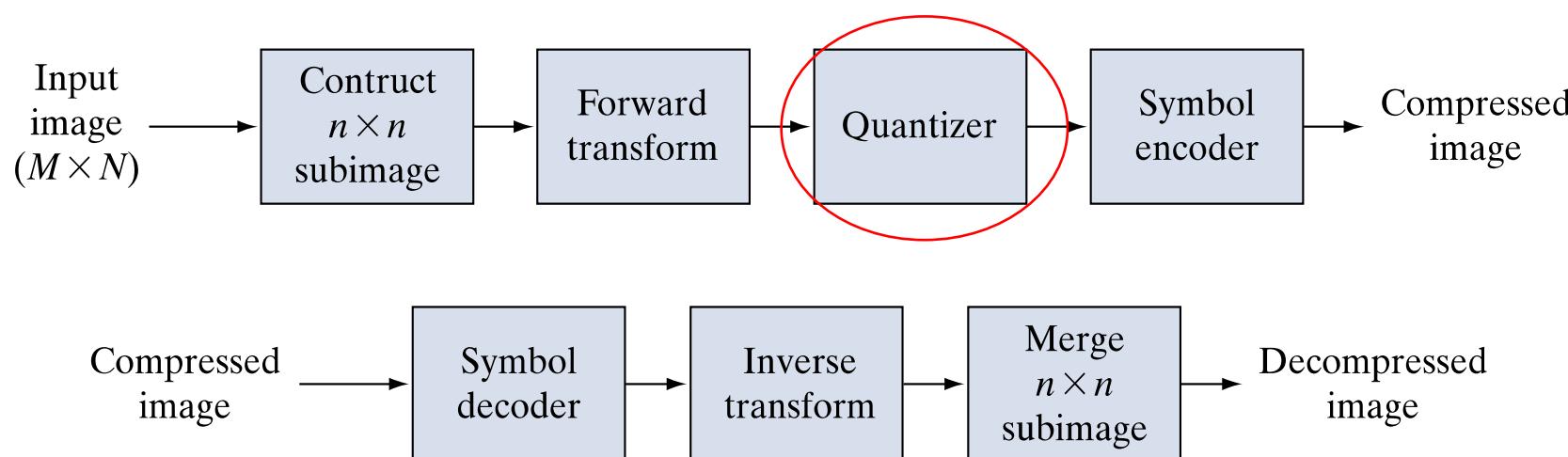
Here, \oplus denotes the exclusive OR operation.

Bit Plane Coding

- This code has the unique property that successive code words differ in only one bit position.
- Thus, small changes in intensity are less likely to affect all m bit planes.
- For instance, when intensity levels 127 and 128 are adjacent, only the highest-order bit plane will contain a 0 to 1 transition, because the Gray codes that correspond to 127 and 128 are 01000000 and 11000000, respectively.

Transform Coding

- *Transform Coding* is a lossy compression technique where some information loss is acceptable
- Block diagram of a transform coder:



Transform Coding

- This is a compression technique that divides an image into small non-overlapping blocks of equal size (e.g., 8 x 8) and processes the blocks independently using a 2-D transform.
- In *block transform coding*, a reversible, linear transform (such as the Fourier transform) is used to map each block or subimage 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 (or discarded entirely) with little image distortion
- Variety of transforms like DFT, DCT, DWT can be used for compression

Transform Coding

- An $M \times N$ input image is subdivided first into subimages of size $n \times n$,
- These are then transformed to generate MN/n^2 subimage transform arrays, each of size $n \times n$.
- The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients.
- The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least amount of information in a predefined sense.

Transform Coding

- These coefficients have the smallest impact on reconstructed subimage quality.
- The encoding process terminates by coding (normally using a variable-length code) the quantized coefficients.
- Any or all of the transform encoding steps can be adapted to local image content, called *adaptive transform coding*, or fixed for all subimages, called *nonadaptive transform coding*.
- The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and the computational resources available.
- Compression is achieved during the quantization of the transformed coefficients (not during the transformation step)

DIGITAL IMAGE PROCESSING - 2

Test Images

Some popular test images:



Lena



Barbara



Cameraman



Goldhill

Transform Coding Example

Consider the uncompressed 512 x 512, 8-bit image of Lena



- Now divide the original image into subimages of size 8×8 , representing each subimage using three of the transforms DFT, WHT and DCT
- Truncate 50% of the resulting coefficients, and take the inverse transform of the truncated coefficient arrays.

DIGITAL IMAGE PROCESSING - 2

Transform Coding Example

- In each case, the 32 retained coefficients were selected on the basis of maximum magnitude.



Original



Using DFT



Using WHT



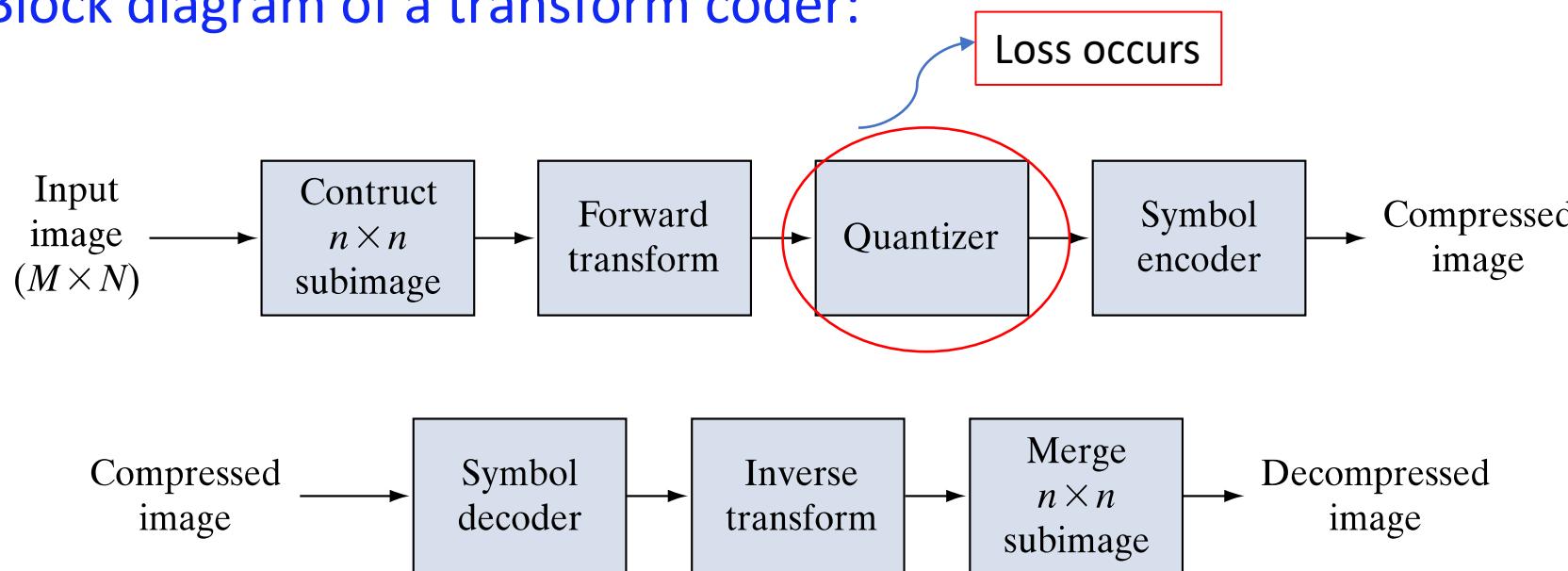
Using DCT

Scaled error images

DIGITAL IMAGE PROCESSING - 2

Transform Coding

- *Transform Coding* is a lossy compression technique where some information loss is acceptable
- Block diagram of a transform coder:



DIGITAL IMAGE PROCESSING - 2

Transform Coding Example

Consider the uncompressed 512×512 , 8-bit image of Lena



- Now divide the original image into subimages of size 8×8 , representing each subimage using three of the transforms

DFT, WHT and DCT

- Truncate 50% of the resulting coefficients, and take the inverse transform of the truncated coefficient arrays

DIGITAL IMAGE PROCESSING - 2

Transform Coding Example

- In each case, the 32 retained coefficients were selected on the basis of maximum magnitude.



Original



Using DFT



Using WHT



Using DCT

Scaled error images

Transform Coding Example analysis

- In all cases, the 32 discarded coefficients had little visual impact on the quality of the reconstructed image.
- Their elimination, however, was accompanied by some mean-squared error, which can be seen in the scaled error images
- The actual rms errors were 2.32, 1.78, and 1.13 intensities, respectively.
- The small differences in mean-squared reconstruction error noted in the example are related directly to the **energy or information packing properties of the transforms employed**

- **Inference: Information packing ability of the DCT is superior to that of the DFT and WHT.**

Comparison of Transforms for Compression

- Although this condition usually holds for most images, the Karhunen-Loëve transform ([KL transform](#)), not the DCT, is the optimal transform in an information packing sense
- This is due to the fact that the KLT minimizes the mean-squared error for any input image and any number of retained coefficients
- However, because the KLT is data dependent, obtaining the KLT basis images for each subimage, in general, is a nontrivial computational task
- **For this reason, the KLT is used infrequently for image compression.**
 - Instead, a transform, such as the DFT, WHT, or DCT, whose basis images are fixed (input independent) is normally used

Comparison of Transforms for Compression

- Of the possible input independent transforms, the **nonsinusoidal transforms (such as the WHT transform)** are the simplest to implement
- The **sinusoidal transforms (such as the DFT or DCT)** more closely approximate the information packing ability of the optimal KLT.
- **Hence, most transform coding systems are based on the DCT, which provides a good compromise between information packing ability and computational complexity**

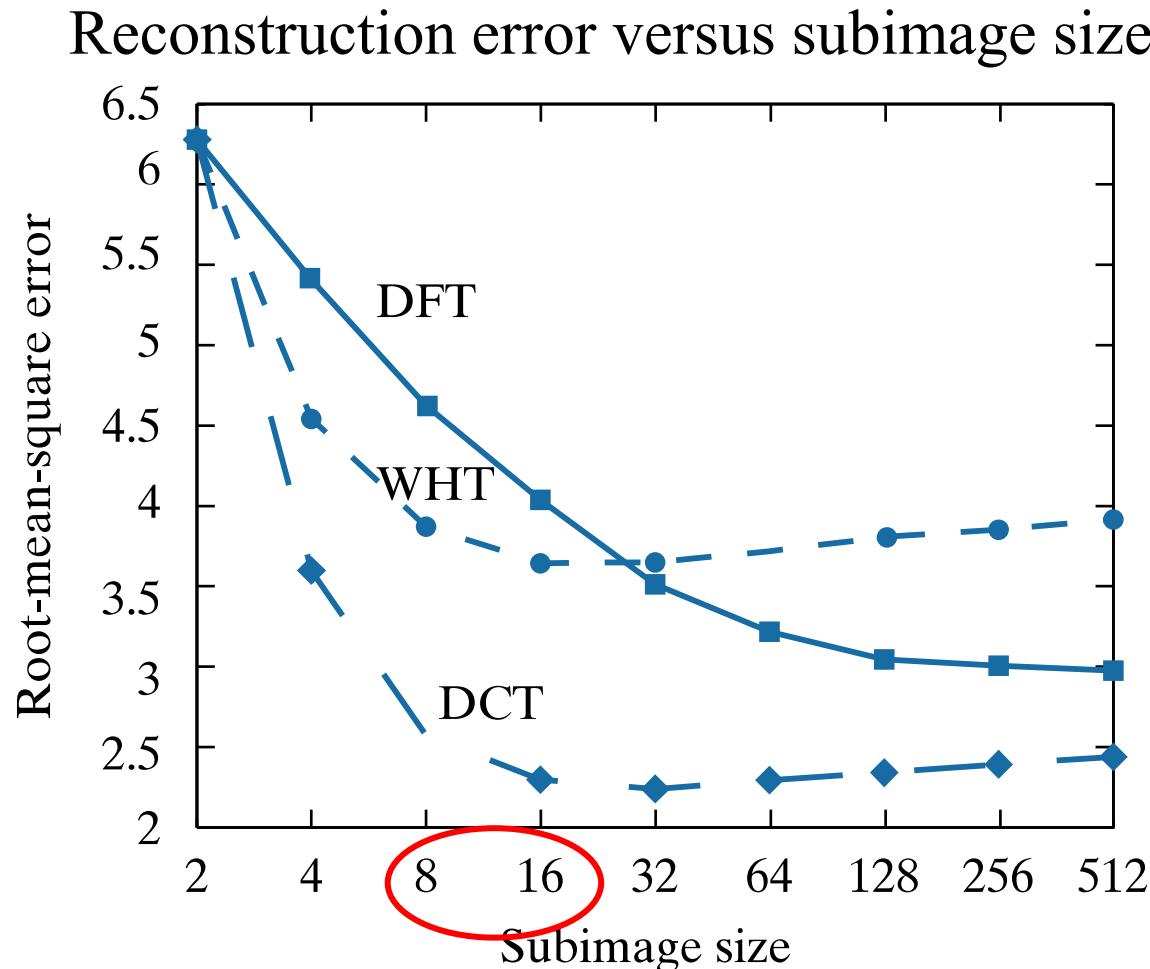
Comparison of Transforms for Compression

- Properties of the DCT have proved to be of such practical value that the DCT is an international standard for transform coding systems(JPEG, MPEG...)
- Compared to the other input independent transforms, it has the advantages of having been implemented in a single integrated circuit, packing the most information into the fewest coefficients (for most images), and minimizing the block-like appearance, called *blocking artifacts*, that results when the boundaries between subimages become visible

Size of Subimage

- Another significant factor affecting transform coding error and computational complexity is **subimage size**
- In most applications, images are subdivided so the correlation (redundancy) between adjacent subimages is reduced to some acceptable level and so n is an integer power of 2 where, as before, n is the subimage dimension
- The latter condition simplifies the computation of the subimage transforms.
- In general, both the level of compression and computational complexity increase as the subimage size increases.
- The most popular subimage sizes are 8×8 and 16×16

Size of Subimage



Size of Subimage

- The transform of each subimage is computed. 75% of the resulting coefficients are truncated, and the inverse transform of the truncated arrays is computed.
- The Hadamard and cosine curves flatten as the size of the subimage becomes greater than 8×8 , whereas the Fourier reconstruction error continues to decrease in this region.
- As n further increases, the Fourier reconstruction error crosses the Walsh-Hadamard curve and approaches the cosine result.
- This result is consistent with the theoretical and experimental findings reported by Netravali and Limb [1980] and by Pratt [2001] for a 2-D Markov image source.

DIGITAL IMAGE PROCESSING - 2

Example

Consider compression and reconstruction of the following 8×8 subimage with the JPEG baseline standard:

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

DIGITAL IMAGE PROCESSING - 2

Example

The original image consists of 256 or 2^8 possible intensities (grayscale), so the coding process begins by level shifting the pixels of the original subimage by $\square 2^7$ or $\square 128$ intensity levels (-128 to +127). The resulting shifted array is

-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

DIGITAL IMAGE PROCESSING - 2

Example

When transformed in accordance with the forward DCT for $n = 8$
it becomes

-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

DIGITAL IMAGE PROCESSING - 2

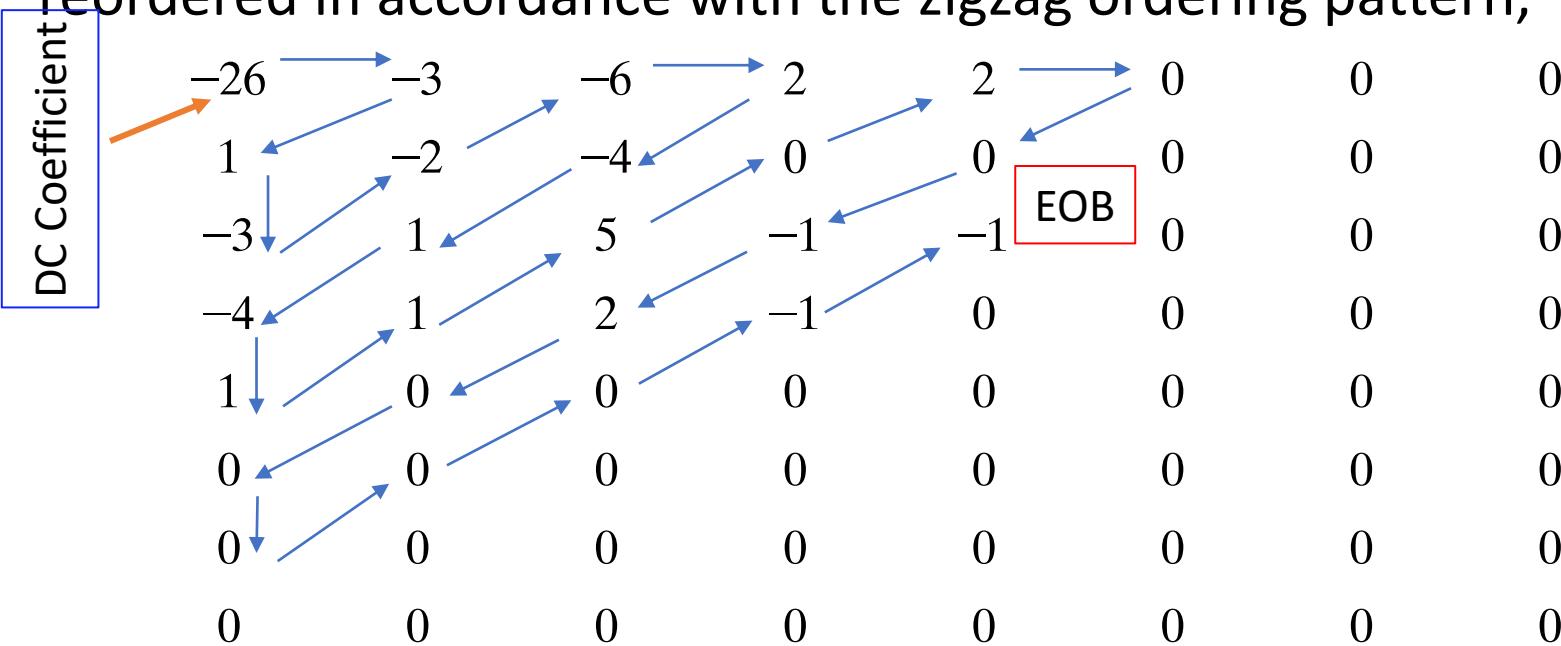
Example

If the JPEG normalization array is used to quantize the transformed array, the scaled and truncated coefficients are

DC Coefficient	-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

The transformation and normalization process produces a large number of zero-valued coefficients. The coefficients are reordered in accordance with the zigzag ordering pattern,



DIGITAL IMAGE PROCESSING - 2

Example

The resulting 1-D coefficient sequence is

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

End of Block

Any coding technique can be applied to this stream to code it and then the reverse process decompresses the image to its original size



THANK YOU

Prof. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu

+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 3: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Types of Redundancies Cont..

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Introduction to image compression
 - Redundancies in image

Types of Redundancy

- Coding Redundancy
- Spatial and temporal redundancy
- Irrelevant information

Coding Redundancy Cont..

Coding Redundancy Example:

- Consider the computer generated image
- Consider its intensity distribution as shown in table below

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

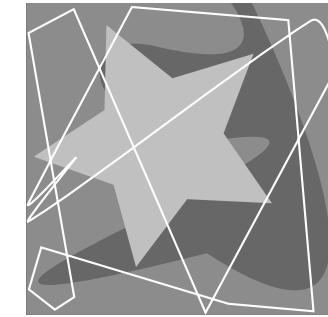


Coding Redundancy Example

Code 1:

- If a natural 8-bit binary code is used to represent its four possible intensities

r_k	$p_r(r_k)$	Code 1
$r_{87} = 87$	0.25	01010111
$r_{128} = 128$	0.47	01010111
$r_{186} = 186$	0.25	01010111
$r_{255} = 255$	0.03	01010111
r_k for $k \neq 87, 128, 186, 255$	0	—

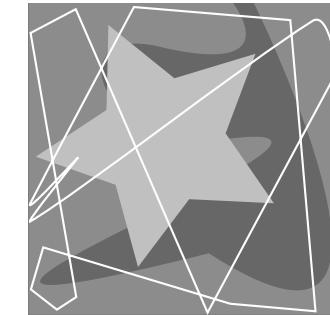


- L_{avg} (the average number of bits for code 1) is 8 bits
(because $I_1(r_k) = 8$ bits for all r_k)

Coding Redundancy Example

Code 2:

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	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0



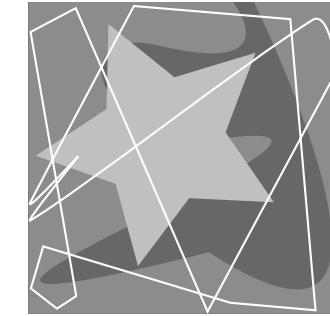
- The average length of the encoded pixels is $L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$

$$L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$
- The total number of bits needed to represent the entire image is

Coding Redundancy Example

Code 2:

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	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0



- The resulting compression is $C = \frac{256 \times 256 \times 8}{118,621} = \frac{8}{1.81} \approx 4.42$
- And corresponding relative redundancy $R = 1 - \frac{1}{4.42} = 0.774$
- Thus, 77.4% of the data in the original 8-bit 2-D intensity array is redundant

Coding Redundancy Cont...

Analysis:

- The compression achieved by code 2 results from assigning fewer bits to the more probable intensity values than to the less probable ones.
- In the resulting *variable-length code*, r_{128} (the image's most probable intensity) is assigned the 1-bit code word 1 [of length $I_2(128) = 1$], while r_{255} (its least probable occurring intensity) is assigned the 3-bit code word 001 [of length $I_2(255) = 3$]
- The best *fixed-length code* that can be assigned to the intensities of the image is the natural 2-bit counting sequence {00,01,10,11}, but the resulting compression is only 8/2 or 4:1 which is about 10% less than the 4.42:1 compression of the variable-length code

Coding Redundancy Cont...

Analysis:

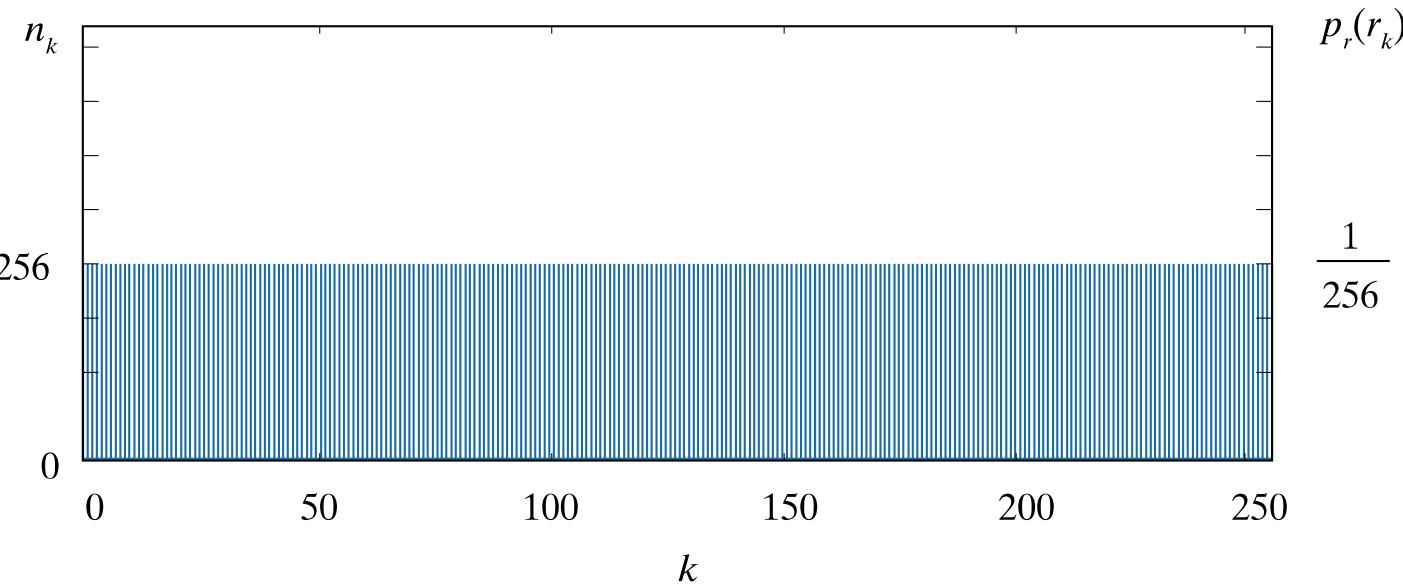
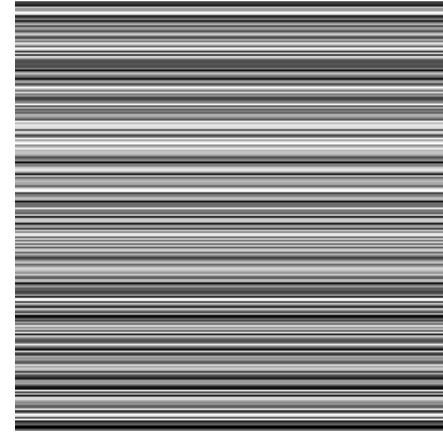
- *coding redundancy* is present when the codes assigned to a set of events (such as intensity values) do not take full advantage of the probabilities of the events
- Coding redundancy is almost always present when the intensities of an image are represented using a natural binary code
- The reason is that most images are composed of objects that have a regular and somewhat predictable morphology (shape) and reflectance, and are sampled so the objects being depicted are much larger than the picture elements
- The natural consequence is that for most images, certain intensities are more probable than others (that is, the histograms of most images are not uniform). A natural binary encoding assigns the same number of bits to both the most and least probable values, failing to minimize L_{avg} , and resulting in coding redundancy

Types of Redundancy

- Coding Redundancy Cont.
- Spatial and temporal redundancy
- Irrelevant redundancy

Spatial and Temporal Redundancy

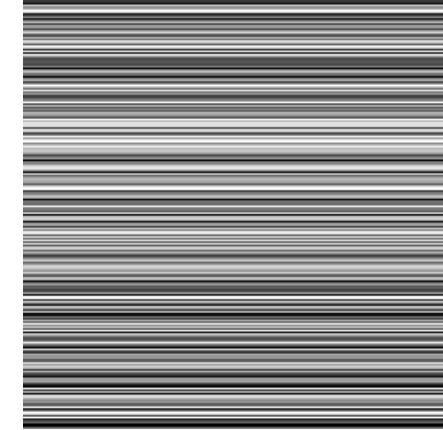
- Consider the computer-generated collection of constant intensity lines
1. All 256 intensities are equally probable. The histogram of the image is uniform.



Spatial and Temporal Redundancy

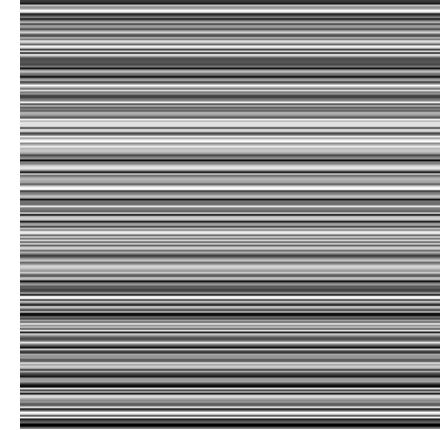
- The image in Figure (when represented as a conventional 8-bit intensity array) cannot be compressed by variable-length coding alone.
- Unlike the images whose histogram is *not* uniform, a fixed-length 8-bit code in this case minimizes

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



Spatial and Temporal Redundancy

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 (completely dependent on one another) in the horizontal direction.



Observation: This indicates a significant spatial redundancy that can be eliminated by representing the image as a sequence of ***run-length pairs***

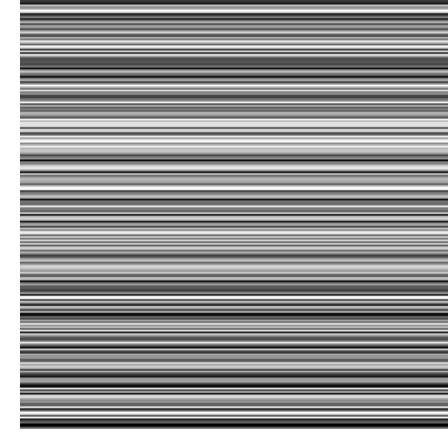
- where each **run-length pair** specifies the start of a new intensity and the number of consecutive pixels that have that intensity.

Spatial and Temporal Redundancy

- The compression ratio using a run-length based representation of the original 2-D, 8-bit intensity array would be

$$(256 * 256 * 8) / [(256 + 256) * 8] \text{ or } 128:1.$$

- 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.



Spatial and Temporal Redundancy

- In most images, pixels are correlated spatially (in both x and y) and in time (when the image is part of a video sequence).
- Because most pixel intensities can be predicted reasonably well from neighboring intensities, 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 spatially and temporally correlated pixels, a 2-D intensity array must be transformed into a more efficient but usually “non-visual” representation.

Spatial and Temporal Redundancy

- For example, run-lengths or the differences between adjacent pixels can be used.
- Transformations of this type are called *mappings*.
- A mapping is said to be *reversible* if the pixels of the original 2-D intensity array can be reconstructed without error from the transformed data set; otherwise, the mapping is said to be *irreversible*.

Types of Redundancy

- Coding Redundancy
- Spatial and temporal redundancy
- Irrelevant information

Irrelevant Information

- One of the simplest ways to compress a set of data is to remove superfluous data from the set.
- In the context of digital image compression, information that is ignored by the human visual system, or is extraneous to the intended use of an image, are obvious candidates for omission
- Thus, the computer-generated image in Figure, can be represented by its average intensity alone—a single 8-bit value.
 - because it appears to be a homogeneous field of gray
 - The original $256 * 256 * 8$ bit intensity array is reduced to a single byte



Irrelevant Redundancy

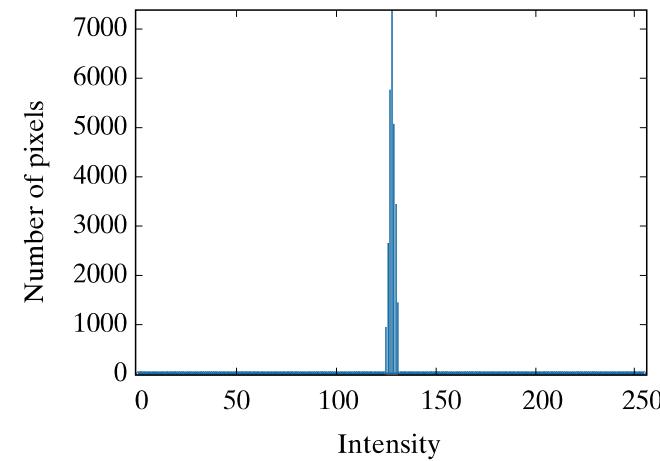
- The resulting compression is

$(256 * 256 * 8) / 8$ or 65,536:1.

- Of course, the original 256 * 256 * 8 bit image must be recreated to view and/or analyze it, but there would be little or no perceived decrease in reconstructed image quality.

Irrelevant Redundancy

- In the histogram of the image, there are several intensity values (125 through 131) actually present.
- The human visual system (HVS) averages these intensities, perceives only the average value, then ignores the small changes in intensity that are present in this case



Irrelevant Information

- The redundancy examined here is fundamentally different from the first two types of redundancies
- Its elimination is possible because the information itself is not essential for normal visual processing and/or the intended use of the image.
- Because its omission results in a loss of quantitative information, its removal is commonly referred to as *quantization*.
- This terminology is consistent with normal use of the word, which generally means the mapping of a broad range of input values to a limited number of output values. Because information is lost, quantization is an irreversible operation

Summary

- Types of Redundancies
 - Coding Redundancy
 - Spatial and temporal redundancy
 - Irrelevant redundancy

Next Session

- Element of Information theory
- Image Compression Models



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 9482219115

DIGITAL IMAGE PROCESSING - 2

Lecture 5: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Information Theory Approach

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Coding Redundancy
- Spatial and temporal redundancy
- Irrelevant redundancy

This Session

- Element of Information theory
- Image Compression Models

Measuring Image Information

- **Image Compression:** Goal is to reduce the size of image to its minimum / optimum size **without losing information**
- What is the minimum size actually needed to represent the information in an image?
- Is there a minimum amount of data that is sufficient to describe an image **without losing information?**
- ***Information theory*** provides the mathematical framework to answer this and related questions.
 - Its fundamental premise is that the generation of information can be modeled as a probabilistic process which can be measured in a manner that agrees with intuition.

Measuring Image Information

- In accordance with this supposition, a random event E with probability $P(E)$ is said to contain $I(E)$ units of information where

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

- If $P(E) = 1$ (that is, the event always occurs), $I(E) = 0$ and no information is attributed to it.
- Because no uncertainty is associated with the event, no information would be transferred by communicating that the event has occurred [it *always* occurs if $P(E) = 1$]
- The base of the logarithm determines the unit used to measure information. If the base m logarithm is used, the measurement is said to be in m -ary units.
- If the base 2 is selected, the unit of information is the *bit*.

Measuring Image Information

- if $P(E) = 1/2$, $I(E) = - \log_2(1/2)$ or 1 bit.
- That is, 1 bit is the amount of information conveyed when one of two possible equally likely events occurs.
- Given a source of statistically independent random events from a discrete set of possible events $\{a_1, a_2, \dots, a_J\}$ with associated probabilities $\{P(a_1), P(a_2), \dots, P(a_J)\}$, the average information per source output, called the *entropy* of the source, is

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

- The a_j in this equation are called *source symbols*. Because they are statistically independent, the source itself is called a *zero-memory source/memoryless 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.
- Then, the intensity source’s entropy becomes

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

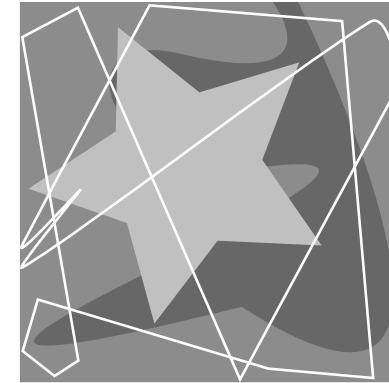
where variables L , r_k , and $p_r(r)$ are as defined earlier

- Because the base 2 logarithm is used, \tilde{H} is the average information per intensity output of the imaginary intensity source in bits.
- 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.

Measuring Image Information

- Example:
- Determine the entropy of the image given the probability as in table

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 $k = 87, 128, 186, 255$	0



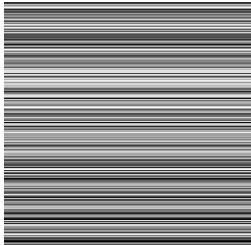
- Using

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

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

Measuring Image Information

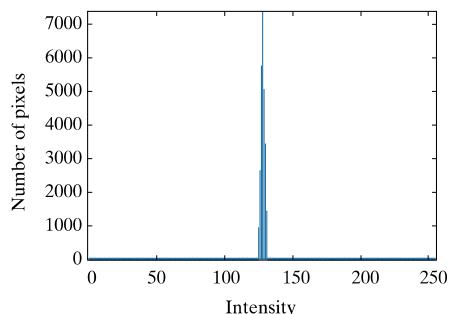
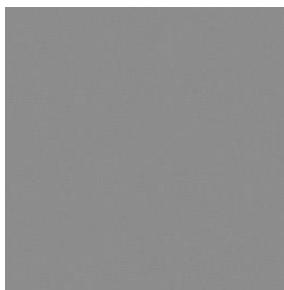
- Similarly entropy of



$$I_S = -(1/256 \log_2 256) \times 256 = 8 \text{ bpp}$$

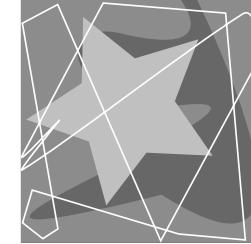
- And entropy of

$$I_S = 1.566 \text{ bpp}$$



Analysis

- Note that the first image appears to have the most visual information, but has almost the lowest computed entropy— 1.66 bits/pixel.
- The second image has almost five times the entropy of the first image, but appears to have about the same (or less) visual information.
- The third image, which seems to have little or no information, has almost the same entropy as the first image.
- **The obvious conclusion is that the amount of entropy, and thus information in an image, is far from intuitive.**



$H=1.6614$ bpp



$H=8$ bpp

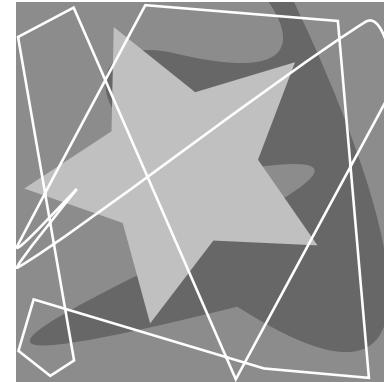


$H=1.566$ bpp

Shannon's First Theorem

Consider the image with VLC

r_k	$p_r(r_k)$	Code 2	$I_2(r_k)$
$r_{87} = 87$	0.25	01	2
$r_{128} = 128$	0.47	1	1
$r_{186} = 186$	0.25	000	3
$r_{255} = 255$	0.03	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	0



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

Entropy, $H=1.6614$ bpp

Shannon's First Theorem

- The variable-length code in the example was able to represent the intensities of the image 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 Figure can be represented with as few as 1.6614 bits/pixel.

Shannon's First Theorem

- Shannon looked at representing groups of consecutive source symbols with a single code word (rather than one code word per source symbol), and showed that

$$\lim_{n \rightarrow \infty} \left[\frac{L_{\text{avg},n}}{n} \right] = H$$

where $L_{\text{avg},n}$ is the average number of code symbols required to represent all n -symbol groups

- This equation tells us that $L_{\text{avg},n}/n$ can be made arbitrarily close to H by encoding infinitely long extensions of the single-symbol source.
- That is, it is possible to represent the output of a zero-memory source with an average of H information units per source symbol.

Summary

- Measuring information
- Average Information: Entropy
- Shannon's theorem for optimum code

Next Session

- Image Compression Models
- Lossless coding



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 6: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Measuring information
- Average Information: Entropy
- Shannon's theorem for optimum code

This Session

- Image Compression Models
- Lossless coding
 - Huffman Codes

Measuring Image Information

- A random event E with probability $P(E)$ is said to contain $I(E)$ units of information where

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

- Average Information, Entropy of the source is given by

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

- The amount of entropy, and thus information in an image, is far from intuitive

Shannon's First Theorem

- Shannon looked at representing groups of consecutive source symbols with a single code word (rather than one code word per source symbol), and showed that

$$\lim_{n \rightarrow \infty} \left[\frac{L_{\text{avg},n}}{n} \right] = H$$

where $L_{\text{avg},n}$ is the average number of code symbols required to represent all *n*-symbol groups

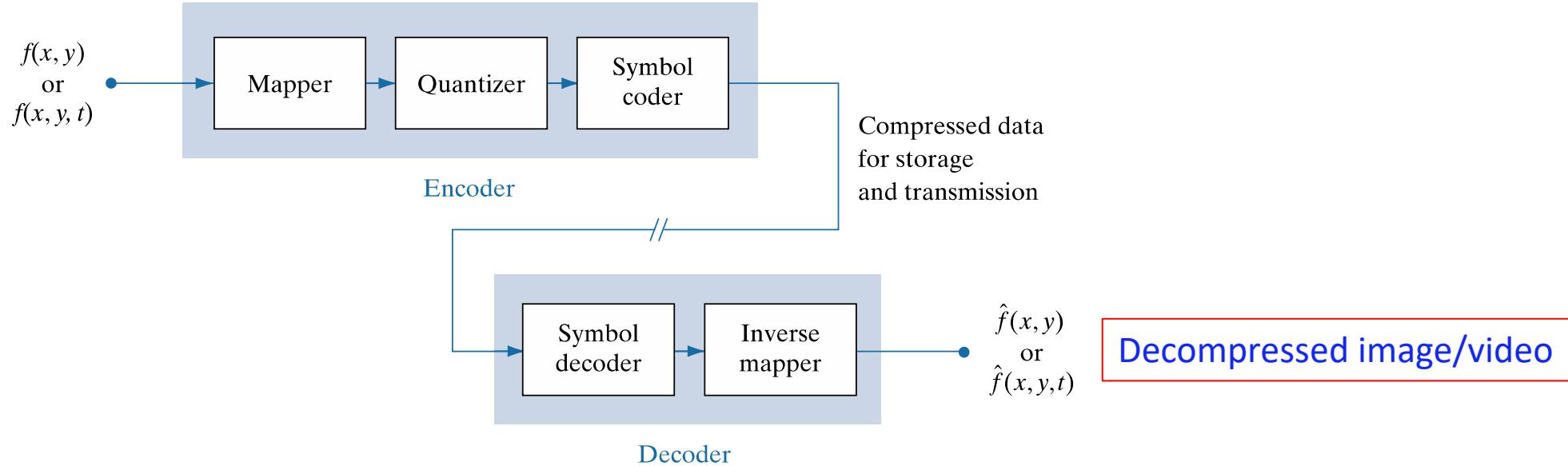
- This equation tells us that $L_{\text{avg},n}/n$ can be made arbitrarily close to H by encoding infinitely long extensions of the single-symbol source.
- That is, it is possible to represent the output of a zero-memory source with an average of H information units per source symbol.

Image Compression Models

- An image compression system is composed of two distinct functional components: an *encoder* and a *decoder*
- The *encoder* performs *compression*, and the decoder performs the complementary operation of *decompression*
- Both operations can be performed in software, as is the case in Web browsers and many commercial image-editing applications, or in a combination of hardware and firmware, as in commercial DVD players
- A *codec* is a device or program that is capable of **both encoding and decoding**

Image Compression Models

Functional block diagram of a general image compression system



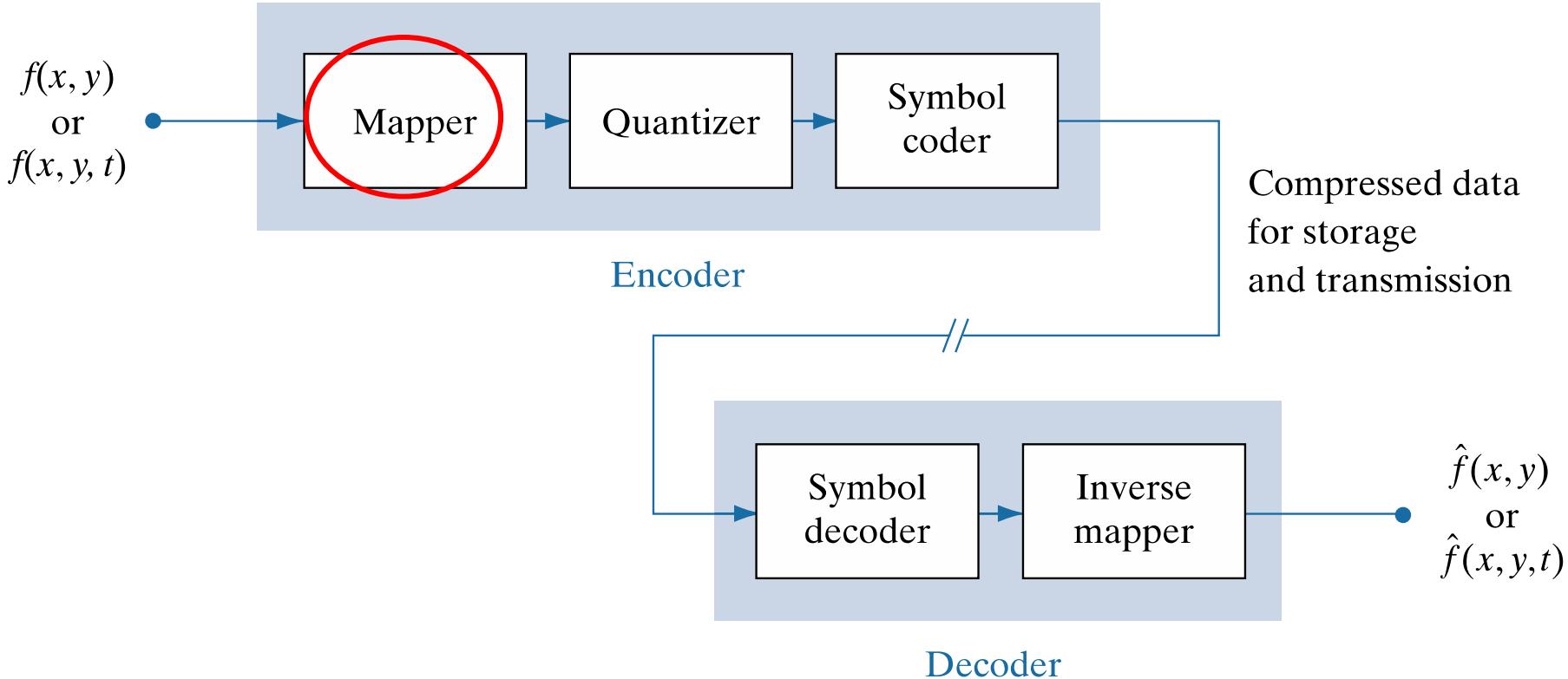
- Input image $f(x, \dots)$ is fed into the encoder, which creates a compressed representation of the input
- This representation is stored for later use, or transmitted for storage and use at a remote location
- When the compressed representation is presented to its complementary decoder, a reconstructed output image $\hat{f}(x, \dots)$ is generated

Image Compression Models

- In still-image applications, the encoded input and decoder output are $f(x, y)$ and $\hat{f}(x, y)$, respectively
- In video applications, they are $f(x, y, t)$ and $\hat{f}(x, y, t)$, where the discrete parameter t specifies time
- In general, $f(x, \dots)$ may or may not be an exact replica of $\hat{f}(x, \dots)$
- If it is, the compression system is called *error free, lossless*, or *information preserving*
- If not, the reconstructed output image is distorted, and the compression system is referred to as *lossy*

Image Compression Models

Functional block diagram of a general image compression system.

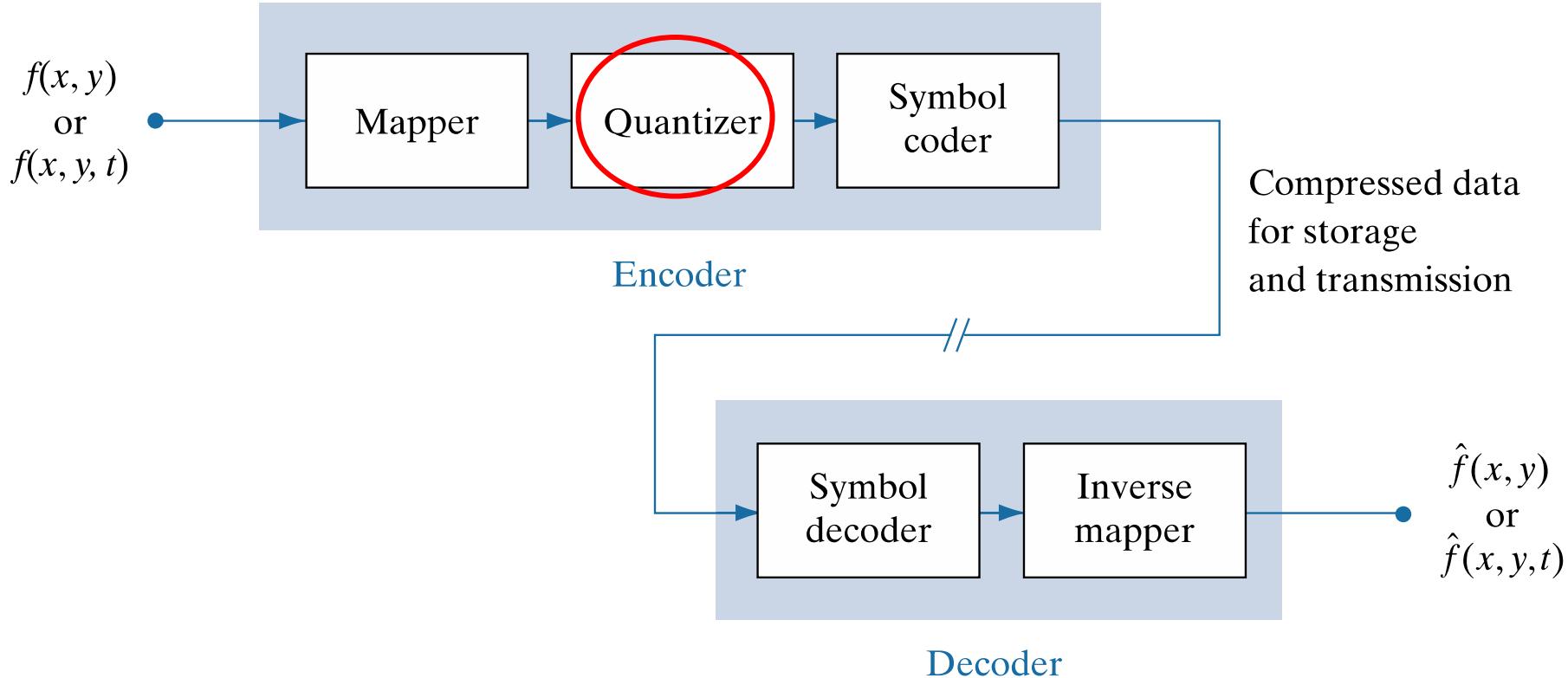


Encoding or Compression Process

- In the first stage of the encoding process, a *mapper* transforms $f(x, \dots)$ into a (usually nonvisual) format **designed to reduce spatial and temporal redundancy**
- This operation generally is reversible, and may or may not directly reduce the amount of data required to represent the image.
- Run-length coding is an example of a mapping that normally yields compression in the first step of the encoding process.
- The mapping of an image into a set of less correlated transform coefficients is an example of the opposite case (the coefficients must be further processed to achieve compression).
- In video applications, the mapper uses previous (and, in some cases, future) video frames to facilitate the removal of temporal redundancy.

Image Compression Models

Functional block diagram of a general image compression system.

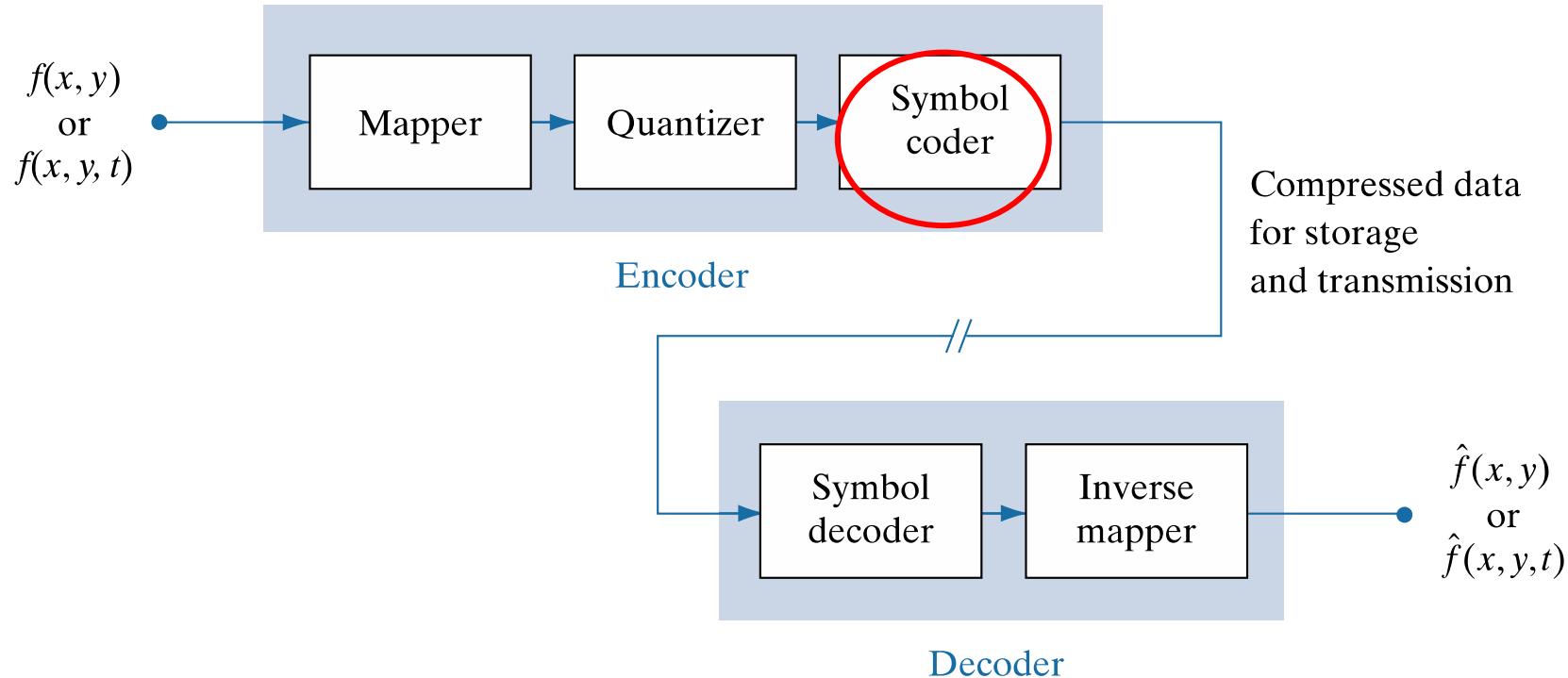


Encoding or Compression Process

- The *quantizer* in the figure 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.
- As noted earlier, this operation is irreversible. It must be omitted when error-free compression is desired.
- In video applications, the *bit rate* (bpp for image) of the encoded output is often measured (in bits/second), and is used to adjust the operation of the quantizer so a predetermined average output rate is maintained.
- Thus, the visual quality of the output can vary from frame to frame as a function of image content.

Image Compression Models

Functional block diagram of a general image compression system.

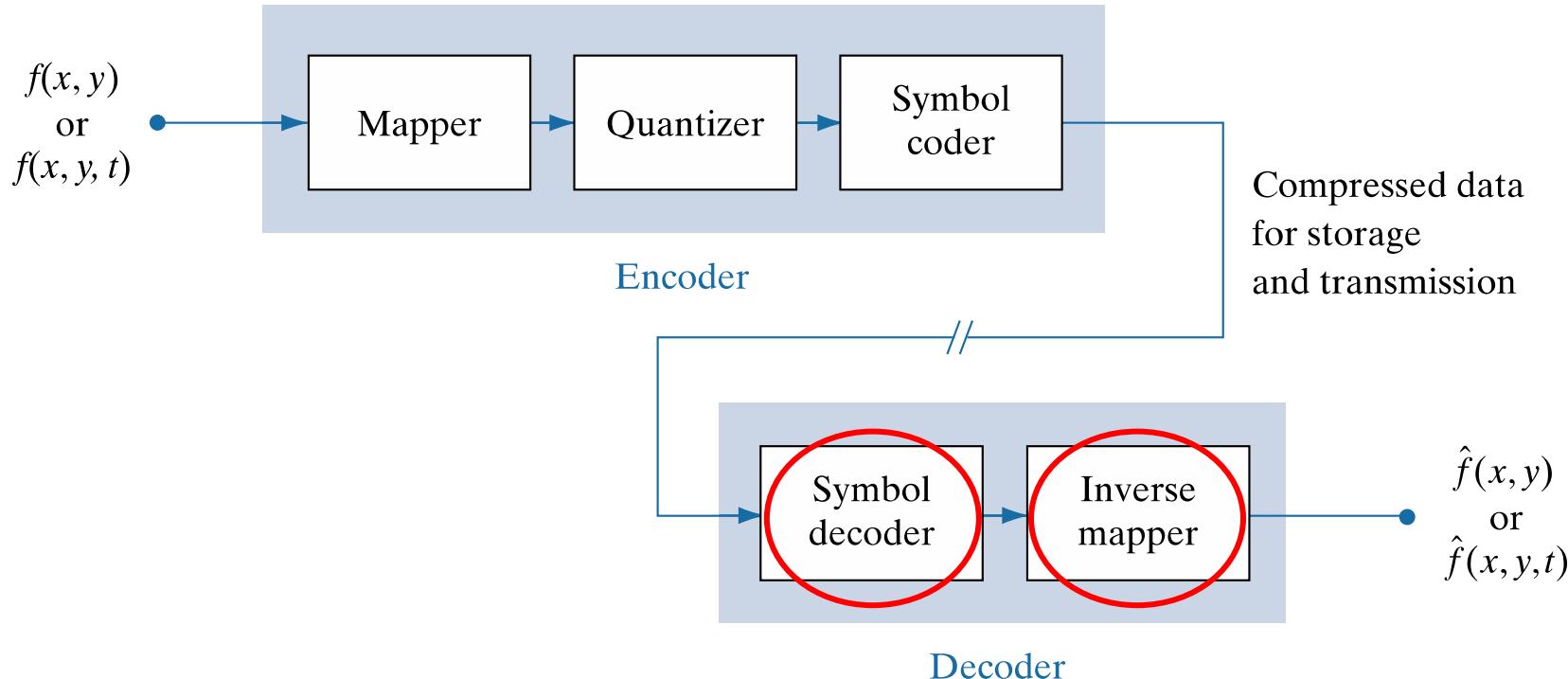


Encoding or Compression Process

- In the third and final stage of the encoding process, the *symbol coder* generates a fixed-length or variable-length code to represent the quantizer output, and maps the output in accordance with the code.
- In many cases a variable-length code is used.
 - The shortest code words are assigned to the most frequently occurring quantizer output values, thus minimizing coding redundancy. This operation is reversible.
 - Upon its completion, the input image has been processed for the removal of each of the three redundancies described earlier.

Image Compression Models

Functional block diagram of a general image compression system.



Decoding or Decompression Process

- The decoder contains only two components: a *symbol decoder* and an *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.
- In video applications, decoded output frames are maintained in an internal frame store (not shown) and used to reinsert the temporal redundancy that was removed at the encoder.

Image format, Container and Compression Standards

- An *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.
- An *image container* is similar to a file format, but handles multiple types of image data.
- Image *compression standards*, on the other hand, define procedures for compressing and decompressing images
 - that is, for reducing the amount of data needed to represent an image.
 - These standards are the underpinning of the widespread acceptance of image compression technology

Image format, Container and Compression Standards

Some popular image compression standards, file formats, and containers.

(Internationally sanctioned entries are shown in blue; all others are in black)

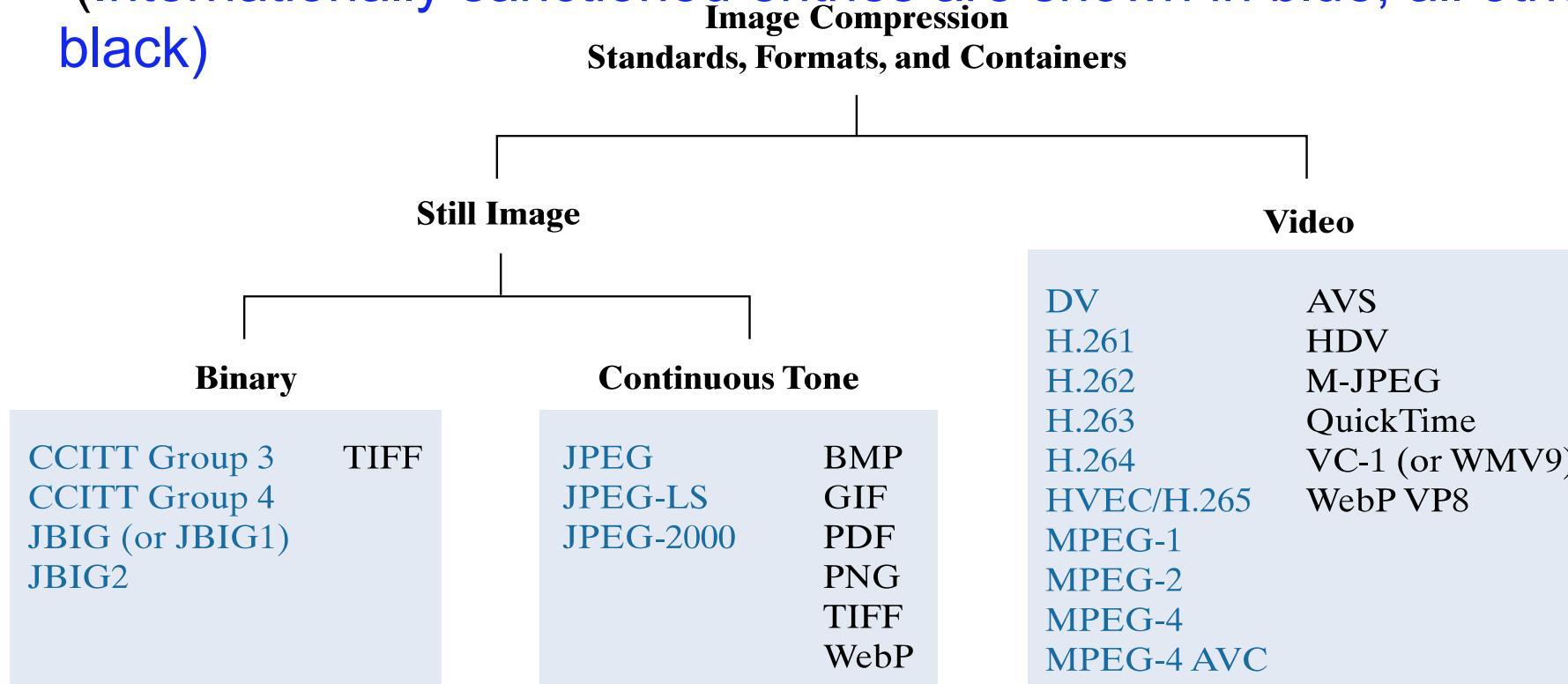
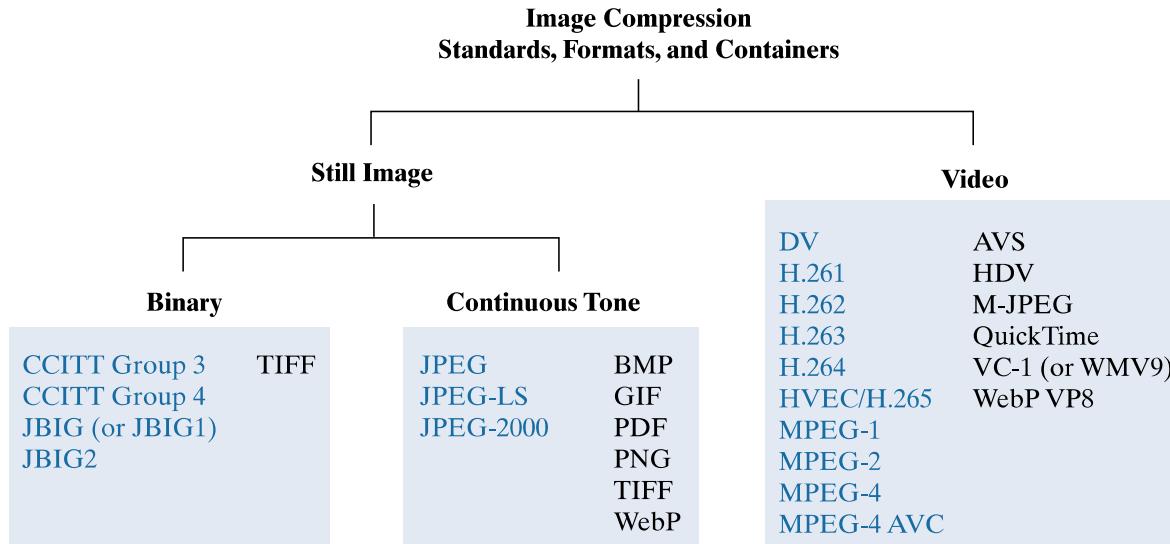


Image format, Container and Compression Standards



Two video compression standards, VC-1 by the *Society of Motion Pictures and Television Engineers* (SMPTE) and AVS by the *Chinese Ministry of Information Industry* (MII), are also included.

Note that they are shown in black, which is used in the Figure to denote entries that are not sanctioned by an international standards organization.

- This figure lists the most important image compression standards, file formats, and containers in use today, grouped by the type of image handled.
- The entries in blue are international standards sanctioned by the *International Standards Organization* (ISO), the *International Electrotechnical Commission* (IEC), and/or the *International Telecommunications Union* (ITU-T)—a *United Nations* (UN) organization that was once called the *Consultative Committee of the International Telephone and Telegraph* (CCITT)

Summary of Image format, Container and Compression Standards

Name	Organization	Description
<i>Bi-Level Still Images</i>		
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 [8.6] and Huffman [8.2] 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 <i>Joint Bi-level Image Experts Group</i> 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 [8.8]. Context-sensitive arithmetic coding [8.4] 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 [8.7] for text and halftone regions, and Huffman [8.2] or arithmetic coding [8.4] for other image content. It can be lossy or lossless.

The focus of these sections is on methods that have proven useful in mainstream binary, continuous-tone still-image, and video compression standards. The standards themselves are used to demonstrate the methods presented.

This table summarizes the standards, formats, and containers listed in Figure earlier.

Responsible organizations, targeted applications, and key compression methods are identified.

Summary of Image format, Container and Compression Standards

Continuous-Tone Still Images

JPEG	ISO/IEC/ ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy baseline coding system (most commonly implemented) uses quantized discrete cosine transforms (DCT) on image blocks [8.9], Huffman [8.2], and run-length [8.6] 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 [8.10], context modeling [8.4], and Golomb coding [8.3].
JPEG- 2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.4] and quantized discrete wavelet transforms (DWT) [8.11] are used. The compression can be lossy or lossless.

Summary of Image format, Container and Compression Standards

Internationally sanctioned video compression standards. The numbers in brackets refer to sections in this chapter

Name	Organization	Description
DV	IEC	<i>Digital Video.</i> A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (<i>integrated services digital network</i>) lines. It supports non-interlaced 352×288 and 176×144 resolution images, called CIF (<i>Common Intermediate Format</i>) and QCIF (<i>Quarter CIF</i>), respectively. A DCT-based compression approach [8.9] similar to JPEG is used, with frame-to-frame prediction differencing [8.10] 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 (<i>Sub-Quarter CIF</i> 128×96), 4CIF (704×576) and 16CIF (1408×512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, streaming, and television. It supports prediction differences within frames [8.10], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.4].
H.265 MPEG-H HEVC	ISO/IEC	<i>High Efficiency Video Coding</i> (HVEC). An extension of H.264 that includes support for macroblock sizes up to 64×64 and additional intraframe prediction modes, both useful in 4K video applications.
MPEG-1	ISO/IEC	A <i>Motion Pictures Expert Group</i> 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 at up 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 [8.10] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 <i>Advanced Video Coding</i> (AVC). Identical to H.264.

Summary of Image format, Container and Compression Standards

Popular image and video compression standards, file formats, and containers not included in Tables 8.3 and 8.4. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
<i>Continuous-Tone Still Images</i>		
BMP	Microsoft	<i>Windows Bitmap</i> . A file format used mainly for simple uncompressed images.
GIF	CompuServe	<i>Graphic Interchange Format</i> . A file format that uses lossless LZW coding [8.5] for 1- through 8-bit images. It is frequently used to make small animations and short low-resolution films for the Internet.
PDF	Adobe Systems	<i>Portable Document Format</i> . 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)	<i>Portable Network Graphics</i> . 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 [8.10].
TIFF	Aldus	<i>Tagged Image File Format</i> . A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
WebP	Google	<i>WebP</i> supports lossy compression via WebP VP8 intraframe video compression (see below) and lossless compression using spatial prediction [8.10] and a variant of LZW backward referencing [8.5] and Huffman entropy coding [8.2]. Transparency is also supported.
<i>Video</i>		
AVS	MII	<i>Audio-Video Standard</i> . Similar to H.264 but uses exponential Golomb coding [8.3]. Developed in China.
HDV	Company consortium	<i>High Definition Video</i> . An extension of DV for HD television that uses compression similar to MPEG-2, including temporal redundancy removal by prediction differencing [8.10].
M-JPEG	Various companies	<i>Motion JPEG</i> . 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	SMPTE	The most used video format on the Internet. Adopted for HD and <i>Blu-ray</i> high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.9 and 8.10] and context-dependent variable-length code tables [8.2], but no predictions within frames.
WMV9	Microsoft	
WebP VP8	Google	A file format based on block transform coding [8.9] prediction differences within frames and between frames [8.10]. The differences are entropy encoded using an adaptive arithmetic coder [8.4].

Lossless Compression: Huffman Coding

Lossless Coding: Compressing data without loss of information

- One of the most popular techniques for removing coding redundancy is due to **Huffman** (**Huffman [1952]**).
- When coding the symbols of an information source individually, **Huffman coding** yields the smallest possible number of code symbols per source symbol.
- In terms of Shannon's first theorem, the resulting code is optimal for a fixed value of n , subject to the constraint that the source symbols be coded *one at a time*.
- In practice, the source symbols may be either the intensities of an image or the output of an intensity mapping operation (pixel differences, run lengths, and so on).

Huffman Coding

Steps in the coding technique:

Step1: To create a series of source reductions by ordering the probabilities of the symbols under consideration, then combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a “compound symbol” with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source also are ordered from the most to the least probable.
- This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Original source	
Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

Summary

- Image Compression Models: Encoder and decoder
- Lossless coding: Huffman codes
 - Encoding process: step 1

Next Session

- Huffman Codes Cont...
- More Lossless coding techniques



PES

UNIVERSITY

CELEBRATING 50 YEARS



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu

+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 6: Image Compression

Prof. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Image Compression Models

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Measuring information
- Average Information: Entropy
- Shannon's theorem for optimum code

This Session

- Image Compression Models
- Lossless coding
 - Huffman Codes

Measuring Image Information

- A random event E with probability $P(E)$ is said to contain $I(E)$ units of information where

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

- Average Information, Entropy of the source is given by

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

- The amount of entropy, and thus information in an image, is far from intuitive

Shannon's First Theorem

- Shannon looked at representing groups of consecutive source symbols with a single code word (rather than one code word per source symbol), and showed that

$$\lim_{n \rightarrow \infty} \left[\frac{L_{\text{avg},n}}{n} \right] = H$$

where $L_{\text{avg},n}$ is the average number of code symbols required to represent all *n*-symbol groups

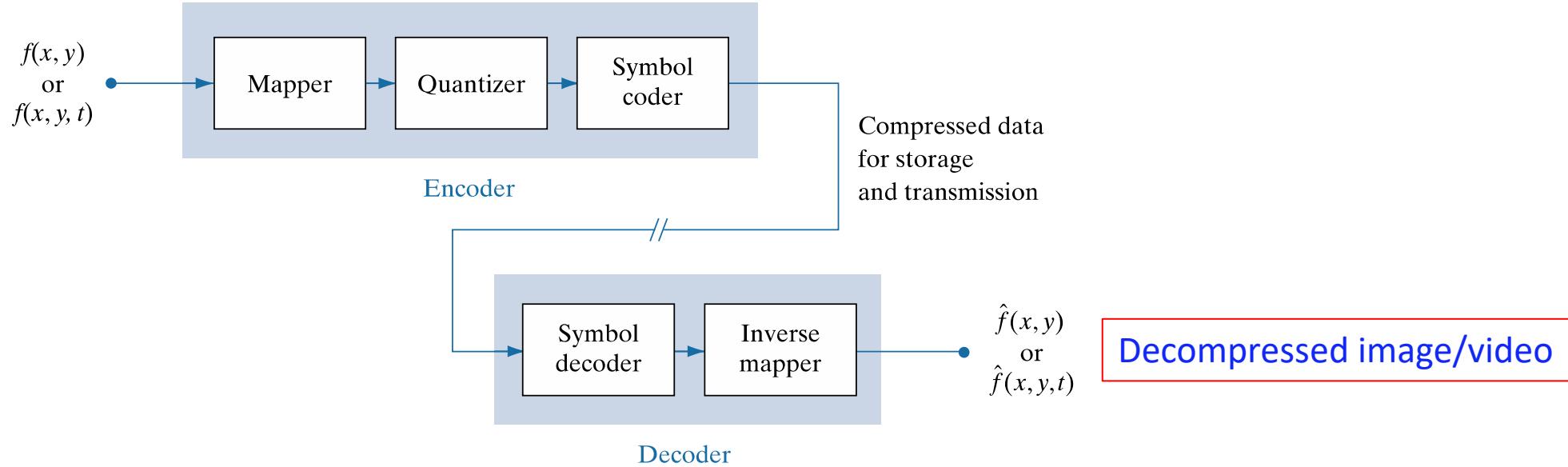
- This equation tells us that $L_{\text{avg},n}/n$ can be made arbitrarily close to H by encoding infinitely long extensions of the single-symbol source.
- That is, it is possible to represent the output of a zero-memory source with an average of H information units per source symbol.

Image Compression Models

- An image compression system is composed of two distinct functional components: an *encoder* and a *decoder*
- The *encoder* performs *compression*, and the decoder performs the complementary operation of *decompression*
- Both operations can be performed in software, as is the case in Web browsers and many commercial image-editing applications, or in a combination of hardware and firmware, as in commercial DVD players
- A *codec* is a device or program that is capable of **both encoding and decoding**

Image Compression Models

Functional block diagram of a general image compression system



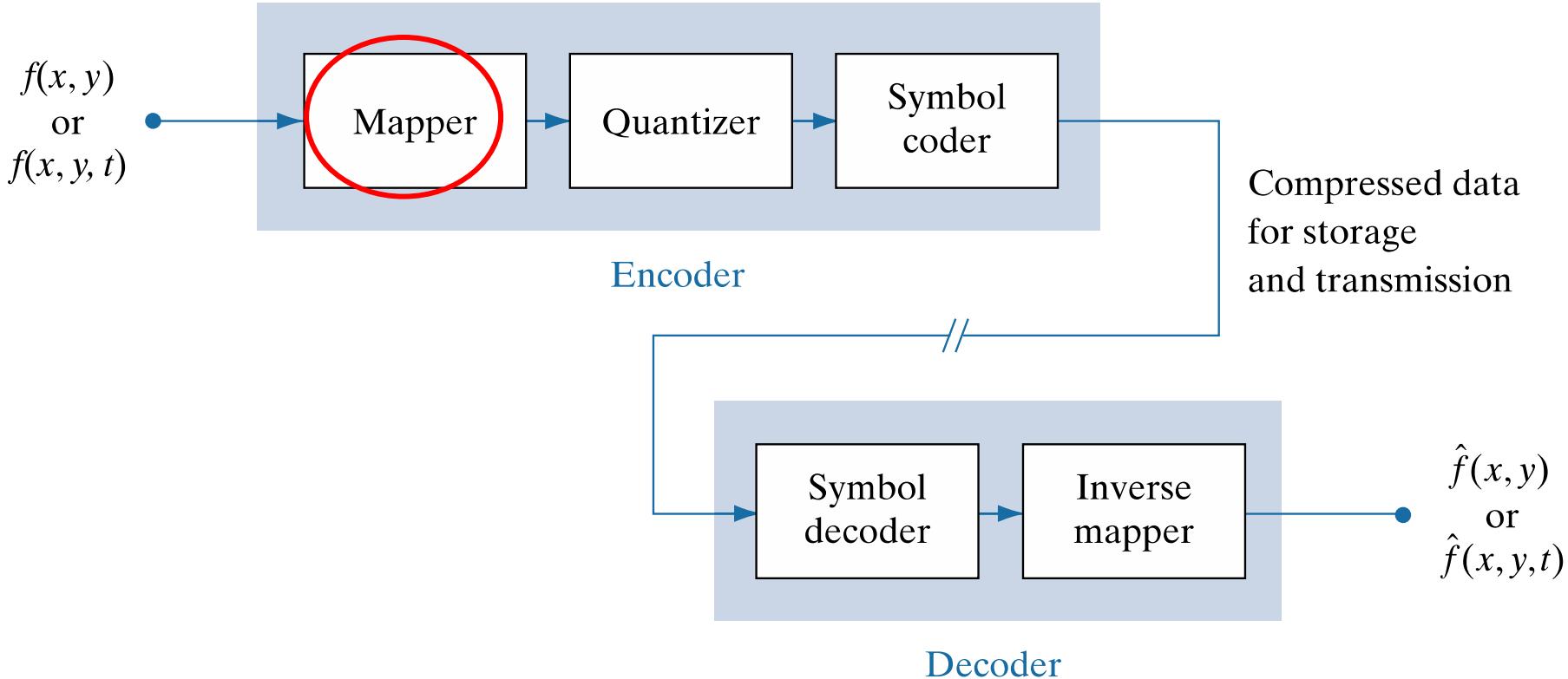
- Input image $f(x, \dots)$ is fed into the encoder, which creates a compressed representation of the input
- This representation is stored for later use, or transmitted for storage and use at a remote location
- When the compressed representation is presented to its complementary decoder, a reconstructed output image $\hat{f}(x, \dots)$ is generated

Image Compression Models

- In still-image applications, the encoded input and decoder output are $f(x, y)$ and $\hat{f}(x, y)$, respectively
- In video applications, they are $f(x, y, t)$ and $\hat{f}(x, y, t)$, where the discrete parameter t specifies time
- In general, $f(x, \dots)$ may or may not be an exact replica of $\hat{f}(x, \dots)$
- If it is, the compression system is called *error free, lossless*, or *information preserving*
- If not, the reconstructed output image is distorted, and the compression system is referred to as *lossy*

Image Compression Models

Functional block diagram of a general image compression system.

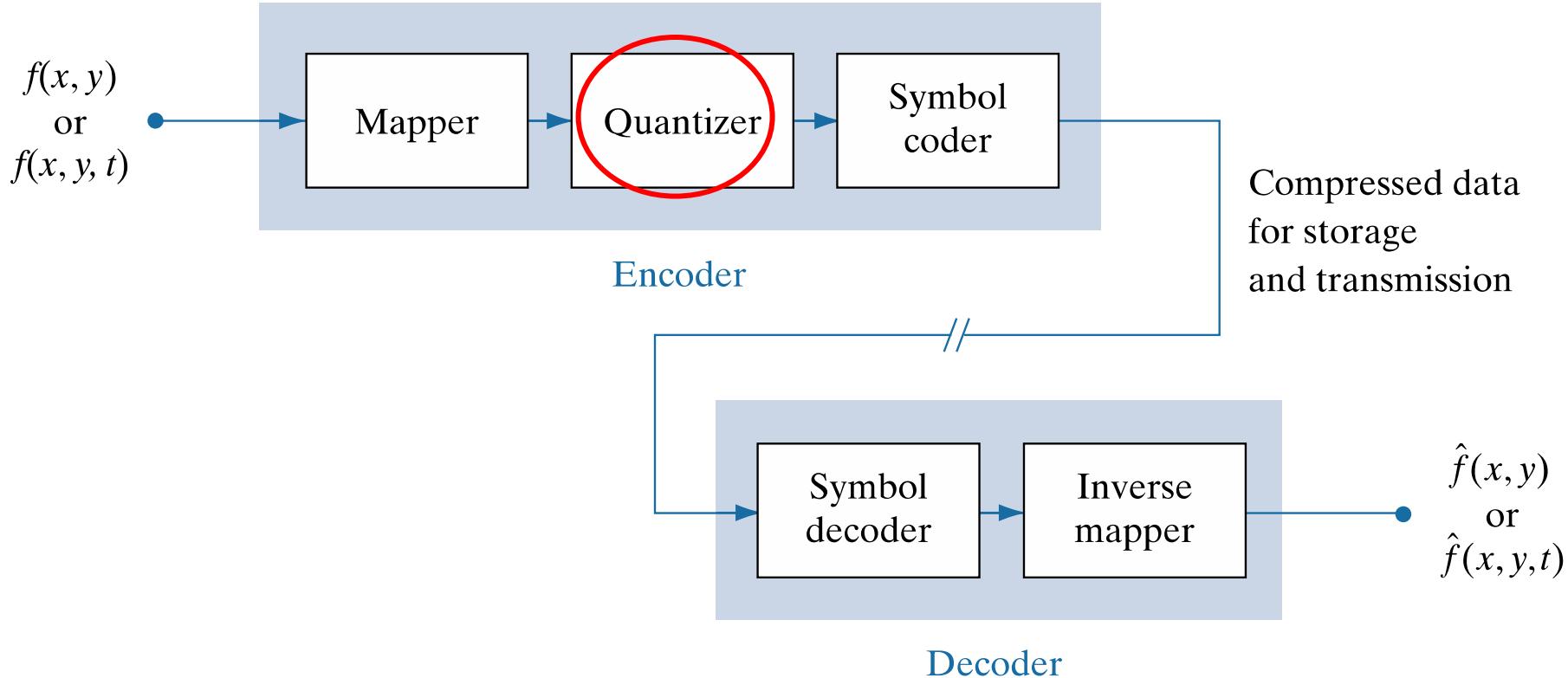


Encoding or Compression Process

- In the first stage of the encoding process, a *mapper* transforms $f(x, \dots)$ into a (usually nonvisual) format **designed to reduce spatial and temporal redundancy**
- This operation generally is reversible, and may or may not directly reduce the amount of data required to represent the image.
- Run-length coding is an example of a mapping that normally yields compression in the first step of the encoding process.
- The mapping of an image into a set of less correlated transform coefficients is an example of the opposite case (the coefficients must be further processed to achieve compression).
- In video applications, the mapper uses previous (and, in some cases, future) video frames to facilitate the removal of temporal redundancy.

Image Compression Models

Functional block diagram of a general image compression system.

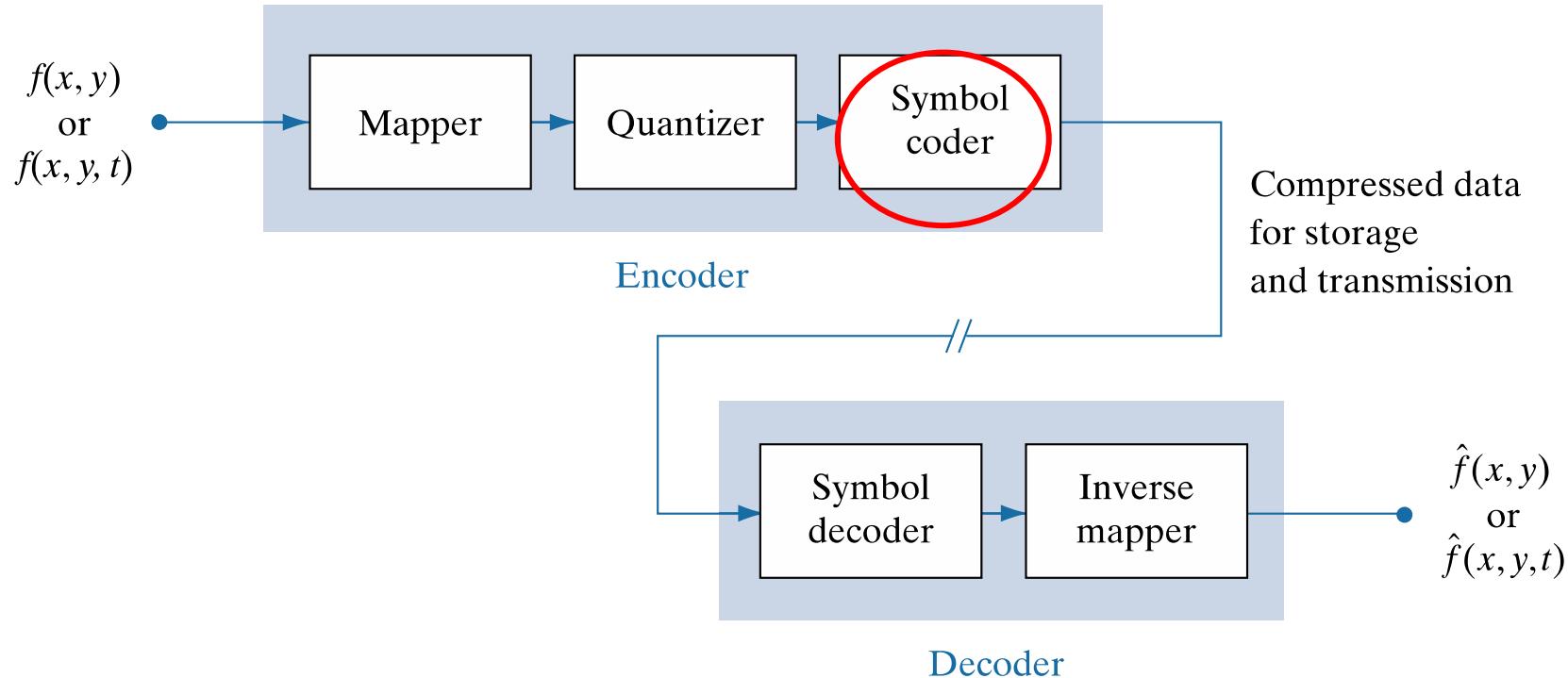


Encoding or Compression Process

- The *quantizer* in the figure 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.
- As noted earlier, this operation is irreversible. It must be omitted when error-free compression is desired.
- In video applications, the *bit rate* (bpp for image) of the encoded output is often measured (in bits/second), and is used to adjust the operation of the quantizer so a predetermined average output rate is maintained.
- Thus, the visual quality of the output can vary from frame to frame as a function of image content.

Image Compression Models

Functional block diagram of a general image compression system.

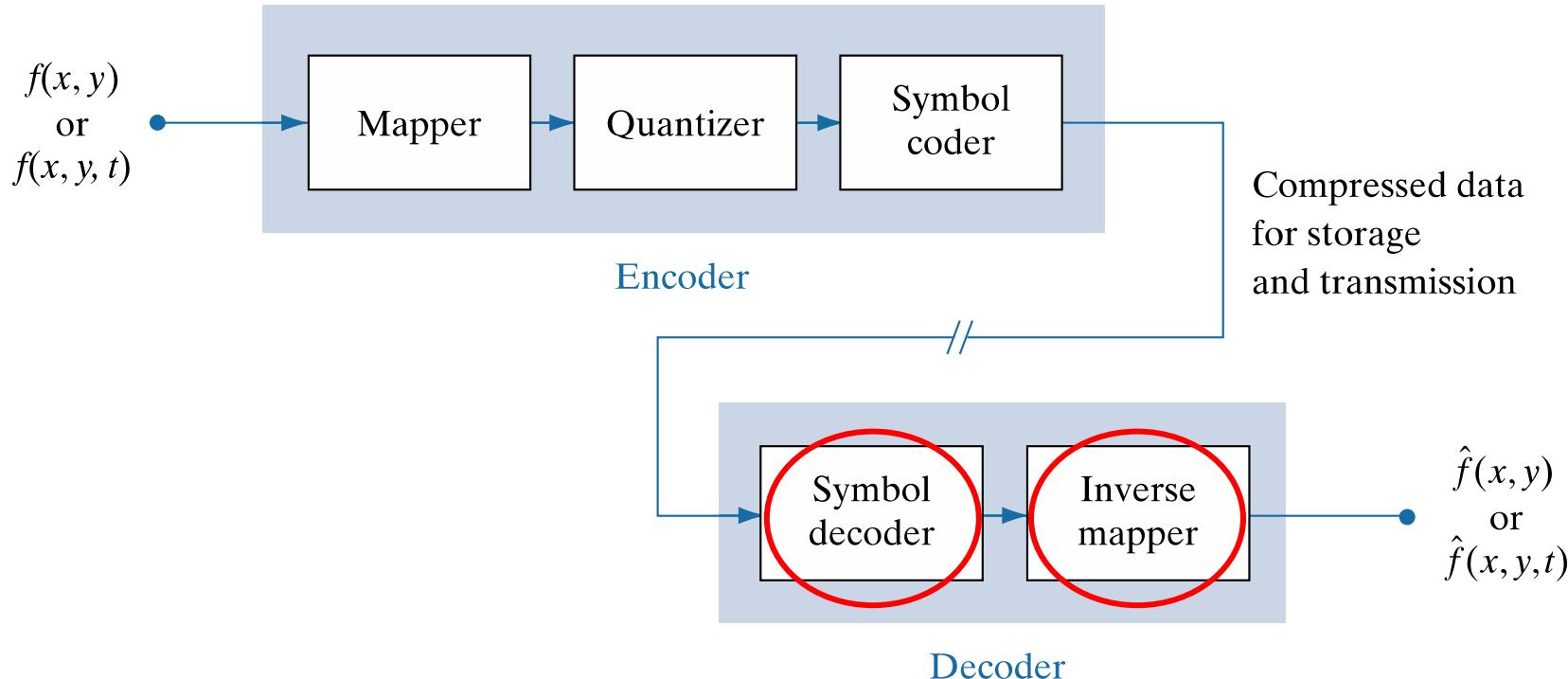


Encoding or Compression Process

- In the third and final stage of the encoding process, the *symbol coder* generates a fixed-length or variable-length code to represent the quantizer output, and maps the output in accordance with the code.
- In many cases a variable-length code is used.
 - The shortest code words are assigned to the most frequently occurring quantizer output values, thus minimizing coding redundancy. This operation is reversible.
 - Upon its completion, the input image has been processed for the removal of each of the three redundancies described earlier.

Image Compression Models

Functional block diagram of a general image compression system.



Decoding or Decompression Process

- The decoder contains only two components: a *symbol decoder* and an *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.
- In video applications, decoded output frames are maintained in an internal frame store (not shown) and used to reinsert the temporal redundancy that was removed at the encoder.

Image format, Container and Compression Standards

- An *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.
- An *image container* is similar to a file format, but handles multiple types of image data.
- Image *compression standards*, on the other hand, define procedures for compressing and decompressing images
 - that is, for reducing the amount of data needed to represent an image.
 - These standards are the underpinning of the widespread acceptance of image compression technology

Image format, Container and Compression Standards

Some popular image compression standards, file formats, and containers.

(Internationally sanctioned entries are shown in blue; all others are in black)

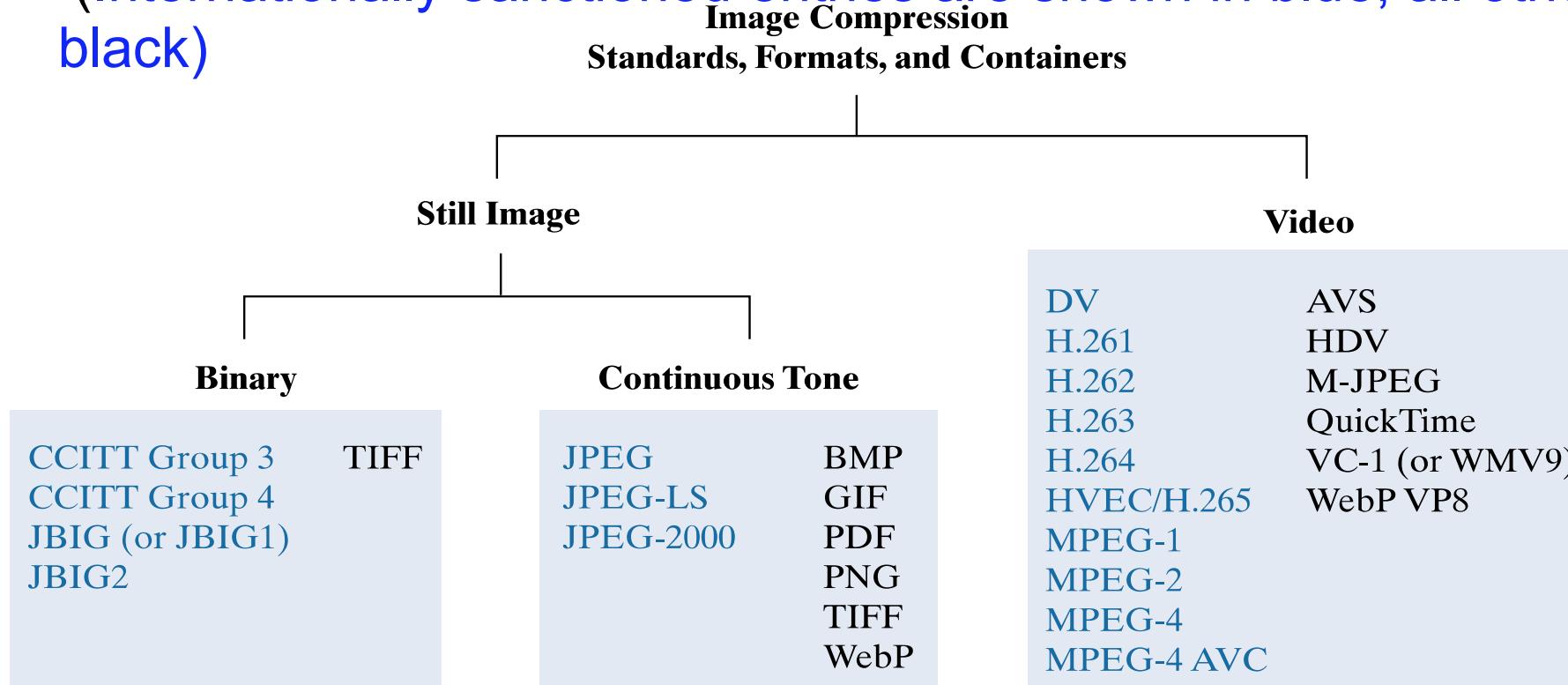
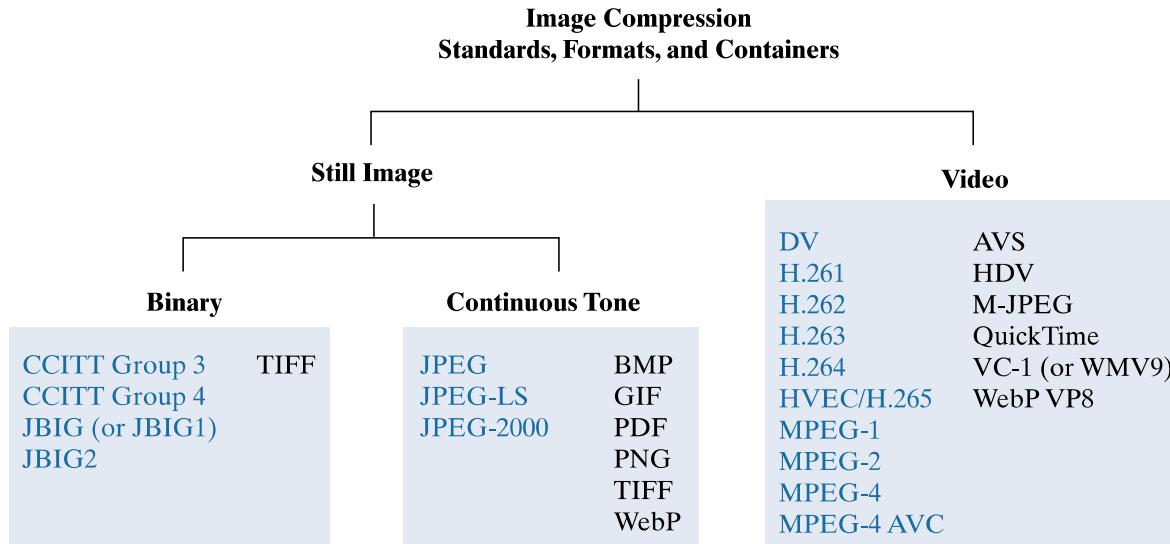


Image format, Container and Compression Standards



Two video compression standards, VC-1 by the *Society of Motion Pictures and Television Engineers* (SMPTE) and AVS by the *Chinese Ministry of Information Industry* (MII), are also included.

Note that they are shown in black, which is used in the Figure to denote entries that are not sanctioned by an international standards organization.

- This figure lists the most important image compression standards, file formats, and containers in use today, grouped by the type of image handled.
- The entries in blue are international standards sanctioned by the *International Standards Organization* (ISO), the *International Electrotechnical Commission* (IEC), and/or the *International Telecommunications Union* (ITU-T)—a *United Nations* (UN) organization that was once called the *Consultative Committee of the International Telephone and Telegraph* (CCITT)

Summary of Image format, Container and Compression Standards

Name	Organization	Description
<i>Bi-Level Still Images</i>		
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 [8.6] and Huffman [8.2] 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 <i>Joint Bi-level Image Experts Group</i> 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 [8.8]. Context-sensitive arithmetic coding [8.4] 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 [8.7] for text and halftone regions, and Huffman [8.2] or arithmetic coding [8.4] for other image content. It can be lossy or lossless.

The focus of these sections is on methods that have proven useful in mainstream binary, continuous-tone still-image, and video compression standards. The standards themselves are used to demonstrate the methods presented.

This table summarizes the standards, formats, and containers listed in Figure earlier.

Responsible organizations, targeted applications, and key compression methods are identified.

Summary of Image format, Container and Compression Standards

Continuous-Tone Still Images

JPEG	ISO/IEC/ ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy baseline coding system (most commonly implemented) uses quantized discrete cosine transforms (DCT) on image blocks [8.9], Huffman [8.2], and run-length [8.6] 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 [8.10], context modeling [8.4], and Golomb coding [8.3].
JPEG- 2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.4] and quantized discrete wavelet transforms (DWT) [8.11] are used. The compression can be lossy or lossless.

Summary of Image format, Container and Compression Standards

Internationally sanctioned video compression standards. The numbers in brackets refer to sections in this chapter

Name	Organization	Description
DV	IEC	<i>Digital Video.</i> A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (<i>integrated services digital network</i>) lines. It supports non-interlaced 352×288 and 176×144 resolution images, called CIF (<i>Common Intermediate Format</i>) and QCIF (<i>Quarter CIF</i>), respectively. A DCT-based compression approach [8.9] similar to JPEG is used, with frame-to-frame prediction differencing [8.10] 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 (<i>Sub-Quarter CIF</i> 128×96), 4CIF (704×576) and 16CIF (1408×512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, streaming, and television. It supports prediction differences within frames [8.10], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.4].
H.265 MPEG-H HEVC	ISO/IEC	<i>High Efficiency Video Coding</i> (HVEC). An extension of H.264 that includes support for macroblock sizes up to 64×64 and additional intraframe prediction modes, both useful in 4K video applications.
MPEG-1	ISO/IEC	A <i>Motion Pictures Expert Group</i> 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 at up 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 [8.10] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 <i>Advanced Video Coding</i> (AVC). Identical to H.264.

Summary of Image format, Container and Compression Standards

Popular image and video compression standards, file formats, and containers not included in Tables 8.3 and 8.4. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
<i>Continuous-Tone Still Images</i>		
BMP	Microsoft	<i>Windows Bitmap</i> . A file format used mainly for simple uncompressed images.
GIF	CompuServe	<i>Graphic Interchange Format</i> . A file format that uses lossless LZW coding [8.5] for 1- through 8-bit images. It is frequently used to make small animations and short low-resolution films for the Internet.
PDF	Adobe Systems	<i>Portable Document Format</i> . 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)	<i>Portable Network Graphics</i> . 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 [8.10].
TIFF	Aldus	<i>Tagged Image File Format</i> . A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
WebP	Google	<i>WebP</i> supports lossy compression via WebP VP8 intraframe video compression (see below) and lossless compression using spatial prediction [8.10] and a variant of LZW backward referencing [8.5] and Huffman entropy coding [8.2]. Transparency is also supported.
<i>Video</i>		
AVS	MII	<i>Audio-Video Standard</i> . Similar to H.264 but uses exponential Golomb coding [8.3]. Developed in China.
HDV	Company consortium	<i>High Definition Video</i> . An extension of DV for HD television that uses compression similar to MPEG-2, including temporal redundancy removal by prediction differencing [8.10].
M-JPEG	Various companies	<i>Motion JPEG</i> . 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	SMPTE	The most used video format on the Internet. Adopted for HD and <i>Blu-ray</i> high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.9 and 8.10] and context-dependent variable-length code tables [8.2], but no predictions within frames.
WMV9	Microsoft	
WebP VP8	Google	A file format based on block transform coding [8.9] prediction differences within frames and between frames [8.10]. The differences are entropy encoded using an adaptive arithmetic coder [8.4].

Lossless Compression: Huffman Coding

Lossless Coding: Compressing data without loss of information

- One of the most popular techniques for removing coding redundancy is due to **Huffman** (**Huffman [1952]**).
- When coding the symbols of an information source individually, **Huffman coding** yields the smallest possible number of code symbols per source symbol.
- In terms of Shannon's first theorem, the resulting code is optimal for a fixed value of n , subject to the constraint that the source symbols be coded *one at a time*.
- In practice, the source symbols may be either the intensities of an image or the output of an intensity mapping operation (pixel differences, run lengths, and so on).

Huffman Coding

Steps in the coding technique:

Step1: To create a series of source reductions by ordering the probabilities of the symbols under consideration, then combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a “compound symbol” with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source also are ordered from the most to the least probable.
- This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Original source	
Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

Summary

- Image Compression Models: Encoder and decoder
- Lossless coding: Huffman codes
 - Encoding process: step 1

Next Session

- Huffman Codes Cont...
- More Lossless coding techniques



PES

UNIVERSITY

CELEBRATING 50 YEARS



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu

+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 7: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Lossless Coding Techniques

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Image Compression Models
- Lossless coding
 - Huffman Codes: Encoder

This Session

- Huffman Codes Cont...
- Arithmetic coding

Lossless Compression: Huffman Coding

Lossless Coding: Compressing data without loss of information

- One of the most popular techniques for removing coding redundancy is due to **Huffman** (**Huffman [1952]**).
- When coding the symbols of an information source individually, **Huffman coding** yields the smallest possible number of code symbols per source symbol.
- In practice, the source symbols may be either the intensities of an image or the output of an intensity mapping operation (pixel differences, run lengths, and so on).

Huffman Coding

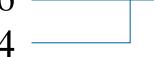
Steps in the coding technique:

Step1: To create a series of source reductions by ordering the probabilities of the symbols under consideration, then combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a “compound symbol” with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source also are ordered from the most to the least probable.
- This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Original source	
Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

Huffman Coding

Steps in the coding technique:

- **Step2:** To code each reduced source, starting with the smallest source and working back to the original source.
 - The minimal length binary code for a two-symbol source, of course, are the symbols 0 and 1.
 - These symbols are assigned to the two symbols on the right. (The assignment is arbitrary; reversing the order of the 0 and 1 would work just as well.)
 - As the reduced source symbol with probability 0.6 was generated by combining two symbols in the reduced source to its left, the 0 used to code it is now assigned to both of these symbols, and a 0 and 1 are arbitrarily appended to each to distinguish them from each other.
 - This operation is then repeated for each reduced source until the original source is reached

Huffman Coding

Original source		Source reduction
Symbol	Probability	
a_2	0.4	
a_6	0.3	
a_1	0.1	
a_4	0.1	
a_3	0.06	
a_5	0.04	

The average length of this code is

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

The entropy of the source is 2.14 bits/symbol

Analysis: 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.
- After the code has been created, coding and/or error-free decoding is accomplished in a simple lookup table manner.
- The code itself is an **instantaneous uniquely decodable block code**.
 - It is called a **block code** because each source symbol is mapped into a fixed sequence of code symbols.
 - It is **instantaneous** because each code word in a string of code symbols can be decoded without referencing succeeding symbols.
 - It is **uniquely decodable** because any string of code symbols can be decoded in only one way.
- Thus, any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in a left-to-right manner.

Decoding of Huffman Code

For the binary code obtained in previous example, decode the encoded string **010100111100**

Decoding:

Original source		
Symbol	Probability	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010-
a_5	0.04	01011-

- First valid code word is 01010, which is the code for symbol a_3 .
- The next valid code is 011, which corresponds to symbol a_1 .
- Continuing in this manner reveals the completely decoded message to be $a_3 \ a_1 \ a_2 \ a_2 \ a_6$

Analysis: Huffman Coding

- When a large number of symbols is to be coded, the construction of an optimal Huffman code is a nontrivial task.
- When source symbol probabilities can be estimated in advance, “near optimal” coding can be achieved with pre-computed Huffman codes.
- Several popular image compression standards, including the JPEG and MPEG standards specify default Huffman coding tables that have been pre-computed based on experimental data.

Arithmetic Coding

- Unlike the variable-length codes, *arithmetic coding* generates nonblock codes.
- In arithmetic coding, which can be traced to the work of Elias (Abramson [1963]), a **one-to-one correspondence between source symbols and code words does not exist.**
- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.
- The code word itself defines an interval of real numbers between 0 and 1.
- At the start of the coding process, the message is assumed to occupy the entire half-open interval $[0, 1)$

Arithmetic Coding

- As the number of symbols in the message increases, the interval used to represent it becomes smaller, and the number of information units (say, bits) required to represent the interval becomes larger.
- Each symbol of the message reduces the size of the interval in accordance with its probability of occurrence.
- Because the technique does not require, as does Huffman's approach, that each source symbol translate into an integral number of code symbols (that is, that the symbols be coded one at a time), it achieves (but only in theory) the bound established by Shannon's first theorem

Arithmetic Coding Examples

Example 1:

Generate an arithmetic code for sequence $a_1 a_2 a_3 a_3 a_4$ given the following symbols and their probabilities.

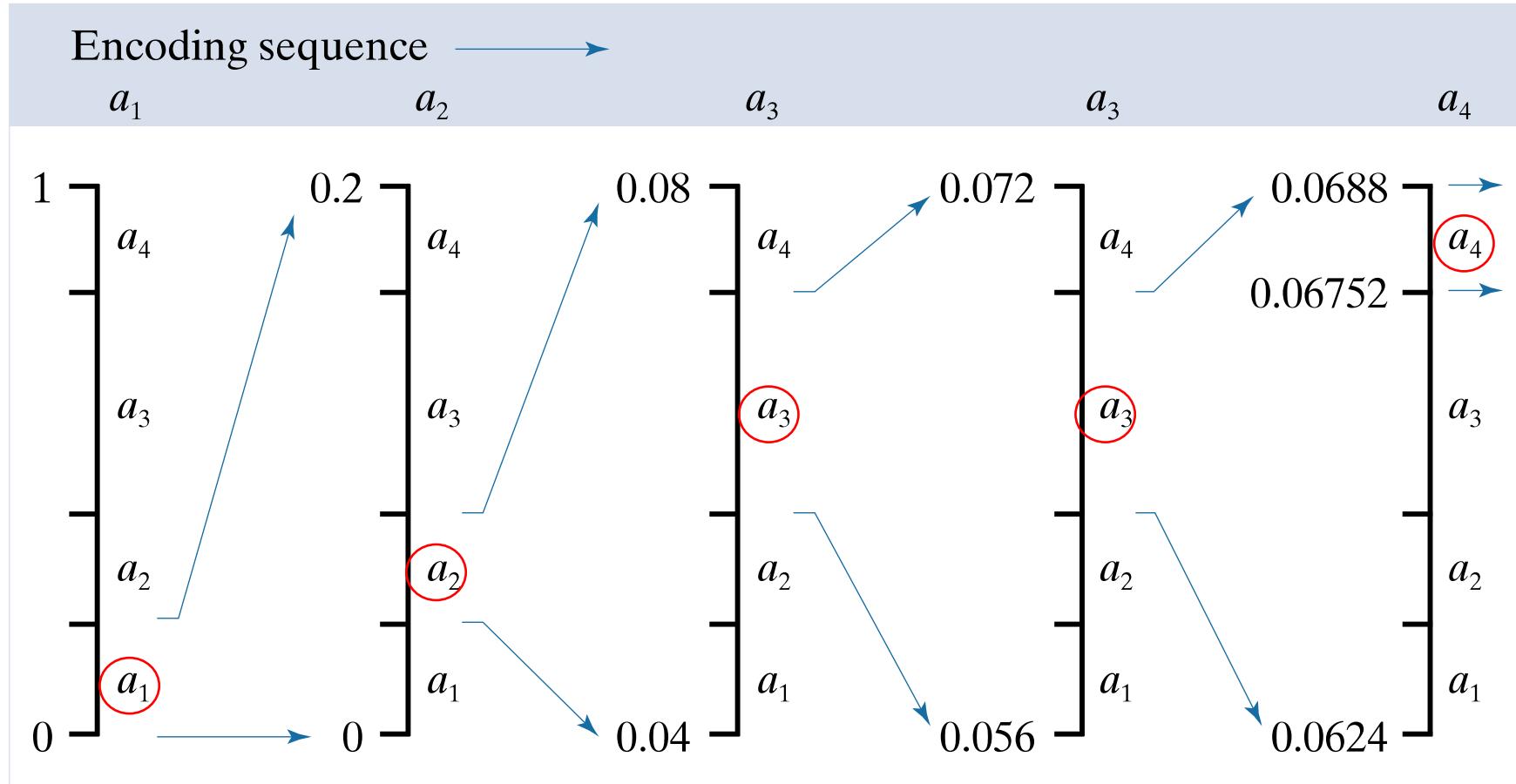
Symbol	Probability	Range
a_1	0.2	[0.0,0.2)
a_2	0.2	[0.2,0.4)
a_3	0.4	[0.4,0.8)
a_4	0.2	[0.8,1)

Soln: $d = \text{Upper bound} - \text{lower bound}$

Upper bound of symbol = lower bound + $d(\text{prob. of that symbol})$

Arithmetic Coding Examples

Coding procedure:



Summary

- Lossless coding techniques:
 - Huffman codes
 - Encoding and decoding technique
 - Arithmetic Codes

Next Session

- Arithmetic Codes Cont..
- LZW coding technique



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012



PES
UNIVERSITY

CELEBRATING 50 YEARS

DIGITAL IMAGE PROCESSING - 2

Lecture 8: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Lossless Coding Techniques

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Lossless coding
 - Huffman Codes: Encoder and decoder
 - Arithmetic codes

This Session

- Arithmetic coding example cont..
 - Decoding example
 - LZW encoder

Arithmetic Coding

- Unlike the variable-length codes, *arithmetic coding* generates non block codes.
- In arithmetic coding, which can be traced to the work of Elias (1963), a **one-to-one correspondence between source symbols and code words does not exist.**
- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.
- The code word itself defines an interval of real numbers between 0 and 1.
- At the start of the coding process, the message is assumed to occupy the entire half-open interval $[0, 1)$

Arithmetic Coding Examples

Example 1:

Generate an arithmetic code for sequence $a_1 a_2 a_3 a_3 a_4$ given the following symbols and their probabilities.

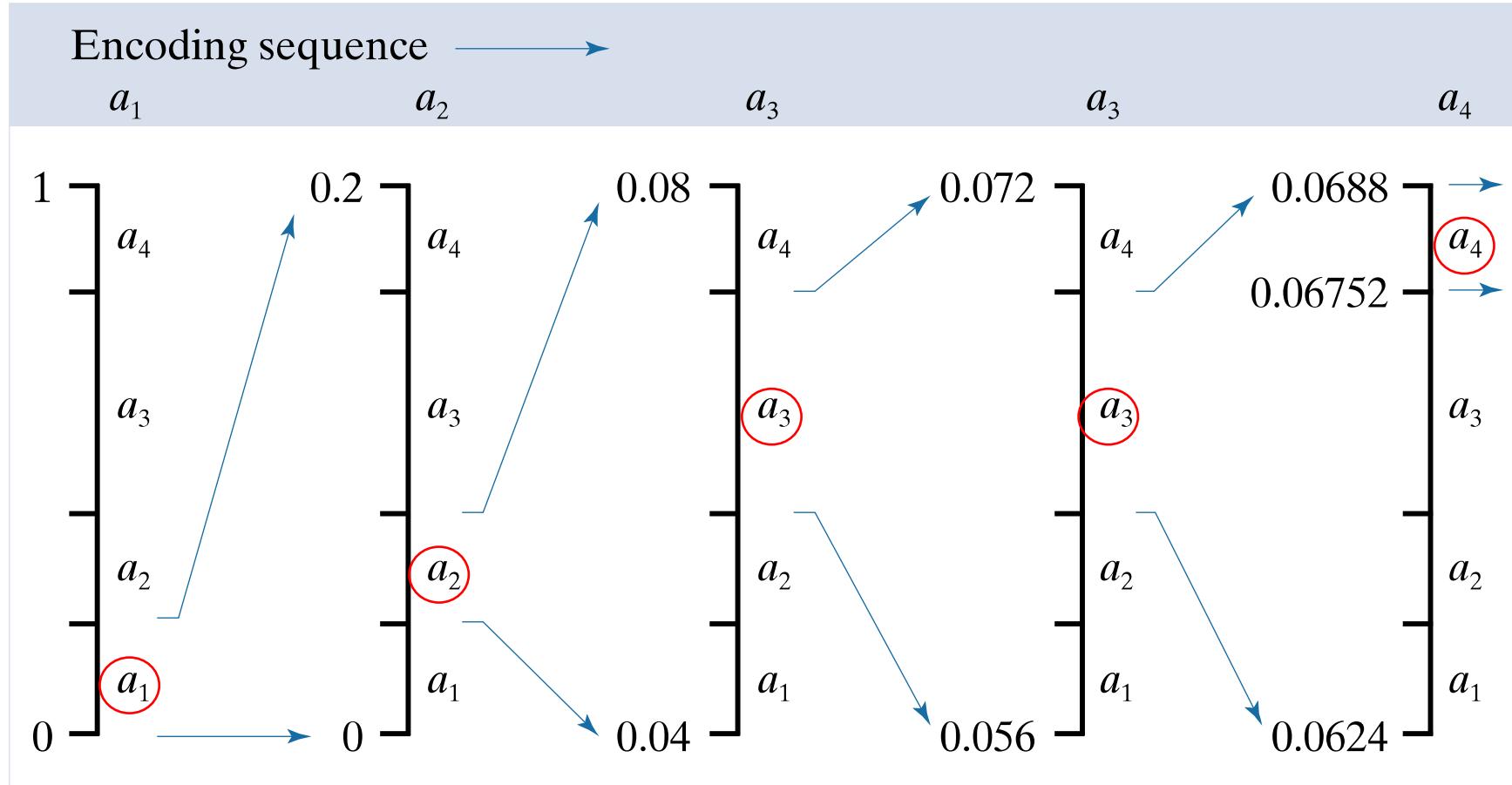
Symbol	Probability	Range
a_1	0.2	[0.0,0.2)
a_2	0.2	[0.2,0.4)
a_3	0.4	[0.4,0.8)
a_4	0.2	[0.8,1)

Soln: $d = \text{Upper bound} - \text{lower bound}$

Upper bound of symbol = lower bound + $d(\text{prob. of that symbol})$

Arithmetic Coding Examples

Coding procedure to code for sequence $a_1 a_2 a_3 a_3 a_4$:



Arithmetic Coding

Arithmetic code for this sequence is:

$0.06756 < \text{Codeword} < 0.0688$

Tag: $(0.06756 + 0.0688) / 2 = 0.0682$

- Any value in this range can be used to represent the message sequence
- **If using 3 decimal digits then 0.068 can be the code**
- Hence 3 decimal digits are used to represent the 5 symbol message
- This translates into 0.6 decimal digits per source symbol(3/5)
- Entropy of the source is 0.58 decimal digits per source symbol
- As the length of the sequence being coded increases, the resulting arithmetic code approaches the bound established by Shannon's first theorem.

Arithmetic Coding Example

Example 2:

Decode the message 0.572 given the following coding model:

Symbol	Probability
C	0.4
E	0.5
.	0.1

Sol: The decoded stream is '**ECE.**'

Drawback of Arithmetic Coding

Two factors cause coding performance to fall short of the bound:

1. The addition of the end-of-message indicator that is needed to separate one message from another
2. The use of finite precision arithmetic.
3. Practical implementations of arithmetic coding address the latter problem by introducing a scaling strategy and a rounding strategy

LZW Coding

- The techniques discussed so far are focused on the removal of coding redundancy.
- We now consider an error-free compression approach that also addresses spatial redundancies in an image.
- The technique, called *Lempel-Ziv- Welch (LZW) coding*, assigns fixed-length code words to variable length sequences of source symbols.
- It is an optimal prefix code
- A key feature of LZW coding is that it requires **no apriori knowledge of the probability of occurrence** of the symbols to be encoded.
 - Despite the fact that until recently it was protected under a United States patent, LZW compression has been integrated into a variety of main-stream imaging file formats, including GIF, TIFF, and PDF.
 - The PNG format was created to get around LZW licensing requirements.

LZW Coding Technique

- LZW coding is conceptually very simple (Welch [1984])
- 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 at locations 1,2,.....,256
- As the encoder sequentially examines image pixels, intensity sequences that are not in the dictionary are placed in algorithmically determined (e.g., the next unused) locations.
 - Shortest subsequence not encountered previously is placed in pre-determined locations

LZW Coding Technique

- LZW coding is accomplished by passing the source data stream into segments that are the Shortest subsequence not encountered previously

Step 1: Form the subsequences → Subsequence generation which results in 'phrases'

Step 2: Code the phrase by writing the location of prefix(in binary) and value of the last bit

Coding Example of LZW Coding

1. Code the given sequence using LZW coding:

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1.....

Assume binary sequence 0,1 are already stored in memory in numerical position (say 1,2...)

Summary

- Arithmetic Coding example
 - Encoder and Decoder
 - LZW encoding technique and examples

Next Session

- LZW encoding technique and examples
- Run Length coding
- Lossy compression techniques



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012



DIGITAL IMAGE PROCESSING - 2

Lecture 9: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Lossless Coding Techniques

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

- Arithmetic coding example cont..
 - Decoding example
- LZW encoder

This Session

- LZW encoder Cont..
- Run Length Encoding
- Bit-plane Coding

LZW Coding

- *Lempel-Ziv- Welch (LZW) coding* is an error-free/lossless compression approach that also addresses spatial redundancies in an image.
- The technique, assigns fixed-length code words to variable length sequences of source symbols.
- It is an optimal prefix code

LZW Coding Technique

- LZW coding is accomplished by passing the source data stream into segments that are the Shortest subsequence not encountered previously

Step 1: Form the subsequences → Subsequence generation which results in 'phrases'

Step 2: Code the phrase by writing the location of prefix(in binary) and value of the last bit

Coding Example of LZW Coding

1. Code the given sequence using LZW coding:

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1.....

Assume binary sequence 0,1 are already stored in memory in numerical position (say 1,2...)

2. Code the given sequence using LZW coding:

A A B A B B B A B A A B A B B B A B B A B B.....

Assume A and B are stored in locations 1 and 2

A → 0

B → 1

Analysis of LZW Coding

- Clearly, the size of the dictionary is an important system parameter
- If it is too small, the detection of matching intensity level sequences will be less likely
- If it is too large, the size of the code words will adversely affect compression performance
- A unique feature of the LZW coding is that the coding dictionary or code book is created while the data are being encoded.
- Remarkably, an LZW decoder builds an identical decompression dictionary as it simultaneously decodes the encoded data stream

Run-Length Coding (RLE)

- Images with **repeating intensities** along their rows (or columns) can often be compressed by representing runs of identical intensities as *run-length pairs*
 - where each run-length pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity.
- *Run-length encoding* (RLE), was developed in the 1950s and became, along with its 2-D extensions, the standard compression approach in facsimile (FAX) coding.
- Compression is achieved by eliminating a simple form of spatial redundancy—groups of identical intensities.
- When there are few (or no) runs of identical pixels, run-length encoding results in data expansion.

Run-Length Coding (RLE)

- Run-length encoding is particularly effective when compressing binary images.
 - Because there are only two possible intensities (black and white), adjacent pixels are more likely to be identical.
 - In addition, each image row can be represented by a sequence of lengths only, rather than length-intensity pairs.
 - The basic idea is to code each contiguous group (i.e., run) of 0's or 1's encountered in a left-to-right scan of a row by its length *and* to establish a convention for determining the value of the run.
 - The most common conventions are (1) to specify the value of the first run of each row, or (2) to assume that each row begins with a white run, whose run length may in fact be zero.

Run-Length Coding (RLE)

- Although run-length encoding is in itself an effective method of compressing binary images, additional compression can be achieved by variable-length coding the run lengths themselves.
- The black and white run lengths can be coded separately using variable-length codes that are specifically tailored to their own statistics.

Bit Plane Coding

- The previous techniques can be applied to images with more than two intensities by individually processing their bit planes.
- *Bit-plane coding*, is based on the concept of decomposing a multilevel (monochrome or color) image into a series of binary images and compressing each binary image via one of several well-known binary compression methods
- The intensities of an m -bit monochrome image can be represented in the form of the base-2 polynomial

$$a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

Bit Plane Coding

$$a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

- Based on this property, a simple method of decomposing the image into a collection of binary images is to separate the m coefficients of the polynomial into m 1-bit bit planes.
- The lowest-order bit plane (the plane corresponding to the least significant bit) is generated by collecting the a_0 bits of each pixel, while the highest-order bit plane contains the a_{m-1} bits or coefficients.
- In general, each bit plane is constructed by setting its pixels equal to the values of the appropriate bits or polynomial coefficients from each pixel in the original image.

Bit Plane Coding

- The inherent disadvantage of this decomposition approach is that small changes in intensity can have a significant impact on the complexity of the bit planes.
- If a pixel of intensity 127 (01111111) is adjacent to a pixel of intensity 128 (10000000), for instance, every bit plane will contain a corresponding 0 to 1 (or 1 to 0) transition.
- For example, because the most significant bits of the binary codes for 127 and 128 are different, the highest bit plane will contain a zero-valued pixel next to a pixel of value 1, creating a 0 to 1 (or 1 to 0) transition at that point.

Bit Plane Coding

- An alternative decomposition approach (which reduces the effect of small intensity variations) is to first represent the image by an m -bit *Gray code*.
- The m -bit Gray code $g_{m-1} \dots g_2g_1g_0$ that corresponds to the earlier binary polynomial can be computed from

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

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

Here, \oplus denotes the exclusive OR operation.

Bit Plane Coding

- This code has the unique property that successive code words differ in only one bit position.
- Thus, small changes in intensity are less likely to affect all m bit planes.
- For instance, when intensity levels 127 and 128 are adjacent, only the highest-order bit plane will contain a 0 to 1 transition, because the Gray codes that correspond to 127 and 128 are 01000000 and 11000000, respectively.

Next Session

- Lossy compression techniques
 - Block Transform Coding



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012



PES
UNIVERSITY

CELEBRATING 50 YEARS

DIGITAL IMAGE PROCESSING - 2

Lecture 10: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Lossy Coding Techniques

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

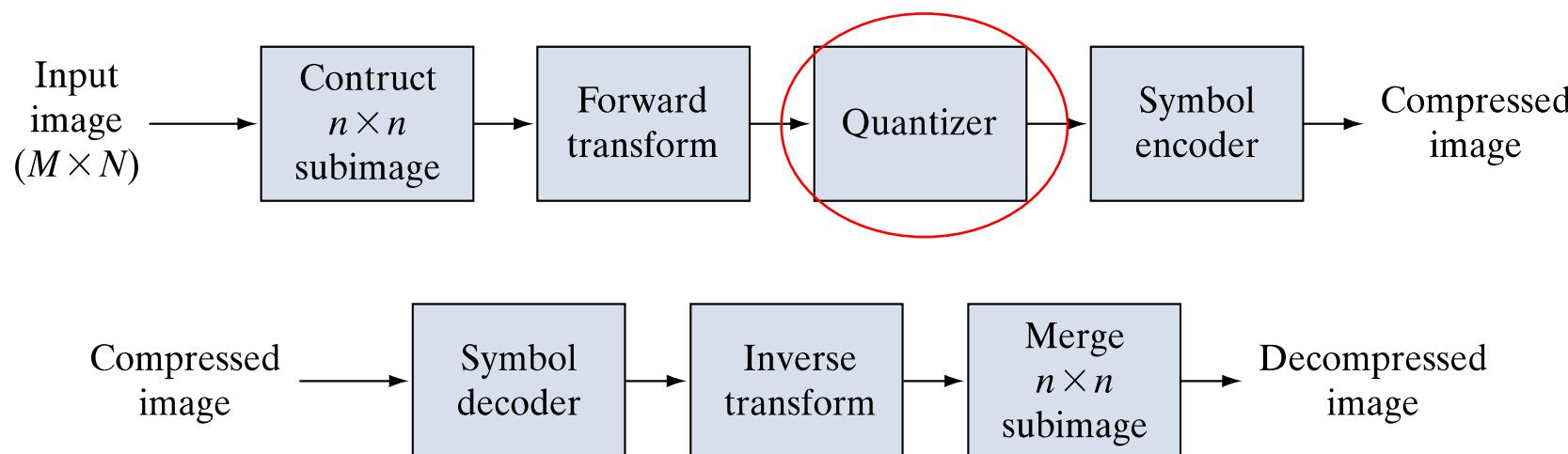
- LZW encoder Cont..
- Run Length Encoding
- Bit-plane Coding

This Session

- Lossy compression techniques
 - Block Transform Coding

Transform Coding

- *Transform Coding* is a lossy compression technique where some information loss is acceptable
- Block diagram of a transform coder:



Transform Coding

- This is a compression technique that divides an image into small non-overlapping blocks of equal size (e.g., 8 x 8) and processes the blocks independently using a 2-D transform.
- In *block transform coding*, a reversible, linear transform (such as the Fourier transform) is used to map each block or subimage 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 (or discarded entirely) with little image distortion
- Variety of transforms like DFT, DCT, DWT can be used for compression

Transform Coding

- An $M \times N$ input image is subdivided first into subimages of size $n \times n$,
- These are then transformed to generate MN/n^2 subimage transform arrays, each of size $n \times n$.
- The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients.
- The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least amount of information in a predefined sense.

Transform Coding

- These coefficients have the smallest impact on reconstructed subimage quality.
- The encoding process terminates by coding (normally using a variable-length code) the quantized coefficients.
- Any or all of the transform encoding steps can be adapted to local image content, called *adaptive transform coding*, or fixed for all subimages, called *nonadaptive transform coding*.
- The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and the computational resources available.
- Compression is achieved during the quantization of the transformed coefficients (not during the transformation step)

Test Images

Some popular test images:



Lena



Barbara



Cameraman



Goldhill

Transform Coding Example

Consider the uncompressed 512 x 512, 8-bit image of Lena



- Now divide the original image into subimages of size 8×8 , representing each subimage using three of the transforms DFT,WHT and DCT
- Truncate 50% of the resulting coefficients, and take the inverse transform of the truncated coefficient arrays.

Transform Coding Example

- In each case, the 32 retained coefficients were selected on the basis of maximum magnitude.



Original



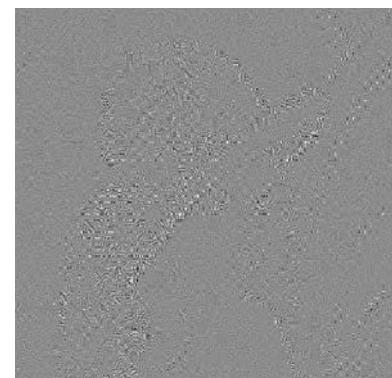
Using DFT



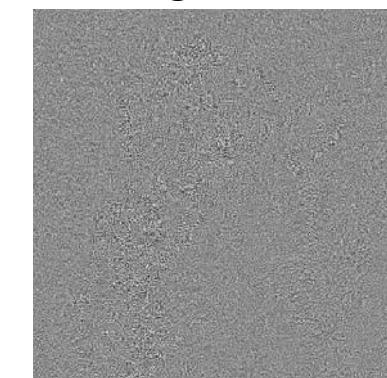
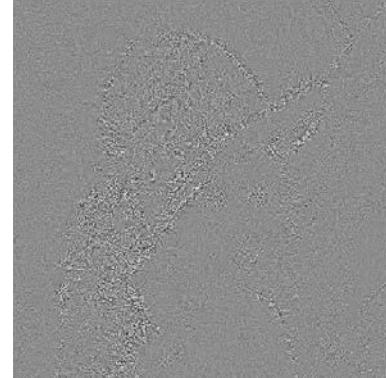
Using WHT



Using DCT



Scaled error



Next Session

- Lossy Compression Cont..
 - Block Transform Coding Cont..
 - Image Morphology



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012

DIGITAL IMAGE PROCESSING - 2

Lecture 11: Image Compression

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

DIGITAL IMAGE PROCESSING - 2

Lossy Coding Techniques

Dr. Shruthi M L J

Department of Electronics & Communication Engineering

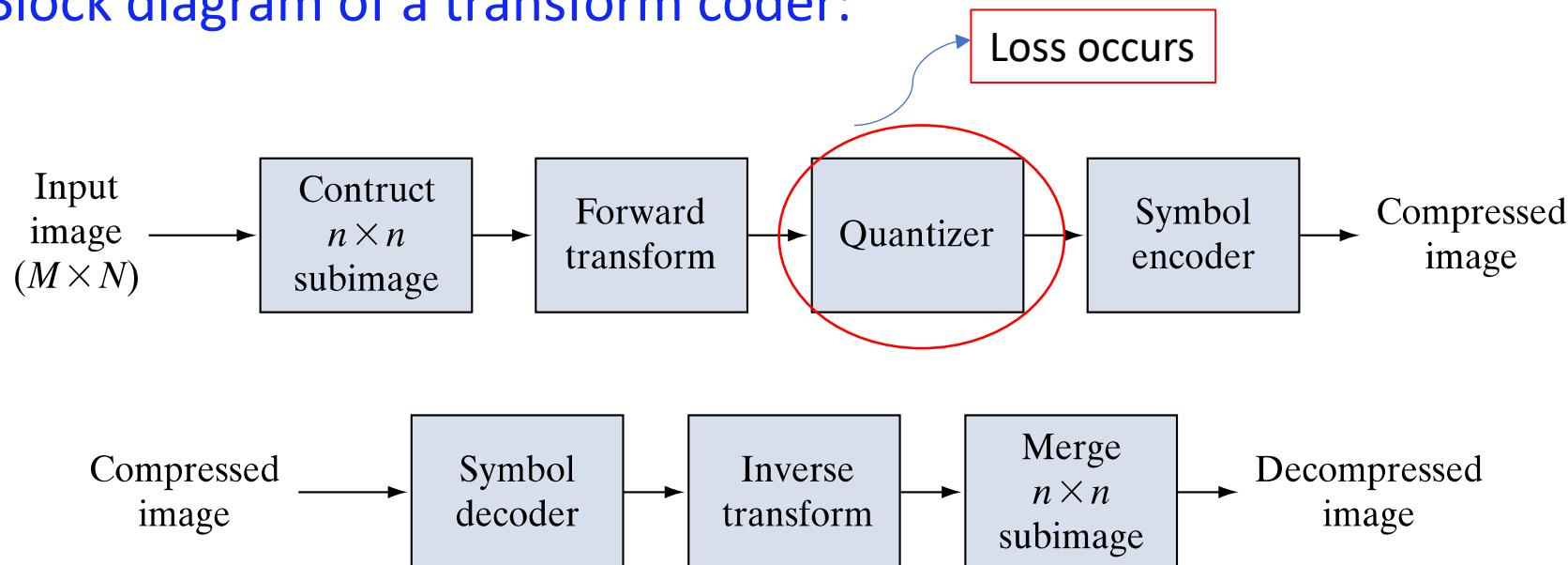
- Lossy compression techniques
 - Block Transform Coding

This Session

- Lossy compression techniques
 - Block Transform Coding Cont..

Transform Coding

- *Transform Coding* is a lossy compression technique where some information loss is acceptable
- Block diagram of a transform coder:



Transform Coding Example

Consider the uncompressed 512×512 , 8-bit image of Lena



- Now divide the original image into subimages of size 8×8 , representing each subimage using three of the transforms
DFT, WHT and DCT
- Truncate 50% of the resulting coefficients, and take the inverse transform of the truncated coefficient arrays

Transform Coding Example

- In each case, the 32 retained coefficients were selected on the basis of maximum magnitude.



Original



Using DFT



Using WHT



Using DCT

Scaled error

Transform Coding Example analysis

- In all cases, the 32 discarded coefficients had little visual impact on the quality of the reconstructed image.
- Their elimination, however, was accompanied by some mean-squared error, which can be seen in the scaled error images
- The actual rms errors were 2.32, 1.78, and 1.13 intensities, respectively.
- The small differences in mean-squared reconstruction error noted in the example are related directly to the **energy or information packing properties of the transforms employed**

- **Inference: Information packing ability of the DCT is superior to that of the DFT and WHT.**

Comparison of Transforms for Compression

- Although this condition usually holds for most images, the Karhunen-Loëve transform (**KL transform**), not the DCT, is the optimal transform in an information packing sense
- This is due to the fact that the KLT minimizes the mean-squared error for any input image and any number of retained coefficients
- However, because the KLT is data dependent, obtaining the KLT basis images for each subimage, in general, is a nontrivial computational task
- **For this reason, the KLT is used infrequently for image compression.**
 - **Instead, a transform, such as the DFT, WHT, or DCT, whose basis images are fixed (input independent) is normally used**

Comparison of Transforms for Compression

- Of the possible input independent transforms, the **nonsinusoidal transforms (such as the WHT transform)** are the simplest to implement
- The **sinusoidal transforms (such as the DFT or DCT)** more closely approximate the information packing ability of the optimal KLT.
- **Hence, most transform coding systems are based on the DCT, which provides a good compromise between information packing ability and computational complexity**

Comparison of Transforms for Compression

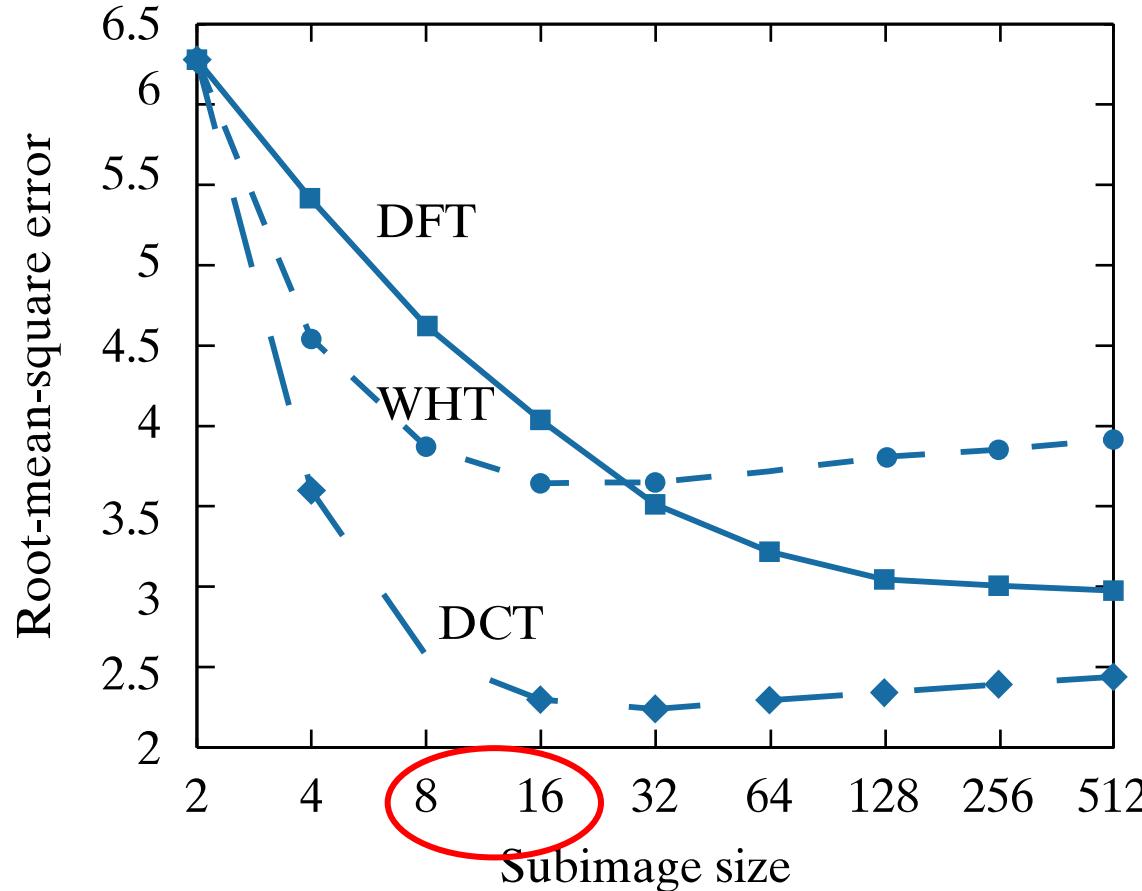
- Properties of the DCT have proved to be of such practical value that the DCT is an international standard for transform coding systems(JPEG, MPEG...)
- Compared to the other input independent transforms, it has the advantages of having been implemented in a single integrated circuit, packing the most information into the fewest coefficients (for most images), and minimizing the block-like appearance, called *blocking artifacts*, that results when the boundaries between subimages become visible

Size of Subimage

- Another significant factor affecting transform coding error and computational complexity is **subimage size**
- In most applications, images are subdivided so the correlation (redundancy) between adjacent subimages is reduced to some acceptable level and so n is an integer power of 2 where, as before, n is the subimage dimension
- The latter condition simplifies the computation of the subimage transforms.
- In general, both the level of compression and computational complexity increase as the subimage size increases.
- The most popular subimage sizes are 8×8 and 16×16

Size of Subimage

Reconstruction error versus subimage size



Size of Subimage

- The transform of each subimage is computed. 75% of the resulting coefficients are truncated, and the inverse transform of the truncated arrays is computed.
- The Hadamard and cosine curves flatten as the size of the subimage becomes greater than 8×8 , whereas the Fourier reconstruction error continues to decrease in this region.
- As n further increases, the Fourier reconstruction error crosses the Walsh-Hadamard curve and approaches the cosine result.
- This result is consistent with the theoretical and experimental findings reported by Netravali and Limb [1980] and by Pratt [2001] for a 2-D Markov image source.

Example

Consider compression and reconstruction of the following 8×8 subimage with the JPEG baseline standard:

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

The original image consists of 256 or 2^8 possible intensities (grayscale), so the coding process begins by level shifting the pixels of the original subimage by 2^7 or 128 intensity levels (-128 to +127). The resulting shifted array is

-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

When transformed in accordance with the forward DCT for $n = 8$
it becomes

-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

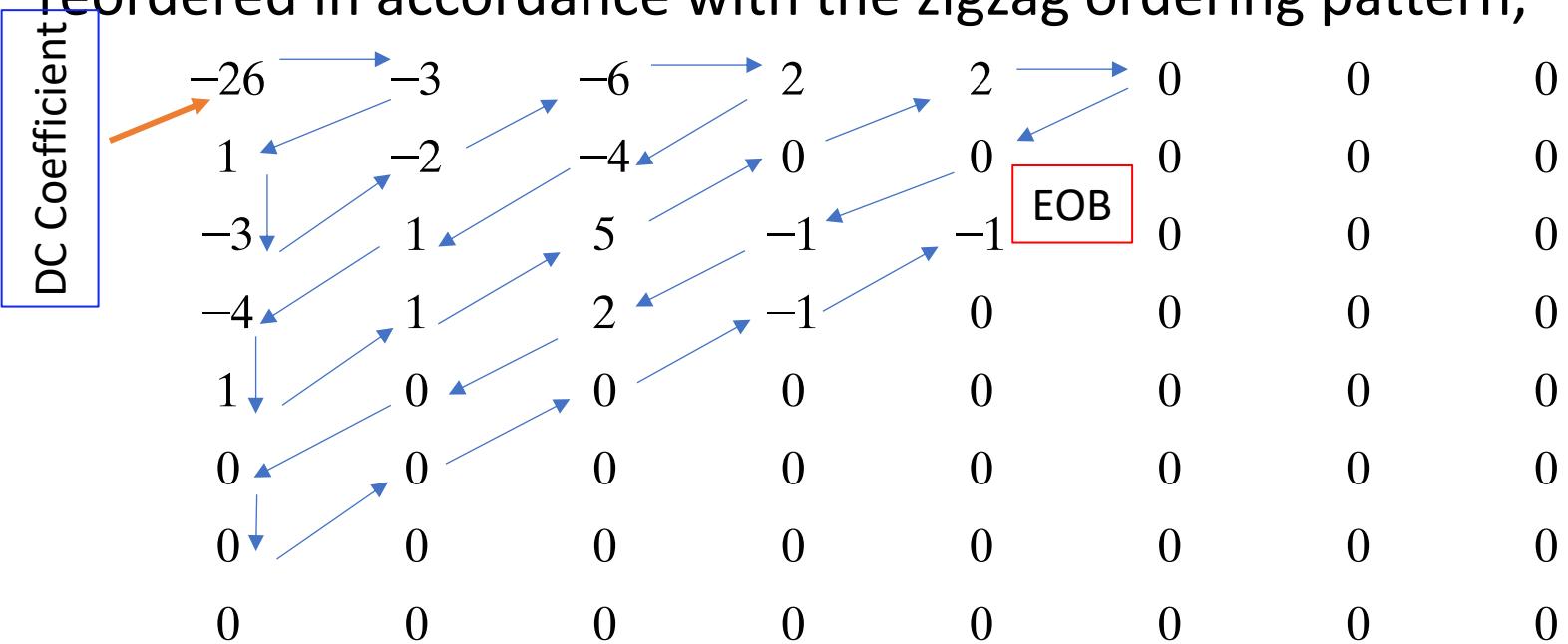
Example

If the JPEG normalization array is used to quantize the transformed array, the scaled and truncated coefficients are

DC Coefficient	-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

The transformation and normalization process produces a large number of zero-valued coefficients. The coefficients are reordered in accordance with the zigzag ordering pattern,



Example

The resulting 1-D coefficient sequence is

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

End of Block

Any coding technique can be applied to this stream to code it and then the reverse process decompresses the image to its original size

Next Session

- Morphological Image Processing



THANK YOU

Dr. Shruthi M L J

Department of Electronics &
Communication Engineering

shruthimlj@pes.edu
+91 8147883012