Parul® University

Image Processing (203105470)

Pragya Devi

Assistant Professor
Computer science and Engineering

UNIT-5

Image Compression

Outline

1. Fundamental of Image Compression
2. Image Compression Models
3. Element of Information Theory
4. Error-Free Compression
5. Lossy Compression
6. Image Compression Standard

Outcome of this unit:

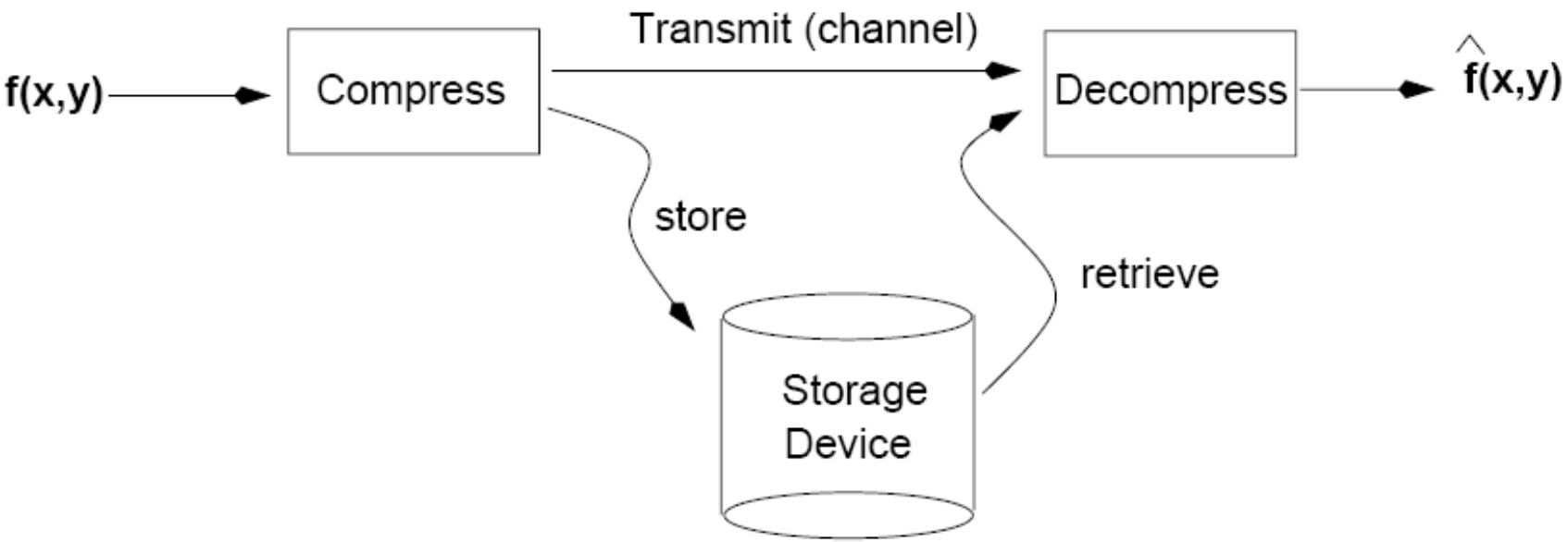
After completing the students will be able to:

- To learn Fundamental of Image Compression.
- To understand image compression Model.
- To know Elements of Information theory.
- to know Image compression standard.

1. Fundamental of Image compression

Image compression refers to the process of reducing the amount of data required to represent a given quantity of information.

Important for reducing storage requirements and improving transmission rates



Data redundancy

Data redundancy refers to the duplication of data within a database or information system. It occurs when the same piece of data is stored in multiple locations

$$R = 1 - 1/\{CR\}$$

Where:

- R is the relative data redundancy.
- CR is the compression ratio

Compression Ratio

$$C_R = \frac{n_1}{n_2}$$

Example

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then C_R=1, R_D=0

if $n_2 \ll n_1$, then $C_R \to \infty$, $R_D \to 1$

if $n_2 \gg n_1$, then $C_R \to 0$, $R_D \to -\infty$

Types of Data Redundancy

1.Coding redundancy

2.Spatial and temporal redundancy

3.Irrelevant information

Coding Redundancy

Different coding methods yield different amount of data needed to represent the same information.

Data compression can be achieved using an appropriate encoding scheme.

Example: Binary Coding

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

Coding Redundancy

Coding redundancy refers to the degree to which the information content of data is repeated or unnecessarily duplicated in a given coding scheme

A Code: a list of symbols (letters, numbers, bits etc.)

Code word: a sequence of symbols used to represent a piece of information or an event (e.g., gray levels)

Code word length: number of symbols in each code word

Definitions

- **N x M image**
- **r_k: kth gray level**
- **$P(r_k)$: probability of r_k**

Expected Value

$$E(X) = \sum_x xP(X = x)$$

$l(r_k)$: # of bits for r_k

Average # of bits: $L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$

Total # of bits: NML_{avg}

Example of Coding Redundancy

Variable Length Coding vs. Fixed Length Coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Example of Coding Redundancy

Concept: assign the longest code word to the symbol with the least probability of occurrence.

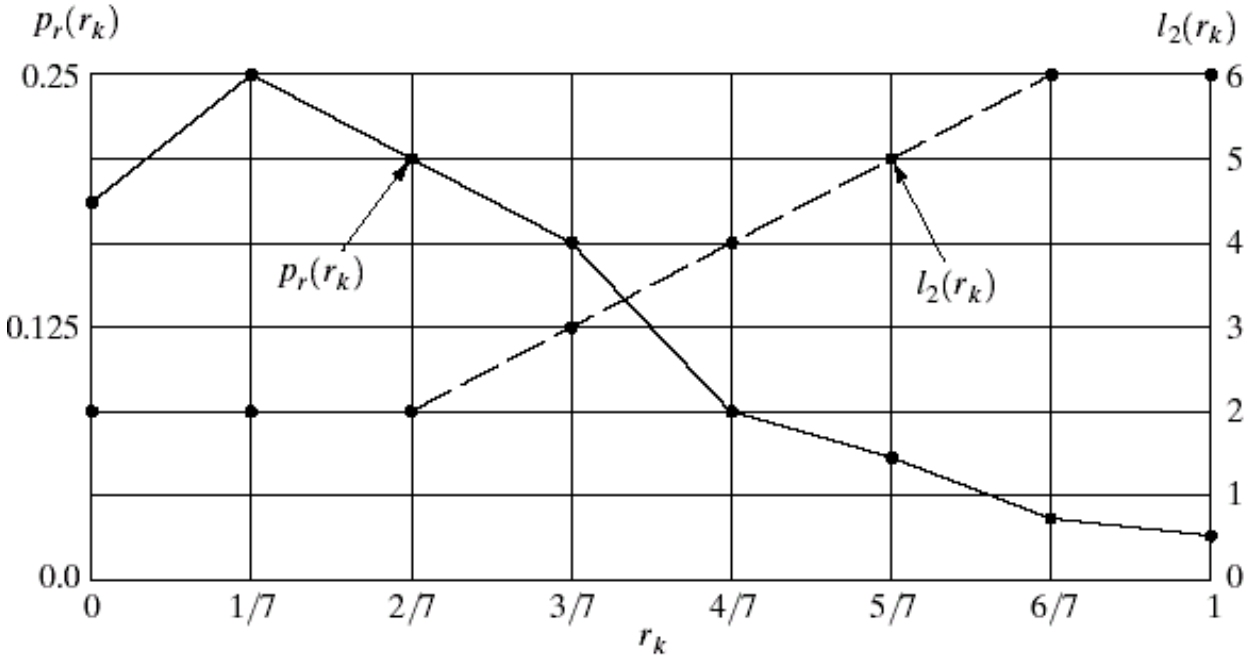


FIGURE 8.1 Graphic representation of the fundamental basis of data compression through variable-length coding.

Spatial and Temporal Redundancy

Spatial redundancy refers to the duplication or repetition of information within the same spatial location or domain.

Temporal redundancy refers to the duplication or repetition of information over time.

Irrelevant information

irrelevant information refers to data that does not contribute meaningfully to the redundancy within the dataset

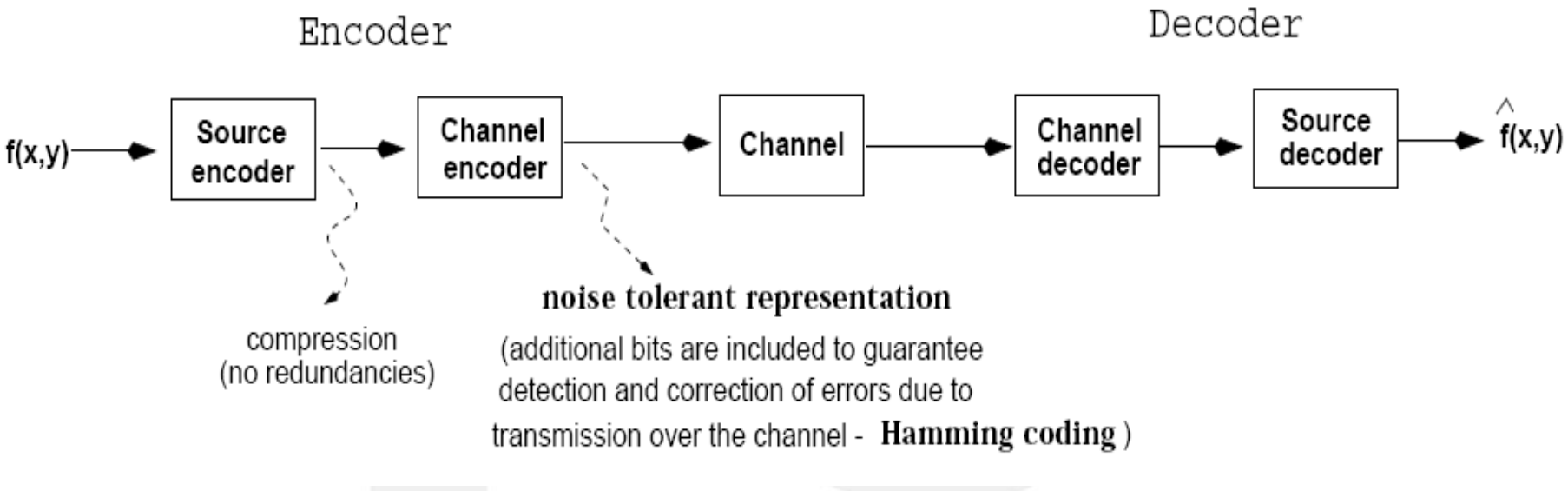
Modeling the Information Generation Process

- **Assume that information generation process is a probabilistic process.**
- **A random event *E* which occurs with probability *P*(*E*) contains:**

$$I(E) = log(\frac{1}{P(E)}) = -log(P(E)) \text{ units of information}$$

note that when $P(E) = 1$, then $I(E) = 0$: no information !

2.Image Compression Model



- **Entropy:** the average information content of an image

$$E = \sum_{k=0}^{L-1} I(r_k) \Pr(r_k)$$

using

$$I(r_k) = -\log(P(r_k))$$

- **Assumption:** statistically independent random events

$$H = -\sum_{k=0}^{L-1} P(r_k) \log(P(r_k))$$

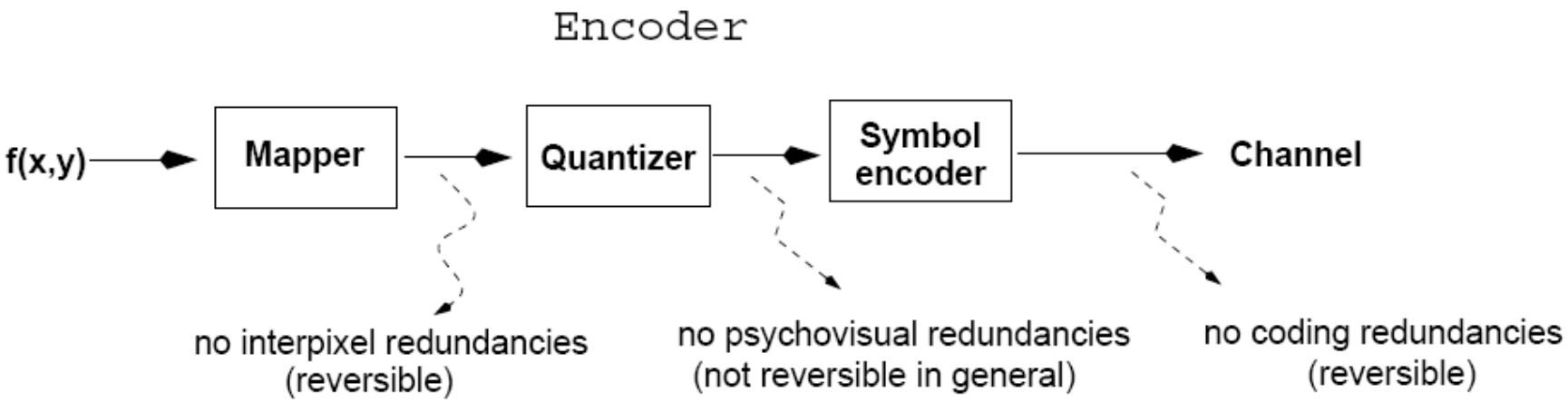
Redundancy:

$$R = L_{avg} - H$$

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

note that if $L_{avg} = H$, then $R = 0$ - no redundancy

Image Compression Model :Encoder



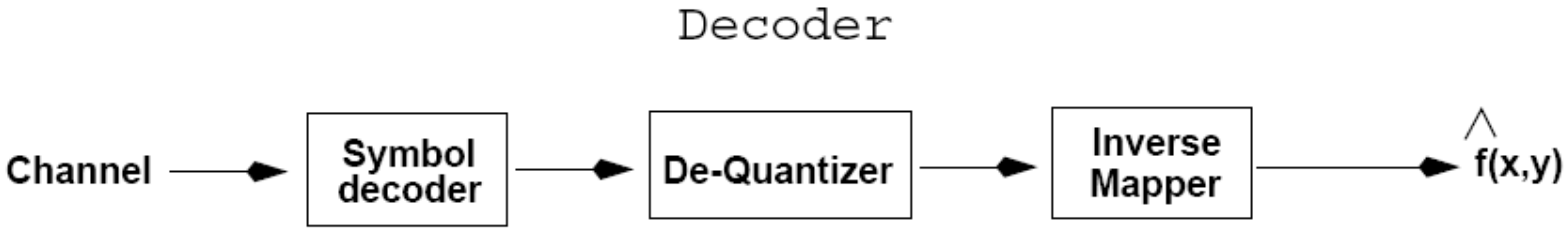
Mapper: transforms the input data into a format that facilitates reduction of inter pixel redundancies

Image Compression Model

Quantizer: reduces the accuracy of the mapper's output in accordance with some pre-established fidelity criteria.

- **Symbol encoder:** assigns the shortest code to the most frequently occurring output values.

Image Compression Model: Decoder



The inverse operations are performed.

But … quantization is irreversible in general.

Image Compression Models

- **Lossless Compression**: This method reduces file size without losing any image data. Examples include:

1. Run-Length Encoding (RLE)
2. Lempel-Ziv-Welch (LZW)
3. Predictive Coding
4. Huffman Coding

- **Lossy Compression**: Sacrifices some image quality for higher compression ratio.

4. Lossless Compression:- Huffman Coding

- It is a variable-length coding technique.
- It creates the *optimal code* for a set of source symbols.

Following step in Huffman's Procedure

1. Sort probabilities per symbol (upper :0 , lower :1)
2. Combine the lowest two probabilities
3. Repeat *Step2* until only two probabilities remain.

Lossless Compression: Huffman Coding

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

Lossless Compression: Run-length coding

- Used to reduce the size of a repeating string of characters (i.e., runs)

a a a b b b b b b c c → (a,3) (b, 6) (c, 2)

- Encodes a run of symbols into two bytes, a count and a symbol.
- Can compress any type of data but cannot achieve high compression ratios compared to other compression methods.

Lossless Compression: Run-length Coding

- Code each continuous group of 0's and 1's, encountered in a left to right scan of a row, by its length.

1 1 1 1 1 0 0 0 0 0 0 1 → (1,5) (0, 6) (1, 1)

Lossless Compression: Arithmetic Coding

- No assumption on encoding symbols one at a time.

  No one-to-one correspondence between source and code words.
- Slower than Huffman coding but typically achieves better compression.
- A sequence of source symbols is assigned a single arithmetic code word which corresponds to a sub-interval in [0,1]

Lossless Compression: Arithmetic Coding

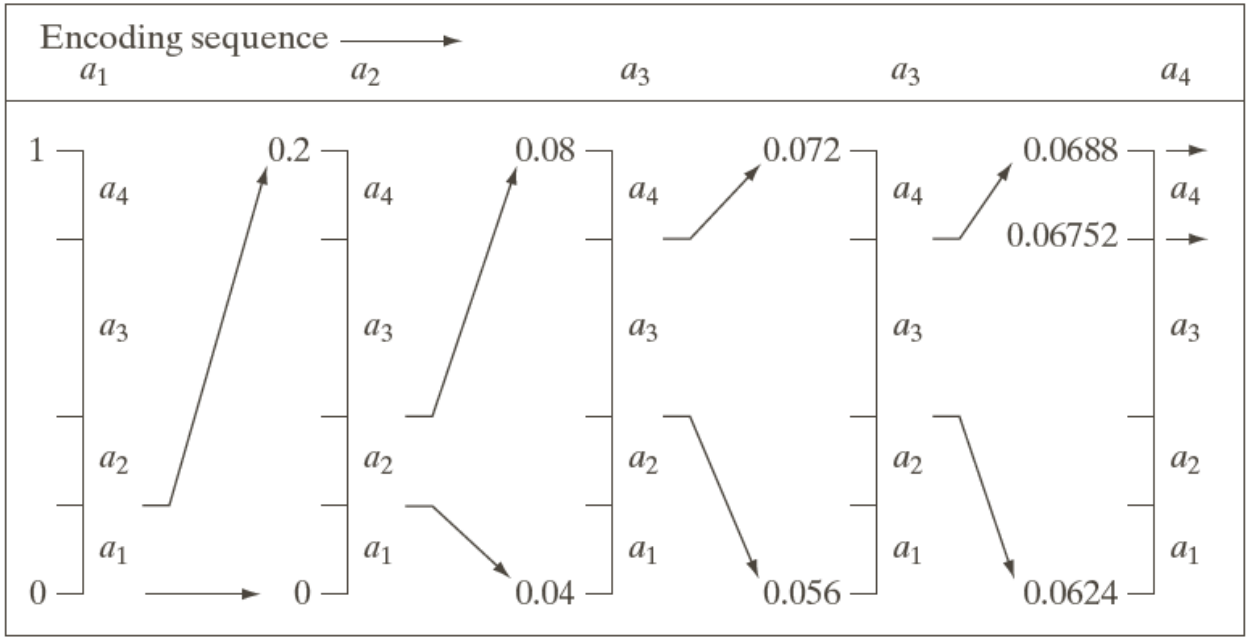
- As the number of symbols in the message increases, the interval used to represent it becomes smaller.

  Each symbol reduces the size of the interval according to its probability.

- Smaller intervals require more information units (i.e., bits) to be represented

Lossless Compression: Arithmetic Coding

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



Lossless Compression: LZW Coding

- Requires no priori knowledge of probability distribution of pixels
- Assigns fixed length code words to variable length sequences
- Patented Algorithm US 4,558,302
- Included in GIF and TIFF and PDF file formats

Lossless Compression: LZW Coding

- A codebook or a dictionary has to be constructed.

  Single pixel values and blocks of pixel values
- For an 8-bit image, the first 256 entries are assigned to the gray levels 0,1,2,..,255.
- As the encoder examines image pixels, gray level sequences (i.e., pixel combinations) that are not in the dictionary are assigned to a new entry.

Lossless Compression: LZW Coding

Consider the following 4 x 4 8 bit image

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

Dictionary Location	Entry
0	0
1	1
.	.
.	.
.	.
255	255
256	-
.	.
-	.
.	.
511	-

Lossless Compression: LZW Coding

If CS is found:

(1) No Output

(2) CR=CS

If CS not found:

(1) Output D(CR)

(2) Add CS to D

(3) CR=P

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

Lossless Compression: Bit-Plane Coding

- An effective technique to reduce inter pixel redundancy is to process each bit plane individually
- The image is decomposed into a series of binary images.
- Each binary image is compressed using one of well known binary compression techniques.
  - e.g., Huffman, Run-length, etc.

Lossless Compression: Bit-Plane Coding

- Once a message has been encoded using Huffman coding, additional compression can be achieved by encoding the lengths of the runs using variable-length coding!

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

e.g.,
(0,1)(1,1)(0,1)(1,0)(0,2)(1,4)(0,2)

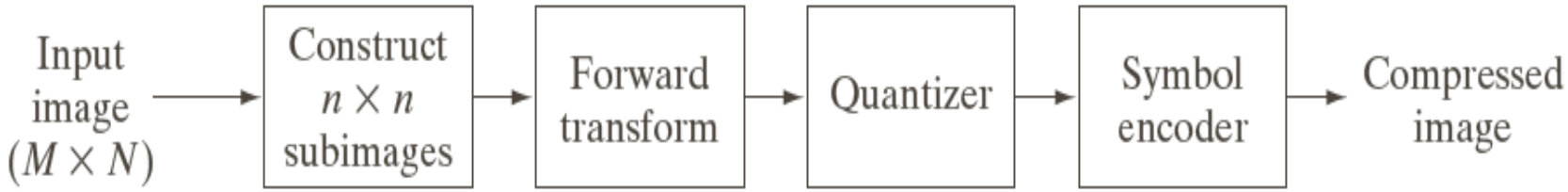
Lossless Compression: Bit-Plane Coding

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

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

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

Lossless Compression: Block Transform Coding



a
b

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

Lossless Compression: Block Transform Coding

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

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

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

Lossless Compression: Block Transform Coding

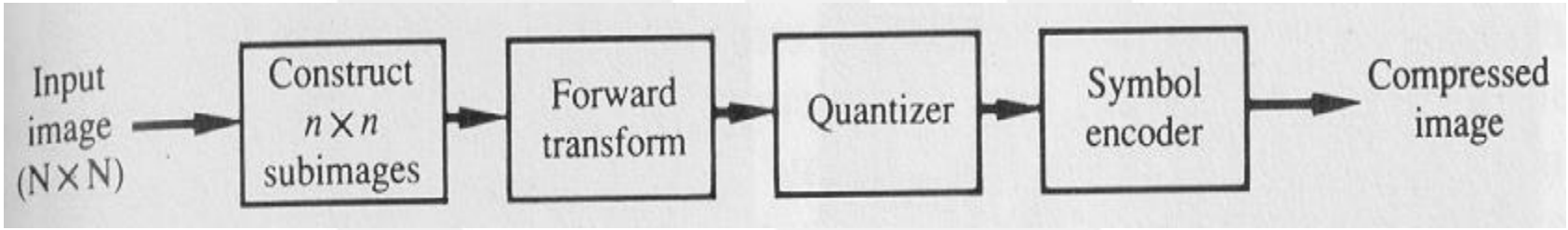
Given $T(u,v)$, $g(x,y)$ similarly can be obtained using the generalized inverse discrete transform

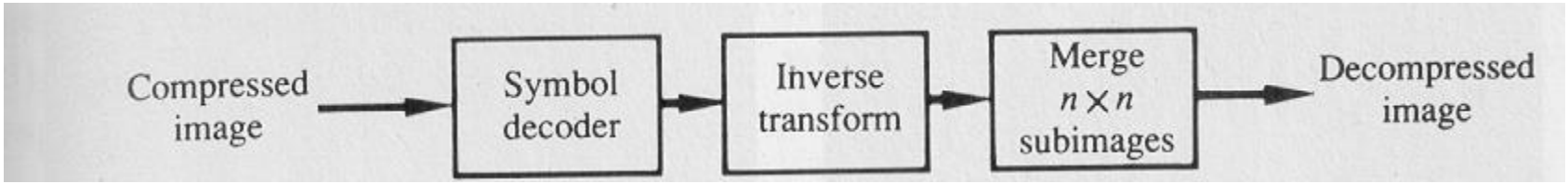
$$g(x,y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u,v)s(x,y,u,v)$$

for $x, y = 0,1,2,\ldots,n-1$.

5.Lossy Compression

- **Transform the image into a domain where compression can be performed more efficiently.**
- **Note that the transformation itself does not compress the image!**

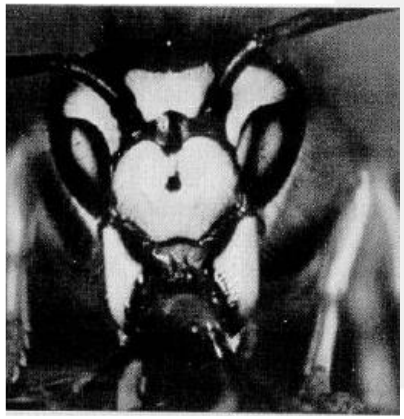


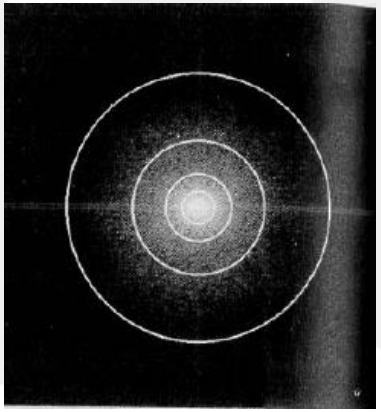


Lossy Compression

- Example: Fourier Transform

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) e^{\frac{j2\pi(ux+vy)}{N}}, \quad x,y=0,1,\ldots,N-1$$





The magnitude of the FT decreases, as *u*, *v* increase!

$$\hat{f}(x,y) = \frac{1}{N} \sum_{u=0}^{N/2-1} \sum_{v=0}^{N/2-1} F(u,v) e^{\frac{j2\pi(ux+vy)}{N}}, \quad x,y=0,1,\ldots,N-1$$

$\sum_{x,y} (\hat{f}(x,y) - f(x,y))^2$ is very small !!

Transform Selection

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)h(x,y,u,v)$$

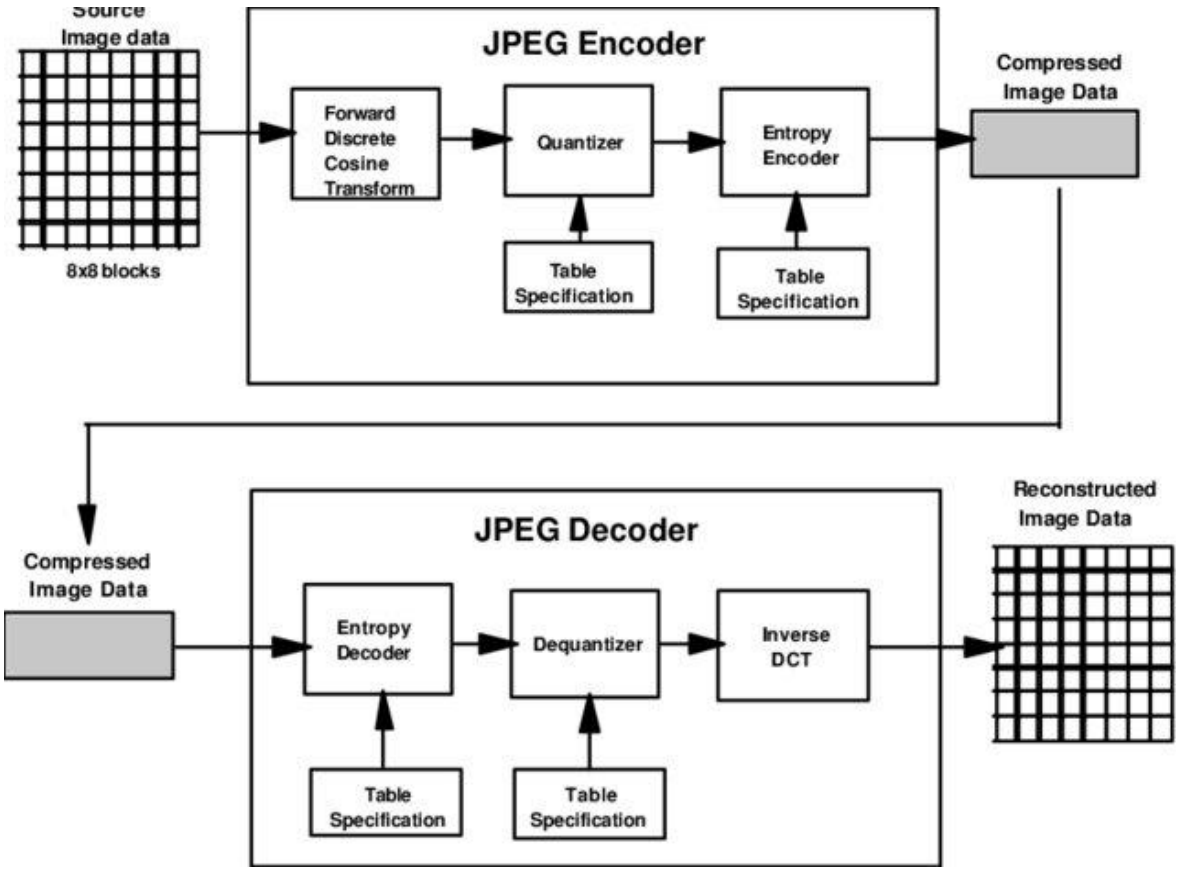
- (u,v) can be computed using various transformations, for example:
  - DFT
  - DCT (Discrete Cosine Transform)
  - KLT (Karhunen-Loeve Transformation)

Lossy Compression :JPEG Compression

JPEG uses DCT for handling inter pixel redundancy.

- Modes of operation:
  - (1) Sequential DCT-based encoding
  - (2) Progressive DCT-based encoding
  - (3) Lossless encoding
  - (4) Hierarchical encoding

Lossy Compression :JPEG Compression



Lossy Compression: JPEG Steps

1. Divide the image into 8x8 sub images;

   For each sub image do:

2. Shift the gray-levels in the range [-128, 127]
3. Apply DCT (64 coefficients will be obtained: 1 DC coefficient F(0,0), 63 AC coefficients F(u,v).
4. Quantize the coefficients (i.e., reduce the amplitude of coefficients that do not contribute a lot).

$$C_q(u,v) = Round[\frac{C(u,v)}{Q(u,v)}]$$

Lossy Compression : JPEG Steps

5. Order the coefficients using zig-zag ordering

- Place non-zero coefficients first

- Create long runs of zeros (i.e., good for run-length encoding)

6. Encode coefficients.

DC coefficients are encoded using predictive encoding

All coefficients are converted to a binary sequence:

6.1 Form intermediate symbol sequence

6.2 Apply Huffman (or arithmetic) coding (i.e., entropy coding)

Shifting and DCT

(a) Original 8 8 block

140	144	1471140	140	155	179	175	
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

(b) Shifted block

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	20	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

(c) Block after FDCT Eqn. (5)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

Quantization

(d) Quantization table (quality = 2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Quantization Table Example

```
for i=0 to n;
  for j=0 to n;
    Q[i,j]= 1 + (1+i+j)*quality;
  end j;
end i;
```

$$1 \leq quality \leq 25$$

(best - low compression)

(worst - high compression)

MPEG Compression

(c) Block after FDCT Eqn. (5)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

(d) Quantization table (quality = 2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

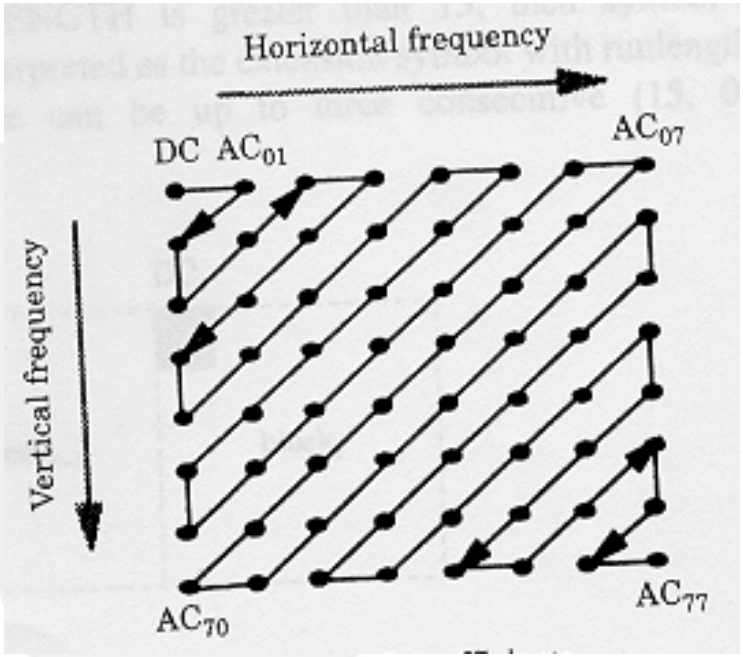
(e) Block after quantization Eqn. (6)

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	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

Zig-Zag Ordering (cont'd)

(e) Block after quantization Eqn. (6)

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	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



(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Intermediate Coding (cont'd)

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

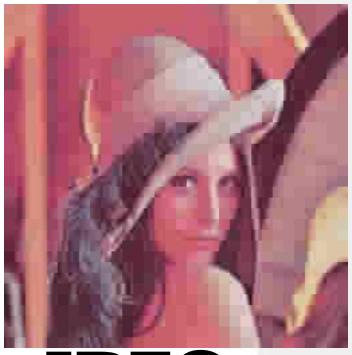
(g) Intermediate synbol sequence

(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
[redacted](5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

	symbol_1,	symbol_2
DC	(6)	(61)
AC	(0,2)	(-3)

JPEG - JPEG2000

- Two different standards
- Major differences:



JPEG

Created for natural imagery

Discrete Cosine Transform algorithm

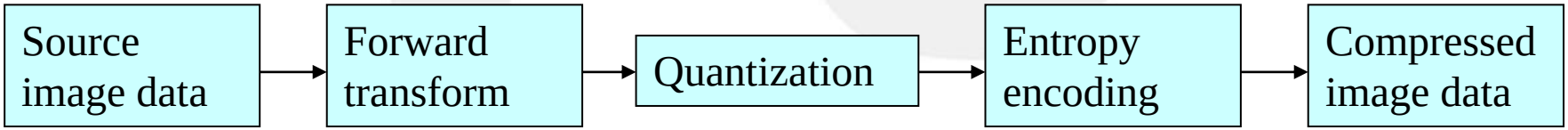
JPEG2000

Created for computer generated imagery

Discrete Wavelet Transform algorithm

JPEG2000

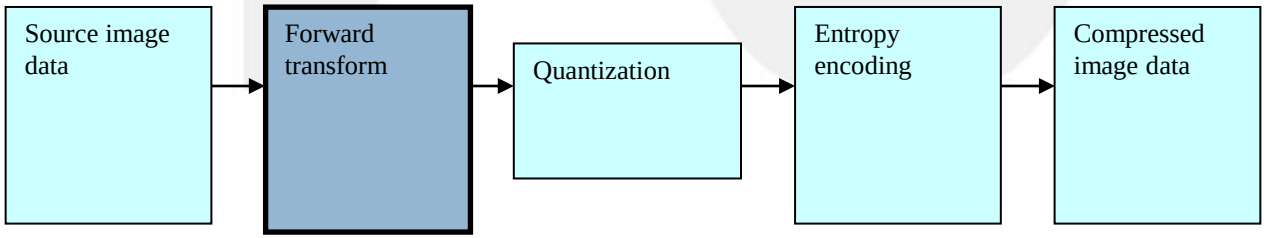
- Architecture of standard
  - Forward transform
  - Quantization
  - Entropy coding



JPEG2000

Forward transform

1. Level shifted
2. Divided into blocks
3. Forward Discrete Wavelet Transform



Lossy Compression

- Quantization

$$l_{ij} = \left\lfloor \frac{\theta_{ij}}{\Delta_b} \right\rfloor \qquad \Delta_b = 2^{R_b - \delta_b}\left(1 + \frac{\mu_b}{2}\right)$$

R_b Dynamic range: Depends on the number of bits and the choice of the wavelet

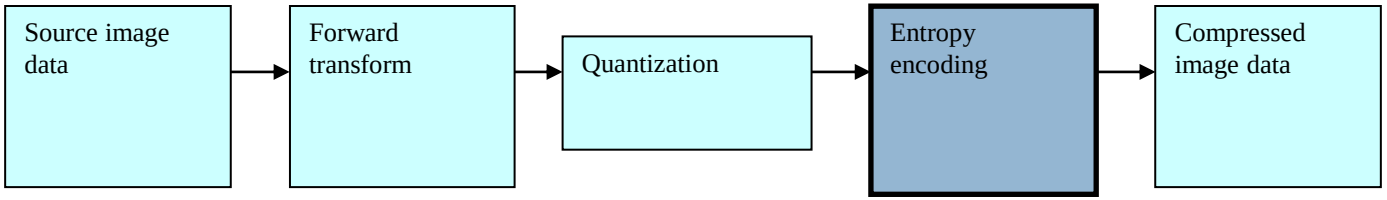
δ_b Exponent

μ_b Mantissa

JPEG2000

Entropy encoding

- Embedded Block Coding with Optimization Truncation of the embedded bit stream (EBCOT) :
  - Each sub band is divided into rectangular blocks which are coded independently called code blocks.
  - Bitstream is organized in a succession of layers.
  - Each layer corresponds to a certain distortion level.
  - The quality of the reproduction is proportional to the numbers layers received.



JPEG - JPEG2000 - Quality

Compression ratio 43:1

JPEG

JPEG2000

JPEG - JPEG2000 - Quality

- Method of picture quality measurement: PSNR

Peak Signal To Noise Ratio in dB:

$$PSNR = 20 * \log_{10}\left(\frac{255}{RMSE}\right) \qquad RMSE = \sqrt{\frac{\sum\left(f_{ij} - F_{ij}\right)^2}{N^2}}$$

f_{ij}	Pixel of the image
F_{ij}	Pixel of the reconstructed image
$N * N$	Size of the image
$RMSE$	Root Mean Squared Error

JPEG Compression

- JPEG: DCT algorithm
- JPEG2000: DWT algorithm
  - Layered structure
  - Many functionalities
- For a similar quality of PSNR, JPEG2000 compresses almost twice more than JPEG

References

1. Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing,* 3rd Edition Pearson Education Publisher.
2. Pratt William, *Digital Image Processing,* John Wiley & Sons
3. S Jayaraman, S Esakirajan and T Verakumar *Digital Image Processing,* McGraw Hill
4. John C. Russ, *The Image Processing Handbook*, by; CRC Press, 2007
5. A. K. Jain, *Fundamentals of Digital Image Processing,* Pearson Education
6. Mark Nixon and Alberto Aguado, *Feature Extraction and Image Processing,* Academic Press, 2008

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in