

# **DIGITAL IMAGE PROCESSING MODULE-V**

---

**PARVATHY LAKSHMY**

**FACULTY**

**ELECTRICAL DEPARTMENT**

**VJTI**

# FUNDAMENTALS OF IMAGE COMPRESSION

---

## What is Image Compression?

- Art and science of reducing the amount of data required to represent an image.
- Image compression is obtained by data compression.

Original  
Image



Compressed  
Image

# FUNDAMENTALS OF IMAGE COMPRESSION

---

- **Data Compression:** Reducing the amount of data required to represent a given information
- **Data:** Means by which information is conveyed
- **Example:**
- **Statement 1:** Your friend, Harsh will meet you at Rajiv Gandhi Airport in Hyderabad at 5 minutes past 6pm tomorrow night.
- **Statement 2:** Your friend will meet you at Rajiv Gandhi Airport at 5 minutes past 6 pm tomorrow night
- **Statement 3:** Harsh will meet you at Rajiv Gandhi at 6 pm tomorrow night
- All 3 statements represent same information with different levels of redundancy, first line containing maximum redundant data.

# FUNDAMENTALS OF IMAGE COMPRESSION

---

- **Redundant data:** Data representations that contain repeated or irrelevant information
- Let  $b$  and  $b^1$  are different data representations of the same information

$$\text{Compression ratio, } C = \frac{b}{b^1}.$$

$$\text{Redundancy , } R = 1 - \frac{1}{C}$$

Types of redundancy for 2-D Intensity Array:

- Coding Redundancy---Coded value of intensity
- Spatial or Temporal Redundancy-----Pixel positions
- Irrelevant Information

# TYPES OF REDUNDANCY

---

## Coding Redundancy:

- Code word: Information or event is assigned a sequence of code symbols
- Length of code word: No of bits to represent intensities of the image
- Let  $b$  be the number of bits to represent intensities of an image of  $M \times N$
- Length of code word      $I(r)=b$
- Assume that number of bits used is same for all pixels
- If representation of one pixel requires  $I(r)$  number of bits
- Total number of bits required to represent an image=  $M \times N \times I(r)$

# TYPES OF REDUNDANCY

---

- If the code word is dependent on the probability of its occurrence in an image, then the coding technique is called **Variable length coding**
- Average number of bits required to represent MXN image is given by

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

- Now the total number of bits =  $M \times N \times L_{avg}$

# TYPES OF REDUNDANCY

---

- Example

Table: An example with intensities, probability and code words

$r_k$	$p_r(r_k)$	Code 1	$I_1(r_k)$
$r_{87} = 87$	0.25	01010111	8
$r_{128} = 128$	0.47	01010111	8
$r_{186} = 186$	0.25	01010111	8
$r_{255} = 255$	0.03	01010111	8
$r_k$ for $k = 87, 128, 186, 255$	0	—	8

1. Calculate the  $L_{avg}$  and total number of pixels required when code 1 is used.

# TYPES OF REDUNDANCY

---

- Example

Table: An example with intensities, probability and code words

$r_k$	$p_r(r_k)$	Code 1	$I_1(r_k)$	Code 2	$I_2(r_k)$
$r_{87} = 87$	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 $k = 87, 128, 186, 255$	0	—	8	—	0

1. Calculate the  $L_{avg}$  and total number of pixels required when code 2 is used
2. Calculate the compression and redundancy for code 2

# TYPES OF REDUNDANCY

---

## **Spatial Redundancy:**

- Type of redundancy is related with the inter-pixel correlations with an image
- Much of visual contribution of a single pixel is redundant and can be guessed from the value of its neighbours
- Can be present in single frame or among multiple frames
- Can be solved by algorithms like: Predictive coding, Bit plane coding, run length coding algorithms

# MEASURING IMAGE INFORMATION

---

- If an event E is occurring with probability P(E) then the amount of information that exists in the event E is

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

- Base of log function will decide the units to measure information
- Commonly used units are bits
- If given a source of discrete set of possible events as  $\{a_1, a_2, a_3, a_4, a_5, \dots, a_n\}$  with probabilities  $\{P(a_1), P(a_2), P(a_3), P(a_4), P(a_5), \dots, P(a_n)\}$  then

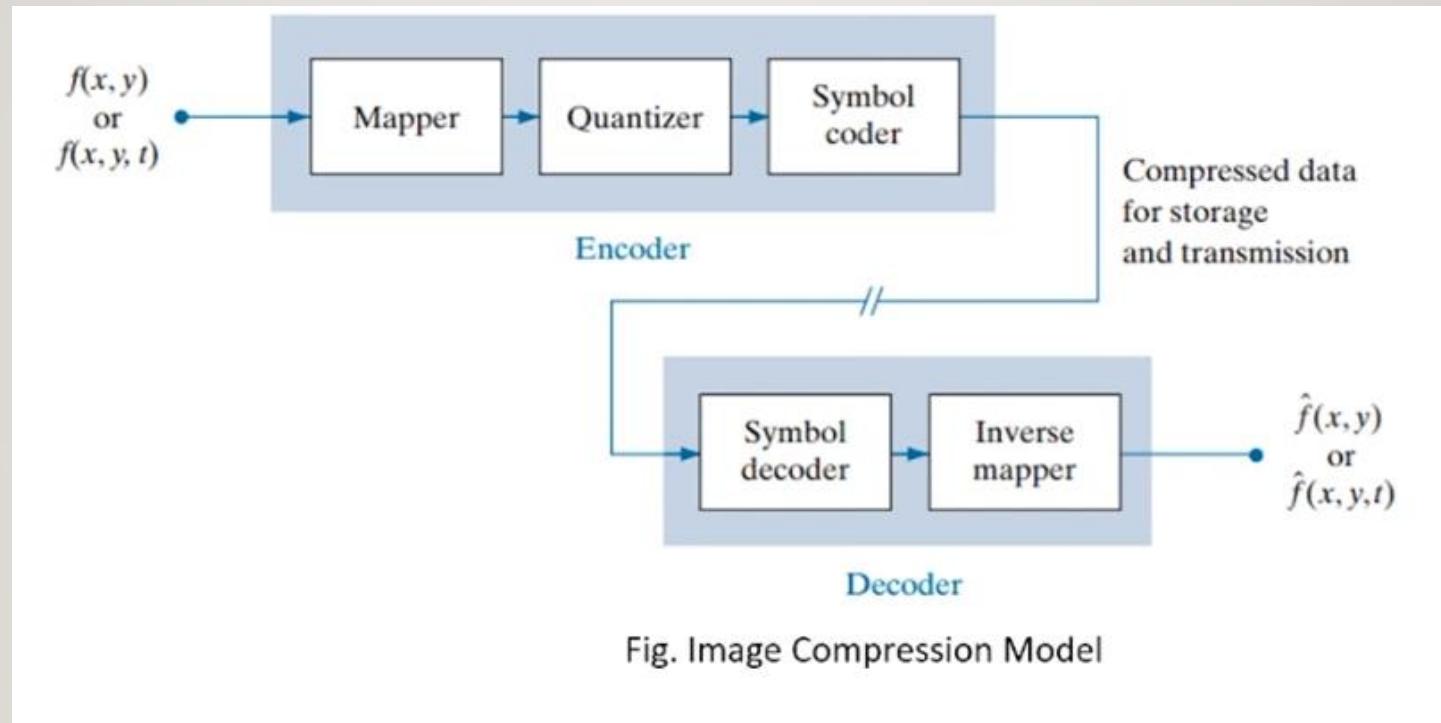
# MEASURING IMAGE INFORMATION

---

- Average information per source output= Entropy H
- $H = - \sum_{i=1}^N P(a_i) \log P(a_i)$  bits/pixel

# IMAGE COMPRESSION MODEL

---



# IMAGE COMPRESSION MODEL

---

- **Mapper:** Transforms the input image  $f(x,y)$  to  $f'(x,y)$  with reduced spatial and temporal redundancy Eg : RLC Coding
- **Quantizer:** Choose best quantization method to reduce irrelevant information Eg: Delta modulation, Differential PCM
- **Symbol coder:** Generates the fixed/ variable length code to represent quantizer output intensities
- Types of variable length coding: Lossy techniques, lossless/error free

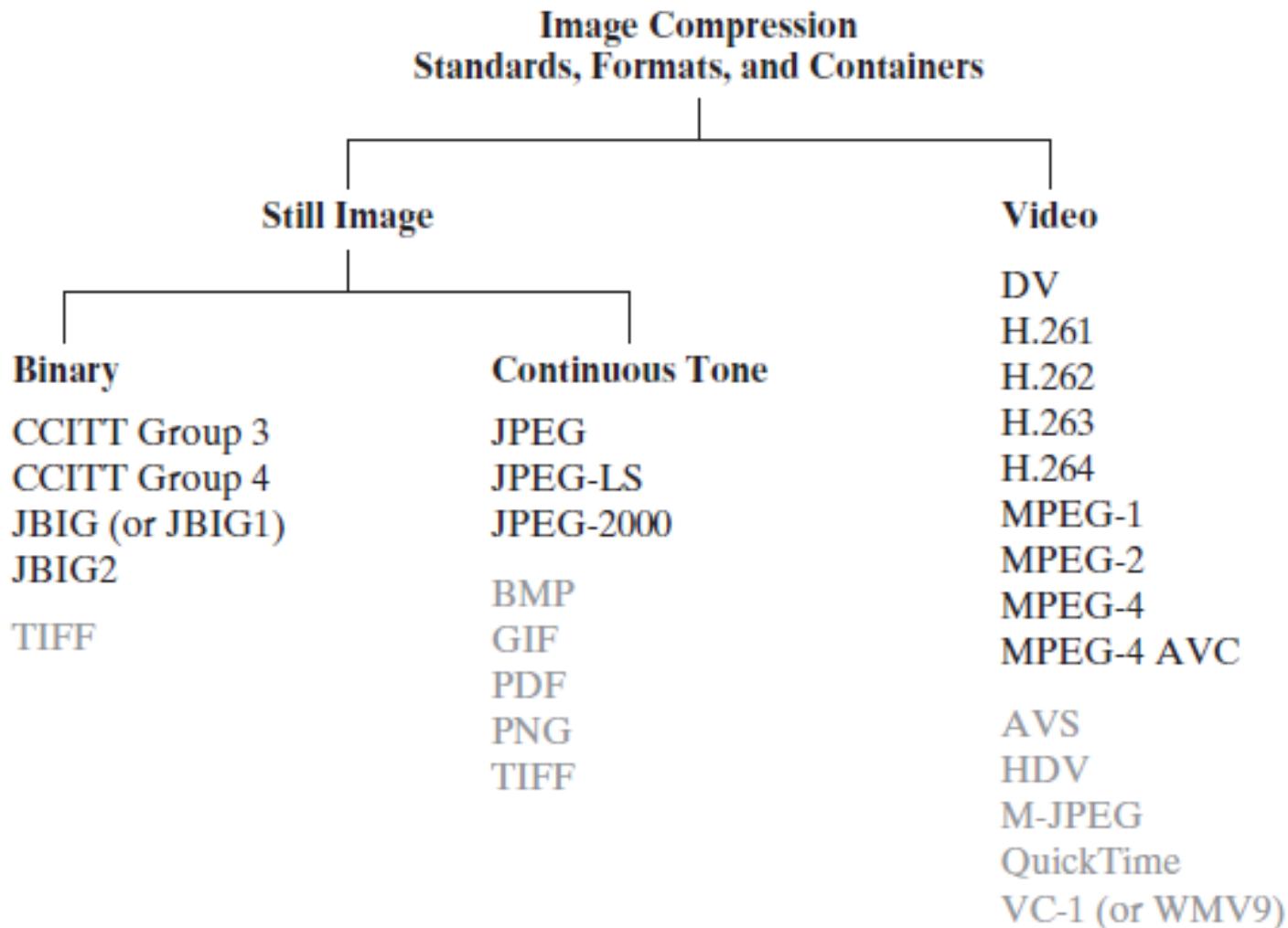
# IMAGE FORMATS, CONTAINERS AND COMPRESSION STANDARDS

---

- **Image File Format:** Standard way to organize and store image data.
- It defines how the data is arranged and the type of compression—if any—that is used.
- **Image Container:** Similar to a file format but handles multiple types of image data.
- **Image Compression Standards:** Procedures for compressing and decompressing images—that is, for reducing the amount of data needed to represent an image.

**FIGURE 8.6** Some popular image compression standards, file formats, and containers.

Internationally sanctioned entries are shown in black; all others are grayed.



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

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 [8.2.5] and Huffman [8.2.1] coding.
CCITT Group 4	ITU-T	A simplified and streamlined version of the CCITT Group 3 standard supporting 2-D run-length coding only.
JBIG or JBIG1	ISO/IEC/ITU-T	A <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 [8.2.7]. Context sensitive arithmetic coding [8.2.3] is used and an initial low resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary based methods [8.2.6] for text and halftone regions, and Huffman [8.2.1] or arithmetic coding [8.2.3] for other image content. It can be lossy or lossless.
<i>Continuous-Tone Still Images</i>		
JPEG	ISO/IEC/ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy <i>baseline coding system</i> (most commonly implemented) uses quantized discrete cosine transforms (DCT) on $8 \times 8$ image blocks [8.2.8], Huffman [8.2.1], and run-length [8.2.5] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ITU-T	A lossless to near-lossless standard for continuous tone images based on adaptive prediction [8.2.9], context modeling [8.2.3], and Golomb coding [8.2.2].
JPEG-2000	ISO/IEC/ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.2.3] and quantized discrete wavelet transforms (DWT) [8.2.10] are used. The compression can be lossy or lossless.

(Continues)

**TABLE 8.3**  
(Continued)

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

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

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 [8.2.4] for 1- through 8-bit images. It is frequently used to make small animations and short low resolution films for the World Wide Web.
PDF	Adobe Systems	<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 [8.2.9].
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.
<i>Video</i>		
AVS	MII	<i>Audio-Video Standard</i> . Similar to H.264 but uses exponential Golomb coding [8.2.2]. Developed in China.
HDV	Company consortium	<i>High Definition Video</i> . An extension of DV for HD television that uses MPEG-2 like compression, including temporal redundancy removal by prediction differencing [8.2.9].
M-JPEG	Various companies	<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 WMV9	SMPTE Microsoft	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 [8.2.8 and 8.2.9] and context dependent variable-length code tables [8.2.1]—but no predictions within frames.

# TYPES OF IMAGE COMPRESSION

---

- There are 2 types of image compression: **Error free(Lossless) compression and Lossy Compression**
- Lossy compression reduces file size by removing data that is not easily noticeable, resulting in a smaller file but a permanent loss of quality.
- Lossless compression reduces file size by reorganizing data to eliminate redundancy, but it allows the original file to be perfectly reconstructed upon decompression, so there is no loss in quality.

# HUFFMAN CODING

---

- **Huffman Coding:**
- Lossless coding technique
  - Coding technique to remove coding redundancy
  - Yields smallest possible number of code symbols/source symbols
  - It is a variable length coding technique
  - Minimum length code is given to the highest probability intensity

# HUFFMAN CODING-STEPS

---

- Calculate the Histograms
- Calculate the normalized histogram= $n_k/MN$
- Arrange the symbols in decreasing order of their probability
- Add the last two probability values
- Repeat steps 3 and 4 until there are only probabilities left which add upto give '1'.
- Now start giving the codes to symbols from right to left
  - If a probability of 1<sup>st</sup> symbol > Probability of 2<sup>nd</sup> symbol, then give 0 to 1<sup>st</sup> symbol and 1 to other.
  - Move to the previous stage and repeat the above till 1<sup>st</sup> stage is reached



Forward Steps



Backward Ste

Example: Generate codeword for intensities  $a_1, a_2, a_3, a_4, a_5, a_6$  using Huffman coding

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

8 bit image 10x10 pixels

Symbol $\pi_k$	Probability	Source reduction			
		1	2	3	4
$a_2$ 40	0.4	0.4	0.4	0.4	0.6
$a_6$ 30	0.3	0.3	0.3	0.3	0.4
$a_1$ 10	0.1	0.1	0.2	0.3	0.3
$a_4$ 10	0.1	0.1	0.1	0.1	0.1
$a_3$ 6	0.06	0.1	0.1	0.1	0.1
$a_5$ 4	0.04	0.1	0.1	0.1	0.1

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

Huffman source reductions

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

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

Huffman code assignment procedure

010100111100  
 $a_3$     $a_1$   $a_2$   $a_2$   $a_6$

$$\text{Entropy} = H = - \sum_{k=0}^{L-1} p(\pi_k) \cdot \log_2 P(\pi_k)$$

$$= - [0.4 \log_2 0.4 + 0.3 \log_2 0.3 + 0.1 \log_2 0.1 + 0.1 \log_2 0.1 + 0.06 \log_2 0.06 + 0.04 \log_2 0.04]$$

$$= 2.1435 \text{ bits/symbol}$$

Efficiency of Huffman code  $\eta = \frac{H}{L_{avg}} = \frac{2.1435}{2.2} = 0.9743 = 97.43\%$

Compression ratio  $C_R = \frac{b}{b'} = \frac{4 \times 4 \times 8}{4 \times 4 \times 2.2} = 3.63$

Redundancy  $R = 1 - \frac{1}{C_R} = 1 - \frac{1}{3.63}$   
= 1 - 0.2754  
= 0.7246  
= 72.46%

**Example:**

Perform Huffman algorithm for the following intensity distribution, for a 64 x 64 image.  
Obtain the coding efficiency for  $r_0 = 1008$ ,  $r_1 = 320$ ,  $r_2 = 456$ ,  $r_3 = 686$ ,  $r_4 = 803$ ,  
 $r_5 = 105$ ,  $r_6 = 417$ ,  $r_7 = 301$ .

**Solution:**

$$64 \times 64 = 4096$$

$$P(r_0) = \frac{1008}{4096} = 0.24$$

$$P(r_1) = \frac{320}{4096} = 0.07$$

$$P(r_2) = \frac{456}{4096} = 0.11$$

$$P(r_3) = \frac{686}{4096} = 0.16$$

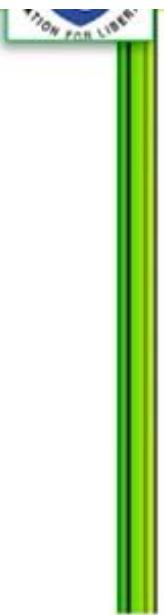
$$P(r_4) = \frac{803}{4096} = 0.19$$

$$P(r_5) = \frac{105}{4096} = 0.02$$

$$P(r_6) = \frac{417}{4096} = 0.10$$

$$P(r_7) = \frac{301}{4096} = 0.07$$

<u>Length(l)</u>	<u>P(R)</u>						
2	0.24 → 0.24 → 0.24 → 0.24	→ 0.32	→ 0.4	→ 0.56 (0)			
2	0.19 → 0.19 → 0.19 → 0.21	→ 0.24	→ 0.32	→ 0.4 (1)	(0)		
3	0.16 → 0.16 → 0.16 → 0.19	→ 0.21	→ 0.24	(0)	(01)		
3	0.11 → 0.11 → 0.16 → 0.16	→ 0.19	(00)	(11)			
3	0.10 → 0.10 → 0.11 → 0.16	(10)	(00)	(01)			
4	0.07 → 0.09 → 0.10	(000)	(01)				
5	0.07 → 0.07	(00000)	(0001)				
5	0.02	(00000)					



$$L_{avg} = \sum_{i=1}^n p(R_i) l(R_i)$$

$$= \underbrace{0.24(2)} + 0.19(2) + 0.16(3) + \underbrace{0.11(3)} + \underbrace{0.10(3)} + \underbrace{0.07(4)} + \underbrace{0.07(5)} + 0.02(5)$$
$$= 0.48 + 0.38 + 0.48 + 0.33 + 0.3 + 0.28 + 0.35 + 0.1 \quad \checkmark$$

$$\boxed{L_{avg} = 2.7}$$

$$H(x) = \sum_{i=1}^m P_i \log_2 \left( \frac{1}{P_i} \right)$$

$$= 0.24 \log_2 \left( \frac{1}{0.24} \right) + 0.19 \log_2 \left( \frac{1}{0.19} \right) + 0.16 \log_2 \left( \frac{1}{0.16} \right) + 0.11 \log_2 \left( \frac{1}{0.11} \right) + \\ 0.10 \log_2 \left( \frac{1}{0.10} \right) + 0.07 \log_2 \left( \frac{1}{0.07} \right) + 0.07 \log_2 \left( \frac{1}{0.07} \right) + 0.02 \log_2 \left( \frac{1}{0.02} \right)$$

$$= 0.24 \log_2 (4.16) + 0.19 \log_2 (5.26) + 0.16 \log_2 (6.25) + 0.11 \log_2 (9.09) + \\ 0.10 \log_2 (10) + 0.07 \log_2 (14.28) + 0.07 \log_2 (14.28) + \\ 0.02 \log_2 (50)$$

$$\Rightarrow 0.24(2.05) + 0.19(2.39) + 0.16(2.64) + 0.11(3.18) + \\ 0.10(3.32) + 0.07(3.83) + 0.07(3.83) + 0.02(5.64)$$

$$\boxed{H(x) = 2.69}$$

$$\therefore \eta = \frac{H(x)}{L_{avg}} \times 100 = \frac{2.69}{2.7} \times 100 = 99.6\%.$$

# ARITHMETIC CODING

---

- Lossless coding technique
- Also called Range Coding technique
- Sequences of code symbols are encoded together

Arithmetc Coding	Huffman Coding
1. Complex for short message	1. Simple technique
2. Optimum result	2. Not optimum result
3. One – to – one correspondence between source and codeword doesn't exist	3. One – to –one correspondence exists
4. Compression ratio more	4. Compression ratio is less
5. Execution time is more	5. Execution time is less

# ARITHMETIC CODING

---

- Instead of assigning single intensity by some codeword, we take an entire sequence of source symbols as single arithmetic code
- Eg ABC = Codeword ----- Given a single arithmetic code

- A 0110 } Code symbols
- B 1001 }

# ARITHMETIC CODING STEPS

---

- Divide the range (0, 1) into subranges of length equal to probabilities and name each range by the symbol
- Choose the range which is depicted by the first symbol in codeword
- Now divide the chosen range and repeat the step 1,2 till all the symbols in the codeword are coded
- New subgroups division are calculated as
  - 1<sup>st</sup> subgroup end division = starting range value + (range length X Probability of the 1<sup>st</sup> symbol)
  - 2<sup>nd</sup> subgroup end division= 1<sup>st</sup> subgroup end division +(range length X Probability of the 2<sup>nd</sup> symbol)
  - Similarly so on

# ARITHMETIC CODING STEPS

---

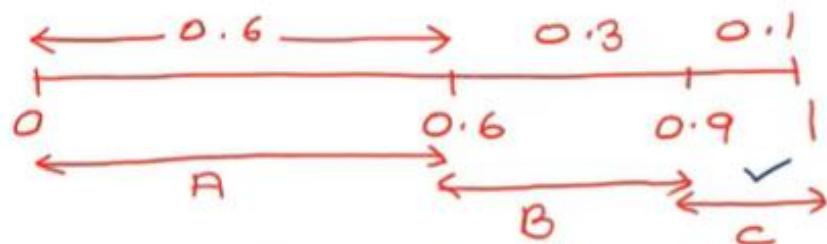
Example 1: Using arithmetic code generated code for sequence ‘CAB’ if

Intensity	A	B	C
Probability	0.6	0.3	0.1

**Example:** Code the string CAB using arithmetic coding.

**Solution:**

① Divide the range into  
0 - 1



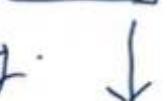
\* First character is  $\rightarrow$  C



Range  $\rightarrow$  0.9 - 1

0.1

code would start in  
this range only.



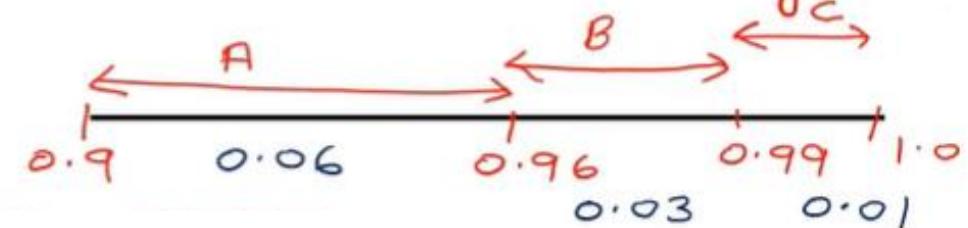
Divided among symbols  
according to prob.

Character	A	B	C
Probability	0.6	0.3	0.1

### Cumulative Character Probability Table

Character	Cumulative Probability
A	$0.9 + 0.6 \times 0.1 = 0.96$ ✓
B	$0.96 + 0.3 \times 0.1 = 0.99$
C	$0.99 + 0.1 \times 0.1 = 1.0$ ✓

(2) Resultant new range is:-



\* Next character  $\rightarrow$  A  
 $0.9 - 0.96$   
 code  $\leftarrow$   
 $\text{Range} = 0.96 - 0.9 = 0.06$   
 as per given prob

## Cumulative Character Probability Table

Character	Cumulative Probability
A	$0.9 + 0.6 \times 0.06 = 0.936$
B	$0.936 + 0.3 \times 0.06 = 0.954$
C	$0.954 + 0.1 \times 0.06 = 0.96$

2

(3) Resultant Range is →

\* Next character → B → END  
↓

Range : 0.936 - 0.954

\* Final code for string CAB

$$0.936 - 0.954$$

# ARITHMETIC CODING

---

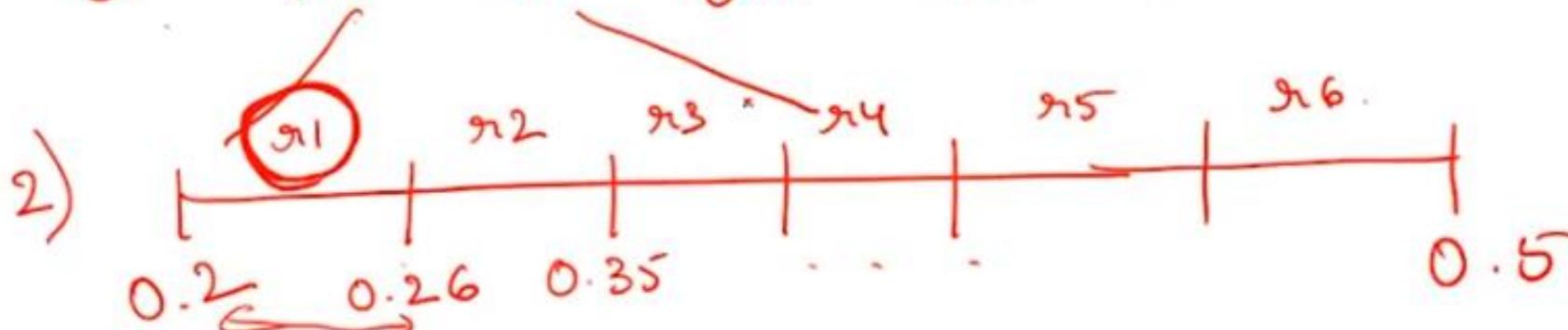
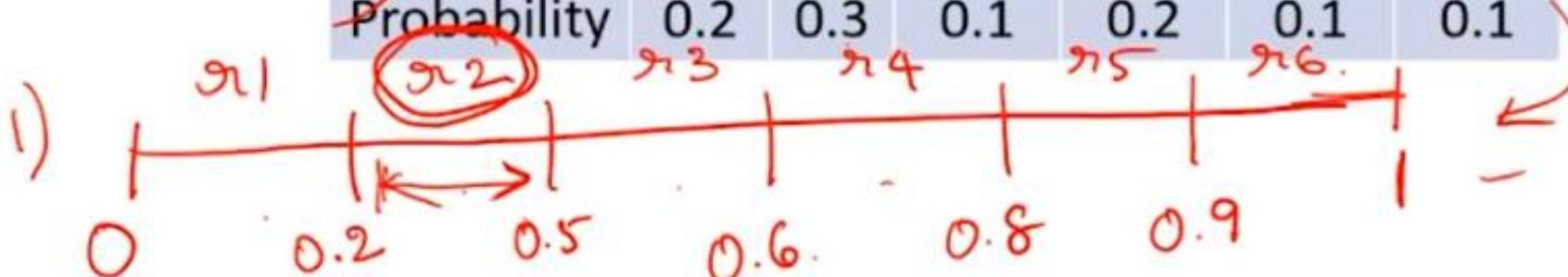
Example 2: Decode the message 0.2355 given the coding model is

Intensity	r1	r2	r3	r4	r5	r6
Probability	0.2	0.3	0.1	0.2	0.1	0.1

Example 2: Decode the message 0.2355 given the coding model is

Intensity	r1	r2	r3	r4	r5	r6
Probability	0.2	0.3	0.1	0.2	0.1	0.1

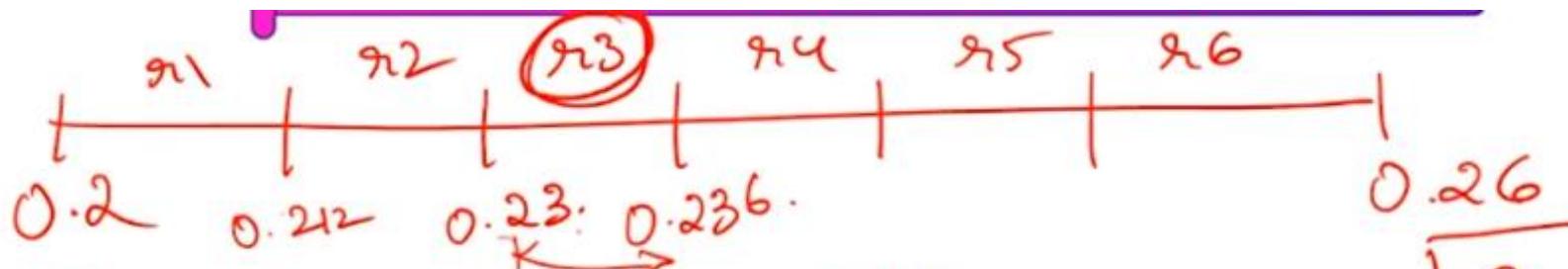
6 Subgroups



$$0.2 + 0.3 \times 0.2 = 0.26$$

$$0.26 + 0.3 \times 0.3 = 0.26 + 0.9 = 0.35$$

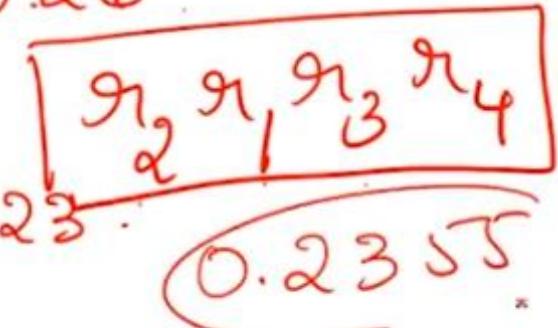
3)



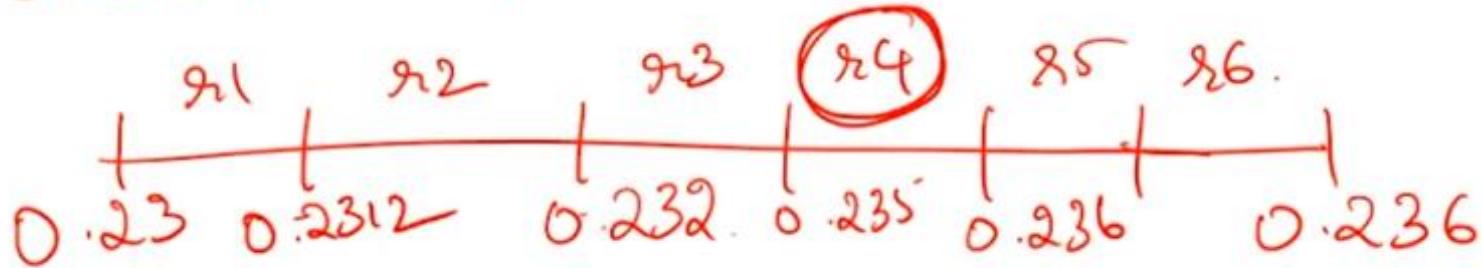
$$0.2 + 0.06 \times 0.2 = 0.212$$

$$0.212 + 0.06 \times 0.3 = 0.212 + 0.018 = 0.23$$

$$0.23 + 0.06 \times 0.1 = 0.236$$



4)



$$0.23 + 0.006 \times 0.2 = 0.23 + 0.0012 =$$

$$0.2312 + 0.006 \times 0.3 = 0.2312 + 0.0018 = 0.232$$

# DICTIONARY BASED CODING-LZW COMPRESSION

---

- Dictionary based coding techniques replace input strings with a code to an entry in a dictionary
- Most well-known dictionary-based coding techniques is LZW algorithm

- The idea behind Lempel-Ziv-Welch (LZW) coding is to **use a dictionary** to store the string patterns that have already been encountered
- Indices are used to encode the repeated patterns
- Encoder reads the input string and identifies the **recurrent words**, and outputs their indices from the dictionary
- If a new word is encountered, the word is sent as output in the uncompressed form and is entered into dictionary as a new entry**

# DICTIONARY BASED CODING-LZW COMPRESSION

---

## **Advantages:**

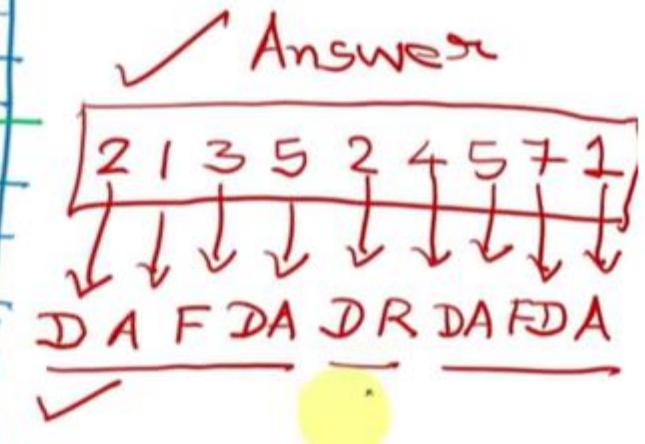
- Methods are faster
- Adaptive in nature
- These methods are not based on statistics
- Hence there is no dependency of the quality of the model on the distribution of data.

## **Encoding:**

- Objective is to identify the longest pattern for each collected segment of the input string
- Then it is checked in the dictionary
- If there is no match, the segment becomes a new entry in the dictionary

\* Implement LZW Compression for "DAFDADAFDA"

Currently recognized sequence	Pixel being processed	Encoded output	Dictionary location	Dictionary entry
/	/	/	1	A
/	/	/	2	D
/	/	/	3	F
/	/	/	4	R
D	A	2	5	DA
A	F	1	6	AF
F	D	3	7	FD
DA	D	5	8	DAD
D	R	2	9	DR
R	D	4	10	RD
DA	F	5	11	DAF
FD	A	7	12	FDA
A		1		



\* Implement LZW Compression for

Currently recognized sequence	Pixel being processed	Encoded output	Dictionary location	Dictionary entry
				39
				126
39	39	39	256	
39	126	39	257	39 39
126	126	126	258	39 126
126	39	126	259	126 126
39 39	126	256	260	126 39 126
126 126	39	258	261	126 126 39
39 39 126	126	260	262	39 39 126 126
126 39	39	259	263	126 39 39
39 126	126	257	264	39 126 126
126				

→ 39 39 126 126 256 258 260 259 253 126

→ 4x4, 8 bit image

[ 39 39 126 126  
39 39 126 126  
39 39 126 126  
39 39 126 126 ]

39 39 126 126 39 39  
→ 126 126 39 39 126 126  
→ 39 39 126 [126]

✓ [ 39 39 126 126  
39 39 126 126  
39 39 126 126  
39 39 126 126 ]

↑

# RUN LENGTH CODING

---

- Run Length Coding exploits the repetitive nature of the image
- RLC tries to identify the length of the pixel values and encodes the image in the form of a run
- Each row of the image is written as a sequence
- Then length is represented as a run of black and white pixels. This is known as run length coding.
- It is a very effective way of compressing an image
- If required, further compression can be done using variable length coding to code the run lengths themselves
- RLC is used to code binary and grey level images following CCITT standard

**Example:** Given a sample binary image. Apply Run-length Coding.

Solutions → ① Horizontal RLC

\* Run Length vectors  $\Rightarrow (0, 5)$

\* No. of vectors = 6  $(0, 3), (1, 2)$

\* Max. length = 5  
 $\downarrow$   
3 bits in binary  $(1, 5)$

\* No. of bits per pixel = 1 (0 or 1)

\* Total no. of pixels =  $6 \times (3+1) = 24$  ✓

\* No. of pixels for original image =  $5 \times 5 = 25$  ✓

\* compression Ratio =  $\frac{25}{24} = 1.042:1$  =

② vertical RLC

\* Run length vectors

$(0, 2), (1, 3)$

$(0, 2), (1, 3)$

$(0, 2), (1, 3)$

$(0, 1), (1, 4)$

$(0, 1), (1, 4)$

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

$5 \times 5$

\* No. of vectors = 10

\* 3 bits

\* 1 bit per pixel

\* Total no. of pixels  $\Rightarrow$   
 $= 10 \times (3 + 1) = 40$  ✓

\* original Image =  $5 \times 5 = 25$

\* compression Ratio =  $\frac{25}{40} = 0.625:1$  = 

# RUN LENGTH CODING

---

- Compression ratio changes with the scan line
- Theoretically :
- Estimate of the entropy of the black run is  $H_0$
- Estimate of entropy of white run is  $H_1$
- Estimate of average value of black run is  $L_0$
- Estimate of average value of white run is  $L_1$
- Approximate run length entropy of the image is given by:
- $$H_{Run\ length} = \frac{H_0 + H_1}{L_0 + L_1}$$

↗ 12 bits

str = [ 5 5 5 5 5 5 4 4 4 3 3 2 ] ;

(5,6)      (4,3)      (3,2)      (2,1)

∴ Encoded string = 5 6 4 3 3 2 2 1 → 8 bits

$$C_R = \frac{12 \text{ bits}}{8 \text{ bits}} = 1.5$$

$$13 + 7 + 9 = 29 \text{ bits}$$

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
<u>0111</u>	7
1000	8
<u>1001</u>	9
1010	10
1011	11
1100	12
<u>1101</u>	13
1110	14
1111	15

str = [ 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 ] ;

(1,13)      (0,7)      (1,9)

1,1101      0,0111      1,1001

Encoded = 11101 00111 1001  
                 ↓  
                 5 bits

∴  $C_R = \frac{29}{15} = 1.933$



Example 1: Consider the following  $8 \times 8$  image.

4	4	4	4	4	4	4	0
4	5	5	5	5	4	0	
4	5	6	6	6	5	4	0
4	5	6	7	6	5	4	0
4	5	6	6	6	5	4	0
4	5	5	5	5	4	0	
4	4	4	4	4	4	0	
4	4	4	4	4	4	0	

$$C_R = \frac{64}{40} = 1.6$$

The run-length codes using vertical (continuous top-down) scanning

mode are:

(4,9)	(5,5)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,1)	(6,1)	(7,1)
(6,1)	(5,1)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,5)	(4,10)	(0,8)

i.e. total of 20 pairs = 40 numbers. The horizontal scanning would lead to 34 pairs = 68 numbers, which is more than the actual number of pixels (i.e. 64).

→ (4,7) (0,1) (4,1) (5,5) (4,1) (0,1) (4,1)  
(5,1) (6,3) (5,1) (4,1) (0,1) (4,1) (5,1) (6,1) (7,1)  
(6,1) (5,1) (4,1) (0,1) (4,1) (5,1) (6,3) (5,1) (4,1)  
(0,1) (4,1) (5,5) (4,1) (0,1) (4,7) (0,1) (4,1) (0,1)

$$34 \times 2 = 68$$

$$C_R = \frac{64}{68} = 0.9411$$

# BIT PLANE CODING

---

- Idea of Run Length coding can be extended to multilevel images
- Bit plane coding techniques splits a multilevel image into a series of bi-level images
- It is an effective approach to provide bit stream truncation ability
- Any m-bit grey level image can be represented in the form:

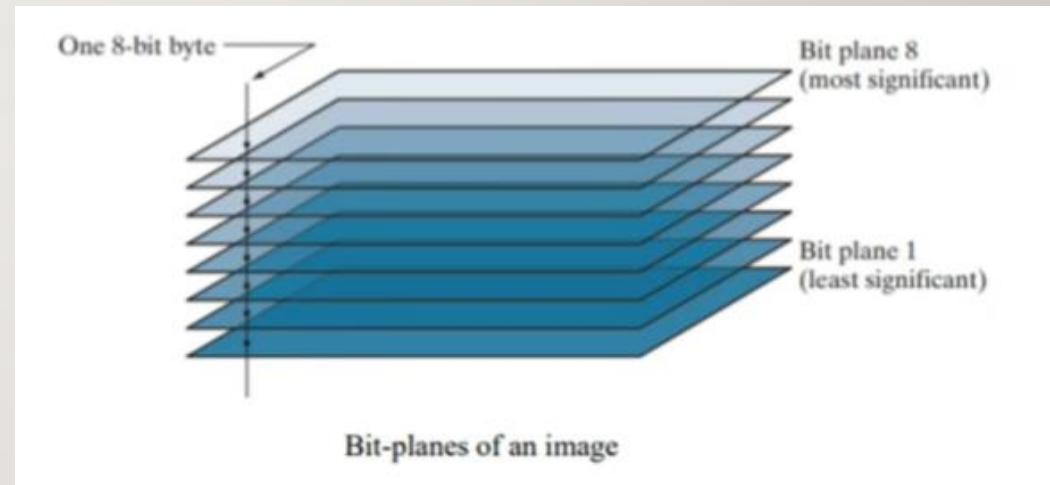
$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \dots + a_12^1 + a_02^0$$

- Zero order bit plane is generated by collecting the  $a_0$  bits of each pixel
- First order bit plane is generated by collecting all first bits of each pixel
- Similarly  $m-1$  order bit plane is generated by collecting all the  $a_{m-1}$  bits of each pixel

# BIT PLANE CODING

---

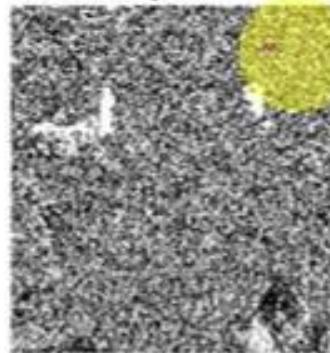
- Type of Lossless compression algorithm and an effective technique for reducing image's interpixel redundancy
- Idea to process image's bit planes individually
- Based on the concept of decomposing a multilevel(monochrome or color) image into a series of binary images and compressing each binary image via one of the several well known binary compression methods



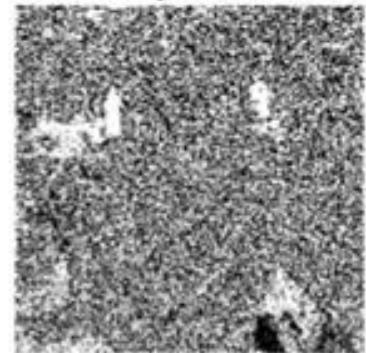
**Original Image**



**Bit plane 1**



**Bit plane 2**



**Bit plane 3**



**Bit plane 4**



**Bit plane 5**



**Bit plane 6**



**Bit plane 7**



**Bit plane 8**



**Example:** In a given grey scale image is given as apply Bit Plane Coding algorithm.

Discussion:

The binary equivalent of image is:

010	110	110
110	111	111
001	010	100

$A =$



Three Planes

\* MSB

\* Middle Bit

\* LSB

$$A = \begin{array}{|ccc|} \hline & 2 & 6 & 6 \\ & 6 & 7 & 7 \\ & 1 & 2 & 4 \\ \hline \end{array}$$

$3 \times 3$

The individual plane of the image can be compressed using RLC technique

$$\Rightarrow A(\text{MSB}) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad A(\text{LSB}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$A(\text{Middle}) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

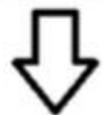
Individual Planes of image can be compressed by using RLC



6	7	6	6	7
0	0	0	1	2
1	1	1	2	3
4	5	5	4	2
6	6	6	7	7

0 - 7  
 $\frac{2^3}{2} = 8$   
  
 3 bit image

110	111	110	110	111
000	000	000	001	010
001	001	001	010	011
100	101	101	100	010
110	110	110	111	111



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

MSB plane

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

Centre bit plane

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

LSB plane

Decimal	Binary
0	000
1	<u>001</u>
2	010
3	011
4	100
5	101
6	<u>110</u>
7	<u>111</u>

# BIT PLANE CODING

---

## Disadvantage of scheme:

- Neighbourhoods in spatial domain like 3 and 4 having binary coded 011 and 100 are not present together in any of the planes
- To avoid this problem, grey codes can be used instead of binary codes
- In grey codes, successive codes differ only by one bit

	4-bit Binary	4-bit Gray Code
0	0000	0 0000
1	0001	1 0001
2	0010	2 0011
3	0011	3 0010
4	0100	4 0110
5	0101	5 0111
6	0110	6 0101
7	0111	7 0100
8	1000	8 1100
9	1001	9 1101
10	1010	10 1111
11	1011	11 1110
12	1100	12 1010
13	1101	13 1011
14	1110	14 1001
15	1111	15 1000

# BIT PLANE CODING

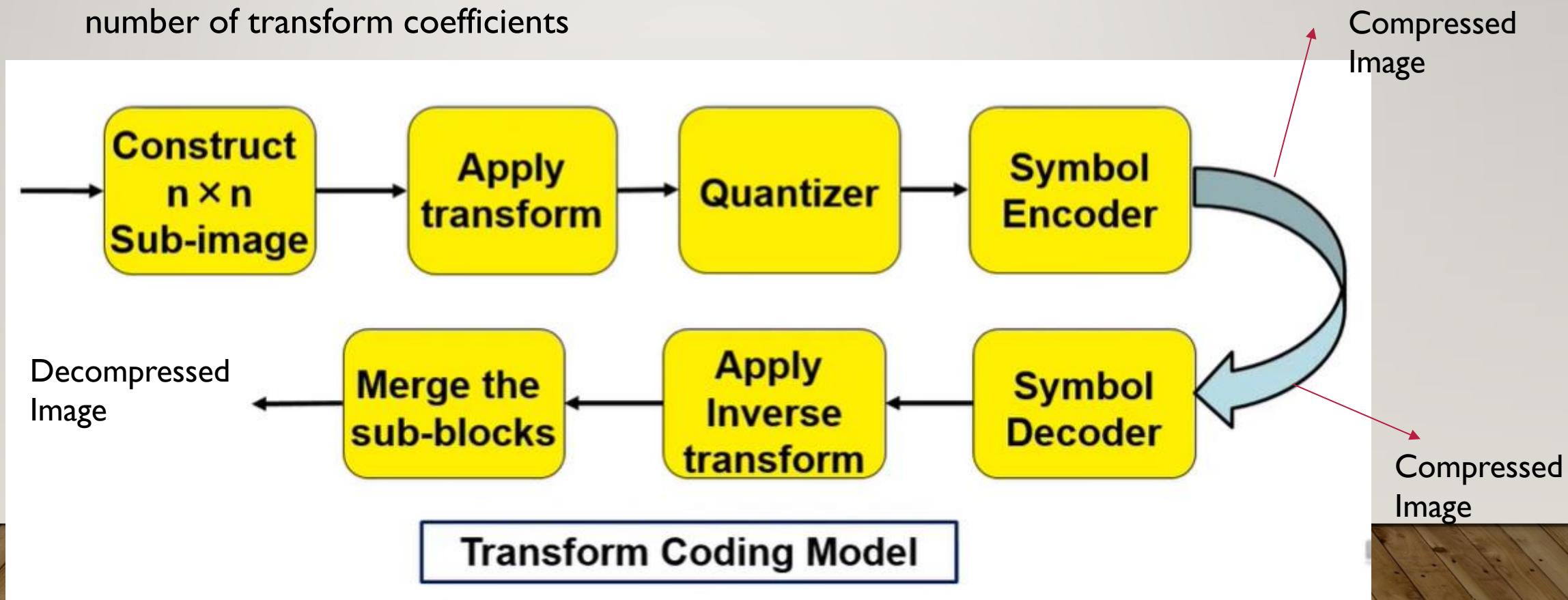
---

- Algorithm for generating grey code can be used instead of binary codes
- In Grey codes, successive codes differ only by one bit
- Algorithm for generating grey code can be written as:
- $$g_i = a_i + a_{i+1} \quad 0 \leq i \leq m - 2$$
$$= a_i \quad i = m - 1$$

# BLOCK TRANSFORM CODING

---

- Transform coding compresses image data by representing the original signal with a small number of transform coefficients



# BLOCK TRANSFORM CODING

---

## **Sub Image Selection:**

- At this stage image is divided into a set of sub-images.
- NXN image is decomposed into set of sub images of size  $n \times n$ .
- The values of  $n$  is a power of 2
- This process is done to ensure that correlation among the pixel is minimum
- This step is necessary to reduce computational complexity

# BLOCK TRANSFORM CODING

---

## Transform Selection:

- Idea behind transform coding is to use mathematical transforms for data compression
- **Goal:** Decorrelate pixels of each sub image and pack as much information possible to the smallest number of transform coefficients.
- Transforms like DFT, DCT and wavelet transforms are used
- Choice of transforms depends on the resources and amount of error associated with the reconstruction process
- Mathematical transforms are tools for information packing
- DCT-Better information packing capacity. KL transforms-effective but data dependent. DCT preferred-faster and can pack more information

# BLOCK TRANSFORM CODING

---

## **Bit Allocation(Quantization):**

- Bits must be assigned so that compressed image have minimum distortion
- Bits allocation must be done on the importance of the data.

## **Steps involved in bit allocation are:**

- Assign predefined bits to all classes of data in the image
- Reduce the number of bits by one and calculate the distortion
- Identify the data which is associated with minimum distortion and reduce one bit from its quota
- Find the distortion rate again
- Compare the target and if required repeat steps 1-4 to get the optimal rate.

# BLOCK TRANSFORM CODING

---

## Zonal Coding:

- Process involves multiplying each transform coefficient by the corresponding element in zonal mask, which has 1 in locations of maximum variance and 0 in other places
- A zonal mask is designed as part of this process
- These coefficients are retained as they convey more information
- The locations are identified based on the image models used for source symbols encoding
- The retained coefficients are quantized and coded
- Number of bits allocated may be fixed or may vary based on some optimal quantizer

$$M(u, v) = \begin{cases} 1 & \text{if } T(u, v) \text{ has large coefficients} \\ 0 & \text{otherwise} \end{cases}$$

# BLOCK TRANSFORM CODING

---

## **Threshold mask:**

- Works on the fact that transform coefficients having the maximum magnitude make the most contribution to the image
- Threshold may be one of the following:
  - Single global threshold
  - Adaptive threshold for each sub image
  - Variable threshold as a function of location for each coefficient in the sub image

# BLOCK TRANSFORM CODING

---

- Thresholding and quantization process can be combined; their approximation is:

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

$z(u,v) \rightarrow$  Transformed Normalized  
array

$$T = \hat{T}(u,v) \times z(u,v)$$

=

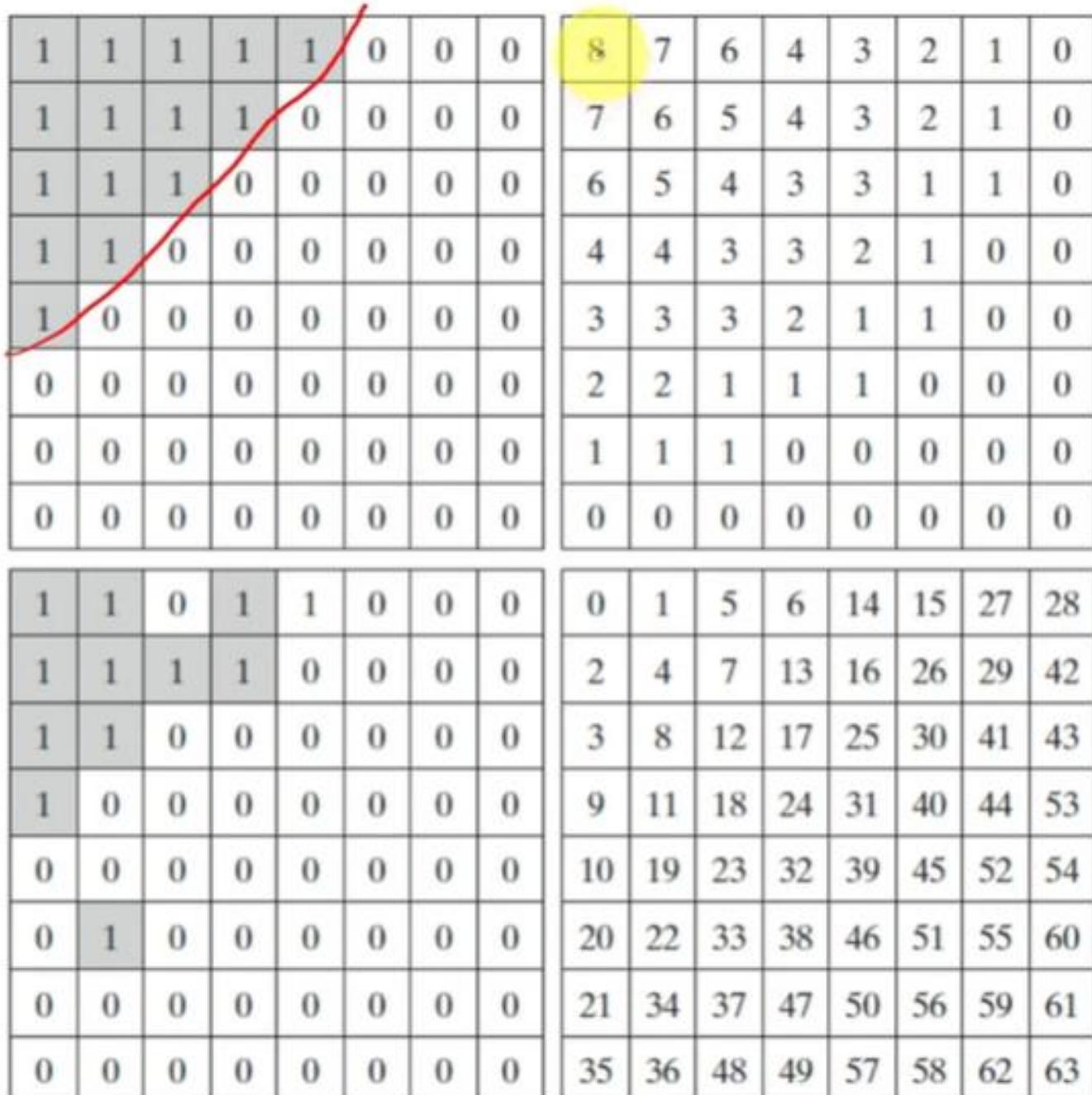
a	b
c	d

**FIGURE 8.29**

A typical  
(a) zonal mask,  
(b) zonal bit  
allocation,

(c) threshold  
mask, and  
(d) thresholded  
coefficient  
ordering

sequence. Shading  
highlights the  
coefficients that  
are retained.



# HUFFMAN CODING-NUMERICALS

---

- How many bits may be required for encoding the message ‘mississippi’ using Huffman Coding?
- The characters a to h have the set of frequencies based on the first 8 Fibonacci numbers as follows:
- a : 1, b : 1, c : 2, d : 3, e : 5, f : 8, g : 13, h : 21
- A Huffman code is used to represent the characters. What is the sequence of characters corresponding to the following code?
- 1101110011010

# ARITHMETIC CODING-NUMERICALS

---

- Encode "ABBCAB" with Uniform Probabilities
- Encode "DAD" with Given Probabilities

A: 0.4

D: 0.3

B: 0.2

C: 0.1

# LZW COMPRESSION-NUMERICALS

---

- Perform LZW Compression of the sequence "BABAABRRRA"
- Perform LZW Compression of "BABAABAAA"

# BITPLANE CODING

---

- Perform bitplane coding for a 3x3 image matrix with pixel values:

167, 133, 111

144, 140, 135

159, 154, 148