# UNIT 4
# IMAGE COMPRESSION

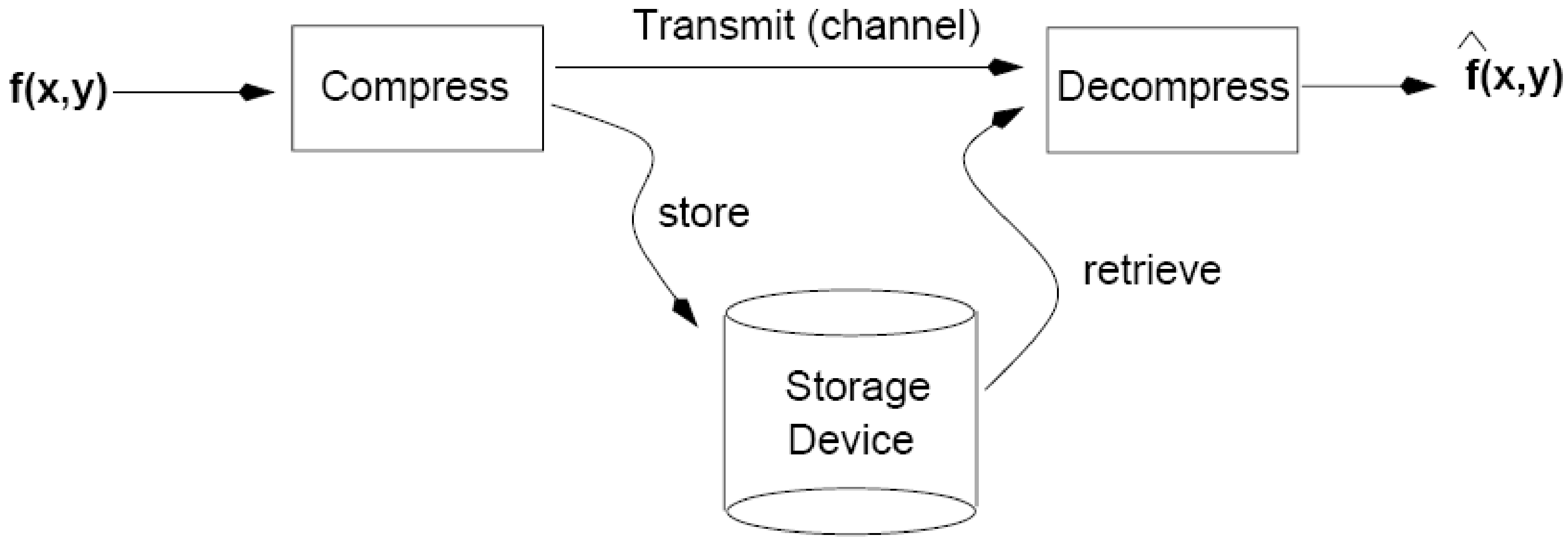
# Contents

- Introduction
- Lossless Compression
  - Variable length coding , LZW coding
  - Bit plane coding, Predictive coding
  - DPCM
- Lossy Compression
  - Transform coding
  - Wavelet coding
- Basics of  Image Compression Standards
- Basics of  Vector quantization

# Introduction

- Because much of the day to day information is graphical or pictorial in nature, the storage and communications requirements are immense.
- The goal of image compression is to reduce the amount of data required to represent a digital image.
- Image Compression plays an important role in video conferencing, remote sensing, satellite TV, FAX, document and medical imaging.

# Steps involved in image compression & decompression



# Need for compression

- The ***objective of image compression*** is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form.
- Sometimes the given data contains data which has no relevant information,or restates/repeats the known information: ***Data redundancy.***

Image=Information+redundant data

- Data is the means by which information is conveyed.
- Data compression aims to reduce the amount of data required to represent a given quantity of information while preserving as much information as possible.

# Data redundancy

- Let $n_1$ and $n_2$ denote the number of information carrying units in two data sets that represent the same information
- The relative redundancy $R_D$ is define as :

$$R_D = 1 - \frac{1}{C_R}$$

where $C_R$, commonly called the compression ratio, is

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

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 $C_R \to \infty$, $R_D \to 1$

# Types of data redundancy

There are three main data redundancies used in image compression:

- ***Coding redundancy:*** The uncompressed image usually is coded with each pixel by a fixed length.
- → Using some variable length code schemes such as Huffman coding and arithmetic coding may produce compression.
- ***Interpixel redundancy:*** Spatial redundancy,it exploit the fact that an image very often contains strongly correlated pixels, large regions whose pixel values are the same or almost the same.
- ***Psychovisual redundancy:*** Human eye does not respond with equal sensitivity to all incoming visual information.
- → Some piece of information are more important than others.
- → Removing this type of redundancy is a lossy process and the lost information can not be recovered.

# Coding redundancy

**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

Example: (binary code, symbols: 0,1, length: 3)

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

# Contd..

The gray level histogram of an image can be used in construction of codes to reduce the data used to represent it. Given the normalized histogram of a gray level image where,

$$p_r(r_k) = n_k / n \qquad k = 0, 1, 2, \ldots\ldots\ldots\ldots L\text{-}1$$

$r_k$ is the **discrete random variable** defined in the interval [0,1]

$p_r(k)$ is the **probability of occurrence** of $r_k$, L is the number of **gray levels**

$n_k$ is the **number of times** that $k^{th}$ **gray level appears** , **n** is the **total number of pixels** in image.

Average number of bits required to represent each pixel is given by:

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

Where, $l(r_k)$ is the **number of bits** used to represent each value of $r_k$.

# Types of compression

**Lossless:**

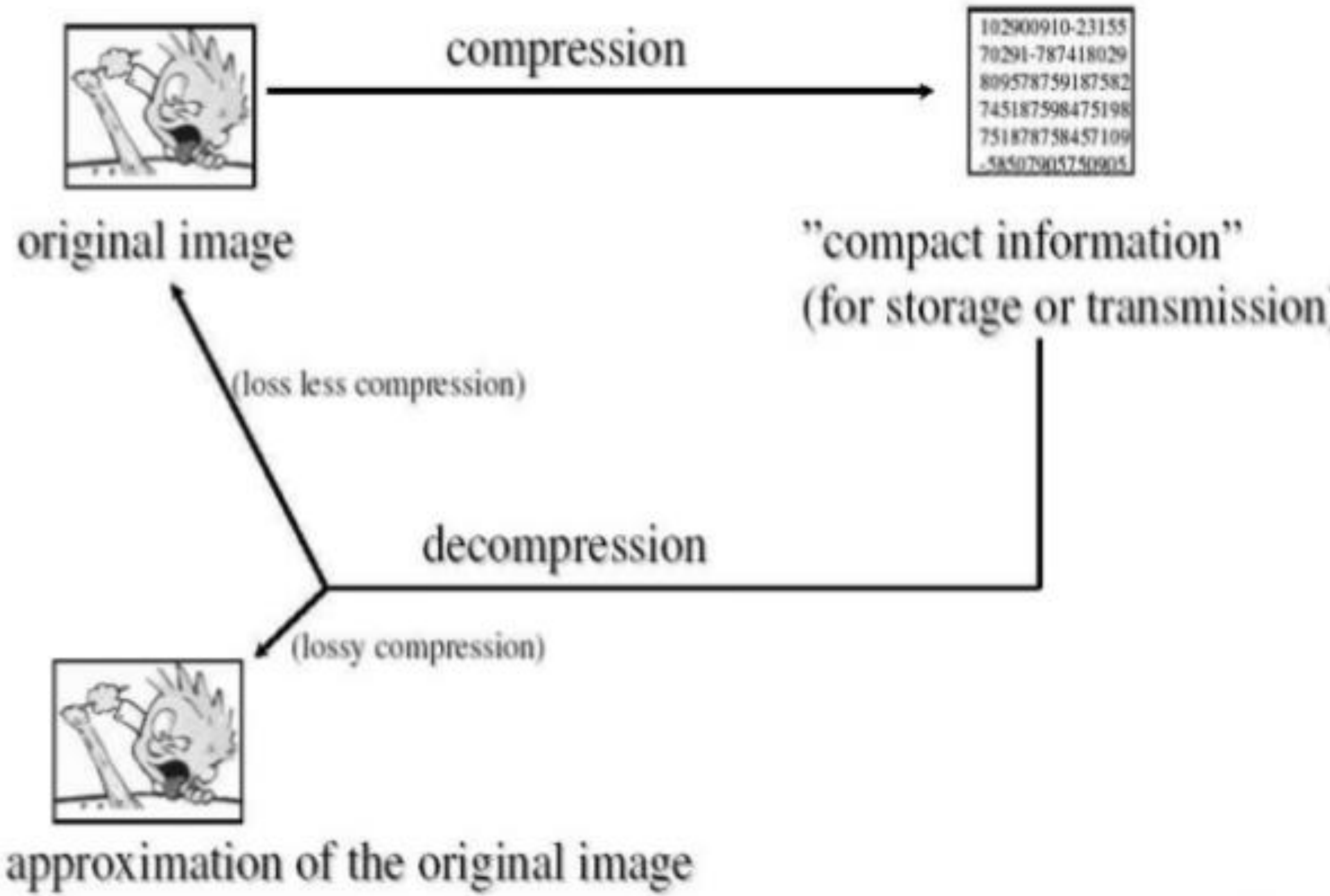
- In lossless data compression, the integrity of the data is preserved,i.e. no part of the data is lost in the process.
- Lossless compression methods are used when we cannot afford to lose any data.

- lossless compression for legal and medical documents, computer programs
- exploit only code and inter-pixel redundancy

**Lossy:**

- Lossy compression can achieve a high compression ratio, since it allows some acceptable degradation. Yet it cannot completely recover the original data

- digital image and video where some errors or loss can be tolerated
- exploit both code and inter-pixel redundancy and sycho-visual perception properties


compression
original image
"compact information"
(for storage or transmission)
(loss less compression)
decompression
(lossy compression)
approximation of the original image

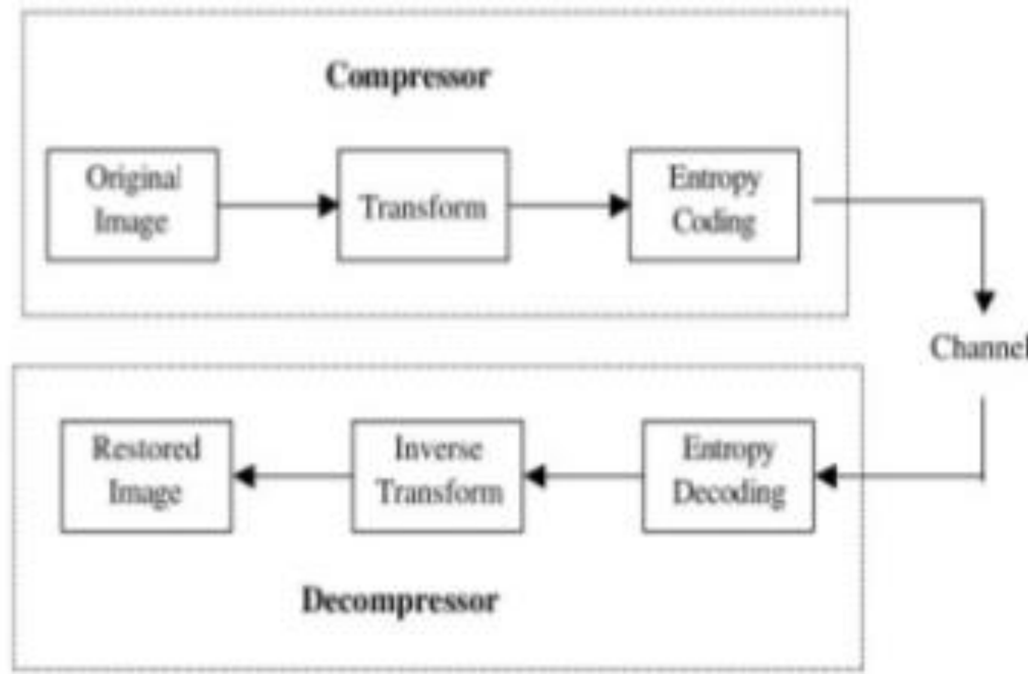
# Lossless Compression:

Two-step algorithms:

1. Transforms the original image to some other format in which the inter-pixel redundancy is reduced.

2. Use an entropy encoder to remove the coding redundancy.

The lossless decompressor is a ***perfect inverse*** process of the lossless compressor.



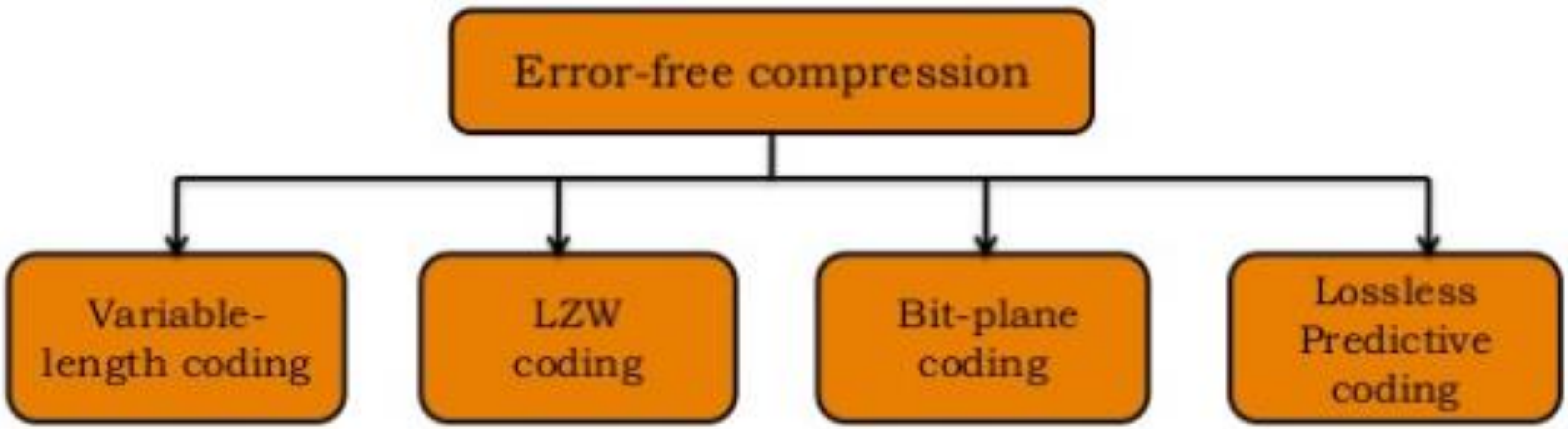

Applications:

- Archive of medical or business documents
- Satellite imaging
- Digital radiography

They provide: Compression ratio of 2 to 10.

# Classification of Lossless / Error free compression

- **Error-free compression** is generally composed of two relatively independent operations: (1) reduce the interpixel redundancies and (2) introduce a coding method to reduce the coding redundancies.



# Variable length coding

The coding redundancy can be minimized by using a variable-length coding method where the shortest codes are assigned to most probable gray levels.

The most popular variable-length coding method is the **Huffman Coding**. **Huffman Coding:** The Huffman coding involves the following 2 steps.

1) Create a series of source reductions by ordering the probabilities of the symbols and combining the lowest probability symbols into a single symbol and replace in the next source reduction.

2) each Code reduced source starting with the smallest source and working back to the original source.

# Huffman coding

- The most popular technique for removing coding redundancy is due to Huffman (1952)
- Huffman Coding yields the smallest number of code symbols per source symbol
- The resulting code is *optimal*

## 1)Huffman source reductions:

ai's corresponds to the available gray levels in a given image.

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

**2) Huffman code assignments:**

The first code assignment is done for **a2** with the **highest probability** and the last assignments are done for **a3** and **a5** with the **lowest probabilities.**

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

The shortest codeword (1) is given for the symbol/pixel with the highest probability (a2). The longest codeword (01011) is given for the symbol/pixel with the lowest probability (a5).

The **average length of the code** is given by:

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

It is **uniquely decodable. Because** any string of code symbols can be decoded by examining individual symbols of string from left to right.

Ex. 01010 011 1 1 00

First valid code: 01010 – a3, 011 – a1,

Thus, completely decoding the message, we get, a3a1a2a2a6

Slower than Huffman coding but typically **achieves better compression.**

# LZW Coding

- Remove Inter-pixel redundancy and addresses spatial redundancies in the image
- Requires no priori knowledge of probability distribution of pixels
- Assigns fixed length code words to variable length sequences
- Patented Algorithm – in GIF, TIFF and PDF file formats

- It is an example of a category of algorithms called *dictionary-based encoding* .
- The idea is to create a dictionary (a table) of strings used during the communication session.
- If both the sender and the receiver have a copy of the dictionary, then previously-encountered strings can be substituted by their index in the dictionary to reduce the amount of information transmitted.

# LZW coding

**Lempel–Ziv–Welch (LZW)**is a universal lossless data compression algorithm created by **Abraham Lempel, Jacob Ziv, and Terry Welch.**

The key to LZW is building a dictionary of sequences of symbols (strings) as the data is read and compressed.

Whenever a string is repeated, it is replaced with a single code word in the output.

At **decompression time**, the same dictionary is created and used to replace code words with the corresponding strings.

# Contd..

A **codebook** (or **dictionary**) needs to be constructed. **LZW** compression has been integrated into a several images file formats, such as **GIF** and TIFF and **PDF**.

Initially, the **first 256** entries of the dictionary are assigned to the **gray levels 0,1,2,..,255 (i.e., assuming 8 bits/pixel)**

Consider a 4x4, 8 bit image

| | | | |
|---|---|---|---|
| 39 | 39 | 126 | 126 |
| 39 | 39 | 126 | 126 |
| 39 | 39 | 126 | 126 |
| 39 | 39 | 126 | 126 |

**Initial Dictionary**

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

# Contd..

As the encoder examines image pixels, gray level sequences (i.e., blocks) that are not in the dictionary are assigned to a new entry.

| | | | |
|---|---|---|---|
| 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 | 39-39 |
| 511 | - |

- Is 39 in the dictionary........**Yes**
- What about **39-39**.............No
  * Add 39-39 at location 256

# Contd..

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

CR = empty

**If CS is found:**
(1) No Output
(2) CR=CS

**else:**
(1) Output D(CR)
(2) Add CS to D
(3) CR=P

**Concatenated Sequence: CS = CR + P**

| Currently (CR) Recognized Sequence | (P) 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 | | |

# Decoding LZW

- Use the dictionary for decoding the “encoded output” sequence.
- The dictionary need not be sent with the encoded output.
- Can be built on the “fly” by the decoder as it reads the received code words.

# Run length coding

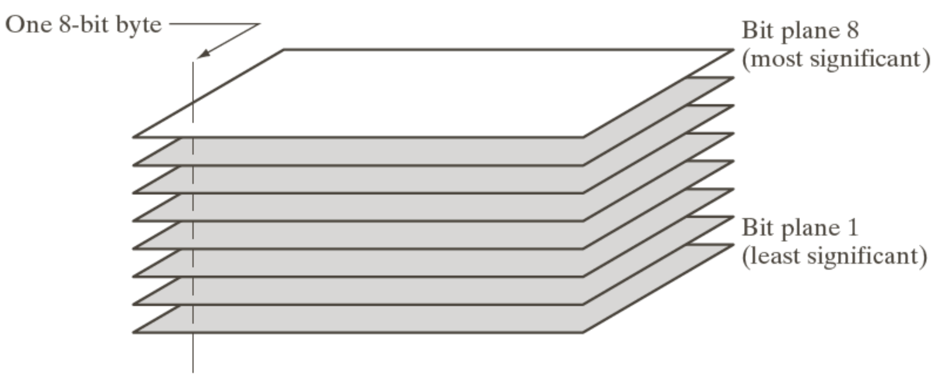
- It can be used to compress data made of any combination of symbols.
- It does not need to know the frequency of occurrence of symbols.
- The method is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences.

# Bit Plane Coding

- Process each bit plane individually.

(1) Decompose an image into a series of binary images.

(2) Compress each binary image (e.g., using run-length coding)



# Contd..

- Another effective method to reduce interpixel redundancies.
- Image's bit planes are processed individually.
- Based on decomposing a multilevel (monochrome / color) image into a series of binary images & compressing each binary using any binary compression method.

Bit plane decomposition:

- Gray levels of an m-bit gray level image can be represented in form of base 2 polynomial.

$$a_{m-1}2^{m-1}+a_{m-2}2^{m-2}+........+a_1 2^1+a_0 2^0 \quad .........(1)$$

# Contd..

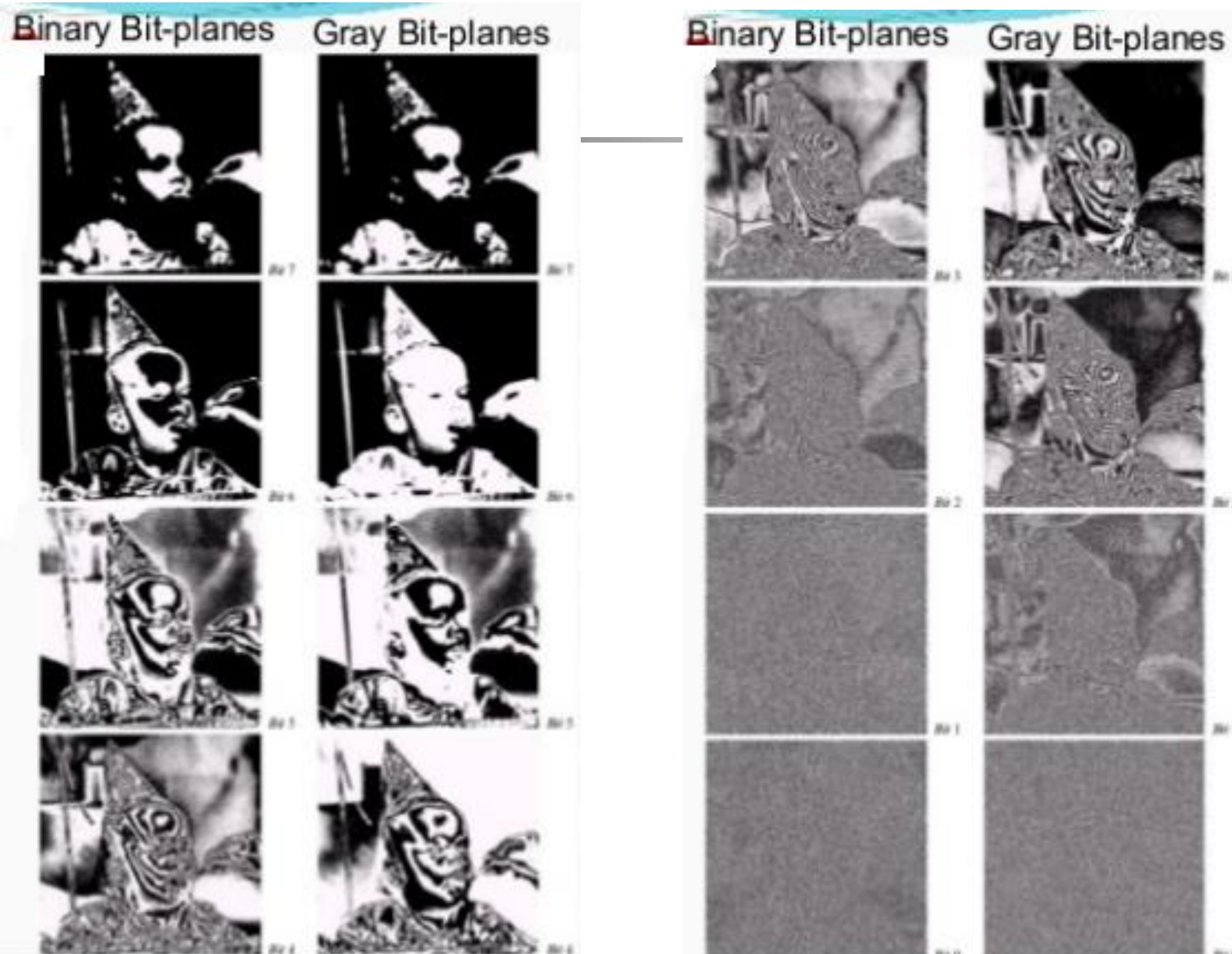
- A simple method of decomposing the image into a collection of binary image is to separate the m coefficients of the polynomial into m-1 bit planes.

Disadvantage:

- Small changes in gray level can have significant impact on complexity of bit planes.

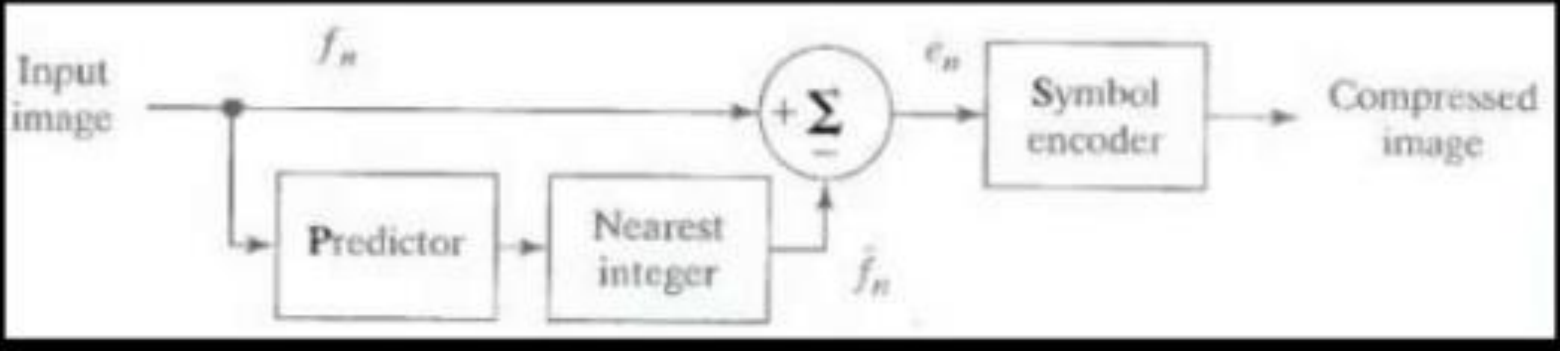
Ex. If two adjacent pixels have intensity of 127 (01111111) and 128 (10000000), every bit plane will contain a corresponding 0 to 1 (or 1 to 0) transition.

# Bit plane decomposition Example



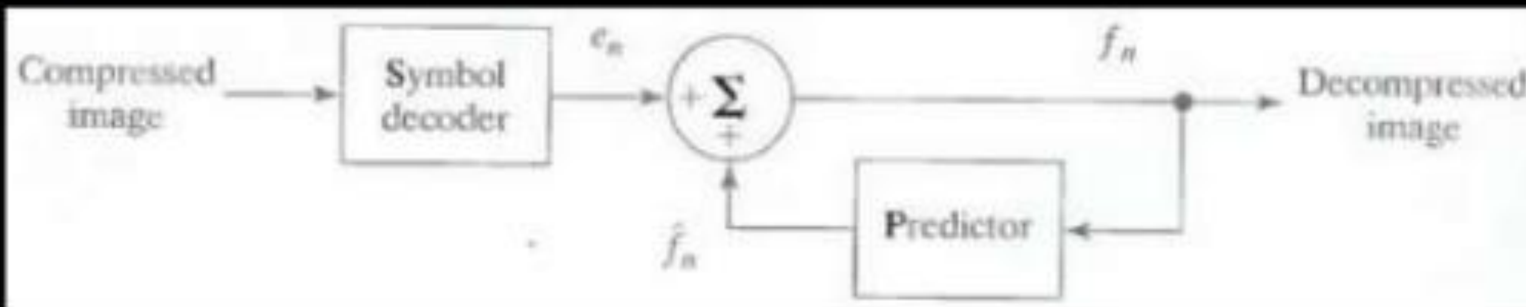
# Lossless Predictive Coding

- Based on eliminating the interpixel redundancies of closely spaced pixels by extracting & coding only the new information in each pixel.
- New information: difference between the actual & predicted value of that pixel.



# Contd..

- Figure shows basic component of a lossless predictive coding system.




- It consists of an encoder & a decoder each containing an identical predictor.

- As each successive pixel of input image f(n) is introduced to the encoder, predictor generates its anticipated value.

Output of the predictor is then rounded to the nearest integer f(n)bar & used to form the difference or prediction error.

e(n) = f(n)– f(n)bar

# Contd..

- It is coded using a variable length to generate the next element of the compressed data stream.
- The decoder reconstruct the fn from the received variable-length code words & perform the inverse operation
- f(n) = e(n) + f(n)bar
- f(n)bar is generated by prediction formed by a linear combination of m previous pixels.

$$f(n)\text{bar} = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$ where, m – order of linear predictor

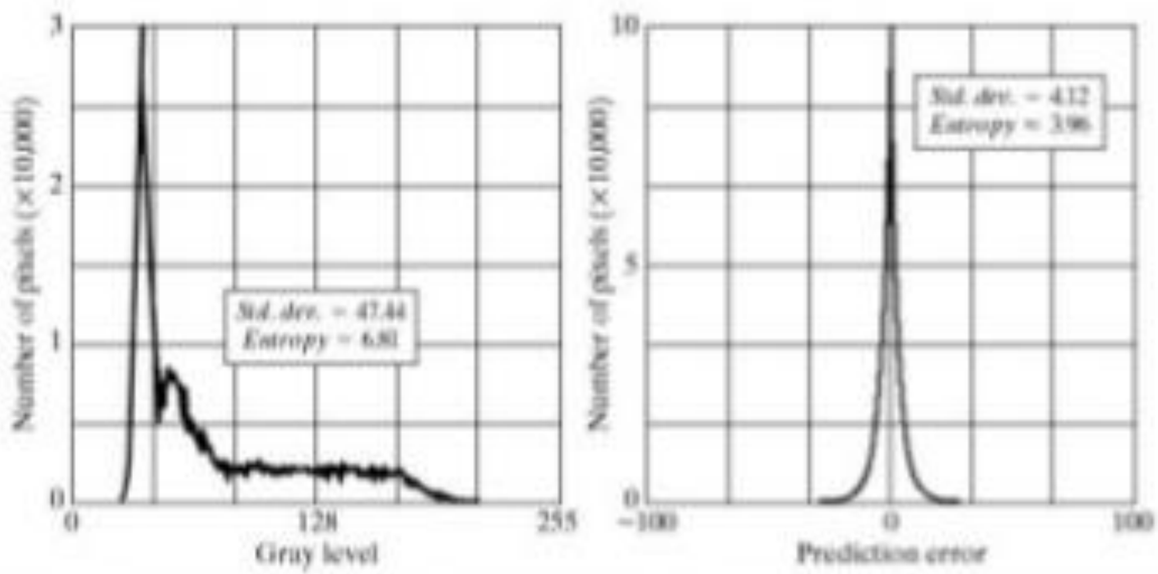
round – function used to denote rounding

$\alpha_i$ – for i = 1, 2, 3, ..... m are prediction coefficients.

$$\hat{f}(x,y) = round[\alpha f(x, y-1)]$$

**First-order linear predictor**

Prediction error image

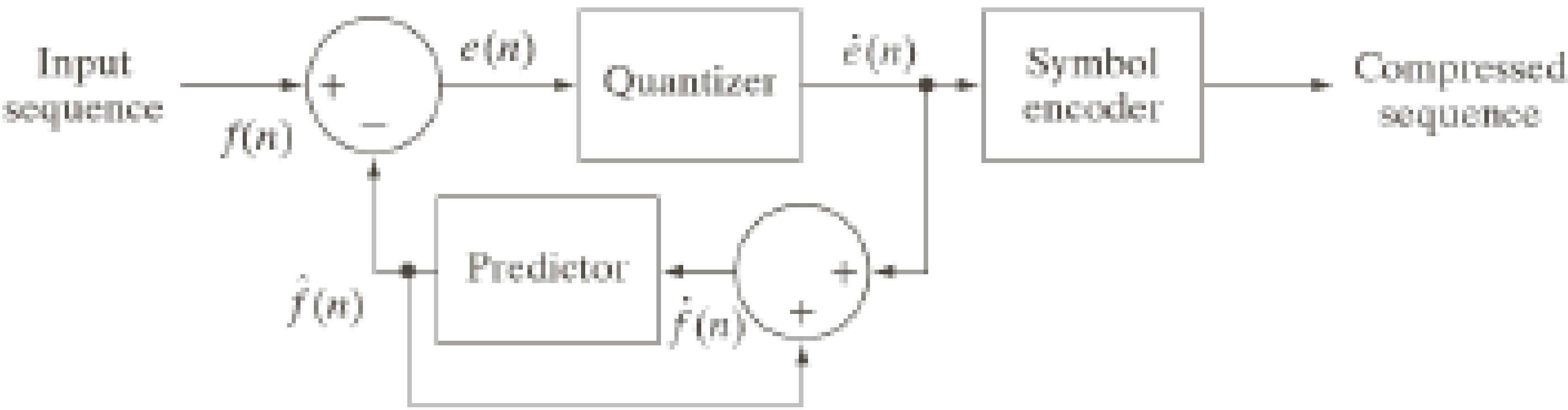



Histogram of original image and prediction error

# Lossy predictive coding

*The encoder expects a discrete samples of a signal f(n).*

1. A predictor is applied and its output is rounded to the nearest integer, $\hat{f}(n)$
2. The error is mapped into limited rage of values (quantized) $\dot{e}(n)$
3. The compressed stream consist of first sample and the mapped errors, encoded using variable length coding



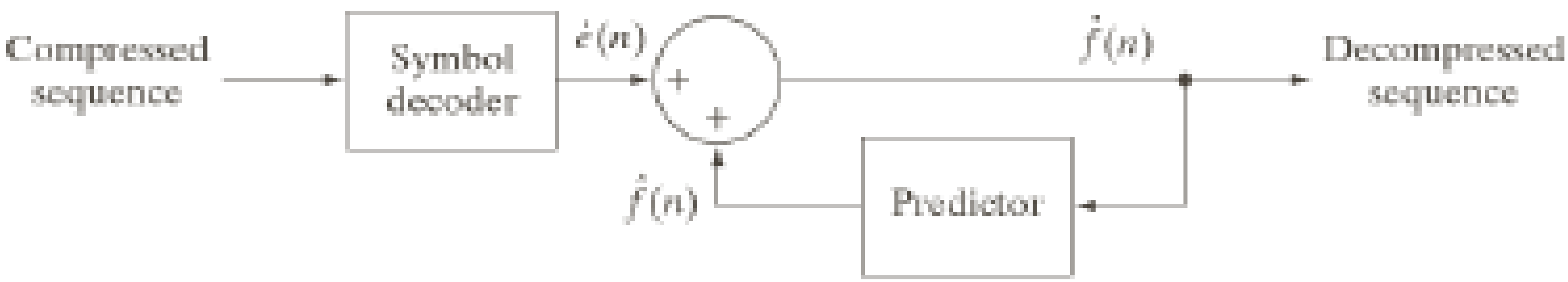
# Contd..

The decoder uses error stream to reconstructs an approximation of the original signal, $\dot{f}(n)$

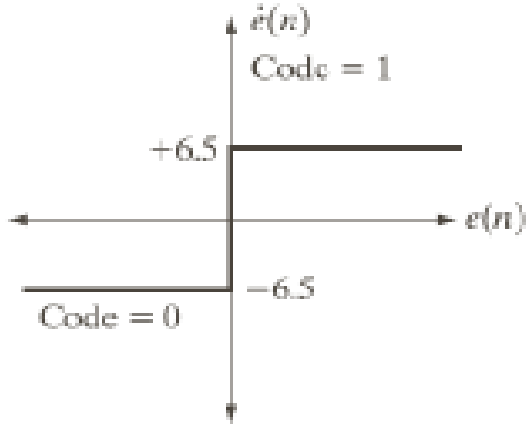
1. The predictor is initialized using the first sample.
2. The received error is added to predictor result.

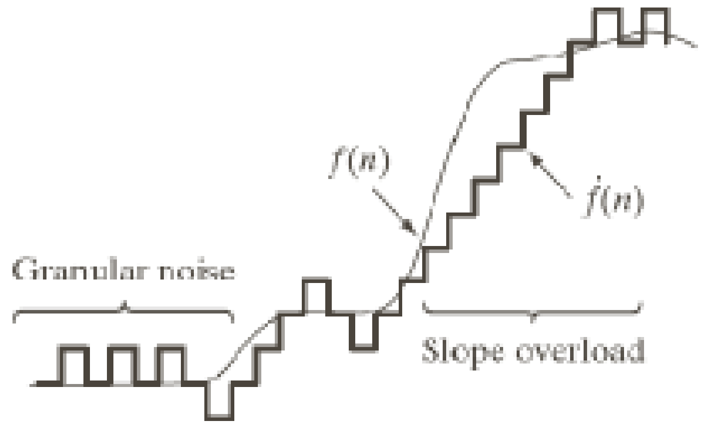
$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

$$\dot{e}(n) = \begin{cases} +\xi & e(n) > 0 \\ -\xi & otherwise \end{cases}$$



# Contd..



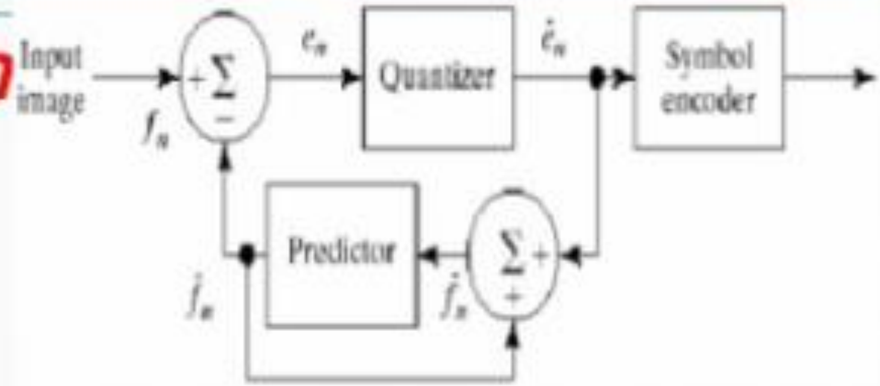




| Input | | Encoder | | | | Decoder | | Error |
|---|---|---|---|---|---|---|---|---|
| $n$ | $f(n)$ | $\hat{f}(n)$ | $e(n)$ | $\dot{e}(n)$ | $\dot{f}(n)$ | $\hat{f}(n)$ | $\dot{f}(n)$ | $f(n) - \dot{f}(n)$ |
| 0 | 14 | — | — | — | 14.0 | — | 14.0 | 0.0 |
| 1 | 15 | 14.0 | 1.0 | 6.5 | 20.5 | 14.0 | 20.5 | −5.5 |
| 2 | 14 | 20.5 | −6.5 | −6.5 | 14.0 | 20.5 | 14.0 | 0.0 |
| 3 | 15 | 14.0 | 1.0 | 6.5 | 20.5 | 14.0 | 20.5 | −5.5 |
| . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . |
| 14 | 29 | 20.5 | 8.5 | 6.5 | 27.0 | 20.5 | 27.0 | 2.0 |
| 15 | 37 | 27.0 | 10.0 | 6.5 | 33.5 | 27.0 | 33.5 | 3.5 |
| 16 | 47 | 33.5 | 13.5 | 6.5 | 40.0 | 33.5 | 40.0 | 7.0 |
| 17 | 62 | 40.0 | 22.0 | 6.5 | 46.5 | 40.0 | 46.5 | 15.5 |
| 18 | 75 | 46.5 | 28.5 | 6.5 | 53.0 | 46.5 | 53.0 | 22.0 |
| 19 | 77 | 53.0 | 24.0 | 6.5 | 59.6 | 53.0 | 59.6 | 17.5 |
| . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . |

# DPCM

## Lossy Compression

### Optimal Prediction




$$E\{e_n^2\} = E\{[f_n - \hat{f}_n]^2\}$$

$$\dot{f}_n = \dot{e}_n + \hat{f}_n \approx e_n + \hat{f}_n = f_n$$

$$\hat{f}_n = \sum_{i=1}^{m} \alpha_i f_{n-i}$$

$$E\{e_n^2\} = E\left\{\left[f_n - \sum_{i=1}^{m} \alpha_i f_{n-i}\right]^2\right\}$$

$$\sum_{i=1}^{m} \alpha_i \leq 1$$

**Differential Pulse Code Modulation (DPCM)**

# Contd..

**Prediction Error**

$$\hat{f}(x,y) = 0.97 f(x, y-1)$$ **Pred. #1**

$$\hat{f}(x,y) = 0.5 f(x, y-1) + 0.5 f(x-1, y)$$ **Pred. #2**

$$\hat{f}(x,y) = 0.75 f(x, y-1) + 0.75 f(x-1, y) - 0.5 f(x-1, y-1)$$ **Pred. #3**

$$\hat{f}(x,y) = \begin{cases} 0.97 f(x, y-1) & \textit{if } \Delta h \le \Delta v \\ 0.97 f(x-1, y) & \textit{otherwise} \end{cases}$$ **Pred. #4**

$$\Delta h = |f(x-1, y) - f(x-1, y-1)| \textit{ and } \Delta v = |f(x, y-1) - f(x-1, y-1)|$$

# Output

## *Prediction Error for different predictors*

FIGURE 8.24 A comparison of four linear prediction techniques.

# Lossy Compression

Add what is lossy compression, applications

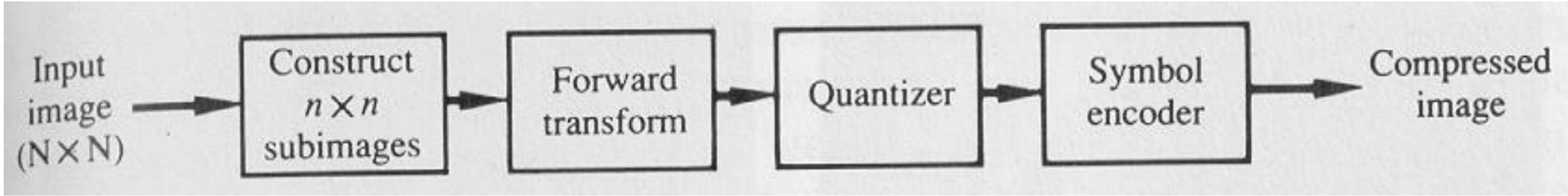
# Transform coding

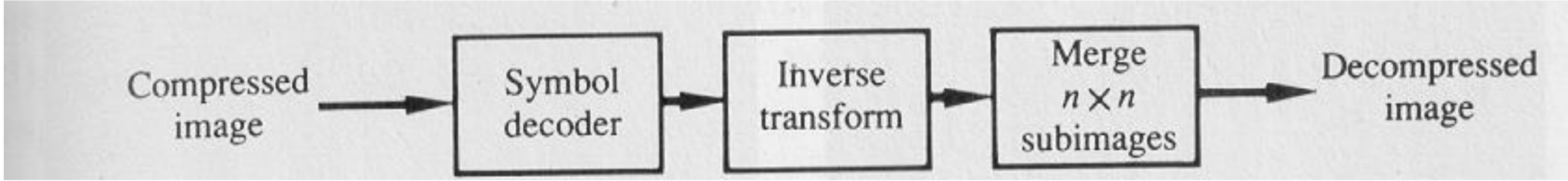
- In this coding scheme, transforms such as DFT and DCT are used to change the pixels in the original image into frequency domain coefficients (called transform coefficients).
- These coefficients have the energy compaction property i.e. energy of the original data being concentrated in only a few of the significant transform coefficients. Only those few significant coefficients are selected and the remaining are discarded.

# Contd..

- Transform the image into a domain where compression can be performed more efficiently (i.e., reduce interpixel redundancies).

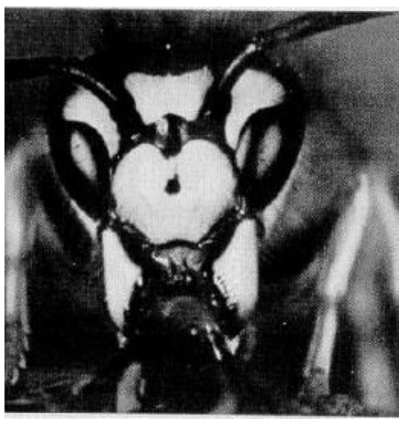
~ $(N/n)^2$ subimages



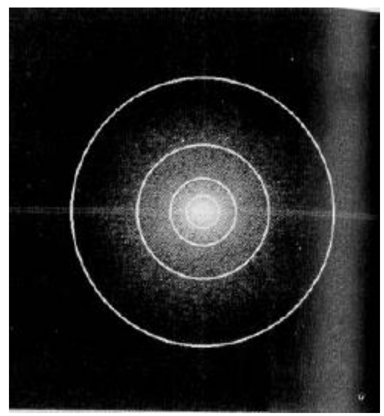



# Example: Fourier Transform

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



**K-1**



**K-1**

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

[redacted]

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

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

# Transform Selection

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

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

# DCT (Discrete Cosine Transform)

Forward: $$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos(\frac{(2x+1)u\pi}{2N})cos(\frac{(2y+1)v\pi}{2N}),$$

$u, v$=0,1,…,N-1

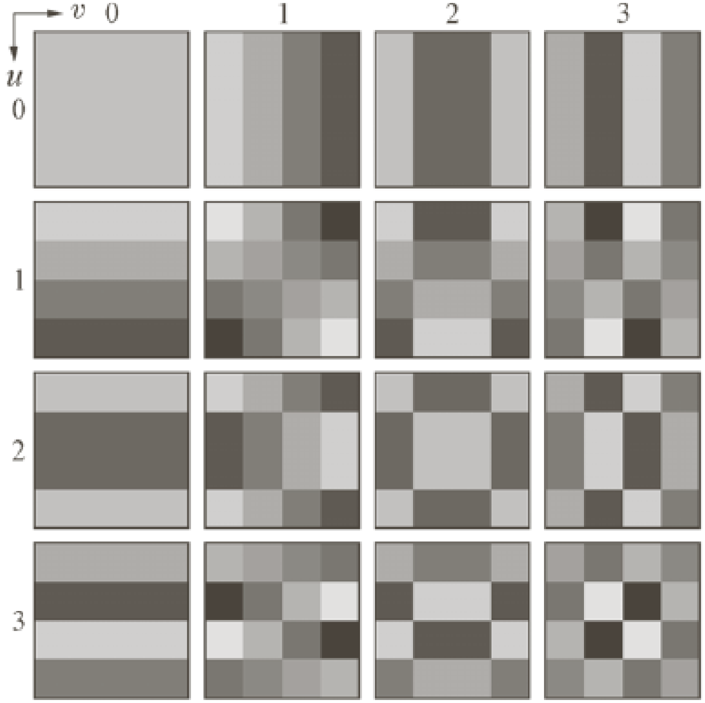
Inverse: $$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)cos(\frac{(2x+1)u\pi}{2N})cos(\frac{(2y+1)v\pi}{2N}),$$

$x, y$=0,1,…,N-1

$$\alpha(u) = \begin{cases}\sqrt{1/N} & \text{[redacted]} \\ \sqrt{2/N} & \text{[redacted]}\end{cases} \quad \alpha(v) = \begin{cases}\sqrt{1/N} & \text{[redacted]} \\ \sqrt{2/N} & \end{cases}$$

# DCT (cont'd)

- Set of basis functions for a 4x4 image (i.e., cosines of different frequencies).



# DCT (cont'd)

DFT WHT DCT

Using
8 x 8 subimages

64 coefficients
per subimage

50% of the
coefficients
truncated

RMS error: 2.32 1.78 1.13

# DCT (cont'd)

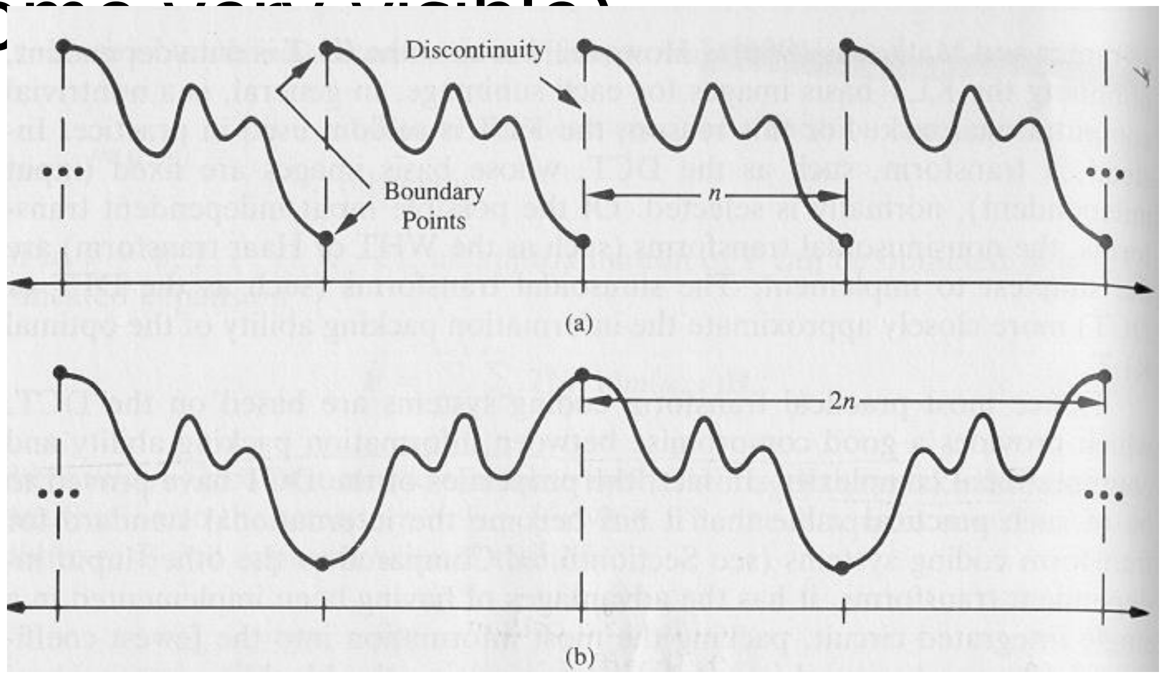
- DCT minimizes "blocking artifacts" (i.e., boundaries between subimages do not become very visible)

DFT

i.e., n-point periodicity gives rise to discontinuities!

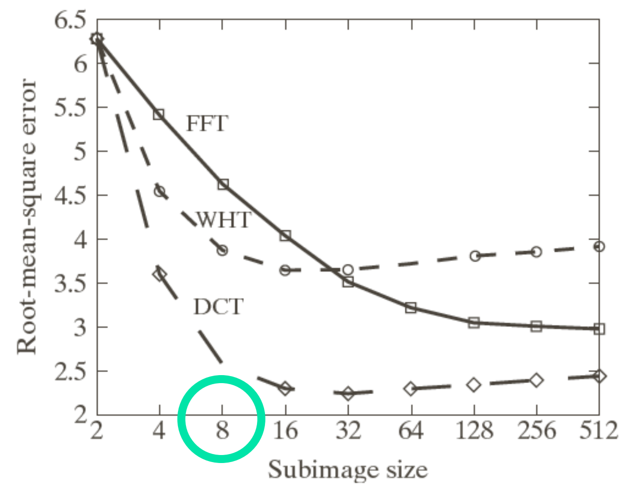
DCT

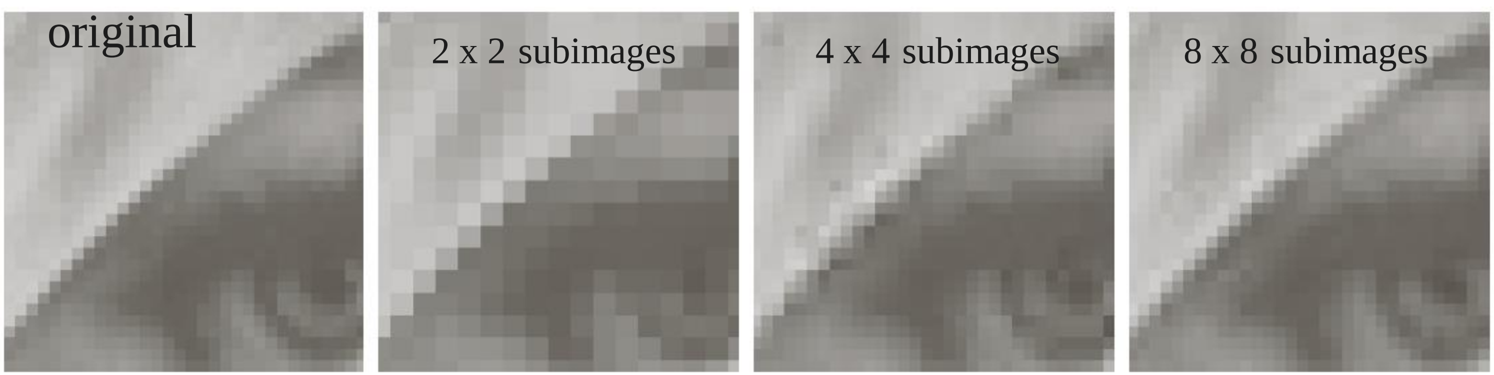
i.e., 2n-point periodicity prevents discontinuities!



# DCT (cont'd)

- Subimage size sel



# Wavelet coding

# Basics of image compression standards

- JPEG
- MPEG

# The JPEG Standard

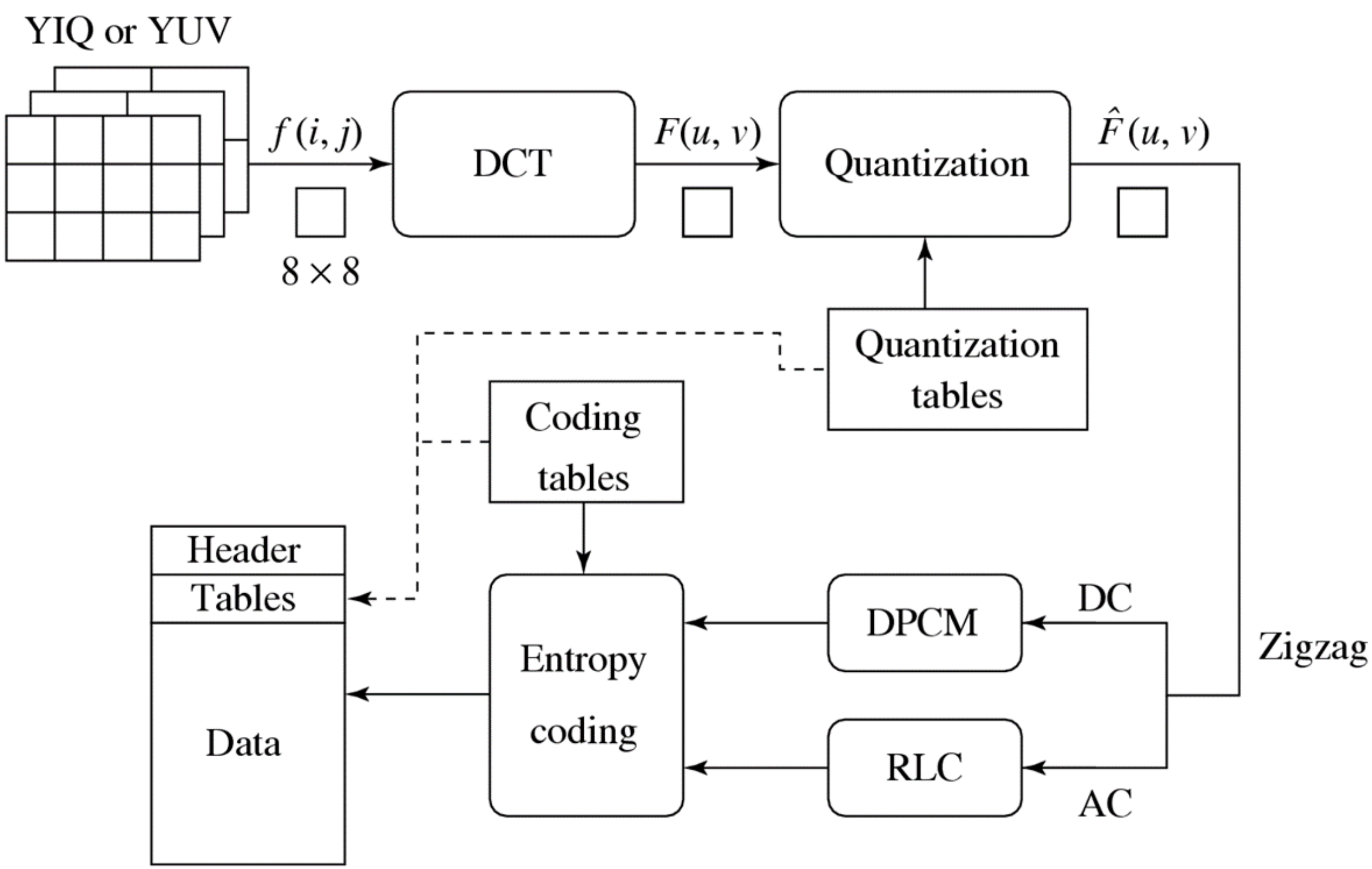
- JPEG is an image compression standard that was developed by the "Joint Photographic Experts Group". JPEG was formally accepted as an international standard in 1992.
- JPEG is a lossy image compression method. It employs a transform coding method using the DCT (Discrete Cosine Transform).

- An image is a function of i and j (or conventionally x and y) in the spatial domain.  The 2D DCT is used as one step in JPEG in order to yield a frequency response which is a function F(u, v) in the spatial frequency domain, indexed by two integers u and v.

# Observations for JPEG Image Compression

- The effectiveness of the DCT transform coding method in JPEG relies on 3 major observations:

- Observation 1: Useful image contents change relatively slowly across the image, i.e., it is unusual for intensity values to vary widely several times in a small area, for example, within an 8×8 image block.
- much of the information in an image is repeated, hence “spatial redundancy


YIQ or YUV
f(i, j)
8 × 8
DCT
F(u, v)
Quantization
F̂(u, v)
Quantization tables
Coding tables
Header
Tables
Data
Entropy coding
DPCM
RLC
DC
AC
Zigzag

# Main Steps in JPEG Image Compression

- Transform RGB to YIQ or YUV and subsample color.
- DCT on image blocks.
- Quantization.
- Zig-zag ordering and run-length encoding.
- Entropy coding

# Basics of MPEG

- Moving Picture Experts Group
- Established in 1988

- Generally produces better quality than the other formats such as:
  - Video for Window
  - Index and QuickTime
- MPEG audio/video compression can be used many applications:
  - DVD player
  - HDTV recorder
  - Internet Video
  - Video Conferences
  - Others

- MPEG-1 : a standard for storage and retrieval of moving pictures and audio on storage media
- MPEG-2 : a standard for digital television
- MPEG-4 : a standard for multimedia applications

- MPEG-7 : a content representation standard for information search
- MPEG-21: offers metadata information for audio and video files

# BASICS OF VECTOR QUANTIZATION