IMAGE COMPRESSION

INTRODUCTION :-

Data compression: It is the process of reducing the no. of bits needed to represent data. Compression of data saves storage capacity, speeds up file transfer, decreases the cost for storage hardware & network bandwidth.

Decompression :- The process of retrieving the compressed data without loss of information is called decompression

Data :- a means in which information can be expressed. It can be larger or smaller than information.

Data redundancy :- A data that contains irrelevant or repeated information.

let n_1 represent data & n_2 represent data redundancy after compression.

The relative redundancy of information is given as

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

where coding redundancy $C_R = \frac{n_1}{n_2}$

→ If $n_1 = n_2$; $C_R = 1 \Rightarrow R_D = 0$

this indicates no data redundancy & no need for compression

→ If $n_2 << n_1$; $C_R = \infty \Rightarrow R_D = 1$

this indicates high data redundancy & need for high amount of compression.

→ If $n_2 >> n_1$; $C_R \to 0 \Rightarrow R_D = -\infty$

This indicates that data after compression is large compared to original one

---

TYPES OF REDUNDANCIES :-

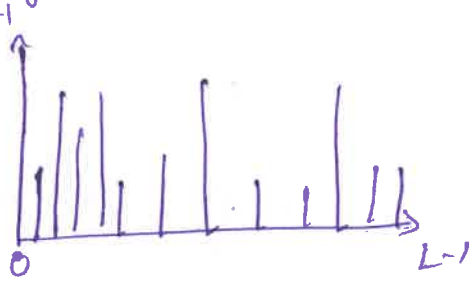
In Digital Image Processing, we have three types of redundancies

(1) Coding Redundancy
(2) Interpixel Redundancy
(3) Psycho visual Redundancy

(1) **Coding Redundancy :-**

It is associated with representation of information the information is represented in the form of codes. If the gray levels of an image are coded in a way that uses more code symbols than necessary to represent each gray level, then the resulting image is said to contain coding redundancy.

Ex :- Consider histogram of a image.



If we start coding histogram from dark levels as 00000000 to max. brightness 11111111, we observe dark levels do not require all 8 bits for coding. Hence to compress image, redundancy can be applied over these excess bits.

Disadvantage :- there is no correlation b/w pixels.

(2) Interpixel Redundancy :-

It is of two types :-

→ Inter-pixel spatial Redundancy :- It is due to the correlation b/w the neighbouring pixels in an image. That means neighbouring pixels are not statistically independant. The gray levels are not equally probable. The value of any given pixel can be predicted from the value of its neighbours that are highly correlated. Individual pixels have very less information.

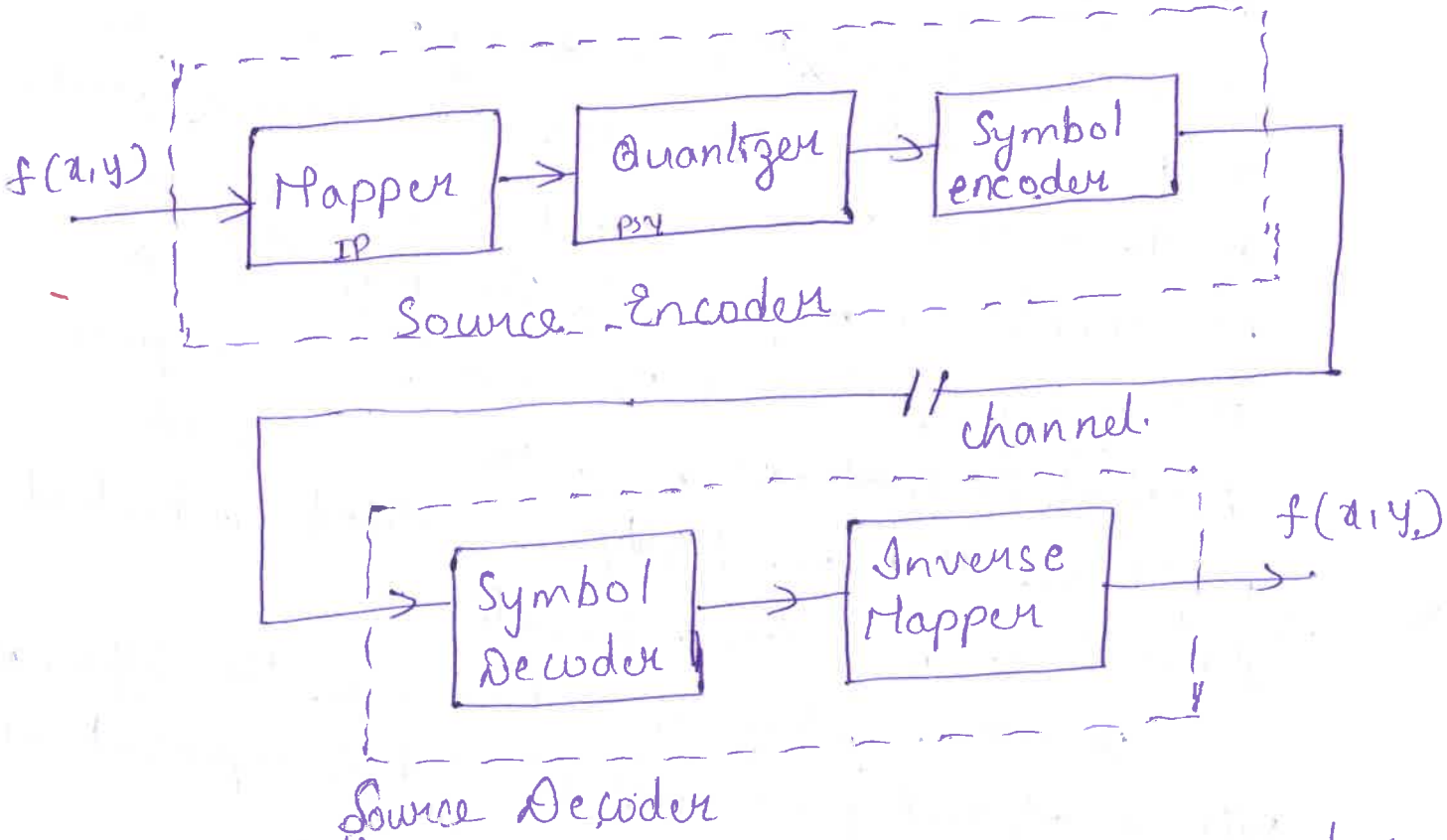
To reduce interpixel redundancy, the difference between adjacent pixels can be used to represent an image.

→ Interpixel Temporal Redundancy :-

It is due to statistical correlation b/w pixels from successive frames in a video sequence. This is also called inter frame redundancy Removing large amount of redundancy leads to efficient video compression.

(3) Psychovisual Redundancy :- This exists because human perception does not involve quantitative analysis of every pixel. Elimination of this redundant data results in loss of quantitative information called as quantization which results in lossy data compression

IMAGE COMPRESSION MODELS :-



Source Encoder :- It is responsible for reducing or eliminating any redundancies in the i/p image.

Mapper :- It transforms i/p data into a format designed to reduce interpixel redundancies in the i/p image.

Symbol Encoder :- It uses fixed or variable length coding to the quantizer o/p.

Channel :- A medium of transmission

Source Decoder :- Decodes the received data. It consists of a symbol decoder & inverse mapper

Quantizer :- It is used to reduce psychovisual redundances

Symbol Decoder :- Decodes the information that is encoded. Inverse quantizer is not used because it results in irreversible loss of information

Inverse Mapper :- Maps data to original image.

TYPES OF COMPRESSION :-

1) Lossless Compression :-

In lossless compression, the distorted data file is identical to original one. This compression is used for data such as executable code, word processing files, tabulated numbers etc. Even a single bit of information cannot be misplaced.

2) Lossy Compression :- In this method, data need not be stored or transmitted in perfect condition. Noise is allowed.

LOSSLESS (ERROR FREE) COMPRESSION TECHNIQUES :-

a) Run length coding
b) Variable length coding or Huffman Coding
c) Lempel - Ziv - Welch (LZW) coding
d) Lossless predictive coding

(a) Run length Coding :- When we take a data file, same character may be repeated a no. of times in a row.

Ex :- spaces, indent paragraphs, tables, charts etc

Digital signals can also have runs of same value, indicating that the signal is not changing.

Ex :- Image of night sky represents long runs of the characters representing black background. Similarly, digitized music might have a long run of zeroes b/w songs.

This coding is the simplest method of coding in lossless compression techniques

To represent it let us consider a data stream as given below :

1 7 8 5 4 0 0 0 9 7 5 16 0 45 23 0 0 0 0 3 6 7 0 0 8

→ Start comprening data. Verify bits from left to right

→ Each time a zero is encountered in the i/p, two values are written to o/p file. The first zero is a flag that indicates runlength in beginning. The second value represents no. of zeroes in the run.

→ If runlength is longer than two bits, compression will take place.

→ More single zeroes can make a file larger than original.

Encoded bits: 1 7 8 5 4 0 3 9 7 5 16 0 1 45 23 0 5 3 6 7 0 2 8

(0 3 → flag, no. of zeroes.)

The i/p data can be considered as individual bytes, or groups of bytes such as floating point numbers. This coding technique can be used on only one of the characters, many or all of characters

Ex 2:- 2 4 8 10 5 0 0 0 2 18 21 0 1 2 6 8 9 0 0 0 7 6

Encoded 2 4 8 10 5 0 3 2 18 21 0 1 1 2 6 8 9 0 3 7 6

(2) Variable length Coding (Huffman coding)

It is one of the most popular techniques for removing coding redundancy.

Procedure :-

(1) Arrange source symbols in descending order of probability.

(2) The two source symbols of lowest probability are assigned a_0 & a_1.

(3) Add these two source symbols to obtain a new source symbol.

(4) Repeat the above steps until source symbols are reduced to two.

(5) The code for each source symbol is found by working backward & tracking the sequence of 0's & 1's.

Ex :- Consider 6 source symbols with probabilities

[0.1, 0.4, 0.1, 0.04, 0.06, 0.3]

Step 1 :- Arrange source symbols in descending order.

[0.4, 0.3, 0.1, 0.1, 0.06, 0.04]

step 2 :- Assign the probabilities $a_0, a_1, \ldots$ starting from lowest source symbol.

0.4	0.3	0.1	0.1	0.06	0.04
a_5	a_4	a_3	a_2	a_1	a_0

step 3 :- Add these two lowest source symbols to obtain a new source symbol ($a_0 + a_1 = 0.04 + 0.06 = 0.1$

step 4 :- Repeat the process with other source symbols till source symbols reduce to two

original source		source reduction			
symbol	Probability	1	2	3	4
a_5	0.4	0.4	0.4	0.4	0.6
a_4	0.3	0.3	0.3	0.3	0.4
a_3	0.1	0.1	0.2	0.3	
a_2	0.1	0.1	0.1		
a_1	0.06	0.1			
a_0	0.04				

step 5 :- Work backwards starting with smallest source until original source is reached. The minimal length binary code for a two-symbol source are 0 & 1.

Signal Source

symbol	Probability	code	1	code	2	code	3	code	4	code
a_6	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_5	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_4	0.1	011	0.1	011	0.2	010	0.3	01		
a_3	0.1	0100	0.1	0100	0.1	011				
a_2	0.06	01010	0.1	0101						
a_1	0.04	01011								

the average length of this code is

L_{avg} = [No. of bits in code × probability]

$\Rightarrow L_{avg} = [0.4 \times 1] + [0.3 \times 2] + [0.1 \times 3] + [0.1 \times 4]$
$+ [0.06 \times 5] + [0.04 \times 5]$

$= 0.4 + 0.6 + 0.3 + 0.4 + 0.3 + 0.2$

$\Rightarrow L_{avg} = 2.2$ bits / pixel.

Disadvantages of variable length or Huffman coding:-

1) Produces rounding errors.
2) Efficiency less than arithmetic coding
3) Code word lengths must be integers.

(8) Arithmetic Coding :-

In arithmetic coding, an entire sequence of source symbols [block coder or message] is assigned a single arithmetic code word. The code word defines an interval of real no.s b/w 0 & 1. As the no. of symbols in the message increases, the interval used to represent it becomes smaller & the no. of information units required to represent interval becomes larger. Each symbol of the message reduces the size of the interval in accordance with problem.

Ex :- Suppose we have alphabets a, b, c, d, e with probability of occurance of 30%, 15%, 25%, 10% & 20% in a data.

symbol	Probability	Range
a	0.30	[0.00 - 0.30]
b	0.15	[0.30 - 0.45)
c	0.25	[0.45 - 0.70)
d	0.10	[0.70 - 0.80)
e	0.20	[0.80 - 1.00)

value = (Encoded value − lower bound of new symbol) ÷ current range

ode the three symbol string encoded as

ng of first symbol starts with search range
ch 0.20 lies that is [0.00, 0.30).
20 lies within [0.00, 0.30)

encodes 'a'

ove effect of 'a'
nt Range = (0.30 − 0.00) = 0.30
ded value = (0.20 − 0.00) ÷ 0.30
= 0.67.

for decoding second symbol 0.67 lies
[0.45, 0.70)

encodes 'c' :-

oving the effect of 'c'
t Range = 0.70 − 0.45 = 0.25
ed value = (0.67 − 0.45) ÷ 0.25 = 0.88.
ded value 0.88 lies in the range [0.80 − 1],
e third symbol is 'e'

is included
is excluded.

ll previous

= 0.45 (0.30 + 0.15)
0.30 + 0.15 + 0.25
= 0.70

wer bound
ent range ×
und of new symbol)
ent range ×
nd of new symbol)
sth upper bound

) = 1
) = 0.30
= 0.00

(3) LZW (Lempel-Ziv Welch) Coding or Compression

→ This is a lossless compression technique.

→ This compression algorithm is used in Portable Document Format (PDF) & Graphics Interchange format (GIF)

→ simple to implement

→ Idea is based on reccurring patterns to save data space.

→ LZW compression works by
  - > reading a sequence of symbols
  - > grouping symbols into strings
  - > converting strings into codes

→ LZW compression uses code table with 4096 entries.

→ Usually ASCII codes are from 0-255 representing lower case, upper case characters, numbers, symbols, special characters etc. Each symbol or character is assigned a Byte.

→ LZW compression table contains the ASCII codes as first 256 entries & the rest of the table is empty.

→ As LZW encoding continues, it identifies repeated sequences in the data & adds them to code table.

⇒ Decoding is done by translating each code compressed file & translating it through the table to find what character or characteristics represents.

LZW encoding algorithm :-

1) At the start, the dictionary contains all possible individual characters & P is empty
2) C = next character in the character stream
3) Is the string P+C present in the dictionary
   a) If it is present, P = P+C (Extend P with C)
   b) If not
      → o/p the code word which denotes P to the code stream
      → add the string P+C to the dictionary
      → P = C (P now contains only character C)

Example :- Encode the string of characters

w a b b a w a b b a

A) Initialize table with string of characters

w a b b a w a b b a

Now in dictionary a = 1 ; b = 2 ; w = 3

Step 1 :- Identify the characters in string & define index

P = ___ (Initially P is empty)

C = w (first character of string)

P = P+C = 0+w = w ⇒ **P = w**

Index	Dictionary
1	a
2	b
3	w
4	wa
5	ab
6	bb
7	ba
8	aw
9	wab
10	bba

Step 2 :- P = w

c = a

P + c = wa

Hence add to the table

Assign code [P = w = 3] [o/p]

Now [P = c = a]

Step 3 :-

P = a

c = b

P + c = ab

(b) Not in table. Add to table.

Assign code [P = a = 1] o/p

Step 4 :- Now P = c = b

c = b → P

P + c = bb

No entry. Add to table.

Assign code [P = b = 2] o/p

Step 5 :- Now P = b

c = a → P

P + c = ba.

No entry. Hence add to table & assign code

[P = b = 2] o/p

step 6 :- Now P = a
c = w → P

P + c = aw

No entry - Hence add to table & assign code

P = a = 1 o/p

step 7 :- Now P = w
c = a

P + c = wa → P

Value is in string table P = P + c = wa

so no o/p.

step 8 :- Now P = wa
c = b

P + c = wab

No entry in table. Add to table & assign code

P = wa = 4 o/p

step 9 :- Now P = b
c = b

P + c = bb

Value is in string table P = P + c = bb

so no o/p

step 10 :- Now p = bb
c = a
p+c = bba

No entry. Add to table & assign code

p = bb = 6	- o/p

p = a — o/p 1

string ends. Assign code

Hence the code generated for a given string is

3 1 2 2 1 4 6 1

LZW de-compression algorithm :-

1) At the start the dictionary contains all possible characters
2) cw = first code word
3) o/p the string cw to the char. stream
4) pw = cw
5) cw = next code word.
6) Is the string cw present in the dictionary,
If it is
a) o/p the string cw, to the char. stream
b) charp : = string pw
c) charc : = the first character of the string, cw
d) add the string p+c to dictionary

If not,

a) P = string PW;

b) C = the first character of string PW

c) o/p the string P+C to the charstream & add it to the dictionary. (now it corresponds to CW.)

7) Are there more code words in the code stream

a) If YES, go back to step 4

b) If not, END

Example :-

Consider the code 3 1 2 2 1 4 6 1 and decode using LZW compression technique

Given Dictionary 1 = a ; 2 = b ; 3 = w

Code word

3 1 2 2 1 4 6 1

step 2, 3 & 4

step 2&3 string CW = 3 → char CW = (w) - o/p

step 4 string PW = CW = 3

step 5 :-

string PW = 3 → char PW := w

string CW = 1 → char CW = (a) - o/p

P = P+C = wa → Add to dictionary

Index string	Dictionary character	
1	a	Given step 1
2	b	
3	w	
4	wa	
5	ab	
6	bb	
7	ba	
8	aw	
9	wab	
10	bba	

pw = cw

Now string pw = 1 ; char pw = a
string cw = 2 ; char cw = (b) dp.
p = p+c = ab → Add to dictionary

Now string pw = 2 ; char pw = b
string cw = 2 ; char cw = (b) dp
p = p+c = bb → Add to dictionary

Now string pw = 2 ; char pw = b
string cw = 1 ; char cw = (a) dp
p = p+c = ba → Add to dictionary

Now string pw = 1 ; char pw = a
string cw = 4 ; char cw = (wa) dp
p = p+c = ~~aaaa~~ aw → Add to dic
(consider only first letter of cw)

Now string pw = 4 ; char pw = wa
string cw = 6 ; char cw = (bb) dp
p = p+c = wab — Add to dictionary
(consider only first letter of cw)

Now string pw = b → char pw = bb

string cw = 1 → char cw = (a) − dp

p = p+c = bba → Add to dictionary

Now write all the dps in sequence to obtain decoded string

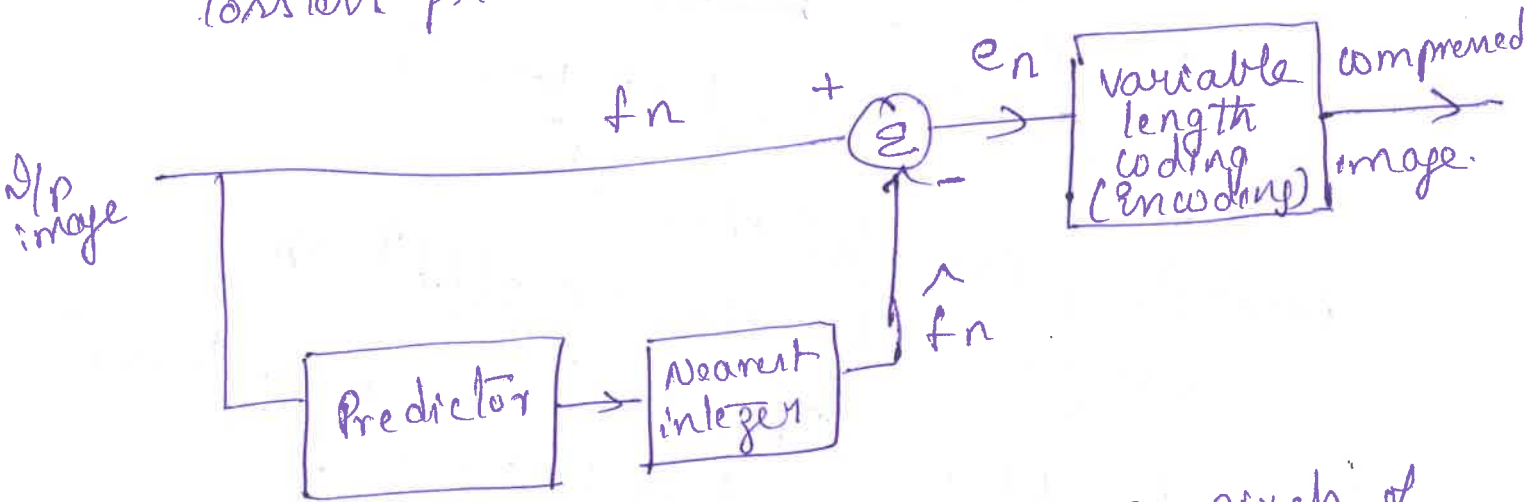
wabbawabba

---

(4) Lossless Predictive Coding :-

This coding technique helps in removing Interpixel Redundancy by predicting new information which is obtained by taking the difference b/w actual & predicted value of that pixel.

Fig shows the basic components of lossless predictive encoding.



At the encoder side, successive pixels of image are applied as input. On the basis of

successive pixels the prediction will generate an anticipated value $\hat{f}_n$ for the current pixel f_n.

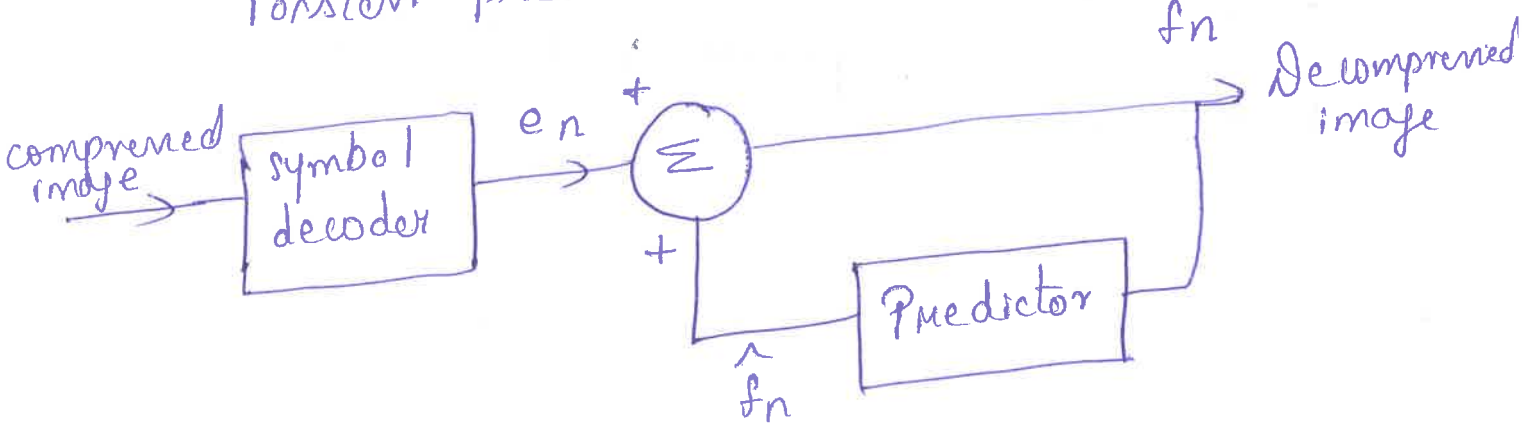
The nearest integer is a logic circuit that causes to round-off the value generated by predictor. The error e_n is the difference b/w the actual & predicted value.

$$e_n = f_n - \hat{f}_n$$

This error is also called prediction error.

To e_n, variable length coding is applied to remove coding redundancy.

Fig shows the basic components of a lossless predictive decoder



The decoder performs the inverse operation

$$f_n = e_n + \hat{f}_n$$

If the prediction is a linear prediction, then

$$\hat{f}_n = \text{round}\left[\sum_{i=1}^{m} a_i f_{n-i}\right]$$

where m is the order of linear prediction

a_i – Prediction coefficients where $i = 1, 2, \ldots m$

For a 1-D linear predictive image coding

$$\hat{f}(x,y) = \text{round}\left[\sum_{i=1}^{m} a_i f(x, y-i)\right]$$

$\hat{f}(x,y)$ is a function of previous pixels on current line alone.

In 2D images, the prediction is a function of previous pixels in a left to right, top to bottom scan of an image. In 3D case, it is based on there pixels & previous pixels of preceeding frames.

LOSSY COMPRESSION :-

There techniques are used where some error is tolerable. This loss is called distortion lossy compression techniques are as follows:

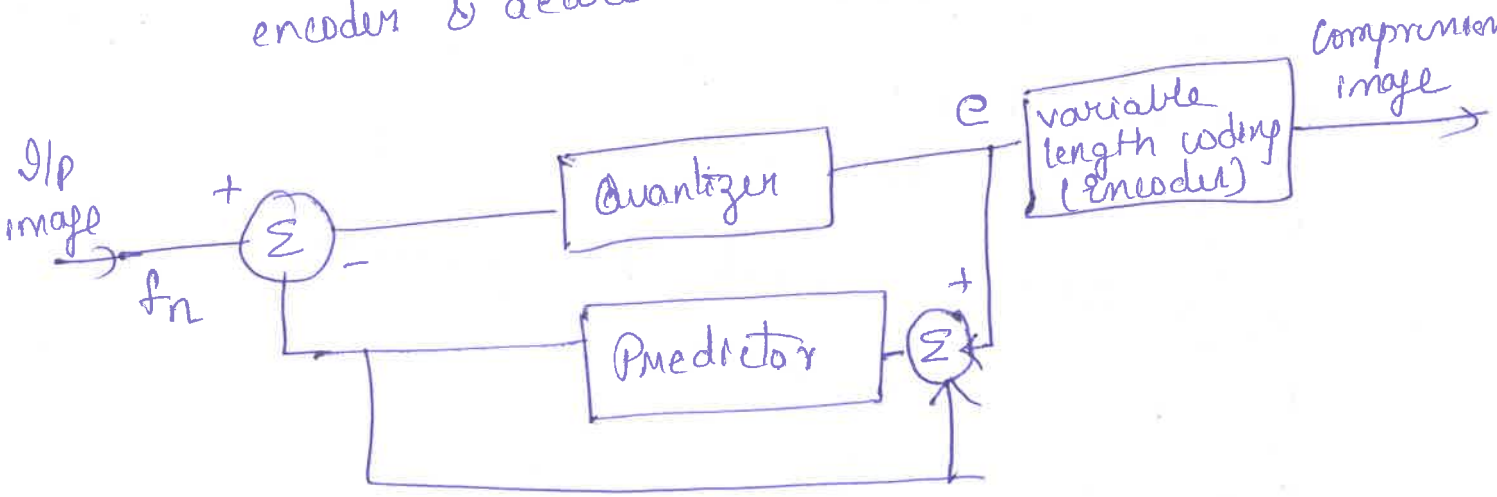
(1) Lossy predictive coding
(2) Transform coding (or Block Transform coding)
(3) Wavelet coding.

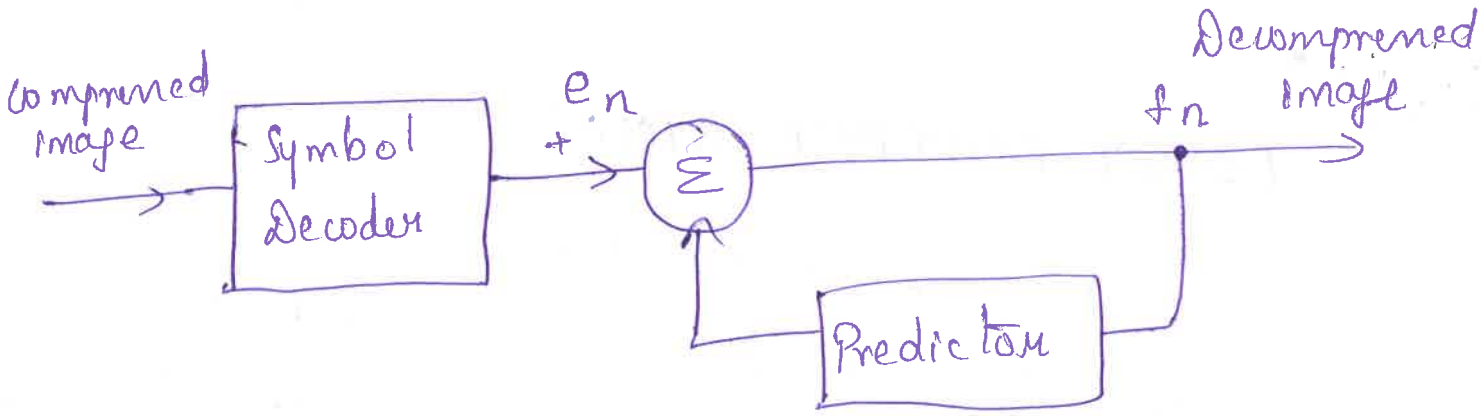
1) Lossy Predictive Coding :-

Predictive coding can also be implemented for a lossy compression scheme. The basic difference b/w lossless & lossy predictive coding system is that we have a quantizer in lossy compression system

Ex :- DPCM, DM

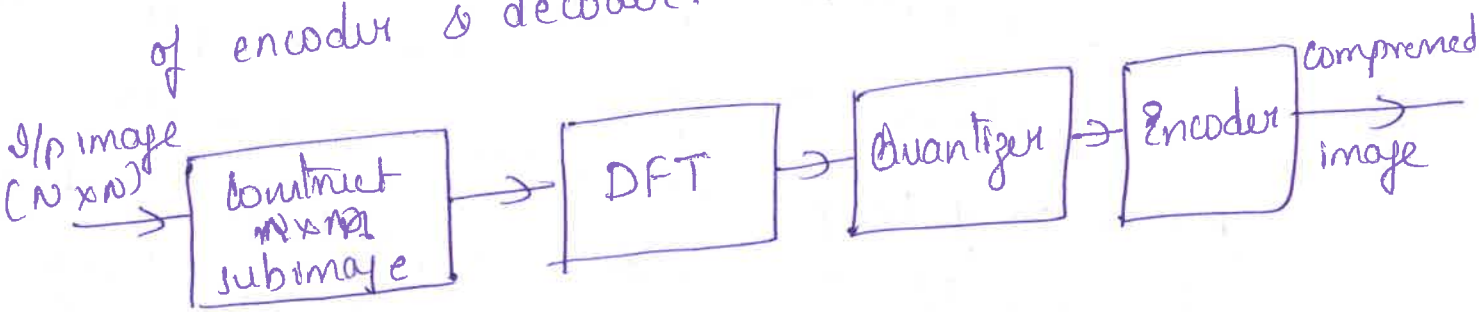
Fig shows block diagram of lossy predictive encoder & decoder

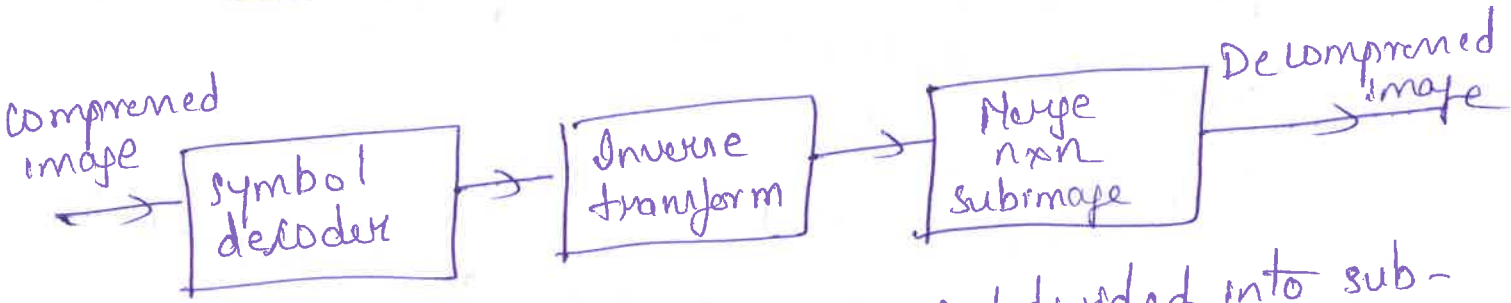




(2) Block Transform Coding :-

In this technique, we apply compression on the transform of an image. Fig show the block diagram of encoder & decoder.





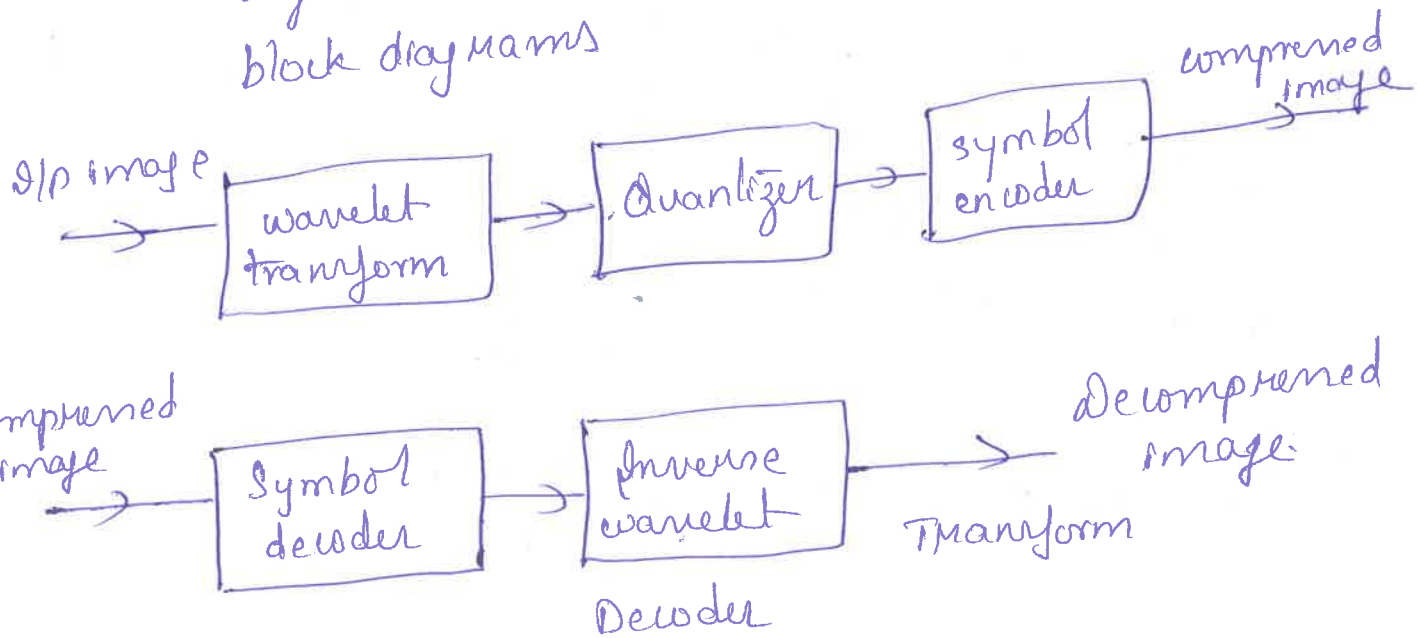
An I/p N×N image is subdivided into sub-images of n×n. Most popular subimage size are 8×8 & 16×16. Thus image is converted into small pack of information that are easy to process

→ Apply DFT & generate transformation coefficient by sub images. DCT, wavelet & K-L transforms can also be used

→ Quantizer removes redundant information
→ coding redundancy is removed by applying variable length coding technique in encoder.
→ the best use of transform coding is JPEG.
→ The decoder performs the reverse of encoder

(3) Wavelet Coding:-

Wavelet transform function subdivides the image into subimages & generates transform coefficient Hence we do not need the block of subimage construction
→ Quantizer removes redundant information
→ Symbol encoder removes coding redundancy.
Figs show wavelet encoding & decoding block diagrams



DIGITAL IMAGE WATER MARKING :-

The compression methods & standards available make wide distribution of images (photos, videos) over digital media & internet. There available images can be copied repeatedly without error thereby, making the rights of their owners at risk. Illegal duplication may be discouraged by inserting one or more items of information collectively called watermark. The water mark is inserted into vulnerable images in such a way that they are inseparable from the images themselves. Water marks protect the rights of owners in variety of ways like

a) copy right identification - provides proof of owner ship
b) User identification or finger printing - Identity of legal users can be encoded in water marks
c) Authenticity determination - guarantees image has not been altered.
d) Automated monitoring - Royalty collection & monitoring of illegal users
e) Copy protection - Specifies rules of image usage & copying.

Types of watermarks :-

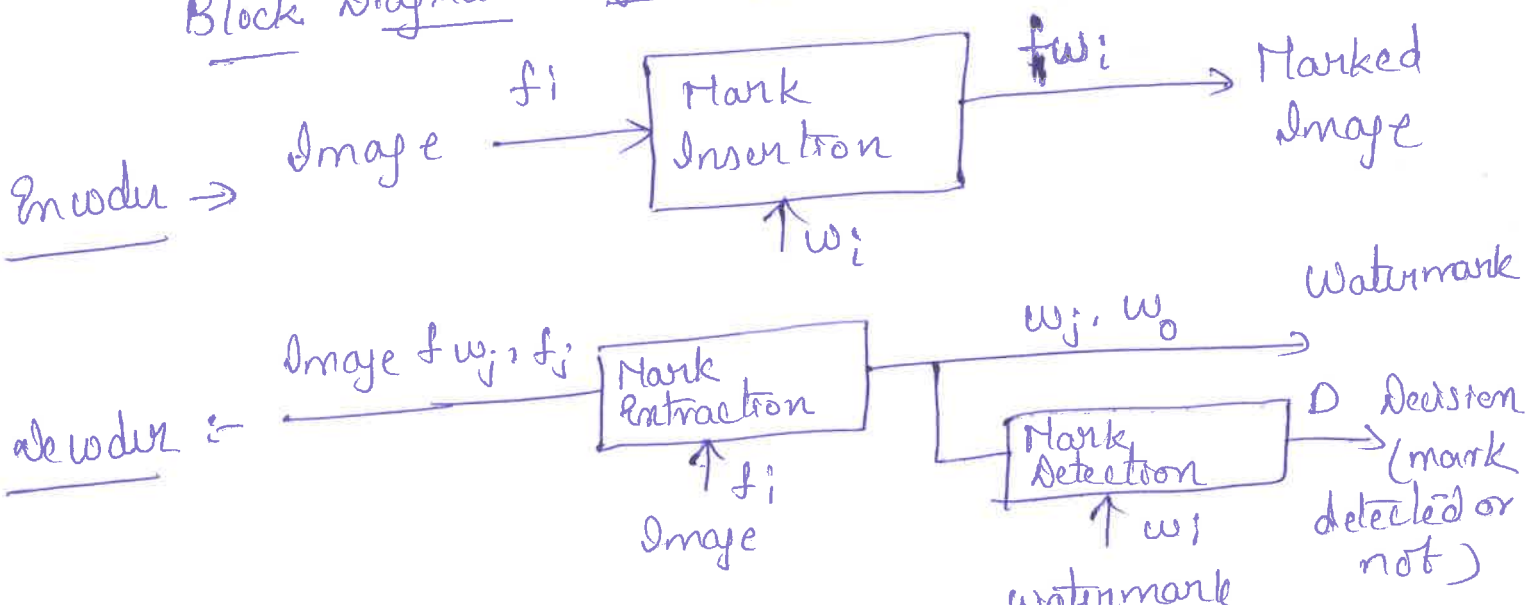
1) Visible watermark :- The logo/signature that is watermarked by this technique is visible on the background of the digital image.

2) Invisible Fragile Watermark :- In this technique, the watermark logo is not visible to naked eye. These watermarks are destroyed by any modification of the images in which they are embedded.

3) Invisible Robust watermark :-

These watermarks are designed to survive image modification whether the so called attacks are inadvertent or intentional

Block Diagram of a Image Watermarking system :-



Encoder inserts watermark w_i into image f_i producing watermarked image fw_i

Decoder extracts & validates the presence of w_i in watermarked i/p f_{w_j} or unmarked i/p f_j.

If w_i is visible, the decoder is not needed. If it is invisible, the decoder may or may not require a copy of f_i & w_i to do its job. If f_i and/or w_i are used, the watermarking system is known as a private or restricted-key system; if not it is a public or unrestricted-key system.

To determine the presence of w_i in an image, the decoder must correlate extracted watermark w_j with w_i & compare the result to a predefined threshold. The threshold sets the degree of similarity that is acceptable for a match.