Unit - 4

Image Compression.

The term Image Compression refers to the process of reducing the amount of data required to represent the given quantity of information.

Data & information are not synonymous, data means by which the information is conveyed. (represented)

Data also contains Nonessential data known as data redundancy.

Data redundancy is a central issue in digital image compression. It is a mathematically qualified entity. not a abstraction.

If n_1 & n_2 denotes the no of information-carrying units in two data sets that represents the same information, the relative data redundancy R_D of first data set can be defined as $R_D = 1 - \frac{1}{C_R}$

$C_R = \frac{n_1}{n_2}$ C_r = Compression ratio.

1. $n_2 = n_1$ $C_R = 1$ $R_D = 0$ $\Rightarrow$ representation contains no redundant data.
2. $n_2 \ll n_1$ $C_R = \infty$ $R_D = 1$ $\Rightarrow$ highly redundant data. significant Compression
3. $n_2 >> n_1$ $C_R = 0$ $R_D = -\infty$ $\Rightarrow$ 2nd data set contains more data set than the original one

4. C_R & R_D lie in the open interval $(0, \infty)$ $(-\infty, 1)$

In Digital image Compression. There are 3 basic data redundancies
1. Coding redundancy
2. Interpixel redundancy
3. Psychovisual redundancy

1. Coding Redundancy

Let us assume r_k – represents the gray levels of an image

each r_k appears with the probability – $P(r_k)$.

$\therefore P(r_k) = \frac{n_k}{n}$ $k = 0, 1, 2 \ldots L-1$ L – no of gray level

n_k – no of times the k^{th} gray level appears.

if the no of bits used to represent each value of r_k is $l(r_k)$ then the average no of bits required to represent each pixel

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

Total no of bits required to code an $M \times N$ image is $M \times N \times L_{avg}$.

* Assigning fewer bits to the more probable gray levels than to the less probable ones achieves data compression.

This process is ~~known as~~ referred as Variable-length Coding

* Assigning equal no of bits all the gray level in a set is reffered as fixed-length coding.

~~Interpixel Redundancy~~

* Coding Redundancy is due to poor selection of coding technique
* Coding technique assigns a unique code for all symbols of message
* wrong choice of coding technique creates extra bits called redundancy
* Coding Redundancy = Avg bits used to code – Entropy

$$= \sum_{k=0}^{n} l(r_k) P(r_k) - \sum_{i=1}^{n} P_i \log_2(P_i)$$

* Examples – Huffman code & arithmatic code.
* Entropy is a measure of the no of bits required to encode image data. High entropy means → More detailed the image will be

Inter pixel redundancy.

a) Inter pixel Temporal redundancy is the statistical Co-relation b/n pixels from successive frames in vedio sequences

* Temporal redundancy is also called interframe redundancy
Temporal redundancy can be exploited using motion Compensated predictive coding.

* Removing large amount of redundancy leads to efficient Vedio Compression.

** b). Inter pixel Spatial Redundancy -

Inter pixel redundancy is due to the Co-relation b/n the neighboring pixel in an image.

That means neighboring pixels are not statistically independent. The gray levels are not equally probable

The value of any given pixel can be predicated from the value of its neighbors that is they are highly correlated.

The information carried by individual pixel is relatively small. To reduce the interpixel redundancy the difference b/n adjacent pixels can be used to represent an image.

Ex: Consider an image with a constant background.

It can be solved by algorithms like.
Predictive coding, Bitplane Algorithm, Runlength coding
Dictionary based

Physcho - Visual Redundancy.

* The eye and the brain do not respond to all visual info with same sensitivity.
* Some information is neglected during the processing by the brain; elimination of this information does not affect the interpretation of the image by the brain.
* Edges & textural regions are interpreted as important features and the brain groups & correlates such grouping to produce its perception of an object.

Chromatic Redundancy.

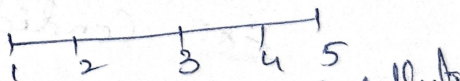
* It refers to the presence of unnecessary colors in an image.
* The color channels of color images are highly correlated & human visual system cannot perceive millions of colors.
* Therefore the colors that are perceived by human visual system can be removed without affecting the image quality.

Fidelity → Accurate reproduction of data.

Distortion → is difference b/n original & Reconstructed image

Objective fidelity - Error, SNR, PSNR

Subjective fidelity → based on rating

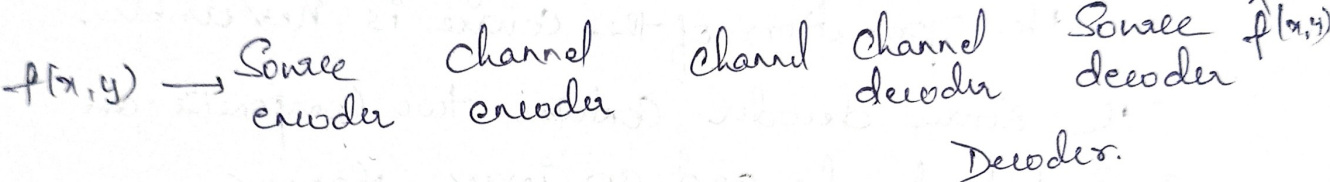


Considering group of Observers

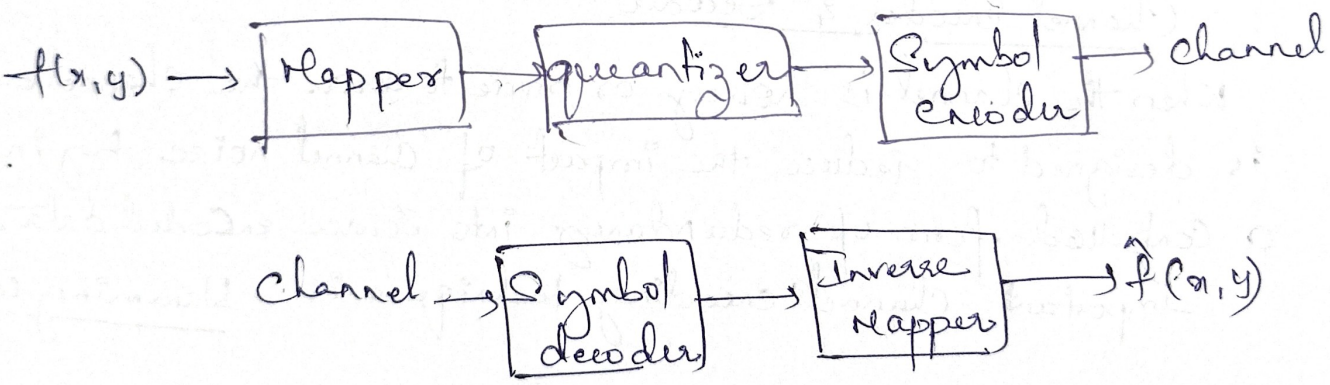
Image Compression Models

The Compression system consists of two distinct structural blocks. an encoder and a decoder.

An input image $f(x,y)$ is fed into the encoder, which creates a set of symbols from the input data. After transmission over the channel, the encoder representation is fed to the decoder, where a reconstructed output image $\hat{f}(x,y)$ is generated.



Both encoder & decoder consists of relatively independent subblocks. as shown below.



- The Source encoder is responsible for reducing or eliminating any coding, interpixel or psychovisual redundancies. in the i/p image.
- * Mapper reduces the interpixel redundancies and it is reversible.
  Eg. Runlength coding

The quantizer block reduces the psychovisual redundancies and also reduces the accuracy of the mapper's output in accordance with some pre-established fidelity criterion.

The Symbol coder creates a fixed or variable length code to represent the quantizer output & maps the output in accordance with the code. It assigns the fewer bits to the most frequently occuring output values. Thus reduces coding redundancy.

The operation of the source is Reversible.

The source decoder contains two components as a Symbol decoder and an inverse mapper.

Since quantization results in irreversible information loss, and inverse quantizer block is not included in general source decoder model.

Channel Encoder & Decoder.

When the channel is noisey or prone to error. the channel encoder is designed to reduce the impact of channel noise by inserting a controlled form of redundancy into source encoded data.

Important channel encoding technique is Hamming code.

Run length coding

* RLC → exploits the repetitive nature of the image
* RLC tries to identify the length of the pixel values & encodes the image in the form of a run
* Each row of the image is written as a sequence.
* Run length is represented as a run of black & white pixels. This is known as Run length coding
* It is very effective way of compressing an image
* If required further compression can be done using Variable length coding to code the Runlengths themselves
* It is a lossless data compression.

Example Given a Sample binary Image. Apply Run length Coding

Image

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

5×5.

Soln - Scanning in done horizontally Scanning starts from top left goes to Right.

Run length vectors (0,5)
(0,3) (1,2)
(1,5)
(1,5)
(1,5) ← Max length

* No of vectors = 6
* Max length is = 5. ∴ 3 bits in binary
* No of bits/pixel = 1 (because the pixel are 0/1) → intensity bit representation
* Total no of pixels = 6×(3+1) = 24.
* No of pixels for the Original image = 5×5 = 25 pixel.
* Compression Ratio = $\frac{\text{no of bits Original}}{\text{no of bits after reduction}} = \frac{25}{24} = 1.042:1$

(2) Vertical RLC → Scanning is done in vertical lines.

Runlength pairs: (0,2) (1,3)
(0,2) (1,3)
(0,2) (1,3)
(0,1) (1,4)
(0,1) (1,4), max length.

∴ no of vectors = 10
∴ Max length = 4. ∴ no bits Required = 3bits
∴ 1 bit per pixel.
∴ Total no of pixels = 10 * (3+1) = 40

Original image 5×5 = 25
Compression ratio = 25/10 = 0.0625 : 1

It has been observed that the Compression ratio changes with Scan lines

Theoritically, if
The estimate of the entropy of the black run is H_0
The estimate of the entropy of the white run is H_1
The estimate of the average value of the black run is L_0
The estimate of the average value of the white run is L_1

The approximate run-length entropy of the image can be given by

$$H_{runlength} = \frac{H_0 + H_1}{L_0 + L_1}$$

Eg2 Use runlength encoding for Compressing the string

00000 11111 00 1 0000 101

Sol: 00000 (5) 11111 (5) 00 (2) 1 (1) 0000 (4) 10 (1,1) 1 (1)

vectors (0,5) (1,5) (0,2) (1,1) (0,4) (1,1) (0,1) (1,1)

no of vectors = 8 ∴ 3bits are Required. $\log_2 8 = \log_2 2^3 = 3\log_2 2 = 3$

⇒ (0,101) (1,101), (0,010) (1,001) (0,100) (1,001) (0,001)
(1,001)

Final code is

0.101 1101 0010 1001 0100 1001 0001 [00] = 32bits

Eg3 W WW WWWW B BBB WW B WWWWW BBBBBBB W

7W 4B 2W 1B 5W 7W 1W

Eg4 Singal bit stream. = 49bits

1 1111 000 00000 11111111 00 1111111 00000 1111111

(5,1) (8,0) (9,1) (2,0) (6,1) (5,0) (9,1)

(1,5) (0,8) (1,9) (0,2) (1,6) (0,5) (1,9)

Binary Representation

1, 0101, 01000, 11001, 00011, 10110, 00101, 11001 = 35bits

Symbol based coding

In Symbol or token based coding an image is represented as a collection of frequently occuring sub-images called Symbols.

Each Symbol is stored in a Symbol dictionary

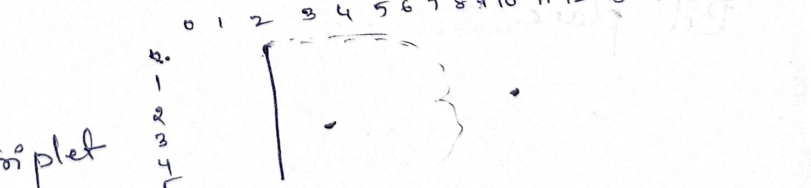
Image is coded as a set of triplet

$\{(x_1, y_1, t_1)(x_2, y_2, t_2) \ldots\}$

Represents location, Symbol-token

Eg Banana

Token	Symbol	Triplet
0	b	(0,2,0)
1	a	(3,10,1)
2	n	(3,18,2)
		(3,26,1)
		(3,34,2)
		(3,42,1)



Refer the figure (a) in the next page

Principles

→ Encodes an image to a set of triplets (r, y, t).

→ Symbol dictionary set of Ts.

Encoding Method.

→ Recognize Required sub-image & store them in a dictionary.

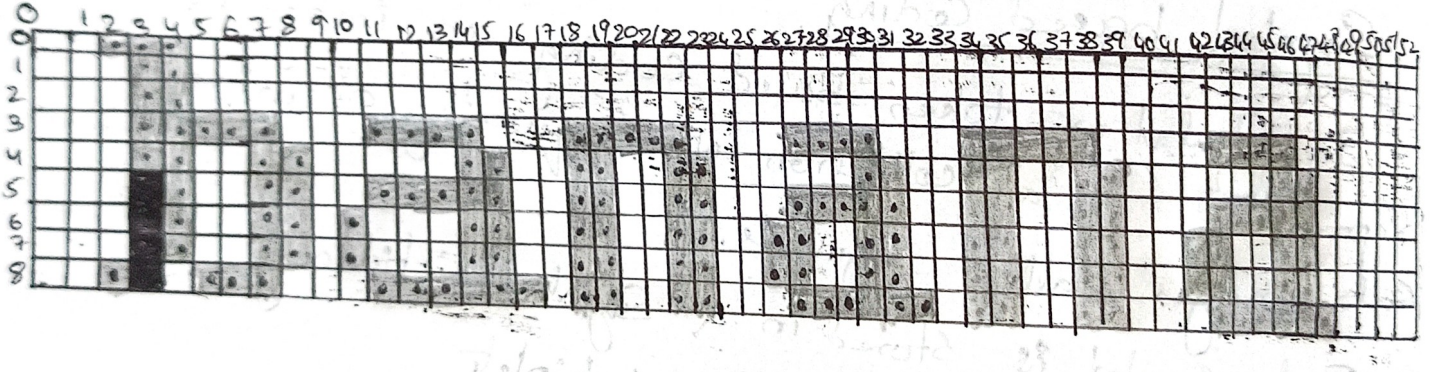
→ Locate each occurance of each sub-image & store its co-ordinates.

Decoding Method.

→ Create a empty range

→ For each triplet (r, y, t) place the topleft pixel of Sub-image T at (r, y).

fig9



Bit plane Coding

* The idea of Run-length coding can be extended to multilevel images.
* Bit plane coding technique 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 $a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$

Procedure:

1. The Zeroth-order bit plane is generated by collecting the a_0 bits of each pixel.
2. The first order bit plane is generated by collecting all the first bits of each pixel.
3. Similarly m-1 order bit plane is generated by collecting all the a_{m-1} bits of each pixel.

Eg In a gray scale image is given apply Bitplane coding

$$A = \begin{matrix} 2 & 6 & 6 \\ 6 & 7 & 7 \\ 1 & 2 & 4 \end{matrix}$$

3x3

Soln

$$A = \begin{matrix} 010 & 110 & 110 \\ 110 & 111 & 111 \\ 001 & 010 & 100 \end{matrix}$$

Binary equivalent of the image

1. As there are 3 bits in each pixels it can be divided into 3 planes.
   1. MSB bit plane
   2. Middle bit
   3. LSB bit

$$A_{MSB} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad A_{middle} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad A_{LSB} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

* Individual planes can be compressed using RLE.

Disadvantage

The neighborhoods in the spatial domain like 3 & 4 having binary coded 011 & 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 by only one bit

eg:

180	1	50	33	201	28
120	2	160	33	67	144
224	161	180	11	133	144

Step1 obtain the binary equivalent of the values in image.

10110100	00000001	00001000	00100001	11001000	00011100
01111000	00000010	10100000	00100001	01000011	10010000
11100000	10100001	10110100	00001011	10000101	10010100

Step2 Extract the MSB.

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

Step3 Rearrange the MSB values. such that each row contains 8 bits in a row

Step 4 - convert to decimal

1	0	0	0	1	0	0	1	→ 137
1	0	0	1	1	1	1	0	→ 158
1	1	0	0	0	0	0	0	→ 192

Zero padding

Step 4

Decompression

step 1

137 1 0 0 0 1 0 0 1

158 1 0 0 1 1 1 1 0

192 1 1 0 0 0 0 0 0

step 2 - Remove extra Zero

step 3 - Reshape the matrix to size of the original image.

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

1 bit / pixel.

step 4 Preallocate a matrix same size as original.

1000 0000 0000 0000 1000 0000
000 0000 0000 0000 0000 0000

8 bit / pixel

step 5. Display the final data.

128 0 0 0 128 0

0 128 128 0 0 128

128 128 128 0 128 128

LZW Coding Lempel - Ziv - Welch.

* It focusses on spatial redundancy
* LZW Compression is a method to reduce the size of Tag Image file Format [TIFF] or Graphics interchange format (GIF) files.
* It is table based look-up algorithm to remove duplicate data & compresses an original file into a smaller file.

How LZW Compression works.

The LZW Compression algorithm reads a sequence of symbols, groups those symbols into strings and then converts each string into codes.

In the begining of a coding process a dictionary containing the source symbols to be coded is constructed.

For 8bit monochrome images

→ The first 256 words of the dictionary are assigned to intensities 0, 1, 2 ... 255.

→ As the encoder sequentially examines image pixels, Gray level sequences that are not in the dictionary are placed in algorithmically determined locations.

Eg If the 1st 2 pixels of the image are white, the sequence 255-255 might be assigned to location 256, 256 is the address following the locations Reserved for 0-255 gray levels.

Next time the consecutive white pixels are encountered Code word 256 is used to Represent them.

→ Dictionary size is an important parameter.

if it is too small → detection of matching gray level sequences will be less likely

if it is too big → codewords will adversely affect Compression Performance.

Eg. Consider an image

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

Consider a 9bit (512) word dictionary

Dictionary locations	entry:
0	0
1	1
.	.
.	.
255	255
256	-
.	
.	
511	

① To code a 16 pixels. 512 word dictionary with the following starting content is assumed.

* location 256-511 are initially unused.

* The is encoded by processing its pixels in a left to right, top-to-bottom manner.

Pixel Being Encoded	Encoded	Dictionary	Dictionary

(2) Use LZW coding algorithm for aaaaaaaaaaaa to encode the 7 bit ascii string

7 bit data
"a a a a a a a a a a a a"

7 bit Means
0 - 127 : 128

Soln

65.
a a a a a a a a a a a a

Currently Recognized Sequence	Pixel being processed	Encoded o/p locations	Dictionary locations	Dictionary entry
	a	—	—	—
—	a	65	128	aa
a	a	—	—	—
aa	a	128	129	aaa
a	a	—	—	—
aa	a	—	—	—
aaa	a	129	130	aaaa
a	a	—	—	—
aa	a	—	—	—
aaa	a	—	—	—
aaaa	a	130	131	aaaaa
a	a	65		

Total bits before Compression = 11 X 7 = 77 bits (∵ 65, 65, 65... 11 times)

After Compression = (65, 128, 129, 130, 65)
8 bit = 8 × 5
= 40 bits.

Arithmetic Coding

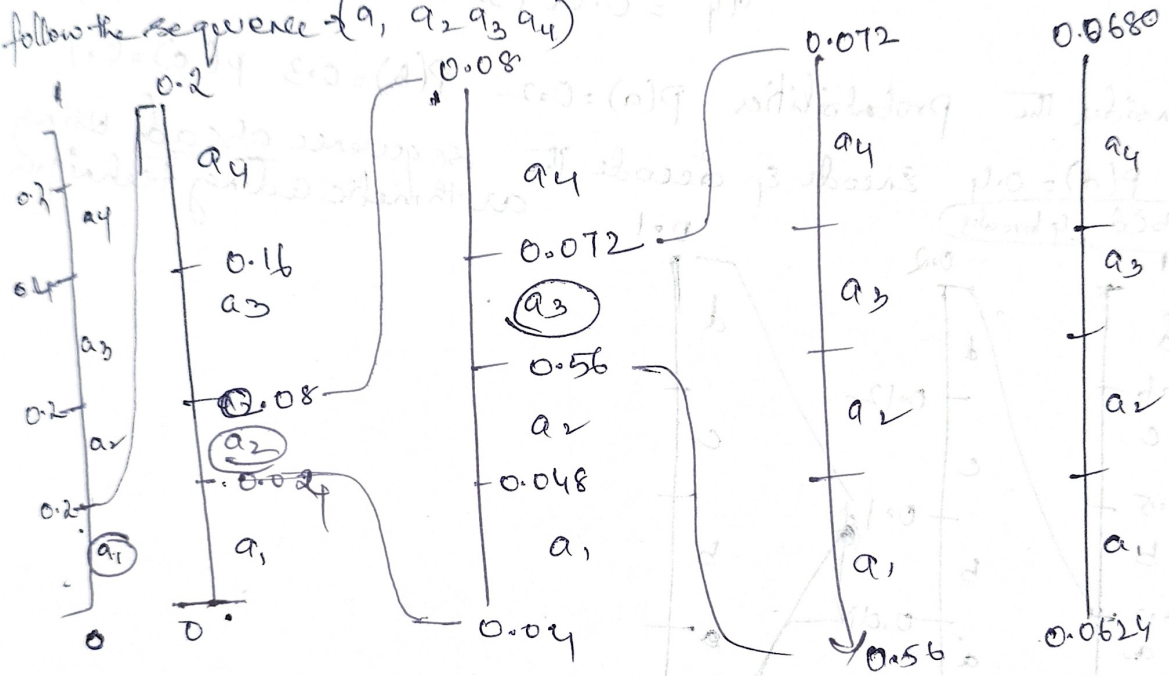
is based on the concept of interval subdividing.

Decode the five symbol sequence or message, $a_1 a_2 a_3 a_4$ from a four symbol source is coded.

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]

Sub 0 to 1

follow the sequence (a_1, a_2 a_3 a_4)



① $R = H - L$

H - High Value = 0.2

= 0

L -

② $R = H - L$

$= 0.08 - 0.04$

$= 0.04$

$L + R \times P$

$a_1 = 0 + 0.2 \times 0.2$

$= 0.04$

$a_2 = 0.04 + 0.2 \times 0.2 = 0.08$

$a_3 = 0.08 + 0.2 \times 0.4 = 0.16$

$a_4 = 0.16 + 0.2 \times 0.2 = 0.2$

$L + R \times P$

$a_1 = 0.04 + 0.04 \times 0.2 = 0.048$

$a_2 = 0.048 + 0.04 \times 0.2 = 0.056$

$a_3 = 0.56 + 0.04 \times 0.4 = 0.072$

$a_4 = 0.072 + 0.04 \times 0.2 = 0.08$

3. $R = H - L$ =

$0.072 - 0.056$

$= 0.016$

L + R × P

$a_1 = 0.056 + 0.016 \times$ ___ $= 0.592$

$a_2 = 0.0592 + 0.016 \times 0.2 = 0.0624$

$a_3 = 0.0624 + 0.016 \times 0.4 = 0.0688$ = 0.0688

$a_4 = 0.0688 + 0.016 \times 0.2 = 0.072$

4. $R = H - L$

$= 0.0688 - 0.0624$

$= 0.0064$

L + R × P

$a_1 = 0.0624 + 0.0066 \times 0.2 = 0.06268$

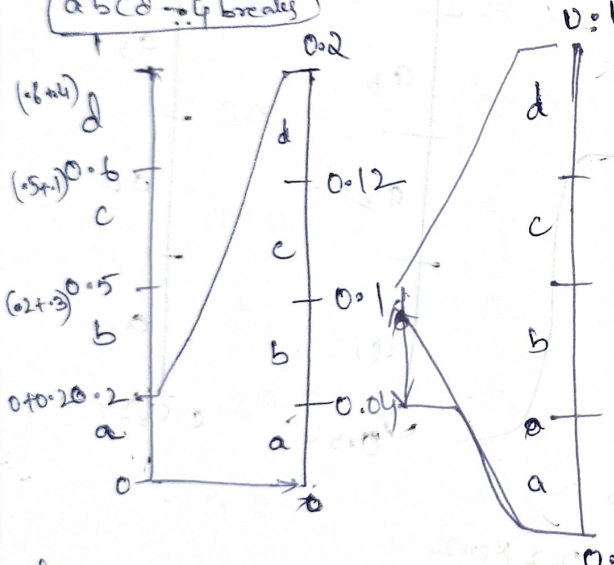
$a_2 = 0.06288 + 0.0066 \times 0.2 = 0.06368$

$a_3 = 0.06496 + 0.0064 \times 0.4 = 0.06732$

$a_4 = 0.06732 + 0.0064 \times 0.2 = 0.0688$

(2) Consider the probabilities $P(a) = 0.2$ $P(b) = 0.3$ $P(c) = 0.1$ $P(d) = 0.4$ Encode & decode the sequence abad using arithmetic coding technique

abcd → 4 breaks



difference

$D = H - L$

$= 0.2 - 0$

$= 0.2$

Range of a Symbol = L : L + D (Probability of Symbol)

Range of a. = 0 : 0 + 0.2 (0.2) = 0 : 0.04

b = 0.04 : 0.04 + 0.2 (0.3) = 0.04 : 0.1

c = 0.1 : 0.1 + 0.2 (0.1) = 0.12

d = 0.12 : 0.12 + 0.2 (0.4) = 0.12 : 0.2

Arithmetic coding :-

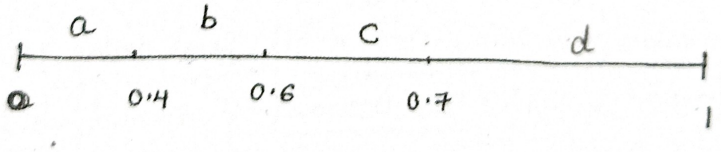
In arithmetic coding, the interval from zero to one is divided according to the probabilities of the occurrences of the intensities. Arithmetic coding does not generate individual codes for each character but performs arithmetic operations on a block of data, based on the probabilities of the next character.

Using arithmetic coding, it is possible to encoded characters with a fractional number of bits, thus approaching the theoretical optimum.

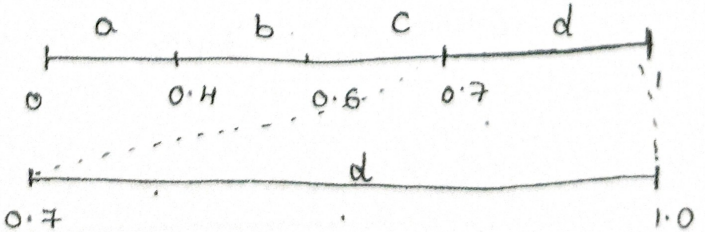
Graphical Method of Arithmatic coding :-

Ex :- A source emits four symbols {a, b, c, d} with the probabilities 0.4, 0.2, 0.1 and 0.3 respectively. construct arithmatic coding to encode and decode the 'coord dad'.

Sol :- Step 1 :- To code coord dad', initially assume range of intervals between 0 & 1.



Step 2 :- (a) first symbol to be transmitted is 'd'. Hence the new range is from 0.7 to 1.0.



formula to compute low & high range of each symbol within the interval is given by

$$low = low + range \times (low\_range)$$

$$high = low + range \times (high\_range)$$

low_range refers to symbol of low range.

(i) To compute interval for the symbol a in the interval 0.7 to 1.0

Range = 1.0 − 0.7 = 0.3

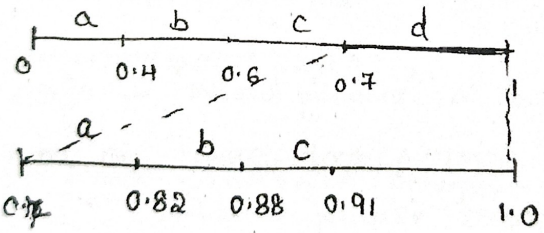
low = 0.7 + 0.3 × 0 = 0.7 , high = 0.7 + 0.3 × (0.4) = 0.82

∴ a has lower interval 0.7 & higher interval 0.82.

(ii) To compute interval for b in the interval of 0.7 to 1.0

Range = 0.3 , low = 0.7 + 0.3 × (0.4) = 0.82

high = 0.7 + 0.3 × (0.6) = 0.88

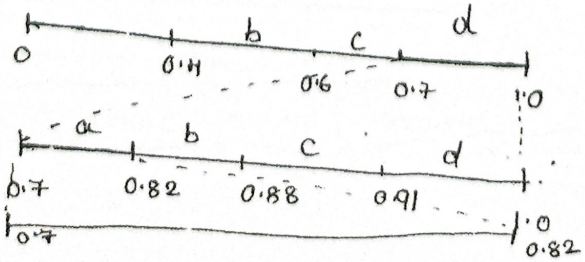


(iii) Ly for c in interval 0.7 to 1.0

range = 0.3, low = 0.7 + 0.3(0.6) = 0.88

high = 0.7 + 0.3 (0.7) = 0.91

c → (0.88, 0.91)



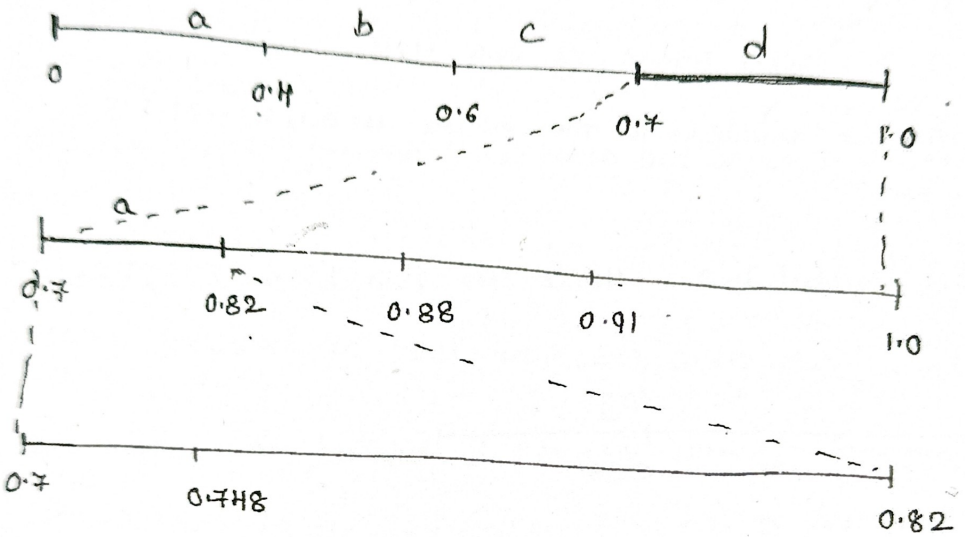
Step 3a : To find the interval for the symbols in the range 0.7 to 0.82.

(i) To find the interval for the symbol 'a' we find the lower interval which is given by

$$low = low + range \times (low - range)$$

$$Range = 0.82 - 0.70 = 0.12$$

$$low = 0.7 + 0.12 \times 0.0 = 0.7$$



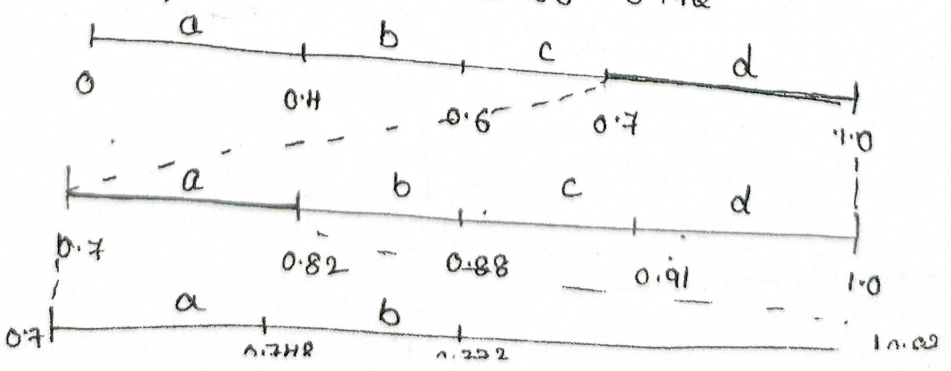
upper interval is given by $high = 0.7 + 0.12 \times 0.4 = 0.748$

(ii) To find the lower and upper interval for the symbol 'b', we find the lower interval which is given by

$$low = 0.7 + 0.12 \times 0.4 = 0.748$$

The upper interval is given by

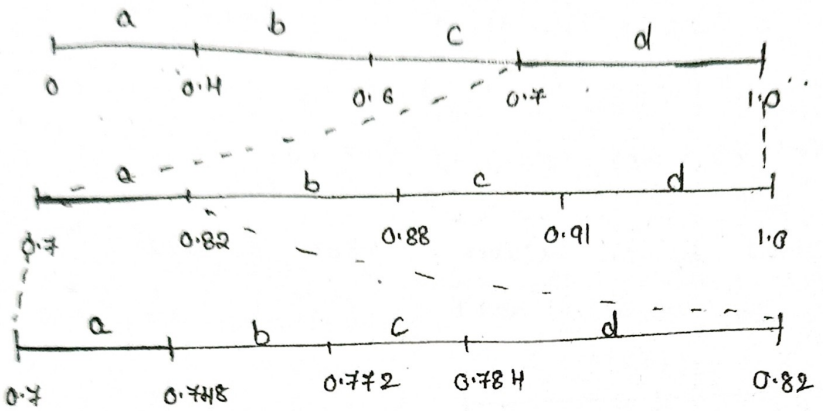
$$high = 0.748 \cdot 12 \times 0.6 = 0.772$$



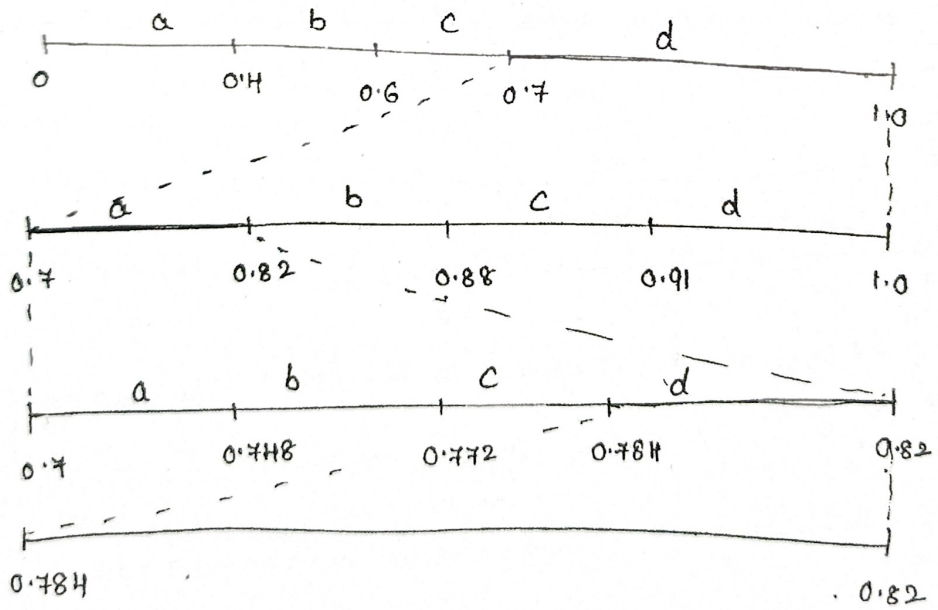
(iii) To find the lower and upper interval for the symbol c, we find the lower limit which is given by

$$low = 0.7 + 0.12 \times 0.6 = 0.772$$

The upper limit is given by $high = 0.7 + 0.12 \times 0.7 = 0.784$



Step 4 : To transmit the symbol 'd' in the 'word dad'. In order to transmit the next symbol d the intervals 0.784 & 0.82 are taken as the new lower and upper intervals which is illustrated below.



Step 5 : To compute the tag value. Tag is average of lower and upper intervals.

$$tag = \frac{0.784 + 0.82}{2} = 0.802$$

decoding procedure :-

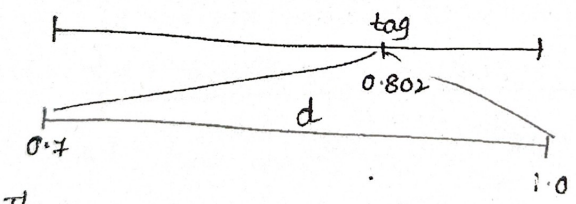
The tag value and symbol probabilities will be sent to the receiver. After receiving the tag value, the decoder decodes the encoded data.

Symbol	a	b	c	d
Probability	0.4	0.2	0.1	0.3
Sub-Range	(0 - 0.4)	(0.4 - 0.6)	(0.6 - 0.7)	(0.7 - 1.0)

Step 1: The tag received by the receiver is 0.802 The initial interval is assumed to be between 0 and 1.

The tag value is compared with the symbol sub-range. we find that 0.802 lies between 0.7 and 1.0, hence the corresponding decoded symbol is d.

Step 2 :- The decoded symbol in Step 1 is d and hence the new interval is fixed as 0.7 to 1.0 which is illustrated below.



Step 3 :- The new tag value is computed in this step. The new tag value is given by

$$t^* = \frac{\text{tag} - \text{low}}{\text{range}}$$

tag = 0.802, low = 0.7, Range = high - low = 1.0 - 0.7 = 0.3

$$t^* = \frac{0.802 - 0.7}{0.3} = 0.34$$

∴ The tag value lies in 0 to 0.4 interval ∴ decoded value is 'a'

Image Enhancement in Frequency Domain

Basics of filtering in frequency Domain :-

Additional Characteristics of the frequency Domain :-

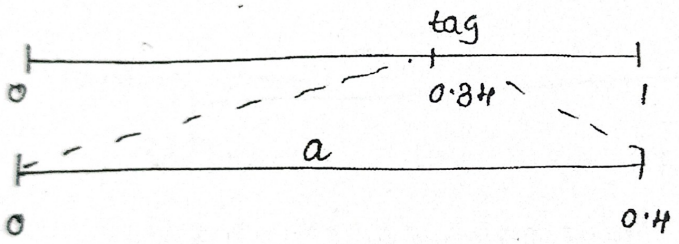
$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi(ux/M + vy/N)}$$

where $f(x,y)$ is a digital image of size $M \times N$.

u & v are discrete variables in the ranges $u = 0,1,2 \cdots M-1$ &

$v = 0,1,2 \cdots N-1$

step 4:- Decoded symbol is a. Hence the new interval is change to 0 to 0.4.

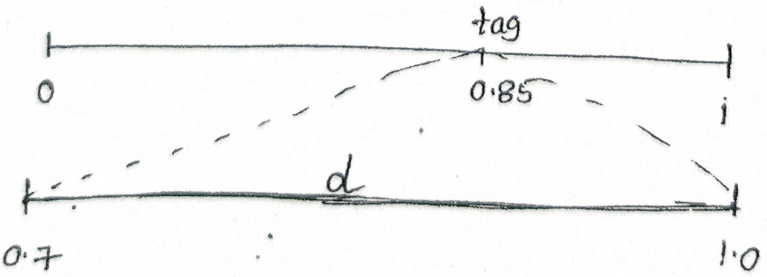


step 5:- The new tag value for the interval given is

$$t^* = \frac{0.34 - 0}{0.4} = 0.85$$

This tag value lies between 0.7 & 1.0 & hence the decoded symbol is d.

It is clear the decoded word is 'dad'.



Golomb coding → used for lossless data compression.

1. This is used to design encoder integer value
2. This coding scheme having two schemes → Unary codes (q) & Rice codes (r)
3. Golomb codes is based on the assumption that larger the integer lesser is the probability to occur.
4. Unary Codes. Can be represented as $q = \left\lfloor \frac{n}{m} \right\rfloor$ $r = n - qm$

Unary code representation 'N' integer
- No of 1's followed by Zero
- No of 0's followed by One's

for example → 111110
→ 000001

Steps of algorithm to find the Golomb code for any integer

Step1 → Find the unary code of $q = \lfloor n/m \rfloor$
floor value of n/m
n – integer
m – divisor

Step2 → let $k = \lceil \log_2 m \rceil$ → Ceiling of
$c = 2^k - m$ & $r = n \bmod m$

$$r' = \begin{cases} r \text{ truncated to } k-1 \text{ bits} & 0 \le r < c \\ r+c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

Step3. → Concatinate the result of step1 & step2.

Design golomb code for 9 with divisor 4.

Sol: n = 9 m = 4.

$G_4(9) = ?$

1. $q = \lfloor 9/4 \rfloor = 2$ → Unary code = 110 — (a)

2. $k = \lceil \log_2 4 \rceil = 2$.

$c = 2^2 - 4$

$c = 0$

$r = n \mod m$
$= 9 \mod 4$
$r = 1$.

$r = 1$ $c = 0$ $r > 0$

$r' = r + c = 1$ → Represent ~~1~~ 2 bit. ⇒ 01 — (b)

3. 11001 $G_4(9) = 11001$.

9
−4
5
−4
1

4)9(2
8
1

Block transform coding

→ is a lossy compression algorithm.

→ Data loss is acceptable if it is tolerable

→ Loss of information occurs during compression. & This loss is known as distortion.

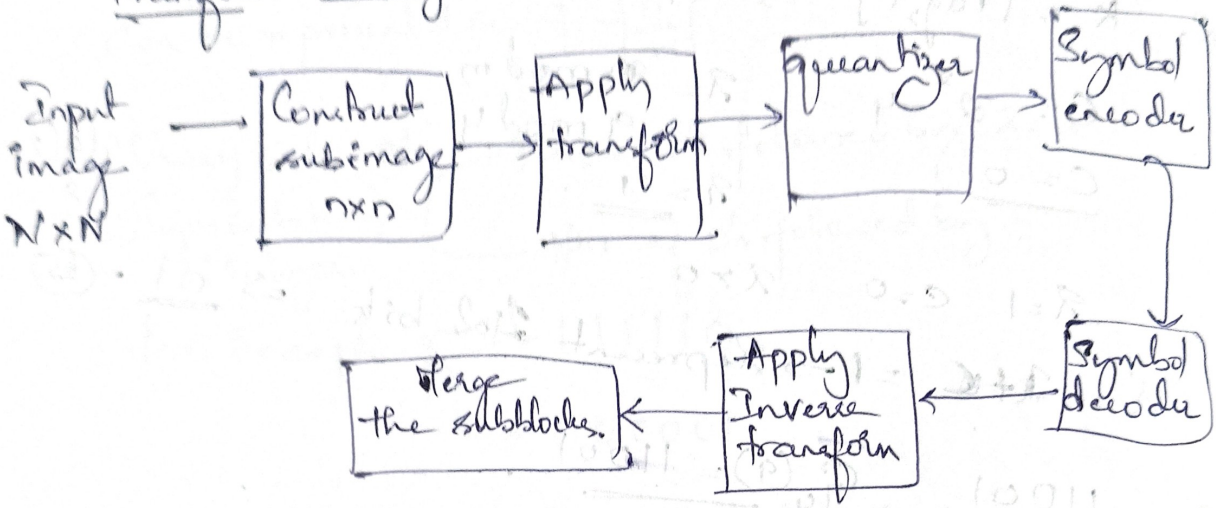
→ The compression ratio is very large compared to lossless compression algorithm.

→ Popular lossy compression algorithm are
1. lossy predictive coding
2. Vector quantization
3. Block transform coding

Block transform Coding:

Transform Coding Compresses the imag data by representing the original signal with a small no of transform Co-efficients.

Transform coding Model.



* Sub-image selection → At this stage image is divided into set of subimages.
* N×N imag is decomposed into set of Subimages of the size n×n.
* The value of n is a power of 2
* This process is done to ensure that Co-relation among the pixel is minimum.
* This step is necessary to reduce Computational Complexity.
* Transform selection →
* The idea behind transform coding is to use Mathematical transforms for data Compression.
* Transformations like DCT, DFT & wavelet transforms are used
* Choice of transforms. - depends on the resources & amount of error associated with reconstruction process.
* DCT → offers better information packing capacity.
* KL transforms are also effective but they are data

* DCT is preferred because it is faster & can pack more information

Bit allocation.

* Transform coding is the process of truncating, quantizing & coding the co-efficients of the transformed subimage.
* Bits must be assigned so that compressed image have minimum distortion.

Steps involved in bit allocation are

1. Assign pre defined bits to all clusters of data in the image
2. Reduce the no of bits by one & calculate the distortion
3. Identify the data which is associated with minimum distortion & reduce one bit from its quota.
4. Find the distortion rate again
5. Compare the ~~target~~ & if required Repeat steps 1-4 to get the optimal rate

Zonal coding -

This process involves multiplying each transform co-efficient by the corresponding element in zonal mask, which has 1 in the locations of maximum variance & 0 in the other places.

* A zonal mask is designed as a part of this process
* These co-efficients are retained because they convey more imag information.
* The locations are identified based on the image Models used for source symbols encoding
* The retained co-efficients are quantized and coded
* The Number of bits allocated may be fixed or may vary based on some optimal quantizer.

Threshold Mask.

Threshold coding mode works based on the fact that transform co-efficients having the maximum magnitude make the most contribution to the image.

The threshold may be one of the following

1. Single global threshold
2. Adaptive threshold for each subimage
3. Variable threshold as a function of the location for each co-efficient in the subimage.

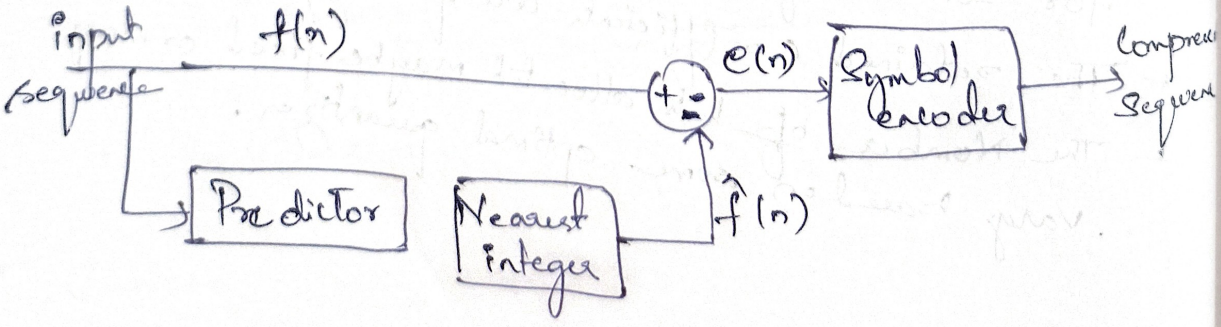
Predictive Coding

Predictive coding achieves good compression without significant computational overhead and can be either lossless or lossy.

It is based on eliminating the redundancies of closely spaced pixels - in space/time - by extracting & coding only the new information in each pixel.

The new information of a pixel is defined as the difference b/n the actual & predicted values of the pixel.

Model of the predictive coding model. (lossless)



Prediction error = $e(n) = f(n) - \hat{f}(n)$

* $f(n)$ → input sequence
* $\hat{f}(n)$ → is generated by predictor by taking the values of $f(n)$ or its previous value & the calculated new value is rounded off &
* $e(n)$ is calculated by $f(n) - \hat{f}(n)$
* This error function is encoded to generate compressed sequence.

* **Predictor** → to compare present & previous value.
* Predictor is formed as a linear combination of m previous samples, that is

$$\hat{f}(n) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

m → order of the linear Predictor
α_i – Predictor – co-efficients while i = 1, 2 ...

* In case of images

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

* $f(n-i)$ – is the previous value of $f(n)$ which as lapped i instance back.
* In case of images. pixels in a same row is considered

Eg:- $f(n) = \{20, 20, 21, 21, 22, 22, 23, 23, 23\}$.

$$\hat{f}(n) = \text{round}\left[\sum_{j=1}^{m} \alpha_i f(n-i)\right] \quad m=1 \quad \alpha_i = 1$$

$$= \text{round}\left[\alpha_1 f(n-1)\right]$$

$$= \text{round}\left(f(n-1)\right)$$

$$e(n) = f(n) - \hat{f}(n)$$

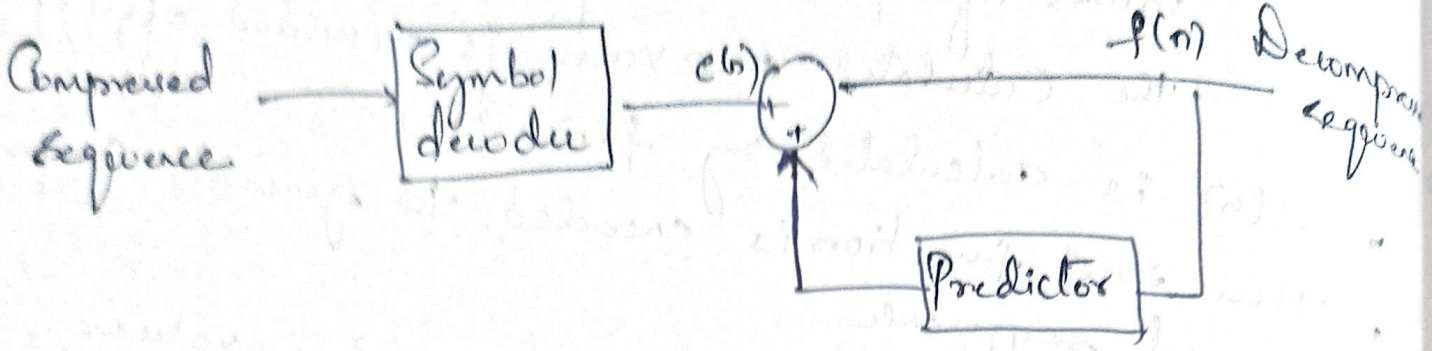
$$e(n) = \{20, 0, 1, 0, 1, 0, 1, 0, 0\}$$

The larger range values are changed to smaller range

* Entropy indicate how many different values are there in the image.

Higher entropy indicates higher value of data is need to store this image.

lossless predictive coding model Decoder



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

* The Compressed sequence is given to symbol decoded (like huffman decoder) to get the error value
* Error value will be using a o/p value $f(n)$ to get $\hat{f}(n)$ to get $f(n)$.

Eg $e(n) \in \{20, 0, 1, 0, 1, 0, 1, 0, 0\}$

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

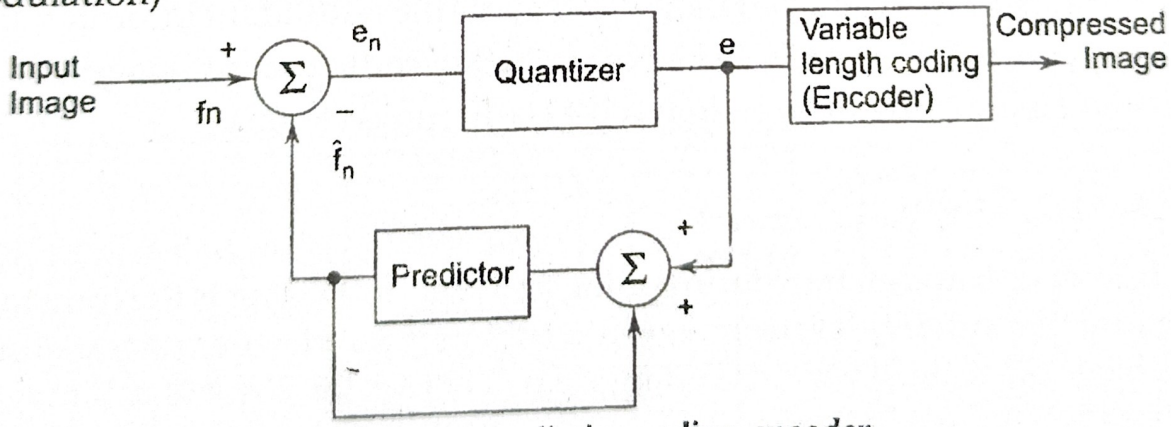
$f(n) = e(n) + f(n-1)$

$= 20, 20\ 21, 21, 22, 22, 23, 23, 23$

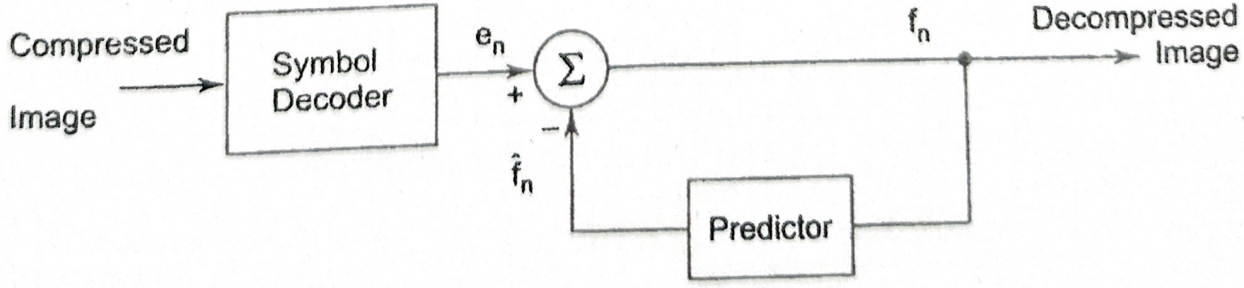
6.7.1 Lossy Predictive Coding

We have already discussed the lossless predictive coding. Here we have drawn the block diagram of lossy predictive coding. Here we have drawn the block diagram of lossy predictive coding system as shown in Fig. 6.8. The practical difference between lossless predictive coding and lossy predictive coding system is that we use quantizer in the case of lossy compression system.

The examples of such type of coding are – DPCM (Differential Pulse Code Modulation) and DM (Delta Modulation)



(a) Lossy predictive coding-encoder



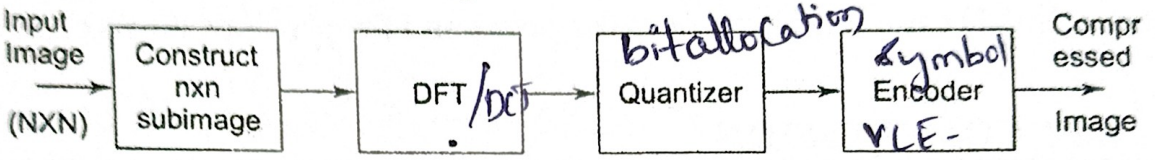
(b) Lossy predictive coding-decoder.

Fig. 6.8 *Lossy predictive coding.*

6.7.2 Transform Coding

Transform coding is a technique in which we apply compression on the transform of an image. In transform coding, a reversible, linear transform (such as fourier transform) is used to map the image into set of transform co-efficients. These transform coefficients are then quantized and coded.

Basic block diagram of transform coding has been drawn in figure 6.9.



(a) Transform Coding-Encoder

(b) Transform Coding-Decoder.

Fig. 6.9 *Transform Coding*

Let us discuss the basic steps of encoding using transform coding.

1. First of all an N × N input image is subdivided into subimages of size $n \times n$. During the subdivision of an image into subimages it should be cared that correlation (redundancy) between adjacent subimages is reduced to minimum level. Most popular subimage sizes are 8×8 and 16×16. Thus image has been converted into small pack of information that are easy to process.
2. Now by transformation (Fourier transform in case of digital signal is referred as discrete fourier transform, DFT) we generate transformation coefficient by the subimages.
3. By quantization process, generally we remove the redundant information.
4. Now apply variable length coding for reducing coding redundancy.

The good use of transform coding is JPEG (discussed latter).

Wavelets & Multiresolution Processing

1. Image Pyramids –

Wavelets in Ip → FT is used for Image processing since 1950
More recently wavelet transform is being used.

WT → has made the images even easier to Compresses & transmit & analize

FT → The basic functions are sinusoids

WT → Uses small waves (wavelets of varying frequency & limited duration.)

In 1987 - the wavelets are 1st shown as powerfull approach in signal processing & analysis. → known as Multiresolution theory.

It incorporates subband coding, quadrature Mirror filtering & Pyramid Imag processing.

Multiresolution theory. → is used for representation & analysis of signal at more than one resolution.

Background

→ Images are Connected regions of similar texture & gray levels are combined to form Object.

→ When the object is very small or low in Constrast, we examine in them at high resolution.

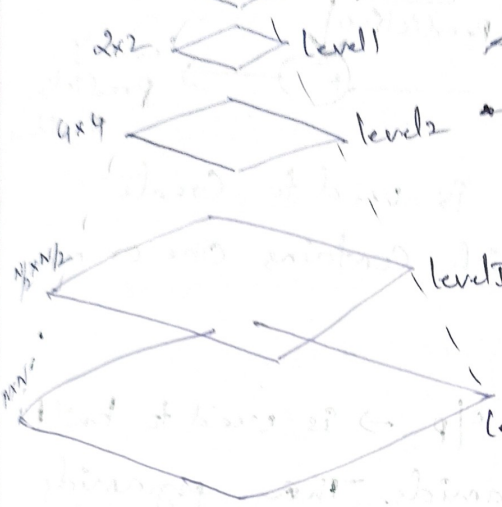
→ If small & large objects or low & high Constrast objects are present it is advantageous to study them at several Resolutions.

→ This is the fundamental motivation for multiresolution processing

Image pyramids is a tool for representing image at more than one resolution.

The image pyramid is a collection of decreasing resolution representation of images that are arranged in a shape of a pyramid.

It was originally used for image compression application



The fig shows the pyramid image structure

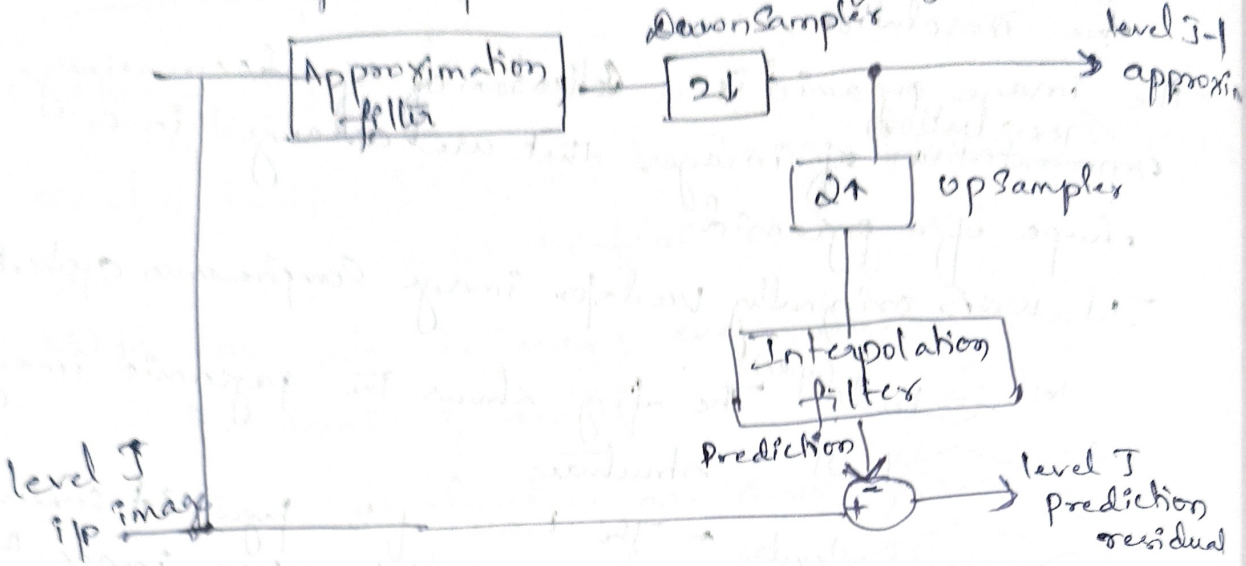
- The base of the pyramid consists of high resolution of an image that are being processed & the apex contains a low resolution approximation of an image.

In this image pyramid as we move up both size & resolution of an image decrease

- The base level J is have having the size $2^J \times 2^J$ or $N \times N$. $\underline{J = \log_2 N}$
- The intermediate levels are of the size $j \Rightarrow 2^j \times 2^j$.
  where $j \longrightarrow 0$ to J $(0 \le j \le J)$
- The fully populated pyramid are compared of $J+1$ resolution level from $2^J \times 2^J$ to $2^0 \times 2^0$
- Most pyramid are truncated to $P+1$ levels $1 \le P \le J$
  $j = J-P \cdots J-2, J-1, J$ &
- Total no of elements in $P+1$ level, $P > 0$

$$N^2 \left\{ 1 + \frac{1}{4^1} + \frac{1}{4^2} \cdots \frac{1}{4^P} \right\} \le \frac{4}{3} N^2$$

Block diagram for Construction of pyramids



1. Level J-1 approximation o/p is used to Create approximation pyramid which contains one or more approximations of original

→ level J prediction residual o/p → is used to build Prediction residual pyramids. These pyramids Consists of low resolution approximations of original imag & information for the Construction of higher Resolution approximations at other level.

→ The information at level j is the difference of b/n level J i/p image and prediction of level J-1

→ This difference can be coded & stored more efficiently.

→ The approximation & prediction of pyramids can be Computed in iterative fashion.

→ During the first iteration J=1 original image is applied as level J i/p image.
This will produce level J-1 approximation & level J prediction residual result

→ for the next approximations J-1, J-2 ... J-(P+1) approximations is used as i/p

→ each pass is composed of 3 steps.

1. Compute a reduced-resolution approximation of the image (down samp). down sampling is done on approximation filter - which can be Neighbourhood averaging, No filter);

2. UpSample the o/p of previous step by the factor of 2. & then filter the result - this creates an prediction image with the

3. Compute the difference b/n the prediction of step 2 & the i/p of step 1.

This produces the level J prediction residual o/p that can be later used to reconstruct the original image.

* The of Digital filter approximation problems consists of selecting Co-efficients of the rational transfer function H(z). in order to achieve some desired result when the filter is applied to a signal.

* Down sampling - The process of reducing sampling rate by an integer factor. By maintaining essential info.

Subband Coding.

Subband code is used for multiresolution analysis.

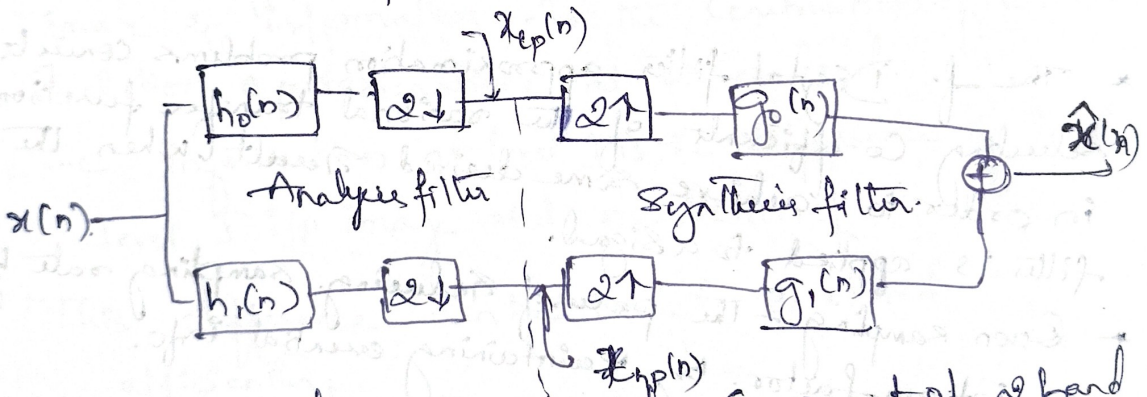
An image is decomposed into a set of band limited components known as subbands.

These subbands can be reassembled to reconstruct the original image without any error.

Subband coding is used for speech & Image Compression. where each subband is generated by bandpass filtering the i/p image.

Since the bandwidth of subband is smaller than the original image. it can be easily downsampled without any loss of information.

The reconstruction of original image can be done by downsampling, filtering & summing the subbands.



This fig shows the principle component of 2 band subband coding of the & decoding system.

The i/p $x(n)$ is the 1-D bandlimited DT signal where

$n = 0, 1, 2 \ldots\ldots$

The o/p $\hat{x}(n)$ is formed by the decomposition of $x(n)$ into $g_0(n)$ & $g_1(n)$ using Analysis filter $[h_0(n), h_1(n)]$ & synthesis filter $[g_0(n), g_1(n)]$. the

Here filters $h_0(n)$ & $h_1(n)$ are half band digital filters where idealized transform characteristics is shown below.

It shows the spectrum splitting property of the filter where $|H_0(\omega)|$ - i/p, LPF where o/p is approximation of $x(n)$

$|H_1(\omega)|$ is HPF where o/p is high frequency

All the filtering is done in time domain by converting each filter i/p into impulse response $\delta(n)$.

To Reconstruct the i/p select $h_0(n)$ & $h_1(n)$, $g_0(n)$ & $g_1(n)$

To analyze subband coding z-transform is used as it is a ideal tool for studying DT sampled datas/g

-> Z-transform of $x(n) = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$ $n = 0, 1, 2 \cdots$ —①

In time domain, downsampling by 2 is

$$x_{down}(n) = x(2n) = X(z) = \frac{1}{2}\left[X(z^{1/2}) + X(-z)^{1/2}) \right] \text{—②}$$

upsampling by 2 is

$$X^{up}(n) \begin{cases} X(n/2) ; n = 0, 2, 4, \cdots \\ 0 \text{ otherwise} \end{cases}$$

$$X^{up}(z) = X(z^2) \text{ —③}$$

Eqn ② & ③ Combined it will yield $\hat{x}(n)$

$$\hat{X}(z) = \frac{1}{2}\left[X(z) + X(-z) \right] \text{—④}$$

The System o/p

$$\hat{X}(z) = \frac{1}{2} G_0(z) \left[H_0(z) X(z) + H_0(-z) X(-z) \right]$$

$$+ \frac{1}{2} G_1(z) \left[H_1(z) X(z) + H_0(z) X(-z) \right] \text{—⑤}$$

Rearranging eqn (5)

$$\hat{X}(z) = \frac{1}{2}\left[H_0(z)G_0(z) + H_1(z)G_1(z)\right]X(z) + \frac{1}{2}\left[H_0(-z)G_0(z) + H_1(-z)G_1(z)\right]X(-z) \quad -(6)$$

for error free Reconstruction of i/p Condn.

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0 \quad -(7)$$

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2 \quad -(8)$$

Eqn (7) & (8) can be represented in matrix

$$\begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix} H_m(z) = \begin{bmatrix} 2, 0 \end{bmatrix} \quad -(9)$$

$H_m(z)$ → analysis Modulation matrix.

$$H_m(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \quad -(10)$$

Assuming the non Singularity. $\det(A) \neq 0$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \frac{2}{\det(H_m(z))} \begin{bmatrix} H_1(-z) \\ H_0(-z) \end{bmatrix} \quad -(11)$$

The analysis & synthesis filters are Cross Modulated.

FIR filters, det H_m is pure delay. $\det(H_m(z)) = \alpha z^{-(2k+1)}$ -(12)

Exact form of Cross modulation is fune of α.

if $\alpha = 2$ then
$$\left.\begin{aligned} g_0(n) &= (-1)^n h_1(n) \\ g_1(n) &= (-1)^n h_0(n) \end{aligned}\right\} \quad -(13)$$

if $\alpha = -2$ then
$$\left.\begin{aligned} g_0(n) &= (-1)^{n+1} h_1(n) \\ g_1(n) &= (-1)^n h_0(n) \end{aligned}\right\} \quad -(14)$$

Hence, FIR filters are Cross modulated copies of the analysis filters with one being sign Reversed

eqn can also be used to show biorthogonality of the analysis & synthesis filter.

To do this, Let $P(z)$ be the product of LPP analysis & Synthesis filter transfer func.

$$P(z) = G_0(z)\,H_0(z) = \frac{2}{\det H_m(z)}\,H_0(z)\,H_1(z) \quad (15)$$

$$P(-z) = G_1(z)\,H_1(z) = \frac{-2}{\det H_m(z)}\,H_0(z)\,H_1(z) \quad (16)$$

Hence we can write

$$G_0(z)\,H_0(z) + G_0(-z)\,H_0(-z) = 2 \quad (17)$$

Taking Inverse z-transform.

$$\sum_k g_0(k)\,h_0(n-k) + (-1)^n \sum_k g_0(k)\,h_0(n-k) = 2\delta(n) \quad (18)$$

Since odd indexed terms cancel,

$$\sum_k g_0(k)\,h_0(2n-k) = \langle g_0(k), h_0(2n-k)\rangle = \delta(n) \quad (19)$$

It can be shown that.

$$\left.\begin{aligned} \langle g_1(k), h_1(2n-k)\rangle &= \delta(n) \\ \langle g_0(k), h_1(2n-k)\rangle &= 0 \\ \langle g_1(k), h_0(2n-k)\rangle &= 0 \end{aligned}\right\} \quad (20)$$

Generally

$$\langle h_i(2n-k), g_j(k)\rangle = \delta(i-j)\,\delta(n), \quad i,j = \{0,1\} \quad (21)$$

+ Filter banks satisfying these God Conditions ⇒ biorthogonal.

→ Table gives 3 general sols to eqn.

Filter	QMF	CQF	Orthogonal
$H_0(z)$	$H_0^2(z) - H_0^2(-z) = 2$	$H_0(z)H_0(z^{-1}) + H_0(z)H_0(-z^{-1})$	$G_0(z^{-1})$
$H_1(z)$	$H_0(-z)$	$z^{-1}H_0(-z^{-1})$	$G_1(z^{-1})$
$G_0(z)$	$H_0(z)$	$H_0(z^{-1})$	$G_0(z)\,G_0(z^{-1}) + G_0(-z)\,G_0(-z^{-1}) = 2$
$G_1(z)$	$-H_0(-z)$	$zH_0(-z)$	$-z^{-2k+1}G_0(-z^{-1})$

What is wavelet & multiresolution processing?

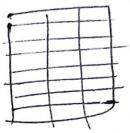
* wavelet transform is used to analyze the signal into different frequency components at different resolution scales.

This allows revealing image's spatial & frequency attributes simultaneously.

In addition, features that might go undetected at one resolution may be easy to spot at another.

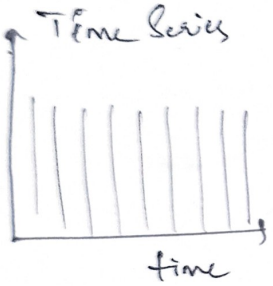
Fourier transform provides only frequency information but wavelet transform provides time-frequency info.

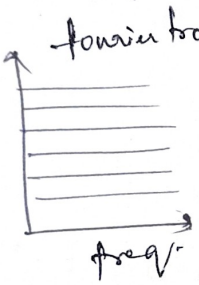
window.



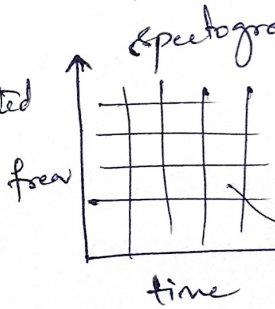
Wavelets.

FT → decompose into sum of sines & cosines.





so motivated vs ↑



← Exclusive resolution in time

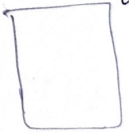
* Uncertainity in freq ∵ no freq info

↑ Resolution in freq

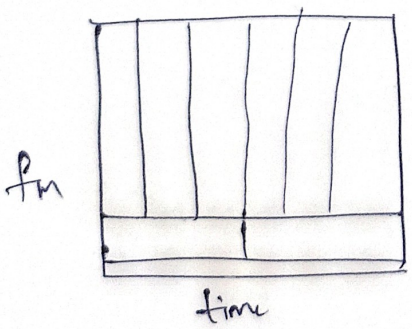
* no idea about time of a
* when were those freq started

↓ freq resolut
↓ time resolution

but when the particular freq is turned on and off



Wavelets → is often called as multiresolution processing



Multi scaled time freq decomposition

- lower frequency is changed more slowly in time
- Higher freq requires more temporal decomposition.

Resolution.

Mother wavelet $\psi(t)$

$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)$

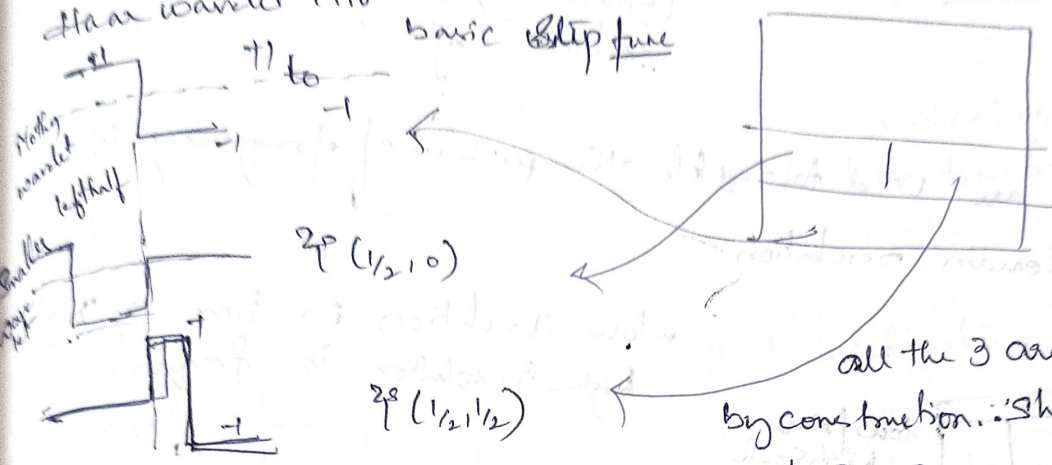
ψ gaussian

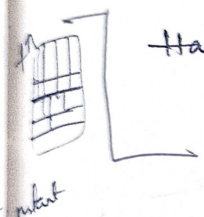
a → makes the wave bigger or small

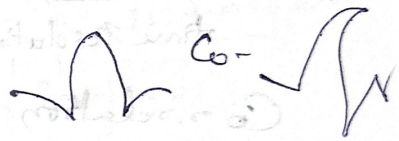
a↑ going up the levels making the windows smaller & smaller

$W_\psi(a,b) = \langle f(t), \psi_{a,b}(t) \rangle$

Haar wavelet 1910



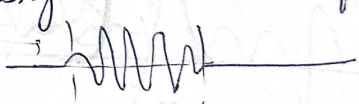




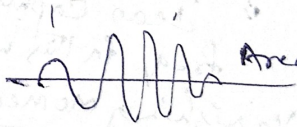
FT → approximates the signal as sum of Sines & Cosines

WT → ~~Sum of~~ as shot bit wave

WT → ① Characteris is Compact support → Signal doesnot last forever



② Area underneath the curve = 0



③ $X(F) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi Ft}\, dt$

F → frequency; $e^{-j2\pi Ft}$ → analysing fn.

$X(a,b) = \int_{-\infty}^{\infty} x(t)\, \psi_{a,b}^{*}(t)\, dt$

x(t) → time

2×2 Matrix

Co-efficients identified by Scale transformation → 3D

amplitude → scale; translation

Translation

Scale (↑ ↓ pitch)

High freq
Low scale

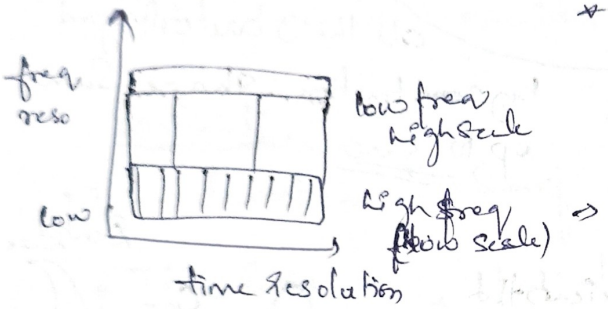
low scale

better localization in time

low freq
scale
(high scale)
stretch ↑

wavelets are used to tackle the problem of frequency in time domain resolution

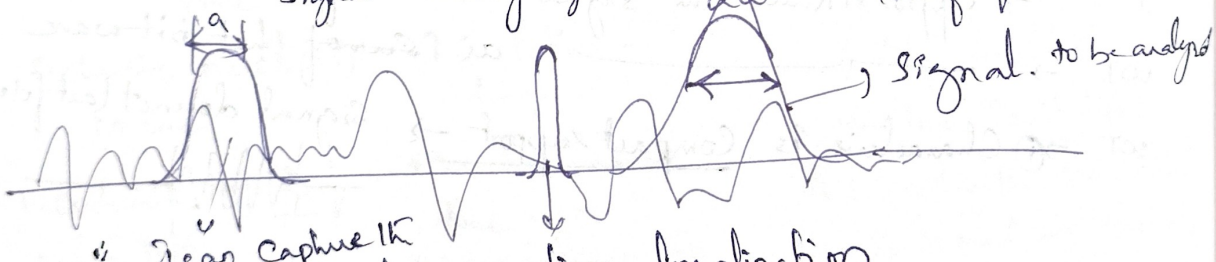
* low resolution in time & high
High resolution in freq



Co relation

$$X(a,b) = \int_{-\infty}^{\infty} x(t) \; \psi^2_{a,b}(t)\, dt$$

signal — analysing



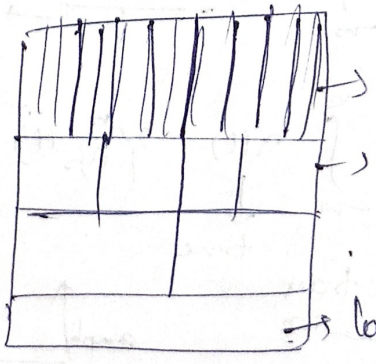
* Can capture the freq in this window.

Vanishing moments.

time localization
Capture high freq.

* higher no of vanishing moments = more complex wavelets

Multi resolution analysis



Big wind - lot of freq Cos
No time resolution

Gabor

short time wavelets (a) : - Scaling

translation in time (τ) a

eg: a slide τ

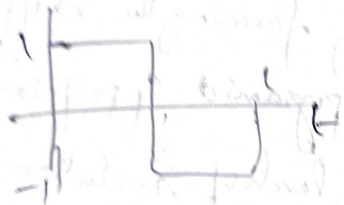
Big window for low freq
small ——— ↑ freq → localized in time

Mother wavelet → little wave

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)$$ a, b, real a ≠ 0

a - scaling
b - translation

Haar $\psi(t)$



Excellent localization

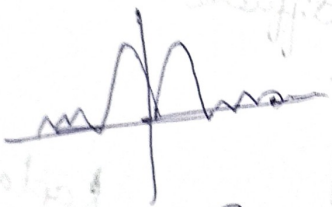
1 : $0 \le t < 1/2$

-1 : $1/2 \le t < 1$

0 : else

F.T of step fun is Sinc fun

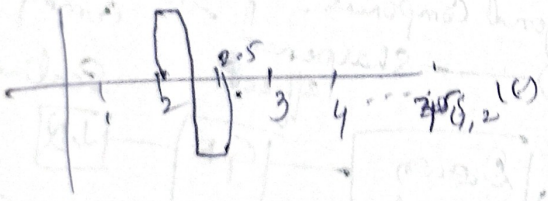
bad localization in freq



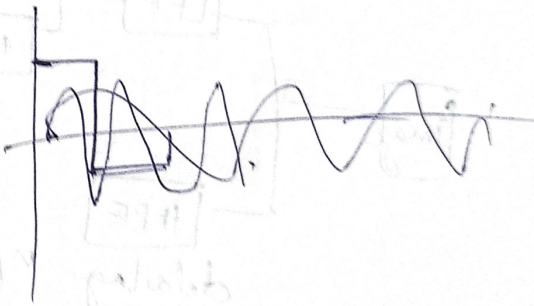
a = 1/2 b = 2.

$\psi_{(a,b)}(t) = \psi_{1,0}(t)$

↑ ↑
Sa translation



$\psi_{0.5,2}(t)$



in freq

Wavelet transforms.

Wavelet → A wavelet is a waveform of effectively limited duration (Some variable BW) that has a average value of zero.

Sine wave

wavelet (.

$$\psi_t(s,\tau) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

s → Scaling factor for expansion & Compression

τ → translation.

Different wavelet → ① Haar ② Bi orthogonal → (forward transform uses one set of waveform & Reverse transform uses the other set of waveform) → Together they are orthogonal but individually they are not).

③. Gaussian 

A wavelet function is always associated with a scaling function which covers frequencies from dc.

* Continuous wavelet transform (CWT) given by

$$C(\tau,s) = \frac{1}{\sqrt{s}} \int_{t=-\infty}^{\infty} x(t)\,\psi^*\left(\frac{t-\tau}{s}\right) dt$$

→ Convolution operation

$x(t)$ ↓ 1-D signal. ↓ transform to cover the full signal.

u, v → Cartesian transform.

τ, s → translation & Scaling

Discrete wavelet transform. (DWT) given by

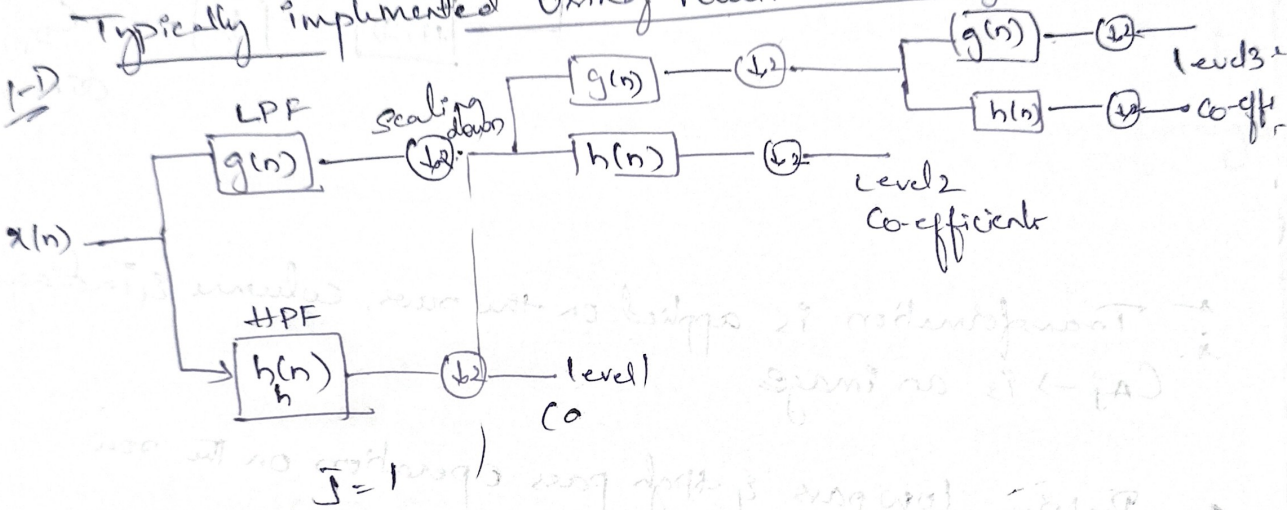
$$D(m, a^j) = \frac{1}{\sqrt{a^j}} \sum_{n=0}^{N-1} x[n]\, \psi^*\left(\frac{n - m}{a^j}\right)$$

Convolution b/n Original function & wavelet function; same it is Repeated for each of the scales.

The expression for DWT is similar to the standard Convolution form and hence implemented using digital filter.

Typically implemented using multi-rate Signal processing

1-D

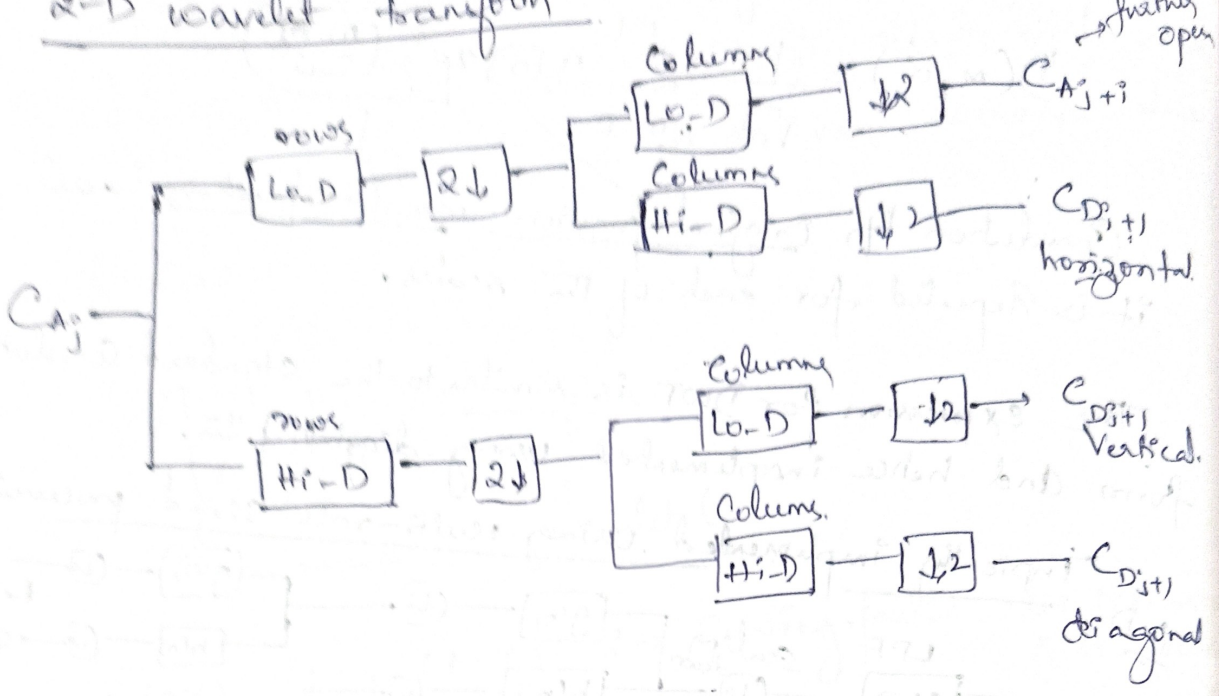


h(n) — father wavelet

g(n) → Mother wavelet.

2-D wa

2-D wavelet transform



- Transformation is applied on the rows, columns & in diagonals.
- C_{A_j} → is an image
- Perform low pass & high pass operation on the rows.
- Lo_D is the Convolution operation.
  2↓ Scaling in row dim.
- Low changing Region LPF → approximation co-efficients
  High freq Components → Detailed co-efficients.

Visualization of wavelet transform

1-level decomposition