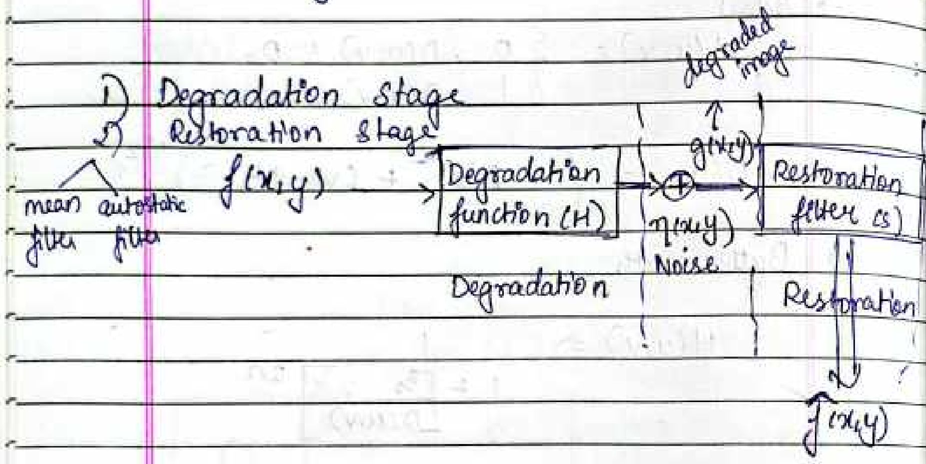
25/08/25 Unit-3 Image Restoration

* Recovering / Reconstructing the image from the noises added in any stage of image enhancement.
* Noise during image acquisition → blurring of image → recovering that → Restoration.

1) Degradation Stage
2) Restoration Stage



→ In Spatial Domain:

$$g(x,y) = f(x,y) \overset{\text{convolution op}^n}{*} h(x,y) + \eta(x,y)$$

If no noise in degradation stage, $g(x,y) = f(x,y) + \eta(x,y)$

→ In Frequency domain:

$$G(u,v) = F(u,v)H(u,v) + N(u,v)$$

refer properties of Fourier transform (mult → convolution op^n for spatial freq? domain)

If no noise in degradation stage:
$G(u,v) = F(u,v) + N(u,v)$ mult → in spatial

⇒ **Noise Models**

* adding noise in image degradation stage.

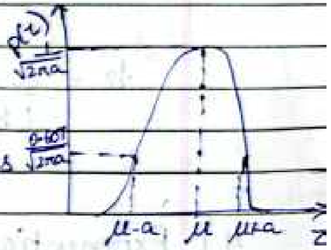
Types of Noises:

1) Gaussian noise: any kind of random noise during image processing system

* defined by a new Probability density func $p(z)$ (particular intensity value)

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$$

where,
z - intensity of pixel
μ - mean value of intensities
σ - standard deviation.

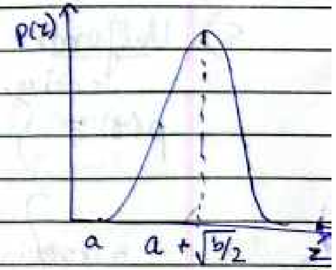


2) Rayleigh Noise
* Because of distribution of noise / prob. density func.

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b}, & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases}$$

mean $= a + \sqrt{\pi b/4}$

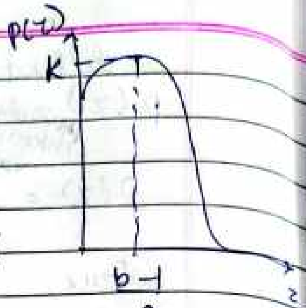
variance $= \frac{b(4-\pi)}{4}$



3) Erlang (Gamma) Noise
* used in medical fields (Refer textbook)
for medical restoration of images.

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} z^{-ab}, & \text{for } z \geq 0 \\ 0, & \text{for } z < 0 \end{cases}$$

mean = b/a



standard deviation = b/a^2

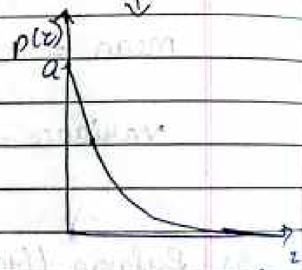
$$k = \frac{a(b-1)^{b-1}}{(b-1)!} e^{-(b-1)}$$

4) Exponential Noise :

* observed in medical imaging, laser technology.

$$p(z) = \begin{cases} a e^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases} \quad a > 0, \ \mu = 1/a, \ \sigma = 1/a^2$$

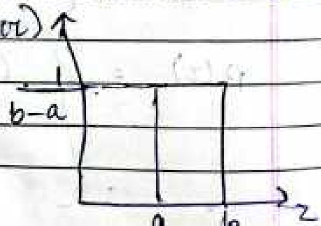
5) Uniform Noise



$$p(z) = \begin{cases} \frac{1}{b-a} & , a < b \\ 0 & , a > b \end{cases}$$

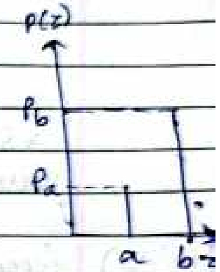
$M = \frac{a+b}{2}$; $S.D = \frac{(b-a)^2}{2}$

* ~~does~~ this type of noise doesn't exist practically.



6) Impulse Noise (Salt and Pepper Noise)

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$



refer Gonzales's book → Need for degradation

* If either of a or b becomes zero, It's called unicollas noise (unicolumn?)

26/08/25. ~~Perid~~

Periodic Noise → electrical / electromagnetic

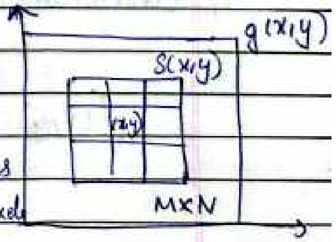
* happens mainly in image acquisition phase
* sinusoidal noises of varying freq.
* can be reduced using frequency domain noise reduction methods.

Parameters for Estimation of Noises:

* Mean → $\bar{z} = \sum_{i=0}^{L-1} z_i P_s(z_i)$
* Variance

↳ $\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 P_s(z_i)$

1. → Mean Filter :



(x,y) → center pixel.

↳ applying mean filter on this & estimating value of all the pixels

1) Arithmatic Mean filter (AMF)

$$\hat{f}(x,y) = \frac{1}{MN} \sum_{(s,t) \in S_{x,y}} g(s,t)$$

* Smoothens the image thereby reduces the noise by blurring.

2) Geometric Mean filter (GMF)

$$\hat{f}(x,y) = \left[\prod_{(s,t) \in S_{x,y}} g(s,t) \right]^{1/mn}$$

* comparatively, smoothens the image more.
* disadv: some fine details of the img will be lost.

3) Harmonic Mean filter

$$\hat{f}(x,y) = \frac{MN}{\sum_{(s,t) \in S_{x,y}} 1/g(s,t)}$$

* salt noise & gaussian

4) Contraharmonic mean filter: Q = order of filter (order imp)

$$\hat{f}(x,y) = \frac{\left[\sum_{(s,t) \in S_{x,y}} g(s,t)\right]^{Q+1}}{\left[\sum_{(s,t)} g(s,t)\right]^{Q}}$$

+ve → pepper
-ve → salt noise

Q = 0 → AMF
Q = -1 → HMF

01/09/25

2. ⟹ Order Static Filters
* based on ordering or ranking of neighbouring pixels.

1) Median Filter

$g(x,y)$

$S(x,y)$

18	5	15
23	(x,y)	17
25	19	10

M×N

arrange in order:
~~18, 5, 15, 23, 17~~:
5, 10, 15, 17, 18, 19, 23, 25
Median: 18 (replace (x,y) with 18)

$$\hat{f}(x,y) = \text{median}\{g(s,t)\} \quad (s,t) \in S_{xy}$$

* Reduces impulse noise
* 50^{th} percentile filter.

2) Max and Min filter

Max filter $\hat{f}(x,y) = 25 \Rightarrow \max_{(s,t) \in S_{xy}} \{g(s,t)\}$

→ Max filter : used to find the brightest part/point of the image.
* reduce pepper noise
* 100^{th} percentile filter.

→ Min filter : $\min_{(s,t) \in S_{x,y}} \{g(s,t)\} = 5$

* used to find the darkest point of image
* reduces salt noise.
* 0^{th} percentile filter

3) Midpoint filter

$$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\}\right]$$

* reduces Gaussian & Uniform noise

4) Alpha - trimmed mean filter.

* remove the $d/2$ no. of lowest & highest intensity values when arranged in order

$$g_R(s,t) = MN - d$$

$$\hat{f}(x,y) = \frac{1}{mn - d} \sum_{(s,t)\in S_{x,y}} g_R(s,t)$$

$d/2$ cutoff | $d/2$ cutoff

* If $d=0$, arithmatic mean filter
* Avg. of intensity values of $MN-d$ pixels.
* if $d = MN-1 \Rightarrow$ median filter.
* reduce salt & pepper noise, gaussian noise

3. ⇒ Adaptive Filters:

* behaviour changes based on statistics of surrounding pixels (statistics → mean & variance)

Adaptive local noise production filter | Adaptive median filter

Mean:
* avg. intensity value

Variance:
* contrast of the Image.

1) $g(x,y)$ = noise present in (x,y) location (img with noise / digraded image)
2) $\sigma_\eta^2 \rightarrow$ variance of $\eta(x,y) + f(x,y)$ ⇐ * response of centre pixel depends on 4 var.
3) M_L - mean value of all pixels in S_{xy}
4) σ_L - variance of S_{xy}

* if $\sigma_\eta^2 = 0$,
$g(x,y) = f(x,y)$ $[\eta(x,y) = 0]$
* $\sigma_L^2 > \sigma_\eta^2$, $f(x,y) > g(x,y)$
* if $\sigma_L^2 = \sigma_\eta^2$, ⇒ arithmatic mean filter.

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}\{g(x,y) - M_L\}$$

Q)

2	6	4	3
5	8	10	12
14	16	18	12
18	8	10	3

Find median, max & min, alpha trimmed, mid-point filters for the pixel in the box $[d=2]$.

⇒ 2, 3, 3, 4, 5, 6, 8, 8, 10, 10, 12, 12, 14, 18, 18

02/09/25

Adaptive Median Filter

* Size of the window pixels (in box) (x,y) is increasing.
* Z_{min} - Minimum intensity value in $S_{x,y}$
* Z_{max} - Maximum " "
* Z_{med} - Median " "
* Z_{xy} - intensity values at (x,y)
* S_{max} - maximum allowed size of $S_{x,y}$

→ Algorithm happens in two stages A & B

Explanation from Gonzalez's book.

Stage A:

$A1 = Z_{med} - Z_{min}$

$A2 = Z_{med} - Z_{max}$

If $A1 > 0$ AND $A2 < 0$, goto Stage B

Else Increase the window size of S_{xy}

If window size $\leq S_{max}$, repeat stage A.

Else output Z_{med}.

Stage B:

$B1 = Z_{xy} - Z_{min}$

$B2 = Z_{xy} - Z_{max}$

If $B1 > 0$ and $B2 < 0$, output Z_{xy}

Else Output Z_{med}.

Note * Order Static filters, Mean & Median Filters → In spatial domain (for image enhancement)

• Image Restoration in Frequency domain

→ Selective Filters:

* Bandreject filter
* Bandpass "
* Notch "

→ Bandreject Filters: ~~act~~ blocks a range of freq.

i) Ideal Band Reject filter.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - w/2 \\ 0 & \text{if } D_0 - w/2 \leq D(u,v) \leq D_0 + w/2 \\ 1 & \text{if } D(u,v) > D_0 + w/2 \end{cases}$$

$D(u,v)$ → distance from the centre of frequency rectangle

W → width of the band.

D_0 → the radial center of the band.

* Remove specific range of freq. while allowing freq. outside that range to pass through.

ii) Butterworth Band Reject filter

$$H(u,v) = 1 \Big/ \left[1 + \left(\frac{D(u,v)W}{D^2(u,v) - D_0^2} \right)^{2n} \right]$$

* n → order of the filter.
* freq that are allowed to pass through the filter do so smoothly w/o any bumps or unevenness.

iii) Gaussian Band Reject filter.

* Stopband → the range of freq. blocked.

$$H(u,v) = 1 - \exp\left[-\frac{1}{2} \left(\frac{D^2(u,v) - D_0^2}{D(u,v)W} \right)^2 \right]$$

* block specific range of freq while allowing others to pass through with a minimal change.

⇒ Bandpass filter:

* allows a specific range of freq to be passed & blocks the rest.

$$H_{BP}(u,v) = 1 - H_{BR}(u,v).$$

BP → Bandpass

BR → Bandreject

⇒ Notch filter:

* can act as both of above.

$$H_{NP}(u,v) = 1 - H_{NR}(u,v)$$

NP → notch pass

NR → " reject

→ Inverse Filtering → (also in freq domain)

* Restore original image from degradation stage

(Fourier Transforms → freq domain)

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

$\hat{F}(u,v)$ = Transformed image in freq domain

$G(u,v)$ = degraded Image

$H(u,v)$ = degradation func

$$G(u,v) = F(u,v)H(u,v) + \underbrace{N(u,v)}_{\text{Noise}}$$

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

→ also in freq domain

→ Minimum mean square error (Wiener) filtering

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right] G(u,v)$$

$$e^2 = E\{[f(x) - \hat{f}(x)]^2\}$$

↳ error measure

$K = \dfrac{S_\eta(u,v)}{S_f(u,v)}$ → Power spectrum of noise / → Power spectrum of ungraded image

03.09.25

Q) Adaptive Median filter

value of circled pixel

18	8	10	14	16	20	18
4	6	8	10	8	4	12
5	7	10	13	14	15	16
8	10	12	20	22	24	18
4	8	2	10	6	4	12
3	5	4	7	9	12	14
8	10	13	20	22	12	18

$S_{x,y} \rightarrow 3 \times 3$

$S_{max} \rightarrow 5 \times 5$

↳ max. size that $S_{x,y}$ can grow to from the current size.

Order: 2, 6, 10, 10, 12, 13, 14, 20, 22

Soln. Z_{med} : 12 ~~Zmin~~ Z_{min} : 2 Z_{max} : 22

$A1 = 12 - 2 = 10$

$A2 = 12 - 22 = -10$

$Z_{xy} = 20$ $B_1 = 20 - 2 = 18$ $B_2 = -2$

Q)

2	6	4	3
5	8	10	12
14	16	18	12
18	8	10	3

8, 10, 16, 18

7.09.25 **Image Compression Model** (Unit 4).

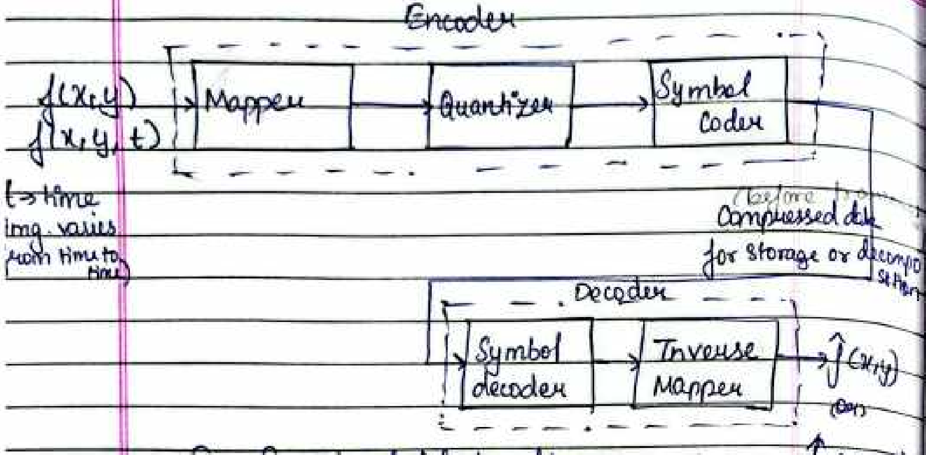


Fig. Fractional block diagram

Mapper: reduce the temporal & spatial redundancy
Quantizer: remove any irrelevant info from image
Symbol coder: all intensities converted into a coded info (uses codeword - some bit representation)

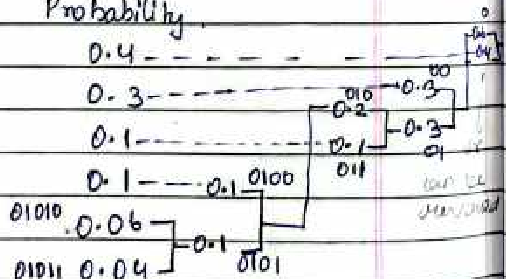
1) → **Huffman Coding**

* for each intensity value, the info is coded (codeword) ↓ different for each
* find probabilistic values for each intensity/symbols (for compression) (as in histogram)
* only assign binary values (0 & 1) — variable compression, Block coding scheme

$a_1 - 0.1,\ a_2 - 0.4,\ a_3 - 0.06,\ a_4 - 0.1,\ a_5 - 0.04,\ a_6 - 0.3$

(descending order)

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



(or use the tree representation)

codeword for $a_1 a_2 a_3 a_4 a_5 a_6$
011 1 01010 0100 01011 00

* More probability → less bits required & vice versa

$L_{avg} = \sum P_r(r_k)\, l(r_k) = 0.1\times3 + 0.4\times5 \ldots$ (probab., length of bits used to rep)
$= 2.2$ bits/symbol → after compression (b') (no. of bits)

* comparison → symbols → ASCII value (before compression) → 7 bits
no. of bits in rep after compression.

→ Compression ratio:

$\dfrac{\text{no. of bits before compression}}{\text{no. of bits after compression}} = \cancel{\dfrac{42}{20} = 2.1}$ (7×6)

no. of bits before comp: $0.4\times7 + 0.3\times7 - - - 0.04\times7 = 7$ (ASCII value) $7/2.2 = 3.18$

→ Entropy: (use formula for entropy)

$$H = -\sum_{k=1}^{L-1} P_r(r_k)\log_2 P_r(r_k)$$

$= +0.65$

10/09/25

Q) Perform huffman coding & calculate compression ratio & entropy. → use histogram

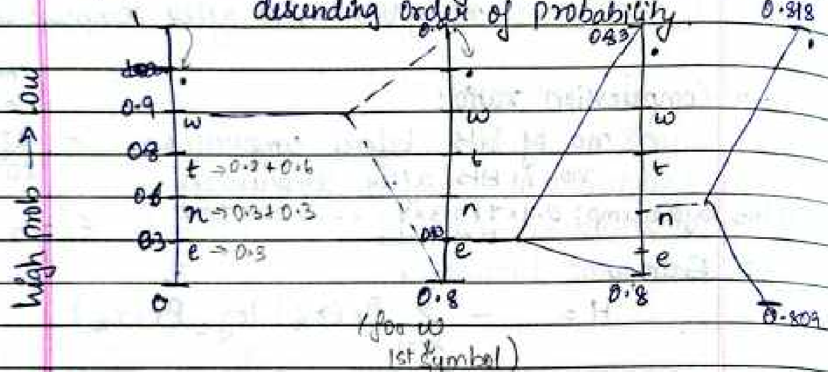
7	4	3	1
1	3	6	5
6	3	2	1
0	0	7	7

2) ⇒ Arithmetic Coding

→ Non block codes [Single codeword for all symbols]
→ Lossless data compression

Ex: data : went.
probability of e : 0.3, n = 0.3, t = 0.2, w = 0.1, ·=0.1

Step 1: Divide and assign values b/w 0 to 1 in descending order of probability.



Step 2: Find range of each symbol. For that find d first. which is:

d = upper limit − lower limit.

range of symbol: lower limit : lower limit + d × {prob. of symbol}

Step 3 for w:
d = 0.1   lower limit = 0.8
~~range~~ = 0.8 : 0.8 + 0.1 × 0.3 = 0.83

range for e
n = 0.86   t = 0.88   w = 0.89   · = 0.9 (upper limit)

Step 3. for e :
d = 0.83 − 0.8 = 0.03
e = 0.8 + 0.03 × 0.3 = 0.809   ~~t = 0.809 + 0.03 × 0.2 = 0.815~~
~~w = 0.815~~   n = 0.818   t = 0.824   w = 0.827

Step 4 for n :
d = 0.009
~~e = 0.809 + 0.009 × 0.3 = 0.8117~~
e = 0.8117   n = 0.8144   t = 0.8162   w = 0.8171

Step 5 for t :
d = 0.8162 − 0.8144 = 0.0018

e = 0.8144 + 0.0018 × 0.3 = 0.81494
n = 0.81548   t = 0.81584   w = 0.81602

Step 6 for · :   Step 6 → do for · → will be in below range.
d = 0.81664 − ~~0.8144~~ 0.8162 = 0.00
L.L   U.L
0.81602 < codeword < 0.8162

codeword/tag = (upper limit + lower limit)/2 = 0.81611

15/09/25

3) ⇒ Run length Encoding
* vertically
* horizontally
* Zig-zag

Image:

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

5x5

→ Horizontal :
1st row : (0, 5) (value, count)
2nd row : (0, 3) (1, 2)
3rd row : (1, 5) & 4th, 5th row

* run length vectors or pairs.

bits reqd to represent one pixel : 1 ⇒ (0 or 1)
Max count : 5 ⇒ no. of bits reqd : 3
6 rep ⇒ 6 × (3+1) = 24 bits reqd to represent the image

before compression bits reqd : 25
after " " : 24

compression reqd : $\frac{25}{24} = 1.04$

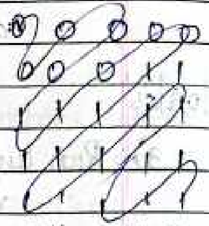
⇒ Vertical :

1st col. = (0,2) (1,3) ; 4th col = (0,1), (1,4)
2nd col. = (0,2) (1,3)   5th col = (0,1), (1,4)
3rd col = (0,2) (1,3) ⇒ 10 vectors
max length : 4 ⇒ 3 bits.
$10 \times (3+1) = 40$ bits are reqd.

compression ratio = $\frac{25}{40} = 0.625$

↳ No compression (no. of bits reqd after compression is more)

⇒ Zig-Zag :



1st scan : (0,1)
2nd : (0,2)
3rd : (0,2) (1,1)
4th : (1,2) (0,2)   6th : (1,4) ; 8th : (1,2)
5th : (0,1) (1,4)   7th : (1,3) ; 9th : (1,1)

bits reqd = $12 \times 4 = 48$
⇒ No compression

4) Bit plane coding
* divide img into different planes.

monochrome → max value = 7 (3 bits)

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

so for a monochrome img : max bits reqd ⇒ 3

binary rep : 111 100 011 001
001 011 110 101
110 011 010 001
000 000 111 111

MSB	mid Bit	LSB
1100	1010	1011
0011	0110	1101
1000	1110	0101
0011	0011	0011

⇒ 3 planes
next step : use run length coding separately for each plane.

MSB : Horizontal	Mid : ×	LSB : ×
(1,2) (0,2)	(1,2) (0,2)	(1,3) (0,1)
(0,2) (1,2)	(0,2) (1,2)	(1,3) (0,1)
(1,1) (0,3)	(1,3) (0,1)	(0,2) (1,2)
(0,2) (1,2)	(0,2) (1,2)	(0,2) (1,2)
bits : $8 \times 4 = 32$	$8 \times 4 = 32$ ↓	$8 \times 4 = 32$ ↓

compression = $\frac{48}{32} = 1.5$

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

16/09/25

5) LZW (Lempel - Ziv - Welch) Coding
* reduces spatial redudancy
* fixed length codeword

spatial redudancy (some intensities are repeated)

img :	127	127	127	127
	25	25	25	25
	25	25	127	127
	127	127	127	127

here, max value ⇒ 127 ⇒ 7 bits
16×7 totally.
Instead of that, if A = 4(127)s, B = 2(127s) 127
C = 4(25)s, D = 2(25)s ⇒ reduces no of bit for same values
↳ 4 values (25) grouped togthr

Dictionary location	Entry
0	0
1	1
2	2
⋮	⋮
255	255
⋮	127-127 } ex
⋮	25-25 } ex
511	⋮

Q) 39 39 126 126
39 39 126 126
39 39 126 126
39 39 126 126

→ if not in dict then current val = encoded o/p

Currently recognised seq.	pixel being processed	Encoded output (where we hv stored the value) ↳ location	Dictionary location	Dictionary Entry
already in dict no need to process —	39	—	—	—
39	39	39	256 (dict has 0-255 already)	39-39
39-39 ↓ not in dict so encoded o/p will be the value 39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
already exist 39	39	—	—	—
39-39	126	~~39-39~~ 256	260	39-39-126
126	126	—	—	—
126-126	39	258	261	126-126-39
39	39	—	—	—
39-39	126	~~260~~ —	—	—
39-39-126	126	260	262	39-39-126-126
126	39	—	—	—
126-39	39	257	263	126-39-39
39	126	—	—	—
39-126	126	257	264	39-126-126
126	—	126	—	—

Encoded o/p combo:

39-39-126-126-256-258-260-257-257-126

↳ 10 values ⇒ bits reqd ⇒ 10 × 9 = 90 bits
↳ dict. storage

before compression: 16 × 8 = 128 bits.

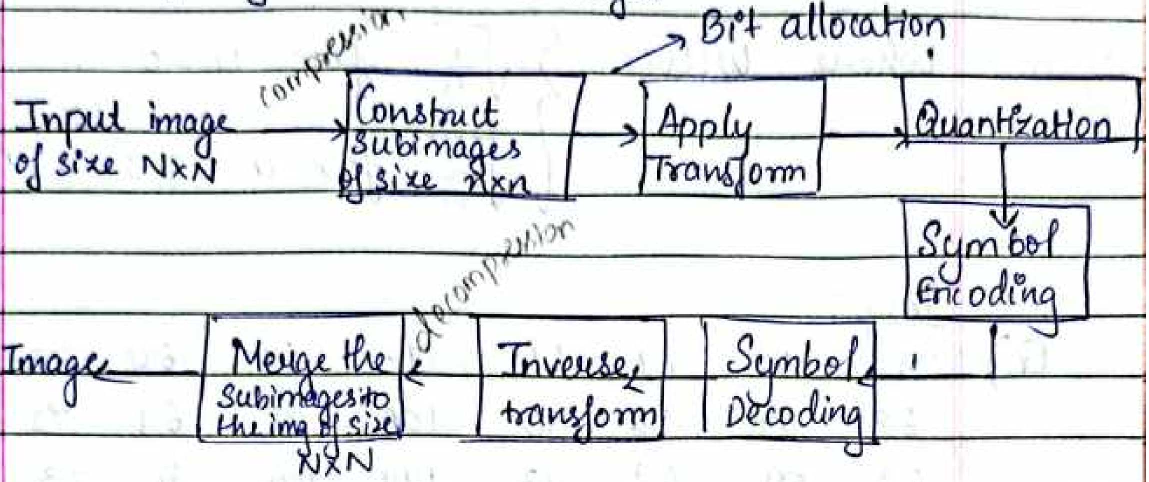
23.09.25

⇒ Block Transform Coding

* Lossy compression method.
* Divide org. img into subimages

$N \times N \longrightarrow n \times n$

I/p Img. — Subimages



Bit allocation:

deciding no. of bits reqd for transformation

→ Coding Types:

Refer Gonzalo's book

⇒ Zonal Coding → conc. in one region

⇒ Threshold Coding → separating values based on threshold.

* Here, DCT is preferred over DFT coz DFT has complex values also in the result. (i,j etc). but DCT gives integral values.
* JPEG img uses DCT.

general form:

n should be in power of 2. (usually 8)

Forward kernel:

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

for $u,v = 0,1,2 - - -, n-1$

Inv./Reverse kernel

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

for $x,y = 0,1,2 - - -, n-1$

$r(x,y,u,v)$ and $s(x,y,u,v)$ basis func/img

$\rightarrow$ **DCT.** (Discrete Cosine Block Transform Coding)

$$F(u,v) = w(u)\,w(v) \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

Where $w(u) = \begin{cases} \sqrt{\frac{1}{n}} & \text{for } u = 0 \\ \sqrt{\frac{2}{n}} & \text{for } u = 1,2,3,\ldots,n-1 \end{cases}$

Power of 2 marks

8×8

Img:

Q) $\boxed{n=8}$

52	55	61	66	70	61	64	73
63	59	66	90	109	85	64	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	65	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Step 1: Level Shifting

Subtract 128 from each intensity value.

Final rang: -128 to 127.

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-64	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-63	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

Date | Page

Step 2: Apply DCT. using the given formula

$f(0,0) = -76$ intensity value
$x\ y$

$\rightarrow$

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	11	-4	-1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	1	0	-1

$\Rightarrow$ Transfo Coefficient

Step 3: Apply Quantization

$$F_Q(u,v) = \text{Round}\left[\frac{F(u,v)}{Q(u,v)}\right]$$

$\hookrightarrow$ table given. (Quantization table).

For JPEG imgs:

$Q(u,v)$

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Step 4 : Take values in zig-zag order. Omit the 0's & encode the remaing values

-26 -3 1 -3 -2 -6 2 -4
1 -4 1 1 5 0 2 0 0 -1
2 0 0 0 0 0 -1 -1 EOB
(End of block)

Next: Decompress (Refer Gonzalez's book)

24.09.25

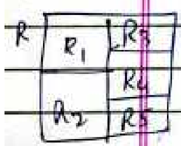
Image Segmentation

* extracting a particular region from the image for processing & ignoring the surrounding ↳ called ROI (Region of Interest).
* Ex: tumor separation from surrounding parts
* Partitioning img. into different regions/images and getting boundaries b/w them.
* I/P: org img O/P: Features/attributes of the img.

ROI:-

* divided into two parts: based on different properties of pixels

Similarity principle (Region approach) Discontinuity principle (boundary approach)



Img R:

10	10	25	25	25
10	10	25	25	25
20	20	25	25	25
20	20	25	25	25
20	20	25	25	25

R_1, R_2, R_3 ⇒ 3 regions

⇒ Properties

Suppose R is an image, R_i is a region in the image

1) $R_1 \cup R_2 \cup R_3 \cup \dots \cup R_n = I(R)$
i.e $\bigcup_{i=1}^{n} R_i = R$

2) R_i should be a connected region, $i = 1, 2, \dots, n$

3) $R_i \cap R_j = \phi$ (R_i, R_j are diff regions), for all i, j ; $i \neq j$

4) $P(R_i) = TRUE$ for $i = 1, 2, \dots, n$
P ⇒ Predicate ⇒ properties of the region (texture, colour, intensity val, etc)

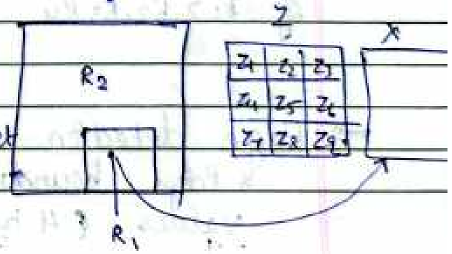
5) $P(R_i \cup R_j) = FALSE$, $i \neq j$ (R_i, R_j are diff region)

→ Similarity principle
- → Thresholding
- → Region growing
- → Region split
- → Region Merge

Discontinuity principle
- → Isolated point
- → Line detection
- → Edge detection ↳ 2 totally diff region

using the mask from sharpening filter (Laplacian mask)

⇒ Isolated point:
* Superimposing the image & wherever we get greater than threshold value → isolated point



Response value, $R = \sum_{i=1} z_i x_i$

$$g(x,y) = \begin{cases} 1 & \text{if } R(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

T = Threshold value

* Laplacian mask with $-4/-8$ as centre pixel. $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - ↳ First order derivative.

⇒ Line detection

* Line / edge / boundary are different.
* 4 masks with (of the masks) $-45/+45°$ as centre pixel ↳ slanted mask.
* uses 2nd order derivative
* 4 masks → 4 Responses (for each direction)

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Horizontal mask, vertical mask, $+45°$, $-45°$ (Slanted masks)

$$\boxed{R = \sum_{k=1}^{4} w_k x_k}$$ → Response value for 4 masks

30.09.25

* If $R_i > R_j$; $\forall j \neq i$ ⇒ the point is in dirn of mask i.

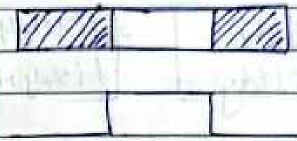
Ex: R_1 (Horiz.), R_2 (vert.), R_3 (45°), R_4 (+45°) (Find all response values by superimposing the masks on img)

If $R_1 > R_2, R_3, R_4$ ⇒ The point belongs to a horizontal line.

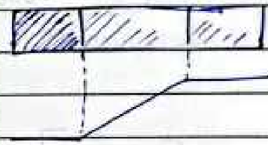
⇒ Edge detection

* Edge : boundary b/w 2 different intensity values. 4 types :

1) Step Edge

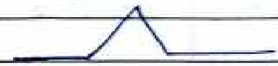


2) ~~Roun~~ Ramp edge (Gradual & slow change in value).



3) Spike edge

4) Rod edge.



⇒ Steps :

1) Image smoothing for noise reduction
2) Detection of edge points – potential candidate
3) Edge localization.

Edge detection

- First order derivative using gradient mask
  - ↳ Sobel
  - ↳ Robert
  - ↳ Prewitt.
- Second order derivative Gaussian based
  - ↳ Laplacian of Gaussian
  - ↳ Canning (scanning) edge detector.

Gradient of image $f(x,y)$:

$$\Delta f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f/dx \\ \partial f/dy \end{bmatrix} \rightarrow \begin{bmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{bmatrix}$$

$$\frac{\partial f}{\partial y} = \nabla f$$

$$\boxed{\frac{\partial f}{\partial x} = f(x+1) - f(x) \quad (1^{st}\text{ order})}$$

→ **Robert mask:**

$$\begin{matrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{matrix} \qquad \text{mask} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$\frac{\partial f}{\partial x} \Rightarrow f(x+1,y) - f(x,y) = z_9 - z_5 \rightarrow \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$
↳ Horizontal

$$\frac{\partial f}{\partial y} = f(x,y+1) - f(x,y) = z_8 - z_6 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$
↳ vertical

→ **Sobel's operator:**

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 - 2z_2 + z_3)$$

$$= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{(horizontal)}$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{(vertical)}$$

---

* disadv of robert mask → led to sobel mask

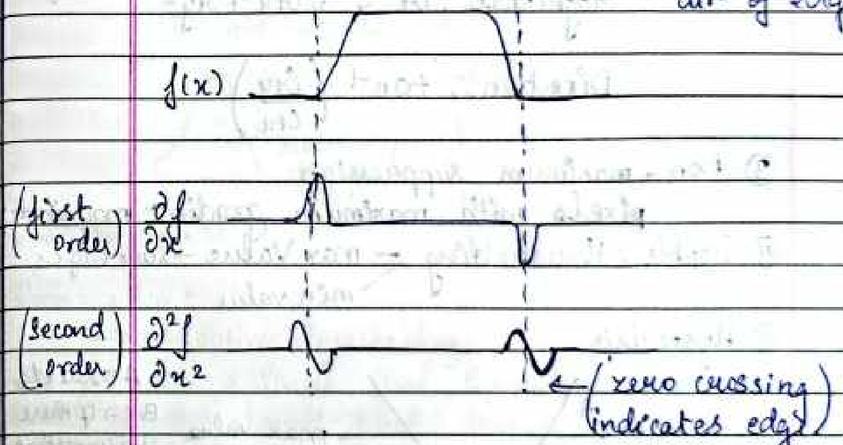
→ **Prewitt operator.**

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$
↳ horizontal ↳ vertical

06.10.25

⇒ **Edge detection using 2nd order derivative**

→ **Laplacian of gaussian:** ⇒ (can't find the dir^n of edge)



$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

→ $\nabla^2 G$:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \rightarrow (1)$$

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \rightarrow (2)$$

→ Canny Edge Detection →(can find the dirⁿ of edge)

① Noise Reduction – Apply the gaussian filter
② Finding the intensity gradient of the image

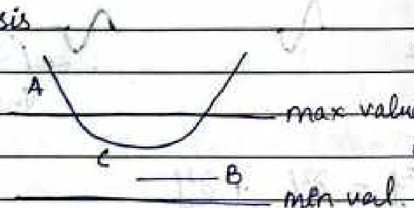
Magnitude, $G = \sqrt{Gx^2 + Gy^2}$

Direction, $\tan^{-1}\left(\frac{Gy}{Gx}\right)$

③ Non-maximum suppression
- pixels with maximum gradient magnitude

④ Double-thresholding ← max Value → sure edges (edge with value > max)
← min value ; value < min

⑤ Hysterisis



A → sure edge
B, C → if there is a connectivity with a sure edge we take that as edge
∴ C → sure edge
B → discarded

- Based on Similarity Principle:
- → Thresholding → Region growing
- → Region split → Region merge

→ Thresholding:
* key parameter → thresholding value.

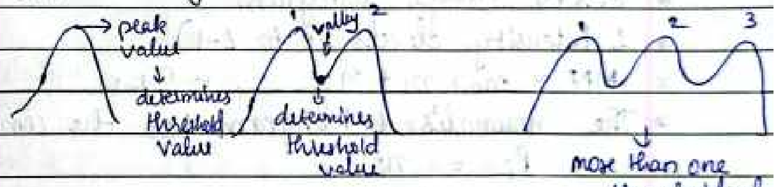
① Global thresholding → gray level value only $f(x,y)$
② Local thresholding → depends on neighbourhood pixel $p(x,y)$ values
③ Adaptive or Dynamic thresholding ← based on Spatial Location (x,y) of pixel

$Th = T[x, y, f(x,y), p(x,y)]$

x, y → Spatial location; $f(x,y)$ → intensity value of pixel; $p(x,y)$ → neighbourhood pixel property.

Types of histogram

Unimodal histogram · Bimodal · Multimodal



→ Adaptive Thresholding
* Single level thresholding → $f(x,y) > T$ → identifies object
* Multi-level thresholding → $f(x,y) < T$ → identifies background

→ Global Thresholding

$$G(x,y) = \begin{cases} 1 & f(x,y) > T \\ 0 & f(x,y) \le T \end{cases}$$

Algorithm:
1) Randomly select an initial threshold value T
2) Segment the image into two groups G_1 and G_2 based on T.

3) Determine mean (m_1) of the pixels in G_1 that lies below T.
Determine mean (m_2) of the pixels in G_2 that lies above T.

4) New Threshold, $T_{new} = \frac{1}{2}(m_1 + m_2)$

5) Repeat the steps from 2 to 4 until the difference in T in successive iterations is less than a particular limit T_0.

07.10.25

→ Otsu Thresholding (Optimum Global thresholding method)

Algo → Gonzalez book

* $M \times N$ image histogram
* L intensity levels (0 to L-1)
* $MN = n_0 + n_1 + n_2 \dots + n_{L-1}$ ⇒ pixels with intensity value L-1
* The normalised histogram has the component:

$$P_i = \frac{n_i}{MN}$$

$$\sum_{i=0}^{L-1} P_i = 1 \quad ; \; P_i \geq 0$$

$T(k) = k \qquad 0 < k < L-1 \qquad C_1 \,\&\, C_2$

$C_1 \to$ 0 to K intensity values

$C_2 \to$ K+1 to L-1 pixel values

$$P_1(K) = \sum_{i=0}^{K} P_i$$

$$P_2(K) = \sum_{i=K+1}^{L-1} P_i = 1 - P_1(K)$$

Mean intensity value of the pixels in C_1 is $m_1(K) = \sum_{i=0}^{K} i P(i/C_1)$

$\hookrightarrow m = \sum^{L-1} r_i P(r_i)$

$$m_1(K) = \sum_{i=0}^{K} i P(i/C_1) = \sum_{i=0}^{K} i \frac{P(C_1/i) P(i)}{P(C_1)}$$

$$= \frac{1}{P_1(K)} \sum_{i=0}^{K} i P_i \quad , \text{where } \sum_{i=0}^{K} P(C_1/i) = 1$$

$$m_2(K) = \sum_{i=K+1}^{L-1} i P(i/C_2) = \frac{1}{P_2(K)} \sum_{i=K+1}^{L-1} i P_i$$

The cummulative mean upto level k is given by

$$\boxed{m(K) = \sum_{i=0}^{K} i P_i}$$

Average intensity or Global mean of entire image is :-

$$\boxed{m_G = \sum_{i=0}^{L-1} i P_i}$$

$\eta = \frac{\sigma^2 B}{\sigma^2 G}$, $\sigma^2 G$ is the Global variance, $\sigma^2 B$ is the interclass variance

$$\sigma^2 G = \sum_{i=0}^{L-1} (i - m_G)^2 P_i$$

$$\sigma^2 B = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2$$

$$= \frac{(m_G . P_1 - m)^2}{P_1 (1 - P_1)}$$

$$\boxed{\sigma^2 B(k) = \max_{0 \leq K < L-1} \sigma^2 B(k)}$$

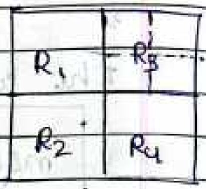
$$\eta(K) = \frac{\sigma^2 B(K)}{\sigma^2 G}$$

$$\sigma^2 B(K) = \frac{[m_G P_1(K) - m(K)]^2}{P_1(K)[1 - P_1(K)]}$$

- Region based Segmentation
  * Region growing
  * Region split & merge.

① Region Growing
  * make subregions into a single group, so some similarity / homogenity should be there.

Steps :
1. Selection of initial seed
2. Seed growing criteria
3. Termination of segmentation process



1	0	7	8	7
0	1	8	9	8
0	0	7	9	8
0	0	8	8	9
1	2	8	8	9

S_1 S_2
9   1

$T \le 4$

$|f(x,y) - f'(x,y)| \le 4$

$(1-9) = 8 \le 4$ x

$7-2 = 5 \le 4$ x

possible values of S_1:

$S_1 \to \{\cancel{5}, \cancel{6}, 7, 8, 9\} = 1$

$S_2 \to \{0, 1, 2\} = 0$

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

Q) Perform the Global thresholding Algo for the given image :

5	3	9
2	1	7
8	4	2

initial threshold = avg of pixels

Date
Page

$Th = \frac{41}{9} = 4.56$

$G_1 = 1, 2, 2, 4, 3$

$m_1 = \frac{\cancel{24}\ 12}{5} = 2.4$

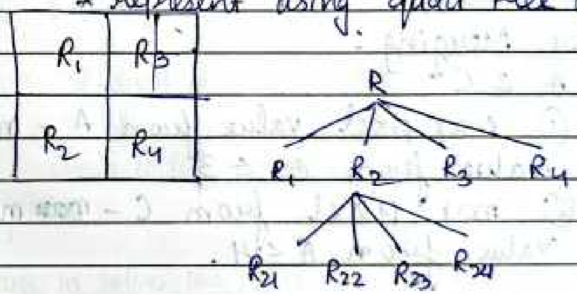
$G_2 = 5, 7, 8, 9$

$m_2 = \frac{29}{4} = 7.25$

$T_{new} = \frac{1}{2}(2.4 + 7.25) = 4.825$

08.10.25

② Region split and Merge Algorithms :
  * Subregions should not have common properties ⇒ disjoint subregions. ⇒ for splitting
  * common properties → for merging
  * represent using quad tree (all the quadrants)



no 2 regions have same properties

Steps / Phases :
1. splitting until some criteria. $P(R_i) = FALSE$
2. merging till the regions have some common properties
   ↳ $P(R_i \cup R_j) = TRUE$

Q) img :

max = 7, min = 4, 7 - 4 = 3, cond^n satisfies, don't divide

max = 7, min = 2, 7 - 2 = 5, 5 ≰ 3 (so divide)

(A) (B) (B_1, B_2, B_3, B_4)

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

(C) (D) (D_1, D_2, D_3, D_4)

|max val - min value| ≤ 3

T ≤ 3

|7 - 0| ≰ 3

* Divide into 4 Quadrants until the cond^n satisfies.

Cond^n for Merging :

A & C

1. max pixel value from A - min. pixel Value from C : ≤ 3.
2. max pixel from C - min pixel Value from A ≤ 4.

For B_1 & B_2 ⇒ the cond^n satisfies so merge

B_{12} & B_3 ⇒ cannot merge

B_{12} & B_4 ⇒ merge.

B

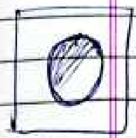
7	7	6	6
5	5	4	7
3	2	5	6
2	3	4	6

(B_{12}, B_3, B_4)

III^ly, do for all the quandrants.

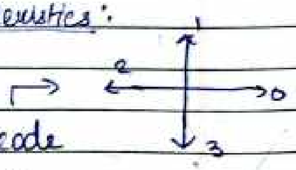
→ Boundary Representation and discription

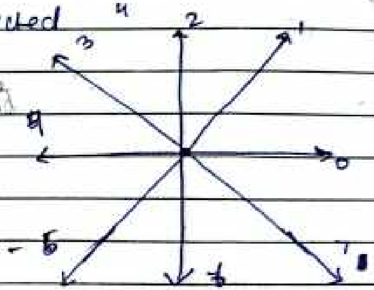
* Based on :
  → shape etc.
  → External characteristics → based on boundary
  → Internal Charocteristics → based on pixel properties



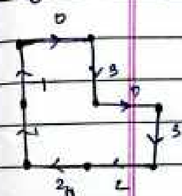
→ Based on External Characteristics :

1. Chain code :
   i) 4 - directed chain code
   ii) 8 - directed





obj 1



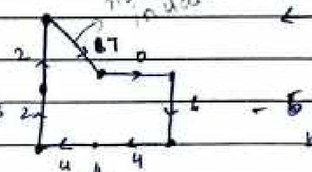
Apply 4-directed.

⇓

Chain code = 03032211

obj 2

slanted edge not available in 4 directed

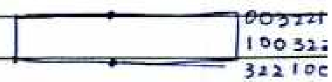


8-directed

⇓

Chain code = 7064422

Q)

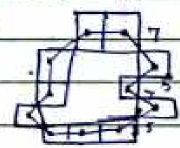


multiple rep is possible

→ ways to solve the problem (which to choose)

1) Normalize w.r.t Starting point → Whichever has less magnitude will be considered as chain code
   here, 3221 is smallest (as a no.) so, i.e the chain code.
2) Normalize wrt direction. → anticlockwise always
   → path thro the image.

Q)



→ 07575443 12

order = 12 (whichever has min. order, consider that)

First difference : 7626707 61

Shape no. : 0762670761