DIGITAL IMAGE PROCESSING NOTES

AKSHANSH CHAUDHARY

Digital Image Processing Notes, First Edition

Presented by: Akshansh Chaudhary
Graduate of BITS Pilani, Dubai Campus
Batch of 2011

Course content by: Dr. Jagadish Nayak
Then Faculty, BITS Pilani, Dubai Campus

Layout design by: AC Creations © 2013

The course content was prepared during Spring, 2014.
More content available at: www.Akshansh.weebly.com

5.2.14

CLASSMATE

Digital Image Processing

* Digital signal processing : 1D ≡ 1 variable ≡ $f(x)$
* Digital image processing : 2D ≡ 2 variables ≡ $f(x,y)$

* CONVOLUTION

1D : for $h(x)$: impulse response,

$$y(x) = f(x) * h(x)$$

2D : for $h(m,n)$: impulse response

$$y(k,l) = f(x,y) * h(x,y)$$

* When we see an image / anything, we see the reflection after light falls on it.
So, an image, taking it a f^n,

$$f(x,y) = i(x,y) \cdot r(x,y)$$

$i(x,y)$ → 0 to ∞

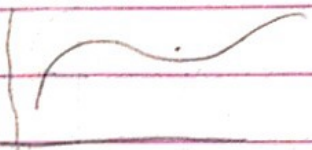
$r(x,y)$ → 0 to 1 (1 → total reflection; 0 → no reflection)

* DIGITIZATION
ADC (Analog to Digital Conversion)

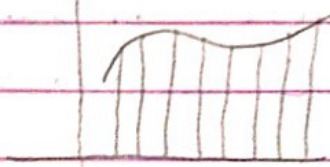
Digitization = Sampling + Quantization (+Coding)

1D Digitizn :

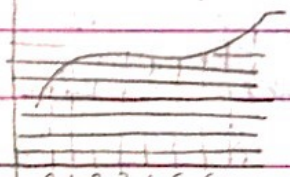
Analog



Sampling



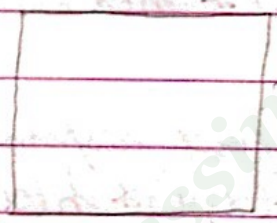
Quantization



↗ depends on the no. of levels used
i.e, for 8 bit data, 2^8 levels]

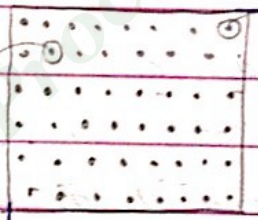
2D Digitizn :

Signal



→ can be considered a picture

Sampling



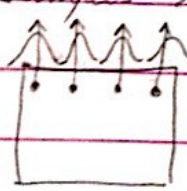
→ represents columns

→ each dot is a sample, called as PIXEL

→ represents rows

Nyquist criteria for 2D

Idea : every sample is a pixel. Pixel is basically a color. That color has a spectrum. So, the criteria is, choose samples s.t. spectrum doesn't overlap



→ spectrum of pixel

* Light intensity levels

0 to 255

(Black/Dark to White/Bright)

CLASSMATE

- Human eye can see these levels of light intensity.
- these levels are called GRAY LEVEL(Scale)
- 0 to 255 are called Grayscale values of light intensity levels
- this is QUANTIZATION in 2D.
  - each pixel value varies from 0 to 255 (for Grayscale images only)

Image is represented as a matrix.

values of pixels

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ \vdots & & & \vdots \\ f(M-1,0) & f(M-1,0) & & f(M-1,N-1) \end{bmatrix}$$

- values in matrix depends on intensity levels, which is decided upon the quantizn of the amplitude level.

* Stegnography : hiding info. inside image.

0 - 255 in binary bits

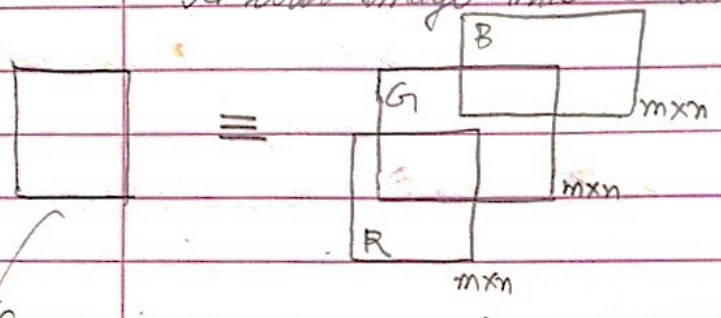
- Idea : every pixel is represented by bits & info. can also be written in bits. → ASCII values
- Replace LSB of pixel bits by the info. bits
- Human eye won't be able to detect the change in image.

✱ COLOR IMAGES.

They are modelled by a model called as RGB model.

RGB → Red, Green, Blue → Primary colors.

A color image has 3 components :



∃ grayscale in red, green and blue. So, each pixel in red (green or blue) can have value from 0–255.
So, 8 bits are req^{d} to represent them.

✱ Each pixel in a color image requires 24 bits (8 bits R + 8 bits G + 8 bits B) to represent it.

↳ ✱ processing of each R, G & B sheets happens separately

Image → (converted to) → Grayscale → Processed.

✱ **Digital Image Representation**

↳ Levels : Gray levels (Range 0 to L−1)

✱ Dynamic range of imaging sys :
Ratio of max. measurable intensity to min. detectable intensity level in the sys.

✱ Upper limit is determined by the satur^{n} & lower limit is noise.

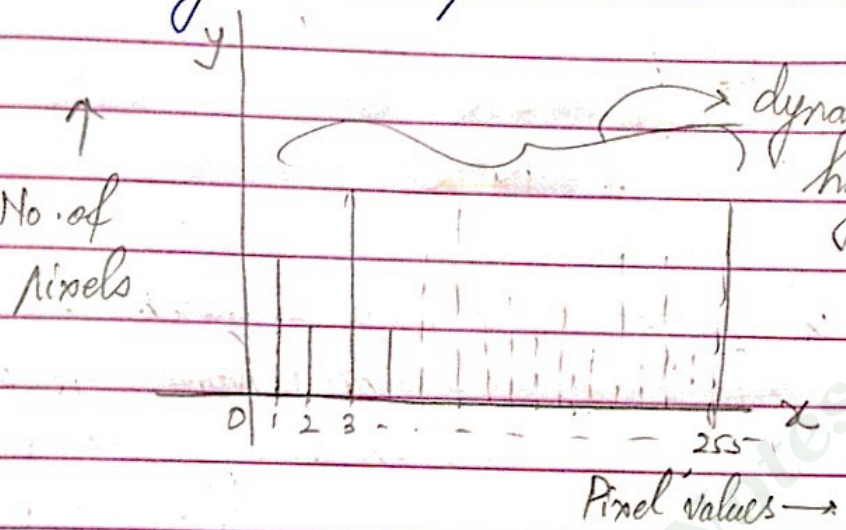
✱ Contrast : Diff. in intensity b/w highest & lowest intensity level in the image.

✱ When dynamic range is very high, image is said to have high contrast.

CLASSMATE

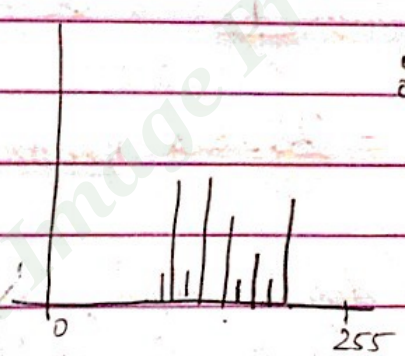
* Histogram:

A plot having no. of occurences (pixels) ~~(pixel values)~~ on y-axis & pixel values on x-axis



* If diff. b/w max. value of pixel & min. value of pixel is large, it's called Good Contrast image. (i.e, an image should have all values from 0-255)

eg:-



∴ Suppose this is the histogram. It doesn't have all the values from 0-255. So, it's a Poor contrast image.

* To a process called contrast stretching to get good contrast image from a poor one

* Baud Rate : Speed / Data rate in bits/sec.

CLASSMATE

* Total no. of bits reqd to store a digitized $M \times N$ image is :-

$$b = M \times N \times k$$

(k → no. of bits reqd to store a pixel)

for $M = N$

$$b = N^2 k$$

eg: Find time reqd to transmit 1024×1024 color image through a communication channel with data rate of 1 Mega bits/sec.

no. of bits reqd $= b = 1024 \times 1024 \times 24$

$= 2^{23} \times 3$

$\Rightarrow b = 25165824$

10^6 bits $\longrightarrow$ 1 sec.

25165824 bits $\longrightarrow \frac{1}{1000000} \times 25165824$

$= 25.1658$ seconds

* Spatial Resolution : (define dpi, ~~M~~)

Measure of the smallest discernible detail in an image.

↳ line pairs per unit distance

↳ Dots (pixels) per unit distance (dpi)

* Intensity Resolution : (define no. of levels, k)

Not always we require 256 levels to represent each pixel. We can use less no. of levels (to reduce size). But, if no. of levels are greatly

Page No

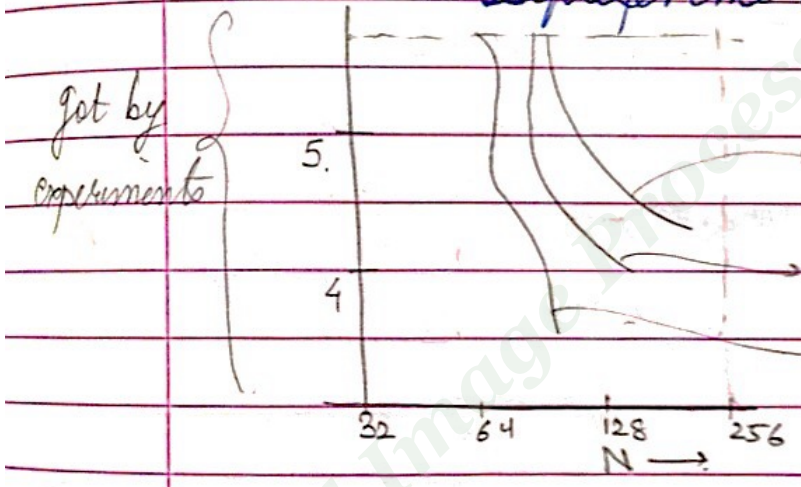
decreased, contour effect occurs : little effect started.

Levels : 256, 128, 64, 32 : Visually no difference
16, 8, 4, 2 : Contour effect takes place.

→ In the areas of constant intensity, suppose for a region with a value 250. It has to be approximated b/w values (from 0-127 for 128 bcoz) So, ∃ some diff in view.

* Spatial & Intensity resolution (N & k) : Values are found experimentally. (based on how humans can see)

↳ Isopreference curve tells rel^n b/w N & k.



* Image Interpolation

↳ used in zooming, shrinking, rotating and geometric corrections

i.e we are making 8X8 → 16X16, say.
i.e, we are adding data in the added space.

So, in ~~the~~ interpol^n, we use known data to estimate values of unknown locations.

• Resampling method :

Gives the effect of zooming & shrinking

* technique used for zooming purpose :

Method 1 : Nearest neighbour interpolation -

eg :- Given a 2×2 image :

5	10
15	20

Make 4×4 image :

i.e, seeing nearest neighbour & putting them as shown

5	5	10	10
5	5	10	10
15	15	20	20
15	15	2	0.

↳ Ideo : fill the pixels (image values) alternatively & then see nearest neighbours & fill

5	10
15	20

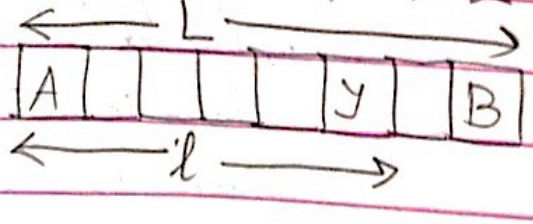
→

5		10	
15		20	

Method 2 : Bilinear interpolation.

↳ makes use of 4 nearest neighbours to estimate intensity at a given loc".

eg : Given A, B, L & l. Find y



let A = 23

B = 30

L = 8, l = 6

Formula :- $\frac{Y-A}{l} = \frac{B-A}{L}$

$\Rightarrow Y = A + l\left(\frac{B-A}{L}\right)$

$= 23 + \frac{36}{4}(7)$

$= \frac{23\times4 + 21}{4} = \frac{113}{4} = 28.5 = 28$ (rounding off)

$\Rightarrow Y = 28$

eg: Using Bilinear interpol^n to following image :

34	30	28	25
20	25	10	28
25	32	30	35
12	34	35	25

L

34	Y=31 (l)	30		23		25
20	.	25		10		28
25		32		30		35
12		34		35		25

Now, find all values in empty space using Bilinear interpol^n.

$Y = 34 + \frac{2}{3}(-4) = 34 - \frac{8}{3} = \frac{102-8}{3} = \frac{94}{3} = 31.3 = 31$

Ily, find ∀ blanks

Ch: Image Enhancement Techniques

Objective: improve quality of image by processing it.

Methods: (1) Spatial domain → Alter pixels directly.
direct manipul^n of pixels of image

(2) Frequency domain → Pixels → fourier coeff → alter
modifying Fourier transform of image

(M1) Spatial domain technique:

- ✓ technique operate directly on pixels.
- ✓ more efficient comput^n & require less processing resources to implement.
- ✓ Spatial domain process is defined by

$$g(x,y) = T[f(x,y)]$$

T → an operator on f, defined over a neighbourhood of a pt. (x,y)

★ if we want to process any pixel with coordinates (x,y), we take neighbours for 3×3 or 5×5. Then based on the oper^n, T, we get the new value of desired coordinate

mainly ←

eg:-

35	6	56	10
2	9	18	19
3	35	17	20
25	30	45	60

→ neighbours of 9

Say, its x,y

Suppose T: average oper^n

⇒ new value = 35+6+56+2+3+35+17+18+9

CLASSMATE

* 2D SIGNALS & SYSTEM:

- ✓ Image is considered as a 2D signal, $f(x,y)$
- ✓ Most popular sys oper^n: convolution.
  - ↳ filtering oper^n performed by convolution

Consider a sys:

$x(n) \rightarrow [h(n)] \rightarrow y(n)$

1D Convolution:

$$y(n) = x(n) * h(n)$$

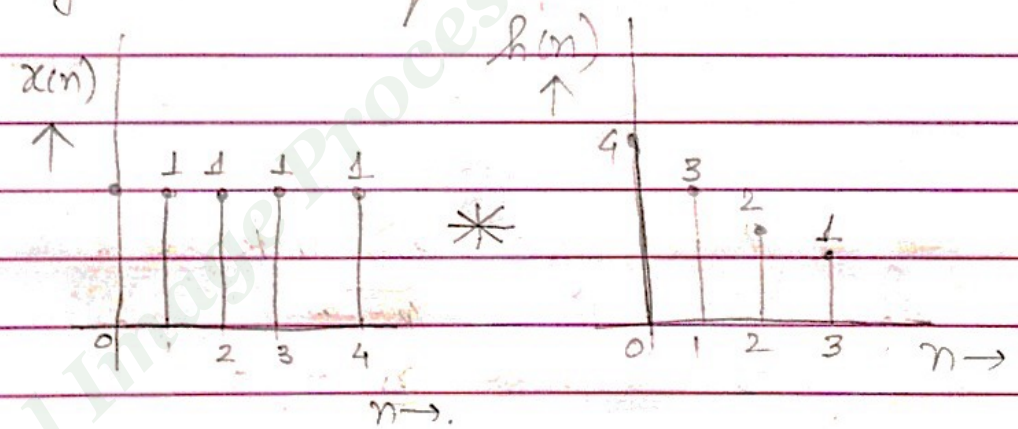
$$y(n) = \sum_{k=0}^{M-1} x(k)\,h(n-k)$$

↳ $n = 0, 1, 2, \ldots (M+N-1)$

↳ M: length of sequence $x(n)$

N: length of sequence $h(n)$

eg: Consider a sequence:-



here, length of n = length of $x(n)$ + length of $h(n)$ − 1

$= 5 + 4 - 1 = 8$

$= 0, 1, 2, 3, 4, 5, 6, 7 \rightarrow y(n)$

Put $n=0$

$$\Rightarrow y(0) = \sum_{k=-\infty}^{\infty} x(k)\,h(-k)$$

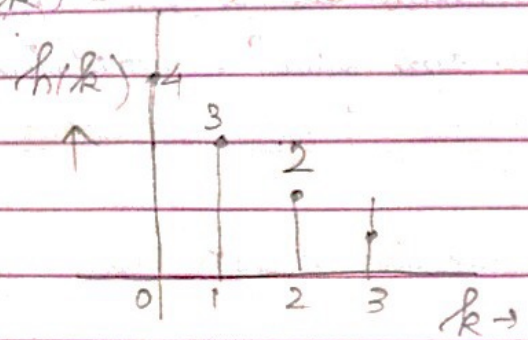
i.e multiply $x(k)$ & $h(-k)$ with $k \in (-\infty, \infty)$

& $h(-k)$ = mirror image of $h(k)$

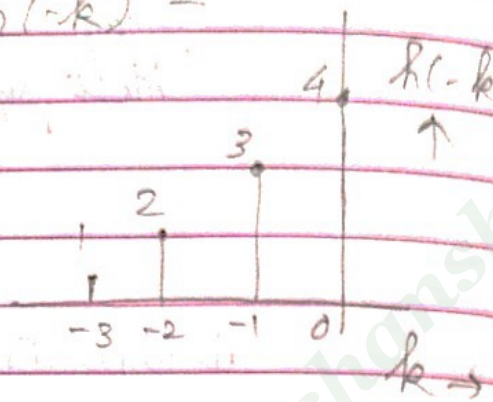
Page No

We had

$h(k)$ =



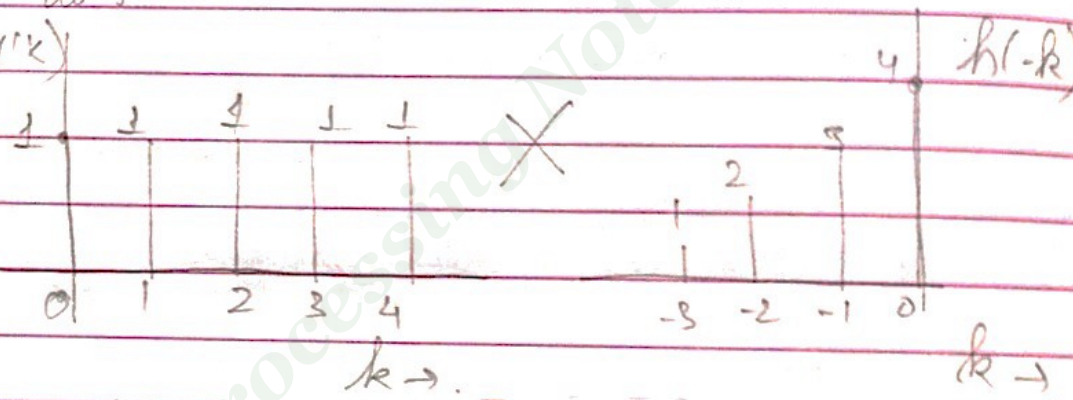
$h(-k)$ =



Idea: Now, multiply $h(-k)$ & $x(k)$

So, multiply coinciding elements & add.

So, we do :-



So, we have, $y(0) = \sum_{k=-\infty}^{\infty} x(k)\, h(-k) = 1 \times 4 = 4$

Ily, do ∀ values of n

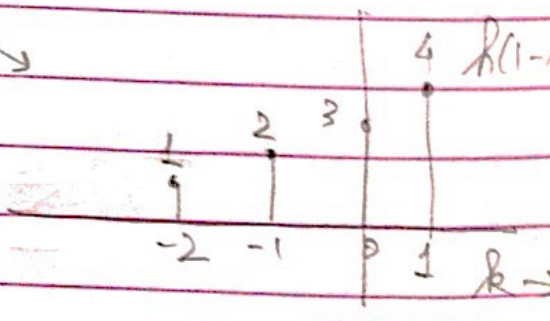
For $n=1$

$$y(1) = \sum_{K=-\infty}^{\infty} x(k)\, h(1-k)$$

Coinciding elements are : 0, 1

$\Rightarrow (1 \times 3) + (1 \times 4)$

$= 7$



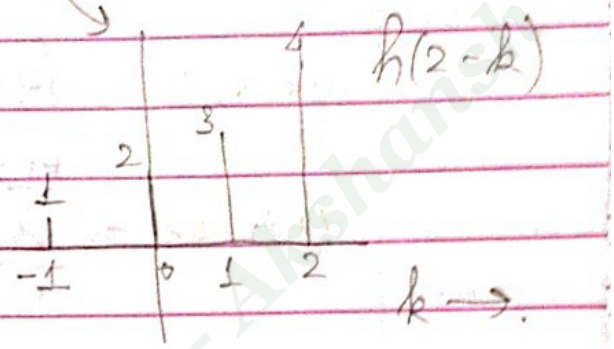
Now, for $n=2$

$$y(2) = \sum_{k=-\infty}^{\infty} x(k)\, h(-k+2)$$

Coinciding elements are at $k = 0, 1, 2$.

So,

$$y(2) = 2\times1 + 3\times1 + 4\times1$$
$$= 9.$$



Now,

$n = 3$,

$$y(3) = 1\times1 + 1\times2 + 1\times3 + 1\times4$$
$$= 10.$$

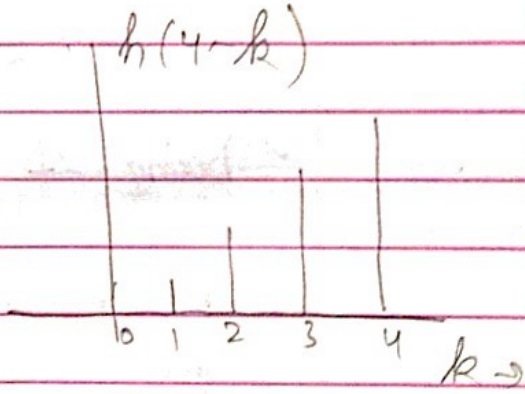
$n = 4$

$$y(4) = \sum_{k=-\infty}^{\infty} x(k)\, h(-k+4)$$

Coinciding $= k = 1, 2, 3, 4$.

$\Rightarrow$

$$y(4) = 1\times1 + 2\times1 + 3\times1 + 4\times1$$
$$= 10,$$



$n = 5$

$$y(5) = 1\times1 + 1\times2 + 1\times3 = 6$$
$$y(6) = 1\times1 + 1\times2 = 3.$$
$$y(7) = 1\times1 = 1.$$

So, finally, discrete convolution values are :-

n	0	1	2	3	4	5	6	7
y(n)	4	7	9	10	10	6	3	1

Ans

Now, considering a 2D system:

$$f(x,y) \rightarrow \boxed{h(m,n)} \rightarrow g(k,l)$$

$$g(k,l) = f(x,y) * h(m,n)$$

$$g(k,l) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, h(k-x, l-y)$$

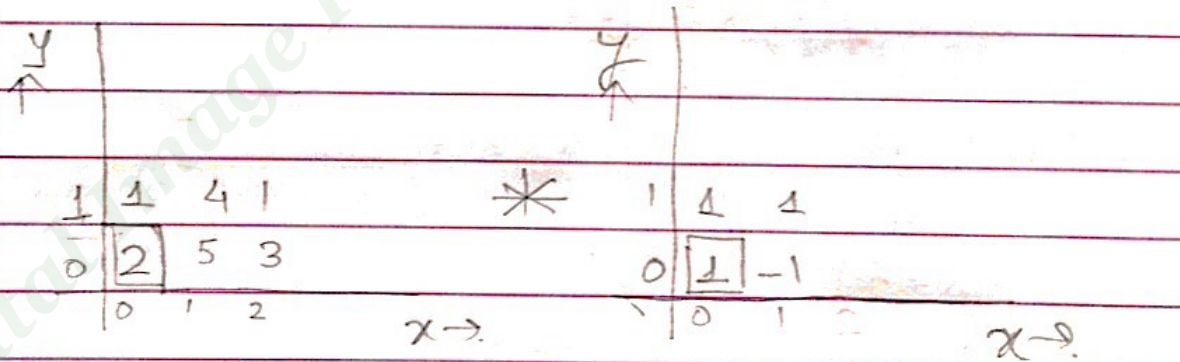
↳ $k = 0, 1, 2, 3, \ldots, (M1+M2-1)$

↳ $l = 0, 1, 2, 3, \ldots, (N1+N2-1)$

* length of signal rows$^{(k)}$ = no. of rows of sequence 1$^{(M1)}$ + no. of rows of sequence 2$^{(M2)}$ − 1

(Ily for columns)

✓ Doing 2D Convolution:



So, in convolved image, no. of columns$^{(k)}$ = 3+2−1 = 4

no. of rows$^{(l)}$ = 2+2−1 = 3

So, $k = 0, 1, 2, 3$

$l = 0, 1, 2$.

So, finally, we'll have:

l ↑				
2	?	?	?	?
1	?	?	?	?
0	?	?	?	?
	0	1	2	3 → k

$g(k, l)$

Now starting with (0, 0)

$$g(0,0) = \sum_{x=0} \sum_{y=0} f(x,y)\, h(-x,-y)$$

So,

y ↑		
	1	1
	[1]	-1
	x →	

$h(x,y)$

Idea :-

$h(x,y)$

↓ flipped wrt x axis

$h(x,-y)$

y ↑			
	[1]	-1	
	1	1	x →

$h(x,-y)$

↓ flipped wrt y axis

$h(-x,-y)$

-1	[1]	
1	1	x →

$h(-x,-y)$

So, now we do :-

1	4	1
[2]	5	3

$f(x,y)$

-1	[1]
1	1

$h(-x,-y)$

=

?	?	?
?	?	?
[2]	?	?

Next, do for (1, 0)

So,

$g(1,0) = \sum\sum f(x,y)\, h(1-x,-y)$

↳ Shift $h(-x,-y)$ by 1 unit in x-axis

[-1]	1
1	1

So,

$g(1,0) = 2\times -1$

$+$

5×1

$\Rightarrow g(1,0) = 3$

Next, for (2, 0) $\Rightarrow k=2,\ l=0$

$\Rightarrow -1\times 5 + 1\times 3$

$= -5+3 = -2$

=.

[-1]	1
1	1

for (3,0) → h(3−x, −y)

Dividing = −1 × 3 = −3

= g(3,0)

	−1	1
	1	1

for (0,1)

h(−x, 1−y)

g(0,1) = 1×1 + 2×1

= 3.

	−1	1
	1	1

for (0,2) h(−x, 2−y)

g(0,2) = 1×1

= 1.

	−1	1
	1	1
		□

for (1,1)

h(1−x, 1−y)

g(1,1) = 1×(−1)
+ 2×1
+ 4×1
+ 5×1 = 10.

	−1	1
	1	1

for (1,2)

h(1−x, 2−y)

g(1,2) = 1×1 + 4×1

= 5.

	−1	1
	1	1
		□

for (2, 1)

$g(2,1) = 4x(-1) + 5(1) + 1(1) + 3(1)$

$= 5.$

$h(2-x, 1-y)$

	-1	1
□	1	1

for (2, 2)

$g(2,2) = 4(+1) + 1(1)$

$= 5.$

$h(2-x, 2-y)$

	-1	1
□	1	1

for (3, 1)

$g(3,1) = 1(-1) + 3(1)$

$= 2.$

$h(3-x, 1-y)$

	-1	1
□	1	1

for (3, 2)

$g(3,2) = 1(1)$

$= 1.$

$h(3-x, 2-y)$

	-1	1
□	1	1

So, finally

$g(k, l) =$

l \ k	0	1	2	3
2	1	5	5	1
1	3	10	5	2
0	[2]	3	-2	-3

Ans

- Smallest possible neighbourhood size is 1×1.
- ∀ elements, there should be 8 neighbours.
- Hence, order of matrix is odd. (like 3×3, 5×5, 7×7, 9×9...) for neighbours (i.e, how many elements do I use to change value of 1 pixel — for enhancement)

eg: Consider an image:

$f(x,y)$ =

2	3	4	5	6
7	8	9	10	10
11	11	12	13	14
3	2	4	5	6
7	8	9	10	11

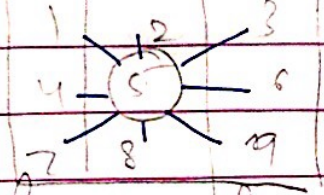
Suppose we take 1×1 neighbour, i.e, using a single point, & lets say, we use the opⁿ:- Add 5 ∀ elements. So, we get

2+5	3+5	4+5	5+5	...
-	-	-	-	-
-	-	-		
	-	-		

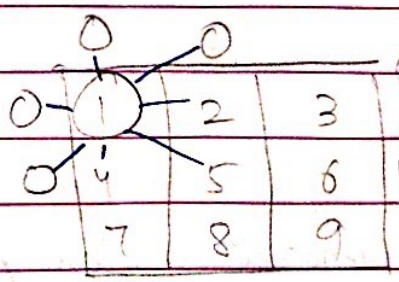
- Point processing or 1×1 & neighbours processing.

Now, suppose we take 3×3 neighbour.
For that, we introduce extra rows & columns for corner elements:

eg



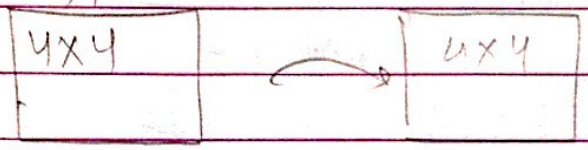
∴ element 5 has 3×3 neighbour.

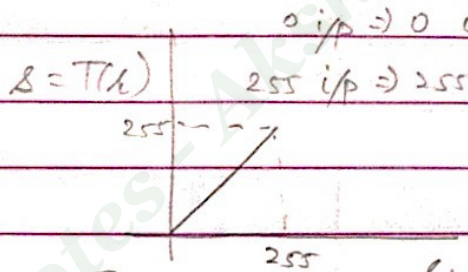


for element 1, add zeros & do opⁿ

* 1×1 neighbourhood operⁿ is called Point Processing and is represented by transformⁿ fⁿ :- $s = T(r)$
  ↳ s & r represent the intensity of g & f respectively.

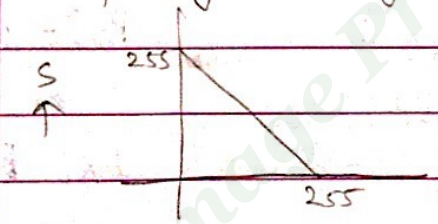
Now, we want to transform :-



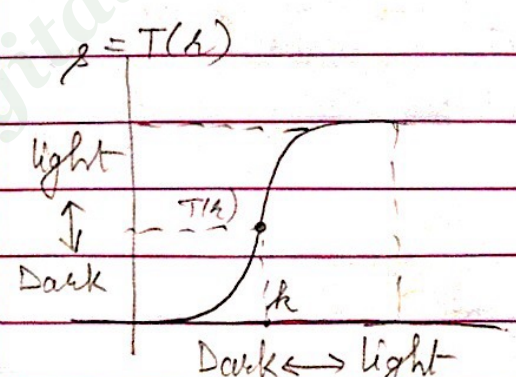


r Vs s
≡ r Vs T(r)
≡ g(x,y) Vs T[f(x,y)]

If we want Negative of image, transformⁿ curve :

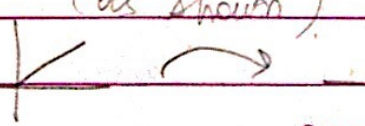


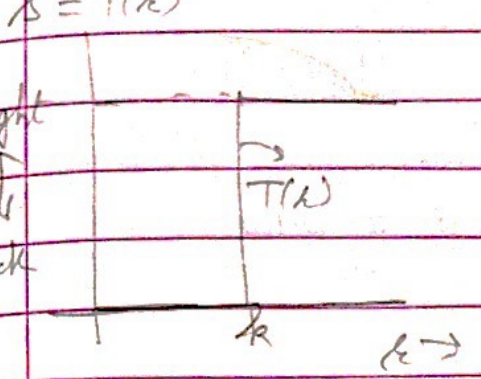
0 i/p ⇒ 255 o/p
255 i/p ⇒ 0 o/p
↳ used in medical.



Contrast stretching fⁿ.

Idea : for a low contrast image, we need to improve contrast. So, we take center point (k) & stretch it towards light & darker side (as shown)

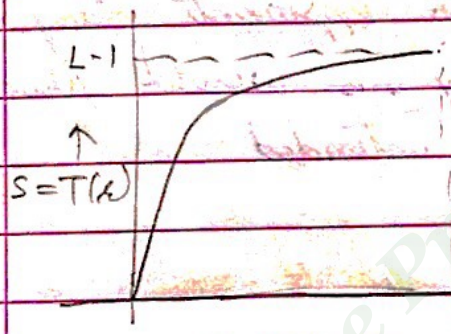




here, we will have only 2 levels : 0 & 255. So only 1 bit is req^d to represent it.

Threshold f^n.

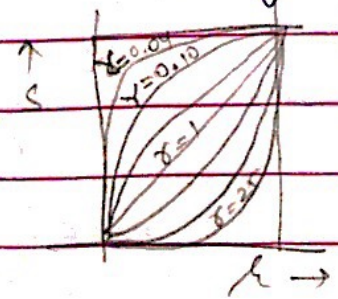
* log transform^ns :
for an image having intensity ranging from [0 L-1], log transform^n is given by
$s = c \log(1+r)$ → for biasing, to make log(+ve)
↳ c: constt
L=256, (taken)



Used when i/p pixels are of small range. So, they are raised to higher level at o/p as shown.
Also, when intensity is high at i/p, it is mapped to smaller range of o/p.
↳ used in displaying Fourier spectrum.

for 1×1 neighbours

* Power-law (Gamma) Transform^n :
↳ Basic form : $s = c r^{\gamma}$
↳ c, γ : +ve constts.
≡ for γ < 1 ≡ log transform^n.



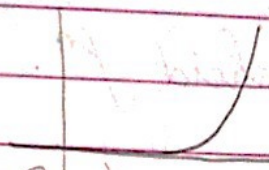
CLASSMATE

* MATLAB Assignment

for γ = less (0.04, say),

for i/p of low intensity (dark value), o/p is of high intensity

for γ = high (25, say),

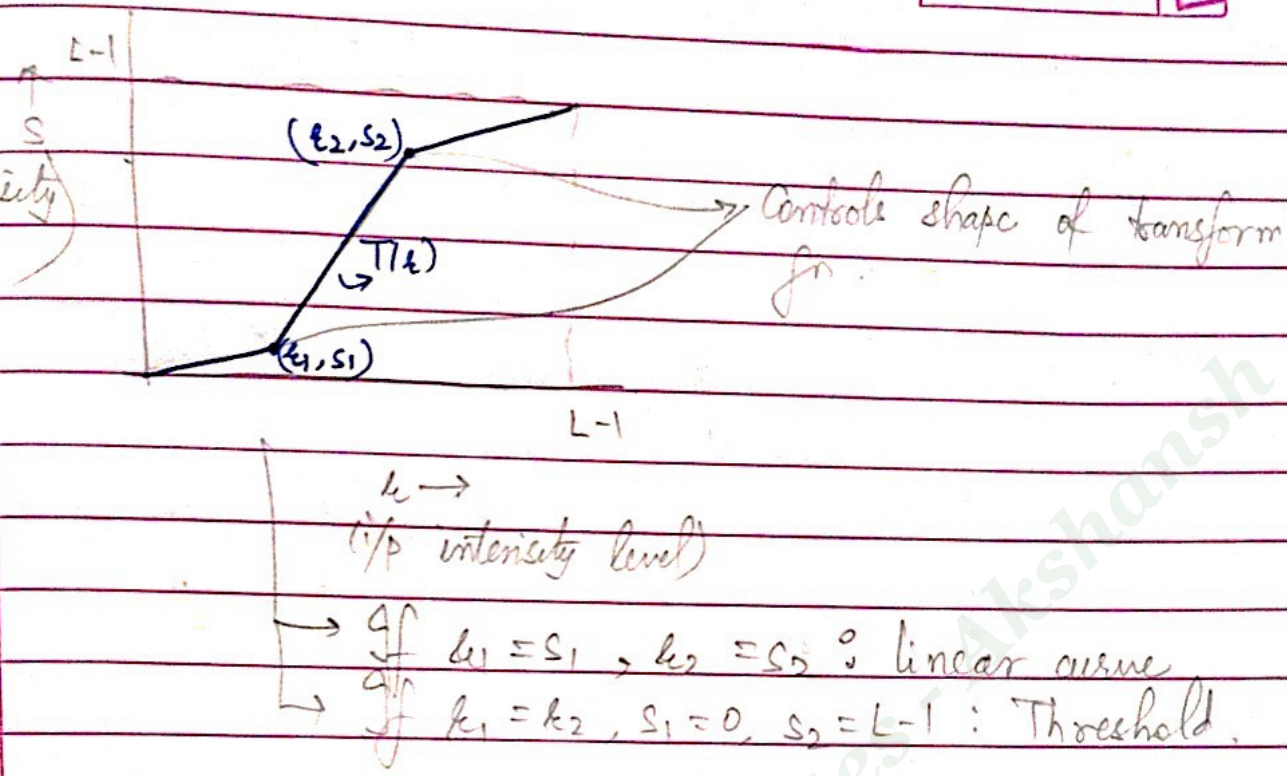


for i/p of high intensity (bright), o/p is of low intensity (darker)

This is also called Gamma correction.

- CRT device have an intensity to voltage response that is a power fn with exponent varying from 1.8 to 2.5. Such display sys. would produce which are darker than intended.
  - eg: when our laptop monitor shows colors and projector doesn't show those colors.
  - eg: when MRI image is not clear
    - Reduce value of γ (say $\gamma = 0.3$) to see the details that were dark
    - Increase value of γ (say, $\gamma = 5$) to see the details that were bright.

* Contrast stretching:
  - low contrast results due to:
    - Poor illumination
    - lack of dynamic range in imaging sensor
    - Wrong settings of lens aperture during acquisition (focus improper)



if $r_1 = s_1$, $r_2 = s_2$: linear curve

if $r_1 = r_2$, $s_1 = 0$, $s_2 = L-1$: Threshold.

eg Contrast stretching :-

```
close all
clear all;
clc;
[filename, pathname] = uigetfile('*.tif');
im = imread([pathname filename]);
imshow(im);
title('POOR CONTRAST IMAGE');

k = im2bw(im, 0.42);
figure, imshow(k)
```

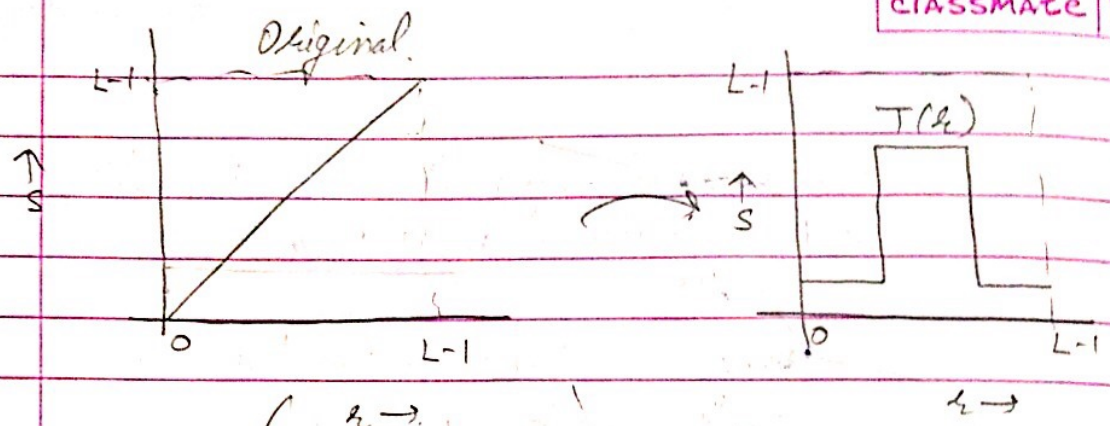
to select file (upload)

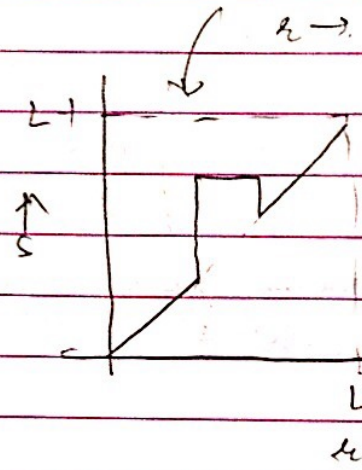
grayscale to B & W image

* Intensity Level Slicing :

↳ highlights specific range of intensities.

eg: enhancing features such as masses of water in satellite imagery, enhancing flaws in X-ray images.





(highlights some portion of image, rest zero)

(highlights some portion of image, rest constant)

eg

```
[filename, pathname] = uigetfile('*.tif');
im = imread([pathname filename]);
z = double(im);          → note all pixels in a matrix
[row, col] = size(z);    → arrange in rows & columns
for i = 1:1:row,
for j = 1:1:col.
if ((z(i,j) > 142) & (z(i,j) < 250))      ← If image is of this range
z(i,j) = 255;            → highlight it (make it brightest)
else
z(i,j) = im(i,j)         → rest, keep constant
end
end
end
figure(1);
imshow(im);
figure(2);
imshow(uint8(z));
```

z(i,j) = 0. (to make other pixels = 0)

MATL
Assign

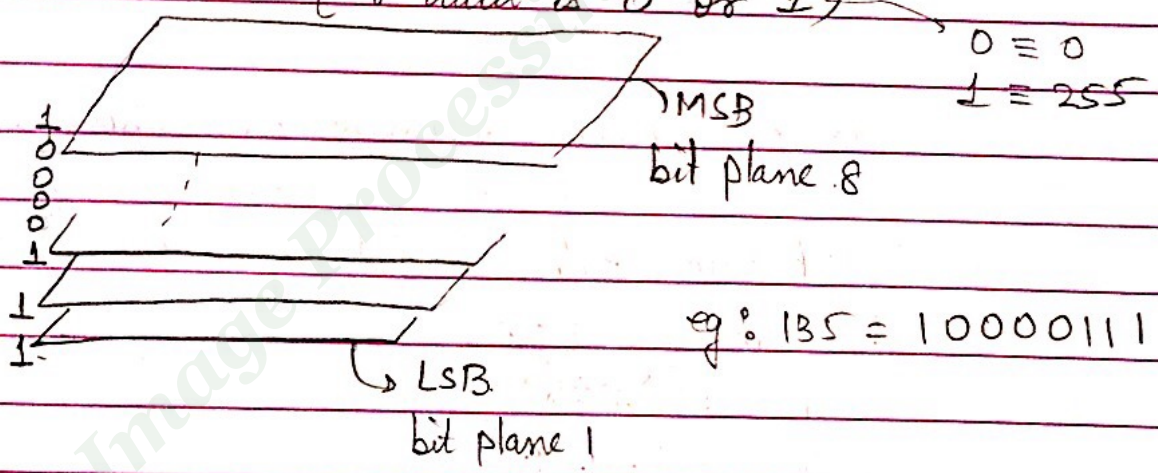
✡ Bit Plane Slicing :

- ✓ Each pixel = digital no. ⇒ has bits
- ✓ For 256 level grayscale image → 8 bits for each pixel
- ✓ we can highlight contribution of these bits to total image appearance.

Idea :- For a 256 level → It has 8 bits
Similarly, every pixel has 8 bits
So, we can make a plane of bits (all LSB's in one plane, — to all MSB's in one plane)
So, total 8 planes are possible.

So, each plane is a **Binary Image** (∵ data is 0 or 1)

MATLAB Assignment



Now, we find, by observation, the details of the image mainly depend on MSB plane; lower level planes have less contribution. So, LSB plane nearly has (no) contribution. So, LSB plane can be used to put any value → STEGNOGRAPHY (Data Hiding)

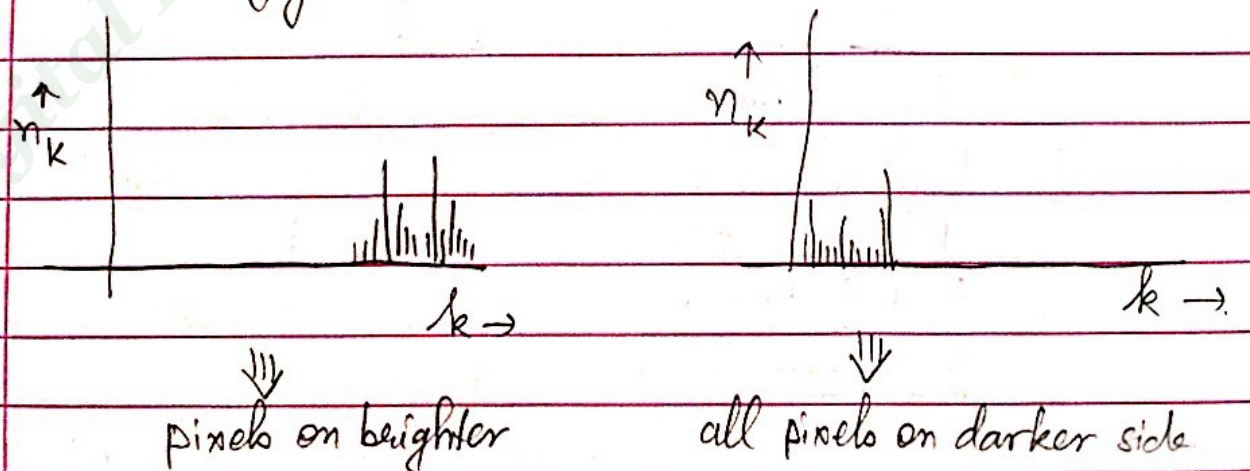
* Histogram Processing

- Let the intensity level in the image lie in the range [0 L-1] (for grayscale → [0 255])
- Histogram is a discrete fn, $h(r_k) = n_k$; r_k is the k^{th} intensity value and n_k is no. of pixels in the image with pixel level r_k.
- This histogram is normalized by dividing each component by total no. of pixels in the image. Normalised histogram is given by:

$$p(r_k) = \frac{n_k}{MN} \quad ; \quad k = 0, 1, 2, \ldots, L-1$$

↳ $p(r_k)$ is an estimate of probability of occurence of intensity level r_k in an image (Sum of all components = 1)

eg :-
```
[filename, pathname] = uigetfile('*.tif');
im = imread([pathname filename]);
imshow(im);
figure, imhist(im);
```



pixels on brighter side

all pixels on darker side

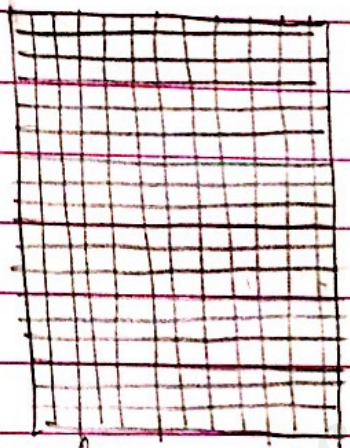
★ Histogram equilization

→ Let us denote $r \in [0\ L-1]$ as intensities of the image to be processed. $r=0$ corresponding to black and $r=L-1$ representing white.

Conditions (P.T.O)

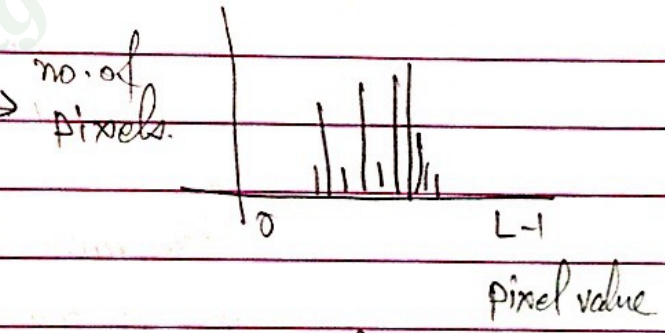
Let intensity transform^n is defined by $s=T(r)$ where $0 \leq r \leq L-1$ (& $0 \leq s \leq L-1$)

- $T(r)$ is monotonically ↑ fn in $0 \leq r \leq L-1$
- $0 \leq T(r) \leq L-1$ & $0 \leq r \leq L-1$



grayscale M×N

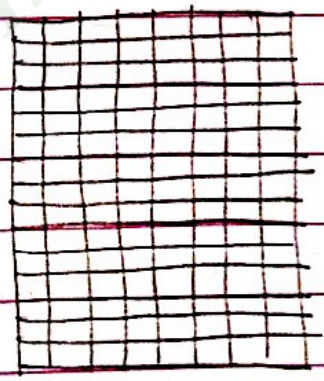
Consider an image $f(x,y)$ (or r) & its histogram, as shown.



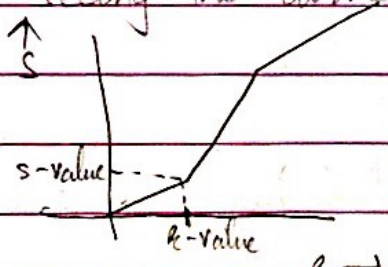
Suppose we want to improve contrast of this image.

M1) Contrast stretching

i.e, make the image $r \longrightarrow s$ by seeing the curve.



M×N



Method:

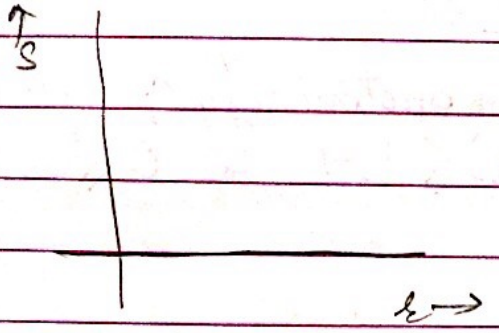
take one pixel's value (r-value). Go to the curve & find s-value for that. Replace at same place. We get contrast stretched image

✓ Time consuming method.

M2) Histogram Equaliz^{n}.

Instead of replacing every pixel, just use the histogram for contrast stretching.

Consider an image s.t pixel values range from 0 to L-1. (r = 0 ∴ black & r = L-1 ∴ white)



Intensity transform^{n}

$s = T(r)$

→ monotonically increasing

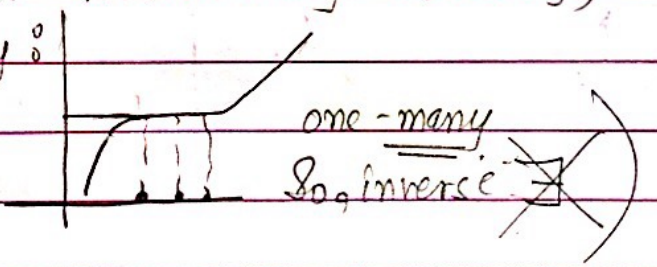
→ T(r) & r both lie b/w [0 L-1]

Suppose we write $r = T^{-1}(s)$

↳ only possible if T(r) is strictly monotonically ↑

(∵ f^{n} should be one-one for inverse to exist)

(If not monotonically ↑, strictly,



Now, consider intensity levels in image as random variables in interval 0 to L-1

We Define : Probability distrib^{n} f^{n} (PDF) as $P_r(r)$ & $P_s(s)$ for r & s respectively. (differentiable)

* If $P_r(r)$ & T(r) is known, T(r) is (cts) & (dfb), (continuous) over PDF range, then,

$$P_s(s) = P_r(r)\left|\frac{dr}{ds}\right| \rightarrow (1)$$

* Transform^n f^n is of the form :-

$$s = T(k) = (L-1) \int_0^k P_k(w)\, dw$$

$P_k(w)$ → Cumulative distrib^n f^n of random variable r.

We know

$$s = T(k)$$

Differentiating

$$\Rightarrow \frac{ds}{dk} = \frac{d}{dk}(T(k)) \quad \cdots (1)$$

Now, $T(k) = L-1 \int_0^k P_k(w)\, dw$.

$$\Rightarrow \frac{ds}{dk} = \frac{d}{dk}\left((L-1)\int_0^k P_k(w)\, dw\right)$$

$$\Rightarrow \frac{ds}{dk} = (L-1)\, P_k(k) \rightarrow (2)$$

Using (2) in (1).

$$\Rightarrow P_s(s) = P_k(k) \left[\frac{1}{(L-1)P_k(k)}\right]$$

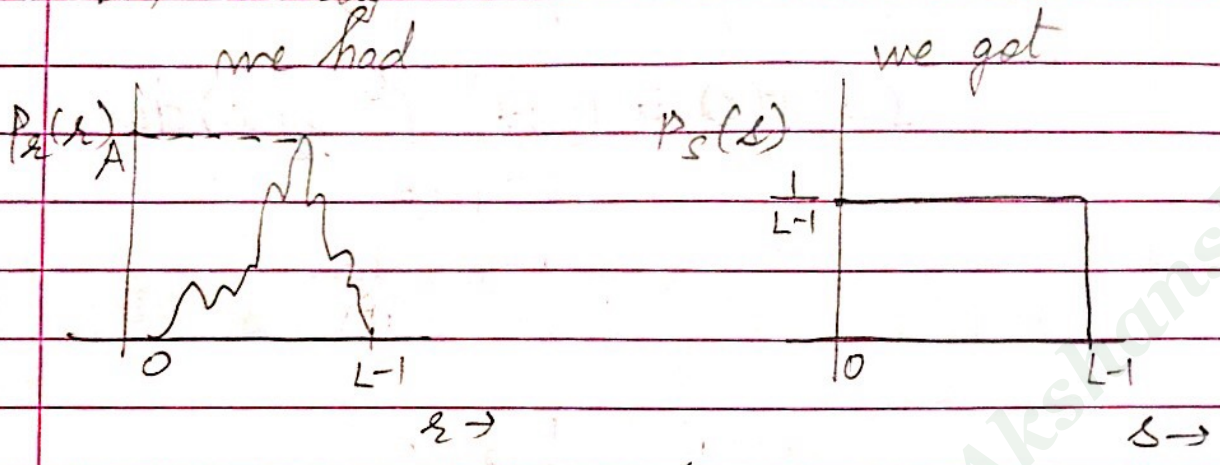
$$\Rightarrow P_s(s) = \frac{1}{L-1} \; ; \; 0 \leq s \leq L-1$$

↳ $P_k(k)$: distrib^n of i/p is sth else

But, at o/p, we got uniform distrib^n. i.e, uniformly distributed

∴, contrast stretching ✓

So, basically.



PDF : uniform (now)

eg : Continuous case

Consider :

$$P_r(r) = \begin{cases} \frac{2r}{(L-1)^2} & ; 0 \le r \le (L-1) \\ 0 & ; \text{otherwise} \end{cases}$$

We know $s = T(r)$

So, find $T(r) = (L-1)\int_0^r P_r(w)\,dw$

$$= (L-1)\int_0^r \frac{2w}{(L-1)^2}\,dw$$

$$= \left(\frac{1}{L-1}\right)\left(w^2\right)_0^r$$

$$\Rightarrow T(r) = \frac{r^2}{L-1}$$

↳ Suppose $L = 9$ (instead of 255)

So, pixel values vary from 0 to 8.

Suppose we take a pos^n (x, y) where $r = 3$

$$\therefore s = \frac{9}{9-1} = \frac{9}{8} \simeq 1$$

Now, finding $P_s(s)$

$$\Rightarrow P_s(s) = P_r(r)\left|\frac{dr}{ds}\right|$$

$$= \frac{2r}{(L-1)^2} \times \left|\left[\frac{ds}{dr}\right]^{-1}\right|$$

$$= \frac{2r}{(L-1)^2}\left|\left[\frac{d}{dr}\frac{r^2}{(L-1)}\right]^{-1}\right|$$

$$= \frac{2r}{(L-1)^2} \times \frac{L-1}{2r} = \frac{1}{L-1}$$

↳ uniform PDF

eg: Discrete case :-

Consider :

Discrete PDF : $P(r_k) = \frac{n_k}{MN}$, $k = 0, 1, 2, \ldots L-1$

(n_k → no. of pixels with pixel value = k; MN → Total pixel no.)

Then,

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_r(r_j)$$

$$\Rightarrow s_k = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \; ; \; k = 0, 1, \ldots L-1$$

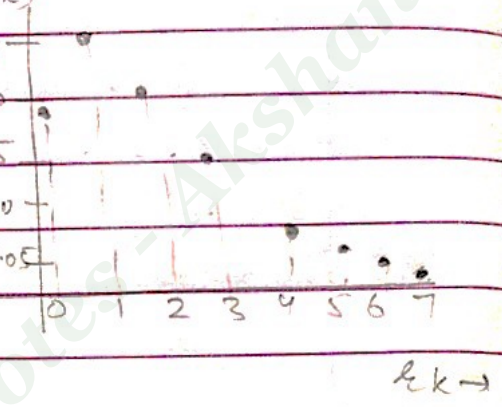
s_k → o/p

example (A) (used later)

Let us consider a 3 bit image ($L=8$) of 64×64 ($MN = 4096$) has intensity distribution shown below.

(MN → total no. of pixels)

r_k	n_k (no. of pixels having value r_k)	$P_r(r_k) = \frac{n_k}{MN}$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02



Now, finding each s_k

$$s_0 = T(r_0) = 7 \sum_{j=0}^{0} P_r(r_j) = 7 P_r(r_0) = 1.33$$

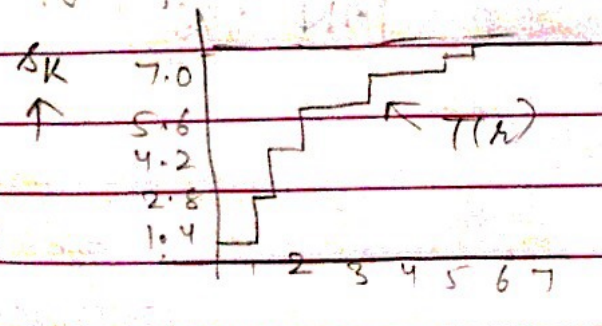
$$s_1 = T(r_1) = 7 \sum_{j=0}^{1} P_r(r_j) = 7 P_r(r_0) + 7 P_r(r_1)$$

$$= 1.33 + 7(0.25)$$

$$= 1.33 + 1.75$$

$$\Rightarrow s_1 = 3.08$$

Similarly, find $s_2, s_3, s_4 \dots, s_7$



Forming table :-

			Rounding up (i.e $\geq 0.5 \rightarrow 1$)
S_0	1.33	≡	1
S_1	3.08		3
S_2	4.5		5
S_3	5.67		6
S_4	6.23		6
S_5	6.65		7
S_6	6.86		7
S_7	7		7

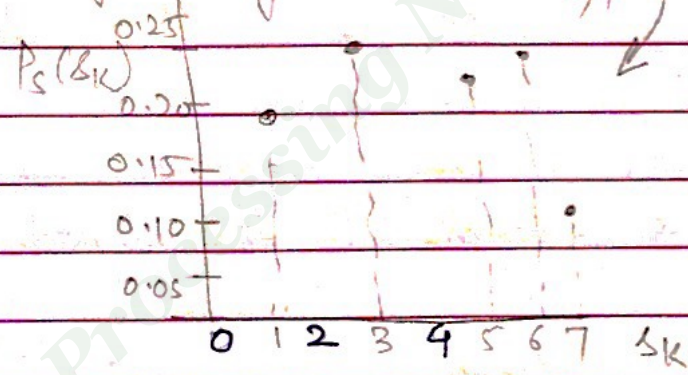
(You may use Round down i.e $\geq 0.5 \rightarrow 0$)

$\exists$ no 0, 2, 4 in o/p

S_0 pixel value with $V_p = 0$ is mapped to 1.

⇒ pixel with value 0 in i/p will get value 1 at o/p.

Making histogram :-



we see 3 → value 3 in table. This 3 is corresponding to (S1) S_0. See $n_k|_{k=1}$

S_0 at o/p :-

pixel with value 0 → $S_0 = 0$

1 → $S_1 = 790$ (k_0, $n_k = 790 \rightarrow S_1$)

2 → $S_2 = 0$ (∵ it becomes 5 in o/p)

$S_3 = 1023$

1023 pixels with value 1 is becoming S_3 in o/p image

$S_4 = 0$

$S_5 = 850$

no. of pixels having value 2 in i/p becomes 5

$S_6 = k_3$'s n_k value + k_4's n_k value

$= 656 + 329 = 985$

$S_7 = 245 + 122 + 81 = 448$

$n_k|_{k=5}$ $n_k|_{k=6}$ $n_k|_{k=7}$

eg

16×16

8	9	10
10	10	12
8	8	8

Apply histogram equaliz^{r} on the given 4 bit image.

⇒ 0 to 15 values of the pixels can exist

S1) Make Histogram :-

i/p : Pixel Values	No. of pixels.
r_0	
r_1	
r_2	
r_3	
⋮	
r_{15}	
	256

may/may not be given in exam.

no. of pixels should add to 256. (16×16)

✓ Note no. of pixels for each value

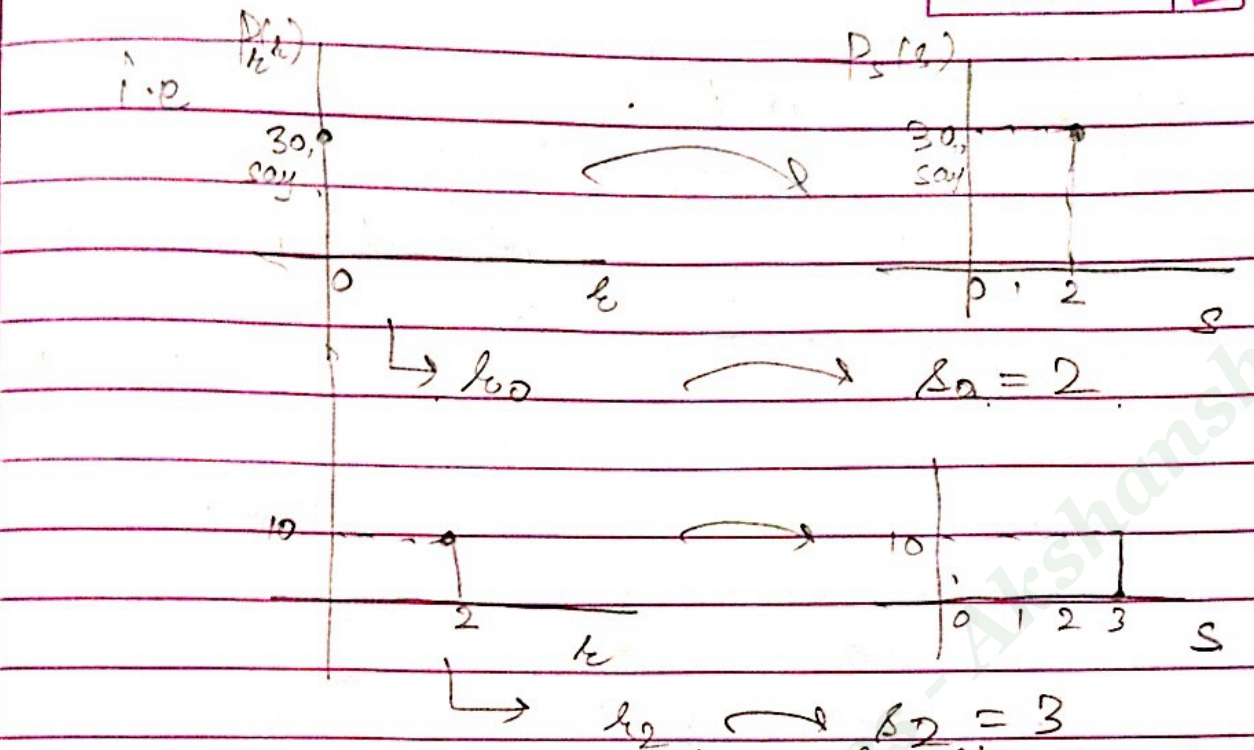
✓ Plot $P_r(r)$

✓ Find $s_k = (L-1)\sum_{j=1}^{k} P_r(r_j)$

Now, with these values of s, map with values of r.

i.e, eg. $s_0 \longrightarrow 2$ (value)

⇒ What i/p was r_0 it is becoming 2 at o/p



i.e, we take the no. of pixels for k_2 & write it for s_2.

✱ Histogram Specification / Histogram matching

→ Method in which we improve contrast of an image according to a reference image that we have.

→ In above method, we just apply the method. It blindly changes the contrast. We don't have control over how to make it better. Now, if we have histogram of any good reference image, we can then change the image w.r.t reference. Then, it'll be good.

(continued after following eg.)

Q) Consider a 8 bit image 4x4.

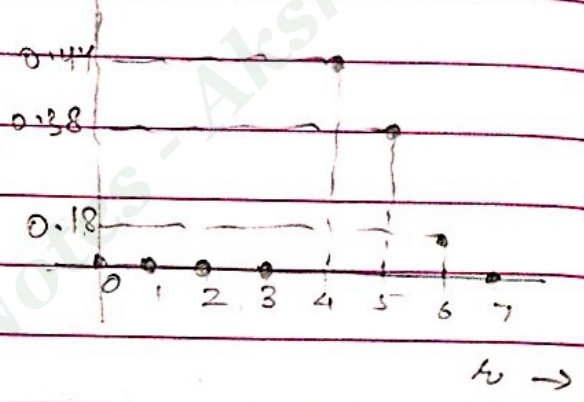
5	5	6	4
5	5	6	4
4	5	6	4
4	4	4	5

From observation, it is of poor contrast as all values in the box are b/w 4 & 6.

Now, making histogram :- $P_k(k)$

	No. of pixels	$P_k(k)$
k_0	0	0
k_1	0	0
k_2	0	0
k_3	0	0
k_4	7	$7/16 = 0.44$
k_5	6	$6/16 = 0.38$
k_6	3	$3/16 = 0.18$
k_7	0	0
	16	1

ensure total = 1. (Truncate it otherwise)



$\rightarrow 4 \times 4$ ✓

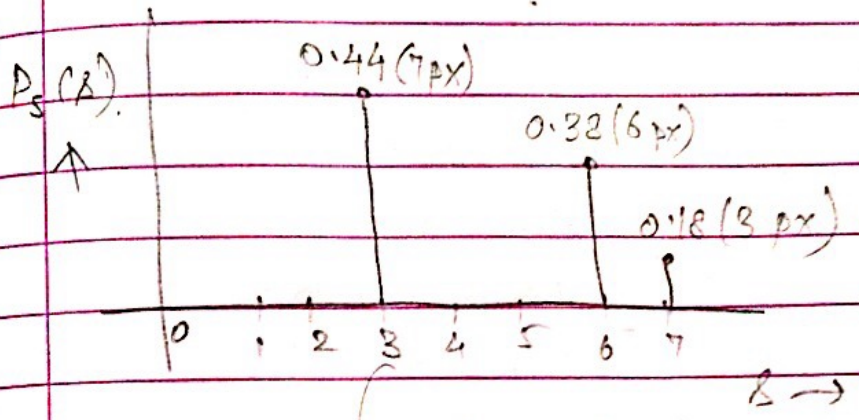
Now, finding

$$S_k = (L-1)\sum_{j=1}^{k} P_k(k_j)$$

S_0	0	$= 0$	$\simeq 0$
S_1	0	$= 0+0$	$\simeq 0$
S_2	0	$= 0+0$	$\simeq 0$
S_3	0	$= 0+0$	$\simeq 0$
S_4	$(8-1)(0.44)$	$= 3.08 + 0$	$\simeq 3$
S_5	$(8-1)(0.38)$	$= 2.66 + 3.08$	$\simeq 6$
S_6	$(8-1)(0.18)$	$= 1.26 + 5.74$	$\simeq 7$
S_7	0	$= 0 + 7 = 7$	$\simeq 7$

Page No

Histogram of o/p



→ Value of k_4 (s_4) becomes 3 finally.
Now, so, we plot at s = 3, value 0.44
(value of k_4, having 7 pixels)

Now,

o/p image:
All 4's is becoming 3
5 is becoming 6
6 is becoming 7.

Do this in the image:

6	6	7	3
6	6	7	3
3	6	7	3
3	3	3	6

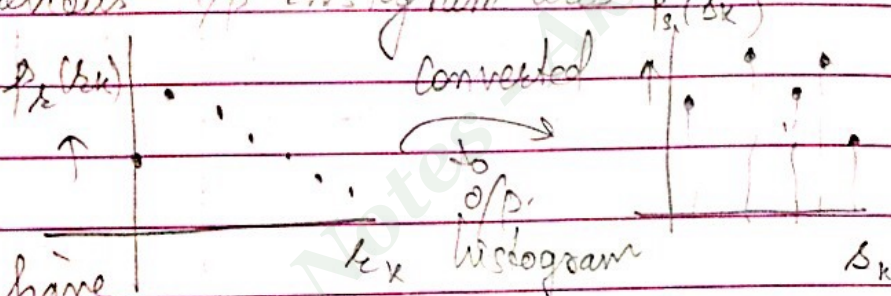
Now, using Histogram Specification.

from example (A) values :- → from histogram

S1) find S_k

$$S_k = (L-1)\sum_{j=0}^{k} P_r(r_j) \; ; \; k = 0, 1, 2, \ldots 7$$

Now, our previous i/p histogram was



Now, we have got the o/p histogram.

Q. Use this histogram to make it to a reference histogram.

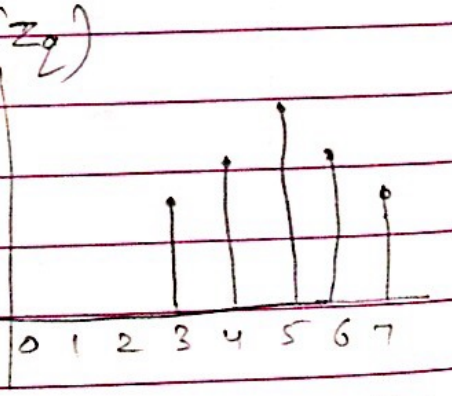
Given :-

Specified histogram :-

(Reference)

z_q	$P_z(z_q)$
$z_0 = 0$	0.0
$z_1 = 1$	0.0
$z_2 = 2$	0.0
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15

&

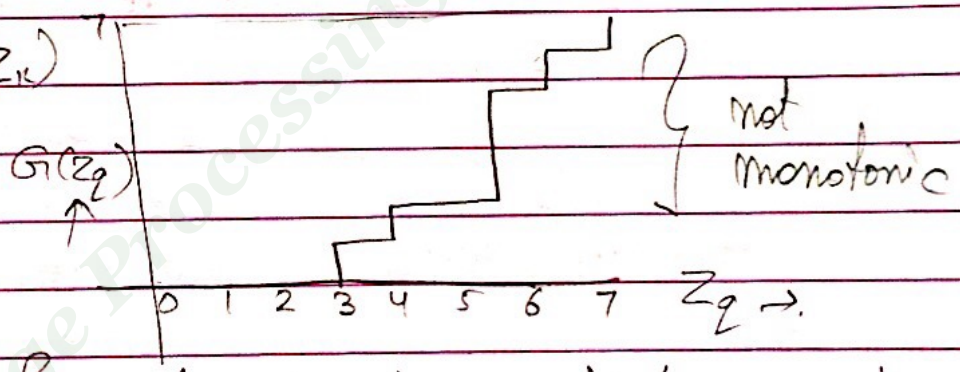


S2) Find $G(Z_k)$

$$G(Z_k) = (L-1) \sum_{j=0}^{k} P_z(Z_j)$$

$G(Z_0)$	0	= 0
$G(Z_1)$	0	≃ 0
$G(Z_2)$	0	≃ 0
$G(Z_3)$	1.05	≃ 1
$G(Z_4)$	= 2.45	≃ 2
$G(Z_5)$	4.55	≃ 5
$G(Z_6)$	5.95	= 6
$G(Z_7)$	7	= 7

S3) Plot $G(Z_k)$



Now, we have to convert s-histogram to this reference.

∴ Non-monotonic

So, Procedure:

Find smallest value of Z_q s.t $G(Z_q)$ is closest to S_k.

eg: $S_0 = 1$ & $G(Z_3) = 1$ → a perfect match

Here, $S_0 \rightarrow Z_3$ i.e every pixel whose value is 1 in histogram equalised image is mapped to pixel valued 3 in histogram specified image.

Now, we do $S_k \xrightarrow{\text{mapped to}} G(Z_k)$

S_0	1	$G(Z_0)$	0
S_1	3	$G(Z_1)$	0
S_2	5	$G(Z_2)$	0
S_3	6	$G(Z_3)$	1.05
S_4	6	$G(Z_4)$	2.45
S_5	7	$G(Z_5)$	4.50
S_6	7	$G(Z_6)$	5.95
S_7	7	$G(Z_7)$	7.

1 is close to $G(Z_3) = 1.05$

So, 1 is mapped to 3 $\Rightarrow 1 \to 3$

3 is close to 2.45 ($G(Z_4)$)

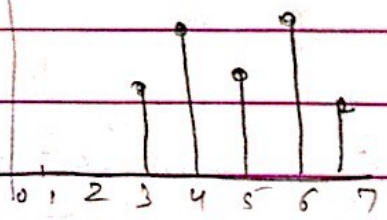
So, $3 \to 4$

Also, $5 \to 5$

$6 \to 6$

$7 \to 7$

So, this gives histogram as:-



matches the given reference histogram (closely).

790

($k_0 \to k_1 \to Z_3$)

So, o/p,

$Z_3 = 790$

$Z_4 = 1023$

$Z_5 = 850$

$Z_6 = 985$

$Z_7 = 448$

CLASSMATE

→ Mapping explained in detail.

Eg: Consider a 3 bit image :-

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

Given an image (3 bit) & values of r vary from 0 to 7. Hence, for equalising it we have to find S_K values & then round them off.

S1) Equalise it

	no. of steps	$P_r(r_K)$	S_K	Rounded off
$r_0 = 0$	1	0.015	7(0.015) = 0.105	= 0
$r_1 = 1$	2	0.03	0.315 = 0.3	= 0
$r_2 = 2$	1	0.015	0.42	= 0
$r_3 = 3$	2	0.03	0.63	= 1
$r_4 = 4$	16	0.25	2.38	= 2
$r_5 = 5$	19	0.29	4.41	= 4
$r_6 = 6$	23	0.37	7	= 7
$r_7 = 7$	0	0	7	= 7
	64 → 8×8 ✓	$P_{Total} = 1$	Total should be 7.	Rounding off the values.

$$S_K = (L-1)\sum_{j=0}^{k} P_r(r_j) = 7[P_r(r_K)]$$

$S_0 = 7P_r(r_0) = 7(0.015)$

$S_1 = 7[P_r(r_0) + P_r(r_1)] = 7[0.015 + 0.03] = 0.315$

& so on ~ ~ ~

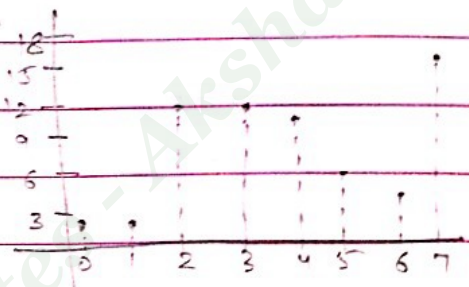
Page No

S2) Find values for given rose :

Reference :-

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

→ Histogram of Reference



	No. of pixels	$P_z(z_q)$ → $\frac{2}{50}$	$G(z_k)$	Rounding off
$z_0 = 0$	2	0.03	$7(0.03) = 0.21$	0
$z_1 = 1$	2	0.03	$0.42 = (0.21 + 7(0.03))$	0
$z_2 = 2$	12	0.18	$1.68 = 0.42 + 7(0.18)$	2
$z_3 = 3$	13	0.2	$3.08 = 1.68 + 7(0.2)$	3
$z_4 = 4$	10	0.15	4.13	4
$z_5 = 5$	6	0.1	4.83	5
$z_6 = 6$	4	0.06	5.25	5
$z_7 = 7$	15	0.25	7	7

S3) Mapping (done in Part ① & Part ②)

Summary for Part ②

Idea: Make a table, containing of values of $s_0, s_1, s_2 \ldots$ & also of $G(z_1), G(z_2), G(z_3)$

Now, see the value of s_k that matches (or is close to) some $G(z_q)$'s value. Then, the value of s_k maps to q of $G(z_q)$

Table 1

r_k	s_k values	z_q	$G(z_q)$
r_0	0	z_0	0.21
r_1	0	z_1	0.42
r_2	0	z_2	1.68
r_3	1	z_3	3.08
r_4	2	z_4	4.13
r_5	4	z_5	4.83
r_6	7	z_6	5.25
r_7	7	z_7	7

Part (1) Part (2)

Seeing mapping from $r \rightarrow s$. Then from $s \rightarrow z$.

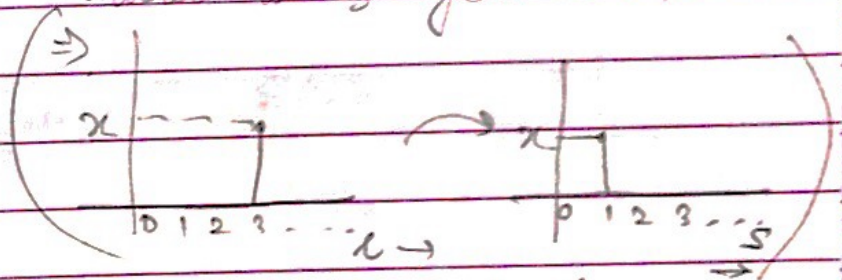
Part (1) → $s_3 \rightarrow 1$ ⇒ value of s has changed from 3 to 1

s_4 2 ⇒ r_3 becomes r_1.

s_5 4 r_0 in the histogram, no. of pixels

s_6 7 alloted to r_3 goes to s_1

s_7 7



Now, no. of pixels for $r_3 = 2$. r_0 at o/p ⇒ $s_1 = 2$

Illy, $r_4 \rightarrow s_2 = 16$.

$r_5 \rightarrow s_4 = 19$

$r_6 \rightarrow s_7$ ⇒ s_7 = r_6's pixels + r_7's pixels

$r_7 \rightarrow s_7$ = 23 + 0

⇒ $s_7 = 23$

$r_0 + r_1 + r_2$'s pixels = 1+1+2 = 4

r_0 after mapping $r \rightarrow s$, we have

$s_0 = 4$, $s_1 = 2$, $s_2 = 16$, $s_3 = 0$, $s_4 = 19$, $s_5 = 0$

$s_6 = 0$, $s_7 = 23$ → histogram can be made

Part (2) Next, mapping from s_k to z.

Idea: See s_k value column & see $G(z_q)$ column. The value of s_k close to value of $G(z_q)$ gets mapped.

After mapping, the "q" in z_q is given the no. of pixels, which were there with s_k value.

From Table 1 from previous page,

$s_3 = 1$ is close to $z_2 = 1.68$

So, $1 \longrightarrow 1.68$ mapped.

& q in z_q is 2 in z_2. So, z_2 is given no. of pixels = 2 (which were there with s_1)

Now

$s_4 = 2$ is close to $z_3 = 3.08$

So,

$2 \longrightarrow 3.08$.

& no. of pixels of z_3

= no. of pixels of s_2

$\Rightarrow$ no. of pixels of $z_3 = 16$.

s_1 (not s_3) { ∵ s_3 changes to s_1 }

Also, we map

$s_0 \longrightarrow z_0$,

(0 close to 0.2)

So, no. of pixels of $z_0 = 4$

Similarly, $4 \longrightarrow 4.13$

& $z_4 = 19$

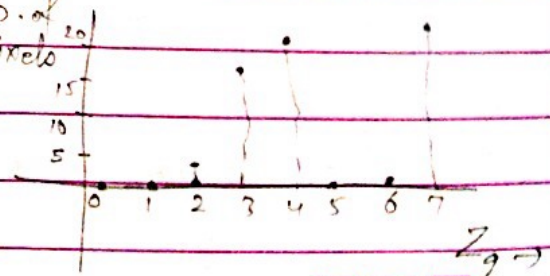
$7 \longrightarrow 7$

of s_6 & s_7

& z_7 = no. of pixels of $s_7 + s_6 = 0 + 23 = 23$

So, finally, we have

$z_0 = 4$	$z_4 = 19$
$z_1 = 0$	$z_5 = 0$
$z_2 = 2$	$z_6 = 0$
$z_3 = 16$	$z_7 = 23$

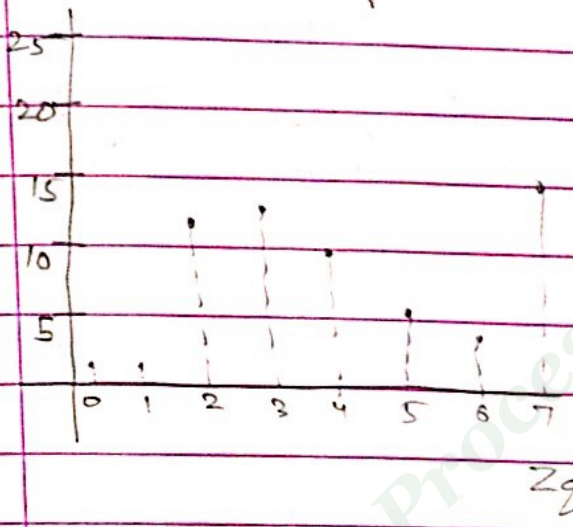


What was the idea behind doing this?

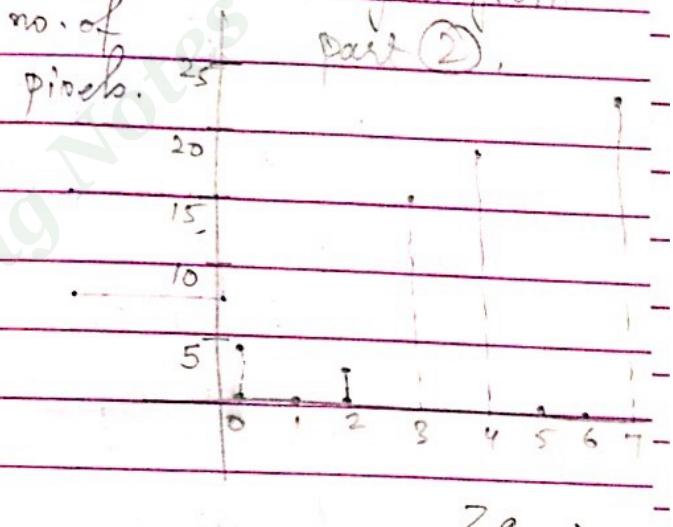
↳ The histogram of reference image given should match (or be close to) the histogram (of the final image) got in part (2). This will indicate that they are of similar contrasts.

It can be seen:

Reference Histogram



Histogram from part (2).



(sort of similar)

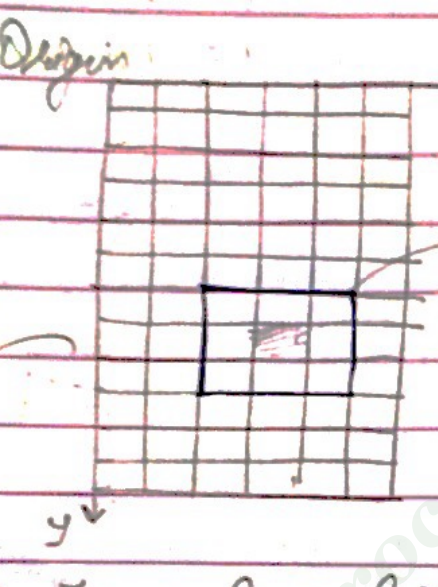
Ans=

V. Imp ✡ Note:

The total no. of pixels in all the domains (be it r, s or z) remains the same.

★ NEIGHBOURHOOD OPERATION

- operates on a larger neighbourhood of pixels than point oper^{n}.
- Neighbourhoods are mostly a rectangle around a central pixel.
- Any size rectangle and any shape filter are possible.



Considering a 3×3 neighbour for (x,y). Taking average oper^{n} (say), so, do avg. of all neighbour value & replacing it with (x,y).

For each pixel in original image, outcome is written at same loc^{n} at target image.

for corner pixels, pad with 0's or same value of rows & columns in the corner

- There can be many oper^{ns} possible
  eg: average, minimum, maximum ... etc.
  - ↳ Set pixel value which is min. in neighbourhood

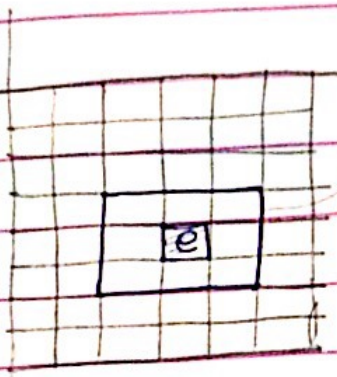
★ For corner pixels (edges)
- ✓ Omit missing pixels
- ✓ Pad the image
- Replicate borders
- Truncate the image.

If mask is of size 3×3, the neighbourhood will be 3×3.

SPATIAL FILTERING PROCESS:

For image processing applic", fillers are given in terms of MASK.

Now, how to filter image using filter mask?



→ considering a 3×3 neighbourhood, we do CONVOLUTION

a	b	c
d	e	f
g	h	i

*

j	k	l
m	n	o
p	q	r

Original image pixel — Filter (w)

Idea: Overlap original image with filter.
Multiply coinciding elements & add.
Replace the got value with pixel e.

fo,

$$e_{processed} = a*j + b*k + c*l + d*m + e*n + f*o + g*p + h*q + i*r$$

eg Consider Original image: & filter mask:

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

0	1	0
1	-4	1
0	1	0

Find: Revised image using oper^n:
(a) Min.
(b) Max
(c) Average
(d) Filter mask

(a) For min oper^n,

0	0	0		
0	[2]	4	6	7
0	0	2	3	4
	0	1	1	1
	2	3	4	4

If padding done with 0
min (0, 2, 4) = 0
So, 1stly all values become 0.
Hence, instead do padding with same elements

2	2	4	6	7	7
2	[2]	4	6	7	7
0	0	2	3	4	4
0	0	1	1	1	1
2	2	3	4	4	4
2	2	3	4	4	4

for first element,

2	2	4
2	[2]	4
0	0	2

min = 0

1stly do ✓

(b) Max. oper^n. (using padding = 0)

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

processed image (taking max value for 3×3 neighbour)

4	6	7	7
4	6	7	7
3	4	4	4
3	4	4	4

(c) Average (padding with 0, say)

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

first element.

$2 \xrightarrow{\text{changes to}} \frac{2+4+2}{9}$

2nd element

$4 \rightarrow \frac{4+2+6+2+3}{9}$.

⋮

& so on.

(d) Weiner mask (padding using same element, say)

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

for first element.

$$\begin{bmatrix} 2 & 2 & 4 \\ 2 & 2 & 4 \\ 0 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$\Rightarrow$ 2 | processed

$= (2\times0)+(2\times1)+(4\times0)$
$+(2\times1)+(2\times-4)+(4\times1)$
$+(0\times0)+(0\times1)+(2\times0)$

Similarly change value of every pixel.

* Filters like min., max., don't require any masking.

* Spatial filtering : Equation form

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} g(s,t) f(x+s, y+t)$$

* Smoothing Spatial Filters
  - one of simplest spatial filtering operns
  - Simply averages all pixels in a neighbourhood around a central value.
  - Useful for removing noise from images
  - highlights gross details

Averaging filter :

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

eg :- Suppose we had.
3 bits ⇒ 0 to 7

0	1	1
0	7	1
0	0	0

→ In a dark background, 7 is noise.

Now, use filter.

$$= \frac{10}{9} = 1.2 \simeq 1.$$

So, $7 \rightarrow 1$.

So, noise removed.

* Idea:
Class with students not in uniform : high freq
Class with students wearing uniform : low freq

* When we ↑ size of neighbourhood, we get a more grossed image (having less detail & blurred)

* Weighted Smoothing filters:-
↳ using weighted filter:

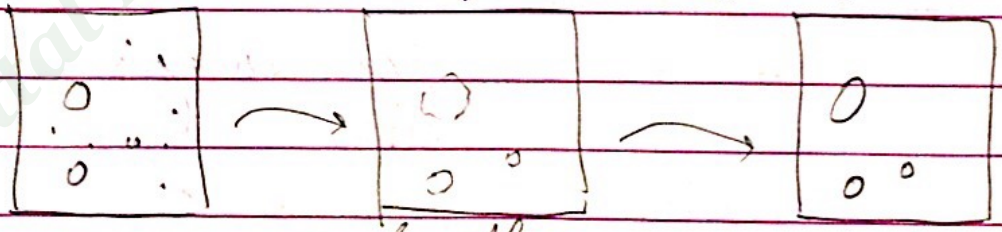
1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

→ Basically distribute elements' weight s.t. Total = 1

* Pixels closer to central pixel are more important.

* Basically we are doing convolution oper", only using diff+ filters

* Thresholding ⇒ getting a binary image.
I want to remove dots



Smoothing (gives blurred image)

Thresholding (B & W image (dots removed))

• Median filter: Take nxn neighbourhood of an image's pixel. Arrange in ascending/descending order (the pixel values). Take the middle value for that pixel.

- Smoothing filter : Low pass filter
- Sharpening filter : High pass filter

CLASSMATE

* Sometimes median filter works better than averaging filter.

☆ CORRELATION & CONVOLUTION

Same oper^n.

Difference : → dont flip
→ we flip filter mask.

→ Convolution = Correlation if filter mask is symmetric.

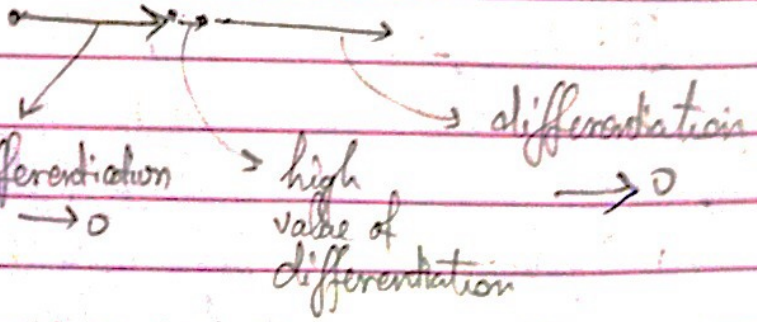
☆ Sharpening filter : (based on spatial differentiation)

Used to highlight fine details.
↳ remove blurring from images
↳ highlight edges

Note : Edge is always sharp ∵ it highlights boundary b/w 2 areas/images

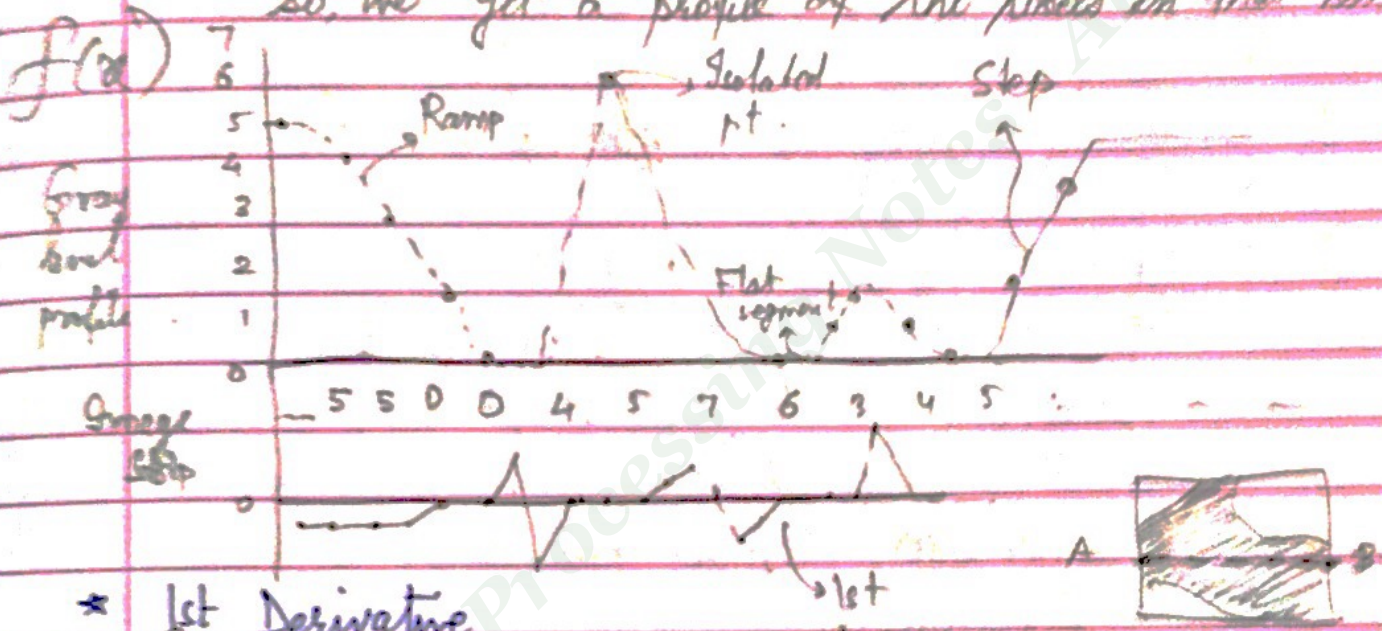
Spatial differentiation ⇒ I differentiate i.e I see variation in colors (pixels)

When I go from



* Differential oper" used for sharpening.
* Differentiation: measure of rate of change of a fn w.r.t a variable

Idea:- Suppose we have any image. Consider one line (one row in it) as 1D image.
So we get a profile of the pixels in that line



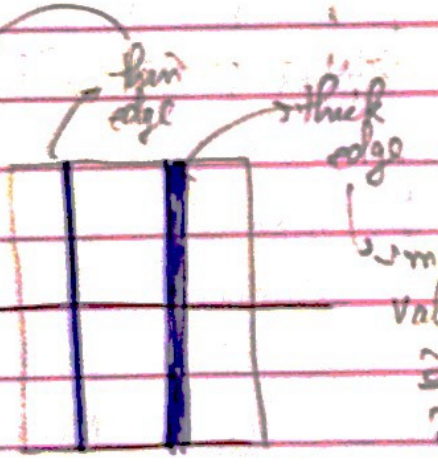
* 1st Derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

↳ diff. b/w subsequent values

- the info about existence of edge

We find :- small change : derivative value is less,
large change : derivative value is high.



* 2nd Derivative :-

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

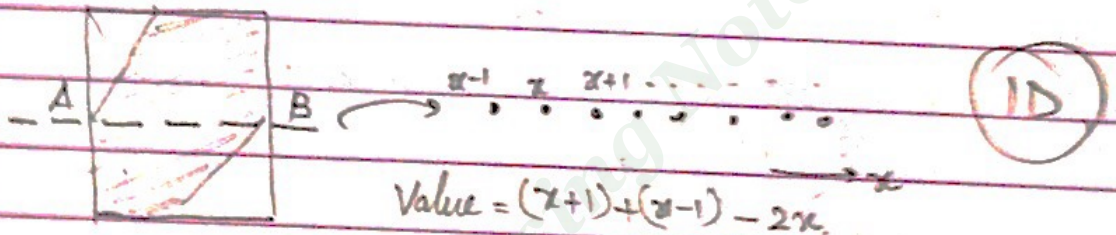
→ takes into account both before & after the current value

$$\equiv [f(x+1) - f(x)] + [f(x-1) - f(x)]$$

after — before

tells dirⁿ of edge (like black to white or white to black)

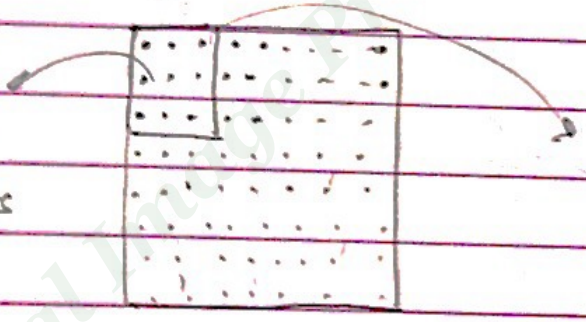
Suppose the profile of one line :-



Value = $(x+1)+(x-1) - 2x$.

Now, for (2D) include both x & y axis

Taking 3×3 neighbours



$(x-1, y-1)$	$(x-1, y)$	$(x-1, y+1)$
$(x, y-1)$	(x, y)	$(x, y+1)$
$(x+1, y-1)$	$(x+1, y)$	$(x+1, y+1)$

Now, to see if (x, y) is an edge pixel, & apply 1st order derivative in all 4 directions ($\updownarrow, \leftrightarrow, \nearrow, \searrow$)

Then, we add all 4 values & put in (x, y).

↕ : x dirⁿ

Applying $\frac{\partial^2 f}{\partial x^2} = [f(x+1, y) + f(x-1, y) - 2f(x, y)]$

$\frac{\partial^2 f}{\partial y^2} = [f(x, y-1) + f(x, y+1) - 2f(x, y)]$

↔ : y dirⁿ

$\frac{\partial^2 f}{\partial x \partial y} = [f(x-1, y+1) + f(x+1, y-1) - 2f(x, y)]$

↗↙ dirⁿ (+45°)

$\frac{\partial^2 f}{\partial y \partial x} = [f(x-1, y-1) + f(x+1, y+1) - 2f(x, y)]$

↘↖ dirⁿ (-45°)

Now, Total 2nd derivative = $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial x \partial y} + \frac{\partial^2 f}{\partial y \partial x}$

This value is set in (x, y) → Sharpened value.

* 2nd derivative: more useful for image enhancement (as compared to 1st derivative)
  - → gives stronger response to fine detail.
  - → Simpler implementation

☆ LAPLACIAN filter (A sharpening filter)
- Isotropic (same on all dirs)
- One of simplest sharpening filters.

3/3/14

CLASSMATE

Consider an 8x8 image, f & its filter mask, w (3x3)
Then, its convolved o/p image, $y = f * w$

↳ For convolving, as done before: Take coinciding elements

Idea: Make a mask s.t the values present in it, when multiplied with coinciding elements, gives the results of 2nd derivative. So, I will get a sharpened image.

↳ Such a filter mask is called OPERATOR

(recall: operator for filtering oper^n was: $\frac{1}{9}$ [3x3 matrix of 1's])

Operator for sharpening is called Laplacian Operator.

* Defin^n: Laplacian is defined as:-

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

From previous page,

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

& $$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

So, $$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Page No

CLASSMATE

Building filter mask :-

0	1	0
1	-4	1
0	1	0

≡

0	(x-1, y)	0
(x, y-1)	(x, y)	(x, y+1)
0	(x+1, y)	0

not considering diagonal dir^n

eg: Apply Laplacian operator on the image :

0	0	0	0	0
0	4	5	6	0
0	10	3	40	0
0	20	30	50	0
0	0	0	0	0

=

-16	?	?
?	-12	?
?	?	?

Now, put the above filter mask on this image & multiply coinciding elements

For 4 :-

0	0	0
0	(4)	5
0	10	3

×

0	1	0
1	-4	1
0	1	0

=

0	0	0
0	(-16)	5
0	10	0

→ So put -16 in place of 4.

For 3 :-

4	5	6
10	(3)	40
20	30	50

×

0	1	0
1	-4	1
0	1	0

=

0	5	0
10	(-12)	40
0	30	0

put -12 in place of 3

Ily do ∀ pixels.

- Applying Laplacian to an image, we get a new image that highlights edges & other discontinuities.

Note: Laplacian highlights the edges. But, the image is not very clear (not enhanced). Edges are not clear.

So, some enhancement can be done:
Subtract laplacian (result) from original image to generate final sharpened enhanced image.

So, $g(x,y) = f(x,y) - \nabla^2 f$

($g(x,y)$: sharpened image; $f(x,y)$: original image; $\nabla^2 f$: Laplacian)

So, we can make a single function for complete sharpened image:-

$$g(x,y) = f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)]$$

$$\Rightarrow g(x,y) = f(x+1,y) + f(x-1,y) + f(x,y-1) + f(x,y+1) + 5f(x,y)$$

0	-1	0
-1	5	-1
0	-1	0

(5 → $f(x,y)$; y →; x ↓)

Ily, ∃ different versions of Laplacian:

0	1	0
1	-4	1
0	1	0

Simple laplacian

1	1	1
1	-8	1
1	1	1

Variant of Laplacian (done by adding diagonal eqns, ⇒ in diagonal directions as well)

1	1	1
1	8	1
1	1	1

Now, making $g(x,y)$ from the variant of laplacian :-

$f(x,y) - \nabla^2 f =$

-1	-1	-1
-1	9	-1
-1	-1	-1

↳ gives details of diagonals as well ($+45°$, $-45°$ also)

Basically :

0	1	0
1	-4	1
0	1	0

Simple Laplacian.

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

→

1	1	1
1	-8	1
1	1	1

Variant of Laplacian

$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial x \partial y}\Big|_{45°} + \frac{\partial^2 f}{\partial y \partial x}\Big|_{-45°}$

↙ Including diagonals.

→

-1	-1	-1
-1	9	-1
-1	-1	-1

$g(x,y) = f(x,y) - \nabla^2 f$

↓ Variant

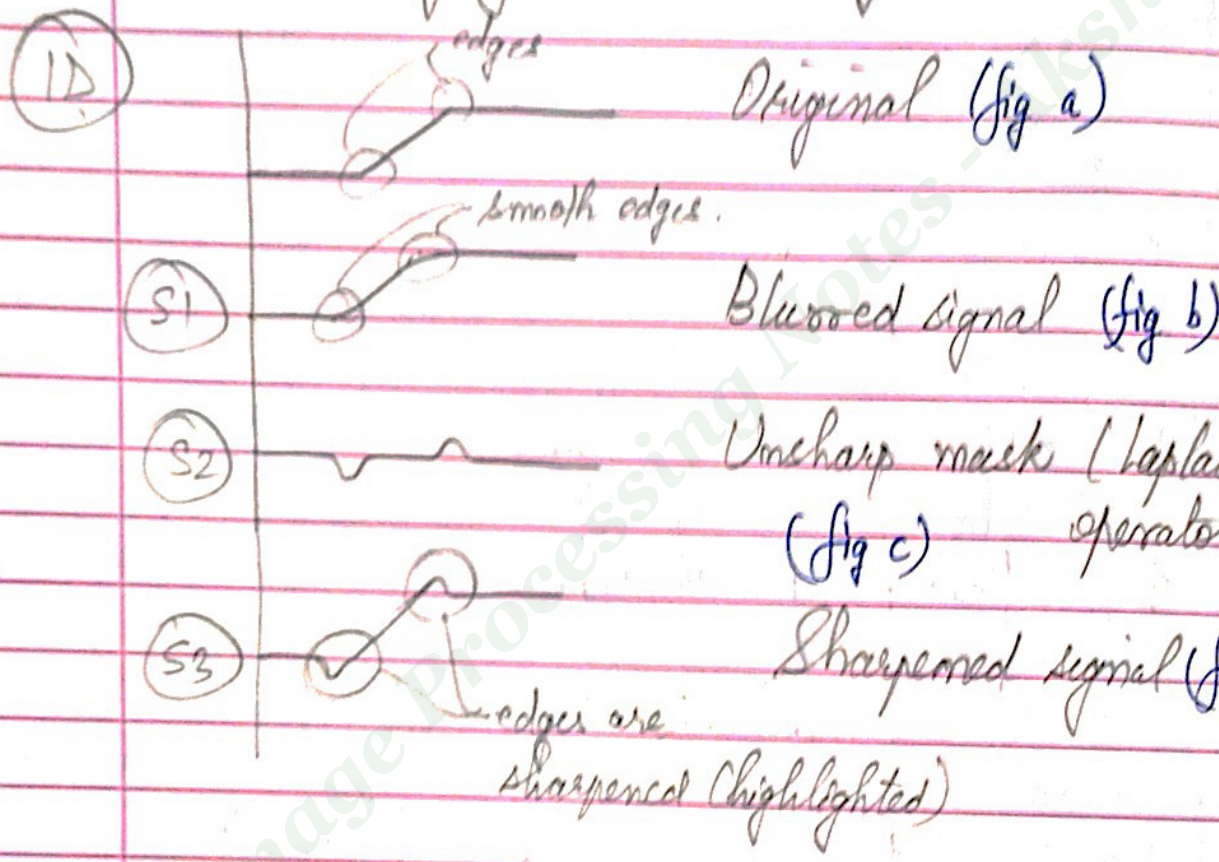
✱ Unsharp Mask & Highboost Filtering

Using sequence of linear spatial filters in order to get sharpening effect.

↓ like subtraction, averaging, sharpening

Steps: S1) ✓ Blur (averaging filter)
S2) ✓ Subtract from original image.
S3) ✓ Add resulting mask to original image.

* Highboost filtering
  ↳ trying to detect edge



* $fig(c) = fig(a) - fig(b)$
  $fig(d) = fig(a) + fig(c)$.

* 1st Derivative Filtering.
  ↳ Implementation: difficult

For a fn $f(x,y)$, the gradient of f at coordinates (x,y) is given as column vector &

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Gradient of f

Magnitude of vector :

$$\nabla f = mag(\nabla f)$$
$$= [G_x^2 + G_y^2]^{1/2}$$
$$\Rightarrow \nabla f = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

↳ for practical reasons :

magnitude

$$\nabla f \simeq |G_x| + |G_y|$$

Simple technique to perform derivatives :-

$$\nabla f \simeq \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| \rightarrow \frac{\partial f}{\partial x}$$
$$+ \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right| \rightarrow \frac{\partial f}{\partial y}$$

↳ Based on :

z_1	z_2	z_3	R_1
z_4	z_5	z_6	R_2
z_7	z_8	z_9	R_3
C_1	C_2	C_3	

$$\nabla f \simeq |R_3 - R_1| + |C_3 - C_1|$$

(with weights added to z_8, z_2, z_6, z_4)

Sobel operator mask will be :

$\frac{\partial f}{\partial x}$ →

-1	-2	-1
0	0	0
1	2	1

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

-1	0	1
-2	0	2
-1	0	1

above method referred to as : SOBEL OPERATION

eg. Given image. Find ∇f

0	0	0	0	
0	7	8	3	4
	2	3	8	8
	2	4	4	4
	2	3	5	4

Idea : To find ∇f, find G_x & G_y

G_x =

use mask =

-1	-2	-1
0	0	0
1	2	1

G_y =

use mask =

-1	0	1
-2	0	2
-1	0	1

PTO

0	0	0		-1	-2	-1
0	7	8	×	0	0	0
0	2	3		1	2	1

= 2×2 + 3×1 + 7×0 + 8×0 = 7

L_0 remains same.

L_0

0	0	0
0	7	8
0	2	3

→

0	0	0
0	7	8
0	2	3

First element of G_y

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 8 \\ 0 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = 8\times 2 + 3\times 1 = (19)$$

So,

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 8 \\ 0 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 19 & 8 \\ 0 & 2 & 3 \end{bmatrix}$$

Why do ∀ elements of G_x, G_y ...

Now,

$$\nabla f = \sqrt{G_x^2 + G_y^2}$$

or $|G_x| + |G_y|$

$$= \begin{bmatrix} 7 & 18 & ? & ? \\ ? & ? & & \\ & & & \\ & & & \end{bmatrix} + \begin{bmatrix} 19 & 14 & ? & ? \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

$$\Rightarrow \nabla f = \begin{bmatrix} 26 & 30 & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

} fill all elements.

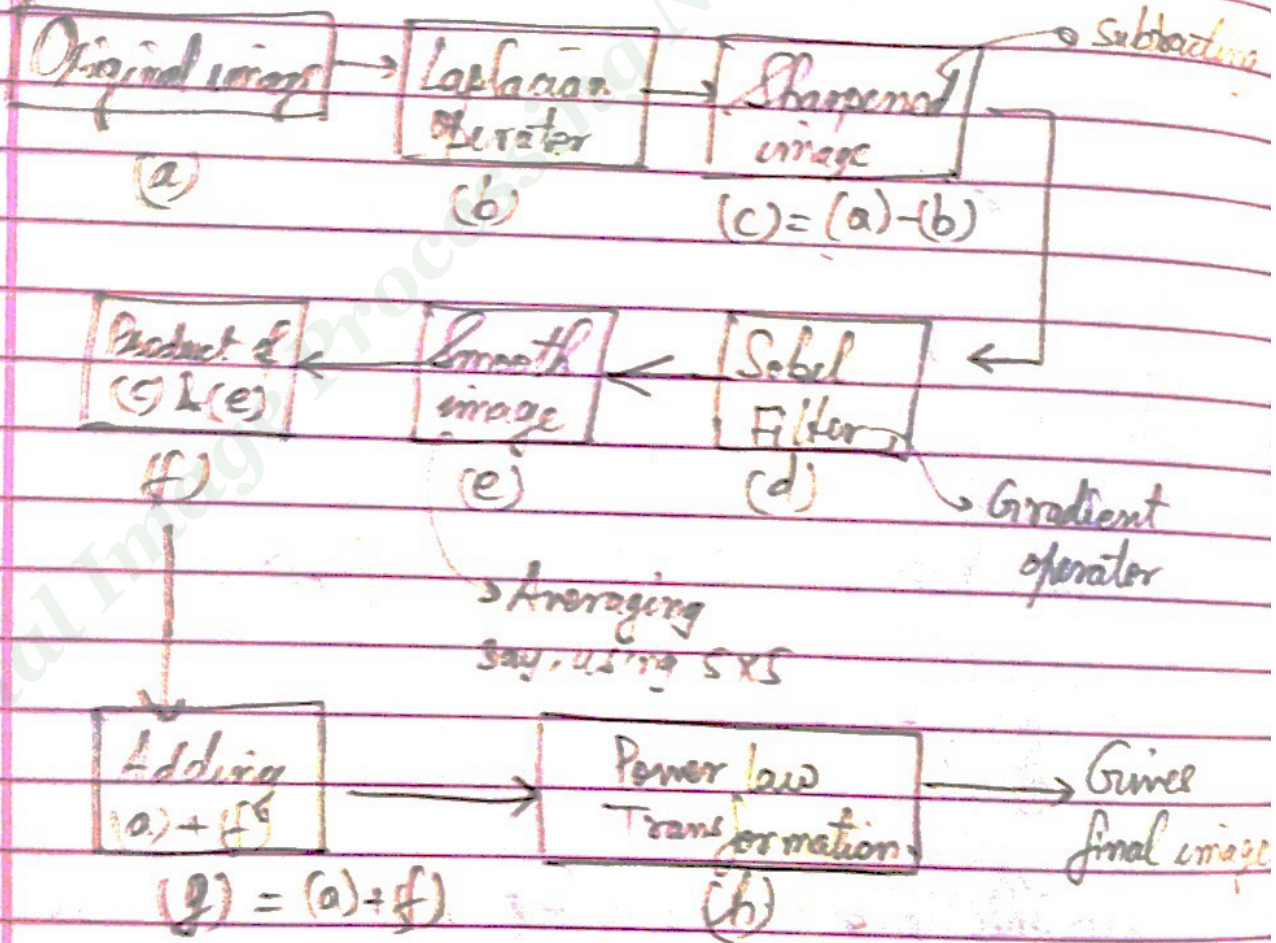
This is a SOBEL Operation.

1st Derivative	2nd Derivative
> Produces thicker edge	> Has thin lines
> Stronger response to gray level steps	> Has stronger response to fine details → produce double response at steps changes in gray level.

★ Combining Spatial Enhancement Methods

↳ Spatial domain processing

↳ operate directly on pixels



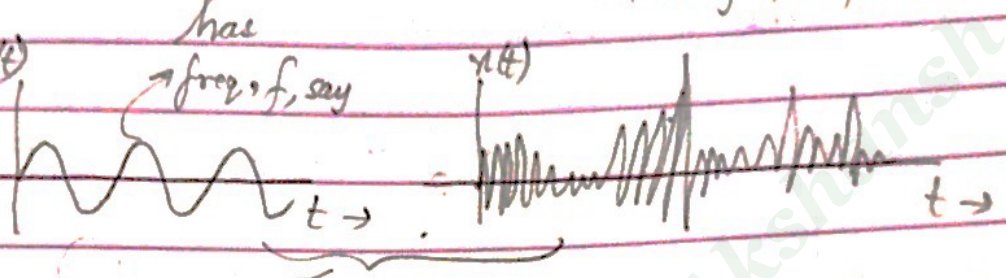
* above comes from trial and error (→ other enhancement methods also, not used here).

General nomenclature : $x(t)$: cts (continuous)
$x(n)$: discrete

CLASSMATE

IMAGE TRANSFORMS → Transforming special domain pixel values to some domain using existing transform methods

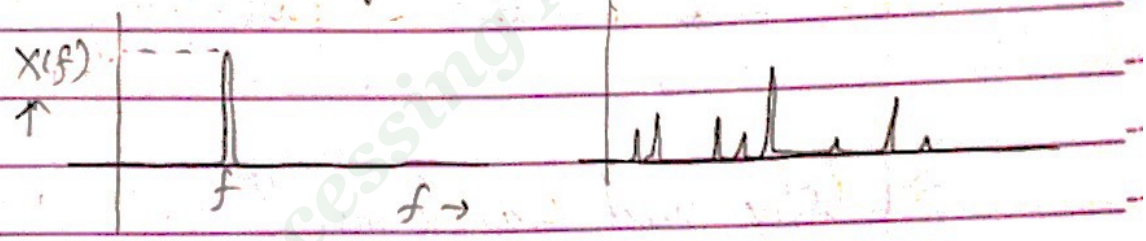
1D



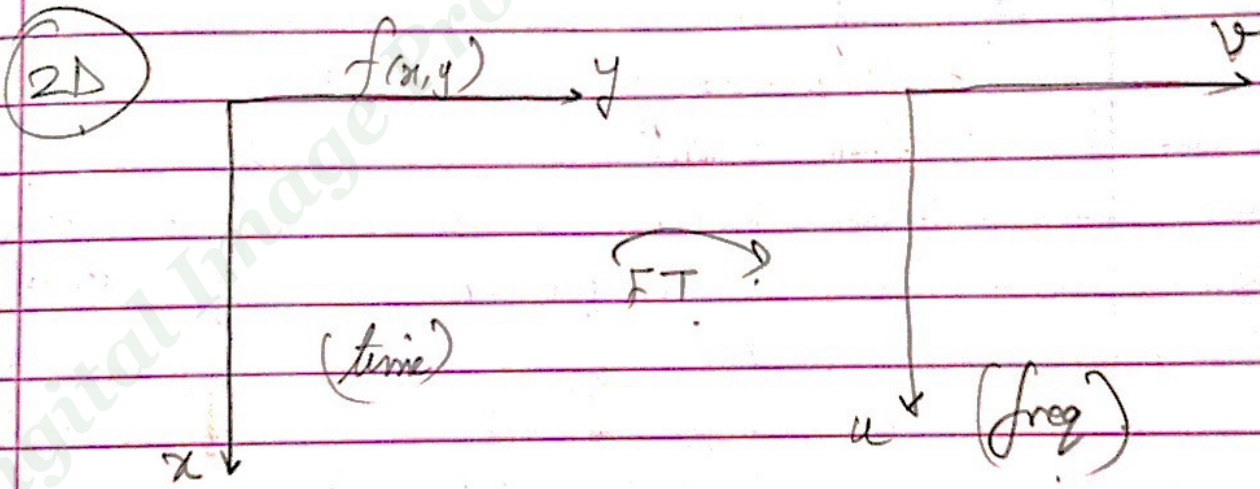
Fourier transform

Its a signal.

Frequency, given by (f or ω) is rate of change of signal w.r.t time.



2D



Spatial domain has variables x & y. And, its Fourier transform gives again (2) variables (for 2D)

→ rate of change of pixels in x dir^n, u

" " " " " " y dir^n, v

Note: FT distorts the image. We dont get original back

Now, consider a 1D signal, x(t)

* 1) Continuous Fourier Transform:

$$W(f) = \mathcal{F}\{w(t)\} = \int_{-\infty}^{\infty} w(t)\, e^{-j2\pi ft}\, dt$$

we get complex values of FT

$$W(f) = X(f) + jY(f)$$

$$W(f) = |W(f)|\, e^{j\theta(f)}$$

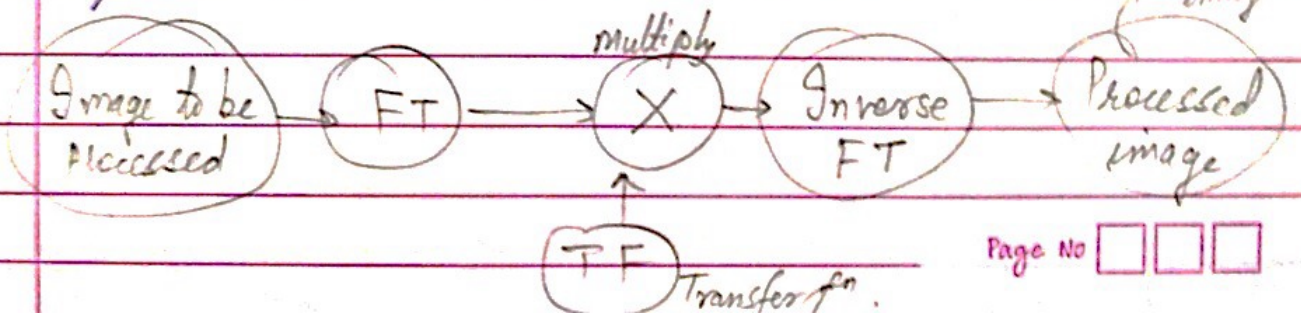
& $$w(t) = \mathcal{F}^{-1}\{W(f)\} = \int_{-\infty}^{\infty} W(f)\, e^{+j2\pi ft}\, df$$

* Why need transforms?
  - filtering (Fourier)
  - restoration "
  - enhancement "
  - compression (cosine transform)
  - image analysis (Haar, KL, Sine, PCA etc.)

PCA: Principle component analysis

* How transform is used in filtering & restoration oper^n.

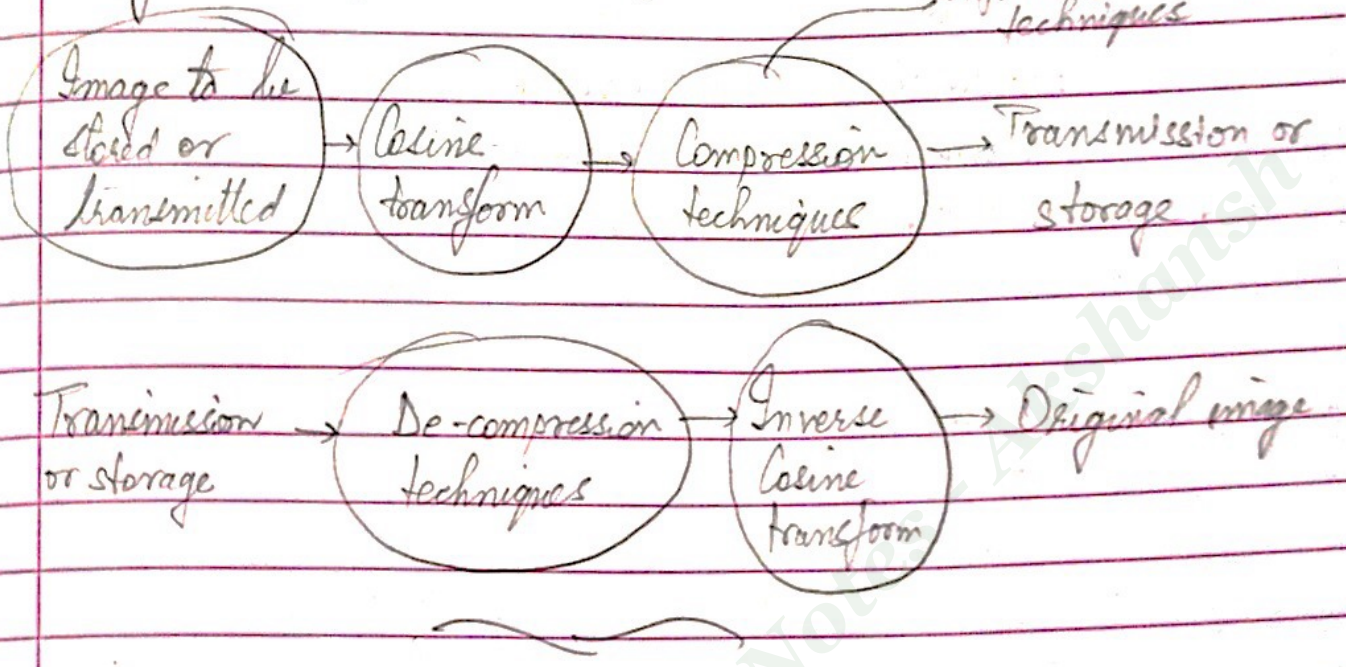
1D



★ $FT = LT\big|_{s=j\omega} = \int_{-\infty}^{\infty} f(t)\, e^{-st}\, dt$

CLASSMATE

Image transform in compression :-



★ 2D FOURIER TRANSFORM:

Say : Time variables : x, y.
Freq. variables : u, v

★ CONTINUOUS fourier transform (CFT)

Consider $f(x,)$ — 2nd variable yet to be added

$$\therefore F(v,) = \int_{-\infty}^{\infty} f(x,)\, e^{-j2\pi ux}\, dx$$

$\therefore f(x, y)$

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\, e^{-2\pi ux}\, e^{2\pi vy}\, dx\, dy$$

$$\Rightarrow F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\, e^{-2\pi(ux + vy)}\, dx\, dy$$

* Transformation Kernel.

Forward Transform^n Kernel

- eg: $e^{-j2\pi ft}$ is forward transform^n kernel for FT

eg: In 2D, $e^{-j2\pi(ux+vy)}$ → forward transform^n kernel for FT

Reverse Transform^n kernel.

- eg: $e^{+j2\pi ft}$ is reverse transform^n kernel for FT.

eg: In 2D, $e^{+j2\pi(ux+vy)}$ → reverse transform^n kernel for FT

* Discrete Fourier Transform (DFT)

(1D): Discrete time: $k = 0, 1, \ldots N-1$

Discrete freq: $n = 0, 1, \ldots, N-1$

N: sampling interval

$$\text{DFT: } X[n] = \sum_{k=0}^{N-1} x[k]\, e^{-j\left(\frac{2\pi}{N}\right)nk} \quad ; n = 0, \ldots N-1$$

$$\text{Inverse DFT: } x[k] = \frac{1}{N} \sum_{n=0}^{N-1} X[n]\, e^{+j\left(\frac{2\pi}{N}\right)nk} \quad ; k = 0, 1, \ldots N-1$$

→ Normalizing factor

(2D)

Discrete time : Variables $x, y = \in \{0, 1, 2, \ldots N-1\}$

Discrete freq : Variables $u, v = \in \{0, 1, \ldots N-1\}$

DFT : $F[u,v] = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j\frac{2\pi}{N}(ux+vy)}$ or

Inverse DFT: $f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v)\, e^{+j\frac{2\pi}{N}(ux+vy)}$

* Seeing 1D - DFT

$X(0) = x(0)(\,) + x(1)(\,) + \ldots + x(N-1)(\,)$

$X(1) = x(0)(\,) + x(1)(\,) + \ldots + x(N-1)(\,)$

$X(N-1) = x(0)(\,) + x(1)(\,) + \ldots + x(N-1)(\,)$

In matrix form :

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix}_{N\times 1} = \begin{bmatrix} & & \\ & W & \\ & & \end{bmatrix}_{N\times N} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}_{N\times 1}$$

→ kernel matrix or Transform matrix

So, in (1D) form, vector notation,

DFT $\Rightarrow$ $X = Wx$

$(N \times 1) = (N \times N)(N \times 1)$

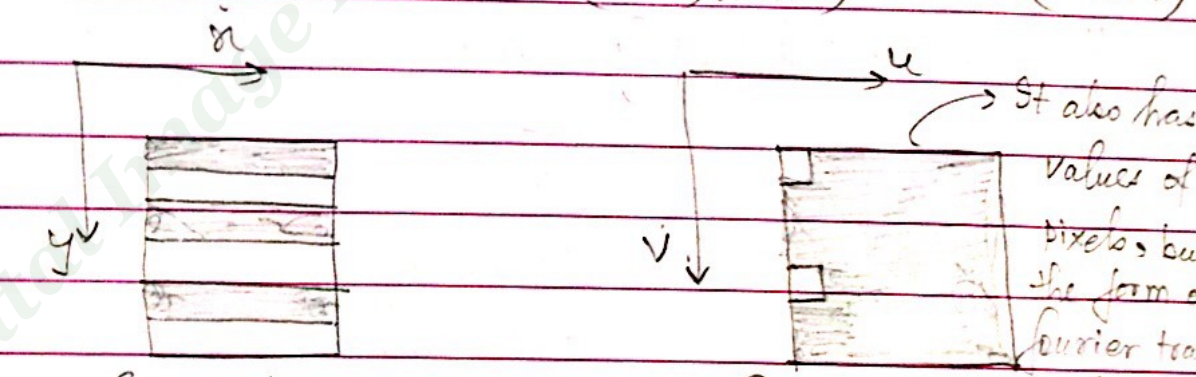
for (2D), vector notation,

$$DFT \rightarrow (X) = (W)(x)$$

$$N \times N = (N^2 \times N^2)(N \times N)$$

Using 2D-DFT:

$$\underset{N \times N}{[F]} = \underset{(N \times N)(N \times N)}{[W]} \times \underset{(N \times N)}{[f]}$$



Spatial Domain

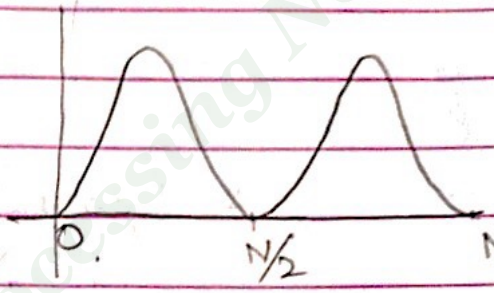
Spatial freq. domain

* Note: If we rotate the image in spatial domain, the change will be reflected in Spatial frequency domain. It is called ROTATION property.
→ i.e, do transpose

* Idea to see Spatial freq domain matrix on screen :
  - ↳ We have a large image, say.
    Then, its difficult to accomodate the freq values in a matrix & analyse that
    So, we use grayscale values to correspond to the freq values (quantized)
    0 grayscale = lowest freq
    255 grayscale = highest freq

* 0 to $\frac{N}{2}$ is the max. freq. DFT.

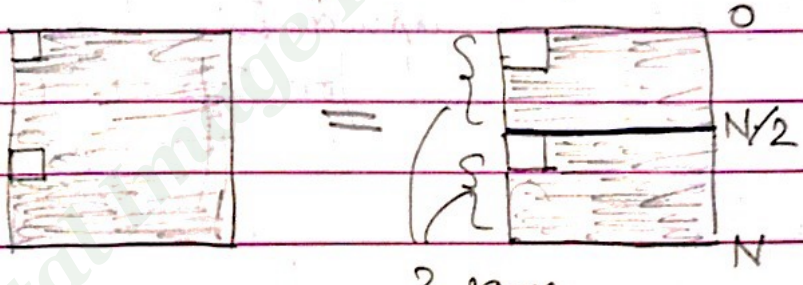
So, we have :



∴ we get same curve twice.

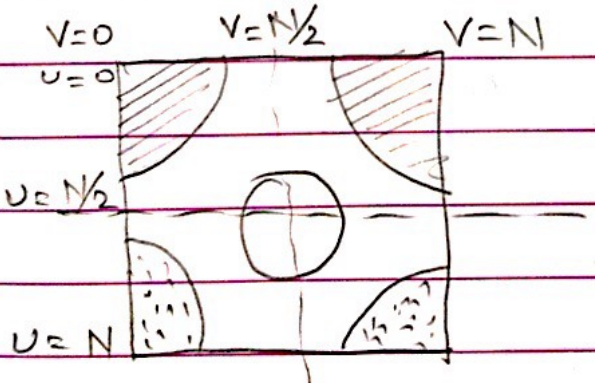
0, ... $\frac{N}{2}$ N-1

as we saw on previous page



2 same parts.

This is called SYMMETRY PROPERTY :-



* For 1D, DFT has a formula for quick computation: FFT (Fast Fourier transform)
∃ no separate 2D FFT available

* Inverse 2D DFT:

$$f(x,y) = \frac{1}{N \cdot N} \sum_{U=0}^{N-1} \sum_{V=0}^{N-1} F(U,V)\, e^{j 2\pi \frac{Ux+Vy}{N}}$$

- $\frac{1}{N \cdot N}$: a const^n, a normaliz^n factor — can be anything
- $x = 0, 1, 2, \ldots, N-1$
- $y = 0, 1, 2, \ldots, N-1$

A 2D DFT can also be written as: (Separable form)

$$F(U,V) = \frac{1}{N} \sum_{x=0}^{N-1} e^{-j\frac{2\pi Ux}{N}} \sum_{y=0}^{N-1} f(x,y)\, e^{-j\frac{2\pi Vy}{N}}$$

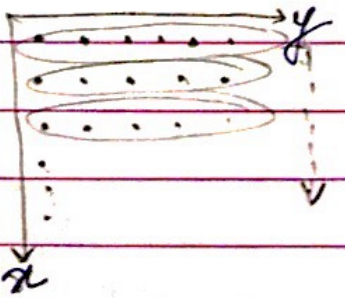
(1) (2)

If not used, coeff. value will be high, but, char. remains same.

* Separable transform:
A 2D eq^n can be split into 2 1D eq^{ns} & FFT algo. can be applied two times to compute result for 2D DFT.

Consider part (2), we find only y is varying
i.e, only column is varying.
⇒ its like 1D

So, in part (2), we take 1D DFT of all columns.

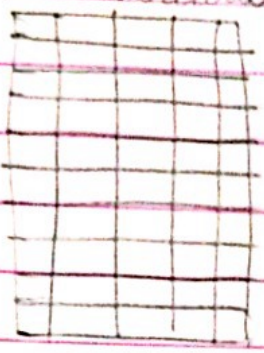


After doing 1D DFT of all columns, we get another matrix of 1D FT of all rows.

* Notation : ~~(x,y)~~ $(x, y) \equiv$ ~~(k, l)~~ (i, j)

$(u, v) \equiv (k, l)$

Note : for part (2), after doing 1D DFT, I get values. Now, these coefficients, multiplied with variation in x (rows) make a matrix.



Now, applying DFT on this, we again get a FT matrix of diff. (This is 2D DFT.

Now, seeing it mathematically.

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

$$\Rightarrow F(u,v) = \frac{1}{N} \sum_{y=0}^{N-1} P(u,y)\, e^{-j2\pi \frac{ux}{N}}$$

$$\hookrightarrow P(u,y) = \frac{1}{N} \sum_{x=0}^{N-1} f(x,y)\, e^{-j2\pi \frac{vy}{N}}$$

* Now, on doing FT, we may get coeff. in the range very low in value (0, 1, 2 ..., say)
Using transformns (like log transforms) can shift the pixel values from low to high. (range, 0 to 255)

* Separable transform : Only in 2D, not in 1D.

* Shift Property

Shift in time domain = Multiplicⁿ by exponential in freq. domain.
(& vice versa)

** $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0+vy_0)/N}$

$f(x,y)e^{j2\pi(u_0x+v_0y)/N} \Leftrightarrow F(u-u_0, v-v_0)$

↳ This property can be used to shift an image from corner to center
↳ used in printing operⁿ.

* Scaling Property

Shrinking in one domain causes expansion in other domain.

$$f(ax, by) \longleftrightarrow \frac{1}{|ab|} F\left(\frac{u}{a}, \frac{v}{b}\right)$$

* Rotation Property

Rotation in one domain causes rotation in other domain.

eg. In shift Property, substitute $u_0 = \frac{N}{2}, v_0 = \frac{N}{2}$

⇒ we get

$$F\left(u-\frac{N}{2}, v-\frac{N}{2}\right) \Leftrightarrow f(x,y)e^{j\pi(x+y)}$$

CLASSMATE

$\Rightarrow F(U-\frac{N}{2}, V-\frac{N}{2}) \Longleftrightarrow f(x,y)\left(e^{j\pi}\right)^{(x+y)}$

$\Rightarrow F(U-\frac{N}{2}, V-\frac{N}{2}) \Longleftrightarrow f(x,y)\,(-1)^{x+y}$

↳ So, if we want to shift pixels to center, multiply every pixel by $(-1)^{x+y}$. Then, take FT & we get $F(U-\frac{N}{2}, V-\frac{N}{2})$.

MATLAB

fn : fftshift : can do the above oper^n.

↳ Proof for these properties can come in test (search web & write)

✡ WALSH-HADAMARD TRANSFORM (done later)

• General form of any transform:
Consider a 2D image $g(x,y)$ of size $n \times n$, whose forward Discrete transform $T(x,u,v)$ can be expressed as:

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

$r(x,y,u,v)$ → forward transform kernel

↳ $\forall\ u, v = 0, 1, 2, \ldots n-1$

Inverse discrete transform $g(x,y)$

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

$s(x,y,u,v)$ → reverse transform kernel

↳ $\forall\ x, y = 0, 1, 2, \ldots n-1$

✡ Diff transforms have diff kernels. Rest remains same.

✡ Transform kernel is also called Basis f^n or basis image. → forward & reverse

∴ 2D

Properties: P1) If Kernel is separable,

$$r(x,y,u,v) = r_1(x,u)\, r_2(y,v)$$

P2) If Kernel is symmetric,

$$r(x,y,u,v) = r_1(x,u) \cdot r_1(y,v)$$

(same for reverse transform) ✡

P3) Transform kernels decide the type of transform.

↳ eg, 2D DFT.

forward transform kernel.

$$k(x,y,u,v) = e^{-j\frac{2\pi(ux+vy)}{n}}$$

$$s(x,y,u,v) = \frac{1}{n^2} e^{+j\frac{2\pi(ux+vy)}{n}}$$

✱ Walsh-Hadamard Transform

↳ useful in transform coding.

Forward (&) Reverse Transform kernel is given by:

$$g(x,y,u,v) = h(x,y,u,v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1}[b_i(x)p_i(u) + b_i(y)p_i(v)]}$$

↳ $n = 2^m$

✱ Forward & reverse transform kernel is same.

✱ Note: Summation of terms in exponent is performed in modulo 2 arithmetic.

↓

binary arithmetic (i.e, basically dont take carry)

✱ $b_k(z)$: k^{th} bit (Right to left) of binary value of z.

eg: $m = 3$, $z = 6$ (110), then, $k = 0, 1, 2$.

$m = 3$

$\Rightarrow b_2(z) = 1 \quad b_1(z) = 1 \quad b_0(z) = 0$

Page No

* $P_1(u)$ can be computed as:

$$P_0(u) = b_{m-1}(u)$$
$$P_1(u) = P_0(u) + b_{m-2}(u) = b_{m-1}(u) + b_{m-2}(u)$$
$$P_2(u) = b_{m-2}(u) + b_{m-3}(u)$$
$$\vdots$$
$$P_{m-1}(u) = b_1(u) + b_0(u)$$

Q.(A)

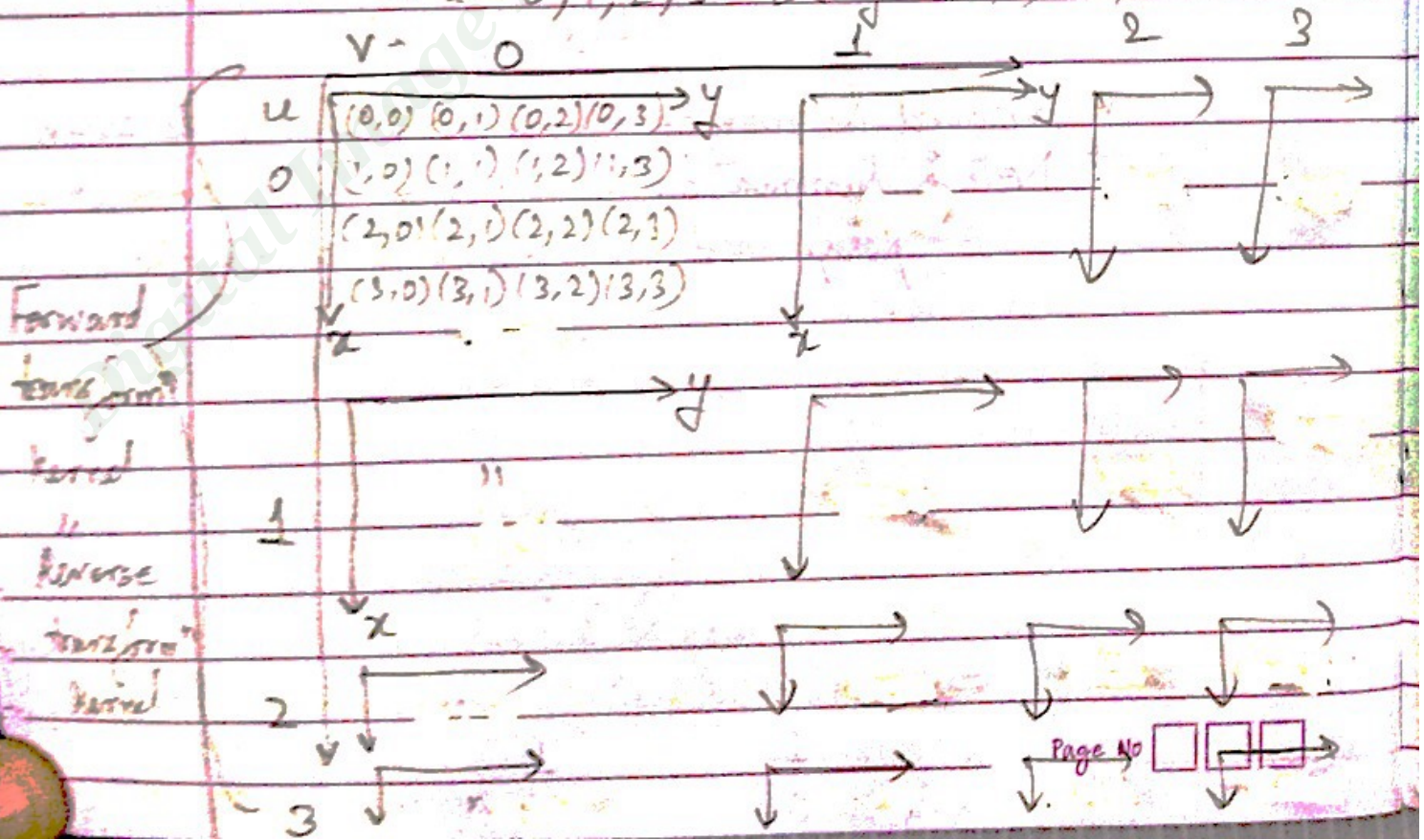
Consider $n=4 \Rightarrow u = 0,1,2,3$, $v = 0,1,2,3$

Find $b(x,y,u,v)$ → basis image

↳ Idea: for any n, general size of kernel = $n^2 \times n^2$

Here: $n=4$

$\Rightarrow x = 0,1,2,3$ & $y = 0,1,2,3$



CLASSMATE

Now, finding for $\boxed{u=0,\ v=0}$, $x=0,\ y=0$

$$\Rightarrow L(0,0,0,0) = g(0,0,0,0) = \frac{1}{4}(-1)^{\sum_{i=0}^{n-1}[b_i(0)P_i(0)+b_i(0)P_i(0)]}$$

$$= \left(\frac{1}{4}\right)(-1)^{[b_0(0)P_0(0)+b_0(0)P_0(0)+b_1(0)P_1(0)+b_1(0)P_1(0)]}$$

$a_k(z)$ = k^{th} bit of z

$b_k(0)$ = k^{th} bit of 0 = 0

$z = (0\ 0)$ → $b_1(0)$, $b_0(0)$

↳ $b_0(0) = 0 = b_1(0)$

↳ $P_0(0) = b_0(0) = 0$

$P_1(0) = b_1(0) + b_0(0) = 0$

$\Rightarrow L(0,0,0,0) = g(0,0,0,0) = 1/4$

Now, $L(0,0,0,1) = g(0,0,0,1)$

$$= \left(\frac{1}{4}\right)(-1)^{\sum_{i=0}^{n-1}[b_i(0)P_i(x)+b_i(0)P_i(y)]}$$

& $x=0,1,2,3 = y$

doing for all $u, v = 0, v_k$'s we get

$v = 0$ →

$u=0$ ↓				
	1/4	1/4	1/4	1/4
	1/4	1/4	1/4	1/4
	1/4	1/4	1/4	1/4
	1/4	1/4	1/4	1/4

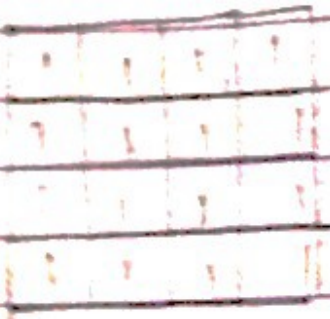
↳ gives one matrix (for $u=0=v$)

out of 16 matrices.

In we can find matrices for diff^t

Page No

Q. Given an image of size 4×4



Take this image, superimpose for $u=0, v=0$ ... Ans $u=3, v=3$ for WH transform Kernel multiply coinciding element & add.

→ WH kernel for $u=0, v=0$ was

Kernel			
1/4	1/4	1/4	1/4
1/4	1/4	1/4	1/4
1/4	1/4	1/4	1/4
1/4	1/4	1/4	1/4

×

Image			
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

$$= (1/4)(1) + (1/4)(1) + (1/4)(1) + \dots + (1/4)(1)$$

$$= 16\left[\frac{1}{4}\right] = 4$$

So, Transform =

	V=0	1	2	3
u=0	4			
1				
2				
3				

Similarly do for all values of Kernel
($u=0, v=1$; $u=0, v=2$; $u=0, v=3$; ..... $u=3, v=3$)

Idea :- We are finding transform kernel from the WH transform. This kernel is basically 16 matrices (for n=4). Now, each matrix, when convolved with my image gives WH transform for that matrix (& hence pixel → out of 16)

* Note : eg :- for $b_k(2) = 10$ (2 in binary)

$\rightarrow$ ∴ $b_0(2) = 0$

$b_1(2) = 1$.

Also, $b_k(8) = 100$.

$\rightarrow b_0(8) = 0$

$\rightarrow b_1(8) = 0$

$\rightarrow b_2(8) = 1$

* Type of Question on this topic

Given an image. (Say, for n=4, image = 4×4)

Say, image is

6	8	7	8
9	10	20	30
3	4	5	6
7	8	9	1

Find WH transform of any of the element of this image. Say, for (1,1) element

find u=0, v=0 & multiply coinciding elements and add

Q(A) continued) Previously we did WH transform matrix calculation for the matrix (0,0) [u=0=v]. Now, finding the matrix for (3,3) [u=3=v]

So, we want to fill this : (Part 16 (out of 16))

		V = 3		
	?	?	?	?
	?	?	?	?
U = 3	?	?	?	?
	?	?	?	? → U=3, V=3, x=3, y=3

Now,

We know :

$$h(x,y,u,v) = \left(\frac{1}{4}\right)(-1)^{\sum_{i=0}^{m-1}[b_i(x)P_i(u)+b_i(y)P_i(v)]}$$

$$= \left(\frac{1}{4}\right)(-1)^{\left(\begin{matrix} b_0(x)P_0(u)+b_0(y)P_0(v) \\ +b_1(x)P_1(u)+b_1(y)P_1(v)\end{matrix}\right)}$$

* Note : $m = 2$ ($\because n = 2^m \Rightarrow 4 = 2^m$)

$$\Rightarrow h(x,y,3,3) = \left(\frac{1}{4}\right)(-1)^{\sum_{i=0}^{1}[b_i(x)P_i(3)+b_i(y)P_i(3)]}$$

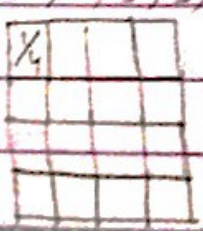
↳ Solve for $x = 0,1,2,3$ & $y = 0,1,2,3$

① $x = 0, y = 0$.

$$\Rightarrow h(0,0,3,3) = \left(\frac{1}{4}\right)(-1)^{\left(\begin{matrix} b_0(0)P_0(3)+b_0(0)P_0(3) \\ +b_1(0)P_1(3)+b_1(0)P_1(3)\end{matrix}\right)}$$

Now, $(0)_{10} = (\overset{b_1\ b_0}{00})_2 \Rightarrow b_0(0) = 0,\ b_1(0) = 0$

$$\Rightarrow h(0,0,3,3) = \left(\frac{1}{4}\right)(-1)^{(0+0+0+0)} = \left(\frac{1}{4}\right),$$



As we proceed further, we find that the got values are $\frac{1}{4}$ or $-\frac{1}{4}$. Taking $\left(\frac{1}{4}\right)$ common for whole WH transformed matrix, we get ± 1 inside

Let $+1$ = white
-1 = black.

So, the whole transformed pixel set can be made into B & W image — This is called BASIS IMAGE
(Here, Basis image is BINARY)

✓ It has 2 values → +1 & -1. So, comput^{n} is fast as compared to Fourier transform.
✓ has applic^{n} in image compression

Note :- $(3)_{10} = (11)_2$ → $\Rightarrow b_0(3) = 1$ & $b_1(3) = 1$

& $P_0(3) = b_1(3) = 1$

$P_1(3) = P_0(3) + b_0(3) = 1 + 1 = 0$
(modulo-2 addition)

(2) $x = 2, y = 2$

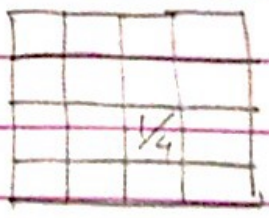
$$g(2,2,3,3) = \left(\frac{1}{4}\right)(-1)^{\sum_{i=0}^{1} [b_i(2)P_i(3) + b_i(2)P_i(3)]}$$

$$= \left(\frac{1}{4}\right)(-1)^{[b_0(2)P_0(3) + b_1(2)P_1(3)] + (b_0(2)P_0(3) + b_0(2)P_0(3))}$$

$$= \left(\frac{1}{4}\right)(-1)^{[0\times 1 + 1(0)] + (0+0)}$$

$$= \left(\frac{1}{4}\right)(-1)$$

$\Rightarrow g(2,2,3,3) = \left(\frac{1}{4}\right)$



So, we have filled →

(3) $(x,y) = (3,1)$

$$c(3,3,3) = \left(\frac{1}{4}\right)(-1)^{\frac{3}{2}} \sum_{i=0}^{1} a_i(3) A_i(3) + b_i(3) B_i(3)$$

$$= \left(\frac{1}{4}\right)(-1)^{3} \; a_0(3)[b_0(3) + b_0(1)] + A_1(3)[b_1(2) + b_1(1)]$$

$$= \left(\frac{1}{4}\right)(-1) \; 1[0+1] + 0[1+0]$$

$$= \frac{1}{4}(-1) = -\frac{1}{4}$$

(4) $(x,y) = (3,1)$

$$c(3,3,3) = \left(\frac{1}{4}\right)(-1)^{\frac{3}{2}} \sum_{i=0}^{1} a_i(3) A_i(3) + b_i(2) B_i(3)$$

$$= \left(\frac{1}{4}\right)(-1) \; B_0(3)[b_0(3) + b_0(1)] + B_1(3)[b_1(3) + b_1(1)]$$

$$= \left(\frac{1}{4}\right)(-1) \; 1[1+0] + 0[1+0] = 1/4$$

(5) $(x,y) = (3,3)$

$$c(3,3,3) = \left(\frac{1}{4}\right)(-1)^{\frac{3}{2}} \sum_{i=0}^{1} b_i(3) A_i(3) + b_i(3) B_i(3)$$

$$= \left(\frac{1}{4}\right)(-1) \; \overbrace{b_0(3) b_0(3)}^{0} + \overbrace{b_0(3) B_0(3)}^{0} + b_1(3) A_1(3) + b_1(3) B_1(3)$$

$$= \left(\frac{1}{4}\right)(-1)^{0+0} = \left(1/4\right)$$

We can find all values for $v = V = 3$ without using $\frac{1}{4}$ to fill the matrix.

☆ QUIZ-1 Notes:

- Harmonic mean filter: $f = \frac{MN}{\sum \frac{1}{g(x,y)}}$
- Arithmetic mean filter: $f = \frac{1}{MN} \cdot \sum g(x,y)$
- Geometric mean filter: $f = \left(\prod g(x,y)\right)^{1/MN}$
- Shifting Property in F(u,v)

Suppose N=4 & we are going $F\left(u-\frac{N}{2}, v-\frac{N}{2}\right)$

$\Rightarrow F(u-2, v-2)$

Consider image

10	1	4	40
1	1	4	4
3	3	2	2
30	3	2	20

1st (u ↓), 2nd (u ↓), 1st (v →), 2nd (v →)

Transformed

2	2	3	3
2	20	30	3
4	40	1	1
4	4	1	1

nature: [illegible]

- Original image - blurred image = Mask
- Original image + k x Mask
  - ↳ If k = 1 : UNSHARP MASK
  - ↳ If k > 1 : HIGH BOOST FILTERING
- Rotation Property of DFT:

If f(x,y) is rotated, F(u,v) rotates by equal amount

i.e. If $f(x,y) \leftrightarrow F(u,v)$ then $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

& $f(x',y') \leftrightarrow F(u',v')$ & $\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$

☆ Self: Find 2D DFT of $f(x,y) = \frac{x+y}{32}$

✱ Introduction to Discrete Cosine Transform

✱ Consider a data sequence

d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7

doing 1D DFT →

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7

↳ $w_0, w_1 \ldots w_7$: 1D DFT coefficients

All these coeffs are reqd (as it is) to get back my data

doing Discrete cosine transform (DCT) (1D) →

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7

↳ Unique property is ∃ even if 4 remove coeffs from C_3 to C_7, still, I get 95% of data back.

How come this compression?

Quantiz^n of DCT coefficients

C_0	C_1	C_2	X X X

Human eye can't detect diff b/w original image & 95% of original.

Now,

for my data sequence, let each data is in 8bits

↳ 8×8 = 64 bits are reqd.

In DFT also, we again need 64 bits

Using DCT, 5 coefficients are made 0.

So, 8 + 8 + 8 + 1 + 1 + 1 + 1 + 1 = 29 bits.

C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7

$C_3 \to 0$, $C_4 \to 0$, $C_5 \to 0$, $C_6 \to 0$, $C_7 \to 0$

Page No

Energy compaction property of DCT:
Entire energy is concentrated in the lower frequency part.

* DCT is a lossy compression method
  ↳ used in JPEG compression

* 1D DCT

Discrete cosine transform:

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

↳ $\alpha(k) = \begin{cases} \sqrt{1/N} & ; k=0 \\ \sqrt{2/N} & ; \text{otherwise} \end{cases}$

* Strength of 'u' is a sinusoid given by C(u)
  ↳ Process f on basis of u.
  ↳ All samples of f contribute the coeff.
  ↳ C(0): ZERO FREQ component.
  ∴ called AVERAGE value

2D DCT

eg Consider a digital image. One row of it is 1D.

Let values are

Index	0	1	2	3	4	5	6	7
Value	20	12	18	56	83	10	104	114

We get 8 DCT coeff → 0 to 7.

Finding coeffs :-

$$C(0) = \alpha(0) \sum_{x=0}^{N-1} f(x) \cos(0)$$

$\alpha(0) \rightarrow \sqrt{1/N}$

$$\Rightarrow C(0) = \sqrt{\frac{1}{N}} \sum_{x=0}^{N-1} f(x) \equiv \text{avg. Value}$$

↳ for $N = 8$.

we get

$$C(0) = 0.35\,(20 + 12 + 18 + 56 + 83 + 10 + 104 + 114)$$

$$\Rightarrow C(0) = 183.14$$

Hy, find $C(1), C(2) - - - C(7)$

Ex, 1st row of image's freq. coeff :

Index	C(0)	C(1)	C(2)	- - -	C(7)
Value	183.14				

✡ 2D DCT

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

basis fr

- used in JPEG image compression
- For MxN image, ∃ MxN coeffs
- Each image sample contributes to each coeff
- Each (u, v) pair corresponds to a "pattern" or 'BASIS FUNCTION'.
  - To get basis image:

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

(just like in WH Transform)

(In case of WH transform we had only 2 values Black (-1) & White (+1))

✡ DCT is separable

Image → Do DCT along rows

↓

We get transformed image ← Do DCT along columns

★ DCT is Invertible

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

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

★ Energy Compaction:

MATLAB

- Low freq coeffs. have larger magnitude.
- High freq coeffs have smaller magnitude.
- Most info is compacted into lower freq. coeff.
- From matrix pt. of view — take only upper triangular value
- Done in JPEG.

CLASSMATE

QUIZ-1 DISCUSSION

Q3 Give a 3×3 mask for performing unsharp masking through a single pass.

Given Average image get key : $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Unsharp masking : Image − (LP avg) + Image

So, $f(x,y) + f(x,y) - \underbrace{\bar{f}(x,y}_{\text{Smoothed or average}}$

$$= 2f(x,y) - \frac{1}{9}\underbrace{\left[f(x,y) + f(x-1,y) + f(x,y-1) + \dots\right]}_{\text{all 9 terms of matrix}}$$

$$= 2f(x,y) - \frac{1}{9}f(x,y) - \frac{1}{9}\left[f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) + \dots\right]$$

$$= \frac{17}{9}f(x,y) - \frac{1}{9}\left[\quad\right]$$

$$= \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Q.4) Ans: $e^{j2\pi\left(\frac{u_0 x}{M} + \frac{v_0 y}{N}\right)}$ → for M×N image

Q.5) Complete following wrt linearity property of 2D DFT:

$f_1(x,y) \longleftrightarrow F_1(u,v)$

$f_2(x,y) \longleftrightarrow F_2(u,v)$

$a_1 f_1(x,y) + a_2 f(x,y) \longleftrightarrow a_1 F_1(u,v) + a_2 F_2(u,v)$

Q.6) Filter mask for Laplacian:

0	1	0
1	-4	1
0	1	0

or

-1	-1	-1
-1	8	-1
-1	-1	-1

Q.7) Apply Sobel operator on (2, 2)

$\nabla^2 f = \text{mag} = |G_x| + |G_y|$

Q.8) Given image $f(x,y)$, find 2D DFT of image & find $F(u-\frac{N}{2}, v-\frac{N}{2})$

Multiply each image by $(-1)^{x+y}$ (Spatial domain)

2D DFT, DCT, WH transform.

✡ Using Transforms to Filter Image.

• Seeing Filtering in 1D.

Consider a 1D image $f(x)$ & its FILTER/ IMPULSE RESPONSE as $h(x)$

O/P is seen as: $y(x) = f(x) * h(x)$ (time domain)

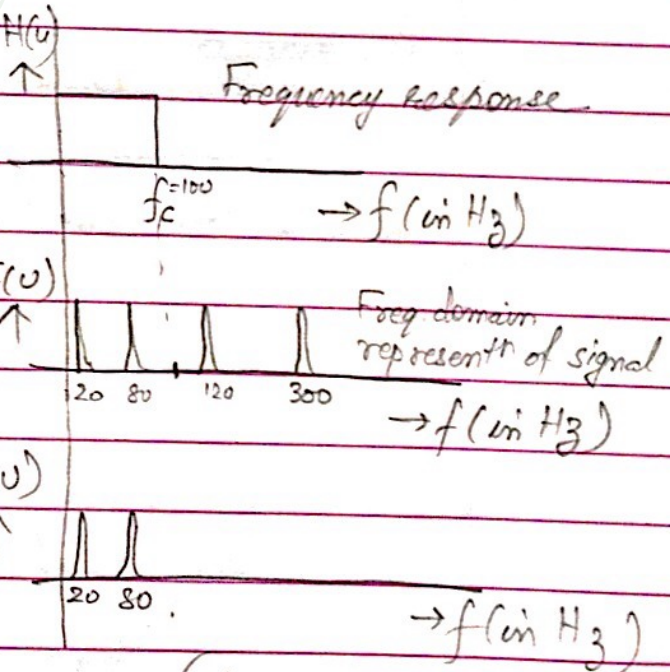
writing in freq. domain:- ⇕

Image $F(U)$, filter: $H(U)$ → FREQUENCY RESPONSE.

& , $Y(U) = F(U) \cdot H(U)$

Normally,

Ideal freq. response:



Suppose freqs are 20, 80, 120 & 300

cutoff = 100 Hz.

So, $Y(U) = F(U) \cdot H(U)$ is filtered signal

↓ do inverse FT to get original signal back

Seeing for 2D.

Given an image $f(x,y)$
Its corresponding FT : $F(U,V)$

$$f(x,y) \overset{\text{2D DFT}}{\Longleftrightarrow} F(U,V)$$

$$*$$

$$\& \quad h(x,y) \overset{\text{2D DFT}}{\Longleftrightarrow} H(U,V)$$

$$\| \qquad\qquad \| \qquad\qquad \|$$

$$= \quad g(x,y) \overset{\text{2D DFT}}{\Longleftrightarrow} G(U,V)$$

Filtering in time (spatial) domain :

$$g(x,y) = f(x,y) * h(x,y)$$

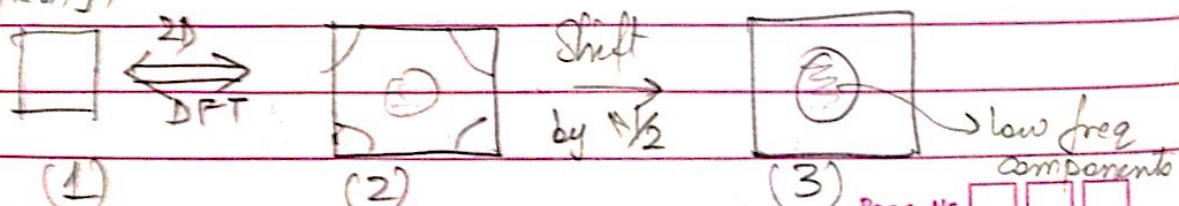
In freq. domain :

$$G(U,V) = F(U,V) \cdot H(U,V)$$

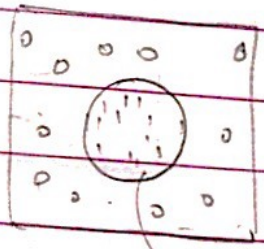
Note : As done before :

$f(x,y)$ is an image. When done FT, we again get coeff., s.t low (zero) freq at corners & high freq in middle.
Now, shifting by $\frac{N}{2}$ will make

Zero freq in centre & high freq at corners



Now, H(U,V) can be seen as an image s.t all low freq's are 1 and remaining freq's are 0 & 1 is in centre & 0's elsewhere

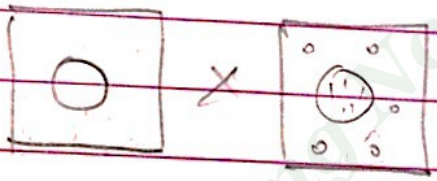


(4) → f(c)

So,

G(U,V) = (3) × (4)

=



↓ doing Inverse 2D DFT

g(x,y)

(low pass operation done)

MATLAB

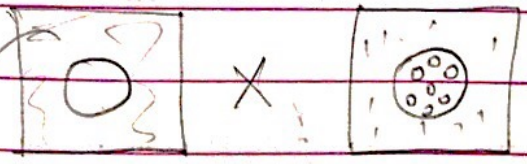
Applying high pass operation:

(Image + 2D DFT + Shift by N/2)

×

(H(U,V) + Inverse)

1 become 0 & 0 becomes 1



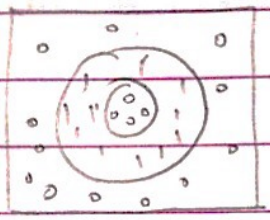
= G(U,V)

(high pass)

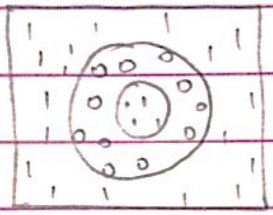
high freq components

Note :- If we want to do Band pass & Band stop oper^n, H(u, v) changes as :

Band pass :



Band stop :



Note : In DFT, we have diff^t intensity levels. So, they are represented in the form of values from 0 to 255 for diff^t values of freqs

0 → lowest freq
255 → highest freq

OR

255 → lowest freq
0 → highest freq

Use any 1 for coding.

* Frequency Domain Filtering

Note : In Freq. domain multiplic^n happens.

Idea : FT coeff × Transfer fn. or mask

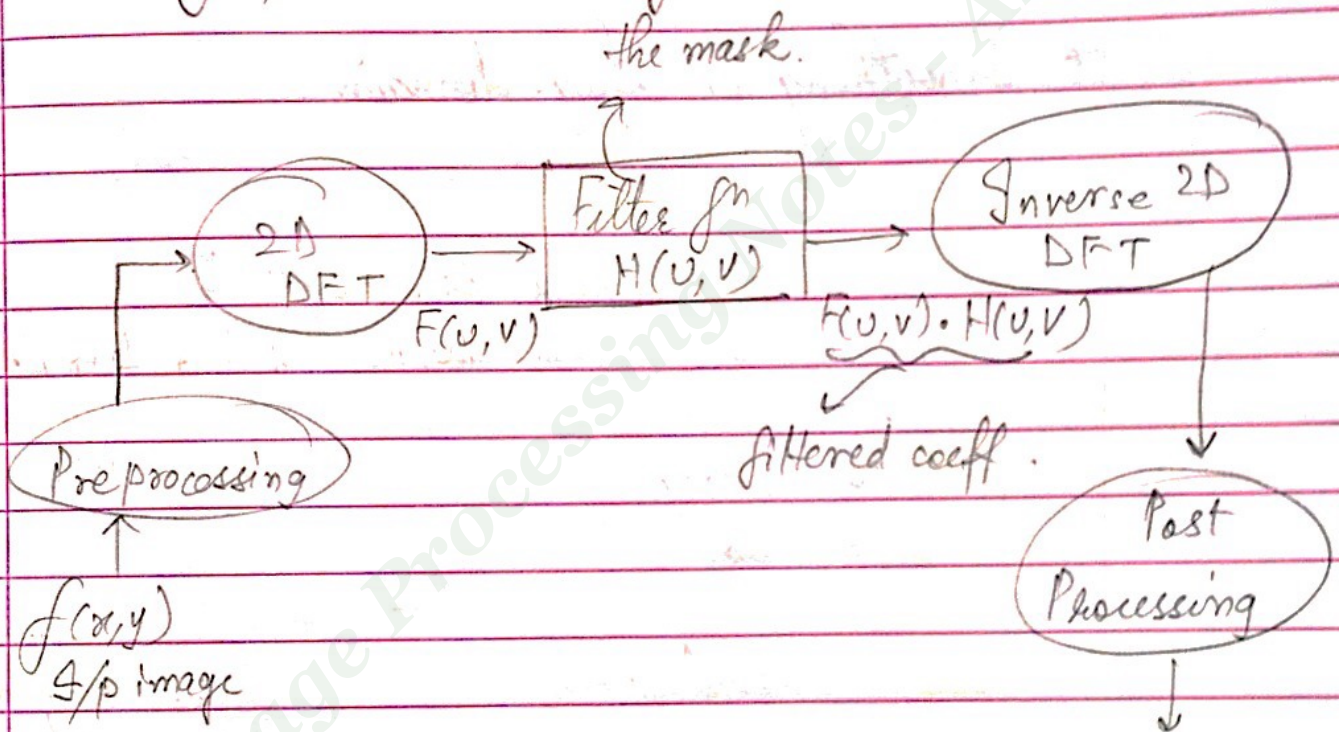
I want to retain a particular section of my FT coeff :

$$(\text{FT coeff. matrix})_{M \times N} \times (\text{A mask})_{M \times N}$$

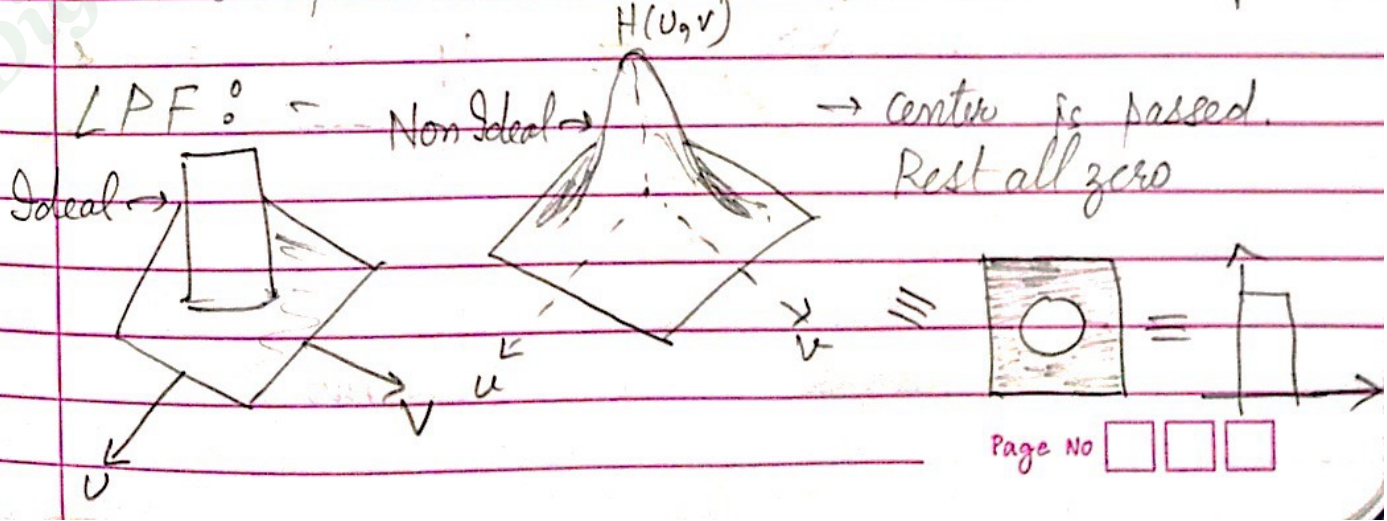
* LPF : Smoothens.
* HPF : Sharpens.

This mask is chosen by our choice.
If we need high freq components :
centre values = 0, outer values = 1
That will show high freq coeffs only for FT image.

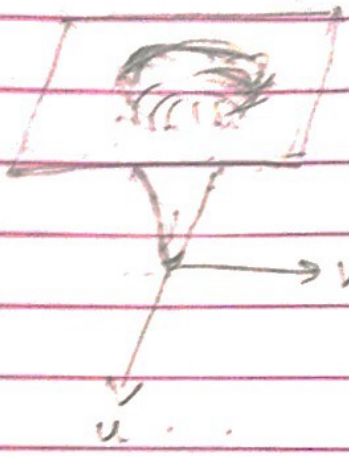
Filtering oper^n block diagram



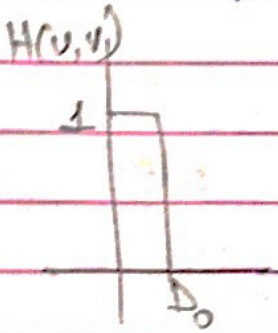
* 3D view of 2D mask



HPF :-



* Smoothing in freq. domain.
  ↳ remove all high freq.

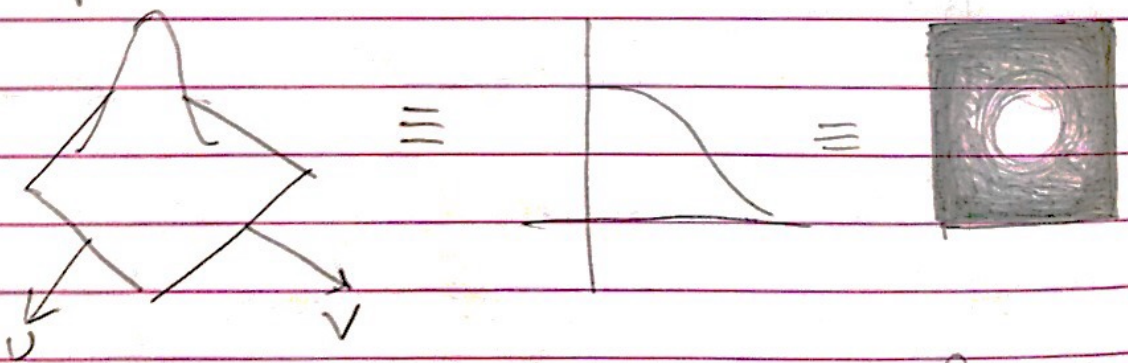


Ideal LPF eqn :

$$H(U,V) = \begin{cases} 1, & D(U,V) \leq D_0 \\ 0, & D(U,V) > D_0 \end{cases}$$

↳ $D(U,V) = \sqrt{\left(U - \frac{M}{2}\right)^2 + \left(V - \frac{N}{2}\right)^2}$

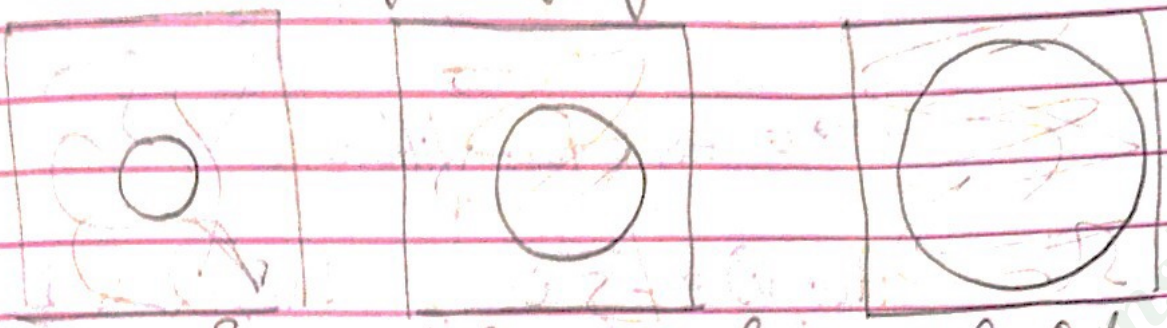
Moving to Non-Ideal filter :-



(Similar are the char. of Butterworth filter → PTO)

A 2D DFT of my image gives us an image :

MATLAB



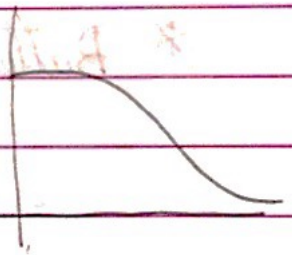
We are choosing diff^t radii of D_0 (the cut off freq. pt).

If D_0 is small (radius in pixels), very small freq's are passed. So, image is nearly blurred.

as we increase value of D_0, we start getting actual image back.

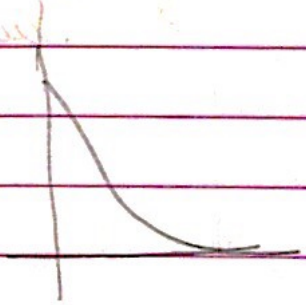
* Butterworth filter : LPF with order n.

$$H(u,v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$$



* Gaussian LowPass Filter :-

$$TF : H(u,v) = e^{\frac{-D^2(u,v)}{2D_0^2}}$$



(PB width is lower as compared to Butterworth)

- used to connect Broken text.

- used to remove blemishes in photographs

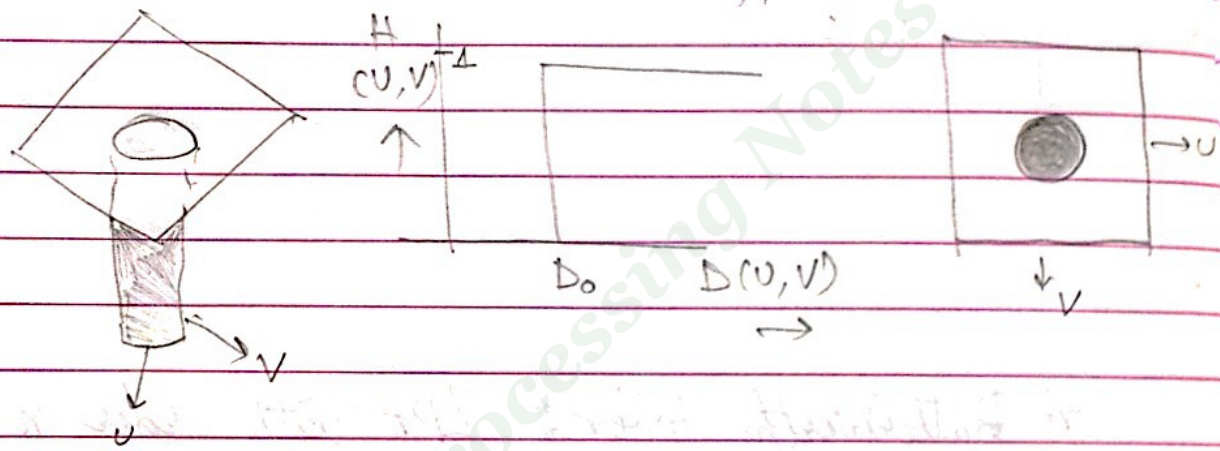
✱ Sharpening in Frequency Domain.

→ edges and fine details in images are associated with high freq. components.

→ ≡ High Pass filters.

$$H(u,v) = \begin{cases} 0, & D(u,v) \leq D_0 \\ 1, & D(u,v) > D_0 \end{cases}$$

↳ D_0, cut off distance



✱ Butterworth filter : HPF with order n.

≠

✱ Gaussian High Pass filter

$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0}}$$

✓ applic^n in medical field

↳ image + HPF + Histogram equaliz^n.

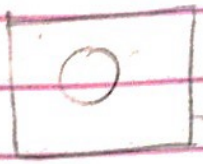
CLASSMATE

* Idea : Transfer function

1D

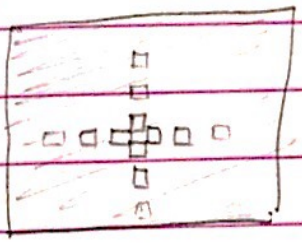
$$H(s) = \frac{(\quad)}{(\quad)}$$

2D

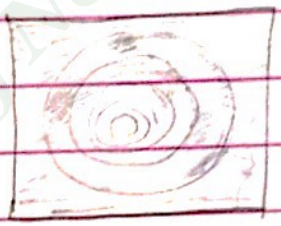
$H(u, v) =$  → Image

☆ Laplacian in the Frequency Domain

Spatial domain



Frequency domain



$\Downarrow$

0	1	0
1	-4	1
0	1	0

Note :- Applying Laplacian in
- Spatial domain $\Leftrightarrow$ Freq. domain
  - → Result is same

- ✓ consists of lot of operⁿs MAC (Multiplicⁿ & Accumulⁿ)
- ✓ Slower (more time consuming)

- ✓ Consists of Multiplicⁿ operⁿ & FFF.
- ✓ Fast (∵ we have hardware implementation)

FFT: time kept to perform FT reduced by 100–600 times.

* MMX technology : Multi Media Exchange
(a vdo can be processed in micro proc.)

CLASSMATE

* $O(N^2)$; O : Operator
  - $O(\log_2 N)$
  - N^2 : no. of operns

Chapter

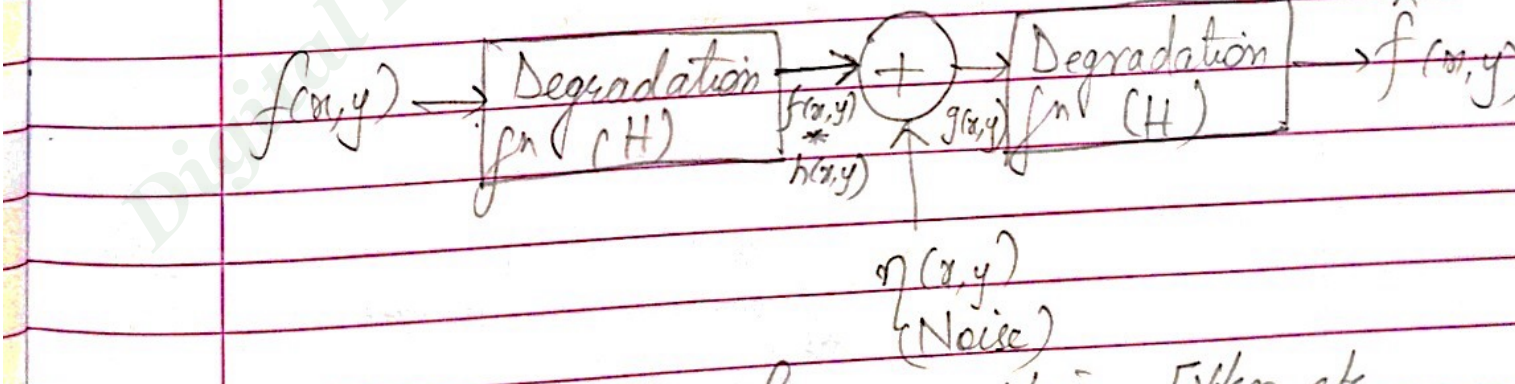
§ IMAGE RESTORATION

- ✓ restoring an image from noise.
- ✓ more complex and mathematical than image enhancement

✓ we were blindly applying our methods on our image to see if it's enhanced.
Now, we'll look into what type of noise it is

eg: I eat chilli (having a noisy image)
So, I'll know its chilli and how to cure it (studying the type of noise).
I'll take cigar etc. to balance effect (using restoration method to remove noise and get image with no noise / lesser noise)

* Model for Image Degradation / Restoration process



Degradation can happen as Noise, Filter etc

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

Page No

* The fn. H gives the mathematical represent^{n} of the degradation that happened to the image.

*  : When an object's picture is taken on motion (relatively), the pixels have been moved from their original pos^{n}. So, when brought back to original pos^{n}, we get image back.

* A degradation fn (H) can be seen as a filter.
* Idea: Whichever degradation and noise was added, an inverse fn is made to remove its effect. Now, plan is to estimate $h(x,y)$ (With that known, inverse can be found & image can be restored.
  - estimation done by seeing type of image.

→ or $H(u,v)$ in freq. domain.

$$g(x,y) = f(x,y) * h(x,y) + \eta(x,y) \quad \eta = 0, \text{ say}$$

$$\updownarrow$$

$$G(u,v) = F(u,v) \cdot H(u,v) + N(u,v) \quad \eta \to 0 \text{ (considered zero)}$$

↳ Let the estimate of $H(u,v)$ be $\hat{H}(u,v)$

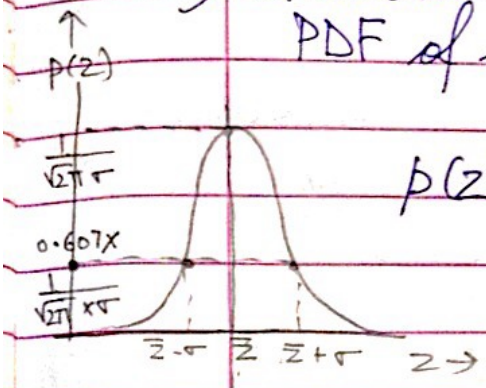
& $\underbrace{F(u,v)}_{\text{Image}} = G(u,v) \times \underbrace{\frac{1}{H(u,v)}}_{\text{Inverse of } H(u,v)}$

Inverse Filtering Process

✡ NOISE MODELS. $(\eta(x,y))$

1) Gaussian Noise.

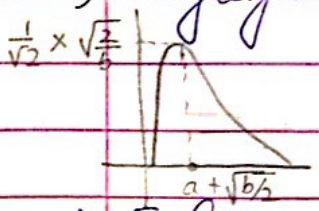
PDF of a Gaussian random variable:



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

z : Intensity.
$\bar{z}$: Mean
σ : Std. deviation
σ^2 : Variance of z

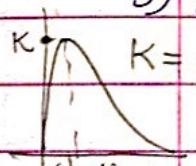
2) Rayleigh Noise



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

Mean, $\bar{z} = a + \sqrt{\pi b/4}$, Variance, $\sigma^2 = b(4-\pi)/4$

3) Erlang (Gamma) Noise

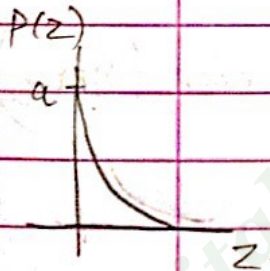


$$K = \frac{a(b-1)^{b-1}}{(b-1)!} e^{-(b-1)}$$

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & ; z \geq 0 \\ 0 & ; z < 0 \end{cases}$$

Mean $= \bar{z} = b/a$
Var. $= \sigma^2 = b/a^2$

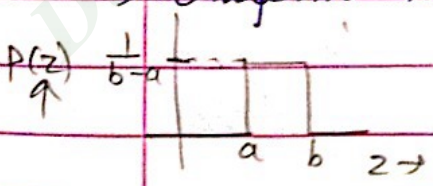
4) Exponential Noise (Special case of erlang noise ($b=1$))



$$p(z) = \begin{cases} ae^{-az} & ; z \geq 0 \\ 0 & ; z < 0 \end{cases}$$

$$\hookrightarrow \bar{z} = \frac{1}{a} = \text{Mean}, \quad \text{Var}, \sigma^2 = \frac{1}{a^2}$$

5) Uniform Noise



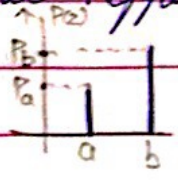
$$p(z) = \begin{cases} 1/b-a & ; a \leq z \leq b \\ 0 & , \text{elsewhere} \end{cases}$$

Mean, $\bar{z} = \frac{a+b}{2}$
Var, $\sigma^2 = \frac{(b-a)^2}{12}$

6) Impulse Noise or Salt and Pepper Noise.

as light & dark dots.

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

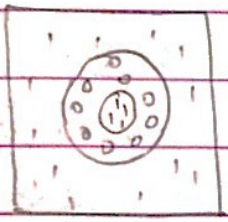


Page No

entire noise can take only 2 values → black dot (pepper) or white dot (salt)

7) Periodic Noise
↳ Noise like a mesh. Mesh is having a uniform shape distributed uniformly & its repeating periodically, rate of change is same.
↳ In freq. domain, it looks like :

→ these dots correspond to noise. To remove the dots, use a notch filter →



MATLAB Assignment.

---

* Restoration in the presence of external noise ($\eta(x,y)$)
↳ Only Spatial filtering.
ASSUMING $h(x,y) = 0$

↳ $g(x,y) = f(x,y) + \eta(x,y)$

$\Updownarrow$

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

✓ Mean filter :

1) Arithmetic mean filter (averaging)

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

↳ $S_{x,y}$: 3×3 neighbour of my pixel under consideration

2) Geometric mean filter :

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

→ Smoothening filter (just like arithmetic). But, loses less details

HW Do geometric mean of

2	3	6	7
8	10	12	13
4	6	7	8
4	7	7	8

3) Harmonic mean filter → salt noise, gaussian noise ✓ pepper noise ✗

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

4) Contraharmonic mean filter :

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

; Q : order of filter

$Q = 0$: arithmetic mean filter
$Q = -1$: harmonic mean filter.

✓ Order Statistics Filter.

1) Median filter & Also does smoothing operⁿ

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

2) Max and Min filter.

$$\hat{f}(x,y) = \underset{(s,t) \in S_{x,y}}{(\min) \text{ or } (\max)} \{ g(s,t) \}.$$

↳ useful in finding points & Brightest → max, Darkest → min in an image, due to which pepper → max, salt → min noise can be reduced.

3) Mid point filter :

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

↳ best for randomly distributed noise.

4) Alpha trimmed mean filter :
Suppose we delete $d/2$ lowest & $d/2$ highest intensity values of $g(s,t)$ in the neighbourhood $S_{x,y}$.

★ Notation :

MXN : Size of image

mxn : Size of neighbourhood (usually 3×3)

CLASSMATE

let $g_r(s,t)$ represents the minimum mn-d pixels. This is filter, formed by averaging these remaining pixels.

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

↳ d ranges from 0 to (mn − 1)

↓ Size of neighbourhood

eg. Consider :-

4	6	7	8
2	(3)	4	6
7	8	7	8
8	9	10	8

→ Consider element 3

Here, m = n = 3 (3×3 neighbourhood)

⇒ d varies from 0 to 8.

Arranging elements of neighbours in ascending order

4 6 7 2 3 4 7 8 7

→ 2 3 4 4 6 7 7 7 8

let d = 4

So, discard $\frac{d}{2} = 2$ pixel values from lower & higher.

So, 4 4 6 7 7

So, now take avg. of these values

$$\frac{4+4+6+7+7}{5} = 5.6 \simeq 6$$

Replace it with 3.

Page No

Note: d could vary from 0 to 8.
But, we chose 4. (= d).
For any problem, take one value of d & use it to replace

elements.

(1,1) 0 0 | 0 0 0 2 3 | 4 6

$= \frac{1}{5}$.

(1,2) 0 0 | 0 2 3 4 4 | 6 7

$\frac{13}{5} \simeq 3$.

(1,3) 0 0 0 | 3 4 6 6 | 7 8

$= \frac{19}{5} \simeq 4$

(1,4) 0 0 | 0 0 0 4 6 | 7 8

$= \frac{10}{5} = 2$.

(2,1) 0 0 0 | 2 3 4 6 | 7 8

$= 3$.

Illy, do for values:

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

* MATLAB function:-
"imnoise" : Adds noise.

CLASSMATE

* Note: For Alpha Trimmed Filter:

d = 0 : Arithmetic mean filter
d = mn - 1 : median filter.

▷ It is useful in noise, which is a combination of salt and pepper and Gaussian noise.

§ * Restoration in the presence of Degradation fn h(x,y)

Assuming $n(x,y) \approx 0$

So, now, we have:- $g(x,y) = f(x,y) * h(x,y)$

$\Updownarrow$

$G(u,v) = F(u,v) \cdot H(u,v)$

degraded image

* How to estimate $h(x,y)$ or $H(u,v)$

$F(u,v)$: we have this

$H(u,v)$: we want to estimate this

M1) Observation
M2) Experimentation
M3) Mathematical Modelling.

M1) Estimation by OBSERVATION.

Let image is degraded by unknown degradation fn H.

→ ≡ $h(x,y)$ or $H(u,v)$

Page No

just like: linear time invariant

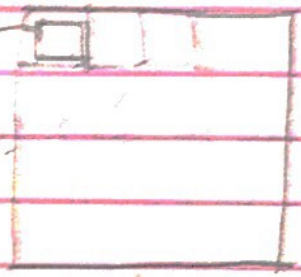
CLASSMATE

⇒

- Assume H is linear & pos^n invariant.
- Gather info. from image.

(1) ✱ Look at the small rectangular section of a blurred image which contain sample structure.

Consider a degraded image (eg blurred image)



We assumed $h(\alpha,\beta)$ is linear & pos^n invariant.

⇓

degradation happens uniformly everywhere

let 2 sections be $h_1(\alpha,\beta)$ & $h_2(\alpha,\beta)$

By linearity principle:

$$H(h_1(x,y)+h_2(x,y)) = H(h_1(x,y)) + H(h_2(x,y))$$

(2) ✱ To reduce noise effect

⇓

look at strong signal content area (area of high contrast)

→ i.e region having all intensity values.

(3) ✱ Now, process this sub-image (small portion) with whatever oper^{ns} known so as to get an unblurred image — Doing image correction.

(So, we had $g(x,y)$ & on applying oper^{ns} we are getting close to $f(x,y)$)

let observed sub-image be denoted by $g_s(x,y)$ & processed sub-image $f_s(x,y)$

So, $g_s(x,y) = f_s(x,y) * h_s(x,y)$

$\Rightarrow$ $G_s(u,v) = F_s(u,v) \cdot H_s(u,v)$

$G_s(u,v)$: known

$F_s(u,v)$: known (by applying deblurring)

$H_s(u,v)$: ?

So,

$$H_s(u,v) = \frac{G_s(u,v)}{F_s(u,v)}$$; (Assuming negligible noise effect)

Doing inverse, we get $h_s(x,y)$

As $h(x,y)$ was assumed to be pos^n invariant, this is my $h(x,y)$ to be applied to all.

eg Suppose radial plot of $H_s(u,v)$ has approx. shape of gaussian curve, same info. is used for $H(u,v)$ in larger scale.

M2 Estimation by EXPERIMENTATION

↳ requires equipment

* Imagine we took a degraded image using camera
* If we have the equipment with which we took the degraded image, accurate estim^n of degraded fn can be got ..

1D ≡ 2D

Impulse response — Point spread fn

CLASSMATE

Its based on the idea of impulse response.

Impulse response :- The output of a filter when i/p is impulse.

$\delta(n) \rightarrow [\ \ ] \rightarrow h(n)$

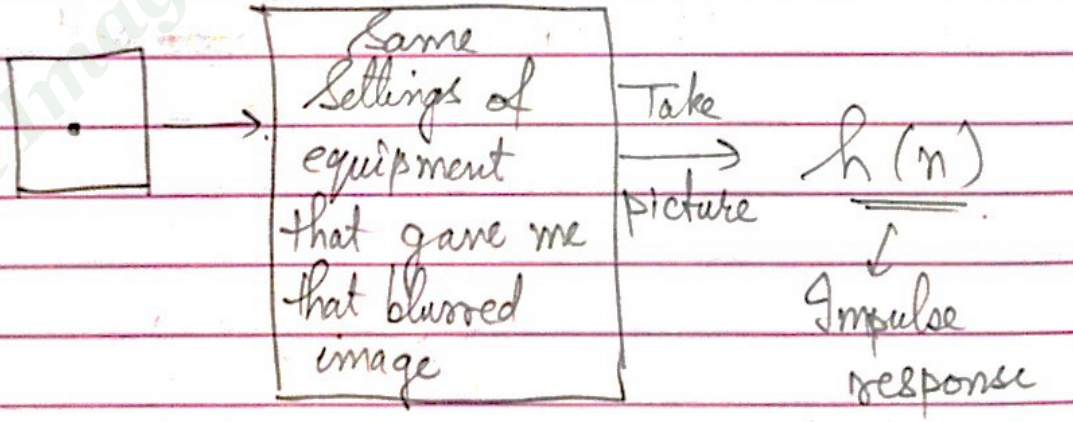
So, our equipment got us $g(x,y)$.

Now, I have my camera (equipment)

Using the Settings of the equipment in such
↳ By changing Settings of equipment

a way that we get the image that is as close as to our degraded image.

Now we apply an impulse to my equipment.
↳ in 2D, its a black image with a White dot in center.



* An impulse is simulated by bright dot of light as bright as possible to reduce the effect of noise.

So, Fourier transform of an impulse is a constt.

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

↳ A : A constt. describing strength of impulse

M3) Estimation by MODELING.

- It's a mathematical modeling.
- $h(x,y)$ is given by a mathematical fn
  - ↳ eg. mathematical model for atmospheric disturbances.

Idea : If we couldn't get $h(x,y)$ from M1 & M2, then use a ready made $h(x,y)$ fn for atmospheric turbulence given by Hufnagel & Stanley.

$$H(u,v) = e^{-k(u^2+v^2)^{5/6}}$$

↳ k : constt, depending upon nature of turbulence

MATLAB Assignment : Take an image & degrade it using convolution with $h(x,y)$ (= Inverse(H(u,v))) with various values of k.

Now, instead of seeing $h(x,y)$ by atmospheric disturbances, lets make another $h(x,y)$

eg: Image gets blurred on acquisition when taken as uniform linear motion b/w target & sensor.

how to make?

① Let $f(x,y)$ be the image

② As the picture is moving, that means the pixels are moving. So, the coordinates $x_0(t)$ & $y_0(t)$ are changing in x & y dir^n w.r.t time

(Same idea comes in Motion Blur)

③ See total exposure (total time taken to acquire the image) → integrate the instt. exposure

$$g(x,y) = \int_0^T f[x - x_0(t), y - y_0(t)]$$

Total time shutter opens

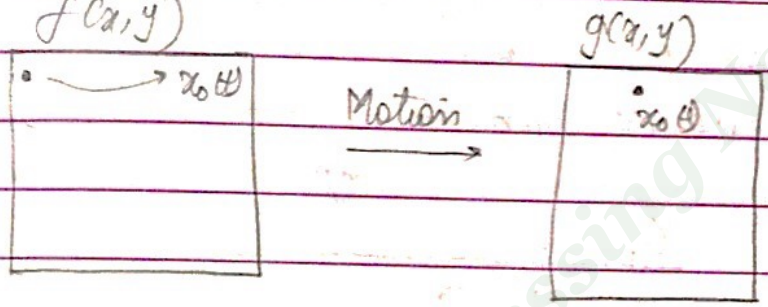
→ $g(x,y)$ is blurred image.

Doing FT of (1) (↙)

$$G_1(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\, e^{-j2\pi(ux+vy)}\, dx\,dy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[\int_0^T f[x-x_0(t), y-y_0(t)]\,dt\right] \times e^{-j2\pi(ux+vy)}\, dx\,dy$$

→ Note: It's NOT discrete FT



Note: our pixel was moved to a new loc^n by degrad^n fn H(u,v) Now, estimate value of H(u,v) to get original image back.

Re ordering ↘

$$G_1(u,v) = \int_0^T \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[x-x_0(t), y-y_0(t)]\, e^{-j2\pi(ux+vy)}\, dx\,dy\right] dt$$

→ Spatial domain shifting

→ FT of shifted $f(x,y)$

$\equiv F(u,v)\, e^{-j2\pi(ux_0(t)+vy_0(t))}$

$$\Rightarrow G(u,v) = \int_0^T F(u,v)\, e^{-j2\pi[ux_0(t)+vy_0(t)]}\, dt$$

→ Notice: $F(u,v)$ is independent of t

$$\Rightarrow G(u,v) = F(u,v) \int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))}\, dt$$

Also,

$$G(u,v) = F(u,v) \cdot H(u,v) + N(u,v)$$ (N(u,v) → 0, assumed)

$\Rightarrow$ Comparing

$$H(u,v) = \int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))}\, dt$$

with $x_0(t)$, $y_0(t)$ known $H(u,v)$ is got.

($x_0(t)$, $y_0(t)$ — speed of pixel motion)

Now, we want original image $f(x,y)$:

$$F(u,v) = \frac{G(u,v)}{\int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))}\, dt}$$

Doing inverse FT

$$\Rightarrow f(x,y) = F^{-1}\left[\frac{G(u,v)}{\int_0^T e^{-j2\pi(\,)}\, dt}\right]$$

↳ not exactly got back.

Q. Suppose the image undergoes uniform linear motion in x-dir^n only @

$$x_0(t) = \frac{at}{T}$$

↳ when $t = T$, image has been displaced by distance a

let

$$H(u,v) = \int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))} dt$$

$$= \int_0^T e^{-j2\pi u x_0(t)} dt$$

$$= \int_0^T e^{-j2\pi u at/T} dt$$

$$\Rightarrow H(u,v) = \frac{T}{\pi ua} \sin(\pi ua)\, e^{-j\pi ua}$$

(sinc function) (after solving (sell))

Part b) : In the question, if we take both dir^ns :-

$H(u,v)$ is got as :-

$$H(u,v) = \frac{T}{\pi(ua+vb)} \sin[\pi(ua+vb)] e^{-j\pi(ua+vb)}$$

→ assuming $y_0(t) = \frac{bt}{T}$

* When we cannot identify the type of degradation, we use statistical models.

6.4.14

Ch - Image Segmentation

→ Segmenting part of image.
I want to identify shapes in an image, to segment the image.
↳ we deal with segmentation using computer.
↳ after segmentⁿ operⁿ, we get binary image.

- Let R be the entire spatial region occupied by an image. R is a set containing all shapes.
Process that partitions R into n sub regions $R_1, R_2, \ldots R_n$,

s.t ① $\bigcup_{i=1}^{n} R_i = R$ (Segmentation must be completed)

union

② R_i is a connected set ($i = 1, 2, \ldots n$)
(Pts. in region must be connected in predefined sense)

③ $R_i \cap R_j = \phi$ $\forall$ i, j & $i \neq j$
(Regions must be disjoint)

intersection

④ $Q(R_i)$ = True, for $i = 1, 2, \ldots n$
↳ ∀ pixels is satisfied.

⑤ $Q(R_i \cup R_j)$ = False for any adjacent regions R_i & R_j
↳ 2 adjacent regions must be diff in the sense predicate Q.

✡ 2 APPROACHES on Segmentation

algorithm approach

Discontinuity based

Identific^n of
- Isolated points
- Edges
- Lines

Similarity based

(prev. page talked about it).
- Threshold oper^n
- Region growing
- Region Splitting & Merging

* Detection of isolated points.

We saw: laplacian operator, gradient operators
↳ identifies edges ⇒ identifies regions

↳ $\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$; Mask

0	1	0
1	-4	1
0	1	0

↳ only horizontal & vertical.

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

1	1	1
1	-8	1
1	1	1

↳ also including diagonals

eg.

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

→

4	7	6
2	5	4
1	1	2

×

1	1	1
1	-8	1
1	1	1

→ Multiply coinciding elements and add.

we get Sum = -13

On applying Laplacian, we still didn't get isolated points. So, apply THRESHOLD
↳ fix one value & compare other values to that.

If absolute value (→ +ve value) of sum is greater than some positive threshold value, then, that value is set to 1 (else, zero).
↳ Proceeding like this, we'll only have 1's & 0's in the end. Hence, 1 gives edges
↳ Set a threshold ⇒ usually equal to 90% of max. pixel values

Hence, thresholding oper^n gives binary image.

eg: If we want to detect isolated point:

Image → Laplacian Operator → Thresholding oper^n

★ Detection of LINES

↳ use of Laplacian mask
↳ detects the discontinuity (lines) in areas of interest

MATLAB

To detect

Horizontal	+45°	Vertical dir^n	−45°
−1 −1 −1	2 −1 −1	−1 2 −1	−1 −1 2
2 2 2	−1 2 −1	−1 2 −1	−1 2 −1
−1 −1 −1	−1 −1 2	−1 2 −1	2 −1 −1

Image →	Laplacian operator applied :
	Horizontal → Response, R_1
	$+45^\circ$ → Response, R_2
	Vertical → R_3
	-45° → R_4

Suppose the image that we get has the following trend :

$R_1 > R_2 > R_3 > R_4$

So, it has more horizontal component & min component in -45° dir^n.

* Note : Constant intensity area gives zero response.

2	2	2	Laplacian →	0	0	0
2	2	2		0	0	0
2	2	2		0	0	0

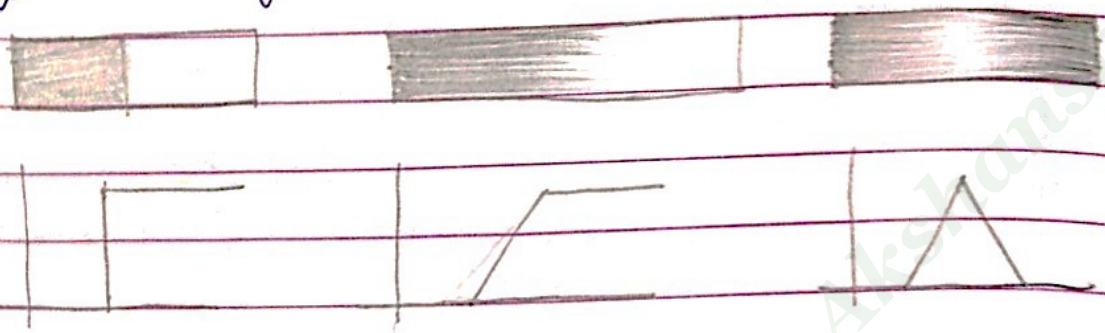
* After getting Laplacian we can choose to take values of the pixels :
  - ✓ Magnitude (both +ve & -ve : as it is)
  - ✓ Absolute (make everything +ve)
  - ✓ +ve (take +ve ; -ve made 0)

* If (all) masks (on prev. page) for laplacian are used and added ($I_1 + I_2 + I_3 + I_4$), we get Laplacian in all dir^{ns}.

Finally, thresholding can be applied

✱ Detection of EDGES

Types of Edges:



We need an algorithm so that any type of edge can be detected.

Way of detection : Gradient, or differential operator & Laplacian operator.

Detects presence of edge

← 1st Derivative → Sobel operator (1st Derivative)

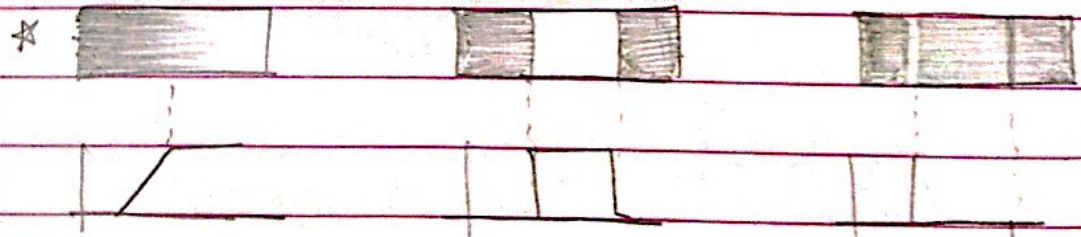
(2nd Derivative)

gives 2 values. The sign of value can be seen to know if it is from brighter or darker side or vice versa

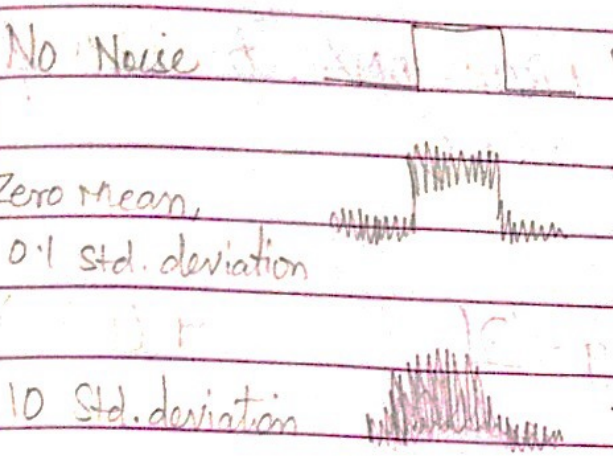
← 2nd Derivative

Zero crossing (Used to locate centre of thick edges)

→ This is the thin edge of the image.



✱ Use smoothening operator in case of noise.



Noise can come
Has to be removed / reduced
↳ smoothen image. Then apply edge detection

• We know, edges can be detected using 1st order or 2nd order derivatives.
→ Laplacian Operator
Gradient

* First Order Derivative for Edge detection
Finding edge strength & dir^n at a loc^n (x,y) in an image f

Gradient
eg. using sobel operator

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

vector

↳ Choose any pixel. Apply gradient operator on it. Find its STRENGTH & DIRECTION
Gives/Tells if its edge/no

→ points in dir^n of greatest rate of change of f at loc^n (x,y)

Strength i.e magnitude (length) of vector ∇f,

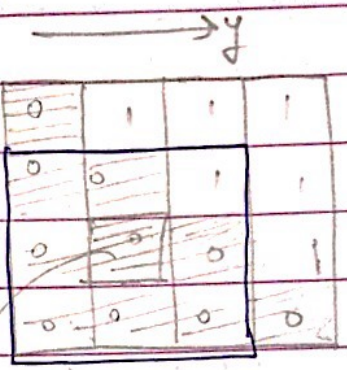
$$M(x,y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \quad (\text{or } |g_x| + |g_y|)$$

Direction of gradient vector

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_x}{g_y}\right]$$ ↘ w.r.t x-axis

* Dir^n of edge is ORTHOGONAL to dir^n of gradient vector, $\alpha(x,y)$.

eg.



pixel of interest

$g_x = \frac{\partial f}{\partial x} = (0-0)+(0-0)+(0-1) = -1$

$g_y = \frac{\partial f}{\partial y} = (1-0)+(0-0)+(0-0)$

(Subtracting rightmost pixels from leftmost column of neighbourhood)

Apply Gradient oper^n $\frac{\partial f}{\partial y}$ using Sobel operator (or do gradient oper^n directly)

g_x

-1	-1	-1
0	0	0
1	1	1

$\frac{\partial f}{\partial x}$

(Subtract pixels in top row of neighbourhood from bottom row)

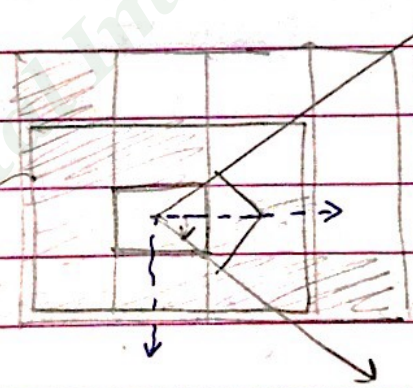
$\therefore g_x = -1, g_y = 1$.

Now,

$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

$\therefore M(x,y) = \sqrt{1^2 + (-1)^2} = \sqrt{2}$

$\alpha(x,y) = \tan^{-1}\left(\frac{-1}{1}\right) = -45°$ or $135°$.



Orthogonal to edge direction = Direction of gradient operator

$-45°$ or $135°$. If is Edge Direction

* If $M(x,y)$ is very high, its edge pixel.

If $M(x,y) >$ Threshold, its Edge pixel.

(Threshold) decided by us.

CLASSMATE

* Note: General form of partial derivative

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \equiv \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y-1) - f(x,y) \equiv \begin{bmatrix} -1 & 1 \end{bmatrix}$$

For 2D image, use Roberts ~~cross~~ gradient operator (X) (not much used)
Prewitt Operator ✓
Sobel Operator ✓
→ applying weight also in Prewitt Operator

Prewitt:

$\frac{\partial f}{\partial x}$ Along x

-1	-1	-1
0	0	0
1	1	1

(Why for g_y)

Sobel:

-1	-2	-1
0	0	0
1	2	1

- Other Operators like modified Prewitt & Sobel operators also exist.

↳ continued after Homomorphic filtering

13/4/14

CLASSMATE

HOMOMORPHIC FILTERING

MatLab Assignment

An image can be represented as $f(x,y) = i(x,y)\, r(x,y)$

(i: Incidence, r: Reflectance)

* Note: Freq components of reflectance & illumin^n component is not possible through FT

i.e

$$\mathcal{F}\{f(x,y)\} \neq \mathcal{F}(i(x,y)) \cdot \mathcal{F}(r(x,y))$$

rather,

$$\mathcal{F}\{f(x,y)\} = \mathcal{F}(i(x,y)) * \mathcal{F}(r(x,y))$$

→ can be made possible using Homomorphic filtering

* If we have an image $f(x,y)$ s.t either one of $i(x,y)$ or $r(x,y)$ is improper. Now, if we can separate $i(x,y)$ & $r(x,y)$, then we can correct that component (filtering it) $i(x,y)$.

Homomorphic filtering

↳ using logarithmic approach.

Lets define $z(x,y) = \ln\{f(x,y)\}$

$$= \ln\{i(x,y)\} + \ln\{r(x,y)\}$$

Then, $\mathcal{F}\{z(x,y)\} = \mathcal{F}\{\ln(i(x,y))\} + \mathcal{F}\{\ln[r(x,y)]\}$

i.e

$$Z(u,v) = F_i(u,v) + F_r(u,v)$$

components separated.

Page No

Now, we can use the required filter to correct the components & hence, the image ($z(x,y)$ or $Z(u,v)$).
Using filter $H(u,v)$ on $Z(u,v)$
So, we have

$$S(u,v) = Z(u,v) \cdot H(u,v) \quad = o/p$$
$$= H(u,v) \cdot F_i(u,v) + H(u,v) F_r(u,v)$$

So, in spatial domain

$$s(x,y) = \mathcal{F}^{-1}[S(u,v)] = \underbrace{\mathcal{F}^{-1}[H(u,v) F_i(u,v)]}_{i'(x,y)} + \underbrace{\mathcal{F}^{-1}[H(u,v) \cdot F_r(u,v)]}_{r'(x,y)}$$

Filtered incidence component

So, $i'(x,y) = \mathcal{F}^{-1}[H(u,v) F_i(u,v)]$

$r'(x,y) = \mathcal{F}^{-1}[H(u,v) F_r(u,v)]$

$\Rightarrow s(x,y) = i'(x,y) + r'(x,y)$

* Note: We took log. Now, in the end do antilog to get image (components) back.

So, taking inverse (exponential)

Let, $g(x,y) = e^{s(x,y)}$

$\Rightarrow g(x,y) = e^{s(x,y)} = \underbrace{e^{i(x,y)}}_{i_0(x,y)} \cdot \underbrace{e^{r(x,y)}}_{r_0(x,y)}$

$$\Rightarrow \boxed{g(x,y) = i_0(x,y)\, r_0(x,y)}$$

- $g(x,y)$: Filtered image
- Homomorphic filtering is used.
- $i_0(x,y)$: Illumination component
- $r_0(x,y)$: Reflectance component.

Block diagram :

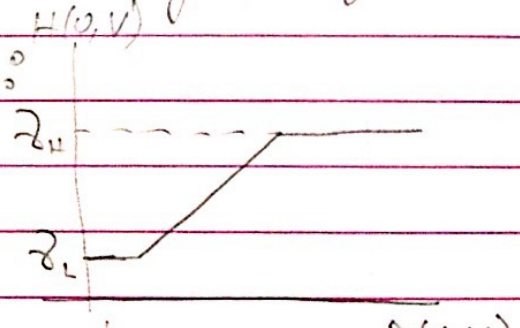
$f(x,y)$ → [ln] → [DFT] → [H(u,v)] → [DFT^{-1}] → [exp] → $g(x,y)$

Note :

* Illumin^{n} component should have less variation, or very slowly varying ⇒ Less freq.
* Reflectance component should have the ability to show all variations, varying abruptly, particularly at junction of dissimilar objects ⇒ High freq.

So, we need ° Illumin^{n} component → low pass filter
a filter ° Reflectance component → High pass filter

→ So, if H(u,v) is like :



↳ $\gamma_L < 1$, $\gamma_H > 1$

⇒ low freq components are lowered & high freq components are highered

✡ MARR-HILDRETH EDGE DETECTOR.

Previous methods in image segmentation (Gradient, laplacian operators) don't make use of char. of image. We didn't bother about how the image was like, if it had noise, anything else)

→ This methods addresses this issue

* Done by development of operators which changes its size based on blurry images & sharply focused fine details.
* Makes use of laplacian, Gradient & Smoothing
* Operator used :-

$$\nabla^2 G$$

→ ∇^2 : Laplacian

Note : $\nabla^2 G$ (not $\nabla^2 F$)

So, laplacian not being applied to image

* G = Gaussian function,

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

→ σ : Std. deviation

So, expression for $\nabla^2 G(x,y)$

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

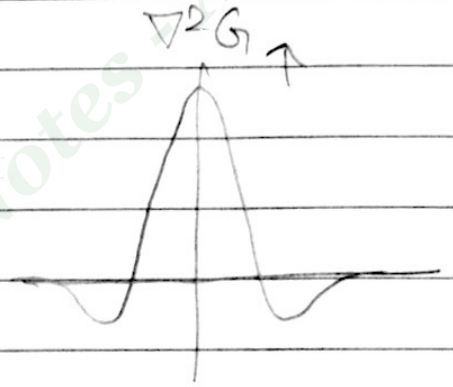
Solving

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

→ Laplacian of Gaussian Operator (LoG)

Pixel values of filter :-

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	-16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



So, we have :- $g(x,y) = [\nabla^2 G(x,y)] * f(x,y)$

27/4/14

CLASSMATE

✡ EDGE LINKING & BOUNDARY DETECTION

Edge detection operators detect edges → dont link them. These edges are not linked to each other. Linking edges gives a boundary.

* Edge linking ≡ assemble edge pixels into meaningful boundaries.
  * edge detection typically followed by edge linking.
  → 2 types : LOCAL & GLOBAL.

(M1) * Local Processing

Analyse char of pixels in small neighbourhood (3x3, 5x5 --) about every point that has undergone edge detection.

All similar points are linked - forming a boundary.

*2 Properties used to link edges

P1) Strength of response of gradient operator; used to produce edge pixel

≡ Magnitude → $\sqrt{g_x^2 + g_y^2}$ or $|g_x| + |g_y|$

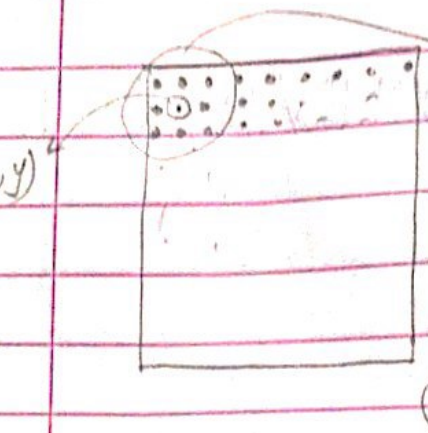
→ commonly used : Sobel operator.

P2) Direction of gradient → $\tan^{-1}\left(\frac{g_y}{g_x}\right)$

What to do?

A pt. in the predefined neighbourhood of (x, y) is linked to a pixel (x, y) if both magnitude & dir^n criteria are satisfied. This process is repeated for every loc^n in image.

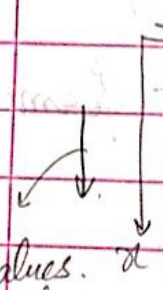
Page No



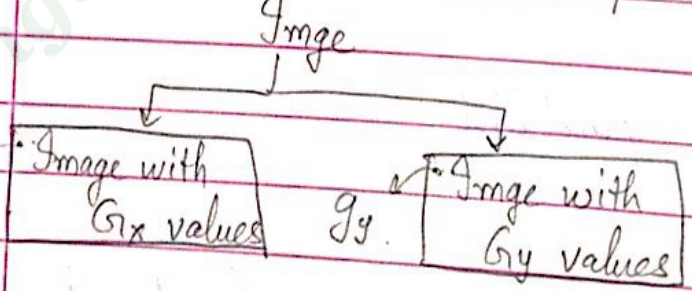
The central pixel is related to any neighbours : Rel^n seen by gradient operator

Suppose we have a set of magnitudes & directions :
$S = \{S_1, S_2, S_3 \ldots\}$ & $\alpha = \{\alpha_1, \alpha_2 \ldots\}$.
Choose any neighbourhood pixel of (x, y). If the magnitude & dir^n match with any value in the set, then it can be linked.

Idea : If I have an image, apply sobel operator
in horizontal dir^n ⟶ ① ⇒ we get g_y values
& in vertical dir^n ⟶ ② ⇒ we get g_x values



* Link the points if :
① . Gradient value > 25.
& ② . Gradient dir^{ns} do not differ by more than 15°.
→ If these 2 cond^{ns} are satisfied, replace it with 1 (else, 0)



* First point in image is called SEED POINT.

Link if $|g_x| + |g_y| > 25$

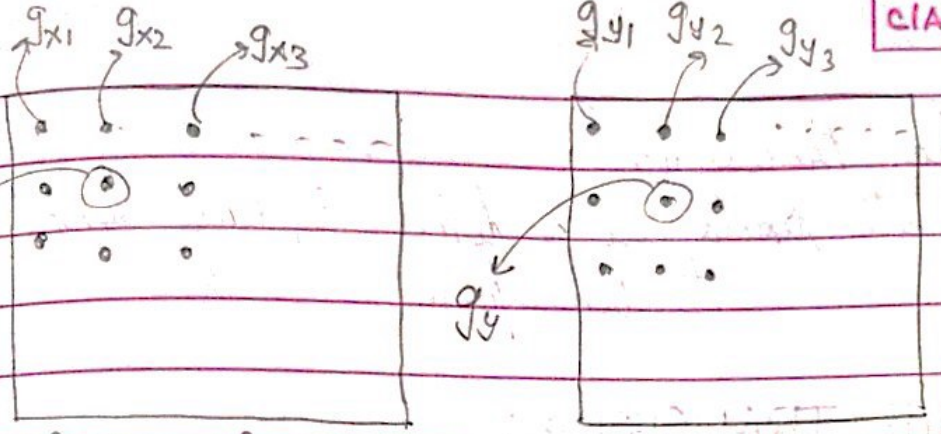


Image with g_x values. Image with g_y values.

① g_{x1}, g_{y1} : seed pixel
↳ $|g_{x1}| + |g_{y1}| > 25$.

② Now, on taking g_{x2} & g_{y2}
↳ $|g_{x2}| + |g_{y2}| > 25$
& $\left| \tan^{-1}\left(\frac{g_{y2}}{g_{x2}}\right) - \tan^{-1}\left(\frac{g_{y1}}{g_{x1}}\right) \right| < 15^\circ$

③ on taking g_{x3} & g_{y3}
↳ $|g_{x3}| + |g_{y3}| > 25$
& $\left| \tan^{-1}\left(\frac{g_{y3}}{g_{x3}}\right) - \tan^{-1}\left(\frac{g_{y2}}{g_{x2}}\right) \right| < 15^\circ$

↓
Ily proceed ∀ neighbours of pixel (x, y)

* Every pixel in the neighbourhood of (x, y) are checked for magnitude & angle to decide linking.

* CCS : Cartesian Coordinate Sys.

CLASSMATE

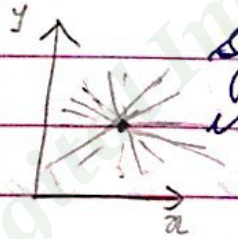
M2 * Global Processing
Take all pixels at a time (don't use neighbours — 3×3, 5×5 ....) to transform.

§ • HOUGH TRANSFORM:
Technique used to isolate features of a particular shape within an image.
- used for detecting lines, circles, ellipses ....etc
- Main advantage : It is tolerant of gaps in feature boundary descriptions & is relatively unaffected by noise.

(A) ✱ Detection of lines through Hough Transform :

Mapping a st. line $y = mx+c$ in Cartesian coordinate sys. into a single point in the (ρ, θ) plane
or $\rho = x\cos\theta + y\sin\theta$
(i.e basically, cartesian ⟶# polar)

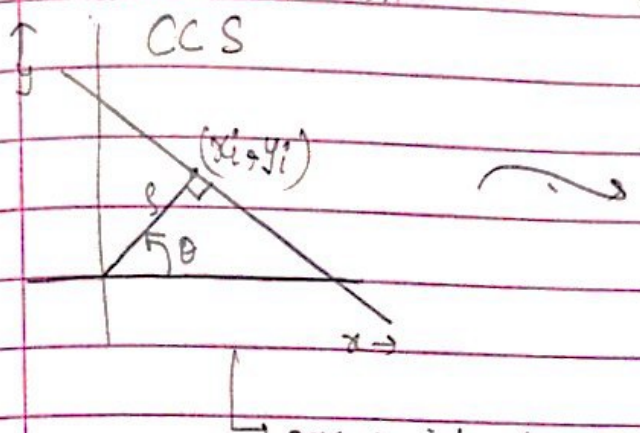
* For every point (x,y) in Cartesian coordinate ~~sys~~ plane, ∃ INFINITE NO. OF CURVES in (ρ,θ) plane.



(idea : consider a point on any plane (CCS), ∃ infinite lines passing through that point)
(1 line gives one curve ⟶ ∞ lines ⟶ ∞ curves)
(One pt. x,y ⟶ ∃ ∞ values of ρ & θ)

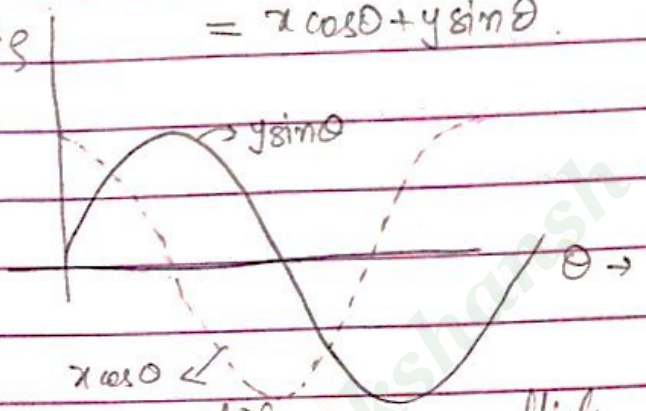
CLASSMATE

Consider a line in CCS



(ρ, θ) parametric plane.

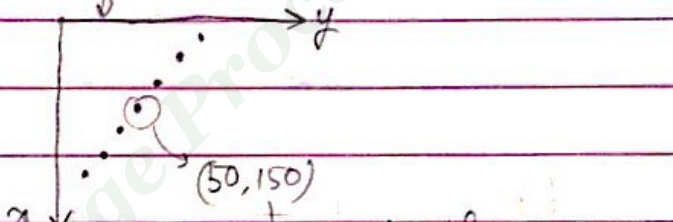
$= x\cos\theta + y\sin\theta$.



↳ one point gives one curve ⇒ multiple pts gives multiple curves

Steps: * Given an image → detect its edges (using various edge detection operators) → we get binary image after detecting edges → Hough transform is applied on this Binary image ((x, y) values of binary image are used)

↳ eg: for an image (after edge detection)



↳ using this pt. (x, y) find ρ & θ values

x	y	θ	ρ

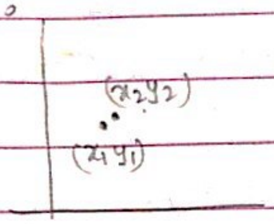
* We make table.

* Note: ~~If~~ One pt. on CCS gives one curve in (ρ, θ) plane.

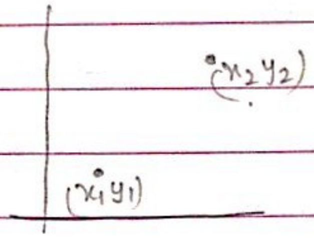
If the pts. on CCS lie on one line, we get intersection of curves at one pt. in (ρ, θ) plane. (ρ, θ plane)

eg: If 100 curves intersect at 1 pt, ⇒ ∃ 100 points one one st. line (CCS)

Idea :



: 2 Close pts gives 2 curves in (ρ, θ). These 2 curves will intersect

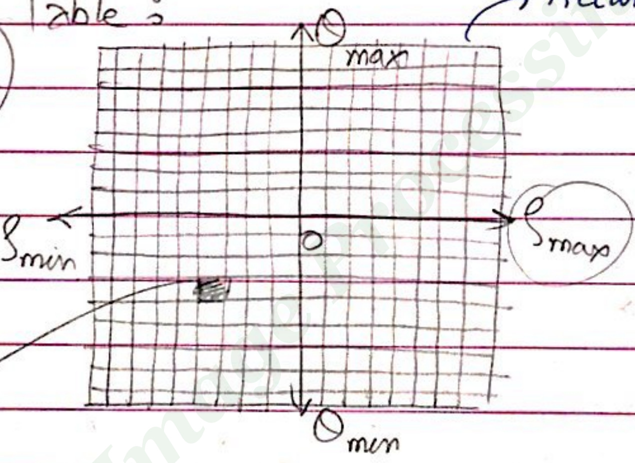


: 2 pts. (faraway) gives 2 curves in (ρ, θ) plane. These 2 curves will ALSO intersect.

* These curve intersections are useful to know more about the lines in CCS.

(Detect lines MATLAB Assignment)

Table : → Accumulator array (2D)



Take every pt. in CCS (x, y) & vary θ from θ_{min} to θ_{max}. Corresponding value of ρ are got. This value corresponds to a box on the plane. Put "1" in that box. Suppose another curve (for some other pt (x_2, y_2)) has the same value of ρ, then, that box's value is increased from 1 to 2.

If the value = 100
⇒ 100 curves are intersecting
⇓
the 100 points are in the same line (x-y plane)

* $\rho_{min}, \rho_{max}, \theta_{min}$ & θ_{max} are known values

Imp * Each box value basically tells the count or the number of curve intersections.

* All cells have to be initialized to zero.

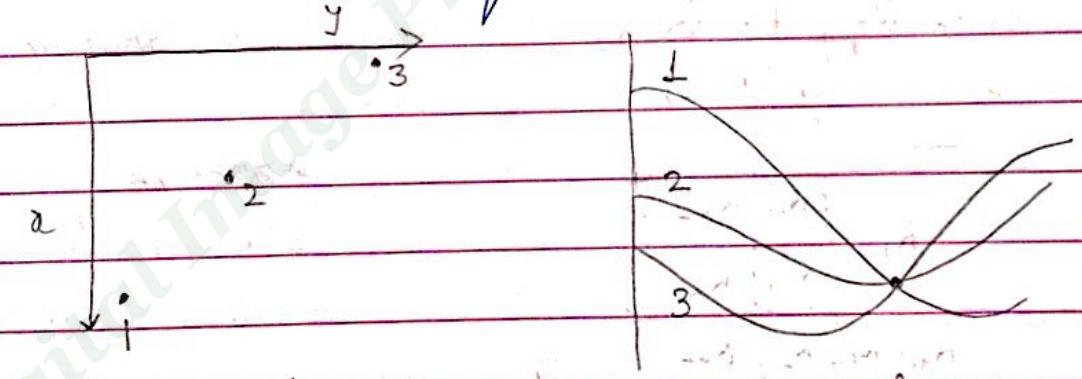
Pt (x_i, y_i) ⟶ values of ρ_i for various values of θ_i
$(\rho_i = x\cos\theta_i + y\sin\theta_i)$

↳ If ∃ * N collinear pts lying on the line, we get N sinusoidal curves intersecting at one pt. (ρ_i, θ_i).

(B) Detection of circles through Hough transform:

CCS: eqⁿ: $(x-a)^2 + (y-b)^2 = r^2$.

↳ ⇒ we get 3D accumulator array (for parameters x, y & r).



X-y plane (3 pts. in one line)

ρ-θ plane (3 curves intersect at one point)

* Once we identify max. no. of intersections, we know corresponding x & y values ⟶ ⇒ make a line.

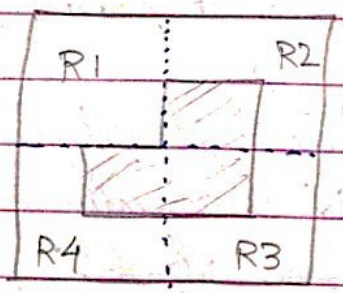
30.4.14

CLASSMATE

✱ IMAGE SPLITTING & MERGING ALGORITHM

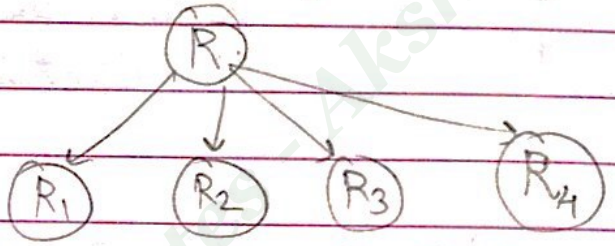
REGION

Consider an image :



Using this algorithm, I want to detect "⌐" shape

S1) Split the image in 4 regions

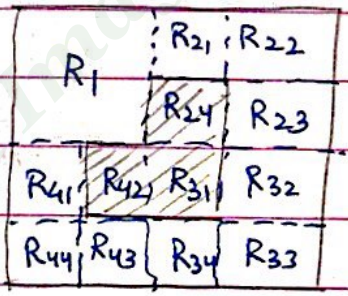


S2) Merge : Merging is done on similarity criterion.

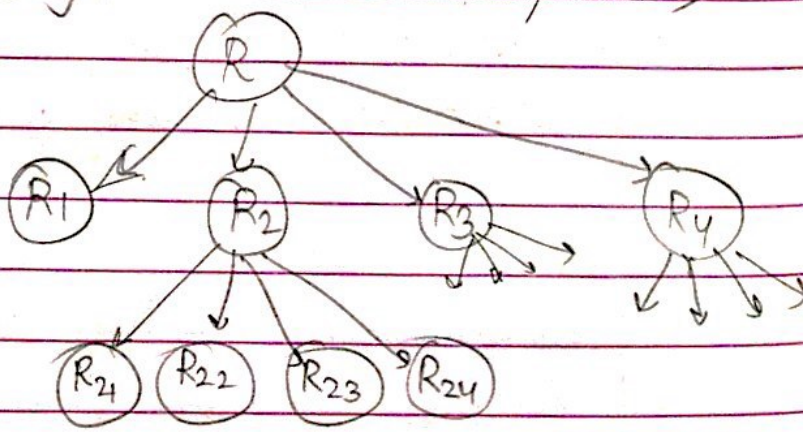
If pixels in entire region belongs to one set (i.e, that region has everything constant), merge them like R_1 has every pixel similar (no predicates) so, it can be merged.

But R_2, R_3 & R_4 are not constt. throughout so, again split them into 4 regions

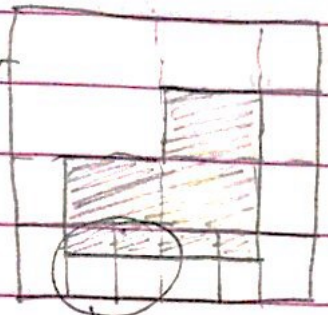
sub-region



Now, each part can be merged into separate set. (R_{21}, R_{22} ... they dont need to be splitted)



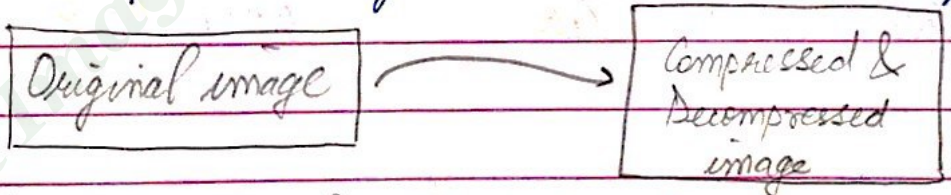
eg :-



→ further splitting is req^d. if non similar.

IMAGE COMPRESSION

* Consider 2024x2024 size color image. Each pixel is represented by 24 bits. So, data size = 2024x2024x24
* Image Compression is all about reducing data size WITHOUT losing the info ⇒ Remove REDUNDANT DATA
  • Process of removing redundancy is compression.



* If its not image & just data, then, on compressing we are removing some data ⇒ LOSSY Compression
* If its image & we get back same image after compression & decompression ⇒ LOSSLESS Compression
* Compression ratio : LOSSY > LOSSLESS

Applying Huffman Algorithm

Consider a 4×4, 3 bit image

If no compression, no. of bits req to store image = 4×4×3 = 48 b
(i.e Symbol for 0, S_0 = 000, S_1 = 001, ... S_7 = 111)

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

Average length = Lavg = 3

	$P_k(k_x)$
S_0 = 000	2/16
S_1 = 001	2/16
S_2 = 010	4/16
S_3 = 011	3/16
S_4 = 100	2/16
S_5 = 101	1/16
S_6 = 110	1/16
S_7 = 111	1/16

Lavg = 3 b

(arrange in decreasing prob)

S_2	010 = 4/16	01	4/16	01	4/16	01	4/16	01	→ 5/16	00	→ 7/16	1	→ 9/16	0	
S_3	011 = 3/16	11	3/16	11	3/16	11	→ 4/16	10	→ 4/16	01	5/16	00	→ 7/16	1	
S_0	000 2/16	001	2/16	001	→ 3/16	000	→ 3/16	11	→ 4/16	10	4/16	01			
S_1	001 2/16	100	2/16	100	→ 2/16	001	→ 3/16	000	→ 3/16	11					
S_4	100 2/16	101	2/16	101	→ 2/16	100	→ 2/16	001							
S_5	101 1/16	0001	→ 2/16	0000	→ 2/16	101									
S_6	110 1/16	0000	→ 1/16	0001											
S_7	111 1/16	0001													

length of S_2

Now, Lavg = 2(4/16) + 2(3/16) + 3(2/16) + 3(2/16) + 3(2/16) + 4(1/16) + 5(1/16) + 5(1/16)

Lavg = 2.875

Now,
No. of bits req to represent image = Σ (no. of bits to represent any symbol) × (no. of times it occurs in image)

= 2×4 + 2×3 + 3×2 + 3×2 + 3×2 + 4×1 + 5×1 + 5×1

= 46 bits

- Why do Image Compression?
  - ✓ Rapid growth in comp. power.
  - ✓ Everything is digital in today's world. Transfer of digital information takes place. As channel width is fixed (BW limited), data compression is required.
  - ✓ To meet storage requirement in computer databases.
- Reducing size of data, retaining necessary info. is called Image Compression.

Nomenclature:

- Original image: Uncompressed image.
  - .bmp is uncompressed image
  - .jpg file is compressed, decompressed image → Stored in browser. On viewing decompression algorithm gives our data.

* Compression ratio = $\frac{\text{Uncompressed file size}}{\text{Compressed file size}}$.
  = Size u / Size c
  = Size u : Size c
* Bits per pixel = $\frac{\text{No. of bits}}{\text{No. of pixels}}$

eg. Original image = 256×256, 8 bits per pixel
After compression, = 6554 bits
⇒ Compression ratio = $\frac{256 \times 256 \times 8}{6554} = 10:1$

eg Transmission time eg

Time taken to transmit :

① Single 512×512×8 bit image.

② Transmit a color image 512×512 (RGB) via modem at 28.8 Kbaud (Kilobits/sec)

① Storage req^d = 512×512×8.
= 2097152 b.

* Compression Fundamentals :-

Entropy ≡ how much info. in your data.

Data : pixel gray level values that correspond to brightness of a pixel at a pt. in space

Info : interpret^n of data in meaningful way

eg : binary image of text only – info is text being readable

eg ② : medical image : info. is every minute detail of image (original)

* Compression types :

① Lossless Compression

∃ no much loss b/w original & compressed data i.e, original can be recreated exactly. Typical compression ratios → 2:1, 3:1 (can be higher with text only image)

② Lossy

Loss in actual data of image occurs. So, cannot be recreated exactly from compressed file.

Compression ratio 10:1, 20:1 ---

* How to detect:

If Pixel to pixel are compared b/w original & compressed image, we see the error

No error : lossless compression.

∃ error : lossy compression

* How to achieve compression.

↳ Reduce data size → REDUNDANCY

(all data may not be req^d)

** Redundancy

Coding	Interpixel	Psychovisual
like, in Hoffman Coding	like, in run length coding	used in quantiz^n. ↳ exact data cannot be recovered (same sequence is not got back).
	If ∃ 100 1's continuously, instead of writing 111111 --- 1111 (100 bits), write 1, 100. (say 8 bits needed) (8 bits needed).	Its done till the amount human eye cannot detect (psychovisual extent of human eye).

CLASSMATE

* Interpixel redundancy

eg: Consider

250 251 252 253 252 256 255 = 8 × 7 bits

keep 1st as it is → 250

251 − 250 → 1

1 1 −1 4 −1 ...

9 bits → 8 + 1 (sign)

4 → 3 + 1 signed bit

4 4 4 4 4 bits

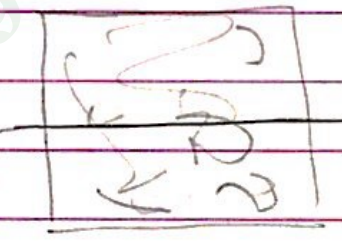
= 9 + (4 × 6) = 33 bits

So, interpixel redundancy has been removed & no. of pixels got reduced from 56 to 33

* To reduce size of image we can sometimes convert it from grayscale to binary image (by Thresholding)

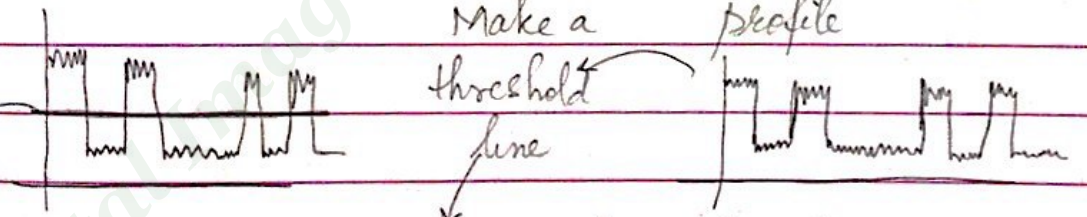
RUN LENGTH CODING example

→ Suppose an image is:



take one line

draw its profile



All points above that line = 1
below that line = 0.

If the codes (in binary) of the line are

1111111 (63 times) 00000 (87 times) 11111 (37 times) → requires 63+87+37 = 187 bits

So, if we write (1, 63)(0, 87)(1, 37). → requires (1×3) + (10×3) = 33 bits

no. of times 0 comes, no. of times 1 comes (for 1 line)

1 bit

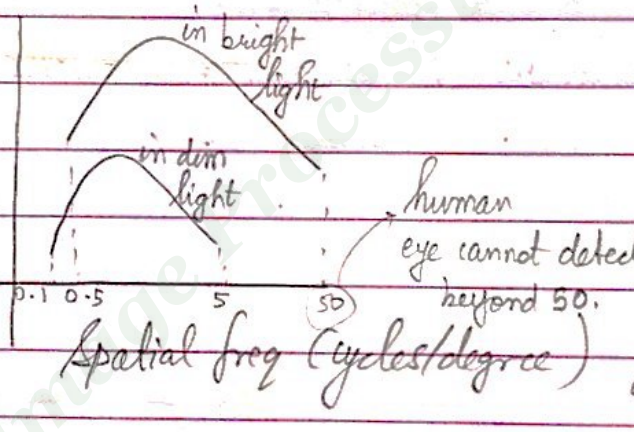
say, we represent using 10 bits

* Cycles/degree → ⇒ data of image is changing in what dir^{n}

* Usually, coding redundancy is done at the last stage.

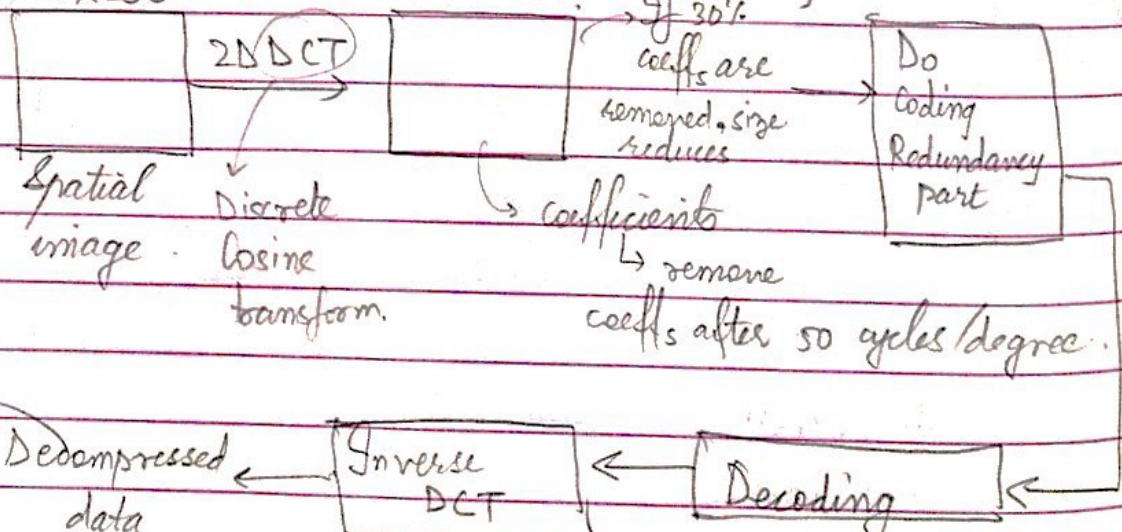
* $C_R = 1 - \frac{1}{R_D}$

* Psycho-Visual redundancy:
  We can only perceive spatial freq below about 50 cycles/degree so that higher freq info. is of little interest to us.



Idea: remove all parts of the image having freq > 50 cycles/degree.

Suppose:



* JPEG : Joint Photographic Experts Group

CLASSMATE

FIDELITY CRITERIA

↳ Seeing quality of compressed & decompressed image.

- 2 CLASSES.

(1) Objective Fidelity Criteria :
Provide us with the eqns that can be used to measure the amount of error in the re-constructed (decompressed) image.
↳ M1 : Seeing error b/w pixels

eg : RMS error
SNR_{RMS} (root mean sq. - signal to noise ratio)
SNR_{peak}

(2) Subjective Fidelity Criteria :
Requires definition of the image based on some criterion. It is done by the users. They decide & rate & differentiate b/w images & see the quality

(1) Objective
- root mean square error (eRMS)

$$eRMS = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]^2 \right]^{1/2}$$

$\hat{f}(x,y)$: estimate resulting from compressing & decompressing
$f(x,y)$: i/p image.

* Error value should be as small as possible, for good quality compression.

- SNR_{RMS}

$$SNR_{RMS} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y)\right]^2}$$

↳ larger the number, better the image

(2) Subjective

eg: A set of 100 people sitting in a hall judge an image.

☆ They compare image on following criterions :-

Impairment	Quality	Comparison
5 Imperceptible	A Excellent	+2 much better
4 Perceptible	B Good	+1 better
3 Slightly annoying	C Fair	0 same
2 Severely annoying	D Poor	-1 worse
1 Unusable	E bad	-2 much worse

comparison is done on the basis of comparing the type of filtering. Suppose I did averaging to image. Then, rating it by comparing with original or any other filter.

CLASSMATE

✱ Compression System Model

2 parts :

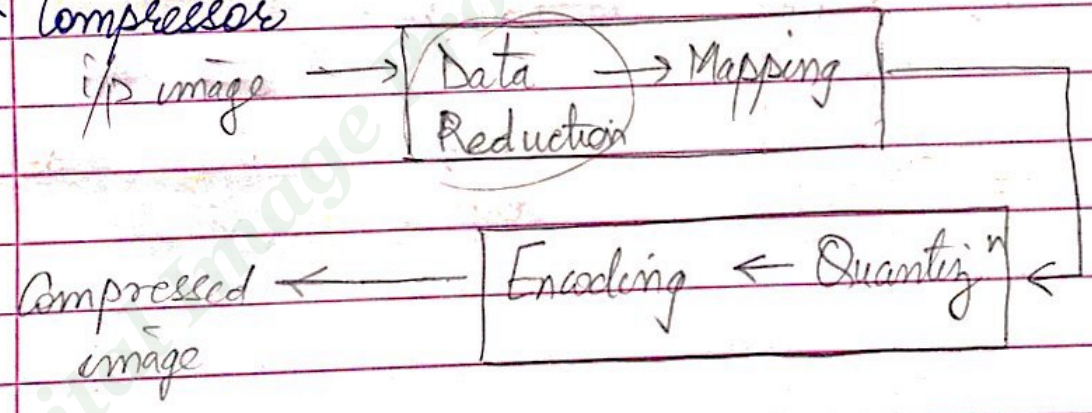
Coding = (1) Compressor
Pre-processing & encoding

Decoding = (2) Decompressor
decoding followed by post processing

(1) • Compression

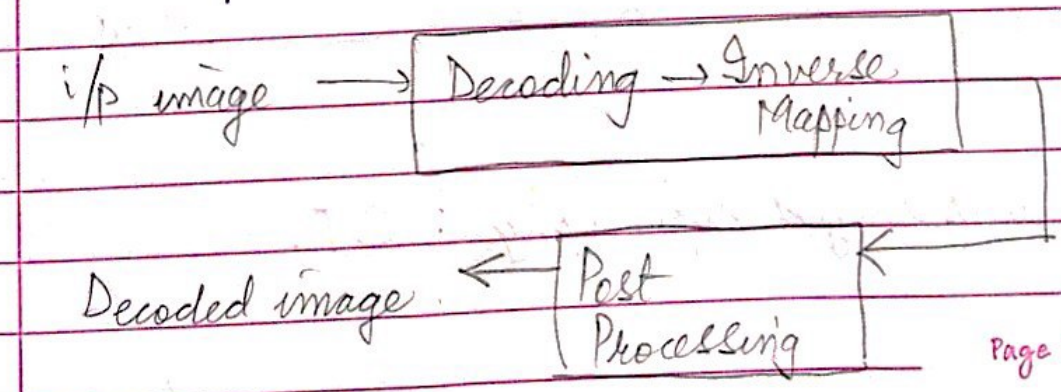
i/p image → pre-processing → encoding → compressed image

✱ Compressor



Source Coding

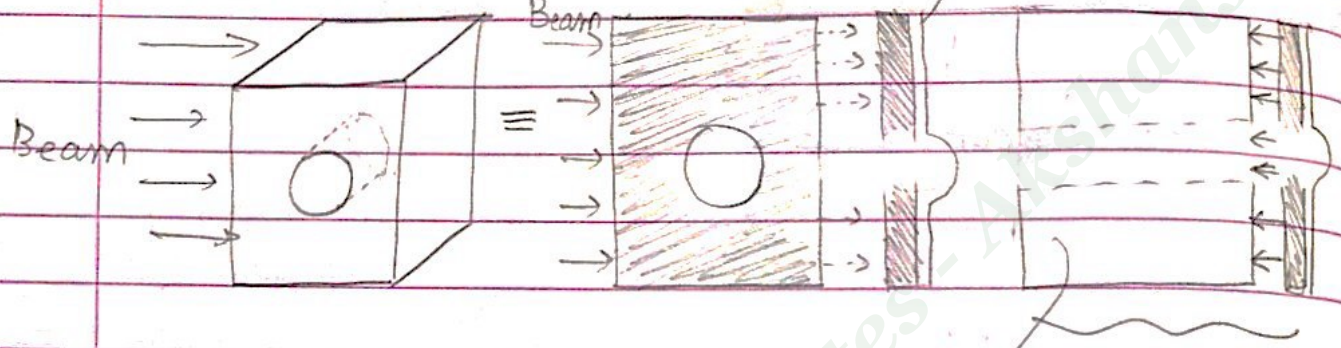
(2) • Decompression.



* Compression types : Lossy & Lossless.

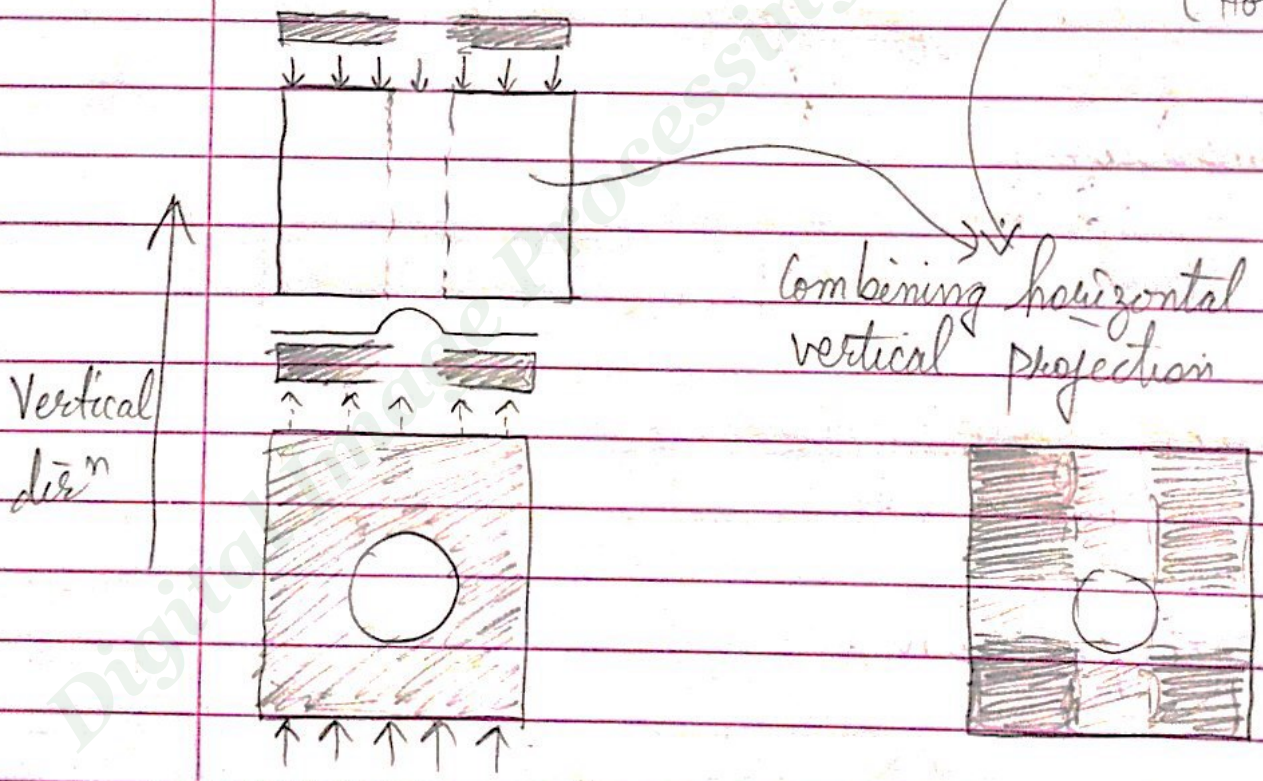
IMAGE RECONSTRUCTION FROM PROJECTIONS

Special case of image restoration



IIly, take/find back projection for vertical dir^{n}.

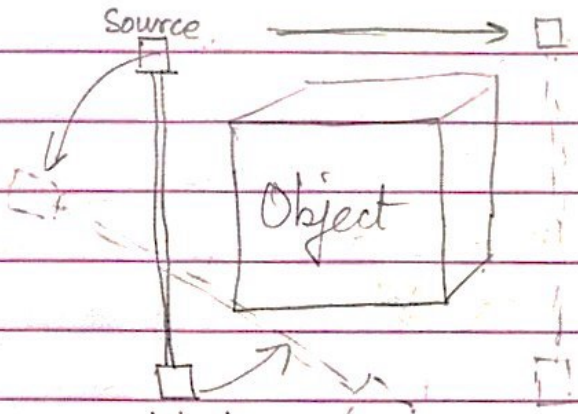
Back projection result (Horizontal dir^{n}.)



* Computerized topography : CT.

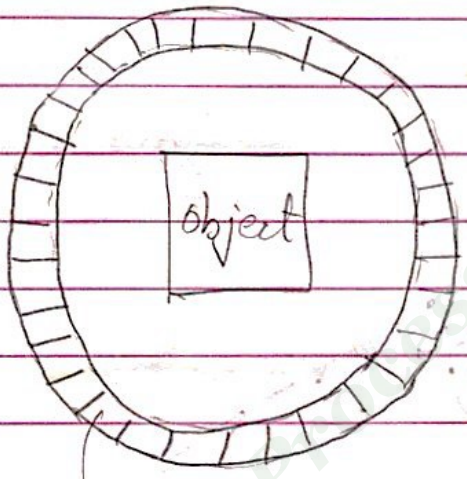
These absorption profiles are studied.

Parallel Projection



∴ It is rotated to take absorption projection from all dirⁿˢ

↳ 1st CT machine (1st generⁿ)

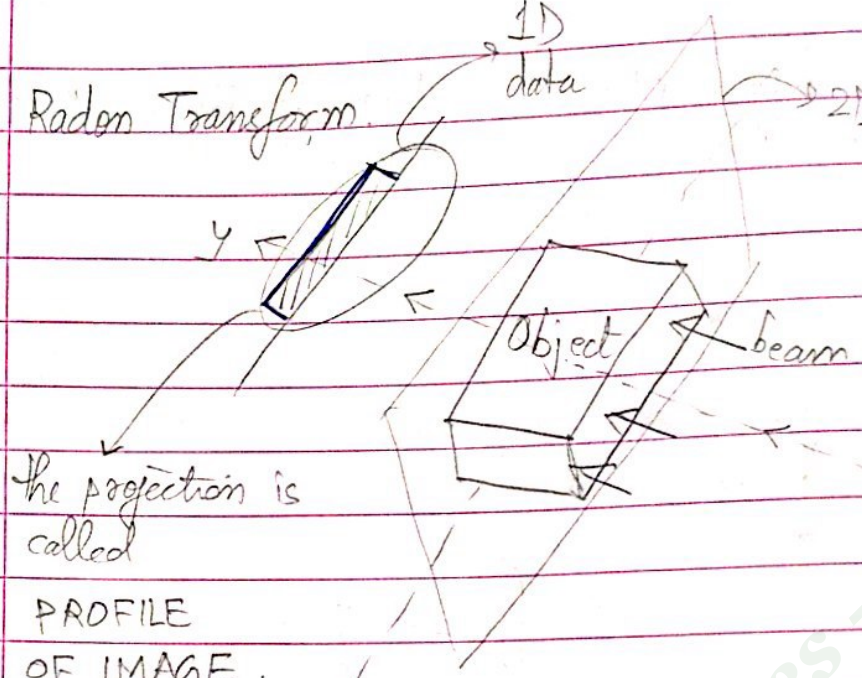


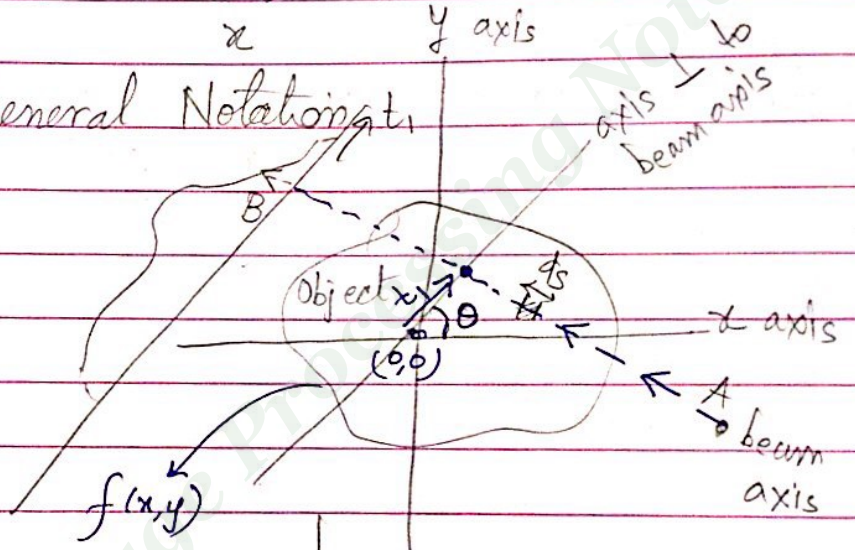
→ 7th generⁿ CT (latest)

multiple source & detectors are used to completely see all projections & hence, reconstruction is easy.

* In image reconstruction, a 2D object is reconstructed from several 1D projections.
* Projection : Obtained by projecting parallel (x-rays (or magnetic rays) (or other penetrating radiation) beam through the object.

☆ Radon Transform





General Notation:-

→ When the beam is passed through the object, we get value of θ & t_1 ($\equiv r, \theta$)

→ distance from origin

✓ A line running through f(x,y) is called RAY.
✓ Integral of f(x,y) along ray is called RAY integral.
✓ Set of ray integral forms the projections.

☆ We are taking (1D) projections & trying to get (2D) data (from various 1D projections)
↳ actual picture of the crossection.

→ (t, θ)

☆ Consider ray AB at an angle θ, distance t_1 from origin. So, ray integral is given by

$$P_\theta(t_1) = \int_{\text{along ray AB}} f(x,y)\, ds$$

Imp ☆ In terms of $x\cos\theta$ & $y\sin\theta$ & f^n shifted by t_1

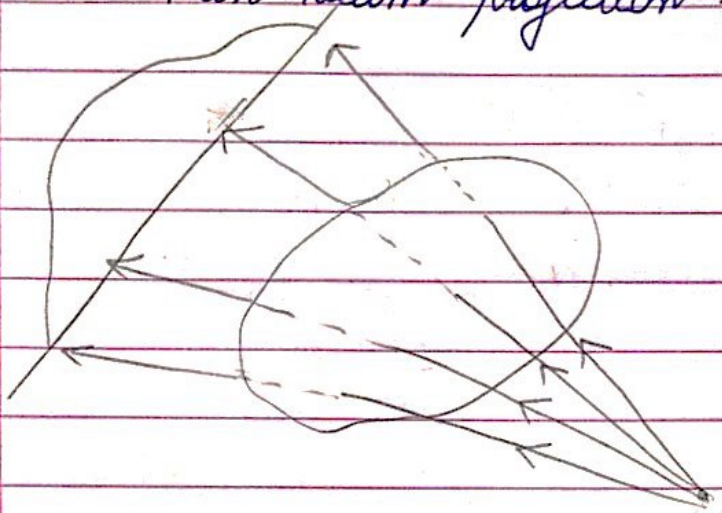
$$P_\theta(t_1) = \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} f(x,y)\, \delta(x\cos\theta + y\sin\theta - t_1)\, dx\, dy$$

↳ $P_\theta(t)$ as a f^n of t (for a given value of θ) defines the parallel projection of $f(x,y)$ for an angle θ. The 2D f^n $P_\theta(t)$ is called Radon Transform of $f(x,y)$

Note : $x(n) = \sum x(n_0)\, \delta(n - n_0)$

↳ SIFTING Property (not shifting)

* Methods of obtaining projection data:
M1: Parallel beam projection (as shown on prev. page)
M2: Fan beam projection.



: Fan beam projection

* Both x-ray source & body are rotated to get various projections
These projections are fed to computer & with the help of reconstruction algorithm, we get image

Back projection algorithm

This is done by FOURIER SLICE THEOREM
It relates 1D FT of a projection of a fn to the 2D FT of $f(x,y)$
i.e., do FT of $P_\theta(t)$ & use it (relate it to) to find $F(u,v)$. Then, our image, $f(x,y)$ is got by $F^{-1}(F(u,v)) = F(x,y)$
↳ $F(u,v)$ = FT of $f(x,y)$

As done before,

$$F(u,v) = \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} f(x,y)\, e^{-j2\pi(ux+vy)}\, dx\, dy$$

Let $S_\theta(\omega)$ be the FT of projection $P_\theta(t)$.

So,

$$S_\theta(\omega) = F\{P_\theta(t)\} = \int_{t=-\infty}^{\infty} P_\theta(t)\, e^{-j\omega(2\pi)t}\, dt \quad \rightarrow (1)$$

Consider $F(u, v)$ on the line $v=0$ i.e, we have only ONE FREQUENCY VARIABLE, $F(u,0)$

($\equiv$ Projection in vertical dirⁿ)

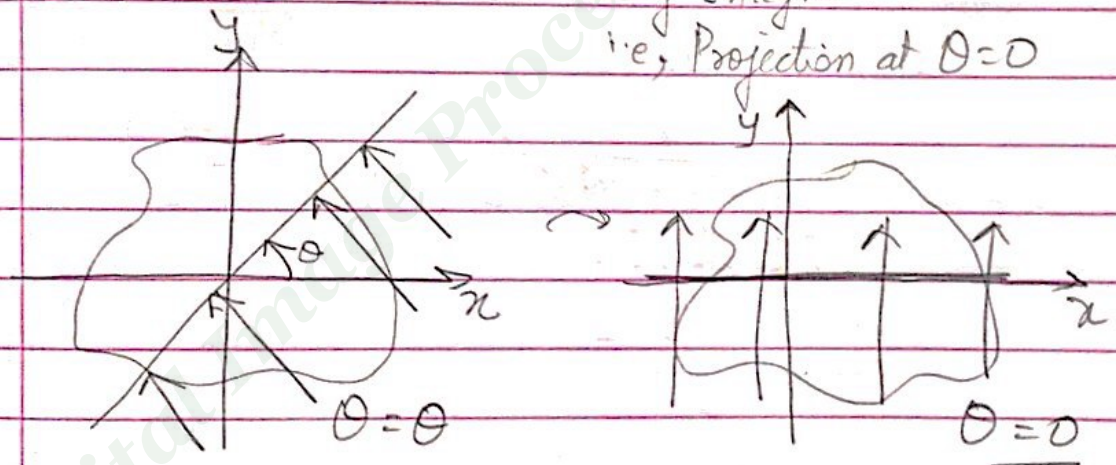
$$\Rightarrow F(u,0) = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} f(x,y)\, e^{-j2\pi ux}\, dx\, dy \quad \rightarrow (2)$$

Now, we have to relate (1) & (2) to get $f(x,y)$

Rearranging (2)

$$\Rightarrow F(u,0) = \int_{x=-\infty}^{\infty} \underbrace{\left[\int_{y=-\infty}^{\infty} f(x,y)\, dy\right]}_{\text{Ray Integral}} e^{-j2\pi ux}\, dx$$

i.e, Projection at $\theta=0$



Projection : $P_\theta(t)$ Projection : $P_0(t)$

$$\Rightarrow F(u,0) = \underbrace{\int_{x=-\infty}^{\infty} P_0(t)\, e^{-j2\pi ux}\, dx}_{\equiv \text{FT of projection}} = S_0(u)$$

$$\Rightarrow F(u,0) = S_0(u)$$

✱ $F(\omega, \theta)$ denotes the values of $F(U,V)$ along a line at angle θ, with ω as axis, then

$$F(\omega,\theta) = S_\theta(\omega)$$

2D FT of image $f(x,y)$ — FT of 1D projection

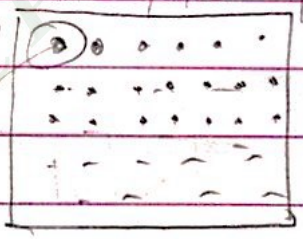
✱ Idea to Implement in MATLAB

➤ Previous topic: STEGANOGRAPHY

In this method, text info. is hidden in the image.

This text info. can be taken from keyboard or from a text file

So, we have: ① pixels of image ② text info.

eg: Pixels: 

& Text: BITSPILANI

Convert the text to ASCII equivalent & get their binary values & concatinate. We get 1111001011100011

read this pixel in a for loop 10011100 → replace it.

Now, do this ∀ pixels: text info is put.

(If image = 256×256, then 2^{16} = 64 K bits of text info can be put)

The replacement work (text info → pixels) is done until length of the text info is exhausted. So, use a "for" loop with this cond".

* Note: Steganography cannot be done on a lossy compression format.
So, don't use image file formats like jpeg.
Instead, use .bmp.

JPEG COMPRESSION

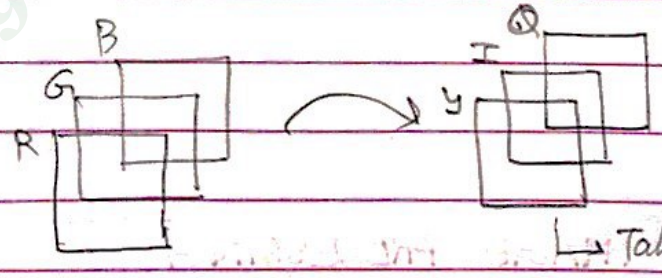
In oldern times, data from website was taken and compressed using JPEG compression. So, the image was loaded in parts.

Now, when we load any image, a blurred image is shown completely. Then, finally, we get our image. This is : JPEG-2000.

Imp for Compre

JPEG COMPRESSION METHOD

Image is displayed in display devices using RGB. But, the image processing is done on YIQ or YUV (RGB converted to YIQ or YUV)



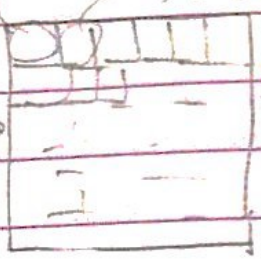
↳ Take one plane.

(PTO)

one of the planes of YIQ.

It's a matrix basically → divide it into small chunks of (8×8) for that plane.

8×8 chunk



→ After this, normalizing can be done

256 → −128 to +127

→ Do DCT of image chunks

DCT matrix F(U,V)

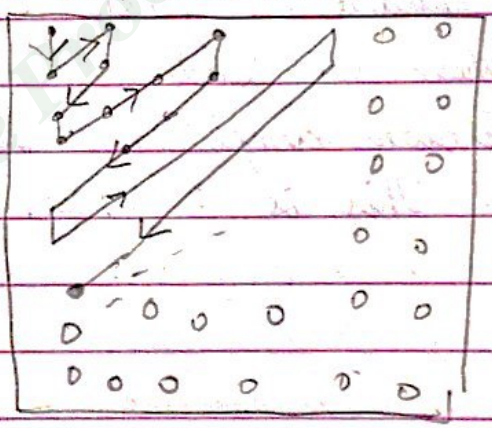
Quantize image

We get quantized matrix

Quantized matrix written in an array

Observ^n: Most of lower matrix components become zero.

Now, these remaining (Non-zero) DCT coeff. are presented in ZIG-ZAG manner



Divide DCT coeff. with std. quantiz^r matrix

16 11 10 16 24 . . .

8×8

Hoffman Coding done to the array

We get these values

1010110 0100 001

0100 0101 . . . .

Can be further reduced by applying Run-length coding.

✡ Decompression: Reverse process as above

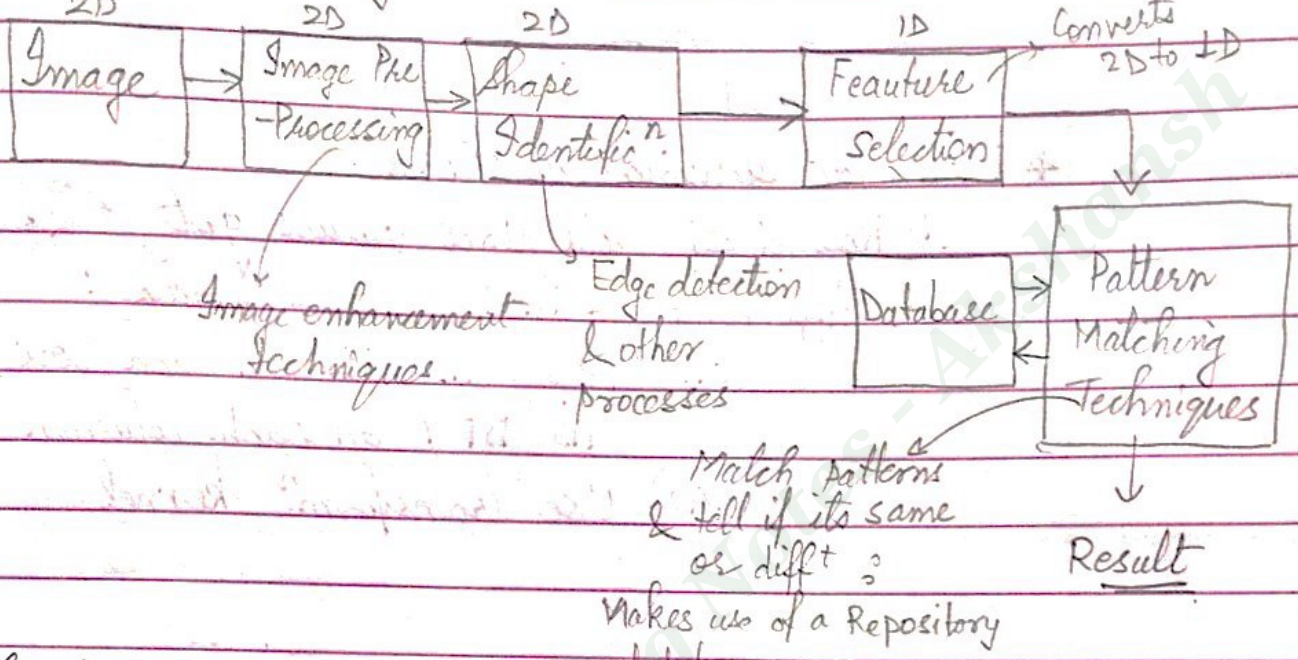
Compre: Short notes will be asked

✡ APPLICATIONS OF IMAGE PROCESSING

Object Recognition, Character recognition, Remote sensing, Biomedical applic^n

* OBJECT RECOGNITION

↳ To recognise an object from an image, first, we need an image

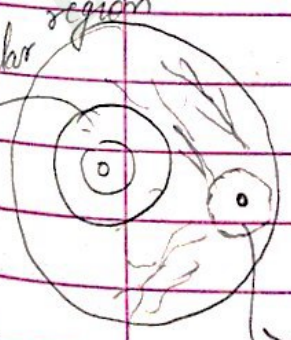


Similar is applied in Characters recognition. We have character instead of image

Biomedical Applic^n :

* Diabetes affects retina of eye. So, the fragile blood vessels of retina, under excess pressure break. This fluid gets deposited on retina. To detect this : take picture of retina (focus camera at eye lens). With enhancement of picture, we can see the optic nerve pos^n & any problems in that.

Eye also has a portion called Macula. If anything happens to it, we get blind. So, image processing techniques are applied to know the macular region. So, if laser treatment is done, this region can be avoided.



As pressure is in excess, exudates might spread near the macular region. So, we can know the severity of problem (how close to macular region.

* Note for solving questions.
  (1) Round off all pixel values got ($\geq .5 \rightarrow$ round up, $< .5 \rightarrow$ round down)
  (2) Applying 2D DFT on any image:
    long M1) Apply 1D DFT on each row, then 1D DFT on each column.
    short M2) Use transform^n kernel (if given)
      If image = F, kernel = W, Transformed image, T = WFW^T
      ↳ Use magnitude when we filter any image.

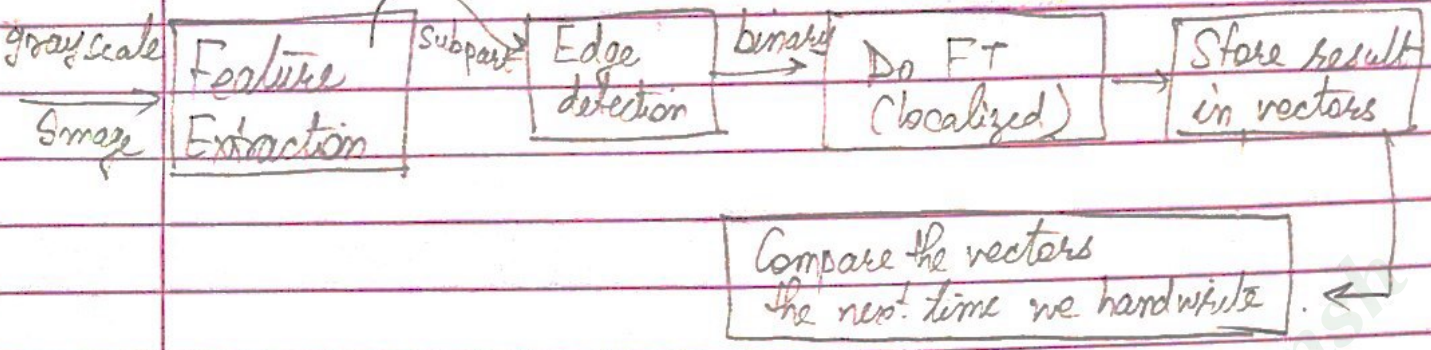
* Character Recognition.
  2 Parts: ① Data Acquisition
  ② Pattern Recognition.

Inside brain
- Part ①:- I see the image & analyse how the handwriting looks (image formed on retina & few parameters out of that go to brain)
- Part ②:- When I see that handwriting again, I COMPARE with previous parameters & understand the image (recognition)

Inside Computer :-

Acquire Image → Pre-Processing → Extract FEATURE → PTO

→ techniques: edge, boundary detection



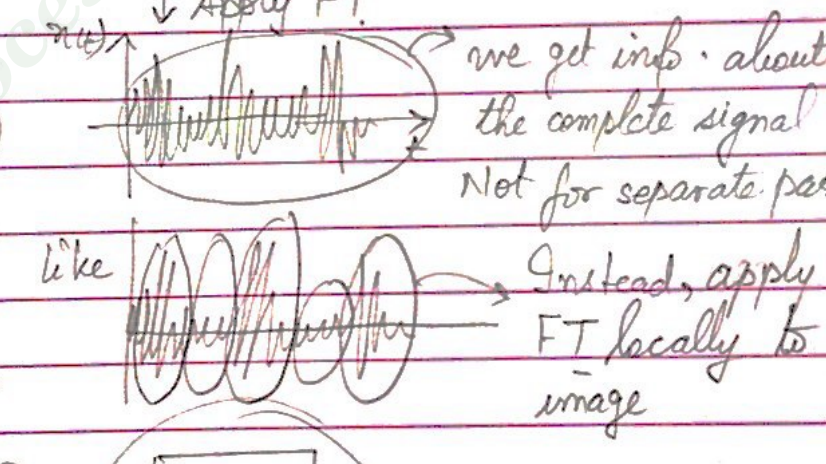
eg:- Processing an image :- C

After edge detection, we get

Apply Merging & splitting to take the required part of this image

(Image size reduced) C → 2D Image

↓ Apply FT



* This vector info. is compared everytime to recognize handwriting