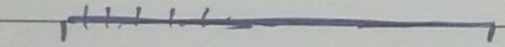
Date 2 / 03 / 2022

Digital Image Processing

Digital Images can be classified as.

(i) Coloured
(ii) Gray scale
(iii) Binary

Gray Scale:



White ← 255 Intensity levels → Dark

2^8

8 bit binary for each value.

for each pixel

Binary

Black / White

0 (or) 1

→ single bit for every pixel.

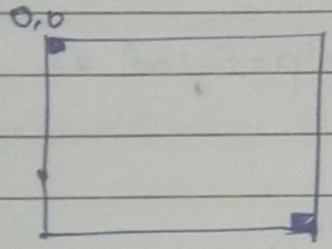
Properties of Digital image processing:

(i) Non-causal
(ii) 2-Dimensional processing
(iii) Shift invariant
(iv) 2-D DTFT.

Date 11/8/2022

Processing Operations:

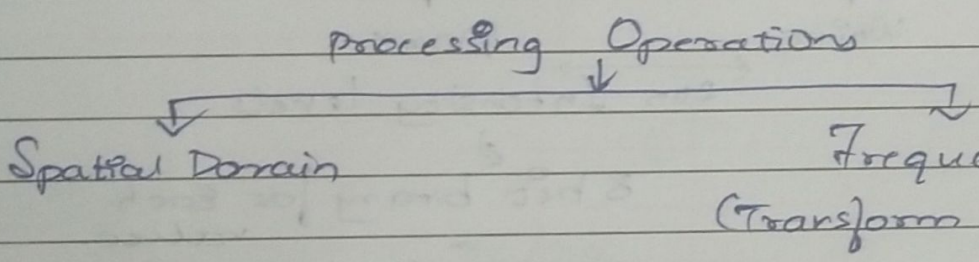
↳ Limited to Gray scale images (mostly)

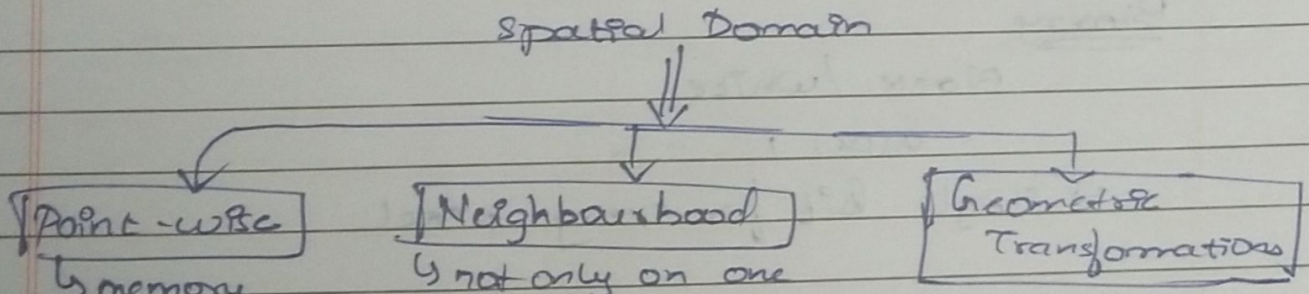


$f(x,y)$

x, y ⇒ spacial coordinates

↳ 8 bits per pixel ⇒ 0 to 255 intensity levels.





Local Average:

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

$f(x,y) \Rightarrow$ Input image, $g(x,y) \Rightarrow$ output Image.

$$g(x,y) = \frac{1}{9} \sum_{u=x-1}^{x+1} \sum_{v=y-1}^{y+1} f(u,v).$$

This 'local average' method is used to Blur an image.

↳ The amount of blur is controlled by the size of the neighbourhood we take.

i) Point-wise Operations:

$$f(x,y) \longrightarrow g(x,y)$$

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

Let 'r' be the value of pixel at any point (x,y). Then the corresponding value at output image be 's'

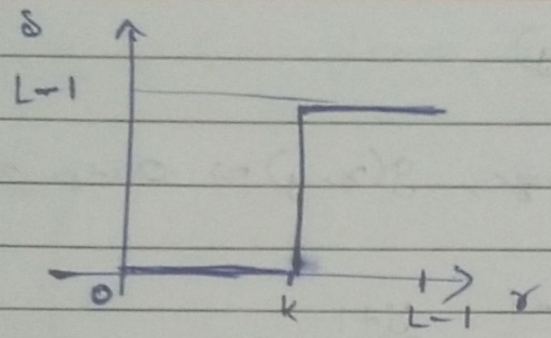
Then,

$$\boxed{s = T[r]}$$

$r \in [0, 1, \ldots, L-1]$.

Date / /

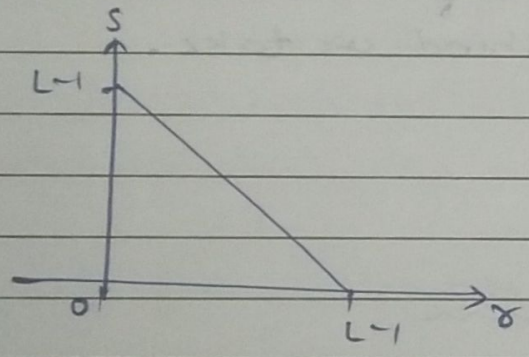
(i)



(binarisation of an image).

(ii) Negative of an image:-

$$s = L-1-r$$

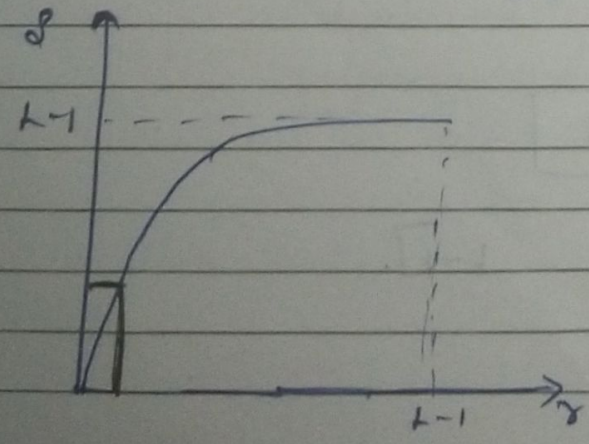


Applications of negative

↳ To make an image more clear, when the background develops with some of it's colors

(iii) Log-transformation:-

$$s = c \log(r+1)$$

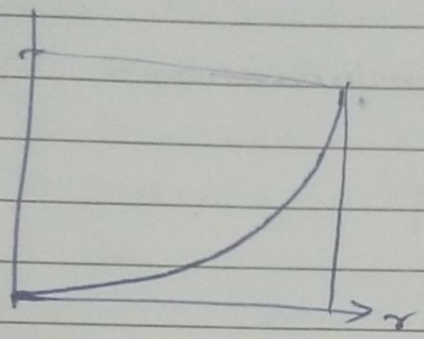


↳ Enhance the features of darker portions which was lost due to high intensity of brighter regions

↳ Darker's streched

Date / /

(iv) Inverse Log Transformation:

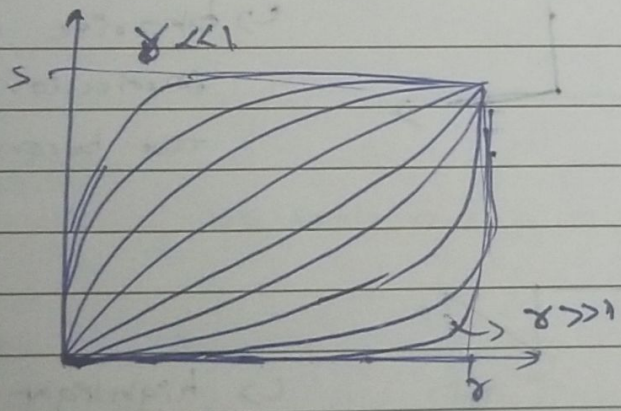


Brighter regions are stretched & Darker regions are compressed

(v) Power law Transformation:

↳ Gamma Transformation

$$s = c(r)^{\gamma}$$



↳ for $\gamma > 1$, Brighter regions are stretched & for $\gamma < 1$, Darker regions are stretched.

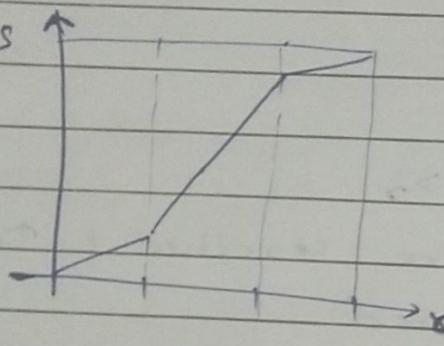
Inverse γ-transform → $s = c(r)^{1/\gamma}$ //

↳ used in Contrast Enhancement.

Date / - /

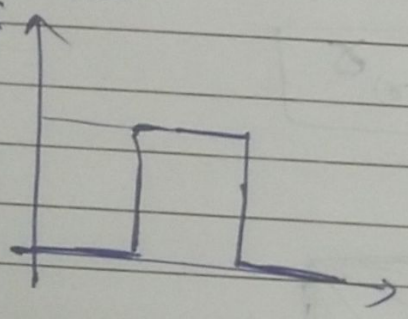
~~(Contrast Enhancement)~~

(vi) Piecewise linear transformation:

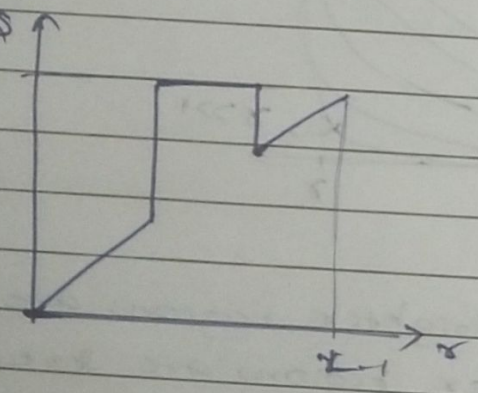


↳ contrast streching.

(vii) Intensity slicing:



↳ binarise a particular region rest becomes black



↳ highlight a particular intensity level, rest levels stays the same.

Date / /

Image Enhancement:

(i) Contrast Enhancement
- ↳ log - Transformation
- ↳ Powerlaw transformation
- ↳ Intensity level slicing

(viii) Bitplane Slicing:

for 256 Intensity levels,

↳ 8 Bit Planes

↳ as you consider only 3 (or) 4 bit planes, the rest bit places are zero

↳ reduces memory ~~space~~ required

↳ As you increase the no of bit planes the finer details are more visible.

Histogram:

Intensity levels => $0, 1, 2 \cdots, L-1$

$r_0, r_1, r_2 \cdots r_n$

r_n → Intensity level

$$h(r_k) = \frac{n_k}{M \times N}$$

$M \times N \Rightarrow$ Total no. of pixels

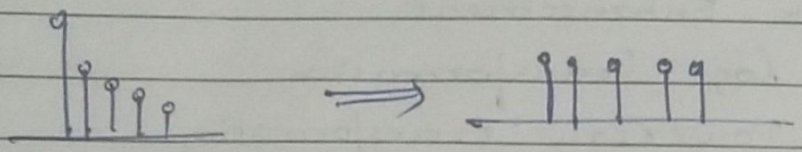
Normalized histogram

Unnormalized histogram:

$$h(r_k) = n_k$$

Date / /

Histogram Equalization:



Continuous Case:

Let x & y be two random variables &

let $y = g(x)$

then,

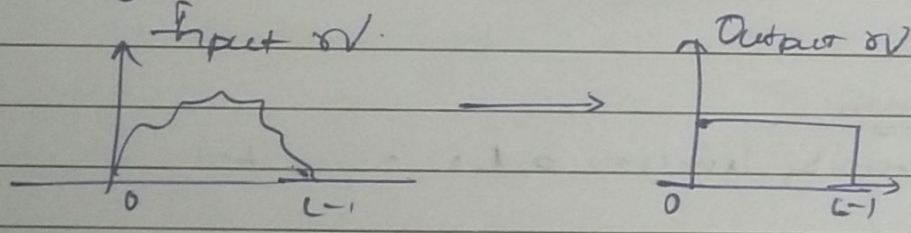
we have

$$f_y(y) = f_x(x) \left| \frac{dx}{dy} \right|$$

Here, we should consider only monotonically increasing functions - so

$$f_y(y) = f_x(x) \cdot \left(\frac{dx}{dy} \right)$$

So, the goal is to



r

(0, L-1)

s

(0, L-1)

$$s = T(r) = (L-1) \int_0^r P_r(u)\, du.$$

$P_r(u) \Rightarrow f_r(y) \Leftarrow \frac{1}{MN}$

$$\boxed{P_s(s) = P_r(r) \cdot \left(\frac{dr}{ds} \right).}$$

Date / /

$$\frac{ds}{dr} = (L-1)\, P_r(r)$$

$$\therefore \quad P_s(s) = P_r(r) \cdot \frac{1}{P_r(r)\cdot(L-1)}$$

$$\boxed{P_s(s) = \left(\frac{1}{L-1}\right)}$$

Discrete case:

$r_0, r_1, \ldots, r_{L-1}$

$s_0, s_1, \ldots, s_{L-1}$

$\boxed{r_i \rightarrow s_i}$ (r_i is mapped to s_i)

$$\boxed{s_k = \left[(L-1)\sum_{j=0}^{k} P_r(r_j)\right]} \quad \text{rounding operation.}$$

Histogram Bin:

↳ Bin size (typically 1)

Advantages over other contrast Enhancing techniques:

(i) Independent of input image and any other parameters.

Disadvantage:

↳ May not work for all images. ⇒ A large chunk of pixels lie on the same intensity level (ex: ①).

Date / /

Histogram Matching:

↳ Histogram Specification.

↳ Target Histogram → We try to match our output to that of the target

r → $P_r(r)$

z → $P_z(z)$

$s = T(r) = (L-1)\int_0^r P_r(r)\,dr$

$G(z) = (L-1)\int_0^z P_z(v)\,dv$.

Therefore $G(z) = s$.

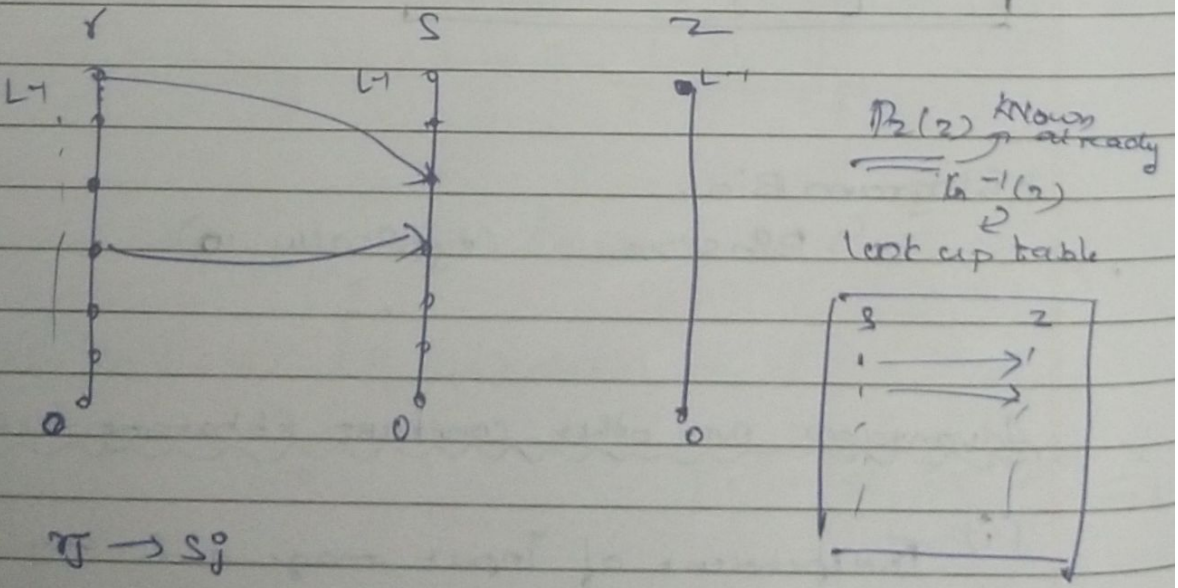
$G^{-1}(T(r))$

Problems

↳ If $T(r)$ exceeds the range of $G(z)$

↳ If there are many values in $G(z)$ pointing to the same spot in $T(r)$

↳ This has a sdh that CDF should be strictly increasing in nature



$r_j \rightarrow s_j$

Date / /

Global histogram Processing:

↳ processing based on the histogram of the entire image.

↳ some cases, it is not desirable, so we opt for local histogram processing.

$$\mu_n = \sum_{i=0}^{L-1} (r_i - m)^n \, p_r(r_i)$$

↳ n^{th} central moment of the function.

where

$$m \Rightarrow \sum_{i=0}^{L-1} r_i \, p_r(r_i)$$, which is mean

Neighbourhood Operations:

(i) Min ⇒ Set the pixel value to the minimum of the neighbourhood

(ii) Max ⇒ Set the pixel value to the maximum in the neighbourhood

(iii) Median ⇒ the median value of a set of numbers is the midpoint value in the set. Sometimes, median works better than average.

↳ good for noise removal.

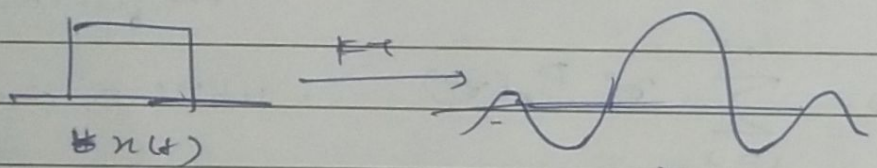
Date / /

Smoothing Spatial Filters

↳ Average of all pixels in a neighbourhood around a central value

↳ Useful for removing noise from image

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



↳ approximate LPF

↳ The high frequencies (i.e sharp edges) will become smooth.

Local mean about (x, y)

$$m_{s_{x,y}} = \sum_{i=0}^{L-1} r_i P_{s_{x,y}}(r_i)$$

Local variance around (x, y)

$$\sigma^2_{s_{(x,y)}} = \sum_{i=0}^{L-1} (r_i - m_{s_{x,y}})^2 P_{s_{x,y}}(r_i)$$

Date / /

Local histogram processing

$$g(x,y) = \begin{cases} E\,f(x,y) & \text{if } k_0 m_G \le m_{Sxy} \le k_1 m_G \text{ \& } k_2 \sigma_G \le \sigma_{Sxy} \le k_3 \sigma_G \\ f(x,y) & \text{elsewhere} \end{cases}$$

$m_G \Rightarrow$ Global mean

$\sigma_G \Rightarrow$ Global ~~variance~~ SD

Spatial Filtering

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

↳ neighbourhood operations.

$w(s,t) \Rightarrow$ mask : Kernal

Multiplying the mask and the neighbourhood of the point (x,y) is called Correlation

$$g(x,y) = (w \overset{g}{\star} f)(x,y).$$

Convolution in 2D domain:

Equivalent to flipping by 180°

↳ horizontal & vertical flipping

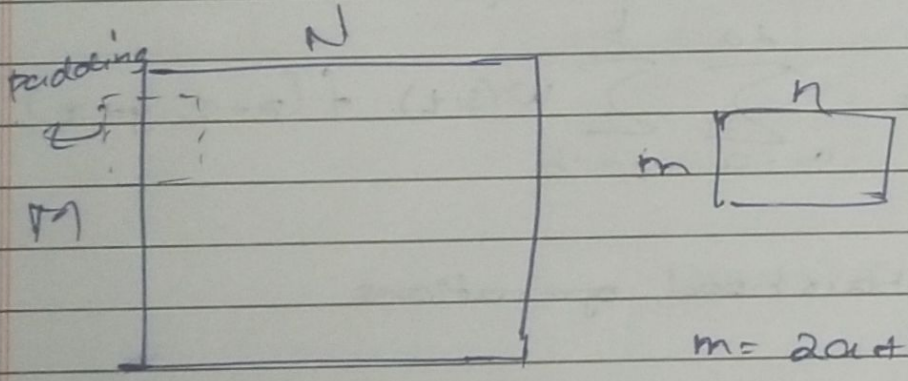
$$g(x,y) = (w \odot g)(x,y)$$

Date / /

2D convolution

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

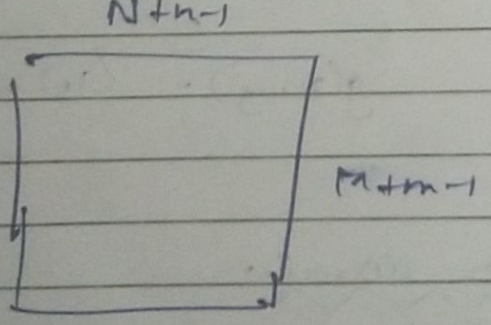
convolution is correlation after rotating the mask by 180°.



$m = 2a+1$

$n = 2b+1$

after doing edge correction, the size of image is (padding)



Date / /

Properties of convolution:

$$f * g = g * f$$

$$f * (g * z) = (f * g) * z$$

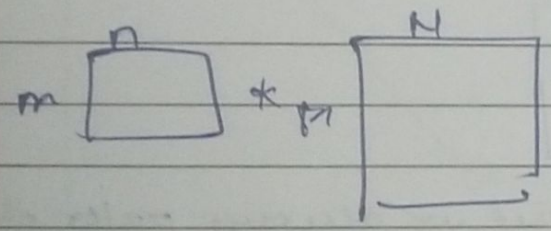
Seperable fn:

$$w = w_1 * w_2$$

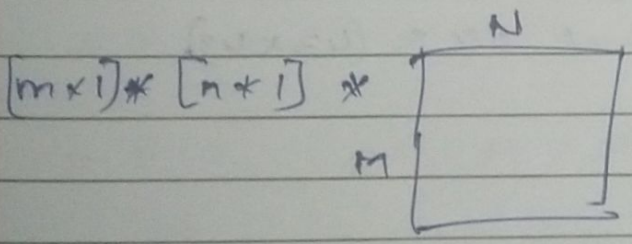
$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

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

↳ box filter (all values are same)



⇒ No of additions & multiplications req is "MNmn"



⇒ No of additions and multiplications req is MN (m+n)

Gaussian filtering:

↳ Also used for spatial Averaging

$$w(s,t) = k e^{-\left(\frac{s^2+t^2}{2\sigma^2}\right)}$$

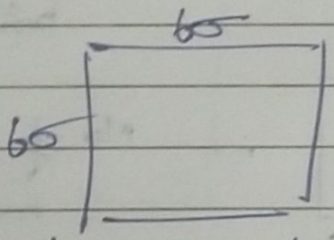
Date / /

↳ It's like a weighted average
↳ circularly symmetric (or) isotropic

$\frac{1}{4.8976}$

0.3679	0.6065	0.3679
0.6065	1	0.6065
0.6065	0.6065	0.3679

↳ has the seperable property
↳ first vertically & then horizontally
↳ only circularly symmetric kernel mask to be seperable.
↳ follows 3σ rule
↳ So the width of the Gaussian kernel is 6σ x 6σ



if σ = 7, 6σ = 42, but we always prefer the size to be an odd integer,
So the kernel size is (43 x 43).

Date / /

Sharpening Spatial filters

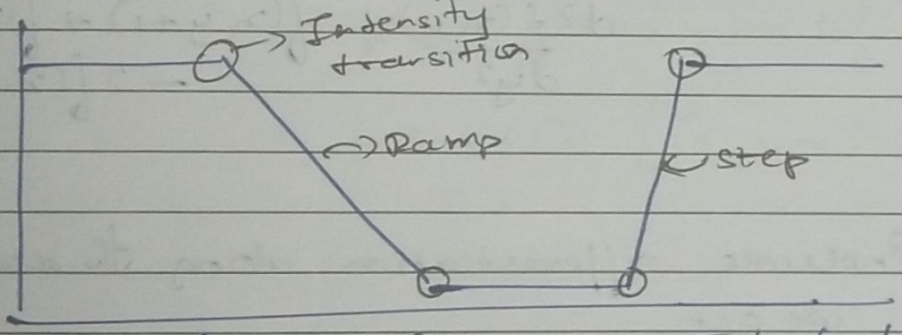
$\frac{df(x)}{dx}$ ⇒ first difference = $f(x+1) - f(x)$

$\frac{d^2 f(x)}{dx}$ ⇒ Second difference = $f'(x+1) - f'(x)$

$= f^{\#}(x+1) - f(x) - f(x) + f(x-1)$

$= f(x+1) + f(x-1) - 2f(x)$ //

Let's consider the horizontal intensity profile,



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

f ⇒ 0 0 0 -1 -1 -1 -1 0 0 0 1 2 1 1 1

S ⇒ 0 0 0 -1 0 0 0 +1 0 0 1 1 -1 0 0 0

Zero crossing at the intensity transition of the ramp

↳ we use second order filter for better edge detection

Date / /

Laplacian filter:

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

Second derivative in x dir. — second direction in y direction

0	1	0
1	-4	1
0	1	0

along x,

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

along y,

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

If we include differentiation along the diagonal direction we get

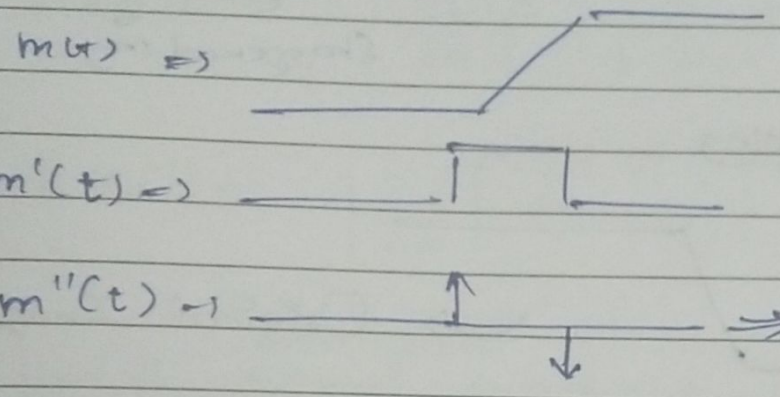
1	1	1
1	-8	1
1	1	1

$$f'(x,y) = f(x+0,y) - f(x-1,y)$$

$$f''(x,y) = f'(x,y) - f'(x-1,y)$$

Date / /

Sharpening



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

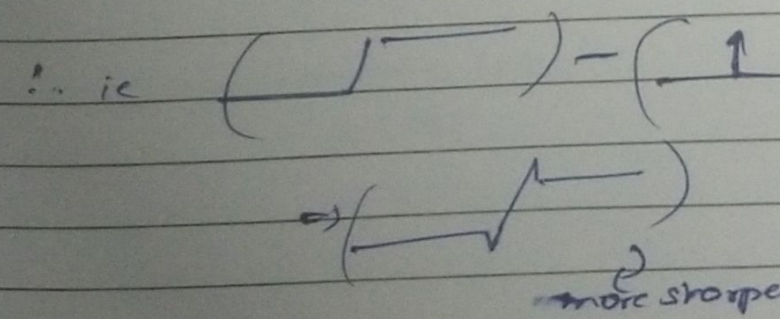
$$= f(x+1, y) + f(x-1, y) - 2f(x, y)$$
$$+ f(x, y+1) + f(x, y-1) - 2f(x, y)$$

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

↓

The resulting Image shows the Intensity change. (sharpened)

↳ Sharpening effect will be better visualised if we subtract $f''(x, y)$ from $f(x, y)$

∴ ie 

↓

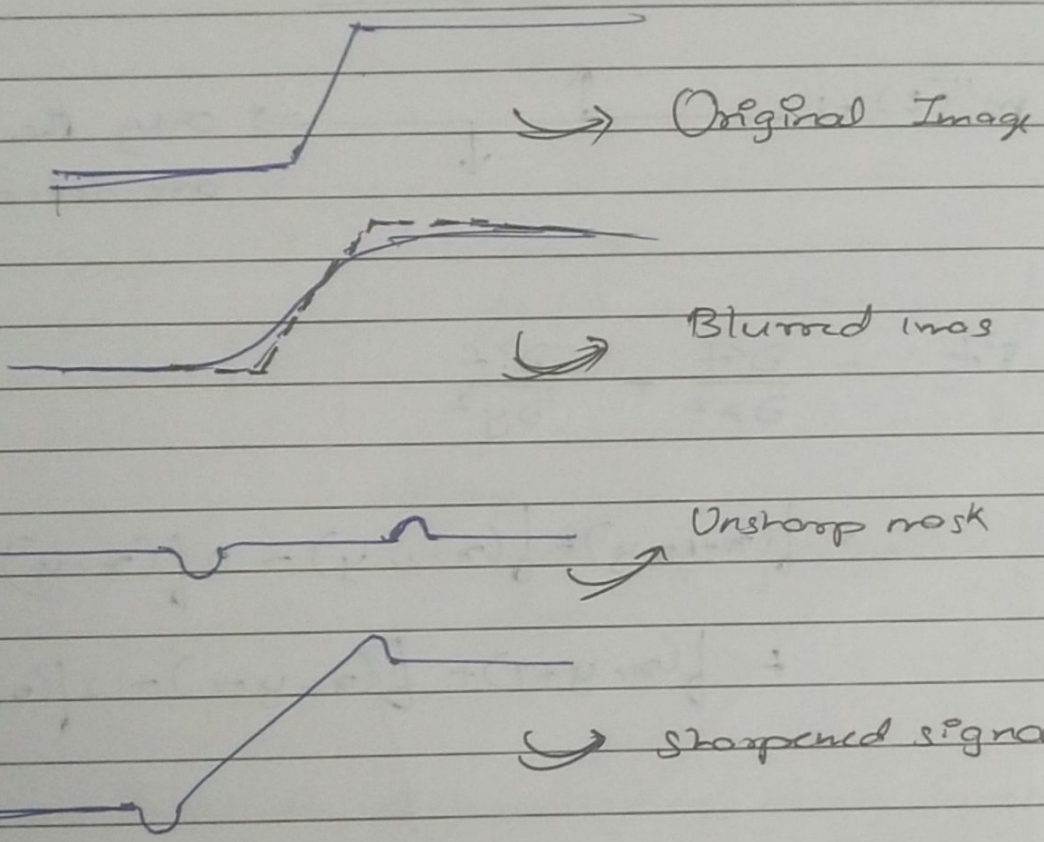
more sharpened

Date / /

$\Rightarrow f(x,y) + c\nabla^2 f(x,y) \Rightarrow g(x,y)$

Sharpened img

Unsharp masking



$$g_{mask}(x,y) = f(x,y) - \overbrace{f(x,y)}^{\text{Blurred image}}$$

$$\text{Sharpened img} = f(x,y) + K\, g_{mask}(x,y)$$

If $K=1$, Unsharp masking

$K>1$, Highboost filtering

Date / /

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\|\nabla f\| \approx \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|.$$

To find the horizontal gradient we have the filter,

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

computes $\frac{\partial f}{\partial x}$

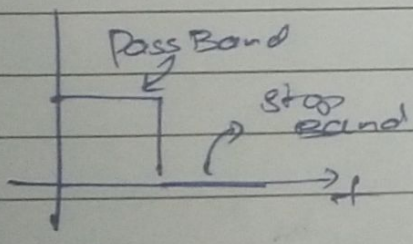
To find the vertical gradient we have filter

Sobel mask →

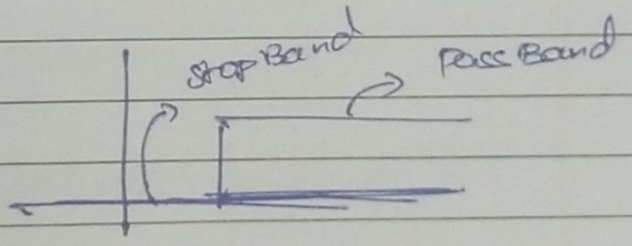
-1	0	1
-2	0	2
-1	0	1

computes $\left(\frac{\partial f}{\partial y}\right)$.

Filters



LPF

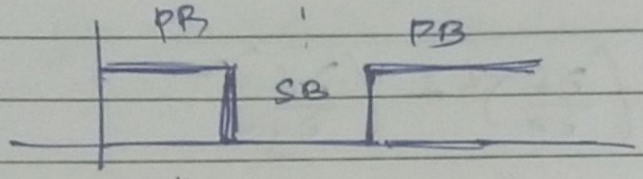


HPF

HPF ⟶ LPF

1 − LPF

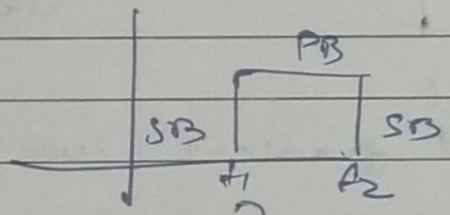
Date / /



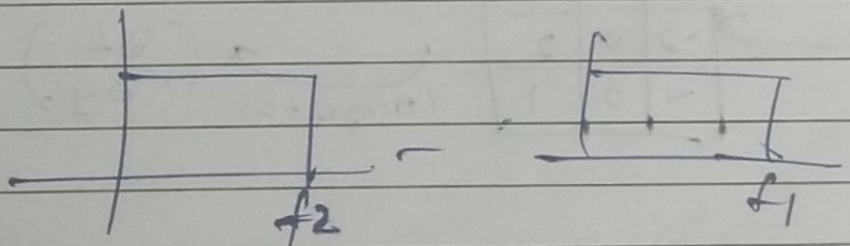
Notch filter

Notch filter ⇒ LPF + HPF
(s from LPF

Bandpass filter



2



Date / /

Frequency Domain Filtering:

$$x(t) \xrightarrow{FS;\ T_0} a_k$$

$$x(t) = \sum_{k=-\infty}^{\infty} a_k \cdot e^{\pm j k \omega_0 t}$$

for

$$x(t) = \sum_{k=0}^{\infty} b_k \sin(\omega_0 k t) + \sum_{k=0}^{\infty} c_k \cos(\omega_0 k t)$$

$$\Rightarrow$$

sum of harmonics of ω_0.

for periodic signals:

$$a_k = \langle x(t), e^{jk\omega_0 t} \rangle$$

$$= \frac{1}{T_0} \int_{\langle T_0 \rangle} x(t)\, e^{-jk\omega_0 t}\, dt.$$

Fourier Transform

$$x(t) \longrightarrow X(f)$$

$$x(t) = \int_{-\infty}^{\infty} X(f) \cdot e^{j 2\pi f t}\, df$$

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j 2\pi f t}\, dt$$

$$x[n] \xrightarrow{DTFT} X(e^{j\omega})$$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}.$$

↙ continuous in ω with 2π period

Date / /

DFT

↳ Discrete Fourier Transform:

Signal $\xrightarrow{DFT}$ (FT)
(Digital) (Digital)

N samples in both time domain & frequency domain

$$N = \frac{T_0}{T}$$

$T_0 \Rightarrow$ width of the windowing used
$T \Rightarrow$

$$x[n] \xleftrightarrow{DFT} X[k]$$

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j(2\pi/N) \cdot k \cdot n}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \, e^{j(2\pi/N) \cdot k \cdot n}$$

$x[n]$ and $X[k] \Rightarrow$ periodic signals
↓
period : N

we consider only the samples from 0 to N-1

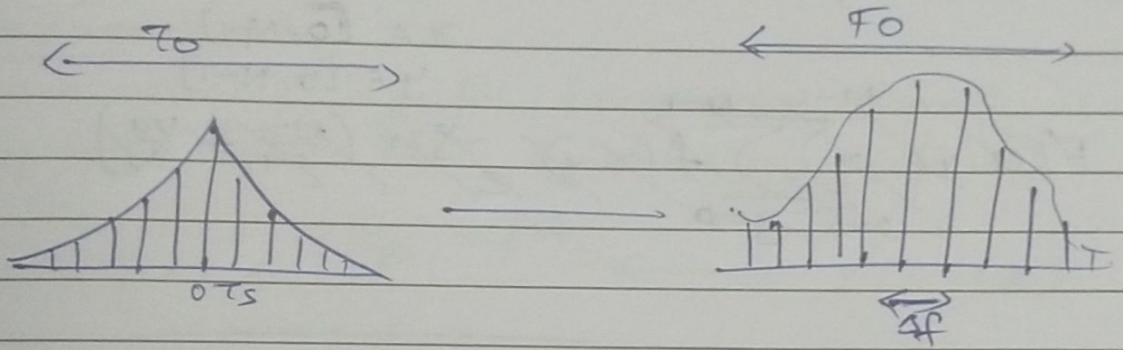
Date / /

Time resolution

↳ Time difference btw two successive samples in $x[n]$

Frequency resolution

↳ Frequency diff. between two successive samples in $X[K]$.



$$N = T_o / T_s$$

$$\Delta f = \frac{1}{T_o}$$

Properties

(i) $x[n-n_0] \xrightarrow{DFT} X[K] \cdot e^{-j(2\pi/N)Kn_0}$

(ii) $x[n] \cdot e^{j(2\pi/N)K_0 n} \longrightarrow X[K-K_0]$

(iii) $x_1[n] \xrightarrow{N} X_1[K]$

$x_2[n] \xrightarrow{N} X_2[K]$.

$$x_1[n] \textcircled{*} x_2[n] = \sum_{m=\langle N \rangle} x_1[m] \cdot x_2[n-m]$$

circular convolution

$x_1[n] \textcircled{*} x_2[n] \xrightarrow{DFT} X_1[K] \cdot X_2[K]$.

Date / /

Notation

$$f(x) \xleftrightarrow{DFT} F(u)$$

for 2D signals

$$f(x,y) \xleftrightarrow{2D\ DFT} F(u,v)$$

$x \in [0, M-1]$

$y \in [0, N-1]$

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

and inverse 2D-DFT is

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

$$f(x,y) \longleftrightarrow F(u,v) = |F(u,v)| \cdot e^{j\phi(u,v)}$$

$|F(u,v)|$: magnitude spectrum; $\phi(u,v)$: phase spectrum

$$F(u,v) = R(u,v) + jI(u,v)$$

$R(u,v)$: Real part spectrum; $I(u,v)$: imaginary part spectrum

Date / /

$$\phi(u,v) = \tan^{-1}\left(\frac{I(u,v)}{R(u,v)}\right)$$

$$\boxed{F(0,0) = MN\bar{f}}$$

Shifting

We usually shift the frequency domain waveform By $N/2$, to see the symmetric property.

$$f(x).e^{+j\left(\frac{2\pi}{N}\right)x.(N/2)} \longleftrightarrow F(u-N/2)$$

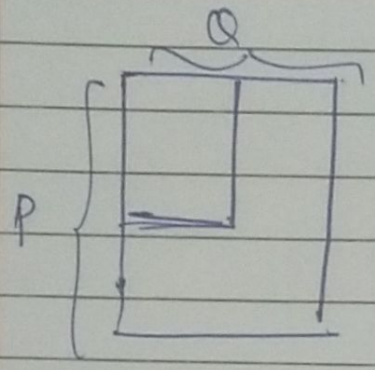
$$f(x).(-1)^x \longleftrightarrow F(u-N/2)$$

In 2-D case,

$$(-1)^{x+y} f(x,y) \longrightarrow F(u-M/2,\ v-N/2)$$

Date / /

Date / /



$M \times N$

$P \geq 2M-1$

$Q \geq 2N-1$

Generally $P, Q \in I$, then we have

$$[P = 2M \text{ & } Q = 2N]$$

Method I

Image $\Rightarrow f(x,y) \Rightarrow M \times N$

↳ append zero values

$f'(x,y) \Rightarrow P \times Q$

$M \times N$ filter mass in frequency domain $H(u,v)$

↳ IDFT

$h(x,y)$ $[M \times N]$

↳ Append with zero

$h'(x,y)$

$[P, Q]$

$h'(x,y) \xrightarrow{DFT} h'(u,v)$

$[P \times Q] \qquad [P \times Q]$

$f'(x,y) \circledast h'(x,y) \xrightarrow{DFT} \square \xrightarrow{IDFT} g(x,y)$

$(P \times Q) \qquad [P \times Q] \qquad X(M,N)$

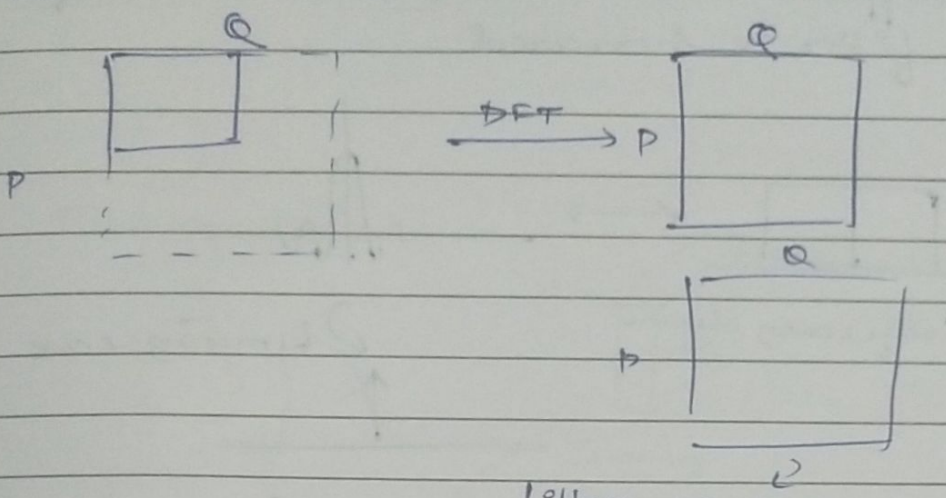
↳ Filtered image

$[P \times Q]$

↳ This will create "ringing", this is one of the key limitations

Date / /

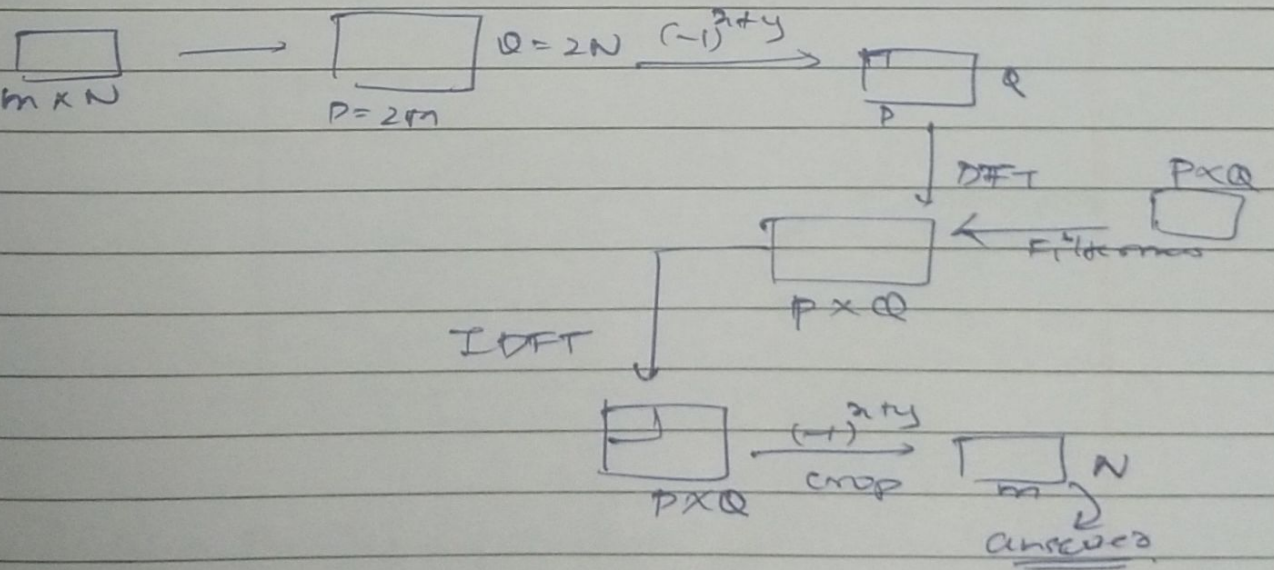
Method II



↳ The filter mask ~~is~~ ~~not~~ in this approach have not been padded & this may introduce "wraparound error".

↳ But still it is preferred over method I.

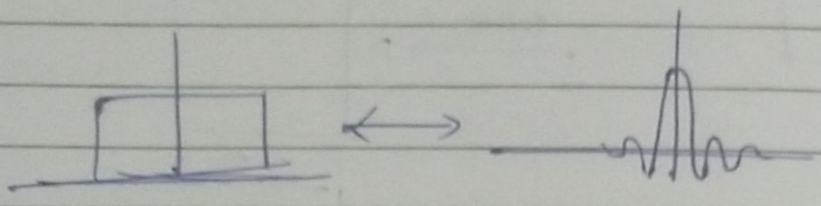
Frequency domain filtering:-



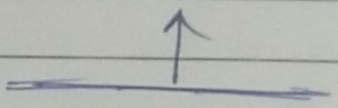
Date / /

Note:-

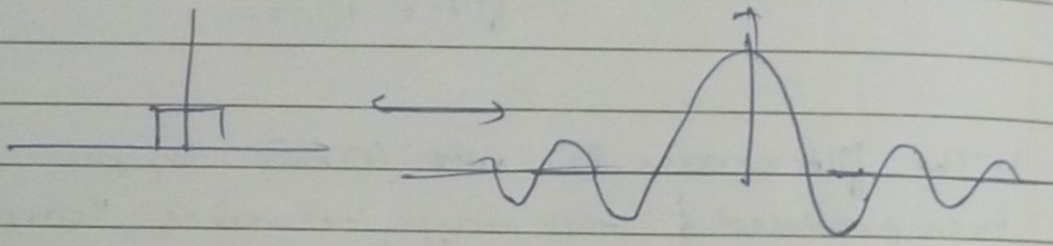
Ringing effect decreases as the width of the lowpass filter is increased



Frequency domain



limiting case

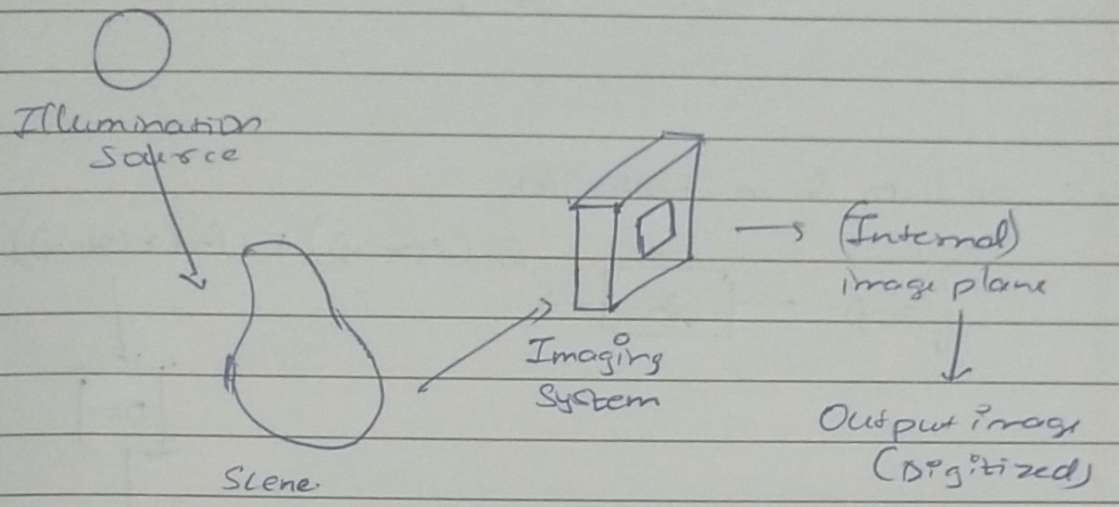


The filter that is used to remove ringing effects,

- Gaussian filter ✓✓
- Butterworth filter.

Date / /

Homomorphic Filtering

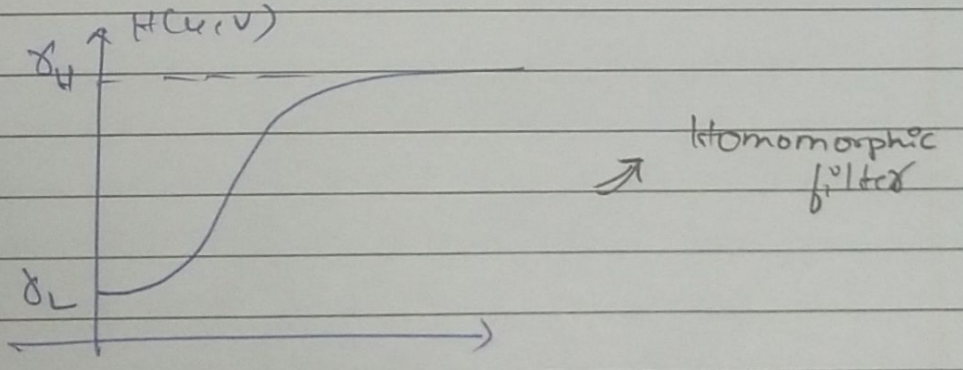


$$f(x,y) = i(x,y) \cdot r(x,y)$$

$$0 < f(x,y) < \infty \quad \& \quad 0 \leq r(x,y) \leq 1$$

$R(u,v) \Rightarrow$

$I(u,v) \Rightarrow$



↳ Reduce the intensity range ⇒

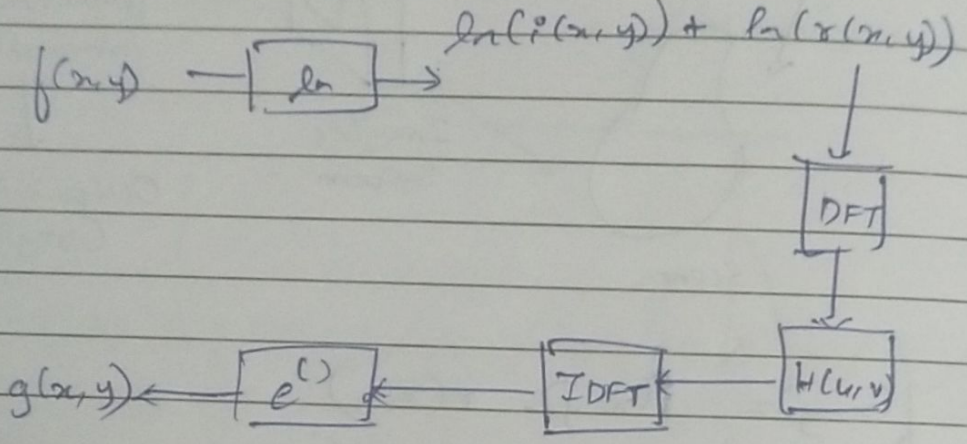
↳ Increase sharpening ⇒

Date / /

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

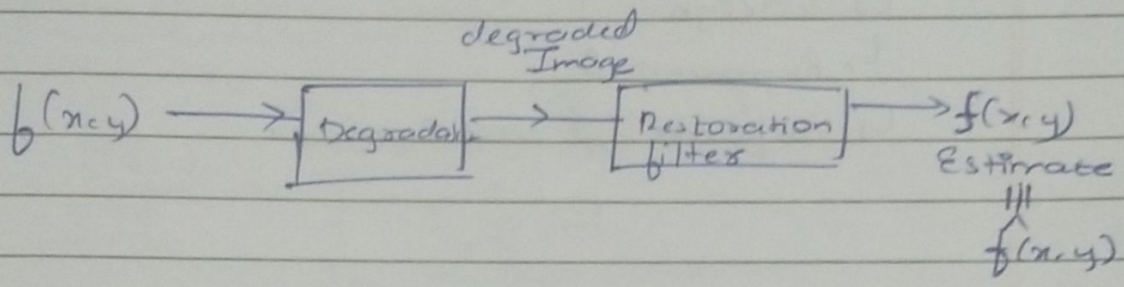
can't apply linear filtering because

So we take ln on both sides.



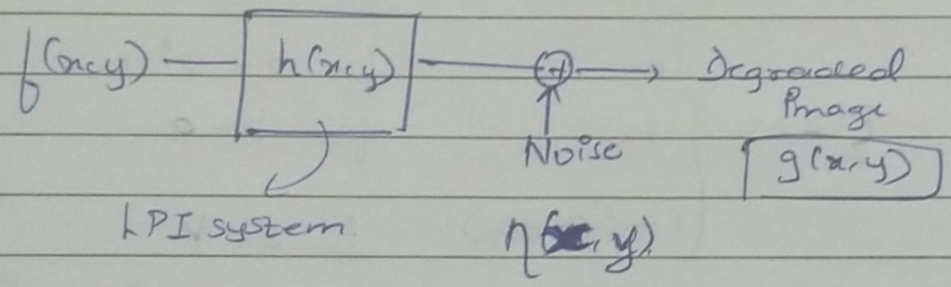
Date / /

Image Restoration:



Degradation Model:-

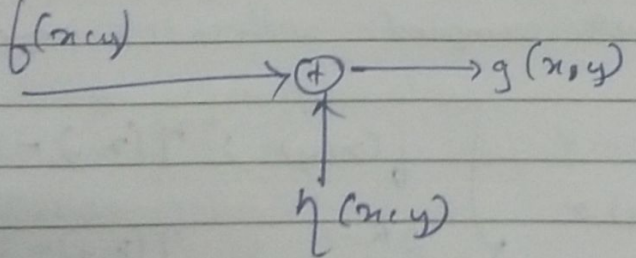
(i)



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

Linear Convolution

(ii)



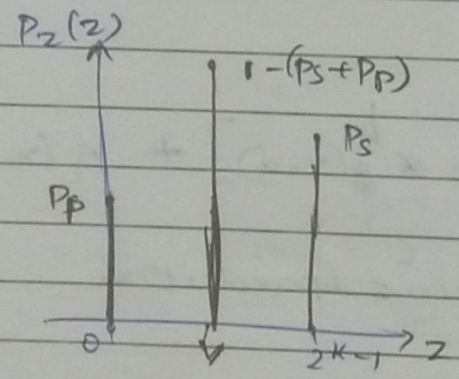
We model $\eta(x,y)$ as a random variable.

Date / /

Some ways of denoting the noise signals are
- Gaussian
- Reileigh
- Exponential
- Uniform & Salt & pepper

Salt and Pepper

$$P_z(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \\ P_p & \text{for } z = 0 \\ 1 - (P_s + P_p) & \text{for } z = V \end{cases}$$



So, for the noise sampled with salt & pepper pdf's given by $g(x,y)$ where $\hat{}$

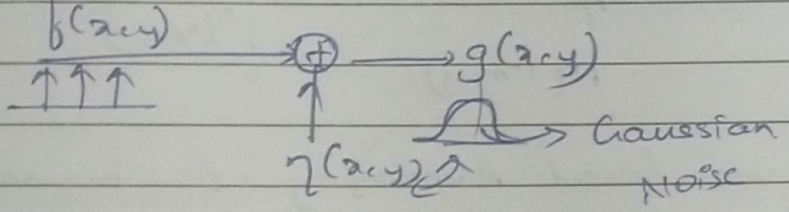
$$g(x,y) = \begin{cases} f(x,y) ; \eta(x,y) = V \\ 2^k - 1 ; \eta(x,y) = 2^k - 1 \\ 0 ; \eta(x,y) = 0 \end{cases}$$

Random places has black / white spots

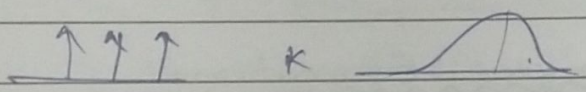
Date / /

~~Gaussian PDF~~

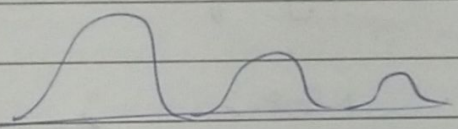
Histogram of Noisy Image



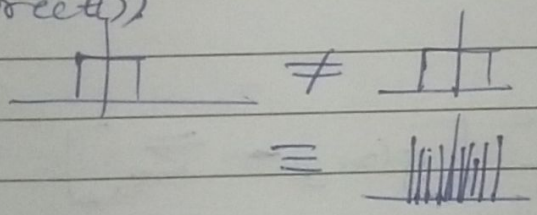
$g(x,y) = Pdf(f(x,y)) * Pdf(\eta(x,y))$



pdf of g(x,y) =



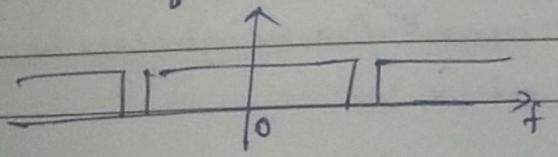
(rect(t))



Periodic Noise:

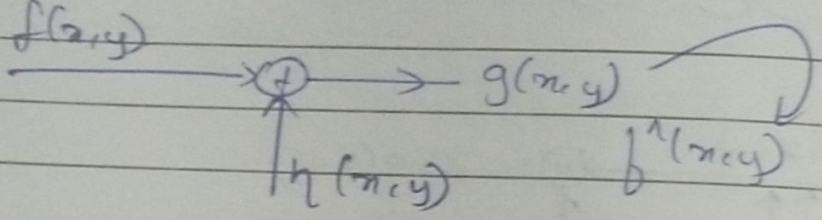
↳ Some correlation within the pixels of the noise $\eta(x,y)$ this noise

↳ removing is Easier, by using a band-stop filter (Notch filter)



Date / /

Removing any General Noise:



↳ we can use local Averaging can be used Image Enhancement.

Noise parameter Estimation:

↳ Take a relatively flat area ⇒ histogram that area

Noise Variance ⇐ Approximated the Noise η(x,y)

for any Additive Noise

(P) Adaptive filter

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_n^2}{\sigma_{S_{xy}}^2}\left[g(x,y) - \bar{z}_{S_{xy}}\right]$$

σ_n^2 → Global Noise Variance

$\sigma^2_{S_{xy}}$ ⇒ Variance of a small neighbourhood centered at (x,y) a.k.a local Variance

$\bar{z}_{S_{xy}}$ ⇒ local mean of a neighbourhood centered at (x, y)

$\frac{\sigma_n^2}{\sigma_{S_{xy}}^2}$ ⇒ it is between [0,1]

Date / /

(ii) Changing the window size:

↳ At edges, reduces the ~~window~~ neighbourhood size, so that the details are not blurred. And in the remaining areas, A larger neighbourhood is considered for noise removal.

(iii) Median filtering (Enhancement). Image

↳ Particularly useful in salt & pepper noise images.

↳ The window size ↑, Bluriness in the sharp edges ↑↑

↳ very effective if the no of pixels affected ≤ 20%.

↳ If the % of affected pixels is larger, then window size needs to be increased

This inturn causes bluriness (A compromise)

(iv) Adaptive Median filtering

Z_{min}: minimum intensity value in S_{xy}

Z_{max}: maximum intensity value in S_{xy}

Z_{med}: median of intensity values in S_{xy}

Z_{xy}: intensity at coordinates (x, y)

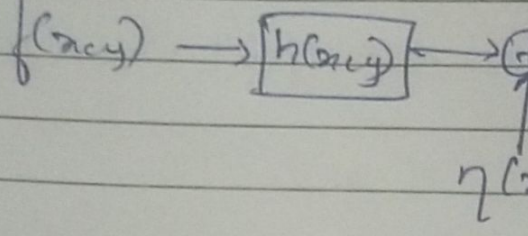
S_{max}: Max allowed size of S_{xy}

Level A: If $Z_{min} < Z_{med} < Z_{max}$, go to level B
Else increase size of S_{xy}
If $S_{xy} \leq S_{max}$, repeat level A.
Else output Z_{med}.

Level B: If $Z_{min} < Z_{xy} < Z_{max}$, output Z_{xy}
Else output Z_{med}.

Date / /

Blind & Non-blind Restoration:



$= F(u,v) \cdot H(u,v) + \eta(u,v)$

Non-blind : ⇒ we know $h(x,y)$

Blind. Estimate of $h(x,y) \Rightarrow \hat{h}(x,y)$ is used

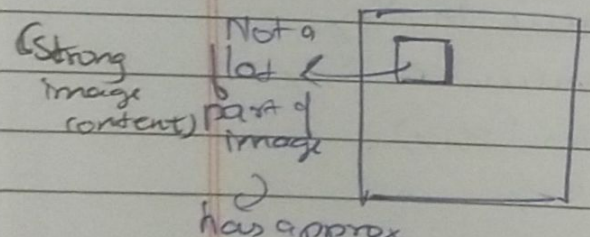
Different approaches to get $\hat{h}(x,y)$

(i) Observation

(ii) Experimentation

(iii) Modeling

(i) Observation:



$$G_s(u,v) = H(u,v) \cdot \hat{F}(u,v)$$

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

Date / /

(ii) Experimentation:-

If degradation is caused by the camera

{Camera, Image} ⇒ Estimation of the degradation function.

We pass a test image having Impulse structure (keep the radius as small as possible)

⇒ $h(x,y)$ (Point spread function) PSF

(iii) Modeling

Ex: Blurring due to object motion

T ⇒ Shutter opening (or) closing time

The general expression for pixel values due to object motion

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

$g(x,y)$ → Blurred Image

f → original image.

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

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

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

Date / /

$$G(u,v) = \int_0^T F(u,v)\, e^{-j2\pi [x_0(t).u + V y_0(t)]}\, dt$$

$$G(u,v) = F(u,v) * \int_0^T e^{-j2\pi (x_0(t).u + y_0(t)v)}\, dt$$

we have

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

$$\therefore H(u,v) = \int_0^T e^{-j2\pi (x_0(t)u + v.y_0(t))}\, dt$$

Let's assume that from 0 - T, let the object move from 0 to a horizontally & 0 to b vertically in a linear fashion. So,

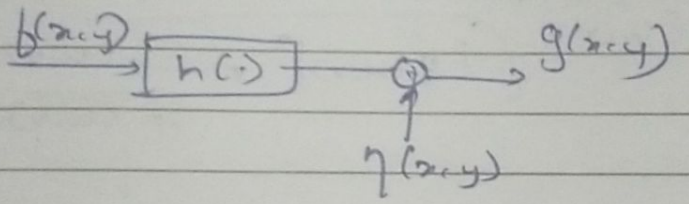
$$\boxed{x_0(t) = \frac{at}{T}, \quad y_0(t) = \frac{bt}{T}}$$

∴ In this linear motion case

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

Date / /

a) Inverse filtering



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

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

Even though, Noise is very negligible, but on dividing with H(u,v) it gets amplified.

H(u,v) is typically a low pass fn, it has very smaller values at high frequencies. So at high frequency N(u,v) will be amplified

↳ In Inverse filtering, we neglect the certain frequencies outside a fixed range
↳ Manual filtering.

b) Wiener Filtering:
↳ Minimum mean Squared Error (MMSE). Filtering

i.e minimise $E[(f(\cdot) - f''(\cdot))^2]$.

↳ For low & Medium noise.

Date / /

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

$S_N(u,v)$ ⇒ Power spectrum of Noise

$S_F(u,v)$ ⇒ power spectrum of original image

If the original image is $F(u,v)$, then power spectrum is $|F(u,v)|^2$.

Estimating $S_N(u,v)$, $S_F(u,v)$:

In most cases, the noise is white gaussian Noise

$\therefore S_N(u,v)$ = constant ⇒ $N_0/2$

Or another way is,

$$\frac{S_N(u,v)}{S_F(u,v)} = K \text{ (a constant)}$$

thus,

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

Date / /

Image Transforms

Vector space:-

↳ A set of vectors make a vector space 'V' if it satisfies the following properties

(i) $u, v \in V$ & $u+v \in V$

(ii) $\alpha u \in V$

(iii) $u \in V, -u \in V$

Inner Product

(i) Dot product:

Let u & v be N-dimensional vectors, given by,

$u = (u_0, u_1, u_2, \ldots u_{N-1})$

$v = (v_0, v_1, v_2, \ldots v_{N-1})$

$$u \cdot v = \sum_{i=0}^{N-1} u_i \cdot v_i \qquad \langle u, v \rangle = \sum_{i=0}^{N-1} u_i^* \cdot v_i$$

(complex)

It maps 2 vectors from vector space to a integer value is inner product.

Dot product is one such inner product.

$\langle u, v \rangle$

Properties

(i) $\langle u, u \rangle \geq 0$

(ii) $\langle u, v \rangle = \langle v, u \rangle^*$

(iii) $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$

(iv) $\langle u+v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$

↳ Any fn that satisfies these above condtn, it belongs to inner product

Date / /

Inner Product Space:

↳ A vector space 'V' along with an "inner product".

Ex: Euclidean space ($\mathbb{R}^N$), $\mathbb{C}^N$

$$u = (u_0, u_1, \dots u_{N-1})$$

If u_i are real, then u, v is $\mathbb{R}^N$; if u_i is imaginary, then u, v is $\mathbb{C}^N$.

Length of a vector

Length of a vector 'u' in a vector space is given by,

$$\|u\|^2 = \langle u, u \rangle$$

$$|u| = \sqrt{u_0^2 + u_1^2 + \dots u_{N-1}^2}$$

Angle between vectors

$$\theta = \cos^{-1}\left[\frac{u \cdot v}{\|u\| \|v\|}\right]$$

Basis Vectors:

Let's consider an N-dimensionless vector space 'V', any set of vectors $\{w_0, w_1, \dots w_m\}$ such that, any vector in 'V' can be expressed as a linear combination of this set of vectors.

Date / /

for a N- dimensional vector space, there are N- basis vectors.

$$u = \alpha_0 w_0 + \alpha_1 w_1 + \cdots \alpha_{N-1} w_{N-1}$$

The Basis is called "orthogonal Basis" if,

$$\langle u_i, v_j \rangle = 0, \quad i \neq j.$$

↳ Orthonormal basis is an Orthogonal basis which has the magnitudes of all the basis vectors as unity.

↳ Easier to get the values of α_is.

$$u = \sum_{i=0}^{N-1} \alpha_i w_i.$$

$$\langle u, w_j \rangle = \alpha_j \quad \text{(assuming orthonormal vector basis).}$$

1D-DFT

$$x[n] \longrightarrow X[k]$$

0 to N-1 0 to N-1

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j}{N} kn}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j \frac{2\pi}{N} \cdot kn}$$

Date / /

$x[n] \longrightarrow X[k]$

0 to N-1

$$x = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} \qquad X = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}$$

N-dimensional vector space.

Basis vectors for above space =

$$\underset{N\times 1}{\underline{X}} = \underset{N\times N}{A} \; \underset{N\times 1}{\underline{x}}$$

Let $W_N = e^{j 2\pi/N}$

Then the transformation matrix 'A' is given by,

$$A = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ \vdots & & & & \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)(N-1)} \end{bmatrix}$$

forward transformation matrix.

$$\underset{N\times 1}{C} = \underset{N\times N}{A} \times \underset{N\times 1}{B}$$

$$\begin{bmatrix} c[0] \\ c[1] \\ \vdots \\ c[N-1] \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} * \begin{bmatrix} b[0] \\ b[1] \\ \vdots \\ b[N-1] \end{bmatrix}$$

Date / /

$$\underline{c} = \sum_{i=0}^{N-1} a_i \cdot b[i]$$

Basis vectors

∴ The basis vectors are the columns of the transformation matrix for $x[n]$.

$x[n]$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j\left(\frac{2\pi}{N}\right) kn}$$

$A^{-1} = \underline{\underline{A}}_{x[n]}$

inverse of forward transform matrix

$$\frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \cdots & & \omega_N^{N-1} \\ \vdots & & & & & \\ 1 & \omega_N^{(N-1)} & \cdots & & & \omega_N^{(N-1)(N-1)} \end{bmatrix}$$

∴ The basis vectors of this vector space is the columns of $\underline{\underline{A}}_{x[n]}$ (i.e A^{-1}).

Now, the inner product of two basis vectors ω_k & ω_e are

$$\langle \omega_k, \omega_e \rangle = \sum_{n=0}^{N-1} \left(e^{-j\frac{2\pi}{N} kn}\right) \cdot \left(e^{j 2\pi/N \cdot en}\right)$$

$$= \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}(k-e)n}$$

Date / /

for $k=l$, $\langle w_k, w_l \rangle = N$

for $k \neq l$, $\langle w_k, w_l \rangle = \sum_{n=0}^{N-1} e^{-j\left(\frac{2\pi}{N}\right)(k-l)n}$

Geometric series

$$= \frac{1 - e^{-j(2\pi/N)(k-l)(N)}}{1 - e^{-j(2\pi/N)(k-l)}}$$

$$= 0$$

$$\therefore \langle w_k, w_l \rangle = \begin{cases} 0, & k \neq l \\ N, & k = l \end{cases}$$

∴ These basis vectors are orthogonal but it is not orthonormal vectors, because its magnitude is not 1.

To make them orthonormal basis, we consider the following,

$$X[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x[n] \, e^{j\left(\frac{2\pi}{N}\right)kn}$$

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[k] \, e^{j(2\pi/N) \cdot kn}$$

Date / /

Properties (General)

(i) $A = A^T$

(ii) $A = (A^*)^T$ ↳ Hermitian matrix.

(iii) $A^{-1} = A^T$ ↳ Orthogonal matrix

(iv) $A^{-1} = (A^*)^T$ ↳ Unitary matrix

In our case,

$$x = A^T X, \quad X = Ax.$$

We can see that, A is an "Unitary Matrix". Thus, we call it "Unitary transform"

Orthogonal Matrix:-

$$A^{-1} = A^T$$

(or) $$I = A \cdot A^T$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & & & \\ 0 & 0 & 0 & \cdots 1 \end{bmatrix} = \begin{bmatrix} \cdots & a_0^T & \cdots \\ \cdots & a_1^T & \cdots \\ & \vdots & \\ \cdots & a_{N-1}^T & \cdots \end{bmatrix} \begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ | & | & & | \end{bmatrix}$$

$$\Rightarrow a_i \, a_j^T = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

∴ The columns and rows of an orthogonal matrix are "orthonormal"

Date / /

Rank

↳ No of Independent rows & columns

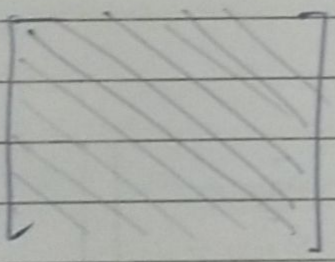
↳ for a matrix to be invertible, it must have full rank

$$\underset{N\times 1}{X} = \underset{(N\times M)}{A} \cdot \underset{M\times 1}{z}$$

↳ This is not invertible

Toeplitz matrix

↳ If all the elements in a diagonal (i.e main (or) sub diagonal) are same, then it is a Toeplitz matrix



$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

Circulant Matrix:

↳ Each ~~below~~ row is a right shifted row of the row above

i.e $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}$

↳ This is also Toeplitz

All circulant matrices are Toeplitz but not all Toeplitz matrices are circulant

Date / /

let $\underline{X_1} = A\underline{x_1}$ & $\underline{X_2} = A\underline{x_2}$

Now, $\langle X_1, X_2 \rangle = \underset{1\times N}{(x_1^*)^T} \cdot \underset{N\times 1}{X_2}$

$\Rightarrow ((A\underline{x_1})^*)^T \cdot (A\underline{x_2})$

$= (\underline{x_1}^*)^T (A)^{*T} \cdot A \cdot \underline{x_2}$

Since 'A' is an unitary matrix, $A^{-1} = (A^*)^T$

(or) $I = A \cdot (A^*)^T$

$\langle X_1, X_2 \rangle = (\underline{x_1}^*)^T (\underline{x_2})$

If $(x_1 = x_2)$, then we can say that,

$\langle X_1, X_2 \rangle \Rightarrow \|x_1\|^2$.

Note

↳ Unitary transform will conserve the length of the vector, i.e the energy of a field.

Notations:

$f(x) \longrightarrow T(u)$

$$T(u) = \sum_{x=0}^{N-1} f(x)\, r(x,u)$$ ⇝ forward transformation kernel

$$f(x) = \sum_{x=0}^{N-1} T(u) \cdot s(x,u)$$

↳ Inverse transformation kernel.

Date / /

for DFT,

$$r(x,u) = \frac{1}{\sqrt{N}} \cdot e^{-j\frac{2\pi}{N}xu}$$

$$s(x,u) = \frac{1}{\sqrt{N}} \cdot e^{j\frac{2\pi}{N}xu}$$

$$f(x) = \sum_{u=0}^{N-1} T(u) \cdot s(x,u)$$

Then the basis vector is given by.

$$B_v(u) = \begin{bmatrix} s(0,u) \\ s(1,0) \\ \vdots \\ s(N-1,u) \end{bmatrix}^T$$

$u = 0$ to $N-1$.

$$\underline{T} = A \cdot \underline{f}$$

$$\underline{f} = A^{-1} \cdot \underline{T}$$

2D

$$f(x,y) \longrightarrow T(u,v)$$

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

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

Date / /

basis vectors for 2D:

Separable transform

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

$s(x,y,u,v) = s_1(x,u) \cdot s_2(y,v)$

If $r_1 = r_2$ → symmetric & separable

Note

If a matrix is symmetric & separable,

$G = A f A^T$

$N \times N \quad N \times N \quad N \times N$

(i) column-wise

$$[G] = [A][f]$$

$N \times N \quad N \times N$

(ii) Row wise

$$T = (A G^T)^T$$

$T^T = (A G^T)^T$

$T = G \cdot A^T$

$$T = A f A^T$$

Inverse

is for a unitary matrix, (i.e $A^{-1} = (A^*)^T$)

$$f = (A^*)^T T A^*$$

Date / /

General case

for a matrix f of size "M×N",

In general the Transformation matrix is given by

$$T = A_M \cdot f \cdot A_N^T$$

$$f = (A_M^*)^T \, T \, (A_M)^*$$

Transform Images from the Given matrix:

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

$$F = (A^*)^T \, T(A^*)$$

$$\Rightarrow \begin{bmatrix} a_0 \\ \vdots \\ \end{bmatrix} \begin{bmatrix} T(0,0) & \cdots & T(0,N-1) \\ \vdots & & \vdots \\ T(N-1,0) & \cdots & T(-) \end{bmatrix} \begin{bmatrix} r_0 \\ \vdots \\ \end{bmatrix}$$

N×N N×N N×N

$$\begin{bmatrix} a_0 \\ \vdots \\ \end{bmatrix} \begin{bmatrix} r_0 \\ \vdots \\ \end{bmatrix}$$

Then the corresponding basis image is given by for (0,0)

$$\Rightarrow \begin{bmatrix} f \\ \vdots \\ \end{bmatrix} * \begin{bmatrix} \quad \end{bmatrix}$$

1st column of $[A^*]^T$; 1st row of A^*

Date / /

for a general case, the basis image for the coefficient T(i,j) is

⇒ i^{th} column of $(A^*)^T$ * j^{th} row of (A^*).

Ex Given the transformation matrix,

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$ ↓ orthogonal matrix.

let the input image is

$$F = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$T = AFA^T$$

& $F = A^T T A$

The basis images are given by

$$T = \begin{bmatrix} T(0,0) & T(0,1) \\ T(1,0) & T(1,1) \end{bmatrix}$$

T(0,0)-

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$T(0,0) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

T(0,1)

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix},$$

Date / /

T(1,0)

$$\Rightarrow \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \times \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix}$$

T(1,1)

$$\Rightarrow \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \times \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}$$

verification

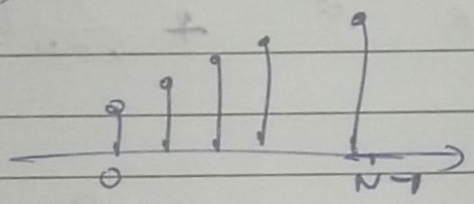
Date / /

2D-DCT

↳ Discrete Cosine Transform.

⇒ Drawbacks of DFT:

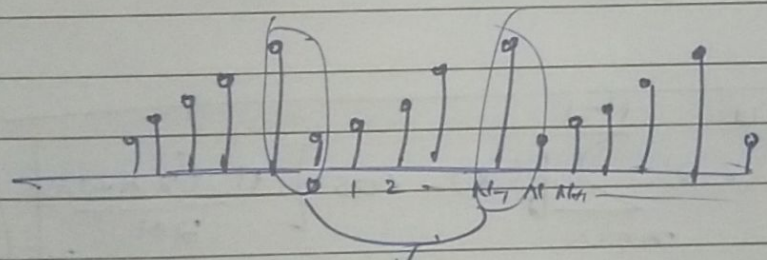
1 Dimensional case:



$$T(u) = \sum_{x=0}^{N-1} f(x)\, e^{-j2\pi/N xu}$$

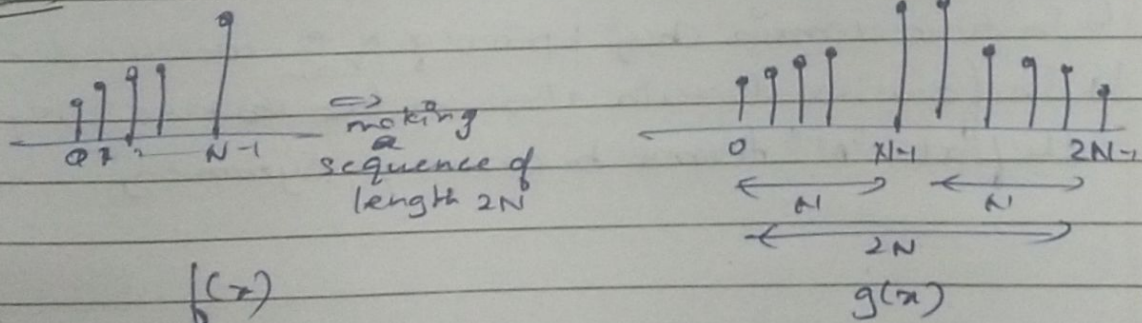
⇒ 1D-DFT.

Periodic extension of f(x):



High Frequencies, this will lead to distortion at the edges.

Soln



$$g(x) = \begin{cases} f(x), & x \in [0, N-1] \\ f(2N-1-x), & x \in [N, 2N-1]. \end{cases}$$

Date /

Thus the d/t of g(x) would be

$$G(u) = \sum_{x=0}^{2N-1} g(x) \cdot e^{-j 2\pi/2N (xu)}$$

$$= \sum_{x=0}^{N-1} f(x) \cdot e^{-j 2\pi/2N (xu)} + \sum_{x=N}^{2N-1} f(2N-1-x) \cdot e^{-j \frac{2\pi}{2N} (2N-1-x)u}$$

Let, $2N-1-x = y$

$$G(u) = \sum_{x=0}^{N-1} f(x) e^{-j 2\pi/2N (xu)} + \sum_{y=0}^{N-1} f(y) \cdot e^{-j \frac{2\pi}{2N} (2N-1-y)u}$$

$$G(u) = \sum_{x=0}^{N-1} f(x) \left[e^{-j \frac{2\pi}{2N} xu} + e^{+j \frac{2\pi}{2N} (x+1)u} \right]$$

$$G(u) = e^{j \frac{\pi}{2N} u} \cdot 2 \sum_{x=0}^{N-1} f(x) \cdot \cos\left[\frac{\pi (2x+1)u}{2N} \right]$$

DCT is used to overcome the high frequency issues in DFT

i) In time domain only 1 period of N is considered instead of "2N"
ii) In fourier domain also we can take only the first "N" terms because of symmetry

Date / /

$$T(u) = \sum_{x=0}^{N-1} \alpha(u) \cdot f(x) \cdot \cos\left[\frac{(2x+1)\pi u}{2N}\right].$$

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

↳ Normalising term

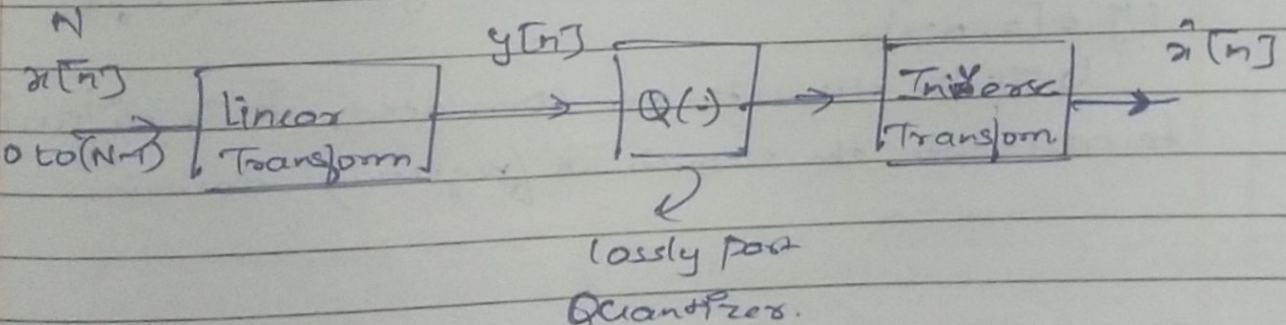
$$\alpha(u) = \begin{cases} \sqrt{1/N}, & u=0 \\ \sqrt{2/N}, & u \neq 0 \end{cases}$$

Note
↳ If $f(x)$ is real, $T(u)$ is also real.

Image Coding

Image compression can be lossy (or) lossless

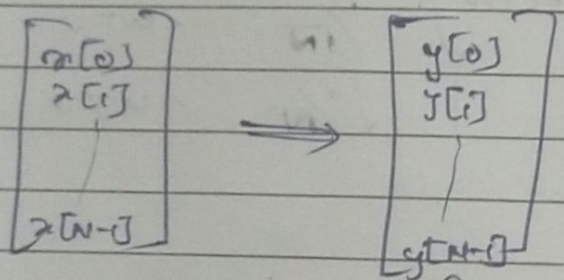
Transform coding:
↳ Lossy coding



Date / /

1. Energy compaction
2. Decorrelation

1. Energy Compaction:



If there are very few elements that have most of the energy concentrated in them

↳ It can be done such that those elements with less energies can be approximated as '0'.

(ii) Decorrelation:

↳ No correlation between y[i] & y[j] for i,j ∈ [0, N-1].

↳ If the elements are correlated, it is difficult to calculate the elements in frequency domain

Correlation:

x[0], x[i], ——— x[N-1]

X[i] - X[i] - X[N-1]

↳ Random variable

Date / /

Assumption

⇒ The random variables have zero mean & are wide sense stationary (WSS) random variables

Thus the covariance matrix is given by,

$$E[x \cdot x^T]$$

$$x = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$\therefore E[x \cdot x^T] = \begin{bmatrix} E[x[0] \cdot x[0]] & \cdots & E[x[0] x[N-1]] \\ E[x[1] \cdot x[0]] & \cdots & E[x[1] \cdot x[N-1]] \\ \vdots & & \\ E[x[N-1] x[0]] & \cdots & E[x[N-1] \cdot x[N-1]] \end{bmatrix}$$

$$C_x = \begin{bmatrix} R_x[0] & R_x[1] & \cdots & R_x(N-1) \\ R_x[1] & R_x[0] & & R_x[N-2] \\ \vdots & \vdots & & \vdots \\ R_x[N-1] & R_x[N-2] & \cdots & R_x[0] \end{bmatrix}$$

Note

↳ C_x is a Toeplitz matrix & a Symmetric matrix

↳ Real valued matrix

Date / /

↳ For decorrelated input $R_x[\tau], \tau \neq 0) = 0$, then for a decorrelated input, the covariance matrix C_x is a diagonal matrix.

(*) Note

If a matrix is a real valued & symmetric square matrix of order $N \times N$, then it has N- distinct eigen values

⇓

N- distinct Eigen vectors.

and the N- eigen vectors form an "orthogonal" set.

Let the Eigen vectors of C_x be $P_0, P_1, P_2 \ldots P_{N-1}$, then

$$C_x \cdot P_i = \lambda_i P_i$$

↳ We have forward & reverse linear transformation

$$Y = AX$$
$$X = A^{-1}Y$$

Let A^{-1} be that,

$$A^{-1} = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ P_0 & P_1 & \cdots & P_{N-1} \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}$$

Date / /

If $A^{-1} = A^T$,
then A is given by

$$A = (A^{-1})^T$$

$$= \begin{bmatrix} \leftarrow & \phi_0^T & \rightarrow \\ \leftarrow & \phi_1^T & \rightarrow \\ & \vdots & \\ \leftarrow & \phi_{N-1}^T & \rightarrow \end{bmatrix}_{N \times N}$$

$$E[Y \cdot Y^T] = C_Y$$

$$\Rightarrow E[AX \cdot (AX)^T]$$

$$= E[AXX^TA^T]$$

$$\therefore E[YY^T] = C_Y = \boxed{E[A \, C_X \, A^T]}$$

$$= E\left[A \cdot \left[\lambda_0 \phi_0 \;\; \lambda_1 \phi_1 \;\; \cdots \;\; \lambda_{N-1} \phi_{N-1}\right]\right]$$

$A \Rightarrow$ rows are EoV

$A^T \Rightarrow$ columns are EoV

$$(C_X A^T).$$

$$C_Y \Rightarrow A\left[\lambda_0 \phi_0 \;\; \lambda_1 \phi_1 \;\; \cdots \;\; \lambda_{N-1} \phi_{N-1}\right] \Rightarrow \begin{bmatrix} \lambda_0 & \cdots & 0 \\ 0 & \lambda_1 & \\ & & \ddots \\ & & \lambda_{N-1} \end{bmatrix}$$

Ideal case.

"KL Transform"
↳ Karhunen-Loeve Transform

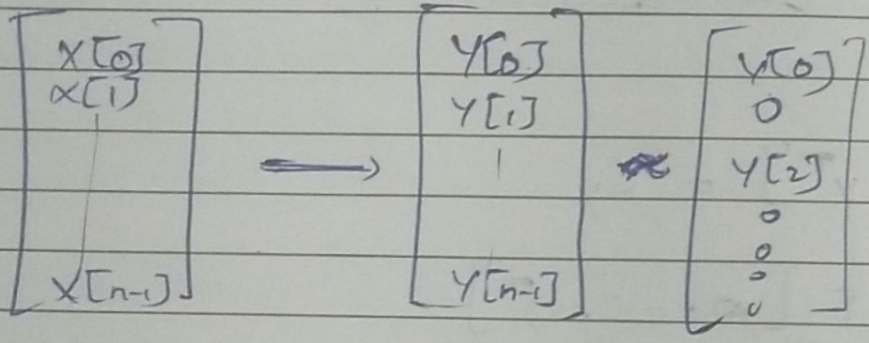
↳ It is an optimal transformation in terms of decorrelation.

Date / /

Disadvantages

- Here the transformation matrix C_x will depend on the input image, unlike DFT, DCT

Energy Compaction:



- Energy of the system is preserved in an Unitary transformation
- Most of signal energy is concentrated in a few elements
  - Even when we make the remaining elements zero, it won't make a large difference.

Note

In terms of Energy Compaction also "KL-Transform" is the optimal transform (i.e. minimum Mean squared Error $E[(x - \hat{x})^2]$)

Note

DCT, DFT → Both have similar energy consumption, Complexity etc.

- DCT is preferred for Image compressing techniques.

Date / /

Hadmard Transformation:

↳ Walsh- Hadmard Transformation

↳ N×N , N is always a power of 2 (i.e 2^n)

Ex. 2×2, 4×4, 8×8

↓ ↓ ↓

H_2 H_4 H_8

$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$ ↳ orthogonal ~~transpose~~ matrix.

Kronecker Product:

A B

$N_1 \times N_2$ $M_1 \times M_2$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N_2} \\ a_{N_1 1} & a_{N_1 2} & \cdots & a_{N_1 N_2} \end{bmatrix} B$$

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1N_2}B \\ a_{N_1 1}B & a_{N_1 2}B & \cdots & a_{N_1 N_2}B \end{bmatrix}$$

Sub-matrix

$N_1M_1 \times N_2M_2$

Date / /

$$H_N = H_{N/2} \otimes H_2$$

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = H_2 \otimes H_2$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\Rightarrow \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$ → orthogonal matrix //

↳ All the Hadamard matrixes are orthogonal matrix

Date / /

Image coding:
Compression

QF
↳ 0 to 100

Basic idea ⇒ Exploit Redundancy.

Redundancies can be of the type :-

(i) Spatial Redundancy (or)
Temporal Redundancy (in case of videos)

(ii) Psychovisual Redundancy.

(iii) Coding Redundancy.

Coding Redundancy:

Let r_k denote an intensity level from 0 to 255.

∴ 8 bits are required to store a value for a pixel

512 × 512 Image

r_k	freq. (normalized)	Variable length coding
87	0.25	01
128	0.47	1
186	0.25	000
255	0.03	001

Average codeword length = $1 \times 0.47 + 2 \times 0.25 + 3 \times 0.25 + 0.03 \times 3$

$= 1.81$

Date / /

↳ This is a lossless compression.

HD

↳ for 1 sec, 30 frames.

Size ⇒ (1920 × 1080) × (30) × 3 × 8
(3 → RGB, 8 → bits)

Size ≈ 1.5 Gbps

Psychovisual Redundancy:

↳ Human eye can't detect small small errors. This is exploited for compressing images.

Temporal Redundancy:

Spatial correlation:

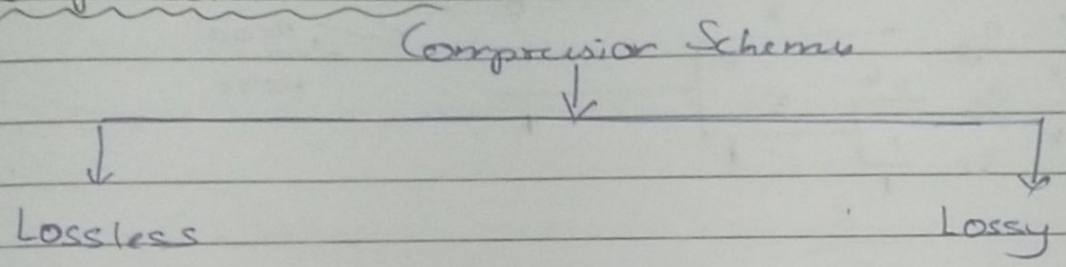
↳ The value of each pixel is closely depends on the values of the neighbouring pixels ⇒ Natural Images.

Temporal redundancy:

↳ Dependence between 2 frames of a video

Date / /

Compression Schemes:



Lossless Coding Schemes:

(i) Huffman Coding Scheme:

7 x 7 image and 4 intensity levels

	P_r
S_0	$4/49$
S_1	$1/49$
S_2	$36/49$
S_3	$8/49$

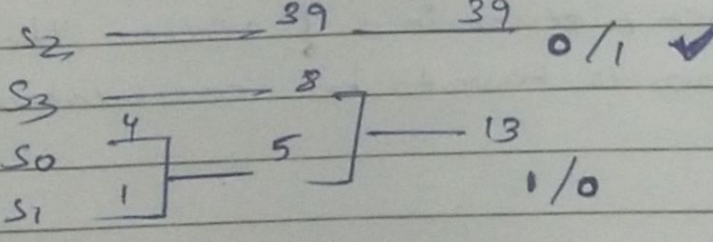
Huffman coding assigns a variable length code for each of the intensity levels.

Procedure:

(i) arrange the intensity in the decreasing order of probability

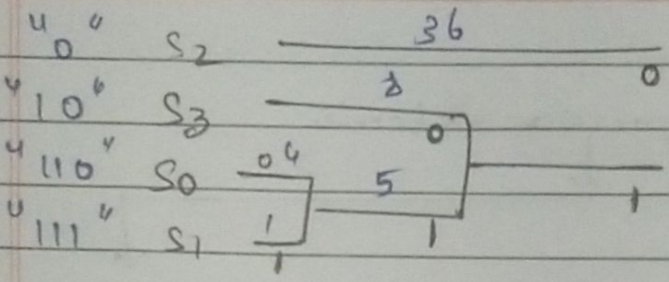
S_2, S_3, S_0, S_1

(ii) Merge two symbols having least 2 probabilities



over and over.

Date / /



The average codeword length:

$$\Rightarrow 1 \times \frac{36}{49} + 2 \times \frac{8}{49} + 3 \times \frac{4}{49} + 3 \times \frac{1}{49}$$

$$\boxed{L \Rightarrow 1.36 \approx 1.4}$$

↳ This is a uniquely decodable variable length coding.

Note

↳ If after merging the new value is already existing, you can place the new term anywhere from the first among them or to the last among them. But make sure that same convention is used throughout the process.

Drawbacks

↳ Both the encoder & decoder must know the keys for each image

↳ for each picture in a f video, the probabilities & the keywords have to be calculated again & again.

Date / /

(i) Compression Ratio:

Original representation = b

Compressed representation = b'

$$\text{Compression ratio}(c) = \frac{b'}{b}$$

(ii) Redundancy

$$R = \left(\frac{b - b'}{b}\right).$$

Prefix code:- ⇒ If a codeword is a prefix of another codeword.

Symbol	Prob.	length
a_1	P_1	L_1
a_2	P_2	L_2
a_n	P_n	L_n

$$\text{Average code length } (\bar{L}) = \sum_{i=1}^{n} P_i \cdot L_i$$

The minimum possible value of $(\bar{L})$:

From Shannon's Theorem, the Entropy is given by,

$$H \Rightarrow \sum_{i=0}^{N} P_i \log\left[\frac{1}{(P_i)}\right]$$

$$\text{Self Entropy} \Rightarrow \log\left[\frac{1}{(P_i)}\right]$$

Date / /

↳ less frequent images carry more information and have more randomness.

from shannon's theorem, the average codeword length cannot be lower than it.

$$H = \sum_{i=0}^{N} P_i \log\left[\frac{1}{(P_i)}\right].$$

Instantaneous Code:

↳ In order to decode the bit, the future bit is not required.

Ex

0
10
110
111

↳ Huffman Coding is an Instantaneous Code.

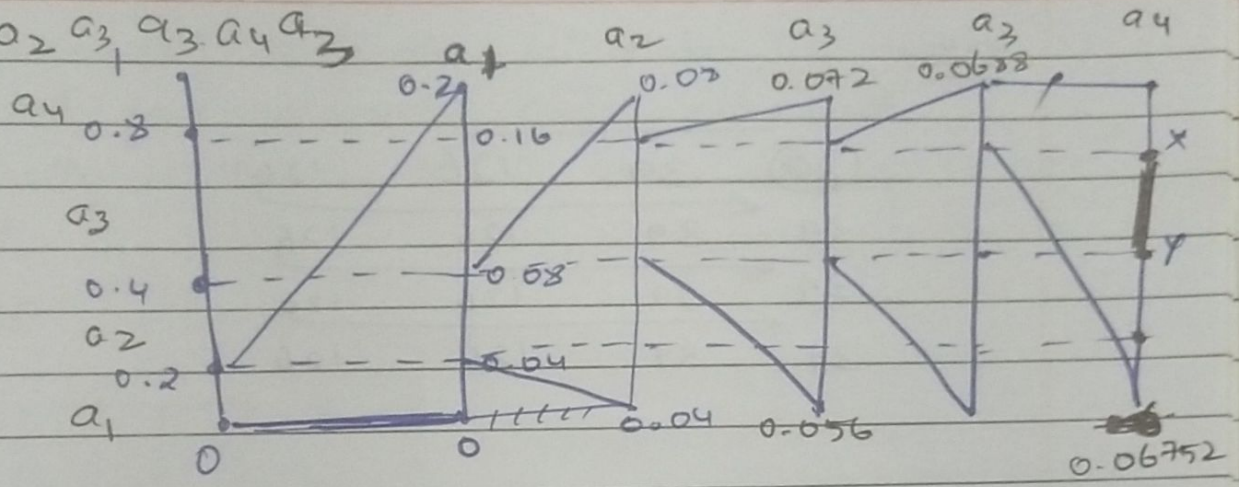
(ii) Arithmetic Coding:

Symbols	Probability
a_1	0.2
a_2	0.2
a_3	0.4
a_4	0.2

↳ Codeword is assigned to a sequence of symbols. For example, a single codeword for "$a_1 a_2 a_3 a_3 a_4$"

Date / /

Ex: $a_1 a_2 a_3 a_3 a_4 a_3$



A value between x & y is chosen, let's say it is "0.068"

Then, we transmit 0.068 in place of "$a_1 a_2 a_3 a_3 a_4 a_3$".

(i) One drawback is that we don't know when to stop.. if we have to know the length of the codeword transmitted prior).

↳ to overcome this, we use "a_e", to denote the end of the codeword

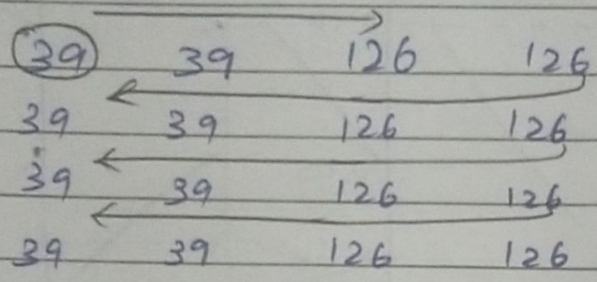
(iii) LZW coding

↳ Lempel - Ziv - Weltch

Let's consider an example where we have a 4x4 image with grey pixels of length 8 bits (i.e values from 0 to 255).

Date / /

4×4



↳ Before LZW coding, we have 256 entries already (0 to 255).

Location	Sequence
0	0
1	⋮
⋮	⋮
255	255
256	39-39
257	39-126
258	126-126
⋮	⋮
259	126-39

39-39-126 ⇔ 260

Currently recognised	Pixel being Processed	Encoded O/P
	39	
39	39	39
39	126	39
126	126	126
126	39	126
39	39	
39-39	126	256
⋮	⋮	⋮

Date / /

(b) In the decoder's side, the dictionary is not required for decoding the codeword.
Usually dictionary size is 4096, i.e of 12 bit.

Decoding

Ex

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

39	39	126	126	256	258	260	259	257	126

Process

Recognized seq.	Encoded val.	Pixel val.
	39	39
39	39	39
39	126	126
126	126	126
126	256	39-39
256	258	126-126
258	260	39-39-126

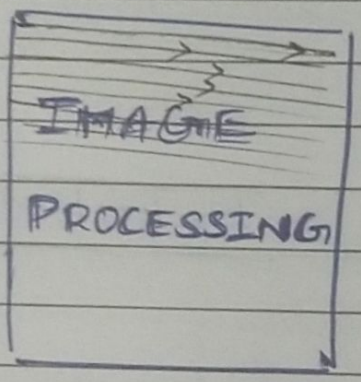
New Dictionary

256 → 39-39
257 → 39-126
258 → 126-126
259 → 126-39
260 → 39-39-126

Date / /

Run length coding

- For example consider a black text on a white background.

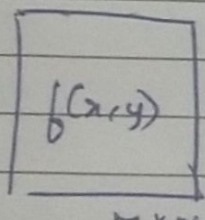


The encoding of the Run length coding is,

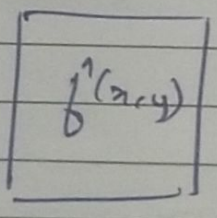
(255,1000), (0,10), (255, 100)

(255,1000): pixel value; No of pixels that are continuous & has that pixel value

LOSSY COMPRESSION:



Original Image



Compressed Image

Mean squared Error $\Rightarrow \frac{1}{MN} \sum_x \sum_y \left(f(x,y) - f'(x,y)\right)^2$

Date / /

→ Objective Measure

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right)$$

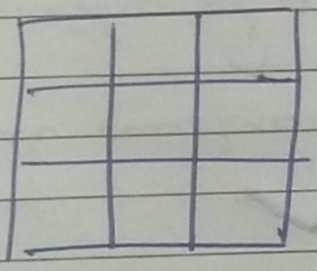
$$(or) \Rightarrow 10 \log_{10} \left(\frac{\max(f(x,y))^2}{MSE} \right).$$

Subjective measures

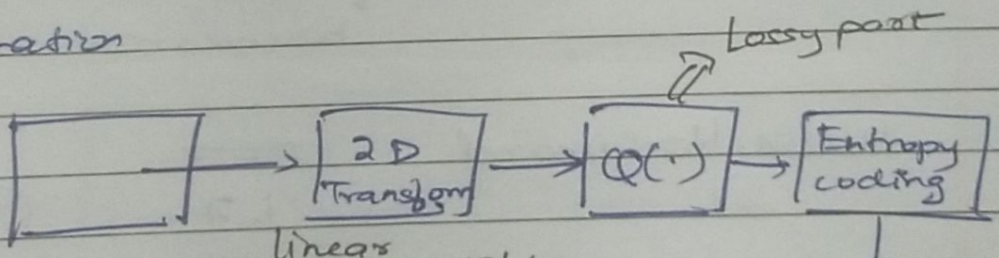
↳ Rated by humans on a scale of 1 to 5 & is taken average of it.

Block Transform Coding

(i) Dividing the given image into blocks



(ii) Each subblock is trasformed using a 2D transformation



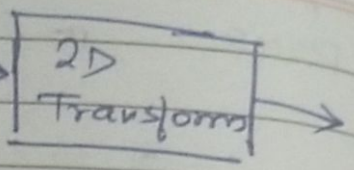
(iii) Quantize

(iv) Entropy Coding

Date / /

Choosing proper Transform:

Criteria desired:-



(i) Energy Compaction

↳ Energy is compacted into a few values

↳ which is only due to Spatial Correlation

(ii) Decorrelation

↳ The transform coefficients are not correlated

↳ The best transform for maximum compression is "KL transform"

↳ But KLT is not used because the transform matrix is different for different images.

↳ So, the most used transforms are "DFT" and "DCT" ✓

↳ Because of the distortions at the boundaries.

Walsh. Hadamard	DFT	DCT
2.1	1.7	(1-3) ✓

↳ DCT is better in Energy compaction than the other 3 transforms.

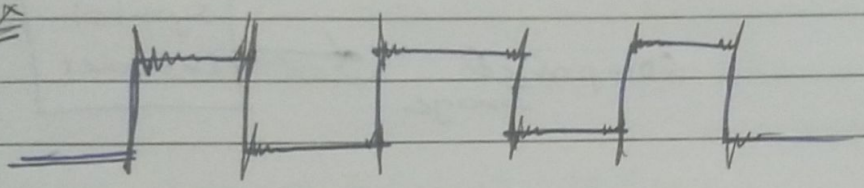
Date / /

Gibb's Phenomena:-

⇒ Truncation - Quantization

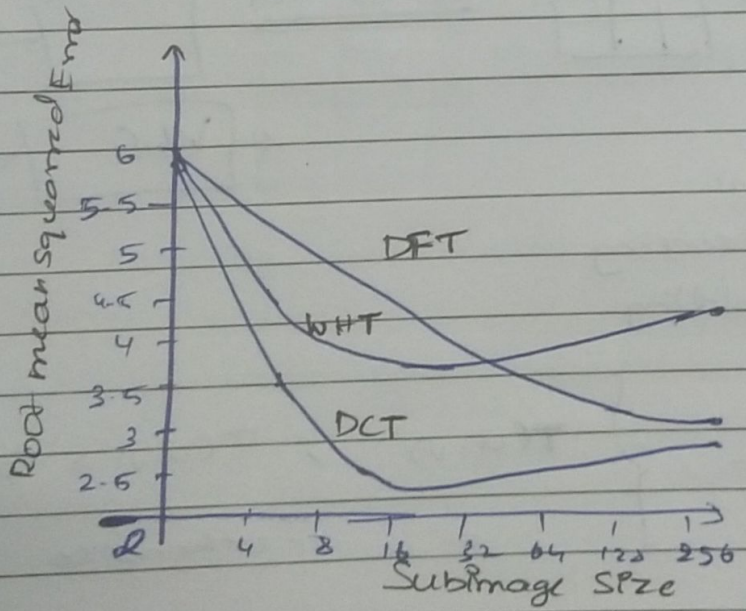
⇒ If we truncate & Quantize the function we will observe small distortions. (i.e Overshoots etc)

Ex

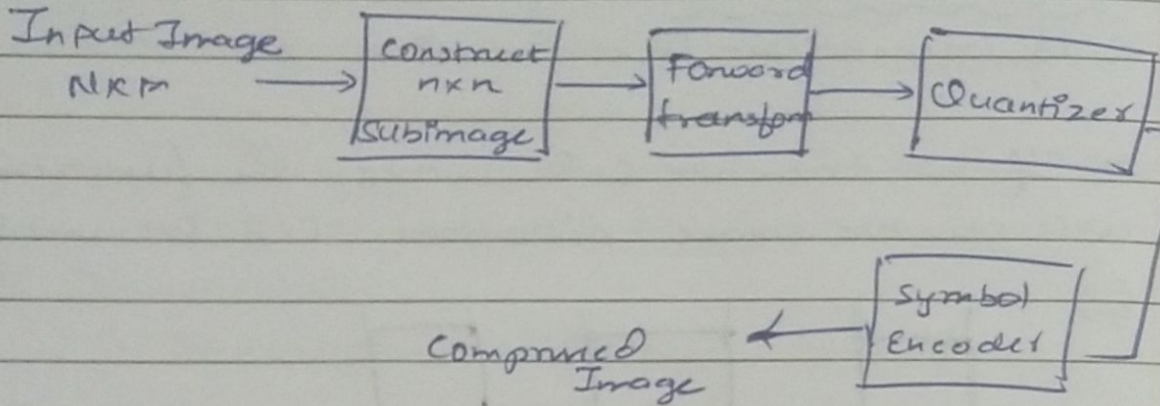


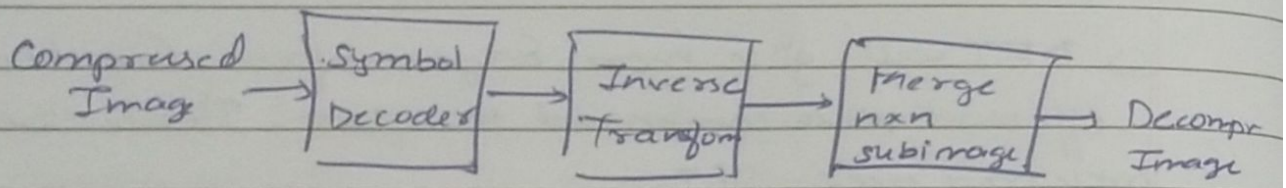
⇒ DFT will give more Edge artifact (i.e blocky effect) when compared to DCT. Hence DCT is preferred over DFT.

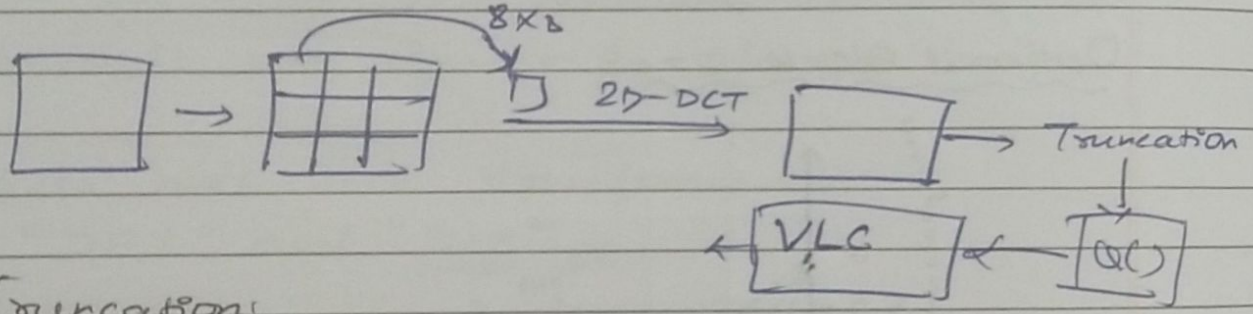
Optimal Block size:



Date / /







Truncation:

(i) Zonal coding

(ii) Thresholding

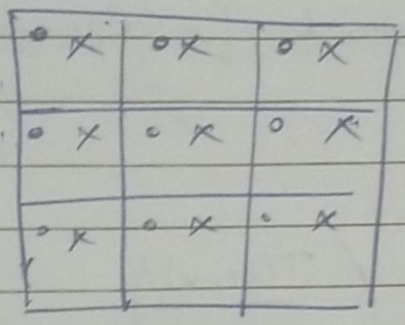
$$\hat{T}(u,v) = \begin{cases} T(u,v) & ; T(u,v) \geq T_0 \\ 0 & ; \text{otherwise} \end{cases}$$

Zonal coding

- works with Zonal mask
- based on variance. (variance is high → kept & if variance ↓ → truncated).

$$\sigma^2(u,v) > c$$

Date / /



(Each sub block has different masks)

$f(x,y) \longrightarrow F(u,v)$.

$\hat{F}(u,v) \longrightarrow F(u,v) \cdot M(u,v)$

Thresholding mask

Thresholding

(i) Global Thresholding:

$$\hat{T}(u,v) = \begin{cases} T(u,v), & |T(u,v)| > Th \\ 0, & \text{otherwise} \end{cases}$$

(ii) N-largest Coding

- In each sub-block, keep the "N" largest coefficient value.
- No of coefficients is fixed unlike Global Thresholding.

(iii)

$T(u,v)$ $\qquad$ $Z(u,v)$

Threshold Matrix

$$\begin{bmatrix} T(0,0) & \cdots & T(0,7) \\ \vdots & & \\ T(7,0) & \cdots & T(7,7) \end{bmatrix} \qquad \begin{bmatrix} Z(0,0) & \cdots & Z(0,7) \\ \vdots & & \\ Z(7,0) & \cdots & Z(7,7) \end{bmatrix}$$

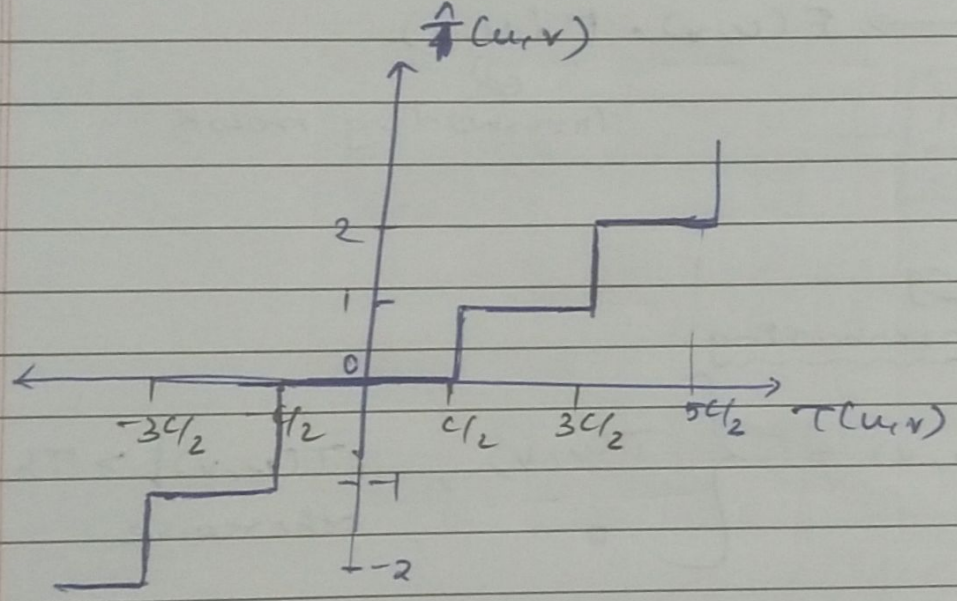
Date / /

↳ Different threshold for different location of (u,v).

$$\hat{T}(u,v) = \text{round}\left(\frac{T(u,v)}{Z(u,v)}\right)$$

↓ will Truncate and Quantization

$Z(u,v) = C$ for each (u,v) & it's different



↳ Mid tread Quantizer

Then, the approximate image in spatial domain is given by:

$$\text{IDCT}\left(\hat{T}(u,v) \cdot Z(u,v)\right)$$

Date / /

JPEG (Joint Photographic Experts Group):-

↳ Baseline coding.
  ↳ Lossy.

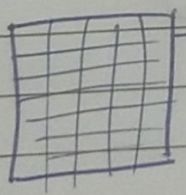
↳ JPEG-LS
  ↳ Lossless coding.

i) Divide the given image into 8x8 blocks which are non-overlapping. And consider a single sub-block on further steps

(ii) Apply level shifting
$F'(x,y) \Rightarrow F(x,y) - 128$

(iii) Apply 2D-DCT
$\Rightarrow F(x,y) \xrightarrow{\text{2D-DCT}} F'(u,v)$

(iv) Quantization.

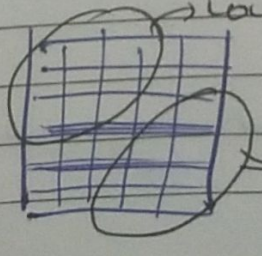


→ Quantization matrix

8x8 $Q(u,v)$ → Depends on Q [1 to 100].

$$\hat{T}(u,v) = \text{round}\left(\frac{T(u,v)}{Q(u,v)}\right)$$

↳ Each value of 'Q' has a unique Quantization Matrix



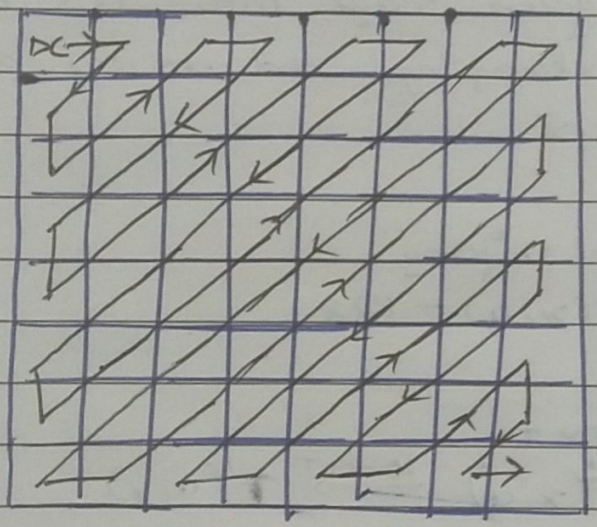
Date / /

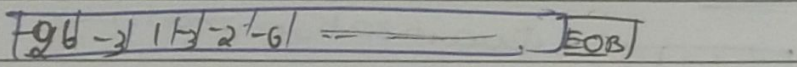
↳ The magnitude of the DCT coefficients will be larger in the low frequency (i.e top-left)

(v). Zig-Zag Scanning:

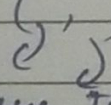
↳ After Quantization we have a 2D matrix.

↳ convert this 2D matrix to a feature vector





(vi) Variable Length Coding:



value Times it is occuring Continuously.

**

DC value is encoded seperately

Date / /

=> [(-3,4), (1,1), (-3,1) ____ ____]

(vii). Encoding

(Run, CAT)	Huffmann coded Codeword
0,0 (EOB)	1010
0,1	0
0,2	1
⋮	⋮

↳ Encode the above variable length coding ~~encod~~ those with the above "standard table"

for DC values

↳ It's efficiently encoded if we use DPCM technique.