After Mid Sem

Contd.--

PAGE NO.:
DATE: / /

Frequency Domain : Fourier Transform

Earlier we talked about image restoration in spatial domain. Now, we will talk about freq. domain

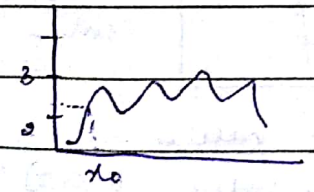
low frequency : not much change in intensity, smoothening
high " : sudden change " " edge.

Spatial domain $\xrightarrow[\text{Transform}]{\text{Fourier}}$ Freq. domain.

Fourier series : for periodic signal
" Transform : " aperiodic "

Transform : Any signal can be represented by sum of sine & cos

Continuous $\longrightarrow$ Discrete [sampling + Quantization]



$\longrightarrow$ Quantize to closer one

M samples.

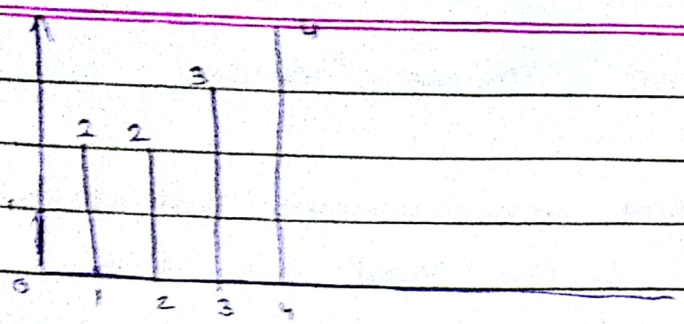
1-D :

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x)\, e^{-j 2\pi u x/M}$$

u → freq.
x → coordinate in spatial domain

$$f(x) = \sum_{u=0}^{M-1} F(u)\, e^{j 2\pi u/M^2}$$

(go back to spatial → Inverse Fourier)



find fourier

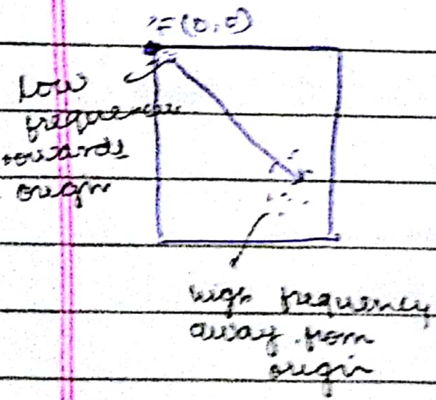
F(0) → gives avg. value of the signal

$|F(u)|^2$ → Power Spectrum of Signal ($f(x)$)

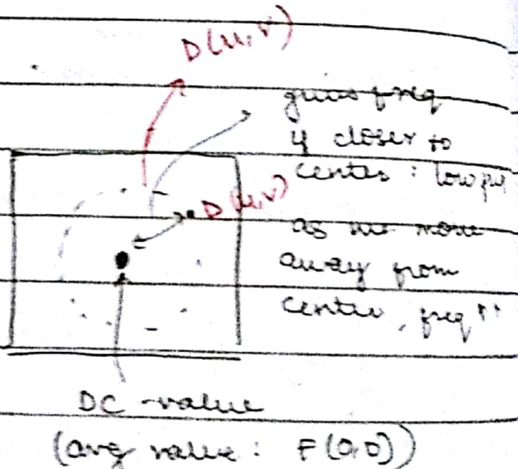
2-D :

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

F(0, 0) → gives avg. intensity .



for better visualization



Basic Filtering :

1. Multiply i/p image by $(-1)^{x+y}$: move DC value to center

suppose img is corrupted by high freq. noise.
applied fourier : apply some filter so all high freq. are removed & inverse.

eg. → Bolter : corrupted by sine wave (noise)

H/W : Convert into fourier transform
Shift image towards center. Check high freq components & remove them. DO Inverse fourier transform

PAGE NO.:
DATE: / /

Why only getting 1 point in freq. domain?
If it has 1 sine wave : it'll have 1 freq.
so, just have a point

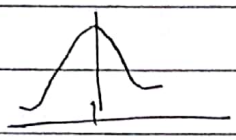
In Fourier Tranform : each freq. component has conjugate pair (complex) also.

can have filter in circle form. st all points on the circumference =0. (by masking) [Rest values are 1]

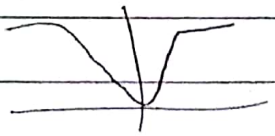
Place Mask & multiply. Then, freq. at circumference of circle will be 0 ⇒ Removed them.
Then, we can do inverse fourier to get original image

→ Salt & Pepper Noise : Impulse noise

→ low pass : all ~~low~~ high freq. → rejected
⇓
only smooth areas are observed.
no edges " "

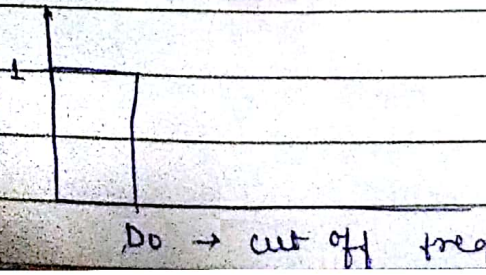


→ high pass : all edges are clearly defined.



Spatial domain:
low pass : Avg. filter high pass : Laplacian

Ideal low Pass filter



Gaussian & Butterworth : almost same
↓ slightly narrow ↓ slightly broader

PAGE NO.:
DATE: / /

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

$D(u,v)$ → distance from center

→ **Butterworth :**

Graph : $n = 4$: steep cut-off
$n = 1$: smooth → gradual cut-off
} → tells about smoothening of cutoff

n → Order of filter

→ **Ideal High Pass :**

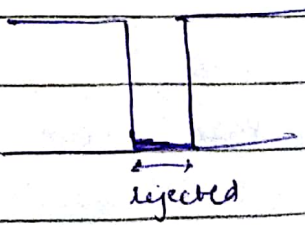
$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

$D_0 = 15$ → everything outside 15 is allowed
$D_0 = 30$ → only ~~~~ ~~~~ → very high freq. are allowed

Reject filter = 1 − Pass filter

Periodic Noise Red^n → have to use selective filter to isolate point
↓
have burst kind of freq.

→ Band Reject filter :



→ Can [illegible] Fourier Transform & then shift to center using [illegible] filter

PAGE NO.:
DATE: / /

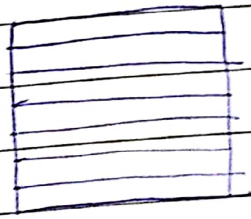
Notch filter : very similar to bandreject (remove freq within specific nbd.)

→ better to have its conjugate reflection also. (to remove conjugate ~~filter~~ pair also)

$$H_{BP} = 1 - H_{BR}$$

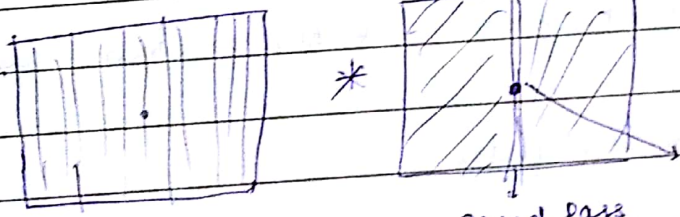
(Bandpass) (Band Reject)

Eg Noise :



Freq. : change of Intensity
in horizontal dir" : No Δ freq.
in vertical " : ∃ Δ freq.

Fourier Transform



Band Pass filter

⇒ get : (get only vertical freq image only the noise

Now, they have removed, because, don't want to change low freq

1 - this noise ≐ good image

or → apply Band reject image.

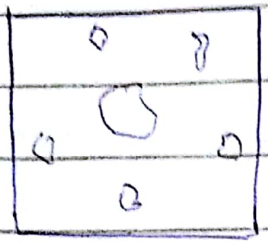
Mens :(6)→ Not very sharp freq. comp. ⇒ diff. to decide what to remove & what not to

Use

Optimum Notch filtering

Optimum NOTCH Filtering

$\hat{f}(x,y)$: estimate of original image $f(x,y)$



$G(u,v)$

$g(x,y) - \eta(x,y) \equiv \hat{f}(x,y)$

$g(x,y)$ (degraded img), $\eta(x,y)$ noise

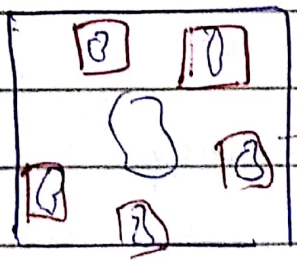
don't know how much is image & how much is original image

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

Task → Minimize diff. b/w $f(x,y)$ & $\hat{f}(x,y)$

Here, we do weighted filtering & try to min. weight

H_{NP} : Notch Pass filter



→ allowing these freq. (high freq) in Notch Pass filter (noise in freq domain)

Frequency Transform of Noise $\hat{N}(u,v) = H_{NP}(u,v)\, G(u,v)$

Take inverse to get noise.

DATE: / /

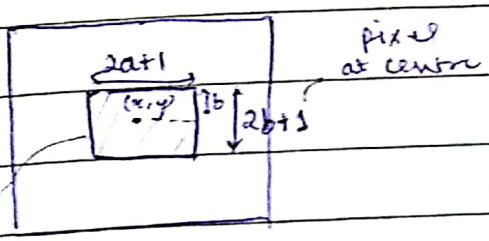
→ Says, do a weighted noise subtraction, don't remove noise directly as it may contain original image

$$\hat{f}(x,y) = g(x,y) - w(x,y)\,\eta(x,y)$$

select $w(x,y)$ st variance of $\hat{f}(x,y)$ is minimized

↓

image is app. is the same.



$s = a$ ~~&~~ $t = b$

local variance

$$\sigma^2(x,y) = \frac{1}{(2a+1)(2b+1)} \sum_{s=a}^{a} \sum_{t=b}^{b} \left[\hat{f}(x+s, y+t) - \hat{f}(x,y) \right]^2$$

Suppose, in a ~~noise~~ nbd weight is almost const.

$$\Rightarrow \quad w(x,y) = w(x,y) \approx w(x+s, y+t)$$

Solve for (w)

→ Now, we can obtain $\hat{f}(x,y)$

* Till now, we didn't consider Degradation func^n.

Estimating the Degradation function

① Mathematical modeling

Use mathematical func^{n}s to get degradation func^n.

If $k \downarrow\downarrow \Rightarrow$ turbulence is $\uparrow$

Inverse filtering

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

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

If $H(u,v)$ is high $\rightarrow$ small noise is added to $F(u,v)$
won't able to get original image

—— low $\rightarrow$ estimate will be wrong

Ex. → Guessing cut off frequencies seeing responses

Min Mean Square Error Filtering → Wiener

$$\min \quad e^2 = E\left\{(f - \hat{f})^2\right\}$$

S_η : Power spectrum of noise

S_f : Power spectrum of ~~extra~~ undegraded image

DATE: / /

→ SNR Ratio should be ↑

① math. modelling of optimum NP & weiner filter

② Try to remove img.

16/10/18

Image Compression

Reason to Compress :

1) Transmission

2) Storage

→ Video : 720 x 480 → have 30 frames per sec.

2 hr movie

No. of bits in each pixel : 24 (RGB)

Size for 2 hr movie = 30 x 2 x 60 x 60 x 720 x 480 x 24

≃ 200 GB (DVD Capacity : max 8.5GB)

diff. to transmit data

→ irrelevant / repeated

⇒ Will compress it (remove the redundant data)

→ Will use some coding technique

Relative Data Redundancy

Compression Ratio

$C = \frac{b}{b'}$ → Original Representation

b' → New Representation

$R = 1 - \frac{1}{C}$ → relative data redundancy.

Teacher's Signature.....

if $b = 8$ & $b' = 2 \Rightarrow C = 4$

$R = 0.75 \rightarrow 75\%$ of data is redundant (in original image)

→ Implement Compression :

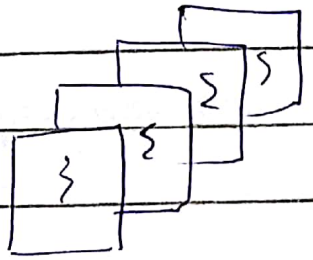
(1) coding redundancy

eg. gray image just using 3 or 4 gray levels. now, we don't need 8 bits but we're representing it in 8 bits

(2) Spatial & temporal redundancy :

Spatial (nbd) eg. large areas of gray / some color { current pixel is same as next ---- } can I represent it in other way

Temporal (time) : In videos represented by : $f(x, y, t)$



→ All images are almost same. Can I represent it in some other way ?

(3) Irrelevant information :

Take broader view of image (removing detailed part) Can I remove this info. to compress image ?

PAGE NO.
DATE : / /

2a. Fig 2.1 (a) Only 4 or 5 gray levels here but we're using 8 bits → Coding redundancy

(b) Have lots of same color → Spatial

(c) Can't see gray color, not interested in details

Coding Redundancy

→ intensity with highest prob, give it smallest code either give same length to each, or do as above

r_k		code 1	code 2
$r_{87} = 87$	(0.25)	01	2
$r_{128} = 128$	(0.47)	1	1
$r_{186} = 186$	(0.25)	000	3
$r_{255} = 255$	(0.03)	001	3

↓

code is assigned in such a way that when you get a seq of codes, there's no discrepancy in decoding (only 1 possible way of decoding)

No of bits used

$$L_{avg} = (0.25)(2) + (0.47)(1) + (0.25)(3) + (0.03)(3)$$

$$= 1.81 \text{ bits}$$

$$C = \frac{b}{b'} = \frac{8}{1.81} \simeq 4.42$$

$$R = 1 - \frac{1}{C} = 0.774$$

If I assign 2-bits to each, then

$$C = \frac{b}{b'} = \frac{8}{2} = 4$$

Spatial & Temporal Redundancy

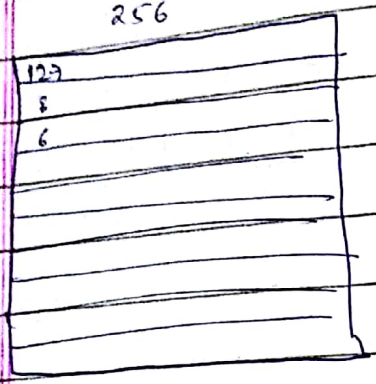
Spatial:-

→ In histogram : all intensities are equally probable

→ Eliminate by using a seq. of run-length pairs

↓

specifies start of new intensity & no. of consecutive pixels that have that intensity



Right Now : Using 256 x 256 x 8 bits

Now, we can represent as:

(127, 256), (8, 256), (2, 256)

↑ intensity ↑ frequency of 127 (run length)

Now : Using 256 x 8 x 2 bits

$$C = \frac{256 \times 256 \times 8}{256 \times 8 \times 2} = \frac{128}{}$$

* If 100 intensity comes 2 times, we can add them together

Irrelevant Information

Slide

→ did some Histogram equalisation

→ If you need that info., code it. (Otherwise not)

$$C_2 = \frac{256 \times 256 \times 8}{8}$$

8 → whole img is 1 color → Just need 8 bits

DATE: / /

$I(E) = \log \frac{1}{P(E)} = -\log_m P(E)$

$m = 2 \Rightarrow$ info. is bit

$P(E) = 1/2 \Rightarrow$ 2 options (Y/N) → need 1 bit

$P(E) = 1 \Rightarrow$ it will definitely occur ⇒ no info.

→ Entropy tells how many pixels should be atleast there so that property of image is not lost.

→ If $e_{rms} = 0 \Rightarrow$ lossless compression

$e_{rms} \neq 0$ → lossy "

Original image : $f(x,y)$

compressed & transmitted : decoded at dest^n : $\hat{f}(x,y)$

↑ signal

Some basic Compression Models

① Huffman Coding

Symbol → may be intensity values

→ Combine lowest prob.
→ Put in descending order.
→ Ultimately, I just 've 2 prob.

} Src Red^n

↳ Most frequently occuring gray value has shortest code.

Higher prob. → Assign 0

Lower prob. → " 1

→ Trace back route

Teacher's Signature.....

0.4 1

0.3 00

0.3 01

0.6 0

0.4 1

* whenever I write a sequence, I will always decode in same unique way

a_2 a_6 a_1

1 00 11

* code by which you encoded should be available at decoding side also

decoding end: a_2 a_6 a_1

3 1 2 2 6

Eg. 0 1 0 1 0 0 11 11 0 0 (a_3 a_1)

a_3 a_1 a_2 a_2 a_6

② LZW (Dictionary Coding)

(We're providing code for patterns)

→ don't require prob.

Eg.				
(can use run-length)	39	39	126	126
	39	39	126	126
	39	39	126	126
	39	39	126	126

→ (39, 2) (126, 4) (126, 2) ....

dictionary is not needed here at decoding side as it 'll be automatically created

PAGE NO.:
DATE: / /

We know 256 gray levels are there

0 to 255 → give entries 0 - 255) 9-bits for dictionary
256 to 511 → empty right now.

→ Algo will try to find next intensity (to check spatial redundancy)

39 → next pixel value : also 39
stored pattern 39-39 & gave value 256
[There is no entry for 39-39 earlier, only for 39]
↓
39

Once it has recognized a sequence 39-39, it is checking for next pixel also.

Currently Recognized sequence	Pixel being recognized	Encoded o/p	Dictionary Location	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			

where pattern is replaced by 9 bits (earlier 16 bits)

Teacher's Signature

17/10/18

DATE: / /

→ Don't need to send dictionary at dest^n side

→ Don't need probability knowledge

③. Run Length Coding

Ex: from slide (W.s & Bs)

Run length coding: 12W 1B 12W 3B 24W 1B 14W

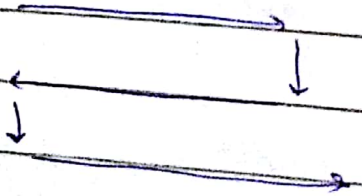
Can write also as: (count, symbol)

→ Can keep any code for symbols which should be uniquely decodable on other side

↳ More use when few brightness values: (less no. of intensity levels)

Reason: If you've just 2 intensity level → run length will ↑↑ & so, you can compress more.

For run length, we usually go this way



why like this?

Very high prob. that nbd would have same values, so, more like this.

PAGE NO.:
DATE: / /

④ symbol based Coding

Assign code for each symbol

have triplet (x, y, t)
- x, y → starting point of symbol
- t → code for symbol

(, 0) b
(, 1) a
(, 2) n
(, 1) a
(, 2) n
(, 1) a

⑤ Bit plane coding

Code the bit planes separately using RLE

39	39	126	126
00100111	00100111	01111110	01111110

RLE

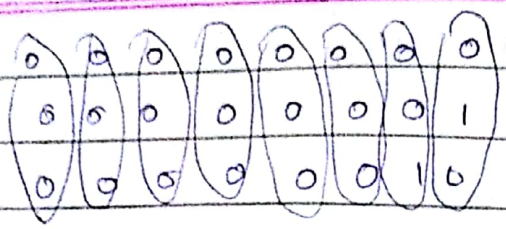
Bit Plane 0 → 1 1 0 0 (Bit 0 of each pixel) → (2,1)(2,0)
Bit 1 → 1 1 1 ....
.
.
.
8 →

After separating bit plan, do RLE for each bit plan

Teacher's Signature.....

PAGE NO.:
DATE: / /

Why using this ?

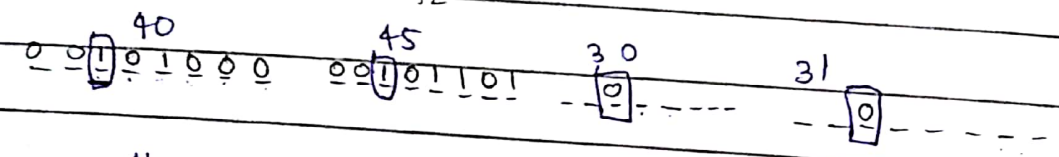


↳ many bit planes will have (long) cont. run lengths (of 0 & 1) which 'll make it easier to compress

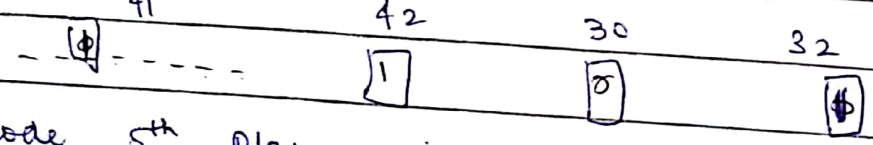
→ Downfall :

neighboring intensity but there's change in bit (last bit)

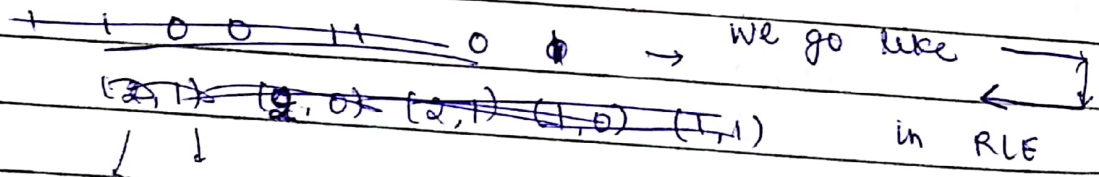
Ex. 40 45 30 31



41 42 30 32



Code 5th plane using RLE

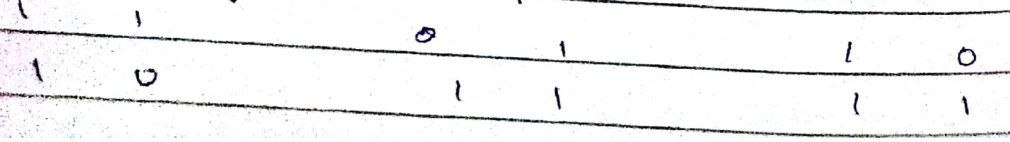


So, code will be :

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

Ex. Slide : 5 6
7 3

Corresponding 3 bit planes



5, 6, 7 → nbing intensity

2nd bit plane → bits are same } → What we like ? All should be same.

1st & 0th → " " diff.

→ A small change in intensities of nbd pixels disrupt in run length.

Ex 127 & 128 ⇒ Bits are totally changed although we have small Δ in intensity

↙ Soln (Instead of binary code, use gray code)

GRAY CODE

5 = 0 0 0 0 0 1 0 1

In gray code : MSB remains as it is

Other bits : XOR b/w that bit & just prev. bit.

5 = 0 0 0 0 0 1 0 1

Gray code = 0 0 0 0 0 1 1 1

6 = 0 0 0 0 0 1 1 0

Gray code = 0 0 0 0 0 1 0 1

7 = 0 0 0 0 0 1 1 1

G_7 = 0 0 0 0 0 1 0 0

→ Adv. : If there are adj gray levels, change would only be in 1 bit

Eg. 7 & 8

0111 1000 → all bits changed

Teacher's Signature.....

DATE: / /

127 : 0 1 1 1 1 1 1 1
G_7 : 0 1 0 0 0 0 0 0

128 : 1 0 0 0 0 0 0 0
G_8 : 1 0 0 0 0 0 0 0

→ More or less, i^{th} bit plane code will be smooth (same everywhere)

Slide : Binary → Intensity level changing very freq.
Gray → " " almost const.

MSB → get general view
LSB → " detailed " } → Bit Plane

Decoding Gray code

XOR with prev. bit we got in decoding part (below line)

got → 0 0 0 0 0 1 0 0 (G_7)

decode : (a) → 0 0 0 0 0 1 1 1 → 7

Block Tranform Coding

→ Now, we take blocks of image
→ Take square blocks (if not possible → do padding)

Quantizer : Ceil / Floor / ...

→ Decompressed Image will not be same as Input Image

Reason : Quantization (Quantized 5.2 to 5, lost some extra info.) → loosing some part of image here

Teacher's Signature

PAGE NO.:
DATE:

→ If I want, I can convert it into some other space also & then construct block & transform

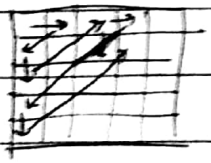
↓

maybe using fourier transform

- Entropy code : code after calculating entropy

DCT : discrete cosine transform

Zig - Zag technique : Convert 2-D array → 1-D



Slide Ex → Quantizing using Normalization Matrix.
(dividing by values in that Matrix)

After quantise ⟹ [79 0] → freq components now (DC value - 79)

↳ Most we become 0.
↓
can be neglected or run length of 0

Slide $\alpha(u/v)$: either u or v.

SubImage Size Selection

With size of 8, error is min. usually for all case.

So, 8 X 8 is used usually.

JPEG Compression

3 coding system

→ lossy baseline : used mostly.

→ Level shift :

If $\neq 256$ intensity levels : from each pixel, subtract 128

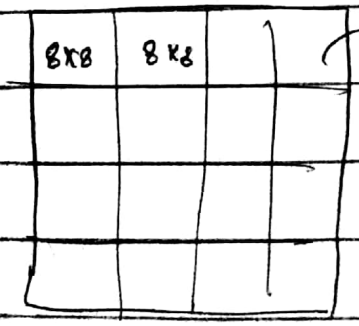
(2^K) (2^{K-1})

→ DCT is being used in JPEG

→ Quantize using std. Normaliz^n Matrix

Note : after DCT : high freq. have smaller values (bottom right)

low — large values (In top left)



→ computed for each block.

↳ DC value

from each DC value, subtract the DC value in ~~last~~ left block just left to it. Then, convert to binary & do huffman coding.

JPEG 2000 → lossless

DATE : / /

Project → Mosaicing Assignment (20%.)

Documentation → Literature review ✗

Coding

~~Docum~~ Have to do :

① Report

② Code

③ Presentation / Viva

Good Programming Practices

① Comments

② Indentation

③ Modularity

④ Structured

⑤ significant variable name

⑥ Exception Handling

project → 10% Coding

→ 10%. Documentation / Presentation / Viva

Documentation : → Have heading called Design decision & write atleast 1 para on your thoughts

- May use other algo (other than what we will do in class today)
  - ↳ Explain algo + Design decisions (Where you applied Reverse / Forward Mapping)
  - ↳ Paste Result after each & every ~~for~~ step.

- Inbuilt fns can only be used for mathematical opns (eg. → Inverse)

Projective

① Need to choose 4 pts (correspondance)

Have to choose pt on blackboard only, because only it is planar.

i) Load images 0.jpg & 1.jpg

```
FILE *srcFile;          → argv[3], "r"
if (srcFile fopen(  ) == NULL) {
    cout << "Error";
    exit 0;
}
```

Read src file to get all coordinates

```
for (i=0; i<8; i++)
```

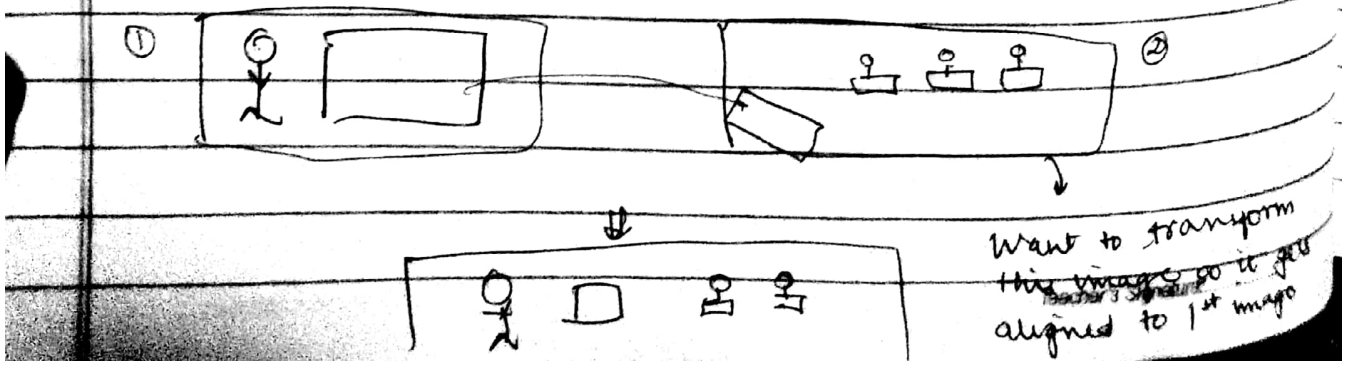
2.) Find homography matrix

do using SVD

Write now : find homography,

C++ : Mat H12 = findHomography (SrcCoord, DstCoord);

can check : H12 at <float> (2,2)



= H { . }
(pSelect (w

PAGE NO.:
DATE: / /

~~Now~~ ② is ~~transformed to get aligned~~

We want to get ② aligned with ①

Take inverse of homography matrix (H12) you got

H21 = H12 inv ( ) ;

Have to apply this matrix on corner points of 2nd image & ~~store~~ store these points in a vector (C++) or a list (python) (Keep corner points of 1st image also in a vector.

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$ H x

[ ] [ ]

(0,0) (0, 1664)

(1, 2496.) (9

$$px = \frac{x \times h_1 + y \times h_2 + h_3}{x \times h_7 + y \times h_8 + h_9}$$

$$py = \frac{x \times h_4 + y \times h_5 + h_6}{x \times h_7 + y \times h_8 + h_9}$$

(1,1) (1664,1)

(1, 2496) (166

(H) [

→ Why doing it ?

To find range (min & max) of 2nd image so we can get iterate whole image Then, find

max x min x → Out of corner points → won't get value in negative

max Y min y

Most of cases, dimension are ↑. So, not necessary to store corner points

We get :

2496 x 1664

width height

Scanned by CamScanner

0.007 — 0.0010

0.0035 — 0.0023

this much → has overlapped with original image

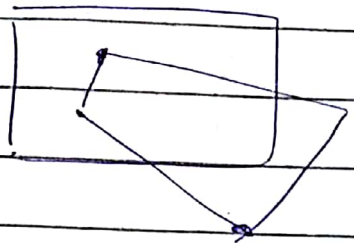
2666 maxWidth

2197 Height minWidth

0 minHeight

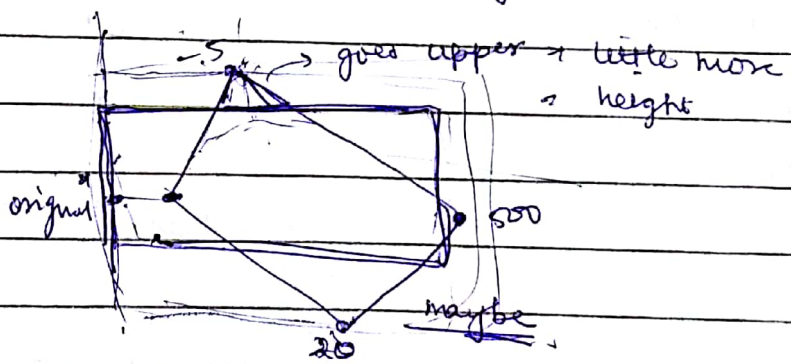
0 Height minWidth

2496
2496
4992



if minX = 0 ⇓ jyaada left nhi jaa rahi

minY = -130 ⇓ upar jaa rahi h image



y = 5 ≤

height New = max Height + abs (Y offset)

if (y < 0)

yoffset = abs (minY) {adding upar wala part}

widthNew = max Width + abs (X offset)

↓ which we calculated above

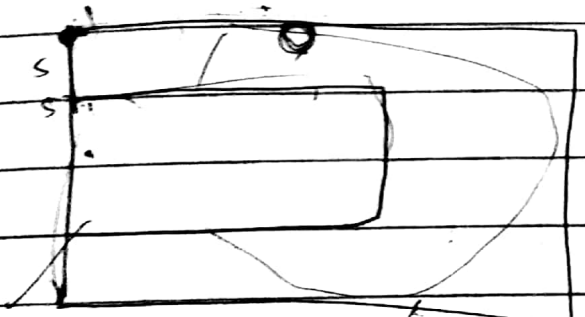
DATE: / /

Eg minX = 0 minY = -130

→ OffsetX = 0 OffsetY = 130

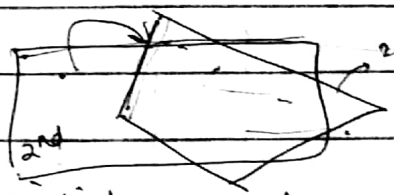
→ Want to paste 2nd image as it is on 1st image

Iterate starting from y + yoffset



I12

Paste src image on empty canvas



For pasting 2nd image, can check where no color is assigned on canvas & assign color to it. If

Create canvas acc to final image coordinates you got earlier

Apply :

H12 x RingImg Coord

Iterate it. If you get corresponding points in 2nd image (final). If it exists, → assign on canvas

otherwise (if get out of bounds)

↓

discard it

25/10/18

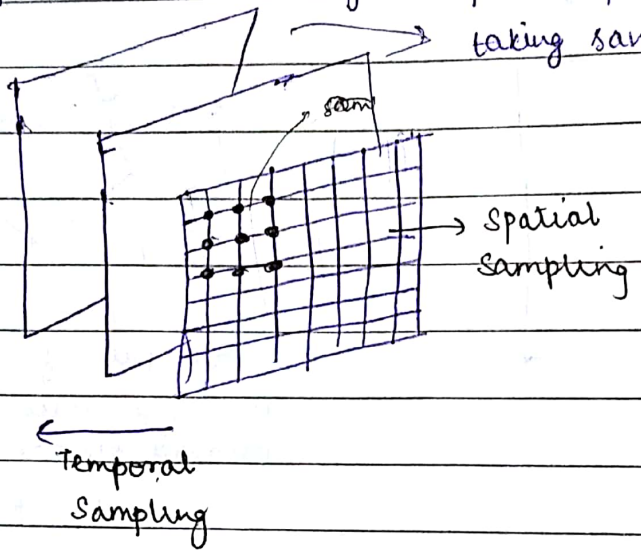
PAGE NO.:
DATE: / /

Digital Video Processing

Represent" of visual scene at a particular time : instance of sampling ~~Digital Image~~

Sampling gives → 1 frame / Image.

Digital Video : Taking sample repeatedly at diff instances.



Adv. of digital videos : (vs Analog Video)

↳ ~~Can convert~~ Small Compatibility (In analog, need a h/w to support video, digital can convert easily) → ~~compatibility~~ Compression & transmission

↳ Noise can affect Analog Video

digital : less noise sensitive, have

Other adv. : Same as digital images (storage, ---)

Spatial

Why to do Sampling ?

↳ Just to avoid need of storage.

(Don't want that much of detail in some cases)

↳ More detailed : High sampling

Temporal sampling : After what durⁿ of time I'm generating a frame.

Generally : $\frac{1}{25th}$ or $\frac{1}{30th}$ sec

But we can have $\frac{1}{10^{th}}$ sec: not smooth but our work will be done

2 things:

→ ① Bandwidth : kitni badi image pass krte h

→ ② Data rate : kithe frames pass krte h at a point of time

↓

little more costlier

① & ② ⇒ expensive

2 ways to do Temporal Sampling :

1.) Progressive Sampling (p)

↓

complete frames are sampled

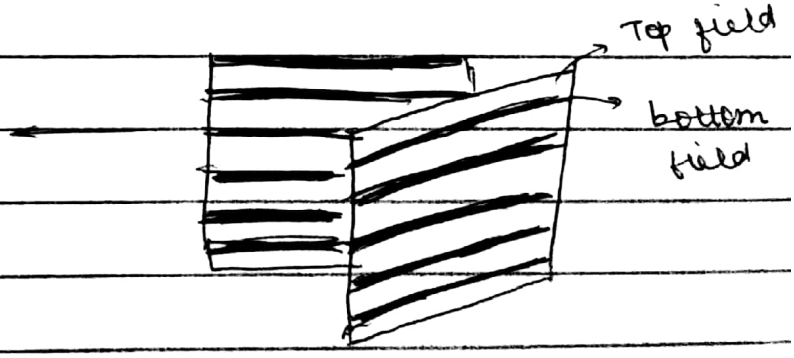
Ex: 480p, 720p, 1080p (movies)

2.) Interlaced Sampling (i)

Ex: 480i, 720i, 1080i

At a time, I either sample even lines or odd lines

1st sample at even, then odd (or vice-versa)



→ If I sample very fast, they appear very smooth.

* When bandwidth is small, I can't transport whole image (don't have enough bandwidth) That case, I can ↑ data rate.

PAGE NO.:
DATE: / /

→ Progressive : talk about frames
→ Interlaced : " " field

Color Spaces for Digital Videos

Color Space : ~~import~~ universally std. color model (like RGB) in which colors in diff. propor^n^ make a unique color

Luminance (intensity value) / : tone of src
") Brightness : what we perceive (subjective) } can't be measured
Chroma / Color : to specify color

↳ RGB : In RGB, each R, G, B have equal proportion (8 bits each) (Resol^n^ is same for all)

In videos, we're color model :

Y Cr Cb → because : Human Visual System has more sensitivity to luminance than Color (Intensity values), so, we use this color space in which luminance is given more resol^n^ than color.

Y → luminance
Cr Cb → color

resol^n^ → no. of pixels
(can see more clearly)
(more details clearly)

→ for display : use RGB
" transmission : - YCrCb → because ek component ka resolution kam kar diya h

DATE: / /

YCrCb

$Y = k_r R + k_g G + k_b B$

Luminance → weighted average of R, G, B

k → weights

Colour difference $C_r = Y - R$

$C_g = Y - G$ ⇒ storing 4 things

$C_b = Y - B$ (Y, C_r, C_b, C_g)

In RGB : (R, G, B)

→ So, YCrCb is bad than RGB because size is ↑ than in RGB.

→ But, it's good, because $C_r + C_g + C_b = \text{const}$. ; so even if I know 2 values we can get 3rd value.

generally, we store 3 values only : Y, C_r, C_b

Conversion formula :

$Y = 0.299 R + 0.587 G + 0.144 B$

$C_b = 0.564 (B - Y)$

$C_r = 0.713 (R - Y)$

$R = Y + 1.402 C_r$

$G = Y - 0.344 C_b - 0.714 C_r$

$B = Y + 1.772 C_b$

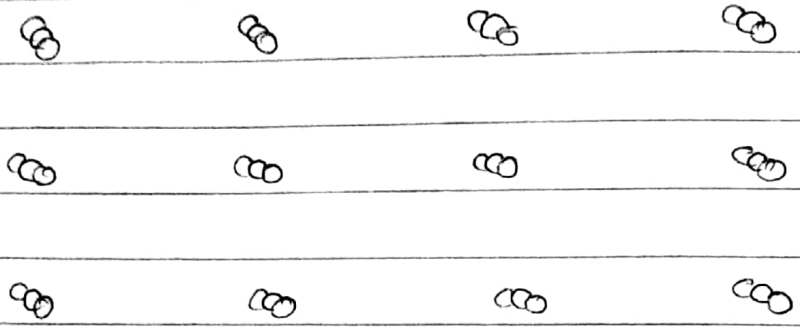
Diff sampling std :

1) 4 : 4 : 4 sampling :

Each pixel has equal proportion of all three components

(Y, C_r, C_b)

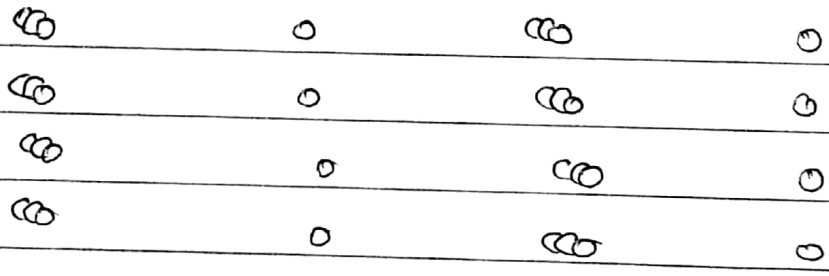
o — Y
o — C_r
o — C_b



→ In each row, propor" of each component is same

2) 4 : 2 : 2 Sampling : (YUY2)

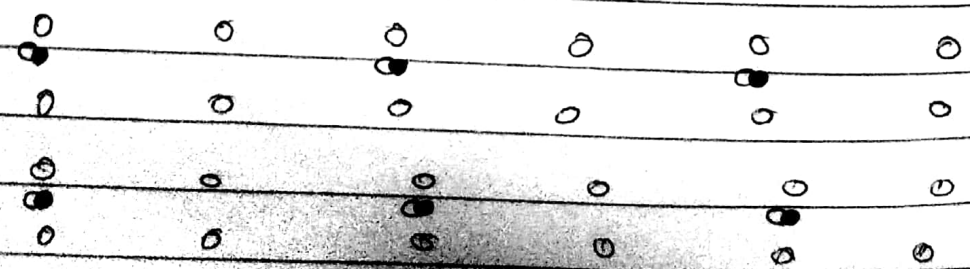
In vertical dir", chroma has equal proportion as luma, in horizontal direction, it is half



↳ Used in case of DVDs & std. Definition TV displays

3) 4 : 2 : 0 sampling ('YV12') :

C_r and C_b have half the resolution of Y in both horizontal & vertical dir".



Used in Video Conf

Teacher's Signature

Multimedia file

PAGE NO.:
DATE: / /

Ex. Image Resolution : 720 (width) × 576 (height) pixels

Y (luminance) resolution 720 × 576 samples each with 8 bits

4 : 4 : 4 $720 \times 576 \times 3 \times 8$ (each has same resoln)

$\{(720 \times 576 \times 8) \times 3\}$

↓ Luminance ↳ all have same resoln

4 : 2 : 0 $720 \times 576 \times 8 + \frac{720 \times 576}{4} \times 8 \times 2$ (360 × 288)

$= 720 \times 576 \times (8 + 4)$

$= 720 \times 576 \times 12$

4 : 2 : 2 $720 \times 576 \times 8 + \frac{720 \times 576}{2} \times 8 \times 2$ (360 × 576)

$= 720 \times 576 \times 16$

→ 4 : 4 : 4 → used in HDs

4 : 2 : 2 → Chroma sub-sampling methods

4 : 2 : 0 → taking least amount of bandwidth

Best method : depends on our appln. (out of 3)

Video Formats

1) Intermediate
2) Standard Definition
3) High Definition

DATE: / /

① → Convert 1 format to another

Basic one :→ Common Intermediate Format (CIF)

~~It has Res of 352 × 288~~ (Luminance)

Luminance Res^n : Bits per frame

352 × 288 (4:2:0 ; 8 bits per pixel)

Bits in each frame = 352 × 288 × ~~12~~ ~~352~~

= 1216512 bits

= 152064 B

= 148.5 MB

Other

QCIF (Quarter CIF : half res^n in both → & ↓ dir^n) $\left[\frac{352}{2}\right.$

4CIF (: twice " " " " ") $\left[2 \times 352\right.$

② Standard Definition

720 × 576

③ High Definition

720p Res^n : 1280 × 720 25 frames per second

1080i 1920 × 80

1080p → 25 frames per sec

→ 50 fields per sec.

Res^n

Total ~~res of bits~~ with be same in both i&p

bits ⇒ talk about even-odd frame pairs

PAGE NO.:
DATE: / /

Video Compression

→ In compression : Try to remove redundancy without affecting quality

→ Quality : Fidelity

Objective Criteria
measure on basis of parameters. Ex:
→ Signal to Noise Ratio
→ Peak " "
→ Mean Square error

Subjective Criteria
Depends on your perspective, whether you like it or not
Can't measure

Q. Why to have subjective when we already have Objective ?
MSE : match pixel by pixel Even if small error : square ↓ Error ↑↑

If I shift my image : MSE ↑↑ (intensity value changed completely)
subjective : won't make any diff.

⇒ Depends on app" (compression)
For storage : Higher "

Image Compression : Transform Coding → JPEG
Redundancy is removed

Q1 How you choose Transf" func" ? (Why DCT, not others ?) KLT, FFT

Q2 Why divide into n×n sub-image ? How to decide value of n ?

can restore complete image

Quantizer :→ causes lossy in compression

PAGE NO.:
DATE: / /

MJPEG

To compress video:
We can compress each frame of video. → Motion JPEG
(using JPEG)

→ We're only talking about spatial redundancy here (like in images)
↓
intra-frame

Temporal Redundancy :
→ In videos : successive frames are also similar

Intuitive Methods :

- sub-sampling → drop frames which are similar
  ↓ lossy
  ↓ Problem
  Fluid motion won't come, it'll become jerky. won't look good as a video

For visual : so ~~Bad~~ we'll add same frames

" transmission/storage : drop similar frames
↳ can store 1 frame & write its freq. (3, 4, ..) → will take less storage

- Differencing → (To find changes in 1 frame)
  ↓
  loseless → white area : diff was more (more motion)
  (ex. in slide)

- Block Diff. →
- Motion Compensation : most widely used.

Block Diff. : same ex : Background me no change
Foreground me only small Δ
↓
have to encode Δ blocks.
(pixels will be same)

↳ good for gradual motion
(not abrupt)

Teacher's Signature

Ques. why motion compensat" & estim" is better for video compression than differencing?

Ans when I take diff b/w 2 frames I may get white region (large difference in intensities) while in error frame: Nearly 0 (most of frames)
And we'll motion vector for 1 block → giving more compression

Top 3 → good for gradual motion

Motion Compensation

→ Estimate where motion is occuring. (+5, +15)

→ Compensate this by subtracting motion from original image

Error : (x - 5, y - 15) : small diff. will come (almost 0)

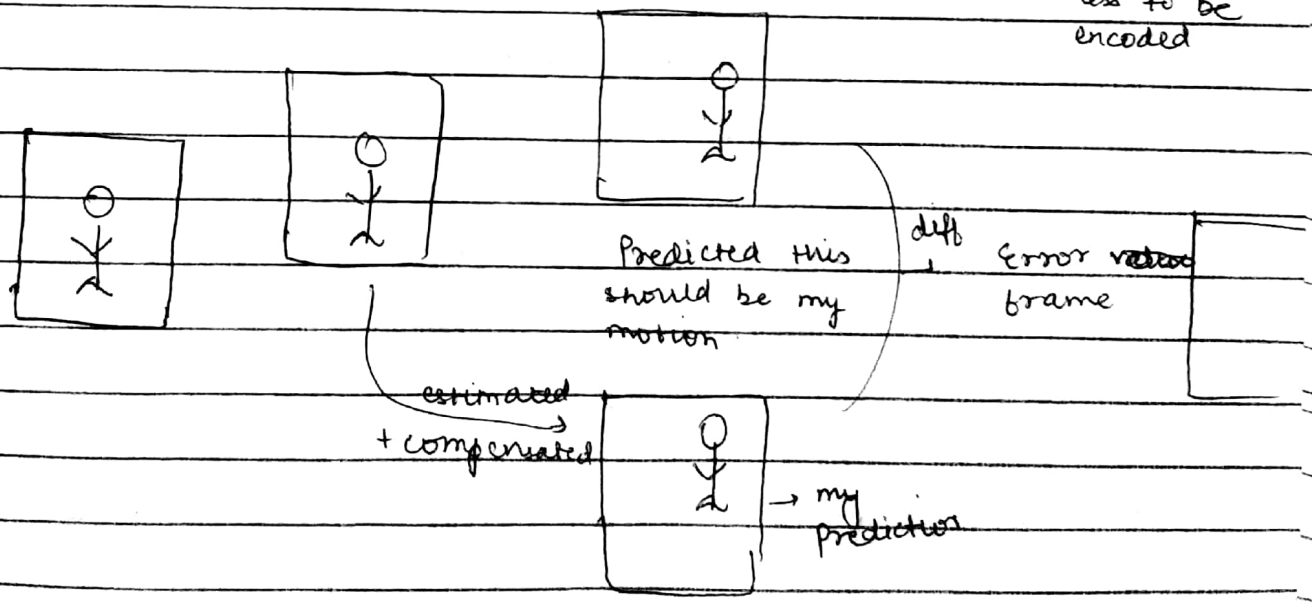
{Translation}

Can predict yahe Ritna motion hoga (Estimate)

find diff. from original image : Error ≈ 0 (Compensate)

Compression? → Coeff value ↓. { Reverse Translation }

→ less to be encoded



In case of video: for 1st frame: we don't consider Temporal Redundancy, only takes care of spatial

2nd frame onwards: both spatial & temporal

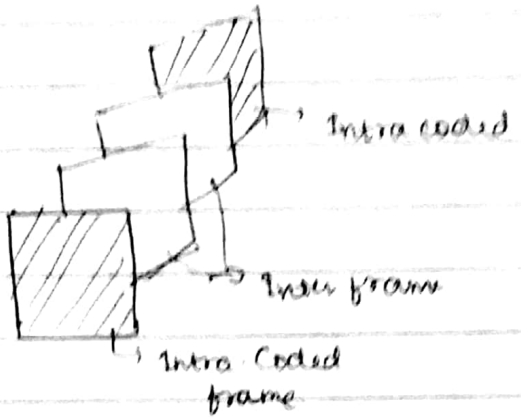
Periodically introduce to prevent accumulation of prediction error over frames.

If 1 we accelerate we are estimating linearly so, real error will get ↑ by (accumulates)

what we do? After some frames, we again send next frame as I (don't consider temporal, just spatial)

↓

2 - frame now predicting using original frame



Motion Estimation

How to estimate motion?

1) segment video into moving objects → Difficult
   describe (model) object motion
2) Segmentation - free

↳ compute displacement b/w current frame & stored (reference) frame

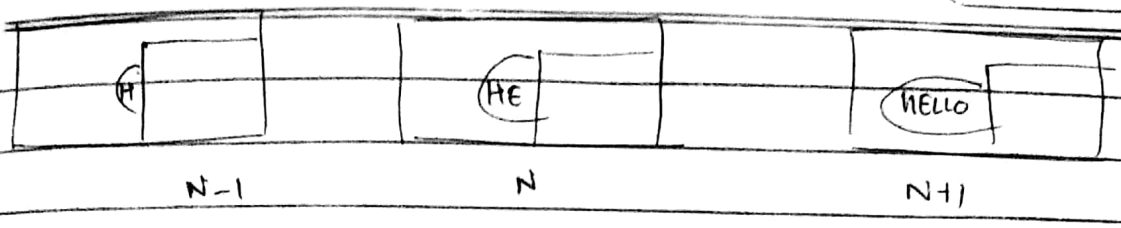
↓

motion vector (x as well y dir^n)

→ Previous frame | Backward Motion Estimation | Current frame | ← Forward Estimation | Future frame

↓ use both: bidirectional → will have twice no of motion vector so take avg of both

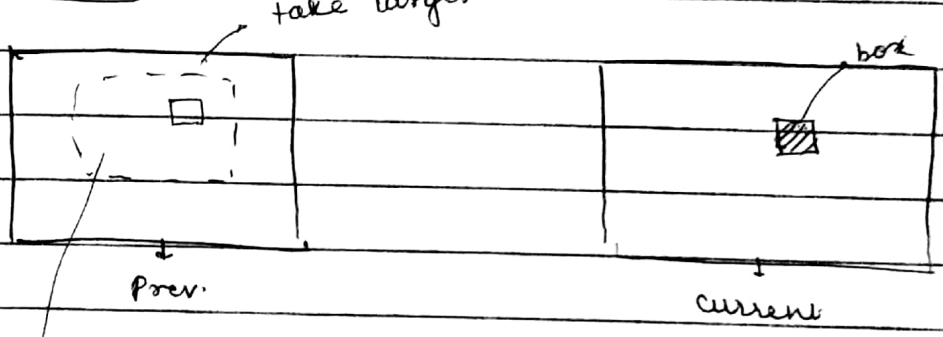
Teacher's Signature



Bidirectional gives better result in most cases

Basic approaches for Motion Estimation

1) Pixel-based



check each pixel & if they match ✓

- If have shadow in 1 case & not in other: though Object is moving, it won't have motion
- Though ~~image~~ Obj is not moving, it appears moving due to Δ in brightness : Optical Flow

↳ Computationally not efficient.

** ~~In most~~

⇒ We're sending Motion vector + Error frame also along with compression (current frame) isn't this much to send ? why is it better compression method

Matching criteria for Block-Based Motion Estimation

1) Mean square error (each pixel in block) → computational cost ↑
2) " of abs diff
3) Matching Pixel Count → In every dirn, find matching pixel : wherever max → take that vector.

- Block size : 16×16
- Search Range : Acc to size of block.
  (Reference Window) ⇒ If P ↑ : Window size should be ↑ & accuracy ↑
- Mostly, we use linear motion (translation)
- Complex : include Projective, Affine motion

Motion Estimation

→ Uniform motion inside each block.
→ Only consider 'Y' component (luminance)
→ for each block : motion vector

→ Inverse Quantize / DCT : for our feedback (correct or not)
→ Generally : 4:2:0 subsampling

HB

Video Compression : used in WhatsApp, Video Calls.

Video Coding Stds

→ Error in temporal predic^n is encoded by transform domain techniques (eg. DCT)

Project Report

10 - Coding
10 - Viva

PAGE NO.
DATE: / /

Book → Ch - 11

Since we exploit both spatial & temporal : called hybrid

→ Video compression system consists of :
- An encoder
- Compressed bit stream
- A decoder

→ don't describe how encoding/decoding is done
↓
provides great flexibility
(we can design our own algo.)

Major Video Coding Stds

1) VCEG : limited bandwidth
2) MPEG : store videos in devices

MPEG-1 Std → low bit rate data (video conferencing)

→ Supports only non-interlaced video
→ 4:2:0 chroma - subsampling

Motion Compensation in MPEG-1

P-frame : Predicted frame (Using forward (backward predic^n)
MPEG uses this in general

→ Bidirectional : better → 2 motion vectors (1 for fwd, 1 for bwd)

In general case : we take avg. of
If we don't get proper match in fwd : we will only consider 1 motion vector of bwd

1. I-frame : Spatial → least compression in 1 frame.
2. P-frame : Temporal (Bwd)
3. B-frame : Temporal → Max^m Compression
(both fwd & bwd)

Teacher's Signature

Considering compression in just 1 frame [not video compression]

Error frame : less info. than video frame.
- need 2 frames atleast
- (near to 0) → less bits needed.
- color info, channel info.

Inter-frame has more compression than Intra frame.

↳ If 2 frames aren't almost same. (error frame not near to 0.
Ex. : T.V. pe ekdum se commercial aa gaya) → can't remove temporal redundancy there (Inter frame). Have to consider spatial " " (Intra frame)

P-frame : Just taking 1 frame & finding error frame

B : " 2 " " "

gives max^m compression

Prediction will be better.
→ error b/w " & original will be ↓↓

* DCT ko motion vector nhi dete, error frame dete h.
motion vector ↓ given at time of entropy.

ide have P-frame : if only B-frame → quality may be ↓
so, have P-frame in b/w.

P-frame : can be predicted only using I or P frame (prev.)

B-frame : " I or P frame
↕ (Prev. / Next)

Page No.:
Date: / /

Hierarchical structure for MPEG-1 Bitstream

→ I/B/P.

→ GOP : collection of pictures

Interval b/w 2 I-frames : create 1 GOP.

° In starting of every GOP, 1 an I-frame.

→ Group of GOPs : sequence

5/11/18

MPEG-2 Std. → high bit rate data (HD systems)

- supports interlaced videos (supports progressive also)

adv. / disadv.

More the no. of frames, better is the quality of video

→ Scalable bit stream : scalable for diff. video quality

Q. Why interlaced.
How it helps in HD videos that MPEG supports interlaced ?

Ans Frame pictures are good → slower frame
Field → faster (rate is double)

→ Frames & fields : supports both.

→ Seq. can contain comp^n of frames & fields.
for prediction
We need to know whether it's coming from frame & field.

→ Field : separated into 2 pictures.

→ Frame predic^n : same as in MPEG 1

Teacher's Signature

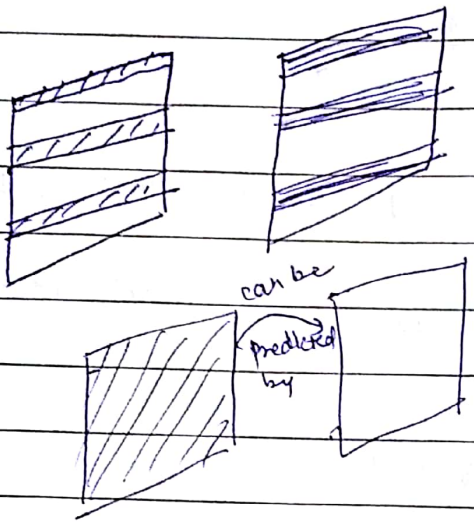
DATE: / /

Field pred^n :

Top field : → prev. top field of I/P
→ " bottom " " "

Bottom " : → " top " "
→ " bottom " "

Bottom field can be predicted from top field of same frame. (frame pred^n → can't do this)



Frame & Field :

→ a Diff. of size

size (field) = size (frame / 2)

→ Frame ~~can't predict~~ pred^n using prev. field ⇒ can't use top field of same frame to predict bottom field.

→ Scalability :

Bit stream organized into layers.

- Base layer : mandatory (carry min. req. info)
- Enhancement layer (can be > 1)

To ↑ quality, Enhancement layers ↑

Enhancement (To ↑ quality)

Base

∘ If my app^n ~~need~~ don't need high quality : decoder can just use base layer

→ Sending 1 bit stream, decoder can use that many layers that it needs (on basis of app^n) ⇒ Robustness ↑.

Teacher's Signature

DATE: / /

→ SNR scalability :

For base layer : take very . .

enhan ... : take inverse of DCT coeff & use very fine DCT

→ Spatial :

Base : More subsampling

→ Temporal : drop frames

Base : reduce to 1/2.

*→ Data Partitioning : not exactly scaling

MPEG 4

MPEG 1 & 2 : doing frame wise coding.

→ Here, it is content based video coding.

will separate objects in video & apply coding on separate objects → video object

Snapshot : 1 frame me 1 " . → video object in plane

Ex. News me ~~video~~ video chal rhi h usme kisi ko encircle kr rakha h.

→ combine synthetic & natural scene & objects.

Ex. Animations, Ganesh ji ke samne aarti ki thali ghoom rhi h

→ 1 video object can be in multiple frames

→ can change locations of video objects or exchange

→ Each VO has texture, shape.

Have to encode : " " & motion ← like earlier

→ VOL : pos" details should also be sent.

↓ 3 components

- shape - Motion - Texture (features) → combine nodes bit stream me jayegi

Each VO is sent separately by VOL.

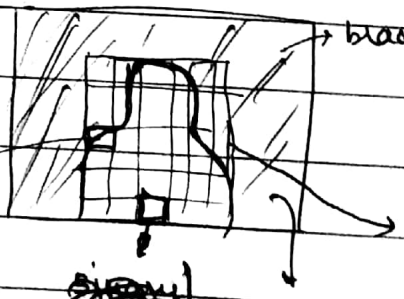
~~For~~ e

- doing this frame by frame : Motion Coding

Shape Coding

- need to compress shape while transmitting.

circle : Ckt : ~~of~~ identifying object (like blob)



① → Identify boundary

② → Divide ~~to~~ into 16×16 macroblocks (can be space in sides if boundary is not divisible by 16)

Binary & plane

outside boundary → certain blocks : all black ~~Binary & Plane~~

: all white

: comb" of black & white → Binary & Blocky

↓

These be needed to be encoded to transmit.

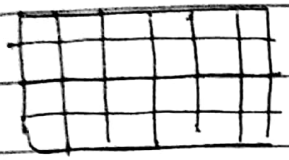
Teacher's Signature

DATE : / /

- Use context based Arithmatic coding (CAE)
  ↓
  take nbd instead of just 1 pixel

- Have : I - VOP , P - VOP , B - VOP

- Intra - CAE : limited to that frame only



→ Take 10 nbd pixels.

$2^{10} \rightarrow 1024$ (Considering Binary & Plane)

Prev. stats ke based pe we form a table of prob. using context info. (the pixels W ke baad 1 B ke aane ki prob. kya h). I have seq of prob. → can apply Arithmatic encoding then.

- Inter - CAE : have Reference & Target frame. → Uses motion estimation & compensation (VOs, not VOPs ↳ in just 1 frame)

  - Find best matching macroblock.

Q" : Can't use B-frame for pred". Why?

Q" : Have just I & B. Have to use B. How to maintain quality (if P is - nt) ⇓ (don't have much I-frames)

- 4 nbd pixels : Target frame
  5 " : Reference " } → 2^9 : 512
  ⇓
  again, apply above method & apply CAE.

Teacher's Signature.....

Texture Coding

Bounding box are used.

- we're just doing for object. Background is almost static.

so, error frame will be 0 for large area.

Error will be ↓ as compared to frame based coding in Object based coding.

↳ can add artificial thing

↳ can change content in bit stream

Sprite Coding

Sprite : large part of image almost static

Panaroma : stitch multiple images.

↓ sprite panaroma

- Here, stitch multiple scenes which're almost static.
- Can't apply warping wala algo (not planar here)
  (if distance ↑ ⇒ can be treated as ∪)

send panorama only once, send, object & camera param jab change hoga, bhej dega. So, not sending whole image everytime.

(naya object we got "due to rot" of camera)

↳ Used in Animations : sprite coding

↓

Meshing + Texture info : sent as sprite

MPEG-7 :→ advanced version of MPEG-4

Used : ↳ Searching & Retrieval from video

↳ not