CSE6706:
Advanced Digital Image Processing

Dr. Md. Monirul Islam

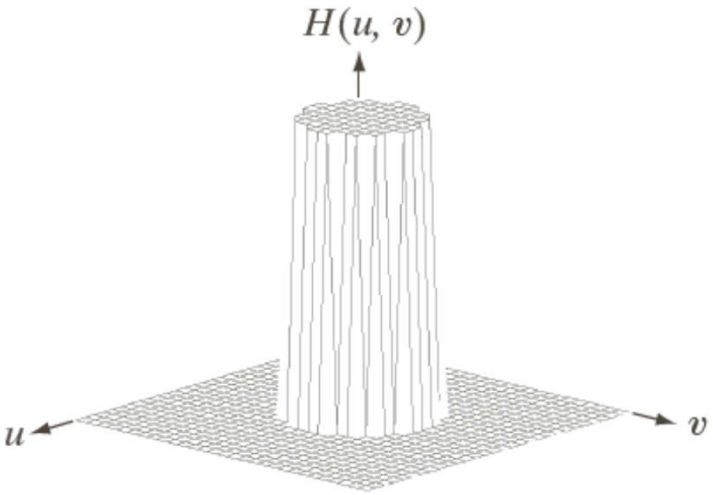
CSE-BUET

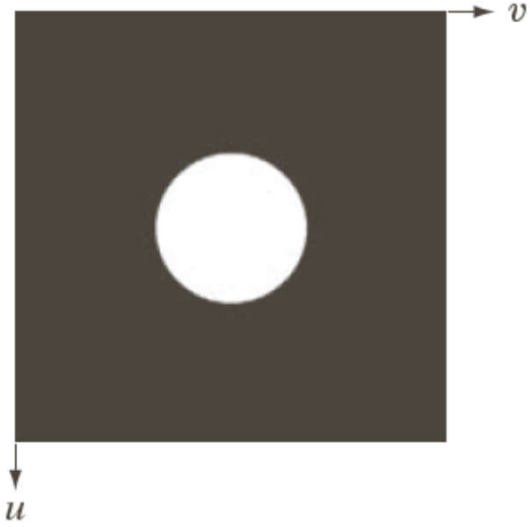
Spectral/Frequency Domain Analysis of Images

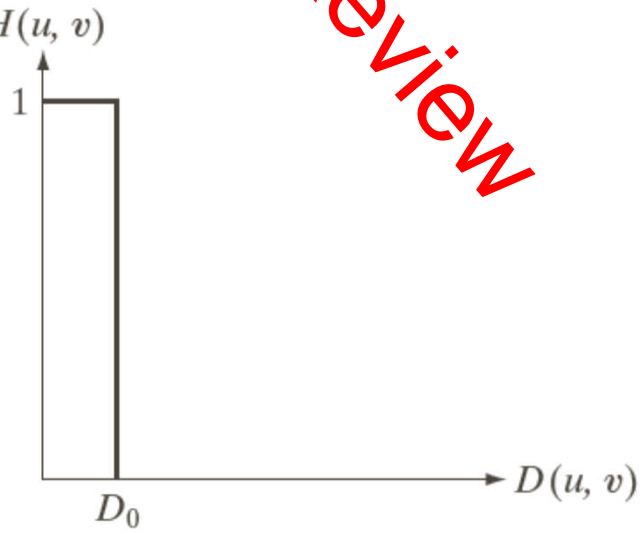
CSE-BUET

Ideal Low Pass Filter

Review







3D Surface View

Image View

Radial Cross Section

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

CSE-BUET

Image Sharpening Using High Pass Filters

$$H_{HP}(u,v) = 1 - H_{LP}(u,v)$$

High pass Filter

Low pass Filter

CSE-BUET

Recall: Ideal Low Pass Filter

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

$$\text{where, } \quad D(u,v) = \left[(u - M/2)^2 + (v - N/2)^2 \right]^{1/2}$$

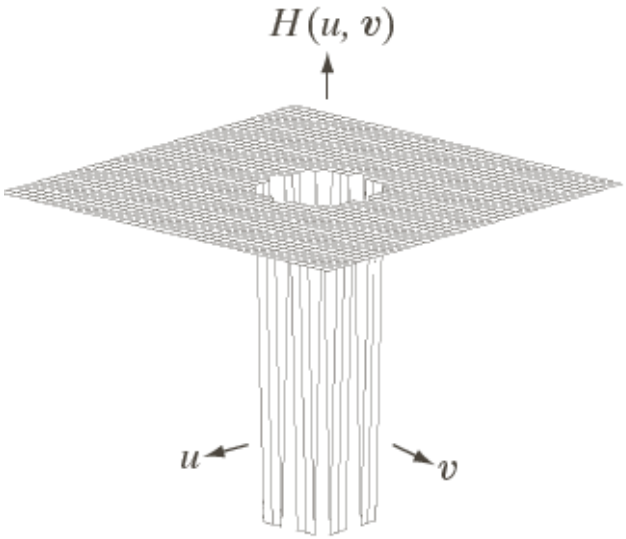
Ideal High Pass Filter (IHPF)

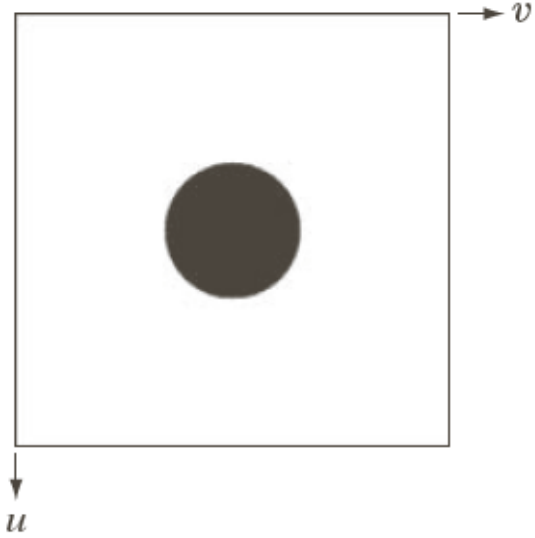
$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

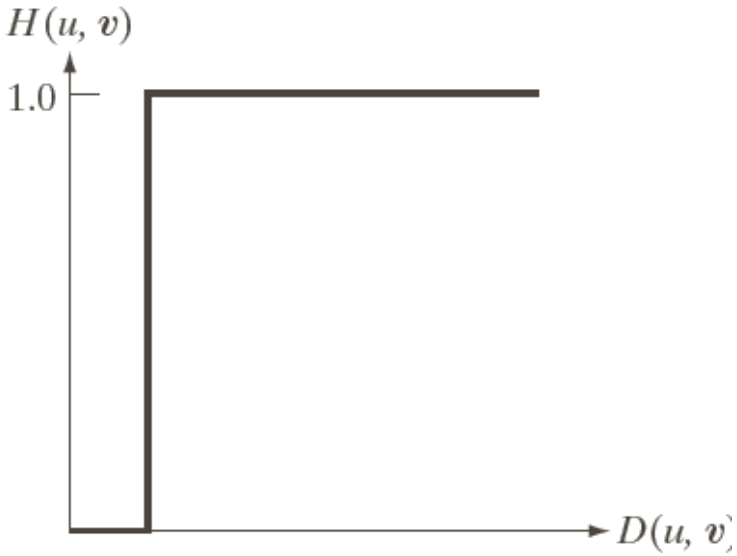
$$\text{where, } \quad D(u,v) = \left[(u - M/2)^2 + (v - N/2)^2 \right]^{1/2}$$

CSE-BUET

Ideal High Pass Filter







3D Surface View

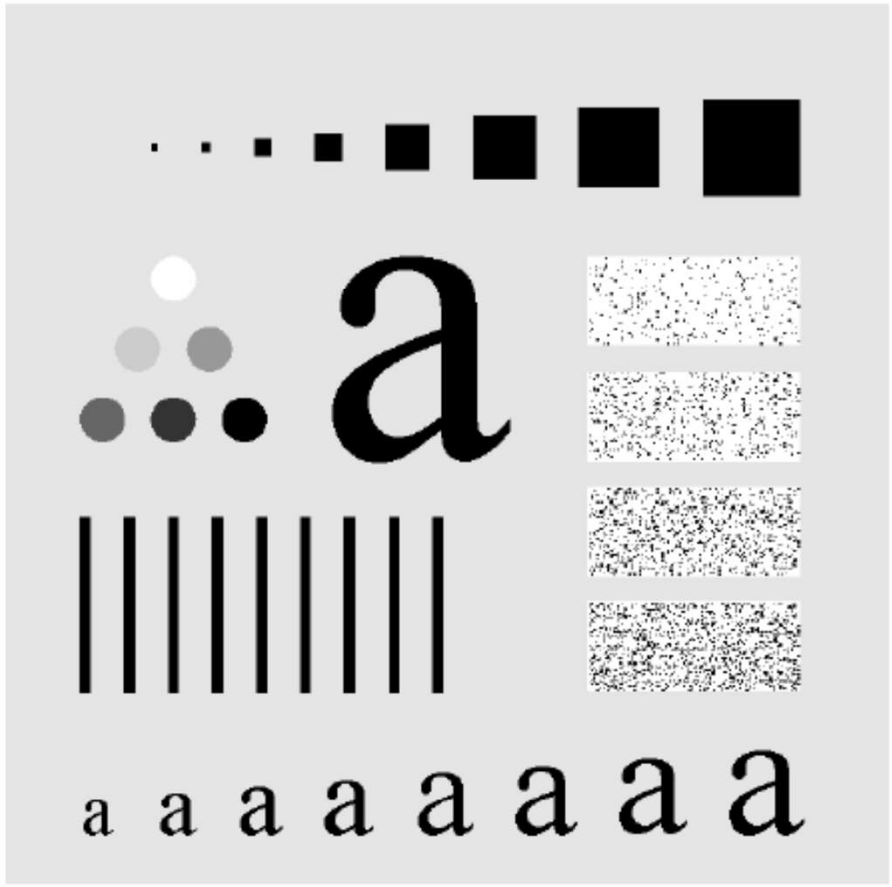
Image View

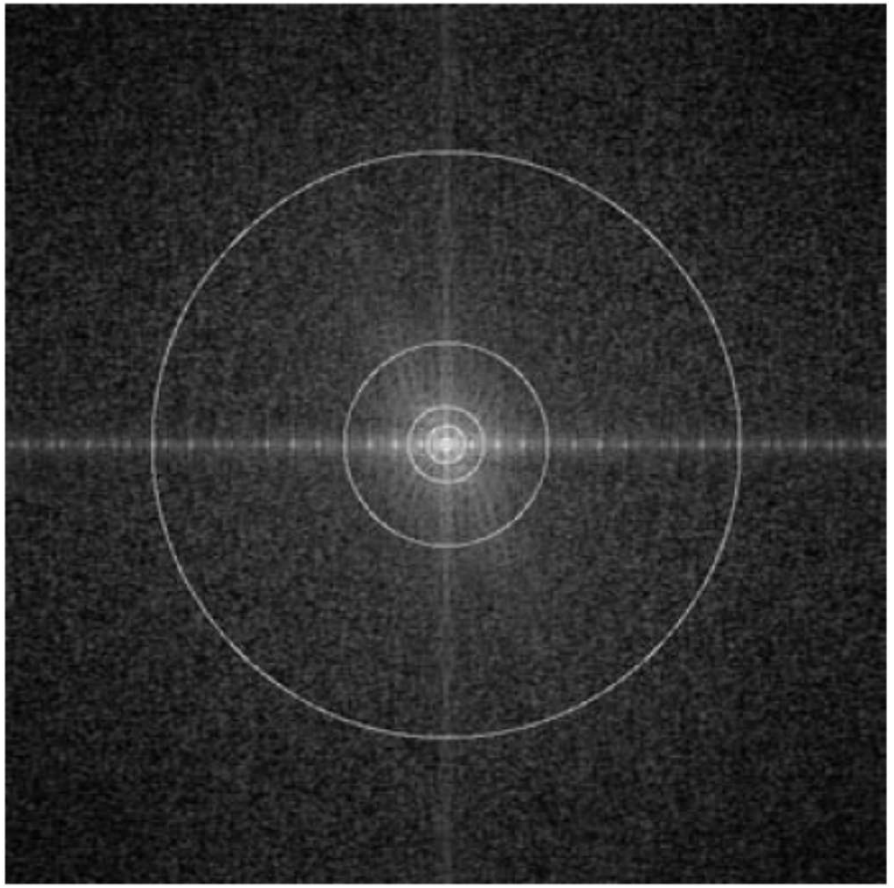
Radial Cross Section

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

CSE-BUET

Ideal High Pass Filter





The superimposed circles have radius 10, 30, 60, 160, 460 and enclose 87.0, 93.1, 95.7, 97.8 and 99.2% energy

CSE-BUET

Ideal High Pass Filter

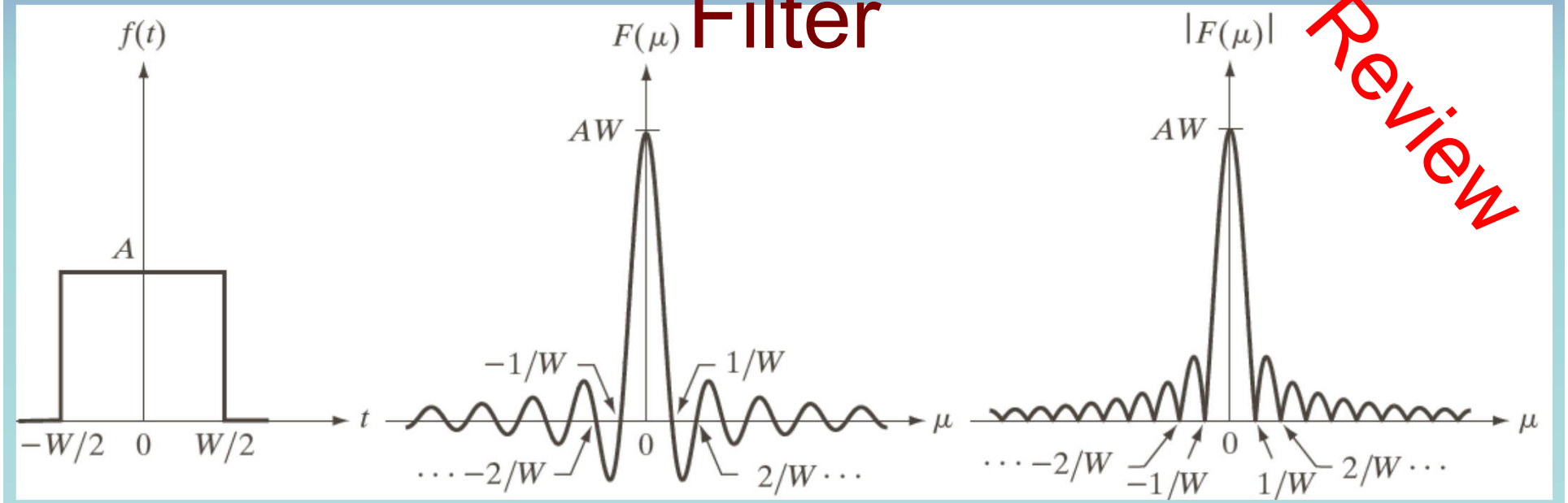
30 : 93.1 60 : 95.7 160 : 97.8

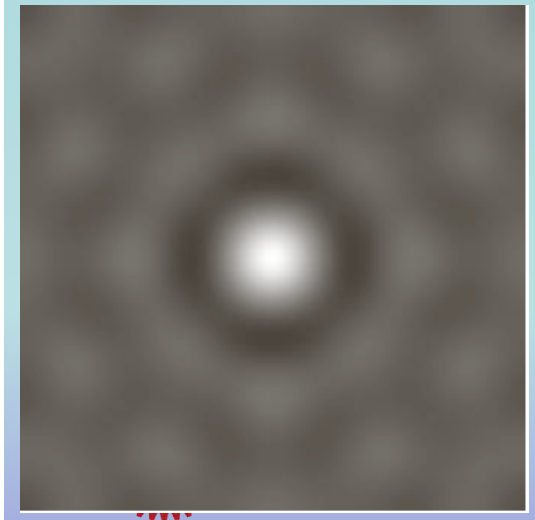
- Severe ringing effect with D_0 =30
  - Distorted, thickened object boundaries

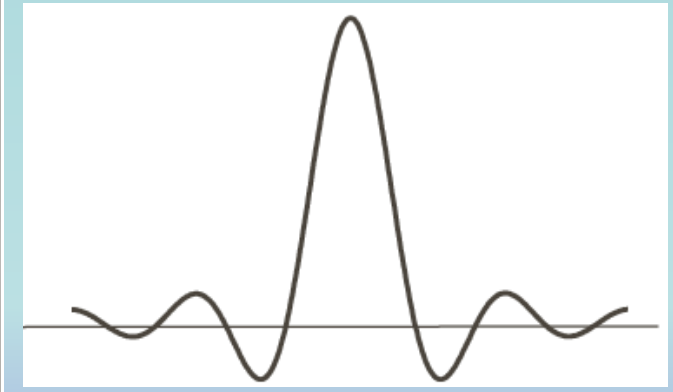
CSE-BUET

Box Function and Ideal Low Pass Filter

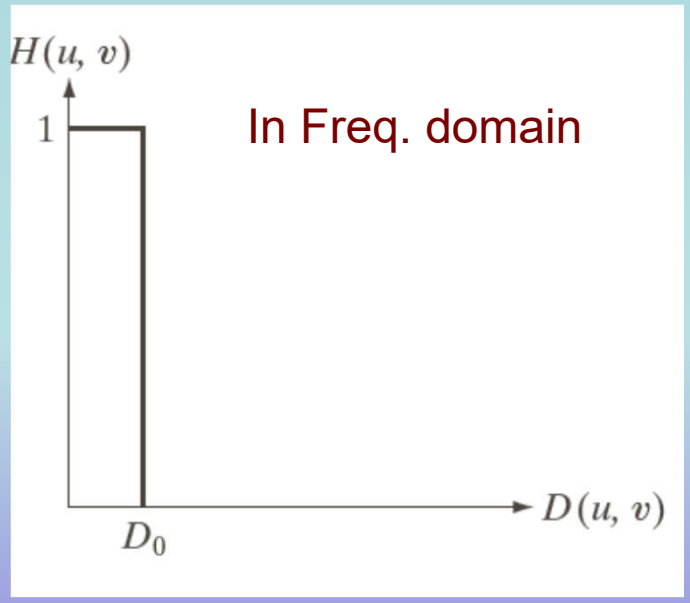
Review







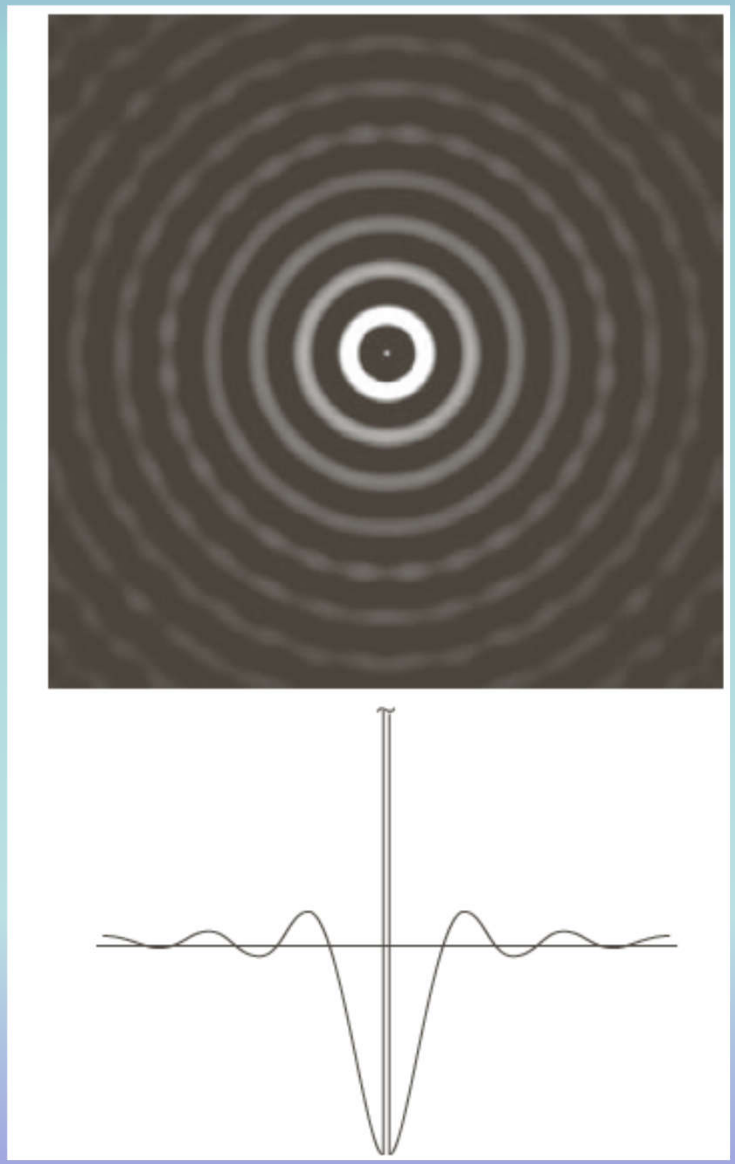
In Spatial domain



In Freq. domain

Ideal High Pass Filter

Spatial Domain Representation: $h(x, y)$



Intensity profile through the centre: $h(x)$

CSE-BUET

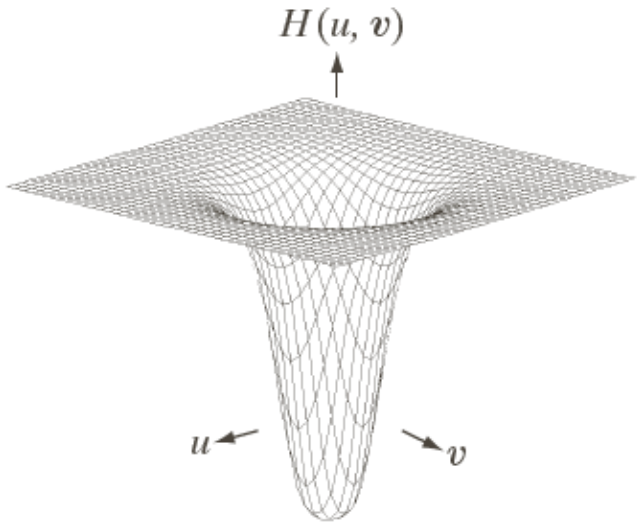
Ideal High Pass Filter

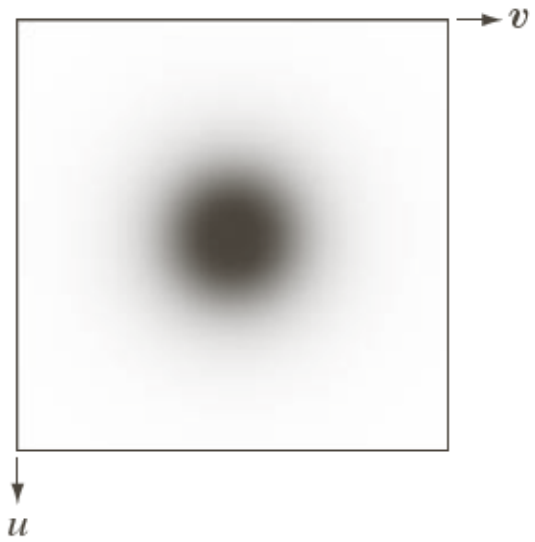
30 : 93.1 60 : 95.7 160 : 97.8

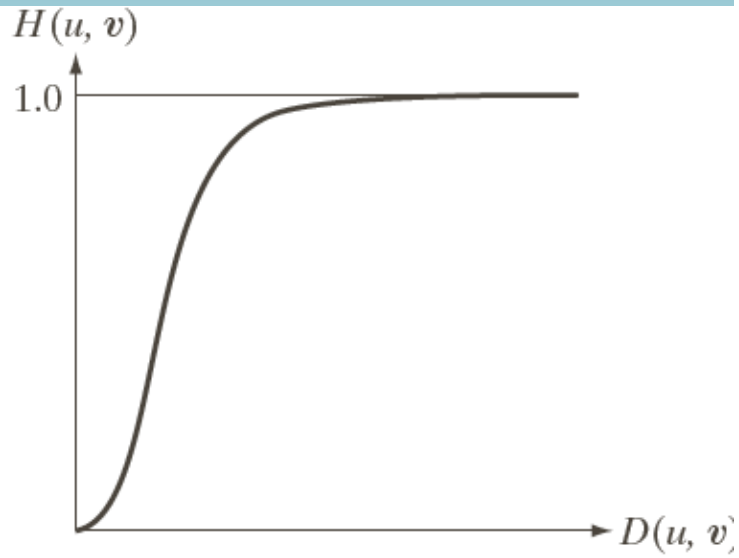
- Smaller circles and square are almost invisible when D_0 =30
  - Filtered magnitude is too small

CSE-BUET

Butterworth High Pass Filter (BHPF)







3D Surface View

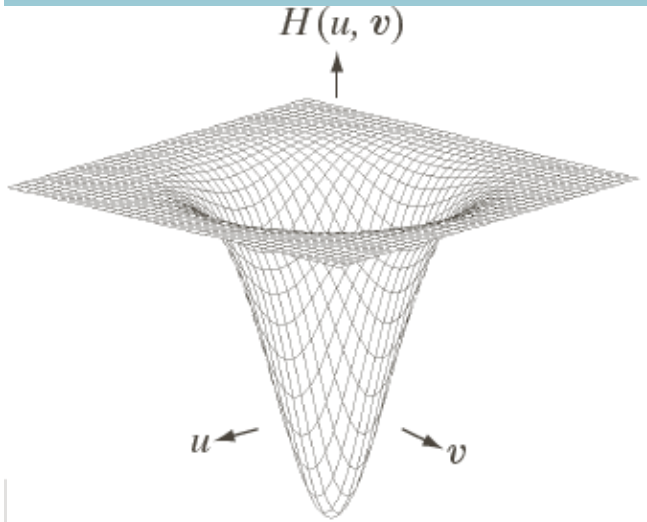
Image View

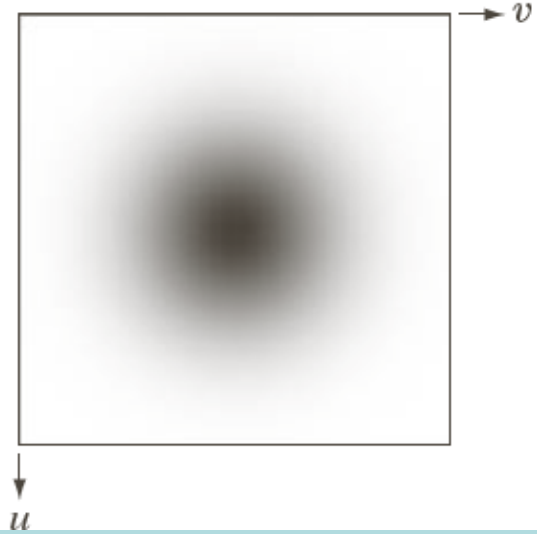
Radial Cross Section

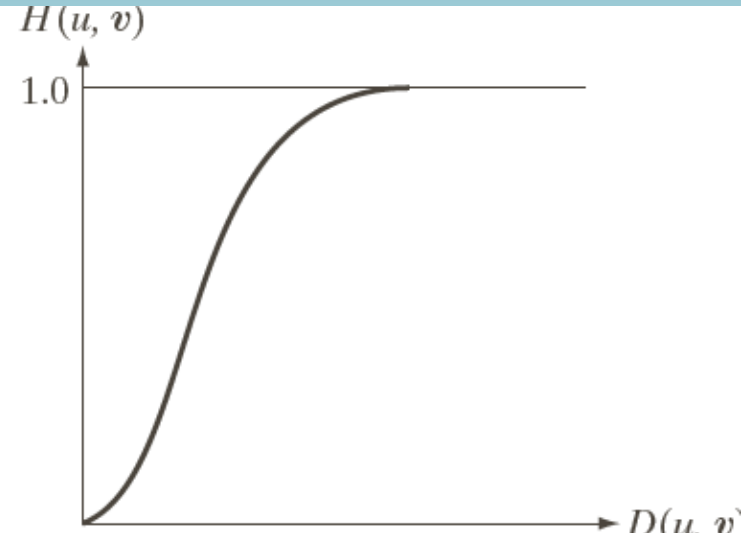
$$H(u,v) = \frac{1}{1+\left[D_0 / D(u,v)\right]^{2n}}$$

CSE-BUET

Gaussian High Pass Filter (GPLF)







3D Surface View

Image View

Radial Cross Section

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

or

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

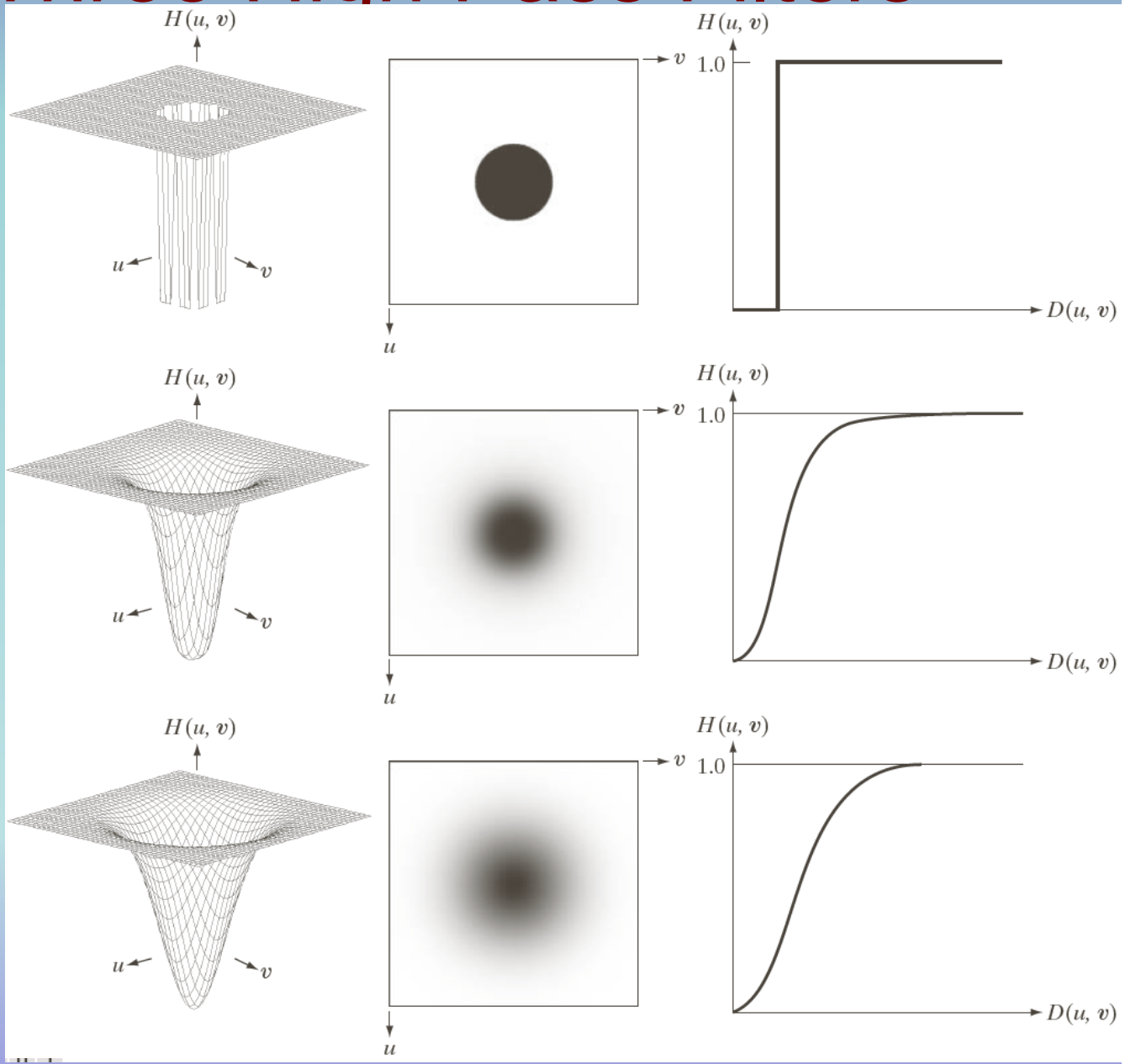
CSE-BUET

All the Three High Pass Filters

IHPF

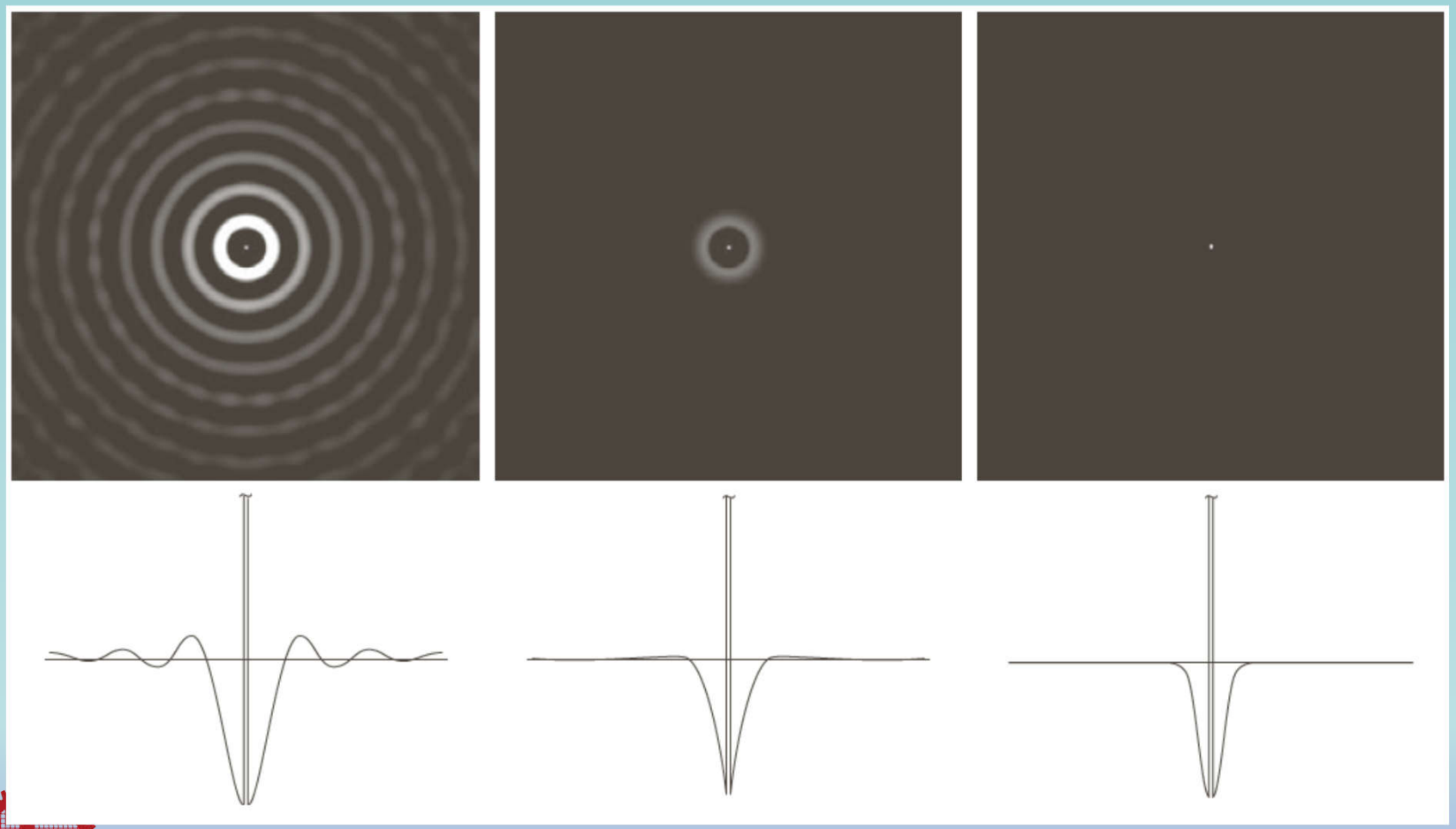
BHPF

GHPF



CSE-BUET

Ringing in the Three High Pass Filters



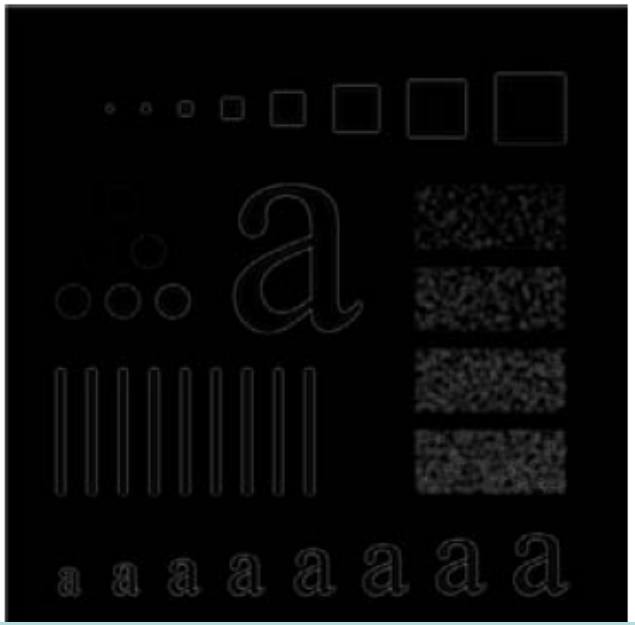
IHPF BHPF GHPF

Butterworth High Pass Filter with $n = 2$

30 : 93.1

60 : 95.7

160 : 97.8



CSE-BUET

Gaussian High Pass Filter

30 : 93.1

60 : 95.7

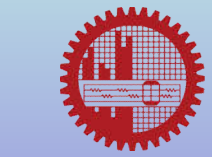
160 : 97.8

CSE-BUET

Laplacian in Frequency Domain

Laplacian or 2nd order derivative can be written in the frequency domain as

$$H(u,v) = -4\pi^2(u^2 + v^2)$$



CSE-BUET

Laplacian in Frequency Domain

$$H(u,v) = -4\pi^2(u^2 + v^2)$$

or

$$H(u,v) = -4\pi^2\left[(u - M/2)^2 + (v - N/2)^2\right] \text{ when centered}$$

CSE-BUET

Laplacian in Frequency Domain

$$H(u,v) = -4\pi^2(u^2 + v^2)$$

or

$$H(u,v) = -4\pi^2\left[(u - M/2)^2 + (v - N/2)^2\right] \text{ when centered}$$

$$= -4\pi^2 D^2(u,v)$$

CSE-BUET

Laplacian in Frequency Domain

Laplacian image:

$$\nabla^2 f(x,y) = \Im^{-1}\{F(u,v)H(u,v)\}$$

CSE-BUET

Laplacian in Frequency Domain

Laplacian image:

$$\nabla^2 f(x,y) = \Im^{-1}\{F(u,v)H(u,v)\}$$

Enhanced Image is obtained by

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

CSE-BUET

Laplacian in Frequency Domain

Laplacian image:

$$\nabla^2 f(x,y) = \Im^{-1}\{F(u,v)H(u,v)\}$$

This component has values that are far larger than that of $f(x, y)$

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

CSE-BUET

Laplacian in Frequency Domain

Steps :

1. normalize $f(x, y)$ into $[0\,1]$
2. divide $\nabla^2 f(x, y)$ by its maximum value
3. then find $g(x, y)$ as

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

Laplacian in Frequency Domain

Original image

After Laplacian filtering in frequency domain

CSE-BUET

Unsharp Masking

$$g_{mask}(x,y) = f(x,y) - f_{LP}(x,y)$$

Sharp Image

Smoothed Image

CSE-BUET

Unsharp Masking

$$g_{mask}(x,y) = f(x,y) - f_{LP}(x,y)$$

with

$$f_{LP}(x,y) = \Im^{-1}\{H_{LP}(u,v)F(u,v)\}$$

CSE-BUET

Highboost filtering

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

highboost filtering when $k > 1$

unsharp masking when $k = 1$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\Im\{g(x,y)\} = \Im\{f(x,y) + k \times g_{mask}(x,y)\}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\begin{aligned}\Im\{g(x,y)\} &= \Im\{f(x,y) + k \times g_{mask}(x,y)\} \\ &= \Im\{f(x,y)\} + k \times \Im\{g_{mask}(x,y)\}\end{aligned}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\begin{aligned}\Im\{g(x,y)\} &= \Im\{f(x,y) + k \times g_{mask}(x,y)\} \\ &= \Im\{f(x,y)\} + k \times \Im\{g_{mask}(x,y)\} \\ &= F(u,v) + k \times \Im\{f(x,y) - f_{LP}(x,y)\}\end{aligned}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\begin{aligned}\Im\{g(x,y)\} &= \Im\{f(x,y) + k \times g_{mask}(x,y)\} \\ &= \Im\{f(x,y)\} + k \times \Im\{g_{mask}(x,y)\} \\ &= F(u,v) + k \times \Im\{f(x,y) - f_{LP}(x,y)\}\end{aligned}$$

because, $g_{mask}(x,y) = f(x,y) - f_{LP}(x,y)$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\begin{aligned}
\Im\{g(x,y)\} &= \Im\{f(x,y) + k \times g_{mask}(x,y)\} \\
&= \Im\{f(x,y)\} + k \times \Im\{g_{mask}(x,y)\} \\
&= F(u,v) + k \times \Im\{f(x,y) - f_{LP}(x,y)\} \\
&= F(u,v) + k \times F(u,v) - k \times \Im\{f_{LP}(x,y)\}
\end{aligned}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

$$\begin{aligned}
\Im\{g(x,y)\} &= \Im\{f(x,y) + k \times g_{mask}(x,y)\} \\
&= \Im\{f(x,y)\} + k \times \Im\{g_{mask}(x,y)\} \\
&= F(u,v) + k \times \Im\{f(x,y) - f_{LP}(x,y)\} \\
&= F(u,v) + k \times F(u,v) - k \times \Im\{f_{LP}(x,y)\} \\
&= F(u,v) + k \times F(u,v) - k \times H_{LP}(u,v)F(u,v) \\
&= [1 + k \times [1 - H_{LP}(u,v)]]F(u,v)
\end{aligned}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = \Im^{-1}\{[1 + k \times [1 - H_{LP}(u,v)]]F(u,v)\}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = \Im^{-1}\left\{\left[1 + k \times H_{HP}(u,v)\right]F(u,v)\right\}$$

CSE-BUET

Highboost Filtering in Frequency Domain

$$g(x,y) = \Im^{-1}\{[1 + k \times H_{HP}(u,v)]F(u,v)\}$$

High Emphasis Filter

CSE-BUET

Generalized High Emphasis Filter

$$g(x,y) = \Im^{-1}\{[k_1 + k_2 \times H_{HP}(u,v)]F(u,v)\}$$

Generalized High Emphasis Filter

CSE-BUET

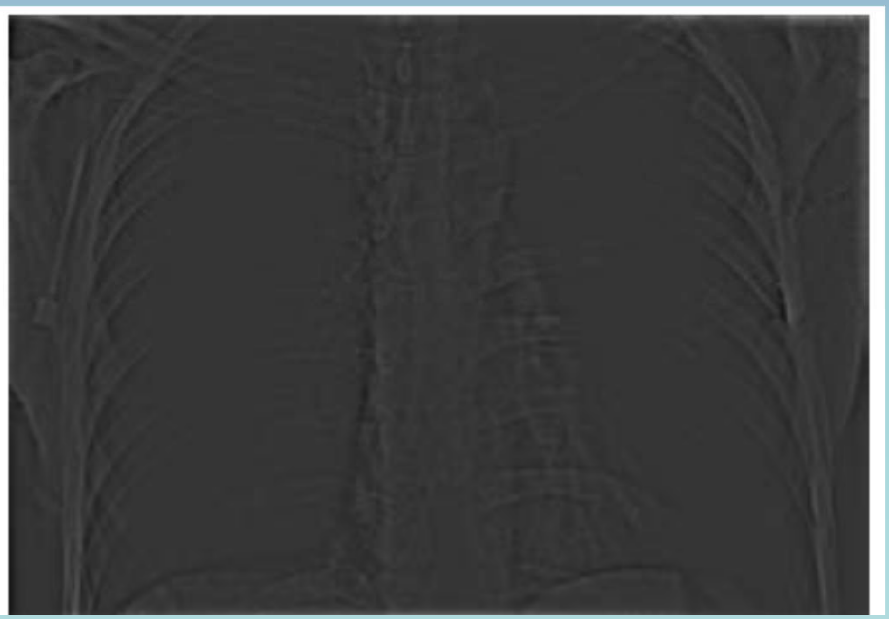
Example: Generalized High Emphasis Filter

Original Chaste X-ray Image

CSE-BUET

Example: Generalized High Emphasis Filter

Original Chaste X-ray Image

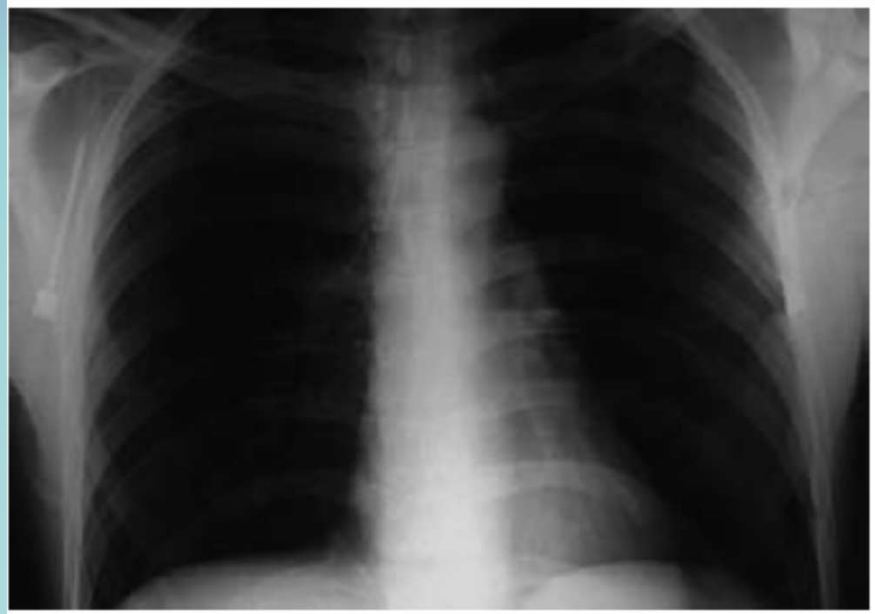


After GHPF with D_0 =40

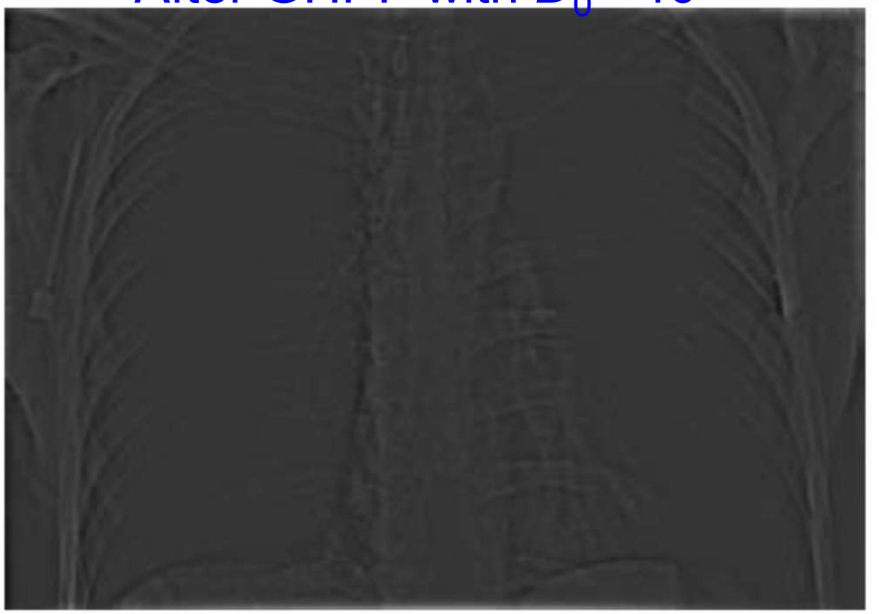
CSE-BUET

Example: Generalized High Emphasis Filter

Original Chaste X-ray Image



After GHPF with D_0 =40



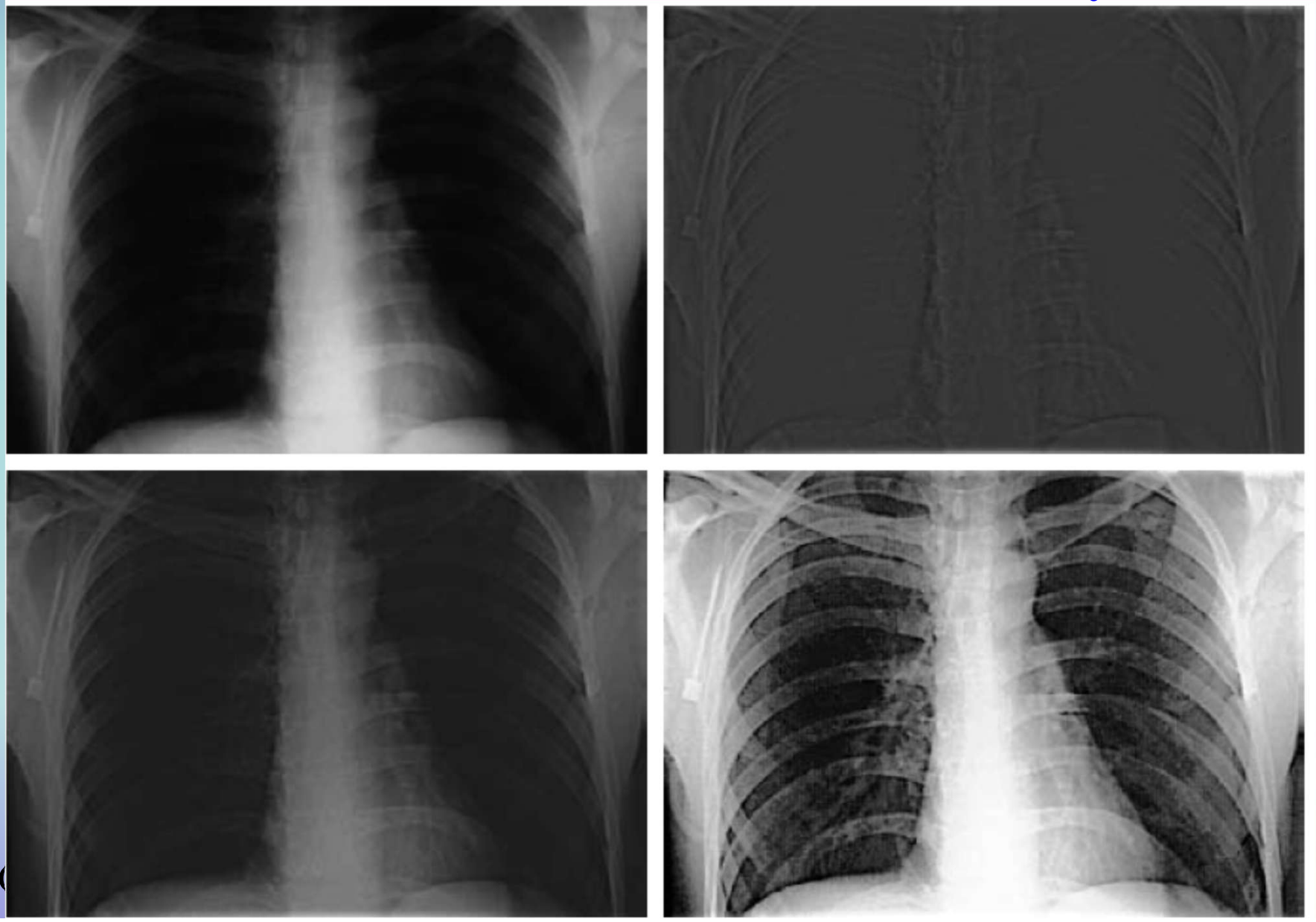
High Emphasis Filtering with

k_1 = 0.5 and
k_2= 0.75

Example: Generalized High Emphasis Filter

Original Chaste X-ray Image

After GHPF with D_0 =40



Histogram Equalization

Homomorphic Filtering

Recall the
illumination
reflectance model

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

CSE-BUET

Homomorphic Filtering

illumination
reflectance model

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

- Illumination is slowly varying component
  - Background, etc
- Reflectance tends to change abruptly
  - Sharp edges, object boundaries, etc

Homomorphic Filtering

illumination
reflectance model

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

- Illumination is slowly varying component : Low frequencies in FT
  - Background, etc
- Reflectance tends to change abruptly : High frequencies in FT
  - Sharp edges, object boundaries, etc

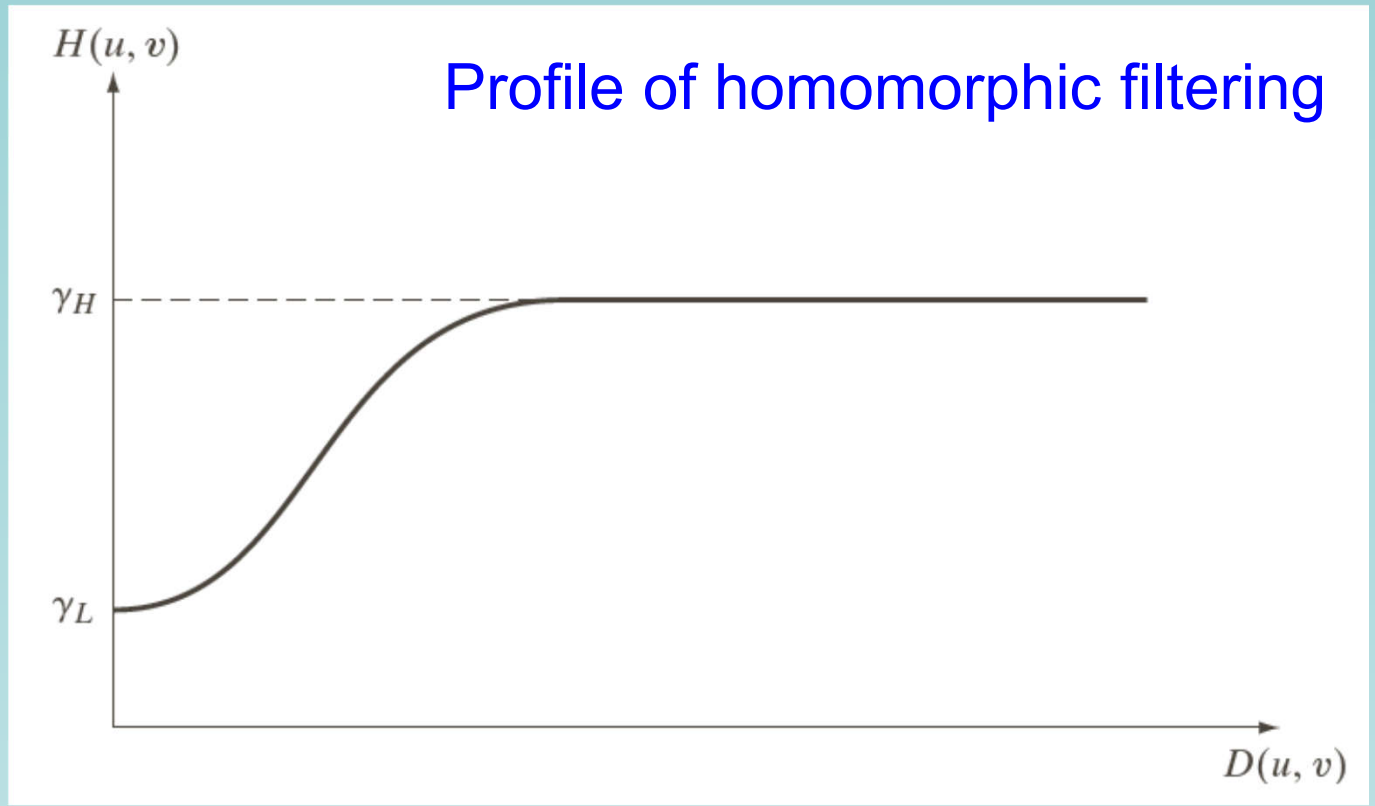
Homomorphic Filtering

illumination
reflectance model

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

- A good deal would be to treat both of them simultaneously but in different ways

Homomorphic Filtering



$$H(u,v) = (\gamma_H - \gamma_L)\left[1 - e^{c\left[D^2(u,v)/D_0^2\right]}\right] + \gamma_L$$

CSE-BUET

Homomorphic Filtering

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

$$\Im\{f(x,y)\} \neq \Im\{i(x,y)\}\Im\{r(x,y)\}$$

CSE-BUET

Homomorphic Filtering

$$z(x,y) = \ln f(x,y)$$
$$= \ln i(x,y) + \ln r(x,y)$$

CSE-BUET

Homomorphic Filtering

$$z(x,y) = \ln f(x,y)$$
$$= \ln i(x,y) + \ln r(x,y)$$

Then

$$\Im\{z(x,y)\} = \Im\{\ln f(x,y)\}$$
$$= \Im\{\ln i(x,y)\} + \Im\{\ln r(x,y)\}$$

CSE-BUET

Homomorphic Filtering

$$z(x,y) = \ln f(x,y)$$
$$= \ln i(x,y) + \ln r(x,y)$$

Then

$$\Im\{z(x,y)\} = \Im\{\ln f(x,y)\}$$
$$= \Im\{\ln i(x,y)\} + \Im\{\ln r(x,y)\}$$

OR

$$Z(u,v) = F_i(u,v) + F_r(u,v)$$

CSE-BUET

Homomorphic Filtering

Let a Filter $H(u,v)$

CSE-BUET

Homomorphic Filtering

Let a Filter $H(u,v)$

$$S(u,v) = H(u,v)Z(u,v)$$

$$= H(u,v)F_i(u,v) + H(u,v)F_r(u,v)$$

CSE-BUET

Homomorphic Filtering

Let a Filter $H(u,v)$

$$S(u,v) = H(u,v)Z(u,v)$$

$$= H(u,v)F_i(u,v) + H(u,v)F_r(u,v)$$

Then

$$s(x,y) = \Im^{-1}\{S(u,v)\}$$

CSE-BUET

Homomorphic Filtering

Let a Filter $H(u,v)$

$$S(u,v) = H(u,v)Z(u,v)$$
$$= H(u,v)F_i(u,v) + H(u,v)F_r(u,v)$$

Then

$$s(x,y) = \Im^{-1}\{S(u,v)\}$$
$$= \Im^{-1}\{H(u,v)F_i(u,v)\} + \Im^{-1}\{H(u,v)F_r(u,v)\}$$
$$= i'(x,y) + r'(x,y)$$

CSE-BUET

Homomorphic Filtering

Let a Filter $H(u,v)$

$$S(u,v) = H(u,v)Z(u,v)$$
$$= H(u,v)F_i(u,v) + H(u,v)F_r(u,v)$$

Then

$$s(x,y) = \Im^{-1}\{S(u,v)\}$$
$$= \Im^{-1}\{H(u,v)F_i(u,v)\} + \Im^{-1}\{H(u,v)F_r(u,v)\}$$
$$= i'(x,y) + r'(x,y)$$

Thus

$$s(x,y) = i'(x,y) + r'(x,y)$$

Homomorphic Filtering

Thus

$$s(x,y) = i'(x,y) + r'(x,y)$$

CSE-BUET

Homomorphic Filtering

Thus

$$s(x,y) = i'(x,y) + r'(x,y)$$

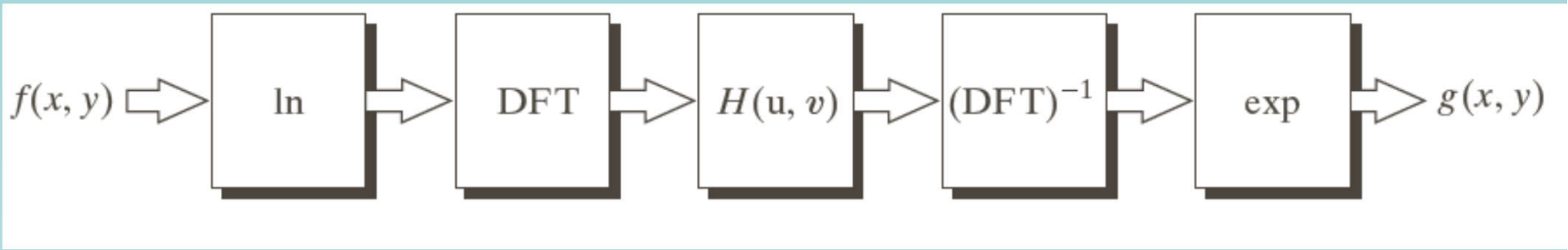
Finaly reversing the process,

$$g(x,y) = e^{s(x,y)} = e^{i'(x,y)+r'(x,y)}$$

$$= e^{i'(x,y)} e^{r'(x,y)}$$

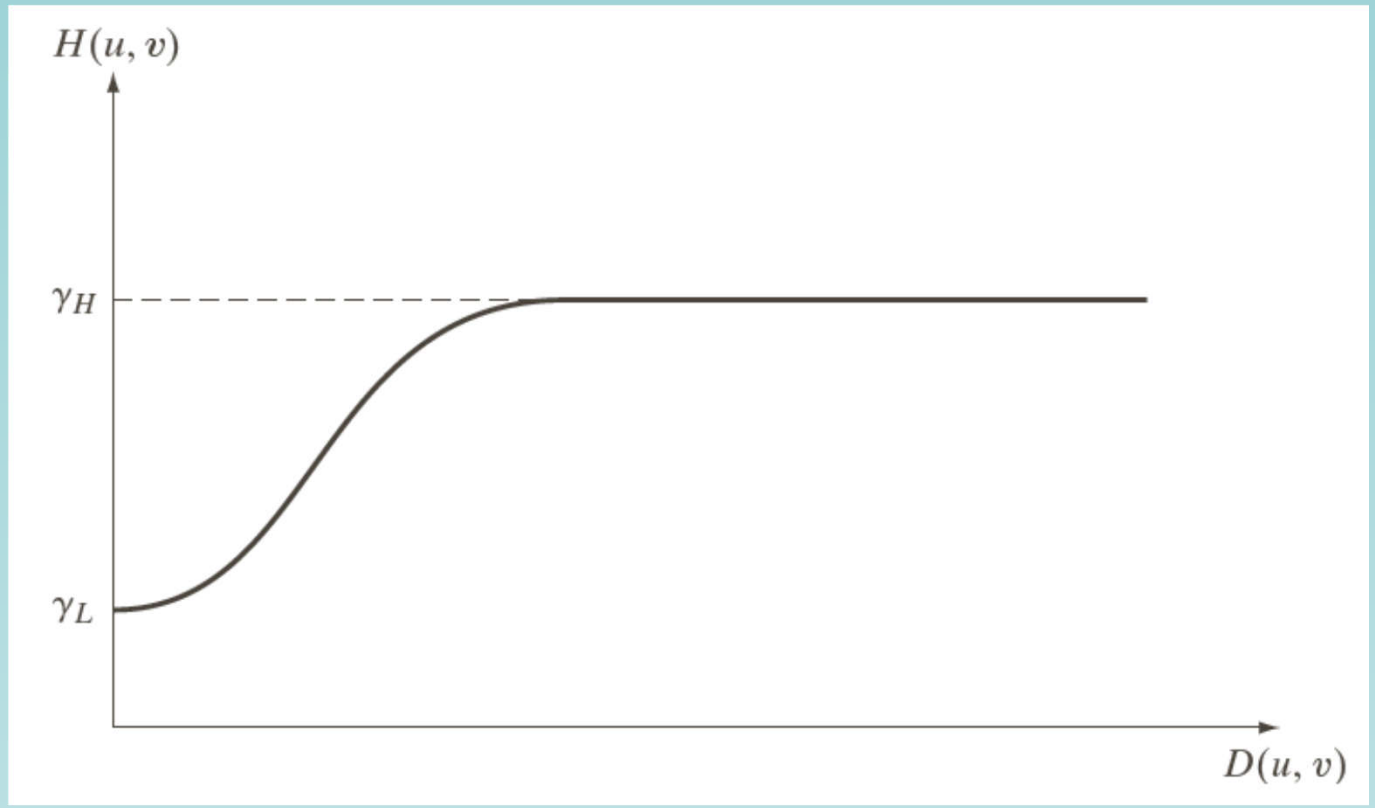
$$= i_0(x,y) r_0(x,y)$$

Homomorphic Filtering



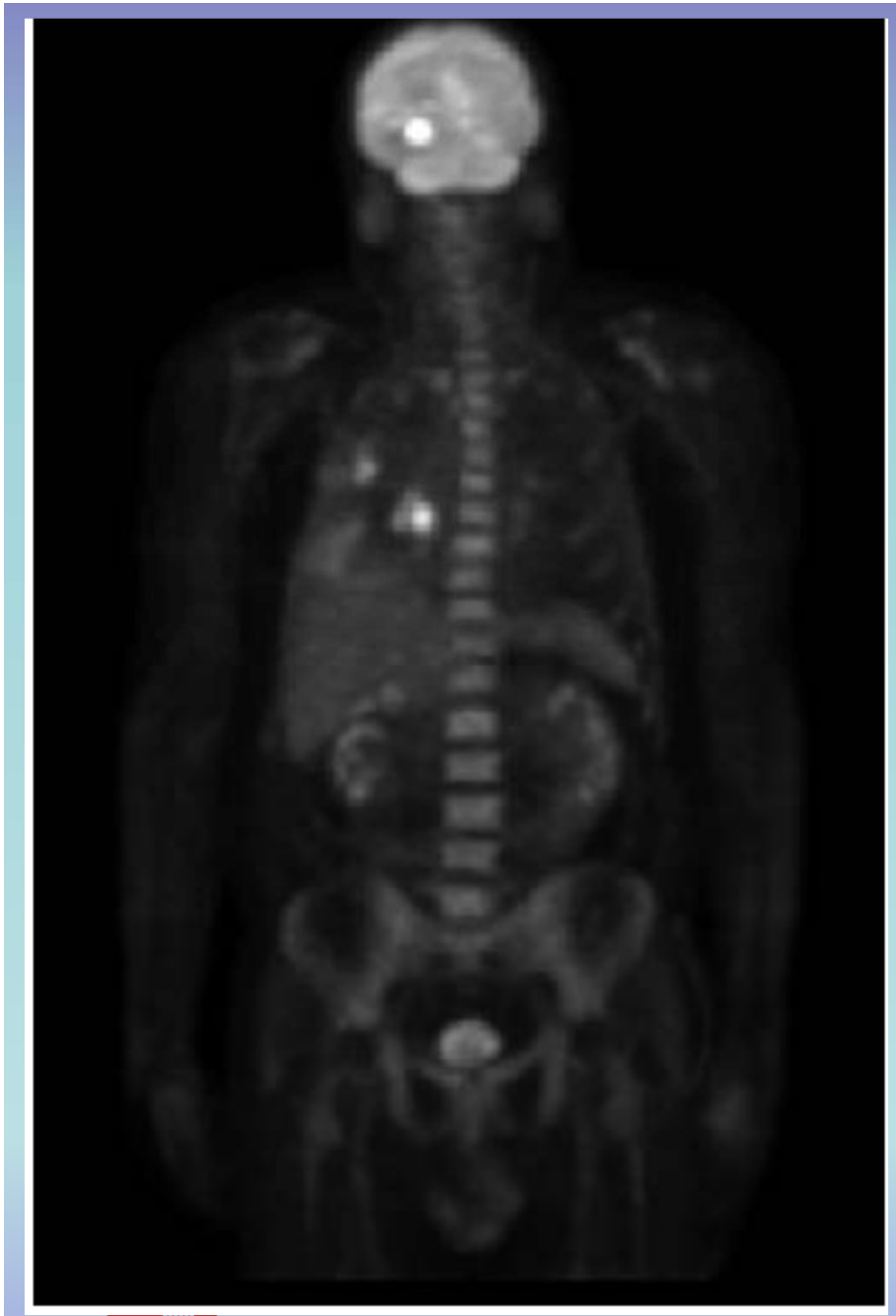
CSE-BUET

Homomorphic Filtering

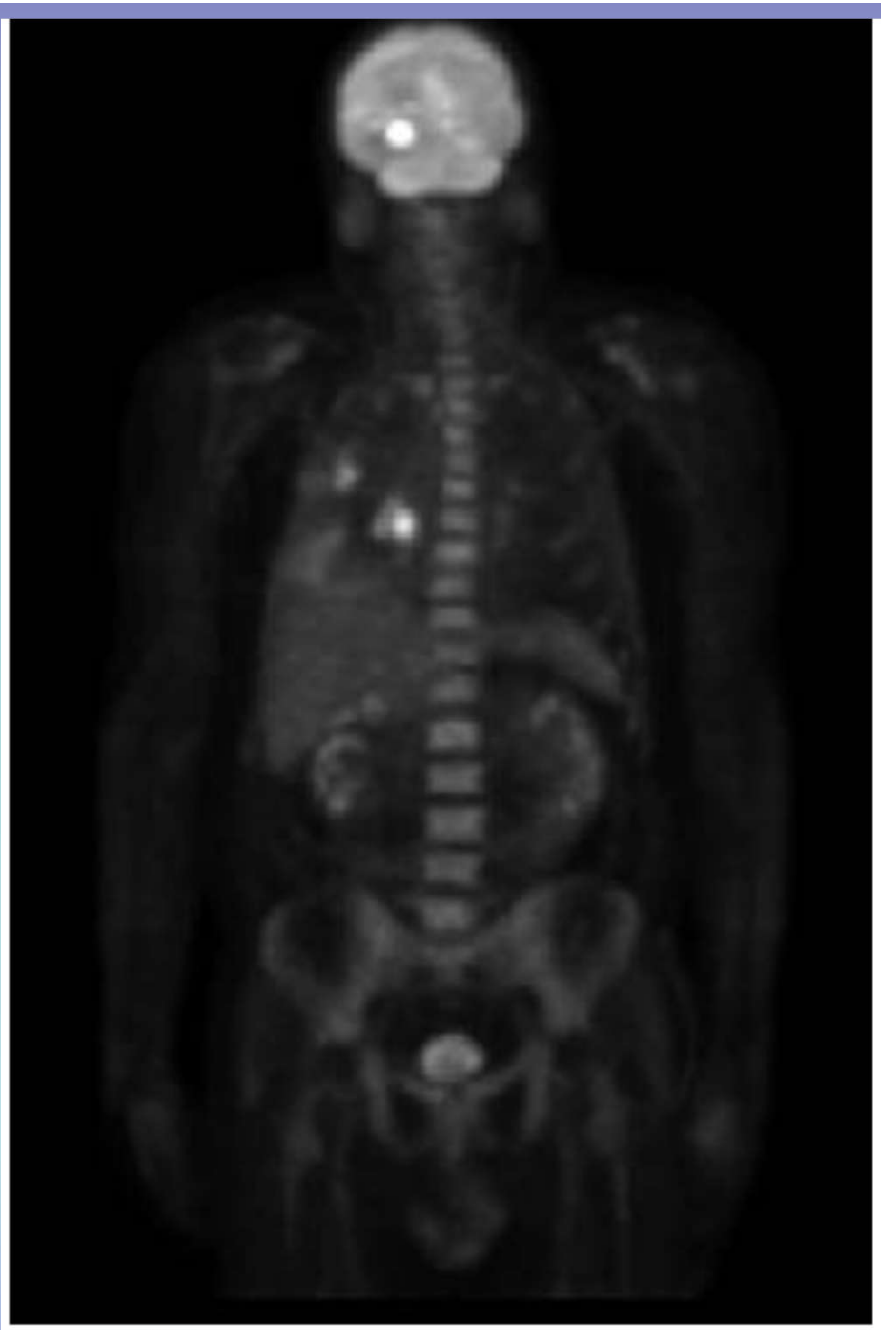


$$H(u,v) = (\gamma_H - \gamma_L)\left[1 - e^{c\left[D^2(u,v)/D_0^2\right]}\right] + \gamma_L$$

CSE-BUET



Example: Homomorphic Filtering



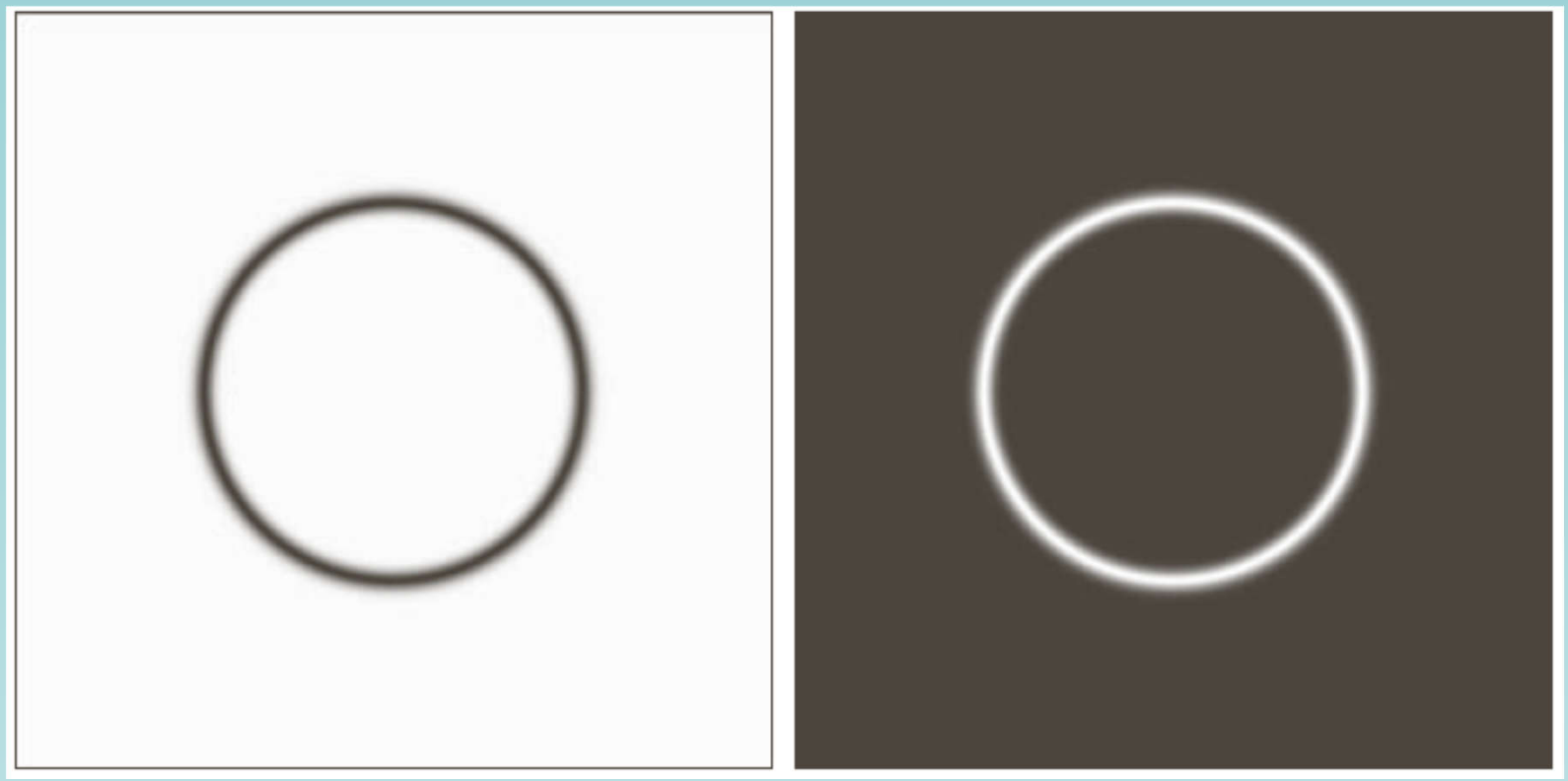
CSE-BUET

$$\gamma_L = 0.25, \gamma_H = 2, c = 1, D_0 = 80$$

Selective Filtering

- Filtering in a small or a particular range of frequencies instead of the whole frequency rectangle
  - Bandpass filtering
  - Band reject filtering
  - Notch filtering

Bandpass or Bandreject Filtering

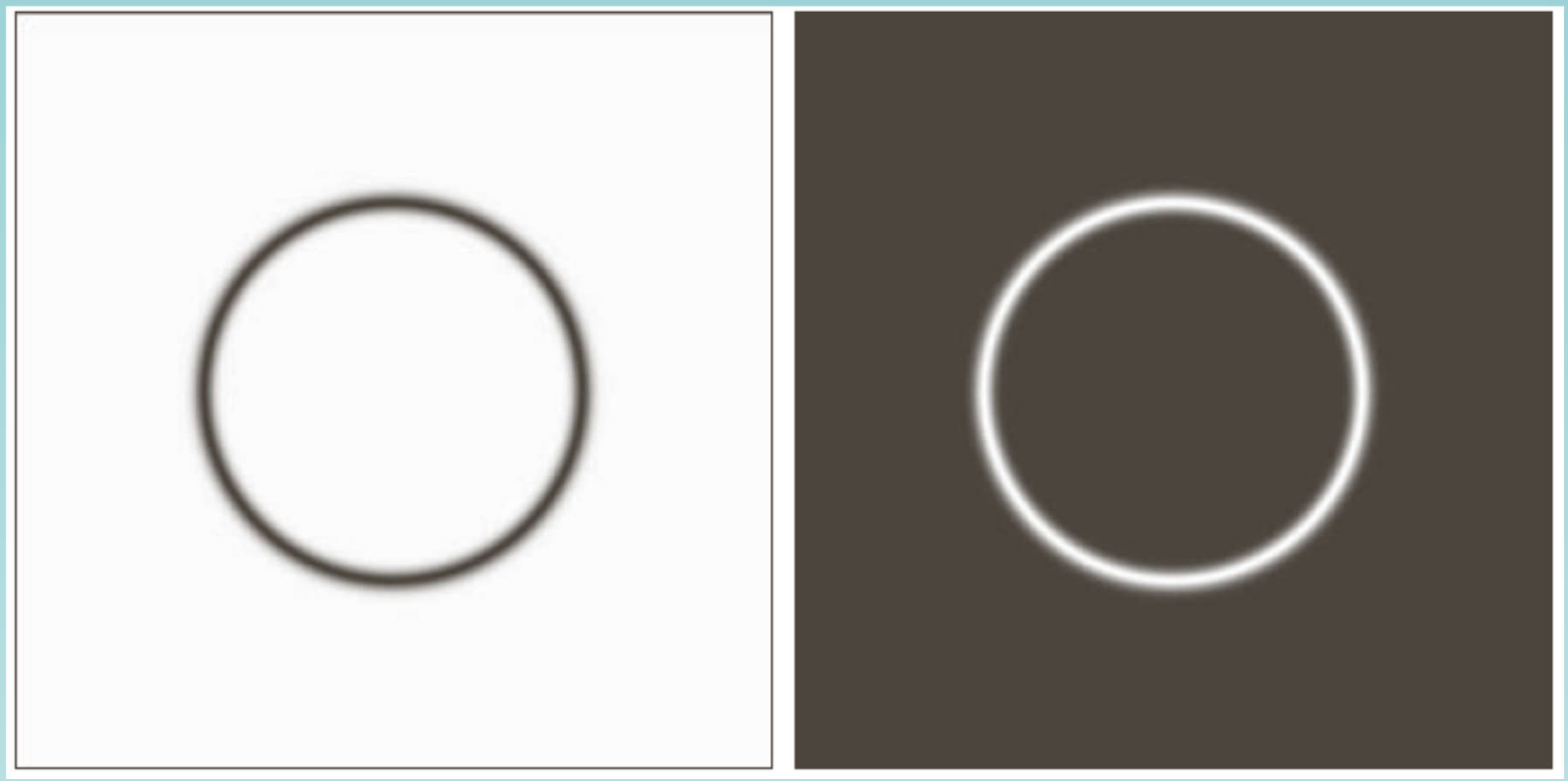


Bandreject Filter

Bandpass Filter

CSE-BUET

Bandpass or Bandreject Filtering



Bandreject Filter

Bandpass Filter

CSE-BUET

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

Different Bandreject Filters

Ideal Bandreject Filter:

$$H(u,v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \le D \le D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$$

Butterworth Bandreject Filter:

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

Gaussian Bandreject Filter:

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

Notch Filter

- Notch filter passes or rejects frequencies in a predefined region
- Comes as symmetric twins centred at
  - (u_0, v_0) and $(-u_0, -v_0)$

Notch Reject Filter

- Notch reject filter is product of highpass filters centered at notches

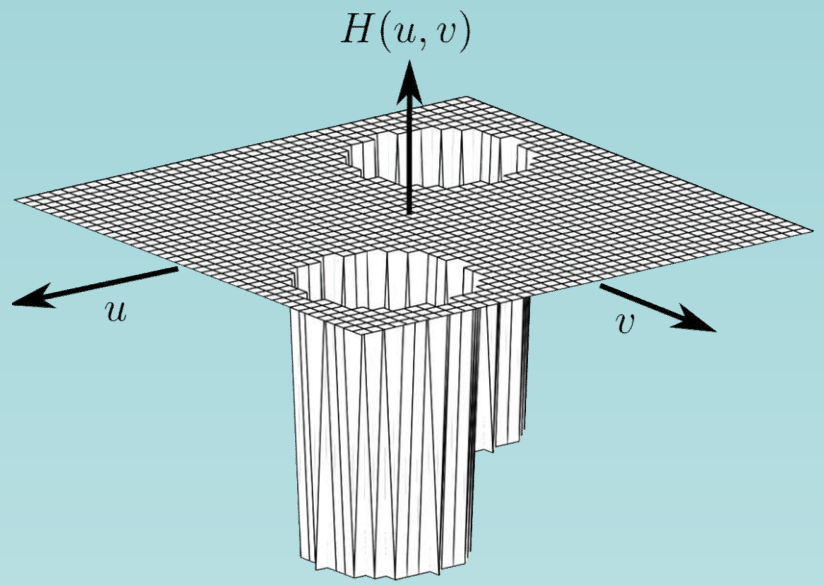
$$H_{NR}(u,v) = \prod_{k=1}^{Q} H_k(u,v) H_{-k}(u,v)$$

where

$H_k(u,v)$ and $H_{-k}(u,v)$ are highpass filters

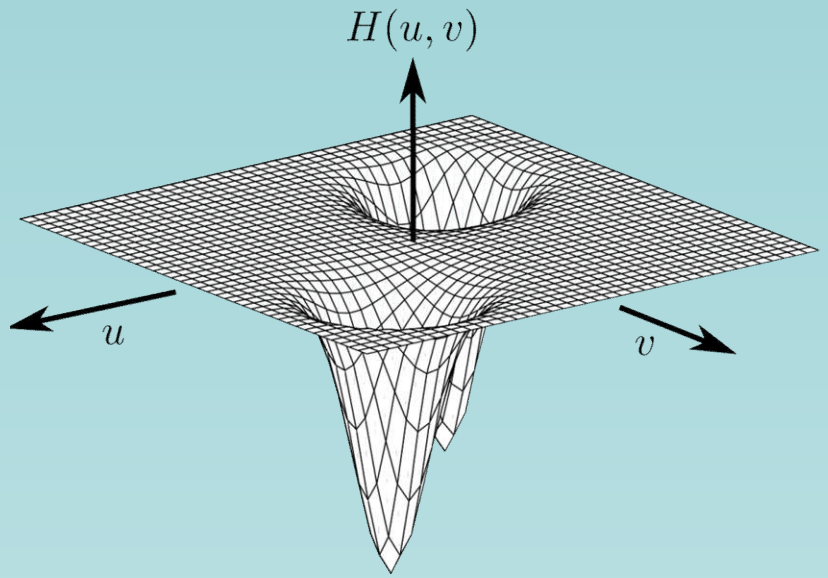
centred at (u_k, v_k) and $(-u_k, -v_k)$

Ideal Notch Reject Filter



CSE-BUET

Gaussian Notch Reject Filter



CSE-BUET

Butterworth Notch Reject Filter

$$H_{NR}(u,v) = \prod_{k=1}^{Q} \left[\frac{1}{1+\left[D_{0k} / D_k(u,v)\right]^{2n}} \right] \left[\frac{1}{1+\left[D_{0k} / D_{-k}(u,v)\right]^{2n}} \right]$$

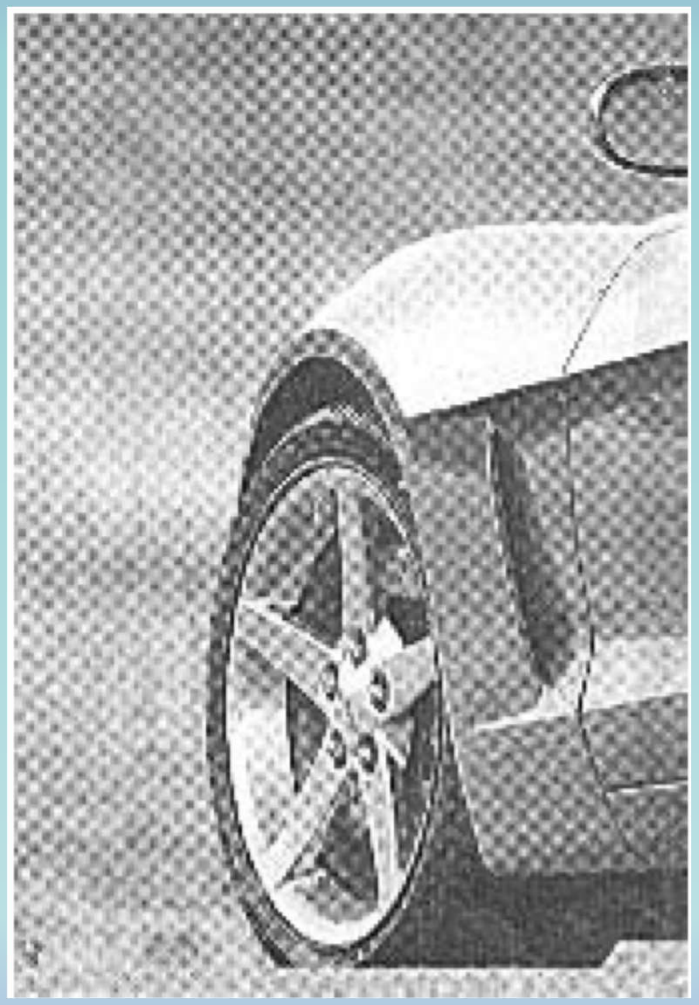
where,

$$D_k(u,v) = \left[(u - M/2 - u_k)^2 + (v - N/2 - v_k)^2 \right]^{1/2} \text{ and}$$

$$D_{-k}(u,v) = \left[(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2 \right]^{1/2}$$

CSE-BUET

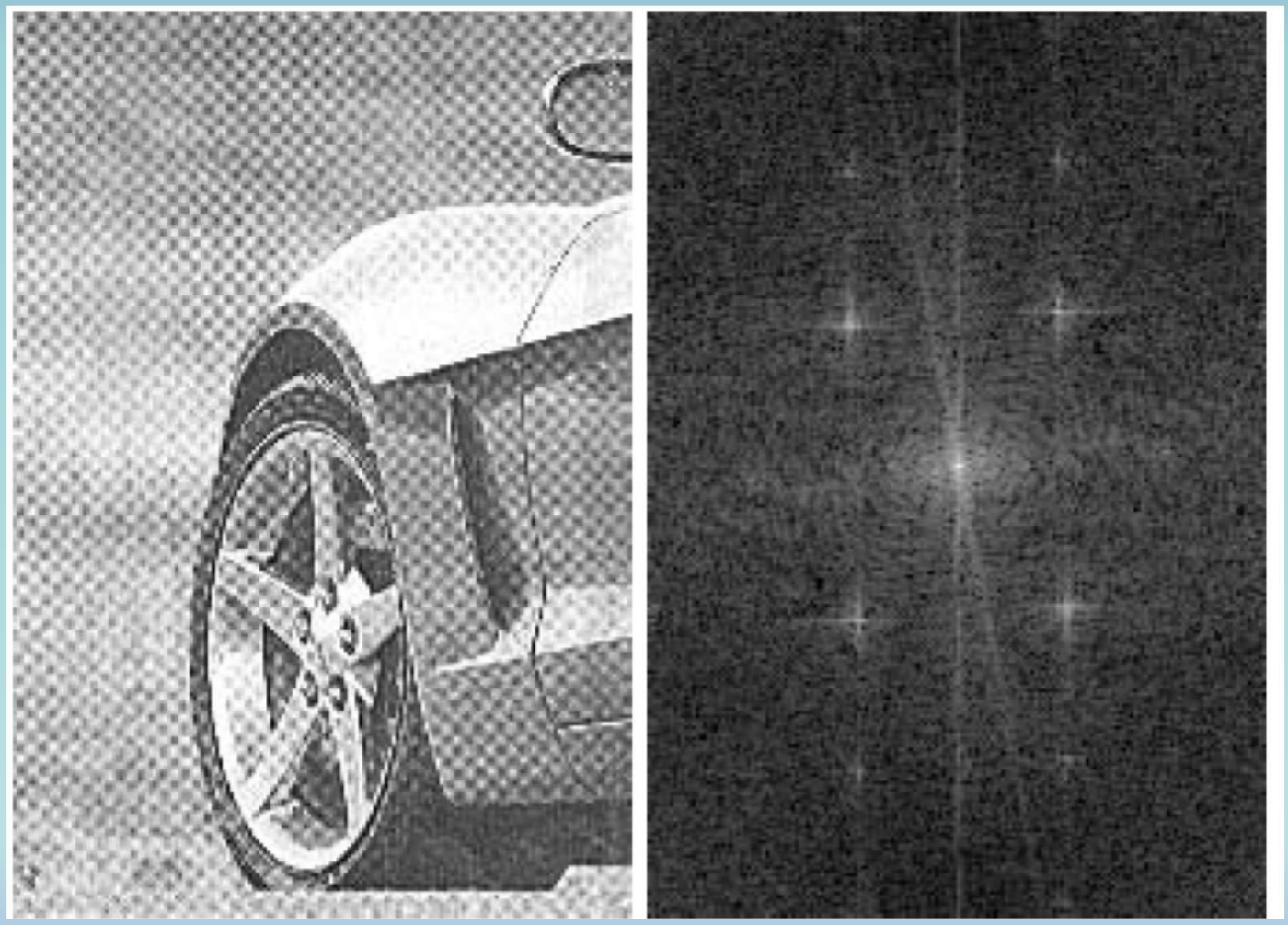
Example: Butterworth Notch Reject Filter



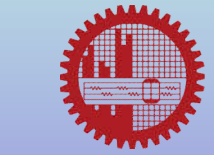
A scanned news paper image with moire effect

CSE-BUET

Example: Butterworth Notch Reject Filter

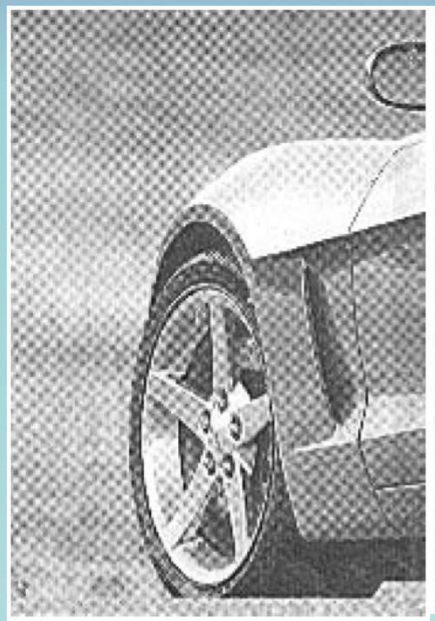


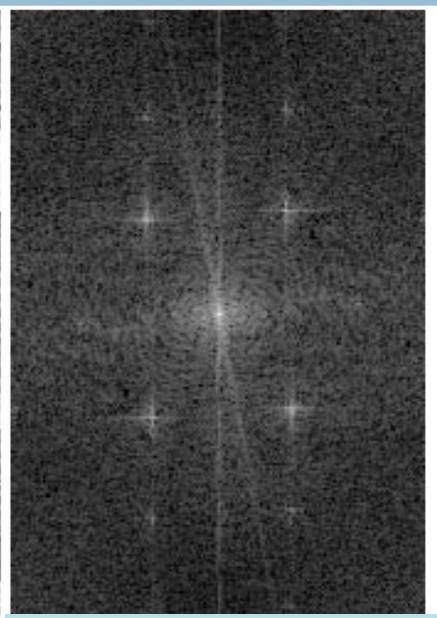
Burst of energy at some symmetric locations

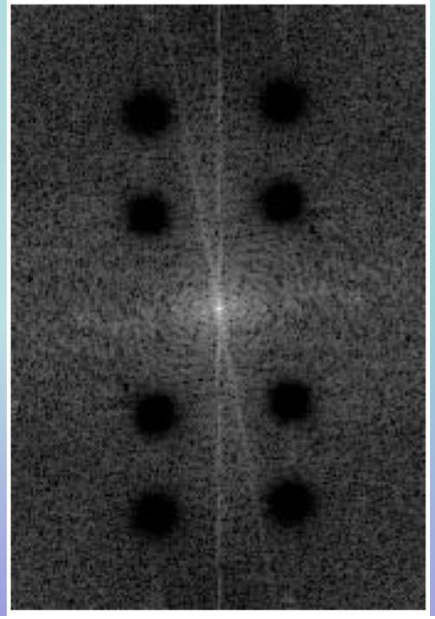


CSE-BUET

Example: Butterworth Notch Reject Filter



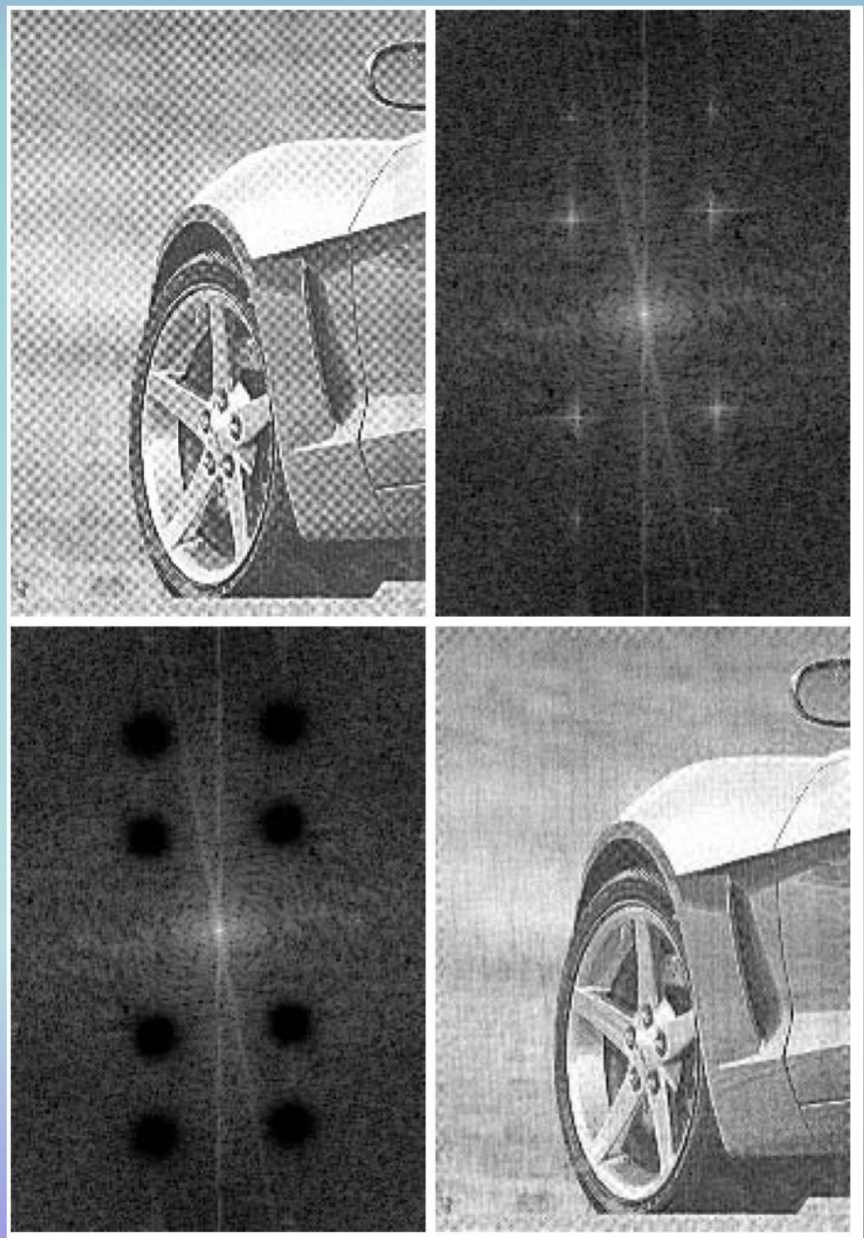




After applying
notch reject filter

CSE-BUET

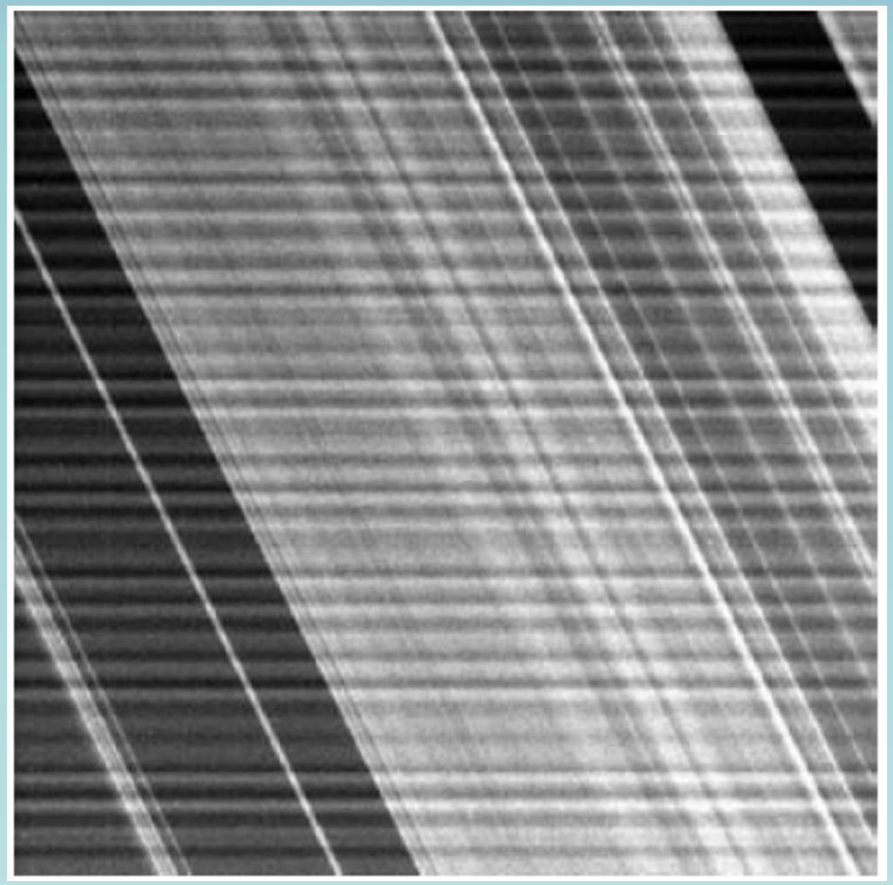
Example: Butterworth Notch Reject Filter



Enhanced Image

CSE-BUET

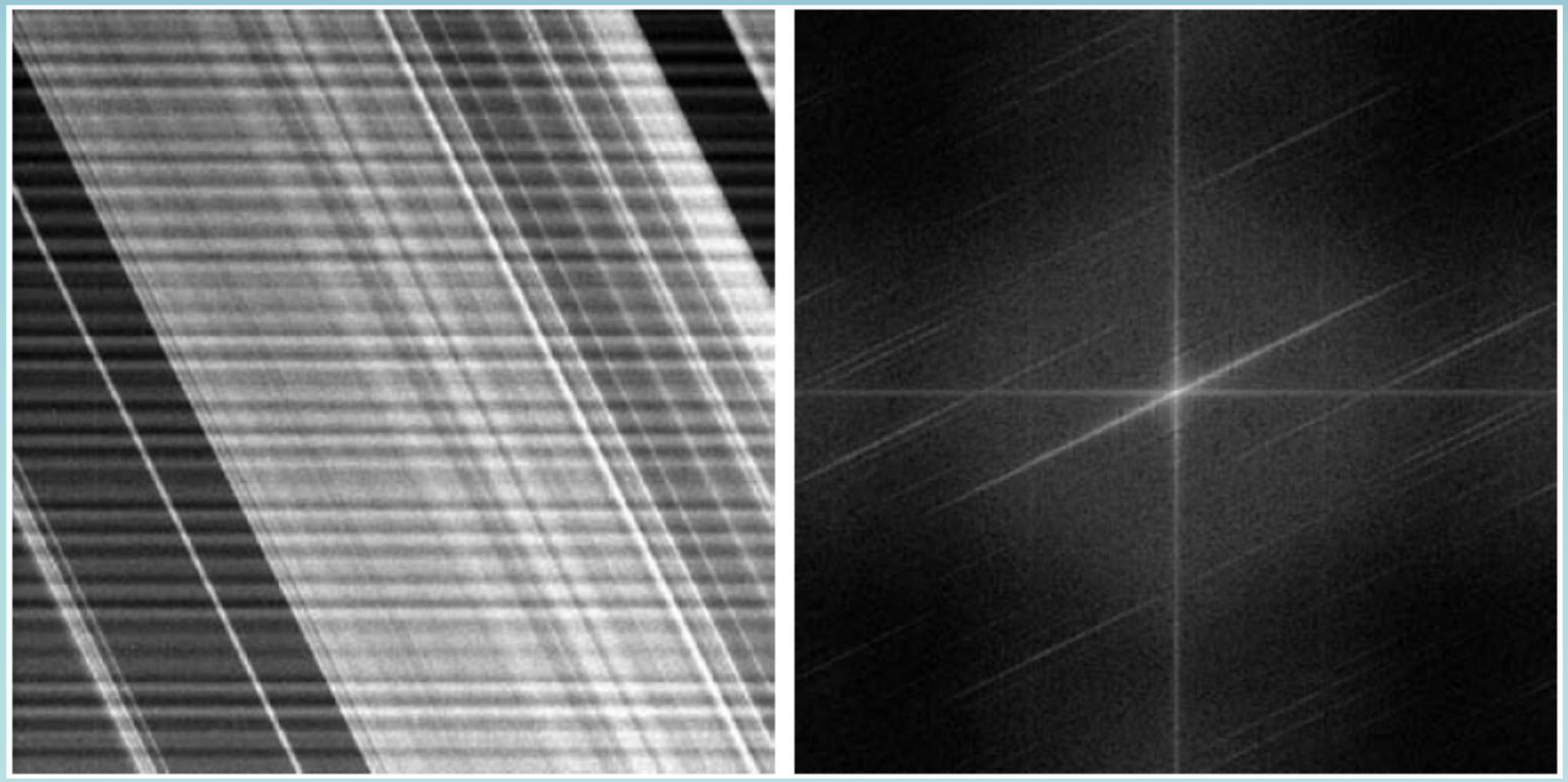
Example-2: Notch Reject Filter



- Image of Saturn Rings captured by ‘Cassini’ spacecraft
- Corrupted by sinusoidal AC signal while being transmitted

CSE-BUET

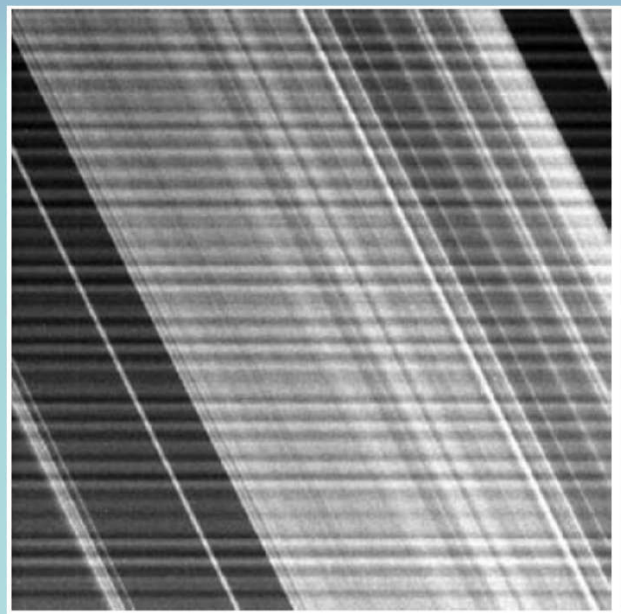
Example-2: Notch Reject Filter

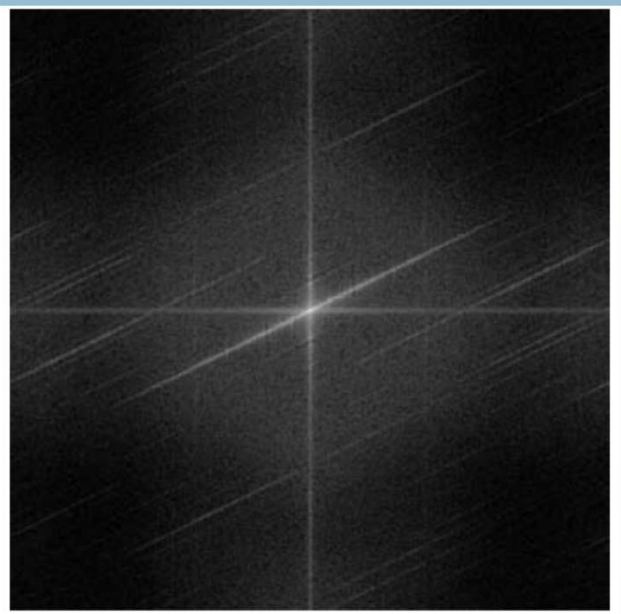


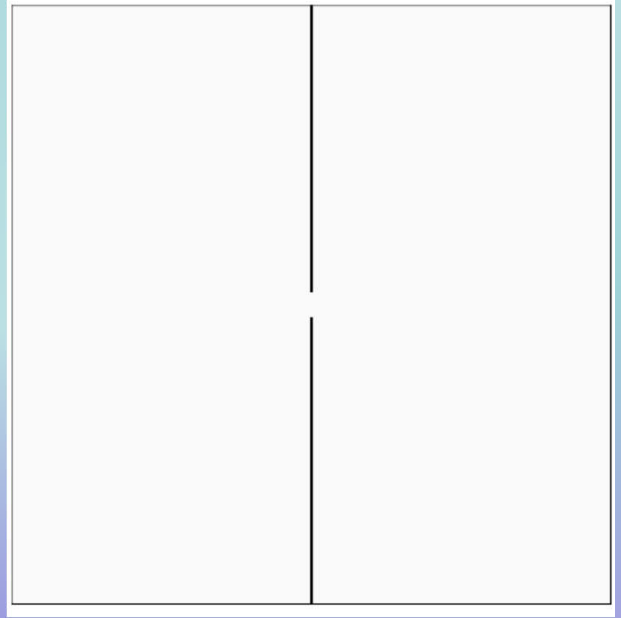
Burst of energy along the vertical axis

CSE-BUET

Example-2: Notch Reject Filter



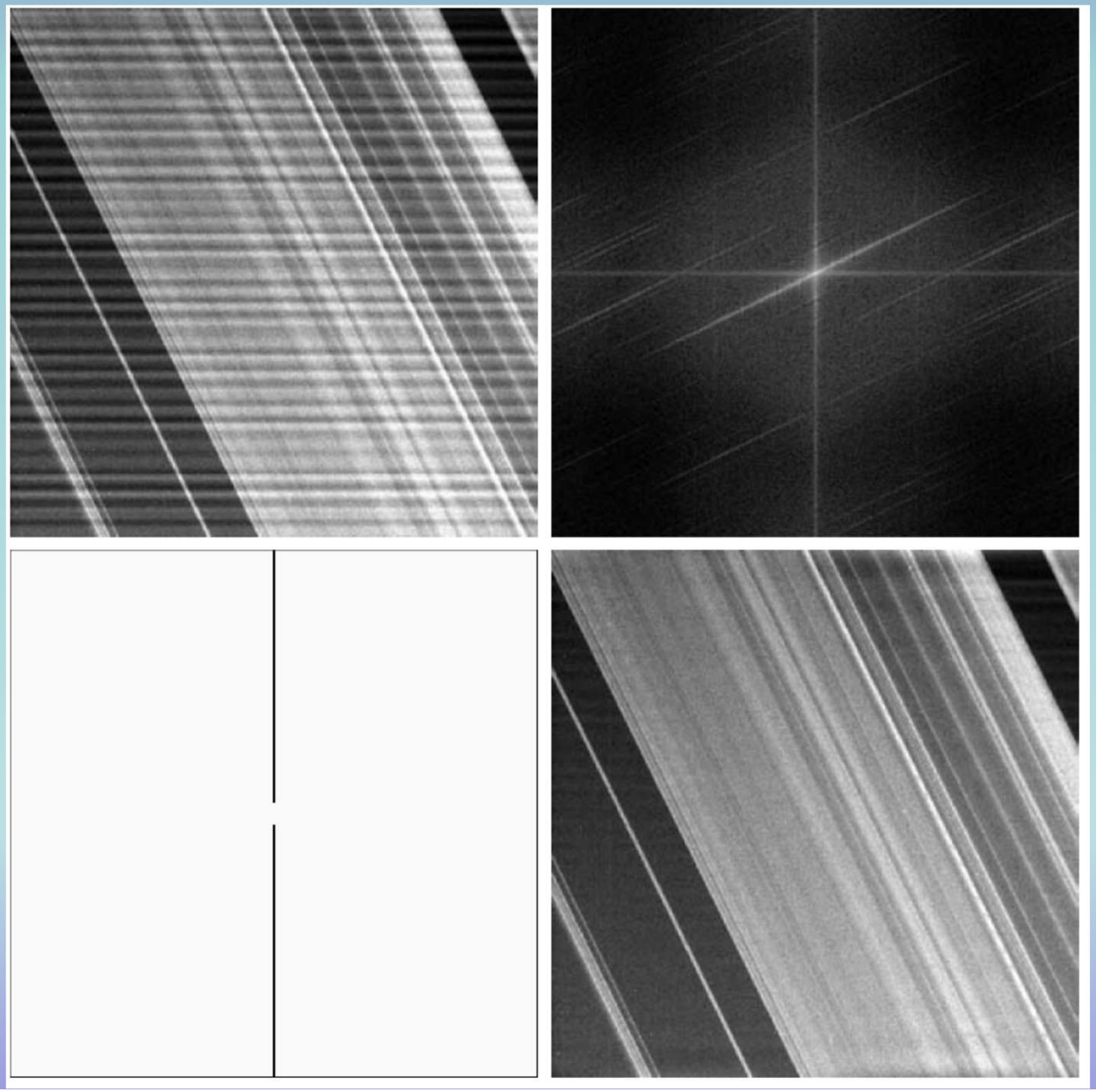




Narrow notch
rectangle filter

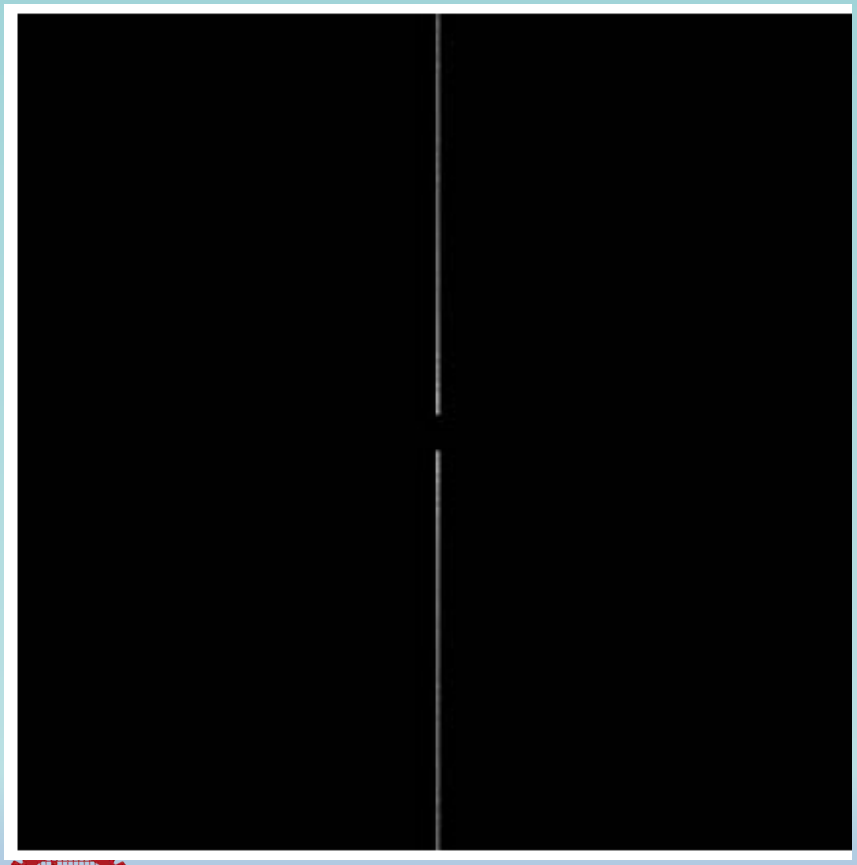
CSE-BUET

Example-2: Notch Reject Filter



CSE-BUET

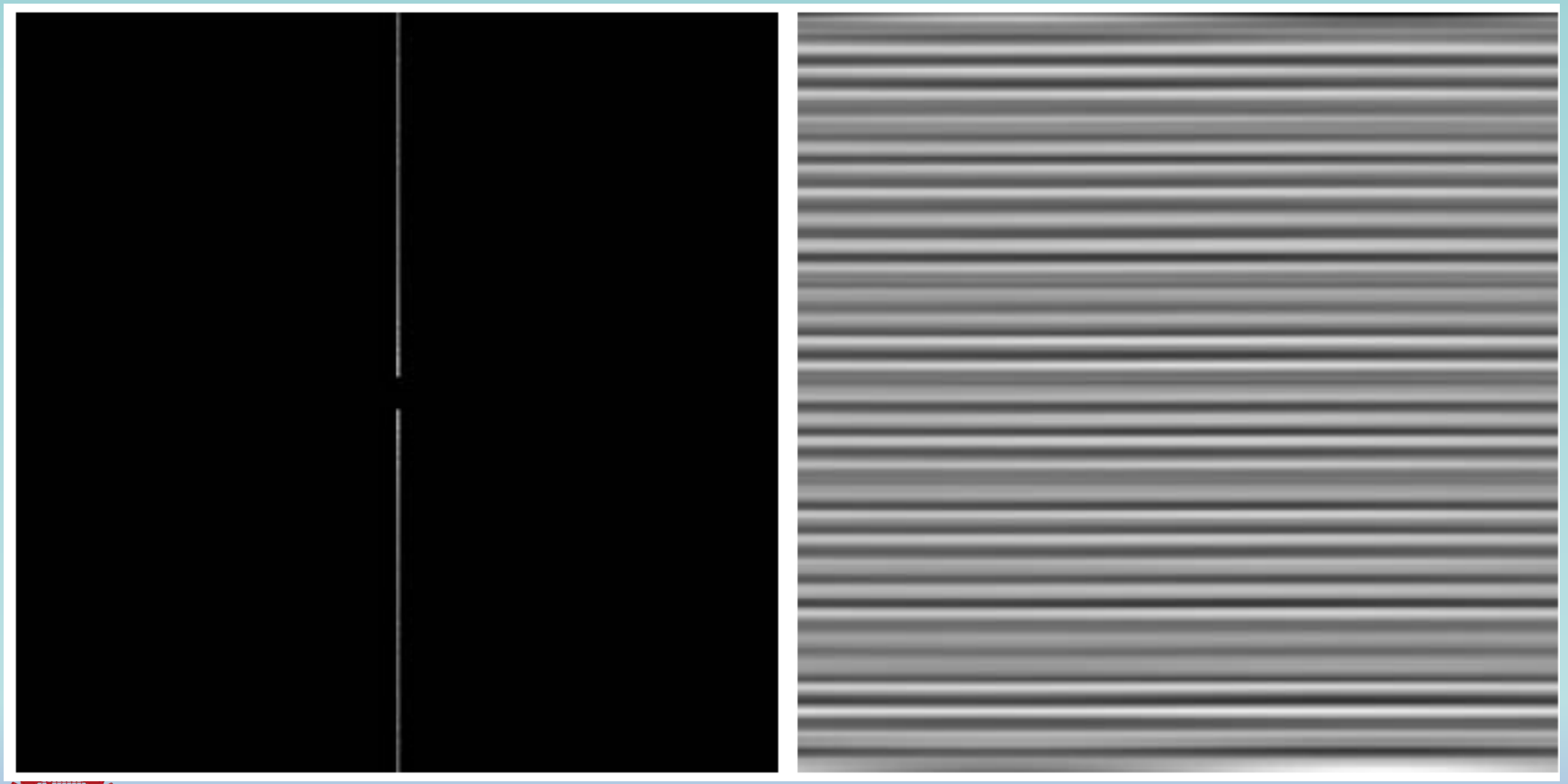
Example-2: Notch Pass Filter



Narrow notch pass filter to isolate the interference

CSE-BUET

Example-2: Notch Pass Filter



CSE-BUET

Image Compression

CSE-BUET

Why Images Are Compressed?

- Let a SD TV 2-hour color video sequence
  - each of size 720×480
  - frame rate: 30 frames/sec

CSE-BUET

Why Images Are Compressed?

- Let a SD TV 2-hour color video sequence
  - each of size 720×480
  - frame rate: 30 frames/sec

- For a single second, the amount of data to be accessed is

$$30\,frames \times (720 \times 480)\frac{pixels}{frame} \times 3\frac{bytes}{pixel} \approx 31MB$$

CSE-BUET

Why Images Are Compressed?

- For a 2-hour long SD TV video, the data is

$$31{,}104{,}000 \frac{bytes}{\text{sec}} \times (60)^2 \frac{\text{sec}}{hour} \times 2 \ hours$$

$$\approx 2.24 \times 10^{11} \ bytes$$

$$= 224 \text{ GB}$$

CSE-BUET

Why Images Are Compressed?

- *High Definition* (HD) TV color sequence
  - each of size 1920×1080 pixels

CSE-BUET

Why Images Are Compressed?

- *High Definition* (HD) TV color sequence
  - each of size 1920×1080 pixels
- Download Web Images
  - color images of size 128 × 128 through 56 Kbps to 12 Mbps connections
  - requires 7.0 to 0.03 seconds to download

Why Images Are Compressed?

- *High Definition* (HD) TV color sequence
  - each of size 1920×1080 pixels
- Download Web Images
  - color images of size 128 × 128 through 56 Kbps to 12 Mbps connections
  - requires 7.0 to 0.03 seconds to download
- 1 GB Flash memory and 8 Megapixel Digital camera
  - can store at most ~41 uncompressed images

Compression Ratio and Redundancy

Let b and b' be the number of bits before and after the compression

$$\text{Compression ratio, } C = \frac{b}{b'}$$

$$\text{Redundancy} = 1 - \frac{1}{C}$$

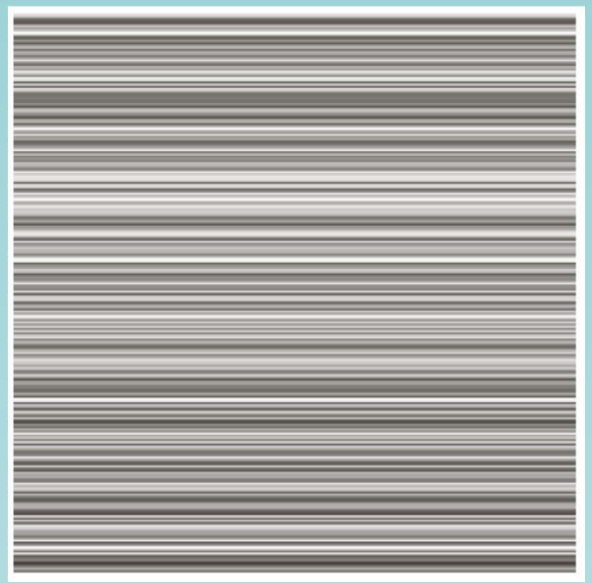
Different Types of Redundancies in Image

- *Coding redundancy*
- *Spatial redundancy*
- *Irrelevant information*

Coding Redundancy

- Code is system of symbols to represent info
- Codeword or sequence of symbols represents a piece of info
- The number of symbols reqd. is its length
- Try to minimize codeword length

Spatial Redundancy



- Intensity of each pixel is correlated to its neighbors
- Many info is *unnecessarily* replicated
- Try to minimize unnecessary data

CSE-BUET

Irrelevant Information

- Many images has info that are ignored by the *Human Visual System* (HVS)
- Try to remove these extraneous data

CSE-BUET

Coding Redundancy

Let r_k be gray level and a discrete random variable in $[0, L-1]$

we know, $p(r_k) = \dfrac{\text{No. of pixels with gray level } r_k}{MN}$, for $k = 0,1,2,\ldots,L-1$

CSE-BUET

Coding Redundancy

Let r_k be gray level and a discrete random variable in $[0, L-1]$

we know, $p(r_k) = \dfrac{\text{No. of pixels with gray level } r_k}{MN}$, for $k = 0,1,2,\ldots, L-1$

if $l(r_k)$ is the number of bits reqd. to represent r_k

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

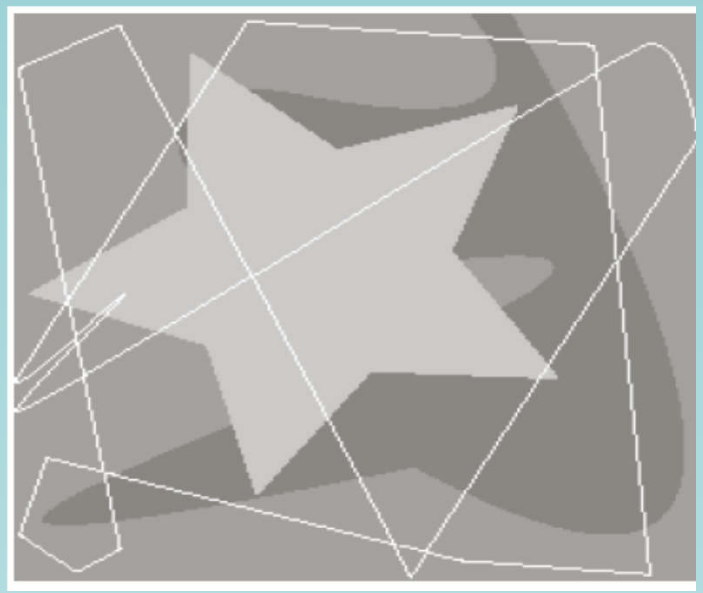
CSE-BUET

Coding Redundancy

Total bit requirment is $MNL_{avg} = MN \sum_{k=0}^{k=L-1} l(r_k)\, p(r_k)$

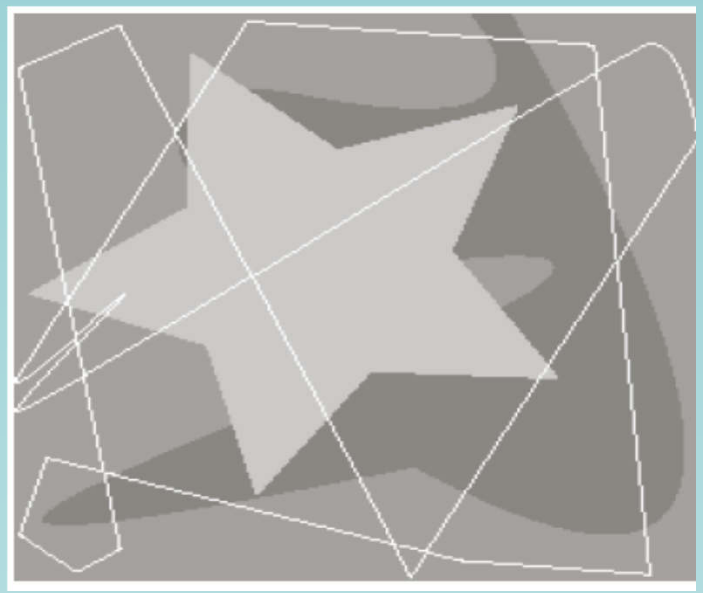
CSE-BUET

Coding Redundancy



r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$
$r_{87} = 87$	0.25	01010111	8
$r_{128} = 128$	0.47	10000000	8
$r_{186} = 186$	0.25	11000100	8
$r_{255} = 255$	0.03	11111111	8
r_k for $k \neq 87, 128, 186, 255$	0	—	8

Coding Redundancy



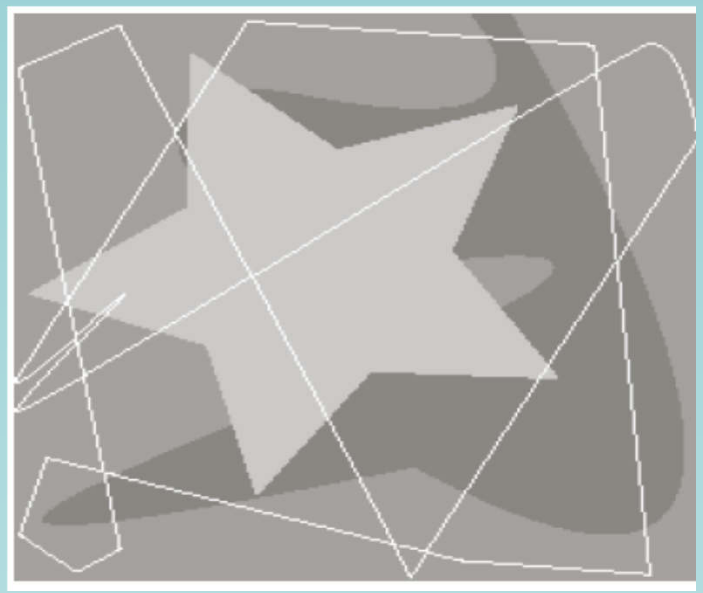
Total Bit required is

=256X256X8

without compression

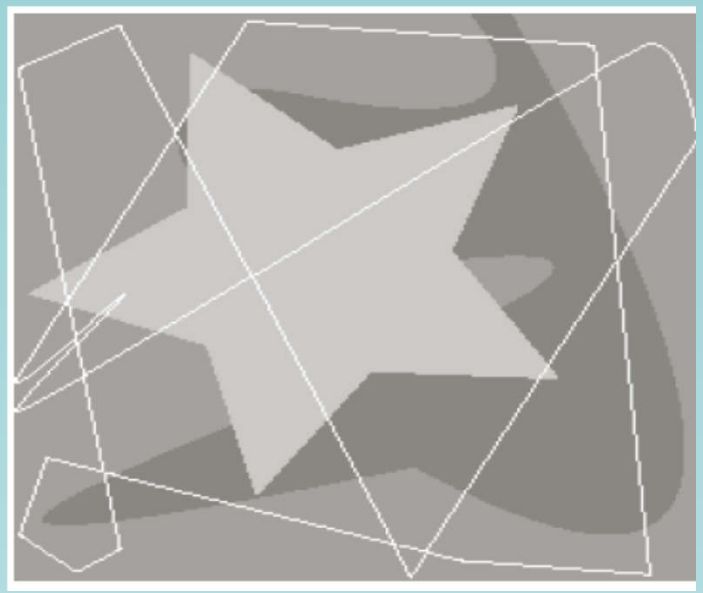
r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$
$r_{87} = 87$	0.25	01010111	8
$r_{128} = 128$	0.47	10000000	8
$r_{186} = 186$	0.25	11000100	8
$r_{255} = 255$	0.03	11111111	8
r_k for $k \neq 87, 128, 186, 255$	0	—	8

Coding Redundancy



r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

Coding Redundancy

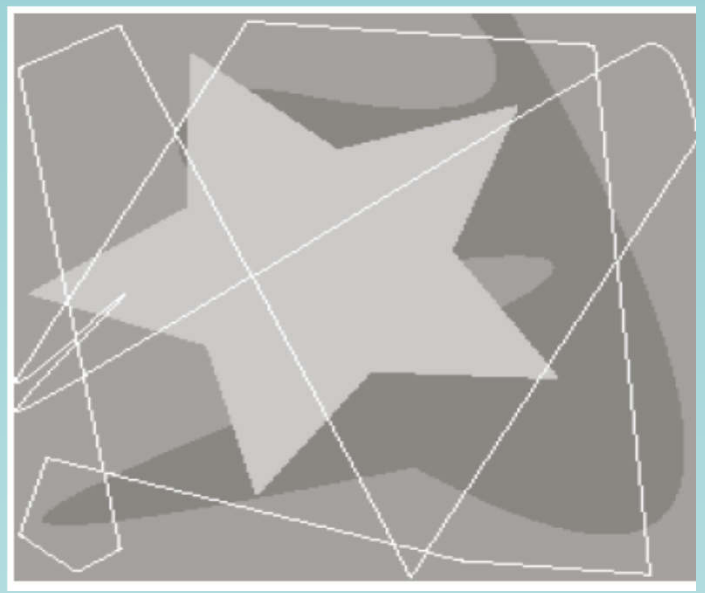


L_{avg}
=.25X2+.47X1+.25X3
+.03X3

=1.81 bits

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

Coding Redundancy

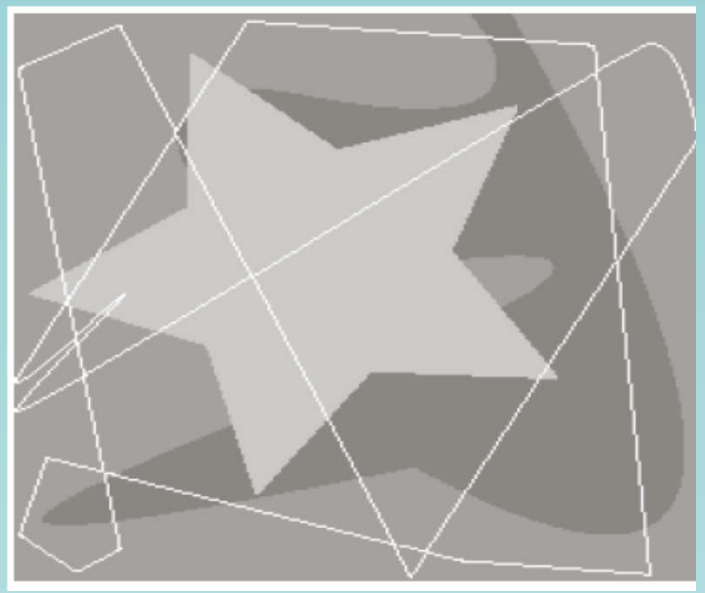


L_{avg}
=.25X2+.47X1+.25X3+.03X3=1.81 bits

Total
=256X256X1.81 bits

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

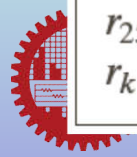
Coding Redundancy



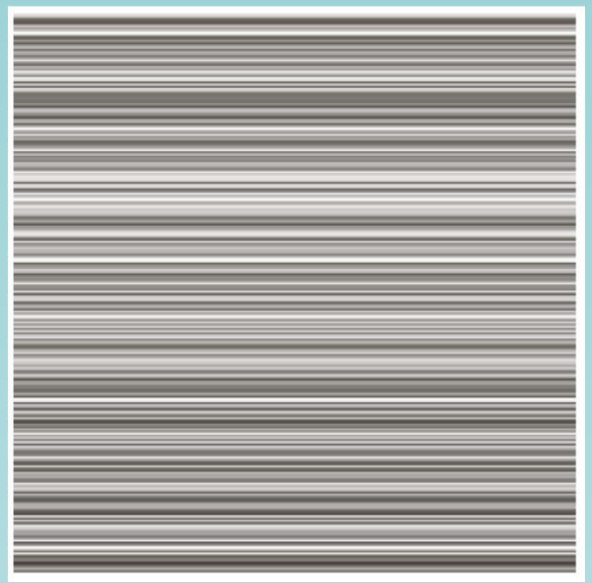
C =8/1.81=~4.42

R=1-1/C=.774

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
r_{87} = 87	0.25	01010111	8	01	2
r_{128} = 128	0.47	10000000	8	1	1
r_{186} = 186	0.25	11000100	8	000	3
r_{255} = 255	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0



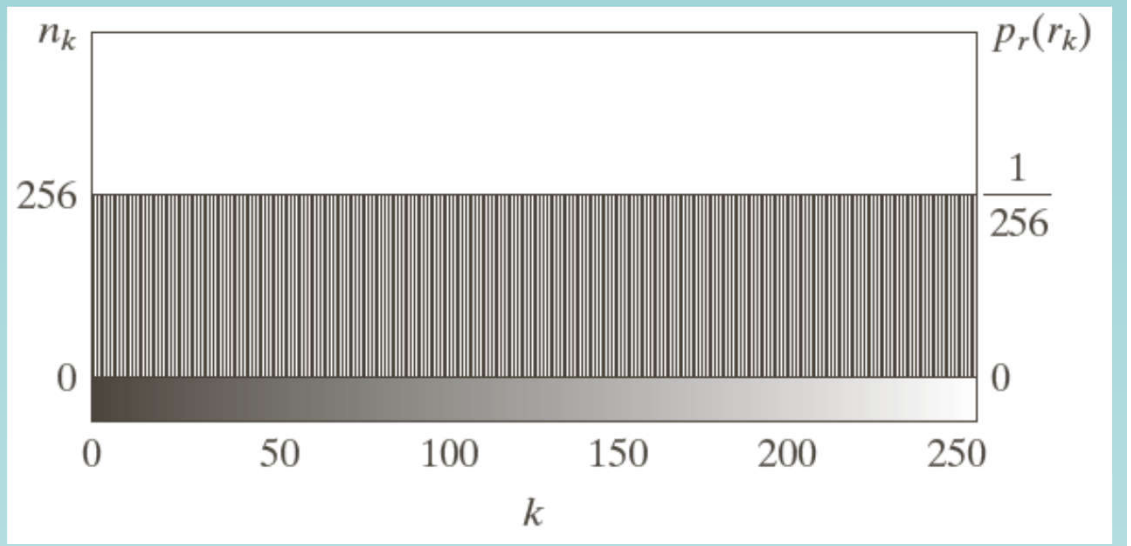
Spatial and Temporal Redundancy



- Histogram is equiprobable

CSE-BUET

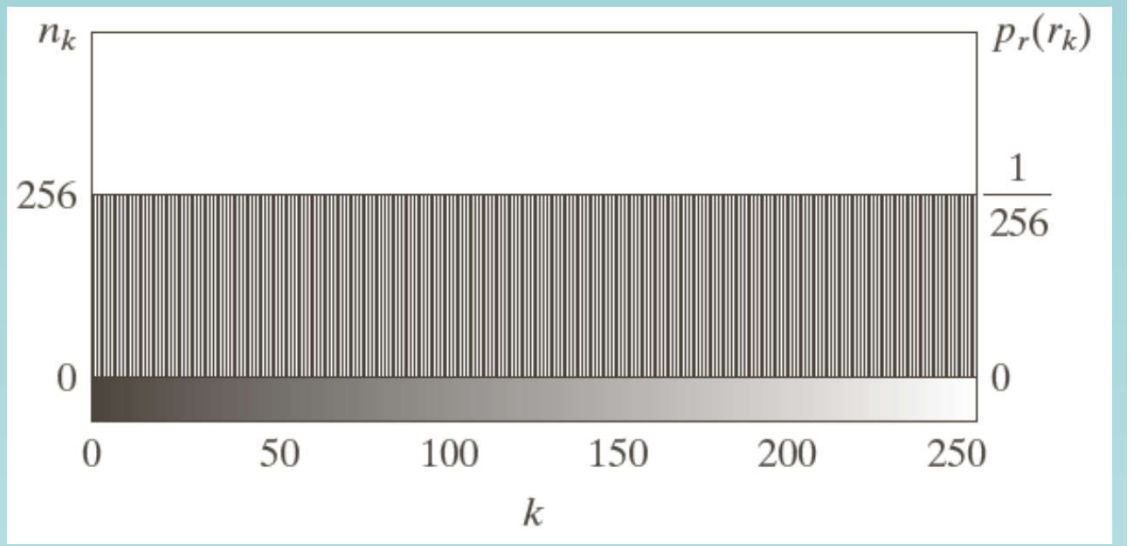
Spatial and Temporal Redundancy



- Histogram is equiprobable

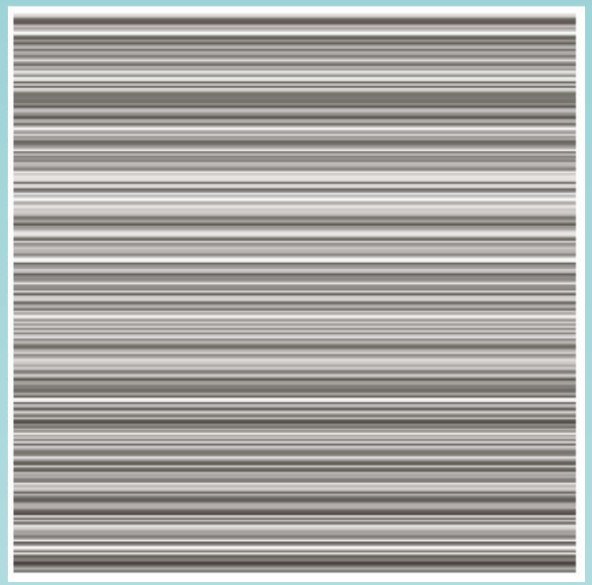
CSE-BUET

Spatial and Temporal Redundancy



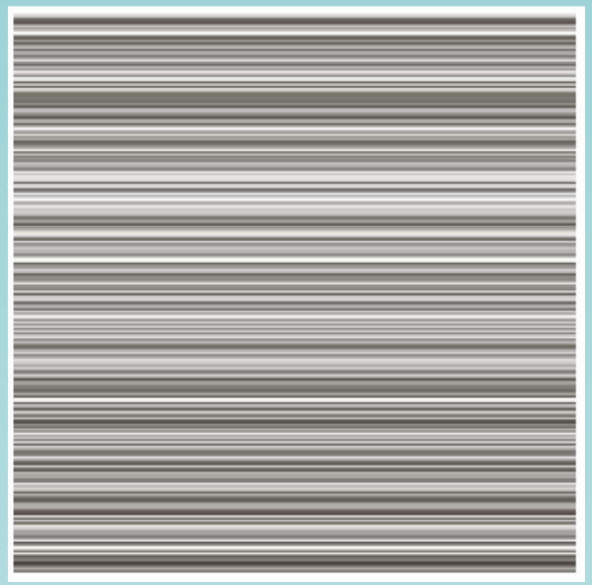
- Histogram is equiprobable
- The image cannot be compressed using variable length coding

Spatial and Temporal Redundancy



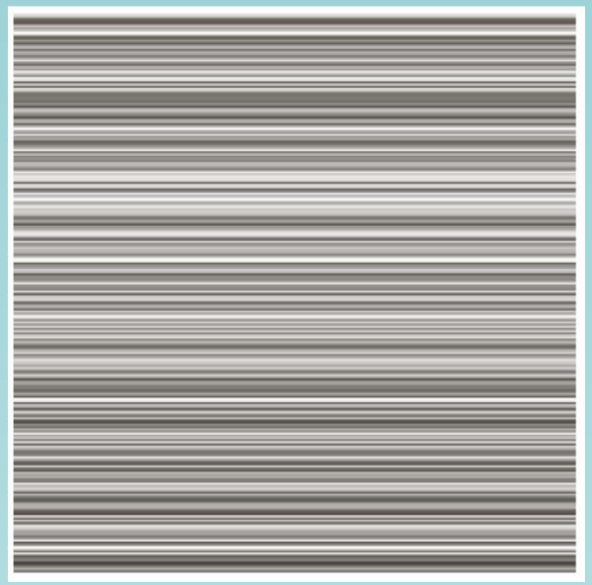
- Histogram is equiprobable
- Vertically, pixels are independent
- Horizontally, they are maximally correlated

Spatial and Temporal Redundancy



- Vertically, pixels are independent
- Horizontally, they are maximally correlated
- The image can be represented as a sequence of run length pair : *intensity value* and *rum of pixels with that intensity*

Spatial and Temporal Redundancy



Original Bit required

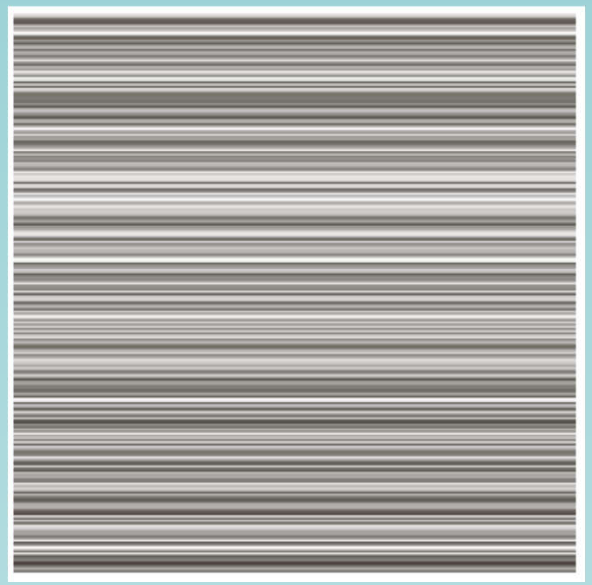
=256X256X8

In Run length

=256X8+256X8

- Vertically, pixels are independent
- Horizontally, they are maximally correlated
- The image can be represented as a sequence of run length pair : *intensity value* and *rum of pixels with that intensity*

Spatial and Temporal Redundancy



Original Bit required

=256X256X8

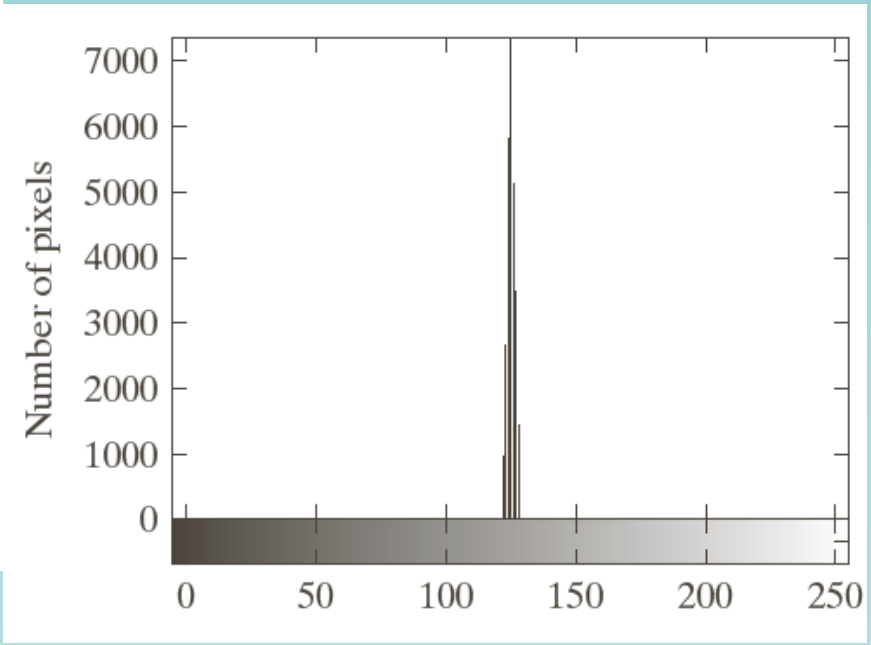
In Run length

=256X8+256X8

C =128:1

- Vertically, pixels are independent
- Horizontally, they are maximally correlated
- The image can be represented as a sequence of run length pair : *intensity value* and *rum of pixels with that intensity*

Irrelevant Information



Can be represented using a single avg. gray level

CSE-BUET

Irrelevant Information

Can be represented using a single avg. gray level

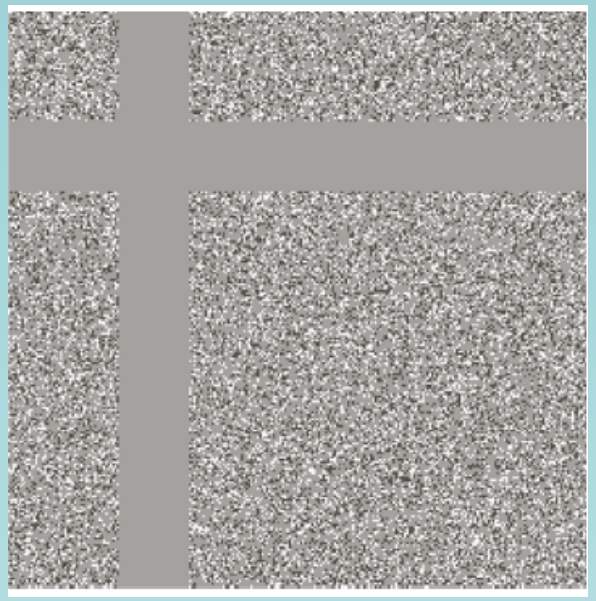
C =256X256X8:8

=65536:1

Irrelevant Information

- However, this coarse quantization may remove useful *invisible* information

Irrelevant Information



- However, this coarse quantization may remove useful *invisible* information

CSE-BUET

Redundancies We Tried to Remove

- *Coding redundancy*
- *Spatial redundancy*
- *Irrelevant information*

CSE-BUET

Redundancies We Tried to Remove

- *Coding redundancy*
- *Spatial redundancy*
- *Irrelevant information*

How many bits do we really need to represent image-information?

CSE-BUET

Redundancies We Tried to Remove

- *Coding redundancy*
- *Spatial redundancy*
- *Irrelevant information*

How many bits do we really need to represent image-information?

Information theory: does is it have any answer?

CSE-BUET

Information Theory Review

A random event E with probability $P(E)$
carries $I(E)$ units of information,

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

CSE-BUET

Information Theory Review

A random event E with probability $P(E)$
carries $I(E)$ units of information,

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

- *Higher the uncertainty, higher the information content*

Information Theory Review

If we have

a source of random events from a discrete set of events $\{a_1, a_2, a_3, \ldots, a_J\}$ with probabilities, $P(a_1), P(a_2), P(a_3), \ldots, P(a_J)$

then the average information per event or the entropy of the source,

$$H = \sum_{j=1}^{J} -P(a_j)\log P(a_j)$$

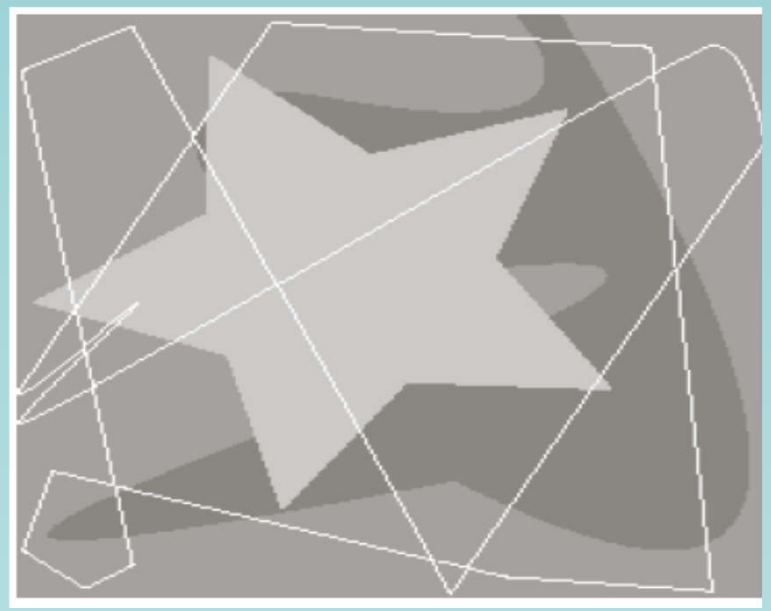
Information Theory Review

If we consider the pixel intensities as random events, then the intensity histogram is the approximation of the probabilities

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

CSE-BUET

Information Theory Review



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

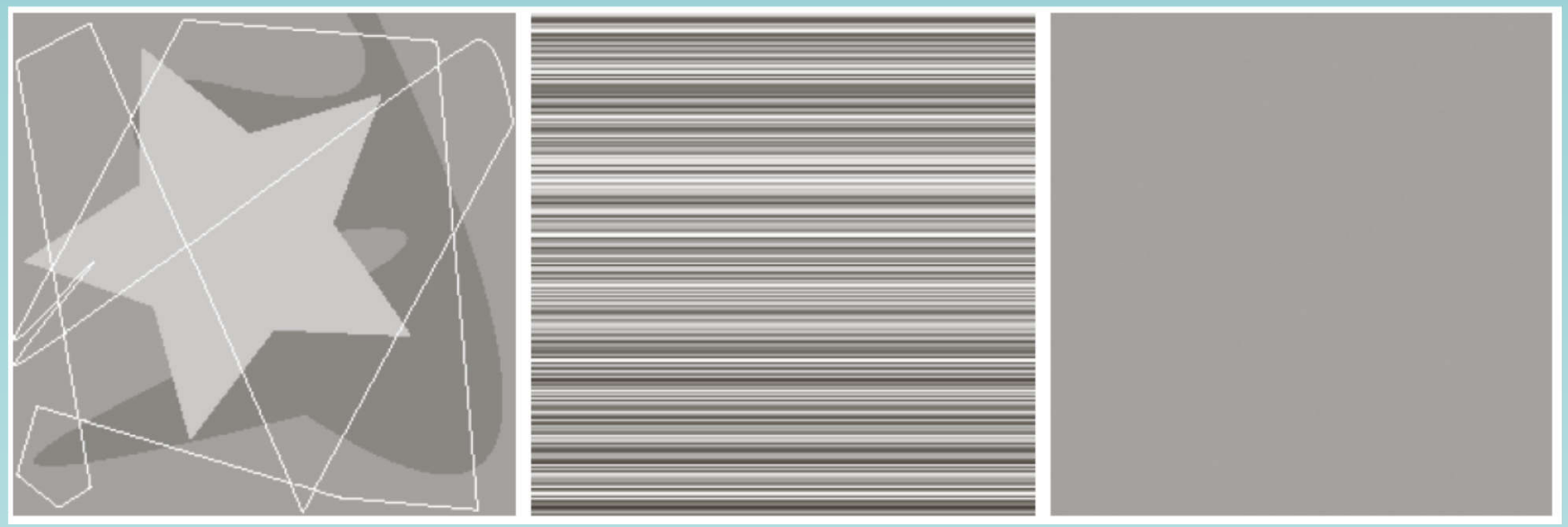
$$= -[0.25\log_2 0.25 + 0.47\log_2 0.47 + 0.25\log_2 0.25 + 0.03\log_2 0.03]$$

$$\approx -[0.25(-2) + 0.47(-1.09) + 0.25(-2) + 0.03(-5.06)]$$

$$\approx 1.6614 \text{ bits/pixel}$$

r_k	$P_r(r_k)$
$r_{87} = 87$	0.25
$r_{128} = 128$	0.47
$r_{186} = 186$	0.25
$r_{255} = 255$	0.03
r_k for $k \neq 87, 128, 186, 255$	0

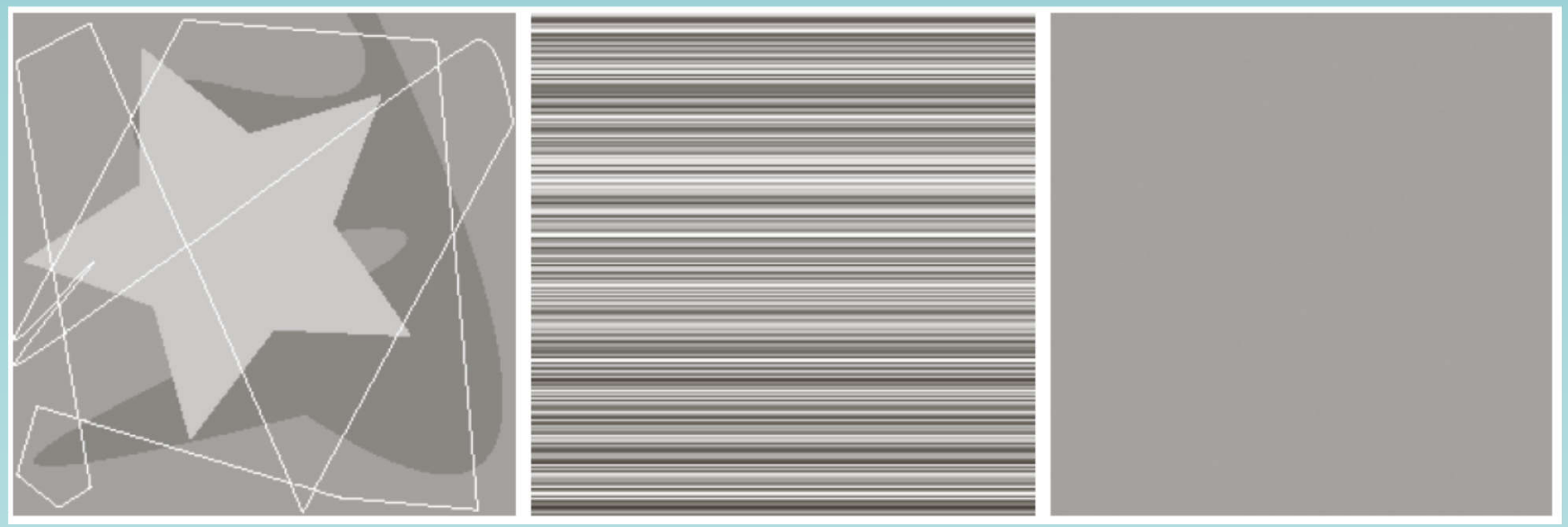
Information Theory Review



H=1.6614 *H*=8 *H*=1.566

CSE-BUET

Information Theory Review

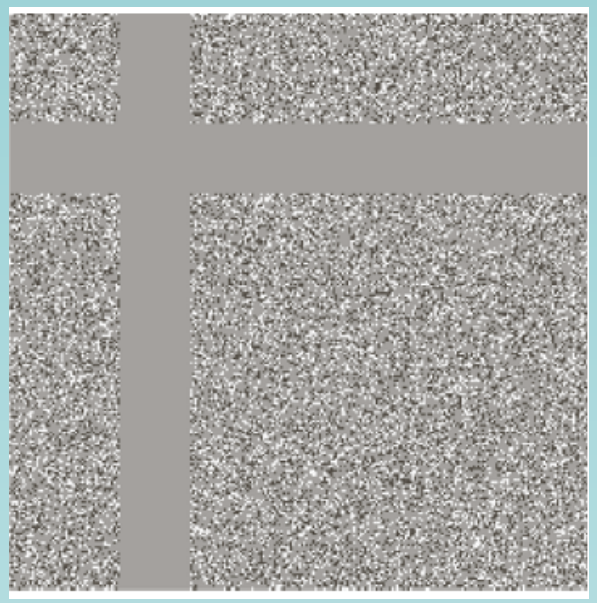


H=1.6614 *H*=8 *H*=1.566

- Visual perception and entropy (information) are far from compliance

CSE-BUET

Image Fidelity Criteria: *assessment of loss*



removal of irrelevant information removes useful *invisible* information, too

CSE-BUET

Image Fidelity Criteria

information is lost in many other compression techniques : *Lossy compression*

CSE-BUET

Image Fidelity Criteria

Assessment of loss or infidelity:

- Objective fidelity: *quantizable*
- Subjective fidelity: *person dependant*

Image Fidelity Criteria: Objective Fidelity

Original image, $f(x, y)$

Compressed, then reconstructed image, $f'(x, y)$

$$e(x, y) = \hat{f}(x, y) - f(x, y)$$

CSE-BUET

Image Fidelity Criteria: Objective Fidelity

Original image, $f(x, y)$

Compressed, then reconstructed image, $f'(x, y)$

$$\text{total error} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]$$

CSE-BUET

Image Fidelity Criteria: Objective Fidelity

Original image, $f(x, y)$

Compressed, then reconstructed image, $f'(x, y)$

$$\text{rms error, } e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2 \right]^{1/2}$$

CSE-BUET

Image Fidelity Criteria: Objective Fidelity

mean square *signal-to-noise ratio* (SNR_{ms}) is given by,

$$\text{SNR}_{\text{ms}} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)-f(x,y)\right]^2}$$

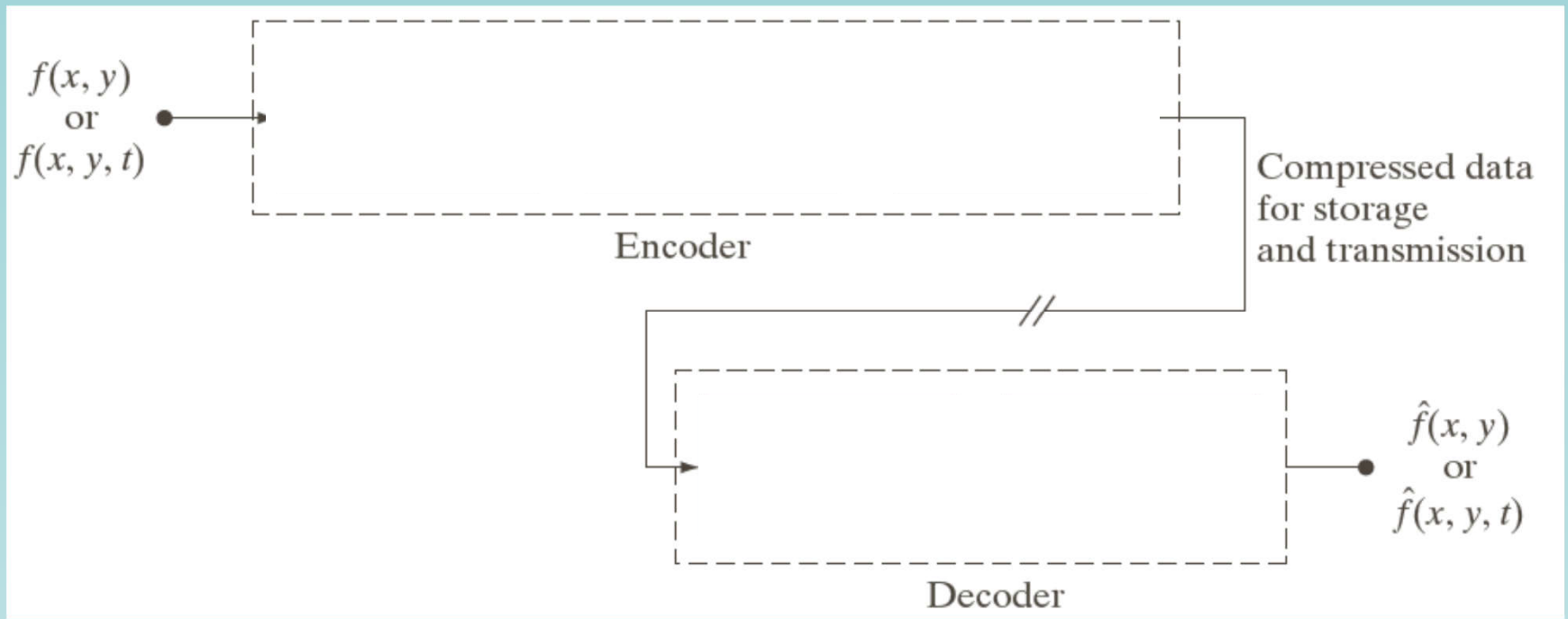
CSE-BUET

Image Fidelity Criteria: Example

Image Fidelity Criteria: Subjective Fidelity

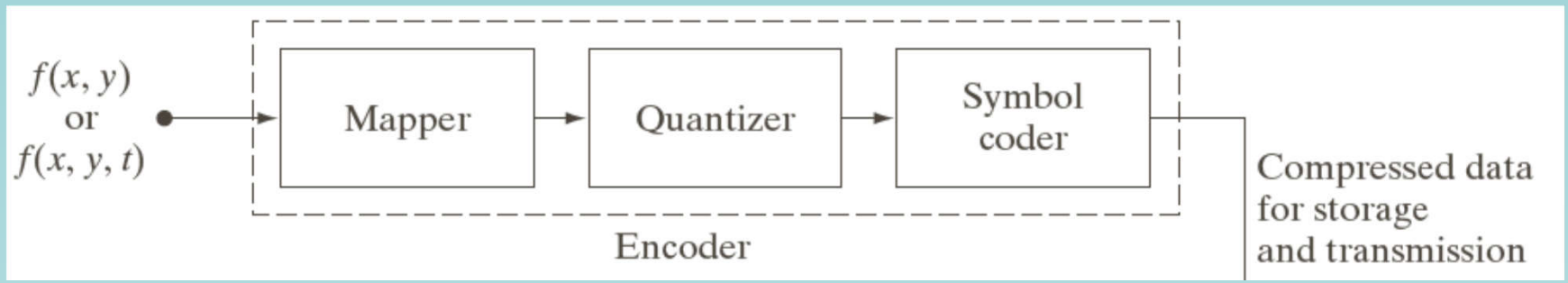
- Humans are the ultimate judges
- Uses subjective ranking, either numeric or categorical
- Example:
  - {-3, -2, -1, 0, 1, 2, 3}
  - {*much worse*, *worse*, *slightly worse*, *the same*, *slightly better*, *better, much better*}

Image Compression Models



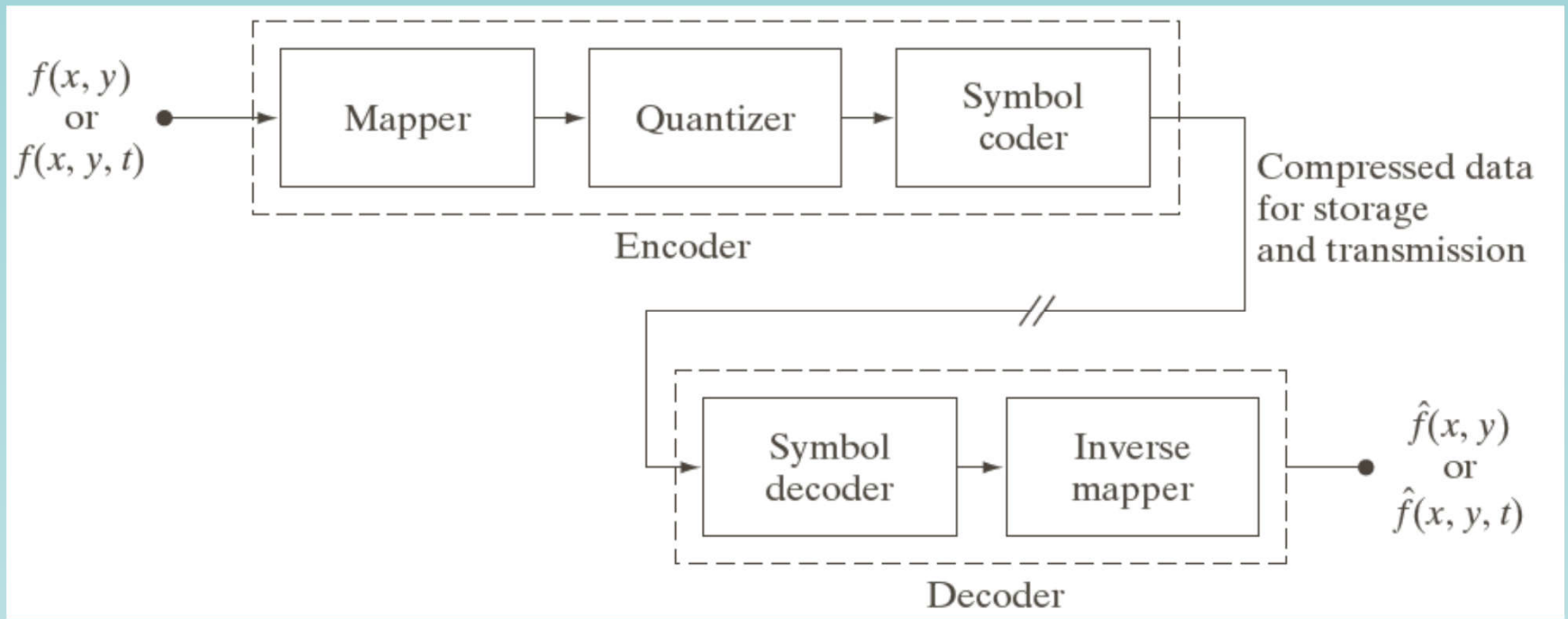
CSE-BUET

Image Compression Models



- Mapper:
  - Transforms $f(x, y)$ to reduce spatial and temporal redundancy
  - Reversible process
  - Ex: run length coding
- Quantizer:
  - Removes irrelevant information
  - Irreversible process
  - Reduces accuracy
- Symbol coder:
  - Generates fixed or variable length code for *Quantizer* outputs
  - Reversible process

Image Compression Models



CSE-BUET

Different Error Free Compression Techniques

- Variable length coding
  - Huffman coding
  - Arithmetic coding
- LZW coding
- Bit plane coding
  - Constant area coding
  - Run length coding

CSE-BUET

Loss Less (Error Free) Compression

Huffman Coding:

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

Loss Less (Error Free) Compression

Huffman Coding:

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

Loss Less (Error Free) Compression

Huffman Coding:

Original source		
Symbol	Probability	1
a_2	0.4	0.4
a_6	0.3	0.3
a_1	0.1	0.1
a_4	0.1	0.1
a_3	0.06	0.1
a_5	0.04	

Loss Less (Error Free) Compression

Huffman Coding:

Original source		Source Reduction	
Symbol	Probability	1	2
a_2	0.4	0.4	0.4
a_6	0.3	0.3	0.3
a_1	0.1	0.1	0.2
a_4	0.1	0.1	0.1
a_3	0.06	0.1	
a_5	0.04		

Loss Less (Error Free) Compression

Huffman Coding:

Original source		Source Reduction		
Symbol	Probability	1	2	3
a_2	0.4	0.4	0.4	0.4
a_6	0.3	0.3	0.3	0.3
a_1	0.1	0.1	0.2	0.3
a_4	0.1	0.1	0.1	
a_3	0.06	0.1		
a_5	0.04			

Loss Less (Error Free) Compression

Huffman Coding:

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

CSE-BUET

Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

CSE-BUET

Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

CSE-BUET

Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction			
Symbol	Probability	Code	1	2	3	4
a_2	0.4	1	0.4 1	0.4 1	0.4 1	0.6 0
a_6	0.3	00	0.3 00	0.3 00	0.3 00	0.4 1
a_1	0.1	011	0.1 011	0.2 010	0.3 01	
a_4	0.1	0100	0.1 0100	0.1 011		
a_3	0.06	01010	0.1 0101			
a_5	0.04	01011				

Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

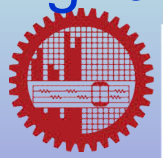
Loss Less (Error Free) Compression

Huffman Code assignment

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Avg. bit requirement/symbol:

$$L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2 \text{ bits/symbol}$$



Loss Less (Error Free) Compression

Properties of Huffman Coding:

- *codes one symbol at a time*
- *Block code*
  - each symbol is mapped to a block of code
- *Instantaneously decodable*
  - does not need to foresee the future codes while decoding
- *Uniquely decodable*
  - any string of code symbol can be decoded in only one way

Loss Less (Error Free) Compression

Arithmetic Coding:

- *Non block coding*
  - no one-to-one correspondence between symbol and code
  - a string of symbol is mapped to a single number

Loss Less (Error Free) Compression

Arithmetic Coding:

- *Maintains an interval between* [0 1] based on the probabilities of the symbol

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

EOF symbol

CSE-BUET

Loss Less (Error Free) Compression

Arithmetic Coding:

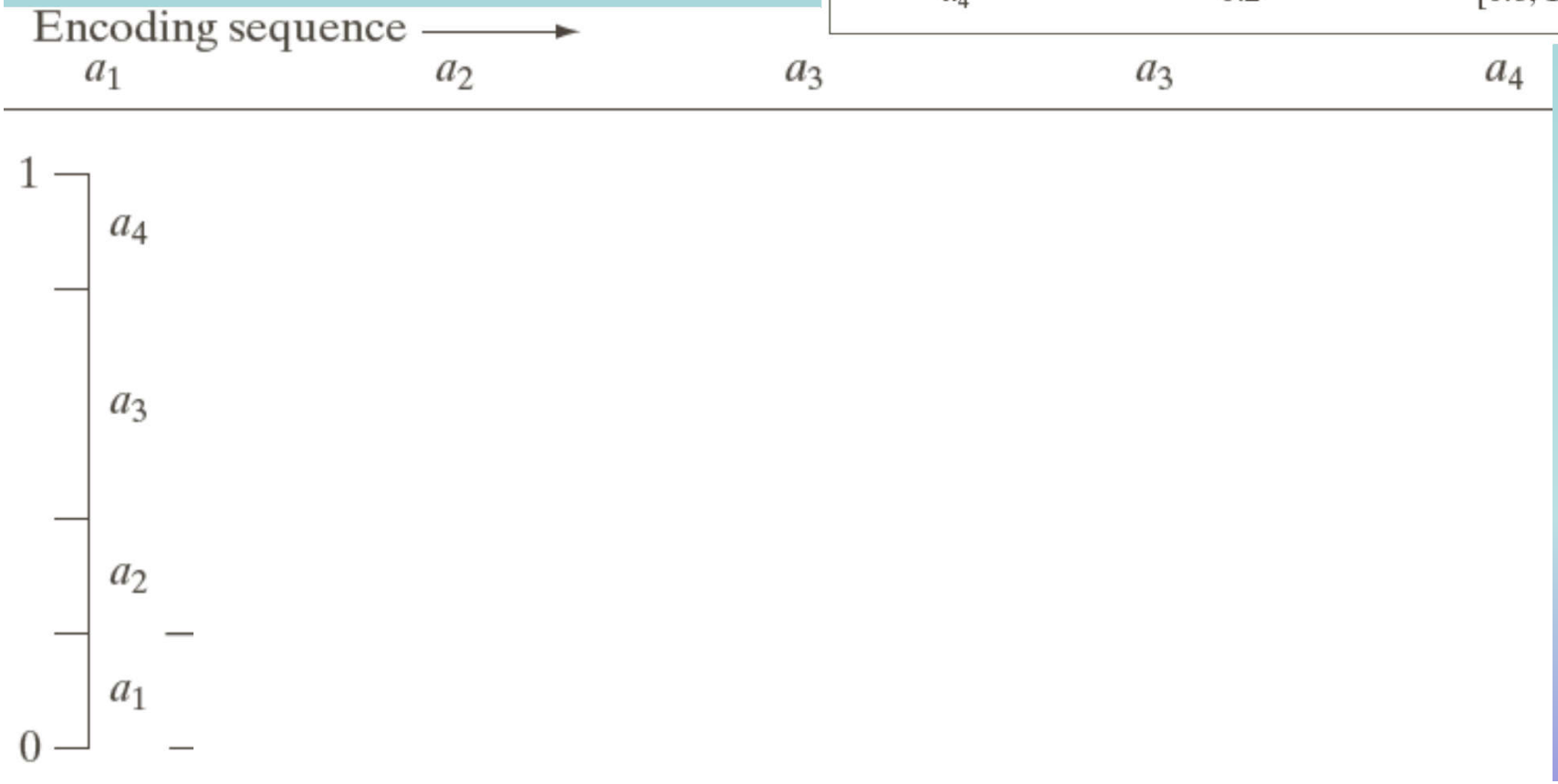
Let we have to code the sequence

$$a_1 a_2 a_3 a_3 a_4$$

Loss Less (Error Free) Compression

Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



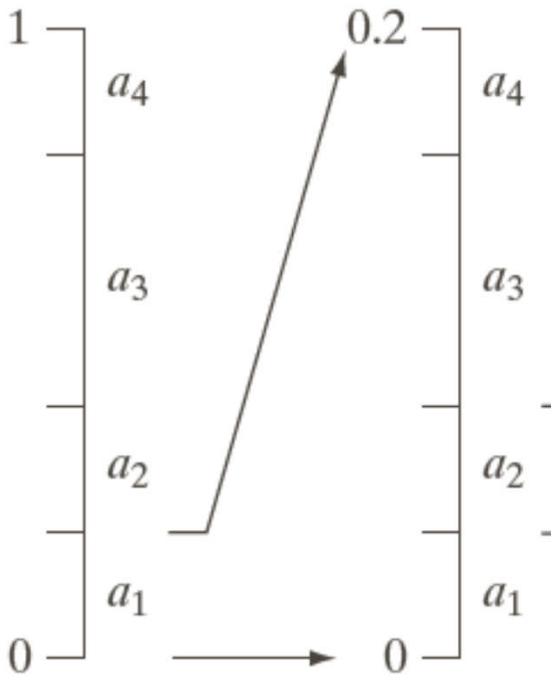
Loss Less (Error Free) Compression

Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Encoding sequence →

a_1 a_2 a_3 a_3 a_4



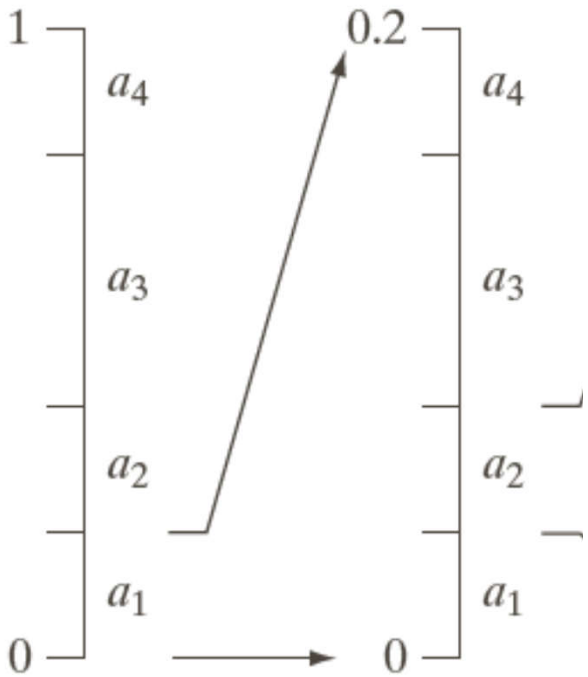
Loss Less (Error Free) Compression

Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Encoding sequence →

a_1 a_2 a_3 a_3 a_4



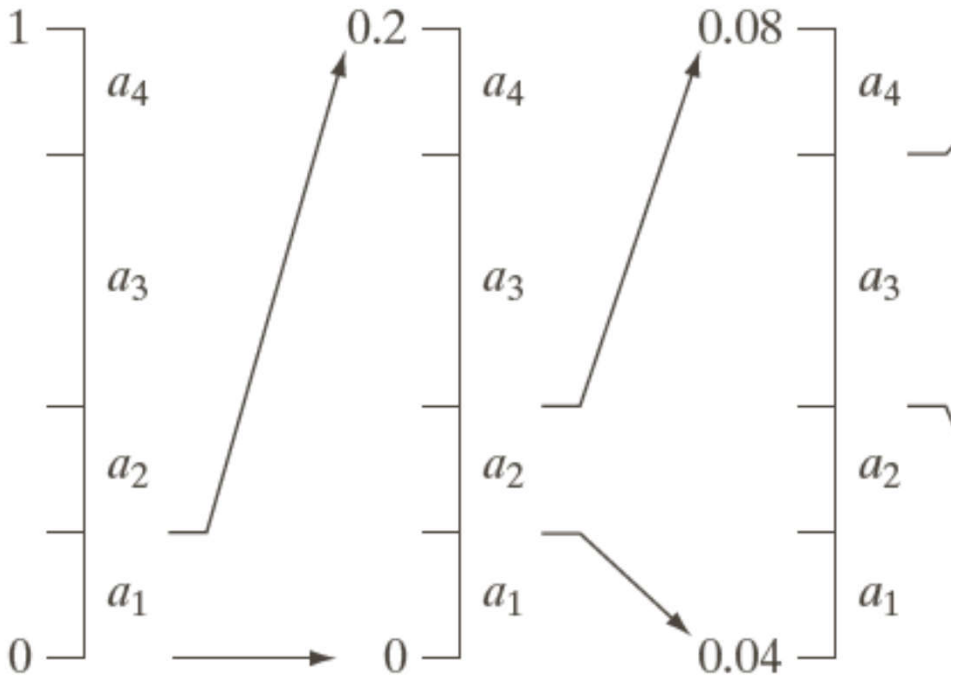
Loss Less (Error Free) Compression

Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Encoding sequence →

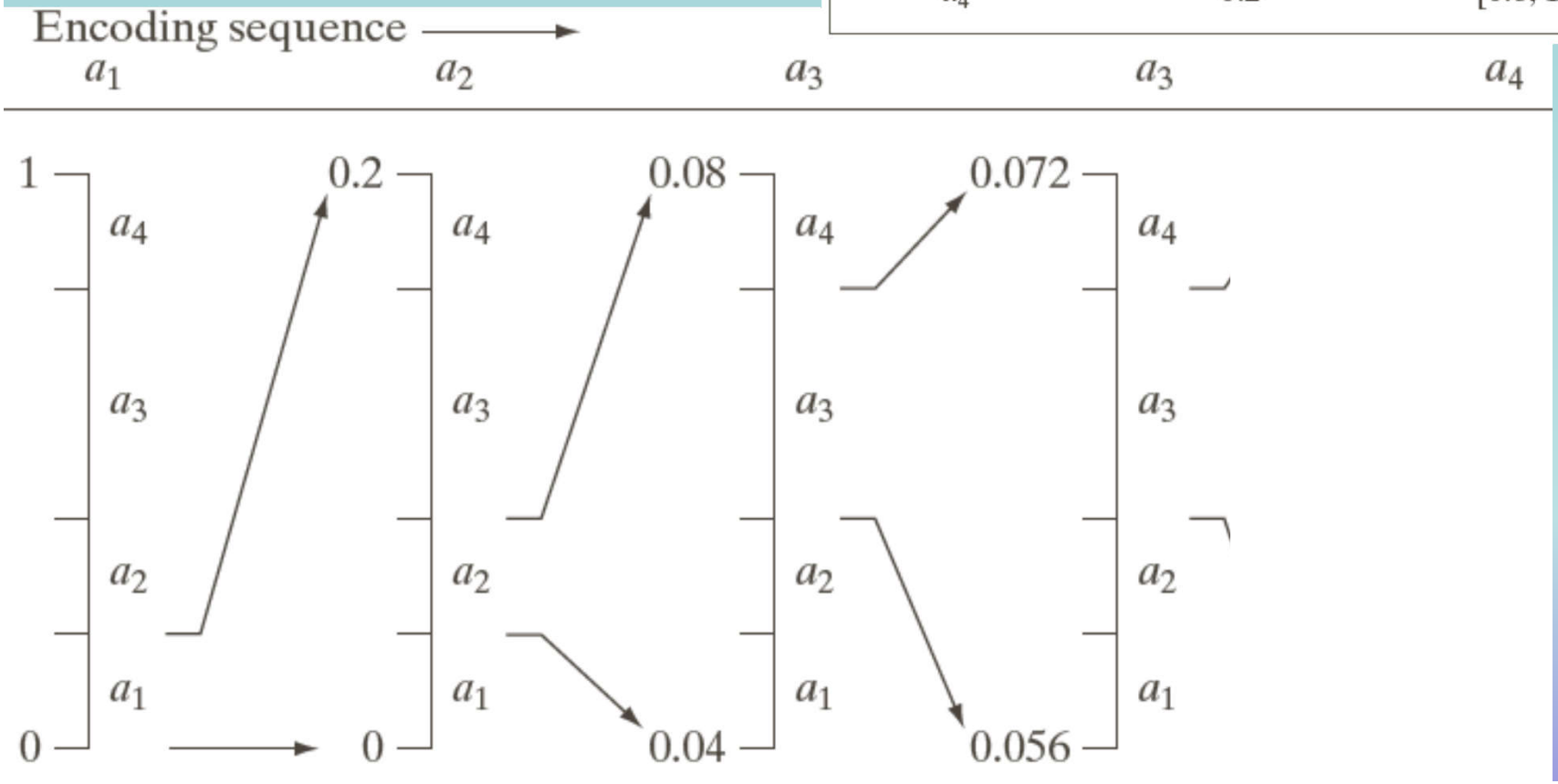
a_1 a_2 a_3 a_3 a_4



Loss Less (Error Free) Compression

Arithmetic Coding:

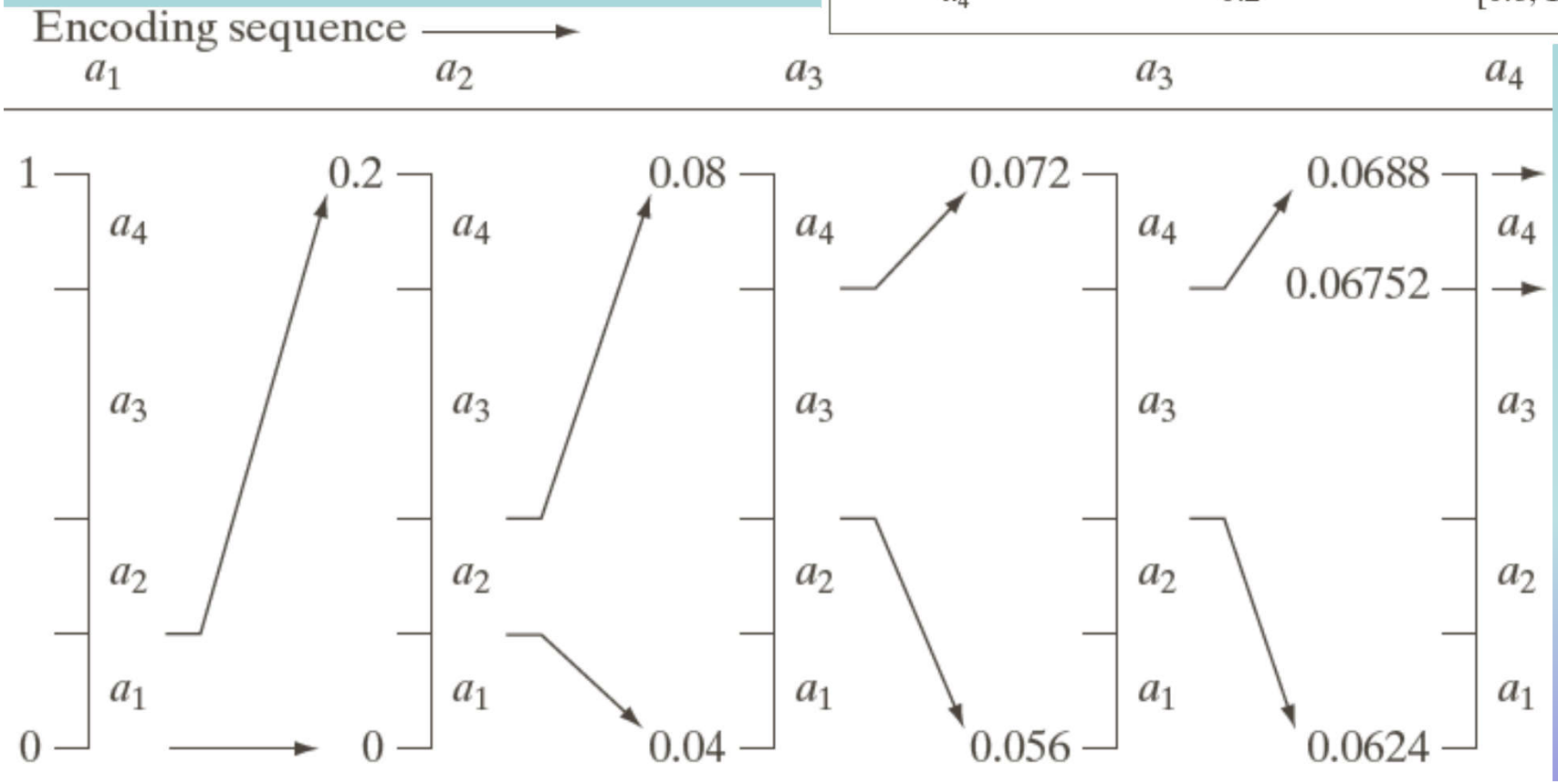
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Loss Less (Error Free) Compression

Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Loss Less (Error Free) Compression

Arithmetic Coding:

The arithmetic code for the sequence

$$a_1 a_2 a_3 a_3 a_4$$

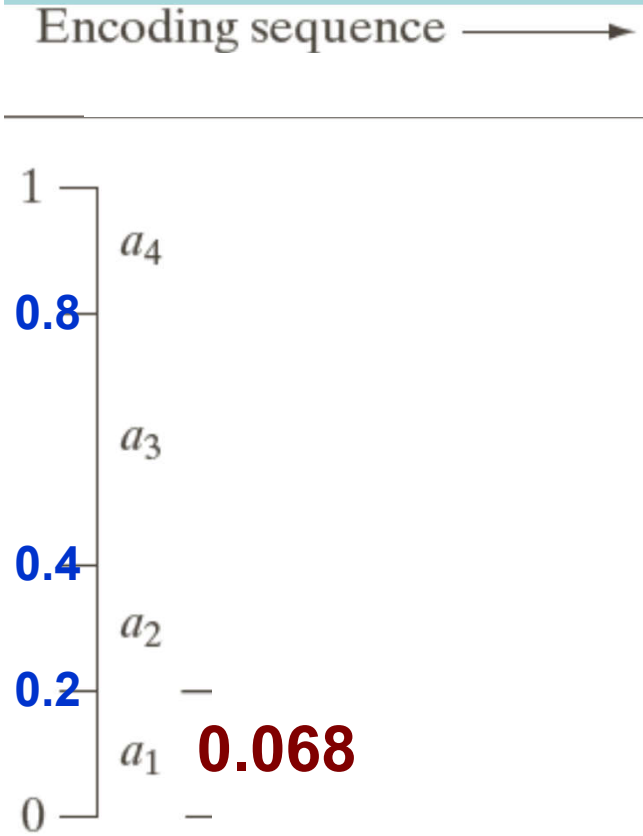
is **0.068**

CSE-BUET

Loss Less (Error Free) Compression

Decoding for Arithmetic Coding:

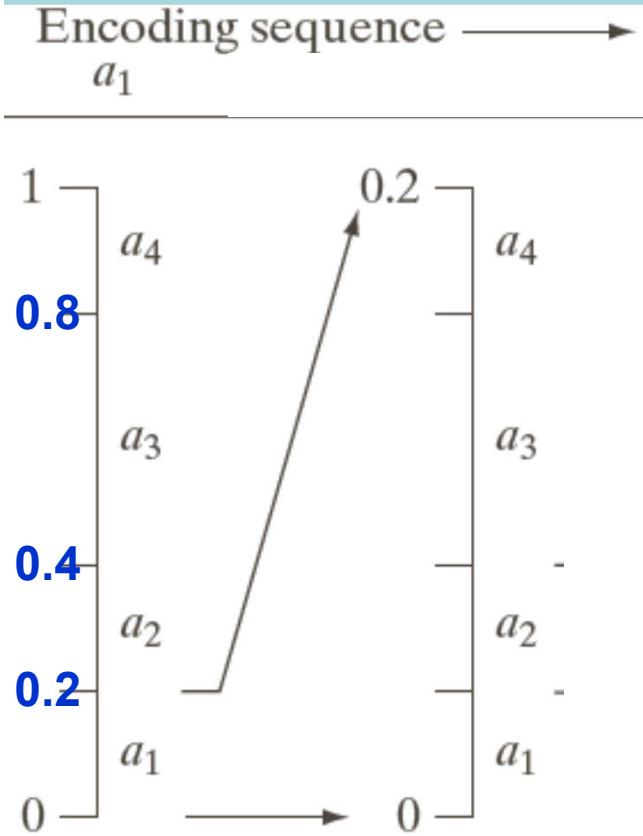
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Loss Less (Error Free) Compression

Decoding for Arithmetic Coding:

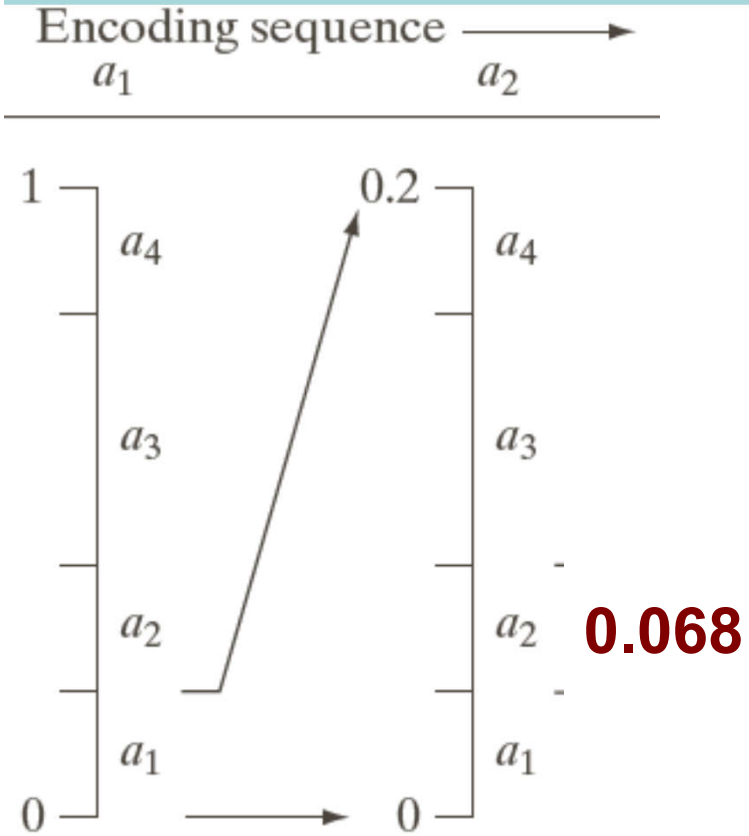
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Loss Less (Error Free) Compression

Decoding for Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

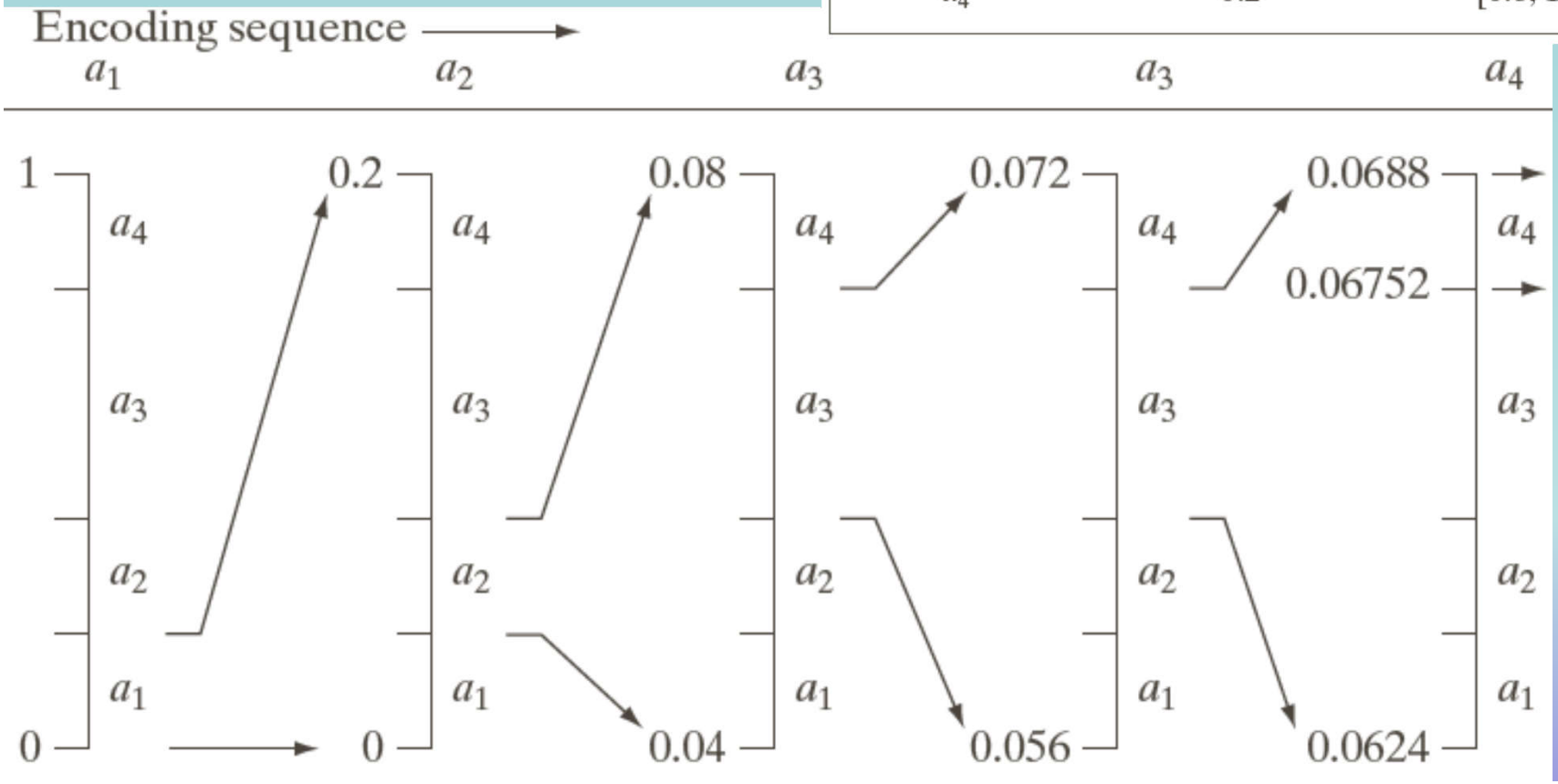


0.068

Loss Less (Error Free) Compression

Decoding for Arithmetic Coding:

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



0.068

Lempel-Ziv-Welch (LZW) Coding

- uses fixed length code for variable length sequence of source symbols
- requires no prior knowledge
- used in
  - TIFF
  - GIF
  - PDF

Lempel-Ziv-Welch (LZW) Coding

Coding Technique

- Generate a codebook/dictionary
  - first 256 entries are assigned to gray levels 0,1,2,..,255.
  - New gray level sequences not already in the dictionary are assigned to a new entry.
  - Example: 255-255 can be assigned to entry# 256, the address following the locations reserved for gray levels 0 to 255.

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

- Example

Consider the following 4 x 4 8 bit image

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Initial Dictionary

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			

Sequence= Concatenate(‘ ’, ‘39’) = 39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39				

Sequence= Concatenate(‘ ’, ‘39’) = 39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39			

Sequence= Concatenate('39', '39') = 39-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
39	39	39	256	39-39

Address of '39'

Sequence= Concatenate('39', '39') = 39-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39				

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126			

Sequence= Concatenate('39', '126') = 39-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126

Address of '39'

Sequence= Concatenate('39', '126') = 39-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126				

Sequence= Concatenate('39', '126') = 39-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126				

Sequence= Concatenate('126', '126') = 126-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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				

Sequence= Concatenate('126', '39') = 126-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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			

Sequence= Concatenate('39', '39') = 39-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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			

Sequence= Concatenate('39', '39') = 39-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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				

Sequence= Concatenate('39', '39') = 39-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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				

Sequence= Concatenate('39-39', '126') = 39-39-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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				

Sequence= Concatenate('126', '126') = 126-126

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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				

Sequence= Concatenate('126-126', '39') = 126-126-39

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

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

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

CSE-BUET

Lempel-Ziv-Welch (LZW) Coding

Encoding of

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

is

39 39 126 126 256 258 260 259 257 126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	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			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

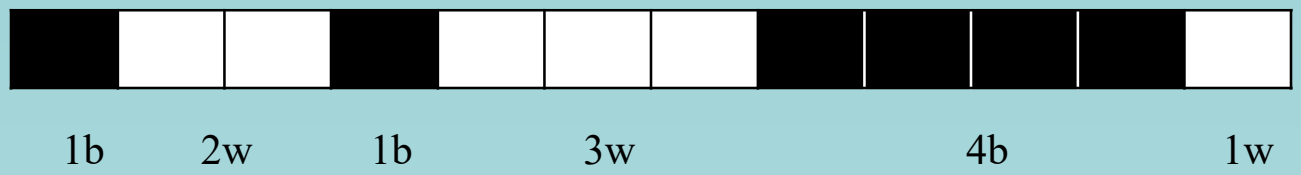
CSE-BUET

Compression Ratio using LZW Coding

- Sequence 39-39-126 is replaced by 260

- In other words 3X8=24 bits are replaced by 9 bits
- Original image = 16X8 bits = 128 bits
- Compressed Image = 10X9 bits = 90 bits
- Ratio = 128:90 = 1.42:1

Run Length Encoding (RLE)



- Uses run length pairs
  - (0, 1), (1, 2), (0, 1), (1, 3), …
- Eliminates spatial redundancies
- However, small runs results in expansion instead of compression

Run Length Encoding in BMP

- Uses a combination of *encoded* and *absolute* mode
- Either mode can appear anywhere in the image
- Encoded mode:
  - 2 byte RLE
  - First byte: run length (*must be non-zero*)
  - Second byte: gray/color index

CSE-BUET

Run Length Encoding in BMP

- Uses a combination of *encoded* and *absolute* mode
- Either mode can appear anywhere in the image
- Absolute mode:
  - 2 byte RLE
  - First byte: 0
  - Second byte: as follows

Second Byte Value	Condition
0	End of line
1	End of image
2	Move to a new position
3–255	Specify pixels individually

CSE-BUET

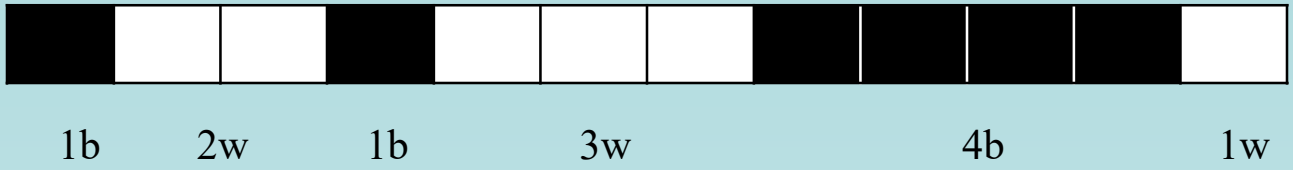
Run Length Encoding in Binary Image

- Effective
  - Adjacent pixels are more likely to be same
- More efficient encoding
  - can be represented as a sequence of runs only

CSE-BUET

Run Length Encoding in Binary Image

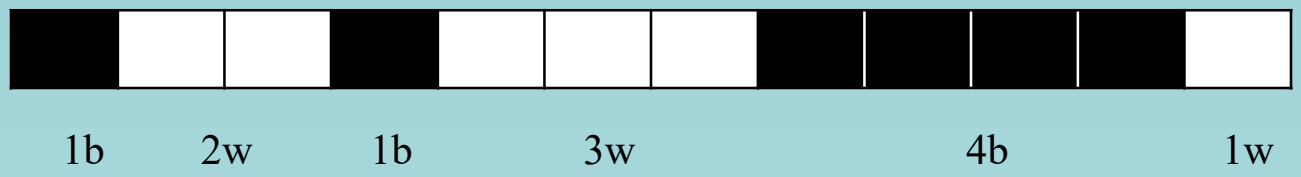
- Effective
  - Adjacent pixels are more likely to be same
- More efficient encoding
  - can be represented as a sequence of runs only
  - Example:



  - Its representation: 121341

CSE-BUET

Run Length Encoding in Binary Image



- Representation: 121341
- Fix a way to determine the run values:
  - Specify the value of first run
  - Assume each row begins with a *white* run!!

Additional Encoding on Run Length Encoding

- Runs lengths: 1 2 1 3 4 1
- Run lengths can be compressed using variable lengths encoding
  - White or black run-lengths can be encoded separately

CSE-BUET

Symbol Based Coding

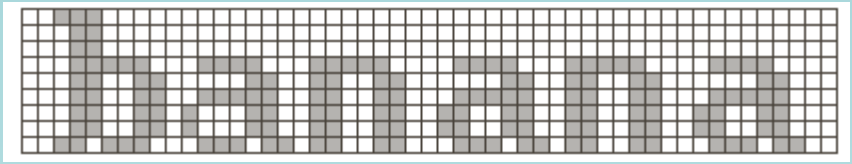
- Construct a dictionary of symbols
  - Identifies a collection of frequently occurring sub-images/symbols/tokens
- Represent an image using symbols
- An image is encoded as

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

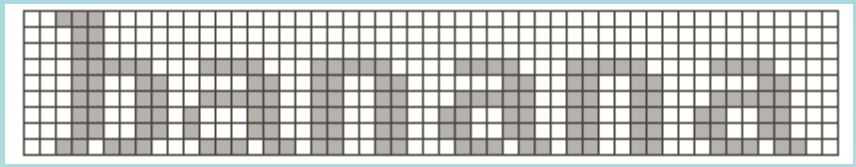
where, (x_i, y_i) is the location where token t_i will be placed in the image

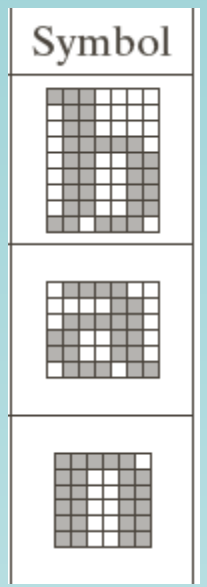
CSE-BUET

Symbol Based Coding

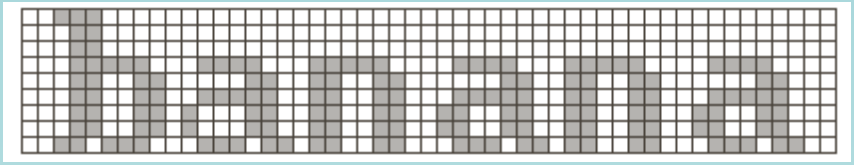


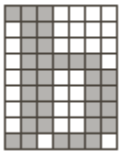
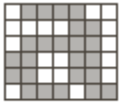
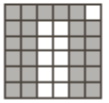
Symbol Based Coding





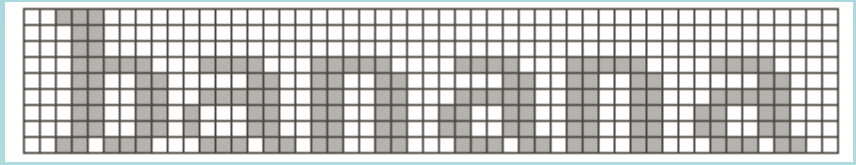
Symbol Based Coding

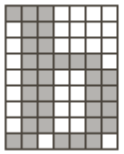
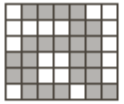
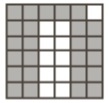


Token	Symbol
0	
1	
2	

CSE-BUET

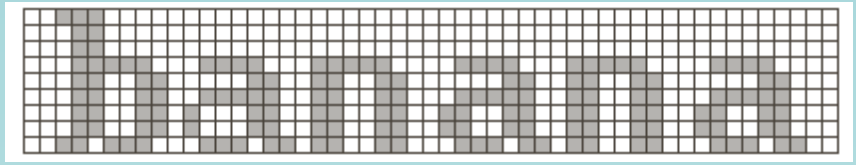
Symbol Based Coding

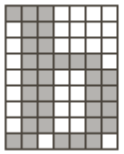
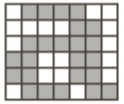
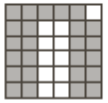


Token	Symbol	Triplet
0		(0, 2, 0) (3, 10, 1) (3, 18, 2) (3, 26, 1) (3, 34, 2) (3, 42, 1)
1		
2		

CSE-BUET

Symbol Based Coding



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

- Compression ratio:
  - Original: 9X51X1 = 459 bits
  - Compressed: (6 *entries* X 3 X 8 *bits*)+[(9X7)+(6X7)+(6X6)] = 285 bits
  - Ration: 1.61

CSE-BUET

Bit Plane Coding

- image is decomposed into a series of binary images (bit planes)
- process each bit plane individually
- Each bit plane is compressed using one of well known binary compression techniques.

CSE-BUET

Bit Plane Coding

Any gray value can be expressed as

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$$

CSE-BUET

Bit Plane Coding

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$$

- 0^{th} bit plane retains only bits at position 0

- Similarly (*m-1*)th order bit plane retains bits at position (*m*-1)

CSE-BUET

Bit Plane Coding

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$$

- Consider two gray values:
  - 128 (10000000_b) and
  - 127 (01111111_b)

CSE-BUET

Bit Plane Coding

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$$

- Every plane will have totally different value for these two gray values:
  - 128 (10000000_b) and
  - 127 (01111111_b)

CSE-BUET

Bit Plane Coding

- Alternate use gray code while finding bit planes:

$$g_i = a_i \oplus a_{i+1} \quad \text{for } 0 \le i \le m-2$$

$$g_m = a_{m+1}$$

CSE-BUET

Bit Plane Coding

A gray scale image

CSE-BUET

Bit Plane Coding