CSE6706:

Advanced Digital Image Processing

Dr. Md. Monirul Islam

CSE-BUET

Image Compression

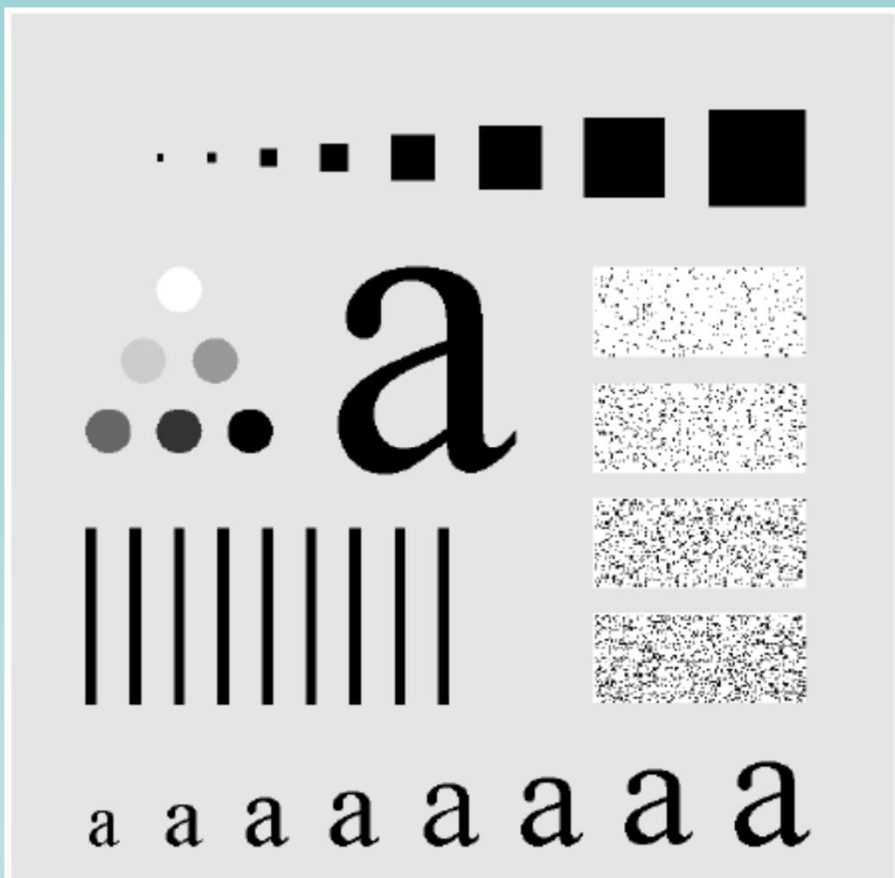
in transform domain and others …

CSE-BUET

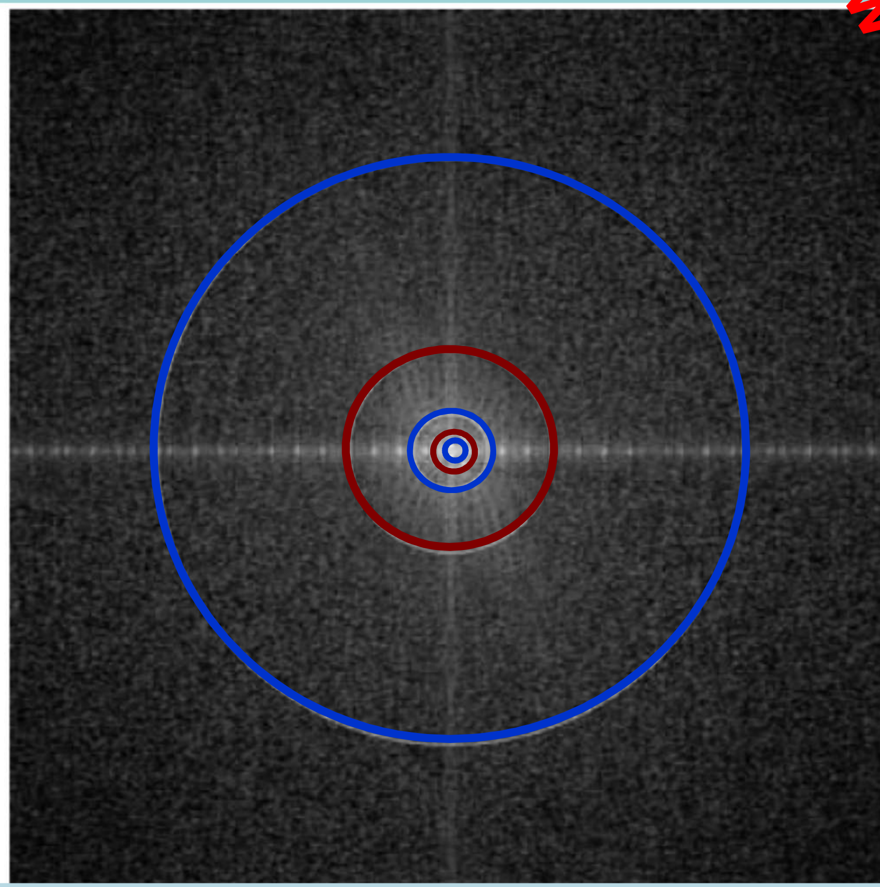
Ideal Low Pass Filter

Review

Example image



Centered Fourier Transform

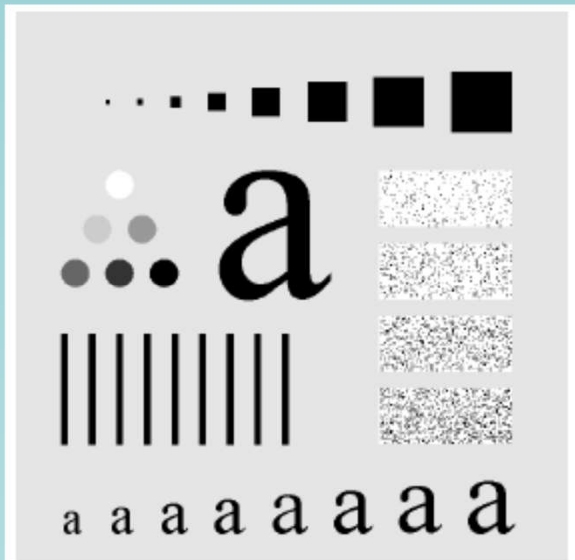


CSE-BUET

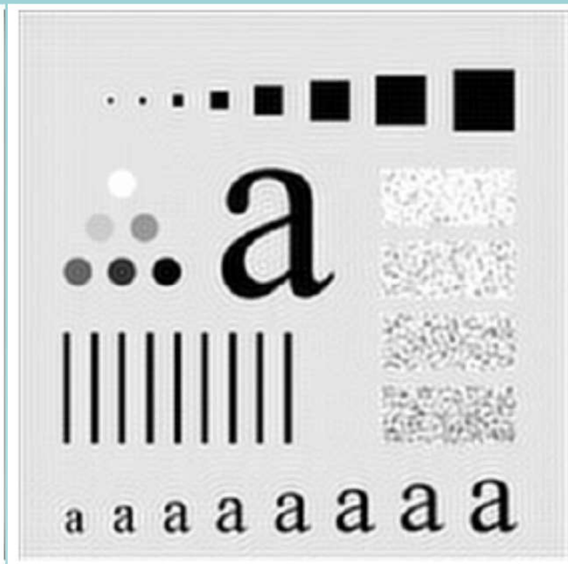
The superimposed circles have radius 10, 30, 60, 160, 460 and enclose 87.0, 93.1, 95.7, 97.8 and 99.2% energy

Ideal Low Pass Filter

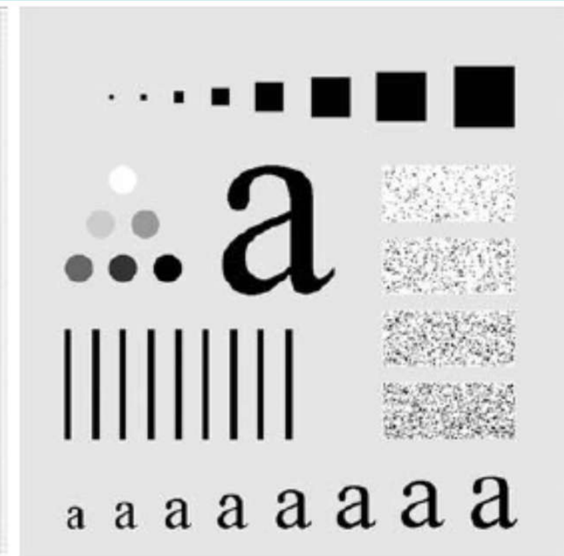
Review



Original



Reconstructed with 97.8% energy (160)

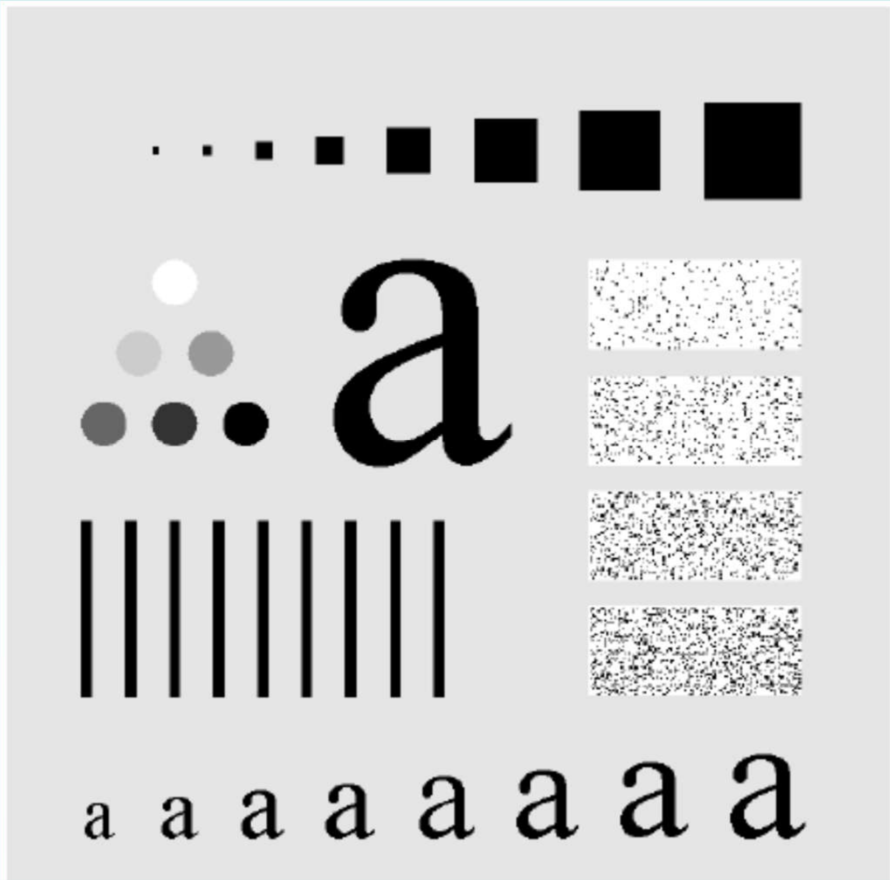


Reconstructed with 99.2% energy (460)

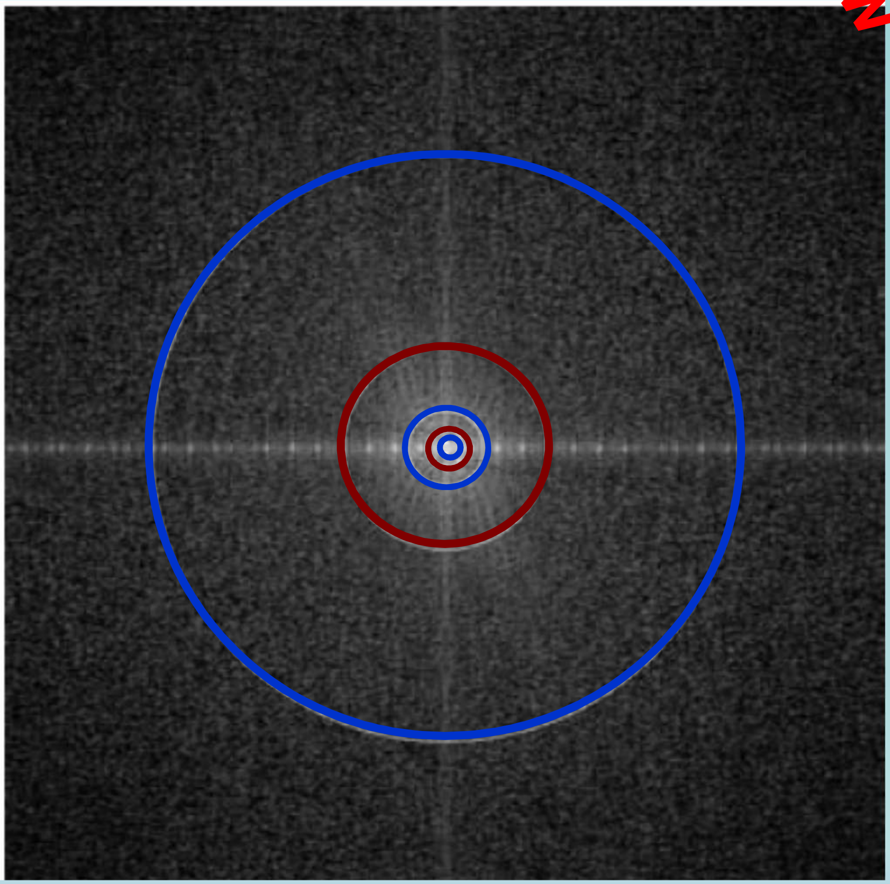
Ideal Low Pass Filter

Review

Example image



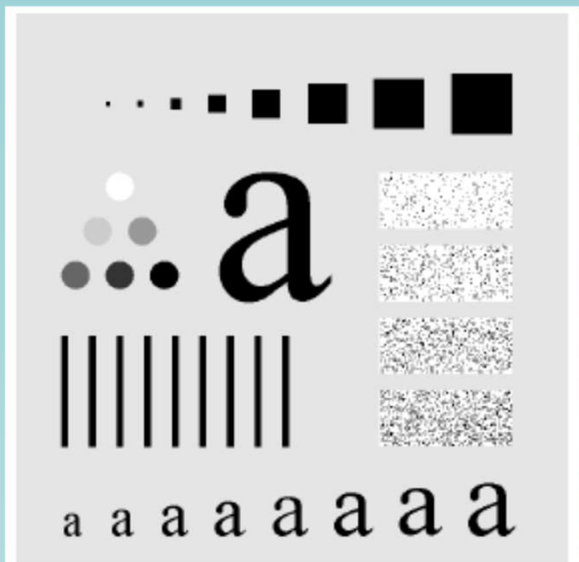
Centered Fourier Transform



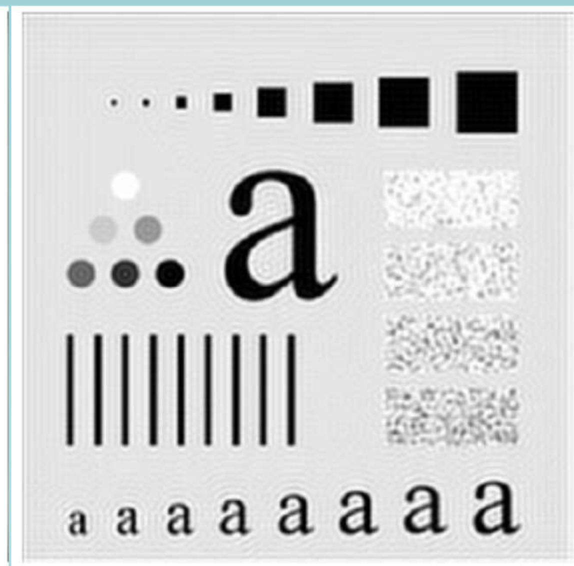
The superimposed circles have radius 10, 30, 60, 160, 460 and enclose 87.0, 93.1, 95.7, 97.8 and 99.2% energy

CSE-BUET

Ideal Low Pass Filter

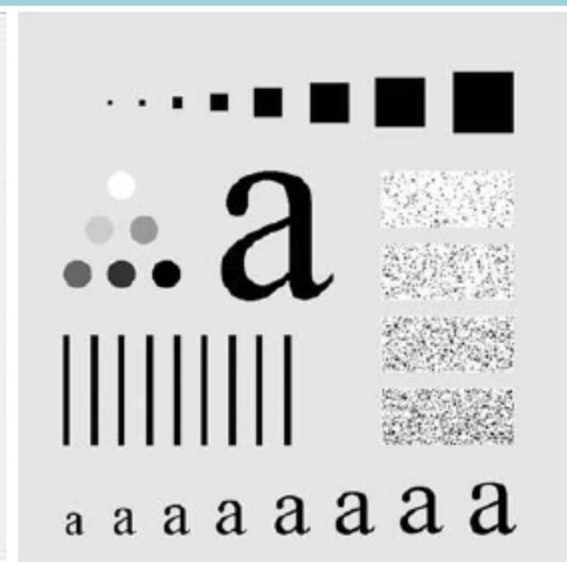


Original



Reconstructed with 97.8% energy (160)

around 10% coefficients used



Reconstructed with 99.2% energy (460)

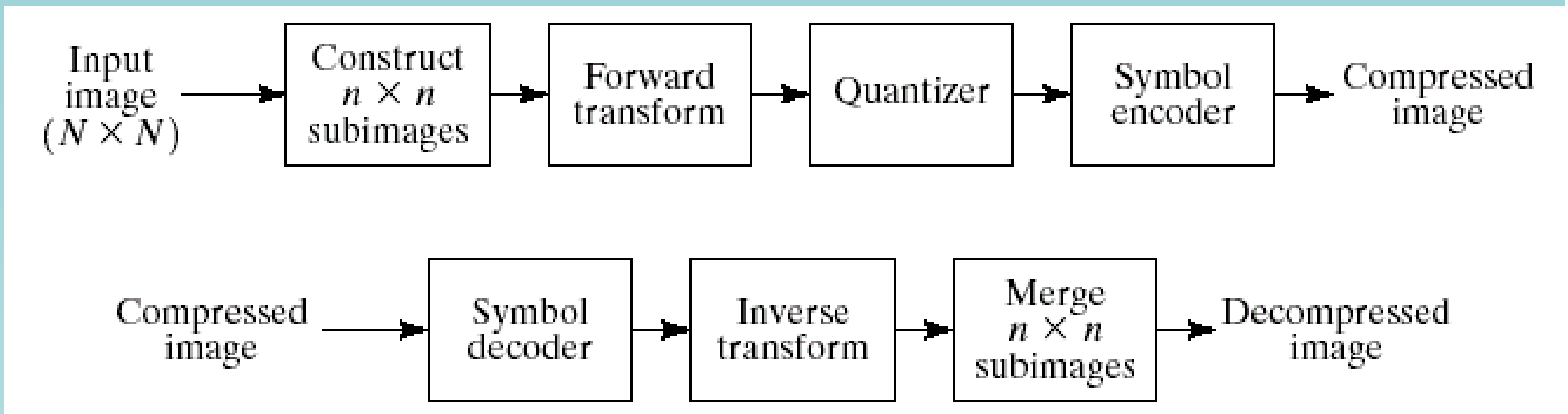
around 50% coefficient used

CSE-BUET

Image Compression in Transform Domain

- Recall the transform
  - All transform coefficients are not equal
  - Coefficient values contributes to image quality
  - Some of them may be coarsely quantized or even discarded

Image Compression in Transform Domain



- Total $(N/n)^2$ sub-images are processed
- Different transformations can be used
- Quantizer removes some of the coefficients or does further coarse quantization

Image Compression in Transform Domain

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

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

Transformation pair

- r, s: transform kernel, basis image, or basis matrix
- $T(u, v)$: transform coefficients

CSE-BUET

Properties of Transform Functions: Separability

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

Properties of Transform Functions: Symmetry

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

Fourier Transform Kernel

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

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

$$r(x,y,u,v)=e^{-j2\pi(ux+vy)/n}$$

$$s(x,y,u,v)=\frac{1}{n^2}e^{j2\pi(ux+vy)/n}$$

A computationally Simpler Kernel

Walsh-Hadamard

Transform

$$r(x,y,u,v) = s(x,y,u,v)$$

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

$$\text{where, } n = 2^m$$

CSE-BUET

A computationally Simpler Kernel

Walsh-Hadamard Transform

$$r(x,y,u,v) = s(x,y,u,v)$$

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

- $b_k(z) = k^{th}$ bit of z from right

A computationally Simpler Kernel

Walsh-Hadamard Transform

$$r(x,y,u,v) = s(x,y,u,v)$$

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

- b_k(z) = k^{th} bit of z from right
- Example:
  - Let $m = 3$, $z = 6 = 110_b$
  - $b_0(z) = 0$, $b_1(z) = 1$, $b_2(z) = 1$

A computationally Simpler Kernel

Walsh-Hadamard Transform

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

- All sum are in modulo 2
- $p(u)$'s and $p(v)$'s are defined as

$$\begin{aligned}
p_0(u) &= b_{m-1}(u) \\
p_1(u) &= b_{m-1}(u) + b_{m-2}(u) \\
p_2(u) &= b_{m-2}(u) + b_{m-3}(u) \\
&\vdots \\
p_{m-1}(u) &= b_1(u) + b_0(u)
\end{aligned}$$

A computationally Simpler Kernel

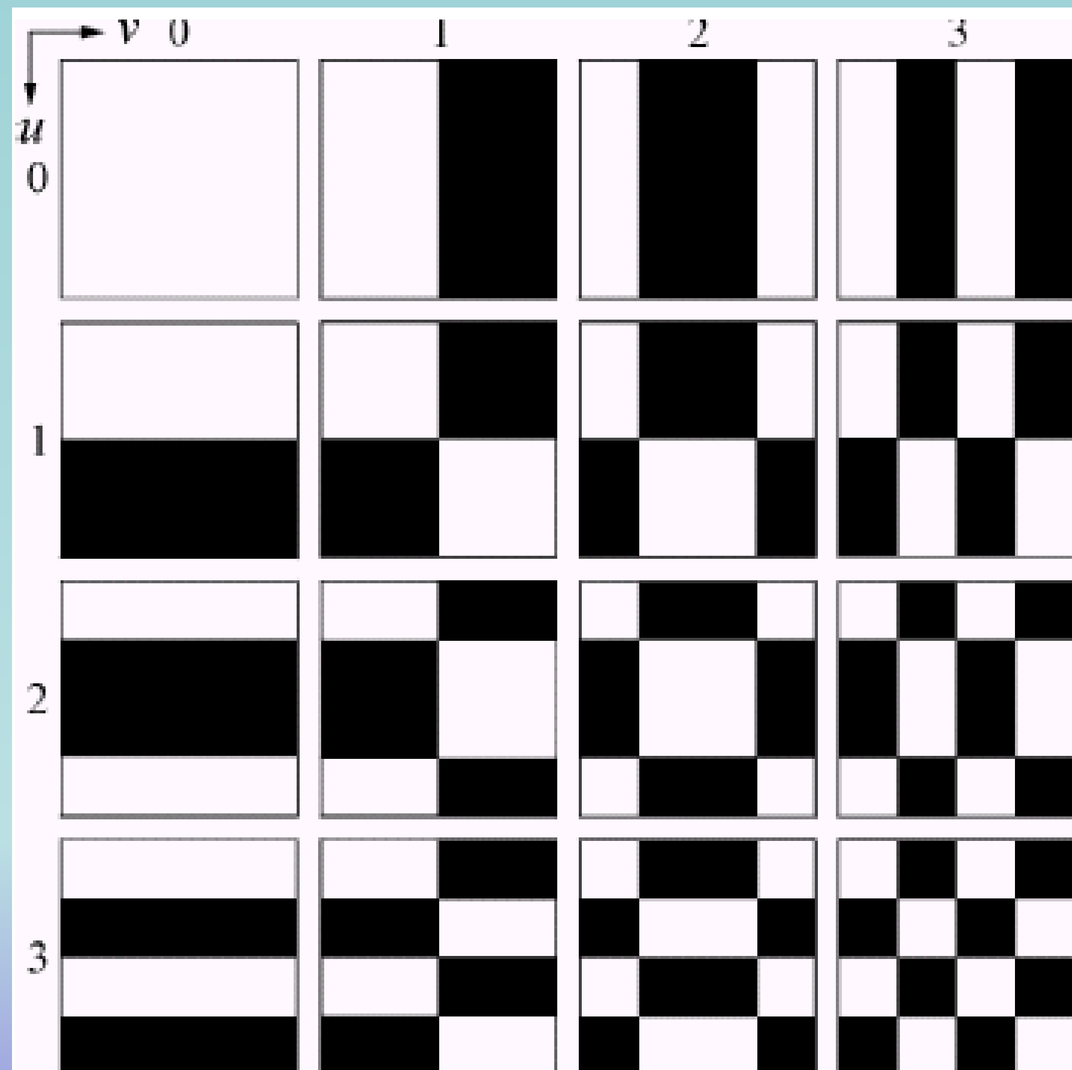
$$r(x, y, u, v)$$

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

Basis matrix for

Walsh-Hadamard

Transform



CSE-BUET

Discrete Cosine Transform (DCT)

$$r(x,y,u,v) = s(x,y,u,v)$$
$$= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

where,

$$\alpha(u) = \alpha(v) = \begin{cases} \sqrt{\frac{1}{n}} & \text{for } u,v = 0 \\ \sqrt{\frac{2}{n}} & \text{for } u,v = 1,2,\cdots,n-1 \end{cases}$$

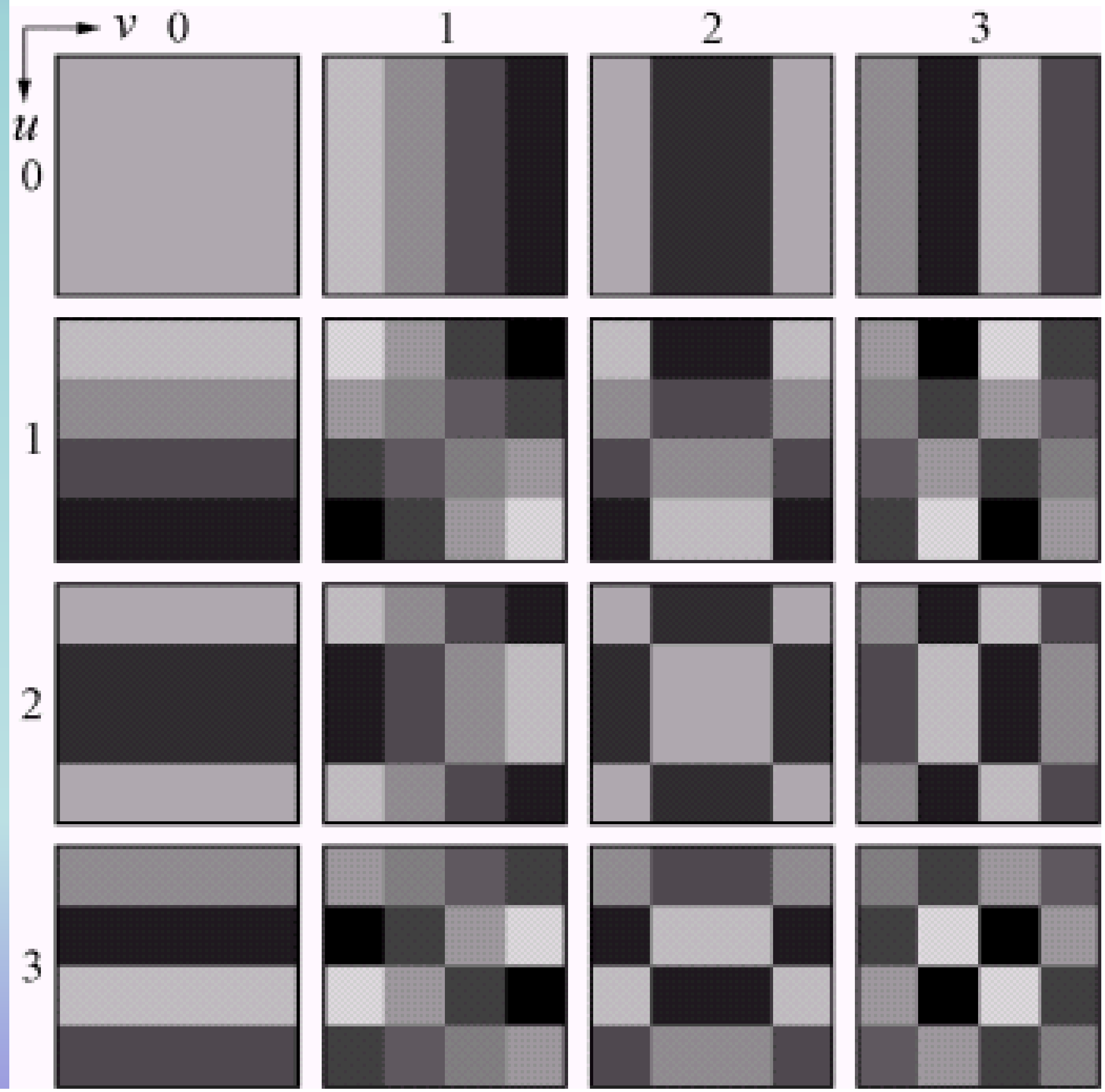
CSE-BUET

DCT Basis Function

$$r(x,y,u,v) = s(x,y,u,v)$$

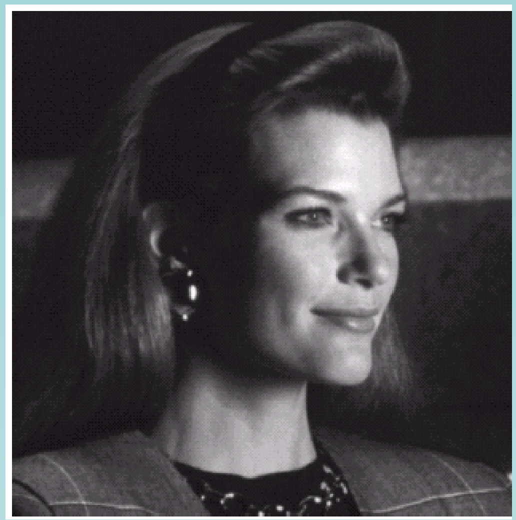
$$= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]$$

$$\times\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$



CSE-BUET

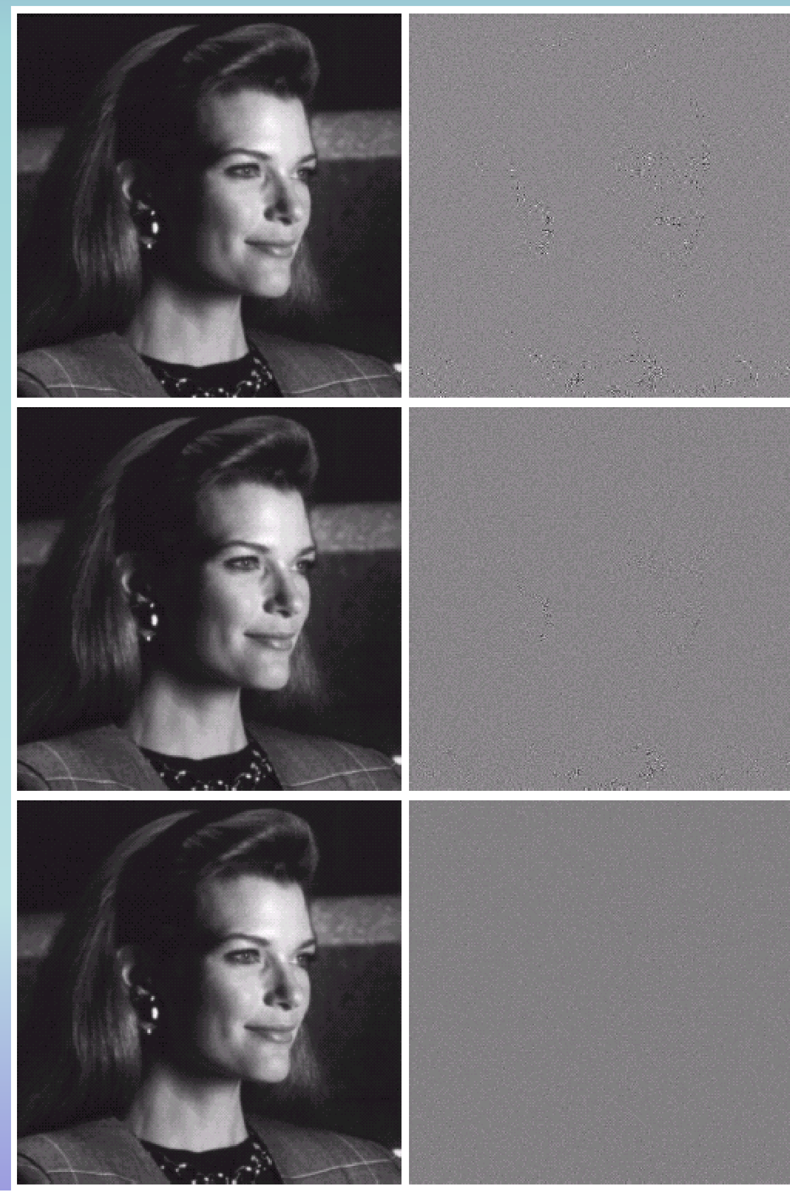
Example of Image Compression



A Monochrome Image

Reconstructed image with 50% Coefficients

Scaled Error



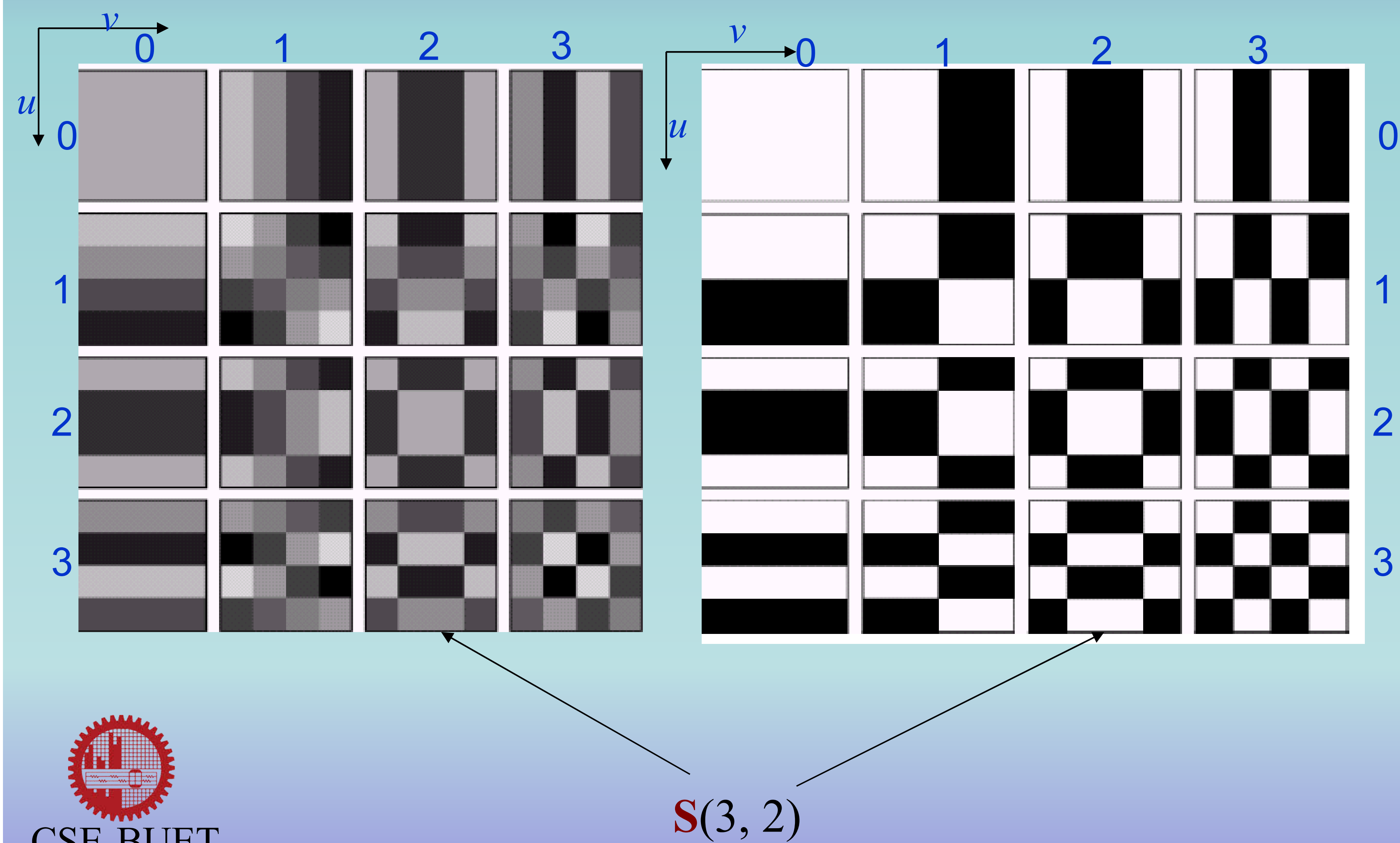
Fourier — *rms error* = 1.28

WHT — *rms error* = 0.86

DCT — *rms error* = 0.68

CSE-BUET

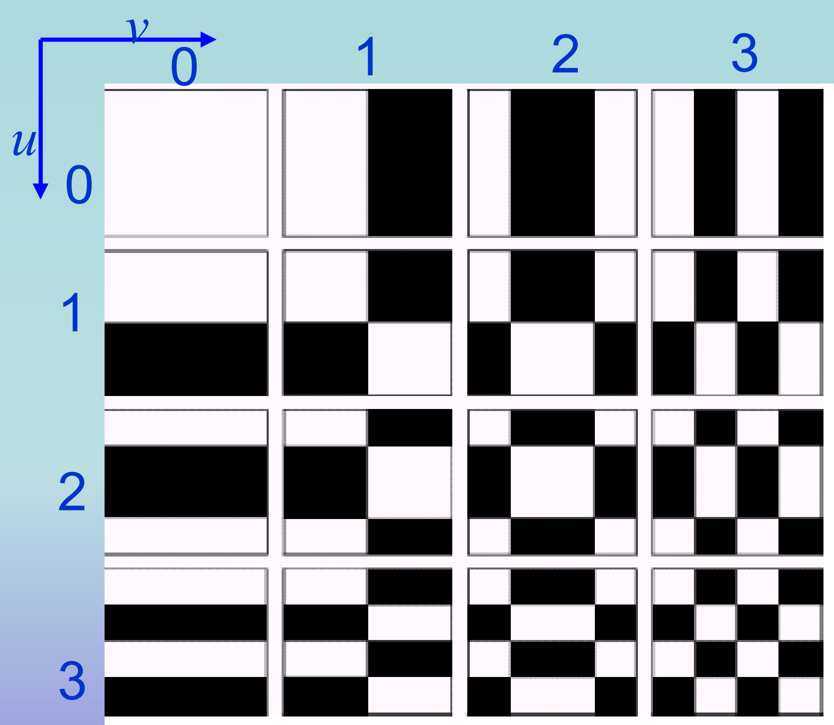
Basis Function



CSE-BUET

Basis Function

$$S_{uv} = \begin{bmatrix} s(0,0,u,v) & s(0,1,u,v) & \cdots & s(0,n-1,u,v) \\ s(1,0,u,v) & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ s(n-1,0,u,v) & s(n-1,1,u,v) & \cdots & s(n-1,n-1,u,v) \end{bmatrix}$$



Inverse Transform for a sub-image

- For a sub-image of size nXn, the reconstruction equation will be:

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

A single pixel
of a sub-image

Inverse Transform for a sub-image

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

A single pixel
of a sub-image

- In terms of basis functions:

$$\mathbf{G} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}$$

sub-image

CSE-BUET

Inverse Transform for a sub-image

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

A single pixel
of a sub-image

- In terms of basis functions:

$$\mathbf{G} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) \mathbf{S}_{uv}$$

sub-image

  - **G** is now a linear combination of basis matrices

CSE-BUET

Inverse Transform for a sub-image

$$\mathbf{G} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}$$

- If **G** is recalculated so, where is the compression?

CSE-BUET

Inverse Transform for a sub-image

$$\mathbf{G} \quad = \quad \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}$$

- We need to truncate some of the coefficients from $T(u, v)$:

$$\gamma(u,v) = \begin{cases} 0 & \text{if } T(u,v) \text{ satisfies a specified truncation criteria} \\ 1 & \text{otherwise} \end{cases}$$

Inverse Transform for a sub-image

$$\mathbf{G} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}$$

- We need to truncate some of the coefficients from $T(u, v)$:

$$\gamma(u,v) = \begin{cases} 0 & \text{if } T(u,v) \text{ satisfies a specified truncation criteria} \\ 1 & \text{otherwise} \end{cases}$$

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \gamma(u,v) T(u,v)\mathbf{S}_{uv}$$

Reconstructed sub-image

CSE-BUET

Compression Error

$$e_{ms} = E\left\{ \left\| \mathbf{G} - \hat{\mathbf{G}} \right\|^2 \right\}$$

CSE-BUET

Compression Error

$$e_{ms} = E\left\{\left\|\mathbf{G} - \hat{\mathbf{G}}\right\|^2\right\}$$

$$= E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv} - \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \gamma(u,v)T(u,v)\mathbf{S}_{uv}\right\|^2\right\}$$

Compression Error

$$e_{ms} = E\left\{\left\|\mathbf{G} - \hat{\mathbf{G}}\right\|^2\right\}$$

$$= E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv} - \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \gamma(u,v)T(u,v)\mathbf{S}_{uv}\right\|^2\right\}$$

$$= E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}\left[1-\gamma(u,v)\right]\right\|^2\right\}$$

Compression Error

$$e_{ms} = E\left\{\left\|\mathbf{G} - \hat{\mathbf{G}}\right\|^2\right\}$$

$$= E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\hat{\mathbf{S}}_{uv} - \sum_{u=0}^{n-1}\sum_{v=0}^{n-1}\gamma(u,v)T(u,v)\mathbf{S}_{uv}\right\|^2\right\}$$

$$= E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}\left[1-\gamma(u,v)\right]\right\|^2\right\}$$

$$\vdots$$

$$= \sum_{u=0}^{n-1}\sum_{v=0}^{n-1}\sigma_{T(u,v)}^2\left[1-\gamma(u,v)\right]$$

Compression Error

$$e_{ms} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \sigma^2_{T(u,v)} \left[1 - \gamma(u,v)\right]$$

- Error = sum of variances of discarded coefficients
- Transformations that redistribute or pack info into a fewer coefficients, produce less error

Transformation with Different Compression Error

- Karhunen-Loeve (KL) transform
  - produces minimum m.s. error
  - Basis functions are image dependent
  - Needs optimization
  - High computational complexity

Transformation with Different Compression Error

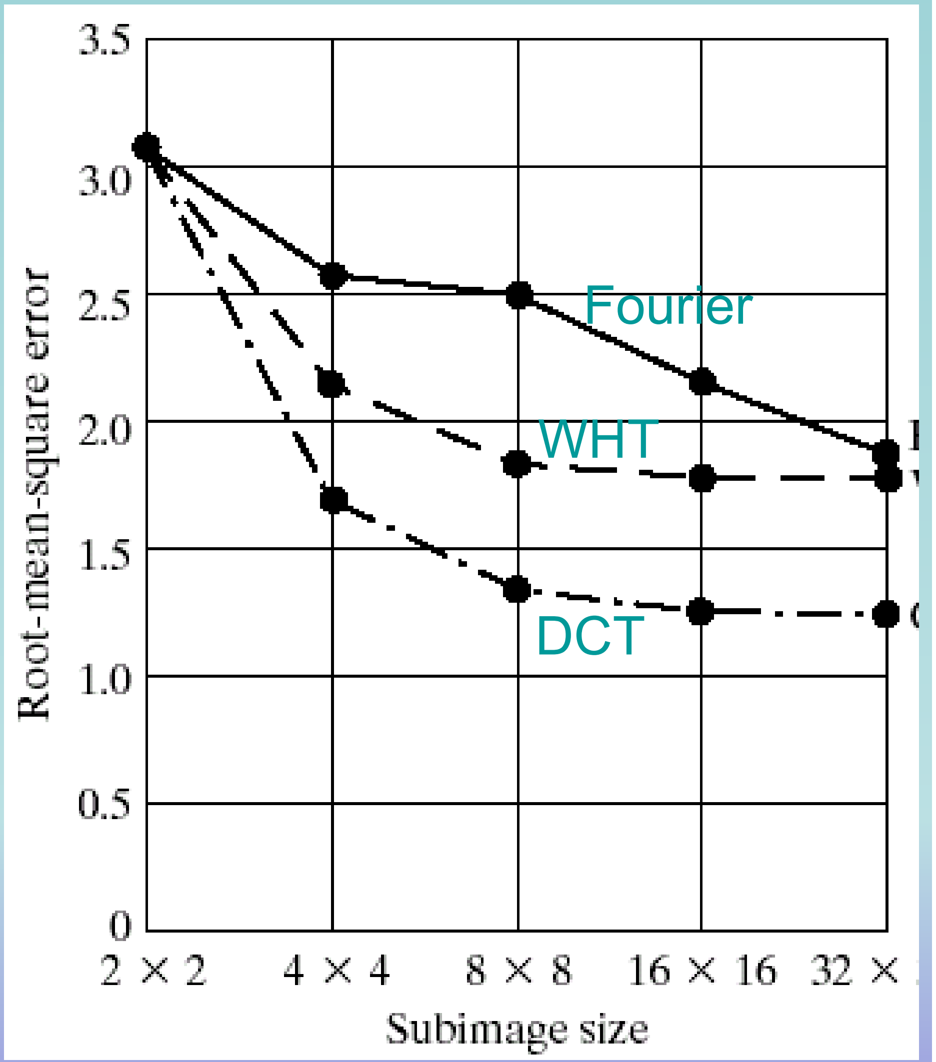
- WT, DCT, DFT
  - sinusoidal
  - Fixed basis functions
  - Higher error than KLT

- WT
  - simplest to implement

- DCT and DFT
  - Approximate KLT

Effect of sub-image size on Compression Error

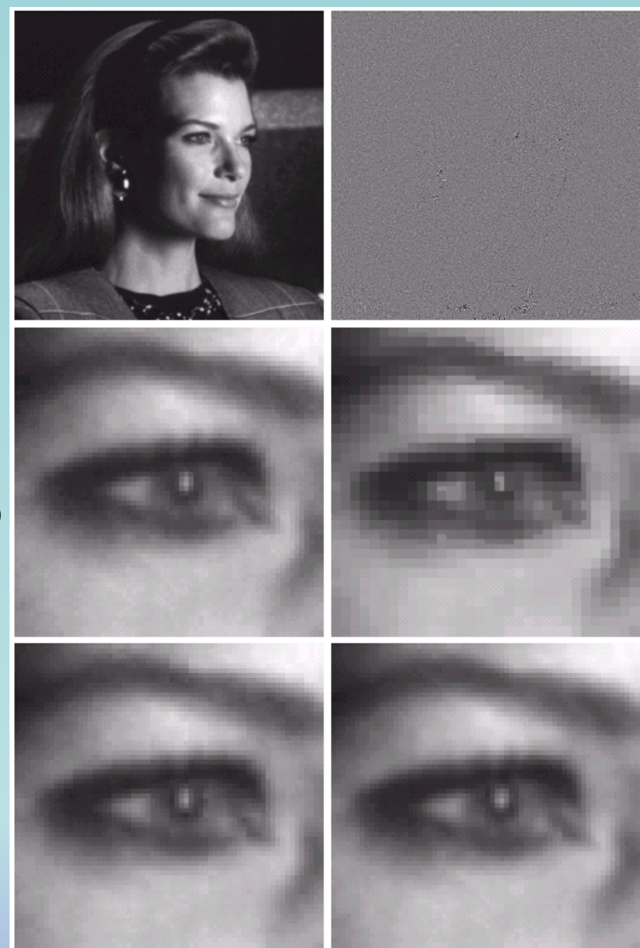
- Images are subdivided to reduce correlation (redundancy) between adjacent sub-images
- Complexity and compression quality increase with sub-image size
- However error reduces with the size



CSE-BUET

Reconstructed with 25% coeff.

Error Image



Original close up

Using subimage of size 2X2

Using subimage of size 4X4

Using subimage of size 8X8

CSE-BUET

Bit Allocation

- Means: truncating, quantizing and assigning code
- 2 way for truncation:
  - Zonal coding
  - Thresholding

Bit Allocation

- Means: truncating, quantizing and assigning code
- 2 way for truncation:
  - Zonal coding
    - Retained coefficients from a *zone*
    - *Zone* usually determined by variance
    - Coefficients of largest variances retained
    - They usually appears near the origin
    - Discarded zone may contain significant information
    - Same mask for every sub-image

Bit Allocation

- Means: truncating, quantizing and assigning code
- 2 way for truncation:
  - Thresholding
    - Retain coefficient of larger coefficients
    - May appear anywhere
    - So, both coefficients and locations are transmitted
    - Better than zonal coding
    - Different mask for different sub-image

Zonal Mask, $\gamma(u, v)$

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

Zonal Mask, $\gamma(u, v)$

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

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

Threshold Mask, $\gamma(u, v)$

Zonal Mask, $\gamma(u, v)$

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

Zonal Bit Allocation

8	7	6	4	3	2	1	0
7	6	5	4	3	2	1	0
6	5	4	3	3	1	1	0
4	4	3	3	2	1	0	0
3	3	3	2	1	1	0	0
2	2	1	1	1	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

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

Threshold Mask, $\gamma(u, v)$

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Threshold Code sequence

Zonal Mask, $\gamma(u, v)$

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

Zonal Bit Allocation

8	7	6	4	3	2	1	0
7	6	5	4	3	2	1	0
6	5	4	3	3	1	1	0
4	4	3	3	2	1	0	0
3	3	3	2	1	1	0	0
2	2	1	1	1	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Threshold Mask, $\gamma(u, v)$

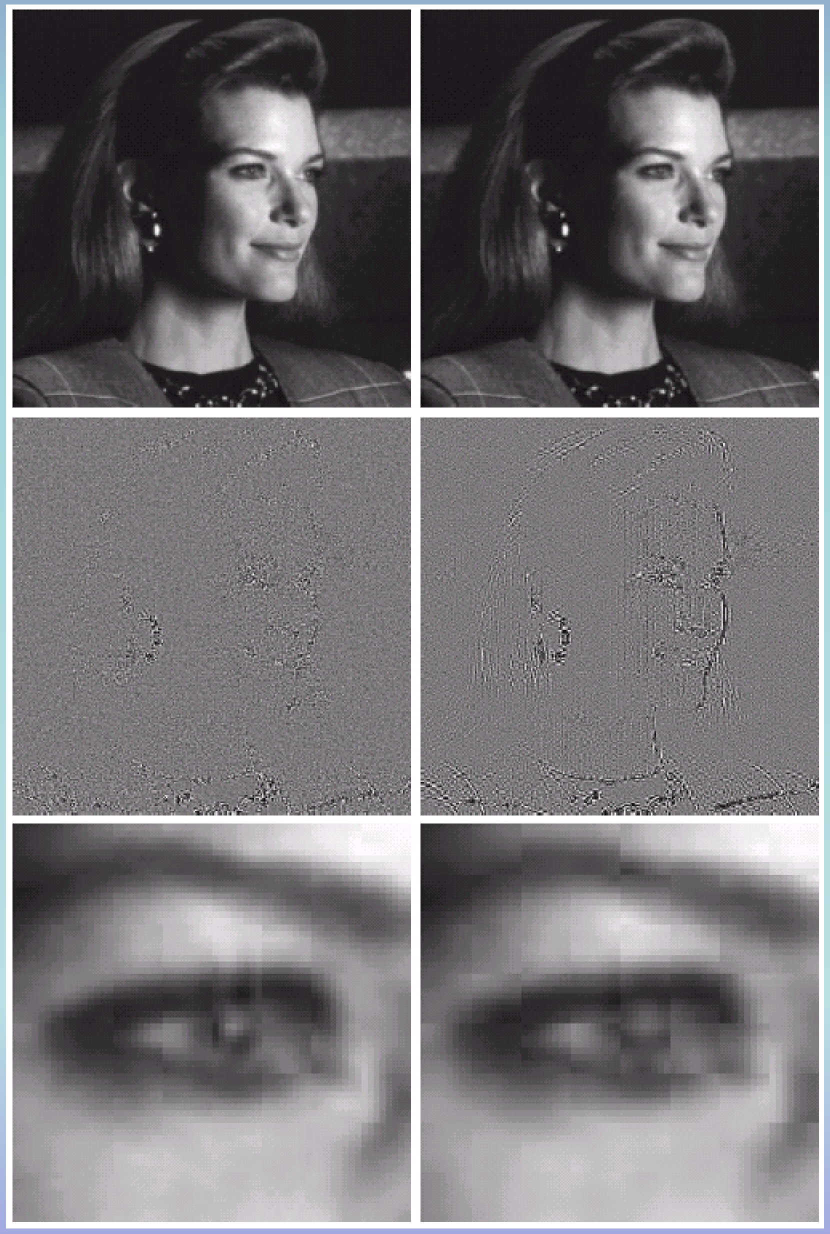
1	1	0	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Threshold Code sequence

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Using 12.5%
coefficients only

Threshold coding Zonal coding



CSE-BUET

Thresholding a Transform Image

- 3 ways to threshold
  - A single global threshold for all sub-images
  - Different threshold for different sub-image
  - Location dependant threshold

Thresholding a Transform Image

- 3 ways to threshold
  - A single global threshold for all subimages
    - Depending on the threshold, different no. of coefficients survive in different sub-images
    - Level of compression differs from image to image
  - Different threshold for different sub-image
  - Location dependant threshold

Thresholding a Transform Image

- 3 ways to threshold
  - A single global threshold for all subimages
    - Depending on the threshold, different no. of coeff. survives in different subimages
    - Level of compression differs from image to image
  - Different threshold for different subimage
    - *N-largest coding*: same number of coeff survives
    - Fixed code rate is known in advance
  - Location dependant threshold

CSE-BUET

Thresholding a Transform Image

- 3 ways to threshold
  - A single global threshold for all subimages
    - Depending on the threshold, different no. of coeff. survives in different subimages
    - Level of compression differs from image to image
  - Different threshold for different subimage
    - *N-largest coding*: same number of coeff survives
    - Fixed code rate is known in advance
  - Location dependant threshold
    - Variable code rate like global thresholding
    - But advantage of thresholding and quantizing together

Thresholding and Quantizing in a single Step: Location Depending Thresholding

Recall the reconstruction equation

$$\hat{\mathbf{G}} \quad = \quad \sum_{u=0}^{n-1}\sum_{v=0}^{n-1}\gamma(u,v)T(u,v)\mathbf{S}_{uv}$$

Thresholding and Quantizing in a single Step: Location Depending Thresholding

Recall the reconstruction equation

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \gamma(u,v) T(u,v) \mathbf{S}_{uv}$$

We can replace $\gamma(a, b)T(u, v)$ by

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

CSE-BUET

Thresholding and Quantizing in a single Step: Location Depending Thresholding

We can replace $\gamma(a, b)T(u, v)$ by

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

where, $Z(u, v)$ is a normalizing element as given by,

$$\mathbf{Z} = \begin{bmatrix} Z(0,0) & Z(0,1) & \cdots & Z(0,n-1) \\ Z(1,0) & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ Z(n-1,0) & Z(n-1,1) & \cdots & Z(n-1,n-1) \end{bmatrix}$$

Thresholding and Quantizing in a single Step: Location Depending Thresholding

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1}\gamma(u,v)T(u,v)\mathbf{S}_{uv}$$

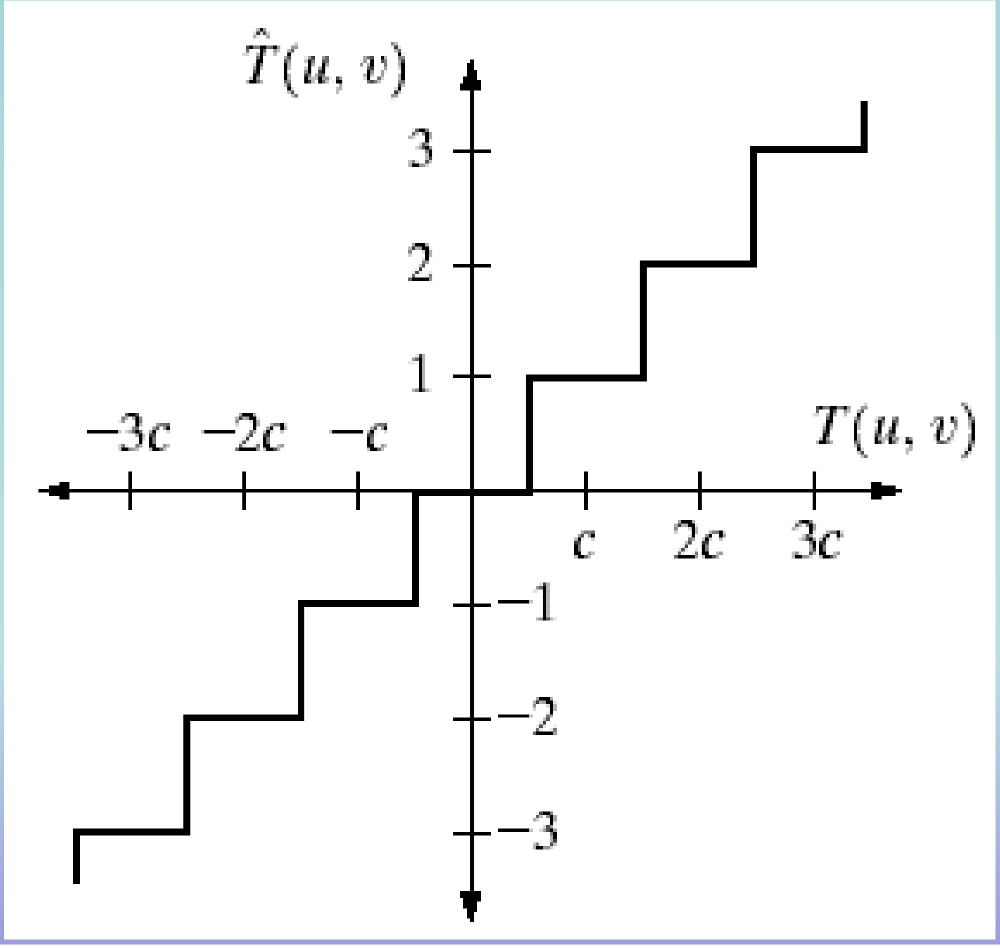
When reconstruct, use $\dot{T}(u,v) = \hat{T}(u,v)Z(u,v)$

Thus, we get

$$\dot{\mathbf{G}} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1}\dot{T}(u,v)\mathbf{S}_{uv}$$

Graphical Representation of $\hat{T}(u,v)$

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



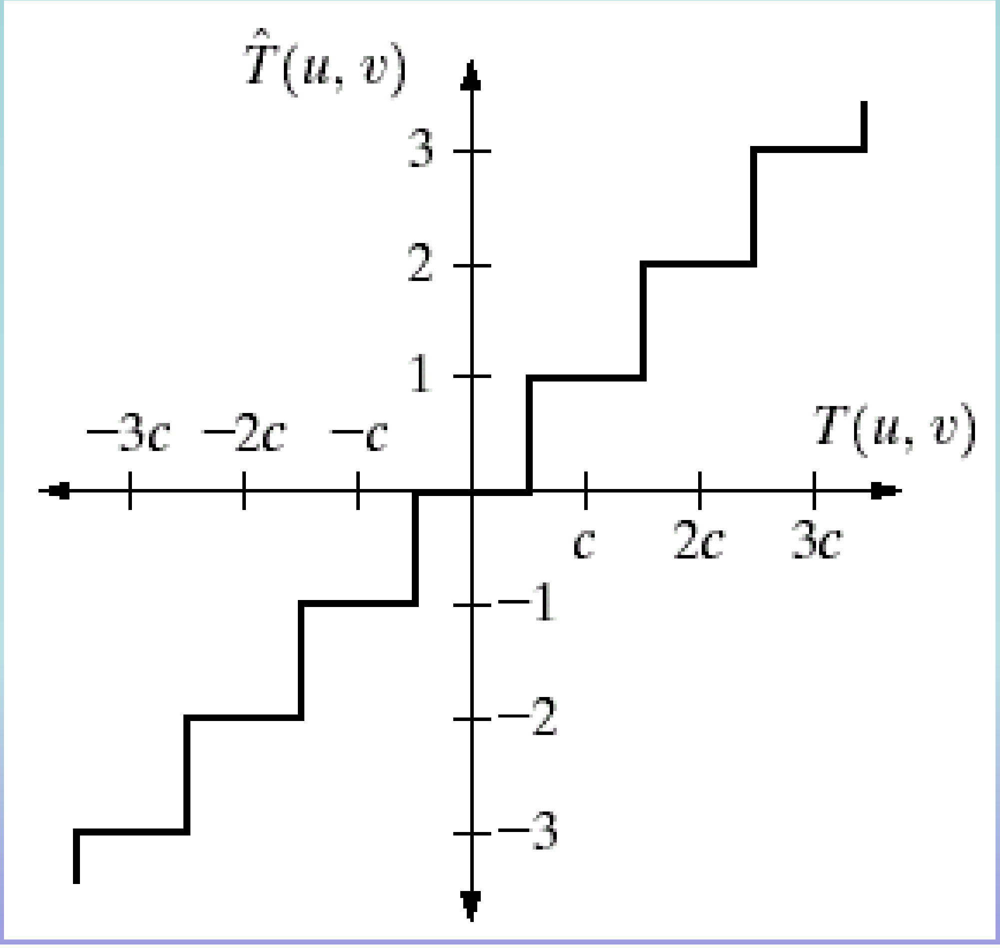
CSE-BUET

Graphical Representation of $\hat{T}(u,v)$

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

Value of $\hat{T}(u,v)$ assumes k if

$$kc - \frac{c}{2} \le T(u,v) \le kc + \frac{c}{2}$$



CSE-BUET

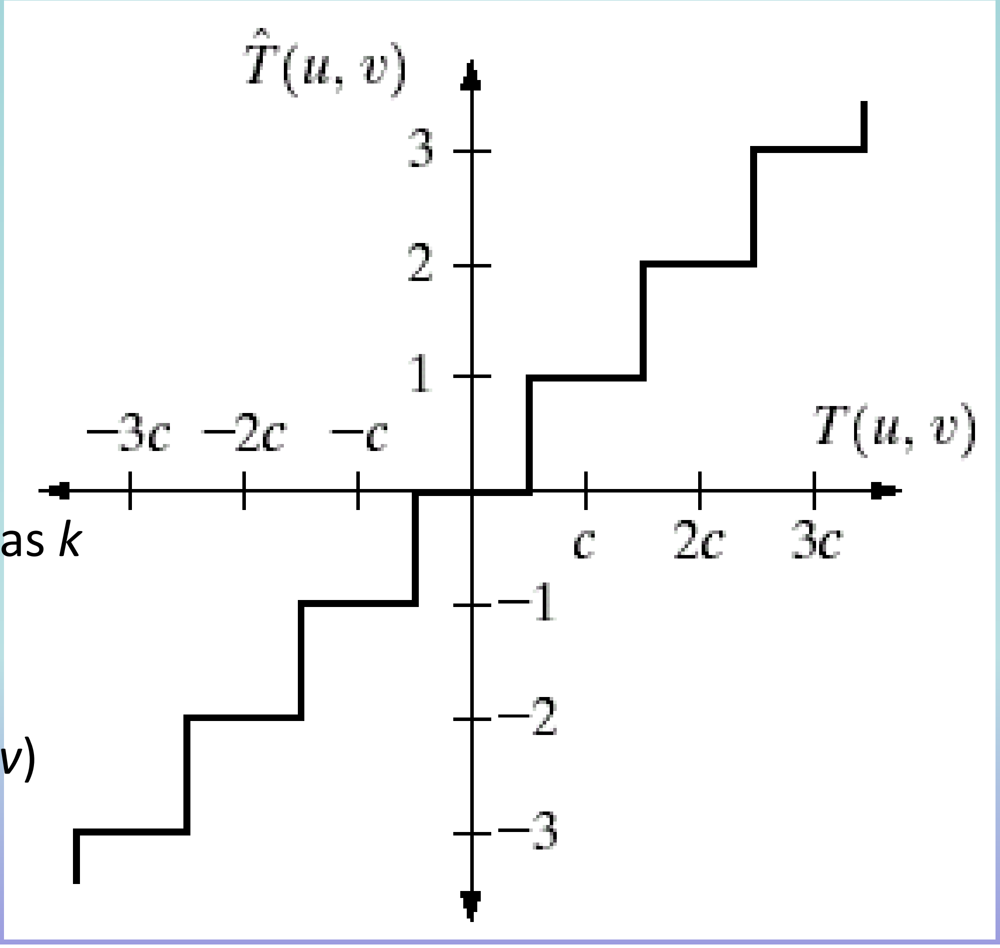
Graphical Representation of $\hat{T}(u,v)$

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

Value of $\hat{T}(u,v)$ assumes k if

$$kc - \frac{c}{2} \leq T(u,v) \leq kc + \frac{c}{2}$$

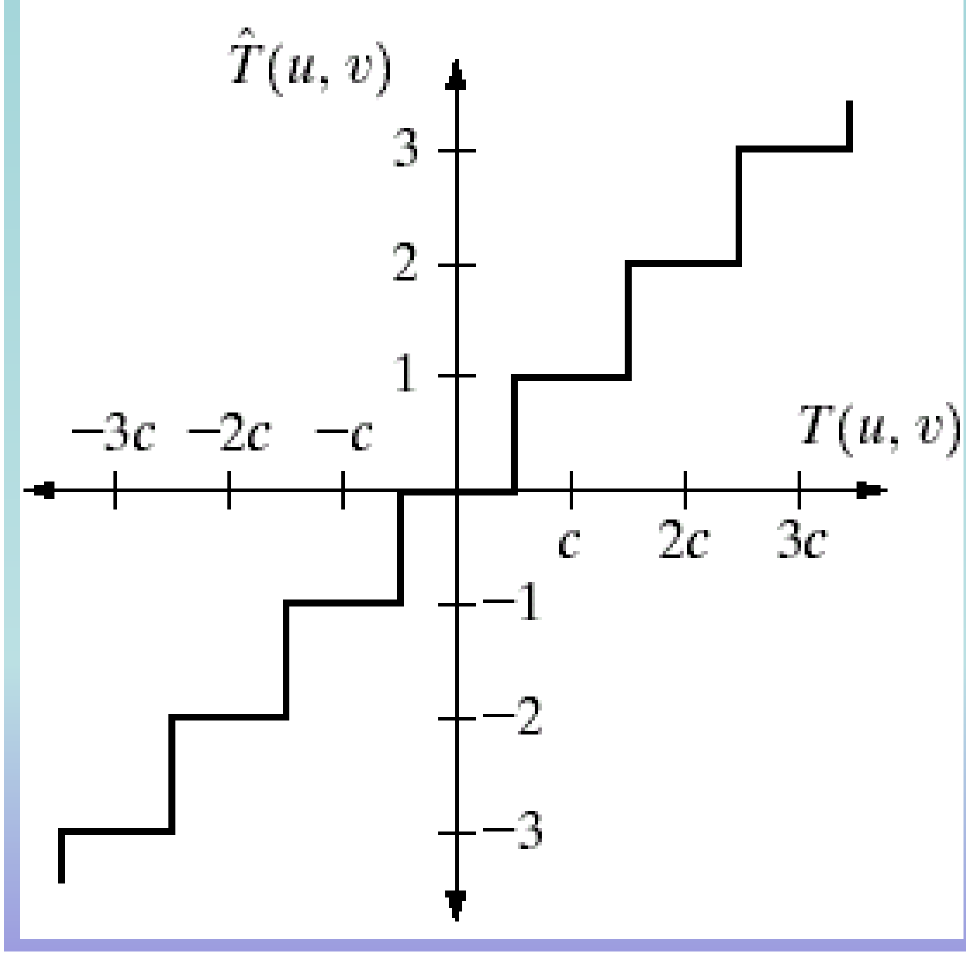
- Code length for $\hat{T}(u,v)$ increases as *k* increases
- *c* controls *k*
- *c* controls representations of *T*(*u*, *v*)



CSE-BUET

Graphical Representation of $\hat{T}(u,v)$

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

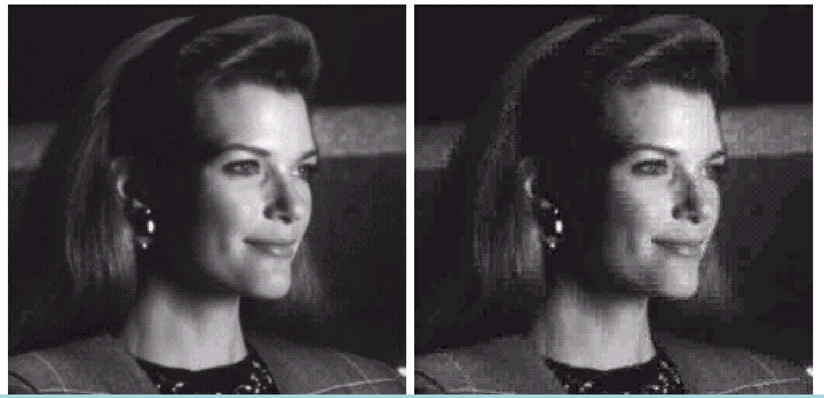


A typical **Z** matrix

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

34:1 | 67:1

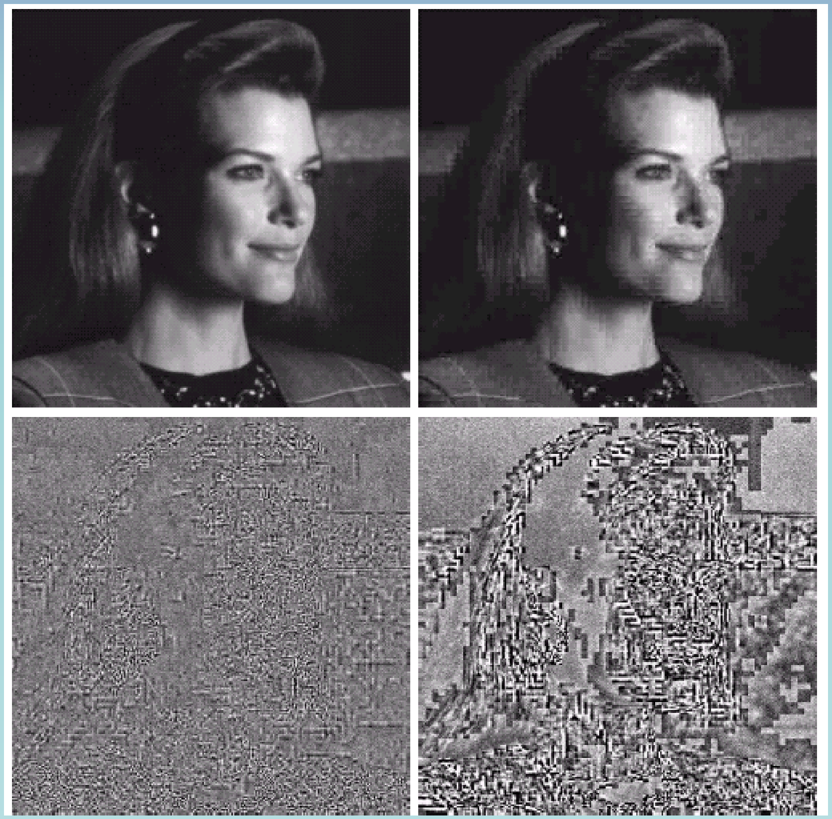
With normalizing matrix, **Z**



With normalizing matrix, 4**Z**

34:1 67:1

With normalizing matrix, **Z**



With normalizing matrix, 4**Z**

CSE-BUET

34:1 67:1

With normalizing matrix, **Z**

With normalizing matrix, 4**Z**

CSE-BUET

JPEG Compression Standard

- Joint photographic Experts Group (JPEG)
- 3 coding systems
  - Lossy baseline coding system
  - Extended coding system
  - Loss less independent coding system

CSE-BUET

JPEG Compression Standard

- Joint photographic Experts Group (JPEG)
- 3 coding systems
  - Lossy baseline coding system
    - *Based on DCT*
    - *Appropriate for most compression applications*
  - Extended coding system
  - Loss less independent coding system

CSE-BUET

JPEG Compression Standard

- Joint photographic Experts Group (JPEG)
- 3 coding systems
  - Lossy baseline coding system
    - *Based on DCT*
    - *Appropriate for most compression applications*
  - Extended coding system
    - *Greater compression*
    - *Higher precision*
    - *Progressive reconstruction*
  - Loss less independent coding system

CSE-BUET

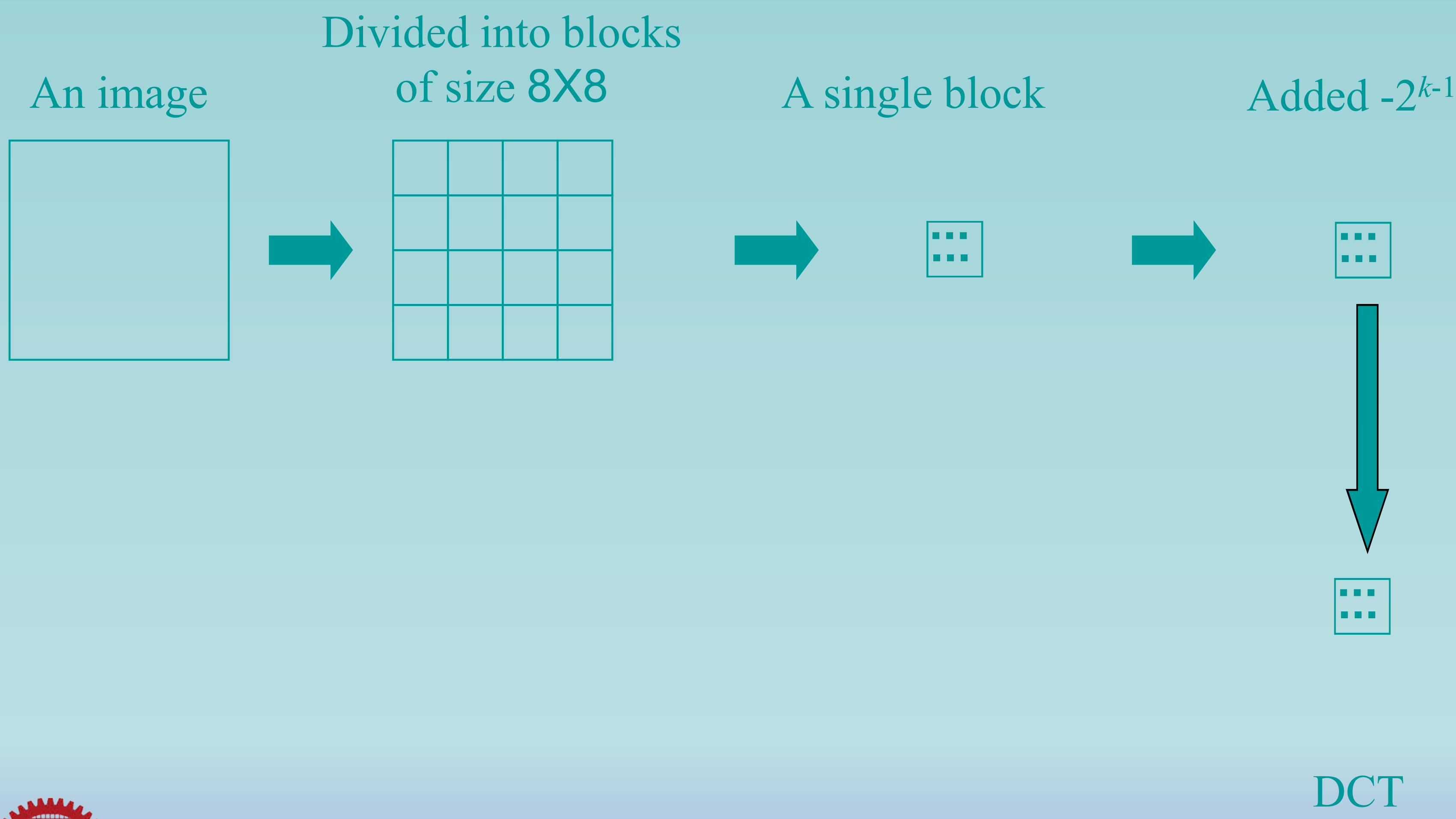
JPEG Compression Standard

- Joint photographic Experts Group (JPEG)
- 3 coding systems
  - Lossy baseline coding system
    - *Based on DCT*
    - *Appropriate for most compression applications*
  - Extended coding system
    - *Greater compression*
    - *Higher precision*
    - *Progressive reconstruction*
  - Loss less independent coding system
    - *Suitable for reversible compression*

JPEG Lossy Baseline Coding System

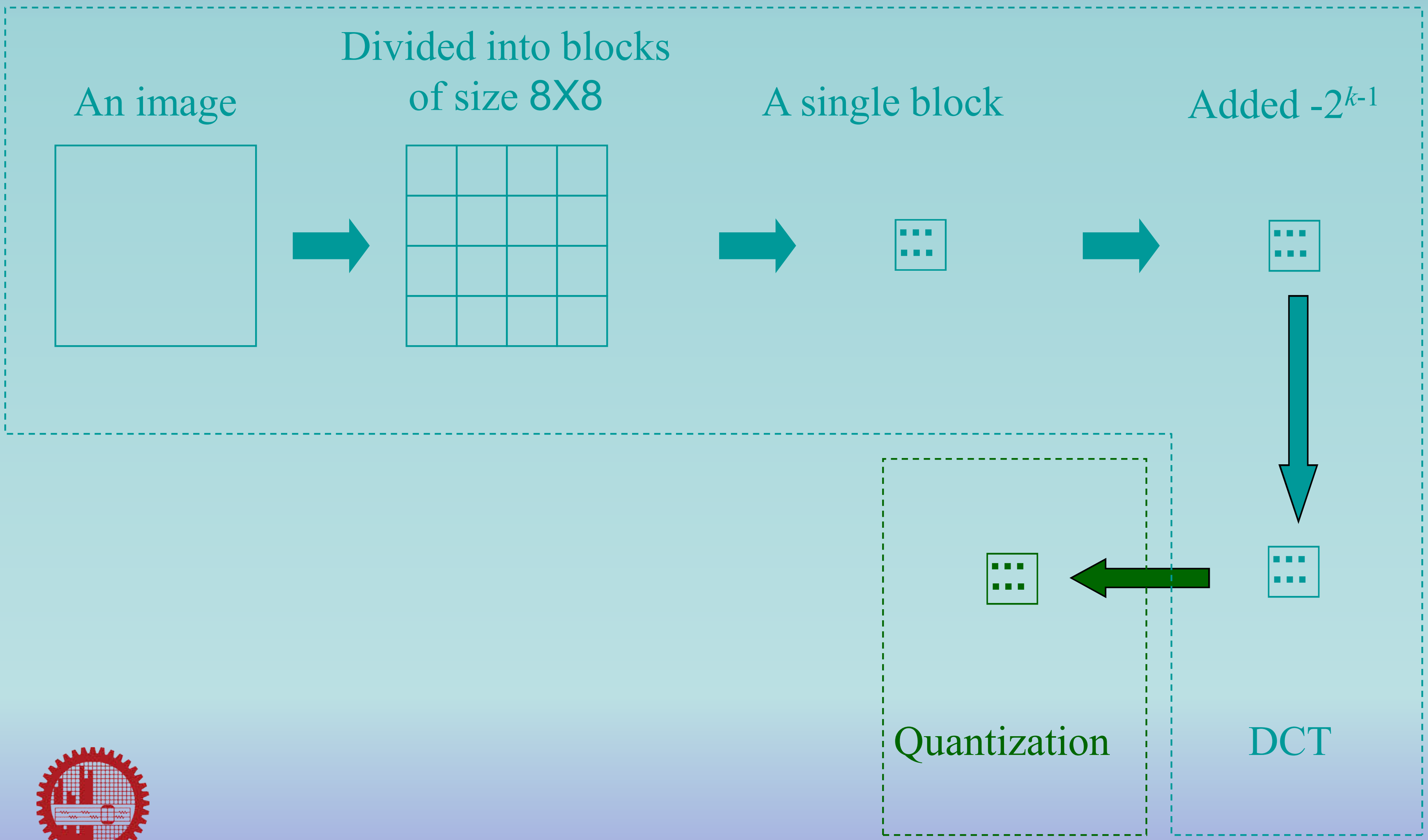
- 3 sequential steps
  - DCT computation
  - Quantization
  - Variable length code assignment

JPEG Lossy Baseline Coding System



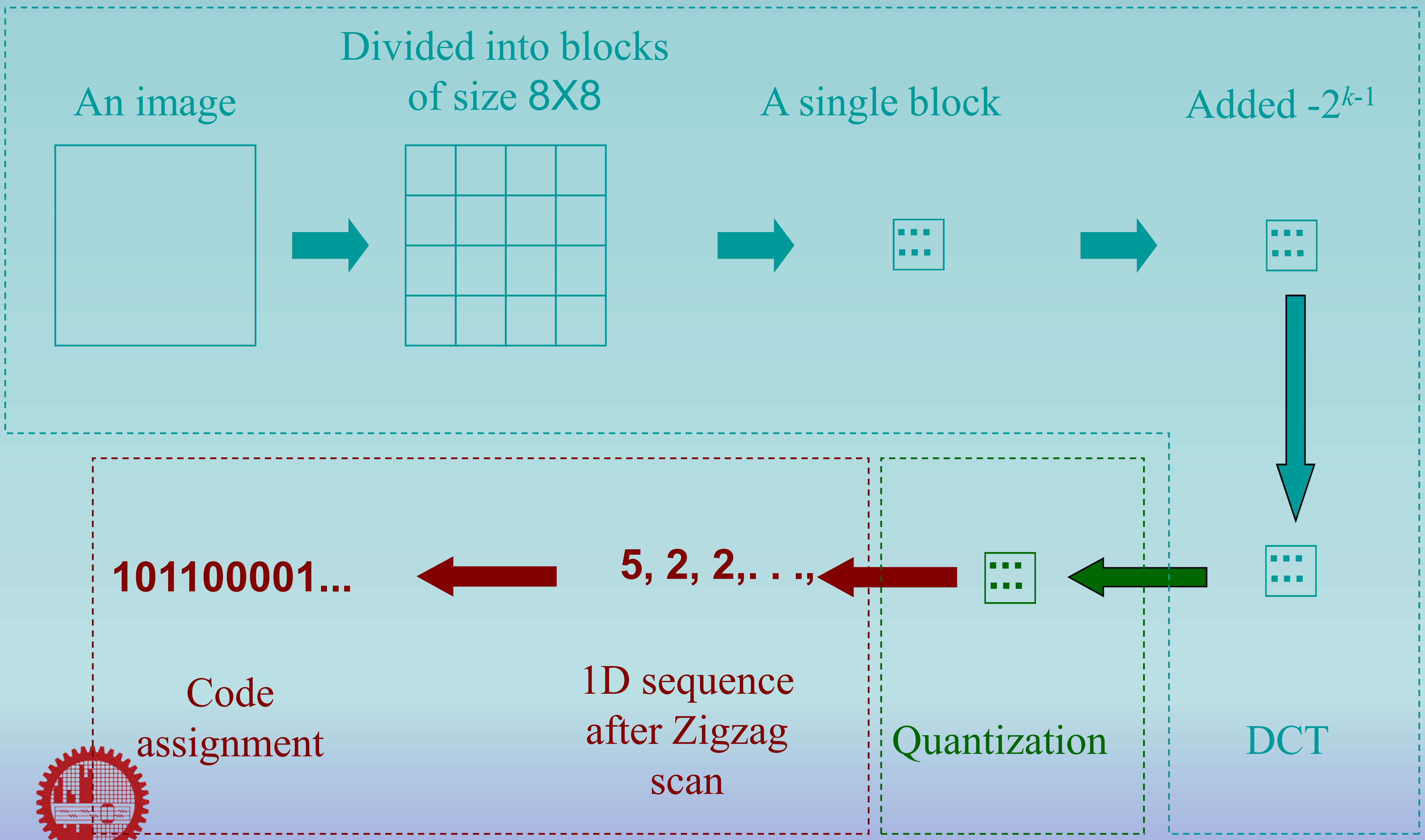
CSE-BUET

JPEG Lossy Baseline Coding System



CSE-BUET

JPEG Lossy Baseline Coding System



CSE-BUET

JPEG Lossy Baseline Coding System

- Assign code to *nonzero* AC components
  - Assign variable length code
  - Indicate coefficient value and number of leading zeros
- Assign code to DC component
  - Find the difference of the current DC and the DC of prev block
  - Code the difference

CSE-BUET

JPEG Coefficient Coding Categories

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Code Assignment for DC Coefficient

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Code Assignment for DC Coefficient

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Example: DC difference = -9.

CSE-BUET

JPEG Coefficient Coding Categories

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Code Assignment for DC Coefficient

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Example: DC difference = -9.

Base Code 101 with length 3

Concatenate other 4 bit from -9.

For – diff: last 4 bits of -9-1 =0110

So, final: 1010110 which is 7 bits

Codes for AC Coeff.

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Codes for AC Coeff.

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26

Codes for AC Coeff.

Example: AC -8.

With 2 leading 0

Code?

CSE-BUET

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

JPEG Coefficient Coding Categories

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Codes for AC Coeff.

Example: AC -8.

With 2 leading 0

Code: 1111111110001001 = 16 bits

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Codes for AC Coeff.

Example: AC -8.

With 2 leading 0

Code: 1111111110001001 = 16 bits

Last 4 bits from

$-8-1 = 0111_b$

CSE-BUET

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
$\boxed{2/4}$	$\boxed{1111111110001001}$	$\boxed{20}$	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Example

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

Example 8X8 block

Max gray value in the image 2^8

Example

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

After subtracting by 2^{8-1} =128

Then, find its DCT

Example

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

DCT Coefficients

Normalize them by $\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$

Example

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

Normalized Coefficients

Example

The normalization **Z** matrix

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

Example

$$\hat{T}(0,0) = \text{round}\left[\frac{T(0,0)}{Z(0,0)}\right]$$

$$= \text{round}\left[\frac{-415}{16}\right] = -26$$

CSE-BUET

Example

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

Find the 1D sequence by Zigzag scan

Example

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

CSE-BUET

Example

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

Code of -26

It is Dc

Let the prev is -17

Diff is =-26 – (-17)= -9

Code of -9 is 1010110

CSE-BUET

Example

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

Code of -3

It is AC with 0 leading 0’s

Codes for AC Coeff.

Example: AC -3.

With 0 leading 0

Code?

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

CSE-BUET

JPEG Coefficient Coding Categories

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Codes for AC Coeff.

Example: AC -3.

With 2 leading 0

Code: 01 = 2 bits

Concatenate 00

Final: 0100

CSE-BUET

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Example

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

Code of -1

It is AC with 5 leading 0

JPEG Coefficient Coding Categories

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Codes for AC Coeff.

Run/ Category	Base Code	Length
5/1	1111010	8
5/2	1111111001	12
5/3	1111111110011111	19
5/4	1111111110100000	20
5/5	1111111110100001	21
5/6	1111111110100010	22
5/7	1111111110100011	23
5/8	1111111110100100	24
5/9	1111111110100101	25
5/A	1111111110100110	26

Example: AC -1.

With 5 leading 0

Code: 1111010 = 7 bits

Codes for AC Coeff.

Run/ Category	Base Code	Length
5/1	1111010	8
5/2	1111111001	12
5/3	1111111110011111	19
5/4	1111111110100000	20
5/5	1111111110100001	21
5/6	1111111110100010	22
5/7	1111111110100011	23
5/8	1111111110100100	24
5/9	1111111110100101	25
5/A	1111111110100110	26

Example: AC -1.

With 5 leading 0

Code: 1111010 = 7 bits

Concatenate 1 bit form -1-1 = -2 = 0_b

Codes for AC Coeff.

Run/ Category	Base Code	Length
5/1	1111010	8
5/2	1111111001	12
5/3	1111111110011111	19
5/4	1111111110100000	20
5/5	1111111110100001	21
5/6	1111111110100010	22
5/7	1111111110100011	23
5/8	1111111110100100	24
5/9	1111111110100101	25
5/A	1111111110100110	26

Example: AC -1.

With 5 leading 0

Code: 1111010 = 7 bits

Concatenate 1 bit form -1-1 = -2 = 0_b

Final: 11110100

Example

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

Code of EOB

Codes for AC Coeff.

Example: EOB

Code: 1010 = 4 bits

CSE-BUET

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

Special case

-26 -3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 EOB

16 zero's

Code of 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0?

CSE-BUET

Special case

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
5/1	1111010	8	D/1	11111111010	12
5/2	1111111001	12	D/2	1111111111100011	18
5/3	1111111110011111	19	D/3	1111111111100100	19
5/4	1111111110100000	20	D/4	1111111111100101	20
5/5	1111111110100001	21	D/5	1111111111100110	21
5/6	1111111110100010	22	D/6	1111111111100111	22
5/7	1111111110100011	23	D/7	1111111111101000	23
5/8	1111111110100100	24	D/8	1111111111101001	24
5/9	1111111110100101	25	D/9	1111111111101010	25
5/A	1111111110100110	26	D/A	1111111111101011	26
6/1	1111011	8	E/1	111111110110	13
6/2	11111111000	13	E/2	1111111111101100	18
6/3	1111111110100111	19	E/3	1111111111101101	19
6/4	1111111110101000	20	E/4	1111111111101110	20
6/5	1111111110101001	21	E/5	1111111111101111	21
6/6	1111111110101010	22	E/6	1111111111110000	22
6/7	1111111110101011	23	E/7	1111111111110001	23
6/8	1111111110101100	24	E/8	1111111111110010	24
6/9	1111111110101101	25	E/9	1111111111110011	25
6/A	1111111110101110	26	E/A	1111111111110100	26
7/1	11111001	9	**F/0**	**11111111011**	**12**
7/2	11111111001	13	F/1	1111111111110101	17
7/3	1111111110101111	19	F/2	1111111111110110	18
7/4	1111111110110000	20	F/3	1111111111110111	19
7/5	1111111110110001	21	F/4	1111111111111000	20
7/6	1111111110110010	22	F/5	1111111111111001	21
7/7	1111111110110011	23	F/6	1111111111111010	22
7/8	1111111110110100	24	F/7	1111111111111011	23
7/9	1111111110110101	25	F/8	1111111111111100	24
7/A	1111111110110110	26	F/9	1111111111111101	25
			F/A	1111111111111110	26

CSE-BUET

Example

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

Final bit sequence

Example: find the Inverse

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

Multiply them by Z(u, v)

Example: find the Inverse

-416	-33	-60	32	48	0	0	0
12	-24	-56	0	0	0	0	0
-42	13	80	-24	-40	0	0	0
-56	17	44	-29	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

After Multiplying by Z(u, v)

Then 1) find IDCT

2) Level shift by adding 2^{7+1}

Example

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

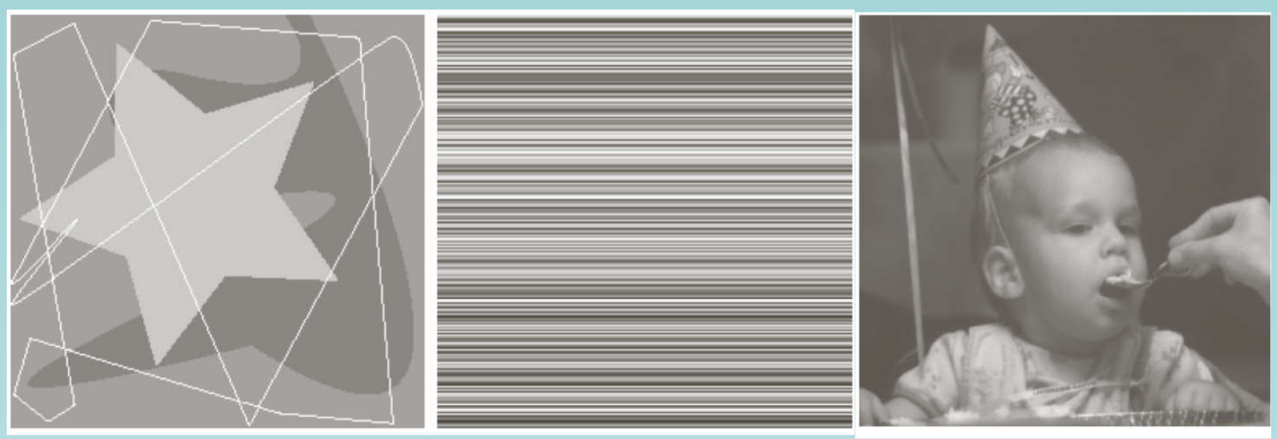
After level shifting

Example

−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	11	4	3	−2
3	8	4	−4	2	11	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

Difference Image

Predictive Coding



There are similarities between adjacent pixels

CSE-BUET

Predictive Coding

Predictive coding:

encodes the *difference* instead of the *true gray values*

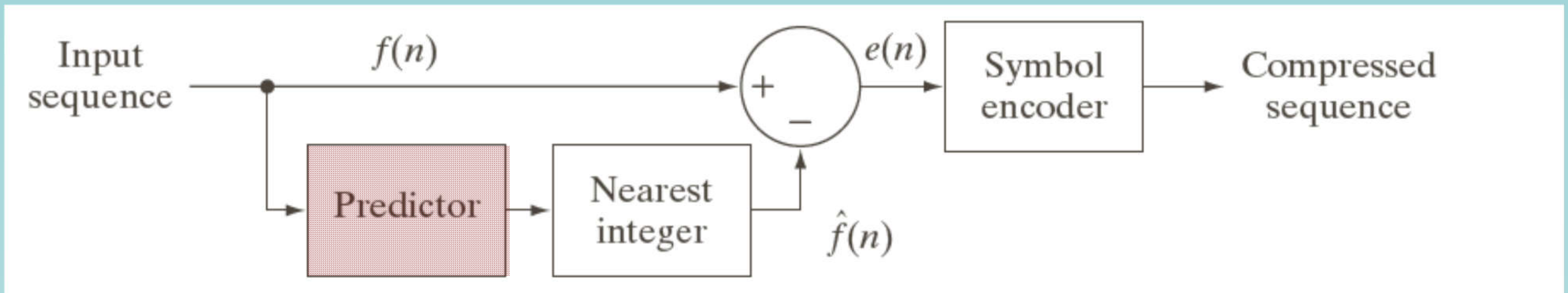
CSE-BUET

Predictive Coding

2 types:

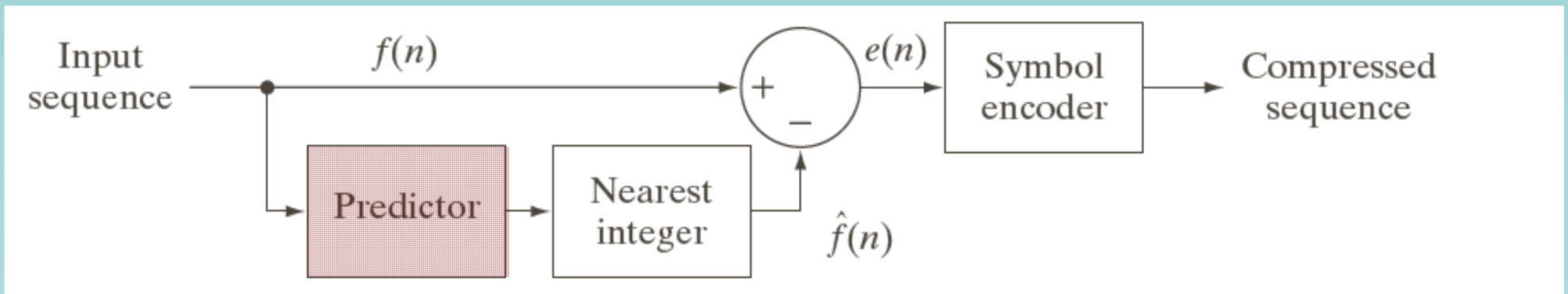
- Loss less
- Lossy

Lossless Predictive Coding



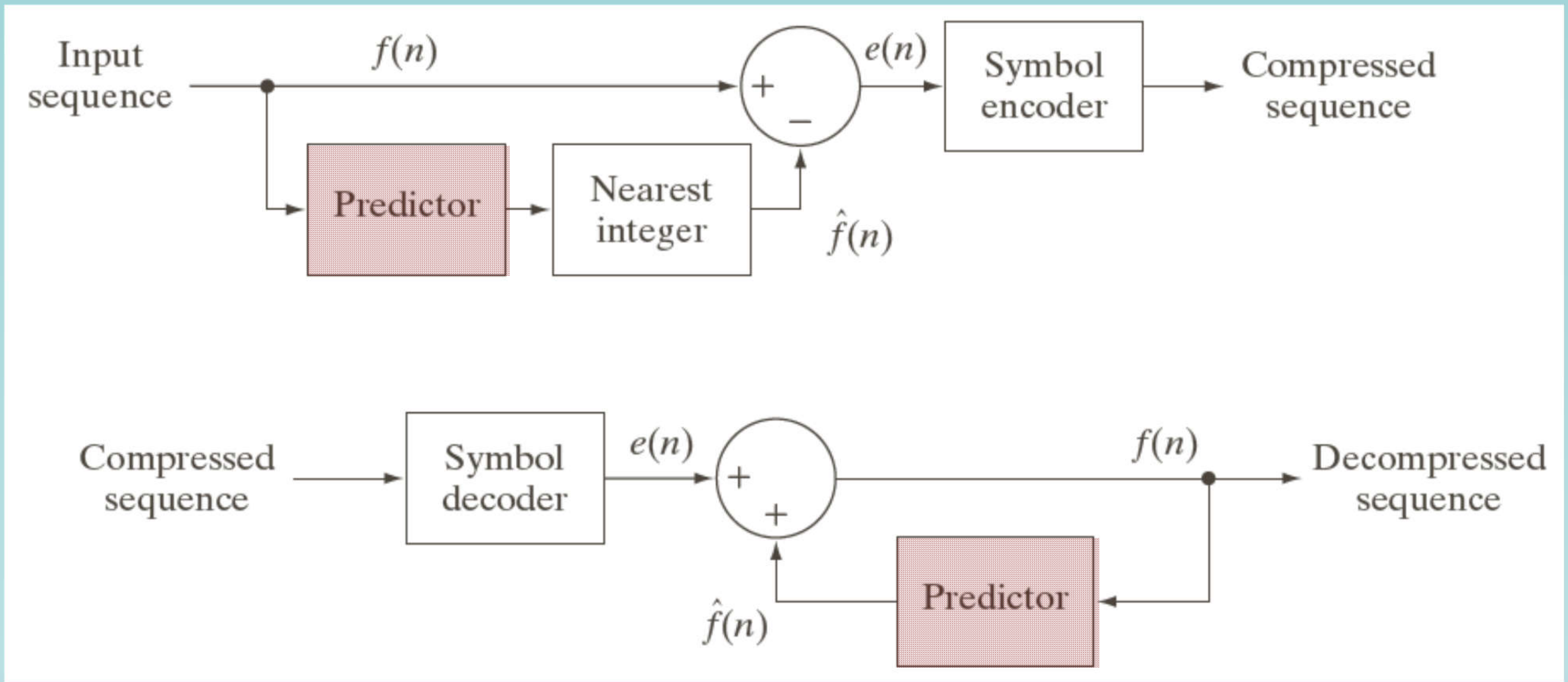
CSE-BUET

Lossless Predictive Coding



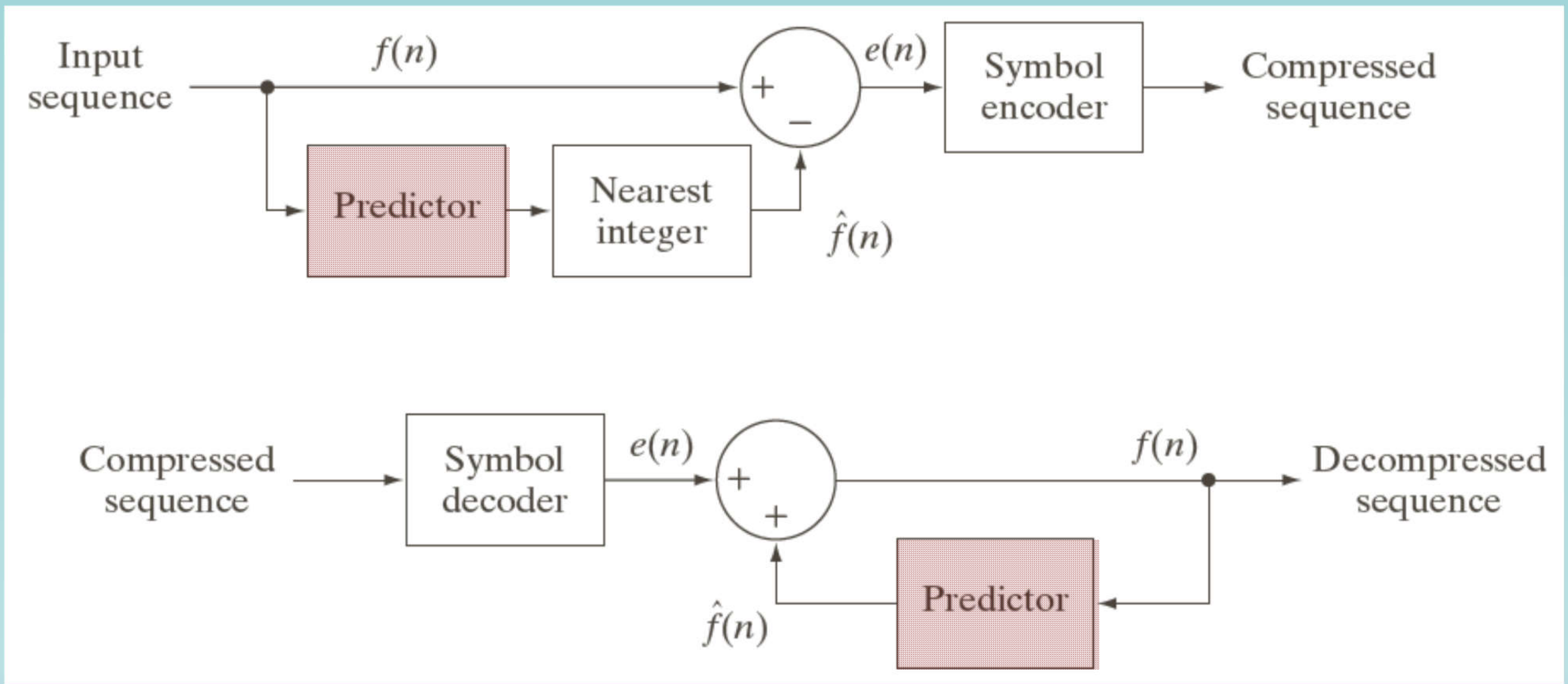
$$e(n) = f(n) - \hat{f}(n)$$

Lossless Predictive Coding



CSE-BUET

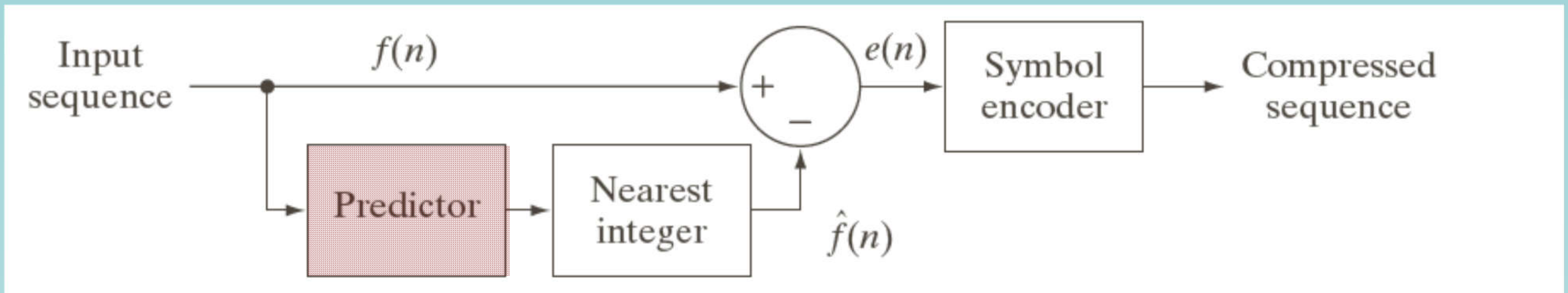
Lossless Predictive Coding



$$f(n) = e(n) + \hat{f}(n)$$

CSE-BUET

Lossless Predictive Coding



Predictor:

$$\hat{f}(n) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

CSE-BUET

Lossless Predictive Coding

$$\hat{f}(n) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

Prediction function:

depends on *m* previous values that come from

- current scan line (*1D linear prediction*)
- current and previous scan line (*2D linear prediction*)
- current image and previous image (*3D linear prediction*)

Lossless Predictive Coding

1D linear prediction for image coding:

$$\hat{f}(x,y) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(x, y-i)\right]$$

CSE-BUET

Lossless Predictive Coding: A Simplified Example

$$\hat{f}(x, y) = \text{round}\left[\alpha f(x, y-1)\right]$$

- depends only on the previous pixel
- predictor is known as previous pixel predictor

Lossless Predictive Coding: A Simplified Example

$$\hat{f}(x, y) = \text{round}\left[\alpha f(x, y-1)\right]$$

- depends only on the previous pixel
- predictor is known as previous pixel predictor
- coding is known as
  - previous pixel coding
  - differential coding

Lossless Predictive Coding: A Simplified Example

$$\hat{f}(x,y) = \text{round}\left[\alpha f(x,y-1)\right]$$

Original Image

prediction error

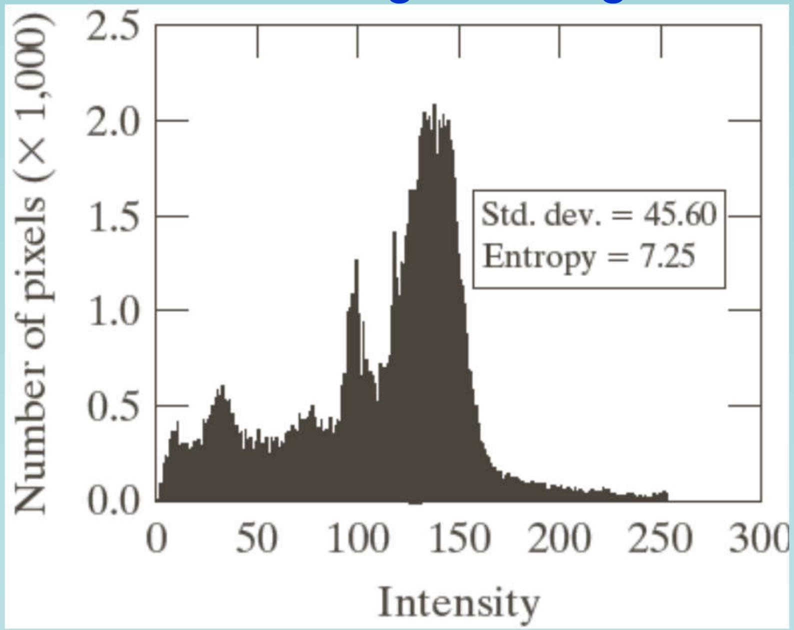
$$e(x,y) = f(x,y) - \hat{f}(x,y)$$

assuming $\alpha = 1$ in

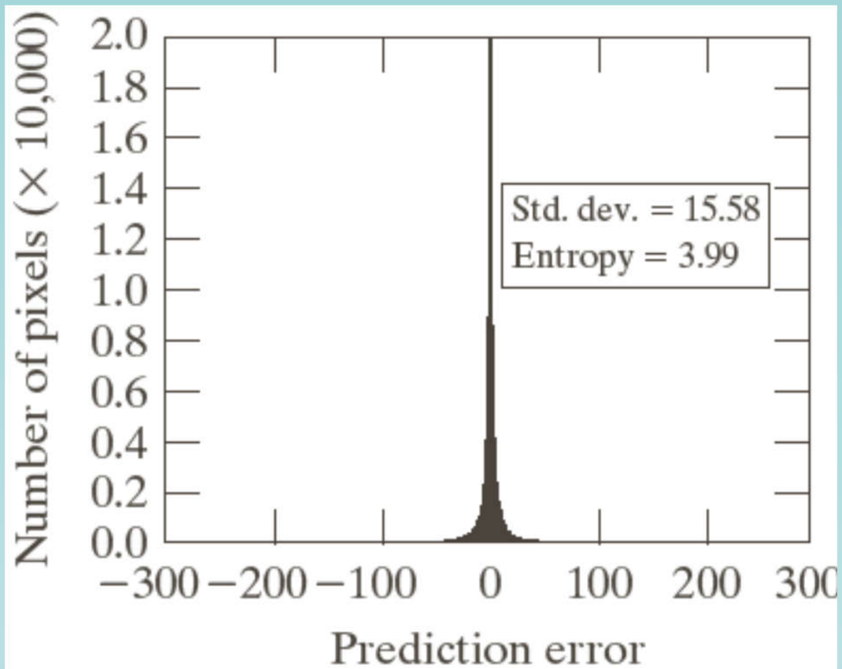
$$\hat{f}(x,y) = \text{round}\left[\alpha f(x,y-1)\right]$$

Lossless Predictive Coding: A Simplified Example

For original image



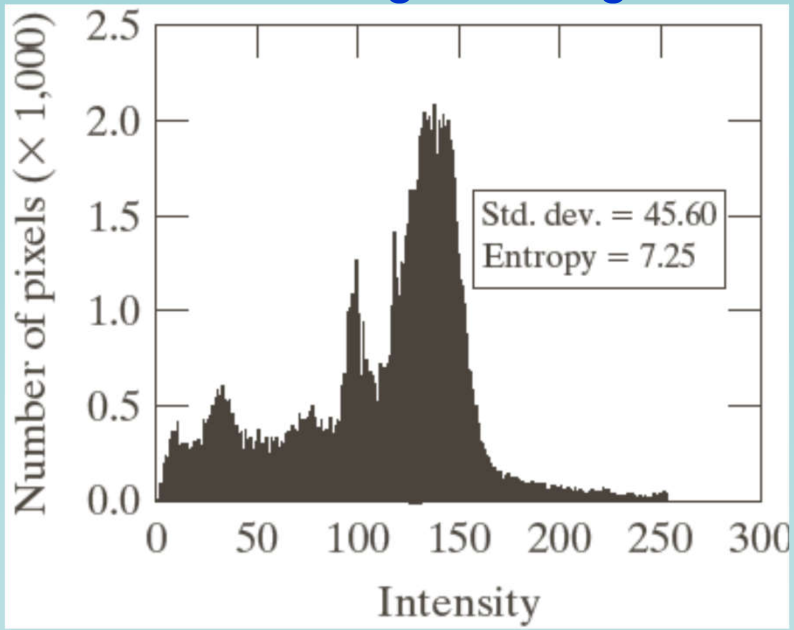
For error image



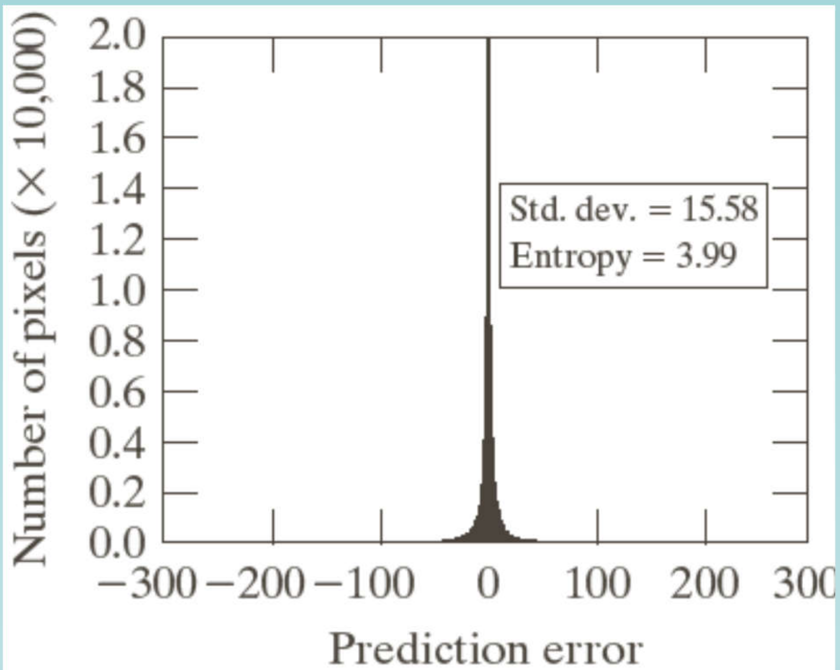
CSE-BUET

Lossless Predictive Coding: A Simplified Example

For original image



For error image



Compression Ratio = 8/3.99 = 2:1

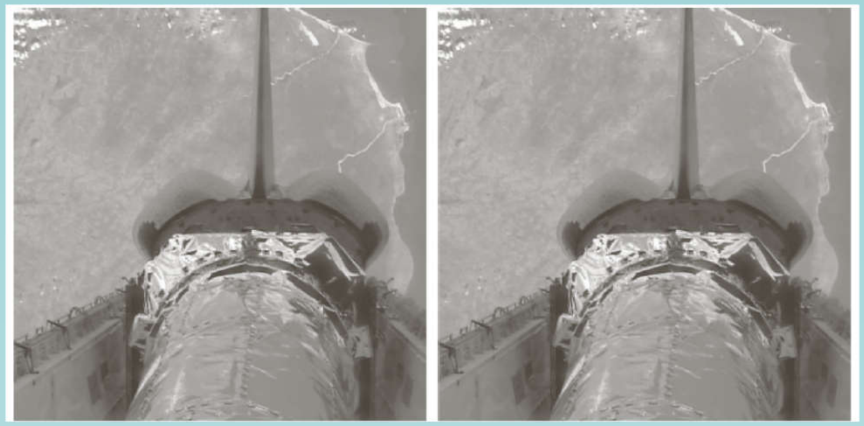
Lossless Predictive Coding: for Video Sequence

$$\hat{f}(x,y,t) = \text{round}\left[\alpha f(x,y,t-1)\right]$$

Lossless Predictive Coding for Video : Example 2

same image of the previous example

Lossless Predictive Coding for Video : Example 2

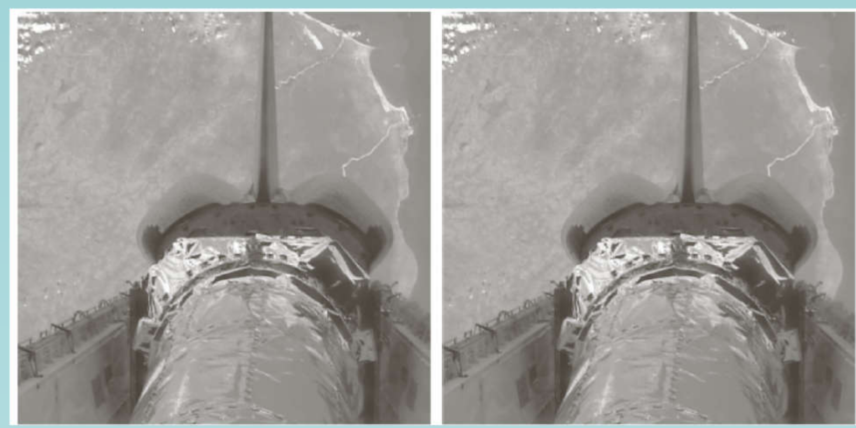


immediate preceding image

same image of the previous example

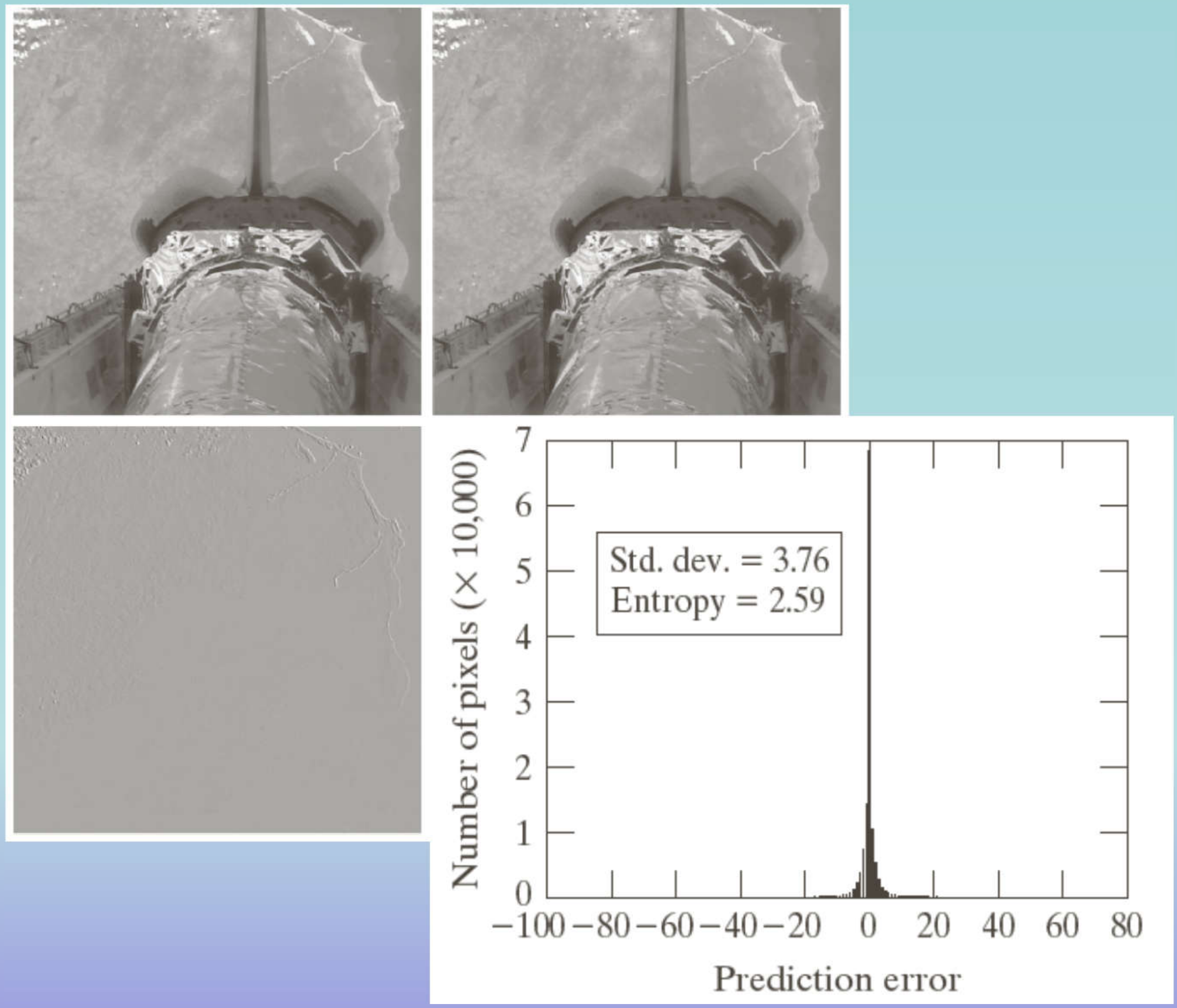
CSE-BUET

Lossless Predictive Coding for Video : Example 2



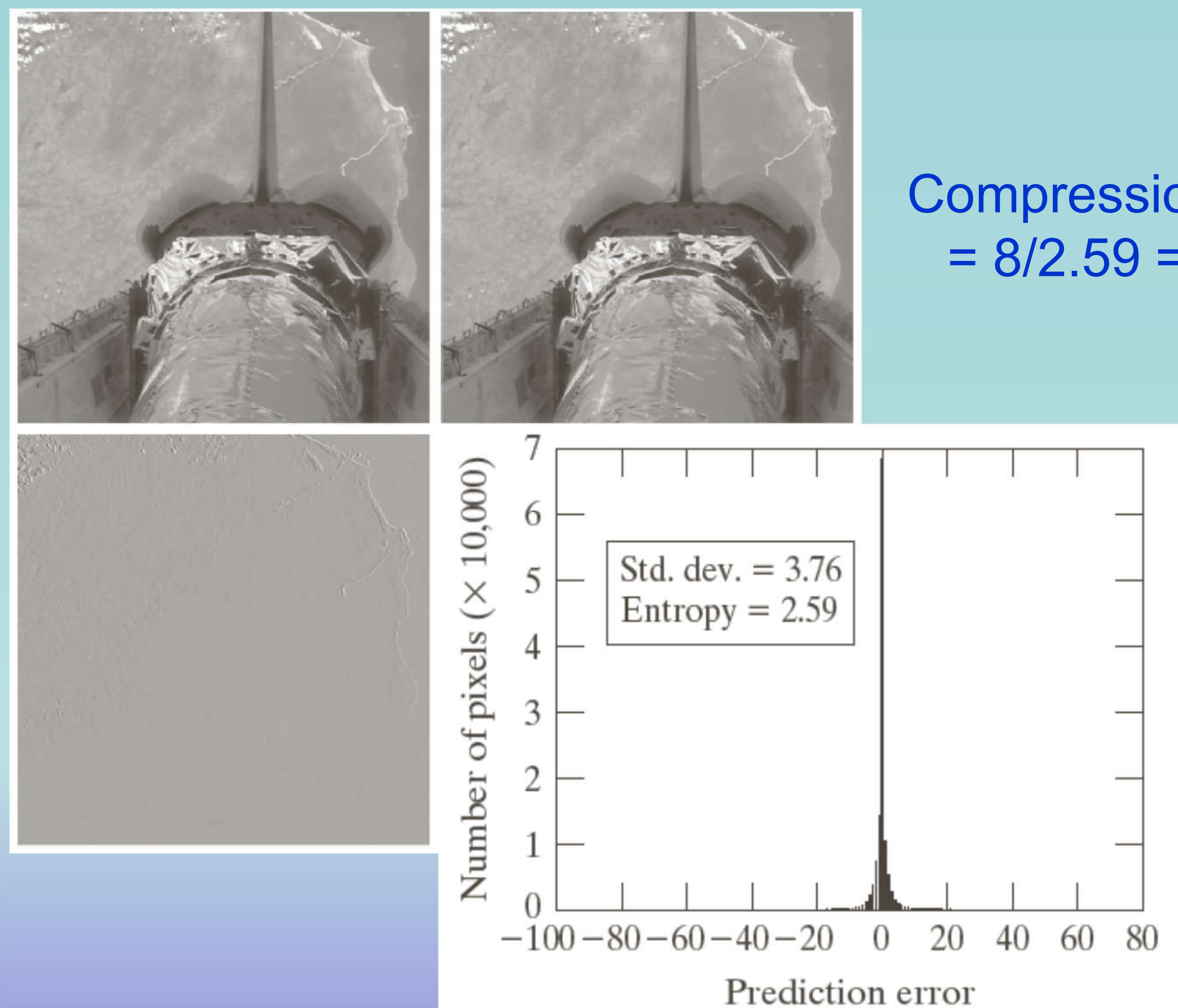
use this image	predict this image
(time t -1 image)	(time t image)

Lossless Predictive Coding for Video : Example 2



CSE-BUET

Lossless Predictive Coding for Video : Example 2

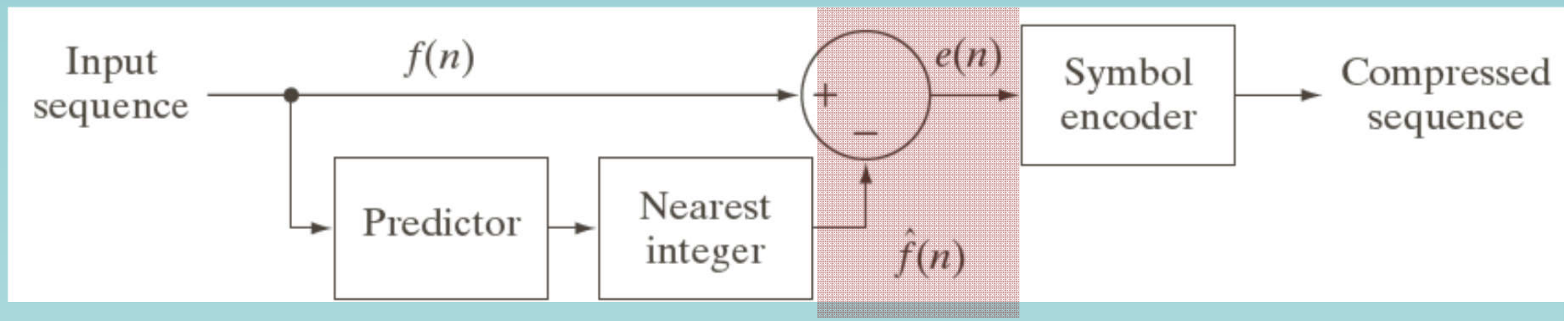


Compression Ratio
= 8/2.59 = 3.2:1

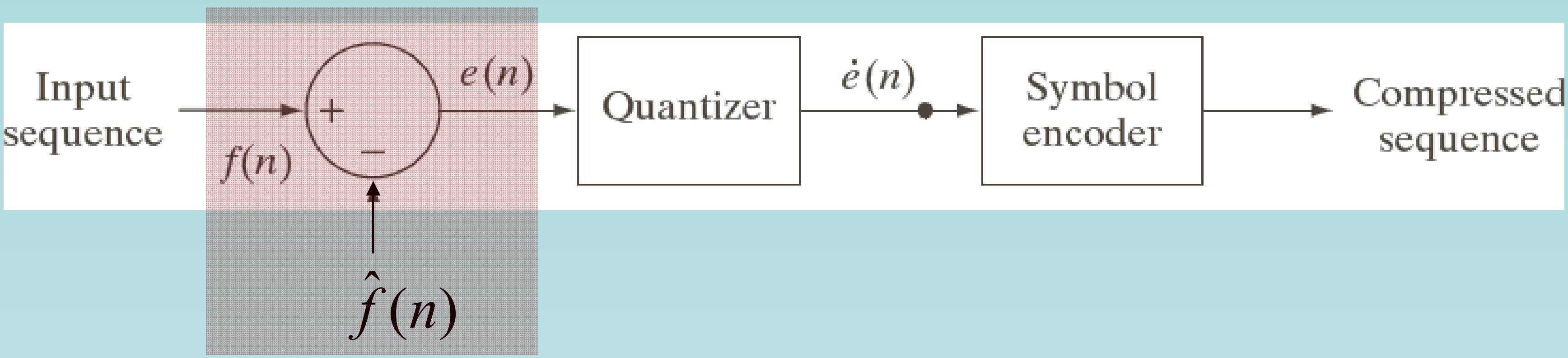
CSE-BUET

Lossy Predictive Coding

Lossless



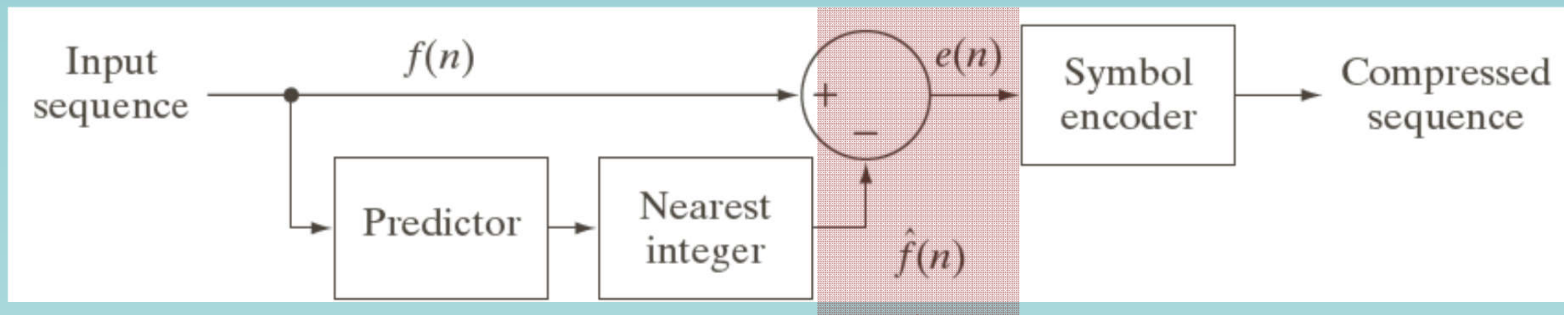
Lossy



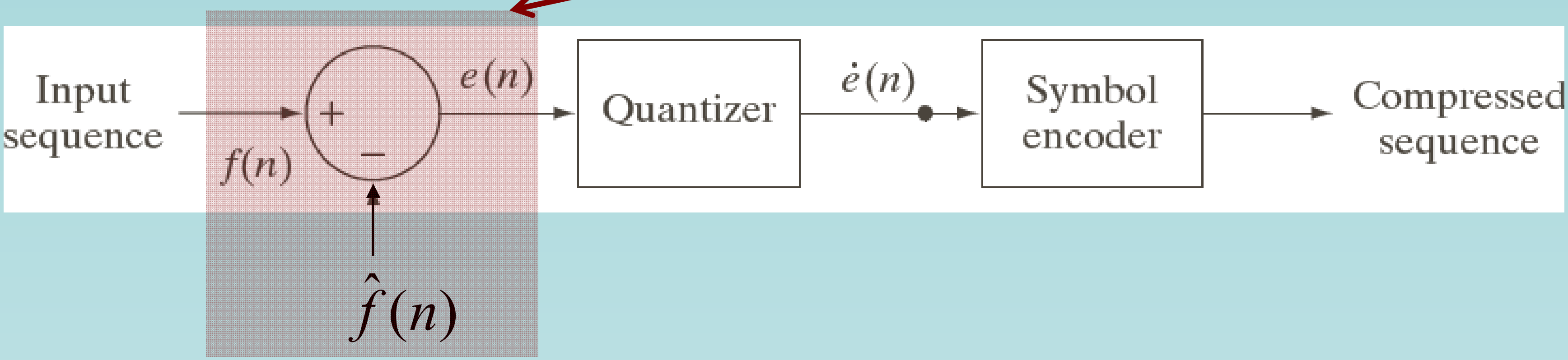
CSE-BUET

Lossy Predictive Coding

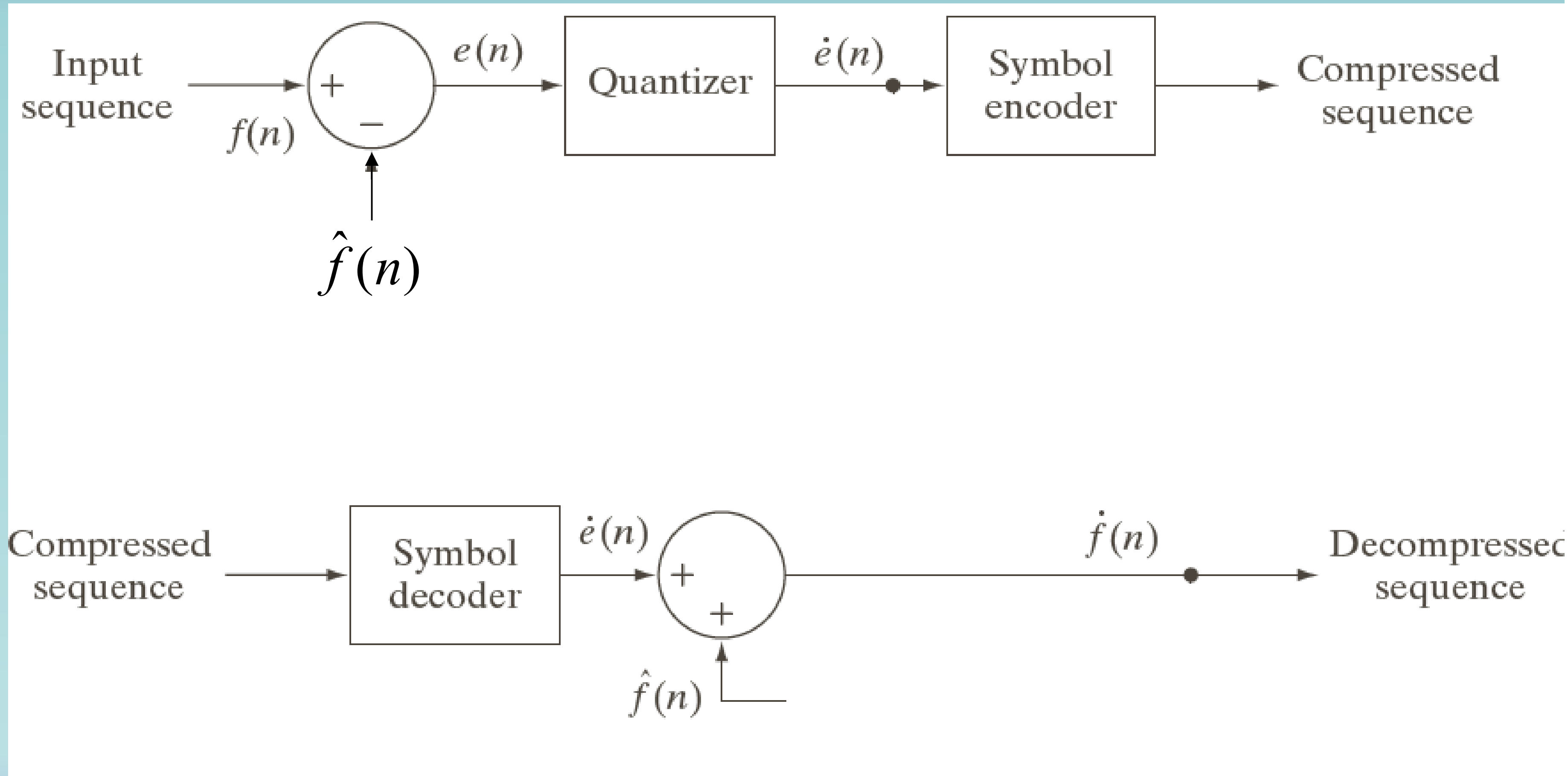
Lossless



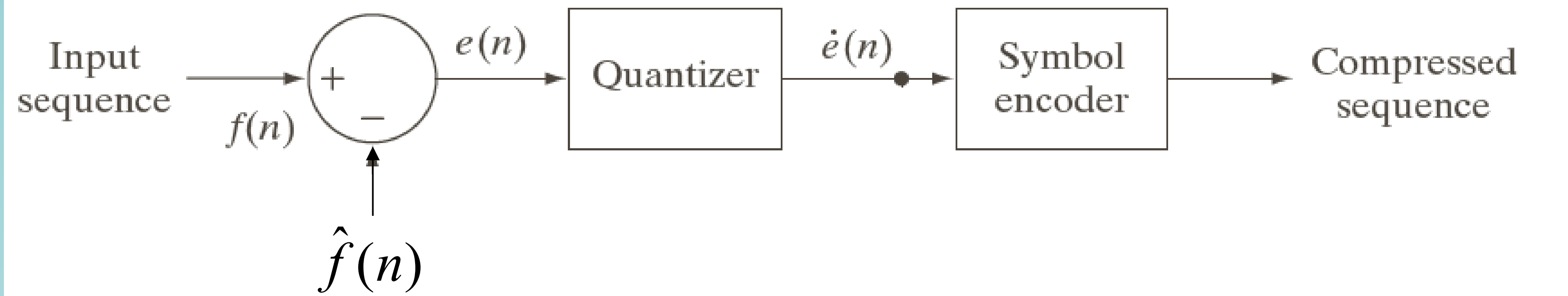
Lossy



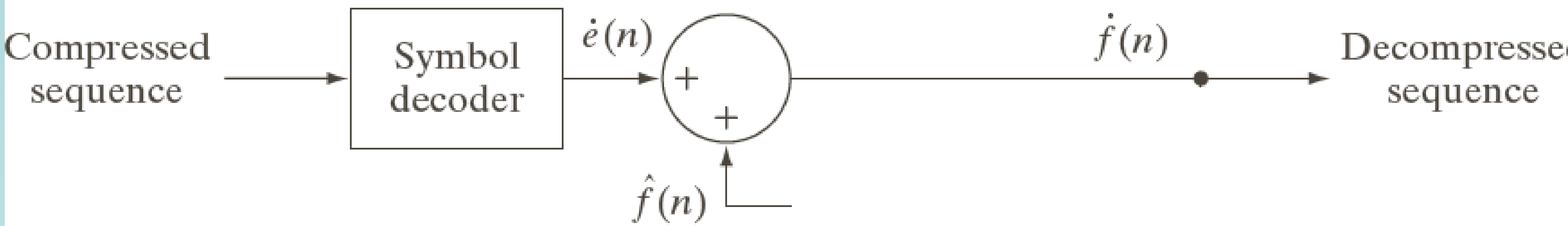
Lossy Predictive Coding



Lossy Predictive Coding

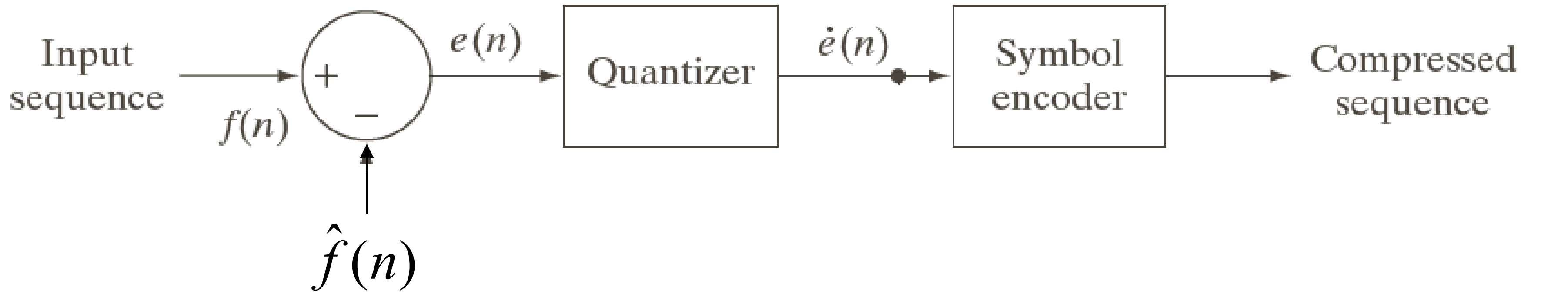


$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

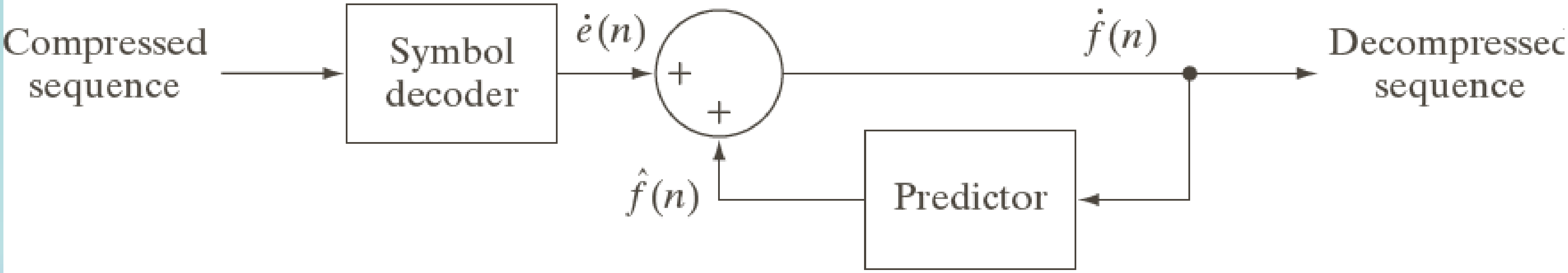


CSE-BUET

Lossy Predictive Coding

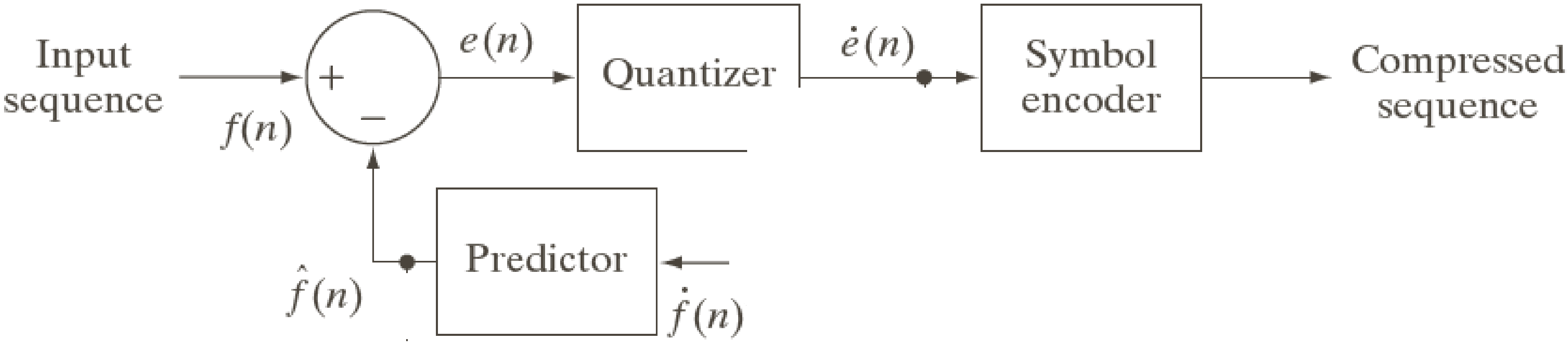


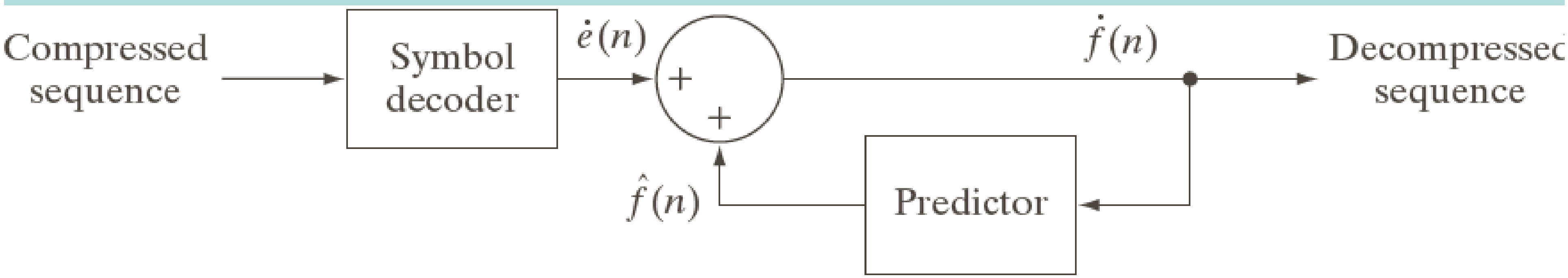
$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$



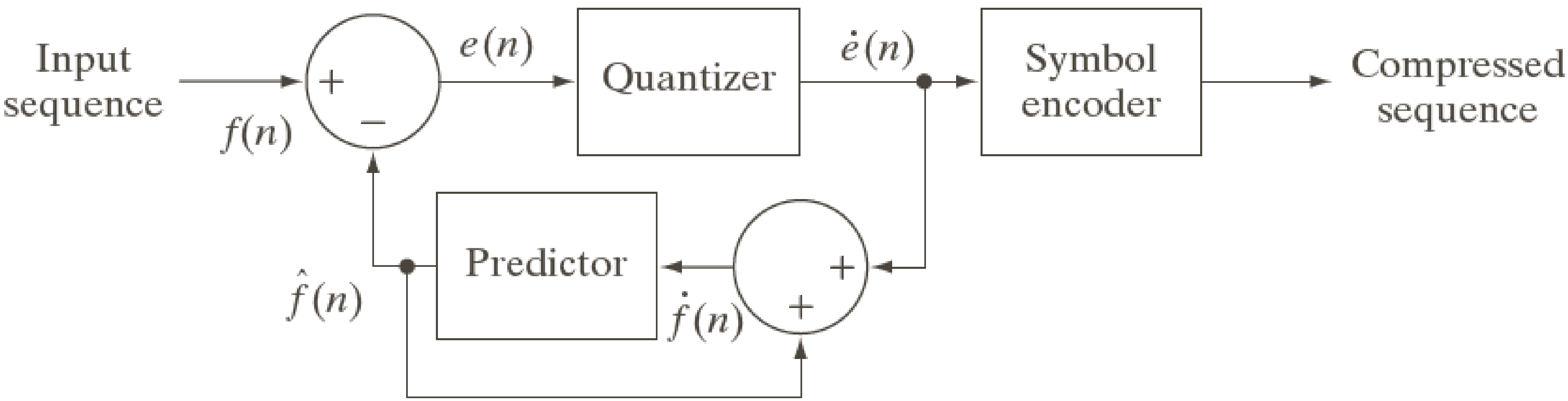
CSE-BUET

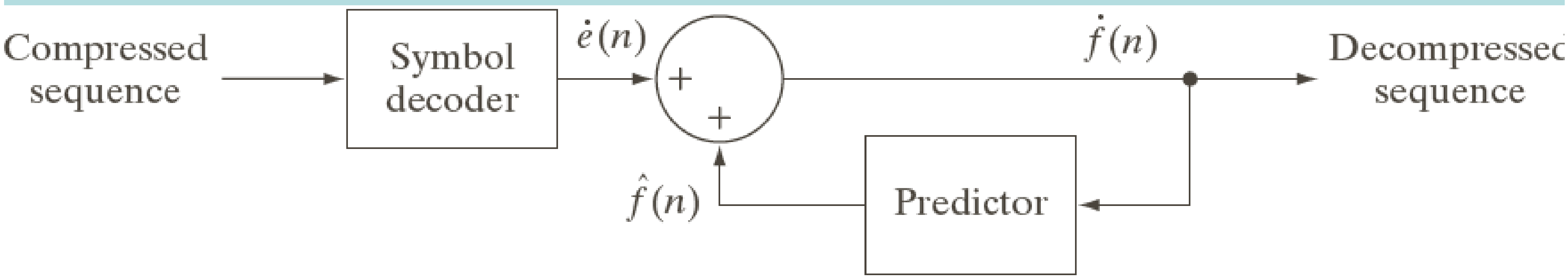
Lossy Predictive Coding



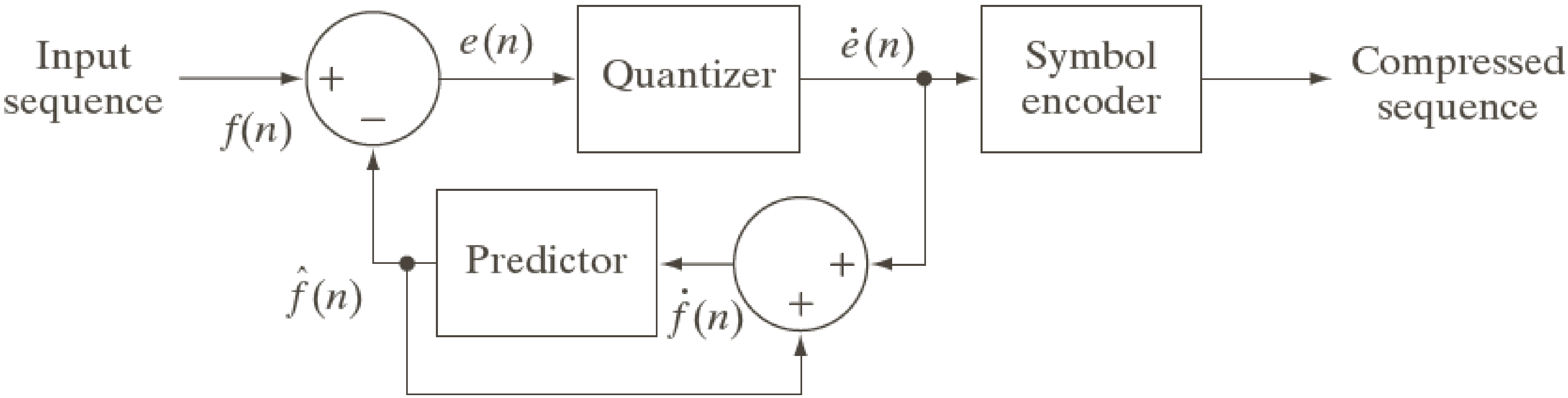


Lossy Predictive Coding





Lossy Predictive Coding: Delta Modulation



$$\hat{f}(n) = \alpha\dot{f}(n-1)$$

and

$$\dot{e}(n) = \begin{cases} +\xi & \text{for } e(n) > 0 \\ -\xi & \text{otherwise} \end{cases}$$

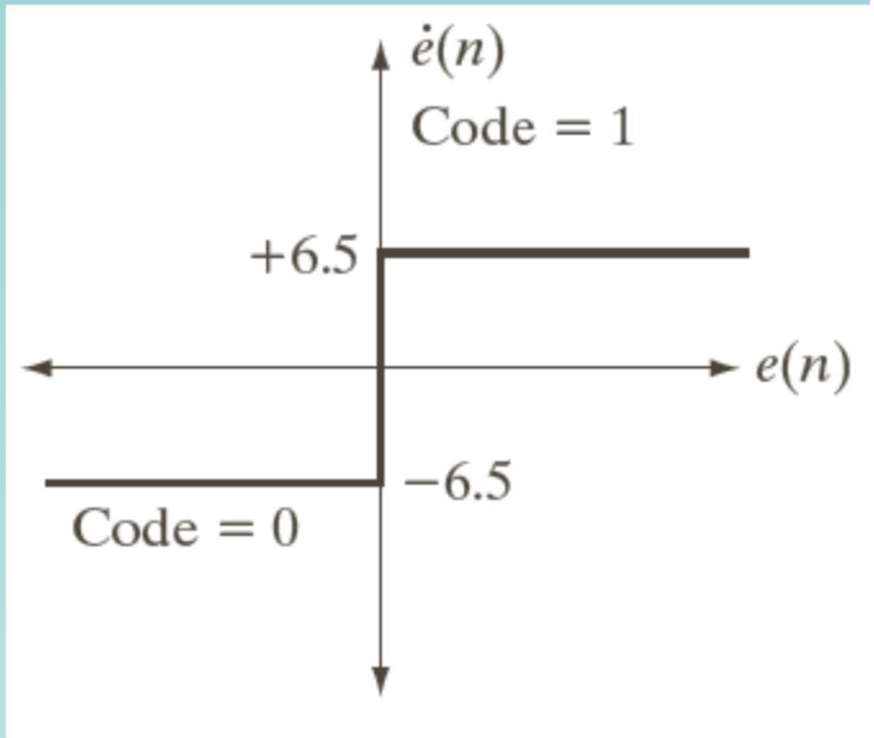
in addition to

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

CSE-BUET

Lossy Predictive Coding: Delta Modulation

$$\dot{e}(n) = \begin{cases} +\xi & \text{for } e(n) > 0 \\ -\xi & \text{otherwise} \end{cases}$$



Delta Modulation Coding: Example

Sequence to be coded:

{14, 15, 14, 15, 13, 15, 15, 14, 20, 26, 27, 28, 27, 27, 29, 37, 47, 62, 75, 77, 78, 79, 80, 81, 81, 82, 82}

Delta Modulation Coding: Example

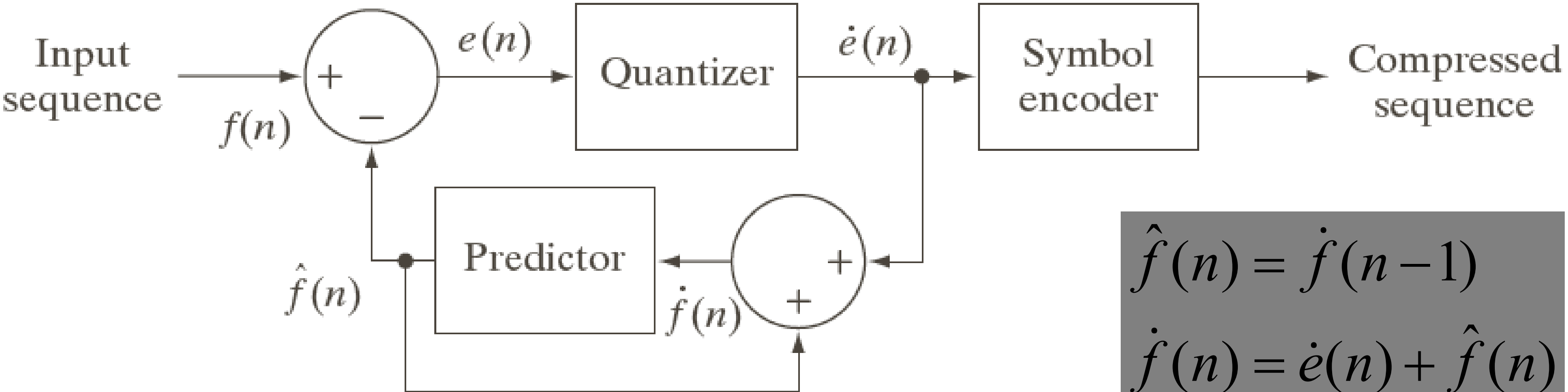
Sequence to be coded:

{14, 15, 14, 15, 13, 15, 15, 14, 20, 26, 27, 28, 27, 27, 29, 37, 47, 62, 75, 77, 78, 79, 80, 81, 81, 82, 82}

$$\hat{f}(n) = \alpha \dot{f}(n-1), \text{ assuming } \alpha = 1$$

CSE-BUET

Delta Modulation Coding: Example



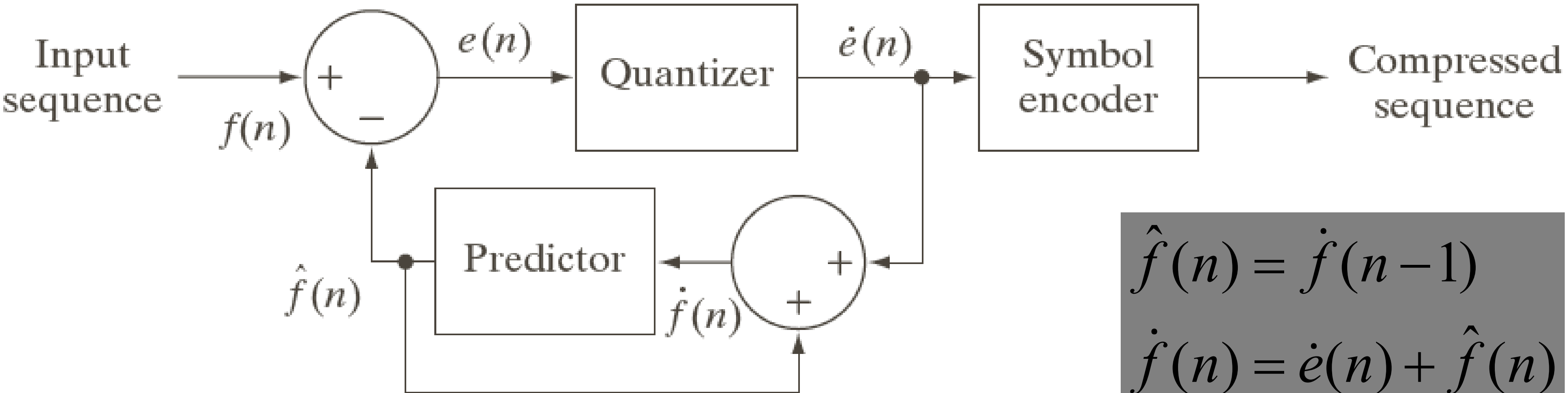
$$\hat{f}(n) = \dot{f}(n-1)$$

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0

CSE-BUET

Delta Modulation Coding: Example



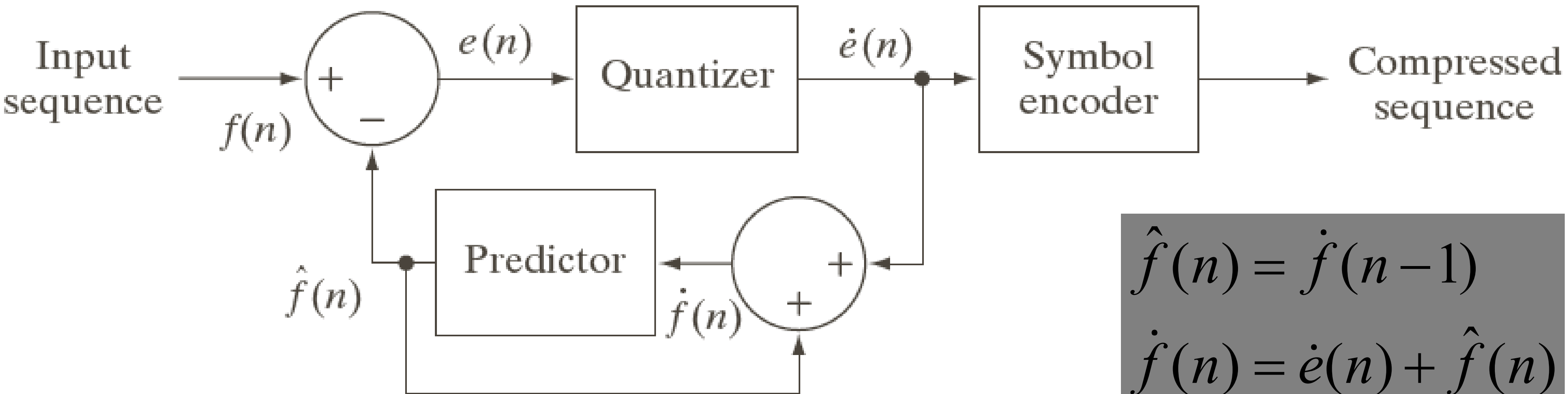
$$\hat{f}(n) = \dot{f}(n-1)$$

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5

CSE-BUET

Delta Modulation Coding: Example

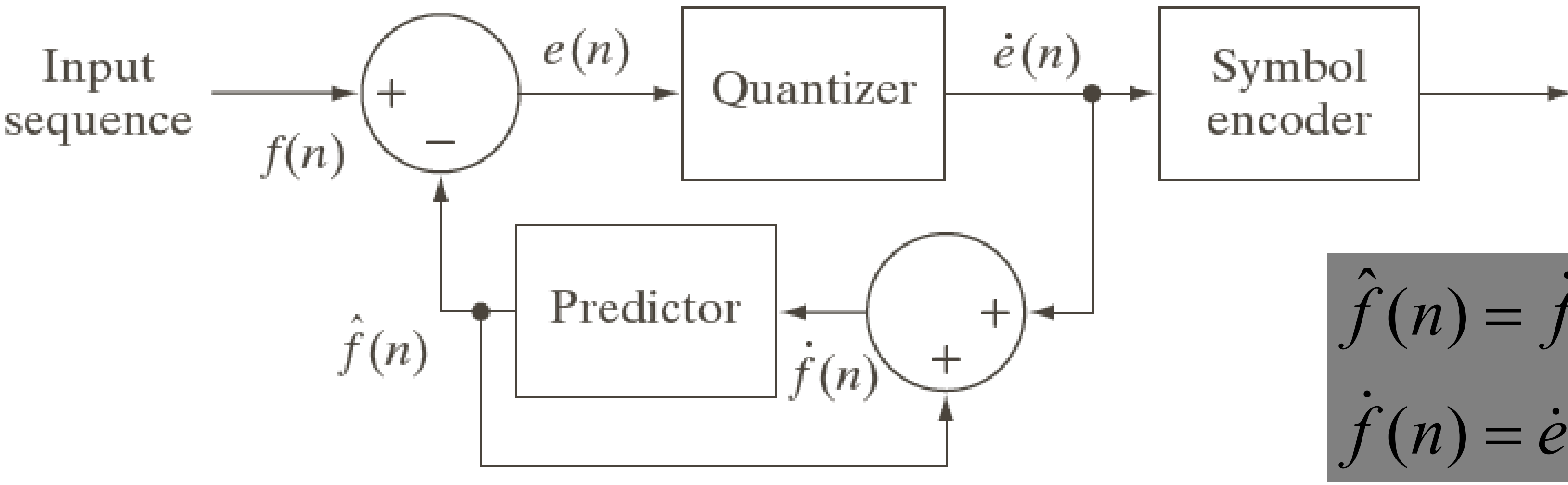


$$\hat{f}(n) = \dot{f}(n-1)$$

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0

Delta Modulation Coding: Example



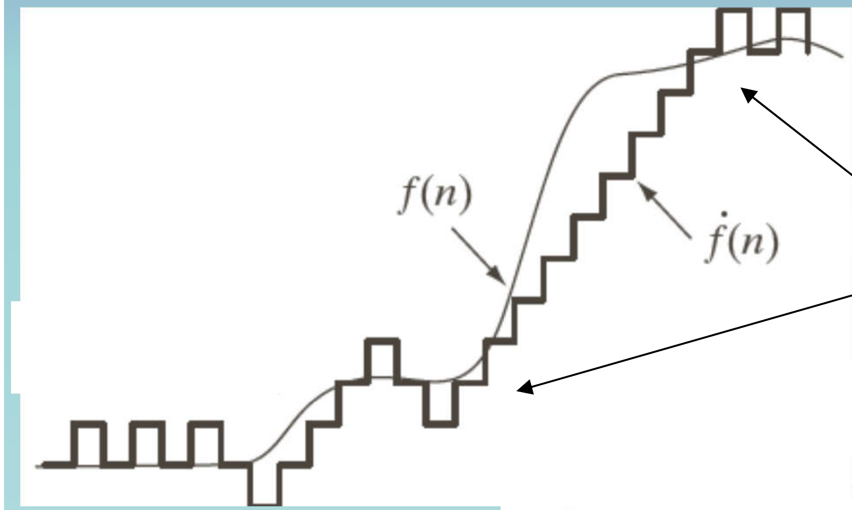
$$\hat{f}(n) = \dot{f}(n-1)$$

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.

CSE-BUET

Delta Modulation Coding: Example

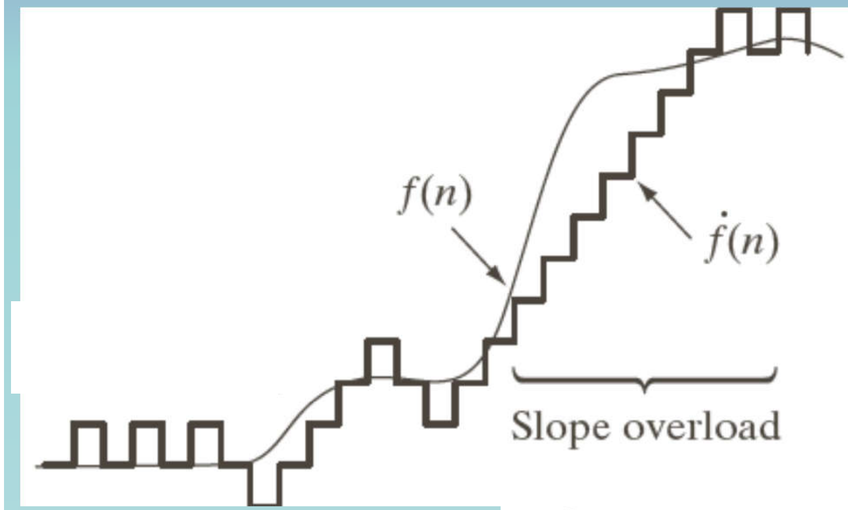


ξ is too small to represent the largest changes in data

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.

CSE-BUET

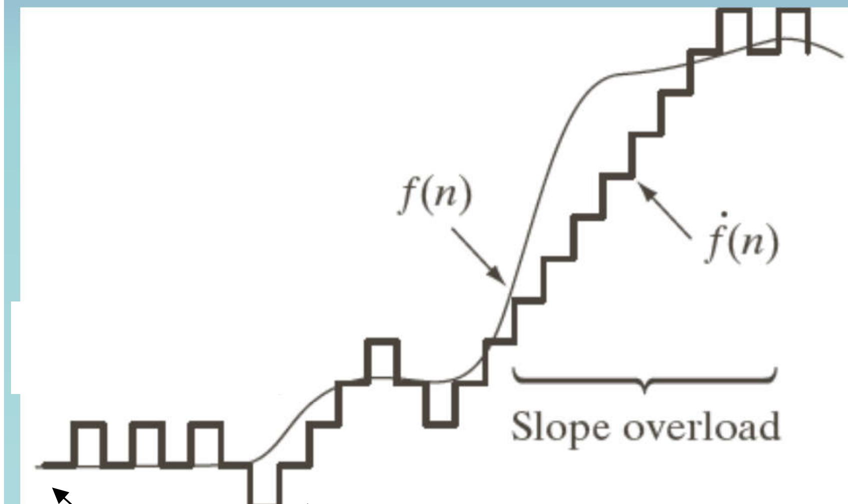
Delta Modulation Coding: Example



Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.

CSE-BUET

Delta Modulation Coding: Example

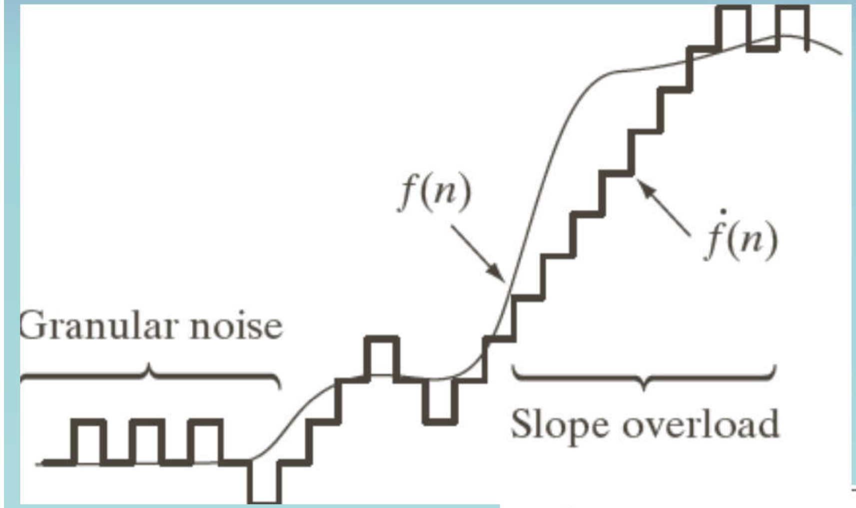


ξ is too large to represent the smallest changes in data

Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.

CSE-BUET

Delta Modulation Coding: Example



Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n) - \dot{f}(n)$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
2	14	20.5	−6.5	−6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	−5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.

CSE-BUET