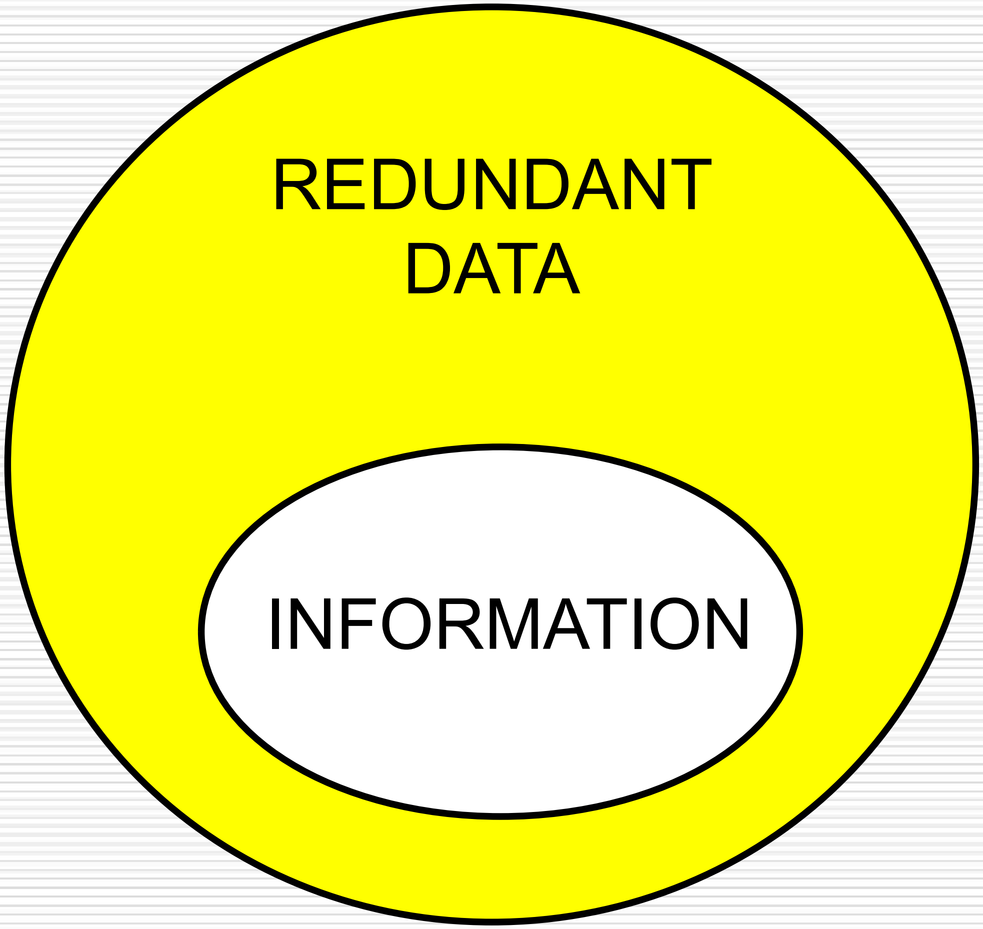
Image Compression

Anjali Malviya
Dept. of IT
TSEC

Syllabus

Image Compression	Entropy, Redundancy and Types, Compression Ratio, Compression Methods. **Lossless Compression:** Run-Length Encoding, Huffman Coding, Arithmetic Coding, LZW Coding, Lossless Predictive coding. **Lossy Compression**: Fidelity Criterion, Improved Gray scale Quantization, Symbol-Based Coding, Bit-Plane Coding, Vector Quantization. **Self-Learning Topics:** DPCM, Block Transform Coding, JPEG compression.

Information vs Data



DATA = INFORMATION + REDUNDANT DATA

Data & Information

- Data are the *means* by which information is conveyed
- Various amounts of data may be used to represent the same amount of information
- Data redundancy: if n_1 and n_2 denote the number of *information-carrying units* in two data sets that represent the same information, the *relative data redundancy* of the first data set with respect to the second is

$$R_D = 1 - \frac{1}{C_R} \quad \text{where} \quad C_R = \frac{n_1}{n_2}$$

compression ratio

- Case 1: $n_2 = n_1$

$$C_R = 1 \text{ and } R_D = 0$$

the first data set contains no redundant information

- Case 2: $n_2 \ll n_1$

$$C_R \to \infty \text{ and } R_D \to 1$$

highly redundant data significant compression

- Case 3: $n_2 \gg n_1$

$$C_R \to 0 \text{ and } R_D \to -\infty$$

(undesirable) data expansion

Redundancy

- $$C_R \in (0, \infty) \text{ and } R_D \in (-\infty, 1)$$

- In digital image compression there exist three basic data redundancies:

1. Coding redundancy
2. Interpixel redundancy
3. Psychovisual redundancy

Example

Suppose that: $C_R = 10 \quad (\text{or } 10:1)$

- This means that the first dataset needs 10 information-carrying units (e.g., bits) whereas the second dataset just needs 1!

$$R_D = 0.9$$

- The corresponding redundancy is

which says that 90% of the data in the first set is redundant

Coding Redundancy

- Let $r_k \in [0,1]$ be a discrete random variable representing the gray levels (L) in an image
- Its probability is represented by

$$p_r(r_k) = \frac{n_k}{n}, \quad k = 0,1,\ldots,L-1$$

- Let $l(r_k)$ be the total number of bits used to represent each value of r_k
- The average number of bits required to represent each pixel is

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

- The average lenght of the code words assigned to the various gray-level values: the sum of the product of the number of bits used to represent each gray level and the probability that the gray level occurs.

- Total number of bits to code an MxN image is MNL_{avg}

- Usually $l(r_k) = m$ bits (constant). $\Rightarrow L_{avg} = \sum_k m p_r(r_k) = m$

Coding

- It makes sense to assign fewer bits to those r_k for which $p_r(r_k)$ are large in order to reduce the sum.
- $\Rightarrow$ achieves data compression and results in a variable length code.
- More probable gray levels will have fewer number of bits.

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

Example-Variable length Coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Gray-level distribution

Code 1: $L_{avg} = 3 bits$

Code 2: $L_{avg} = 2.7 bits$

$2(0.19)+2(0.25)+...$

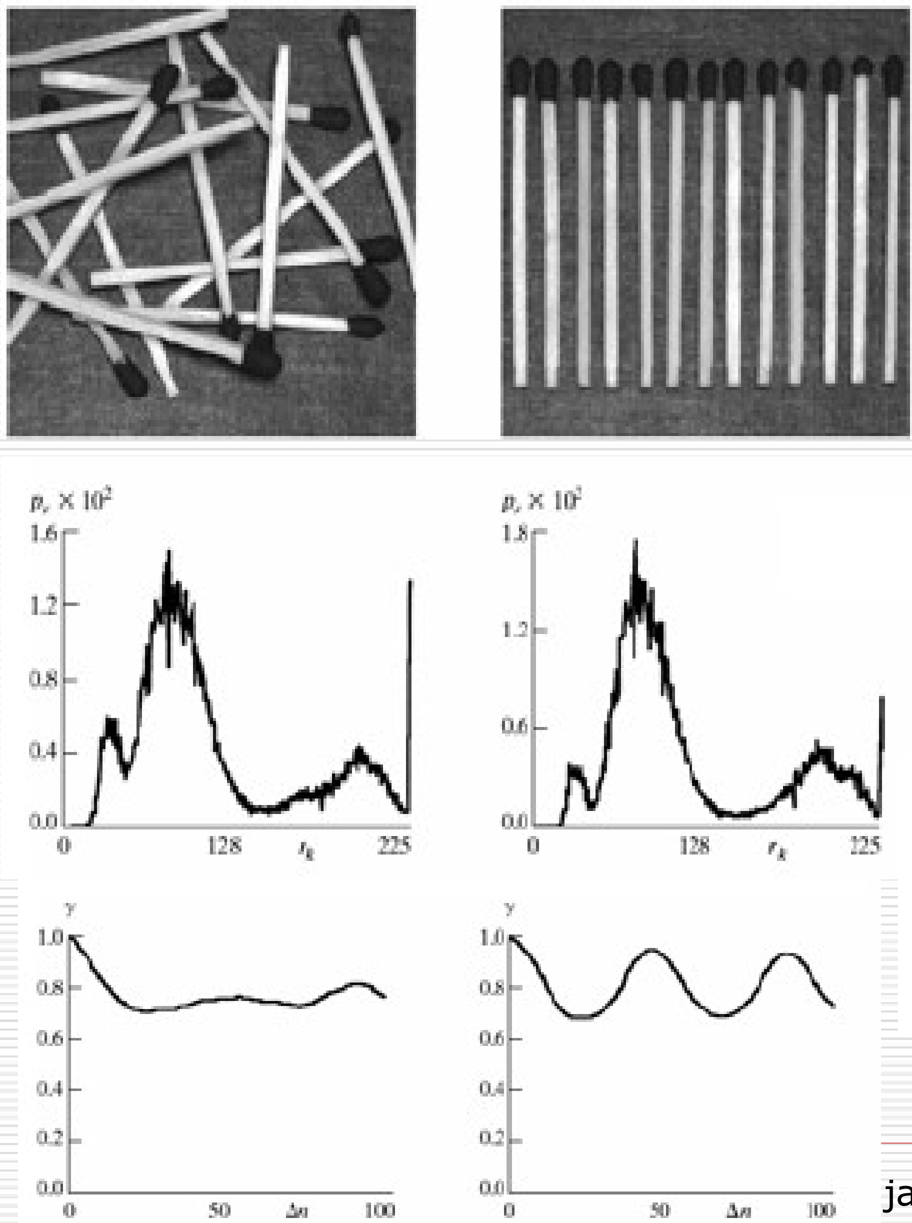
$$C_R = \frac{3}{2.7} = 1.11$$

$$\boxed{R_D = 1 - \frac{1}{1.11} = 0.099}$$

Rationale behind Variable Length Coding

- Assign shortest codewords to the most probable gray levels and longest codewords to the least probable

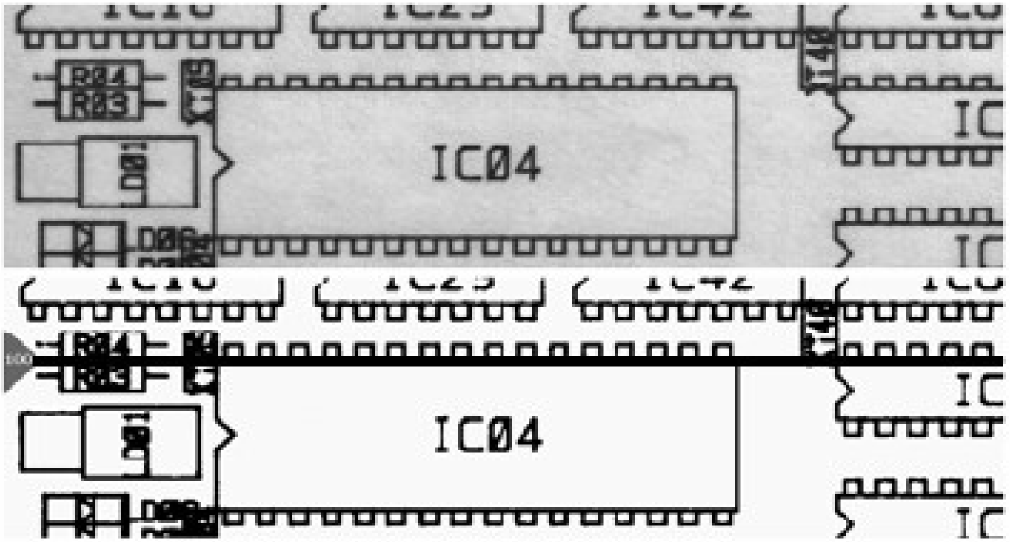
Interpixel Redundancy

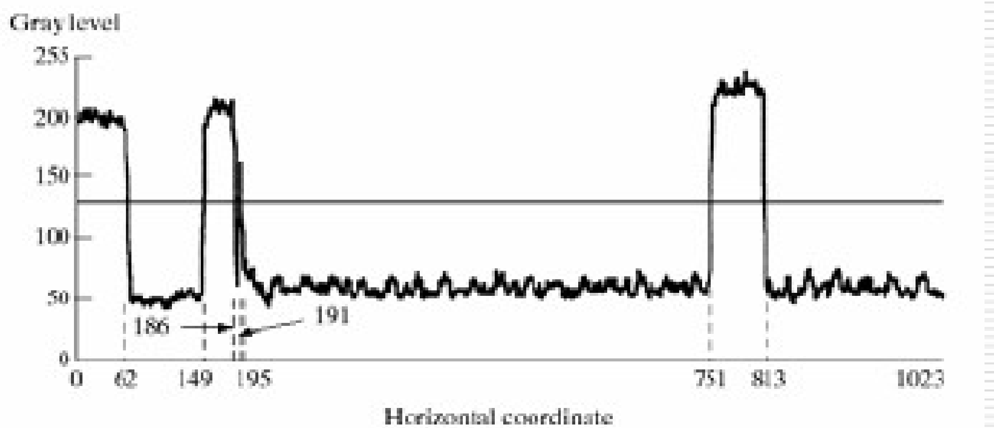


autocorrelation coefficients along one line of each image

- Second image shows high correlation between pixels 45 and 90 samples apart
- Adjacent pixels of both images are highly correlated
- Interpixel (or spatial) redundancy: the value of any given pixel can be reasonably predicted from the values of its neighbors; as a consequence, any pixel carries a small amount of information
- Interpixel redundancy can be reduced through mappings (e.g., differences between adjacent pixels)

Example:Run-length Coding





Line 100: (1, 63) (0, 87) (1, 37) (0, 5) (1, 4) (0, 556) (1, 62) (0, 210)

$$C_R = \frac{1024 \cdot 343 \cdot 1}{12{,}166 \cdot 11} = 2.63$$

number of runs

11 bits are necessary to represent each run-length pair

$$R_D = 1 - \frac{1}{2.63} = 0.62$$

Psychovisual Redundancy

- The eye does not respond with equal sensitivity to all visual information
- Certain information has less relative importance than other information in normal visual processing (*psychovisually redundant*)
- It can be eliminated without significantly impairing the quality of image perception

Compression by Quantization

8 bits | 4 bits (2:1) | Improved Gray Scale (IGS) Quantization (2:1)

Pixel	Gray Level	Sum	IGS Code
$i-1$	N/A	0000 0000	N/A
i	0110 1100	0110 1100	0110
$i+1$	1000 1011	1001 0111	1001
$i+2$	1000 0111	1000 1110	1000
$i+3$	1111 0100	1111 0100	1111

IGS

Pixel	Gray Level	Sum	IGS Code
$i-1$	N/A	00000000	N/A
i	01101100	01101100	0110
$i+1$	10001011	10010111	1001
$i+2$	10000111	10001110	1000
$i+3$	11110100	11110100	1111

IGS Quantization Procedure

1. LSBs of previous “sum” are added to current pixel
2. New 4 MSBs of “sum” are taken as IGS code
3. Repeat steps 1 n 2 for next pixel
4. If MSB = 1111, add 0000 to current pixel, instead of LSBs of previous sum.

Exercise

- Construct the IGS code of given gray level data set:

{ 100, 110, 125,125, 130, 110, 200, 210}

Compute the e_{rms} and SNR_{rms} introduced.

e_{rms}= 5.85 SNR_{rms}=24.6471

Objective Fidelity Criteria

- $f(x, y)$ is the input image
- $\hat{f}(x, y)$ is the estimate or approximation of $f(x, y)$ resulting from compression and decompression

- Error between the two images

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

- Root-mean square error:

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

- $SNR_{rms} = \left[\dfrac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2} \right]^{1/2}$

Signal to Noise Ratio

- Mean-square SNR of the output image

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

Subjective Fidelity Criteria

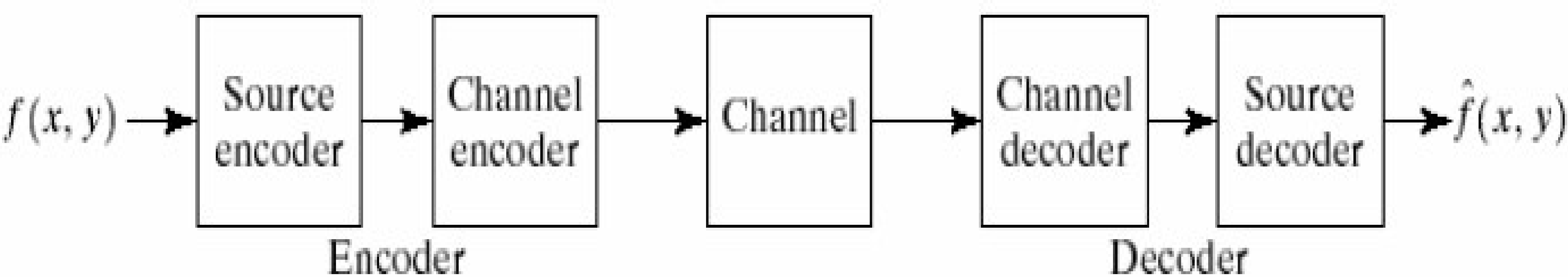
Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Ex.2

- Consider an 8-pixel line of gray scale data[12,12,13,13,10,13,57,54] which has been uniformly quantized with 6-bit accuracy. Construct its 3-bit IGS code.
- Compute the rms error and rms signal to noise ratio for the decoded IGS code.

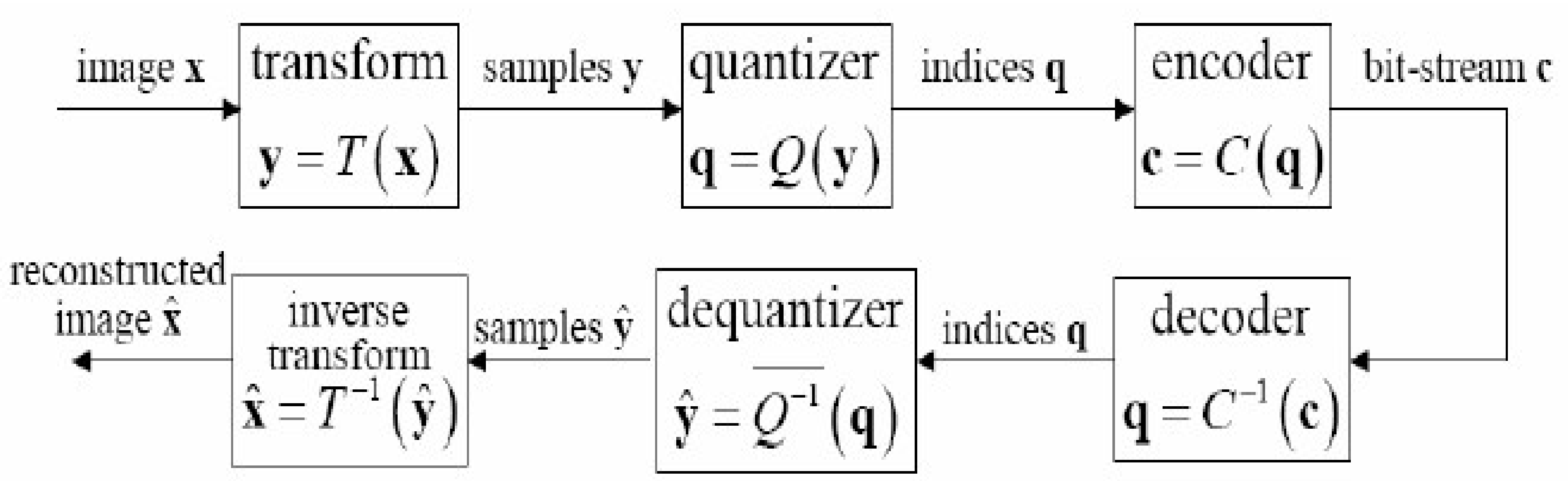
- IGS=1,2,1,2,1,2,7,6
- $e_{rms} = 3.807$
- $SNR_{rms} = 7.801$

Image Compression Model



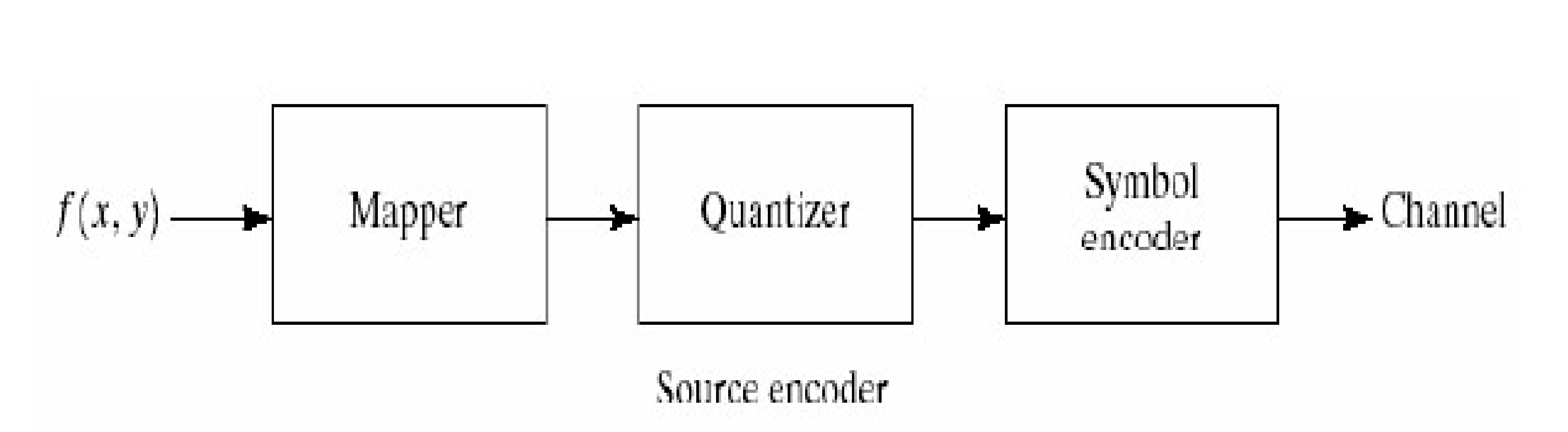
- Source encoder: removes input redundancies
- Channel encoder: increases the noise immunity of the source encoder's output

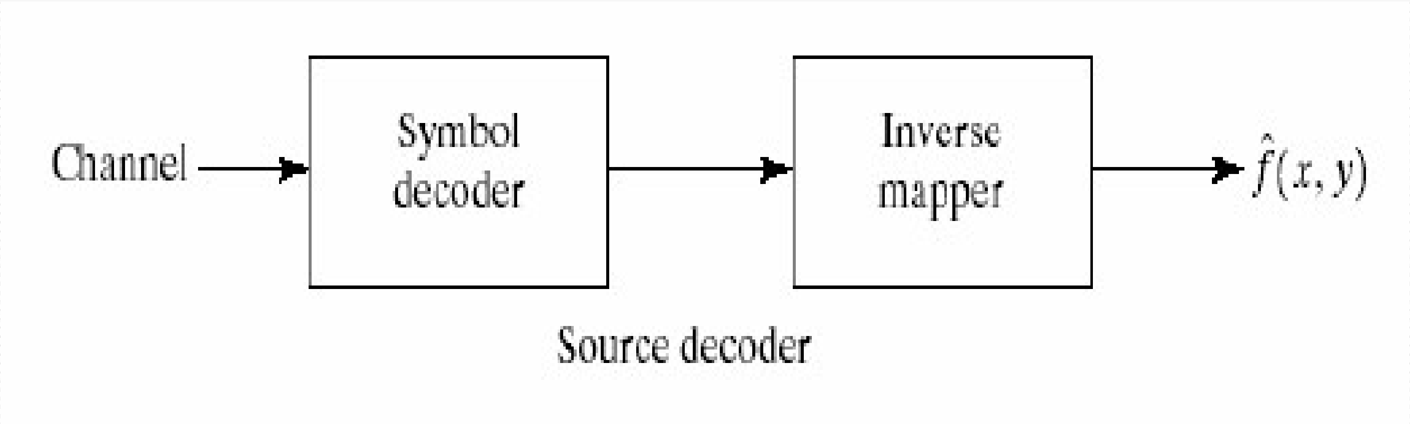
Typical Structured Compression System



- Transform $T(x)$ usually invertible
- Quantization $Q(\mathbf{y})$ not invertible, introduces distortion
- Combination of encoder $C(\mathbf{q})$ and decoder $C^{-1}(\mathbf{c})$ lossless

Source Encoder & Decoder





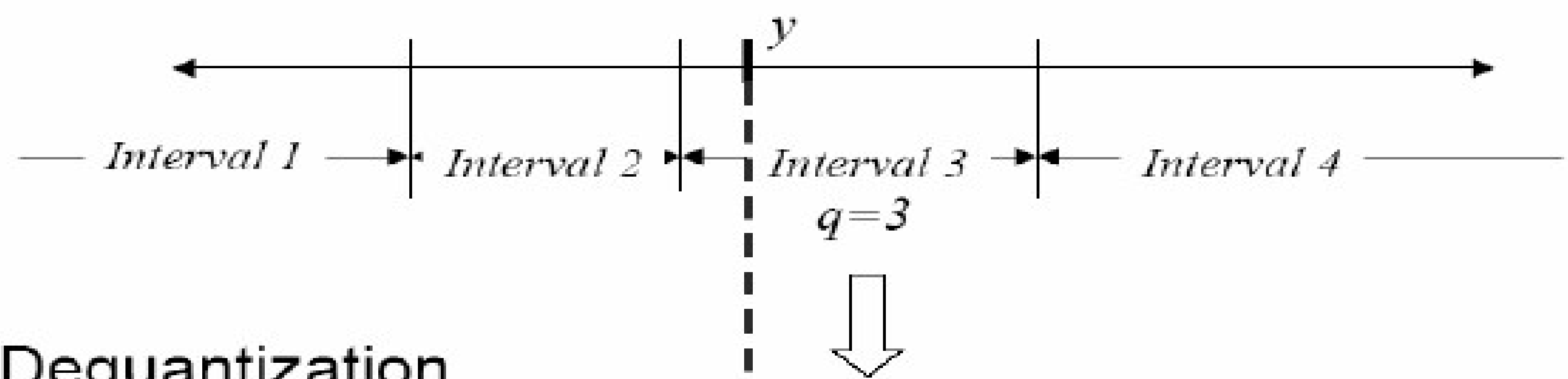
- Mapper: designed to reduce interpixel redundancy; e.g.:
  - Run-length encoding
  - Transform encoding (e.g., DCT in JPEG standard)

- Quantizer: reduces psychovisual redundancies (it cannot be used in lossless compression)

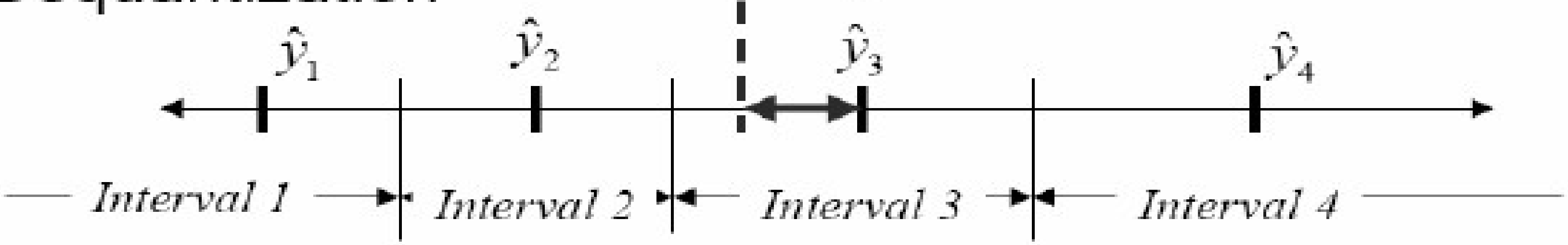
- Symbol Encoder: creates a fixed/variable length code; it reduces coding redundancies

Quantizer

- Goal: reduce the number of possible amplitude values for coding
- Simple scalar quantizerwith four output indices



Dequantization



Channel Encoder and Decoder

- The channel encoder adds "controlled redundancy" to the data to protect it from channel noise

- Hamming encoding: based on appending enough bits to the data to ensure that some minimum number of bits must change between valid code words thereby providing resiliency against transmission errors

Self Information

- If $P(E) = 1$, $I(E) = 0$ (no information)
- If the base of the logarithm is 2, the unit of information is called a "bit"
- If $P(E) = 1/2$ $\quad I(E) = -\log_2(1/2) = 1 \text{ bit}$

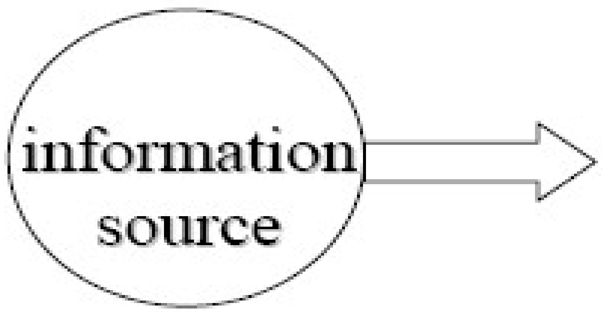
  Example: flipping a coin and communicating the result requires one bit of information
- *1 bit is the amount of information conveyed*

Entropy

- Average information of an image.
- **measure of the degree of randomness in the image**.
- Useful in the context of image coding: it is a lower limit for the average coding length in bits per pixel which can be realized by an optimum coding scheme without any loss of information.

Entropy

Source alphabet



$$\{a_1, a_2, \ldots, a_L\}$$

- Let $p(a_l), l = 1, 2, \ldots, L$ be the probability of each symbol

- Then the entropy (or uncertainty) of the source is given by

$$H = -\sum_{l=1}^{L} p(a_l) \log(p(a_l)) \quad \text{bits/symbol}$$

Computing the Entropy of an Image

8-bit gray level source- statistically independent pixels emission

Consider the 8-bit image:

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray Level	Counts	Probability
21	12	3/8
95	4	1/8
169	4	1/8
243	12	3/8

$$H = 1.81 \text{ bits/pixel}$$

(first-order estimate)

Using Mapping to Reduce Entropy

- Keep first column and replace following with the arithmetic difference between adjacent columns

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

Gray Level or Difference	Counts	Probability
0	~~12~~ 16	1/2
21	4	1/8
74	12	3/8

$$H = 1.41 \text{ bits/pixel}$$

(first-order estimate)

Error-Free/Lossless Compression

- Some applications accept only lossless compression:

  Medical, business documents, satellite data

- Applicable to both binary and gray-scale images.

- Consists of two operations: mapping and coding.

Huffman Coding

- Devised by Huffman in 1952 for removing coding redundancy
- Property: If the symbols of an information source are coded individually, the Huffman coding yields the smallest possible number of code symbols per source symbol

- <u>Method</u>: create a series of source reductions by ordering the probabilities of the symbols under consideration and combining the lowest probability symbols into a single symbol that replaces them in the next source reduction

- <u>Rationale</u>: To assign the shortest possible codewords to the most probable symbols

Example: Huffman Source Reductions

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				

symbols are ordered according to
decreasing probability

Original source			Source reduction							
Sym.	Prob.	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								

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

$$H = -\sum p\log_2(p) = 2.14 \text{ bits/symbol}$$

Huffman Coding

- Coding/decoding is accomplished with a lookup table
- It is a block code: each source symbol is mapped into a fixed sequence of code symbols
- It is instantaneous: each code word in a string of code symbols can be decoded without looking at succeeding symbols
- It is uniquely decodable: any string of code symbols can be decoded only in one way;

example:

$$\underbrace{01010}_{a_3}\ \underbrace{011}_{a_1}\ \underbrace{1}_{a_2}\ \underbrace{1}_{a_2}\ \underbrace{00}_{a_6}$$

a2 - 1
a6 - 00
a1 - 011
a4 - 0100
a3 - 01010
a5 - 01011

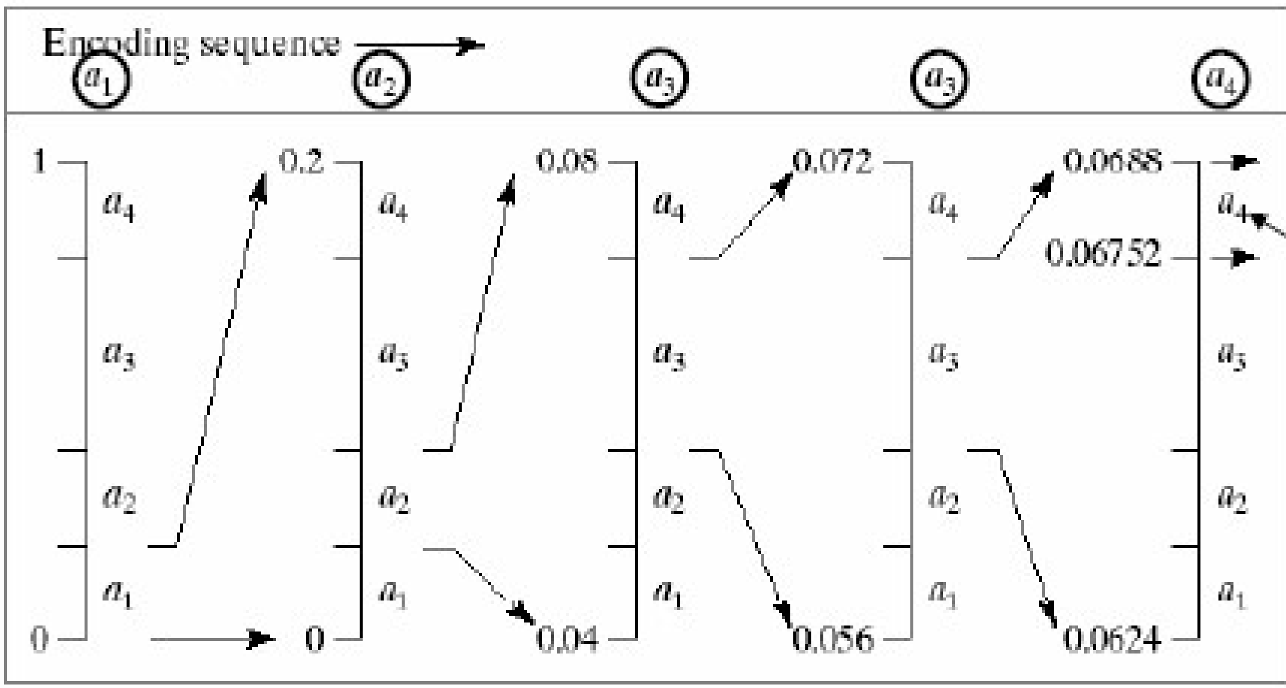
Arithmetic Coding

- Unlike the Huffman coding, arithmetic coding generates nonblock codes
- A one-to-one correspondence between source symbols and code words does not exist
- An entire sequence of source symbols (or message) is assigned a single arithmetic code word

- Property: The code word itself defines an interval of real numbers between 0 and 1.
- As the number of symbols increases, the interval becomes smaller and the number of bits necessary to represent it becomes larger
- Each symbol of the message reduces the size of the interval in accordance with its probability of occurrence

Suppose that a 4-symbol source generates the sequence (or message) $a_1 a_2 a_3 a_3 a_4$

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)



any number in this subinterval, e.g. 0.068, can be used to represent the message

LZW=Lempel-Ziv-Welch Coding

- Uses a dictionary
- Dictionary is adapted to the data
- It assigns fixed-length codewords to variable-length sequences of source symbols
- Decoder builds the matching dictionary based on the codewords received
- Used in GIF, TIFF, and PDF formats

LZW Working

- Reading a sequence of symbols
- Grouping the symbols into strings
- Converting the strings into codes.

- Compression = Codes take up less space than the strings they replace.

Example

Consider the following 4 × 4, 8-bit image of a vertical edge:

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

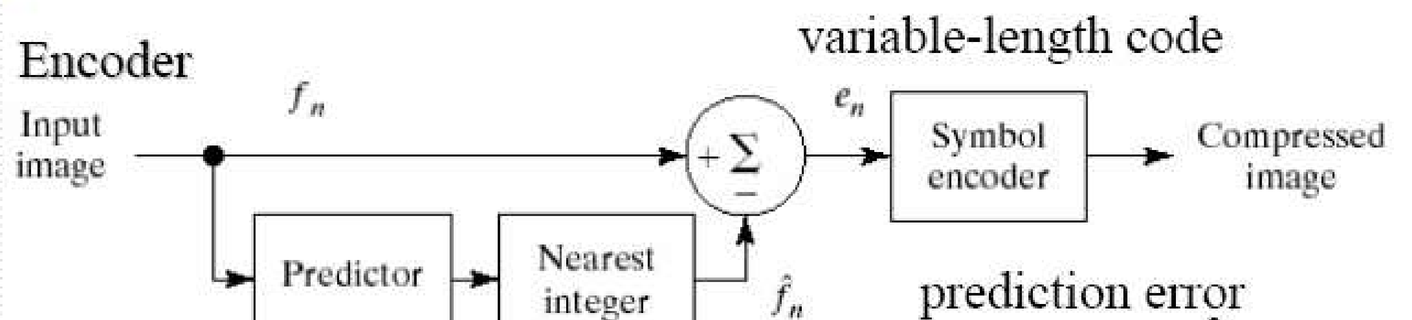
Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

For 8-bit gray-level images, the first 256 words of the dictionary are assigned to the gray values 0 to 255.

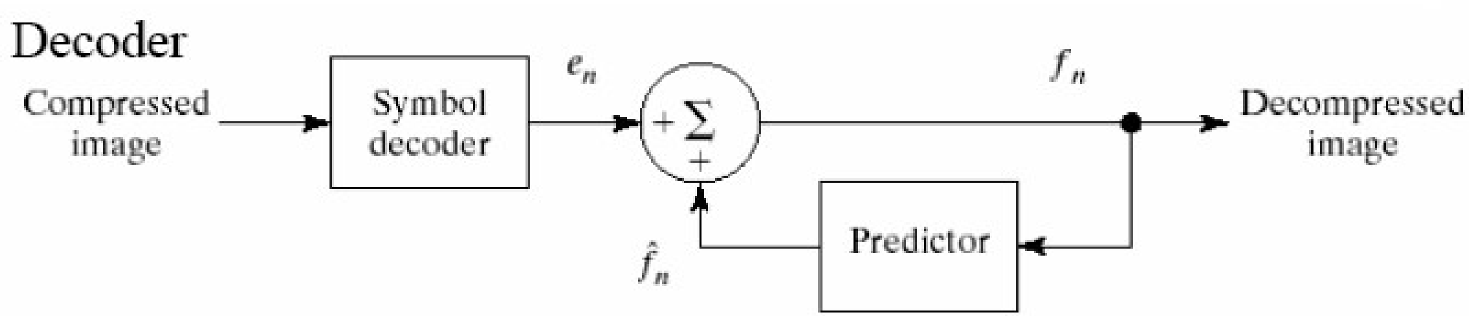
39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Dictionary Location (Code Word)	Dictionary Entry
256	39-39
257	39-126
258	126-126
259	126-39
260	39-39-126
261	126-126-39
262	39-39-126-126
263	126-39-39
264	39-126-126

Lossless Predictive Coding



$$e_n = f_n - \hat{f}_n$$



$$f_n = e_n + \hat{f}_n$$

Predictor

- Generates an estimate of the value of a given pixel based on the values
  - of some past input pixels (temporal prediction) or
  - of some neighboring pixels (spatial prediction)

- Example:

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

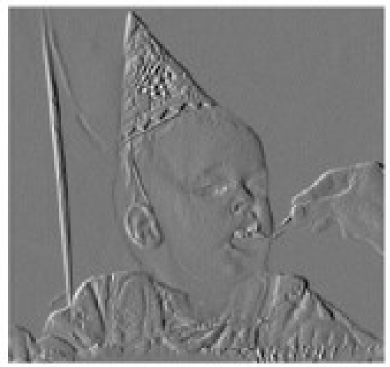
m=order of predictor

Example: Predictive Coding

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

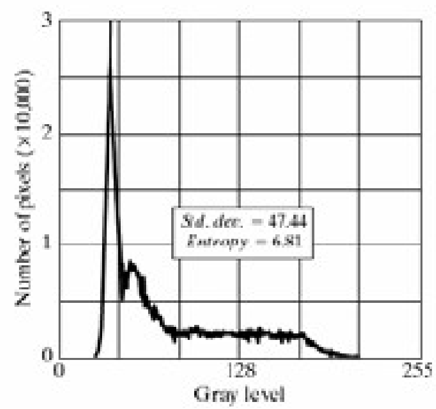
$f(x,y)$

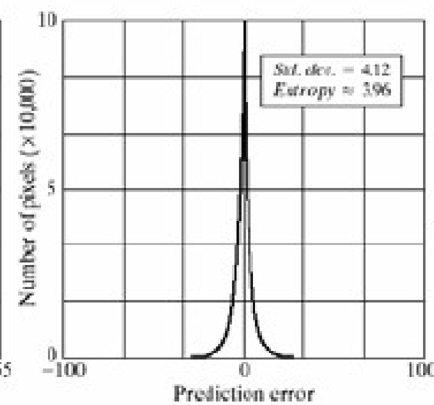
$e(x,y) = f(x,y) - \hat{f}(x,y)$



$\alpha = 1$

histograms of original and error image

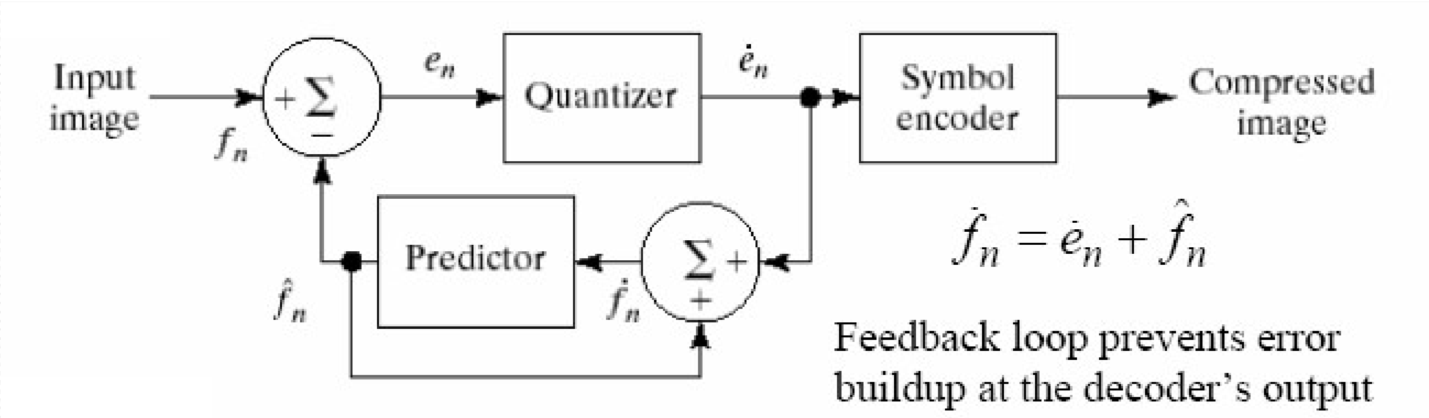




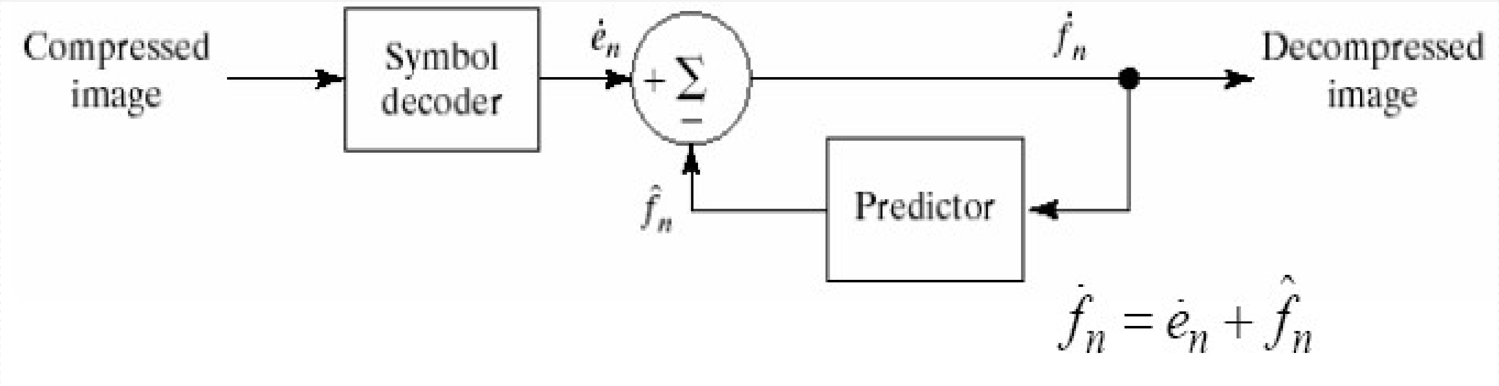
Laplacian PDF

Lossy Predictive Coding

- Encoder:



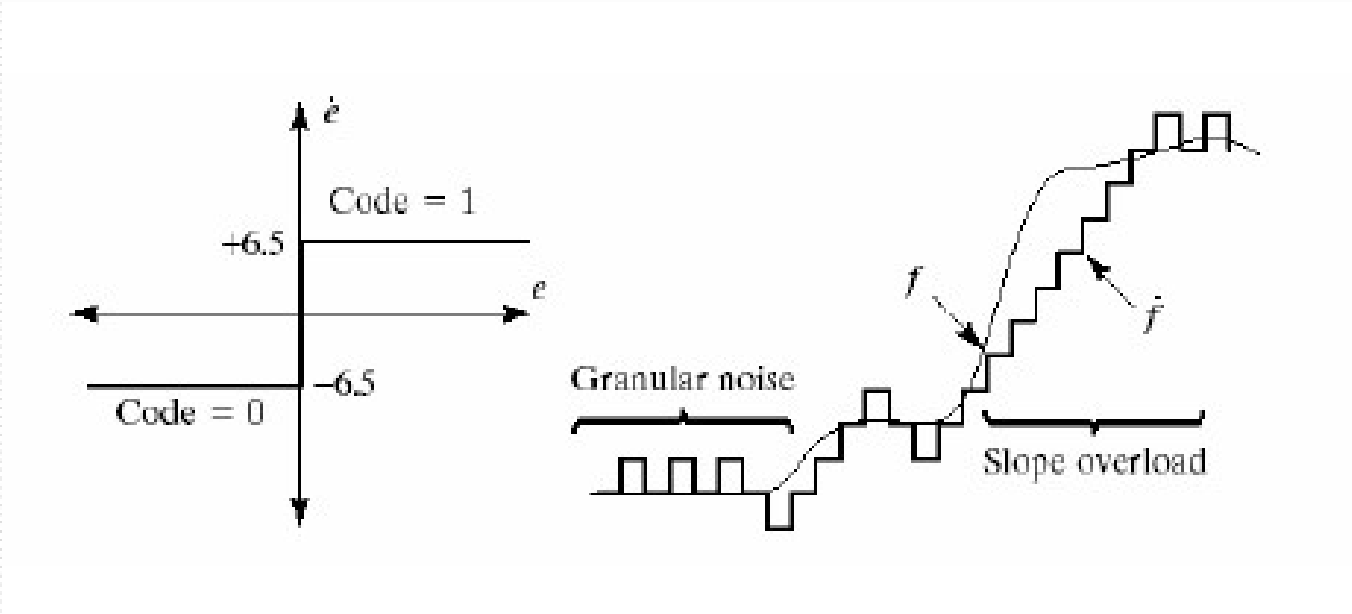
- Decoder:



Input		Encoder				Decoder		Error
n	f	$\hat{f}$	e	$\dot{e}$	$\dot{f}$	$\hat{f}$	$\dot{f}$	$[f - \dot{f}]$
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
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

$$\hat{f}_n = \alpha \dot{f}_{n-1} \quad (0 < \alpha < 1)$$

$$\dot{e}_n = \begin{cases} +\xi & \text{for } e_n > 0 \\ -\xi & \text{otherwise} \end{cases}$$



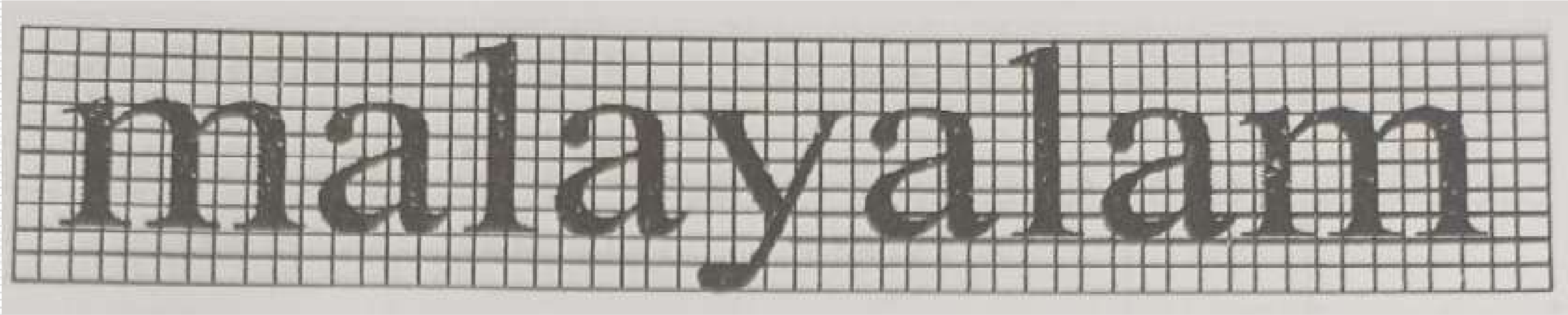
Symbol Based Coding

- Also called Token based
- Image is modelled as a collection of frequently occurring sub-images or tokens.
- Encodes each symbol as a set of triples in a symbol dictionary.
- Triple (x,y,t) :

  (x,y) = position of symbol in the image

  T = position of symbol in the dictionary

Symbol Based Coding



Symbol Based Coding

Token	Symbol
0	m
1	a
2	l
3	y

Triples
(2,1,0)
(2, 10,1)
(0,15,2)
(2,19,1)
(3,23,3)
(2,28,1)
0,33,2)
(2,36,1)
(2,41,0)

Vector Quantization

Image Compression	Entropy, Redundancy and Types, Compression Ratio, Compression Methods. **Lossless Compression:** Run-Length Encoding, Huffman Coding, Arithmetic Coding, LZW Coding, Lossless Predictive coding. **Lossy Compression**: Fidelity Criterion, Improved Gray scale Quantization, Symbol-Based Coding, Bit-Plane Coding, Vector Quantization. **Self-Learning Topics:** DPCM, Block Transform Coding, JPEG compression.