

# Review



Trac D. Tran

ECE Department

The Johns Hopkins University

Baltimore, MD 21218

# Outline

---

- ◆ Review of probability
  - Definition, properties, examples
  - Random variable
  - Random process
  - Signal modeling via random process
- ◆ Multimedia signal properties & formats
  - Digital signals
  - Image/video signals: properties & formats
  - Color space
  - Error & similarity measurements



# Deterministic versus Random

---

## ◆ Deterministic

- Signals whose values can be specified explicitly
- Example: a sinusoid

## ◆ Random

- Digital signals in practice can be treated as a collection of random variables or a random process
- The symbols which occur randomly carry information

## ◆ Probability theory

- The study of random outcomes/events
- Use mathematics to capture behavior of random outcomes and events

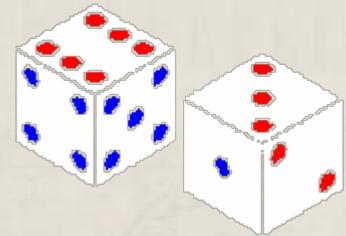
# Probability

## ◆ Events and outcomes

- Let  $X$  be an event with  $N$  possible mutually exclusive outcomes  $\{X_1, X_2, \dots, X_N\}$

### ■ Example

- A coin toss is an event with 2 outcomes: Head ( $H$ ) or Tail ( $T$ )
- A dice toss is an event with 6 outcomes:  $\{1, 2, 3, 4, 5, 6\}$



## ◆ Probability

- The likelihood of observing a particular outcome  $X_i$  above
- Standard notation  $P[X = X_i]$

# Important Properties

- ◆ Probability computation or estimation

$$P[X = X_i] = \frac{N_i}{N_{total}} = \frac{\text{number of possible outcomes } X_i}{\text{total number of outcomes}}$$

- ◆ Basic properties

- Every probability measure lies inclusively between 0 and 1

$$0 \leq P[X = X_i] \leq 1, \quad \forall i$$

- Sum of probabilities of all outcomes is unity:

$$\sum_{i=1}^N P[X = X_i] = 1$$

- For  $N$  equally likely outcomes

$$P[X = X_1] = P[X = X_2] = \dots = P[X = X_N] = \frac{1}{N}$$

- For two statistically independent event  $P[AB] = P[A]P[B]$

# Probability Examples

- ◆ Fair coin flip

$$P[X = 1] = P[X = 0] = \frac{1}{2}$$

- ◆ Tossing two honest coins: what is the probability of observing two heads or two tails?

Four equally likely outcomes

$\{00, 01, 10, 11\}$

$$P[00 \text{ or } 11] = \frac{1}{2}$$

- ◆ Poker game with a standard deck of 52 cards, what is the probability of getting a 5-card heart flush?

Possible flush outcome  $\binom{13}{5} = \frac{13!}{8!5!} = 1287$

$$P[\text{heart flush}] = 0.0495\%$$

Total possible outcome  $\binom{52}{5} = \frac{52!}{47!5!} = 2598960$

# Conditional Probability

---

- ◆ Conditional probability of an event  $A$  assuming that event  $B$  has occurred, denoted  $P[A/B]$ , equals

$$P[A | B] = \frac{P[A \cap B]}{P[B]}$$

- ◆ Bayes' Rule:  $P[A | B] = \frac{P[A]P[B | A]}{P[B]}$

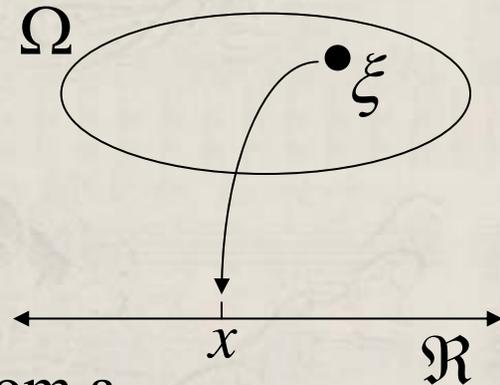
- ◆ Independent events:

$$P[A \cap B] = P[B]P[A | B] = P[B]P[A]$$

# Random Variable

## ◆ Random variable (RV)

- A random variable  $X$  is a mapping which assigns a real number  $x$  to each possible outcome of a random experiment  $\xi$
- A random variable  $X$  takes on a value  $x$  from a given set. Thus it is simply an event whose outcomes have numerical values
- Examples
  - $X$  in coin toss,  $X=1$  for Head,  $X=0$  for Tail
  - The temperature outside Barton Hall at any moment  $t$
  - The pixel value at location  $x, y$  in frame  $n$  of a future Hollywood blockbuster



# Probability Density Function

- ◆ Probability density function (PDF) of a RV  $X$

- Function  $f_X(x)$  defined such that:

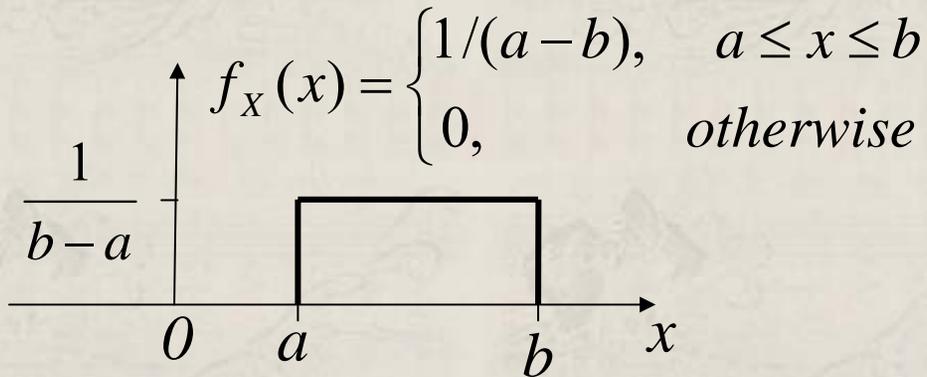
$$P[x_1 \leq X \leq x_2] = \int_{x_1}^{x_2} f_X(x) dx$$

- Histogram of  $X$  !!!
- Main properties:

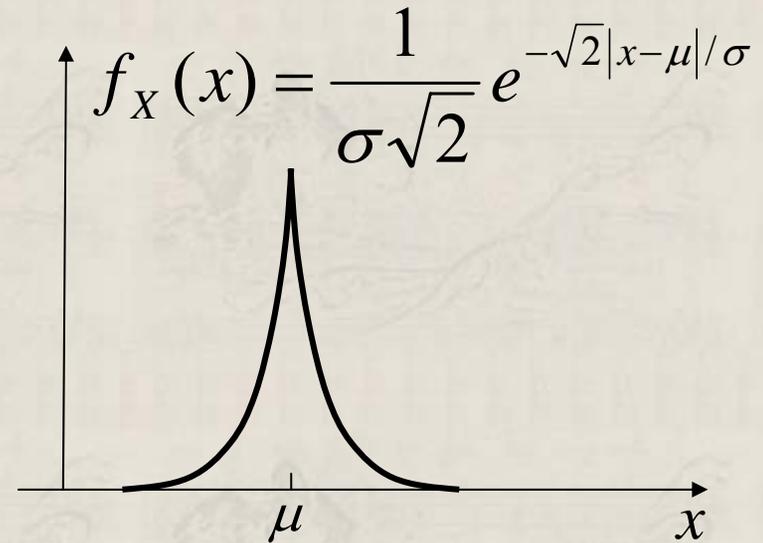
- $\int_{-\infty}^{\infty} f_X(x) dx = 1$

- $f_X(x) \geq 0, \forall x$

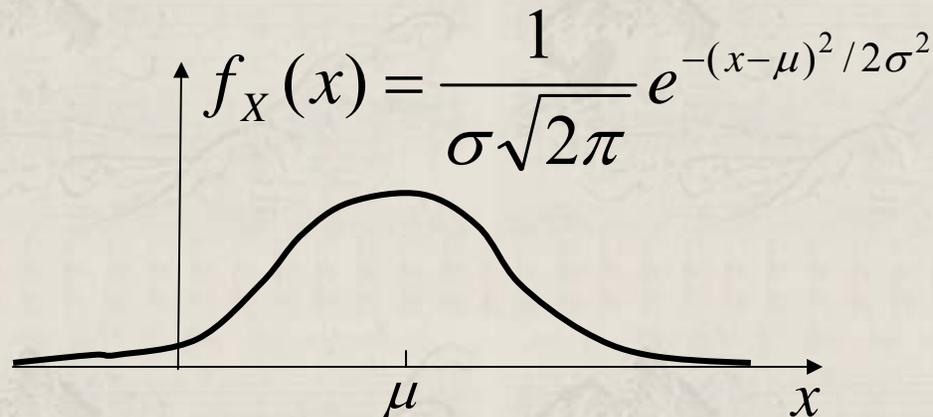
# PDF Examples



**Uniform PDF**



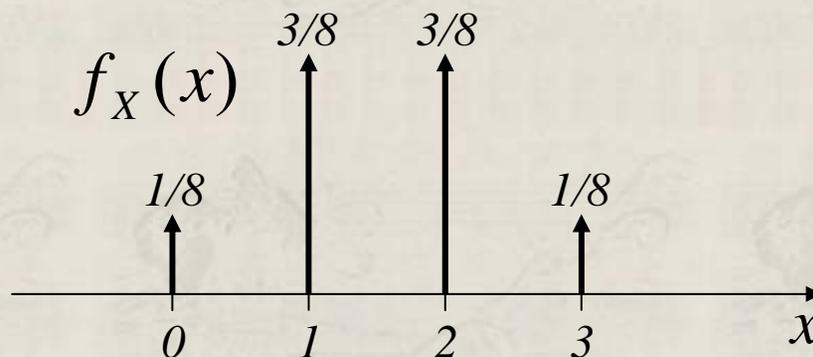
**Laplacian PDF**



**Gaussian PDF**

# Discrete Random Variable

- ◆ RV that takes on discrete values only
- ◆ PDF of discrete RV = discrete histogram
- ◆ Example: how many Heads in 3 independent coin tosses?



$$f_X(x) = \sum_k P_X(x_k) \delta[x - x_k] \quad \text{with} \quad P_X(x_k) = P[X = x_k]$$

# Expectation

## ◆ Expected value

- Let  $g(X)$  be a function of RV  $X$ . The expected value of  $g(X)$  is defined as

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx$$

- Expectation is linear!
- Expectation of a deterministic constant is itself:  $E[C] = C$

## ◆ Mean

$$\mu_X = E[X] = \int_{-\infty}^{\infty} x f_X(x) dx$$

## ◆ Mean-square value

$$E[X^2]$$

## ◆ Variance

$$\sigma_X^2 = E[(X - \mu_X)^2]$$

$$E[X^2] = \sigma_X^2 + \mu_X^2$$

# Cross Correlation & Covariance

## ◆ Cross correlation

- $X, Y$ : 2 jointly distributed RVs

- Joint PDF:

$$P[x_1 \leq X \leq x_2, y_1 \leq Y \leq y_2] = \int_{y_1}^{y_2} \int_{x_1}^{x_2} f_{XY}(x, y) dx dy$$

- Expectation:

$$E[g(X, Y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) f_{XY}(x, y) dx dy$$

- Cross-correlation:

$$R_{XY} \equiv E[XY]$$

## ◆ Cross covariance

$$C_{XY} \equiv E[(X - \mu_X)(Y - \mu_Y)]$$

$$\Rightarrow R_{XY} = C_{XY} + \mu_X \mu_Y$$

# Independence & Correlation

- ◆ Marginal PDF:  $f_X(x) = \int_{-\infty}^{\infty} f_{XY}(x, y) dy$   
 $f_Y(y) = \int_{-\infty}^{\infty} f_{XY}(x, y) dx$

- ◆ Statistically independent:  $f_{XY}(x, y) = f_X(x)f_Y(y)$

- ◆ Uncorrelated:  $E[XY] = E[X]E[Y]$ , *i.e.*  $C_{XY} = 0$

- ◆ Orthogonal:  $E[XY] = 0$

↕ with 0-mean RVs



# Random Process

---

- ◆ Random process (RP)
  - A collection of RVs
  - A time-dependent RV
  - Denoted  $\{X[n]\}$ ,  $\{X(t)\}$  or simply  $X[n]$ ,  $X(t)$
  - We need N-dimensional joint PDF to characterize  $X[n]!$
  - Note: the RVs made up a RP may be dependent or correlated
  - Examples:
    - Temperature  $X(t)$  outside Barton Hall
    - A sequence of binary numbers transmitted over a communication channel
    - Speech, music, image, video signals

# Wide-Sense Stationary

- ◆ Wide-sense stationary (WSS) random process (RP)
  - A WSS RP is one for which  $E[X[n]]$  is independent of  $n$  and  $R(m, n) \equiv E[X[m]X[n]]$  only depends on the difference  $(m - n)$
  - Mean:  $m_X = E[X[n]]$
  - Auto-correlation sequence:  $R_{XX}(k) = E[X[n]X[n-k]]$
  - Energy:  $E[X^2[n]] = R_{XX}(0)$
  - Variance:  $\sigma_X^2 = E[(X[n] - m_X)^2] \Rightarrow \sigma_X^2 = R_{XX}(0) - m_X^2$
  - Co-variance:  $C_{XX}(k) = E[(X[n] - m_X)(X[n-k] - m_X)]$

What happens if the WSS RP has 0-mean?

# White Random Process

## ◆ Power spectral density

- The power spectrum of a WSS RP is defined as the Fourier transform of its auto-correlation sequence

$$S_{XX}(e^{j\omega}) = \sum_k R_{XX}(k) e^{-j\omega k}$$

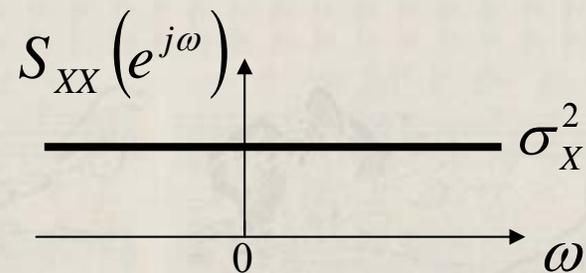
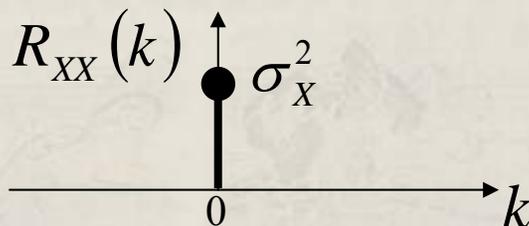
## ◆ White RP

- A RP is said to be white if any pair of samples are uncorrelated, i.e.,  $E[X[n]X[m]] = E[X[n]]E[X[m]]$ ,  $m \neq n$

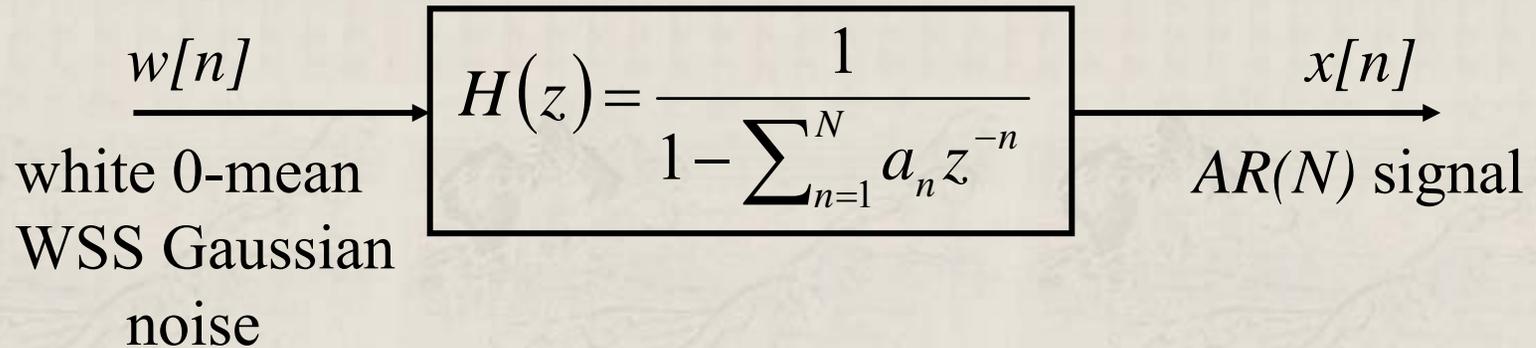
## ◆ White WSS RP

$$R_{XX}(k) = \begin{cases} m_X^2, & k \neq 0 \\ \sigma_X^2 + m_X^2, & k = 0 \end{cases}$$

## ◆ White 0-mean WSS RP

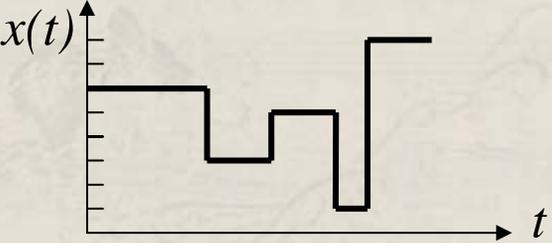
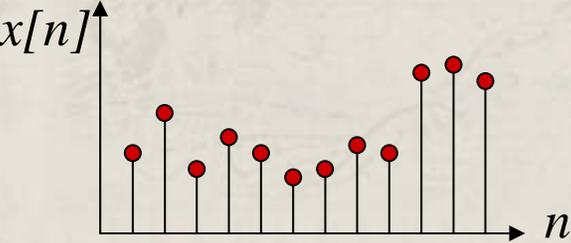
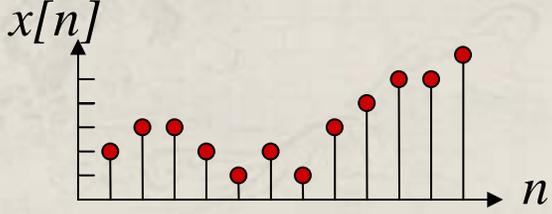


# Stochastic Signal Model



- ◆ For speech:  $N = 10$  to  $20$
- ◆ For images:  $N = 1!$  and  $a_1 = 0.95$

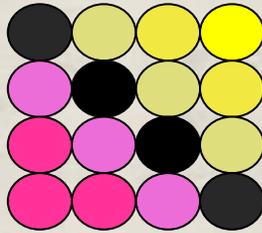
# Continuous & Discrete Representations

	Continuous-Amplitude	Discrete-Amplitude
Continuous -Time (Space)	 <p>Local telephone, cassette-tape recording &amp; playback, phonograph, photograph</p>	 <p>telegraph</p>
Discrete -Time (Space)	 <p>Switched capacitor filter, speech storage chip, half-tone photography</p>	 <p>CD, DVD, cellular phones, digital camera &amp; camcorder, digital television, inkjet printer</p>

# Multi-Dimensional Digital Signals

- ◆ **Images:** 2-D digital signals

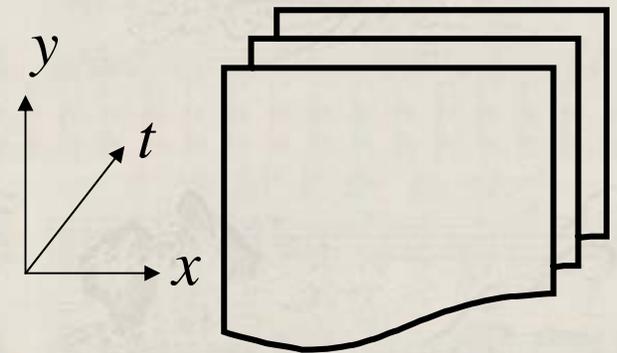
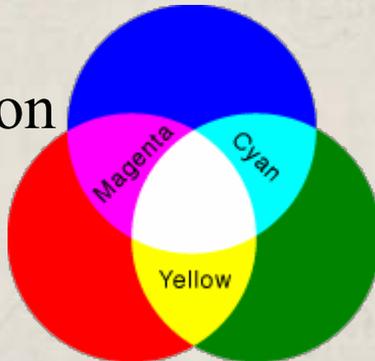
pixel  
or  
pel



● black p=0    ● gray p=128    ● white p=255

- ◆ **Video Sequences:** 3-D digital signals, a collection of 2-D images called frames

colors:  
combination  
of RGB



# Popular Signal Formats

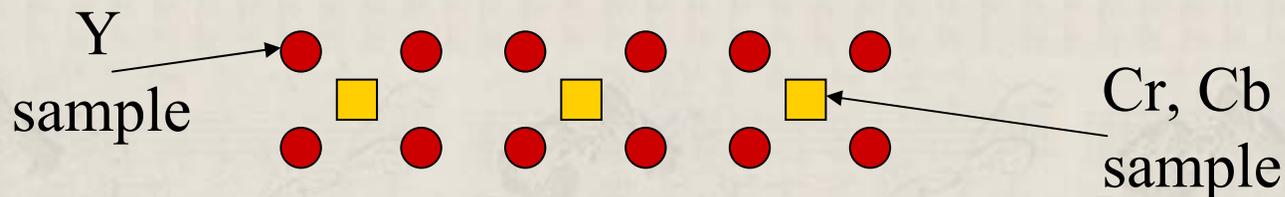
## ◆ RGB

- Red Green Blue, typically 8-bit per sample for each color plane

## ◆ YCrCb

- Y: luminance, gray-scale component
- Cr & Cb: chrominance, color components, less energy than Y
- Chrominance components can be down-sampled without much aliasing

$$\begin{bmatrix} Y \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ 0.439 & -0.368 & -0.071 \\ -0.148 & -0.291 & 0.439 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$



# Popular Signal Formats

## ◆ CIF: Common Intermediate Format

- Y resolution: 352 x 288
- Cr, Cb resolution: 176 x 144
- Frame rate: 30 frames/second progressive
- 8 bits/pixel(sample)

## ◆ QCIF: Quarter Common Intermediate Format

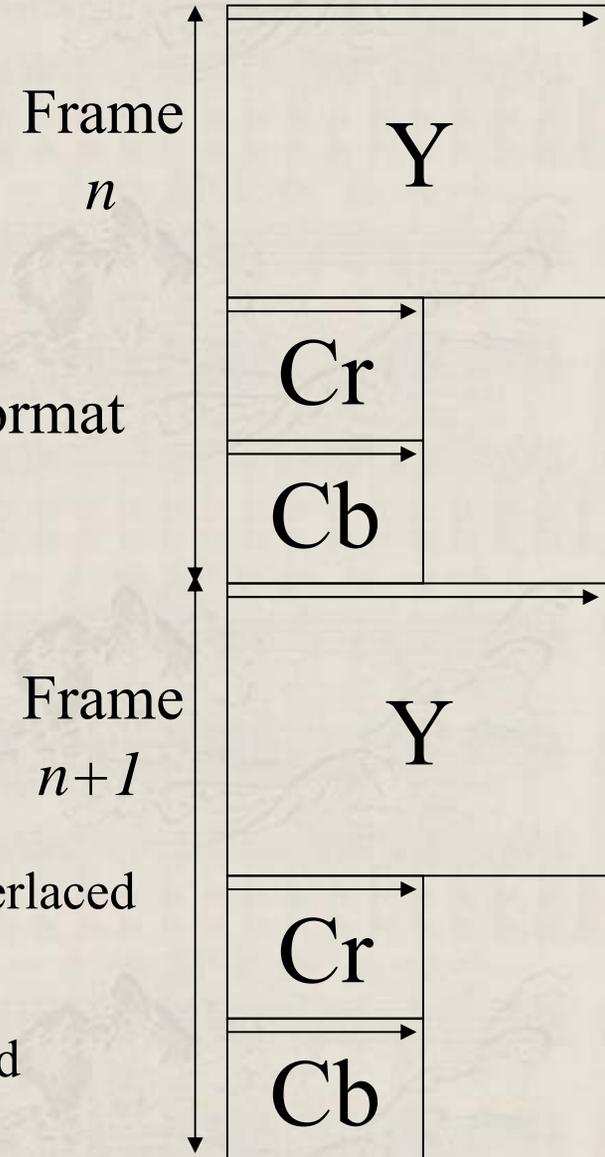
- Y resolution: 176 x 144
- Cr, Cb resolution: 88 x 72
- Frame rate: 30 frames/second progressive
- 8 bits/pixel (sample)

## ◆ TV – NTSC

- Resolution: 704 x 480, 30 frames/second interlaced

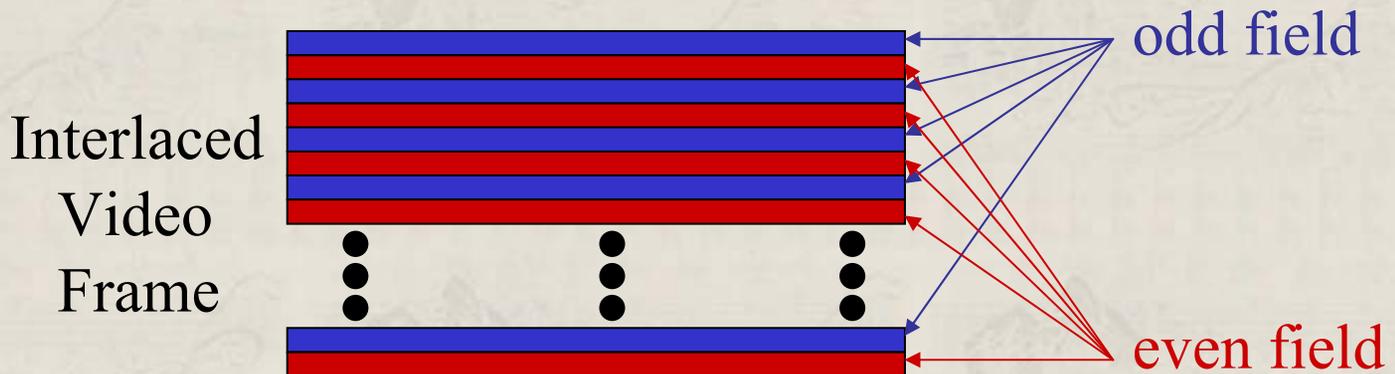
## ◆ DVD – NTSC

- Resolution: 720 x 480, 24 – 30 frames/second progressive



# High-Definition Television (HDTV)

- ◆ 720i
  - Resolution: 1280 x 720, interlaced
- ◆ 720p
  - Resolution: 1280 x 720, progressive
- ◆ 1080i
  - Resolution: 1920 x 1080, interlaced
- ◆ 1080p
  - Resolution: 1920 x 1080, progressive



# Digital Bit Rates

---

- ◆ *A picture is worth a thousand words?*
- ◆ Size of a typical color image
  - For display
    - $640 \times 480 \times 24 \text{ bits} = 7372800 \text{ bits} = 92160 \text{ bytes}$
  - For current mainstream digital camera
    - $2560 \times 1920 \times 24 \text{ bits} = 117964800 \text{ bits} = 14745600 \text{ bytes}$
  - For an average word
    - 6 characters/word, 7 bits/character: 42 bits  $\approx$  5 bytes
- ◆ Bit rate: bits per second for transmission
  - Raw digital video (DVD format)
    - $720 \times 480 \times 24 \times 24 \text{ frames: } \sim 200 \text{ Mbps}$
  - CD Music
    - $44100 \text{ samples/second} \times 16 \text{ bits/sample} = 689 \text{ kbps}$

# Error or Similarity Measures

- ◆ Mean Square Error (MSE)

$$L_2 \text{ - norm error : } MSE = \frac{1}{N} \sum_{i=0}^{N-1} E\left(\left|X_i - \hat{X}_i\right|^2\right)$$

- ◆ Mean Absolute Difference (MAD)

$$L_1 \text{ - norm error : } MAD = \frac{1}{N} \sum_{i=0}^{N-1} E\left(\left|X_i - \hat{X}_i\right|\right)$$

- ◆ Max Error

$$L_\infty \text{ - norm error : } MaxError = \max_i \left\{ E\left(\left|X_i - \hat{X}_i\right|\right) \right\}$$

- ◆ Peak Signal-to-Noise Ratio (PSNR)

$$PSNR = 10 \log_{10} \frac{M^2}{MSE};$$

$M$  = maximum peak - to - peak value

# Variable Length Coding: Introduction to Lossless Compression



Trac D. Tran

ECE Department

The Johns Hopkins University

Baltimore, MD 21218

# Outline

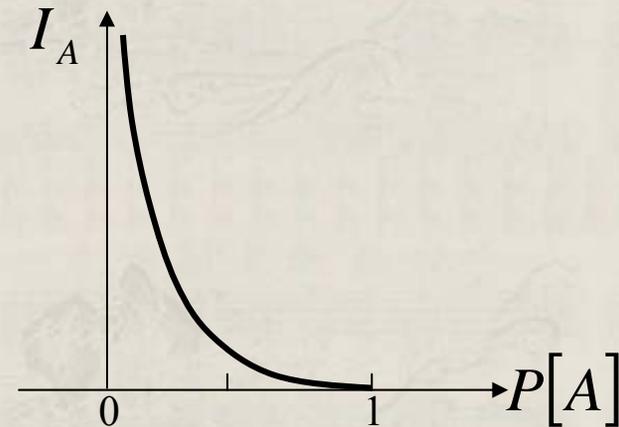
---

- ◆ Review of information theory
- ◆ Fixed-length codes
  - ASCII
- ◆ Variable-length codes
  - Morse code
  - Shannon-Fano code
  - Huffman code
  - Arithmetic code
- ◆ Run-length coding

# Information Theory

- ◆ A measure of information
  - The amount of information in a signal might not equal to the amount of data it produces
  - The amount of information about an event is closely related to its probability of occurrence
- ◆ Self-information
  - The information conveyed by an event  $A$  with probability of occurrence  $P[A]$  is

$$I_A = \log_2 \frac{1}{P[A]} = -\log_2 P[A]$$



# Entropy

## ◆ Entropy

- Average amount of information of a source, more precisely, the average number of bits of information required to represent the symbols the source produces
- For a source containing  $N$  independent symbols, its entropy is defined as

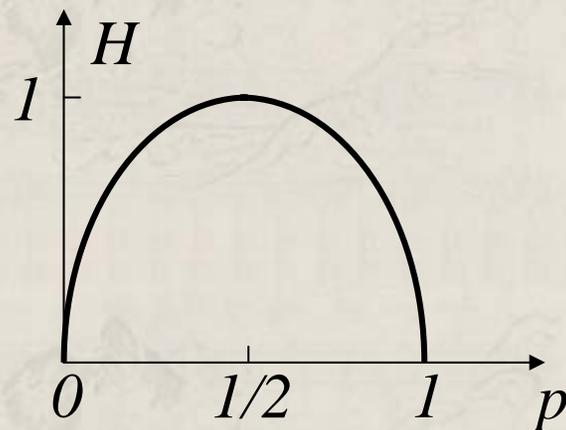
$$H = -\sum_{i=1}^N P[X_i] \log_2 P[X_i]$$

- Unit of entropy: bits/symbol
- C. E. Shannon, “A mathematical theory of communication,” Bell Systems Technical Journal, 1948

# Entropy Example

- ◆ Find and plot the entropy of the binary code in which the probability of occurrence for the symbol  $1$  is  $p$  and for the symbol  $0$  is  $1-p$

$$\begin{aligned} H &= -\sum_{i=1}^2 P[X_i] \log_2 P[X_i] \\ &= -p \log_2 p - (1-p) \log_2 (1-p) \end{aligned}$$



$$p = \frac{1}{2} \Rightarrow H = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = \frac{1}{2} + \frac{1}{2} = 1 \text{ bit/symbol}$$

$$p = \frac{1}{4} \Rightarrow H = -\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} = 0.8113 \text{ bits/symbol}$$

$$p = 0 \Rightarrow H = -0 \log_2 0 - 1 \log_2 1 = 0 \text{ bit/symbol}$$

# Entropy Example

- ◆ Find the entropy of a DNA sequence containing four equally-likely symbols  $\{A, C, T, G\}$

$$H = \left( -\frac{1}{4} \log_2 \frac{1}{4} \right) \times 4 = \log_2 4 = 2 \text{ bits/symbol}$$

- ◆  $P[A]=1/2; P[C]=1/4; P[T]=P[G]=1/8; H=?$

$$\begin{aligned} H &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{8} \log_2 \frac{1}{8} \\ &= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4} \text{ bits/symbol} < 2 \text{ bits/symbol} \end{aligned}$$

- ◆ So, how do we design codes to represent DNA sequences?

# Fixed-Length Codes

---

## ◆ Properties

- Use the same number of bits to represent all possible symbols produced by the source
- Simplify the decoding process

## ◆ Examples

- American Standard Code for Information Interchange (ASCII) code
- Bar codes
  - One used by the US Postal Service
  - Universal Product Code (UPC) on products in stores
  - Credit card codes

# ASCII Code

- ◆ ASCII is used to encode and communicate alphanumeric characters for plain text
- ◆ 128 common characters: lower-case and upper-case letters, numbers, punctuation marks... 7 bits per character
- ◆ First 32 are control characters (for example, for printer control)
- ◆ Since a byte is a common structured unit of computers, it is common to use 8 bits per character – there are an additional 128 special symbols
- ◆ Example

<i>Character</i>	5	2	0	.	4	4	3
<i>Dec. index</i>	53	50	48	46	52	52	51
<i>Bin. code</i>	00110101	00110010	00110000	00101110	00110100	00110100	00110011

# ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

# Variable-Length Codes

---

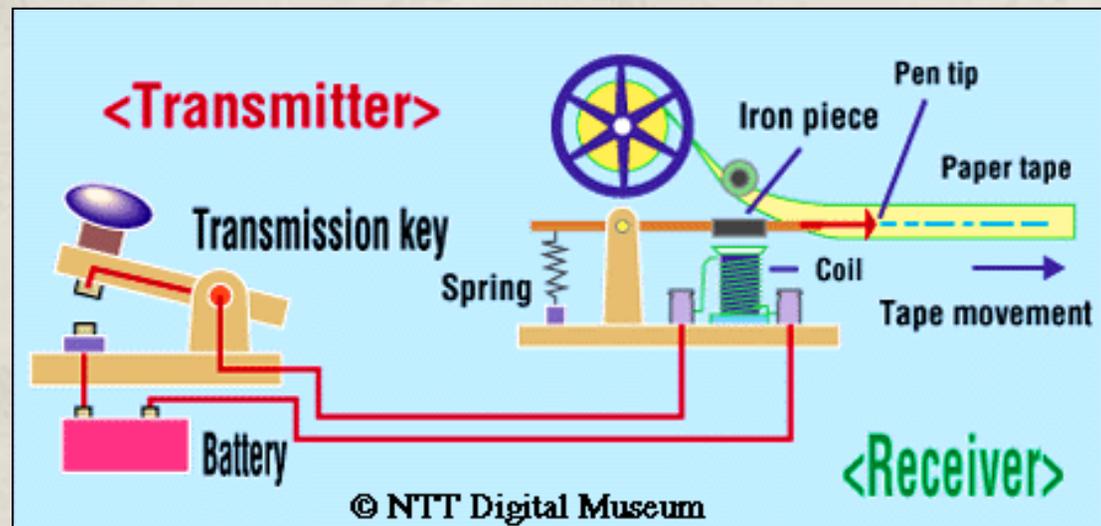
- ◆ Main problem with fixed-length codes: inefficiency
- ◆ Main properties of variable-length codes (VLC)
  - Use a different number of bits to represent each symbol
  - Allocate shorter-length code-words to symbols that occur more frequently
  - Allocate longer-length code-words to rarely-occurred symbols
  - More efficient representation; good for compression
- ◆ Examples of VLC
  - Morse code
  - Shannon-Fano code
  - Huffman code
  - Arithmetic code

# Morse Codes & Telegraphy

## ◆ Morse codes

E	•	T	-
I	••	M	--
S	•••	O	---
H	••••	CH	----
A	•-	N	-•
U	••-	D	-••
V	•••-	B	-•••
R	•-•	K	-•-
W	•--	G	--•
L	•-••	Y	-••-
P	•--•	X	-•••
F	••••	Q	--••
J	•---	C	-•••

- ◆ “What hath God wrought?”, DC – Baltimore, 1844
- ◆ Allocate shorter codes for more frequently-occurring letters & numbers
- ◆ Telegraph is a binary communication system – dash: **1**; dot: **0**





# Issues in VLC Design

---

- ◆ Optimal efficiency
  - How to perform optimal code-word allocation (in an efficiency standpoint) given a particular signal?
- ◆ Uniquely decodable
  - No confusion allowed in the decoding process
  - Example: Morse code has a major problem!
    - Message: *SOS*. Morse code: **000111000**
    - Many possible decoded messages: *SOS* or *VMS*?
- ◆ Instantaneously decipherable
  - Able to decipher as we go along without waiting for the entire message to arrive
- ◆ Algorithmic issues
  - Systematic design?
  - Simple fast encoding and decoding algorithms?

# VLC Example

Symbol	Prob.	FLC	Code 1	Code 2	Code 3	Code 4
A	$P[A]=1/2$	000	1	1	0	00
B	$P[B]=1/4$	001	01	10	10	01
C	$P[C]=1/8$	010	001	100	110	10
D	$P[D]=1/16$	011	0001	1000	1110	11
E	$P[E]=1/16$	100	00001	10000	1111	110
Average Length	$H=30/16$	3	$31/16$	$31/16$	$30/16$	$33/16$

$$\begin{aligned}
 H &= -\sum_{i=1}^N P[X_i] \log_2 P[X_i] \\
 &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{2}{16} \log_2 \frac{1}{16} \\
 &= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{8}{16} = \frac{30}{16} = \frac{15}{8} \text{ bits / symbol}
 \end{aligned}$$

Code 1:

$$\begin{aligned}
 E[L] &= 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 4 \times \frac{1}{16} \\
 &\quad + 5 \times \frac{1}{16} = \frac{31}{16} \text{ bits / symbol}
 \end{aligned}$$

# VLC Example

Symbol	Prob.	FLC	Code 1	Code 2	Code 3	Code 4
A	$P[A]=1/2$	000	1	1	0	00
B	$P[B]=1/4$	001	01	10	10	01
C	$P[C]=1/8$	010	001	100	110	10
D	$P[D]=1/16$	011	0001	1000	1110	11
E	$P[E]=1/16$	100	00001	10000	1111	110
Average Length	$H=30/16$	3	$31/16$	$31/16$	$30/16$	$33/16$

- ◆ Uniquely decodable – Self-synchronizing: Code 1, 2, 3.  
No confusion in decoding
- ◆ Instantaneous: Code 1, 3. No need to look ahead.
- ◆ Prefix condition = uniquely decodable & instantaneous:  
no codeword is a prefix of another

# Shannon-Fano Code

## ◆ Algorithm

- Line up symbols by decreasing probability of occurrence
- Divide symbols into 2 groups so that both have similar combined probability
- Assign **0** to 1<sup>st</sup> group and **1** to the 2<sup>nd</sup>
- Repeat step 2

## ◆ Example

$$H=2.2328 \text{ bits/symbol}$$

Symbols	Prob.	Code-word
A	0.35	00
B	0.17	01
C	0.17	10
D	0.16	110
E	0.15	111

$$\begin{aligned} \text{Average code-word length} = & 0.35 \times 2 + 0.17 \times 2 + 0.17 \times 2 \\ & + 0.16 \times 3 + 0.15 \times 3 \\ = & 2.31 \text{ bits/symbol} \end{aligned}$$

# Huffman Code

---

- ◆ Shannon-Fano code [1949]
  - Top-down algorithm: assigning code from most frequent to least frequent
  - VLC, uniquely & instantaneously decodable (no code-word is a prefix of another)
  - Unfortunately not optimal in term of minimum redundancy
- ◆ Huffman code [1952]
  - Quite similar to Shannon-Fano in VLC concept
  - Bottom-up algorithm: assigning code from least frequent to most frequent
  - Minimum redundancy when probabilities of occurrence are powers-of-two
  - In JPEG images, DVD movies, MP3 music

# Huffman Coding Algorithm

## ◆ Encoding algorithm

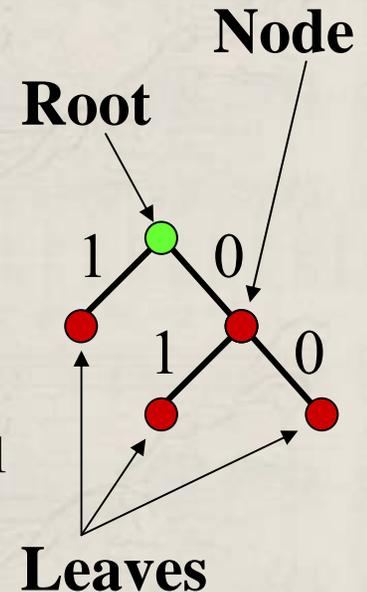
- Order the symbols by decreasing probabilities
- Starting from the bottom, assign **0** to the least probable symbol and **1** to the next least probable
- Combine the two least probable symbols into one composite symbol
- Reorder the list with the composite symbol
- Repeat Step 2 until only two symbols remain in the list

## ◆ Huffman tree

- Nodes: symbols or composite symbols
- Branches: from each node, 0 defines one branch while 1 defines the other

## ◆ Decoding algorithm

- Start at the root, follow the branches based on the bits received
- When a leaf is reached, a symbol has just been decoded



# Huffman Coding Example

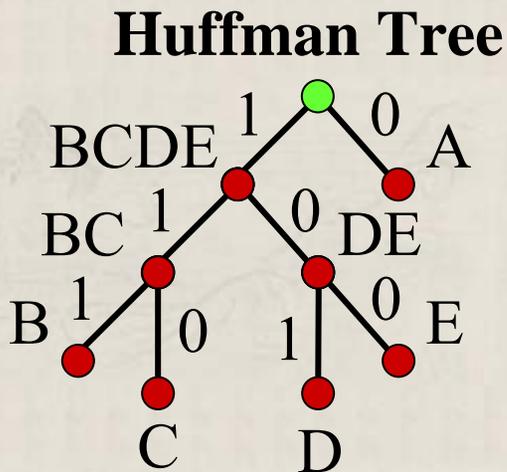
Symbols	Prob.	
A	0.35	
B	0.17	
C	0.17	
D	0.16	1
E	0.15	0

Symbols	Prob.	
A	0.35	
DE	0.31	
B	0.17	1
C	0.17	0

Symbols	Prob.	
A	0.35	
BC	0.34	1
DE	0.31	0

## Huffman Codes

A	0
B	111
C	110
D	101
E	100



Symbols	Prob.	
BCDE	0.65	1
A	0.35	0

Average code-word length =  $E[L] = 0.35 \times 1 + 0.65 \times 3 = 2.30$  bits/symbol

# Huffman Coding Example

Symbols	Prob.	
A	1/2	
B	1/4	
C	1/8	
D	1/16	0
E	1/16	1



Symbols	Prob.	
A	1/2	
B	1/4	
C	1/8	0
DE	1/8	1



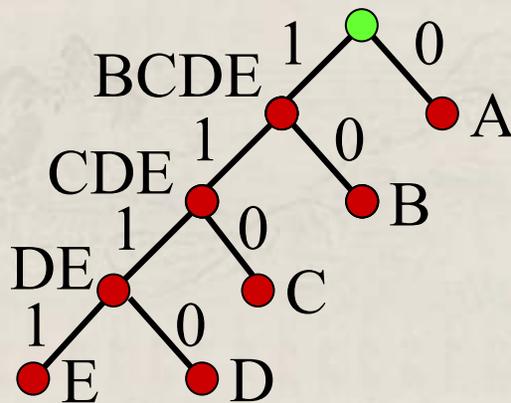
Symbols	Prob.	
A	1/2	
B	1/4	0
CDE	1/4	1



## Huffman Codes

A	0
B	10
C	110
D	1110
E	1111

## Huffman Tree



Symbols	Prob.	
A	1/2	0
BCDE	1/2	1

Average code-word length =  $E[L] =$

$$0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 4 = 1.875 \text{ bits/symbol} = H$$

# Huffman Shortcomings

---

- ◆ Difficult to make adaptive to data changes
- ◆ Only optimal when  $P[X_i] = \frac{1}{2^{k_i}}$
- ◆ Best achievable bit-rate = 1 bit/symbol
  
- ◆ Question: What happens if we only have 2 symbols to deal with? A binary source with skewed statistics?
  - Example:  $P[0]=0.9375$ ;  $P[1]=0.0625$ .  
 $H = 0.3373$  bits/symbol. Huffman:  $E[L] = 1$ .
  - One solution: combining symbols!

# Extended Huffman Code

**Symbols Prob.**

A=0 15/16

B=1 1/16

H=0.3373 bits/symbol



**Symbols**

AA 225/256

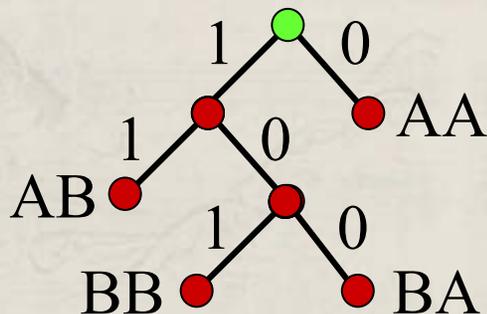
AB 15/256

BA 15/256

BB 1/256

H=0.6746 bits/symbol

## Huffman Tree



- ◆ Larger grouping yield better performance
- ◆ Problems
  - Storage for codes
  - Inefficient & time-consuming
  - Still not well-adaptive

Average code-word length =  $E[L] = 1 \times 225/256 + 2 \times 15/256 + 3 \times 15/256 + 3 \times 1/256 = 1.1836 \text{ bits/symbol} \gg 2$

# Arithmetic Coding: Main Idea

- ◆ Peter Elias in Robert Fano's class!
- ◆ Large grouping improves coding performance; however, we do not want to generate codes for all possible sequences
- ◆ Wish list
  - a tag (unique identifier) is generated for the sequence to be encoded
  - easy to adapt to statistic collected so far
  - more efficient than Huffman
- ◆ Main Idea: tag the sequence to be encoded with a number in the unit interval  $[0, 1)$  and send that number to the decoder
- ◆ Review: binary representation of fractions
  - $0.75_{10} = 0.5 + 0.25 = 2^{-1} + 2^{-2} = 0.11_2$
  - $0.384765625_{10} = 2^{-2} + 2^{-3} + 2^{-7} + 2^{-9} = 0.011000101_2$

# Coding Example

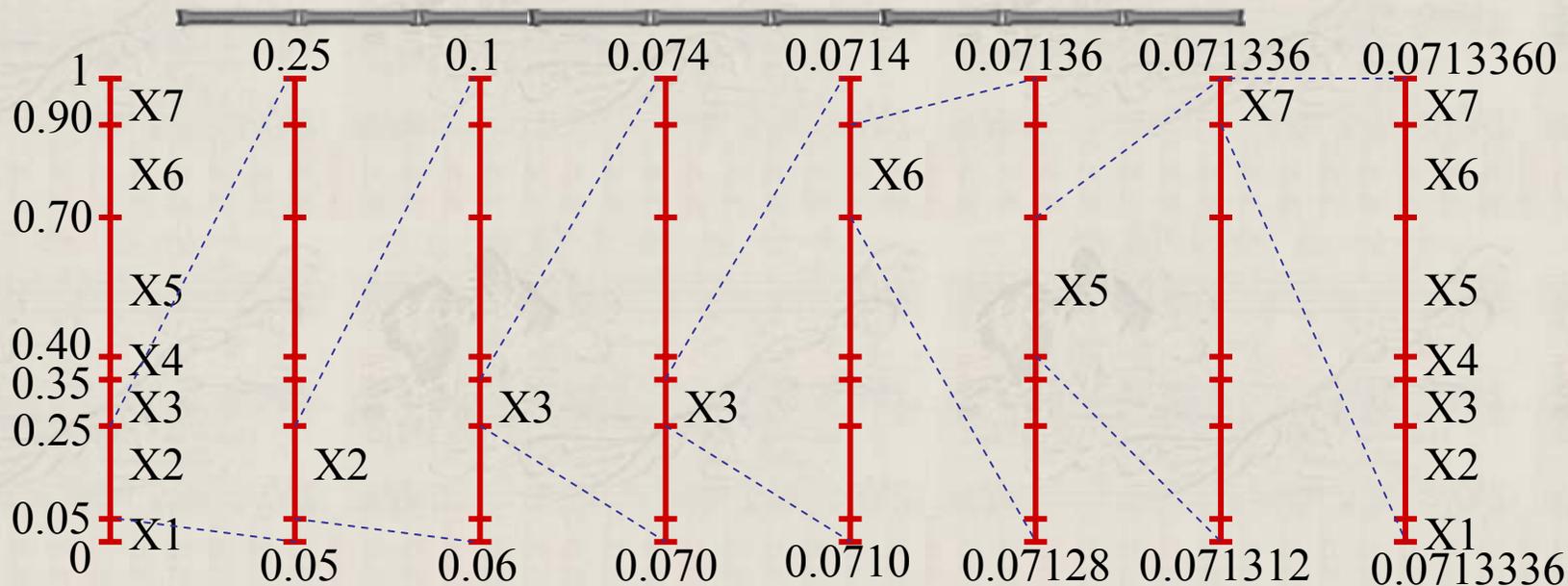
Symbol	Probability	Huffman Code
X1	0.05	10101
X2	0.2	01
X3	0.1	100
X4	0.05	10100
X5	0.3	11
X6	0.2	00
X7	0.1	1011

String to encode: X2 X2 X3 X3 X6 X5 X7

Huffman: 01 01 100 100 00 11 1011

18 bits

# Arithmetic Encoding Process



String to encode: X2 X2 X3 X3 X6 X5 X7

range = high – low

new\_high = low + range x subinterval\_high

new\_low = low + range x subinterval\_low

Final interval = [0.0713336, 0.0713360)

16 bits

Send to decoder: 0.07133483886719

$2^{-4} + 2^{-7} + 2^{-10} + 2^{-15} + 2^{-16} = 0.0001001001000011_2$

Sym	Prob	Huffman
X1	0.05	10101
X2	0.2	01
X3	0.1	100
X4	0.05	10100
X5	0.3	11
X6	0.2	00
X7	0.1	1011

# Arithmetic Decoding Process

◆ low=0; high=1; range=high – low

◆ REPEAT

▪ Find index  $i$  such that

$$\text{subinterval\_low} \leq \frac{\text{value} - \text{low}}{\text{range}} \leq \text{subinterval\_high}$$

▪ OUTPUT SYMBOL

▪  $\text{high} = \text{low} + \text{range} \times \text{subinterval\_high}$

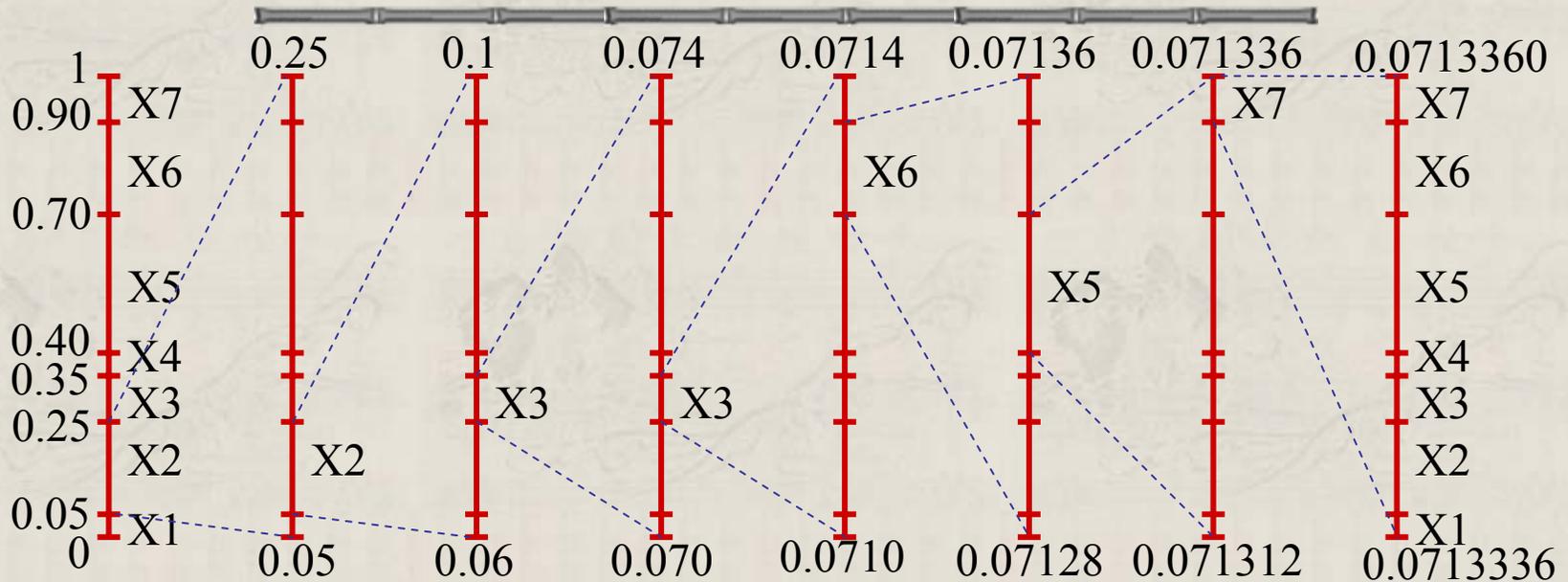
▪  $\text{low} = \text{low} + \text{range} \times \text{subinterval\_low}$

▪  $\text{range} = \text{high} - \text{low}$

} UPDATE

◆ UNTIL END

# Arithmetic Decoding Example



$low = 0; high = 1; range = 0$

$$0.05 \leq \frac{0.071334... - 0}{1} \leq 0.25 \Rightarrow X_2$$

$$high = 0 + 1 \times 0.25 = 0.25$$

$$low = 0 + 1 \times 0.05 = 0.05$$

$$0.05 \leq \frac{0.071334... - 0.05}{0.20} = 0.1067 \leq 0.25 \Rightarrow X_2$$

$$high = 0.05 + 0.2 \times 0.25 = 0.1$$

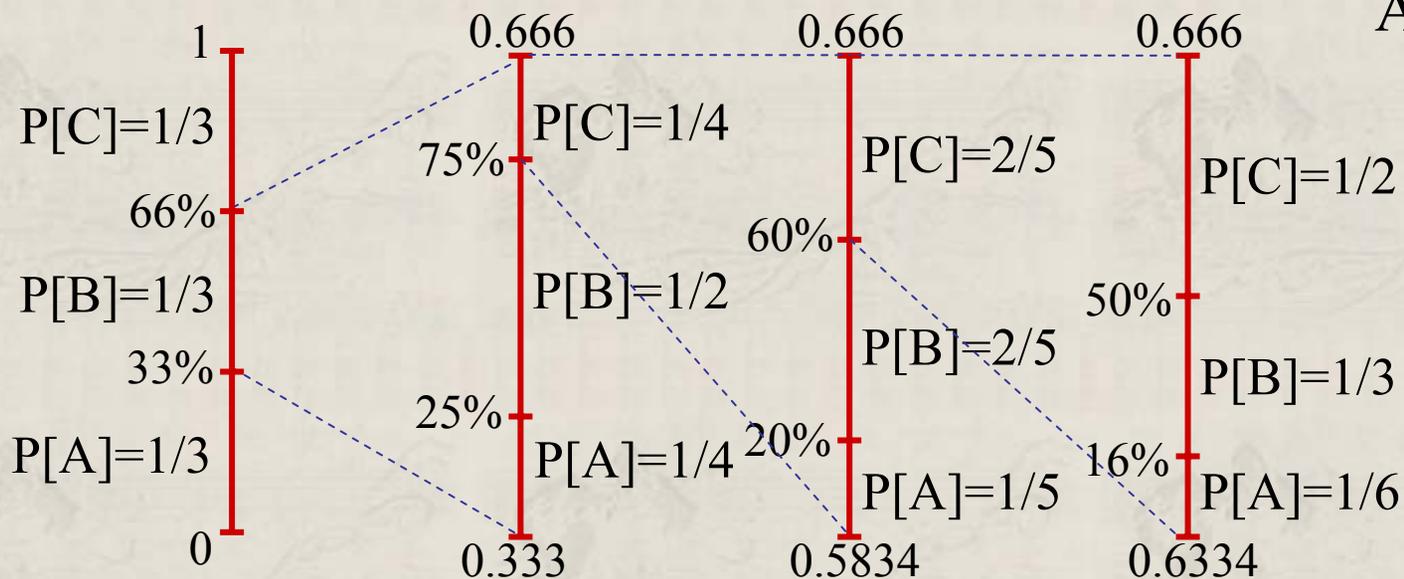
$$low = 0.05 + 0.2 \times 0.05 = 0.06$$

$$0.25 \leq \frac{0.071334... - 0.06}{0.04} = 0.2834 \leq 0.35 \Rightarrow X_3...$$

# Adaptive Arithmetic Coding

- Three symbols {A, B, C}. Encode: BCCB...

A	B	C



Final interval = [0.6390, 0.6501)

Decode?

output =  $0.6396_{10} =$

$$2^{-1} + 2^{-3} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} = 0.1010001111_2$$

# Arithmetic Coding: Notes

---

- ◆ Size of final interval =  $\prod_i P[X_i]$
- ◆ Symbol  $X_i$  of probability  $P[X_i]$  contributes  $\log_2 P[X_i]$  bits to the output
- ◆ Arithmetic coding approaches entropy!
- ◆ Near-optimal: finite-precision arithmetic, a whole number of bits or bytes must be sent
- ◆ Implementation issues:
  - Incremental output: should not wait until the end of the compressed bit-stream; prefer incremental transmission scheme
  - Prefer integer implementations by appropriate scaling

# Run-Length Coding

---

- ◆ Main idea
  - Encoding long runs of a single symbol by the length of the run
- ◆ Properties
  - A lossless coding scheme
  - Our first attempt at inter-symbol coding
  - Really effective with transform-based coding since the transform usually produces long runs of *insignificant* coefficients
  - Run-length coding can be combined with other entropy coding techniques (for example, run-length and Huffman coding in JPEG)

# Run-Length Coding

- ◆ Example: How do we encode the following string?

14 0 0 5 0 -3 0 0 0 0 0 1  $\overbrace{0 \dots 0}^{14 \text{ zeros}}$  -1  $\overbrace{0 \dots 0}^{37 \text{ zeros}}$

- ◆ Run-length coding:

(run-length, size) binary amplitude value

number of consecutive zeros  
before current non-zero symbol

number of bits needed to  
encode this non-zero symbol

actual value of the  
non-zero symbol in binary

(0,4) 14 (2,3) 5 (1,2) -3 (5,1) 1 (14,1) -1 (0,0)

# Run-Length Coding

14 0 0 5 0 -3 0 0 0 0 0 1 0 ... 0 -1 0 ... 0

14 zeros                      37 zeros

always 1, no need to encode

(run-length, size) binary value



0: positive  
1: negative

$$\text{binary of } S = \begin{cases} S - 2^{\text{size}-1}, & S > 0 \\ |S|, & \text{otherwise} \end{cases}$$

(0,4) 14    (2,3) 5    (1,2) -3    (5,1) 1    (14,1) -1    (0,0)

Huffman or  
arithmetic coding

raw binary

4	000
3	01
2	00
1	0
-1	1
-2	10
-3	11
-4	100

# Quantization



Trac D. Tran

ECE Department

The Johns Hopkins University

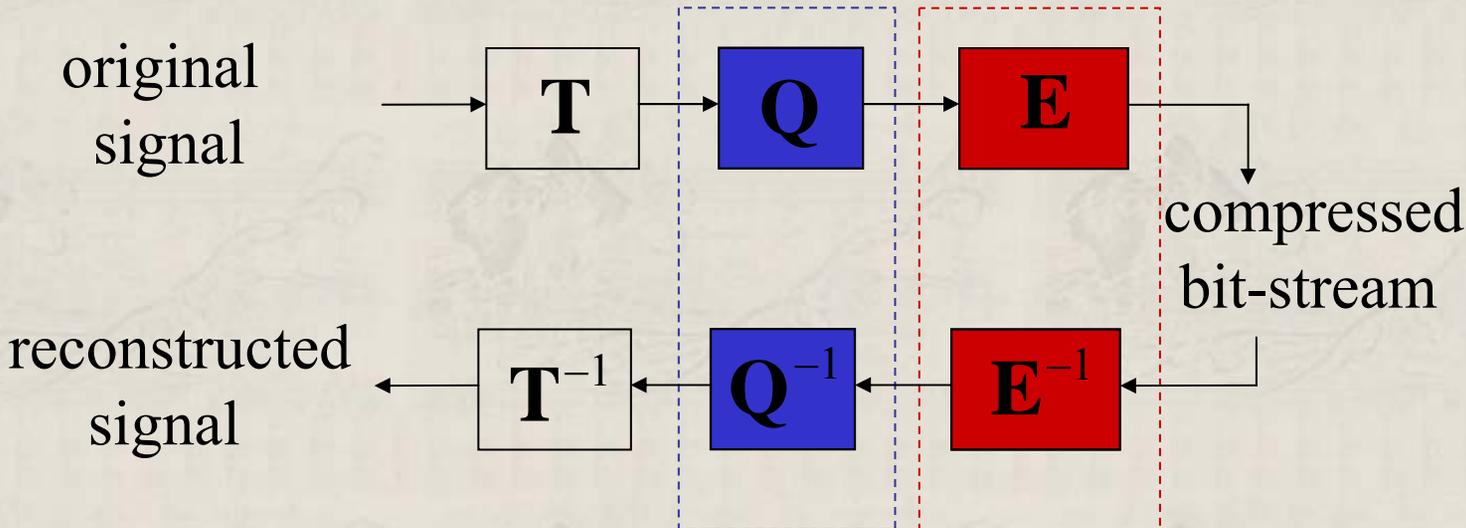
Baltimore, MD 21218

# Outline

---

- ◆ Review
- ◆ Quantization
  - Nonlinear mapping
  - Forward and inverse quantization
  - Quantization errors
    - Clipping error
    - Approximation error
    - Error model
  - Optimal scalar quantization
  - Examples

# Reminder



- Information theory
- VLC
- Quantization
- Huffman
- Arithmetic
- Run-length

# Quantization

---

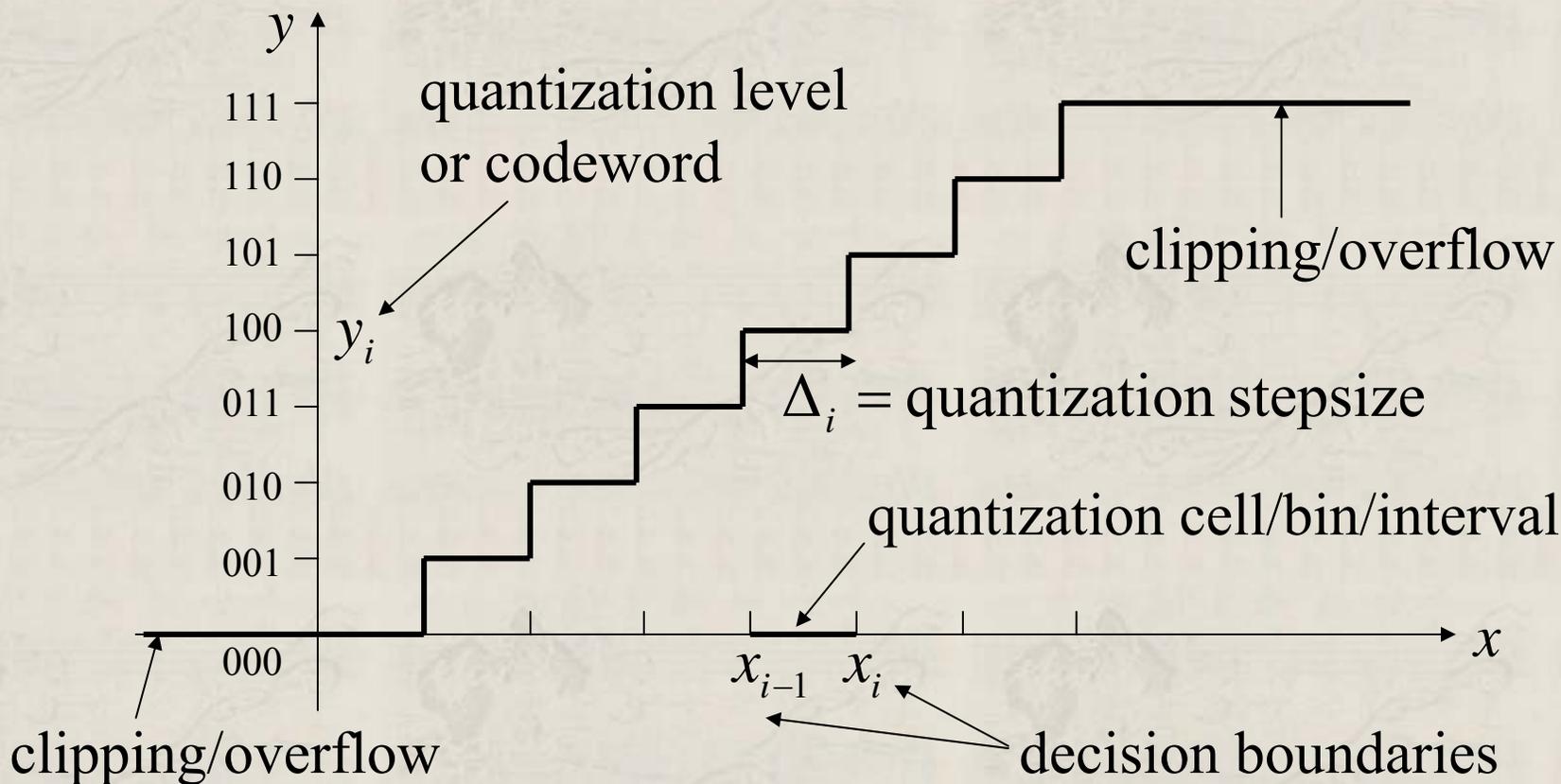
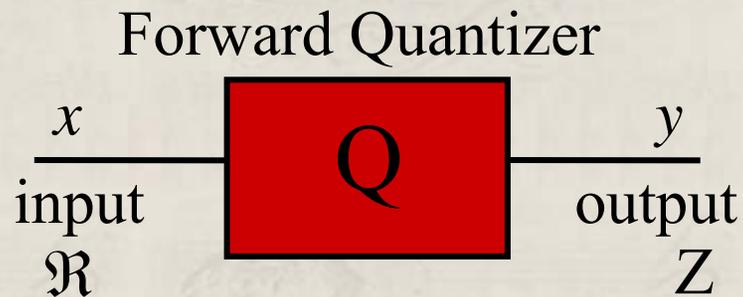
- ◆ Entropy coding techniques
  - Perform lossless coding
  - No flexibility or trade-off in bit-rate versus distortion
- ◆ Quantization
  - Lossy non-linear mapping operation: a range of amplitude is mapped to a unique level or codeword
  - Approximation of a signal source using a finite collection of discrete amplitudes
  - Controls the rate-distortion trade-off
  - Applications
    - A/D conversion
    - Compression

# Typical Quantizer

$$x \in I_i = [x_{i-1}, x_i)$$

$$\text{i.e., } I_i = \{x \mid x_{i-1} \leq x < x_i\}$$

$$\Rightarrow y = y_i$$



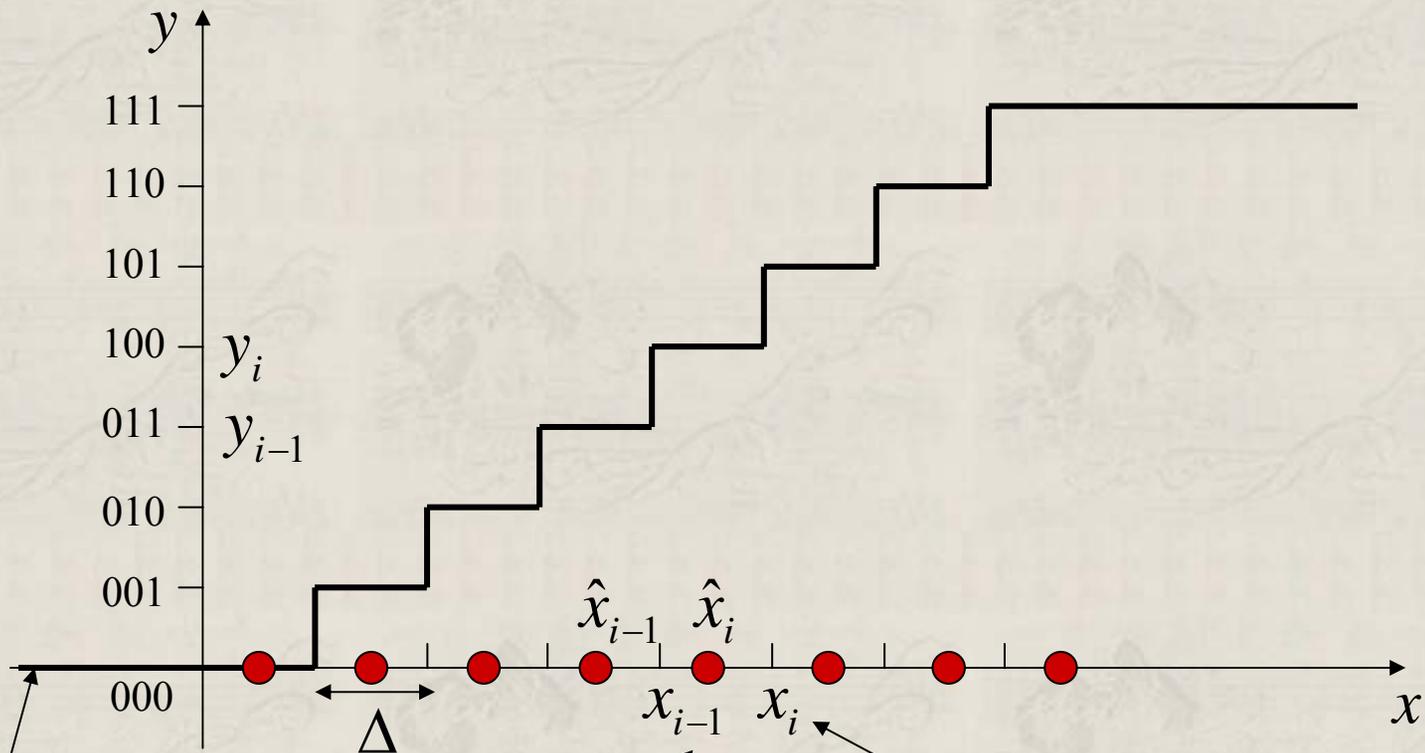
# Typical Inverse Quantizer

- Typical reconstruction

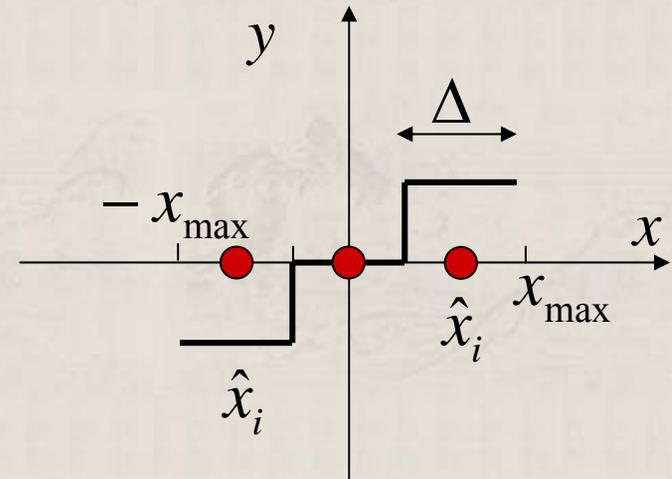
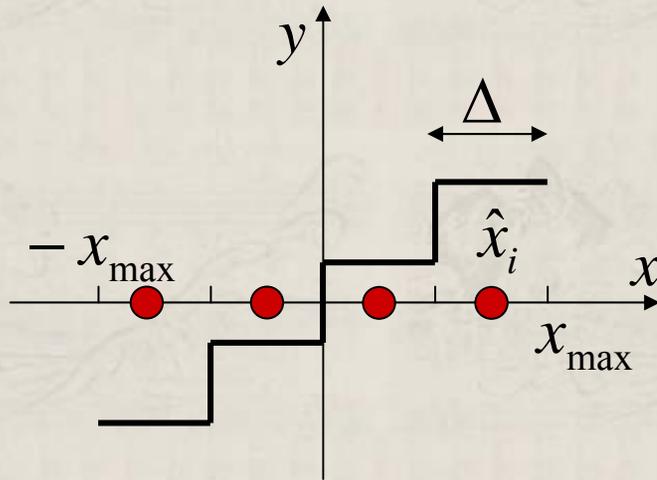
$$\hat{x}_i = \frac{x_i + x_{i-1}}{2}$$

- Quantization error

$$q = \hat{x} - x$$



# Mid-rise versus Mid-tread



## Uniform Midrise Quantizer

- ◆ Popular in ADC
- ◆ For a  $b$ -bit midrise

$$\Delta = \frac{2x_{\max}}{2^b}$$

## Uniform Midtread Quantizer

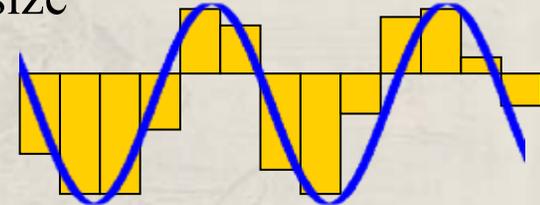
- ◆ Popular in compression
- ◆ For a  $b$ -bit midtread

$$\Delta = \frac{2x_{\max}}{2^b - 1}$$

# Quantization Errors

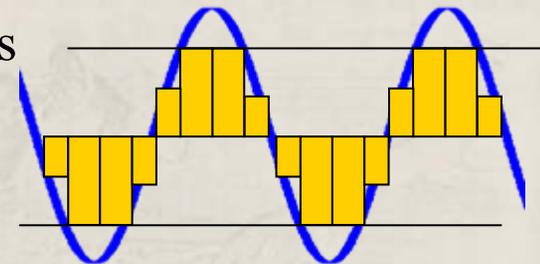
## ◆ Approximation error

- Lack of quantization resolution, too few quantization levels, too large quantization step-size
- Causes staircase effect
- Solution: increases the number of quantization levels, and hence, increase the bit-rate



## ◆ Clipping error

- Inadequate quantizer range limits, also known as overflow
- Solution
  - Requires knowledge of the input signal
  - Typical practical range for a zero-mean signal

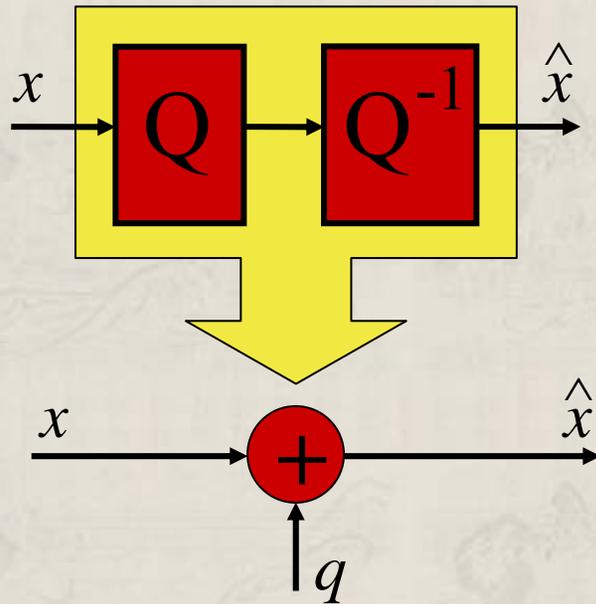


$$x_{\max} = 4x_{RMS}$$

$$x_{\min} = -4x_{RMS}$$

$$x_{RMS} = \sqrt{\frac{1}{T} \int_0^T x^2(t) dt}$$

# Quantization: Error Model



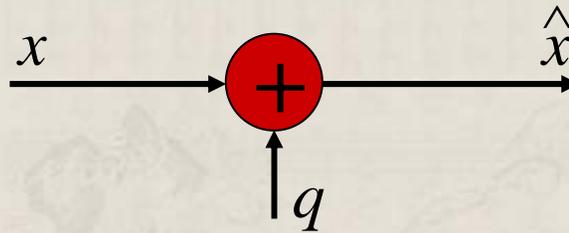
- ◆ Assumptions:
  - $x$  RV with PDF  $f_x(x)$
  - $y$  RV;  $\hat{x}$  RV
  - $q$  0-mean RV independent of  $x$

- ◆ Quantization error:  
 $\hat{x} = x + q \Rightarrow q = \hat{x} - x$

- ◆ Mean-squared distortion measure:

$$\begin{aligned} D(x, \hat{x}) &= E[(\hat{x} - x)^2] = E[\underbrace{(x - \hat{x})^2}_{\text{function of } x}] \\ &= E[q^2] = \text{quantization error variance} \end{aligned}$$

# Quantization Error Variance



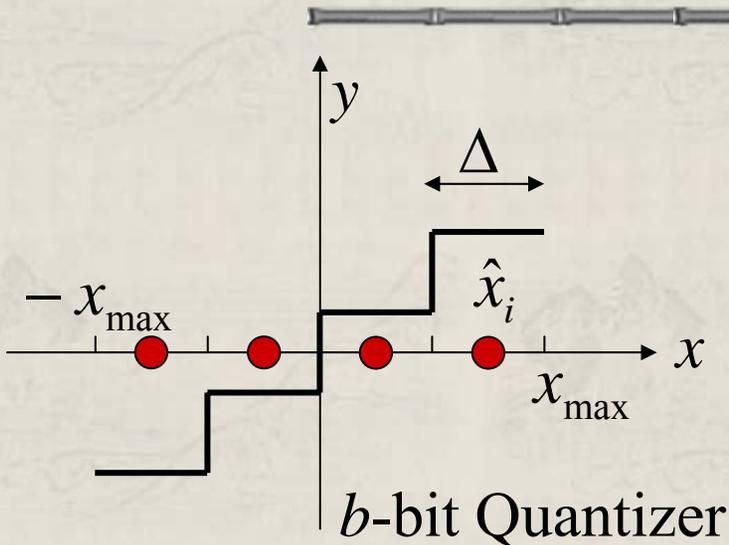
$$D(\hat{x}, x) = E[(x - \hat{x})^2] = \int_{-\infty}^{\infty} (x - \hat{x})^2 f_X(x) dx$$

$$= \int_{x_1}^{x_2} (x - \hat{x}_1)^2 f_X(x) dx + \int_{x_2}^{x_3} (x - \hat{x}_2)^2 f_X(x) dx + \dots$$

$$= \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx$$

would like to minimize

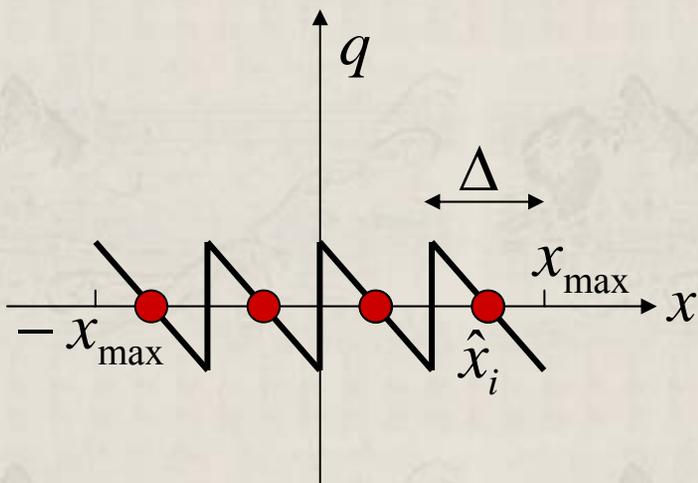
# Uniform Quantization – Bounded Input



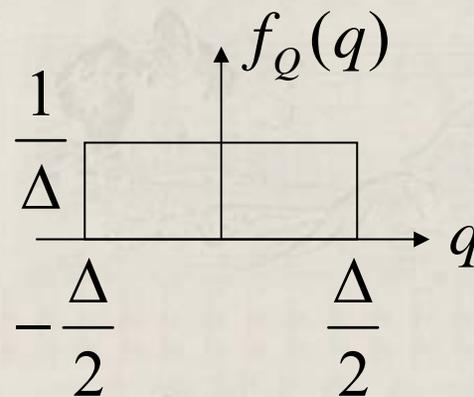
Bounded input :  $-x_{\max} \leq x \leq x_{\max}$

Center reconstruction :  $\hat{x}_i = \frac{x_i + x_{i-1}}{2}$

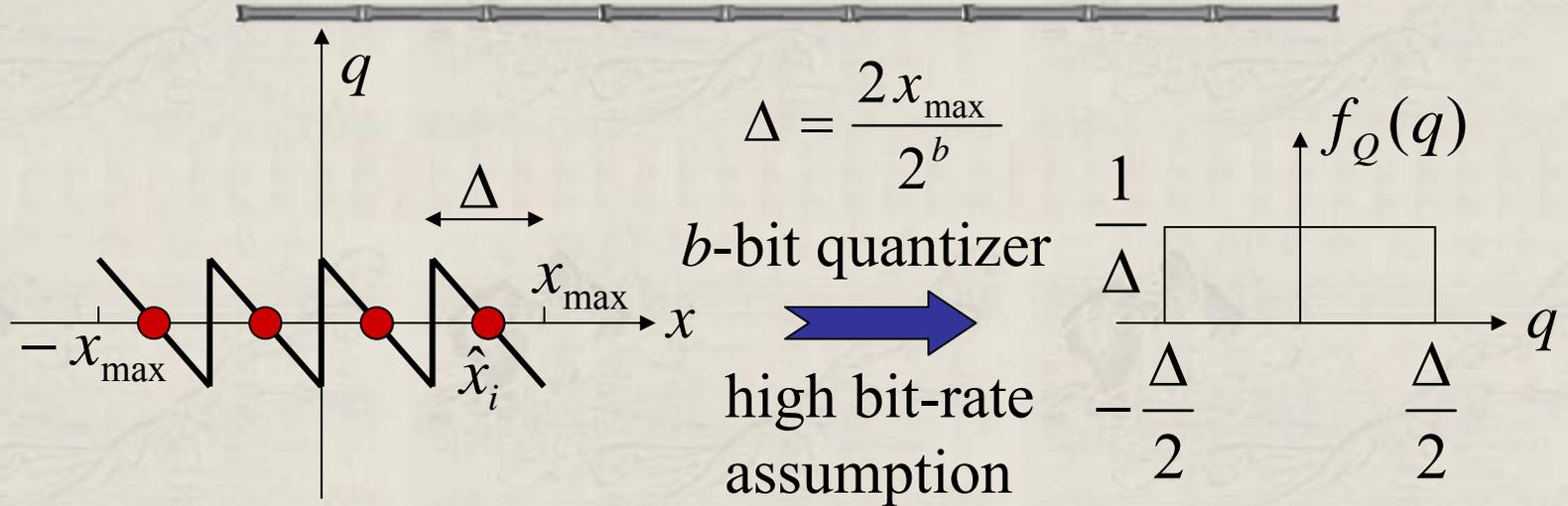
Error bound :  $-\frac{\Delta}{2} \leq q \leq \frac{\Delta}{2}$



high bit-rate  
assumption



# Uniform Quantization – Bounded Input



$$\begin{aligned}
 D(\hat{x}, x) &= E[q^2] = \int_{-\infty}^{\infty} q^2 f_Q(q) dq \\
 &= \int_{-\Delta/2}^{\Delta/2} q^2 \frac{1}{\Delta} dq = \frac{1}{3\Delta} q^3 \Big|_{-\Delta/2}^{\Delta/2} = \frac{2}{3\Delta} \left(\frac{\Delta}{2}\right)^3 = \frac{\Delta^2}{12} = \frac{1}{3} \frac{x_{\max}^2}{2^{2b}}
 \end{aligned}$$

Increase  $b$  by 1 bit/symbol  $\Rightarrow$  double  $L$   
 $\Rightarrow$  reduce  $\Delta$  by a factor of 2

# Signal-to-Noise Ratio

- ◆ Definition of SNR in decibel (dB)

$$SNR_{dB} = 10 \log_{10} \frac{\sigma_x^2}{\sigma_q^2}$$

power of the signal

power of the noise

- ◆ For quantization noise

$$SNR_{dB} = 10 \log_{10} \frac{12\sigma_x^2}{\Delta^2} = 10 \log_{10} 12\sigma_x^2 - 20 \log_{10} \Delta$$

Suppose that we now add 1 more bit to our Q resolution:

$$\begin{aligned} \Delta' &= \frac{\Delta}{2} \Rightarrow SNR'_{dB} = 10 \log_{10} 12\sigma_x^2 - 20 \log_{10} \Delta' \\ &\Rightarrow SNR'_{dB} = 10 \log_{10} 12\sigma_x^2 - 20 \log_{10} \Delta + 20 \log_{10} 2 \\ &= SNR_{dB} + 20 \log_{10} 2 \\ &\Rightarrow SNR'_{dB} \approx SNR_{dB} + 6dB \end{aligned}$$

# Example

Design a 3-bit uniform quantizer for a signal with range  $[0,128]$

- ◆ Maximum possible number of levels:  $L = 2^3 = 8$
- ◆ Quantization stepsize:  $\Delta = \frac{x_{\max} - x_{\min}}{L} = \frac{128}{8} = 16$
- ◆ Quantization levels:  $y_i = \{0,1,2,3,4,5,6,7\}$
- ◆ Reconstruction levels:  $\hat{x}_i = \{8,24,40,56,72,88,104,120\}$
- ◆ Maximum quantization error:  $|q| \leq 8$

# Example of Popular Quantization

- ◆ Round

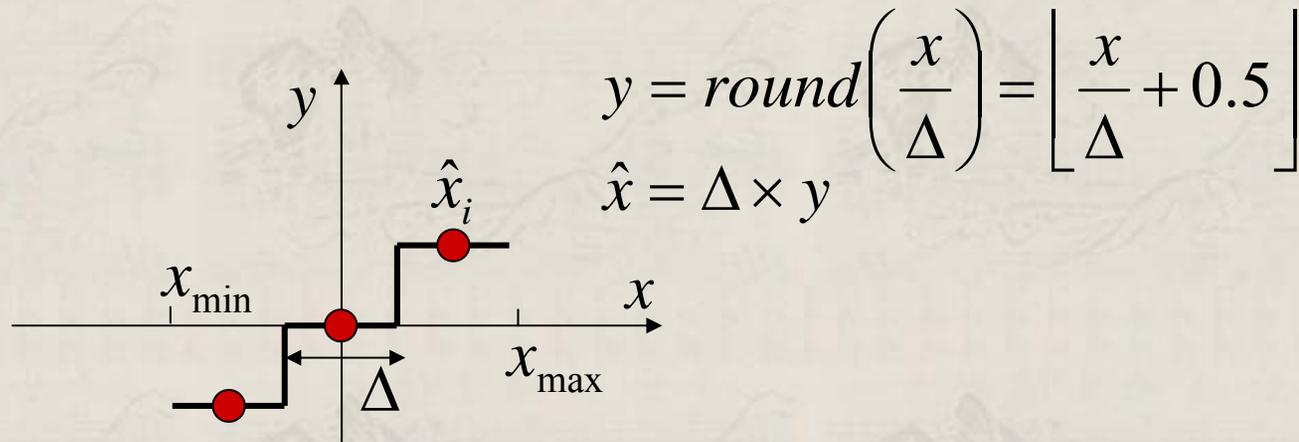
$$y = \text{round}(x) = \text{nearest integer to } x$$

- ◆ Floor

$$y = \text{floor}(x) = \lfloor x \rfloor = \text{largest integer smaller than } x$$

- ◆ Ceiling

$$y = \text{ceil}(x) = \lceil x \rceil = \text{smallest integer larger than } x$$

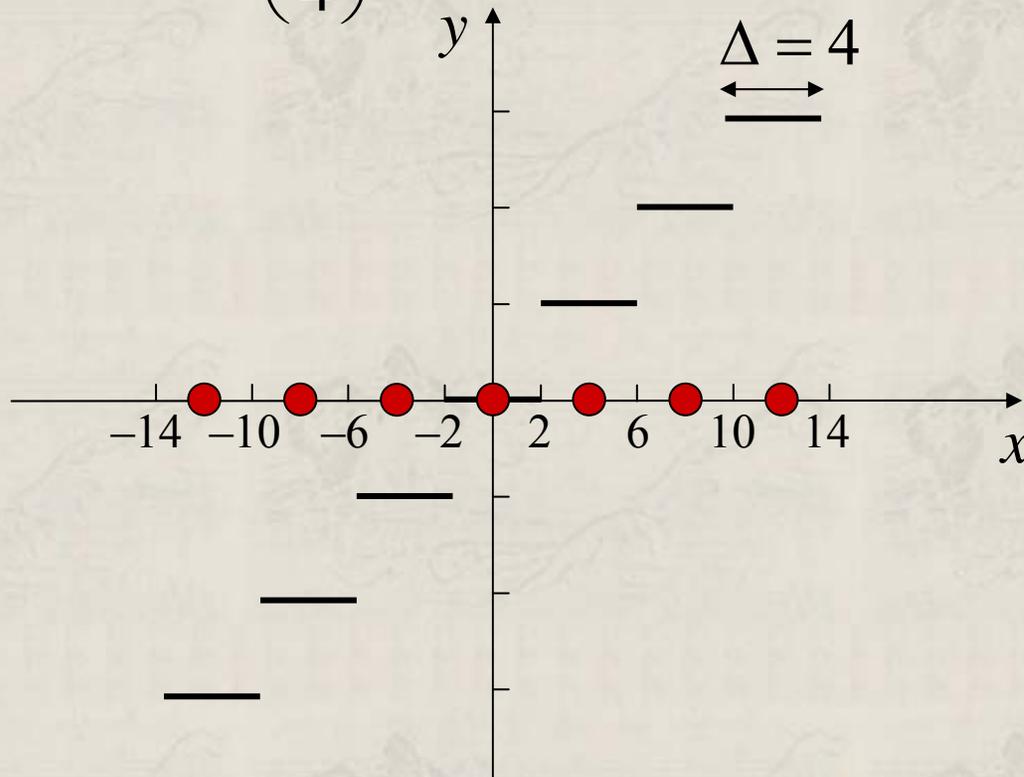


Uniform midtread quantizer from Round and Floor

# Quantization from Rounding

Bounded input :  $x \in (-16, 16)$

$$\Delta = 4; y = \text{round}\left(\frac{x}{4}\right); \hat{x} = 4y$$



Uniform Quantizer, *step-size=4*

# Optimal Scalar Quantization

- ◆ Problem Statement:

$x$  RV with known PDF  $f_X(x)$

Find  $x_k$  &  $\hat{x}_k$  such that  $D(\hat{x}, x)$  is minimized

- ◆ Optimal Encoder for a Given Decoder:

Given  $\hat{x}_k$ , find  $x_k$  such that  $D(\hat{x}, x)$  is minimized

$$\text{Minimize } D(\hat{x}, x) = \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \quad \text{w.r.t } x_k$$

- ◆ Notes:

- Non-uniform quantizer under consideration
- Reconstruction can be anywhere, not necessarily the center of the interval

# Optimal Scalar Quantization

$$\text{Minimize } D(\hat{x}, x) = \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \quad \text{w.r.t } x_k$$

$$\begin{aligned} \frac{\delta}{\delta x_k} D(\hat{x}, x) &= \frac{\delta}{\delta x_k} \left\{ \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \right\} \\ &= \frac{\delta}{\delta x_k} \left\{ \int_{x_{k-1}}^{x_k} (x - \hat{x}_{k-1})^2 f_X(x) dx + \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \right\} \end{aligned}$$

Fundamental Theorem of Calculus

$$F(x) = \int_C^x f(t) dt \Rightarrow \frac{d}{dx} F(x) = \frac{d}{dx} \int_C^x f(t) dt = f(x)$$

$$= (x_k - \hat{x}_{k-1})^2 f_X(x_k) - (x_k - \hat{x}_k)^2 f_X(x_k) = 0$$

$$\Rightarrow x_k - \hat{x}_{k-1} + x_k - \hat{x}_k = 0 \Rightarrow x_k = \frac{\hat{x}_{k-1} + \hat{x}_k}{2}$$

Nearest  
Neighbor  
Rule

# Optimal Scalar Quantization

- ◆ Optimal Decoder for a Given Encoder:

Given  $x_k$ , find  $\hat{x}_k$  such that  $D(\hat{x}, x)$  is minimized

$$\text{Minimize } D(\hat{x}, x) = \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \quad \text{w.r.t } \hat{x}_k$$

$$\begin{aligned} \frac{\delta}{\delta \hat{x}_k} D(\hat{x}, x) &= \frac{\delta}{\delta \hat{x}_k} \left\{ \sum_{k=1}^L \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \right\} \\ &= \frac{\delta}{\delta \hat{x}_k} \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx = \int_{x_k}^{x_{k+1}} -2(x - \hat{x}_k) f_X(x) dx = 0 \end{aligned}$$

$$\Rightarrow \int_{x_k}^{x_{k+1}} x f_X(x) dx - \int_{x_k}^{x_{k+1}} \hat{x}_k f_X(x) dx = 0$$

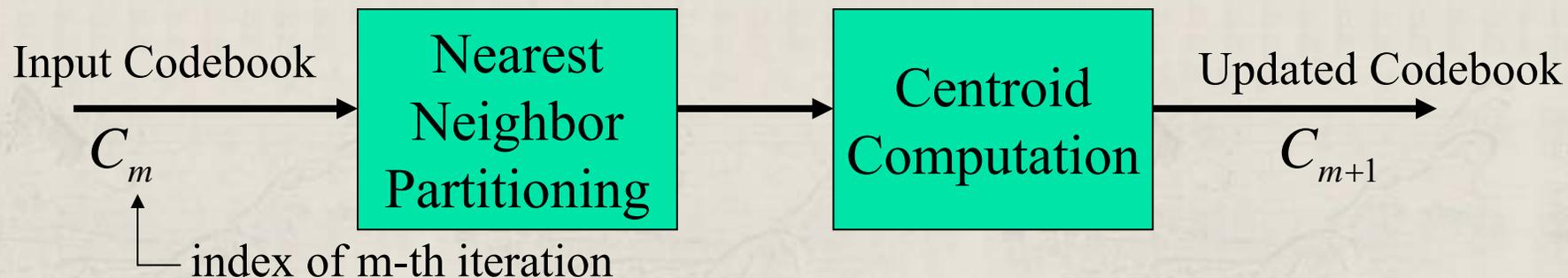
$$\Rightarrow \hat{x}_k = \frac{\int_{x_k}^{x_{k+1}} x f_X(x) dx}{\int_{x_k}^{x_{k+1}} f_X(x) dx}$$

Centroid  
Rule

Optimal reconstruction value  $\hat{x}_k$  is the centroid of  $f_X(x)$  in  $[x_k, x_{k+1}]$

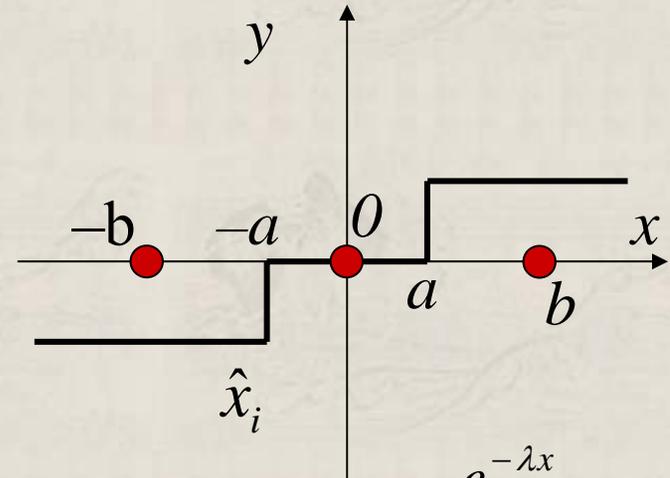
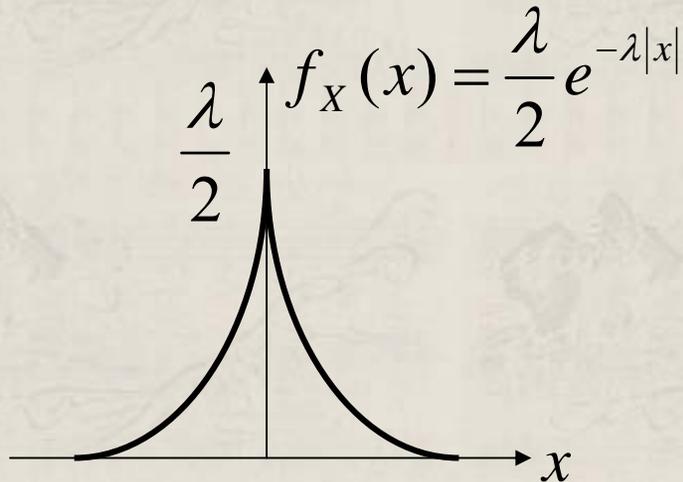
# Lloyd-Max Quantizer

- ◆ Unfortunately, we need  $\hat{x}_k$  to solve for  $x_k$  and  $x_k$  to solve for  $\hat{x}_k$ !
- ◆ Main idea [Lloyd 1957] [Max 1960]
  - solving these 2 equation iteratively until D converges



- ◆ Assumptions
  - Input PDF is known and stationary
  - Entropy has not been taken into account

# Example

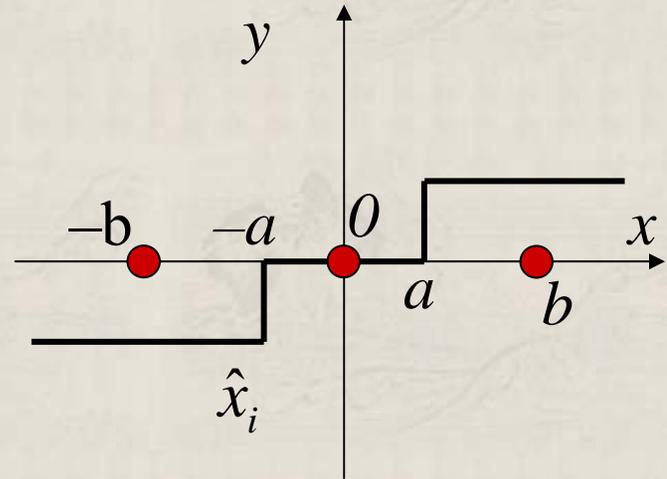
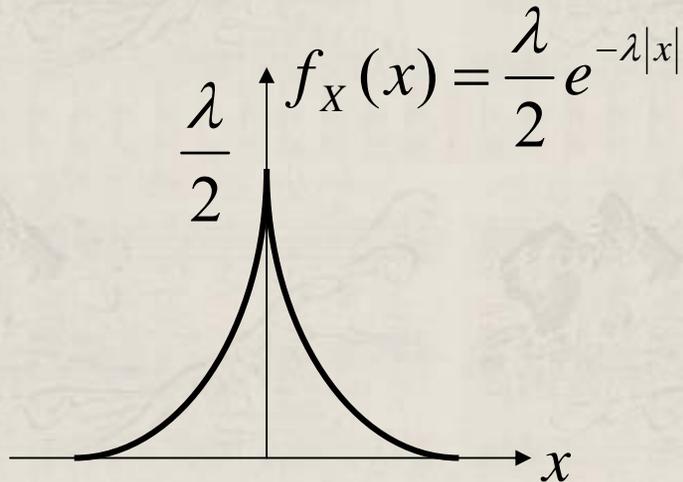


$$\int e^{-\lambda x} dx = -\frac{1}{\lambda} e^{-\lambda x} + C$$

$$\int x e^{-\lambda x} dx = -\frac{e^{-\lambda x}}{\lambda^2} (\lambda x + 1) + C$$

$$\begin{aligned} b &= \frac{\int_a^\infty x \frac{\lambda}{2} e^{-\lambda x} dx}{\int_a^\infty \frac{\lambda}{2} e^{-\lambda x} dx} = \frac{-\frac{e^{-\lambda x}}{2\lambda} (\lambda x + 1) \Big|_a^\infty}{-\frac{1}{2} e^{-\lambda x} \Big|_a^\infty} \\ &= \frac{\frac{e^{-\lambda a}}{2\lambda} (\lambda a + 1)}{\frac{1}{2} e^{-\lambda a}} = \frac{1}{\lambda} (\lambda a + 1) = \boxed{a + \frac{1}{\lambda}} \end{aligned}$$

# Example

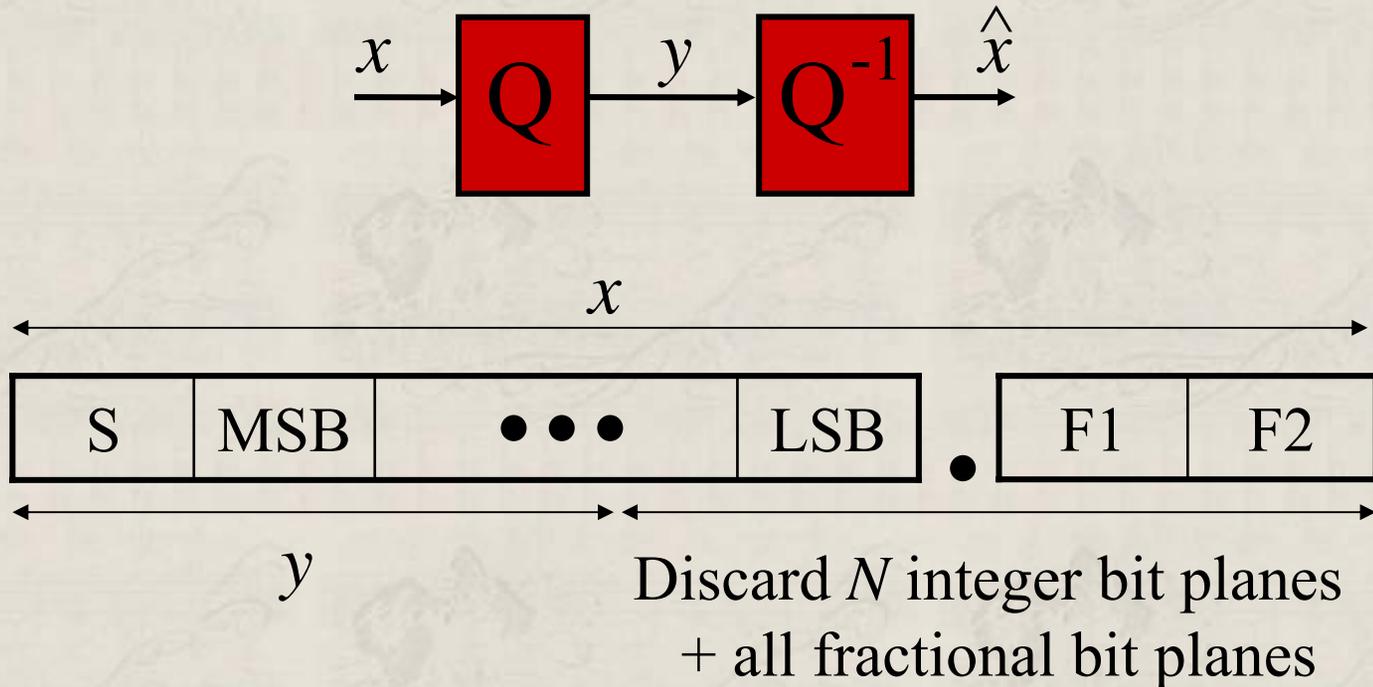


Centroid Rule :  $b = a + \frac{1}{\lambda}$

Nearest Neighbor Rule :  $a = \frac{0+b}{2} = \frac{b}{2} \Rightarrow a = \frac{1}{2} \left( a + \frac{1}{\lambda} \right) \Rightarrow a = \frac{1}{\lambda}$

Distortion :  $D = \int_{-a}^a (x-0)^2 f_X(x) dx + 2 \int_a^{\infty} (x-b)^2 f_X(x) dx$

# Embedded Quantization

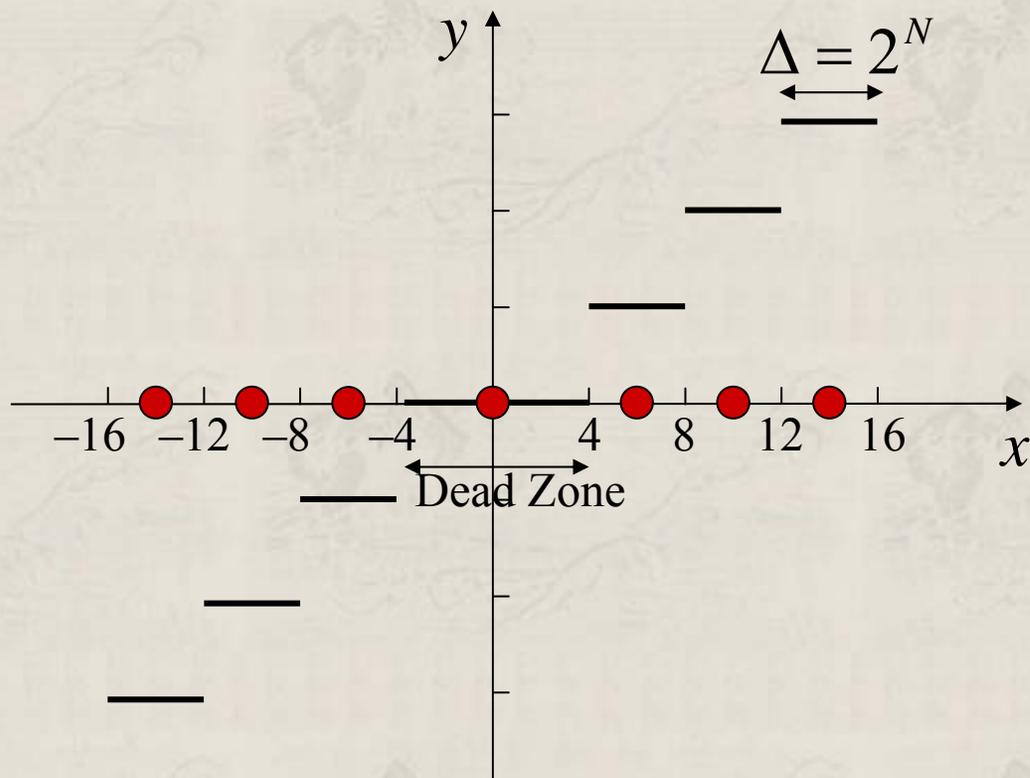


- ◆ Also called bit-plane quantization, progressive quantization
- ◆ Most significant information is transmitted first
- ◆ JPEG2000 quantization strategy

# Embedded Forward Quantization

Bounded input :  $x \in (-16, 16)$

$N = 2$  least significant bit planes discarded



Embedded Quantizer,  $N=2$

# Embedded Inverse Quantization

1
0
1
1
0

Original  
symbol  
 $x = 22$

1
X
X
X
X

Truncate  
4 bit planes  
Range=[16, 32)  
 $\hat{x} = 24$

1
0
X
X
X

Receive 1  
refinement bit  
Range=[16, 24)  
 $\hat{x} = 20$   
 $= 24 - 4$

1
0
1
X
X

Receive 2  
refinement bits  
Range=[20, 24)  
 $\hat{x} = 22$   
 $= 20 + 2$

- ◆  $N$ -bit-plane truncation = scalar quantization with  $\Delta = 2^N$

# Vector Quantization

- ◆  $n$ -dimensional generalization of scalar quantizer

- ◆  $Q: \mathcal{R}^n \rightarrow C$

$n$ -dimensional  
input vectors

codebook, containing  
code-vectors or codewords

- ◆ Nearest neighbor and centroid rule still apply

