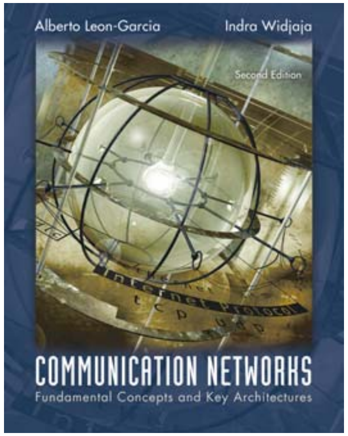
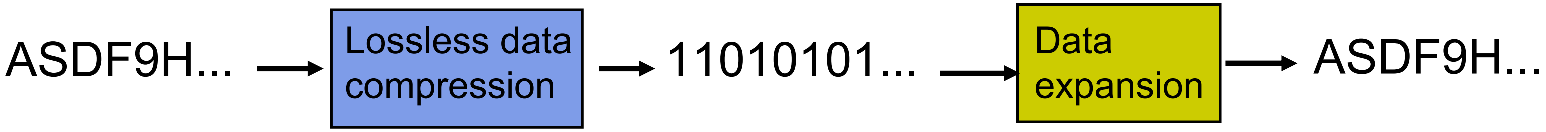
Chapter 12
Multimedia Information



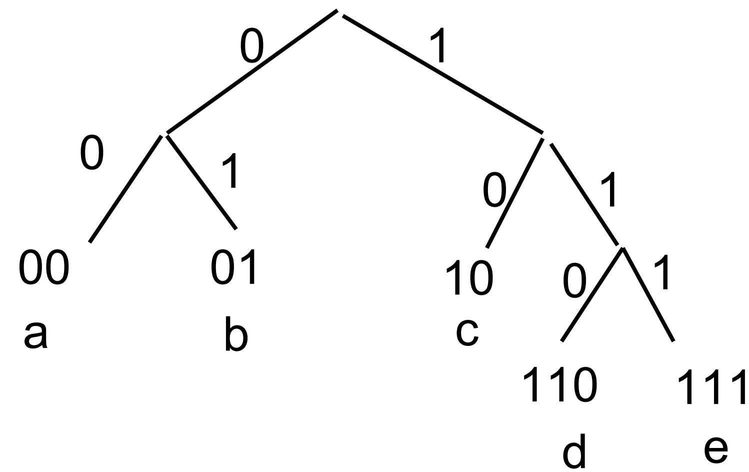
Chapter Figures


ASDF9H...
Lossless data compression
11010101...
Data expansion
ASDF9H...

Assume

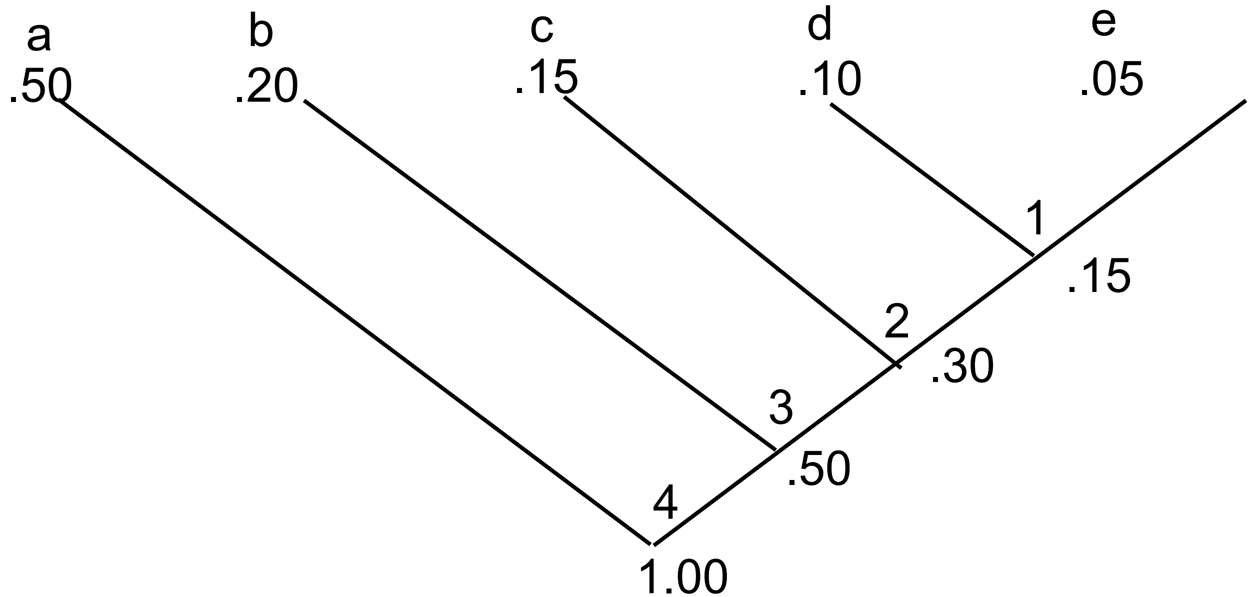
- 5 symbol information source: {a,b,c,d,e}
- symbol probabilities: {1/4, 1/4,1/4,1/8,1/8}

Symbol	Codeword
a	00
b	01
c	10
d	110
e	111

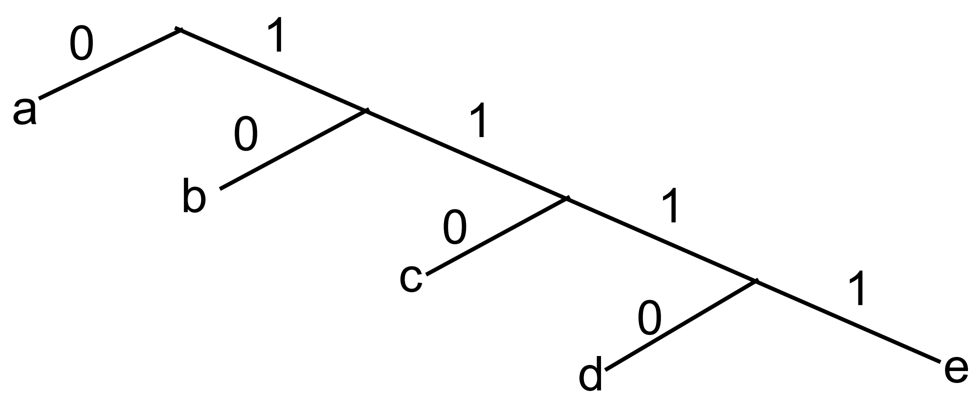


aedbbad.... mapped into 00 111 110 01 01 00 110 ... 17 bits
Note: decoding done without commas or spaces

(a) Building the tree code by Huffman algorithm



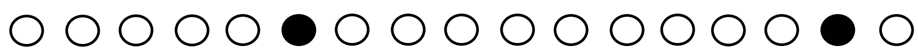
(b) The final tree code



a	0
b	10
c	110
d	1110
e	1111

- “Blank” in strings of alphanumeric information

  ------$5----3-------------$2--------$3------



- “0” (white) and “1” (black) in fax documents

Inputs: ■ □ □ □ □ □ □ □ □ □ □ □ ■ □ □ □ □ □ □ □ ■ □ □ □ ■ □ □ □ ■ □ □ □ □ □ □

Run	Length	Codeword	Codeword (m = 4)
1	0	00..00	0000
01	1	00..01	0001
001	2	00..10	0010
0001	3	00..11	0011
00001	4	.	.
000001	5	.	.
0000001	6	.	.
.	.	.	.
.	.	.	.
000...01	$2^m - 2$	11..10	1110
000...00	run $>2^m - 2$	11..11	1111

← m →

Example: Code 1, $m = 4$

Runs 000…001000…01000…0100…001… 137 bits

25 57 36 15

Symbols >14 10>14>14>14 12 >14>14 6 >14 0

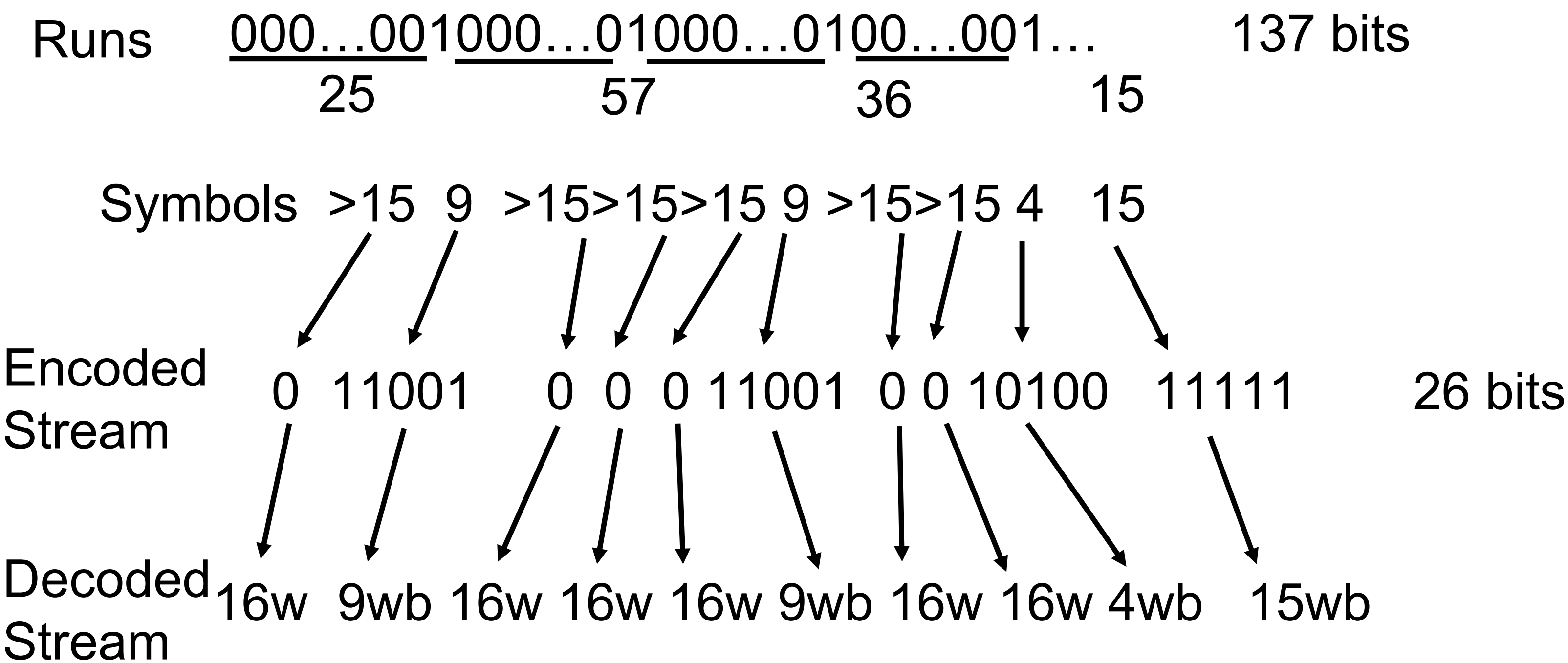
1111 1010 1111 1111 1111 1100 1111 1111 0110 1111 0000 44 bits

15w 10wb 15w 15w 15w 12wb 15w 15w 6wb 15w b

Inputs: ■ □ □ □ □ □ □ □ □ □ □ □ ■ □ □ □ □ □ □ □ ■ □ □ □ ■ □ □ □ ■ □ □ □ □ □ □

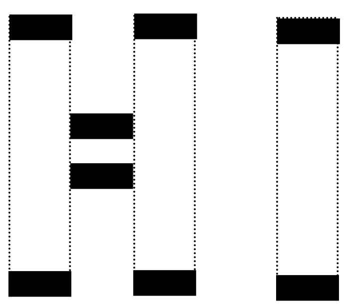
Run	Length	Codeword	Codeword (m = 4)
1	0	10..00	10000
01	1	10..01	10001
001	2	10..10	10010
0001	3	10..11	10011
00001	4	.	.
000001	5	.	.
0000001	6	.	.
.	.	.	.
.	.	.	.
000... 01	$2^m - 1$	11..11	11111
000... 00	run $>2^m - 1$	0	0
		←$m + 1$→	

Example: Code 2, $m = 4$



(a) Huffman code applied to white runs and black runs

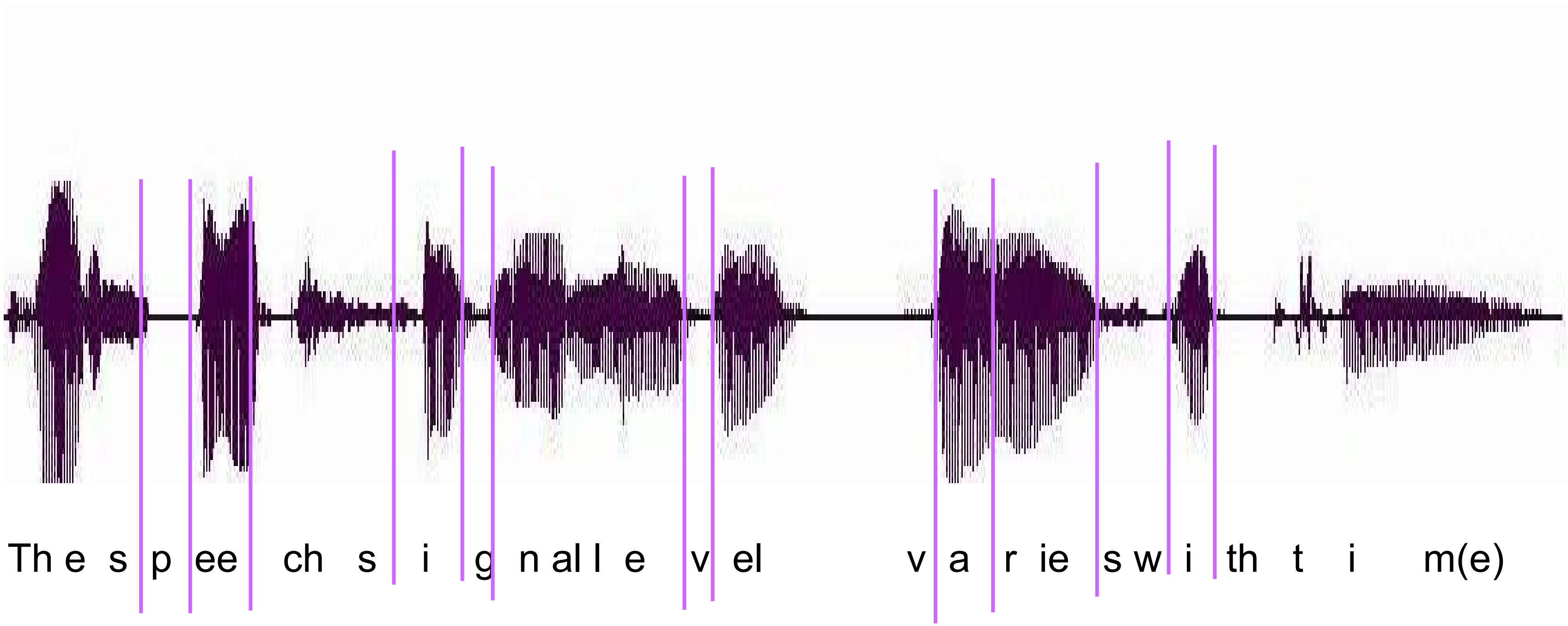
(b) Encode differences between consecutive lines

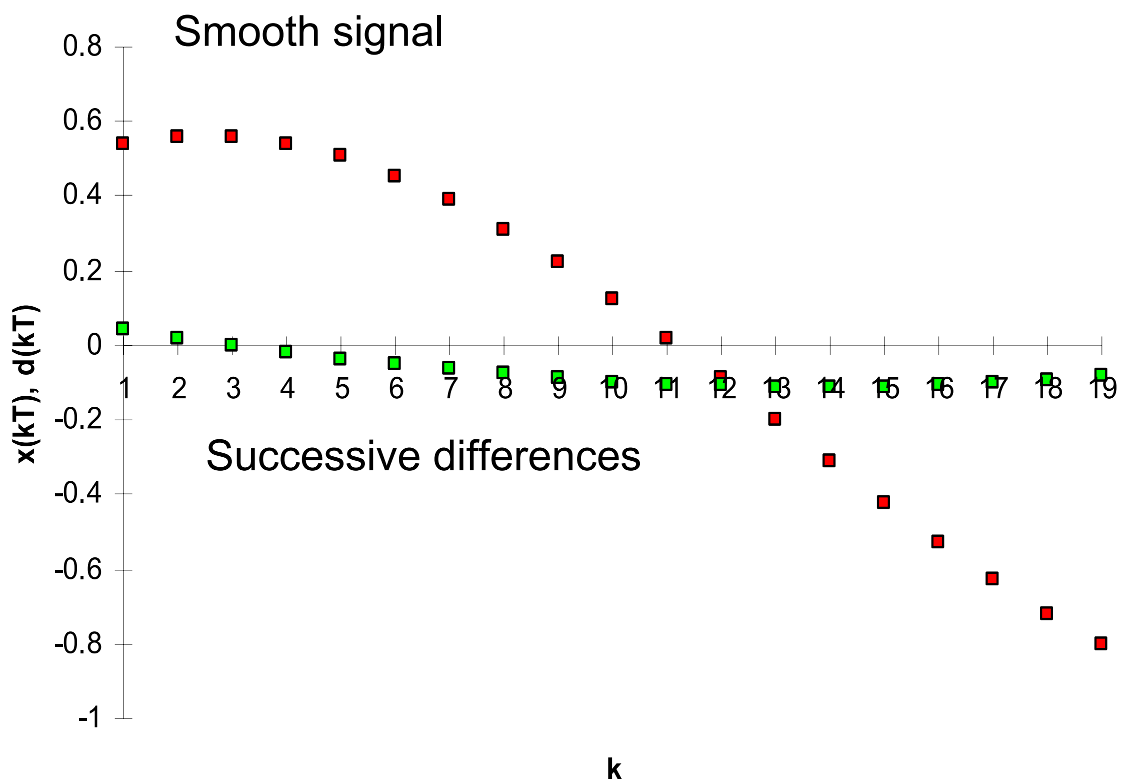


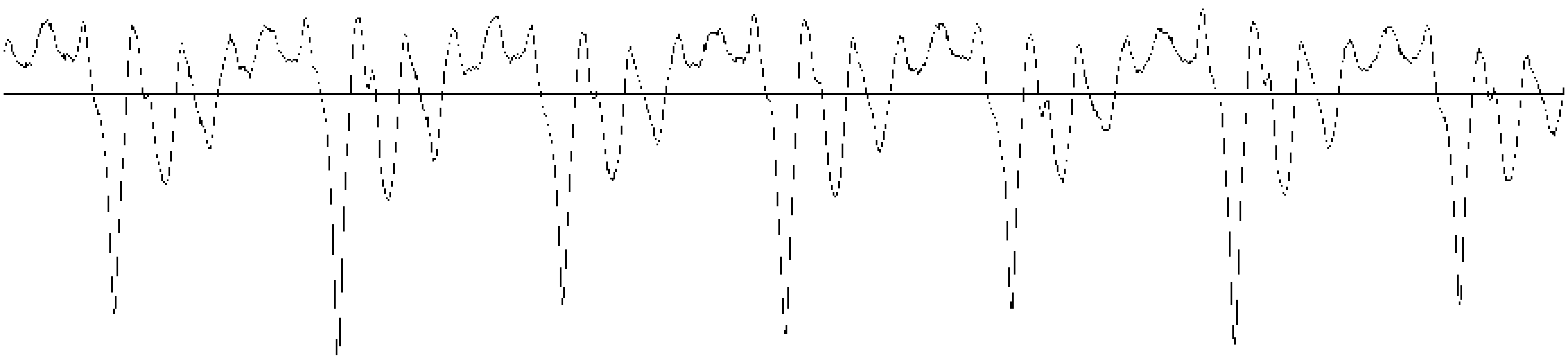
"All tall We all are tall. All small We all are small."

Can be mapped into:

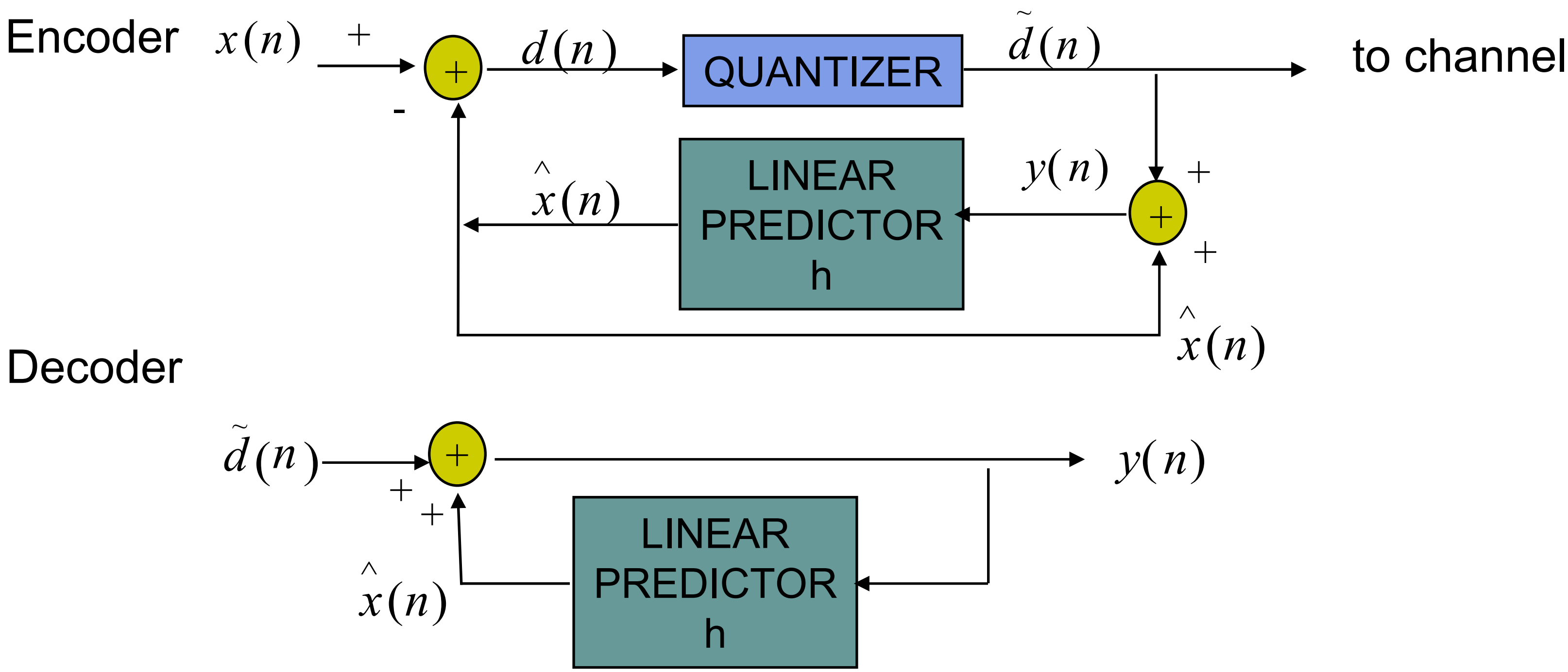
"All_ta[2,3]We_[6,4]are[4,5]._[1,4]sm[6,15][31,5]."


Th e s p ee ch s i g n al l e v el v a r ie s w i th t i m(e)


Smooth signal
Successive differences
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
x(kT), d(kT)
k

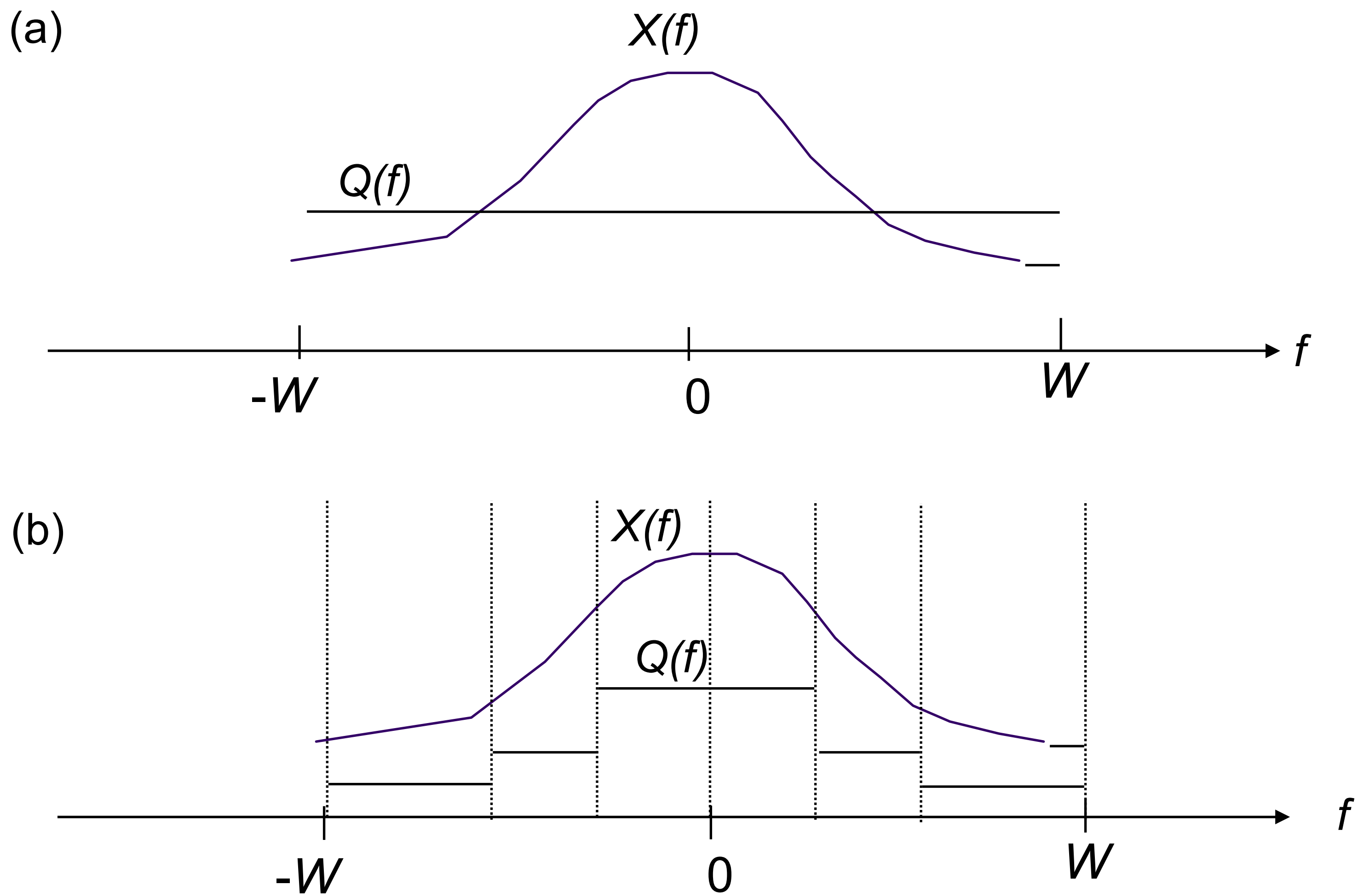


Quantize the difference between prediction and actual signal:

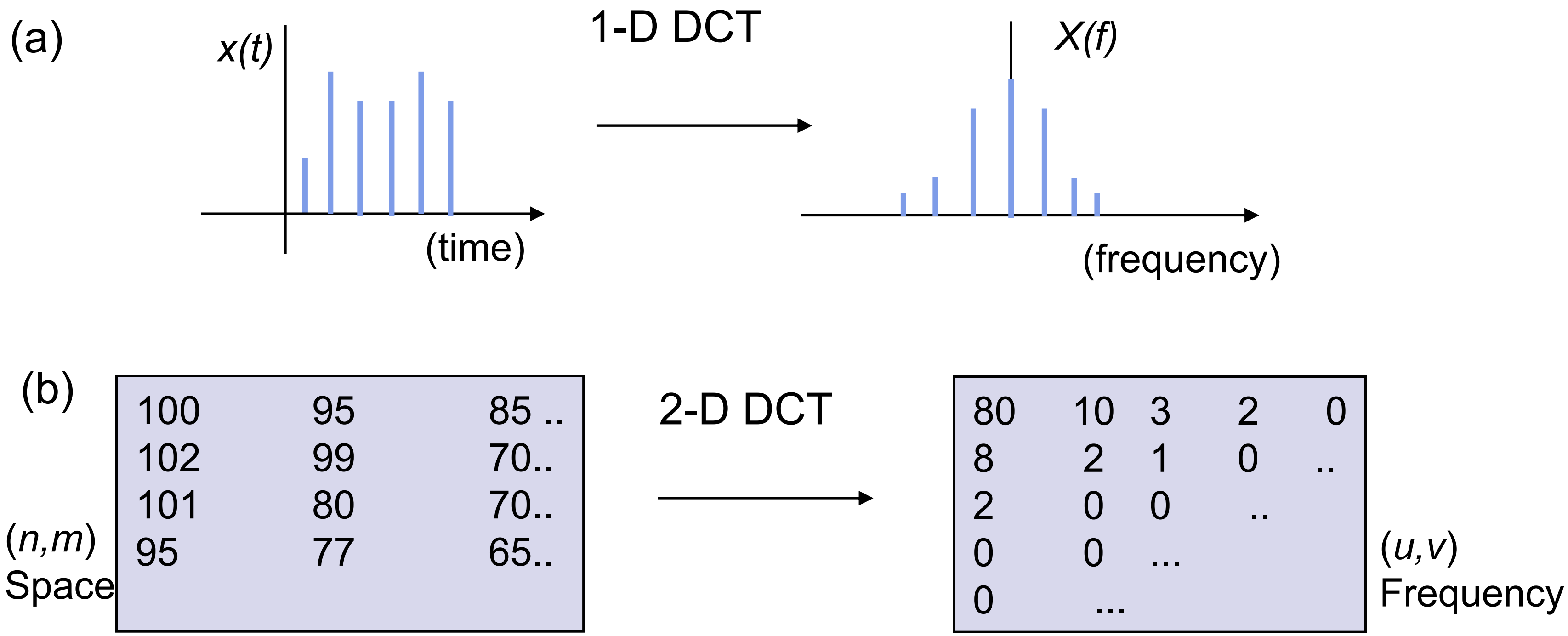


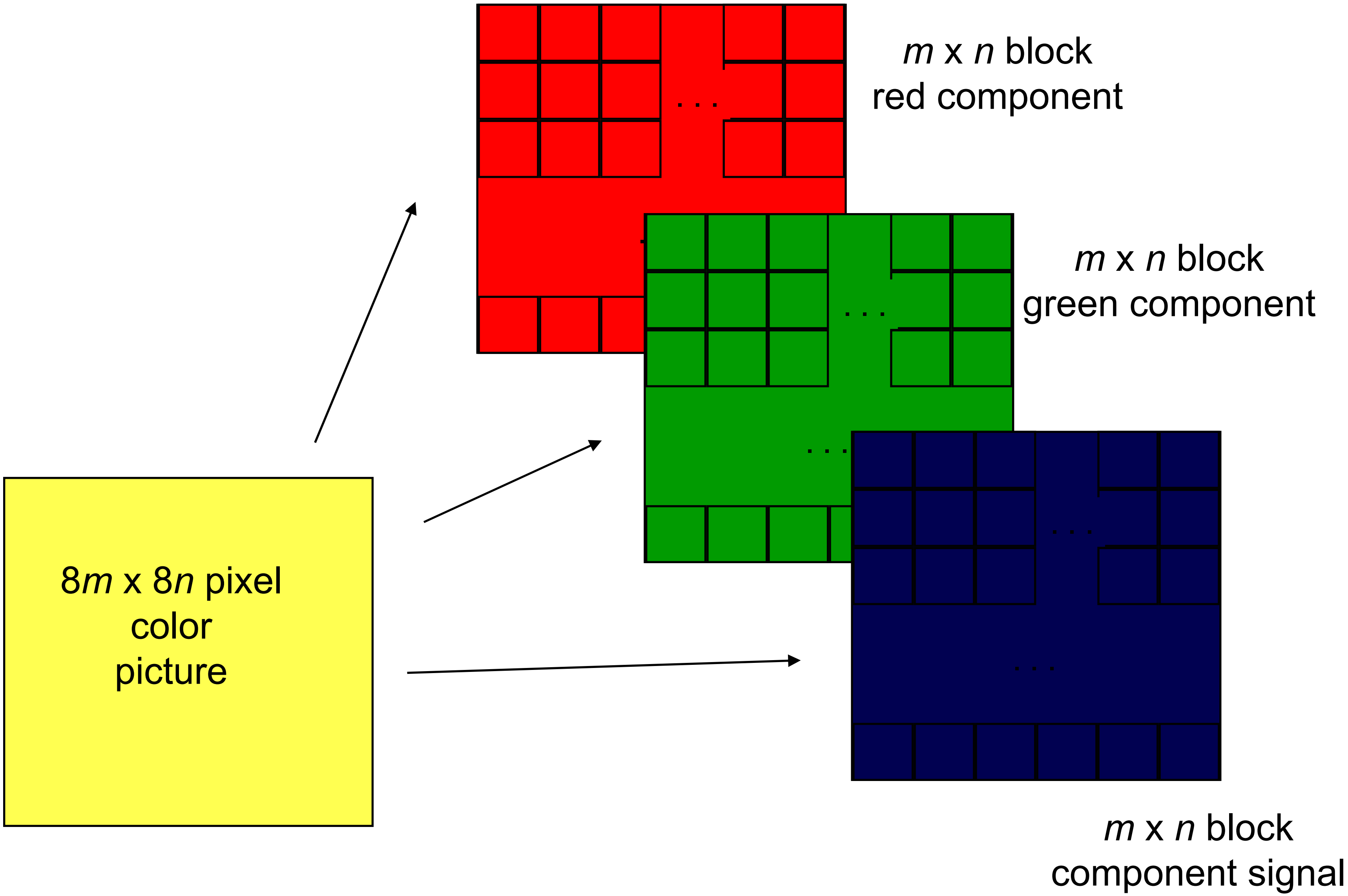
$$y(n) - x(n) = \hat{x}(n) + \tilde{d}(n) - x(n) = \tilde{d}(n) - d(n) = e(n)$$

The end-to-end error is only the error introduced by the quantizer!



(a)
X(f)
Q(f)
f
-W
0
W
(b)
X(f)
Q(f)
f
-W
0
W


(a)
x(t)
(time)
1-D DCT
X(f)
(frequency)
(b)
100 95 85 ..
102 99 70..
101 80 70..
95 77 65..
(n,m) Space
2-D DCT
80 10 3 2 0
8 2 1 0 ..
2 0 0 ..
0 0 ...
0 ...
(u,v) Frequency


m x n block
red component
m x n block
green component
8m x 8n pixel
color
picture
m x n block
component signal

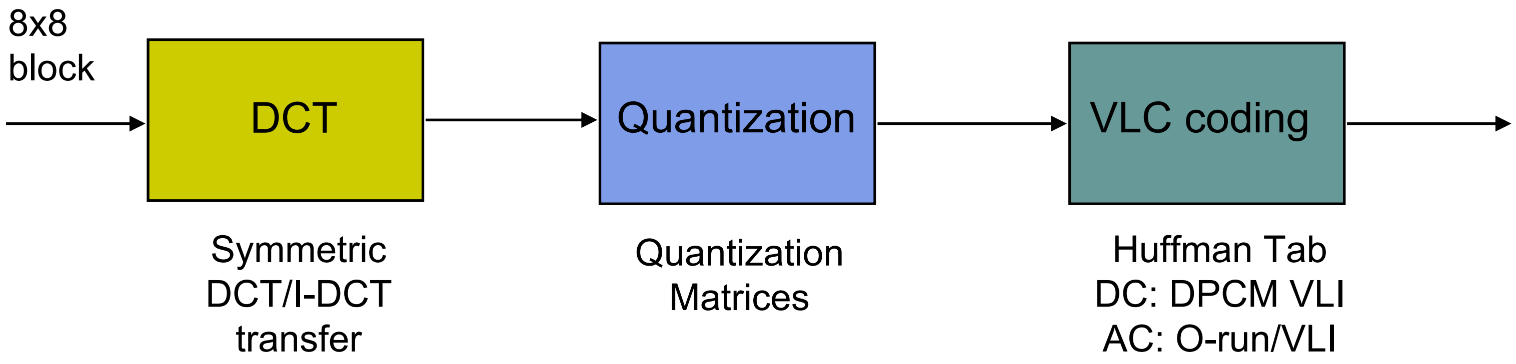
180	150	115	100	100	100	100	100
250	180	128	100	100	100	100	100
190	170	120	100	100	100	100	100
160	130	110	100	100	100	100	100
110	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100

8x8 block of 8-bit pixel values

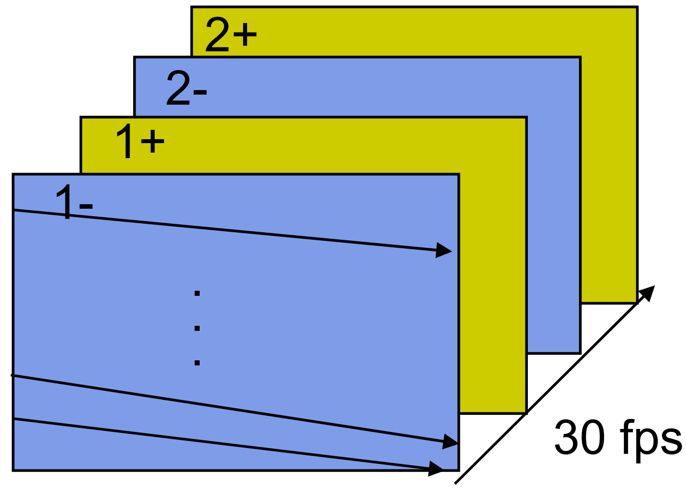
DCT →

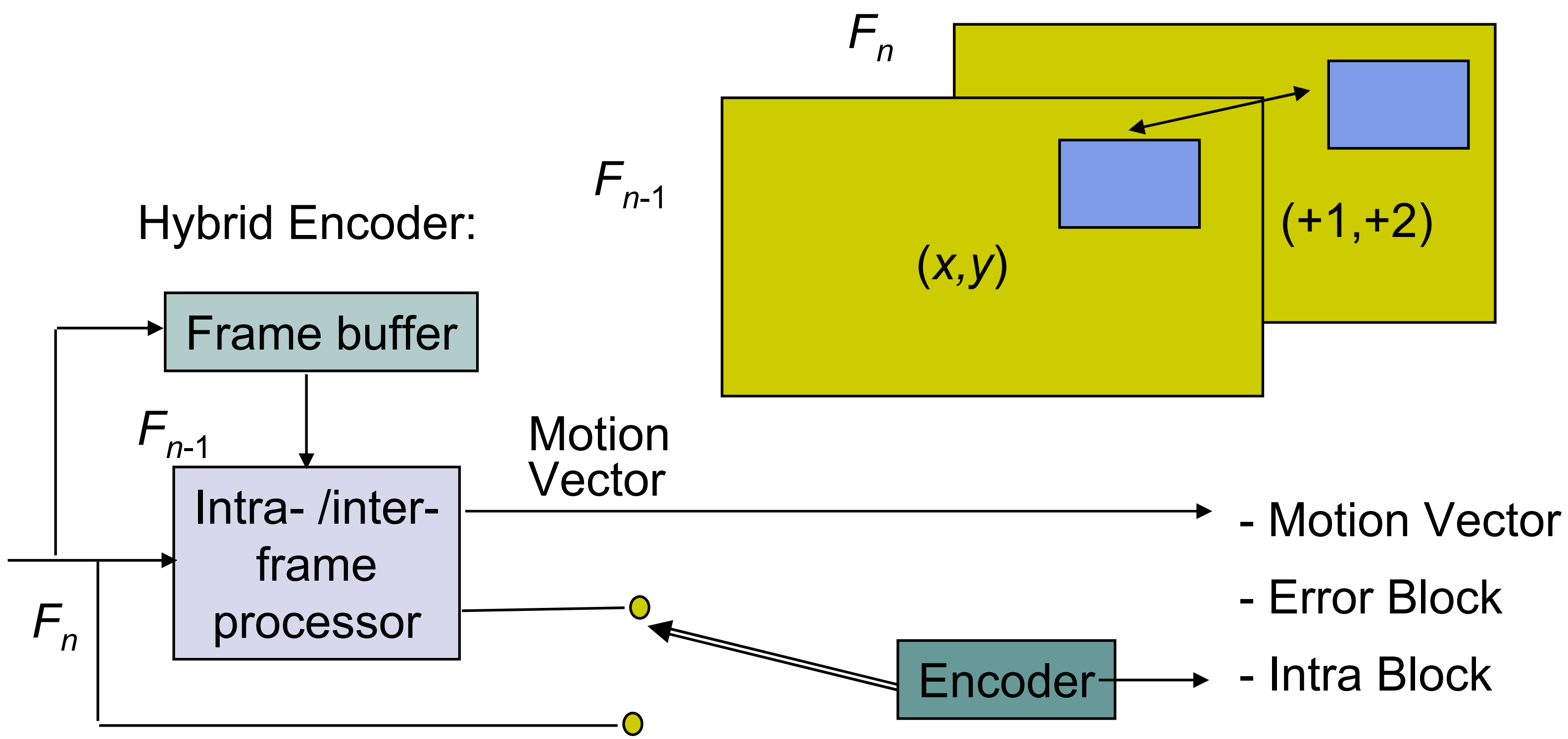
111	22	15	5	1	0	0	0
14	17	10	4	1	0	0	0
2	2	1	0	0	0	0	0
-4	-4	-2	-1	0	0	0	0
-3	-3	-1	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized DCT Coefficients

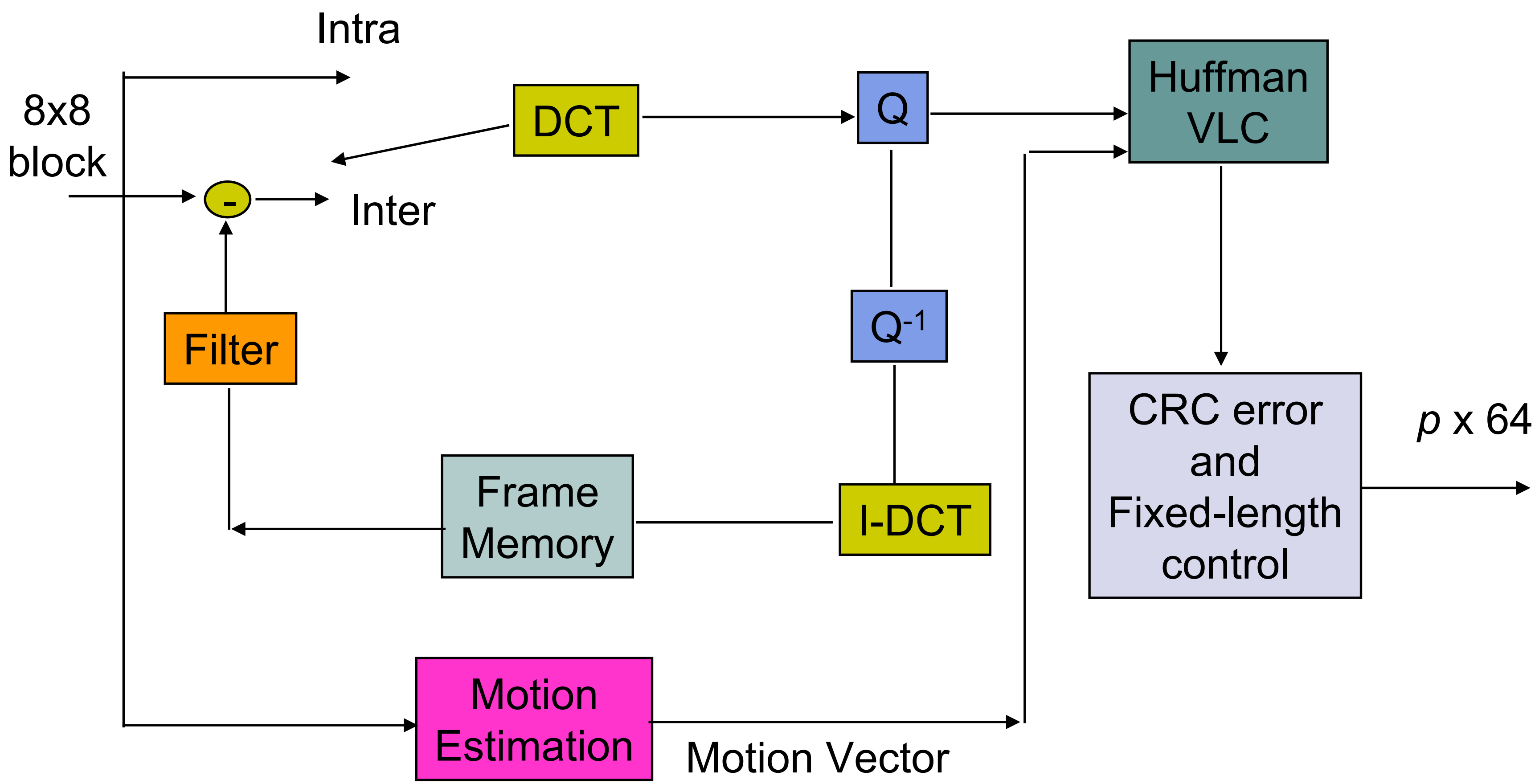

8x8
block
DCT
Quantization
VLC coding
Symmetric
DCT/I-DCT
transfer
Quantization
Matrices
Huffman Tab
DC: DPCM VLI
AC: O-run/VLI

Info = M bits/pixel x (WxH) pixels/frame x F frames/second

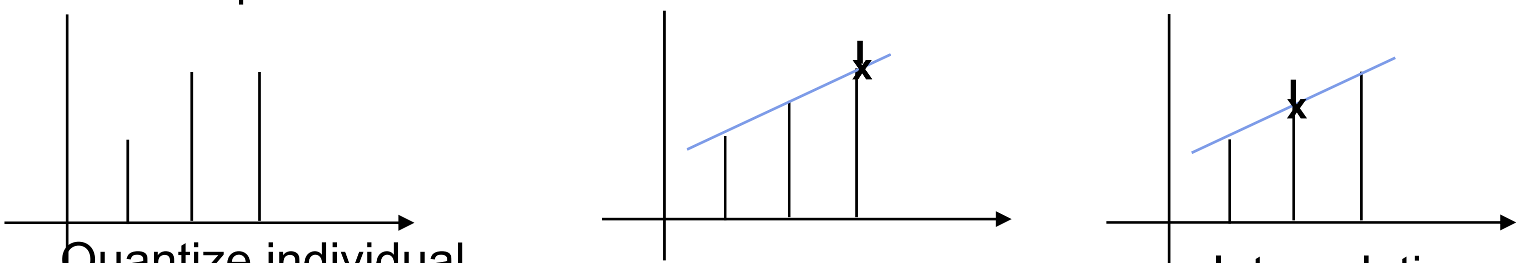




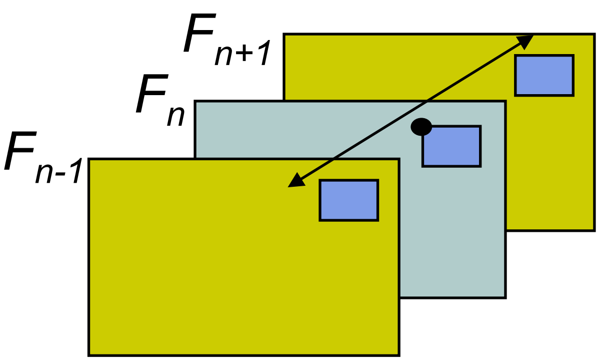
F_n
F_{n-1}
Hybrid Encoder:
(+1,+2)
(x,y)
Frame buffer
F_{n-1}
Motion Vector
Intra- /inter-frame processor
- Motion Vector
- Error Block
F_n
Encoder
- Intra Block

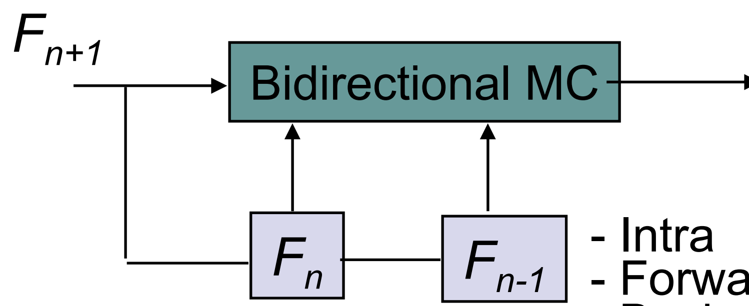

Intra
8x8
block
-
Inter
DCT
Q
Huffman
VLC
Filter
Q-1
CRC error
and
Fixed-length
control
p x 64
Frame
Memory
I-DCT
Motion
Estimation
Motion Vector

1-D examples:



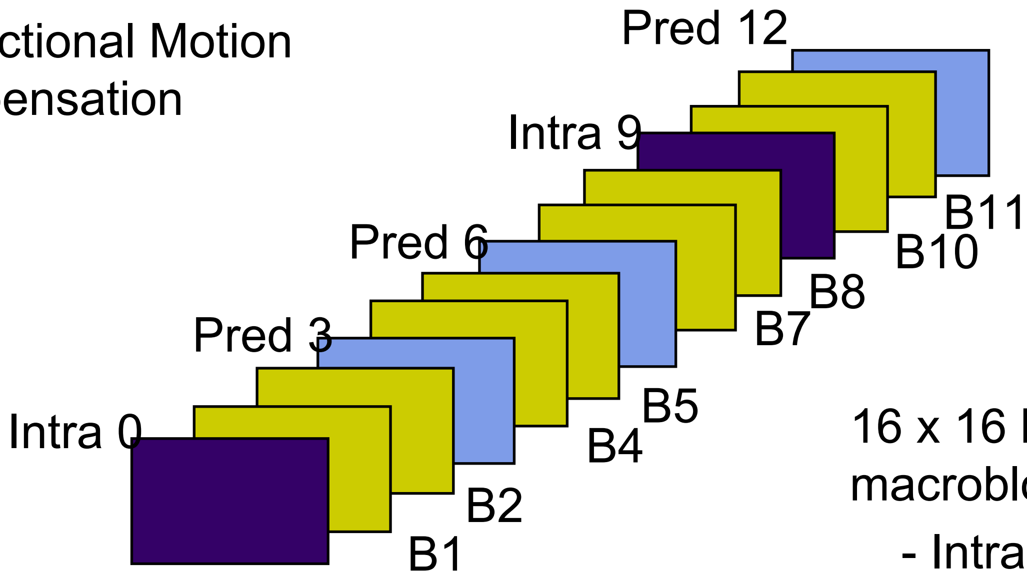
Quantize individual samples | Linear prediction | Interpolation





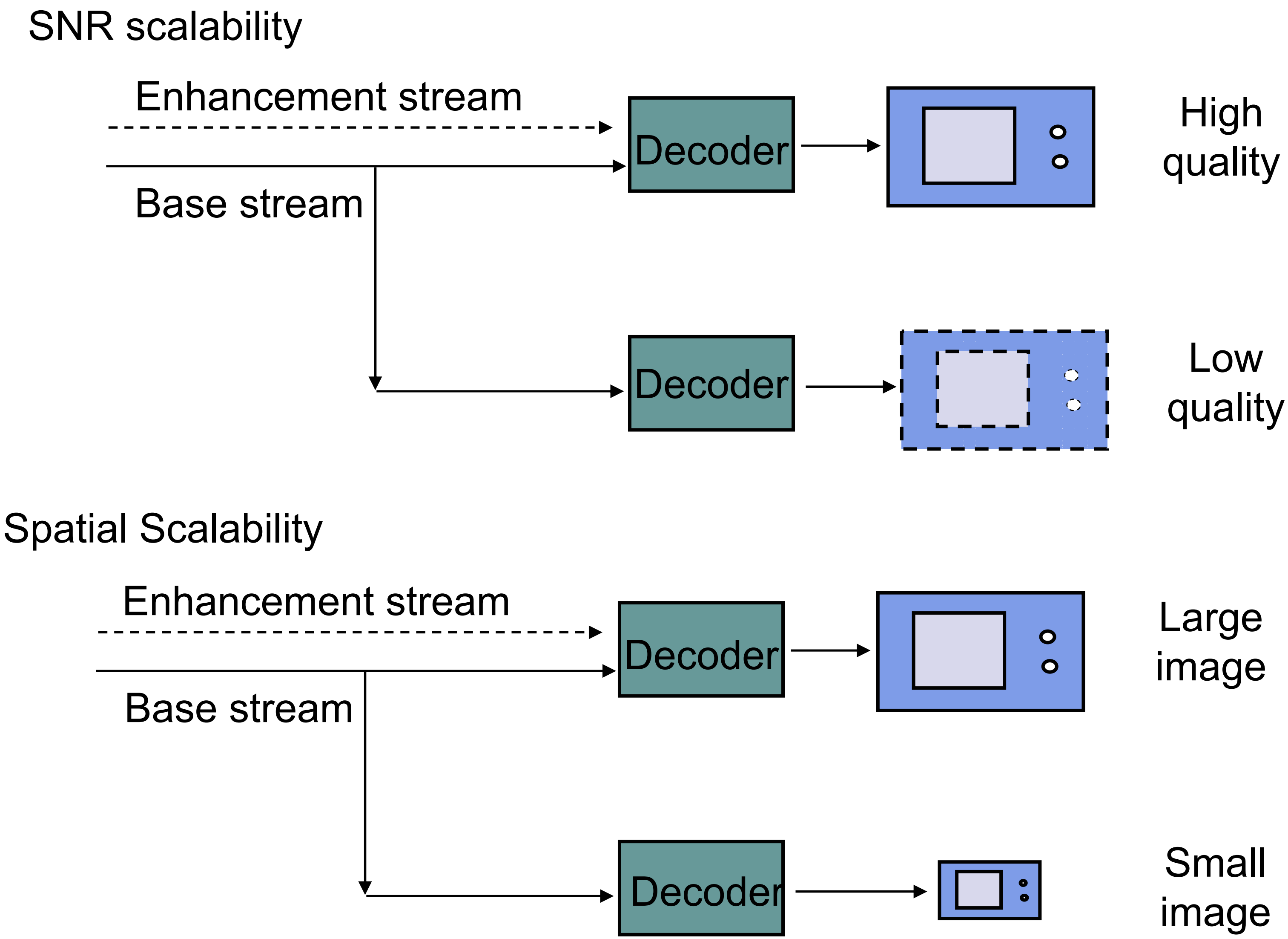
- Intra
- Forward
- Backward
- Bidirectional

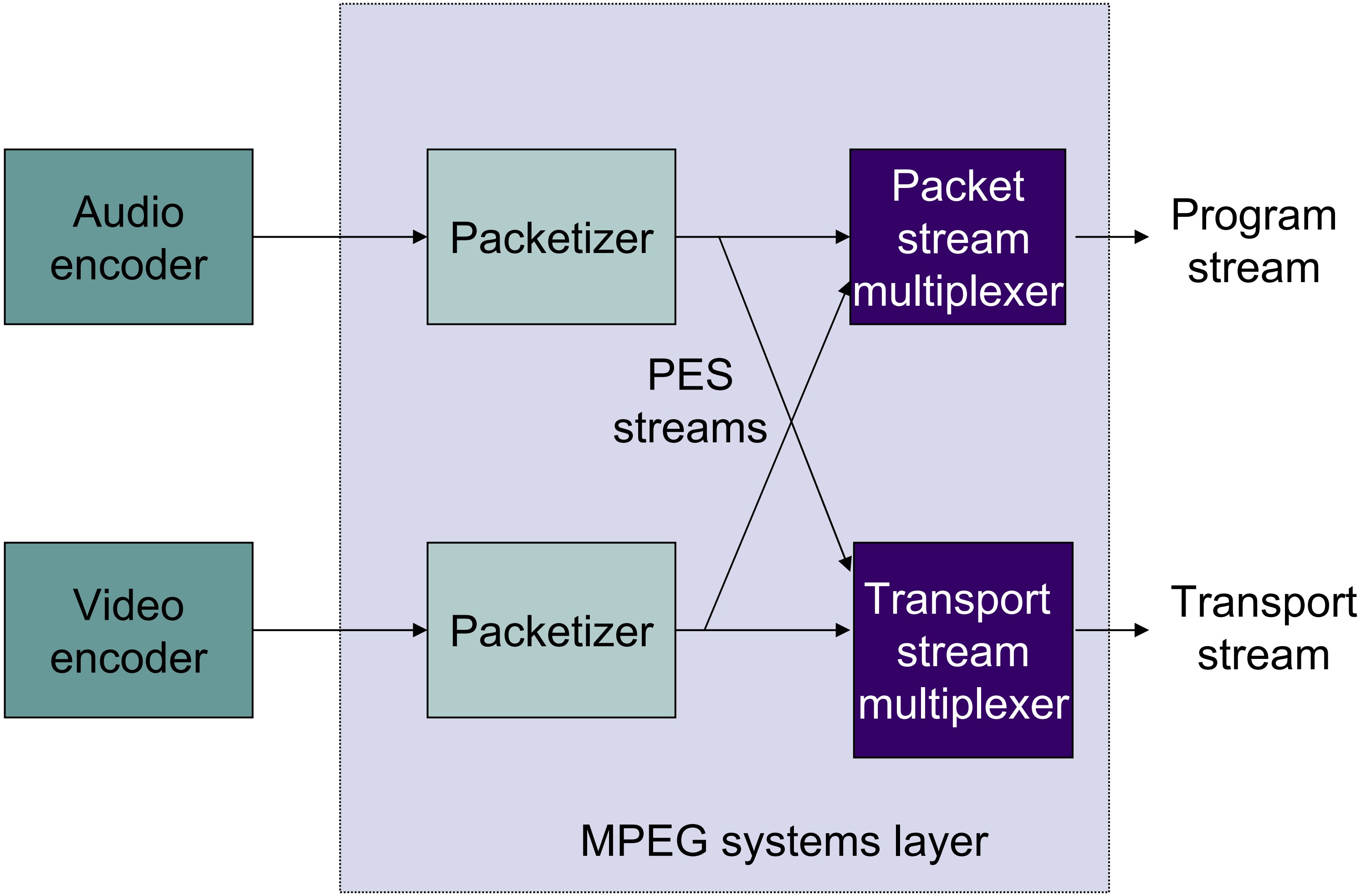
Bidirectional Motion Compensation



16 x 16 bidirectional macroblocks

- Intra
- Forward
- Reverse
- Bidirectional


SNR scalability
Enhancement stream
Base stream
Decoder
Decoder
High quality
Low quality
Spatial Scalability
Enhancement stream
Base stream
Decoder
Decoder
Large image
Small image


Audio
encoder
Packetizer
Packet
stream
multiplexer
Program
stream
PES
streams
Video
encoder
Packetizer
Transport
stream
multiplexer
Transport
stream
MPEG systems layer