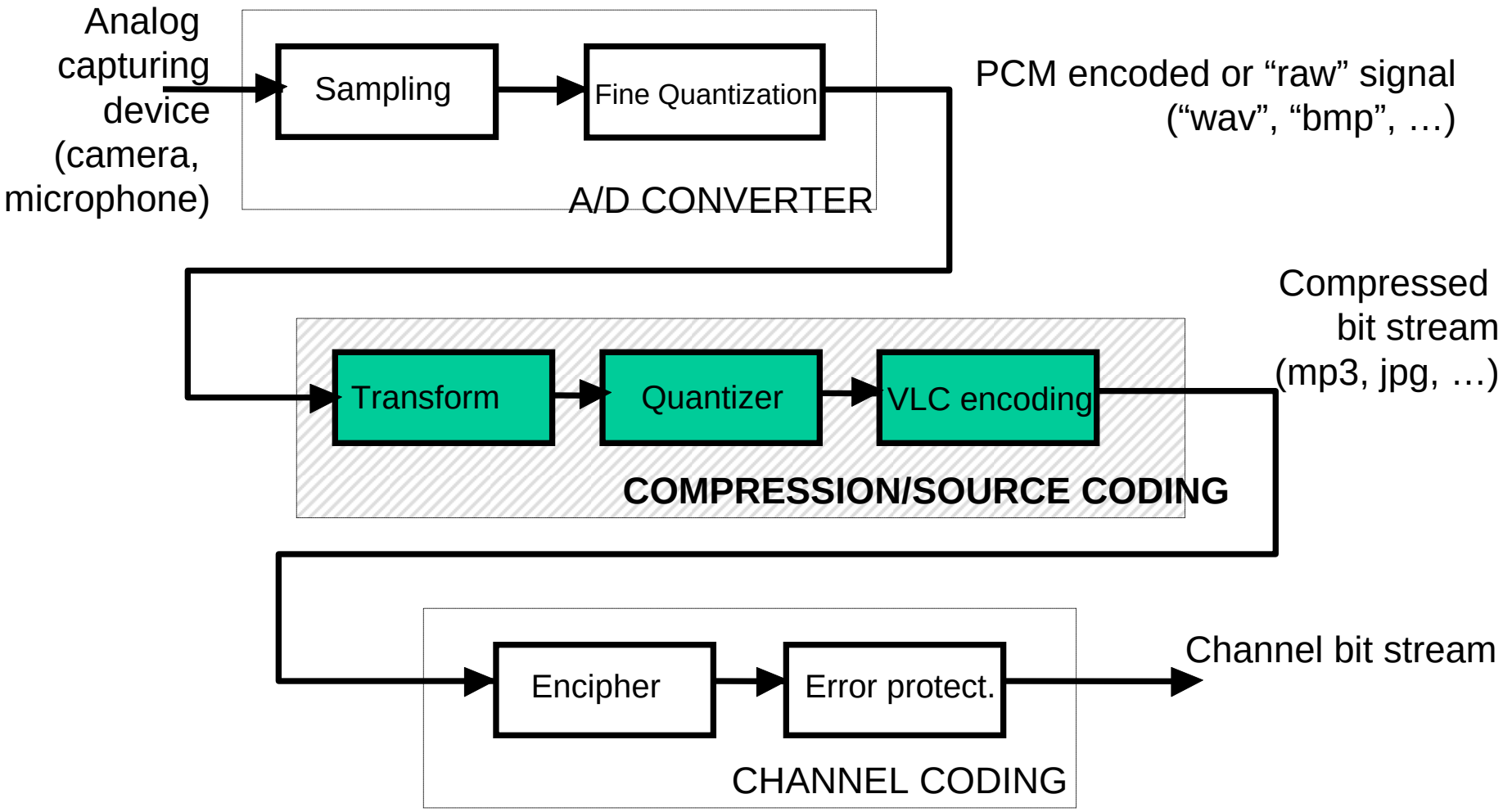
Overview


Analog capturing device (camera, microphone)
Sampling
Fine Quantization
A/D CONVERTER
PCM encoded or “raw” signal (“wav”, “bmp”, …)
Transform
Quantizer
VLC encoding
COMPRESSION/SOURCE CODING
Compressed bit stream (mp3, jpg, …)
Encipher
Error protect.
CHANNEL CODING
Channel bit stream

Reduce the #Amplitudes

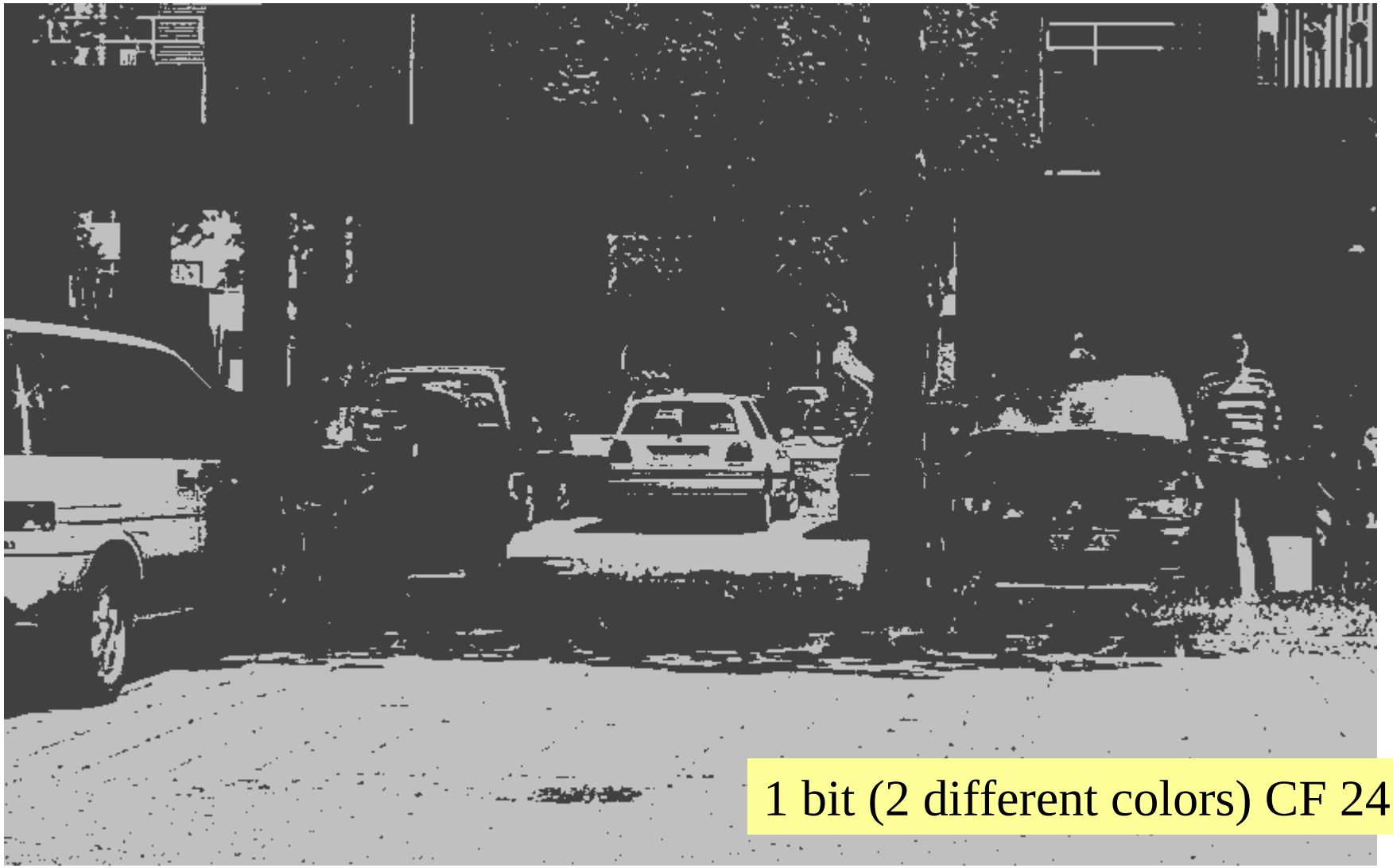
Reduce the #Amplitudes

Reduce the #Amplitudes

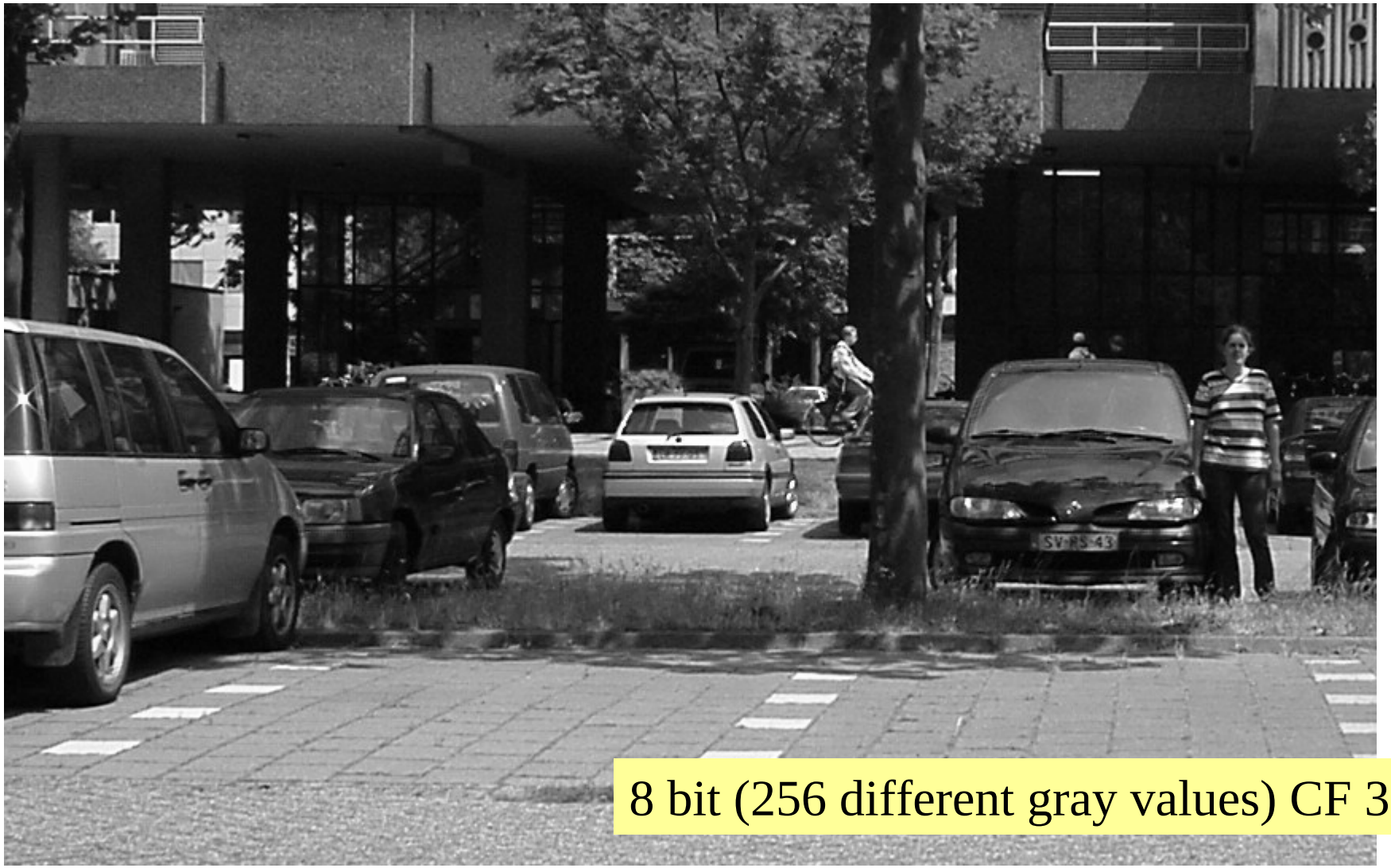
Reduce the #Amplitudes

4 bit (16 different colors) CF 6

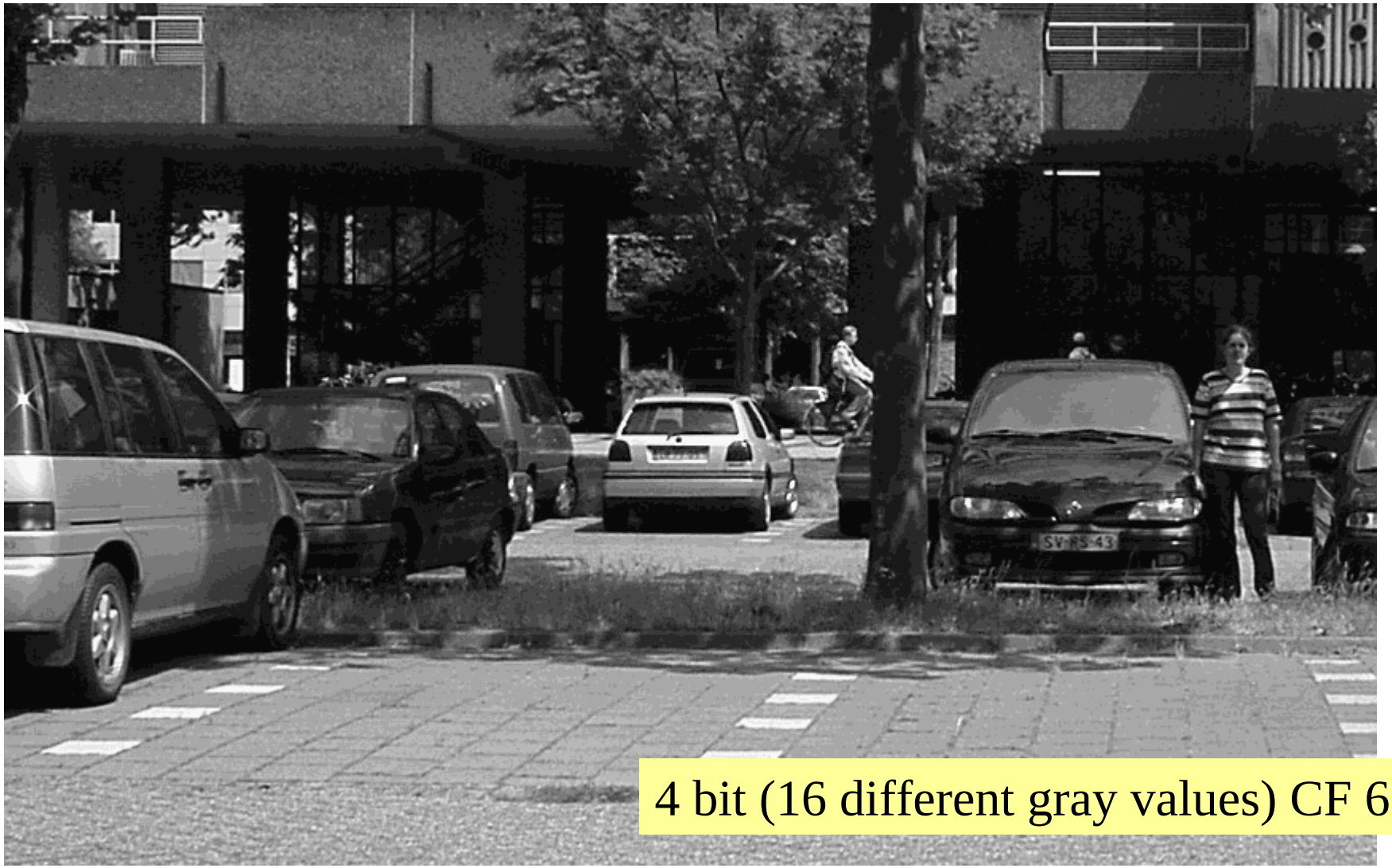
Reduce the #Amplitudes



Reduce the #Amplitudes



Reduce the #Amplitudes



Reduce the #Amplitudes

Reduce the #Amplitudes

But there are More Cleaver Ways

Why can Signals be Compressed?

- Because infinite accuracy of signal amplitudes is (perceptually) irrelevant

24 bit (16777200 different colors)

8 bit (256 different colors)
Compression factor 3

Rate-distortion theory, scalar/vector quantization

Why can Signals be Compressed?

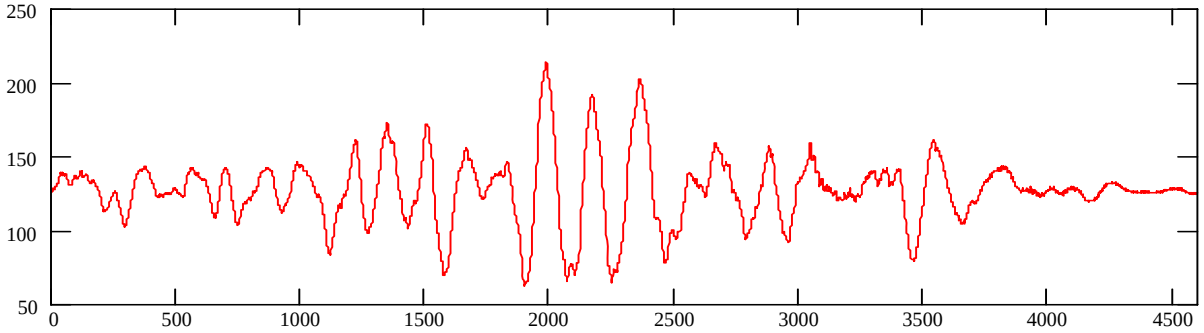
- Because signal amplitudes are statistically redundant

s_i	$P_S(s_i)$	$I(s_i)$
0	0.125	3 bit
1	0	
2	0.5	1 bit
3	0	
4	0.125	3 bit
5	0.125	3 bit
6	0	
7	0.125	3bit

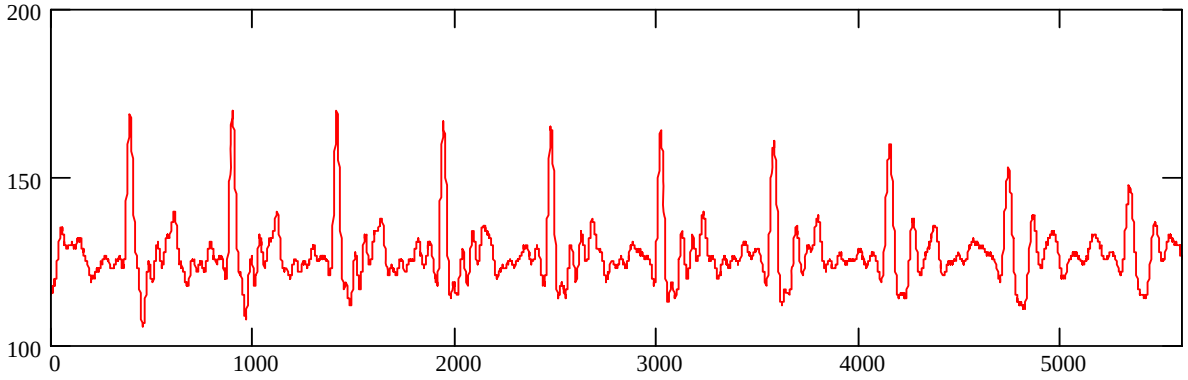
- Information theory, Huffman coding

Why can Signals be Compressed?

- Because signal amplitudes are mutually dependent



"S"

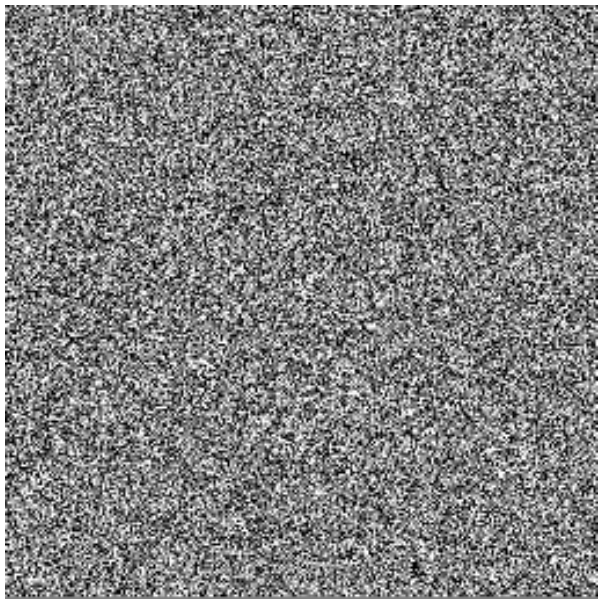


"O"

Rate-distortion theory, transform coding

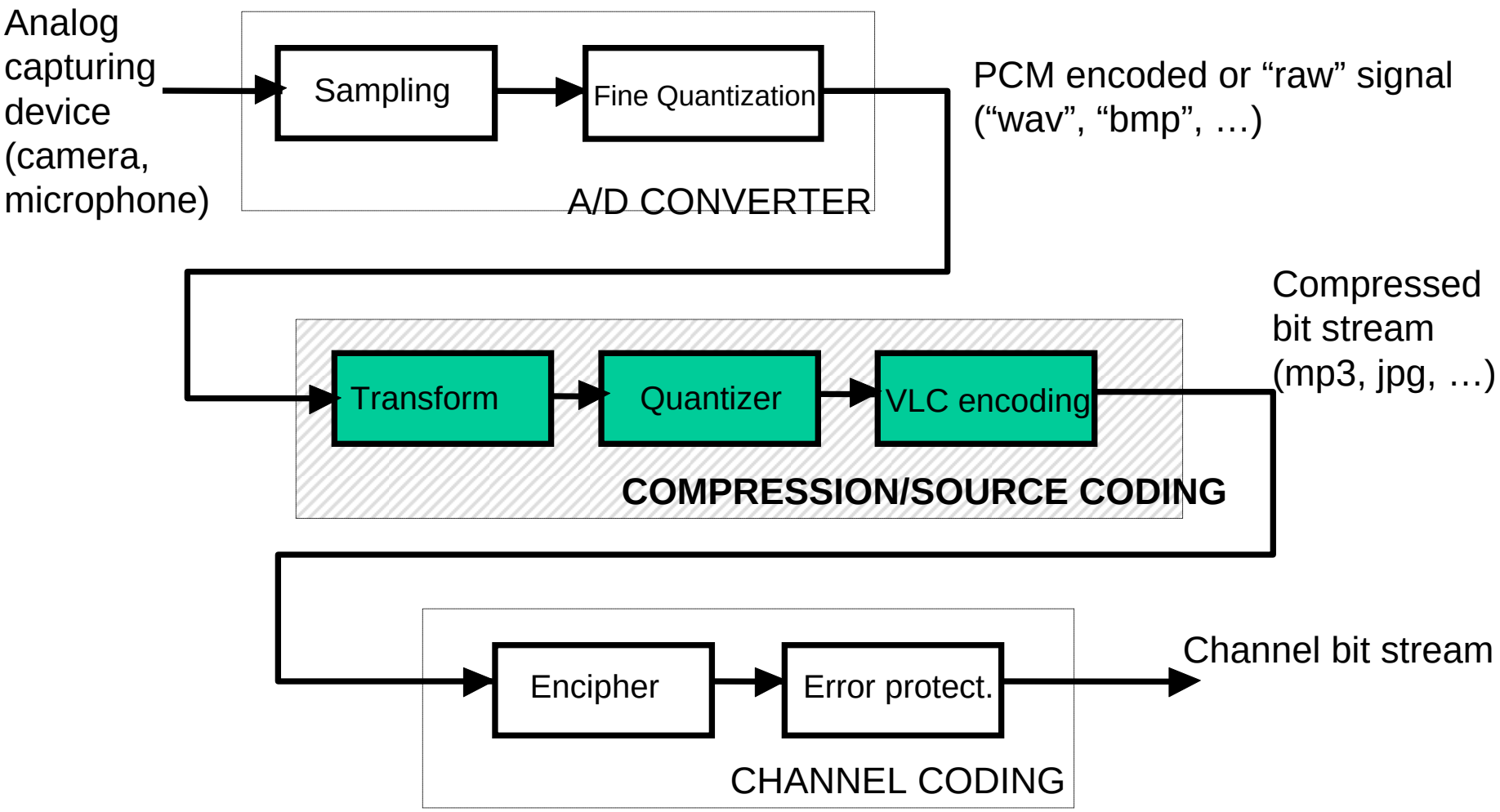
Why can Signals be Compressed?

- Example of signal with no dependencies between successive amplitudes (Gaussian uncorrelated noise)



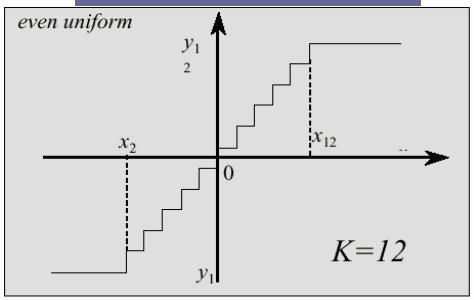
Indeed "noise" compresses badly

System Overview

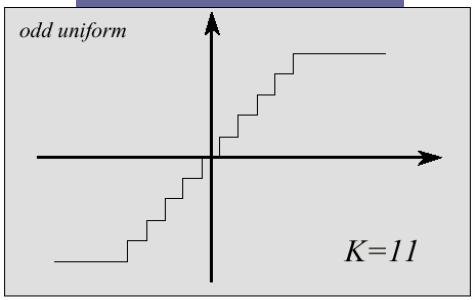


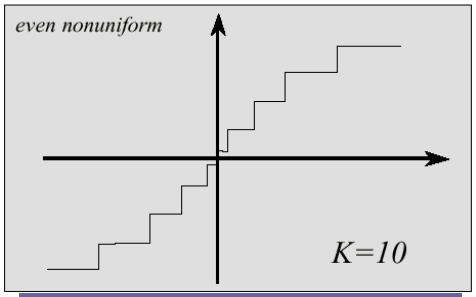
Quantization

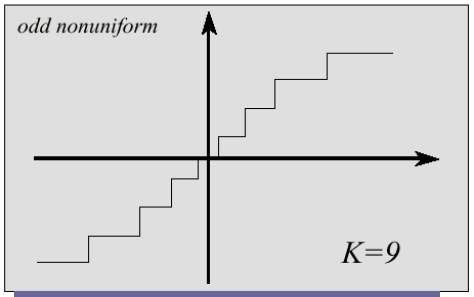
Even Uniform



Odd Uniform

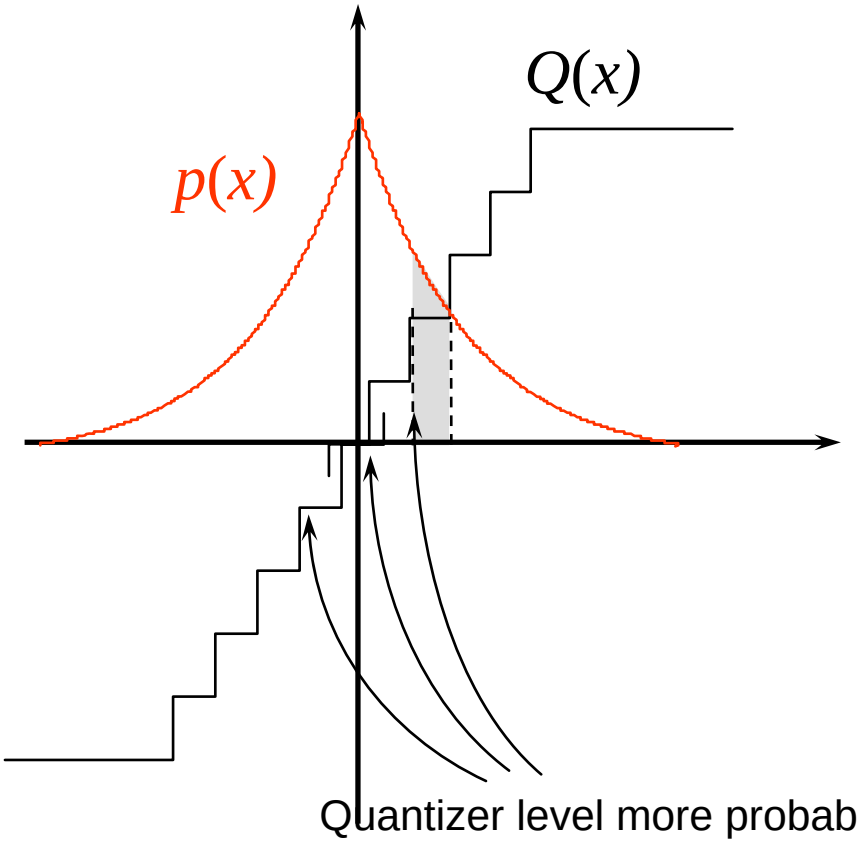






Even Non-uniform

Odd Non-uniform



Quantizer level more probable

$$\sigma_q^2 = \int_{-\infty}^{\infty} \underbrace{(x - Q[x])^2}_{\text{Amount of error}} \underbrace{p_X(x)\,\mathrm{d}\,x}_{\text{Probability of that amount}}$$

Information and Entropy

$I(s_i) = -\log_2[P_S(s_i)]$ (bit of information)

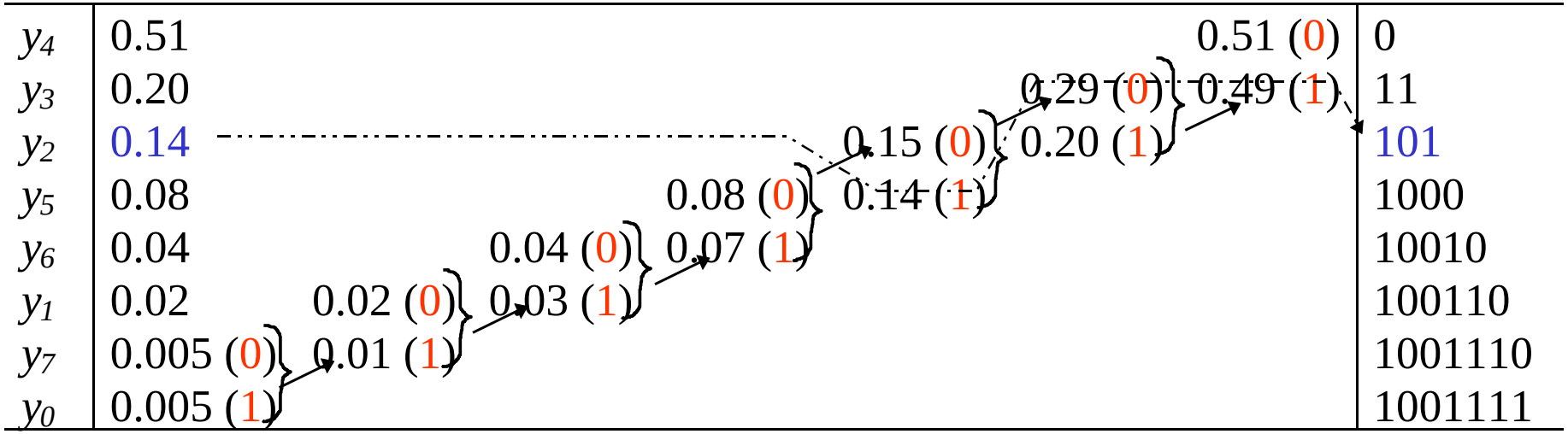
$$H(S) = -\sum_{i=1}^{N} P_S(s_i)\log_2[P_S(s_i)] \quad \text{(bit/symbol)}$$

More Probable
Less information

s_i	$P_S(s_i)$	$I(s_i)$
0	0.125	3 bit
1	0	
2	0.5	1 bit
3	0	
4	0.125	3 bit
5	0.125	3 bit
6	0	
7	0.125	3bit

$H(S)$ = 2 bit of information per amplitude

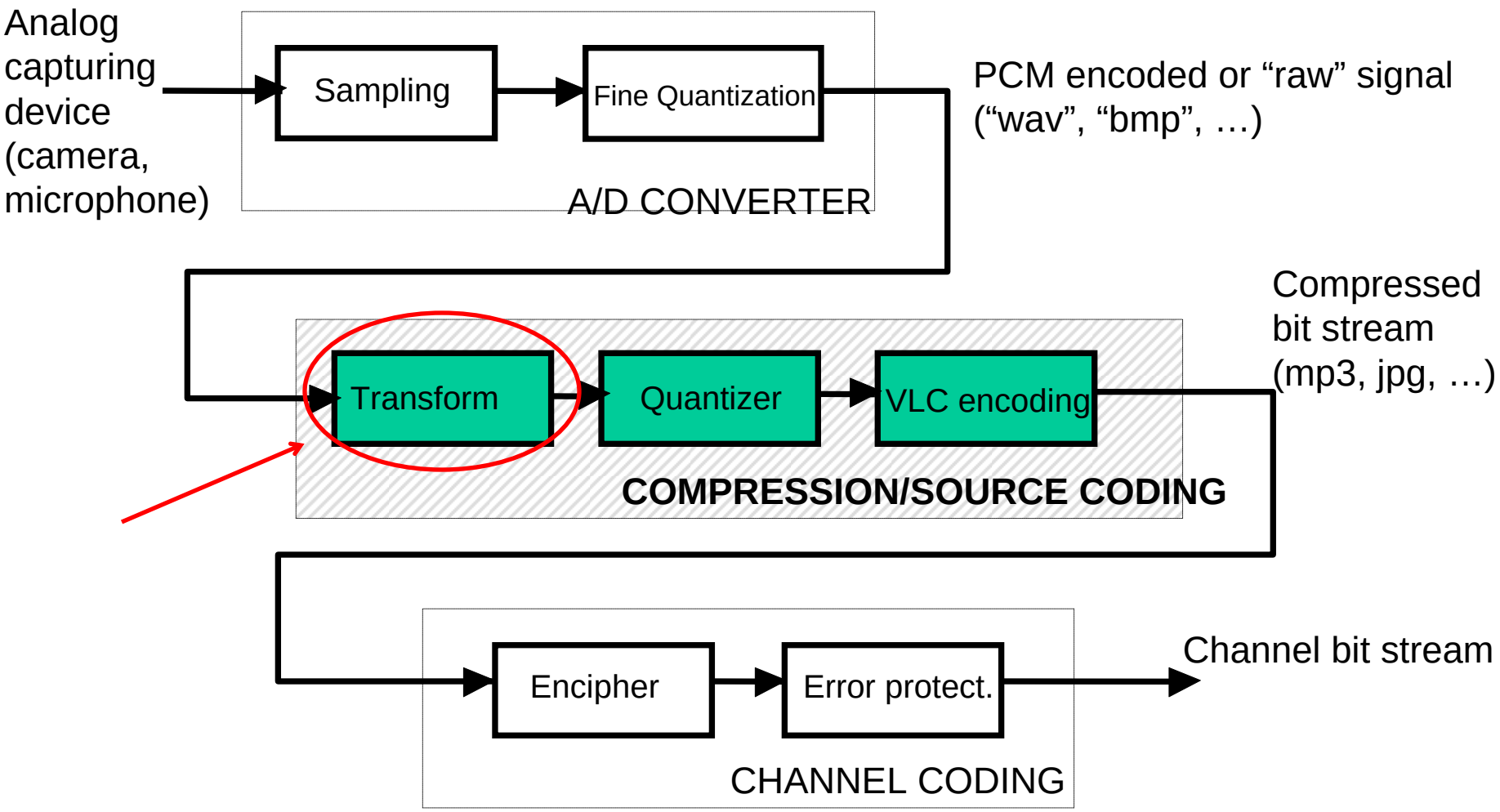
Huffman Code



Run Length Coding

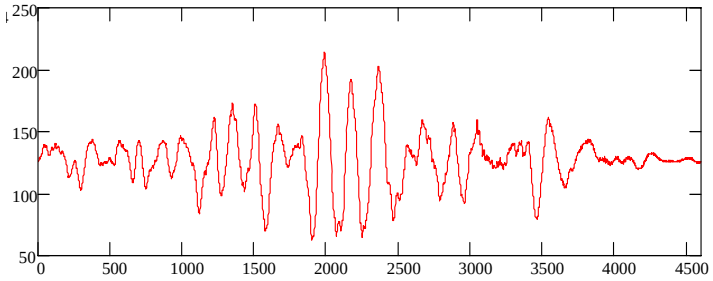
- Representing “0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 ...”
- “Runs” : 8 (“zeros”) 2 (“ones”) 6 (“zeros”) 5 (“ones”) ……
- The run lengths are also encoded (e.g. with Huffman coding)
- Efficient transforms (like DCT) used in compression produce
  - A lot of “zero” values
  - And a “few” (significant) non-zero values
- Typical symbol sequences to be coded “5 1 0 0 0 0 0 0 0 3 0 0 6 0 0 0 0 1 0 0 0 0 …..”
  - will be done by {zero-run, non-zero symbol/0} pairs
  - Here: “{0,5},{0,1},{7,3},{2,6},{4,1},…..”
  - The pairs will now be assigned a Huffman code
  - This is used in JPEG

General Compression System

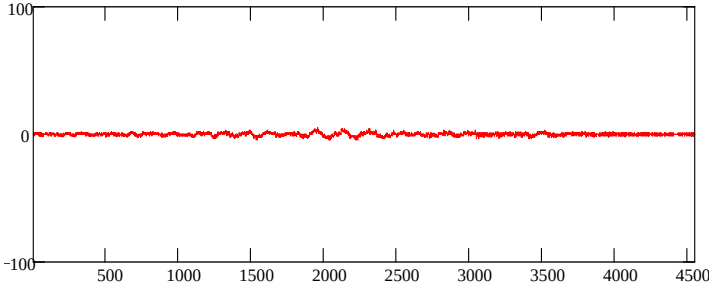


Correlation in Signals - I

- Meaningful signals are often highly predictable:



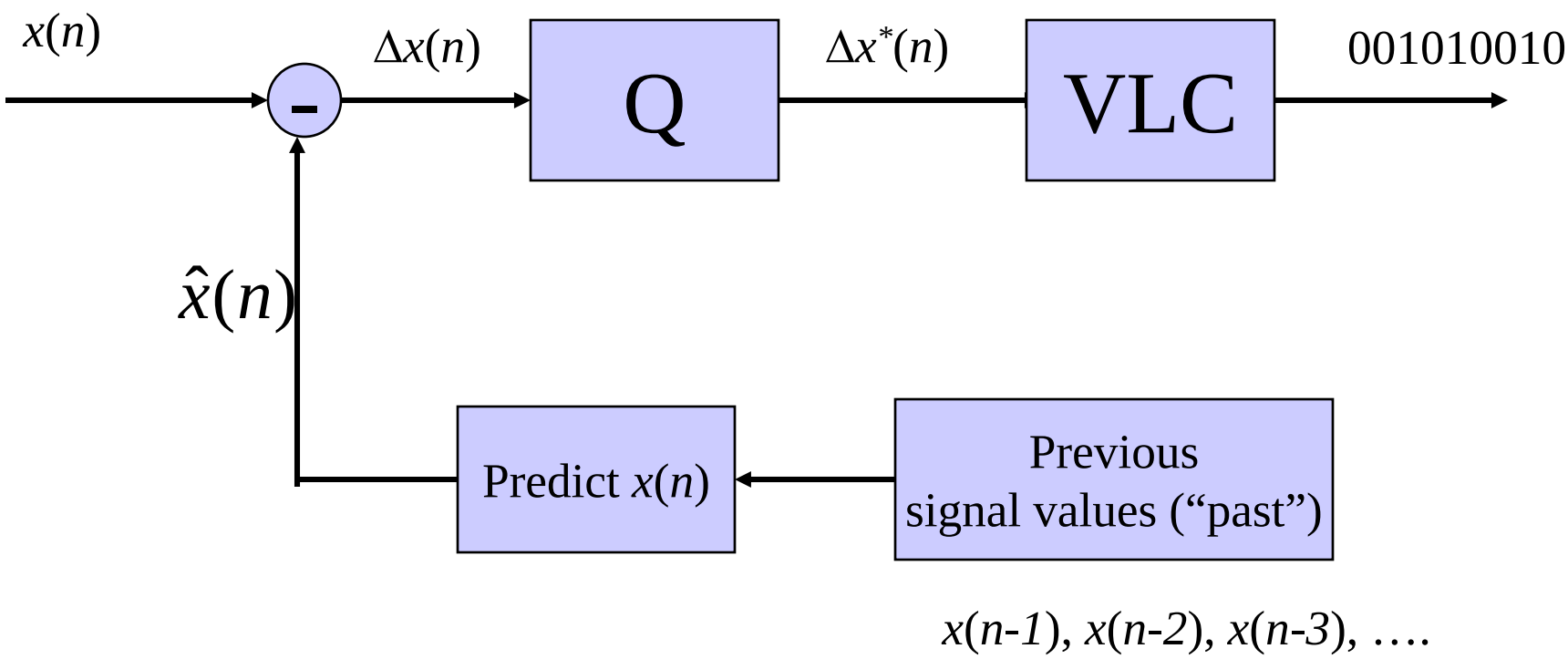
$\Delta x(n)=x(n)-x(n-1)$



- (Linear) Predictability has something to do with the autocorrelation function

n

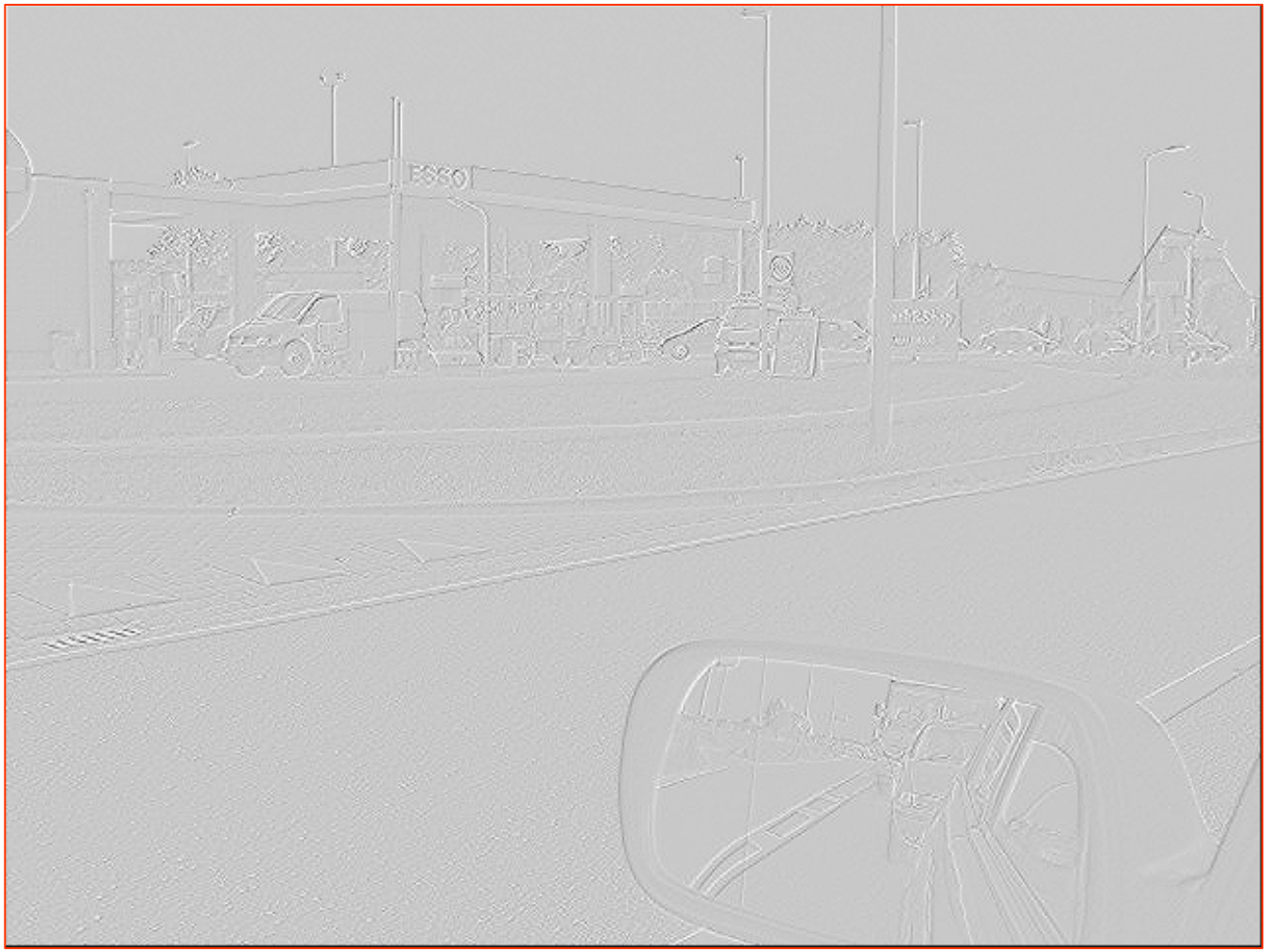
Principle of Differential PCM



Works Really Good - I

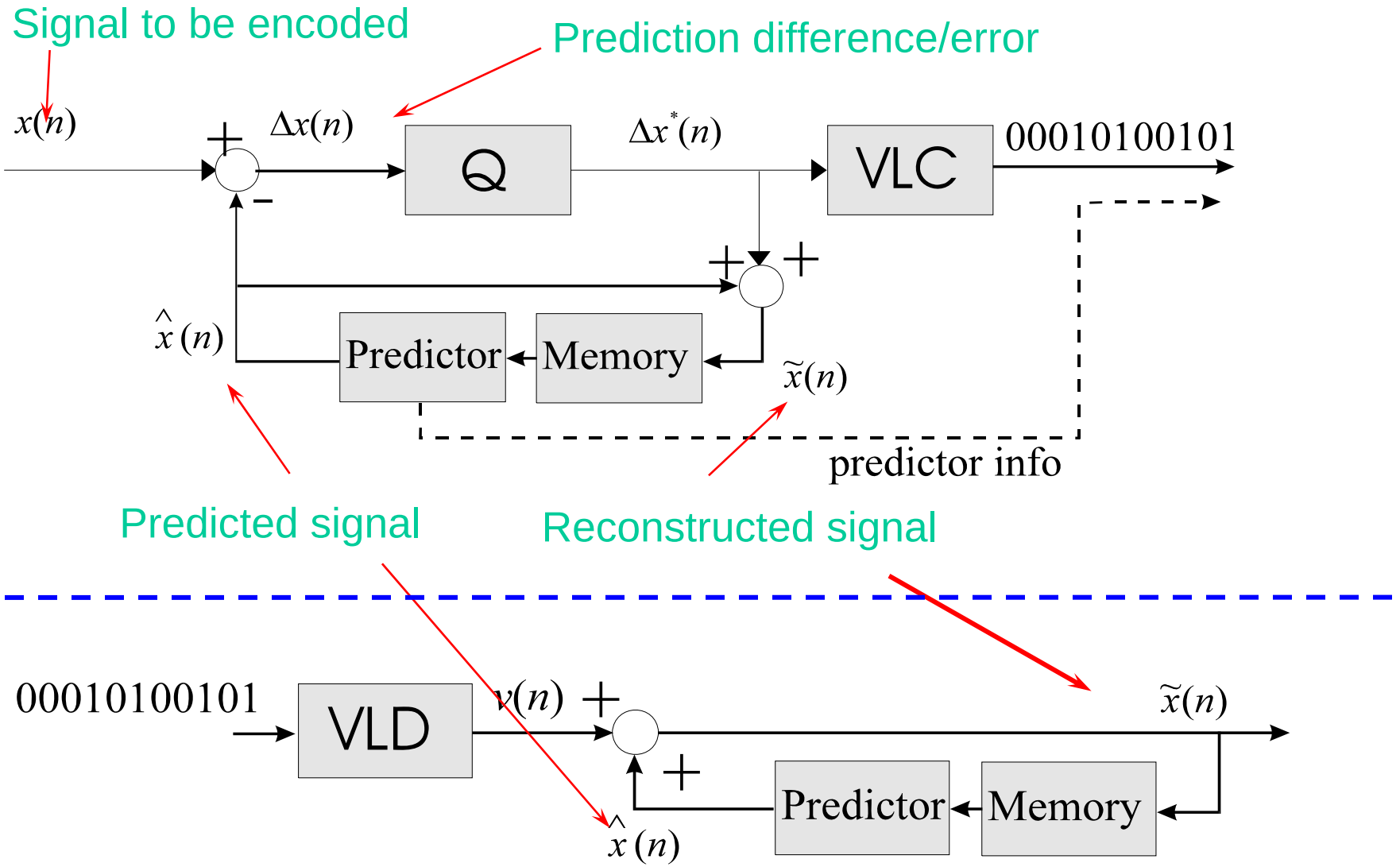
Variance = 3240

Works Really Good - II



Variance = 315

DPCM

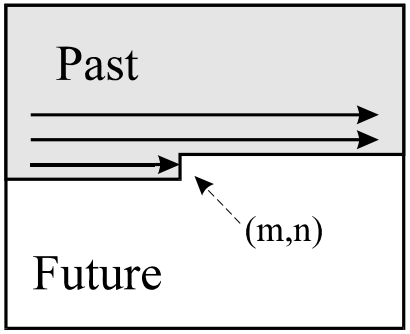

Signal to be encoded
Prediction difference/error
x(n)
Δx(n)
Q
Δx*(n)
VLC
00010100101
Predictor
Memory
predictor info
Predicted signal
Reconstructed signal
00010100101
VLD
v(n)
Predictor
Memory

What Linear Predictor to Use?

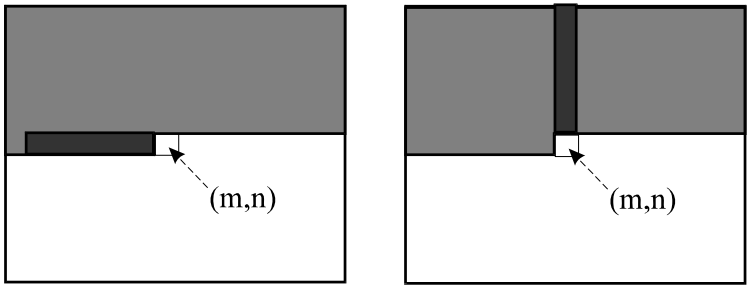
- Examples:
  - PCM $\hat{x}(n) = 0$
  - Simple differences $\hat{x}(n) = \tilde{x}(n-1)$
  - Average last two samples $\hat{x}(n) = h_1 \tilde{x}(n-1) + h_2 \tilde{x}(n-2)$
  - General linear predictor $\hat{x}(n) = \sum_{k=1}^{N} h_k \tilde{x}(n-k)$

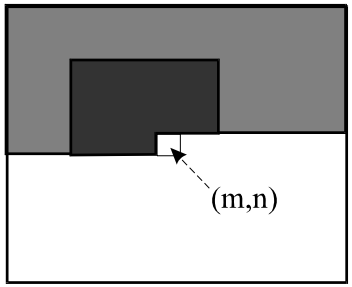
DPCM on Images

- Same principle as 1-D
- Definition of “Past” and “Future” in Images:

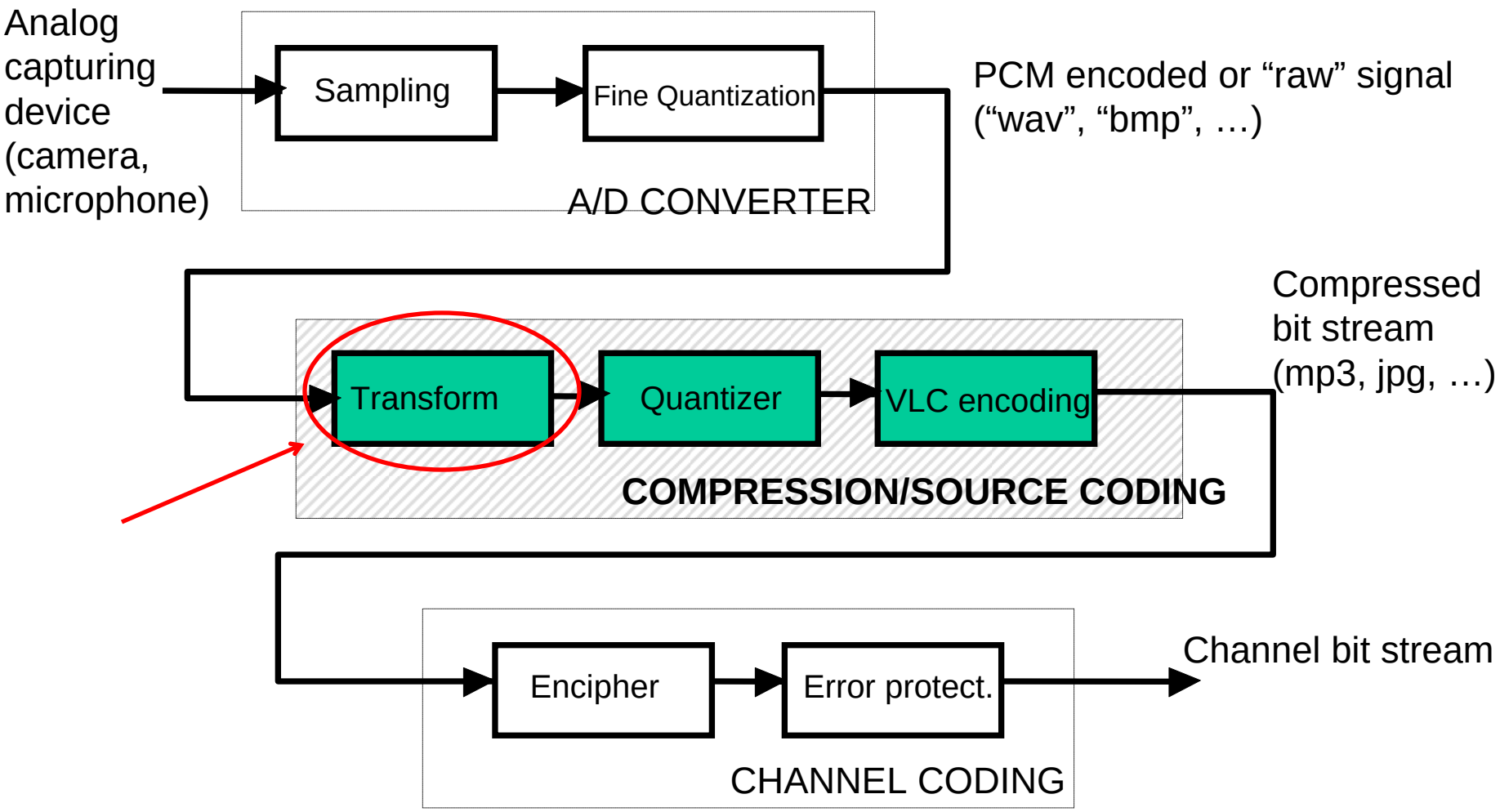


- Predictions:
  - horizontal (scan line)
  - vertical (column)
  - 2-dimensional

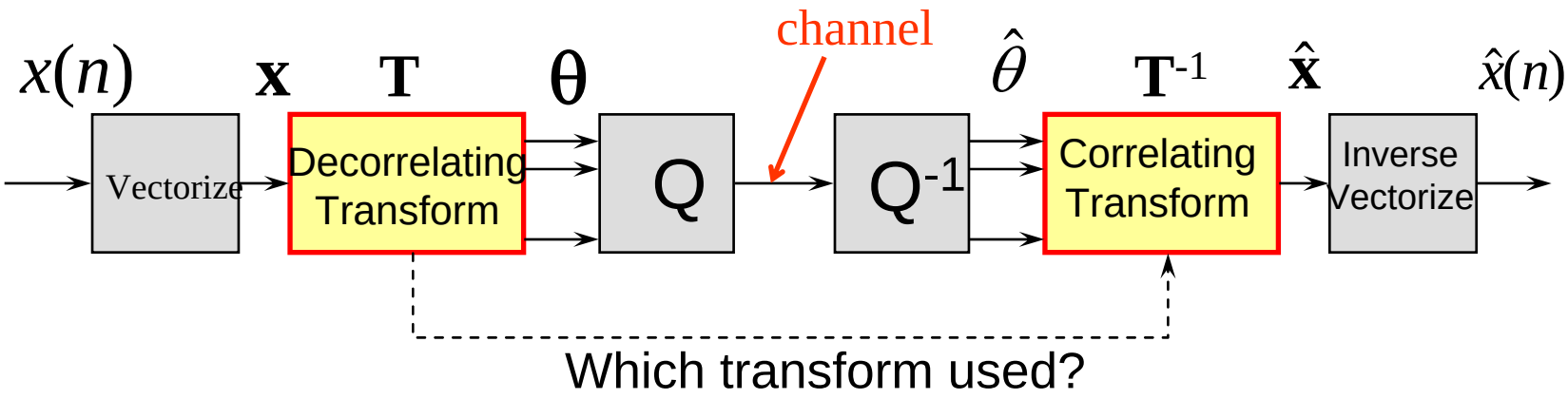




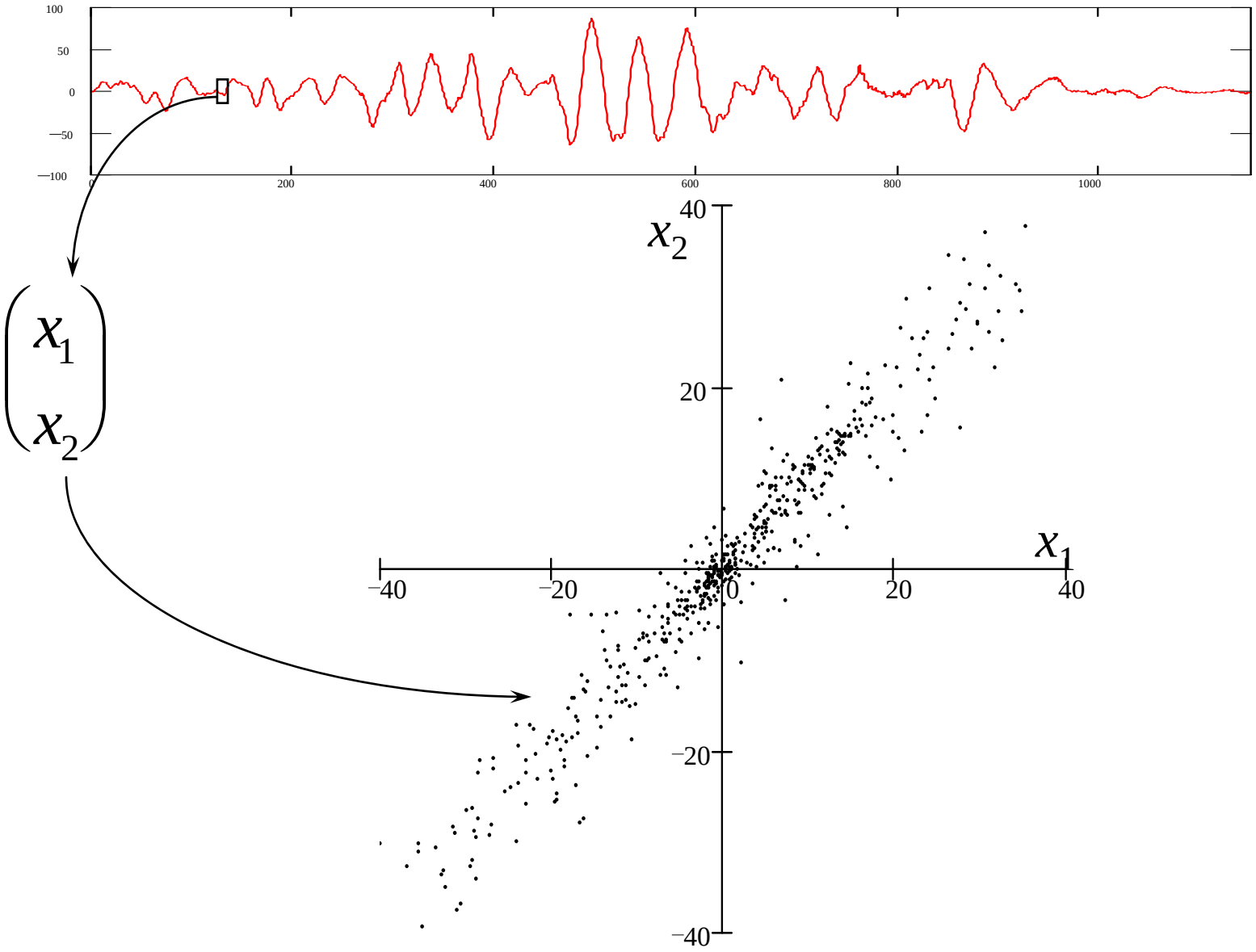
General Compression System



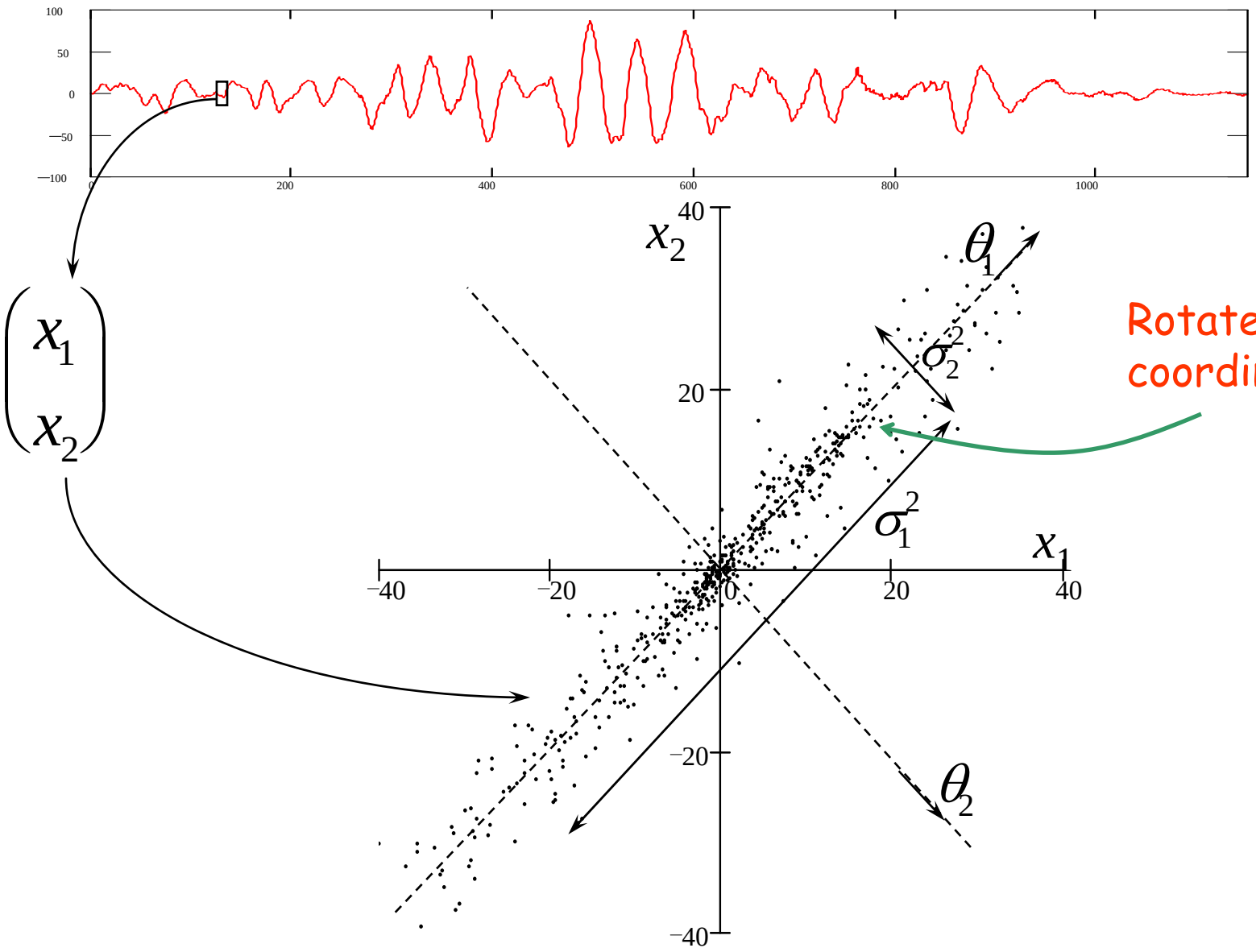
Transform Coding



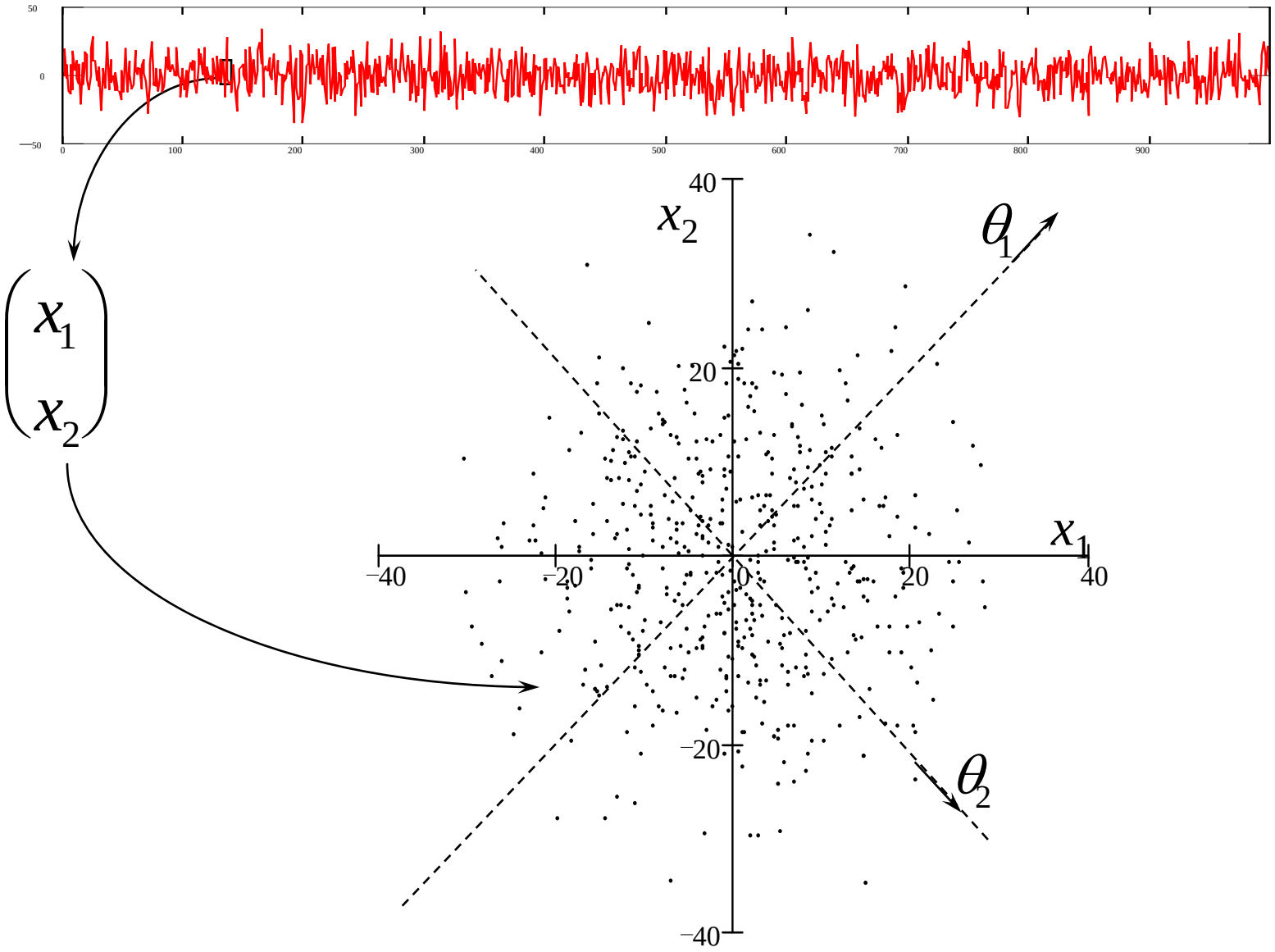
Removing Correlation - I



Removing Correlation - II



Removing Correlation - III



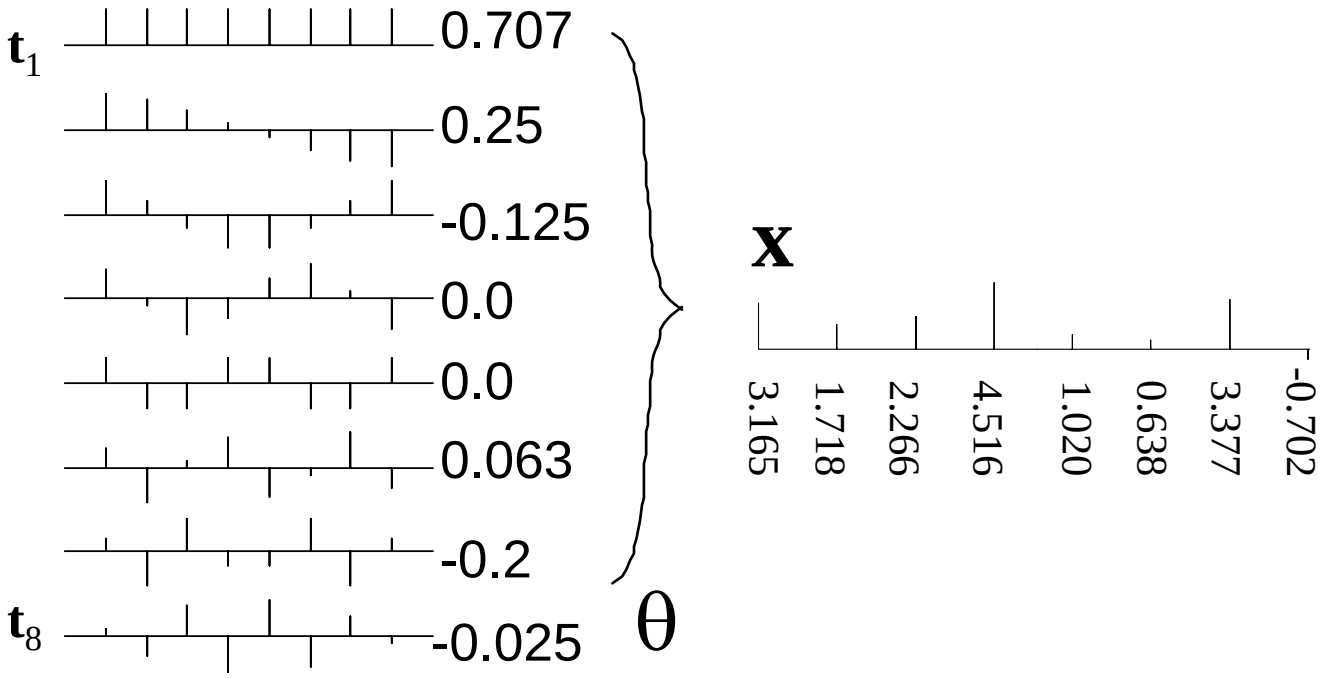
Decomposition

$$\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_N \end{pmatrix} = \begin{pmatrix} t_{1,1} & \cdots & t_{1,N} \\ \vdots & \ddots & \vdots \\ t_{N,1} & \cdots & t_{N,N} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \mathbf{Tx}$$

$$\theta_k = \sum_{n=1}^{N} t_{k,n} x_n \qquad k = 1,2,\ldots,N$$

$$x_n = \sum_{k=1}^{N} \theta_k t_{n,k} \qquad n = 1,2,\ldots,N$$

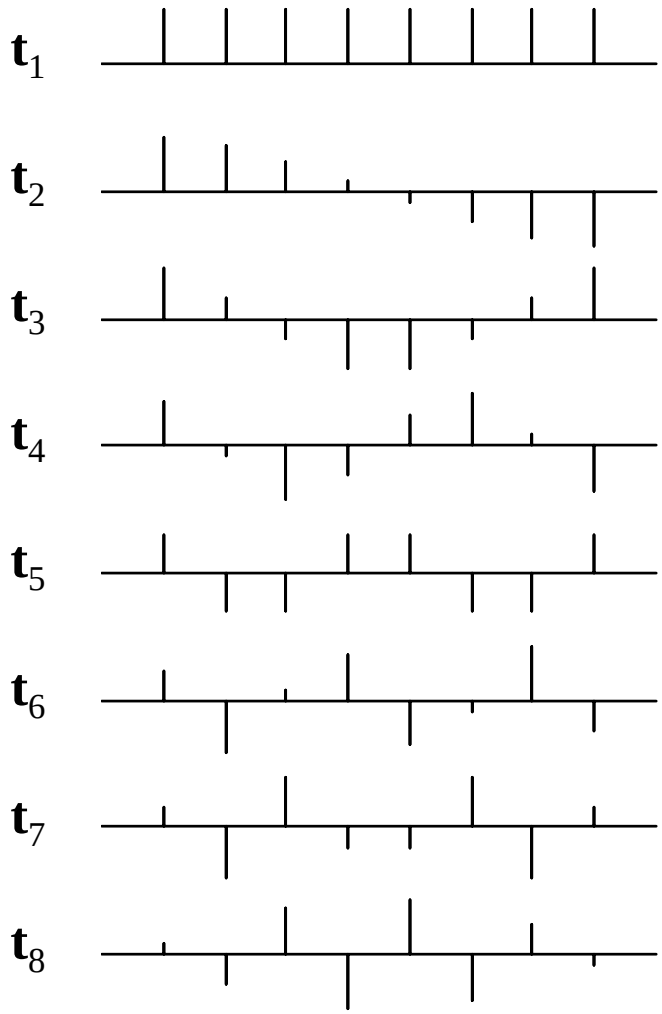
$$\mathbf{x} = \sum_{k=1}^{N} \theta_k \mathbf{t}_k = \theta_1 \mathbf{t}_1 + \theta_2 \mathbf{t}_2 + \cdots + \theta_N \mathbf{t}_N$$



(Example)

Discrete Cosine Transform

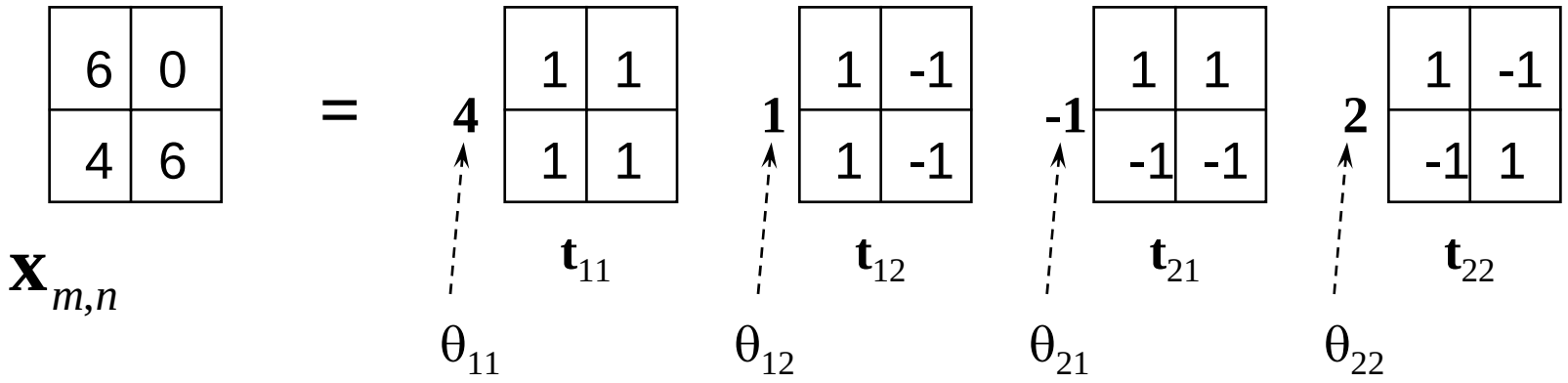
$N=8$



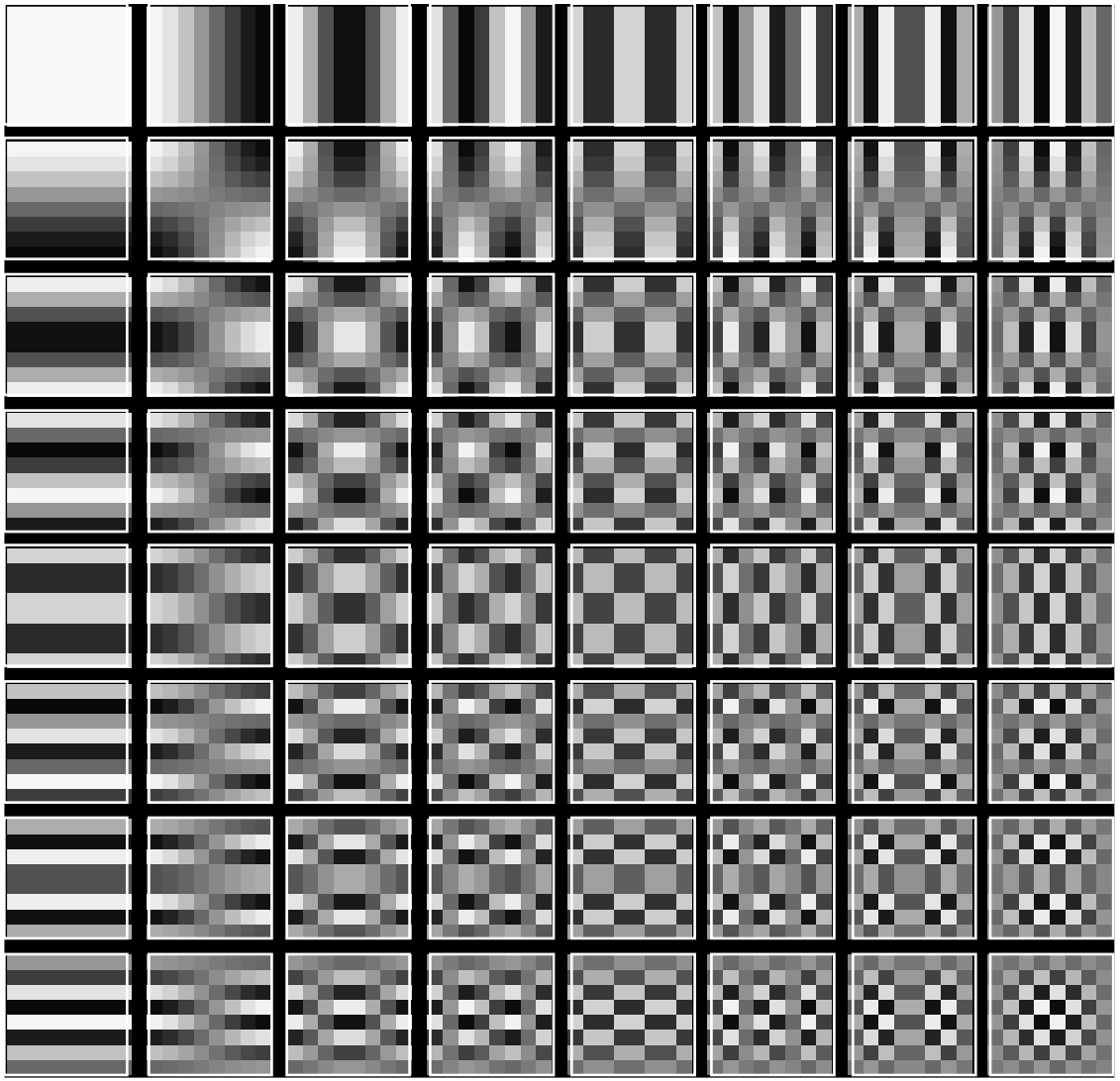
$$\mathbf{T}^t = \begin{bmatrix}
0.354 & 0.49 & 0.462 & 0.416 & 0.354 & 0.278 & 0.191 & 0.098 \\
0.354 & 0.416 & 0.191 & -0.098 & -0.354 & -0.49 & -0.462 & -0.278 \\
0.354 & 0.278 & -0.191 & -0.49 & -0.354 & 0.098 & 0.462 & 0.416 \\
0.354 & 0.098 & -0.462 & -0.278 & 0.354 & 0.416 & -0.191 & -0.49 \\
0.354 & -0.098 & -0.462 & 0.278 & 0.354 & -0.416 & -0.191 & 0.49 \\
0.354 & -0.278 & -0.191 & 0.49 & -0.354 & -0.098 & 0.462 & -0.416 \\
0.354 & -0.416 & 0.191 & 0.098 & -0.354 & 0.49 & -0.462 & 0.278 \\
0.354 & -0.49 & 0.462 & -0.416 & 0.354 & -0.278 & 0.191 & -0.098
\end{bmatrix}$$

(All picture compression standards implement this matrix in a clever way)

Image Transforms - I



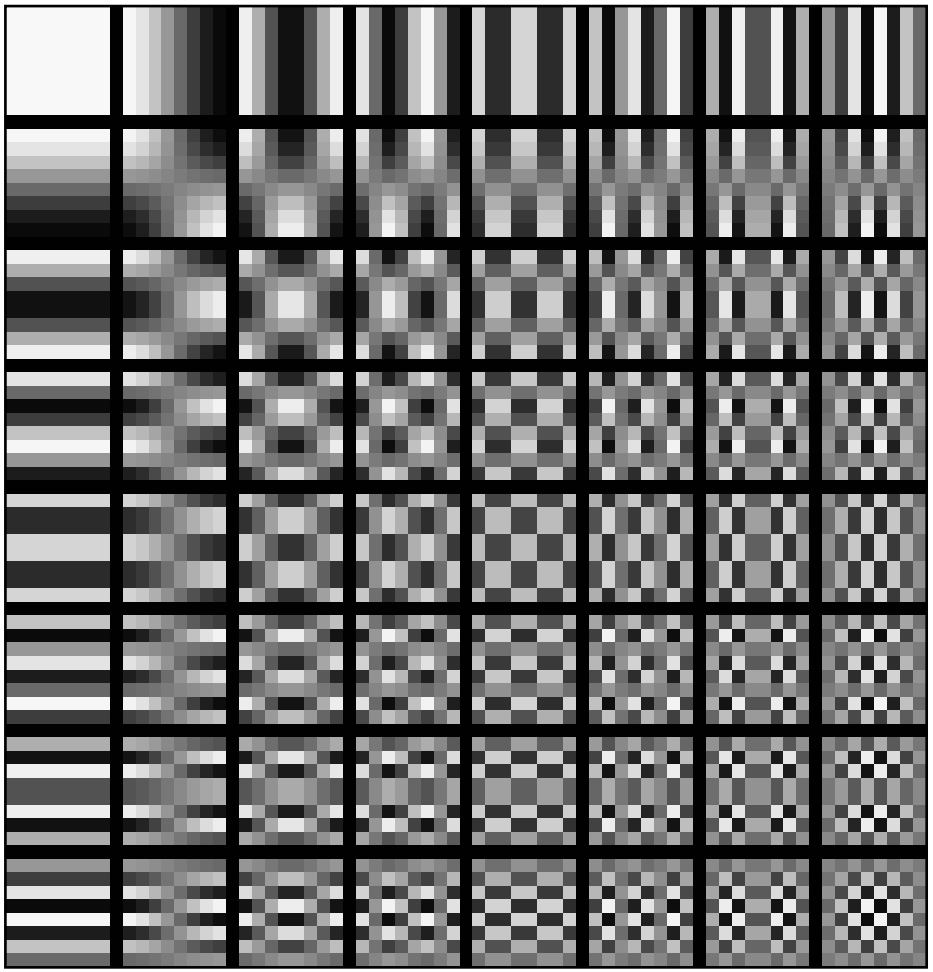
Basis Images for 2-D DCT



Discrete Cosine Transform (DCT) (1)

Discrete Cosine Transform (DCT) (2)

64 pixels



Cosine patterns/DCT basis functions

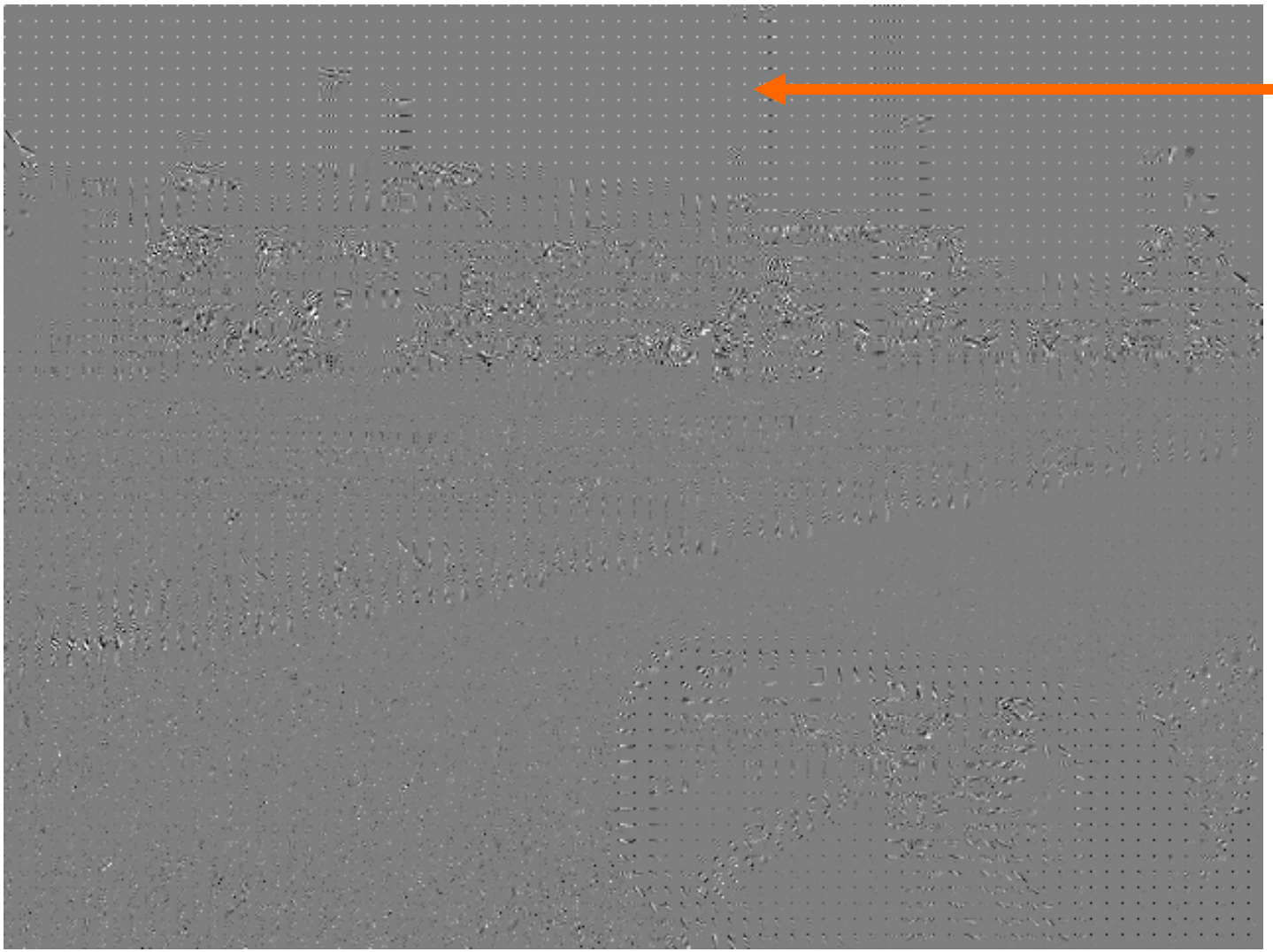
Discrete Cosine Transform (DCT) (3)

64 pixels

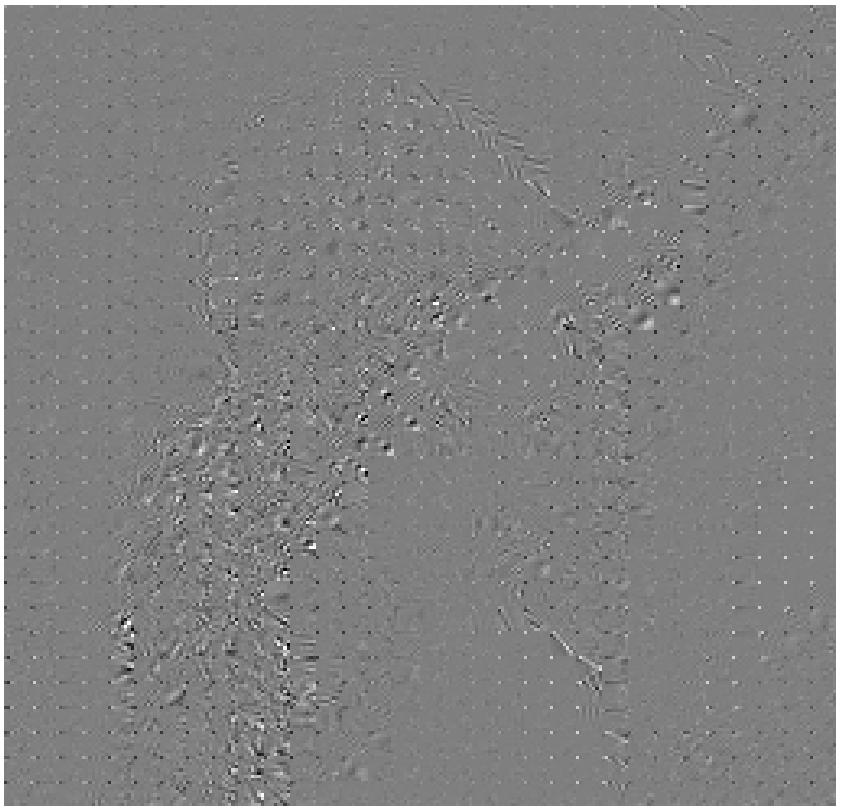
89	8	8	1	0	-1	1	0
2	3	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	0	0	1	0	0	0	0
2	0	0	1	0	1	0	0
0	0	-1	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

Cosine patterns/DCT basis functions

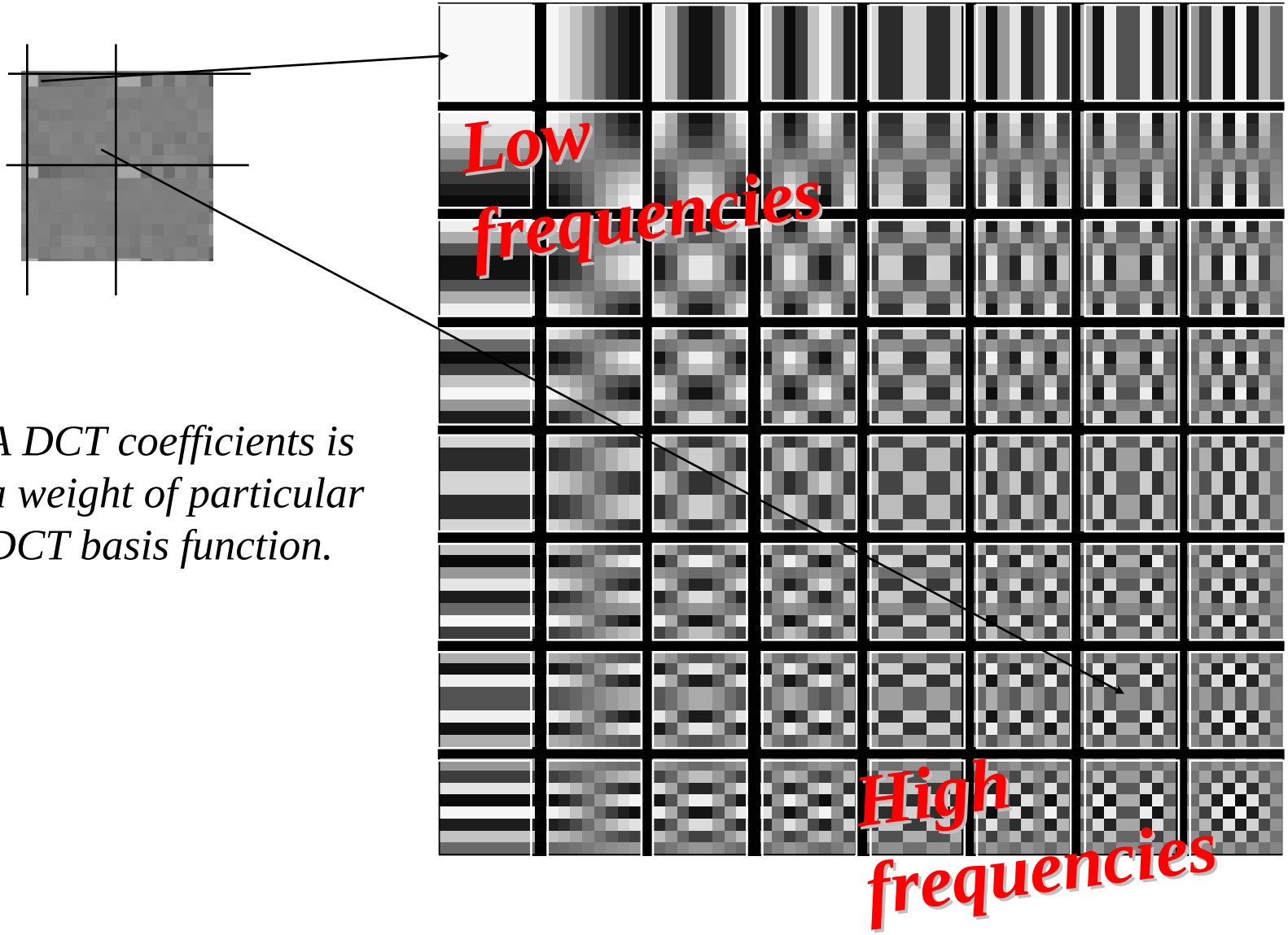
Discrete Cosine Transform (DCT) (4)



2-D DCT of Picture - I

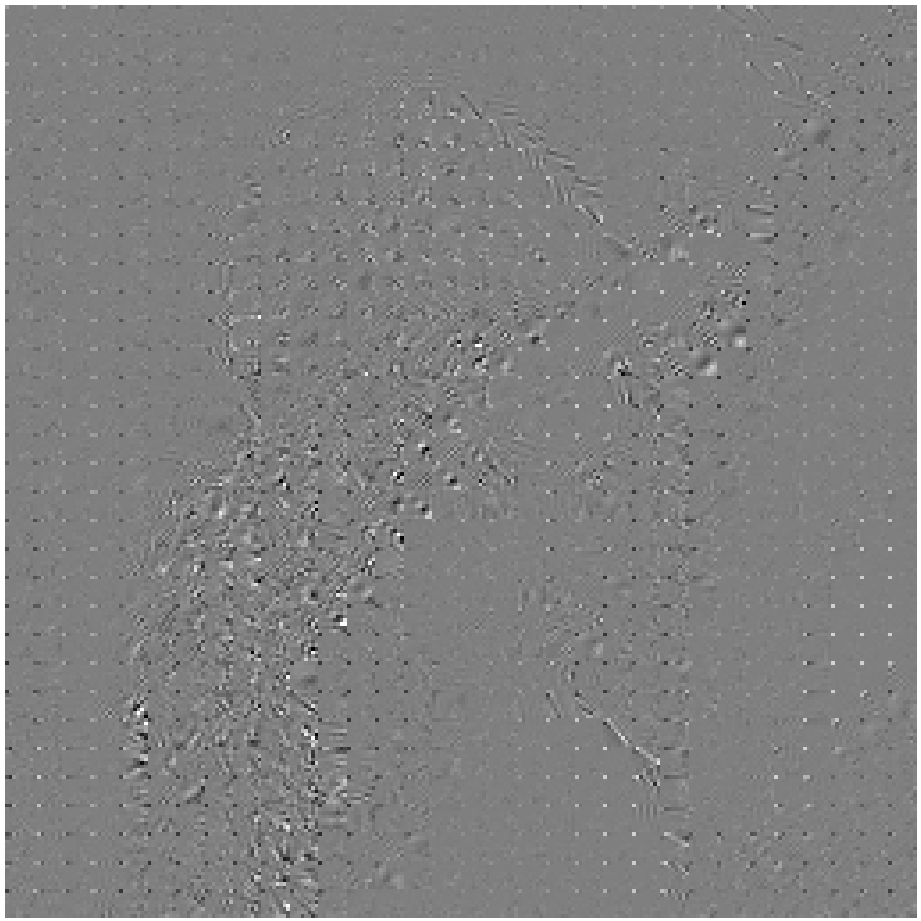


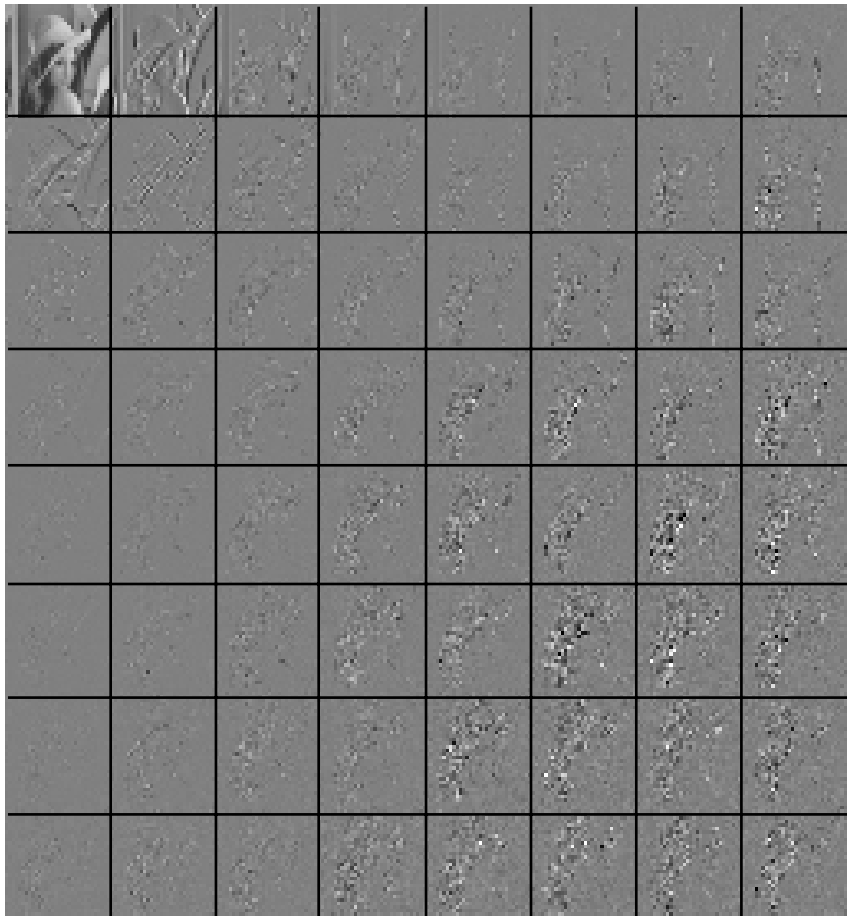
2-D DCT of Image - II



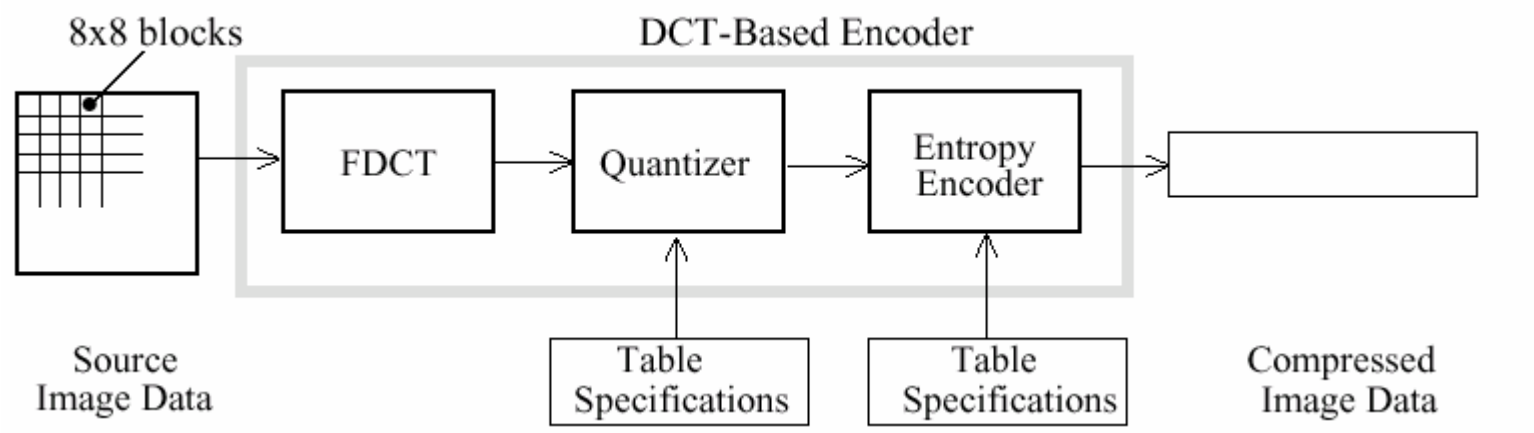
A DCT coefficients is a weight of particular DCT basis function.

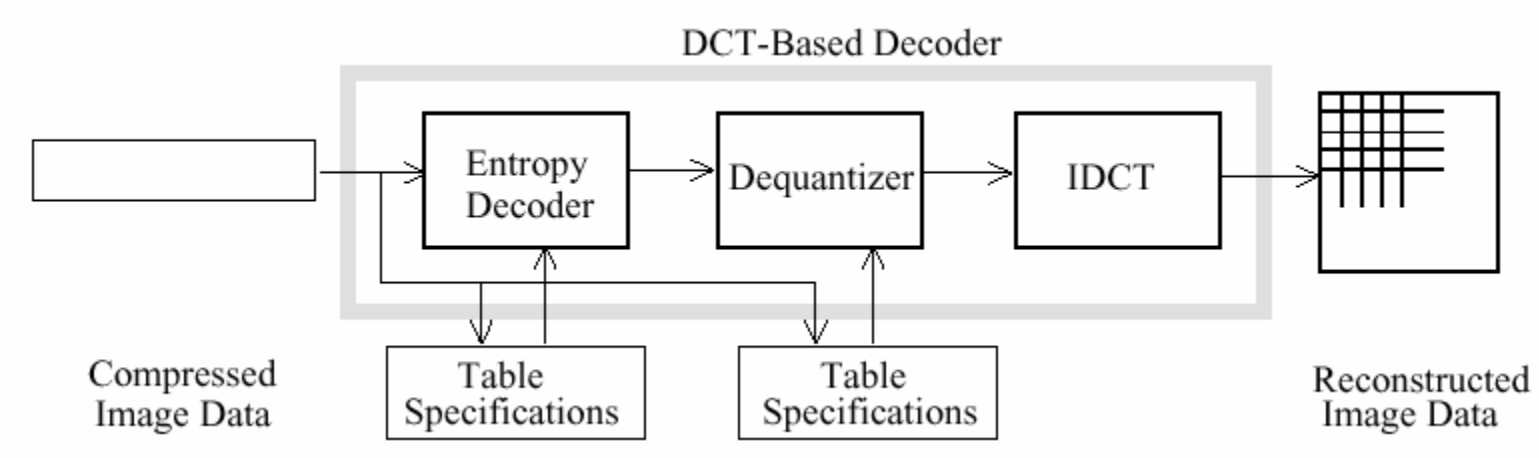
“Grouped” DCT Coefficients





JPEG Compression System





DCT Weight Matrix and Quantization

- Quantization:

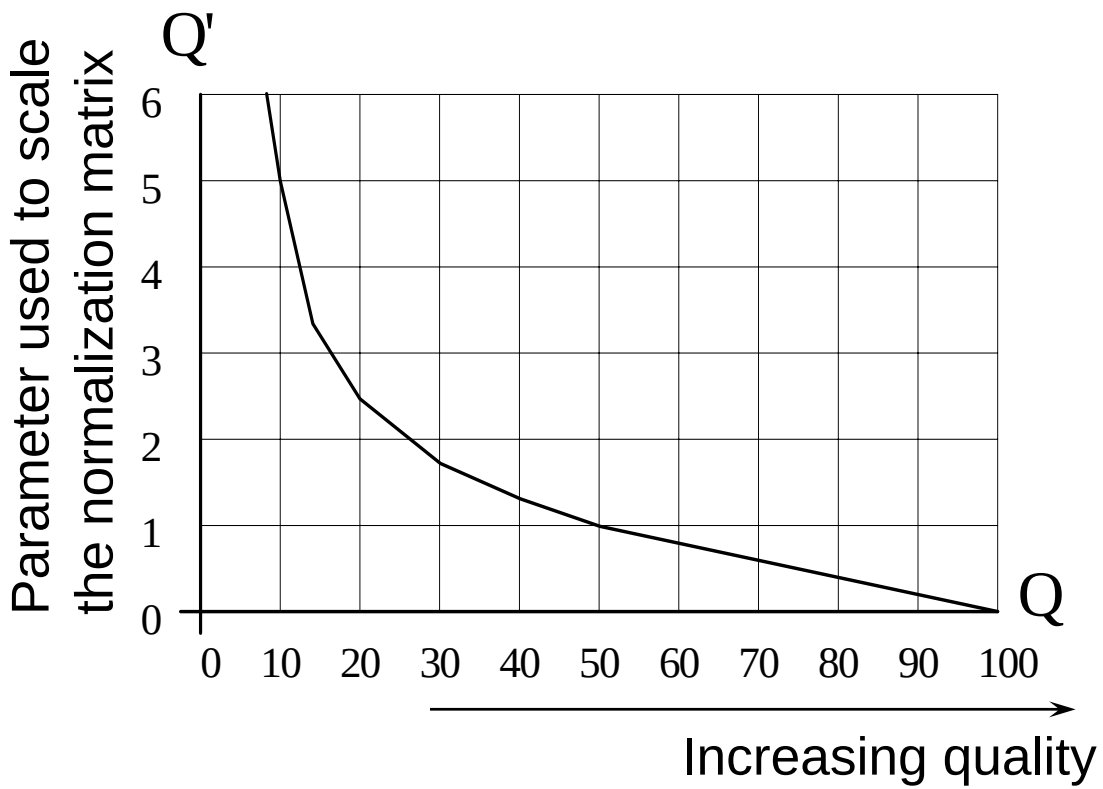
$$\hat{\theta}_{k,l} = Q[\theta_{k,l}] = \text{round}\left(\frac{\theta_{k,l}}{Q' N_{k,l}}\right)$$

- Recommended JPEG normalization matrix

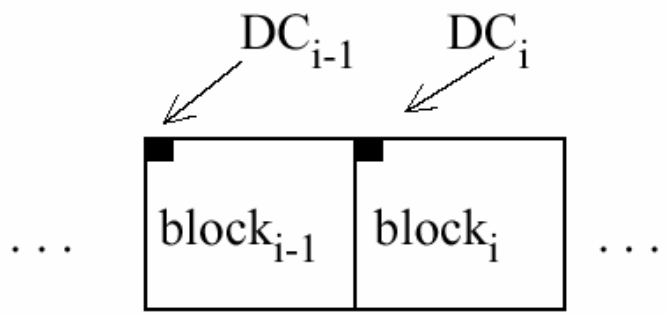
$$N_{k,l} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

User Controllable Quality

- User has control over a “quality parameter” that runs from 100 (“*perfect*”) to 0 (“*extremely poor*”)



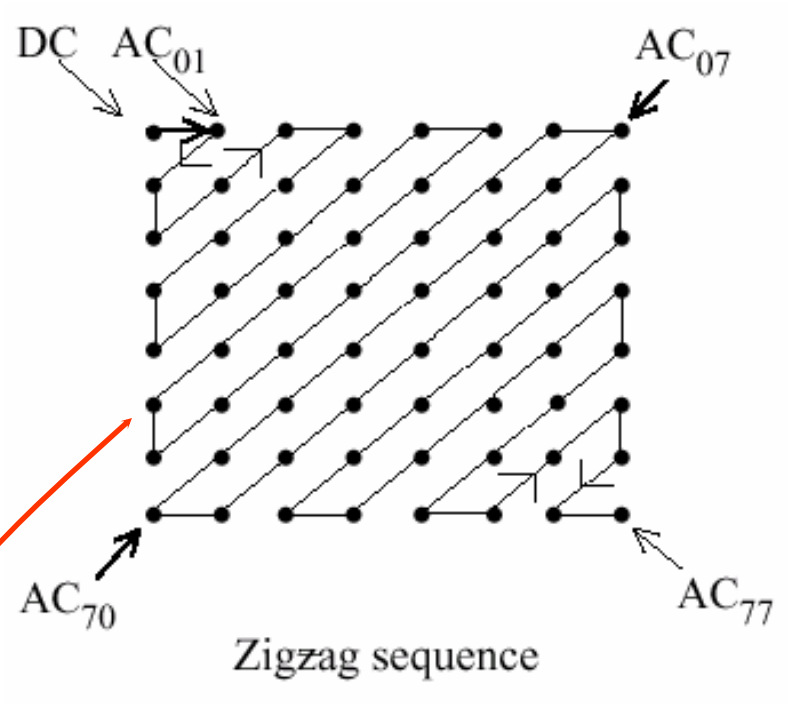
Entropy coding



DIFF = DC$_i$ - DC$_{i-1}$

Differential DC encoding

Huffman coded



Zigzag sequence

Runlength coding
then huffman or
arithmetic coding

Example

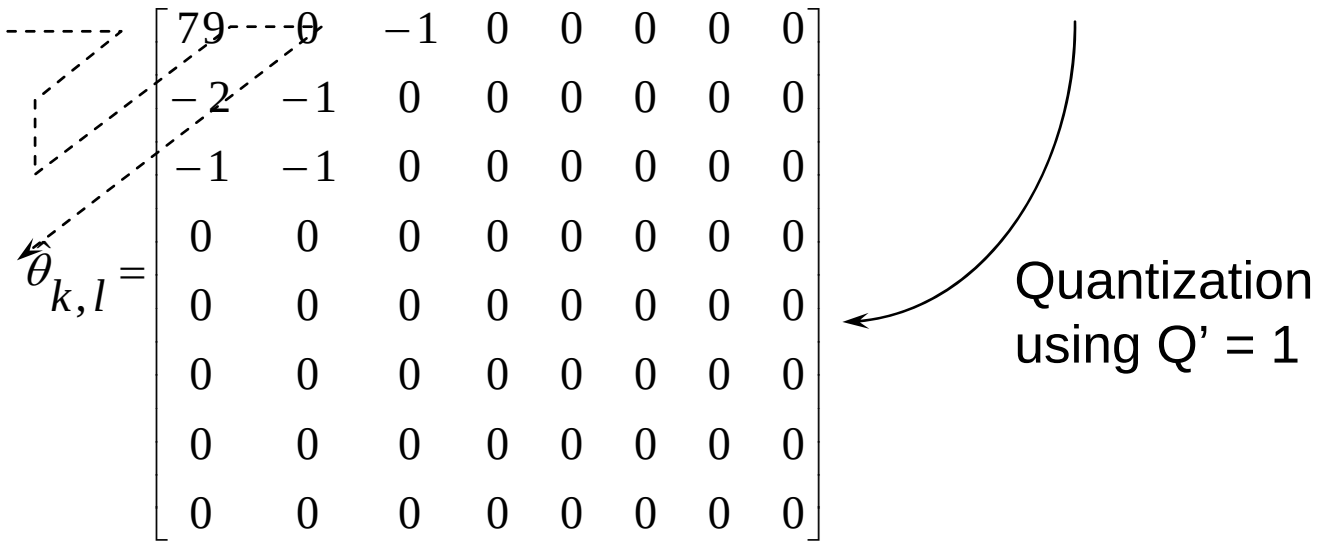
$$f_{k,l} = \begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

average * 8 →

$$\theta_{k,l} = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & 3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

DCT transform is exactly defined in JPEG standard

Example



- *DC*: Difference with quantized DC coefficient of previous block is Huffman encoded
- *AC*: Zig-zag scan coefficients, and convert to (*zero run-length*, *amplitude*) combinations:
  - (79) 0 -2 -1 -1 -1 0 0 -1 EOB
  - {1,-2}{0,-1}{0,-1}{0,-1} {2,-1} EOB

VLC Coding of AC Coefficients

- The (zero run-length, amplitudes) are put into categories

Category	*AC Coefficient Range*
1	-1,1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255
9	-511,...,-256,256,...,511
10	-1023,...,-512,512,...,1023

- The (*zero run-length*, *categories*) are Huffman encoded
- The sign and offset into a category are FLC encoded (required #bits = category number)

JPEG AC Huffman Table

Zero Run	*Category*	*Code length*	*Codeword*
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
.	.	.	.
.	.	.	
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
.	.	.	.
.	.	.	.
2	1	5	11011
2	2	8	11111000
.	.	.	.
.	.	.	.
3	1	6	111010
3	2	9	111110111
.	.	.	.
.	.	.	.
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
11	1	9	111111010
.	.	.	.
.	.	.	.
EOB		4	1010

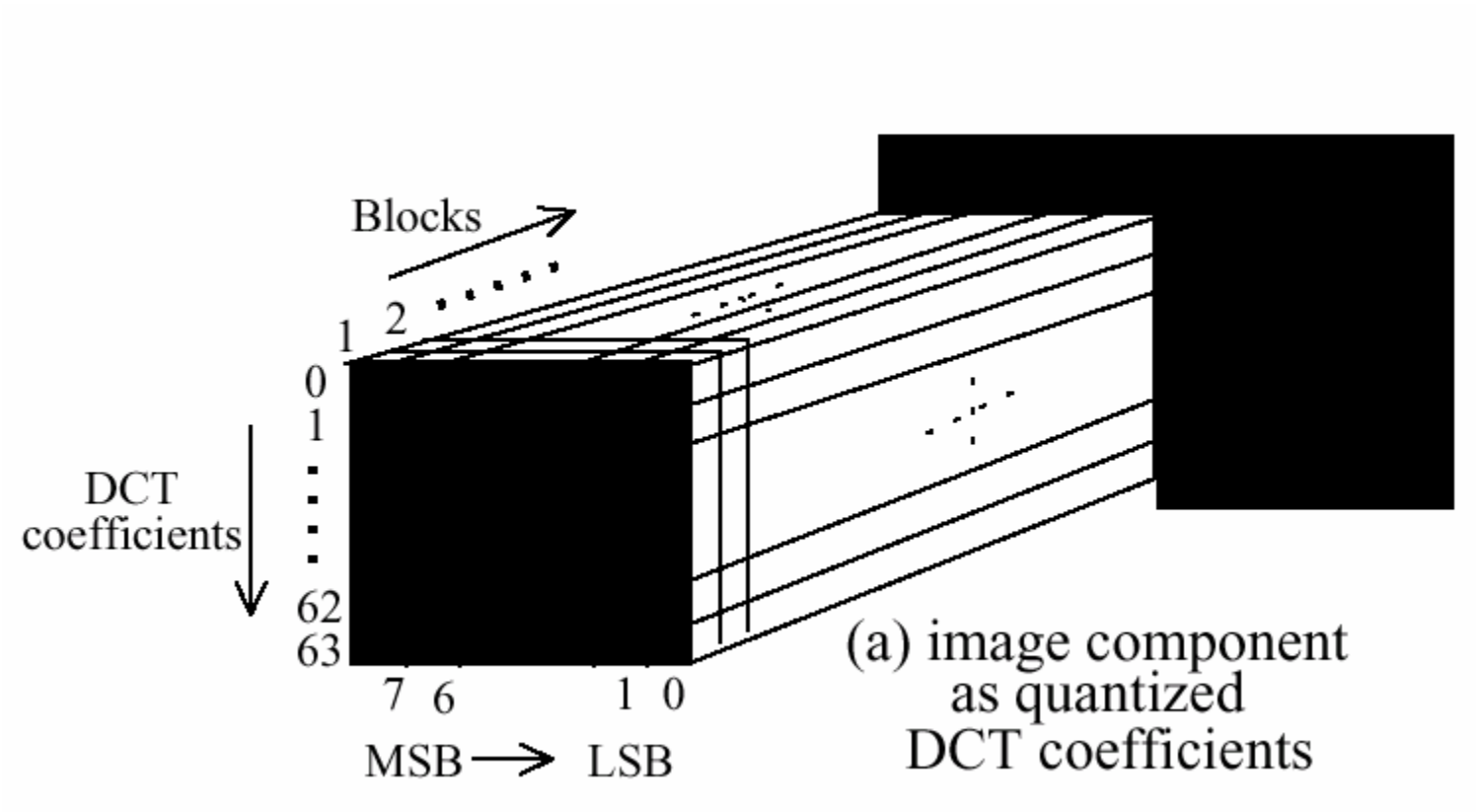
Example - III

- The series {1,-2}{0,-1}{0,-1}{0,-1} {2,-1} EOB now becomes

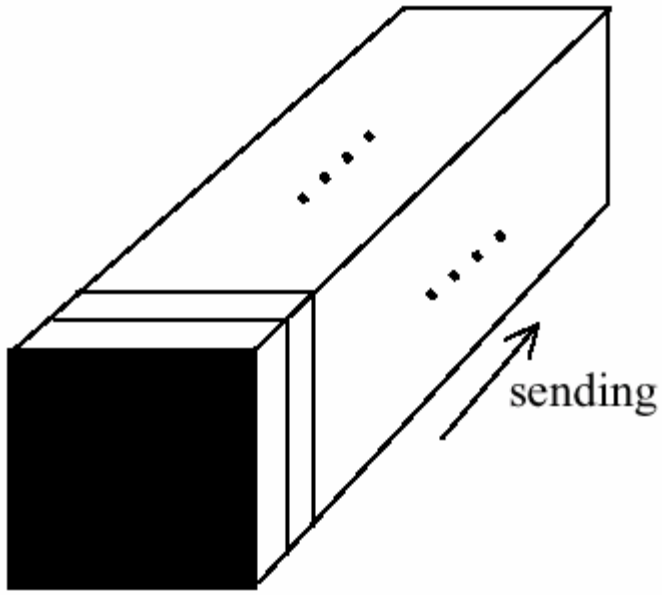
  111001 01/00 0/00 0/00 0/11011 0/1010

- Bit rate for AC coefficients in this DCT block 27 bits/64 pixels = 0.42 bit/pixel

Ordering of Coefficient



Sequential Encoding



Acknowledgement

- Prof. Inald Lagendijk, Delft University of Technology
- Also try the software VcDemo
  - http://ict.ewi.tudelft.nl/index.php?Itemid=124