Lossy Data Compression System

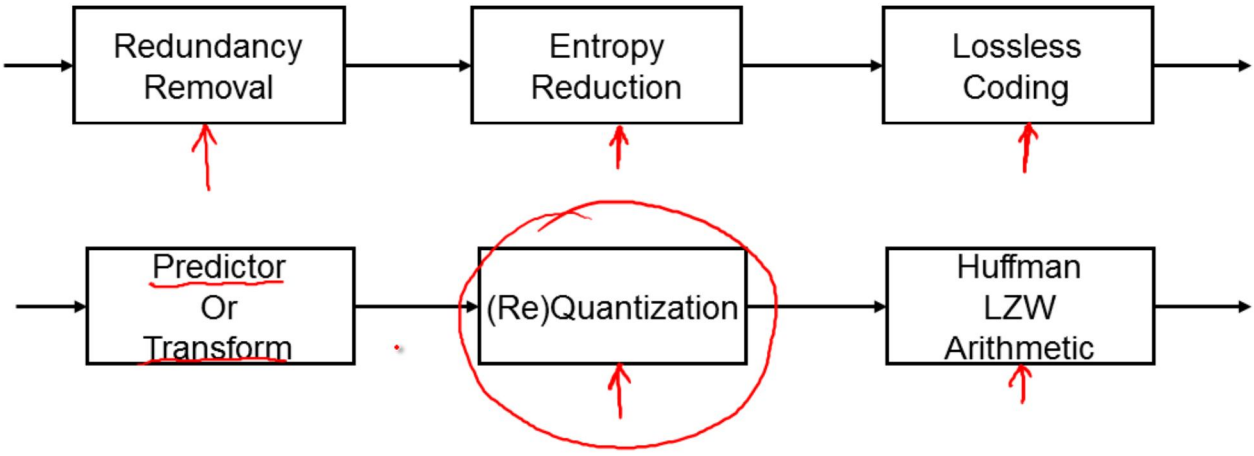
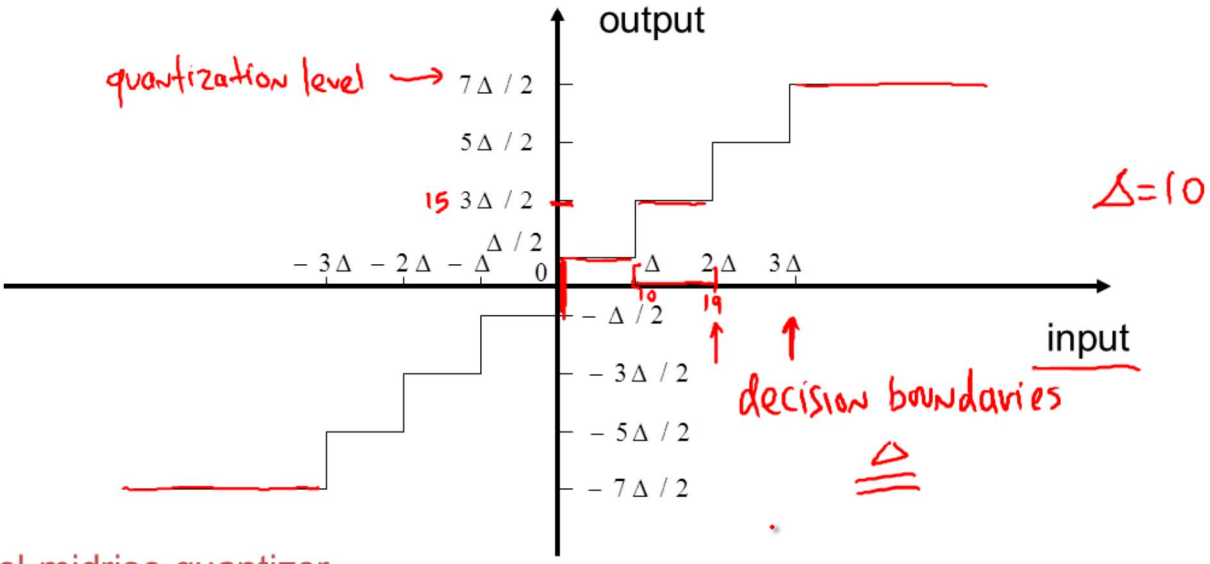


Image Compression

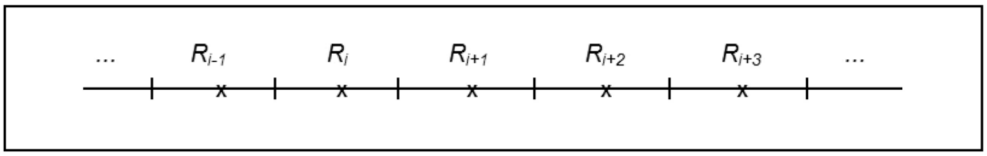
- Quantization (scalar and vector)
- Basic Approaches
  - Subsampling, PCM
- Differential Encoding
  - DPCM
- Fractal Encoding
- Transform Coding and JPEG

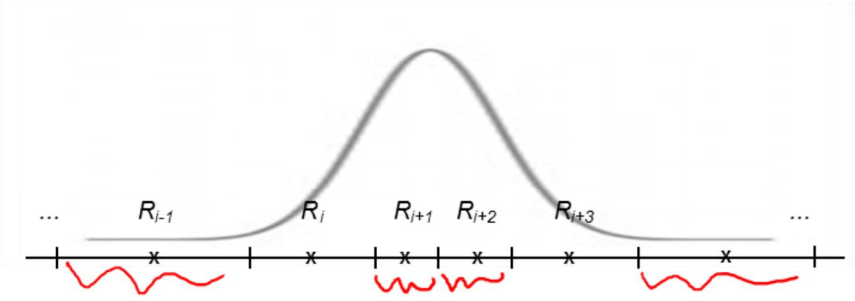
Uniform Quantization



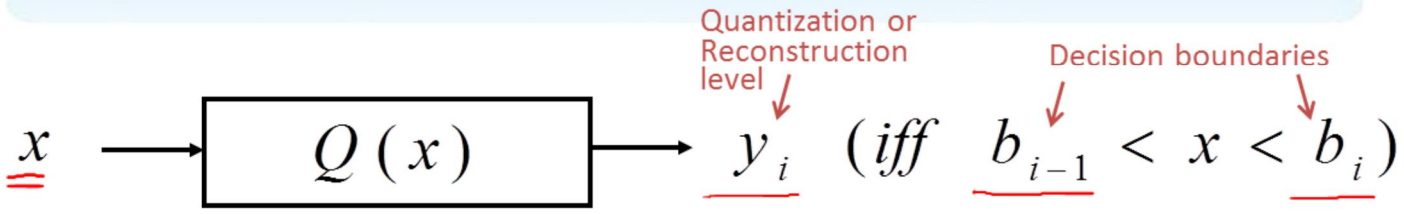
8 level midrise quantizer

Scalar Quantization





The Quantization Problem



Variance of quantization error

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_x(x)\, dx$$

input pdf

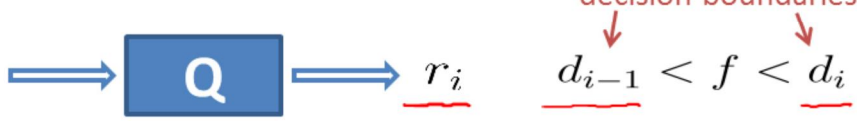
codeword length

rate

$$R = \sum_{i=1}^{M} l_i \int_{b_{i-1}}^{b_i} f_x(x)\, dx$$

Minimize variance of quantization error wrt decision boundaries, reconstruction levels, and binary codes, subject to a rate constraint (or its dual)

Optimal Max-Lloyd Quantizer



Design Problem

Assuming we use fixed codes to represent r_i

design an L-level quantizer (find d_i's and r_i's)

s.t. the variance of the quantization error is minimized

$$D = \sum_{i=1}^{L} \int_{d_{i-1}}^{d_i} (r_i - f)^2 p(f)\, df$$

Necessary conditions for a minimum:

$$\frac{\partial D}{\partial r_k} = 0,\ 1 \le k \le L; \quad \frac{\partial D}{\partial d_k} = 0,\ 1 \le k \le L-1, \text{ given } d_0, d_L$$

Optimal Max-Lloyd Quantizer

$$\frac{\partial D}{\partial r_k} = \int_{d_{k-1}}^{d_k} p(f) 2(r_k - f) df = 0$$

$$\Rightarrow \quad r_k \int_{d_{k-1}}^{d_k} p(f) df = \int_{d_{k-1}}^{d_k} f p(f) df$$

$$\Rightarrow \quad \boxed{r_k = \frac{\int_{d_{k-1}}^{d_k} f p(f) df}{\int_{d_{k-1}}^{d_k} p(f) df}}$$

Optimal Max-Lloyd Quantizer

Leibnitz's formula

$$\frac{d}{da} \int_{\phi_1(a)}^{\phi_2(a)} F(x,a) dx = \int_{\phi_1(a)}^{\phi_2(a)} \frac{\partial F(x,a)}{\partial a} dx + F(\phi_2, a) \frac{d\phi_2(a)}{da} - F(\phi_1, a) \frac{d\phi_1(a)}{da}$$

$$\frac{\partial}{\partial d_k} \left(\int_{d_{k-1}}^{d_k} p(f)(r_k - f)^2 df + \int_{d_k}^{d_{k+1}} p(f)(r_{k+1} - f)^2 df \right)$$

$$= p(d_k)(r_k - d_k)^2 \frac{dd_k}{d_k} - p(d_k)(r_{k+1} - d_k)^2 \frac{dd_k}{d_k} = 0$$

$$\Rightarrow \quad (r_k - d_k)^2 - (r_{k+1} - d_k)^2 = 0 \qquad (a-b)^2 = (a-b)(a+b)$$

$$\Rightarrow \quad (r_k - d_k - r_{k+1} + d_k)(r_k - d_k + r_{k+1} - d_k) = 0$$

Optimal Max-Lloyd Quantizer

Uniform pdf $\quad p(f) = c$

$$r_k = \frac{c\int_{d_{k-1}}^{d_k} f\,df}{c\int_{d_{k-1}}^{d_k} df} = \frac{\frac{1}{2}(d_k^2 - d_{k-1}^2)}{d_k - d_{k-1}} = \frac{1}{2}(d_k + d_{k-1})$$

$$d_k = \frac{r_k + r_{k+1}}{2}$$

➡ The optimal quantizer is the uniform quantizer

Uniform Quantization

Optimum step size and SNR for uniform quantizers
for different distributions and alphabet sizes

	Alphabet Size	Uniform step size	Uniform SNR	Gaussian step size	Gaussian SNR	Laplacian step size	Laplacian SNR
$M=2^1$	2	1.732	6.02 ✓	1.596	4.40	1.414	3.00
$M=2^2$	4	0.866	12.04	0.995	9.24	1.087	7.05
	8	0.433	18.06	0.586	14.27	0.730	11.39
$M=2^4$	16	0.217	24.08 ✓	0.335	19.36	0.460	15.84

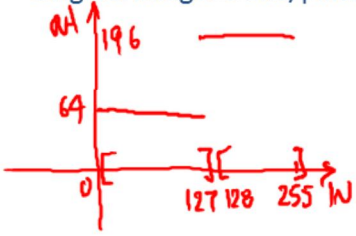
Uniformly Distributed Source
SNR (dB) = 6.02n dB (M=2 exp(n))

Non-Uniform Quantizers

Levels	Gaussian d_i	Gaussian r_i	Gaussian SNR	Laplacian d_i	Laplacian r_i	Laplacian SNR
4	0.0	0.4528	9.3 dB	0.0	0.4196	7.54 dB
	0.9816	1.510		1.1269	1.8340	
6	0.0	0.3177	12.41 dB	0.0	0.2998	10.51 dB
	0.6589	1.0		0.7195	1.1393	
	1.447	1.894		1.8464	2.5535	
8	0.0	0.2451	14.62 dB	0.0	0.2334	12.64dB
	0.7560	0.6812		0.5332	0.8330	
	1.050	1.3440		1.2527	1.6725	
	1.748	2.1520		2.3796	3.0867	

Uniform Quantization

Original image: 8 bits/pixel



1 bit/pixel

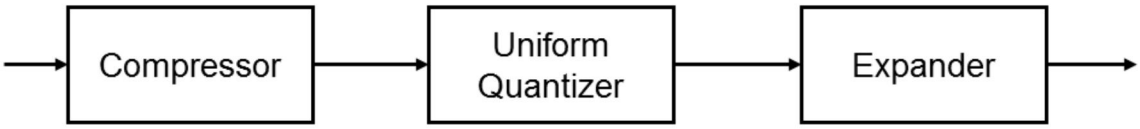
3 bits/pixel

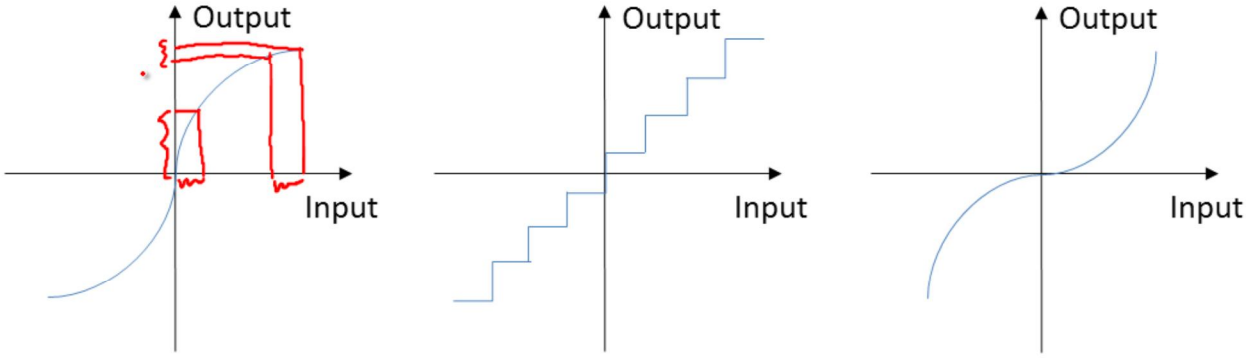
2 bits/pixel

4 bits/pixel

x 2

Companding

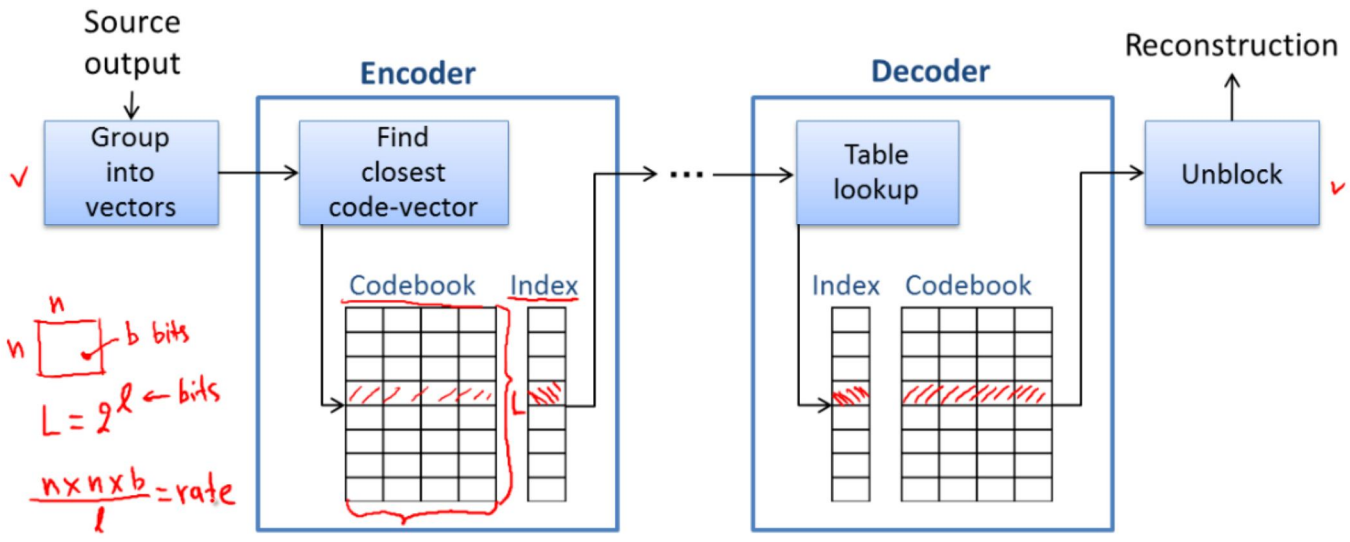




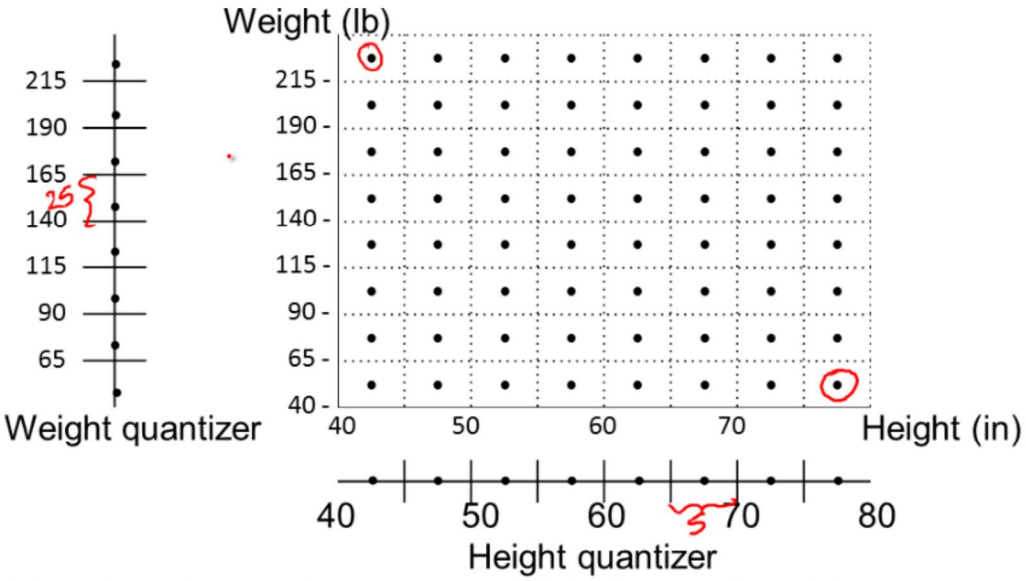
Compressor | Uniform quantizer | Expander

Examples: mu-law, A-law

Vector Quantization

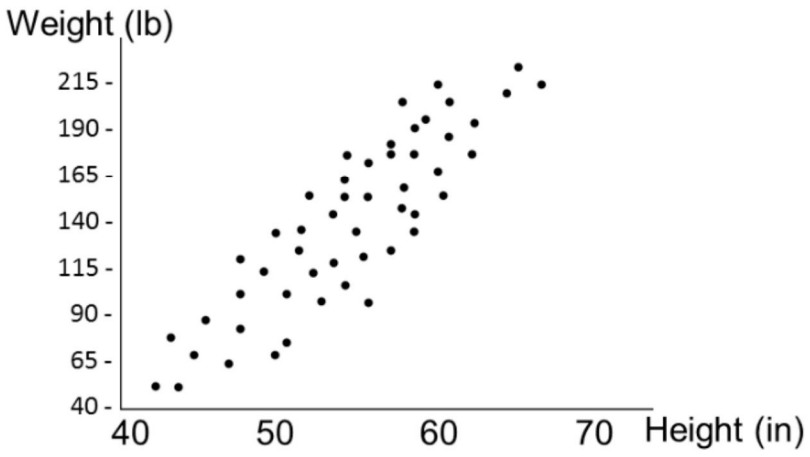


Vector vs Scalar Quantization



K. Sayood, Introduction to Data Compression, 2nd ed., Morgan Kaufmann, 2000.

Vector vs Scalar Quantization

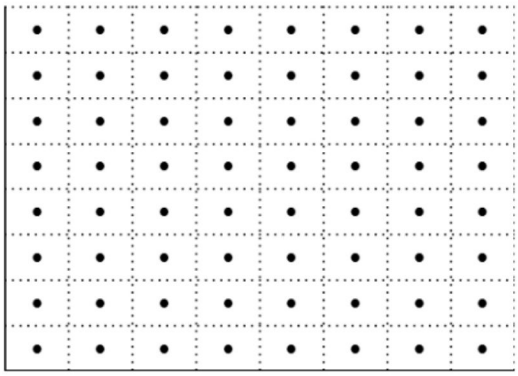


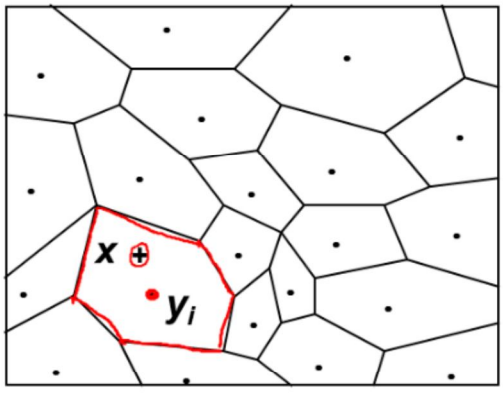
Example

- Codebook:
  - i=1 : $y_1 = (0, 0)$ $(0-0)^2+(0-1)^2=1$
  - i=2 : $y_2 = (2, 1)$
  - i=3 : $y_3 = (1, 3)$
  - i=4 : $y_4 = (1, 4)$ $(1-0)^2+(4-1)^2=10$

- Signal : 0 1 2 3 2 0
- Transmit to decoder : 1 3 2
- Decoded signal : 0 0 1 3 2 1
- Quantization error : 0 -1 -1 0 0 1

VQ Partitions





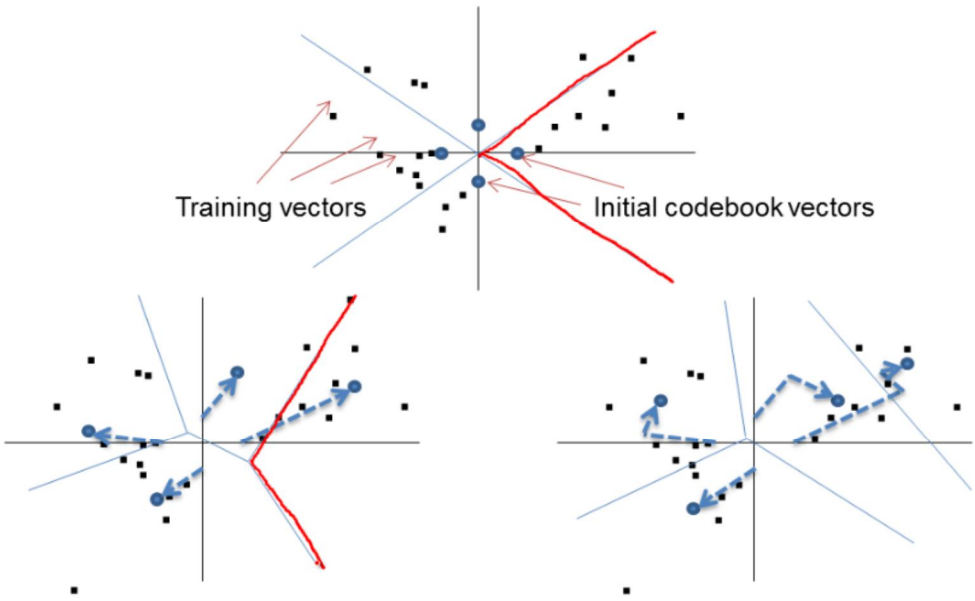
VQ Design Technique

Generalized Lloyd Algorithm or LBG Algorithm

1. Begin with an initial Codebook C_1 of size N. Let the iteration counter be m=1 and the initial distortion $E_1 = \infty$
2. Then using codebook $C_m = \{ \boldsymbol{y}_i \}$, partition the training set into cluster sets R_i using the Nearest Neighbor Condition.
3. Once the mapping of all the input vectors to the initial codevectors is made, compute the centroids of the partition region found in step 2. This gives an improved codebook C_{m+1}.
4. Calculate the average distortion E_{m+1}. If $E_m - E_{m+1} < T$ then stop, otherwise $m = m + 1$ and repeat step 2.

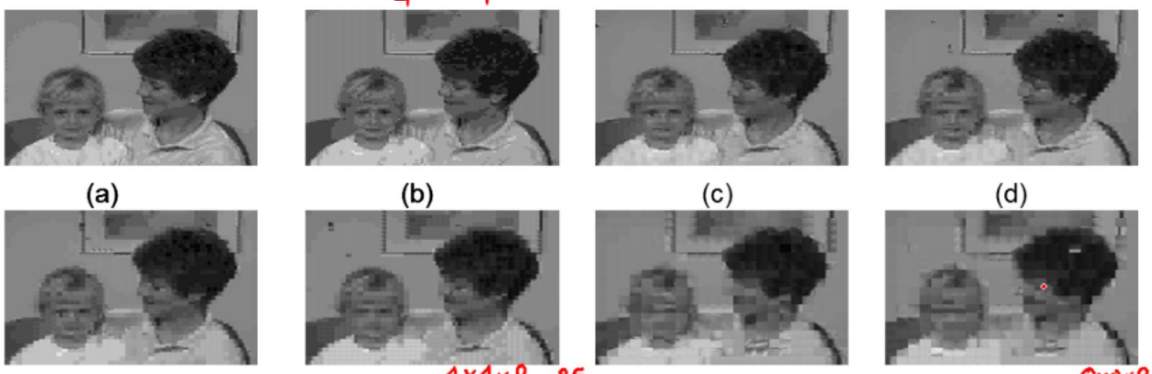
- Image to be encoded must not be in the training set
- Nature of VQ very sensitive to initial codebook and image being encoded
- GLA generates only a locally minimal codebook
- No global minimal codebook design technique currently exists

Codebook Design Example



VQ Results

$\frac{2\times2\times8}{4} = \frac{8}{1}$



(a) (b) (c) (d)

(e) (f) $\frac{4\times4\times8}{5} \approx \frac{25}{1}$ (g) (h) $\frac{8\times8\times8}{7} \approx \frac{64}{1}$

Basic Vector Quantization results. The vector size and number of codevectors in the codebook are respectively: (a) 2x2 and 32, (b) 2x2 and 16, (c) 4x4 and 256, (d) 4x4 and 128, (e) 4x4 and 64, (f) 4x4 and 32, (g) 8x8 and 256, and (h) 8x8 and 128 codevectors.

Subsampling

MxN $\frac{M}{2}\times\frac{N}{2}$ 4:1

Factor of 2

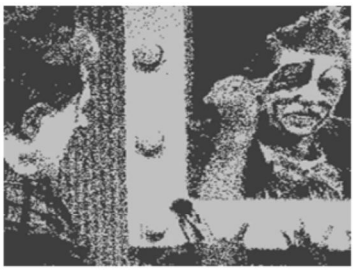
$\frac{16}{1}$

Factor of 4

Pulse Code Modulation (PCM)

Original

1-bit



1-bit/dither

1-bit/dither subtracted

Differential PCM (DPCM)

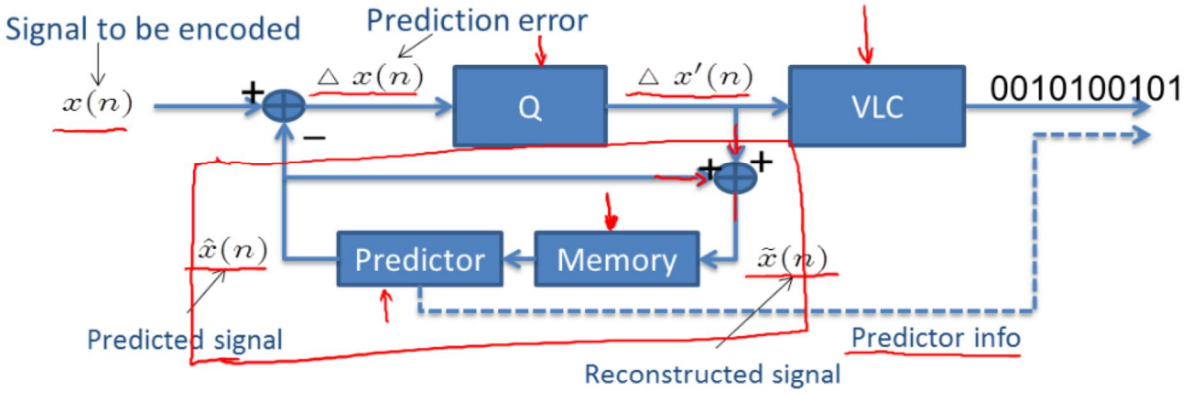
- Useful signals are often highly predictable, i.e, *from the behavior of the past the future signal values can be estimated*
- (Linear) predictability has something to do with the autocorrelation of the signal
- DPCM works on the principle that

Anything that can be predicted from the signal's past can be reconstructed by the decoder

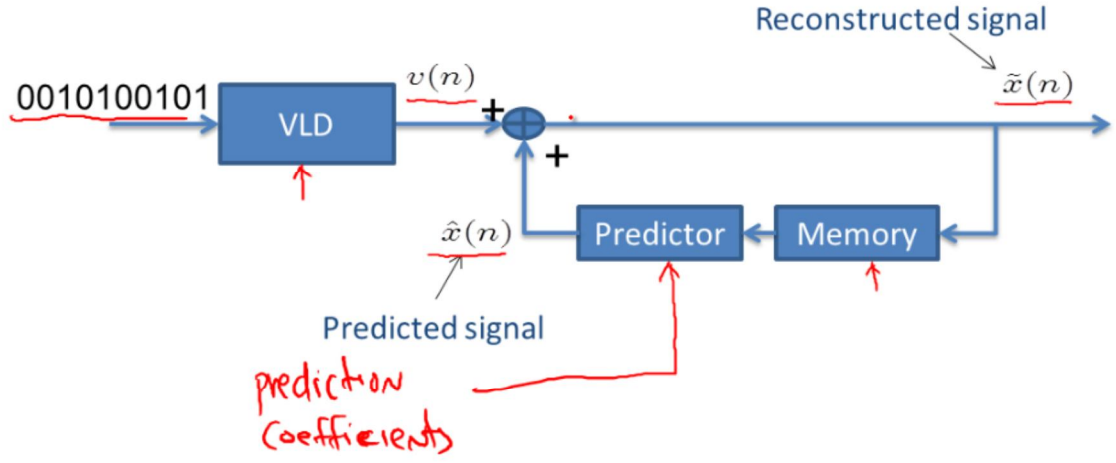
Therefore, only the unpredictable part (prediction error) needs to be encoded and sent to the decoder

- Prediction coefficients Ra=b; R: autocorrelation values, a: predictions coefficient vector, b: autocorrelation values

DPCM Encoder



DPCM Decoder



Prediction in DPCM

x_i : Original signal

Predict: $p_n = f(x_{n-1}, x_{n-2}, \ldots, x_0)$

So that $\sigma_d^2 = E\left[(x_n - p_n)^2\right]$ is minimum

$\Rightarrow E[x_n / x_{n-1}, \ldots]$

Special case:

Linear predictor N-th order: $p_n = \sum_{i=1}^{N} a_i x_{n-i}$ (prediction coefficients)

$$\sigma_d^2 = E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right)^2\right]$$

Prediction in DPCM

$$\frac{\partial \sigma_d^2}{\partial a_i'} = 0, \quad i = 1, \ldots, N \qquad \sigma_d^2 = E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right)^2\right]$$

$$\frac{\partial \sigma_d^2}{\partial a_i'} = E\left[2\left(x_n - \sum_i a_i x_{n-i}\right)(-x_{n-i'})\right]$$

$a_{i'} x_{n-i'}$

$$= -2\left[E[x_n x_{n-i'}] - \sum_i a_i E[x_{n-i} x_{n-i'}]\right] = 0$$

$E[x_{n-i} x_{n-i'}] = R_{xx}(i - i')$

$$\Rightarrow \sum_{i=1}^{N} a_i R_{xx}(i - i') = R_{xx}(i), \quad i = 1, \ldots, N$$

Prediction in DPCM

$$\begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(N-1) \\ R_{xx}(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_{xx}(1) \\ R_{xx}(N-1) & \dots & R_{xx}(1) & R_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} R_{xx}(1) \\ \vdots \\ R_{xx}(N) \end{bmatrix}$$

N×N N×1 N×1

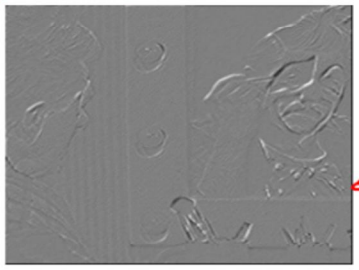
$$Ra = p \quad \Rightarrow \quad a = R^{-1}p$$

Example

original

encoded

grey 127 → 0
white → 255
black → -255
[-255 – 255]



prediction error

prediction mask

Optimal Prediction Coefficients

-0.599	0.847
0.746	x

Normalized Autocorrelation matrix

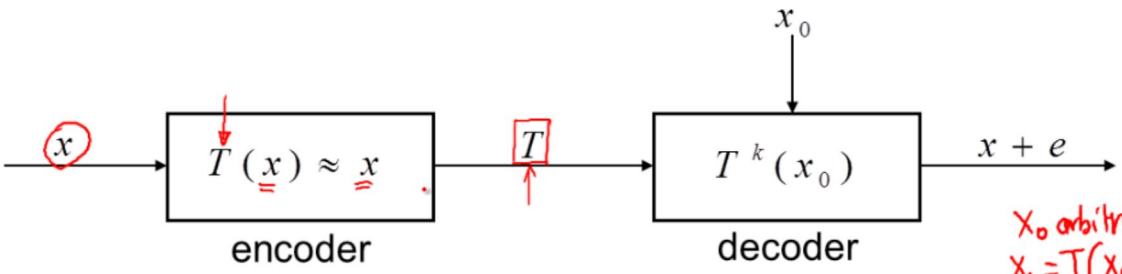
0.851	0.902	0.926	0.899	0.851
0.875	0.939	0.974	0.938	0.877
0.888	0.958	1.000	0.958	0.888
0.877	0.938	0.974	0.939	0.875
0.851	0.899	0.926	0.902	0.851

Coded bitrate: 2.0 (bpp)
Est. entropy-coded bitrate: 1.7 (bpp)
Mean square error: 67.7
Signal-to-noise ratio: 17.7 (dB)
PSNR: 29.8 (dB)

Fractal Image Compression

- "inverse" of fractal computer graphics algorithms
- based on self-similarities
- image encoded as a transformation
- similar to VQ, but does not require transmission of a codebook
- iterative nature

Fractal Codec Structure

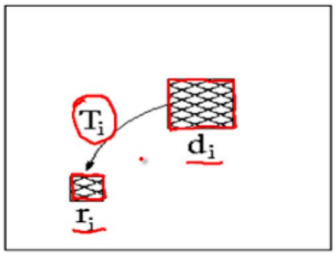


X_0 arbitrary
$X_1 = T(X_0)$
$X_2 = T(X_1) = T(T(X_0)) = T^2(X_0)$

$$x = \lim_{k \to \infty} T^k(x_0) + e$$

Fractal Mapping

Fractal Theory of Iterated Transformations *relies on the assumption that image redundancy can be efficiently exploited through SELF TRANSFORMABILITY on a block-wise basis*

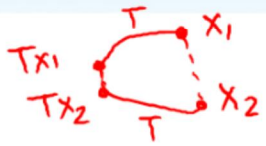


$$r_i = P_i D_i I_i X_i d_i = T_i d_i$$

X_i contractive operator
I_i isometry operator
D_i decimation operator
P_i put operator

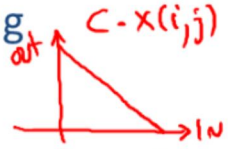
Examples of Contractive Operators

$T: \|Tx_1 - Tx_2\| \le \alpha \|x_1 - x_2\|, \quad \alpha < 1$



- absorption at a certain gray level
- luminance shift $x(i,j) - c$
- contrast scaling $c \cdot x(i,j)$
- color reversal

$x(i,j) = c$



Not all transformations are applied to all blocks

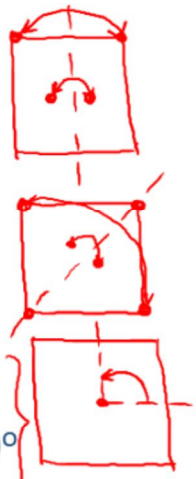
Block types can be classified into

- *shade*
- *midrange*
- *edge*

Examples of Isometries

$T: \|Tx_1 - Tx_2\| \leq \|x_1 - x_2\|$

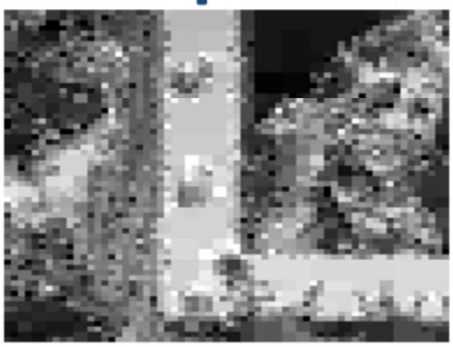
- Identity
- vertical reflection about mid-vertical axis
- vertical reflection about mid-horizontal axis
- vertical reflection about 1st diagonal
- vertical reflection about 2nd diagonal
- rotation around center of block through 90°
- rotation around center of block through 180°
- rotation around center of block through -90°



VQ vs. Fractal

- Codebook <=> virtual codebook
  - set of images ~ original image
  - codebook transmission ~ not necessary
- decoding <=> reconstruction
  - code= list of addresses ~ fractal code = list of transformations consistent with an image transmission
  - direct reconstruction by look-up table ~ iterative reconstruction

Example



it#1
PSNR=17.8dB

it#3
PSNR=26.4dB

it#2
PSNR=22.2dB

it#4
PSNR=29.2dB

Transform Coding

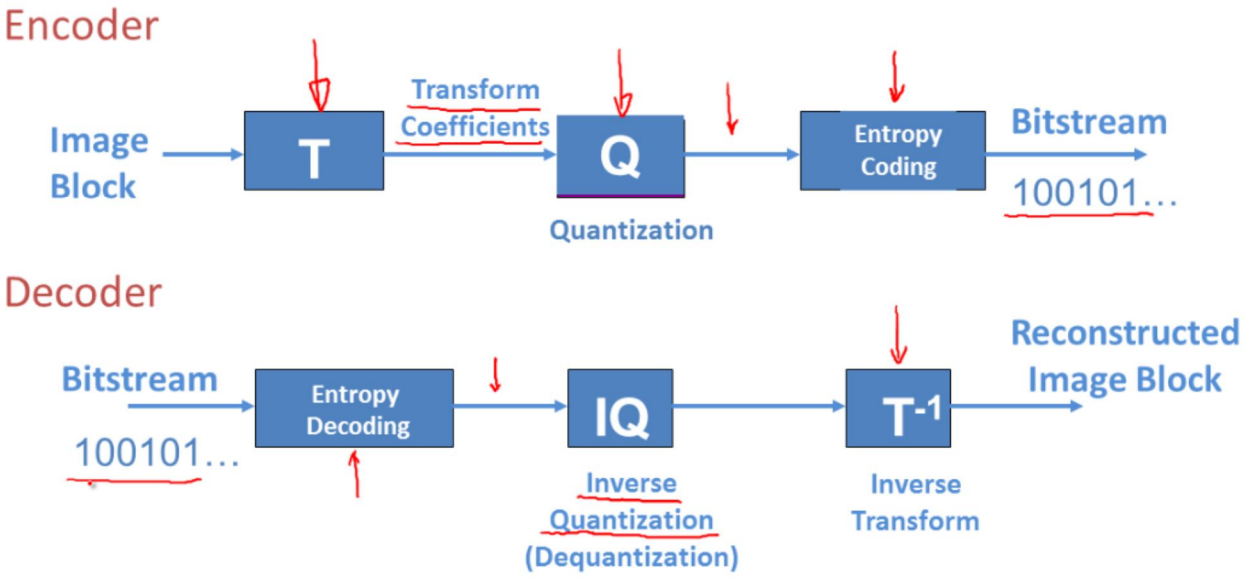
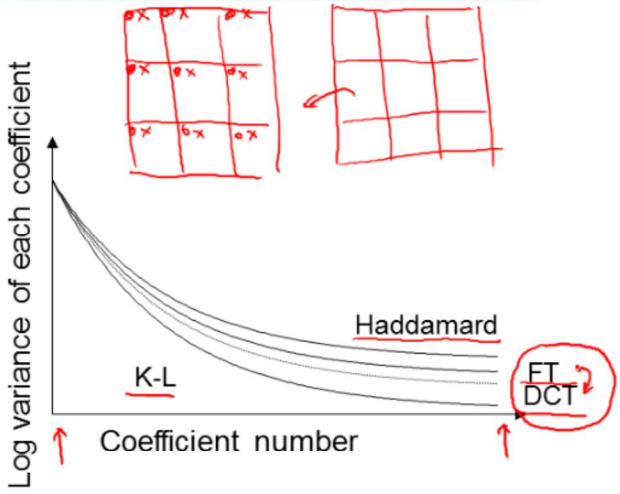


Image Transforms: Characteristics

- Image decorrelation
  - Energy compaction
- Image independent basis functions
- Fast implementation



Linear Transforms

- Karhunen-Loeve Transform (KLT)
  - statistically optimal
  - basis functions image dependent
- Discrete Cosine Transform (DCT)
  - close to KLT for typical images
  - basis functions image independent
  - efficient implementations
  - widely used in JPEG, H.26x and MPEGx

Image Inner Product

Dot product of 2 *N*x*N* matrices

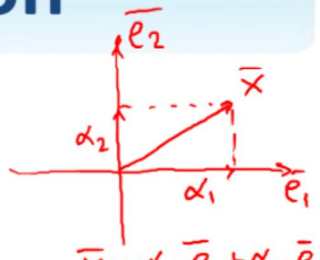
$$[\varphi] \cdot [\Gamma] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \varphi(m,n) \Gamma^*(m,n)$$

Family of matrices $[\varphi^{(u,v)}]$ orthonormal if

$$[\varphi^{(u,v)}] \cdot [\varphi^{(r,s)}] = \begin{cases} 0, & u \neq r \text{ or } v \neq s \\ 1, & u = r \text{ and } v = s \end{cases}$$

Image Decomposition

Given a set of *N*x*N* orthonormal matrices $[\varphi^{(u,v)}]$
we can expand an arbitrary *N*x*N* matrix $[f]$ as

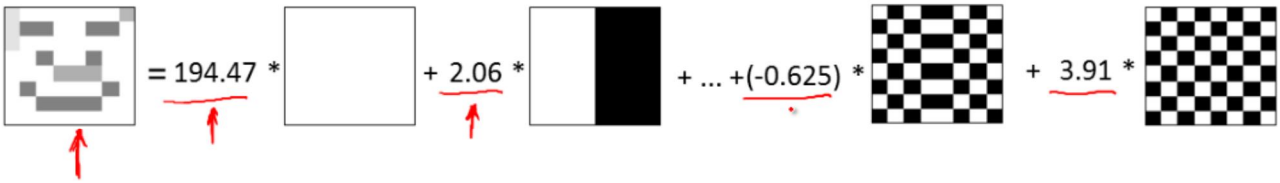


$$[f] = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \, [\varphi^{(u,v)}]$$

$$f = F(0,0)\,\varphi^{(0,0)} + F(0,1)\cdot\varphi^{(0,1)} + \cdots + F(N-1,N-1)\,\varphi^{(N-1,N-1)}$$

where $F(u,v) = [f] \cdot [\varphi^{(u,v)}]$ $\quad u = 0, \ldots, N-1$, $v = 0, \ldots, N-1$

Image Decomposition



Decomposition of an 8*8 gray-scale image using Hadamard bases.
The image is supposed to be a smile face.

Karhunen-Loeve Transformation

Let $R(m,n,p,q)$ be the autocorrelation of $f(m,n)$

$$R(m,n,p,q) = E[f(m,n)f(p,q)]$$

For zero-mean random fields, the orthonormal matrices

$[\varphi^{(u,v)}]$ that result in uncorrelated $F(u,v)$ satisfy

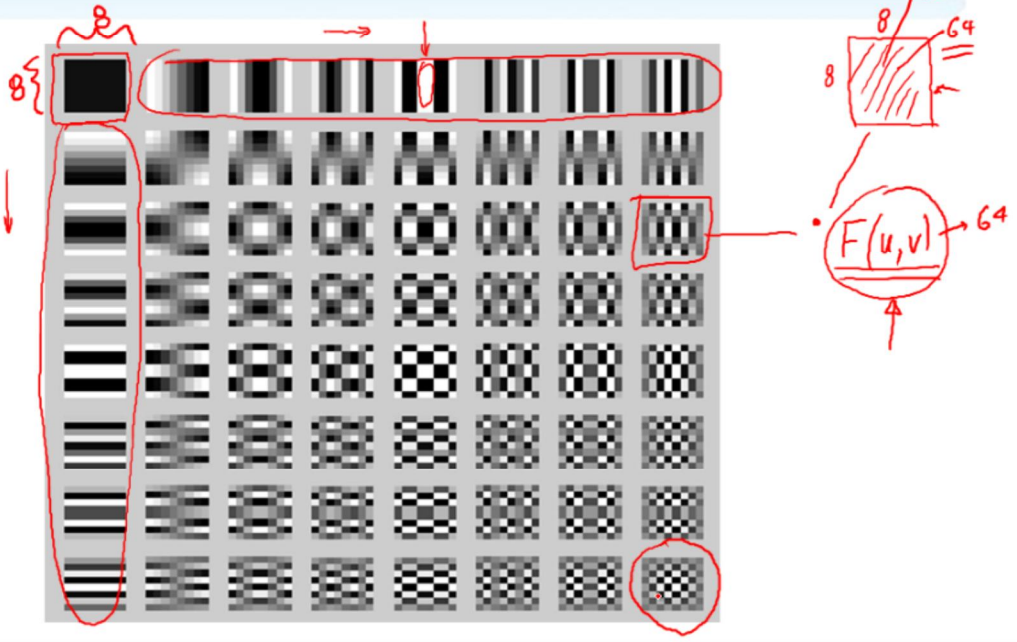
$$\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} R(m,n,p,q)\varphi^{(u,v)}(p,q) = \delta_{uv}\varphi^{(u,v)}(m,n)$$

where $\varphi^{(u,v)}(p,q)$ and $\varphi^{(u,v)}(m,n)$ are the (*p*,*q*)th and (*m*,*n*)th elements of $[\varphi^{(u,v)}]$

$$\delta_{u,v} = E[|F(u,v)|^2]$$

DCT Basis Function

black = 1
white = -1
grey=0



Hadamard Basis Function

black = 1
white = -1

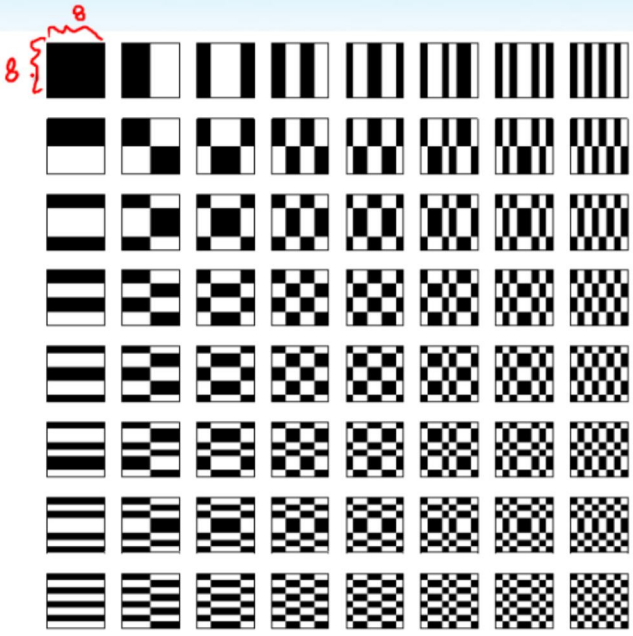
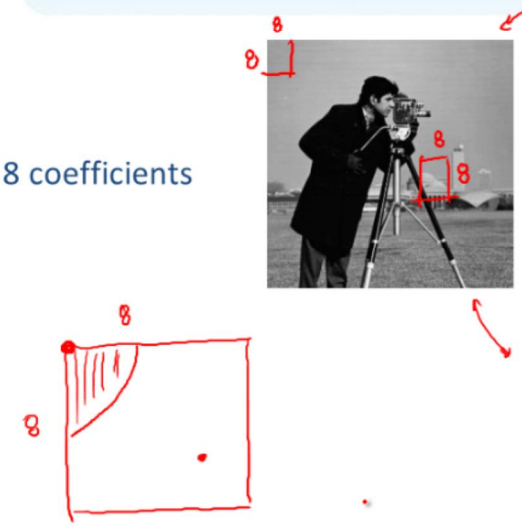
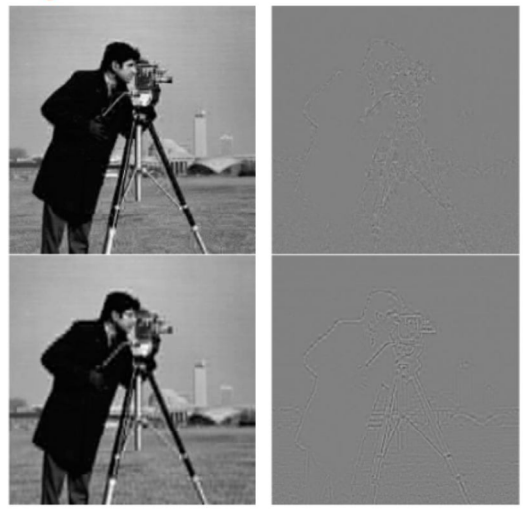


Image Representation Example

8 coefficients



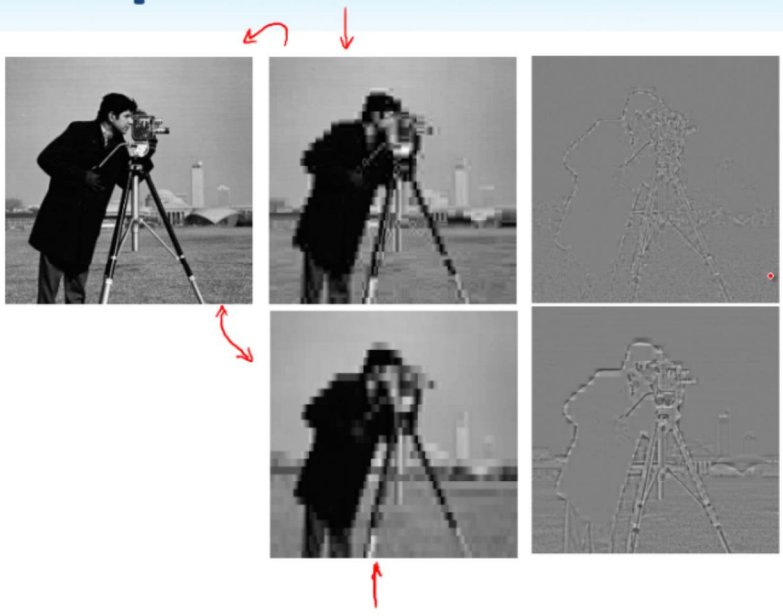


Thresholding

Zonal

Image Representation Example

2 coefficients



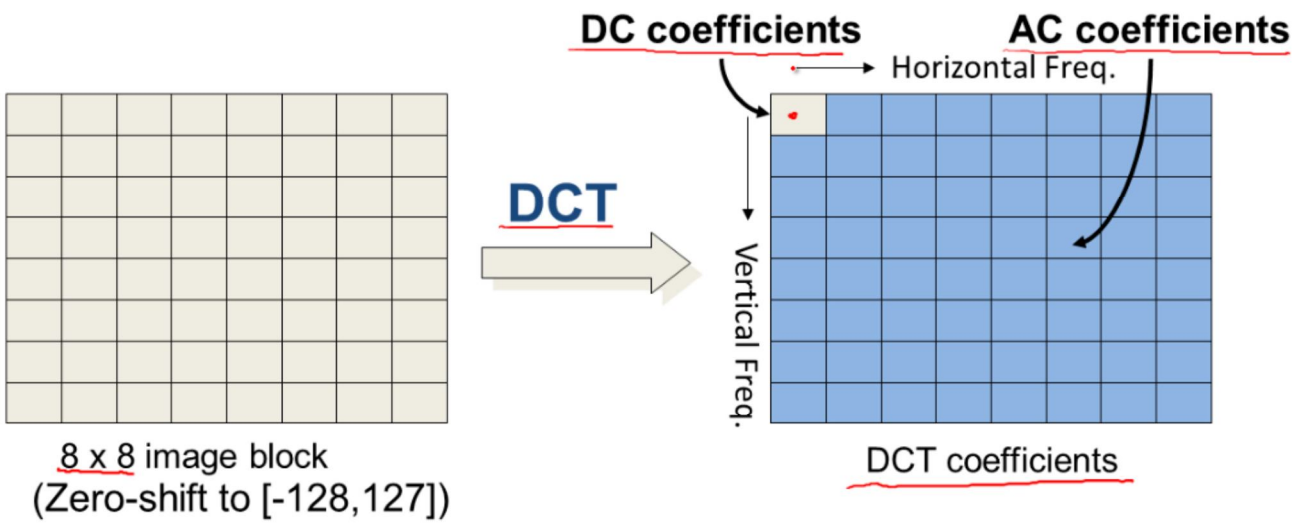
Thresholding

Zonal

JPEG

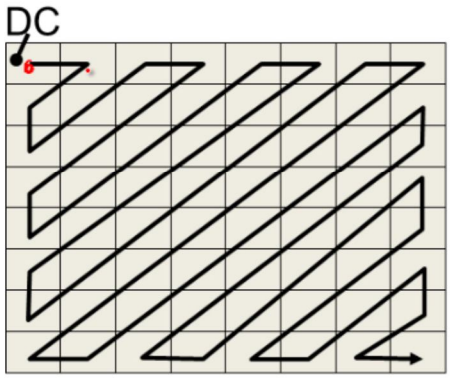
- Joint Photographic Experts Group
  - ISO/IEC JTC1 SC29 WG1
  - Formed in 1986 by ISO and CCITT (ITU-T)
  - Became International Standard (IS) in 1991
  - Compression ratio 10 to 50; 0.5 to 2 bpp.
  - Digital Compression and Coding of Continuous-Tone Still Images (Grayscale or Color)

DC and AC Coefficients



Zig-zag Scan

- Convert two dimensional coefficient block to one dimensional coefficients
- To generate long runs of zeros



JPEG Entropy Encoding

- DC coefficients
  - DC coefficients are coded differentially
  - Each difference is transmitted as a combination of a **Size** codeword (T1) and an **Amplitude** value (T2)
- AC coefficients
  - AC coefficients are encoded after zigzag scan
  - Each *non-zero* coefficient is first coded with a **(Run,Size)** codeword (T3)
  - An **Amplitude** value is then sent (T2)

T1: Huffman for size

Size	Code Length	Codeword
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

T2: Codes for Amplitude

Size	Amplitude	Code
0	0	—
1	-1, 1	0, 1
2	-3, -2, 2, 3	00, 01, 10, 11
3	-7, ..., -4, 4, ..., 7	000, ..., 011, 100, ..., 111
4	-15, ..., -8, 8, ..., 15	0000, ..., 0111, 1000, ..., 1111
5	-31, ..., -16, 16, ..., 31	00000, ..., 01111, 10000, ..., 11111
6	-63, ..., -32, 32, ..., 63	000000, ..., 011111, 100000, ..., 111111
7	-127, ..., -64, 64, ..., 127	0000000, ..., 0111111, 1000000, ..., 1111111
8	-255, ..., 128, 128, ..., 255	00000000, ..., 01111111, 10000000, ..., 11111111
9	-511, ..., -256, 256, ..., 511	000000000, ..., 011111111, 100000000, ..., 111111111
10	-1023, ..., -512, 512, ..., 1023	0000000000, ..., 0111111111, 1000000000, ..., 1111111111
11	-2047, ..., -1024, 1024, ..., 2047	00000000000, ..., 01111111111, 10000000000, ..., 11111111111

T3: Run/Size Huffman

Run/ Size	Code Length	Codeword	Run/ Size	Code Length	Codeword	Run/ Size	Code Length	Codeword
0/0 (EOB)	4	1010						
0/1	2	00	**1/1**	4	1100	**2/1**	5	11100
0/2	2	01	**1/2**	5	11011	**2/2**	8	11111001
0/3	3	100	**1/3**	7	1111001	**2/3**	10	1111110111
0/4	4	1011	**1/4**	9	111110110	**2/4**	12	111111110100
0/5	5	11010	**1/5**	11	11111110110	**2/5**	16	1111111110001001
0/6	7	1111000	**1/6**	16	1111111110000100	**2/6**	16	1111111110001010
0/7	8	11111000	**1/7**	16	1111111110000101	**2/7**	16	1111111110001011
0/8	10	1111110110	**1/8**	16	1111111110000110	**2/8**	16	1111111110001100
0/9	16	1111111110000010	**1/9**	16	1111111110000111	**2/9**	16	1111111110001101
0/A	16	1111111110000011	**1/A**	16	1111111110001000	**2/A**	16	1111111110001110

JPEG Coding Example

- Divide Image into Blocks

181	181	181	178	180	176	179	177
174	179	177	176	175	178	180	173
190	178	176	176	174	175	177	176
194	177	173	174	175	176	175	177
182	176	178	178	178	179	172	176
140	105	174	121	114	223	95	133
175	161	227	196	199	239	205	204
199	201	232	218	214	235	222	213

JPEG Coding Example

- Calculate the DCT

181	181	181	178	180	176	179	177
174	179	177	176	175	178	180	173
190	178	176	176	174	175	177	176
194	177	173	174	175	176	175	177
182	176	178	178	178	179	172	176
140	105	174	121	114	223	95	133
175	161	227	196	199	239	205	204
199	201	232	218	214	235	222	213

DCT

1444	-12	-17	13	-22	8	57	-6
-50	29	29	-3	24	-5	-55	6
97	-19	-15	-13	-12	5	-8	8
-93	-3	-12	9	-13	-8	45	-12
58	12	11	-7	17	-1	-42	8
15	-9	0	1	2	5	7	7
-58	13	2	11	-6	-5	29	-13
55	-5	1	-11	10	4	-29	12

JPEG Coding Example

1444	-12	-17	13	-22	8	57	-6
-50	29	29	-3	24	-5	-55	6
97	-19	-15	-13	-12	5	-8	8
-93	-3	-12	9	-13	-8	45	-12
58	12	11	-7	17	-1	-42	8
15	-9	0	1	2	5	7	7
-58	13	2	11	-6	-5	29	-13
55	-5	1	-11	10	4	-29	12

X

round $\left(\frac{X(u,v)}{Q(u,v)}\right)$

Q

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

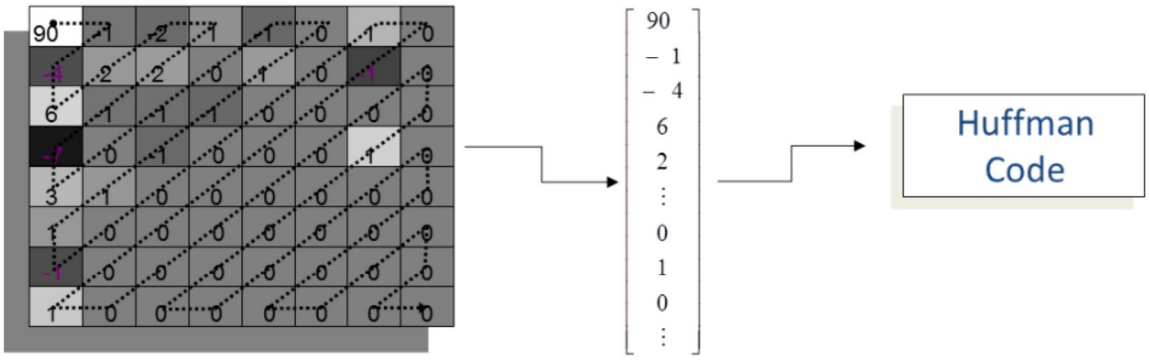
Typical JPEG Quantizer

- Quantize the DCT Coefficients

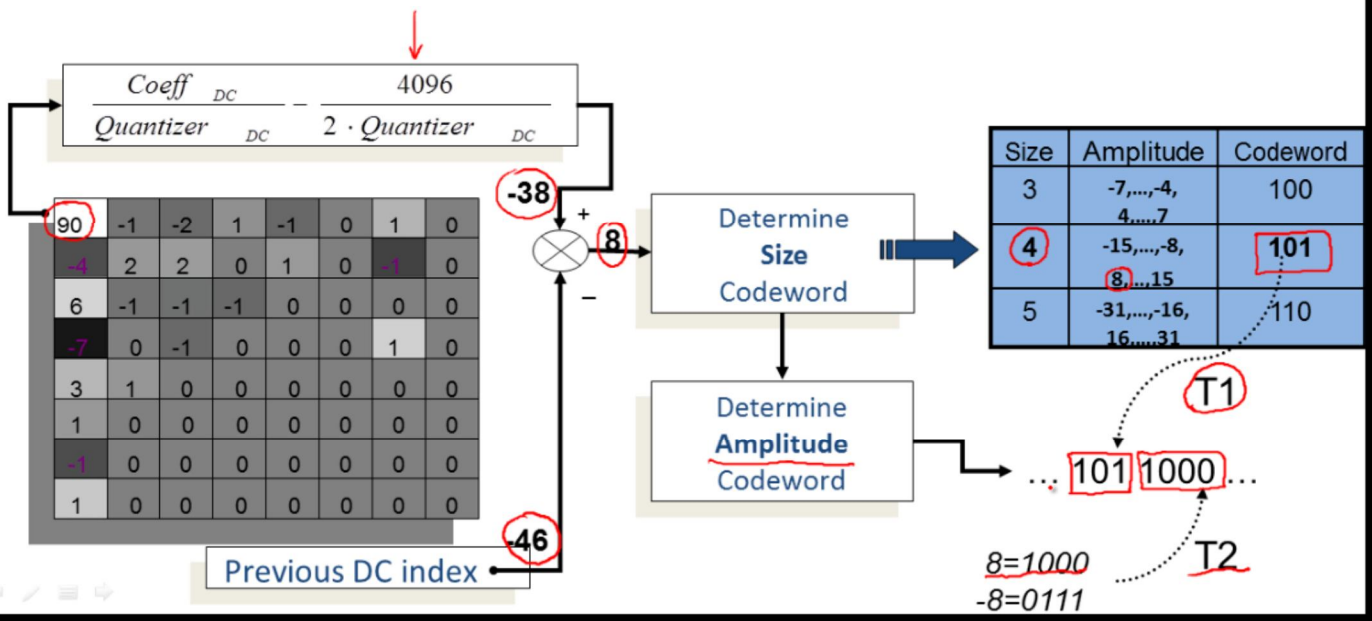
90	-1	-2	1	-1	0	1	0
-4	2	2	0	1	0	-1	0
6	-1	-1	-1	0	0	0	0
-7	0	-1	0	0	0	1	0
3	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0

JPEG Coding Example

- Vectorize the Coefficients

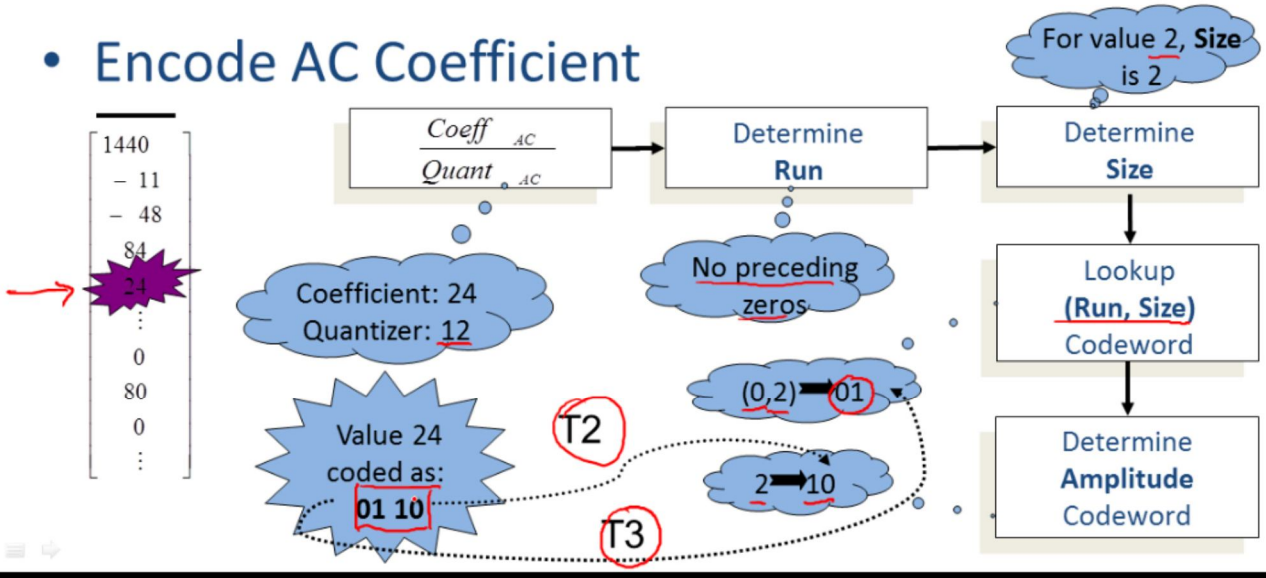


JPEG Coding Example



JPEG Coding Example

- Encode AC Coefficient



JPEG Coding Example

- Calculate the Inverse-DCT

1440	-11	-20	16	-24	0	51	0
-48	24	28	0	26	0	-60	0
84	-13	-16	-24	0	0	0	0
-98	0	-22	0	0	0	80	0
54	22	0	0	0	0	0	0
24	0	0	0	0	0	0	0
-49	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0

IDCT

183	170	195	184	179	184	161	178
170	177	162	173	175	165	181	173
185	202	158	180	186	169	206	179
189	187	166	167	174	176	180	163
189	160	198	165	172	210	157	168
131	95	170	122	131	189	111	138
182	170	229	194	199	244	191	205
192	207	232	214	213	234	218	211

JPEG Coding Example

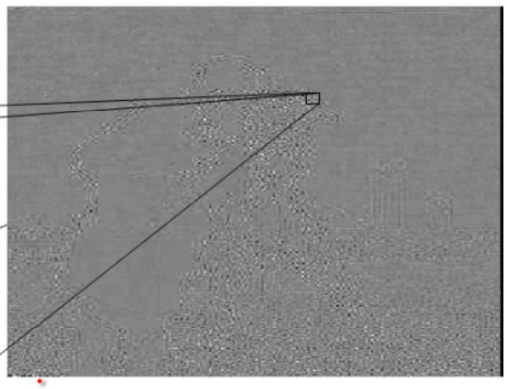
- Assemble the Image from Blocks

183	170	195	184	179	184	161	178
170	177	162	173	175	165	181	173
185	202	158	180	186	169	206	179
189	187	166	167	174	176	180	163
189	160	198	165	172	210	157	168
131	95	170	122	131	189	111	138
182	170	229	194	199	244	191	205
192	207	232	214	213	234	218	211

JPEG Coding Example

- Compression Errors

2	-11	14	6	-1	8	-18	1
-4	-2	-15	-3	0	-13	1	0
-5	24	-18	4	12	-6	29	3
-5	10	-7	-7	-1	0	5	-14
7	-16	20	-13	-6	31	-15	-8
-9	-10	-4	1	17	-34	16	5
7	9	2	-2	0	5	-14	1
-7	6	0	-4	-1	-1	-4	-2



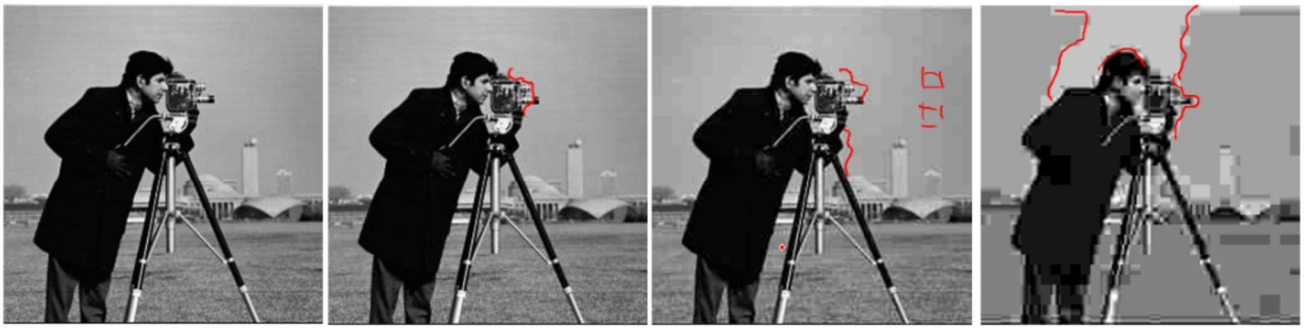
JPEG Picture Quality

- Quality vs. bit rates

	CGA (320x240)	VGA (640x480)	SVGA (800x600)
0.5 bpp	poor	fair	good
1.0 bpp	fair	good	excellent
2.0 bpp	good	excellent	excellent+

- Perceptually lossless at 1.5-2bpp

JPEG Examples



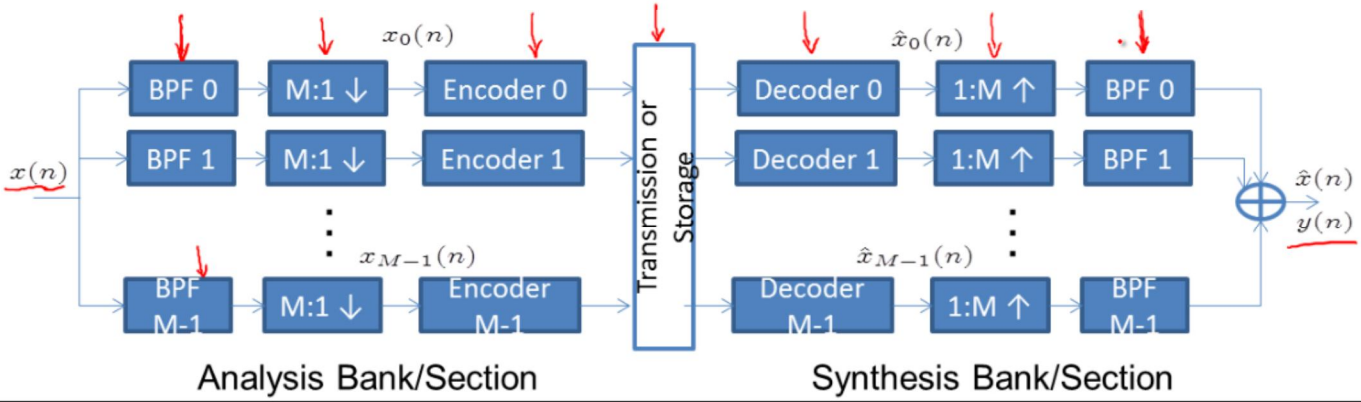
1.5 bpp 1.0 bpp 0.5 bpp 0.2 bpp

Sub-band Coding

- Filter image first to create a set of images (sub-bands –SB-) each of which contains a set of spatial frequencies
- Each SB has reduced bandwidth
- SBs have the same (equal rate) or different bandwidths (multirate)
- Different bit rates or different coding techniques can be used for each SB
  - take advantage of the properties of each SB
  - allow for errors to be distributed across SBs in a visually optimal manner

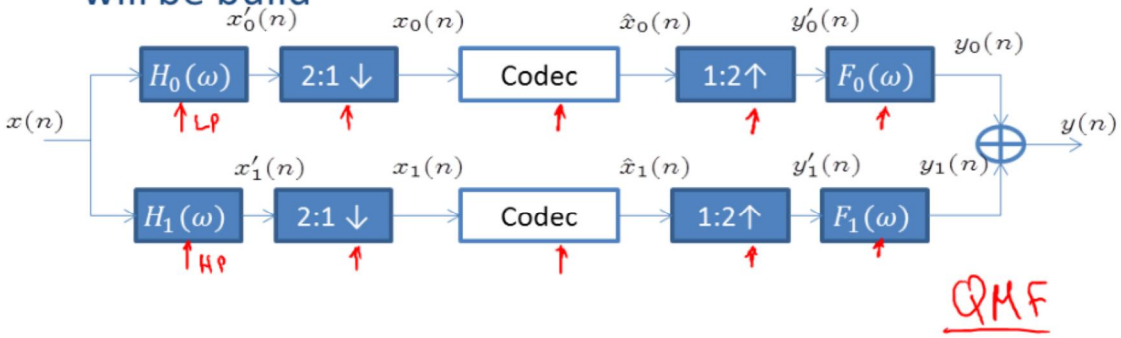
Principle of Sub-band Coding

- Split input signal $x(n)$ into several subbands or subband signals
- Encoder uses analysis filter bank and decoder a synthesis filter bank



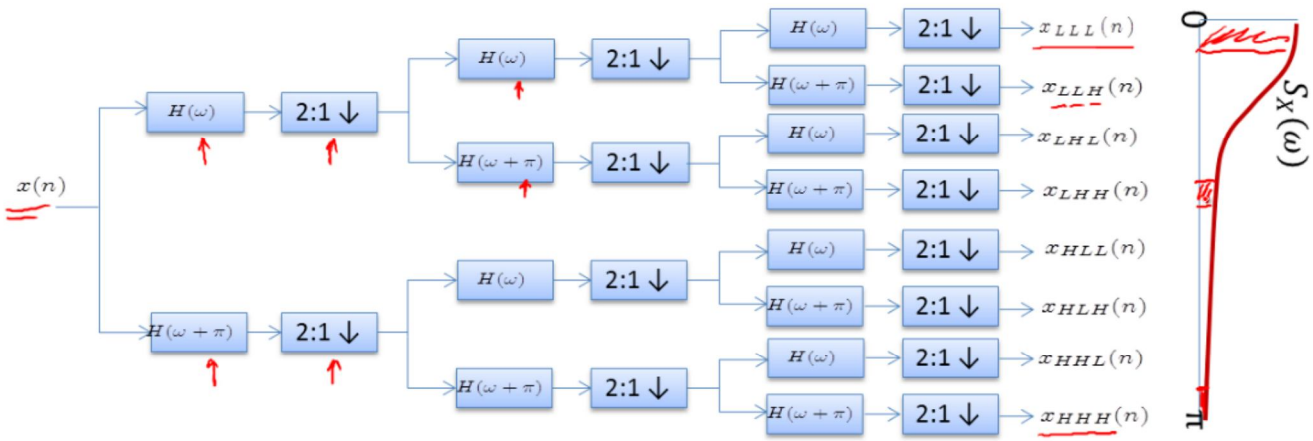
Two-Channel Sub-band Decomposition

- We will start with a 2-channel digital subband analysis and synthesis system
- From this multi-band and multi-dimensional systems will be build



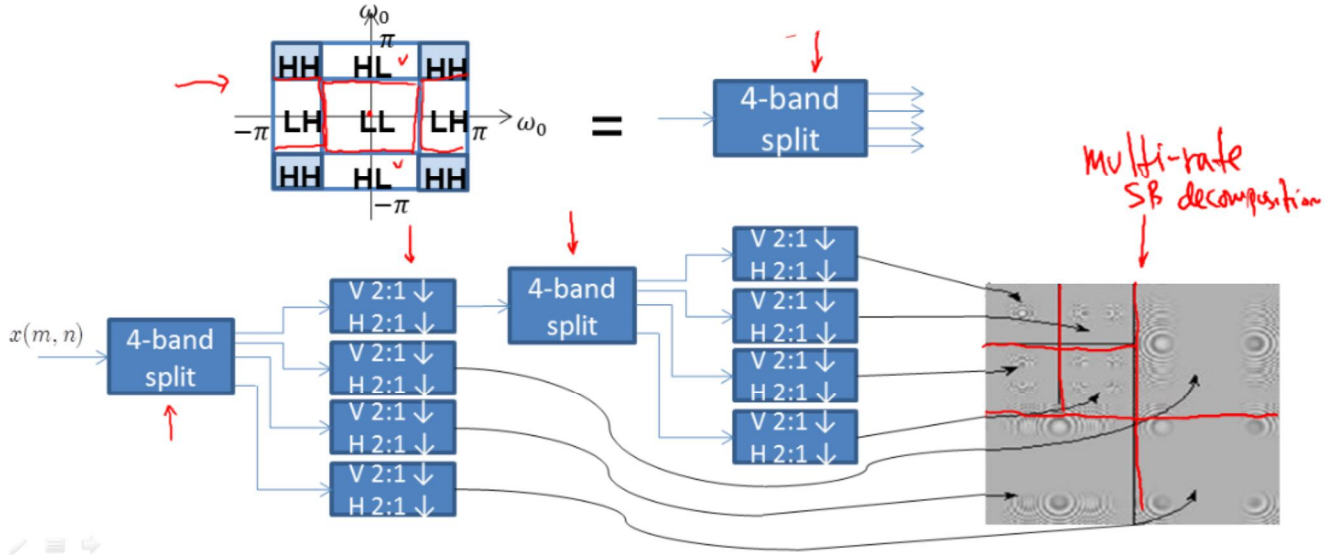
Multiple Sub-band Decomposition

Uniform-band tree-structure (analysis) QMF filter bank



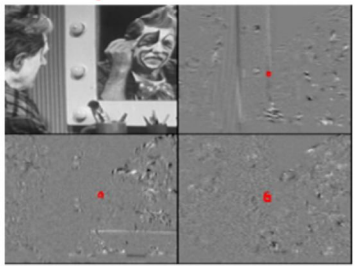
2D Filter banks

- Example: 2-D Octave-band tree-structure

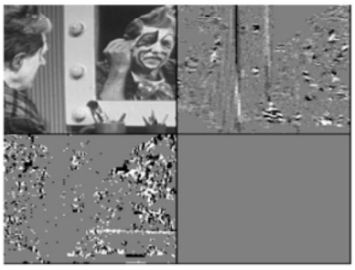


Example #1

Original



4-band decomposition
16-tap filters

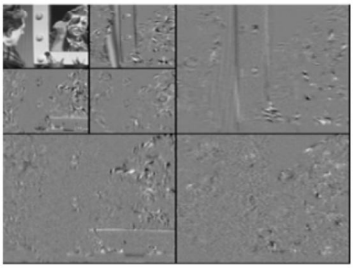


Quantized subbands
1st: DPCM
Others: PCM
1.6 bpp

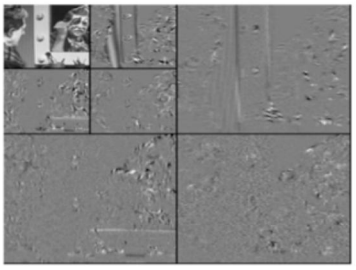
Encoded image
PSNR=36.5 dB

Example #2

Original



7-band decomposition
16-tap filters



Quantized subbands
1st: DPCM
Others: PCM
1.55 bpp

Encoded image
PSNR=36.2 dB