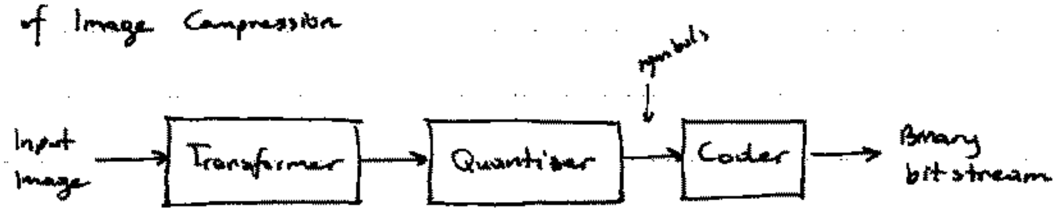
IMAGE COMPRESSION

Image compression methods exploit:

1) Spatial redundancy, due to correlation between neighboring pixels
2) Spectral redundancy, due to correlation among the color components
3) Psychovisual redundancy, due to properties of the human visual system

1. Basics of Image Compression



Transformer : * Applies a one-to-one transformation to the input image
* The output is more suitable for efficient compression than raw data
* DCT : Packs the energy of the signal to a small # of coefficients
* Wavelet Transform : Space - frequency decomposition

Quantiser : * Generates a limited # of symbols that can be used in the representation of the compressed image
* Quantization is a many-to-one mapping → It is irreversible

Coder : * Assigns a codeword, a binary bitstream, to each symbol at the output of the quantiser

* Lossless compression methods aim to minimize the bitrate without any distortion in the image. Quantization is not employed.
* Lossy compression methods aim to obtain the best possible fidelity for a given bitrate or to minimize the bitrate to achieve a given fidelity measure.

* Information Theoretic Concepts:

Source $\mathcal{X} = X_1 X_2 X_3 \ldots$ → a sequence of random variables X_i

X_i takes a value from the alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$

Symbols

* Discrete Memoryless Source (DMS) : Successive symbols are statistically independent.

→ The information content of a symbol is related to the extent that the symbol is unpredictable or unexpected.

→ If a symbol with low probability occurs, a larger amount of information is transferred than in the ocurance of a more likely symbol.

→ The amount of information that a symbol a_i carries is defined as

$$I(a_i) = \log_2\left(\frac{1}{p(a_i)}\right), \text{ for } a_i \in \mathcal{A}$$

where $p(a_i)$ is the probability that a_i occurs.

Note that if $p(a_i) = 1 \Rightarrow I(a_i) = 0$

as $p(a_i) \to 0 \Rightarrow I(a_i) \to \infty$

In practice, the probability of ocurance of each symbol is estimated from the histogram of a specific source, or a training set of sources.

→ The entropy $H(\mathcal{X})$ of a DMS $\mathcal{X}$ is defined as the average information per symbol in the source:

$$H(\mathcal{X}) = \sum_{a \in \mathcal{A}} p(a) \log_2\left(\frac{1}{p(a)}\right) = -\sum_{a \in \mathcal{A}} p(a) \log_2 (p(a))$$

→ The entropy is maximized when all symbols are equally likely.

Example: Entropy of a raw image data

Suppose an 8-bit image is taken as a realization of a DMS $\mathcal{X}$.

The symbols a_i are the gray levels. $\Rightarrow \mathcal{A} = \{0, 1, \dots, 255\}$

The entropy of the image is $H(\mathcal{X}) = -\sum_{a_i=0}^{255} p(a_i) \log_2 p(a_i)$

The entropy of an image consisting of a single gray level is zero.

* Markov-K Source: The probability of occurance of a symbol depends on the values of K preceding symbols.

→ The entropy of a Markov-K source is defined as

$$H(\mathcal{X}) = \sum_{S^K} p(X_{j-1}, \dots, X_{j-K}) \underbrace{H(\mathcal{X} \mid X_{j-1}, \dots, X_{j-K})}$$

All possible realizations of $X_{j-1}, \dots, X_{j-K}$

$$= -\sum_{a \in \mathcal{A}} p(a_i \mid X_{j-1}, \dots, X_{j-K}) \log p(a_i \mid X_{j-1}, \dots, X_{j-K})$$

<u>Lossless Coding Theorem</u>: The minimum bitrate that can be achieved by lossless coding of a DMS $\mathcal{X}$ is given by

$$\min\{R\} = H(\mathcal{X}) + \epsilon \quad \text{bits/symbol}$$

transmission rate

a positive quantity that can be made arbitrarily close to zero.

→ Lossless coding theorem establishes the lower bound for the bitrate necessary to achieve zero coding-decoding error.

<u>Source Coding Theorem:</u> There exists a mapping from the source symbols to codewords such that for a given distortion D, $R(D)$ bits/symbol are sufficient to enable source reconstruction with an average distortion that is arbitrarily close to D. The actual rate R should obey

$$R \geqslant R(D)$$

for the fidelity level D. The function $R(D)$ is called the rate-distortion function. Note that $R(0) = H(\mathcal{X})$

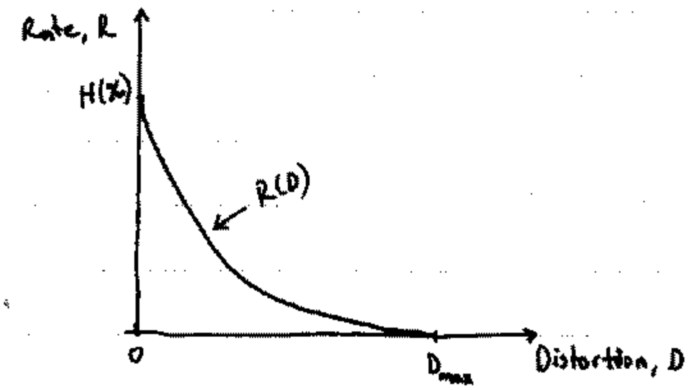


Figure : A typical rate-distortion function for a lossy coding scheme.

→ The rate distortion function can be computed analytically for simple source and distortion models.

→ Computer algorithms exist to compute $R(D)$ when analytical methods fail or are unpractical.

→ In general, we are interested in designing a compression system to achieve either the lowest bitrate for a given distortion or the lowest distortion at a given bitrate.

2. Symbol Coding

→ Symbol coding is the process of assigning a bit string to individual symbols or to a block of symbols comprising the source.

→ The simplest scheme is to assign equal-length codewords to individual symbols or a fixed-length block of symbols, which is known as fixed-length coding.

→ Compression is generally achieved by assigning shorter-length codewords to more probable symbols. When the codeword length is not fixed, such a coding scheme is called variable-length coding.

2.1. Fixed-Length Coding

→ Equal-length codewords are assigned to each symbol in the alphabet regardless of their probabilities.

Example:

Symbol	Codeword 1	Codeword 2	...
a_1	00	00	
a_2	01	01	
a_3	10	11	
a_4	11	10	

If there are M symbols ⇒ the length of codeword $\geqslant \lceil \log_2 M \rceil$

→ Fixed-length coding is optimal only when

1) The # of symbols is equal to a power of 2
2) All the symbols are equiprobable

2.2 Huffman Coding

→ Assigns variable-length codewords to a fixed-length block of symbols.

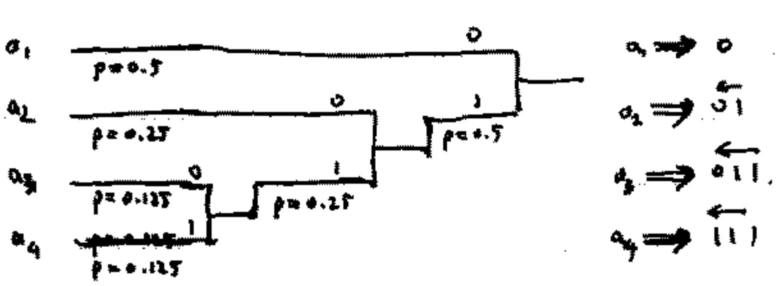
→ Huffman procedure requires a series of source reduction steps.

- At each step, two symbols with the smallest probabilities are merged, which results in a new source with a reduced alphabet.
- The probability of the new symbol is the sum of the probabilities of the two merged symbols.
- The procedure is continued until we reach a source with only two symbols, for which the codeword assignments are 0 and 1.
- Then we work backwards towards the original source, each time splitting the codeword of the merged symbol into two new codewords by appending it with a zero and one.

<u>Example</u>: Consider an alphabet with four symbols having probabilities 0.5, 0.25, 0.125, 0.125. Find the Huffman codes.

Symbol	Probability	Codeword
a_1	0.5	0
a_2	0.25	10
a_3	0.125	110
a_4	0.125	111

Form a tree:



→ No codeword is a prefix of another codeword.

→ When the symbol probabilities are all powers of 2, the average length of Huffman codes is equal to the lower bound, the entropy of the source.

For the previous example,

$$\text{Entropy } H(X) = \sum_i p(a_i) \log_2 \frac{1}{p(a_i)}$$

$$= 0.5 \log_2 \frac{1}{0.5} + 0.25 \log_2 \frac{1}{0.25} + 0.125 \log_2 \frac{1}{0.125} + 0.125 \log_2 \frac{1}{0.125}$$

$$= 0.5 \log_2 2 + 0.25 \log_2 4 + 0.125 \log_2 8 + 0.125 \log_2 8$$

$$= 0.5 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3$$

$$= \underline{\underline{1.75}}$$

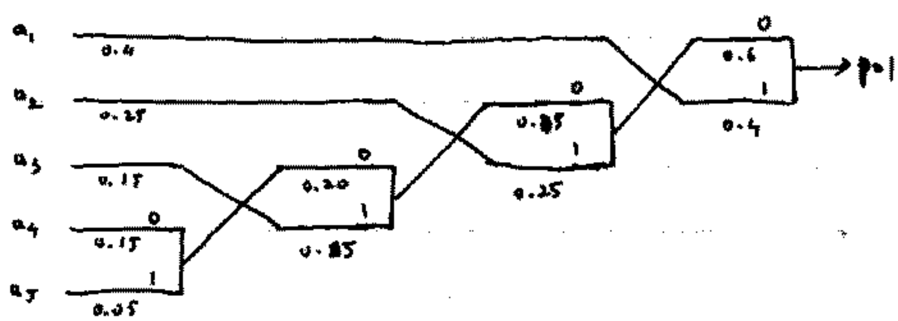
$$\text{Average codeword length} = 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = \underline{\underline{1.75}}$$

→ Huffman codes are uniquely decodable because no codeword is a prefix of another. For example,

$$a_3\ a_1\ a_2\ a_1\ a_2\ a_1\ a_1\ a_4 \cdots \longrightarrow \underbrace{110}_{a_3}\underbrace{0}_{a_1}\underbrace{10}_{a_2}\underbrace{0}_{a_1}\underbrace{10}_{a_2}\underbrace{0}_{a_1}\underbrace{0}_{a_1}\underbrace{111}_{a_4}\cdots$$

Example

Symbol	Probability	Codeword
a_1	0.4	1
a_2	0.25	01
a_3	0.15	001
a_4	0.15	0000
a_5	0.05	0001



Average codeword length $= 0.4\times 1 + 0.25\times 2 + 0.15\times 3 + 0.15\times 4 + 0.05\times 4 = 2.15$

$$\text{Entropy} = 0.4 \log_2 \frac{1}{0.4} + 0.25 \log_2 \frac{1}{0.25} + 0.15 \log_2 \frac{1}{0.15} + 0.15 \log_2 \frac{1}{0.15} + 0.05 \log_2 \frac{1}{0.05}$$

$$= 2.07$$

2.3. Arithmetic Coding

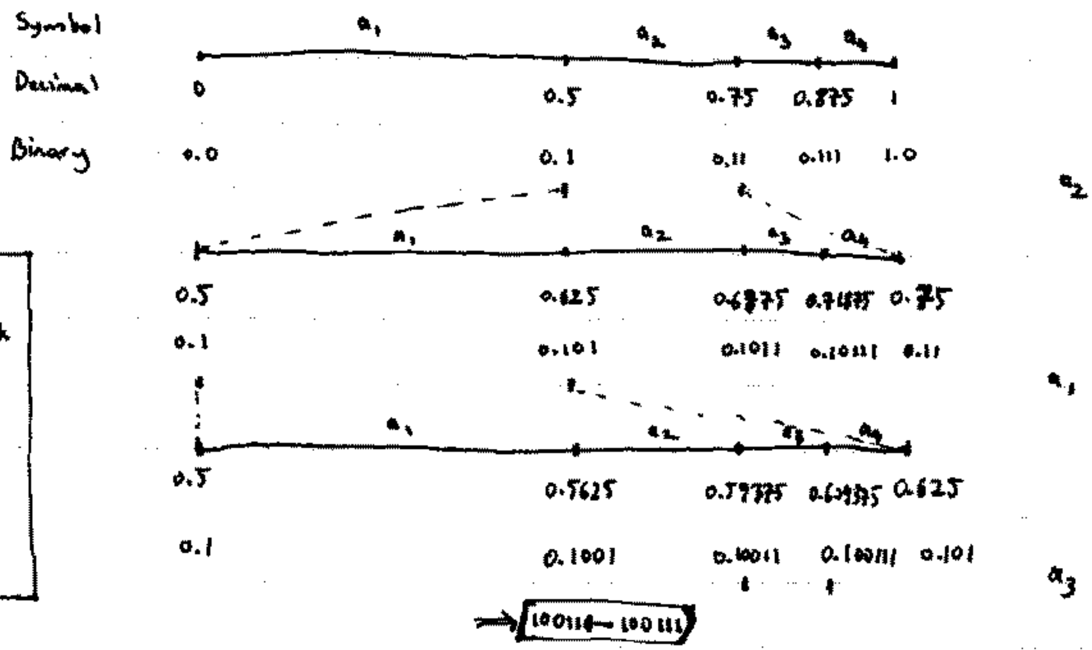
→ In arithmetic coding, one-to-one correspondence between source symbols and codewords does not exist. Instead, an entire sequence of source symbols is assigned a single arithmetic codeword.

→ The codeword defines an interval of real numbers between 0 and 1. As the number of symbols in the message increases, the interval used to represent it becomes smaller and the number of bits to represent that interval becomes larger.

→ Each symbol of the message reduces the size of the interval in accordance with its probability of occurrence.

→ Because the technique does not require, unlike the Huffman coding, that each source symbol translate into an integer ~~number of~~ length codeword, it can achieve better bitrates.

<u>Example</u>: Consider an alphabet with four symbols having probabilities 0.5, 0.25, 0.125, 0.125. ~~Name~~ these symbols as a_1, a_2, a_3, a_4.
What is the arithmetic code to represent the sequence $a_2 a_1 a_3 \ldots$



Note:
$\text{Decimal} = \sum_k b_k 2^k$

Example:
5 = 101
6 = 110
0.5 = 0.1
0.75 = 0.11

Received Bit		Interval	Symbol
1	0.1.....	$[\overset{0.1}{0.5}, \overset{1}{1})$	–
0	0.10....	$[\overset{0.10}{0.5}, \overset{0.11}{0.75})$	a_2
0	0.100....	$[\overset{0.100}{0.5}, \overset{0.101}{0.625})$	a_1
1	0.1001....	$[\overset{0.1001}{0.5625}, \overset{0.101}{0.625})$	–
1	0.10011....	$[\overset{0.10011}{0.59375}, \overset{0.101}{0.625})$	–
0	0.100110...	$[\overset{0.100110}{0.59375}, \overset{0.10011}{0.609375})$	a_3

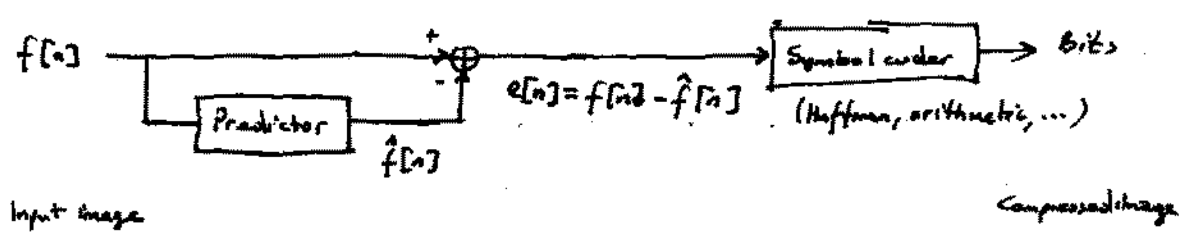
3. Lossless Compression Methods

3.1. Lossless Predictive Coding

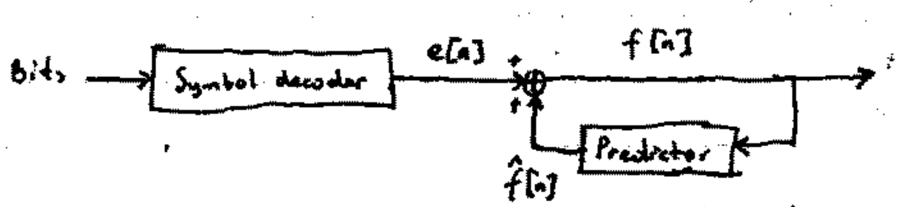
→ The first step is to form an integer-valued prediction of the next pixel intensity to be encoded based on a previously encoded neighboring pixels.

→ Than the difference between the actual intensity of the pixel and its prediction is entropy coded.

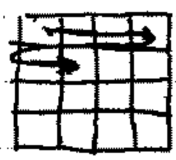
Encoder :



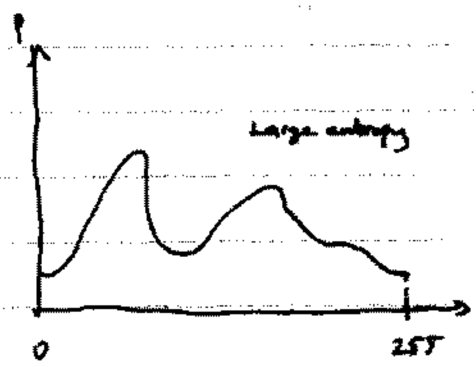
Decoder :



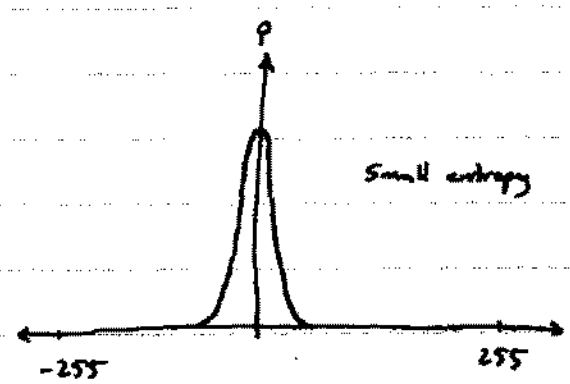
Example:



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



Histogram of original image



Histogram of prediction error

↳ Assign a unique codeword to every difference value in the range (-15, 16)

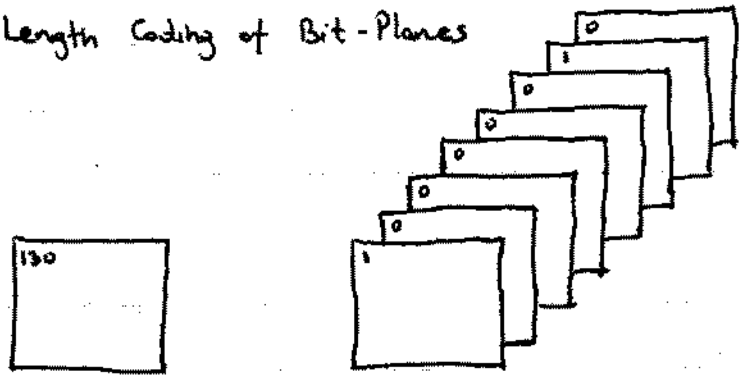
↳ Assign codewords to a shift up (SU) symbol and a shift down (SD) symbol to shift by 32

(For example,
100 = SU SU SU (4))

Example: $\hat{f}[n,m] = \text{round}\left\{\sum_{i=1}^{p} \alpha_i f[n,m-i]\right\}$

Example:

3.2. Run-Length Coding of Bit-Planes



8-bit gray level image   Bit planes

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

Note:

$\oplus$	0	1
0	0	1
1	1	0

Example:

	Binary	Gray
0	00	00
1	01	01
2	10	11
3	11	10

* A disadvantage of the above bit-plane representation is that small changes in gray level may cause edges in all bit planes.

For example:

127 = 01111111

128 = 10000000

* An alternative decomposition is <u>m-bit Gray code</u>.

exclusive or (XOR)

$$\left.\begin{aligned} g_i &= a_i \oplus a_{i+1}, \quad 0 \le i \le m-2 \\ g_{m-1} &= a_{m-1} \end{aligned}\right\} \Rightarrow$$ Successive codewords differ in only one bit position.

↳ Small changes in gray levels are less likely to affect all m bit planes.

For example:

127 = 11000000

128 = 01000000

* 1-D Run-Length Coding

→ Each row is represented as a sequence of lengths that describe successive runs of white (1) and black (0) pixels.

→ The resulting run-lengths ~~can be~~ Huffman coded.

Example:

1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	→ (2)(5)(4)(1)(4)
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	→ (0)(4)(8)(4)

~~*1-2-D Run-Length Coding~~

3.3. Lempel - Ziv Coding

→ Instead of assigning codewords to known source symbols or messages, LZ coding assigns fixed-length codewords to repeating source words in the input stream.

→ A dictionary is constructed from the input sequence

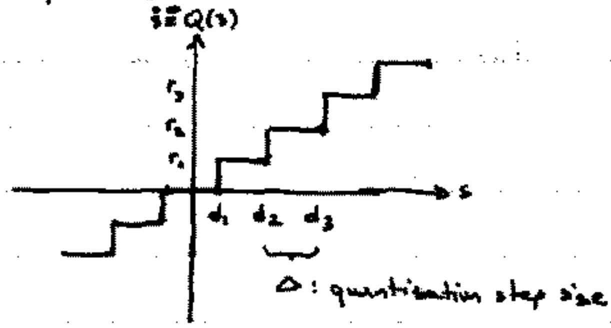
→ The dictionary does not need to be transmitted. It is reconstructed at the decoder.

Index	Dictionary	Input	Output	Codeword
0	0			
1	1			
2	01	0	0	000
3	11	1	1	001
4	10	1	1	001
5	00	0	0	000
6	011	0 1	2	010
7	100	1 0	4	100
8	010	0 1	2	010
9	0110	0 1 1	6	110
10	000	0 0	5	101
		0	0	000

4. Lossy Compression

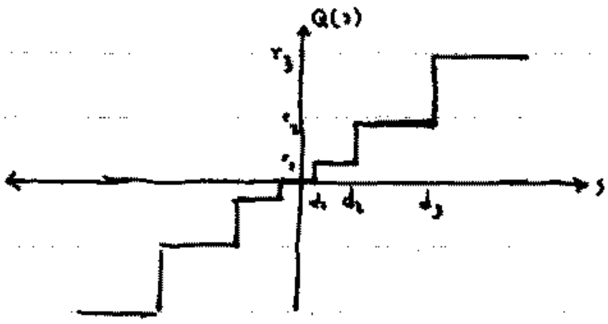
Lossy compression methods employ a quantization process. With quantization, continuous-valued samples are represented with a finite number of states.

* Uniform Quantization



Scalar quantization: Each sample is quantized independently.
Vector quantization: Continuous valued vectors are represented with a finite number of vectors.

* Nonuniform Quantization



* Lloyd-Max quantizer minimizes

$$E\left\{\sum_i (s-r_i)^2\right\} = \sum_i \int_{d_i}^{d_{i+1}} (s-r_i)^2\, p(s)\, ds$$

probability density function of input s

→ If s is uniformly distributed over an interval $[A, B]$,

$$p(s) = \begin{cases} \frac{1}{B-A} & A \le s \le B \\ 0 & \text{elsewhere} \end{cases}$$

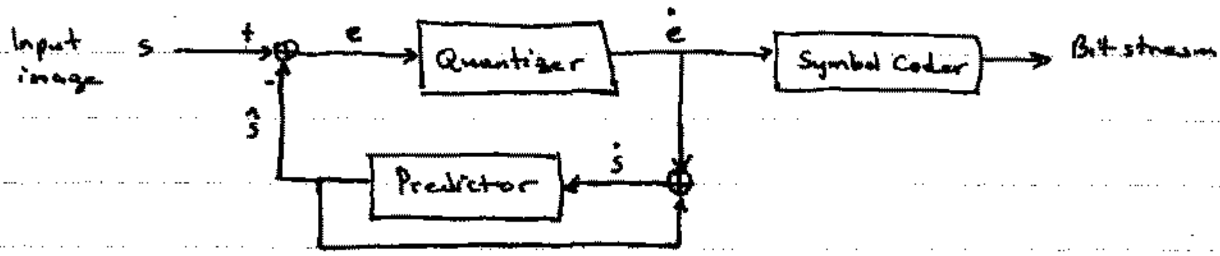
then Lloyd-Max quantizer becomes a uniform quantizer with

$$\left(\Delta = \frac{B-A}{\#\text{ of output levels}}\right)$$

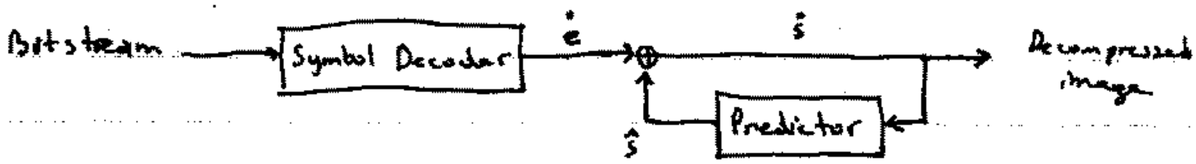
4.1. Differential Pulse Code Modulation (DPCM)

* DPCM predicts the intensity of the next pixel based on its neighbors.
* The difference between the actual and predicted intensity is then quantised and coded.

Encoder

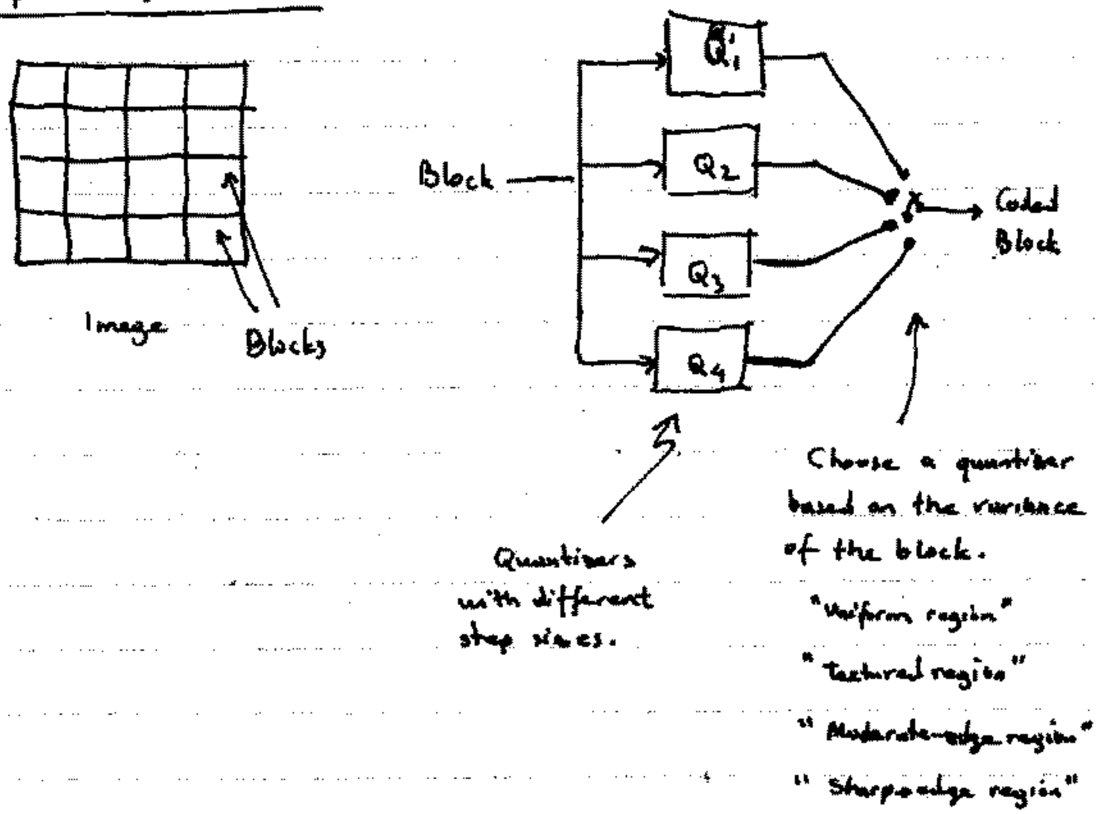


Decoder



* Note that the predictor at the encoder also uses $\dot{s}$ in the prediction instead of the actual values s. This prevents error build-up at the decoder. (To this effect, the encoder simulates the decoder in the prediction loop.)

* Adaptive Quantization :



Alternatively,
1) Calculate the variance of the prediction error for different quantizers.
2) Pick up the quantizer with minimum prediction error.

* Delta Modulation :

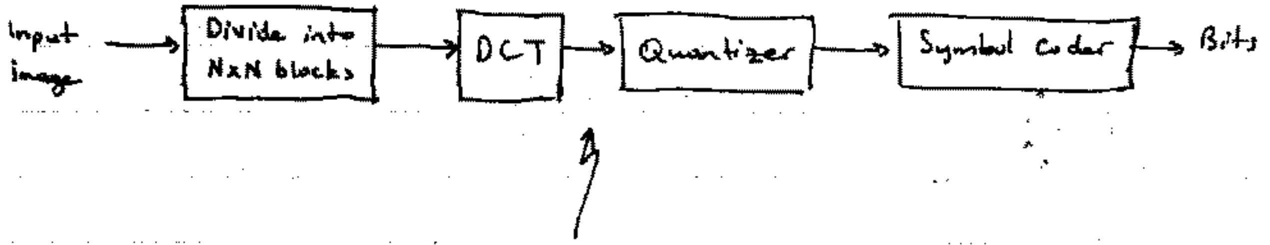
$$\hat{s}[n] = \alpha \, \dot{s}[n-1]$$

$$\dot{e}[n] = \begin{cases} \epsilon & \text{for } e[n] > 0 \\ -\epsilon & \text{for } e[n] \leq 0 \end{cases}$$

$\Rightarrow$ 1 bit/pixel coding

4.2. Transform Coding

Encoder



DCT coefficients having the highest energy are most finely quantized, those with the least energy are coarsely quantized.

Decoder

* Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), Hadamard Transform, Karhunen-Loeve Transform (KLT) have been used for image compression
* The transformation should produce uncorrelated coefficients (1), and pack the maximum amount of energy into the smallest # of coefficients (2). (1) justifies the use of scalar quantization. (2) is desirable because we would like to discard as many coefficients as possible without seriously affecting image quality.
* The transformation that satisfies both these properties is the KLT.

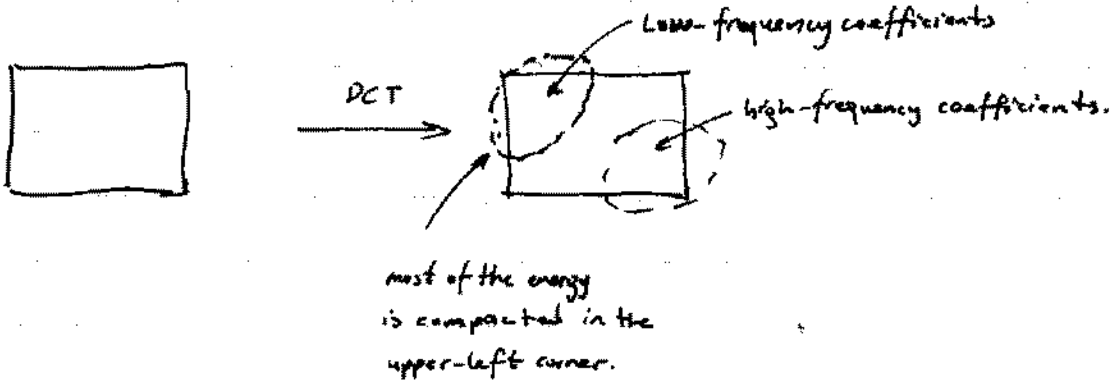
  To calculate KLT:

  1) Calculate the covariance matrix $R \triangleq E[x x^T]$, where x is the vector obtained by ordering the pixel values within the block.
  2) Find the eigenvectors of R, and construct the eigenvector matrix $\Gamma = [v_1 \; v_2 \; \dots \; v_N]$. $(R v_i = \lambda_i v_i) \Rightarrow R\Gamma = \Gamma\Lambda \Rightarrow R = \Gamma\Lambda\Gamma^T$
  3) The KLT of x is equal to $\Gamma^T x$

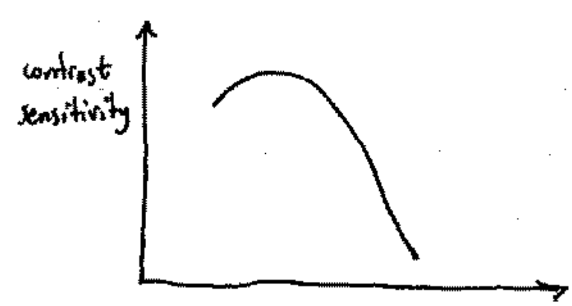
*DCT

* KLT is not used in practice because
    1) It has to be recomputed and transmitted for every image
    2) There are no fast computational algorithms for its implementation.

* DCT has been found to be the most effective ~~d~~transformation with a performance close to that of the KLT.

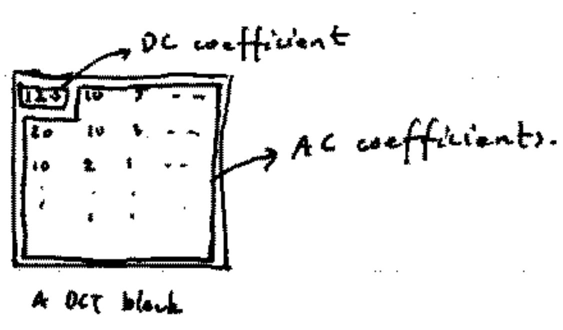


* Human eye is less sensitive to high spatial variations.



→ This is exploited in the quantization of DCT coefficients.

→ DCT coefficients corresponding to less sensitive frequency components are more coarsely quantized.

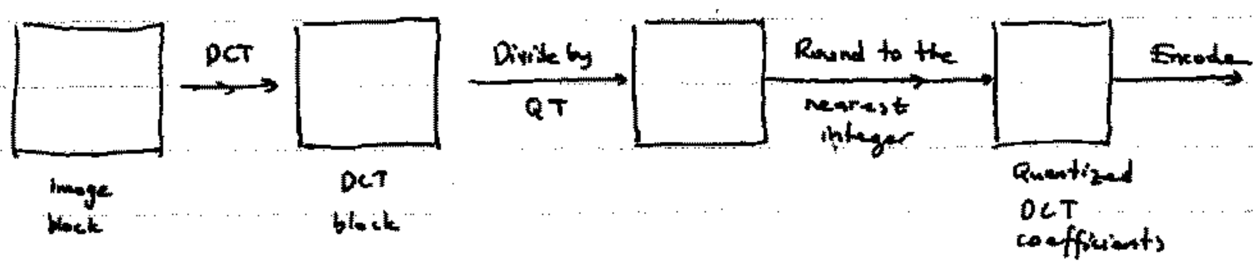


A DCT block

Quantization Table for the Luminance Channel in JPEG Standard

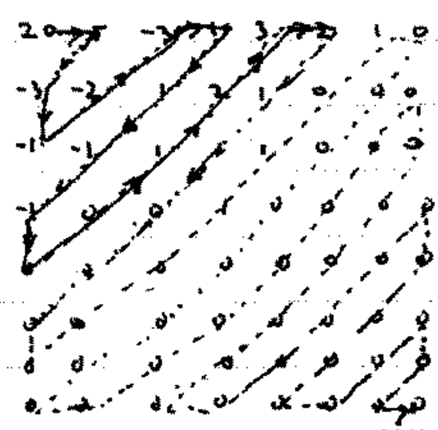
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

QT



* To encode, zig-zag scanning is used to obtain a coefficient sequence, which is then Huffman coded.

<u>Example</u>:



[20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, ..., EOB]

* The DC coefficient is DPCM coded.
* The AC coefficients are mapped into run / level pairs.

\# of zeros before the coefficient.

For the previous example:

$(0,5), (0,-3), (0,-1), (0,-2), (0,-3), (0,1), (0,1), (0,-1), (0,-1), (2,1), \ldots$

↳ These pairs are then Huffman coded using the predetermined tables. (Check out the textbook for the tables!)