UNIT V
IMAGE DATA COMPRESSION

Introduction

In recent years, there have been significant advancements in algorithms and architectures for the processing of image, video, and audio signals. These advancements have proceeded along several directions. On the algorithmic front, new techniques have led to the development of robust methods to reduce the size of the image, video, or audio data. Such methods are extremely vital in many applications that manipulate and store digital data. Informally, we refer to the process of size reduction as a compression process. We will define this process in a more formal way later.

On the architecture front, it is now feasible to put sophisticated compression processes on a relatively low-cost single chip; this has spurred a great deal of activity in developing multimedia systems for the large consumer market. One of the exciting prospects of such advancements is that multimedia information comprising image, video, and audio has the potential to become just another data type. This usually implies that multimedia information will be digitally encoded so that it can be manipulated, stored, and transmitted along with other digital data types. For such data usage to be pervasive, it is essential that the data encoding is standard across different platforms and applications. This will foster widespread development of applications and will also promote interoperability among systems from different vendors. Furthermore, standardization can lead to the development of cost effective implementations, which in turn will promote the widespread use of multimedia information. This is the primary motivation behind the emergence of image and video compression standards.

Background

Compression is a process intended to yield a compact digital representation of a signal. In the literature, the terms *source coding, data compression, bandwidth compression,* and *signal compression* are all used to refer to the process of compression. In the cases where the signal is defined as an image, a video stream, or an audio signal, the generic problem of compression is to minimize the bit rate of their

digital representation. There are many applications that benefit when image, video, and audio signals are available in compressed form. **Without compression, most of these applications would not be feasible!**

Example 1: Let us consider **facsimile image transmission**. In most facsimile machines, the document is scanned and digitised. Typically, an 8.5x11 inches page is scanned at 200 dpi; thus, resulting in 3.74 Mbits. Transmitting this data over a low-cost 14.4 kbits/s modem would require 5.62 minutes. With compression, the transmission time can be reduced to 17 seconds. This results in substantial savings in transmission costs.

Example 2: Let us consider a video-based CD-ROM application. **Full-motion video**, at 30 fps and a 720 x 480 resolution, generates data at 20.736 Mbytes/s. At this rate, only 31 seconds of video can be stored on a 650 MByte CD-ROM. Compression technology can increase the storage capacity to 74 minutes, for VHS-grade video quality.

Image, video, and audio signals are amenable to compression due to the factors below.

• There is considerable statistical redundancy in the signal.

1. Within a single image or a single video frame, there exists significant correlation among neighbor samples. This correlation is referred to as *spatial correlation.*

2. For data acquired from multiple sensors (such as satellite images), there exists significant correlation amongst samples from these sensors. This correlation is referred to as *spectral correlation.*

3. For temporal data (such as video), there is significant correlation amongst samples in different segments of time. This is referred to as *temporal correlation.*

• There is considerable information in the signal that is irrelevant from a perceptual point of view.

• Some data tends to have high-level features that are redundant across space and time; that is, the data is of a fractal nature.

For a given application, compression schemes may exploit any one or all of the above factors to achieve the desired compression data rate.

There are many applications that benefit from data compression technology. Table 1.1 lists a representative set of such applications for image, video, and audio data, as well as typical data rates of the corresponding compressed bit streams. Typical data rates for the uncompressed bit streams are also shown.

Application	Data Rate	
	Uncompressed	Compressed
Voice 8 ksamples/s, 8 bits/sample	64 kbps	2-4 kbps
Slow motion video (10fps) framesize 176x120, 8bits/pixel	5.07 Mbps	8-16 kbps
Audio conference 8 ksamples/s, 8 bits/sample	64 kbps	16-64 kbps
Video conference (15fps) framesize 352x240, 8bits/pixel	30.41 Mbps	64-768 kbps
Digital audio 44.1 ksamples/s, 16 bits/sample	1.5 Mbps	1.28-1.5 Mbps
Video file transfer (15fps) framesize 352x240, 8bits/pixel	30.41 Mbps	384 kbps
Digital video on CD-ROM (30fps) framesize 352x240, 8bits/pixel	60.83 Mbps	1.5-4 Mbps
Broadcast video (30fps) framesize 720x480, 8bits/pixel	248.83 Mbps	3-8 Mbps

In the following figure, a systems view of the compression process is depicted.

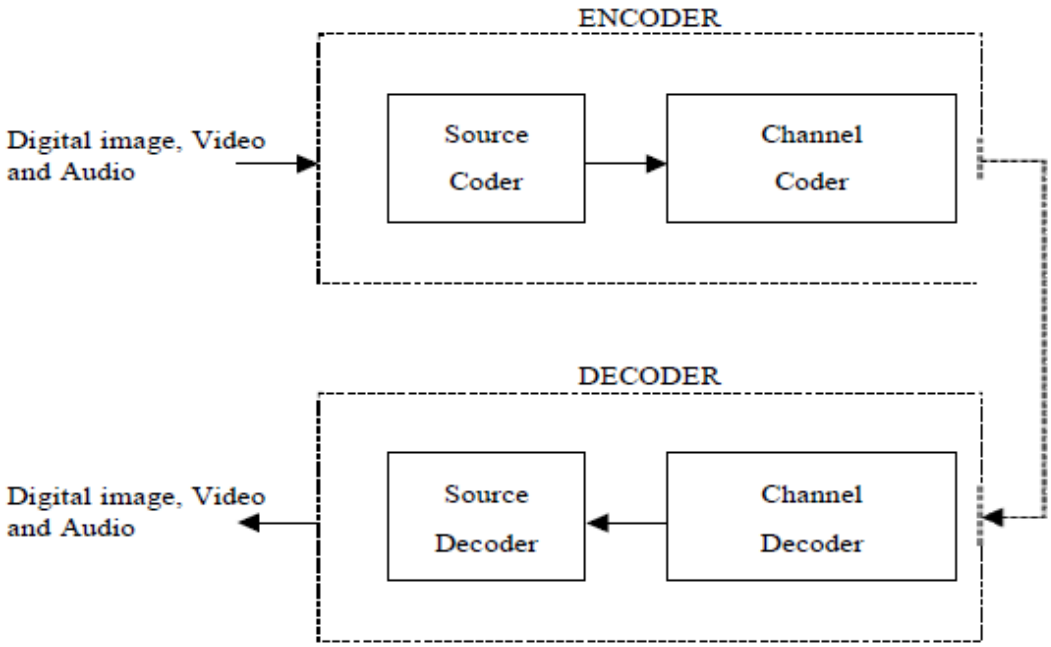


Figure Generic compression system

The core of the encoder is the source coder. The source coder performs the compression process by reducing the input data rate to a level that can be supported by the storage or transmission medium. The bit rate output of the encoder is measured in bits per sample or bits per second. For image or video data, a pixel is the basic element; thus, bits per sample are also referred to as bits per pixel or bits per pel. In the literature, the term *compression ratio,* denoted as *cr* , is also used instead of *bit rate* to characterize the capability of the compression system. An intuitive definition of *cr* is

$$c_r = \frac{source\ coder\ input\ size}{source\ coder\ output\ size}$$

This definition is somewhat ambiguous and depends on the data type and the specific compression method that is employed. For a still-image, size could refer to the bits needed to represent the entire image. For video, size could refer to the bits needed to represent one frame of video. Many compression methods for video do not process each frame of video, hence, a more commonly used notion for size is the bits needed to represent one second of video.

In a practical system, the source coder is usually followed by a second level of coding: the channel coder. The channel coder translates the compressed bit stream into a signal suitable for either storage or transmission. In most systems, source coding and channel coding are distinct processes. In recent years, methods to perform combined source and channel coding have also been developed. Note that, in order to reconstruct the image, video, or audio signal, one needs to reverse the processes of channel coding and source coding. This is usually performed at the decoder.

From a system design viewpoint, one can restate the compression problem as a bit rate minimisation problem, where several constraints may have to be met, including the following:

- **Specified level of signal quality**. This constraint is usually applied at the decoder.
- **Implementation complexity**. This constraint is often applied at the decoder, and in some instances at both the encoder and the decoder.
- **Communication delay**. This constraint refers to the end to end delay, and is measured from the start of encoding a sample to the complete decoding of that sample.

Note that, these constraints have different importance in different applications. For example, in a two-way teleconferencing system, the communication delay might be the major constraint, whereas, in a television broadcasting system, signal quality and decoder complexity might be the main constraints.

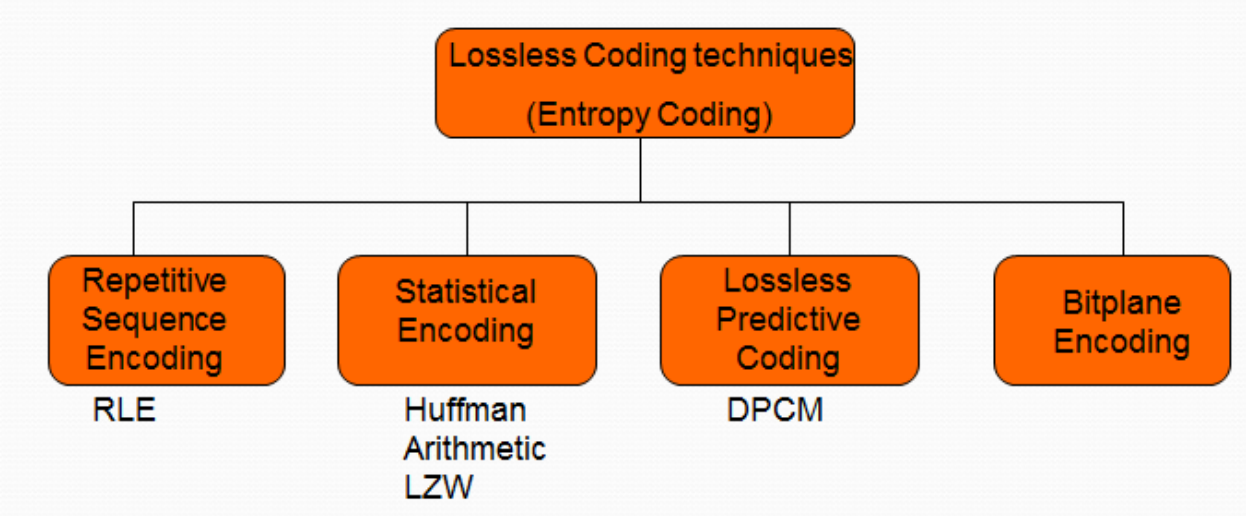
Types of image compression

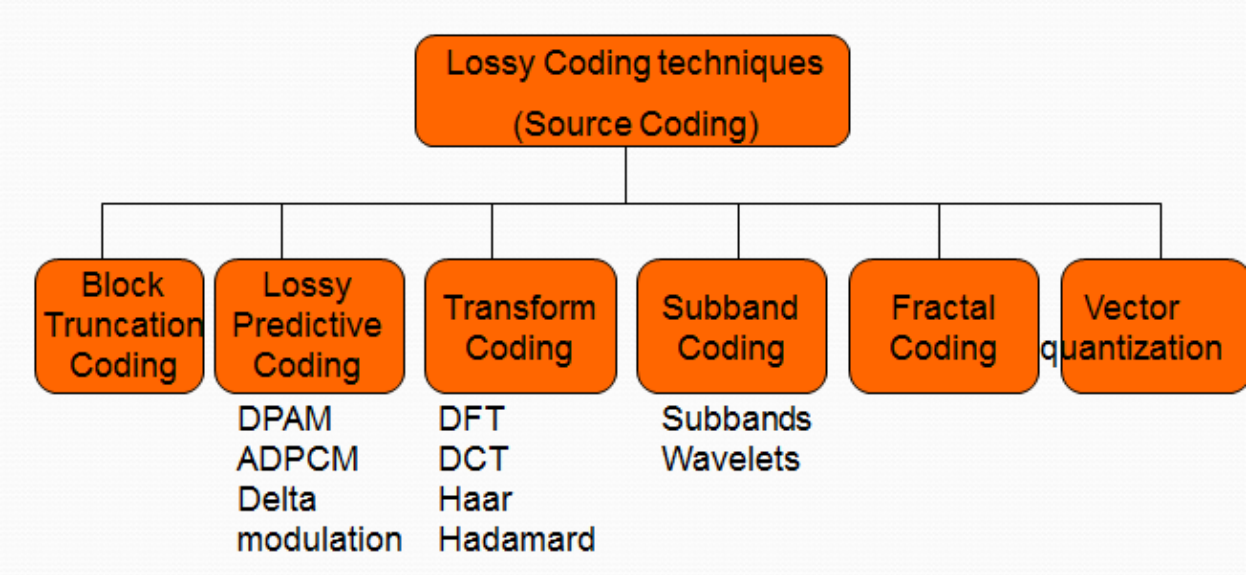
Image compression can be:

- **Reversible** (loss less), with no loss of information.
    - A new image is identical to the original image (after decompression).
    - Original data exactly recovered from compressed data
    - Lower compression ratio
    - Reversibility is necessary in most image analysis applications.

- The compression ratio is typically 2 to 10 times.
- Examples are Huffman coding and run-length coding.

- **Non reversible** (lossy), with loss of some information.
  - Lossy compression is often used in image communication, video,WWW, etc.
  - It is usually important that the image visually is still *nice*.
  - The compression ratio is typically 10 to 30 times.
  - Loss of information
  - Perceptual loss of information reduced (controlled)
  - Higher compression ratio





Lossless versus lossy compression

Lossless compression

In many applications, the decoder has to reconstruct without any loss the original data. For a lossless compression process, the reconstructed data and the original data must be identical in value for each and every data sample. This is also referred to as a reversible process. In lossless compression, for a specific application, the choice of a compression method involves a trade-off along the three dimensions depicted in Figure 1.2; that is, coding efficiency, coding complexity, and coding delay.

Coding Efficiency

This is usually measured in bits per sample or bits per second (bps). Coding efficiency is usually limited by the information content or *entropy* of the source. In intuitive terms, the entropy of a source X provides a measure for the "randomness" of X. From a compression theory point of view, sources with large entropy are more difficult to compress (for example, random noise is very hard to compress).

Coding Complexity

The complexity of a compression process is analogous to the computational effort needed to implement the encoder and decoder functions. The computational effort is usually measured in terms of memory requirements and number of arithmetic operations. The operations count is characterised by the term millions of operations per second and is often referred to as MOPS. Here, by operation, we imply a basic arithmetic operation that is supported by the computational engine. In the compression literature, the term MIPS (millions of instructions per second) is sometimes used. This is specific to a computational engine's architecture; thus, in this text we refer to coding complexity in terms of MOPS. In some applications, such as portable devices, coding complexity may be characterised by the power requirements of a hardware implementation.

Coding Delay

A complex compression process often leads to increased coding delays at the encoder and the decoder. Coding delays can be alleviated by increasing the processing power of the computational engine; however, this may be impractical in environments where there is a power constraint or when the underlying computational engine cannot be improved. Furthermore, in many applications, coding delays have to be constrained; for example, in interactive communications.

Lossy compression

The majority of the applications in image or video data processing do not require that the reconstructed data and the original data are identical in value. Thus, some amount of loss is permitted in the reconstructed data. A compression process that results in an imperfect reconstruction is referred to as a lossy compression process. This compression process is irreversible. In practice, most irreversible compression processes degrade rapidly the signal quality when they are repeatedly applied on

previously decompressed data. The choice of a specific lossy compression method involves trade-offs along the four dimensions shown in Figure 1.3. Due to the additional degree of freedom, namely, in the signal quality, a lossy compression process can yield higher compression ratios than a lossless compression scheme.

Signal Quality

This term is often used to characterize the signal at the output of the decoder. There is no universally accepted measure for signal quality. One measure that is often cited is the signal to noise ratio *SNR,* which can be expressed as

$$SNR = 10\log_{10} \frac{encoderinput\ signal\ energy}{noisesignalenergy}$$

The noise signal energy is defined as the energy measured for a hypothetical signal that is the difference between the encoder input signal and the decoder output signal. Note that, *SNR* as defined here is given in decibels (dB). In the case of images or video, *PSNR* (peak signal-to noise ratio) is used instead of *SNR.* The calculations are essentially the same as in the case of *SNR* , however, in the numerator, instead of using the encoder input signal one uses a hypothetical signal with a signal strength of 255 (the maximum decimal value of an unsigned 8-bit number, such as in a pixel).

High *SNR* or *PSNR* values do not always correspond to signals with perceptually high quality.

Another measure of signal quality is the mean opinion score, where the performance of a compression process is characterized by the subjective quality of the decoded signal.

METHODS FOR LOSSLESS COMPRESSION

Lossless compression refers to compression methods for which the original uncompressed data set can be recovered exactly from the compressed stream. The need for lossless compression arises from the fact that many applications, such as the compression of digitized medical data, require that no loss be introduced from the compression method. Bitonal image transmission via a facsimile device also imposes such requirements. In recent years, several compression standards have been

developed for the lossless compression of such images. We discuss these standards later. In general, even when lossy compression is allowed, the overall compression scheme may be a combination of a lossy compression process followed by a lossless compression process. Various image, video, and audio compression standards follow this model, and several of the lossless compression schemes used in these standards are described in this section. The general model of a lossless compression scheme is as depicted in the following figure.

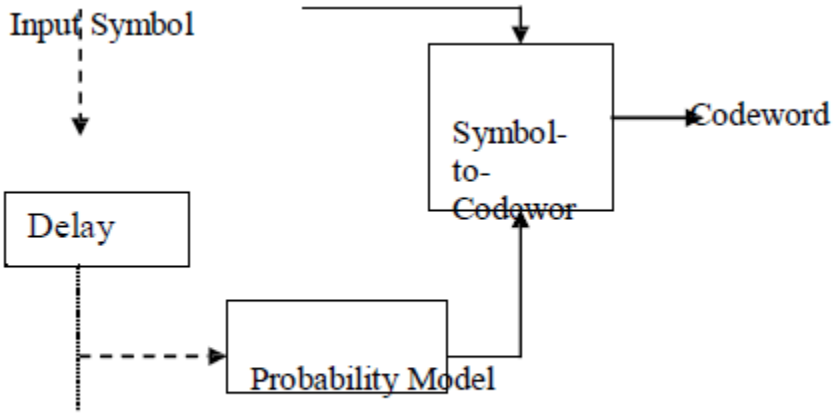


Figure 1.1: A generic model for lossless compression

Basic Concepts of image compression

The term *data compression* refers to the process of reducing the amount of data required to represent a given quantity of information. A clear distinction must be made between *data* and *information*. They are not synonymous. In fact, data are the means by which information is conveyed. Various amounts of data may be used to represent the same amount of information. Such might be the case, for example, if a long-winded individual and someone who is short and to the point were to relate the same story. Here, the information of interest is the story; words are the data used to relate the information. If the two individuals use a different number of words to tell the same basic story, two different versions of the story are created, and at least one includes nonessential data. That is, it contains data (or words) that either provide no relevant information or simply restate that which is already known. It is thus said to contain *data redundancy*.

Data Redundancy

Data redundancy is a central issue in digital image compression. It is not an abstract concept but a mathematically quantifiable entity. If n_1 and n_2 denote the number of information-carrying units in two data sets that represent the same information, the *relative data redundancy* R_D of the first data set (the one characterized by n_1) can be defined as

$$R_D = 1 - \frac{1}{C_R} \tag{8.1-1}$$

where C_R, commonly called the *compression ratio*, is

$$C_R = \frac{n_1}{n_2}. \tag{8.1-2}$$

For the case $n_2 = n_1$, $C_R = 1$ and $R_D = 0$, indicating that (relative to the second data set) the first representation of the information contains no redundant data. When $n_2 \ll n_1$, $C_R \to \infty$ and $R_D \to 1$, implying significant compression and highly redundant data. Finally, when $n_2 \gg n_1$, $C_R \to 0$ and $R_D \to -\infty$, indicating that the second data set contains much more data than the original representation. This, of course, is the normally undesirable case of data expansion. In general, C_R and R_D lie in the open intervals $(0, \infty)$ and $(-\infty, 1)$, respectively. A practical compression ratio, such as 10 (or 10:1), means that the first data set has 10 information carrying units (say, bits) for every 1 unit in the second or compressed data set. The corresponding redundancy of 0.9 implies that 90% of the data in the first data set is redundant.

- Data that provide no relevant information=*redundant data* or *redundancy*.
- Image compression techniques can be designed by reducing or eliminating the Data Redundancy
- Image coding or compression has a goal to reduce the amount of data by reducing the amount of redundancy.

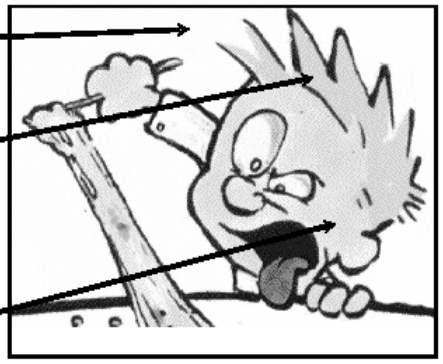
Three basic data redundancies

- Coding Redundancy
- Interpixel Redundancy
  - Psychovisual Redundancy

CR: some graylevels are more common than others

IR: the same graylevel covers large areas

PVR: the eye can only resolve 32 graylevels locally



- Coding
  - if grey levels of image are coded in such away that uses more symbols than is necessary
- Inter-pixel
  - can guess the value of any pixel from its neighbours
- Psyco-visual
  - some information is less important than other info in normal visual processing

- **Coding redundancy**
  - Fewer bits to represent frequent symbols
  - Huffman coding : Lossless
  - Occurs when the data used to represent the image is not utilized in an optimal manner

- **Interpixel redundancy**
  - Neighboring pixels have similar values
  - Occurs because adjacent pixels tend to be highly correlated, in most images the brightness levels do not change rapidly, but change gradually
  - Predictive coding : Lossless
  - Correlation between pixels is not used in coding
    - Correlation due to geometry and structure
  - Value of any pixel can be predicted from the value of the neighbours

        - Information carried by one pixel is small
        - Take 2D visual information
    - transformed → NONVISUAL format
        - This is called a MAPPING
        - A REVERSIBLE MAPPING allows original to be reconstructed after MAPPING
        - Use run-length coding

- **Psychovisual redundancy**
    - Some information is more important to the human visual system than other types of information
        - Quantization : Lossy
        - Remove information that human visual system cannot perceive
        - Removal of high frequency data : Lossy
        - Due to properties of human eye
        - Eye does not respond with equal sensitivity to all visual information (e.g. RGB)
        - Certain information has less relative importance
        - If eliminated, quality of image is relatively unaffected
        - This is because HVS only sensitive to 64 levels
        - Use fidelity criteria to assess loss of information

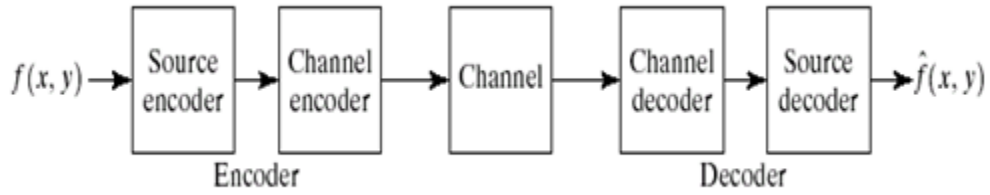
Coding redundancy

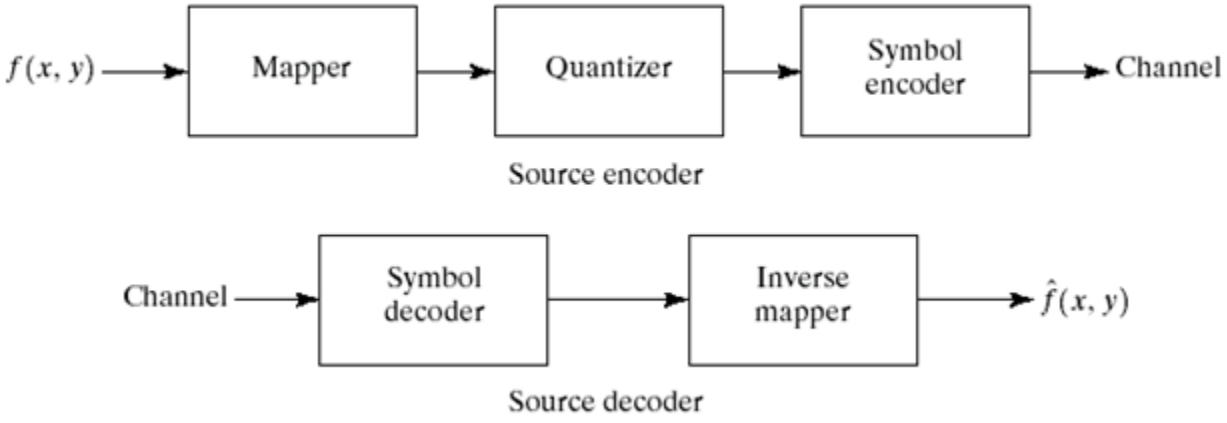
If the gray level of an image is coded in a way that uses more code words than necessary to represent each gray level, then the resulting image is said to contain coding redundancy.

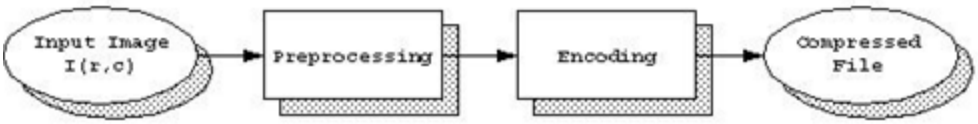
Interpixel redundancy

The value of any given pixel can be predicted from the values of its neighbors. The information carried by is small. Therefore the visual contribution of a single pixel to an image is redundant. Otherwise called as spatial redundant geometric redundant or interpixel redundant. Eg: Run length coding

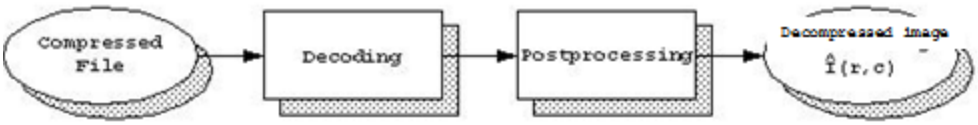
General compression model







a) Compression

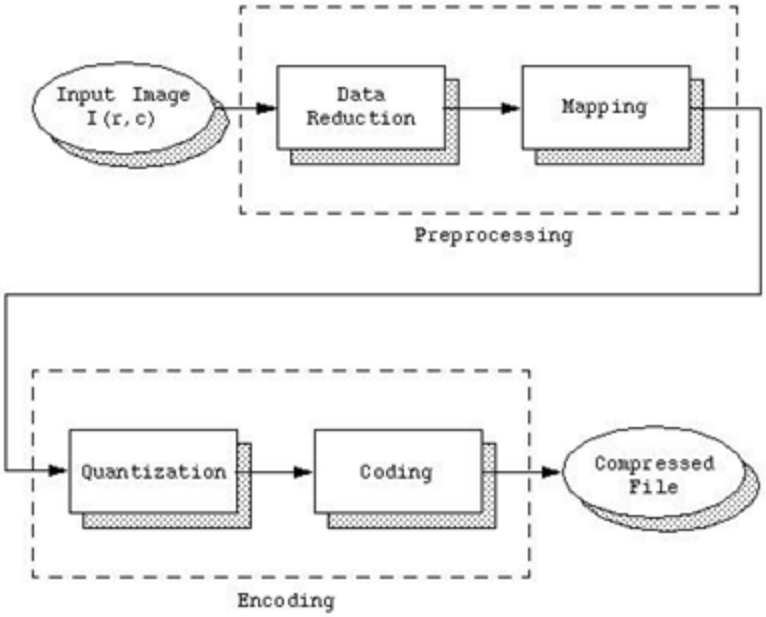


b) Decompression

- Before encoding, preprocessing is performed to prepare the image for the encoding process, and consists of any number of operations that are application specific
- After the compressed file has been decoded, postprocessing can be performed to eliminate some of the potentially undesirable artifacts brought about by the compression process
- The compressor can be broken into following stages:

1. *Data reduction*: Image data can be reduced by gray level and/or spatial quantization, or can undergo any desired image improvement (for example, noise removal) process
2. *Mapping*: Involves mapping the original image data into another mathematical space where it is easier to compress the data
3. *Quantization*: Involves taking potentially continuous data from the mapping stage and putting it in discrete form
4. *Coding*: Involves mapping the discrete data from the quantizer onto a code in an optimal manner

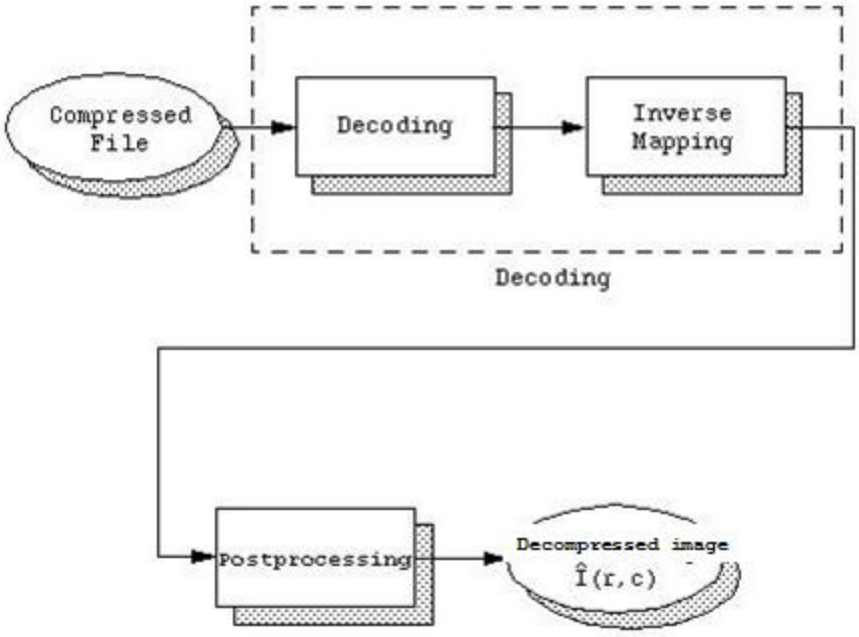
- A compression algorithm may consist of all the stages, or it may consist of only one or two of the stages



The decompressor can be broken down into following stages:

1. *Decoding*: Takes the compressed file and reverses the original coding by mapping the codes to the original, quantized values
2. *Inverse mapping*: Involves reversing the original mapping process
3. *Postprocessing*: Involves enhancing the look of the final image

- This may be done to reverse any preprocessing, for example, enlarging an image that was shrunk in the data reduction process
- In other cases the postprocessing may be used to simply enhance the image to ameliorate any artifacts from the compression process itself



- The mapping process is important because image data tends to be highly correlated
- Specifically, if the value of one pixel is known, it is highly likely that the adjacent pixel value is similar
- By finding a mapping equation that decorrelates the data this type of data redundancy can be removed

Huffmann Coding

- The Huffman code, developed by D. Huffman in 1952, is a *minimum length code*
- This means that given the statistical distribution of the gray levels (the histogram), the Huffman algorithm will generate a code that is as close as possible to the *minimum bound, the entropy*

- The method results in an *unequal* (or *variable*) *length code*, where the size of the code words can vary
- For complex images, Huffman coding alone will typically reduce the file by 10% to 50% (1.1:1 to 1.5:1), but this ratio can be improved to 2:1 or 3:1 by preprocessing for irrelevant information removal

The Huffman algorithm can be described in five steps:

1. Find the gray level probabilities for the image by finding the histogram
2. Order the input probabilities (histogram magnitudes) from smallest to largest
3. Combine the smallest two by addition
4. GOTO step 2, until only two probabilities are left
5. By working backward along the tree, generate code by alternating assignment of 0 and 1

The most popular technique for removing coding redundancy; yields the smallest possible number of code symbols per source symbol

Huffman Coding Algorithm

- Arrange the symbol probabilities p_i in a decreasing order; consider them (p_i) as leaf nodes of a tree
- While there is more than one node:
  - *merge the two nodes with smallest probability to form a new node whose probability is the sum of the two merged nodes*
  - *Arrange the combined node according to its probability in the tree*
  - *Repeat until only two nodes are left*
- Starting from the top, arbitrarily assign 1 and 0 to each pair of branches merging into a node
- Continue sequentially from the root node to the leaf node where the symbol is located to complete the coding

Coding and decoding are done by simple look-up tables

Example:

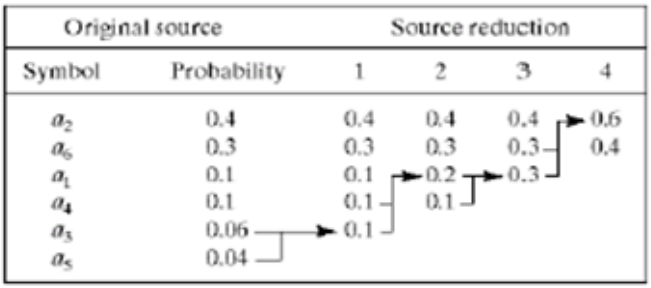
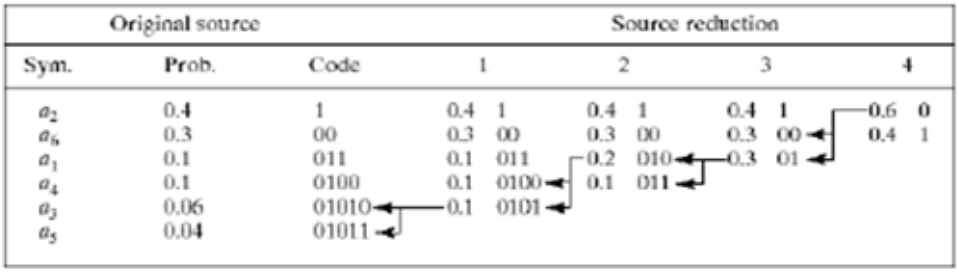


FIGURE Huffman source reductions.

FIGURE Huffman code assignment procedure.



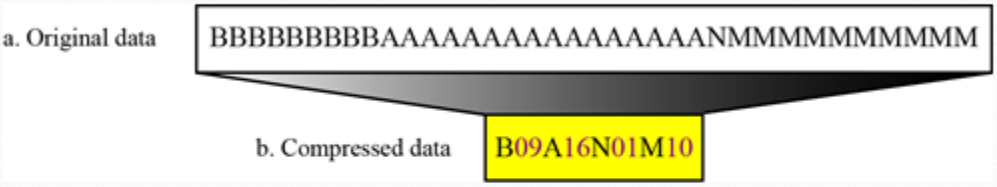
•The Huffman code results in an unambiguous code, i.e. no code can be created by combining other codes.

•The code is reversible without loss.

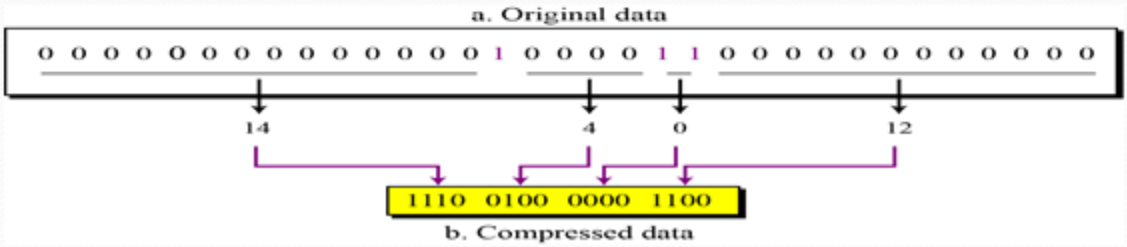
•The table for the translation of the code has to be stored together with the coded image.

Run Length Coding

- Simplest method of compression.
- How: replace consecutive repeating occurrences of a symbol by 1 occurrence of the symbol itself, then followed by the number of occurrences.



- The method can be more efficient if the data uses only 2 symbols (0s and 1s) in bit patterns and 1 symbol is more frequent than another.



- Run-length coding (RLC) works by counting adjacent pixels with the same gray level value called the *run-length,* which is then encoded and stored

- RLC works best for binary, two-valued, images

- RLC can also work with complex images that have been preprocessed by thresholding to reduce the number of gray levels to two

- RLC can be implemented in various ways, but the first step is to define the required parameters
- Horizontal RLC (counting along the rows) or vertical RLC (counting along the columns) can be used
    - In basic horizontal RLC, the number of bits used for the encoding depends on the number of pixels in a row
    - If the row has 2^n pixels, then the required number of bits is n, so that a run that is the length of the entire row can be encoded

- Every code word is made up of a pair (g, l) where g is the gray level, and l is the number of pixels with that gray level (length, or "run").
- E.g.,

  56 56 56 82 82 82 83 80
  56 56 56 56 56 80 80 80

  creates the run-length code (56, 3)(82, 3)(83, 1)(80, 4)(56, 5).
- The code is calculated row by row.

- Very efficient coding for binary data.
- Important to know position, and the image dimensions must be stored with the coded image.
- Used in most fax machines.

Example

0	0	0	0	0	0	0	0
0	0	1	1	2	3	3	3
0	1	1	3	3	3	4	4
0	1	3	3	5	5	4	4
0	2	3	3	5	5	5	4
0	0	2	3	3	4	6	6
0	0	0	2	2	3	4	4
0	0	0	0	0	0	0	0

gray level	0	1	2	3	4	5	6
#pixels	26	5	5	13	8	5	2

row #	**run-length code (gl,rl)**
0	(0,8)
1	(0,2), (1,2), (2,1), (3,3)
2	(0,1), (1,2), (3,3), (4,2)
3	(0,1), (1,1), (3,2), (5,2), (4,2)
...	...
7	(0,8)

row #	binary code
0	000 111
1	000 001 001 001 010 000 011 010
2	000 000 001 001 011 010 100 001
...	...
7	000 111

Compression Achieved

Original image requires 3 bits per pixel (in total - 8x8x3=192 bits).

Compressed image has 29 runs and needs 3+3=6 bits per

run (in total - 174 bits or 2.72 bits per pixel).

Image fidelity criteria

Quantify the nature and extent of information loss.

Objective fidelity criteria:

Level of information loss can be expressed as a function of the original (input) and compressed-decompressed (output) image.

Given an $M \times N$ image $f(x,y)$ (original image), its compressed-then-decompressed image: $\hat{f}(x,y)$, then the error between corresponding values are given as:

$$e(x,y) = \hat{f}(x,y) - f(x,y)$$

Total error is given by:

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]$$

Normally the objective fidelity criterion parameters are as follows:

e_{rms} (root-mean-square error):

$$e_{rms} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}$$

SNR_{ms} (mean-square signal-to-noise ratio):

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}.$$

Subjective fidelity criteria:

Since most decompressed images ultimately are viewed by human beings, measuring image quality by the subjective evaluations of a human observer often is more appropriate.

Method: evaluations are made using an absolute rating scale, by means of side-by-side comparisons of $f(x,y)$ and $\hat{f}(x,y)$.

TABLE 8.3 Rating scale of the Television Allocations Study Organization. (Frendendall and Behrend.)

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

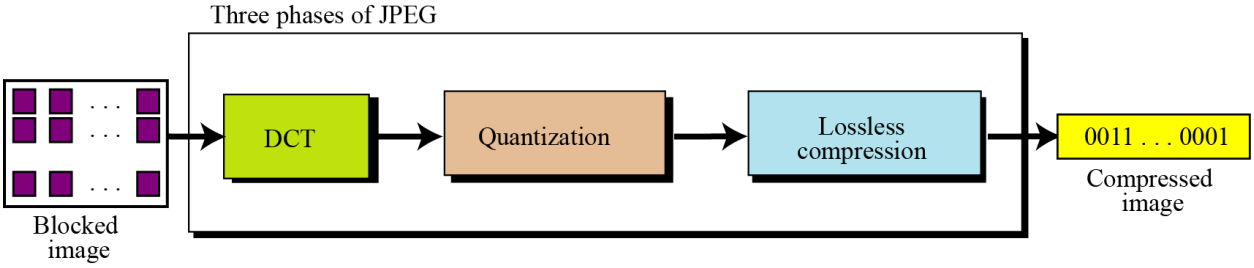
IMAGE COMPRESSION STANDARDS

Standard	Possible Application Areas
CCITT	Facsimile, Document
CCITT	Facsimile, Document
JPEG	Photographic Imaging
JBIG	Facsimile, Document
CCITT	Teleconferencing
MPEG-1	Video, Digital Storage
MPEG-2	Video, HDTV, DSM
MPEG-4	Audio-visual Communications

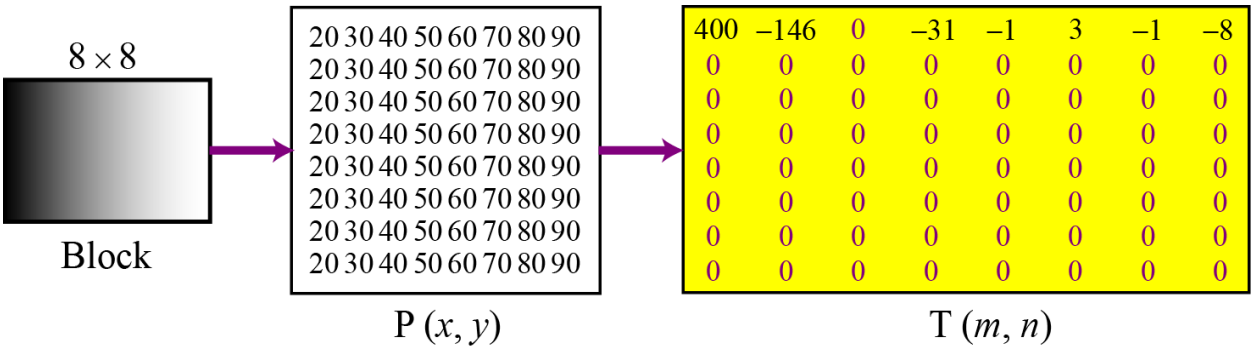
JPEG Encoding

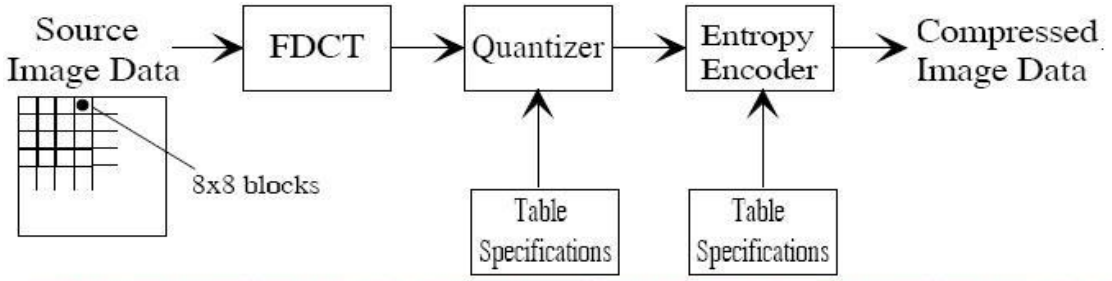
- Used to compress pictures and graphics.
- In JPEG, a grayscale picture is divided into 8x8 pixel blocks to decrease the number of calculations.

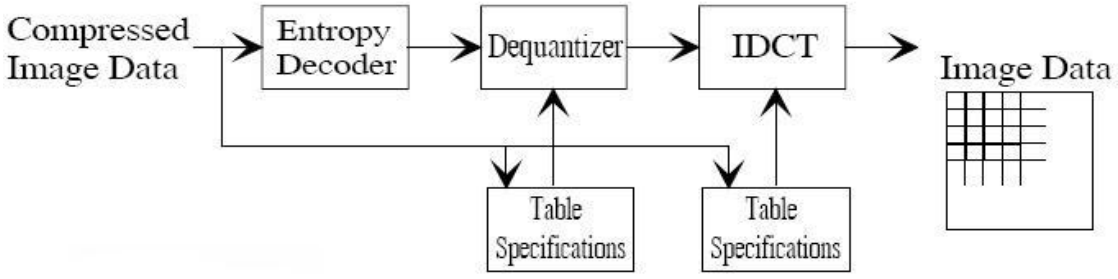
Basic idea: Change the picture into a linear (vector) sets of numbers that reveals the redundancies. The redundancies is then removed by one of lossless compression methods.



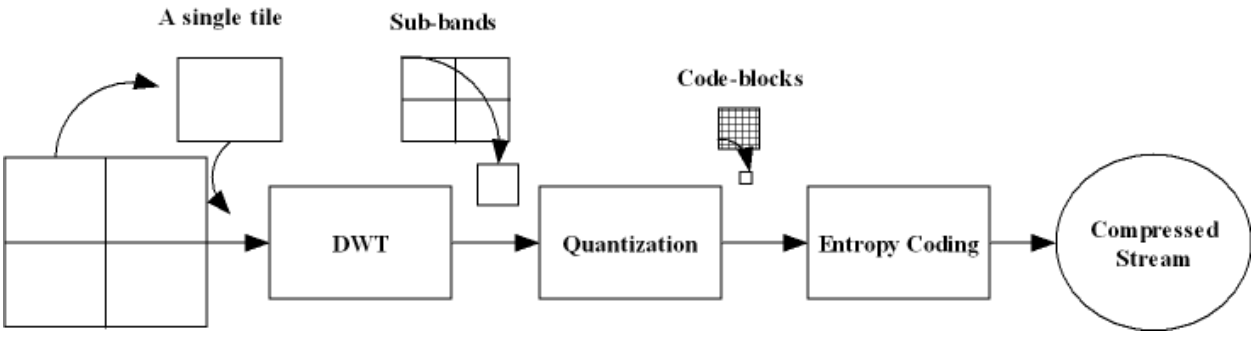
- DCT: Discrete Concise Transform
- DCT transforms the 64 values in 8x8 pixel block in a way that the relative relationships between pixels are kept but the redundancies are revealed.
- Example:







JPEG 2000

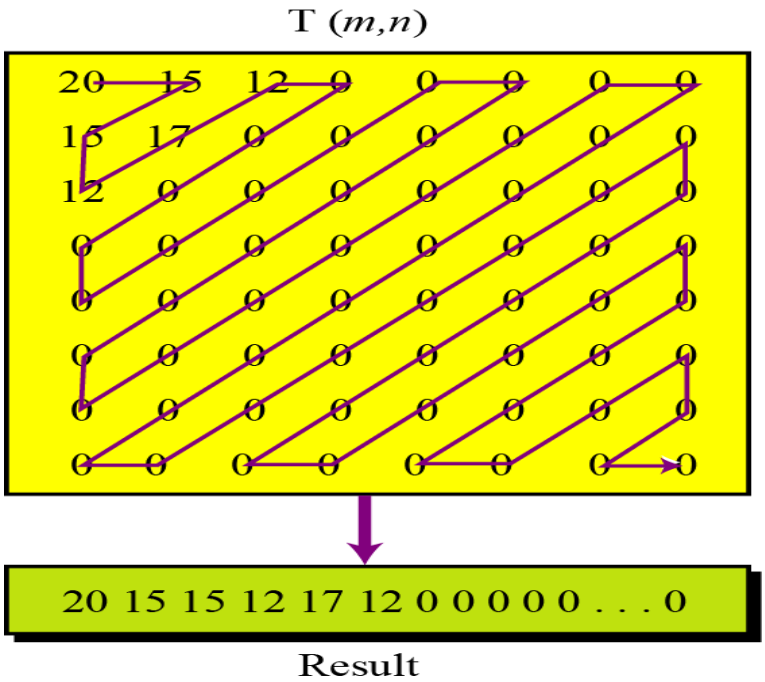


Quantization:

- After T table is created, the values are quantized to reduce the number of bits needed for encoding.
- Quantization divides the number of bits by a constant, then drops the fraction. This is done to optimize the number of bits and the number of 0s for each particular application.

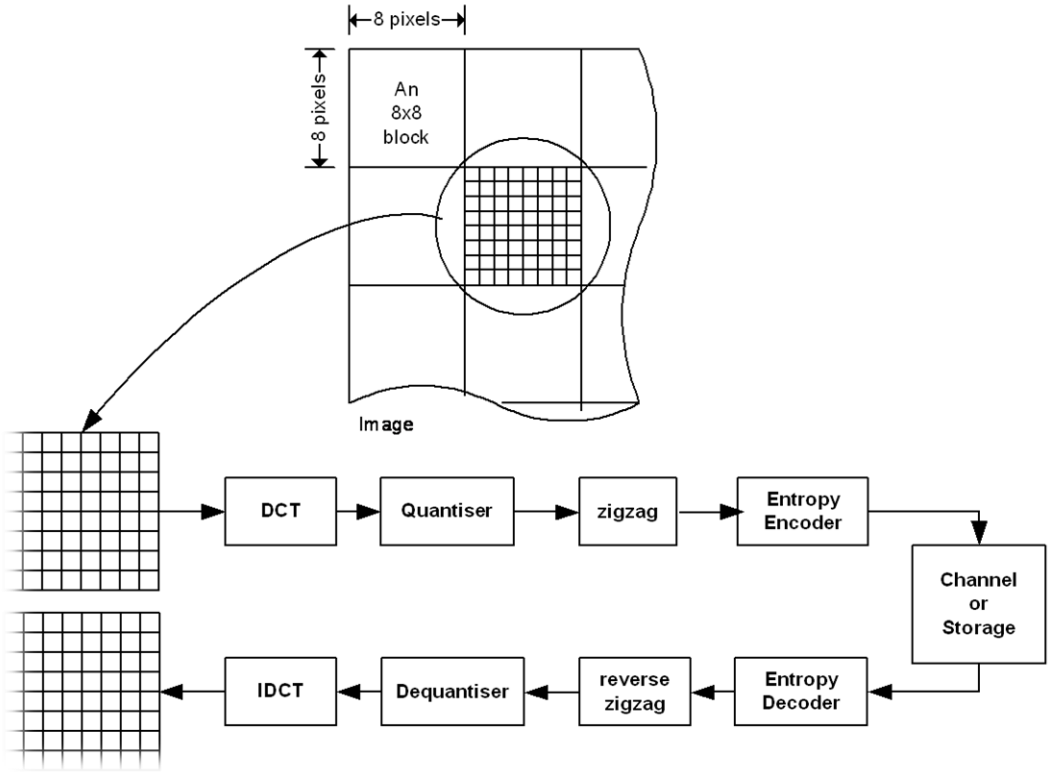
Compression:

- Quantized values are read from the table and redundant 0s are removed.
- To cluster the 0s together, the table is read diagonally in an zigzag fashion. The reason is if the table doesn't have fine changes, the bottom right corner of the table is all 0s.
- JPEG usually uses lossless run-length encoding at the compression phase.



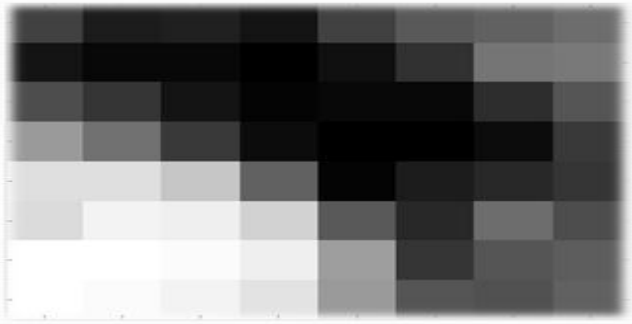
STEPS IN JPEG COMPRESSION

- Divide Each plane into 8x8 size blocks.
- Compute DCT of each block
- Treat separately DC components of each block.
- Apply Quantization to discard values
- Separately encode DC components and transmit data.



How JPEG Compression performed?

The example image matrix before transformation.

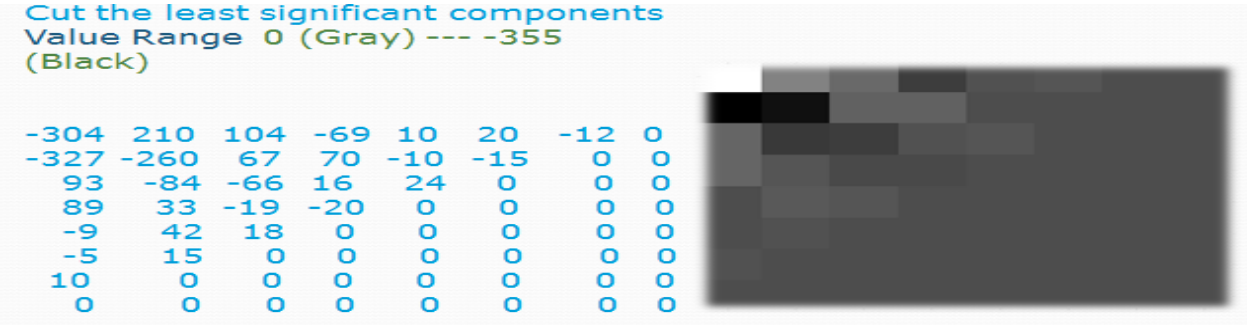


Gray-Scale Example:
Value Range 0 (black) --- 255 (white)

63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86

2D-DCT of matrix
Value Range 0 (Gray) --- -355 (Black)

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1



- The JPEG2000 standard is also based on the wavelet transform
- It provides high quality images at very high compression ratios
- The committee that developed the standard had certain goals for JPEG2000
- The goals are as follows:

1. To provide better compression than the DCT-based JPEG algorithm
2. To allow for progressive transmission of high quality images
3. To be able to compress binary and continuous tone images by allowing 1 to 16 bits for image components
4. To allow random access to subimages

5. To be robust to transmission errors

- To allow for sequentially image encoding
-

- The JPEG2000 compression method begins by level shifting the data to center it at zero, followed by an optional transform to decorrelate the data, such as a color

transform for color images The one-dimensional wavelet transform is applied to the rows and columns, and the coefficients are quantized based on the image size and number of wavelet bands utilized

- These quantized coefficients are then arithmetically coded on a bitplane basis

Transform Coding

- A reversible linear transform (such as Fourier Transform) is used to map the image into a set of transform coefficients
- These coefficients are then quantized and coded.
- The goal of transform coding is to decorrelate pixels and pack as much information into small number of transform coefficients.
- Compression is achieved during quantization not during the transform step

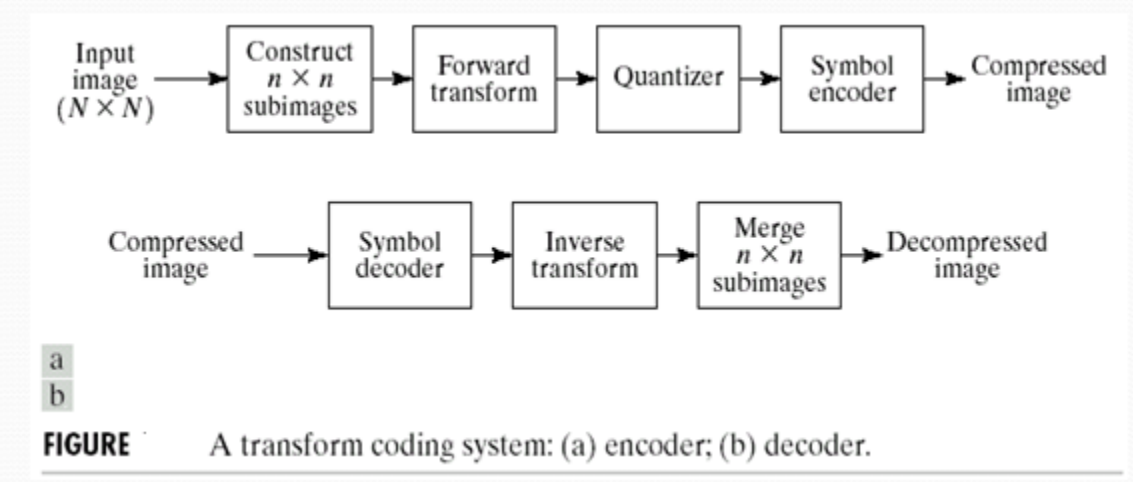


FIGURE A transform coding system: (a) encoder; (b) decoder.

Procedure

- Divide the image into $n \times n$ sub-images.
- Transform each sub-image using a reversible transform (e.g., the Hotelling transform, the discrete Fourier transform (DFT) or the discrete cosine transform (DCT)).
- Quantify, i.e., truncate the transformed image (e.g., by using DFT,and DCT frequencies with small amplitude can be removed without much information

loss). The quantification can be either image dependent (IDP) or image independent (IIP).

- Code the resulting data, normally using some kind of “variable length coding”, e.g., Huffman code.
- The coding is not reversible (unless step 3 is skipped). Divide the image into $n \times n$ sub-images.

Transform coding, is a form of block coding done in the transform domain. The image is divided into blocks, or subimages, and the transform is calculated for each block

- Any of the previously defined transforms can be used, frequency (e.g. Fourier) or sequency (e.g. Walsh/Hadamard), but it has been determined that the discrete cosine transform (DCT) is optimal for most images
- The newer JPEG2000 algorithms uses the wavelet transform, which has been found to provide even better compression
-
- After the transform has been calculated, the transform coefficients are quantized and coded
- This method is effective because the frequency/sequency transform of images is very efficient at putting most of the information into relatively few coefficients, so many of the high frequency coefficients can be quantized to 0 (eliminated completely)
- This type of transform is a special type of mapping that uses spatial frequency concepts as a basis for the mapping
- The main reason for mapping the original data into another mathematical space is to pack the information (or energy) into as few coefficients as possible
- The simplest form of transform coding is achieved by *filtering* by eliminating some of the high frequency coefficients

- However, this will not provide much compression, since the transform data is typically *floating point* and thus 4 or 8 bytes per pixel (compared to the

original pixel data at 1 byte per pixel), so quantization and coding is applied to the reduced data

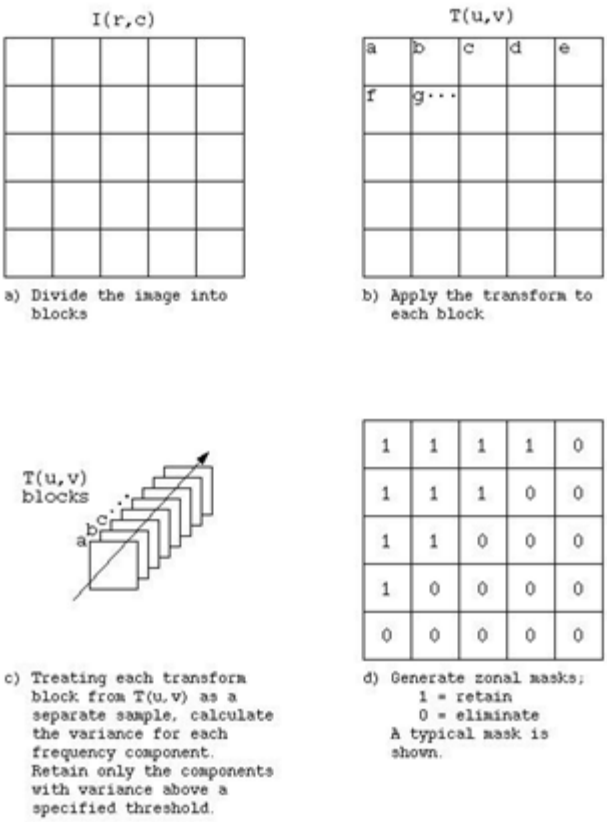
- Quantization includes a process called *bit allocation*, which determines the number of bits to be used to code each coefficient based on its importance

- Typically, more bits are used for lower frequency components where the energy is concentrated for most images, resulting in a *variable bit rate* or *nonuniform quantization* and better resolution
    - Two particular types of transform coding have been widely explored:
    1. *Zonal coding*
    2. *Threshold coding*
    - These two vary in the method they use for selecting the transform coefficients to retain (using ideal filters for transform coding selects the coefficients based on their location in the transform domain)

Zonal coding

- It involves selecting specific coefficients based on maximal variance
- A zonal mask is determined for the entire image by finding the variance for each frequency component
- This variance is calculated by using each subimage within the image as a separate sample and then finding the variance within this group of subimages

Figure ': Zonal Coding



- The *zonal mask* is a bitmap of 1's and 0', where the 1's correspond to the coefficients to retain, and the 0's to the ones to eliminate
- As the zonal mask applies to the *entire image,* only one mask is required

Threshold coding

- It selects the transform coefficients based on specific value
- A different threshold mask is required for each block, which increases file size as well as algorithmic complexity
- In practice, the zonal mask is often predetermined because the low frequency terms tend to contain the most information, and hence exhibit the most variance
- In this case we select a fixed mask of a given shape and desired compression ratio, which streamlines the compression process
- It also saves the overhead involved in calculating the variance of each group of subimages for compression and also eases the decompression process

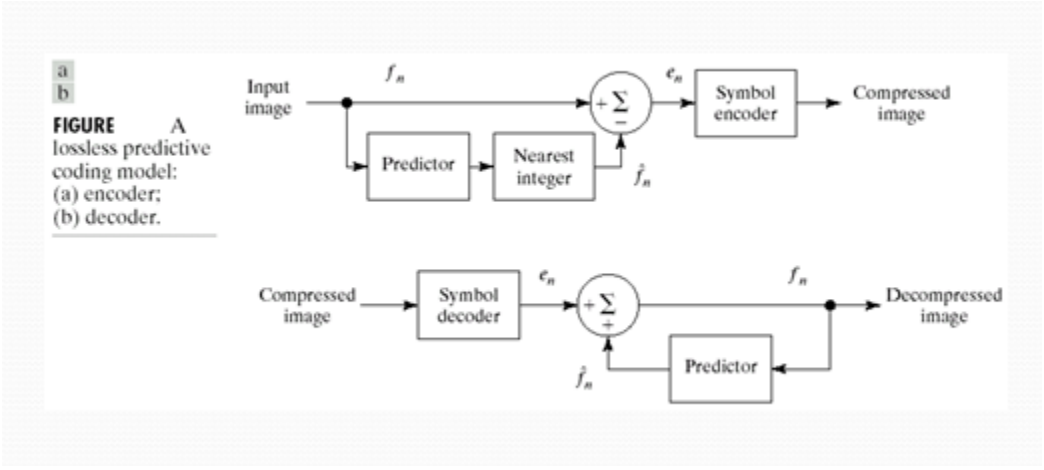
- Typical masks may be square, triangular or circular and the cutoff frequency is determined by the compression ratio

Lossless Predictive Coding

Let us now turn to an error-free compression approach that does not require decomposition of an image into a collection of bit planes The approach, commonly referred to as *lossless predictive coding*, is based on eliminating the interpixel redundancies of closely spaced pixels by extracting and coding *only* the new information in each pixel The *new information* of a pixel is defined as the difference between the actual and predicted value of that pixel

Figure shows the basic components of a lossless predictive coding system. The system consists of an encoder and a decoder, each containing an identical *predictor*. As each successive pixel of the input image, denoted f_n, is introduced to the encoder, the predictor generates the anticipated value of that pixel based on some number of past inputs. The output of the predictor is then rounded to the nearest integer, denoted $\hat{f}_n$, and used to form the difference or *prediction error*

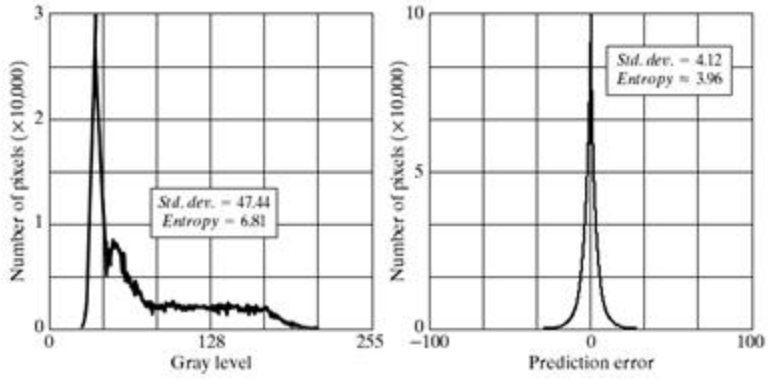
$$e_n = f_n - \hat{f}_n,$$



a b
FIGURE A lossless predictive coding model: (a) encoder; (b) decoder.

a
b c

FIGURE
(a) The prediction error image resulting from Eq. (8.4-9). (b) Gray-level histogram of the original image. (c) Histogram of the prediction error.



which is coded using a variable-length code (by the symbol encoder) to generate the next element of the compressed data stream. The decoder of Fig. (b) reconstructs e_n from the received variable-length code words and performs the inverse operation

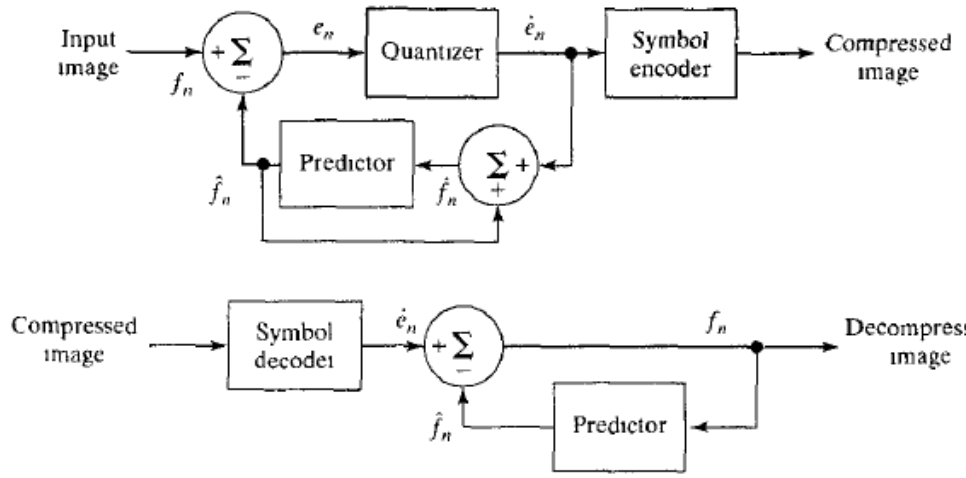
$$f_n = e_n + \hat{f}_n$$

Various local, global, and adaptive methods can be used to generate $\hat{f}_n$. In most cases, however, the prediction is formed by a linear combination of m previous pixels. That is,

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

Lossy Predictive Coding

In this section, we add a quantizer to the model and examine the resulting trade-off between reconstruction accuracy and compression performance. As Fig. shows, the quantizer, which absorbs the nearest integer function of the error-free encoder, is inserted between the symbol



a
b

FIGURE A lossy predictive coding model: (a) encoder and (b) decoder.

encoder and the point at which the prediction error is formed. It maps the prediction error into a limited range of outputs, denoted $\dot{e}_n$, which establish the amount of compression and distortion associated with lossy predictive coding.

In order to accommodate the insertion of the quantization step, the error-free encoder of Fig. 8.19(a) must be altered so that the predictions generated by the encoder and decoder are equivalent. As Fig. 8.21(a) shows, this is accomplished by placing the lossy encoder's predictor within a feedback loop, where its input, denoted f_n, is generated as a function of past predictions and the corresponding quantized errors. That is,

$$\dot{f}_n = \dot{e}_n + \hat{f}_n$$

CCITT Compression method

Most of the standards discussed are sanctioned by the International Standardization Organization (ISO) and the Consultative Committee of the International Telephone and Telegraph (CCITT). They address both binary and continuous-tone (monochrome and color) image compression, as well as both still-frame and video (i.e., sequential-frame) applications.

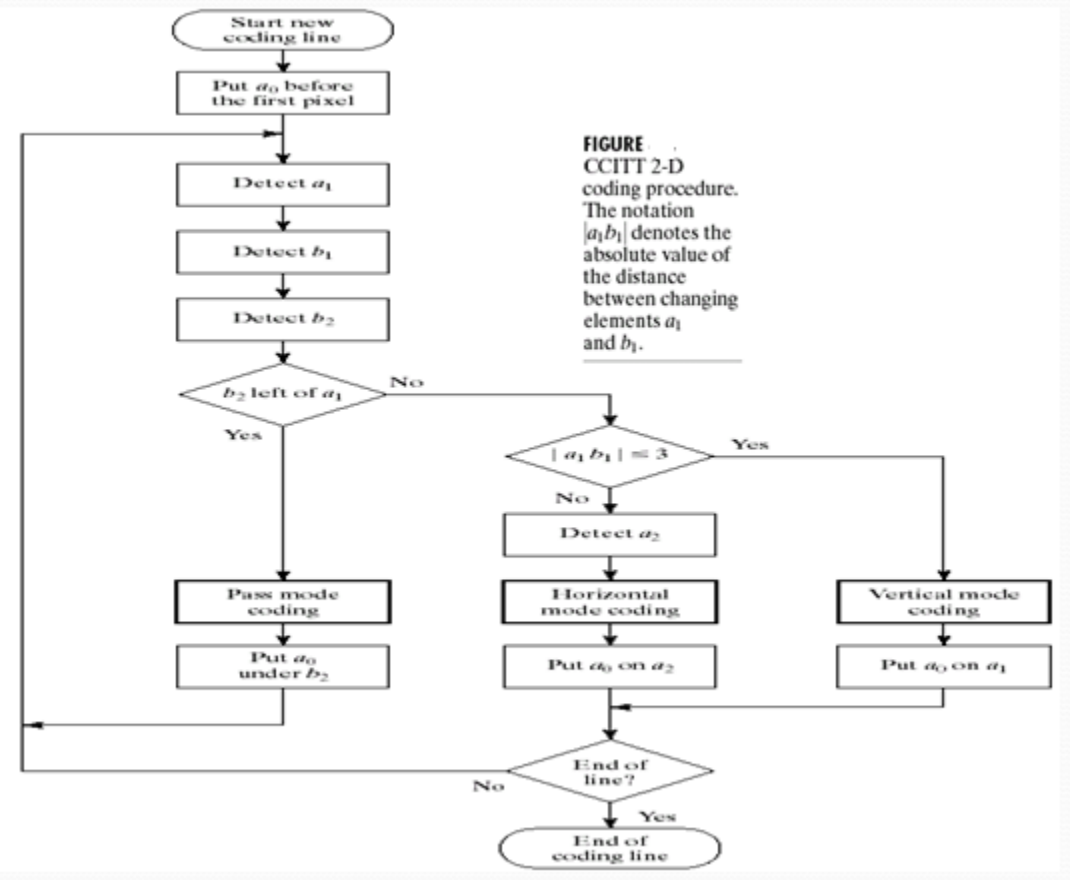
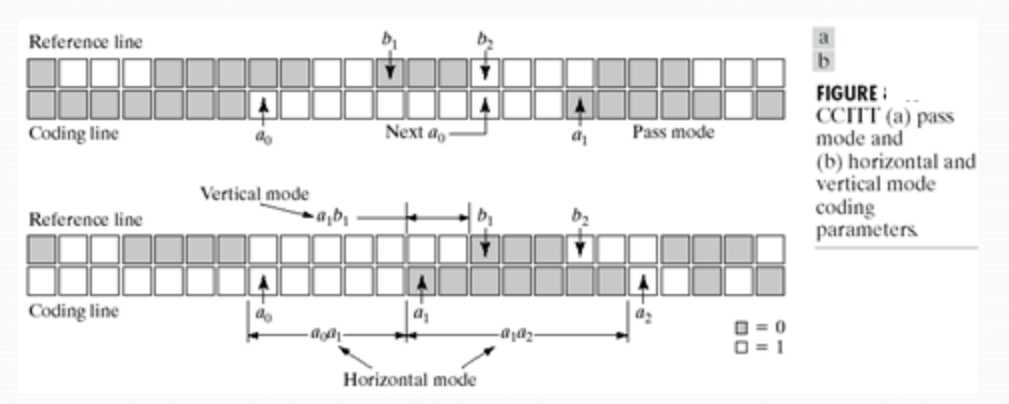


FIGURE
CCITT 2-D coding procedure. The notation $|a_1 b_1|$ denotes the absolute value of the distance between changing elements a_1 and b_1.



a
b

FIGURE : CCITT (a) pass mode and (b) horizontal and vertical mode coding parameters.

IMPORTANT QUESTIONS

TWO MARKS QUESTIONS

1. What is image compression?
2. What are two main types of compression?
3. What is the need for Compression?
4. What are different Compression Methods?
5. Define is coding redundancy?
6. Define interpixel redundancy?
7. What is run length coding?
8. Define compression ratio.
9. Define psycho visual redundancy?
10. Write notes on threshold coding.

12 MARKS QUESTIONS

1. Explain the schematics of image compression standard JPEG.
2. Explain CCITT Image compression standard.
3. Explain lossy and lose less predictive coding with a neat sketch.

4. Briefly explain (a) variable length coding (b) Transform coding (c) Zonal coding.
5. Explain Shannon’s coding with a suitable example.
6. Explain Huffman coding with a suitable example.
7. Explain Pixel and threshold coding with a suitable example.