UNIT-V

IMAGE COMPRESSION

1.1 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.

1.2 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 minimise 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.

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
HDTV (59.94 fps) framesize 1280x720, 8bits/pixel	1.33 Gbps	20 Mbps

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 shown above 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.

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

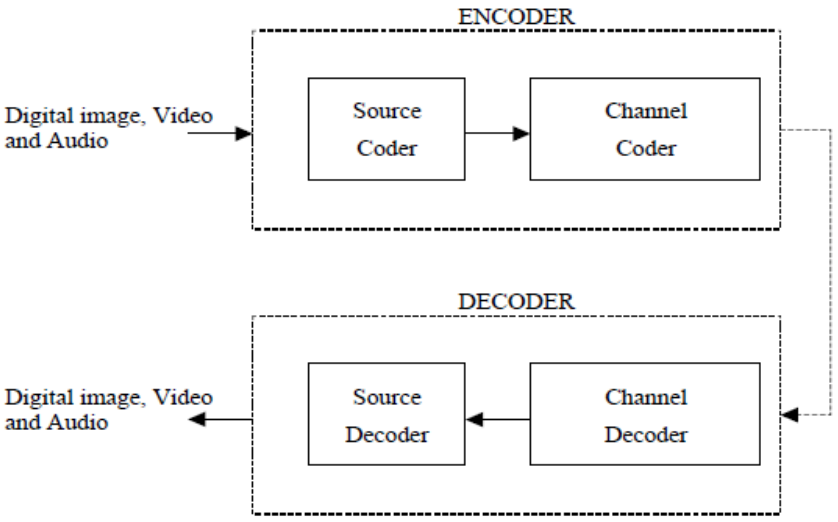


Fig: Generic compression system

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 is also referred to as bits per pixel or bits per pel. In the literature, the term *compression ratio,* denoted as c_r , is also used instead of *bit rate* to characterise the capability of the compression system. An intuitive definition of c_r is

c_r =(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 minimization 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.

1.2 Types of compression:

There are two types og mage compression namely

- Lossless compression
- Lossy compression

(i)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 5 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 that is, coding efficiency, coding complexity, and coding delay.

Coding Efficiency:

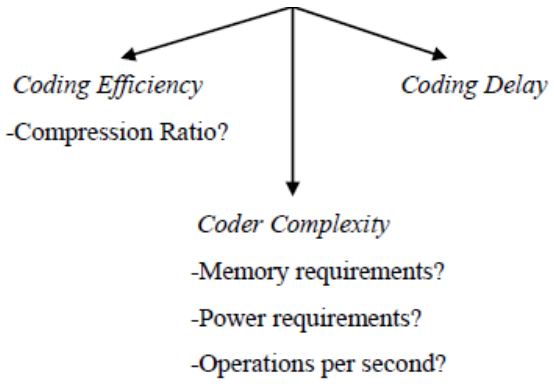
This is usually measured in bits per sample or bits per second (bps). Coding efficiency is usually limited by the information content or *entropy* of the source. In intuitive terms, the entropy of a source X provides a measure for the "randomness" of X. From a compression theory point of view, sources 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 characterized 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 characterized 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. The need to constrain the coding delay often forces the compression system designer to use a less sophisticated algorithm for the compression processes.



From this discussion, it can be concluded that these trade-offs in coding complexity, delay, and efficiency are usually limited to a small set of choices along these axes. In a subsequent section, we will briefly describe the trade-offs within the context of specific lossless compression methods.

(ii)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. 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.

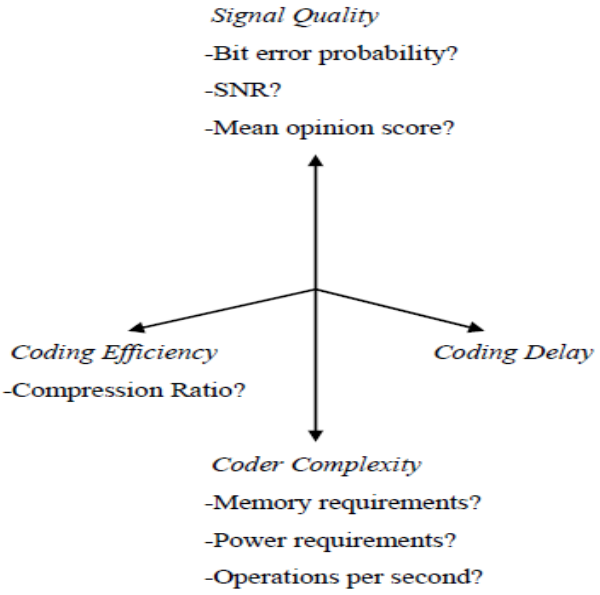
Signal Quality: This term is often used to characterize the signal at the output of the decoder. There is no universally accepted measure for signal quality.

One measure that is often cited is the signal to noise ratio, which can be expressed as SNR.

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. For instance, a five point scale such as *very annoying, annoying, slightly annoying, perceptible but not annoying,* and *imperceptible* might be used to characterize the impairments in the decoder output.

In either lossless or lossy compression schemes, the quality of the input data affects the compression ratio. For instance, acquisition noise, data sampling timing errors, and even the analogue-to-digital conversion process affects the signal quality and reduces the spatial and

temporal correlation. Some compression schemes are quite sensitive to the loss in correlation and may yield significantly worse compression in the presence of noise.



1.3 Issues in compression method selection:

In this chapter, we have introduced some fundamental concepts related to image, video, and audio compression. When choosing a specific compression method, one should consider the following issues:

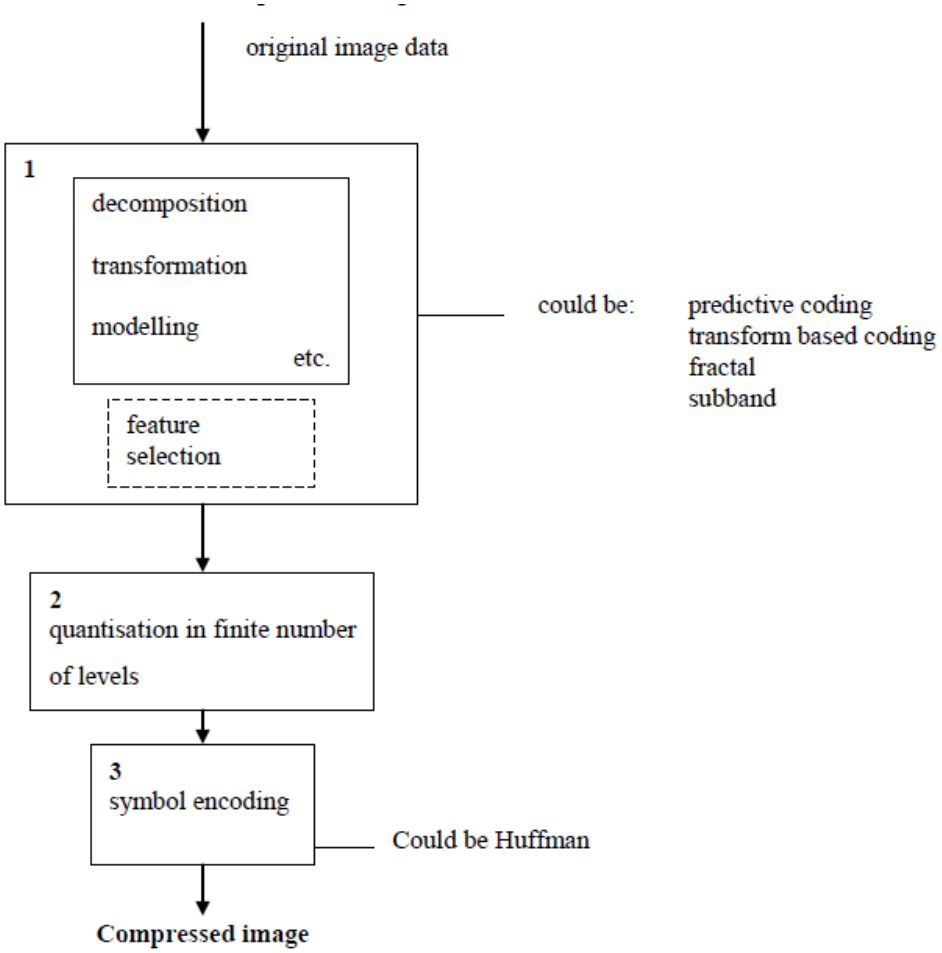
- Lossless or lossy. This is usually dictated by the coding efficiency requirements. Coding efficiency. Even in a lossy compression process, the desirable coding efficiency might not be achievable. This is especially the case when there are specific constraints on output signal quality.
- Variability in coding efficiency. In some applications, large variations in coding efficiency among different data sets may not be acceptable.
- Resilience to transmission errors. Some compression methods are more robust to transmission errors than others. If retransmissions are not permitted, then this requirement may impact on the overall encoder- decoder design.
- Complexity trade-offs. In most implementations, it is important to keep the overall encoder-decoder complexity low. However, certain applications may require only a low decoding complexity.
- Nature of degradations in decoder output. Lossy compression methods introduce artifacts in the decoded signal. The nature of artifacts depends on the compression method that is employed. The degree to which these artifacts are judged also varies

from application to application. In communication systems, there is often an interplay between the transmission errors and the coding artifacts introduced by the coder. Thus, it is important to consider all types of error in a system design.

- Data representation. In many applications, there is a need to support two decoding phases. In the first phase, decoding is performed to derive an intelligible signal; this is the case in data browsing. In the second phase, decoding is performed to derive a higher quality signal. One can generalise this notion to suggest that some applications require a hierarchical representation of the data. In the compression context, we refer to such compression schemes as *scalable compression methods.* The notion of scalability has been adopted in the compression standards.
- Multiple usage of the encoding-decoding tandem. In many applications, such as video editing, there is a need to perform multiple encode-decode operations using results from a previous encode-decode operation. This is not an issue for lossless compression; however, for lossy schemes, resilience to multiple encoding-decoding cycles is essential.
- Interplay with other data modalities, such as audio and video. In a system where several data modalities have to be supported, the compression methods for each modality should have some common elements. For instance, in an interactive videophone system, the audio compression method should have a frame structure that is consistent with the video frame structure. Otherwise, there will be unnecessary requirements on buffers at the decoder and a reduced tolerance to timing errors.
- Interworking with other systems. In a mass-market environment, there will be multiple data modalities and multiple compression systems. In such an environment, transcoding from one compression method to another may be needed. For instance, video editing might be done on a frame by frame basis; hence, a compression method that does not exploit temporal redundancies might be used here. After video editing, there might be a need to broadcast this video. In this case, temporal redundancies can be exploited to achieve a higher coding efficiency. In such a scenario, it is important to select compression methods that support transcoding from one compressed stream format to another. Interworking is important in many communications environments as well.

1.4 The Source coder:

In this course we are interested in exploring various compression techniques referring to the **source coder** only, where the image compression takes place. A general procedure for image data compression is shown in the following block diagram.



1.5 Image Formats and compression standards:

(a)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.

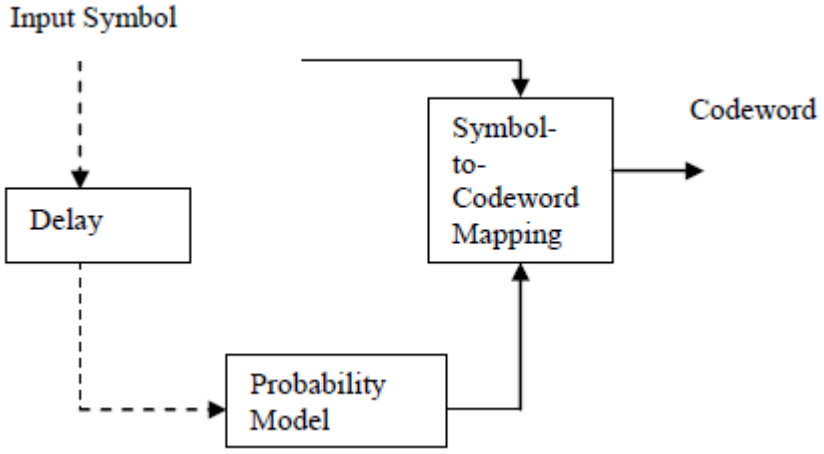


Fig: A generic model of lossless compression

Given an input set of symbols, a modeler generates an estimate of the probability distribution of the input symbols. This probability model is then used to map symbols into code words. The combination of the probability modeling and the symbol-to-codeword mapping functions is usually referred to as **entropy coding**. The key idea of entropy coding is to use short code words for symbols that occur with high probability and long code words for symbols that occur with low probability.

The probability model can be derived either from the input data or from a priori assumptions about the data. Note that, for decodability, the same model must also be generated by the decoder. Thus, if the model is dynamically estimated from the input data, causality constraints require a delay function between the input and the modeler. If the model is derived from a priori assumptions, then the delay block is not required; furthermore, the model function need not have access to the input symbols. The probability model does not have to be very accurate, but the more accurate it is, the better the compression will be. Note that, compression is not always guaranteed. If the probability model is wildly inaccurate, then the output size may even expand. However, even then the original input can be recovered without any loss.

Decompression is performed by reversing the flow of operations shown in the above Figure. This decompression process is usually referred to as **entropy decoding**.

(i)Differential coding:

Another preprocessing technique that improves the compression ratio is differential coding.

Differential coding skews the symbol statistics so that the resulting distribution is more amenable to compression. Image data tend to have strong inter-pixel correlation. If, say, the pixels in the image are in the order $x_1, x_2, \ldots . x_N$, then instead of compressing these pixels, one might process the sequence of differentials $y_i = x_i - x_{i-1}$, where $i = 0,1,2 \ldots . N-1$, and $x_0 = 0$. In compression terminology, $i\ y$ is referred to as the **prediction residual** of $i\ x$. The notion of compressing the prediction residual instead of $i\ x$ is used in all the image and video compression standards. For images, a typical probability distribution for $i\ x$ and the resulting distribution for $i\ y$ are shown in Figure.

Let symbol S_i have a probability of occurrence P_i. From coding theory, the ideal symbol-to codeword mapping function will produce a codeword requiring $\log (1/2P_i)$ bits. A distribution close to uniform for P_i, such as the one shown in the left plot of Figure, will result in codewords that on the average require eight bits; thus, no compression is achieved. On the other hand, for a skewed probability distribution, such as the one shown in the right plot of Figure, the **symbol-to-codeword mapping function** can on the average yield codewords requiring less than eight bits per symbol and thereby achieve compression.

We will understand these concepts better in the following Huffman encoding section.

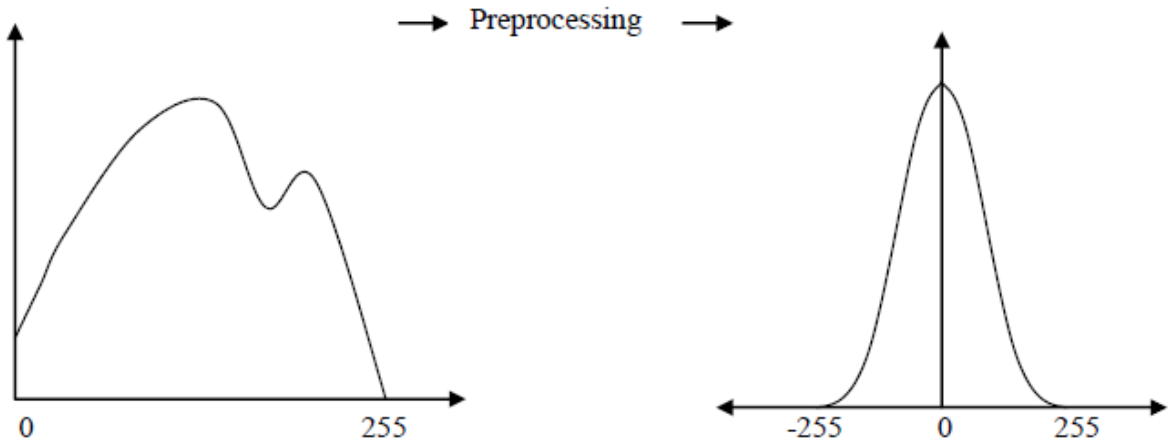


Fig: Typical distribution of pixel values for x_i and y_i.

(ii)Huffman coding:

In 1952, D. A. Huffman developed a code construction method that can be used to perform lossless compression. In Huffman coding, the modeling and the symbol-to-codeword mapping functions of Figure 1.1 are combined into a single process. As discussed earlier, **the input data are partitioned into a sequence of symbols** so as to facilitate the modeling process. In most image and video compression applications, the size of the alphabet

composing these symbols is restricted to at most 64000 symbols. The Huffman code construction procedure evolves along the following parts:

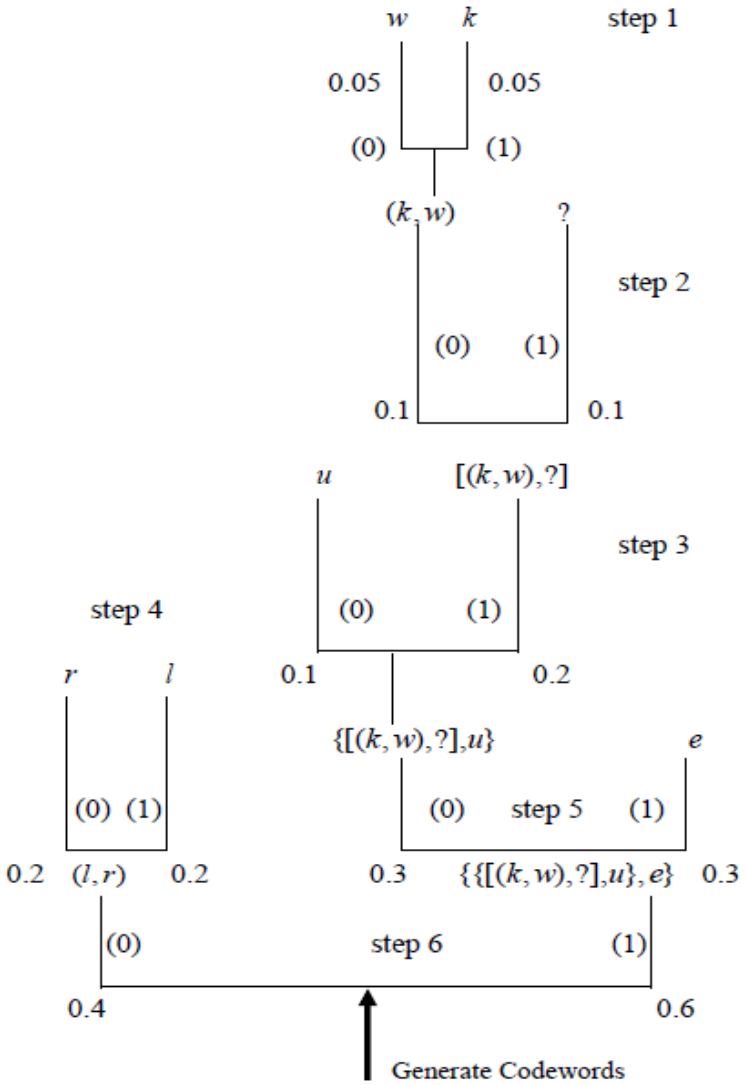
1. Order the symbols according to their probabilities. For Huffman code construction, the frequency of occurrence of each symbol must be known a priori. In practice, the frequency of occurrence can be estimated from a training set of data that is representative of the data to be compressed in a lossless manner. If, say, the alphabet is composed of *N* distinct symbols $s_1,s_2,s_3\ldots s_{N-1}$ and the probabilities of occurrence are $p_1,p_2\ldots p_{N-1}$, then the symbols are rearranged so that.

2. Apply a contraction process to the two symbols with the smallest probabilities. Suppose the two symbols are $s_1..s_N$. We replace these two symbols by a hypothetical symbol, say, $H_{N-1}=(s_{N-1},s_N)$that has a probability of occurrence *N N p* □ *p* □1 . Thus, the new set of symbols has N-1 members $s_1,s_2\ldots s_{N-1}$

3. We repeat the previous part 2 until the final set has only one member.

The recursive procedure in part 2 can be viewed as the construction of a binary tree, since at each step we are merging two symbols. At the end of the recursion process all the symbols *N s1,s2..sN-1* will be leaf nodes of this tree. The codeword for each symbol S_i is obtained by traversing the binary tree from its root to the leaf node corresponding to S_i

We illustrate the code construction process with the following example depicted in Figure. The input data to be compressed is composed of symbols in the alphabet *k, l,u,w,e, r,*? . First we sort the probabilities. In Step 1, we merge the two symbols *k* and *w* to form the new symbol (*k,w*) . The probability of occurrence for the new symbol is the sum of the probabilities of occurrence for *k* and *w.* We sort the probabilities again and perform the merge on the pair of least frequently occurring symbols which are now the symbols (*k,w*) and ? . We repeat this process through Step 6. By visualizing this process as a binary tree as shown in this figure and traversing the process from the bottom of the tree to the top, one can determine the code words for each symbol. For example, to reach the symbol *u* from the root of the tree, one traverses nodes that were assigned the bits 1,0 and 0 . Thus, the codeword for *u* is 100.



	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
k 0.05	e 0.3	e 0.3	e 0.3	e 0.3	(l,r) 0.4	$\{\{[(k,w),?],u\},e\}$ 0.6
l 0.2	l 0.2	l 0.2	l 0.2	$\{[(k,w),?],u\}$ 0.3	e 0.3	(l,r) 0.4
u 0.1	r 0.2	r 0.2	r 0.2	l 0.2	$\{[(k,w),?],u\}$ 0.3	
w 0.05	u 0.1	u 0.1	$[(k,w),?]$ 0.2	r 0.2		
e 0.3	? 0.1	? 0.1	u 0.1			

r 0.2	k 0.05	(k,w) 0.1				
? 0.1	w 0.05					

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011

Fig: An example for Huffman coding

In this example, the average codeword length is 2.6 bits per symbol. In general, the average codeword length is defined as

$$l_{avg} = \sum l_i p_i$$

where l_i is the codeword length (in bits) for the codeword corresponding to symbol s_i . The average codeword length is a measure of the compression ratio. Since our alphabet has seven symbols, a fixed-length coder would require at least three bits per codeword. In this example, we have reduced the representation from three bits per symbol to 2.6 bits per symbol; thus, the corresponding compression ratio can be stated as 3/2.6=1.15. For the lossless compression of typical image or video data, compression ratios in excess of two are hard to come by.

Standards of Lossless compression:

Standards related to the coding and transmission of signals over public telecommunication channels are developed under the auspices of the telecommunication standardization sector of the International Telecommunication Union (ITU-T). This sector was formerly known as the CCITT. The first standards for lossless compression were developed for facsimile applications. Scanned images used in such applications are bitonal, that is, the pixels take on one of two values, black or white, and these values are represented with one bit per pixel.

(iii)Run-length coding:

In every bitonal image there are large regions that are either all white or all black. For instance,in Figure, we show a few pixels of a line in a bitonal image. Note that, the six contiguous pixels of the same color can be described as a run of six pixels with value 0 . Thus, if each pixel of the image is remapped from say, its (position, value) to a **run** and

value, then a more compact description can be obtained. In our example, no more than four bits are needed to describe the six-pixel run. In general, for many document type images, significant compression can be achieved using such preprocessing. Such a mapping scheme is referred to as a run-length coding scheme.

The combination of a run-length coding scheme followed by a Huffman coder forms the basis of the image coding standards for facsimile applications. These standards include the following:

- ITU-T Rec. T.4 (also known as Group 3). There are two coding approaches within this standard.

1. Modified Huffman (MH) code. The image is treated as a sequence of scanlines, and a runlength description is first obtained for each line. A Huffman code is then applied to the (run, value) description. A separate Huffman code is used to distinguish between black and white runs, since the characteristics of these runs are quite different. The Huffman code table is static: that is, it does not change from image to image. For error-detection purposes, after each line is coded, an EOL (end of line) codeword is inserted.

   2. Modified Read (MR) code. Here, pixel values in a previous line are used as predictors for the current line. This is followed by a run-length description and a static Huffman code as in the MH code. An EOL codeword is also used. To prevent error propagation, MR coding is mixed with MH coding periodically.

- ITU-T Rec. T.6 (also known as Group 4). The coding technique used here is referred to as a Modified Modified Read (MMR) code. This code is a simplification of the MR code, wherein the error-protection mechanisms in the MR code are removed so as to improve the overall compression ratio.

These compression standards yield good compression (20:1 to 50:1) for business-type scanned documents. For images composed of natural scenes and rendered as bitonal images using a halftoning technique the compression ratio is severely degraded. In Figure shown, the image on the left possesses characteristics representative of business document images whereas the image on the right is a typical halftone image. The former is characterized by long runs of black or white, and the static Huffman code in the facsimile compression

standards is matched to these run-lengths. In the latter image, the run-lengths are relatively short, spanning only one to two pixels and the static Huffman code is not matched to such runs. An adaptive arithmetic coder is better suited for such images.

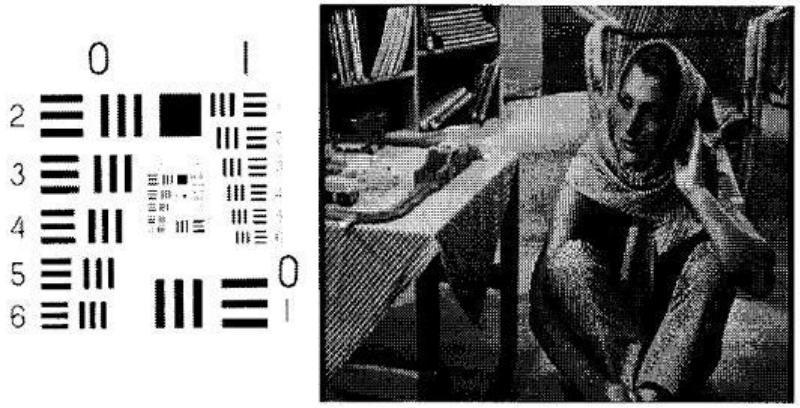


Fig: Typical Bitional Image

The JBIG Standard:

Recently, a compression standard was developed to efficiently compress halftone as well as business document type images. This is the JBIG (Joint Binary Image Experts Group) compression standard. Its standard nomenclature is ISO/IEC IS 11544, ITU-T Rec. T.82. The JBIG compression standard consists of a modeler and an arithmetic coder. The modeler is used to estimate the symbol probabilities that are then used by the arithmetic coder. **The JBIG is out of the scope of this course.** In Tables shown below, we provide a simple complexity analysis between the JBIG coding scheme and the ITU-T Rec. T.6 facsimile compressionstandard.

	JBIG-Baselayer	**ITU-T Rec. T.6**
Complexity Parameters	Three-line template, AT-max=16	2-D runlength, Huffman code
Memory	1589 bytes	1024 bytes
Buffer	Three scanlines	Two scanlines
Operations	Add, shift	Add, shift, compare

Compression		
Halftone Image	5.2:1	1.5:1
Letter Image	48:1	33.3:1

Table: Comparative analysis between JBIG and the ITU-T Rec. T.6 facsimile compression standards

We also provide compression ratios for two typical images: a 202 Kbyte halftone image and a 1 Mbyte image, primarily comprised of text. The latter image is referred to as letter in the table. For business-type documents, JBIG yields 20 percent to 50 percent more compression than the facsimile compression standards ITU-T Rec. T.4 and Rec. T.6. For halftone images, compression ratios with JBIG are two to five times more than those obtained with the facsimile compression standards. However, software implementations of JBIG compression on a general purpose computer are two to three times slower than implementations of the ITU-T Rec. T.4 and T.6 standards. The JBIG standard can also handle grayscale images by processing each bit-plane of a grayscale image as separate bitonal images.

The Lossless JPEG Standard:

Most people know JPEG as a transform-based lossy compression standard. JPEG (Joint Photographic Experts Group), like JBIG, has been developed jointly by both the ITU-T and the ISO. We will describe this standard in greater detail in a subsequent section; however, here, we describe briefly the lossless mode of compression supported within this standard. The lossless compression method within JPEG is fully independent from transform-based coding. Instead, it uses differential coding to form prediction residuals that are then coded with either a Huffman coder or an arithmetic coder. As explained earlier, the prediction residuals usually have a lower entropy; thus, they are more amenable to compression than the original image pixels. In lossless JPEG, one forms a prediction residual using previously encoded pixels in the current line and/or the previous line. The prediction residual for pixel in Figure is r = y-x.

where can be any of the following functions:

y =0
y = a
y = b
y = c
y = a + b – c
y = a (+b – c) / 2
y = b (+a – c) / 2
y (=a + b) / 2

Note that, pixel values at pixel positions, and, are available to both the encoder and the *X y* decoder prior to processing. The particular choice for the function is defined in the scan header of the compressed stream so that both the encoder and the decoder use identical functions. Divisions by two are computed by performing a one-bit right shift.

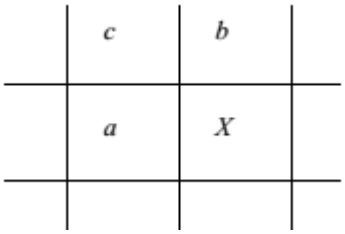


Fig: Lossless JPEG Prediction kernel

The prediction residual is computed modulo 2. This residual is not directly Huffman coded. Instead, it is expressed as a pair of symbols: the category and the magnitude. The first symbol represents the number of bits needed to encode the magnitude. Only this value is Huffman coded. The magnitude categories for all possible values of the prediction residual are shown in Table. If, say, the *X* prediction residual for is 42, then from Table 4.2 we determine that this value belongs to category 6; that is, we need an additional six bits to uniquely determine the value 42. The prediction residual is then mapped into the two-tuple (6, 6-bit code for 42). Category 6 is Huffman coded, and the compressed representation for the prediction residual consists of this Huffman codeword followed by the 6-bit representation for the magnitude. In general, if the value of the residual is positive, then the code for the magnitude is its direct binary representation. If the residual is negative, then the code for the magnitude is the one's complement of its absolute value. Therefore, code words for negative residual always start wish a zero bit.

1.6 Fundamental Coding Theorem:

At various points in their study of the image coding literature readers will come across with the R(D) application of rate-distortion theory, or its equivalent, distortion-rate theory. The significant introduction by Shannon (1959) of a fidelity criterion into his work on coding theory led to an extensive body of work which sought to characterize the relationship between **coding rate** and **measured distortion** in a consistent way. *R*(*D*) Briefly, the general form of the curve is as shown in the following figure where, as expected, for smaller levels of distortion we require a higher coding rate. If the attainable level of distortion is no smaller than *D*max then no information need be sent anyway. For an initially analogue input signal, as the distortion falls to zero the quantisation intervals must have a width which tends to zero and so the rate curve moves towards infinity (dotted curve in Figure). For a discrete signal we know that we can encode at a rate equivalent to the entropy and incur [*R*=(0) *H*] zero distortion, at least in principle . We may therefore set our operating point anywhere

≤0 $D \leq D_{max}$ within the region . Deviations between the results achieved with practical algorithms and the theory are to be expected, and are usually due to lack of knowledge of the true source $R(D)$ distribution (and the associated relation) and an inability to allocate fractional numbers of bits to encoded symbols (some kinds of block schemes, vector quantisation, for example, allow this to be done).

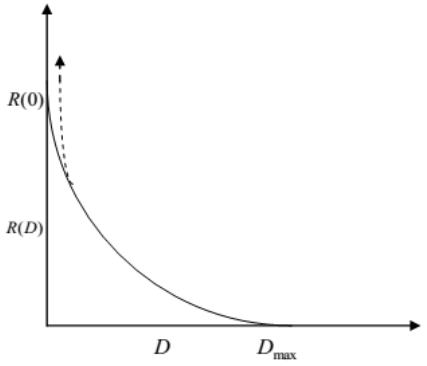


Fig: Rate-distortion relationship. For a discrete signal zero distortion coding is *R*=(0) achieved when the source entropy. For a continuous source zero distortion implies that the rate rises without limit (---).