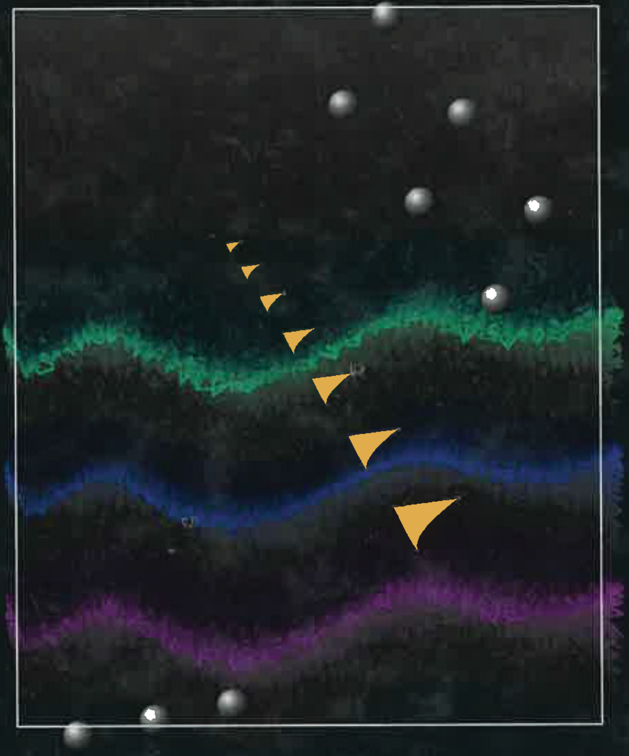
digital video processing



A. MURAT TEKALP

PRENTICE HALL SIGNAL PROCESSING SERIES

Digital Video Processing

A. Murat Tekalp
University of Rochester

Prentice Hall PTR
Upper Saddle River, NJ 07458

```
Tekalp, A. Murat.
     Digital video processing / A. Murat Tekalp.
          p.  cm. -- (Prentice-Hall signal processing series)
     ISBN 0-13-190075-7 (alk. paper)
     1. Digital video. I. Title. II. Series.
  TK6680.5.T45  1995
  621.388'33--dc20                                    95-16650
                                                          CIP
```

Editorial/production supervision: **Ann Sullivan**
Cover design: **Design Source**
Manufacturing manager: **Alexis R. Heydt**
Acquisitions editor: **Karen Gettman**
Editorial assistant: **Barbara Alfieri**

Printed on Recycled Paper

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN: 0-13-190075-7

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Chapter 18

LOSSLESS COMPRESSION

The need for effective data compression is evident in almost all applications where storage and transmission of digital images are involved. For example, an 8.5 $\times$ 11 in document scanned at 300 pixels/in with 1 bit/pixel generates 8.4 Mbits data, which without compression requires about 15 min transmission time over a 9600 baud line. A 35 mm film scanned at 12 micron resolution results in a digital image of size 3656 pixels $\times$ 2664 lines. With 8 bits/pixel per color and three color channels, the storage required per picture is approximately 233 Mbits. The storage capacity of a CD is about 5 Gbits, which without compression can hold approximately 600 pages of a document, or 21 color images scanned from 35 mm film. Several world standards for image compression, such as ITU (formerly CCITT) Group 3 and 4 codes, and ISO/IEC/CCITT JPEG, have recently been developed for efficient transmission and storage of binary, gray-scale, and color images.

Compression of image data without significant degradation of the visual quality is usually possible because images contain a high degree of i) spatial redundancy, due to correlation between neighboring pixels, ii) spectral redundancy, due to correlation among the color components, and iii) psychovisual redundancy, due to properties of the human visual system. The higher the redundancy, the higher the achievable compression. This chapter introduces the basics of image compression, and discusses some lossless compression methods. It is not intended as a formal review of the related concepts, but rather aims to provide the minimum information necessary to follow the popular still-image and video compression algorithms/standards, which will be discussed in the subsequent chapters. Elements of an image compression system as well as some information theoretic concepts are introduced in Section 18.1. Section 18.2 discusses symbol coding, and in particular entropy coding, which is an integral part of lossless compression methods. Finally, three commonly used lossless compression algorithms are presented in Section 18.3.

18.1 Basics of Image Compression

In this section, we first present the elements of a general image compression system, then summarize some results from the information theory which provide bounds on the achievable compression ratios and bitrates.

18.1.1 Elements of an Image Compression System

In information theory, the process of data compression by redundancy reduction is referred to as source encoding. Images contain two types of redundancy, statistical (spatial) and pyschovisual. Statistical redundancy is present because certain spatial patterns are more likely than others, whereas psychovisual redundancy originates from the fact that the human eye is insensitive to certain spatial frequencies. The block diagram of a source encoder is shown in Figure 18.1. It is composed of the following blocks:

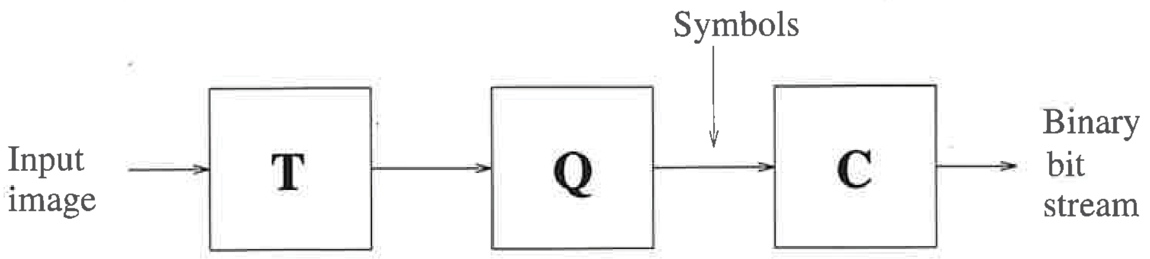


Figure 18.1: Block diagram of an image compression system.

i) Transformer (T) applies a one-to-one transformation to the input image data. The output of the transformer is an image representation which is more amenable to efficient compression than the raw image data. Typical transformations are linear predictive mapping, which maps the pixel intensities onto a prediction error signal by subtracting the predictible part of the pixel intensities; unitary mappings such as the discrete cosine transform, which pack the energy of the signal to a small number of coefficients; and multiresolution mappings, such as subband decompositions and the wavelet transform.
ii) Quantizer (Q) generates a limited number of symbols that can be used in the represention of the compressed image. Quantization is a many-to-one mapping which is irreversible. It can be performed by scalar or vector quantizers. Scalar quantization refers to element-by-element quantization of the data, whereas quantization of a block of data at once is known as vector quantization.
iii) Coder (C) assigns a codeword, a binary bitstream, to each symbol at the output of the quantizer. The coder may employ fixed-length or variable-length codes. Variable-length coding (VLC), also known as entropy coding, assigns codewords in such a way as to minimize the average length of the binary representation of the

symbols. This is achieved by assigning shorter codewords to more probable symbols, which is the fundamental principle of entropy coding.

Different image compression systems implement different combinations of these choices. Image compression methods can be broadly classified as:
i) Lossless (noiseless) compression methods, which aim to minimize the bitrate without any distortion in the image.
ii) Lossy compression methods, which aim to obtain the best possible fidelity for a given bitrate, or to minimize the bitrate to achieve a given fidelity measure.
The transformation and encoding blocks are lossless. However, quantization is lossy. Therefore, lossless methods, which only make use of the statistical redundancies, do not employ a quantizer. In most practical cases a small degradation in the image quality must be allowed to achieve the desired bitrate. Lossy compression methods make use of both the statistical and psychovisual redundancies.

In the following, we first briefly review some results from the information theory which gives bounds on the achievable bitrates in both lossless and lossy compression. In Section 18.2, we present techniques for symbol coding (the third box in Figure 18.1). The discussion of the second box, the quantizer, is deferred until the next chapter. Finally, some commonly used lossless image compression methods are presented in Section 18.3.

18.1.2 Information Theoretic Concepts

A source $\mathcal{X}$ with an alphabet $\mathcal{A}$ is defined as a discrete random process (a sequence of random variables X_i, $i = 1, \ldots$) in the form $\mathcal{X} = X_1\ X_2\ \ldots$, where each random variable X_i takes a value from the alphabet $\mathcal{A}$. In the following, we assume that the alphabet contains a finite number (M) of symbols, i.e., $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$. Here, we introduce two source models, a discrete memoryless source (DMS) and a Markov-K source.

A DMS is such that successive symbols are statistically independent. It is completely specified by the probabilities $p(a_i) = p_i$, $i = 1, \ldots, M$ such that $p_1 + \ldots + p_M = 1$. In VLC, the optimum length of the binary code for a symbol is equal to the information (in bits) that the symbol conveys. According to information theory, 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 occurence of a more likely symbol. This quantitative concept of surprise is formally expressed by the relation

$$I(a_i) = \log_2(1/p(a_i)), \quad \text{for } a_i \in \mathcal{A} \tag{18.1}$$

where $I(a_i)$ is the amount of information that the symbol a_i with probability $p(a_i)$ carries. The unit of information is bit when we use logarithm with base-2. Observe that if $p = 1$, then $I = 0$ as expected, and as $p \to 0$, $I \to \infty$. In practice, the probability of occurrence 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 $\mathcal{X}$ with an alphabet $\mathcal{A}$ is defined as the average information per symbol in the source, given by

$$\begin{aligned} H(\mathcal{X}) &= \sum_{a \in \mathcal{A}} p(a) \log_2 (1/p(a)) \\ &= -\sum_{a \in \mathcal{A}} p(a) \log_2 p(a) \end{aligned} \tag{18.2}$$

The more skewed the probability distribution of the symbols, the smaller the entropy of the source. The entropy is maximized for a flat distribution, that is, when all symbols are equally likely. It follows that a source where some symbols are more likely than others has a smaller entropy than a source where all symbols are equally likely.

Example: Entropy of raw image data

Suppose an 8-bit image is taken as a realization of a DMS $\mathcal{X}$. The symbols i are the gray levels of the pixels, and the alphabet $\mathcal{A}$ is the collection of all gray levels between 0 and 255. Then the entropy of the image is given by

$$H(\mathcal{X}) = -\sum_{i=0}^{255} p(i) \log_2 p(i)$$

where $p(i)$ denotes the relative frequency of occurrence of the gray level i in the image. Note that the entropy of an image consisting of a single gray level (constant image) is zero.

Most realistic sources can be better modeled by Markov-K random processes. That is, the probability of occurence of a symbol depends on the values of K preceding symbols. A Markov-K source can be specified by the conditional probabilities $p(X_j = a_i | X_{j-1}, \ldots, X_{j-K})$, for all j, $a_i \in \mathcal{A}$. The entropy of a Markov-K source is defined as

$$H(\mathcal{X}) = \sum_{S^K} p(X_{j-1}, \ldots, X_{j-K}) H(\mathcal{X} | X_{j-1}, \ldots, X_{j-K}) \tag{18.3}$$

where S^K denotes all possible realizations of $X_{j-1}, \ldots, X_{j-K}$, and

$$H(\mathcal{X} | X_{j-1}, \ldots, X_{j-K}) = \sum_{a \in \mathcal{A}} p(a_i | X_{j-1}, \ldots, X_{j-K}) \log p(a_i | X_{j-1}, \ldots, X_{j-K})$$

In the following, we present two fundamental theorems, the Lossless Coding Theorem and the Source Coding Theorem, which are used to measure the performance of lossless coding and lossy coding systems, respectively.

Lossless Coding Theorem: [Shannon, 1948] The minimum bitrate that can be achieved by lossless coding of a discrete memoryless source $\mathcal{X}$ is given by

$$\min\{R\} = H(\mathcal{X}) + \epsilon \text{ bits/symbol} \tag{18.4}$$

where R is the transmission rate, $H(\mathcal{X})$ is the entropy of the source, and ϵ is a positive quantity that can be made arbitrarily close to zero.

The Lossless Coding Theorem establishes the lower bound for the bitrate necessary to achieve zero coding-decoding error. In the case of a DMS, we can approach this bound by encoding each symbol independently. For sources with memory, we need to encode blocks of N source symbols at a time to come arbitrarily close to the bound. For example, a Markov-M source should be encoded M symbols at a time. In the next section, we introduce two coding techniques, Huffman coding and arithmetic coding, that approach the entropy bound.

In lossy coding schemes, the achievable minimum bitrate is a function of the distortion that is allowed. This relationship between the bitrate and distortion is given by the rate distortion function [Ber 71].

Source Coding Theorem: 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

$$R \geq R(D) \tag{18.5}$$

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

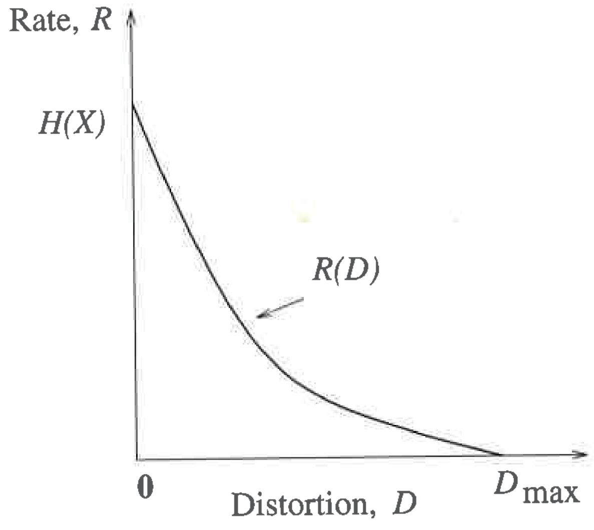


Figure 18.2: Rate distortion function.

A typical rate-distortion function is depicted in Figure 18.2. 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 [Ber 71]. 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. Note that the source coding theorem does not state how to design algorithms to achieve these desired limits. Some well-known lossy coding algorithms are discussed in Chapters 19, 20, and 21.

18.2 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. Because compression is generally achieved by assigning shorter-length codewords to more probable symbols, we next describe two variable-length coding, also known as entropy coding, schemes. The first, Huffman coding, assigns variable-length codes to a fixed-length block of symbols, where the block length can be one. In Huffman coding, the length of the codewords is proportional to the information (in bits) of the respective symbols or block of symbols. The latter, arithmetic coding, assigns variable-length codes to a variable-length block of symbols.

18.2.1 Fixed-Length Coding

In fixed-length coding, we assign equal-length code words to each symbol in the alphabet $\mathcal{A}$ regardless of their probabilities. If the alphabet has M different symbols (or blocks of symbols), then the length of the code words is the smallest integer greater than $\log_2 M$. Two commonly used fixed-length coding schemes are natural codes and Gray codes, which are shown in Table 18.1 for the case of a four-symbol source. Notice that in Gray coding, the consecutive codewords differ in only one bit position. This property of the Gray codes may provide an advantage in error detection. We will see in Section 18.3 that Gray codes are also better suited for run-length encoding of bit-planes.

Table 18.1: Fixed-length codes for a four-symbol alphabet.

Symbol	Natural code	Gray code
a_1	00	00
a_2	01	01
a_3	10	11
a_4	11	10

It can easily be shown that fixed-length coding is optimal only when:

1) the number of symbols is equal to a power of 2, and
2) all the symbols are equiprobable.

Only then would the entropy of the source be equal to the average length of the codewords, which is equal to the length of each codeword in the case of fixed-length coding. For the example shown in Table 18.1, both the entropy of the source and the average codeword length is 2, assuming all symbols are equally likely. Most often, some symbols are more probable than others, where it would be more advantageous to use entropy coding. Actually, the goal of the transformation box in Figure 18.1 is to obtain a set of symbols with a skew probability distribution, to minimize the entropy of the transformed source.

18.2.2 Huffman Coding

Huffman coding yields the optimal integer prefix codes given a source with a finite number of symbols and their probabilities. In prefix codes, no codeword is a prefix of another codeword. Such codes are uniquely decodable since a given binary string can only be interpreted in one way. Huffman codes are optimal in the sense that no other integer-length VLC can be found to yield a smaller average bitrate. In fact, the average length of Huffman codes per codeword achieves the lower bound, the entropy of the source, when the symbol probabilities are all powers of 2.

Huffman codes can be designed by following a very simple procedure. Let $\mathcal{X}$ denote a DMS with the alphabet $\mathcal{A}$ and the symbol probabilities $p(a_i)$, $a_i \in \mathcal{A}$, $i = 1, \ldots, M$. Obviously, if $M = 2$, we must have

$$c(a_1) = 0 \quad \text{and} \quad c(a_2) = 1 \tag{18.6}$$

where $c(a_i)$ denotes the codeword for the symbol a_i, $i = 1, 2$. If $\mathcal{A}$ has more than two symbols, the Huffman procedure requires a series of source reduction steps. In each step, we find and merge the two symbols with the smallest probabilities, which results in a new source with a reduced alphabet. The probability of the new symbol in the reduced alphabet is the sum of the probabilities of the two "merged" symbols from the previous alphabet. This procedure is continued until we reach a source with only two symbols, for which the codeword assignment is given by (18.6). 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, respectively. The following examples demonstrate this procedure.

Example: Symbol probabilities are powers of 2

Let the alphabet $\mathcal{A}$ consist of four symbols, shown in Table 18.2. The probabilities and the information content of the symbols in the alphabet are also listed in the table. Note that all symbol probabilities are powers of 2, and consequently the symbols have integer information values.

Table 18.2: An alphabet where the symbol probabilities are powers of 2.

Symbol	Probability	Information
a_1	0.50	1 bit
a_2	0.25	2 bits
a_3	0.125	3 bits
a_4	0.125	3 bits

The Huffman coding procedure is demonstrated for this alphabet in Table 18.3. The reduced alphabet in Step 1 is obtained by merging the symbols a_3 and a_4 in the original alphabet which have the lowest two probabilities. Likewise, the reduced alphabet in Step 2 is obtained by merging the two symbols with the lowest probabilities after Step 1. Since the reduced alphabet in Step 2 has only two symbols, we assign the codes 0 and 1 to these symbols in arbitrary order. Next, we assign codes to the reduced alphabet in Step 1. We recall that the symbol 2 in Step 2 is obtained by merging the symbols 2 and 3 in Step 1. Thus, we assign codes to symbols 2 and 3 in Step 1 by appending the code for symbol 2 in Step 2 by a zero and one in arbitrary order. The appended zero and one are shown by bold fonts in Table 18.3. Finally, the codes for the original alphabet are obtained in a similar fashion.

Table 18.3: Illustration of alphabet reduction.

Original Alphabet		Reduced Alphabet Step 1		Reduced Alphabet Step 2	
p	c	p	c	p	c
0.50	0	0.50	0	0.50	**0**
0.25	10	0.25	1**0**	0.50	**1**
0.125	11**0**	0.25	1**1**		
0.125	11**1**				

This procedure can alternatively be described by the tree diagram shown in Figure 18.3.

Observe that in this case, the average codeword length is

$$\bar{R} = 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75$$

and the entropy of the source is

$$H = -0.5 \ln 0.5 - 0.25 \ln 0.25 - 0.125 \ln 0.125 - 0.125 \ln 0.125 = 1.75$$

which is consistent with the result that Huffman coding achieves the entropy of the source when the symbol probabilities are powers of 2.

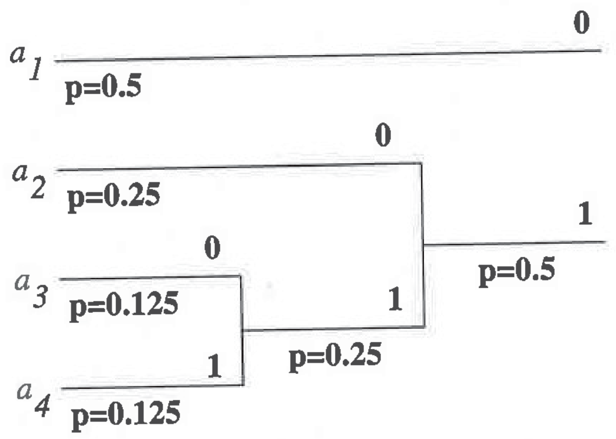


Figure 18.3: Tree-diagram for Huffman coding.

Next, we present an example in which the symbol probabilities are not powers of 2.

Example 2: Symbol probabilities are not powers of 2

The information content of each symbol is a real number, as shown in Table 18.4, when the probabilities of the symbols are not powers of 2.

Table 18.4: An alphabet with arbitrary symbol probabilities.

Symbol	Probability	Information
a_1	0.40	1.32 bits
a_2	0.25	2.00 bits
a_3	0.15	2.73 bits
a_4	0.15	2.73 bits
a_5	0.05	4.32 bits

Since the length of each codeword must be an integer, it is not possible to design codewords whose lengths are equal to the information of the respective symbols in this case. Huffman code design for the alphabet in Table 18.4 is shown in Table 18.5. It can be easily seen that for this example the average length of codewords is 2.15, and entropy of the source is 2.07.

Notice that Huffman codes are uniquely decodable, with proper synchronization, because no codeword is a prefix of another. For example, a received binary string

$$001101101110000\ldots$$

can be decoded uniquely as

$$a_3\ a_1\ a_2\ a_1\ a_2\ a_1\ a_1\ a_4\ \ldots$$

Table 18.5: Huffman coding when probabilities are not powers of 2.

Original Alphabet		Step 1		Step 2		Step 3	
p	c	p	c	p	c	p	c
0.40	1	0.40	1	0.40	1	0.60	**0**
0.25	01	0.25	01	0.35	0**0**	0.40	**1**
0.15	001	0.20	00**0**	0.25	0**1**		
0.15	000**0**	0.15	00**1**				
0.05	000**1**						

Huffman coding can also be used as a block coding scheme where we assign codewords to combinations of L symbols from the original alphabet at a time. Of course, this requires building a new block alphabet with all possible combinations of the L symbols from the original alphabet and computing their respective probabilities. Huffman codes for all possible combinations of the L symbols from the original alphabet can be formed using the above design procedure with the new block alphabet. Thus, Huffman coding is a block coding scheme, where we assign variable-length codes to fixed-length (L) blocks of symbols. The case $L = 1$ refers to assigning an individual codeword to each symbol of the original alphabet, as shown in the above two examples. It has been shown that for sources with memory, the coding efficiency improves as L gets larger, although the design of Huffman codes gets more complicated.

18.2.3 Arithmetic Coding

In arithmetic coding a one-to-one correspondence between the symbols of an alphabet $\mathcal{A}$ and the codewords does not exist. Instead, arithmetic coding assigns a single variable-length code to a source $\mathcal{X}$, composed of N symbols, where N is variable. The distinction between arithmetic coding and block Huffman coding is that in arithmetic coding the length of the input sequence, i.e., the block of symbols for which a single codeword is assigned, is variable. Thus, arithmetic coding assigns variable-length codewords to variable-length blocks of symbols. Because arithmetic coding does not require assignment of integer-length codes to fixed-length blocks of symbols, in theory it can achieve the lower bound established by the noiseless coding theorem.

Arithmetic coding associates a given realization of $\mathcal{X}$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, with a subinterval of $[0, 1)$ whose length equals the probability of the sequence $p(\mathbf{x})$. The encoder processes the input stream of symbols one by one, starting with $N = 1$, where the length of the subinterval associated with the sequence gets smaller as N increases. Bits are sequentially sent to the channel starting from the most significant bit towards the least significant bit as they are determined according to a procedure, which is presented in an algorithmic form in the following. At the end of the transmission, the transmitted bitstream is a uniquely decodable codeword representing the source, which is a binary number pointing to the subinterval associated with this sequence.

The Procedure

Consider an alphabet $\mathcal{A}$ that has M symbols a_i, $i = 1, \ldots, M$, with the probabilities $p(a_i) = p_i$, such that $p_1 + \ldots + p_M = 1$. The procedure starts with assigning each individual symbol in the alphabet a subinterval, within 0 to 1, whose length is equal to its probability. It is assumed that this assignment is known to the decoder.

1. If the first input symbol $x_1 = a_i$, $i = 1, \ldots, M$, then define the initial subinterval as $I_1 = [l_1, r_1) = [p_{i-1}, p_{i-1} + p_i)$, where $p_0 = 0$. Set $n = 1$, $L = l_1$, $R = r_1$, and $d = r_1 - l_1$.

2. Obtain the binary expansions of L and R as

$$L = \sum_{k=1}^{\infty} u_k 2^{-k}, \qquad \text{and} \qquad R = \sum_{k=1}^{\infty} v_k 2^{-k}$$

where u_k and v_k are 0 or 1.

Compare u_1 and v_1. If they are not the same, send nothing to the channel at this time, and go to step 3.

If $u_1 = v_1$, then send the binary symbol u_1, and compare u_2 and v_2. If they are not the same, go to step 3.

If $u_2 = v_2$, also send the binary symbol u_2, and compare u_3 and v_3, and so on, until the next two corresponding binary symbols do not match, at which time go to step 3.

3. Increment n, and read the next symbol. If the nth input symbol $x_n = a_i$, then subdivide the interval from the previous step as

$$I_n = [l_n, r_n) = [l_{n-1} + p_{i-1}d, l_{n-1} + (p_{i-1} + p_i)d).$$

Set $L = l_n$, $R = r_n$, and $d = r_n - l_n$, and go to step 2.

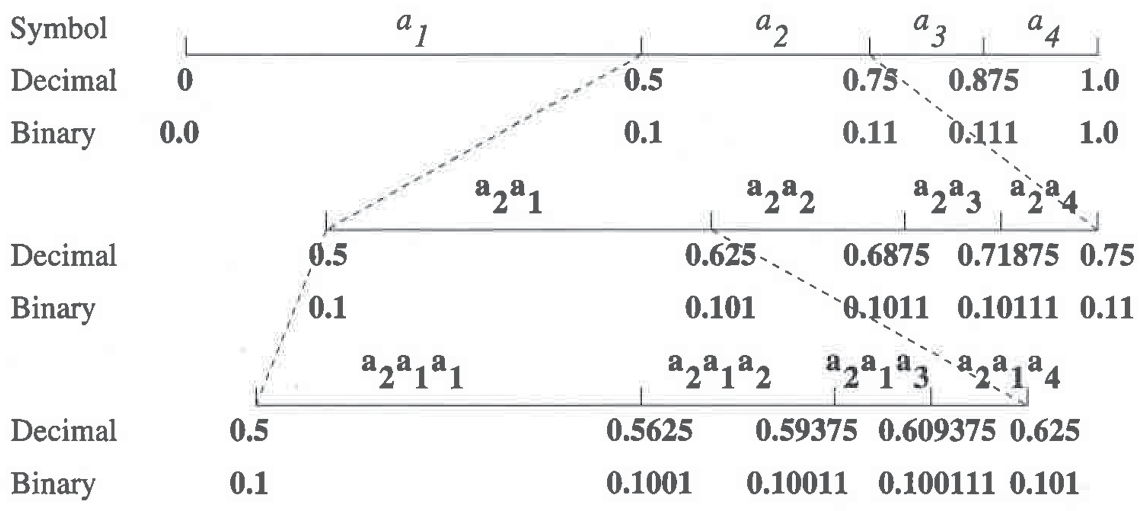
Note that the decoder may decode one binary symbol into several source symbols, or it may require several binary symbols before it can decode one or more source symbols. The arithmetic coding procedure is illustrated by means of the following example.

Example

Suppose we wish to determine an arithmetic code to represent a sequence of symbols,

$$a_2 \; a_1 \; a_3 \; \ldots$$

from the source shown in Table 18.2. Because we have four symbols in the alphabet, the interval from 0 to 1 is initially subdivided into 4, where the lengths of the subintervals are equal to 0.5, 0.25, 0.125 and 0.125, respectively. This is depicted in Figure 18.4.



Symbol
a_1
a_2
a_3
a_4
Decimal 0 0.5 0.75 0.875 1.0
Binary 0.0 0.1 0.11 0.111 1.0
a_2a_1
a_2a_2
a_2a_3
a_2a_4
Decimal 0.5 0.625 0.6875 0.71875 0.75
Binary 0.1 0.101 0.1011 0.10111 0.11
$a_2a_1a_1$
$a_2a_1a_2$
$a_2a_1a_3$
$a_2a_1a_4$
Decimal 0.5 0.5625 0.59375 0.609375 0.625
Binary 0.1 0.1001 0.10011 0.100111 0.101

Figure 18.4: Illustration of the concept of arithmetic coding.

The first symbol defines the initial interval as $I_1 = [0.5, 0.75)$, where the binary representations of the left and right boundaries are $L = 2^{-1} = 0.1$ and $R = 2^{-1} + 2^{-2} = 0.11$, respectively. According to step 2, $u_1 = v_1 = 1$; thus, 1 is sent to the channel. Noting that $u_2 = 0$ and $v_2 = 1$, we read the second symbol, a_1. Step 3 indicates that $I_2 = [0.5, 0.625)$, with $L = 0.10$ and $R = 0.101$. Now that $u_2 = v_2 = 0$, we send 0 to the channel. However, $u_3 = 0$ and $v_3 = 1$, so we read the third symbol, a_3. It can be easily seen that $I_3 = [0.59375, 0.609375)$, with $L = 0.10011$ and $R = 0.100111$. Note that $u_3 = v_3 = 0$, $u_4 = v_4 = 1$, and $u_5 = v_5 = 1$, but $u_6 = 0$ and $v_6 = 1$. At this stage, we send 011 to the channel, and read the next symbol. A reserved symbol usually signals the end of a sequence.

Let's now briefly look at how the decoder operates, which is illustrated in Figure 18.5. The first bit restricts the interval to $[0.5, 1)$. However,

Received Bit	Interval	Symbol
1	$[0.5, 1)$	-
0	$[0.5, 0.75)$	a_2
0	$[0.5, 0.609375)$	a_1
1	$[0.5625, 0.609375)$	-
1	$[0.59375, 0.609375)$	a_3
⋮	⋮	

Figure 18.5: The operation of the decoder.

three symbols are within this range; thus, the first bit does not contain sufficient information. After receiving the second bit, we have 10 which points to the interval $[0.5, 0.75)$. All possible combinations of two symbols pointing to this range start with a_2. Hence, we can now decode the first symbol as a_2. The information that becomes available after the receipt of each bit is summarized in Figure 18.5.

In practice, two factors cause the performance of the arithmetic encoder to fall short of the theoretical bound: the addition of an end-of-message indicator, and the use of finite precision arithmetic. Practical implementations of the arithmetic coder overcome the precision problem by a scaling and a rounding strategy.

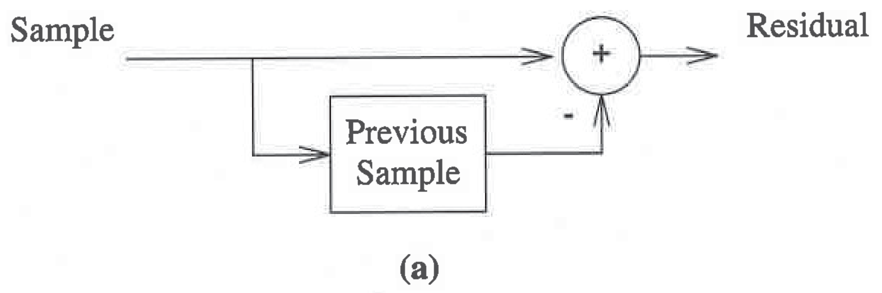
18.3 Lossless Compression Methods

Error-free coding is the only acceptable means of compression in some applications for various reasons. For example, in the transmission or archival of medical images lossy compression is not allowed for legal reasons. Recalling the elements of a compression system, lossless coding schemes do not employ a quantizer. They consist of a transformation, which generates symbols whose probability distribution is highly peaked, followed by an entropy coder. The transformation aims to minimize the entropy of its output, so that significant compression becomes possible by variable-length coding of the generated symbols.

In this section, we present three popular methods for lossless compression: i) Lossless predictive coding, where an integer predictive mapping is employed, followed by entropy coding of the integer prediction errors. ii) Run-length coding of bit-planes, where the image is decomposed into individual bit-planes (binary images), and the run-lengths of zeros and ones in these planes are entropy coded. iii) Ziv-Lempel coding, which is a deterministic coding procedure, where the input bit string is parsed into blocks of variable length to form a dictionary of blocks (symbols), each of which is represented by a fixed-length codeword. The achievable compression ratio using lossless coding methods ranges between 2:1 to 5:1, depending on the characteristics of the input image.

18.3.1 Lossless Predictive Coding

The first step in lossless predictive coding is to form an integer-valued prediction of the next pixel intensity to be encoded based on a set of previously encoded neighboring pixels. Then the difference between the actual intensity of the pixel and its prediction is entropy coded. Assuming each pixel is integer-valued, then the prediction errors are also integer-valued, which facilitates lossless compression. The block diagram of a simple predictor is shown in Figure 18.6, where the prediction is taken as the intensity of the previous pixel encoded. Some other commonly used integer predictors are shown in the following example.



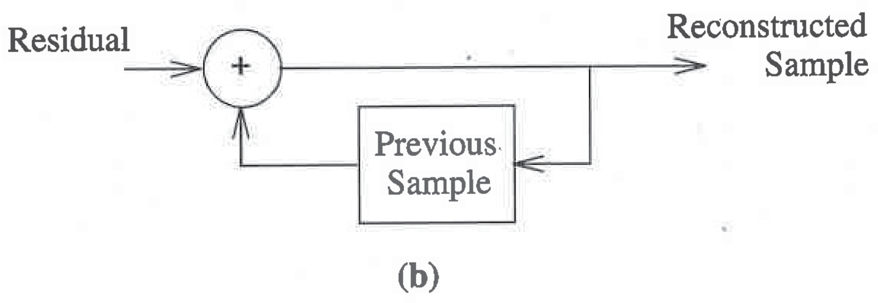


Figure 18.6: Block diagram of a) an encoder, and b) a decoder using a simple predictor.

Example: Integer prediction

In order for the decoder to be able to duplicate the prediction step, the prediction operation must be based on already-encoded pixels. In 2-D, such prediction models are called recursively computable. The support of a recursively computable predictor is shown in Figure 18.7, where the coefficients a, b, c, and d denote the intensities of the respective pixels.

Two simple predictors based on this model can be written as

$$\hat{x} = \text{int}\{(a+b)/2\} \tag{18.7}$$

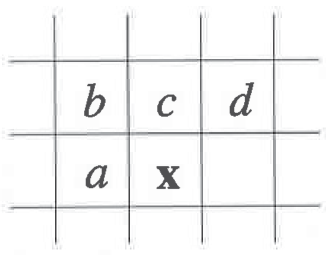


Figure 18.7: Prediction schemes.

and

$$\hat{x} = \text{int}\{(a + b + c + d)/4\} \tag{18.8}$$

where $\hat{x}$ denotes the predicted value for the pixel x. In both expressions, the prediction is rounded to the nearest integer to ensure an integer prediction error. We note that many other forms of prediction, such as edge-adaptive prediction, also exist.

If the input image intensity x has a dynamic range of (0,255), then the prediction error $x - \hat{x}$ has a theoretical dynamic range of (-255,255). The reader may have noticed that the predictive mapping in fact results in an expansion of the dynamic range of the signal to be encoded. However, inspection of the histogram (probability density) of the prediction error shows that it is highly peaked about 0, as compared to the histogram of the actual image intensities, as shown in Figure 18.8. Therefore, the prediction error always has much smaller entropy than the original intensity values, which implies that the prediction process removes a great deal of interpixel (statistical) redundancy.

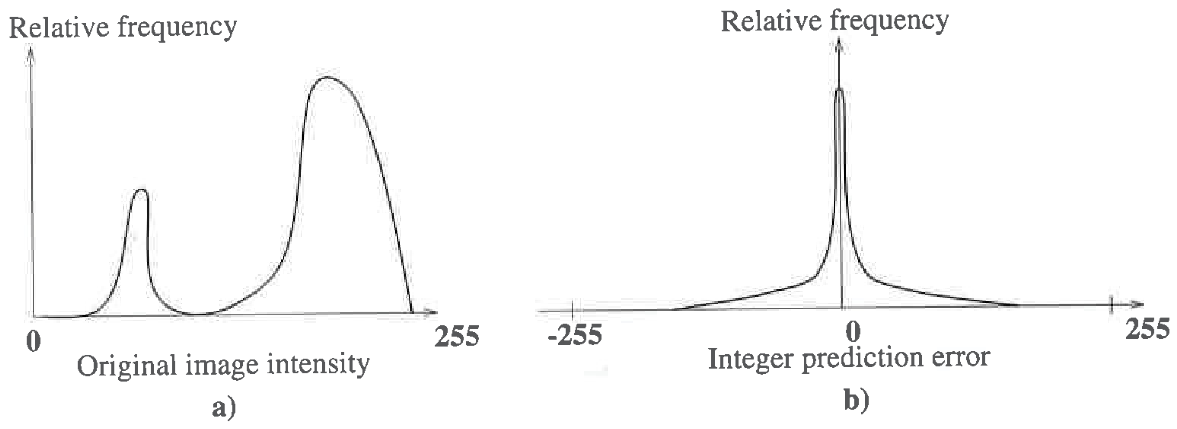


Figure 18.8: Histograms of a) the original image intensity and b) integer prediction error.

For lossless coding, every possible difference value in the range (-255,255) needs to be taken as a different symbol. Binary codewords for these symbols can be assigned by entropy coding, such as Huffman coding or arithmetic coding. However, because it is usually very costly to assign a different codeword to 513 different symbols, we usually assign a unique codeword to every difference value in the range (-15,16). In addition, codewords are assigned to a shift up (SU) symbol and a shift down (SD) symbol, which shifts a given difference value up or down by 32, respectively. Using these codewords, every possible difference value can be uniquely coded by using an appropriate number of SU or SD operators followed by a difference code in the range (-15,16). For example, a difference value of 100 can be represented by cascading the codes for the symbols SD, SD, SD, and 4. This scheme results in a slightly higher bitrate than does designing 513 different codes, but offers significant reduction in complexity. The probabilities of each of these symbols are estimated by analyzing a histogram of the prediction errors obtained from a training set of images.

18.3.2 Run-Length Coding of Bit-Planes

Bit-plane decomposition refers to expressing a multilevel (monochrome or color) image by a series of binary images, one for each bit used in the representation of the pixel intensities. Let the gray levels of an m-bit gray-scale image be represented as

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0 \qquad (18.9)$$

where a_i, $i = 0, \ldots, m-1$ are either 0 or 1. The zeroth-order bit-plane is generated by collecting the a_0 bits of each pixel, while the $(m-1)$st-order bit-plane contains the a_{m-1} bits. For example, for the case of an 8-bit image, a pixel in the most significant bit-plane is represented by a 1 if the corresponding pixel intensity is equal to or greater than 128. Observe that a binary image can be represented by a single bit-plane. Bit-plane decomposition is illustrated in Fig, 18.9.

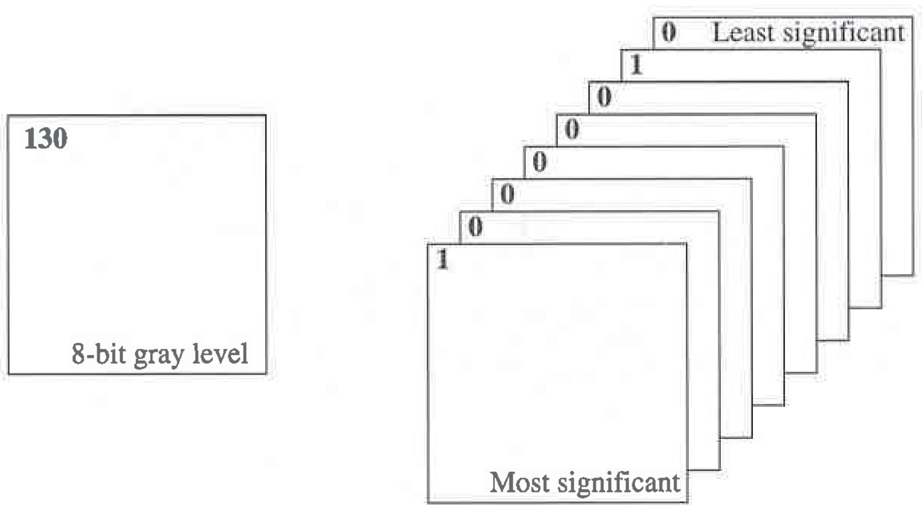


Figure 18.9: Bit-plane decomposition of an 8-bit image.

A disadvantage of the above bit-plane representation is that small changes in gray level, such as variations due to noise, may cause edges in all bit planes. For example, the binary representation for 127 is 01111111, and for 128, it is 10000000. To reduce the effect of such small gray-level variations in the bit-planes, we may choose to represent the pixel intensities by an m-bit Gray code where successive codewords differ only in one bit position. The m-bit Gray code $g_{m-1} \ldots g_2 g_1 g_0$ is given by

$$\begin{aligned} g_i &= a_i \oplus a_{i+1} \quad \text{for } 0 \leq i \leq m-2 \\ g_{m-1} &= a_{m-1} \end{aligned} \tag{18.10}$$

where $\oplus$ denotes the exclusive OR operation.

An effective approach to encode bit-planes is to employ run-length coding (RLC), which is often used for binary image compression. RLC algorithms can be classified as 1-D RLC and 2-D RLC. In 1-D RLC, the length of each contiguous group of 0's or 1's encountered in a left-to-right scan of a row of bit-plane is entropy coded. In most implementations, the length of each run is limited by the number of pixels in a line. For unique decodability, we need to establish a convention to either specify the first run of each row or assume that each row begins with a white run (i.e., the first symbol in each row is 1), whose run length may be zero. A different variable-length code is designed for each possible value of runs. Because the statistics of 0-runs and 1-runs are usually different, we design different codes for the white-runs and black-runs. Once again, the statistics of these runs are estimated from a training set of images. RLC has been adopted in the international standards for fax transmission, such as ITU (formerly CCITT) Group 3 and Group 4 codes. A more detailed discussion of 1-D and 2-D RLC is provided in Chapter 21.

18.3.3 Ziv-Lempel Coding

Ziv-Lempel coding is a block coding method which assigns fixed-length codes to variable-size blocks of input symbols by means of a table lookup using a dictionary of variable-length blocks of symbols [Ziv 94]. The input sequence of symbols is parsed into nonoverlapping blocks of variable length, whose length depends on the size of the blocks in the present dictionary, while updating the dictionary of blocks of symbols according to the following algorithm.

The length of the next block to be parsed, L, is defined to be equal to that of the longest word that is already in the dictionary. The initial dictionary is set equal to the list of all symbols in the alphabet $A = \{a_1, a_2, \ldots, a_M\}$. Thus, initially $L = 1$. If the next parsed block, w, is already in the dictionary, the encoder sends to the channel a fixed-length code for the index of this block. Before continuing parsing, w is concatenated with the next input symbol and added to the dictionary. If w is not in the dictionary, then the encoder sends a fixed-length code for the first $L-1$ symbols of w (which must be in the dictionary) and adds w to the dictionary. This process is repeated until the entire input sequence is coded. The Ziv-Lempel procedure is demonstrated by an example in the following.

Example

We demonstrate the Ziv-Lempel coding procedure for the case of a binary alphabet, that is, an alphabet which contains only two symbols, 0 and 1. The initial dictionary in this case contains only two symbols, 0 and 1, and $L = 1$. The first parsed symbol in the sequence to be coded is 0, as shown in Table 18.6. Clearly, 0 is in the dictionary; therefore, its code, 0, is transmitted. Then we check the next symbol, which is a 1, and append the block 01 in the dictionary. Now that $L = 2$, we read the next block of two symbols in the input sequence, which is 11. The block 11 is not in the dictionary, so it is added to the dictionary, and the code for the first $L - 1$ symbols in this block, which is a 1, is transmitted. The operation of the algorithm is illustrated in Table 18.6, which shows a list of input symbols, the blocks that enter the dictionary, the index that is coded by fixed-length coding, and the output of the coder.

Observe from Table 18.6 that Ziv-Lempel codes have the so-called "last-first" property which states that the last symbol of the most recent word added to the table is the first symbol of the next parsed block.

Table 18.6: An example for Ziv-Lempel coding.

Input	Dictionary	Index	Output
	0 1	0 1	
0	01	2	0
1	11	3	1
1	10	4	1
0	00	5	0
0 1	011	6	2
1 0	100	7	4
0 1	010	8	2
0 1 1	0110	9	6
0 0 0	. . .	. . .	

Ziv-Lempel coding is noiseless, does not require probabilities of the source symbols to be known or estimated, and is optimum in the limit of unbounded dictionary size. In practice, one places a bound on the dictionary size. Once this size limit is reached, the encoder can no longer add codewords and must simply use the existing dictionary. However, methods exist to adapt the dictionary to varying input characteristics.

Ziv-Lempel coding has been succesfully used for compression of binary data files, with a compression ratio of approximately 2.5:1. Indeed, it forms the basis of the "compress" utility in UNIX and the "arc" program for the PC environment.

18.4 Exercises

1. Suppose we have a discrete memoryless source with the alphabet $\mathcal{A}$ and the symbol probabilities $p(a_i)$ for $a_i \in \mathcal{A}$ specified by the following table:

Symbol	a_0	a_1	a_2	a_3	a_4	a_5
Probability	0.3	0.2	0.2	0.1	0.1	0.1

   a) Find the entropy of this source.

   b) Design a Huffman code for this source.

   c) Find the average codeword length.

   d) How good is this code?

2. Let $\mathbf{X}$ be a binary, Markov-2 source with the alphabet $\mathcal{A} = \{ 0, 1 \}$. The source is modeled by the conditional probabilities

$$\begin{array}{ll} P(0|0,0) = 0.7 & P(1|0,0) = 0.3 \\ P(0|0,1) = 0.6 & P(1|0,1) = 0.4 \\ P(0|1,0) = 0.4 & P(1|1,0) = 0.6 \\ P(0|1,1) = 0.3 & P(1|1,1) = 0.7 \end{array}$$

   a) What is the entropy of this source?

   b) Design a block Huffman code for $N = 3$.

   c) What is the average codeword length? What would be the average codeword length if we design a scalar Huffman code?

3. Suppose we have an 8-bit gray-level image. How would you estimate the entropy of this image:

   a) Assuming that it is a discrete memoryless source?

   b) Assuming that it is a Markov-1 source?

   Which one do you guess will be smaller? Why?

4. Assume that the histogram of the differential image shown in Figure 18.8 can be modeled by a Laplacian distribution, given by

$$p(e) = \frac{1}{\sqrt{2}\sigma_e} \exp\left\{\frac{-\sqrt{2}|e|}{\sigma_e}\right\}$$

with $\sigma_e = 15$. Write an expression for the entropy of the differential image. Estimate its value. What is the expected compression ratio (using lossless differential encoding)?

5. How do you compare arithmetic coding with block Huffman coding?

6. What is the primary motivation of using the Gray codes in bit-plane encoding?

Bibliography

[Ber 71] T. Berger, *Rate Distortion Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1971.

[Ger 92] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Norwell, MA: Kluwer, 1992.

[Jai 81] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, pp. 349–389, Mar. 1981.

[Jay 84] N. S. Jayant and P. Noll, *Digital Coding of Waveforms, Principles and Applications to Speech and Video*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[Jay 92] N. S. Jayant, "Signal compression: Technology targets and research directions," *IEEE Trans. Spec. Areas Comm*, vol. 10, pp. 796–819, June 1992.

[Net88] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, New York, NY: Plenum Press, 1988.

[Rab 91] M. Rabbani and P. Jones, *Digital Image Compression Techniques*, Bellingham, WA: SPIE Press, 1991.

[Ziv 77] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Info. Theory*, vol. IT-23, pp. 337–343, 1977.

[Ziv 94] J. Ziv, "On universal data compression - An intuitive overview," *J. Vis. Comm. Image Rep.*, vol. 5, no. 4, pp. 317–321, Dec. 1994.

Chapter 22

INTERFRAME COMPRESSION METHODS

Video compression is a key enabling technology for desktop digital video. Without compression, digital transmission of an NTSC color video, with 720 pixels $\times$ 480 lines, 8 bits/pixel per color, and 30 frames/sec, requires a transmission capacity of 248 Mbps. Likewise, an HDTV color video, with 1920 pixels $\times$ 1080 lines, 8 bits/pixel per color, and 30 frames/sec, needs a channel capacity of 1.5 Gbps. A super-35 format motion picture is usually digitized with 4096 pixels $\times$ 3112 lines and 10 bits/pixel per color. At a rate of 24 frames/sec, one second of a color movie requires approximately 9 Gbits (1.15 Gbytes) storage space. These data rates suggest that a CD with a storage capacity of about 5 Gbits can hold, without compression, approximately 20 sec of NTSC video, 3 sec of HDTV video, and one-half second of a movie. A typical data transfer rate for a CD-ROM device is about 1.5 Mbps (although faster devices that can transfer up to 4 Mbps are appearing in the market). Then full-motion NTSC quality video can be played back from a CD (1.2 Mbps for video and 0.3 Mbps for stereo audio) with 200:1 compression of the video signal. At 200:1 compression, a single CD can hold 3400 sec or about one hour of video. The HDTV signal needs be broadcast over a 6 MHz channel which can support about 20 Mbps. As a result, a compression ratio of 75:1 is required for broadcasting HDTV signals. For transmission using fiber optic networks or satellite links, similar compression is needed to transmit multiple channels.

An elementary approach to video compression would be to employ any of the still-frame compression techniques discussed in Chapters 18-21 on a frame by frame basis. However, the compression that can be achieved by such an approach will be limited because each frame is treated as an independent image. Interframe compression methods exploit the temporal redundancies due to similarity between neighboring frames, in addition to the spatial, spectral, and pyschovisual redundancies to provide superior compression efficiency. Note, however, that some application specific requirements, such as random access capability at all frames, may dictate the use of intraframe compression rather than interframe methods in some cases. In general, interframe compression methods take advantage of temporal redundancies through i) 3-D waveform coding strategies, which are based on statistical signal models, ii) motion-compensated (MC) coding strategies, which use elementary motion models, or iii) object/knowledge based coding strategies, which utilize more sophisticated scene models. Three-dimensional waveform coding strategy is discussed in Section 22.1. Various motion-compensated compression schemes are presented in Section 22.2. Note that all international video compression standards, covered in Chapter 23, utilize the MC transform coding strategy. Section 22.3 provides an overview of model-based coding methods, which are studied in detail in Chapter 24.

22.1 Three-Dimensional Waveform Coding

The simplest way to extend still-frame image compression methods to interframe video compression is to consider 3-D waveform coding schemes, which include 3-D transform and 3-D subband coding. These methods exploit spatio-temporal redundancies in a video source through statistical signal models.

22.1.1 3-D Transform Coding

Three-dimensional DCT coding is a straightforward extension of the 2-D DCT coding method, where the video is divided into $M \times N \times J$ blocks (M, N, and J denote the horizontal, vertical, and temporal dimensions of the block, respectively). The transform coefficients are then quantized subject to zonal coding or threshold coding, and encoded similar to 2-D transform coding of still-frame images. Since the DCT coefficients are closely related to the frequency content of the blocks, for temporally stationary blocks the DCT coefficients in the temporal direction will be close to zero, and will be truncated in a threshold coding scheme. For most blocks, the DCT coefficients will be packed towards the low spatial and temporal frequency zone. The 3-D DCT coding scheme has the advantage that it does not require a separate motion estimation step. However, it requires J frame stores both in the transmitter and receiver. Therefore, J is typically chosen as 2 or 4, to allow for practical hardware implementations. Observe that random access to video is possible once for every J frames, as shown in Figure 22.1.

A related 3-D waveform coding approach is the hybrid transform/DPCM coding method, which has been proposed to overcome the multiple frame-store requirement [Roe 77, Nat 77]. This is an extension of the 2-D hybrid DCT/DPCM coding concept proposed by Habibi [Hab 74] to 3-D, where a 2-D orthogonal transform is performed on each spatial block within a given frame. A bank of parallel DPCM coders, each tuned to the statistics of a specific DCT coefficient, is then applied to

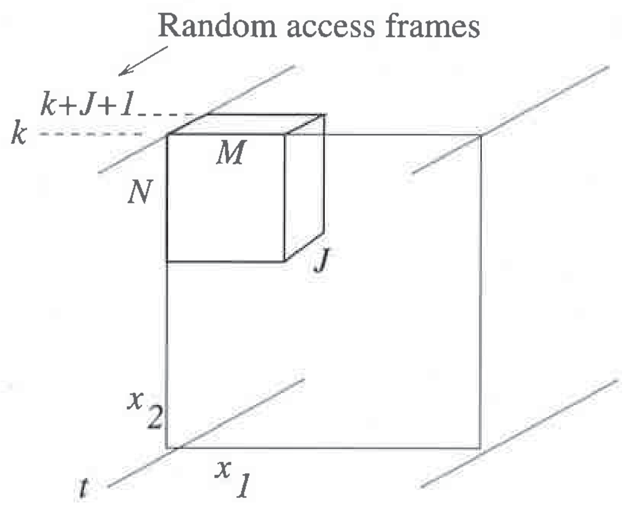


Figure 22.1: Three-dimensional transform coding.

the transform coefficients in the temporal direction. Thus, the differences in the respective DCT coefficients in the temporal direction are quantized and encoded, which eliminates the need for multiple frame-stores. This scheme generally requires adaptation of the DPCM quantizers to the temporal statistics of the 2-D DCT coefficients for results comparable to that of the 3-D DCT coding [Roe 77]. Note that neither 3-D DCT coding nor hybrid DCT/DPCM coding has been widely used in practical applications.

22.1.2 3-D Subband Coding

3-D subband coding, an extension of 2-D subband coding, has recently received increased attention [Vet 92, Bos 92, Luo 94] motivated by the following considerations: i) it is almost always free from blocking artifacts, which is a common problem with 3-D DCT and MC/DCT coding methods, especially at low bitrates, ii) unlike MC compression methods, it does not require a separate motion estimation stage, and iii) it is inherently scalable, both spatially and temporally. Scalability, which refers to availability of digital video at various spatial and temporal resolutions without having to decompress the entire bitstream, has become an important factor in recent years due to the growing need for storage and transmission of digital video that is comformable with various format standards. For example, standard TV and high-definition TV differ only in spatial resolution, whereas videophone systems offer lower spatial and temporal resolution.

In 3-D subband coding, the video is decomposed into various properly subsampled component video signals, ranging from a low spatial and temporal resolution component to various higher-frequency detail signal components. These various component video signals are encoded independently using algorithms adapted to the statistical and pyschovisual properties of the respective spatio-temporal frequency bands. Compression is achieved by appropriate quantization of the various components and entropy coding of the quantized values. Higher-resolution video, in both spatial and temporal coordinates, can be recovered by combining the decompressed low-resolution version with the decompressed detail components. Most 3-D subband decomposition schemes utilize 2- or 4-frame temporal blocks at a time due to practical implementation considerations.

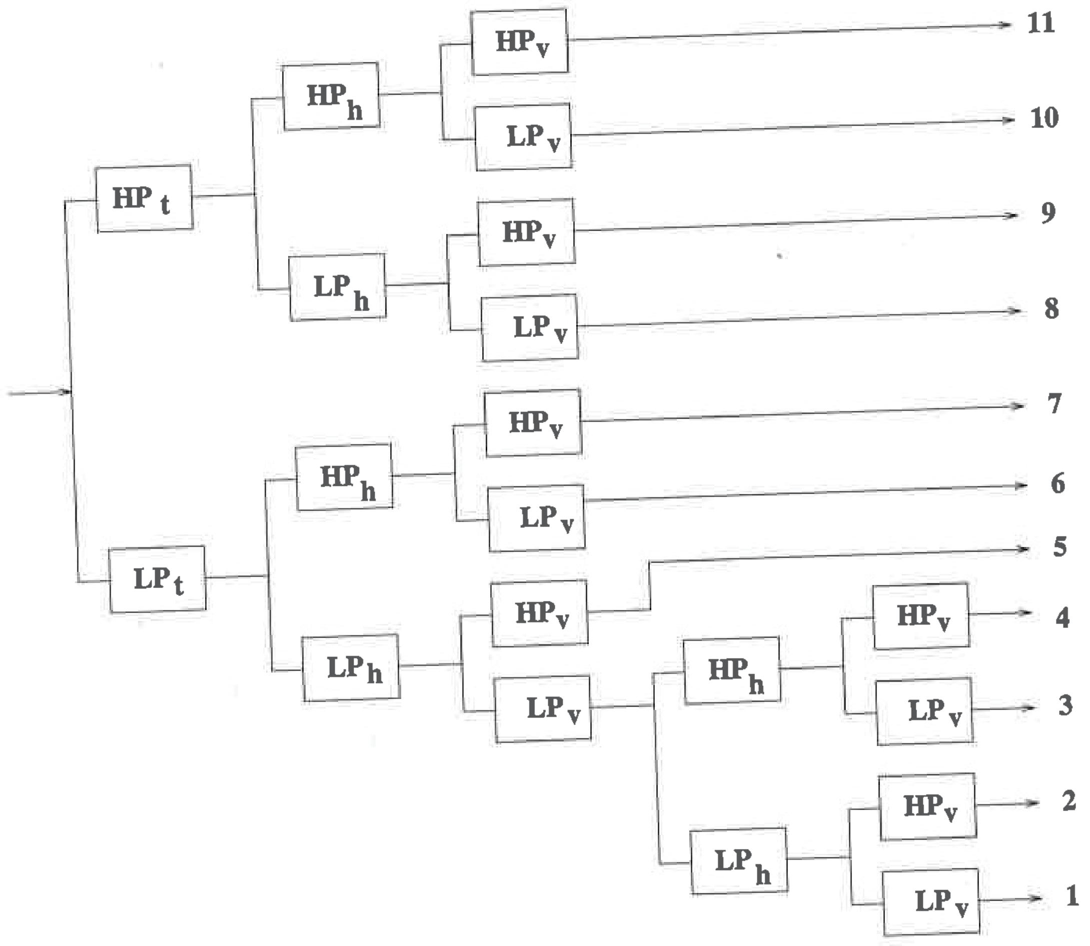


Figure 22.2: A typical 3-D subband decomposition.

An 11-band 3-D subband decomposition is illustrated in Figure 22.2. Typically, the temporal decomposition is based on a simple 2-tap Haar filterbank [Luo 94], which in the case of two frame blocks gives the average and the difference of the two frames for the low-pass (LP) and high-pass (HP) temporal components, respectively. This choice minimizes the number of frame-stores needed as well as the computational burden for the temporal decomposition. In the second stage, both the low and high temporal subbands are decomposed into low and high horizontal subbands, respectively. In the next stage, each of these bands are decomposed into low and high vertical subbands, as depicted in Figure 22.2. Subsequently, the low (temporal)-low (horizontal)-low (vertical) band is further decomposed into four spatial subbands to yield the 11-band decomposition. Note that longer filters can be applied for the spatial (horizontal and vertical) decompositions, since these filters can be operated in parallel and do not affect the frame-store requirements. To this effect, Luo *et al.* report using wavelet filterbanks for the spatial decompositions.

The resulting component video signals can be subsampled consistent with the spatio-temporal frequency characteristics of each band, which is illustrated in Figure 22.3. In this figure, the left- and right-hand side templates correspond to the low- and high-temporal-frequency components, respectively. For example, the component 1, which is subsampled by a factor of 2 in time and by a factor of 4 in each spatial direction, represents a spatio-temporally blurred version of a 2-frame block. As a result, a time-sequence composed of the component 1 for all 2 frame blocks constitutes a low-resolution (both spatially and temporally) version of the original video. Likewise, time sequences composed of the other components (e.g., 2-11) constitute auxilary video signals containing high-frequency detail information needed to reconstruct the original video from the low-resolution video. The reader is referred to [Vet 92, Bos 92, Luo 94] for various approaches offered to compress the individual component signals.

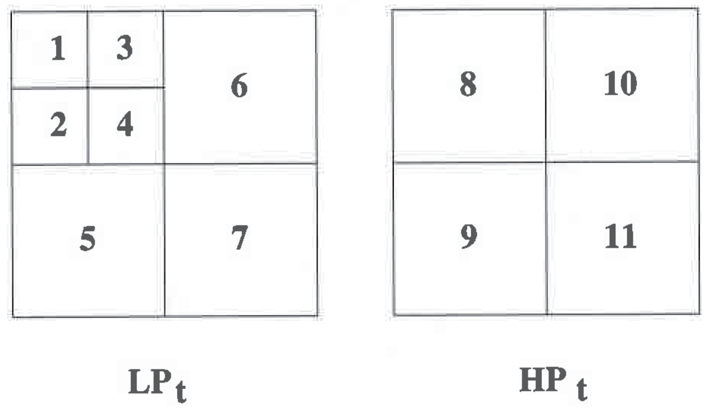


Figure 22.3: Representation of 3-D subband video.

The basic approach of 3-D DCT and subband coding is quite different from that of motion-compensated (MC) coding, which is presented next. MC techniques characterize the temporal correlation in a video by means of motion vectors rather than through the respective transform coefficients.

22.2 Motion-Compensated Waveform Coding

One of the earliest approaches in interframe image compression has been the so-called conditional replenishment technique, which is based on segmenting each frame into "changed" and "unchanged" regions with respect to the previous frame. Then information about the addresses and intensities of the pixels in the changed region would be transmitted using a bitrate that is matched to the channel rate. Intensities in the changed region are encoded by means of a DPCM method. Since the amount of changed information varies from frame to frame, the information to be transmitted needs to be buffered, and the quantization scheme is regulated according to the fullness of the buffer. A review of conditional replenishment algorithms can be found in [Has 72]. Note that conditional replenishment is a motion-detection based algorithm rather than a motion-compensated algorithm, since it does not require explicit estimation of the motion vectors.

The conditional replenishment was later extended to motion-compensated (MC) DPCM by encoding displaced frame difference values for those pixels in the changed area with respect to the previous frame [Has 78]. MC-DPCM yields more efficient compression provided that we can accurately estimate the displacement vectors. Most commonly used motion estimation methods in MC compression fall into pixel recursive algorithms or block-matching algorithms. Since these motion estimation methods were covered in detail in Chapters 5-8, here we assume that the motion vectors are known, and deal only with the encoding of the differential signal, also known as the temporal prediction error. We have already mentioned, in still-frame image compression, that transform coding and vector quantization both provide better compression efficiency compared with scalar DPCM. Thus, the next two sections are devoted to transform coding and vector quantization of the temporal prediction error.

22.2.1 MC Transform Coding

In MC transform coding, the temporal prediction error is 2-D transform coded by segmenting the displaced frame difference into blocks, and encoding the DCT coefficients of each block as in 2-D DCT coding. The temporal prediction aims at minimizing the temporal redundancy, while the DCT encoding makes use of the spatial redundancy in the prediction error. MC transform coding algorithms feature several modes to incorporate both progressive and interlaced inputs. These include intrafield, intraframe, and interfield and interframe prediction with or without motion compensation. In the two intra modes, the DCT blocks are formed by actual pixel intensities from a single field or from an entire frame. In the interfield and interframe modes the prediction is based on the previous field or frame, respectively. The MC transform coding is the basis of several world standards for video compression that are summarized in Table 22.1, with the possible exception of MPEG-4. The details of the MC transform coding will be covered in the next chapter where we discuss these world standards.

Table 22.1: World standards for video compression.

Standard	Description
H.261	ITU (CCITT) Expert Group on Visual Telephony; developed for ISDN applications at $p \times 64$ kbps ($p = 1, 2, \ldots, 30$); standardized in December 1990. *Application:* Videoconferencing and videophone using ISDN
MPEG	ISO Moving Picture Expert Group; PHASE 1: Storage and retrieval of digital video + audio at about 1.5 Mbps; draft finalized in June 1992. *Application:* Storage on CD-ROM and hard disk PHASE 2: Storage and retrieval of digital video + audio at about 10-20 Mbps; tech. spec. frozen in Mar. 1993. *Application:* Higher definition digital video including HDTV PHASE 4: Low-bitrate digital video + audio at below 64 kbps; embedded functionalities; just about to start. *Applications:* Videophone; database queries

The basic MC transform coding scheme employs block-based motion estimation and compensation. It has been argued that block motion models are not realistic for most image sequences, since moving objects hardly ever manifest themselves as rectangular blocks in the image plane. Recently, improved motion-compensation schemes, where more than a single motion vector per block are used without increasing the number of motion vectors to be transmitted, have been proposed [Orc 93] to circumvent this problem.

22.2.2 MC Vector Quantization

In MC-VQ the prediction error signal is encoded by vector quantization. The choice of using transform coding versus VQ for encoding the prediction error depends on several factors, including encoder and decoder complexity, cost of encoder and decoder, target bitrate, and real-time operation requirement. In MC transform coding, the complexity of the encoder and decoder is more or less symmetric, whereas in MC-VQ the encoder is significantly more complex than the decoder. Although the MC-VQ approach is capable of providing lower bitrates, a significant advantage the MC transform coding scheme enjoys is the availability of special-purpose hardware which enables encoding and decoding in real-time.

A typical MC-VQ scheme partitions pixels in the current frame into 4×4 or 8×8 blocks. First, a motion-detection test is applied to each block. The outcome of the motion-detection test determines one of three options: do nothing if no motion is detected, interframe VQ if motion is detected and the motion vector can be estimated with sufficient accuracy, or intraframe VQ if motion is detected but cannot be estimated within an acceptable accuracy limit. In the interframe VQ mode, the estimated motion vector, one for each block, and the displaced block difference are transmitted. The motion vectors for each block are generally DPCM encoded. The displaced block difference is VQ encoded using an interframe codebook. In the intraframe mode, the actual pixel intensities are VQ encoded using an intraframe codebook. The reader is referred to the literature for implementational details [Che 92, Mer 93]. It is also possible to replenish the intraframe and interframe codebooks at regular intervals to adapt the codebooks to the changing image and prediction error statistics [Gol 86].

22.2.3 MC Subband Coding

In MC subband coding the frame prediction error, also called the residual, is decomposed into 2-D subbands. The residual signal energy is typically unevenly distributed among the subbands which facilitates compression by simply truncating some of the subbands. Schemes have been proposed where the subbands are compressed using VQ [Mer 93]. Woods and Naveen [Woo 89] proposed integrating subband coding with hierarchical motion estimation. A motion-compensated subband decomposition technique which employs VQ has also been proposed [Nic 93].

22.3 Model-Based Coding

Three-dimensional and motion-compensated waveform coding provide satisfactory results with CIF images at bitrates over 1.5 Mbps. However, the quality of images that these techniques offer at very low bitrates, e.g. 10 kbps for videophone over existing telephone networks, is deemed unacceptable. In particular, decompressed images obtained by MC/DCT type methods generally suffer from blocking artifacts, which originate from the assumed translational block-motion model. To this effect, a variety of new motion-compensated coding schemes, generally known as model-based or analysis-synthesis coders, which are based on more realistic structural motion models, have recently been proposed for very-low-bitrate applications. An analysis-synthesis encoder can be characterized by the following steps:

- Image analysis: The frame to be encoded (present frame) is segmented into individually moving objects using the knowledge of the previously coded frame(s). Each object in the present frame is characterized by a set of shape (contour) and motion parameters.
- Image synthesis: The present frame is synthesized based on the estimated shape and motion parameters and the knowledge of the previously coded frame(s). The difference between the actual and the synthesized frame provides a measure of model compliance. Those regions where this difference is more than a threshold are labeled as "model failure" regions.
- Coding: The shape, motion and color (for the model failure regions) parameters are separately entropy encoded and transmitted.

Table 22.2: Overview of source models.

Method	Source Model	Encoded Information
MC/DCT	Translatory blocks	Motion vectors and color of blocks
Object-based	Moving unknown 2-D or 3-D objects	Shape, motion, and color of each moving object
Knowledge-based	Moving known objects	Shape, motion, and color of the known object
Semantic	Facial expressions	Action units

The image analysis and synthesis steps usually make extensive use of sophisticated computer vision and computer graphics tools, such as 3-D motion and structure estimation, contour modeling, and texture mapping [For 89]. Analysis-synthesis coding includes object-based, knowledge-based, and semantic coding approaches. Table 22.2 provides an overview of the source models used by various motion-compensated coding schemes, including the MC/DCT and several analysis-synthesis coding schemes. Note that MC/DCT coding, which forms the basis of several international video compression standards such as H.261, MPEG 1 and 2, is based on the overly simplistic source model of 2-D translatory blocks. Object-based, knowledge-based, and semantic coding methods are introduced in the following.

22.3.1 Object-Based Coding

Object-based coding (OBC) methods are based on structural image models derived from 3-D representation of a scene in terms of moving *unknown* (arbitrary) objects. The unknown objects can be treated as [Mus 89]: i) 2-D rigid or flexible objects with 2-D motion, ii) 2-D rigid objects with 3-D motion (affine or perspective mappings), or iii) 3-D rigid or flexible objects with 3-D motion.

Table 22.3: Expected bits per CIF frame (352 × 288) for different source models. r_s denotes bits/pixel for encoding the color information [Mus 93] (©1993 IEEE).

Source Model	Motion	Shape	Color
2-D rigid object 3-D motion	600	1300	15000 r_s
2-D flexible object 2-D motion	1100	900	4000 r_s
3-D rigid object 3-D motion	200	1640	4000 r_s

Efficiency of the source model can be measured by the data rate required for encoding the model parameters. The average bitrates for these object models for encoding a typical CIF format (386×288) frame are given in Table 22.3 [Mus 93]. It can be seen that the compression ratio increases as the complexity of the model increases.

22.3.2 Knowledge-Based and Semantic Coding

Knowledge-based coding deals with cases where we have some *a priori* information about the content of the video, since dealing with unknown arbitrary objects is, in general, quite difficult. For example, in videophone applications, head-and-shoulders-type images are common. Knowledge-based coding of facial image sequences using model-based techniques requires a common 3-D flexible wireframe model of the speaker's face to be present at both the receiver and transmitter sides. 3-D motion and structure estimation techniques are employed at the transmitter to track the global motion of the wireframe model and the changes in its structure from frame to frame. The estimated motion and structure (depth) parameters along with changing texture information are sent and used to synthesize the next frame in the receiver side. The *knowledge-based* approach can be summarized by the following source model and algorithm.

Source Model:

A moving known object which is characterized by
- A generic wireframe model to describe the shape of the object, e.g., the head and shoulders
- 3-D global motion parameters, e.g., to track the rotation and translation of the head from frame to frame
- 3-D local motion (deformation) parameters, e.g., to account for the motion of the eyes, lips, etc. due to facial expressions
- Color parameters to describe the model failure areas

Algorithm:

1. Detection of the boundaries of the object in the initial frame.
2. Adaptation of the generic wireframe model to the particular object, by proper scaling in the x_1, x_2, and x_3 directions.
3. Estimation of the 3-D global and local motion parameters, using 3-D motion and structure estimation methods.
4. Synthesis of the next frame.
5. Determination of model failure areas.
6. Coding of motion and color parameters.

A detailed presentation of knowledge-based coding schemes can be found in Chapter 24. While the knowledge-based scheme is generally successful in tracking the global motion of the head, the estimation of the local motion of the eyes, lips, and so on due to facial expressions is usually difficult. Errors in local motion estimation generally result in several model failure regions. *Semantic coding* is a subset of knowledge-based methods, which attempts to model the local motion in terms of a set of facial action units. After compensating for the global motion, the encoder estimates a combination of action units that best fits a given facial expression and encodes their indices along with the global motion and shape parameters. Semantic coding can then be summarized by the following source model and algorithm.

Source Model:

Facial expressions of a head, described by
- A wireframe model
- A limited set of action units

Coding Algorithm:

1. Detection of known object boundaries.
2. Adaptation of the wireframe model.
3. Estimation/compensation of the global motion.
4. Estimation of action units (AUs).
5. Synthesis of next frame.
6. Determination of model failure areas.
7. Coding of AUs, global motion, and color parameters.

In knowledge-based and semantic coding, we can effectively decrease the bitrate by conveying the information in a head-and-shoulders-type sequence in terms of a set of global motion and facial-action-unit parameters. Clearly, knowledge-based coding poses a trade-off between compression efficiency and the generality of the compression algorithm.

22.4 Exercises

1. Discuss the relative advantages and disadvantages of 3-D waveform coding versus motion-compensated coding methods.

2. It is well-known that the MC-DCT approach suffers from blocking artifacts, especially at low bitrates. This is mainly due to insufficiency of the assumed translational block motion model. There are, in general, two approaches to address this problem: i) improving the motion model by using spatial transformations (generalized block motion), and ii) 3-D subband coding. Evaluate the relative merits and demerits of both approaches, especially in terms of motion artifacts versus spatio-temporal blurring.

3. The generalized block motion model, coupled with a motion segmentation algorithm, results in a 2-D object-based image compression scheme. Compare 2-D versus 3-D object-based methods for image compression. What advantages do 3-D object-based methods offer over 2-D methods, if any?

Bibliography

[Bos 92] F. Bosveld, R. L. Lagendijk, and J. Biemond, "Compatible spatio-temporal subband encoding of HDTV," *Signal Proc.*, vol. 28, pp. 271–290, Sep. 1992.

[Che 92] H.-H. Chen, Y.-S. Chen, and W.-H. Hsu, "Low-rate sequence image coding via vector quantization," *Signal Proc.*, vol. 26, pp. 265–283, 1992.

[For 89] R. Forchheimer and T. Kronander, "Image coding-from waveforms to animation," *IEEE Trans. Acoust. Speech Sign. Proc.*, vol. 37, no. 12, pp. 2008–2023, Dec. 1989.

[Gha 91] H. Gharavi, "Subband coding of video signals," in *Subband Image Coding*, J. W. Woods, ed., Norwell, MA: Kluwer, 1991.

[Gol 86] M. Goldberg and H. Sun, "Image sequence coding using vector quantization," *IEEE Trans. Comm.*, vol. 34, pp. 703–710, July 1986.

[Hab 74] A. Habibi, "Hybrid coding of pictorial data," *IEEE Trans. Comm.*, vol. 22, no. 5, pp. 614–624, May 1974.

[Hab 81] A. Habibi, "An adaptive strategy for hybrid image coding," *IEEE Trans. Comm.*, vol. 29, no. 12, pp. 1736–1740, Dec. 1981.

[Has 72] B. G. Haskell, F. W. Mounts, and J. C. Candy, "Interframe coding of videotelephone pictures," *Proc. IEEE*, vol. 60, pp. 792–800, July 1972.

[Has 78] B. G. Haskell, "Frame replenishment coding of Television," in *Image Transmission Techniques*, W. K. Pratt, ed., Academic Press, 1978.

[Hot 90] M. Hotter, "Object oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Proc: Image Comm.*, vol. 2, no. 4, pp. 409–428, Dec 1990.

[Luo 94] J. Luo, C. W. Chen, K. J. Parker, and T. S. Huang, "Three dimensional subband video analysis and synthesis with adaptive clustering in high-frequency subbands," *Proc. IEEE Int. Conf. Image Processing*, Austin, TX, Nov. 1994.

[Mer 93] R. M. Mersereau, M. J. T. Smith, C. S. Kim, F. Kossentini, and K. K. Truong, "Vector quantization for video data compression," in *Motion Analysis and Image Sequence Processing*, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Mus 89] H. G. Musmann, M. Hotter, and J. Ostermann, "Object-oriented analysis synthesis coding of moving images," *Signal Proc: Image Comm.*, vol. 1, pp. 117–138, 1989.

[Mus 93] H. G. Musmann, "Object-oriented analysis-synthesis coding based on source models of moving 2D and 3D objects," *Proc. IEEE Int. Conf. Acoust. Speech and Sign. Proc.*, Minneapolis, MN, April 1993.

[Nat 77] T. R. Natarajan and N. Ahmed, "On interframe transform coding," *IEEE Trans. Comm.*, vol. 25, no. 11, pp. 1323–1329, Nov. 1977.

[Nic 93] A. Nicoulin, M. Mattavelli, W. Li, A. Basso, A. C. Popat, and M. Kunt, "Image sequence coding using motion compensated subband decomposition," in *Motion Analysis and Image Sequence Processing*, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Orc 93] M. Orchard, "Predictive motion-field segmentation for image sequence coding," *IEEE Trans. Circ. and Syst: Video Tech.*, vol. 3, pp. 54–70, Feb. 1993.

[Plo 82] A. Ploysongsang and K. R. Rao, "DCT/DPCM processing of NTSC composite video signal," *IEEE Trans. Comm.*, vol. 30, no. 3, pp. 541–549, Mar. 1982.

[Pur 91] A. Puri and R. Aravind, "Motion-compensated video coding with adaptive perceptual quantization," *IEEE Trans. Circ. Syst. Video Tech.*, vol. 1, pp. 351–361, 1991.

[Roe 77] J. A. Roese, W. K. Pratt, and G. S. Robinson, "Interframe cosine transform image coding," *IEEE Trans. Comm.*, vol. 25, no. 11, pp. 1329–1339, Nov. 1977.

[Sab 84] S. Sabri and B. Prasada, "Coding of broadcast TV signals for transmission over satellite channels," *IEEE Trans. Comm.*, vol. 32, no. 12, pp. 1323–1330, Dec. 1984.

[Saw 78] K. Sawada and H. Kotera, "32 Mbit/s transmission of NTSC color TV signals by composite DPCM coding," *IEEE Trans. Comm.*, vol. 26, no. 10, pp. 1432–1439, Oct. 1978.

[Vet 92] M. Vetterli and K. M. Uz, "Multiresolution coding techniques for digital television: A review," *Multidim. Syst. Signal Proc.*, vol. 3, pp. 161–187, 1992.

[Woo 89] J. W. Woods and T. Naveen, "Subband encoding of video sequences," *Proc. SPIE Visual Comm. Image Proc. IV*, vol. 1199, pp. 724–732, 1989.

Chapter 23

VIDEO COMPRESSION STANDARDS

Standardization of compressed digital video formats facilitates manipulation and storage of full-motion video as a form of computer data, and its transmission over existing and future computer networks, or over terrestrial broadcast channels. Envisioned areas of application for digital video compression standards include all-digital TV, videoconferencing, videophone, video mail, multimedia stations, digital movies, video games, other forms of entertainment, and education. In this chapter, we discuss the international video compression standards for videoconferencing (H.261), multimedia (MPEG-1), and all-digital TV (MPEG-2) applications.

23.1 The H.261 Standard

ITU (CCITT) Recommendation H.261 is a video compression standard developed to facilitate videoconferencing and videophone services over the integrated services digital network (ISDN) at $p \times 64$ kbps, $p = 1, \ldots, 30$. For example, 64 kbps ($p = 1$) may be appropriate for a low quality videophone service, where the video signal can be transmitted at a rate of 48 kbps, and the remaining 16 kbps is used for the audio signal. Videoconferencing services generally require higher image quality, which can be achieved with $p \geq 6$, i.e., at 384 kbps or higher. Note that the maximum available bitrate over an ISDN channel is 1.92 Mbps ($p = 30$), which is sufficeint to obtain VHS-quality (or better) images.

ITU (CCITT) Recommendation H.261 has emerged as a result of studies performed within the European Project COST (CoOperation in the field of Scientific and Technical research) 211bis during the period 1983-1990. In 1985, the COST 211bis videoconference hardware subgroup developed an initial codec operating at bit rates of $n \times 384$ kbps, $n = 1, \ldots, 5$, which was adopted in 1987 as ITU (CCITT) Recommendation H.120. Later, it became clear that a single standard can cover all ISDN rates, $p \times 64$ kbps, $p = 1, \ldots, 30$. The specifications of such a codec were completed in 1989, and the corresponding Recommendation H.261 was adopted by ITU (CCITT) in 1990.

In addition to forming a basis for the later video compression standards such as MPEG-1 and MPEG-2, the H.261 standard offers two important features: i) It specifies a maximum coding delay of 150 msec. because it is mainly intended for bidirectional video communication. It has been determined that delays exceeding 150 msec. do not give the viewer the impression of direct visual feedback. ii) It is amenable to low-cost VLSI implementation, which is rather important for widespread commercialization of videophone and teleconferencing equipment. The important aspects of the H.261 standard are summarized below. Further details can be found in [Lio 90, Lio 91, CCI 90].

23.1.1 Input Image Formats

To permit a single recommendation for use in and between regions using 625- and 525-line TV standards, the H.261 input picture format is specified as the so-called Common Intermediate Format (CIF). For lower-bitrate applications, a smaller format, QCIF, which is one-quarter of the CIF, has been adopted. The specifications of the CIF and QCIF formats are listed in Table 23.1, where the numbers in the parenthesis denote the modified specifications so that all four numbers are integer multiples of 8. Note that, at 30 frames/s, the raw data rate for the CIF is 37.3 Mbps, and for QCIF it is 9.35 Mbps. Even with QCIF images at 10 frames/s, 48:1 compression is required for videophone services over a 64 kbps channel. CIF images may be used when $p \geq 6$, that is, for videoconferencing applications. Methods for conversion to and from CIF/QCIF are not subject to recommendation.

Table 23.1: H.261 input image formats.

	CIF	QCIF
Number of active pels/line		
Lum (Y)	360 (352)	180 (176)
Chroma (U,V)	180 (176)	90 (88)
Number of active lines/pic		
Lum (Y)	288	144
Chroma (U,V)	144	72
Interlacing	1:1	1:1
Temporal rate	30, 15, 10, or 7.5	30, 15, 10 or 7.5
Aspect ratio	4:3	4:3

23.1.2 Video Multiplex

The video multiplex defines a data structure so that a decoder can interpret the received bit stream without any ambiguity. The video data is arranged in a hierarchical structure consisting of a picture layer, which is divided into several group-of-blocks (GOB) layers. Each GOB layer in turn consists of macroblocks (MB), which are made of blocks of pixels. Each layer has a header identifying a set of parameters used by the encoder in generating the bitstream that follows.

A macroblock is the smallest unit of data for selecting a compression mode. (The choices for the modes of compression are described below.) It consists of four 8 ×8 (i.e., 16 pixels by 16 lines) of Y (luminance) and the spatially corresponding 8 ×8 U and V (chrominance) blocks. Since the chrominance channels are subsampled in both directions, there is only one U and one V block for every four luminance blocks. The composition of a macroblock is shown in Figure 23.1.

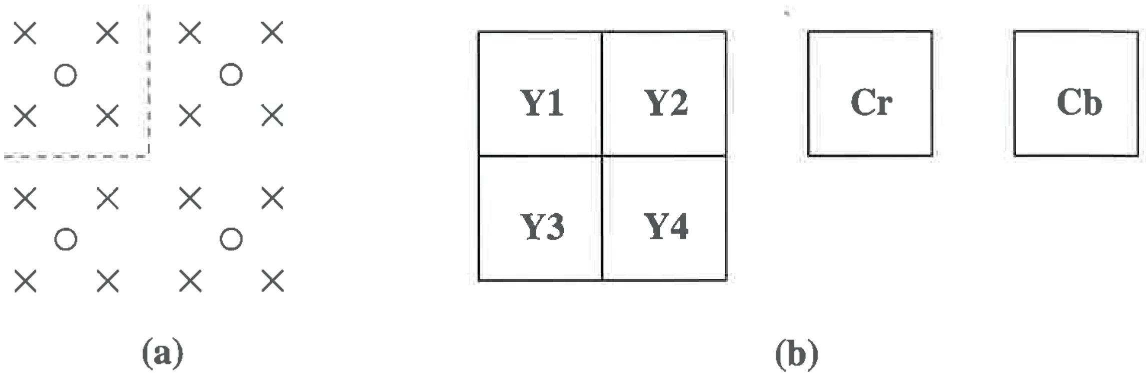


Figure 23.1: a) Positioning of luminance and chrominance pixels; b) the composition of a macroblock.

The GOB layer is always composed of 33 macroblocks, arranged as a 3 × 11 matrix, as depicted in Figure 23.2. Note that each MB has a header, which contains a MB address and the compression mode, followed by the data for the blocks.

Finally, the picture layer consists of a picture header followed by the data for GOBs. The picture header contains data such as the picture format (CIF or QCIF)

Table 23.2: The composition of the picture layer.

Format	Number of GOB in a frame	Number of MB in a GOB	Total number of MB in a frame
CIF	12	33	396
QCIF	3	33	99

directions are larger than the corresponding Nyquist intervals, aliasing may result. It is essential that no anti-alias filtering be used prior to sampling in order to achieve superresolution, which follows from the discussion in Section 16.1.

17.1.2 Discrete-Discrete Model

Next, we relate a set of low-resolution observations, $g_k(n_1, n_2)$, to the desired high-resolution frame(s) to be reconstructed, which are defined as

$$s_i(m_1, m_2) = s_c(x_1, x_2, t)|_{[x_1\ x_2\ t]^T = \mathbf{V}_h [m_1\ m_2\ i]^T} \tag{17.8}$$

where $\mathbf{V}_h$ is the sampling matrix of the high-resolution sampling grid.

Let's assume that the high-resolution video $s_i(m_1, m_2)$ is sampled above the Nyquist rate, so that the continuous intensity pattern is more or less constant within each high-resolution pixel (cells depicted in Figure 17.4). Then, for any frame i within the temporal span of the motion trajectory passing through (n_1, n_2, k), we have, from (17.7) and (17.5),

$$g_k(n_1, n_2) \approx s_i(m_1, m_2) \int\int h(u_1, u_2, \tau; x_1, x_2, t) du_1 du_2 \tag{17.9}$$

where $[x_1\ x_2\ t]^T = \mathbf{V}_l[n_1\ n_2\ k]^T$, $[u_1\ u_2\ \tau]^T = \mathbf{V}_h[m_1\ m_2\ i]^T$, and $(u_1, u_2) = c(\tau; x_1, x_2, t)$. Next, we define

$$h_{ik}(m_1, m_2; n_1, n_2) = \int\int h(u_1, u_2, \tau; x_1, x_2, t) du_1 du_2 \tag{17.10}$$

to arrive at our discrete-input (high-resolution video), discrete-output (observed low-resolution video) model, given by

$$g_k(n_1, n_2) = \sum_{m_1} \sum_{m_2} s_i(m_1, m_2) h_{ik}(m_1, m_2; n_1, n_2) + v_k(n_1, n_2) \tag{17.11}$$

where the support of the summation over the high-resolution grid (m_1, m_2) at a particular low-resolution sample (n_1, n_2, k) is depicted in Figure 17.4. The size of the support in Figure 17.4 depends on the relative velocity of the scene with respect to the camera, the size of the support of the low-resolution sensor PSF $h_a(x_1, x_2)$ (depicted by the solid line, assuming no out-of-focus blur) with respect to the high resolution grid, and whether there is any out-of-focus blur. Because the relative positions of low- and high-resolution pixels in general vary from pixel to pixel, the discrete sensor PSF is space-varying.

The model (17.11) establishes a relationship between any high-resolution frame i and observed low-resolution pixels from all frames k which can be connected to the frame i by means of a motion trajectory. That is, each low-resolution observed pixel (n_1, n_2, k) can be expressed as a linear combination of several high-resolution pixels from the frame i, provided that (n_1, n_2, k) is connected to frame i by a motion trajectory. Note that both the continuous-discrete and discrete-discrete models are invalid in case of occlusion. We assume that occlusion regions can be detected *a priori* using a proper motion estimation/segmentation algorithm.

dbd becomes a block difference (bd). Various compression modes are discussed in detail in the following.

3) Process each MB to generate a header followed by a data bitstream that is consistent with the compression mode chosen.

The motion estimation method, the criterion for the choice of a mode, and whether to transmit a block or not are not subject to recommendation. They are left as design parameters for a particular implementation. The choices that are presented below are those that are used in the Reference Model 8 (RM8), which is a particular implementation [RM8 89].

Compression Modes

Selecting a compression mode requires making several decisions, for each MB, including: i) should a motion vector be transmitted, ii) inter vs. intra compression, and iii) should the quantizer stepsize be changed? All possible compression modes are listed in Table 23.3, where "Intra," "Inter," "Inter+MC," and "Inter+MC+FIL" denote intraframe, interframe with zero motion vector, motion-compensated interframe, and motion-compensated interframe with loop-filtering, respectively. MQUANT stands for the quantizer step size, and an "x" in this column indicates that a new value for MQUANT will be transmitted. MVD stands for motion vector data, CBP for coded block pattern (a pattern number signifying those blocks in the MB for which at least one transform coefficient is transmitted), and TCOEFF for the transform coefficients that are encoded. Finally, VLC is the variable-length code that identifies the compression mode in the MB header.

In order to select the best compression mode, at each MB, the variance of the original macroblock, the macroblock difference (bd), and the displaced macroblock difference (dbd) with the best motion vector estimate are compared as follows:

Table 23.3: H.261 compression modes [CCI 90].

Prediction	MQUANT	MVD	CBP	TCOEFF	VLC
Intra				x	0001
Intra	x			x	0000 001
Inter			x	x	1
Inter	x		x	x	0000 1
Inter+MC		x			0000 0000 1
Inter+MC		x	x	x	0000 0001
Inter+MC	x	x	x	x	0000 0000 01
Inter+MC+FIL		x			001
Inter+MC+FIL		x	x	x	01
Inter+MC+FIL	x	x	x	x	0000 01

a) If the variance of *dbd* is smaller than *bd* as determined by a threshold, then the "Inter+MC" mode is selected, and the motion vector MVD needs to be transmitted as side information. The difference of the motion vector between the present and previous macroblocks is VLC coded for transmission. Observe from Table 23.3 that the transmission of the prediction error characterized by the DCT coefficients TCOEFF is optional.

b) Otherwise, a motion vector will not be transmitted, and a decision needs to be made between the "Inter" and "Intra" modes. If the original MB has a smaller variance, then the "Intra" mode is selected, where the DCT of each 8×8 block of the original picture elements are computed; otherwise, the "Inter" mode (with zero displacement vector) is selected. In both "Inter" and "Inter+MC" blocks the respective difference blocks (also called as the prediction error) are DCT encoded. The reader is referred to [CCI 90] for an exact specification of the various decision functions.

For MC blocks, the prediction error can be chosen to be modified by a 2-D spatial filter for each 8×8 block before the transformation by choosing the "Inter+MC+FIL" mode. The filter is separable, obtained by cascading two identical 1-D FIR filters. The coefficients of the 1-D filter are given by 1/4, 1/2, 1/4 except at the block boundaries should one of the taps fall outside the block, where they are modified as 0, 1, 0.

Thresholding

A variable thresholding is applied before quantization to increase the number of zero coefficients. The accuracy of the coefficients is 12 bits with dynamic range in [-2048,2047]. The flowchart of the variable thresholding algorithm is shown in Figure 23.3. In the flowchart, "g" refers to the quantizer step size, ξ is current value of the threshold, and "coef" is the value of the DCT coefficient. The variable thresholding scheme is demonstrated below by means of an example.

Quantization

The coefficient values after variable thresholding are quantized using a uniform quantizer. Within a macroblock the same quantizer is used for all coefficients except for the intra DC coefficient. The same quantizer is used for both luminance and chrominance coding. The intra DC coefficient is linearly quantized with a stepsize of 8 and no dead zone. Other coefficients are also linearly quantized, but with a central dead-zone about zero and with a stepsize MQUANT of an even value in the range 2 to 62 (31 stepsizes are allowed). This stepsize is controlled by the buffer state. To prevent overflow/underflow, a clipping of the image pixel values to within the range [0,255] is performed in both the encoder and decoder loops.

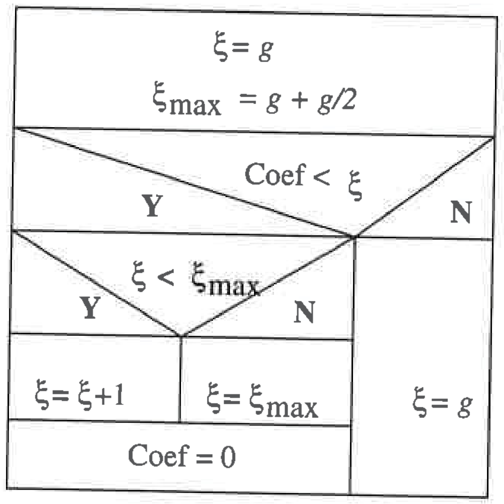


Figure 23.3: Flowchart of the thresholding algorithm [RM8 89].

Example: (with permission of PTT Research Labs., The Netherlands.)

An example of the variable thresholding and quantization is shown in Table 23.4 with $g = 32$. In this example, the threshold is incremented starting from 32 to 38, at which point the coefficient 40 is more than the value of the variable threshold. Thus, the threshold is reset to 32.

Table 23.4: An example to demonstrate variable thresholding [RM8 89].

Coefficients	50	0	0	0	33	34	0	40	33	34	10	32
Threshold ξ	32	32	33	34	35	36	37	38	32	32	32	33
New Coefficient	50	0	0	0	0	0	0	40	33	34	0	0
Quantized value	48	0	0	0	0	0	0	48	48	48	0	0

Coding

In order to increase the coding efficiency, the quantized coefficients are zigzag scanned, and events are defined which are then entropy coded. The events are defined as a combination of a run length of zero coefficients preceding a nonzero coefficient, and the level (value) of the nonzero coefficient, that is, $EVENT = (RUN, LEVEL)$. This is illustrated by an example [RM8 89].

Example: (with permission of PTT Research Labs., The Netherlands.)

For the block of transform coefficients shown in Figure 23.4, the events that represent this block, following a zigzag scan, are

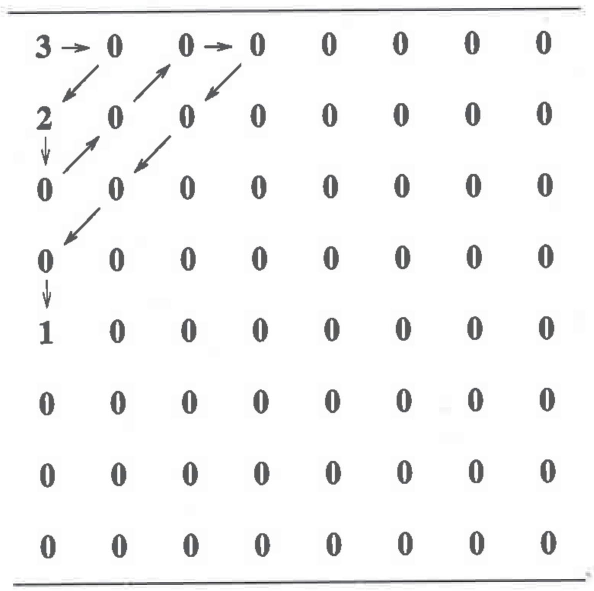


Figure 23.4: Illustration of the events [RM8 89].

(0,3) (1,2) (7,1) EOB

For 8 × 8 blocks, we have $0 \leq RUN < 64$, and the dynamic range of the nonzero coefficients, $LEVEL$, is [-128g, 127g], where g is the quantizer step size. These events are VLC coded. The VLC tables are specified in [CCI 90].

Rate/Buffer Control

Several parameters can be varied to control the rate of generation of coded video data. They are: i) processing prior to source coder, ii) the quantizer (step size), iii) block significance criterion, and iv) temporal subsampling (performed by discarding complete pictures). The proportions of such measures in the overall control strategy are not subject to recommendation.

In most implementations the quantizer step size is adjusted based on a measure of buffer fullness to obtain the desired bitrate. The buffer size is chosen not to exceed the maximum allowable coding delay (150 msec.) which also imposes a limit on the maximum bit count that can be generated by a single frame. Furthermore, the block significance criterion, which is employed to decide whether to transmit any data for a block, can be varied according to the desired bitrate.

Forced updating is used to control the accumulation of errors due to mismatch of the inverse DCT implementation at the encoder and decoder. The allowable bounds on the accuracy of the IDCT is specified in the standard. Forced updating refers to use of intra mode for a macroblock at least once every 132 times it is transmitted.

23.2 The MPEG-1 Standard

MPEG-1 is an ISO standard that has been developed for storage of CIF format video and its associated audio at about 1.5 Mbps on various digital storage media such as CD-ROM, DAT, Winchester disks, and optical drives, with the primary application perceived as interactive multimedia systems. The MPEG-1 algorithm is similar to that of H.261 with some additional features. The quality of MPEG-1 compressed/decompressed CIF video at about 1.2 Mbps (video rate) has been found to be similar (or superior) to that of VHS recorded analog video.

MPEG committee started its activities in 1988. Definition of the video algorithm (Simulation Model 1) was completed by September 1990. MPEG-1 was formally approved as an international standard by late 1992. Work is currently in progress to finalize the second phase algorithm, MPEG-2, for data rates up to 20 Mbps for high-definition video and associated audio. Efforts are also just underway for MPEG-4, which is concerned with very-low-bitrate compression (8-32 kbps) for videophone applications.

23.2.1 Features

MPEG-1 is a generic standard in that it standardizes a *syntax* for the representation of the encoded bitstream and a method of decoding. The syntax supports operations such as motion estimation, motion-compensated prediction, discrete cosine transformation (DCT), quantization, and variable-length coding. Unlike JPEG, MPEG-1 does not define specific algorithms needed to produce a valid data stream; instead, substantial flexibility is allowed in designing the encoder. Similar to H.261, MPEG-1 does not standardize a motion estimation algorithm or a criterion for selecting the compression mode. In addition, a number of parameters defining the coded bitstream and decoders are contained in the bitstream itself. This allows the algorithm to be used with pictures of a variety of sizes and aspect ratios and on channels or devices operating at a wide range of bitrates.

MPEG-1 also offers the following application-specific features: i) Random access is essential in any video storage application. It suggests that any frame should be decodable in a limited amount of time. In MPEG-1, this is achieved by allowing independent access points (I-frames) to the bitstream. ii) Fast forward/reverse search refers to scanning the compressed bit stream and to display only selected frames to obtain fast forward or reverse search. Reverse playback might also be necessary for some interactive applications. iii) Reasonable coding/decoding delay of about 1 sec to give the impression of interactivity in unidirectional video access. Recall that the coding delay in H.261 has been strictly limited to 150 msec to maintain bidirectional interactivity [Gal 92].

23.2.2 Input Video Format

MPEG-1 considers progressive (noninterlaced) video only. In order to reach the target bitrate of 1.5 Mbps, the input video is usually first converted into the MPEG standard input format (SIF). The (Y,Cr,Cb) color space has been adopted, as in CCIR Recommendation 601. In the MPEG-1 SIF, the luminance channel is 352 pixels $\times$ 240 lines and 30 frames/s. Luma and chroma components are represented by 8 bits/pixel, and the chroma components are subsampled by 2 in both the horizontal and vertical directions. The respective locations of the luma and chroma pixels are the same as in the H.261 standard.

While many video parameters, such as the picture size and temporal rate, can be specified in the syntax, and therefore are arbitrary, the following set of constrained parameters are specified to aid hardware implementations:

Maximum number of pixels/line: 720
Maximum number of lines/picture: 576
Maximum number of pictures/sec: 30
Maximum number of macroblocks/picture: 396
Maximum number of macroblocks/sec: 9900
Maximum bitrate: 1.86 Mbps
Maximum decoder buffer size: 376,832 bits.

Note, however, that the constrained parameter set does not suggest that a 720 pixels $\times$ 576 lines $\times$ 30 pictures/s video can be compressed artifact-free at 1.86 Mbps. For example, a CCIR 601 format video, 720 pixels $\times$ 488 lines $\times$ 30 pictures/sec, is usually downsampled to SIF before compression, which trades compression artifacts to spatial blurring in order to reach the target bitrate of 1.5 Mbps.

23.2.3 Data Structure and Compression Modes

Similar to H.261, the MPEG-1 bitstream also follows a hierarchical data structure, consisting of the following six layers, that enables the decoder to interpret the data unambiguously.

1) *Sequences* are formed by several group of pictures.

2) *Group of pictures* (GOP) are made up of pictures.

3) *Pictures* consist of slices. There are four picture types indicating the respective modes of compression: I-pictures, P-pictures, B-pictures, and D-pictures.

I-pictures are intra-frame DCT encoded using a JPEG-like algorithm. They serve as random access points to the sequence. There are two types of interframe encoded pictures, P- and B-pictures. In these pictures the motion-compensated prediction errors are DCT encoded. Only forward prediction is used in the P-pictures, which are always encoded relative to the preceding I- or P-pictures. The prediction of the B-pictures can be forward, backward, or bidirectional relative to other I- or P-pictures. D-pictures contain only the DC component of each block, and serve for browsing purposes at very low bitrates. The number of I, P, and B frames in a GOP are application-dependent, e.g., dependent on access time and bitrate requirements. The composition of a GOP is illustrated by an example.

Example

A GOP is shown in Figure 23.5 which is composed of nine pictures. Note that the first frame of each GOP is always an I-picture. In MPEG, the order in which the pictures are processed is not necessarily the same as their time sequential order. The pictures in Figure 23.5 can be encoded in one of the following orders:

$$0, 4, 1, 2, 3, 8, 5, 6, 7$$
or
$$0, 1, 4, 2, 3, 8, 5, 6, 7$$

since the prediction for P- and B-pictures should be based on pictures that are already transmitted.

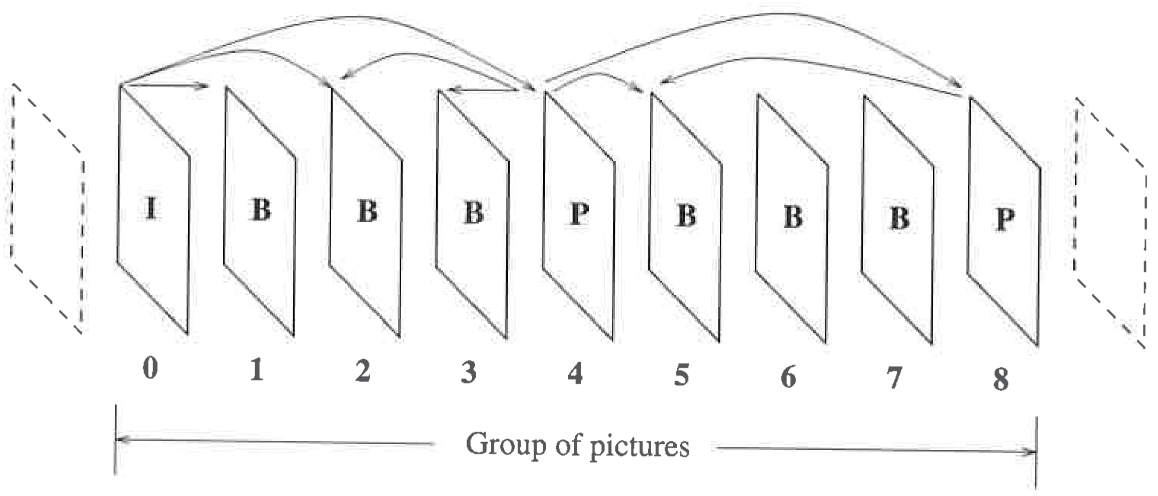


Figure 23.5: Group of pictures in MPEG-1.

4) *Slices* are made up of macroblocks. They are introduced mainly for error recovery.

5) The composition of *macroblocks* (MB) are the same as in the H.261 standard. Some compression parameters can be varied on a MB basis. The MB types depending on the choice of these parameters are listed in Table 23.5. We will take a closer look at each of these MB types in the following when we discuss the video compression algorithm.

6) *Blocks* are 8×8 pixel arrays. They are the smallest DCT unit.

Headers are defined for sequences, GOPs, pictures, slices, and MBs to uniquely specify the data that follows. For an extensive discussion of the MPEG-1 standard, the reader is referred to [ISO 91].

Table 23.5: Macroblock types in MPEG-1.

I-pictures	*P-pictures*	*B-pictures*
Intra	Intra	Intra
Intra-A	Intra-A	Intra-A
	Inter-D	Inter-F
	Inter-DA	Inter-FD
	Inter-F	Inter-FDA
	Inter-FD	Inter-B
	Inter-FDA	Inter-BD
	Skipped	Inter-BDA
		Inter-I
		Inter-ID
		Inter-IDA
		Skipped

23.2.4 Intraframe Compression Mode

The pixel intensity values are DCT encoded in a manner similar to JPEG and the intra mode of H.261. Compression is achieved by a combination of quantization and run-length coding of the zero coefficients.

Quantization

Assuming 8-bit input images, the DC coefficient can take values in the range [0,2040], and the AC coefficients are in the range [-1024,1023]. These coefficients are quantized with a uniform quantizer. The quantized coefficient is obtained by dividing the DCT coefficient value by the quantization step size and then rounding the result to the nearest integer. The quantizer step size varies by the frequency, ac-

Table 23.6: MPEG default intra quantization matrix.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

cording to psycho-visual characteristics, and is specified by the *quantization matrix*. The MPEG default intra quantization matrix is shown in Table 23.6. According to this matrix, the DC coefficient is represented by 8 bits since its weight is 8. The AC coefficients can be represented with less than 8 bits using weights larger than 8.

MPEG allows for spatially-adaptive quantization by introducing a quantizer scale parameter $MQUANT$ in the syntax. As a result, there are two types of MBs in the I-pictures: **"Intra"** MBs are coded with the current quantization matrix. In **"Intra-A"** MBs, the quantization matrix is scaled by $MQUANT$, which is transmitted in the header. Note that $MQUANT$ can be varied on a MB basis to control the bitrate or for subjective quantization. Human visual system models suggest that MBs containing busy, textured areas can be quantized relatively coarsely. One of the primary differences between MPEG intra mode and JPEG is the provision of adaptive quantization in MPEG. It has been claimed that MPEG intra mode provides 30% better compression compared with JPEG due to adaptive quantization.

Coding

Redundancy among the quantized DC coefficients is reduced via DPCM. The resulting signal is VLC coded with 8 bits. The fixed DC Huffman table has a logarithmic amplitude category structure borrowed from JPEG. Quantized AC coefficients are zigzag scanned and converted into [run, level] pairs as in JPEG and H.261. A single Huffman-like code table is used for all blocks, independent of the color component to which they belong. There is no provision for downloading custom tables. Only those pairs which are highly probable are VLC coded. The rest of them are coded with an escape symbol followed by a fixed-length code to avoid extremely long codewords. The codebook is a superset of that of H.261, which is completely different from that of JPEG.

23.2.5 Interframe Compression Modes

In interframe compression modes, a temporal prediction is formed, and the resulting prediction error is DCT encoded. There are two types of temporal prediction modes allowed in MPEG-1: forward prediction (P-pictures) and bidirectional prediction (B-pictures).

P-Pictures

P-pictures allow motion-compensated forward predictive coding with reference to a previous I- or P-picture. The temporal prediction process is illustrated in Figure 23.6, where the prediction for a MB $\mathbf{b}$ is given by

$$\hat{\mathbf{b}} = \tilde{\mathbf{c}} \tag{23.2}$$

and $\tilde{\mathbf{c}}$ denotes the MB corresponding to $\mathbf{b}$ in the "reconstructed" previous frame.

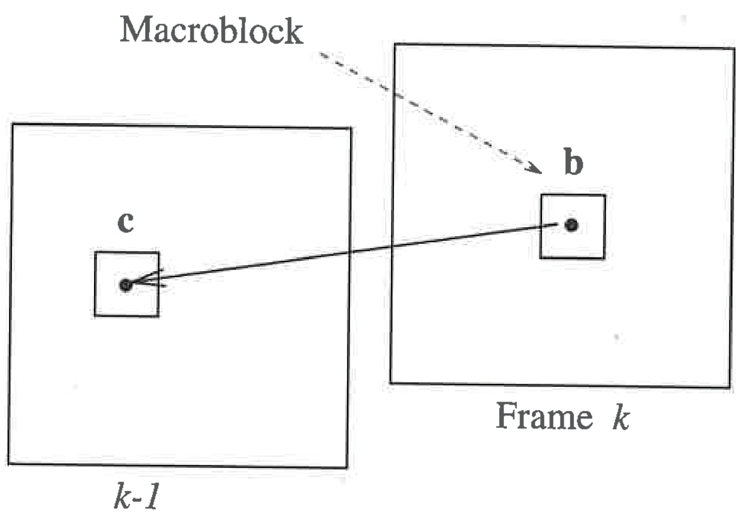


Figure 23.6: MPEG-1 forward prediction.

The mode of compression for each MB is selected by the encoder from the list of allowable modes for a P-picture shown in Table 23.5. **"Intra"** and **"Intra-A"** MBs in P-pictures are coded independently of any reference data just like MBs in the I-pictures. MBs classified as **"Inter"** are interframe coded, and the temporal prediction may use motion compensation (MC) and/or adaptive quantization. The subscript "D" indicates that the DCT of the prediction error will be coded, "F" indicates that forward MC is ON, and "A" indicates adaptive quantization (a new value of MQUANT is also transmitted). That is, if a MB is labeled **"Inter-F"** then the motion-compensated prediction $\hat{\mathbf{b}}$ is satisfactory, so we need to transmit just the motion vector $\mathbf{d}$ for that MB, **"Inter-FD"** denotes that we need to transmit a motion vector and the DCT coefficients of the prediction error, and **"Inter-FDA"** indicates that in addition to a motion vector and the DCT coefficients, a new value of MQUANT is also being transmitted for that MB. A macroblock may be **"Skipped"** if the block at the same position in the previous frame (without MC) is good enough, indicating a stationary area.

B-Pictures

B-pictures is a key feature of MPEG-1, which allows MC interpolative coding, also known as bidirectional prediction. The temporal prediction for the B-pictures is given by

$$\hat{\mathbf{b}} = \alpha_1 \tilde{\mathbf{c}}_1 + \alpha_2 \tilde{\mathbf{c}}_2 \qquad \alpha_1,\ \alpha_2 = 0,\ 0.5,\ 1 \qquad \alpha_1 + \alpha_2 = 1 \tag{23.3}$$

where $\tilde{}$ denotes "reconstructed" values. Then $\alpha_1 = 1$ and $\alpha_2 = 0$ yields forward prediction, $\alpha_1 = 0$ and $\alpha_2 = 0$ gives backward prediction, and $\alpha_1 = \alpha_2 = 0.5$

corresponds to bidirectional prediction. This is illustrated in Figure 23.7. Note that in the bidirectional prediction mode, two displacement vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ and the corresponding prediction error $\mathbf{b}-\hat{\mathbf{b}}$ need to be encoded for each macroblock $\mathbf{b}$.

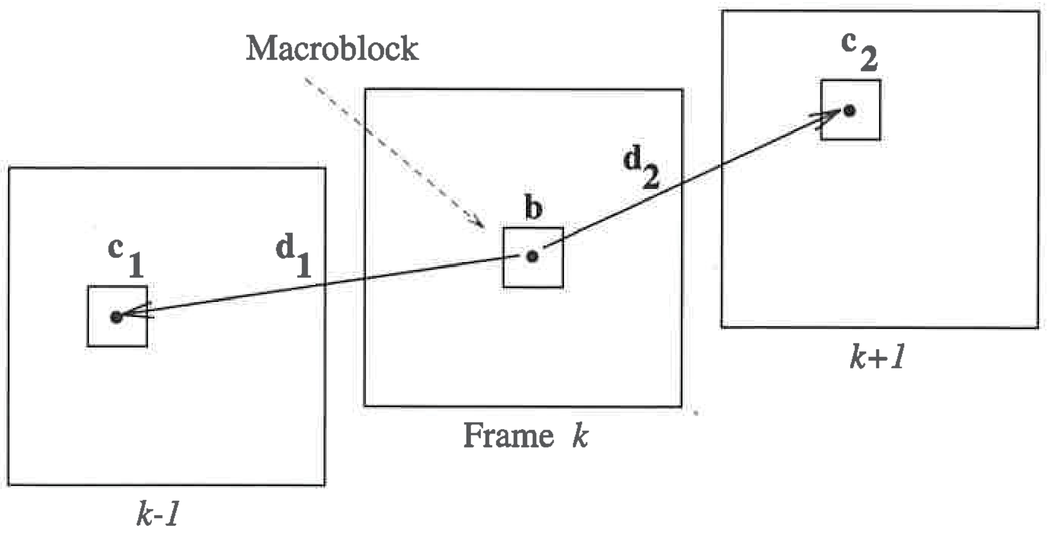


Figure 23.7: MPEG-1 bi-directional prediction.

The concept of bidirectional prediction or interpolative coding can be considered as a temporal multiresolution technique, where we first encode only the I- and P-pictures (typically 1/3 of all frames). Then the remaining frames can be interpolated from the reconstructed I and P frames, and the resulting interpolation error is DCT encoded. The use of B-pictures provides several advantages:

- They allow effective handling of problems associated with covered/uncovered background. If an object is going to be covered in the next frame, it can still be predicted from the previous frame or vice versa.
- MC averaging over two frames may provide better SNR compared to prediction from just one frame.
- Since B-pictures are not used in predicting any future pictures, they can be encoded with fewer bits without causing error propagation.

The trade-offs associated with using B-pictures are:

- Two frame-stores are needed at the encoder and decoder, since at least two reference (P and/or I) frames should be decoded first.
- If too many B-pictures are used, then i) the distance between the two reference frames increases, resulting in lesser temporal correlation between them, and hence more bits are required to encode the reference frames, and ii) we have longer coding delay.

The mode of compression for each MB in a B-picture is selected independently from the list of allowable modes shown in Table 23.5. Again, **"Intra"** and **"Intra-A"** MBs are coded independently of any reference frame. MBs classified as **"Inter"** have the following options: "D" indicates that the DCT of the prediction error will be coded, "F" indicates forward prediction with motion compensation, "B" indicates backward prediction with motion compensation, "I" indicates interpolated prediction with motion compensation, and "A" indicates adaptive quantization. A macroblock may be **"Skipped"** if the block from the previous frame is good enough as is; that is, no information needs to be sent.

Quantization and Coding

In the interframe mode, the inputs to the DCT are in the range [-255,255]; thus, all DCT coefficients have the dynamic range [-2048,2047]. The quantization matrix is such that the effective quantization is relatively coarser compared to those used for I-pictures. All quantized DCT coefficients, including the DC coëfficient, are zigzag scanned to form [*run*, *level*] pairs, which are then coded using VLC. Displacement vectors are DPCM encoded with respect to the motion vectors of the previous blocks. VLC tables are specified for the type of MB, the differential motion vector, and the MC prediction error. Different Huffman tables are defined for encoding the macroblock types for P- and B-pictures, whereas the tables for motion vectors and the DCT coefficients are the same for both picture types.

23.2.6 MPEG-1 Encoder and Decoder

An MPEG-1 encoder includes modules for motion estimation, selection of compression mode (MTYPE) per MB, setting the value of MQUANT, motion-compensated prediction, quantizer and dequantizer, DCT and IDCT, variable-length coding (VLC), a multiplexer, a buffer, and a buffer regulator. The dequantizer and the IDCT are needed in the encoder because the predictions are based on reconstructed data. The IDCT module at the encoder should match within a prespecified tolerance the IDCT module at the decoder to avoid propogation of errors in the prediction process. This tolerance is specified in IEEE Standard 1180-1990 for 64-bit floating-point IDCT implementations.

The relative number of I-, P- or B-pictures in a GOP is application-dependent. The standard specifies that one out of every 132 pictures must be an I-picture to avoid error propagation due to IDCT mismatch between the encoder and decoder. The use of B-pictures is optional. Neither the motion estimation algorithm nor the criterion to select MYTPE and MQUANT are part of the standard. In general, motion estimation is performed using the luminance data only. A single displacement vector is estimated for each MB. One-half (0.5) pixel accuracy is allowed for motion estimates. The maximum length of the vectors that may be represented can be changed on a picture-by-picture basis to allow maximum flexibility. Motion vectors that refer to pixels outside the picture are not allowed.

In summary, a typical MPEG encoder performs the following steps:

1. Decide on the labeling of I-, P- and B-pictures in a GOP.
2. Estimate a motion vector for each MB in the P- and B-pictures.
3. Determine the compression mode MTYPE for each MB from Table 23.5.
4. Set the quantization scale, MQUANT, if adaptive quantization is selected.

An MPEG-1 decoder reverses the operations of the encoder. The incoming bit stream (with a standard syntax) is demultiplexed into DCT coefficients and side information such as MTYPE, motion vectors, MQUANT, and so on. The decoder employs two frame-stores, since two reference frames are used to decode the B-pictures.

We conclude this section by summarizing the main differences between the H.261 and MPEG-1 standards in Table 23.7.

Table 23.7: Comparison of H.261 and MPEG-1 Standards

H.261	**MPEG-1**
Sequential access	Random access
One basic frame rate	Flexible frame rate
CIF and QCIF images only	Flexible image size
I and P frames only	I, P and B frames
MC over 1 frame	MC over 1 or more frames
1 pixel MV accuracy	1/2 pixel MV accuracy
121 filter in the loop	No filter
Variable threshold + uniform quantization	Quantization matrix
No GOF structure	GOF structure
GOB structure	Slice structure

23.3 The MPEG-2 Standard

The quality of MPEG-1 compressed video at 1.2 Mbps has been found unacceptable for most entertainment applications. Subjective tests indicate that CCIR 601 video can be compressed with excellent quality at 4-6 Mbps. MPEG-2 is intended as a compatible extension of MPEG-1 to serve a wide range of applications at various bitrates (2-20 Mbps) and resolutions. Main features of the MPEG-2 syntax are: i) it allows for interlaced inputs, higher-definition inputs, and alternative subsampling of the chroma channels, ii) it offers a scalable bitstream, and iii) it provides improved quantization and coding options.

Considering the practical difficulties with the implementation of the full syntax on a single chip, subsets of the full syntax have been specified under five "profiles," simple profile, main profile, SNR scalable profile, spatially scalable profile, and high profile. Furthermore, a number of "levels" have been introduced within these profiles to impose constraints on some of the video parameters [ISO 93]. It is important to note that the MPEG-2 standard has not yet been finalized. The syntax of the Main Profile was frozen in March 1993. However, work in other profiles is still ongoing. It is likely that the all-digital HDTV compression algorithm will conform with one of the profiles in MPEG-2.

In the following we discuss the MB structure in MPEG-2, how MPEG-2 handles interlaced video, the concepts related to scalability, and some extensions for encoding of higher-definition video along with a brief overview of the profiles and levels.

23.3.1 MPEG-2 Macroblocks

A macroblock (MB) refers to four 8 × 8 luminance blocks and the spatially associated chroma blocks. MPEG-2 allows for three chroma subsampling formats, 4:2:0 (same as MPEG-1), 4:2:2 (chroma subsampled in the horizontal direction only), and 4:4:4 (no chroma subsampling). The spatial locations of luma and chroma pixels for the 4:2:0 and 4:2:2 formats are depicted in Figure 23.8. Therefore, in MPEG-2 a MB may contain 6 (4 luma, 1 Cr, and 1 Cb), 8 (4 luma, 2 Cr, and 2 Cb), or 12 (4 luma, 4 Cr, and 4 Cb) 8 × 8 blocks.

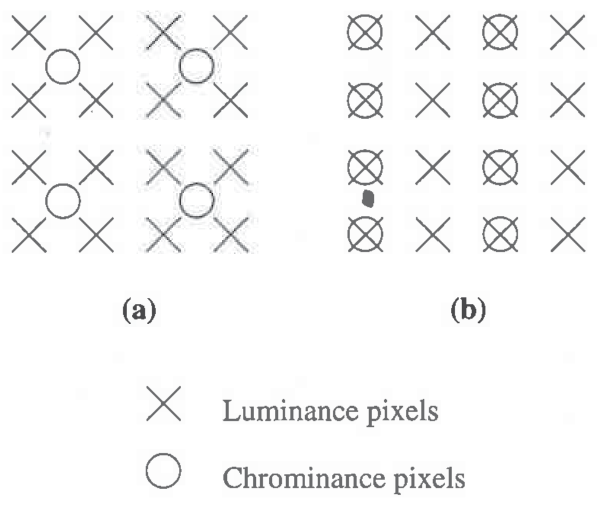


Figure 23.8: Chrominance subsampling options: a) 4:2:0 format; b) 4:2:2 format.

23.3.2 Coding Interlaced Video

MPEG-2 accepts both progressive and interlaced inputs. If the input is interlaced, the output of the encoder consists of a sequence of fields that are separated by the field period. There are two options in coding interlaced video: i) every field can be encoded independently (field pictures), or ii) two fields may be encoded together as a composite frame (frame pictures). It is possible to switch between frame pictures and field pictures on a frame-to-frame basis. Frame encoding is preferred for relatively still images; field encoding may give better results when there is significant motion. In order to deal with interlaced inputs effectively, MPEG-2 supports:

- two new picture formats: frame-picture and field-picture,
- field/frame DCT option per MB for frame pictures, and
- new MC prediction modes for interlaced video,

which are described in the following.

New Picture Types for Interlaced Video

Interlaced video is composed of a sequence of even and odd fields separated by a field period. MPEG-2 defines two new picture types for interlaced video. They are:

i) *Frame pictures*, which are obtained by interleaving lines of even and odd fields to form composite frames. Frame pictures can be I-, P-, or B-type. An MB of the luminance frame picture is depicted in Figure 23.9.

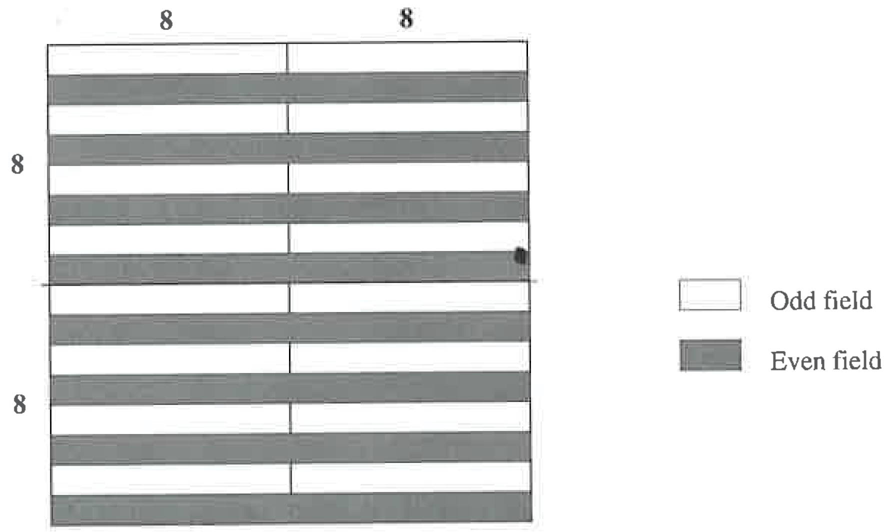


Figure 23.9: The luminance component of a MB of a frame picture.

ii) *Field pictures* are simply the even and odd fields treated as separate pictures. Each field picture can be I-, P- or B-type.

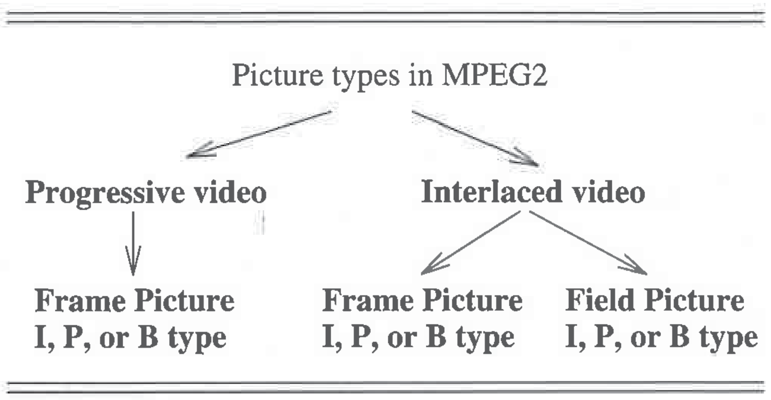


Figure 23.10: Summary of picture types in MPEG-2.

A summary of all picture types is shown in Figure 23.10. Clearly, in progressive video all pictures are frame pictures. A group of pictures can be composed of an arbitrary mixture of field and frame pictures. Field pictures always appear in pairs (called the top field and bottom field) which together constitute a frame. If the top field is a P- (B-) picture, then the bottom field must also be a P- (B-) picture. If the top field is an I-picture, then the bottom field can be an I- or a P-picture. A pair of field pictures are encoded in the order in which they should appear at the output. An example of a GOP for an interlaced video is shown in Figure 23.11.

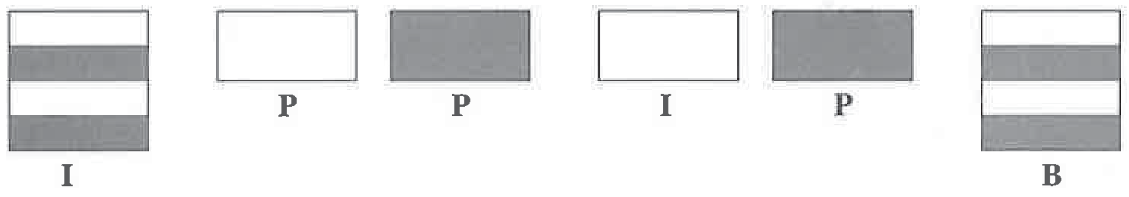


Figure 23.11: A GOP for an interlaced video.

Field/Frame DCT Option for Frame Pictures

MPEG-2 allows a field- or frame-DCT option for each MB in a frame picture. This allows computing DCT on a field-by-field basis for specific parts of a frame picture. For example, field-DCT may be chosen for macroblocks containing high motion, whereas frame-DCT may be appropriate for macroblocks with little or no motion but containing high spatial activity. The internal organization of a MB for frame (on the left) and field (on the right) DCT is shown in Figure 23.12. Note that in 4:2:0 sampling, only frame DCT can be used for the chroma blocks to avoid 8×4 IDCT.

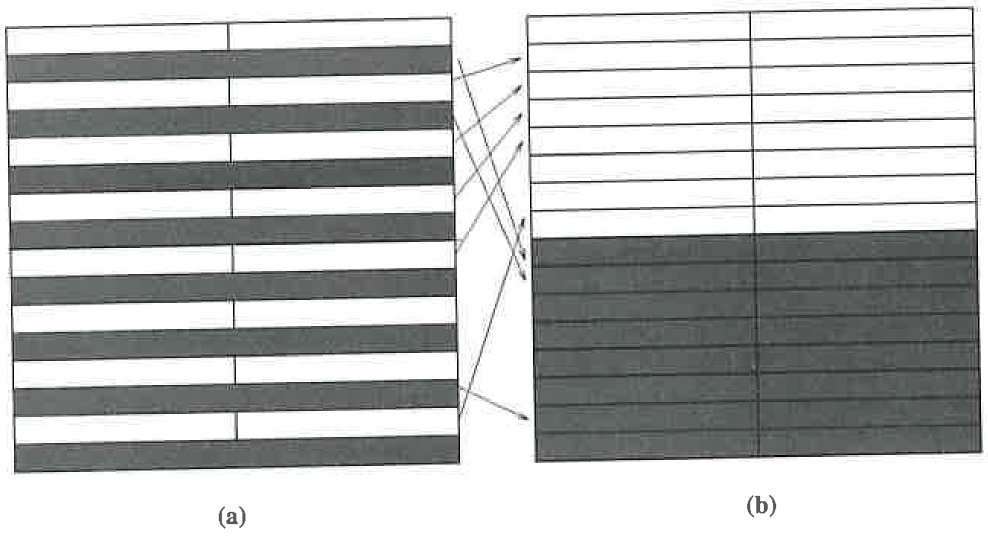


Figure 23.12: DCT options for interlaced frame pictures: a) Frame DCT and b) field DCT.

MC Prediction Modes for Interlaced Video

There are two main types of predictions: simple field and simple frame prediction. In simple field prediction, each field is predicted independently using data from one or more previously decoded fields. Simple frame prediction forms a prediction for an entire frame based on one or more previously decoded frames. Within a field picture only field predictions can be used. However, in a frame picture either field or frame prediction may be employed on an MB-by-MB basis. Selection of the best prediction mode depends on presence/absence of motion in an MB, since in the presence of motion, frame prediction suffers from strong motion artifacts, while in the absence of motion, field prediction does not utilize all available information.

There are also two other prediction modes: 16×8 MC mode and dual-prime mode. 16×8 MC mode is only used in field pictures, where two motion vectors are used per MB, one for the upper and the other for the lower 16×8 region, which belong to the top and bottom fields, respectively. In the case of bidirectional prediction, four motion vectors will be needed. In dual-prime mode, one motion vector and a small differential vector are encoded. In the case of field pictures, two motion vectors are derived from this information and used to form predictions from two reference fields, which are averaged to form the final prediction [ISO 93]. Dual-prime mode is used only for P-pictures.

23.3.3 Scalable Extensions

Scalability refers to ability to decode only a certain part of the bit-stream to obtain video at the desired resolution. It is assumed that decoders with different complexities can decode and display video at different spatio-temporal resolutions from the same bitstream. The minimum decodable subset of the bitstream is called the base layer. All other layers are enhancement layers, which improve the resolution of the base layer video. MPEG-2 syntax allows for two or three layers of video. There are different forms of scalability:

Spatial (pixel resolution) scalability provides the ability to decode video at different spatial resolutions without first decoding the entire frame and decimating it. The base layer is a low spatial resolution version of the video. Enhancement layers contain successively higher-frequency information. MPEG-2 employs a pyramidal coding approach. The base layer video is obtained by decimating the original input video. The enhancement layer is the difference of the actual input video and the interpolated version of the base-layer video.

SNR scalability offers decodability using different quantizer step sizes for the DCT coefficients. The base-layer video is obtained by using a coarse quantization of the DCT coefficients. It is at the same spatio-temporal resolution with the input video. The enhancement layer simply refers to the difference of the base layer and the original input video.

Temporal scalability refers to decodability at different frame rates without first decoding every single frame. *Hybrid scalability* refers to some combination of the above. An important advantage of scalability is that it provides higher resilience to transmission errors as the base-layer video is usually transmitted with better error correction capabilities.

23.3.4 Other Improvements

MPEG-2 also features some extensions in the quantization and coding steps for improved image quality in exchange to slightly higher bitrate. In particular, it allows for i) a new scanning scheme (alternate scan) in addition to the zigzag scanning of the DCT coefficients, ii) finer quantization of the DCT coefficients, iii) finer adjustment of the quantizer scale factor, and iv) a separate VLC table for the DCT coefficients for the intra macroblocks, which are explained in the following.

Alternate Scan

In addition to the zigzag scanning, MPEG-2 allows for an optional scanning pattern, called the "alternate scan." The alternate scan pattern, which is said to fit interlaced video better, is depicted in Figure 23.13.

Finer Quantization of the DCT Coefficients

In intra macroblocks, the quantization weight for the DC coefficient can be 8, 4, 2 or 1. That is, 11 bits (full) resolution is allowed for the DC coefficient. Recall that this weight is fixed to 8 in MPEG-1. AC coefficients are quantized in the range [-2048,2047], as opposed to [-256,255] in MPEG-1. In non-intra macroblocks, all coefficients are quantized into the range [-2048,2047]. This range was [-256,255] in MPEG-1.

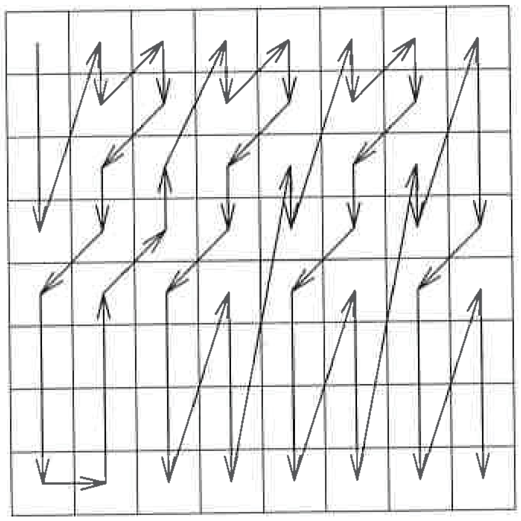


Figure 23.13: Alternate scan.

Finer Adjustment of MQUANT

In addition to a set of MQUANT values that are integers between 1 and 31, MPEG-2 allows for an optional set of 31 values that include real numbers ranging from 0.5 to 56. These values are listed in Table 23.8.

Table 23.8: Optional set of MQUANT values.

0.5	5.0	14.0	32.0
1.0	6.0	16.0	36.0
1.5	7.0	18.0	40.0
2.0	8.0	20.0	44.0
2.5	9.0	22.0	48.0
3.0	10.0	24.0	52.0
3.5	11.0	26.0	56.0
4.0	12.0	28.0	

23.3.5 Overview of Profiles and Levels

MPEG-2 full syntax covers a wide range of features and free parameters. The five MPEG-2 profiles define subsets of the syntax while the four levels impose constraints on the values of the free parameters for the purpose of practical hardware implementations. The parameter constraints imposed by the four levels are summarized in Table 23.9.

Table 23.9: Parameter constraints according to levels.

Level	Max. Pixels	Max. Lines	Max. Frames/s
Low	352	288	30
Main	720	576	30
High-1440	1440	1152	60
High	1920	1152	60

The Simple profile does not allow use of B-pictures and only support the Main level. The maximum bitrate for the Simple profile is 15 Mbps. The Main profile supports all four levels with upper bounds on the bitrates equal to 4, 15, 60, and 80 Mbps for the Low, Main, High-1440 and High levels, respectively. The Main profile does not include any scalability. The SNR Scalable profile supports Low and Main levels with maximum bitrates 4 (3) and 15 (10) Mbps, respectively. The numbers in parentheses indicate the maximum bitrate for the base layer. The Spatially Scalable profile supports only High-1440 level with a maximum bitrate of 60 (15) Mbps. The High profile includes Main, High-1440 and High levels with maximum bitrates of 20 (4), 80 (20), and 100 (25) Mbps, respectively.

23.4 Software and Hardware Implementations

There are several software and hardware implementations of the H.261 and MPEG algorithms. The Portable Video Research Group (PVRG) at Stanford University has public-domain source codes for both H.261 and MPEG standards, called PVRG-P64 and PVRG-MPEG, respectively. They can be obtained through anonymous ftp from "havefun.stanford.edu" - IP address [36.2.0.35] - (/pub/p64/P64v1.2.tar.Z) and (/pub/mpeg/MPEGv1.2.tar.Z). Other public-domain software includes the INRIA H.261 codec, which can be obtained from "avahi.inria.fr" (/pub/h261.tar.Z), and the Berkeley Plateau Research Group MPEG encoder, which can be obtained from "toe.cs.berkeley.edu" - IP address [128.32.149.117] - (/pub/multimedia/mpeg/mpeg-2.0.tar.Z).

Single- or multichip implementations of the video compression standards are available from many vendors, including

- C-Cube: CL-450, single-chip MPEG-1, SIF rates. CL-950, MPEG-2, CL-4000, single-chip, can code MPEG-1, H.261, and JPEG,
- SGS-Thomson: STi-3400, single-chip MPEG-1, SIF rates. STi-3500, the first MPEG-2 chip on the market,
- Motorola: MCD250, single-chip MPEG-1, SIF rates, and

- GEC Plassey: Chip sets for the H.261 algorithm.

For more complete listings, the reader is referred to MPEG-FAQ, available from "phade@cs.tu-berlin.de," and to Compression-FAQ (part 3), available by ftp from "rtfm.mit.edu" (/pub/usenet/news.answers/compression-faq/part[1-3].

Bibliography

[CCI 90] CCITT Recommendation H.261: "Video Codec for Audio Visual Services at $p \times 64$ kbits/s," COM XV-R 37-E, 1990.

[Che 93] C.-T. Chen, "Video compression: Standards and applications," *J. Visual Comm. Image Rep.*, vol. 4, no. 2, pp. 103–111, Jun. 1993.

[Chi 95] L. Chiariglione, "The development of an integrated audiovisual coding standard: MPEG," *Proc. IEEE*, vol. 83, no. 2, pp. 151–157, Feb. 1995.

[Gal 91] D. J. LeGall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, 1991.

[Gal 92] D. J. LeGall, "The MPEG video compression algorithm," *Signal Proc.: Image Comm.*, vol. 4, pp. 129–140, 1992.

[ISO 91] ISO/IEC CD 11172: "Coding of Moving Pictures and Associated Audio - For Digital Storage Media at up to 1.5 Mbits/sec," Dec. 1991.

[ISO 93] ISO/IEC CD 13818-2: "Generic Coding of Moving Pictures and Associated Audio," Nov. 1993.

[Lio 91] M. L. Liou, "Overview of the $p \times 64$ kbits/sec video coding standard," *Commun. ACM*, vol. 34, pp. 59–63, 1991.

[Lio 90] M. L. Liou, "Visual telephony as an ISDN application," *IEEE Communications Magazine*, vol. 28, pp. 30–38, 1990.

[Oku 92] S. Okubo, "Requirements for high quality video coding standards," *Signal Proc.: Image Comm.*, vol. 4, pp. 141–151, 1992.

[Pen 93] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still image data compression standard*, New York, NY: Van Nostrand Reinhold, 1993.

[RM8 89] COST211bis/SIM89/37: "Description of Reference Model 8 (RM8)," PTT Research Laboratories, The Netherlands, May 1989.

Chapter 24

MODEL-BASED CODING

Due to growing interest in very-low-bitrate digital video (about 10 kbps), significant research effort has recently been focused on new compression methods based on structural models, known as model-based analysis-synthesis coding. Scientists became interested in model-based coding because the quality of digital video provided by hybrid waveform encoders, such as the CCITT Rec. H.261 encoder, has been deemed unacceptable at these very low bitrates. The general principles of model-based coding, including general object-based, knowledge-based, and semantic coding, were presented in Chapter 22. These techniques employ structural models ranging from general purpose 2-D or 3-D object models [Mus 89, Die 93, Ost 93] to application-specific wireframe models [Aiz 89, Li 93, Boz 94]. In the following, we elaborate on general 2-D/3-D object-based coding in Section 24.1, and knowledge-based and semantic coding in Section 24.2.

24.1 General Object-Based Methods

A major deficiency of the MC/DCT compression is that it is based on a 2-D translatory block-motion model. This model is not adequate for a precise description of most motion fields, because it does not include rotation and zooming, and boundaries of moving objects hardly ever coincide with those of the rectangular blocks. Object-based methods aim to develop more realistic 2-D motion field models by considering affine, perspective, and bilinear spatial transformations and/or segmentation of the scene into individually moving objects. Typically, a frame is partitioned into an "unchanged region," an "uncovered background," and a number of moving objects which are either model-compliant (MC) or model-failure (MF) objects, as depicted in Figure 24.1. Segmentation is an integral part of these schemes, because different regions require different model parameters. We can classify object-based models as: i) piecewise planar or arbitrary 3-D surfaces with 3-D motion, ii) 2-D flexible models with 2-D translational motion, and iii) spatial transformations with triangular/rectangular patches, which are described in the following.

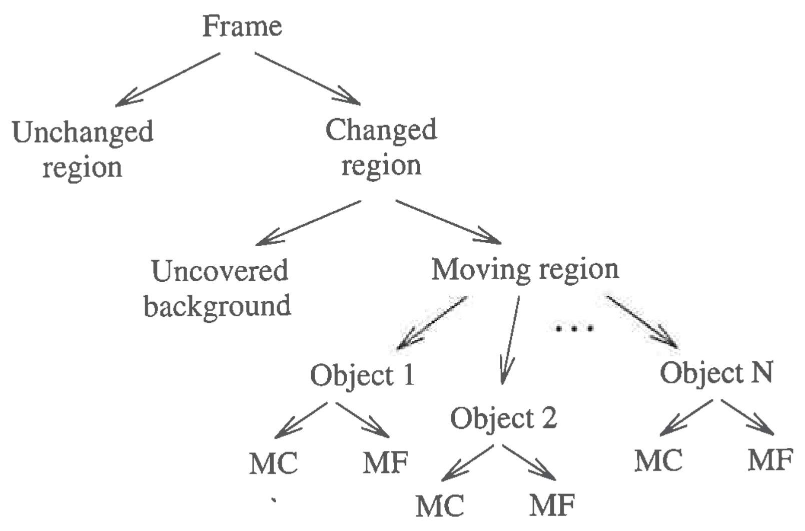


Figure 24.1: Segmentation of a frame into objects.

24.1.1 2-D/3-D Rigid Objects with 3-D Motion

In this approach, a 3-D model of a moving scene is estimated, which is consistent with a given sequence of frames. A realistic 2-D motion field model can then be obtained by projecting the resulting 3-D motion into the image plane. Two approaches are commonly used to represent 3-D surfaces: i) approximation by a piecewise planar model, also known as the case of 2-D rigid objects with 3-D motion [Hot 90, Die 93], and ii) estimation of a global surface model under a smoothness constraint, which leads to the so-called 3-D object with 3-D motion models [Mus 89, Ost 90, Ost 93, Mor 91, Koc 93]. Here, we provide a brief overview of the former approach.

We have seen in Chapter 9 that the orthographic and perspective projections of arbitrary 3-D motion of a rigid planar patch into the image plane yield the 6-parameter affine and the 8-parameter perspective models, respectively. It follows that, using the 2-D rigid object with 3-D motion approach, the 2-D motion field can be represented by a piecewise affine or a piecewise perspective field, where the boundaries of the patches and the respective model parameters can be estimated by a simultaneous motion estimation and segmentation algorithm (see Chapter 11). Then the parameter set for each independently moving object (motion parameters) together with the segmentation mask denoting the boundaries of each object in the image plane (shape parameters) constitute a complete description of the frame-to-frame pixel correspondences, which provide an MC-prediction of the next frame. Because of motion estimation/segmentation errors and problems related to uncovered background, the frame prediction error or the synthesis error (color parameters) also needs to be transmitted to improve upon image quality in the model failure regions.

An object-oriented analysis-synthesis encoder implements the following steps:

1. *Analysis:* Perform simultaneous motion parameter estimation and segmentation, between frame k and the reconstructed frame $k-1$, to find the best motion and shape parameters. Two specific algorithms that have been reported with successful results are those of Hotter *et al.* [Hot 90] and Diehl [Die 93]. They can be summarized using the flowchart shown in Figure 24.2, where the frame is initially segmented into a changed and an unchanged region. Each contiguous changed segment is assumed as an independent moving object, and a set of mapping parameters, such as a 6-parameter affine or an 8-parameter perspective mapping, is estimated for each moving object. The present frame (frame k) can then be synthesized from the previously reconstructed frame $k-1$ using these parameters. Those objects, where the synthesis error is

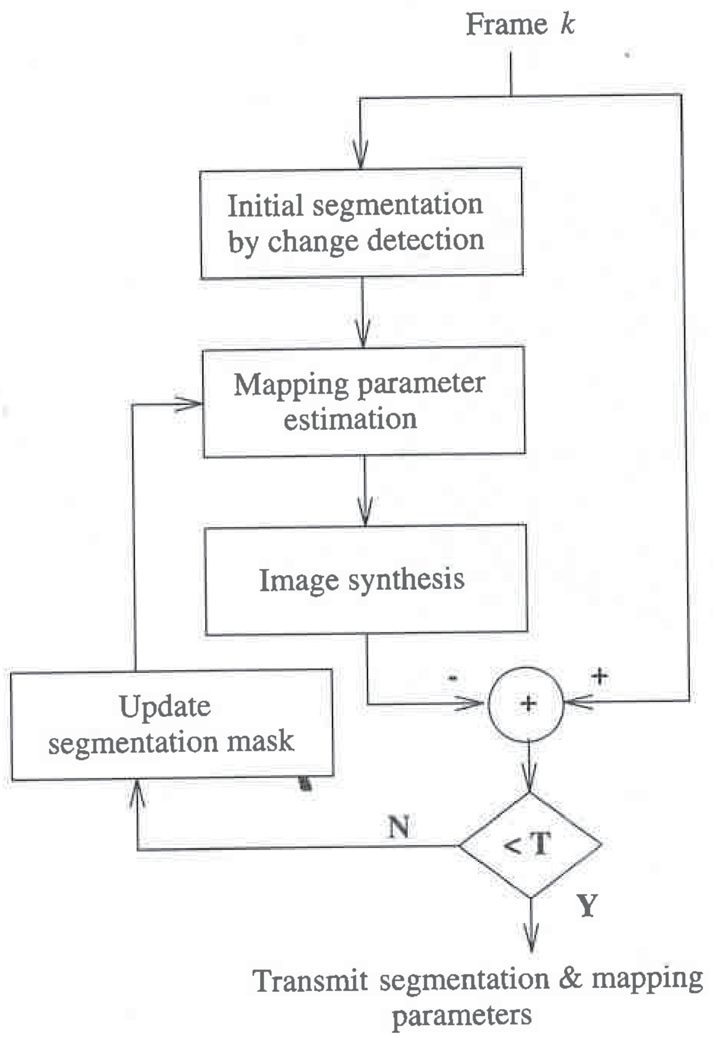

Frame k
Initial segmentation by change detection
Mapping parameter estimation
Image synthesis
Update segmentation mask
-
+
+
N
< T
Y
Transmit segmentation & mapping parameters

Figure 24.2: Flowchart of the mapping-parameter-based method.

above a threshold T, are classified as model-failure (MF) objects, which are then subdivided into more objects. The process is repeated until a predefined performance measure is satisfied.

2. *Synthesis:* Given the segmentation field and the motion parameters for each region, synthesize the present frame from the previous reconstructed frame. Compute the synthesis error, which specifies the color parameters.

3. *Coding:* In the case of the 8-parameter model, the parameters $(a_1, \ldots, a_8)$ are first normalized by a factor K, except for a_3 and a_6 which describe the horizontal and vertical translation of the object. The factor K is coded by 4 bits. The normalization aims at making the accuracy of the motion description independent of the spatial extent of the objects. The normalized parameters $(a_1, a_2, a_4, a_5, a_7, a_8)$ are coded with 6 bits each. The parameters a_3 and a_6 are quantized with quarter pixel accuracy, and coded with 7 bits each. Clearly, the amount of motion information to be transmitted depends on the number of independently moving objects. There exist several approaches for encoding the segmentation mask (shape) and the frame prediction error (color), which are described in [Mus 89, Die 93, Sch 93].

The difficulty with this approach is in the simultaneous estimation of the model parameters and scene segmentation, which is computationally demanding.

24.1.2 2-D Flexible Objects with 2-D Motion

Here, the source model is a flexible 2-D object with translational motion. This model, which forms the basis of the COST 211ter simulation model, has been proposed because it does not require complex motion estimation schemes, and hence leads to practical codec realizations. The main idea is to segment the current frame into three regions: i) unchanged or stationary areas, ii) moving areas, where each pixel corresponds to a pixel from the previous frame, and iii) uncovered background for which no corresponding pixel exists in the previous frame. Motion estimation is performed using a hierarchical block matching algorithm on a sparse grid of pixels within the moving region. A dense motion field is then computed via bilinear interpolation of the estimated motion vectors. Model-compliant (MC) moving objects are encoded by the respective 2-D motion vectors and their shape parameters. Shape and color information is encoded and transmitted for model-failure (MF) moving areas and the uncovered background.

The resulting algorithm, called the object-based analysis-synthesis coder (OBASC), whose flowchart is shown in Figure 24.3, can be summarized as follows:

1. Compute the Change Detection Mask (use 3×3 averaging, thresholding, 5×5 median filtering, and/or morphological operations to eliminate small changed/unchanged regions) The change detection mask is a binary mask that marks the changed and unchanged regions of the current picture with respect to the previous reconstructed picture.

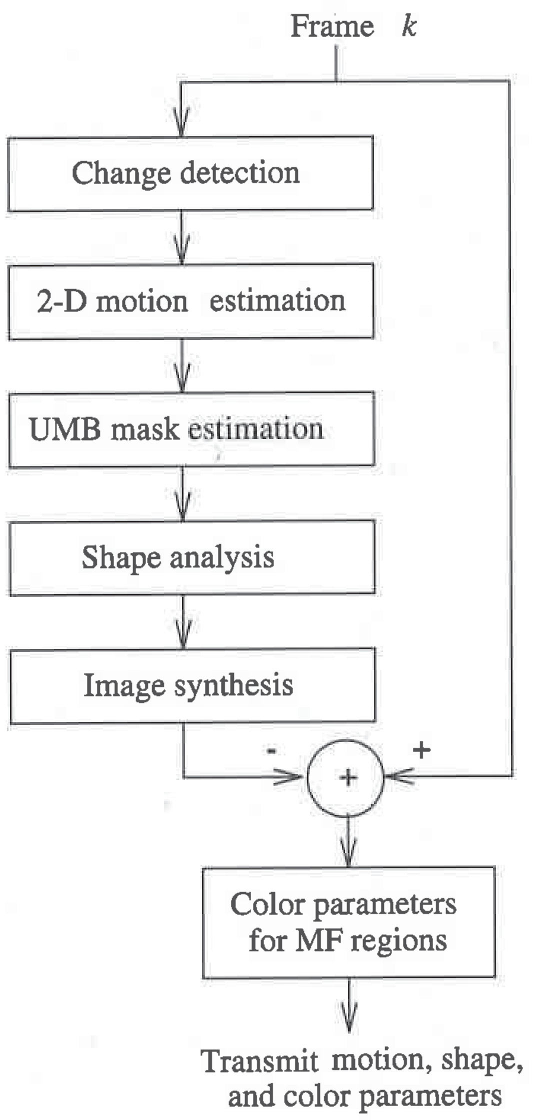


Figure 24.3: Flowchart of the 2-D translatory flexible model method.

2. Estimate motion vectors for those pixels within the changed region using three-level hierarchical block matching (HBM). The parameters used in HBM is tabulated in [Ger 94].

3. Compute the ternary UMB mask that marks unchanged areas, moving areas, and the uncovered background in the current picture. In order to distinguish moving pixels from the uncovered background, for each pel in the CHANGED region, we invert its motion vector. If the pel pointed to by the inverse of the motion vector is not in the CHANGED region, it is said to belong to the UNCOVERED background.

4. Approximate the shape of moving regions using the combined polygon/spline approximation method.

5. Synthesize moving regions in the present picture using motion and shape information. All analysis is performed on the luma component only, but synthesis must be performed for both luma and chroma components.

6. Determine Model Failure (MF) Regions.

7. Code pel data in MF and uncovered background regions. Several methods for coding of pel (color) data have been discussed in [Sch 93].

8. Code motion and shape parameters for Model Compliance (MC) regions. The sparse motion field is encoded using DPCM. The x_1 and x_2 components of the motion vectors are predicted separately using three-point spatial predictors whose coefficients are encoded using 6 bits each. The prediction error is usually run-length encoded [Hot 90]. The effectiveness of 2-D object-based coding methods depends strongly on how efficiently and accurately the shape (segmentation) information can be encoded. Several contour coding methods, including Fourier descriptors and polygon approximations, have been tested [Hot 90]. A combination of polygon approximation and spline-based representation of contours has been found to be most effective. In this approach the vertices of the polygon approximation are used to fit a spline to represent the segment boundary. The spline approximation is constrained to be within a prespecified distance from the actual boundary. The vertices of the polygon approximation are coded by MC-temporal DPCM. That is, first the vertices from the previous frame are translated by the estimated motion vectors. The MC-predicted vertices are tested for whether to accept or reject them. Additional vertices may need be inserted to account for the flexible nature of the object model. The positions of the newly inserted vertices are encoded by relative addressing.

9. Buffer regulation ensures smooth operation of the encoder at a fixed bitrate.

24.1.3 Affine Transformations with Triangular Meshes

Because of the computational complexity of simultaneous mapping parameter estimation and segmentation, and shape analysis algorithms, simpler motion-compensation schemes using spatial transformations based on a predetermined partition of the image plane into triangular or rectangular patches have recently been proposed [Nak 94]. These methods, although based on patches with a predetermined shape, provide results that are superior to those of block-based MC/DCT approach, because the spatial transformations accomodate rotation and scaling in addition to translation, and the implicit continuity of the motion field alleviates blocking artifacts even at very low bitrates.

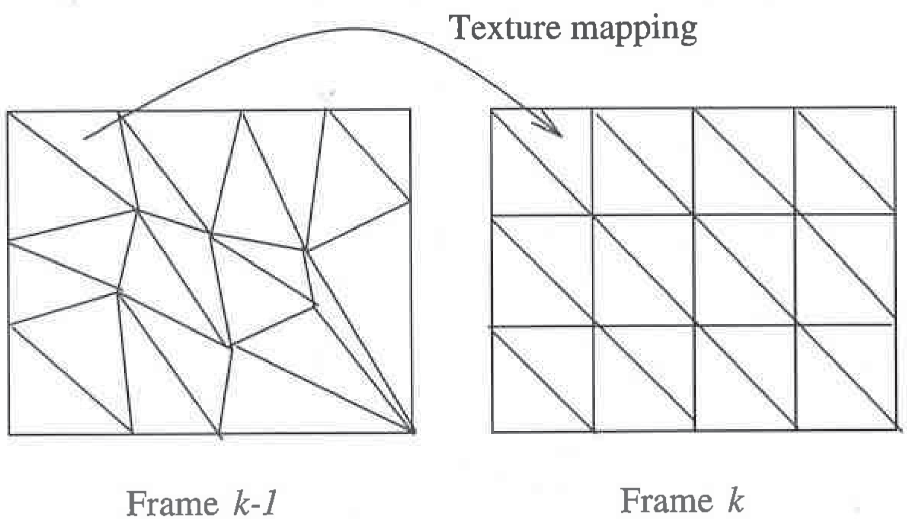


Figure 24.4: Motion compensation using triangular patches.

Nakaya and Harashima [Nak 94] propose using affine, bilinear, or perspective transformations, along with some new motion estimation algorithms, for improved motion compensation. In the case of the affine transformation, the present frame is segmented into triangular patches, because the affine transform has six free parameters which can uniquely be related to the (x_1, x_2) coordinates of the vertices of a triangular patch. Note that both the bilinear and perspective transforms have eight free parameters which can be related to the vertices of a quadrilateral patch. Assuming continuity of the motion field across the patches, it is sufficient to estimate the motion vectors at the vertices of the patches in frame k, called the grid points. Then texture within warped triangles in frame $k-1$, whose boundaries are determined by the estimated motion vectors, is mapped into the respective triangles in frame k, as depicted in Fig, 24.4.

An encoder using affine motion compensation (AFMC) implements the following steps:

1. Partition the present frame into triangular patches.
2. Estimate the motion vectors at the grid points in frame k. At the first stage, a rough estimate of the motion vectors can be obtained by using a standard block matching algorithm with rectangular blocks centered at the grid points. These estimates are then refined by the hexagonal search, which is a connectivity-preserving search procedure [Nak 94].
3. Determine the affine mapping parameters for each triangle given the displacement vectors at its vertices. Synthesize the present frame by mapping the color information from the previous reconstructed frame onto the corresponding patches in the present frame. Compute the synthesis error.
4. Encode both the motion vectors at the grid points and the synthesis error.

Note that no shape information needs to be transmitted in this approach. The number of grid points and the bit allocation for the transmission of synthesis error (color information) vary with the available bitrate. For example, at very low bitrates (under 10 kbps) no color information is usually transmitted. Segmentation of the current frame with adaptive patch boundaries that coincide with the boundaries of the moving objects (which would necessitate transmission of additional shape information) is left as a future research topic.

24.2 Knowledge-Based and Semantic Methods

Estimation of 3-D motion and structure of an unknown 3-D object from two frames (the case of 2-D/3-D rigid objects with 3-D motion) is a difficult problem, in that it requires high computational complexity, and the solution is very sensitive to noise. On the other hand, methods based on the simpler 2-D models, that is, the 2-D flexible model with translational motion and the affine transform using triangular patches, make certain assumptions that may not be satisfied. For example, in the former case motion may not be entirely translational, and in the latter, some triangles may cover two different objects. Knowledge-based 3-D modeling offers a compromise. It is applicable to cases where we have a priori information about the content of the scene in the form of a wireframe model, which is a mesh model composed of a set of triangular planar patches that are connected. However, the solution is considerably simpler than the case of unknown 3-D objects.

In knowledge-based coding, it is assumed that generic wireframe models have been designed off-line for certain objects of interest, which are available at both the transmitter (encoder) and the receiver (decoder). The encoder selects a suitable wireframe model for a particular scene, which is then scaled according to the size of the object in the reference frame. The motion of the object can then be described by the displacement of the vertices of the wireframe model. The objects can be modeled as moving 3-D rigid objects, where the global motion of all vertices on the same object can be characterized by a single set of six rigid motion parameters, or moving 3-D flexible objects (flexibly connected rigid components), where the wireframe model undergoes local motion deformations. Local motion deformations can be described by a set of motion parameters for each individual patch, where the vertices may move semi-independently under the geometrical constraints of a connected mesh model [Boz 94]. Alternatively, in the case of facial images, they can be described by semantic modeling techniques [Cho 94].

One of the main applications of model-based coding has been videophone, where scenes are generally restricted to head-and-shoulder types. In designing a wireframe model, the first step is to obtain the depth map of the speaker's head and shoulders, usually by scanning the speaker using collimated laser light. Once the depth map is obtained, the 3-D wireframe model is obtained through a triangularization procedure where small triangles are used in high-curvature areas and larger ones at low-curvature areas. The wireframe model is stored in the computer as a set of linked arrays. One set of arrays lists the X_1, X_2, X_3 coordinates of each vertex and another set gives the addresses of the vertices forming each triangle. There are several wireframe models used by different research groups. An extended version of the CANDIDE model [Ryd 87] is shown in Figure 24.5.

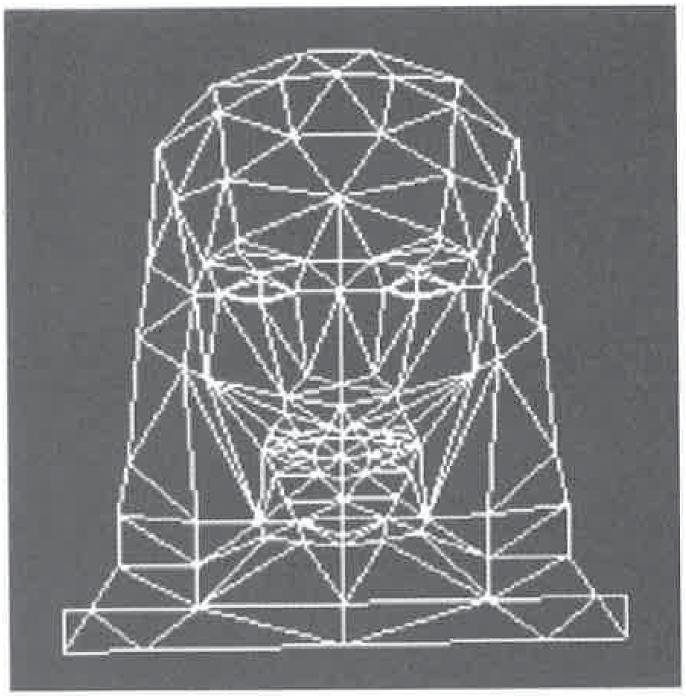


Figure 24.5: Wireframe model of a typical head-and-shoulder scene [Wel 91].

In the following, we first discuss the basic principles of the knowledge-based approach. Then we present two specific algorithms, the MBASIC algorithm [Aiz 93], and a more sophisticated adaptive scaling and tracking algorithm [Boz 94].

24.2.1 General Principles

A block diagram of the 3-D knowledge-based coding scheme is shown in Figure 24.6. The encoder is composed of four main components: i) image analysis module, which includes scaling of the 3-D wireframe model, global and local motion estimation, ii) image synthesis module, which includes texture mapping, iii) model update, which is updating of the coordinates of the wireframe model and the texture information, and iv) parameter coding. Some of these steps are described below.

1. *Wireframe model fitting*

   The accuracy of tracking the motion of the wireframe model from frame to frame strongly depends on how well the wireframe model matches the actual

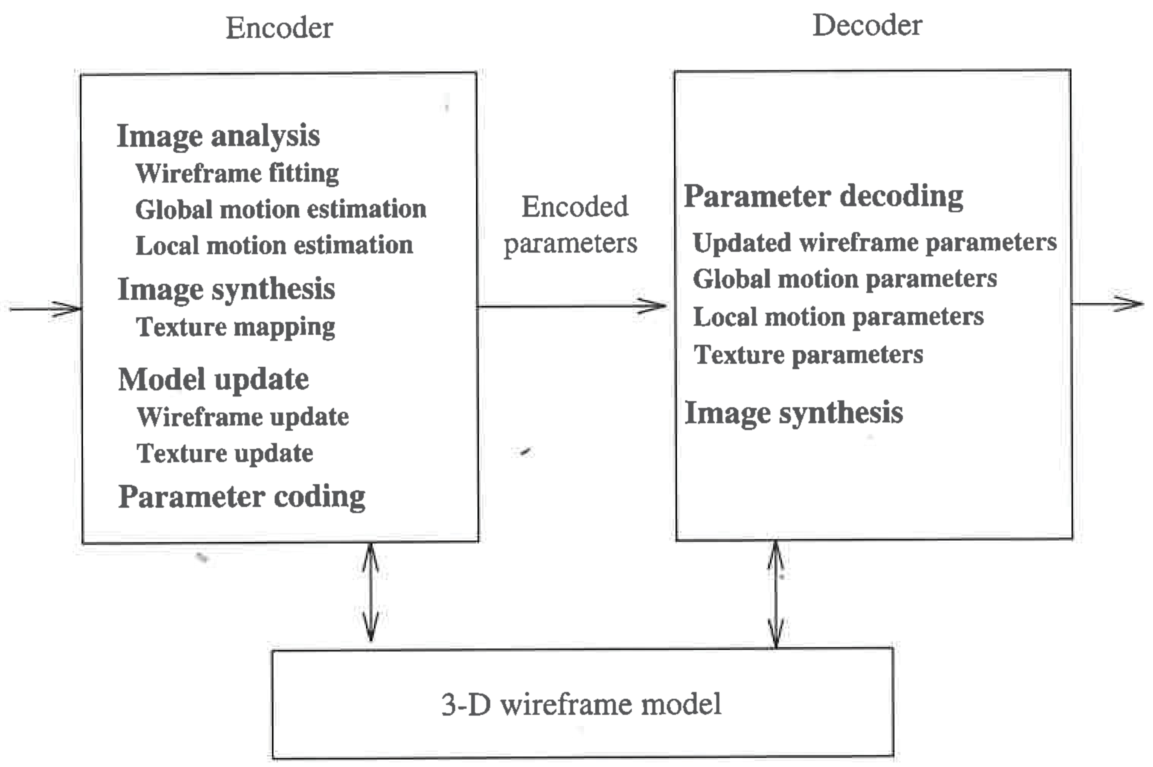


Figure 24.6: Block diagram of knowledge-based coding.

speaker in the scene. Since the size and shape of the head and the position of the eyes, mouth, and nose vary from person to person, it is necessary to modify the 3-D wireframe model to fit the actual speaker. Thus, the first step in 3-D knowledge-based coding of a facial image sequence is to adapt a generic wireframe model to the actual speaker.

Initial studies on 3-D model-based coding have scaled the wireframe model by fitting the orthographic projection of the wireframe model to a frontal view of the actual speaker by means of affine transformations and a set of manually selected feature points [Aiz 89, Aiz 93, Kan 91]. The four feature points used by Aizawa *et al.* [Aiz 89, Aiz 93], tip of the chin, temples, and a point midway between the left and right eyebrows, are shown in Figure 24.7. The points (x_1', x_2') in frame k that correspond to the selected feature points (x_1, x_2) on the orthographic projection of the wireframe model are marked interactively. The parameters of an affine transform, $x_1' = ax_1 + bx_2 + c$ and $x_2' = dx_1 + ex_2 + f$, are then estimated using a least squares procedure to obtain the best fit at the selected feature points. This transformation is subsequently applied to the coordinates of all vertices for scaling. The depth at each vertex is modified according to the scaling factor $\sqrt{(a^2 + e^2)/2}$. In an attempt to automatic scaling, Huang *et al.* [Hua 91] propose using

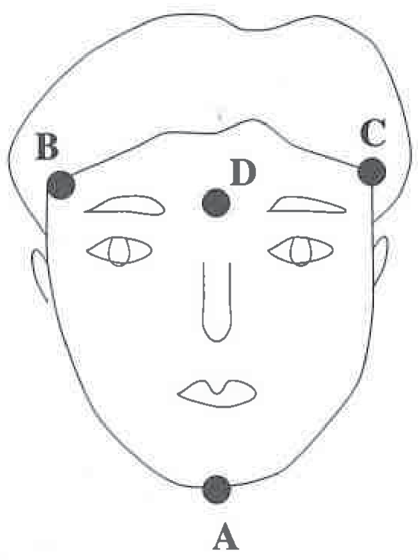


Figure 24.7: Feature points to be used in scaling the wireframe.

spatial and temporal gradients of the image to estimate the maximum height and width of the actual face and scale the wireframe model accordingly.

An alternative approach is to use snakes or ellipses to model the boundary of the face. Recently, Reinders *et al.* [Rei 92] consider automated global and local modification of the 2-D projection of the wireframe model in the x_1 and x_2 directions. They segment the image into background, face, eyes, and mouth, and approximate the contours of the face, eyes, and mouth with ellipses. Then local transformations are performed using elastic matching techniques. Waite and Welsh use snakes to find the boundary of the head, which is claimed to be a robust method [Wel 90]. However, all of the above methods have applied an approximate scaling in the z-direction (depth) since they are based on a single frame.

2. *Motion Analysis*

The facial motion can be analyzed into two components, the global motion of the head and the local motion due to facial expressions, such as motion of the mouth, eyebrows, and eyes. The global motion of the head can be characterized by the six rigid motion parameters. These parameters can be estimated using point correspondence or optical flow based approaches, which were discussed in Chapters 9-12. The MBASIC and the flexible-wireframe-model based approaches that are presented in the next two subsections are representative of the respective approaches. These methods also provide depth estimates at the selected feature points, which may be used for improved scaling of the depth parameters of the generic wireframe model.

A popular approach to characterize local motion is to describe it in terms of the so-called facial action units (AUs) [Aiz 93, Cho 94] which are based on the Facial Action Coding system (FACS) [Ekm 77]. FACS describes facial expressions in terms of AUs that are related to movement of single muscles or clusters of muscles. According to FACS, a human facial expression can be divided into approximately 44 basic AUs, and all facial expressions can be synthesized by an appropriate combination of these AUs. Several algorithms have been proposed to analyze local motion using AUs [For 89, Kan 91, Aiz 93, Li 93, Cho 94]. Among these Forchheimer [For 89] used a least squares estimation procedure to find the best combination of AUs to fit the residual motion vector field (the displacement field after compensating for the global motion). That is, the vector **a** of AU parameters is estimated from

$$\Delta d = Aa$$

where $\mathbf{\Delta d}$ denotes the residual displacement vector field, and **A** is the matrix of displacement vectors for each AU. Recently, Li *et al.* [Li 93] proposed a method to recover both the local and global motion parameters simultaneously from the spatio-temporal derivatives of the image using a similar principle.

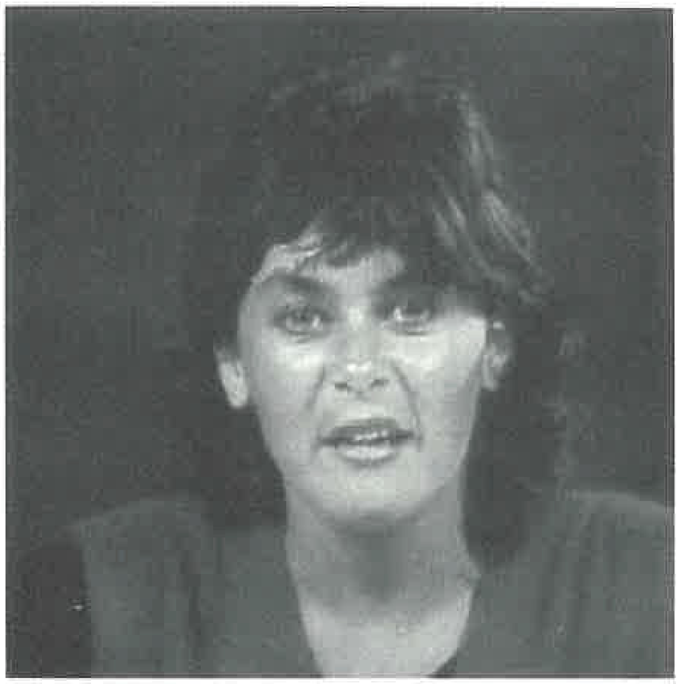


Figure 24.8: Demonstration of facial expression synthesis using AUs.

Facial action units are demonstrated in Figure 24.8 using a frame of the sequence "Miss America." The picture illustrates the synthesis of the AUs 2, 17, and 46, corresponding to "outer brow raiser," "chin raiser," and "blinking." Deformable contour models have also been used to track the nonrigid motion of facial features [Ter 90]. Huang *et al.* [Hua 91] used splines to track features, such as eyes, eyebrows, nose and the lips.

3. *Synthesis*

   Synthesis of facial images involves transformation of the wireframe model according to a particular set of global and local motion parameters followed by texture mapping. Texture mapping is a widely studied topic in computer graphics to obtain realistic images [Yau 88, Aiz 93]. It refers to mapping a properly warped version of the texture observed in the first frame onto the surface of the deformed wireframe model.

   Texture mapping under orthographic projection can be described as follows:
   i) collapse the initial wireframe model onto the image plane (by means of orthographic projection) to obtain a collection of triangles,
   ii) map the observed texture in the first frame into the respective triangles,
   iii) rotatate and translate the initial wireframe model according to the given global and local motion parameters, and then collapse again to obtain a set of deformed triangles for the next frame, and
   iv) map the texture within each triangle in the first frame into the corresponding triangle by means of appropriate decimation or interpolation.

   Texture mapping is illustrated in Figure 24.9, where Figure 24.9.a shows one of the collapsed triangles in the initial frame. Figure 24.9.b depicts the corresponding triangle with its appropriately warped texture. Texture warping can be accomplished with orthographic or perspective projection techniques.

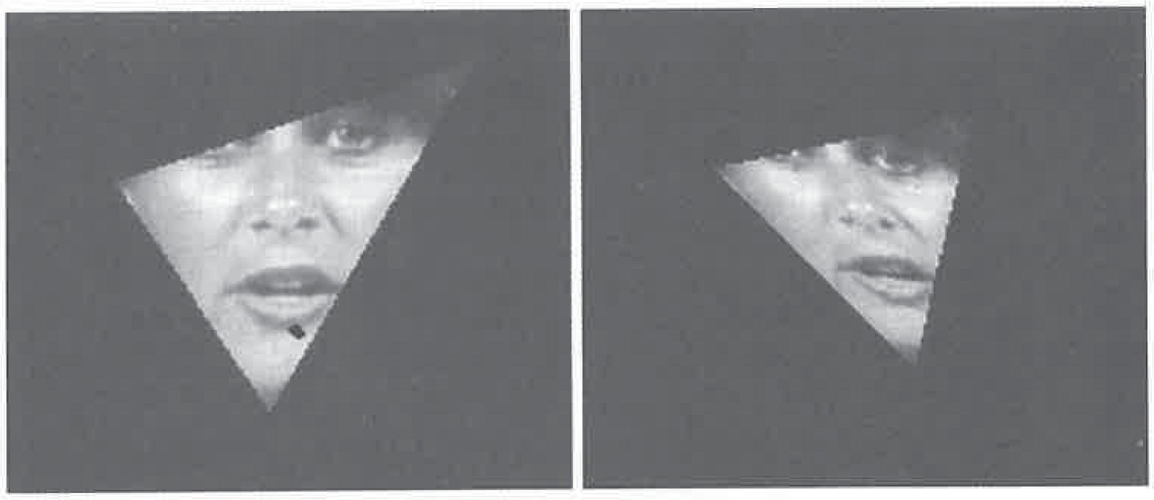

(a) (b)

Figure 24.9: Texture mapping: a) before and b) after processing.

3-D knowledge-based coding schemes have been shown to yield higher compression ratios when applied to typical videophone scenes than do waveform coding and/or 2-D object-based coding, since 3-D models provide a more compact description of the scene for these special class of images. Clearly, knowledge-based schemes offer a compromise between generality and higher compression ratio. Practical implementations of object-based schemes usually feature a fall-back mode, such as DCT frame coding, to cope with frame-to-frame accumulation of image analysis and synthesis errors or sudden scene changes.

24.2.2 MBASIC Algorithm

The MBASIC algorithm is a simple knowledge-based coding scheme that is based on the source model of a known wireframe model (up to a scaling factor) subject to 3-D rigid motion (to model the global motion of the head). Facial action unit analysis has been used to model the local motion deformations. The algorithm can be summarized through the following steps [Aiz 89]:

- *Scale the wireframe model.* First, an edge detection is performed to find the boundaries of the face. Certain extreme points on the edge map, such as the corners of the two ears, the middle of the chin, and the forehead, are then detected to compute the maximum horizontal and vertical distances. The x_1 and x_2 scaling factors are then calculated to match the respective distances on the generic wireframe model to these values. The scaling factor in the X_3 direction is approximated by the mean of the x_1 and x_2 scaling factors

- *Determine N matching point pairs.* Mark seven to ten characteristic points on the rigid parts of the face, such as tip of the nose, around the eye brows, etc., on the initial frame. Find the best matching points in the next frame using block matching or another technique.

- *Estimate the global 3-D motion and structure parameters.* The two-step iteration process described in Chapter 9 is used to estimate the 3-D motion and structure parameters. That is, we initialize the unknown depth parameters using the values obtained from the scaled wireframe model. Next, we estimate the 3-D motion parameters given the depth parameters from (9.10), and estimate the depth parameters from (9.11) using the 3-D motion parameters estimated in the previous step, iteratively.

- *Estimate action units.* After compensating for the global motion of the head, the AUs that best fit the residual motion field can be estimated.

Since scaling of the wireframe model in the X_3 (depth) direction is approximate (because it is based on a single frame), there is an inevitable mismatch of the initial depth (X_3) parameters of the wireframe model and the actual speaker in the image sequence. The two-step motion estimation procedure has been found to be sensitive to these errors if it exceeds 10%. To overcome this problem, an improved iterative algorithm (see Section 9.2.2) can be employed for 3-D motion and structure parameter estimation in the above procedure.

24.2.3 Estimation Using a Flexible Wireframe Model

Many existing methods consider fitting (scaling) a generic wireframe to the actual speaker using only the initial frame of the sequence [Aiz 89, Rei 92]. Thus, the scaling in the z-direction (depth) is necessarily approximate. Furthermore, although the utility of photometric cues in 3-D motion and structure estimation are well known [Ver 89, Pea 90, Pen 91, Dri 92], photometric information has not commonly been used in the context of motion estimation in knowledge-based coding. In this section, we introduce a recent formulation where the 3-D global and local motion estimation and the adaptation of the wireframe model are considered simultaneously within an optical-flow-based framework, including the photometric effects of motion. The adaptation of the wireframe model serves two purposes that cannot be separated: to reduce the misfit of the wireframe model to the speaker in frame $k-1$, and to account for the local motion from frame $k-1$ to frame k without using any *a priori* information about the AUs.

The source model is a flexible wireframe model whose local structure is characterized by the normal vectors of the patches which are related to the coordinates of the nodes. Geometrical constraints that describe the propagation of the movement of the nodes are introduced, which are then efficiently utilized to reduce the number of independent structure parameters. A stochastic relaxation algorithm has been used to determine optimum global motion estimates and the parameters describing the structure of the wireframe model. The simultaneous estimation formulation is motivated by the fact that estimation of the global and local motion and adaptation of the wireframe model, including the depth values, are mutually related; thus, a combined optimization approach is necessary to obtain the best results. Because an optical-flow-based criterion function is used, computation of the synthesis error is not necessary from iteration to iteration, and thus, results in an efficient implementation. The synthesis error at the conclusion of the iterations is used to validate the estimated parameters, and to decide whether a texture update is necessary.

In the following we summarize the estimation of the illuminant direction, the formulation of the simultaneous motion estimation and wireframe adaptation problem including the photometric effects of motion, and the algorithm for the proposed simultaneous estimation, including an efficient method to update the nodes of the wireframe model.

Estimation of the Illuminant Direction

Photometric effects refer to the change in shading due to 3-D motion of the object. For example, in the case of rotational motion, because the surface normals change, the shading of the objects varies even if the external illumination remains constant. Recently, Pentland [Pen 91] showed that the changes in image intensity because of photometric effects can dominate intensity changes due to the geometric effects of motion (changes in projected surface geometry due to 3-D motion). Similar discussions can be found in [Ver 89, Pea 90, Dri 92]. Here we briefly discuss estimation of the illuminant direction with the aim of incorporating photometric effects into the above optical-flow-based formulation.

Recall from Section 2.3 that the image intensity $s_c(x_1, x_2, t)$ can be expressed as

$$s_c(x_1, x_2, t) = \rho \mathbf{N}(t) \cdot \mathbf{L} \tag{24.1}$$

where $\mathbf{L} = (L_1, L_2, L_3)$ is the unit vector in the mean illuminant direction and $\mathbf{N}$ is the unit surface normal of the scene at position $(X_1, X_2, X_3(X_1, X_2))$ given by

$$\mathbf{N}(t) = (-p, -q, 1)/(p^2 + q^2 + 1)^{1/2} \tag{24.2}$$

and $p = \frac{\partial X_3}{\partial x_1}$ and $q = \frac{\partial X_3}{\partial x_2}$ are the partial derivatives of depth $X_3(x_1, x_2)$ with respect to the image coordinates x_1 and x_2, respectively.

Note that the illuminant direction $\mathbf{L}$ can also be expressed in terms of tilt and slant angles as

$$\mathbf{L} = (L_1, L_2, L_3) = (\cos\tau \sin\sigma, \sin\tau \sin\sigma, \cos\sigma) \tag{24.3}$$

where τ, the tilt angle of the illuminant, is the angle between $\mathbf{L}$ and the $X_1 - X_3$ plane, and σ, the slant angle, is the angle between $\mathbf{L}$ and the positive X_3 axis. In order to incorporate the photometric effects of motion into 3-D knowledge-based coding, the illuminant direction $\mathbf{L}$ must be known or estimated from the available frames.

A method to estimate the tilt and slant angles of the illuminant, based on approximating the 3-D surface by spherical patches, was proposed by Zheng *et al.* [Zhe 91]. They estimate the tilt angle as

$$\tau = \arctan\left(\frac{E\{\hat{L}_1/\sqrt{\hat{L}_1^2 + \hat{L}_2^2}\}}{E\{\hat{L}_2/\sqrt{\hat{L}_1^2 + \hat{L}_2^2}\}}\right) \tag{24.4}$$

where $E\{.\}$ denotes expectation over the spatial variables (which is approximated by 4×4 local averaging), and $\hat{L}_1$ and $\hat{L}_2$ are the x_1 and x_2 components of the local estimate of the tilt of the illuminant, respectively, computed as

$$\begin{bmatrix} \hat{L}_1 \\ \hat{L}_2 \end{bmatrix} = (B^t B)^{-1} B^t \begin{bmatrix} \delta I_1 \\ \delta I_2 \\ \vdots \\ \delta I_N \end{bmatrix}, \quad \text{and} \quad B = \begin{bmatrix} \delta x_{11} & \delta x_{21} \\ \delta x_{12} & \delta x_{22} \\ \vdots & \vdots \\ \delta x_{1N} & \delta x_{2N} \end{bmatrix}$$

Here, δI_i, $i = 1, \ldots, N$ is the difference in image intensity along a particular direction $(\delta x_{1i}, \delta x_{2i})$, and N is the number of directions (typically $N = 8$ for each 4×4 window).

The slant angle σ can be uniquely estimated from

$$\frac{E\{I\}}{E\{I^2\}} = f_3(\sigma) \tag{24.5}$$

since $f_3(\sigma)$ (defined in [Zhe 91]) is a monotonically decreasing function of σ, where $E\{I\}$ and $E\{I^2\}$ are the expected values of the image intensities and the square of the image intensities, respectively, estimated from the image area where the wireframe model is fitted.

Finally, the surface albedo can be estimated from

$$\rho = \frac{E\{I\} \cdot f_1(\sigma) + \sqrt{E\{I^2\} \cdot f_2(\sigma)}}{f_1^2(\sigma) + f_2(\sigma)} \tag{24.6}$$

where $f_1(\sigma)$ and $f_2(\sigma)$ are seventh-order polynomials in $\cos\sigma$ as defined in [Zhe 91].

Incorporation of the Photometric Effects into the OFE

Since we represent the 3-D structure of a head-and-shoulders scene by a wireframe model and the surface of the wireframe model is composed of planar patches, the variation in the intensity of a pixel due to photometric effects of motion will be related to a change in the normal vector of the patch to which this pixel belongs.

Assuming that the mean illuminant direction $\mathbf{L} = (L_1, L_2, L_3)$ remains constant, we can represent the change in intensity due to photometric effects of motion, based on the photometric model (2.32), as

$$\frac{ds_c(x_1, x_2, t)}{dt} = \rho \mathbf{L} \cdot \frac{\mathbf{dN}}{\mathbf{dt}} \tag{24.7}$$

Then, substituting Equations (24.7) and (2.33) into the optical flow equation (5.5), and expressing 2-D velocities in terms of the 3-D motion parameters, we include the photometric effects into the optical-flow-based formulation as [Boz 94]

$$\frac{\partial s_c}{\partial x_1}(\Omega_3 x_2 - \Omega_2 X_3 + V_1) + \frac{\partial s_c}{\partial x_2}(-\Omega_3 x_1 + \Omega_1 X_3 + V_2) + \frac{\partial s_c}{\partial t} =$$
$$\rho \mathbf{L} \cdot \left[\frac{(-p', -q', 1)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p, -q, 1)^T}{\sqrt{p^2 + q^2 + 1}} \right] \tag{24.8}$$

The term on the right-hand side of Equation (24.8) may be significant, especially if the change in the surface normal has components either toward or away from the illuminant direction [Pen 91].

Structure of the Wireframe Model

Next, we introduce geometrical constraints about the structure of the wireframe model and discuss formulation of the simultaneous estimation problem. The wireframe model is composed of triangular patches which are characterized by the

(X_1, X_2, X_3) coordinates of their respective vertices. Given the (X_1, X_2, X_3) coordinates of the vertices of a patch, we can write the equation of the plane containing this patch. Let $P_1^{(i)} = (X_{11}^{(i)}, X_{21}^{(i)}, X_{31}^{(i)})$, $P_2^{(i)} = (X_{12}^{(i)}, X_{22}^{(i)}, X_{32}^{(i)})$ and $P_3^{(i)} = (X_{13}^{(i)}, X_{23}^{(i)}, X_{33}^{(i)})$ denote the vertices of the ith patch, and $P^{(i)} = (X_1^{(i)}, X_2^{(i)}, X_3^{(i)})$ be any point on this patch. Then,

$$\overrightarrow{P^{(i)}P_1^{(i)}} \cdot (\overrightarrow{P_2^{(i)}P_1^{(i)}} \times \overrightarrow{P_3^{(i)}P_1^{(i)}}) = 0$$

gives the equation of the plane containing $P_1^{(i)}$, $P_2^{(i)}$ and $P_3^{(i)}$, where $\overrightarrow{P^{(i)}P_1^{(i)}}$, $\overrightarrow{P_2^{(i)}P_1^{(i)}}$, and $\overrightarrow{P_3^{(i)}P_1^{(i)}}$ are the vectors from the former point to the latter, respectively. We can express this equation in the form

$$X_3^{(i)} = p_i X_1^{(i)} + q_i X_2^{(i)} + c_i \tag{24.9}$$

where

$$p_i = -\frac{(X_{22}^{(i)} - X_{21}^{(i)})(X_{33}^{(i)} - X_{31}^{(i)}) - (X_{32}^{(i)} - X_{31}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)})}{(X_{12}^{(i)} - X_{11}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)}) - (X_{22}^{(i)} - X_{21}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)})}$$

$$q_i = -\frac{(X_{32}^{(i)} - X_{31}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)}) - (X_{12}^{(i)} - X_{11}^{(i)})(X_{33}^{(i)} - X_{31}^{(i)})}{(X_{12}^{(i)} - X_{11}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)}) - (X_{22}^{(i)} - X_{21}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)})}$$

and

$$\begin{aligned} c_i &= X_{31}^{(i)} + X_{11}^{(i)} \frac{(X_{22}^{(i)} - X_{21}^{(i)})(X_{33}^{(i)} - X_{31}^{(i)}) - (X_{32}^{(i)} - X_{31}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)})}{(X_{12}^{(i)} - X_{11}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)}) - (X_{22}^{(i)} - X_{21}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)})} \\ &\quad + X_{21}^{(i)} \frac{(X_{32}^{(i)} - X_{31}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)}) - (X_{12}^{(i)} - X_{11}^{(i)})(X_{33}^{(i)} - X_{31}^{(i)})}{(X_{12}^{(i)} - X_{11}^{(i)})(X_{23}^{(i)} - X_{21}^{(i)}) - (X_{22}^{(i)} - X_{21}^{(i)})(X_{13}^{(i)} - X_{11}^{(i)})} \end{aligned}$$

Using Equation (24.9), the X_3 coordinate of any point on the ith patch can be expressed in terms of the parameters p_i, q_i, and c_i and the X_1 and X_2 coordinates of the point. Then, we can eliminate X_3 from Equation (24.8) by substituting (24.9) into (24.8) with $X_1^{(i)} = x_1^{(i)}$, $X_2^{(i)} = x_2^{(i)}$, where the patch index i is determined for each (x_1, x_2) according to the orthographic projection.

It is important to note that the parameters p_i, q_i, and c_i of each planar patch of the wireframe are not completely independent of each other. Each triangular patch is either surrounded by two (if it is on the boundary of the wireframe) or three other triangles. The requirement that the neighboring patches must intersect at a straight line imposes a constraint on the structure parameters of these patches in the form

$$p_i x_1^{(ij)} + q_i x_2^{(ij)} + c_i = p_j x_1^{(ij)} + q_j x_2^{(ij)} + c_j \tag{24.10}$$

where p_j, q_j, and c_j denote the parameters of the jth patch, and $(x_1^{(ij)}, x_2^{(ij)})$ denote the coordinates of a point that lie at the intersection of the ith and jth patches.

Problem Statement

The 3-D global motion parameters Ω_1, Ω_2, Ω_3, V_1, V_2, and the structure parameters p_i, q_i, c_i, can be simultaneously estimated by minimizing the sum square error in the optical flow equation (24.8) over all pixels in a frame, given by

$$E = \sum_i \sum_{(x_1,x_2) \in i^{th} patch} e_i^2(x_1, x_2) \tag{24.11}$$

where

$$\begin{aligned} e_i(x_1, x_2) &= \frac{\partial s_c}{\partial x_1}(\Omega_3 x_2 - \Omega_2(p_i x_1 + q_i x_2 + c_i) + V_1) \\ &+ \frac{\partial s_c}{\partial x_2}(-\Omega_3 x_1 + \Omega_1(p_i x_1 + q_i x_2 + c_i) + V_2) + \frac{\partial s_c}{\partial t} \\ &-\rho(L_1, L_2, L_3) \cdot \left(\frac{(-\frac{-\Omega_2+p_i}{1+\Omega_2 p_i}, -\frac{\Omega_1+q_i}{1-\Omega_1 q_i}, 1)}{((\frac{-\Omega_2+p_i}{1+\Omega_2 p_i})^2 + (\frac{\Omega_1+q_i}{1-\Omega_1 q_i})^2 + 1)^{1/2}} - \frac{(-p_i, -q_i, 1)}{(p_i^2 + q_i^2 + 1)^{1/2}} \right) \end{aligned}$$

with respect to Ω_1, Ω_2, Ω_3, V_1, V_2, p_i, q_i, c_i, and $i = 1, \ldots,$ number of patches, subject to the geometrical constraints given by (24.10). It is assumed that the values of ρ and (L_1, L_2, L_3) are estimated *a priori*. The constraints, Equation (24.10), not only enable us to preserve the structure of the wireframe during the adaptation, but also facilitate reducing the number of independent unknowns in the optimization process, as described in the following.

The Algorithm

The criterion function E, defined by Equation (24.11), can be minimized using a stochastic relaxation algorithm (see Chapter 8) to find the global optima of Ω_1, Ω_2, Ω_3, V_1, V_2, p_i, q_i, c_i. Each iteration consists of perturbing the state of the system in some random fashion in a manner consistent with the constraint equations (24.10). The constraints are enforced as follows: At each iteration cycle, we visit all patches of the wireframe model in sequential order. If, at the present iteration cycle, none of the neighboring patches of patch i has yet been visited (e.g., the initial patch), then p_i, q_i, c_i are all independently perturbed. If only one of the neighboring patches, say patch j, has been visited (p_j, q_j, c_j have already been updated), then two of the parameters, say p_i and q_i, are independent and perturbed. The dependent variable c_i is computed from Equation (24.10) as

$$c_i = p_j x_1^{(ij)} + q_j x_2^{(ij)} + c_j - p_i x_1^{(ij)} - q_i x_2^{(ij)} \tag{24.12}$$

where $(x_1^{(ij)}, x_2^{(ij)})$ is one of the nodes common to both patches i and j that is either in the boundary or has already been updated in the present iteration cycle. If two of the neighboring patches, say patches j and k, have already been visited, i.e., the variables p_j, q_j, c_j and p_k, q_k, c_k have been updated, then only one variable, say p_i, is independent and perturbed. In this case, c_i can be found from Equation (24.12), and q_i can be evaluated as

$$q_i = \frac{p_k x_1^{(ik)} + q_j x_2^{(ik)} + c_k - p_i x_1^{(ik)} - c_k}{x_2^{(ik)}} \tag{24.13}$$

where $(x_1^{(ik)}, x_2^{(ik)})$ is one of the nodes common to both patches i and k that is either in the boundary or has already been updated in the present iteration cycle.

The perturbation of the structure parameters p_i, q_i, and c_i for each patch i results in a change in the coordinates of the nodes of the updated wireframe. The new coordinates $(X_1^{(n)}, X_2^{(n)}, X_3^{(n)})$ of the node n can be computed given the updated structure parameters of three patches that intersect at node n. Let the patches i, j, and k intersect at node n. Then the relations

$$\begin{aligned} p_i X_1^{(n)} + q_i X_2^{(n)} + c_i &= p_j X_1^{(n)} + q_j X_2^{(n)} + c_j \\ p_i X_1^{(n)} + q_i X_2^{(n)} + c_i &= p_k X_1^{(n)} + q_k X_2^{(n)} + c_k \end{aligned} \tag{24.14}$$

specify $X_1^{(n)}$ and $X_2^{(n)}$. Therefore,

$$\begin{bmatrix} X_1^{(n)} \\ X_2^{(n)} \end{bmatrix} = \begin{bmatrix} p_i - p_j & p_j - p_k \\ q_i - q_j & q_j - q_k \end{bmatrix}^{-1} \begin{bmatrix} c_j - c_i \\ -c_j + c_k \end{bmatrix} \tag{24.15}$$

The new $X_3^{(n)}$ can be computed from Equation (24.9) given $X_1^{(n)}$, $X_2^{(n)}$, and the p_i, q_i, c_i for any patch passing through that node. It is this updating of the coordinates of the nodes that allows the adaptation of the wireframe model to lower the misfit error and accommodate the presence of local motion, such as the motion of the eyes and the mouth.

The proposed algorithm can be summarized as follows:

1. Estimate the illumination direction using Equations (24.4) and (24.5), and the surface albedo using Equation (24.6).

2. Initialize the coordinates $(X_1^{(n)}, X_2^{(n)}, X_3^{(n)})$, of all nodes n, using an approximately scaled initial wireframe model. Determine the initial values of p_i, q_i, and c_i for all patches. Set the iteration counter $m = 0$.

3. Determine the initial motion estimates using Equation (9.10) based on a set of selected feature correspondences and their depth values obtained from the wireframe model.

4. Compute the value of the cost function E given by (24.11).

5. If $E < \epsilon$, stop.

   Else, set $m = m + 1$, and

Perturb the motion parameters $\mathbf{\Omega} = [\Omega_1\ \Omega_2\ \Omega_3\ V_1\ V_2]^T$ as

$$\mathbf{\Omega}^{(m)} \longleftarrow \mathbf{\Omega}^{(m-1)} + \alpha^m \mathbf{\Delta}, \tag{24.16}$$

where the components of $\mathbf{\Delta}$ are zero-mean Gaussian with the variance $\sigma^{2^{(m)}} = E$; and the structure parameters p_i, q_i, and c_i through the procedure:

```
Define count_i as the number of neighboring patches to patch i
whose structure parameters have been perturbed. Set count_i=0,
for all patches i.
Perturb p_1, q_1, c_1 as
```

$$\begin{aligned} p_1^{(m)} &\longleftarrow p_1^{(m-1)} + \alpha^m \Delta_1 \\ q_1^{(m)} &\longleftarrow q_1^{(m-1)} + \alpha^m \Delta_1 \\ c_1^{(m)} &\longleftarrow c_1^{(m-1)} + \alpha^m \Delta_1 \end{aligned} \tag{24.17}$$

where $\Delta_i = N_i(0, \sigma_i^{2^{(m)}})$, i.e., zero mean Gaussian with variance $\sigma_i^{2^{(m)}}$, where $\sigma_i^{2^{(m)}} = \sum_{(x,y)\epsilon \text{patch } i} e_i^2(x,y)$.

```
increment count_j, for all j denoting neighbors of patch 1.
for( i=2 to number of patches)
  {
  if(count_i==1) {
    Perturb p_i and q_i.
    Increment count_m, for all m denoting neighbors of patch i.
    Compute c_i using Equation 24.12, where x_ij and y_ij are
    the coordinates of a fixed or a precomputed node on
    the line of intersection between patches i and j. }

  if(count_i==2) {
    Perturb p_i.
    Increment count_m, for all m denoting neighbors of patch i.
    Compute c_i using Equation 24.12 and q_i using Equation 24.13,
    where (x_ij,y_ij) and (x_ik,y_ik) denote coordinates of
    a fixed or a precomputed node on the line of intersection
    between patches i,j and i,k respectively. }

  If p_i, q_i, and c_i for at least three patches intersecting
  at a node are updated, then update the coordinates of the node
  by using Equation 24.15.
  }
```

6. Go to (4).

The synthesis error, which may be due to (i) misfit of the wireframe, (ii) error in global motion estimation, (iii) error in local motion estimation, and (iv) the photometric effects of the motion, can also be coded and transmitted to improve upon image quality.

24.3 Examples

We have implemented six algorithms using the first and fourth frames of the Miss America sequence (at the frame rate 10 frames/s): a scaled-down version of the H.261 algorithm, the OBASC algorithm (Section 24.1.2), two variations of the AFMC algorithm (Section 24.1.3), the global motion compensation part of the MBASIC algorithm, and the flexible 3-D wireframe model-based algorithm.

The first two algorithms are executed on QCIF frames (176 pixels × 144 lines). The specific implementation of the H.261 coder that we used is the PVRG-P64 coder with the target bitrate of 16 kbps. The reconstructed fourth frame is shown in Figure 24.10. In the OBASC algorithm, the size of the model failure region, hence the amount of color information that needs to be transmitted, is adjusted to meet the same target bitrate. Figure 24.11 (a) and (b) show the model failure region and the reconstructed fourth frame using the OBASC algorithm. Visual comparison of the images confirm that the PVRG-P64 reconstructed image suffers from annoying blocking artifacts. The PSNR values are shown in Table 24.1.

The latter four algorithms are executed on CIF format images. The reconstructed images are then converted into QCIF format to compare their PSNR with those of the first two algorithms. Note that Table 24.1 reports the PSNR for these images in both QCIF and CIF formats. In all cases, model-failure regions are encoded using a DCT based approach with a target bitrate of 16 kbps. In the 3-D model-based algorithms, we have used the extended CANDIDE wireframe, depicted in Figure 24.5, with 169 nodes. The 2-D triangular mesh model that is used in the AFMC has been generated by computing the orthographic projection of CANDIDE into the image plane. The wireframe has been fitted to the first frame by a least squares scaling procedure using 10 preselected control points. Figure 24.12 (a) depicts the wireframe model overlayed onto the original fourth frame. Two alternative approaches have been tried with the AFMC method: i) motion vectors at the nodes of the 2-D mesh are estimated using the Lucas-Kanade (LK) method, which are then used to compute affine motion parameters (AFMC-LK method), and ii) a hexagonal search procedure starting with the LK motion estimates is employed (AFMC-Hex method). The PSNR for both are shown in Table 24.1. The reconstructed fourth frame using the AFMC-Hex method is depicted in Figure 24.12 (b).

Among the 3-D model-based methods, we have implemented the global-motion compensation module of the MBASIC algorithm (Global-3D) and the method based on the 3-D flexible wireframe model (Flexible-3D). The fourth frame after global motion compensation by the five global motion parameters (3 rotation and 2 translation) is depicted in Figure 24.13 (a). Finally, the result obtained by the Flexible-3D method which provides both global and local motion compensation is shown in Figure 24.13 (b). We conclude by noting that the 2-D model-based implementations are more general and robust [Tek 95]. Methods based on the 3-D object models may provide more compact motion representations, with possibly lower accuracy. Note however that, the method based on the flexible 3-D wireframe model facilitates incorporation of photometric effects, which may sometimes prove significant.

Table 24.1: Comparison of model-based motion-compensation results.

Method	PSNR (dB)	
	QCIF	CIF
Frame Difference	33.01	31.24
PVRG-64	34.84	-
OBASC	38.01	-
AFMC-LK	38.19	36.47
AFMC-Hex	41.08	37.40
Global-3D	36.01	34.24
Flexible-3D	38.67	36.53

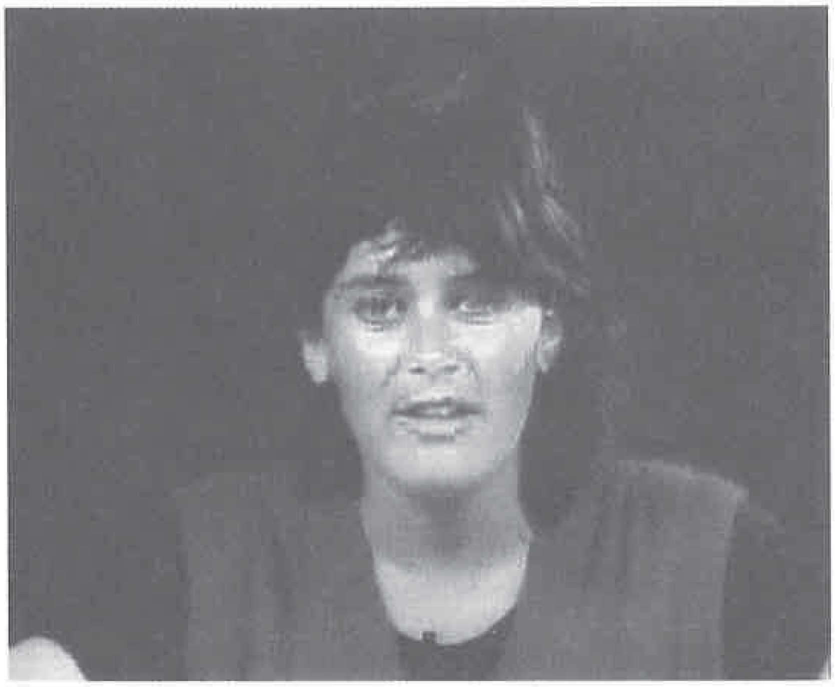
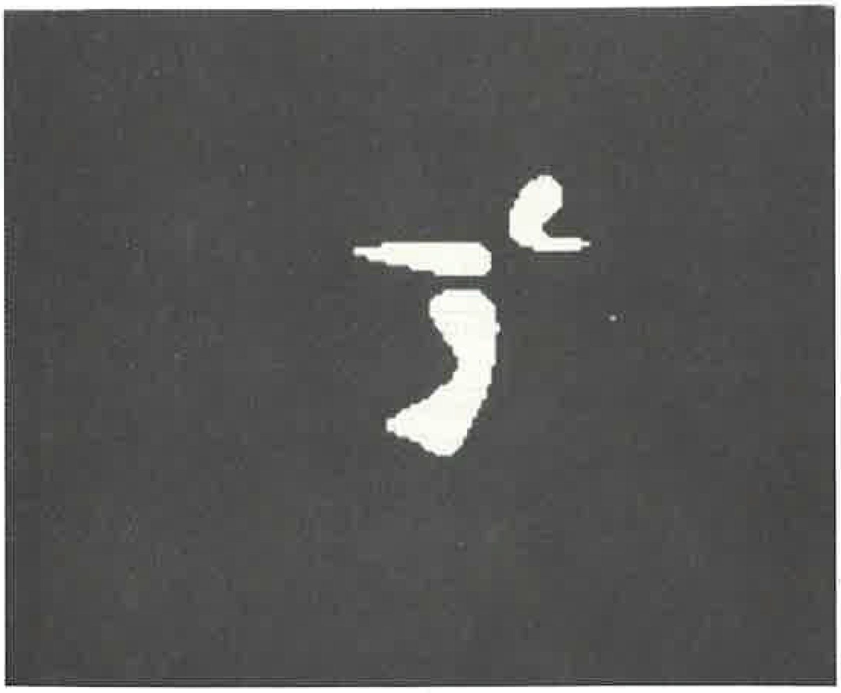
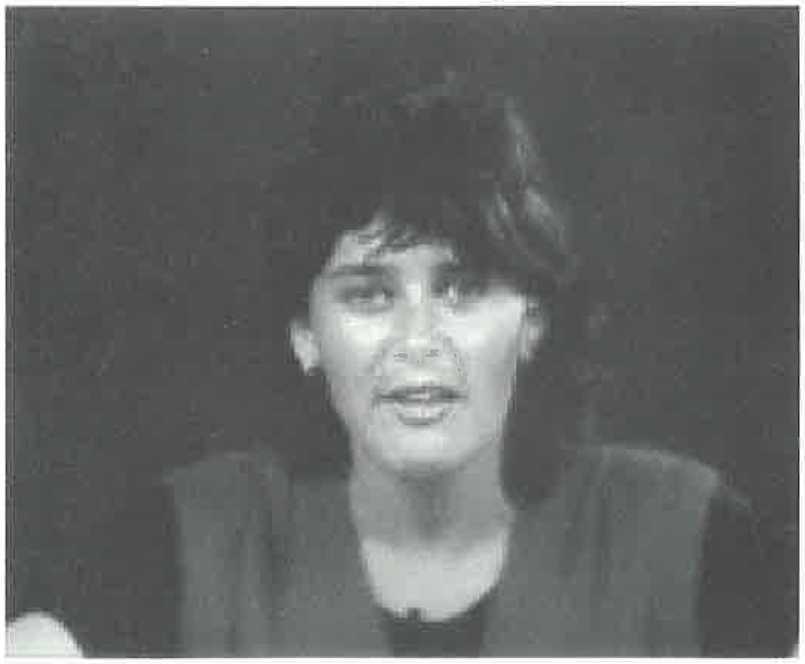


Figure 24.10: A scaled-down implementation of the H.261 algorithm.

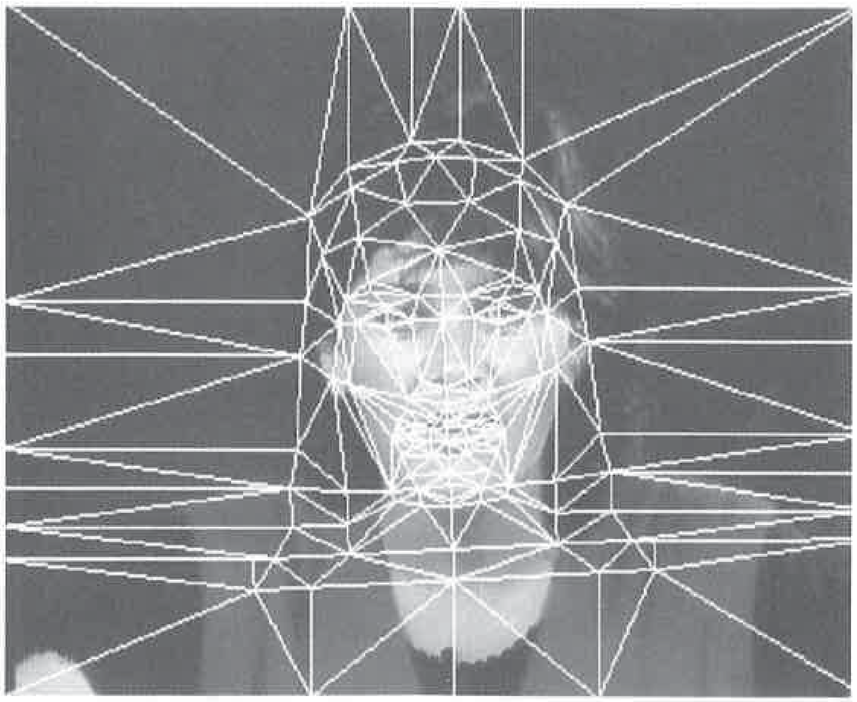


(a)

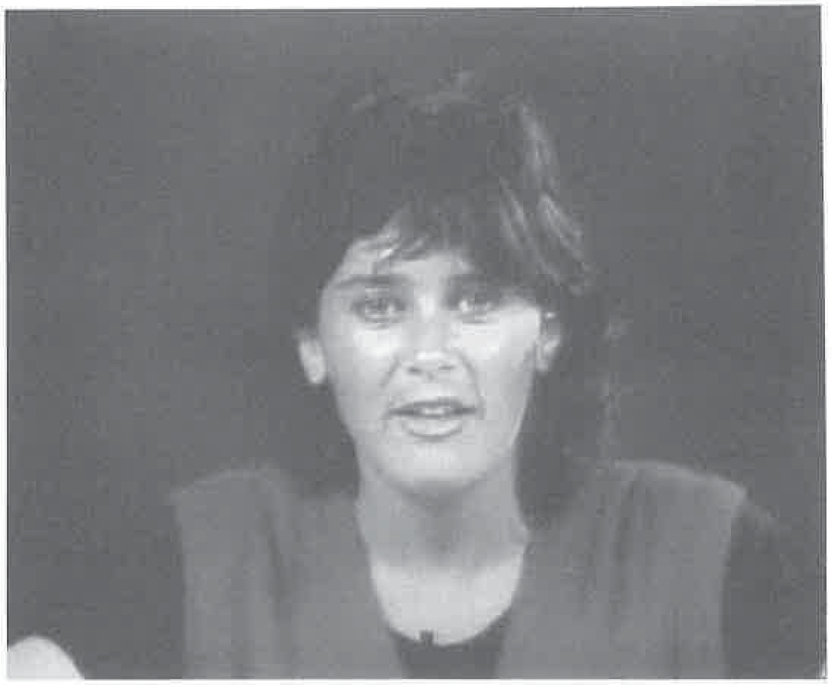


(b)

Figure 24.11: Flexible 2-D object based method: a) the model-failure region and b) the decoded the third frame of Miss America. (Courtesy Francoise Aurtenechea)

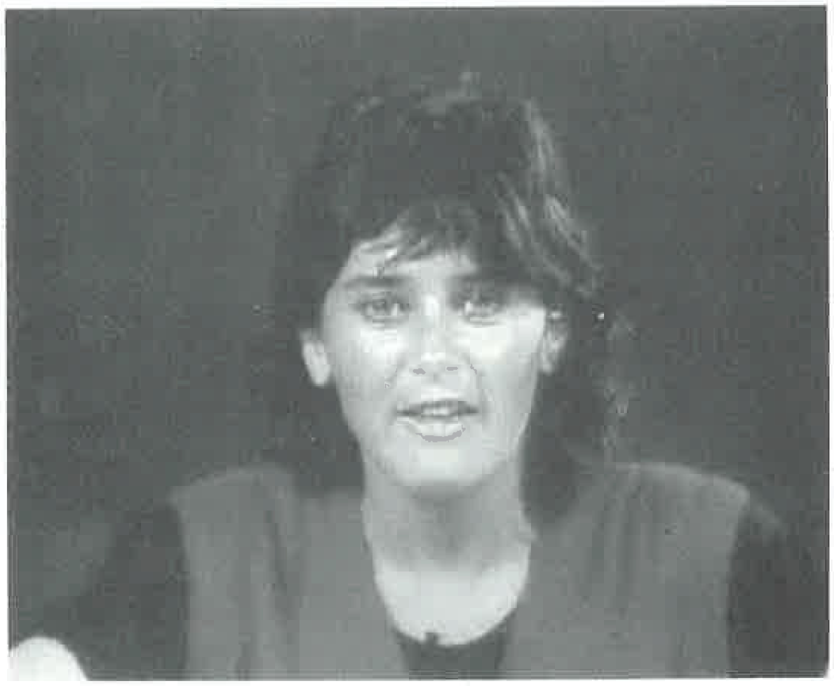


(a)

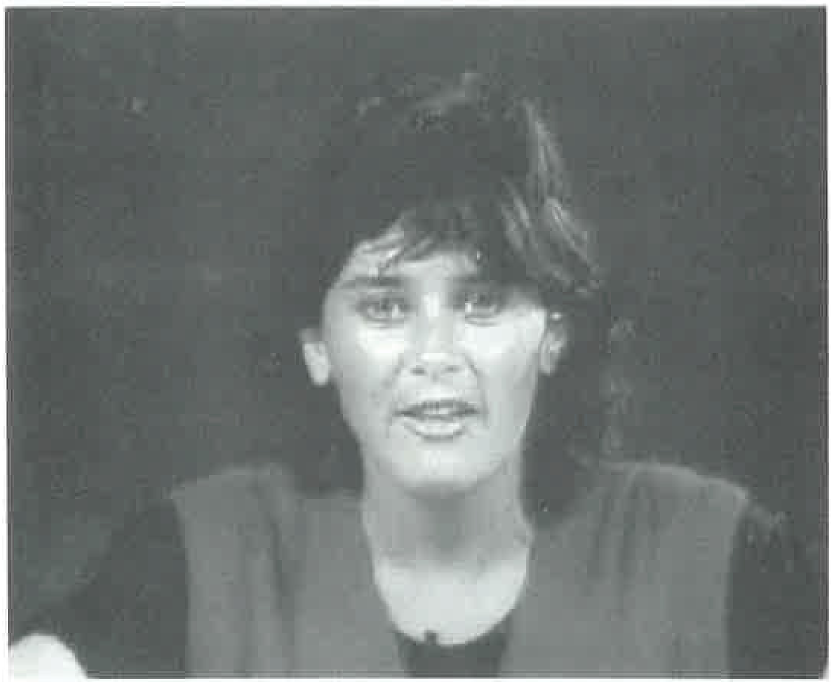


(b)

Figure 24.12: 2-D object-based coding using deformable mesh models: a) an irregular mesh fitted to the first frame of Miss America and b) the decoded third frame. (Courtesy Yucel Altunbasak)



(a)



(b)

Figure 24.13: Decoded third frame of Miss America using 3-D object-based coding: a) global motion compensation using the two-step iteration (part of the MBASIC algorithm) and (b) global and local motion compensation using a flexible wireframe model. (Courtesy Gozde Bozdagi)

Bibliography

[Aiz 89] K. Aizawa, H. Harashima, and T. Saito, "Model-based analysis-synthesis image coding (MBASIC) system for a person's face," *Signal Proc.: Image Comm.*, no. 1, pp. 139–152, Oct. 1989.

[Aiz 93] K. Aizawa, C. S. Choi, H. Harashima, and T. S. Huang, "Human facial motion analysis and synthesis with application to model-based coding," in *Motion Analysis and Image Sequence Processing*, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Aiz 95] K. Aizawa and T. S. Huang, "Model-based image coding: Advanced video coding techniques for very low bit-rate applications," *Proc. IEEE*, vol. 83, no. 2, pp. 259–271, Feb. 1995.

[Boz 94] G. Bozdağı, A. M. Tekalp, and L. Onural, "3-D motion estimation and wireframe adaptation including photometric effects for model-based coding of facial image sequences," *IEEE Trans. Circ. and Syst: Video Tech.*, vol. 4, pp. 246–256, Sept. 1994.

[CCI 90] CCITT Recommendation H.261: "Video Codec for Audiovisual Services at $p \times 64 Kbit/s$," COM XV-R 37-E, 1990.

[Cho 94] C. S. Choi, K. Aizawa, H. Harashima, and T. Takebe, "Analysis and synthesis of facial image sequences in model-based image coding," *IEEE Trans. Circ. and Syst: Video Tech.*, vol. 4, pp. 257–275, Sept. 1994.

[Die 93] N. Diehl, "Model-based image sequence coding," in *Motion Analysis and Image Sequence Processing*, M. I. Sezan and R. L. Lagendijk, eds., Norwell, MA: Kluwer, 1993.

[Ekm 77] P. Ekman and W. V. Friesen, *Facial Action Coding System*, Consulting Psychologist Press, 1977.

[For 89] R. Forchheimer and T. Kronander, "Image coding-from waveforms to animation," *IEEE Trans. Acoust, Speech Sign. Proc.*, vol. 37, no. 12, pp. 2008–2023, Dec. 1989.

[Fuk 93] T. Fukuhara and T. Murakami, "3-D motion estimation of human head for model-based image coding," *IEE Proc.-I*, vol. 140, no. 1, pp. 26–35, Feb. 1993.

[Ger 94] P. Gerken, "Object-based analysis-synthesis coding of image sequences at very low bit rates," *IEEE Trans. Circ. and Syst: Video Tech.*, vol. 4, pp. 228–237, Sept. 1994.

[Hot 90] M. Hotter, "Object oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Proc: Image Comm.*, vol. 2, no. 4, pp. 409–428, Dec 1990.

[Hua 91] T. S. Huang, S. C. Reddy, and K. Aizawa, "Human facial motion modeling, analysis, and synthesis for video compression," *SPIE Vis. Comm. and Image Proc'91*, vol. 1605, pp. 234–241, Nov. 1991.

[Kan 86] K. Kanatani, "Structure and motion from optical flow under orthographic projection," *Comp. Vis. Graph. Image Proc.*, vol. 35, pp. 181–199, 1986.

[Kan 91] M. Kaneko, A. Koike, and Y. Hatori, "Coding of facial image sequence based on a 3D model of the head and motion detection," *J. Visual Comm. and Image Rep.*, vol. 2, no. 1, pp. 39–54, March 1991.

[Kla 86] B. Klaus and P. Horn, *Robot Vision*, Cambridge, MA: MIT Press, 1986.

[Koc 93] R. Koch, "Dynamic 3-D scene analysis through synthesis feedback control," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. 15, pp. 556–568, June 1993.

[Lav 90] F. Lavagetto and S. Zappatore, "Customized wireframe modeling for facial image coding," *Third Int. Workshop on 64kbits/s Coding of Moving Video*, 1990.

[Lee 89] H. Lee and A. Rosenfeld, "Improved methods of estimating shape from shading using light source coordinate system," in *Shape from Shading,* B. K. P. Horn and M. J. Brooks, eds., pp. 323–569, Cambridge, MA: MIT Press, 1989.

[Li 93] H. Li, P. Roivainen, and R. Forchheimer, "3-D motion estimation in model-based facial image coding," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. 15, no. 6, pp. 545–555, June 1993.

[Li 94] H. Li, A. Lundmark, and R. Forchheimer, "Image sequence coding at very low bitrates: A Review," *IEEE Trans. Image Proc.*, vol. 3, pp. 589–609, Sept. 1994.

[Mor 91] H. Morikawa and H. Harashima, "3D structure extraction coding of image sequences," *J. Visual Comm. and Image Rep.*, vol. 2, no. 4, pp. 332–344, Dec. 1991.

[Mus 89] H. G. Musmann, M. Hotter, and J. Ostermann, "Object-oriented analysis synthesis coding of moving images," *Signal Proc: Image Comm.*, vol. 1, pp. 117–138, 1989.

[Nak 94] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations," *IEEE Trans. Circ. and Syst.: Video Tech.*, vol. 4, pp. 339–356, June 1994.

[Ost 90] J. Ostermann, "Modeling of 3D moving objects for an analysis-synthesis coder," in *Sensing and Reconstruction of Three-Dimensional Objects*, B. Girod, ed., *Proc. SPIE*, vol. 1260, pp. 240–249, 1990.

[Ost 93] J. Ostermann, "An analysis-synthesis coder based on moving flexible 3-D objects," *Proc. Pict. Cod. Symp.*, Lausanne, Switzerland, Mar. 1993.

[Pea 90] D. Pearson, "Texture mapping in model-based image coding," *Signal Processing: Image Comm.*, vol. 2, pp. 377–395, 1990.

[Pen 82] A. Pentland, "Finding the illuminant direction," *J. Opt. Soc. Am.*, vol. 72, pp. 448–455, April 1982.

[Pen 91] A. Pentland, "Photometric motion," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. PAMI-13, no. 9, pp. 879–890, Sept. 1991.

[Rei 92] M. J. T. Reinders, B. Sankur, and J. C. A. van der Lubbe, "Transformation of a general 3D facial model to an actual scene face," *11th Int. Conf. Patt. Rec.*, pp. 75–79, 1992.

[Ryd 87] M. Rydfalk, "CANDIDE: A parametrised face," Rep. LiTH-ISY-I-0866, Dept. Elec. Eng., Linköping Univ., Sweeden, Oct. 1987.

[Sch 93] H. Schiller and M. Hotter, "Investigations on colour coding in an object-oriented analysis-synthesis coder," *Signal Proc.: Image Comm.*, vol. 5, pp. 319–326, Oct. 1993.

[Ter 90] D. Terzopoulos and K. Waters, "Physically-based facial modeling, analysis and animation," *J. of Visualization and Computer*, vol. 1, pp. 73–80, 1990.

[Tek 95] A. M. Tekalp, Y. Altunbasak, and G. Bozdagi, "Two- versus three-dimensional object-based coding," *Proc. SPIE Visual Comm. and Image Proc.*, Taipei, Taiwan, May 1995.

[Wan 94] Y. Wang and O. Lee, "Active mesh - A feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Proc.*, vol. 3, pp. 610–624, Sept. 1994.

[Wel 90] W. J. Welsh, S. Searby, and J. B. Waite, "Model based image coding," *J. Br. Telecom. Tech.*, vol. 8, no. 3, pp. 94–106, Jul. 1990.

[Wel 91] W. J. Welsh, "Model-based coding of videophone images," *Electronics and Communication Eng. Jour.*, pp. 29–36, Feb. 1991.

[Yau 88] J. F. S. Yau and N. D. Duffy, "A texture mapping approach to 3-D facial image synthesis," *Computer Graphics Forum*, no. 7, pp. 129–134, 1988.

[Zeg 89] K. Zeger and A. Gersho, "Stochastic relaxation algorithm for improved vector quantizer design," *Electronic Letters*, vol. 25, pp. 96–98, July 1989.

[Zhe 91] Q. Zheng and R. Chellappa, "Estimation of illuminant direction, albedo, and shape from shading," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. PAMI-13, no. 7, pp. 680–702, July 1991.

Chapter 25

DIGITAL VIDEO SYSTEMS

Developments in broadcasting, computers, communication technologies such as the emergence of better image compression algorithms, optical-fiber networks, faster computers, dedicated hardware, and digital recording promise a variety of digital video and image communication products in the very near future. Driving the research and development in the field of digital video are the consumer and commercial applications (ordered according to the bitrate requirement), such as

- Digital TV, including HDTV
  @ 20 Mbps over 6 Mhz taboo channels

- Multimedia, desktop video
  @ 1.5 Mbps CD-ROM or hard disk storage

- Videoconferencing
  @ 384 kbps using p $\times$ 64 kbps ISDN channels

- Videophone and mobile image communications
  @ 10-25 kbps

Other applications include surveillance imaging for military or law enforcement, intelligent vehicle highway systems, harbor traffic control, cine medical imaging, aviation and flight control simulation, and motion picture production. In the following, we overview some of these applications in the order in which international compression standards have become/are becoming available for them. In particular, we discuss video communication over ISDN (Integrated Services Digital Network) lines, multimedia in PC and workstation platforms, digital video broadcasting, and low-bitrate applications, in Sections 25.1 through 25.4, respectively.

25.1 Videoconferencing

Videoconferencing using digital techniques has been in existence for some time. Generally speaking, it refers to interactive distance conferencing using ISDN services with clear sound and sharp full-motion video. Each videoconferencing site employs a codec that converts analog video signals to digital and compresses them for transmission. At the receiver, digital signals are converted back to analog for display.

Early systems required special videoconference rooms with high-cost equipment and T1 links that operate at 1.544 Mbps. These systems were quite expensive for everyday use; for example, a typical T1 system would cost $120,000, and the cost of a T1 link would be about $750/hour. Advances in video compression and the adoption of the CCITT H.261 standard helped the emergence of several newer and better videoconferencing systems which cost about $30,000 to $40,000. The newer systems operate over ISDN lines at p $\times$ 64 kbps, ranging from 64 kbps up to 2 Mbps. A typical number for p is 6, putting the bandwidth at 384 kbps. Lower-bandwidth systems that operate at 64 kbps (56 kbps for video and 8 kbps for audio), known as desktop ISDN videophones [Ald 93], are also available. In comparison, the cost of a 128 kbps line is about $35/hour. Some examples of existing videoconferencing equipment are listed in Table 25.1.

Table 25.1: Available videoconferencing products

Vendor	Name	Codec Speed	Max Frame	Comp. Alg.
BT North America	Videocodec VC2200 Videocodec VC2100	56 and 112 kbps 56 kbps to 2048 kbps	30 per sec	H.261
GPT Video Systems	System 261 Twin chan. System 261 Universal	56 and 112 kbps 56 kbps to 2048 kbps	30 per sec	H.261
Compres. Labs.	Rembrandt II/VP	56 kbps to 2048 kbps	30 per sec	H.261, CTX CTX Plus
NEC America	VisualLink 5000 M20 VisualLink 5000 M15	56 kbps to 384 kbps 56 kbps to 2048 kbps	30 per sec	H.261, NEC proprietary
PictureTel Corp.	System 4000	56 kbps to 768 kbps	10 per sec mono	H.261, SG3 SG2/HVQ
Video Telecom	CS350	56 kbps to 768 kbps	15 per sec	H.261, Blue Chip

Besides videoconferencing and desktop videophone, video communication using an ISDN basic rate interface (BRI), operating at a total of 128 kbps, may be used for such applications as distance learning and access to multimedia information services. In distance learning, an instructor teaches students who are at remote locations. The students can interact with the instructor, by direct talking or sharing a whiteboard [Ald 93]. Multimedia information services include image-based electronic library systems and shopping-catalogs that can be browsed from home.

25.2 Interactive Video and Multimedia

Multimedia can mean different things to different people. Here it refers to the ability to provide full-motion interactive digital video in the personal computer (PC) or desktop workstation environment. Multimedia also involves text, graphics, sound, and still-images, which have long existed in the PC or workstation environment. The main components of a multimedia system are:

- data capture tools, such as video recorders and digitizers,
- data editors and authoring tools, for animation, audio and video editing, user-interface development, etc., and
- database storage and retrieval tools, for searching large databases, archiving, and backup.

The difficulty with full-motion video has been in the large data rates required. Multimedia workstations use CD-ROMs and hard disks for video storage at about 1.5 Mbps, which require approximately 30:1 compression of standard TV resolution images. The latest developments in image compression make this possible in real time. Two of the earliest initiatives in this area have been Intel's digital video interactive (DVI) technology, and compact disc-interactive (CD-I), jointly funded by NV Philips and Sony.

DVI technology is a general-purpose hardware for providing full-motion video in the PC environment, and uses CD-ROM for the storage medium. The design goal is to provide multimedia functionality at a cost suitable for desktop computers. It is based on two custom VLSI chips: the 82750PB pixel processor and the 82750DB display processor. The pixel processor mainly performs compression and decompression of images along with functions like YUV-RGB conversion, bilinear interpolation, and so on. It is able to decompress 640×480 JPEG encoded images in less than one second. The display processor retrieves decompressed images from memory, converts them to the format needed for display, and produces timing and control signals to drive various displays. Popular formats such as NTSC, PAL, VGA, and SVGA are supported. This chip set is compatible with 16 or 32 bit microprocessors operating at 6 MHz or higher clock speed, and utilizes 16 Mbyte video RAM (VRAM). The chip-set is programmable; that is, the compression algorithms can be modified according to the application.

The Compact Disc-Interactive (CD-I) system is a CD-based interactive audio/video special-purpose computer. It is based on Motorola's 680X0 processor, and the CD-RTOS, an operating system developed for the CD-I environment. Its original design was capable of providing full-motion video only on part of the screen. However, full-screen, full-motion video has become available with an add-in module [Mee 92]. This module reproduces video from an encoded bitstream with a frame rate of 24 Hz, 25 Hz, or 30 Hz. The frame rate converter transforms this video to 50 Hz or 60 Hz. Finally, the YUV representation of the encoded bitstream is converted into RGB for display. It includes a number of multimedia capabilities, such as a CD file manager for audio, a user communication manager of the video system, and a motion picture file manager. CD-I players currently exist that can be connected to standard TV sets to play back movies that are recorded on CDs. A full-length feature movie usually requires two CDs.

Recently, almost every computer and workstation manufacturer, including Apple, IBM, SUN, and Silicon Graphics, have added full-motion video capabilities. However, no industry-wide standards have been established yet. Recall that the MPEG-1 video compression standard addresses video compression for multimedia stations, but not full compatibility between various multimedia products. As computers and communication equipment move closer towards "grand unification," the line between multimedia systems and videoconferencing and videophone products is getting more and more blurred; that is, most multimedia systems can now be interfaced with LAN (Local Area Networks), WAN (Wide Area Networks), ISDN, or ATM (Asynchronous Transfer Mode) networks for interactive desktop videoconferencing. At present several companies are introducing plug-in cards and software for videoconferencing over LAN, WAN, and ATM networks. A nice feature of these products is that they provide multiplatform support, so that people using different workstations and personal computers may still share a whiteboard, text tools, and full-motion video.

With the newer generation of higher-speed CD-ROMs and faster processors entering the market every day, multimedia remains an active and exciting field. An emerging technology in interactive multimedia is effective human/machine interaction. Stereo vision, talker verification, speech synthesis, tactile interaction, and integration of multiple sensor modalities are active research areas to expand the present capabilities for human/machine interaction.

25.3 Digital Television

TV is arguably the most commonly used image communication system in the world today. However, present TV transmission standards are based on technology that is more than 40 years old. As a result, there has been widespread interest in the consumer electronics industry to develop more advanced TV systems that benefit from recent technological advances. The major advanced TV research programs in the world are tabulated in Table 25.2.

Table 25.2: Major programs for HDTV research.

Japan	NHK
Europe	EUREKA 95
U.S.A.	Grand Alliance (AT&T, General Instrument, Mass. Inst. Tech., Philips N. A. David Sarnoff Res. Cen., Thomson Cons. El., Zenith)

Development efforts pioneered by the Japanese in late the 70s and early 80s resulted in hybrid advanced TV systems, with digital processing at the transmitter and receiver but using analog transmission. Later, in the early 90s, studies in the U.S. proved the feasibility of the all-digital TV approach, which was believed to be unrealistic not too long ago. All-digital TV will not only offer better image quality, easy conversion between multiple standards, and more channels within the same bandwidth (thanks to advances in digital data compression technology), but more important, it will unite computers with TV sets and telecommunication services with cable TV services in a revolutionary fashion. In the following, we provide an overview of the developments around the world that led to the advent of all-digital TV.

25.3.1 Digital Studio Standards

Although the present TV transmission standards are analog (see Chapter 1), digital TV signals find routine use in TV studios for image processing and digital storage. Digital techniques are commonly used for such tasks as program editing, generating special effects, and standards conversion. Digital storage is preferred at the production level, because consumer-quality video storage devices, such as VHS recorders, introduce degradations that are objectionable in the production studio environment.

In the following, we first summarize the existing analog TV standards. The corresponding CCIR 601 standards for digitization of the respective signals (also known as digital studio standards) are listed in Table 25.3.

- **NTSC** (National Television Systems Committee) - Accepted for B&W in 1941, and extended to color in 1954
  Used in the USA, Canada, Japan, and Latin America
  2:1 interlace, 4:3 aspect ratio
  525 lines/frame, 29.97 frames/sec (262.5 lines/field, 59.94 fields/sec)
  Perceptually 340 lines/frame, 420 resolvable pels/line (Kell factor)
  Analog transmission over 6 MHz channel
  There are 68 channels assigned in the US: 54-88 MHz (ch 2 to 6), 174-216 MHz (ch 7 to 13), 470-806 MHz (ch14 to 69). However, less than 20 channels are used in a locality to prevent interference.

Table 25.3: CCIR 601 standards.

Format	Sampling Frequency (Y/U,V)	Field rate	Interlace/ Aspect	Active lines/frame	Active pixels/frame
NTSC	13.5/6.75	60	2:1/4:3	488	720
PAL/ SECAM	13.5/6.75	50	2:1/4:3	576	720

- **PAL** (Phase Alternation Line) - Accepted in 1967 for color broadcast
  Used in most of Europe, and Australia
  625 lines/frame, 2:1 interlace, 50 fields/sec, 4:3 aspect ratio
  Analog transmission over 8 MHz channel.
- **SECAM** (Systeme Electronique Color Avec Memoire) -.Accepted in 1967 for color broadcast, used in France, Russia, and Eastern Europe
  625 lines/frame, 2:1 interlace, 50 fields/sec, 4:3 aspect ratio
  Analog transmission over 8 MHz channel.

Note that these standards are incompatible with each other, and conversion from one to another requires digital techniques.

25.3.2 Hybrid Advanced TV Systems

There were several proposals, before the advent of all-digital TV, that offered various degrees of improvements on the quality of present TV systems. We refer to them as "hybrid advanced TV (ATV) systems." ATV systems generally feature better spatial and/or temporal resolution, wider screens, improved color rendition, and CD-quality stereo sound. They fall under two broad categories: compatible ATV systems and incompatible ATV systems.

1) Compatible system proposals may be summarized as follows:

- IDTV (improved definition TV) systems which use digital processing at the receiver for picture quality improvement using existing analog TV transmission standards. Picture quality may be improved through: i) spatial and temporal resolution enhancement by nonlinear interpolation to recover super-Nyquist frequencies, including motion-compensated deinterlacing and frame rate conversion, ii) luminance-chrominance separation to eliminate cross-luminance and cross-chrominance artifacts due to imperfect Y/C separation in conventional NTSC demodulators, and iii) ghost elimination to reduce time-delayed, attenuated, and distorted versions of the TV signal.
- EQTV (extended-quality TV) systems which require transmission of an augmentation signal. They include systems that feature 16:9 aspect ratio by

means of transmitting an augmentation signal at another frequency band. Conventional receivers that do not receive this augmentation signal can display ordinary-quality TV signals.

Early efforts in Europe have been in the direction of EQTV system development. PAL-plus and D2-MAC are two examples of EQTV systems which require transmission of augmentation signals. They have 16:9 aspect ratio, and are compatible with PAL and MAC (Multiplexed Analog Components), respectively. Note that MAC is a 625/50/2:1 analog satellite transmission standard developed by the British around 1970.

2) Incompatible systems are generally known as HDTV (high-definition TV). CCIR 801, adopted in 1990, defines HDTV as follows: "A high definition TV system is a system designed to allow viewing at about three times picture height such that the transmission system is virtually or nearly transparent to the level of detail that would have been perceived in the original scene by a viewer with average visual acuity." HDTV systems feature a video signal that has about twice the current resolution in both the horizontal and vertical directions, a wider aspect ratio of 16:9, separate luminance and chrominance signals, and CD-quality sound. Initial HDTV proposals, developed first in Japan and later in Europe, were hybrid (mixed analog/digital) systems, which use digital signal/image processing at the transmitter and receiver, but transmission was by analog means. We elaborate on these systems below.

ATV in Japan

Studies towards an advanced TV system were started in the 1970s in Japan at the NHK Laboratories. NHK applied to CCIR, a world standards organization, in 1974 for the standardization of an analog HDTV format, called the MUSE (MUltiple sub-Nyquist Sampling Encoding). CCIR Study Group 11 worked from 1974 to 1986 to achieve a single world standard for production and international exchange of HDTV signals with no apparent success.

The MUSE system is based on a motion adaptive subsampling strategy to reduce the transmission bandwidth requirement by a factor of 3:1. The HDTV signal is defined by the following parameters:

1125 lines/frame, 60 fields/sec, 2:1 interlace, 16:9 aspect
8.1 MHz DBS transmission
Sampling rates: Y at 48.6 MHz and C at 16.2 MHz

It is primarily intended for broadcasting over 24 MHz direct broadcast satellite (DBS) channels, and is not compatible with other transmission standards. The basic idea of the MUSE system is motion-adaptive subsampling. That is, if motion is detected at a certain region, spatial lowpass filtering is applied before subsampling. On the other hand, in still-image areas, subsampling is applied without any lowpass filtering. There is no motion estimation or motion compensation involved in the MUSE system. Major electronic manufacturers in Japan have participated in the project and builded mixed analog/digital HDTV receivers. First broadcast using the MUSE system was realized on Nov. 25, 1991. At present, there are daily HDTV broadcasts via DBS in Japan.

ATV in Europe

European efforts for HDTV research have been organized under the EUREKA-95 project, which resulted in HD-MAC, an analog HDTV standard for DBS transmission developed around 1988. While the Japanese advocated an incompatible HDTV approach, HD-MAC has emerged as a MAC-compatible standard with 1250 lines/frame, 50 fields/sec, 2:1 interlace, and 16:9 aspect ratio.

HD-MAC achieves reduction of bandwidth from 54 MHz to 10.125 MHz by advanced motion-compensated subsampling. Motion information is transmitted digitally to assist the decoding process. The first HD-MAC broadcast was made from the 1992 Winter Olympics in France to selected test centers in Europe via DBS. The HD-MAC system has already been abondoned in Europe in favor of all-digital systems currently under development in the U.S.A.

25.3.3 All-Digital TV

All-digital TV refers to digital representation and processing of the signal as well as its digital transmission. The nature of the digital broadcast removes the synchronicity and real-time requirements of the analog TV, and offers many different options. A digital TV standard will unify the computer/workstation and TV industries in the future; hence, the introduction of the term "telecomputer." Although a standard for all-digital HDTV has not yet been formally approved, there is steady progress toward the adoption of a standard by the FCC before the end of 1995, and there are already some companies offering digital TV services conforming with the present standards using DBS broadcasting. Digital TV broadcast media include:

- Terrestial broadcast
- Direct broadcast satellite
- Cable and broadband ISDN distribution

In the U.S., the Federal Communications Commission (FCC) has ruled that the existing 6 MHz taboo-channels will be used for terrestrial broadcast. For digital terrestrial transmission, a 6 MHz channel can support about 20 Mbps data rate with sophisticated digital vestigial sideband modulation. Considering the parameters of a typical HDTV system, we still need about 545:20 = 28:1 compression to achieve this bitrate. Direct broadcast satellite (DBS) transmission is widely used in Japan and Europe as an analog transmission medium. In the U.S., some companies have already started providing digital TV transmission (at the NTSC resolution) using DBS. Cable distribution systems are heavily employed in the U.S. at present. Each cable channel is allotted 6 MHz, and typically 30 to 50 channels are available. All adjacent channels can be used. Some cable networks offer limited two-way communication capability where upstream data transmission is allowed. With digital transmission and effective compression, cable companies may offer approximately 150 channels using the presently available bandwidth. The broadband ISDN (B-ISDN) offers a unified network capable of providing voice, data, video, LAN, and MAN services [Spe 91]. The H4 access provides approximately 135 Mbps. Asynchronous transfer mode (ATM) is being considered for standardization as the multiplexing and switching vehicle of various services. DBS, cable, and B-ISDN services offer the possibility of more advanced video services, such as video-on-demand [Spe 95]. In the following, we summarize the all-digital HDTV and TV efforts in the U.S.A.

U.S. Grand Alliance

In the U.S.A. efforts to establish a terrestrial HDTV broadcast standard were initiated by the FCC in 1987. At the time, it was generally believed that more than 6 MHz was required to broadcast analog HDTV, and nobody thought that a digital HDTV broadcast would fit within a 6 MHz channel until 1990. In 1993, after only 6 years, it was decided that the HDTV standard in the U.S. will be an all-digital simulcast system. Simulcasting requires every existing TV broadcaster to have a second 6 Mz channel for digital HDTV broadcast. The NTSC source with 4.2 MHz bandwidth will continue to be transmitted through the usual existing 6 MHz channels until the year 2008.

When the tests began at the Advanced TV Test Center (ATTC) in Alexandria, Virginia in 1991, there were six proposals: one NTSC-compatible system, one analog simulcast system, and four digital systems. Early in 1993, all but the four digital system proposals were eliminated. The remaining proposals were:

1) *American TV Alliance* - General Instruments and MIT
System: DigiCipher
1050 lines/frame, 29.97 frames/sec, 2:1 interlace
Horizontal scan rate: 31.469 kHz
Sampling frequency: 53.65 MHz
Active pixels: 1408×960 luma, 352×480 chroma
Video Comp.: MC 32×16, Field/Frame DCT
RF Modulation: 32/16 QAM

2) *Zenith and AT&T*
System: Digital Spectrum Compatible (DSC-HDTV)
787.5 lines/frame, 59.94 frames/sec, progressive (1:1)
Horizontal scan rate: 47.203 kHz
Sampling frequency: 75.3 MHz
Active pixels: 1280×720 luma, 640×360 (chroma)
Video Comp.: MC 32×16, 8×8 leaky pred., DCT, VQ
RF Modulation: 2/4 level VSB

3) *Advanced TV Research Consortium* - Thomson Consumer Electronics, Philips Consumer Electronics, NBC, David Sarnoff Res. Center, and Compression Labs.

System: AD-HDTV
1050 lines/frame, 29.97 frames/sec, 2:1 interlace
Horizontal scan rate: 31.469 kHz
Sampling frequency: 54 MHz
Active pixels: 1440 × 960 luma, 720 × 480 chroma
Video comp.: MPEG++ (MC-DCT)
RF modulation: 32/16 SS-QAM

4) *American TV Alliance* - MIT and General Instruments

System: CC-Digicipher
787.5 lines/frame, 59.94 frames/sec, progressive (1:1)
Horizontal scan rate: 47.203 kHz
Sampling frequency: 75.3 MHz
Active pixels: 1280 × 720 luma, 640 × 360 chroma
Video comp.: MC 16 × 16, 8 × 8 DCT
RF modulation: 32/16 QAM

The following excerpt from Newslog, *IEEE Spectrum*, April 1993, summarizes the recommendation of the FCC special panel. "On February 11, 1993, a special FCC panel said there were flaws in all five of the systems. It is recommended that the FCC's *Advisory Committee on Advanced Television* hold a new round of testing after the four groups with all-digital systems fixed their problems. Officials from the four groups said they had begun talking about merging their systems into one - an idea that the FCC has been promoting." In May 1993, the four groups agreed to form a "Grand Alliance" to merge their systems into a single system incorporating the best features of each.

In October 1993, the Grand Alliance announced i) that the video compression algorithm will be MPEG-2, main profile, high level, ii) that the MPEG-2 transport mechanism will be used, iii) that the Dolby AC-3 audio system will be used, and iv) that three modulation techniques, 4-level VSB, 6-level VSB, and 32 QAM (quadrature amplitude modulation), will be tested to complete the specification. In order to facilitate interoperability of broadcasting, computer multimedia, computer graphics, industrial imaging, and the National Information Infrastructure multiple scanning formats have been adopted which resembles the open-architecture TV concept [Bov 91]. In an open-architecture video representation, the number of lines in the display depends on the display hardware, and is not coupled to the number of lines employed by the production equipment. The scanning formats supported by the Grand Alliance proposal include progressive and interlaced scanning, at two spatial resolutions 720 lines × 1280 pixels and 1080 lines × 1920 pixels. In addition to 60 Hz and 30 Hz frame rates, a 24 Hz film mode is included with both progressive and interlaced scanning for motion-picture source material. The Grand Alliance scanning formats are summarized in Table 25.4. All formats support 16:9 aspect ratio with square pixels.

Table 25.4: Grand Alliance HDTV formats.

Spatial resolution	Line structure	Frame rate
720 × 1280	Progressive	60, 30, 24
1080 × 1920	Interlaced	30
1080 × 1920	Progressive	30, 24

Because there exist multiple scanning formats, transconversion may be required at the transmitter and/or at the receiver. A transconverter at the decoder output performs the necessary format conversion if the display scanning parameters differ from those of the received signal. The decoder accepts both 1080-line interlaced and 720-line progressive formats, and supports bidirectional prediction with motion estimation parameters up to ±127 horizontal and ±31 vertical, fully compliant with MPEG-2 requirements. Field tests of the proposed all-digital HDTV standard is currently underway.

Digital DBS TV

Digital transmission of TV signals at today's TV resolution is already becoming available through DBS services using 18 in antennas and digital decoders. These services use MPEG-2 compression starting with RGB source material to provide images sharper than NTSC pictures. Hughes Communications and the United States Satellite Broadcasting Company have just announced digital DBS services, called DIRECTV^{TM} and USSB^{TM}, respectively. At the moment, these services are available only in selected test markets. However, they are expected to become available nationally soon.

Interactive Networks and Video-on-Demand

The available cable TV networks today offer one-way traffic with a fixed set of channels. New high-bandwidth network architectures and protocols, such as fiber optic networks with ATM switching, are needed to provide users with a variety of interactive services. Video-server computers will be integral components of these interactive services to offer customized programming and video-on-demand [Lin 95] [Spe 95]. With the adoption of digital signal formats and transmission standards, video storage is also expected to be dominated by digital technologies, such as CD-ROMs and VTRS [Eto 92].

25.4 Low-Bitrate Video and Videophone

Low-bitrate video (LBV) generally refers to applications that require less than 64 kbps. There are many applications of low-bitrate coding, including:

- *Videophone:* Videophone service on PSTN (Public Switched Telephone Network), mobile, and LANs. Real-time encoder and decoder with easy implementation.
- *Mobile multimedia communication* such as cellular videophones and other personal communication systems.
- *Remote sensing:* One way communication of audio-visual information from a remote location, for example, surveillance, security, intelligent vehicle highway systems (IVHS), harbor traffic management.
- *Electronic newspaper:* Multimedia news service on PSTN, radio channels, and Smart cards.
- *Multimedia videotex:* Videotex is currently a multimedia database environment but lacks capability for full-motion video.
- *Multimedia electronic mail*

These applications require huge compression factors which generally cannot be met satisfactorily with the existing compression standards. Typical compression ratios to reach 10 kbps starting from several video formats are shown in Table 25.5.

Table 25.5: Compression requirements to reach 10 kbps

Frames/s	CCIR 601 (720x576)	CIF (352x288)	QCIF (176x144)
7.5	4979:1	915:1	229:1
10	6637:1	1216:1	304:1
15	9952:1	1824:1	456:1
30	19904:1	3648:1	912:1

Videophone is at the low end of the videoconferencing market, where low cost and operation over the existing subscriber network become extremely important. A basic setup requires at least one desktop monitor to view the video signals, a camera for capturing the video signal, and an audio connection. Due to high cost and long timeframe associated with wide range deployment of a fiber/coaxial-based local subscriber network, the target bitrate for videophone products has been set below 28.8 kbps (using the V.34 modems over the existing PSTN). Although, there are no established video compression standards at such low bitrates, a few representative products have already appeared in the market [Ear 93]. These systems use variations of the MC-DCT coding scheme, which are similar to the H.261 standard;

Table 25.6: Available videophone products

Product	Data Rate	Compression Algorithm
AT&T Videophone 2500	16.8/ 19.2 kbps	MC DCT 10 frames/s (max)
British Telecom/Marconi Relate 2000 Videophone	9.6/ 14.4 kbps	H.261 like 7.5 (3.75) frames/s
COMTECH Labs. STU-3 Secure Videophone	9.6 kbps	MC DCT QCIF resolution
Sharevision	14.4 kbps	MC DCT

that is, using the macroblock concept, DCT, motion estimation/compensation, and run-length and VLC coding. Some examples of early videophones are listed in Table 25.6.

Recognizing the increasing demand for low bitrate applications, especially mobile video communications, efforts for standardization in low bitrate coding have been initiated in 1993 by the ISO/MPEG-4 Ad-Hoc Group and ITU-T/LBC (Expert Group on Low Bitrate Coding). Because of the urgent need to provide a common platform of communication between various products by different vendors (using the available technology) and the need for fundamentally different technologies to provide improved performance and embedded functionality, the work has been divided into two phases: a near-term solution and a far-term solution. The near-term solution has very recently resulted in the ITU Draft Recommendation H.263. The far-term solution refers to a fundamentally new standard expected to be completed by November 1988. This task will be handled by ISO/MPEG-4, with liaison to the ITU.

25.4.1 The ITU Recommendation H.263

The ITU/LBC group has drafted the near-term H.324 specifications, that include audio/video coding, multiplexing, error control, and overall system integration targeted at videophone applications on PSTN and mobile channels. The ITU Draft Recommendation H.263 (frozen in March 1995) specifies the video coding algorithm, which is an "H.261" like (MC-DCT) algorithm, at about 22 kbps (out of 28.8 kbps overall).

The major differences between the H.263 and H.261 standards are:

- Motion estimation with one-half-pixel accuracy, which eliminates the need for loop filtering.
- Overlapped motion compensation to obtain a denser motion field at the expense of more computation.

- Adaptive switching between motion estimation at macroblock (16 × 16) and block (8 × 8) levels.
- Support for sub-QCIF bitstreams.

It has been claimed that the Test Model 5 (TMN5) provides 3-4 dB higher PSNR than the H.261 algorithm at below 64 kbps. It can be used as a milestone to assess the performance of future low-bitrate coding algorithms and standards.

25.4.2 The ISO MPEG-4 Requirements

The MPEG-4 Ad-hoc group was organized in September 1993 with the mission of developing a fundamentally new generic video coding standard at rates below 64 kbps. In November 1994, this mission has been modified as "to provide an audio-visual coding standard allowing for interactivity, high compression, and/or universal accessibility, with high degree of flexibility and extensibility" [Zha 95].

The new MPEG4 vision includes eight functionalities that are not supported by the existing standards. They are:

- Content-based manipulation and bitstream editing
- Content-based multimedia data access tools
- Content-based scalability
- Coding of multiple concurrent data streams
- Hybrid natural and synthetic data coding
- Improved coding efficiency
- Improved temporal access at very low bitrates
- Robustness in error-prone environments

MPEG-4 intends to cover a wide range of applications, including "virtual" conference and classroom; interactive mobile videophone; content-based multimedia database query, searching, indexing, and retrieval; interactive home shopping; wireless monitoring; and so on. At present MPEG-4 structure consists of four elements: syntax, tools, algorithms, and profiles. The syntax is an extensible language that allows selection, description, and downloading of tools, algorithms, and profiles. A tool is a specific method. An algorithm is a collection of tools that implement one or more functionalities. A profile is one or more algorithms to cover a specific class of applications. Interested parties can submit proposals for potential tools, algorithms, and profiles. Deadline for submissions of proposals is October 1995.

ELECTRICAL ENGINEERING
Signal Processing

digital video processing

A. MURAT TEKALP

The rapid development of products and services offering full-motion digital video suggests that, in the coming decade, digital video will greatly impact the computer, telecommunications, and imaging industries. To help readers ride the wave of the future, this timely volume provides, for the first time, comprehensive coverage of the principles of digital video processing, including leading algorithms for various applications, in a highly accessible tutorial style.

Digital Video Processing is organized into six comprehensive sections covering:

- Representation of digital video including modeling of video image formation and spatio-temporal sampling.
- Two-dimensional motion estimation.
- Three-dimensional motion estimation and segmentation.
- Video filtering.
- Still-image compression.
- Video compression.

The author carefully develops detailed treatment of the mathematical principles behind representation of digital video as a form of computer data and processing of this data for 2-D and 3-D motion estimation, digital video standards conversion, frame-rate conversion, de-interlacing, noise filtering, resolution enhancement, and motion-based segmentation. In addition, Tekalp covers the fundamentals of image and video compression, and the emerging world standards for various image and video communication applications, including high-definition TV, multimedia, video-conferencing, videophone, and mobile image communications.

PRENTICE HALL
Upper Saddle River, NJ 07458

ISBN 0-13-190075-7
90000>
9 780131 900752