# Library-based Image Coding using Vector Quantization of the Prediction Space

by

Nuno Miguel Borges de Pinho Cruz de Vasconcelos

Licenciatura, Electrical and Computer Engineering
Faculdade de Engenharia da Universidade do Porto, Portugal
(1988)

SUBMITTED TO THE PROGRAM IN
MEDIA ARTS AND SCIENCES
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1993



Author: _____
Program in Media Arts and Sciences
August 6, 1993

Certified by: _____
Andrew B. Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted by: _____
Stephen A. Benton
Chairperson, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

# Library-based Image Coding using Vector Quantization of the Prediction Space

by

Nuno Miguel Borges de Pinho Cruz de Vasconcelos

Submitted to the Program in Media Arts and Sciences,
on August 6, 1993 in partial fulfillment of the
requirements for the degree of
Master of Science

## Abstract

Traditional image compression algorithms based on interframe coding rely on a block-matching algorithm to estimate motion between frames and minimize temporal correlation. The simplicity of the coding model behind block-matching, which assumes a world uniquely composed of 2-D planar objects subject to translational motion, leads to clearly sub-optimal performance when more complex events (such as newly revealed objects, or objects subject to non-translational motion), very common in real world scenes, occur. In addition, its complexity limits, in practical applications, the number of frames used for prediction (prediction space) to one or at most two, further limiting the coding efficiency.

This thesis presents a library-based encoder, which extends the block-based coding model into an object-oriented direction that leads to higher efficiency in the handling of those complex events; and which is capable of exploiting a significantly larger prediction space without significant increase in complexity.

Under this new approach, during the encoding process, the coder continuously builds a library containing information relative to all or part of the previously encoded images, and the prediction for each new input block is obtained from a search of the closest match in this library. Vector quantization is used to obtain the library that optimally represents the prediction space, by clustering the blocks of the prediction space which belong to the same or similar objects and assigning a representative block to each of the clusters.

Thesis Supervisor: Andrew B. Lippman
Associate Director, MIT Media Laboratory

The work reported herein is supported by a contract from the Movies of the Future consortium, including Apple Computer Inc., Bellcore; Eastman Kodak Company; Intel Corporation; Viacom International Inc., and Warner Brothers Inc.

**Library-based Image Coding using**
**Vector Quantization of the Prediction Space**

by

Nuno Miguel Borges de Pinho Cruz de Vasconcelos

Reader: _____

Edward H. Adelson
Associate Professor of Vision Science
MIT Media Laboratory

Reader: _____

Arun N. Netravali
Executive Director, Research, Communications Sciences Division
AT&T Bell Laboratories

*To my parents,*

*Assunção and Manuel.*

# Contents

# Chapter 1

# Introduction

A typical sequence of images presents a high degree of correlation between consecutive frames and between adjacent regions within each frame. By exploiting that correlation, an image coder can achieve significant savings of transmission bandwidth at the cost of none or small distortion in image quality.

Typical image compression algorithms are based on motion-compensated interframe difference coding. However, and due to the vast amount of research spent on this type of algorithms, the gains to be made from straightforward waveform image coding are diminishing as they are reaching an asymptote in efficiency. The best way to make improvements is to employ a better model of the image or the image formation process, an idea that is becoming known as "object-oriented" or "object-based" coding. Such a model is, however, very hard to compute for most of the natural scenes, and this fact restricts the use of model-based or object-oriented encoders to a very limited range of applications, such as video-telephone or video-conference.

This thesis extends the traditional image coding model used (or proposed) for television transmission towards an object model, by combining conventional image coding techniques with an object-based approach. In this way, it is possible to obtain higher performance while retaining the robustness of the conventional coding tech-

niques, which is very important for applications where high-coding quality of general scenes (as opposed to restricted images sets such as "head-and-shoulders" conference) is required.

## 1.1 Motivation

In the general sense, an interframe coder works by splitting each input frame into regions (usually square blocks), finding the best match between each of those regions and a region of predefined size in a previously transmitted frame (block-matching), and transmitting a coded version of the error between the original region and the best match. This coding operation is what varies most from implementation to implementation. In general, a decorrelating transformation in the frequency domain (DCT, subbands, etc.) is used, followed by quantization (scalar or vectorial), and variable-length coding (Huffman coding, arithmetic coding, etc.).

The MPEG algorithm [18] is probably the best example of state of the art in interframe coding. Its main characteristics are:

- input image is split into fixed size blocks;
- motion estimation/compensation based on a previous (P-pictures) or on a previous and a future (B-pictures) frame;
- bypassing of the prediction when the closest match is not good enough (intra blocks);
- use of the DCT as decorrelating transform, scalar quantization, and Huffman coding.

The quality of the prediction has a crucial importance in the performance of an interframe coder since most of the redundancy of the input signal is eliminated when the prediction is subtracted. This can be verified by the bit allocation necessary to

maintain approximately constant quality among different picture types in an MPEG coder, which is typically in the ratio of 3:2:1 for I-pictures (all blocks intra), P-pictures, and B-pictures, respectively.

### 1.1.1 Problem

Since the prediction quality depends exclusively on the quality of the motion estimation, it is obvious that this is a determinant factor in the quality of the coding algorithm. A good motion estimator is, however, difficult to implement since accurate motion estimation would imply some kind of knowledge by the encoder of the 3-D objects present in the scene. Although some research has been presented in this field in the last few years, model-based coding is still in its infancy and will not be a viable answer to the problem of high-quality video coding in the years to come. As a consequence, block-matching is widely accepted as the more efficient motion-estimation technique.

The main problem with block-matching motion estimation is, however, that it assumes a very simple model of the world, purely composed of 2-D planar objects subject to 2-D planar shifts, and cannot account for more complex events, such as non-translational motion or newly revealed data, that are very common in the real world. Also, and although conceptually simple, block-matching is computationally intensive, and this limits, in practice, the search region to one or at most two frames (as in MPEG), which is insufficient for several types of motion that occur in real life scenes. Better prediction quality can be achieved by extending the prediction space (space where the search for the best match is performed) to a higher number of frames.

This thesis presents a new algorithm to achieve the goals of extending the coding model and exploiting a larger prediction space without a significant increase in the complexity of the coding process.

### 1.1.2 Approach

The main idea is to improve the traditional motion estimation/compensation structure by incorporating library-based processing in the traditional interframe encoder.

A library is a collection of representative blocks that the encoder builds during the encoding process. Library-based encoding can be viewed as an extension of block-matching motion-compensated encoding where, for each block, a search is done on a library limited to a fixed region of a previous frame. Since adjacent regions of the same frame are in general highly correlated, a smaller set of blocks of the previous frame than this fixed region would, in general, be sufficient to obtain equivalent motion-estimation performance, but with reduced computation. Similarly, maintaining the number of entries in the library, blocks from other frames could be incorporated, originating a better prediction and higher coding quality, while maintaining the search complexity.

Library-based encoding can also be seen as form of object-oriented image coding since each library entry is representative of a set of blocks with similar characteristics and, therefore, associated with the same or similar objects. In this way, the coding model becomes capable of handling complex events, such as objects being revealed after temporary occlusion, objects subject to cyclic motion, or objects subject to non-translational motion.

In addition to the potential for improved coding efficiency, a library-based encoder has the potential to use this improvement to provide new access techniques, and to exploit the coding as part of the analysis one would wish to do to summarize or search through a movie. For example, by downloading the library alone, it is possible to recreate a low-quality version of the movie that allows the user to browse through it with minimal decoding complexity and transmission rate. In this context, the library ultimately becomes a sort of "shorthand" or keymap for the movie itself, and the coding analysis is extended to perform user level tasks.

The fundamental questions that need to be answered in the implementation of a library-based coder are how to design the library, how to update it, and how to search it.

A Vector Quantizer (VQ) provides an efficient answer for the first question. A VQ divides its input space (in this case the prediction space) into a set of clusters, by grouping together vectors (blocks) with similar characteristics, and assigning a representative vector (block) to each of these clusters in order to minimize overall distortion. This representative vector is the centroid of the cluster which, under the mean-squared error distortion measure, reduces to the vector average of all the vectors in the cluster. In simpler words, a VQ provides a codebook (vector library) that is an optimal representation of the prediction space.

The major problem of a VQ-based solution resides in the complexity of codebook design. As a good probabilistic description of the input is, in general, not available, the codebook is usually designed by applying a computationally heavy iterative algorithm to a training sequence. Since, for this particular application, an adaptive codebook matched to the local characteristics of the input sequence would be preferable to a unique global codebook, the design complexity becomes a very important factor.

To minimize this design complexity, a frame adaptive codebook, replenished at the scene level, is proposed in this thesis. With this scheme, the design complexity is distributed between several frames and the codebook gradually improved, while maintaining the implementation complexity between reasonable bounds.

An input block is library-encoded by searching the closest match in the library, and coding the difference between this prediction and the block itself. If a close match cannot be found, the input block is coded, transmitted, and at the same time incorporated on the library. In this way, only blocks that convey new information are added to the library, which can therefore maintain a reasonable size while being representative of a set of various frames.

To avoid an endless growth of the library, some of the blocks contained in it must be dropped when new ones are incorporated. A library update scheme, based on conditional replenishment techniques, is also proposed to reduce the overhead associated with the transmission of the library refinements.

Whenever the library fails to perform well, the coder switches to traditional motion-compensated prediction based on the previous frame, assuring that the overall performance is never worst than the achieved with this type of coding.

The thesis is organized as follows. Chapter 2 presents a review of the fundamental principles of the traditional image compression techniques relevant to the work developed in the thesis. Some of the recent advances in the field, in particular model-based and object-oriented approaches, are discussed in chapter 3. Chapter 4 presents the new library-based algorithm that combines aspects from both the traditional and the object-oriented approaches to achieve increased coding efficiency. The performance of this library-based scheme is analyzed in chapter 5. Finally, chapter 6 presents the conclusions and suggestions for future work.

# Chapter 2

# Image compression fundamentals

The objective of an image compression system is to find a representation of the input data which requires the minimum possible number of bits to achieve a desired coding fidelity. This chapter reviews of some of the most popular traditional compression techniques, based on concepts from the fields of digital signal processing and information theory. Obviously, this review is not intended to be exhaustive, but to establish the background necessary for the understanding of the work later presented. More detailed studies about the techniques here presented and some others can be found in the references.

Since efficient quantization plays a crucial role in the efficiency of a (lossy) compression system, this topic is covered in some detail in the following sections. Section 2.1 presents the principles of scalar quantization, and some theoretical results about the expected performance of algorithms based on it. Section 2.2 discusses transform coding, a pre-processing technique that leads to a representation of the input which increases the efficiency of scalar quantization. Section 2.3 describes the extension of the concept of quantization to multi-dimensional spaces provided by a technique known as vector quantization. Section 2.4.1 deals with predictive methods commonly used for the temporal processing of image sequences. Finally, section 2.5 presents lossless techniques which can be used in conjunction with any of those dis-

cussed in the previous sections.

## 2.1 Quantization

A *quantizer* performs the mapping of a large (infinite for analog signals) set of input amplitude values into a smaller, finite set of reconstruction (or quantized) values. It can be considered the fundamental stage of any image coding system because it is the element which provides the most significant degree of compression of the input signal.

Quantizers can be divided in two major groups: *scalar quantizers*, which process a sample at a time, and *vector quantizers*, which process a group (vector) of samples simultaneously.

### 2.1.1 Scalar Quantization

A scalar quantizer maps the amplitude of the input sample into a finite set of *reconstruction levels*. The range of input amplitudes is partitioned into a set of $N$ *regions* ($\mathcal{R}_i$) and a reconstruction value ($y_i$) associated with each region. The regions $\mathcal{R}_i$ are delimited by *decision levels* or *thresholds* ($d_i$), and all the input amplitudes in region $\mathcal{R}_i$ ($x \in [d_{i-1}, d_i[$) are mapped into the reconstruction value $y_i$.

Mathematically,

$$\mathcal{R}_i \subset \{x : d_{i-1} \leq x < d_i\}, \tag{2.1}$$

and

$$\mathcal{Q}(x) = y_i, \text{ if } x \in \mathcal{R}_i,\ i = 1, \ldots, N. \tag{2.2}$$

The distance between two consecutive decision thresholds $\Delta_i = d_i - d_{i-1}$ is known as the *quantizer step-size* in the region $i$, and can be used to distinguish two major

groups of quantizers. If $\Delta_i$ is constant ($\Delta_i = \Delta, \forall_i$) the quantizer is classified as *uniform*, otherwise it belongs to the class of *non-uniform* quantizers. Figures 2.1 a) and b) present the characteristic of a uniform and a non-uniform quantizer, respectively.

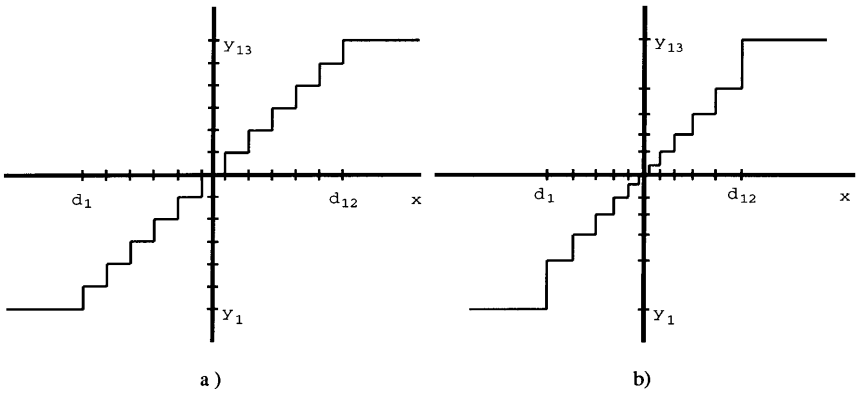


Figure 2.1: Quantizer transfer function: a) uniform, b) non-uniform.

### 2.1.2 The structure of the scalar quantizer

A very insightful analysis of the structure of a quantizer is presented by Gersho in [10]. Although oriented to vector quantizers, this analysis is equally useful to understand scalar quantization and, in particular, the conditions for optimality to be presented later.

A quantizer can be best understood if decomposed into two major building blocks: an encoder and a decoder. The encoder analyses the input, i.e. it is responsible for finding the region to which the input amplitude belongs. The decoder synthesizes the output, i.e. it is responsible for the generation of the reconstruction value associated with the region determined by the encoder.

The encoder can be modeled by a group of *selector* functions $\mathcal{S}_i$, each associated with one of the quantization regions. If the input is in the region $\mathcal{R}_i$, the output of the selector function $i$ will be one and the outputs of all the other selection functions will be zero

$$\mathcal{S}_i(x) = \begin{cases} 1, & \text{if } x \in \mathcal{R}_i \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

The outputs of all the selector functions are connected to a binary *address generator* which encodes the $N$-tuple $\mathcal{S}(x) = (\mathcal{S}_1(x), \ldots, \mathcal{S}_N(x))$ made of one "1" and $N-1$ "0"s, into a binary $b$-tuple, where

$$b = \log_2(N) \tag{2.4}$$

is the bit rate of the quantizer. This binary $b$-tuple is then transmitted to the decoder.

In the decoder, an *address decoder* recovers the $N$-tuple $\mathcal{S}(x)$ and outputs the reconstruction value associated with the non-zero position of $\mathcal{S}(x)$. Since the reconstruction values are predetermined, this operation can be expressed as

$$\mathcal{Q}(x) = \sum_{i=1}^{N} y_i \, \mathcal{S}_i(x). \tag{2.5}$$

The overall structure of the quantizer can thus be represented as in figure 2.2. It is interesting to note that the nonlinearity of the quantizer is inherent to the encoder. The decoder performs a linear operation.

### 2.1.3 Evaluation of quantization performance

Since it performs a many-to-one type of mapping, the use of a quantizer always implies the introduction of a certain amount of distortion in the quantized signal. The goal of a good quantizer design procedure is, therefore, to minimize this distortion. In order to treat this minimization problem mathematically, two parameters need to be established: a mathematical model of the input signal, and a function to provide a measure of the distortion.

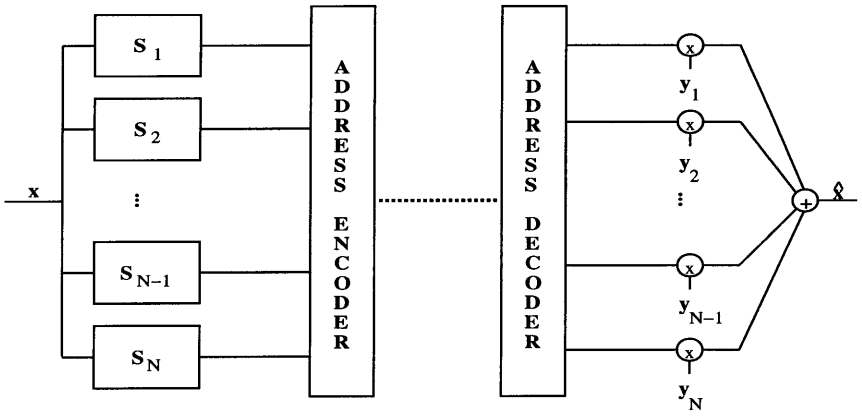


Figure 2.2: The structure of a quantizer.

Since the input is typically not known in advance (otherwise there would be no point in coding it), the input signal is generally modeled according to its statistical properties, i.e. considering it as a set of samples of a stochastic process with characteristics that resemble those of the real data.

As far as the distortion measure is concerned, an universally acceptable measure has not yet been found. A good measure of distortion should be simple enough to be mathematically tractable, and yet provide an objective criteria for evaluation of coding fidelity well matched to the complex subjective criterion used by humans. These two requirements turn out to be highly conflicting and, in practice, more emphasis is given to the aspect of mathematical tractability. Thus, despite its incapacity to take into account the properties of the *Human Visual System (HVS)*, the *Mean Square Error (MSE)*

$$MSE = ||x - y||^2, \tag{2.6}$$

is the most commonly used distortion measure.

Let us, for the moment, stick with a general distortion measure $d(x, y)$. The *average distortion* $\mathcal{D}$ originated by a scalar quantizer, with input $x$ characterized by

the *probability density function* $p_x(x)$, is

$$\mathcal{D} = \int_{-\infty}^{\infty} d(x, \mathcal{Q}(x))\, p_x(x) dx. \tag{2.7}$$

For an $N$-level quantizer, from equations 2.1, 2.2 and 2.7

$$\mathcal{D} = \sum_{i=1}^{N} \int_{d_{i-1}}^{d_i} d(x, y_i) p_x(x) dx, \tag{2.8}$$

and an *optimal quantizer*, should minimize $\mathcal{D}$.

### 2.1.4 Optimal quantization

Due to the highly non-linear nature of quantization, a closed-form solution to the problem of designing the quantizer that, for a specific bit rate, minimizes the distortion is, in general, not available. It is possible, however, to find a set of conditions *necessary* for optimality, that are also *sufficient* for some of the most common pdfs. The key to find these conditions consists in dividing the global minimization problem into two distinct sub-problems, matched to the structure of the quantizer as described in section 2.1.2:

- Problem 1: Find the optimum encoder for a given decoder.
- Problem 2: Find the optimum decoder for a given encoder.

#### The optimal encoder for a given decoder

Suppose that the optimum decoder (set of reconstruction levels $y_i, i = 1, \ldots, N$) is known, and we wish to find the encoder (set of decision thresholds $d_i, i = 0, \ldots, N$) that minimizes the overall distortion as given by equation 2.7.

Obviously, the overall distortion will be minimum if each input sample is quantized

with minimum distortion, i.e. if for a given input amplitude, the closer (in the sense of the distortion measure) reconstruction level is chosen. Mathematically,

$$\mathcal{R}_i \subset \{x : d(x, y_i) \leq d(x, y_j), \forall j \neq i\}, \tag{2.9}$$

or

$$\mathcal{Q}(x) = y_i, \text{ if } d(x, y_i) \leq d(x, y_j), \ \forall j \neq i, \tag{2.10}$$

since, in this case,

$$\mathcal{Q}(x) = \min_{y_i} [d(x, y_i)] \tag{2.11}$$

and

$$\mathcal{D}_j = \int d(x, y_j) p_x(x) dx > \int d(x, y_i) p_x(x) dx = \mathcal{D}_{min}, \ \forall j \neq i. \tag{2.12}$$

When the *MSE* is considered, equation 2.10 determines that, for a particular input value $x$, the chosen reconstruction level must be the one which minimizes $(x - y_i)^2$ (or equivalently, $|x - y_i|$). This will happen if and only if the decision threshold is the midpoint between the successive reconstruction levels, i.e.

$$d_{i-1} = \frac{y_{i-1} + y_i}{2}, \ i = 1, \ldots, N-1 \tag{2.13}$$
$$d_0 = -\infty, \ d_N = \infty.$$

**The optimal decoder for a given encoder**

Suppose now that the optimal encoder is known, and we want to find the optimal decoder for this encoder. From the definition of distortion (equation 2.7),

$$\mathcal{D} = E[d(x, \mathcal{Q}(x))] = \sum_{i=1}^{N} P_i \, E[d(x, y_i) | x \in \mathcal{R}_i]. \tag{2.14}$$

Thus, for any non-negative distortion measure, the distortion is minimum when all the terms in the summation are minimized. The problem of finding the optimal

decoder can, therefore, be reduced to finding the $y_i$ such that

$$\mathcal{Q}(x) = \min_{y_i}\{E[d(x, y_i)|x \in \mathcal{R}_i]\}. \tag{2.15}$$

The term on the right-hand side of this equation is usually known as the *generalized centroid* of the random variable $x$, in the region $\mathcal{R}_i$, with respect to the distortion measure $d(x, y)$, and the equation is generally referred to as the *generalized centroid condition.* When the distortion measure is the MSE, the solution consists in the set of $y_i$ that minimizes

$$\mathcal{D}' = \int_{\mathcal{R}_i} (x - y_i)^2 \, p_{x|\mathcal{R}_i}(x)\, dx, \tag{2.16}$$

and can be easily obtained by solving $\partial\mathcal{D}'/\partial y_i = 0$. In this case, the optimal decoder is given by

$$y_i = E[x|x \in \mathcal{R}_i] = \frac{\int_{d_{i-1}}^{d_i} x p_x(x) dx}{\int_{d_{i-1}}^{d_i} p_x(x) dx}, \; i = 1, \ldots, N. \tag{2.17}$$

This result could also be obtained from the theory of optimal least-square estimation [35].

Equations 2.13 and 2.17 establish a pair of conditions *necessary* for optimal quantization *under the MSE criterion.* There are, however, input pdfs for which those conditions are not *sufficient* for optimality [27], in which case the distortion is only locally minimum.

### 2.1.5 High-rate quantization

It was shown in the previous sections that, due to the non-linear nature of quantization, it is impossible to obtain, in general, a single, closed-form solution to the problem of optimal quantizer design. There is, however, one situation - *high-rate quantization* where such a solution exists, making the problem much simpler. The assumption is that, for high bit-rate, the decision thresholds are so closely spaced that the input pdf can be considered constant between any two consecutive decision

thresholds

$$p_x(x) = p_x(y_i),\ d_{i-1} < x \leq d_i, \tag{2.18}$$

and, under the MSE criterion, equation 2.8 becomes

$$\mathcal{D} = \sum_{i=1}^{N} \frac{p_i}{\Delta_i} \int_{d_{i-1}}^{d_i} (x - y_i)^2 dx, \tag{2.19}$$

where $p_i$ is the probability of $x \in \mathcal{R}_i$

$$p_i = \int_{d_{i-1}}^{d_i} p_x(x)dx = p_x(y_i)\Delta_i. \tag{2.20}$$

Setting $\partial\mathcal{D}/\partial y_i = 0, i = 1, \ldots, N$ results in

$$y_i = \frac{d_i + d_{i-1}}{2}, \tag{2.21}$$

i.e., for the high-rate quantizer, the optimal reconstruction levels are the midpoints between successive decision thresholds. Plugging equation 2.21 into 2.19, it is possible to obtain a closed-form solution for the minimum distortion as a function of the quantizer step-size

$$\mathcal{D} = \sum_{i=1}^{N} \frac{p_i {\Delta_i}^2}{12}. \tag{2.22}$$

### Uniform quantization

Since for a uniform quantizer $\Delta_i = \Delta, \forall_i$, equation 2.22 is, in this case, simplified into $\mathcal{D} = \Delta^2/12$. Assuming a bounded input of amplitude range equal to the range of quantizer reconstruction levels $R = d_N - d_0$ and a quantizer with $b$ bits, the step-size is given by $\Delta = R/2^b$.

Defining the *Signal to Noise Ratio (SNR)* as

$$SNR = 10\log_{10}(\frac{\sigma_x^2}{\mathcal{D}}), \tag{2.23}$$

where $\sigma_x^2$ is the variance of the input signal, it is possible to obtain a direct relation between the quantizer bit-rate and distortion

$$SNR = 6.02b - 10\log_{10}(\frac{\gamma^2}{3})dB, \tag{2.24}$$

where $\gamma = R/2\sigma_x$ is the *loading factor* of the quantizer.

Equation 2.24 provides a useful rule of thumb for quantizer design which says that, for an uniform quantizer, the SNR improves by approximately 6 dB per quantization bit. It also seems to suggest that the SNR increases with the decrease of the loading factor. This is, however, only partially true since decreasing $R$ implies that the input and output amplitude ranges will no longer be the same, and overload errors, which may cause significant performance degradation, will appear [20].

### Non-uniform quantization

A non-uniform quantizer can be seen as a composition of the three building blocks represented in figure 2.3: a *compressor*, a *uniform quantizer*, and an *expander*. The most important parameter of this model is the compressor characteristic $c(x)$, which defines the non-uniformity of the quantizer. The expander characteristic is just the inverse of that of the compressor.

The analysis carried out for the uniform quantizer is significantly more complex in the non-uniform case. However, it can be shown [20] that the optimal high-rate non-uniform quantizer is characterized by

$$c_{opt}(x) = \frac{R}{2}\frac{\int_0^x \sqrt[3]{p_x(x)}dx}{\int_0^{R/2} \sqrt[3]{p_x(x)}dx}, \tag{2.25}$$

originating

$$\mathcal{D}_{min} = \frac{2}{3N^2}[\int_0^\infty \sqrt[3]{p_x(x)}dx]. \tag{2.26}$$

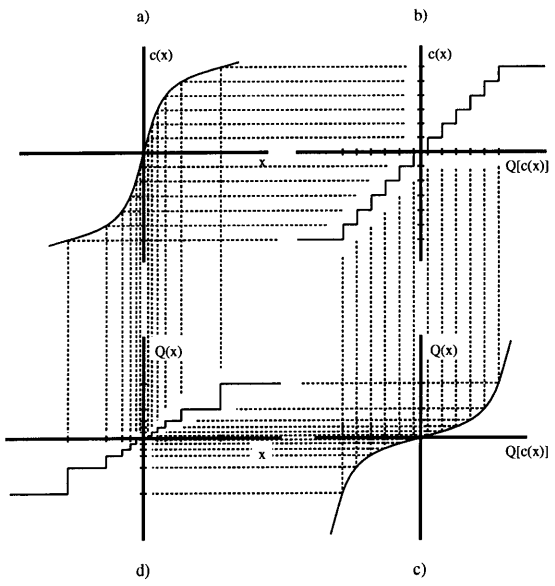


Figure 2.3: The compander model of a non-uniform quantizer: a) compressor, b) uniform quantizer and c) expander d) equivalent non-uniform quantizer.

Obviously, these equations are not as useful as those obtained for the uniform quantizer. In any case, equation 2.21 together with equation 2.24 in the uniform case, and equation 2.26 in the non-uniform one, provide a closed-form solution to the problem of optimal quantization under the assumptions of high-bit rate, a bounded input, and the MSE distortion criterion. In practice, these assumptions do not always hold, and iterative algorithms based on equations 2.13 and 2.17 are necessary (see chapter 4).

## 2.2 From scalars to vectors: Transform coding

The previous section presented the necessary conditions to achieve the optimal scalar quantizer for a given input. However, typical inputs are by nature not scalar, and so, even when this optimal quantizer can be found, it does not provide (by itself) an efficient solution to the problem of image coding. In fact, the main limitation of the scalar quantizer is that, processing a sample at a time, it cannot explore any of the redundancies that typically exist in the input signal. This can be illustrated with the

following example, originally presented in [28].

Suppose a source which generates two-dimensional random vectors $\mathbf{x} = [x_1\, x_2]^T$, characterized by the joint pdf $p_{\mathbf{x}}(\mathbf{x}) = p_{x_1,x_2}(x_1, x_2)$ represented in figure 2.4.

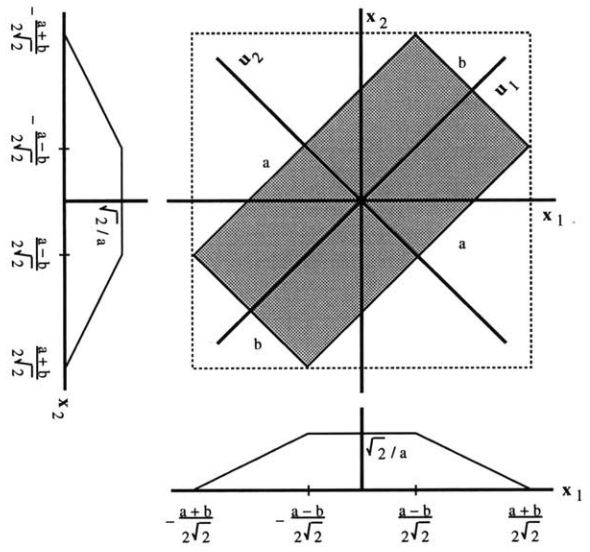


Figure 2.4: Example of a two dimensional pdf, $p_{x_1,x_2}(x_1, x_2)$, uniform in the shaded area and null on the outside. The marginal pdfs, $p_{x_1}(x_1)$ and $p_{x_2}(x_2)$, are also represented. From [28].

It is clear from the marginal probability densities also represented that $p_{x_1,x_2}(x_1, x_2) \neq p_{x_1}(x_1)p_{x_2}(x_2)$, i.e. that $x_1$ and $x_2$ are not independent. This can be confirmed intuitively by noticing that the pdf is oriented in the direction of the line $x_1 = x_2$, and so the probability that the two components have similar amplitudes is higher than that of the components having very distinct values. In other words, the value of $x_1$ will be, in general, close to that of $x_2$, i.e. there is redundancy between the two components.

Figure 2.5 shows the partition of the two-dimensional space when an optimal 8 level scalar quantizer is used separably on each of the components. This partition is clearly sub-optimal since some of the partitions are in regions of zero input probability, and lower distortion could be obtained if it were possible to have all the two-dimensional reconstruction values inside the shaded rectangle.

However, significantly higher coding efficiency can be achieved with a simple ro-

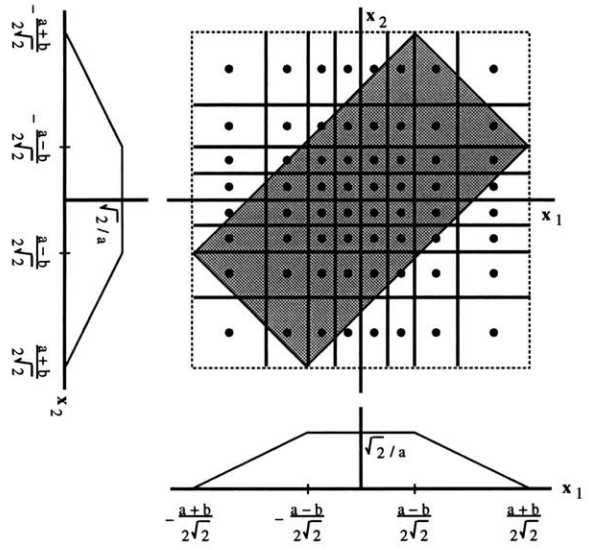


Figure 2.5: Partition of the 2-D space with scalar quantization. The dots represent the reconstruction value associated with each region of the partition.

tation of the coordinate axis. Figure 2.6 shows the same joint pdf of figure 2.4, but rotated such that the new coordinate axis are $u_1 = x_1 + x_2$ and $u_2 = x_2 - x_1$. Since, after the rotation, $p_{u_1,u_2}(u_1, u_2) = p_{u_1}(u_1)p_{u_2}(u_2)$ the random variables become independent, and it is possible to avoid zero-probability partitions even with scalar quantization. The distortion is, therefore, smaller than that of figure 2.5 for the same number of quantization levels.

When, as in this example, the input random variables can be rendered independent by a simple linear transformation, they are said to have *linear dependencies.* In this case, the use of scalar quantization preceded by this linear transformation is a very efficient coding procedure. In practice, it is generally not possible to make the variables independent due to the presence of *non-linear dependencies* in addition to the linear ones. It is, however, always possible to render the variables uncorrelated, and still achieve high coding efficiency. This is the principle of the *transform coding.*

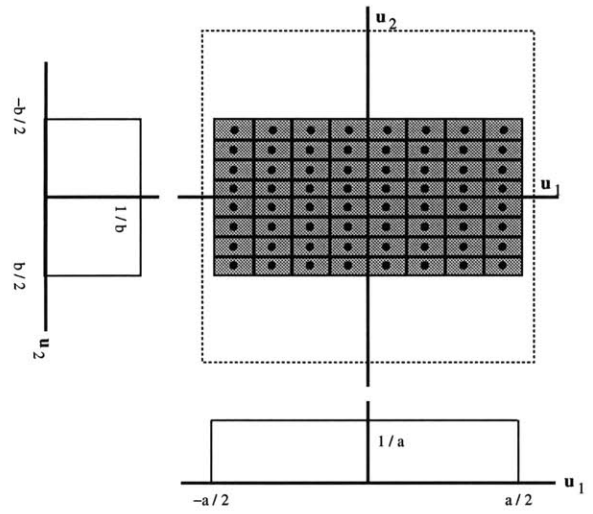


Figure 2.6: Joint pdf after rotation of the coordinate axis. The marginal densities and the partition regions associated with separable scalar quantization with 8-levels are also shown. The rotation makes the random variables independent, and the distortion is minimized. The dashed line is included for comparison with figures 2.4 and 2.5.

### 2.2.1 Transform coding

Suppose that the input samples are grouped into vectors of dimension $K$, processed by a linear transformation, quantized, and transformed back to the original domain. This set of operations is commonly known as transform coding.

The linear transformation maps the input vector $\mathbf{x} = [x_1, \ldots, x_K]^T$ into the vector $\mathbf{u} = [u_1, \ldots, u_K]^T$ according to

$$\mathbf{u} = \mathbf{T}\mathbf{x}. \tag{2.27}$$

The row vectors of $\mathbf{T}$, $\mathbf{t_i}^T$, are generally known as the *basis functions* of the transform, and since

$$u_i = \mathbf{t_i}^T\mathbf{u} = \sum_{j=1}^{K} t_{ik}\, u_k,\ k = 1, \ldots, K, \tag{2.28}$$

the components of the transformed vector $\mathbf{u}$ (known as the *transform coefficients*) are nothing more than the projections of $\mathbf{x}$ into these basis functions. It is, therefore, a

necessary condition that the set of basis functions can span the entire $K$-dimensional space, i.e. the basis vectors have to be *linearly independent.* Otherwise, different input vectors would be mapped into the same output, introducing distortion.

When, in addition to being independent, the basis vectors satisfy

$$\mathbf{t_i}^T\mathbf{t_j} = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j, \end{cases} \tag{2.29}$$

the transform matrix is *orthonormal.* Orthonormality is a desirable property because the energy of the input vectors is not modified by an orthonormal transformation, and the inverse of an orthonormal transformation is very simple to compute. This follows directly from 2.29, since

$$\mathbf{T}\mathbf{T}^T = \mathbf{I}, \tag{2.30}$$

where $\mathbf{I}$ is the identity matrix, and thus

$$\mathbf{T}^{-1} = \mathbf{T}^T, \tag{2.31}$$

i.e. the inverse transformation can be obtained with a simple transposition of the transform matrix. The invariance of signal energy under orthonormal transformation has the desirable side effect that, under the MSE, the distortion introduced by quantization in the transform domain is not modified by the inverse transformation to the input data domain

$$MSE = E[||\mathbf{x} - \hat{\mathbf{x}}||^2] = E[(\mathbf{T}^T(\mathbf{u} - \hat{\mathbf{u}}))^T(\mathbf{T}^T(\mathbf{u} - \hat{\mathbf{u}}))] = E[||\mathbf{u} - \hat{\mathbf{u}}||^2], \tag{2.32}$$

and the quantizer can be optimized independently of the transform operation.

An efficient transform for image coding must have two properties:

- decorrelate the input samples so that, as seen in the example of figures 2.4 and 2.6, the quantization can be effective;

- compact the energy of the input signal in as few coefficients as possible so that, by discarding (or quantizing coarsely) low-energy coefficients, it can be possible to achieve high compression ratios with minimal distortion.

Both of these properties are maximized by the *Karhunen-Loeve Transform (KLT).*

### Karhunen-Loeve Transform

Consider a sequence of random variables $x_k$, $k = 1, \ldots, K$ grouped into a random vector $\mathbf{x}$ of mean $\bar{\mathbf{x}}$, and covariance matrix $\mathbf{\Lambda}_x$

$$\mathbf{\Lambda}_x = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T] \tag{2.33}$$

where $\lambda_{ij}$, the element in the $i^{th}$ row and $j^{th}$ column of $\mathbf{\Lambda}_x$, is the covariance between the random variables $x_i$ and $x_j$

$$\lambda_{ij} = E[(x_i - \bar{x}_i)(x_j - \bar{x}_j)]. \tag{2.34}$$

Since $\lambda_{ij} = \lambda_{ji}$, the matrix $\mathbf{\Lambda}_x$ is symmetric and, consequently, has a complete set of orthonormal eigenvectors $\mathbf{v_i}$ [40], defined by

$$\mathbf{\Lambda}_x \mathbf{v_i} = \mu_i \mathbf{v_i}, \; i = 1, \ldots, K, \tag{2.35}$$

where $\mu_i$ are the eigenvalues associated with the eigenvectors $\mathbf{u_i}$.

Consider the matrix $\mathbf{T}$ whose rows are the eigenvectors $\mathbf{v_i}$ of $\mathbf{\Lambda}_x$, and suppose that this matrix is used to perform the linear transformation of the input vector $\mathbf{x}$, described by equation 2.27, before quantization. The covariance matrix of the vector of transform coefficients $\mathbf{u}$ thus obtained is, from equations 2.27 and 2.33,

$$\mathbf{\Lambda}_u = E[\mathbf{T}(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{T}^T] = \mathbf{T}\mathbf{\Lambda}_x \mathbf{T}^T. \tag{2.36}$$

Since the rows of $\mathbf{T}$ are the eigenvectors of $\mathbf{\Lambda}_x$ and are orthonormal, it is easy to show from equation 2.35 that 2.36 reduces to

$$\mathbf{\Lambda}_u = diag(\mu_1, \ldots, \mu_K), \tag{2.37}$$

i.e. the input vector is transformed into an array of *uncorrelated* coefficients of variance $\mu_i$.

This transform is commonly known as the Karhunen-Loeve Transform, and is optimal in the sense that it always originates a set of totally uncorrelated coefficients. What it does is precisely the rotation to a system of coordinates aligned with the principal components of the data, exemplified by figures 2.4 and 2.6.

It can be shown [3] that the KLT is also optimal in the sense of *energy compaction.* The distortion introduced by the truncation of a number $n$ of transform coefficients is minimum when the KLT is used, and the $K - n$ coefficients retained are the projections of the input $\mathbf{x}$ onto the eigenvectors of the covariance matrix associated with the $K - n$ eigenvalues of largest magnitude.

Although optimal, the KLT is generally not used in practice due to its high complexity. As seen above, the basis functions of this transform are the eigenvectors of the covariance matrix of input data which must be, in a practical application, computed on the fly if optimality is to be achieved.

The computation of eigenvectors is a difficult task in the general case, but becomes particularly heavy in the case of the KLT since the matrix itself is not known in advance and must also be computed. This not only increases the computational burden, but also introduces delay since several input vectors must be known for the covariance matrix to be computed. In addition to complexity, the KLT also introduces overhead since the basis functions must be transmitted to the decoder that has no access to the covariance matrix of the input.

Obviously, some of these disadvantages (such as delay and transmission overhead)

can be eliminated by using sub-optimal recursive procedures based on previously transmitted data, at the cost of increased complexity in the decoder. In the extreme case, the additional complexity of the KLT can be completely eliminated by assuming a statistical model for the input, pre-computing the basis functions, and downloading them to the decoder before transmission. This solution is, however, clearly sub-optimal and, since a probabilistic model suited to the vast range of possible inputs to a typical encoder has not been found to date [3], turns out to be worst than the use of other sub-optimal but of much simpler implementation transforms, such as the *Discrete Cosine Transform (DCT).*

### 2.2.2 Discrete Cosine Transform

Several definitions of the DCT have been presented in the literature. In this thesis, we will use the one given in [20]. According to this definition the 1-D DCT is an orthonormal transform described by

$$u_k = \sqrt{\frac{2}{N}}\alpha_k \sum_{i=0}^{N-1} x_i \cos\frac{(2i+1)k\pi}{2N},\ k = 0,\ldots,N-1 \tag{2.38}$$

where $N$ is the dimension of the input vector, and

$$\alpha_k = \begin{cases} 1/\sqrt{2}, & \text{if } k = 0 \\ 1, & \text{if } k \neq 0. \end{cases} \tag{2.39}$$

The 1-D *inverse DCT (IDCT)* is defined as

$$x_i = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha_k u_k \cos\frac{(2i+1)k\pi}{2N},\ i = 0,\ldots,N-1. \tag{2.40}$$

By writing equation 2.38 in the matrix form of 2.27, it can easily be seen that the

basis functions of the DCT are

$$\mathbf{t}_k = \sqrt{\frac{2}{N}}\alpha_k \cos\frac{(2i+1)k\pi}{2N},\ i = 0, \ldots, N-1. \tag{2.41}$$

Figure 2.7 presents the basis function of the 1-D DCT for $N = 8$. From equa-

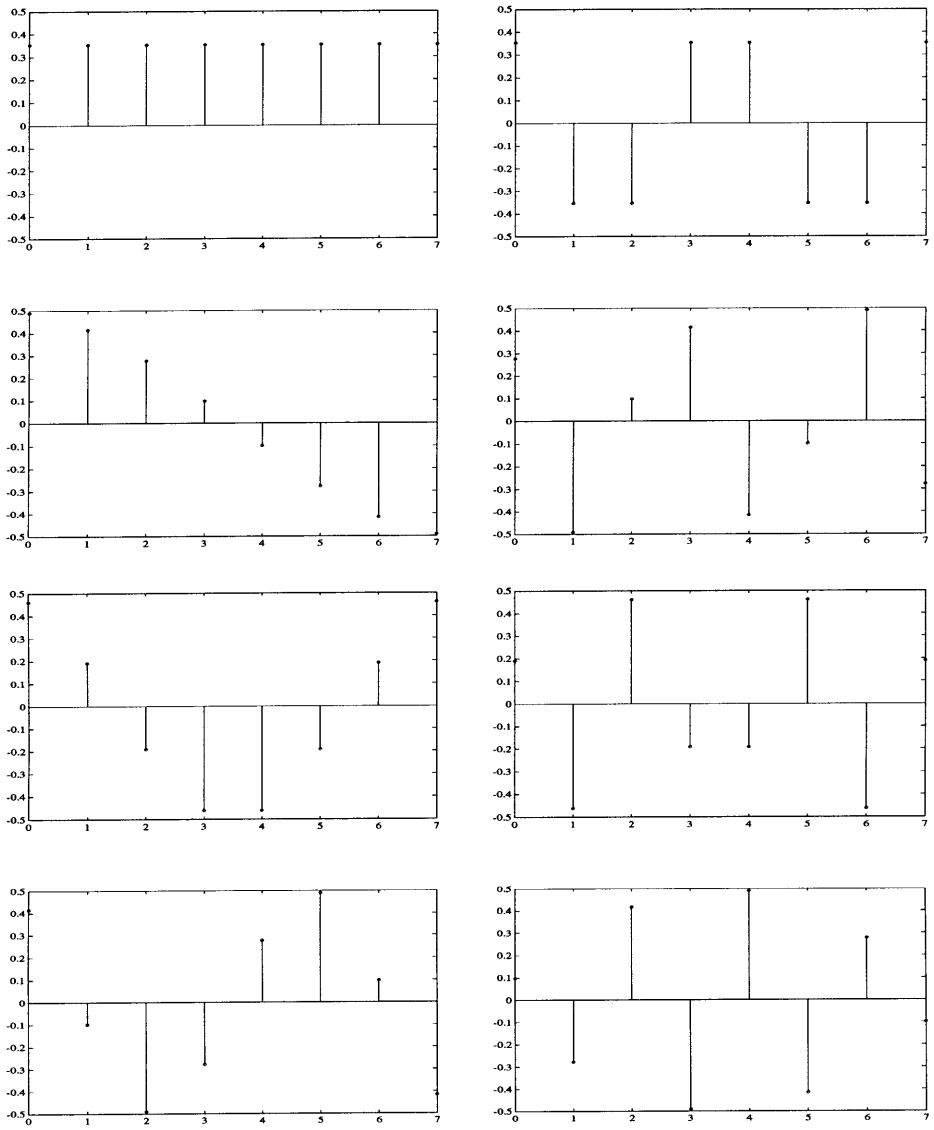

Figure 2.7: Basis Functions of the 1-D DCT of dimension 8. Left column, top to bottom: basis functions $\mathbf{t}_0$ to $\mathbf{t}_3$. Right column, top to bottom: basis functions $\mathbf{t}_4$ to $\mathbf{t}_7$.

tion 2.27, and since the DCT is an orthonormal transform,

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{u} = \mathbf{T}^{T}\mathbf{u} = \sum_{i=0}^{N} u_i \, \mathbf{t_i}, \tag{2.42}$$

i.e. the transform domain coefficients are the weights with which the basis functions must be combined to obtain the input vector. From this and the representation of the DCT basis functions in fig 2.7, it results that this transform performs a decomposition of the input signal into a sum of cosines of increasing frequency. In particular, the value of coefficient $u_0$ is the mean or DC value of the input vector and, thus, this coefficient is commonly known as the *DC coefficient.* The other coefficients, associated with basis vectors of zero-mean, are known as *AC coefficients.*

The extension of the 1-D DCT to the 2-D DCT, commonly used in image coding, is straightforward. In fact, the 2-D DCT can be obtained by the separable application of the 1-D DCT to the rows and to the columns of the input data. In this case, the $N$ basis vectors of figure 2.7 are extended to a set of $N^2$ "basis pictures" that can be obtained by performing the outer products between all the 1-D basis vectors [3].

It was mentioned before that the desired properties for an efficient transform are high-energy compaction and high-signal decorrelation. It was also seen that the KLT is optimal in respect to these aspects, but very complex. The advantage of the DCT is that, although being sub-optimal, it can be easily implemented through fast algorithms [37] and still achieve performance close to that of the KLT. In fact, it can be shown that, for simple statistic models of the input commonly used for the theoretical evaluation of transform efficiency such as the first-order Gauss-Markov, the performance of the DCT is asymptotically close to that of the KLT for highly correlated inputs [20]. It is, therefore, common to trade the slight increase of performance obtained with the KLT by the much simpler implementation of the DCT.

### 2.2.3 Quantization in the transform domain

The application of a transform maps the input vector of correlated pixels into a vector of approximately uncorrelated coefficients. Therefore, unlike the original data, the coefficients of the transformed vector have distinct properties, which must be taken into account in the design of an efficient transform domain quantizer. Also, since distinct coefficients are associated with different frequency components of the input signal, an efficient coding scheme must be able to explore the sensitivity of the human eye to distortion at those different frequencies.

The quantization of a vector of transform coefficients with a given average bit rate is composed by two distinct problems: bit allocation and quantizer design.

#### Bit allocation

Since the transform coefficients have different statistical properties and perceptual weights, it would not make sense to divide the available bit rate equally between them. It was seen in the previous section that one of the objectives of applying a transform is to compact the energy in as few coefficients as possible. It would, therefore, be inefficient to code these coefficients with the same step size as those that have no energy at all. This intuitive argument reflects a property that is characteristic of the theoretical optimal solution, first presented by Huang and Schulteiss [16]: *the bit rate allocated to each coefficient must be proportional to the ratio of the energy in that coefficient to the total energy.* The theoretically optimal bit allocation is given by

$$b_i = \frac{B}{N} + \frac{1}{2}\log\frac{\sigma_i^2}{\left[\prod_{k=0}^{N-1}\sigma_k^2\right]^{1/N}}, \tag{2.43}$$

where $b_i$ and $\sigma_i^2$ are, respectively, the bit rate allocated to and the variance of coefficient $i$, $B$ is the overall bit rate, and $N$ the vector dimension.

This equation can easily be extended to accommodate the perceptual relevance

of each of the bands by substituting all the $\sigma_i^2$'s by $\omega_i \sigma_i^2$, where $\omega_i$ is the perceptual weight of each band, and $\sum_{i=0}^{N-1} \omega i = 1$. Notice that equal division of the bit rate between all the coefficients $b_i = B/N$ is optimal only if the coefficients have equal variances, which is clearly not the case in transform coding.

Equation 2.43 provides the theoretically optimal bit allocation, but is not always usable in practice because it does not satisfy the practical constraints that the individual bit rates $b_i$ must be integer[1] and positive. In practice, equation 2.43 can be used as an initial estimate for the bit allocation, but has to be followed by an algorithm that, based on this estimate, distributes the available bits by the different coefficients satisfying the constraints mentioned above.

A possible alternative to equation 2.43 is to use an algorithm to distribute the available bits between the coefficients from the beginning. The idea is to start from a point where all coefficients are assigned zero bits, and iteratively assign the next bit to the quantizer that contributes most to the overall distortion. This procedure is not optimal since the best decision at each instant may not result in the overall best distribution of bits, but is guaranteed to provide an integer and non-negative distribution.

The bit allocation obtained by one of the procedures described above is, in practice, usually fine-tuned using heuristics and common-sense. For example, if a coefficient has variance smaller than the desired average MSE, there is no need to spend any bits at all in this coefficient since, in this case, the distortion will be equal to the variance. This is the idea behind a very common technique known as *threshold coding* [33], where a coefficient is quantized only if its amplitude exceeds a pre-defined threshold. Another, even simpler, approach is that of *zonal sampling* [33], where only a pre-determined set of coefficients of the transformed vector is transmitted. The rea-

[1] The constraint of an integer number of bits can be relaxed in some applications. An example of one such application is the use of a variable word-length, or entropy, coder after quantization. Another example is when vector quantization is used instead of scalar quantization.

soning behind this approach comes from the energy compaction property of transform operation mentioned in section 2.2.1. Due to this property, for typical inputs, most of the energy will be concentrated in a subset of the coefficients (in the case of the DCT, the low-frequency ones) and, for most of the vectors, the remaining coefficients can be discarded without significant degradation. Obviously, there are vectors for which this property does not hold (such as those in active areas of the input) and care must be taken to avoid significant degradation of those vectors.

### Quantizer design

After an efficient bit allocation is achieved, there still remains the problem of the design of the optimal quantizer for each coefficient. Since distinct coefficients have different characteristics, it seems natural for each coefficient to require its own quantizer. In practice, it turns out that some of the coefficients have similar characteristics, and simpler solutions can be implemented without significant decrease in performance. The characteristics of the coefficients are obviously dependent on the transform used, but the basic ideas of what will be next discussed for the DCT hold for all the commonly used transforms.

As we have seen in section 2.2.2, the transform coefficients are, in the case of the DCT[2], associated with a frequency decomposition of the input vector. In particular, the first coefficient is just the DC value of this vector. This property makes this DC coefficient different from the remaining ones.

The first characteristic of the DC coefficient is that it is very important in terms of subjective coding quality. In fact, the human eye is very sensitive to the artificial discontinuities on average luminance, "blocking effect", that appear between adjacent vectors when this coefficient is coarsely quantized. This is generally taken into account by the bit allocation algorithm, allocating a relatively high bit-rate to the

[2] And, also, in the case of various other transforms such as the DFT, Walsh-Hadamard, etc.

DC coefficient.

The second characteristic is that the pdf for this coefficient is similar to that of the input. This can be easily understood with the following procedure. Suppose an input image of size $M$x$M$ is split into vectors of dimension $N$x$N$, a 2-D DCT is applied to each of these vectors, and all the DC coefficients grouped together to form an image of size $M/N$x$M/N$. This smaller image is just a low-pass downsampled replica of the larger original, and thus the probability of occurrence of each possible amplitude value in it will be similar to that of the larger original image. Therefore, the pdf of the DC coefficient is similar to that of the input samples. Since the input pdf is typically uniform, the same happens to the DC coefficient, which is normally coded using a *uniform* quantizer.

The AC coefficients do not have the crucial subjective importance of the DC, and turn out to have probability densities which are approximately Gaussian or generalized Gaussian[3] [3]. This leads to the common assumption that the AC coefficients are all Gaussian, making it possible to use a unique quantizer optimized for a unit-variance *Gaussian* distribution, if each coefficient is divided by its variance prior to quantization.

## 2.3 Vector Quantization

In the previous sections, we have seen that a scalar quantizer alone cannot provide efficient compression, since it fails to exploit the dependencies between consecutive samples of the input. It was shown that a significant performance increase can be obtained by the use of a decorrelating transform prior to quantization. However, the use of a scalar quantizer is always sub-optimal since, for $K$-dimensional vectors, the

[3]Note that, since each coefficient is a weighted sum of the input samples, from the central limit theorem of probability [5], a Gaussian pdf should be theoretically expected if the input samples were independent and the vector dimension large enough.

space can only be partitioned into $K$-dimensional hypercubes. This restriction can be eliminated through the use of a *Vector Quantizer (VQ)* [1, 28, 13, 32].

Vector quantization is an extension of scalar quantization for vector spaces, and a VQ $\mathcal{Q}$ is a mapping from a $K$-dimensional vector space of input samples to a finite set of *reconstruction vectors*, usually known as *codevectors* or *codewords*. The set of reconstruction vectors is generally designated by *codebook*. The $N$-dimensional input vector space is partitioned into a set C of $N$ $K$-dimensional *regions* $\mathcal{R}_i$, also known as *partitions* or *cells*, and a reconstruction vector $\mathbf{y}_i$ associated with each region. The partitions are delimited by $K-1$-dimensional hyperplanes, and all the input vectors in the partition $\mathcal{R}_i$ are mapped into the codeword $\mathbf{y}_i$. Mathematically,

$$\mathcal{Q} : \mathcal{R}^k \rightarrow \mathcal{C}, \tag{2.44}$$

where $\mathcal{C} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, $\mathbf{y}_i \in \mathcal{R}^k$, and

$$\mathcal{R}_i = \{\mathbf{x} \in \mathcal{R}^k : \mathcal{Q}(\mathbf{x}) = \mathbf{y}_i\}. \tag{2.45}$$

For a $K$-dimensional VQ of size $N$, the average bit rate per input sample is

$$b = \frac{\log_2 N}{K}. \tag{2.46}$$

The comparison of this equation with 2.4 reveals one of the advantages of vector over scalar quantization: unlike scalar quantizers, vector quantizers are not constrained to integer bit rates, i.e. it is possible with VQ to quantize input samples at a *fractional bit rate*. This is useful because of two main reasons. First, and as seen in section 2.2.3, integer bit-allocations can lead to sub-optimal performance. Second, it is possible to achieve bit rates below 1 *bit per sample (bps)*, which are impossible to obtain with scalar quantization alone.

### 2.3.1 The structure of the vector quantizer

The structure of a VQ is very similar to that presented in section 2.1.2 for the scalar quantizer. Once again, the quantizer can be split into an encoder-decoder pair. The encoder finds the partition to which the input vector belongs, and signals it to the decoder which performs a simple *table look-up operation* to find the reconstruction codeword associated with that partition. Notice that the encoder does not necessarily have to know the reconstruction codebook, although this is the case in common practical implementations.

The mathematical model presented in section 2.1.2, based on a set of selector functions, an address encoder/decoder and a summation, and figure 2.2 are still valid if all the building blocks are extended to perform vector operations. Thus, the output of the VQ can be seen as

$$\mathcal{Q}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{y}_i \, \mathcal{S}_i(\mathbf{x}), \tag{2.47}$$

where $\mathbf{x}$ is the input vector, $N$ the codebook size, $\mathbf{y}_i$ the reconstruction codewords, and $\mathcal{S}_i(\mathbf{x})$ the selector functions

$$\mathcal{S}_i(x) = \begin{cases} 1, & \text{if } \mathbf{x} \in \mathcal{R}_i \\ 0, & \text{otherwise.} \end{cases} \tag{2.48}$$

### 2.3.2 Evaluation of quantizer performance

The general expressions for the distortion introduced by a vector quantizer are very similar to those presented in section 2.1.3. For an input characterized by a $K$-dimensional pdf $p_{\mathbf{x}}(\mathbf{x})$ and a distortion measure $d(\mathbf{x}, \mathcal{Q}(\mathbf{x}))$, the average distortion introduced by a VQ with partitions $\mathcal{R}_i$ and reconstruction codewords $\mathbf{y}_i$ is

$$\mathcal{D} = \int d(\mathbf{x}, \mathcal{Q}(\mathbf{x})) \, p_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}, \tag{2.49}$$

which, for a quantizer of size $N$ can be simplified to

$$\mathcal{D} = \sum_{i=1}^{N} \int_{\mathcal{R}_i} d(\mathbf{x}, \mathbf{y}_i) p_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{N} p_i \int d(\mathbf{x}, \mathbf{y}_i) p_{\mathbf{x}|\mathcal{R}_i}(\mathbf{x}|\mathcal{R}_i) d\mathbf{x} = \sum_{i=1}^{N} p_i E[d(\mathbf{x}, \mathbf{y}_i)|\mathbf{x} \in \mathcal{R}_i]. \tag{2.50}$$

A significant difference between the scalar and vector cases is, however, that VQ allows the use of more general, and still tractable, distortion measures than the MSE. Among these, two classes are commonly used in practice:

- The class of *l-norm* distortion measures, defined by

$$d(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^l = \frac{1}{K} \sum_{i=1}^{K} |x_i - \hat{x}_i|^l, \tag{2.51}$$

  where $l$ is a positive integer. Among this class, the most commonly use distortion metrics are the *sum of absolute errors* ($l = 1$), the MSE ($l = 2$), and the *maximum absolute error* ($l = \infty$).

- The *weighted MSE* defined by

$$d(\mathbf{x} - \hat{\mathbf{x}}) = (\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{W} (\mathbf{x} - \hat{\mathbf{x}}), \tag{2.52}$$

  where $\mathbf{W}$ is a positive-definite weighting matrix. Common versions of this metric include the MSE ($\mathbf{W} = \mathbf{I}$), and a component weighted MSE provided by a diagonal $\mathbf{W}$ with distinct elements. This second type can be used to give distinct perceptual weights to the components of the input vector, which can be useful for some applications like the VQ of transform coding coefficients.

### 2.3.3 Optimal vector quantization

As in the case of scalar quantization, the non-linearity inherent to the operation of quantization makes it impossible to achieve a single, closed-form solution to the problem of optimal vector quantization. It is however possible to find two *necessary*

conditions for optimality by using the decomposition of the problem into two smaller ones: finding the optimal encoder for a given decoder, and the optimal decoder for a given encoder. The solutions to these two problems are obtained from equation 2.50, by a straightforward extension of the analysis carried in section 2.1.4.

The optimal partition (encoder) for a fixed codebook (decoder) must satisfy the *nearest-neighbor condition*

$$\mathcal{R}_i \subset \{\mathbf{x} : d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \forall j \neq i\}, \tag{2.53}$$

i.e.

$$\mathcal{Q}(\mathbf{x}) = \mathbf{y}_i, \text{ if } d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \forall j \neq i; \tag{2.54}$$

while the optimal codebook for a given partition must satisfy the *generalized-centroid condition*

$$\mathcal{Q}(\mathbf{x}) = \min_{\mathbf{y}_i}\{E[d(\mathbf{x}, \mathbf{y}_i)|\mathbf{x} \in \mathcal{R}_i]\}. \tag{2.55}$$

When the MSE is used as distortion measure, the optimal partition for a given codebook becomes

$$\mathcal{R}_i \subset \{\mathbf{x} : ||\mathbf{x} - \mathbf{y}_i||^2 \leq ||\mathbf{x} - \mathbf{y}_j||^2, \forall j \neq i\}, \tag{2.56}$$

or

$$\mathcal{Q}(\mathbf{x}) = \mathbf{y}_i, \text{ if } ||\mathbf{x} - \mathbf{y}_i||^2 \leq ||\mathbf{x} - \mathbf{y}_j||^2, \forall j \neq i; \tag{2.57}$$

and the optimal codebook for a given partition

$$\mathbf{y}_i = \mathcal{Q}(\mathbf{x}) = \{E[\mathbf{x}|\mathbf{x} \in \mathcal{R}_i]\}. \tag{2.58}$$

As in the case of scalar quantization, the above conditions are only *necessary* for global optimality, i.e. a VQ for which they are satisfied can be locally, instead of globally, optimal. Notice that a local optimum can sometimes be very poor, as

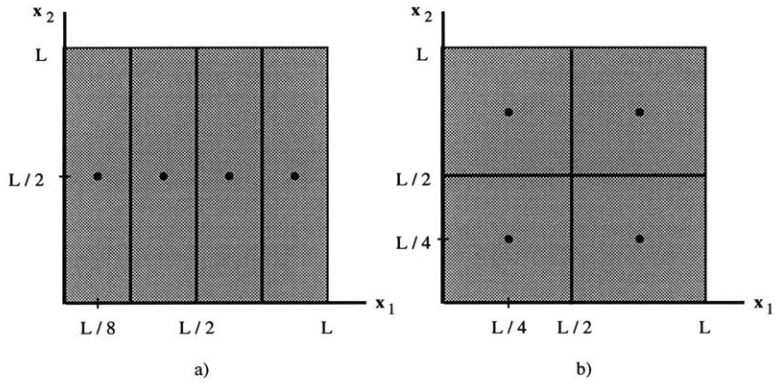


Figure 2.8: Example of poor VQ performance due to local optimality.

exemplified by figure 2.8.

This figure presents two possible partitions of the input space, and respective optimal codeword assignment, for the quantization of two-dimensional vectors with the uniform pdf of amplitude $1/L^2$ in the shaded area and null outside. Both quantizers satisfy the necessary conditions of optimality of equations 2.57 and 2.58, and yet the two solutions are significantly different. Using equation 2.50 and the MSE as distortion measure, it can be shown that for a) the distortion is $\mathcal{D}_a = 65L/192$, while for b) $\mathcal{D}_b = L/12$ that is $\mathcal{D}_b/\mathcal{D}_a \approx 25\%$! Thus, despite satisfying the necessary conditions for optimality, the quantizer of a) is clearly sub-optimal.[4]

### 2.3.4 The efficiency of vector quantization

It was seen in the previous sections that a scalar quantizer is very inefficient, since it fails to explore the redundancies that generally exist between successive samples of the input. It was also seen that the use of a linear transformation prior to quantization can increase significantly the coding efficiency when the input samples are linearly dependent (correlated).

[4] This result makes intuitive sense since in a) all the bits are allocated to the first vector component, while in b) the bits are equally distributed among the two components.

However, for most of sources found in practice, the input samples are not only linearly, but also non-linearly dependent. Since transform coding, being a linear operation, cannot explore non-linear dependencies, it fails to achieve the optimal performance that can only be obtained through the use of vector quantization.

In fact, since transform coding performs only a rotation of coordinates, a transform coder with scalar quantization is just a special case of a vector quantizer, whose cells are obtained from a rotation of the partition of the space obtained with a scalar quantizer. This is illustrated by figure 2.9, which presents the partition associated with a VQ that, for the 2-D input of figure 2.4, achieves the coding performance of transform coding plus scalar quantization as represented in figure 2.6. Therefore, VQ has the same capacity to explore linear dependencies between samples as transform coding plus scalar quantization.

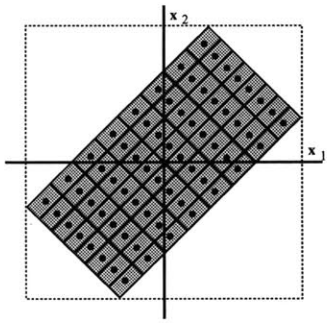


Figure 2.9: Partition of the input space of figure 2.4 associated with a vector quantizer with equal performance to that of transform coding and scalar quantization as shown in figure 2.6.

The great advantage of VQ over transform coding is that it can also explore non-linear dependencies between the input samples. This derives from the fact that a VQ is not constrained to the regular partition of the space imposed by the scalar quantization of the transform coefficients, as illustrated by the following example.

Suppose a 2-D input random vector with the pdf of figure 2.10. Since $E(x_1x_2) = E(x_1)E(x_2) = 0$, there is no linear dependence between the vector components, and thus no benefits can be gained from the application of a linear transformation prior to quantization. However, scalar quantization is still very inefficient as can be seen in

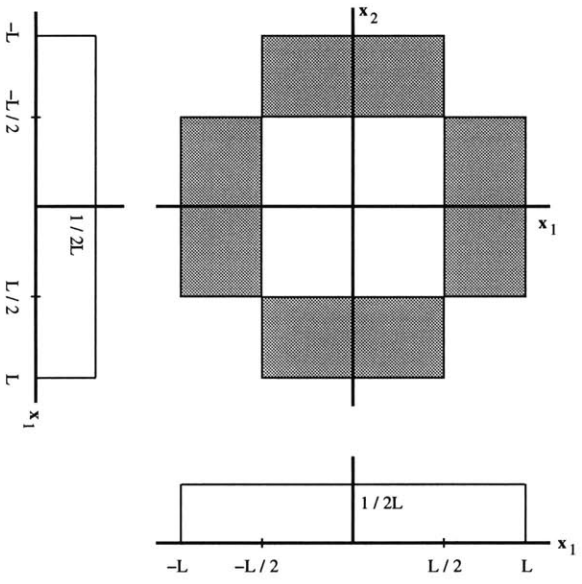


Figure 2.10: 2-D input pdf of a vector with linearly independent, but non-linearly dependent components. From [24].

figure 2.11 a), which shows the partition of the space provided by separable optimal scalar quantization with an overall rate of 2 bps. Notice that half of the codewords are in regions of zero input probability, and that exactly the same performance can be obtained by simply eliminating those codewords as shown in figure 2.11 b). This structure is, however, impossible to achieve with a scalar quantizer, demonstrating the advantage of VQ, which achieves the same performance at 3/4 of the bit rate.

In addition to the capacity to explore non-linear dependencies between input samples, VQ has also the capability to explore the *dimensionality* of the input vector space, a property that allows increased performance over the achievable with scalar quantization even when the input samples are statistically independent. This property is due to the freedom with which the shape of each quantizer cell can be chosen in VQ, as opposed to the strictly cubic shape required for separable scalar quantization of the vector components. An interesting example of this property, where 2-D VQ with hexagonal cells is shown to be more efficient than separable scalar quantization, even when the input vector components are *uniform independent and identically distributed (iid)* random variables, can be found in [28]. Another example is given by

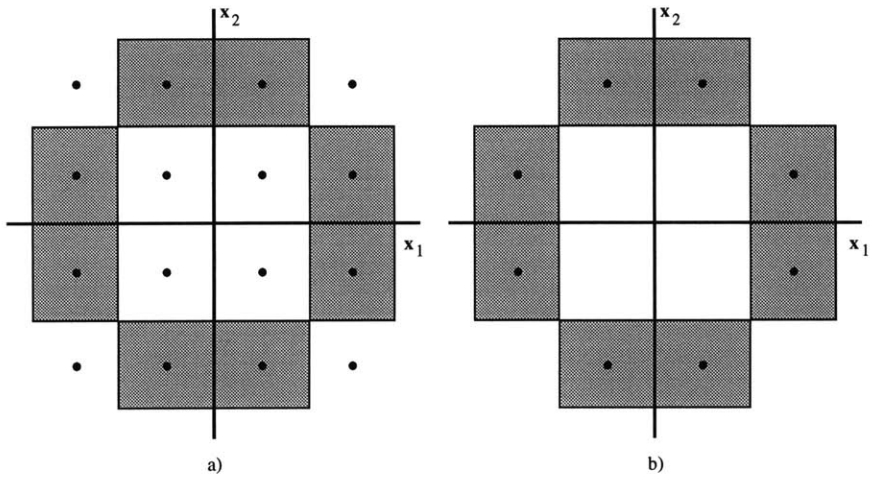


Figure 2.11: Partition of the 2-D space, and respective codewords, obtainable with a) optimal scalar quantization at 2 bps and b) vector quantization at 1.5 bps.

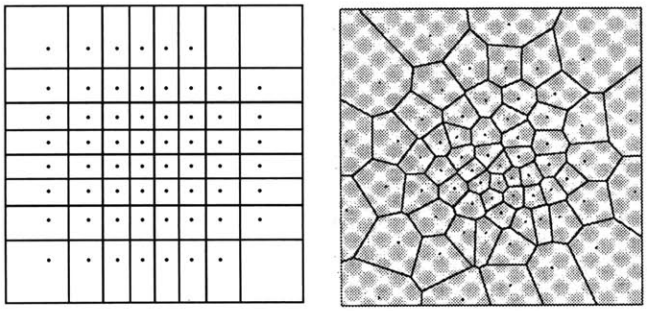

Figure 2.12: Optimal partition of the space for an iid Gaussian source, achieved with scalar quantization (left) and vector quantization (right). The picture on the right was taken from [11].

figure 2.12, which shows the optimal partition of the space for an iid Gaussian source and 2-D vectors, obtained with separable scalar quantization and vector quantization. Being restricted to rectangular cells, the scalar quantizer is less efficient than the optimal vector quantizer.

Due to its capability to explore non-linear dependencies of the input data and dimensionality, a vector quantizer is the *theoretical optimal solution* to the problem of coding sequences of waveform samples. In practice, however, some limitations prevent VQ from achieving this optimal theoretical performance.

One of this limitations, already mentioned in the previous section, is the inexis-

tence of a set of *sufficient* conditions for optimal VQ design. As shown above, the design of a vector quantizer can, in practice, lead to a local maximum of performance significantly inferior to the best theoretical maximum. In this case, the performance can be substantially worse than the expected, and inferior to that obtained with other methods such as transform coding, etc.

Another important limitation is imposed by implementation complexity constraints. Due to the lack of structure of a typical VQ, the process of encoding consists, in general, of measuring the distance from the input vector to all the codewords in the codebook and picking the closest one. For a bit rate $b$ and $k$-dimensional vectors, this process requires the evaluation of $N = 2^{bk}$ distortion measures by input vector, i.e. the complexity grows exponentially with vector dimension. On the other hand, since the number of degrees of freedom in the partition of the space increases with vector dimension, the greater efficiency of VQ can only be truly achieved with large vector sizes. A trade-off between efficiency and computational complexity is therefore needed in practice, which usually limits the gains obtained with VQ.

## 2.4 From 2 to 3 dimensions: interframe coding

The techniques discussed in the previous sections achieve high-coding efficiency by processing vectors of input samples, and exploring the redundancy between those samples. In typical image coding applications, the vectors are formed by two-dimensional blocks of pixels, enabling the coding algorithms to explore the spatial correlation existent in both the horizontal and vertical dimensions of the image.

Since image sequences typically present high correlation in the temporal dimension (in addition to the spatial correlation), it would seem natural to extend the techniques previously discussed to the temporal domain. Such an extension would require the use of 3-D cubes processed by 3-D transform coding or 3-D vector quantization and, consequently, the capacity to store a number of frames equal to the vector size in

the temporal dimension at both the encoder and the decoder. Since, as seen in the previous sections, the increase in efficiency obtained with transform coding or VQ is proportional to the vector dimension, an efficient 3-D scheme would require the storage of several frames.

The need for frame storage has two main practical drawbacks: it imposes hardware complexity leading to expensive implementation, and it introduces delay which can be unacceptable for many applications. In practice, these drawbacks limit the maximum number of frames that is possible to store to one or two, imposing a significant limitation to the performance of any of the 3-D schemes referred above. Furthermore, these schemes are not suited to incorporate motion compensation techniques that, as will be seen in section 2.4.2, provide a very efficient way of reducing temporal redundancy. Therefore, 3-D transform coding or vector quantization are in general not used, and most practical implementations of image sequence coding rely, in the temporal dimension, on a technique that does not require large frame storage and is well suited to incorporate motion compensation: *predictive coding*.

### 2.4.1 Predictive coding

The basic idea behind predictive coding is that if there is redundancy in a sequence of samples, it is possible to estimate a given sample from those already processed with reasonable accuracy. By subtracting this *prediction* to the sample to code, the redundancy existent in the input signal can be extracted before quantization, resulting in a *prediction error signal* which is much easier to compress than the original itself. If the prediction is based uniquely on samples already transmitted, the decoder in the receiving end is able to replicate it and, by adding it to the quantized prediction error signal, reconstruct a replica of the original input.

Figure 2.13 presents the block diagrams of a predictive encoder and decoder. The introduction of the decoder in the encoder feedback loop guarantees that, being based in reconstructed samples, the prediction of the decoder is equal to that of the encoder.

In fact, since

$$e_i - \hat{e}_i = x_i - \tilde{x}_i - (\hat{x}_i - \tilde{x}_i) = x_i - \hat{x}_i \tag{2.59}$$

the reconstruction error is equal to the quantization error, i.e. the quantization error does not propagate to subsequent samples.

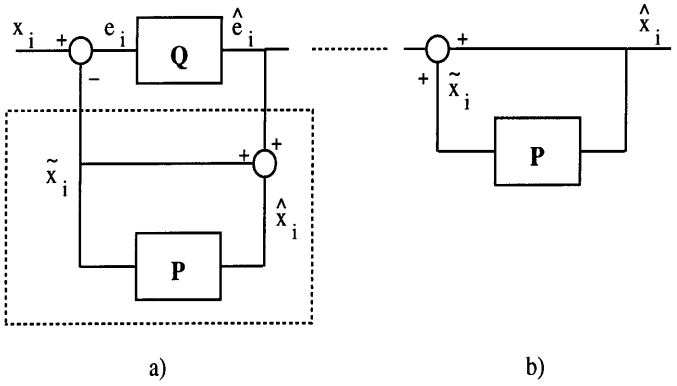


Figure 2.13: Block diagram of a) a predictive encoder and b) decoder. P is the predictor and Q the quantizer. The encoder portion inside the dashed rectangle is a replica of the decoder.

Due to the feedback loop structure and the non-linearity of quantization, it is difficult to determine the optimal predictor and quantizer to be used in a predictive coder. In practice, these components are designed separately, the quantizer using the formulas presented in section 2.1.4, and the predictor assuming perfect-reconstruction, i.e. no quantizer in the loop. The predictor is in general restricted to a linear combination of previous samples weighted by a set of *predictor coefficients* $\omega_k$ according to

$$\tilde{x}_i = \sum_{k=1}^{N} \omega_k x_{i-k}, \tag{2.60}$$

where $N$ is the predictor order.

If all the samples used for prediction are from the same frame as the sample to be coded, the predictor is referred to as *intraframe predictor*, and the coding process as *intraframe coding*. Otherwise, the predictor is an *interframe predictor*, and the coding process *interframe coding*.

Unlike transform coding or vector quantization, the efficiency of predictive coding

does not improve significantly with the number of samples considered in the prediction of the input value. In fact, it has been shown that for typical images the gain achieved with predictive coding saturates with $3^{rd}$ order predictors [14]. This can be explained by the recursive nature of the prediction operation which implies that, independently of the predictor order, the current prediction is influenced by a large number of previous samples. Therefore, the only effect of a change in prediction order is a different weighting of the most recent prediction samples, which does not have a drastic effect in the overall coding efficiency.

In practice, predictors of order greater than two or three are rarely found in image coding applications, and although less efficient than VQ or transform coding with large vector dimensions (4x4, 8x8 or 16x16), predictive coding is more efficient than these techniques when the vector dimensions are constrained to small values, as in the case of temporal processing. Also, unlike the other techniques, predictive coding can easily incorporate motion compensation, becoming a very powerful technique for the elimination of the temporal redundancy characteristic of typical image sequences.

### 2.4.2 Motion estimation and compensation

While spatial redundancy is determined mainly by the detail of each input frame, temporal redundancy is a function of the motion between successive frames. Since typical scenes are composed by moving objects, and the movement of each object is constrained by the physical laws of nature, it is natural to expect temporal redundancy to be more deterministic than spatial redundancy. In particular, if it were possible to estimate the motion of each object in the scene, a very accurate prediction of frame $t$ could be obtained by applying the estimated displacement to each of the objects in frame $t-1$.

Unfortunately, the problem of estimating the motion of the objects in a scene is very complex, mainly because the problem of decomposing a frame into the set of objects that compose it (usually designed by *segmentation*) is itself complex. In fact,

these two problems are highly interrelated since it is difficult to perform an efficient segmentation without motion information, and it is even harder to estimate motion without knowing which objects compose the scene (segmentation).

In practice, common motion estimation methods perform an arbitrary segmentation of the image into blocks (ignoring its content), and try to estimate the motion relative to each block. Although somewhat ad-hoc, block-based techniques have the advantage of matching well with the vector-based compression techniques discussed in the previous sections, and this has gave them wide-spread acceptance by the image compression community.

Among several techniques presented in the literature, such as frequency domain (also known as *phase correlation*) methods [36], methods based on image gradients [25], and recursive methods [34], *block matching* [31] has proven to be the most effective and is generally used for interframe coding.

Block matching consists of breaking the image to code into blocks and, for each block, find the closest match in a previous frame. In general, only a small area of pre-specified size of the previous frame, usually referred to as the *search window*, is used in the search. The best match is the one which minimizes a pre-specified distortion measure, typically the MSE or the *mean absolute error (MAE)*. Under the MSE criteria, block matching finds, for each block, the motion vector $(d_x, d_y)$ which minimizes

$$\sum_{x,y \in \mathcal{R}} [I(x,y,t) - I(x - d_x, y - d_y, t-1)]^2, \tag{2.61}$$

where $I(x,y,t)$ is the intensity of frame $t$ for the pixel with coordinates $(x,y)$, and $\mathcal{R}$ is the search window. Under the MAE criterion, the squaring operation in equation 2.61 is substituted by the absolute value. The two criteria lead to similar performance, and the choice between them is generally related to implementation constraints.

Block matching is based on three assumptions:

- all the motion in the scene is translational and parallel to the plane of the

camera;

- nearby pixels of the same object have equal motion;
- the size of revealed or occluded areas is small relative to the image size.

Since these assumptions hold, at least approximately, for most of typical input sequences (specially if the block size is small when compared with the average size of the objects in the scene), block matching is a very robust and reasonably efficient motion estimation technique. Its main drawback is the computational weight of the search operation which, in practice, limits the prediction to be based on one or at most two previously transmitted frames.

Once the best estimate of motion is achieved, motion compensation can be performed easily. For each block in frame $t$ the best match in frame $t-1$, according 2.61, is displaced by the corresponding motion vector $(d_x, d_y)$ and subtracted from it, originating a low-energy residual that can then be easily coded.

## 2.5 Entropy coding

Using quantization to achieve compression, all the image coding techniques presented so far imply degradation of the input signal. For some applications, this distortion is not allowable, and a different type of compression, generally referred as *lossless coding*, has to be applied. Also, lossless coding techniques can be applied after quantization to remove any redundancy still present in the encoded signal.

Lossless coders perform a one-to-one mapping, where each possible output codeword is associated with a *unique* input *symbol* or *word*, and achieve compression by exploring the statistical properties of the input data. The main idea is that, if smaller codewords are associated with more probable input symbols and larger codewords with the less probable ones, it is possible to obtain an average rate smaller to

that achievable with a *fixed rate code*, where all the input symbols are mapped into codewords of the same length.

### 2.5.1 Scalar entropy coding

Consider a scalar source which produces symbols from a finite alphabet $\mathcal{A} = \{\omega_1, \ldots, \omega_N\}$[5] and a lossless encoder that maps these symbols into codewords of length $l_i, i = 1, \ldots, N$. Given the probabilities of occurrence of the input symbols $(p_i)$, the theoretically optimal lossless code is the one which minimizes the average rate per symbol of the encoded output

$$b = \sum_{i=1}^{N} p_i \, l_i. \tag{2.62}$$

In practice, in addition to minimize the average rate, the optimal code has to satisfy the requirement of *unique decodability*. The importance of this requirement is illustrated by table 2.5.1, where two 3-symbol codes, $\mathcal{C}_1$ and $\mathcal{C}_2$, of equal average rate are represented. If code $\mathcal{C}_1$ is used, when the sequence of bits `110` is transmitted

Table 2.1: Two possible codes for a 3-symbol source. The code on the right is uniquely decodable, while that on the left is not.

| symbol | $\mathcal{C}_1$ | $\mathcal{C}_2$ |
|---|---|---|
| $\omega_1$ | 0 | 0 |
| $\omega_2$ | 11 | 10 |
| $\omega_3$ | 110 | 110 |

the decoder has no way to determine if the input sequence of symbols was $\omega_2\omega_1$ or simply $\omega_3$. Thus, code $\mathcal{C}_1$ is not uniquely decodable and is, in practice, totally useless. Code $\mathcal{C}_2$ is uniquely decodable, since it satisfies the *prefix condition* that none of the codewords is a prefix of any other codeword. It can be shown [11] that for a given

[5] One example of such a discrete source is the output of a scalar quantizer.

alphabet of size $N$, a uniquely decodable code must satisfy the *Kraft inequality*

$$\sum_{i=0}^{N} 2^{-l_i} \leq 1, \tag{2.63}$$

and, if this inequality can be satisfied, there exists a uniquely decodable code. Thus the optimal lossless code is the one that minimizes equation 2.62, constrained to the condition 2.63.

From equation 2.62, with $q_i = 2^{-l_i}$,

$$b = \sum_{i=1}^{N} p_i \log_2 \frac{1}{q_i} \tag{2.64}$$

and since $\ln x \leq x - 1$,

$$\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} - \sum_{i=1}^{N} p_i \log_2 \frac{1}{q_i} = \frac{1}{\ln 2} \sum_{i=1}^{N} p_i \ln \frac{q_i}{p_i} \leq \frac{1}{\ln 2} \sum_{i=1}^{N} (q_i - p_i). \tag{2.65}$$

Since $\sum_{i=1}^{N} p_i = 1$ and, from the Kraft inequality, $\sum_{i=0}^{N} q_i \leq 1$,

$$b \geq H(p), \tag{2.66}$$

where $H(p)$ is the *entropy* of the source

$$H(p) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}. \tag{2.67}$$

Equations 2.66 and 2.67 state one of the fundamental results of the information theory, first established by Shannon [39]: *"the minimum possible rate achievable by lossless coding the symbols of a give source is the entropy of that source"*. The optimal lossless coder is, therefore, the one whose output rate is closer to the entropy of the source.

Notice that from equations 2.62 and 2.67, the entropy bound of 2.66 can be achieved with equality only if $p_i = q_i = 2^{-l_i}$, i.e. if the symbol probabilities are

powers of two. In the general case, any code with codewords of length given by

$$2^{-l_i} \leq p_i \leq 2^{-l_i+1} \tag{2.68}$$

will satisfy the Kraft inequality since, in this case, $\sum_{i=1}^{N} 2^{-l_i} \leq \sum_{i=1}^{N} p_i = 1$. Writing equation 2.68 as

$$-\log_2 p_i \leq l_i \leq -\log_2 p_i + 1, \tag{2.69}$$

multiplying each term by $p_i$ and summing for all $i$,

$$H(p) \leq b \leq H(p) + 1, \tag{2.70}$$

i.e. *a uniquely decodable scalar code of average rate less that one bit higher than the source entropy can always be found.*

### 2.5.2 Vector entropy coding

Suppose now that, instead of coding each input symbol separately, $K$ symbols are grouped into a vector which is then lossless coded. In this case, the input alphabet becomes $\mathcal{A}' = \{\omega_1, \ldots, \omega_{\mathbf{M}}\}$, and the *average rate per vector* is

$$\mathbf{b} = \sum_{i=0}^{M} \mathbf{p}_i \, l_i, \tag{2.71}$$

where $M$ is the number of possible input vectors, $\mathbf{p}_i$ the probability of the $i^{th}$ vector, and $l_i$ the length of the associated codeword. The *vector entropy* of the source is, in this case, defined by the generalization of equation 2.67

$$\mathbf{H(p)} = \sum_{i=1}^{M} \mathbf{p}_i \log_2 \frac{1}{\mathbf{p}_i}. \tag{2.72}$$

Since only the length of the codewords assigned to the input symbols, and not the symbols itselves, are constrained by the Kraft inequality, this inequality is still valid

in the vector case. Thus, the analysis carried out in the previous section can be easily extended to obtain the vector equivalent of equation 2.70

$$\mathbf{H(p)} \leq \mathbf{b} \leq \mathbf{H(p)} + 1. \tag{2.73}$$

From this equation, and since the *average bit rate* and *entropy per symbol* are, for an iid source, $\mathbf{b}/K$ and $\mathbf{H(p)}/K$

$$H(p) \leq b \leq H(p) + \frac{1}{K}. \tag{2.74}$$

*It is thus theoretically possible to achieve average bit rates arbitrarily close to the entropy of the source by coding blocks of symbols, if the complexity and delay introduced by block-based processing are allowable.*

### 2.5.3 Huffman coding

It was seen in the previous sections that it is theoretically possible to achieve average bit rates close to, but not lower than the entropy of the source. In practice several methods have been proposed to design optimal codes, with performance close to the established by the entropy bounds of equation 2.74 [17, 22, 4, 42]. Among these, *Huffman coding* [17] has received wide-spread attention and use in the data compression community.

A Huffman code is a table mapping input messages to variable-length codewords designed according to the probability distribution of the input and satisfying the prefix condition, through the following algorithm:

1. List all possible input messages and consider these messages as leaves (terminal nodes) of a binary tree. Associate with each node the probability of the respective message.

2. Find the nodes in the list associated with the two least probable messages and create a parent node of probability equal to the sum of the probabilities of these nodes. Label the branch from the parent node to one of the children with a binary `1` and the branch to the other child with a binary `0`.

3. Replace the two child nodes in the list by their parent and associated probability. Stop if this probability is one, otherwise go to step 2.

4. Follow the path from the root node to each of the leaves. The sequence of bits associated with the branches in this path is the codeword of the message corresponding to the leaf.

It can be shown [17] that this algorithm achieves the theoretically optimal code satisfying the boundaries of 2.74. However, and since in practice it is impossible to enumerate all possible source messages and keep track of their probabilities, this optimality has to be traded with a more easily implementable procedure. Thus, the algorithm is typically applied to source symbols instead of messages, and since the codewords are restricted to have integer length this can lead to sub-optimal performance, as illustrated by the following example.

Suppose a source with an alphabet B of $N+1$ symbols $\mathcal{B} = \{\omega_1, \ldots, \omega_{N+1}\}$ with an highly skewed probability distribution $p_1 = 0.9$ and $p_2 = \ldots = p_{N+1} = 1/10N$. Since the last $N$ symbols are equi-probable the optimal code for the alphabet $\mathcal{B}' = \{\omega_2, \ldots, \omega_{N+1}\}$ is a fixed length code of $\log_2 N$ bits. Thus, when applied to B, the Huffman algorithm will originate a set of codewords with length $l_1 = 1$ and $l_2 = \ldots = l_{N+1} = 1+\log_2 N$, and an average length of $b = 1+(\log_2 N)/10$, significantly higher that the entropy $H(p) = 0.46+(\log_2 N)/10$, for reasonable values of N. Notice, that due to the restriction of integer-length codewords, the first symbol requires one bit as opposed to the $-log_2 0.9 = 0.15$ bit required to achieve the entropy. Since this symbol is highly probable, this difference is the main reason for the sub-optimal performance of the code.

In general, it can be shown that the difference between average rate and entropy is bounded, for an Huffman code, by $p + \log_2[2(\log_2 e)/e] = p + 0.086$, where $p$ is the probability of the most probable symbol in the input alphabet [9]. I.e., the higher the non-uniformity of the input probability distribution, the more sub-optimal Huffman coding becomes.

Obviously, the inefficiency imposed by integer-length codewords can be minimized by coding groups of symbols. However, since the size of the code tables grow exponentially and the encoding delay linearly with the block dimension, this solution turns out to be impractical for most applications. In such cases, an alternative solution is provided by *arithmetic coding*.

### 2.5.4 Arithmetic coding

Like Huffman coding, arithmetic coding achieves compression by mapping more probable input symbols to shorter codewords and less probable ones to longer codewords. Since it does not constrain the number of bits used to represent a symbol to be integer, arithmetic coding can achieve higher efficiency.

The basic algorithm of arithmetic coding is the following:

1. Divide the interval [0,1) on the real line into a set of sub-intervals each of length equal to the probability of each symbol in the input alphabet. Make this the main-interval.

2. To code the next input symbol, choose the sub-interval associated with that symbol. Make this sub-interval the new main-interval and sub-divide it in sub-intervals according to the input probabilities, as in 1.

3. Indicate to the decoder the boundaries of the current main-interval. If there are more symbols in the message go to 2, otherwise stop.

Notice, that a sequence of symbols translates into a successively smaller interval, and from this interval the decoder can *uniquely* identify the input sequence. In fact, it is not necessary for the encoder to transmit to the decoder both ends of the current interval. Any number inside the interval can be used, provided that the end of the message is signaled by a terminating symbol.

In practice, the implementation of an arithmetic coder is more complex than the simple algorithm above suggests. The main reason for this is that the arithmetic precision required to represent the interval grows with the number of symbols coded. Since real applications are constrained to finite arithmetic precision, the interval must be periodically renormalized, in a way to prevent underflow or overflow. A detailed explanation of the intricacies of arithmetic coding as well as a complete implementation in the C programming language can be found in [2].

### 2.5.5 Entropy-coded quantization

In section 2.1.4, two necessary conditions for optimal scalar quantization, given a predefined number of quantization levels, were presented. Ignoring the fact that the mentioned conditions can lead to local, as opposed to global, optimality, a quantizer that meets these conditions will achieve the minimum possible distortion obtainable with a fixed-length code. However, as discussed above, a fixed-length code is optimal only for equiprobable alphabets, and entropy coding of the quantized samples can lead to a reduction of the bit rate without any additional degradation. Thus, in practical implementations, quantization is, in general, followed by an entropy coder.

However, the use of an entropy coding stage introduces a new variable in the problem of optimal quantization, since a slight increase in average distortion may be desirable if it leads to a set of codewords with smaller entropy. In fact, the constraint of a finite number of reconstruction levels becomes unnecessary (from a theoretical point of view) when entropy coding is used since codewords of zero probability have null contribution to the average rate. It is, therefore, possible to achieve a finite

average rate with an infinite number of reconstruction levels if quantization is followed by entropy coding. Since the optimality conditions of section 2.1.4 were obtained with the assumption of a finite number of reconstruction levels, they do not provide the optimal solution for the problem of entropy-coded quantization.

Once again, given the non-linear nature of the problem, it is difficult to find a theoretically optimal solution for it. However, under the high-rate assumption of section 2.1.5, it is possible to show that, surprisingly, the optimal *entropy-constrained quantizer*[6] is the uniform quantizer [20]! It has also been shown experimentally [7] that this result is still a very good approximation in the case of low bit rates.

The best performance of the uniform quantizer over the nonuniform when entropy coding is used can be intuitively explained by the ability of entropy coding to explore the non-uniformity of the input alphabet to achieve compression. Suppose two quantizers: $\mathcal{Q}_1$, satisfying the optimality conditions of section 2.1.4, and $\mathcal{Q}_2$, uniform with an unlimited number of levels. To minimize the average distortion, the decision levels of $\mathcal{Q}_1$ will be closer in regions of high input probability and further apart in regions of low probability. Therefore, the area under the pdf between successive decision thresholds will be approximately constant for the quantizer input range, and the quantized codewords will have approximately equal probability. On the other hand, for $\mathcal{Q}_2$, which has equally spaced decision thresholds, codewords associated with high probability regions will have high probability, while those associated with low probability regions will have low probability. Therefore, although the distortion may be higher in the uniform case, the entropy will certainly be lower, and this can lead to more efficient performance. Furthermore, since $\mathcal{Q}_2$ has an unlimited number of reconstruction levels, it is not affected by the problem of quantization overload, and large input values (outside the input range of $\mathcal{Q}_1$) are quantized with smaller distortion than with $\mathcal{Q}_1$, reducing the increase in distortion inherent to the uniform quantizer. The global result of these two effects is that $\mathcal{Q}_2$ is able to achieve the average distortion of $\mathcal{Q}_1$

[6] Quantizer that minimizes the average distortion for a fixed codeword entropy.

with smaller codeword entropy, resulting in more efficient coding performance.

The optimality of uniform quantization with entropy coding and its simple implementation make this the most widely used solution to the problem of scalar quantization, when variable rates are acceptable.

# Chapter 3

# Recent advances in image coding

The fundamental techniques discussed in the previous chapter have been a subject of exhaustive research during the last decade. Thus, their potential to provide substantial gains in coding efficiency has diminished considerably in the last few years. This chapter presents some recent developments in the field of image coding.

Section 3.1.1 presents the MPEG-2 coding algorithm, the result of a world-wide research effort for the standardization of high-quality image coding. MPEG-2 can be considered as the state of the art in terms of traditional compression methods, and its performance considered to be very close to the theoretical efficiency bounds achievable with these methods.

The rest of the chapter presents a review of some new approaches to the problem of image compression that have begun to be explored in the last few years. These approaches seem to have great potential, but are still in a very early stage of development. The scope of the review is, therefore, more oriented to present the general ideas behind each technique than to discuss the techniques with the detail of chapter 2. Section 3.2 covers model-based coding, an approach based on techniques that have been used for a while in the field of computer graphics. Section 3.3 discusses object-oriented coding, an approach which combines model-based coding with the

traditional compression methods.

## 3.1 Traditional image coding

Traditional image compression algorithms have been based on digital signal processing (DSP) concepts, like those discussed in chapter 2. The main advantages of these DSP-based approaches are their robustness and mathematical tractability that have led to the establishment of theoretical performance bounds, which in turn have guided the development of practical coding schemes. Although a large number of techniques has been reported in the literature, the results of recent standardization efforts seem to point out that, in practice, it is difficult to beat the performance of interframe-DCT coders. Interframe-DCT encoding is the basis of the MPEG-2 compression algorithm, which can be presented as an example of the state of the art in DSP-based image compression.

### 3.1.1 The MPEG-2 video coding standard

The MPEG-2 standard is intended to cover a wide-range of applications. Since some of these applications require different algorithmic features, the standard is organized as a set of *profiles*. Each profile provides a specific functionality and can have different *levels*, which are generally associated with different resolutions of the input material (SIF, CCIR 601, HDTV, etc). The reasoning behind this organization is to allow efficient inter-networking between different encoders and decoders. Inter-networking between different equipment is possible on the profile and level, or profiles and levels, that they share. Also, higher level decoders must be able to decode lower level bitstreams and more complex profile decoders must be able to decode simpler profile bitstreams. This section describes briefly the *main level* of the *main profile*, which can be seen as the core of the standard, and was used in this thesis.

Although MPEG-2 only standardizes a syntax to which all valid bitstreams must conform, and the elements of the coding algorithm which constrain that syntax (such as block-size, etc.); the syntactic issues will not be dealt with special concern in here. Instead, relevance will be given to the coding procedure and its relationship with what was discussed in chapter 2.

As was already mentioned, both temporal and spatial processing are used by MPEG-2 to achieve efficient compression. In the temporal domain, the algorithm is based on motion-compensated predictive coding; while, in the spatial domain, the DCT is used to remove the redundancy of the input signal.

### Input and coding formats

Sequences to be coded according to the MPEG-2 main profile must conform to the 4:2:0 YBR format, and are, in general, derived from the 4:2:2 YBR format defined in the CCIR 601 Recommendation [19]. CCIR 601 4:2:2 YBR sequences are composed by a luminance component with 720 pixels/line, 480 lines/frame, and 8 bps; and two color components with the same number of lines and bps, and half horizontal resolution. These components can be obtained from a 24 bps RGB signal by a linear transformation defined in the 601 Recommendation. The 4:2:0 MPEG format is typically obtained from the input 4:2:2 format by down-sampling the color components to half the vertical resolution.

MPEG-2 supports both progressive and interlaced sequences, and several features of the algorithm are dependent on the type of input. For example, two picture structures are defined for interlaced sequences: *field pictures*, where each of the fields is considered as a separate entity; and *frame pictures*, obtained by merging the two fields that compose each frame into a unique entity.

The basic processing element in the MPEG-2 coding algorithm is a 8x8 block. Both the luminance and chrominance input components are segmented into a set of

8x8 blocks, which are then grouped into *macroblocks* constituted by four luminance blocks and one block from each of the color components, as shown in figure 3.1.

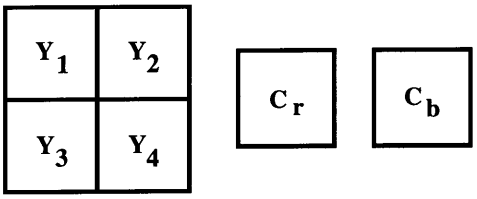


Figure 3.1: Macroblock structure.

### Temporal processing

Three types of pictures are defined by MPEG according to three different degrees of temporal processing used to minimize temporal redundancy:

- *I-frames*, which do not use temporal processing at all, and for which only intraframe coding is allowed;
- *P-frames*, where a motion-compensated prediction for the current frame is obtained from a previous frame;
- *B-frames*, where the prediction for the current frame is obtained by motion-compensated bilinear interpolation of a previous frame and a frame in the future.

Input sequences are typically processed in a *Group of Pictures (GOP)* structure, where the image sequence is broken into GOPs that start with an I-frame, followed by a pre-specified number of P-frames, evenly spaced, and separated by an also pre-specified number of B-frames. Figure 3.2 presents a common GOP configuration, composed by 15 frames, with two B-frames between consecutive P-frames or I and P-frames. The number of pictures in a GOP as well as the spacing between P-frames are parameters that can be specified according to implementation requirements.

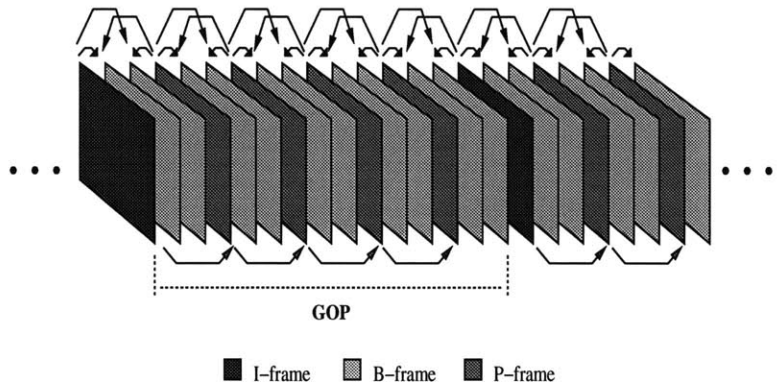


Figure 3.2: Example of the GOP structure. In this particular example, each GOP contains 15 frames and there are two B-frames between consecutive P or I and P-frames.

Due to the non-causal character of B-frames, the natural order (commonly referred as *display order*) of the frames in the input sequence cannot be used for coding. The sequence is thus re-shuffled into a *coding order*, where all the P or I-frames used in the prediction of a B-frame are coded before that B-frame. The following example shows this re-shuffling operation for a sequence with a 9-frames GOP, and two B-frames between consecutive P or I and P-frames. In this case, the display order is

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | B | B | P | B | B | P | B | B | I | B | B | P | B | B | P | ... |

and the coding order is

| 1 | 4 | 2 | 3 | 7 | 5 | 6 | 10 | 8 | 9 | 13 | 11 | 12 | 16 | 14 | 15 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | P | B | B | P | B | B | I | B | B | P | B | B | P | B | B | ... |

The re-shuffling implies that the first or the last GOP in the sequence are always smaller than the others, by the number of B-frames between consecutive P-frames.

For P and B-pictures, motion-compensation vectors with half-pixel resolution are obtained by block-matching performed on the luminance component of the input frame macroblocks, for which the best match is searched in a prediction frame through the procedures described in section 2.4.2. For B-frames, the best match from the past frame and that from the future frame are individually computed, and the macroblocks associated with these matches are averaged to obtain the best interpolative prediction,

as illustrated in figure 3.3.

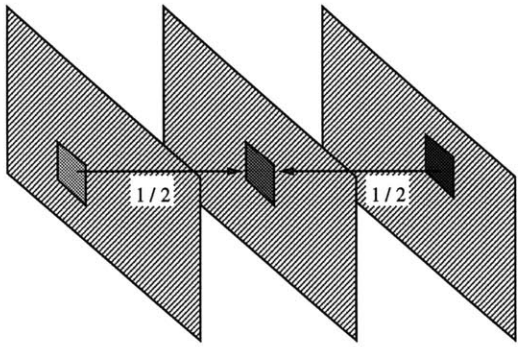


Figure 3.3: Bilinearly-interpolated prediction. The best match from the past prediction frame and the one from the future prediction frame are averaged to obtain the interpolated prediction.

After motion estimation is performed, several macroblock types are possible, depending on the frame type.

- P-frame macroblocks can be predicted from a past frame (*forward predicted macroblocks*), or not predicted at all (*intraframe macroblocks*). Typically, the energy of the residual prediction error is compared with that of the original macroblock, and the predictive mode is chosen if the former is smaller that the latter.

- B-frame macroblocks can be predicted from a past frame (*forward prediction*), from a future frame (*backward prediction*), from an interpolation of both past and future (*interpolative prediction*), or not predicted at all (*intraframe mode*). Typically, the best match of the three predictive methods is selected, and then the comparison described above is performed to choose between the best predictive mode and the intraframe mode.

Several extensions to these basic motion-compensation modes are included in MPEG to increase the coding efficiency for interlaced inputs. For frame-structured pictures, two prediction modes are defined:

- *frame prediction*, where the same motion vector is applied to the lines of both fields;
- *field prediction*, where the lines of each field can be predicted from the lines of any of the fields in the frame used for prediction, using an extra motion vector;

Generally these two modes are combined into a *field/frame adaptive* mode, where the best of the two prediction methods is chosen at the macroblock level. The field mode is more efficient in areas of strong motion, where the two fields can be displaced by different amounts; while the frame mode is more efficient in less active areas since only half the number of motion vectors is required.

An additional prediction mode, commonly referred as *dual-prime* is allowed for P-frames, trying to combine the efficiency of the field mode with the reduced overhead required by the frame mode. In the dual-prime mode, although only one motion vector is transmitted per macroblock, the lines of each field of each macroblock are predicted from lines of both fields of the prediction frame, using an interpolation process similar to that described above for B-frames. Starting with a motion vector between fields of the same parity, a new motion vector is obtained for prediction between opposite parity fields by scaling appropriately the original motion vector and adding to it a differential value of small amplitude (restricted to 0 or $\pm 1/2$), selected to optimize the prediction. The prediction for the lines of each field is then obtained by averaging the prediction from the lines of the field with the same parity and the lines from the field with opposite parity. This is exemplified in figure 3.4, where the lines of the first field of frame $t$ are predicted from the first field of frame $t-1$ with the motion vector $v_1$, and from the second field with vector $v_2$, which is nothing more than a scaled copy of $v_1$ summed to a differential $d$. In this way, at the cost of transmitting only one vector and two differentials (one per field), the prediction is actually based on four different vectors (two for each field).

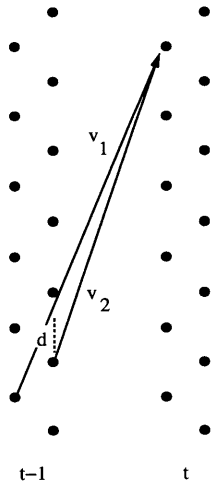


Figure 3.4: Dual-prime prediction.

### Spatial processing

The spatial redundancy existent on the input frame (for I-frames) or on the prediction error (for P and B-frames) is minimized through the application of a 8x8 DCT to each of the blocks of the input or prediction-error signal. Once again, interlace is accounted for by the definition of two DCT modes, represented in figure 3.5.

- In the *frame-DCT mode*, a 8x8 DCT is applied to the luminance and chrominance blocks that compose each macroblock, without any consideration about the interlaced nature of the input.
- In the *field-DCT mode*, the lines belonging to the first field of the four macroblock luminance blocks are merged into two blocks, and the lines belonging to the second field are merged into two other blocks. The chrominance blocks are processed as in the frame-DCT mode.

The field mode is more effective when there is motion between the fields and significant vertical detail. In this case, the vertical edges become "serrated", originating high vertical-frequencies that are hard to code if the lines from opposite fields are not separated. The frame mode is more effective when there is no motion or significant

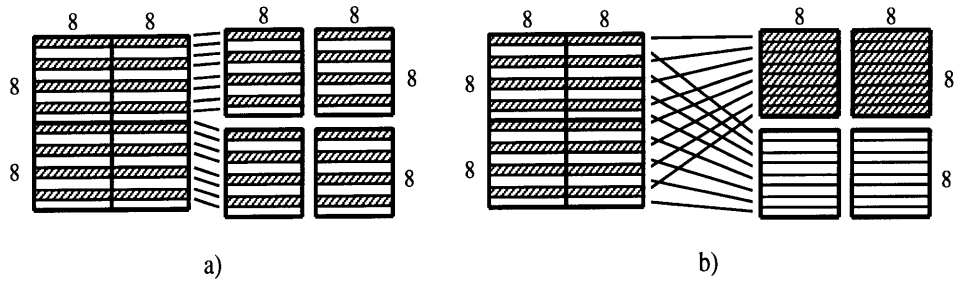


Figure 3.5: DCT modes: a) frame-DCT, b) field-DCT.

vertical detail since, for this mode, the vertical resolution is twice that obtained with the field mode.

After the DCT, the coefficients of each block are scalar quantized by a two-step procedure. First, a pre-specified quantization matrix, common to all the blocks in the frame, is applied to each block coefficients, which are afterwards scalar quantized according to the local image activity and the available transmission bit-rate.

Typical quantization matrices are slanted, so that higher-frequency coefficients are quantized with larger step-sizes, and provide a practical way to implement some of the bit allocation procedures described in section 2.2.3. First, the use of quantization matrices allows an unequal distribution of bits by the different coefficients, assuring that more bits are allocated to the low-frequency coefficients, which are the most important both subjectively (the sensibility of the human eye to quantization noise is smaller at high-frequencies) and objectively (most of the block energy is concentrated on the low-frequency coefficients). Second, quantization matrices provide an easy way to perform threshold coding because any coefficient of amplitude less that the respective quantization step-size is automatically zeroed. Since the step-size increases with frequency and high-frequency coefficients have in general small energy, this results in runs of zeros that can be easily coded.

In general, two different matrices are used for intra and predictive frames, and the matrix used with I-frames is more slanted. The reason for this is that a prediction-error image is typically high-pass; and, although the eye is still less sensitive to

quantization in active areas, if the high frequencies were very coarsely quantized, most of the information would be lost. The value of the step-size as a function of the coefficient for two typically used matrices [18] is represented in figure 3.6.

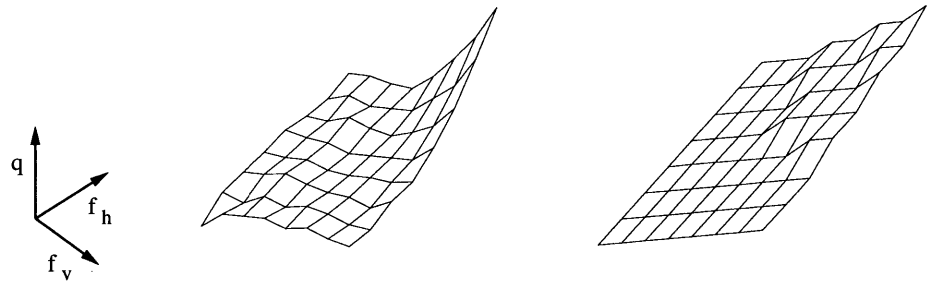


Figure 3.6: Typical quantization matrices for a) I and b) P or B-frames. $F_h$ and $f_v$ are the horizontal and vertical frequencies, and q the quantization step-size.

After matrix quantization, each coefficient is individually quantized with a scalar quantizer according to the local image activity and the fullness of the transmission buffer. Since, as will be described shortly, entropy coding is used to increase coding efficiency, a MPEG-2 coder originates a variable-rate bitstream. However, for most applications, the coded data must be transmitted over a fixed-rate channel; which can be accomplished by guaranteeing that the average rate originated by the encoder is equal to the channel rate, and introducing a transmission buffer to accommodate instantaneous deviations from the average rate.

The most common procedure used to guarantee that the average rate is equal to the channel rate consists in introducing feedback from the buffer state to the quantization step-size. When the buffer starts to fill (empty), the step-size is increased (decreased), reducing (increasing) the coding rate to a value closer to the channel rate. In MPEG-2, a different quantizer, known as *macroblock quantizer* or *mquant*, can be computed for each macroblock, and rate control is usually achieved by taking into account the buffer fullness in the computation of the mquant step-size. In addition to rate control, the chosen step-size can also take into account the local image activity, quantizing more coarsely areas of high activity (where the human eye is less sensitive

to distortion) and more accurately smooth areas.

After quantization, the DCT coefficients are entropy-coded. To reduce the overhead, the macroblocks are analyzed, and only blocks with at least one non-zero coefficient are encoded. A *coded-block pattern*, which indicates these encoded blocks, is associated with each macroblock. Also, since neighbor blocks typically have similar means, their DC coefficients are differentially coded by subtracting from the current DC coefficient the previously transmitted one.

To explore the zero runs originated by quantization, the 2-D array of coefficients is transformed into a 1-D vector by applying one of the zig-zag scan patterns illustrated in figure 3.7. The pattern a) is used with progressive inputs while the pattern b) is more suited to interlaced pictures. This vector is then coded by a run-length/amplitude encoder using the following algorithm.

1. Define a coefficient pointer, pointing to the first vector position, and initialize a zero counter to zero.

2. While the current coefficient is zero and the coefficient pointer is smaller than the last vector position, increment the pointer and the coefficient counter.

3. If a non-zero coefficient is found, transmit the contents of the zero counter and the amplitude of the non-zero coefficient. If the last vector position was not yet reached, reset the counter and go to 2.

4. Transmit an *end-of-block* marker.

Before transmission, the run/amplitude pairs and all the overhead information associated with each macroblock (motion vectors, coded-block pattern, macroblock type, etc.) is Huffman coded, as discussed in section 2.5.3.

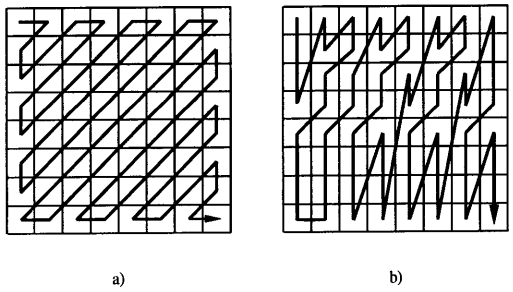


Figure 3.7: Zig-zag scan patterns.

## 3.2 Model-based coding

The facts that the main profile of the MPEG-2 standard is basically equal to the MPEG-1 algorithm [23] (previously developed for the transmission of lower resolution video - 360x240 pixels/frame - at bit-rates below 1.5 Mbps) and that MPEG-1 is itself very similar to the previous H.261 CCIR standard (for the transmission of reduced-quality video at low bit-rates - multiples of 64 Kbps) support the conclusion that it is difficult to beat the performance of interframe-DCT coding with algorithms based uniquely on the traditional compression techniques described on chapter 2.

A totally different approach to the image compression problem, that seems to have the potential to achieve much larger compression ratios, is *model-based coding*. Model-based coders rely on a pre-specified model of all the objects in the scene that can be described by a small set of parameters and animated by continuously updating those parameters. A model-based coder can be divided into the two main blocks depicted in figure 3.8. In the encoder or analyses stage, the input is compared to the model, and the representative parameters extracted. In the decoder or synthesis stage, these parameters are used to animate the model, tracking the variations of the input.

Obviously, the efficiency of a model-based scheme is determined by the accuracy with which the scene can be characterized by the model. Unfortunately, the majority of typical scenes (in particular those that are natural, non man-made) are very difficult to model with an acceptable degree of accuracy, and this has limited model-based

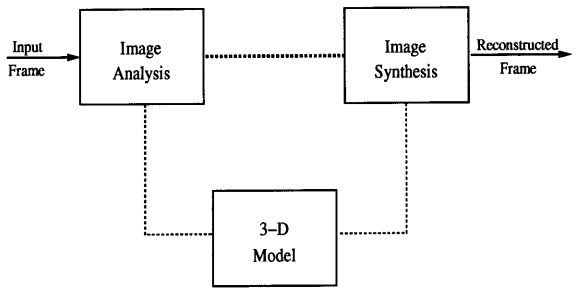


Figure 3.8: Block diagram of a model-based encoder.

implementations to a restricted set of applications. In particular, model-based coding has been applied in the recent years, with a relatively promising degree of success, to the problem of coding the "head and shoulders" scenes characteristic of video-conference or video-telephone applications. Figure 3.9 represents a model for these types of scene, based on a wire-frame of polygons. In this case, the spatial coordinates of polygon vertices are the parameters that characterize the model; and, by moving the relative positions of these vertices, it is possible to animate it.

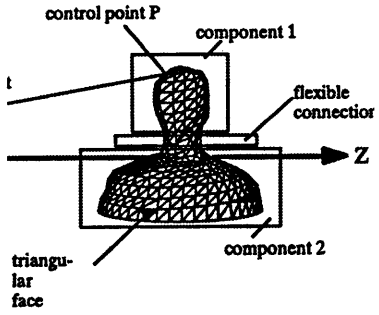


Figure 3.9: Typical model for an "head and shoulders" scene. From [21]

### 3.2.1 Image analysis

Image analysis is the most complex task to be performed by a model-based coder. An effective analysis algorithm should be able to identify object features, even when faced with adverse conditions, such as noise, camera blur, changes of illumination, and object occlusion or re-shaping. While some of these adverse conditions, such as

noise or changes of illumination, can be avoided if the scene occurs in a controlled environment; others cannot, making the analysis task very difficult. In addition, the model of the objects is, in general, crude, which obviously does not help the analysis task.

Due to these problems, analysis algorithms tend to be very specific to the particular type of scenes to be coded, and reasonably complex. The following is an example of an analysis algorithm for "head and shoulders" scenes, presented by Welsh in [36], which tries to solve the problem by exploring the symmetries of a human face.

First, an edge detector is applied to the input frame to extract the contours of all the objects in the face (eyes, nose, and mouth). The resulting image is then processed to eliminate false contours due to noise and contour discontinuities due to inefficient performance of the edge detector. A window with the length of the input frame, but only eight lines high, is then used to scan the contour image from top to bottom, with an overlap of four lines between successive window positions. For each vertical position of the scan window, the number of edge points in each of its columns is summed and stored in an array, which is then searched for two vertical lines corresponding to the sides of the head. The area in between each candidate pair is then searched for symmetric features corresponding to the eyes; and, once the eyes are found, the symmetry axis of the face is determined. Then, another window is placed below the eyes, and the number of edge points along each of its rows is summed and stored in an array which is, afterwards, searched for horizontal activity corresponding to the mouth. Incorrect identifications that may occur along the this analysis procedure are handled by incorporating a backtracking algorithm that restarts the process from a different initial guess for the head boundaries if an identification error is detected. This backtracking mechanism increases significantly the accuracy of the analysis algorithm at the cost of a greater computational load.

This analysis algorithm is conceptually very simple, and is probably not the best in terms of efficiency or speed. It can be used, however, to illustrate some of the

difficulties associated with model-based coding. One could ask, for example, what would happen if part of the components of face were occluded due to a rotation of the head? In this case, the assumption of symmetry, that is crucial to the performance of the algorithm, would no longer hold and the result would be an error of large proportions. It is also assumed that the face has no beard or glasses, which can lead to a considerable degradation of performance. Finally, no attempt is made to identify the hair, a task that is obviously very complex.

The problems of how to handled mismatches between the model and the real footage, and how to extract features that can be used to characterize complex patterns (such as human hair, or the leaves of a tree moving with the wind) are central to model-based coding and have not been solved so far.

### 3.2.2 Image synthesis

Once the analysis stage determines the parameters of the model for the current frame, the model can be updated according to the new parameters and a coded version of the input synthesized. The problem of image synthesis has been a subject of research in the field of computer graphics for a number of years, and some of the computer graphics techniques can be used with model-based coding. The two basic problems of the synthesis stage are how to display the 3-D model in a 2-D display, and how to give the wire-frame representation a realistic appearance, i.e. how to synthesize a replica of the textures of the objects modeled.

The problem of displaying a 3-D model on a 2-D display is usually known as *rendering*, and is commonly handled by a *scan-conversion* procedure composed of three steps.

1. Projection of the 3-D wire-frame polygons into the 2-D display plane.

2. Determination of the objects or parts of the objects that are visible.

3. Computation of the intensity or color of each pixel in the synthesized image.

The first step is usually based on the application of simple geometric transformations to all the vertices of the polygons in the wire-frame.

The second step, hidden surface removal, can be solved by a series of different techniques, the simplest of whose consists in the use of a *depth-buffer* with the size of the final image to be displayed. This buffer is initialized with infinity in all its positions and, for each pixel in every polygon, the depth value in the buffer is compared with the depth computed for that pixel. If the pixel's depth is smaller than that in the buffer, the pixel is considered visible, copied to the display frame store, and its depth value copied to the depth-buffer. When all the pixels have been processed, only the visible ones will appear on the synthesized image. This algorithm is very simple, but requires a considerable amount of memory and cannot handle transparent objects. However, since the applications of model-based coding are, as mentioned above, restricted to scenes with few objects and not very large image dimensions, these drawbacks may not be very significant, and, in this case, it provides an efficient solution. More complex solutions to the problem of hidden surface removal can be found in the computer graphics literature [8].

The third step, where the problem of determining textures and surface intensities is handled, is usually solved in the field of computer graphics by the use of shading techniques, based on an illumination model. The simplest shading models assume that the illumination is absorbed by the surface of each object and re-emitted isotropically in all directions. The main problem of such models is that objects which do not receive direct illumination are not visible at all. More complex models try to eliminate this problem by considering the illumination incident in each object as the sum of two terms, one due to direct illumination and the other due to ambient illumination. In this way, it is possible to shade each polygon according to the position of the light sources in the scene.

Shading each polygon separately can lead to luminance discontinuities around

their edges, originating annoying artifacts. A technique known as *Gouraud shading* [12] achieves smooth shading by interpolating the shading intensities associated with adjacent polygons to obtain the shading intensity of the regions near the edge between those polygons. However, and since it does not guarantee the continuity of the derivatives of the intensity across polygon boundaries, Gouraud shaded images still present some visible artifacts in these regions. A more complex interpolation technique, known as *Phong shading* [38], based on an improved illumination model is generally used to reduce this problem.

Shading techniques are very efficient to reproduce objects with smooth and uniform surfaces, but fail when this is not true, as is the case for most natural objects. In these situations, a more efficient solution is to use *texture-mapping techniques* [8], in which the original image is first mapped onto the 3-D model surface, and then projected onto the 2-D display.

## 3.3 Object-oriented image coding

As mentioned in the previous section, one of the major difficulties associated with model-based coding is the determination of a 3-D model that can accurately represent the objects in the scene. Object-oriented algorithms avoid this difficulty by eliminating the requirement of this explicit 3-D model to represent the objects.

The best example of object-oriented coding is the *analysis-synthesis coder (ASC)* developed by Musmann et al. at the University of Hannover [30, 15]. The ASC describes the objects present in the scene by three parameters: shape, motion and texture. The core of the algorithm is an iterative procedure to determine both the shape and the motion of each object, which involves the joint use of analysis and synthesis techniques. First, a global estimate of the motion between the present ($F_k$) and the previous reconstructed frame ($\hat{F}_{k-1}$) is performed. This leads to a set of motion parameters mainly determined by the global motion of the scene (such as the

originated by a change in the camera position). Then, these motion parameters are used to synthesize a motion-compensated ($\tilde{F}_k^0$) version of $F_k$, which will be similar to $F_k$ in the background areas, but a very poor reproduction of $F_k$ in the areas of objects subject to motion different than the global. These areas are identified (by the analysis of the difference between $\tilde{F}_k^0$ and $F_k$) and a binary mask $s_i^0$ (where $i$ is the object index) associated with each of them. These masks contain the information about the shape of each object in the scene.

Once the masks are obtained, the process is repeated to improve the motion estimates. Starting with $F_k$ and $\tilde{F}_k^0$, the motion analysis is performed separately for each of the regions covered by an object. This leads to the set of motion parameters $m_i^1$ associated with each object. A new motion-compensated replica ($\tilde{F}_k^1$) of $F_k$ is then synthesized and the reconstruction error computed. If there are no objects moving in front of each other, the replica will be close to the original. If there are objects partially occluding each other, the analysis of the reconstruction error leads to a decomposition of some of the regions $s_i^0$ into a set of smaller sub-regions, leading to an improved shape representation $s_i^1$. The process is then repeated the number of times necessary for the decomposition of the image into the set of all the objects that compose it. At the end of $j$ iterations, this analysis-synthesis procedure produces, for each object $i$, a set of motion parameters $m_i^j$, a set shape parameters $s_i^j$, and the object texture, which consists on the pixel values of the region $s_i^j$ of $F_k$. This information enables the decoder to reconstruct $\hat{F}_k = \tilde{F}_k^j$.

Motion estimation is based on the assumption of planar 2-D objects moving arbitrarily in 3-D space. In this case, the motion of a given object, covering a specific image area, can be characterized by a set of 8 parameters computed through a procedure presented in [41]. Once the motion parameters are found, they are used for image synthesis as described above and then predictively coded for transmission.

The coding of the shape information is based on an algorithm for contour coding using both polygonal and spline-approximation of the shape [15]. First, the object

silhouette is approximated by a set of polygons characterized by a number of vertices, which depends on the accuracy desired for contour coding. These vertices are then used to compute a spline representation of the contour, and the best of the two approximations is chosen. For transmission, the vertices are differentially coded, using those of the previous frame as prediction. The performance of the algorithm is dependent on the number of vertices allowed: the introduction of more vertices leads to higher accuracy at the cost of increased overhead. Figure 3.10 presents an example of the performance of this algorithm.

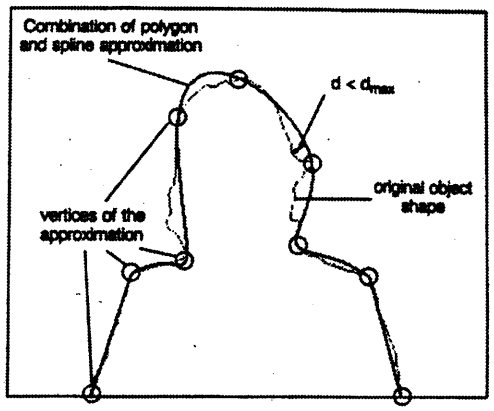


Figure 3.10: Example of contour coding. From [15].

Finally, the texture information is coded using traditional motion-compensated interframe coding, using the synthesized replica of the input frame as prediction. When the coder finds objects which it is unable to synthesize (due to model inadequacies), or which are very small (leading to considerable shape overhead), it switches to conventional block-based processing.

In addition to the algorithm described so far, a slightly different coding approach based on a model of moving 3-D objects is also discussed in [30]. Here, a wire-frame model similar to those described in section 3.2 is progressively built by the encoder using an iterative procedure similar to the presented above. For each object in the scene, motion parameters are first computed and, if the changes between successive frames cannot be accommodated by motion compensation alone, the vertices of the

wire-frame polygons are recomputed, accounting for changes in object shape. This method seems, however, to be of much more complex implementation than the one discussed above, and the authors do not present any coding results achieved with it.

Object-oriented coding fills the gap between the traditional DSP-based techniques and model-based coding. Here, although there is no pre-specified model of the input scene, the encoder tries to build that model, which becomes a better representation of the input than the assumption of a "sinusoidal and translational world" as with the DSP techniques. There are, however, two costs that are inherent to object-oriented coding and which are particularly important for applications where high coding quality is required.

The first is the smaller robustness and general purpose character. Although not as much as with model-based coding, object-oriented coders are still directed to specific applications, involving reduced complexity scenes. The example of a tree moving in the wind is still a case where it is hard to image an object-oriented encoder performing efficiently.

The second is the overhead associated with the transmission of shape parameters, which is not necessary for a DSP-based solution. This overhead is particularly important when several small objects compose the scene. For example, when the camera faces a crowd attending a sports event, all the little heads will require shape information, leading to high overhead. However, the object-oriented segmentation will not provide a large coding gain over a block-based segmentation (characteristic of the DSP solutions), which does not require any segmentation overhead at all. In this example, it is likely that DSP-based coding will lead to higher efficiency.

# Chapter 4

# Library-based image coding

The previous chapters presented several image coding techniques from both the fields of conventional (DSP-based) image compression and the more recent model-based or object-oriented coding approaches. It was already mentioned that the main limitations of conventional image coding techniques are due to the poor model of input in which they are based. The performance obtainable with this limited model is further constrained by the computational complexity associated with block-matching motion estimation, which limits the number of frames usable for prediction. On the other hand, model-based and object-oriented approaches suffer from high complexity and lack of robustness to mismatches between the models in which they are based and the input scenes. These factors have prevented their use for high-quality coding applications, such as the broadcast of digital television.

The purpose of this thesis is to extend the traditional coding model towards an object-oriented framework, while maintaining its robustness and appropriateness for high-quality coding. Since these are conflicting goals, a solution combining aspects from both classes of coding techniques is necessary to achieve them. A library-based encoding scheme provides such a solution.

Theoretically, a library of objects is a very powerful tool for image representation

since typical scenes can be characterized as small sets of objects moving with time. An encoder based on an object library would only need to determine the warp applied to each object from frame to frame and, for a given frame and a given object, transmit the object index in the library and its motion parameters.

In practice, the implementation of such an encoder faces some difficult problems. First, it would require an accurate image analysis stage to identify the objects in the scene which, as seen in the previous chapter, is a very complex task in itself. Then, if it were possible to determine the objects accurately, it would require the capacity to determine a set of parameters which described the motion of each object. This is a relatively easy task in the case of rigid objects, but turns out to be very complex in the case of deformable bodies, such as human figures, and many other natural, non man-made, objects.

As a result of these implementation difficulties, an encoder based solely on a library of objects, and suitable for the encoding of general purpose scenes, does not seem feasible, at least in the near future. The concept of library-based encoding can, however, be merged with the traditional block-based techniques to eliminate some of their limitations and provide increased coding performance.

## 4.1 Vector quantization of the prediction space

Vector quantization (VQ) was discussed in section 2.3 from a data compression perspective. In addition to its optimal compression performance, a vector quantizer can also be seen as a clustering technique, a property which makes VQ particularly suited for the implementation of a library-based encoder.

One of the desirable properties for efficient library encoding is the capability to extend the coding model beyond the set of sinusoidal basis functions characteristic of conventional encoders. A vector quantizer splits its input vector space into a set of vector clusters with common characteristics, and associates to each of these clusters

a representative library entry. If the input vector space is the set of vectors that compose a number of frames in the *prediction space* (frames previously transmitted), vectors with common characteristics will be those belonging to the same or similar objects; and a VQ applied to the prediction space can be seen as a clustering operator grouping together blocks belonging to the same objects. In this way, the underlying coding model is extended into an object-oriented direction.

Clearly, the restriction to block-based processing prevents the existence of a truly object-oriented coding model. It has, however, the property of compatibility with conventional techniques, which is desirable for applications requiring high-coding quality, due to the possibility of compensating for prediction errors (originated from model inaccuracies) by transmitting a block-based prediction error signal. Such a correction signal may not be easily implementable in the case of a purely object-oriented coder since in this case the errors are, in general, associated with geometric distortions which may not be easily recoverable with a block-based correction error signal.

Ideally, the design of a vector quantizer would consist in simply finding the solution to equations 2.54 and 2.55. In practice, and since an input probability distribution function is in general not available, the design of the vector quantizer is based on a training sequence of vectors that are representative of the input vector space. This makes VQ particularly suited for the implementation of a library-based encoding scheme since, by incorporating several past frames in the training set, the library computed for the prediction of a given frame will be representative of an extended prediction space, incorporating several frames, as opposed to the limited prediction space of one or two frames characteristic of the conventional interframe encoders.

In addition to providing the capability to explore an extended prediction space and an enhanced coding model, a vector quantizer of the prediction space has also some characteristics that can lead to a simple implementation of library-based encoding. First, a VQ-based scheme requires very simple decoding (a trivial table look-up operation); a desirable property for high-quality encoders which are typically used

in asymmetric applications (where a single encoder encoder feeds several decoders); where the cost of the decoder is much more relevant than that of the encoder. Second, since the most popular algorithm known for VQ design - the *LBG algorithm* [26] is recursive, it is possible, by using VQ, to achieve a recursive library design procedure, avoiding the computational burden of an entire library re-computation for each new frame.

## 4.2 The LBG algorithm for VQ design

The LBG algorithm for VQ design is an extension of an algorithm for scalar quantizer design first presented by Lloyd [27]. It consists of an iterative procedure that gradually improves the performance of an initial non-optimal quantizer, until the necessary conditions for optimality (equations 2.54 and 2.55) are satisfied. The procedure is very simple, and follows almost intuitively from the structural characterization of a quantizer presented in section 2.1.2 for the scalar case, and extended in section 2.3.1 for the case of vector quantization. It consists in, iteratively, finding the best encoder for the present decoder, and then the best decoder for the new encoder, until a termination criterion is satisfied.

The basic iteration assumes that a probabilist description of the input is available, and uses the optimality conditions to obtain an improved quantizer.

1. Given a codebook $\mathcal{C}_m = \mathbf{y}_1, \ldots, \mathbf{y}_N$, find the cells $\mathcal{R}_i$ that originate the optimal partition of the input vector space given $\mathcal{C}_m$, by applying the nearest neighbor condition of equation 2.54.

2. Using the centroid condition of equation 2.55, find the new optimal codebook $\mathcal{C}_{m+1}$ for this new partition of the input space.

Due to the nature of the optimality conditions, each iteration must improve the codebook, or leave it unchanged, and, in this way, the solution converges to an op-

timum. This optimum can be, however, only local since, as seen in chapter 2, the conditions on which it is based are not sufficient to guarantee global optimality.

In practice, a probabilist description of the input is typically not available and, even if it were possible to obtain such a description, the analytic computation of the centroids on step 2 of the iteration would be extremely complex. To avoid these limitations, practical vector quantizers are designed using a sample distribution based on a set of empirical observations of the input, usually referred to as the *training set* for the quantizer design.

This training set is used to define a random vector whose pmf is obtained by assigning the probability mass of $1/M$ to each vector in the set, where $M$ is the number of vectors that compose it. If the training set is large enough, this pmf converges to a close approximation of the true probability distribution, leading to coding performance close to that obtainable using the true probabilistic description. The use of the training set simplifies, however, the computations associated with the optimality conditions referred above due to its discrete and finite nature.

The main simplification introduced by the use of a training stage is verified in the computation of the partition centroids, which under the MSE distortion measure are given by

$$\mathbf{Y}_j = E[\mathbf{X}|\mathbf{X} \in \mathcal{R}_j]. \tag{4.1}$$

Given a training set $\mathcal{T} = \mathbf{t}_1, \ldots, \mathbf{t}_M$, this expression simplifies into

$$\mathbf{Y}_j = \frac{\sum_{i=1}^{M} \mathbf{t}_i S_j(\mathbf{t}_i)}{\sum_{i=1}^{M} S_j(\mathbf{t}_i)}, \tag{4.2}$$

where the $S_j$ are the selector functions defined in section 2.3.1. I.e. the centroid of a given cell reduces, when the VQ design is based on a training set, to the vector average of the training vectors belonging to that cell.

Given a training set $\mathcal{T} = \mathbf{t}_1, \ldots, \mathbf{t}_M$, the task of determining the cell $\mathcal{R}_i$ associated

with the codeword $\mathbf{y}_j$, for the optimal partition of the input space, is also very simple. It reduces to finding the vectors in $\mathcal{T}$ which satisfy

$$\mathcal{R}_j = \{\mathbf{t} \in \mathcal{T} : ||\mathbf{t} - \mathbf{y}_j|| < ||\mathbf{t} - \mathbf{y}_i||, \forall i\}. \tag{4.3}$$

Using equations 4.3 and 4.2, the basic iteration can be applied to the discrete distribution provided by the training set to obtain an optimum quantizer. This basic iteration is the core of the LBG algorithm, which consists of the following steps.

1. Define an initial codebook $\mathcal{C}_1$.
2. Given $\mathcal{C}_m$, find an improved codebook $\mathcal{C}_{m+1}$ by applying the basic iteration described above.
3. Compute the average distortion for $\mathcal{C}_{m+1}$. Stop if this distortion is small enough, otherwise goto step 1.

The average distortion is the average distortion of representing each vector in the training set by the closest codebook entry. Several criterion can be used for the termination of the algorithm. Ideally, the codebook should be accepted only when the distribution of the training vectors by the partition cells remained exactly the same on two consecutive iterations. In practice this would require, for a reasonably sized training set, a very large number of iterations, and alternative criterion are usually employed. Among these, the most common consists in terminating the algorithm whenever the decrease of the average distortion from one iteration to the next is smaller than a pre-specified threshold.

A flow chart of the LBG algorithm is presented in figure 4.1. Several methods have been reported in the literature for the determination of the initial codebook [26, 6, 11], although none seems to be consistently better than the others. A simple and effective solution for this problem consists in picking up a set of vectors at random from the training set, and using this set as the initial codebook.

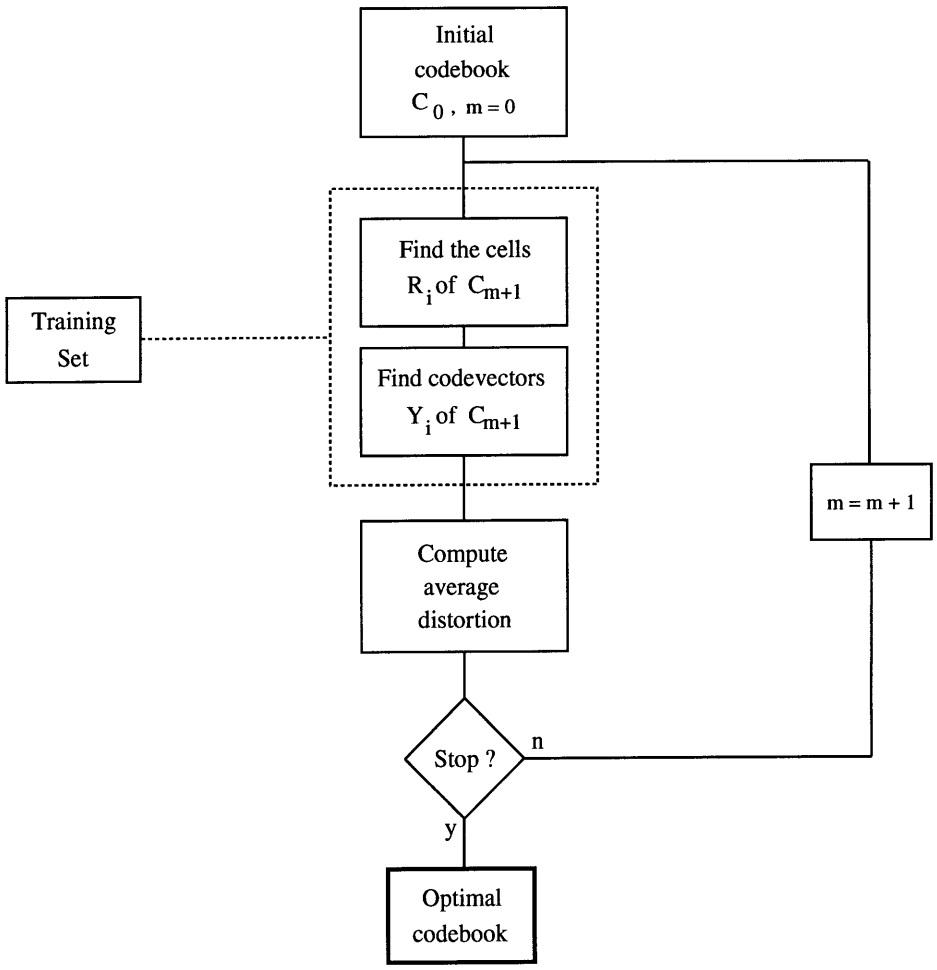


Figure 4.1: Flow chart of the LBG algorithm for VQ design.

## 4.3 Implementation of the library-based encoder

The implementation of an efficient library-based encoder requires the solution of two distinct, but interrelated problems: *library design* and *library updating*. The reason why these two problems are interrelated is that the efficiency of the library updating will be strongly dependent on the temporal properties of the library which, in turn, are determined by characteristics of the algorithm chosen for library design.

Obviously, the efficiency of library update would not be a problem if it were possible to replicate the library design at the decoder. This can, indeed, seem to be a natural procedure since all the information needed for the design is, in the general

case (prediction space exclusively composed by previously transmitted frames), available to the decoder. However, the library design is a computationally heavy task, in particular when a VQ-based solution is considered, and thus incompatible with the requirement for simplicity of implementation of typical decoders already mentioned. Furthermore, a non-causal prediction space (incorporating frames from both the past and the present or future) can lead to increased prediction efficiency over that achievable with the causal one available at the decoder.

Therefore, for a practically implementable and efficient solution, the library design must be performed entirely by the encoder and the efficiency of library update should be considered as an important side-effect of the library design algorithm. In this context, an efficient library design procedure cannot aim uniquely to obtain the best possible library for every frame in the MSE sense. It must also be able to achieve an easily encodable library, whose updating requires the transmission of small overhead information. This requirement for easy encoding can only be satisfied if the library is *temporally smooth*, i.e. its entries do not change significantly from frame to frame unless there is a significant change of the prediction space. A recursive library design algorithm, in addition to making the implementation simpler (since there is no need to recompute the library from scratch for every frame), has the potential to satisfy this smoothness constraint.

### 4.3.1 Library design

In this work, library design was based on the LBG algorithm for the design of vector quantizers presented in section 4.2. The LBG algorithm has several properties desirable for library design:

- it is, by nature, recursive, allowing the use of a previous library as the starting point for the design of the next one and, consequently, progressive library refinement;

- it provides an, at least locally, optimal solution;
- if the previous library is used as the initial codebook and the input space has not changed significantly, the next library will be similar to the previous (temporal smoothness) and will require few additional iterations (reduced complexity);
- it can easily account for an extended prediction space by simply including the vectors from the desired frames in the training set.

Its main drawback is, however, the associated computational complexity. As discussed in section 2.3.4, vector quantization is, itself, a computationally heavy operation and, since each iteration of the LBG algorithm implies the vector quantization of all the vectors in the training set, its complexity increases proportionally to the size of this training set and to the number of iterations required to obtain the final codebook. These parameters have, therefore, to be controlled in a way such that practical complexity limitations are not exceeded.

A large training set is desirable not only due to the inherently larger prediction space, but also because the library will tend to change less from frame to frame and will be, therefore, easier to encode. However, if the training set becomes too large, the weight of recent frames in the final library becomes very small and the library will no longer be able to represent local features, leading to poor prediction efficiency. In practice, the number of frames used for prediction must provide a trade-off between the capability to represent both the long-term and the short-term dependencies between the frame to encode and those in the prediction space.

Theoretically, the higher the number of iterations allowed in the library design, the better the library. In practice, however, the LBG algorithm is characterized by a significant improvement in codebook performance during the first iterations, and then an increasingly slower rate of improvement during the remaining ones. If the initial codebook estimate is relatively close to the final solution, as is the case when the prediction space does not change significantly from one frame to another and the

previous library is used for this estimate, one or two iterations will be sufficient to get very near the optimal solution. In this case, further iterations will not produce a significant increase in the prediction efficiency, but will allow for some change in the library entries, reducing the efficiency off library update. Since additional iterations increase the computational complexity, they bring no real benefit and the maximum number of iterations allowed per frame should be kept small.

The algorithm developed in this work for library design takes these considerations into account in order to maximize the library efficiency, while maintaining the implementation complexity within reasonable bounds. The core of this algorithm, represented in the flow chart of figure 4.2, is the basic LBG iteration, described in section 4.2, which is applied to a training set representative of the prediction space, according to the following steps.

1. Set $f = 0$. Read frame 0.
2. If a scene change has occurred in frame $f - 1$, discard all the vectors in the previous training set.
3. Split frame $f$ into a set of BxB blocks, and incorporate these blocks into the training set.
4. Discard, from the training set, the blocks from frame $f - T$ (if this is a valid frame number, i.e. higher than that of the oldest frame in the training sequence).
5. Apply a maximum of $I$ iterations of the basic LBG iteration, using the library $f - 1$ as initial codebook, to obtain the improved library $f$.
6. Read the next frame, increment $f$, and go to step 2.

The parameters $I$ (number of iterations per frame) and $T$ (number of frames in the prediction space) were determined experimentally, as described in chapter 5, to achieve a compromise between efficiency and implementation complexity. The block dimensions were set at 8x8. These dimensions allowed the use of a relatively small

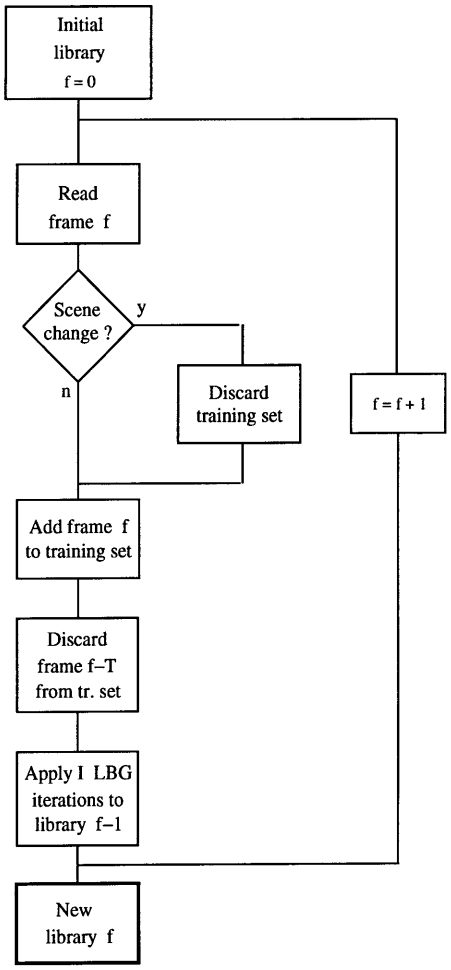


Figure 4.2: Flow chart of the library design algorithm.

library and, consequently, the need for reduced transmission overhead to obtain satisfactory library performance. Figure 4.3 illustrates how the prediction space (frames in the training set) is updated for the encoding of each frame. The inclusion of the frame to be coded in this prediction space (non-causal predictor) allows improved prediction efficiency, at the cost of a small encoding delay.

A very simple scene change detector was employed in step 2 by comparing the number of intra and predictive blocks in the previous frame. Whenever the number of intra blocks is higher, is is assumed that a scene change has occurred. Despite its simplicity, this detector proved to be very reliable. Obviously, with this procedure the scene change can only be detected on the frame following to that in which it really

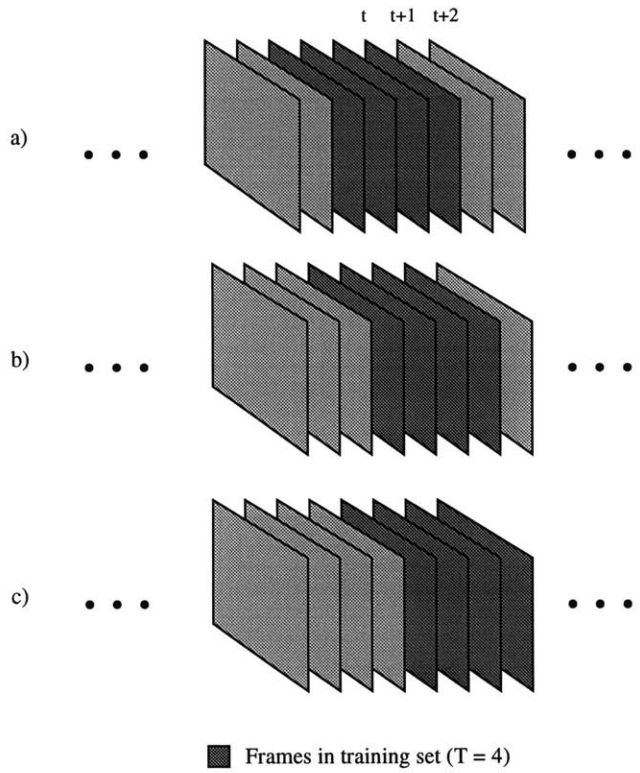


Figure 4.3: Updating of the prediction space. Frames in the training set for the encoding of a) frame $t$, b) frame $t+1$, and c) frame $t+2$.

occurs. This drawback is compensated by the reduced sensitivity of the human eye to distortion in the immediate location of a scene change, the simplicity of implementation of the procedure, and a smoother transition between the libraries suited for each of the scenes which leads to reduced overhead.

Notice that due to the limitation of iterations allowed, the codebook obtained for each frame is not guaranteed to be optimal. This algorithm provides, however, a solution which is recursively improved from frame to frame and, in general, an efficient codebook is obtained a small number of frames after a scene change occurs.

### 4.3.2 Library update

Once the library used in the encoding of a given frame is designed, the new library entries must be transmitted as side-information that will be used by the decoder. If

a new library had to be transmitted for each frame, the cost of the associated side-information would be high, reducing the efficiency of the library-based approach. Two complementary mechanisms were implemented to explore the temporal redundancy of the library, minimizing the overhead associated with library encoding:

- selective transmission,
- conditional replenishment.

### Selective library transmission

Since, during periods of reduced motion, the prediction space does not change significantly from frame to frame, successive libraries will be very similar during these periods. In this case, if the library $L_{t-1}$ is used in the encoding of frame $t$, the coding performance will be similar to the one achievable with the newly computed library $L_t$. However, if $L_{t-1}$ is used, no overhead information has to be transmitted. This property was explored by the introduction of the following selective transmission mechanism in the library-based encoder.

1. Define a distortion threshold ($D$), set $t = 0$, $d_{-1} = \infty$.
2. Read frame $t$, and find the associated library $L_t$ with the algorithm of section 4.3.1. Compute the average distortion $d_t$ resulting of the design procedure (encoding of the vectors in the training set).
3. If $|d_t - d_{t-1}|/d_t \leq D$, use $L_{t-1}$ for the encoding of frame $t$. Set $d_t = d_{t-1}$, $t = t + 1$, and go to step 2.
4. Use $L_t$ for the encoding of frame $t$, and transmit $L_t$ to the decoder. Set $t = t+1$, and go to step 2.

In this way, random fluctuations of the library, due to small variations of the prediction space, are eliminated and the overhead minimized. It can be argued that the

equation in step 3 can lead to incorrect decisions since the distortions involved are relative to the design of different libraries, based on different training sets. Since the training sets are different it would be, in theory, possible that two significantly different codebooks would lead to similar distortions. In practice, due to the overlapping of the training sets used for the design of the libraries associated with consecutive frames (if the number of frames in the prediction space is greater than one) and the reduced number of iterations allowed for library design, this situation is very unlikely if the distortion threshold is maintained within reasonable bounds. This problem was never found in several experiments performed using selective library transmission.

### Conditional library replenishment

In typical image sequences, even during periods of significant motion, where new objects are revealed and some of the old occluded, a reasonable area of the scene remains relatively unaltered from frame to frame. This will obviously have a similar effect on the library entries since the entries associated with objects that remain unaltered between consecutive frames will tend to be relatively stable. Therefore, even when a library is selected for transmission by the previous mechanism, it is possible to explore the information provided by previously transmitted libraries to minimize the transmission overhead. A technique particularly suited for this purpose is *conditional replenishment*.

Conditional replenishment [29] was introduced in the early days of video coding to explore the temporal redundancy existent between successive frames of an image sequence. It is based on the very simple idea that areas of the scene which do not change from frame to frame should not be transmitted, and only the areas of motion should be updated. This idea can be easily adapted to the problem of library update by transmitting only the entries which change between consecutive libraries. However, in the context of the present work, this method would not be efficient due to the nature of the LBG algorithm used for library design.

As discussed in section 4.2, each codebook entry is the average of the training vectors that fall, during the training stage, in the partition cell associated with it. Thus, even in the case of moderately sized training sets, the transition of a training vector from one cell to another can lead to a noticeable change in the codebook entries associated with the partition of the input space. Since, in the current implementation, for each frame to encode a new frame is added to training set and one frame removed from it, it is likely that, even in the absence of significant motion, all the library entries will suffer from an at least slight disturbance. This disturbance can be due to factors such as varying illumination, small displacements in the camera position, noise, etc.

A straightforward implementation of conditional replenishment will, therefore, not be efficient for library update. It is, however, possible to improve the update efficiency, with a variation of conditional replenishment, such as the implemented by the following algorithm.

1. For each entry in the current library, find the best match in the previously transmitted one.

2. Subtract the current entry from this best match. If the energy of the remaining residual signal is smaller than that of the entry, transmit the residual and the index of the best match. Otherwise, transmit the new entry.

In this way, most of the entries will require the transmission of a scalar index and a low-energy, easy to encode, residual signal, and only the entries that are really different from the ones already existing in the previous library will originate a noticeable transmission overhead.

## 4.4 The library-based image encoder

The algorithms discussed in section 4.3 were used for the implementation of the library based-encoder represented in the block diagram of figure 4.4.

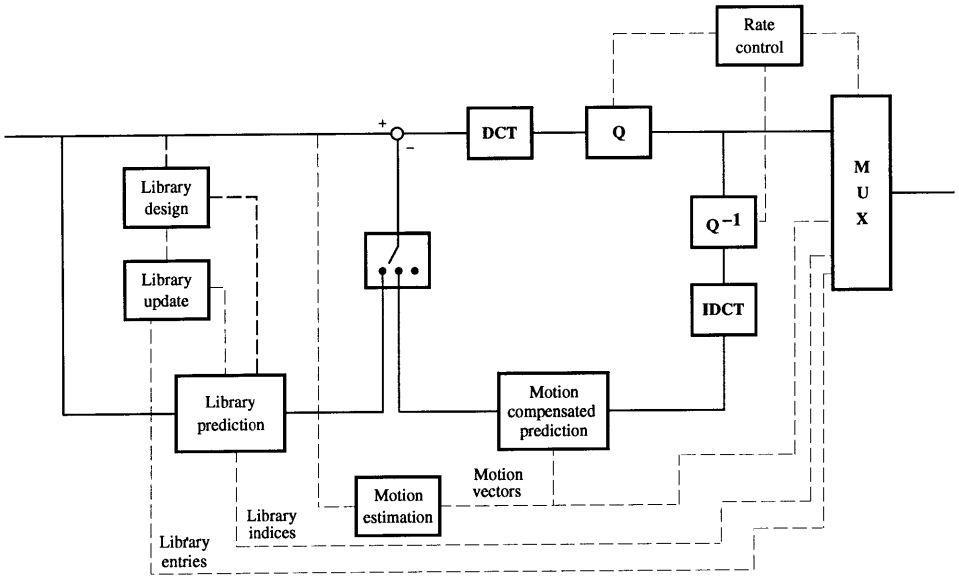


Figure 4.4: Block diagram of the library-based coder.

Each input frame is segmented into a set of 8x8 blocks, which are processed to minimize both temporal and spatial correlation. Two different prediction structures are used for temporal processing: the library-based predictor (using vector quantization of the prediction space) discussed above, and a conventional motion-compensated predictor.

The conventional motion-compensated predictor was introduced to improve the quality of the prediction in areas of purely translational motion and motion of reduced amplitude. In these areas, motion-compensated prediction based on block-matching motion estimation has two advantages over library-based prediction.

- First, it leads to approximately constant prediction quality, independently of the objects contained in the area. This is a consequence of the fact that motion-compensated prediction simply ignores the scene content. Whenever the as-

sumption of purely translational motion is verified, the prediction error signal will be very close to zero, independently of the shape or texture of the objects in the area being motion compensated. This does not happen with library-based prediction because, in this case, each library entry is intended to represent an object (or at least a part of it) in the scene. Since quantization is required to compromise this goal with that of achieving a small (easily implementable and encodable) representative library, the efficiency of the predictor will depend on factors such as textures and object shape and will, in general, be inferior to that obtained with motion-compensation.

- Second, it produces smooth vector fields, which are easier to code than library indices. Areas covered by large objects under translational motion or where no motion occurs (such as still backgrounds) lead to vector fields where neighboring vectors have similar amplitudes, and can therefore be coded very easily with predictive techniques. This is not guaranteed for the library-based predictor since, in this case, due to the unstructured nature of vector quantizers, two similar library entries can have completely different library indexes.

By using the two prediction modes, it is possible to combine the higher efficiency of motion-compensated prediction in areas of translational or reduced amplitude motion, with the increased performance of library prediction in areas of non-translation motion, object occlusion, or where new objects are revealed.

The encoding of the prediction error signal is similar to that used by MPEG-2, and described in section 3.1.1. The DCT is used for temporal decorrelation, and the DCT coefficients are scalar quantized and entropy coded. The rate-control algorithm described in the MPEG-2 Test Model [18] is employed to guarantee a constant bit rate bitstream. Temporally, the MPEG GOP structure is implemented, but only I and P-frames are used. Most of the benefits of interpolated prediction, such as increased prediction efficiency in areas of revealed objects, pans, etc., are also available with library prediction and a non-causal prediction space; enabling the elimination of B-

frames and, consequently, significant savings in terms of implementation complexity and encoding delay. In P-frames, those blocks which cannot be efficiently predicted by any of the two prediction methods discussed above are coded as intrablocks.

For each frame, a new library is computed with the design algorithm of section 4.3.1, and updated by the application of the selective transmission and conditional replenishment mechanisms of section 4.3.2. Before transmission, the new library entries and prediction error residuals resulting from conditional replenishment are coded as regular image blocks, using the DCT, scalar quantization, and variable-length coding. Care is taken, however, to assure that the quantizer step-size is, for this blocks, smaller than that applied to normal blocks, in order to avoid degradation of the prediction quality.

It was already mentioned that the transmission of the library indexes associated with the prediction of library encoded blocks is more expensive than that of the motion vectors associated with the prediction of motion-compensated blocks. The reasons for this are the unstructured nature of vector quantizers and the smaller size of the blocks coded in the library mode. While motion compensation can be efficiently performed at the macroblock (16x16 pixels) level, this is not possible with library-based prediction since, in this case, a large library, too expensive in terms of both computational and transmission overhead, would be required.

The requirement for a manageable library, capable of providing acceptable coding efficiency, limits the dimensions of the blocks used for prediction to 8x8. This originates an increase of the overhead associated with the transmission of library indexes as compared with that associated with the transmission of motion vectors. This increase is reinforced by the unstructured nature of the vector quantizer used for the implementation of the library, which prevents the use of differential predictive techniques for the encoding of library indexes.

To minimize the increase in overhead due to the unstructured nature of the VQ, the library entries are ordered according to their mean after library design. This guar-

antees that neighboring image blocks are predicted by entries with similar indexes, making it possible to use a predictive technique (where the previous index is used to predict the current one) to encode more efficiently these indexes. The overhead associated with the library transmission is also reduced by this new codebook ordering since it is now possible, for the entries transmitted in the replenishment mode, to efficiently predict the current library index from that of the best match in the previous library. The conditional replenishment mode is illustrated by figure 4.5, where $L_t$ and $L_{t+1}$ are the previously transmitted and current libraries.

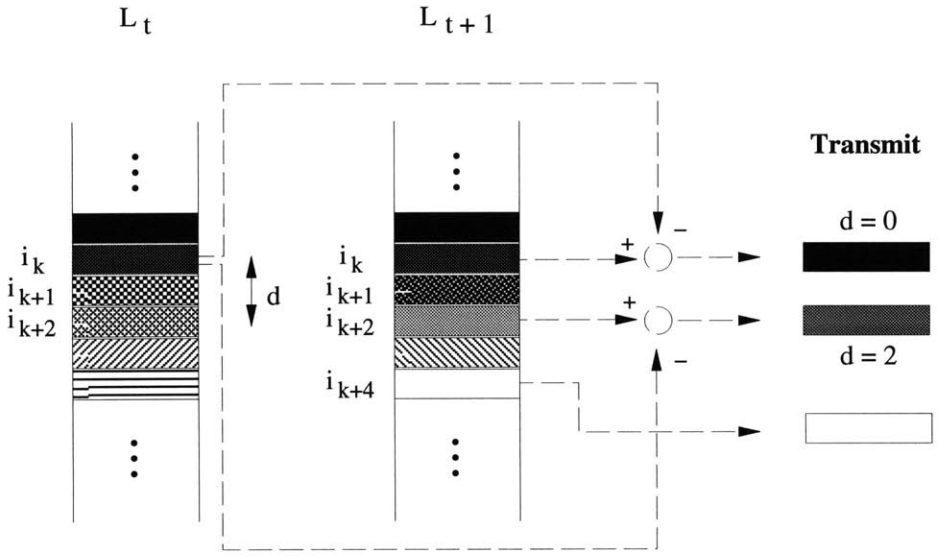


Figure 4.5: Conditional replenishment mode for the transmission of library entries. If the prediction residual has smaller energy than the codeword ($i_k$, $i_{k+2}$), it is transmitted together with $d$. Otherwise ($i_{k+4}$), the codeword is transmitted.

In this example, the $k^{th}$ library entry remains unaltered between $t$ and $t+1$. The $k^{th}$ entry of $L_t$ is the best match for both the entries $k+1$ and $k+2$ in $L_{t+1}$. The replenishment is accomplished by the transmission of either the distance to the best match ($d$) and a low-energy prediction error signal or the new library entry itself (if a good match cannot be found). Due to the codebook ordering, the probability distribution of $d$ is highly peaked at zero and an additional saving in overhead is achieved by employing entropy coding.

# Chapter 5

# Simulation results

This chapter presents the results of the simulations performed to analyze the performance of the library-based encoder and the gains introduced by library prediction. Two sets of experiments were performed: the first, to determine the values of the encoding parameters, referred in section 4.3, for which the coding efficiency is maximized; and, the second, to compare the performance of the library encoder with that of a standard MPEG-2 encoder.

All the experiments were performed on a sequence of 100 frames from the movie "Sharky's Machine", containing two scenes with significantly different characteristics. The first scene consists of a political rally with the camera facing the crowd, and has significantly high temporal and spatial activity. In the second scene, the camera faces a small number of subjects in front of a smooth background. An hat is waved in front of the camera leading to the occlusion of significant parts of the image. This second sequence also has significant temporal activity, but has reduced spatial activity.

The quality of the encoded sequences was measured both subjectively and objectively. The quality measure chosen for objective quality evaluation was the *Signal to*

*Noise Ratio (SNR)* defined by:

$$SNR = 10 * \log_{10}(\frac{255^2}{MSE})\, dB, \tag{5.1}$$

where $MSE$ is the mean square error (equation 2.6) defined by:

$$MSE = \frac{1}{M} \sum_{ij \in \mathcal{R}} (x_{ij} - \hat{x}_{ij})^2, \tag{5.2}$$

where $M$ is the number of pixels in the image, $ij$ the pixel coordinates, $\mathcal{R}$ the image region of support, $x$ the original, and $\hat{x}$ the reconstructed image.

Section 5.1 presents the results of the simulations performed to determine the optimal values of the library encoder parameters. Section 5.2 compares the efficiency of library-based prediction to that of the MPEG motion-compensated prediction. Finally, section 5.3 analyses the global performance of the library-based encoder, and compares it with that of the standard MPEG-2 encoder.

## 5.1 Optimization of encoder parameters

It was mentioned in section 4.3 that the performance of the library-based encoder is affected by the values of three parameters: the number of frames ($T$) in the training set, the maximum number of iterations allowed per frame for library design ($I$), and the distortion threshold applied by the mechanism of selective library transmission ($D$). It was also mentioned that the causality or non-causality of the prediction space might affect significantly the coding performance. This section presents the results of the simulation experiments performed to analyze the influence of each of these factors in the overall coding efficiency and to determine their optimal values.

### 5.1.1 Composition of the prediction space

Two sets of experiments were performed to determine the optimal composition of the prediction space. In the first experiment, the prediction space was restricted to be causal (composed exclusively of previously transmitted frames), and the number of frames composing it ($T$) varied from one (the previous frame) to seven. In the second experiment, the restriction of causality was eliminated by introducing the frame to be coded in the prediction space. The goal was to increase the prediction efficiency without increasing the coding delay significantly (as would happen if further frames from the future were included in the training sequence).

Figure 5.1 presents a comparison between the prediction efficiency achieved with the two types of prediction space as well as the influence of the parameter $T$ in this efficiency.

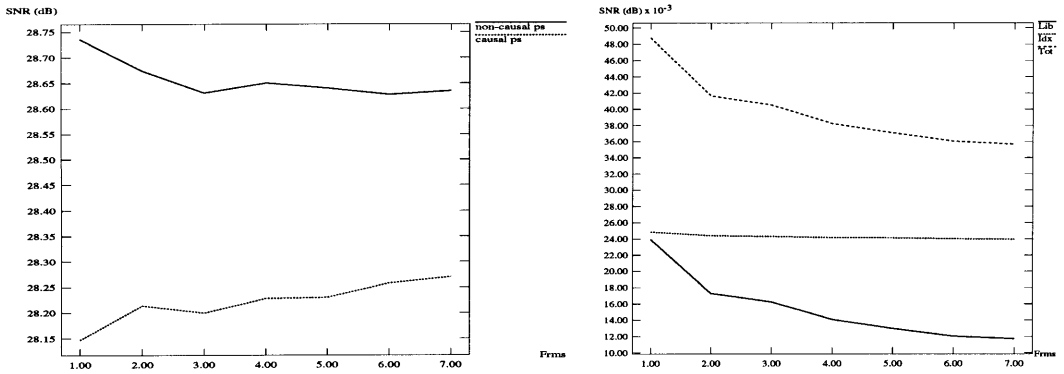


Figure 5.1: Left: Influence of the causality of the prediction space on the prediction efficiency. SNR curves for the prediction signal as a function of the number of frames in the training set ($T$). Right: Overhead in bits per pixel associated with library prediction as a function $T$, for the non-causal predictor.

The performance achieved with the non-causal predictor is consistently better than the achieved by the causal predictor, leading to a gain of about 0.45 dB. This is explained by the fact that, by including the frame to predict in the training set, the non-causal predictor has more information about this frame.

As expected, the dependence of the prediction efficiency on $T$ is different for the

two predictors. For a causal predictor, the prediction improves when more frames are included because this leads to the inclusion, in the prediction space, of more information about objects that were temporarily occluded, or which are under periodic types of motion. On the other hand, for the non-causal predictor, a larger $T$ leads to a decrease in efficiency. This happens because the inclusion of more frames does not introduce any information useful for prediction other than that already contained in the frame to predict, and the library becomes representative of a larger number of frames loosing some of its capability to efficiently reproduce local input characteristics.

Figure 5.1 also presents the dependency of the overhead associated with library encoding on the number of frames in the prediction space. Three curves are represented, all for the case of the non-causal predictor: the overhead associated with the library transmission (*Lib*), the overhead associated with the transmission of the library indexes used for prediction of the blocks coded in the library mode (*Idx*), and the total of these two types of overhead (*Tot*). As expected, the overhead of transmitting the library is significantly reduced for a large $T$ due to the increased temporal smoothness of the library originated by the larger overlap between consecutive training sets. Similar results were observed for the causal predictor.

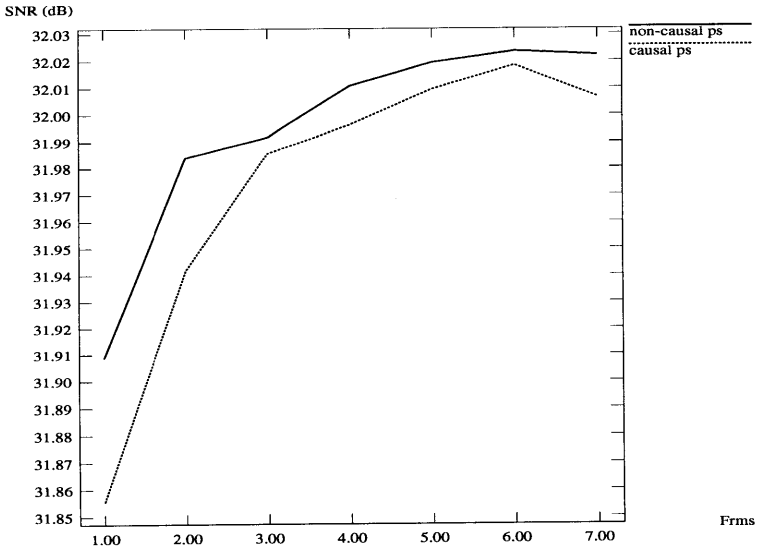


Figure 5.2: SNR of the sequences coded with causal and non-causal prediction as a function of $T$.

Figure 5.2 presents a comparison of the overall coding efficiency achieved with the two predictor types and its dependence on $T$. As anticipated by figure 5.1, the non-causal predictor performs consistently better than the causal one. It is, however, interesting to observe that the overall coding efficiency increases with $T$ in both cases, demonstrating the importance, for the final coding quality, of the overhead reduction due to the increase in library smoothness achieved with large values of $T$. From the results presented by this curve, and since a large prediction space increases the complexity of the encoder, it was decided to use the non-causal predictor with $T = 3$ for the remaining experiments.

### 5.1.2 Distortion threshold for library update

A set of experiments was performed to analyze the dependency of the coding efficiency on the distortion threshold ($D$) of the selective transmission mechanism described in section 4.3.2.

Figure 5.3 shows the dependence on $D$ of both the prediction efficiency and the overhead associated with the library. The prediction efficiency remains approximately the same for values of $D$ below 16%, decreases rapidly for values between 16% and 20%, and then remains approximately unaltered. The library overhead decreases with the increase of $D$ because, for larger values of $D$, the library is transmitted a smaller number of times.

Figure 5.4 illustrates the dependency of the overall coding efficiency on the distortion threshold. As expected from the curves of figure 5.3, the overall efficiency tends to increase with $D$ (due to the decrease in overhead) until the point where the significant decrease in the prediction efficiency leads to degradation of the overall coding performance. Given the results presented by this figure, the value of $D = 12\%$ was chosen, and used in the remaining experiments.

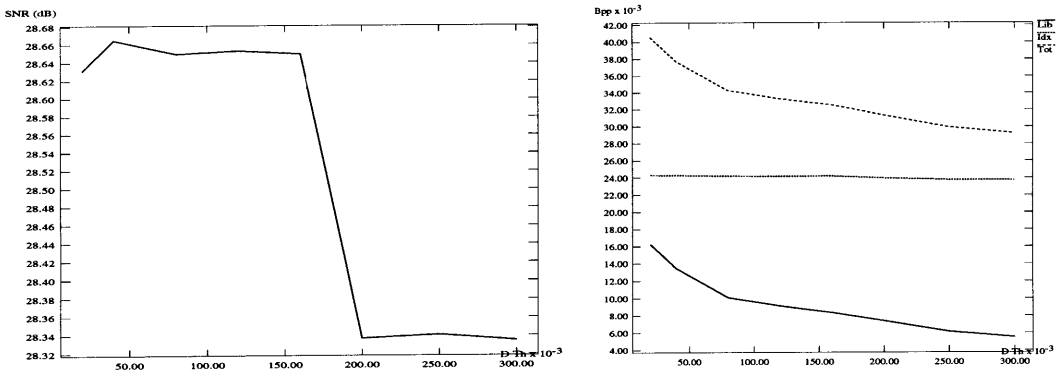


Figure 5.3: Left: SNR of the prediction signal as a function of the distortion threshold for selective codebook transmission ($D$). Right: Overhead associated with library prediction as a function of $D$.

### 5.1.3 Number of iterations of library design

A set of experiments was performed to analyze the dependency of the coding efficiency on the maximum number of iterations ($I$) allowed for library design described in section 4.3.2.

It can be seen from figure 5.5 that both the prediction efficiency and the overhead associated with the library transmission increase with the number of iterations allowed in the library design. This behavior makes sense since a higher number of iterations will lead to a better library (closer to the optimal for each frame), but will also allow for greater variations between successive libraries, originating reduced temporal smoothness and increased overhead.

Figure 5.6 demonstrates that the gain achieved by the improved prediction efficiency is not enough to compensate for the loss in encoding quality due to the increased overhead. It can be concluded from this figure that, in general, the coding efficiency decreases with the increase of the number of iterations. Thus, the maximum number of iterations was limited to $I = 1$ in the implementation of the library-based encoder. This is a fortunate result since the complexity of the encoder grows proportionally to the number of iterations that it must perform and, due to this result, it was possible to limit the complexity without any sacrifice in coding quality.

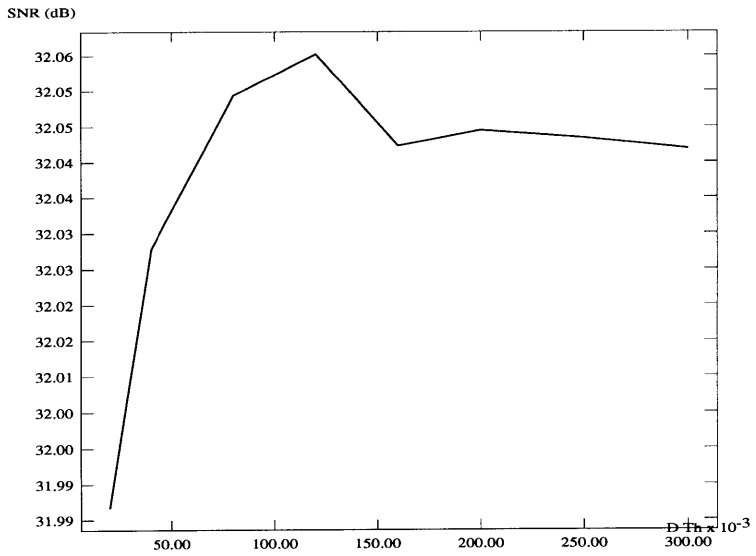


Figure 5.4: SNR of the coded sequence as a function of $D$.

## 5.2 Efficiency of the library predictor

According to the results of the experiments described in the previous section, the values presented in table 5.1 were chosen for the parameters of the library-based encoder.

To evaluate the efficiency of library prediction, the library-based encoder of figure 4.4, a standard MPEG-2 encoder, and a library-based encoder without motion

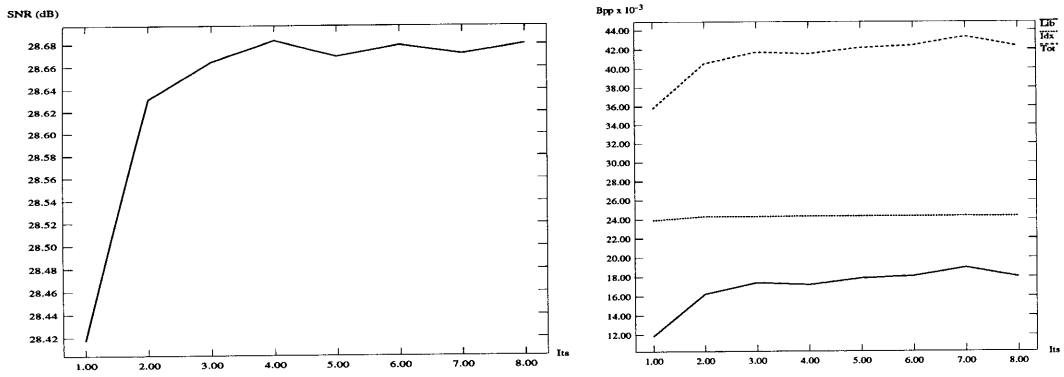


Figure 5.5: Left: SNR of the prediction signal as a function of the number of iterations allowed for library design ($I$). Right: Overhead associated with library prediction as a function of $I$.

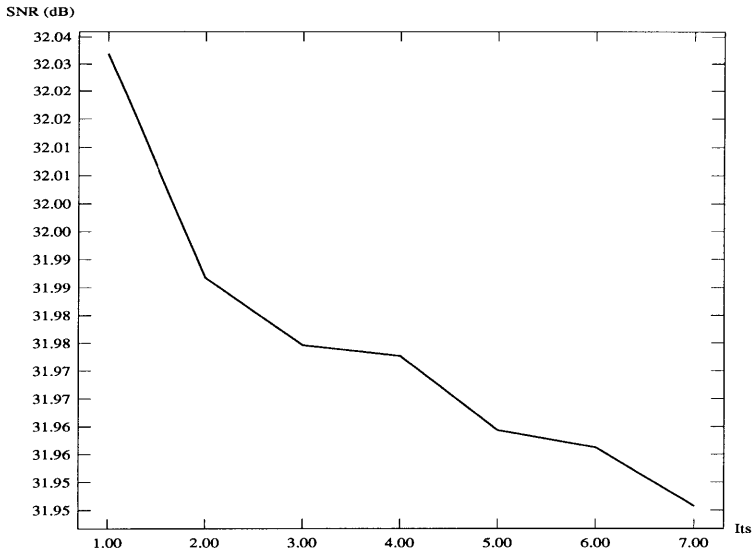


Figure 5.6: SNR of the coded sequence as a function of $I$.

Table 5.1: Optimal values for the parameters of the library-based encoder.

| Parameter | Value |
|---|---|
| T | 3 |
| D | 0.12 |
| I | 1 |

compensation were used to encode the test sequence. Figure 5.7 presents a comparison between the quality of the predictions obtained with the library alone ($Lib$), motion compensation alone ($MC$), and the library-based encoder ($Lib+MC$) using both library and motion-compensated prediction.

It is clear from figure 5.7 that the library-based encoder incorporating both prediction structures is considerably more efficient than any of the encoders based uniquely on one of the predictors. The behavior of the SNR curves, when library and motion-compensated prediction are used alone, can be explained by the different characteristics of the two scenes contained in the sequence.

Figure 5.8 presents both the prediction and prediction-error signals for frame 52

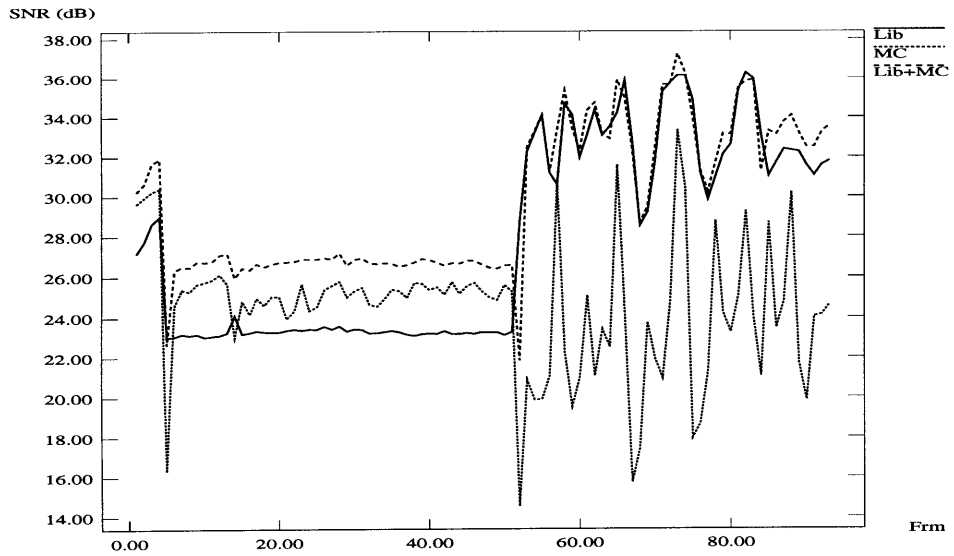


Figure 5.7: Comparison of the prediction efficiency obtained with a library predictor (*Lib*), a MPEG motion-compensated predictor (*MC*), and both (*Lib+MC*).

of the first scene (frames 0 to 54) obtained with the three prediction structures. As referred in the beginning of this chapter, this scene contains both high temporal and high spatial-activity. Given the high spatial-information content, and since the library is constrained (by implementation and efficiency reasons) to a relatively small size (256 entries), the prediction obtained with the library alone is very blocky and a lot of the detail is lost, leading to an overall reduced quality. However, this low quality is uniform, i.e. all the objects that compose the picture are equally degraded.

On the other hand, the quality of the motion-compensated prediction is very non-uniform. In the areas where there is no motion, the motion is small (such as the background) or the motion is translational, the MC predictor works very well and the prediction has high quality. However, when these assumptions are not verified (due to newly revealed objects, or complex, non-translational, motion), the prediction becomes very poor. Notice, for example, the severe block degradation in the top-right and top-left regions of the crowd, where hats are waved in a non-translational way.

The joint use of library and motion-compensated prediction leads to a much better quality than that obtained by any of these techniques alone. In this case, motion

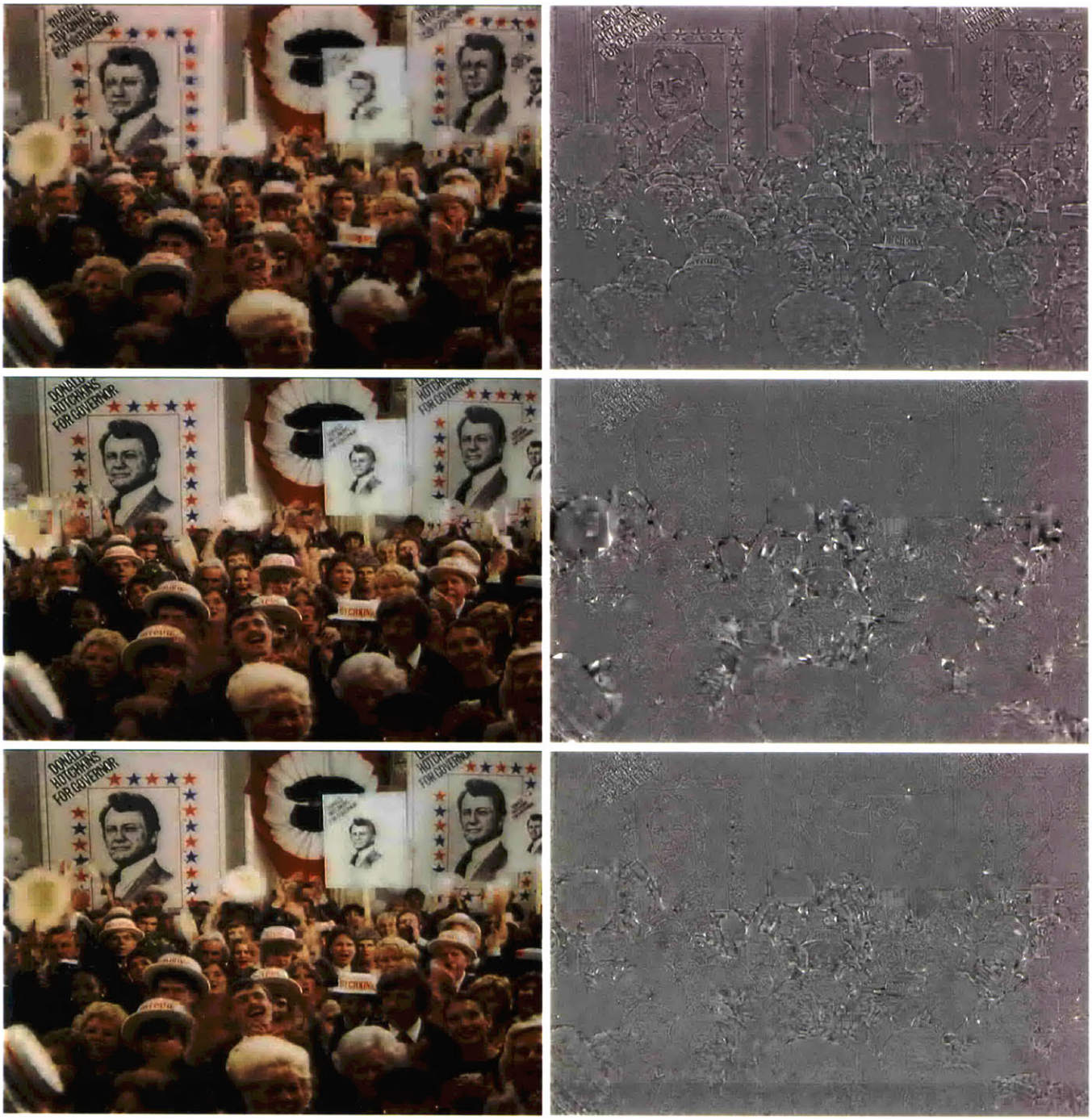


Figure 5.8: Comparison of the prediction efficiency achieved with the three different prediction structures. Scene with high-spatial activity. Top: Library prediction alone. Middle: Motion-compensated (MC) prediction alone. Bottom: Library and MC prediction. Left: Prediction. Right: Prediction error.

compensation is, in general, chosen for the prediction of blocks without motion or under translational motion (like the background), and the library prediction handles the more complex cases of revealed objects and non-translational motion.

Figure 5.9 presents the prediction and prediction error for frame 188 of the second scene (frames 55 to 100) obtained with the three prediction structures. In this case, since the number of objects in the scene is much smaller, the library performs much better. Notice that since these images are to be used for prediction, the blockyness is not very important if the prediction has high SNR. This is what happens with the library predictor, as can be observed from figure 5.7.

Since this scene has a high degree of object occlusions and objects being revealed, the performance of the motion-compensated predictor becomes much worst than that achieved in the previous case. Notice the very bad prediction in the area revealed by the motion of the hat and, in particular, the inability of the predictor to reproduce the microphone which was occluded in the previous frame. So, when MC is combined with the library, the prediction performance of the adaptive mode is similar to that obtained with the library alone (figure 5.7).

The higher efficiency of the MC mode in the first scene and of the library mode in the second can also be observed from figure 5.10, which displays the percentage of the total number of blocks predicted using each of the modes when these are combined into the adaptive mode. While the percentage of motion-compensated blocks is higher during the first scene than during the second, the percentage of library-predicted blocks increases in the second scene.

As was already referred in section 4.4, the better performance of the motion-compensated predictor in the first scene and of the library prediction in the second scene is due to the difference on the model behind the two approaches. Due to the great number of objects in the first scene, the limited size of the library prevents it from providing a good representation for all those objects. On the other hand, the small number of objects in the second scene can be satisfactorily represented, even

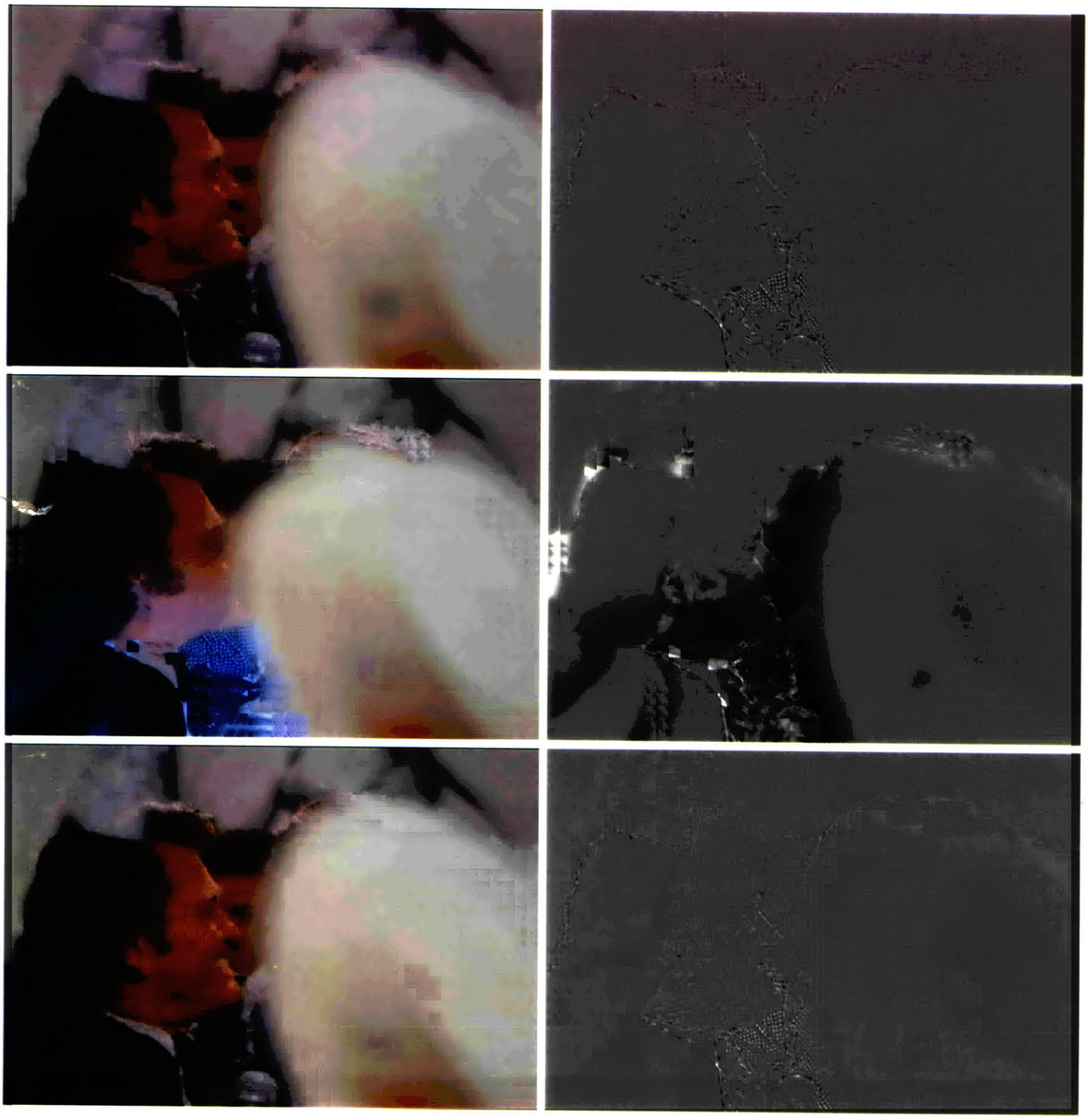

Figure 5.9: Comparison of the prediction efficiency achieved with the three different prediction structures. Scene with low-spatial activity. Top: Library prediction alone. Middle: Motion-compensated prediction alone. Bottom: Library and MC prediction. Left: Prediction. Right: Prediction error.

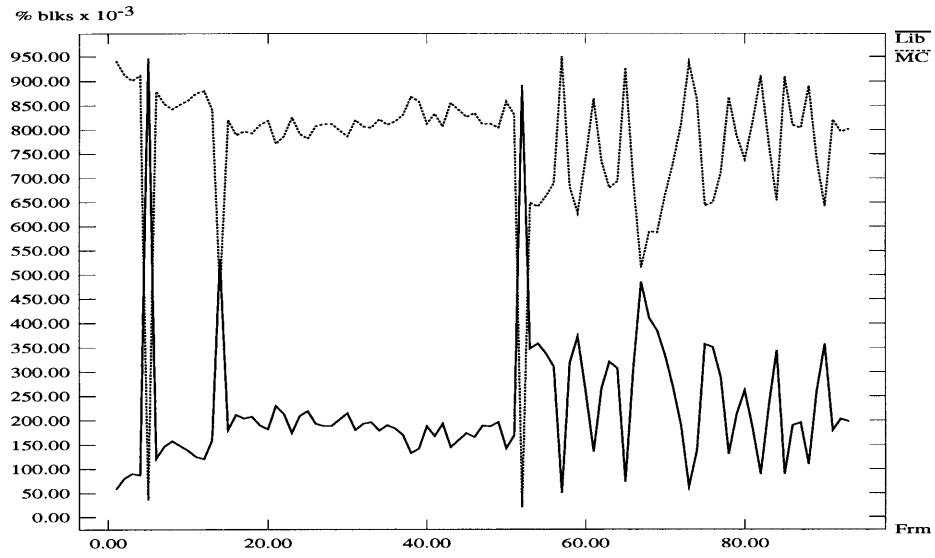


Figure 5.10: Comparison of the percentage of blocks library-predicted and the percentage of blocks motion-compensated, when the adaptive mode is used.

with a small library.

In the case of the motion-compensated predictor, the spatial content of the scene does not affect the prediction efficiency because the model of the input on which it is based makes no attempt to consider the objects in the scene. The performance is, however, significantly affected by the existence of objects being occluded or revealed or subject to non-translational motion due to the assumptions on which the motion-compensated predictor is based and the reduced prediction space available.

The peaks in figure 5.10 are due to the scene changes or other portions of the sequence where there is significant change between successive frames. These peaks illustrate the complete incapacity of the motion-compensated predictor to handle such type of events, as opposed to the library predictor for which no degradation in performance is visible when they occur (figure 5.7). This characteristic of the library predictor is a consequence of the non-causal prediction space and the scene-change mechanism incorporated in the library-design algorithm of section 4.3.1.

Figure 5.11 illustrates the effect in the prediction efficiency of the scene change

Figure 5.11: Comparison of the prediction efficiency during a scene change. Top-left: Library prediction alone. Top-right: Motion-compensated prediction alone. Bottom-left: Original image. Bottom-right: Library+MC prediction.

occurring in frame 55. Notice the completely useless prediction provided by the motion-compensated predictor, as opposed to the acceptable quality of that provided by the library.

The capacity of the library predictor to adapt rapidly to large variations in the input sequence (such as when scene changes occur) is demonstrated by figure 5.12, where successive libraries are presented alongside the respective input images, in the vicinity of a scene change. Notice that, in this example, the library does not change significantly before the scene change (temporally smoothness), and only two frames are required for the library to stabilize after the scene change (fast convergence of the library design algorithm).

## 5.3 Global efficiency of the library-based encoder

It was shown, in the previous section, that the joint use of a library predictor and a motion-compensated predictor leads to a significant increase in prediction efficiency over that achieved by any of the two techniques alone. However, as seen in section 4.4, the introduction of library prediction also originates increased overhead, and so, although it results in a considerable improvement of prediction quality, it may not lead to a significant improvement of the overall coding quality. To evaluate the effect of the introduction of library prediction in the overall coding quality, the library-based encoder of figure 4.4, and a standard MPEG-2 encoder were used to encode the test sequence. Since the library mode is responsible for the largest portion of the overhead and the prediction obtained with the library alone is worst than that obtained with the adaptive structure, the encoder without motion-compensation was not considered in this experiment.

Two experiments were performed to compare the efficiency of the encoders. In the first experiment, aimed to compare the overall performance improvement due to the better prediction achieved with library prediction, the intra mode was not allowed in

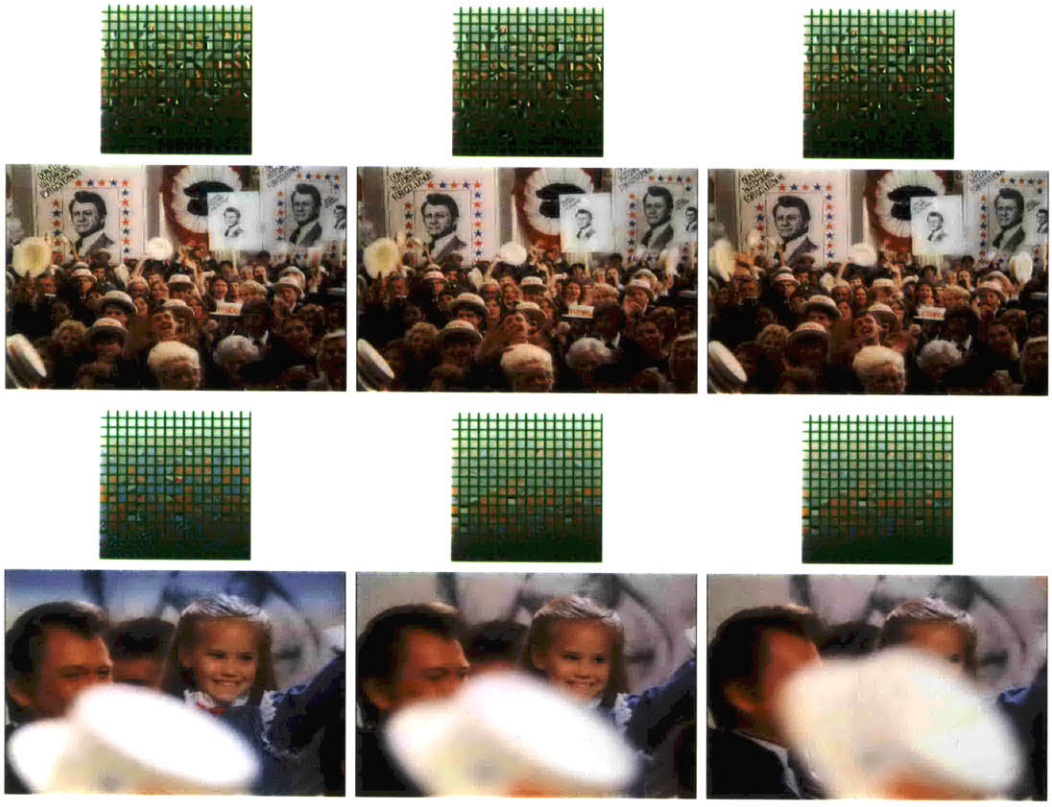

Figure 5.12: Library evolution in the vicinity of a scene change. Top to bottom, left to right: successive input frames (shown with quarter resolution) and associated libraries.

predicted pictures. In the second experiment, this restriction was eliminated to allow a global comparison between the performance of the standard MPEG-2 encoder and that of the library-based encoder of figure 4.4.

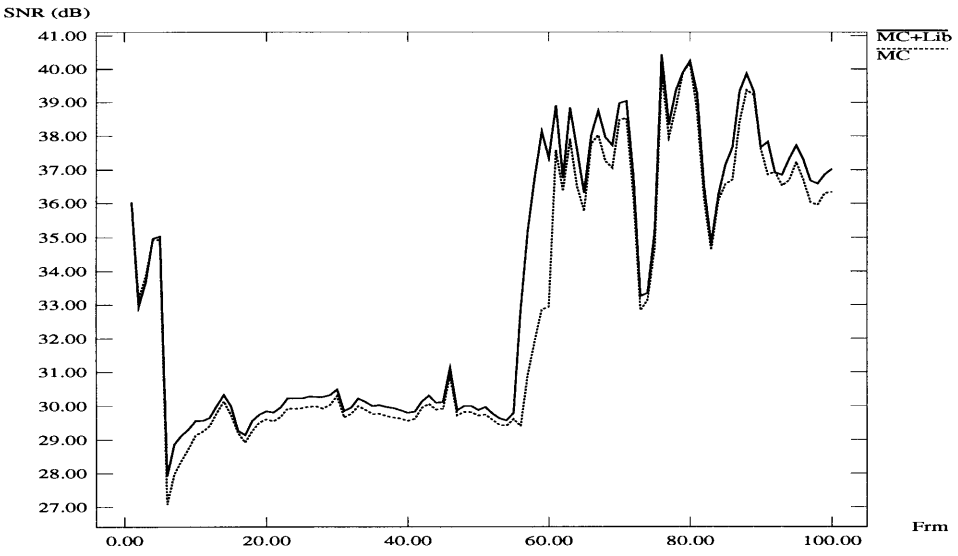


Figure 5.13: Comparison of the coding quality achieved with (*MC+Lib*) and without (*MC*) library prediction, when the intra mode is not allowed in predicted frames.

Figure 5.13 presents the SNRs obtained with (*MC+Lib*) and without (*MC*) the library predictor when the intra mode is not allowed. It can be seen from the figure that the performance of the library-based coder is always better. Notice in particular the much faster response to the scene change.

Table 5.2 presents the values of the average and peak gains achieved with the library-based encoder for the first scene, the second scene, and the scene change. The overall gain provided by the library-based encoder is of approximately 0.56 dB.

Figure 5.14 presents pictures from both scenes of the test sequence, coded at 4 Mbps with and without the library predictor and no intra-macroblocks allowed in P-frames. It can be observed from this figure that the use of library prediction also leads to superior subjective quality.

Figure 5.13 presents the SNRs obtained with (*MC+Lib*) and without (*MC*) the


Figure 5.14: Comparison of the coding efficiency when intra-macroblocks are not allowed in predicted frames. Left: Encoder without library prediction. Right: Encoder with library prediction. The pictures in the bottom occur 4 frames after the scene change.

Table 5.2: SNR gains achieved by the library-based encoder at 4 Mbps.

| Scene | Average gain | Peak gain |
|---|---|---|
| 1 | + 0.27 dB | + 0.89 dB |
| 2 | + 0.52 dB | + 1.32 dB |
| SC | + 3.73 dB | + 5.27 dB |

library predictor when the intra mode is allowed.

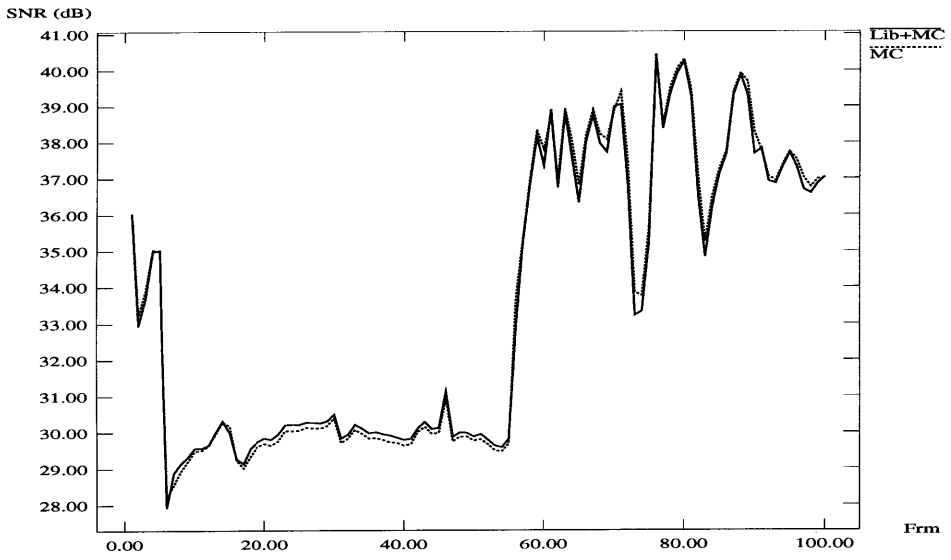


Figure 5.15: Comparison of the coding quality achieved with (*MC+Lib*) and without (*MC*) library prediction, when the intra mode is allowed.

In this case, the gain obtained with the library predictor is much smaller, in particular during the second scene. This is due to the low spatial-information content of this scene that allows the encoder using only motion compensation to recover from the inefficient prediction by introducing several intra-blocks in each predicted frame. Since the spatial activity is low, the intra mode is efficient in filling the gap previously existent between the library-based encoder and the purely motion-compensated encoder. This does not happen for the first scene because the high-spatial information content of this scene makes the intra macroblocks very expensive.

# Chapter 6

# Conclusions and future work

## 6.1 Conclusions

The main conclusion taken from the work developed in this thesis is that it is possible to improve the quality of the prediction achieved with current motion compensation techniques used for interframe coding. This thesis presents a new framework, library-based encoding, to achieve this goal.

The main advantages of a library-based scheme are its capability to exploit a more elaborate model of the input (not simply based on the assumption of 2-D translational motion) and an extended prediction space (including more than just one or two frames). These characteristics lead to much higher performance in regions of newly revealed objects or objects subject to non-translational motion, where the motion compensated-predictor fails completely.

With the particular implementation of library-based encoding presented in this work, vector quantization of the prediction space, the coding model is extended into an object-oriented direction, without loosing the properties of robustness and simplicity of implementation required by applications involving the high-quality encoding of a wide variety of video material (such as television broadcast).

Several characteristics were identified as important to the efficiency of a library-based coding system:

- the existence of a recursive design algorithm, capable of distributing the complexity of the design by several frames and producing a progressively better library;
- a design algorithm leading to a temporally sooth library, reducing the overhead associated with its update;
- an efficient update mechanism, capable of exploring this smoothness to reduce the cost of library transmission.

All these characteristics were incorporated in the library-based encoder described in section 4.4.

One additional desirable characteristic would be the capability for the library-based encoder to produce smooth library index fields, leading to reduced overhead, comparable to the achieved by the smooth motion vector fields characteristic of motion-compensated prediction. This characteristic can only be achieved by a structured library which, unfortunately, cannot be achieved with a raw vector quantizer. The ordering of the codebook by mean, described in section 4.4, was a step in the direction of achieving a more structured library and reducing the overhead.

The overhead associated with the transmission of the library indexes remains, however, the largest limitation on the overall performance of the present implementation of library-based encoding, preventing an overall gain over the MPEG-2 encoder so significant as that achieved in terms of the prediction efficiency. More work remains to be done in this area.

## 6.2 Directions for future work

The work developed in this thesis could evolve in two different directions. In the first, a set of enhancements could be added to the library-based encoder without changing its structure significantly. As mentioned above, the more important task here is to come up with a better solution for the minimization of the overhead associated with the transmission of the library indexes. This could probably be achieved with a different codebook structure and more clever encoding of these indexes. It seems likely, however, that the overhead will always be higher than that associated with motion-compensated prediction, limiting the performance gain of the library-based encoder using vector quantization.

A more challenging, and probably also more efficient, direction of evolution consists in the extension of the coding model towards a truly object-oriented model. This would imply the elimination of block-based processing, and the use of more powerful image segmentation techniques such as those referred in section 3.3. Once the objects in the scene were identified and decomposed into a set of representative features (such as shape and texture), a clustering technique similar to the used in this thesis could be applied to create a library of these features and achieve the desired compression. For example, object shape can be approximated by a polygonal representation characterized by a small set of points in the 2-D plane, which can be seen as a vector for the purpose of library design.

Obviously, the implementation of such an encoder is significantly more complex than that of one using block-based processing and it is not clear that it can be simultaneously efficient and suitable for a wide variety of input material (robust). This because a more accurate model of the scene will be more efficient, but will also be more specific and less tolerant to different types of input material. However, as partially shown by this thesis, the use of coding schemes based on improved models of the image formation process seems to be the only way to achieved a significant increase in coding efficiency over that of current block-based techniques.

# Bibliography


[1] H. Abut, editor. *Vector Quantization.* IEEE Press, 1990.

[2] T. Bell, J. Cleary, and I. Witten. *Text Compression.* Prentice Hall, 1990.

[3] R. Clarke. *Transform Coding of Images.* Academic Press, 1985.

[4] T. Cover. Enumerative Source Coding. *IEEE Trans. on Information Theory*, Vol. IT-17, June 1973.

[5] A. Drake. *Fundamentals of Applied Probability Theory.* McGraw-Hill, 1987.

[6] W. Equitz. A New Vector Quantization Clustering Algorithm. *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 37, October 1989.

[7] N. Farvardin and J. Modestino. Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources. *IEEE Trans. on Information Theory*, May 1984.

[8] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, 1990.

[9] R. Gallager. Variations on a theme by Huffman. *IEEE Trans. on Information Theory*, Vol. IT-24, November 1978.

[10] A. Gersho. On the Structure of Vector Quantizers. *IEEE Trans. on Information Theory*, Vol. IT-28, March 1982.

[11] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Press, 1992.

[12] H. Gouraud. Continuous Shading of Curved Surfaces. *IEEE Trans. on Computers*, Vol. 20, June 1971.

[13] R. Gray. Vector Quantization. *IEEE Acoustics, Speech, and Signal Processing Magazine*, Vol. 1, April 1984.

[14] A. Habibi. Comparation of the N-th order DPCM Encoder with Linear Transformations and Block Quantization Techniques. *IEEE Trans. on Communications*, Vol. COM-19, December 1971.

[15] M. Hotter. Object-oriented Analysis-synthesis Coding Based on Moving Two-dimenional Objects. *Signal Processing: Image Communication*, Vol. 2, December 1990.

[16] Y. Huang and P. Shultheiss. Block Quantization of Correlated Gaussian Random Variables. *IEEE Trans. on Communication Systems*, September 1963.

[17] D. A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proceedings of the IRE*, Vol. 40, September 1952.

[18] ISO-IEC/JTC1/SC29/WG11. *MPEG Test Model*, MPEG93/457.

[19] ITU, CCIR. *Recomendation 601: Encoding Parameters of Digital Television for Studios*, 1986.

[20] N. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice Hall, 1984.

[21] R. Koch. Automatic Modeling of Natural Scenes for Generating Synthetic Movies.

[22] G. G. Langdon. An Introduction to Arithmetic Coding. *IBM Journal Res. Devel.*, Vol. 28, March 1984.

[23] D. LeGall. MPEG: a Video Compression Standard for Multimedia Applications. *Communications of the ACM*, Vol. 34, April 1991.

[24] J. S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice Hall, 1992.

[25] J. Limb and H. Murphy. Measuring the Speed of Moving Objects from Television Signals. *IEEE Trans. on Communications*, Vol. 23, 1975.

[26] Y. Linde, A. Buzo, and R. Gray. An Algorithm for Vector Quantizer Design. *IEEE Trans. on Communications*, Vol. 28, January 1980.

[27] S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on Information Theory*, Vol. IT-28, March 1982.

[28] J. Makhoul, S. Roucos, and H. Gish. Vector Quantization in Speech Coding. *Proceedings of the IEEE*, Vol. 73, November 1985.

[29] F. Mounts. Video Encoding Systems with Conditional Element Replenishment. *Bell Syst. Tech. Journal*, Vol. 48, 1969.

[30] H. Musmann, M. Hotter, and J. Ostermann. Object-oriented Coding of Moving Images. *Signal Processing: Image Communication*, Vol. 1, October 1989.

[31] H. Musmann, P. Pirsch, and H. Grallert. Advances in Picture Coding. *Proceedings of the IEEE*, Vol. 57, April 1985.

[32] N. Nasrabadi and R. King. Image Coding Using Vector Quantization: A Review. *IEEE Trans. on Communications*, Vol. 36, August 1988.

[33] A. Netravali and B. Haskell. *Digital Pictures: Representation and Compression.* Plenum Press, 1988.

[34] A. Netravali and J. Robbins. Motion Compensated Television Coding: Part I. *Bell Syst. Tech. Journal*, Vol. 58, March 1979.

[35] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 1991.

[36] D. Pearson, editor. *Image Processing.* McGraw-Hill, 1991.

[37] W. Pennebaker and J. Mitchell. *JPEG: Still Image Data Compression Standard.* Van Nostrand Reinhold, 1993.

[38] B. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, Vol. 18, June 1975.

[39] C. E. Shannon. A Mathematical Theory of Communication. *Bell Syst. Tech. Journal*, Vol. 27, Jully 1948.

[40] G. Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, Inc., 1985.

[41] R. Tsai and T. Huang. Estimating Three-dimensional Motion Parameters of a Rigid Planar Patch. *IEEE Trans. on Acoust. Speech and Signal Processing*, Vol. 29, December 1981.

[42] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Trans. on Information Theory*, Vol. IT-23, 1977.

# Acknowledgments

I would like to say thanks to some people which, in a way or another, made this work possible.

First, thanks to Andy Lippman, my advisor, for the good ideas and direction, and an unique perspective of research and life; to the thesis readers, Ted Adelson and Arun Netravali, for the patience to read it and the comments they provided; to Mike Bove for reviewing parts of the thesis; to Henry Holtzman for his suggestions, and the sense of relief provided by knowing that he keeps a secretary messier than mine; to everybody else in the garden for making this a fun place to work; and to Gillian, Linda, Santina, and Jill for the efficiency, and support.

Thanks also to all my old friends with whom I have a great time whenever I go home. In particular, thanks to Paulo (my colleague in this US adventure) for the long telephonic chats, and for first challenging me to study abroad.

Special thanks to all my family for the love and support that made my stay away from home much easier: my mother and father for the constant display of affection, the "royal" treatment I get when I go home, the cookies and cakes, the handling of all the problems that needed to be solved in Portugal, and for keeping me updated on all the news; my brother Pedro for the endless telephone conversations which made me feel closer to home, the news about the soccer and the "noite", and the letters and Benfica videotapes; my younger brother Ricardo and sister Verinha for visiting me, sending me fun letters and postcards, and giving me all their support; and my

grandmother for never letting me forget the pleasure of the old style Portuguese food, when I go home.

Finally a very special thanks to Manuela, the real strength behind the success of my stay abroad. Thanks for the love, the care, the company, the nice meals, the long hours of study together, the good ideas, and all the fun. I am glad I was lucky enough to find you.