

# Video Coding in a Broadcast Environment

by

Manuela Alexandra Trigo Miranda de Sousa Pereira

Licenciatura, Electrical Engineering  
Faculdade de Engenharia da Universidade do Porto, Portugal  
(1986)

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

©1993 Massachusetts Institute of Technology.  
All rights reserved.

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

**Rotch**

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

**[OCT 18 1993**

LIBRARIES

# Video Coding in a Broadcast Environment

by

Manuela Alexandra Trigo Miranda de Sousa Pereira

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

A new image-coding algorithm based on MPEG-2 and specially oriented to applications involving the transmission, at various bit rates, of previously stored digital video material is presented. Coding of the input material is performed in two stages. An interframe lossless or nearly lossless coder is first used to reduce storage requirements and collect global information about the entire footage. This information is then used to improve the coding quality and reduce complexity of a second stage consisting of a bank of MPEG-2 based encoders, working in parallel, at the desired bit rates.

This algorithm, when compared with straight storage of the uncompressed footage followed by a bank of MPEG-2 coders, requires less storage, is less complex, and provides better image quality.

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

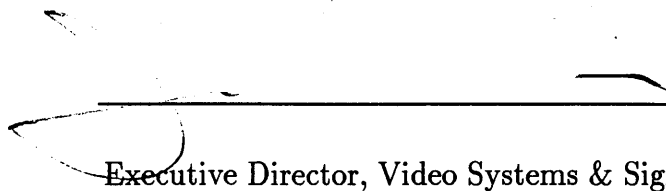
The work reported herein is supported by a contract from the Movies of the Future consortium, including Apple Computer Inc., Bellcore; Eastman Kodak Company; Intel Corporation; Viacom International Inc., and Warner Brothers Inc., and by DARPA/ISTO under contract DAAD-05-90-C-0333.

# Video Coding in a Broadcast Environment

by

Manuela Alexandra Trigo Miranda de Sousa Pereira

Reader:



---

Jules Bellisio  
Executive Director, Video Systems & Signal Processing Research  
Bellcore

Reader:



---

Donald Mead  
Chief Scientist  
Hughes Aircraft Corporation

*To my mother,  
and the memory of my father.*

# Contents

- 1 Introduction** **7**
  - 1.1 The problem . . . . . 7
  - 1.2 The approach . . . . . 8
  
- 2 Image coding overview** **10**
  - 2.1 Predictive coding . . . . . 11
    - 2.1.1 Differential Pulse Coded Modulation . . . . . 11
    - 2.1.2 Prediction . . . . . 13
  - 2.2 Transform coding . . . . . 17
    - 2.2.1 Karhunen-Loeve Transform . . . . . 18
    - 2.2.2 Discrete Cosine Transform . . . . . 20
  - 2.3 Quantization . . . . . 22
    - 2.3.1 Optimal quantization . . . . . 23
    - 2.3.2 Uniform quantization . . . . . 24
    - 2.3.3 Quantizer performance . . . . . 25
    - 2.3.4 Quantization of transform coefficients . . . . . 26
  - 2.4 Entropy coding . . . . . 29
    - 2.4.1 Huffman coding . . . . . 30
    - 2.4.2 Arithmetic coding . . . . . 31
  
- 3 The MPEG-2 video standard** **33**
  - 3.1 The MPEG-2 syntax . . . . . 34
  - 3.2 The MPEG-2 compression algorithm . . . . . 35

3.2.1	Input formats . . . . .	35
3.2.2	Temporal decorrelation . . . . .	37
3.2.3	Spatial decorrelation . . . . .	41
3.2.4	Quantization . . . . .	41
3.2.5	Entropy coding . . . . .	43
<b>4</b>	<b>The 2-stage coding algorithm</b>	<b>45</b>
4.1	Motion estimation . . . . .	47
4.2	Lossless and nearly-lossless coding . . . . .	49
4.3	Footage characteristics . . . . .	52
4.3.1	Processing of hard to code frames . . . . .	53
4.3.2	Preload coding . . . . .	54
4.3.3	Improved rate-control . . . . .	57
4.4	Re-coder stage . . . . .	61
<b>5</b>	<b>Simulation results</b>	<b>63</b>
5.1	First-stage encoding experiments . . . . .	64
5.2	Motion estimation experiments . . . . .	67
5.3	Second-stage encoding experiments . . . . .	70
5.3.1	Processing of hard to code frames and preload coding . . . . .	70
5.3.2	Improved rate-control . . . . .	73
5.4	Overall 2-stage encoder performance . . . . .	74
<b>6</b>	<b>Conclusions</b>	<b>77</b>
	<b>Bibliography</b>	<b>79</b>
	<b>Acknowledgments</b>	<b>82</b>

# Chapter 1

## Introduction

The main goal of a scalable coding system is to provide a coded representation of the original footage that is flexible enough so that the user can decide how much picture quality to use (or buy). This thesis presents a different approach to the same problem. The motivation here is to find the representation most suited to the distribution of movies from a central database that allows each user to get as much picture quality as the channel to which he is connected can support, or he wants to pay, or his display device can depict. Since it is difficult to determine the minimum degradation satisfactory to all types of users, this representation should be able to provide up to perfect reconstruction and allow the user to set the quality limits.

### 1.1 The problem

The system presented in this thesis enables a fileserver for a large movie database to feed several distribution outlets, each at a different bandwidth or bit rate. This is a form of scalable “pre-coding” where each user is provided with as much image quality as his communications path allows.

This system is oriented to the class of applications where the input video material

is available on a digital storage format. The most common example of this type of applications is the broadcasting of movies or any other pre-recorded material. Another one can be conceived in the future where, from his workstation, a remote user logs in a movie server and requests a movie to be displayed on a window in his terminal. These applications present some particular characteristics:

- the delay inherent to the encoding process is not a constraint;
- since the footage is entirely available before transmission, some a priori knowledge of its content can be used in the encoding stage;
- different users can have different quality requirements, i.e. the encoder should be able to provide bitstreams at distinct bit rates;

which should be taken into account in order to reduce storage requirements, minimize complexity, and improve coding efficiency.

## 1.2 The approach

As the basis of this system, a new coding scheme, which can be viewed as an extension to MPEG-2, is proposed. MPEG-2 is a generic standard oriented to serve a large number of applications. Therefore, although providing a good quality general propose coder, its performance can be improved for particular applications, as the ones discussed above, by tailoring the algorithm to specific characteristics of these applications.

The scheme now proposed is constituted by two distinct coding stages. In the first stage, the raw footage is losslessly or nearly losslessly compressed, leading to an intermediate format that minimizes storage requirements. This operation is done once, off-line, and, therefore, its complexity is not a constraint. During this stage, global information about the footage, which will be used in the second stage, is



acquired. The second stage consists of the encoding for transmission at various bit rates. This operation is MPEG-2 based. Since it is done on-line, and possibly by different encoders working in parallel at different bit rates, it is important that it can be kept as simple as possible.

This thesis is organized as follows. Chapter 2 presents an overview of the area of image coding, with particular emphasis on the aspects relevant to the understanding of the concepts later presented in the body of the thesis. Chapter 3 describes the MPEG-2 video compression algorithm, which provides a good example of the state of art in this field, and was the starting point for all the work developed. Chapter 4 describes the 2-stage coding algorithm proposed, pointing out the differences and gains over straightforward storage of the uncompressed footage followed by MPEG-2 coding. Finally, simulation results and conclusions are presented in chapters 5 and 6.

# Chapter 2

## Image coding overview

Image coding has been, during the last decade, one of the main areas of research in the field of digital signal processing. This chapter provides a brief description of the basic concepts behind the most common image coding and compression techniques. Particular emphasis is given to the ones that are relevant to understand the fundamental ideas later presented in the thesis. This review is not intended to be exhaustive, the reader is referred to [1] - [7] for a complete overview on this subject.

A typical sequence of images presents a high degree of correlation or redundancy either in the temporal (between consecutive frames) and in the spatial (between pixels within the same frame) domains. Also, the human visual system is insensitive to some types of degradation that can occur when an image is processed. The goal of an image compression system is to exploit the existing redundancy and to place the allowable degradations in a visually optimum way in order to minimize transmission bit rate or storage requirements. Among the large number of techniques presented in the literature, predictive coding in the temporal domain and transform coding in the spatial domain have gained widespread acceptance by the image coding community. Also important is the role of entropy coding, which provides compression without quality degradation.

The following sections describe in some detail the image coding techniques referred above. Section 2.1 presents predictive coding and, in particular, interframe prediction. Section 2.2 discusses transform coding with special emphasis on the KLT and the DCT. Quantization is discussed in section 2.3, particularly optimal quantization, uniform quantization, and quantization of transform coefficients. Finally, section 2.4 presents entropy coding.

## 2.1 Predictive coding

The most basic form of transmission of a signal in a digital format consists in the use of *Pulse Coded Modulation (PCM)*. In this technique, the continuous amplitude of the input signal is discretized into a finite set of amplitude values. For most image applications, a set of 256 distinct levels (which can be digitally represented with 8 bits/sample) is sufficient to provide a high-quality reconstructed monochrome signal<sup>1</sup>.

Since PCM treats each sample independently of all the others, it cannot exploit the correlation that may exist between them. More elaborated techniques do not transmit the input sample in itself, but a value which results from processing that takes into account the dependencies between samples. One such technique is *predictive coding* which consists in the transmission of the difference between the original sample and a prediction of it, based on previously transmitted samples.

### 2.1.1 Differential Pulse Coded Modulation

The block diagram of figure 2.1 presents the most basic configuration of predictive coding. This configuration is generally referred to as *DPCM (Differential Pulse Coded Modulation)* and consists of a prediction loop that produces a estimate of the input

---

<sup>1</sup> For a color signal, 8 bits for each of the RGB components are sufficient for high quality.

sample, and a quantizer which provides compression for the desired bit rate.

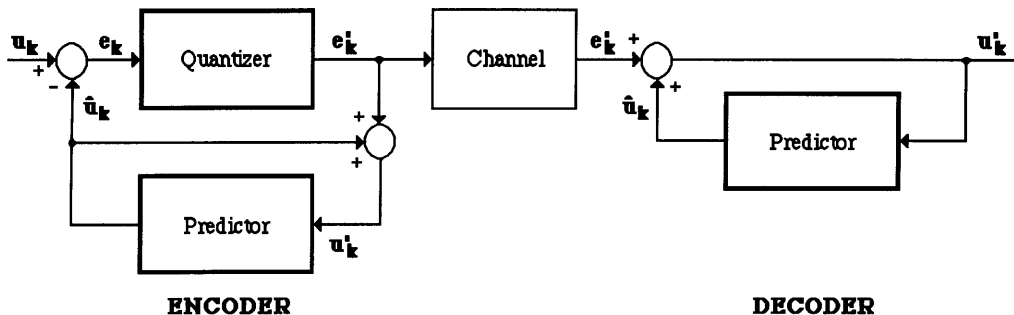


Figure 2.1: Block diagram of a DPCM system.

For an input signal  $u_k$ , the predictor output  $\hat{u}_k$  is given by

$$\hat{u}_k = \sum_{k=1}^N a_k u'_{N-k}, \quad (2.1)$$

where  $a_k$  and  $u'_{N-k}$  represent, respectively, the  $k$ th coefficient of the  $N$ th order predictor and the decoder output at the instant  $N - k$ .

The quantizer input  $e_k$ , usually referred to as *prediction error*, is the difference between the input amplitude  $u_k$  and the prediction  $\hat{u}_k$

$$e_k = u_k - \hat{u}_k. \quad (2.2)$$

The quantizer performs a non-linear mapping of the input amplitudes ( $e_k$ ) to a set of pre-established output levels ( $e'_k$ ), as described in section 2.3.

In the decoder, a replica of the original signal is reconstructed by adding the prediction  $\hat{u}_k$  to the quantized value  $e'_k$

$$u'_k = \hat{u}_k + e'_k. \quad (2.3)$$

The introduction of the decoder in the encoder loop eliminates the propagation

of quantization errors since

$$u'_k - u_k = (\hat{u}_k + e'_k) - (\hat{u}_k + e_k) = e'_k - e_k = e_q, \quad (2.4)$$

i.e., for each sample, the reconstruction error is equal to the *quantization error*  $e_q$  of the difference signal  $e_k$ .

Since the energy of the quantization noise is proportional to that of the signal to be quantized [8] and the prediction error  $e_k$  has considerably smaller energy than the input  $u_k$ , DPCM provides higher reconstruction quality than PCM, for the same bit rate<sup>2</sup>.

The design of a DPCM encoder consists in the joint design of the optimal predictor and quantizer for the desired bit rate. Given the dependence between these two elements, this results in a highly non-linear problem for which a solution has not yet been found. In practice, the predictor is optimized considering the quantization error negligible, and then the optimal quantizer for the prediction error thus obtained is determined.

### 2.1.2 Prediction

Ignoring the quantization error, considering a linear predictor, and assuming that the input values are samples of a wide-sense stationary stochastic process, the optimum predictor coefficients can be found by solving the Yule-Walker equations [9]

$$R_l = \sum_{k=1}^N a_k R_{l-k} \quad l = 1, 2, \dots, N, \quad (2.5)$$

where  $a_k$  is the  $k$ th coefficient of the  $N$ th order predictor and  $R_l$  the autocorrelation function of the input sequence  $u_k$ . Due to the complexity involved in solving this system of equations and the inadequacy of the assumptions on which it is based

---

<sup>2</sup> Or, analogously, the same distortion with a lower bit rate.

to typical image sequences, the optimal predictor is generally not used in practice. Instead, simpler implementations, based on a reduced number of samples and intuitive heuristics, are usually considered.

Common predictors can be classified in three types, according to the location of the samples used for prediction:

- *unidimensional predictors* are based on the last or on a set of previous samples in the current line;
- *two-dimensional predictors* are based not only on samples of the current line, but also on samples of previous lines;
- *three-dimensional predictors* are based on samples of previous lines of the same frame as well as samples from previously transmitted frames.

Predictors of the first two types are commonly known as *intraframe predictors*, while those belonging to the last class are designated by *interframe predictors*.

In the case of intraframe prediction, it has been shown [10] that, although there is no theoretical limitation on the number of samples that can be used in the predictor, its gain saturates for about three elements. There is, therefore, no advantage in using higher order predictors, which makes intraframe schemes fairly simple.

On the other hand, interframe predictors, having the capability to explore temporal correlation (which is high for typical image sequences), provide higher coding efficiency at the cost of increased complexity. However, with the recent advances in semiconductor technology, complexity constraints are becoming an issue of lesser importance, and interframe prediction is a technique of widespread use in video coding.

## Interframe prediction

The simplest form of interframe prediction consists in subtracting from each input pixel the corresponding one (i.e. with the same spatial coordinates) in the previous frame. This approach is efficient for the stationary areas of the sequence, such as the background or still objects, but fails when in the presence of motion. More accurate prediction can be achieved incorporating *motion estimation/compensation* techniques into this scheme.

The basic idea behind motion estimation is to find for each moving object its location in the previous frame. Once this location is found, the previous image can be warped (motion compensated), minimizing the prediction error. This procedure originates some side information (the motion parameters that describe the warp), which must be transmitted so that the decoder can perform the same operation as the encoder. However, since pixels belonging to the same object will have similar motion, this side information can be coded without significant increase of the total bit rate. In this way, the cost due to the overhead is largely compensated by the gain due to the reduced energy of the prediction error; and, overall, motion compensation provides higher coding efficiency.

The best approach to motion estimation would be to perform a segmentation of the sequence of frames into the objects that compose it, and then find the motion parameters associated with each object. However, the complexity of such an approach is quite high and, in practice, much simpler implementations are used. The most common solution consists in splitting each frame into a set of blocks of pixels and motion compensating each of these blocks. This solution has the advantages of simplicity and easy integration with the block-based spatial decorrelation techniques to be described later.

The most widely used method to perform block-based motion estimation is the *block matching* [11] technique. For each input block, the encoder searches for the closest match in the previous frame. The measure of similarity between the current

and the block candidate for prediction is typically the *Sum of the Squared Errors (SSE)*:

$$SSE = \sum_{x,y \in R} [f(x, y, t) - f(x - d_x, y - d_y, t - 1)]^2, \quad (2.6)$$

where  $x$  and  $y$  are the coordinates of the pixels that make the block,  $R$  its region of support (typically 8x8 or 16x16),  $d_x$  and  $d_y$  the displacements in the horizontal and vertical directions, and  $f(x, y, t)$  the amplitude of the pixel  $(x, y)$  in the frame  $t$ .

A slightly different measure, the *Sum of Absolute Differences (SAD)*, is sometime used to avoid the complexity of the squaring operation:

$$SAD = \sum_{x,y \in R} |f(x, y, t) - f(x - d_x, y - d_y, t - 1)|. \quad (2.7)$$

As can be inferred from the above equations, this block-matching operation is very heavy in terms of computation because the distance measure has to be computed for each of the possible displacements. Therefore, these displacements are usually confined to a subset of pixels of the previous frame, designated by *search window*. Nevertheless, the computational load can still be heavy and some sub-optimal but faster algorithms have been proposed in the literature [12]. Among these, the most popular is the *three-step search* method.

After motion estimation, i.e. upon finding the closest match, this is then displaced and used as prediction to the input block. The resulting prediction error is then coded and transmitted with the *displacement* or *motion vectors* ( $d_x$  and  $d_y$ ). The decoder uses these vectors to add the motion-compensated block from the previous frame to the coded residual.

Different implementations of block matching differ mainly in the number of frames used for prediction, and in the accuracy of the motion vectors. The most basic solution is to use the previous frame only, and integer motion vectors. More elaborate schemes use two frames (one from the past and one from the future) and fractional accuracy. Obviously, these extensions originate extra complexity due to the need of extra frame



stores and interpolation.

## 2.2 Transform coding

The term *transform coding* is used to characterize image coders where the input signal is transformed to a new domain, typically the frequency domain, before quantization and entropy coding. The input image is first split into a set of blocks and a reversible transformation is then applied to each block, mapping its pixels into a set of *transform coefficients*. To achieve high efficiency, the transform operation must provide:

- maximum decorrelation between output coefficients;
- maximum energy compaction, i.e. concentration of the energy in a minimum number of output coefficients.

Maximum decorrelation between coefficients is a desirable property because it increases the efficiency of the scalar quantization stage that usually follows the transform. As will be seen in section 2.3, a scalar quantizer cannot exploit the redundancy between samples, which results in decreased performance when they are correlated. However, if after the transform operation there is no correlation, each sample can be quantized by itself, using the optimal scalar quantizer as determined by its statistical characteristics, without significant decrease in coding efficiency.

Energy compaction is desirable because it enables the quantizer stage to quantize accurately a few coefficients that contain most of the energy and quantize coarsely, or even throw away, most of the others. In this way, the bit rate can be significantly reduced without considerable degradation in image quality.

### 2.2.1 Karhunen-Loeve Transform

Among all the transforms [13], the *Karhunen-Loeve transform (KLT)* is optimal in the sense that it provides complete coefficient decorrelation and the best energy compaction. The basic idea behind this transform is that complete decorrelation can be achieved by a rotation of coordinate axes.

Consider the unidimensional<sup>3</sup> vector  $\mathbf{v} = (v_1, \dots, v_N)^T$ , where the  $v_i$  are samples of the input image, and  $\mathbf{\Lambda}_v$  is the covariance matrix of this vector. The element in the *i*th row and *j*th column of  $\mathbf{\Lambda}_v$ ,  $\lambda_{ij}$ , is by definition

$$\lambda_{ij} = E[(v_i - m_{v_i})(v_j - m_{v_j})], \quad (2.8)$$

where  $m_{v_i}$  is the expected value of  $v_i$ . Or, in matrix representation,

$$\mathbf{\Lambda}_v = E[(\mathbf{v} - m_v)(\mathbf{v} - m_v)^T]. \quad (2.9)$$

Since, from 2.8,  $\lambda_{ij} = \lambda_{ji}$ , the matrix  $\mathbf{\Lambda}_v$  is symmetric and, therefore, has a complete set of orthonormal eigenvectors [14]. Consider now the vector

$$\mathbf{u} = \mathbf{T} \mathbf{v}, \quad (2.10)$$

where  $\mathbf{T}$  is the matrix whose rows are the transposes of the orthogonal eigenvectors of  $\mathbf{\Lambda}_v$ . The linear transformation of the vector  $\mathbf{v}$  into the vector of transform coefficients  $\mathbf{u}$  defined by this matrix is defined as the Karhunen-Loeve Transform<sup>4</sup> (KLT) of  $\mathbf{u}$ .

From equation 2.9, with  $\mathbf{v}$  replaced by  $\mathbf{u} = \mathbf{T} \mathbf{v}$ , it is easily shown that the covariance matrix of  $\mathbf{u}$ ,  $\mathbf{\Lambda}_u$ , is

$$\mathbf{\Lambda}_u = \mathbf{T} \mathbf{\Lambda}_v \mathbf{T}^T. \quad (2.11)$$

---

<sup>3</sup> The results presented can easily be extended to higher dimensions.

<sup>4</sup> In the general case, where  $\mathbf{T}$  is any orthonormal matrix, the operation in equation 2.10 is designated by transform coding.

But, since  $\mathbf{T}$  is an orthonormal matrix,  $\mathbf{T}^T = \mathbf{T}^{-1}$ , and 2.11 is equivalent to

$$\mathbf{\Lambda}_{\mathbf{u}} = \mathbf{T} \mathbf{\Lambda}_{\mathbf{v}} \mathbf{T}^{-1} = \text{diag}(\lambda_{11}, \lambda_{22}, \dots, \lambda_{NN}), \quad (2.12)$$

i.e. the vector  $\mathbf{u}$  has a diagonal covariance matrix and thus, from equation 2.8, its components are uncorrelated.

From the above, it can be seen that the KLT is nothing more than a rotation of the vectors of the basis, which spans the  $N$ -dimensional vector space of input blocks. This can be illustrated by a simple example [15] of a block constituted by two adjacent samples and a quantizer with 3 bits/sample. The set of the two samples can assume, after quantization, one of the 64 values represented in figure 2.2.

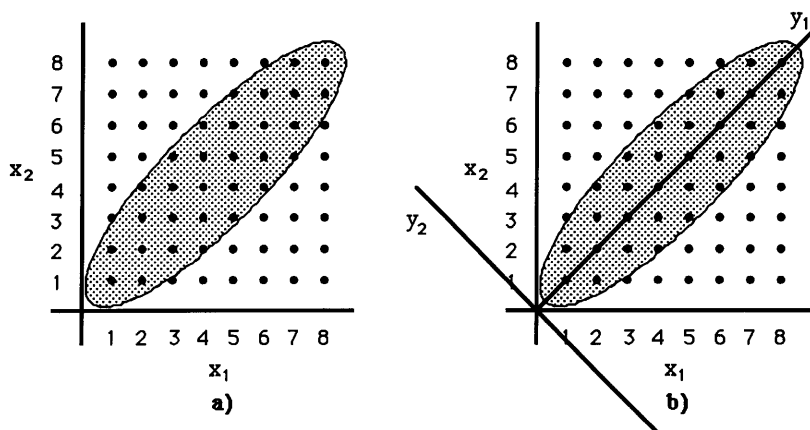


Figure 2.2: Example of spatial decorrelation provided by transform coding. From [15].

Given the redundancy presents on a typical image, the two samples are more likely to have identical amplitudes, which results in the high concentration of points near the line  $x_1 = x_2$  represented in figure 2.2 a). By rotating the coordinate system, from the axes  $(x_1, x_2)$  to the axes  $(y_1, y_2)$ , the concentration of points occurs in the vicinity of the axis  $y_1$  (figure 2.2 b)), and the variables  $y_1$  and  $y_2$  become uncorrelated.

This “decorrelating effect” can be better understood from the following arguments. In a), if the value of  $x_1$  is known, then the value of  $x_2$  can be guessed with reasonable accuracy. On the other hand, on b), by knowing the value of  $y_1$ , very little can be said

about the value of  $y_2$ . This would even be easier to see if, instead of ellipsoidal, the shaded region were rectangular. In this case, and supposing an uniform probability density,  $y_1$  and  $y_2$  would clearly be independent.

The energy compaction property is also exemplified in the figure. In a), the two variables  $x_1$  and  $x_2$  have similar energy; while, in b),  $y_1$  has much higher energy than  $y_2$ . Notice, however, that the total energy remains unchanged since only a rotation of axes was performed.

Despite being optimal, the KLT is not used in typical applications because of its computational complexity. This complexity is mainly due to the need of computing the covariance matrix and associated eigenvalues for the input. The importance of the KLT relies on the fact that, being optimal, it provides a reference to which the performance of another sub-optimal transforms can be compared.

Various sub-optimal transforms [16] of easier implementation, such as Fourier, cosine, Walsh-Hadamard, Haar, and Slant, have been considered for image processing. Among them, the *Discrete Cosine Transform (DCT)* [17] [18] is widely accepted as the more efficient since it achieves performance closer to the KLT, both in terms of energy compaction and coefficient decorrelation.

### 2.2.2 Discrete Cosine Transform

Several slightly different definitions of the DCT have been presented in the literature. The following one [6] will be adopted in the remaining of this thesis. Consider a set of pixels  $v(n)$ ,  $0 \leq n \leq N - 1$ . The *1-D DCT* of  $v(n)$  is defined as

$$C_v(k) = \sum_{n=0}^{N-1} 2v(n) \cos \frac{\pi}{2N} k(2n + 1), \quad 0 \leq k \leq N - 1, \quad (2.13)$$

mapping the  $N$  pixels of  $v(n)$  into  $N$  coefficients  $C_v(k)$ . From this, the original set of pixels  $v(n)$  can be reconstructed by the inverse operation, the *1-D IDCT*

$$v(n) = \frac{1}{N} \sum_{k=0}^{N-1} w(k) C_v(k) \cos \frac{\pi}{2N} k(2n+1), \quad 0 \leq n \leq N-1, \quad (2.14)$$

where  $w(k)$  is  $1/2$  for  $k=0$ , and  $1$  for  $1 \leq k \leq N-1$ . The *2-D DCT* can be obtained by separably applying the 1-D DCT in the vertical and horizontal dimensions.

From equation 2.13, it can be seen that the first transform coefficient  $C_v(0)$  is just the average or DC value of the input vector  $v(n)$ . This coefficient is, therefore, usually designated as the *DC coefficient*. The other coefficients, associated with the non-zero frequency components of the input, are designated as *AC coefficients*.

As referred above, the main advantage of the DCT comes from the fact that its energy compaction and decorrelation are very close to those of the KLT. Experimental results [13] have shown that the energy compaction of the DCT is very robust against different characteristics of the input image, i.e. it stays very close to that of the KLT for a large range of input statistics. This behavior is not observed for the decorrelating property. However, since the decorrelating efficiency of the KLT is itself very sensitive to variations of statistical properties (covariance matrix), it turns out that the DCT is close or better than a KLT not matched to the image statistics.

Furthermore, since the DCT is based on the set of basis functions represented in 2.13, it requires no coding overhead. On the other hand, unless a specific statistical model of the input is assumed (sacrificing optimality), the KLT requires the transmission of the basis functions to allow the inverse transform in the decoder.

From all these factors, the possible (marginal) gain of using the KLT is not enough to justify its increased complexity over that of the DCT. The DCT is, therefore, unanimously accepted as the most efficient transform for image coding.

## 2.3 Quantization

A quantizer maps each input sample  $e_k$  into one of a finite set of values<sup>5</sup>, designated by *reconstruction levels*. Typically, the range of possible input values is divided into  $N$  subsets, associated with the  $N$  possible reconstruction levels. The boundaries of these subsets are designated by *decision levels*. The design of a quantizer (see figure 2.3) consists in the determination of these reconstruction levels  $r_n$ ,  $n = 1, 2, \dots, N$ , and corresponding decision levels  $d_n$ ,  $n = 0, 1, \dots, N$ . An input amplitude  $e_k$  between  $d_{n-1}$  and  $d_n$  is mapped into the output  $e'_k = r_n$ .

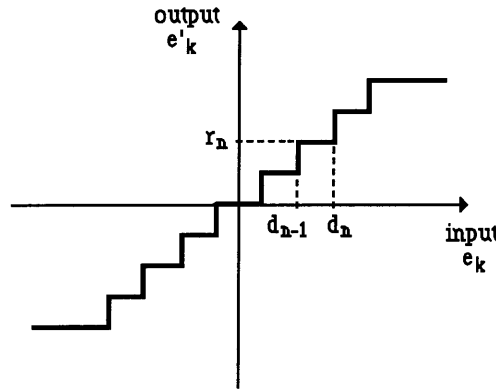


Figure 2.3: Quantizer characteristics.  $d_n$  and  $r_n$  are the decision and reconstruction levels, respectively. An input  $e_k$  between  $d_{n-1}$  and  $d_n$  is quantized to  $r_n$ .

Usually,  $d_n$  and  $r_n$  are chosen to minimize a distortion measure  $\mathcal{D}$ , which is given by

$$\mathcal{D} = E(e_q^2) = E[(e'_k - e_k)^2] = \int_{e_{k0}=-\infty}^{\infty} p_{e_k}(e_{k0}) (e'_k - e_{k0})^2 de_{k0}, \quad (2.15)$$

when the *Mean Square Error (MSE)* criteria is used. In this equation,  $p_{e_k}(e_{k0})$  is the *probability density function (pdf)* of the input  $e_k$ .

---

<sup>5</sup> In this discussion, it is assumed that each input value is quantized independently, what is known as *scalar quantization*. Alternatively, when a vector quantizer is used, the same index is assigned to a set of input values.

Since  $e'_k = r_n$  for  $d_{n-1} \leq e_k \leq d_n$ , equation 2.15 can be simplified to

$$\mathcal{D} = \sum_{n=1}^N \int_{e_{k0}=d_{n-1}}^{d_n} p_{e_k}(e_{k0}) (r_n - e_{k0})^2 de_{k0}. \quad (2.16)$$

Due to the lack of a better mathematically tractable distortion measure than the MSE, this equation is used to determine the optimal quantizer for a particular input pdf.

### 2.3.1 Optimal quantization

The minimization of  $\mathcal{D}$  leads to a pair of equations that determines the optimal quantizer, usually known as *Lloyd-Max* [19] [20], where the decision levels are the middle points between adjacent reconstruction levels

$$d_n = \frac{r_n + r_{n+1}}{2}, \quad 1 \leq n \leq N - 1, \quad (2.17)$$

with  $d_0 = -\infty$  and  $d_N = \infty$ , and the reconstruction levels are the centroids of  $p_{e_k}(e_{k0})$  between consecutive decision levels

$$r_n = \frac{\int_{e_{k0}=d_{n-1}}^{d_n} e_{k0} p_{e_k}(e_{k0}) de_{k0}}{\int_{e_{k0}=d_{n-1}}^{d_n} p_{e_k}(e_{k0}) de_{k0}}, \quad 1 \leq n \leq N. \quad (2.18)$$

The solution to both equations 2.17 and 2.18 is a non-linear problem. It has, however, been solved for some common probability density functions, such as the uniform, the Gaussian, and the Laplacian [8].

The performance of the MSE optimal quantizer is limited by two factors. First, an accurate probabilistic description of the input is not usually available. Second, the optimization criteria (MSE) is itself very weak in the sense that it does not take into account the characteristics of the human visual system. Due to these limitations, and since the implementation of a Lloyd-Max quantizer is generally more complex than

that of an uniform quantizer, the later is usually used in practical applications.

### 2.3.2 Uniform quantization

An *uniform quantizer* is one where the distance between any two consecutive decision levels (*step-size*) is constant. Consequently (see equation 2.17), the distance between any two consecutive reconstruction levels is also constant. As can be derived from 2.18, the uniform quantizer is the optimal Lloyd-Max quantizer when the pdf of the input is uniform.

The main attractive of uniform quantization is that it can be implemented with a simple division of the input sample by the step-size. Moreover, it turns out that the output of an uniform quantizer has lower entropy than that of the corresponding optimal quantizer [4] at the cost of some sacrifice in quantization error.

This result is intuitively acceptable by the following argument. For the optimal quantizer, the decision levels will be closer in the range of amplitudes where the input pdf is larger, and distant in the one where the input pdf has a small value. Therefore, the area under the pdf curve given a reconstruction level will be approximately the same, independently of the reconstruction value. Consequently, the probability of occurrence of the reconstruction values will be approximately uniform. On the other hand, for an uniform quantizer, and since the decision levels are equally spaced, the reconstruction levels associated with the regions of higher probability of the input will be more likely than those associated with the regions of lower probability. So, the probability of occurrence of the reconstruction levels will be peaked at the areas of greater probability of the input. Since uniform distributions have higher entropy, the optimal quantizer generates an output with larger entropy than the uniform one.

Due to this “low-entropy” property, the uniform quantizer can, if used with an entropy-coding stage, provide overall coding efficiency larger than that obtainable with the optimal Lloyd-Max quantizer. For this reason and the simplicity of im-



plementation, the joint use of uniform quantization and entropy coding is generally preferred to the use of optimal non-uniform quantization.

### 2.3.3 Quantizer performance

Due to the non-linearities involved, it is difficult to establish a closed-form equation relating the number of reconstruction levels (or equivalently, the number of bits) of the quantizer and the distortion originated by quantization. There is, however, a particular case for which the problem is solvable and that can be used as a reference of performance for the general case. This particular situation is known as *high-resolution quantization*, and occurs when the number of reconstruction levels is very high. When this happens, the pdf of the input signal can be assumed to be smooth enough such that it is constant between any two successive decision levels. If a uniform quantizer of step-size  $\Delta$  is used, equation 2.16 simplifies to

$$\mathcal{D} = \sum_{n=1}^N p(n) \int_{r_n-\Delta/2}^{r_n+\Delta/2} (r_n - e_{k0})^2 de_{k0} = \sum_{n=1}^N p(n) \frac{\Delta^3}{3}, \quad (2.19)$$

where  $p(n)$  is the value of  $p_{e_k}(e_{k0})$  for  $e_{k0}$  in  $[d_{n-1}, d_n[$ .

Since the probability  $p_n$  of the input being in  $[d_{n-1}, d_n[$  is, under the assumptions stated above, approximately equal to  $p(n)\Delta$ , after some simple manipulations of 2.19, the total distortion is given by

$$\mathcal{D} = \frac{\Delta^2}{12}. \quad (2.20)$$

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

$$SNR = 10 \log\left(\frac{\sigma^2}{\mathcal{D}}\right) = 10 \log\left(12 \frac{\sigma^2}{\Delta^2}\right), \quad (2.21)$$

and, since

$$\Delta = \frac{R}{N} = \frac{R}{2^b}, \quad (2.22)$$

where  $\sigma^2$  is the input variance,  $R$  the range of input values, and  $b = \log_2 N$  the number of quantization bits,

$$SNR = 6.02b + C, \quad (2.23)$$

with  $C = 10 \log(12\sigma^2/R^2)$ . Equation 2.23 provides a useful rule of thumb which says that the SNR increases by approximately 6 dB per quantization bit.

As a final remark, it should be emphasized that all the analysis above is directed to the minimization of the MSE. As has already been referred, this distortion measure is not well suited to the properties of the human eye. An alternative approach would be to design quantizers optimized according to subjective-fidelity criterion. Some results of studies in this area can be found in [21] and [22], where quantizers that limit the quantization error below a threshold of perceptibility of the human eye are presented.

### 2.3.4 Quantization of transform coefficients

It was seen in section 2.2 that efficient coding can be achieved by the use of a decorrelating transform followed by quantization. The advantage of this two-step procedure comes from the fact that distinct frequency coefficients have different characteristics and can, therefore, be quantized with different accuracies without decrease of the overall coding performance.

For example, it was mentioned in section 2.2.2 that the first coefficient of the DCT of an input vector is just the mean (DC) value of that vector, while the other coefficients represent the AC frequencies of the input. Efficient quantization of the DC coefficient is crucial because the human eye is very sensitive to the block artifacts (usually known as *blocking effect*) created by discontinuities in average luminance between consecutive blocks. On the other hand, the AC coefficients can be quantized more coarsely since the eye is less sensitive to degradation in areas of high activity (such as edges).

So, to fully explore the advantages of transform coding, it is necessary to solve two major problems:

- how to distribute the total bit-rate between the different coefficients according to their characteristics;
- given a desired coefficient bit-rate, how to design the best quantizer for that coefficient.

The first problem, *bit allocation*, has been addressed in the literature, although an optimal and easily implementable solution has not yet been found. Huang and Schultheiss were the first to present a solution to this problem [23], using an approach based in Lagrange multipliers, and showing that the optimal bit allocation should be based on the variance of the different coefficients according to

$$b_i = \frac{B}{N} + \frac{1}{2} \log_2 \frac{\sigma_i^2}{[\prod_{j=1}^N \sigma_j^2]^{1/N}}, \quad 1 \leq i \leq N, \quad (2.24)$$

where  $b_i$  is the bit rate allocated to  $i$ th coefficient,  $B$  the total bit rate,  $N$  the number of coefficients in the block, and  $\sigma_i^2$  the variance of coefficient  $i$ . This equation can be extended to take into account the perceptual importance of each of the coefficients by introducing coefficient weights  $w_i$

$$b_i = \frac{B}{N} + \frac{1}{2} \log_2 \frac{w_i \sigma_i^2}{[\prod_{j=1}^N w_j \sigma_j^2]^{1/N}}, \quad 1 \leq i \leq N, \quad (2.25)$$

where  $\sum_{i=1}^N w_i = 1$ .

The bit allocation thus obtained is theoretically optimal but, in general, cannot be implemented since the  $b_i$ 's are not guaranteed to be integer or even positive. In practice, it is, therefore, necessary to apply an additional algorithm that reassigns some of the bits so that these constraints are not violated. The constraint of an integer number of bits is not a problem if quantization is followed by a variable-length coding stage, such as the ones described in the next section. The constraint

of a positive number has, however, to be met under any circumstances.

Notice that, if a particular value of distortion is allowable for a certain coefficient and the coefficient variance is smaller than this distortion, there is no need to code the coefficient at all. In this case, the MSE due to this coefficient will be equal to its variance and lower than the desired distortion. This is the principle of an allocation technique widely used in practice: *threshold coding* [8]. In a threshold coder, the amplitude of the coefficients is compared with a pre-defined threshold, and the coefficient is transmitted only if its amplitude exceeds the threshold. This technique is efficient because it provides an adaptive selection of the coefficients that contain most of the energy. However, since the decoder cannot know in advance the position of the relevant coefficients, their position has to be transmitted, originating bit-rate overhead.

To avoid this overhead, a simpler approach, *zonal sampling* [8], is sometimes used. Here, the reasoning is that, due to the energy compaction provided by transform coding, most of the energy is always contained in a subset (generally, low-frequencies) of the coefficients. This subset is pre-defined, and only the coefficients in it are transmitted. Obviously, this implies a loss of efficiency when “non-standard” blocks with significant coefficients outside the subset have to be coded.

After a suitable bit-allocation procedure is determined, the problem of quantizer design must be addressed. As mentioned before, different coefficients have different characteristics, which should be taken into account in the design of the quantizers. Since a transform coefficient is a weighted sum of the random variables that compose the input vector (see equation 2.13), it would be expected from the *central limit theorem* [24] that the transform coefficients were Gaussian. However, since the input samples are not independent, this does not hold for every coefficient.

In fact, while a Gaussian or generalized Gaussian probability density is well suited for the AC coefficients, this does not happen for the DC coefficient [13]. Since the DC coefficient is just the mean of the input vector, its pdf is similar to that of the input

which is, for typical images, uniform. From this and the fact that the human eye is particularly sensitive to errors in the DC component, the DC coefficient is generally quantized by an uniform quantizer at high bit-rate (typically 8 bits).

As far as the AC coefficients are concerned, the best quantizer depends on the existence of an entropy coder after quantization. When entropy coding is not used, optimal Lloyd-Max quantizers (suited to the coefficient pdfs) should be chosen. In this case, for the sake of simplicity, it is common to assume that all AC coefficients are Gaussian and divide each coefficient by its variance before quantization. In this way, only one quantizer optimized for a unit-variance Gaussian pdf is required, reducing complexity at the cost of some overhead necessary to transmit the coefficient variances. When entropy coding is used, and as has seen in section 2.3.2, the uniform quantizer provides the highest efficiency and should be chosen.

## 2.4 Entropy coding

*Entropy coding* is usually the last stage of a typical image compression system, and can be jointly used with any of the compression techniques described above.

The goal of entropy coding is to reduce the bit rate, without additional quality degradation, by exploring the statistical properties of the signal to be coded. According to the information theory [25], the information contained in a symbol (which can be an amplitude value, a transform coefficient, etc.) is a function of the probability of occurrence of that symbol. For example, a particular amplitude value with high probability of occurrence contains less information than one with low probability.

The average amount of information, or *entropy*  $H$ , originated from a source that generates symbols from an alphabet  $s_i$ ,  $i = 1, \dots, N$ , with associated probabilities of

occurrence  $P_i$ , is given by

$$H = - \sum_{i=1}^N P_i \log_2 P_i \quad (\text{bits/symbol}). \quad (2.26)$$

Equation 2.26 provides the *theoretical lower bound* on the average bit-rate required to code symbols from that source and, therefore, sets a reference for the performance of an efficient entropy coder.

The basic idea behind entropy coding is to assign long codewords to unlikely input symbols and short codewords for the more likely ones, a process which is usually known as *variable-length coding*.

### 2.4.1 Huffman coding

One of the most common variable-length coding techniques is *Huffman coding* [26], which is optimal in the sense that it minimizes the average bit-rate. In fact, when each symbol is coded independently, the average symbol rate  $\bar{L} = \sum_{i=1}^N L_i P_i$  is

$$H \leq \bar{L} \leq H + 1 \quad (\text{bits/symbol}), \quad (2.27)$$

i.e. within 1 bit/symbol of the entropy. The algorithm used for the construction of a Huffman code [4] involves ordering the source symbols by probabilities  $P_i$ , consider them as nodes of a tree, and recursively:

- combine the two nodes with lowest probabilities, forming a new node with probability given by the sum of the probabilities of the combined nodes;
- assign a “0” or a “1” to each of the two branches associated with the new node;

and this process is repeated until a unique node with probability one is reached. The codeword associated with the symbol  $s_k$  can be found by, starting from the node

with probability one, following the branches necessary to reach the initial node with probability  $P_k$  and reading the “0s” and “1s” associated with these branches.

The performance given by 2.27 can be increased if, instead of coding each symbol independently, blocks of symbols are coded jointly. If each block has  $r$  symbols, the bound for the average symbol rate  $\bar{L}$  becomes

$$H \leq \bar{L} \leq H + \frac{1}{r} \quad (\text{bits/symbol}). \quad (2.28)$$

From 2.28, it can be seen that, theoretically, a bit rate arbitrarily close to the entropy  $H$  can be achieved by increasing the number of symbols coded jointly. In practice, however,  $r$  is limited to small values to avoid long encoding delays and large encoder buffers.

## 2.4.2 Arithmetic coding

A different approach to variable-length coding is *arithmetic coding* [27]. Here, unlike the case of Huffman coding, there is no unique correspondence between symbols and codewords. A sequence of  $k$  symbols is initially associated with the interval of real numbers between 0 and 1. This association is done by breaking the interval into  $k$  sub-intervals of size proportional to the probability of occurrence of each of the  $k$  symbols. According to the first symbol of the sequence, the coder chooses the sub-interval that is associated with it, and this sub-interval is then subdivided by applying the same procedure. In this way, the size of the interval becomes smaller as the number of symbols grows up, and the number of bits required to represent that interval increases.

It can be show [28] that, after a large number of symbols, the number of bits required to uniquely identify the sub-interval approaches the entropy  $H$  of the source. However, due to implementation constraints (such as the use of finite precision arith-

metic), this theoretical bound cannot be achieved in practice.

Nevertheless, arithmetic coding represents an improvement over Huffman coding because the implementation constraints posed to the former are less restrictive than those posed to the latter.



# Chapter 3

## The MPEG-2 video standard

Traditional image compression algorithms have been based on temporal predictive coding, where a displaced version of a region in a previously transmitted image is used to predict a region of the present frame. Most of the state of the art coders are based on this approach, differing in the way of handling the spatial correlation of the prediction error signal. The best example of the state of art is the MPEG-2<sup>1</sup> algorithm [29] [30] [31] based on this motion-compensation loop structure, and where, in the spatial domain, a DCT is used to remove spatial correlation before quantization and entropy coding.

MPEG-2 is a generic standard oriented to serve a large number of applications. It can be seen as a toolkit where, in order to satisfy the specific requirements of a particular application, a subset of its features can be used. Each of these subsets composes a different *profile* and provides a specific functionality. A profile can have different *levels*, generally associated with different resolutions of the input video material (such as SIF, SDTV, HDTV, etc.). In this chapter, we will focus on the *main level* of the *main profile*, which was the starting point for the work developed in this

---

<sup>1</sup> MPEG is the Moving Pictures Expert Group, a group of the International Organization for Standardization (ISO) working in the creation of a standard for the compression of video and associated audio.

thesis.

The MPEG-2 standard only specifies a bitstream syntax and associated decoder. Some degree of freedom is left in the encoder that can be used to optimize coding quality. Any encoder that generates a MPEG-2 bitstream will, however, have to be based on the basic interframe predictive coding structure mentioned above.

The following sections describe in some detail the main level of the MPEG-2 video standard main profile. Section 3.1 describes the MPEG-2 syntax. Section 3.2 presents the compression algorithm and discusses its input formats (section 3.2.1), the techniques used to achieve temporal (3.2.2) and spatial (3.2.3) decorrelation, quantization (3.2.4), and entropy coding (3.2.5).

### 3.1 The MPEG-2 syntax

The *MPEG-2 syntax* defines a *bitstream* composed of five hierarchical layers, delimited by the corresponding headers, and described as follows.

- The *sequence layer* defines global parameters, such as picture dimensions, frame rate, and transmission parameters. It comprises all the information for which these parameters are valid (for example, an entire movie).
- The *picture layer* comprises a single frame, defining parameters specific to that particular frame, such as picture type, and temporal reference.
- The *slice layer* enables data packetization in the bitstream. The slice header is always aligned on a byte boundary, constituting the lowest level entry point in the bitstream in case of a loss of synchronization. It comprises a set of contiguous macroblocks delimited by two slice layer headers.
- The *macroblock layer* is comprised of macroblocks. A macroblock is formed by six blocks and constitutes the processing unit for motion estimation/com-

pensation. Several parameters specific to each macroblock, such as the macroblock type and its motion vectors, are specified in its header.

- The *block layer* comprises 8x8 blocks of pixel data, and is the processing unit for the DCT.

An optional layer is also defined by the MPEG-2 syntax.

- The *Group of Pictures (GOP) layer* provides the capability of random access to any point of the sequence with a granularity of a pre-specified number of frames. For example, by setting the number of pictures in the GOP to ten, every tenth frame can be decoded without decoding the frames in between.

This hierarchical structure provides logical (random access, resynchronization, etc.) and signal processing (motion estimation/compensation, DCT, etc.) functionality with division of tasks between the different layers.

## 3.2 The MPEG-2 compression algorithm

The basic interframe-predictive coding structure, which will be from now on referred to as the *MPEG-2 encoder*, is based on motion estimation/compensation, the DCT, and entropy coding. The block diagrams of the MPEG-2 encoder and decoder are shown in figures 3.1 and 3.2.

### 3.2.1 Input formats

The input video signal for an encoder of the main level at the MPEG-2 main profile must conform to the 4:2:0 format of the CCIR 601 Recommendation [32] specifications. In particular, it must be represented in the Y, Cr, and Cb color space, where the Y component carries the luminance information, and the Cr and Cb components

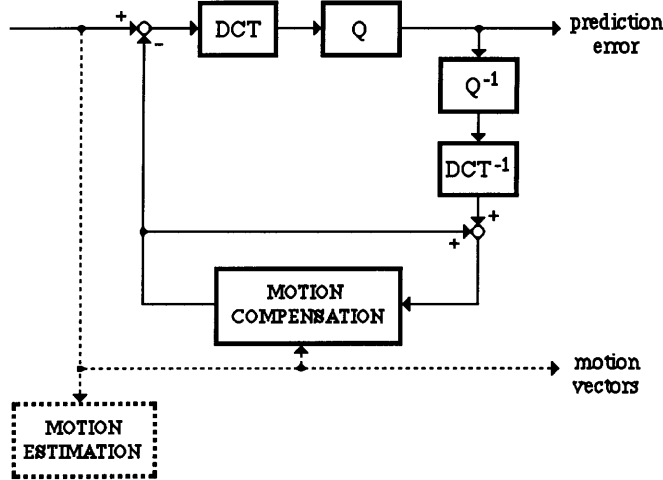


Figure 3.1: Block diagram of a MPEG-2 encoder.

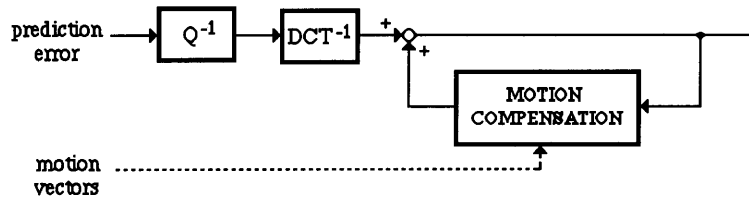


Figure 3.2: Block diagram of a MPEG-2 decoder.

convey the chrominance information. The CCIR 601 signal is composed of 60 fields/s interlaced and displayed at the rate of 30 frames/s<sup>2</sup>. Table 3.1 presents the main parameters of this input format. Several input formats that can be derived from the CCIR 601 input and are supported by MPEG-2 are described in [32]. For the 4:2:0 format, each of the chrominance components of the CCIR 601 signal is vertically decimated by a factor of two, originating a color representation of 240 lines by 360 pixels.

As mentioned above, all the processing performed by a MPEG-2 encoder is macroblock or block-based. Each input frame is split into 8x8 blocks. A macroblock is a set of four luminance blocks and two chrominance blocks, grouped as shown in figure 3.3.

<sup>2</sup>These numbers apply to the 525-lines system used in the U.S. Other countries use the 625-lines system, also supported in MPEG-2.

Table 3.1: Input format characteristics according to the CCIR 601 Recommendation.

Number of active lines	
luminance (Y)	480
chrominance (Cr, Cb)	480
Number of active pixels per line	
luminance (Y)	720
chrominance (Cr, Cb)	360
Frame aspect ratio (hor:ver)	4:3

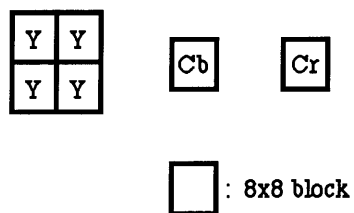


Figure 3.3: Macroblock structure.

### 3.2.2 Temporal decorrelation

Three types of pictures are defined by MPEG-2:

- intra pictures (*I-pictures*) are coded without reference to any other neighboring frames (i.e. without motion estimation/compensation), and are, therefore, decodable on their own;
- predicted pictures (*P-pictures*) are first motion compensated with reference to the most recently transmitted I or P-picture (forward prediction), and only the prediction error is coded;
- interpolated or bidirectional pictures (*B-pictures*) are motion compensated with reference to both a past and future I or P-picture.

Input frames are usually grouped into GOPs that always start with an I-picture and may contain any number of P and B-pictures (see figure 3.4), depending on the requirements of the application. The number of P and B-pictures is usually described by two parameters:  $N$ , the number of pictures in the GOP, and  $M$ , the number of B-pictures between consecutive P-pictures plus one.

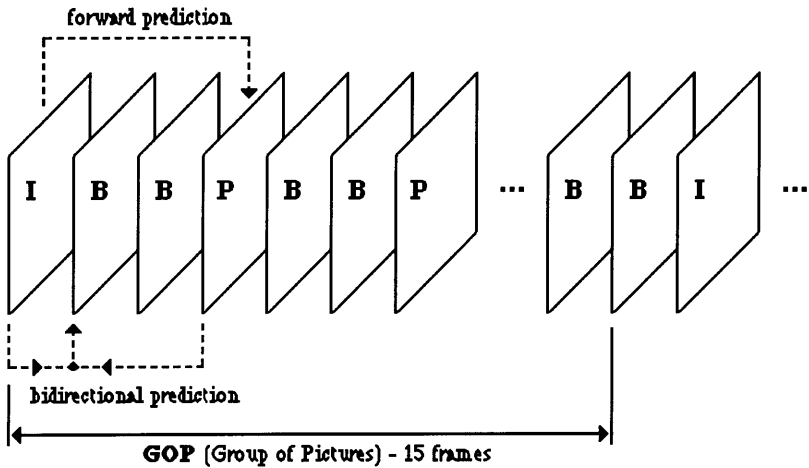


Figure 3.4: The GOP structure. In this example,  $N=15$  and  $M=3$ .

Since the coding of each B-picture requires motion compensation with reference to a future I or P-picture, the concept of *transmission order*, as opposed to *display order*, was introduced in MPEG. Display order is the one that corresponds to that of the input frames. For transmission, all the pictures that are used as a references for motion compensation of a B-picture are sent before that B-picture. An example of these orderings for the case of a 9-frames GOP ( $N = 9$ ) is:

display order:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	...

transmission order:

1	4	2	3	7	5	6	10	8	9	13	11	12	16	14	15	19	...
I	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	...

Temporal decorrelation is achieved in P and B-pictures by motion estimation/compensation. Motion estimation is performed only by the encoder, and as such is not subject to standardization. Some of its parameters are, however, common to motion compensation, which is both an encoder and decoder operation, and are defined by MPEG-2. These parameters are half-pixel accuracy for motion estimation/compensation, and the definition of the macroblock as the processing unit for motion estimation/compensation.

Two motion estimation/compensation modes are defined by MPEG-2 for both P and B-pictures: *field MC* and *frame MC*.

- In the field MC mode, there are two motion vectors associated with each macroblock. Macroblock rows belonging to the same field are motion compensated by the same vector. In this way, pixel rows from any of the two fields in the current image can be predicted from pixel rows from any of the two fields in the previous one.
- In the frame MC mode, there is only one motion vector, which is applied to all the pixel rows within the macroblock.

The reasoning behind the existence of two distinct modes is that, in areas of fast motion, the two fields (which are temporally spaced by 1/60 seconds) may require two different motion vectors for accurate prediction. However, when there is small or no motion at all, or with progressive material, the second vector is not needed and its transmission would imply unnecessary overhead.

In addition to these two modes, a third motion estimation/compensation mode is also allowed in P-pictures: *dual prime MC*. In this mode, only one vector  $v$  and two differentials  $d_1$  and  $d_2$  (restricted to values of 0 and  $\pm 1/2$ ) are transmitted by macroblock, but the prediction for the rows of each field is actually obtained using two displacement vectors, and a total of four motion vectors is emulated without the cost of significant increase in overhead. These four motion vectors are obtained from

$v$ ,  $d_1$ , and  $d_2$  as described by table 3.2.

Table 3.2: Motion vectors for dual prime MC.

reference field of reference frame	predicted field of predicted frame	motion vector
first	first	$v_{11} = v$
second	first	$v_{12} = \alpha v + d_1$
first	second	$v_{21} = \beta v + d_2$
second	second	$v_{22} = v$

The constants  $\alpha$  and  $\beta$  are defined to appropriately scale  $v$  according to the temporal distance between fields. Once the motion vectors are computed, the prediction for the lines of the first field is obtained by averaging the predictions associated to  $v_{11}$  and  $v_{12}$ , and that for the lines of the second field by averaging the predictions associated with  $v_{21}$  and  $v_{22}$ . In this way, dual prime MC tries to combine the higher flexibility of field MC with the reduced overhead inherent to frame MC to achieve an overall higher efficiency.

In addition to these different motion estimation/compensation modes, a macroblock from a P or B-picture can have one of several types.

- P-picture macroblocks can be coded as *intra*, if no motion compensation is performed, or *inter*, if there is motion compensation.
- B-picture macroblocks can also be classified in the same way. Additionally, if classified as *inter*, they can be:
  - *forward predicted* if only a previous frame is used for prediction;
  - *backward predicted* if only a frame from the future is used for prediction;
  - *interpolative predicted* if both a previous and a future frame are used for prediction.



In the interpolative case, the prediction macroblock is obtained by linear interpolation (typically, averaging) of the past and future macroblocks.

The decision criteria to choose the prediction mode is not standardized by MPEG-2. However, a common procedure is to compare the energy of the prediction error obtained with all the modes and select the mode which minimizes this energy. If this is still greater than the energy of the original macroblock, the intra mode is preferred.

### 3.2.3 Spatial decorrelation

The spatial correlation existent in the input blocks in I-pictures and in the prediction error residuals in P and B-pictures is minimized by applying a 2-D 8x8 DCT.

Two DCT coding modes are defined by MPEG-2: *frame DCT* and *field DCT*. These two modes differ in the ordering of pixel rows within a luminance macroblock (chrominance macroblocks use only the frame mode), as shown in figure 3.5. The motivation behind this is to perform the DCT on blocks with maximum correlation between rows in order to achieve higher coding efficiency. Whenever there is motion, adjacent rows of a block (that belong to different fields) have low vertical correlation, and the decorrelation provided by the DCT is higher if rows of the same field are grouped together. On the other hand, when there is no motion, the correlation is highest between adjacent rows and the frame mode is more efficient.

### 3.2.4 Quantization

The use of both motion compensation and DCT minimizes the correlation between samples of the original sequence of images. Neither of these techniques provides, however, compression to the desired bit-rate. In MPEG-2, this is achieved by the quantization and entropy coding stages, described in this and in the next section.

After DCT, intra and non-intra blocks present different characteristics.

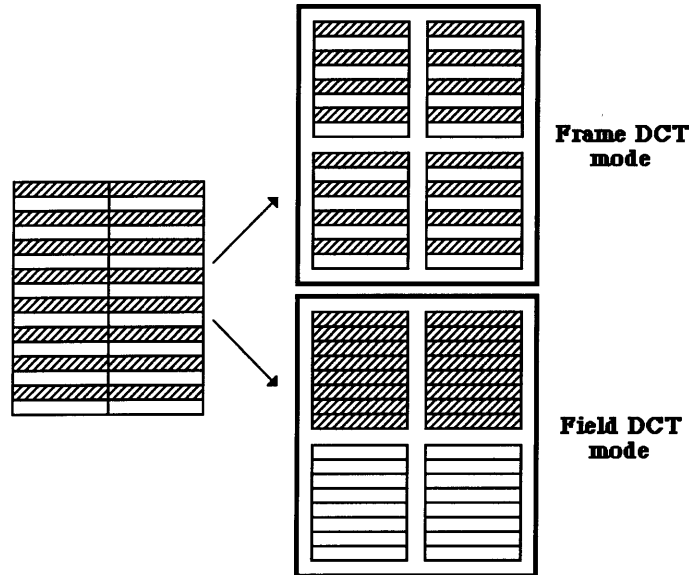


Figure 3.5: Luminance macroblock organization in frame and field DCT coding modes.

- *Intra blocks* have considerable low-frequency energy content. A fine quantization of its DC coefficient is, therefore, needed to avoid the “blocking effect”, described in section 2.3.4.
- *Non-intra blocks* have predominantly high-frequency energy content, and do not require any special treatment of the DC coefficient.

In MPEG-2, quantization is performed in two stages. A fixed quantization matrix, selected according to the block type, is first applied to the entire block. Then, uniform scalar quantization is performed on each of the block coefficients.

The reasoning behind the first stage is to explore the characteristics of the human visual system, which is less sensitive to quantization noise in regions of high-frequency content. For the same overall quantization noise, higher subjective quality can be obtained with a weighted distribution of this noise into different frequency ranges. In MPEG-2, this is achieved with quantization matrices that quantize more coarsely the high-frequency coefficients.

Two quantization matrices are defined: one for the quantization of intra mac-

roblocks, and another for the quantization of inter macroblocks. Typically, the intra quantization matrix is more slanted, originating coarser quantization of the high-frequencies; in contrast to the inter one, where a more uniform quantization is performed. This is due to the fact that, since the prediction-error signal is by nature high-pass, the high-frequency information is more important in predictive frames than in intra ones. Also, to avoid the “blocking effect” above mentioned, no intra-quantization matrix value is defined for the DC coefficient.

The reasoning behind the second quantization stage is to adapt the quantization to the varying characteristics of the input signal. Associated with each macroblock, there is a *macroblock quantizer* (*mquant*) which is applied to all the coefficients in the macroblock. The only exception to this rule are the DC coefficients of intra macroblocks, which are always quantized with 8 bits.

The variation of quantization step-size, provided by the *mquant*, allows the control of the output bit-rate. In fact, *mquant* is the only mechanism within MPEG-2 that provides rate-control functionality, i.e. the necessary adaptation between the bit rate generated by the encoder and the one supported by the transmission channel.

### 3.2.5 Entropy coding

After quantization of the DCT coefficients, these are ordered according to one of the zig-zag scanning patterns shown in figure 3.6: the pattern a) is generally used for progressive sources; while the pattern b) is more suited for interlaced material. The goal of this coefficient ordering is to maximize the length of runs of zeros originated by quantization.

The resulting array of coefficients is coded using a run-length/amplitude scheme. Starting from the first position in the array, the encoder counts the number of consecutive zeros until the next non-zero coefficient. The zero count (*run-length*) and the amplitude of this coefficient are Huffman-coded and transmitted.

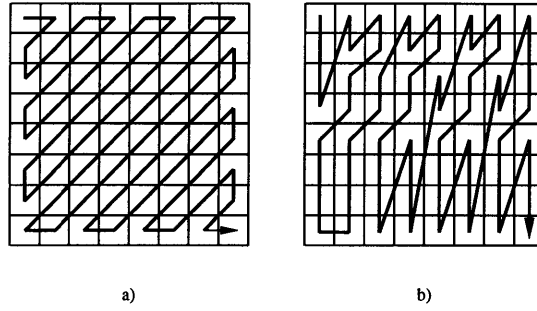


Figure 3.6: Zig-zag scan patterns for the 8x8 transform coefficient blocks.

Apart from the DCT data, most of the side information originated in the encoding process is also Huffman coded before transmission. In particular, motion vectors and DC coefficients of intra macroblocks are DPCM coded to remove some remaining correlation, before the entropy coding stage.

# Chapter 4

## The 2-stage coding algorithm

This thesis addresses the problem of efficient coding of video material recorded off-line, i.e. before the start of the encoding procedure, in a way such that it can be made available to different users according to their different qualities requirements.

This type of material is very common, particularly in television broadcast applications where a significant portion of the transmission time is allocated to previously recorded footage, such as movies, documentaries, etc. In addition to being common, these applications present specific characteristics that can be exploited to achieve high-coding efficiency. In particular, they are suited to the use of *multiple-pass encoding* techniques where the entire footage (or parts of it) is first globally analyzed, and the characteristics thus revealed used in the subsequent encoding stages.

To understand more clearly the specific characteristics of these type of applications, consider the following scenario: a previously stored sequence of images (for example, a movie) is to be transmitted for different users. Suppose that the footage is stored digitally (in some kind of digital equivalent to a VCR tape), and the users are equipped with decoders of different complexities (capable of decoding different bit rates and, therefore, providing different image qualities).

A simple solution to this problem would be to use various MPEG-2 encoders at

different bit rates, working in parallel. Obviously, this approach would require the multiplication of the encoding power by the number of rates desired. Also, enough storage space should be available to keep the entire raw footage. Clearly, this is not a good solution since, by pre-processing the image data, this storage space and complexity requirements can be reduced.

This thesis presents an algorithm capable of achieving reduced storage and complexity with increased coding efficiency through the application of a 2-stage encoding procedure, whose block diagram is shown in figure 4.1.

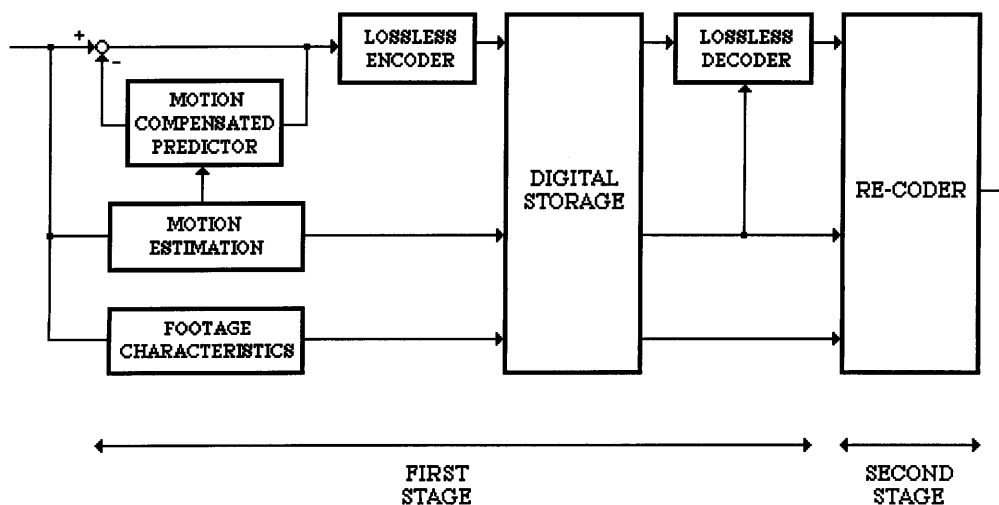


Figure 4.1: Block diagram of the 2-stage coding scheme.

In this coding scheme, the first stage has a double purpose. First, it is responsible for the analysis of the footage as a whole, or at least by considering a set of significant frames at each time, in order to obtain characteristics such as local activity, the location of frames that are particularly hard to code (like scene changes), etc. Second, it provides an intermediate representation of the video data which requires reduced storage capacity, while maintaining the data integrity.

The second stage uses the footage characteristics collected in the first one to achieve high-coding efficiency. This stage provides larger compression ratios at the cost of loss of data integrity.

The functionalities associated with each stage are implemented through the application of the image coding techniques discussed in chapters 2 and 3. In the first stage, the input sequence is first motion compensated, and the resulting motion-vector field stored. The prediction-error signal is then losslessly encoded providing, together with the motion-vector field, the intermediate representation referred above. Global information about the footage, which will be used in the second stage, is also acquired in this stage. The second stage consists of a bank of MPEG-2 based encoders (re-coders), operating at different bit rates, and using the motion-vector field previously stored as well as the enhancements provided by the prior knowledge of the footage characteristics.

The following sections describe in more detail each of the different blocks that compose the block diagram of figure 4.1. Section 4.1 presents the motion-estimation procedure used in the first stage. Section 4.2 discusses the lossless encoding of the prediction error signal. Section 4.3 describes the methods used to extract the footage characteristics. Finally, section 4.4 presents the re-coder stage.

## 4.1 Motion estimation

It was seen in section 2.1.2 that motion-compensated predictive coding is a powerful technique for the minimization of the temporal redundancy existent in a video sequence. This technique was, therefore, chosen to achieve temporal decorrelation; and, since compatibility with MPEG-2 was desired, the block-matching procedure also described in section 2.1.2 was selected to perform motion estimation, based on the MPEG-2 GOP structure with I, P, and B-frames.

Motion estimation can be performed between original images, i.e. before coding takes place, or between the input original image and the previously coded one. Using the originals, the estimation is more accurate, originating motion vectors that are closer to the real motion of the input sequence. Using the previously coded image,

the estimate is less accurate, resulting in the appearance of motion vectors that do not fully recreate the original motion, but the prediction error is minimized.

In general, the two approaches originate roughly equal results, and the best performance depends on the application. For example, motion-compensated interpolation requires an accurate reproduction of the motion-vector field and the use of the original images; while, for coding applications, it is more important to minimize the energy of the prediction-error signal. In this case, it is more appropriate to use the previous reconstructed frame for prediction. However, by using the originals, the motion vectors will be the same, independently of the transmission bit rate and coding quality.

In the case of a 2-stage encoder, this independence of the bit rate makes it possible to perform the motion estimation once, in the first stage, and use the same motion vectors for all the different rate encoders of the second stage. Since motion estimation is typically the more expensive encoding step, this provides extremely significant computational savings when compared to the use of various MPEG-2 coders in parallel.

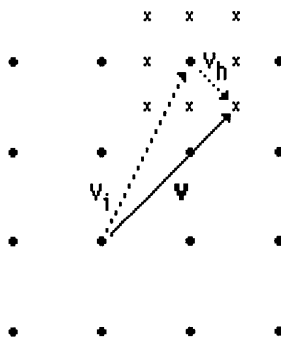


Figure 4.2: Two-step motion estimation.  $v_i$  and  $v_h$  are the integer and half-pel motion vectors, respectively, and  $v$  is the resulting motion vector. The half-pel samples (x) are obtained by linear interpolation of the neighboring pixels.

In this work, the motion estimation was implemented with half-pixel accuracy. In order to reduce the computational load, the two-step procedure exemplified in figure 4.2 was employed. First, the best match with integer-pel accuracy is found by the



exhaustive search on a window of pre-specified size. Then, the half-pel motion vector refinement is found through a search in a window of  $\pm 1/2$  pixel in each dimension, centered on the position of that best integer-pel match. The samples associated with half-pel displacements are obtained by linear interpolation.

## 4.2 Lossless and nearly-lossless coding

It was already mentioned that the simpler solution to the problem addressed by this thesis would be to start from a digital representation of the data, and apply several coders in parallel. It was also seen that this would result in an unnecessary replication of the computational resources required and, therefore, totally inefficient.

In addition, it would require the storage of the entire digital footage without any compression, which would also lead to inefficient use of the available storage resources. This aspect is particularly important due to the large amount of memory required to store digital video sequences of considerable duration. For example, the storage of a 2 hours-long digital movie with 30 frames/s, CCIR 601 resolution (480 lines of 720 pixels), and 24 bits/pixel would require a storage capacity of approximately 224 Gbytes.

Since the bit rates and coding qualities provided by the second stage are not known in advance and should be set by the user, the intermediate representation should be able to allow up to perfect reconstruction. Therefore, the coding techniques applied in the first stage should belong to the class of lossless encoding techniques.

The lossless coder implemented is based on predictive coding. As in MPEG-2, the coded frames are divided in three groups according to the type of motion-compensated prediction used for their encoding: I, P, and B-frames. However, a significant difference to MPEG-2 is that, in a lossless scheme, the DCT cannot be used for spatial decorrelation due to the requirement of finite-arithmetic precision characteristic for its practical implementation. Consequently, a different approach

was required for the encoding of I-frames, and the motion-compensated prediction error in P and B-frames.

In I-frames, and since the DCT is not used, there is no need for a block or even a macroblock layer. The smallest processing unit considered is a picture slice of 16 by 720 pixels. This processing unit provides a trade-off between coding efficiency and capacity to recover from errors originated by the storage media. The lossless coding algorithm used is based on intraframe prediction using the predictor shown in figure 4.3.

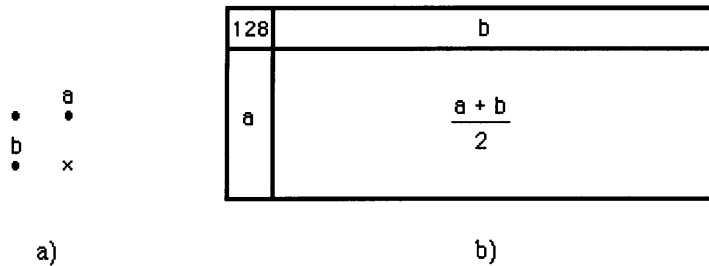


Figure 4.3: Intraframe predictor used in I-frames. a) Basic predictor structure:  $x$  is the pixel to be coded, and  $a$  and  $b$  are previously coded pixels. b) Predictor used as a function of location in the slice.

With this predictor, for a given pixel  $x$  to be coded, designating by  $b$  the previous pixel in the same line and by  $a$  the pixel immediately above, the prediction  $p$  of  $x$  is given, outside the slice boundaries, by

$$p = \frac{a + b}{2}. \quad (4.1)$$

In the first line of the slice,  $p = b$ ; and, in the first column of the slice,  $p = a$ . The predictor is initialized in the first pixel of the slice with the value of  $p = 128$ . In this way, the propagation of any storage errors is limited to one slice.

This predictor was chosen according to the results presented in [33], where the performance of intraframe encoding with several different third-order predictors, using contiguous pixels from the same and previous lines, was compared. As it was

already referred in section 2.1.2, the use of predictors of order greater than three does not provide any significant coding gain, and so such types of predictors were not considered.

The prediction error  $e_p$ , given by

$$e_p = x - p, \quad (4.2)$$

is Huffman-coded using the table provided by MPEG-2 [29] for the predictive encoding of the DCT DC-coefficients in intraframes.

In P and B-frames, and due to the block-based nature of the motion-estimation procedure used (block matching), the smallest processing unit of the lossless encoder is a block of 16 by 16 pixels. For each block, a spatial decorrelation operation similar to that used at the slice level on I-frames is applied to the temporal prediction error signal. Those blocks for which the motion estimation is not efficient are classified as intrablocks and, for these blocks, the temporal decorrelation operation is applied to the image pixels. The prediction error residuals are encoded with the Huffman table referred above.

The requirement of preserving the data integrity and the use of lossless coding techniques limits the achievable compression ratios to values around 2:1. Due to the massive storage requirements referred above, this performance boundary can result in insufficient compression. To achieve higher compression ratios, an alternative approach, based on MPEG-2, using the DCT and high-rate quantization was also considered. This nearly-lossless mode is acceptable if the distortions introduced are below the threshold of visibility of the human eye, resulting in reconstructed images that are visually indistinguishable from the originals. In this case, the coding performance will be satisfactory for the greater majority of the common users.

## 4.3 Footage characteristics

In addition to providing a losslessly compressed intermediate representation of the digital data, the first encoding stage performs two functions that are useful for an efficient operation of the second encoding stage. The first of these functions is the computation of the motion-vector field that, as seen above, greatly simplifies the implementation of each of the second stage re-coders. The second function consists in the gathering of information about the input footage that is used to increase the efficiency of the second stage.

As discussed in chapter 2, image coding techniques are based on statistical models because the content of the data to compress is not known in advance and is, therefore, best modeled as a stochastic process. Although this still remains true in the case of the 2-stage encoder here described (otherwise there would be no real transmission of information, at least in the sense of its definition by Shannon), it is possible to extract, during the first stage, global information that characterizes the footage and to tailor the encoding of the second stage according to this information in order to achieve higher efficiency.

In the 2-stage encoder implemented, this tailoring of the encoding of the second stage to the footage characteristics is obtained through the following mechanisms:

- initialization of a new GOP in the location of scene changes or frames that are particularly difficult to code;
- the use of preload coding;
- an improved rate-control based on the bit distribution associated with the lossless/nearly lossless coding stage;

which will be discussed in detail in sections 4.3.1, 4.3.2, and 4.3.3, respectively.

### 4.3.1 Processing of hard to code frames

A *hard to code frame* is defined in the present context as a frame for which the prediction fails, originating a prediction-error signal with large energy that is difficult to encode. The reason for a frame to belong to this category is the result of a mismatch between the underlying coding model associated with the coding algorithm, which assumes mainly translational motion from frame to frame, and the content of the input sequence. There are several different causes which can lead to this mismatch, namely large areas of object newly revealed or occluded, the existence of strong non-translational motion, or simply the occurrence of a scene change.

Whenever a hard to code frame occurs, the use of motion-compensated prediction results in both large prediction errors and a highly non-smooth motion-vector fields. The possibility of encoding blocks of P and B-frames as intra blocks attenuates the effect of the large prediction errors. The lack of smoothness of the motion-vector fields remains, however, a problem and leads in general to a poor performance than the obtained if the entire frame were coded as an I-frame from the beginning.

In addition, since the GOP structure implies a periodicity for I-frames, it is likely that an I-frame occurs in the vicinity of this hard to code frame. Since I-frames are expensive, this would result in a bad distribution of I-frames and, consequently, a decrease in coding efficiency. This problem can be minimized by classifying each hard to code frame as an I-frame and, simultaneously, re-initializing a new GOP.

The “degree of coding difficulty” of a P or B-frame is measured, during the motion estimation operation, using the following algorithm.

- For each macroblock, the energy of the motion-compensated prediction error is compared with that of the original pixel amplitudes in the macroblock. If the prediction error has smaller energy, the macroblock is classified as inter; otherwise, it is classified as intra.
- If the number of intra macroblocks in the frame is greater than the number of

inter macroblocks:

- in the case of a P-frame, this is coded as intraframe, and a new GOP is started;
  - in the case of a B-frame, this is coded as interframe (as would be usual) but using only backward prediction, the next P-frame is coded as intraframe, and a new GOP is then started.
- Otherwise, the frame is not considered as hard to code, and the usual procedures are used.

Obviously, if a hard to code frame coincides with an I-frame, no special action takes place.

This procedure provides a reliable detector of hard to code frames without any significant computational overhead since the macroblock classification would have to take place anyway. The only possible problem originated by this approach would be the existence of several hard to code frames in succession, resulting in several GOP re-initializations and, consequently, an increase in overhead and in the complexity of the rate control. Such a situation could be prevented by the introduction of a lower limit in GOP size. However, and since it was never verified in the several simulations performed, this limitation was not introduced.

### **4.3.2 Preload coding**

As mentioned in the previous section, the underlying coding model associated with the coding algorithm can be inappropriate in some occasions, originating frames that are hard to code. In result, to maintain a constant coding quality, an encoding algorithm should be capable of allocating a variable number of bits per frame depending on the coding complexity. In particular, hard to code frames require a much larger number of bits than average frames. It would be desirable to distribute that large number of

bits by a large number of surrounding frames in order to avoid degradation in quality imposed by the limitation in the number of bits per frame associated with a fixed-rate channel.

Although this can, in part, be done with a rate-control mechanism using channel buffers, the hard to code frames always present a problem due to the large ratio between the number of bits required by them and that required by common frames. The distribution of this peak would require a large number of frames and, therefore, a large channel buffer, introducing large encoding delays. In practice, such delays are not tolerable, and common decoders avoid the problem by allocating less bits to these hard to code frames, allowing some quality degradation and relying on the fact that the human eye is in general less sensitive to distortion in them. Such a solution fails, however, not only to maintain the quality during these active periods, but also originates poor prediction for the frames that follow them.

An alternative solution to this problem, only possible to implement with a two-pass encoder, is the use of *preload coding*. Preload coding consists in transmitting, during low activity periods or even prior to the beginning of transmission, stills corresponding to high activity periods. These stills are used by the decoder as the basis for prediction when these high-activity periods occur, resulting in increased coding quality. This technique is particularly suited to use with the method of processing hard to code frames described in section 4.3.1 since, with that method, these frames are coded in intramode, i.e. can be coded on their own.

An example of preload coding is represented in figure 4.4. In this case, a scene change occurs in the tenth frame. If the first and tenth frame are coded (as intraframes) and transmitted before of the start of the “regular transmission”, they can be used in the decoder for more accurate prediction.

In the 2-stage coder, preload coding was implemented by creating an additional bitstream of reduced size, incorporating all the hard to code frames. This bitstream should be downloaded by the user before the start of the transmission, and stored

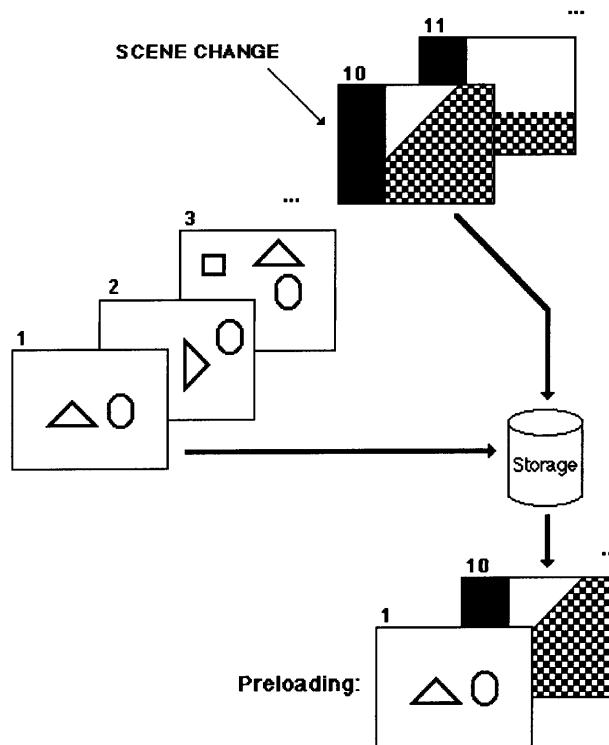


Figure 4.4: Example of preload coding.

locally in disk. As the transmission progresses, the required hard to code frames are decompressed into RAM memory, and used as needed.

This method requires the existence of memory in the decoder in the form of hard disk. The memory requirements are, however, within reasonable limits as pointed out by some simple calculations indicating that, for a 10 Mbits/s rate, a 80 Mbytes disk unit would be capable to store approximately 2,000 frames. Although no attempt was made to encode an entire movie, this number of frames seems capable of holding at least the majority of scene changes. The downloading time for such a bitstream would be of approximately one minute.

An alternative solution, avoiding the requirement of a dedicated hard-disk, consists in the use of a CD-ROM for the storage of the additional bitstream. This CD-ROM could be provided to the user in advance of the broadcast, with the advantage that a much larger number of frames could be incorporated in the preloaded



bitstream.

### 4.3.3 Improved rate-control

It was already seen that different input frames have different amounts of activity. To maintain a constant encoding quality, an efficient coding algorithm should have the capability of assigning different number of bits to different frames. This is particularly true in the case of a MPEG-2 encoder due to the different type of encoding applied to distinct frames.

The ability to achieve a variable bit-allocation can only be obtained through the use of a channel buffer and a rate-control algorithm that, by introducing some feedback from the state of this buffer into the coding procedure, is able to maintain the instantaneous bit-rate within the bounds imposed by the transmission capacity.

As referred in section 3.2.4, rate control is performed in a MPEG-2 encoder by varying the value of the macroblock quantizer (mquant). A typical rate-control algorithm starts by defining a *target number of bits* for each frame, and then dynamically updates the macroblock quantizer in a way such that the number of bits generated to encode the frame is as close to the target as possible. For efficient performance, the mquant should be updated taking into account not only the state of the transmission buffer, but also the local activity of the picture.

For a standard MPEG-2 encoder, and since no information about the footage is known in advance, the target number of bits per frame is set in a linear fashion, i.e. attributing to each frame of the same type (I, P, or B) the same number of bits. Obviously, this kind of distribution cannot take into account local variations of activity and can lead to inefficient performance. For example, on a GOP where strong motion occurs during the first frames and then nothing happens during the remaining frames, a linear distribution of bits as described above is clearly sub-optimal.

On a 2-stage encoder, the knowledge of the footage characteristics acquired during

the first encoding stage (which, being lossless, does not require any rate control) can be used to achieve an improved bit-allocation in the second stage. In fact, the lossless encoding can be considered as a first pass where information about the number of bits necessary to code each frame (based on the particular characteristics of this frame) is collected.

In principle, the number of bits necessary to losslessly encode each frame is proportional to the difficulty associated with that frame. If the target distribution of the second stage is made proportional to the natural distribution of bits of the first stage, it will be possible to achieve a smoother quality variation from frame to frame than that achievable with a linear target strategy.

An example of such a linear strategy is the mechanism for the distribution of bits between frames defined in the MPEG-2 Test Model [29]. The *target number of bits* for the next picture (I, P, and B) in the GOP is computed, using only the available knowledge about previously transmitted frames, by the following equations:

$$T_I = \frac{1}{1 + \frac{N_P X_P}{X_I K_P} + \frac{N_B X_P}{X_I K_B}} \cdot R \quad (4.3)$$

$$T_P = \frac{1}{N_P + \frac{N_B K_P X_B}{X_P K_B}} \cdot R \quad (4.4)$$

$$T_B = \frac{1}{N_B + \frac{N_P K_B X_P}{X_B K_P}} \cdot R, \quad (4.5)$$

where  $R$  is the remaining number of bits in the GOP.  $X_I$ ,  $X_P$ , and  $X_B$  are *complexity measures* defined by:

$$X_I = S_I Q_I, \quad X_P = S_P Q_P, \quad X_B = S_B Q_B, \quad (4.6)$$

where  $S_I$ ,  $S_P$ , and  $S_B$  are the number of bits generated by encoding the previous respective frame; and  $Q_I$ ,  $Q_P$ , and  $Q_B$  are the average quantization step-size for all the macroblocks in that frame.  $K_P$  and  $K_B$  are constants dependent on the

quantization matrices.  $R$  is defined, at the beginning of the GOP, as the average number of bits per frame (obtained by dividing the bit rate by the frame rate) times the number of frames in the GOP, and is updated after the encoding of each picture by:

$$R = R - S_{I,P,B}, \quad (4.7)$$

where  $S_{I,P,B}$  is the number of bits generated in the I, P, or B picture just encoded. Finally,  $N_P$  and  $N_B$  are the number of P and B-frames remaining in the current GOP.

Clearly, this strategy assumes an equal distribution of bits by the future frames of the same type in the GOP, introducing only weighting factors when considering the influence of frames of different types. This is a wise decision since, in the absence of information about future frames, we might as well assume an equal distribution of bits between frames of the same type.

It is not wise, however, in the case of a 2-stage encoder since here the first stage makes available information about future frames, which can be exploited to achieve an improved rate-control in the second stage. Such a rate control was implemented in this work by considering that, for a given bit rate, a nearly constant image quality can be achieved if the number of bits allocated to each frame of the same type is proportional to the number of bits required for its lossless encoding. The reasoning behind this assumption is that the number of bits necessary to encode each frame in the first stage is a measure of the difficulty associated with that frame; and, for constant quality, the target number of bits allocated to each frame in the second stage should be proportional to this difficulty.

The 2-stage rate control was implemented as follows. Considering  $frm\_bits[k]$  as the number of bits required to encode the frame  $k$  during the first stage,  $gop\_bits$  the number of bits spent on the lossless encoding of the GOP to which frame  $K$  belongs, and  $cum\_bits$  the number of bits required to lossless encode all the frames in the GOP up to frame  $k$ , the target number of bits for the second-stage encoding of frame  $k$  (I,

P, or B) is determined by:

$$T_{I,P,B}[k] = w_{I,P,B} \cdot \frac{frm\_bits[k]}{gop\_bits - cum\_bits} \cdot R \quad (4.8)$$

where  $R$  is the number of remaining GOP bits as defined above, and  $w_{I,P,B}$  are weighting factors introduced to compensate the effect of the different quantization matrices applied to different picture types<sup>1</sup>:

$$w_I = 1.6, \quad w_P = 1.3, \quad w_B = 1.0. \quad (4.9)$$

The idea behind equation 4.8 is that the target  $T_{I,P,B}[k]$  must be proportional to  $frm\_bits[k]$ , and the constant of proportionality the ratio between the remaining bits in the GOP of the lossy encoder (second stage) and that of the lossless encoder (first stage).

This target strategy is capable of providing a totally non-linear bit allocation if such is required by the footage characteristics. For example, if the complexity of the first frame is such that it required 50% of the total number of the GOP bits in the first stage, the target set for the second-stage encoding will also be 50% of the available GOP bits. If a linear strategy, such as the described by equations 4.3 - 4.5, were applied, this frame would receive approximately 7% of the GOP bits (assuming a 15-frames GOP). On the other hand, if a linear allocation is required by the footage characteristics, it can also be easily achieved with the new rate-control algorithm.

Once the targets are determined, the computation of the actual macroblock quantizers (mquants) is carried out according to the equations provided in the MPEG-2 Test Model, taking into account both the fullness of the transmission buffer and the local picture activity.

---

<sup>1</sup> The values of these weighting factors are optimized for rates bellow 10 Mbit/s. A different set of values may be required for higher rates.

## 4.4 Re-coder stage

The re-coder stage is constituted by a bank of MPEG-2 based encoders, operating at different bit-rates, and using the previously stored motion-vector field and the enhancements provided by the knowledge of the footage characteristics acquired in the first stage. The block diagram of one of the re-coders used in this stage is shown in figure 4.5.

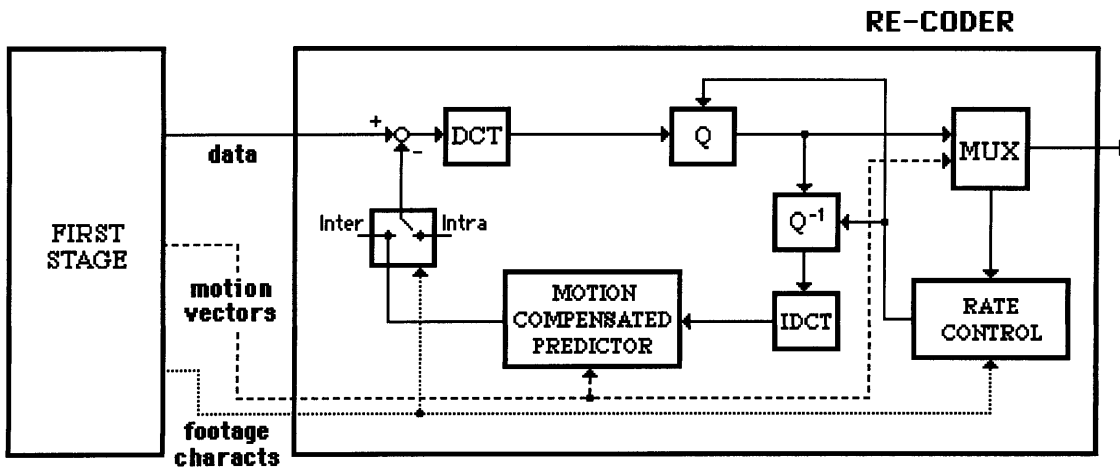


Figure 4.5: Block diagram of the re-coder.

As in MPEG-2, each re-coder uses interframe motion-compensation in the temporal domain (in the case of P and B-frames), and the DCT followed by uniform scalar quantization and variable-length coding in the spatial domain (as described in detail in chapter 3). The motion-vector field already computed in the first stage is used by the motion-compensated predictor and multiplexed into the bitstream originated by the second stage. The footage characteristics, also collected during the first stage, are used to improve the encoding of “hard to code frames” (as discussed in section 4.3.1), in the formatting of the preloaded bitstream (section 4.3.2), and in the rate-control mechanism applied in the re-coder (section 4.3.3).

In a practical implementation of this type of 2-stage encoding, two different configurations can be considered. These configurations differ only on the second stage,

which can be totally distributed or concentrated in a unique broadcast unit.

In the distributed configuration, one re-coder is allocated to each user. In this way, the user will have complete control over the transmission bit-rate. Since the motion estimation and part of the decision making involved in the encoding process are performed on the first stage, the implementation of each re-coder will be only slightly more expensive than that of a standard MPEG-2 decoder, making this solution economically viable.

In the centralized configuration, a unique broadcast unit composed by a bank of re-coders covering a pre-specified range of transmission bit-rates feeds all the users. In this case, the user will only have the possibility of choosing one of the pre-specified rates, but the system will be less expensive.

# Chapter 5

## Simulation results

This chapter presents simulation results of the experiments performed to determine the efficiency of the 2-stage encoder, described in the previous chapter, and some of its implementation parameters. These experiments were performed with several CCIR 601, 30 frames/s, input sequences.

Since there was no intention to optimize the algorithm to handle interlaced inputs, most of the experiments were performed on three progressive sequences from the “Sharky’s machine” movie. These sequences are representative of several types of scenes, ranging from those composed by a few human figures moving slowly until those composed by a vast number of objects subject to different types of fast motion.

Interlaced sequences were also used whenever it was thought that interlace might affect significantly the coding performance. This situation was not very frequent because, although no new features to handle interlace were introduced in the algorithm, the ones already existing in MPEG-2 were maintained in the 2-stage encoder.

In all the experiments run, the simulation results were analyzed both subjectively (by comparing original and reconstructed images) and objectively. The distortion metric chosen for objective evaluation was the *Mean Squared Error (MSE)* defined

by

$$MSE = \frac{1}{N} \sum_{i,j \in R} (x_{ij} - \hat{x}_{ij})^2, \quad (5.1)$$

where  $N$  is the number of image pixels,  $i$  and  $j$  the pixel coordinates,  $R$  the image's region of support,  $x_{ij}$  the amplitude of the pixel  $ij$  in the original image, and  $\hat{x}_{ij}$  the correspondent amplitude in the reconstructed image. Objective results are also presented in the form of the usual *Signal to Noise Ratio (SNR)* defined as

$$SNR = 10 \log \frac{255^2}{MSE}. \quad (5.2)$$

The performance of the first encoding stage is analyzed in section 5.1. Section 5.2 presents the results of the experiments performed to optimize the efficiency of the motion estimator implemented. The efficiency of the features introduced in the second encoding stage (processing of “hard to code frames”, preload coding, and improved rate control) is discussed in section 5.3. Finally, section 5.4 presents an overall comparison between the efficiency of the 2-stage encoder and that of a standard MPEG-2 encoder.

## 5.1 First-stage encoding experiments

The bit rate required for the storage, without any compression, of a CCIR 601 4:2:0 input sequence (with 704 by 480 pixels) is

$$R_{CCIR\_601} = 704 \times 480 \times 1.5 \times 30 = 121.65 \text{ Mbit/s}.$$

To evaluate the performance of the lossless encoding algorithm, this algorithm was run on several 100 frames input sequences, originating the results presented in



table 5.1. The compression ratio  $comp$ , defined by

$$comp = \frac{R_{CCIR.601}}{R},$$

where  $R$  is the bit rate achieved with the lossless algorithm, was chosen to measure the performance obtained in the first stage.

Table 5.1: Performance of the lossless encoding algorithm for different input sequences.

Sequence	R (Mbit/s)	comp
Sharky (frame 0 to 100)	65.1	1.87
Sharky (frame 100 to 200)	66.3	1.83
Sharky (frame 200 to 300)	64.7	1.88
Mobile & Calendar	68.2	1.78
Flower Garden	68.4	1.77

The efficiency of the lossless encoder is approximately constant for the three progressive “Sharky” sequences, and slightly inferior for the remaining two interlaced sequences. This is a consequence of the lack of optimization of the spatial predictor to handle interlaced inputs. The decrease in efficiency is, however, small (approximately only 4.5 %); and, in average, the lossless encoder achieves a compression ratio of

$$comp_{av} = 1.826.$$

This value could be improved by further optimization of certain encoding parameters, such as the replacement of the Huffman coder by a slightly more powerful arithmetic coder, or the introduction of more elaborated spatial predictors (for example, an adaptive structure where each predictor is selected based on the local activity). However, the gains to be obtained with these improvements would not be substantial since they would probably not exceed 5 to 10%, leading to the empiric bound of 2:1 characteristic of the best known lossless encoding techniques for the compression of

noiseless data, such as computer files, text, etc.

However, even if this bound were satisfied, the resulting compression would be insufficient for several applications. For example, with 2:1 compression, a 2 hour-long movie would still require approximately 55 Gbytes of storage. Significantly larger compression ratios cannot be obtained with lossless coding, and it is necessary to allow some loss in the data integrity to achieve them.

As referred in section 4.2, an acceptable solution for this problem is to use MPEG-2 encoding with high-rate quantization (nearly-lossless encoding) in the first stage, and assuring that the distortion introduced is below the threshold of visibility of the human eye.

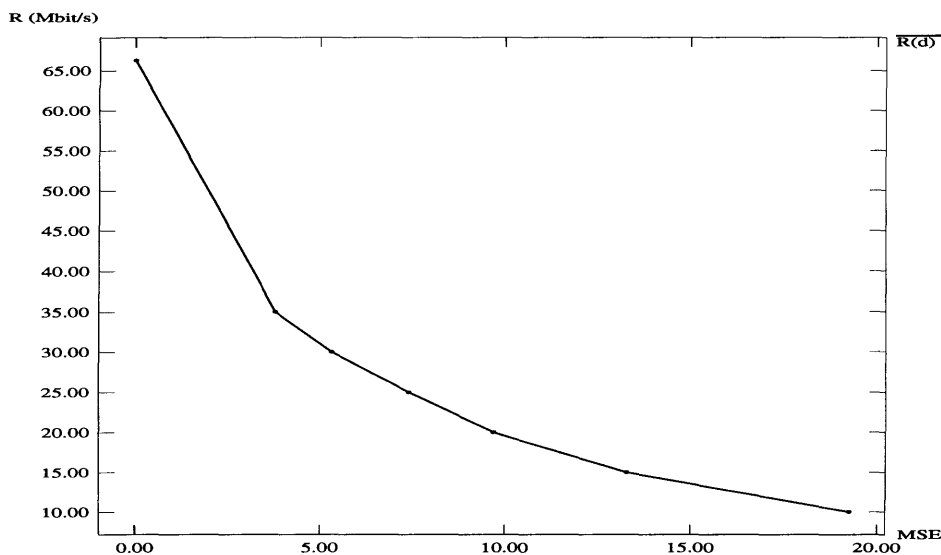


Figure 5.1: Rate-distortion curve for MPEG-2 with high-rate quantization.

An experiment was performed to determine this threshold, and correspondent bit rate, by running the MPEG-2 encoder at different bit rates and observing the associated degradation in picture quality. The rate-distortion curve of figure 5.1 was obtained as a result of this experiment.

Figures 5.2 and 5.3 present two reconstructed frames of the “Sharky” sequence and corresponding coding errors obtained by running the MPEG-2 coder at 10, 15, and

20 Mbit/s. These two frames are those for which the objective coding performance (MSE) is the lowest among all the 300 frames in sequence. This is, therefore, a worst case situation for performance analysis.

It was observed that, even on a display device in still mode, at 15 Mbit/s, the distortion introduced by the loss of data integrity is not noticeable, i.e. the subjective quality of the reconstructed image is indistinguishable from that of the original. At this rate, the storage requirements would be reduced to less than a quarter of those associated with the lossless encoder.

Therefore, the main conclusion provided by this experiment is that a nearly-lossless MPEG-2 mode at or above 15 Mbit/s is an efficient alternative to lossless encoding when the compression ratio obtained with the latter is considered insufficient.

## 5.2 Motion estimation experiments

It was pointed out in section 4.1 that the choice of a specific type of pictures (original or reconstructed) for prediction affects both the coding efficiency and the implementation complexity of the second-stage re-coders. Typically, the use of reconstructed pictures leads to some improvement in efficiency at the cost of a slight increase in complexity.

To evaluate the importance of using reconstructed frames for prediction, two different solutions were implemented. In the first, the entire motion estimation was performed using original frames. In the second, the integer-pel motion vectors were computed using original frames, but the half-pel refinements were computed using reconstructed frames for prediction.

In the first experiment, all the computations are carried in the first stage, imposing no additional computational load to the second stage; while, in the second experiment, the half-pel refinements are re-computed in the second stage re-coders. Table 5.2



Figure 5.2: MPEG-2 with high-rate quantization. Left: reconstructed images. Right: coding errors magnified by 10. Top: at 20 Mbit/s, center: at 15 Mbit/s, bottom: at 10 Mbit/s.



Figure 5.3: MPEG-2 with high-rate quantization. Left: reconstructed images. Right: coding errors magnified by 10. Top: at 20 Mbit/s, center: at 15 Mbit/s, bottom: at 10 Mbit/s.

presents the results of the two experiments for different bit rates (3, 4, and 8 Mbit/s), and using 100 frames of the “Sharky” sequence.

Table 5.2: SNR obtained for different motion estimation approaches. In experiment #1, the entire motion estimation was performed using original frames. In experiment #2, the half-pel refinements were computed using reconstructed frames.

	Experiment #1	Experiment #2
R (Mbit/s)	SNR (dB)	SNR (dB)
3	30.58	31.12
4	31.73	32.19
8	34.53	34.70

For all the three bit rates, the SNR is always greater if the half-pel refinements are computed using the reconstructed frames for prediction. It can be seen from the table that the gain obtained with this solution is greater for low bit-rates, which makes sense since, at high bit-rates, the reconstructed frames have high quality and the two solutions will lead to similar results.

Since the computational load of the half-pel refinements is reduced due to the small size of the search window ( $\pm 1/2$  pixel in each dimension) associated with it, the complexity overhead thus imposed to each re-coder of the second stage is not very significant. Therefore, given the gain of about 0.5 dB that it provides at low bit-rates, this approach was chosen for the implementation of the 2-stage encoder.

## 5.3 Second-stage encoding experiments

### 5.3.1 Processing of hard to code frames and preload coding

In order to evaluate the performance gain provided by the special processing of “hard to code frames” (described in section 4.3.1) and preload coding (section 4.3.2), both

based on the footage characteristics collected in the first stage, the 2-stage encoder incorporating these two algorithmic features was compared with a standard MPEG-2 encoder.

Table 5.3 presents the results of this experiment for 100 frames of the “Sharky” (frames 100 to 200) sequence at 3, 4, and 8 Mbit/s. This sequence has 6 “hard to code frames” (of which 3 are scene changes), originating a preload bitstream with approximately 100 kbytes. It was chosen because of the high number of “hard to code frames” that it contains (the other two “Sharky” sequences contain only a total of 4 of these frames). Based on these higher than average figures, we can estimate that the coding of the entire 2 hour movie would require the preload of approximately 7,200 frames, originating a preload bitstream of 120 Mbytes and a downloading time of approximately 3 minutes on a 5 Mbit/s channel.

Table 5.3: Comparison between the SNR obtained by using the MPEG-2 coder and the 2-stage coder with the enhancements provided by the processing of “hard to code frames” and preload coding.

	MPEG-2	2-stage encoder
R (Mbit/s)	SNR (dB)	SNR (dB)
3	30.68	31.10
4	31.74	32.19
8	34.37	34.82

Based on the results of table 5.3, it can be concluded that the introduction of the two new algorithmic features leads to an improvement of approximately 0.45 dB. A higher improvement is, however, observed in terms of the subjective image quality. Figure 5.4 illustrates this improvement by presenting a comparison for both the reconstructed images and coding errors obtained for a frame of the “Sharky” sequence with MPEG-2 and 2-stage encoding, both running at 3 Mbit/s.

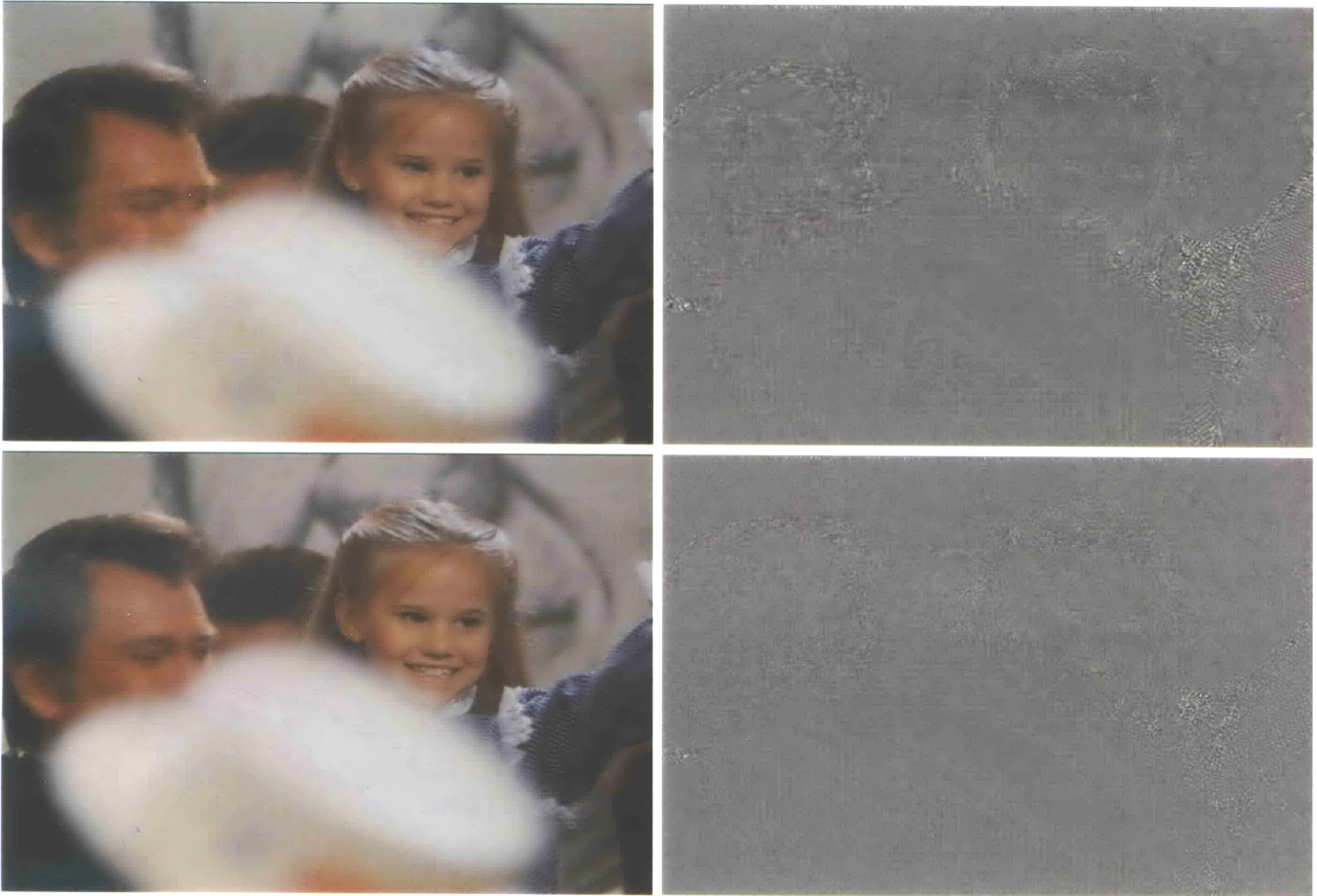


Figure 5.4: Comparison of the subjective coding quality achieved by MPEG-2 (top) and 2-stage (bottom) encoding, both running at 3 Mbit/s. Left: reconstructed images. Right: coding errors magnified by 5.



### 5.3.2 Improved rate-control

To evaluate the gain in performance obtained with the improved rate-control described in section 4.3.3, the 2-stage encoder incorporating this new rate-control was compared with the standard MPEG-2 encoder. Table 5.4 presents the results of this experiment for 100 frames of the “Sharky” (frames 200 to 300) sequence at 3, 4, and 8 Mbit/s.

Table 5.4: Comparison of both the SNR obtained and the bit rate generated by using MPEG-2 and 2-stage encoding with improved rate-control.

MPEG-2		Improved rate-control	
R (Mbit/s)	SNR (dB)	R (Mbit/s)	SNR (dB)
3.0667	35.00	3.0044	35.05
4.0871	35.94	4.0031	36.04
8.1710	37.61	8.0018	37.82

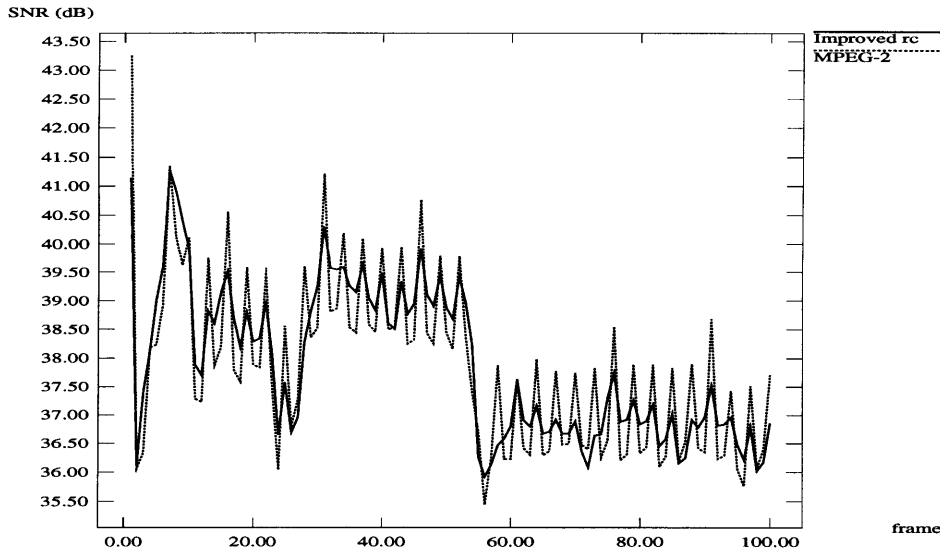


Figure 5.5: Comparison between the SNR obtained with 2-stage encoding using the improved rate-control scheme and the standard MPEG-2 encoder, both running at 8 Mbit/s.

Table 5.4 shows that a small overall improvement in objective coding quality

(0.05 dB for 3 Mbit/s, 0.1 dB for 4 Mbit/s, and 0.21 dB for 8 Mbit/s) is associated with the new rate-control. The subjective quality is, however, increased much more significantly due to the much smoother variation of coding quality from frame to frame achieved with the improved rate-control scheme. This is illustrated by figure 5.5, where the coding quality of the MPEG-2 and the 2-stage encoder (without the features of section 5.3.1) are compared.

An additional advantage of the new rate-control algorithm is the much higher accuracy with which the target bit-rate is met. This is also illustrated by table 5.4, where the target bit-rates are met with an accuracy of at least 0.15 % for the 2-stage encoder as opposed to an accuracy of only 2.3 % for the standard MPEG-2 encoder.

## 5.4 Overall 2-stage encoder performance

In order to analyze the global improvement obtainable with the various enhancements, individually analyzed in the previous sections, a 2-stage encoder incorporating all these features was simulated, and its performance compared with that of the standard MPEG-2 encoder.

Table 5.5: Comparison between the SNR obtained with the MPEG-2 and the 2-stage encoder.

R (Mbit/s)	MPEG-2 SNR (dB)	2-stage encoder SNR (dB)
3	30.68	31.16
4	31.74	32.25
6	33.24	33.82
8	34.37	35.01
10	35.20	35.83
15	36.83	37.47

Table 5.5 presents a comparison, in terms of coding efficiency, between the 2-stage and the MPEG-2 encoders for 100 frames of the “Sharky” (frame 100 to 200) sequence at several bit rates. The 2-stage encoder is consistently better in terms of objective coding quality, as is also illustrated by the rate-distortion curves of figure 5.6.

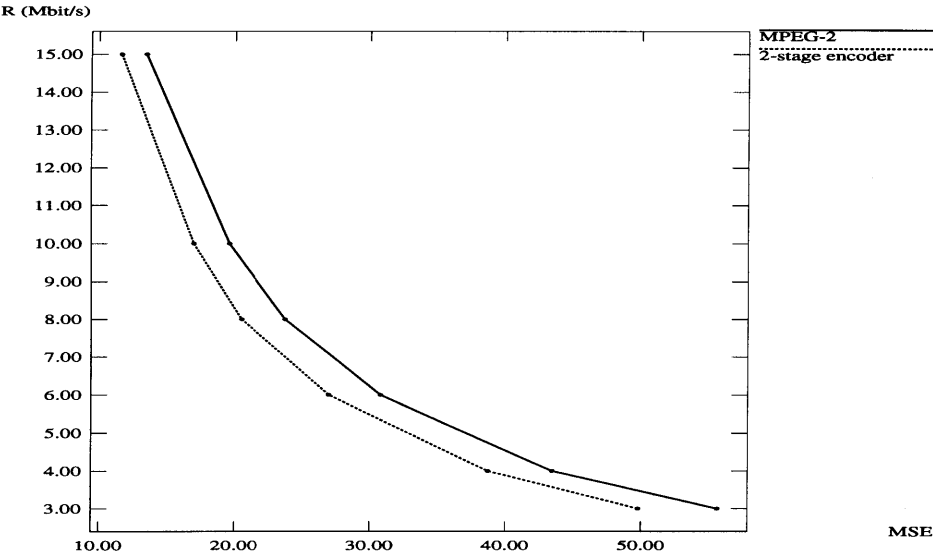


Figure 5.6: Rate-distortion curves for MPEG-2 and 2-stage encoding.

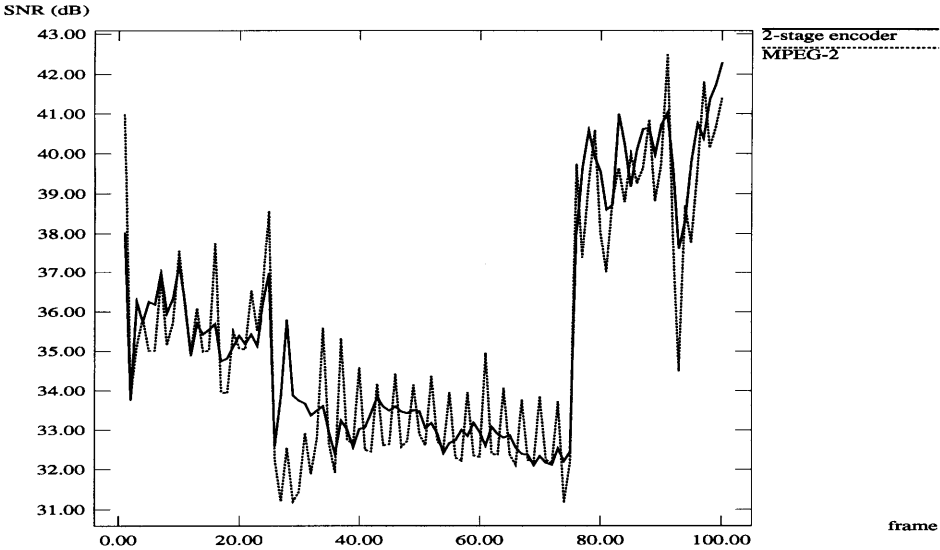


Figure 5.7: Comparison between the SNRs obtained with the 2-stage encoder and MPEG-2, both running at 8 Mbit/s.

The improvement in subjective quality is much larger than what might be ex-

pected from the average 0.6 dB gain of table 5.5 alone. This is due to the increased smoothness in the temporal variations of the coding quality and the elimination of the negative peaks associated with the “hard to code frames” provided by the 2-stage encoder. Both of these aspects are illustrated by figure 5.7.

Notice, in particular, the elimination of the negative peaks in frames 26, 28, 80, and 92, leading to a peak coding gain of approximately 3 dB. Notice also that, in the case of frame 27, the use of preload coding and the improved rate-control lead to significantly increased prediction, originating a considerable improvement in coding quality not only in the preloaded frame, but also in the neighboring ones.

# Chapter 6

## Conclusions

This thesis presents a new 2-stage encoding algorithm for applications involving the broadcast of previously recorded materials. It was shown by this work that a 2-stage structure can lead to both an increased coding efficiency and reduced coding complexity, when it is desired to fulfill several different and parallel quality requirements. In addition, the 2-stage structure provides an intermediate representation of the original footage that allows significant decrease of the storage requirements with no or small (subjectively not noticeable) quality degradation.

It was also shown by this thesis that the potential for improvement of 2-stage encoding resides mainly in the capability to acquire knowledge about the characteristics of the source during the encoding of the first stage. This structure can, therefore, be seen as a form of two pass look-ahead encoding from which the performance of the second stage benefits considerably, leading to an overall improved coding performance. In this particular implementation, this potential was exploited by the introduction of three complementary features: special processing of scene changes, preload coding, and an intelligent rate-control.

Preload coding can be seen by itself as a useful technique to implement the idea of “quality on demand”, where standard quality is provided to all the users and each

user has the choice to obtain higher quality by preloading of the necessary number of frames.

The simplicity and high flexibility of the 2-stage algorithm make it a solution for the implementation of a movie server, capable of satisfying diverse quality requirements of different users, that can be achieved with current technology. In this way, 2-stage encoding can also be seen as a different approach to satisfy the requirements posed to an efficient scalable coding system.

# Bibliography

- [1] W. K. Pratt. *Digital Image Processing*. John Willey, 1991.
- [2] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1992.
- [3] A. Netravali and B. Haskell. *Digital Pictures: Representation and Compression*. Plenum Press, 1988.
- [4] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.
- [5] A. Rosenfeld and A. Kak. *Digital Picture Processing (Vol. I and II)*. Academic Press, 1982.
- [6] J. S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1992.
- [7] D. Pearson, editor. *Image Processing*. McGraw-Hill, 1991.
- [8] N. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice Hall, 1984.
- [9] J. Proakis. *Digital Communications*. McGraw-Hill, 1983.
- [10] A. Habibi. Comparison of the N-th order DPCM Encoder with Linear Transformations and Block Quantization Techniques. *IEEE Trans. on Communications*, Vol. COM-19, December 1971.
- [11] A. Netravali and J. Robbins. Motion Compensated Television Coding: Part I. *Bell Syst. Tech. Journal*, Vol. 58, March 1979.

- [12] H. Musmann, P. Pirsch, and H. Grallert. Advances in Picture Coding. *Proceedings of the IEEE*, Vol. 57, April 1985.
- [13] R. Clarke. *Transform Coding of Images*. Academic Press, 1985.
- [14] G. Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, Inc., 1985.
- [15] P. Wintz. Transform Picture Coding. *Proceedings of the IEEE*, Vol. 60, July 1972.
- [16] N. Ahmed and K. Rao. *Orthogonal Transforms for Digital Signal Processing*. Academic Press, 1975.
- [17] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Trans. on Computers*, Vol. C-23, January 1974.
- [18] K. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.
- [19] J. Max. Quantization for Minimum Distortion. *IRE Trans. on Information Theory*, Vol. IT-6, 1960.
- [20] S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on Information Theory*, Vol. IT-28, March 1982.
- [21] W. Thoma. Optimizing the DPCM for Video Signals using a Model of the Human Visual System. *Proceedings of the 1974 IEEE Int. Zurich Seminar on Digital Communications*, March 1978.
- [22] D. Sharma and A. Netravali. Design of Quantizers for DPCM Coding of Picture Signals. *IEEE Trans. on Communications*, Vol. COM-25, November 1971.
- [23] Y. Huang and P. Shultheiss. Block Quantization of Correlated Gaussian Random Variables. *IEEE Trans. on Communication Systems*, September 1963.
- [24] A. Drake. *Fundamentals of Applied Probability Theory*. McGraw-Hill, 1987.



- [25] C. E. Shannon. A Mathematical Theory of Communication. *Bell Syst. Tech. Journal*, Vol. 27, July 1948.
- [26] D. A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proceedings of the IRE*, Vol. 40, September 1952.
- [27] G. G. Langdon. An Introduction to Arithmetic Coding. *IBM Journal Res. Devel.*, Vol. 28, March 1984.
- [28] W. Pennebaker and J. Mitchell. Probability Estimation of the Q-Coder. *IBM Journal Res. Devel.*, Vol. 32, November 1988.
- [29] ISO-IEC/JTC1/SC29/WG11. *MPEG Test Model*, MPEG93/457.
- [30] D. LeGall. MPEG: a Video Compression Standard for Multimedia Applications. *Communications of the ACM*, Vol. 34, April 1991.
- [31] D. LeGall. The MPEG Video Compression Algorithm. *Signal Processing: Image Communication*, April 1992.
- [32] ITU, CCIR. *Recomendation 601: Encoding Parameters of Digital Television for Studios*, 1986.
- [33] W. Pennebaker and J. Mitchell. *JPEG: Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

# Acknowledgments

I would like to thank to several people that, in different ways, influenced this work.

Andy Lippman, my adviser, for his support and teaching, and the opportunity to work at the Media Lab. Don Mead and Jules Bellisio, my readers, and Ron Burns for their comments. Everyone in the Computer Garden for making this a great place to work, and in particular Henry Holtzman for all the help with the MPEG stuff. Gillian Galloway and Linda Peterson for making my stay at MIT much easier.

My family for all the love, encouragement, and support. My mother for being always present (and waking up at 6 a.m. for calling me in my birthday ...), and never letting me get homesick (even finding a way to send me “ovos moles”). My sister, Terezinha, for all the phone calls, fun letters (with her special way of saying things), and cooking recipes. My brother, André, for all the soccer news and basket discussions (once again our teams are different ...). I miss you all!!

Nuno for his special way of supporting me, both professionally and emotionally. Thanks for all the thesis suggestions, image coding discussions, and the right words when the work stress came up. But, most of all, thanks for your affection and the great times that we have spent together. I love you ...