OPTIMIZATION METHODS FOR DATA COMPRESSION

A Dissertation

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Computer Science

James A. Storer Advisor

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

Giovanni Motta

May 2002

This dissertation, directed and approved by Giovanni Motta's Committee, has been accepted and approved by the Graduate Faculty of Brandeis University in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

_____

Dean of Arts and Sciences

Dissertation Committee

_____

James A. Storer

_____

Martin Cohn

_____

Jordan Pollack

_____

Bruno Carpentieri

To My Parents.

ACKNOWLEDGEMENTS

I wish to thank: Bruno Carpentieri, Martin Cohn, Antonella Di Lillo, Jordan Pollack, Francesco Rizzo, James Storer for their support and collaboration.

I also thank Jeanne DeBaie, Myrna Fox, Julio Santana for making my life at Brandeis easier and enjoyable.

ABSTRACT

Optimization Methods for Data Compression

A dissertation presented to the Faculty of the

Graduate School of Arts and Sciences of Brandeis

University, Waltham, Massachusetts

by Giovanni Motta

Many data compression algorithms use ad–hoc techniques to compress data efficiently. Only in very few cases, can data compressors be proved to achieve optimality on a specific information source, and even in these cases, algorithms often use sub–optimal procedures in their execution.

It is appropriate to ask whether the replacement of a sub–optimal strategy by an optimal one in the execution of a given algorithm results in a substantial improvement of its performance. Because of the differences between algorithms the answer to this question is domain dependent and our investigation is based on a case–by–case analysis of the effects of using an optimization procedure in a data compression algorithm.

The question that we want to answer is how and how much the replacement of a sub–optimal strategy by an optimal one influences the performance of a data compression algorithm. We analyze three algorithms, each in a different domain of data compression: vector quantization, lossless image compression and video coding. Two algorithms are new, introduced by us and one is a widely accepted and well–known standard in video coding to which we apply a novel optimized rate control.

Besides the contributions consisting of the introduction of two new data compression algorithms that improve the current state of the art, and the introduction of a novel rate control algorithm suitable for video compression, this work is relevant for a number of reasons:

- A measure of the improvement achievable by an optimal strategy provides powerful insights about the best performance obtainable by a data compression algorithm;
- As we show in the case of low bit rate video compression, optimal algorithms can frequently be simplified to provide effective heuristics;
- Existing and new heuristics can be carefully evaluated by comparing their complexity and performance to the characteristics of an optimal solution;
- Since the empirical entropy of a “natural” data source is always unknown, optimal data compression algorithms provide improved upper bounds on that measure.

CONTENTS

INTRODUCTION

Many data compression algorithms use ad–hoc techniques to compress data efficiently. Although heuristics are derived from reasonable assumptions on the nature of the data being compressed, it is frequently unknown how and how much these assumptions affect compression.

In a very few cases data compressors can be proved to achieve entropy on specific information sources; however, even in these cases, algorithms often use sub–optimal procedures in their execution. An example is the well–known Lempel–Ziv data compression algorithm (see Ziv and Lempel [1977, 1978], Storer [1988]). Although this algorithm is provably optimal, in the sense that asymptotically it achieves entropy on some information sources, the input string is greedily parsed while it is being compressed. A paper by Horspool [1995] analyzes the performance of the greedy parsing versus an optimal parsing in Lempel–Ziv compression and concludes that it is possible to improve compression. While an optimal parsing will not change its asymptotical behavior, in practice, the new algorithm will converge faster to the entropy and will be more stable with respect to different data sources.

Fast convergence is a crucial issue since an algorithm that converges slowly to the entropy is of little use on real data that are finite in nature and presented to the compressor in the form of small files. Slow convergence is also a problem in adaptive

algorithms since when the input is not stationary and changes its characteristics over time, the algorithm has little time to capture and exploit the changing statistics.

Stability with respect different sources is also particularly important since it is a good indication that the algorithm will work correctly on data sources that have not been tested. Even when the algorithm is provably optimal on a theoretical data source there is in general little or no indication of the performance degradation when compressing natural data, so stability is a clue that the algorithm will work properly on most data.

For all these reasons, the problem of evaluating the effects of optimal data compression strategies has been addressed in the past by a number of authors, frequently resulting in interesting insights on algorithms or even suggesting better heuristics derived from the simplification of the optimal solution.

Besides the previously mentioned work by Horspool, Lempel et al. [1973], Hartman and Rodeh [1985], Katajainen and Raita [1987], Helfgott and Cohn [1997, 1998], Apostolico and Lonardi [1998, 2000] also addressed issues arising in the optimal parsing of sequences in text compression.

In lossy compression, bit allocation is a critical factor to achieve the best rate–distortion trade–off and optimal bit allocation strategies have been explored by a number of authors under different assumptions. Only to mention a few, the works by Hoang et al. [1997, 1999], Ortega [1996] and Wu [1993] deal with the problem of allocating bits from a finite bit pool to each coding unit so that the global coding quality is maximized. The problem of rate–distortion optimization of various aspects of video coding has received the attention of several authors like Rajala et al. [1992], Wiegand et al. [1995], Carpentieri and Storer [1994b], Lee and Dickinson [1994], Ramchandran and Vetterli

[1994], Nosratinia and Orchard [1996], Sullivan and Wiegand [1998], Flierl et al. [1998]. Relevant also is the work by Kozen, Minsky and Smith [1998] who present an algorithm that optimally discards frames in an encoded video sequence in order to lower bit rate while minimizing the gaps.

Even more effort has been put in deriving scalar and vector quantizers that are optimal with respect some principle. Besides the seminal work on scalar quantization conducted by Lloyd [1957] and Max [1960], a number of authors derived optimality conditions for vector quantizers both unconstrained (Linde, Buzo and Gray [1980]) or constrained with respect to one or more features. Gray [1984] and Gersho and Gray [1992] present a comprehensive introduction to the subject while more recent results can be found in Barnes [1989], Barnes and Frost [1990], Frost et al. [1991], Kossentini et al. [1993], Wang and Moayeri [1992], Wu [1990]. Application specific rate–distortion optimization of scalar quantizers is also an issue that has been widely studied. See for example the work by Ratnakar and Livny [1995, 1996] on the optimization of the quantization tables used in JPEG image compression.

While data coding methods, like vector quantization, apply to every kind of data and achieve compression close to the source entropy, practical state–of–the art data compression algorithms are instead strictly data dependent. The reasons that preclude general methods like vector quantization from being used as efficient universal source coders are the exponential complexity of building an optimal encoder and the speed with which the source entropy is reached. Since input data are relatively short sequences, vector quantizers don't have the time to fully exploit the existing redundancies. In practice, compression methods used, for example, for video coding are completely

different from the methods used in speech, audio or text compression and while a number of techniques are shared by most compressors, in order to fully capture the redundancy existing in the input signal, these methods are used or combined in very different ways.

For all these reasons, coherently with the trend existing in the literature in the field, our investigation is based on a case–by–case analysis of the effects of using an optimization procedure in a data compression algorithm. The question that we want to answer is how and how much the replacement of a sub–optimal strategy by an optimal one influences the performance of a data compression algorithm. With a case–by–case approach we analyze three algorithms, each in a different domain of data compression: vector quantization, lossless image compression and video coding. Two algorithms are new, introduced by us and one is a widely accepted and well–known standard in video coding to which we apply a novel optimized rate control. Although most of the experiments that we report are mainly focused on three different flavors of digital image compression (lossy, lossless and moving pictures), some of the algorithms are much more general and cover broader areas of data compression. In particular the vector quantizer is also analyzed in the framework of very low bit rate speech coding and in the compression of random sources.

Besides the contributions consisting of the introduction of two new data compression algorithms that improve the current state of the art, and the introduction of a novel rate control algorithm suitable for video compression, this work is relevant for a number of reasons:

- A measure of the improvement achievable by an optimal strategy provides powerful insights about the best performance obtainable by a data compression algorithm;

- As we show in the case of low bit rate video compression, optimal algorithms can frequently be simplified to provide effective heuristics;
- Existing and new heuristics can be carefully evaluated by comparing their complexity and performance to the characteristics of an optimal solution;
- Since the empirical entropy of a "natural" data source is always unknown, optimal data compression algorithms provide improved upper bounds on that measure.

Since the work spans three different fields of data compression, the first two Chapters of this thesis introduce the state of the art by presenting in Chapter 2 the key technologies that are commonly used and in Chapter 3 by giving an overview of how these techniques are used to compress audio, speech, image and video signals with the most important compression standards.

In the Chapter 4 we address the problem of designing and analyzing a low complexity vector quantizer whose codebook is designed both with optimal centroids and with a sub–optimal stagewise design. A novel combination of residual and trellis quantization is presented and compared to existing methods. Then we derive the necessary conditions on the centroids in an optimal codebook that minimizes the mean square quantization error. Because of the complexity involved, these conditions turn out to be not practical, so an alternative sub–optimal codebook design is proposed. A number of experiments on random sources, on direct coding of digital images and on very low bit rate coding of linear prediction parameters in a speech codec are presented and compared to the current state of the art. Both an exhaustive (optimal) and a Viterbi–based (sub–optimal) trellis search are used and compared. We conclude that this novel Trellis Coded Vector Residual Quantizer provides very good performance, in particular at low bit rates, and a

computational complexity lower than other existing methods. In all cases, the performance loss introduced by the Viterbi search is widely compensated by a lower computational complexity.

In Chapter 5 we propose and study a new lossless image compression algorithm that uses linear prediction and embedded context modeling. A linear predictor is computed for each pixel by minimizing the square error in a set of causal neighbors. Two different methods to determine the optimal linear predictor are compared and discussed. The prediction error is entropy coded by using two different methods: arithmetic coding and Golomb coding. Since the algorithm tries to exploit local statistics, the model for the arithmetic encoder is based on a small number of samples and it may give no indication of the probability of some error values. To overcome this lack of information, also called the "zero probability" problem, we have successfully used the Laplacian approach. A number of results on standard test images are presented and compared to state of the art lossless image codecs. While pixel–by–pixel optimization is computationally expensive and still far from being practical, the compression that we get is competitive with the state of the art and the performance is stable on every image of the test set.

Chapter 6 addresses the problem of designing an optimal frame–rate control algorithm that, embedded in a low bit rate video encoder, minimizes the number of frames that are skipped during transmission buffer overflows. Overflows are due to the attempt of matching video variability to the finite capacity of the transmission channel. We present an optimal solution to this problem with an algorithm that is based on dynamic programming. The rate control is tested in a H.263+ encoder by using video sequences that have a high number of scene cuts. The optimal solution always allows the

transmission of a much higher number of frames with slightly better quality and with the same number of bits. The analysis of the pattern of the optimal solution suggests that most of the gain depends on the skipping of the first frame of the new scene. Based on this observation, a heuristic is developed and embedded in the H.263+ video encoder. This enhancement entails no change in the standard or in the decoder. Experiments on the test sequences show that this heuristic achieves most of the gain of the optimal solution with a computational complexity that is substantially lower. Finally an unrestricted version of the optimization problem is studied and proved to be NP–complete.

KEY TECHNOLOGIES IN DATA COMPRESSION

Digital data compression aims at a space efficient representation of digital data. The need for compression is mainly motivated by the increasingly large amount of data used by current applications.

While the interest in the compression of digital data is relatively recent, the concept of compression itself has been present in the analog world for a long time. Examples of analog compression are PAL, NTSC and SECAM color systems, all employed to reduce the full bandwidth R, G and B signals into a single 5.5 or 4.2 MHz channel suitable for broadcast TV transmission.

In the following we describe compression methods that apply both to signals that are digital in nature (like text) as well as signals that are obtained by digitization of a sequence of analog measures.

Storage and transmission are the most common operations performed on digital data. Which one of these two operations must be performed more efficiently is a decision that sometimes affects the choice among the available compression methods. Transmission and storage applications determine two classes (not necessarily disjoint) of compression algorithms that have different requirements, characteristics and performance.

The transmission–storage duality is captured by the Rate–Distortion theory. An encoder suitable for transmission must minimize the distortion while keeping the rate

fixed (or by keeping its average close to a given threshold), a storage–oriented algorithm typically minimizes the rate while achieving a given distortion.

This is not obviously a rigid rule and each compression algorithm has to tradeoff among some strictly correlated parameters. Each of them can be increased (in some extent) despite of one or more of the others (Bhaskaran and Konstantinides [1995]):

- **Coding Efficiency**: usually defined in terms of Compression Ratio, bits per symbol, etc.;
- **Signal Quality**: measured by the Signal to Noise Ratio, with perception motivated metrics or empirically assessed with subjective experiments;
- **Coding Delay**: measures both the delay introduced in the coding process and the size of the encoding buffer required;
- **Coder Complexity**: expressed in terms of algorithmic complexity, number of operations, memory space, or electrical power needed by a hardware implementation of the encoder.

The relation between the first two parameters is modeled by the Rate–Distortion theory (Gallager [1968], McEliece [1977], Cover and Thomas [1991]) and obeys well–known bounds; the others two are typical of each implementation.

Compression is achieved by removing some information present in the signal. It is possible to distinguish between two kinds of information that are commonly removed:

- **Redundant** information, that is statistical in nature and can be reconstructed by observing other parts of the signal;
- **Irrelevant** information, that is not useful in the target application.

Algorithms chosen to compress data that have to be used by humans, are frequently based on the removal of irrelevant information that the user cannot perceive.

Another important difference between the two classes is that while statistical methods that are used to remove redundancies can be lossless (the original signal can be reconstructed from its compressed representation without any error) or lossy (some errors are introduced in a controlled manner), the elimination of irrelevancies is an intrinsically lossy process (Storer [1992]).

Methods that remove redundancies and irrelevancies are often combined. Figure 2.1 shows a typical configuration used by a video coder. The second stage removes the irrelevant part of the signal by quantizing a transformation of the signal; the third stage removes the remaining statistical redundancy by using, for example, a variable–length or arithmetic encoder.

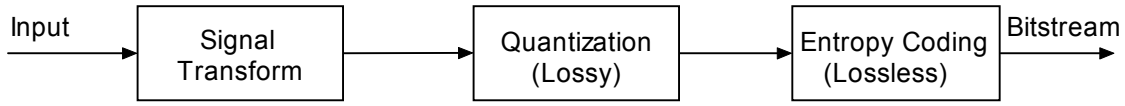


Figure 2.1: Stages of a generic video encoder.

2.1 Signal Representation

Digital signal compression is applied to signals that are discrete in nature or obtained from the digitization of analog measures. A signal is usually described by a function $f(x_1, x_2, ..., x_n)$ of n independent variables, also called *dimensions*; a common classification groups together signals having the same number of dimensions.

With this formalism, an audio signal is a one–dimensional function $f(t)$ that describes the air pressure acting on a transducer as a function of the time. An image is a two–dimensional function $f(x, y)$ describing the luminosity of a pixel in a scene as a function of two spatial coordinates. A video signal, being a temporal sequence of images, adds a temporal parameter to the image description thus becoming a function $f(x, y, t)$.

Since in a digital signal both free variables and values of the function must be discrete, when the signal originates from a sequence of analog measures, digitization must be performed both in the time dimension (*sampling*) and in the values of the domain (*quantization*). While from a theoretical point of view sampling and quantization can be applied in any order, for technological reasons sampling is often performed first (Jayant and Noll [1984]).

2.1.1 Sampling

Sampling is the process that converts a continuous time signal into a discrete time signal by measuring its amplitude at regular time intervals. If the signal being sampled is band

limited, i.e. if its frequency F is comprised between two frequencies $F_{\min}$ and $F_{\max}$, the *Sampling Theorem* (Jayant and Noll [1984]) guarantees that a perfect reconstruction of the original signal is possible only if the signal is sampled at regular intervals $T = 1/F_s$ where F_s is the sampling frequency that must be greater than or equal twice the bandwidth:

$$F_s \geq 2 * (F_{\max} - F_{\min}).$$

Formally, the Sampling Theorem states:

Sampling Theorem: For a sampling interval, T, there exists a critical frequency, $F_S = \frac{1}{2T}$, known as the Nyquist frequency. If the data under observation is band limited to a frequency less than the Nyquist frequency then that function can be sampled and represented, digitally, in a representation similar to the continuous limit as the sampling interval tends to zero.

In other words, if a signal has a maximum frequency comprised between $F_{\min}$ and $F_{\max}$, no information is lost by sampling the signal at $F_s \geq 2 * (F_{\max} - F_{\min})$ and a perfect reconstruction of the original signal from its samples is always possible. When a signal is sampled at a frequency lower than its Nyquist frequency, aliasing occurs and the reconstructed signal will exhibit frequency components that were not present in the original (see Fig. 2.2). In the frequency domain, aliasing causes part of the signal that resides in a region greater than the critical frequency to be spuriously moved into a valid region between zero (or DC) and the Nyquist frequency.

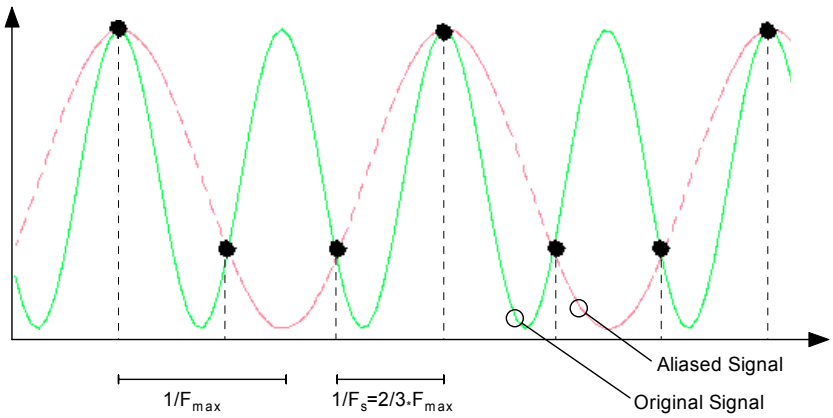


Figure 2.2: Aliasing in a signal sampled with $F_S = \frac{3}{2} F_{\max} \leq 2F_{\max}$

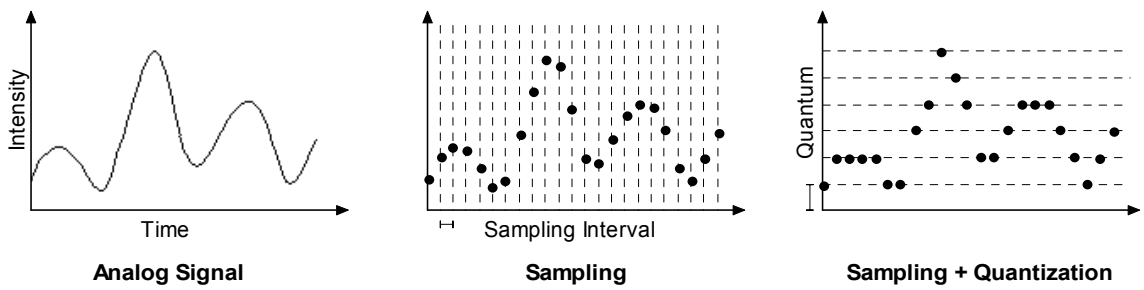


Figure 2.3: Sampling and Quantization of an analog signal.

2.1.2 Quantization

Discretization in the values of the domain is called quantization. This process approximates a continuous–amplitude signal by a discrete–amplitude signal, while minimizing a given distortion measure.

Unlike sampling, quantization is an intrinsically lossy process, and after the quantization the original signal cannot be recovered without errors. In a quantized signal each sample can be represented by the index of a value selected from a finite set. It is also

common to perform quantization on a signal that already has discrete amplitude, meaning with this a reduction in the number of the possible values that a sample can assume.

The simplest way to quantize a signal is one sample at a time, by using a uniform division of the value interval. In this process, called uniform scalar quantization, the value of each sample is divided by an interval (quantization step or quantum) and the integer part of the result is encoded. If the samples are $x_1, x_2, \ldots, x_n, \ldots$ and the quantization interval is q, the quantizer determines $\hat{x}_i = \text{int}(x_i / q)$ and outputs the corresponding code word (or index) for $\hat{x}_i$.

Better performance, at the cost of a greater complexity, can be obtained by considering the signal statistics (non–uniform quantization) or by quantizing the signal in blocks composed by a number of consecutive samples (Vector Quantization). See Gersho and Gray [1992], Proakis and Manolakis [1996] for an extensive coverage of this topic.

2.2 Digital Data Formats

In the following we introduce the most common digital data formats and discuss their main characteristics.

2.2.1 Audio Formats

Sound is a wave of pressure differences that travels through the air, so sound can be detected by measuring the pressure level at a specific location as a function of the time. A transducer (as a microphone for example) is a device that converts air pressure differences into variations of some electrical quantity. Its output can be considered a continuous (or analog) signal and it is usually digitized with the techniques that we have already discussed.

In audible signals, sampling is performed at frequencies that range from 96KHz for high quality music to 8KHz for telephone quality speech.

Quantization can be performed by using linear intervals (as in linear Pulse Code Modulation or PCM) or more appropriately with logarithmic intervals. The speech signal that is transmitted over the digital telephone lines is frequently quantized by using μ – law or A–law, two logarithmic scales that are designed to take advantage of the non–linear sensitivity of the human ear.

To represent complex sounds generated by multiple sources it is also possible to measure the wave from two or more different space locations, generating in this way multiple measures of the same event (stereophonic audio).

A full specification of a digital audio signal is characterized by the following four parameters:

- Sampling rate: expressed in number of samples per second;
- Number of bits per sample: in general 8, 12, 16 or 24;
- Digitization law: linear, log, μ – law, A–Law, etc.;
- Number of channels: 1 for monaural, 2 for stereo, etc..

While most audio compression algorithms are designed to be general–purpose and work independently on signal content and parameters, an interesting exception is constituted by the class speech codecs. Speech is susceptible of a special treatment because it is possible to model its source (the human vocal tract) thus making possible the design of model–based algorithms that are capable of extremely high compression.

2.2.2 Still Image Formats

A gray–level image can be considered a two dimensional signal where a light intensity $s(x, y)$ is associated to each point (x, y) of a uniform sampling grid. Even in this case, the considerations made in the previous section about sampling and quantization are still valid. However, due to physical and technological limitation, image sampling is always performed at a lower spatial resolution (with respect the original), so perfect reconstruction cannot be usually achieved.

A special class of gray–level images consists of images in which $s(x, y)$ can only assume two values, conventionally designated by zero and one. These images, called *bi–level* images, are commonly generated by fax machines or by scanning high–contrast paper documents. They are important because special algorithms have been designed for their compression.

Color images are commonly described as three superimposed gray–level pictures, also called *color planes*. Each pixel in a color plane represents the spatial measure of the intensity of a primary color (for example Red, Green and Blue as in the common RGB scheme). These representations are "hardware oriented" in the sense that they reflect the representation used by color monitors, printers or photographic processes. From this point of view a color picture is a three–channel, three–dimensional signal.

In natural images, where a strong correlation between the three channels is expected, an alternative color representation can be used in order to exploit this correlation. This representation, used by most image and video compression algorithms, will be briefly described in a following section.

2.2.3 Digital Video Formats

Since a video signal can be viewed as a sequence of pictures, digital video combines the temporal aspects of audio signals and the spatial characteristics of still pictures.

When the digital video originates from the sampling of analog broadcast television, the differences existing between NTSC, PAL and SECAM national standards limit or even preclude device interoperability. With the purpose of standardizing digital video

format several representations have been defined; each standard specifies the following three parameters:

- *Spatial Sampling*: the number of horizontal and vertical pixels (or lines) in a picture;
- *Temporal Sampling*: the number of pictures per second and whether the signal is interlaced (odd lines grouped and transmitted before even lines) or non–interlaced;
- *Color Sampling*: how the color of each pixel is represented.

Here we review some common standard video formats:

- **CCIR–601**: the official sampling standard for the conversion of analog television signals; it specifies a different spatial and temporal sampling for NTSC (the United States National Television Systems Committee) and PAL (Phase Alternating Line) television signals. The NTSC signal is sampled with 720x485 pixels and 30 frames/sec and the PAL signal is 720x576 at 25 frames/sec. Colors are represented with in a color space called YC_bC_r. Color components are sub–sampled with a scheme named 4:2:2 (see section on Color Decorrelation). CCIR–601 uses interlaced scan. The vertical resolution for each field is one half of the full resolution since each frame consists of two interlaced *fields* that are spatially but not temporally adjacent.
- **Source Input Format or SIF**: specifies a spatial sampling of 360x240 pixels at 30 frames/sec for the NTSC video signal and 360x288 pixels at 25 frames/sec for the PAL signal. The color sub–sampling for the YC_bC_r components follows the format named 4:2:0.

- **Common Intermediate Format or CIF**: proposed to bridge the differences between the NTSC and PAL video formats. It uses progressive (non–interlaced) scan, an image size of 352x288 pixels at 30 frames/sec and 4:2:0 color subsampling for the YC_bC_r components. These values represent half the active lines of a PAL television signal and the picture rate of the NTSC signal. Therefore, when converting an analog signal to CIF, PAL systems need only to perform a picture rate conversion and NTSC systems need only to perform a line–number conversion. Multiples of the CIF format are also defined and usually referred as SQCIF (128x96 pixels), QCIF or Quarter CIF (176x144 pixels), 4CIF (704x576 pixels) and 16CIF (1408x1152 pixels).

2.3 Basic Methods

Almost every "natural" information source exhibits some sort of correlation among its samples. The following paragraphs introduce a number of basic data compression techniques that are frequently combined to form state–of–the–art algorithms. Each of these methods uses a slightly different approach to remove or reduce the amount of correlation present in the signal. Some methods are based on assumptions on the nature of this correlation, so their use is restricted to data sources that satisfy these assumptions.

2.3.1 Entropy Coding

Entropy coding exploits the fact that in a signal symbols may occur with different frequencies and, when encoding source symbols one at a time, the a priori probability of occurrence of a symbol may change depending on the past observations. In an English text, for example, the letter "e" occurs with the highest frequency and the probability that a letter is an "e" given that past three letters being " th" is extremely high. On the contrary, the probability of a "q" following " th" is very low.

An entropy encoder uses these considerations by estimating for each symbol a probability that is based on the past observations, then encodes highly probable symbols with the shortest code words. This method, called variable–length coding, is the base of most entropy coding algorithms (Storer [1988]).

2.3.2 Run Length Coding

A simple lossless compression technique is Run Length Encoding (Golomb [1966]). RLE is used when compressing scanned documents and faxes. These images alternate long sequences of pixels having the same color. A sequence (or *run*) of contiguous pixels of the same value is encoded by the pair (R, L) where R is the recurrent value and L is the length of the run.

For example, the sequence of values "10000000220000001111 1..." can be RL encoded as "$(1,1)(0,7)(2,2)(0,6)(1,5)...$". Uniquely decipherable codes (fixed length, prefix, etc.) must be used to represent the pairs so that no ambiguity arises when decoding the data stream. As previously introduced, a common application of RLE is the encoding of bi–level images. On this kind of images, because the black and white runs alternate, except for the first run, it is not even necessary to send the recurrent value R. In fact, the CCITT Group 3 and 4 facsimile standards are mainly based on RLE (Sayood [1996]). RLE is also used to encode the sequences of zeros present in the quantized DCT coefficients in the JPEG standard (Pennebaker and Mitchell [1993]).

2.3.3 Huffman Coding

Huffman [1952] codes are an optimal way of assigning variable length code words to an alphabet of symbols with a known probability distribution.

Starting from the symbol probabilities, a (binary) Huffman code is built by grouping together the two symbols that have the smallest probabilities and assigning to the last bit of the their code words a 0 and a 1 respectively. The construction is repeated by

considering a new set of symbols in which the two symbols with the lowest probability have been substituted by a super symbol having probability equal to the sum of the two lowest probabilities. The construction stops when a single symbol with probability 1 remains.

This method builds bottom–up a binary tree in which each branch corresponds to a bit. Every leaf of this tree corresponds to a symbol whose code word is obtained by the sequence of bits along the path from the root to the leaf (See Fig. 2.4).

Symbol	Probability	Code word
A	0.25	01
B	0.125	000
C	0.5	1
D	0.125	001

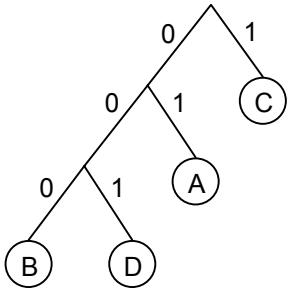


Figure 2.4: Construction of a Huffman code.

The code generated by this construction is optimal because the bit at every branch distinguishes among two roughly equiprobable left and a right sub–trees. When the probabilities are not powers of two, Huffman codes have some redundancy that arises from the fact that it is not possible to partition the symbols so that all internal nodes in the tree have perfectly equiprobable sub–trees (see Capocelli et al. [1986], Capocelli and De Santis [1988, 1991], Johnsen [1980]). Encoding sequences rather than individual symbols can reduce the redundancy introduced by this method.

Locating the symbols in the leaves only, guarantees that this construction always results in a *prefix* (originally *prefix free*) code. In a prefix code no codeword is prefix of another codeword; because of this property, any sequence of code words is

instantaneously uniquely decodable (Gallager [1968]). So in a left–to–right decoding of the bit stream it is always possible with no look ahead to determine where the code for one symbol ends and the code for the next symbol begins.

Huffman codes can be generalized to non–binary trees (when the coding alphabet is non binary) as well as to the case in which the probability distribution for the next symbol depends on the previous symbols. Unfortunately, in this case the size of the data structure that maintains the probabilities grows exponentially with the number of symbols.

When the symbol probabilities are not known or when the probabilities change over time, this approach can be made adaptive by incrementally updating the data structure as new symbols are processed. The updates reflect the probability distribution of symbols seen thus far. This approach is used by the UNIX "compact" utility.

For further reading, see the original paper by Huffman [1952], Gallager [1978] and the book by Storer [1988],

2.3.4 Arithmetic Coding

Arithmetic coding (Moffat, Neal and Witten [1995]) is an entropy coding method useful to encode symbols with arbitrary probabilities provided by an adaptive model that will be discussed shortly. The idea is to encode a data sequence as a single long binary number that specifies a point on the real line between 0 and 1. This point represents the value of the cumulative density function of the sequence. With this method, not only do less frequent characters get shorter codes, but "fractions" of a bit can be assigned, avoiding in this way the redundancy common to most Huffman codes.

Both the encoder and decoder proceed incrementally by maintaining a coding interval between the points low and $high$; initially $high = 1$ and $low = 0$. Each time a new symbol s is read by the encoder or written by the decoder, they both refine the coding interval by doing:

$$
\begin{aligned}
length &= high - low \\
high &= low + (RIGHT(s) * length) \\
low &= low + (LEFT(s) * length)
\end{aligned}
$$

As more and more symbols are encoded, the leading digits of $LEFT$ and $RIGHT$ become the same and can be transmitted to the decoder (See Fig. 2.5). Assuming binary notation, if the current digit received by the decoder is the ith digit to the right of the decimal point in the real number r formed by all of the digits received thus far, then the interval from r to $r + 2^{-i}$ is a bounding interval and the $LEFT$ and $RIGHT$ ends of the final interval will both begin with the digits of r.

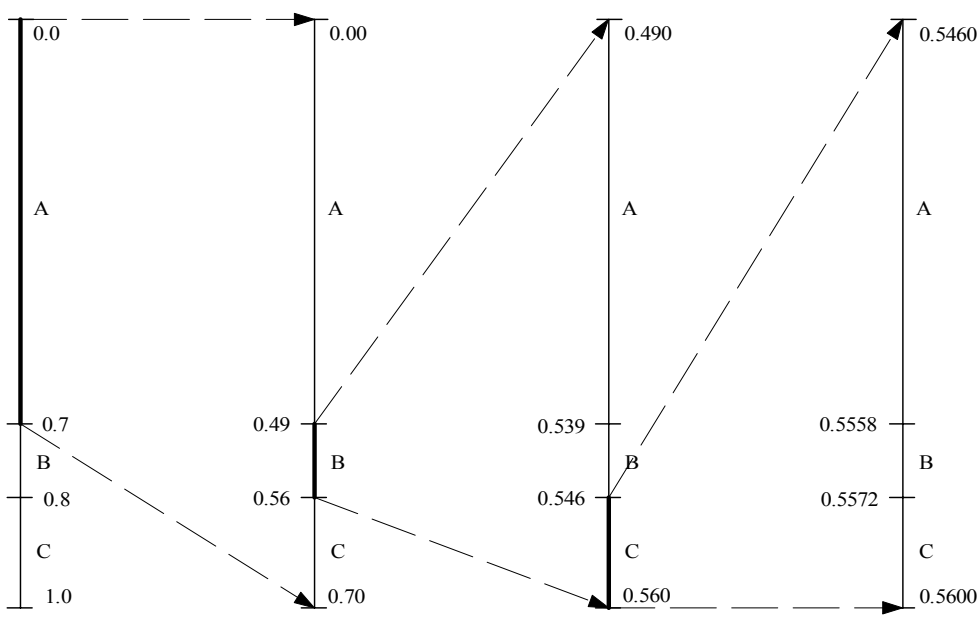


Figure 2.5: Arithmetic encoding of the input "ABC".

Practical implementations are complicated by the use of limited precision arithmetic to avoid expensive arithmetic operations and a number of methods have been devised to overcome this problem (Witten, Neal and Cleary [1987]). For further reading, see the books by Bell, Cleary, and Witten [1990] and Witten, Moffat and Cleary [1994].

2.3.5 Golomb Coding

Golomb [1966] codes were introduced to optimally encode the result of run–length coding. These codes are optimal for exponentially decaying distributions of positive integers and, when applicable, lead to encoding algorithms that are much more efficient than other entropy coding methods (see Howard [1989]).

A Golomb code of parameter m encodes a positive integer n with a binary representation of $(n \bmod m)$ followed by a unary representation of the remaining most significant bits of n and, finally, a stop bit.

A special case is the family of codes in which the parameter m is a power of two: $m = 2^k$. In this case, the modulus operation is particularly simple and the coding process is even more efficient. In such a code the length of a code word is given by:

$$k + 1 + \left\lfloor \frac{n}{2^k} \right\rfloor$$

Several methods have been devised to estimate the parameter of an optimal Golomb code from a sequence of observations. For a further discussion on this topic see for example Seroussi and Weinberger [1997].

In practice, Golomb codes are important because they are frequently used to encode prediction errors. In most cases the probability of a prediction error $e(t)$ can be modeled by a symmetrical, zero centered, Laplacian distribution. Even in this case a Golomb code can be used after the application of a mapping that rearranges error values into a positive exponentially decaying distribution. One of these mappings can be found in LOCO–I and in the lossless image compression standard JPEG–LS (Weinberger, Seroussi and Sapiro [1996]):

$$M\left(e(t)\right) = \begin{cases} 2 \cdot e(t) & \text{if } e(t) \geq 0 \\ 2 \cdot \left|e(t)\right| - 1 & \text{otherwise.} \end{cases}$$

2.3.6 Textual Substitution Methods

In a textual substitution method, encoder and decoder cooperate to maintain identical copies of a dictionary D that is constantly changing during the compression. The encoder reads symbols from the input stream, matches these symbols with an entry of D, transmits a reference to this entry, and updates the dictionary with a method that depends only on the current contents of D and on the current match. The index of the match in the dictionary is the compressed text fragment.

Textual substitution methods are often referred to as “LZ–type” methods due to the work of Lempel and Ziv [1976] and Ziv and Lempel [1977, 1978], where “LZ1–type” or “LZ’77–type” methods are those based on matching to the preceding n characters and “LZ2–type” (see also Storer and Szymanski [1978, 1982]) or “LZ’78–type” are those based on a dynamic dictionary.

In LZ’78, the dictionary consists of all substrings of the data most recently processed. Compression starts by prepending a dummy string containing the entire alphabet, then the method matches the longest possible prefix of unprocessed characters with a substring that has occurred in the past. That prefix is encoded by sending a reference pointer to its earlier location, its length and one additional character, called “*innovation*”, that is encoded verbatim. The process repeats until the end of the file.

While LZ’77 uses the previously encoded text as a dictionary, LZ’78 maintains instead an explicit structure that contains only previously cited phrases, each augmented by one new character. Compression is achieved by citations of these dictionary entries. Since not every location in the past may begin a citation, the cost of coding is reduced. LZ’78 scans unprocessed data while searching for the longest prefix not in the dictionary. That prefix is encoded as a dictionary index plus one innovation character, and then inserted as a new dictionary entry. In 1984, Welch proposed a variation on LZ'78 that was named LZW. In this variant the augmented string enters the dictionary as usual but the encoding of the innovation character is deferred to the succeeding match.

Methods that use a dynamic dictionary take advantage of a trie data structure to efficiently implement the pattern matching. For further reading see the book of Storer [1988]. Practical implementation of textual substitution algorithms are, for example, the UNIX “gzip” utility that employs the LZ’78 approach, the UNIX “compress” utility and the V.42bis modem compression standard that employs the LZ’77 approach. Because any difference between the dictionaries maintained by the encoder and the decoder will cause a long (possibly infinite) sequence of errors, unless specific techniques are employed, this method is very sensitive to transmission errors. Storer and Reif [1997] address the

problem of error propagation in dictionary methods and show how a hashing based solution can be used to prevent and limit error propagation.

2.3.7 Statistical Methods

Statistical compression methods develop a statistical model of the data being encoded. Typically a common statistical method consists of a modeling stage followed by a coding stage. The model assigns probabilities to the input symbols, and the coding stage performs the actual coding with an assignment of symbols based on those probabilities.

The model can be either static or dynamic (adaptive). Most models are based on one of the following two approaches.

- **Frequency:** The model assigns probabilities to the text symbols based on their frequencies of occurrence, such that commonly occurring symbols are assigned short codes. A static model uses fixed probabilities, while a dynamic model modifies the probabilities and adapts to the text being compressed.
- **Context:** The model assigns probability by considering the context in which a symbol appears. Contexts and symbol probabilities are determined from the data already encoded. The context of a symbol is often composed by a number of symbols preceding it and the method is said to "predict" the symbol based on that context.

PPM, a sophisticated compression method originally developed by Cleary and Witten [1984] and extended by Moffat [1990], is an example of context–based compression. This method is based on an encoder that maintains an explicit statistical model of the

data. The encoder inputs a new symbol, encodes it with an adaptive arithmetic encoder that uses its estimated a priori probability.

2.3.8 Vector Quantization

Since Shannon [1948] determined its theoretical optimality, Vector Quantization (or VQ) has been proved to be a powerful coding technique that is effective and independent of the data source. It works by dividing the sequence of input symbols into blocks (or *vectors*) of length n. Each vector is independently encoded by searching for the closest match in a dictionary (*codebook*) of representative vectors. The codebook is common to both encoder and decoder.

Formally, if x is an n–dimensional random vector in $\mathbb{R}^n$, an N–levels Vector Quantizer of $\mathbb{R}^n$ is a triple $Q = (A, F, P)$ where:

- $A = \{y_1, y_2, ..., y_N\}$ is a finite indexed subset of $\mathbb{R}^n$ called codebook. Its elements y_i are the code vectors;
- $P = \{S_1, S_2, ..., S_N\}$ is a partition of $\mathbb{R}^n$ and its equivalence classes (or *cells*) S_i satisfy:

$$\bigcup_{i=i}^{N} S_i = \Re^n \text{ and } S_i \cap S_j = \varnothing \text{ for } i \neq j;$$

- $F : \mathbb{R}^n \to A$ is a function that defines the relation between the codebook and the partition as

$$F(\vec{x}) = y_i \text{ if and only if } \vec{x} \in S_i.$$

Vector quantization is asymptotically optimal when the dimension of the vector increases. Unfortunately the construction of the partition for the determination of the optimal code vectors has been proved to be an NP–complete problem by Lin [1992]. Another drawback that limits its use is that the size of the codebook grows exponentially with the dimension of the vector.

Many variations have been proposed to speed up the encoding and simplify the codebook design, many based on trees or on a trellis data structure. For further reading see the book of Gersho and Gray [1992].

In a Vector Quantizer, encoding and decoding are highly asymmetric. Searching for the closest block in the dictionary (encoding) is computationally much more expensive than retrieving the codeword associated to a given index (decoding). Even more expensive is the dictionary design; fortunately this process is usually performed off–line. In Constantinescu and Storer [1994, 1994b] the authors propose a method to perform VQ with an on–line adaptive construction of the dictionary.

2.3.9 Prediction

A general paradigm in data compression is the concept of prediction. Prediction allows a compact representation of data by encoding the error between the data itself and the information extrapolated or "*predicted*" from past observations. If the predictor works well, predicted samples are similar to the actual input and the prediction error is small or negligible, and so it is easier to encode than the original data.

A specific realization of the source is compressed by predicting the value of each sample from a finite number of past observations and sending to the decoder the

difference between the actual sample and its predicted value. The decoder, that shares with the encoder the prediction mechanism, can mimic encoder's prediction, add it to the error that has been sent by the encoder (See Fig. 2.6) and, finally, reconstruct the original sample.

When prediction is used to perform lossy compression, in order to avoid error propagation both encoder and decoder have to base the prediction on the encoded samples. While encoded samples contain noise that has been introduced by the lossy encoding, they constitute the only common knowledge between encoder and decoder.

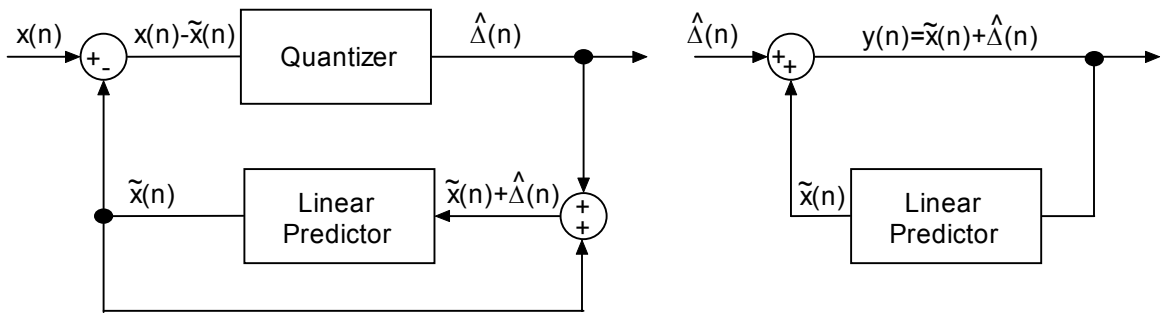


Figure 2.6: Predictor.

In the literature there is a distinction between backward and forward prediction, both mechanisms being very common in data compression. In backward prediction, the value of the current sample is predicted on the base of the past (encoded) observations. Since the prediction is based on the output data, common to both encoder and decoder, there is no need to send side information. Forward prediction instead, collects a number of input samples, determines for this block of data a predictor that is optimal with respect to some criteria and sends to the decoder both a parametric representation of the predictor and the prediction errors. While forward prediction performs better than backward prediction in

presence of rapid signal variations, a number of samples must be collected before the prediction and this process introduces a delay that may be intolerable in some systems.

The most common kind of prediction, called Differential Pulse Code Modulation (or DPCM), assumes that the current sample is identical to the previous. In this simple yet effective method, the prediction error is the difference between the current sample and the last encoded value.

Another common method, widely used in speech coding (Rabiner and Schafer [1978]) and recently found also useful in image coding (Meyer and Tischer [1997, 1998, 2001], Motta, Storer and Carpentieri [1999, 2000], Wu, Barthel and Zhang [1998]) assumes a linear dependence between samples that are spatially or temporally adjacent.

The idea behind linear prediction is that a sample can be approximated by a linear combination of past samples. Using this approach, the sample x_n can be predicted as a weighted sum of p previous samples (prediction of *order* p):

$$\tilde{x}_n = \sum_{k=1}^{p} a_k x_{n-k}$$

where $a_1, a_2, \ldots, a_p$ are the prediction coefficients, optimized to reduce the error variance.

The prediction error is expressed as:

$$e_n = x_n - \tilde{x}_n = x_n - \sum_{k=1}^{p} a_k x_{n-k}$$

LP has been extensively studied because it is an essential tool used in speech compression, and a number of efficient algorithms have been designed to determine the predictor's coefficients.

2.3.10 Transform and Sub–Band Coding

Transform coding designates a family of algorithms that are not compression methods in the literal sense (some transformations even increase the size of their input), but are able to enhance entropy coding by transforming the data into alternative representations that are easier to compress.

In general the transformation exploits characteristics that are peculiar to the signal and increases the performance of an entropy coder or allows a better control of the error introduced by a lossy compressor.

In the case of lossless compression, transforms such as the Burrows–Wheeler (Burrows and Wheeler [1994]) enhance some statistical properties of the data by rearranging the sequence of symbols.

When compressing natural sources like audio, images or video for example, the information that is relevant for a human user is often better described in the frequency domain. Decomposing a signal into its frequency components allows the encoder to prioritize the data and introduce error where the user is less likely to perceive it.

When the transformation is implemented in the form of a filter bank, transform coding frequently takes the name of sub–band coding. In a sub–band coder the input signal is separated by a filter bank into multiple streams, each containing components in a particular interval of frequencies. Such streams are decimated (or sub–sampled) and individually encoded. Division into bands allows the exploitation of phenomena that are better described in the frequency domain, like, for example, the frequency masking.

More details can be found in Tsutsui et al. [1992], Woods [1991], Pennebaker and Mitchell [1993].

2.3.10.1 Discrete Cosine Transform

The human visual system processes information with mechanisms that work in the frequency domain. In the eye, for example, detectors are connected with inhibiting connections so that object contours (spatial high frequencies) are enhanced during perception. Even the human auditory system acts as a filter bank in which the cochlea decomposes sounds into critical bands, each centered on a different frequency. Because of these reasons several time–to–frequency transforms have been used for the compression and the analysis of digital signals. A few of them are variations of the Fourier transform, a powerful tool used in analytic calculus to decompose a function in a weighted sum of periodic functions having different periods and phases.

The frequency transform most used in data compression is the Discrete Cosine Transform or DCT. DCT has the advantage of achieving good decorrelation of the signal while requiring modest computation.

The discrete cosine transform of $n = 2^k$ samples of a real signal $s(0), s(1), \ldots, s(n-1)$ is given by the sequence of n real numbers $C(0), C(1), \ldots, C(n-1)$ given by:

$$C_i(s) = \sum_{j=0}^{n-1} \sqrt{\frac{\delta(i)}{n}} s(j) \cos\left(\frac{(2j+1)i\pi}{2n} \right)$$

The sequence of numbers $C(0), C(1), \ldots, C(n-1)$ can be used to compute the original signal via the inverse DCT given by:

$$s(i) = \sum_{j=0}^{n-1} \sqrt{\frac{\delta(j)}{n}} C_j(s) \cos\left(\frac{(2i+1)j\pi}{2n} \right)$$

In both equations, for any integer $m \geq 0$:

$$\delta(m) = \begin{cases} 1 & \text{if } m = 0 \\ 2 & \text{if } m > 0 \end{cases}$$

When the signal is multi–dimensional, as in image coding for example, the transform is applied along each dimension, one dimension at a time. Similarly to the Fourier transform, the DCT decomposes a signal into a weighted sum of cosine functions having different frequencies. Figure 2.7 shows the 64 base functions in which the bi–dimensional DCT used in the JPEG image coding standard decomposes each 8×8 block of pixels.

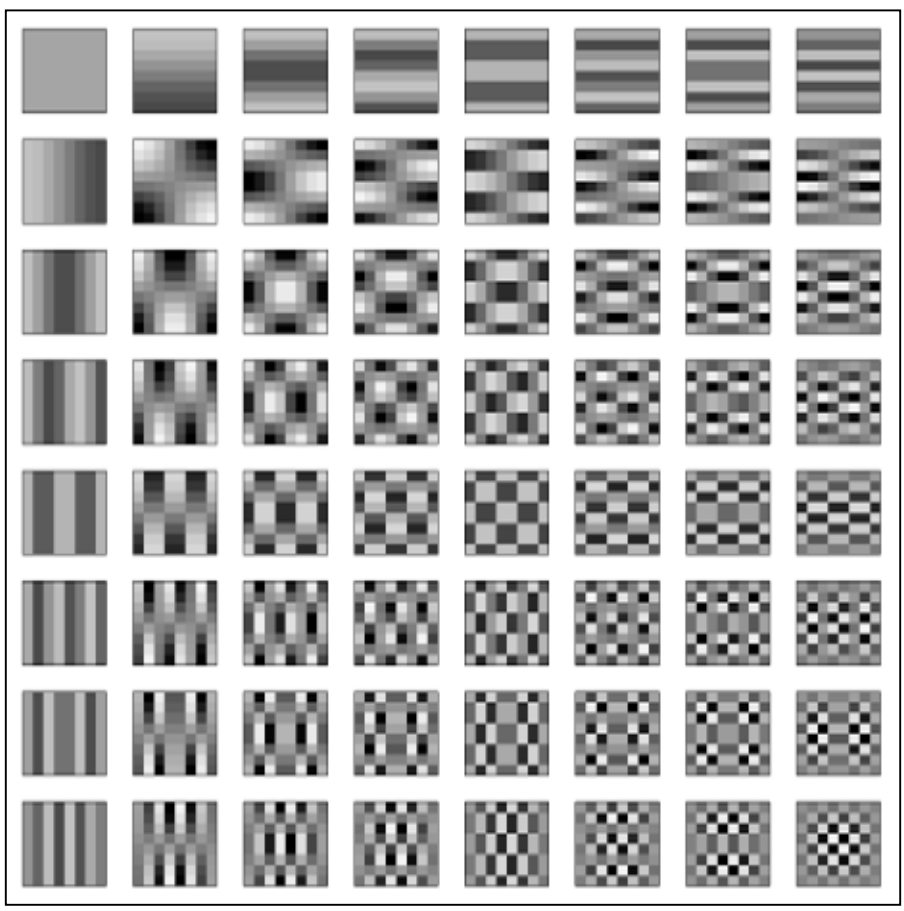


Figure 2.7: Bi–dimensional 8x8 DCT basis function.

2.3.10.2 Burrows–Wheeler Transform

A slightly expanding transform was proposed by M. Burrows and D. Wheeler in [1994]. When used on text data, it achieves performance comparable to the best available methods. It works by dividing the data stream in blocks and then encoding each block independently. Competitive performance is achieved only for blocks that contain more than 10.000 samples.

For each block of length n, a matrix M whose rows contain the n possible "rotations" is constructed (see Figure 2.8.1 for the method applied to a small string). The rows of this matrix are sorted in lexicographic order (see Figure 2.8.2). The transformed sequence is composed of the last column of the matrix augmented by the position of the index in the matrix of the original string. This information is sufficient to recover the original sequence via an inversion algorithm also described in Burrows and Wheeler [1994].

The rotations place in the last column the letter immediately before the letters in the first columns (contexts).

When the matrix is sorted, similar contexts are grouped together and this results in letters that have the same context being grouped together in the last column. See for example Figure 2.8.2, where in the transformed sequence, four "A" and two "B" are grouped together. Compression is achieved by encoding the transformed sequence with a move–to–front coding and then by applying run length and entropy coding.

A	**B**	**R**	**A**	**C**	**A**	**D**	**A**	**B**	**R**	**A**
B	R	A	C	A	D	A	B	R	A	A
R	A	C	A	D	A	B	R	A	A	B
A	C	A	D	A	B	R	A	A	B	R
C	A	D	A	B	R	A	A	B	R	A
A	D	A	B	R	A	A	B	R	A	C
D	A	B	R	A	A	B	R	A	C	A
A	B	R	A	A	B	R	A	C	A	D
B	R	A	A	B	R	A	C	A	D	A
R	A	A	B	R	A	C	A	D	A	B
A	A	B	R	A	C	A	D	A	B	R

Figure 2.8.1: Matrix M containing the rotations of the input string "ABRACADABRA".

A	A	B	R	A	C	A	D	A	B	R
A	B	R	A	A	B	R	A	C	A	D
A	**B**	**R**	**A**	**C**	**A**	**D**	**A**	**B**	**R**	**A**
A	C	A	D	A	B	R	A	A	B	R
A	D	A	B	R	A	A	B	R	A	C
B	R	A	A	B	R	A	C	A	D	A
B	R	A	C	A	D	A	B	R	A	A
C	A	D	A	B	R	A	A	B	R	A
D	A	B	R	A	A	B	R	A	C	A
R	A	A	B	R	A	C	A	D	A	B
R	A	C	A	D	A	B	R	A	A	B

Figure 2.8.2: The matrix $M' = \text{sort\_rows}(M)$. Transformed output is given by the position of the input string in M' (third row) followed by the string "RDARCAAAABB".

While the matrix M makes the description of this transform simpler and is universally used in the literature, a practical implementation, including the one presented in the original paper, will not explicitly build this matrix.

Burrows–Wheeler transform and derived block sorting transforms have been proved to be equivalent to some dictionary method like PPM*, described in Moffat [1990]. A detailed study and some possible improvements on this algorithm are presented in Fenwick [1996]. Sadakane [1998, 1999] and Balkenol, Kurtz and Shtarkov [1999] also propose improved algorithms. A study of the optimality of the BWT can be found in Effros [1999].

One of the main problems presented by this transform is its sensitivity to errors. When a single error is present a whole block is corrupted and cannot be decoded properly. An error resilient variant has been proposed in Butterman and Memon [2001].

2.3.10.3 Wavelets

Another decomposition scheme that adopts transform coding is based on wavelet functions. The basic idea of this coding scheme is to use as a base for the new representation functions that have the best compromise between time and frequency localization (Fourier and Cosine transform are not of this type).

Wavelets process data at different scale or resolution by using filter–bank decomposition. Filtering uses versions of the same prototype function ("mother wavelet" or "wavelet basis") at different scales of resolution. A contracted version of the basis function is used for the analysis in the time domain, while a stretched one is used for the frequency domain.

Localization (both in time and frequency) means that it is always possible to find a particular scale at which a specific detail of the signal may be outlined. It also means that wavelet analysis reduces drastically the amplitude of the input signal. This feature is really appealing for image compression since it means that most of the data is almost zero and then easily compressible. For further reading, see Vetterli and Kovacevic [1995].

The lossy image compression standard JPEG–2000 is based on wavelet decomposition.

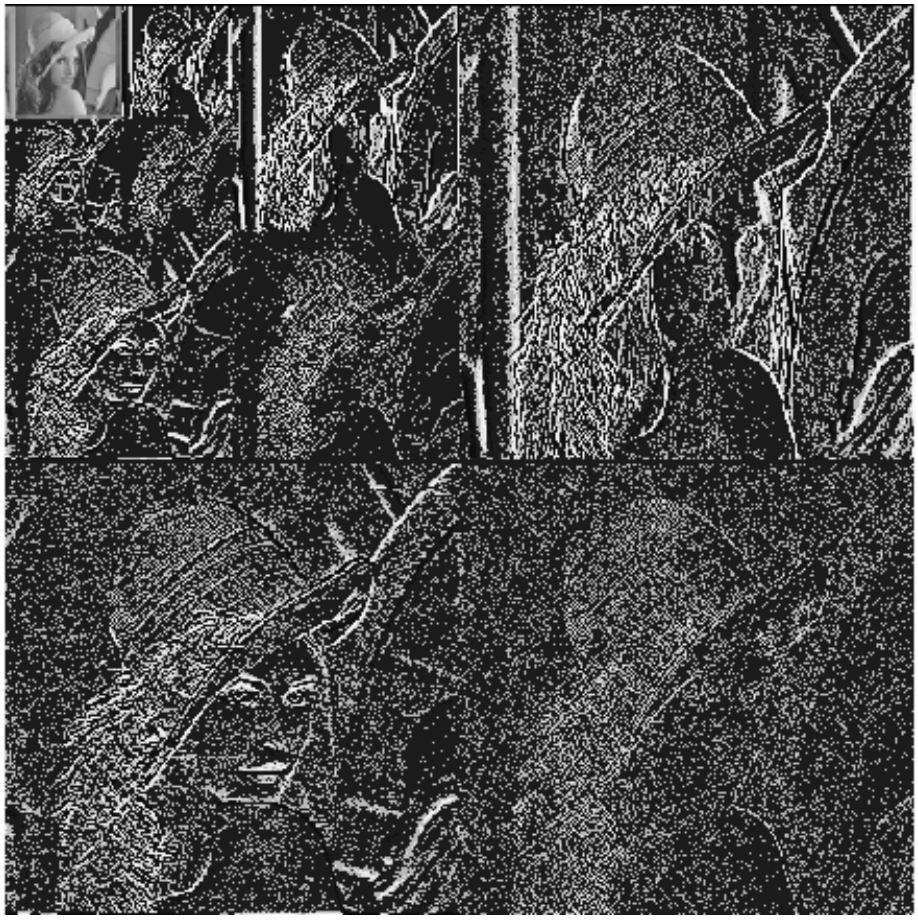


Figure 2.9: Wavelet decomposition used for image coding.

2.3.10.4 Overcomplete Basis

One of the most important features of the transformations that we have described so far is that they use a basis that is “minimal” or “complete’ in the sense that every signal has a unique representation in that space. While this results in fast analytical methods to find that representation, depending on the basis, the result may lead to a better or worse compression. It is well known for example that step functions have an infinite representation in a Fourier–like basis; conversely a sinusoid has an infinite representation in a basis that uses Haar wavelets (that look like step functions).

A technique that has recently received attention from many researchers is based on a decomposition into overcomplete bases. The rationale for the use of overcomplete dictionaries is that non–uniqueness gives the possibility of choosing among many representations the one that is most compressible, for example, the representation with the fewest significant coefficients.

Several new decomposition methods have been proposed, including Method of Frames (MOF), Matching Pursuit (MP), Basis Pursuit (BP) and for special dictionaries, the Best Orthogonal Basis (BOB).

Matching Pursuit is a greedy algorithm proposed in Mallat and Zhang [1993]; it decomposes a signal into a linear expansion of waveforms that are selected from an overcomplete dictionary of functions. These functions are chosen in order to best match the signal structure.

Matching pursuit decomposes a signal $f(t)$ by using basis functions $g_\gamma(t)$ from a dictionary G. At each step of the decomposition, the index γ that maximizes the absolute value of the inner product $p = \langle f(t), g_\gamma(t) \rangle$ is chosen. The value p is the expansion coefficient of the dictionary function $g_\gamma(t)$. The residual signal $R(t) = f(t) - p * g_\gamma(t)$ can be further expanded until a given number of coefficients is determined or until the error falls below a given threshold. After M stages, the signal is approximated by $\hat{f}(t) = \sum_{n=1}^{M} p_n * g_{\gamma_n}(t)$. If the dictionary is at least complete, the convergence (but not the speed) of $\hat{f}(t)$ to $f(t)$ is guaranteed.

Matching pursuit was used with very good results as an alternative of DCT transform in low bit rate video coding by R. Neff and A. Zakhor [1995, 1997].

In video coding applications, for example, the dictionary is composed of several scaled, translated and modulated Gabor functions and it is used to encode the motion compensated residual. Every function in the dictionary is compared with each part of the image maximizing the absolute value of the inner product. The best function is selected, the residual determined and the result eventually further encoded with the same method.

The computation can be very intensive, so techniques have been proposed to speed up the search. In Neff [1994] and in Neff and Zakhor [1995, 1997] a dictionary of separable Gabor functions is used, and an algorithm for a fast inner product is proposed. A heuristic to speed up the search can be also a preprocessing of the signal to match only the parts of the signal than have high energy. Matching pursuit shows very good performance and, with respect the DCT based codecs, has the advantage that it is not "block based", so the blocking artifacts although still present, are less evident.

Another method that decompose a signal into an overcomplete dictionary, Basis Pursuit (Chen [1995], Chen, Donoho and Saunders [1994, 1996]), uses a convex optimization closely related to linear programming. It finds a signal representation where the superimposition of dictionary elements is optimal with respect the smallest L_1 norm of coefficients among all such decompositions. Formally BP involves the solution of the linear programming problem:

$$\min \|\vec{\alpha}\|_1 \text{ subject to } \sum_\gamma \alpha_\gamma * \varphi_\gamma = f$$

BP in highly overcomplete dictionaries leads to complex large–scale optimization problems that can be attacked only because of the recent advances in linear programming. For example, the decomposition of a signal of length 8192 into wavelet packet dictionary requires the solution of an equivalent linear program of size 8192 by 212992.

2.3.11 Fractal Coding

Fractal based algorithms have very good performance and high compression ratios (32 to 1 is not unusual); their use is however limited by the intensive computation required. Fractal coding can be described as a "self Vector Quantization", where a vector is encoded by applying a simple transformation to one of the vectors previously encoded. Transformations frequently used are combinations of scaling, reflections and rotations of another vectors.

Unlike vector quantization, fractal compressors do not maintain an explicit dictionary and the search can be long and computationally intensive. Conceptually, each encoded vector must be transformed in all the possible ways and compared with the current one to determine the best match. Due to the fact that the vectors are allowed to have different size, there is very little "blocking" noise and the perceived quality of the compressed signal is usually very good. For further reading, see the books of Barnsley and Hurd [1993] or Fisher [1995].

2.4 Inter Band Decorrelation

It is common to have data that are oversampled by taking measures of the same event from several reference points that are different in time, space or frequency. When this happens, the data are said to be constituted by "channels", each carrying information on a coherent set of measures. A typical example is the stereophonic audio signal in which, in order to reconstruct the spatial position of the sound source, two microphones placed in

two different spatial locations record the sound produced by the source. Since the channels measure the same event, a large amount of information is common to the two sets of measures and efficient compression methods can be designed to exploit this feature.

A higher number of channels (twelve) are present in an electro cardiogram signal (ECG or EKG) where each channel records the heart's electrical activity from a probe placed in a specific body area (Womble et al. [1977]). As it is possible to see in the Figure 2.10, the twelve measures of an ECG are quite different one from the other while showing strikingly common patterns.

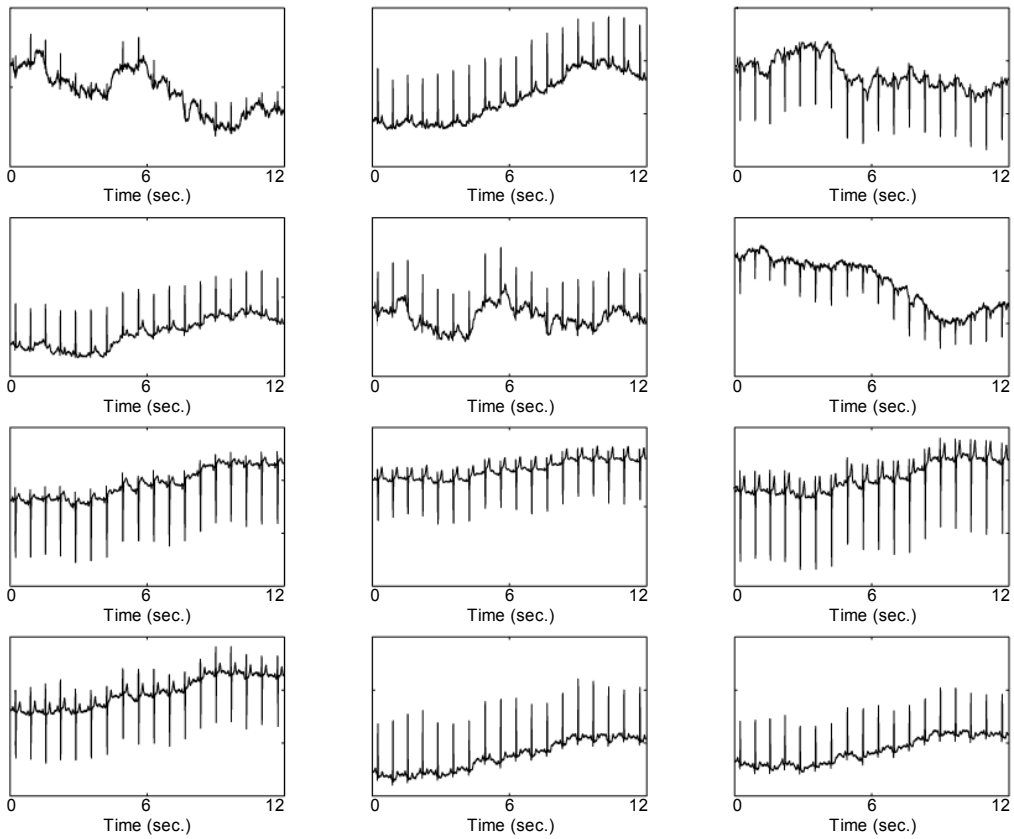


Figure 2.10: The twelve channels of an EKG signal.

Another case of multi channel data is constituted by color images; in digital color images the same scene is represented with three measures of luminosity taken in three frequency domains or "bands" (see Limb, Rubinstein and Thompson [1977] and Chau et al. [1991]).

A generalization is the case of multi and hyper spectral images that combine in the same picture a number of readings (sometimes as many as 224) taken in narrow and regularly spaced frequency bands (Markas and Reif [1993]). In some extent, even the video signal can be included in the category of multi–channel data in the sense that in a video signal several pictures of the same scene are taken at uniformly spaced time intervals.

In these cases, since all channels represent measures of the same event, it is reasonable to expect correlation between these measures. A number of techniques have been designed in order to exploit this correlation to achieve higher compression. We will discuss some of these techniques in the next paragraphs.

2.4.1 Color Decorrelation

Pixels in a color digital image are usually represented by a combination of three primary colors; red, green and blue as in an RGB scheme for example. This representation is "hardware oriented" in the sense that computer monitors generate the color image by combining these three primary colors.

Being that the three signals are luminosity measures of the same point, some correlation between their values is expected, and alternative representations have been designed in order to take advantage of this correlation.

When the color image is lossily encoded and destined to be used by human users in non–critical applications, it is possible to take advantage of the variable sensitivity to colors of the human visual system (Pennebaker and Mitchell [1993]). This is a method widely used in the color schemes that have been developed for commercial TV broadcasting.

The scheme that was used at first in the PAL analog video and subsequently adopted in CCIR–601 standard for digital video is named YC_bC_r (or YUV); the transformation between RGB and YC_bC_r is a linear transformation which uses the following equations (Gonzalez and Woods [1992]):

$$
\begin{aligned}
Y &= 0.299 * R + 0.587 * G + 0.114 * B \\
C_b &= B - Y \\
C_r &= R - Y
\end{aligned}
$$

This color representation divides the signals into a luminosity component Y and two chrominance components C_b and C_r, so by using the lower sensitivity of the human eye to color changes it is possible to achieve some compression by representing more coarsely the chromatic information. The PAL standard, for example, allocates a different bandwidth to each component: 5MHz are allocated to Y and 1.3 MHz to U and V components (U and V are a scaled and filtered version of C_b and C_r). This representation also has the advantage of being backward compatible with black and white pictures and video. Extracting a black and white picture from a YC_bC_r color image is equivalent to decoding the Y component only.

Similarly, in digital video the chrominance components C_b and C_r are frequently sub–sampled at a lower resolution than Y by following one of these conventions (Bhaskaran and Konstantinides [1995]):

- **4:4:4** Represents the original signal. Each pixel is encoded as three bytes: one for the luminance and two for the chrominance;
- **4:2:2** Color components are horizontally sub–sampled by a factor of 2. Each pixel can be represented by using two bytes;
- **4:1:1** Color components are horizontally sub–sampled by a factor of 4;
- **4:2:0** Color components sub–sampled in both the horizontal and vertical dimension by a factor of 2 between pixels.

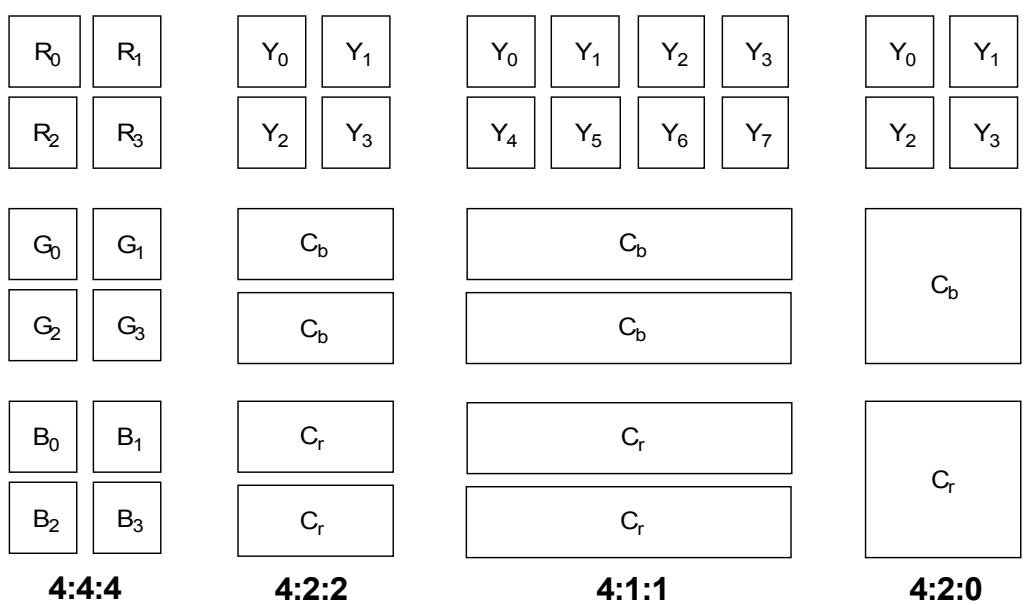


Figure 2.11: Color Subsampling Formats.

2.4.2 Motion Compensation

A particular kind of inter–band prediction used in video compression to exploit temporal redundancies between consecutive frames is called “motion compensated prediction”. It assumes that two consecutive video frames represent the same scene in which some objects have been displaced because of their relative motion. So, instead of predicting the current frame from the previous encoded one as in a DPCM scheme, the frame is divided in blocks and each block is individually matched with the closest block in the previous frame. This process is called “motion compensation” because a block is thought to be a displaced version of a corresponding block in the past frame. The offset between the two blocks is sent to the decoder as a “motion vector” and the difference is performed between a block and its displaced match.

The decoder, which stores the past frame in a memory buffer, inverts the process by applying the motion vectors to the past blocks and recovers the reference blocks that must be added to the prediction error. No motion estimation is necessary on the decoder side.

The Block Matching Algorithm (BMA) that is traditionally used to perform the motion compensation is a forward predictor that finds for each block the displacement vector minimizing the Mean Absolute Difference (MAD) between the current block and a displaced block in the previously encoded frame. In literature, quality improvements are observed in DCT based codecs that match blocks by using different error measures like the Geometric Mean of the DCT coefficient variance (Fuldseth and Ramstad [1995]) or the Spectral Flatness of the DCT coefficients (Fuldseth and Ramstad [1995]).

Block matching motion compensation is the most computationally intensive task in a DCT based codec. It is estimated that more than 60% of the coding time is spent in

performing the motion compensation. Several fast methods have been proposed to speed up block based motion compensation. Some of them use logarithmic or hierarchical search (Jain and Jain [1981]) or, as in Mandal et al. [1996], they use a multi–resolution approach suited for codecs based on the wavelet transform. Speed improvements are also obtained by limiting the range of the motion vectors. Experiments presented in Bhaskaran and Konstantinides [1995] show that for head and shoulders sequences, limiting the range of the motion vectors in a diamond shaped region causes only a very small quality loss in H.261 encoded sequences. Figure 2.11 shows the regions and the SNR achieved in a H.261 encoder for increasing bit rates; the optimal searching region is highlighted.

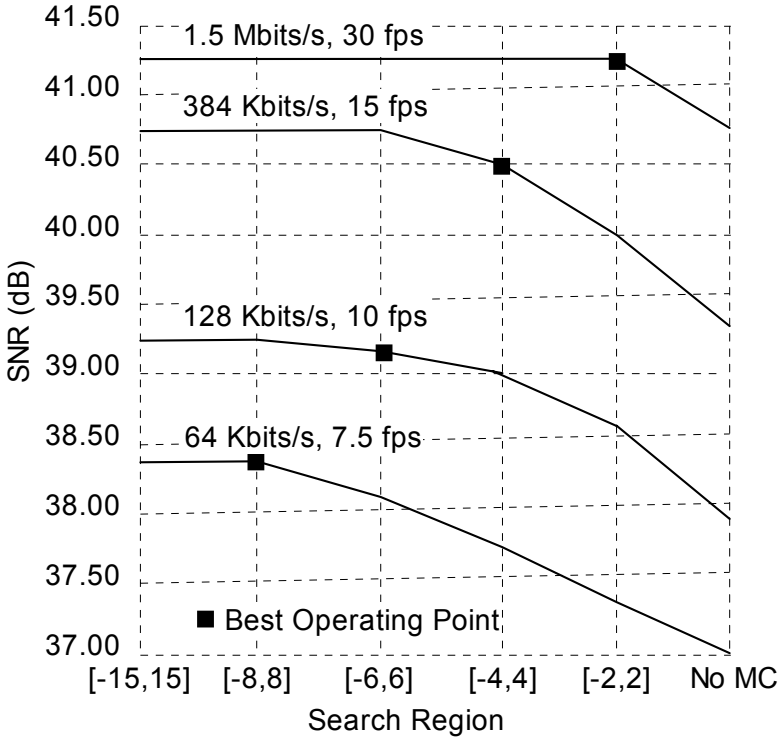


Figure 2.12: Search regions and quality improvements for typical H.261 encoding.

A completely different approach based on backward motion estimation is presented in Armitano, Florencio and Schafer [1996] with the name of "Motion Transform" (MT). Working solely on information available at both encoder and decoder, backward motion estimation has the advantage that it does not require the transmission of the motion vectors. The authors show that with this method it is possible to increase the PSNR quality of the reconstructed video by 2–4dB. The biggest drawback of the motion transform is that it lacks compatibility with the existing standards and that both encoder and decoder have to compute motion estimation.

2.4.3 Multi and Hyperspectral Images

In the last two decades, a technology consisting of the remote acquisition of high definition images has been successfully used both in military and civilian applications to recognize objects and classify materials on the earth's surface. High definition images can be acquired via a space borne platform or an air borne platform, transmitted to a base station and elaborated.

If we want to use an image to recognize objects, very high resolution is necessary. For example to recognize, let's say a corn field, a picture that shows details of the corn leaves is necessary. Such a picture would require the acquisition and elaboration of an enormous amount of data. The approach followed by *multispectral* and *hyperspectral* photography overcomes this problem by considering relatively large areas to be covered by a single pixel (typically of the order of ten square meters), but instead of decomposing the reflected light into three colors, it uses a wider spectrum ranging from infrared to ultraviolet and a band decomposition counting tens to hundreds of very narrow bands.

Since every material reflects sun light in its own peculiar way, the analysis of the spectrum of its reflected light can be used to uniquely recognize it. When a more sophisticated analysis is required, an increment in the spectrum resolution is technically feasible since it only increases data by a linear amount.

In practice such measures are affected by a number of different errors that complicate the interpretation of the image and the classification of a spectrum. For example, several materials can be present in the area covered by a single pixel, so in general, a single pixel will consist in a mixture (linear or non–linear) of several spectra. Clouds, shades, time of the day, season and many other factors affect the reading by changing the properties of the sunlight. Nevertheless, hyperspectral imagery has been used in the past with great success and shows incredible promise of future applications.

Typical algorithms used on hyperspectral images consist of dimensionality reduction, spectral unmixing, change detection, target detection and recognition. Since hyperspectral images are acquired at great cost and destined to applications that are not necessarily known at the time of the acquisition, particular care must be taken in order to assure that relevant data are not lost during lossy compression and ad-hoc quality measures must be used to insure a meaningful preservation of the quality.

2.5 Quality Assessment

One of the main problems in lossy compression consists in the assessment of the quality the compressed signal. Two main approaches can be taken to solve this problem, one involving objective measures and the other relying on subjective assessments. The two methodologies are not mutually exclusive and frequently are both used at different stages of the compressor design. Quality of data that are destined to be used by human users, like music, video or pictures can be assessed with subjective tests. First a reference quality scale is chosen, like for example the Mean Opinion Score used in the evaluation of digital speech, then a number of experts are asked to judge the quality of the compressed data. While subjective methods are very reliable and, when applicable, provide the best possible evaluation, the difficulty of conducting subjective tests and the need for "automatic" assessment has led to a great interest in sophisticate objective metrics that mimic closely the response of the human perceptual system.

Simpler objective measures have the advantage of being mathematically tractable and, with their help novel compression algorithms can be easily evaluated and their performance studied in great detail.

Objective distortion metrics are also used in "closed loop" encoders to control in real time the quality of the compression and dynamically control the parameters of the algorithm.

If we indicate an N-samples signal with $0 \le s(t) \le s_{fs}$, with $\bar{\sigma}_s^2$ its average variance and with $\hat{s}(t)$ the lossily compressed signal obtained after compression and decompression, the most common distortion measures take the following forms:

Mean Absolute Error (MAE) or L_1:

$$MAE = \frac{1}{N}\sum_t \left|s(t) - \hat{s}(t)\right|$$

Mean Squared Error (MSE) or L_2^2:

$$MSE = \frac{1}{N}\sum_t \left[s(t) - \hat{s}(t)\right]^2$$

Root MSE (RMSE) or L_2:

$$RMSE = \sqrt{MSE}$$

Signal to Noise Ratio (SNR):

$$SNR_{(dB)} = 10 * \log_{10} \frac{\bar{\sigma}_s^2}{MSE}$$

Peak SNR (PSNR):

$$SNR_{(dB)} = 10 * \log_{10} \frac{s_{fs}^2}{MSE}$$

Peak error or Maximum Absolute Distortion (MAD) or L_∞:

$$MAD = \max_{(t)} \left\{\left|s(t) - \hat{s}(t)\right|\right\}$$

Percentage Maximum Absolute Distortion (PMAD):

$$PMAD = \max_{(t)} \left\{\frac{\left|s(t) - \hat{s}(t)\right|}{s(t)}\right\} * 100$$

Most measures are derived from the Mean Squared Error because MSE has a simple mathematical expression, is derivable and gives a good measure of random errors. MSE and Signal to Noise Ratio capture an average behavior and very little can be said on the

error that affects single samples. Because of these characteristics, MSE derived metrics are more useful in the evaluation of high bit rate signals.

The Mean Absolute Error and the derived PMAD are mainly used when the encoding must guarantee an error always smaller that a given threshold. Systems in this category are frequently called “near lossless” and their main application is the compression of scientific and medical data.

While these measures are widely used, it can easily be shown that they have little to do with how the distortions are perceived by humans The reason of this poor correlation between SNR and measures obtained via subjective observations is mostly due to the fact that, unlike the human perceptual system, these measures are not sensitive to structured or correlated errors. Structured errors, frequently present in encoded data are known to degrade local features and perceived quality much more than random errors do. The human perceptual system is more sensitive to structured patterns than to random noise and more sophisticate methods have to be used to assess carefully the quality of compressed digital audio, pictures and video. Some perceptually motivated methods that are used to assess digital images and video will be discussed in the next sections.

2.5.1 Digital Images

In some applications the quality of the encoded picture is not very important, for example, in off–line editing it can be necessary only to recognize the pictures to be able to make cutting decisions. In others, especially in distribution and post–production, quality is crucial and carefully supervised.

The coding errors that are introduced by state of the art encoders are mostly structured, and they are not easily modeled in terms of SNR or MSE. In particular at very low bit rates, most coding schemes show visible artifacts that can impoverish the perceived quality. Several artifacts commonly observed in low bit rate image compression are:

- **Blocking**: Occurs in techniques that involve partitioning of the image into blocks of the same size. It appears to be the major visual defect of coded images. Blocking is caused by the coarse quantization of the low frequency components in an area where the light intensity changes gradually. The high frequencies introduced by the quantization error enhance block boundaries. Blocking artifacts are commonly encountered in DCT–based, VQ and fractal–based compression algorithms.
- **Overall Smoothness** or **Blurring**: Is a very common artifact present in the conventional TV standards (NTSC, PAL or SECAM). It occurs also for digital coding techniques at low bit rate and appears in different forms such as the edge smoothness due to the loss of high frequency components, texture and color blur due to loss of resolution. Although segmentation–based coding techniques claim to preserve the major edge components in the image, they often smooth out smaller edge components.
- **Ringing Effect** or **Mosquito**: The ringing effect is another common visual distortion that is observable as periodic pseudo–edges around the original edges in a DCT–compressed, sub–band or wavelet compressed image. It is also visible in the textured region of compressed images where it appears as a distorted texture

pattern. Ringing is caused by the improper truncation of high frequency components. This artifact is also known as Gibbs effect.

- **Texture Deviation**: A distortion called texture deviation, is caused by loss of fidelity in mid–frequency components, and appears as a granular noise or as the "dirty window" effect. The human eye is less sensitive to texture deviation in textured areas with transform–based coding, but texture deviation is often present as an over smoothing of texture patterns, that turns out to be visually annoying.
- **Geometrical Deformation**: In model–based coding, objects in an image (a human face, for example) are compressed by using a geometric model. This compression approach, suitable for very low bit rate coding, may show geometrical deformations, namely the synthesis procedure may change shape and position of some crucial features and lead to perceptual inconsistency.

All these artifacts are also commonly encountered in video encoding. Many algorithms were proposed to decrease the effects of these coding artifacts in transform–based codecs. Most of them involve out–of–the–loop pre or post filtering (see for example Lai, Li and Kuo [1995, 1996], Joung, Chong and Kim [1996] and Jacquin, Okada and Crouch [1997]). The main advantage of this class of techniques is that, since they introduce only a pre or post filtering of the signal, full compatibility with coding standards is preserved. Their main drawback is that filtering is likely to produce unnecessary blurring of the image.

More sophisticated techniques to reduce compression artifacts involve in–loop perceptive measures to drive the bit allocation in standard algorithms. One of these

methods is the Picture Quality Scale (or PQS), proposed by Miyahara, Kotani and Algazi [1996] as an objective metric for still, achromatic pictures. PQS transforms the coding error in five perceptually relevant signals $F_1,...,F_5$ also called Distortion Factors and combines them in a single numeric value by using a regression method. This value, representative of the quality of the given image, is a very good approximation of the Mean Opinion Score (MOS), a subjective scale widely used for the evaluation of the image quality.

The factors considered in PQS are:

- **Distortion Factor** F_1: is the frequency weighted error defined by the CCIR 567 standard;
- **Distortion Factor** F_2: is an error obtained with a different frequency weighting and includes a correction that takes in account Weber's law (see Carterette and Friedman [1975]);
- **Distortion Factor** F_3: measures the horizontal and vertical block discontinuities that are evident in most image coders;
- **Distortion Factor** F_4: measures errors that are spatially correlated. This factor captures textured errors that are well–known to be easily perceived;
- **Distortion Factor** F_5: measures error in the vicinity of high contrast image transitions because errors are more evident when located in high contrast zones.

Error indicators contribute to more than one factor, so it is necessary to use a principal component analysis to decorrelate the distortion factors before composing them into a single value that is representative of the global image quality.

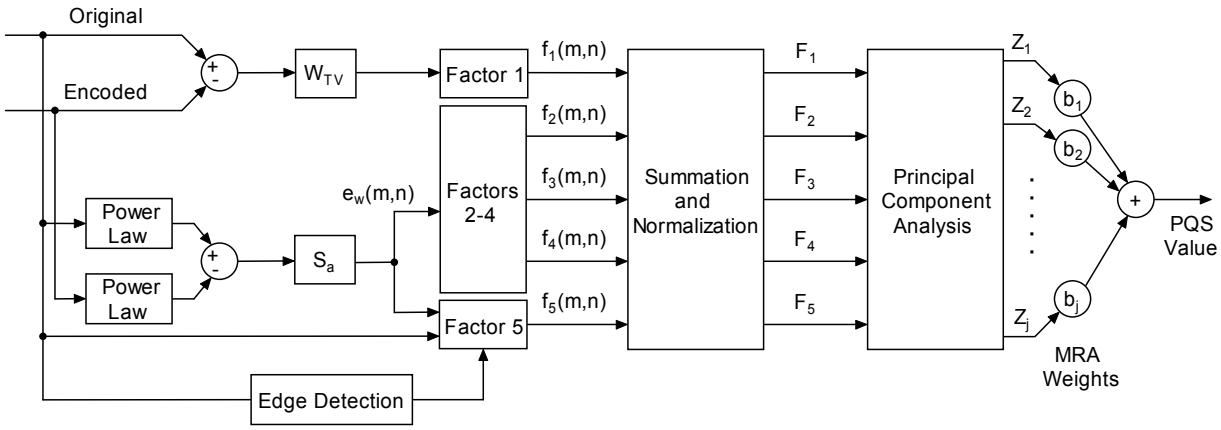


Figure 2.13: PQS.

Figure 2.13 shows the system proposed in Miyahara et al. [1996]. PQS has been also by Lu, Algazi and Estes [1996] to compare wavelet image coders and to improve the quality of a high quality image codec. The main drawback of this system is due to the fact that it is formulated for still achromatic pictures (so it is of little use in color imaging) and that the distortion factors are determined from the whole image, so they do not represent local distortions.

2.5.2 Video

The lack of perceptually motivated distortion measures is particularly relevant in video coding, where a strong variability of the performance is usually observed. Large moving objects, complex scenes and fast scene changing are all cause of extreme variability and it is a well–known fact that no compression technique works well for all scene types (Pearson [1997]).

For example, it is well known that line interlacing, one of the earliest compression methods, produces patterns on certain types of moving object. Color compression schemes such YIQ and YUV used in NTSC or PAL exhibit severe cross–color effects in high spatial–frequency areas. Block–based transform coding, as discussed earlier, has been known to have difficulty with diagonal lines traversing a block. Fractal coding may work spectacularly well with certain types of iterated structure but not so well with others. Model–based coding does not work very well if new objects keep entering the scene, etcetera. Code switching was proposed as a solution to the problem of variability in video coding, nevertheless, a good criterion to drive the switch is still required.

The perception of a video sequence is a complex phenomenon that involves spatial and temporal aspects. Besides all the static characteristics of the visual system (edge and pattern sensitivity, masking, etc.), studies of viewer reaction to variable–quality video have identified the end–section of the video sequence and the depth of the negative peaks as being particularly influential in the evaluation of the quality (Pearson [1997]).

Van den Branden Lambrecht [1996] has proposed a metric specifically designed for the assessment of video coding quality. This quality measure named Moving Pictures Quality Metric (or MPQM) is based on a multi–channel model of the human spatio–temporal vision and it has been parametrized for video coding applications. MPQM decomposes the input signal in five frequency bands, three spatial directions and two temporal bands. MPQM also takes in account perceptive phenomena as spatial and temporal masking.

A block diagram for the proposed system is depicted in Figure 2.14; the input sequence is coarsely segmented into uniform areas by looking at the variance of the

elementary blocks. Both original and reconstructed video sequences are transformed by using a perceptual decomposition. The signal is decomposed by a filter–bank in perceptual components grouped in 5 frequency bands, 3 spatial directions and 2 temporal bands. Contrast sensitivity and masking are calculated and the results are used to weight the transformed decoded sequence.

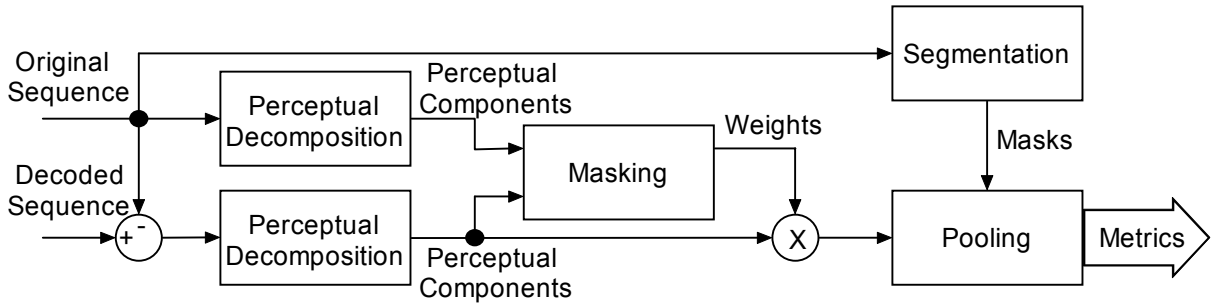


Figure 2.14: MPQS.

Another method, mainly proposed for automatic assessment of video coding algorithms, is Motion Picture Quality Scale or MPQS (van den Branden Lambrecht and Verscheure [1996]). MPQS performs a simple segmentation on the original sequence, by dividing the frames into uniform areas. By using the segmentation, the masked data are pooled together to achieve a higher level of perception. A multi–measurement scheme is proposed as output of the pooling and a global measure of the quality, and some detailed metrics are computed. Measures evaluate quality of three basic components of images: uniform areas, contours and textures.

MPQS was used with some modification in Verscheure and Garcia Adanez [1996] to study the sensitivity to data loss on MPEG–2 coded video streams transmitted over an ATM network and in Verscheure et al. [1996] to define a perceptual bit allocation strategy for MPEG–2.

DATA COMPRESSION STANDARDS

3.1 Audio

3.1.1 Pulse Code Modulation

Pulse Code Modulation (PCM) is the simplest form of waveform coding since it compresses an analog signal by applying only sampling and quantization. It is widely used both in high quality audio encoding and in speech coding. A popular PCM format is the Compact Disc standard, where each channel of a stereophonic audio signal is sampled at 44.1 KHz per channel, 16 bit per sample.

3.1.2 MPEG Audio

One of the tasks of the Movie Picture Expert standardization Group (MPEG) was the definition of an audio coding standard suitable for perceptually "transparent" audio compression at bit rates comprised between 128 Kb/s and 384 Kb/s. When the input is a PCM stereo audio signal, sampled at 44.1 KHz, 16–b/sample (audio CD) this results in a compression factor ranging between 4 and 12.

Three layers of increasing delay, complexity and performance were defined in the MPEG–1 audio coding standard: Layers I, II and III. Since each layer extends the features of its predecessor, the layer organization retains backward compatibility. A decoder that is fully compliant with the most complex Layer III, for example, must be able to decode bitstreams created by Layer I and II encoders. However, in practice when power consumption and cost efficiency are critical constraints, decoders compatible with a single layer only are not uncommon. A good description of Layers I and II can be found in Sayood [1996] and a more general discussion on the standard and the standardization process is in Noll [1997].

MPEG–1 Layer III (also known with the nickname of MP3) provides the highest compression and has recently increased its popularity due to the availability of inexpensive hardware players supporting this file format. The computing power of current microprocessors also makes feasible MP3 software–only encoders and decoders, making this format the most popular choice for the exchange of audio files over the Internet.

Layer III is a hybrid subband and transform coding; it uses a perceptually justified bit allocation which exploits phenomena like frequency and temporal masking to achieve high compression without compromising the final quality.

PCM inputs sampled at rates of 32, 44.1 and 48 KHz are supported both in mono and stereo modes. Input bit rates match the common CD and DAT digital formats. Four encoding modes are available: mono, stereo, dual and joint stereo, where the dual mode is used to encode two channels that are not correlated, like for example a bilingual audio track. More interesting is the joint stereo mode in which a modality called “intensity

stereo coding" is used to exploit channel dependence. It is known that above 2 KHz and within each critical band, the perception of a stereo image is mostly based on the signal envelope and it is not influenced by the details of the temporal structure. This phenomenon is used to reduce the bit rate by encoding subbands above 2KHz with a signal L+R that is the composition of the left and right channels and a scale factor that quantifies the channels' relative intensities. The decoder reconstructs left and right channels by multiplying the composite L+R signal by the appropriate scale factor. While this results in two signals that have same spectral characterization, their different intensity is sufficient to retain a stereoscopic image.

Other psychoacoustic phenomena are exploited within MPEG–1 audio standard. In particular auditory masking is used to determine a perceptually transparent bit allocation for each critical band and temporal masking is used to reduce pre–echoes introduced by coarse quantization while in the presence of a sudden music attack. Since the input is divided into frames and each frame is encoded independently, a frame that contains a period of silence followed by a sudden attack (drums, for example) presents a peculiar problem. After the frame is transformed in the frequency domain and its representation quantized, the inverse transform spreads the quantization error uniformly in the time domain and the period of silence preceding the attack may get corrupted by this noise. When this condition is detected it is useful to reduce the size of the frame. This is not sufficient to prevent errors, but if the frame is small enough, the noise introduced before the attack is likely to be masked by it and the listener will not be able to perceive any quality degradation.

While Layers I and II are very similar (see Figure 3.1) and decompose input by using a filter bank, the more complex Layer III combines a filter bank with a cascaded Modified Discrete Cosine Transform (see Figure 3.2). In both cases, signal decomposition is followed by a perceptual bit allocation and the frequency domain representation of the input frame is quantized, each critical band having a different resolution. Bit allocation starts with a bit pool that depends on the target bit rate and distributes the bits to the single bands while trying to achieve a transparent quantization. Auditory masking is used to determine if a signal present in a critical band masks (raises the auditory threshold of) an adjacent band. When this happens, fewer bits are dedicated to the coding of the masked signals and more bits are allocated to the masker, since this is likely not to result in any audible error.

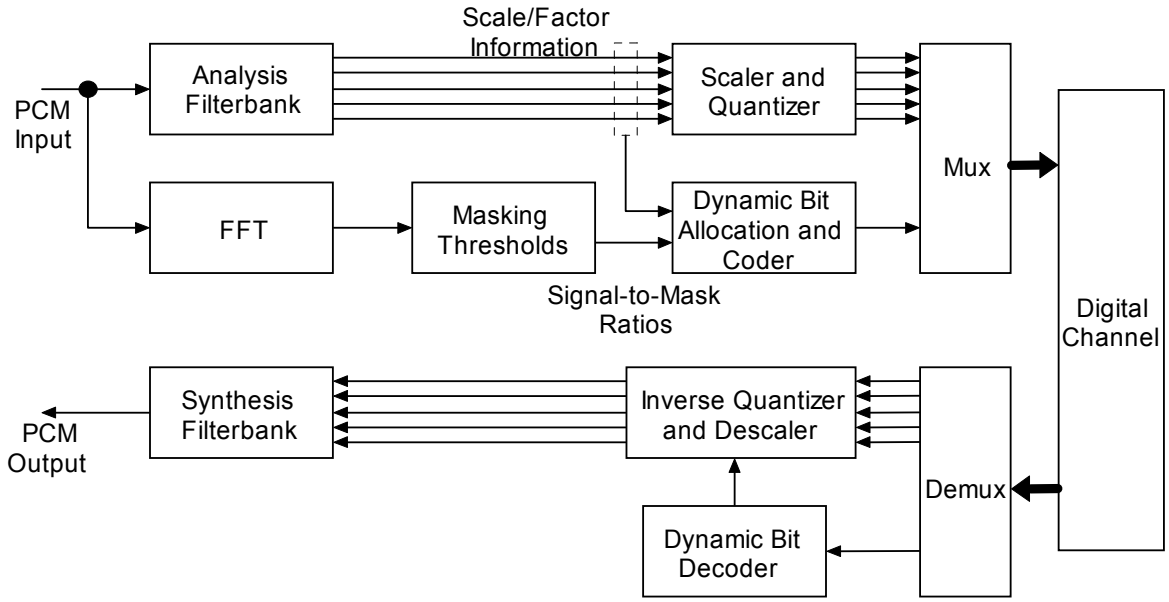


Figure 3.1: MPEG–1 Layers I and II.

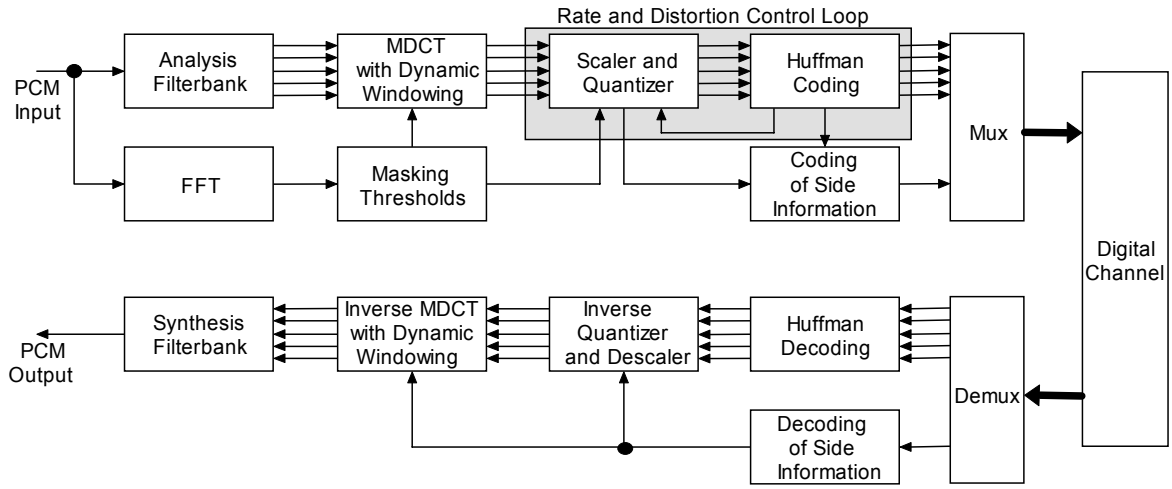


Figure 3.2: MPEG–1 Layer III.

Another feature of Layer III is the use of entropy coding based on static Huffman codes. Perceptual coding makes MPEG–1 audio highly asymmetrical and the encoder is generally more complex than the decoder. Like in other standards, standardization only addresses the bitstream format and, for example, it does not cover any particular strategy to perform this perceptual coding. This is done in order to leave room for encoder improvements and it also allows the realization of very simple encoders that may not use any psychoacoustic model at all.

MPEG–2 audio enhances MPEG–1 by adding a number of features that make the new standard more flexible. Input sampling frequencies are extended to cover medium band applications with 16, 22.05 and 24 KHz.

To support stereo surround, the number of channel is extended from a maximum of two to a maximum of five high–quality, full–range channels (Left, Center, Right, Surround Left and Surround Right) plus the possibility of connecting an additional subwoofer in a configuration called 5.1. When used in this configuration, some backward compatibility with MPEG–1 is retained (see Figure 3.3).

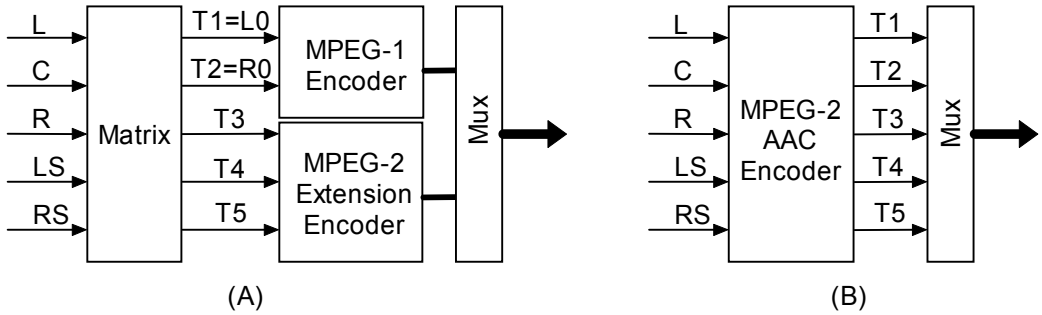


Figure 3.3: MPEG–2 backward compatible configuration (A) vs. AAC (B).

An advanced mode that is not backward compatible was also defined by the standard. This mode, called Advanced Audio Coding or AAC, defines a number of tools arranged in three different profiles, defined to cover typical applications with different quality requirements and coding complexity. Tools include high-resolution filter banks, preprocessing, prediction, and of course, perceptual coding. AAC is intended for both consumer and professional applications and supports up to 46 channels while having as a default, the mono/stereo mode and the 5.1 configuration described before.

3.2 Speech

3.2.1 A–law and μ–law

When PCM is used to encode the narrowband speech signal, a sampling frequency of 8 KHz is sufficient to achieve an intelligible reconstruction of the original. With a linear quantizer, samples are individually represented with a precision of 12 bit and the resulting bit rate is 96 Kbit/s. It has been found that a non–linear quantization at 8 bit per

sample is able to encode speech signals with a quality that is almost indistinguishable from a signal linearly quantized at 12 bit per sample.

The non–linear quantizers used in practice are, in general, logarithmic; this class of quantizers is justified by two reasons:

- The response of the human hear is not proportional to the intensity of the stimulus and low intensity signals are discriminated with higher accuracy;
- Speech samples are distributed according to a Gaussian distribution and choosing a logarithmic step minimizes the average quantization error.

In the 1960s, two non–linear PCM codecs with a bit rate of 64 Kbit/s were standardized: μ–law in the United Stated and A–law in Europe. Because of their simplicity, excellent quality and very low delay, both methods are still widely used today in telephony.

If x is the value of a sample and $x_{\max}$ is the maximum value that a sample can assume, the value of x quantized with the μ–law is given by:

$$\hat{x} = x_{\max} \frac{\log_e\left(1 + \frac{\mu \cdot x}{x_{\max}}\right)}{\log_e(1+\mu)}$$

where $\mu = 255$ in the American and Japanese PCM standards.

Similarly, the sampled value for x when using the A–law is given by:

$$\hat{x} = \begin{cases} \dfrac{\frac{A \cdot x}{x_{\max}}}{1 + \log_e A}; & if \quad 0 \le \dfrac{|x|}{x_{\max}} \le \dfrac{1}{A} \\ \dfrac{1 + \log_e\left(\frac{A \cdot x}{x_{\max}}\right)}{1 + \log_e A}; & if \quad \dfrac{1}{A} \le \dfrac{|x|}{x_{\max}} \le 1 \end{cases}$$

where $A = 87.56$ for the European PCM standard.

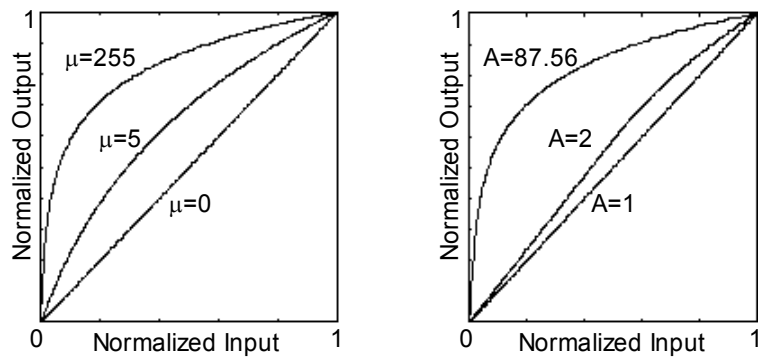


Figure 3.4: μ–law vs. A–law for different μ and A parameters.

3.2.2 Differential PCM

In most signals, temporally adjacent samples are frequently correlated. Differential PCM exploits this feature and achieves some compression by representing a signal as a succession of differences $\Delta(n) = x(n) - \tilde{x}(n)$ between $\tilde{x}(n)$, a causal prediction of the current sample, and $x(n)$, the actual sample value. If the prediction is accurate, then the difference signal has lower variance than the original samples, and it will be accurately quantized with fewer bits. The decoder reconstructs the signal by adding the quantized differences to the predicted values (see Jayant and Noll [1984]).

The best predictor is defined to be the one that minimizes the mean square error, so the best prediction for $x(n)$ is equal to the conditional expectation of $x(n)$:

$$E\left[X(n) \mid X(n-1), X(n-2), \ldots\right].$$

This predictor is not practical because:

- The conditional probability distribution necessary to evaluate E will not in general be available;

- To maintain decoder synchronization without sending any side information, it is necessary to base the prediction on the quantized values $\hat{x}(n-i)$ instead of the original samples $x(n-i)$ that are available only to the encoder.

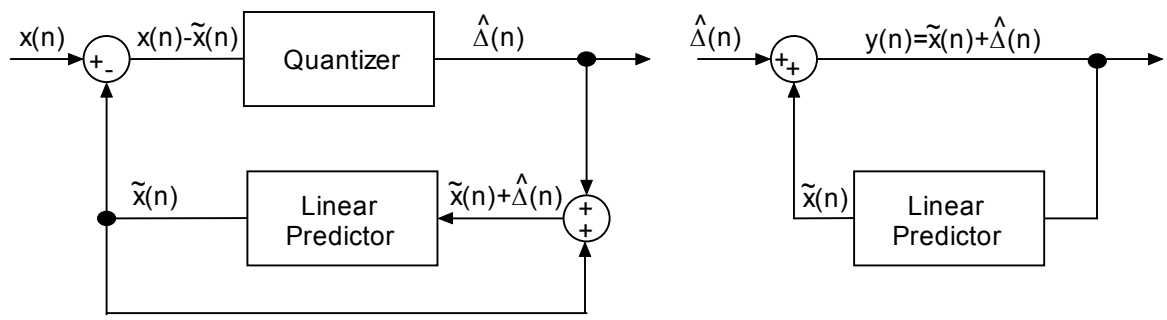


Figure 3.5: Closed–Loop DPCM.

These considerations lead to the definition of the "closed–loop" DPCM encoder depicted in Figure 3.5, where the linear predictor of N–th order has the form:

$$\tilde{x}(n) = \sum_{i=1}^{N} h_i \cdot \hat{x}(n-i)$$

In the previous equation $h_i, 1 \le i \le N$ are the coefficients of the predictor. The basic equations describing a DPCM are (see Figure 3.5):

$$\begin{aligned} \Delta(n) &= x(n) - \tilde{x}(n) \\ \hat{\Delta}(n) &= \Delta(n) - q(n) \\ y(n) &= \tilde{x}(n) + \hat{\Delta}(n) \end{aligned}$$

where $x(n)$ is the input, $\Delta(n)$ is the prediction error, $q(n)$ is the error introduced by the quantization, $\hat{\Delta}(n)$ is the quantized prediction error and $y(n)$ is the decoder output.

In its simplest form, DPCM assumes that the current sample is equal to the sample that has been previously encoded and quantized: $\tilde{x}(n) = \hat{x}(n-1)$. This assumption is

equivalent to setting $\Delta(n) = x(n) - \hat{x}(n-1)$ with $h_1 = 1$ and $h_i = 0$ for every $i > 1$. In this case the decoder acts as an "integrator".

3.2.3 LPC–10 (Linear Predictive Coding of 10th order)

Linear Prediction is one of the most powerful and historically one of the most important speech analysis techniques. The best low bit–rate speech codecs are based on linear prediction. Before introducing the standard LPC–10, it is appropriate to review some of the concepts underlying linear predictive coding. More details on this method can be found in Rabiner and Schafer [1978], Makhoul [1975] and Gersho [1994].

The idea behind linear prediction is the approximation of a speech sample with a linear combination of a number of past samples. Using this approach, the n–th sample s_n can be predicted as a weighted sum of p previous samples (linear prediction of order p):

$$\tilde{s}_n = \sum_{k=1}^{p} a_k \cdot s_{n-k}$$

where $a_1, a_2, ..., a_p$ are the prediction coefficients, optimized in order to reduce the error variance. The prediction error is expressed by:

$$e_n = s_n - \tilde{s}_n = s_n - \sum_{k=1}^{p} a_k \cdot s_{n-k}$$

This equation represents the output of a system whose transfer function is:

$$A(z) = 1 - \sum_{k=1}^{p} a_k \cdot z^{-k}$$

This filter is frequently called "whitening" filter because it transforms the signal into "white noise" by removing the correlation between samples (see Figure 3.6). Its inverse $H(z)$, necessary to the decoder to "reshape" the residual error, is given by:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k \cdot z^{-k}}$$

The basic problem of linear prediction is the determination of the coefficients $a_1, a_2, ..., a_p$ directly from the samples in such a manner as to obtain a good estimate of the spectral properties of the signal.

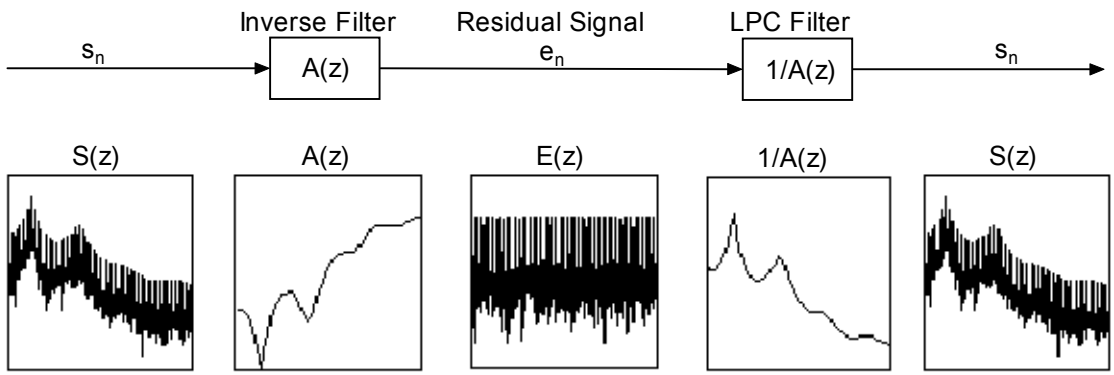


Figure 3.6: Linear Prediction effects in the frequency domain.

Several methods have been proposed for the efficient and accurate determination of the LP coefficients. The best algorithms are based on the solution of a linear system of equations that is obtained by taking the derivative of the error with respect to the prediction coefficients. Because of the special form that this matrix assumes on a speech signal, it is possible to solve the system of linear equation by using the Levinson and Durbin recursive algorithm (Rabiner and Schafer [1978]). Levinson and Durbin recursion

has several interesting properties, including a low computational complexity since it requires only $O(p^2)$ operations instead of $O(p^3)$ operations required by other methods.

The reason linear prediction works so well in removing the correlation existing between speech samples is that there is a close relation between the predictor coefficients $a_1, a_2, ..., a_p$ and a physical model of the human vocal tract.

Voice is produced by forcing a flow of air from the lungs into the vocal tract. During the production of voiced sounds, air is forced through the vocal folds that vibrate at a frequency comprised between 50 to 500 times per second. Voiceless sounds are produced by the airflow being forced into some obstacle in the mouth, or exiting with a sudden burst.

In both cases, it is the position of the vocal tract during the production of the sound that controls the characteristics of the sound. The vocal tract acts as a resonator by shaping an "excitation" signal while changing its position in time.

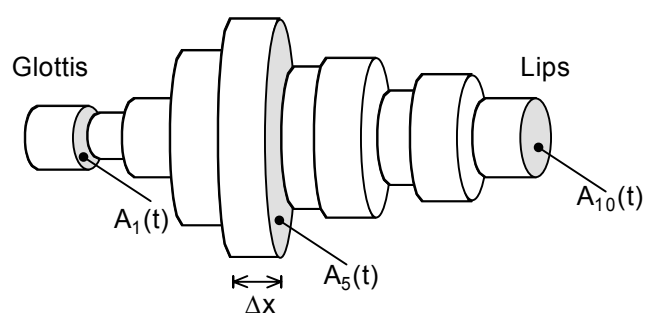


Figure 3.7: Vocal tract – Speech production model.

If we model the vocal tract as a concatenation of lossless tubes that have same length but different diameters (see Figure 3.7), and assume that in a short time interval the tract

doesn't change its shape, it is possible, starting from the speech signal, to find the parameters of this model by using linear prediction (see Rabiner and Schafer [1978]).

In practice, linear prediction, when applied to a speech signal, provides both a good spectral estimation of the input as well as the basis for a rudimental model–based coding.

Several low bit rate speech encoders rely on this model and implement a simplified model–based encoding in which parameters of a physical representation of the source model are estimated and transmitted to the decoder.

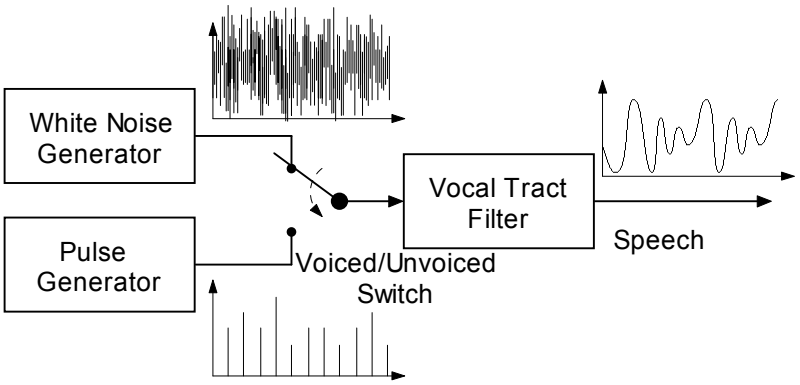


Figure 3.8: Hybrid codebook single–pulse excited voice synthesis.

LPC–10, which in the 1982 became a U.S. federal government standard (FS–1015) for low bit rate speech coding, is an example of a compression method that uses linear prediction. LPC–10 can compress speech down to 2400 bit/sec in real time by using only very modest hardware requirements. Low bit rate speech coding is important in a number of applications where it is necessary to preserve speaker identification even when highly noisy transmission channels are used.

To achieve its compression, LPC–10 assumes that, in a sufficiently short period of time (22.5 ms.), the speech signal can be viewed as a stationary waveform, therefore, the input speech is divided into segments having a duration of 22.5 ms. and linear prediction of 10–th order is used to characterize the spectral properties of the speech segment. Since LPC–10 uses a model similar to the one represented in Figure 3.8, three other characteristics of the speech segment must be estimated and sent to the decoder: pitch period, gain and voicing decision.

The 22.5 ms. segment is individually analyzed and converted into 54 bits of information that describe the coefficients of the LP filter (41 bits), the pitch and the voicing decision (7 bits), and the gain of the segment (5 bits). The remaining extra bit is used for synchronization.

LPC–10 assumes that the input is an analog signal, so before digitizing it, the input is filtered with a band–pass filter with cutoff frequencies of 100Hz and 3600Hz. Frequency components lower than 100Hz are removed because they are primarily 60Hz humming noise and little speech related information is found in that range. The upper cutoff frequency ensures that there is no aliasing during the A/D conversion. The A/D output goes to two separate places, a pre–emphasis filter and a pitch analysis buffer.

The pre–emphasis filter is a first order Finite Impulse Response (FIR) filter that has a frequency response: $H(z) = 1 - 0.9375 \cdot z^{-1}$. Pre–emphasis boosts high frequencies and lowers to 4 bit the precision necessary to achieve a stable reconstruction filter.

Gain and LP coefficients are calculated at the same time. The algorithm uses the covariance method to estimate LP parameters, so the resulting system of equations is solved with Cholesky decomposition. To extract pitch information, the signal from the

A/D is low–pass filtered with a 4th order Butterworth filter having a cutoff frequency of 800Hz. The output from this filter then goes to a second order inverse filter and then is fed into a voicing detector. The second order inverse filter helps the pitch extractor.

The voicing detector estimates if the current segment is voiced, unvoiced or a transition by counting the number of zero crossings and analyzing energy and amplitude of the signal. Unvoiced frames have more zero transitions than voiced frames because of high frequency components. Furthermore voiced frames have much higher amplitudes than unvoiced frames.

The pitch extractor is based upon an average magnitude difference function:

$$ADMF(\tau) = \sum_{n=1}^{130} \left| s_n - s_{n+\tau} \right|$$

In voiced frames AMDF has a clear global minimum at the pitch period of the frame; no such minimum is present in unvoiced speech frames. The period τ in the formula is changed to test 60 possible pitch values, ranging from 50 to 400 Hz.

Once the LP coefficients have been calculated and pitch, gain and voicing decisions have been identified, results are organized into a particular sequence that depends on the voicing type. Synthesis reconstructs the signal by reversing the process.

While in a LPC–10 codec, the general quality of the reconstructed signal is quite good (for such a low bit rate), frequently the voice sounds buzzy and unnatural and some jitter may be present. Voicing errors produce significant distortion and binary voicing decision is sometimes poor. Also, due to the model based coding, LPC–10 only models single–speaker speech, so background noise, multiple voices and music may produce unpredictable results.

3.2.4 Codebook Excited Linear Prediction (CELP)

In 1991 the American Department of Defense (DoD) standardized DoD CELP, a 4.8Kbit/s codec as Federal Standard 1016. DoD CELP is a hybrid codec that uses linear prediction and vector quantization in an analysis–by–synthesis loop. Voicing decision is implicit and handled with a pitch filter. A CELP encoder divides input speech into frames that have duration of 30 ms. For each frame a pitch period M is determined and the signal is passed through the pitch filter to remove this periodicity.

The residual signal is further divided into four sub–frames of 7.5 ms. each. For each sub–frame the encoder calculates a set of 8 filter coefficients for a short–term synthesis filter that models the vocal tract of the speaker (Figure 3.7). A gain is also computed to normalize the residual energy. CELP's most interesting feature is that the excitation for the cascade of filters is determined by using an analysis–by–synthesis loop that tries each vector in a codebook of 1024 pseudo random Gaussian vectors. These vectors are individually fed into the filters and their output is compared to the original signal by using a perceptive quality measure. The encoder then transmits to the decoder:

- Index of the best codeword;
- Gain;
- Vocal tract filter coefficients (every 7.5ms.);
- Pitch predictor coefficients (every 30 ms.).

The quality of the reconstructed speech is very good but real time encoding requires more than 300 MFLOPS, so to run in real time, a dedicated DSP is necessary. At the present, CELP is one of the most effective methods to obtain high quality speech at low bit rates.

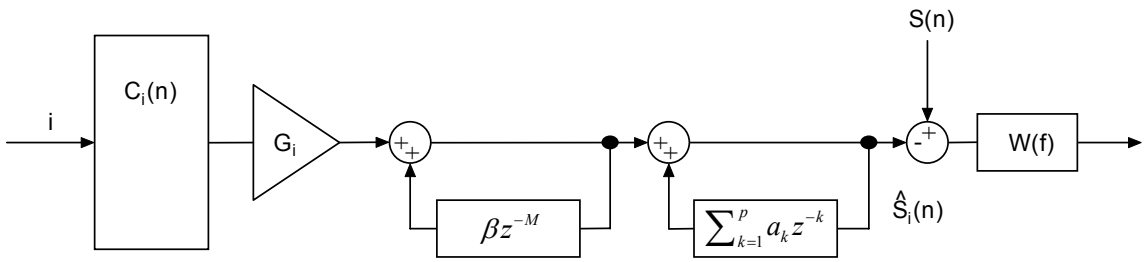


Figure 3.9: CELP decoder.

Many systems, like for example the cellular standard Global System for Mobile communications (GSM), use simplified versions of CELP. GSM is basically a CELP–like algorithm that encodes the residual directly, in order to avoid the expensive codebook search. With this simplification, a GSM codec provides good quality speech at a rate of 13 Kbit/s (3 times bigger than CELP) while running in real time on small processors. A good description of the GSM algorithm is given in Sayood [1996];

3.3 Image

3.3.1 JPEG

The Joint Photographic Expert Group (JPEG) developed and issued in 1990, a widely used standard for color image compression. The standard was mainly targeted for compressing natural gray level and color images. JPEG is a two–step transform–coding algorithm: the first step is lossy and involves a DCT transformation followed by quantization. This part is used to remove information that is perceptively irrelevant for a

human user. The second step involves lossless entropy encoding to eliminate statistical redundancies that could still be present in the transformed representation.

JPEG assumes a color image divided into color planes and compresses each color plane independently. Color planes are represented in terms of luminosity (or Y component) and two chrominance components C_b and C_r. As explained in precedence, the YC_bC_r color representation takes advantage of the fact that the human visual system is more sensitive to luminosity than to color changes. Following a color scheme named "4:2:2", each chrominance is sub sampled 2:1 both in the horizontal and vertical dimension and 2:1 compression is obtained even before applying JPEG by halving the resolution of the chrominance components.

Each color plane is further divided into blocks of 8x8 pixels. This size block was determined to be best compromise between the computational effort necessary to compute the Discrete Cosine Transform and the compression achieved (Pennebaker and Mitchell [1993]). Each block is transformed into the frequency domain by using the DCT; then the DCT coefficients are quantized with 64 different scalar quantizers, one for each frequency. Only the lowest frequency component (DC component) has a slightly different treatment from the other coefficients. The DC component of the previous block is subtracted and the difference is the value being quantized. Quantization steps are stored in a "quantization table" with 64 default values. Performance can be improved by computing the table on an image–by–image basis (see for example Daly [1992] and Ratnakar and Livny [1995, 1996]).

The default table quantizes low frequencies more accurately than high frequencies because, according to experiments made with the human visual system, the low frequency components are better perceived.

Quantization results in an 8x8 matrix of small integers, several of them being zeros. By using a zigzag pattern (see Figure 3.10), the matrix is scanned from the low to the high frequencies and it is converted to a one–dimensional vector. Run–length encoding is applied to compress the sequences of consecutive zeros. The result is further compressed by using a Huffman or an arithmetic coder. According to the standard, the bit stream can be organized in three different ways, each of them targeted to a specific application:

- **Sequential encoding** in which each image component is encoded in a single left–to–right, top–to–bottom scan;
- **Progressive encoding** in which the image is encoded in multiple scans and, during the decoding, the viewer may be able to see the image build up in multiple coarse–to–clear passes (as in many pictures downloaded from the Internet);
- **Hierarchical encoding** in which the image is encoded at multiple resolutions so that lower resolution versions may be accessed without first having to decompress the whole image.

The quality and the size of a JPEG–compressed image depends on the quantization that is JPEG's lossy part. Scaling appropriately the quantization tables, it is possible to control the size and the quality of the output. At very low bit rates (high compression) blocking artifacts become evident. For further reading see Wallace [1990, 1991], Pennebaker and Mitchell [1993]. The works by Bright and Mitchell [1999], Chang and Langdon [1991],

Ramchandran and Vetterli [1994], Vander Kam and Wong [1994] and Schaefer [2001] analyze some of the issues related to various aspects of JPEG encoding.

201	200	142	138	151	170	199	202
250	150	223	203	178	243	200	128
247	248	165	162	180	190	253	195
160	195	232	191	232	202	218	243
149	218	224	153	171	208	232	240
144	179	240	134	212	184	144	209
203	147	168	140	198	176	196	175
160	204	227	234	244	220	246	158

8x8 Image Block

529	-32	24	15	-57	-8	7	53
-3	-30	73	36	4	30	-1	-36
-35	34	-23	61	17	51	-16	-13
-89	-14	32	-35	4	-50	48	7
37	-51	-12	-53	-20	-21	-35	-14
-40	-19	49	1	43	-30	-26	15
6	5	24	-24	-10	-48	-39	-3
-11	-4	85	24	-17	10	-52	-19

DCT

33	2	2	0	2	0	0	0
0	2	5	1	0	0	0	0
2	2	-1	2	0	0	0	0
2	0	1	-1	0	0	0	0
6	2	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized values and ZigZag Scanning

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization Table

Figure 3.10: DCT, Quantization and zigzag scanning for a JPEG 8x8 image block.

3.3.2 JPEG–LS

JPEG was designed as a lossy image compression technique and the method was fine tuned for a specific quality. When compressing images at the highest quality setting, the size of the compressed image grows quickly and the results are not competitive even when compared to general–purpose lossless coding methods. To overcome this problem, the JPEG standardizing group decided to expand JPEG with a lossless mode that is not DCT–based (see Pennebaker and Mitchell [1993] for a general description as well as the more specific Langdon, Gulati, and Seiler [1992]). The lossless mode uses a predictive

scheme that linearly combines three causal neighbors to predict a pixel. The prediction error is then encoded by using a Huffman or arithmetic coder.

Lossless JPEG was not the result of a rigorous competitive evaluation as was the selection of the DCT–based methods and its performance turned out to be not very interesting. For this reason in 1994 the ISO/JPEG group issued a call for contribution for a new standard for lossless and near–lossless compression of continuous tone images from 2 to 16 bit per pixel. The standard, called JPEG–LS, combines various proposals, even though the algorithm is heavily based on an algorithm proposed by the Hewlett Packard Laboratories: LOCO–I (Weinberger, Seroussi and Sapiro [1996]).

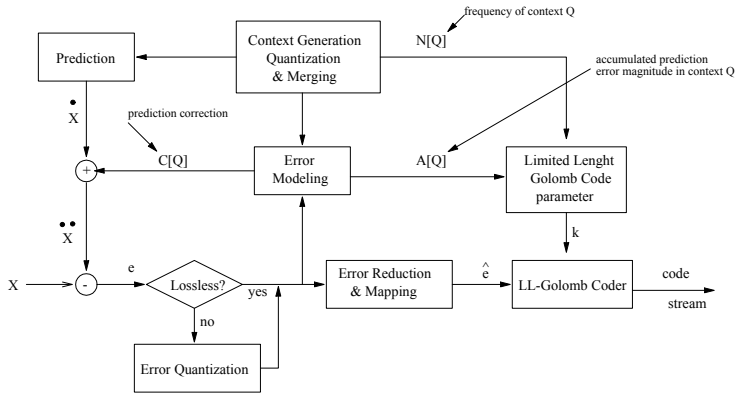


Figure 3.11: JPEG–LS encoder.

Figure 3.11 shows a schematic diagram of a JPEG–LS encoder. JPEG–LS is a predictive coder that uses a fixed predictor and an error–feedback technique to cancel prediction bias. The encoder stores for each class of prediction context the average of the past prediction errors. After the prediction, the encoder classifies the current context and subtracts to the prediction the past average errors in order to remove error bias. Prediction

residuals are first mapped into an exponentially decreasing distribution having positive values and then entropy coded with a Golomb code (Seroussi and Weinberger [1997]).

Besides a better prediction, the advantage of using error feedback is that errors in different context may have different probability distributions and this information can be used to normalize the error before entropy coding.

3.3.3 JBIG

The Joint Bi–level Image Experts Group (JBIG) defined in 1991 an innovative lossless compression algorithm for bi–level images. It consists of a predictive coder that uses a template of causal neighbor pixels to guess the value of the current pixel. The algorithm concatenates the value of the template pixels to identify the context of the pixel that is being predicted. The index of the context is used to choose the probability distribution that models the arithmetic coder.

The "A" pixel in Figure 3.12 is an adaptive pixel. Its position is allowed to change as the image is processed. The use of the adaptive pixel improves compression by identifying repeated occurrences of the same block of information.

JBIG is also able to operate in progressive mode: an approximate version of the original image is first transmitted and then improved as compression proceeds. This is achieved by subdividing the image in layers, each of which is a representation of the original image at a different resolution. The lowest resolution layer is called starting layer and is the first to be coded. Then the other layers (differential layers) are encoded by using information from the previous encoded layer. If the resolution reduction algorithm

used by the progressive mode suits well the input image, this mode of operation is very effective. In fact most of the pixels in the differential layers may be predicted with little or no error and then very little information has to be encoded.

Progressive and sequential modes are completely compatible. This compatibility is achieved by dividing the original image into horizontal stripes. Each stripe is coded separately and then transmitted. An image that has been coded in the progressive mode may be decoded in sequential mode by decoding each stripe sequentially, to its full resolution, starting from the one on the top of the image. Progressive decoding may be obtained by decoding one layer at a time.

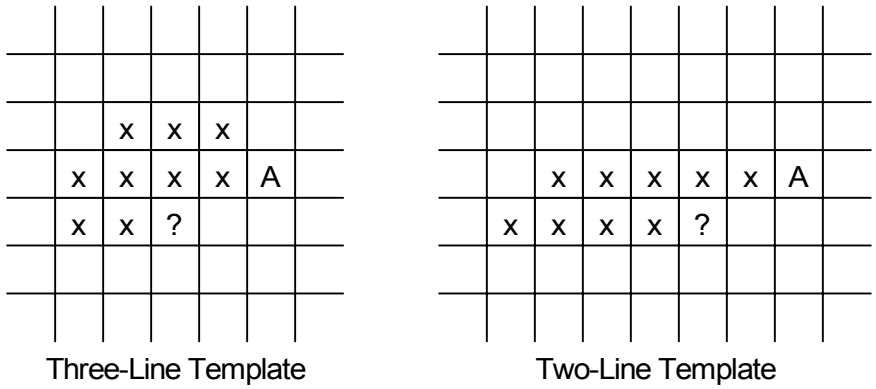


Figure 3.12: JBIG prediction templates.

JBIG may also be successfully used when coding images with more than one bit–per–pixel (grayscale or even color images). The image is decomposed into bit–planes (i.e. if the image is 4–bpp, each pixel is represented by the binary string $b_3b_2b_1b_0$ the i–th bit–plane stores only the bit b_i of each pixel) and each plane is coded separately as a bi–level image. In principle JBIG is also able to work with 255–bpp images, but in practice the algorithm shows interesting performances for images with at most 8 bpp.

For further reading, see Arps and Truong [1994] and Thompkins and Kossentini [1999] for a description of JBIG2. Issues related to the design of the symbol dictionary are addressed in Ye, Schilling, Cosman, and Ko [2000].

3.3.4 JPEG–2000

JPEG–2000 is a newly issued standard for digital image compression (Marcellin, Gormish, Bilgin and Boliek [2000]). While having a rate distortion advantage over JPEG it also provides a number of features that are highly desirable in current applications. For example, JPEG–2000 allows extraction of images at different resolutions, regions of interest and single components, all without requiring full decompression of the bitstream. This feature is essential for editing purposes or when a compressed image must be transmitted to a device that is not capable of full resolution.

The organization of the bitstream supports random spatial access to regions on the image and allows a number of compressed domain processing like pan, zoom, rotation and cropping.

The JPEG–2000 standardization started from Ricoh's submission of the algorithm CREW to the JPEG–LS standardization. Although LOCO–I (Weinberger, Seroussi and Sapiro [1996]) was selected as the basis for JPEG–LS, CREW had such a rich and interesting set of features that it motivated a new standardization effort. One of the most relevant features is the use of an integer wavelet decomposition to transform the image.

3.3.5 GIF

The Graphics Interchange Format was developed by CompuServe Information Services in 1987 (GIF 87a) as an efficient, compressed graphics file format, which allows for images to be sent between different computers. The most recent version is called GIF 89a. More than being a data compression method, GIF is a graphics file format that uses a variant of LZW to compress the data.

Graphics data are compressed using a dynamic growing dictionary that starts with the number of bit per pixel b, as a parameter. Bilevel images use $b = 2$ and images with 256 colors or gray levels use $b = 8$. Intermediate values for b are also allowed.

The initial dictionary contains $2^{(b+1)}$ entries and it doubles its size each time it fills up, until it reaches a size of $2^{12} = 4096$ entries, where it remains static. When the dictionary is full the encoder monitors the compression ratio and when it drops below a threshold, it discards the current dictionary and starts with a fresh, new one. This event is signaled to the decoder by transmitting a special symbol whose value is $ESC = 2^b$.

Because of its lossless nature GIF performs well on synthetic or palletized images, i.e. images with a small number of colors and not corrupted by noise. The compression of natural images, in which the number of colors may be potentially big and sampling noise is present, can be addressed by using a preprocessing that builds a color map and artificially reduces the number of colors.

3.4 Video

3.4.1 H.261

In the 1993 the ITU–T (International Telecommunication Union / Telecommunication Standardization Sector) developed H.261, a video coding standard for audiovisual services at bit rates multiple of 64 Kbit/s. This bit rate was chosen because of the availability of ISDN (Integrated Services Digital Network) transmission lines that could be allocated in multiples of 64Kbit/s. As a consequence of that choice, in older drafts, H.261 is also referred as $p * 64$Kbit/s.

H.261 is a coding standard targeted to videoconference and videophone applications operating at bit rates between 64 Kbit/s and 2 Mbit/s (that is $p * 64$ Kbit/s with $1 \leq p \leq 30$). The coding algorithm combines interframe prediction, transform coding and motion compensation. There are two operational modes, named interframe and intraframe. Intraframe coding is a JPEG–like encoding where the input signal is directly decomposed by the DCT. To improve error resilience, unlike in JPEG, the DC coefficient is not differentially encoded. Intraframe mode is used when the interframe prediction is not efficient, i.e. when a scene changes or when there is too much motion in the scene or, periodically, to improve error resilience. Interframe coding is based on motion compensated prediction that is followed by a DCT coding of the prediction error. Motion vectors exploit temporal redundancy and help the codec to compensate for objects moving in the scene. To remove any further redundancy, DCT coefficients and motion vectors are variable length encoded with a static Huffman code.

H.261 supports two picture resolutions, QCIF (Quarter Common Interchange Format) and CIF (Common Interchange Format). This permits a single recommendation to cover use in and between regions that use 625 and 525 line television standards.

In a H.261 encoder (see Figure 3.13), every input frame is de–interlaced, converted from NTSC or PAL to CIF format, eventually noise filtered, preprocessed and stored into the frame memory. De–interlaced frames are referred to as pictures.

Pictures are divided into blocks of 8×8 pixels. Four Y blocks, a C_b block and a C_r block are grouped into a coding unit called macroblock. Macroblocks can be intra frame, inter frame coded or eventually skipped with criteria that have not been defined by the standard and may vary dynamically depending on the complexity of the input signal and the output data rate constraints.

After the DCT coefficients are quantized and encoded with a Huffman coder, a BCH(511,493) error correction code (McEliece [1977]) is used to protect the bit stream and avoid error propagation in predictive coding. Finally the encoder inverts the DCT and decodes and stores the current frame so that it can be used to perform the next prediction. Since the prediction is based on the quantized output, this “closed–loop” coding keeps track of the coding errors, monitors the quality of the transmitted image and keeps encoder and decoder synchronized.

Output data rate is controlled dynamically by adjusting the quantization steps while monitoring the capacity of an output buffer.

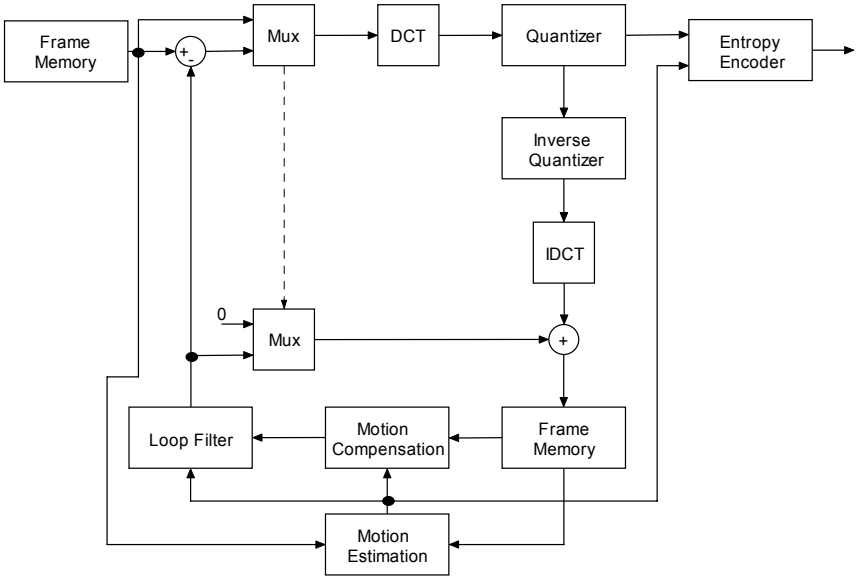


Figure 3.13: H.261 video encoder.

3.4.2 MPEG–1

In 1988 the Moving Picture Experts Group (MPEG) was founded under ISO/SC2 to standardize a coding algorithm targeted for digital storage of video at bit rates around 1.5 Mbit/s. A draft for the first MPEG–1 (formally known also as H.262) appeared in 1991 and a final version was issued in 1992. The MPEG–1 video algorithm was developed with respect to the JPEG and H.261 standards and shares several common features with the H.261, so that implementations supporting both standards are feasible.

The main difference between H.261 and MPEG–1 is in the fact that MPEG–1 was primarily targeted for multimedia CD–ROM applications, so it required additional functionality supported by both encoder and decoder. Among these additional features there are frame based random access of video, fast forward/fast reverse (FF/FR) searches through compressed bit streams, reverse playback of video and the possibility to edit the compressed bit stream.

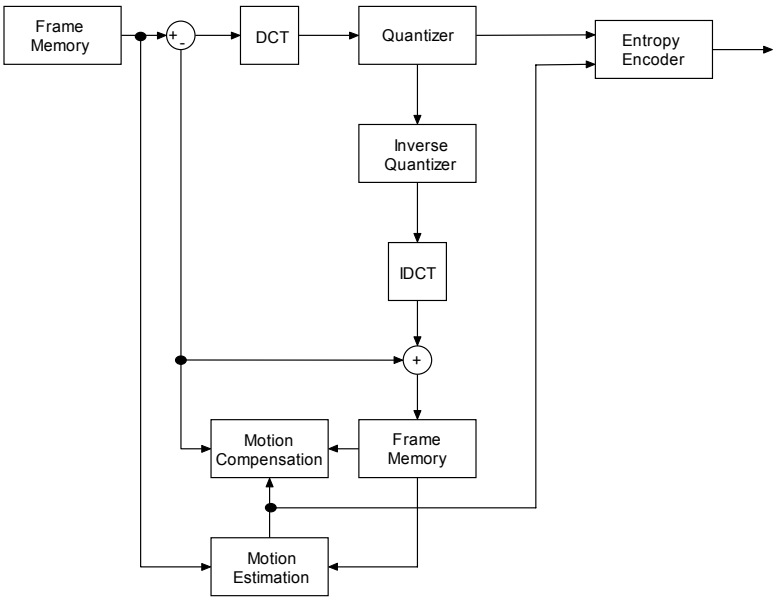


Figure 3.14: MPEG–1 video encoder.

Its parameters were optimized for digital storage media, but the algorithm is intended to be generic. Standardized encoding algorithms are "decoder standards"; this means that the standard does not define the details of the encoding process but only the syntax of the bit stream and, possibly, a suggested decoding process. Standards are also independent of a particular application and therefore they mainly provide a "toolbox". It's up to the user to decide which tools to select to suit a specific applications.

Figure 3.14 outlines the functions that are typically executed by an MPEG–1 encoder that uses a hybrid DCT/Motion Compensated DPCM scheme very similar to the H.261 standard. Encoding may include preprocessing that performs format and color conversion, de–interlacing, pre–filtering and subsampling; none of these operations is specified by the standard.

After the preprocessing, a format for the pictures is selected and the pictures are encoded in the following modes:

- **Intra frame** coded (**I–pictures**): require no motion compensation, each macroblock is DCT coded, the coefficients are linearized with a zig–zag scanning and finally quantized. The encoding method for I–pictures, very close to the JPEG standard, leads to a modest compression but since every macroblock in the picture is independently coded and transmitted, this mode provides fast random access, a functionality required in digital storage media.

- **Inter frame** coded or (**P–pictures**): are those coded by using motion compensated prediction from a previous intra or inter coded picture. Macroblocks in a P–picture can be motion compensated, intra coded or eventually skipped with a functionality called "conditional replenishment". Motion compensation and conditional replenishment typically allow compression rates up to 3 times higher than I–pictures. P–pictures can also be used as references for the motion–compensated prediction.

- **Bi–directionally predicted** or (**B–pictures**): provide the highest degree of compression, typically 10 times higher than the I–pictures. B–pictures are coded using motion–compensated prediction from past and/or future I–pictures and P–pictures. Since B–pictures are not used in the prediction, they can accommodate more distortion, but because B–pictures depend on the future frames, their use may introduce a substantial coding delay.

- **DC coded** or (**D–pictures**): are DCT coded pictures in which only the DC coefficient is quantized and transmitted. They were introduced to allow a fast preview of the video sequence without a full decoding of the bit stream.

Even in MPEG–1 the output bit stream is Huffman coded. Because the size of compressed video is inherently variable in nature, MPEG–1 uses an output buffer that stores the variable bit stream generated in the encoder and provides the possibility of transmitting the video stream at a constant bit rate.

A mechanism not specified by the standard monitors the output buffer and adjusts the bit rate by adapting the quantizer steps. Coarse quantization of the DCT–coefficients enables the storage or transmission of video with high compression ratios but, depending on the level of quantization, it may result in significant coding artifacts. The efficiency of the rate control algorithms heavily affects the visual quality of the video reconstructed at the decoder.

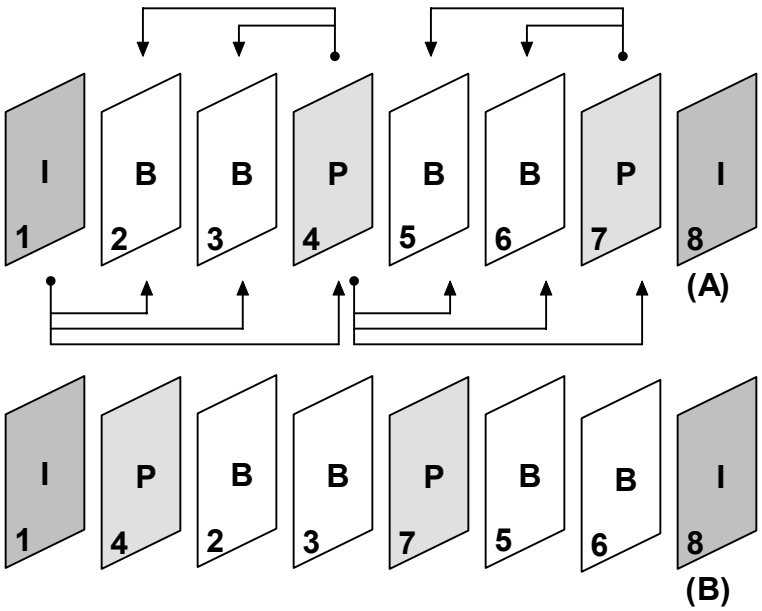


Figure 3.15: Time dependency among I, P and B pictures.

Figure 3.15 shows temporal dependences between I, P and B–pictures. The standard does not suggest a specific interleaving between I, P and B pictures and many choices are possible. Interleaving changes compression rate, quality and coding delay and the end user is free to arrange the picture types in a video sequence with a high degree of flexibility to suit diverse applications requirements.

A video sequence coded using I–pictures only (I, I, I, I, I, I, ...) allows the highest degree of random access, FF/FR, editability and the lowest coding delay, but only achieves a modest compression. This mode is sometimes referred to as M–JPEG (Motion JPEG) because it substantially consists of a sequence of JPEG pictures. Regular I–picture update and no B–pictures (i.e. I, P, P, P, P, P, P, I, P, P, P, P, ...) achieves moderate compression and a certain degree of random access while retaining FF/FR functionality. Incorporation of all three pictures types (like in I, B, B, P, B, B, P, B, B, I, B, B, P, ...), achieves high compression and reasonable random access with some FF/FR functionality but also increases significantly the coding delay that may not be tolerable in applications like video telephony or video conferencing.

3.4.3 H.263

This Recommendation, issued in the 1996, specifies a coded representation that can be used for compressing moving pictures at low bit rates. Even though H.263 was designed for data rates lower than 64 Kbit/s, this limitation has been removed in the final draft. The basic configuration of the coding algorithm is based on ITU–T Recommendation H.261 enriched by changes and optional encoding modes that improve performance and error recovery.

Like other video coding standards, H.263 is based on motion compensated inter–picture prediction that exploits temporal redundancy and transform coding of the prediction error that reduces spatial redundancy. The transmitted symbols are variable length coded.

The main differences between the H.261 and H.263 coding algorithms are listed below:

- Half–pixel precision is used for motion compensation whereas H.261 only uses full–pixel precision and a loop filter. Half–pixel MC is computed by interpolating the image.
- Some parts of the hierarchical structure of the data stream are now optional, so the codec can be configured for a lower data rate or to achieve better error protection.
- In addition to the basic video source coding algorithm, a number of negotiable coding options are included for improved performance:
    - Unrestricted Motion Vectors. In this mode motion vectors are allowed to point outside the picture. Edge pixels are used for the prediction of the missing pixels. This mode achieves a significant gain when there is movement along the edge of the pictures, especially for small picture formats. Additionally, this mode includes an extension of the motion vector range so that larger motion vectors can be used. This mode is especially useful to take into account camera movement (panning).
    - Syntax–based Arithmetic Coding. In this mode arithmetic coding is used instead of Huffman VLC coding. SNR and reconstructed frames will be

the same, but generally 5–10% fewer bits are produced. This gain depends of course on the sequence, the bit rate and other options used.

- Advanced prediction. This option means that overlapped block motion compensation is used for the P–frames. Four 8×8 vectors instead of one 16×16 vector are used for some macroblocks in the picture, and motion vectors are allowed to point outside the picture as in the UMV mode described above. The encoder has to decide which type of vectors to use. Four vectors use more bits, but give better prediction. The use of this mode generally gives considerable subjective improvement because overlapped motion compensation results in less blocking artifacts.

- Forward and backward frame prediction (or P–B frames mode). A PB–frame consists of two pictures being coded as one unit. The name PB comes from the name of picture types in MPEG where there are P–pictures and B–pictures. A PB–frame consists of a P–picture that is predicted from the last decoded P–picture and a B–picture that is predicted from both the last decoded P–picture and the P–picture currently being decoded. For relatively simple sequences, this mode nearly doubles the frame rate without a substantial increase in the bit rate. For sequences with a lot of motion, PB–frames do not work as well as B–pictures in MPEG. This happens because there are no separate bi–directional vectors in H.263; the forward vectors for the P–picture are scaled and added to a small delta–vector. The advantage over MPEG is a low overhead for the

B–picture part. This feature is useful in the low bit rate encoding of the relatively simple sequences often generated by videophones.

These options can be used together or separately and are negotiable in the sense that the decoder signals the encoder whether any of these options are available. If the encoder supports them, it may decide to use one or more options to improve the quality of the video sequence.

H.263 supports five resolutions: QCIF and CIF, already supported by H.261, and SQCIF, 4CIF, and 16CIF. The support of 4CIF and 16CIF allows the codec to compete with video coding standards, such as the MPEGs, targeted for a higher bit rate.

In a subsequent standardization attempt, H.263+ (H.263 Version2) was introduced as a powerful enhancement of the existing H.263 standard. The improvements add the following coding options to the basic algorithm:

- Reversible Variable Length Coding Mode (or RVLC): When this mode is used, variable length coding is achieved by using special tables that contain reversible codewords. Since RVLC allow decoding from both directions and provide excellent error detection, if a packet is corrupted by an error, decoding may be tried starting from the end of the packet and more uncorrupted data can be extracted.

- Advanced Intra Coding Mode (AIC): allows prediction of the first row or the first column of the coefficients of a transformed I block from blocks that have already been encoded. Scanning can be performed in the usual zigzag or in an alternate horizontal or vertical mode.

- Deblocking Filter Mode (DF): introduces a deblocking filter in the coding loop. This improves prediction and reduces blocking artifacts.
- Slice Structured Mode (SS): A slice structure is employed to group and transmit macroblocks. The slice can have an arbitrary shape and contain a variable number of macroblocks. Macroblocks can be transmitted in the usual sequential mode or by using an arbitrary order.
- Supplemental Enhancing Information Mode (SEI): includes additional information in the bitstream that can be used to support extended functionalities like picture freeze, chroma keying, progressive refinement and video segmentation.
- Improved PB Mode (IPB): Improves the PB mode by allowing backward, forward and bi directional prediction in PB frames.
- Reference Picture Selection mode (RPS): Removes the limitation present in most coders that the reference picture must be the previous one. This feature limits error propagation because it allows the specification of a different reference picture when the current reference has been damaged during the transmission.
- Temporal and SNR Scalable Modes: specify a syntax that allows the transmission over a prioritized channel. The bitstream is divided into a base layer and into one or more enhancement layers that improve temporal or spatial resolution or refine the picture quality. Decoders not capable of full rendering may decode only part of the bitstream.

- Reference Picture Resampling (RPR): specifies a warping function that must be applied to the reference picture before being used for prediction.
- Reduced Resolution Update Mode (RRU): allows the encoder to send update information for a picture encoded at lower resolution while maintaining full resolution for the reference frame.
- Independently Segmented Coding mode (ISD): Treats every segment as an independent coding unit. No data dependency across segment boundaries is allowed. It is helpful to prevent error propagation and improves error resiliency and error recovery.
- Alternative Inter VLC Mode (AIV): specifies an alternative VLC table that is optimized for small quantization steps in Intra block coding.
- Modified Quantization Mode (MQ): Improves the range of the quantization steps that can be represented and allows representation of quantized coefficients outside the range [–127,127]. These values are clipped to the boundaries in the basic H.263.

These new coding models improve the performance and the flexibility of this standard so that H.263+ is regarded today as state of the art low bit rate video coding. H.263+ also constitutes the very low bit rate core of MPEG–4. Details on H.263+ can be found in the standard draft or in Erol, Gallant, Cote and Kossentini [1998].

3.4.4 MPEG–2

Studies on MPEG–2 started in 1990 with the initial target to issue a coding standard for TV–pictures with CCIR Rec. 601 resolution at data rates below 10 Kbit/s. In 1992 the scope of MPEG–2 was enlarged to suit coding of HDTV thus making an initially planned MPEG–3 phase redundant.

Basically MPEG–2 can be seen as a superset of the MPEG–1 and since it was designed to be backward compatible to MPEG–1. Every MPEG–2 compatible decoder must be able to decode a valid MPEG–1 bit stream. Several video coding algorithms were integrated into a single syntax to meet the diverse applications requirements. New coding features were added by MPEG–2 in order to achieve sufficient functionality and quality. Specific prediction modes were developed to support efficient coding of interlaced video. In addition scalable video coding extensions were introduced to provide additional functionality, such as embedded coding of digital TV and HDTV as well as graceful quality degradation in the presence of transmission errors.

A hierarchy of "*Profiles*", describing functionalities, and "*Levels*", describing resolutions, was introduced in order to allow low–complexity implementation in products that do not require the wide range of video input formats supported by the standard (e.g. SIF to HDTV resolutions).

As a general rule, each profile (see Table 3.2) defines a new set of algorithms added to the algorithms in the profile below and a level (see Table 3.1) specifies the range of the parameters that are supported by the implementation (image size, frame rate and bit rates). The MPEG–2 core algorithm at MAIN profile features non–scalable coding of both progressive and interlaced video sources.

Level	Parameters
High	1920 samples/line 1152 lines/frame 60 frames/s 80 Mbit/s
High 1440	1440 samples/line 1152 lines/frame 60 frames/s 60 Mbit/s
Main	720 samples/line 576 lines/frame 30 frames/s 15 Mbit/s
Low	352 samples/line 288 lines/frame 30 frames/s 4 Mbit/s

Table 3.1: Upper Bound of Parameters at Each Level.

Five profiles and four levels create a grid of 20 possible combinations. The variations are so wide that it is not practical to build a universal encoder or decoder. So far only the 11 combinations showed in Table 3.3 have been implemented. Interest is generally focused on the Main profile, Main level, sometime written as "MP@ML", which covers broadcast television formats up to 720 pixels x 576 lines at 30 frames/sec and with 4:2:0 subsampling.

MPEG–2 as defined in the MAIN Profile, is a straightforward extension of MPEG–1 that accommodates coding of interlaced video. As well as MPEG–1, MPEG–2 coding is based on the general hybrid DCT/DPCM coding scheme previously described, incorporating macroblock based motion compensation and coding modes for conditional replenishment of skipped macroblocks. The concept of I–pictures, P–pictures and B–pictures is fully retained in MPEG–2 to achieve efficient motion prediction and to assist

random access functionality. The algorithm defined in the MPEG–2 SIMPLE Profile is targeted to transmission systems and it is basically identical to the MAIN Profile, except that no B–pictures are allowed. This keeps the coding delay low and simplifies the decoder that does not need any additional memory to store the past frames.

Profile	Functionalities
High	Supports all functionality provided by the Spatial Scalable Profile plus the provision to support: o 4:2:2 YUV–representation for improved quality
Spatial Scalable	Supports all functionality provided by the SNR Scalable Profile plus algorithms for: o Spatial scalable coding (2 layers allowed); o 4:0:0 YUV–representation.
SNR Scalable	Supports all functionality provided by the Main profile plus algorithms for: o SNR scalable coding (2 layers allowed); o 4:2:0 YUV–representation.
Main	Non–scalable coding algorithm supporting functionality for: o Coding of interlaced video; o Random access; o B–picture prediction modes; o 4:2:0 YUV–representation.
Simple	Includes all functionality provided by the Main profile but does not support: o B–picture prediction; o 4:2:0 YUV–representation.

Table 3.2: Algorithms and Functionalities Supported With Each Profile.

	Low	Main	High 1440	High
Simple		X		
Main	X	X	X	X
SNR Scalable	X	X		
Spatial Scalable			X	
High		X	X	X

Table 3.3: Combinations Implemented.

MPEG–2 also introduces the concept of Field and Frame Pictures to accommodate coding of progressive and interlaced video via a frame and a field prediction mode. When

the field prediction mode is used, two fields of a frame are coded separately and the DCT is applied to each macroblock on a field basis. Alternatively, lines of top and bottom fields are interlaced to form a frame that is encoded in the frame prediction mode as in MPEG–1. Field and frame pictures can be freely mixed into a single video sequence.

Analogously, a distinction between motion compensated field and frame prediction mode was introduced in MPEG–2 to efficiently encode field pictures and frame pictures. Inter–field prediction from the decoded field in the same picture is preferred if no motion occurs between fields. In a field picture all predictions are field predictions. Also, a new motion compensation mode based on 16x8 blocks was introduced to efficiently explore temporal redundancies between fields. MPEG–2 has specified additional YC_bC_r chrominance subsampling formats to support applications that require the highest video quality. Next to the 4:2:0 format already supported by MPEG–1, the specification of MPEG–2 is extended to 4:2:2 format defining a "Studio Profile", written as "$422P@ML$", suitable for studio video coding.

Scalable coding was introduced to provide interoperability between different services and to flexibly support receivers with different display capabilities. Scalable coding allows subsets of the layered bit stream to be decoded independently to display video at lower spatial or temporal resolution or with lower quality. MPEG–2 standardized three scalable coding schemes each of them targeted to assist applications with particular requirements:

- **Spatial Scalability**: supports displays with different spatial resolutions at the receiver; a lower spatial resolution video can be reconstructed from the base layer. Multiple resolution support is of particular interest for compatibility between

Standard (SDTV) and High Definition Television (HDTV), in which it is highly desirable to have a HDTV bitstream that is backward compatible with SDTV. Other important applications for scalable coding include video database browsing and multi–resolution playback of video in multimedia environments where receivers are either not capable or not willing to reconstruct the full resolution video. The algorithm is based on a pyramidal approach for progressive image coding.

- **SNR Scalability**: is a tool developed to provide graceful degradation of the video quality in prioritized transmission media. If the base layer can be protected from transmission errors, a version of the video with gracefully reduced quality can be obtained by decoding the base layer signal only. The algorithm used to achieve graceful degradation is based on a frequency (DCT–domain) scalability technique. At the base layer the DCT coefficients are coarsely quantized and transmitted to achieve moderate image quality at reduced bit rate. The enhancement layer encodes and transmits the difference between the non–quantized DCT–coefficients and the quantized coefficients from the base layer with a refined quantization step size. At the decoder the highest quality video signal is reconstructed by decoding both the lower and the higher layer bitstreams. It is also possible to use this tool to obtain video with lower spatial resolution at the receiver. If the decoder selects the lowest $N*N$ DCT coefficients from the base layer bit stream, a non–standard inverse DCT of size $N*N$ can be used to reconstruct the video at a reduced spatial resolution.

- **Temporal Scalability**: it was developed with an aim similar to spatial scalability. This tool also supports stereoscopic video with a layered bit stream suitable for receivers that have stereoscopic display capabilities. Layering is achieved by providing a prediction of one of the images of the stereoscopic video (the left view, in general) in the enhancement layer. The prediction is based on coded images from the opposite view that is transmitted in the base layer.

Scalability tools can be combined together into a single hybrid codec.

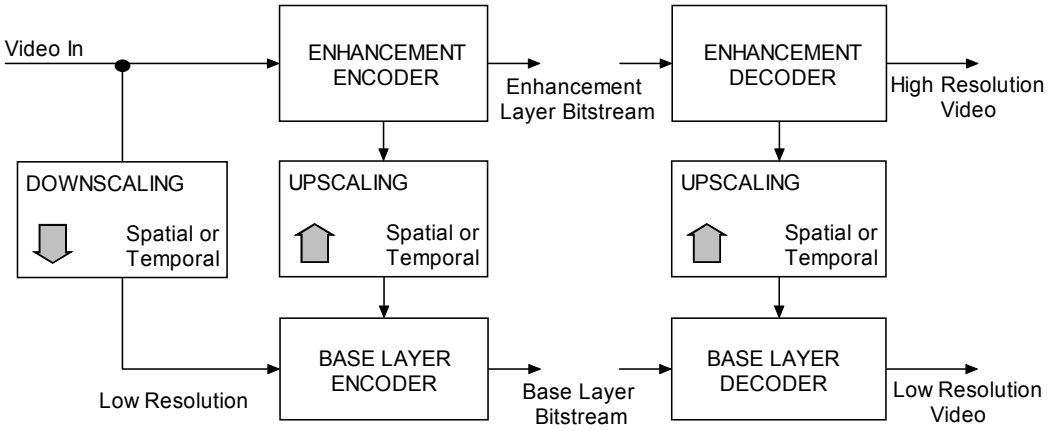


Figure 3.16: Scalable Coding

Figure 3.16 depicts a multiscale video coding scheme where two layers are provided, each layer supporting video at a different resolution. This representation can be achieved by downscaling the input video signal into a lower resolution video (down sampling spatially or temporally). The downscaled version is encoded into a base layer bit stream with reduced bit rate. The up scaled reconstructed base layer video (up sampled spatially or temporally) is used as a prediction for the coding of the original input video signal.

Prediction error is encoded into an enhancement layer bit stream. A downscaled video signal can be reconstructed by decoding the base layer bit stream only.

3.4.5 MPEG–4

MPEG group officially initiated the MPEG–4 standardization phase in 1994 with the mandate to standardize algorithms and tools for coding and flexible representation of audio–visual data for Multimedia applications.

Bit rates targeted for the MPEG–4 video standard range between 5–64 Kbit/s for mobile or PSTN (Public Switched Telephone Network) video applications and up to 2 Mbit/s for TV/film applications so that this new standard will supersede MPEG–1 and MPEG–2 for most applications.

Seven new video coding functionalities have been defined which support the MPEG–4 focus and which provide the main requirements for the work in the MPEG video group.

In particular MPEG–4 addresses the need for:

- Universal accessibility and robustness in error prone environments;
- High interactive functionality;
- Coding of natural and synthetic data;
- Compression efficiency;

One of the most innovative features consists in the definition of Video Object Planes (VOPs) that are units coded independently and possibly with different algorithms.

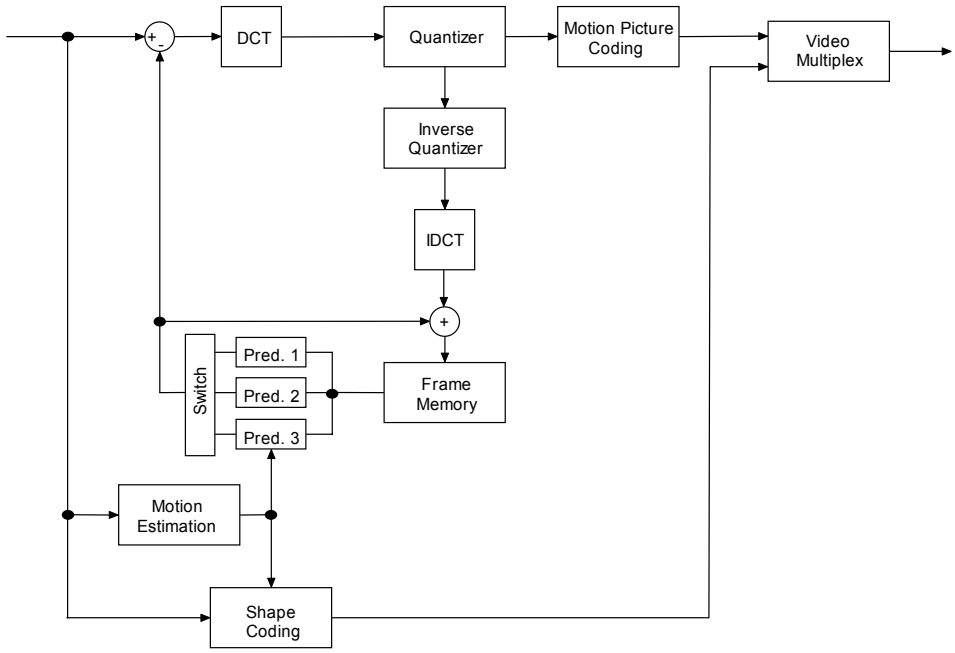


Figure 3.17: MPEG–4.

Figure 3.17 shows a scheme of an MPEG–4 system where the possibility of encoding separate objects and multiplex the result in a single bitstream is made evident.

The decoder reconstructs the objects by using for each of them the proper decoding algorithm and a composer assembles the final scene. A scene is composed by one or more *Video Object Planes* with an arbitrary shape, also transmitted to the decoder. VOPs can be individually manipulated, edited or replaced. VOPs derive from separate objects that have to be composed in a single scene or determined by a segmentation algorithm. To improve compression ratio, bitstream can also refer to a library of video objects available both at the encoder and decoder sides. Another interesting feature is the possibility of using the Sprite Coding Technology, in which an object moving on a relatively still background is encoded as a separate VOP. A good introduction to MPEG–4 features can be found in Sikora [1997].

TRELLIS CODED VECTOR RESIDUAL QUANTIZATION

4.1 Background

Vector Quantization (or in short VQ) is one of the oldest and most general source coding techniques. Shannon [1948, 1959] proved in his "Source Coding Theorem" that VQ has the property of achieving asymptotically the best theoretical performance on every data source.

Vector quantization can be seen as a generalization of scalar quantization to a multi dimensional input. A vector quantizer is often defined as a set of two functions:

- An *encoding* function $E:\mathbb{R}^n \to \mathbb{N}$ that maps n–dimensional vectors from the Euclidean space $\mathbb{R}^n$ to integer *indices*;
- A *decoding* function $D:\mathbb{N} \to \mathbb{R}^n$ that maps every index to one of a set of representative n–dimensional vectors that we will call *reconstruction levels* or *centroids*.

By means of these two functions, an n–dimensional vector can be approximated by one of a small set of vectors carefully selected in order to minimize the average distortion

introduced in the approximation. A quantizer achieves lossy compression by mapping multiple inputs into the same index, so the mapping is intrinsically non–reversible.

Before discussing the peculiarity of vector quantization, it is helpful to introduce some background concepts by making reference to the simpler *Scalar Quantizer* (or SQ). A scalar quantizer can be formally defined in the following manner:

Definition 4.1: Let x be a random point on the real line $\mathbb{R}$; an N–level Scalar Quantizer (or SQ) of $\mathbb{R}$ is a triple $Q = (A, \mathrm{Q}, P)$ where:

1. $A = \{y_1, y_2, ..., y_N\}$ is a finite indexed subset of $\mathbb{R}$ called *codebook*;

2. $P = \{S_1, S_2, ..., S_N\}$ is a partition of $\mathbb{R}$. Equivalence classes (or *cells*) S_j of P satisfy:

$$\bigcup_{j=1}^{N} S_j = \mathbb{R},$$

$$S_j \cap S_k = \varnothing \text{ for } j \neq k;$$

3. $\mathrm{Q} : \Re \mapsto A$ is a mapping that defines the relationship between the codebook and partitions such that:

$$\mathrm{Q}(x) = y_j \text{ if and only if } x \in S_j.$$

The encoder function $E(x)$ associates to x the integer index i such that $Q(x) = y_i$ and the decoder $D(i)$ associates the integer i to the i–th centroid y_i. Quantization is carried out by composing the two functions E and D as:

$$D(E(x)) = y_i = \hat{x}$$

and $\hat{x}$ is said to be the quantized representation of x.

When measuring the distortion introduced by a scalar quantizer, the squared quantization error is frequently assumed to be a good distortion metric both for its relevance and for its mathematical tractability:

$$d\left(x,\hat{x}\right)=\left(x-\hat{x}\right)^2.$$

With this choice, the total distortion expressed in term of mean squared error equals to:

$$D_{MSE}=\sum_{j=1}^{N}\int_{x_{j-1}}^{x_j}\left(x-y_j\right)^2 f(x)d(x)$$

where $f(x)$ is the probability density function of the input X and the partition S_j of an N–level scalar quantizer that has codebook $A=\{y_1,y_2,...,y_N\}$, codeword y_j and boundaries x_{j-1} and x_j.

The average distortion D_{MSE} that the quantizer achieves on a given input distribution depends on the partition boundaries and on the codebook entries. Figure 4.1 compares a uniform and a non–uniform quantizer; in the former, the real line is partitioned in equally–sized intervals. In the figure, the abscissa represents a point on the real line $\mathbb{R}$ and the ordinate shows the reconstruction levels. Which quantizer best fits a given input source depends both on the input statistics and on the distortion measure being minimized.

If we focus on the minimization of D_{MSE}, it is possible, given the number of levels and the input distribution, to derive the necessary conditions for the optimality of a non–uniform scalar quantizer. A quantizer that satisfies both conditions is called a Lloyd–Max quantizer since this type of quantizer was first derived by Lloyd [1957] in an unpublished paper and later by Max [1960].

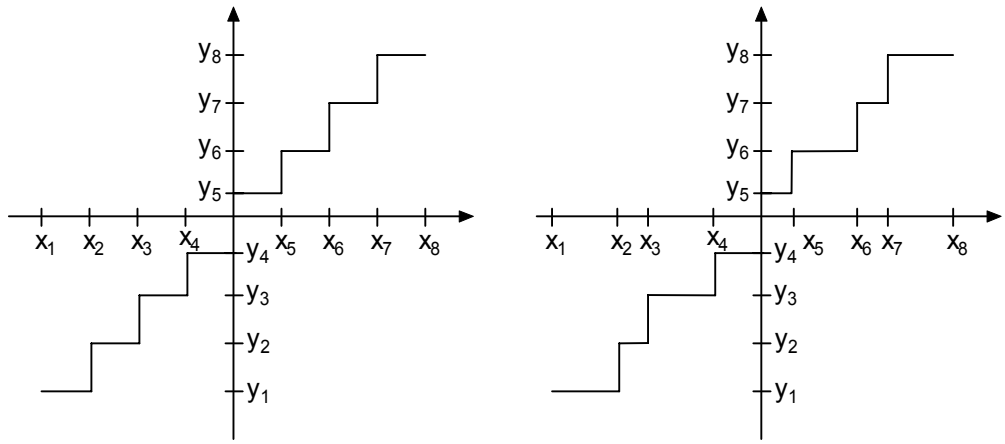


Figure 4.1: Uniform vs. Non–Uniform scalar quantizer.

The partitions of an N–level scalar quantizer that minimizes the mean squared error must have boundaries that satisfy:

$$x_j = \frac{y_j + y_{j+1}}{2} \quad \text{where } 1 \le j \le N-1,$$

$$x_0 = -\infty, \quad x_N = \infty.$$

And its centroids necessarily satisfy:

$$y_j = \frac{\int_{x_{j-1}}^{x_j} x\, f(x) d(x)}{\int_{x_{j-1}}^{x_j} f(x) d(x)} \quad \text{where } 1 \le j \le N.$$

Since the scalar quantizer encodes input symbols one by one, this method is unable to exploit inter–symbol dependence and, unless the input source is memoryless, the achievable compression is relatively poor. To take advantage of existing inter symbol correlation, it is possible to group a number of input symbols together and treat this block

(or *vector*) as a single coding unit. This is the approach that is taken by a vector quantizer.

A vector quantizer works as a scalar one but it groups and encodes vectors instead of scalars. While the dimension of the vector grows, a VQ is able to capture more and more inter symbol dependence and this results in a coding that is theoretically optimal in the sense that, fixing the distortion, it achieves the lowest possible rate.

Shannon's theorem proves the existence of such a quantizer with asymptotically optimal performance; unfortunately the proof is probabilistic in nature and, while demonstrating the existence of the quantizer, doesn't suggest any method to construct it. Further investigation (Lin [1992]) showed that, given the input distribution and the distortion measure being minimized, the design of an optimal codebook is an NP–complete problem.

At the present, the best practical solution for the design of unstructured vector quantizers is to use the codebook design method introduced by Linde, Buzo and Gray [1980]. This method, known as *Generalized Lloyd Algorithm* (or *LBG* from the authors' names) uses a generalization of the Lloyd–Max optimality condition previously described to design a locally optimal codebook with no natural order or structure. LBG algorithm also improves the Lloyd–Max conditions in two important ways:

- First, it designs the codebook starting from a set of input samples and not from the input distribution that may not be available or analytically hard to express;
- Second, it solves the problem of specifying partition boundaries (very hard in a high dimensional space) by observing that the nearest–neighbor encoding rule always generates a *Voronoi* (or *Dirichlet*) partition.

LBG takes as input a *training set* $T = \{x_1, x_2, \ldots, x_L\}$ of n–dimensional vectors generated by the source. Then N vectors in T are randomly chosen to constitute the tentative centroids and by using these centroids, the corresponding Voronoi partition boundaries are determined. After the initialization, the algorithm iteratively refines the centroids and the partition boundaries by using optimality conditions similar to the ones described in the scalar case. Every iteration reduces the total distortion making the quantization error closer to a local minimum. While several stopping criteria are used, it is common to iterate the process until the reconstruction error on the training set is below a given threshold or until there is no further improvement. A number of modifications have been proposed in literature to speed up the convergence, see for example the paper by Kaukoranta et al. [1999].

Since the quantizer can be interpreted as the composition of an encoding function E and a decoding function D, the LBG algorithm can be seen as a process that optimizes in turn encoder and decoder until no further improvement is possible. The refinement of the Voronoi partition has the property of reducing the encoding error and the new set of centroids improves the decoding error.

LBG generates an unstructured, locally optimal vector quantizer. As a consequence of this lack of structure, the memory needed to store the codebook grows exponentially with the dimension of the vector. Furthermore, while the nearest-neighbor encoding rule avoids the explicit specification of the partition boundaries, encoding a source vector requires an exhaustive search in the dictionary to locate the centroid that minimizes the distortion. In the following, we will refer to this kind of vector quantizer as Exhaustive

Search Vector Quantizer (or ESVQ). The performance of an ESVQ provides an upper bound on the performance practically achievable by a VQ.

The interested reader will find further details and an exhaustive discussion on vector quantization in the excellent book by Gersho and Gray [1992].

4.2 Introduction to the Problem

To encode an information source with a VQ, a suitable codebook must be first designed by means of LBG or similar algorithm. Then, to represent each vector, the encoder must perform an exhaustive search in the codebook, locate the closest code vector and send its index to the decoder. The decoder, which shares with the encoder the knowledge of the codebook entries, decodes the index by retrieving the code word associated to it. From this description it is clear how codebook design, encoding and decoding are highly asymmetrical processes. The design of the codebook for an ESVQ is the most time consuming process, so this procedure is usually performed off–line. Then, due to the search, the encoding turns out to be much more expensive that the decoding.

Even if a conventional ESVQ requires exponentially growing computational and memory resources, the quality achieved by this type of VQ is frequently desirable in applications where only a limited amount of resources is available.

Several authors have proposed a number of alternatives to speed up codebook design; one of the most recent and interesting is due to Kaukoranta, Franti and Nevalainen [1999]. The method they propose, monitors cells activity during the LBG execution and performs

the computation only on the active ones. If one or more cells do not show any sign of change, the program does not compute new centroids and partition boundaries for that cell.

An alternative to the off–line construction of the codebook has been proposed by Constantinescu and Storer [1994] and Constantinescu [1995]. With a method similar to dictionary compression, the codebook starts with a default configuration and it is adaptively changed as the data are being compressed. This method called *Adaptive Vector Quantization* or in short *AVQ*, also changes dynamically the dimension of the vectors in the codebook. Recent experiments on variations of AVQ described in Rizzo, Storer and Carpentieri [1999, 2001] and Rizzo and Storer [2001] show that, on some information sources, AVQ exhibits asymptotically optimal compression and on several test images, outperforms image compression standards like JPEG.

When off–line construction is possible or desirable, imposing a structure to the VQ codebook is a practical method to speed up the nearest neighbor search. A structured VQ allows the use of fast search algorithms while marginally compromising compression.

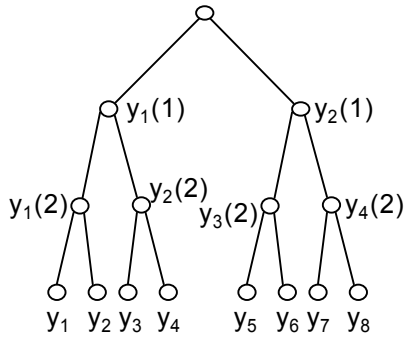


Figure 4.2: Tree Vector Quantizer.

One approach that is often used is to structure the codebook as a tree. The search starts by comparing the input vector to the code vectors in the first level codebook. Once the

closest match is located the search continues in the codebook associated to that code vector. The process is repeated until one of the leaves is reached (see Figure 4.2). In a *Tree Vector Quantizer* the search is performed in time $O(\log N)$ while the memory necessary to store the codebook and the vectors along the tree doubles. Tree VQs have been extensively studied, for instance by Wu [1990], Lin and Storer [1993] and Lin, Storer and Cohn [1991, 1992].

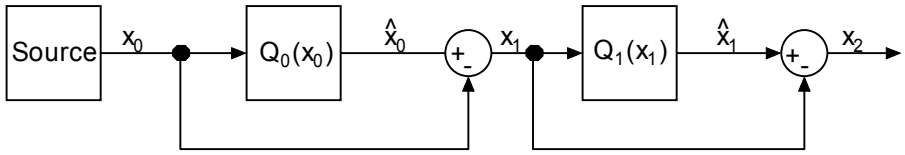


Figure 4.3: Residual Vector Quantizer.

Another solution that speeds up the search without increasing memory usage is the Residual Vector Quantizer depicted in Figure 4.3 and described in Barnes [1989], Barnes and Frost [1990], Frost, Barnes and Xu [1991]. In a residual quantizer a vector is encoded in multiple stages, through successive approximations. At every stage the nearest neighbor is found and subtracted to the vector. The quantization error vector (or *residual*) is sent to the next stage for a similar encoding. In a residual VQ, instead of a single index, the encoding consists of a sequence of indices, each one specifying a reconstruction level for each stage. The decoder retrieves the code words corresponding to the index sequence and adds them together in order to reconstruct the input.

Both being based on successive approximation, tree and residual structured vector quantizers allow progressive encoding. This means that the decoder can stop decoding

the sequence of indices at any time, resulting in a worse approximation of the input vector. Progressive transmission finds use in prioritized transmission channels and in scalable coding. Some applications of VQ to progressive encoding can be found, for example, in the work of Riskin [1990] and Kossentini, Smith and Barnes [1992].

The trellis is another structure that has been found particularly effective in organizing the encoding process. Works by Viterbi and Omura [1974], Colm Stewart [1981] and Ungerboeck [1982] pioneered the use of trellises in source coding and proved that trellises can be effectively used to take advantage of inter symbol dependencies. Trellis structured quantizers were first introduced and studied by Fischer, Marcellin and Wang [1991]. Other papers addressing various issues both in scalar and vector trellis quantization are Marcellin [1990], Marcellin and Fischer [1990], Fischer and Wang [1991], Laroia and Farvardin [1994], Wang and Moayeri [1992]. Jafarkhani and Tarokh [1998] addressed the problem of successive refinable coding with trellis quantizers.

The following section introduces a novel combination of residual and trellis vector quantization named *Trellis Coded Vector Residual Quantizer* (or *TCVRQ*). This quantizer, presented first in Motta and Carpentieri [1997], is a sub–optimal vector quantizer that, by combining residual quantization with a trellis graph, exhibits the memory savings typical of residual quantization while allowing a “virtual increase” of the quantization levels typical of the trellis based VQs.

4.3 Trellis Coded Vector Residual Quantization (TCVRQ)

TCVRQ has been first proposed in Motta and Carpentieri [1997] as a general–purpose sub–optimal VQ with low computational costs and small memory requirement that, despite its good performance, permits considerable memory savings when compared to traditional vector quantizers. In the same paper a greedy method for computing quantization levels has been outlined, and the performances of the TCVRQ have been experimentally analyzed.

In the following we will give a formal description of this quantizer, then present the greedy extension of the LBG that can be used to design the quantization levels.

Definition 4.2: Let x be a random vector in the n–dimensional Euclidean space $\mathbb{R}^n$; an N–level *Exhaustive Search Vector Quantizer* (or *ESVQ*) of $\mathbb{R}^n$ is a triple $Q = (A, \mathrm{Q}, P)$ where:

1. $A = \{y_1, y_2, ..., y_N\}$ is a finite indexed subset of $\mathbb{R}^n$ called codebook; y_i are called code vectors.

2. $P = \{S_1, S_2, ..., S_N\}$ is a partition of $\mathbb{R}^n$. The equivalence classes S_j of P satisfy:

$$\bigcup_{j=1}^{N} S_j = \mathbb{R}^n,$$

$$S_j \bigcap S_k = \varnothing \text{ for } j \neq k;$$

3. $\mathrm{Q} : \mathbb{R}^n \to A$ is a mapping that defines the relationship existing between codebook and partitions: $\mathrm{Q}(x) = y_j$ if and only if $x \in S_j$.

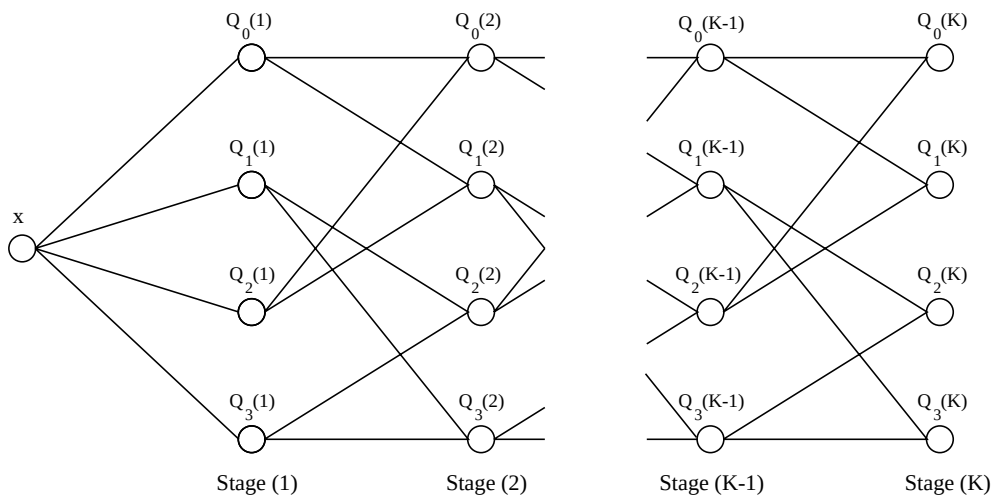


Figure 4.4: A K–stage Trellis Coded Vector Residual Quantizer; each node of the trellis is associated with an ESVQ that encodes the quantization error of the previous stage.

Equivalence classes and code vectors are designed so that the mean squared error introduced during the quantization is minimum. MSE has been chosen because it has a simple mathematical expression that can be easily minimized and gives a good measure of the random error introduced in the compression. However results can be generalized to other metrics (see for example Barnes [1989]).

Definition 4.3: A *Residual Quantizer* consists of a finite sequence of ESVQs $Q_1, Q_2, ..., Q_K$ such that Q_1 quantizes the input $x = x_1$ and each Q_i, $1 < i \leq K$ encodes the error (or residual) $x_i = x_{i-1} - \mathrm{Q}(x_{i-1})$ of the previous quantizer Q_{i-1}, $1 < i \leq K$.

The output is obtained by summing the code words:

$$y = \sum_{i=1}^{K} \mathrm{Q}_i(x_i)$$

Definition 4.4: A *multistage* (or *layered*) K–stage graph is a pair $G = (V, E)$ with the following properties:

1. $V = \{v_1, v_2, ..., v_n\}$ is a finite set of vertices such that:

$$V = \bigcup_{k=1}^{K} V_k \ , \ V_k \subset V \quad \text{for} \quad 1 \le k \le K \ \text{ and}$$

$$V_i \bigcap V_j = \varnothing \quad \text{for} \quad i \ne j, 1 \le i, j \le K \ ;$$

2. $E = \{(v_i, v_j) : v_i \in V_k, v_j \in V_{k+1}, 1 \le k < K\}$ is a finite set of edges.

According to the definition, a trellis is a multistage graph since it can be divided into layers and edges that connect the nodes from one layer to the next.

The trellis coded residual quantizer associates a residual quantizer to each node of a trellis. Since each layer of this graph is not fully connected to the next layer (as, for example, in the trellis described in Ungerboeck [1982]), not every sequence of residual quantizers is allowed and, by designing the residual quantizers appropriately, a bit saving can be achieved for each stage.

Each encoded vector is fully specified by the path on the graph and by the indexes of the code words of the quantizers in nodes along the path.

When we use, for example, the trellis showed in the Figure 4.4 and each $\mathrm{Q}_i(j)$ is a N–level RVQ, an output vector is specified by $K+1$ bit to encode the path and $K \cdot \log_2(N)$ bit for the code vectors indices. In a residual quantizer that achieves the same bit rate, each stage has $4N$ levels and $K \cdot \log_2(4N)$ bit are necessary and the trellis configuration allows a “virtual doubling” of the available quantization levels.

A formal definition of the TCVRQ is the following:

Definition 4.5: A *Trellis Coded Vector Residual Quantizer* is a pair $T = (G, Q)$ where:

1. $G = (V, E)$ is a Trellis multistage graph with $|V| = n$ and K stages;

2. $Q = (Q_1, Q_1, ..., Q_n)$ is a finite set of ESVQs, $|V| = |Q|$ and each $Q_i \in Q$ is associated to the vertex $v_i \in V$;

3. The ESVQ Q_i encodes the residual of Q_j if and only if $(v_i, v_j) \in E$.

With reference to Figure 4.4, TCVR quantization of an input vector starts from the nodes of the first layer. The vector is encoded with the four quantizers present in the first stage nodes. The four code vectors resulting from the quantization are subtracted from the input and four possible residuals propagate to the next stage. Nodes at the second stage have two entering edges, each carrying a residual. Quantization is performed on both residuals and the one that can be better encoded (in the sense that current quantization will generate a smaller error) is quantized and its residual propagated again. Quantization ends at the last stage, where the path and the code vectors that generated the smaller residual are selected.

This method uses an approach similar to Viterbi search algorithm (Viterbi and Omura [1974]) and, in this specific framework, is not optimal. It is well known that the partitions generated by a residual quantizer are not necessarily disjoint. This also happens with our TCVRQ. The Viterbi search algorithm will exclude at every step one half of the paths because they do not look promising. Unfortunately this does not mean that they cannot generate a quantization error that is smaller than the one generated by the selected paths.

When computing power is not an issue, a full search can be used on the trellis, or, as a compromise between a Viterbi and a full search, an M–search algorithm that keeps "alive" several paths instead of only four.

Once the trellis structure has been chosen, the design of the codebooks associated to the node quantizers is the next problem.

A greedy codebook design was proposed in Motta and Carpentieri [1997, 1997b]. This method is based on an extension of the LBG algorithm (Linde, Buzo and Gray [1980]). The design of the quantization levels for the ESVQs associated to each node of the trellis is performed stage by stage, sequentially from stage 1 to stage K, by training the LBG on the residuals generated through the entering edges.

Obviously this design is not optimal; nevertheless since it respects the structure of the quantizer, for a small number of stages it is sufficient to achieve competitive performance. When the number of stages increases, both greedy design and Viterbi search show their weakness. Partitions generated by TCVRQ overlaps and two different paths may encode equally well the same input, resulting in a waste of coding space.

We also note that, increasing of the number of stages, the performance degrade gracefully. This is mainly due to the nature of the residual structure in which the energy of the coding error decreases quickly with the number of stages.

4.4 Necessary Condition for the Optimality of a TCVRQ

Necessary conditions for the minimum distortion of a quantizer are generally determined by taking the derivative of the distortion function with respect to the partition boundaries while keeping the centroids fixed and then by taking the derivative of the distortion with respect to the centroids while keeping the partition boundaries fixed.

This technique has been widely used in literature since the unpublished report written by Lloyd [1957] and the paper by to Max [1960] where the necessary conditions for the optimality of a scalar quantizer are derived.

Unfortunately, this method cannot be applied directly to the determination of similar optimality conditions in the case or a TCVRQ. The distortion introduced by a TCVRQ on the coding of a vector $\mathbf{x}^1$ takes the form:

$$D\left(\mathbf{x}^1, \hat{\mathbf{x}}^1\right) = \int_{\mathbf{x}^1 \in \Re^n} \cdots \int_{\mathbf{x}^P \in \Re^n} d\left[\mathbf{x}^1, \sum_{p=1}^{P} Q^p(\mathbf{x}^p)\right] dF_{\mathbf{x}^1,\ldots,\mathbf{x}^P}$$

where the sum is performed along the winning path.

In general, the joint probability density function $dF_{\mathbf{x}^1,\ldots,\mathbf{x}^P}$ is not known and, because of the residual structure, it depends, in a complicate fashion, on the sequence of codebooks and boundaries.

In his Ph.D. Thesis, Barnes [1989] proposes a very general approach to the solution of this problem. He starts by defining a quantizer that is not residual but that, by construction, is completely equivalent to the RQ that must be analyzed. In Barnes [1989]

the name of "*Equivalent Quantizer*" is used, while in Barnes and Frost [1993] the same concept is used with the name of "*Direct Sum Quantizer*".

The basic idea is to construct an ESVQ that has partitions and code vectors derived from a residual quantizer. Optimality conditions for this ESVQ can be derived with the technique described before and then transformed in the corresponding optimality conditions for the original quantizer.

***Definition 4.6*:** A *Direct Sum* (or *Equivalent*) Quantizer is a triple (A^e, Q^e, P^e) consisting of:

1. A direct sum codebook A whose elements are the set of all the possible sums of stage wise code vectors taken along all the possible paths, one vector from each stage $A^e = A^1 + A^2 + \ldots + A^P$. There are $N^e = \prod_{p=1}^{P} A^p$ direct sum code vectors in A. Direct sum code vectors are indexed by P–tuples $\mathbf{j}^P = (j^1, j^2, \ldots, j^P)$ and can be written as $\mathbf{y}^e(\mathbf{j}^P) = \sum_{p=1}^{P} \mathbf{y}^p_{j^p}$.
2. A direct sum partition P^e is the collection of the direct sum cells. The $\mathbf{j}^P$–th direct sum cell is the subset $S^e(\mathbf{j}^P) \subset \Re^n$ such that all $\mathbf{x}^1 \in S^e(\mathbf{j}^P)$ are mapped by the corresponding residual quantizer into $y^e(\mathbf{j}^P)$, that is

$$S^e(\mathbf{j}^P) = \left\{ \mathbf{x}^1 : \sum_{p=1}^{P} Q^p(\mathbf{x}^p) = \mathbf{y}^e(\mathbf{j}^p) \right\}.$$

3. The direct sum mapping $Q^e : \mathbb{R}^n \to A^e$ is defined as $Q^e(\mathbf{x}^1) = \mathbf{y}^e(\mathbf{j}^P)$ if and only if $\mathbf{x}^1 \in S^e(\mathbf{j}^P)$.

The average distortion of the direct sum quantizer is given in terms of the known source probability density function $F_{\mathbf{X}^1}$ and so its minimization is substantially easier:

$$D\left(\mathbf{x}^1, \hat{\mathbf{x}}^1\right) = \int d\left[\mathbf{x}^1, Q^e(\mathbf{x}^1) dF_{\mathbf{X}^1}\right]$$

By construction, the direct sum single stage quantizer defined before produces the same representation of the source input $\mathbf{x}^1$ as does the corresponding TCVRQ. So the two distortions must be equal and we can minimize the distortion of a TCVRQ by minimizing the distortion of its equivalent direct sum quantizer.

$$Q^e(x^1) = \sum_{p=1}^{P} Q^p(\mathbf{x}^p).$$

It is necessary to observe that in general, TCVRQ partitions, as well as the direct sum cells of its equivalent quantizer, are not disjoint. Because of this reason, the codeword must be specified by giving both the path along the trellis and the indices of the code vectors selected along the coding path.

***Theorem 4.1*:** For a given TCVRQ and for a given random variable $\mathbf{X}^1$ with probability density function $F_{\mathbf{X}^1}$, let the partitions $\left\{P^1, P^2, \ldots, P^P\right\}$ and all the codebooks except A^ρ with $\rho \in \{1, \ldots, P\}$ be fixed, then the $\mathbf{y}_{k_\rho}^\rho$ quanta in A^ρ that minimize the mean squared error must satisfy:

$$\mathbf{y}_{k_\rho}^\rho = \int_{\Re^n} \xi^P f_{\Xi^\rho \left| \mathbf{x}^\rho \in S_{k_\rho}^\rho \right.} \left(\xi^P \left| \mathbf{x}^\rho \in S_{k_\rho}^\rho \right.\right) d\xi^\rho$$

for every $p \in \{1, \ldots, P\}$ and $k_\rho \in \{0, \ldots, N^\rho - 1\}$.

Proof: The proof is based on the condition for the optimality of a residual quantizer proved in Barnes [1989] and Barnes and Frost [1993].

For a given random variable $\mathbf{X}^1$ with probability density function $F_{\mathbf{X}^1}$ we want to find a set of code vectors that locally minimizes the distortion:

$$D_{mse} = \int_{\mathbb{R}^n} \left[\mathbf{x}^1 - Q^e(\mathbf{x}^1) \right]^2 f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1 = \sum_{\text{all } \mathbf{j}^P} \int_{S^e(\mathbf{j}^P)} \left[\mathbf{x}^1 - \mathbf{y}^e(\mathbf{j}^P) \right]^2 f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1$$

while keeping the equivalent partitions P^e (or the TCVRQ partitions $\left\{ P^1, P^2, \ldots, P^P \right\}$) fixed. If we assume that all the codebook, except A^ρ with $\rho \in \{1, \ldots, P\}$, are held fixed, then D_{mse} can be minimized with respect each code vector $\mathbf{y}^\rho_{k_\rho}$ in A^ρ by setting the partial derivative of D_{mse} with respect the direct sum code vectors that contain $\mathbf{y}^\rho_{k_\rho}$ in theirs sum.

Using the distortion formulated in terms of the component code vectors and setting its partial derivative equal to zero gives:

$$D_{mse} = \sum_{\mathbf{j}^P} \int_{S^e(\mathbf{j}^P)} \left[\mathbf{x}^1 - \mathbf{y}^e(\mathbf{j}^P) \right] \left[\frac{\partial \mathbf{y}^e(\mathbf{j}^P)}{\partial \mathbf{y}^\rho_{k_\rho}} \right] f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1 = 0$$

If we indicate with $H^\rho_{k_\rho} = \left\{ \mathbf{j}^P : j^\rho = k_\rho \right\}$ the set of indices $\mathbf{j}^P = \left(j^1, j^2, \ldots, j^\rho, \ldots, j^P \right)$ that have ρ–th component equal to k_ρ, then the partial derivative assumes value:

$$\frac{\partial y^e(\mathbf{j}^P)}{\partial y^\rho_{k_\rho}} = \begin{cases} 1, & \text{if } \mathbf{j}^P \in H^\rho_{k^\rho} \\ 0, & \text{otherwise} \end{cases}$$

Solving with respect to $\mathbf{y}_{k_\rho}^{\rho}$ gives the result:

$$\mathbf{y}_{k_\rho}^{\rho} = \frac{\sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} \int_{S^e(\mathbf{j}^P)} \left(\mathbf{x}^1 - \sum_{\substack{p=1 \\ p \neq \rho}}^{P} \mathbf{y}_{j^p}^{p} \right) f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1}{\sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} \int_{S^e(\mathbf{j}^P)} f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1}$$

The expression $\sum_{\substack{p=1 \\ p \neq \rho}}^{P} \mathbf{y}_{j^p}^{p}$ represents a codeword from which the ρ–th node along the path has been removed. We indicate this quantity as $g^{\rho}\left(\mathbf{j}^P\right) = y^e\left(\mathbf{j}^P\right) - \mathbf{y}_{j^\rho}^{\rho} = \sum_{\substack{p=1 \\ p \neq \rho}}^{P} \mathbf{y}_{j^p}^{p}$

Defining an indicator function $I_{S^e(j^P)}$ as:

$$I_{S^e(j^P)} = \begin{cases} 1, & \text{if } \mathbf{x}^1 \in S^e(j^P) \\ 0, & \text{otherwise} \end{cases}$$

it is possible to rewrite the expression for $\mathbf{y}_{k_\rho}$ and interchange the order of summation and integration:

$$\mathbf{y}_{k_\rho} = \frac{\int_{\mathbb{R}^n} \sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{S^e(j^P)} \left[\mathbf{x}^1 - g^{\rho}\left(\mathbf{j}^P\right) \right] f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1}{\int_{\mathbb{R}^n} \sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{S^e(j^P)} f_{\mathbf{X}^1}(\mathbf{x}^1) d\mathbf{x}^1}$$

For each $\mathbf{j}^P$ define for all $\mathbf{x}^1 \in S^e(\mathbf{j}^P)$ the "grafted" residual values $\xi^{\rho} = \mathbf{x}^1 - g^{\rho}\left(\mathbf{j}^P\right)$. The cell $G^{\rho}(\mathbf{j}^P)$ contains all grafted residuals ξ^{ρ} formed from the $\mathbf{x}^1 \in S^e(\mathbf{j}^P)$. Its indicator function is defined as:

$$I_{G^{\rho}(\mathbf{j}^P)} = \begin{cases} 1, & \text{if } \xi^{\rho} \in G^{\rho}(\mathbf{j}^P) \\ 0, & \text{otherwise} \end{cases}$$

Using the relation between ξ^{ρ} and $\mathbf{x}^1$ we can change the variable of integration:

$$\mathbf{y}_{k_\rho} = \frac{\int_{\mathbb{R}^n} \xi^{\rho} \sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{G^{\rho}(\mathbf{j}^P)} f_{\mathbf{X}^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right] d\xi^{\rho}}{\int_{\mathbb{R}^n} \sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{G^{\rho}(\mathbf{j}^P)} f_{\mathbf{X}^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right] d\xi^{\rho}}$$

Expanding the probability density function $f_{\mathbf{X}^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right]$ as a sum of conditional probability density functions:

$$f_{\mathbf{X}^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right] = \sum_{k_\rho=0}^{N^{\rho}-1} f_{\mathbf{X}^1 \left| \mathbf{x}^1 \in H_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho} \left| \mathbf{x}^1 \in H_{k_\rho}^{\rho}\right.\right] \operatorname{Prob}\left(\mathbf{x}^1 \in H_{k_\rho}^{\rho}\right)$$

Expressing the conditioning in terms of the ρ–th causal residual $\mathbf{x}^{\rho}$ we obtain:

$$f_{\mathbf{X}^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right] = \sum_{k_\rho=0}^{N^{\rho}-1} f_{\mathbf{X}^1 \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right] \operatorname{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right)$$

This expression can be substitute to the sum common to numerator and denominator in the previous expression for $\mathbf{y}_{k_\rho}$ giving:

$$\begin{aligned}\sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{G^{\rho}(\mathbf{j}^P)} f_{X^1}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho}\right] &= \sum_{k^{\rho}=0}^{N^{\rho}-1} \sum_{\mathbf{j}^P \in H_{k_\rho}^{\rho}} I_{G^{\rho}(\mathbf{j}^P)} f_{\mathbf{X}^1 \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right] \operatorname{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right) \\ &= \sum_{\text{all } \mathbf{j}^P} I_{G^{\rho}(\mathbf{j}^P)} f_{\mathbf{X}^1 \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^P) + \xi^{P} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right] \operatorname{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right)\end{aligned}$$

Dividing both sides by $\operatorname{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right)$:

$$f_{\Xi^{\rho} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.}\left(\xi^{P} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right) = \sum_{\text{all } \mathbf{j}^P} I_{G^{\rho}(\mathbf{j}^P)} f_{\mathbf{X}^1 \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^P) + \xi^{\rho} \left| \mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right] \operatorname{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right)$$

Since $\text{Prob}\left(\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right) = \text{Prob}\left(\mathbf{x}^{1} \in H_{k_\rho}^{\rho}\right)$ we can express the last equation as:

$$f_{\Xi^{\rho}\left|\mathbf{x}^{\rho}\in S_{k_\rho}^{\rho}\right.}\left(\xi^{P}\left|\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right) = \frac{\sum_{\mathbf{j}^{P}\in H_{k_\rho}^{\rho}} I_{G^{\rho}(\mathbf{j}^{P})} f_{\mathbf{X}^{1}\left|\mathbf{x}^{\rho}\in S_{k_\rho}^{\rho}\right.}\left[g^{\rho}(\mathbf{j}^{P}) + \xi^{\rho}\right]}{\text{Prob}\left(\mathbf{x}^{1} \in H_{k_\rho}^{\rho}\right)}$$

The substitution of the last three relations in the expression for $\mathbf{y}_{k^{\rho}}$ completes the proof by giving:

$$\mathbf{y}_{k_\rho}^{\rho} = \int_{\mathbb{R}^{n}} \xi^{P} f_{\Xi^{\rho}\left|\mathbf{x}^{\rho}\in S_{k_\rho}^{\rho}\right.}\left(\xi^{P}\left|\mathbf{x}^{\rho} \in S_{k_\rho}^{\rho}\right.\right) d\xi^{\rho}$$

4.5 Viterbi Algorithm

The algorithm that has been used to encode a vector is based on the method introduced by Viterbi [1967] to decode error correcting convolutional codes. Two years after its introduction, Omura [1969] recognized that the Viterbi algorithm is equivalent to the dynamic programming solution of the problem of finding the shortest path through a weighted graph, so in the following, we will use “shortest” as a synonym of “minimum cost” path.

The regular, multi stage structure of a trellis allows an efficient implementation of this algorithm with a minimum book keeping and since its introduction it has been the core of most error correcting decoding algorithms.

Viterbi algorithm works by finding the shortest path on a trellis whose edges have been labeled with an additive cost metric. Starting from an initial node, a simple

computation is carried out stage by stage to determine the shortest path ending in the nodes belonging to the current stage. Each processed node is labeled with the shortest path and with its corresponding cost. These partial solutions are sometimes called "survivor" paths.

Let's suppose that a node n_j^i of the i–th stage has two entering edges (h, j) and (k, j) leaving the nodes n_h^{i-1} and n_k^{i-1} in the stage $i-1$ and that the edges are respectively labeled with costs $W_{h,j}$ and $W_{k,j}$ (see Figure 4.5). If the algorithm has already processed the $(i-1)$–th stage, then the nodes n_h^{i-1} and n_k^{i-1} are labeled with the survivor paths $(PATH_h^{i-1}, COST_h^{i-1})$ and $(PATH_k^{i-1}, COST_k^{i-1})$. Then the survivor path for the node n_j^i will be computed as $(PATH_t^i * (t, j), COST_t^i + W_{t,j})$ where $t = \underset{y \in \{h,k\}}{\operatorname{argmin}} \left(COST_y^{i-1} + W_{y,j} \right)$.

The operator "$*$" is used to indicate the concatenation of a new edge to a path.

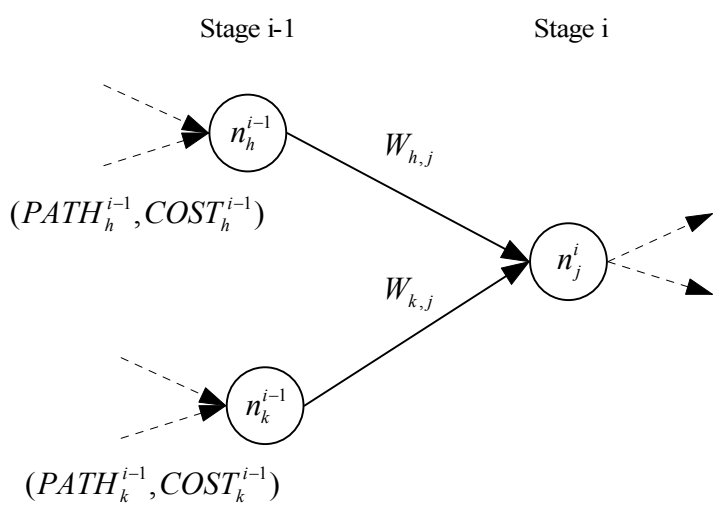


Figure 4.5: Viterbi algorithm

Since all paths start in a node n^0 and end in a node n^N, it is easy to prove that, under the assumption of an additive metric, Viterbi algorithm labels the ending node with the minimum cost path. The proof is carried out by induction, where the inductive step assumes that the nodes n_h^{i-1} and n_k^{i-1} are labeled with the minimum cost paths starting in n^0 and ending in n_h^{i-1} and n_k^{i-1} respectively. Since the only way to reach the node n_j^i is through the edges (h, j) and (k, j), every path starting in n^0 and ending in n_j^i cannot be shorter than $(PATH_t^i * (t, j), COST_t^i + W_{t,j})$ where $t = \underset{y \in \{h,k\}}{\operatorname{argmin}} \left(COST_y^{i-1} + W_{y,j} \right)$. The initial node n^0 is labeled with an empty path of cost zero.

The TCVRQ uses a variant of this algorithm to implement a greedy search of the representative code vector that scores a minimum reconstruction error for a given input. The input is fed to the quantizer through the initial node, then the metric of each of the out edges is computed. The metrics is the minimum error achieved by the small exhaustive search vector quantizer associated to the node. The node is then labeled with the quantization residual vector plus the edge information and the index of the code vector selected in the node ESVQ. The computation is performed stage by stage, until the ending node contains the survivor residual (and so its mean squared error with the input) and the sequence of edges and code vector indices. This sequence is the information that, stored or sent to the decoder, allows a reconstruction of the input.

Since the metric is computed on the fly and depends on the quantization choices performed at early stages, not every combination of code vectors is examined. While this has the advantage of drastically reducing the computing time, the search is clearly sub optimal. In practice, the ESVQs that are associated to each node have code vectors of

decreasing energy. Meaning that, because of the residual structure, the contribution of the first stages to the final error is more relevant. This is enough to guarantee that in practice this search achieves a quantization error that is very close to the one achieved in the case of an exhaustive search that considers all the possible sums of code vectors along the trellis.

4.6 Experimental Results

Vector quantization is a compression technique that can be used alone or it can be combined with other methods to form a complex data compression system. To assess the performance of the TCVRQ we ran three series of experiments. The first set of experiments deals with natural data and involves direct quantization of gray–level still images. In these experiments we used the greedy codebook design outlined before and Viterbi.

An area in which powerful yet simple vector quantizers are extremely useful is low bit rate speech coding; works by Juang and Gray Jr. [1982], Makoul, Roucos, and Gish [1985], Paliwal and Atal [1991] and Bhattacharya, LeBlanc, Mahmoud, and Cuperman [1992] all stress that efficient vector quantization is a key problem in speech coding. Because of this, the second set of experiments assesses the performance of a TCVRQ with a greedy designed codebook and Viterbi search when used to encode linear prediction parameters in a low bit rate speech codec.

The third series of experiments, performed on theoretical random sources, compares the performance of the Viterbi search versus the performance of a more complex and time–consuming full trellis search.

4.6.1 Gray–level Image Coding

TCVRQ performance has been assessed on natural data sources with several experiments that encoded directly several gray–level images taken from a standard data set. Results in quantizing these images were compared to those obtained by ESVQ.

The image set was composed of 28 standard gray–level images, 512x512 pixels, 256 gray levels. Images were downloaded from "ftp//links.uwaterloo.ca/pub/BragZone" and are also available from several other web sites that collect reference images used in data compression.

The set was partitioned into a training set, consisting of 12 images and a test set counting 16 images. The pixels in the training set images were partitioned in vectors of 3x3 and 4x4 pixels.

Following the convention present in the literature, error was measured as Signal to Quantization Noise Ratio (or SQNR) and expressed in dB:

$$SQNR(X,\hat{X}) = 10 \cdot \log_{10} \frac{\sigma^2\left(X\right)}{\sigma^2\left(X-\hat{X}\right)}$$

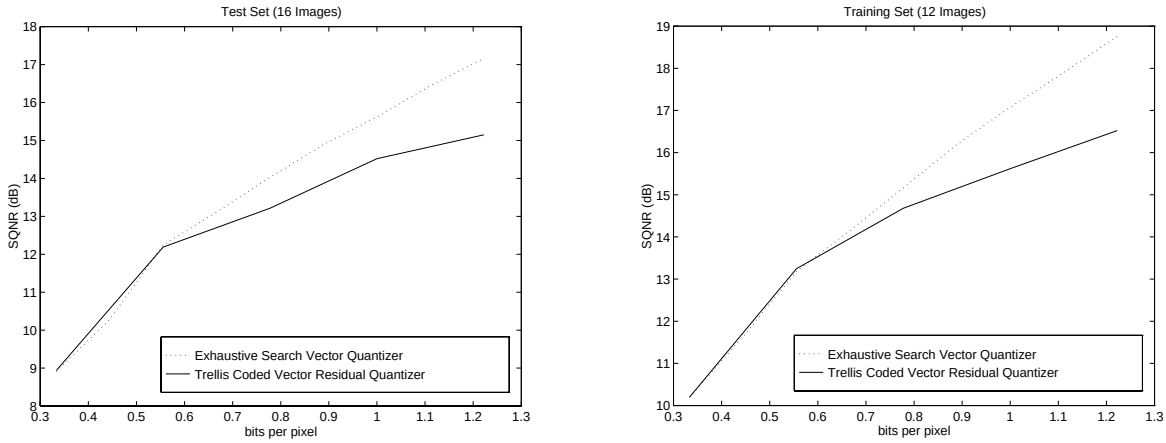


Figure 4.6: SQNR for TCVRQ and ESVQ compared for different bit–rates and for images in the training (left) and in the test sets (right).

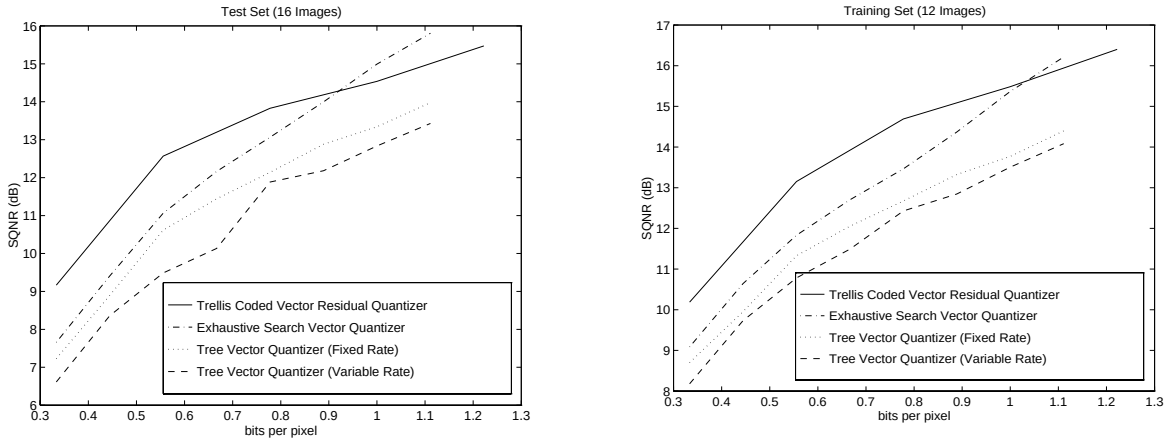


Figure 4.7: Performance of the TCVRQ compared to J. Goldschneider's VQ package.

Figure 4.6 shows that, when used in the direct quantization of gray–level still images, TCVRQ performs very close to a locally optimal ESVQ. For low bit rates, ranging from 0.3 to 1 bit per pixel, TCVRQ quantization error is very close to the optimum while we gain in terms of speed and memory. Increasing the bit rate by increasing the number of stages tests the limits of the greedy search and codebook design. Since partitions overlap,

similar code vectors gets associated to different code words resulting in an inefficient use of the coding space.

TCVRQ was also tested, on the same set of images, with another popular VQ package, written by J. Goldschneider and freely available on Internet at the ftp address: "ftp://isdl.ee.washington.edu/pub/VQ/code". This package consists of two different kinds of tree quantizers that achieve fixed and variable rate and an ESVQ that performs full search on the codebook generated for the tree quantizers.

As it is shown in Figure 4.7, for low bit rates, trellis coded vector quantization outperforms tree–structured quantization in terms of SQNR. It is also important to note that, due to their structure, tree quantizers use twice the memory of an ESVQ with the same number of levels and so their memory usage grows exponentially with the vector size. In contrast, TCVRQ only uses an amount of memory that grows linearly with the vector dimension.

4.6.2 Low Bit Rate Speech Coding

The fact that the TCVRQ is a general–purpose VQ, with low computational costs and small memory requirements makes this quantizer suitable for low bit–rate speech coding applications. High quality, low bit–rate speech codecs are necessary when speech signal have to be encoded on a narrow–band channel while preserving both voice intelligibility and speaker identification.

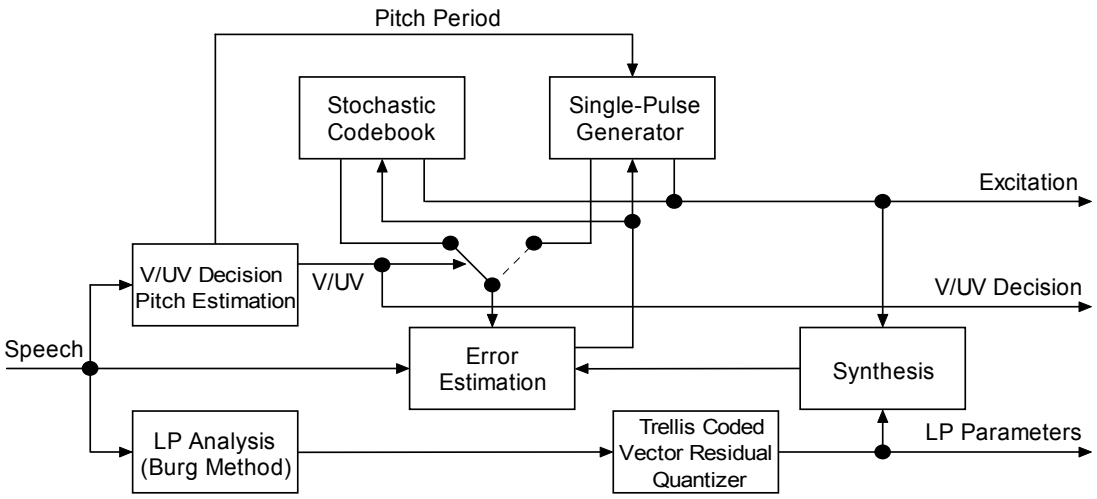


Figure 4.8: The low bit rate speech codec used in our experiments.

We have also experimentally evaluated the performances of TCVR quantization when used in a low bit–rate Linear Prediction based speech codec. We used the low bit–rate speech codec depicted in Figure 4.8. This low bit rate speech codec, originally proposed in Motta [1993], improves upon a scheme previously introduced by Atal and Remde [1982].

The codec is a hybrid single–pulse codebook excited codec where the voice signal, sampled at 8 KHz with 16 bit per sample, is analyzed by using linear prediction. The signal is divided in frames of 80 consecutive samples and every frame is classified as "Voiced" or "Unvoiced" by thresholding the peak of the autocorrelation. For voiced frames, the period of the main pitch is also estimated.

Depending on their voiced/unvoiced classification, frames are synthesized at 2400 bit per second by using a single–pulse or a stochastic excitation vector. If the frame is voiced, number and positions of the pulses are computed with a dynamic programming algorithm that minimizes the squared error between the original and the reconstructed

signal. Unvoiced frames are synthesized in a CELP like manner, by choosing in closed loop the excitation signal in a stochastic codebook that minimizes the reconstruction error.

The TCVRQ is used to quantize linear prediction parameters that are represented in terms of *Line Spectrum Frequencies* (see Rabiner and Schafer [1978]). In this kind of codec, it is well known that the quantizer design is critical and the quality of the synthesized signal strongly depends on accurate quantization of the linear prediction parameters. LSFs were encoded at bit rates between 1.9 – 2.4 bit per parameter with a TCVRQ in a 10–stages trellis configuration.

The quantizer was trained on LSFs derived from a set of 76 sentences pronounced by different speakers of both sexes in the most common European languages: English, French, Dutch, German, Greek, Spanish, Italian.

The test set was composed by 12 English sentences spoken by one male and one female speaker. Used sentences are phonetically rich and well known to be hard to encode:

- "Why were you away a year Roy ?" (Voiced);
- "Nanny may know my meaning" (Nasals);
- "The little blanket lies around on the floor" (Plosives);
- "His vicious father has seizures" (Fricatives);
- "The problem with swimming is that you can drown" (Voiced Fricatives);
- "Which tea–party did Baker go to ?" (Plosives and Unvoiced Stops).

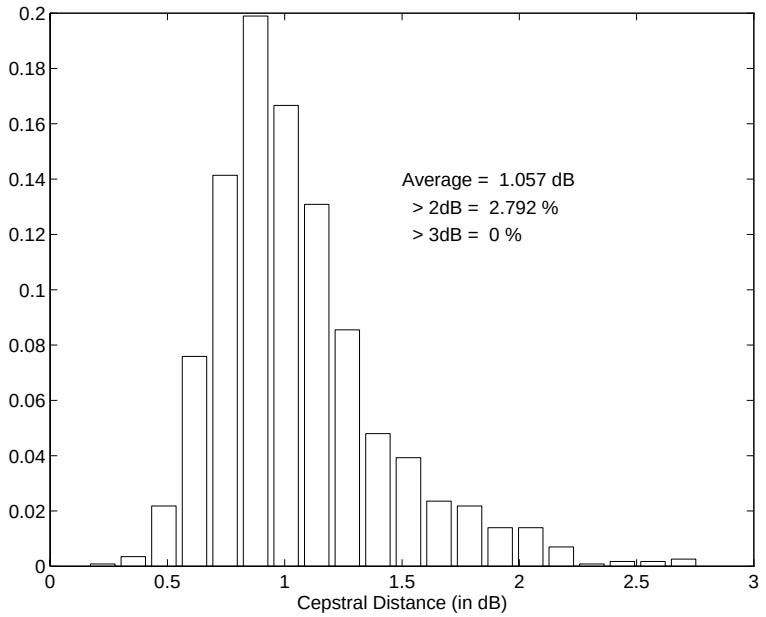


Figure 4.9: Cepstral Distance obtained quantizing the LP coefficients of the test set with a rate of 2.4 bit per parameter.

Low bit rate speech quality cannot be measured by using signal to noise ratio. SQNR gives an unreliable measure that is poorly correlated to the mean opinion score assessed by human listeners. Much more informative is the use of a spectral measure known in literature as *Cepstral Distance*. Based on the work by Bogert, Healy and Tukey [1963], the cepstral distance is a perceptually motivated metric widely used to assess low bit rate speech coding (see Rabiner and Schafer [1978] and Gersho [1994]).

Figure 4.9 shows the distribution of the cepstral distance between the original and the quantized linear prediction parameters. The histogram shows that the average CD obtained on the test set is approximately 1 dB and the percentage of frames with a CD greater than 2 dB is reasonably small. These results satisfy the conditions for a transparent quantization expressed in Paliwan and Atal [1991]. This has been also

verified by informal listening tests that confirmed that the TCVRQ is capable of performing transparent quantization with 1.9 – 2.4 bit per linear prediction parameter.

4.6.3 Random Sources

Degradation introduced by the non–optimal Viterbi search was assessed on a number of random sources. Random sources are commonly used in literature to evaluate scalar and vector quantizers since they can model natural data in a controlled manner.

The sources generate data according to the distributions: Uniform, Gaussian $N(0,1)$, Laplacian $L(0,1)$, Bimodal $f(x) = 0.3 \cdot N(-1,1) + 0.7 \cdot L(1,1)$ and Gauss–Markov with coefficients $a_1 = 1.515$ and $a_2 = -0.752$. Random sources with these parameters were also used in Marcellin [1987] and Barnes [1989].

The Gauss–Markov source is the only one in this group that is not memoryless.

Bps	Uniform	Gauss	Laplace	Bimodal	Markov
0.3	1.24	1.17	1.18	2.20	4.40
0.6	2.80	2.52	2.58	4.16	7.63
0.9	4.56	3.95	4.03	6.50	10.19
1.2	5.89	5.19	5.21	7.80	12.20
1.5	7.23	6.49	6.43	9.03	13.98
1.8	8.70	7.84	7.67	10.32	15.51
2.1	10.00	9.11	8.85	11.46	16.86
2.4	11.31	10.38	10.05	12.57	18.10
2.7	12.70	11.70	11.22	13.68	19.27
3.0	13.99	12.96	12.37	14.75	20.39

Table 4.1: TCVRQ on random data sources, Viterbi search. SNR expressed in dB.

Bps	Uniform	Gauss	Laplace	Bimodal	Markov
0.3	1.24	1.17	1.18	2.87	4.40
0.6	2.82	2.54	2.63	5.41	7.77
0.9	4.64	4.04	4.19	7.46	10.48
1.2	6.26	5.55	5.57	8.86	12.77
1.5	7.82	7.11	7.02	10.26	14.72
1.8	9.44	8.75	8.53	11.74	16.49
2.1	11.08	10.35	10.01	13.10	18.19
2.4	12.77	11.98	11.48	14.51	19.61

Table 4.2: TCVRQ on random data sources, full trellis search. SNR expressed in dB.

Bps	Uniform	Gauss	Laplace	Bimodal	Markov
0.9	5.59	4.74	5.04	7.64	11.47
1.2	8.13	7.20	7.47	10.27	14.62

Table 4.3: Reference ESVQ on random data sources. SNR expressed in dB.

Exploitation of the inter symbol dependence is evident in the Tables 4.1, 4.2 and 4.3.

Quantization error on the Gauss–Markov source is always significantly better than the other sources at any bit rate. In these experiments random data are first generated, then grouped in vectors of ten values and quantized. The trellis has always the same number of levels per stage; higher bit rate is achieved by increasing the number of stages. Tables 4.1 and 4.2 show the SNR in dB introduced by the trellis coded vector quantizer with Viterbi and with a full codebook search and, as a reference, the error introduced by an exhaustive search vector quantizer whose codebook was designed with LBG algorithm.

Table 4.3 shows reference results obtained on the same sources by the exhaustive search vector quantizer. Not every point is present in the Table 4.3 due to the complexity of the ESVQ and TCVRQ with full search. Nonetheless it is possible to observe that the

performance of the TCVRQ is close to that of the ESVQ and the degradation introduced by the Viterbi algorithm increases slowly with the number of stages.

LOSSLESS IMAGE CODING

5.1 Background

Most applications of image compression deal with images that will be used in frameworks that do not require extreme fidelity. If a photographic picture is compressed by using JPEG or JPEG–2000, two state of the art lossy compression algorithms, most viewers will not be able to perceive any quality degradation from the original. There are also situations like the transmission of a picture over a low bandwidth channel like the Internet, where high compression is much more desirable than an exact reproduction of the original.

One of the main drawbacks of the lossy compression algorithms that are commonly used for image compression is that, while the overall quality of the picture can be controlled to some extent, nothing can be said about the error that affect single pixels. This is not sufficient in applications, like medical imaging for example, where uncontrolled distortion in a portion of the image may change significantly the interpretation of the data. In these cases, it is necessary to use lossless compression algorithms, i.e. algorithms in which the compression is reversible and no information is lost in the coding process.

During JPEG standardization, which was mainly focused on the lossy compression of natural images, the committee felt the need for a lossless coding mode that could complete the standard. After evaluating few proposals, a lossless method based on the use of a set of fixed predictors was added to the draft. However, it was immediately clear that the lossless mode added to JPEG was not adequate, so the committee decided to issue a Call for Contributions for a new lossless standard that was named ISO/IEC JTC 1.29.12, also known as lossless JPEG or JPEG–LS. In the same call for contribution, the need for a *near–lossless* mode was also anticipated. In near–lossless mode, some error can be introduced in the coding process but it must be possible to specify a tight upper bound on the maximum error that is introduced on each pixel. Immediately after the call for contributions, the field of gray–level lossless image compression received a renewed attention from many researchers in the data compression community because the design of an efficient lossless compressor turned out to be a challenging task.

The contributions that were submitted were all effective in compressing images, while keeping low the computational complexity and the memory requirements. On the other hand, most of them used ad–hoc heuristics and, even if further improvements appeared unlikely, it was not completely clear how much their compression was close to the entropy of the image.

The best proposal in terms of compression ratio was the Adaptive Lossless Image Codec (CALIC) presented by Wu and Memon [1996, 1997], however due to its higher complexity, LOCO–I, proposed by Weinberger, Seroussi and Sapiro [1996] was selected as the core of the final draft.

In a subsequent paper [1996] Wu discussed some of the experiments performed during the development of CALIC and concluded that:

- Linear prediction is not effective in capturing edges and fast transitions in image luminosity;
- Global optimization seems unable to improve the performance of CALIC;
- CALIC achieves a data rate extremely close to the image entropy.

Contrarily to Wu's findings, Meyer and Tischer [1997, 1998] were able to design TMW, a lossless coding algorithm that, on the same standard test images, improves upon CALIC's compression. Even if TMW has a computational complexity several orders of magnitude greater than CALIC, the results were anyway surprising because TMW uses a two–pass global optimization and a blending of multiple linear predictors. In a TMW compressed file the encoded data are preceded by a header that contains the parameters of the model determined in the fist pass of the encoding.

The results achieved by TMW re–opened the question regarding the effectiveness of linear prediction in lossless image coding. In a following paper Wu, Barthel and Zhang [1998b] studied linear predictors optimized on a pixel–by–pixel basis and concluded that single–pass linear prediction can be used to achieve good compression in gray–level images. Also Meyer and Tischer [2001] have recently proposed a simple algorithm based on single–pass linear prediction that is able to improve upon CALIC's results.

Most literature dealing with lossless image compression focuses on continuous tones gray–level images. This is not a loss of generality since both color and color–mapped images can be preprocessed and then compresses as if they were gray–level images. In lossless compression, standard color transformations like YUV cannot be used because

of their lossy nature, so to compress color images, the color planes must be first decorrelated with the help of a lossless color transform and then each color component can be treated as a stand alone gray–level image. Papers that address more sophisticated approaches, like the ones by Wu, Wai–Kim Choi and Memon [1998] and Barequet and Feder [1999], show that the gain derived from a more sophisticated decorrelation is often marginal. Even color–mapped images (frequently called "*synthetic*" because most of them are computer generated), when not compressed by means of ad–hoc algorithms (Ausbeck [1998]), can be treated as gray–level continuous tone images after reordering and compacting the indices representing the colors. More details on the actual state of the art in lossless image coding can be found in Carpentieri, Weinberger and Seroussi [2000].

The compression of continuous tone gray–level images being general enough, we concentrate our attention on the compression of this particular data type. The images target of our investigation are discrete bi–dimensional signals, representation of natural scenes, in which an integer number measuring the brightness of the pixel is associated to a pair of spatial coordinates. Pixel values typically fall in the ranges [0 – 31], [0 – 255] or [0 – 4095], requiring respectively 4, 8 and 12 bits for the encoding of each pixel. While images that are 8 bit per pixel are common in literature, 12 or even 16 bit per pixel are the preferred choice when dealing with medical or scientific imagery.

The compression method that we propose and study here is based on a single step, adaptive linear prediction combined with a classification mechanism. The predictor is optimized on a pixel–by–pixel basis on a training set that is selected by the classifier in order to maximize the similarity of the samples with the context of the pixel being predicted. This method allows the combination of context–based coding and prediction.

Preliminary results on the application of this method were presented and discussed in Motta, Storer and Carpentieri [1999, 2000b]. This algorithm, that we called *ALPC*, acronym for *Adaptive Linear Prediction and Classification*, uses a set of linear predictors from which, for every pixel, one predictor is selected and refined with a an optimization procedure. The refinement uses samples taken from a causal window. Not all samples in the causal window are used to refine the predictor. The classification selects a fraction of samples that have a context (set of causal neighbors) closer to the context of the pixel being encoded. This classification limits the computational complexity and improves prediction. The size of the causal window used in the classification controls the locality of the adaptation. Large windows result in a more general predictor, while small windows follow more closely the local statistics of the image.

Given the backward adaptation and the lossless nature of the compression, the decoder is able to perform the same steps and derive an identical refined predictor. Finally, entropy coding is used to represent the prediction error efficiently.

Preliminary studies on this method were based on a Gradient Descend minimization while the best results were obtained with an improved version that optimizes the predictor with a Least Squares minimization.

Independently on the minimization procedure, the algorithm bases the prediction on a causal context centered on the current pixels. Missing context pixels encountered during the compression of the image borders are set to a default value.

The context consists of all the pixels that have a *Manhattan Distance* smaller then a given constant. Manhattan distance between two points p and p' having respectively coordinates (x, y) and (x', y') is defined as: $M(p, p') = |x - x'| + |y - y'|$.

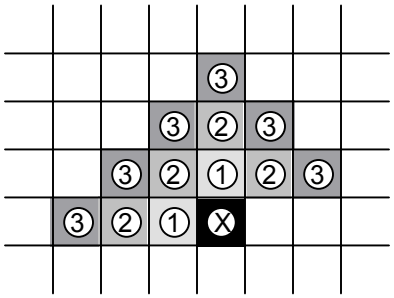


Figure 5.1: Context formed by pixels with Manhattan distance of three or less.

```
for every pixel p(x,y) in the input image do begin
      Collect the pixels in W_{x,y}(R_p) and their context
      Determine n centroids C̄_1,...,C̄_n by applying LBG on
            the contexts in W_{x,y}(R_p)
      Let K_1,...,K_n be the corresponding clusters
      Classify collected pixels with respect to the clusters
      Classify the context of the current pixel p(x,y);
            let k be the index of the cluster
      Let P̄_i be the predictor that achieves smallest error
            on K_k among the predictors P̄_1,...,P̄_n
      Refine the predictor P̄_i on the pixels in K_k
      Use the refined predictor P̄'_i to predict p(x,y)
      Encode the prediction error e(x,y) with an entropy encoder
end
```

Figure 5.2: Pseudocode description of the adaptive prediction.

A context that uses pixels having Manhattan distance of three or less from the current pixel is depicted in Figure 5.1. Distances of two or less and three or less achieve good compression and suffice to the characterization of the relation between a pixel and its context. Larger contexts increase the amount of computation while resulting in little or no improvement on the set of standard test images that we used.

In our work, we addressed the problem of gray–level lossless image compression exclusively from the point of view of the achievable compression ratio, without being concerned about computational complexity or memory requirements.

5.2 Adaptive Linear Prediction Coding

A pseudocode description of ALPC is given in Figure 5.2. The input image is traversed in raster scan order (top to bottom and left to right) and encoded in a single pass. The luminosity of each pixel $p(x, y)$ is predicted according to a weighted sum of its context. The result is rounded to the nearest integer. If a context of size six is used (pixels at a distance $M(p, p') \leq 2$ from p) the equation of the prediction $\tilde{p}(x, y)$ takes the form:

$$\begin{aligned}\tilde{p}(x, y) = round(&w_0 \cdot p(x, y-1) + w_1 \cdot p(x-1, y-1) + w_2 \cdot p(x, y-1) + \\ &+ w_3 \cdot p(x+1, y-1) + w_4 \cdot p(x-2, y) + w_5 \cdot p(x-1, y))\end{aligned}$$

Since the weights are adaptive, the order of the pixels in this equation is not relevant and we can rewrite the equation as:

$$\tilde{p}(x, y) = round\left(\sum_{p_i \ni' M(p, p_i) \leq 2} w_i \cdot p_i \right)$$

The context of $p(x, y)$ has a fixed shape and only the weights w_i are allowed to adapt during the encoding. After the prediction, the residual $e(x, y)$ of the pixel $p(x, y)$ is calculated by subtracting the prediction from the current pixel:

$$e(x, y) = p(x, y) - \tilde{p}(x, y)$$

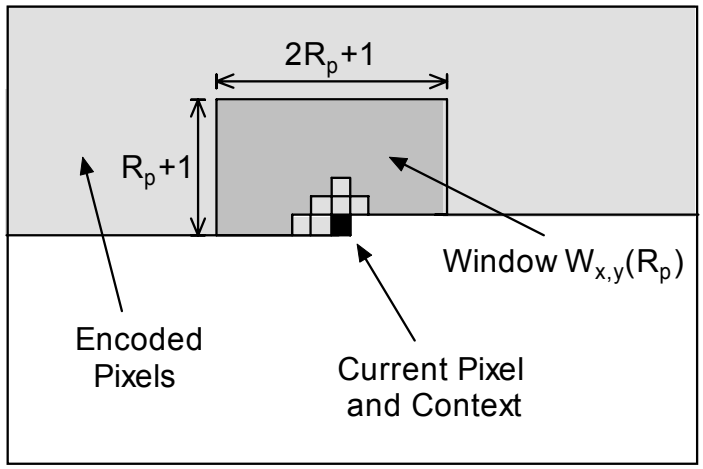


Figure 5.3: Prediction window $W_{x,y}(R_p)$ of radius R_p centered on $p(x,y)$.

If the pixels assume values in the range $[0,\ldots,\alpha-1]$ then the prediction error will be in the alphabet $[-\alpha+1,\ldots,\alpha-1]$. Since the prediction is based on previously encoded pixels, the decoder knows their exact value. This information can be used to reduce the range of the prediction error with a mapping called *Modulo Reduction*. Modulo reduction, described in Weinberger, Seroussi and Sapiro [1996], is commonly used in lossless encoders like, for example, JPEG–LS. It allows a better entropy coding and, when the error distribution is shaped like a Laplacian, this mapping doesn't change significantly the distribution. Modulo reduction has been used in the version of ALPC that refines the predictor with the least squares minimization and it is not necessary when modulo and sign of the prediction error are encoded separately.

Before encoding a pixel, the predictor's weights w_i are adaptively changed in order to model local characteristics of the image. The optimal predictor for the pixel being

encoded is defined as the one that minimizes the energy of the prediction error on all samples in the causal window $W_{x,y}(R_p)$ of radius R_p and centered on $p(x,y)$:

$$\min_{w_i}\left(E(x,y)\right) = \min_{w_i}\left(\sum_{p(x',y')\in W_{x,y}(R_p)} \left(e(x',y')\right)^2\right)$$

Adaptation uses a backward predictor. Backward prediction was preferred because it does not require any side information to be sent to the decoder. On the other hand, it also has as the well–known drawback of having poor performance in the presence of edges. This problem is partially mitigated by the use of the classification because classification selects in the window $W_{x,y}(R_p)$ a subset of pixels that have a context most similar to the context of the current pixel. The predictor is further refined only on these pixels.

The radius R_p of the window $W_{x,y}(R_p)$ (see Figure 5.3) is one of the essential features of ALPC. Its value affects the prediction quality because if R_p is too small, only a few samples are in the window and the predictor "overspecializes", making big errors in the presence of edges. On the other hand, too many samples in the window (R_p too big) tend to generate predictors that are not specific enough to remove local variations in the image. Our algorithm keeps R_p constant and equal for all the images, however it is possible to design algorithms that adaptively change the window radius to follow the characteristics of the image.

As anticipated, optimization only refines the predictor on a subset of the samples collected in the window. The rationale is that we want the predictors' weights to be representative of the relation existing between the context and the pixel being encoded. By discarding samples that have a context that is "too different" from the one of the

current pixel, we can specialize the prediction and follow fine periodic patterns in the window.

This approach is completely different from the ones commonly found in the literature that use a simple pixel predictor and compensate the poor prediction with a sophisticated entropy coding that models the error distribution according to the context in which it occurs (see for example LOCO–I, Weinberger, Seroussi and Sapiro [1996]).

ALPC's classification unifies contextual encoding and prediction into a single step. Samples are grouped into clusters of pixels with a similar context by using a modified Generalized Lloyd Algorithm (Linde, Buzo and Gray [1980]). This classification method, although not optimal in our framework, is able to improve the performance of the basic adaptive predictor.

Once the samples in the window are classified, a representative centroid is determined for each cluster. The cluster of pixels with the minimum squared error between the corresponding centroid and the context of the current pixel is selected. The algorithm then chooses from the pool of available predictors the one that achieves the lowest prediction error on the centroid of the selected cluster.

Before being used, this predictor is refined with a Gradient Descend optimization on the samples in the selected cluster. Gradient descend is a successive refinement algorithm that at every step changes the predictor weights until the prediction error drops below a given threshold. At each step t of the optimization process, the weights w_i of the predictor are changed according to:

$$w_i(t+1) = w_i(t) - \mu \cdot \frac{\partial E}{\partial w_i}$$

where E is the error energy and μ a small constant.

When the number of samples in the window is small, for example when $p(x, y)$ is close to the top or to the left border of the image, a default fixed predictor is used and the gradient descend optimization is not applied. Our implementation uses the classic "planar predictor" (Weinberger, Seroussi and Sapiro [1996]) as a default:

$$\tilde{p}_{def}(x, y) = p(x-1, y) - p(x-1, y-1) + p(x, y-1)$$

The predictor with the refined weights is stored and used in the next iterations. Because of the refining process, predictor initialization is not an issue and $P_1, ..., P_n$ can be initialized with random values without compromising the performance. Better initializations, which use for example the predictors in JPEG–LS, only result in a slightly faster convergence of the gradient descend. Reinitializing the predictors for every pixel, instead of using the previous refined weights, results in a slower convergence, and doesn't improve compression.

In the literature, (see, for example, Howard [1989]) it is commonly assumed that the error generated during the prediction of images can be closely approximated by a Laplacian distribution. Even in our case, adaptive linear prediction generates a skewed Laplacian–like distribution, centered on zero and with long tails.

Prediction error is encoded with an Arithmetic Encoder (Witten, Neal and Cleary [1987]). Arithmetic encoding divides the coding step into the determination of a probabilistic model for the source and in the entropy coding based on that model. This results in a very general framework allowing an easy adaptation of the model. ALPC determines the probability distribution of the error in a causal window centered on the current pixel and similar to the one we used in the prediction step.

During the experiments, we observed that 95% of the errors are concentrated in an interval $[-\Delta,\ldots,\Delta]$ substantially narrower than the error range $[Min,\ldots,Max]$. Typical values for Δ are in the range $[8,\ldots,20]$ while Min and Max span, in general, the range $[-\alpha+1,\ldots,\alpha-1]$. Since we determine the error distribution in a window, only a small number of samples are collected for each pixel and the distribution tails that have low probability, are not represented properly.

This problem, known as *Zero Probability* (see Witten [1991]) consists in the proper modeling of symbols that, because of their low probability of occurrence, have never been observed before. The solution that we used consists in dividing the errors in two sets: the "*typical*" errors that fall in the range $[-\Delta,\ldots,\Delta]$, and the "*non–typical*" errors, which fall outside of that range. A non–typical error is coded by using an alternative model and its code is preceded by a special "*escape*" symbol that is not in the alphabet. When the decoder receives the escape symbol, it switches to the alternative model to decode the non–typical error that follows in the bit stream.

The parameter Δ, as well as Min and Max are determined by an off–line observation of the errors. Sending these parameters to the decoder has a cost that is negligible from the point of view of the compressed file size.

Errors are also encoded by separating magnitude and sign. This method has the advantage of pooling together positive and negative magnitudes in the symmetrical distribution and provides better modeling when only a few samples are in the window. The error sign has no correlation with the error magnitude, and two different probabilistic models are used for the encoding.

Our implementation uses an arithmetic encoder that switches between four different models:

- *ACM_P* A model for the parameters. It is used only to transmit the header of the compressed file and the global parameters (Min, Max, Δ, R_e);
- *ACM_M* Used to encode the magnitude of the typical errors. It has symbols in the range $[0, \ldots, \Delta]$ plus an extra symbol $(\Delta + 1)$ that represents the escape;
- *ACM_E* An adaptive model used to encode the magnitude of the non–typical errors. It has symbols in the range $\left[\Delta + 1, \ldots, \max\left(|Min|, |Max|\right)\right]$;
- *ACM_S* Used to encode the error sign. It has the symbols $\{0,1\}$ used to represent positive and negative errors.

Unlike the other three models, ACM_M is not automatically updated and the probability distribution for the magnitude of the prediction error $e(x, y)$ is determined each time by collecting the previously encoded error magnitudes in a window $W_{x,y}(R_e)$ or radius R_e.

Since the error is distributed according to a Laplacian distribution, we have also experimented with a Golomb entropy encoder. The parameter that characterizes the encoder is determined in the causal window $W_{x,y}(R_e)$ as well.

The compression was assessed on set of continuous tones gray–level images digitized at a resolution of 8 bit per pixel (256 gray levels). The test set includes the nine, 720x576, images that compose the test set used in the JPEG–LS standardization. These and the other images that are widely used in the existing data compression literature allow a

comparison with existing algorithms. Thumbnails of the images used in the experiments are shown in Appendix A.

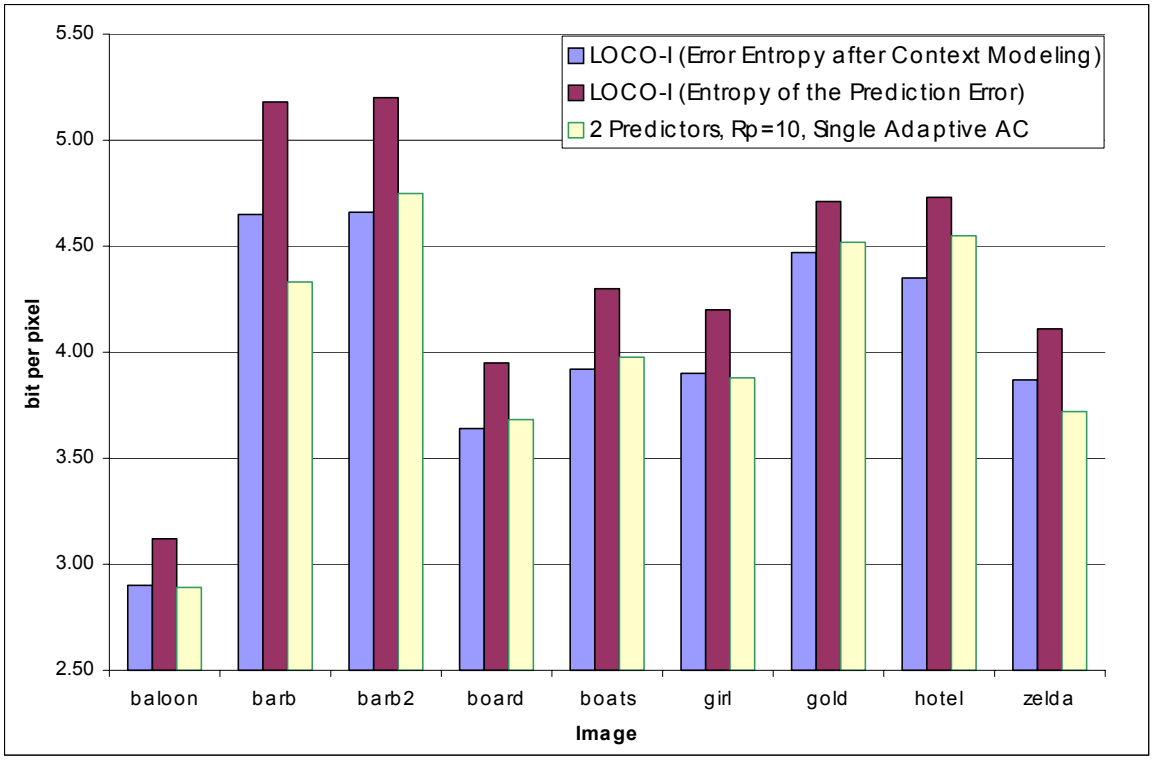


Figure 5.4: Comparisons with the entropy of the prediction error in LOCO–I.

Predictors	1	2	4	6	8
Baloon	154275	150407	150625	**150221**	150298
Barb	227631	**223936**	224767	225219	225912
Barb2	**250222**	250674	254582	256896	258557
Board	193059	**190022**	190504	190244	190597
Boats	210229	**208018**	209408	209536	210549
Girl	204001	**202004**	202326	202390	202605
Gold	**235682**	237375	238728	239413	240352
Hotel	**236037**	236916	239224	240000	240733
Zelda	195052	**193828**	194535	195172	195503
Tot.(bytes)	1906188	**1893180**	1904699	1909091	1915106

Table 5.1: Compressed File Size vs. Number of Predictors. Results shown for a window of radius $R_p = 6$, error is coded by using a single adaptive arithmetic encoder.

Rp	6	8	10	12	14
Baloon	150407	149923	**149858**	150019	150277
Barb	223936	**223507**	224552	225373	226136
Barb2	250674	249361	**246147**	247031	246265
Board	**190022**	190319	190911	191709	192509
Boats	208018	206630	**206147**	206214	206481
Girl	202004	201189	**201085**	201410	201728
Gold	237375	235329	234229	234048	**234034**
Hotel	236916	**235562**	235856	236182	236559
Zelda	193828	193041	**192840**	192911	193111
Tot. (bytes)	1893180	1884861	**1881625**	1884897	1887100

Table 5.2: Compressed File Size vs. window radius R_p. The number of predictors used is 2; prediction error is entropy encoded by using a single adaptive arithmetic encoder.

Re	6	8	10	12	14	16	18	20
Baloon	147518	**147227**	147235	147341	147479	147620	147780	147885
Barb	218411	216678	216082	**215906**	215961	216135	216370	216600
Barb2	237523	234714	233303	232696	232455	**232399**	232473	232637
Board	187058	186351	**186171**	186187	186303	186467	186646	186800
Boats	203837	202168	201585	**201446**	201504	201623	201775	201943
Girl	198050	197243	**197013**	197040	197143	197245	197356	197465
Gold	232617	230619	229706	229284	229111	229026	**229012**	229053
Hotel	231125	229259	228623	**228441**	228491	228627	228785	228949
Zelda	190311	189246	188798	188576	188489	**188461**	188469	188500
Tot.(bytes)	1846450	1833505	1828516	**1826917**	1826936	1827603	1828666	1829832

Table 5.3: Compressed File Size vs. error window radius R_e. The number of predictors is 2 and $R_p = 10$. Modulo and sign of the prediction error are encoded separately.

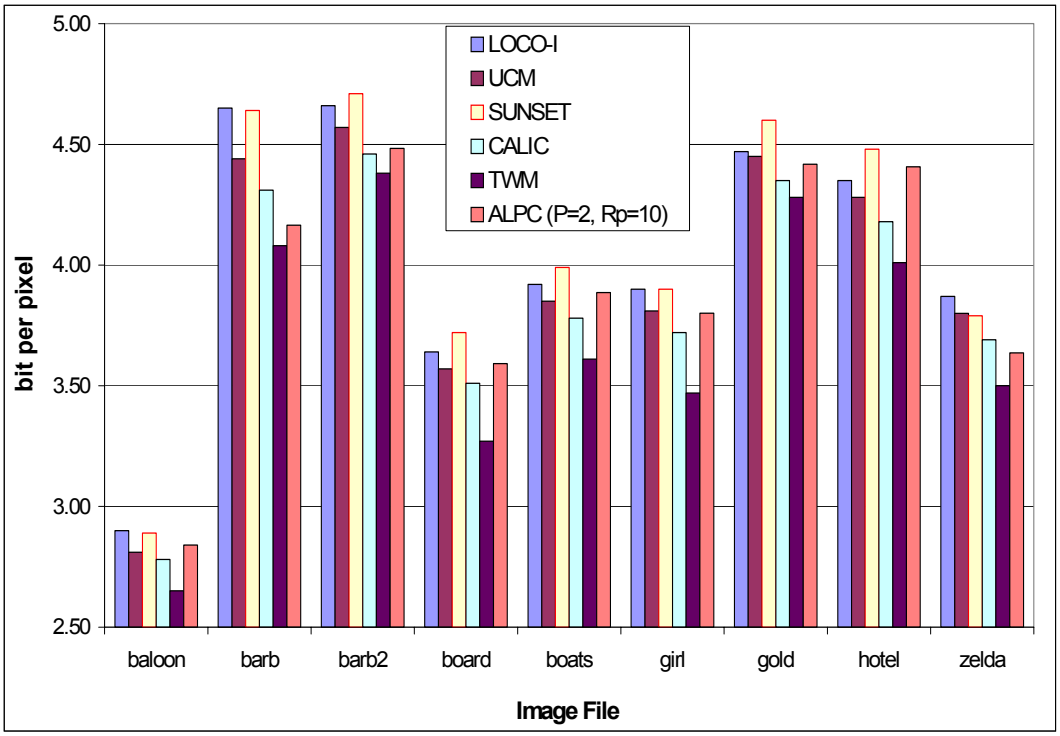


Figure 5.5: Graphical representation of the data in Table 5.5.

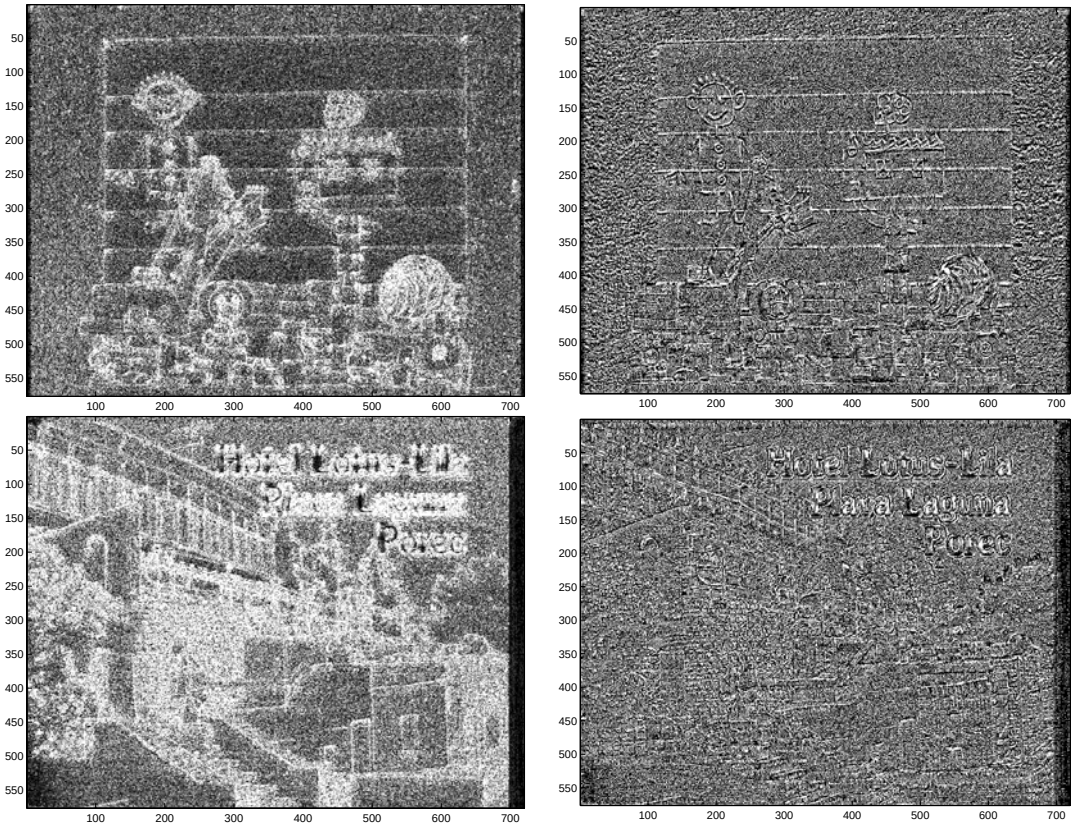


Figure 5.6: Magnitude (left column) and sign (right column) of the prediction error in two images of the Test Set. Images are "board" (top row) and "hotel" (bottom row).

	GR	AC	GR–W	AC–W
Airplane	3.80	3.81	3.66	3.62
Airport	6.65	6.65	6.69	6.60
Crowd	4.31	4.21	3.98	4.03
Goldhill	4.71	4.70	4.69	4.66
Hursley	4.79	4.77	4.54	4.55
Lake	5.07	5.11	4.97	4.93
Landsat	4.16	4.14	4.09	4.04
Lax	5.84	5.83	5.81	5.75
Lena	4.18	4.21	4.12	4.05
Lenna	4.05	4.09	3.97	3.91
Mandrill	5.76	5.79	5.73	5.65
Milkdrop	3.71	3.66	3.64	3.57
Mri	3.49	3.48	3.25	3.13
Mskull	2.78	2.57	2.62	2.26
Peppers	4.27	4.25	4.22	4.16
Woman1	4.65	4.64	4.52	4.49
Woman2	3.26	3.28	3.14	3.06
Xray	2.63	2.63	2.52	2.56
Average	*4.45*	*4.43*	*4.35*	*4.28*

Table 5.4: Comparison between four entropy coding methods: Golomb–Rice coding (GR), Arithmetic Coding (AC), Golomb–Rice with the model in a window $W_{x,y}(R_e)$ (GR–W), Arithmetic Coding with the model in a window $W_{x,y}(R_e)$ (AC–W). Results are shown in bit per pixel. Test images are 512x512 (except mri and xray that are 256x256), 8 bit/pixel.

	CALIC	LOCO	sunset	UCM	TMW	ALPC
Baloon	2.78	2.90	2.89	2.81	2.60	2.84
Barb	4.31	4.69	4.64	4.44	4.24	4.16
Barb2	4.46	4.69	4.71	4.57	3.83	4.48
Board	3.51	3.68	3.72	3.57	3.27	3.59
Boats	3.78	3.93	3.99	3.85	3.53	3.89
Girls	3.72	3.93	3.90	3.81	3.47	3.80
Gold	4.35	4.48	4.60	4.45	4.22	4.42
Hotel	4.18	4.38	4.48	4.28	4.01	4.41
Zelda	3.69	3.89	3.79	3.80	3.50	3.64
Average	*3.88*	*4.06*	*4.08*	*3.95*	*3.63*	*3.91*

Table 5.5: Compression results (in bit per pixel). Test images are 720x756, 8 bit/pixel. The number of predictors, refined with the gradient descend, is 2 and $R_p = 10$. Entropy encoding is performed with an arithmetic coder and the model is determined in a window of radius $R_e = 10$.

	CALIC	LOCO	ALPC
Airplane	3.54	3.61	3.62
Airport	6.55	6.71	6.60
Crowd	3.76	3.91	4.03
Goldhill	4.63	4.71	4.66
Hursley	4.39	4.43	4.55
Lake	4.90	4.98	4.93
Landsat	3.99	4.08	4.04
Lax	5.63	5.78	5.75
Lena	4.11	4.25	4.05
Lenna	3.94	4.07	3.91
Mandrill	5.74	5.89	5.65
Milkdrop	3.56	3.63	3.57
Mri	3.15	3.36	3.13
Mskull	2.16	2.23	2.26
Peppers	4.20	4.29	4.16
Woman1	4.54	4.67	4.49
Woman2	3.20	3.20	3.06
Xray	2.59	2.46	2.56
Average	*4.26*	*4.36*	*4.28*

Table 5.6: Final compression results (in bit per pixel). Test images are 512x512 (except mri and xray that are 256x256), 8 bit/pixel. ALPC uses two predictors optimized with the gradient descend and $R_p = 10$. Entropy encoding is performed with an arithmetic coder and the model is determined in a window of radius $R_e = 10$.

Figure 5.4 compares the entropy of the prediction error achieved by our adaptive predictor with the prediction error achieved by the fixed predictor used in LOCO–I. The results were obtained by using 2 predictors and by optimizing the predictors in a window of radius $R_p = 10$. For comparison also the overall performance of LOCO–I, after the context modeling is reported. Understandably our adaptive linear predictor is more powerful than the fixed predictor used in LOCO–I. However, as it is evident from Figure 5.6, adaptive prediction does not have enough power to capture edges and sharp transitions, present for example in the picture “hotel”.

Tables 5.1, 5.2 and 5.3 summarize the experiments we made in order to understand the sensitivity of the algorithm to its parameters. In these experiments, we measured the variations on the compressed file size when only one of the parameters changes.

In Table 5.1, the number of predictor is changed while keeping the window radius $R_p = 6$. In Table 5.2, compression performance is evaluated with respect to the changes in the window size.

Experiments described in Tables 5.1 and 5.2, were performed using a simple adaptive arithmetic coder that gives a close approximation of the first order entropy of the prediction error.

Table 5.3 reports experiments when the number of predictors is kept fixed to 2, $R_p = 10$ and the performance is evaluated encoding the prediction error with the multiple model arithmetic encoder described before. Results are reported for value of R_e, (size of the window in which the model is determined), varying between 6 and 20. Table 5.4 reports experiments made by replacing arithmetic coding with computationally less intensive Golomb coding.

Comparisons with some popular lossless image codecs are reported in Tables 5.5 and 5.6 and Figure 5.5. Results show that ALPC achieves good performance on the majority of the images in the test set. The cases in which CALIC maintains its superiority confirm that linear prediction, may not be adequate to model image edginess. On the other hand, unlike CALIC, ALPC doesn't use any special mode to encode high contrast image zones, and our results are slightly penalized by images like “hotel” that have high contrast regions. A closer look to the magnitude and sign of the prediction error for “board” and

"hotel" (two images in the test set), shows that most edges in the original image are still present in the prediction error (see Figure 5.6).

5.3 Least Squares Minimization

Optimizing the predictor with the gradient descend has the disadvantage of having slow, image dependent, convergence. In their papers, Meyer and Tischer [2001], Wu, Barthel and Zhang [1998] proposed prediction algorithms based on least square optimization. Unlike in speech coding, when predicting images it is not possible to use Levinson–Durbin recursion to solve the system of equations to derive the predictor, so least squares minimization is a simple and viable alternative.

To find the best predictor for the pixel $p(x, y)$ we start selecting and collecting from a causal window $W_{x,y}(R_p)$ a set of pixels p_i and their contexts $\overline{c}_i$. The order in which the context pixels are arranged in the column vector $\overline{c}_i$ is not important as long as it is consistent for every vector. A matrix $\overline{A}_i$ and a vector $\overline{b}_i$ are computed as:

$$\overline{A}_i = \overline{c}_i \cdot \overline{c}_i^T$$

$$\overline{b}_i = p_i \cdot \overline{c}_i .$$

Finally, matrices $\overline{A}_i$ and the vectors $\overline{b}_i$ are added together to form $\overline{A}_{x,y}$ and $\overline{b}_{x,y}$ as in:

$$\overline{A}_{x,y} = \sum_i \overline{A}_i$$

$$\overline{b}_{x,y} = \sum_i \overline{b}_i$$

The predictor's weights $\overline{w}_{x,y}$ that minimize the expectation of the squared errors are obtained by solving the system of equations:

$$\overline{A}_{x,y} = \overline{w}_{x,y} \cdot \overline{b}_{x,y}$$

A substantial speed up can be achieved by pre–computing the matrices $\overline{A}_i$ associated with every pixel.

For the solution of the system of the equations $\overline{A}_{x,y} = \overline{w}_{x,y} \cdot \overline{b}_{x,y}$ we have used the *Gnu Scientific Library*, a standard library available on UNIX platforms under the GNU license.

When only a small number of samples are present in the window, the predictor overspecializes and its weights assume values that can be very large. It is also possible that the matrix $\overline{A}_{x,y}$ is singular. In these cases, the default predictor is used instead.

Substituting in ALPC gradient descend with least squares minimization results in a much faster procedure. This allows, in turn, to experiment with bigger windows. In order to improve ALPC compression we have also experimented with different classification methods. Since the contexts that we want to use for the predictor computation must be similar to the current context, we select a fraction of contexts in the window $W_{x,y}(R_p)$ that have smallest Euclidean distance from the current context.

When calculating the Euclidean distance, it is possible to give different importance to the pixels in the context by weighing appropriately the corresponding difference. The set of weights that gave better results are directly proportional to the existing correlation between the pixel being encoded and the pixel in the context.

bpp	TMW	Glicbawls	CALIC	ALPC G.Descend	ALPC L.Squares
AIRPLANE			3.54	3.62	3.602
Airport			6.55	6.60	6.632
Ballon	2.60	2.64	2.78	2.84	2.748
Barb2	4.24	4.31	4.46	4.48	4.417
Barb	3.83	3.92	4.31	4.16	4.007
Board	3.27	3.39	3.51	3.59	3.484
Boats	3.53	3.63	3.78	3.89	3.775
Crowd			3.76	4.03	3.971
Girl	3.47	3.59	3.72	3.80	3.688
Gold	4.22	4.28	4.35	4.42	4.392
Goldhill			4.63	4.66	4.635
Hotel	4.01	4.18	4.18	4.41	4.336
Hursleyhouse			4.39	4.55	4.548
Lake			4.90	4.93	4.876
Landsat			3.99	4.04	4.039
Lax			5.63	5.75	5.780
Lena			4.11	4.05	4.030
Lenna			3.94	3.91	3.880
Mandrill			5.74	5.65	5.683
Milkdrop			3.56	3.57	3.560
Mri			3.15	3.13	2.826
Mskull			2.16	2.26	2.254
Peppers			4.20	4.16	4.161
Woman1			4.54	4.49	4.508
Woman2			3.20	3.06	3.032
Xray			2.59	2.56	2.452
Zelda	3.50	3.54	3.69	3.64	3.603
TOTAL	**3.630**	**3.720**	**4.206**	**4.240**	**4.189**

Table 5.7: Final compression results (in bit per pixel). The increment in performance observed in ALPC with the least squares minimization is mostly due to the possibility of using a context of bigger size.

Table 5.7 summarizes the results obtained by the improved algorithm on the set of test images. As it is possible to see, when using least squares and the new classification, our algorithm improves upon CALIC on most images in the test set.

This suggests that, since ALPC combines linear prediction and classification, with a more sophisticated encoding of the prediction error, our algorithm can be even more competitive and finally achieve compression ratio even comparable to TMW.

LOW BIT RATE VIDEO CODING

6.1 Background

Current state of the art video compression systems such as H.261, H.263, MPEG–1 and MPEG–2 are hybrid encoders combining motion compensated prediction with block based discrete cosine transform (DCT) coding. Each video frame is first partitioned into macroblocks. Then each macroblock is encoded as a motion compensated difference with respect to a previously encoded macroblock (*Inter frame* coding or *P–mode*) or by direct quantization of its discrete cosine transform coefficients (*Intra coding* or *I–mode*). While matching a preset bit rate, the encoder tries to maximize the peak signal to noise ratio (PSNR) of the encoded video sequence.

Since inter and intra coding of the same macroblock result in different rate/distortion performance, most encoders use an empirical method to decide whether a macroblock should be inter or intra coded. A common heuristic compares the absolute value of the motion compensated prediction error with a fixed threshold; when the prediction error is below the threshold, motion compensated prediction is chosen over intra coding. Unfortunately, due to the high variability of the video signal, it may be hard not to exceed the target bit rate. An output buffer amortizes small discontinuities in the final bit rate however, when the buffer capacity is exceeded, video encoders are forced to skip frames

in order to match the required rate without compromising the quality of the individual frames.

In current video encoding standards, a two-layer rate control strategy keeps the coding rate as close as possible to a preset target. The *macroblock layer* operates at the lower level. Given the number of bits available for the coding of the current frame and some statistics on the most recently encoded macroblocks, it decides how to allocate the available bits to the macroblocks in the current frame. Depending on the frame complexity, the encoding may use more bits than the bits originally allocated, so a higher–level *frame layer* rate control monitors the output buffer and decides whether to skip one or more frames to allocate time for the transmission of the buffer content. Both layers have been widely studied and optimization issues relative to rate control algorithms have been considered by a number of authors.

Kozen, Minsky and Smith [1998] use a linear programming approach to determine the optimal temporal down sampling in an MPEG encoded sequence. Their algorithm discards a fixed number of frames while minimizing the interval of non–playable frames due to frame dependency. Wiegand, Lightstone, George Campbell and Mitra [1995] present a dynamic programming algorithm that jointly optimizes frame type selection and quantization steps in the framework of the H.263 video coding. In their work, optimization is performed on each macroblock on a frame–by–frame basis. Experimental results show consistent improvement upon existing methods.

Lee and Dickinson [1994] address a similar problem in the framework of MPEG encoding, where each group of frames is isolated and both frame type selection and quantization steps are jointly optimized with a combination of dynamic programming and

Lagrange optimization. Unfortunately, their experiments show very little improvement, probably because they use a simplified approach that restricts the number of possible quantization steps to a very small set of values.

Sullivan and Wiegand [1998] and Wiegand and Andrews [1998] present a macroblock rate control that is based on rate–distortion optimization. Their approach constitutes the basis for the H.263+ Test Model Near–Term Version 11 (or TMN–11, see Wenger at al. [1999]). The optimization, based on Lagrange multipliers, shows consistent improvements even when the multiplier is kept constant in order to reduce the computational complexity.

In the following we address the problem of designing a frame layer control algorithm that can be used to minimize the number of skipped frames in a low bit rate video sequence. We assume that the video sequence is encoded at constant bit rate by any of the standard video encoders, like the ones in the MPEG family. As we will see, the reduction of skipped frames allows more frames to be transmitted without increasing the bit rate and, surprisingly, without compromising the per-frame average final SNR. The visual quality also improves because the jerkiness associated to the skips is reduced as well.

6.2 Frame and Macroblock Layer Rate Controls

As mentioned in the previous section, rate control is generally organized into two levels or layers: frame and macroblock. The frame layer works at the highest level. A transmission buffer is monitored after the transmission of each frame. When this buffer

exceeds a preset threshold because the encoding of the last frames has used more bits than the bits that were originally allocated, the encoding of one or more video frames are skipped while the encoder waits for the buffer to empty.

An example of such strategy is the frame layer rate control described in the Test Model Near–Term Version 8 (see Gardos [1997]). This algorithm is characterized by the following parameters:

- B', the number of bits occupied by the previous encoded frame;
- R, the target bit–rate in bits per second;
- F, the number of frames per second;
- M, the threshold for frame skipping, typically $M = R/F$ (M/R is the maximum buffer delay);
- A, a constant, usually set to $A = 0.1$. Used to define the target buffer delay to be $A \cdot M$ seconds.

After encoding a frame, the number of bits in the buffer W is updated as $W = \max(0,\ W + B' - M)$ and the number of frames that must be skipped after that is computed as:

```
Skip=1
While(W > M ) {
        W = max(0,W − M)
        Skip++
}
```

Then the control is transferred to the macroblock layer that decides the number of bits that should be allocated to the next frame and divides these bits among the individual macroblocks.

The target number of bits is computed as $B = M - \Delta$, where

$$\Delta = \begin{cases} \dfrac{W}{F} & \text{if} \quad W > A \cdot M \\ W - A \cdot M & \text{otherwise.} \end{cases}$$

The target bit number B is distributed among the macroblocks of the current frame with an adaptive algorithm that, monitoring the variance of the macroblocks, determines the quantization step that is likely to achieve the target bit rate. The statistics are updated after the encoding of each macroblock. Encoding of the first macroblock in the frame uses model parameters of the last encoded frame. A detailed description of this algorithm can be found in Gardos [1995].

6.3 Problem Description

Our interest is in minimizing the number of skipped frames in a low bit rate video sequence encoded at constant bit rate by any of the standard hybrid video encoders, like for example, the standards belonging to the MPEG family.

Frame skipping generally happens in proximity of a scene change. When a new scene starts, motion compensated prediction is unable to predict the first frame of the new scene from the past and the channel capacity is easily exceeded. In the following, we consider a scene change to be when the prediction model fails, because the frame is completely

different from the previous one or because it may simply contain too much movement to be efficiently predicted.

A scene change causes most macroblocks to be intra coded, and this results in a frame encoded with more bits than the target bit rate. In turn, transmitting a frame exceptionally large requires a time higher than the time available for a single frame. In order to match the capacity of the channel and to keep the sequence time-synchronized, the encoder is forced to wait for its transmission and skip the encoding of a few subsequent frames.

The increment in rate observed during a scene change is illustrated in Figure 6.1, which shows the bit rate for the 900 frames contained in the test sequence Std100.qcif used in our experiments. The scene changes in this sequence were artificially generated by concatenating scenes extracted from a number of standard test sequences.

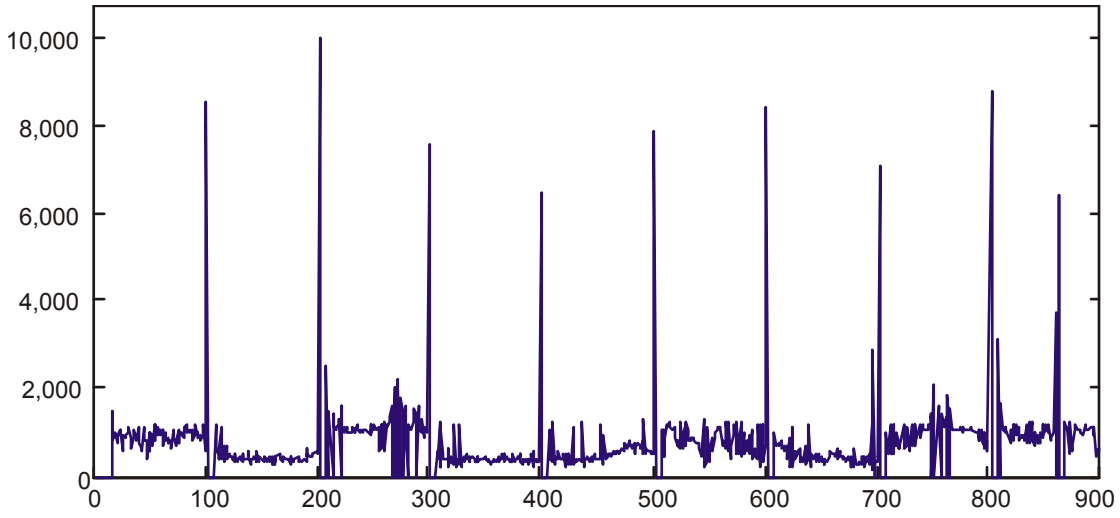


Figure 6.1: Bit rate for Frames 1 to 900 of the sequence Std100.qcif. Frames are numbered starting at 0; Frame 0, which is not shown, goes to about 17,500.

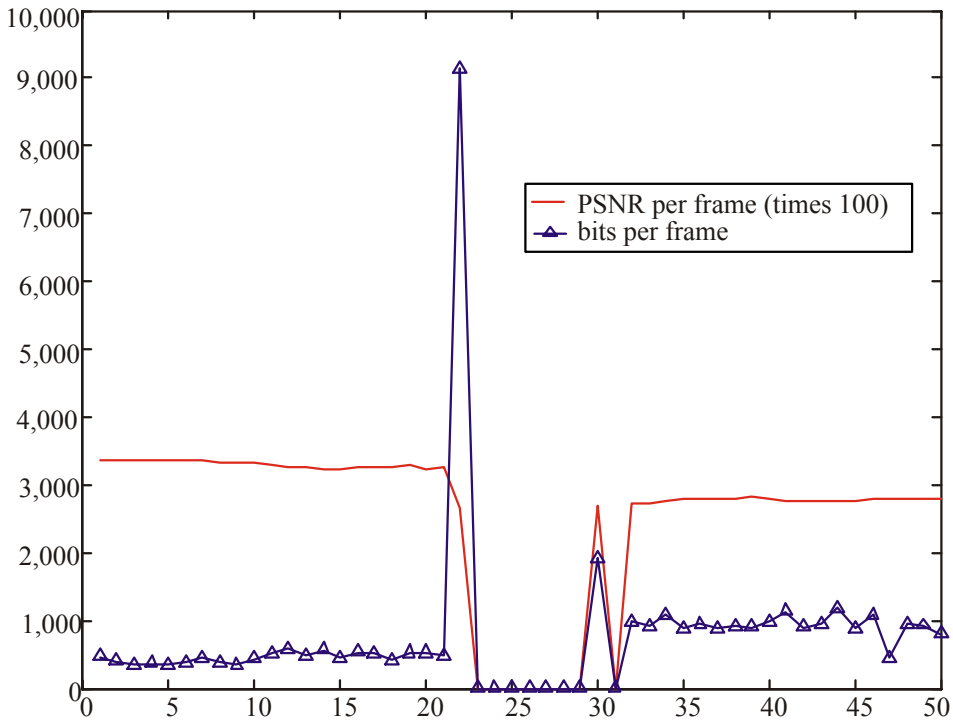


Figure 6.2: Sequence of 50 frames across a scene cut in one of the files used in our experiments (Claire.qcif 80–100 followed by Carphone.qcif 1–29) encoded at 32 Kbit/s with TMN–8 rate control; the bits per frame and the corresponding PSNR per frame are shown.

Frame No.	**n-5**	**n-4**	**n-3**	**n-2**	**n-1**	**n**	**n+1**	**n+2**	**n+3**	**n+4**	**N+5**	**n+6**	**n+7**	**…**
Encode	n-5	n-4	n-3	n-2	n-1	n	*skip*			n+4	*skip*	n+6	n+7	…
Transmit	n-5	n-4	n-3	n-2	n-1	n				n+4		n+6	n+7	...
Display	...	n-5	n-4	n-3	n-2	n-1				N		n+4	n+6	n+7

Figure 6.3: Encoding, transmitting and decoding/displaying a sequence of frames with a H.263+ encoder using TMN–8 rate control. The sequence contains a scene cut between frames $n-1$ and n.

Except for the first frame of the sequence, which is always encoded in intra mode (its size is not depicted in Figure 6.1), frames are encoded by using the inter mode in which individual macroblocks can be inter or intra coded.

The sequence has been encoded with the public domain Telnor/UBC H.263+ encoder (Cote, Erol, Gallant and Kossentini [1998]) that uses the rate control suggested by the

Test Model Near–Term Version 8 (or in short TMN–8) and described in Gardos [1997]. There is no loss of generality in experimenting with TMN–8 based encoder since even more recent test models, like TMN–10 and TMN–11 that use a rate optimized bit allocation, exhibit the same kind of behavior in the presence of scene changes. For a detailed description of the test models, see Gardos [1997, 1998] and Wenger et al. [1999].

Figures 6.2 and 6.3 provide a closer look at the bit-rates during the encoding of a scene change with H.263+. If we focus our attention around the scene change (frame n in Figure 6.3) we note that, because of the number of I–macroblocks, this frame takes a considerable time to be transmitted (3 additional time slots in our example). Meanwhile, while waiting for a complete reception of frame n, the decoder keeps showing frame $n-1$ on the decoder screen. In order to match the channel bit rate and to maintain synchronization with the original sequence, the encoder is forced to skip a number of frames while waiting for the transmission of frame n, and, because of this skipping, the next frame to be encoded will be frame $n+4$: in this example, transmission of frame n requires skipping 3 frames.

In general, after a scene cut, there will be some number $k \geq 0$ such that:

1. There are k extra units of time in which frame $n-1$ is frozen on the screen ($k = 3$ in Figure 6.3);
2. There is a "jerk" between frame n and frame $n+k+1$;
3. Since frames n and $n+k+1$ are not contiguous in time and frame $n+k+1$ is predicted from frame n, it is likely that a large prediction error generates a frame $n+k+1$ that is too big to be sent in one unit of time (frame $n+4$ in Figure 6.3).

This "chain effect" may force the encoder to skip frame $n+k+2$ too, before encoding frame $n+k+3$.

By maximizing the number of transmitted frames (or equivalently by minimizing the number of skipped frames) we reduce the coding artifacts described in points 2 and 3. In doing this, we want to perform the best possible selection of the first frame that should be encoded immediately after the termination of a scene (frame $n-1$ in Figure 6.3). Of course this is only possible if the encoder has some look–ahead capability and the encoding is not done in real–time. Since the decoder is not aware that this optimization has taken place, the decoding process remains completely unaffected and full compatibility with standard decoders is maintained. This is all that matters for many interesting and useful applications, like for example video distribution, where a powerful encoder creates off–line a video stream that is broadcasted to many real time decoders.

6.4 Optimal Frame Skipping Minimization

We address the problem of designing an optimal frame layer rate control algorithm that minimizes the number of frames skipped in a constant, low bit–rate video encoder. Minimizing the number of skipped frames improves the overall quality of the encoded video and reduces the jerkiness associated to the skips. The problem is to maximize the number of video frames that can be sent on a finite capacity channel in a given amount of time. Each frame, except the first, is predicted from the previous encoded frame, called its "anchor" frame. Frames are encoded and transmitted in strict temporal order. A buffer

will store the portion of encoded stream that is waiting to be transmitted over the fixed capacity channel. When the buffer is almost full (i.e. it holds a number of bits greater than a fixed threshold), new frames cannot be encoded until these bits are transmitted, and some of the frames must be skipped.

The solution we propose is a dynamic programming algorithm that can be used in low–bandwidth applications in which the encoder has some look–ahead capability. For a given bit rate, its asymptotic time complexity is linear in the number of frames being encoded. Although this optimization requires additional encoding complexity, there is no change in decoding complexity (in fact, no change to the decoder at all). Possible areas that can benefit of this algorithm are video broadcasting, off–line video coding, wireless video transmission, video distribution over IP, etc.

Definition 6.1: An instance of this optimization problem, that we indicate with the name MAX_TRANS, is given by:

- n : which represents the number of frames in the video sequence $f_0, f_1, \ldots, f_{n-1}$;
- M : the channel capacity, expressed in bits per frame;
- B_0 : the number of bits contained in the coding buffer before encoding the first frame of the sequence. We use B_i to indicate the buffer content before the encoding of the frame f_i;
- A_0 : which represents the "distance" to the most recent (encoded) anchor frame. In general A_i is the distance to the anchor frame of f_i. $A_0 = 0$ if no previous frame has been encoded;

- $C[i][A_i]$: the cost function which counts the number of bits necessary to the transmission of the frame f_i when f_{i-A_i} is the most recent encoded frame preceding f_i (anchor frame).

A solution to this problem takes the form of a sequence of n binary decisions $d_0, d_1, \ldots, d_{n-1}$ where:

$$d_i = \begin{cases} 1 & \text{if the frame } f_i \text{ is transmitted} \\ 0 & \text{if the frame } f_i \text{ is skipped} \end{cases}$$

The goal is the maximization of the number of transmitted frames, or equivalently:

$$Maximize \quad D = \sum_{i=0}^{n-1} d_i$$

While satisfying the capacity constraint:

$$B_0 + \sum_{i=0}^{n-1} d_i \cdot C[i][A_i] \leq n \cdot M$$

The maximization is also subject to the following two conditions:

1. If at the time i the buffer B_i holds a number of bits greater than or equal to M then the frame f_i cannot be transmitted:

$$\forall\ i = 0, \ldots, n-1 \quad B_i \geq M \Rightarrow d_i = 0\,.$$

2. If the frame i is transmitted, being predicted from its anchor frame f_{i-A_i}, then:

$$M \cdot A_i + B_i \geq B_{i-A_i} + C[i - A_i][A_{i-A_i}] + C[i][A_i]\,.$$

This condition states that the $C[i][A_i]$ bits necessary to encode the frame f_i are transmitted, M bits at the time, starting from the time $i - A_i + 1$ when the buffer has a content of $B_{i-A_i} + C[i - A_i][A_{i-A_i}]$ bits. The bits that are not transmitted are stored in the buffer whose content, at time i, is B_i. The inequality holds because the buffer, decremented by M units for each frame, cannot assume negative values.

In order to solve this problem with a dynamic programming algorithm, it is first necessary to verify that the problem has an optimal substructure (see Cormen, Leiserson and Rivest [1990]), i.e. that an optimal solution of a problem instance contains optimal solutions for embedded instances.

***Theorem 6.1 (Optimal Substructure)*:** Let $S = d_0, d_1, \ldots, d_{n-1}$ be a sequence of decisions with cost $D = \sum_{i=0}^{n-1} d_i$ and such that $B_0 + \sum_{i=0}^{n-1} d_i \cdot C[i][A_i] \leq n \cdot M$. If $S = d_0, d_1, \ldots, d_{n-1}$ is an optimal solution of the problem instance $I = (n, M, A_0, B_0, C)$, then for every integer $k = 0, \ldots, n-1$, the sequence of decisions $S_k = d_k, d_{k+1}, \ldots, d_{n-1}$, having cost $D_k = \sum_{i=k}^{n-1} d_i$, is an optimal solution of the problem instance $I_k = (n-k, M, A_k, B_k, C)$.

***Proof*:** The theorem is proved by contradiction. Let's suppose that the problem instance $I_k = (n-k, M, A_k, B_k, C)$ admits a solution $S'_k = d'_k, d'_{k+1}, \ldots, d'_{n-1}$ having cost $D'_k = \sum_{i=k}^{n-1} d'_i$ greater than D_k and satisfying $B_k + \sum_{i=k}^{n-1} d'_i \cdot C[i][A_i] \leq (n-k) \cdot M$.

The solution obtained by concatenating the first $k-1$ decisions in S with the $n-k$ decisions in S'_k is a solution of the problem instance $I = (n, M, A_0, B_0, C)$ that has a cost:

$$D'' = \sum_{i=0}^{k-1} d_i + \sum_{i=k}^{n-1} d'_i = D - D_k + D'_k > D$$

and satisfies

$$B_0 + \sum_{i=0}^{k-1} d_i \cdot C[i][A_i] + B_k + \sum_{i=k}^{n-1} d'_i \cdot C[i][A_i] \leq k \cdot M + (n-k) \cdot M = n \cdot M .$$

This contradicts the assumption that $S = d_0, d_1, \ldots, d_{n-1}$ is an optimal solution for the problem instance I. Therefore, the thesis holds.

The second condition that must hold in order to apply dynamic programming is the presence of overlapping subproblems. It is possible to verify that this condition is satisfied by our problem since the encoding of a subsequence of frames depends only on the anchor frame and on the initial buffer content B_0. In synthesis, every solution of the instance $I = (n, M, A_0, B_0, C)$ that at time k has anchor frame f_{k-A_k} and a buffer content of B_k, shares the subproblem $I_k = (n-k, M, A_k, B_k, C)$.

Given the previous considerations, it is possible to solve an instance I of the problem with a dynamic programming approach that, going from frame f_0 to frame f_{n-1}, determines the sequence of decisions $d_0, d_1, \ldots, d_{n-1}$ by tabulating intermediate results for different anchor frames and for the corresponding buffer contents. At time k, for every anchor frame f_{k-A_k} and buffer content B_k, our algorithm extends the optimal solution of the subproblem $I_k = (n-k, M, A_k, B_k, C)$ which is assumed to be known.

Figure 6.4 describes the algorithm in C-style pseudocode. The function performing the optimization is `DP_optimize`, while `Find_Solution` is called after it and determines, from the pointer matrix, the optimal sequence of decisions D.

```
DP_optimize(C, n, A, M) {
    for(i=0; i<n; i++) {
        for(j=0; (j<=A) and (j<=i); j++) {
            if(j==0) {        // I frame - only used to start sequence
                 T[i][j] = 1
                 B[i][j] = MAX(0, C[i][0] - M * (i+1))
                 P[i][j] = -1
            } else {          // P frame
                 prev = i-j
                 max_T = max_P = -1
                 max_B = +Inf
                 for(k=0; (k<=A) and (k<=prev); k++) {
                      res = MAX(0, B[prev][k] - M * (j-1))
                      if ((res<M) and (T[prev][k]>0) and
                          ((T[prev][k]>max_T) or
                           ((T[prev][k]==max_T) and (res<max_B)))) {
                           max_T = T[prev][k]
                           max_B = res
                           max_P = k
                      }
                 }
                 if(max_P >= 0) {
                      T[i][j] = max_T + 1
                      res = MAX(0, B[prev][max_P] - M * (j-1))
                      B[i][j] = MAX(0, res + C[i][j] - M)
                      P[i][j] = max_P
                 }
            }
        }
    }
}


Find_solution(T, P, n, A) {
    for(i=n-A; i<n; i++) {
        for(j=0; (j<=A) and (j<=i); j++)
            if(T[i][j] > max) {
                max = T[i][j]
                max_i = i
                max_j = j
            }
    }
    i = max_i
    j = max_j
    //      Back to the beginning...
    D[i] = 1
    do {
        old_i = i
        i -= MAX(1, j)
        j = P[old_i][j]
        D[i] = 1
    } while(P[i][j] != -1)
}
```

Figure 6.4: Pseudo code description of the dynamic programming optimization algorithm.

Function parameters are the number of frames n, the cost matrix C, the value $A = \max_{i=0,\ldots,n-1} A_i$ and the number of bits per frame M. The value A, which depends on the transmission rate, is computed from the cost matrix as $A = \left\lfloor \max_{0 \le i, j \le n-1} C[i][j] / M \right\rfloor$ and represents the maximum number of frames that can be skipped during the transmission. The algorithm stores solutions to subproblems into three matrices T, B and P, filled columnwise, from left to right. For every pair (k, A_k) the matrices store the solution of a subproblem in which:

- $T[k][A_k]$ is the number of transmitted frames;
- $B[k][A_k]$ is the corresponding buffer content;
- $P[k][A_k]$ is a pointer to the row in which the column $k - A_k$ achieves the highest number of transmitted frames. This pointer is necessary since there may be more than one row of T with the same number of transmitted frames.

Finally, the matrix P is used by the helper function `Find_solution` to determine the optimal sequence of decisions D.

Figure 6.6 shows a sample execution of the two functions on a random cost matrix C. For simplicity, this example has $n = 15$ frames, $M = 10$ bits per frame and the maximum number of skipped frames is $A = 5$. Shaded entries correspond to the optimal decisions taken by the dynamic program.

0	0	0	0	0	32	28	43	38	31	35	40	41	45	44
0	0	0	0	32	30	25	33	34	27	29	34	32	40	38
0	0	0	24	30	41	24	32	28	21	25	31	28	34	35
0	0	23	17	23	32	19	28	23	18	21	26	24	28	31
0	22	19	21	18	20	15	22	18	13	18	20	16	16	15
40	30	30	44	25	15	30	50	29	36	58	50	40	34	27

Cost Matrix C[ ][ ]

0	0	0	0	0	22	18	33	28	21	25	30	31	35	34
0	0	0	0	22	20	15	23	24	17	19	24	22	30	28
0	0	0	0	20	31	14	22	20	11	17	21	18	24	25
0	0	0	7	13	22	14	18	17	15	18	17	22	25	29
0	0	0	11	15	10	5	17	8	11	8	18	6	12	5
30	10	0	4	0	0	0	0	0	0	0	0	0	0	0

Buffer Matrix B[ ][ ]

0	0	0	0	0	2	2	2	3	4	4	5	5	6	6
0	0	0	0	2	2	2	3	4	4	5	5	6	6	7
0	0	0	0	2	2	3	4	4	5	5	6	6	7	7
0	0	0	2	2	3	4	3	5	4	6	6	7	7	7
0	0	0	2	3	2	2	3	2	3	2	3	2	3	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Number of Transmitted Frames T[ ][ ]

0	0	0	0	0	0	0	0	1	1	2	2	3	2	3
0	0	0	0	0	0	0	1	1	2	2	3	2	3	2
0	0	0	0	0	0	1	1	2	2	3	2	3	2	2
0	0	0	0	0	2	1	1	2	1	2	3	2	2	3
0	0	0	0	2	0	0	1	0	1	0	1	0	1	0
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Pointers Matrix P[ ][ ]

0	1	0	1	1	0	1	0	1	0	1	0	1	0	0

Decision Vector D[ ]

Figure 6.6: Sample execution of the dynamic programming optimization on a random cost matrix C. Algorithm parameters are the number of frames $n = 15$, the number of bits per frame $M = 10$ and the maximum number of skipped frames $A = 5$.

When applied to the video sequences used in our experiments, typical parameters for a 32 Kb/s encoding at 30 fps are $3100 \le n \le 5800$, $M = 32768/30$ and $40 \le A \le 60$.

Given the cost matrix, the time complexity of `DP_optimize` is $\mathrm{O}(n \cdot A^2)$. The value $A = \max_{i=0,\ldots,n-1} A_i$ is a constant computed in linear time from the first row of the cost

matrix and it doesn't depend on the number of frames. `Find_solution` examines $O(A^2)$ matrix entries to locate the maximum value in the last A columns of T then it performs $O(n)$ operations to construct the sequence of decisions. So, the asymptotic complexity of this algorithm grows linearly with the number of frames n.

6.5 Experimental Results with the Optimal Algorithm

In solving the optimization problem with the dynamic programming approach, we have assumed that the cost function C, expressed in the form of a bi–dimensional matrix, is known. We have also assumed that the cost of predicting f_k from its anchor frame f_{k-A_k}, namely $C[k][A_k]$, is independent on how f_{k-A_k} was compressed (so long as the quality of f_{k-A_k} is equal to a predefined target quality).

Although reasonable, the tightness of these approximations depends on the macroblock rate control. If we embed the proposed frame rate control in an existing video encoder, the way the macroblock level rate control is implemented may determine differences in the resulting costs. In the TMN–8, for example, the bit pool available for the current frame is allocated to each macroblock in a strict sequential fashion. A variable representing the variance of the previously encoded macroblocks determines the amount of bits that can be allocated. Since the value of this variable is preserved from one frame to the successive, the exact cost of encoding a frame may be influenced by the coding history and not only by its anchor frame.

To overcome this problem, without replacing the macroblock rate control used by the H.263+ encoder, we have used an iterative algorithm to compute the cost matrix C. This iterative algorithm starts from an empty matrix C having all entries equal to zero, then it applies the dynamic program to the matrix and encodes the sequence by following as close as possible the decisions. Due to the unfeasibility of this solution (which assumes costs lower than the actual ones) the encoder will be forced to perform a number of extra skips.

The encoding determines a sequence of costs that are used to update C. The process is repeated on the new cost matrix until the sequence of decisions suggested by the dynamic program is admissible, i.e. compatible with real encoding. While the computation of the cost matrix is susceptible of other approaches (for example the entire replacement of the macroblock rate control) our solution has the advantage that only a fraction of the entries of the cost matrix are actually computed.

In order to assess the performance of our solution, a set of representative video sequences was be selected. Video sequences used by standard committees for algorithm testing are common, however these files are not challenging because they are short (generally less than 200 frames) and contain only one scene. To test our methods on a set of video sequences with multiple scenes, six test sequences were generated by continuous sampling of television commercials (sequences Commercials_0 trough Commercials_5) at 30 frames per second with a QCIF resolution. Two othrt sequences, Std.qcif and Std100.qcif were artificially generated by concatenating nine standard sequences: Claire.qcif, Carphone.qcif, Foreman.qcif, Grandma.qcif, Miss_am.qcif, Mthr_dotr.qcif, Salesman.qcif, Suzie.qcif and Trevor.qcif.

Figure 6.6: The top strip shows a sample frame from each of the files forming Std.qcif and Std100.qcif; the bottom strips show sample frames from the commercials in the sequences Commercials_0.qcif.

The test sequence Std.qcif consists of the simple concatenation of the nine video sequences while Std100.qcif exhibits more scene cuts because the previous standard files are concatenated by alternating blocks of 100 frames. Despite the limited number of scene cuts present in Std.qcif and Std100.qcif, they were added to the test set to provide a standard reference.

Sample frames from the set of test sequences are illustrated in Figure 6.6.

The proposed frame layer rate control was evaluated by embedding it into the Telnor/UBC H.263+ framework, with the TMN–8 macroblock layer rate control. H.263+ was selected because it is currently regarded as state of the art low bit rate video compression (and our target is to address low bandwidth applications). The Telnor/UBC encoder is publicly available in source code and it is used to evaluate the proposals that the committee members bring to the standard; the core of this encoder provides the kernel for the MPEG–4 low bit rate video coding. There is no loss of generality in restricting experimentation to a single encoder since our algorithms can be adapted to every hybrid video encoder.

Sequence	n	Skipped Frames		Bit Rate (in Kb/s)		PSNR_Y (in dB)	
		TMN–8	***Optimal***	***TMN–8***	***Optimal***	***TMN–8***	***Optimal***
Commercials_0	4250	1782	1597	32.31	32.22	27.97	28.00
Commercials_1	5900	2151	1858	32.27	32.15	27.04	27.05
Commercials_2	4100	1063	940	32.55	32.54	28.83	28.84
Commercials_3	5000	1099	936	32.13	32.01	29.06	29.07
Commercials_4	4800	1897	1697	32.44	32.11	28.46	28.54
Commercials_5	3100	1386	1275	32.77	32.59	27.72	27.71
Std	4000	204	177	32.54	32.55	32.60	32.61
Std100	4000	396	343	32.57	32.60	31.74	31.76

Table 6.1: Optimal frame rate control. Number of skipped frames and PSNR in experimental results for sequences encoded at 32 Kb/s.

Table 6.1 compares the results obtained on the 32 Kb/s encoding of the video sequences with the TMN–8 rate control and a frame rate control based on the proposed dynamic programming algorithm. The standard and the optimized frame control methods are compared in terms of number of skipped frames, final bit rate and average Peak Signal to Noise Ration (PSNR) on the Y component.

As evident from the Table, the optimal frame rate control skips consistently fewer frames, while accurately achieving the target bit rate and, even more important, without compromising the per-frame or the overall quality of the video sequence. The average PSNR is calculated including all transmitted frames. So, while the final figures are comparable, the results listed for the optimal rate control refer to the transmission of a higher number of frames. As we also verified with an informal subjective assessment, the higher number of frames, transmitted at the same average PSNR, results in a higher overall visual quality.

6.6 An efficient sub–optimal heuristic

A careful analysis of the optimal decision sequences shows that most of the gain of our algorithm derives from a drastic reduction of the "chain effect" described previously (see point 3, Section 6.3). Motivated by this finding, we also experimented with a simple heuristic that proved to be very effective.

When a scene change occurs, instead of transmitting the first frame of the new scene, this heuristic skips the beginning of the new scene and transmits only the last frame skipped by the standard H.263+/TMN–8 encoder. This has the advantage of reducing both the jerk and the gap to the next encoded frame. Most of the time, the shorter distance prediction is able to avoid the extra skip outlined in Figure 6.3.

Figure 6.7 shows how this simple strategy works on the same sequence of frames described in Figure 6.3. As it can be seen from Figure 6.7, frame $n+3$ is encoded instead of frame n and, because $n+4$ is now predicted from the closer $n+3$ frame, no additional frame skipping is necessary. Figure 6.8 depicts the resulting improved encoding of the scene cut.

This approach is easy to implement and reduces the computation to at most two encodings of each frame. The application of this heuristic to the test sequences indicates that this new strategy performs almost optimally. Experiments on encoding the test sequences at 32 Kb/s show that an encoder that uses the heuristic frame rate control, consistently outperforms a TMN–8 encoder while improving the visual quality. Scenes are essentially left unchanged and a small gain in PSNR is also observed. Similar results (with smaller gains) are also obtained on the same sequences when encoded at 64 Kb/s.

Frame No.	n-5	n-4	n-3	n-2	n-1	n	n+1	n+2	n+3	n+4	n+5	n+6	n+7	…
Encode	n-5	n-4	n-3	n-2	n-1	*skip*			n+3	n+4	n+5	n+6	n+7	…
Transmit	n-5	n-4	n-3	n-2	n-1	n+3				n+4	n+5	n+6	n+7	…
Display	…	n-5	n-4	n-3	n-2	n-1				n+3	n+4	n+5	n+6	n+7

Figure 6.7: Encoding, transmitting and decoding/displaying a sequence of frames with the proposed heuristic frame rate control. The sequence contains a scene cut between the frames n-1 and *n*.

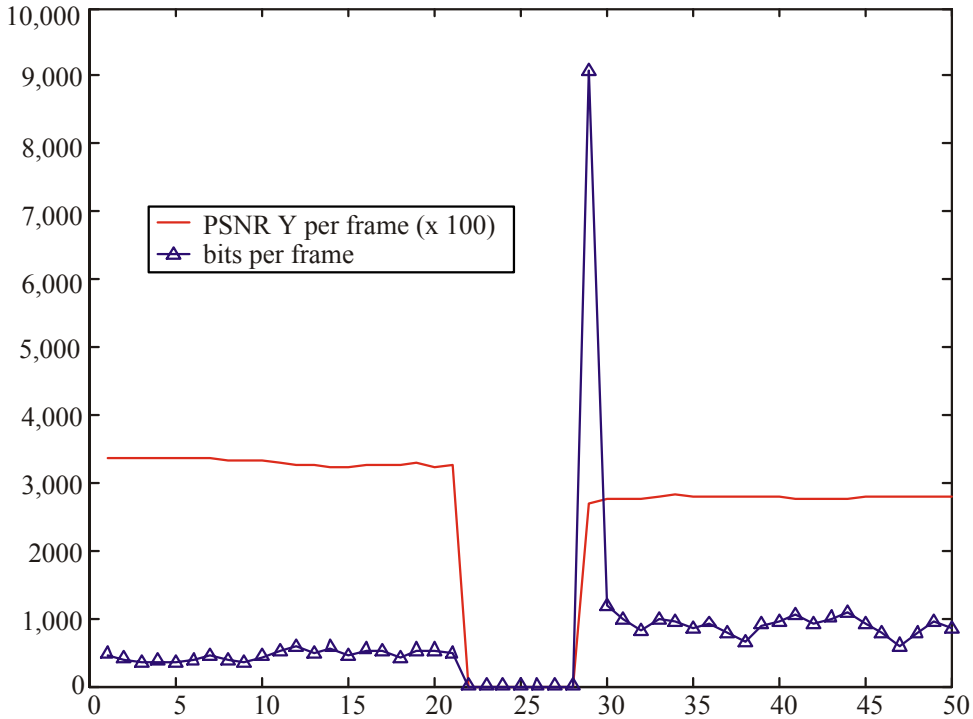


Figure 6.8: Sequence of 50 frames across a scene cut in one of the files used in our experiments (Claire.qcif 80-100 followed by Carphone.qcif 1-29) encoded at 32 Kbit/s with heuristic rate control. Bits per frame and the corresponding PSNR per frame are shown.

Table 6.2 compares the results obtained encoding at 32 Kb/s the test video sequences with the TMN–8 rate control, with the heuristic suggested before and with a frame rate control based on the dynamic programming algorithm. The frame control methods are compared in terms of number of skipped frames, final bit rate and PSNR on the Y component.

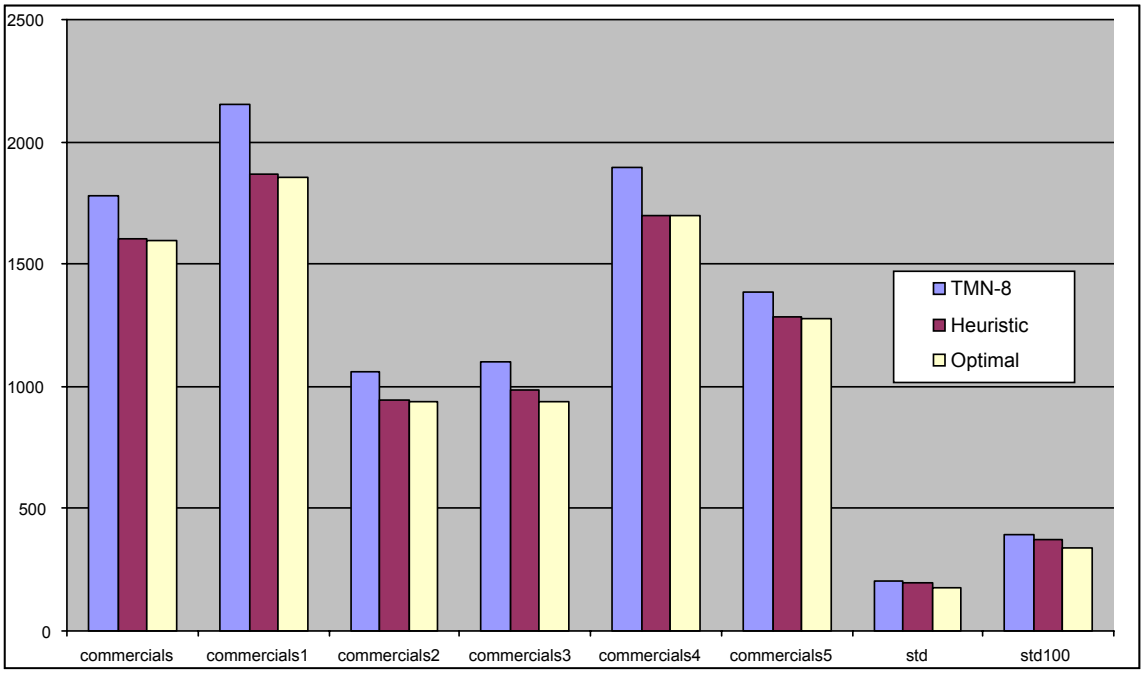


Figure 6.9: Number of frames skipped when encoding the video sequence at 32 Kbit/s with TMN–8, heuristic and optimal frame rate controls.

Sequence	N	Skipped Frames			Bit Rate (in Kb/s)			PSNR_Y (in dB)		
		TMN–8	***Heuristic***	***Optimal***	***TMN–8***	***Heuristic***	***Optimal***	***TMN–8***	***Heuristic***	***Optimal***
Commercials_0	4250	1782	1605	1597	32.31	32.12	32.22	27.97	27.99	28.00
Commercials_1	5900	2151	1866	1858	32.27	32.00	32.15	27.04	27.05	27.05
Commercials_2	4100	1063	947	940	32.55	32.55	32.54	28.83	28.84	28.84
Commercials_3	5000	1099	985	936	32.13	31.96	32.01	29.06	29.09	29.07
Commercials_4	4800	1897	1700	1697	32.44	32.13	32.11	28.46	28.52	28.54
Commercials_5	3100	1386	1282	1275	32.77	32.57	32.59	27.72	27.70	27.71
Std	4000	204	196	177	32.54	32.63	32.55	32.60	32.60	32.61
Std100	4000	396	376	343	32.57	32.64	32.60	31.74	31.75	31.76

Table 6.2: TMN–8, heuristic and optimal frame rate controls. Coding results at 32 Kb/s.

From the Table 6.2 and from Figure 6.9 is clear how the optimal frame rate control skips consistently fewer frames than the heuristic, while accurately achieving the target bit rate and, even more important, without compromising the quality of the video sequence. The small difference between heuristic and optimal algorithm confirms that most of the gain comes from encoding a frame after the scene cut that is closer to the next scene.

As described in precedence, PSNR is averaged on each transmitted frame, so the results listed for heuristic and optimal frame control are achieved by spending the same amount of bits of the TMN–8 rate control, but they refer to a consistently higher number of frames.

6.7 Unrestricted Optimization

In the previous section, we have considered a maximization problem MAX_TRANS characterized by two conditions which imply that a frame f_i with $0 \le i \le n$ can be transmitted immediately after the transmission of its anchor frame f_{i-A_i} is completed (even if this happens at a smaller than i) and that the transmission of f_i must be completed at some time greater than or equal to i, and before starting the next frame. While these two conditions guarantee that frames are temporally aligned within a range of $\pm\left\lfloor \max_{0 \le i, j \le n-1} C[i][j]/M \right\rfloor$, they may also be a source of inefficiency.

For example, consider a section of the video sequence having k consecutive frames $f_h, f_{h+1}, \ldots, f_{h+k}$ each with a cost of at most $M - \Delta_i$ bits. Even if the encoder skips no frame in that time interval, residual buffer space $R \ge \sum_{i=1}^{k-1} \Delta_{h+i}$ will be wasted. The sum of the residuals extends up to the residual generated by f_{h+k-1} because the last residual Δ_{h+k} may be used for the transmission of the frames following f_{h+k}.

For this reason, it may be interesting to consider a slightly different version of our MAX_TRANS optimization problem that is not restricted by these conditions. We prove that the unrestricted version of the optimization problem MAX_TRANS is NP–complete.

Definition 6.2: Let $I_u = (n, M, A_0, B_0, C)$ be an instance of the unrestricted optimization problem MAX_TRANS_U. The problem consists in finding a sequence of decisions $S_u = d_0, d_1, \ldots, d_{n-1}$ such that $D_u = \sum_{i=0}^{n-1} d_i$ is maximized and $B_0 + \sum_{i=0}^{n-1} d_i \cdot C[i][A_i] \leq n \cdot M$.

A video sequence that is encoded with a solution of the problem MAX_TRANS_U can still be decoded by a standard decoder, provided that the video frames are time aligned before being displayed. The value D_u of the unrestricted solution S_u provides an upper bound on the value of the solution S because $D \leq D_u$.

***Definition 6.3*:** An instance of the unrestricted decision problem MAX_TRANS_UD is given by $I_{ud} = (n, M, A_0, B_0, C)$ and a value goal $K \in \mathbb{R}^+$. The problem consists in determining if there exists a sequence of decisions $S_{ud} = d_0, d_1, \ldots, d_{n-1}$ such that:

$$B_0 + \sum_{i=0}^{n-1} d_i \cdot C[i][A_i] \leq n \cdot M \qquad \text{and} \qquad D_{ud} = \sum_{i=0}^{n-1} d_i \geq K.$$

Given a value $K \in \mathbb{R}^+$, solving MAX_TRANS_U provides a solution for the corresponding decision formulation in polynomial time by comparing the sum of the decisions D_u with K. This implies that solving MAX_TRANS_U is at least as hard as solving MAX_TRANS_UD, its formulation in terms of a decision problem, or, in other words, that MAX_TRANS_U is NP–complete if MAX_TRANS_UD is.

***Definition 6.4*:** An instance of the problem LONGEST PATH is a graph $G = (V, E)$ and a positive integer $K \leq |V|$. The problem consists in determining if there is in the graph G a simple path (that is, a path encountering no vertex more than once) with K or more edges (Garey and Johnson [1979]).

Given that LONGEST PATH is a known NP–complete problem we want to prove the following:

***Theorem 6.2*:** The problem MAX_TRANS_UD is NP–complete.

***Proof*:** To prove that the MAX_TRANS_UD is NP–complete we have to prove that:

a) MAX_TRANS_UD $\in$ NP;

b) A known NP–complete problem Π transforms to MAX_TRANS_UD. This means that there is a transformation T, computable in polynomial time by a deterministic Turing machine that transforms every instance of Π in an instance of MAX_TRANS_UD.

Part (a) is true because a non–deterministic Turing machine can compute the solution of the problem MAX_TRANS_UD in polynomial time by analyzing every possible sequence of decisions and selecting one that satisfies the conditions:

$$B_0 + \sum_{i=0}^{n-1} d_i \cdot C[i][A_i] \leq n \cdot M \qquad \text{and} \qquad D_{ud} = \sum_{i=0}^{n-1} d_i \geq K.$$

Part (b) is proved by restriction, i.e. by showing that the problem MAX_TRANS_UD contains an instance of the problem LONGEST PATH that is known to be NP–complete.

Given an instance $G = (V, E)$ and $K \leq |V|$ of the problem LONGEST PATH, we can construct an instance of the problem MAX_TRANS_UD as following:

1. $n = |V|$;
2. $M = 1$;
3. $A_0 = -1$;
4. $B_0 = 0$;
5. $C[i][j] = \begin{cases} 1 & \text{if} \quad (V_i, V_{i-j}) \in E \\ 0 & \text{otherwise} \end{cases}$
6. K has same meaning and assumes the same value in both problems.

The construction can be done in polynomial time and it transforms every instance of LONGEST PATH in an instance of MAX_TRANS_UD. It is also evident how deciding MAX_TRANS_UD decides the existence of a simple path in G with K or more edges.

CONCLUSIONS

Coherently with the trend existing in the literature in the field, our investigation has been based on a case–by–case analysis of the effects of using an optimization procedure in a data compression algorithm. The problem that we have addressed is how and how much the replacement of a sub–optimal strategy by an optimal one influences the performance of a data compression algorithm. We have analyzed three algorithms, each in a different domain of data compression. We introduced two novel algorithms that improve the current state of the art in the fields of low bit rate vector quantization and lossless image coding. We have also proposed and studied a new frame layer bit rate control algorithm compatible with the existing video compression standards.

Although most of the experiments that we have reported are focused on different applications of digital image compression (lossy, lossless and moving pictures), some of the algorithms are much more general and cover broader areas of data compression. For example, the trellis coded vector residual quantizer has been applied with success to the compression of speech at very low bit rate and to the compression of random sources. For the TCVRQ, we have proposed two methods for the design of the codebook: one is based on the optimality conditions that we have derived; the other, a greedy algorithm, is a simplified heuristic that shows remarkable performance.

The lossless image compression algorithm that we have introduced uses linear prediction and embedded context modeling. Two different methods to determine the optimal linear predictor have been compared and discussed. Several alternatives for the entropy coding of the prediction error have been explored successfully. The burden of performing pixel–by–pixel optimization is well compensated by the competitive performance that we were able to achieve. During our investigation, a number of ideas arose on possible improvements; the most promising dynamically adapts the context window to the image contents.

Finally, the problem of designing an optimal rate control algorithm suitable for low bit rate video encoding has been addressed. The proposed scheme minimizes the number of skipped frames and prevents buffer overflows. We present both an optimal procedure and a simplified heuristic based on the insights gathered during the evaluation of the optimal solution. When used in the H.263+ video encoder this heuristic achieves performance extremely close to the optimal but with a much lower computational complexity. Finally, an unrestricted version of the optimization problem has been studied and proved to be NP–complete.

Besides the aforementioned contributions, this work is relevant for a number of reasons:

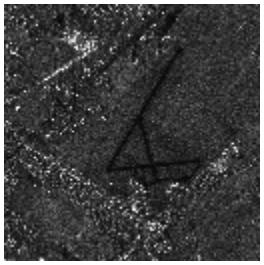
- A measure of the improvement achievable by an optimal strategy provides powerful insights about the best performance obtainable by a data compression algorithm;
- As we show in the case of low bit rate video compression, optimal algorithms can frequently be simplified to provide effective heuristics;

- Existing and new heuristics can be carefully evaluated by comparing their complexity and performance to the characteristics of an optimal solution;
- Since the empirical entropy of a "natural" data source is always unknown, optimal data compression algorithms provide improved upper bounds on that measure.

APPENDIX A

Thumbnails of the images used in the lossless image compression experiments.

Airplane

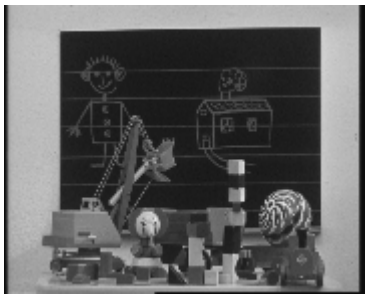


Airport

Ballon

Barb2

Barb



Board

Boats

Crowd

Girl

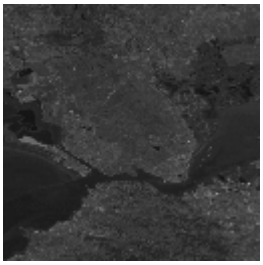

Gold

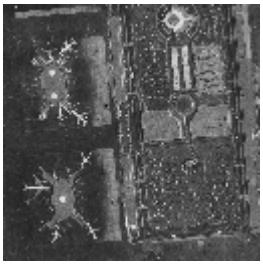

Goldhill

Hotel


Hursleyhouse


Lake

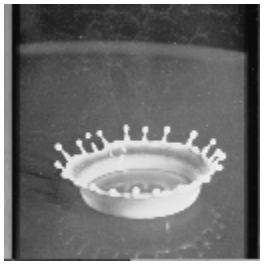

Landsat

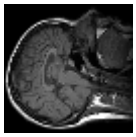

Lax

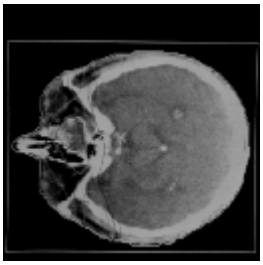

Lena


Lenna


Mandrill


Milkdrop

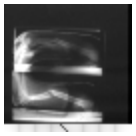

Mri


Mskull


Peppers


Woman1

Woman2



Xray

Zelda

BIBLIOGRAPHY

I. Abdelqader, S. Rajala and W. Snyder [1993]. "Motion Estimation from Noisy Image Data", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, V: 209–212.

G. Abousleman [1995]. "Compression of Hyperspectral Imagery Using Hybrid DPCM/DCT and Entropy–Constrained Trellis Coded Quantization", *Proc. Data Compression Conference, IEEE Computer Society Press*, 322–331.

B. Andrews, P. Chou, M. Effros, R. Gray [1993]. "A Mean–Removed Variation of Weighted Universal Vector Quantization for Image Coding", *Proc. Data Compression Conference, IEEE Computer Society Press*, 302–309.

A. Apostolico and S. Lonardi [1998]. "Some Theory and Practice of Greedy Off–Line Textual Substitution", *Proc. Data Compression Conference,* IEEE Computer Society Press*,* 119–128.

A. Apostolico and S. Lonardi [2000]. "Compression of Biological Sequences by Greedy Off–Line Textual Substitution", *Proc. Data Compression Conference*, IEEE Computer Society Press, 143–152.

R. Armitano, D. Florencio, R. Schafer [1996]. The Motion Transform: A New Motion Compensation Technique, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 2295–2298.

R. Armitano, R. Schafer, F. Kitson, V. Bhaskaran [1997]. "Robust Block-Matching Motion–Estimation Technique for Noisy Sources", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany*,* 2685–2688.

Z. Arnavut and S. Magliveras [1997]. "Block Sorting and Compression", *Proc. Data Compression Conference, IEEE Computer Society Press*, 181–190.

Z. Arnavut [1997]. "A Remapping Technique Based on Permutations for Lossless Compression of Multispectral Images", *Proc. Data Compression Conference, IEEE Computer Society Press*, 407–416.

R. Arnold and T. Bell [1997]. "A Corpus for the Evaluation of Lossless Compression Algorithms", *Proc. Data Compression Conference, IEEE Computer Society Press*, 201–210.

R. Arps [1974]. "Bibliography on Digital Graphic Image Compression and Quality", *IEEE Trans. on Information Theory, IT–20:1*, 120–122.

R. Arps [1980]. “Bibliography on Binary Image Compression”, *Proc. of the IEEE, 68:7*, 922–924.

R. Arps and T. Truong [1994]. “Comparison of International Standards for Lossless Still Image Compression”, Special Issue on Data Compression, J. Storer Editor, *Proc. of the IEEE 82:6*, 889–899.

P. Assuncao, M. Ghanbari [1997]. “Transcoding of MPEG–2 Video in the Frequency Domain”, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany, 2633–2636.

B. Atal [1986]. “High–quality speech at low bit rates: multi–pulse and stochastically excited linear predictive coders”, *Proc. Int. Conf. Acoustics Speech, Signal Process.*, Tokyo, 1681–1684.

B. Atal, V. Cuperman, and A. Gersho [1991]. *Advances in Speech Coding*, Kluwer Academic Press.

B. Atal and J. Remde [1982]. “A new model of LPC excitation for producing natural–sounding speech at low bit rates”, *Proc. IEEE Int. Conf. Acoustics., Speech, Signal Processing, vol. 1*, Paris, 614–617.

B. Atal and M. Schroeder [1979]. “Predictive coding of speech signals and subjective error criteria”, *IEEE Trans. in Signal Processing, ASSP–27, no 3*, 247–254.

P. Ausbeck Jr. [1998]. “Context Models for Palette Images”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 309–318.

E. Ayanoglu and R. Gray [1986]. “The design of predictive trellis waveform coders using the generalized Lloyd algorithm”, *IEEE Trans. Comm. , COM–34*, 1073–1080.

B. Balkenhol, S. Kurtz, and Y. Shtarkov [1999]. “Modification of the Burrows and Wheeler Data Compression Algorithm”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 188–197.

R. Barequet and M. Feder [1999]. “Siclic: A Simple Inter–Color Lossless Image Coder”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 501–510.

C. Barnes [1989]. *Residual Vector Quantizers*, Ph.D. Dissertation, Brigham Young University.

C. Barnes [1994]. “A New Multiple Path Search Technique for Residual Vector Quantizers”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 42–51.

C. Barnes and R. Frost [1990]. “Necessary conditions for the optimality of residual vector quantizers”, *Proc. IEEE International Symposium on Information Theory*.

R. Bascri, J. Mathews [1992]. “Vector Quantization of Images Using Visual Masking Functions”, *Proc. IEEE ICASSP Conference*, San Francisco, CA, 365–368.

B. Beferull–Lozano and A. Ortega [2001]. “Construction of Low Complexity Regular Quantizers for Overcomplete Expansions in R^N”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 193–202.

B. Belzer, J. Villasenor [1996]. “Symmetric Trellis Coded Vector Quantization”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 13–22.

D. Belinskaya, S. DeAgostino, and J. Storer [1995]. “Near Optimal Compression with Respect to a Static Dictionary on a Practical Massively Parallel Architecture”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 172–181.

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

T. Bell, I. Witten, and J. Cleary [1989]. “Modeling for Text Compression”, *ACM Computing Surveys* 21:4, 557–591.

B. Belzer, J. Villasenor [1996]. “Symmetric Trellis Coded Vector Quantization”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 13–22.

V. Bhaskaran and K. Konstantinides [1995]. *Image and Video Compression Standards*, Kluwer Academic Press.

B. Bhattacharya, W. LeBlanc, S. Mahmoud, and V. Cuperman [1992]. “Tree searched multi–stage vector quantization of LPC parameters for b/s speech coding” *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing vol. 1*, San Francisco, California, 105–108.

A. Brandenburg and G. Stoll [1992]. “The ISO/MPEG–audio codec: A generic standard for coding of high quality digital audio”, *92nd Audio Engineering Society Convention*, Vienna, preprint no. 3336.

K. Brandenburg, H. Gerhauser, D. Seitzer, and T. Sporer [1990]. “Transform coding of high quality digital audio at low bit rates — algorithms and implementations” *Proc. IEEE International Conference on Communications, vol. 3*, 932–6.

M. Bright and J. Mitchell [1999]. “Multi–Generation JPEG Images”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 517.

A. Brinkmann, J. I. Ronda, A. Pacheco and N. Garcia [1998]. “Adaptive Prediction Models for Optimization of Video Encoding”, *Proc. of the Visual Communications and Image Processing 1998*, San Jose'.

H. Brunk and N. Farvardin [1996]. “Fixed–Rate Successively Refinable Scalar Quantizers”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 250–259.

H. Brunk and N. Farvardin [1998]. “Embedded Trellis Coded Quantization”*, Proc. Data Compression Conference,* IEEE Computer Society Press*,* 93–102.

M. Burrows and D. Wheeler [1994]. “A Block–Sorting Lossless Data Compression Algorithm”, SRC Research Report, Digital Equipment Corporation Systems Research Center, Palo Alto, CA.

L. Butterman and N. Memon [2001]. “Error–Resilient Block Sorting”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 487.

A. Cafforio and F. Rocca [1983]. “The Differential Model for Motion Estimation”, *Image Sequence Processing and Dynamic Scene Analysis* (T. Huang, editor), Springer–Verlag, New York, NY.

J. Campbell, Jr., T. Tremain, and V. Welch [1991]. “The DOD 4.8 KBPS standard (proposed federal standard 1016)”, *Advances in Speech Coding* (B. Atal, V. Cuperman, and A. Gersho, editors), Kluwer Academic Press, 121–133.

R. Capocelli and A. De Santis [1988]. “Tight upper bounds on the redundancy of Huffman codes”, *Proc. IEEE International Symposium on Information Theory*, Kobe, Japan; also in *IEEE Trans. Inform. Theory IT–35:5* (1989).

R. Capocelli and A. De Santis [1991]. “A note on D–ary Huffman codes”, *IEEE Trans. Inform. Theory IT–37:1*.

R. Capocelli and A. De Santis [1991]. “New Bounds on the Redundancy of Huffman Code”, *IEEE Trans. on Information Theory IT–37*, 1095–1104.

R. Capocelli, R. Giancarlo, and I. Taneja [1986]. “Bounds on the redundancy of Huffman codes”, *IEEE Trans. on Information Theory IT–32:6*, 854–857.

B. Carpentieri [1994c]. “Split–Merge Displacement Estimation for Video Compression”, Ph.D. Dissertation, Computer Science Department, Brandeis University, Waltham, MA.

B. Carpentieri and J. Storer [1992]. “A Split–Merge Parallel Block Matching Algorithm for Video Displacement Estimation”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 239–248.

B. Carpentieri and J. Storer [1993]. “A Video Coder Based on Split–Merge Displacement Estimation”, *Proc. Data Compression Conference*, 492.

B. Carpentieri and J. Storer [1993]. “Split Merge Displacement Estimated Video Compression”, *Proc. 7th International Conference on Image Analysis and Processing*, Bari, Italy.

B. Carpentieri and J. Storer [1994]. “Split–Merge Video Displacement Estimation”, *Proc. of the IEEE 82:6*, 940–947.

B. Carpentieri and J. Storer [1994b]. “Optimal Inter–Frame Alignment for Video Compression”, *International Journal of Foundations of Computer Science 5:2*, 65–177.

B. Carpentieri and J. Storer [1995]. “Classification of Objects in a Video Sequence”, *Proc. SPIE Symposium on Electronic Imaging*, San Jose, CA.

B. Carpentieri and J. Storer [1996]. “A Video Coder Based on Split–Merge Displacement Estimation”, *Journal of Visual Communication and Visual Representation 7:2*, 137–143, 1996.

B. Carpentieri, M. Weinberger and G. Seroussi [2000]. “Lossless Compression of Continuous–Tone Images”, *Proc. of the IEEE,* Special Issue on Lossless Data Compression, Nov. 2000, Vol.88, No.11, 1797-1809.

W. Chan [1992]. “The Design of Generalized Product–Code Vector Quantizers”, *Proc. IEEE ICASSP Conference*, San Francisco, CA, 389–392.

M. Chang and G. Langdon [1991]. “Effects of Coefficient Coding on JPEG Baseline Image Compression”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 430.

P. Chang and R. Gray [1986]. “Gradient Algorithms for Designing Predictive VQs”, *IEEE ASSP–34*, 957–971.

W. Chau, S. Wong, X. Yang, and S. Wan [1991]. “On the Selection of Color Basis for Image Compression”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 441.

W. Chen and W. Pratt [1984]. “Scene adaptive coder”, *IEEE Trans. Comm. 32*, 225.

M. Chen, a. Wilson [1996]. Rate–Distortion Optimal Motion Estimation algorithm for Video Coding, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 2096–2099.

P. Chou, T. Lookabaugh, and R. Gray [1989]. “Entropy–constrained vector quantization”, *IEEE Trans. Acoustics, Speech, Signal Process.* 37:1, 31–42.

P. Chou, S. Mehrotra, and A. Wang [1999]. “Multiple Description Decoding of Overcomplete Expansions Using Projections onto Convex Sets”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 72–81.

R. Clarke [1999]. *Digital Compression of Still Images and Video*, Academic Press.

J. Cleary and I. Witten [1984]. “Data Compression Using Adaptive Coding and Partial String Matching”, *IEEE Trans. on Communications 32:4*, 396–402.

J. Cleary, W. Teahan, and I. Witten [1995]. “Unbounded Length Contexts for PPM”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 52–61.

M. Cohn [1988]. “Performance of Lempel–Ziv Compressors with Deferred Innovation”, Proc. 1988 NASA Conference on Scientific Data Compression, IEEE Computer Society Press, 377–389.

M. Cohn [1989]. “Bounds for lossy text compression” *Proc. Informationstheorie, Mathematische Forschungsinstitut Wolfach*.

M. Cohn [1992]. “Ziv–Lempel Compressors with Deferred–Innovation”, *Image and Text Compression*, Kluwer Academic Press, 145–158.

M. Cohn, R. Khazan [1996]. “Parsing with Suffix and Prefix Dictionaries”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 180–189.

L. Colm Stewart [1981]. *Trellis Data Compression*, Xerox, Palo Alto Research Center.

C. Constantinescu [1995]. *Single–Pass Adaptive Vector Quantization*, Ph.D. Dissertation, Brandeis University.

C. Constantinescu and J. Storer [1994]. "On–Line Adaptive Vector Quantization with Variable Size Codebook Entries", *Information Processing and Management 30:6*, 745–758; an extended abstract of this paper also appeared in *Proc. Data Compression Conference, IEEE Computer Society Press*, 32–41.

C. Constantinescu and J. Storer [1994b]. "Improved Techniques for Single–Pass Adaptive Vector Quantization", *Proc. of the IEEE, 82:6*, 933–939.

C. Constantinescu and J. Storer [1995]. "Application of Single–Pass Adaptive VQ to Bilevel Images", *Proc. Data Compression Conference,* 423.

G. Cormack and R. Horspool [1984]. "Algorithms for Adaptive Huffman Codes", *Information Processing Letters* 18, 159–165.

T. Cormen, C. Leiserson and R. Rivest [1990]. "*Introduction to Algorithms*", McGraw–Hill, 1990.

G. Côté, B. Erol, M. Gallant and F. Kossentini [1998]. "H.263+: Video Coding at Low Bit Rates", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.8, o.7.

G. Côté, M. Gallant and F. Kossentini [1998b]. "Description and Results for Rate–Distortion Based Quantization", *Doc. ITU–T/SG16/Q15–D–51*.

T. Cover and J. Thomas [1991]. *Elements of Information Theory*, Wiley.

D. Crowe [1992]. "Objective quality assessment", *Digest, IEE Colloquium on Speech Coding — Techniques and Applications*, London, 5/1–5/4.

S. Daly [1992]. "Incorporation of Imaging System and Visual Parameters into JPEG Quantization Tables", *Proc. Data Compression Conference*, IEEE Computer Society Press, 410.

A. Das and A. Gersho [1995]. "Variable Dimension Spectral Coding of Speech at 2400 bps and Below with Phonetic Classification", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 492–495.

Y. Dehery, M. Lever, and P. Urcun [1991]. "A MUSICAM source codec for digital audio broadcasting and storage", *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Proc, vol. 1*, 3605–9.

A. DeJaco, W. Gardner, P. Jacobs and C. Lee [1993]. "QCELP: the North American CDMA digital cellular variable rate speech coding standard", *Proc. IEEE Workshop on Speech Coding for Telecommunications*, 5–6.

C. Derviaux, F. Coudoux, M. Gazalet, P. Corlay [1997]. "A Postprocessing Technique for Block Effect Elimination Using a Perceptual Distortion Measure", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany, 3001–3004.

M. Effros [1999]. "Universal Lossless Source Coding with the Burrows Wheeler Transform", *Proc. Data Compression Conference,* IEEE Computer Society Press, 178–187.

N. Ekstrand [1996]. "Lossless Compression of Grayscale Images via Context Tree Weighting", *Proc. Data Compression Conference,* IEEE Computer Society Press, 132–139.

P. Elias [1970]. “Bounds on performance of optimum quantizers”, *IEEE Trans. on Information Theory* IT–16, 172–184.

B. Erol, M. Gallant, G. Cote and F. Kossentini [1998]. “The H.263+ Video Coding Standard: Complexity and Performance”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 259–268.

M. Feder and A. Singer [1998]. “Universal Data Compression and Linear Prediction”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 511–520.

P. Fenwick [1996]. “The Burrows–Wheeler Transform for Block Sorting Text Compression: Principles and Improvements”, *The Computer Journal 39:9*, 731–740.

T. Fischer, M. Marcellin and M. Wang [1991b]. “Trellis Coded Vector Quantization”, *IEEE Trans. on Information Theory*, IT–37, 1551–1566.

T. Fischer and M. Wang [1991]. “Entropy–Constrained Trellis Coded Quantization”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 103–112.

Y. Fisher, Ed. [1994]. *Fractal Encoding – Theory and Applications to Digital Images*, Springer–Verlag.

Y. Fisher, Ed. [1995]. *Fractal Image Compression: Theory and Application*, Springer–Verlag.

J. Flanagan, M. Schroeder, B. Atal, R. Crochiere, N. Jayant, and J. Tribolet [1979]. “Speech Coding”, *IEEE Trans. on Communications, COM–27:4*, 710–737.

M. Flierl, T. Wiegand and B. Girod [1998]. “A Locally Optimal Design Algorithm for Block–Based Multi–Hypothesis Motion–Compensated Prediction”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 239–248.

S. Forchhammer, X. Wu, and J. Andersen [2001]. “Lossless Image Data Sequence Compression Using Optimal Context Quantization”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 53–62.

R. Frost, C. Barnes, and F. Xu [1991]. “Design and Performance of Residual Quantizers”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 129–138.

M. Garey and D. Johnson [1979]. “*Computers and Intractability - A Guide To The Theory of NP-completeness*”, Freeman Pub.,1979.

R. Gallager [1968]. *Information Theory and Reliable Communication*, Wiley.

R. Gallager [1978]. “Variations on a Theme by Huffman”, *IEEE Trans. on Information Theory* 24:6, 668–674.

T. Gardos [1997]. “Video Codec Test Model, Near–Term, Version 8 (TMN–8)”, *Doc. ITU–T/SG16/Q15–D65d1*.

T. Gardos [1998]. “Video Codec Test Model, Near–Term, Version 9 (TMN–9)”, *Doc. ITU–T/SG16/Q15–A–59*.

A. Gersho [1994]. “Advances in Speech and Audio Compression”, Special Issue on Data Compression, J. Storer ed., *Proc. of the IEEE 82:6*, 900–918.

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

S. Golomb [1966]. “Run–Length Encoding”, *IEEE Trans. on Information Theory* 12, 399–401.

R. Gonzalez and P. Wintz [1987]. *Digital Image Processing*, Addison–Wesley.

R. Gonzalez and R. Woods [1992]. *Digital Image Processing*, Addison–Wesley.

U. Graef [1999]. “Sorted Sliding Window Compression”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 527.

R. Gray [1984]. “Vector quantization”, *IEEE ASSP Magazine* 1, 4–29.

A. Gray, Jr. and J. Markel [1976]. “Distance measures for speech processing”, *IEEE Trans. on Acoustics, Speech, and Signal Processing,* ASSP–24:5, 380–391.

R. Gray, P. Cosman, and E. Riskin [1992]. “Image Compression and Vector Quantization”, in *Image and Text Compression*, Kluwer Academic Press.

A. Hartman and M. Rodeh [1985]. “Optimal Parsing of Strings”, *Combinatorial Algorithms on Words*, Springer–Verlag (A. Apostolico and Z. Galil, editors), 155–167.

B. Haskell, A. Puri, and A. Netravali [1997]. *Digital Video: An Introduction to MPEG–2*, Chapman and Hall.

H. Helfgott and M. Cohn [1997]. “On Maximal Parsing of Strings”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 291–299.

H. Helfgott and M. Cohn [1998]. “Linear Time Construction of Optimal Context Trees”*, Proc. Data Compression Conference,* IEEE Computer Society Press*,* 369–377.

J. Herre, E. Eberlein, H. Schott, and K. Brandenburg [1992]. “Advanced audio measurement system using psychoacoustics properties” *92nd Audio Engineering Society Convention* Vienna, 3321.

D. T. Hoang [1997]. *Fast and Efficient Algorithms for Text and Video Compression*, Ph.D. Dissertation, Department of Computer Science, Brown University.

D. Hoang [1999]. “Real–Time VBR Rate Control of MPEG Video Based Upon Lexicographic Bit Allocation”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 374–383.

D. Hoang, E. Linzer, and J. Vitter [1997]. “A Lexicographic Framework for MPEG Rate Control”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 101–110.

D. Hoang, P. Long, and J. Vitter [1994]. “Explicit Bit Minimization for Motion–Compensated Video Coding”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 175–184.

D. Hoang, P. Long, J. Vitter [1996]. "Efficient Cost Measures for Motion Compensation at Low Bit Rates", *Proc. Data Compression Conference, IEEE Computer Society Press*, 102–111.

R. Horspool [1995]. "The Effect of Non–Greedy Parsing in Ziv–Lempel Compression Methods", *Proc. Data Compression Conference, IEEE Computer Society Press*, 302–311.

P. Howard [1989]. *Design and Analysis of Efficient Lossless Compression Systems*, Ph.D. Dissertation, Computer Science Dept., Brown University, Providence, RI.

P. Howard, J. Vitter [1993]. "Fast and Efficient Lossless Image Compression", *Proc. Data Compression Conference, IEEE Computer Society Press*, 351–360.

D. Huffman [1952]. "A Method for the Construction of Minimum–Redundancy Codes", *Proc. of the IRE* 40, 1098–1101.

I. Ismaeil, A. Docef, F. Kossentini, and R. Ward [1999]. "Motion Estimation Using Long Term Motion Vector Prediction", *Proc. Data Compression Conference,* IEEE Computer Society Press, 531.

A. Jacquin, H. Okada, and P. Crouch [1997]. "Content–Adaptive Postfiltering for Very Low Bit Rate Video", *Proc. Data Compression Conference, IEEE Computer Society Press*, 111–121.

H. Jafarkhani and V. Tarokh [1998]. "Successfully Refinable Trellis Coded Quantization"*, Proc. Data Compression Conference,* IEEE Computer Society Press*,* 83–92.

J. Jain and A. Jain [1981]. "Displacement Measurement and its Applications in Interframe Image Coding", *IEEE Trans. on Communications COM–29:12*, 1799–1808.

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

O. Johnsen [1980]. "On the Redundancy of Binary Huffman Codes", *IEEE Trans. on Information Theory* 26:2, 220–222.

K. Joo, D. Gschwind, T. Bose [1996]. ADPCM Encoding of Images Using a Conjugate Gradient Based Adaptive Algorithm, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 1942–1945.

B. Juang and A. Gray, Jr. [1982]. "Multiple stage vector quantization for speech coding", *Proc. Int. Conf. Acoustics, Speech and Signal Processing,* Vol. 1, Paris, 597–600.

J. Kari and M. Gavrilescu [1998]. "Intensity Controlled Motion Compensation"*, Proc. Data Compression Conference,* IEEE Computer Society Press*,* 249–258.

J. Katajainen and T. Raita [1987]. "An Analysis of the Longest Match and the Greedy Heuristics for Text Encoding", Technical Report, Department of Computer Science, University of Turku, Turku, Finland.

J. Katto and M. Ohta [1995]. "Mathematical Analysis of MPEG Compression Capability and its Applications to Rate Control", *Proc. of the ICIP95*, Washington D.C..

T. Kaukoranta, P. Franti, and O. Nevalainen [1999]. "Reduced Comparison Search for the Exact GLA", *Proc. Data Compression Conference,* IEEE Computer Society Press, 33–41.

A. Kess and S. Reichenbach [1997]. "Capturing Global Redundancy to Improve Compression of Large Images", *Proc. Data Compression Conference, IEEE Computer Society Press*, 62–71.

A. Kondoz [1994]. *Digital Speech*, Wiley.

F. Kossentini, M. Smith, C. Barnes [1992]. "Image Coding with Variable Rate RVQ", *Proc. IEEE ICASSP Conference*, San Francisco, CA, 369–372.

F. Kossentini, M. Smith and C. Barnes [1993]. "Entropy–Constrained Residual Vector Quantization", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, V: 598.

D. Kozen, Y. Minsky, and B. Smith [1998]. "Efficient Algorithms for Optimal Video Transmission", *Proc. Data Compression Conference,* IEEE Computer Society Press*,* 229–238.

A. Lan and J. Hwang [1997]. "Scene Context Dependent Reference Frame Placement for MPEG Video Coding", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany*,* 2997–3000.

G. Langdon [1991]. "Sunset: A Hardware–Oriented Algorithm for Lossless Compression of Gray Scale Images", *SPIE Medical Imaging V: Image Capture, Formatting, and Display 1444*, 272–282.

G. Langdon, A. Gulati, and E. Seiler [1992]. "On the JPEG Context Model for Lossless Image Compression", *Proc. Data Compression Conference*, IEEE Computer Society Press, 172–180.

R. Laroia and N. Farvadin [1994]. "Trellis–based scalar–vector quantizers for memoryless sources", *IEEE Trans. on Information Theory*, IT–40, No.3.

D. Le Gall [1991]. "MPEG: A video compression standard for multimedia applications", *Communications of the ACM* 34:4, 46–58.

J. Lee and B.W. Dickinson [1994]. "Joint Optimization of Frame Type Selection and Bit Allocation for MPEG Video Encoders", *Proc. of the ICIP94*, Vol.2.

D. Lelewer and D. Hirschberg [1987]. "Data Compression", *ACM Computing Surveys 19:3*, 261–296.

A. Lempel, S. Even, and M. Cohn [1973]. "An Algorithm for Optimal Prefix Parsing of a Noiseless and Memoryless Channel", *IEEE Trans. on Information Theory* 19:2, 208–214.

A. Lempel and J. Ziv [1976]. "On the Complexity of Finite Sequences", *IEEE Trans. on Information Theory* 22:1, 75–81.

A. Li, S. Kittitornkun, Y. Hu, D. Park, and J. Villasenor [2000]. "Data Partitioning and Reversible Variable Length Codes for Robust Video Communications", *Proc. Data Compression Conference*, IEEE Computer Society Press, 460–469.

J. Lin [1992]. *Vector Quantization for Image Compression: Algorithms and Performance*, Ph.D. Dissertation, Computer Science Dept. Brandeis University, MA.

J. Lin and J. Storer [1993]. "Design and Performance of Tree–Structured Vector Quantizers", *Proc. Data Compression Conference, IEEE Computer Society Press*, 292–301.

J. Lin and J. Vitter [1992]. "Nearly Optimal Vector Quantization via Linear Programming", *Proc. Data Compression Conference*, IEEE Computer Society Press, 22–31.

K. Lin and R. Gray [2001]. "Video Residual Coding Using SPIHT and Dependent Optimization", *Proc. Data Compression Conference*, IEEE Computer Society Press, 113–122.

K. Lin and R. Gray [2001]. "Rate–Distortion Optimization for the SPHIT Encoder", *Proc. Data Compression Conference*, IEEE Computer Society Press, 123–132.

J. Lin, J. Storer, and M. Cohn [1991]. "On the Complexity of Optimal Tree Pruning for Source Coding", *Proc. Data Compression Conference*, IEEE Computer Society Press, 63–72.

J. Lin, J. Storer, and M. Cohn [1992]. "Optimal Pruning for Tree–Structured Vector Quantization", *Information Processing and Management 28:6*, 723–733.

Y. Linde, A. Buzo, and R. Gray [1980]. "An algorithm for vector quantizer design", *IEEE Trans. on Communications* 28, 84–95.

E. Linzer, P. Tiwari, M. Zubair [1996]. High Performance algorithms for Motion Estimation for MPEG Encoder, *Proc. IEEE International Conference on*

S. Lloyd [1957]. "Least Squares Quantization in PCM", *Bell Laboratories Technical Note*, 1957.

M. Luttrell, J. Wen, J. D. Villasenor and J. H. Park [1998]. "Simulation Results for Adaptive Quantization Using Trellis Based R–D Information", *Doc. ITU–T/SG16/Q15–E–21*.

J. Markel and A. Gray, Jr. [1976]. *Linear Prediction of Speech*, Springer–Verlag.

J. Makhoul [1975]. "Linear prediction: A tutorial review", *Proc. IEEE 63*, 561–580.

J. Makhoul, S. Roucos, and H. Gish [1985]. "Vector Quantization in Speech Coding", *Proc. of the IEEE* 73:11, 1551–1588.

H. Malvar [2001]. "Fast Adaptive Encoder for Bi–Level Images", *Proc. Data Compression Conference*, IEEE Computer Society Press, 253–262.

M. Marcellin [1990]. "Transform coding of images using trellis coded quantization", *Proc. International Conference on Acoustics, Speech and Signal Process.* , 2241–2244.

M. Marcellin and T. Fischer [1990]. "Trellis coded quantization of memoryless and Gauss–Markov sources", *IEEE Trans. on Communications, COM–38*, 82–93.

M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek [2000]. "An Overview of JPEG–2000", *Proc. Data Compression Conference*, IEEE Computer Society Press, 523–541.

T. Markas and J. Reif [1993]. "Multispectral Image Compression Algorithms", *Proc. Data Compression Conference, IEEE Computer Society Press*, 391–400.

J. Max [1960]. "Quantizing for Minimum Distortion", *IRE Trans. in Information Theory*, IT-6(2), 7-12, 1960.

R. McEliece [1977]. *The Theory of Information and Coding*, Addison–Wesley.

B. Meyer and P. Tischer [1997]. "TMW — a New Method for Lossless Image Compression", *International Picture Coding Symposium PCS97 Conference Proc.*.

B. Meyer and P. Tischer [1998]. "Extending TMW for Near Lossless Compression of Greyscale Images", *Proc. Data Compression Conference,* IEEE Computer Society Press*,* 458–470.

B. Meyer and P. Tischer [2001]. "Glicbawls – Grey Level Image Compression By Adaptive Weighted Least Squares", *Proc. Data Compression Conference*, IEEE Computer Society Press, 503.

B. Meyer and P. Tischer [2001]. "TMW–Lego — An Object Oriented Image Modeling Framework ", *Proc. Data Compression Conference*, IEEE Computer Society Press, 504.

J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall [1997]. *MPEG Video Compression Standard*, Chapman and Hall.

A. Moffat, R. Neal, and I. Witten [1995]. "Arithmetic Coding Revisited", *Proc. Data Compression Conference, IEEE Computer Society Press*, 202–211.

A. Moffat [1990]. "Implementing the PPM Data Compression Scheme", *IEEE Trans. on Communications 38:11*, 1917–1921.

G. Motta [1993]. *Compressione della Voce a 2.4 Kbit/s*, Technical Report, CEFRIEL – Politecnico di Milano.

G. Motta, B. Carpentieri [1997]. "A New Trellis Vector Residual Quantizer: Applications to Image Coding", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany*,* 2929–2932.

G. Motta and B. Carpentieri [1997b]. "Trellis Vector Residual Quantization", *Proc. International Conference on Signal Processing Applications and Technology (ICSPAT97)*.

G. Motta, J. Storer and B. Carpentieri [1999]. "Adaptive Linear Prediction Lossless Coding", *Proc. Data Compression Conference,* IEEE Computer Society Press, 491–500.

G. Motta, J. Storer, and B. Carpentieri [2000]. "Improving Scene Cut Quality for Real–Time Video Decoding", *Proc. Data Compression Conference*, IEEE Computer Society Press, 470–479.

G. Motta, J. Storer, and B. Carpentieri [2000b]. "Lossless Image Coding via Adaptive Linear Prediction and Classification", *Proc. on the IEEE, Special Issue on Lossless Compression*, Nov. 2000, Vol.88, No.11, 1790-1896.

A. Nosratinia, M. Orchard [1996]. Optimal Warping Prediction for Video Coding, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 1986–1989.

K. Oehler and R. Gray [1993]. "Mean–Gain–Shape Vector Quantization", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, V: 241–244.

A. V. Oppenheim, R. W. Schafer, J. R. Buck [2000]. *Discrete–Time Signal Processing*, Prentice–Hall.

A. Ortega [1996]. "Optimal Bit Allocation under Multiple Rate Constraints", *Proc. Data Compression Conference, IEEE Computer Society Press*, 349–358.

K. Paliwal and B. Atal [1991]. "Efficient Vector Quantization of LPC Parameters at 2.4KBits/Frame", *IEEE Int. Conf. on Acoustics, Speech., Signal Processing*, 661–664.

K. Panusopone, K. Rao [1997]. "Efficient Motion Estimation for Block Based Video Compression", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany, 2677–2680.

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

W. Pennebaker, J. Mitchell, G. Langdon and R. Arps [1988]. "An overview of the basic principles of the Q–coder", *IBM Journal of Research and Development*, 32:6, 717–726.

J. G. Proakis and D. G. Manolakis [1996]. *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice Hall.

L. Rabiner and R. Schafer [1978]. *Digital Processing of Speech Signals*, Prentice–Hall.

M. Rabbani and P. Jones [1991]. *Digital Image Compression Techniques*, SPIE Optical Eng. Press.

S. Rajala, I. Abdelqader, G. Bilbro, W. Snyder [1992]. "Motion Estimation Optimization", *Proc. IEEE ICASSP Conference*, San Francisco, CA, 253–256.

K. Ramchandran and M. Vetterli [1994]. "Syntax–Constrained Encoder Optimization Using Adaptive Quantization Thresholding for JPEG/MPEG Coders", *Proc. Data Compression Conference, IEEE Computer Society Press*, 146–155.

K. Rao and P. Yip [1990]. *Discrete Cosine Transform – Algorithms, Advantages, Applications*, Academic Press.

V. Ratnakar and M. Livny [1995]. "RD–OPT: An Efficient Algorithm for Optimizing DCT Quantization Tables", *Proc. Data Compression Conference, IEEE Computer Society Press*, 332–342.

V. Ratnakar and M. Livny [1996]. "Extending RD–OPT with Global Thresholding for JPEG Optimization", *Proc. Data Compression Conference, IEEE Computer Society Press*, 379–386.

T. Reed, V. Algazi, G. Ford, and I. Hussain [1992]. "Perceptually Based Coding of Monochrome and Color Still Images", *Proc. Data Compression Conference*, IEEE Computer Society Press, 142–151.

H. Reeve and J. Lim [1984]. “Reduction of blocking effects in image coding”, *Optical Engineering* 23:1, 34–37.

J. Reif and J. Storer [1998]. “Optimal Lossless Compression of a Class of Dynamic Sources”, *Proc. Data Compression Conference,* IEEE Computer Society Press*,* 501–510.

J. Ribas–Corbera and S. Lei [1997]. “A Quantizer Control Tool for Achieving Target Bit Rates Accurately”, *Doc. LBC–97–071*.

J. Ribas–Corbera and S. Lei [1997b]. “Rate–Control for Low–Delay Video Communications”, *Doc. ITU–T/SG16/Q15–A–20*.

E. Riskin [1990]. *Variable Rate Vector Quantization of Images*, Ph.D. Dissertation, Stanford University, CA.

J. Rissanen [1983]. “A Universal Data Compression System”, *IEEE Trans. on Information Theory 29:5*, 656–664.

J. Rissanen and G. Langdon [1981]. “Universal Modeling and Coding”, *IEEE Trans. on Information Theory 27:1*, 12–23.

F. Rizzo and J. Storer [2001]. “Overlap in Adaptive Vector Quantization”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 401–410.

F. Rizzo, J. Storer and B. Carpentieri [1999]. “Experiments with Single–Pass Adaptive Vector Quantization”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 546.

F. Rizzo, J. Storer and B. Carpentieri [2001]. “LZ-based Image Compression”, *Information Sciences,* 135 (2001), 107-122.

J. Ronda, F. Jaureguizar and N. Garcia [1996]. “Overflow–Free Video Coders: Properties and Optimal Control Design”, *Visual Communications and Image Processing 1996*, Proc. SPIE Vol. 2727.

J. Ronda, F. Jaureguizar and N. Garcia [1996b]. “Buffer–Constrained Coding of Video Sequences with Quasi–Constant Quality”, *Proc. of the ICIP96*, Lausanne.

J. Ronda, M. Eckert, S. Rieke, F. Jaureguizar and A. Pacheco [1998]. “Advanced Rate Control for MPEG–4 Coders”, *Proc. of the Visual Communications and Image Processing '98*, San Jose'.

D. Salomon [1997]. *Data Compression: The Complete Reference*, Springer–Verlag.

K. Sayood [1996]. *Introduction to Data Compression*, Morgan Kaufmann Publishers.

G. Schaefer [2001]. “JPEG Compressed Domain Image Retrieval by Colour and Texture”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 514.

M. Schroeder and B. Atal [1985]. “Code–Excited Linear Prediction (CELP) High Quality Speech at Very Low Bit Rate”. *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 937–940.

G. Seroussi and M. Weinberger [1997]. "On Adaptive Strategies for Extended Family of Golomb–Type Codes", *Proc. Data Compression Conference, IEEE Computer Society Press*, 131–140.

C. Shannon [1948]. "A mathematical theory of communication", *Bell Syst. Tech. J.* 27, 379–423, 623–656; also in *The Mathematical Theory of Communication*, C. Shannon and W. Weaver, University of Illinois Press, Urbana, IL (1949).

C. Shannon [1959]. "Coding Theorems for a Discrete Source with a Fidelity Criterion", *Proc. IRE National Conference*, 142–163; also in *Key Papers in the Development of Information Theory* (D. Slepian, editor), IEEE Press, New York, NY (1973).

T. Sikora [1997]. "MPEG Digital Audio and Video Coding Standards", *IEEE Signal Processing Magazine* (September), 58–81.

H. Song, J. Kim and C. C. Jay Kuo [1998]. "Real–Time Motion–Based Frame Rate Control Algorithm for H.263+", *Doc: ITU–T/SG16/Q15–F–14*.

H. Song, J. Kim and C. C. Jay Kuo [1999]. "Performance Analysis of Real–Time Encoding Frame Rate Control Proposal", *Doc: ITU–T/SG16/Q15–G–22*.

J. Storer [1977]. "NP–Completeness Results Concerning Data Compression", Technical Report 234, Dept. of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ.

J. Storer [1979]. "Data Compression: Methods and Complexity Issues", Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, Princeton University Princeton, NJ.

J. Storer [1983]. "An Abstract Theory of Data Compression", *Theoretical Computer Science* 24 221–237; see also "Toward an Abstract Theory of Data Compression", *Proc. Twelfth Annual Conference on Information Sciences and Systems,* The Johns Hopkins University, Baltimore, MD, 391–399 (1978).

J. A. Storer [1988]. *Data Compression: Methods and Theory*, Computer Science Press (a subsidiary of W. H. Freeman Press).

J. A. Storer, Ed. [1992]. *Image and Text Compression*, Kluwer Academic Press.

J. Storer and H. Helfgott [1997]. "Lossless Image Compression by Block Matching", *The Computer Journal 40:2/3*, 137–145.

J. Storer and J. Reif [1995]. "Error Resilient Optimal Data Compression", *SIAM Journal of Computing 26:4*, 934–939.

J. Storer and J. Reif [1997]. "Low–Cost Prevention of Error Propagation for Data Compression with Dynamic Dictionaries", *Proc. Data Compression Conference*, IEEE Computer Society Press, 171–180.

J. Storer and T. Szymanski [1978]. "The Macro Model for Data Compression", *Proc. Tenth Annual ACM Symposium on the Theory of Computing*, San Diego, CA, 30–39.

J. Storer and T. Szymanski [1982]. “Data Compression Via Textual Substitution”, *Journal of the ACM*, 29:4 928–951.

G. Sullivan and T. Wiegand [1998]. “Rate–Distortion Optimization for Video Compression”, Draft for submission to *IEEE Signal Processing Magazine*, Nov. 1998 issue.

P. Tai, C. Liu and J. Wang [2001]. “Complexity–Distortion Optimal Search Algorithm for Block Motion Estimation”, *Proc. Data Compression Conference*, IEEE Computer Society Press, 519.

W. Teahan, J. Cleary [1996]. “The Entropy of English Using PPM–Based Models”, *Proceedings Data Compression Conference, IEEE Computer Society Press*, 53–62.

D. Thompkins and F. Kossentini [1999]. “Lossless JBIG2 Coding Performance”, *Proc. Data Compression Conference,* IEEE Computer Society Press, 553.

P. Tivari, E. Viscito [1996]. A Parallel MPEG2 Video Encoder with Look–Ahead Rate Control, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 1994–1997.

T. Tremain [1982]. “The government standard linear predictive coding algorithm: LPC–10”, *IEEE Trans. Acoustics, Speech and Signal Processing, ASSP–36:9*, 40–49.

K. Tsutsui, H. Suzuki, O. Shimoyoshi, M Sonohara, K. Agagiri, and R. Heddle [1992]. “ATRAC; Adaptive Transform. Acoustics coding for MiniDisc”, *Conf. Rec. Audio Engineering Society Convention* San Francisco.

G. Ungerboeck [1982]. “Channel coding with multilevel/phase signals”, *IEEE Trans. on Information Theory*, IT–28, 55–67.

R. Vander Kam and P. Wong [1994]. “Customized JPEG Compression for Grayscale Printing”, *Proc. Data Compression Conference, IEEE Computer Society Press*, 156–165.

A. Vetterli and J. Kovacevic [1995]. *Wavelets and Subband Coding*, Prentice–Hall.

A. Viterbi and J. Omura [1974]. “Trellis encoding of memoryless discrete–time sources with a fidelity criterion”, *IEEE Trans. on Information Theory IT–20*, 325–332.

E. Wallace [1990]. “Overview of the jpeg (iso/ccitt) Still Image Compression Standard”, *SPIE Image Processing Algorithms and Techniques 1244*, 220–223.

G. Wallace [1991]. “The JPEG: Still Picture Compression Standard”, *Communications of the ACM 34:4*, 31–44.

H. Wang and N. Moayeri [1992]. “Trellis Coded Vector Quantization”, *IEEE Trans. on Communications*, Vol.28, N.8.

S. Wang, A. Sekey, and A. Gersho [1992]. “An objective measure for predicting subjective quality of speech coders”, *IEEE J. Selected Areas in Communications, vol. 10* 819–829.

M. Weinberger and G. Seroussi [1999]. “From LOCO–I to the JPEG-LS Standard”, *Technical Report*, Information Theory Group, HP Laboratories Palo Alto, HPL-1999-3, Jan 1999.

M. Weinberger, G. Seroussi, G. Sapiro [1996]. "LOCO–I: A Low Complexity, Context–Based, Lossless Image Compression Algorithm", *Proc. Data Compression Conference, IEEE Computer Society Press*, 140–149.

M. Weinberger, J. Ziv, and A. Lempel [1991]. "On the Optimal Asymptotic Performance of Universal Ordering and Discrimination of Individual Sequences", *Proc. Data Compression Conference*, IEEE Computer Society Press, 239–246.

J. Wen and J. Villasenor [1998]. "Reversible Variable Length Codes for Efficient and Robust Image and Video Coding", *Proc. Data Compression Conference,* IEEE Computer Society Press, 471–480.

S. Wenger, G. Côté, M. Gallant and F. Kossentini [1999]. "Video Codec Test Model, Near–Term, Version 11 (TMN–11) Rev.2", *Doc. ITU–T/SG16/Q15–G–16 rev 2*.

T. Wiegand and B. Andrews [1998]. "An Improved H.263 Coder Using Rate–Distortion Optimization", *Doc. ITU–T/SG16/Q15–D–13*.

T. Wiegand, M. Lightstone, D. Mukherjee T. George Campbell and S. K. Mitra [1995]. "Rate–Distortion Optimal Model Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard", *IEEE Trans. on Circuits and Systems for Video Technology*.

D. Wilson, M. Ghanbari [1997]. "Optimisation of Two–Layer SNR Scalability for MPEG–2 Video", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Munich, Germany, 2637–2640.

I. Witten and T. Bell [1991]. "The Zero Frequency Problem: Estimating the Probability of Novel Events in Adaptive Text Compression ", *IEEE Trans. on Information Theory*, 37(4), 1085-1094.

I. Witten, A. Moffat, and T. Bell [1994]. *Managing Gigabytes*, Van Nostrand Reinhold.

I. Witten, R. Neal, and J. Cleary [1987]. "Arithmetic Coding for Data Compression", *Communications of the ACM 30:6*, 520–540.

J. Woods [1991]. *Subband Image Coding*, Kluwer Academic Press.

X. Wu [1990]. "A tree–structured locally optimal vector quantizer", *Proc. Tenth International Conference on Pattern Recognition*, Atlantic City, NJ, 176–181.

X. Wu [1993]. "Globally Optimal Bit Allocation", *Proc. Data Compression Conference, IEEE Computer Society Press*, 22–31.

X. Wu [1996]. "An Algorithm Study on Lossless Image Compression", *Proc. Data Compression Conference, IEEE Computer Society Press*, 150–159.

X. Wu [1996b]. "Lossless Compression of Continuous–Tone Images Via Context Selection, Quantization, and Modeling", *IEEE Trans. on Image Processing*.

X. Wu, K. Barthel and W. Zhang [1998b]. "Piecewise 2D Autoregression for Predictive Image Coding", *International Conference on Image Processing Conference Proc.*, Vol.3.

X. Wu, N. Memon [1996]. CALIC — A Context–based, Adaptive, Lossless Image Codec, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 1890–1893.

X. Wu and N. Memon [1997]. "Context–based, Adaptive, Lossless Image Codec", *IEEE Trans. on Communications*, Vol.45, No.4.

X. Wu, Wai–Kin Choi and N. Memon [1998]. "Lossless Interframe Image Compression via Context Modeling", *Proc. Data Compression Conference,* IEEE Computer Society Press, 378–387.

Y. Ye, D. Schilling, P. Cosman, and H. Ko [2000]. "Symbol dictionary design for the JBIG2 standard", *Proc. Data Compression Conference*, IEEE Computer Society Press, 33–42.

K. Zhang, M. Bober, J. Kittler [1996]. Video Coding Using Affine Motion Compensated Prediction, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, IV: 1978–1981.

J. Ziv and A. Lempel [1977]. "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory*, 23:3, 337–343.

J. Ziv and A. Lempel [1978]. "Compression of Individual Sequences Via Variable–Rate Coding", *IEEE Trans. on Information Theory*, 24:5, 530–536.