MASTER'S THESIS

Hierarchical Coding of High Data Rate Video for
Asynchronous Transfer Mode Networks

by R. Sivarajan
Advisor: P. Narayan

CSHCN M.S. 94-1
(ISR M.S. 94-7)

Abstract

Title of Thesis: Hierarchical Coding of High Data Rate Video for Asynchronous Transfer Mode Networks

Name of degree candidate: Rajesh Sivarajan

Degree and year: Master of Science, 1994

Thesis directed by: Professor Prakash Narayan
Department of Electrical Engineering

High data rate video is an integral part of high-quality multimedia for broadband networks. Owing to the high rate, compression of video information is required for an efficient use of network bandwidth. A hierarchical DCT-based video codec is examined that prioritizes and compresses high data rate video for transmission over ATM networks.

The video codec utilizes intraframe coding by independently processing each frame of the video sequence. The lossless compression part consists of run length coding to exploit zero values in the high frequency DCT coefficients and variable length coding (VLC) to further reduce the bit rate. Three compression schemes are examined: adaptive Huffman, arithmetic coding, and Lempel-Ziv-Welch coding. For the model-based compression algorithms, we study several models to characterize the input bit stream to the VLC: memoryless, and Markov with either fixed orders or orders determined by an order estimator. For the three VLCs in the codec, the best performance was obtained from a combination of a memoryless Huffman codec and two first-order Huffman codecs. Many of the models incorporating memory performed poorly due to the small size of the input files.

Due to the VLC, the output rate of the system is variable; however, since intraframe coding is utilized, rate variations are small. In order to fully utilize available bandwidth, we examine the rate control problem of converting the codec from a variable rate system to a fixed rate system. The rate control problem is formulated as one of constrained minimization, and analyzed for optimal solutions. Algorithms are presented for optimal rate control.

Hierarchical Coding of High Data Rate Video for Asynchronous Transfer Mode Networks

by

Rajesh Sivarajan

Thesis submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1994

Advisory Committee:

Professor Prakash Narayan, Chairman/Advisor
Professor Armand Makowski
Associate Professor Steven Tretter
Associate Professor Thomas Fuja

Dedication

To my parents

Acknowledgements

My deepest gratitude goes to Prof. Prakash Narayan for his guidance and encouragement throughout this research. His advice was significant in not only the direction of the research, but in stimulating interest for the research. I always left our discussions with fresh enthusiasm for the research work.

I also thank Dr. Kou-Hu Tzou and Mr. Sanjai Bhargava from the Image Processing Department at COMSAT Laboratories, Clarksburg, MD, for their help during the preliminary stages of this research. The video codec analyzed here is based on a publication by Dr. Tzou. I am grateful to Dr. Armand Makowski, Dr. Steven Tretter, and Dr. Thomas Fuja for agreeing to be the committee members. I thank them for taking time to review this thesis.

I am grateful to Mr. Sanjeev Khudanpur who helped with many useful discussions and invaluable insights. His willingness to help at the calling is deeply appreciated. My thanks also to Mr. Nol Rananand for many helpful discussions. A special thanks to these two gentlemen for reading an early draft of this thesis and suggesting improvements.

Support for this work was provided by the Institute for Systems Research (ISR), University of Maryland at College Park in form of a graduate fellowship. Computing facilities were provided by the Systems Integration Laboratory and Open Workstation Laboratory of the ISR.

Finally, I must thank my family for their support throughout my educational pursuits. I am eternally grateful.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 B-ISDN and Digital Video

Digital video is an important service providing an impetus for the development of a Broadband Integrated Services Digital Network (B-ISDN), the future network that will accommodate voice, video, and data. Video is an integral part of the concept of multimedia; video conferencing, video-on-demand, broadcast digital video, and HDTV are speculated to have a substantial consumer market.

For high data rate (HDR) motion video [18], efficient transmission as well as a maintenance of very high image quality is desirable. Thus, the video system must utilize compression schemes for bandwidth efficiency and must be robust to channel errors. Due to the real-time nature of video, the options of retransmission and forward error control[1] (FEC) are infeasible. Lossy systems, in which an exact reconstruction of the original data is impossible, are usually used to provide the best compression.

[1]That is, error detection and correction are performed at the receiver.

The high rate transmission is via the Asynchronous Transfer Mode (ATM) [4] over a fiber network. Assuming an optical fiber backbone, ATM networks allow high transmission rates combined with low error probabilities. As discussed below, inherent properties of ATM networks make it an attractive protocol for HDR video transmission.

1.2 Interframe/Intraframe Coding

The expected transmission rate for HDR video is higher than 150 Mb/s. In consideration of this high rate, the coder is kept simple and only *intraframe coding* is considered. Intraframe coding treats each frame of the video sequence separately and takes advantage of spatial correlation within the frame. In contrast, *interframe coding* exploits temporal correlation between frames to achieve compression. Interframe coding as compared with intraframe coding can provide gains in compression ratios up to a factor of 3 [14]. However, in addition to the higher coder complexity needed for interframe coding, uncorrectable errors propagate between frames at the receiver. Although it might appear that this propagation augments error effects, as compared with systems using intraframe coding where error effects are isolated to a single frame, studies have shown the reverse to be true. Errors in coders utilizing intraframe coding are more undesirable because they appear as flashes in the received video sequence; errors in interframe coders are smoother [25].

Thus, for low rate video, where current circuits can process the data with tolerable delay, interframe coding is desirable. Not only does the compression gain improve, the effects of transmission errors are (subjectively) less costly.

The Moving Pictures Expert Group (MPEG) standard, designed for current VCR-like quality at 1.5 Mb/s, utilizes interframe coding by applying motion compensation and interpolation between frames [14]. For HDR video, the high processing delay incurred by applying such complex interframe techniques make it infeasible for real-time video services. The system considered here is solely intraframe in nature.

1.3 Hierarchical Coding

We prioritize the video data and transmit it in a hierarchical (or layered) fashion. This method of transmission can be applied to solve various types of network problems.

For instance, such a pyramidal scheme is useful for applications that have receivers of varying qualities (e.g., in multicasting where a source transmits the same information to a group of receivers). The lower quality receivers use only the higher priority data whereas the higher quality receivers utilize all the priorities. We can think of the lower priority information as being refinements used by the higher quality receivers to enhance the quality of the received image. The advantage of such a system is that the source need not transmit separate information to each class of receivers.

This type of transmission is useful [23],[9] for low-rate image transmission: For applications such as interactive image transmission or browsing in an image database, the receiver can first construct low-resolution versions of the image. If the transmission is not terminated, the encoder can proceed to refine the image to a quality level set by the user.

Another application for hierarchical coding is in achieving *graceful degradation* in quality of service from the network in the event of transmission errors. The separation into layers is now designed to combat channel errors by having different error rates for different layers. We focus on this second framework and apply hierarchical coding to diminish the effects of transmission errors. One method of achieving lower error rates is to use FEC for the higher priority data. For HDR video, the complexity and delay introduced by applying FEC makes it infeasible. As discussed below, the ATM standard accommodates two priorities and provides different error rates without the use of FEC.

1.4 ATM Networks

1.4.1 Introduction

The ATM protocol is a natural evolutionary step given the capabilities of fiber optic networks. Fiber accommodates large capacities (in the gigabit per second range) coupled with very small error rates (typically one erroneous bit for every 10^{12} transmitted bits). See [4] for a lucid treatise on the features of ATM.

The goal of ATM is to integrate voice, video, and data traffic at very high network speeds. ATM simplifies the internal network to exploit the advantages of a fiber subnet by relegating redundant and unnecessary routines to the boundaries of the network. For example, error protection on the information payload is handled end-to-end by the users (i.e., the payload is transparent to the ATM network). A complete set of ATM standards has yet to be written by the ITU-T (International Telecommunications Union-Telecommunications) standardization bureau, formerly known as the CCITT (International Consultative Committee for Telecommunications and Telegraphy); only certain elements are finalized.

In an ATM network, data bits are combined to form fixed-sized packets (referred to as *cells*) of length 53 bytes: 5 bytes for a header field and 48 bytes for the payload. The adoption of fixed-sized packets simplifies switching hardware and thereby improves network speed. The choice for 53 bytes is small enough to allow voice calls without intolerable delay; too large of a delay requires echo cancellation circuitry [4].

The ATM standard applies error detection and error correction for the header field only. Transmission errors resulting in an undelivered cell is referred to as a *cell loss*. Cell loss occurs due to either network congestion or non-correctable bit errors in the cell header. If congestion inside the network go unchecked, internal buffers will overflow resulting in unregulated cell loss. Thus cells are dropped within the network to relieve congestion.

ATM accommodates hierarchical transmission by allowing one bit for priorities: The cell loss priority (CLP) bit in the cell header. Low-priority (LP) cells are initially dropped within the network to avoid catastrophic loss of high-priority (HP) cells. Note that the different error rates for the two priorities are due to errors controlled by the network—namely, the dropping of cells due to network congestion. The (physical) channel error rates are identical for both priorities.

Assuming sufficient (end-to-end) error protection on the payload, the only end-to-end transmission errors are cell losses and cell insertions. Cell insertions occur if cells are incorrectly switched within the network and mistakenly delivered to the wrong address.

1.4.2 Effects of Cell Loss

One lost cell results in 48 bytes—or 384 bits—of lost information. For uncoded video systems, such a loss will have a noticeable effect on the received picture quality: Many consecutive horizontal pixels will be affected. For coded systems, such as those utilizing variable length coding (VLC) schemes, the effect of cell loss is even worse since the decoder loses synchronization with the encoder [24]. That is, the decoder is unable to "track" the steps taken by the encoder. Unless suitably dealt with, a cell loss can result in the erasure of many horizontal lines in the received image. Since the considered video codec employs only intraframe coding, the received erroneous video sequence will have noticeable flashes.

The VLC algorithms considered here are *adaptive* in that the parameters needed to code a source symbol depends on past symbols. These parameters, for example, can be estimates of the source symbol probabilities. To mitigate the effects of code synchronization loss, the VLC can be initialized periodically. However, such an initialization reduces the effectiveness of the coder: Adaptive VLC performance generally improves with increasing source samples. For example, estimates of the source statistics converge (in a suitable sense) to their true values as the number of source samples increases. Since the video system considered in this thesis utilizes intraframe coding, we assume that the VLCs are initialized at the beginning of every frame.

The HP-LP partition takes the human visual system into account to maintain visual quality even with the loss of LP cells. Since ATM standards allow one bit for priorities, a two-level hierarchical scheme is examined. Two levels of priorities are sufficient to provide good protection against the effects of cell loss [24]. A higher number of priorities provide greater control but at the cost of an increase in complexity and delay.

The error rate for the HP cells should be very low to allow the main information to be received correctly. In our analysis, we assume that the HP cells are received without any cell loss and only the LP cells are subject to a non-zero error rate.

1.5 Contributions of Thesis

Digital video is an important application in the future B-ISDN. Since ATM is the mode of choice for B-ISDN, it is worthwhile to study the problems and issues associated with transmitting high data rate video over ATM networks.

We consider the video codec proposed by Tzou in [24],[13] which increases bandwidth efficiency due to significant compression gains, and is robust to ATM cell loss due to hierarchical coding. Our main objective is to study compression techniques to achieve the best compression performance. This type of experimental approach is needed since it is difficult to characterize the input bit stream to the VLCs due to the preprocessing. In addition, we study the problem of rate control to convert the system from variable rate to fixed rate.

The three main contributions of this thesis are

1. *Run length coder.* We optimize the run length coder for the specific type of input data. That is, we exploit the characteristics of the processed data that is relayed to the run length coder.

2. *Lossless compression.* We analyze three widely-used lossless compression schemes. For the model-based algorithms, we consider modeling the source by certain classes of stationary and ergodic sources. We examine the

memoryless model and Markov models that incorporates memory. For the Markovian assumption, we consider the order (i.e., the "memory") to be either fixed or determined by an order estimator.

3. *Rate control.* The rate control scheme presented in [24] is ad hoc in that it is designed based on a set of training images. We provide a general framework for modeling and analyzing adaptive rate control policies (adaptive in the sense of being robust to the video source). We formulate the rate control problem and present algorithms for optimal strategies.

The thesis is organized as follows. In Chapter 2 the video codec—without the rate control mechanism—is presented in detail. Chapter 3 examines the problem of lossless compression. We first discuss some of the main features of the three considered compression algorithms; then we consider the source modeling problem. Chapter 4 is devoted to the rate control problem. The video codec emits a variable rate bit stream owing to the use of variable length coding. This rate variability results in a constant quality of service at the receiver. We first present justifications to convert the codec to a fixed bit rate system and then address the problem of rate control. We formulate the rate control problem as a constrained minimization problem and investigate properties of optimal solutions. Simulation results for compression gains are presented in Chapter 5. The system presented in Chapter 2 was implemented in the c programming language on a Unix platform; Sun Sparcstations were used for the execution. Chapter 6 concludes with a summary and identification of areas requiring further research.

Chapter 2

Description of System

The video codec is based on the discrete cosine transform (DCT) and utilizes run length coding and variable length coding to achieve very high compression gains. The human visual system (HVS) is employed in the codec to further improve performance.

System performance for the codec is measured via simulation studies on grayscale image files[2] with luminance (or Y) components represented by 8 bits per pixel (bpp). For color images, the chrominance information is given by two additional components, the U and V components, also sampled at 8 bpp. The structure of the codec for the UV components is identical to that of the Y component; the only changes are in the system parameters [13]. The codec is also tested on video sequences and viewed in the MPEG format to judge performance on real-time motion video. We begin by describing the encoder (see Fig. 2.1) in detail.

[2]The test images were provided by COMSAT Laboratories, Clarksburg, MD.

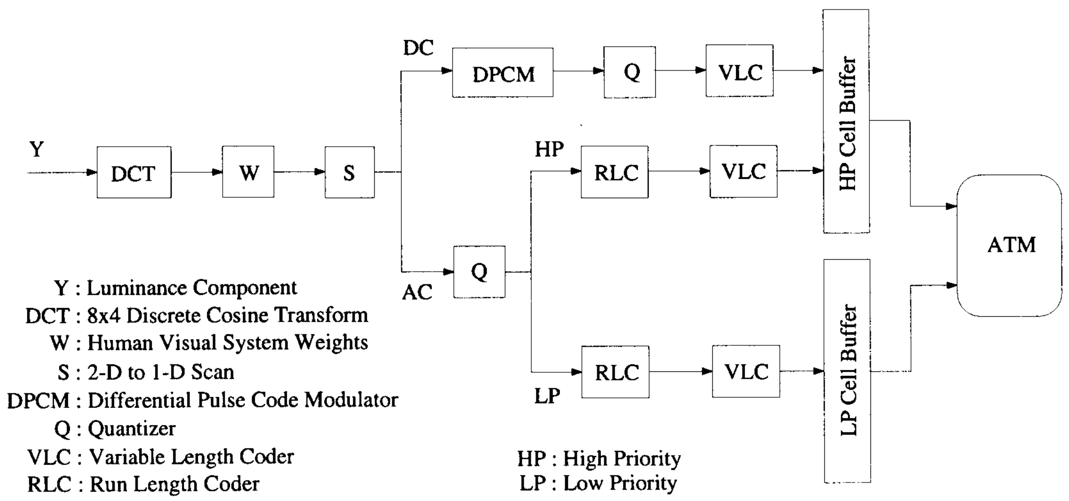


Figure 2.1: Block diagram for encoder.

2.1 Discrete Cosine Transform

The image data is transformed into the frequency domain by the discrete cosine transform [1]. The DCT is well established for image and video processing systems (e.g., two standards for still image compression and low-rate motion video compression are DCT-based). More recently, there has been interest in the wavelet transform [2]. The popularity of the DCT is due to two reasons: (1) The DCT basis set provides good decorrelation for image data; and (2) its computational complexity is low (e.g., the DCT can be implemented using a fast Fourier transform).

The relation between a data sequence $X(i), i = 0, 1, ..., N-1$, and the corresponding DCT sequence $Z(m), m = 0, 1, ..., N-1$ is given by

$$Z(m) = \begin{cases} \frac{\sqrt{2}}{N} \sum_{i=0}^{N-1} X(i), \ m = 0, \\ \frac{2}{N} \sum_{i=0}^{N-1} X(i) \cos\left(\frac{(2i+1)m\pi}{2N}\right), \ m = 1, 2, ..., N-1. \end{cases}$$

In our system, a two-dimensional DCT is applied on 8×4 pixel blocks (i.e., 8 columns by 4 rows). A small block size is chosen to maintain low computational complexity and low system delay. For each 8×4 pixel block, we first take an 8-point DCT along the four rows, then apply a 4-point DCT on the transformed coefficients along the eight columns. Although the aim of the DCT is to decorrelate the image data, the small block size (rather than the original size of the image) rarely results in a data stream that is independent. In addition to correlation between the same (i, j) component among adjacent blocks, correlation might exist within a block (e.g., runs of zeros in the higher frequency components).

2.2 Human Visual Weights and Scan Pattern

The human visual system is more sensitive to low-frequency DCT coefficients. The DCT blocks are appropriately scaled by a weighting matrix to reflect the HVS [17]. This process reduces precision (after quantization) on the less DCT coefficients in accordance with the HVS. Due to the delay requirements for the HDR video, the weights are limited to 1, $\frac{1}{2}$, or $\frac{1}{4}$ so that multiplication can be performed quickly by shifting the binary values. A weighting matrix for the luminance component is provided in [24] (see Table 2.1); the weights were determined based on a set of test images.

For each block, the 2-D to 1-D scan results in a 1-D array of size 32 that prioritizes the coefficients according to the HVS model. Table 2.2 shows the scanning pattern for each DCT block; this pattern corresponded to decreasing signal energy for a set of test images [24].

0.5	1.0	1.0	1.0	0.5	0.5	0.5	0.25
1.0	1.0	1.0	0.5	0.5	0.5	0.25	0.25
1.0	1.0	0.5	0.5	0.5	0.25	0.25	0.25
1.0	0.5	0.5	0.5	0.25	0.25	0.25	0.25

Table 2.1: HVS weight matrix.

1	3	6	7	11	15	19	23
2	8	9	14	18	22	26	29
4	10	13	17	21	25	28	31
5	12	16	20	24	27	30	32

Table 2.2: Scanning pattern for 8×4 DCT coefficients.

2.3 DPCM Codec

The DC component of a DCT block is of crucial importance in the HVS model and is thus treated separately from the AC components. Errors affecting the DC terms produce a "blocking" effect in the received image that makes the 8×4 block boundaries more pronounced.

To represent the DC components efficiently, the DC terms are passed through a Differentially Pulse Coded Modulator (DPCM) and finely quantized. A DPCM outputs the difference between a sample and a predicted value for that sample. Since these *prediction errors* tend to be clustered around zero, a suitable quantizer (one whose quantization levels are clustered around zero) provides an efficient binary representation.

Here, due to the correlation between DC terms in adjacent blocks, the predicted value of a sample is the reconstructed value of the previous sample (see Fig. 2.2). The reconstructed value—rather than the actual sample—is used to avoid propagation of errors in the DPCM decoder.

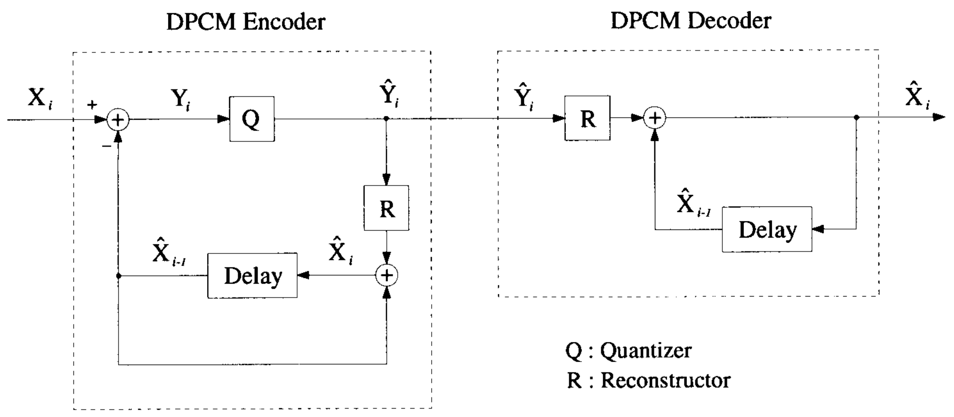


Figure 2.2: Schematic for DPCM.

2.4 Quantization

The resulting prediction errors from the DPCM are quantized by an 8-bit Lloyd-Max [8] modified quantizer based on a Gaussian distribution. It has been shown that the Gaussian distribution is a good model for the DC components of DCT coefficients [21]. Since any linear combinations of Gaussian distributed variables preserves the Gaussian behavior, this is a good choice for the prediction errors. The "modification" from the optimal quantizer is performed because the received video suffers noticeable errors for large distortions (see Fig. 2.3 [24]). The modification is to place additional quantization values for higher input values.

The AC coefficients are quantized by a 7-bit Lloyd-Max modified quantizer based on a Laplacian distribution. It is also shown in [21] that the Laplacian

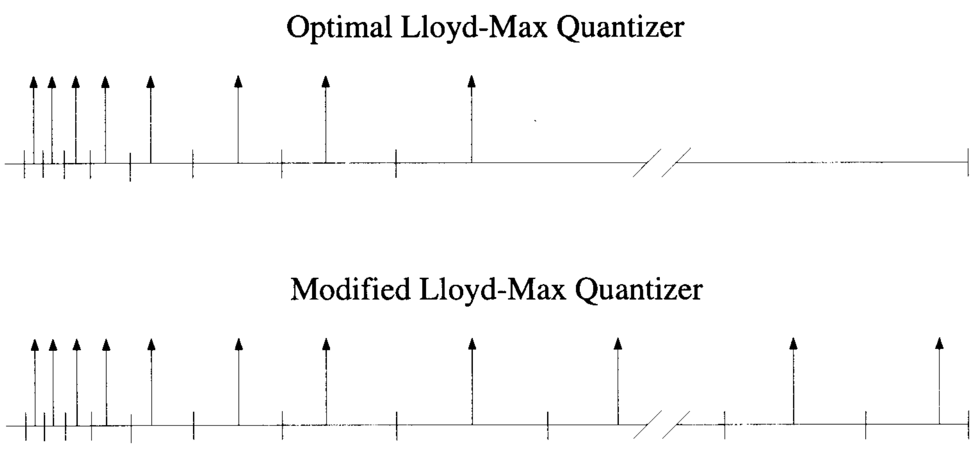


Figure 2.3: Modification to optimal quantizer design.

distribution is a good model for the AC terms of DCT coefficients.

The quantized AC terms are separated into priorities by grouping consecutive elements from the scanned 1-D array. For a two priority case, Fig. 2.4 shows how the 32 quantized terms (AC and DC components) are separated into K_p HP elements and $(32 - K_p)$ LP elements, where $1 \leq K_p \leq 31$. In the following, we will denote the *priority threshold* as K_p.

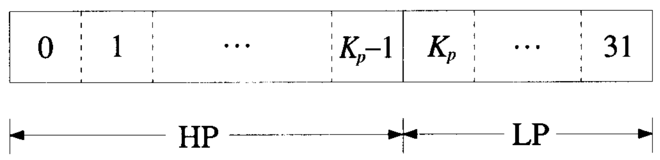


Figure 2.4: Prioritization for 8×4 DCT block.

Note that the choice of intraframe coding allows the prioritization to be based simply on a threshold. For systems utilizing interframe coding, the priority assignment is more complicated. Since interframe coding reduces the spatial correlation in each frame, the high amplitude DCT coefficients are distributed among the entire DCT block. That is, the coefficients no longer satisfy the property that higher frequency components are mostly zero-valued.

2.5 Run Length Coding

In run length coding, the key idea is to group runs of zeros into "meta" symbols. For symbol streams with large runs of zeros, run length coding provides very high compression gains. For a large class of images, the AC components of the DCT coefficients are mostly zeros. This is due to the correlation among components in each 8×4 pixel block and the decorrelating capabilities of the DCT. This property is less pronounced for images such as medical scans and satellite pictures.

We apply run length coding on the quantized values of the HP and LP components of the AC coefficients. If the priority threshold is chosen accordingly, almost all the LP components will be zero. The run length coder (RLC) algorithm used for our system is similar to that of the Joint Photographic Experts Group (JPEG) still image compression standard [20] [27]. Fig. 2.5 shows the format used for the run length coder.

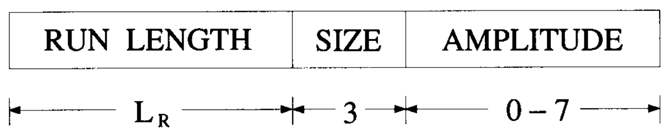


Figure 2.5: Run length coder format.

L_R designates the (fixed) size of each run length word and SIZE is the size of AMPLITUDE in bits. Since the AC coefficients are 7-bit symbols, we fix SIZE to be 3 bits. As mentioned above, the LP component of the AC terms tend to have more zero-valued symbols than the HP component, thus the value of L_R that provides the best RLC performance should be different for the two priorities. The parameter L_R is optimized over a group of test images and the optimum value is used for the system simulation.

2.6 Variable Length Coding

Variable length coding is utilized to losslessly compress the data before transmission. Three lossless techniques are examined: arithmetic coding, Huffman coding, and Lempel-Ziv-Welch (LZW) coding[3]. All three schemes are adaptive in that past data samples determine parameters needed to code each symbol. The compression algorithms are tested assuming a stationary ergodic source [10] that is either memoryless (current symbol independent of all previous symbols), or Markov with varying orders of memory (current symbol dependent on a finite number of past symbols).

As mentioned above, the output symbols from the DCT are rarely independent; further, the RLC does not completely decorrelate the information. Markov models or those with more sustained memories may better capture the (residual) correlation. However, we restrict ourselves to simple memoryless models and Markov models for reasons of analytical tractability and low system complexity. The VLC is examined in detail in Chapter 3.

[3]Computer code for the adaptive Huffman algorithm was generously provided by Sanjeev Khudanpur of the University of Maryland at College Park; the memoryless version of the arithmetic coding algorithm is from [29]; and the Unix command `compress` was used for the LZW algorithm.

2.7 ATM Cell Loss Simulation

Subsequent to the lossless compression, the HP bit stream (representing DC terms and HP AC terms) are assembled into HP ATM cells, and the LP bit stream are assembled into LP ATM cells. Both types of cells are sent to the ATM network.

ATM cell loss is simulated by dropping LP cells according to a fixed probability of cell loss. For each LP cell, a pseudo-random number is generated and the cell is dropped if the number falls within a specified interval. We assume the decoder is cognizant of lost cells. This can be achieved by attaching a sequence number with every LP cell. The HP cells are guaranteed to arrive at the receiver without suffering any cell loss.

The decoder for the system is schematically the reverse of the encoder (see Fig. 2.6). The destination receives the HP and LP bit streams and extracts the video data in real-time.

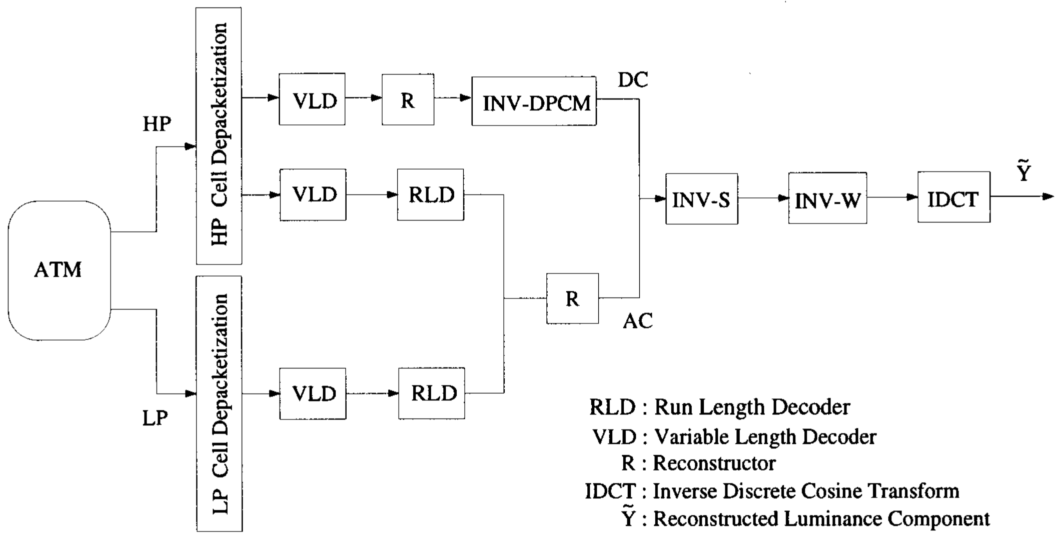


Figure 2.6: Block diagram for decoder.

Chapter 3

Lossless Compression

In this chapter, we focus on the efficient representation of digital data. The coding schemes are "lossless" since an exact reproduction of the original information can be extracted from the compressed version. We examine properties of compression schemes, discuss the three algorithms that are considered in this thesis, and study the source[4] modeling problem.

3.1 Introduction

For a given sequence of source symbols, the VLC removes the redundancy in such a way that the coding process is reversible. The theoretical lower limit on the average codeword length per source symbol is known as the *entropy* of the source [3]. If a coder provides an average codeword length per source symbol that is less than the entropy, an exact reproduction of the original sequence is impossible.

[4]Throughout this chapter, we use the term "source" to mean the source generating the bit stream that is conveyed to the VLC; this is not to be confused with the actual video source.

One way to reduce the redundancy is to represent more probable symbols with a fewer number of bits and less probable symbols with a greater number of bits. Such schemes are *model-based* since a source model is needed to estimate the symbol probabilities (e.g., a memoryless model). Of the three considered compression algorithms, two are model-based—namely, the Huffman coder and the arithmetic coder.

For all compression schemes, a knowledge of the source is required for an efficient code. There are two ways to address this problem when the source statistics are unknown: a two-pass method or adaptive coding. In a two-pass system, the file to be coded is initially scanned to gather the needed source characteristics (e.g., statistics of the source symbols). Usually, these parameters are then fixed during the compression of the file. Disadvantages of this method are increased delay and the overhead needed to transmit the characteristics.

In the adaptive scheme, the source characteristics are gathered in real-time or "near" real-time. In addition to a lower system delay as compared with the two-pass method, the coder can adapt to nonstationarity behavior in the source. The tradeoff to the improved performance is that any transmission error results in a loss of synchronization and thus a loss of the rest of the symbols in the file being decoded. That is, the decoder is unable to track the steps of the encoder due to the loss of synchronization.

3.2 Implemented Algorithms

We briefly discuss the main characteristics of the three compression algorithms that are tested on the system. These algorithms are widely used in many systems that utilize lossless compression. Two of the key characteristics when considering a compression algorithm are the complexity of implementation—and thereby the speed of the algorithm—and the compression performance.

3.2.1 Huffman Coding

See [7] for a description of the adaptive Huffman algorithm. The algorithm works by maintaining a *tree* representation of the source such that more frequent symbols are represented by fewer *branches.* Each branch is assigned a label—a zero or a one for the binary Huffman algorithm. Each source symbol is represented by a unique *codeword*—a concatenation of branch labels—that is used in place of the symbol to achieve compression.

The Huffman algorithm is perhaps the most well known compression scheme. Its popularity is due to simple implementation and its "optimality." If the symbol probabilities are integral multiples of $\frac{1}{2}$, the Huffman algorithm is optimal in the sense that no other lossless compression scheme results in a smaller average codeword length per source symbol [3]. If the source probabilities does not satisfy this property, the Huffman algorithm, encoding symbol-by-symbol, can exceed the entropy by up to one bit per symbol [29]. If groups of K symbols are encoded simultaneously, the suboptimality is by $\frac{1}{K}$ bits.

One disadvantage of the Huffman coder is that, for symbol-by-symbol coding, it assigns at least one bit to each source symbol regardless of how high the symbol probability is. Another disadvantage is the speed as compared with other lossless schemes. The Unix compression program `compact`, an adaptive Huffman coder, was replaced by a faster and more efficient lossless algorithm.

3.2.2 Arithmetic Coding

An introductory tutorial for binary arithmetic coding is presented in [12]. The scheme used in this algorithm is to divide the unit interval [0,1) into regions such that each region corresponds to a source symbol. The codeword for a symbol consists of enough bits to specify a value in that symbol's region. The length of a region is related to the probability of the corresponding symbol. The larger the probability of a symbol, the larger the corresponding region, and the smaller the needed number of bits to represent that region.

Arithmetic coding, like Huffman coding, is model-based since it requires source probabilities. Unlike the Huffman coder, the arithmetic coder can represent a source symbol by *fractional* bits. That is, arithmetic coding does not require an integral number of bits to represent a symbol. This property results in a more efficient code [29].

3.2.3 Lempel-Ziv-Welch Coding

The Lempel-Ziv-Welch (LZW) algorithm [28] is a refinement of the algorithm proposed in [31]. This algorithm uses a dictionary whose entries are source symbols that have occurred previously. Compression is attained by using the index to an entry as the representation for that entry. Note that this scheme does not require any source symbol probabilities.

For stationary ergodic sources, the average codeword length per source symbol of the Lempel-Ziv algorithm converges to the entropy as the number of data samples goes to infinity [3]. Although this is an asymptotic result, the LZW algorithm affords a simple implementation and provides good performance for finite-sized files—especially text files. LZW coding is widely used for many text compression systems (e.g., the Unix `compress` program and the PC compression program `arc`).

The LZW algorithm is popular because of its fast processing speed and good compression performance [3]. One disadvantage is that there is no flexibility to use different source models to improve performance since this algorithm is not model-based.

3.3 Source Models

Of the three compression algorithms considered, the Huffman and the arithmetic coders are model-based and require a source model. The source is usually non-stationary and non-ergodic. However, assumptions of stationarity and ergodicity can sometimes be made over a local time interval if the source doesn't "change" too quickly. The two classes of source models that we consider, memoryless and Markov, are further simplifications to yield statistics in a tractable manner.

For the memoryless assumption, the relative frequency[5] of each source symbol is used as an estimate for the symbol probability. If the source is stationary and ergodic, the relative frequencies converge to the true probabilities (of the memoryless model) in the limit as the number of data samples goes to infinity.

For the Markovian assumption, a knowledge of the order (or the "memory") is needed. However, one method is to compress the file with a predetermined fixed order. The value of this fixed order can be arrived at, for example, by experiments on a set of test images. This approach might not give the best

[5]The number of times a symbol occurs in a sequence of source samples normalized by the total number of samples in the sequence.

performance since the order of the model may change with changing video data. The second approach is to estimate the order of the Markov source based on source samples and use that order for the model. This adaptive method can be applied if the estimation does not introduce too much delay. The problem of order estimation is addressed in [15] and [16]. We highlight the main results for Markov sources.

3.3.1 Probability Estimation Problem

We begin by setting the framework for the problem of estimating the probabilities of the source [15]. The source samples are assumed to be drawn from the alphabet $\mathcal{X} = \{1, ..., r\}$, $r \geq 2$. The unknown order of the source, k^*, is assumed to be fixed in $\{1, ..., k_0\}$, where $k_0 \geq 1$ is assumed known. Let Θ^k denote the set of all $r^{k-1} \times r$ stochastic matrices that generate a stationary and ergodic Markov process. Let $\mathcal{X}^\infty$ be the set of all infinite sequences of symbols drawn from $\mathcal{X}$. For every $\theta \in \Theta^k$, $1 \leq k \leq k_0$, let P_θ denote the stationary and ergodic distribution on $\mathcal{X}^\infty$.

The main problem is to estimate the probability mass function (pmf) of the source based on the source samples $x_1^n = (x_1, x_2, ..., x_n)$ and convey it to the VLC. Given the "true" order, k^*, the "true" pmf of the source, P^*, lies within a family of pmfs indexed by k^* (see Fig. 3.1). Note that there is a problem of ambiguity since measures equivalent to P^* are contained in all families with indices $k > k^*$. That is, the problem of overestimating the order is possible. Although the measures are identical, a higher order increases complexity. This difficulty is circumvented by deleting all such identical measures from higher-order families.

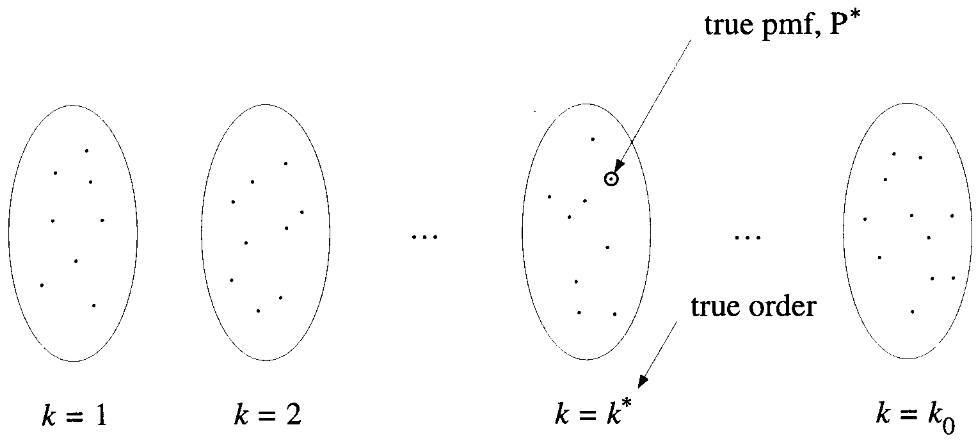


Figure 3.1: Families of probability functions for Markov model.

There are two ways to proceed to estimate the requisite probabilities:

1. First estimate the order; then estimate the pmf within the family indexed by this order.

2. Compute order and pmf in one step via the method of mixtures (to be discussed below).

The maximum likelihood (ML) estimator[6] can be applied for the first method. One of the shortcomings of such an approach is that it is computationally intensive. For each order, all pmfs in the corresponding family must be computed. The maximum of all these functions is then chosen as the ML estimate for the pmf. The delay introduced from such computations is too high for real-time applications. Another disadvantage is that this estimator is *inconsistent* since it overestimates the true order [22].

[6]The ML estimate here maximizes the conditional probability of the samples given the order.

The second method avoids high computational complexity by using one distribution as a *representative* for each family. The computation of this distribution can be performed recursively (as discussed below) to reduce the complexity even further. The mixture distributions are first used to estimate the order; then the mixture distribution that corresponds to the estimated order is used as an estimate for the pmf[7] (see [15] for further details).

3.3.2 Method of Mixtures

For each family of distributions, we *mix* over all the distributions in that family to obtain one representative distribution. To perform this mixing, we assign prior probabilities to all the pmfs. For $k = 1, ..., k_0$, let ν_k be a prior distribution on Θ^k—i.e., $\nu_k(p_1, ..., p_r)$ is the prior distribution that $P_\theta = (p_1, ..., p_r)$, $\theta \in \Theta^k$, is the true pmf.

For source samples $x_1^n \in \mathcal{X}^n$ and $k = 1, ..., k_0$, the mixture distribution is then defined by

$$Q_k(x_1^n) = \int_{\Theta^k} P_\theta(x_1^n)\nu_k(\theta)d\theta \tag{3.1}$$

where $P_\theta(x_1^n)$ is the marginal probability of x_1^n under P_θ. Any prior distribution suffices for our purposes as long as it assigns a positive mass to every measure [15]. We select the Dirichlet distribution [16] which is defined by

$$\nu_k(p_1, ..., p_r) = \frac{p_1^{\alpha_1} \cdots p_r^{\alpha_r}}{C(\alpha_1, ..., \alpha_r)}, \quad \alpha_i > -1, \; i = 1, ..., r, \tag{3.2}$$

[7]In the computer implementation, the mixture distribution is used only to estimate the order. Relative frequencies in the appropriate contexts were used for the actual pmf.

where

$$C(\alpha_1, ..., \alpha_r) = \frac{\Gamma(\alpha_1 + 1) \cdots \Gamma(\alpha_r + 1)}{\Gamma(\sum_{i=1}^{r} \alpha_i + r)}, \tag{3.3}$$

and Γ is the *gamma function*[8]. For our use, we set $\alpha_i = -\frac{1}{2}$, $i = 1, ..., r$.

Given the samples x_1^n, let $v_i \in \mathcal{X}$ denote the *value* of x_i and $c_i \in \mathcal{X}^k$ the (k^{th} order) *context* of x_i in x_1^n, $i = 1, ..., n$. Let $\eta_k(c, v)$ denote the number of occurrences of the symbol $v \in \mathcal{X}$ with context $c \in \mathcal{X}^k$ in x_1^n. Also, let $\eta_k(c) = \sum_{x \in \mathcal{X}} \eta_k(c, x)$ denote the number of occurrences of the context $c \in \mathcal{X}^k$ in x_1^n.

The mixture distribution can then be simplified [16] to

$$Q_k(x_1^n) = Q(x_1) \prod_{i=2}^{n} Q_k(x_i | x_1^{i-1}) \tag{3.4}$$

where

$$Q_k(x_i | x_1^{i-1}) = \left(\frac{\eta_k(c_i, v_i) + \frac{1}{2}}{\eta_k(c_i) + \frac{r}{2}} \right), \tag{3.5}$$

and the initial distribution $Q(x_1)$ assigns the same probability to all symbols. Note that $Q_k(x_1^n)$ can be recursively computed using $Q_k(x_1^{n-1})$ and the realization for x_n. The values of $\eta_k(c, v)$ and $\eta_k(c)$ can also be recursively updated.

[8]The gamma function is defines as

$$\begin{aligned} \Gamma(\alpha) &= \int_0^\infty x^\alpha e^{-x} dx, \\ \Gamma(n) &= (n-1)!, \; n \text{ positive integer.} \end{aligned}$$

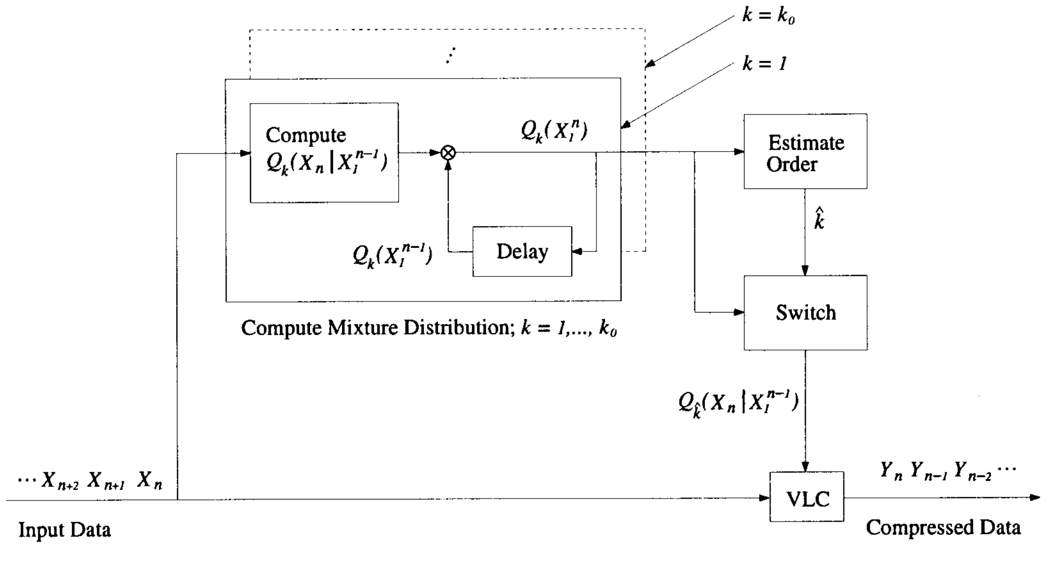
3.3.3 Order Estimator

Given the source samples x_1^n and mixture distribution $Q_k(x_1^n), 1 \leq k \leq k_0$, the order estimate [16] is

$$\hat{k}(x_1^n) = \max_{1 \leq k \leq k_0} \left\{ k : \log_2 Q_k(x_1^n) - \log_2 Q_{k-1}(x_1^n) > \frac{r^{k-1}(r-1)+5}{2} \log_2 n \right\}$$

with $Q_0(x_1^n) = 1 \; \forall x_1^n \in \mathcal{X}^n$. This estimator is *consistent* (i.e., $\lim_{n\to\infty} \hat{k}(x_1^n) = k^* \; P_\theta$—*a.s.*) and thus avoids the problem of overestimating the order.

See Fig. 3.2 for a schematic of incorporating this order estimator into the VLC process. For the VLC, $Q_{\hat{k}} = Q_{\hat{k}}(x_n|x_1^{n-1})$ can be used to code the symbol x_n. Such coding can be shown to be optimal in that the average codeword length per source symbol converges to the entropy at the "optimal" rate with n. Note that the system in Fig. 3.2 can be implemented in real-time due to the low-complexity associated with computing $\hat{k}$ and $Q_{\hat{k}}$.



$k = k_0$
$k = 1$
Compute $Q_k(X_n|X_1^{n-1})$
$Q_k(X_1^n)$
$Q_k(X_1^{n-1})$
Delay
Compute Mixture Distribution; $k = 1,\ldots, k_0$
Estimate Order
$\hat{k}$
Switch
$Q_{\hat{k}}(X_n|X_1^{n-1})$
$\cdots X_{n+2} \; X_{n+1} \; X_n$
Input Data
VLC
$Y_n \; Y_{n-1} \; Y_{n-2} \cdots$
Compressed Data

Figure 3.2: Variable length coding with order estimation.

Chapter 4

Rate Control

We have thus far not considered the implications of the real-time nature of motion video on the design of our system. The video codec, as described above, emits a variable rate bit stream owing to the use of variable length coding. One of the attractive features of ATM is the gain (in bandwidth efficiency) that can be achieved by statistically multiplexing variable rate sources [4].

We, however, convert the video codec to a fixed bit rate (FBR) system for two primary reasons. First, the advantage of statistical multiplexing is diminished since rate variations are small. Due to the temporal redundancy among consecutive frames, the codec is not bursty since only intraframe coding is utilized. That is, the bit rate averaged over an entire frame is correlated from frame to frame. A system that is based on interframe coding results in higher rate variations than a system using solely intraframe coding. The second reason to convert the variable rate bit stream to one of constant rate is that a FBR system is simpler for the network. In addition, a FBR system is cost effective in that the allotted bandwidth is more fully utilized.

Accordingly this "traffic shaping" is accomplished by a buffer with a constant output rate that absorbs rate fluctuations. The control on the rate is accomplished by varying the quantizer resolution in response to feedback buffer state information (e.g., buffer occupancy for the previous time interval). The rate control problem thus becomes one of buffer management, since the goal is to maintain the highest possible image quality while avoiding buffer overflow. Note that buffer underflow need not be addressed since a higher picture quality corresponds to a higher bit rate. A desirable feature of the buffer control strategy is that it be adaptive with respect to the source. That is, we wish to avoid strategies that do not change with changing sources.

The rate control strategy presented in Tzou [24] is static and ad hoc in that it is designed based on a set of training images. Though the strategy works well for a large class of test images, it is worthwhile to investigate adaptive rate control systems. Unlike the other components of the video codec that are also based on training sets (e.g., the run length codec), an adaptive rate control system is amenable to analysis. We first examine some of the approaches to the rate control problem in the literature. We then formulate the rate control problem as one of constrained minimization and investigate some properties of the optimal solutions.

4.1 Previous Approaches

The framework for our rate control mechanism is set in [6], which presents rate-distortion performance for memoryless sources. The decision rule (or buffer management scheme) in [6] is what we refer to as *first-order* since only one past buffer state (i.e., buffer occupancy) is used in the algorithm for each quantizer choice. The performance of a rate control scheme can do no worse by relying on more past buffer states. For most cases, we expect some improvement; however, the growth in computational complexity can be prohibitive. Below, in addition to first-order decision rules, we also examine second-order rules that utilize the past two buffer states for each time instant.

In [11] a finite number of past buffer states are used for the decision rule. The past states are, however, summed and the result is used as input to the decision function. In our analysis, for second-order decision rules, the decision function explicitly depends on two past buffer states.

A first-order buffer control scheme that is transparent to the source is compared in [30] with two others that utilize source characteristics. One of the algorithms that exploits source information is similar to that presented in [24]. These schemes are static since a training set is used to select the parameters of the system.

The rate control problem is examined in [19] in a deterministic setting. The system analyzed does not incorporate variable length coding. The corresponding optimization problem becomes one of integer programming and the optimal solution can be found by dynamic programming via the Viterbi algorithm [19]. Since the Viterbi algorithm can introduce large delays, the system studied is applicable only to non-real-time services (e.g., storing a video sequence to a database or CD-ROM).

Our analysis is closest to [19] but differs in that variable length coding is utilized and thus the rate control problem involves randomness. Due to this aspect, a constraint is placed on the probability of buffer overflow; in [19] the buffer is guaranteed to never overflow. Further, large processing delays are avoided by efficient search algorithms for optimal solutions.

4.2 Problem Formulation

We begin with an informal description of the rate control problem. Our objective is to characterize some of the salient qualitative features of the problem of rate control. A mathematical model is subsequently presented.

4.2.1 Rate Control Problem

For a given threshold $0 < \epsilon < 1$ and $M > 0$, the M-step problem is

$$(P_M) \quad \begin{array}{l} \textbf{minimize } \mathrm{C}_1^{(M)} = \mathrm{E}[\text{sum of distortion for times } 1, 2, ..., M\,] \\ \textbf{subject to } \Pr\{\text{buffer overflow at time } n\,\} < \epsilon, \;\; n = 1, ..., M \end{array}$$

as a function of all rate control decision rules. A suitable choice for M would be a frame boundary. That is, the solution to (P_M) minimizes the cost function over each frame. If the decision rule is allowed to change at each time instant, and the distortion measure is additive, (P_M) reduces to a dynamic programming problem[9]. For such cases, we can examine the problem at each time instant. The 1-step subproblem is

$$(P_1) \quad \begin{array}{l} \textbf{minimize } \mathrm{C}_2^{(n)} = \mathrm{E}[\text{distortion for time } n\,] \\ \textbf{subject to } \Pr\{\text{buffer overflow at time } n\,\} < \epsilon. \end{array}$$

[9]A dynamic programming problem can be reduced to subproblems that have the same features as the main problem.

Note that the solution to the 1-step problem when applied to the entire time window M is a greedy approach in that we use the finest quantizer possible for each time interval. In the case of the M-step problem, it might hold that suffering distortion at some time instants will provide the smallest distortion when summed over M. That is, concatenated local optimal solutions for (P_1) need not lead to a global solution for (P_M).

The problem of interest is (P_M): We seek a solution to the rate control problem for a window of size M. However, the 1-step problem (P_1) is first examined because it is simpler to analyze and it provides insights into the structure of optimal solutions for the M-step problem.

4.2.2 Mathematical Model

See Fig. 4.1 for a block diagram of the rate control scheme. It consists of a bank of K quantizer-VLC pairs of varying resolutions [6]. The outputs of the VLCs are connected to a finite-length buffer of size B. A fixed number of bits from the buffer are forwarded to the network for each time cycle. The choice of which quantizer to use is determined by a decision function ψ, which can be thought of as a switch. The decision function acts on feedback information from the buffer in the form of buffer occupancies.

We use the above system for purposes of analysis but Fig. 4.2 shows a practical realization that uses only one quantizer-VLC pair [6]. The highest resolution quantizer from the set of quantizers in Fig. 4.1 is chosen for this system. Scaling the input data controls the bit rate to the buffer: The highest scale parameter corresponds to the lowest resolution.

To proceed with the analysis of the rate control system in Fig. 4.1, we make

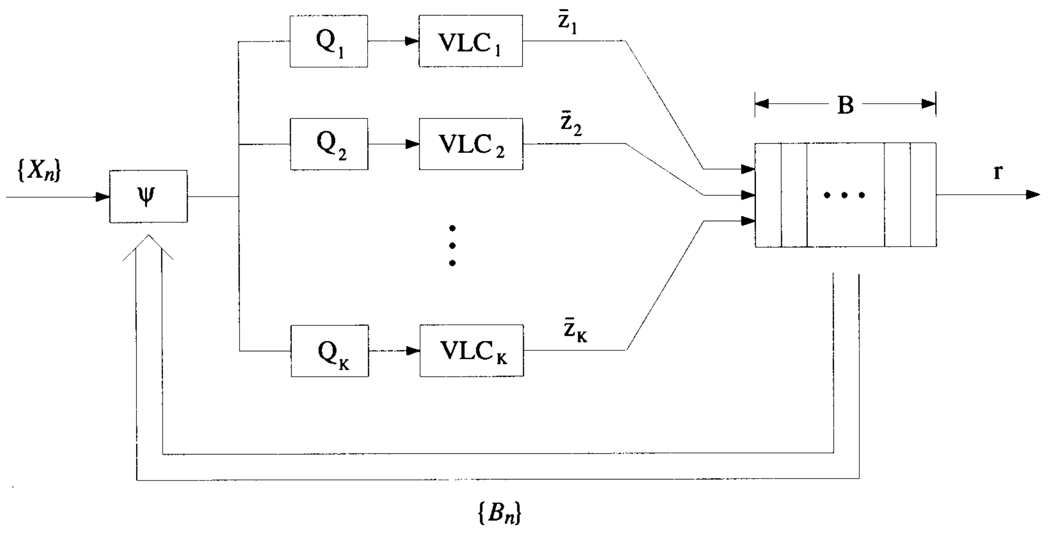


Figure 4.1: Schematic for rate control.

the following two assumptions:

(1) *Memoryless source.* For $n = 1, 2, ...$, the input process $\{X_n\}$ is assumed to be an $\Re$-valued iid random process with known probability density function (pdf) $f_X(\cdot)$. We synchronize the system such that the time between two successive input samples corresponds to a fixed value. The iid assumption is unrealistic, but it is made here, as in [6], for purposes of tractable analysis.

(2) *Fixed quantizers and VLCs.* The optimal quantizers are designed with respect to a given distortion measure $d(\cdot, \cdot)$ and the source pdf $f_X(\cdot)$. (For example, the squared Euclidean distance, $d(x, y) = (y - x)^2$, is traditionally used for tractable analysis.) The quantizers are assumed to be N-level entropy-constrained quantizers (i.e., the quantizer has as its output N bits for each input symbol); see [5] for further details. This choice simplifies the VLC design and also accommodates the simplified system of Fig. 4.2. In practice, the source pdf is unknown and the quantizers are designed as discussed in Chapter 2.

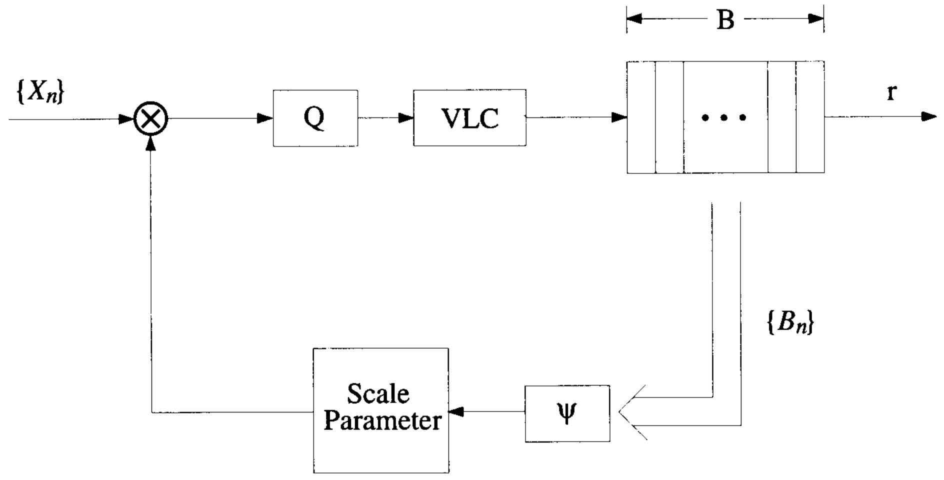


Figure 4.2: Practical rate control system.

The i^{th} quantizer is given by a set of thresholds $\boldsymbol{T}^{(i)} = \{T_0^{(i)}, ..., T_N^{(i)}\}$ and a set of quantization levels $\boldsymbol{Q}^{(i)} = \{q_1^{(i)}, ..., q_N^{(i)}\}$. Then for $x \in \Re$, the action of the i^{th} quantizer is specified by

$$Q_i(x) = \sum_{j=1}^{N} q_j^{(i)} I(T_{j-1}^{(i)} < x \leq T_j^{(i)})$$

where $I(\cdot)$ is the indicator function defined as

$$I(\text{expression}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{expression} = \textit{true} \\ 0, & \textit{else.} \end{cases}$$

The average distortion for the i^{th} quantizer is then given by

$$\begin{aligned} D_i &= \int_{-\infty}^{\infty} d(x, Q_i(x))\ f_X(x)\ dx \\ &= \int_{-\infty}^{\infty} \sum_{j=1}^{N} d(x, q_j^{(i)}) I(T_{j-1}^{(i)} < x \leq T_j^{(i)})\ f_X(x)\ dx \\ &= \sum_{j=1}^{N} \int_{T_{j-1}^{(i)}}^{T_j^{(i)}} d(x, q_j^{(i)})\ f_X(x)\ dx. \end{aligned}$$

Assume the quantizers are ordered such that $D_1 < D_2 < \cdots < D_K$ (i.e., the finest quantizer is Q_1 and the coarsest is Q_K).

The structure of the variable length coders (e.g., Huffman coders) are designed from the quantizer parameters and the source pdf. We represent the length of codewords (in bits) exiting VLC_i by the random variable Z_i, where $Z_i \in \mathcal{Z}_i = \{l_1^{(i)}, ..., l_{N_i}^{(i)}\}$—i.e., a finite set of integers. If an input x to VLC_i results in a codeword Z, we say $\text{VLC}_i(x) = Z$. The pmf describing Z_i is denoted $P_i(l)$, $l \in \mathcal{Z}_i$. Note that $P_i(\cdot)$ is completely specified by the i^{th} quantizer-VLC pair and source pdf $f_X(\cdot)$. Indeed, for any integer l,

$$\begin{aligned} P_i(l) &= \Pr\{Z_i = l\} \\ &= \sum_{\{j \,:\, |\text{VLC}_i(q_j^{(i)})| = l\}} \Pr\{Q_i(X) = q_j^{(i)}\} \\ &= \sum_{\{j \,:\, |\text{VLC}_i(q_j^{(i)})| = l\}} \int_{T_{j-1}^{(i)}}^{T_j^{(i)}} f_X(x)\, dx, \end{aligned}$$

where $|x| \overset{\text{def}}{=}$ length of x in bits.

The feedback process, $\{B_n\}$, is $\mathcal{B}$-valued, where $\mathcal{B} = \{0, 1, ..., B\}$. The value of B_n denotes the buffer occupancy at the end of the n^{th} time cycle. We assume that in the event of buffer overflow, the extra bits are truncated. An alternative is to convey a special symbol whenever the buffer overflows. In that case, bit stuffing needs to be incorporated in to the system to avoid false declarations of buffer overflow. We assume truncation of overflow bits to simplify the analysis.

A general decision map depends on $L \geq 1$ past values of the buffer occupancy and has the form $\psi : \mathcal{B}^L \longmapsto \{1, ..., K\}$. We say such a decision function is L^{th}-ordered. Denote ψ_n as the index of the quantizer-VLC pair used during the n^{th} time interval (i.e., $\psi_n = \psi_n(B_{n-1}, ..., B_{n-L})$). Because of its relative simplicity, we initially analyze the first-order decision function of the form $\psi : \mathcal{B} \longmapsto \{1, ..., K\}$.

Denoting r as the (fixed) output rate of the buffer, we can trace the buffer occupancy over time (see Fig. 4.3). At the end of each time cycle, the buffer occupancy is sampled and is conveyed in the feedback loop to the decision function. The dynamic equation governing the size of the buffer is given by

$$B_n = \min\left\{(B_{n-1} - r + Z_{\psi_n})^+, B\right\}, \tag{4.1}$$

where $(x)^+ \overset{\text{def}}{=} \max\{0, x\}$ and $B_0 = 0$. Since, in general $\psi_n = \psi_n(B_{n-1}, ..., B_{n-L})$, $\{B_n\}$ forms an L^{th}-order discrete-time finite-state time inhomogeneous Markov chain with state-space $\mathcal{B}^L$. The time inhomogeneity is due to the fact that the decision rule can change with time.

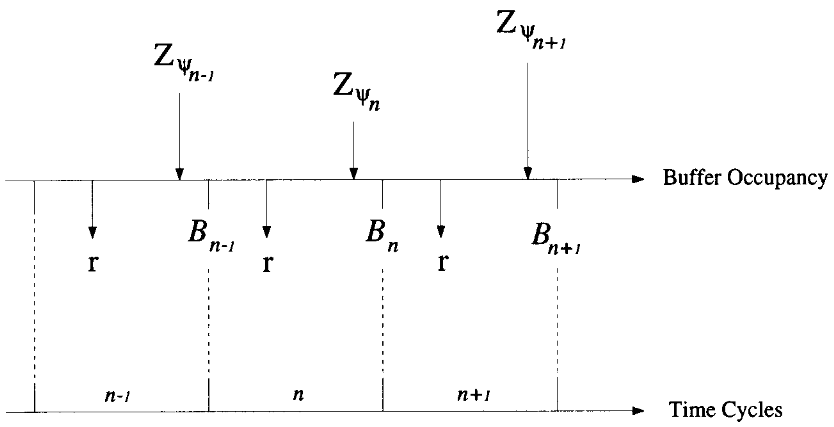


Figure 4.3: Timing pattern for buffer occupancy.

The complexity of the rate control mechanism can be lowered by reducing the state space of the buffer. For example, rather than considering the entire set of possible points $\mathcal{B} = \{0, 1, ..., B\}$, we can consider $\mathcal{B}' = \{0, n, 2n, ..., kn\}$, where $1 \leq n \leq B$ is the resolution reduction parameter and $k = \lfloor \frac{B}{n} \rfloor$. In such a case, the complexity is reduced at the cost of a decrease in the efficiency of buffer usage.

4.3 Cost Function Simplification

In this section, we present simplified forms of the cost functions for both the 1- and M-step problems. We begin by analyzing the costs for $K = 2$ and then extend the results to an arbitrary K.

For $K = 2$, the cost function of (P_1) can be simplified as

$$\begin{aligned}
\mathrm{C}_2^{(n)}\Big|_{K=2} &= \mathrm{E}\left[D_1 I(\psi_n = 1) + D_2 I(\psi_n = 2)\right] \\
&= \mathrm{E}\left[D_1(1 - I(\psi_n = 2)) + D_2 I(\psi_n = 2)\right] \\
&= \mathrm{E}\left[D_1 + (D_2 - D_1) I(\psi_n = 2)\right] \\
&= D_1 + (D_2 - D_1)\Pr\{\psi_n = 2\}
\end{aligned}$$

By assumption, $D_1 < D_2$, thus for $K = 2$

$$\begin{aligned}
&\min \mathrm{C}_2^{(n)} \Longleftrightarrow \min \mathrm{J}_2^{(n)}, \\
&\mathrm{J}_2^{(n)} = \Pr\{\psi_n = 2\}.
\end{aligned} \tag{4.2}$$

The cost function of (P_M) for $K = 2$ can similarly be simplified as

$$\begin{aligned}
\mathrm{C}_1^{(M)}\Big|_{K=2} &= \mathrm{E}\left[\sum_{n=1}^{M} (D_1 I(\psi_n = 1) + D_2 I(\psi_n = 2))\right] \\
&= D_1 + (D_2 - D_1)\sum_{n=1}^{M} \Pr\{\psi_n = 2\}
\end{aligned}$$

Thus for $K = 2$

$$\begin{aligned}
&\min \mathrm{C}_1^{(M)} \Longleftrightarrow \min \mathrm{J}_1^{(M)}, \\
&\mathrm{J}_1^{(M)} = \sum_{n=1}^{M} \Pr\{\psi_n = 2\}.
\end{aligned} \tag{4.3}$$

Note for both problems, the optimal solution maximizes the usage of the finer quantizer subject to satisfying the buffer overflow constraint.

For an arbitrary K, the cost function of (P_M) can be simplified as

$$\mathrm{C}_1^{(M)} = \mathrm{E}\left[\sum_{n=1}^{M}\sum_{k=1}^{K} D_k I(\psi_n = k)\right]$$

$$\begin{aligned} &= \mathrm{E}\left[\sum_{k=1}^{K} D_k \sum_{n=1}^{M} I(\psi_n = k)\right] \\ &= \sum_{k=1}^{K} D_k \left(\sum_{n=1}^{M} \Pr\{\psi_n = k\}\right) \\ &= \sum_{k=1}^{K} D_k \alpha_k^{(M)} \end{aligned} \tag{4.4}$$

where

$$\alpha_k^{(M)} = \sum_{n=1}^{M} \Pr\{\psi_n = k\}. \tag{4.5}$$

And for (P_1), the cost function for $K > 2$ can be reduced as

$$\begin{aligned} \mathrm{C}_2^{(n)} &= \mathrm{E}\left[\sum_{k=1}^{K} D_k I(\psi_n = k)\right] \\ &= \sum_{k=1}^{K} D_k \beta_k^{(n)} \end{aligned} \tag{4.6}$$

where

$$\beta_k^{(n)} = \Pr\{\psi_n = k\}. \tag{4.7}$$

We first present results for first-order decision rules followed by results for higher order rules. For both sections, we start by considering the 1-step problem, then the M-step problem. Also, the rate control scheme for $K = 2$ is used throughout to draw insights into the structure of optimum rules.

4.4 First-Order Decision Rules

4.4.1 Results for 1-Step Problem

We first present a useful property of first-order decision rules for the 1-step problem. This result is used to simplify the optimization problem. Algorithms are then presented to find an optimal solution.

Lemma 1: *For first-order decision rules, the solution to (P_1) is a threshold test.*

Proof: Consider $K = 2$. Let L_1^*, L_2^* be decision regions corresponding to an optimal rule $\psi^*(\cdot)$, where

$$L_1^* \cap L_2^* = \emptyset,$$

$$L_1^* \cup L_2^* = \mathcal{B},$$

$$\psi^*(b) = \begin{cases} 1, & b \in L_1^* \\ 2, & b \in L_2^* \end{cases}, \; b \in \mathcal{B}.$$

It suffices to show that for $b = 0, 1, ..., B-1$

$$b + 1 \in L_1^* \Rightarrow b \in L_1^*.$$

Suppose $b+1 \in L_1^*$ for some $b = 0, 1, ..., B-1$. If using the finer quantizer Q_1 satisfies the buffer overflow constraint when buffer occupancy is $b+1$, then the constraint is certainly satisfied by using Q_1 if buffer occupancy is b. Furthermore, since the cost is decreased by increasing the number of uses of Q_1 (see (4.2)), it follows that the rule must have $b \in L_1^*$.

Thus if $\gamma^* = \max\{b : b \in L_1^*\}$, the optimal rule can be written as a threshold test:

$$\psi^*(b) = \begin{cases} 1, & b \leq \gamma^* \\ 2, & b > \gamma^* \end{cases}, \; b \in \mathcal{B}.$$

For $K > 2$, an optimal rule partitions $\mathcal{B}$ by $L_1^*, ..., L_K^*$. Using similar reasoning as above, it follows that for $k = 1, ..., K$ and $b = 0, ..., B-1$

$$b + 1 \in L_k^* \Rightarrow b \in \bigcup_{j=1}^{k} L_j^*.$$

If we denote $\gamma_k^* = \max\{b : b \in L_k^*\}, k = 1, ..., K-1$, the optimal rule can be written as

$$\psi^*(b) = \begin{cases} 1, & b \leq \gamma_1^* \\ 2, & \gamma_1^* < b \leq \gamma_2^* \\ \vdots & \\ K, & \gamma_{K-1}^* < b \leq B \end{cases}, \quad b \in \mathcal{B}.$$

□

From Lemma 1 and (4.2), (P_1) reduces to

$$(P_1) \qquad \min_{\gamma \in \{0,\ldots,B-1\}} \Pr\{\gamma < B_{n-1} \leq B\} \text{ s.t. } \Pr\{\mathrm{OF}_n \,|\, \gamma\} < \epsilon,$$

where $\Pr\{\mathrm{OF}_n \,|\, \gamma\}$ denotes the probability of buffer overflow at time n given the threshold γ. Since $\Pr\{\gamma < B_{n-1} \leq B\}$ is a non-increasing function of γ, a solution to (P_1) for $K = 2$ is

$$\gamma^* = \max_{\gamma \in \{0,\ldots,B-1\}} \{\gamma : \Pr\{\mathrm{OF}_n \,|\, \gamma\} < \epsilon\}. \tag{4.8}$$

The buffer overflow constraint can be simplified as

$$\begin{aligned} \Pr\{\mathrm{OF}_n \,|\, \gamma\} &= \sum_{b \in \mathcal{B}} \Pr\{\mathrm{OF}_n \,|\, B_{n-1} = b, \gamma\} \Pr\{B_{n-1} = b \,|\, \gamma\} \\ &= \sum_{b \in \mathcal{B}} t_b^{(\gamma)}(n) \Pr\{B_{n-1} = b \,|\, \gamma\} \\ &= \sum_{b \in \mathcal{B}} t_b^{(\gamma)}(n) \Pr\{B_{n-1} = b\} \end{aligned} \tag{4.9}$$

where the last equation holds since B_{n-1} does not depend on the γ being computed for the current time n.

The "transition probabilities" for $b \to \mathrm{OF}$, $b \in \mathcal{B}$, are

$$\begin{aligned} t_b^{(\gamma)}(n) &= \Pr\{\mathrm{OF}_n \,|\, B_{n-1} = b, \gamma\} \\ &= \Pr\{Z_{\psi(b)} > B - b + r \,|\, B_{n-1} = b, \gamma\} \end{aligned}$$

$$\begin{aligned} &= 1 - [\Pr\{Z_1 \leq B - b + r\} I(b \leq \gamma) \\ &\quad + \Pr\{Z_2 \leq B - b + r\} I(b > \gamma)] \\ &= 1 - \sum_{i=0}^{B-b+r} [\mathrm{P}_1(i) I(b \leq \gamma) + \mathrm{P}_2(i) I(b > \gamma)] \end{aligned} \tag{4.10}$$

where the second equation follows from the expression for B_n in (4.1). Since $t_b^{(\gamma)}(n)$ is independent of n (i.e., $t_b^{(\gamma)}(n) \equiv t_b^{(\gamma)}$), we can precompute these values. To reduce computational complexity, we exploit the redundancy in (4.10) to arrive at the following relation:

$$t_{b-1}^{(\gamma)} = t_b^{(\gamma+1)} - \Delta_b^{(\gamma)}, \ \ b = 1, 2, ..., B, \tag{4.11}$$

where

$$\begin{aligned} \Delta_b^{(\gamma)} &= \mathrm{P}_1(B - b + r + 1) I(b \leq \gamma + 1) \\ &\quad + \mathrm{P}_2(B - b + r + 1) I(b > \gamma + 1), \ b = 1, 2, ..., B. \end{aligned} \tag{4.12}$$

Combining (4.10)-(4.12), we arrive at the following algorithm to precompute $t_b^{(\gamma)}$, $b \in \mathcal{B}$, $\forall \gamma \in \{0, ..., B-1\}$.

Algorithm 1: Pre-computation of $t_b^{(\gamma)}$.

Step 1: *Set $\gamma \leftarrow B - 1$.*

Step 2: *Compute $t_b^{(\gamma)}$, $b \in \mathcal{B}$ from (4.10).*

Step 3: *Set $\gamma \leftarrow \gamma - 1$. Compute $\Delta_b^{(\gamma)}$, $b = 1, 2, ..., B$ from (4.12). Set $t_{b-1}^{(\gamma)} \leftarrow t_b^{(\gamma+1)} - \Delta_b^{(\gamma)}$, $b = 1, 2, ..., B$, and compute $t_B^{(\gamma)}$ from (4.10).*

Step 4: *If $\gamma = 0$, stop; otherwise go to Step 3.*

The search algorithm to find the solution to (P_1) is as follows

Algorithm 2: Computation of optimal threshold for (P_1).

```
Step 1: Compute $t_b^{(\gamma)}$ via Alg. 1, $b \in \mathcal{B}$, $\gamma \in \{0, ..., B-1\}$.
Step 2: Set $\gamma \leftarrow B - 1$.
Step 3: Compute $\Pr\{OF_n \mid \gamma\}$ from (4.9).
Step 4: If $\Pr\{OF_n \mid \gamma\} < \epsilon$, stop with $\gamma^* \leftarrow \gamma$.
Step 5: If $\gamma = 0$, stop and print error message ``no solution
        exists.''
Step 6: Set $\gamma \leftarrow \gamma - 1$ and go to Step 3.
```

The main computational complexity in this algorithm is due to the $B+1$ additions and $B+1$ multiplications involved in Step 3 yielding an overall complexity of order $\mathcal{O}(B)$[10]. Although the worst case search involves $B-1$ computations of Step 3, the complexity of Algorithm 2 can be kept close to linear by a suitable choice of the initial value. For example, if we apply the 1-step algorithm M times, we expect the optimal thresholds to be correlated. Rather than starting each run with the initial value $\gamma = B-1$, an improvement in complexity can be achieved by using the optimal threshold from the previous time interval as the initial value for each run.

Other than the precomputed values of $t_b^{(\gamma)}$, $b \in \mathcal{B}$, the only additional information needed for computing the optimal solution to the 1-step problem is $\Pr\{B_{n-1} = b\}, b \in \mathcal{B}$. This information is easily maintained as an array of size $B+1$ and updated each time instant by observing the buffer occupancy.

The application of the 1-step solution for consecutive time instants is depicted in Fig. 4.4. This approach, as mentioned above, leads to a greedy solution. There

[10]That is, the complexity is bounded above by αB, for some constant α.

is no guarantee that this solution is optimal for a window of M time instants. We address the problem of optimizing the first-order rate control mechanism over M time instants in the next section.

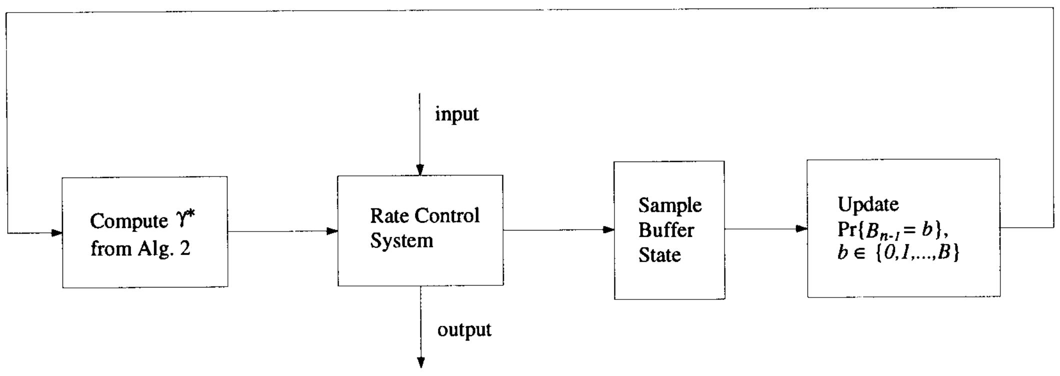

input
Compute Υ^* from Alg. 2
Rate Control System
Sample Buffer State
Update $\Pr\{B_{n-1} = b\}$, $b \in \{0,1,\ldots,B\}$
output

Figure 4.4: Block diagram for consecutive application of 1-step solution.

4.4.2 Results for M-Step Problem

The M-step problem (P_M) can be solved by forward dynamic programming using the Viterbi algorithm [26]. This method is also applied in [19], wherein the problem is simpler owing to its deterministic formulation; consequently, the problem treated in [19] reduces to one of integer programming.

The cost of using the Viterbi algorithm is the introduction of delay. This method is thus suitable for applications such as encoding video onto a medium (e.g., magnetic tape, CD-ROM) for future FBR transmission.

Since we assume a knowledge of the source pdf $f_X(\cdot)$, an alternative method to solve the M-step problem is to use Monte Carlo simulation. The rate control system can be simulated for M time instants to solve for the sequence of optimal thresholds. Then with a certain degree of confidence (i.e., depending on how accurately $f_X(\cdot)$ describes the source), the optimal solution can be utilized.

Although this method avoids the problem of processing delay, in practice a good knowledge of the source is difficult to attain. The method described below—where the buffer statistics are updated at each time instant—is more reliable in limiting the probability of buffer overflow to a tolerable level.

Before presenting the solution via the Viterbi algorithm, we present the framework of the method. The solution is found by tracing viable paths through a trellis (see Fig. 4.5). The states of the trellis are all the possible thresholds $\{0, 1, ..., B-1\}$. A path is said to terminate if using that path violates the buffer overflow constraint of the problem.

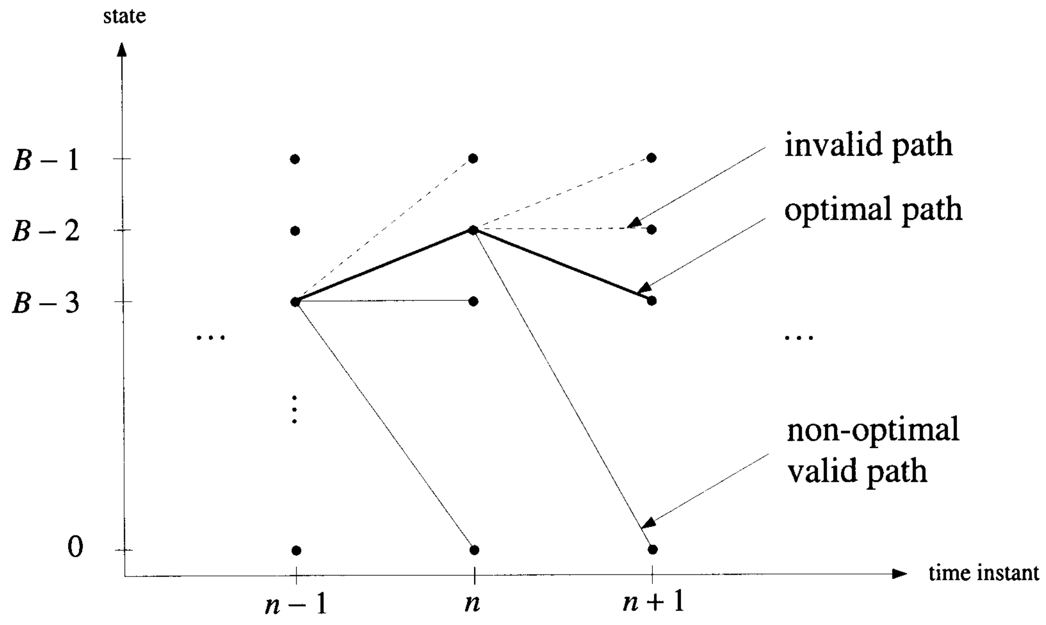


Figure 4.5: Diagram of trellis used in the Viterbi solution.

Let the cost associated with using path p be $c(p)$ and let the probability of buffer overflow using path p be $\lambda(p)$. At each time instant, all surviving paths from the previous instant are extended by adding branches. A branch (i, j) at time instant n is defined as the transition of thresholds $\gamma_{n-1} = i \rightarrow \gamma_n = j$ for

$i, j \in \mathcal{B}$. Adding branch (i, j) to path p at time instant n results in the path p'. This new path is a valid path if and only if $\lambda(p') < \epsilon$; if path p' is invalid, it is terminated.

The solution to the M-step problem can be computed by the following algorithm.

Algorithm 3: Computation of optimal solution for (P_M).

Step 1: *Set* $n \leftarrow 1$, *initialize* $\Pr\{B_0 = b\}$, $b \in \mathcal{B}$, *and initialize the set of valid paths for* $n = 0$ *to the singleton* $p = \{\lfloor \frac{B}{2} \rfloor\}$, *where* $c(p) = \lambda(p) = 0$.

Step 2: *For all valid paths* p, *extend* p *to* p' *by concatenating branch* (i, j), $i, j \in \mathcal{B}$. *If* $\lambda(p') < \epsilon$, *list* p' *as a valid path.*

Step 3: *If* $n < M$, *set* $n \leftarrow n + 1$ *and go to Step 2.*

Step 4: *From all valid paths, select that path corresponding to the smallest cost* $\left(i.e., \, p^* = \arg \min_{p \text{ valid}} \{c(p)\}\right)$. *In the event of a tie, select one at random.*

4.5 Second-Order Decision Rules

By increasing the number of past buffer states in the decision process, we expect some type of improvement. For example, by considering two past values of the buffer state, the trend of the buffer occupancy can be better understood. We study general second-order decision rules that are a function of two variables. The questions that arises are: (1) what type of gains can be achieved by increasing the order of the decision rules; and (2) at what cost?

4.5.1 Results for 1-Step Problem

Since second-order decision rules utilize two past buffer states, the decision regions are two-dimensional. For the case of $K = 2$, the decision regions are

$$R_1 \subseteq \mathcal{B} \times \mathcal{B}$$
$$R_2 = \mathcal{B} \times \mathcal{B} \setminus R_1.$$

The decision function is then

$$\psi(a,b) = \begin{cases} 1, & (a,b) \in R_1 \\ 2, & (a,b) \in R_2 \end{cases}$$

for all $(a,b) \in \mathcal{B} \times \mathcal{B}$.

Similar to the analysis for first-order rules, we wish to minimize (cf. (4.2))

$$\begin{aligned} J_n(R_2) &= \Pr\{\psi_n = 2 | R_2\} \\ &= \Pr\{(B_{n-1}, B_{n-2}) \in R_2 | R_2\}. \end{aligned}$$

To apply the results from the first-order analysis, we condition on B_{n-2}

$$\begin{aligned} J_n(R_2) &= \sum_{b \in \mathcal{B}} \Pr\{(B_{n-1}, b) \in R_2 | B_{n-2} = b, R_2\} \Pr\{B_{n-2} = b | R_2\} \\ &= \sum_{b \in \mathcal{B}} \Pr\{(B_{n-1}, b) \in R_2 | B_{n-2} = b, R_2\} \Pr\{B_{n-2} = b\} \qquad (4.13) \end{aligned}$$

where the last equation holds since the pmf of B_{n-2} does not depend on the current computation of R_2. The minimization of (4.13) is subject to the following constraint on the probability of buffer overflow

$$\Pr\{\mathrm{OF}_n | R_2\} = \sum_{b \in \mathcal{B}} \Pr\{\mathrm{OF}_n | B_{n-2} = b, R_2\} \Pr\{B_{n-2} = b\} < \epsilon \qquad (4.14)$$

where we again use the fact that B_{n-2} does not depend on R_2.

Theorem 1: *The optimal solution, R^*, has the following property: For a fixed $b \in \mathcal{B}$, the decision region along $B_{n-2} = b$, denoted $R^*(b)$, is completely described by*

$$\tau^*(b) = \max_{a \in \mathcal{B}} \{a : a \in R_1^*(b)\}, \tag{4.15}$$

where $R_i^*(b) = \{a : a \in \mathcal{B}, (a,b) \in R_i^*\}, i = 1, 2.$

That is, $R^*(b)$ is a threshold test:

$$\psi(a,b) = \begin{cases} 1, & a \leq \tau^*(b) \\ 2, & a > \tau^*(b) \end{cases}$$

for all $(a,b) \in \mathcal{B} \times \mathcal{B}$.

Proof: Fix $B_{n-2} = b, b \in \mathcal{B}$. By Lemma 1, $R_1^*(b)$ and $R_2^*(b)$ are contiguous intervals. Thus a threshold is sufficient to describe $R^*(b)$, the optimal region corresponding to $B_{n-2} = b$. □

The second-order problem is thus solved by finding a set of optimal thresholds, $R^* = \{\tau^*(b) : b \in \mathcal{B}\}$, that minimize the cost function (cf. (4.13))

$$J_n(R_2) = \sum_{b \in \mathcal{B}} \Pr\{\tau(b) < B_{n-1} \leq B \,|\, B_{n-2} = b\} \Pr\{B_{n-2} = b\} \tag{4.16}$$

while satisfying the buffer overflow constraint in (4.14).

The expression for the probability of buffer overflow can be simplified as follows:

$$\begin{aligned} \Pr\{\mathrm{OF}_n | R_2\} &= \sum_{b \in \mathcal{B}} \Pr\{\mathrm{OF}_n | B_{n-2} = b, \tau(b)\} \Pr\{B_{n-2} = b\} \\ &= \sum_{b \in \mathcal{B}} \sum_{a \in \mathcal{B}} \Pr\{\mathrm{OF}_n | B_{n-1} = a, B_{n-2} = b, \tau(b)\} \times \\ &\qquad \Pr\{B_{n-1} = a | B_{n-2} = b, \tau(b)\} \Pr\{B_{n-2} = b\} \\ &= \sum_{b \in \mathcal{B}} \sum_{a \in \mathcal{B}} \Pr\{\mathrm{OF}_n | B_{n-1} = a, B_{n-2} = b, \tau(b)\} \times \\ &\qquad \Pr\{B_{n-1} = a | B_{n-2} = b\} \Pr\{B_{n-2} = b\} \end{aligned} \tag{4.17}$$

where the last equation uses the fact that B_{n-1} does not depend on R_2. The "transition probability" for $(a, b) \rightarrow \text{OF}$, $(a, b) \in \mathcal{B} \times \mathcal{B}$, can be simplified as in the analysis for first-order rules (cf. (4.10)):

$$\begin{aligned} t_{(a,b)}^{(\tau(b))} &\overset{\text{def}}{=} \Pr\{\text{OF}_n \mid B_{n-1} = a, B_{n-2} = b, \tau(b)\} \\ &= 1 - \sum_{i=0}^{B-a+r} \left[\text{P}_1(i) I(a \leq \tau(b)) + \text{P}_2(i) I(a > \tau(b))\right]. \end{aligned} \tag{4.18}$$

Note that $t_{(a,b)}^{(\tau(b))}$ depends on b only through the threshold $\tau(b)$. Thus we have the relation

$$t_{(a,b)}^{(\tau(b))} = t_a^{(\tau(b))}, \tag{4.19}$$

and $t_a^{(\tau(b))}$ can be precomputed via Algorithm 1. Combining (4.17)-(4.19), we can write the probability of buffer overflow for second-order rules as

$$\Pr\{\text{OF}_n \mid R_2\} = \sum_{b \in \mathcal{B}} \Pr\{B_{n-2} = b\} \sum_{a \in \mathcal{B}} t_a^{(\tau(b))} \Pr\{B_{n-1} = a \mid B_{n-2} = b\}. \tag{4.20}$$

A naive approach to solving for the optimal solution involves a search over a space of size B^B. Thus the total computational complexity is $\Omega(B^B)$[11]. To reduce complexity, we attempt to find relationships among the optimal thresholds.

Proposal: *For $b = 0, 1, ..., B-1$, the optimal thresholds satisfy*

$$\tau^*(b) \leq \tau^*(b+1). \tag{4.21}$$

This relation is heuristically satisfying in that if B_{n-2} decreases, the corresponding optimal threshold should not increase. That is, as the trend increases toward a larger buffer occupancy, the optimal threshold should decrease resulting in a less frequent use of the finer quantizer. The reduction in the total number of search regions is at least of one order.

[11]That is, B^B is a lower bound to the total complexity. For example, if each search for a given decision region involves $\mathcal{O}(B)$ computations, the overall complexity would be $\mathcal{O}(B^{B+1})$.

4.5.2 Results for M-Step Problem

The analysis for second-order decision rules for the M-step problem is similar to that of first-order rules. A greedy approach can be taken by simply applying the 1-step solution for consecutive times instant. For a global minimum over the window $[1, M]$, the Viterbi algorithm can be applied as discussed above. However, the number of states in the trellis now increases to B^2 due to the consideration of two past buffer states. A less complex search algorithm based on Lagrange multipliers is presented in [19] with a slight decrease in performance from the optimal solution.

4.6 Higher-Order Decision Rules

For the simpler 1-step problem, by considering one additional buffer state, the complexity of computing the optimal solution in a naive way increases from $\mathcal{O}(B^2)$ to $\Omega(B^B)$. Although these complexities can be decreased by exploiting characteristics of optimal regions, the difference between them remains very large. Since the input to the rate control system is assumed to be preprocessed to reduce some intersymbol correlation, considering a very high number of past buffer states is not expected to provide significant performance improvements. Indeed the computational costs associated with increasing the order above 2 will most likely outweigh the performance gains.

A Heuristic Extension

One motivation for extending the decision function dependency was that the trends in buffer occupancy could be better captured. We now present a solution

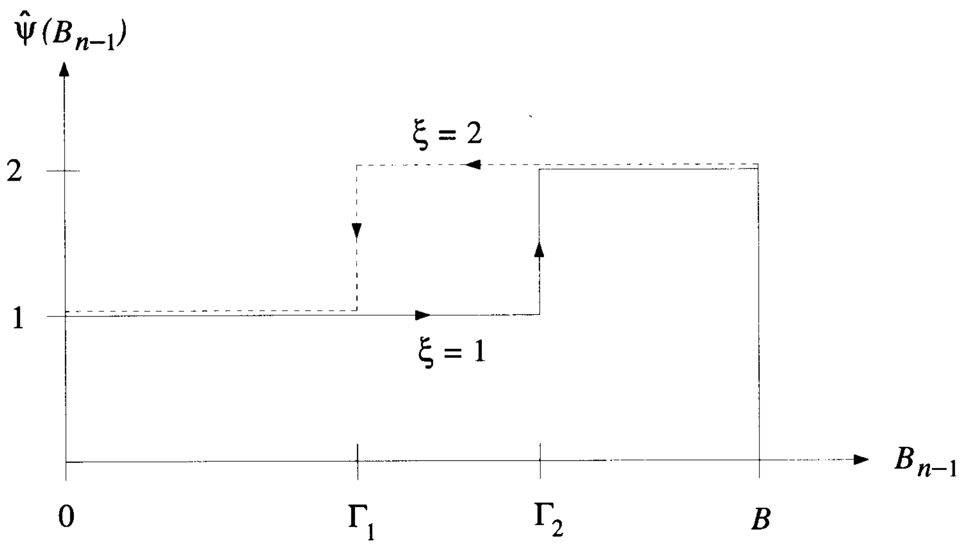


Figure 4.6: Heuristic decision rule for the 1-step problem.

to the 1-step problem with an *arbitrary-order* decision rule that is based on heuristics. This solution uses hysteresis to provide stability for the rate control system. In practice, we wish to avoid frequent changes of quantizer choice.

This simple scheme utilizes a *memory* parameter, ξ, to track the correct hysteresis path. See Fig. 4.6 for a diagram of this decision rule. It is interesting to note that this rule has an arbitrary order. For $B_{n-1} = b$, $b \in \mathcal{B}$, the decision rule is

$$\hat{\psi}(b) = \begin{cases} 1, \text{ if } (0 \leq b \leq \Gamma_1) \textbf{ or } (\Gamma_1 < b \leq \Gamma_2 \text{ and } \xi = 1) \\ 2, \text{ else.} \end{cases}$$

The heuristic rule is given by the following algorithm:

Algorithm 4: Heuristic decision rule.

```
Step 1: Set $n \leftarrow 1$ and $\xi \leftarrow 1$.
Step 2: If $0 \leq B_{n-1} \leq \Gamma_1$, set $\hat{\psi}_n \leftarrow 1$ and $\xi \leftarrow 1$.
Step 3: If $\Gamma_1 < B_{n-1} \leq \Gamma_2$, set $\hat{\psi}_n \leftarrow \xi$.
Step 4: If $\Gamma_2 < B_{n-1} \leq B$, set $\hat{\psi}_n \leftarrow 2$ and $\xi \leftarrow 2$.
Step 5: Set $n \leftarrow n + 1$ and go to Step 2.
```

The thresholds Γ_1 and Γ_2 can be found by solving for γ^* from Algorithm 2 and setting $\Gamma_1 = \gamma^* - \delta_1$ and $\Gamma_2 = \gamma^* + \delta_2$. The choice of the parameters δ_1 and δ_2 depend on the desired level of stability and the corresponding tolerable degradation from the optimal performance.

Chapter 5

Simulation Results

Since the system is intraframe-based, experiments with still images—rather than a sequence of frames—is sufficient to judge performance. The video codec is tested on four 720×576 grayscale image files:

1. *country:* a picture of a hilly European village. The image is characterized by many edges in the foreground.

2. *zelda:* a shot of a woman above the shoulders. This image has a very smooth background and some details in the hair.

3. *girl:* an image of a small girl surrounded by her toys. This picture has many sharp edges and sudden contrast changes.

4. *boats:* a picture of boats tied on a dock. This image is also characterized by sharp edges and many details on the boat in the foreground.

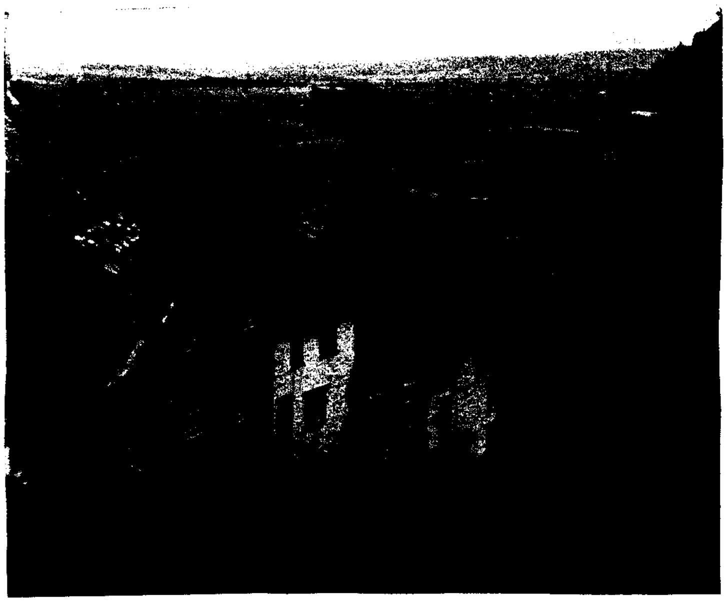
These images are from a set that were used in testing the JPEG standard.

We begin by displaying system robustness to cell loss. Then we examine the compression aspect of the codec: First we consider the run length codec and optimize it based on test images. Then results for the variable length coding methods and various source models are presented.

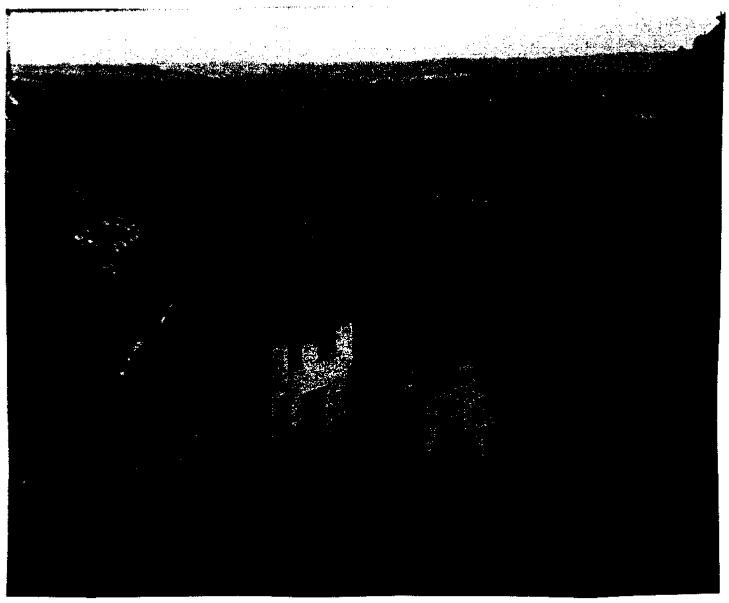
5.1 Robustness to Cell Loss

We first display the effectiveness of the codec to combat cell loss. Since these experiments are subjective, we select the loss of LP cells to correspond to a region where the effects are most noticeable. For comparative reasons, the region affected by cell loss is fixed for each test image (the middle portion of each image). Further, the number of cells dropped is exaggerated to show the distortion effects. In practice, the affected region will consist of a small number of horizontal lines.

Figs. 5.1 and 5.2 show the effects of cell loss on the test image country.Y. The loss of LP cells results in a "blocking" effect. In addition to the original image, corrupted images for various priority thresholds are listed. Note that the effects of cell loss are indistinguishable for a priority threshold of $K_p = 15$ (i.e., the LP component is approximately 50% of the total bit stream). Higher thresholds were not tested for this reason. The same experiments on zelda.Y are shown in Figs. 5.3 and 5.4.

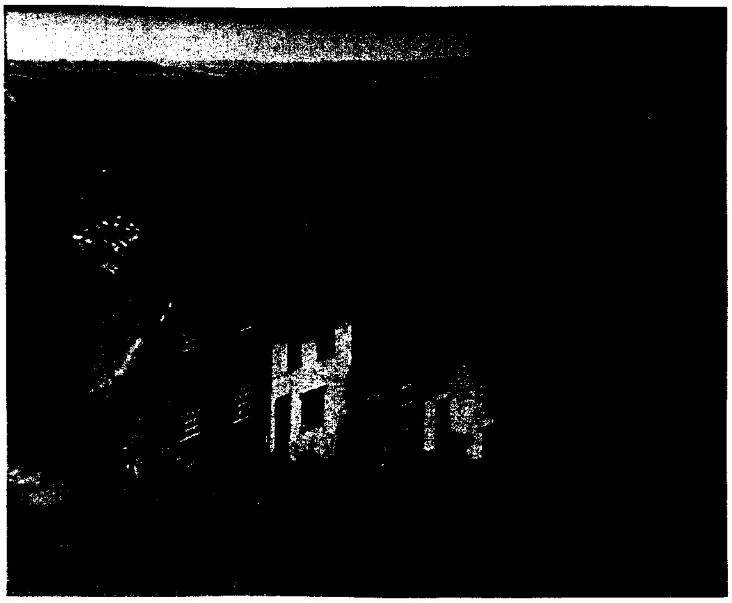


(a) Original image.

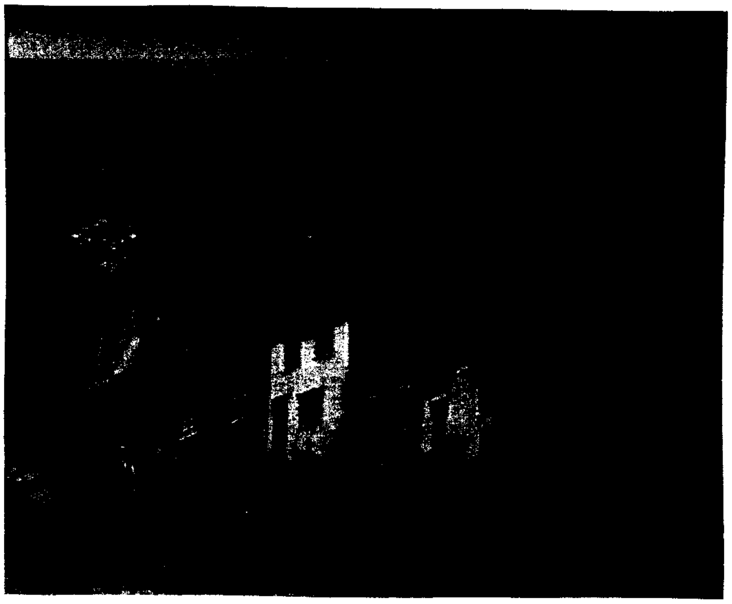


(b) $K_p = 15$.

Figure 5.1: Cell loss effects on country.Y.

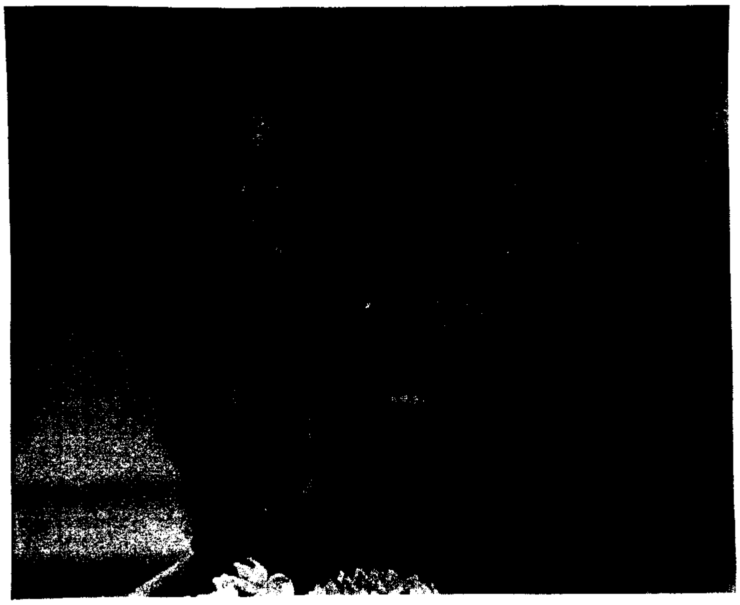


(a) $K_p = 10$.

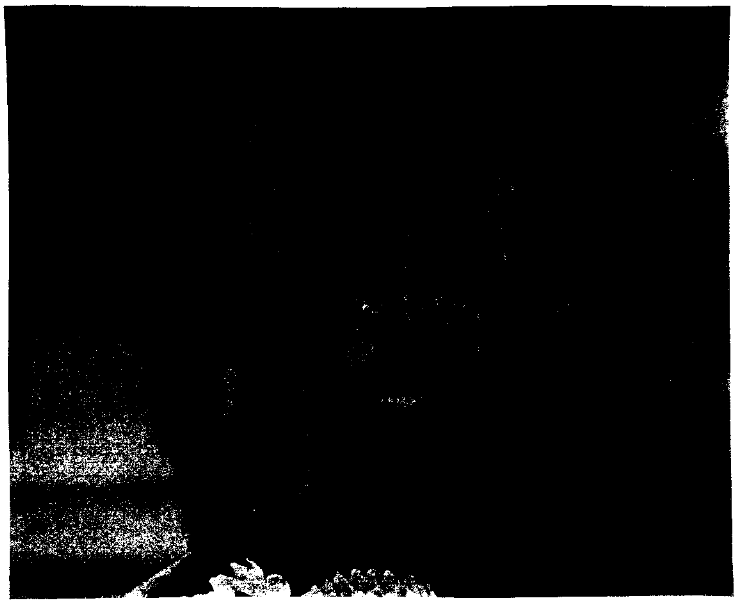


(b) $K_p = 5$.

Figure 5.2: Cell loss effects on country.Y (cont.).



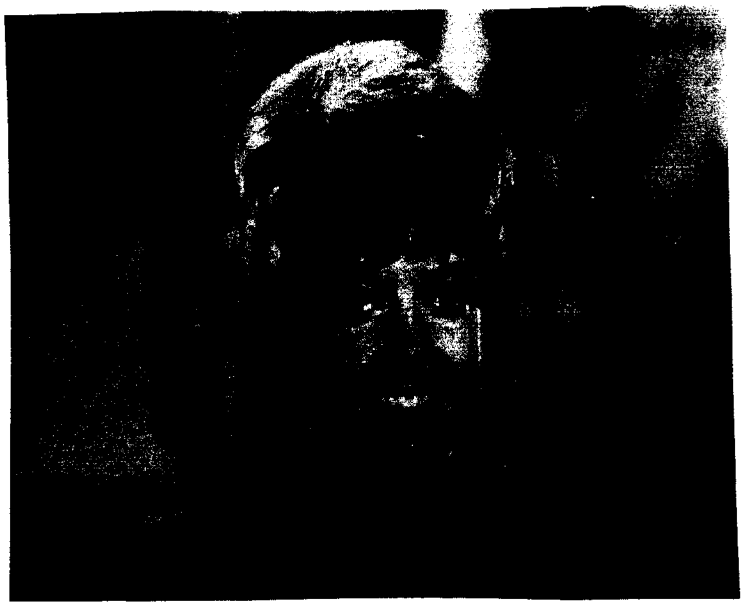
(a) Original image.



(b) $K_p = 15$.

Figure 5.3: Cell loss effects on zelda.Y.

(a) $K_p = 10$.



(b) $K_p = 5$.

Figure 5.4: Cell loss effects on zelda.Y (cont.).

5.2 Compression Performance

5.2.1 Run Length Codec Optimization

We test the RLC performance as a function of L_R, the run length size, and K_p, the priority threshold. Since varying K_p results in different-sized files, a fair comparison is to consider the RLC performance as a function of L_R for a fixed K_p. We expect the LP bit stream to contain many zero-valued symbols and the HP bit stream to have more non-zero-valued symbols. Thus the optimal L_R for each priority class should be different.

High-Priority Bit Stream

See Figs. 5.5 and 5.6 for performance results on the HP part of country.Y and zelda.Y. For all values of K_p considered, the smallest L_R performed the best. This is intuitively satisfying since the HP file will not contain many zero-valued symbols. However, a RLC still provides a compression gain higher than 2:1. Note that the compression behavior for both test images are similar.

See Fig. 5.7 for further tests for finer values of L_R on the HP component of country.Y. Though there are some fluctuations, a value of 4 was chosen as L_R for the HP bit stream for all K_p.

Low-Priority Bit Stream

Figs. 5.8 and 5.9 show compression performance on the LP component of country.Y and zelda.Y. Note the very high gains achieved by run length coding. Although the sub-optimal values of L_R are in disagreement for the two test images, the optimal values are almost identical. Thus for both priority classes, the

RLC performance is similar for different image types.

For the LP case, the optimal L_R is a function of K_p. For low thresholds, the lowest run length considered ($L_R = 4$) is optimal; and for higher thresholds, the middle values of run lengths considered ($L_R = 8$) outperformed the other run lengths. A higher threshold corresponds to an LP file with many runs of zeros, and thus a higher L_R is expected. See Fig. 5.10 for tests of the RLC to refine the optimal value of L_R. We select five as the value of L_R for $K_p \leq 5$ and six for $K_p > 5$.

To examine the overall compressed performance, we plot the absolute sum of the HP and LP compressed components of country.Y in Fig. 5.11. The best values of L_R were chosen from those tested in Figs. 5.5 and 5.8. Note that the total RLC performances are comparable for varying values of K_p, and that slight improvement is noticed for higher values. The tradeoff to high values of K_p is that of a smaller LP component. Although this results in higher-quality images in the event of LP cell loss, bandwidth efficiency is decreased. That is, a higher K_p results in more HP cells causing an increase in the allocation of "guaranteed" bandwidth.

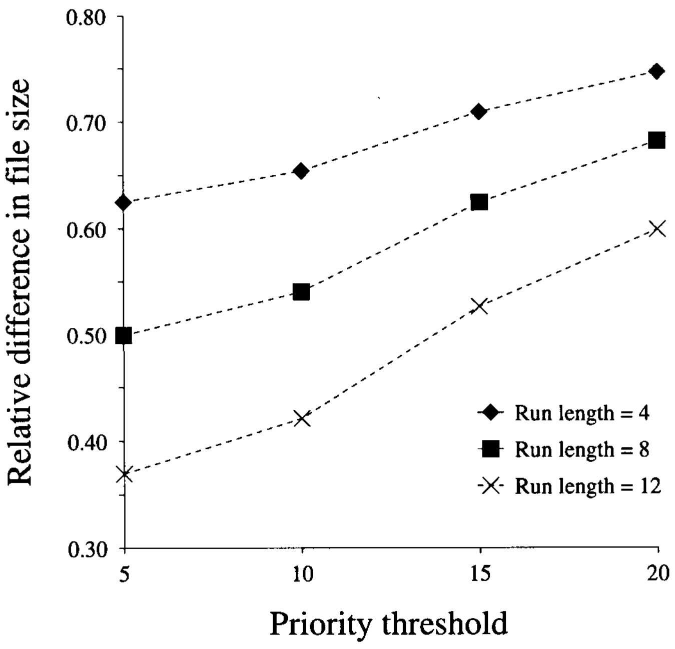


Figure 5.5: RLC performance vs. K_p for HP component of country.Y.

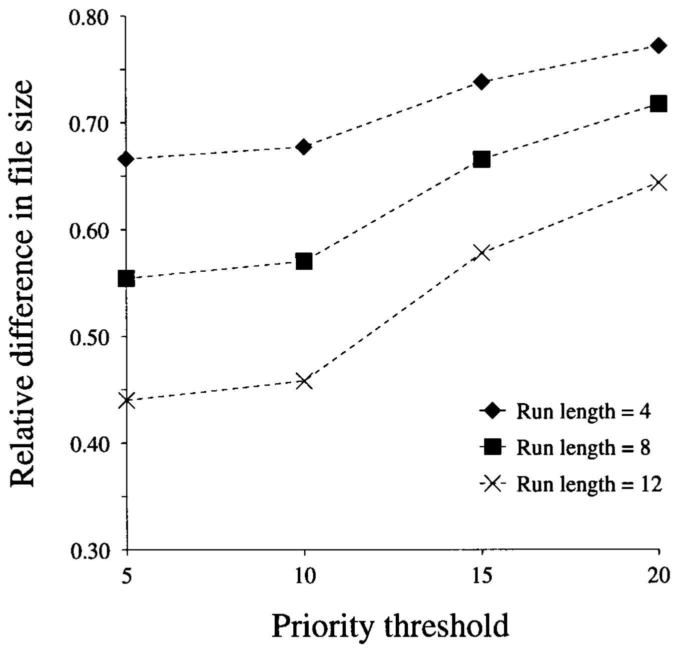


Figure 5.6: RLC performance vs. K_p for HP component of zelda.Y.

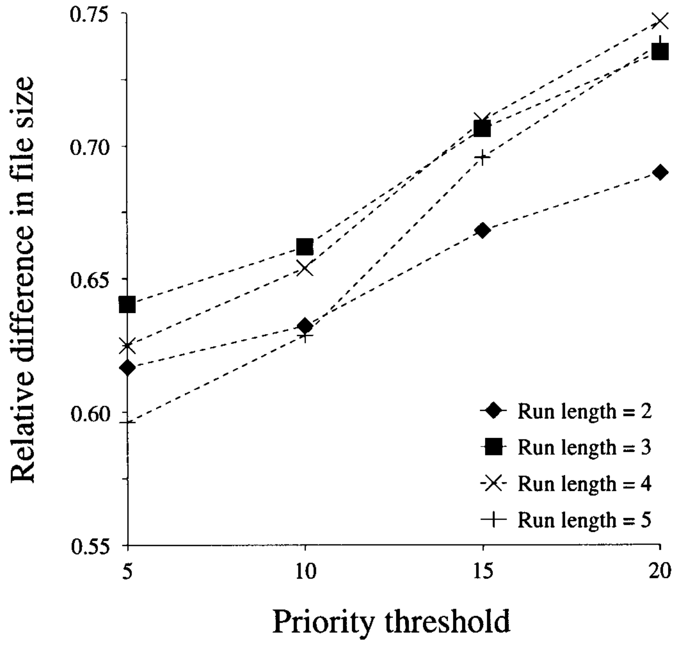


Figure 5.7: RLC performance vs. K_p for HP component of country.Y (fine).

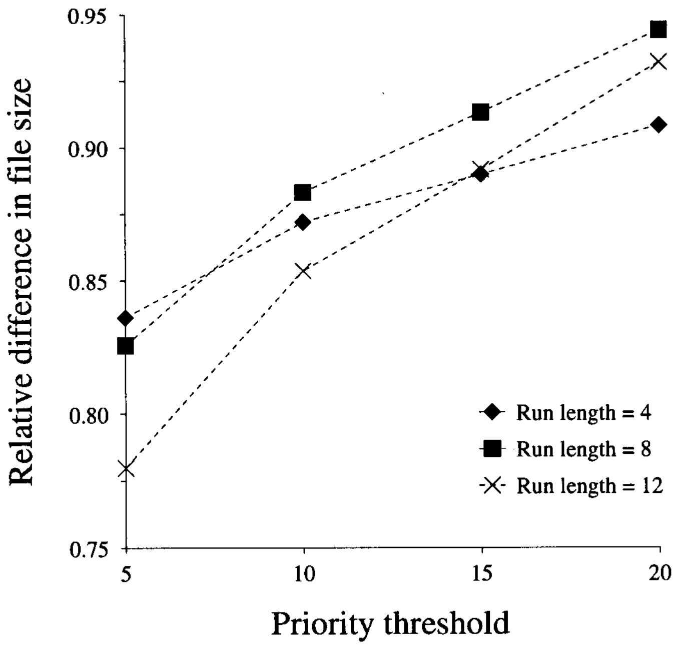


Figure 5.8: RLC performance vs. K_p for LP component of country.Y.

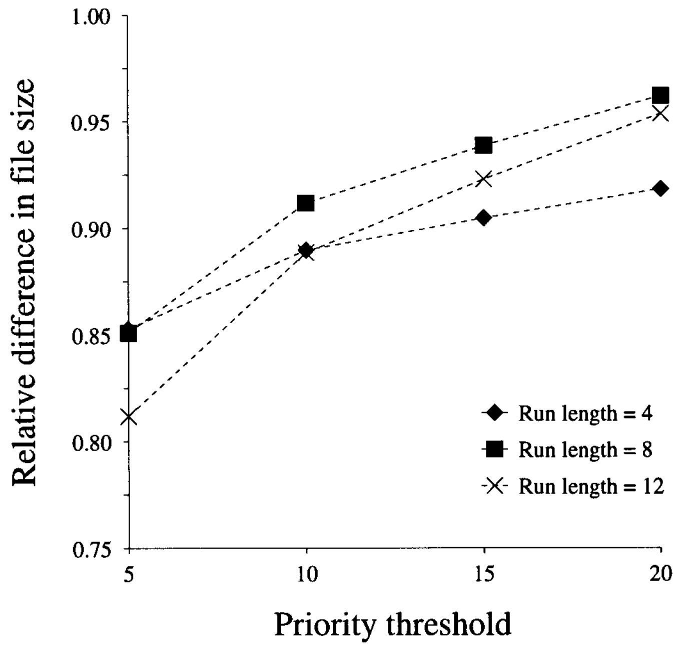


Figure 5.9: RLC performance vs. K_p for LP component of zelda.Y.

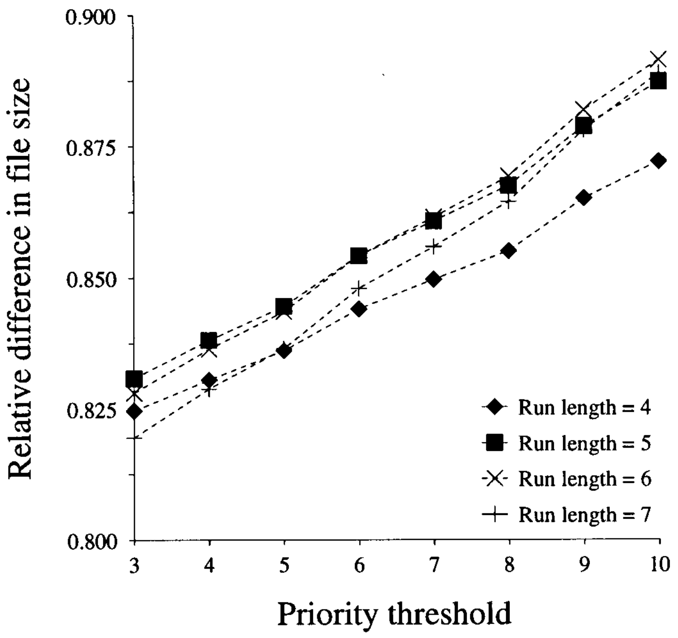


Figure 5.10: RLC performance vs. K_p for LP component of country.Y (fine).

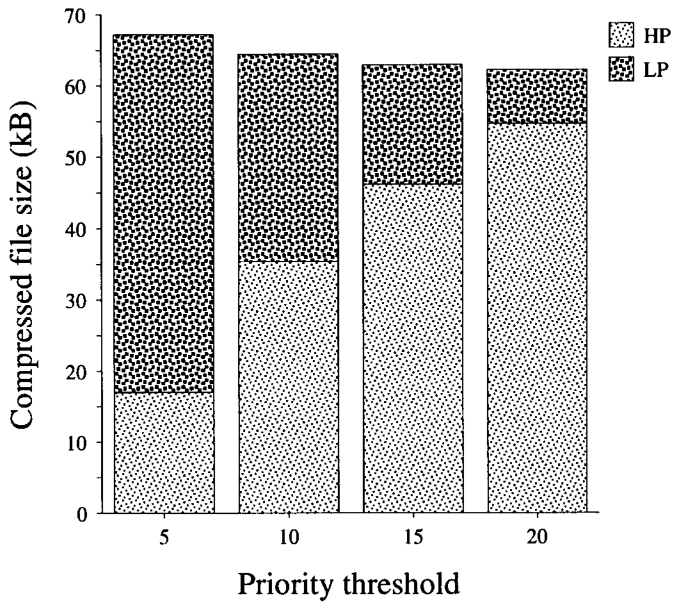


Figure 5.11: RLC performance vs. K_p for HP and LP components of country.Y.

5.2.2 Variable Length Coder Performance

For notation simplicity, Table 5.1 lists abbreviations used for the compression analysis for all source model/compression algorithm combinations. All the algorithms are based on input symbols of length 8 bits. For algorithms based on contexts of length less than 8 bits, the most significant bits of the past symbol(s) were utilized.

Abbr.	**Algorithm**	**Model**
LZW	Lempel-Ziv-Welch	n/a
H0	Huffman	Memoryless model
H4	Huffman	4 bits of prev. symbol as context
H8	Huffman	Prev. symbol as context
A0	Arithmetic coding	Memoryless model
A2-k	Arithmetic coding	2-bit context with order $1 \leq k \leq 6$
A4-k	Arithmetic coding	4-bit context with order $1 \leq k \leq 3$
A8-1	Arithmetic coding	Prev. symbol as context
A2-OE	Arithmetic coding	Markov with order estimator (2-bit context with order $0 \leq k \leq 6$)
A4-OE	Arithmetic coding	Markov with order estimator (4-bit context with order $0 \leq k \leq 3$)

Table 5.1: Abbreviations for compression analysis.

The limit on the maximum order for the arithmetic coding algorithm is for practical reasons. Memory requirements for the software implementation precluded consideration of higher values for the order.

Source Model/Compression Algorithm Comparison

We first analyze the effects of different models on VLC compression performance. In particular, the arithmetic coding algorithm is examined for different (fixed) orders and the order estimation method. A *frequency refresh* scheme was used for all cases. The use of this scheme generally resulted in better compression performance. The goal of this refresh scheme was to weigh the more recent data as more important. The memoryless system actually worked better without the frequency refresh; but for fairness in the comparisons, we state the results of the memoryless model with frequency refresh. The results for the HP and LP components of country.Y[12] are presented in Figs. 5.12–5.15.

For all the files presented, no other model outperformed the memoryless model. The order estimation scheme provided the same compression as the memoryless model for some of the test files. For these cases, the order estimator stayed at order $\hat{k} = 0$ for the duration of the test file.

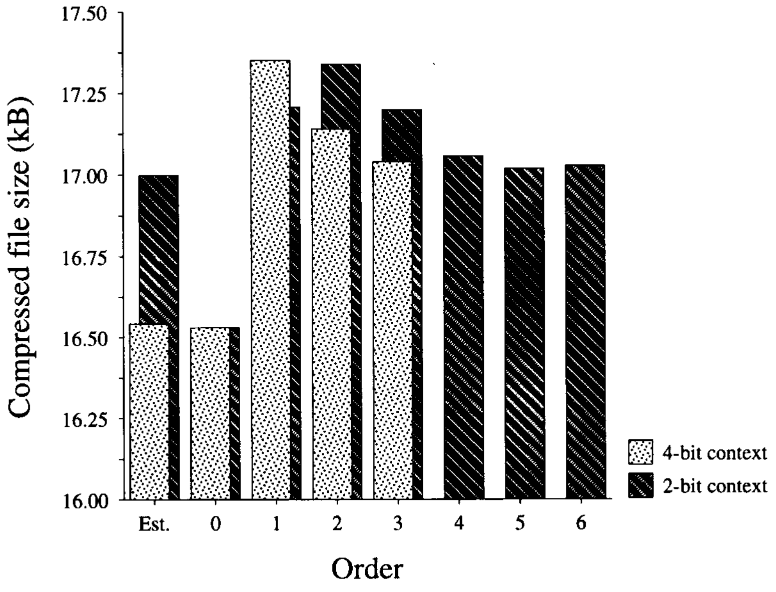
Note a general trend in these figures: Compression is lowest for $k = 0$ then there is a sharp increase for a higher value of k (for the 4-bit context, the worst performance is for $k = 1$; for the 2-bit context system, the worst is at $k = 2$) then a steady decrease for higher values of k. One explanation for the discrepancy in compression behavior for $k = 1$ is that the 4-bit system requires more data to build a good representation of the source. That is, the 2-bit system works better since the corresponding total number of contexts is smaller than for the 4-bit system.

A curious phenomenon is that the 4-bit system outperforms the 2-bit system for $k > 1$. The number of contexts grows exponentially in k and one would

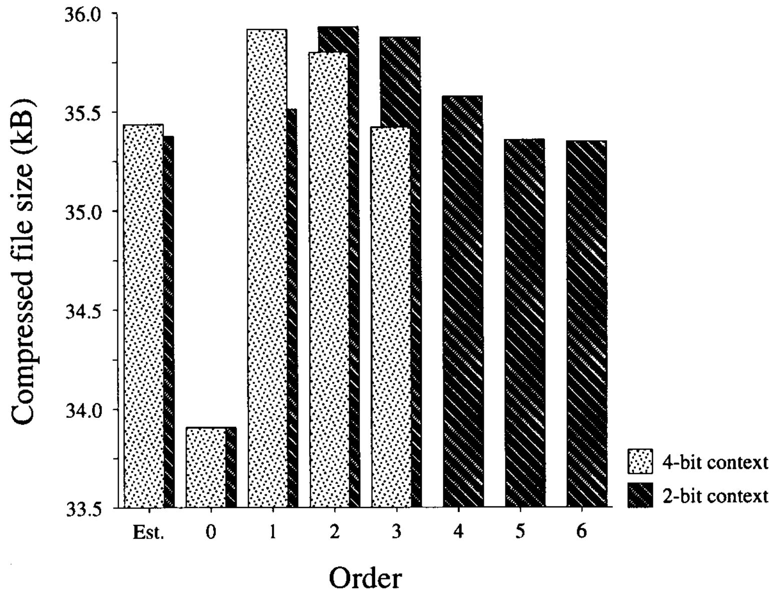
[12]The results for the other test images were similar.

expect that, for the reason given above, the 2-bit system work better than the 4-bit system. However, considering a higher number of most significant bits seems to provide a better representation for the true probabilities.

The ineffectiveness of incorporating the order estimator into the compression algorithm can be primarily attributed to the small size of the test files. The order estimator optimality (i.e., convergence to the true order) is an asymptotic result. The RLC prior to the VLC provided significant compression and thus reduced the effectiveness of the order estimator.

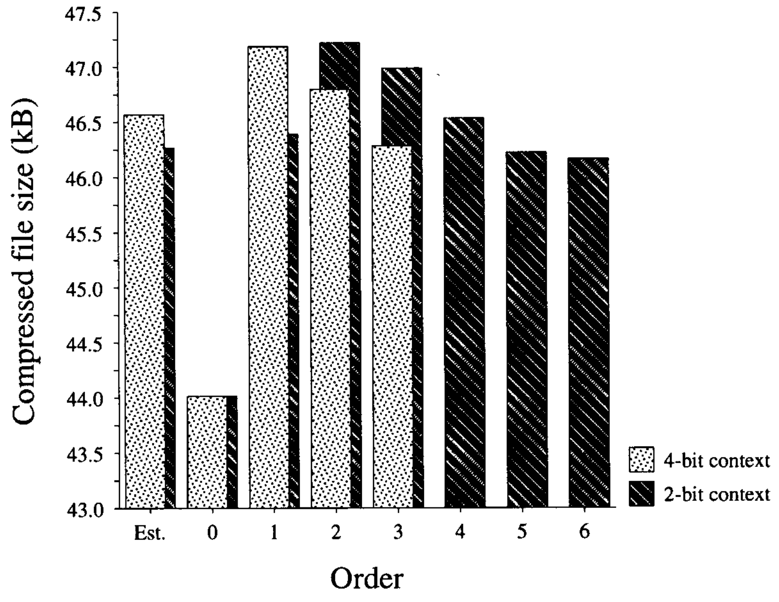


(a) $K_p = 5$.

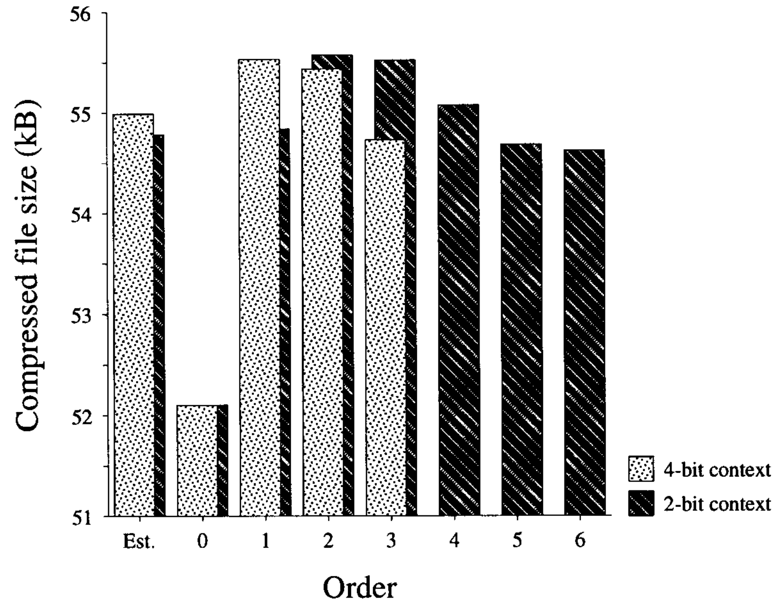


(b) $K_p = 10$.

Figure 5.12: VLC perf. for varying models: HP comp. of country.Y.

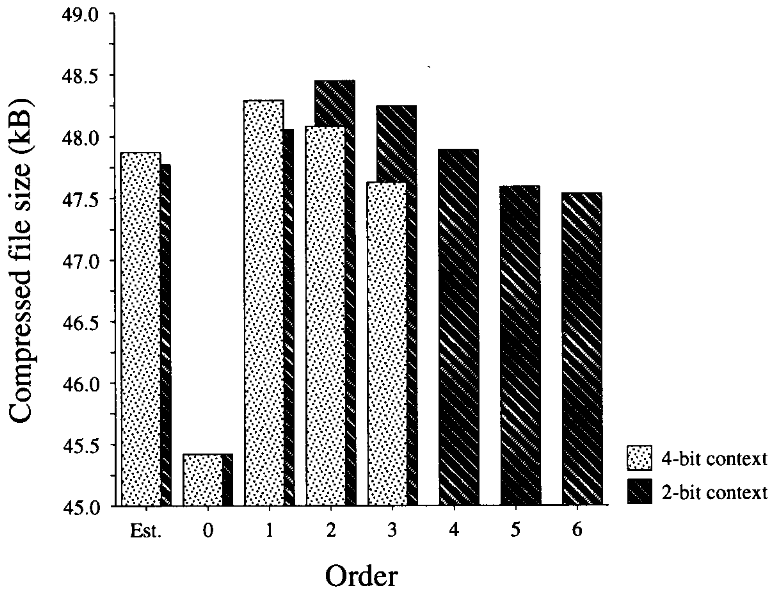


(a) $K_p = 15$.

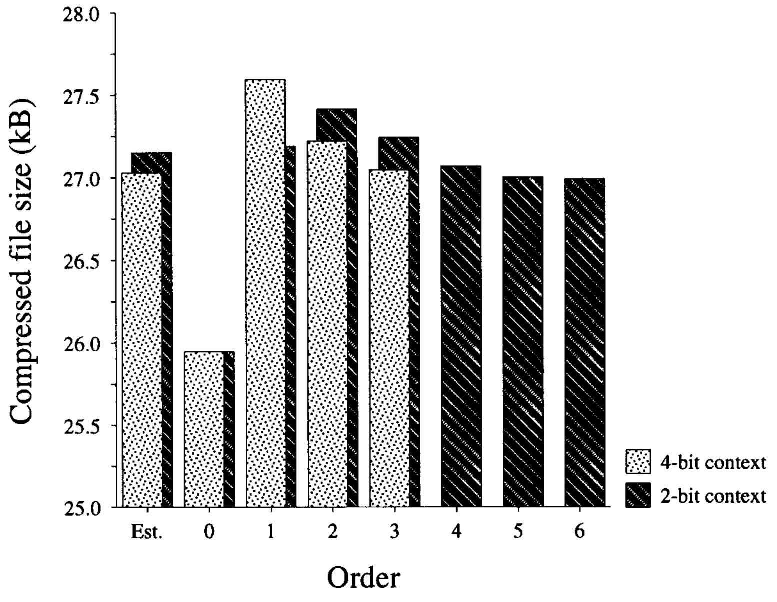


(b) $K_p = 20$.

Figure 5.13: VLC perf. for varying models: HP comp. of country.Y (cont.).

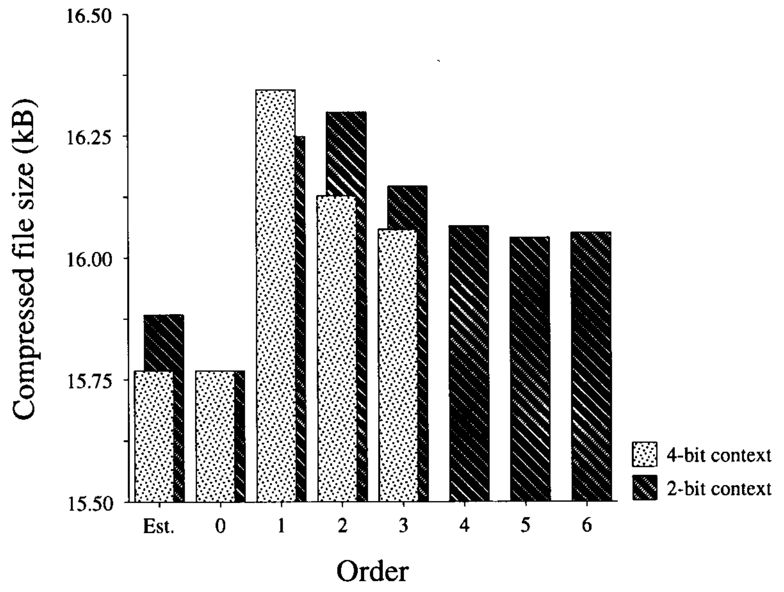


(a) $K_p = 5$.

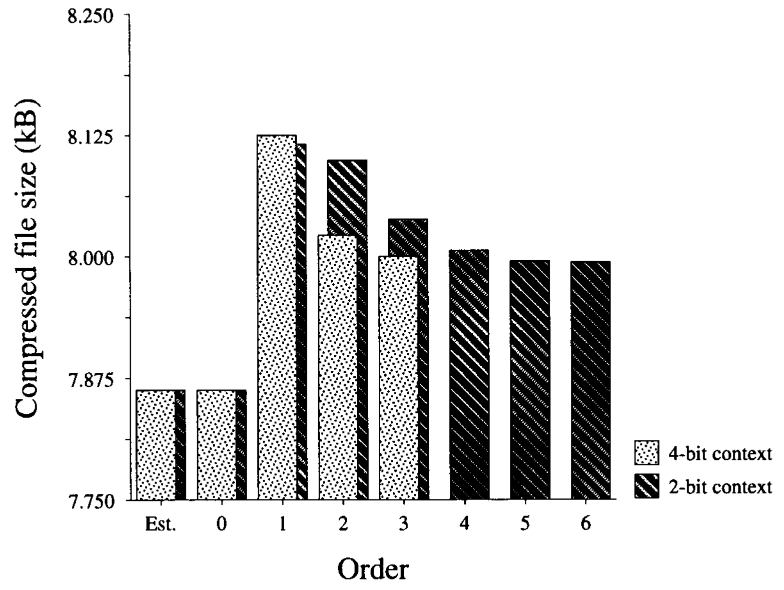


(b) $K_p = 10$.

Figure 5.14: VLC perf. for varying models: LP comp. of country.Y.



(a) $K_p = 15$.



(b) $K_p = 20$.

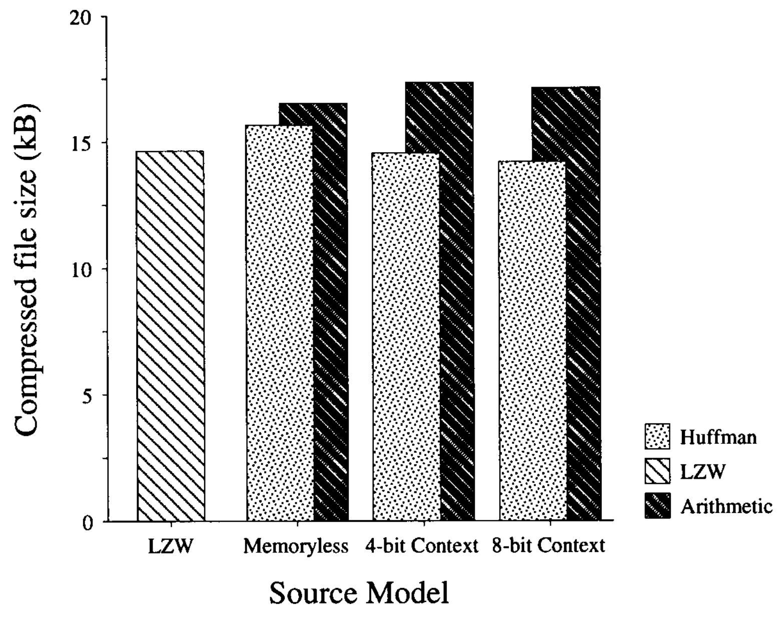
Figure 5.15: VLC perf. for varying models: LP comp. of country.Y (cont.).

Above, we analyzed the effect of different source models for a fixed compression algorithm, namely arithmetic coding. Now we study the performance of the three compression algorithms for a given source model. For the LZW algorithm, which uses parsing and maintenance of a dictionary, there is no source model; however, we state its compression performance for comparative purposes. For the Huffman and arithmetic coding algorithms, results for both the 4-bit context model and the full 8-bit context model are presented. See Figs. 5.16–5.19 for compression results for the HP and LP components of country.Y[13].

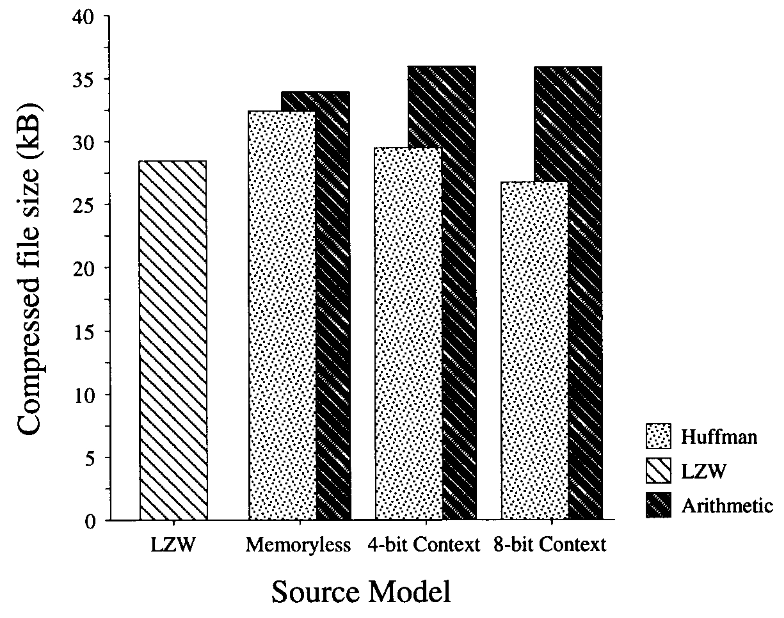
The first observation of these results is the improvement in compression for models with memory when using the Huffman algorithm. There is a steady improvement for both HP and LP files as memory is utilized. In contrast, arithmetic coding performs best for a memoryless model and poorly when incorporating memory. One reason for the decrease in performance for models with memory is that more data is required to build a good estimate of the source statistics. The arithmetic coding algorithm used frequency counts to represent the source statistics; the tree representation of the Huffman algorithm must have provided faster convergence to an accurate estimate of the source statistics.

As a final remark before presenting the best overall compression performance, note that the general behavior of the VLC does not drastically change by varying priority thresholds or test images. This is good news since it says that the system is robust to different source characteristics and the results are not limited to the test images. The key system component that reduce the video characteristics in the data is the run length codec.

[13]Again, the results for the other test images were very similar.

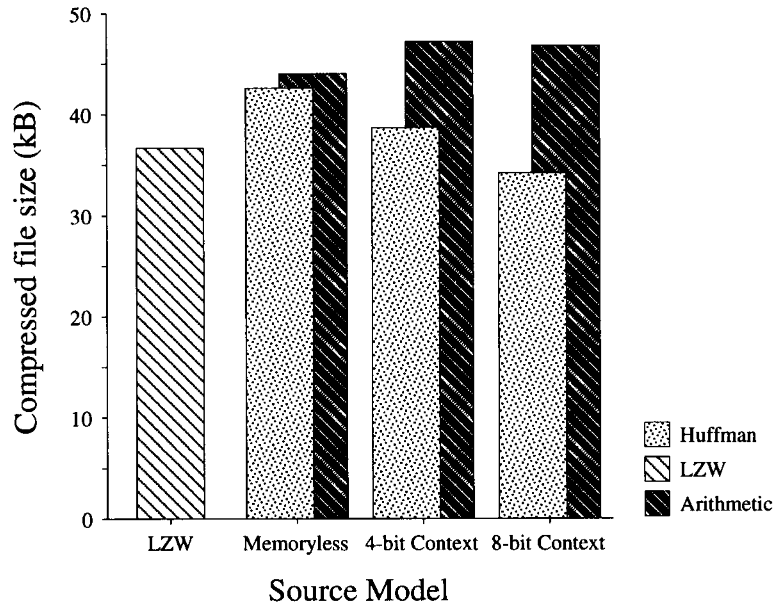


(a) $K_p = 5$.

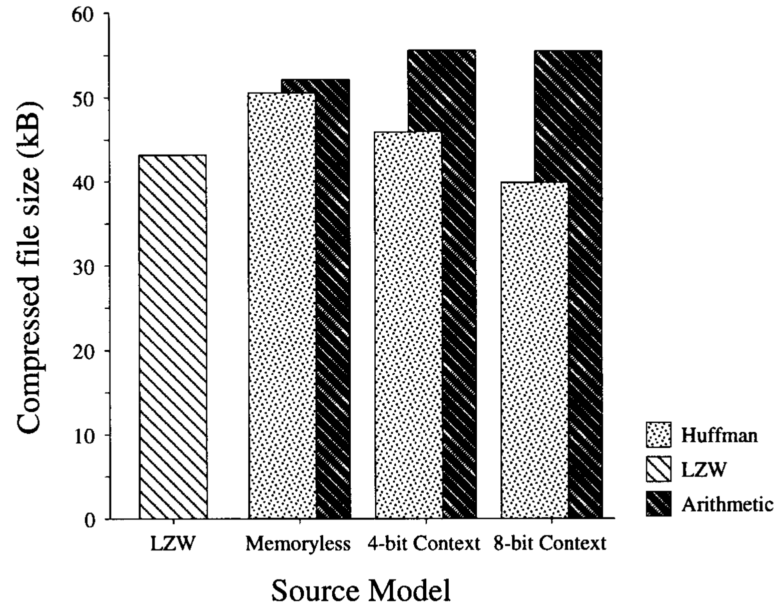


(b) $K_p = 10$.

Figure 5.16: VLC perf. for varying algorithms: HP comp. of country.Y.

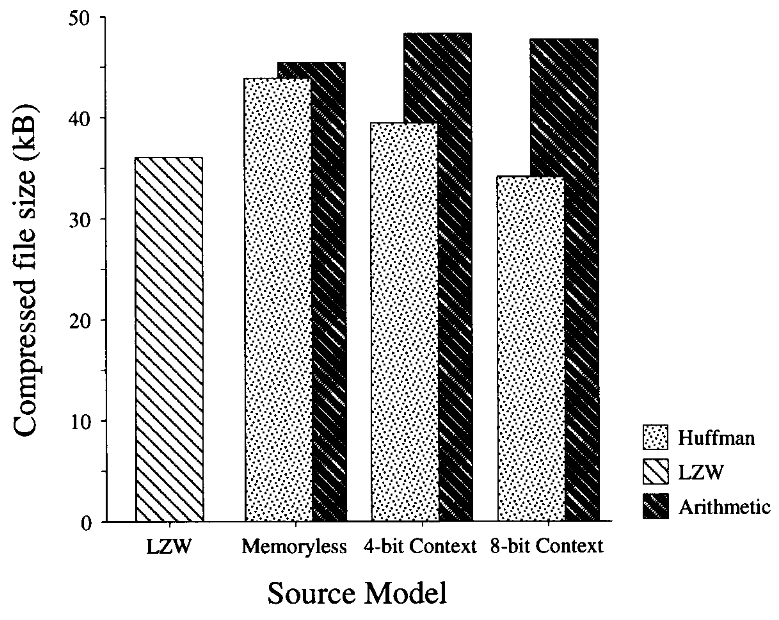


(a) $K_p = 15$.

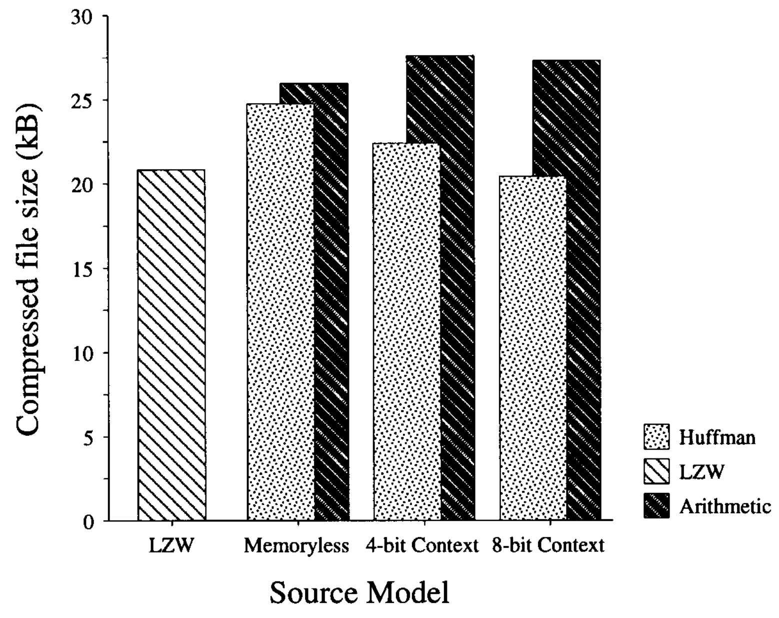


(b) $K_p = 20$.

Figure 5.17: VLC perf. for varying algorithms: HP comp. of country.Y (cont.).

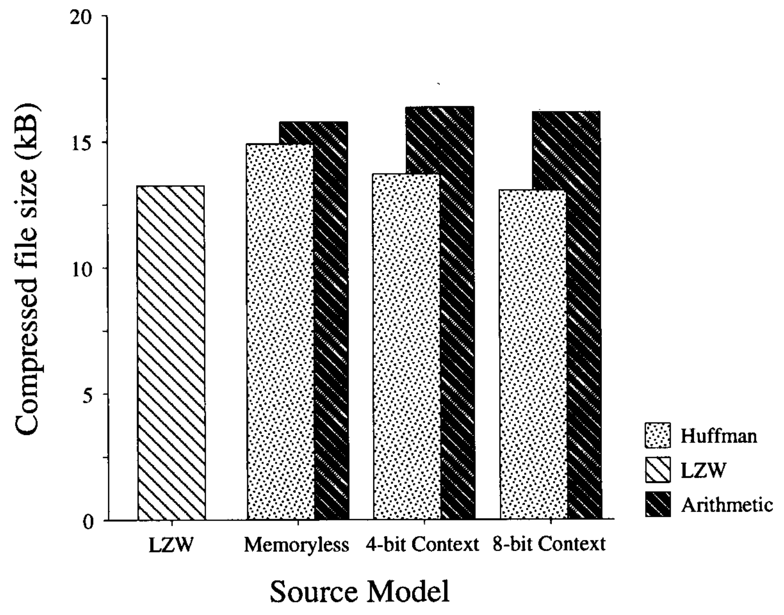


(a) $K_p = 5$.

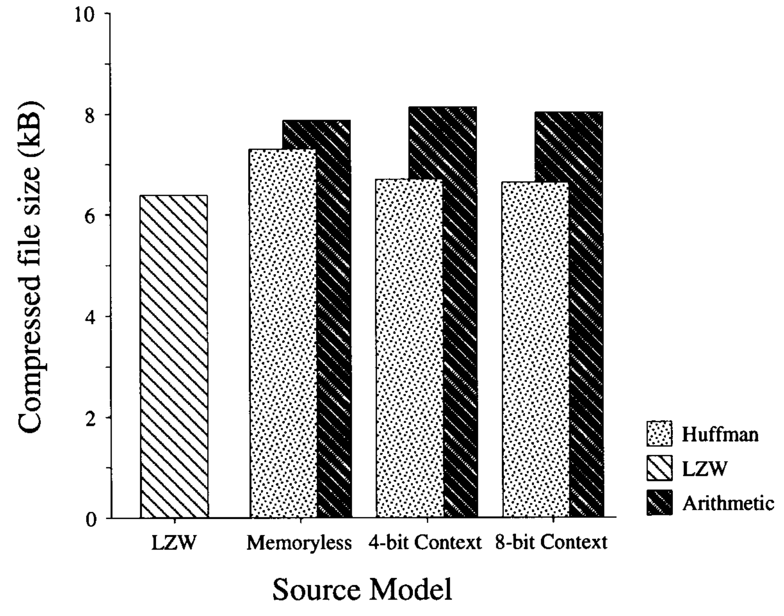


(b) $K_p = 10$.

Figure 5.18: VLC perf. for varying algorithms: LP comp. of country.Y.



(a) $K_p = 15$.



(b) $K_p = 20$.

Figure 5.19: VLC perf. for varying algorithms: LP comp. of country.Y (cont.).

Best Compression

Next we state the best compression performance over all algorithms and source models that were tested. Tables 5.2–5.5 lists compression results of AC components for priority thresholds $K_p = 5,10,15$, and 20, respectively. The results for the DC components are given in Table 5.6.

As the tables show, the Huffman coder with an 8-bit previous symbol context generally outperformed all other compression schemes. The only exception was that the LZW algorithm worked better for small files; however, the relative gains were not significant. Additional compression can be attained by designing the VLC in accordance with the RLC structure. Rather than working on 8-bit input symbols, the VLC can incorporate the parameters of the RLC: the run length size, the 3 bits representing the size of the amplitude, and the amplitude. The fact that a Markov model is optimal says that the output bit streams of the RLCs have some correlation.

For the DC files, the Huffman coder with a memoryless model worked best. This result makes sense since the DC files are small: VLCs with memory models require more data to suitably represent the source.

Next we analyze average bit rates associated with compression results from Tables 5.2–5.6. Assuming the transmission consists of grayscale images of size 720×576 at 45 frames/sec, the uncoded bit rate is 149.30 Mb/s (for color video, where two additional components at 8 bits per sample are transmitted, the raw data rate becomes 447.90 Mb/s). Table 5.7 gives an idea of the types of gains that can be attained (for $K_p = 15$). Note that, on average, a rate reduction ratio of 7:1 appears reasonable.

Priority	Original Size (kB)	File	After RLC (kB)	Best VLC (kB)	Algorithm
High	45.360	boat	11.099	9.346	LZW
		country	17.008	14.213	H8
		girl	15.410	13.050	H8
		zelda	15.158	12.651	H8
Low	306.180	boat	27.558	20.223	H8
		country	47.575	34.125	H8
		girl	49.889	37.090	H8
		zelda	41.422	30.841	H8

Table 5.2: Compression results of AC components with $K_p = 5$.

Priority	Original Size (kB)	File	After RLC (kB)	Best VLC (kB)	Algorithm
High	102.060	boat	23.029	18.704	H8
		country	35.309	26.713	H8
		girl	36.942	28.436	H8
		zelda	32.968	25.254	H8
Low	249.480	boat	12.778	9.822	LZW
		country	27.088	20.411	H8
		girl	25.789	20.097	H8
		zelda	21.059	16.981	H8

Table 5.3: Compression results of AC components with $K_p = 10$.

Priority	**Original Size (kB)**	**File**	**After RLC (kB)**	**Best VLC (kB)**	**Algorithm**
High	158.760	boat	29.046	22.818	H8
		country	46.133	34.219	H8
		girl	48.550	36.654	H8
		zelda	41.648	31.673	H8
Low	192.780	boat	8.101	6.097	LZW
		country	16.078	13.065	H8
		girl	12.983	10.873	LZW
		zelda	12.047	10.281	H8

Table 5.4: Compression results of AC components with $K_p = 15$.

Priority	**Original Size (kB)**	**File**	**After RLC (kB)**	**Best VLC (kB)**	**Algorithm**
High	215.460	boat	34.392	25.745	H8
		country	54.559	39.808	H8
		girl	56.571	42.723	H8
		zelda	49.283	36.990	H8
Low	136.080	boat	4.625	2.912	LZW
		country	7.994	6.386	LZW
		girl	6.072	4.447	LZW
		zelda	6.115	4.738	LZW

Table 5.5: Compression results of AC components with $K_p = 20$.

Original Size (kB)	**File**	**Best VLC (kB)**	**Algorithm**
12.960	boat	11.829	H4
	country	11.210	H0
	girl	11.234	H0
	zelda	11.890	H0

Table 5.6: Compression results of DC components.

File	**Best Compression (kb)**	**Rate[†] (Mb/s)**	**Rate Reduction Ratio[††]**
boat	325.952	14.668	10.18 : 1
country	467.952	21.058	7.09 : 1
girl	470.088	21.154	7.06 : 1
zelda	430.752	19.384	7.70 : 1

[†] At 45 frames per second.
[††] As compared with the uncoded rate of 149.3 Mb/s.

Table 5.7: Average rate reduction.

Chapter 6

Conclusions

Digital video is an important application in the future B-ISDN. We focused on transmitting high data rate video over an ATM network. The question of what compression technique gives best performance was examined by comparing three compression algorithms and various source models.

Our results show that an average ratio of 7:1 is possible for the compression performance. This gain is coupled with graceful degradation in the event of cell loss. The inherent priority feature of ATM networks allowed hierarchical coding to combat the effects of cell loss without FEC.

We optimized the run length codec and presented optimal parameters for each type of input stream. For the model-based compression schemes, several source models were examined. It was found that, for the AC component of the bit stream, a Huffman coder with a first-order Markov model with an 8-bit context outperformed all other combinations of models and compression algorithms. For the DC component, a memoryless Huffman combination performed the best.

Incorporating the order estimator into the compression scheme was not beneficial for the test files. The reason for this was primarily due to the small size of the test files. (It is ironic that, for the AC files, the high compression gains from the RLC caused the order estimator to perform poorly.) However, order estimation is useful since VLCs utilizing a Markov model of arbitrary order can better adapt to the given video source. The arithmetic coder performed worse than the other algorithms due to the limitation of fixed-length arithmetic operations. The VLC results were similar regardless of the priority threshold and test images. The codec is thus robust to changing video sources.

For the problem of rate control, we provided a framework for adaptive schemes by formulating the problem as one of constrained minimization and studying the properties of optimal solutions. Algorithms were presented to solve for these optimal solutions. A key feature of our analysis is the consideration of more than one previous buffer state for the decision process. Although an optimal solution with the Viterbi algorithm introduces large delays, we expect that some gains can be attained by increasing the order of the decision rule. A heuristic arbitrary-order rule was introduced based on hysteresis. This practical scheme provides stability by avoiding frequent changes in quantizer choice. For adaptive rate control systems, we expect some form of steady-state behavior. Such long-term results can be used to fix a decision rule to avoid the problem of delay introduced by the Viterbi search algorithm.

The rate control system contains many problems for future research. Further simplifications in computing optimal solutions is needed (e.g., to compute the solution to the M-step problem in real-time). Also, simulation analysis of the rate control mechanism is needed to see how performance is affected by increasing the order of the decision rules. All the analysis for this area involves much software work. Finally, the incorporation of the rate control system into the video codec is needed to judge the performance of the entire system.

Bibliography

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Computers*, vol. C-23, pp. 90-93, Jan. 1974.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Proc.*, vol. 1, no. 2, pp. 205-220, Apr. 1992.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, Inc., 1991.

[4] M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN [2nd ed.].* New York: Ellis Horwood, 1993.

[5] N. Farvardin and J. W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Info. Th.*, vol. IT-30, no. 3, pp. 485-497, May 1984.

[6] N. Farvardin and J. W. Modestino, "Adaptive buffer-instrumented entropy-coded quantizer performance for memoryless sources," *IEEE Trans. Info. Th.*, vol. IT-32, no. 1, pp. 9-22, Jan. 1986.

[7] R. G. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Info. Th.*, vol. IT-24, no. 6, pp. 668-674, Nov. 1978.

[8] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers, 1992.

[9] M. Goldberg and L. Wang, "Comparative performance of pyramid data structures for progressive image transmission," *IEEE Trans. Comm.*, vol. 39, no. 4, pp. 540-548, Apr. 1991.

[10] R. M. Gray and L. D. Davisson, *Random Processes: A Mathematical Approach for Engineers.* New Jersey: Prentice-Hall, 1986.

[11] D. D. Harrison and J. W. Modestino, "Analysis and further results on adaptive entropy-coded quantization," *IEEE Trans. Comm.*, vol. 36, no. 5, pp. 1069-1088, Sep. 1990.

[12] G. G. Langdon, Jr., "An introduction to arithmetic coding," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 135-149, Mar. 1984.

[13] D. S. Lee and K. H. Tzou, "Hierarchical DCT coding of HDTV for ATM networks," *Proc. ICASSP*, pp. 2249-2252, Apr. 1990.

[14] D. J. Le Gall, "MPEG: A video compression standard for multimedia applications," *Comm. of the ACM*, vol. 34, no. 4, pp. 46-58, Apr. 1991.

[15] C. C. Liu and P. Narayan, "Order estimation and sequential universal data compression of a hidden Markov source via the method of mixtures," *IEEE Trans. Info. Th.*, to be published.

[16] P. Narayan, "Information theoretic methods in system identification and universal data compression," *Class notes*, University of Maryland, College Park, MD, Fall 1991.

[17] N. B. Nill, "A visual model weighted cosine transform for image compression and quality assessment," *IEEE Trans. Comm.*, vol. COM-33, no. 6, pp. 551-557, Jun. 1985.

[18] N. Ohta, *Packet Video: Modeling and Signal Processing.* Boston: Artech House, 1994.

[19] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based compression and fast approximations," *IEEE Trans. Image Proc.*, vol. 3, no. 1, pp. 26-40, Jan. 1994.

[20] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1993.

[21] R. C. Reininger and J. D. Gibson, "Distributions of the two-dimensional DCT coefficients for images," *IEEE Trans. Comm.*, vol. COM-31, no. 6, pp. 835-839, Jun. 1983.

[22] J. Rissanen, *Stochastic Complexity in Statistical Inquiry.* New Jersey: World Scientific, 1989.

[23] K. H. Tzou, "Progressive image transmission: a review and comparison of techniques," *Optical Eng.*, vol. 26, no. 7, pp. 581-589, Jul. 1987.

[24] K. H. Tzou, "An intrafield DCT-based HDTV coding for ATM networks," *IEEE Trans. CSVT*, vol. 1, no. 2, pp. 184-196, Jun. 1991.

[25] W. Verbiest, L. Pinnoo, and B. Voeten, "The impact of the ATM concept on video coding," *IEEE JSAC*, vol. 6, no. 9, pp. 1623-1632, Dec. 1988.

[26] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979.

[27] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Elec.*, vol. 38, no. 1, pp. 18-34, Feb. 1992.

[28] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8-19, Jun. 1984.

[29] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Comm. of the ACM*, vol. 30, no. 6, pp. 520-540, Jun. 1987.

[30] J. Zdepski, D. Raychaudhuri, and K. Joseph, "Statistically based buffer control policies for constant rate transmission of compressed digital video," *IEEE Trans. Comm.*, vol. 39, no. 6, pp. 947-957, Jun. 1991.

[31] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Info. Th.*, vol. IT-24, no. 5, pp. 530-536, Sep. 1978.