JOINT SOURCE-CHANNEL MATCHING
FOR WIRELESS VIDEO TRANSMISSION

BY

LEIMING QIAN

B.E., Tsinghua University, Beijing, 1996
B.E., Tsinghua University, Beijing, 1996

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1999

Urbana, Illinois

ABSTRACT

With the rapid growth of multimedia content in today's wireless communication, there is an increasing demand for efficient image and video transmission systems. Various research work has shown that significant gain in transmission performance can be achieved from the application of joint source-channel matching [1] techniques, which minimize the final end-to-end distortion by optimally allocating limited system resources (transmission power, channel capacity, etc.) between the source and channel coders.

Although significant results have been achieved using matching techniques designed for specific source and channel coders, communication system designers usually have to deal with preselected source-channel coder pairs, varying transmission conditions, and heterogeneous multimedia content. Thus, it is desirable to develop a more general technique for parametric-model-based, real-time adaptive matching, which not only can be applied to a wide variety of source-channel coder pairs and varying channel conditions but also will provide optimal achievable visual quality under all circumstances.

This thesis presents such a scheme, whose effectiveness is illustrated using two block-transform-based video source coders (motion-JPEG and conditional block replenishment) and the Reed-Solomon channel coder. The system should also work with other source and channel coders, such as motion-compensation-based video coders (MPEG, H.263) and convolutional channel coder (RCPC).

DEDICATION

To my parents, whose unconditional love has supported me throught all these years of hardship, and to my uncle Shie Qian, without whose direction I would have never entered the field of engineering in the first place.

ACKNOWLEDGMENTS

This work has largely been supported by the U.S National Science Foundation through grant no. MIP-9707742.

I would like to thank my advisor, Professor Douglas L. Jones, for all the kindness and patience he has shown toward me and for all the helpful and insightful suggestions he has given me. Working with him has made research both a fruitful and enjoyable experience.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this chapter, we explain the basic concept of joint source-channel matching, briefly review the research work that has been done on this topic, and present a general system diagram of our approach. Some background knowledge related to video coding and wireless video transmission is also given.

1.1 Joint Source-Channel Matching

Roughly speaking, *source coding* is a data-compression process that aims at removing as much redundancy as possible from the source signal, whereas *channel coding* is the process of intelligent redundancy insertion to create protection against channel noise. In this aspect, they seem to act in opposition [1].

The joint source-channel coding theorem of Shannon [2] points out that if the minimum achievable source-coding rate of a given source is below the capacity of a channel, then the source can be reliably transmitted through the channel, assuming that the sequences of source samples are appropriately long; furthermore, if the source-coding rate is strictly greater than the channel capacity, then reliable transmission is impossible. This theorem implies that source coding and channel coding can be treated separately without any loss of overall performance; in other words, they are fundamentally separable.

However, this separability holds only if the communication is point-to-point, which is not realistic in a broadcasting environment. It is also not yet known whether this result holds for practical systems with finite delay or multipath fading [3]. Furthermore, it has been shown that despite its optimality, Shannon's separation theorem does not necessarily result in *the best economic solution*; for these reasons, joint source-channel matching may offer better performance in practical situations than a separable solution.

The objective of joint source-channel matching is to include both source and channel coding modules in the same processing block in order to reduce the overall complexity of the system, while maintaining satisfactory performance. Most current image and video source coders are optimized to give the best performance at a certain rate, assuming all the coded bits are received perfectly. Likewise, most existing channel coders are not necessarily well matched to specific source coders. The reason behind matching jointly the source and channel coder can be best represented by Figure 1.1.

In Figure 1.1, the left plot shows that increasing the source coding rate (less compression) decreases the final end-to-end distortion; the right plot shows that increasing channel protection can reduce possible channel errors, which also decreases the final end-to-end distortion. However, because we have a limited channel capacity, there is a fundamental trade-off between these two rates. The curves ($p1$, $p2$, $p3$) in the middle plot show how a conventional system with fixed source rate and channel protection works for different channel situations. Each point on the convex hull of these curves corresponds to the optimal coding configuration with regard to the current channel situation.

A great amount of research has been done on this topic with significant results; here we present an incomplete list: Lan and Tewfik [4] examined the trade-off between source and transmission power; Davis and Danskin [5] described a joint source-channel allocation scheme for transmitting images losslessly over block erasure channels such as the Internet; Ramchandran et al. [3] studied multiresolution coding and transmission in a broadcasting scenario; Azami et al. [6] acquired performance bounds for joint source-channel coding of uniform memoryless sources using binary decomposition; Belzer et al. [7] developed a joint source-channel image coding method using trellis-coded quantization and convolutional codes; Sherwood and Zeger [8] investigated unequal error protection for the binary symmetric channel; and Man et al. [9] examined unequal error protection and source quantization. An extensive survey on general joint source-channel matching was done by Azami et al. in [1].

However, as pointed out in [10], we generally lose flexibility as we reduce complexity. If one opts for a jointly coded system, one can no longer easily adapt one's system later to a different source (or channel). This is a fundamental shortcoming of all the existing joint source-channel matching (JSCM) schemes; though they successfully show that significant

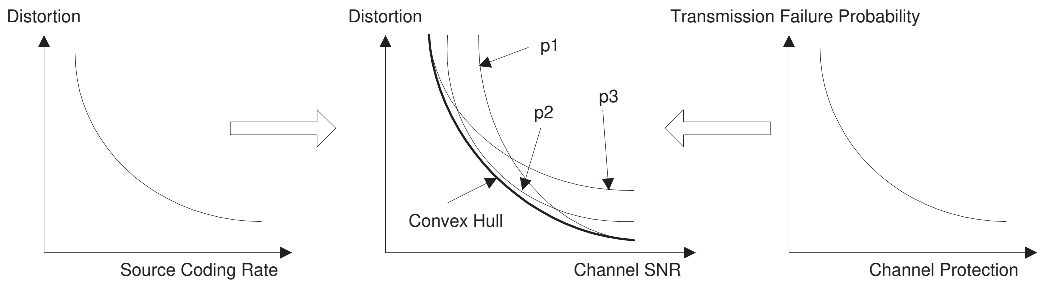

Figure 1.1 Joint Source-Channel Matching

performance improvements can be achieved by joint source-channel optimization, they are all based on some specific source and channel coder pairs and thus rely heavily on detailed knowledge about the internal working mechanism of those coders. In practice, wireless multimedia links contain heterogeneous contents (audio, image, video, etc.), each with its own coder, over which we may only have minimal control for such things as adjusting the rate. Also, the channel coder is usually preselected by the vendor. We thus need to develop a joint source-channel matching scheme that can support a wide variety of source and channel coders without going deep into the coder details. This thesis proposes such a scheme, which performs an end-to-end optimization over both source and channel characteristics based on parametric models.

1.2 Parametric-Model-Based Approach

The key to the generality and adaptivity of our scheme is the parametric-model-based approach (more specifically, the parametric model for the source is usually the rate-distortion curve, whereas the model for the channel is usually the rate-error probability curve) [11], [12].

In practice we need to cope with a wide range of video source coders, for which we may only have minimal control (such as adjusting the rate or compression ratio), and also channel coders which are preselected. Current joint source-channel matching schemes usually consider a particular pair of source and channel coders. These schemes, while able to utilize the specific internal characteristics of those particular coders, may not yield a satisfactory performance if another source or channel coder is used. The parametric model approach, on the other hand, does not rely on heavy utilization of the detailed technical information of the coders. The parameters can be estimated on-line by evaluating a limited number of "anchor points" followed by piecewise interpolation [13], [14]. This coder-independence helps our system achieve generality.

The parametric-model approach also enables our scheme to adapt to varying channel situations. In practical situations, the channel is usually changing. In our scheme, except for fast-varying channels (where the channel has already changed again while the optimization is performed), the change in channel situation merely corresponds to another rate-error probability curve, which can be precalculated off-line or even evaluated on-line. Thus, as long as we have accurate channel situation feedback or estimation, our scheme can effectively track the variation, adapt itself to the current channel, and maintain optimality under all conditions.

1.3 Reed-Solomon Channel Coder and Packet-Based Approach

Reed-Solomon (RS) codes [15] are a well-known class of block codes with good error correction properties. We chose them to demonstrate our scheme because of their ability to correct channel burst errors, which are common in a wireless environment. An RS code defined by (n, k, t) is a length-n code which contains $k = n - 2t$ source symbols, $2t$ protection symbols, and can correct t symbol errors. There are RS codes for various n, most commonly $n = 2^m - 1$ where m is the symbol length in bits (in our scheme we will use $m = 8$, because we usually store images in bytes). An (n, k, t) code will be unable to recover the original k data symbols if more than t errors occur.

Because the RS coder has a fixed-length configuration, we employ a packet-based approach in channel coding, in which each transmission packet is an RS coding unit and contains both the source symbols and protection symbols. In order to calculate expected distortion, we define, for video transmission, the *frame transmission success probability* P_{fsucc} as:

$$P_{fsucc} = \prod_{n=1}^{M} P_{psucc} \tag{1.1}$$

$$P_{psucc} = \sum_{k=0}^{p/2} \binom{L}{k} {P_{es}}^{k} {P_{ss}}^{L-k} \tag{1.2}$$

$$
\begin{aligned}
P_{ss} &= 1-(1-P_{eb})^m && (1.3)\\
P_{eb} &= Q\left(\sqrt{\frac{2E_b}{N_0}}\right) && (1.4)
\end{aligned}
$$

where P_{psucc} is the *packet transmission success probability*, M is the total number of packets to transmit, P_{ss} is the *symbol transmission success probability*, L is the packet length, p is the number of protection symbols in each packet, P_{eb} is the bit-error probability, m is the number of bits per symbol, E_b is the fixed power per bit, and N_0 is the channel noise variance. In order for the above equations to be valid, sufficient interleaving must be employed so that the independence assumption is satisfied.

1.4 The JPEG Image Coder

We are motivated to study the JPEG [16] image source coder in detail because JPEG or its underlying discrete cosine transform (DCT) is also used by many video coding standards in individual frame coding or block coding. We are particularly interested in the effect of a bit error occurring in a certain location in the transmitted bit stream, i.e, the relationship between bit error location and the effect it has on the final distortion.

1.4.1 JPEG coder basics

The basic coding flow of JPEG is as follows. First, the raw image is divided into blocks (usually of size 8×8). Second, a DCT transform is performed on each block. Third, the resulting coefficients are quantized using a quantization matrix. Fourth, the dc coefficients are transmitted directly. And finally, the ac coefficients are zig-zag scanned and symbol-coded to form the rest of the JPEG bit stream. Figure 1.2 illustrates the JPEG encoder.

From the diagram, we observe that the most direct way to adjust the JPEG encoder is to change the quantization matrix. In practice, JPEG provides two default quantization matrices (one for luminance and one for chrominance) and also a scaling variable called the *quality factor*, q. We use this factor to adjust the JPEG source coder because of its simplicity. One thing to note is that although q is commonly a linear scaling factor of the

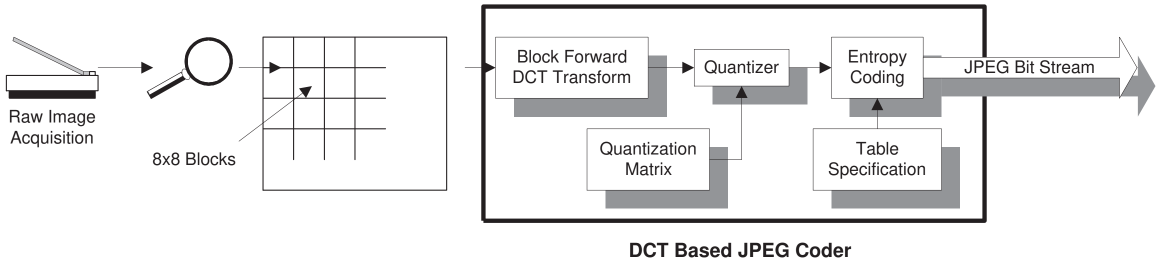

Figure 1.2 JPEG Encoder

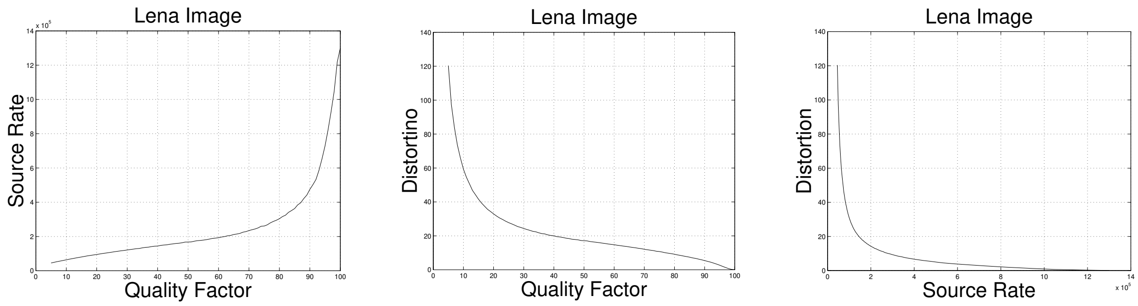

Figure 1.3 Quality Factor, Source Rate, and End-to-End Distortion

quantization matrix, it does not have a linear relationship with either the resulting coded bit stream length or the final end-to-end distortion, as shown in Figure 1.3. (However, over certain ranges of the quality factor, there is an approximately linear relationship, which can be exploited to simplify the optimization computation.)

1.4.2 Baseline JPEG

Baseline JPEG is the simplest and most commonly used mode of the JPEG coding algorithm. It does not have any progressive nature, as shown in Figure 1.4, which is generated by introducing an error in a certain bit of the encoded JPEG bit stream, decoding it, and calculating the final end-to-end distortion. We can see multiple and irregular "spikes" in the figure, which means that there are certain bits in the stream which are so important that an error will cause a total loss of the image. Further investigation revealed that these important bits usually belong to the marker bytes.

Note the right part of Figure 1.4 shows the situation when we add in 10 restart markers per row into the JPEG bit stream. The restart markers can effectively "localize" the distortion, since it essentially restarts the decoding procedure periodically.

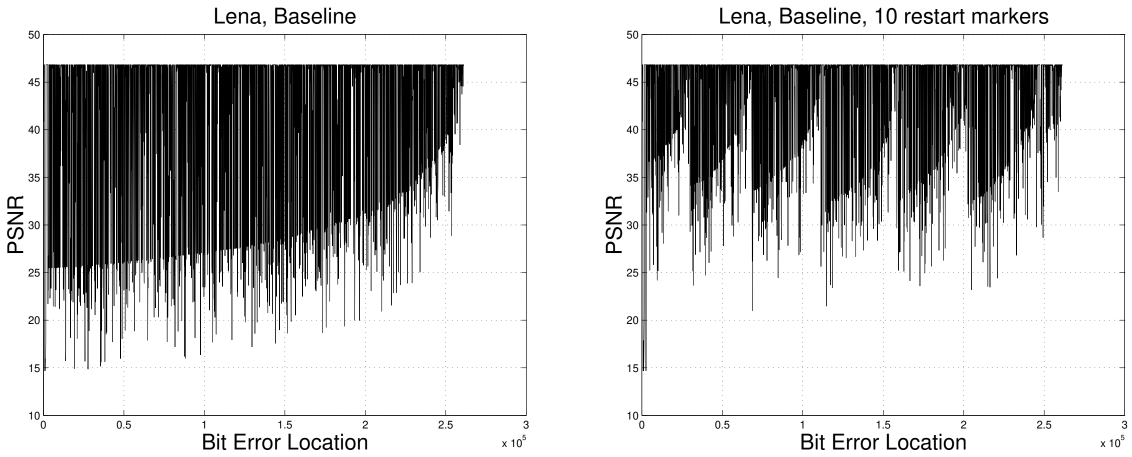

Figure 1.4 Bit Error Location versus PSNR, Baseline JPEG

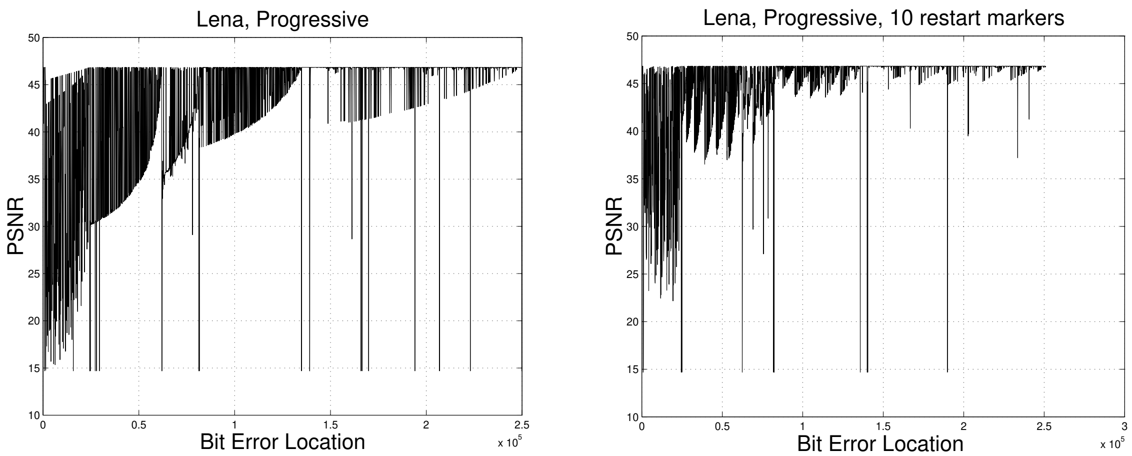

Figure 1.5 Bit Error Location versus PSNR, Progressive JPEG

1.4.3 Progressive JPEG

Progressive JPEG, despite its name, is not a truly progressive coder, because of the two basic strategies in which it is implemented: (a) *spectral selection*, in which the DCT coefficients are ordered into spectral bands and successively transmitted, or (b) *successive approximation*, in which the dc coefficients are transmitted first, and the ac coefficients are transmitted by bitplanes. Neither of these is really progressive in the sense that there are still important bits which can not be in error, thus resulting in "spikes" in the bit error location versus distortion curve, as shown in Figure 1.5.

1.4.4 Expected distortion

Due to the nonprogressive nature of the JPEG coder and the "spikes" we observed, we cannot claim that we will always be able to recover part of the information if an error has occurred during the transmission. Although it is theoretically possible to employ a "spike removal" procedure by locating all the marker bytes (and those data bytes which tend to become marker bytes when an error occurs) and transmitting them with extra protection, the computational overhead involved would obviously be very high. Thus we introduce the concept of *image transmission failure probability*, P_{fail}; and use the *expected distortion* as the final end-to-end distortion, which will be defined as:

$$D = P_{fail} \times D_{loss} + P_{succ} \times D_{succ}(q) \tag{1.5}$$

where D_{loss} is the distortion if we lose the entire image and $D_{succ}(q)$ is the distortion if the frame gets transmitted without any error, which is obviously a function of the quality factor q used in compressing the original raw image.

1.5 Video Coding Issues

Because of its temporal dimension, video coding has a few specific issues which need to be addressed before a joint matching scheme can be implemented.

1.5.1 Video quality evaluation

The quality of a video sequence is difficult to evaluate because of our limited knowledge of the human visual system (HVS). In terms of individual frame quality, *mean square error* (MSE) has always been a favorite because of its quadratic nature, which facilitates mathematical analysis (*mean absolute error* (MAE) is also a commonly used alternative). For video, average MSE is normally used.

However, it is known that the HVS is very sensitive to image quality *variation*; in other words, a video sequence with a constant or near-constant level of distortion may be more desirable than one with lower average distortion but higher variability. Thus, frame quality variation combined with average frame quality can also be a candidate for evaluating video sequences.

In order to demonstrate the generality of our scheme, both MSE and frame quality variation are used in our simulations.

1.5.2 Distortion concealment

In video transmission it is a common practice to employ *distortion concealment* techniques, such as using the previous successfully received frame in place of the current frame when we fail to receive it. The resulting difference between these two frames is the *Distortion Concealment Error*, which is a function of the previous *encoded* frame and the current *unencoded* frame. However, as it would introduce considerable computational complexity into our optimization scheme, we use the difference between the *unencoded* previous and current frame as the distortion concealment error.

1.5.3 Scene change detection

Whenever there is a drastic change in the nature of video sequence content, such as a change from a football sequence to a scenery sequence, we say that a *scene change* has occurred. We need to detect scene changes because they usually require an update of the source model parameters.

Because scene change detection is itself a current research topic, there is no as yet optimal method proposed. For the sake of simplicity, we use the difference between frames as the detection criterion, which means that when the mean square difference between two consecutive frames exceeds a certain threshold, we say that a scene change has occurred, thus requiring an update of the rate-distortion function curve. This does not add to the computational complexity of the overall scheme, because the framewise mean square difference has to be calculated to obtain the distortion concealment error in any case.

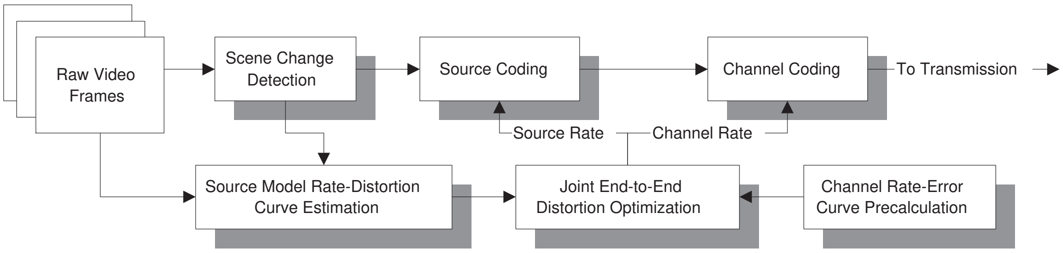

Figure 1.6 General System Diagram

1.5.4 Rate control

One problem unique to video transmission is the rate-control [17], [18] constraint, because the decoder usually has a finite size decoding buffer and a constant decode-flow.

Suppose we have a video frame sequence divided into GOPs (groups of pictures) of size N. Each frame is coded to have a stream length of $s(i, q_i)$, where i is the frame index and q_i is the source-coding rate-control factor (for example, the quality factor in JPEG coding). For the transmission of the ith frame, a certain amount of protection, denoted by p_i, is used. Then, the buffer occupancy b_i after frame i is transmitted can be expressed as

$$b_i = \max(b_{i-1} + s(i, q_i) + p_i - R, 0) \tag{1.6}$$

where R is the constant decoding stream flow (stuffing-bits will be added to avoid underflow when $b(i)$ is smaller than zero). The buffer size constraint is then represented as

$$b_i \leq b_{max}, i = 1...N \tag{1.7}$$

where b_{max} is the finite decoding buffer size. The optimization thus becomes a constrained problem.

1.6 General System Diagram

Finally, we present our general system diagram (Figure 1.6) for arbitrary source and channel coders. The system contains six major blocks, each fulfilling an essential function in the entire scheme: the scene change detection block detects a scene change and informs the system when an update of the model parameters is necessary; the source model rate-distortion curve estimation block estimates the source rate-distortion characteristics, usually by evaluating several "anchor points" and performing piecewise interpolation to get the entire curve; the joint optimization block minimizes the end-to-end distortion and yields the optimal coding parameters for both the source and channel coders. Exactly how some of these blocks are implemented may differ slightly as we change coders, but they can always be implemented using the minimal available control we have on the coders.

In the next two chapters, we demonstrate our system using two video source coders: motion-JPEG and conditional block replenishment.

CHAPTER 2

MOTION-JPEG AND RS CODER

Motion-JPEG is perhaps the simplest existing video coder; it simply codes each frame in JPEG format and transmits the coded frames sequentially. In spite of its simplicity, it is commonly used because of the constant frame quality and its ability to start decoding at any particular frame.

2.1 Problem Formulation

To formulate the problem when the source coder is Motion-JPEG, we assume that we have a series of video frames that are divided into GOPs of size N.

We define the optimization problem as minimizing the final end-to-end distortion with respect to two size-N vectors Q and P, where $Q = (q_1, q_2, ..., q_N)^T$ contains the quality-factors used to JPEG-encode each frame, and $P = (p_1, p_2, ..., p_n)^T$ contains the amount of protection we use for transmitting all the packets contained in a certain frame. Because the source rate and distortion are both functions of the quality factor q, we define the source model to be two sets of curves: the distortion versus quality-factor curve $d_i(q_i)$ and the source-rate versus quality-factor curve $s_i(q_i)$ (i is the frame index).

However, further investigation into the problem revealed that the $d_i(q_i)$ and $s_i(q_i)$ curves are very similar for all the frames in the same GOP (see Figure 2.1). Except when there is a scene change, we can thus safely assume that almost all the frames are similar in the sense that they have identical $d(q)$ and $s(q)$ functions. This leads to the natural conclusion that we can use the same quality factor and amount of protection for all the frames in the same GOP, provided that the buffer size limitation is also satisfied. Though this yields a slightly suboptimal solution, this approach is comparable in performance

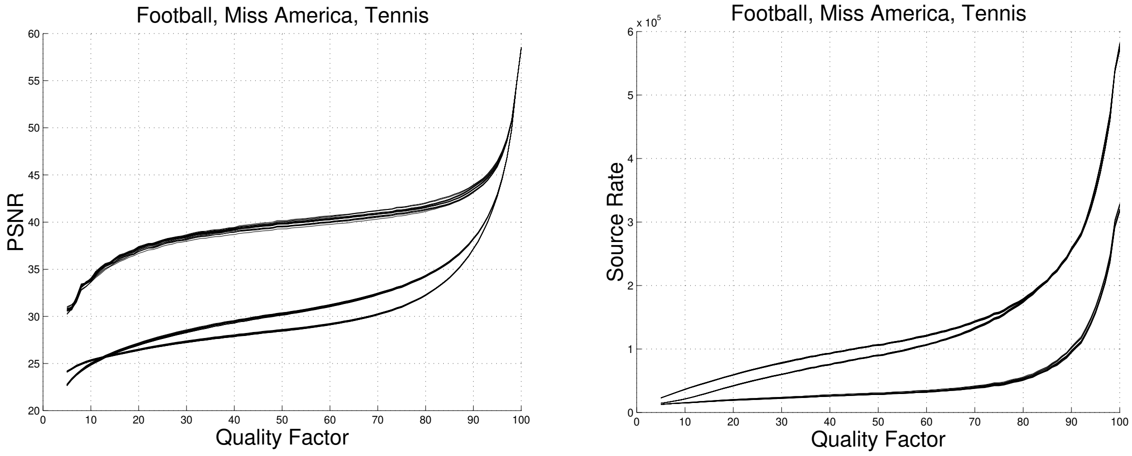

Figure 2.1 PSNR/Rate versus Quality-factor curves

with the truly optimal solution, but is much simpler in terms of computational complexity because we reduced the $2N$-D optimization problem to a 2-D problem.

The final optimization problem can be formulated as finding the optimal scalars q and p that minimize the final distortion D:

$$D = \frac{1}{N}\sum_{i=1}^{N} D(i) \tag{2.1}$$

$$D(i) = P_{fsucc}(p) \times d(q) + P_{ffail}(p) \times d_{conceal}(i) \tag{2.2}$$

$$P_{fsucc} = \prod_{n=1}^{M(q)} P_{psucc} \tag{2.3}$$

$$M(q) = \frac{s(q)}{L} \tag{2.4}$$

Here D is the final end-to-end distortion, $D(i)$ is the expected distortion for the ith frame, $P_{fsucc}(p)$ and $P_{ffail}(p)$ are the frame transmission success and failure probabilities when we use p protection symbols for each block, $d(q)$ is the distortion for each frame when coded using quality-factor q, $d_{conceal}(i)$ is the distortion when we use the $(i-1)$th frame in place of the ith frame, $M(q)$ is the number of packets in each frame, $s(q)$ is the frame length when coded using quality-factor q, and L is the predefined RS packet size.

We must also satisfy the following decoding buffer-size constraints:

$$b_i = max(b_{i-1} + s(q) + p - R, 0) \quad (2.5)$$

$$b_i \leq b_{max}, i = 1, 2, ..., N \quad (2.6)$$

2.2 Implementation Details

Straightforward as the algorithm itself is, there are a number of implementation issues to be solved before the scheme becomes practical. Here we list the problems and our respective solutions:

- Obtaining the source model parameters

  Although we assume that we use the same rate and distortion function across all the frames in the same GOP, a great deal of computation is still required to actually obtain the exact value for all the points on those two curves. We also require an update of the model parameter whenever we detect a scene change in the video sequence, so we have to find some efficient methods to obtain the model parameters. In order to do that, we use the following method [13], [14]: first define several "anchor points" on both of these two curves and obtain the value associated with these points, and then use piecewise cubic spline approximation, followed by interpolation to get all the other points on the curves. Simulations show that, if carefully chosen, only six or fewer points are needed to achieve a satisfactory approximation accuracy.

- P_{psucc} (RS packet transmission success probability) approximation

  As the simulation shows, the calculation of the exact P_{psucc} is a rather computationally expensive task. However, as we can observe from its calculation formula, P_{psucc} is actually the probability distribution function of a binomial random variable; when the packet size is sufficiently large, we can approximate it as a Gaussian random variable; thus, greatly reducing the computational complexity.

- Optimization constraint removal

The optimization problem we have defined is constrained and nonlinear; furthermore, the variables can only take integer values. In order to solve this problem, we need to transform it into an unconstrained problem. We chose the penalty function approach [19], [20] and modified our cost function as:

$$D = \sum_{i=1}^{N} D(i) + C \times \sum_{i=1}^{N} \max(0, b_i - b_{max}) \tag{2.7}$$

where $D(i)$ is defined in Equation (2.2) and the extra term is the penalty function; when C goes to infinity, this forces the solution to converge to the optimal solution to the constrained problem. (Because the solution to our problem is discrete, the converged solution is the truly optimal answer, not a suboptimal one in the continuous case.) In practice, we use a sufficiently large value for C.

- Optimization Initialization

  As always, a good starting point in an optimization problem is very important. A carefully chosen starting point can greatly reduce the iterations needed to reach the optimal point. In our optimization scheme, we use a gradient search method which can effectively reduce the optimization process to under five iterations.

  The probability of frame transmission failure, P_{ffail}, which determines how much protection we should use, is mostly decided by the individual RS packet transmission failure probability, P_{pfail}. It has been observed that the P_{pfail} versus p (p is the number of protection symbols used) curve exhibits a waterfall pattern (Figure 2.2): starting from $p = 0$, P_{pfail} is almost one for a certain range, then decreases quickly until it reaches zero, and stays at zero until p reaches its maximum allowed value. Simulations show that the optimal choice for p is always located somewhere on the waterfall slope. Thus we search the gradient of the P_{pfail} versus p curve and look for the point where the gradient reaches its maximum absolute value (Figure 2.3). This point corresponds to the "middle" point of the waterfall curve and would be our initial choice of p. We can then use the buffer size limit to get our initial choice of q. Simulation results show that we usually only need five or fewer iterations to reach the optimal point if initialization is done this way.

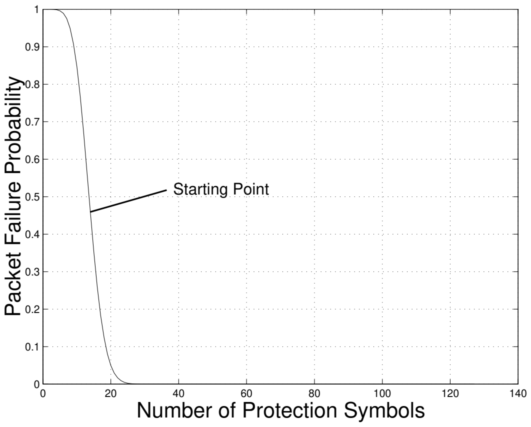

Figure 2.2 Failure versus Protection

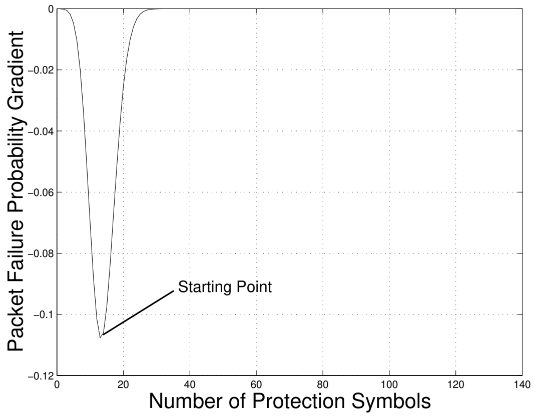

Figure 2.3 Failure Curve Gradient

Once the system is properly initialized and starts generating results, the next iteration for the following video frames will use the previous optimal configuration parameters to initialize, based on the belief that the channel is slowly varying so that the new optimal configuration should be close to the original one. This further reduces the iterations we need to reach the optimal solution.

The optimization procedure is performed as follows:

1. Collect video frames, calculate interframe blockwise MSE, acquire the distortion concealment error curve, and determine if a scene change has occurred.

2. Perform DCT coding on each frame. If the scene-change flag is set, quantize the frame using the predefined "anchor points" of quality factor q, and calculate resulting source rate and distortion; otherwise, skip to step 4.

3. Use piecewise cubic spline interpolation to get the values for all the other points on the source-rate versus q curve and distortion versus q curve.

4. Based on the channel situation feedback, estimate the initial choice for the amount of protection p. Apply the buffer-size constraint to estimate the initial choice of q.

5. Apply the gradient descent algorithm [11], [12] and the penalty function approach to minimize the cost function defined in Equation (2.7), and obtain the optimal value for p and q. (The optimal p and q will be used as the initialization points

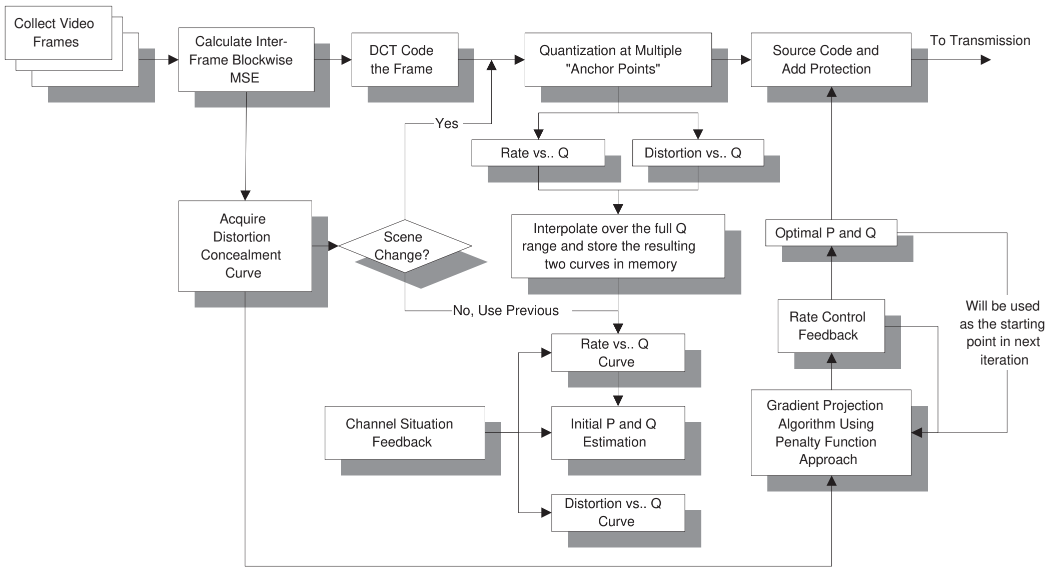

Figure 2.4 Motion-JPEG and RS coder

in the next frame group optimization.) The gradient is calculated numerically; the step sizes used are 1 for q and 2 for p, both being their minimal possible values.

6. Perform source and channel coding using the optimal p and q, return to step 1.

Figure 2.4 shows the block diagram of the system that works in real-time, transmitting video sequences coded in Motion-JPEG format. The system complexity is of the same order as DCT coding, and usually only no more than five iterations are needed to find the optimal solution.

The simulation results are shown in Chapter 4.

CHAPTER 3

CBR AND RS CODER

We chose the conditional block replenishment (CBR) coder as the second source coder to test our scheme because of its wide usage in video conferencing applications, where the frames consist of mostly still background and not much movement. It has certain advantages, such as coding simplicity, over motion-compensation-based video coders like MPEG in these situations.

3.1 CBR Coder Basics

A CBR coder works in the following fashion: first, blockwise differences between two consecutive frames are calculated, and then only those blocks with a difference greater than a certain predefined threshold are coded and sent. There are several ways to encode the blocks; the most commonly used is DCT coding, followed by quantization and symbol coding. For a CBR coder, we define the source model again as two sets of curves: the distortion versus threshold curve $d_i(t_i)$, and the source-rate versus threshold curve $s_i(t_i)$ (i is the frame index).

3.2 Problem Formulation

Instead of using just the average mean-square distortion as the optimization criterion, to demonstrate the generality of our scheme, we utilize here another type of criterion based on the characteristics of the human visual system (HVS). In this optimization scheme, we try to achieve an optimal balance between the average frame distortion and frame quality variation, while at the same time satisfying the rate-control constraint mentioned before. This can be mathematically represented as:

$$D = \frac{1}{N}\sum d_i(t_i) \tag{3.1}$$

$$V = \sigma(\mathbf{d}), \mathbf{d} = (d_1(t_1), d_2(t_2), ..., d_N(t_N)) \tag{3.2}$$

$$d_i(t_i) = P_{ffail} \times d_{conceal}(i) + P_{fsucc} \times d_{succ}(i, t_i) \tag{3.3}$$

$$b_i = \max(b_{i-1} + s_i(t_i) + p - R, 0) \tag{3.4}$$

$$b_i \leq b_{max}, i = 1, 2, ..., N \tag{3.5}$$

where D is the final average frame distortion, V is the standard deviation of the vector $\mathbf{d}$, which consists of individual frame distortion values, $d_i(t_i)$, and t_i is the threshold value used in coding the ith frame. The other variables are defined as in Section 2.1.

3.3 Implementation Issues

As is the case with the Motion-JPEG example, there are several implementation issues affecting this scheme's practicality, especially in obtaining the model parameters and implementing channel protection, for the following reasons:

1. Frame dependency of the source model

   Because the blockwise difference between two frames is heavily dependent on these two frames themselves, we must obtain the model parameter for every single frame, which conceivably increases computational complexity.

2. Blockwise protection scheme

   Unlike the Motion-JPEG situation, where we make the claim that we will discard the entire frame once we encounter an error in the entire transmitted bit stream, in a CBR coder we are transmitting individual image blocks. In this case an error can be effectively localized and only the block in error need be discarded. This blockwise protection scheme requires us to know exactly how long each *individual* coded source image block is; this may be difficult, because the source coding operation involves DCT coding, quantization, and symbol coding, all of which are nonlinear. It also means that the relationship between the threshold t and the source rate is difficult to obtain. Furthermore, as mentioned before, the RS coder has a fixed-length configuration. However, becaues we wish to employ a blockwise protection scheme and the length of individual image blocks varies, we can no longer provide a uniform amount of protection for each block. The computational complexity of optimization over different protection for each block would be prohibitively high. Nevertheless, we need to know the amount of protection in order to calculate the expected distortion.

In order to solve these problems, we introduce several simplifications.

1. Source model modification

   The most important modification we made is that we changed the adjustable parameter of the source coder. Originally it was the threshold value t used to identify those image blocks which need to be sent, and the model parameter curves are highly frame dependent (as shown in Figure 3.1), but now we use the number of image blocks to send n_b as the adjustable factor. Although it made little difference in terms of the coding operation (instead of using a threshold to test each block, we now just send a certain number of blocks that have the highest MSE), it made obtaining the model parameters much easier, because the modified source parameter curve $s(n_b)$ once again becomes approximately frame-independent given that there is no scene change (as shown in Figure 3.2). Although the distortion curve $d_i(n_b)$ still depends on each frame, the computation involved is not as high as initially expected, because there are a significant number of zero operations, and in any case, we will have to calculate the frame difference in order to perform scene-change detection.

2. Groupwise protection

   Instead of protecting coded source image blocks individually, we use an approximation of this approach. We define a certain RS coder size selected to be large enough to hold several coded source image blocks. When transmitting, we first put a fixed amount of protection into each RS-coded packet, and then put in as many coded image blocks into each packet as possible, finally filling the trailing empty

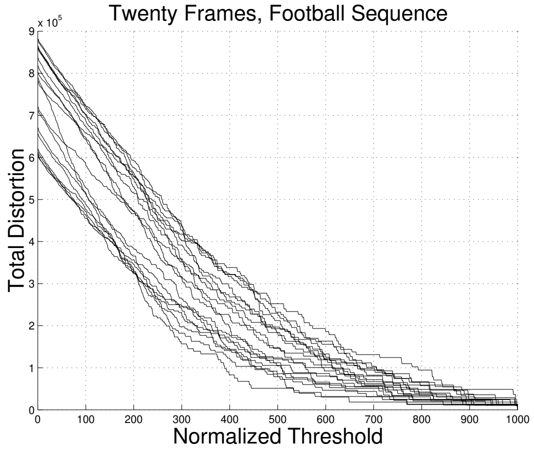

Figure 3.1 Distortion versus Threshold

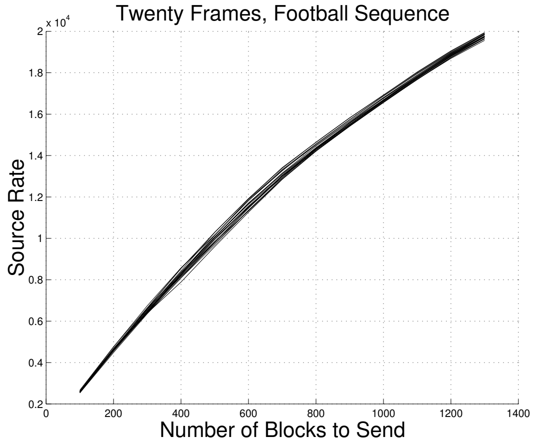

Figure 3.2 Blocks versus Source Rate

slots in each packet with extra protection symbols. This way, we lose only a small number of image blocks when we fail to receive a certain packet, and we have an approximately uniform amount of protection for each packet.

When determining how much protection to add and calculating the corresponding distortion, we apply the following simplifications:

(a) We assign a certain amount of protection so that the *packet error probability* P_{pfail} is lower than a predefined threshold (for example, 0.0001), and then we fix this amount of protection for all the packets during the optimization.

(b) When calculating the distortion, we only consider the worst possible situation; in other words, we assume that all the packets either contain no error or are all in error at the same time, thus simplifying the distortion calculation.

It is obvious that the first simplification yields a suboptimal answer, and the second simplification actually contradicts our groupwise protection scheme, because we would again either get everything or lose everything. However, these simplifications are justified by the waterfall-shaped P_{pfail} versus p curve. And as worst-case assumptions, they are always safe simplifications.

3. Entropy-based stream length allocation

Although we can now derive the final total coded source stream length from n_b, we still need to know the size of the individual coded image blocks to allocate them to packets. Because the entropy of an image characterizes how much it can be compressed, we assume that the coded size of each image block is roughly proportional to its entropy. Based on this assumption we can distribute the entire coded stream length to all blocks.

To simplify the optimization and enable online operation, we try to reach a suboptimal solution by using the following iterative procedure:

1 Assign a common frame distortion d_c to each frame, and determine the corresponding n_b for each frame to satisfy that requirement.

2 Using the piecewise approximated $s(n_b)$ curve and n_b, obtain the final coded stream length, and distribute it across all the blocks according to their individual entropy.

3 Add a certain amount of protection p to each RS coder packet, insert the coded source image blocks plus extra blank-filling protection symbols, and acquire the total stream length for each frame.

4 Check the buffer size constraint, and adjust the d_c accordingly.

5 Go back to step one, iterating until d_c stabilizes, and get the optimal coding configuration.

6 Use the resulting configuration to code and send the frames, employing real-time buffer size checking.

Finding the best starting point is once again crucial to reducing computational complexity. We estimate the starting d_c by using the buffer size constraint. We estimate the fixed amount of protection according to the channel situation E_b/N_0 and the predefined P_{pfail} threshold (the threshold could be adaptively adjusted on-line if the channel situation changed).

Figure 3.3 shows the block diagram of the system using a CBR source coder and a RS channel coder. The simulation results are shown in the Chapter 4.

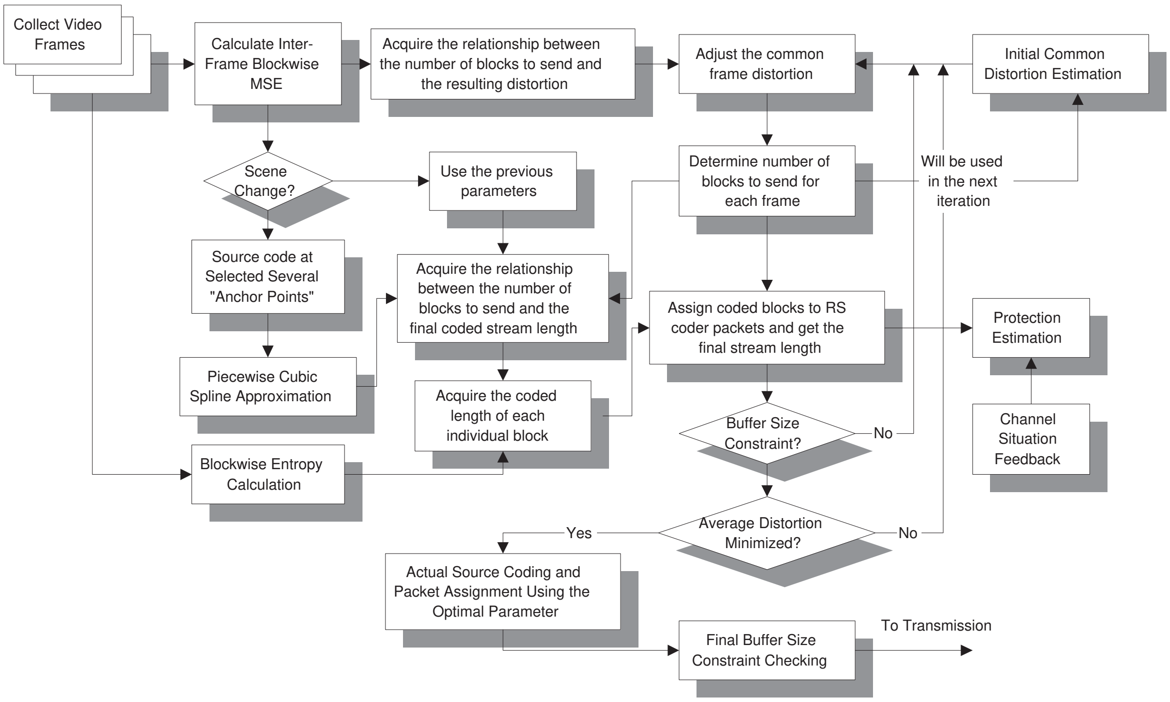

Figure 3.3 Conditional Block Replenishment and RS Coder

CHAPTER 4

RESULTS

In this chapter we present the simulation results of our two systems. The results show that our scheme truly works and gives excellent results.

4.1 Motion-JPEG and RS Coder

For the simulation, we use the following parameters: the testing sequence is the Football sequence (which is considered a difficult sequence), dimension 352×240; we use an RS coder size of 256 symbols (2048 bits), a decoder buffer size of 21 120 bits (approximately the size of two compressed frames), a constant decode flow of 15 840 bits (approximately 1.5 times the compressed frame size), and a GOP size of 20. The result is shown in Figure 4.1.

The result plot shows the following:

- Fixed amount of protection (no source-channel matching)

  1. If the amount of protection is relatively low, the system behaves badly under unfavorable channel situations and becomes better only when channel signal-to-noise ratio (SNR) gets very high.
  2. If the amount of protection is unnecessarily high, the system maintains constant performance under a wide range of channel situations, but cannot take advantage when the channel situation is good.

- Protection determined by source-channel matching

  We can clearly see that the optimal configuration curve is always above the fixed-protection curves under all channel situations.

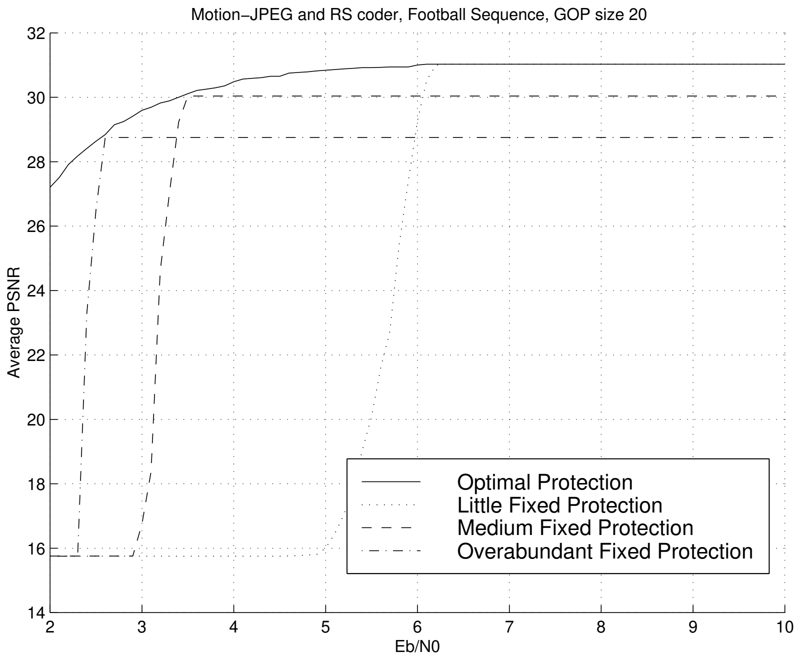

Figure 4.1 Motion-JPEG and RS Coder Simulation Results

4.2 CBR and RS Coder

For this simulation we use the same test sequence in Section 4.1; the quantization matrix we used is the standard default JPEG quantization matrix, and the result is shown in Figure 4.2 and 4.3. (As expected, the CBR coder requires much less bandwidth to achieve the same average peak signal-to-noise ratio (PSNR) than the Motion-JPEG coder. For example, with a PSNR of 30 dB, the CBR coder needs only 26 percent of the bandwidth required by the Motion-JPEG coder.)

From the average frame PSNR plot we can reach the same conclusion as in the Motion-JPEG case: our scheme truly minimizes the end-to-end distortion in all channel situations, whereas systems not employing joint source-channel matching will either behave badly in certain regions or fail to take advantage of a good channel. From the frame-quality-variation plot we can see that although our scheme does not always give the smallest variation, the variation is always below a certain acceptable level, and those configurations that give smaller variations always have a much greater average frame distortion; our scheme truly strikes a balance between the average frame quality and the

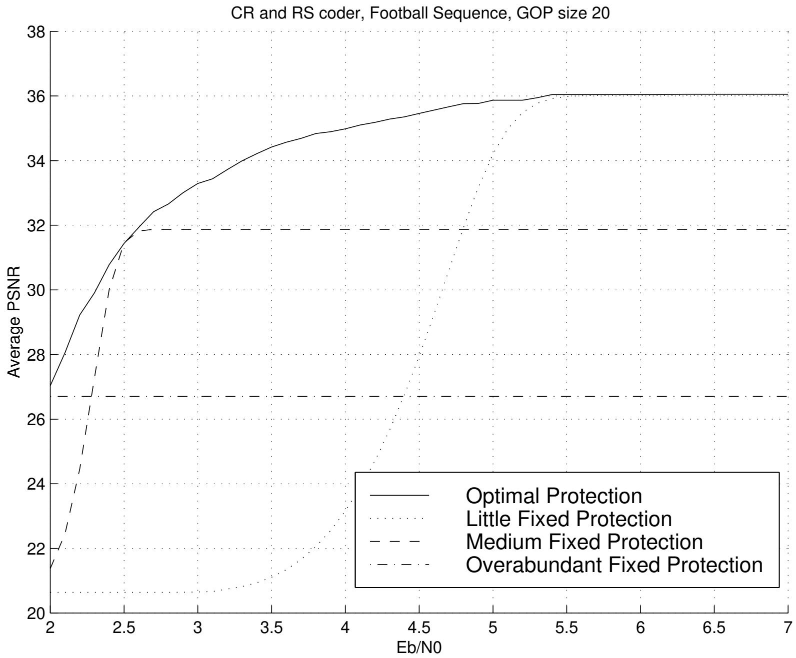

Figure 4.2 Conditional Block Replenishment and RS Coder: Average PSNR

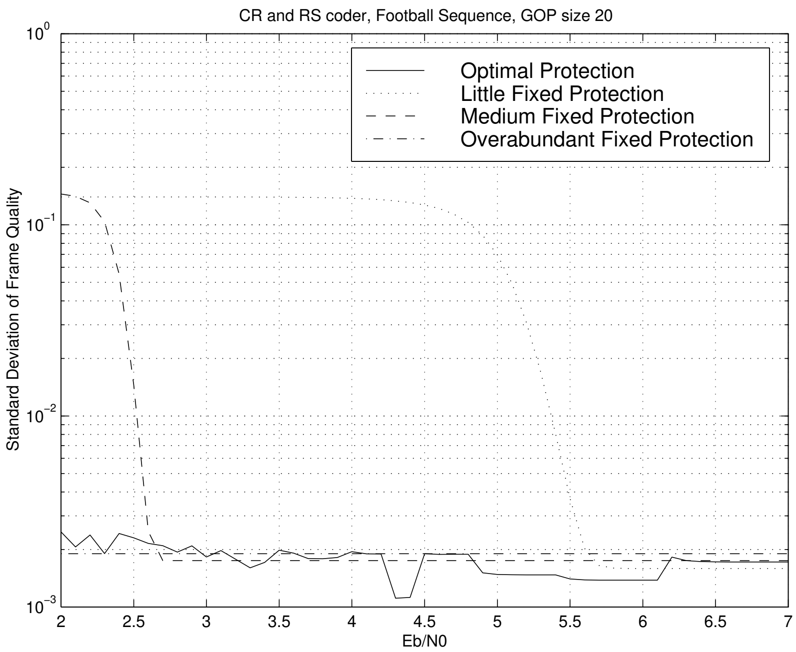

Figure 4.3 Conditional Block Replenishment and RS Coder: Frame Variation

quality variation. By putting different weights on these two criteria, we have another degree of freedom to adapt to user requirements.

CHAPTER 5

CONCLUSIONS

We have developed a general parametric-model-based real-time adaptive scheme to implement joint source-channel matching for wireless video transmission. The results show that our scheme makes a major difference in terms of performance when compared with fixed-parameter systems under varying channel conditions. Furthermore, compared with other current JSCM schemes, our method has the following advantages:

1. Generality: Our scheme can be implemented using most current source-channel coder pairs without detailed knowledge of the coders themselves, thanks to our parametric-model-based approach. To some extent, this approach solves the key problem faced by communication system designers: limited knowledge of available coders, which can be either standardized/proprietary, and the necessity to use preselected coders. Furthermore, our system can use many different optimization criteria (HVS, MSE, etc.)

2. Adaptivity: Again, due to our parametric-model-based approach, our method can track channel variation (except for rapid-varying channels) and yield the corresponding optimal coding configuration. The initialization procedure which we employ ensures that the optimization will be fast enough for channel tracking.

3. Real-time capability: Our system can be implemented using real-time hardware, a fact demonstrated by our two proposed system diagrams. The procedure we employ to obtain the parametric model (evaluating "anchor points" followed by interpolation) is relatively inexpensive. A hardware test bed is being developed concurrently to further test our scheme.

In future work, we will apply our scheme to motion-compensation-based video source coders such as MPEG and H.263, other channel coders such as RCPC coders, and other channel models such as fading channels. We will also compare the results of our general method to those that are more tuned to specific source-channel coder pairs.

REFERENCES

[1] S. B. Z. Azami, P. Duhamel, and O. Rioul, "Joint source-channel coding: Panorama of methods," in *Proceedings of CNES Workshop on Data Compression*, Nov. 1996.

[2] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[3] K. Ramchandran, A. Ortega, K. Uz, and M. Vetterli, "Mutilresolution broadcast for digital HDTV using joint source/channel coding," *IEEE Journal on Selected Areas in Communications*, vol. 11, pp. 6–23, Jan. 1993.

[4] T. Lan and A. Tewfik, "Adaptive low power multimedia wireless communications," in *Conference on Information Sciences and Systems*, 1997.

[5] G. Davis and J. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *SPIE Conference on Wavelet Applications of Digital Image Processing XIX*, Aug. 1996.

[6] S. B. Z. Azami, O. Rioul, and P. Duhamel, "Performance bounds for joint source-channel coding of uniform memoryless sources using a binary decomposition," in *Proceedings of European Workshop on Emerging Techniques for Communication Terminals*, Jul. 1997, pp. 259–263.

[7] B. Belzer, J. D. Villasenor, and B. Girod, "Joint source-channel coding of image with trellis coded quantization and convolutional codes," in *Proceedings of ICIP'95*, Aug. 1995, vol. 2.

[8] G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 72–81, Jul. 1997.

[9] H. Man, F. Kossentini, and M. Smith, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 8–11, Aug. 1997.

[10] J. L. Massey, "Joint source and channel coding," *Communications Systems and Random Process Theory*, vol. 11, pp. 279–293, 1978.

[11] S. Appadwedula, D. L. Jones, K. Ramchandran, and I. Konzintsev, "Joint source-channel matching for a wireless communications link," in *Proceedings of DCC 98*, Mar. 1998.

[12] S. Appadwedula, D. L. Jones, K. Ramchandran, and I. Konzintsev, "Joint source-channel matching for a wireless communications link," in *Proceedings of ICC 98*, Jun. 1998.

[13] L-J. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 3, pp. 665–676, Feb. 1998.

[14] L-J. Lin, A. Ortega, and C.-C Jay Kuo, "Cubic spline approximation of rate and distortion functions for MPEG video," in *Proceedings of IST/SPIE, Digital Video Compression: Algorithms and Technologies '96*, Feb. 1996.

[15] R. E. Blahut, *Digital Transmission of Information*, Reading, MA: Addison-Wesley, 1990.

[16] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 4–21, Apr. 1991.

[17] C-Y. Hsu and A. Ortega, "Rate control for robust video transmission over wireless channels," in *Proceedings of VCIP97*, Feb. 1997.

[18] A. Ortega and M. Khansari, "Rate control for video coding over variable bit rate channels with applications to wireless transmission," in *Proceedings of ICIP'95*, Oct. 1995.

[19] C-Y. Hsu and A. Ortega, “A Lagrangian optimization approach to rate control for delay-constrained video transmission over burst-error channels,” in *Proceedings of ICASSP98*, 1998.

[20] L-J. Lin, A. Ortega, and C.-C. Jay Kuo, “A gradient-based rate control algorithm with applications to MPEG video,” in *Proceedings of ICIP'95*, Oct. 1995, pp. 392–395.