Robust Mode Selection for Block-Motion-Compensated Video Encoding

by

Raynard O. Hinds

S.M., Massachusetts Institute of Technology (1995)
S.B., Massachusetts Institute of Technology (1995)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering
at the
Massachusetts Institute of Technology
June 1999

Author_____
Department of Electrical Engineering and Computer Science
May 17, 1999

Certified by_____
Jae S. Lim
Professor of Electrical Engineering
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Committee of Graduate Students

Robust Mode Selection for Block-Motion-Compensated Video Encoding

by

Raynard O. Hinds

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 1999, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Abstract

Block-based coders rely on motion-compensated block prediction with intra refresh to limit the distortion from channel losses over unreliable channels. We present an algorithm to select the coding mode for each macroblock to minimize the expected mean-square-error (MSE) between the original and decoded frame after concealment for a given rate constraint. The concealment method and the channel loss rate are known at the encoder. Video transmitted over a packet-switched network such as the Internet is one application discussed for robust video coding. For a given rate constraint, there is a tradeoff between compression efficiency and robustness to loss. The optimal mode selection is compared to several simpler schemes suggested in the literature. We present simulation results that show equivalent average PSNR can be achieved with the optimal algorithm at loss rates that are at least eight to ten percent higher than when the best ad hoc methods for robust coding are used.

The novelty of this approach is the inclusion of a simple yet relevant channel model and the concealment method to make the optimal mode selection at the encoder. We present results for one specific temporal concealment method. However, the algorithm can easily be extended to other general concealment methods. By looking to minimize the total expected MSE after concealment, the optimal bit allocation is determined as a function of loss rate. As the probability of error goes to zero, the results from our algorithm converge to that of the most efficient mode selection algorithm which results in the highest frame quality at no loss. The computation involved in our method is large. However, it can be used to evaluate the performance of practical ad hoc methods. It is sufficient to find a suboptimal method with performance near to the minimum MSE solution.

Thesis Supervisor: Jae S. Lim
Title: Professor of Electrical Engineering

Dedication

To

my parents,

Aubrey and Joyce Hinds

Acknowledgements

There are many people I want to acknowledge here who have helped make all of this possible. I first want to thank my thesis supervisor Prof. Jae Lim for giving me both the opportunity and guidance necessary to complete my thesis. I also want to thank my readers: Dr. Thrasyvoulos Pappas, Prof. David Staelin, and Dr. Dan Dudgeon for their useful feedback on my thesis. I would like to give special thanks to Thrasos for his sincere interest and guidance in both my thesis and my professional development. The completion of a Ph.D. thesis is hard work and perseverance combined with faith, and I want to thank him for giving me encouragement to stay the course. I also want to especially thank my family for their continuous love and support through thick and thin. Not enough can be said about the sustaining powers of unconditional love. I am happy that I can now without hesitation answer the question "When do you think you will be done with school?" by referring to the date printed on the front of this document.

I also want to take this time to thank the members of the Advanced Television Research Program (*ATRP*). I want to say thank you to the group administrative assistant Cindy LeBlanc for her reliable help and willingness to take care of all the things that need to be done to keep this lab running smoothly. I want to also acknowledge my friends and current (soon to be former) Ph.D. students David Baylon, Eric Reed, and Wade Wan for the countless beneficial discussions on anyone's thesis and useful feedback on my thesis. I want to give special thanks to Eric and Wade for helping to lighten things up in the lab as well as expand my knowledge of things outside the Ph.D. program. Through Eric, I developed my Wall Street Savvy. Well, I at least learned what an option is. Now, I know what I can do with my money once I begin to earn it, and I can thank (blame) my future financial success (failure) on him. For any sports fan, knowing Wade is better than having any sports almanac. Wade was always able to fill me in on the score, the ranking, draft pick, seed, or any sports team statistics. Friends like this made this whole experience more enjoyable (Well, at least bearable).

Acknowledgements

There are many wonderful things that can be done in life without money. Unfortunately, this is not one of them. So, I would like to thank my sponsors Lucent Technologies (Formerly AT&T) Bell laboratories Cooperative Research Fellowship Program. Without programs like this, the graduate program would be much more difficult than it already is. We need all the encouragement and support we can get.

Finally, I would like to thank some former *ATRP* students for their help. I would like to thank Shiufun Cheung for providing the thesis template which beautifully formatted this thesis. All I had to do was fill in the text of my thesis (<insert text here>). This saved me days possibly weeks of work and additional headaches. I would also like to thank John Apostolopoulos for his willingness and ability to always give insightful feedback when asked.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of digital video coding is prevalent in many applications available today. The bit rate of uncompressed digital video is often too high for efficient storage or transmission. Therefore, compression of the video signal is required to reduce the coded bit rate. Moreover, lossy compression methods must often be utilized which will result in distortion in the decoded video sequence. In addition, since there is typically a large amount of temporal redundancy in video sequences, the use of prediction can also be effective in achieving compression. A video component can often be predicted well from a related decoded element that was previously encoded in the bit stream. For example, a region to be coded in the current frame can be predicted from the same scene information located in the previous decoded frame. When this occurs, only the difference between the original signal and the prediction must be coded. When a good prediction is used this difference will be small and will usually require fewer bits to code when compared to coding the region without prediction. Therefore, more bits can now be devoted to code the region with lower distortion for a given bit rate constraint.

For a block-based coder, the chosen coding mode will designate whether or not a block region is to be coded with prediction. Optimal coding mode selection can produce a decoded video signal with highest fidelity, since it will lead to the most efficient bit allocation. The problem of optimal coding parameter selection for block-based video coders has been well studied in the literature[1, 2, 3, 4]. Traditionally, this problem has been addressed without much consideration of channel loss or errors. There are, however, many harsh transmission environments which are susceptible to unintended data loss. Since the use of prediction creates a dependency between the prediction region and the coded prediction error, it would be very detrimental to apply the same coding strategies developed for reliable channels when coding for this environment. This will result in too much predictive coding. Error persistence will occur, since the assumptions made at the encoder about what is available at the decoder are not always true due to channel loss. More robust coding methods are required for lossy environments.

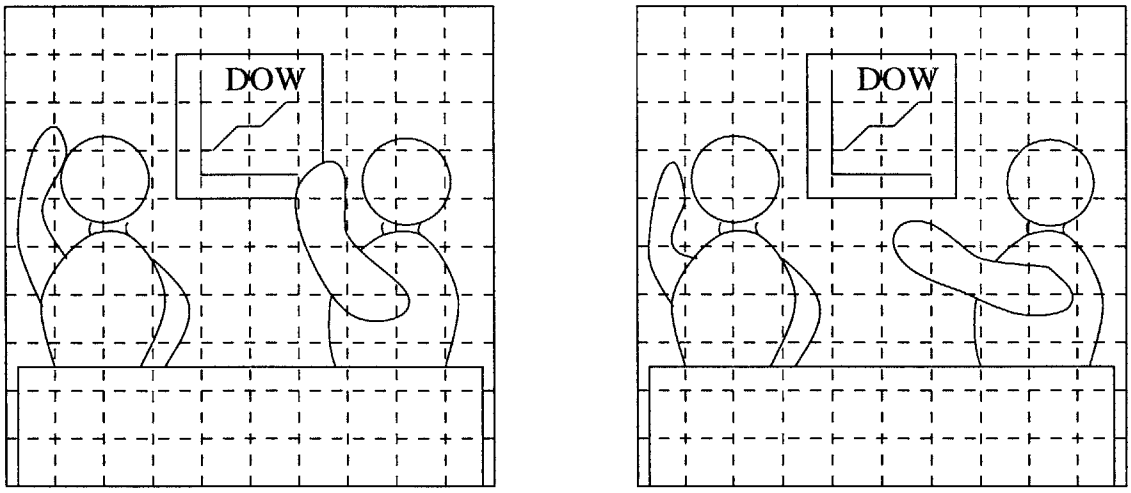


Figure 1.1: Block-based video coding.

When channel loss has been considered in the literature, mainly ad hoc approaches have been proposed to limit the possibility of error propagation. This thesis develops a methodology to select coding modes for block-based coders to achieve optimal operational rate-distortion performance over a lossy channel. A simple yet relevant model for loss is considered, and mode selection is performed to minimize the mean-square-error after concealment for a given bit rate constraint. We restrict ourselves to causal optimization performed at each frame. This is appropriate for real-time applications. In this introduction, we will briefly describe the coding problem for block-based coders. In addition, we will present an example which requires robust coding methods as well as outline techniques designed to address this issue. Some of these methods will be further explored later in this thesis. We will end with an outline of the thesis.

1.1 Review on Robust Block-Based Coding

Block-based coders are used in several current video coding standards. With these coders, each video frame is divided into non-overlapping block regions as shown in Fig. 1.1. The advantage of doing this is twofold. The first is to effectively exploit the spatial correlation that

exists in many video frames for more compression. Neighboring pixels tend to be similar in color and intensity and can be compressed more effectively when coded together. However, the extent and shape of each correlated region within each image frame is unknown a priori. Good compression performance can be safely achieved by first limiting the size of the region over which correlation is assumed. In addition, there are many efficient spatial transformations that have been developed for simple rectangular regions which can be used to compress correlated pixels into a smaller number of transform coefficients. These coefficients are not kept at full precision, but are quantized for more compression. This part of compression is lossy and will result in some distortion in the decoded video frame.

The second advantage to block-based coding is that it allows for adaptability, and each block can be coded in the most efficient manner according to the particular coding standard options. For example, typical video sequences contain some of the same scene information in several consecutive video frames. This can be seen by the stationary background visible in both frames in Fig. 1.1. These regions should be coded using prediction from the matching block-shaped region in the previous decoded frame. When the best prediction region comes from a shifted position in the previous frame, a motion vector for this displacement is coded along with the block. Since the block-shaped region does not directly correspond to individual objects and some distortion is produced from compression in the previous decoded frame, the prediction is often not perfect. Thus, a difference is taken between the prediction region and the original block resulting in a residual error to be coded. On the other hand, there are some regions in the current frame which are not predicted well from the previous frame. This is true for regions that are uncovered in this frame or for regions with non-translational motion such as rotations or zoom. These blocks should be coded in intra mode without using prediction. Because of the overhead involved in transmitting coding mode and motion vector information for each block, it is not efficient to perform mode selection on as small a block as is used for spatial compression. Thus, a 2x2 square of neighboring blocks are grouped together to form a macroblock for mode selection. A single coding mode and motion vector if necessary are assigned to the four block regions in the macroblock.

The number of bits needed to code a macroblock is determined by both the coding mode selected for the macroblock and the quantization step-sizes used for the spatial transform coefficients of the blocks inside the macroblock. Exactly how to choose these coding parameters is not determined by any of the coding standards. Instead, they only specify the range of valid parameter values and how they should be encoded in the bit stream for proper

decoder operation. The coding parameters for the macroblocks should be chosen to result in the most efficient coding scheme that meets all of the constraints. Bits are usually not allocated on a macroblock by macroblock basis. In this thesis, we assume some bit budget is allocated across each frame. There is a trade-off between the coding rate and the resulting distortion in the decoded video sequence. For rate-distortion optimization, the video sequence should be coded at the lowest distortion that meets the rate constraint. This is a difficult problem to solve completely because of the dependencies that exist between the quantization step-sizes, coding modes, and resulting bit-rates for macroblocks. In past work, simpler problems have been investigated [1, 2, 3, 4].

There is additional correlation between the coding parameters for neighboring macroblocks that can be exploited for more compression. In this case, it is more efficient to use the parameter from the previous coded macroblock in raster scan order as a prediction for the current macroblock parameter and only code the difference between the two. This is very useful when coding the motion vectors for neighboring macroblocks coded in inter mode which belong to the same moving object.

When there is a bit budget, the use of prediction in coding will result in more efficient coding. This will allow for more bits to be used to code each frame at higher quality reducing compression distortion. However, when loss occurs the use of temporal prediction produces persistent errors that may even spread throughout the following decoded frames. An example is shown in Fig. 1.2 where a region in frame $t-1$ that will be used as a prediction for a macroblock in frame t is lost and replaced by a gray block. We designate these two regions by a surrounding bold borderline in each frame. Using prediction here will also cause an error to occur in frame t even when none of the coded blocks for that frame are lost. For this example, this region is only used to predict the macroblock at the same spatial location in frame t. In general, overlapping prediction regions may exist and other macroblocks may use parts of this same lost region for prediction. In this case, this error in frame $t-1$ will spread to these other macroblock locations increasing the size of the error region. On the other hand, the other lost macroblock in frame $t-1$ is not used for prediction and that same area is intra coded in the following frame. Therefore, this error from loss is short-lived. It is clear that a tradeoff exists between compression and robustness to errors. Methods which allow for redundancy—therefore, less efficiency—may have to be used to ensure robust video coding. In what follows, we briefly describe an example of an unreliable channel as well as give a brief overview of robust methods that have been attempted in the literature.

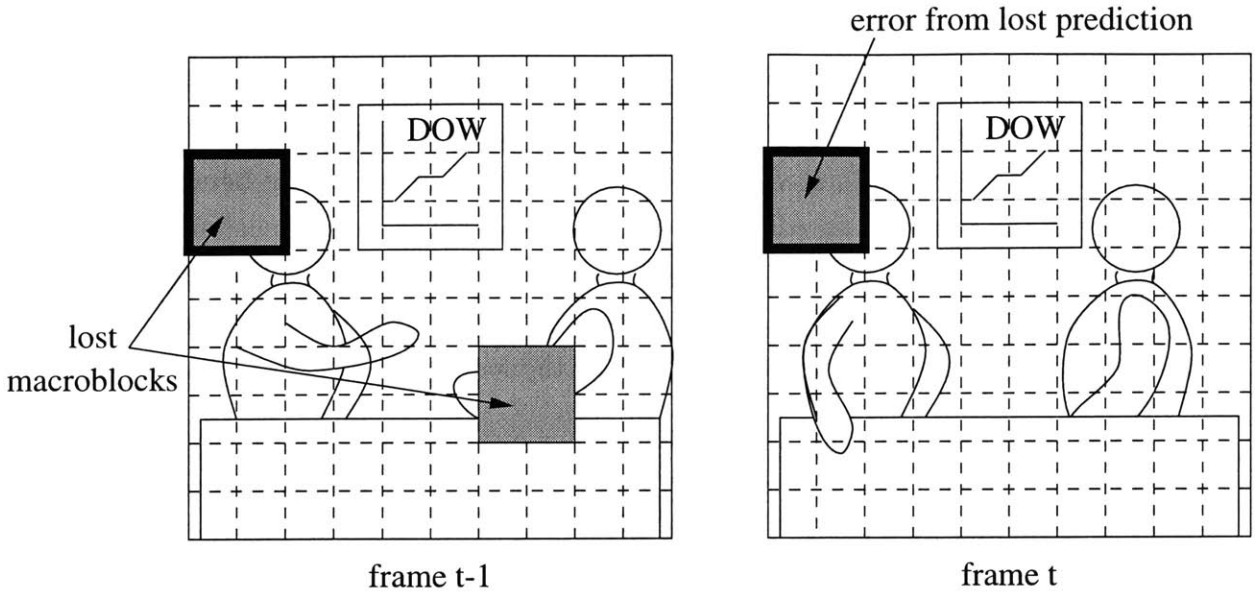


Figure 1.2: Bold square region in frame $t-1$ is used to predict bold square region in frame t. Since prediction region is lost, error propagation occurs in frame t even though none of the macroblocks transmitted for frame t are lost.

One example for the need of robust coding occurs when real-time video is transmitted over a shared resource without adequate quality of service (QOS) guarantees. This is common, for example, in the case of transmission over the Internet or in Asynchronous transfer mode (ATM) networks. Packet loss on congested networks is a major problem for real-time or near-real-time applications. The user is usually forced to send at a very low bit-rate in order not to exceed the available bandwidth. It is safer to have all information loss occur at the source coder where the user can control what is lost with compression. With network transmission, optimal coding methods now have to consider minimizing the distortion from packet loss along with the distortion from source coding. The coding rate is increased to allow the decoder to recover from packet loss and to effectively conceal errors. However, any realistic variable bit rate transmission environment will impose some rate constraint on the video encoder [5].

A common approach to error correction in networks is Automatic Repeat reQuest (ARQ). With ARQ, packets that are lost are retransmitted to the receiver. Depending on the exact protocol implemented in the network, lost packets are signaled to the transmitter

either by lack of an acknowledgment packet from the receiver or the reception of a negative acknowledgment packet which is only sent when a missing packet sequence number is detected at the receiver. *ARQ* is currently used on the Internet for error correction for the normal delay insensitive transmitted data. For data with real-time requirements, *ARQ* can suffer from excessive delay and is limited by the round trip transmission time between the sender and receiver in the network. In addition, *ARQ* can play havoc in multicast situations where one sender is transmitting to several users across different connections with different losses. The resulting increase in network usage from negative acknowledgment packets will only make the congestion problem worse and increase the loss rate.

Forward Error Correction (*FEC*) is another method used in unreliable transmission environments for error control. With *FEC*, redundant information is sent to allow for the recovery of a limited amount of lost data without having to retransmit data. Because of the burst of consecutive bits that is lost with each lost packet, it is difficult to design error correction codes that can recover lost data, and interleaving must often be used to allow for recovery in even modest loss scenarios. However, the biggest argument against *FEC* is that the goal of *FEC* is too rigid, since video unlike normal data can tolerate loss and concealment methods can be very effective at the receiver. The lost bits do not have to be recovered precisely.

There are many error concealment methods for video which have been proposed in the literature [6, 7, 8, 9, 10, 11]. These methods typically involve spatial or temporal interpolation. Robust encoding practices along with reasonable error concealment can be sufficient in reducing the undesirable effects from packet loss and produce adequate video quality. A significant detriment to concealment is the temporal prediction used in inter-frame coding with block-based coders. Both the elimination of redundancy along with the increased dependency among coded macroblocks makes lost macroblocks more difficult to conceal and the video sequence more susceptible to loss. The use of temporal prediction in video coding should be limited to make the video more robust in the network.

To reduce error propagation and improve error concealment at the receiver, one very conservative encoding strategy would be to restrict every macroblock to intra-frame encoding. The benefits of this are twofold. First, at a high enough frame rate, any loss error that occurs will at best be short-lived. Second, the effective error rate is reduced since the error propagation associated with predictive coding is eliminated. However, there is a drawback to only using intra-frame encoding when there is a bit rate constraint. There is a bit allocation

problem, and the increase in robustness to loss with only intra-frame encoding comes at the expense of degradation in either frame quality with an increase in quantization step-size or motion rendition with a decrease in frame rate.

A better robust method uses block-based conditional replenishment to regain some of the efficiency of temporal prediction when there is not too much motion in the scene [12]. With conditional replenishment, the sum of absolute difference between the macroblocks in the current and previous frame at the same spatial location is computed and compared with a threshold T. If this difference is below the threshold T, the macroblock is not transmitted and is replaced at the decoder with the macroblock in the previous frame at the same spatial location. Otherwise, the macroblock is intra coded and transmitted. This is advantageous for scenes with stationary background, since not all of the macroblocks have to be sent. This will free up more bits which hopefully will prevent excessive degradation in frame quality and motion rendition. Overall, even with conditional replenishment, intra-mode encoding alone is still much less efficient than using predictive encoding.

A final method used to encode video with a block-based encoder allows for temporal prediction with inter-frame encoding for more efficiency. However, for robustness to loss, a threshold U is set to force a minimum periodic intra-coded macroblock update rate at each macroblock spatial location in the frame[13]. The threshold is set ad hoc and based on the observed or known loss characteristics of the channel.

These coding strategies provide for robustness to loss. However, how often a macroblock at each spatial location in a frame should be forced to be intra coded as a function of the loss characteristics given for the network is still an open issue. This thesis addresses that issue and presents an algorithm for mode selection which is optimal in the mean-square-error sense for random macroblock loss. This will determine how to allocate bits between macroblock intra-mode encoding for robustness to loss and smaller quantization step-sizes for better picture quality to achieve the best overall video quality at the receiver after loss and concealment. For tractability, we re-formulate the problem for the quantization step-size selections over a frame. This new formulation is still general and will allow as a special case for consistent frame quality to be set for all macroblocks in a frame when no loss occurs.

1.2 Thesis Outline

In Chapter 2, we discuss the application of coding video with a block-based coder to be transported over the current Internet in real-time. We begin with a description of the general source coding algorithm for block-based video coders. This algorithm has many coding parameters that are not set by the coding standards. They are free for the user to choose and determine the rate-distortion performance of the video coding algorithm. We present an algorithm for optimal coding performance when the potential for loss is not considered. Next, we describe how the Internet works and how it can be used to transport real-time video. We end by discussing some of the problems of coding video with the current Internet standard.

In Chapter 3, we continue using the network as an example of an unreliable channel and give some background on current methods used and proposed to increase the robustness of video transport in this environment. This includes improvements in network protocols, methods to deal with loss at the receiver, and simple robust encoding algorithms. We will discuss the remaining drawbacks to these techniques and suggest how possible improvements can be made.

In Chapter 4, we formulate an analytical problem for robust video coding and present an algorithm to find the optimal solution in the mean-square-error sense. We show simulation results when loss is applied to sequences coded with our proposed algorithm and compare it to previous mentioned algorithms for identical loss rates. We show that using our proposed algorithm will result in smaller error after concealment.

Chapter 2

Real-time Video Over the Internet

The Internet is a packet-switched network that provides a good example of an unreliable channel. In this chapter, we describe a system for coding and transmitting real-time video over the Internet. Video that is coded and transmitted over packet-switched networks is often referred to as packet video in the literature. When packet-switched networks are used for real-time applications such as video conferencing, one has to deal with the effects of channel errors in the form of lost packets. Packet losses are mainly due to the fact that the network is a shared resource with limited bandwidth. In addition, real-time video has strict timing requirements. As a result, there are several problems in coding and transmitting packet video using the current Internet protocols.

We begin in the next section by describing the essential parts of the block-based source coding algorithm used in many current video coding standards. There are several coding parameters that must be chosen that determine the performance of this coding algorithm. To help setup the algorithm presented in chapter 4 for robust video coding, we will present an algorithm to select efficient coding parameters without consideration of the potential for channel errors. We will also show a near-optimal solution to this problem which requires much less computation. Following that section, we talk about packet-switched networks and explain how the Internet works. We next discuss how real-time video is currently transmitted over the Internet, and end this chapter with a summary of the key problems that need to be addressed.

2.1 Block-Based Video Source Coding

Many current video coding standards are made up of block-based video coders. This includes for example the Moving Picture Expert Group coders $MPEG$–1 [14] and $MPEG$–2 [15] and

the International Telecommunication Union-Telecommunication $ITU-T$ recommendations $H.261$ [16] and $H.263$ [17] . In what follows, we describe the features of block-based coders that are relevant to this thesis and common to all of the aforementioned coding standards.

2.1.1 Overview

The ultimate goal of video coders is to reduce the bit rate required to represent the original video sequence with minimal distortion. Compression is accomplished both by discarding perceptually irrelevant information and exploiting the high degree of correlation that exists in video sequences. For example, the high spatial frequency information in a single frame can not be perceived by the eye. This information does not need to be coded. In addition, there are both temporal correlation from the same scene information that is captured over several successive frames as well as spatial correlation that exists within each image frame. Ideally, a compression algorithm would de-correlate a video sequence and only transmit the non-redundant information.

The greatest amount of redundancy is typically along the temporal dimension in video sequences. Consequently, this provides the largest potential for bit rate reduction. A region to be coded in the current frame can be predicted from information in the previous decoded frame which has already been transmitted to the receiver. A copy of this decoded frame is stored at the encoder for this purpose. Each frame in a video sequence to be coded is partitioned into M non-overlapping macroblocks. The size of a macroblock is 16 x 16 pixels for typical block based coding standards. Motion compensation and temporal differential pulse coded modulation ($DPCM$) can be performed between the current macroblock to be coded and a decoded region in the previous frame. This produces a prediction error block with smaller variance than the original macroblock which will result in higher compression. An example of this process is shown in Fig. 2.1. This is known as inter frame coding mode. A dependency is created between the decoded region used for prediction and the prediction error that is transmitted. Both are needed to reconstruct the current macroblock.

The best prediction for a macroblock may come from a shifted location in the previous frame due to object motion. Thus, motion compensation is performed for each macroblock before taking the difference. The associated motion vector and macroblock residual signal are grouped together to form the coded macroblock transmitted to the receiver. Macroblocks

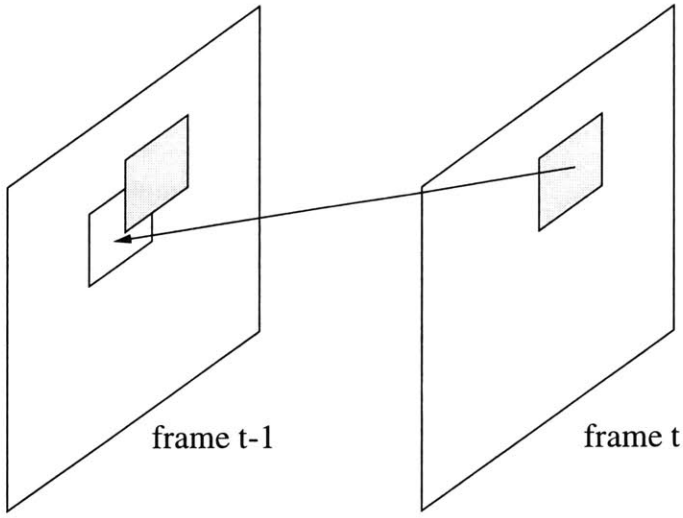


Figure 2.1: Motion compensated prediction.

are the smallest individual coding units in a frame. However, there is still some residual redundancy between the coded macroblocks which should be eliminated for more compression. For example, the motion vectors of neighboring macroblocks are correlated since they typically belong to the same moving object. Motion vectors are differentially encoded as the macroblocks are coded in raster scan order. This creates dependency between neighboring macroblocks. The macroblocks are further grouped into regions which break this dependency called group of blocks GOB (or slices in the $MPEG$ terminology). Differential coding is not permitted across the GOB boundaries. This creates larger independent coding units and is part of the layered structures used for most block-based video coding algorithms.

A good prediction region may not always be found for every macroblock in a frame. A poor prediction might result for uncovered regions occluded in the previous frame. An extreme example of this is when there is a complete scene change. Another reason for poor prediction occurs when the true motion is not approximated well by translational motion within the image frame. It is more efficient to code the macroblocks in these scenarios without prediction. This is known as intra frame coding mode since it does not reference any information in other frames. The standards do not specify how to make the coding mode choices for all macroblocks. The group of picture (GOP) structure in the $MPEG$ coding algorithms will specify complete frames that are forced to be intra coded. These frames are known as I- frames and are periodically dispersed throughout the sequence. The

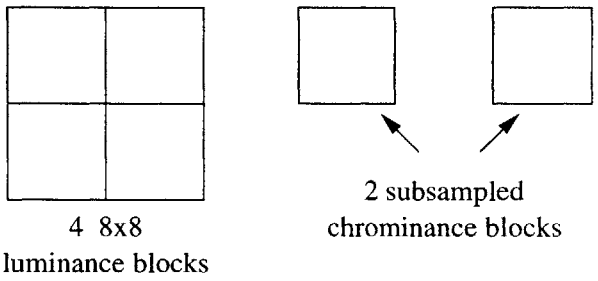


Figure 2.2: Arrangement of blocks in macroblock.

frames in between I- frames are free to use prediction to code macroblocks and are known as P- frames. The $ITU-T$ recommendations $H.261$ and $H.263$ do not specify intra coded frames, and all frames after the first coded frame are essentially P- frames.

Spatial de-correlation is accomplished using an orthogonal spatial transformation. This transform is applied to the original signal or the residual signal depending on whether the macroblock is coded in intra or inter mode. The extent of correlation in a frame is unknown, so the transform is applied over a small block region. Each macroblock consist of smaller blocks of luminance and chrominance components usually 8 x 8 in size. The information in both chrominance channels has lower bandwidth, and can be down-sampled by two in each direction. This results in half as many chrominance blocks as luminance blocks for a total of six blocks per macroblock. Figure 2.2 shows the arrangement of the blocks within a macroblock. The 8 x 8 Discrete Cosine Transform (DCT)[18] is applied to these smaller blocks for energy compaction. This process transforms the 64 pixels in the input block into 64 coefficients with less correlation. Each coefficient is related to a single spatial frequency in the macroblock. There is a single DC coefficient for zero frequency and 63 AC coefficients for higher frequencies. For correlated inputs, most of the energy is now concentrated in the low frequency coefficients. The most significant energy is in the DC coefficient of intra coded macroblocks. With full precision arithmetic this process is completely reversible by applying the inverse DCT transform (DCT^{-1}). These coefficients are then quantized (Q) and entropy encoded using variable length codewords (VLC) for more compression.

Quantization is a lossy process that usually produces distortion in the reconstructed image frame at the decoder. More compression can be achieved by not keeping the trans-

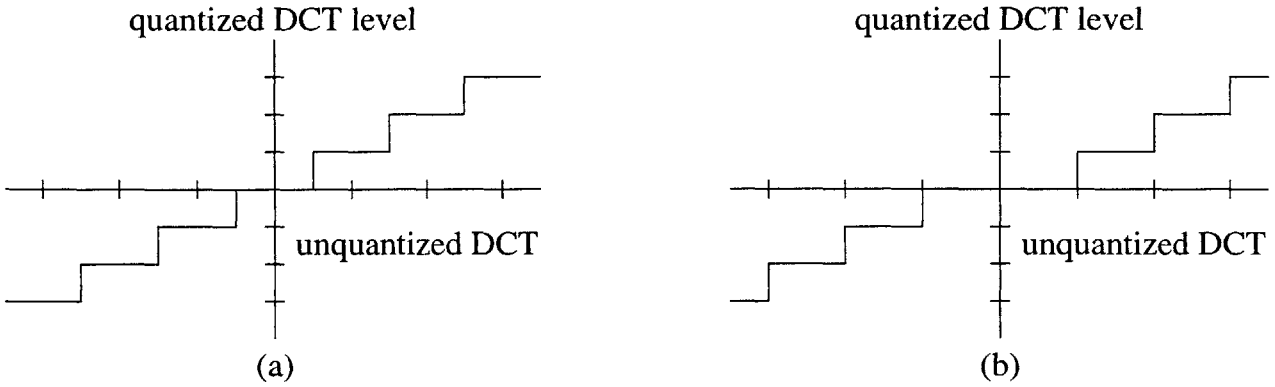


Figure 2.3: Quantization of DCT coefficients (a) without dead zone (rounding) (b) with dead zone (truncating).

form coefficients at full precision. Each coefficient is uniformly quantized to a particular reconstruction level. Except for the *DC* coefficient of intra-coded macroblocks, the spacing of the reconstruction levels for the remaining frequency coefficients is affected by the value of the quantization parameter. *H*.261 and *H*.263 use a single quantization step-size equal to twice the value of the macroblock quantization parameter for all of these frequency coefficients. Whereas, the *MPEG* coders allow each of these frequency coefficients to be quantized differently by scaling a quantization table with different values for each frequency by the quantization parameter. In this manner, larger step-sizes can be used to quantize higher frequencies which have less perceptual relevance.

Quantization is done by dividing the *DCT* coefficient by the appropriate quantization step-size and either rounding or truncating the result. For the *MPEG* coders, rounding is used for intra coded block coefficients, and truncation is used for inter coded block coefficients. For the $ITU-T$ recommendations, the *DC* coefficient for the intra coded block is rounded. The remaining coefficients are truncated. Truncating produces a dead zone around zero. This will potentially set many irrelevant coefficients to zero which can usually be coded with a lower bit rate. The difference between the two methods for quantization is shown graphically in Fig. 2.3. As the *DCT* coefficients are scanned in zigzag order from lowest to highest spatial frequencies as shown in Fig. 2.4, it is typical to find a non-zero coefficient followed by a run of zeros. As a result, run-length-coding provides an efficient intermediate representation before entropy coding is performed. In contrast to coefficient quantization, entropy encoding is lossless and completely reversible at the decoder. Variable length code-

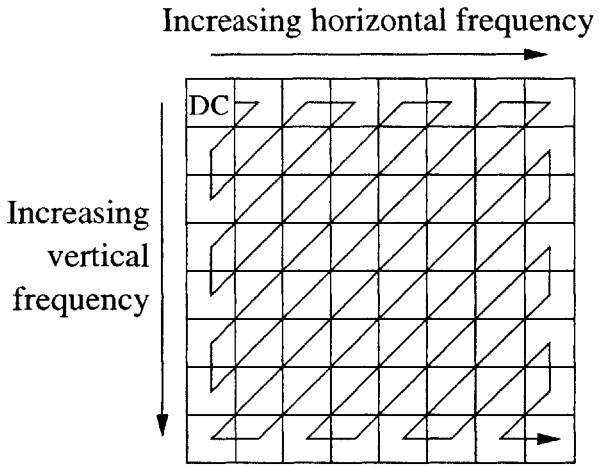


Figure 2.4: Zigzag coding order of DCT coefficients.

words are used to approach the entropy of the signal to be coded. Here, the goal is to assign shorter length codewords to more probable events. Each coding standard defines the set of codebooks that will be used.

The complete diagram of the block-based coding algorithm is shown in Fig. 2.5. To implement this algorithm, the macroblock coding modes and the quantization step-sizes must be chosen. Some coding standards offer additional modes which are slight variations of the two basic modes discussed above and are particular to the specific coding standard. In this thesis, we do not consider these additional modes since they are not generally applicable. However, the principles developed in this thesis could be extended to cover additional modes. For many practical video coding situations, a bit rate constraint is also placed on the final coded bit-stream. Therefore, efficient selection of these coding parameters is important. In the next subsection, we describe a method to select these parameters for efficient coding when there is no potential for loss.

2.1.2 Selecting Efficient Coding Parameters

We consider a video sequence which is captured and coded with a block-based coder. Each frame in the sequence is partitioned into M macroblocks. Let X_{tj} denote the original macroblock to be source coded in the t^{th} frame at spatial location j, and $\boldsymbol{X} = \{X_{tj} \ \forall \ t, j\}$.

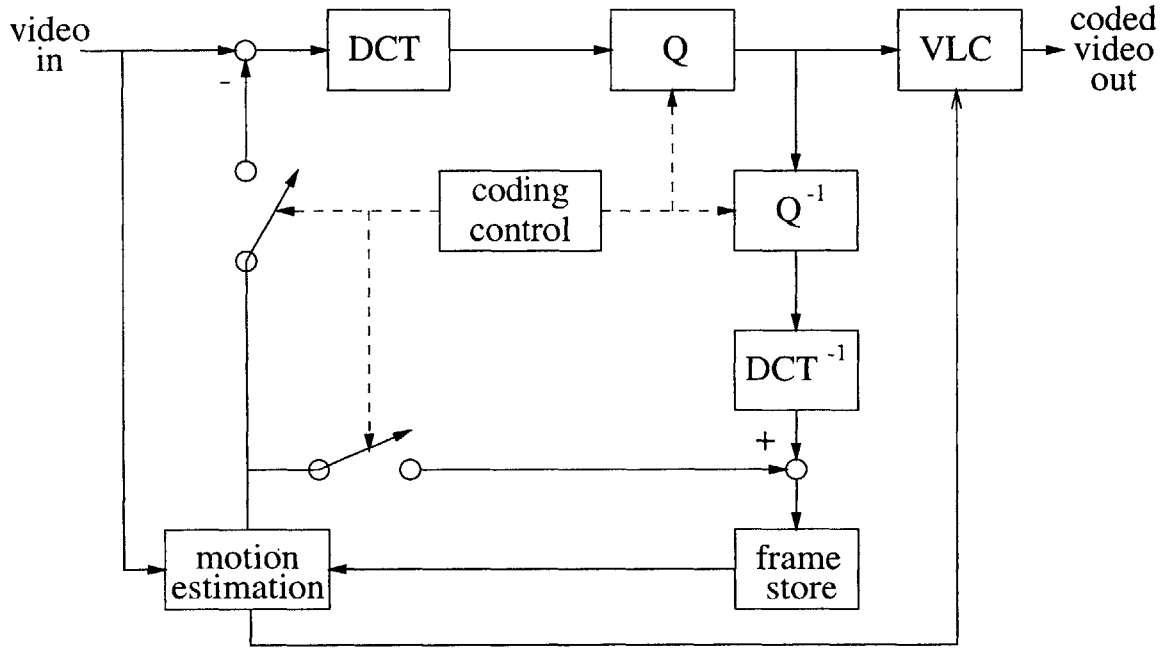


Figure 2.5: Block diagram of block-based source coder.

Let Y_{tj} denote the result of decoding the compressed macroblock X_{tj} at the source, and $\boldsymbol{Y} = \{Y_{tj} \; \forall \; t, j\}$. A rate constraint R_c is given, and the coding parameters for each macroblock must be chosen for optimal rate-distortion performance. The parameters for each macroblock X_{tj} are

q_{tj}	the quantization parameter
l_{tj}	the coding mode
$\boldsymbol{v_{tj}}$	the motion vector for inter mode coded macroblocks

The quantization parameter is constrained to be the integer values $q_{tj} \in \{1, 2, \ldots, |Q|\}$. For typical video coders, $|Q|$ equals 31. For any block-based coding algorithm, each macroblock is coded in one of $|L|$ possible modes, and the value of $l_{tj} \in \{0, 1, \ldots, |L| - 1\}$. As discussed above, we only consider the case $|L| = 2$ for intra mode ($l_{tj} = 0$) when the macroblock itself is coded and inter mode ($l_{tj} = 1$) when motion compensated prediction is used. The motion vector $\boldsymbol{v_{tj}} = (v_{tjx}, v_{tjy})$ determines the region from the previous decoded frame used to predict the current macroblock to be coded. The parameters v_{tjx} and v_{tjy} are the horizontal and vertical pixel displacements respectively between the spatial location of the current

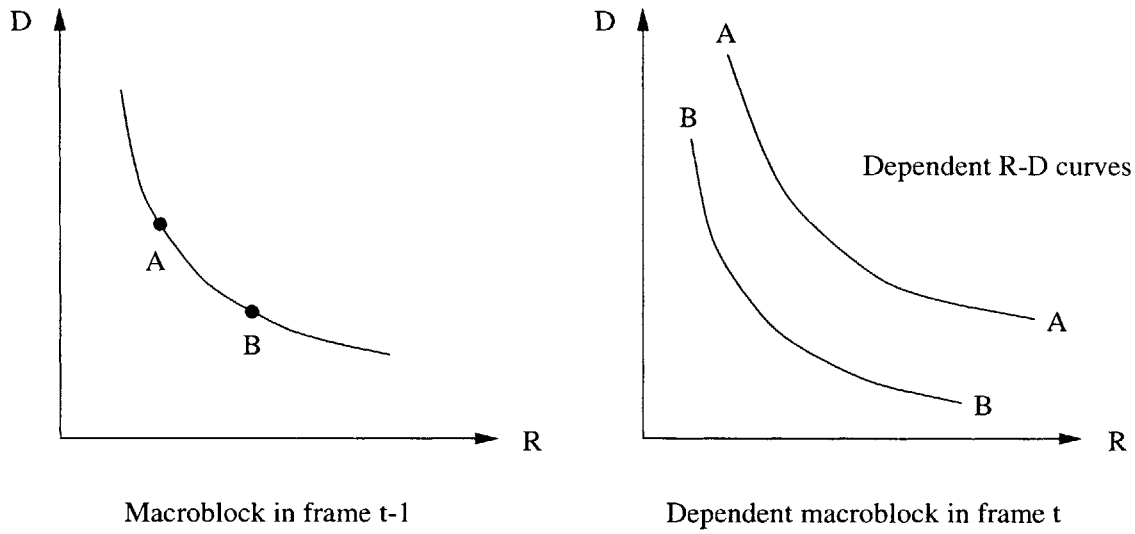


Figure 2.6: Dependent rate-distortion curves.

macroblock and the prediction region in the previous frame. They are usually estimated using a Block Matching Algorithm (BMA) with alternative metrics for block correlation such as mean-square-error (MSE) or mean absolute error (MAE).

Solving the joint problem of quantization parameter and mode selection for optimal source coding has proven to be very difficult. This is because of the dependency of a macroblock's rate and distortion function on its coding parameters as well as those for other macroblocks in the sequence. For example, the operational rate-distortion curve for a macroblock coded with prediction depends on the resulting rate-distortion points chosen for the macroblocks in the prediction region in the previous frame. An example of this is shown by the graphs in Fig. 2.6 where a single macroblock in frame $t-1$ is used to predict a macroblock in frame t. The operational rate-distortion curve for the macroblock in frame t depends on which of the two operating points (A or B) shown in this example is selected for the macroblock in frame $t-1$. We use the term operational rate-distortion curve to designate the achievable minimum distortion as a function of rate over the possible parameter selection values for a particular coder. This is different from the information theoretic rate-distortion curve which denotes the minimum distortion achievable as a function of rate over all possible coding methods.

When selecting coding parameters for block-based video coders that use prediction, the dependent quantization among frames should be taken into consideration. For this case, an algorithm is presented in [2] to find the quantization parameters to minimize a distortion function for a sequence coded with a maximum rate constraint. Only a single quantization parameter is considered for each frame. This means that either the same quantization parameter is used for all the macroblocks in the frame or the parameter can indicate a vector of macroblock quantization parameters among a set of possible choices. Even with this simplification, the dependency among frames is enormous and a monotonicity assumption is made to reduce the amount of computation required resulting in a near-optimal solution. We refer the reader to [2] for details of this particular algorithm. In this thesis, we only consider causal processing of frames and the optimal parameters must be chosen for the current frame given the already selected parameters for the previous frames. This is appropriate when there is a delay constraint. Below, we show how to choose the optimal parameters to minimize the distortion over a single frame for a given rate constraint. We first present an algorithm which is guaranteed to find the optimal solution. We then present a very simple algorithm which will find a near-optimal solution.

We use the MSE between the original macroblock X_{tj} and the decoded macroblock Y_{tj} at the source as a metric for the macroblock distortion. Given the selected coding parameters for the previous frames and the motion vector v_{tj}, each macroblock distortion is a function of the coding parameters for the particular macroblock.

$$D_{tj}(l_{tj}, q_{tj}) = \|X_{tj} - Y_{tj}\|^2 \tag{2.1}$$

The distortion function that we want to minimize for the whole frame can be expressed as the sum of the distortion for each individual macroblock as shown below.

$$D_t(l_{t1}, q_{t1}, , l_{t2}, q_{t2} \ldots, l_{tM}, q_{tM}) = \sum_{j=1}^{M} D_{tj}(l_{tj}, q_{tj}) \tag{2.2}$$

In contrast to the distortion, the rate to code a particular macroblock is typically also dependent on the parameters to code the previous macroblock in raster scan order in the same GOB or slice[1]. There are two reasons for this. The first is from the differential en-

[1]For the remainder of this thesis we will drop the reference to the term "slice" since the GOB and slice structure are conceptually identical

coding described in the previous sub-section. For example, the motion vectors for successive inter-coded macroblocks are coded differentially as well as the DC coefficient for successive intra-coded 8 x 8 blocks. The second reason for the dependency comes from the way the quantization parameter is coded into the bit stream. The quantization parameter is coded at the beginning of the GOB. This sets the state for the quantization parameter value, and without any additional information, this is the parameter used for all of the macroblocks within the GOB. A macroblock can change the quantization parameter state by encoding a new value within the macroblock structure. This new value is used for the current and following macroblocks in the GOB until a new parameter is coded for a macroblock. When coding X_{tj}, the quantization parameter state is completely determined by the quantization parameter used for the previous macroblock in the GOB. Therefore, the values for q_{tj-1} and q_{tj} will determine whether or not additional bits must be used to update the quantization parameter state. We show the formula for the total rate to code a frame below.

$$R_t(l_{t1}, q_{t1}, l_{t2}, q_{t2} \ldots, l_{tM}, q_{tM}) = \sum_{j=1}^{M} R_{tj}(l_{tj-1}, q_{tj-1}, l_{tj}, q_{tj}) \tag{2.3}$$

The explicit dependency between neighboring macroblocks can be seen for each term in the sum. In reality, there is no dependency across GOB boundaries, and the rate term for the first macroblock in a GOB is simply $R_{tj}(l_{tj}, q_{tj})$. We define $NGOB$ to be the number of group of blocks within a frame. Each GOB contains N_i macroblocks such that

$$M = \sum_{i=1}^{NGOB} N_i. \tag{2.4}$$

Given a maximum rate constraint R_c for a frame, we want to choose the macroblock parameters $l_{tj}, q_{tj} \, \forall j$ to minimize the distortion in equation 2.2. We use a Lagrange multiplier to form an equivalent unconstrained problem to find a solution to this constrained optimization problem. In this new formulation, we minimize for an appropriate choice of $\lambda_t \geq 0$ the Lagrangian cost function for frame t which is equal to

$$C_t(l_{t1}, q_{t1}, \ldots, l_{tM}, q_{tM}) = R_t(l_{t1}, q_{t1}, \ldots, l_{tM}, q_{tM}) + \lambda_t \cdot D_t(l_{t1}, q_{t1}, \ldots, l_{tM}, q_{tM}) \tag{2.5}$$

If the optimal rate $R_t^*(\lambda_t)$ that results from minimizing (2.5) for a particular value of λ_t is equal to the rate constraint R_c, then the corresponding optimal parameter selection is also a solution to the constrained problem that we want to solve. This can easily be proven by extending the results of [19]. The monotonic relationship between λ_t and $R_t^*(\lambda_t)$ means the appropriate choice of λ_t to satisfy a specific rate constraint can easily be found by an iterative bisection algorithm. As λ_t is swept from 0 to ∞, the solution to minimize (2.5) traces out the convex hull of the operational rate-distortion curve for our bit allocation problem. Because the solution can only trace out a discrete set of points, not any arbitrary value for R_c can be achieved exactly. Therefore, a solution to the above unconstrained problem is equivalent within a convex hull approximation to the solution of the constrained problem we want to solve. We assume a high density of points in the rate-distortion plane. Thus, the convex hull approximation will be sufficiently close to the optimal solution.

The effectiveness of using a Lagrange multiplier to solve this constrained optimization problem is shown graphically in the rate-distortion plane in Fig. 2.7. Any chosen set of macroblock coding parameters $(l_{tj}, q_{tj}\,\forall j)$ corresponds to a single point in this graph. For optimal coding, we want to choose the point with smallest distortion that meets the rate constraint. Since there are a total of two coding modes and 31 possible quantization choices for a macroblock, there are a total of 62 possible coding choices for each macroblock. This means there are $(62)^M$ points in the rate-distortion plane that correspond to all possible coding parameter choices for the frame.

For a particular operating point (R_t', D_t'), the Lagrangian cost is $C_t' = R_t' + \lambda \cdot D_t'$. We can draw a line in the rate-distortion plane with slope equal to $-1/\lambda$ that passes through the point (R_t', D_t'). This shows all values of (R_t, D_t) that will have identical cost C_t'. The Lagrangian cost value for this operating point is where this line intersects the rate axis. This is because when D_t equals zero, $C_t' = R_t$. Therefore, the optimal operating point for the new unconstrained problem for a particular value of λ_t is the one which will minimize the value where this intersection occurs resulting in an optimal cost C_t^*. As the value for λ_t is changed the slope will change resulting in a different optimal operating point. Solving this problem for different values for $\lambda_t \geq 0$ will trace out the points on the lower convex hull of the set of possible operating points. This is shown by the set of connected points in Fig. 2.7. These points are a very good approximation to the operational rate-distortion curve. In fact, any one of these points is the true optimal solution when the maximum rate constraint is equal to their resulting rate. When R_c is not equal to any of these rates we must find which of

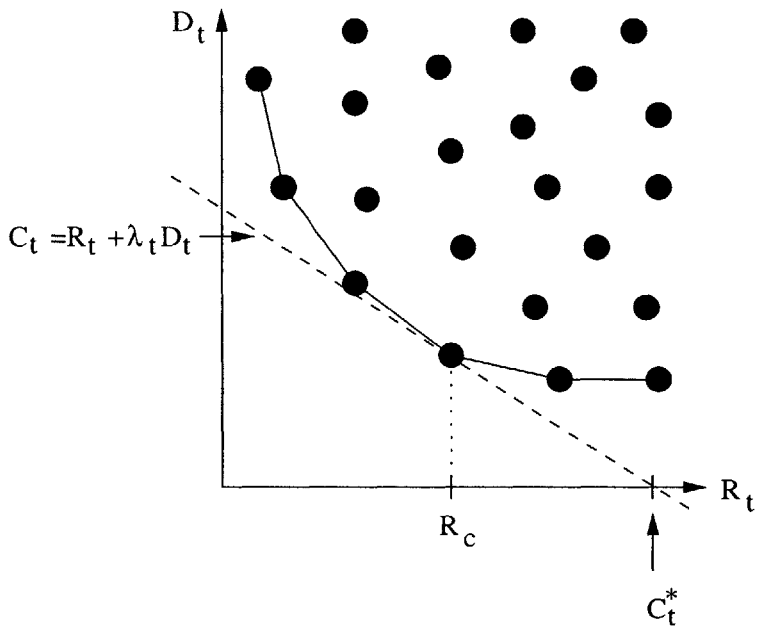


Figure 2.7: Graphical example of the use of a Lagrange multiplier for constrained optimization.

these points results in a rate that is closest to R_c without going over it. In practice, many attempts must be made for different values of λ_t to find the optimal solution that meets the rate constraint.

Since there are an enormous number of operating points, an exhaustive search to find the optimal operating point for a particular value for λ_t can not be performed. If this were possible, the use of the unconstrained Lagrangian problem formulation would be unnecessary, since the true optimal operating point for the original constrained problem could be simply chosen from all the computed rate-distortion pairs for the frame. Instead, a feasible method to find a solution is found when the Lagrangian cost function is further examined. Since the dependency between macroblocks does not extend beyond the *GOB* boundaries, we can independently find a solution over a single *GOB* with N_i macroblocks. The total Lagrangian can be decomposed into the sum of the macroblock cost functions over a *GOB*.

$$C_t(l_{t1}, q_{t1}, l_{t2}, q_{t2} \ldots, l_{tN_i}, q_{tN_i}) = \sum_{j=1}^{N_i} C_{tj}(l_{tj-1}, q_{tj-1}, l_{tj}, q_{tj}) \tag{2.6}$$

The cost function for a macroblock X_{tj} is given in equation 2.7.

$$C_{tj}(l_{tj-1}, q_{tj-1}, l_{tj}, q_{tj}) = R_{tj}(l_{tj-1}, q_{tj-1}, l_{tj}, q_{tj}) + \lambda_t \cdot D_{tj}(l_{tj}, q_{tj}) \tag{2.7}$$

Each macroblock cost function does not have complete dependency over the coding parameters for all N_i macroblocks in the GOB. It is only dependent on the parameters for the current and previous macroblock. If there were no differential encoding between macroblocks or the use of quantization parameter state in a GOB, each macroblock cost would only be a function of its own coding parameters. In this case, the global optimal solution could be found by optimizing over all 62 coding parameter possibilities for each macroblock independently. When there is differential encoding as there is in this case, the limitation of the dependency to such a very small neighborhood composed of the current and previous macroblock provides a perfect opportunity for the use of dynamic programming to efficiently find a solution. Next, we will describe the use of the forward dynamic programming algorithm also known as the Viterbi algorithm to minimize the Lagrangian cost function.

The cost function $C_{t1}(l_{t1}, q_{t1})$ for the first macroblock in a GOB has no dependency on other macroblock coding parameters and can be calculated for all 62 possible coding parameter combinations. However, the cost for the following macroblock depends on what choices are made for the first. To illustrate the Viterbi algorithm used to solve this problem, we form the trellis shown in Fig. 2.8. Each column of dots represents the possible coding parameter combinations for each of the N_i macroblock locations in a GOB. For clarity, we only show three quantization parameter choices ($|Q| = 3$) and only four macroblock locations. As a result only six ($|L| \cdot |Q|$) possible coding parameter combinations are displayed for each of the four macroblock locations shown. For generality, we will continue to use the variables N_i, $|Q|$, and $|L|$ to describe the Viterbi algorithm. We assign the initial costs $C_{t1}(l_{t1}, q_{t1})$ to the $|L| \cdot |Q|$ nodes in the first column of the trellis. We draw a dashed line connecting each node in column $j = 2$ with the $|L| \cdot |Q|$ nodes in column $j = 1$. Each line represents a specific combination of parameter choices for X_{t1} and X_{t2}, and we can compute and assign the cost $C_{t2}(l_{t1}, q_{t1}, l_{t2}, q_{t2})$ to each of the $(|L| \cdot |Q|)^2$ lines. Since no other macroblock depends on the choice for l_{t1} and q_{t1}, we can safely make the optimal selections $l^*_{t1}(l_{t2}, q_{t2})$ and $q^*_{t1}(l_{t2}, q_{t2})$ as a function of each the possible parameter choices for X_{t2}. This is done by applying the following optimization for each (l_{t2}, q_{t2}).

$$C^*_{t2}(l_{t2}, q_{t2}) = \min_{l_{t1}, q_{t1}} [C_{t1}(l_{t1}, q_{t1}) + C_{t2}(l_{t1}, q_{t1}, l_{t2}, q_{t2})] \tag{2.8}$$

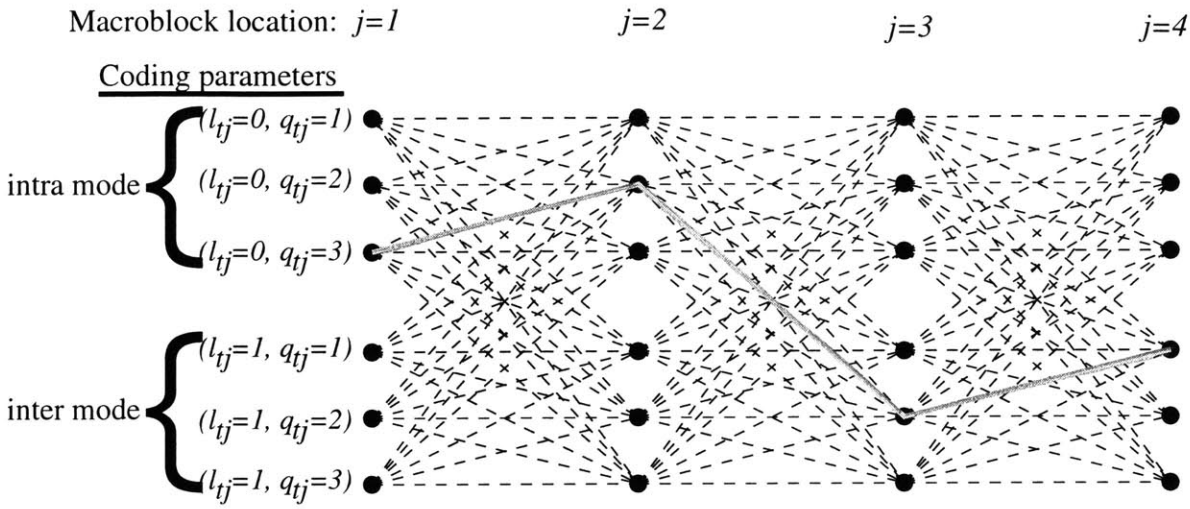


Figure 2.8: Trellis (least cost path shown in gray).

In the trellis, this optimization is shown by eliminating $|L| \cdot |Q| - 1$ dashed lines with suboptimal costs between each node in the column $j = 2$ and the nodes in $j = 1$. This will leave a single line between each possible (l_{t2}, q_{t2}) selection and the corresponding optimal choice for (l_{t1}, q_{t1}). The resulting cost functions $C^*_{t2}(l_{t2}, q_{t2})$ are assigned to each node in column $j = 2$. These cost functions along with the pointers to the optimal parameters $l^*_{t1}(l_{t2}, q_{t2})$ and $q^*_{t1}(l_{t2}, q_{t2})$ are all that need to be stored at this stage in the optimization process. The variables l_{t1} and q_{t1} are effectively eliminated from the optimization problem since they can be determined once the optimal l_{t2} and q_{t2} are found.

The optimization process continues to the next stage between $j = 2$ and $j = 3$. Here, we have an identical situation to the previous stage. The cost functions $C^*_{t2}(l_{t2}, q_{t2})$ at each node in column $j = 2$ do not depend on any other macroblock coding parameters, and the cost functions for X_{t3} depend on the parameters for both X_{t2} and X_{t3}. Like the previous stage, we solve the following optimization problem for each possible coding parameter choices l_{t3} and q_{t3}.

$$C^*_{t3}(l_{t3}, q_{t3}) = \min_{l_{t2}, q_{t2}} \left[C^*_{t2}(l_{t2}, q_{t2}) + C_{t3}(l_{t2}, q_{t2}, l_{t3}, q_{t3}) \right] \tag{2.9}$$

In performing this optimization, we can eliminate $|L| \cdot |Q| - 1$ dashed lines with suboptimal costs between each node in the column $j = 3$ and the nodes in column $j = 2$ in the trellis and

store the resulting cost $C^*_{t3}(l_{t3}, q_{t3})$ at each node. This process of making optimal decisions at each stage in the trellis continues until we reach the last stage for each GOB at N_i. Here, we perform the final optimization shown in (2.10) to select the optimal $l^*_{tN_i}$ and $q^*_{tN_i}$

$$C^*_t = \min_{l_{tN_i}, q_{tN_i}} C^*_{tN_i}(l_{tN_i}, q_{tN_i}) \tag{2.10}$$

Once this is done, the optimal path and parameter selection can be found by following the remaining path in the trellis from $l^*_{tN_i}$ and $q^*_{tN_i}$ through each previous stage until you reach l^*_{t1} and q^*_{t1}. This is shown for example by the gray line in Fig 2.8. To perform the optimization at every stage $|L| \cdot |Q|(|L| \cdot |Q| - 1)$ comparisons must be performed. This is in addition to the $(N_i - 1)(|L| \cdot |Q|)^2 + |L| \cdot |Q|$ cost functions that must be computed for the dashed lines in the trellis. This is an enormous amount of computation. In the following section, we describe a much simpler algorithm which will provide a very near-optimal solution.

2.1.3 Algorithm for Near-optimal Solution

A simplification to the above algorithm can be made with the observation that the most efficient mode selection for a macroblock is mainly determined by the prediction found in the previous frame. While it is true that the use of differential encoding forces dependency between macroblock parameters when determining the rate, the difference in bits when this is taken into consideration is typically quite small. For example, in the $H.261$ coding standard, the difference in bits to code or not to code a new quantization parameter into a macroblock is seven or eight bits depending on whether the current macroblock is to be inter or intra coded. On the other hand, the difference in bits to code a macroblock in intra or inter mode given the prediction region is usually much higher. The coding mode which results in the fewest number of bits when the macroblock is coded with the same distortion is the most efficient mode selection. This selection would be the same regardless of the distortion level chosen. This is the method proposed for mode selection in [20]. With this in mind, mode selection can be made independently without much loss in optimality (if any) using this algorithm. We choose the MSE between each macroblock X_{tj} and decoded macroblock Y_{tj} at the encoder as the distortion to keep constant when comparing the rates to code a macroblock in different modes.

Given the motion vectors and the chosen efficient mode selections ($l^*_{tj}\ \forall\ j$), the rate

to code each frame t is further determined by the quantization parameters for all the macroblocks in the frame. We express this as

$$R_t = f(l_{tj}^*, q_{tj}) \quad \forall\, j \tag{2.11}$$

We could find the optimal quantization using the Viterbi algorithm described in the previous section by first reducing the parameter choices at each stage in the trellis to only include the already determined efficient mode selection. This is still a large amount of computation for $|Q| = 31$. Without loss in generality, we set the values of q_{tj} to be chosen such that the MSE between each macroblock X_{tj} and decoded macroblock Y_{tj} at the encoder is equal to some constant for the frame scaled by a weighting factor.

$$\|X_{tj} - Y_{tj}\|^2 = \alpha_{tj} \cdot K_t \tag{2.12}$$

The allowance for an arbitrary set of scaling factors α_{tj} keeps this problem formulation general. One specific application of this new formulation would be to code the frame with consistent quality. The weighting factor α_{tj} can be used to adjust for the perceptual masking effects of macroblocks with different spatial characteristics. For example, macroblocks with a lot of detail can tolerate more quantization noise due to spatial masking to achieve acceptable perceptual quality, whereas blocks with less detail should have a lower MSE to achieve the same perceptual quality. We do not discuss in this thesis how to choose the weighting factors, but we do present an algorithm which allows for differently weighted values of K_t to be used for each macroblock. With this formulation, the frame quality parameter selection problem now becomes that of choosing the value K_t for each frame, and the rate function to code frame t is simplified to be

$$R_t = f(l_{tj}^*, K_t) \quad \forall j \tag{2.13}$$

Since there is a monotonic relationship between R_t and K_t, finding the optimal K_t is simple. The lowest distortion value K_t that will meet the bit rate constraint is chosen, and can be found using a bisection algorithm.

In this section, we have presented a good method to select efficient coding parameters for block-based coders. Since the potential for loss is not considered this algorithm is most useful when transmitting data over reliable channels. By itself, it is not appropriate for transmission over the Internet which can be very unreliable. We next discuss the Internet

and describe how video and data is transmitted over it. We begin in the next section with a discussion on packet-switched networks.

2.2 Packet-Switched Networks

The classical approach to transmitting data in a telecommunication network is to first establish a connection between a transmitter and receiver that will exist throughout the session. This is known as circuit switching where a fixed bandwidth is allocated for data transfer for each connection. Time division multiplexing (TDM) is used in this case to share a link among several communicating sources connected to the network. Because each source is allocated a periodic time slot at which it can transmit data, this form of multiplexing is also known as Synchronous Transfer Mode (STM). The bandwidth allocated to a session must be large enough to handle the peak rate of transmission for a source. By allowing for this, circuit switching will always guarantee that sufficient bandwidth is available, but may suffer from inefficiency when a source does not always transmit at its peak rate. Circuit switching can be extended to transmit data from bursty sources more efficiently, but the signaling required to quickly set up and take down circuits on demand can be very cumbersome.

In contrast to circuit switching, packet-switched networks rely on asynchronous time division multiplexing ($ATDM$), and have the potential for more efficient use of bandwidth. Packet-switched networks first divide the data stream up into individual units called packets and encapsulate each unit with a header before transmission. The header contains all of the necessary information required to transmit the data to the correct receiver on the network. Routing tables at each network node determine the packet path. In contrast to STM, a transmitter is not restricted to a particular time slot in which it can transmit data. As a result, without the proper precautions, the packet arrival rate from all incoming links at a computer in the network along a transmission path may exceed the rate at which they can be forwarded to the next network node. In addition, there may be packets arriving nearly simultaneously that are competing for the same output link. To alleviate this problem, a buffer is located at each node in the network to store packets waiting to be serviced. There are many different strategies for servicing arriving packets at a network node. The simplest strategy is to service packets on a first-come-first-serve basis at each network node. There could also be more complicated scheduling algorithms implemented for better traffic

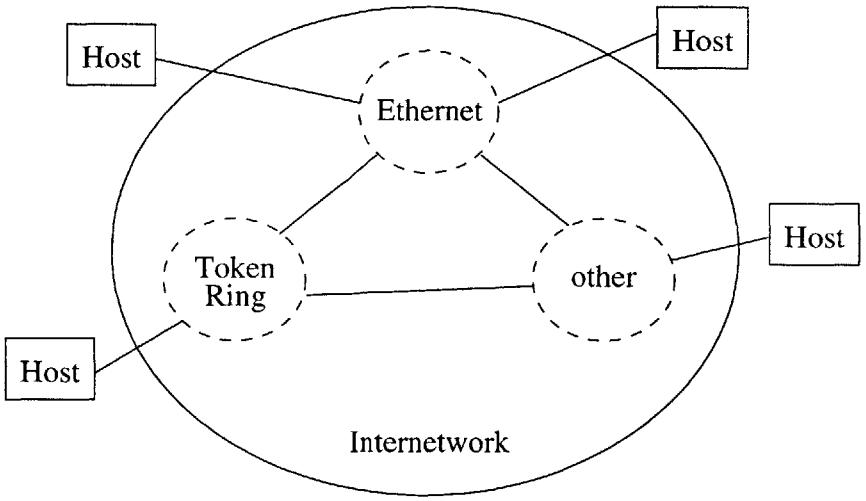


Figure 2.9: Block diagram of Internet.

management. Examples of this will be discussed later. In the following sections, we describe the use of packet-switched networks in transmitting data. We highlight the problems in transmitting real-time data as well as discuss attempts to provide robustness to loss by the network protocols. We begin by describing the Internet and its family of protocols TCP/IP.

2.3 Internet TCP/IP Protocols

The Internet is an interconnection of packet-switched local area networks (LAN) that connects computers worldwide. A block diagram of the Internet is shown in Fig. 2.9. It is composed of potentially many different types of packet-switched $LANs$. Ethernet and Token Ring networks are two examples. The Internet defines a family of protocols called TCP/IP which stands for Transmission Control Protocol and Internet Protocol respectively. These protocols allow for the communication of data between separate networks. A computer called a gateway is responsible for forwarding Internet Packets from one network to another.[21].

TCP/IP is a layered protocol design. A diagram of the different layers is shown in Fig. 2.10 The whole process begins at the application running on one of the hosts which produces a message stream to be transmitted. In our example, this would be the video coder described in section 2.1. The bit-stream is then passed to the transport protocol layer. The

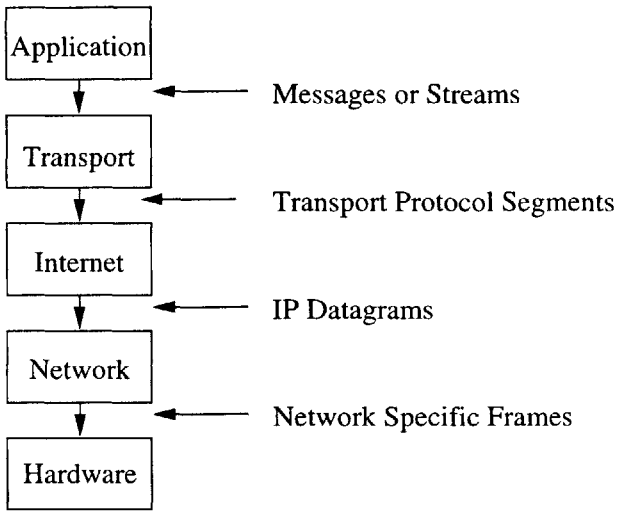


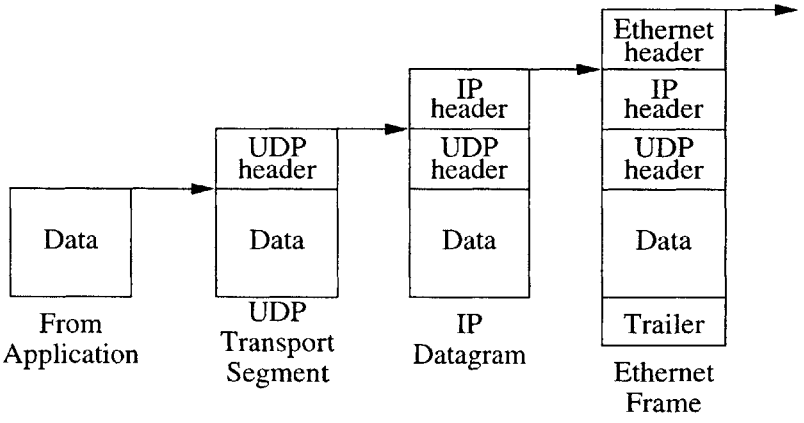
Figure 2.10: Protocol layers.

transport protocol is responsible for dividing the bit-stream into packets and multiplexing data to and from the different applications running on a host computer. The term port is the abstraction used to designate an application running on a host that is transmitting or receiving data. There are mainly two different transport protocols used in the current Internet Standard. The first is *TCP* which is included in the protocol name, and the second is the User Datagram Protocol (*UDP*). The choice of protocol depends on the requirements for the overall message transfer. At this layer, a header is created and combined with each packet of data to form a transport segment. This segment is then passed to the *IP* layer. We defer a description of the transport protocols to the next section and describe the *IP* layer below.

The *IP* protocol is responsible for transporting the Internet packets within the whole network. Every host and gateway connected to the Internet contains an internet module which provides a common framework for routing packets throughout the network. At the *IP* layer in the originating host, a header is attached to the transport segment to form an Internet Packet which is called a datagram. Every host connected to the Internet has a unique *IP* address. The address of both the source and destination host are contained in the *IP* header and are used at each network node to determine the next link the datagram will travel.

The *IP* protocol is a connectionless protocol that transmits datagrams independently. The *IP* header contains a header checksum for error detection. This will prevent misrouting of datagrams from bit errors in the addresses that may occur during transmission. When an error is detected a packet is dropped at a network node without notification by the *IP* to the communicating hosts. A packet is also dropped without notification by the *IP* when arriving at a node with a full buffer. As a result, the *IP* is a very unreliable service and depends on either the end systems or higher layer protocols such as *TCP* to detect and appropriately handle errors in transmission. In the case of normal delay-insensitive data transmission, dropped datagrams are detected by *TCP* and retransmitted. For the case of real-time data, retransmission is not a possibility. In this case, dropped packets go undetected by the network protocols, and the end systems are left responsible for handling the loss appropriately. For example, concealment methods can be used at the receiver when real-time video sequences suffer from loss.

Finally, a header and sometimes trailer is attached to the datagram creating a frame that can be transmitted over the underlying physical network according to the *LAN* protocol. At the receiver, the reverse steps are taken to deliver the data to the appropriate application. As an example, we use Ethernet for the LAN and show a block diagram of the data flow from the application to the network below.



2.3.1 Transport Protocols

Most traditional data transfer over the Internet uses *TCP* for its transport protocol to make up for the unreliable service provided by *IP*. Unlike *IP*, *TCP* is a connection oriented protocol. Before beginning transmission of a data stream, *TCP* executes a handshaking protocol to establish a connection between the communicating ports at the sending and receiving hosts. The transmitting port first sends a packet to the destination host requesting a connection setup. The receiver responds with an acknowledgment packet which establishes a connection between the two hosts. This, of course, is a virtual connection and not a real one since it is not supported by the underlying *IP* network. Upon connection, the transmitter begins to send the data to the *IP* layer. *TCP* uses Automatic Repeat reQuest (*ARQ*) for reliable communication. When each datagram arrives at the receiving host, an acknowledgment packet is sent back to the transmitter according to *TCP*. This is known as positive acknowledgment. The transmitting host begins a timer when a datagram is sent. If an acknowledgment packet has not been received when the timer expires, the *TCP* assumes the original datagram has been dropped by the network and the data is retransmitted. This assumption may result in duplicate arrivals at the receiving host if the packet really was not lost. Here, *TCP* will eliminate the redundant packets and ensure that packets are delivered in the proper order to the receiving application. With *TCP*, all of the data is eventually delivered reliably to the destination host.

To help avoid network congestion and the resulting datagram loss, *TCP* implements flow control. When a connection is first established, *TCP* transmits packets to the *IP* layer at a slow rate. As acknowledgment packets begin to arrive, the protocol assumes there is no congestion and increases the rate at which transport segments are sent to the *IP* layer. This rate is incremented as acknowledgment packets continue to arrive until some maximum transmission rate is reached. This increase in rate leads to more efficient bandwidth use since available bandwidth is not wasted. On congestion, packet loss occurs and acknowledgments fail to arrive. Here, *TCP* resets to the lowest transmission rate for the transport segments and begins the process of incremental rate increases again.

The retransmission of lost packets and the flow control implemented in *TCP* are not appropriate for data with delay constraints. For this type of service *UDP* is used as the transport protocol. This service is connectionless like *IP* and offers no error control or flow control. It does provide a checksum in the header to detect bit errors in the data portion

of the packet that may have occurred during transmission. Any further reliability must be implemented in the application. The Real-time Transport Protocol (RTP) was developed by the Internet Engineering Task Force ($IETF$) to allow for the transmission of real-time data.[22] Although it offers no guarantees or QOS, it provides the necessary hooks that make the transmission of real-time data possible at the very least. We describe the details of the protocol in the next section.

2.3.2 Real-Time Transport Protocol

RTP was designed to transport data with real-time properties. It can not guarantee timely delivery of data since the underlying network does not. Inside the application, an RTP header is attached to the data as it is delivered to the transport layer which is typically UDP. Even though RTP is run in the application, both RTP and UDP implement some of the transport protocol functionality. RTP uses UDP for its multiplexing capabilities among the different ports at a host while adding some flow and congestion control in the application. This is done with the help of the RTP control protocol $RTCP$ which is used to monitor data delivery to the receiver.

The Internet delivers packets with variable delay. This is a result of the variable queueing delays experienced at nodes along the transmission path. This variable delay is known as delay jitter and can be compensated for by setting a maximum delay limit and queueing packets at the receiver until their end to end delay reaches this limit. [23] When the maximum delay in the network is below this set limit, delay jitter has little consequence. There is a tradeoff when setting the delay limit for real-time video. If it is set too low, the video sequence will suffer extensive loss since packets experiencing network delay above this limit will arrive too late to be played at the receiver. If it is set too high, the transmission can no longer be considered real-time, and it will be impossible to carry out interactive conversation. In addition to possible loss from delay jitter, packets may arrive out of order. To remedy this, the RTP header contains a sequence number designating the packet order in the stream as well as a time stamp indicating the sampling time of the first byte in the data packet. At the receiver, this will allow for the detection of lost packets from missing sequence numbers as well as the ability to re-order out-of-order arrivals.

RTP reserves seven bits in the header to specify the payload type. This indicates the

format of the payload data and its interpretation by the application. A mapping of payload numbers to standard audio/video encoders is found in the Internet document [24]. It includes payload types for the example block-based coders listed in section 2.1. The inclusion of payload type will allow for an additional header to be placed between the end of the *RTP* header and the beginning of the bit-stream in the data portion of the packet. The format of this header is specific to the standard audio/video coder indicated by the payload type. This header is typically used to repeat the state information of the decoder at the beginning of the coded bit stream portion that is transmitted in that packet. In addition, a macroblock is not split across packet boundaries. All of this is used to increase the robustness to packet loss, since the bit stream contained in any arriving packet is independently decodable. Examples of some of the state information for the $ITU-T$ video coder recommendations include the following.

- the *GOB* number in effect
- the quantization value in effect
- the previous coded macroblock address
- the previous macroblock motion vector

The information for the first coded macroblock in a packet is coded differentially from the information from the previous coded macroblock transmitted in the previous packet.

The header also contains a 32-bit synchronization source (*SSRC*) identifier which is chosen at random for each *RTP* session in progress. For example, the audio and video streams of a video conference are transmitted over separate *RTP* sessions. This will allow the receiver to know which stream suffered packet loss when a missing sequence number is detected. The reader is referred to [22] and the associated Internet documents for more details on the implementation of *RTP*.

The main purpose of *RTCP* is to provide feedback about the current *QOS* achieved in the network. *RTCP* information packets are periodically sent to all participants in a video conference. It contains statistics which include the packet loss observed at the receivers. This information can be used for flow control implemented in the application. For example, when a high packet loss rate is reported, an adaptive video encoder can reduce the coding rate by applying even more lossy compression. For example, the quantization step-sizes may

be increased. The effect on the network transmission is similar to what happens when TCP lowers its transmission rate on congestion. However, there is a distinct difference between the overall effect on the communication. The TCP protocol will cause the port at the receiver to wait for the correct data to be retransmitted. In addition, the transmitting port will slow down its transmission rate losslessly. Thus, it will eliminate any loss in data fidelity at the expense of an increase in the overall delay. On the other hand, the strategy implemented by the adaptive encoder will keep the delay at a minimum at the expense of an increase in distortion from both loss and higher compression.

When a maximum bit rate constraint is imposed, the feedback can be used to better allocate bits between source and channel coding when forward error correction (FEC) is used at the source. FEC transmits redundant information to give the receiver the potential to recover some errors. The decoder state information replicated in the payload type specific header described above is an example of FEC used for robustness to loss. In addition, the periodic feedback will allow the sender to track long-term network conditions and perhaps allow it to model network service.

2.4 Summary

In this chapter, we presented algorithms to select the (near-)optimal coding parameters for block-based coders. By using these algorithms, the video can be coded at the highest quality when no channel errors occur while meeting the bit rate constraint. The parameter selection as described in this chapter is not appropriate for coding over lossy channels. Since it is likely to code many macroblocks using prediction, the distortion will increase dramatically when channel error occurs. Robust encoding methods should be used to reduce the additional distortion from channel errors. In addition, methods can be implemented at the decoder to conceal these errors.

The current Internet is not capable of transmitting real-time data well because of its inability to offer any QOS guarantees. In the next chapter, we will look at methods that can be implemented in networks to increase robustness for real-time video transmission. This includes both the new Integrated Service Model [25] proposed to supplement the current IP service and the development of the Asynchronous Transfer Mode (ATM) network. The purposes of these methods are to accommodate many different service classes required in

network transmission. This includes real-time data transfer with some *QOS* guarantee. We will also describe common concealment methods and current video coding parameter selection methods to result in robust coding.

Chapter 3

Background on Robust Methods for Unreliable Video Transmission

---

In this chapter, we describe current methods proposed to add robustness to video coding over unreliable channels. We continue with the example of real-time video coding over packet-switched networks discussed in the previous chapter. In the next section, we begin by discussing attempts to increase reliability in network transmission. This includes possible improvements to the current Internet protocol and a discussion of Asynchronous Transfer Mode (ATM) networks. Here, we will present current and proposed network protocol approaches for error control and prevention and some of the problems in implementing these standards to guarantee quality of service (QOS). Next, we will discuss methods that can be used at the receiver to conceal errors from lost data. In the following section, we discuss current ad hoc methods to select the coding parameters for block-based video encoders to add robustness to the coded video realizing that errors may occur. We will also discuss the drawbacks to these methods. Finally, we end this chapter by proposing a new strategy that is presented in the next chapter and overcomes the drawbacks of these traditional approaches.

3.1 Robust Methods in the Network

The Internet currently operates with a "best effort" service class only. It provides equal access to all of its resources and promises not to drop packets unnecessarily. This results in an unreliable network which can offer no guarantees, and it is up to the end systems to provide an adequate quality of service. This can be done by retransmission, rate control, FEC at the transmitter, error concealment at the receiver, or a combination of the above. Retransmission is not an option for real-time data transport. There is no typical value for loss rate for the Internet. However, experiments have been performed across different parts

of the Internet which have reported observed loss rates ranging from two to twenty percent [26, 27, 28].

The end systems are very dependent on the underlying network and can not offer guarantees beyond what the network can provide. Realizing this has led to a proliferation in the literature of new and improved service classes in networks as well as improved asynchronous multiplexing methods that allow the underlying networks and subnets to provide some isolation when needed and resource sharing where appropriate [29, 30, 31, 32, 33]. This includes resource reservation on the Internet and the Asynchronous Transfer Mode (ATM). These networks may result in systems that are more efficient while offering some QOS. However, there still exists many problems in developing admission control policies and an appropriate description for video traffic that will sufficiently characterize it. This description must be able to be verified by the network and compatible with descriptors used for other types of data transmitted on the network. In the following subsections, we discuss the proposed Integrated Service Model and ATM networks.

3.1.1 Integrated Services on the Internet

The aim of integrated service is to allow for the transmission of different types of data requiring different types of network service. At one extreme, there is bulk data transfer which has almost no delay constraints and is adequately transmitted with the Internet's current and only "best effort" service model. Electronic mail is an example of this type of data transfer. At the other extreme, you have real-time data which has a maximum delay constraint. Enough of the network resources must be dedicated to this real-time data flow to ensure that this delay constraint is met (at least statistically). Network administrators have identified an additional service to control the amount of sharing done over a link. Here, the data streams that will be transferred over a particular link are divided into distinct classes. These classes can be arbitrarily defined. Different user groups, protocol families, or stream types are a few examples of classes. When the link becomes overloaded, a certain percentage of the total link capacity is assigned to each class. When the link is not overloaded there is no need to restrict any class. This service is known as controlled-link sharing. The Internet Engineering Task Force ($IETF$) officially defines the term "integrated services model" to allow for the transfer of these three classes: best effort, real-time, and controlled link sharing.[25]

The aim is to allow for these integrated services by extending the current IP protocol without re-designing a completely new network. In this section, we will first present an overview of the integrated service model for the next generation of the Internet protocol. We will focus on the part of this extension that will allow for the use of real-time data transfer over the Internet. We will next discuss implementation of the this service model on the Internet. Finally, we will end this section by outlining some of the challenges and tradeoffs in implementing the integrated service model. It is not clear in what time frame this model will be fully implemented in the network, but there are currently small experimental sections of the Internet that are capable of implementing this model for testing and development purposes.

3.1.1.1 Integrated Service Model

In describing the integrated service model, we denote a flow to be a stream of data from an application to be transmitted on the network. To guarantee QOS for a flow over a shared resource, the network must be able to do two things. The first is to be able to restrict the applications that have access to the network. In doing this, it can prevent a new flow from being transmitted over the network when it will cause congestion and degrade the service of flows currently admitted on the network. The second is to reserve resources throughout the network for each admitted flow to ensure that the agreed upon QOS will be met. In addition, an application must be able to indicate the desired QOS requested from the network as well as provide an adequate traffic description of the flow it will transmit through the network. With this information along with the traffic descriptions for currently transmitted flows, the network can decide if the requested QOS can be supported. If it can, the application is allowed to transmit its data on the network. This function is known as admission control.

A service model is the set of services or commitments that a network can offer an application. This is specifically defined by the QOS that can be requested. For example, the only QOS needed for the transmission of real-time data over the network is some kind of bound on the delay. We define the commitments required for two extreme types of data flows below. The first flow is real-time data and the second type of flow is known as elastic data.

Real-time data from applications such as live audio and video have a playback point which indicates the time at which the data will be used at the receiver. This point is set by

the delay imposed at the receiver before the initial arriving data in a flow is played out and the audio/video capture rate. The playback point for the data in a packet will determine the maximum delay that it can tolerate, since any data arriving after its playback point is effectively lost. For interactive communication the receiver has little freedom at where the playback point can be set. The maximum delay tolerable for video conferencing is said to be around $100ms$ [34].

The maximum delay bound designated in the service contract with the network is used at the receiver to set the playback point for a data flow. Excess delay experienced by individual packets beyond their playback point will result in an incomplete signal. The integrated service model divides real-time applications into two service categories depending on their ability to handle this form of degradation. Some applications can not tolerate any loss or degradation. Service contracts for these applications must provide a perfect maximum delay bound. This type of QOS is called guaranteed service. While other applications are willing and able to tolerate some loss. In this case, a statistical bound on delay is more appropriate. This QOS is known as predictive service.

Of course, applications requiring predictive service can be better satisfied by guaranteed service. However, the use of guaranteed service will often lead to the under utilization of the network. This is because to offer guaranteed service, admission control can only use the worst-case bounds on the data flows being transmitted. For example, bandwidth must always be reserved to support the peak rate of a data flow. In this case, the efficiency gain from statistical multiplexing in a network is lost. It is therefore more costly for this service to be used, since fewer users will be allowed access to the network. In addition, an application may choose to use predictive service as opposed to waiting until its request for guaranteed service can finally be supported by the network.

In contrast to real-time applications, elastic applications will wait for data to arrive before continuing. Examples of these applications are the traditional data transfer done over the Internet which are less sensitive to delay. This includes the file transfer protocol FTP and electronic mail for example. The "best effort" service offered over the current Internet protocol is appropriate for these applications.

An additional service offered by the integrated service model allows for prioritization of packets which will assist the router in packet dropping. Up to this point we have assumed that all packets are equal. This is not usually true for compressed data. The additional

service provides two packet labels. The first allows a packet to be labeled as a preemptable packet. This will assist the router in choosing a packet to discard when it might fail to meet the *QOS* agreement. Packets with this label are included in the traffic description used for admission control. Therefore, it is expected that these packets will rarely be lost. The second allows for packets to be labeled as expendable. The description of the traffic for these packets is not used in admission control. As a result, expendable packets are treated with "best effort" service.

3.1.1.2 Implementation Framework

We now describe the reference framework which will give the network the ability to implement the integrated service model described above. Some of the elements involved were mentioned in the previous subsection. All of the elements in the reference implementation model are listed below.

- Admission Control
- Reservation Setup
- Classifier
- Packet Scheduler

Implementing the integrated service model requires traffic control. This is accomplished through admission control and through functions performed by the network routers and host. A router in the network has a small number of options to choose from when a packet arrives. It can select the route to transmit the packet and forward it, it can drop it, re-order it, or hold it back. Beginning with admission control, the elements listed above are used to help the router select the appropriate function to execute. We will describe each in turn below.

The challenge of admission control is to achieve high link utilization with little or no service degradations. It is difficult to find the best solution to this problem. There are two strategies that are used for admission control. The first one was discussed above and only considers the worst case bounds given in traffic descriptions. This will lead to poor network utilization. The second strategy observes the actual usage of data flows already admitted to the network and uses those statistics to determine whether a new flow should be admitted.

This will possibly lead to better network utilization, but at a higher risk of experiencing multiplexing failure and resulting packet loss.

An additional problem to admission control is in deriving appropriate traffic descriptors which will help the controller achieve the desired efficiency. The traffic descriptor must be accurate enough for the admission control to be able to predict the effects of statistical multiplexing but still be simple enough to be implemented. This will be especially more difficult to do at admission time for data captured and transmitted in real-time. A common traffic descriptor used is a leaky bucket. This descriptor specifies both the average bit rate and the maximum burst size that will be transmitted by the source.

Once a data flow is admitted to the network, a mechanism must be in place to reserve resources throughout the network to ensure the promised QOS. This reservation is accomplished by the Resource ReSerVation Protocol ($RSVP$) developed for the Internet [35]. The flowspec and the filterspec for the data flow are transmitted to routers along the travel path. The flowspec specifies the amount of resources that should be reserved at each router, while the filterspec indicates which subset of the packet streams have reserved these resources. For multicast purposes where there are several receivers downstream for any single data flow, the $RSVP$ protocol will allow for the appropriate combination of reservation requests. For example, the new flowspec given to a router shared by two or more receiving hosts becomes the smallest amount of resources needed to satisfy all of the reservation requests. This will prevent wasteful duplication of resource reservations for a single dataflow. The new filterspec becomes the union of the individual flowspecs.

Packets must be classified at each router to determine what kind of treatment they will receive. The classifier will allow the router to know what resources have been reserved by applying the different filterspecs. As mentioned before, the allowable set of classes are unlimited. This causes a potential complexity issue for the classifier at the router. It is easy for a classifier to use information readily available in the IP header to classify a packet at the router. This is useful for filterspecs which specify the address of transmitting upstream hosts for example. To avoid the difficulty of having to look deeper into the packet to parse the headers of higher layer protocols, it has been proposed that a flow-id be added to the IP header to facilitate the classification process.

The packet scheduler determines what order packets saved in a queue at the router will be transmitted to meet the QOS requirements. One possibility is to prioritize the

packets in the Queue and always transmit the packets with highest priority first. However, this might prevent some low priority packets from ever being transmitted from a queue. On the other hand, weighted fair queuing (WFQ) can be used to allocate the bandwidth into different shares. An additional function that we associate with the scheduler is the controlled dropping of packets. When a packet is dropped, it will decrease the delay of packets in the same data flow queued behind it. Therefore, the loss of one packet may dramatically improve the service of many packets. In addition, some applications will reduce the transmission rate when dropped packets are observed. Therefore, the choice of dropped packets on congestion will provide implicit feedback to an application which it will use for rate adaptation.

3.1.1.3 Implementation Drawbacks

As discussed in the previous subsection, admission control is difficult to implement. The major problem is how to best balance the tradeoff between bandwidth efficiency and service guarantees. One extreme which offers guarantees is very conservative and costly since fewer users will be allowed access to the network. Predictive services are desired for more efficiency and lower cost but not with unacceptable QOS. Unfortunately, the problem of providing appropriate traffic descriptors to avoid multiplexing failures has not been adequately addressed. At best, a predictive services will not provide a perfectly reliable delay bound, and users should expect to get some excess delays and dropped packets.

The argument has been made that in a network that can provide QOS guarantees, the unused bandwidth can be filled with data that does not need any guarantee. An example of this would be traditional datagram traffic. However, the network may still suffer from under or over reservation in this case. At times, the real-time traffic might fill its reserved bandwidth, and datagram traffic can not get through, frustrating users. There may also be times when datagram traffic is not large enough to fill the unused bandwidth.

Taking all of this into consideration, a real-time application should be made robust to loss. This increase in robustness at the application will mitigate the harsh requirements on the network to guarantee a QOS while maintaining high utilization. In addition to expected loss, a real-time application should also expect some form of rate control to be imposed. This is a direct result of the admission control process that will be used. The rate control used in current real-time Internet applications transported with RTP/UDP is generally self imposed since abuse of network resources could halt the network operation.

3.1.2 Asynchronous Transfer Mode Network

The Asynchronous Transfer Mode (ATM) network is designed to achieve very fast packet switching. This increase in speed over traditional packet-switched networks is a direct result of the minimal functionality placed in the switches inside the network. Many of the major functions to enable effective and efficient transport are moved to the edge of the network. Another goal in the design of ATM is to provide a network which is capable of transferring **any** service regardless of its characteristics. This includes burstiness, bit rate, and required QOS. This network is service independent and the switching and multiplexing will remain the same. Therefore, ATM provides a general network that will have long-term appeal.[36]

The goal of universal service application is similar to that of the $IETF$ in its development of the integrated service model. More specifically, since real-time service delivery is a requirement for both, some of the same problems discussed in the previous section must also be addressed in the design of ATM. Not surprisingly, both share many common features. However, ATM and the integrated service model were designed by different organizations with different sets of constraints in mind. As a result, there are notable differences between them which we will focus on in this section.

The two main factors that lead to an increase of speed in ATM networks over previous packet switched networks are the use of small, fixed-sized packets that also have limited header functionality. A packet in an ATM network is called a cell which has a fixed size of 53 bytes. The header portion of the cell is only five bytes, while the payload makes up the remaining 48 bytes. This is in far contrast with the Internet which allows for larger packets of variable size of up to 65,535 bytes. For the Internet, the packet size must be included in the header. Variable packet sizes demand more complicated queue management at the router since the memory can not be divided into fixed-sized memory locations beforehand. As a result, a find-the-best-fit or first-fit algorithm has to be performed to place an arriving packet into the memory. In addition, since the ATM cell size is small, the buffering requirements at a switch are also small. Small buffers will lead to smaller delay in the network as well as smaller timing jitter for arriving cells in a data stream.

ATM provides a connection-oriented transmission as opposed to the connectionless one offered by the IP. In ATM, a virtual circuit is setup over the network between the sender and receiver before transmission begins. Similar to the use of admission control and $RSVP$ in the integrated service model, this will first ensure the necessary network resources

are available and will reserve them at the switches when they are available. This will also determine the route in the network for all packets in a data stream that will travel over this virtual circuit. In this case, only a virtual identifier number is required in the header to associate a packet to a particular virtual circuit and route. This is in contrast to *IP* which requires that both the source and destination address be included in the header to determine the packet route. In addition to a virtual circuit which is typically associated with one particular data stream, *ATM* provides for the setup of a virtual path. A virtual path is a semi-permanent connection between two hosts which can be composed of multiple virtual circuits at any time. These circuits can be set up and taken down throughout a session. For example, in multimedia communication, video and audio are carried over separate virtual circuits which are associated to a single virtual path.

The use of a virtual circuit allows easy multiplexing of different data flows at a switch, and the identification of the virtual circuit number in the packet header is the main function performed at a switch. In addition, since the service is connection-oriented, out-of order arrival is eliminated. There is no use of error or flow control applied within the network during transmission. Everything is done through preventative measures beforehand at setup time. Because of the simplified router functionality, network speeds of 150 *Mbps* up to the order of *Gbps* can be achieved with *ATM* networks. This is in comparison to the 10 *Mbps* speed of an Ethernet *LAN*, or even the 100 *Mbps* speed of a Fast Ethernet *LAN*.

QOS must be offered in addition to increased network speeds. This is especially true for real-time service. The *ATM* services offered are similar to but not identical to the integrated service model's options of best-effort service, guaranteed and predictive delay, and controlled link sharing. Instead, *ATM* offers the following five services:

- Unspecified Bit Rate (*UBR*)
- Variable Bit Rate non-real-time (*VBR−nrt*)
- Variable Bit Rate real-time (*VBR−rt*)
- Constant Bit Rate (*CBR*)
- Available Bit Rate (*ABR*)

UBR is the simplest service which offers no guarantees and is very similar to the "best effort" service currently provided by the Internet. *VBR−nrt* offers loss guarantees but does

not provide any constraint on delay. As a result, these two services are not options for real-time video transport. The only remaining service class options for real-time video are CBR, $VBR\text{–}rt$, and ABR. CBR service offers a fixed bandwidth pipe and does not take advantage of statistical multiplexing. Like classical circuit switching, use of this service will also result in under-utilization in the network for bursty traffic. The only traffic descriptor required for admission control with this service is the peak cell rate (PCR). CBR can offer deterministic QOS guarantees on the cell loss rate (CLR), maximum cell transfer delay ($maxCTD$), and cell delay variation (CDV)[37]. $VBR-rt$ also offers a delay constraint in addition to cell loss guarantees. However, since $VBR-rt$ allows statistical multiplexing, these guarantees are also statistical. The traffic descriptors used in connection setup are the peak cell rate (PCR) and the sustainable cell rate (SCR).

The loss in efficiency of using the CBR service is very high. Since the cost of $VBR\text{–}rt$ can also be high, some researchers have suggested the use of the ABR service with adaptive video coders to achieve more efficiency. The ABR service provides minimum cell rate (MCR) guarantees and is designed to provide low cell rate loss for well behaving sources [38]. The MCR guarantee can be used to provide an acceptable QOS for real-time video applications. The ABR service offers closed loop feedback which can be used to adapt the output rate of the video coder to current network conditions. The source periodically transmits resource management (RM) cells which are returned by the destination. Switches along the path report in the RM cell the maximum bandwidth they can currently tolerate. In addition the ABR service has the advantage of renegotiating its service contract during transmission where CBR and $VBR-rt$ can not. This can be used to increase the allowable cell rate (ACR) as more bandwidth becomes available.

Because of its general applicability, fast switching speeds, and upgradability, ATM has been chosen by the $ITU-T$ as a solution for the Broadband Integrated Service Digital Network ($BISDN$). $BISDN$ will provide a single service independent telecommunication network in the future. In addition, realizing that multimedia traffic will also be transported locally, computer users have looked at building $ATM\ LAN$s. This work is focused on by the ATM Forum. In light of this, the $ATM\ LAN$s can be interconnected as part of the Internet. Unfortunately, while the hosts connected within the LAN can exploit the advantages of all the ATM service classes, only the UBR service can be used to transport data from $non-ATM$ networks across the $ATM\ LAN$. Work is currently underway to allow the new integrated services of the Internet to work efficiently with the capabilities of ATM.

Even with the advantages of ATM networks, many of the problems discussed for integrated services over the Internet exist here as well. Multimedia traffic needs some guarantees on delay and cell loss for adequate quality. Deterministic guarantees can only be obtained at the expense of poor network utilization. To achieve a desired level of efficiency, there are two things that a system designed to transmit real-time video over a network should be prepared for. The first is that some form of rate control will be imposed on data entering the network. This is from the admission control and policing mechanisms required to provide QOS. The second is that loss should be expected and must be concealed at the receiver. In the next section, we will look at some methods which can be used to conceal lost macroblocks from packet loss.

3.2 Review of Concealment Methods

There are many concealment algorithms which use temporal or spatial interpolation to conceal lost macroblocks in decoded video. Before describing some of these concealment methods, we first distinguish between two different loss scenarios that can occur. The first is *coded-block loss* where either an intra-coded macroblock or residual macroblock is lost. For this case, the whole 16 x 16 square macroblock region must be concealed. An example of this case is shown in Fig. 3.1 with the lost and concealed macroblock at position $j3$ in frame $t-1$. The second is *prediction loss only* where the residual arrives with motion vector but any part of the prediction region is lost. This case is also shown in Fig. 3.1 where part of the prediction region for an inter-coded macroblock in frame t contains pixels from the concealed macroblock in frame $(t-1)$. It is very unlikely that the concealment for this macroblock will be perfect. Therefore, there will be a difference between the concealed part of the prediction region and the decoded copy used at the encoder for prediction. In this case, the inter-coded macroblock pixels in frame t which use the lost pixels in frame $t-1$ for prediction must be concealed. The remaining pixels in the macroblock can be decoded correctly.

We review some simple temporal and spatial interpolation methods below. All of these methods rely on information from a small neighborhood surrounding the lost macroblock for concealment. This is an important feature which is exploited by the algorithms presented in chapter 4 for optimal robust video coding. We end this section by showing experimental results of the different concealment methods and discussing the benefits and drawbacks of

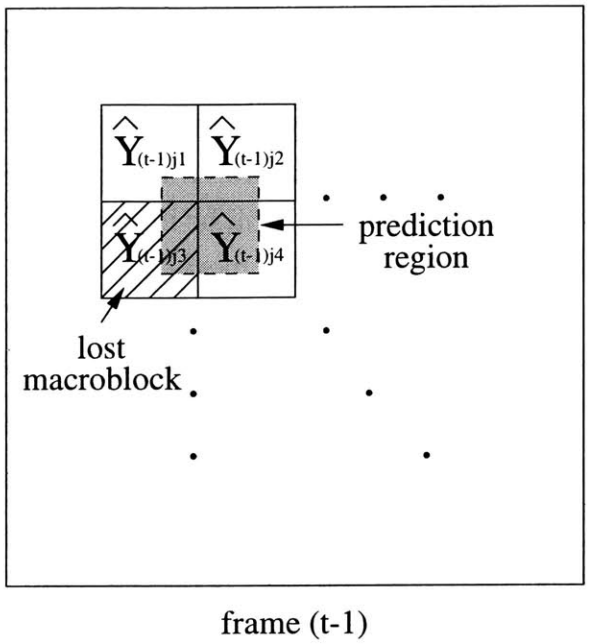


Figure 3.1: Example of lost macroblocks.

the different techniques.

3.2.1 Temporal Replacement

A simple method to conceal lost macroblock pixels in frame t is to replace them with the decoded macroblock pixels at the same location in the previous frame. This method is known as unshifted temporal replacement. If this region has also suffered from loss and has been concealed, then the same concealment pixels in frame $t-1$ are copied to the corresponding errored regions in frame t. This temporal replacement method works very well for regions of the video sequence where there is little or no motion. Stationary background regions are a perfect example of this. On the other hand, annoying artifacts and persistent frame discontinuities can occur when there is motion in the frame region that suffered loss.

3.2.2 Shifted Temporal Replacement

Shifted temporal replacement can use motion vectors to potentially avoid the frame discontinuities from unshifted replacement. For concealment in the case of *coded-block loss*, the motion vector for concealment is estimated, and the lost macroblock is replaced with the estimated prediction region. For concealment in the case of *prediction loss only* the correct motion vector is available at the decoder. The residual error for pixels with lost prediction is set to zero, and decoding proceeds as normal. This results in the lost pixels being replaced with the concealed prediction pixels.

In most moving sequences, there is high spatial correlation among motion vectors. Neighboring macroblocks tend to belong to the same object and move in the same direction. For this concealment algorithm, we use the motion vectors for the four nearest neighbors of a lost macroblock to interpolate the concealment motion vector. If the frame size is H macroblocks down and W macroblocks across ($M = H\text{x}W$), and the macroblocks are numbered in raster scan order, the neighborhood for macroblock X_{tj} in frame t corresponds to macroblocks X_{tj-W}, X_{tj-1}, X_{tj+1}, and X_{tj+W} (see figure 3.2). We actually use the vector median of the motion vectors for the arriving inter-coded neighboring macroblocks. Using a median filter to interpolate the motion vector, tends to produce better results than simply averaging neighboring motion vectors [39]. This is because to conceal macroblocks at motion boundaries, the motion vector of the object in the macroblock should be used for concealment as opposed to the average motion of differently moving objects, and median filtering provides a good guess for the correct motion vector. The median vector $\boldsymbol{v_m} \in \{\boldsymbol{v_{tj-W}}, \boldsymbol{v_{tj-1}}, \boldsymbol{v_{tj+1}}, \boldsymbol{v_{tj+W}}\}$ is the vector that satisfies the following inequality

$$\sum_{x\in\{j-W,j-1,j+1,j+W\}} \|\boldsymbol{v_m} - \boldsymbol{v_{tx}}\| \leq \sum_{x\in\{j-W,j-1,j+1,j+W\}} \|\boldsymbol{v_{ty}} - \boldsymbol{v_{tx}}\| \qquad y \in \{j-W, j-1, j+1, j+W\} \quad (3.1)$$

When no inter-coded neighboring macroblock arrives, we use simple temporal replacement with no shift for concealment. If only two inter-coded neighboring residual macroblocks arrive, we use the following, somewhat arbitrary preference order listed from highest to lowest below to select a concealment motion vector ($\boldsymbol{v_{tj-1}}$, $\boldsymbol{v_{tj+1}}$, $\boldsymbol{v_{tj-W}}$, $\boldsymbol{v_{tj+W}}$).

For median filtering with intra-coded neighbors, we can either consider the macroblocks to have no motion vector, or we can consider the motion vector to be $(0, 0)$ and

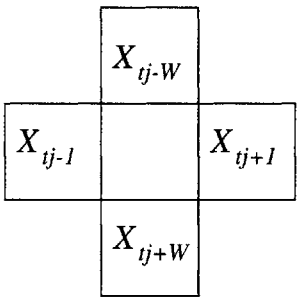


Figure 3.2: Macroblock neighborhood

include it in the vector median calculation. This may be appropriate for stationary background regions in which macroblocks are intra coded for robustness to loss. It will also encourage lost macroblocks surrounded mostly by intra-coded macroblocks to be concealed by simple temporal replacement with no shift. From empirical evidence, we decided that it is better to only use the motion vectors from the arriving inter-coded neighbors in the median calculation.

3.2.3 Bilinear Interpolation

Shifted temporal replacement is adequate for many scenarios. However, there are some sequences which have irregular motion or scene changes. With motion in the scene that is not translational within the frame it is difficult to estimate the correct concealment motion vector. In the case of a scene change or disocclusion, it is impossible to find the right replacement region in the previous frame for concealment. Better results might be achieved with spatial interpolation in these cases. For example, bilinear interpolation will result in good results for smooth image regions. Directional spatial interpolation methods might be more appropriate for lost regions which contain edges in the image frame.

In this section, we describe a bilinear concealment method which performs well for smooth regions and can also reconstruct horizontal and vertical edges that pass through the lost region well. In the case of *coded-block loss*, the whole macroblock is concealed by this spatial interpolation. For the case of *prediction loss only*, the arriving residual signal is added everywhere as normal. Since the residual signal will often contain edge information from the macroblock region, this will add correct structural information to the already smooth,

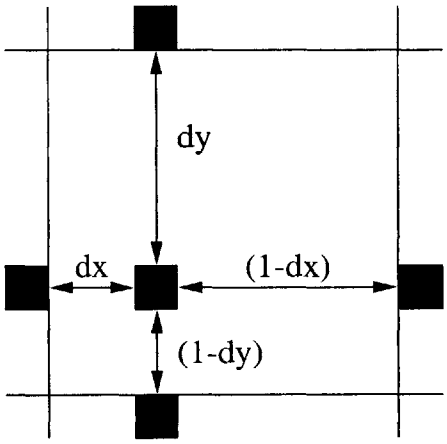


Figure 3.3: Bilinear interpolation concealment method.

concealed prediction regions.

With this bilinear concealment method, lost pixels in a macroblock are replaced by the weighted average of the four nearest pixels along the edge of each neighboring macroblock. An example of this is shown in Fig. 3.3. The weight used for each pixel is inversely proportional to the distance from the lost pixel. Let b_n, b_s, b_e, and b_w denote the value of neighboring pixels used for concealment to the north, south, east, and west of the lost pixel respectively. The pixel to be concealed is located y pixels down from the top edge of the macroblock and x pixels over from the left edge of the macroblock. For a macroblock size of 16x16 pixels, we define the normalized distances $d_x = x/(16+1)$ and $d_y = y/(16+1)$. The formula to calculate the concealed pixel value $\hat{b}_{tji}$ at location i in macroblock j at frame t is shown below.

$$\hat{b}_{tji} = \frac{(1-d_x)}{2} \cdot b_n + \frac{(d_x)}{2} \cdot b_s + \frac{(1-d_y)}{2} \cdot b_e + \frac{(d_y)}{2} \cdot b_w \tag{3.2}$$

Only correctly received and decoded values are used to conceal a lost macroblock. The weight of the corresponding lost neighbor is set to zero, and the rest of the weights are re-normalized to sum to one. If no surrounding macroblock is available, the lost macroblock is replaced by the mid-level gray value of 128.

3.2.4 Concealment Examples

We conducted a few informal subjective experiments to compare bilinear spatial interpolation to the two temporal interpolation methods discussed above as an error concealment approach[40]. We used QCIF video sequences (176×144 pixels, 30 frames/sec). Figs. 3.4 – 3.7 show frames of one of the sequences for different concealment techniques. For this sequence, the bit rate was fixed at 448 kbits/sec and the channel probability of losing each macroblock was 0.1.

Figs. 3.4 and 3.5 compare the spatial and temporal replacement concealment approaches. The first figure shows a frame for which the temporal interpolation strategy works best. Notice that the eyes and nose are missing from the picture on the left. The second figure shows a frame for which the spatial interpolation works best. Notice that the nose on the right picture has shifted and part of the face has moved to the hat. Similarly, Figs. 3.6 and 3.7 compare the spatial and shifted temporal concealment approach. The first figure shows a frame for which the temporal interpolation strategy works best. Again, notice the missing eye and nose on the left. The second figure shows a frame for which the spatial interpolation works best. Notice the broken mouth on the right.

As we discussed, spatial interpolation tends to blur the concealed block. This results in spatial and temporal discontinuities in image sharpness, which are very annoying. Temporal interpolation on the other hand, does not affect image sharpness; it creates discontinuities (broken lines) along the macroblock boundaries. Such artifacts are quite objectionable when we look at still frames. However, when we look at the moving images, temporal interpolation produces sequences that have uniform sharpness with occasional broken lines at block boundaries. Spatial interpolation, on the other hand, produces sequences with spatio-temporal discontinuities in sharpness, which is a lot more objectionable. Thus, the temporal interpolation approach results in lower perceptual distortion in all the cases that we considered.

3.3 Robust Encoding Methods

For robust encoding, the use of prediction should be limited to lower the possibility of error propagation with loss. However, as the number of macroblocks that are forced to be

Bilinear (Spatial) Interpolation

Temporal Replacement

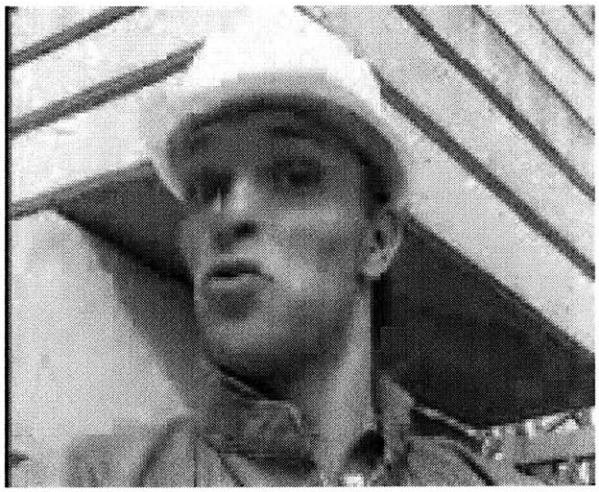
Figure 3.4: Interpolation comparison (QCIF image)

Bilinear (Spatial) Interpolation

Temporal Replacement

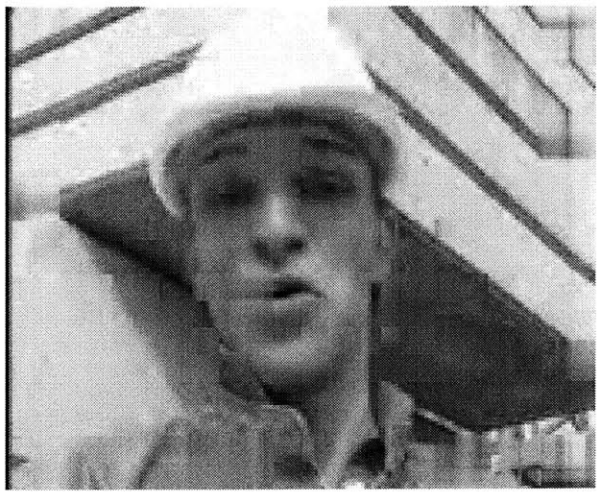
Figure 3.5: Interpolation comparison (QCIF image)

Bilinear (Spatial) Interpolation



Shifted Temporal Replacement

Figure 3.6: Interpolation comparison (QCIF image)



Bilinear (Spatial) Interpolation

Shifted Temporal Replacement

Figure 3.7: Interpolation comparison (QCIF image)

intra coded is increased, other tradeoffs must be made in video quality to meet the rate constraint. This will result in different forms of distortion when the rate constraint is too low. When only (or mostly) intra mode encoding is used, there are two straightforward methods to reduce the bit rate for coded video. The first is to increase the quantization step-sizes for the macroblocks, and the second is to reduce the number of macroblocks that are transmitted. Increasing the quantization step-size will result in more *DCT* coefficients that are quantized to zero. These are typically the high-frequency coefficients since they tend to have smaller amplitudes. Therefore, increasing the quantization step-size will result in blurry image frames. The simplest way to reduce the number of transmitted blocks is to lower the frame rate by dropping frames at the encoder. There is a limit to this method since distortion in motion rendition will result when the frame rate becomes too low. In this case, the video sequence without loss becomes equivalent to watching a slide show. Methods have been proposed to tradeoff a reduction in distortion from loss by mainly intra mode coding and reduction in the coding distortion that results when loss does not occur. In this section, we will review two current methods for robust encoding.

3.3.1 Conditional Replenishment

One method to reduce the number of coded macroblocks is to use conditional replenishment. In this method, all transmitted macroblocks are coded in intra mode for robustness to loss. However, only the macroblocks at locations in the current frame that change significantly from the previous frame are coded. This may result in better motion rendition than skipping frames. Conditional replenishment attempts to recapture some of the efficiency lost from restricting all transmitted blocks to intra mode encoding. With this method, macroblocks that are made up of stationary background are not repeatedly transmitted. Such static macroblocks may make up a significant portion of a scene in video conferencing. This however creates a dependency between the macroblocks in different frames since some locations are not updated and rely on the correct reception of the last sent update at the respective locations in the frame. For robustness to loss, a forced update algorithm can also be used to ensure that macroblock at all locations are at least periodically updated at a low rate. This will avoid persistent errors in the background from loss.

Conditional replenishment requires the accurate identification of macroblocks in stationary background which can be done initially before a macroblock is coded. This will

reduce the amount of computation at the encoder since macroblocks that are not transmitted will also not be coded. The sum of absolute differences between the original macroblock and the macroblock located at the same location in a reference frame is used to determine if a macroblock should be transmitted. We consider a macroblock with Z pixels and denote pixel $i \in \{1, 2, \ldots, Z\}$ in the reference macroblock as $X'_{(t-1)ji}$. The reference frame is made up of all the last macroblocks that were transmitted to the decoder for each location. If this difference is greater than a preset threshold T the macroblock is transmitted. We show the decision algorithm for transmitted macroblocks in the equation below.

$$\sum_{i=1}^{Z} \left| X_{tji} - X'_{(t-1)ji} \right| > T \tag{3.3}$$

Since all macroblocks are intra coded no motion estimation is performed nor prediction error computed. This raises a question about whether or not a copy of the decoder should be run at the encoder to produce an accurate reference frame to decide if a macroblock should be sent. This question becomes more important as the coding rate is lowered and more distortion results from quantization and coding. It may be desirable to use the decoded frame as a reference since the decision algorithm will cause static macroblocks that were coarsely quantized the last time they were transmitted to be updated. This strategy however may not be efficient with only intra coding, since at low bit-rates, the updated blocks may again be coarsely quantized resulting in similar distortion. In addition, using a portion of the bits allocated to a frame to re-code static macroblocks will cause all other transmitted macroblocks in a frame to be coded with higher distortion. If the decoded frame is used, the threshold should be raised to avoid sending too many static macroblocks with large distortion. Unfortunately, raising the threshold may result in missed detection and true macroblocks that change due to motion may not be coded.

On the other hand, using the original transmitted macroblocks in the reference frame has the potential to accurately identify stationary background with the decision algorithm in equation (3.3). This is the strategy used for conditional replenishment for the coder described in [12]. In addition, instead of using the sum of absolute differences, they use the absolute sum of differences which we found not to result in reliable decisions. From our experience, using the original macroblocks in the reference frame results in better detection and lower MSE between the decoded and original frame when no loss occurs. Given the above considerations, this is the approach that we use for conditional replenishment when

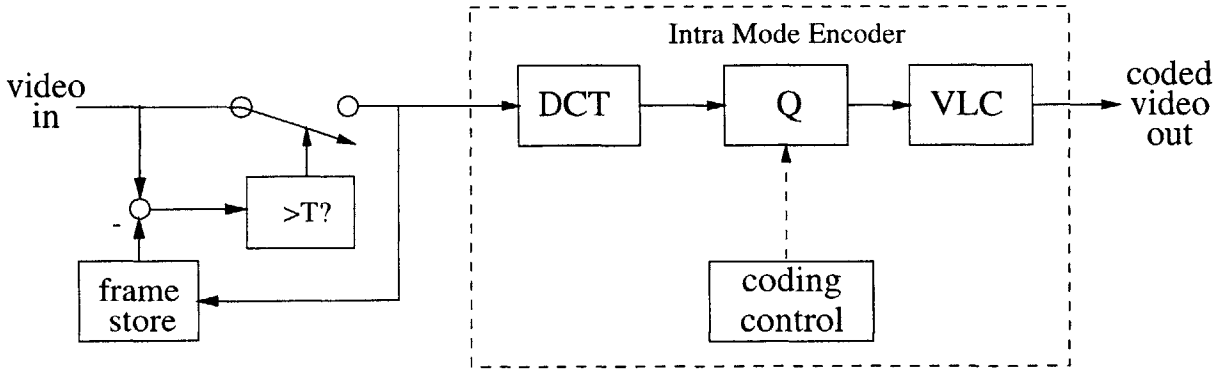


Figure 3.8: Conditional replenishment system.

we restrict ourselves to intra encoding alone. This reference frame is the image that would exist at the decoder in a perfect scenario where there is no distortion from coding or loss in the channel. In this case, we should use a lower threshold T. A block diagram for conditional replenishment algorithm is shown in Fig. 3.8.

3.3.2 Efficient Coding with Forced Intra Updates

Conditional replenishment can work well when only a few macroblocks must be transmitted in each frame. However, it will generally be very inefficient. For robust encoding, it might not always be necessary to code a macroblock in intra mode every time it is transmitted. Improvements may result when we use the much more efficient coding algorithm described in section 2.1.3 and force periodic intra updates to limit error propagation. Since not every transmitted macroblock is intra coded better frame quality will result when no loss occurs since smaller quantization step-sizes can be used while meeting the rate constraint. In addition, when the frequency of forced updates is large enough, distortion from loss might be sufficiently mitigated.

We begin with the algorithm described in 2.1.3 and force a macroblock in each location to be coded in intra mode at least once out of every U times it is transmitted. The suggested value for U in the $ITU-T$ coder recommendations is 132. This may be far too large for significant loss rates. The value for U is preset and can depend on the expected loss rate. Intuitively, higher loss rates should use lower values for U to obtain an appro-

priate tradeoff between quantization distortion and error propagation. The ideas used for conditional replenishment can also be added to this algorithm to avoid transmitting every macroblock. However, the decision algorithm described in the previous subsection may be too conservative when inter mode coding is allowed. Deciding not to transmit a macroblock before coding will prevent this algorithm from being able to transmit a residual that may be coded with a few bits to improve coarsely quantized static macroblocks. We decided to use the implicit decision for transmission that comes from normal coding when prediction is used. That is, we only prevent a macroblock from being sent when the motion vector is zero and all residual coefficients are quantized to zero. This designation is important since some coders allow for a dummy macroblock with motion vectors which do not transmit any coefficients. There is some additional coding overhead associated with sending such empty macroblocks with coded zero motion vectors. This can be eliminated for more efficiency.

When compared to the threshold comparison performed in intra-only conditional replenishment, this decision algorithm works well when the value for U is not too small. As the quantization step-sizes are raised to reduce the coding rate to get below R_c, more residual macroblocks will potentially be set to zero and will not be transmitted. As a result, this method will automatically determine which residuals can still be sent to meet the rate constraint and which should not be transmitted. We compared the MSE between the decoded sequence with no loss and the original sequence to verify that this decision algorithm typically results in smaller MSE than using the threshold method in equation (3.3). However, from our experience using the MSE for verification, we decided it is best to use the decision algorithm in equation (3.3) when the period between forced updates is very low (e.g. $U = 1, 2$). In these cases, the quantization distortion becomes too high and fewer macroblocks should be sent even at the risk of poor motion rendition.

3.4 Summary

There are two broad approaches to a solution to the problems of transmitting real-time video over networks. One attempts to solve all of the networking problems mentioned above with sharing and isolation designing a network capable of having efficient use of bandwidth while offering QOS. The other acknowledges that providing an adequate solution to all of the above problems is difficult at best and robust methods should be included in the end

systems to handle the expected packet loss. In addition, even if a network was designed to guarantee QOS at the expense of excess available bandwidth, a user may be willing to pay less money for services in the network that will not offer any guarantees, or it may not be desirable to wait until these guarantees can be provided. Given this, it would be beneficial to design a system capable of transmitting real-time video over unreliable channels. This would require an understanding of the relationship among the selected coding parameters, the QOS can be provided by the network, and error concealment methods at the receiver. This is a joint problem.

This thesis examines what can be done at the encoder to provide robustness to loss in the channel. To develop an optimal method at the encoder, a model for the packet loss that occurs in networks should be considered. Developing such a model alone is a difficult problem and is out of the scope of this research. Instead, we use a simple model for loss that we argue is still useful for understanding the problems in robust video coding. Specifically, the goal is to examine the tradeoff between compression which exploits the correlation in video to lower the transmission rate and the resilience to packet loss. The encoding method developed should be adaptive to the variable conditions in the channel for optimal performance. To measure the robustness of any proposed solution, it should be considered in conjunction with the network model as well as the concealment method that will be used at the decoder. Here, the goal is not to construct an elaborate concealment method in addition to the improvements provided at the encoder, but to determine optimal parameter selection for block-based video coders that will work well for simple yet reasonable concealment techniques. We present such a method in the next chapter.

Chapter 4

Optimization for Motion-Compensated Packet Video

In this chapter, we describe the problem for robust video coding for transmission over a packet-switched network. The principles gained from this chapter can be extended to develop robust coding methods over other unreliable channels. We first formulate an analytical problem to be solved for robust video coding when both the error characteristics of the channel and concealment method used at the decoder are known. The total mean-square-error (MSE) between the original video frame and the decoded frame at the receiver after concealment is used to measure the distortion from both quantization and channel error. We present an algorithm to select the optimal coding parameters to minimize this distortion metric for a given rate constraint. We perform simulations to compare the results of the optimal MSE solution to the previous methods for robust coding presented in Chapter 3.

4.1 Network Coding Problem Formulation

The block diagram for the total problem of unreliable transmission over networks is shown in figure 4.1. The sequence is first coded, assembled into packets, and then transmitted on the network. Data should be packed to minimize the effects of a single lost packet. Macroblocks which are the smallest coding units in block-based video coders should not be split across packet boundaries on a single priority network. In addition, redundant coding state information should be put into the header of every transmitted packet so that each packet can be independently decoded. An example of how data could be packed for many common block-based coding standards can be found in the Real-Time data Protocol RTP from the Internet Engineering Task Force ($IETF$)[22]. With the addition of sequence numbering in the packet header, lost data packets can be detected at the receiver, and packet loss becomes

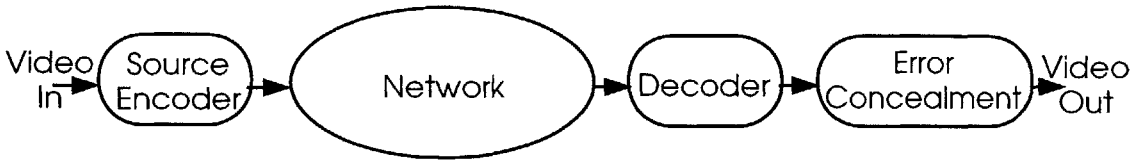


Figure 4.1: Block diagram of network video coding problem.

channel erasure. After decoding, error concealment is performed over regions affected by lost packets.

The coded sequence is to be transmitted in a network in near-real-time with a total end-to-end delay constraint t_D. The delay constraint determines the window of length N frames $(t, t+1, \ldots, t+N-1)$ over which optimization of the source coding parameters can be performed. In this thesis, we assume that the delay constraint t_D is set such that $N = 1$ and optimize the coding parameters causally one frame at a time given the selected parameters for all macroblocks in the previous frames. This assumption is reasonable since it keeps the delay time small and close to real-time. A rate constraint of R_c bits per frame is imposed to encourage efficient use of the network.

Another parameter that needs to be chosen is the frame rate F at which the video will be captured and ultimately displayed at the receiver. In practice given the average transmission bit rate , F can be set to any reasonable value to achieve a desirable trade-off in motion rendition and frame distortion, and it is not a parameter for our optimization. For an average transmission rate constraint R_{av} bits per second, the frame coding rate constraint is $R_c = R_{av}/F$ bits per frame.

For robust coding, we focus on the problem of selecting the modes and quantization step-sizes (l_{tj} and q_{tj}) for each macroblock in frame t to minimize distortion after concealment. We use the same notation for the original macroblocks $\boldsymbol{X} = \{X_{tj} \ \forall \ t, j\}$ and the decoded macroblocks at the encoder $\boldsymbol{Y} = \{Y_{tj} \ \forall \ t, j\}$ presented in section 2.1.2. Let $\hat{Y}_{tj}$ represent the macroblock in the t^{th} frame at spatial location j at the receiver after decoding and concealment, and $\hat{\boldsymbol{Y}} = \{\hat{Y}_{tj} \ \forall \ t, j\}$. In this case, the total distortion between $\boldsymbol{X}$ and $\hat{\boldsymbol{Y}}$ at the receiver consists of the following three components:

D^Q	frame distortion due to coefficient quantization
D^C	channel loss distortion after concealment
D^{mot}	motion rendition distortion which is affected by F

Similarly to what was done in section 2.1.3, we set the values of q_{tj} to be chosen such that the MSE between each macroblock X_{tj} and decoded macroblock Y_{tj} at the encoder is equal to some constant scaled by a weighting factor.

$$\|X_{tj} - Y_{tj}\|^2 = \alpha_{tj} \cdot K_t \tag{4.1}$$

The allowance for an arbitrary set of scaling factors α_{tj} keeps this problem formulation general. For an appropriate choice of scaling factors, we can constrain the decoded video to have consistent quality for each macroblock across a frame when measured with no loss. When optimization is performed on a frame by frame basis, R_c bits can be allocated between minimizing D^Q and minimizing D^C for a frame given all previous parameter selections. If there is either little potential for loss or lost macroblocks in the frame are relatively easy to conceal because of no motion, most of the bits can be allocated to minimize the quantization distortion D^Q. When the loss rate is high or a particular frame is difficult to conceal, this constraint will allow the coder to introduce more quantization distortion which will be spread evenly across the frame while increasing the robustness to loss by coding more macroblocks in intra mode.

The main problem which has not been fully addressed by traditional robust coding methods is a way to best allocate R_c bits between intra-mode encoding and coefficient quantization for a frame. We now begin to form an analytical problem and present a solution to determine if improvements can be made over the ad hoc approaches in traditional methods. The problem we propose is to choose for frame t the coding modes l_{tj} $\forall j$ and the frame distortion constant K_t given all past decisions to minimize some appropriate distortion function $D_t(\cdot)$ for the decoded and concealed frame at the receiver. For a video sequence with Z pixels per macroblock, we define $D_{tji}(K_t, l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_f, l_{fs}\ f < t, \forall s)$ to be the distortion for pixel i in macroblock j at frame t given the coding parameter decisions for all frames less than t. We investigate solutions to minimize a distortion function of the form

$$D_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_f, l_{fs}\ f < t, \forall s) = \sum_{j=1}^{M} \sum_{i=1}^{Z} D_{tji}(K_t, l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_f, l_{fs}\ f < t, \forall s) \tag{4.2}$$

The total distortion for frame t given the past can be expressed as an accumulation of the distortion for each pixel in every macroblock in the frame. In general, there is a dependency for the distortion of a pixel in a macroblock at frame t on the mode selection for the macroblocks up to and including those in frame t. This is from potential error propagation in motion-compensated inter-mode coded macroblocks as well as from the use of arbitrary causal concealment methods.

To ensure that the rate constraint is met, we must compute the rate given all past coding decisions to code a frame as a function of the current coding parameters. The total rate to code frame t can be expressed as the sum of bits used to code each macroblock.

$$R_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_f, l_{fs}\ f < t, \forall s) = \sum_{j=1}^{M} R_{tj}(K_t, l_{tj-1}, l_{tj} \mid K_f, l_{fs}\ f < t, \forall s) \tag{4.3}$$

In general, the rate R_{tj} to code a particular macroblock depends on the frame MSE constant K_t and both l_{tj} and l_{tj-1} the mode selections for the current and neighboring macroblock. The dependency on l_{tj-1} is due to differential encoding of macroblock coding parameters which is used in most block-based coders. For example, some coders use differential encoding for the DC coefficients of the blocks in successive intra-coded macroblocks X_{tj-1} and X_{tj} in a frame. In addition, many standard coders differentially encode the motion vectors for successive inter-coded macroblocks. Of course, only the rate function for macroblocks coded with prediction care about the coding parameter decisions of macroblocks in the previous frame $(K_f, l_{fs}\ f<t, \forall s)$. The motion vectors for those macroblocks determine exactly which macroblocks in the previous frames are used for prediction.

Before we derive the distortion function $D_t(\cdot)$ to minimize, we first choose an error metric $d\left(\boldsymbol{X_t}, \hat{\boldsymbol{Y}}_t\right)$ to measure the difference between the original and decoded/concealed frame respectively. We defer for now a description of exactly how this error metric will be incorporated into the distortion function $D_t(\cdot)$. For the problem of optimal bit allocation between frame quality and robustness to loss, the metric we decided to use is the total MSE between the luminance component for the original and concealed frame at the decoder.

$$d\left(\boldsymbol{X_t}, \hat{\boldsymbol{Y}}_t\right) = \|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_t\|^2 \tag{4.4}$$

This will combine both the error from quantization and the error from loss and concealment. To see the benefits of using this metric, consider the outcome in the case where there is

no potential for loss. Here, all the bits will be allocated to result in a decoded sequence with minimum quantization error for that rate which is what would be desired. In the case where there is the potential for loss, bits will be traded off between intra coding and accurate coefficient representation. Exactly how they are traded off depends on the loss rate and the ability for adequate concealment in the frame. If too many macroblocks are coded in intra at the rate R_c, the distortion from quantization will become too high, whereas if too few macroblocks are coded in intra mode, the error from poor concealment will dominate the MSE. By looking at the total MSE one can determine an optimal trade-off point between both sources of distortion. What will be shown through simulation is that this operating point is pleasing to the viewer and an improvement over traditional approaches.

The coding decisions must be made at the encoder without any knowledge of what will be lost at the receiver. In addition, since we do not consider the use of explicit feedback to the sender for each lost packet, there is no detailed knowledge of what past macroblocks have been lost. Excessive feedback with loss will increase the network congestion. We model the channel and make optimal coding parameter decisions at the encoder to minimize distortion within our model framework. In the next section, we describe the model we use for packet loss and derive the distortion function $D_t(\cdot)$ that is minimized.

4.2 Channel Model and Distortion Function

Accurately modeling packet loss has proven to be very problematic for researchers. In line with other attempts to characterize this process, we decided to use probabilistic models to describe the packet loss that occurs in the network. To facilitate the parameter selection at the encoder that is performed for each macroblock, we focus here on modeling the resulting macroblock loss that occurs with packet loss. It is known that in general packet loss is a bursty process, and it has been shown that certain probabilistic models with correlated loss can describe the loss observed on a network[41]. Unfortunately, analysis becomes much more difficult when using these probabilistic models.

In our analysis, we model the macroblock loss probability as an independent Bernoulli process where each macroblock is lost with probability p. This is still useful for the following three reasons. First, subjective tests show the quality degradation is greater with independent random loss than for bursty loss at equal loss rates [42]. This is because the number

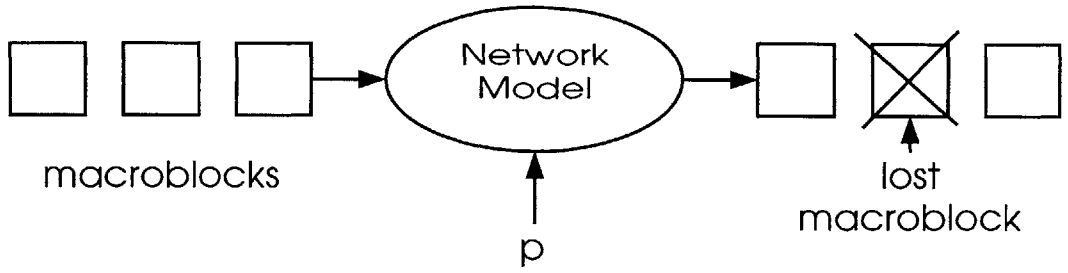


Figure 4.2: Channel loss model.

of different distorted regions in the sequence is increased with random loss. Thus, simulations with random loss are likely to provide an upper bound in quality degradation. Second, block interleaving techniques may be applied before packing to randomize the bursty loss. This practice is also beneficial for concealment since many concealment algorithms rely on information from neighboring macroblocks for interpolation. Finally, the solution derived using this simple model may also be useful in cases with correlated loss. At the very least, it may offer some insight into optimal coding mode selection in these more difficult cases. Figure 4.2 shows a block diagram of our channel model given the assumption of single independent macroblock losses.

Since the channel model is probabilistic, the encoder can select coding parameters to minimize the expected total MSE between the original frame and the concealed frame. Thus, we define the distortion function for frame t to be

$$D_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_f, l_{fs}\ f < t, \forall s) = E\left[\|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2\right]. \tag{4.5}$$

To compute the expectation at the encoder, all of the different macroblock loss combinations must be considered along with the probability of each event which is a function of the channel loss rate p and the mode selection for the macroblocks.

From our packing strategy, each arriving macroblock can be decoded at the receiver regardless of past losses that may have occurred on the network. However when decoding macroblocks coded in inter mode, not all of the pixels of the prediction region in the previous frame are available. Even though any arriving residual error can be decoded, the pixel error for pixels with lost (concealed) prediction pixels are useless. Therefore, these pixels are also considered lost and must be concealed. On the other hand, the effective loss probability for

intra-coded pixels is always smaller and is just equal to p for our model.

To compute the distortion function, the decoder concealment method must be known at the encoder. Concealment can be performed for different loss scenarios, and the MSE calculated. We use shifted temporal replacement described in section 3.2 for the concealment of lost macroblocks in our algorithm described in the following sections. In several simulations that we performed, this temporal interpolation method produced better perceptual quality when compared with spatial interpolation for the same losses[40]. Spatial interpolation does not usually perform well over the large 16x16 macroblock size used in most block-based coders. In addition, the lack of temporal correlation between macroblock locations that are spatially interpolated in successive frames often results in distracting perceptual artifacts when viewing the video sequence after loss and concealment. Although the solution we present is for this particular concealment method, it is applicable to other concealment methods that use a small local neighborhood for concealment. This would require some modification in the algorithm to compute the distortion function for each macroblock.

4.3 Nonserial Dynamic Programming

In this section, we briefly explain nonserial dynamic programming which is used to find a solution to our problem. Any reader who is comfortable with dynamic programming can skip this section. This algorithm is essentially the same Viterbi algorithm described in section 2.1.2. The difference is that in general, the dependency in problems may not be serial, and the order in which variables should be eliminated is not straightforward. In addition, it may not be easy to identify the possible states to use at each stage to readily form a trellis. The efficiency gained over a full search algorithm for a nonserialized problem is a direct function of the order in which variables are eliminated. In fact, if the order in which variables are eliminated is not chosen properly, the algorithm can even be less efficient than performing a full search for optimization. We explain nonserial dynamic programming by the following example. We want to minimize a function of four variables $C(w, x, y, z)$, and each variable can take on one of two possible values ($w, x, y, z \in \{0, 1\}$). The total cost function is composed of the sum of four smaller functions as shown below.

$$C(w, x, y, z) = e(w, z) + f(y, z) + g(w, y) + h(x, y) \tag{4.6}$$

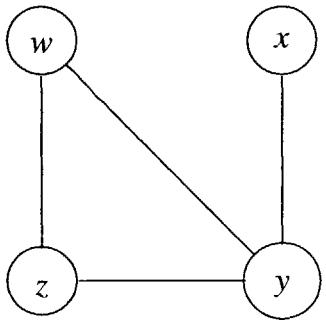


Figure 4.3: Dependency graph for example problem.

We can find the optimal solution by computing all $2^4 = 16$ possibilities of the input variables and performing 15 comparisons. However, we see that each of the four smaller functions in the sum do not have full dependency on all four input variables. We show a graph for the dependency of the input variables in Fig. 4.3. Only variables that jointly influence a single function in the sum in (4.6) are connected. Note that the graph is not fully connected. This is a perfect opportunity for the use of dynamic programming to solve the total problem by solving several smaller problems resulting in overall less computation. The smaller functions are listed in the tables below.

w	z	$e(w,z)$
0	0	5
0	1	7
1	0	12
1	1	17

y	z	$f(y,z)$
0	0	29
0	1	17
1	0	9
1	1	14

w	y	$g(w,y)$
0	0	19
0	1	14
1	0	6
1	1	8

y	x	$h(x,y)$
0	0	24
0	1	18
1	0	6
1	1	15

We group together all functions that depend on the variable w in (4.6) to form the new function

$$c_1(w,y,z) = e(w,z) + g(w,y). \tag{4.7}$$

The total cost function can be rewritten as

$$C(w,x,y,z) = c_1(w,y,z) + f(y,z) + h(x,y). \tag{4.8}$$

We can now eliminate w from the optimization problem by minimizing the following subproblem over the variable w as a function of y and z.

$$c_1^*(y,z) = \min_w c_1(w,y,z) \tag{4.9}$$

The solution to this subproblem is stored as shown in the following table.

y	z	$c_1^*(y,z)$	w^*
0	0	18	1
0	1	23	1
1	0	19	0
1	1	21	0

Four comparisons must be performed to compute this table, and the optimal w^* can be read from the table when the optimal y and z are later determined. Thus w has been eliminated from the problem and the new problem to minimize is only a function of three variables as shown below.

$$C'(x,y,z) = c_1^*(y,z) + f(y,z) + h(x,y) \tag{4.10}$$

The dependency graph for the remaining problem is equivalent to Fig. 4.3 with the w node removed. We next group together all of the functions in (4.10) that depend on the variable z and eliminate that variable by forming a new optimal function of the variable y.

$$c_2^*(y) = \min_z [c_1^*(y,z) + f(y,z)] \tag{4.11}$$

Two comparisons are performed to compute this new optimal function which is shown in the table below.

y	$c_2^*(y)$	z^*
0	40	1
1	28	0

After the variable z has been eliminated, the new total cost $C''(x,y) = c_2^*(y) + h(x,y)$ only depends on the variables x and y. We can do a full search over all the possibilities performing three comparisons to find the optimal solution which is shown by an asterisk in the following table.

x	y	$C''(x,y)$
0	0	64
0	1	46
1	0	46
1	1	43*

A total of nine comparisons is needed to minimize (4.6) using dynamic programming to find the minimum cost $C(0,1,1,0) = 43$. This is a 40% savings in computation over performing a full search over the original cost function. As mentioned before, the amount of savings depends on the order in which variables are eliminated. In addition, this order determines the size of the largest table (memory) needed at any stage to perform the optimization of the subproblems. For example, if we were to eliminate the variable y at the first stage, the optimal y^* must be found as a function of **three** variables (w,x,z) which would result in a larger table at this stage. In addition, eight comparisons would have to be performed at this stage alone. The size of the table required at any stage is important because it determines the maximum amount of memory needed at any one time. Once the next stage is optimized, only pointers to the optimal value for the variable eliminated at a current stage must be saved as a function of the remaining variables. The order of elimination plays a crucial role for larger problems and will determine whether optimization can be performed on practical systems with limited memory and computational resources. In addition, there is nothing that restricts variables to be eliminated one at a time. We have just given a brief introduction into the problem of nonserial dynamic programming which will provide the reader with enough information to follow the algorithms presented in this thesis. We refer the reader to [43] for more information on nonserial dynamic programming.

4.4 Solution

Given the distortion function $D_t(\cdot)$ derived above, we restate the constrained optimization problem to be solved below.

$$\min_{K_t, l_{t1}, l_{t2}, \ldots, l_{tM}} D_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM}),$$
$$\textit{subject to } R_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM}) \leq R_c \tag{4.12}$$

For notational convenience, the dependency on the given coding parameters ($K_f, l_{fs}\ f<t, \forall s$) in previous frames is omitted with the understanding that it still exists. We use a Lagrange multiplier to form an equivalent unconstrained problem to find a solution to (4.12). In this new formulation, we minimize for an appropriate choice of $\lambda_t \geq 0$ the following Lagrangian cost function.

$$C(K_t, l_{t1}, l_{t2}, \ldots, l_{tM}) = R_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM}) + \lambda_t \cdot D_t(K_t, l_{t1}, l_{t2}, \ldots, l_{tM}) \tag{4.13}$$

In section 2.1.2, we showed the relationship between the solution that minimizes an unconstrained Lagrangian cost function to the optimal solution for the corresponding constrained optimization problem. If the optimal rate $R_t^*(\lambda_t)$ that results from minimizing (4.13) for a particular value of λ_t is equal to the rate constraint R_c, then the corresponding optimal parameter selection is also a solution to the constrained problem in (4.12). Since there are only a discrete set of coding parameter selection possibilities and resulting rates, not any arbitrary value for R_c can be achieved exactly. A solution to the above unconstrained problem is equivalent within a convex hull approximation in the rate-distortion plane to the solution of the constrained problem in (4.12). We assume a high density of points of possible rate-distortion pairs for the different coding parameter selections. Thus, the convex hull approximation will be sufficiently close to the optimal solution. The monotonic relationship between λ_t and $R_t^*(\lambda_t)$ means the appropriate choice of λ_t to satisfy a specific rate constraint can easily be found by an iterative bisection algorithm.

We perform an iterated optimization to solve for the coding parameters to minimize (4.13). We describe an algorithm in the next subsection to select the optimal modes to

minimize

$$C(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t) \tag{4.14}$$

which is the Lagrangian cost function given a particular value for K_t. Then, we show how to select K_t to minimize the total cost function for frame t.

4.4.1 Finding Optimal Modes with Fixed K_t

This algorithm consists of two steps: data generation of the necessary rate-distortion points to compute the cost function $C(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t)$ and dynamic programming to find an optimal solution. We will describe each in turn below.

4.4.1.1 Data Generation

At frame t, the optimal coding parameters for all frames less than t have been computed, and these frames have been coded and transmitted. In addition, a decoded copy of frame $t-1$ is stored at the encoder for inter-frame coding. Given this decoded frame, motion estimation is done to find the best prediction region in that frame for each macroblock in the current frame t. This prediction region is only used at the encoder if the macroblock is coded in inter mode. Two coding passes are performed over frame t before it is transmitted. The first coding pass is done for data generation to calculate the rate and distortion points for all of the possible mode selection combinations. After the optimal coding parameters have been selected, a second pass is done to actually code and transmit the frame.

Consider each frame to be divided into separate regions consisting of one or more rows of macroblocks called group of blocks GOB. Data generation is done on a macroblock by macroblock basis in each GOB to compute the rate and distortion function. The macroblocks are coded in raster scan order, and the rate value $R_{tj}(l_{tj-1}, l_{tj} \mid K_t)$ to code each macroblock is computed as a function of all four possible coding mode combinations of $l_{tj-1} \in \{0,1\}$ and $l_{tj} \in \{0,1\}$ for the previous and current macroblock respectively. Of course, there are only two possible rates $R_{tj}(l_{tj} \mid K_t)$ for the first macroblock in a GOB since it does not use differential parameter encoding. Before we can calculate the rate function, we must first find the coefficient quantization step-size q_{tj} for the macroblock to meet the MSE criterion

$\alpha_{tj} \cdot K_t$ when coded in intra mode and the q_{tj} to meet the criterion when coded in inter mode. In most standard coders, $q_{tj} \in \{1, 2, \ldots, 31\}$. Since the MSE is monotonic as a function of q_{tj}, a bisection algorithm can be performed over all discrete quantization step-sizes to find the q_{tj} that results in MSE closest to the given $\alpha_{tj} \cdot K_t$.

The expected distortion function for a macroblock in frame t equals the sum of the macroblock pixel distortions as shown below.

$$D_{tj}(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t) = \sum_{i=1}^{Z} D_{tji}(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t) \tag{4.15}$$

If we were to do a straightforward calculation to find it, we would first have to enumerate all of the loss possibilities at the encoder. Then, for each loss scenario we would have to compute the MSE after simulating the loss and concealment at the encoder and weight it by the probability of that loss event. However, such a straightforward approach would lead to an infeasible amount of computation in practice. For example, at frame t with M macroblocks per frame there are $2^{(t \cdot M)}$ loss combinations. For even the low resolution Quarter-Common-Intermediate-Format ($QCIF$) frame that has $M = 99$ macroblocks, this is an immense number of calculations.

Fortunately, we can calculate the marginal probability distribution over the discrete number of possible pixel luminance values for each decoded/concealed pixel given the concealment method and all past coding decisions. At the encoder, we denote the original luminance value for pixel i in macroblock j at frame t as b_{tji} and the decoded/concealed luminance value by the random variable $\hat{b}_{tji}$. [1] With an 8-bit representation for the pixel luminance component, we have a probability distribution for $\hat{b}_{tji}$ as displayed for example in figure 4.4 over the integers $\{0, 1, \ldots, 255\}$ for a particular set of coding mode selections l_{tj} $\forall$ j in t and channel loss probability p. Hopefully, when the mode selection is performed to minimize the expected total MSE, this distribution will peak around the correct decoded value $\hat{b}^*_{tji}$ as indicated by an asterisk in figure 4.4. At this decoded value, the only error will be from quantization noise.

For any set of mode selections l_{tj} $\forall$ j and the resulting set of marginal distributions $p\left(\hat{b}_{tji}\right)$ $\forall$ i in j, the distortion function for macroblock j is easily calculated by the following

[1] We use the same notation to indicate the random variable at the encoder and the actual value at the decoder. At any time, the intended meaning is clear from the context.

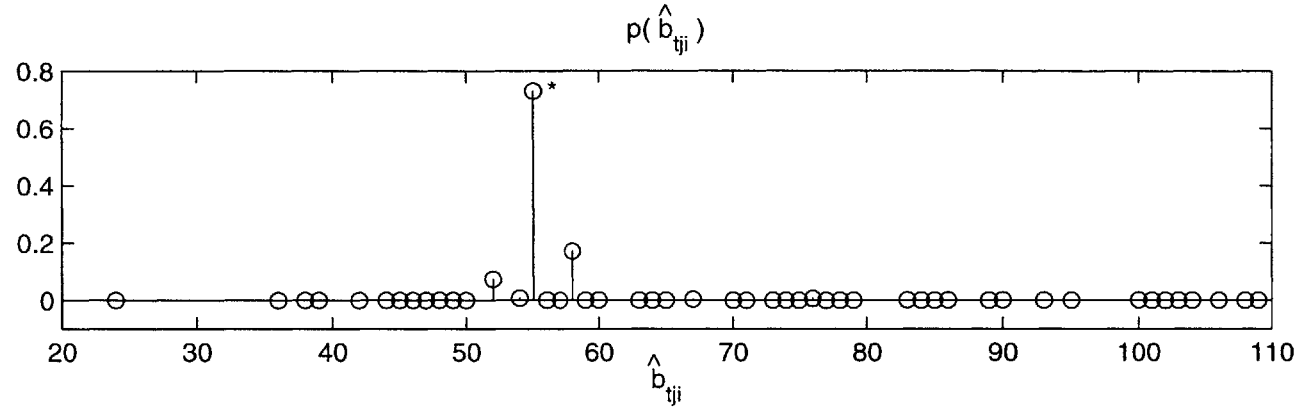


Figure 4.4: Sample marginal probability distribution for pixel luminance.

formula.

$$D_{tj}(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t) = \sum_{i=1}^{Z} \sum_{\hat{b}_{tji}=0}^{255} p\left(\hat{b}_{tji}\right) \cdot \left(b_{tji} - \hat{b}_{tji}\right)^2 \tag{4.16}$$

The explicit dependency of the macroblock distortion function $D_{tj}(\cdot)$ in equation (4.16) is from the dependency of $p\left(\hat{b}_{tji}\right)$ on the mode selections l_{tj} $\forall$ j in t. For a concealment method that uses temporal interpolation, each $p\left(\hat{b}_{tji}\right)$ is also a function of the marginal pixel probability distributions $p\left(\hat{b}_{(t-1)ji}\right)$ in the previous decoded frame. These marginal distributions for $\hat{b}_{(t-1)ji}$ $\forall$ j, i along with the previous decoded frame is all that needs to be stored from the past and adequately summarizes all of the previous given frame MSE constants K_f and coding mode selections l_{fs} $f < t, \forall s$. We emphasize that this is the marginal distribution, and even though macroblock loss may be independent in our loss model, the random variables for pixel luminance in a frame are not independent. This can be seen for example when more than one pixel in a frame use the same pixel in the previous frame for either prediction or for concealment. Then the random variables representing the luminance for these decoded/concealed pixels are correlated. The joint distribution for $\hat{b}_{tji}$ $\forall$ j, i in t is what really gives a complete probability description for the pixel luminance values in a frame. However, we will show that for our loss model and concealment method the marginal distribution for each decoded/concealed pixel in the previous frame $t-1$ is sufficient to compute the marginal distributions in the current frame.

We now show how to combine the probability distributions $p\left(\hat{b}_{(t-1)ji}\right)\ \forall\ j, i$ from the previous frame to compute $p\left(\hat{b}_{tji}\right)$ in the current frame as a function of the current mode selection variables $l_{tj}\ \forall\ j$. The coding modes and motion vectors to code macroblock X_{tj} and its four neighboring macroblocks used in the concealment method determine both the probability that the macroblock will be lost given the past coding decisions and the regions in the previous frame which can be used for concealment. The macroblock distortion function and the residing marginal probability distributions in equation (4.16) only have dependency on mode selections for these five macroblocks $D_{tj}(l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W} \mid K_t)$. The motion vectors ($\boldsymbol{v_{tj-W}}$, $\boldsymbol{v_{tj-1}}$, $\boldsymbol{v_{tj}}$, $\boldsymbol{v_{tj+1}}$, $\boldsymbol{v_{tj+W}}$) for these macroblocks give the five possible concealment vectors when the current macroblock is lost at the decoder. At the encoder, we denote the selected concealment vector for a lost macroblock by the random vector $\boldsymbol{v_s}$. We define a translation function $g(j, i, \boldsymbol{v_s})$ which takes the macroblock and pixel location (j, i) along with the motion vector v_s and returns the new shifted macroblock and pixel location (j', i').

The possible values for the random variable $\hat{b}_{tji}$ include the correct decoded value $\hat{b}^*_{tji}$ that results when the pixel is not affected by loss and all the possible values for the five random variables $\hat{b}_{(t-1)g(j,i,\boldsymbol{v_s})}$ at the encoder which represent the five possible concealment pixels used when the pixel is lost. The probability distribution function for $\hat{b}_{tji}$ can be decomposed into the sum of distributions of these two mutually exclusive events.

$$p\left(\hat{b}_{tji}\right) = p\left(\hat{b}_{tji} = \hat{b}^*_{tji}\right) + p\left(\hat{b}_{tji},\ X_{tji}\ lost\right) \tag{4.17}$$

We first compute the probability that there is no loss and $\hat{b}_{tji}$ equals $\hat{b}^*_{tji}$ which is a function of the macroblock mode selection l_{tj} and is given by

$$p\left(\hat{b}_{tji} = \hat{b}^*_{tji}\right) = \begin{cases} (1-p) & \text{if } l_{tj} = 0 \\ (1-p)\cdot p\left(\hat{b}_{(t-1)g(j,i,\boldsymbol{v_{tj}})} = \hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}\right) & \text{if } l_{tj} = 1 \end{cases} \tag{4.18}$$

When the macroblock is coded in intra mode, it is equal to the probability that the coded block arrives. If it is coded in inter mode, it is equal to the probability that both the residual arrives and the prediction is correct. To compute the probability in this case, the correct value $\hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}$ can be determined from the stored previous decoded frame $\mathbf{Y_{t-1}}$ at the encoder, and the probability value $p\left(\hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}\right)$ can be read from the probability

distribution function for the prediction pixel which is also stored at the encoder.

We now discuss how $p\left(\hat{b}_{tji},\ X_{tji}\ lost\right)$ is calculated when X_{tj} is coded in intra mode ($l_{tj} = 0$) and inter mode ($l_{tj} = 1$) separately. When X_{tj} is coded in intra mode and lost, there is no $\boldsymbol{v_{tj}}$ that can be used for concealment at the encoder. This leaves four possible concealment vectors. Likewise, when any of the other macroblocks in the neighborhood are intra coded, the corresponding motion vectors are not available even when the coded macroblocks are not lost.

The probability distribution $p\left(\hat{b}_{tji},\ X_{tji}\ lost \mid l_{tj} = 0\right)$ for a pixel given X_{tj} is coded in intra mode is equal to the sum of the joint distributions $p\left(X_{tji}\ lost,\ \boldsymbol{v_s} = \boldsymbol{v_x},\ \hat{b}_{(t-1)g(j,i,\boldsymbol{v_x})}\right)$ for the four possible concealment vectors $\boldsymbol{v_x} \in \{\boldsymbol{v_{tj-W}},\ \boldsymbol{v_{tj-1}},\ \boldsymbol{v_{tj+1}},\ \boldsymbol{v_{tj+W}}\}$ from neighboring macroblocks. Since macroblock loss is independent each joint distribution is given by the product of the three marginal probabilities. Therefore, $p\left(\hat{b}_{tji},\ X_{tji}\ lost \mid l_{tj} = 0\right)$ is a linear combination of the marginal distributions for the possible concealment pixel luminance values in the previous frame.

$$p\left(\hat{b}_{tji},\ X_{tji}\ lost \mid l_{tj} = 0\right) = \sum_{x \in \{j-W, j-1, j+1, j+W\}} p \cdot p(\boldsymbol{v_s} = \boldsymbol{v_x}) \cdot p\left(\hat{b}_{(t-1)g(j,i,\boldsymbol{v_x})} = \hat{b}_{tji}\right) \tag{4.19}$$

The macroblock loss probability p is given and the marginal probabilities for the pixels in the previous frame are stored at the encoder. The only item left to be computed is the probability distribution $p(\boldsymbol{v_s})$ for the possible concealment vectors.

Since the actual concealment vector that is used at the decoder for a lost macroblock is a function of the arriving motion vectors, we first create a table for macroblock X_{tj} of the resulting concealment vector $\boldsymbol{v_s}$ as a function of all possible arrival combinations of ($\boldsymbol{v_{tj}}$, $\boldsymbol{v_{tj-1}}$, $\boldsymbol{v_{tj+1}}$, $\boldsymbol{v_{tj-W}}$, $\boldsymbol{v_{tj+W}}$). This is shown in table 4.1 where a zero signifies that the vector is not available either because the macroblock is coded in intra mode or the residual is lost, and $\boldsymbol{v_0}$ represents the zero vector. When more than two vectors arrive the median function is performed to select a vector for $\boldsymbol{v_s}$, and we indicate the result of median filtering by $\boldsymbol{v_m}$ in the table. Given this table and independent macroblock loss probability p, it is simple to calculate $p(\boldsymbol{v_s})$ for the five possible values for v_s as a function of the 32 mode selection combinations (l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W}) for the current macroblock and its four neighbors.

v_{tj}	v_{tj-1}	v_{tj+1}	v_{tj-W}	v_{tj+W}	v_s
0	0	0	0	0	v_0
0	0	0	0	1	v_{tj+W}
0	0	0	1	0	v_{tj-W}
0	0	0	1	1	v_{tj-W}
0	0	1	0	0	v_{tj+1}
0	0	1	0	1	v_{tj+1}
0	0	1	1	0	v_{tj+1}
0	0	1	1	1	v_m
0	1	0	0	0	v_{tj-1}
0	1	0	0	1	v_{tj-1}
0	1	0	1	0	v_{tj-1}
0	1	0	1	1	v_m
0	1	1	0	0	v_{tj-1}
0	1	1	0	1	v_m
0	1	1	1	0	v_m
0	1	1	1	1	v_m
1	x	x	x	x	v_{tj}

Table 4.1: Concealment motion vector selection as function of arriving motion vectors.

As discussed in section 3.2 when X_{tj} is coded in inter mode, there are two mutually exclusive loss cases that can occur: coded-block-lost and prediction-lost-only. As a result, $p\left(\hat{b}_{tji},\ X_{tji}\ lost \mid l_{tj}=1\right)$ can be divide into the sum of two terms.

$$p\left(\hat{b}_{tji},\ X_{tji}\ lost \mid l_{tj}=1\right) = p\left(\hat{b}_{tji},\ block\text{-}lost\right) + p\left(\hat{b}_{tji},\ pred.\text{-}lost\text{-}only\right) \tag{4.20}$$

When the residual block is lost, $\boldsymbol{v_{tj}}$ is not available, and the situation for concealment is identical to an intra block loss. Therefore, the formula for $p\left(\hat{b}_{tji},\ block\text{-}lost\right)$ is identical to equation (4.19).

When the residual block arrives but the prediction is lost, the concealment vector $\boldsymbol{v_s}$ is equal to the correct $\boldsymbol{v_{tj}}$, and we do not have to sum over all of the possible values for $\boldsymbol{v_s}$ to compute $p\left(\hat{b}_{tji},\ pred.\text{-}lost\text{-}only\right)$. In addition, the scaling factor becomes $(1-p)$. However, when the residual block arrives, and the prediction pixel is not lost and is equal to its correct value $\hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}$, this value is not used for concealment. Instead, the residual error is added to $\hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}$ to obtain the new value $\hat{b}^*_{tji}$. This case is accounted for in equation (4.18). Therefore, we only scale the values for $\hat{b}_{(t-1)g(j,i,\boldsymbol{v_{tj}})} \neq \hat{b}^*_{(t-1)g(j,i,\boldsymbol{v_{tj}})}$, and the resulting formula for this case is given below.

$$p\left(\hat{b}_{tji},\ pred.\text{-}lost\text{-}only\right) = (1-p)\cdot p\left(\hat{b}_{(t-1)g(j,i,\boldsymbol{v_{tj}})} = \hat{b}_{tji},\ X_{(t-1)g(j,i,\boldsymbol{v_{tj}})}\ lost\right) \tag{4.21}$$

After computing $p\left(\hat{b}_{tji}\right)$ for each pixel in X_{tj} as a function of the mode selections $(l_{tj-W},\ l_{tj-1},\ l_{tj},\ l_{tj+1},\ l_{tj+W})$, we use equation (4.16) to compute a table of macroblock distortion values $D_{tj}(l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W} \mid K_t)$ for the 32 possible mode selection combinations for X_{tj} and its neighborhood. For a particular value of λ_t and given the tables for the rate and distortion function for X_{tj}, the Lagrange cost function for each macroblock can be tabulated.

$$\begin{aligned} C_{tj}(l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W} \mid K_t) = \\ R_{tj}(l_{tj-1}, l_{tj} \mid K_t) + \lambda_t \cdot D_{tj}(l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W} \mid K_t) \end{aligned} \tag{4.22}$$

The total Lagrange cost function is given by the sum of the macroblock cost functions.

$$C(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t) = \sum_{j=1}^{M} C_{tj}(l_{tj-W}, l_{tj-1}, l_{tj}, l_{tj+1}, l_{tj+W} \mid K_t) \tag{4.23}$$

If there were no dependencies among the different macroblock cost functions $C_{tj}(\cdot)$, the total cost could be minimized by minimizing each macroblock cost function independently. However, there is explicit overlap from the neighborhood structure which creates a first-order dependency among the cost functions of neighboring macroblocks. An exhaustive search can not be performed over the M mode selection variables for a frame to minimize (4.23). For even with the low, $QCIF$ resolution frame, 2^{99} cost function values would have to be computed, and $2^{99}-1$ comparisons performed to optimize the variables for a single frame. This is very impractical. However, since there is not complete dependency among $l_{tj}\,\forall\, j$ in any of the terms in (4.23), dynamic programming may be performed for optimization to reduce the amount of computation. We describe in the next section a dynamic programming algorithm to minimize (4.23).

4.4.1.2 Dynamic Programming Algorithm

When dynamic programming can be applied to a problem, computation is reduced by breaking a large problem into several successive smaller problems where each smaller problem can be solved with less computation, and the accumulated computation is still less than an exhaustive search[43]. By a closer examination of the structure of (4.23), minimizing $C(l_{t1}, l_{t2}, \ldots, l_{tM} \mid K_t)$ can be partitioned into smaller problems by eliminating variables one at a time. The amount of reduction depends on the order in which variables are eliminated. We begin the optimization using the natural raster-scan order. This ordering with a slight modification at the bottom of the image frame turns out to be the most efficient ordering that we could find. We use the $QCIF$ resolution frame in figure 4.5 where $H = 9$ and $W = 11$ to describe our algorithm.

We begin by finding the optimal mode selection l_{t1}^* as a function of the remaining mode selection variables. Keeping in mind that the macroblock cost functions actually consist of two terms $R_{tj}(\cdot)$ and $\lambda_t D_{tj}(\cdot)$, we group all of the terms in (4.23) that are directly influenced

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
12	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
23	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
34	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*
45	*46*	*47*	*48*	*49*	*50*	*51*	*52*	*53*	*54*	*55*
56	*57*	*58*	*59*	*60*	*61*	*62*	*63*	*64*	*65*	*66*
67	*68*	*69*	*70*	*71*	*72*	*73*	*74*	*75*	*76*	*77*
78	*79*	*80*	*81*	*82*	*83*	*84*	*85*	*86*	*87*	*88*
89	*90*	*91*	*92*	*93*	*94*	*95*	*96*	*97*	*98*	*99*

Figure 4.5: QCIF resolution frame t

by l_{t1} and form the following function.

$$H_{l_{t1}}(l_{t1}, l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23} \mid K_t) = C_{t1}(l_{t1}, l_{t2}, l_{t12} \mid K_t) + C_{t2}(l_{t1}, l_{t2}, l_{t3}, l_{t13} \mid K_t) + \lambda_t D_{t12}(l_{t1}, l_{t12}, l_{t13}, l_{t23} \mid K_t) \quad (4.24)$$

Only the distortion term of $C_{t12}(l_{t1}, l_{t11}, l_{t12}, l_{t13}, l_{t23} \mid K_t)$ is influenced by l_{t1}. Since adding the rate term here will only increase the number of variables that influence $H_{l_{t1}}(\cdot)$, we will add the rate term at a later point. The amount of computation for each stage in the algorithm is exponentially proportional to the number of variables that influence the function at that stage. To eliminate l_{t1} from (4.24), we have to determine the optimal l^*_{t1} as a function of only five variables (l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23}). We have to calculate 2^6 functions and perform 2^5 comparisons to construct the following optimized function.

$$H^*_{l_{t1}}(l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23} \mid K_t) = \min_{l_{t1}} H_{l_{t1}}(l_{t1}, l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23} \mid K_t) \quad (4.25)$$

Provided that a table is created to store the optimal value l^*_{t1} as a function of the 32 possible values for (l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23}), we can completely eliminate l_{t1} from the problem by replacing in (4.23) the three terms directly influenced by l_{t1} with $H^*_{l_{t1}}(\cdot)$ to create a new cost function [2] $C^{(1)}(l_{t2}, l_{t3}, \ldots, l_{t99} \mid K_t)$ that depends on only $99 - 1$ variables. This new function can now be optimized, and the resulting optimal values (l^*_{t2},l^*_{t3}, l^*_{t12}, l^*_{t13}, l^*_{t23}) will determine l^*_{t1}

[2] We use $C^{(j)}$ to denote the new total cost function just after the variable l_{tj} is eliminated.

from the stored table.

To minimize $C^{(1)}(l_{t2}, l_{t3}, \ldots, l_{99} \mid K_t)$, the problem reduction is continued in the second stage by eliminating l_{t2}. We group all of the terms in the new cost function that are directly influenced by l_{t2} and calculate the following optimized function.

$$H^*_{l_{t2}}(l_{t3}, l_{t4}, l_{t12}, l_{t13}, l_{t14}, l_{t23}, l_{t24} \mid K_t) = \min_{l_{t2}} \left[H^*_{l_{t1}}(l_{t2}, l_{t3}, l_{t12}, l_{t13}, l_{t23} \mid K_t) + C_{t3}(l_{t2}, l_{t3}, l_{t4}, l_{t14} \mid K_t) + C_{t13}(l_{t2}, l_{t12}, l_{t13}, l_{t14}, l_{t24} \mid K_t)\right] \quad (4.26)$$

We determine the optimal l^*_{t2} which is stored in a table as a function of seven variables. To compute (4.26), we have to calculate 2^8 functions and perform 2^7 comparisons. The whole problem is getting smaller as we eliminate more variables, but the partitioned problem at each stage is getting larger. In this stage, it is because we only eliminate a single variable l_{t2} in (4.26) but add three more (l_{t4}, l_{t14} , and l_{t24}) from the two additional cost functions that are influenced by l_{t2} but are not influenced by l_{t1}.

The increase in partitioned problem size stops when we get to the elimination of l_{t11}. From this point on, the problem size is in steady state since only a single variable is added when one is eliminated. We write out the formula for the optimized function $H^*_{l_{t11}}(\cdot)$ below.

$$H^*_{l_{t11}}(l_{t12}, l_{t13}, l_{t14}, \ldots, l_{t32}, l_{t33} \mid K_t) = \min_{l_{t11}} \left[H^*_{l_{t10}}(l_{t11}, l_{t12}, l_{t13}, \ldots, l_{t31}, l_{t32} \mid K_t) + C_{22}(l_{t11}, l_{t21}, l_{t22}, l_{t33} \mid K_t) + R_{t12}(l_{t11}, l_{t12} \mid K_t)\right] \quad (4.27)$$

It is at this stage that we add the rate term for the cost function C_{t12}, since it is a function of l_{t11}. The optimized function in (4.27) when l_{t11} has been eliminated is a function of the 22 immediately following variables indicated by the shaded region in figure 4.5. As the optimization continues with the elimination of l_{t12} the shaded region shifts by one macroblock in raster-scan order. From this point on, the optimization function for l_{tj} is always a function of the 22 immediately following variables. Therefore, the largest number of comparisons that have to be performed at any stage with our algorithm is 2^{22} for this example. This bound on computation is a function of the image size and is equal to $2^{(2 \cdot W)}$ for this elimination order.

Elimination could continue in raster-scan order until l_{t77} is eliminated, and we are left with the final cost function $C^{(77)}(l_{t78}, l_{t79}, \ldots, l_{t99} \mid K_t) = H^*_{l_{t77}}(l_{t78}, l_{t79}, \ldots, l_{t98}, l_{t99} \mid K_t)$ which can be minimized efficiently by an exhaustive search. However, we can get even more reduction in computation if we only continue in this order until l_{t66} is eliminated

and $H^*_{l_{t66}}(l_{t67}, l_{68}, \ldots, l_{t87}, l_{88} \mid K_t)$ is constructed. We then exploit the smaller dependency for macroblocks along the bottom edge to reduce computation. This is done by next eliminating from $C^{(66)}(l_{t67}, l_{t68}, \ldots, l_{t99} \mid K_t)$ the variables for the last row of macroblocks in reverse raster-scan order until we eliminate l_{89} and obtain the optimized function $H^*_{l_{t89}}(l_{t88}, l_{t87} \ldots, l_{t68}, l_{t67} \mid K_t)$. The dependency of the optimized functions $H^*_{l_{tj}}$ builds up from 5 to 22 variables as it did for the macroblocks along the top edged as we go from eliminating l_{99} to l_{89}. An exhaustive search is then performed over the sum of the two remaining optimized functions $H^*_{l_{t66}}(\cdot)$ and $H^*_{l_{t89}}(\cdot)$ to get the 22 remaining optimal variables $(l^*_{t67}, l^*_{t68}, \ldots, l^*_{t87}, l^*_{t88})$. These optimal values are then used to successively read off the optimal choices for the previously eliminated variables l_{tj} from stored tables.

To minimize (4.23), the total number of comparisons that have to be performed with this algorithm on a QCIF resolution frame is $257,250,623$. This is feasible and is a substantial savings in computation over the $2^{99} - 1$ comparisons that would have to be done with an exhaustive search over the original cost function. However, we usually have to perform several iterations of this algorithm at a frame for a given value of K_t to find the correct value of λ_t to meet the rate constraint. The optimal λ_{t-1} from previous frames provides a useful starting point.

4.4.2 Joint Optimization of Modes and K_t

An exhaustive search must be done over K_t to minimize $C(K_t, l^*_{t1}, l^*_{t2}, \ldots, l^*_{tM})$ for an optimal solution. We now show how we choose K_t for a local optimal solution which we believe to be close to the optimal solution from empirical evidence. Since we have a finite number of choices for quantization parameters, this problem is inherently discrete and could potentially be solved by an exhaustive search. However, the range of values of K_t which lead to a distinct set of macroblock quantization parameters for frame t is large. Since these values for K_t are unknown, we estimate a uniform step-size over which to search K_t. As a result, a local optimal solution can only be guaranteed within the chosen step-size value Δ_{K_t}. This parameter can be chosen to tradeoff accuracy for less computation.

Since there is a rate constraint, there is limit on the best frame quality that can be achieved when no packet loss occurs. To determine this K_t^{min} value for a given R_c, we first choose a sufficiently low value for K_t and find the optimal mode selection with λ_t set to

zero. In this case, the distortion function is disregarded, and the resulting rate is the lowest possible rate that frame t can be coded for this value of K_t. If this rate is greater than R_c, then K_t is raised by Δ_{K_t}. This process is continued, until the lowest possible rate is less than R_c. For this value of K_t, the value of λ_t to find the minimum MSE ($MMSE$) mode selection to meet the rate constraint is determined, and expected distortion is computed. Typically at K_t^{min}, the minimum total MSE at R_c bits is dominated by distortion from potential packet loss and concealment. As K_t is increased, less distortion from packet loss and more distortion from quantization results. Initially, this usually results in a decrease of the total expected MSE until the optimal point K_t^* is reached. After this point the MSE begins to be dominated by quantization noise and the MSE begins to increase until K_t^{max} is reached where all of the quantization step-sizes go to their maximum values.

For any given value of K_t, we can find the optimal mode selection to minimize the expected MSE using the algorithm in section 4.4.1 for a given rate constraint. We denote the resulting optimal solution as a function of K_t below.

$$F(K_t) = E^* \left[\|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2 \mid K_t \right] \tag{4.28}$$

We argue this function might be unimodal and that a local optimal solution is a global one. This new function to minimize over K_t can be decomposed into three terms. We can write one of the terms as an explicit function of K_t. We can compute the other two terms for any particular example, but we can only offer some intuition which is supported by empirical evidence about how these terms vary as a function of K_t in general. After describing what we know about $F(K_t)$, we will show some empirical results that support our argument of unimodality.

The difference between the original and concealed frame can be separated into the sum of the difference due to source coding error and the difference from channel coding error as shown below.

$$\|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2 = \|(\boldsymbol{X_t} - \boldsymbol{Y_t}) + (\boldsymbol{Y_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}})\|^2 \tag{4.29}$$

From this equation we can decompose $F(K_t)$ into the following three terms.

$$F(K_t) = \|\boldsymbol{X_t} - \boldsymbol{Y_t}\|^2 + E^* \left[\|\boldsymbol{Y_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2 \mid K_t \right] + \sum_{j=1}^{M} \sum_{i=1}^{Z} (X_{tji} - Y_{tji}) \cdot E^* \left[(Y_{tji} - \hat{Y}_{tji}) \mid K_t \right] \tag{4.30}$$

The first term in (4.30) is the quantization error from source coding. The expected value does not have to be calculated, since it does not depend on any results from the probabilistic channel model. This term increases linearly with K_t and can be calculated directly as shown below.

$$\|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2 = \sum_{j=1}^{M} \alpha_{tj} \cdot K_t = A \cdot K_t \tag{4.31}$$

The second error term in (4.30) is from loss that occurs in the channel. It will be large when the channel loss rate is high. As a function of K_t, it will generally be large when K_t is small since mostly inter-mode coding must be done in order to meet the rate constraint. It will decrease as K_t increases since more bits can be allocated to code more macroblocks in intra mode. The third term is the interaction between the quantization error and expected channel error. We can not say much about this term other than we expect it to be dominated at low (high) values of K_t by the channel (quantization) error term in (4.30).

We plot $F(K_t) = E^* \left[\|\boldsymbol{X_t} - \hat{\boldsymbol{Y}}_{\boldsymbol{t}}\|^2 \mid K_t \right]$ in Figs. 4.6–4.8 for different frames and loss rates. These examples shown are from two consecutive frames of the *Carphone* sequence coded at 400 *kbps* ($R_c = 13,333$ bits per frame). The values for K_t^{min} in each example is indicated by the dashed line in each plot. The value for Δ_{K_t} is set equal to 1000. These examples support our hypothesis of unimodality. When the loss probability is $p = 0.1$, $F(K_t)$ decreases initially after K_t is increased from its minimum value as shown in Fig. 4.6. This is from a decrease in the expected channel loss distortion from an increase in intra coded macroblocks. It then increases approximately linearly from the quantization error after the minimum $F(K_t)$ is reached. We indicate a source for potential error by an x in the plot for frame 2 in Fig. 4.6. This point will cause an incorrect value to be chosen for the optimal K_t if not properly accounted for. This false increasing point is from a sudden drop in achieved coded rate at this value of K_t. Since only discrete values for the rates can be achieved, the coded rate might not always be very close to R_c for all K_t values. When the resulting rate is significantly lower than the rate achieved for the previous K_t value as it is in this case, the corresponding distortion will increase resulting in a false minimum detection. This can be viewed as a noisy measurement of $F(K_t)$. False minimum detection can easily be avoided by checking the achieved rate as well as verifying an initial increase in $F(K_t)$ by computing a few more points after the initial detected increase.

As the loss rate p is increased, there is more contribution from likely channel loss in the

expected distortion function. This means that $F(K_t)$ is dominated more by the second term in (4.30) which is a decreasing function of K_t. The optimal value for K_t increases to tradeoff an increase in quantization distortion for a decrease in expected channel loss distortion resulting in an overall decrease in distortion. This can be seen in the plots in Fig. 4.7 and Fig. 4.8 where the point where the linear quantization error term in (4.30) begins to dominate at progressively higher values for K_t. We stated before that it is intuitively better to have more loss from quantization at the source than to have unintended loss in the channel, and using the total expected MSE as the distortion to minimize agrees with this intuition. Given these examples, it appears satisfactory to only determine a local optimum for K_t after finding K_t^{min}.

4.5 Simulation Results

To compare our algorithm with traditional methods for coding mode and quantization step-size selection, we obtain results from experiments using four different coding mode selection algorithms.

1. The first coder uses the algorithm described in section 2.1.3 to code the macroblock at the lowest bit rate (LBR) for any chosen frame quality constant K_t. We refer to this algorithm as the LBR algorithm. Here, the underlying mode selection algorithm codes a macroblock at a selected frame MSE distortion K_t in both intra and inter mode and chooses the mode that results in the lowest bit rate. The quantization step-size is found for each macroblock to achieve K_t. With the assumption of monotonicity of the rate as a function of K_t, a bisection algorithm is used to find the best K_t for each frame to achieve the rate constraint R_c bits. Nothing is transmitted for inter-coded macroblocks with zero motion vector and residual completely quantized to zero. The macroblock in the same spatial location in the previous frame is repeated at the decoder for these macroblocks. We denote this process as implicit transmission detection.

2. The second coder is described in section 3.3.2 and uses the LBR algorithm for mode selection with the addition of forced updates for robustness to loss. We set the update threshold $U = 5$, so that a macroblock at any spatial location is forced to be coded in intra-mode at least once every 5 times it is transmitted. Implicit transmission detection is used to avoid sending empty macroblocks.

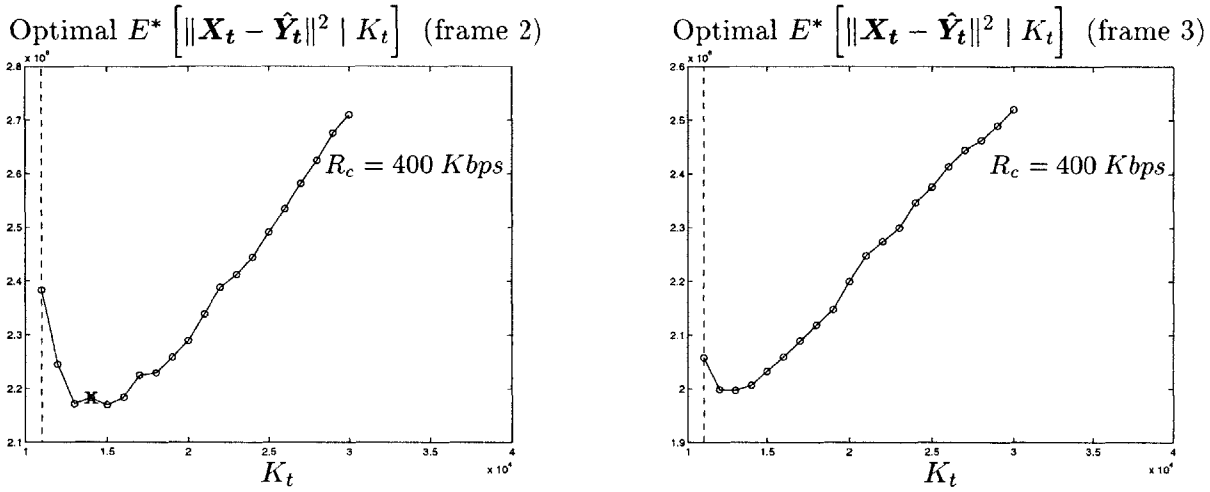


Figure 4.6: Channel loss rate $p = 0.1$

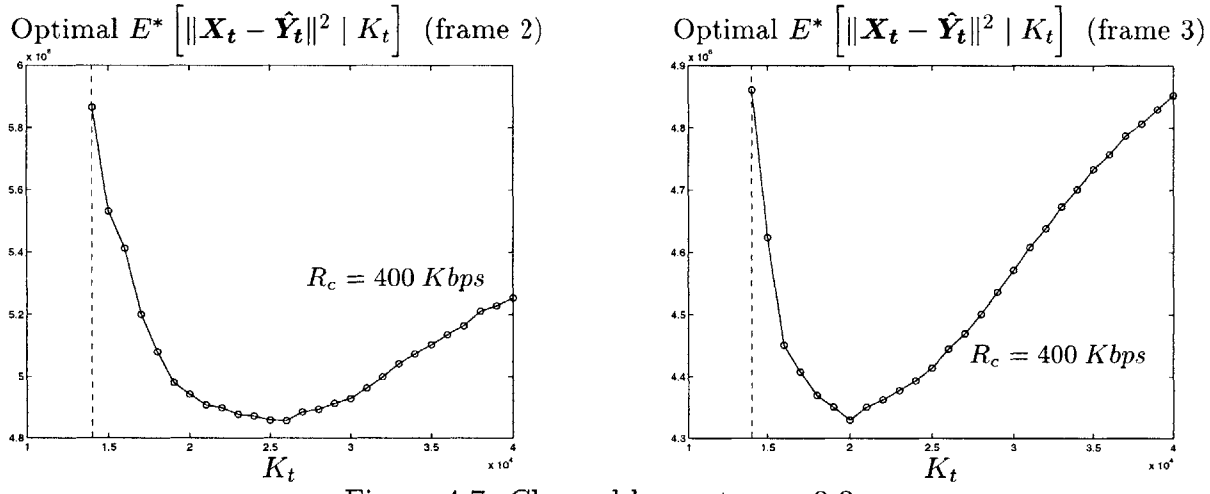


Figure 4.7: Channel loss rate $p = 0.3$

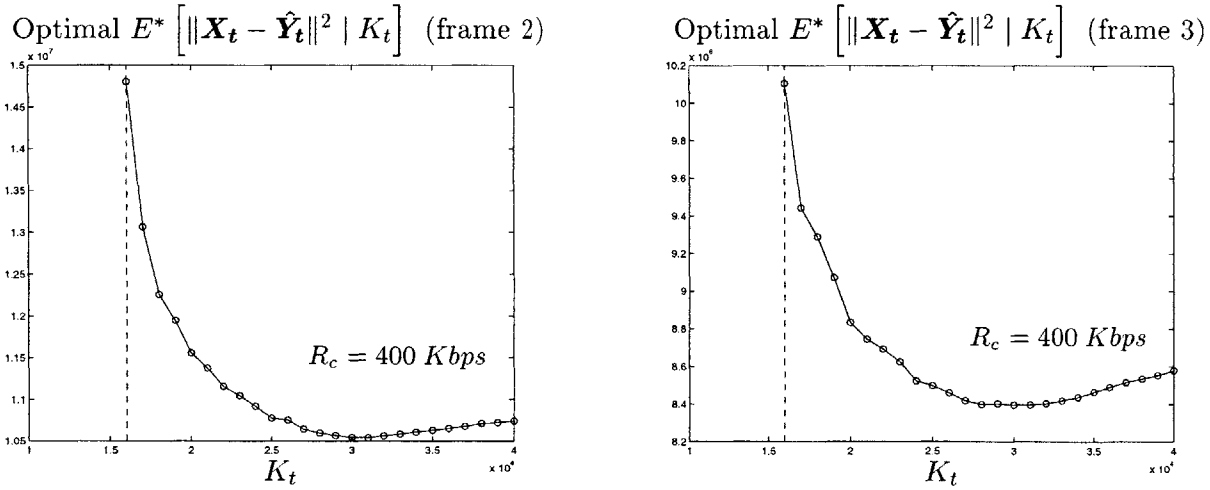


Figure 4.8: Channel loss rate $p = 0.5$

3. The third coder uses conditional replenishment described in section 3.3.1 and codes all transmitted macroblocks in intra mode only. A threshold T is set to determine which macroblocks have changed significantly because of motion and need to be transmitted. A macroblock is transmitted, if the sum of absolute differences between the original and decoded macroblock at the encoder at the same location in frame t and $t-1$ is greater than the threshold. The value for T is set to optimally tradeoff is between quantization distortion from sending too many intra-coded macroblocks and error from not detecting and not sending macroblocks which significantly changed from the previous frame. Several experiments were run to choose the value for T that results in the minimum MSE between the original and decoded video when no loss is considered. This value is typically $T = 6$ or 7 for the sequences in our experiments. Again, a bisection algorithm is used to find the best value of K_t to meet the bit rate constraint R_c.

4. The final experiment is the algorithm presented in this chapter to minimize the total expected MSE. This algorithm uses implicit transmission detection to avoid sending empty macroblocks.

4.5.1 *Carphone* results

For the first experiment we code 30 $QCIF$ resolution frames of the *Carphone* sequence using an $H.261$ encoder and simulate packet loss in an erasure channel. We run simulations using the four parameter selection algorithms mentioned above for three different loss rates($p = 0.1$, $p = 0.3$, and $p = 0.5$). Each macroblock is lost independently. The results of decoded/concealed frames from our three simulations are shown in Figs. 4.9– 4.13. All simulations use the same concealment method described in section 3.2. Figs. 4.9 and 4.10 show the results for all four coding algorithms for $p = 0.1$ for two successive frames in the sequence The results in (a) which use the LBR algorithm suffers from more channel loss distortion than the other three methods as can be seen from the relatively low $PSNR = 23.6399$. Error from channel loss can even be seen in (b) for both frames when forced intra updates are used. However, because inter mode coding is allowed for more efficient coding there is an improvement in resolution over (c) which uses conditional replenishment. The result in (d) from the coder to minimize the total MSE provides the best balance of the four coding algorithms among the distortion from resolution loss, blocking artifacts, and channel loss effects.

(a) (b)

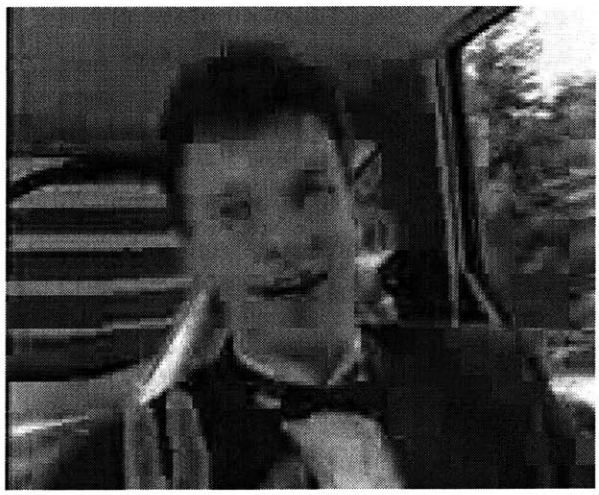
(c) (d)

Figure 4.9: Frame 14 of *Carphone* sequence coded at 256 kbps with loss rate $p = 0.1$. (a) *LBR* algorithm, PSNR=23.6399. (b) Forced Update (U=5), PSNR=25.7785. (c) Conditional Replenishment (T=6), PSNR=26.6008. (d) MMSE solution, PSNR=27.1898.

(a)

(b)

(c)



(d)

Figure 4.10: Frame 15 of *Carphone* sequence coded at 256 kbps with loss rate $p = 0.1$. (a) *LBR* algorithm, PSNR=24.1045. (b) Forced Update (U=5), PSNR=25.4758. (c) Conditional Replenishment (T=6), PSNR=27.5989. (d) MMSE solution, PSNR=27.6951.

The resulting mode selections for the four different coding algorithms are shown in Fig. 4.11 for 30 frames. For each of the 99 macroblock locations in a frame, one of the following three possibilities is shown in the plot. A black rectangular dash mark is printed when the macroblock is coded in intra mode. A gray rectangular dash mark is printed when a macroblock is not transmitted from conditional replenishment or implicit transmission detection. The space is left blank when a macroblock is inter coded and transmitted. We code all macroblocks in the first two frames in intra mode. This is an attempt to get the problem into steady-state where with very high probability all macroblock locations have some decoded/concealed block which will be used for concealment in future frames. The first plot in Fig. 4.11 (a) is for the LBR mode selection algorithm. As expected many macroblocks are coded in inter mode. The second plot in (b) shows the mode selections for the forced intra update algorithm. A quasi-periodic structure resulting from the forced intra mode updates can be seen. However, there are still several inter-coded macroblocks in between the forced update intervals. The third plot in (c) shows the mode selection for conditional replenishment. Because the motion in this sequence tends to be at the same spatial location in a frame, there are some macroblock locations that are always sent (coded in intra mode) while there are other regions whose macroblocks are rarely coded and sent. This is sufficient for this example since the duration of the sequence is short, and we attempt to ensure that all stationary macroblocks arrive in the first two frames. However, for longer sequences a forced intra-update process will have to be imposed at a low periodic rate to make sure there are no long term errors in the frame from lost macroblocks. In the mode selection to minimize the expected MSE shown in (d), you find for the same coding rate a slightly more spread out distribution of intra-coded macroblocks. For this loss rate it is not always necessary to code every macroblock at a location in intra. The mode selection for the other algorithms are done independent of the loss rate. The advantage of the $MMSE$ is that it takes into account the loss rate and will adapt its parameter selection appropriately.

Fig. 4.12 shows the results for a frame for all four coding algorithms when $p = 0.3$. A noticeable increase from channel loss and poor concealment can be seen in the frames in (a), (b), and (c) for the first three algorithms. However, the result from the $MMSE$ method in (d) only shows a slight increase in blockiness compared to Fig. 4.10 (d). This is from an increase in the number of intra coded macroblock to avoid high distortion from channel loss and error propagation. Fig. 4.13 shows the results for a frame when $p = 0.5$. The examples coded with the the LBR And LBR with forced update algorithms really fall apart at this loss rate. Again you can see more distortion from loss for the conditional replenishment

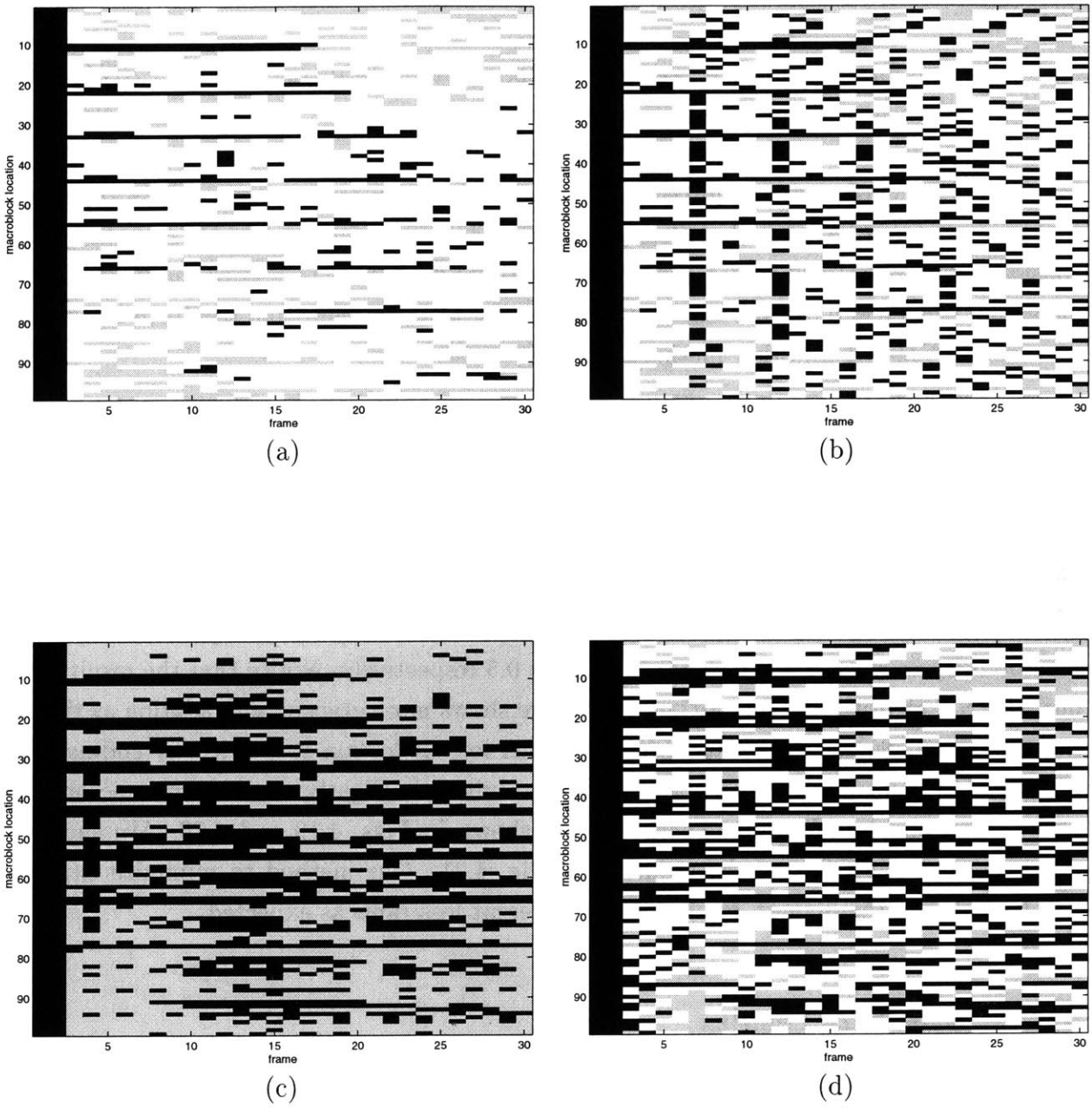


Figure 4.11: Mode selection plots for 30 frames of *Carphone* sequence coded at 256 kbps. (a) *LBR* algorithm. (b) Forced Update (U=5). (c) Conditional Replenishment (T=6). (d) MMSE solution ($p = 0.1$).

than the $MMSE$ case. This is true on average for this loss rate. The mode selection which changes as a function of the loss rate for the $MMSE$ algorithm is shown in Fig. 4.14 for $p = .3$ and $p = .5$. As expected, more macroblocks are coded in intra mode as the loss rate increases.

Finally, for this example, we show a plot in Fig. 4.15 of the average $PSNR$ of the sequence for each of the four coding algorithms at different loss rates. The $PSNR$ drops dramatically for the LBR algorithm which is the least robust. We only plot points for conditional replenishment and forced intra update algorithms for $p > 0$ since these algorithms are designed for situations where loss can occur. However, since the $MMSE$ coder always considers the loss rate when determining the optimal decision it resorts to the most efficient algorithm when $p = 0$. As the loss rate increases the $MMSE$ algorithm results in the highest average $PSNR$ over all other methods.

4.5.2 *Foreman* results

For the second experiment, we code 30 $QCIF$ [3] resolution frames of the *Foreman* sequence using an $H.261$ encoder and simulate packet loss in an erasure channel. Figs. 4.16, 4.17, and 4.18 show the results of applying the four different coding algorithms to channels with macroblock loss rates $p = 0.1$, $p = 0.3$, and $p = 0.5$ respectively. Notice that the results from our $MMSE$ algorithm in part (d) of each figure shows more graceful degradation as the loss rate increases. Severe degradation from channel loss distortion is avoided by coding more macroblocks in intra mode at higher loss rates resulting in slightly more blurry macroblocks. Mode selection plots for this experiment are shown in Figs. 4.19 and 4.20. We show the average frame $PSNR$ over the sequence in Fig. 4.21. Our method to minimize expected MSE results in the highest $PSNR$ across different loss rates as expected.

4.5.3 *Students* results

For our third experiment, we code 30 $QCIF$ resolution frames of the *Students* sequence using an $H.261$ encoder and simulate packet loss in an erasure channel. We show the results to simulations with random macroblock loss for the loss rates $p = 0.1$, $p = 0.3$, and $p = 0.5$

[3] only 20 frames are coded for the case $p = 0.5$

(a) (b)

(c) (d)

Figure 4.12: Frame 15 of *Carphone* sequence coded at 256 kbps with loss rate $p = 0.3$. (a) *LBR* algorithm, PSNR=20.5873. (b) Forced Update (U=5), PSNR=22.8145. (c) Conditional Replenishment (T=6), PSNR=24.0956. (d) MMSE solution, PSNR=25.6380.

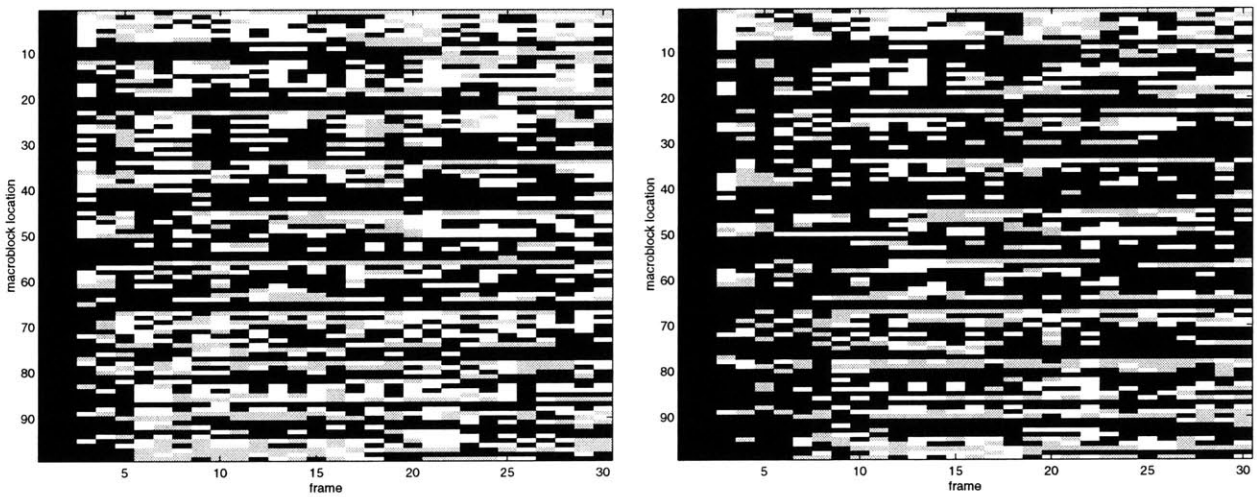
(a)

(b)

(c)

(d)

Figure 4.13: Frame 17 of *Carphone* sequence coded at 256 kbps with loss rate $p = 0.5$. (a) *LBR* algorithm, PSNR=17.0360. (b) Forced Update (U=5), PSNR=19.3745. (c) Conditional Replenishment (T=6), PSNR=22.6527. (d) MMSE solution, PSNR=23.7058.



(a) (b)

Figure 4.14: Mode selection plots for 30 frames of *Carphone* sequence coded at 256 kbps. (a) MMSE solution ($p = 0.3$). (b) MMSE solution ($p = 0.5$).

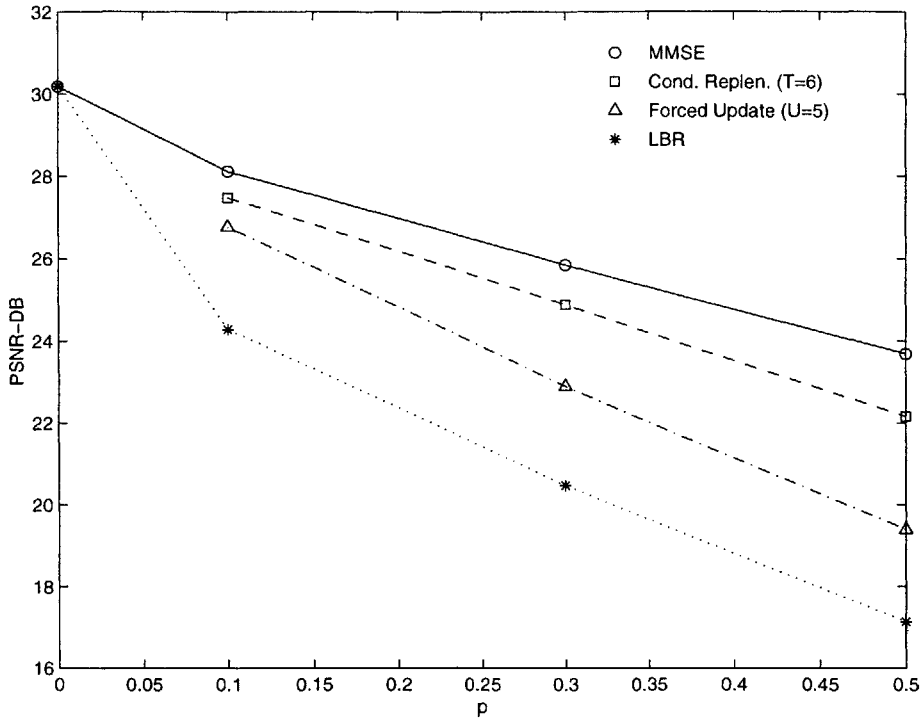


Figure 4.15: Average PSNR for *Carphone* sequence as a function of loss probability.

in Figs. 4.22, 4.23, and 4.24 respectively. The mode selection plots are shown in Figs. 4.25 and 4.26. The relative average frame $PSNR$ is shown in Fig. 4.27. Again, the $MMSE$ algorithm presented in this chapter outperforms the other algorithms across different loss rates.

4.6 Summary

In this chapter, we presented an efficient and robust algorithm to code sequences in the presence of macroblock loss. This algorithm takes both the channel loss characteristics and the decoder concealment method into consideration to select its coding parameters to minimize the total expected MSE at the decoder. Our algorithm determines how to allocate bits between intra-mode encoding for macroblocks and frame quality for best performance. As an illustrative example, perceptual improvements over traditional methods for coding parameter selection in the simple case of independent macroblock loss was presented.

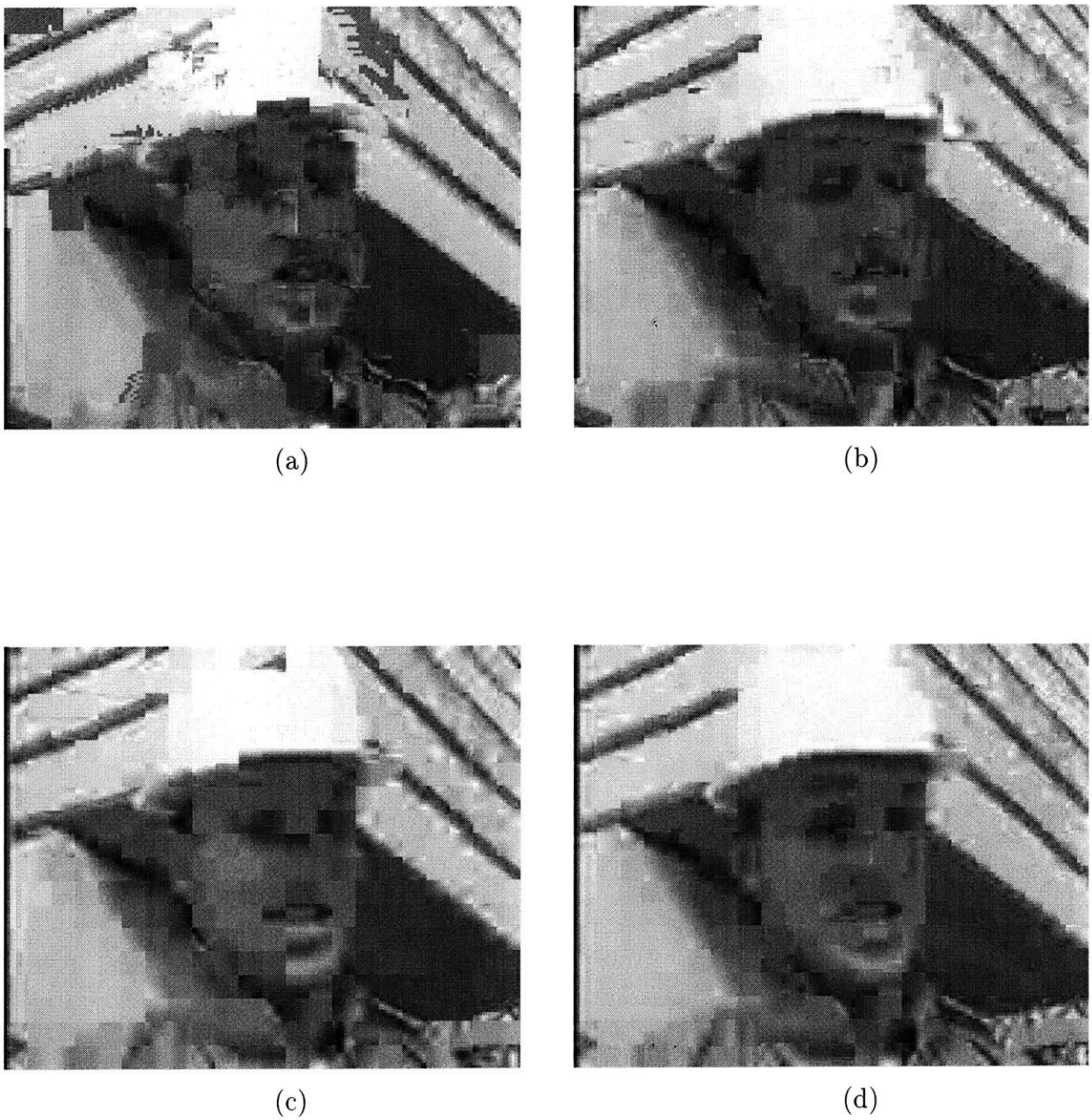
(a)

(b)

(c)

(d)

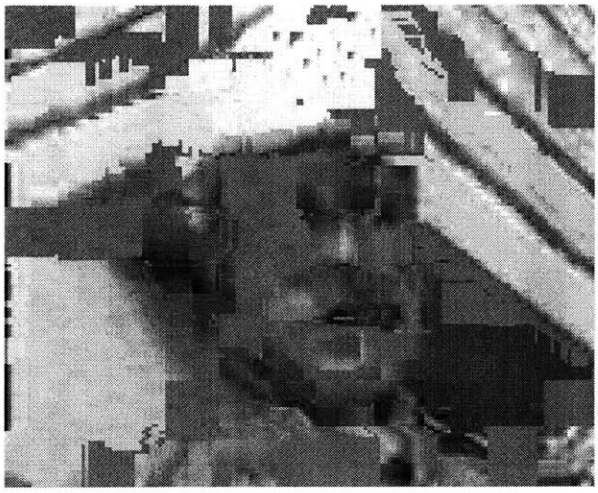
Figure 4.16: Frame 18 of *Foreman* sequence coded at 192 kbps with loss rate $p = 0.1$. (a) *LBR* algorithm, PSNR=23.2831. (b) Forced Update (U=5), PSNR=25.8632. (c) Conditional Replenishment (T=7), PSNR=25.6289. (d) MMSE solution, PSNR=27.5431.



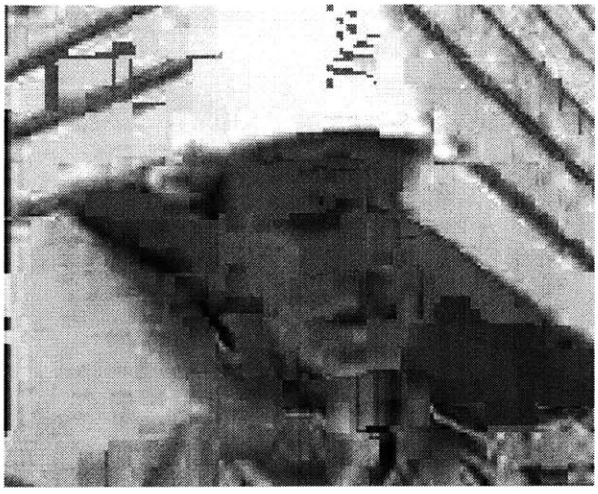
(a) (b)

(c) (d)

Figure 4.17: Frame 18 of *Foreman* sequence coded at 192 kbps with loss rate $p = 0.3$. (a) *LBR* algorithm, PSNR=19.5751. (b) Forced Update (U=5), PSNR=23.4995. (c) Conditional Replenishment (T=7), PSNR=23.9747. (d) MMSE solution, PSNR=26.3366.



(a)



(b)

(c)

(d)

Figure 4.18: Frame 17 of *Foreman* sequence coded at 192 kbps with loss rate $p = 0.5$. (a) *LBR* algorithm, PSNR=17.4392. (b) Forced Update (U=5), PSNR=20.3574. (c) Conditional Replenishment (T=7), PSNR=22.1871. (d) MMSE solution, PSNR=24.6730.

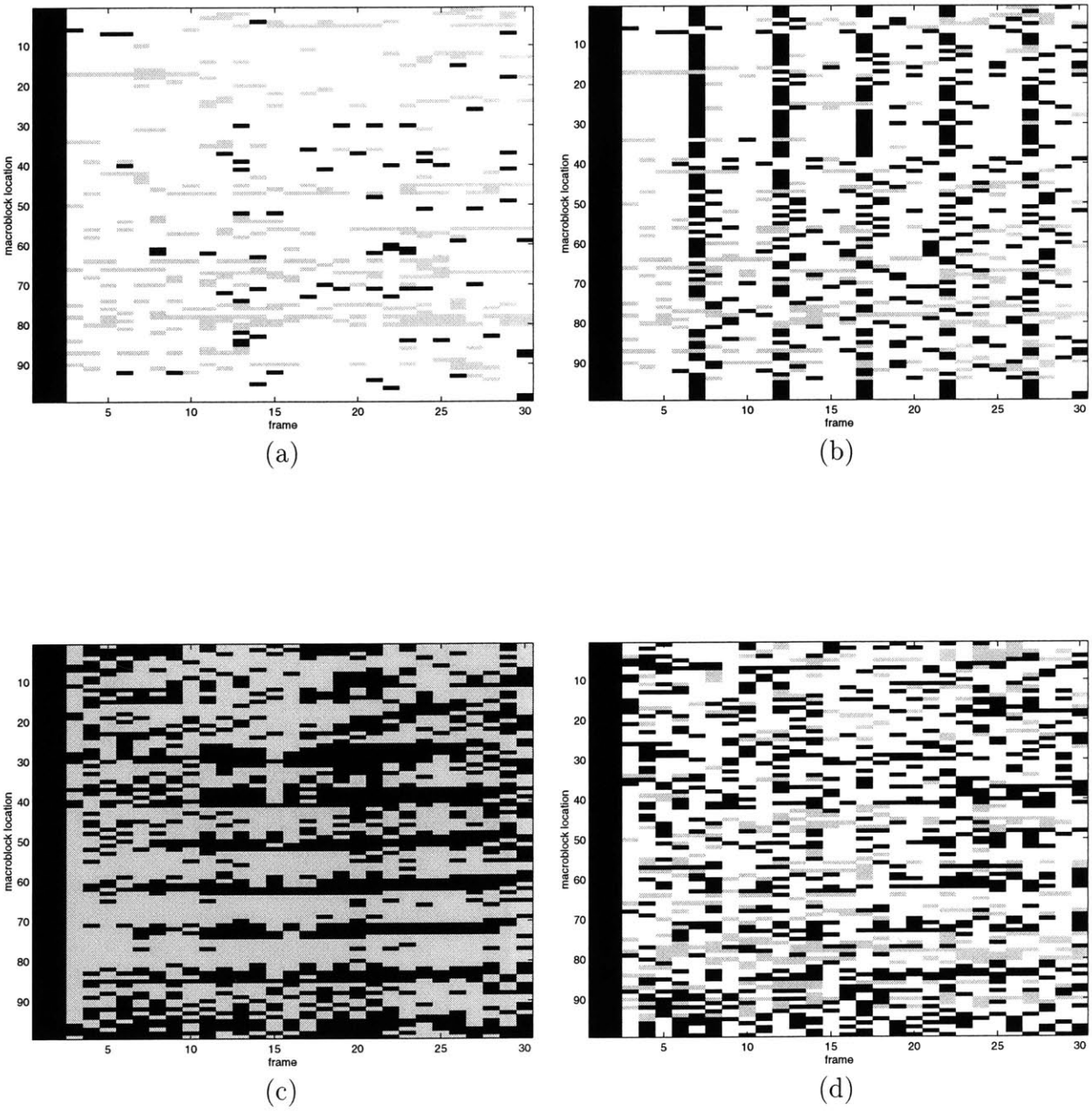


Figure 4.19: Mode selection plots for 30 frames of *Foreman* sequence coded at 256 kbps. (a) *LBR* algorithm. (b) Forced Update (U=5) (c) Conditional Replenishment (T=6). (d) MMSE solution ($p = 0.1$).

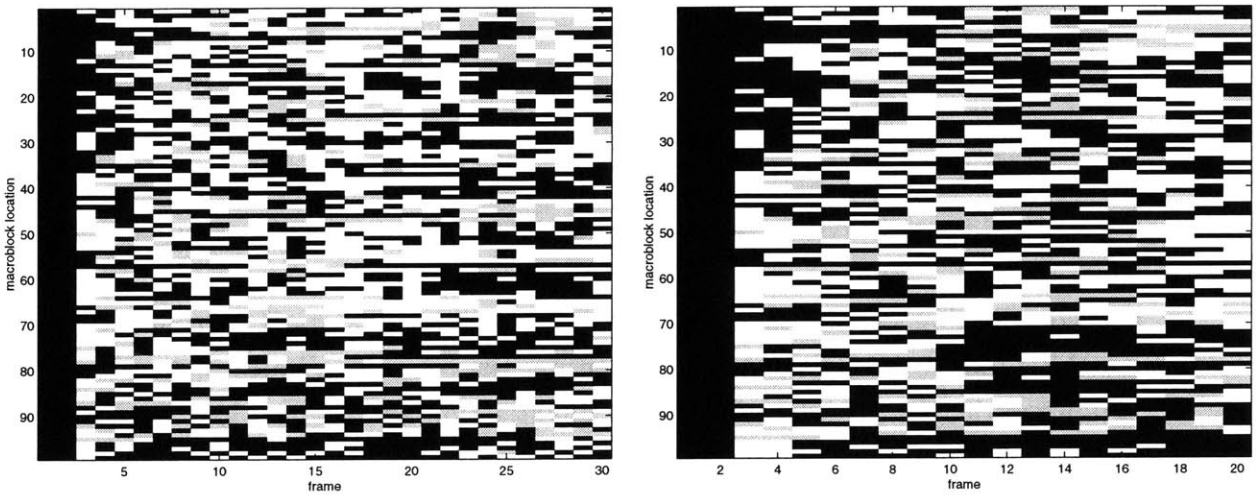


Figure 4.20: Mode selection plots for 30(20) frames of *Foreman* sequence coded at 192 kbps. (a) MMSE solution ($p = 0.3$). (b) MMSE solution ($p = 0.5$).

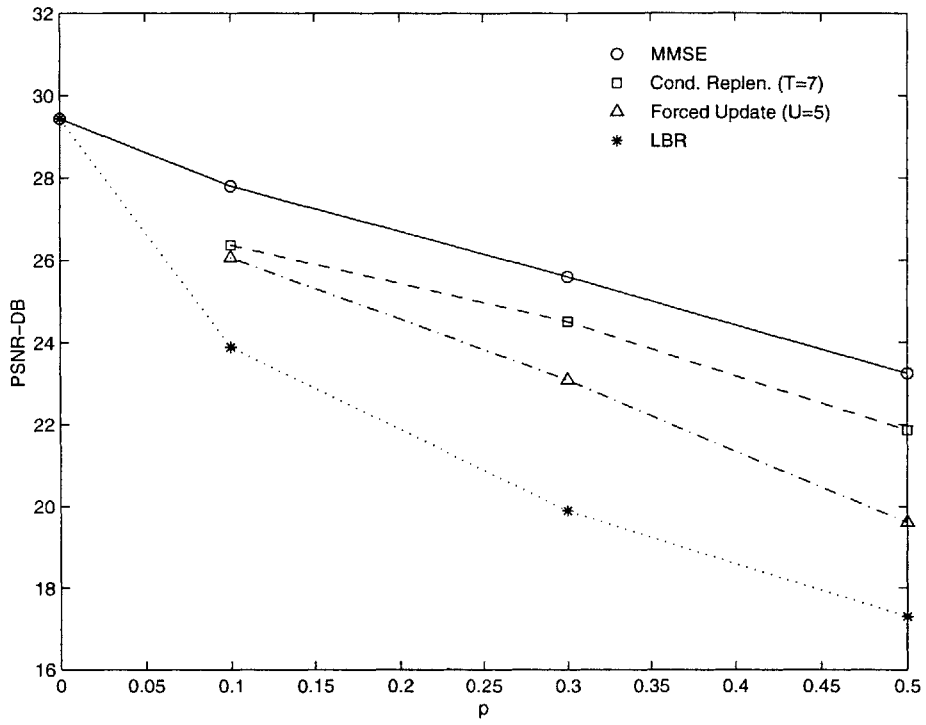


Figure 4.21: Average PSNR for *Foreman* sequence as a function of loss probability.

The contribution of this thesis is that it is the first analytical method developed to address this problem in the right way. By using the total MSE after concealment for the distortion metric, the distortion from both quantization and channel loss are effectively combined into one metric to be minimized. The optimal tradeoff between the two is found as a function of the loss rate. In this thesis, we have presented an algorithm for one particular concealment method. However, the approach is general and can be extended to work with other concealment methods. The computation for this $MMSE$ method is immense. While other heuristic methods will be developed for more practical implementation, the results obtained from this algorithm can be used to measure how good these other methods are. It would be sufficient to find a heuristic method that will result in near-optimal performance which can now be determine.

(a)

(b)

(c)

(d)

Figure 4.22: Frame 25 of *Students* sequence coded at 128 kbps with loss rate $p = 0.1$. (a) *LBR* algorithm, PSNR=23.7234. (b) Forced Update (U=5), PSNR=27.6477. (c) Conditional Replenishment (T=6), PSNR=27.4210. (d) MMSE solution, PSNR=28.3454.

(a)

(b)

(c)

(d)

Figure 4.23: Frame 25 of *Students* sequence coded at 128 kbps with loss rate $p = 0.3$. (a) *LBR* algorithm, PSNR=20.8008. (b) Forced Update (U=5), PSNR=24.8111. (c) Conditional Replenishment (T=6), PSNR=24.2679. (d) MMSE solution, PSNR=27.0668.

(a)

(b)

(c)

(d)

Figure 4.24: Frame 25 of *Students* sequence coded at 128 kbps with loss rate $p = 0.5$. (a) *LBR* algorithm, PSNR=18.8038. (b) Forced Update (U=5), PSNR=21.8308. (c) Conditional Replenishment (T=6), PSNR=22.6768. (d) MMSE solution, PSNR=26.2757.

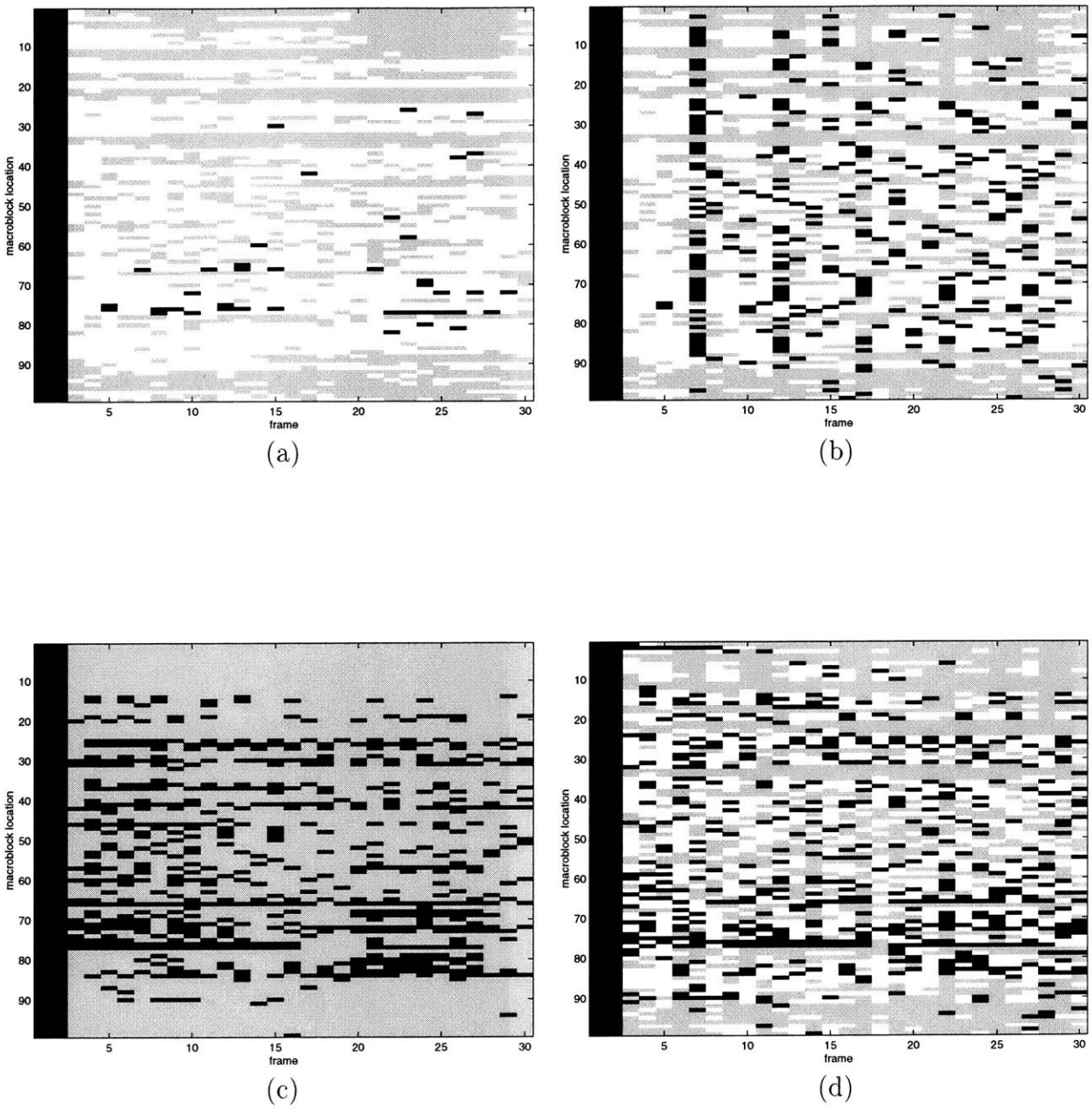


Figure 4.25: Mode selection plots for 30 frames of *Students* sequence coded at 128 kbps. (a) *LBR* algorithm. (b) Forced Update (U=5) (c) Conditional Replenishment (T=6). (d) MMSE solution ($p = 0.1$).

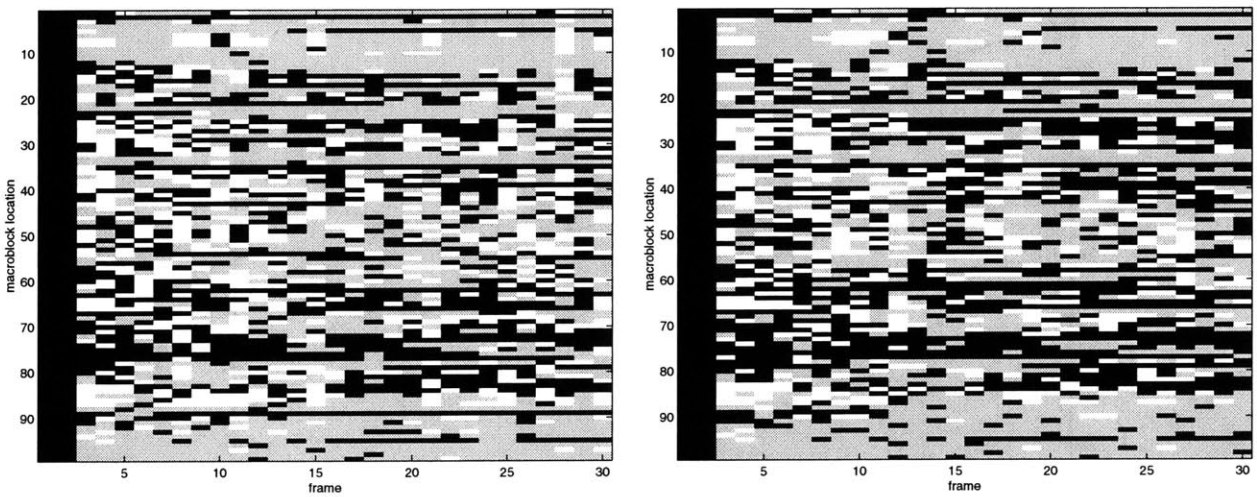


Figure 4.26: Mode selection plots std 30(20) frames of *Students* sequence coded at 128 kbps. (a) MMSE solution ($p = 0.3$). (b) MMSE solution ($p = 0.5$).

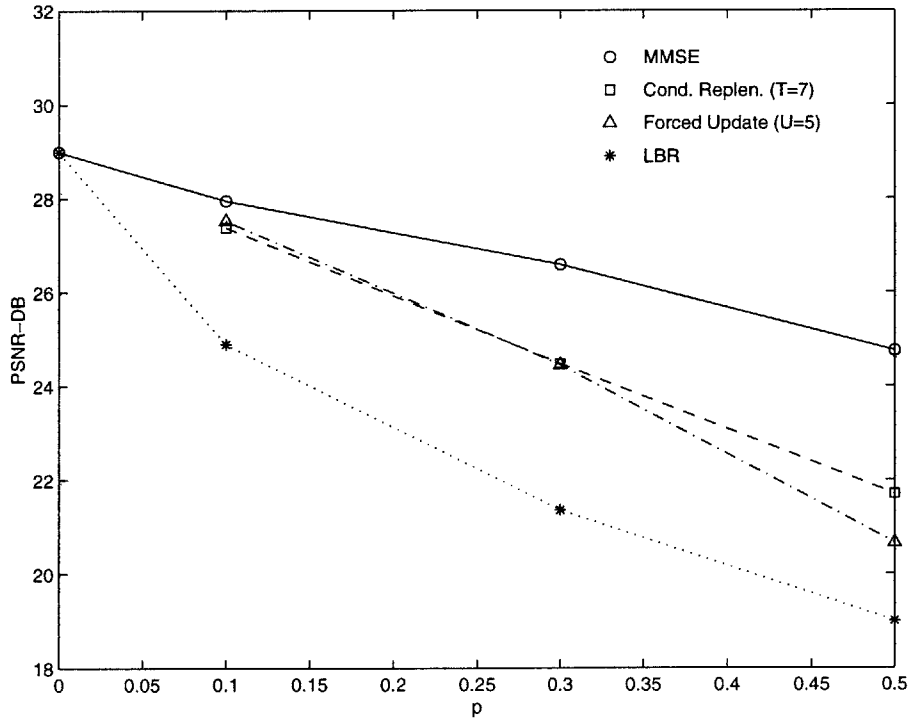


Figure 4.27: Average PSNR for *Students* sequence as a function of loss probability.

Chapter 5

Conclusion

5.1 Summary

We presented a methodology for optimally selecting the macroblock coding modes and quantization step-sizes in the presence of independent macroblock loss for a given bit rate constraint. We used the Internet as an example of a channel with unreliable data transmission. Since the current Internet does not offer any quality of service (QOS) guarantees, it is incapable of transmitting real-time data without the potential for channel loss. We used a simple yet relevant model for channel loss to make optimal coding decisions at the encoder. We argued that macroblock interleaving can be used in cases of bursty loss to randomize the loss among neighboring macroblocks.

First, we presented an algorithm for optimal parameter selection when no channel loss is expected. This algorithm results in best frame quality for a given bit constraint by choosing the most efficient coding modes. We showed in our experiments that this algorithm is not appropriate for coding over lossy channels. Since many macroblocks are coded using prediction, the distortion increased dramatically when random macroblock loss was simulated.

We next discussed traditional robust methods developed for the transmission of real-time data over packet-switched networks, and presented their potential drawbacks. These methods can be divided into two broad categories. The first set of approaches attempted to develop networking protocols capable of providing the necessary isolation for guaranteed reliable transmission while allowing for the sharing of resources for the desired network efficiency. The second set of approaches acknowledged that designing such a network is difficult at best and robust methods should be included in the end systems to handle the expected packet loss. However, these methods developed were ad hoc.

We presented an algorithm to choose the mode selection for the macroblocks in a frame to minimize the total MSE after concealment. We showed that the solution to this analytical formulation provided an effective way to allocate bits between using finer quantization step-sizes for the macroblock transform coefficients and increasing the number of intra coded macroblocks for robustness to loss. We showed the results to our algorithm resulted in improved performance over traditional ad hoc methods for mode selection. In addition, the parameter selection algorithm is a function of the loss probability in the network. As a result, it is adaptive to the variable conditions in the channel.

The contribution of this thesis is that it presents the first real analytical solution for mode selection in the presence of loss. By considering the concealment method to be used when selecting macroblock coding modes at the encoder, the inherent redundancy in video is optimally used to minimize the expected MSE. We presented the results for one particular temporal concealment method, but the algorithm is general and can be extended to other common concealment methods. The computational complexity for this algorithm is large. As a result, it can not be currently used for practical coding applications. However, it can be used to determine when a sufficient suboptimal algorithm has been found. Any approach that results in performance close to the optimal solution is good enough.

5.2 Future Research

In this thesis, we addressed the problem of bit allocation between intra mode encoding for robustness to loss and finer transform coefficient quantization for better frame quality. This represents one possible tradeoff in bit allocation. Bits can be allocated elsewhere for improvements in robustness to loss. Other examples include simple replication of transmitted data or even more sophisticated forward error correction (FEC). A more generalized version of this problem could be formed to find the best way to allocate bits among several methods for robust coding. Since the problem solved in this thesis is a subset of this more general approach, the results from the solution to this new problem will not be worse than the results presented in this thesis. In addition to a more general formulation, a better metric for distortion other than MSE might be found to solve an analytical problem for mode selection.

Another direction for future research would be to solve the problem using a more

realistic correlated loss model for the channel. Interleaving macroblocks is more likely to result in a loss of coding efficiency since the correlation among neighboring macroblock coding parameters can no longer be exploited for more compression. This loss in efficiency should be studied to determine how worthwhile it is to use interleaving.

Finally, the approach presented in this thesis took the channel loss as a given and developed a mode selection algorithm that is robust to the expected loss. The principles learned by developing such methods might be used to help guide the development of future network protocols. Since robust methods are better capable of handling loss the network protocol requirements are reduced. As robust coding methods are developed and studied, the appropriate network protocols can be designed to result in improved video quality at the receiver for video transmitted over networks.

Bibliography

[1] T. Wiegand, M. Lightstone, *et al.*, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging h.263 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 182–190, April 1996.

[2] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and mpeg video coders," *IEEE Transactions on Image Processing*, vol. 3, pp. 533–545, September 1994.

[3] J. Lee and B. W. Dickinson, "Rate-distortion optimized frame type selection for mpeg encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 501–510, June 1997.

[4] H. Sun, W. Kwok, M. Chien, and C. H. J. Ju, "Mpeg coding performance improvement by jointly optimizing coding mode decisions and rate control," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 449–458, June 1997.

[5] C.-Y. Hsu, A. Ortega, and A. R. Reibman, "Joint selection of source and channel rate for vbr video transmission under atm policing constraints," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1016–1028, August 1997.

[6] Q.-F. Zhu, Y. Wang, and L. Shaw, "Coding and cell-loss recovery in dct-based packet video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 248–258, June 1993.

[7] W.-J. Chu and J.-J. Leou, "Detection and concealment of transmission errors in h.261 images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 74–84, February 1998.

[8] M. Ghanbari and V. Seferidis, "Cell-loss concealment in atm video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 238–246, June 1993.

[9] S. S. Hemami and T. H.-Y. Meng, "Transform coded image reconstruction exploiting interblock correlation," *IEEE Transactions on Image Processing*, vol. 4, pp. 1023–1027, July 1995.

[10] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projections onto convex sets," *IEEE Transactions on Image Processing*, vol. 4, pp. 470–477, April 1995.

[11] J. Feng, K.-T. Lo, and H. Mehrpour, "Error concealment for mpeg video transmissions," *IEEE Transactions on Consumer Electronics*, vol. 43, pp. 183–187, May 1997.

[12] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 983–1001, August 1997.

[13] T. Turletti and C. Huitema, "Videoconferencing on the internet," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 340–350, June 1996.

[14] *ISO/IEC 11172-2:1993 Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 2: Video.*

[15] *ISO/IEC DIS 13818-2 Information technology - Generic coding of moving pictures and associated audio information: Video.*

[16] *Recommendation H.261: Video Codec for Audiovisual Services at px64 kbits.*, March 1993. ITU-T (CCITT).

[17] *Recommendation H.263: Video Coding for Low Bitrate Communication.*, February 1998. ITU-T (CCITT).

[18] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press, Inc., 1990.

[19] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1445–1453, September 1988.

[20] *Channel Compatible DigiCipher HDTV System*, May 14 1992. Submitted by Massachusetts Institute of Technology on behalf of The American Television Alliance.

[21] *Internet Protocol, DARPA Internet Program Protocol Specification*, September 1981. RFC 791.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications.* Internet Engineering Task Force, Audio-Video Transport Working Group, January 1996. RFC 1889.

[23] G. Karlsson, "Asynchronous transfer of video," *IEEE Communications Magazine*, pp. 118–126, August 1996.

[24] H. Schulzrinne, *RTP Profile for Audio and Video Conferences with Minimal Control*, January 1996. RFC 1890.

[25] R. Braden, D. Clark, and S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, July 1994. RFC 1633.

[26] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," *Computer Communication Review*, vol. 23, pp. 289–298, October 1993.

[27] D. Sanghi, A. K. Agrawala, O. Gudmundsson, and B. N. Jain, "Experimental assessment of end-to-end behavior on internet," in *IEEE INFOCOM*, vol. 2, pp. 867–874, 1993.

[28] J.-H. Yao, Y.-M. Chen, and T. Verma, "Mpeg-based audio-on-demand experiment for the internet," in *Global Information Infrastructure (GII) Evolution*, pp. 503–511, 1996.

[29] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," *Computer Communication Review*, vol. 22, pp. 14–26, October 1992.

[30] T. Barzilai, D. Kandlur, A. Mehra, and D. Saha, "Design and implementation of an rsvp-based quality of service architecture for an integrated services internet," *IEEE Journal of Selected Areas in Communications*, vol. 16, pp. 397–413, April 1998.

[31] P. White and J. Crowcroft, "The integrated services in the internet: state of the art," *Proceedings of the IEEE*, vol. 12, pp. 1934–1946, December 1997.

[32] P. White, "Rsvp and integrated services in the internet: a tutorial," *IEEE Communications Magazine*, vol. 35, pp. 100–106, May 1997.

[33] T. Nishida and K. Taniguchi, "Qos controls and service models in the internet," *IEICE Transactions on Communications*, pp. 447–457, April 1995.

[34] I. Dalgic and F. A. Tobagi, "Performance evaluation of atm networks carrying constant and variable bit-rate video traffic," *IEEE Journal on Selected Areas in Communication*, vol. 15, pp. 1115–1131, August 1997.

[35] L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, September 1997. RFC 2205.

[36] M. D. Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*. Prentice Hall, third ed., 1995.

[37] M. Krunz, "Bandwidth allocation strategies for transporting variable-bit-rate video traffic," *IEEE Communications Magazine*, pp. 40–46, January 1999.

[38] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal, "Qos and multipoint support for multimedia applications over the atm abr service," *IEEE Communications Magazine*, pp. 53–57, January 1999.

[39] P. Haskell and D. Messerschmitt, "Resynchronization of motion compensated video affected by atm cell loss," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 545–548, 1992.

[40] R. O. Hinds and T. N. Pappas, "Effect of concealment techniques on perceived video quality," in *Proceedings SPIE Human Vision and Electronic Imaging IV*, 1999.

[41] H. Ohta and T. Kitami, "Simulation study of the cell discard process and the effect of cell loss compensation in atm networks," *The Transactions of the IEICE*, vol. E73, pp. 1704–1711, October 1990.

[42] S. Iai and N. Kitawaki, "Effects of cell losses on picture quality in atm networks," *Electronics and Communications in Japan, Part 1*, vol. 75, no. 10, pp. 30–40, 1992.

[43] U. Bertele and F. Brioschi, *Nonserial dynamic programming.* New York, Academic Press, 1972.

[44] G. Davis and J. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *SPIE*, vol. 2847, pp. 376–386, 1996.

[45] E. Steinbach, N. Färber, and B. Girod, "Standard compatible extension of h.263 for robust video transmission in mobile environments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 872–881, December 1997.