H.26L: Analysis and Real-Time Encoding Algorithms

By

Anthony Peter Joch

B. Eng. (Computer Engineering), McMaster University, 1999

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE STUDIES

Department of Electrical and Computer Engineering

We accept this thesis as conforming to the required standard

The University of British Columbia

January 2002

Abstract

The digital video industry has experienced impressive growth in recent years as the driving force behind a number of applications such as videoconferencing, DVD-Video systems, digital cable television and Internet streaming of video. One of the major factors in the success of this industry has been the development and acceptance of international standards for video coding including ITU-T H.263 and ISO/IEC MPEG-2. In the continuation of this work, the ITU-T Video Coding Experts' Group (VCEG) is currently developing a next-generation video coding standard known as H.26L. This draft standard, which is scheduled for completion in 2002, offers new levels of compression performance and additional features beyond those available in earlier standards. However, these advantages come at the cost of increased complexity and computational demands.

In this thesis, we analyze the rate-distortion performance of the H.26L standard and develop algorithms that increase the speed of video encoding while making minimal sacrifices in terms of rate-distortion performance. First, we establish the optimal rate-distortion performance of the emerging standard and compare this to all other popular visual coding standards. Results will illustrate the improved levels of coding performance that H.26L can provide. Next, we perform a detailed analysis of the features of H.26L that lead to improvements in compression performance. Through this analysis, we will establish a foundation for the development of reduced-complexity encoding algorithms that are intended to enable real-time video applications that can benefit from the improved compression performance of H.26L on current and emerging hardware platforms.

Table of Contents

Abstract .... *ii*

Table of Contents .... *iii*

List of Tables .... *v*

List of Figures .... *vii*

Acknowledgements .... *ix*

1 Introduction *1*

1.1 Introduction and Motivation **1**

1.2 Outline of Thesis **4**

2 Background *5*

2.1 Video Coding **5**

2.1.1 Motion Compensated Prediction 10
2.1.2 Transform Coding 17
2.1.3 Quantization 18
2.1.4 Entropy Coding 18

2.2 Rate-Distortion Optimization **19**

2.3 Defining Complexity of a Video Encoder **25**

2.4 Common Techniques for Reduced Complexity Video Coding **26**

3 Overview of Visual Coding Standards *30*

3.1 JPEG **30**

3.2 JPEG 2000 .... **31**

3.3 H.261 .... **34**

3.4 MPEG-1 .... **36**

3.5 MPEG-2 .... **37**

3.6 H.263 .... **38**

3.7 MPEG-4 .... **43**

3.8 H.26L **46**

3.8.1 Introduction 46
3.8.2 Motion Compensation 47
3.8.3 Intra Prediction 50
3.8.4 Residual Coding 51
3.8.5 Deblocking Filter 52
3.8.6 Entropy Coding 53

4 Compression Performance of Video Standards *57*

4.1 Comparison Methodology 57
4.1.1 Test Sequences 59
4.1.2 Measuring Performance 60
4.2 Experimental Results 61
4.3 Intra Coding 62
4.4 Video Coding for Interactive Applications 66
4.5 Video Coding for Streaming Applications 72
4.6 Conclusions 76
5 Performance Analysis of the Draft H.26L Standard ***77***
5.1 Experimental Procedure 77
5.2 Motion Compensation Block Sizes 79
5.3 Multiple Reference Frames 83
5.4 B-Pictures 88
5.5 Spatial Accuracy for Motion Compensation 93
5.6 Entropy Coding Methods 96
5.7 Summary 99
6 Algorithms for Real-Time H.26L Encoding ***100***
6.1 Introduction 100
6.2 H.26L Test Model Low-Complexity Mode 102
6.3 Fast Integer Pixel Motion Estimation 103
6.3.1 Complexity of the TML Integer Search 104
6.3.2 Description of Proposed Fast-Search Algorithm 105
6.4 Fast Sub-Pixel Refinement 111
6.5 Threshold-Based Early Exits from Motion Estimation 117
6.6 Summary 124
7 Conclusions and Future Research ***125***
Bibliography ***127***
Appendix A: Sample Frames from Video Test Sequences ***135***

List of Tables

Table 4-1: Software implementations of video encoders used in performance comparisons. 59

Table 4-2: Video sequences used in performance comparisons. 60

Table 4-3: Relative intra coding performance of encoders over entire test set. 63

Table 4-4: Relative performance of encoders in interactive video coding comparison. 68

Table 4-5: Relative performance of encoders in streaming video test. 73

Table 5-1: Video sequences used in analysis of H.26L coding features. 79

Table 5-2: Key to combinations of motion compensation block sizes. 80

Table 5-3: Delta-SNR summary for motion compensation block sizes. Coding gains are relative to using the 16x16 motion compensation mode only. 81

Table 5-4: Delta-SNR summary for multi-frame prediction in H.26L and H.263. The two right-most columns show the relative coding gain provided in H.26L compared to H.263 85

Table 5-5: Delta-SNR summary of coding gain provided by adding 2 consecutive B-pictures to bitstreams using 1 and 5 prior reference frames. 89

Table 5-6: Coding improvement for using multiple reference frames with and without 2 consecutive B-pictures already present in the bitstream. 91

Table 5-7: Delta-SNR summary for eighth-pel motion vector accuracy. 94

Table 5-8: Delta-SNR summary for CABAC entropy coding 97

Table 6-1: Number of operations required for integer-search specified for H.26L TML low-complexity mode 105

Table 6-2: Comparison of coding performance and complexity for TML encoder and optimized software using floating-center diamond search, for Foreman and Akiyo. 108

Table 6-3: Number of operations required for sub-pixel refinement by TML algorithm.. 112

Table 6-4: Performance of fast sub-pixel refinement algorithm for Foreman and Akiyo. 115

Table 6-5: Performance of adaptive threshold-based early exits for Foreman and Akiyo. 120

Table 6-6: Performance comparison of using conservative and aggressive threshold values for Foreman and Akiyo. 122

List of Figures

Figure 2-1: Forward motion compensation. 9

Figure 2-2: Generalized block diagram of hybrid block-based motion compensation and transform coding video encoder. 10

Figure 2-3: Motion compensation using 16x16 (left) and 8x8 (right) block size. 11

Figure 2-4: Bi-directional prediction. The prediction for the macroblock in the current frame (center) can originate from either the previous or subsequent reference frame, or a combination of the two. 13

Figure 2-5: Re-ordering of frames in bitstream in data-dependence order. The arrows indicate data dependencies 14

Figure 2-6: Multiple reference frame prediction. Each macroblock in the current frame (right) can be predicted from one of several temporally previous reference frames. .. 15

Figure 2-7: Zig-zag scan order for an 8x8 block, as found in H.263 or MPEG-2. H.26L uses a similar scan pattern on 4x4 transform blocks. 19

Figure 3-1: Example of subband decomposition (Mallat decomposition). 33

Figure 3-2: The seven available modes for motion compensation of each 16x16 macroblock. H.26L supports motion compensation blocks as small as 4x4 pixels..... 48

Figure 3-3: Directional prediction of 4x4 Intra block. Pixels at positions A to I are used to generate a prediction for pixels a to p, using one of the 5 directional modes shown on the right, or DC prediction. 50

Figure 3-4: The single-scan (left) and double-scan (right) coefficient scanning patterns... 52

Figure 4-1: Sample rate-distortion curves for intra coding test. 64

Figure 4-2: Sample rate-distortion curves for interactive coding test. 69

Figure 4-3: Sample rate-distortion curves for streaming video test. 74

Figure 5-1: Sample rate-distortion curves illustrating the performance of different combinations of motion compensation block sizes in H.26L. 82

Figure 5-2: Sample rate-distortion curves illustrating the effect of using 5 previous reference frames in H.263 and H.26L. 86

Figure 5-3: Sample rate-distortion curves illustrating the effect of inserting 2 consecutive B-pictures with and without multiple reference frames in H.26L. 90

Figure 5-4: Sample rate-distortion curves illustrating the effect of the spatial motion vector accuracy in H.26L. 95

Figure 5-5: Sample rate-distortion curves for entropy coding modes in H.26L. 98

Figure 6-1: Rate-distortion performance of fast integer-pixel search 110

Figure 6-2: Illustration of integer-pixel (A-I), half-pixel (1-8) and quarter-pixel (a-n) locations. 112

Figure 6-3: Rate distortion performance of fast sub-pixel refinement algorithm. 116

Figure 6-4: Rate-distortion performance of threshold-based early exits. 121

Figure 6-5: Comparison of rate-distortion performance of conservative and aggressive thresholds. Aggressive threshold produce encoding speeds more than 30% faster... 123

Acknowledgements

First, I would like to offer my sincere thanks to my research supervisors, Dr. Faouzi Kossentini and Dr. Panos Nasiopoulos for their valuable guidance and advice. The time that they have spent contributing both to this research and to my personal growth throughout my studies is greatly appreciated.

Many colleagues have shared both technical expertise and friendship, making these past two years a rewarding and enjoyable experience. In particular, I would like to thank Berna Erol, Dave Tompkins, Ismaeil Ismaeil, Michael Gallant, Yong Yu, Mehran Azimi, and Shahram Shirani. I would also like to acknowledge Alen Docef for his valuable teaching and technical assistance in my early days at UBC.

I would like to express special thanks to my parents, brother and sisters, for putting up with me while I've been living at the other end of the country, and more importantly, for all of the love and support that they have given me in life.

Last, I would like to thank Selena, who has been my partner in this adventure. Thank you for contributing to this work by making life easier for me while I was spending long days and nights working, performing research, and writing. Most of all, thank you for your continuous friendship and love. This experience would not have been the same if I had not shared it with you.

ANTHONY JOCH

University of British Columbia
February, 2002

1 Introduction

In this chapter, we present an introduction to the work contained in this thesis. We begin by summarizing the motivation and objectives for this research. We then proceed by presenting an overview of the remaining sections of this thesis along with a brief description of the material contained within.

1.1 Introduction and Motivation

The importance of digital video compression technology has become evident in recent years through the widespread success of applications in which the efficient representation of video data is critical. Such applications range from DVD-Video (digital versatile disk) and direct broadcast satellite (DBS) systems to videoconferencing and streaming video over the Internet. The growth of this industry has been stimulated by the development and acceptance of international standards for efficient video coding, such as ISO/IEC[1] MPEG-2 [1] and ITU-T[2] H.263 [2].

In the continuation of video coding standardization efforts, the ITU-T Video Coding Experts' Group (VCEG) is currently undertaking the development of a next-generation

[1] International Organization for Standardization/International Electrotechnical Commission
[2] International Telecommunications Union Telecommunications Standardization Sector

standard for video coding, known as H.26L [3]. The goal of the project is to enable a new generation of video applications by providing improved compression capability and other important features and functionality. The first version of the standard is currently scheduled for approval late in 2002. Although existing standards provide adequate compression for many current applications, it is hoped that the improved compression performance provided by H.26L will enable new applications, such as entertainment-quality video over computer network connections and acceptable video quality over low-bandwidth wireless links. Furthermore, the economic viability of many video applications can be improved by more efficient compression, by saving costly transmission bandwidth, or enabling the transmission of improved quality or a larger number of video channels over the same communications link.

Aside from coding efficiency, another critical constraint on the viability of many video applications is the computational complexity of encoding video data. Real-time encoding capability – in which a video signal is encoded as fast as it is captured and displayed – is a requirement for applications such as videoconferencing and live digital television broadcasts. Moreover, in applications such as DVD production for which real-time is not a necessity, the time required for encoding video is still an important factor in the production cost, since production generally consists of several iterations of encoding and subjective evaluation of the video quality by experts. In addition to these time constraints, the complexity of the encoding algorithm directly affects the hardware costs and power consumption requirements of any system. An important problem in the development of H.26L video applications is that the encoding algorithm described the current H.26L test model has very large computational requirements – much too high for real-time encoding on any practical hardware platform. Thus, algorithms that reduce the computational requirements for video encoding without making large sacrifices in coding efficiency are necessary for enabling the use of H.26L in applications that require real-time encoding, and can also be beneficial in non-real-time scenarios.

As we will illustrate, H.26L can provide significantly improved compression performance over all existing standards. It has the potential to enable new applications and become a dominant next-generation standard. While its coding methods are based upon the same the fundamental coding model that is employed in existing video coding standards, H.26L includes a number of added features and additional flexibility. These additions result in significantly increased computational complexity relative to currently popular standards. Thus, real-time encoding without sacrificing compression efficiency becomes increasingly difficult. A large body of research exists that focuses on reduced complexity video coding with respect to the existing standards. Due to similarities in the fundamental coding model, much of this research can be applied to H.26L encoding. However, the many details that differentiate H.26L from other standards and lead to its improved performance must be considered in the encoding algorithm in order to find an optimal balance of encoding complexity and rate-distortion performance.

In this thesis, we analyze the performance of H.26L in detail and develop a set of algorithms for H.26L encoding with greatly reduced complexity that introduce minimal losses in coding efficiency. We begin with a comparison of the optimal rate-distortion performance that H.26L and other popular standards can provide in order to evaluate the improvement in coding efficiency that is possible with H.26L. Next, the individual coding efficiency improvements provided by several key features of H.26L are analyzed in detail. These results provide guidance for developing algorithms to tradeoff compression performance for a reduction in encoding complexity. Finally, we present a set of algorithms for H.26L encoding that are of significantly less computational complexity than the H.26L test model algorithms and yet introduce only minor sacrifices in video reproduction quality. We will illustrate that the combination of these algorithms make real-time H.26L encoding feasible on existing and emerging hardware platforms.

1.2 Outline of Thesis

This thesis is organized as follows. Chapter 2 provides an introduction to video compression concepts, as well as background information on the rate-distortion optimized encoding methods that will be used to establish the optimal coding performance of video standards. Methods of measuring performance and complexity, and common existing techniques for reduced complexity video coding are also discussed. A brief technical overview of the major video coding standards and a detailed description of the draft H.26L standard are given in Chapter 3. In Chapter 4, H.26L and all of the popular existing visual coding standards are compared in terms of their optimal rate-distortion performance, establishing the improved coding performance that can be achieved using an H.26L compliant encoder. Next, the performance of several individual features of H.26L are analyzed in detail in Chapter 5. In Chapter 6, we develop a body of algorithms that lead to greatly reduced encoding complexity and illustrate that real-time H.26L encoding can be realized while achieving significantly improved compression performance compared to the most highly rate-distortion optimized implementations of encoders compliant with currently popular video coding standards. Finally, conclusions and directions for future research are presented in Chapter 7.

2 Background

In this chapter, we present background information on the fundamentals of video coding, rate-distortion optimization methods and complexity issues. The first section provides an overview of the block-based motion-compensated hybrid video coding model that is employed in all popular video coding standards. Emphasis is placed on describing the features that characterize a motion compensation model, which is the most critical factor in the performance of a video coding algorithm. Next, we review the techniques for rate-distortion optimized video coding that will be used to establish a theoretical limit for compression performance for several video coding standards. Finally, we briefly introduce some important concepts in complexity analysis and discuss some of the common methods for reducing the complexity of a block-based motion compensated video encoder.

2.1 Video Coding

The need for efficient coding of video is apparent once the extremely large bandwidth required for the transmission or storage of raw video data is recognized. Consider the example of a medium “CIF” resolution (common-intermediate-format, 352x288 pixels, slightly larger than VHS resolution) sequence transmitted at the typical rate of 30 frames per second. Transmission of three color components at 8 bits per pixel would require a data rate of approximately 73 Mbits/s! Even with the great bandwidth expansion in telecommunications networks in recent years, compression by a factor of 150 or 200 is required to enable transmission over a conventional high-speed network connection. Clearly, efficient compression of video is necessary for viable video communication or storage.

Fortunately, high compression ratios can be achieved while maintaining an acceptable level of subjective quality in the decoded video sequence by exploiting and removing the large amounts of statistical redundancy that exist in typical raw video data. Obviously, within a given coding framework, compression at larger factors will introduce larger amounts of distortion. This is the fundamental rate-distortion tradeoff that exists in all lossy compression systems.

A video sequence is essentially a time ordered sequence of pictures (or frames). As such, statistical redundancy exists in two domains: spatial and temporal. Spatial redundancy exists because of the strong correlation between the value of a given pixel and the values of nearby pixels within the same picture. Still image coding methods take advantage of this correlation to achieve efficient compression of individual pictures.

The most popular still image coding methods are transform-based [4][5]. In these methods, a decorrelating transform is applied to the raw image data, compacting the signal energy into a small number of coefficients before these are quantized and entropy coded to form a bitstream. In the popular Baseline JPEG standard [6], the discrete cosine transform (DCT) is applied to 8x8 blocks of image data, decomposing the raw data of the block into its frequency components. On the other hand, the emerging JPEG 2000 standard [7] is based on a wavelet transform that provides an efficient and flexible representation of the image data through a decomposition into multiple resolutions.

Still image coding methods can be used to encode video data by coding each frame in a video sequence as an independent still image. Within the context of video coding, this is referred to as intra coding, since each picture is coded without reference to other pictures in the sequence. While this is not the most efficient way of coding video, corresponding systems such as Motion JPEG are currently popular in production-quality editing applications and low-complexity digital video cameras, in which coding efficiency is not the most important constraint in the system.

Video coders can achieve significantly improved compression performance by exploiting and removing the large amount of temporal redundancy that generally exists in video data [8]. Temporal redundancy is present because consecutive images in a video sequence typically have very similar content. Therefore, pixel values in a picture that is being encoded can be predicted from the values of pixels in a previously transmitted picture. Coding methods that take advantage of correlation between different pictures are referred to as inter coding methods.

A simple way to take advantage of temporal correlation is to only transmit intra coded updates for changing areas in a video scene while leaving the remaining areas of the picture unchanged. This method, called conditional replenishment [9] [10], has a serious shortcoming, namely its inability to refine a prediction. Often the content of an area of a previous picture provides a good approximation to the current picture, requiring only a minor refinement in order to become a better representation. The addition of a third coding mode, in which a prediction from a prior picture is refined by encoding the difference between the current picture and the prediction can improve compression efficiency significantly. Still, this relatively simple model, known as frame difference refinement, breaks down quickly if there is any significant amount of motion or camera movement in the video sequence.

This simple model can be improved to account for much of the typical motion in video content by allowing information about the motion between frames to be transmitted as side information in the video bitstream. This technique is known as motion-compensated prediction (MCP) and is the key to efficient representation of video. While several motion compensation models have been presented in the literature, including pel-recursive [11] and model-based [12] methods, the translational block-based motion compensation model [13] forms the basis of all popular video coding standards since it offers good tradeoffs between complexity and performance. The basis for this model is that in typical video sequences, many of the differences between pictures that are in close temporal proximity are the result of translational motion of objects in a scene relative to the imaging plane. An efficient way of representing such motion is to transmit a spatial displacement (or motion vector) for each block of pixels in a picture, indicating the position within the previous picture of the best match for each block. This block-based model assumes that all pixels that share a common motion vector undergo uniform translational motion. The encoder's search for the best motion vectors to transmit is called motion estimation, and the process of using these motion vectors to form a predicted frame is known as motion compensation. This process is illustrated in Figure 2-1. After motion compensation has been performed, a residual signal that is formed by subtracting the predicted picture from the current picture remains. This signal, called the displaced frame difference (DFD), is often encoded using DCT-based spatial coding methods.

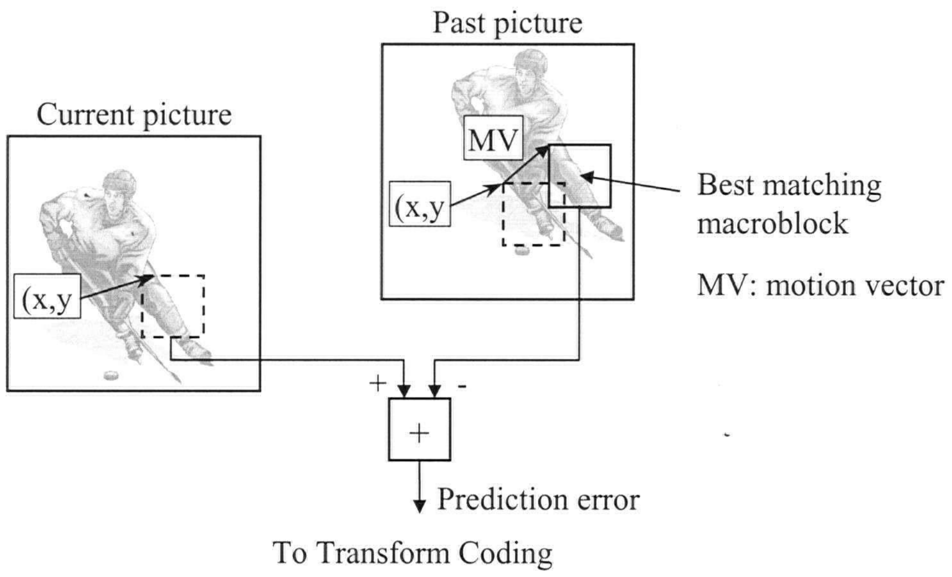


Figure 2-1: Forward motion compensation.

Most successful video coding standards follow essentially this above strategy, which is known as motion-compensated and transform-based hybrid video coding, since it uses a combination of motion compensation and spatial transform techniques. Figure 2-2 shows a generalized block diagram of a hybrid video encoder. Since all of the popular standards share this common framework, the differences that exist between them originate in the details that they specify for each of the key coding processes. In particular, the model used for motion compensation has the most significant impact on the rate-distortion performance achievable for a particular standard. This is discussed next.

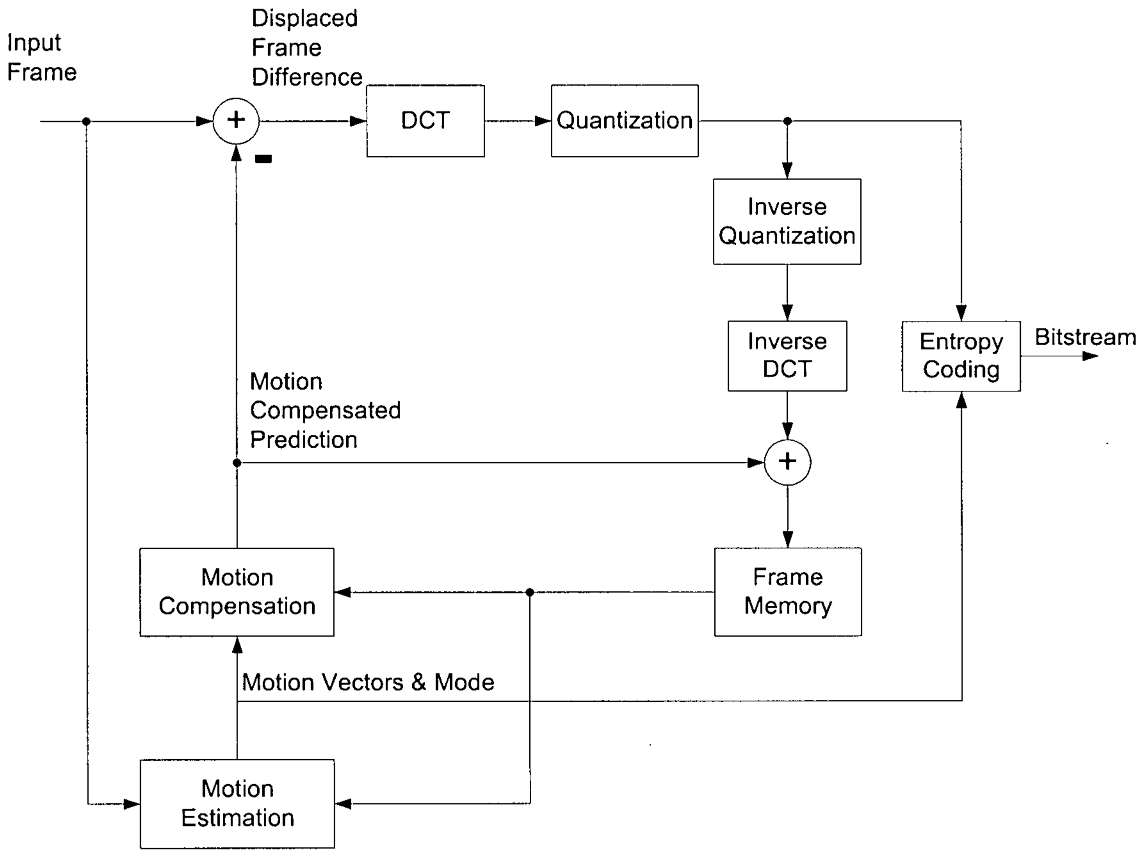


Figure 2-2: Generalized block diagram of hybrid block-based motion compensation and transform coding video encoder.

2.1.1 Motion Compensated Prediction

The upper bound of rate-distortion performance achievable with any compression method is determined by the ability of the underlying coding model to efficiently capture the main characteristics of the source. Since temporal redundancy is the dominant statistical characteristic in typical video source data, the model used to compensate for motion is the most critical factor in determining the optimal rate-distortion performance level for a video encoder. Thus, it is important to understand some of the key issues involved in the design of a motion compensation model in order to appreciate the differences that exist between different standards in terms of coding performance and complexity. Three key issues that will be discussed in detail are the size of the blocks used for motion compensation, the availability of reference frames from which predictions can be derived, and the spatial accuracy with which motion vectors can be specified.

2.1.1.1 Block Size

The block-based translational motion model assumes that all pixels contained within a block with a common motion vector undergo the same uniform translational movement from the picture used for prediction (called the reference picture) to the picture being encoded. The size of these blocks affects the model's ability to capture fine motion detail [14]. While larger blocks may not be able to capture intricate motion, the use of smaller blocks requires transmission of a larger number of motion vectors, adding overhead to the bitstream and leaving fewer bits for coding the frame difference. Hence, smaller blocks tend to be more useful when the available bit rate is relatively high, so that the motion vector overhead occupies only a small fraction of the total bit rate. The use of smaller blocks may also improve subjective visual quality because their presence does not produce large blocking artifacts.

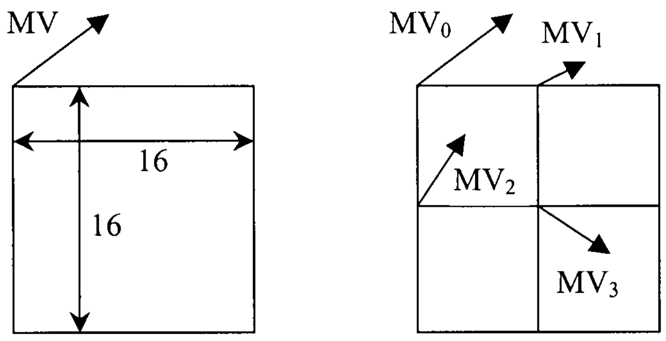


Figure 2-3: Motion compensation using 16x16 (left) and 8x8 (right) block size.

The earliest hybrid video coding standards supported motion compensation for blocks of 16x16 pixels (called a macroblock). Later standards introduced the concept of variable motion-compensation block sizes by providing the ability to alternatively transmit four motion vectors per macroblock, one for each of its 8x8 blocks, as illustrated in Figure 2-3. The next-generation H.26L standard takes this concept a step further by allowing a number of different block sizes and shapes, with blocks as small as 4x4, requiring the transmission of 16 motion vectors per macroblock.

2.1.1.2 Reference Pictures

A second critical issue in a block-based motion compensation model is the specification of which reference pictures are available for predicting the current picture. The simplest method is to use unidirectional forward prediction with a single reference picture, in which the entire picture is predicted only with reference to the temporally previous picture. Pictures that use only forward prediction are referred to as temporally predicted pictures (P-pictures).

Both past and future reference pictures are available when encoding bi-directionally predicted pictures (B-pictures) [8]. The prediction for each macroblock can come from either a temporally previous picture (forward prediction) or a temporally subsequent picture (backward prediction). Additionally, the average of two prediction macroblocks – one from each picture – can be used to generate a bidirectionally predicted macroblock. This concept is illustrated in Figure 2-4. B-pictures are not generally used to predict other pictures.

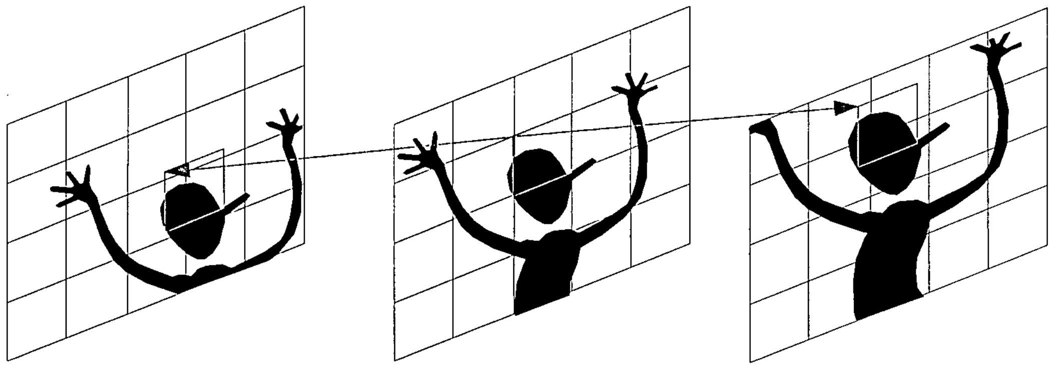


Figure 2-4: Bi-directional prediction. The prediction for the macroblock in the current frame (center) can originate from either the previous or subsequent reference frame, or a combination of the two.

The primary advantage of using B-pictures is significantly improved compression efficiency relative to using forward prediction only [15]. Experimental evidence has shown that the insertion of two B-pictures between each pair of reference pictures is suitable for most video content. By allowing prediction from both past and future pictures, many more options are available to an encoder that is attempting to find the best possible predicted macroblock. Moreover, because they are generally not used as reference pictures, B-pictures can be encoded with slightly lower fidelity than that of the reference pictures, resulting in further bit savings.

In addition to the increased complexity and memory requirements in both the encoder and decoder that B-pictures impose, another significant disadvantage associated with their use is the delay that they introduce into the encoding process. This delay is due to the fact that the pictures in the bitstream must be encoded and transmitted out of their original order because a temporally subsequent reference picture must be decoded before any B-pictures that make use of this reference picture to generate backward predictions. This re-ordering of pictures is illustrated in Figure 2-5. The additional delay of a few frame periods that is introduced by B-pictures makes their use generally unacceptable in interactive video applications, such as videoconferencing, in which delay is not tolerated by users. In applications where strict delay constraints do not exist, B-pictures provide a good means of improving coding efficiency. For example, B-pictures are often used by encoders that are compliant with the widely implemented main profile of the MPEG-2 standard. Such encoders are intended for use in non real-time applications.

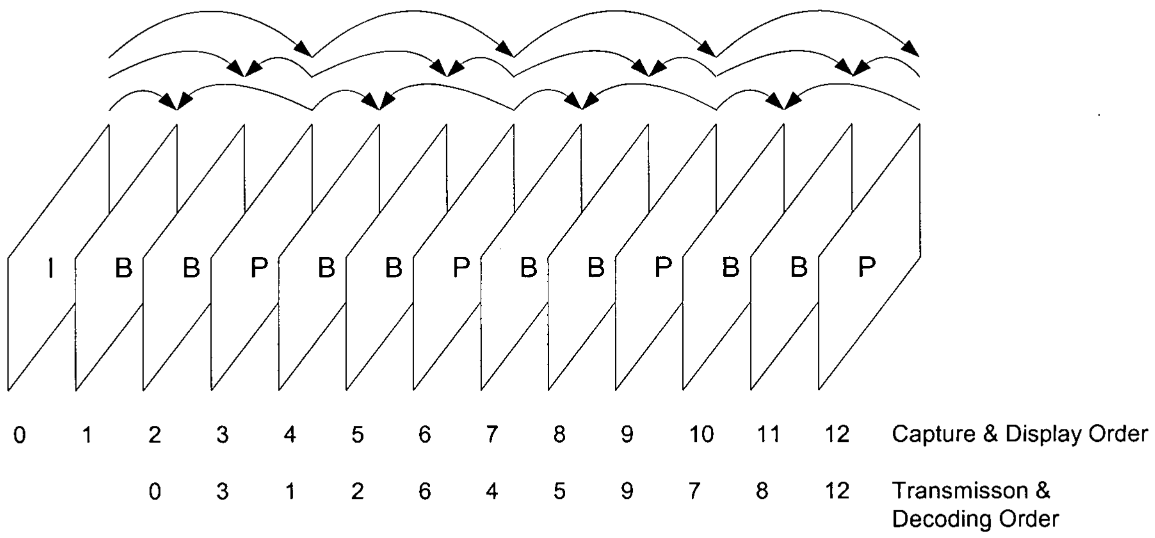


Figure 2-5: Re-ordering of frames in bitstream in data-dependence order. The arrows indicate data dependencies.

More recent work has shown that significant gains in coding efficiency can be achieved by allowing the prediction for each block to be selected from one of several temporally previous reference pictures, instead of just the most recent. This concept, known as long-term memory motion compensated prediction, or multiple reference frame prediction [16], is the basis for a recent addition to the H.263 standard and is also an important technical feature in the emerging H.26L standard. The underlying idea is to extend the motion vector for each block or macroblock into the temporal domain to permit the selection of one of a number of possible reference pictures, as illustrated in

Figure 2-6. Typically, five to ten reference pictures are used, with a larger number of pictures enabling improved coding efficiency at the cost of additional complexity and memory.

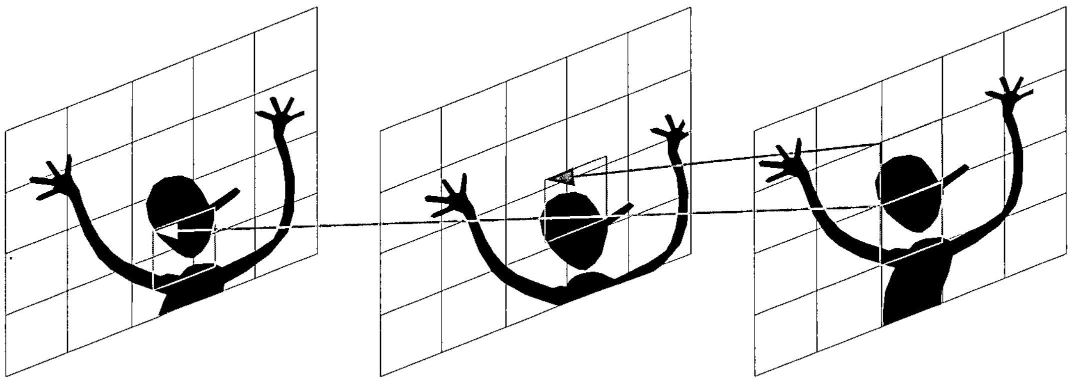


Figure 2-6: Multiple reference frame prediction. Each macroblock in the current frame (right) can be predicted from one of several temporally previous reference frames.

Besides the additional memory requirements, encoding complexity is increased, as with B-pictures, due to the larger search space for motion estimation. However, while rate-distortion improvements provided by long-term memory motion compensation tend to be less than what B-pictures can provide, the advantage of the long-term prediction approach is that the number of pictures available for prediction can be increased while still using only forward prediction. Hence, coding efficiency can be improved without introducing the coding delay that is associated with B-pictures. It should also be noted that both of these techniques can be used simultaneously, usually with only one subsequent reference picture for backward prediction and multiple previous pictures for forward prediction.

2.1.1.3 *Spatial Displacement Accuracy*

The spatial accuracy with which motion vectors are specified is another important factor that can affect the performance of a motion compensation model [17] [18]. With integer-pixel (or full-pixel) accurate motion compensation, motion vectors are confined to indicate points on the integer pixel grid only. Because the true motion in a video sequence is not confined in such a way, a more accurate representation of motion can be obtained by allowing motion vectors to specify locations on a finer, sub-pixel grid. Predictions at these positions are computed by interpolating between the pixel values at full-pixel positions using an interpolation process specific to the coding syntax being used. The properties of the interpolation filter can have a significant effect on the rate-distortion performance and the computational complexity of the system.

Most popular video coding standards support at least half-pixel accurate motion compensation, with bilinear interpolation of the half-pixel values. Quarter-pixel accuracy has been included in more recent standards, and intervals as small as one-eighth of a pixel are being considered for inclusion in H.26L. Finer motion vector accuracies are generally most beneficial for encoding content that contains fine spatial detail at relatively high bit rates. The primary disadvantage of using more accurate motion vectors is that the bit rate required to represent the motion vectors increases in direct proportion to their accuracy. Thus, there is a tradeoff between the improvement in motion compensated prediction that results from allowing more accurate prediction and the additional overhead needed for transmitting the motion vectors. Also, higher spatial accuracy increases the complexity of the codec.

2.1.2 Transform Coding

After motion estimation has been performed to find a suitable matching block, this prediction block is subtracted from the original block to produce a residual signal, which can then be encoded using spatial coding methods. A transform is applied to the residual signal, the purpose of which is to decorrelate the image signal so that its energy is compacted into a small number of transform-domain coefficients. The transform that provides optimal decorrelation is the Karhunen-Loeve transform (KLT) [19]. However, the KLT is data dependent since it is based on the auto-covariance matrix of the input signal and must be recalculated to adapt to non-stationary signal statistics and then transmitted as side information to the receiver. Moreover, the computational complexity of the KLT is relatively high, since no fast algorithms exist for computing this transform. To address these problems, most image and video coding algorithms have adopted the Discrete Cosine Transform (DCT) [20] because it is a suitable approximation to the KLT for most image and video and many fast algorithms exist to reduce its computational complexity [4] [21]. In most image and video coding applications, a two-dimensional DCT is applied to an 8x8 block of data. The forward transform is given by

$$y_{kl} = \frac{c(k)\,c(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} x_{ij}\cos\left(\frac{(2i+1)k\pi}{16}\right)\cos\left(\frac{(2j+1)l\pi}{16}\right) \qquad (1)$$

where $k, l = 0, 1, \ldots, 7$ and $c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \\ 1 & \text{otherwise} \end{cases}$

One drawback of the DCT is that it is defined in terms of floating-point values, while fixed-point arithmetic is much more efficient on digital processors. To efficiently implement the DCT on a fixed-point processor, rounding error must be introduced, which can lead to mismatch between the encoder and decoder due to rounding differences in the inverse transform. In order to eliminate this problem, the transform adopted in the draft H.26L standard approximates the DCT, but is specified using simple fixed-point operations.

2.1.3 Quantization

Following the forward transform, coefficients are quantized through a process of division by a quantization step-size and then rounding to the nearest integer value. Thus, quantization is lossy process that results in less variance between quantized coefficients as compared to DCT coefficients as well as a reduction in the number of non-zero coefficients. Larger quantization step sizes introduce larger amounts of distortion but provide higher compression factors.

2.1.4 Entropy Coding

Prior to entropy coding, the quantized DCT coefficients are arranged into a one-dimensional array using a zig-zag scanning pattern, as illustrated in Figure 2-7. This places the DC coefficient for the block first in the scan, followed by the AC coefficients ordered roughly from low frequency to high frequency. The purpose of this ordering is to produce long runs of consecutive zero-valued coefficients towards the end of the scan, which allows for a more compact representation using run-length codes. Each coefficient is represented as a combination of its run, which is the distance between two non-zero coefficient in the scan order, and the quantized level of the coefficient.

Entropy coding is a lossless process in which these run-level pairs, along with motion vectors and side information such as the picture type, coding modes, and the quantization step-size are efficiently coded to generate a video bitstream. In entropy coding, the most probable values for each syntax element are assigned the shortest codes. The two popular methods that are used in video compression are Huffman coding and arithmetic coding. Huffman coding uses stationary symbol statistics to assign a specific code for each symbol value depending on the long-term probability of each symbol value. Arithmetic coding is more complex, but more efficient, because it includes the ability to adapt to non-stationary symbol statistics and to allow symbols to be encoded using a fractional number of bits.

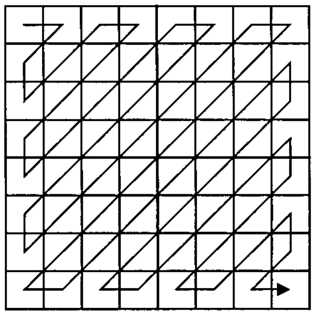


Figure 2-7: Zig-zag scan order for an 8x8 block, as found in H.263 or MPEG-2. H.26L uses a similar scan pattern on 4x4 transform blocks.

2.2 Rate-Distortion Optimization

All modern video coding standards only specify the operation of a compliant decoder – the syntax that it can understand and its method for interpreting the syntax in order to generate a decoded video signal. While the encoder is constrained to operate within the framework supported by the decoder, a great deal of flexibility remains in the selection of a number of important coding parameters, such as the choice of motion vectors, the quantization step size, and the coding mode for each macroblock. The level of rate-distortion performance that is realized by a compliant encoder can vary greatly depending on the values selected for these coding parameters. However, because the encoder's operation is limited by the syntax of the standard, a fixed upper bound exists on the level of rate-distortion performance that can be achieved. This bound is largely determined by the ability of the coding framework supported by the standard to capture the key characteristics of the input source.

Since our objective in Chapter 4 will be to compare the highest rate-distortion performance levels of various standards, it is necessary to use encoder implementations that make coding decisions that are optimized to achieve a rate-distortion performance approaching this upper bound for each standard. However, different methods and levels of optimization exist. For example, the basic coding unit upon which optimized coding decisions are made and the methods used to make the decisions can vary, resulting in different degrees of optimization. In this work, all of the tested video standards used in this comparison are based on the same fundamental coding structure. Therefore, the same optimization algorithm – with minor adjustments to adapt to the specific details of each standard – can be implemented in all encoders, permitting a fair comparison of the true capabilities of each of the standards.

During the past several years, there have been significant research activities in the area of rate-distortion optimization algorithms for video coding [22]. Methods based on the Lagrangian formulation [23] have gained considerable importance, due to their effectiveness, conceptual simplicity, and their ability to effectively evaluate a large number of coding choices in an optimized fashion. One popular Lagrangian-based RDoptimization algorithm [25] has been implemented in several video encoders in order to generate data for the comparison of rate-distortion performance that is presented in Chapter 4. This method is described next.

The objective of rate-distortion optimization, subject to a bit rate constraint, can be stated as follows: *Minimize the distortion D, subject to a constraint R_c on the number of bits used, R.*

$$\min(D), \text{ subject to } R < R_c \qquad (2)$$

This constrained optimization problem can be converted to an unconstrained problem and solved by using the Lagrangian formulation, in which the distortion term is weighted against the rate term in a minimization problem [24]

$$\min\{J\}, \text{ where } J = D + \lambda R, \text{ for some } \lambda \geq 0 \qquad (3)$$

where J is the Lagrangian cost function that must be minimized for a particular value of the Lagrange multiplier λ. Each solution to this unconstrained problem (3) for a given value of λ is also a solution to the constrained problem (2), for some value of R_c.

Obviously, finding an optimal solution to this minimization problem requires the ability to measure distortion – or its inverse – visual quality. However, the amount of distortion perceived by a human viewer is a difficult measure to quantify due to the complex nature of the human visual system. In practice, some widely accepted yet imperfect models are used to obtain approximate measures of visual distortion. Among these, the sum of squared differences (SSD) is the most common distortion measure used in video encoder optimizations. Alternate expressions for this measure include the mean squared error (MSE) and the peak signal-to-noise ratio (PSNR). The PSNR is defined by

$$PSNR_A(F,G) = 10\log_{10}\frac{(255)^2}{MSE_A(F,G)}, where$$

$$MSE_A(F,G) = \frac{1}{|A|}\sum_{s\in A} SSD_A(F,G), and$$

$$SSD_A(F,G) = \sum_{s\in A}(F(s) - G(s))^2$$

where F and G are two arrays, such as the luminance arrays of the original and decoded pictures, s=(x,y) is a pixel location and A represents the area of interest. Another distortion measure that is used commonly in the motion estimation process of an encoder due to its effectiveness [19] and computational simplicity is the sum of absolute differences:

$$SAD_A(F,G) = \sum_{s\in A}|F(s) - G(s)|.$$

In the rate-distortion optimized selection of motion vectors, the distortion measure typically used in the minimization of Lagrangian formulation is the difference between the original block and the predicted block. This can be represented with the SSD, but the less complex SAD is typically used due to the large number of search positions. Thus, the selection of the optimal motion vector for a block can be performed by minimizing:

$$J_{MOTION} = SAD + \lambda_{MOTION}R_{MOTION} \qquad (4)$$

The rate term in the Lagrangian formulation, R_{MOTION}, represents the number of bits used to encode the motion vector. λ_{MOTION} represents the Lagrangian multiplier in the rate-constrained motion vector search. Minimization of the Lagrangian is only guaranteed by considering all candidate motion vectors within the search area.

Once the optimal motion vectors for each possible coding mode have been determined, the optimal coding mode must be selected. The three primary possibilities are SKIP (copying from the same position in the reference picture), INTRA (no temporal prediction), and INTER (motion-compensated prediction plus coding of a residual). Furthermore, several variants of the INTER mode with different blocks sizes, as supported by a given standard, might be tested. The Lagrangian function to be minimized can be written as

$$J(M, Q) = D_{REC}(M,Q) + \lambda_{MODE} R_{REC}(M,Q), \qquad (5)$$

where, $M \in \{SKIP, INTRA, INTER_1 \ldots INTER_N\}$ and Q is the quantization step size. The subscript N indicates the number of valid INTER modes to be tested. $D_{REC}(M,Q)$ is the SSD between the original macroblock and its reconstruction, and $R_{REC}(M,Q)$ represents the number of bits required to code the macroblock using mode M and step size Q.

In [25], Sullivan and Wiegand presented a general relationship between λ_{MODE} and the H.263 macroblock quantization parameter, QP:

$$\lambda_{MODE} = c \cdot (QP)^2$$

where Q is $2 \cdot QP$ and c is a constant that depends on the coding framework that is being used. This relationship was obtained by measuring the quantization step size that minimizes the Lagrangian formulation given in (5) for a given value of λ, over a variety of source material. A value of c equal to 0.85 was determined for an H.263 encoder. Similar experiments have been performed using encoders compliant with other standards in order to select optimal values for their Lagrangian multipliers. To determine the Lagrangian multiplier for the motion search, λ_{MOTION}, a simple modification must be made to compensate for the lack of a squaring operation in the SAD distortion measure. Thus, both Lagrangian multipliers can be expressed as a function of the quantization step size:

$$\lambda_{MODE} = 0.85 \cdot (QP)^2,$$

and

$$\lambda_{MOTION} = \sqrt{\lambda_{MODE}} = 0.92 \cdot QP.$$

Therefore, in a practical video encoder, the step size can be selected by the rate control method, while the Lagrangian multipliers are treated as dependent variables. Then, the coding mode that minimizes the Lagrangian formulation of (5) is selected for the macroblock.

These Lagrangian methods for making coding decisions are employed in all of the video encoders used in our rate-distortion optimized comparisons. It is interesting to note that Lagrangian rate-distortion optimization methods are also employed in the JPEG 2000 verification model implementation for coding decisions specific to its wavelet-transform-based coding framework.

2.3 Defining Complexity of a Video Encoder

Since the goal of this work is to develop reduced complexity encoding algorithms, the problem of how to measure complexity must be addressed. However, measuring the complexity of an algorithm is a multi-dimensional problem. The key dimensions in this problem are the algorithmic complexity, the computational complexity and the size and bandwidth of the memory that is required. Each of these dimensions can have a substantial effect on the cost and performance of a system. The relative significance of each of these factors varies substantially, depending on the hardware platform and whether a hardware or software implementation is required.

Computational complexity is the most conventional method of measuring the complexity of a coding algorithm since it directly affects encoding speed and it is often the simplest to quantify. The computational complexity can be measured in the number of basic operations, such as additions, multiplications, and absolute difference operations that must be performed by an encoder. Quantitative results will be presented using these statistics in Chapter 6, in which reduced complexity H.26L encoding algorithms are presented.

The algorithmic complexity determines the amount of computational logic required to implement the algorithm. This has a direct effect on the number of comparisons and branches in the code, which in turn affects encoding speed in pipelined processors due to pipeline stalls. Furthermore, algorithmic complexity increases development cost, particularly for hardware implementations, where algorithmic complexity is often the most important consideration. For software implementations, code size, which is best kept small, is directly affected by the algorithmic complexity.

The amount of memory required and the bandwidth for memory transfers play an important role in the cost of a system and its performance. The need to frequently access large amounts of memory can be a serious bottleneck. The amount of memory access also affects the performance of the caching system. A large number of cache misses can cause a processor to be underutilized since each cache miss results in a processor stall that can last for hundreds of cycles.

In the development of algorithms in this thesis, we consider all three of these dimensions in order to develop general algorithms that are suitable for most popular platforms. However, development will be geared towards software implementations for PCs and media processors, such as the Texas Instruments C64 and Equator's MAP-CA.

2.4 Common Techniques for Reduced Complexity Video Coding

The most time consuming process in a video encoder is the motion estimation process in which the encoder searches the set of reference frames to find a suitable match for predicting the current block. In fact, motion estimation typically accounts for 50% or more of the total encoding time in an optimized implementation of a current video coding standard [19]. However, with the added degrees of freedom that have been introduced to the motion model in H.26L in terms of the number reference frames, block sizes, and spatial motion vector accuracy, the relative complexity of motion estimation is substantially larger. While this percentage is highly dependent on the search algorithm and implementation details, results have shown that 70-95% of encoding time in H.26L is spent in performing motion estimation [26] [27]. Clearly, the motion estimation algorithm must be the focus of any effort to develop a real-time H.26L encoder.

While performing a full search of the valid search space for block matching is the simplest solution and guarantees that the best match will be found, the time required to match such a large number of blocks would make a real-time implementation impractical. Many algorithms have been developed that achieve block matching performance equal to or slightly worse than the full search at a fraction of the computational complexity [28]. Most of the algorithms work by testing only a small sample of the potential motion vectors in the search area. The search patterns are based on the characteristics of typical video content, such as slowly varying intensity levels. Examples of such algorithms include the logarithmic search [13], three-step search [29], hierarchical search algorithms [28], and the floating-center diamond search [30] [31]. Variants of the floating-center diamond search have been adopted in the H.263 Test Model [32] document and the MPEG-4 Optimization Model [33]. The search used in the H.263 Test Model was developed at UBC and results in an encoder that is approximately 5 times faster than an encoder performing a full search but achieves similar coding performance [31]. This search proceeds by sequentially searching diamond-shaped layers, each of which contains the four immediate neighbors of the current search center. Each subsequent layer is then centered at the point of minimum SAD in the current layer. Thus, successive layers have different centers and contain at most four untested candidate motion vectors. The search is stopped only when the minimum SAD value of the current layer is larger than that of the previous layer, plus some adjustable constant value. This search algorithm will be used as the basis for the fast search algorithm developed for H.26L encoding later in this thesis.

While many search algorithms have been developed to reduce the time required for the integer pixel motion search, a much smaller amount of research has been conducted into reducing the complexity of the sub-pixel refinement of motion vectors. Generally, once the best integer-pixel match has been found, the 8 half-pixel positions that surround it are searched. A similar step is performed for the quarter-pixel refinement, if necessary. When a full integer-pel search is employed, this small number of sub-pixel SAD calculations does not constitute a large part of the overall computational load. However, if a fast integer search is employed, the relative complexity of the sub-pixel stage increases significantly. Again, this complexity increase is amplified in H.26L due to the large number of blocks sizes that may have to be tested, along with the higher spatial motion vector accuracy. Thus, some techniques to reduce the number of SAD calculations in sub-pixel refinement have been developed for MPEG-2 and H.263 encoders [34] [35] [36]. These methods make use of the values of the SADs at integer pixel positions that surround the best integer match in order to determine which of the 8 sub-pixel positions are most likely to provide the best match. Experiments performed in [36] using an H.263 encoder have shown that only a minimal reduction in rate-distortion performance can be achieved while searching only 2-4 of the 8 possible half-pixel positions around an integer motion vector.

One other powerful method to reduce the complexity of the motion estimation process is to include threshold-based exits from the search at certain key points in the algorithm [37]. In these methods, the motion search for a block will be terminated if the minimum SAD found to this point is below a certain threshold. The idea behind these methods is that once a "good enough" match for the current block has been found, there is no need to continue searching since a further substantial reduction in the SAD is not expected. For example, if the SAD for the (0,0) position is very low, as it typically would be for a stationary background block, this motion vector is chosen and no other points will be searched.

The combination of these techniques has been shown to greatly reduce the computational load for the motion search in current video encoders. While some of the basic ideas can be applied to H.26L encoding, special consideration must be made for all of the added dimensions of flexibility that exist in the motion model of H.26L. We will build on these ideas in Chapter 6 to develop reduced-complexity encoding algorithms specific to H.26L.

3 Overview of Visual Coding Standards

In this chapter, we provide an overview of the most popular visual coding standards for both still images and motion video. The focus is on the technical features that determine the optimal rate-distortion performance that is achievable by an encoder that is compliant with each standard. The still image coding standards are presented first, followed by the video coding standards in the order of their development.

3.1 JPEG

Named for the Joint Photographic Experts Group that developed it, this widely successful continuous-tone still picture coding standard [6] [38] [39] is the most common standard format for images on the world-wide-web. The standard was jointly developed by the ISO/IEC JTC1[3] and ITU-T in the late 1980s and was approved in 1992.

Although JPEG also defines progressive, lossless and hierarchical modes of operation, the Baseline sequential coding mode is by far the most widely implemented. In

[3] Joint Technical Committee 1

this mode, a source image is partitioned into a series of 8x8 blocks, which are then decomposed into their spatial frequency components by performing a DCT on each block. The DCT coefficients are then scalar quantized by dividing each coefficient by a corresponding quantization step size and then rounding the result to the nearest integer. The step size for each element within a block is perceptually weighted based on models of the human visual system with a value from a 64-element table. JPEG permits the use of different quantization tables to optimize the encoder for higher compression efficiency.

After quantization, the quantized coefficients are zig-zag scan ordered, leading to improved coding efficiency for the two-dimensional run-level entropy coding of the coefficients. The DC coefficients for each block are differentially coded from the value of the adjacent block. JPEG also supports the use of customized entropy coding tables that can be optimized for specific image content to improve rate-distortion performance. An arithmetic coding mode is also defined in the JPEG standard, but this option is not supported in the popular Baseline mode of operation.

Although it is primarily a still image coding standard, JPEG compliant encoders can also be used as intra-frame video encoders. In this mode of operation known as motion JPEG, each picture is coded as an independent still image. While it provides less coding efficiency than inter-frame methods, this method offers increased flexibility for video editing applications as well as reduced complexity encoding and decoding.

3.2 JPEG 2000

While the original JPEG standard has been widely adopted, many new technologies and applications for still image coding have matured since its completion in the early 1990s.

JPEG 2000 represents an attempt by the JPEG committee to focus these new developments into a next-generation still image coding standard. This standard has been designed to provide improved coding efficiency for still images of many different types and characteristics, and to support many important features for emerging image compression applications within a single unified coding system. Part I of the JPEG 2000 standard [7] [40] [41], which specifies the core coding system, has recently been approved as an international standard by the ISO (ISO 15444-1).

The JPEG 2000 coding engine is based on the discrete wavelet transform (DWT). However, prior to performing a wavelet transform, several optional preprocessing operations may be applied to the image data. These operations sub-divide the image into tiles and condition the raw data so that it can be efficiently coded by the wavelet-based engine. Following this preprocessing stage, each component of each image tile is transformed using one of the two different wavelet transforms that are supported in the core JPEG 2000 system. The (9,7) floating-point wavelet offers the highest compression efficiency, while the (5,3) integer wavelet provides lower complexity and permits mathematically lossless compression. Either wavelet transform decomposes each component into several downsampled frequency bands, or subbands, that can be coded more efficiently than the original data. The decomposition into subbands is generated by iteratively filtering the lowest frequency subband (beginning with the original tile-component) into low and high spatial frequency components in both the horizontal and vertical directions. An example of the resulting subband structure is illustrated in Figure 3-1. After the wavelet transform, all transform coefficients are subjected to uniform scalar quantization with a fixed dead-zone about the origin. Each subband can be quantized using a different step size, depending on the visual importance of the subband. After quantization, the quantized coefficients within each subband are partitioned into rectangular blocks called *code blocks*, each of which is coded independently using a bit-plane coder. This permits truncation of the bitstream at any point to enable scalability on a fine granular level. Truncation points for each of the code block bitstreams are chosen using rate-distortion optimization methods.

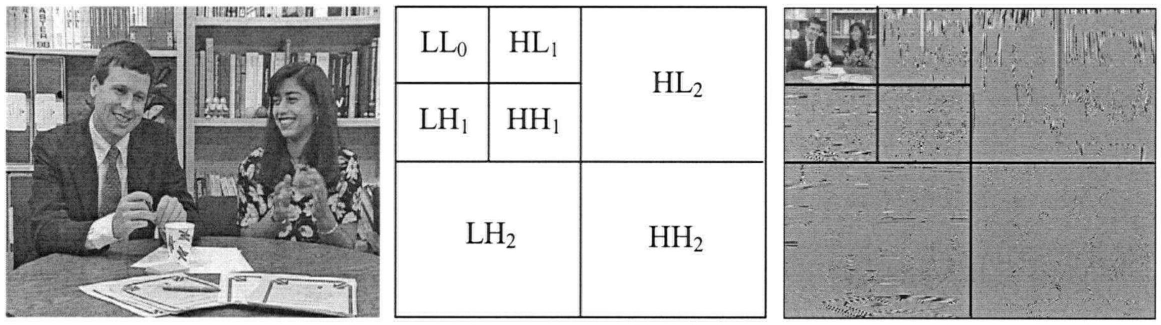


Figure 3-1: Example of subband decomposition (Mallat decomposition).

Using the framework described above, JPEG 2000 provides a rich set of features for image coding that are important to many current and emerging applications. This includes efficient lossy and lossless compression, with especially improved subjective quality at low bit rates compared to the original JPEG standard. JPEG 2000 also supports *scalability* in terms of both image fidelity and resolution. In this way, images of more than one resolution and/or fidelity can be derived from the same coded bitstream. The structure of the JPEG 2000 bitstream supports efficient random access to particular regions of an image without having to decode the entire bitstream. This feature also permits compressed domain image manipulations, such as flipping, rotation and filtering. One final important feature available in JPEG 2000 is robustness in error prone environments, which is achieved using data partitioning, resynchronization, and prioritization techniques.

While Part I of the standard defines the core coding system that must be supported by all implementations of the standard, Part II will define numerous optional coding extensions, such as additional quantization methods and transforms. Part III will define an intraframe video coding format based on the core JPEG 2000 system, called Motion JPEG 2000 (MJ2K). A key feature of MJ2K is the definition of a file format (MJ2) for storing sequences of JPEG 2000 images along with synchronized audio and video metadata. Since each video frame is coded independently, compression efficiency is inferior to that of hybrid video coders, especially at relatively low bit rates. An additional problem with wavelet-based video coders is that wavelet artifacts become much more apparent when a sequence of images is viewed. However, the independent coding of each picture offers advantages in terms of random access and easy editing, reduced memory requirements, and robustness in error prone environments. Hence, MJ2K will be most suitable for relatively high bit rate video coding applications such as professional motion picture production and medical imaging, in which compression efficiency is not the most critical constraint.

3.3 H.261

H.261 [42] [43] was developed from 1988 to 1990 by the ITU-T for teleconferencing applications over ISDN connections. It was approved in early 1991. Although the performance of this standard is not included in any of our comparisons since its performance has been surpassed by more recent standards, H.261 represents a significant achievement in the history of video standardization for two key reasons. First, it was the first video coding standard to become a widespread practical success, finding application in multimedia conferencing systems. In fact, even with the availability of the many improved standards that have been developed after it, H.261 is still currently used in some videoconferencing systems. Second, and perhaps even more importantly, H.261 established the fundamental block-based hybrid motion-compensated and transform coding framework that is employed by all of the more recent and successful video coding standards.

Since it was intended primarily for videophone and videoconferencing applications over ISDN connections that provide channels in multiples of 64 Kbit/s, H.261 is also referred to as a "p x 64" video coding standard. Its target bit rate range is 64-2048 Kbit/s. Only CIF (352x288) and quarter-CIF (QCIF, 176x144) resolutions are supported. Intra coded and temporally predicted pictures (I- and P-pictures) are the only picture types available in the H.261 standard. Bi-directionally predicted pictures (B-pictures) were not included over concerns about delay, complexity and memory requirements. Furthermore, only full-pixel accurate motion compensation on 16x16 macroblocks is supported in H.261 because half-pixel accuracy motion compensation was thought to be too complex at the time of development. However, H.261 includes an optional loop filter that can be applied selectively to each macroblock, allowing an encoder to achieve some of the half-pixel accuracy coding gain. This computationally simple spatial smoothing filter can be used to improve subjective video quality through filtering of the predicted macroblock. Spatial coding in H.261 is very similar to baseline JPEG coding, with the omission of prediction of the DC coefficients and customizable quantization and entropy coding tables. Major similarities include the use of the 8x8 block DCT, scalar quantization and two-dimensional run-level variable-length entropy coding.

3.4 MPEG-1

The Motion Pictures Experts Group (MPEG) was established in 1988 within the framework of the ISO/IEC JTC1 organization (officially ISO/IEC SC29 WG11[4]). The primary objective of the group was to develop standards for the efficient representation of multimedia content for storage and retrieval on digital storage media at bit rates of up to about 1.5 Mbits/s, roughly the data transfer rate supported by standard CD-ROMs. Thus, their first standard, which became known as MPEG-1 [44] [45], was designed to operate at higher resolutions and bit rates than H.261. MPEG-1 is capable of producing VHS quality video at a video bit rate of about 1.2 Mbits/s, with a total target bit rate range for the standard of 1-2 Mbits/s. The MPEG-1 standard was completed in 1992 (ISO 11172-2) and has found application primarily in Video-CD and CD-I systems, as well as in storage of video on computer drives.

Two important technical features that are included in MPEG-1 but are not part of the earlier H.261 standard are bi-directional prediction and half-pixel accurate motion compensation. Both of these features can improve the rate-distortion performance of a compliant encoder significantly, at the cost of additional complexity and memory requirements in both the encoder and decoder. MPEG-1 also includes differential coding of DC coefficients, as in JPEG. In addition to offering improved coding efficiency for its target bit rate range, MPEG-1 was also designed to support enhanced interactivity and

[4] Sub-committee 29, Working Group 11 of ISO/IEC JTC1

access to content. In order to support such features, MPEG-1 bitstreams generally include frequent periodic intra coded pictures that can provide entry points for random access. Like H.261, MPEG-1 was not included in any of our comparisons, since its performance at all bit rates has been well surpassed by later standards.

3.5 MPEG-2

While the MPEG-1 standard was in development, the need was recognized for a standard that supported a more diverse range of applications including entertainment-quality video at high bit rates. Hence, the MPEG-2 standard [1] [46] was developed as an official joint project of both the ISO/IEC JTC1 and ITU-T organizations, and it was completed in late 1994 (ISO 13818-2/ITU-T H.262). While MPEG-2 was originally intended to target interlaced standard-definition television content at bit rates from 4 to 9 Mbits/s, the scope of the standard was expanded to include higher resolutions, such as high-definition television (HDTV), and correspondingly higher bit rates (as high as 80 Mbits/s).

In terms of technical features, MPEG-2 is a superset of MPEG-1, ensuring that every MPEG-2 compliant decoder can decode any valid MPEG-1 bitstream. The most important additional feature relative to MPEG-1 was the inclusion of prediction modes for efficient handling of interlaced-scan video, which is the most common format for broadcast-quality video signals. MPEG-2 also supports scalable video coding, permitting differing levels of quality to be decoded from the same bitstream depending on the computational resources available to each decoder. This mode of operation, however, has not been widely implemented.

In order to manage the larger number of coding tools included in MPEG-2 and the broad range of formats and bit rates supported, the standard introduced the concept of *profiles* and *levels* to define a set of conformance points, each targeting a specific class of applications. These points are designed to facilitate interoperability between different applications of the standard that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a compliant bitstream, whereas a level places constraints on certain key parameters of the bitstream, such as the picture resolution and bit rate.

The most widely implemented conformance point in the MPEG-2 standard is the Main Profile at the Main Level (MP@ML), which finds application in DVD Video systems, digital cable television, terrestrial broadcast of standard definition television, and direct-broadcast satellite (DBS) systems. This conformance point supports coding of CCIR[5] 601 content at bit rates up to 15 Mbits/s and permits use of B-pictures and interlaced prediction modes. In the following chapter, a rate-distortion optimized MPEG-2 encoder is included in a comparison of video encoders for streaming applications, in which several CIF resolution sequences are encoded at bit rates up to 4 Mbits/s. The MPEG-2 bitstreams generated for our comparisons are compliant with the popular MP@ML conformance point.

3.6 H.263

H.263 [2] was the first standard designed specifically to handle transmission of video at bit rates lower than those targeted by H.261, such as those provided by public switched

[5] International Radio Consultative Committee – now the ITU-R, International Telecommunications Union Radio Standardization Sector

telephone networks (PSTN) and wireless networks. It was developed as a project of the ITU-T Video Coding Experts Group (VCEG) and its first version was approved in early 1996. The original target bit rate range of 10-30 Kbits/s, which was designed for the low bit rate modems available at the time, was expanded up to at least 2048 Kbits/s once it became apparent that the coding performance of H.263 was superior to that of H.261 at any bit rate. In fact, comparisons have shown that at low bit rates, H.263 can achieve equivalent quality as H.261 using approximately half the bit rate [47].

The original version of H.263 includes a baseline mode plus four optional advanced coding modes. In the baseline mode, key additional features relative to H.261 include half-pixel accurate motion compensation, median motion vector prediction, and three-dimensional run-level-last variable length coding. The four negotiable advanced coding modes permit improved coding efficiency at the cost of added complexity. Features that can be enabled with these modes include longer motion vectors that can be extrapolated over picture boundaries (Annex D), variable block size motion compensation and overlapped-block motion compensation (Annex F), bi-directional prediction (Annex G) and syntax-based arithmetic coding to replace regular VLC coding (Annex E). H.263 was adopted in several videophone terminal standards, including ITU-T H.324 (PSTN) and H.320 (ISDN), and is currently the most widely used standard for video telecommunications.

Version 2 of H.263 [48], which was officially approved in January 1998, includes 12 new negotiable modes and some additional features that are intended to broaden the range of applications for the standard. The new modes and features improve compression performance, allow the use of scalable bitstreams, enhance performance over packet-switched networks, support custom picture size and clock frequency, and provide supplemental display and external usage capabilities. In this work, the primary interest is in features that can improve compression performance. The unrestricted motion vector mode (Annex D) of version 2 allows longer motion vectors that are coded using reversible VLCs, which can improve both coding efficiency and resilience to bit errors. Coding efficiency for intra macroblocks is improved by the advanced intra coding mode (Annex I). This mode supports prediction of intra DCT transform coefficients from neighboring blocks and specialized quantization and VLC coding methods for intra coefficients. The deblocking filter mode (Annex J) improves subjective visual quality by introducing a deblocking filter inside the motion compensation loop. The filter is applied to the edge boundaries of 8x8 blocks of all reference frames in order to improve prediction and reduce blocking artifacts. B-pictures are included in the scalability mode of H.263 (Annex O), allowing improved coding efficiency and temporally scalable bitstreams. Finally, the modified quantization mode (Annex T) removes some limitations of the baseline syntax in terms of quantization and also improves chrominance fidelity by specifying a smaller step size for chrominance data than for luminance.

A second set of extensions that adds three more optional modes to H.263 was completed and approved late in the year 2000. The data partitioned slice mode (Annex V) [50] can provide enhanced resilience to bitstream corruption, which typically occurs during transmission over wireless channels, by separating header and motion vector information from transform coefficients. Annex W [51] specifies additional backwards-compatible supplemental enhancement information. Interlaced field indications, repeated picture headers, and the indication of the use of a specific fixed-point inverse DCT are some examples of information that can be transmitted using this feature.

Compression efficiency and error resilience against packet loss can be improved by using the enhanced reference picture selection mode (Annex U) [49], which enables long-term memory motion compensation [16]. In this mode, the spatial displacement vectors that indicate motion compensated prediction blocks are extended by variable time delay, permitting the predictions to originate from reference pictures other than the most recently decoded reference picture. Motion compensation performance is improved because of the larger number of possible predictions that are available by including more reference frames in the motion search. This concept removes the assumption that the most recent reference picture always provides better predictions than earlier ones. Thus, a buffer of several previous reference pictures must be maintained by both the encoder and decoder, increasing memory requirements on both sides, and increasing the motion search space for the encoder. Although these drawbacks raised some concerns about the practicality of long-term prediction in real-time video systems, these have largely been set aside by some impressive results, including the demonstration of significantly improved quality in a real-time videoconferencing product [52]. In Annex U, two modes are available for the buffering of reference pictures. The sliding-window mode – in which only the most recent reference pictures are stored – is the simplest and most commonly implemented mode. In the more flexible adaptive buffering mode, buffer management commands can be inserted into the bitstream as side information, permitting an encoder to specify how long each reference picture remains available for prediction, with a constraint on the total size of the picture buffer. The maximum number of reference pictures is typically 5 or 10 when conforming to one of H.263's normative profiles, which are discussed next.

With the large number of optional modes available for H.263 encoding, the ITU-T VCEG recognized the need to define preferred mode combinations in order to facilitate interoperability between H.263 compliant terminals. Consequently, the ITU-T has recently approved Annex X of H.263 [53], which provides a normative definition of profiles, or preferred combinations of optional modes, and levels, which specify maximum values for several key parameters of an H.263 bitstream. Similar to their use in MPEG-2, each profile is designed to target a specific key application, or group of applications that require similar functionality. In this thesis, the rate-distortion capabilities of the *Baseline Profile* and the *Conversational High Compression (CHC) Profile* are compared to other standards for use in low-delay video applications. The Baseline Profile supports only baseline H.263 syntax (i.e. no optional modes) and exists to provide a profile designation to the minimal capability that all compliant decoders must support. The CHC Profile includes most of the optional modes that provide enhanced coding efficiency without the added delay that is introduced by B-pictures and without any optional error resilience features. Hence, it is the best profile to demonstrate the optimal rate-distortion capabilities of the H.263 standard for use in interactive video applications. Additionally, the *High-Latency Profile* of H.263, which adds support for B-pictures to the coding efficiency tools of the CHC Profile, is included in the comparison of encoders for streaming applications, in which the added delay introduced by B-pictures is acceptable.

3.7 MPEG-4

MPEG-4 [54] [55] [56] standardizes efficient content-based coding methods for many types of multimedia data. This includes tools for efficient coding of natural video at high levels of compression, as found in earlier standards for video coding. However, additional key objectives of the MPEG-4 project are to enable content-based interactivity and universal access to audiovisual data, and to provide additional functionalities such as error resilience, scalability, and hybrid coding of synthetic and natural data.

Development of MPEG-4 began in 1993. The first version of the standard (ISO/IEC 14496) was approved early in 1999 and a second version, which contains a set of extensions, was approved early in the year 2000. The core of the MPEG-4 standard consists of three parts: systems, visual and audio. The audio and visual sections specify efficient algorithms for representing audiovisual data, while the systems part addresses the description of the relationship between the audio and visual components that constitute a multimedia scene. In MPEG-4 terminology, these components are referred to as audiovisual objects (AVOs).

Due to the broad range of content, applications, and bit rate categories supported by MPEG-4, the visual part of the standard provides four different types of coding tools: *Video object coding* for coding of a natural and/or synthetically originated, rectangular or arbitrarily shaped video objects, *mesh object coding* for coding of a visual object represented with a mesh structure, *model-based coding* for coding of a synthetic representation and animation of the human face and body, and *still texture coding* for coding of still textures using wavelets. With such a large number of coding tools available, it should not be surprising that MPEG-4 defines a large set of conformance points, or profiles and levels, to facilitate interoperability by grouping sets of tools to target certain key application classes. Since this work deals with natural camera-view content only, the MPEG-4 results generated herein conform to profiles that are intended for such content. Specifically, we consider the rate-distortion performance of the *Simple Profile* of MPEG-4 version 1 and the *Advanced Simple Profile*, which is defined in a recent amendment to the visual standard and makes use of some features that were added in version 2.

Video object coding in MPEG-4 builds upon the features of the baseline H.263 standard, including half-pixel accurate motion compensation on 16x16 macroblocks and spatial coding based on the 8x8 DCT. In fact, all MPEG-4 compliant video decoders must be able to decode any valid H.263 baseline bitstream. Additionally, two different scalar quantization methods are supported in MPEG-4. These are officially called Method 1 and Method 2, but are commonly referred to as MPEG-style and H.263-style, respectively. The main difference between the two methods is that MPEG-style quantization uses customizable perceptual weighting matrices to scale the quantization step size of each AC transform coefficient based on their visual importance, whereas in the H.263 method, all AC coefficients in a block are quantized using the same step size.

The MPEG-4 Simple Profile supports a relatively small feature set within the context of the entire MPEG-4 standard and is designed for efficient and error resilient coding of rectangular video objects (i.e. without shape information) at CIF resolution and smaller. It supports only I- and P-pictures (called I- and P-VOPs, or Video Object Planes, in MPEG-4 terminology). For coding efficiency, the Simple Profile supports many of the key features found in version 2 of H.263, including motion compensation on 8x8 blocks, picture extrapolating motion vectors, and prediction of AC and DC coefficients in intra blocks. Only the less complex H.263-style quantization method is permitted in this profile. For error resilient coding, slice resynchronization, data partitioning, and reversible variable length codes can optionally be used in Simple Profile compliant bitstreams.

The Advanced Simple Profile is also intended for coding rectangular video objects at CIF resolution and smaller, but adds support for several coding tools that can improve coding efficiency significantly, at the cost of added complexity and delay. These additional tools include B-pictures (or B-VOPs), quarter-pixel accurate motion compensation, the MPEG-style quantization method, efficient prediction modes for interlaced video and global motion compensation (GMC). For quarter-pixel motion compensation in MPEG-4, half-pixel values are generated using an 8-tap filter, and then values at quarter-pixel positions are calculated via bilinear interpolation of the half-pixel values. The GMC feature makes it possible to encode global interframe motion using a small number of parameters through warping of the reference frame. The Advanced Simple Profile bitstreams generated in our comparisons made use of quarter-pixel accurate motion compensation and the MPEG-style quantization method in all cases. B-VOPs were used optionally, depending on the delay constraints imposed in each test case (each corresponding to a set of related applications).

3.8 H.26L

3.8.1 Introduction

The ITU-T VCEG (SG16/Q6) is currently undertaking the design of a next-generation video coding standard for natural-scene content in a project known as H.26L. Because it is not designed to be backwards compatible with earlier standards, it is hoped that H.26L will offer significant advantages over all existing standards and therefore empower a new generation of video applications. Development began in 1999 and the current plan calls for approval by the ITU-T by the end of 2002. At the time of writing, H.26L is in its eighth major design draft, or TML-8 (Test Model Long-Term number 8) [3], which was approved at the May/June 2001 meeting of the ITU-T Study Group 16 in Porto Seguro, Brazil. Since H.26L is the focus of this thesis and the features of this emerging standard have not yet been addressed in the literature, we will discuss H.26L in greater detail than the other standards.

The primary objective of the H.26L project is to create a standard that can provide significantly improved compression efficiency relative to prior standards for progressive-scan video content. However, the design of the standard is also influenced by several other important objectives [57], including:

- Using a "back-to-basics" design approach that results in an especially simple and efficient design specification,
- Having a "network-friendly" structure enabling straightforward adaptation for use over a variety of important communication network systems,
- Providing sufficient error and packet-loss resilience for use in mobile (e.g., 3G wireless), Internet, and other environments with unreliable data delivery, and
- Having the capability to be used for low-delay real-time applications.

Although mathematically lossless encoding is not supported in the current draft of the standard, the elimination of encoder/decoder mismatch by specifying the inverse transform with precise integer operations is a good step towards mathematically lossless operation.

The underlying coding system defined by H.26L is superficially similar to that successfully employed in prior video coding standards, such as H.263 and MPEG-2. This includes the use of translational block-based motion compensation, DCT-based residual coding, scalar quantization with an adjustable step size for bit rate control, zigzag scanning, and run-length VLC coding of quantized transform coefficients. However, there are several specific optimizations and additional features that differentiate H.26L from all other standards.

One fundamental concept of H.26L is the separation of the standard design into two distinct layers: a *video coding layer* that is responsible for efficiently representing video content, and a *network adaptation layer* that is responsible for packaging the coded data in an appropriate manner based on the network on which it is transmitted. Our focus in this work is on the rate-distortion performance of the video coding layer.

3.8.2 Motion Compensation

The block-based translational motion compensation model of H.26L supports most of the key features included in earlier video standards and achieves further gains in coding efficiency by providing added flexibility and functionality. In addition to normal P- and B-pictures, H.26L includes support for the use of multiple previous reference frames, as in Annex U of H.263, and a new inter-stream transitional picture called an SP-picture. The inclusion of SP-pictures in a bitstream enables efficient switching between bitstreams with similar content encoded at different bit rates, as well as random access and fast playback modes.

Motion compensation on each 16x16 macroblock can be performed using a large number of different block sizes and shapes, as illustrated in Figure 3-2. H.26L supports seven block sizes, whereas H.263 and MPEG-4 support only two block sizes. Individual motion vectors can be transmitted for blocks as small as 4x4, so up to 16 motion vectors may be transmitted for a single macroblock. Blocks sizes of 16x8, 8x16, 8x8, 8x4, and 4x8 are also supported, as shown. The availability of a large number of prediction modes and smaller motion compensation blocks improves prediction in general, and in particular, the small blocks improve the ability of the motion model to handle fine motion detail and may result in better subjective viewing quality because they do not produce large blocking artifacts.

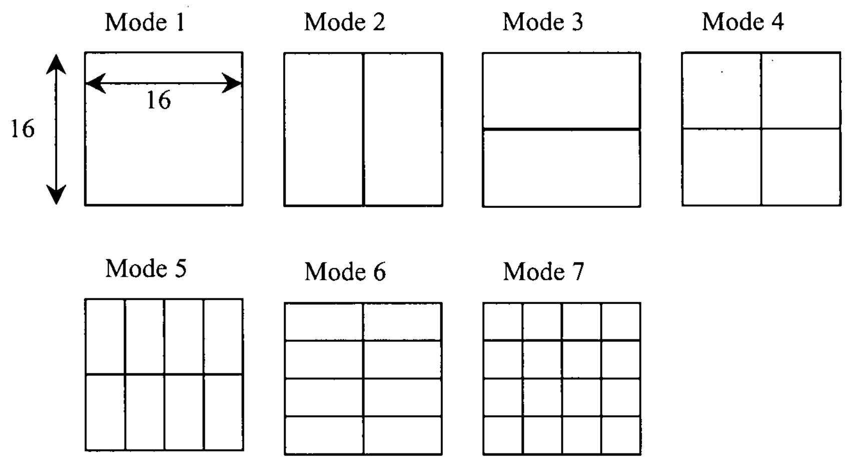


Figure 3-2: The seven available modes for motion compensation of each 16x16 macroblock. H.26L supports motion compensation blocks as small as 4x4 pixels.

The prediction capability of the motion compensation model of H.26L is further improved by allowing motion vectors to be transmitted with higher levels of spatial accuracy than in existing standards. While most existing standards are based primarily on half-pixel accuracy, quarter-pixel accuracy is the lowest accuracy supported in H.26L, and eighth-pixel accuracy has recently been adopted as feature that can sometimes provide increased coding efficiency at the cost of added complexity. In the quarter-pixel mode, values at half-pixel positions are generated using a separable 6-tap filter with coefficients (1, -5, 20, 20, -5, 1)/32. Then, bilinear interpolation is performed on these to generate pixel values at quarter-pixel positions. A feature unique to H.26L is that one out of sixteen quarter-pixel positions is generated using more low-pass filtering than the rest of the positions. This "special position" filtering has been shown to produce improved subjective viewing results [58]. For improved coding efficiency in some circumstances, eighth-pixel accurate prediction using separable 8-tap filters may be used. The interpolation process is performed first in the horizontal direction and then in the vertical direction. Hence, the filtering operation is fully specified by one-dimensional filter taps. The filter taps are given below:

```
Position
Integer:   ( 0,   0,    0, 512,    0,    0,   0,   0)/512  (a simple copy)
1/8:       (-3,  12,  -37, 485,   71,  -21,   6,  -1)/512
2/8:       (-6,  24,  -74, 458,  142,  -42,  12,  -2)/512
3/8:       (-6,  24,  -76, 387,  229,  -60,  18,  -4)/512
4/8:       (-6,  24,  -78, 316,  316,  -78,  24,  -6)/512
5/8:       Mirror image of 3/8 position filter
6/8:       Mirror image of 2/8 position filter
7/8:       Mirror image of 1/8 position filter
```

3.8.3 Intra Prediction

Efficient intra coding is achieved in H.26L through the use of extensive spatial prediction from neighboring blocks for improved decorrelation in areas not using temporal prediction. All pixels are predicted in the spatial domain, rather than predicting transform coefficients, as in H.263 and MPEG-4. This process is illustrated in Figure 3-3, in which pixels A to I from neighboring blocks have already been decoded and may be used for prediction. There are six modes for prediction of 4x4 luminance blocks, including DC prediction (mode 0) and five directional modes, labeled 1 thru 5 in Figure 3-3. The prediction mode for each block is efficiently coded by assigning shorter symbols to more likely modes, where the probability of each mode is determined based on the modes of neighboring blocks. For regions with less spatial detail (i.e. flat regions) H.26L also supports intra coding based on 16x16 macroblocks, in which one of four prediction modes is chosen for the prediction of the entire macroblock and specialized coding of the 16 DC coefficients of each of the 4x4 blocks is performed as described in the following section on residual coding below.

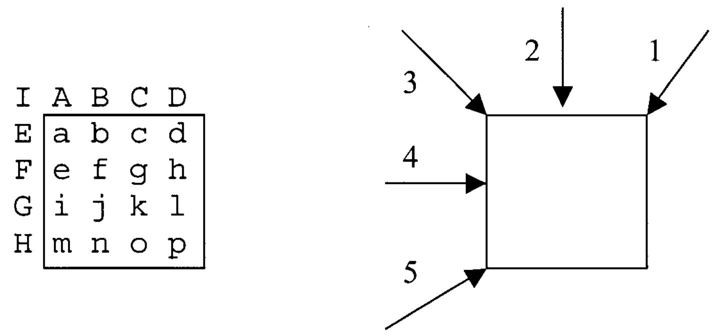


Figure 3-3: Directional prediction of 4x4 Intra block. Pixels at positions A to I are used to generate a prediction for pixels a to p, using one of the 5 directional modes shown on the right, or DC prediction.

3.8.4 Residual Coding

Residual coding in H.26L uses a transform that is primarily 4x4 in shape, as opposed to the 8x8 shape typically found in other standards. The smaller transform size helps to reduce blocking and ringing artifacts. Although the statistical properties of the transform are very similar to those of the DCT, the transform is specified with precise integer operations, as opposed to the usual floating-point DCT specified with rounding-error tolerances, as in earlier standards. This fixed-point specification eliminates the problem of mismatch between encoder and decoder.

Chrominance data is coded using an additional 2x2 transform applied to the DC coefficients of four 4x4 blocks in order to extend the basis functions, since chrominance data tends to have low spatial detail.

When macroblocks are intra coded using 16x16 prediction, an additional 4x4 transform is applied to the DC coefficients of the sixteen 4x4 blocks of the macroblock – this helps by extending the basis functions for regions that are relatively smooth. The forward and inverse transforms are given by:

Definition of transform:

```
A = 13a + 13b + 13c + 13d
B = 17a +  7b -  7c - 17d
C = 13a - 13b - 13c + 13d
D =  7a - 17b + 17c -  7d
```

Definition of inverse transform:

```
a' = 13A + 17B + 13C +  7D
b' = 13A +  7B - 13C - 17D
c' = 13A -  7B - 13C + 17D
d' = 13A - 17B + 13C -  7D
```

The approximate relation between a and a' is: a' = 676a. This is because the expressions defined above contain no normalization. Instead, normalization is performed in the quantization and inverse quantization processes.

Transform coefficients in H.26L are quantized using scalar quantization with no widened dead-zone. Thirty-two different quantization step sizes can be chosen on a macroblock basis – this being similar to the capabilities of prior standards (H.263 supports thirty-one, for example). However, in H.26L, the step sizes are increased at a compounding rate of approximately 12.5%, rather than being increased by a constant increment. The fidelity of chrominance components is improved by using finer quantization step sizes as compared to those used for the luminance coefficients, particularly when the luminance coefficients are coarsely quantized.

For encoding of the quantized transform coefficients, different coefficient-scanning patterns are available in H.26L, as shown in Figure 3-4. The simple zigzag scan is used in most cases, and is identical to the conventional scan used in earlier video coding standards. The double scan is used only for intra blocks that use a small quantization step size, where subdivision of the scan into two parts improves coding efficiency.

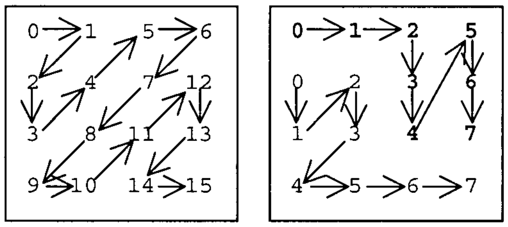


Figure 3-4: The single-scan (left) and double-scan (right) coefficient scanning patterns.

3.8.5 Deblocking Filter

H.26L specifies the use of an adaptive deblocking filter that operates on the horizontal and vertical block edges within the motion compensated prediction loop in order to remove artifacts caused by block prediction errors. The filtering is generally based on 4x4 block boundaries, in which two pixels on either side of the boundary may be updated using a 3-tap filter. Alternatively, if one or both adjacent macroblocks are intra coded, subjective quality is improved by using a stronger 6-tap filter across 16x16 macroblock boundaries. The rules for applying the deblocking filter are intricate. The strength of the filter depends on several factors, including the quantization parameter applied to each block, the length of their motion vectors and whether the blocks were predicted from the same reference frame.

3.8.6 Entropy Coding

The current draft of the H.26L standard specifies two different methods for entropy coding: a simple universal variable length coding (UVLC) method that uses a single table for all syntax elements; and a more complex and effective context-based arithmetic coding (CABAC) alternative. The UVLC method permits simple and efficient encoding and decoding using a single table of codewords that may be written in the following compressed form:

$$\begin{array}{c}
1 \\
0\ x_0\ 1 \\
0\ x_1\ 0\ x_0\ 1 \\
0\ x_2\ 0\ x_1\ 0\ x_0\ 1 \\
0\ x_3\ 0\ x_2\ 0\ x_1\ 0\ x_0\ 1 \\
\ldots\ldots\ldots\ldots\ldots\ldots
\end{array}$$

where x_n take values 0 or 1. A codeword may be uniquely referred to by its length in bits (L) and INFO = x_n .. x_1 x_0. Notice that the number of bits in INFO is L/2 (assuming division with truncation). Each codeword is assigned a code number using:

code number = $2^{L/2}$ + INFO −1,

where L/2 uses division with truncation and INFO = 0 when L = 1.

For each parameter to be coded, there is a conversion rule from the parameter value to the code number, which assigns the shortest codewords to the most frequent values for the parameter. The regular structure of the table makes it easy to create a codeword bit by bit, given the values of L and INFO that identify the required code number. Similarly, a decoder may easily read bit by bit until the last "1", which signals the end of the codeword. L and INFO are then readily available for determining the code number and the corresponding parameter value.

More recently, the use of context-based adaptive binary arithmetic coding (CABAC) has been adopted into the standard as a way of gaining additional compression performance, provided sufficient processing capability is available. In the CABAC scheme, probability models are created for each syntax element and adaptively updated throughout the encoding process. These models are used by the arithmetic coding engine, which permits a non-integer number of bits to be assigned to each symbol based on the probability of each symbol value. The adaptivity of the probability models permits efficient encoding in the presence of non-stationary symbol statistics. The CABAC mode has been shown to increase compression efficiency by 5 to 30% relative to the UVLC mode, with the largest improvements occurring at the bit rate extremes (i.e. very high and very low bit rates) [59]. However, CABAC is a much more complex algorithm than UVLC.

In the CABAC entropy coding scheme, the first stage in encoding a symbol for an arbitrary syntax element is called *context modeling*, in which a suitable probability model for coding the syntax element is generated. The model is conditioned on the values of neighboring symbols that have already been encoded. Different models are maintained for each syntax element (e.g., motion vectors and transform coefficients have different models). Next, in a process called *binarization*, non-binary symbols are mapped onto a sequence of binary decisions, or *bins*, according to a specific binary tree. A unary code tree is used for most syntax elements. Finally, each binary decision is encoded with the *adaptive binary arithmetic coding* (AC) engine using the probability estimates that have been provided by the context modeling stage or the binarization process itself. The arithmetic coding engine can assign a non-integer number of bits to each symbol, and the most probable values are encoded with the fewest bits, providing highly efficient entropy coding. Arithmetic coding is straightforward and similar to the techniques described in [60]. After encoding of each binary decision, the related probability model is updated based on the value that was encoded in order to adapt the model to the actual statistics. These probability models are initialized at the start of each frame with a pre-computed distribution, and as the adaptation occurs, they are periodically rescaled in order to exponentially weigh down past observations. This helps the arithmetic coding engine to adapt to the non-stationary symbol statistics of this source. The adaptability is an important feature of the CABAC scheme, permitting near-optimal entropy coding for all types of content and bit rates.

Since technical work on the standard is still in progress, profiles and levels for H.26L have not yet been defined. H.26L results for the inter-standard comparison in Chapter 4 were generated by configuring the encoder to provide optimal rate-distortion coding performance within the application specific constraints imposed by each of our test cases.

This includes the use of CABAC entropy coding and five previous reference pictures for long-term prediction in all cases, with B-pictures included where permitted by delay constraints.

4 Compression Performance of Video Standards

In this chapter, we compare the rate-distortion performance of several visual coding standards to that of the emerging H.26L standard. Three separate comparisons are described, each designed around a specific set of application-based constraints. The purpose of this comparison is to determine the improvement in coding efficiency that H.26L can provide versus existing standards and to establish an optimal bound of rate-distortion performance for H.26L, as well as the other standards. This analysis is important for determining whether the improved compression performance offered by H.26L will justify its implementation costs by enabling new applications and making existing video applications more cost effective.

4.1 Comparison Methodology

When evaluating the rate-distortion performance of different video coding standards, many parameters must be controlled in order to produce a fair comparison the capabilities of each standard. One key problem that has already been discussed in section 2.2 is the operational control of the encoder. A fair comparison requires that all of the encoders are similarly optimized in terms of the decision making process that is used to generate a bitstream. To address this issue, all of the video encoders that we compare use the same Lagrangian-based method for rate-distortion optimization in the motion estimation process and coding mode decision for each macroblock, as described in section 2.2.

Beyond the mode decision process, there are several other factors that can affect the measured rate-distortion performance of a video encoder. Rate control methods that vary the quantization step size in order to produce a constant bit rate (CBR) video stream can have a significant impact on coding efficiency because of the additional constraints and bitstream overhead that are necessary to achieve rate control. Furthermore, the performance of these algorithms can vary greatly from one encoder to the next. Since rate control is largely an encoder-specific issue and not imposed by the standard itself, no form of rate control was employed by any of the encoders in these tests. Instead, a constant quantization step size for each picture type was used in each coding pass of an input sequence.

Many of the standards that we tested also include a large number of optional coding modes that can affect rate-distortion performance, and other important parameters such as complexity and delay. These standards also generally define profiles, which are logical groupings of these optional modes to target a particular application space. In our comparisons, we select optional modes for each encoder based on the profiles that are defined for that standard and the application-based constraints imposed by each of our test cases.

The software implementations used in our comparisons are described in Table 4-1. Some modifications have been made to the public software packages in order to enable a fair comparison of standards, based on the issues described above. For example, rate control was disabled in the MPEG-2 TM5 software to allow encoding with a constant quantization parameter throughout an entire sequence and Lagrangian-based rate-distortion optimized operational control was added.

Standard	**Implementation**	**Details**
JPEG	Independent JPEG Group, version 6b	Public software [61].
JPEG 2000	JPEG 2000 Verification Model (VM) 8.6	Developed by and available to JPEG committee members.
MPEG-2	MPEG Software Simulation Group version 1.2, with added rate-distortion optimized motion search and mode decision	Public software [62].
H.263	University of British Columbia Signal Processing and Multimedia Group (UBC-SPMG), H.263 code library version 0.3	Available to ITU-T members and research organizations [63].
MPEG-4	UB Video's *UB-Stream* version 2.0	Used to generate the MPEG-4 anchors in MPEG's recent video coding efficiency tests. See http://www.ubvideo.com.
H.26L	ITU-T VCEG TML 8.5 implementation	Developed by ITU-T VCEG members, largely by the Heinrich-Hertz-Institute [64].

Table 4-1: Software implementations of video encoders used in performance comparisons.

4.1.1 Test Sequences

The set of test sequences used in these comparisons is listed in Table 4-2, along with a brief description of each sequence. A sample frame from each sequence is given in Appendix A. All sequences are CIF resolution (352 x 288 luminance pixels) and use the YUV 4:2:0 color format, in which each of the two chrominance components are downsampled by a factor of two in each spatial direction. The sequences are all well-known test sequences that are used in the video standards community and represent a wide variety of video content. Appropriate subsets of these sequences will be selected based on the applications targeted in each of our comparisons.

Name	Resolution	Frames	Characteristics
Akiyo	CIF	300	Low-activity head and shoulders content
Bus	CIF	150	Fast translational motion and camera panning; moderate spatial detail
Carphone	CIF	300	Shaky still camera on human subject; partial background movement
Flower Garden	CIF	250	Slow and steady camera panning over landscape; spatial and color detail
Foreman	CIF	300	Shaking, hand-held camera, fast panning
Mobile and Calendar	CIF	300	Slow panning and zooming; complex motion; high spatial and color detail
Paris	CIF	300	Still camera on human subjects; typical videoconferencing content
Silent Voice	CIF	300	Still-camera on subject using sign-language
Tempete	CIF	260	Camera zoom; spatial detail; fast random motion
Trailblazers	CIF	300	Basketball scene with fast action and heavy camera movement

Table 4-2: Video sequences used in performance comparisons.

4.1.2 Measuring Performance

Since it is the most widely accepted objective measure of visual distortion, PSNR is our primary means of measuring visual distortion. For each test case and sequence, results are presented in a set of rate-distortion curves, with one curve for each encoder being evaluated. A curve is generated by encoding each sequence several times with different quantization step sizes, which are held constant throughout each coding pass. The average PSNR for each of the three components over all of the frames in the sequence is recorded along with the average bit rate. In the distortion measurement, the PSNR for the luminance and each of the two chrominance components must be taken into consideration. This is particularly important when comparing different standards to each other, since the relative fidelity of luminance and chrominance components may differ from one standard to the next, however both are important subjectively. To this end, we have experimentally determined that a weighted PSNR measure in which the luminance component is given 8 times the weight of each of the chrominance components is suitable for comparisons between different video standards. The emphasis on luminance is derived from its greater visual importance and the relative bit expenditures for luminance and chrominance components that are found in video encoders. Thus, we define the weighted PSNR as

$$Weighted\ PSNR = \frac{8 \cdot PSNR_Y + PSNR_U + PSNR_V}{10}.$$

Differences in the weighted PSNR versus bit rate curves will be summarized using the Delta-SNR method that has been adopted by the ITU-T VCEG for expressing differences in coding efficiency between two rate-distortion curves [67]. These results are presented both in terms of the average bit savings and the average PSNR difference between two curves.

4.2 Experimental Results

In our comparisons of the different standards, we perform three separate experiments, each targeting a particular application area. The first experiment compares intra coding efficiency between both video and still image coding standards. The second experiment evaluates performance for interactive video applications, such as videoconferencing, in which minimal delay in the encoder-decoder loop is a key requirement. The delay constraints are relaxed in the third experiment, which addresses the video coding for non-interactive applications (e.g. streaming).

4.3 Intra Coding

An evaluation of intra coding rate-distortion performance provides an opportunity to analyze the capabilities of the different spatial coding methods that exist in both still image and hybrid video coding standards. In this experiment, the intra coding performance of encoders that are compliant with Baseline JPEG, JPEG 2000, H.263, MPEG-4 and H.26L are evaluated. In order to have a sufficient variety of content and to simulate typical use of intra coded pictures in video applications, every 30th frame of several 30 Hz CIF resolution video sequences was encoded. For this comparison, we use the sequences Bus, Flower Garden, Mobile and Calendar, Paris, and Tempete. For the still image coders, individual frames were extracted from the original video sequences and converted to the file format required by each encoder implementation (i.e. portable pixmap (PPM) for JPEG and three portable greymap (PGM) components for JPEG 2000). Bit rate results for this experiment are expressed in average bits per frame over all frames encoded in each sequence.

All of the encoders were configured to provide their best possible PSNR performance for intra coding. For JPEG, this includes using the optimized VLC coding option and flat quantization tables, which improve the PSNR performance significantly. The JPEG 2000 encoder was modified so that the base step size for chrominance components was 60%

larger than the step size for luminance. This adjustment was made so that the chrominance PSNR produced by JPEG 2000 was similar to that of the best video coders, facilitating a direct comparison of the standards based primarily upon luminance PSNR values. The H.263 encoder used Annexes I and T (advanced intra coding mode and modified quantization mode, respectively), which improve intra coding performance. The H.263-style quantization method was used by the MPEG-4 encoder, since this provides better PSNR performance than the MPEG-style method. Finally, context-based arithmetic coding was used to generate the H.26L bitstreams, as it provides more efficient entropy coding than the universal VLC method.

Some typical intra coding rate-distortion curves that illustrate the average bits per frame versus the average weighted PSNR for each of the test sequences are shown in Figure 4-1. The average bit savings and PSNR gain provided by each encoder relative to all of the other encoders are summarized in Table 4-3. As an example of how to read this table, the H.26L encoder provides an average bit savings of 11.25% and an average weighted PSNR gain of 0.99 dB relative to the MPEG-4 encoder.

	Average bit savings and PSNR gain relative to:			
Encoder	**JPEG 2000**	**H.263**	**MPEG-4**	**JPEG**
H.26L	-0.13% -0.00 dB	9.66% +0.94 dB	10.58% +0.98 dB	19.41% +1.84 dB
JPEG 2000	-	10.02% +1.05 dB	10.42% + 0.99 dB	19.67% +1.90 dB
H.263	-	-	0.56% +0.05 dB	10.84% +0.94 dB
MPEG-4	-	-	-	9.64% +0.86 dB

Table 4-3: Relative intra coding performance of encoders over entire test set.

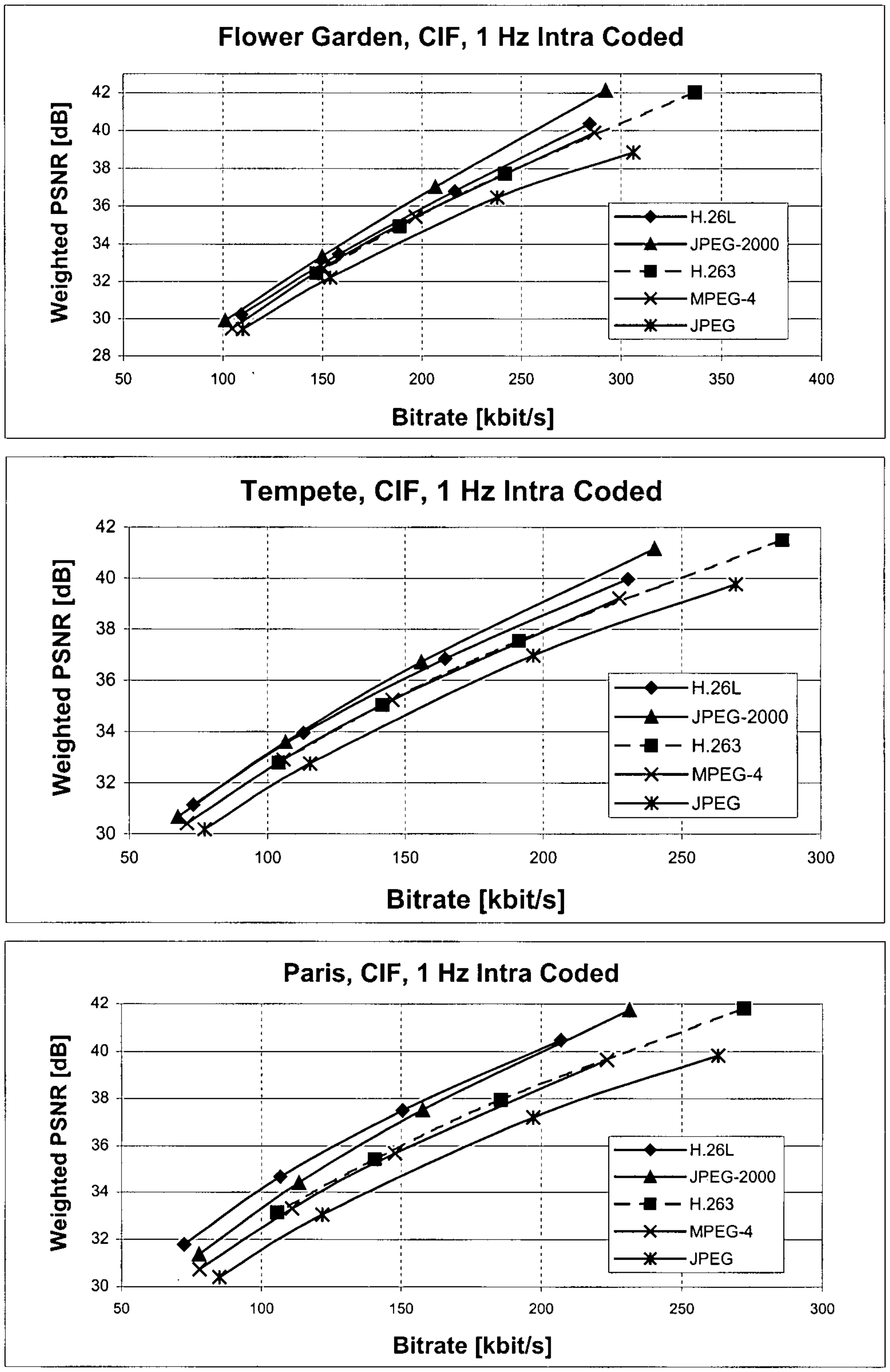


Figure 4-1: Sample rate-distortion curves for intra coding test.

Our results show that – although they are based on fundamentally different coding algorithms – H.26L and JPEG 2000 offer nearly identical average rate-distortion performance, with JPEG 2000 outperforming H.26L by a very small margin. Both provide significant gains in coding efficiency over earlier standards, approximately 20% relative to JPEG and 10% relative to MPEG-4 and H.263. While their average performance levels over our test set are similar, content and bit rate dependent differences in coding efficiency clearly exist between H.26L and JPEG 2000. One key observation is that the performance of JPEG 2000 tends to improve relative to that of H.26L as the bit rate is increased on each sequence. Furthermore, H.26L outperforms JPEG 2000 significantly on the Paris sequence, which contains a large amount of strong directional edges, while JPEG 2000 does best on Flower Garden, which features softer and more random edges. These performance differences originate in the use of the wavelet transform versus spatial prediction and the DCT-like transform.

The similar intra coding performance of these two standards suggest that H.26L might be suitable for use in the intraframe video coding applications that are targeted by Motion JPEG 2000. Since they can provide nearly identical rate-distortion performance, the choice of one standard versus the other for a particular application would likely be determined by constraints other than coding efficiency. For example, JPEG 2000 supports a diverse set of features for still image coding, such as random access, progressive compression, compressed domain manipulations, and region-of-interest coding. High-quality video editing systems could benefit from such functionality. However, if these features are not necessary, H.26L might be preferred since it is composed of relatively simple and widely implemented building blocks, thus implementation and computational complexity are reduced. Of course, in addition to very efficient intraframe coding, H.26L has the advantage of supporting a fully featured hybrid video coding syntax, if the option for such added functionality is desired.

While the similar performance of H.26L and JPEG 2000 is not necessarily expected, it is expected that H.263 and MPEG-4 offer nearly identical performance to each other in this experiment. These standards use identical block sizes and transforms, and very similar prediction techniques and quantization methods for intra blocks. However, at low bit rates, H.263 tends to provide improved chrominance fidelity, but MPEG-4 has better luminance fidelity, because the two standards specify different ratios between the quantization step sizes for luminance and chrominance coefficients. Both standards offer improved intra coding performance versus JPEG primarily because they include prediction of AC coefficients from neighboring blocks, while JPEG only supports prediction of the DC coefficient.

There are a few key technical features that contribute to the superiority of H.26L over the earlier DCT-based coders. These include the use of smaller transform blocks (4x4 instead of 8x8), a larger number of prediction modes, spatial-domain prediction of all pixel values, and context-based arithmetic coding. These improvements in intra coding efficiency come at the cost of increased complexity relative to H.263 and MPEG-4.

4.4 Video Coding for Interactive Applications

Our second experiment evaluates coding performance for interactive conversational video applications, such as videoconferencing, in which minimal delay and real-time encoding capability are the key requirements. These applications generally support low to medium bit rates and picture resolutions, with CIF resolution at 128-512 Kbit/s being the most common operating point. For this test, we select a set of sequences that represent typical for conversational video applications. Specifically, we choose Akiyo and Paris, sampled at 15 frames per second, and Carphone, Foreman, and Silent Voice at their original frame rate of 30 frames per second. Five features sets are included in this comparison: the H.263 Baseline and Conversational High Compression (CHC) Profiles, the MPEG-4 Simple and Advanced Simple Profiles, and H.26L. Since profiles are not yet defined for the ongoing H.26L project, this encoder is configured to provide its best possible rate-distortion performance while excluding any features that would increase delay in the encoder-decoder loop.

In all bitstreams, only the first picture was intra coded, with all of the subsequent pictures being temporally predicted (P-pictures). Both the H.263 CHC and H.26L encoders used five reference pictures for long-term prediction. (This is the maximum number allowed for CIF sequences in Level 40 of H.263's normative profile and level definitions). A motion search range of 32 integer pixels was employed by all encoders with the exception of H.263 Baseline, which is constrained by its syntax to a maximum range of 16 pixels. The H.26L encoder also used the CABAC entropy coding mode and quarter-pixel accurate motion compensation, since eighth-pixel accuracy typically shows little or no improvement for the content and bit rates found in conversational applications.

Since profiles are used to indicate decoder support for a set of optional modes, an encoder that is compliant with a particular profile is permitted – but not required – to use any of the optional modes supported in a that profile. With this in mind, encoders were configured by only including the optional modes from each profile that would produce the best possible rate-distortion performance, while satisfying the low delay and complexity requirements of interactive video applications. Specifically, in the MPEG-4 Advanced Simple Profile bitstreams, B-pictures cannot be included because of the strict delay constraints of interactive applications and global motion compensation is not used due to its high complexity in both the encoder and decoder, and the minimal improvement that it can provide for the tested content. Also, the MPEG-style quantization is supported in the Advanced Simple Profile, but is not used because it does not improve PSNR-based results. Therefore, the only difference between the MPEG-4 Simple Profile and the Advanced Simple Profile results in this experiment is that the Advanced Simple Profile uses quarter-pixel accurate motion compensation, whereas the Simple Profile uses only half-pixel accuracy.

As in the first experiment, we present sample rate-distortion curves for each sequence using the weighted PSNR measure in Figure 4-2. The summary of Delta-SNR values that compare the coding gains of each standard to all of the other standards are found in Table 4-4.

	Average bit savings and PSNR gain relative to:			
Encoder	**H.263 CHC**	**MP4 ASP**	**MP4 SP**	**H.263 Base**
H.26L	24.08% +1.20 dB	28.34% +1.41 dB	33.48% +1.66 dB	42.14% +2.25 dB
H.263 CHC	-	4.25% +0.19 dB	12.18% +0.51 dB	23.51% +1.09 dB
MPEG-4 ASP	-	-	7.77% +0.33 dB	19.69% +0.89 dB
MPEG-4 SP	-	-	-	13.47% +0.56 dB

Table 4-4: Relative performance of encoders in interactive video coding comparison.

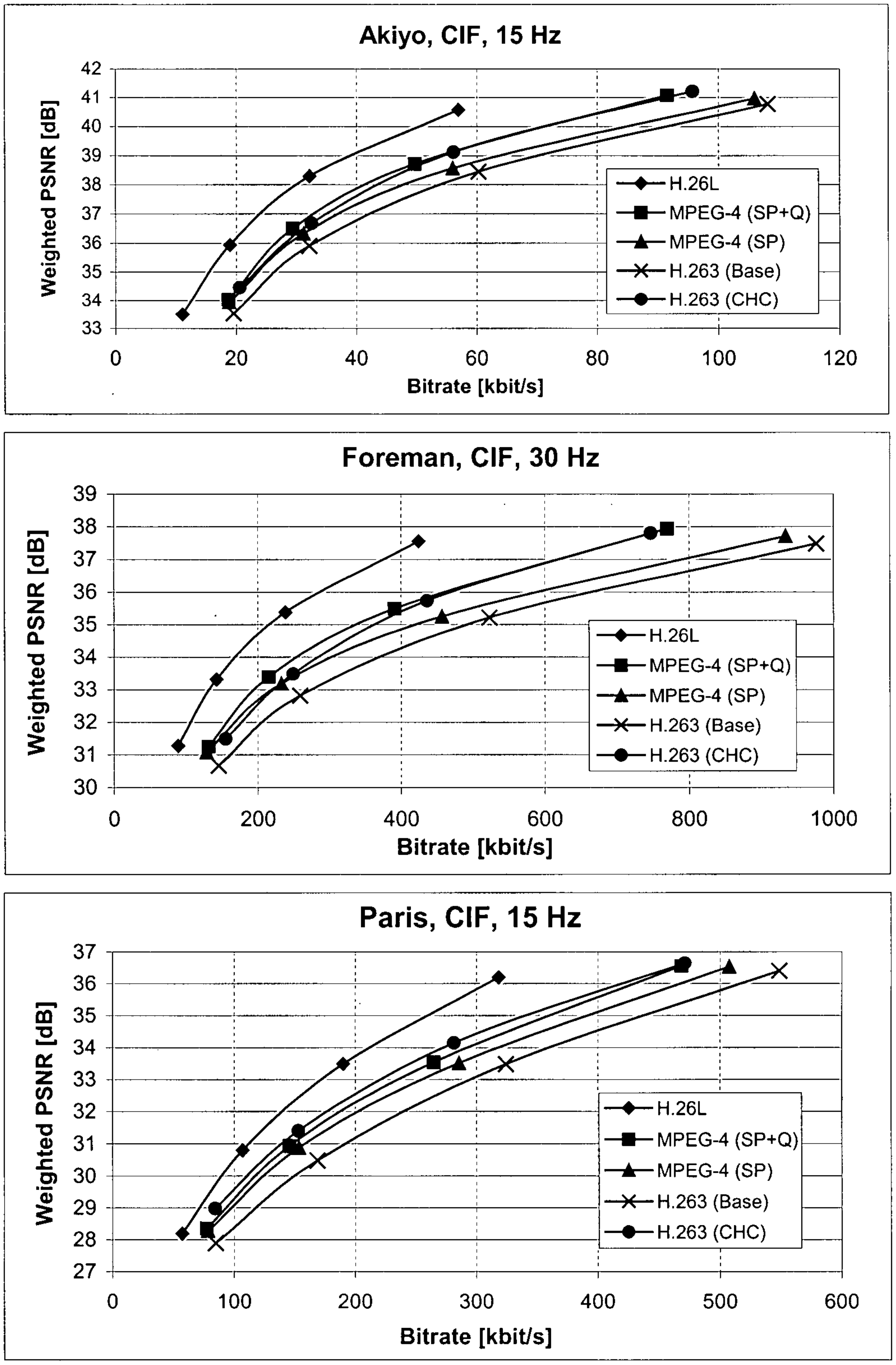


Figure 4-2: Sample rate-distortion curves for interactive coding test.

It is immediately clear from these results that the next-generation H.26L standard outperforms all of the other standards by a substantial margin. Over the entire test set, H.26L provides approximately 25% bit savings relative to its two nearest competitors, MPEG-4 Advanced Simple and H.263 CHC, and 42% bit savings over H.263 Baseline, which is the most common standard currently used in industry for such applications. Note that H.26L includes all of the main technical features used in these other encoder configurations, plus several additional features. We expect that the highly flexible motion model and the very efficient context-based arithmetic coding scheme are the two primary factors that enable the superior rate-distortion performance of H.26L. This will be further addressed in the next chapter in which the coding performance of several features of the H.26L algorithm is analyzed. The differences in rate-distortion performance between the various standards and profiles included in this experiment yield further insight into the gains in coding efficiency provided by some of their key features. For example, the MPEG-4 Simple Profile provides approximately 13% bit savings over H.263 Baseline. The technical features that contribute to this improvement include allowing motion compensation on 8x8 blocks, extrapolation of motion vectors over picture boundaries, and improved intra coding efficiency. The benefit of quarter-pixel accurate motion compensation can also be observed directly from these results by comparing the MPEG-4 Simple and Advanced Simple Profiles. The additional bit savings provided by allowing quarter-pixel accurate motion compensation in the Advanced Simple Profile test cases ranges from almost zero for the Silent sequence to 15% for Foreman. The most notable difference between these two sequences is that Foreman contains faster movement, more sharp edges, and is higher in spatial detail than Silent. Furthermore, it is evident from the

RD-curves that the benefit of using quarter-pixel accuracy increases with the average bit rate. For both Foreman and Akiyo, the coding gain from quarter-pixel is nearly zero at the largest tested quantizer value (lowest bit rate) but approximately 20% at the smallest quantizer value (highest bit rate). Lastly, the H.263 CHC profile offers slightly better performance, on average, than the MPEG-4 Advanced Simple Profile. The main technical difference between these two feature sets is that Advanced Simple includes quarter-pixel accurate motion compensation, which H.263 CHC does not, but H.263 CHC benefits from using multiple reference frames for motion compensation. H.263 also benefits from the improved chrominance fidelity that is realized by using Annex T.

If we compare these results to those of the intra coding experiment, it is evident that the potential gains in coding efficiency that can be realized by using a more recent and flexible standard are much greater when the temporal domain is added. This emphasizes the importance of the motion model for determining the limit of achievable rate-distortion performance in a video coder. The impressive performance of the H.26L compliant encoder in this experiment clearly demonstrates the potential importance of this standard in future applications of interactive video coding. However, a serious stumbling block is that the complexity of the fully rate-distortion optimized H.26L encoder is significantly larger than that of the other encoders. Enabling real-time encoding – which is necessary for such interactive applications – is a difficult challenge. Therefore, intelligent algorithms and implementations that reduce encoding complexity without sacrificing a large amount of rate-distortion performance will be required in order to enable widespread use of this emerging standard in such applications.

4.5 Video Coding for Streaming Applications

Our final inter-standard comparison addresses video coding for streaming and distribution applications, in which delay in the encoder-decoder loop is not a critical constraint. Specific applications addressed in this comparison include streaming of video over the Internet, video-on-demand, and home-entertainment applications such as digital cable and DVD, although the latter examples usually use higher resolutions and bit rates than those included in this test. However, the comparisons performed with CIF content in this test will provide a reasonable indication of how these standards would compare for higher quality content. Similar comparisons are described in [65] and [66].

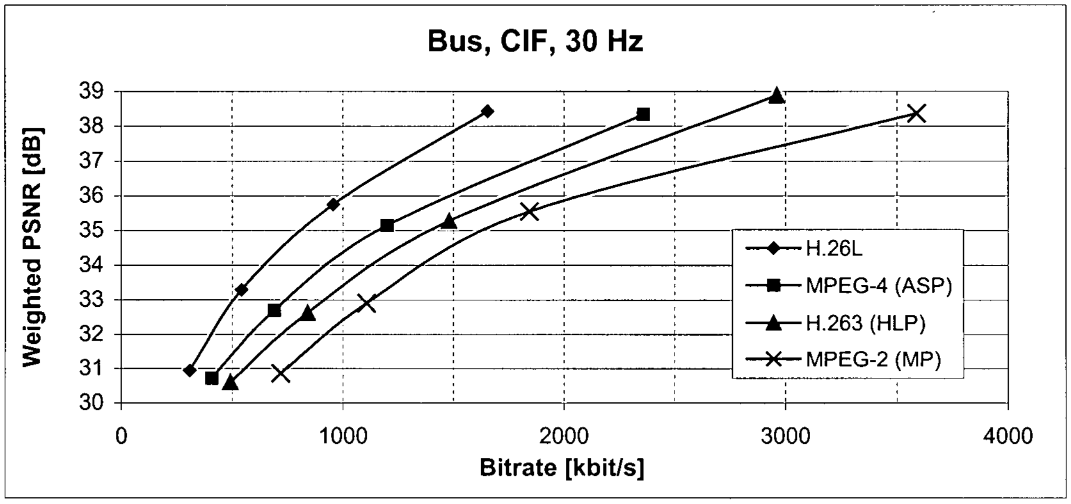
From our set of test sequences in Table 4-2, we select a relatively difficult set of sequences that is representative of what is typically used in streaming applications: Bus, Flower Garden, Mobile and Calender, Tempete and Trailblazers, all at CIF resolution and 30 frames per second. Using this content, we compare the performance of the MPEG-2 Main Profile, the MPEG-4 Advanced Simple Profile (ASP), the H.263 High-Latency Profile (HLP) and H.26L. In terms of coding features, the main technical difference from the interactive test is the insertion of two B-pictures between each pair of reference frames in all of the encoded bitstreams. This generally improves coding efficiency at the cost of added delay and complexity, which are not so critical in streaming applications. As in the previous test, the MPEG-4 ASP encoder did not use its global motion compensation feature, since it provides little or no benefit for this content with a large increase in complexity. Both H.26L and H.263 make use of 5 prior reference frames for prediction, in addition to the one subsequent reference frame that is used in predicting B-pictures. The H.26L encoder also employed eighth-pixel accurate motion compensation and CABAC entropy coding to increase coding efficiency.

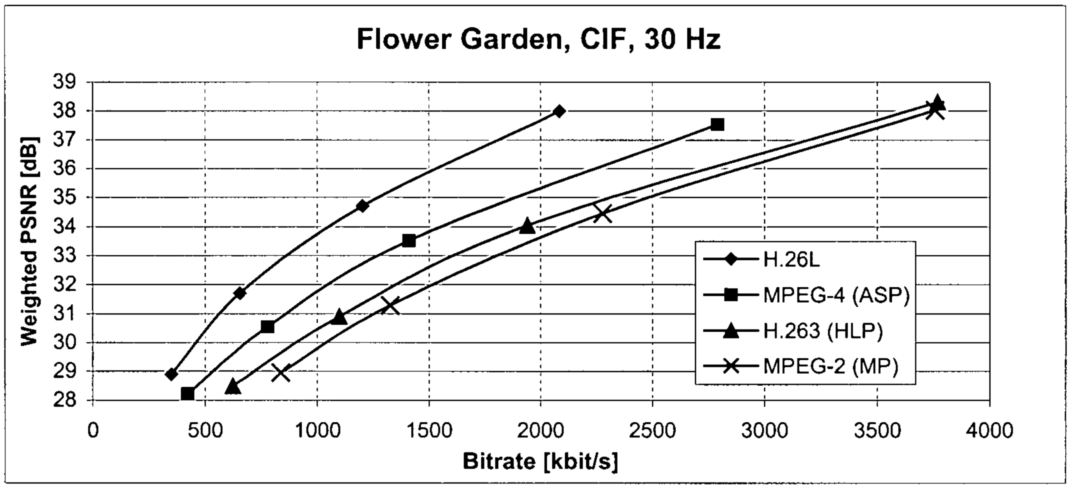
The quantization step size for B-pictures was selected to be approximately 12% larger than the P-picture step size, although this becomes difficult with low integer quantization parameter values. Note that for H.26L, each increment by 1 in the quantization parameter produces a step size that is approximately 12% larger. The quantization values for this test are smaller than those used in the interactive coding test, since streaming applications generally require higher visual quality than interactive applications.

The rate-distortion curves for the 5 test sequences are given if Figure 4-3 and a Delta-SNR-based summary of the differences in coding efficiency between all of the standards is provided in Table 4-5.

	Average bit savings and PSNR gain relative to:		
Encoder	**MP4 ASP**	**H.263 HLP**	**MPEG-2**
H.26L	33.86% +1.79 dB	43.81% +2.63 dB	53.04% +3.77 dB
MPEG-4 ASP	-	15.75% +0.82 dB	30.54% +1.85 dB
H.263 HLP	-	-	17.32% +0.96 dB

Table 4-5: Relative performance of encoders in streaming video test.





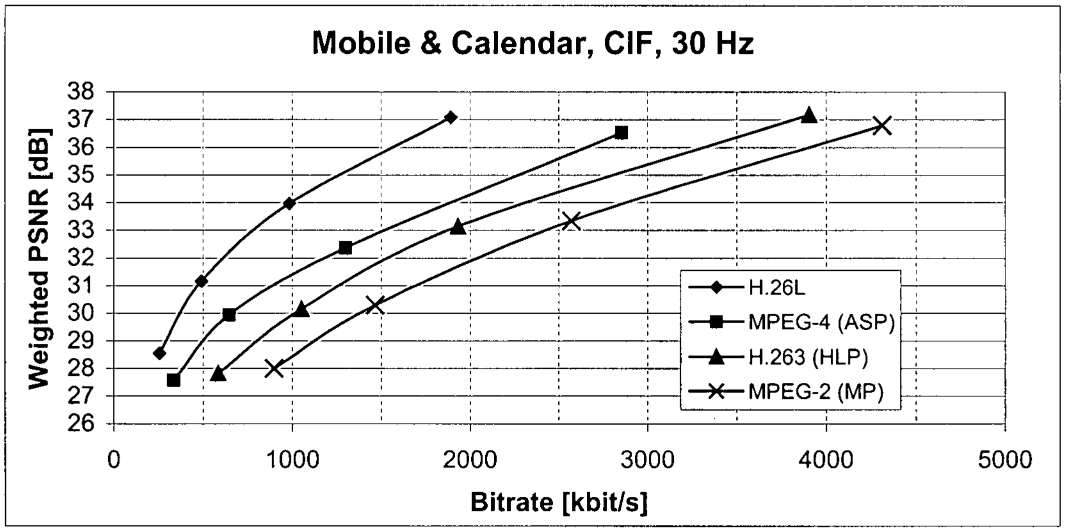


Figure 4-3: Sample rate-distortion curves for streaming video test.

As in the previous test, the impressive performance of the H.26L encoder stands out. Compared to the currently popular MPEG-2 standard, H.26L provides an average bit savings of greater than 50% or a 3.5 dB PSNR improvement. Although MPEG-2 is generally used for higher resolution content, these very large improvements on CIF content offer a good indication that H.26L would outperform MPEG-2 significantly at higher resolutions. Again, the flexible motion compensation model and the efficient CABAC entropy coding are key advantages for H.26L over the other standards.

Another interesting observation is that the MPEG-4 encoder outperforms the H.263 encoder in this test, but in the previous test, in which approximately the same two configurations were tested without B-pictures in either, H.263 was better. If we consider that the key difference between the two encoders is that MPEG-4 uses quarter-pixel motion compensation whereas H.263 uses multiple reference frame prediction, then these results suggest that the benefit of quarter-pixel motion compensation relative to that of multi-frame prediction is larger for streaming than for interactive applications. The diminished performance of multi-frame prediction relative to quarter-pixel accurate motion in the presence of B-pictures makes some intuitive sense, since the reference frames are spaced further apart by the insertion of 2 B-pictures between each pair of reference frames. This should lead to less temporal correlation between the current frame and the reference frames. However, the higher spatial accuracy of quarter-pixel compensation does not suffer from this effect when B-pictures are introduced. Moreover, the higher bit rates and more difficult content used in this test should also contribute to the superiority of MPEG-4 over H.263 in this test. As we observed in the previous test, allowing higher spatial accuracy for motion compensation provides greater benefit at higher bit rates. Finally, the content used in this test is much higher in spatial detail than the content used in the interactive coding test, particularly for the Flower Garden and Mobile and Calendar sequence. Higher motion compensation accuracy has been shown to be most valuable for content that is high in spatial detail.

4.6 Conclusions

The experimental results presented in this chapter illustrate that H.26L can provide significantly improved coding efficiency relative to all other video coding standards in each of the application spaces addressed in our tests. Bit rate savings of more than 50% were demonstrated versus the H.263 Baseline and MPEG-2 standards that are currently popular in industry. The ability to produce equivalent visual quality at half of the bit rate can be a powerful tool in enabling new video-based applications over bandwidth-limited channels, and making existing applications more cost effective (e.g. the ability to deliver twice as many video channels on the same amount of bandwidth). However, a key issue that remains is the very large amount of computational complexity that exists in a rate-distortion optimized H.26L encoder. For example, the public TML-8.5 software used in these comparisons encodes approximately 2 CIF resolution frames per minute on the fastest PCs. The question of whether or not the coding gains realized by H.26L can be achieved with real-time or near-real-time encoding still must be addressed. Another intriguing question that is raised by the results presented in this chapter is: Which features of H.26L produce its impressive gains in coding efficiency? The source of coding improvements in H.26L is analyzed in the following chapter. This analysis leads into the development of the fast H.26L encoding algorithms that will illustrate that the compression benefits of H.26L can be realized in a real-time encoding scenario.

5 Performance Analysis of the Draft H.26L Standard

In this chapter, we provide a detailed analysis of the coding gains provided by several of the key coding features of the draft H.26L standard. The goal is to establish the characteristics of the coding gain that each feature can provide for typical video content. This is the first step in determining the encoding complexity versus coding efficiency tradeoffs that can be achieved in an efficient H.26L encoder. Complexity considerations for each of these features are also discussed.

5.1 Experimental Procedure

We evaluate the performance of the two entropy coding methods available in the current draft as well as several components of the H.26L motion compensation model. These components include various combinations of block sizes for motion compensation, the spatial motion compensation accuracy, multiple reference frame prediction, and B-pictures. Additionally, the coding gain from using multiple reference frames for prediction in H.26L is compared to the gain provided by enabling similar functionality in H.263 using Annex U of that standard. For further results, see [68].

Since our goal is to establish the optimal coding gain provided by each of the features being tested, the rate-distortion optimized mode of the TML-8.5 software was used. The default configuration used throughout these experiments includes 5 reference frames for prediction, UVLC coding, quarter-pixel accurate motion compensation, all 7 motion compensation block types, and no B-pictures. A full search range of 32 pixels from the predictor was only used for the experiments that evaluate multi-frame prediction and B-pictures, where a smaller search range might limit the use of temporally distant frames. In the other experiments, a search range of only 16 pixels was used, since the smaller search range should not provide any significant advantage or disadvantage to the features being tested. Results were collected for each sequence using quantization parameter values of 16, 20, 24 and 28, as specified in the H.26L common conditions for coding efficiency tests [69]. As in the previous set of comparisons, coding efficiency improvements for each feature are summarized using the Delta-SNR method and several sample rate-distortion curves are also presented. In contrast to the experiments in the previous chapter, all results in this chapter are presented based only upon the luminance PSNR, rather than a weighted PSNR. It is reasonable to use only the luminance PSNR in the analysis of H.26L features because all tests are being conducted on a single standard, and none of the features tested modifies the luminance-chrominance tradeoffs. Thus, the luminance PSNR is sufficient for illustrating differences introduced by the features of H.26L. This is the preferred method for presenting coding efficiency results within the ITU-T VCEG that is developing H.26L.

For this analysis, we use a relatively large set of content in order to ensure that our results are representative of general video content at CIF resolution and lower. The large number of sequences can also establish any significant dependence of the performance of coding features on the characteristics of the content and the source resolution. Details of the sequences used in the H.26L analysis are given in Table 5-1. The majority of the sequences are popular test sequences used in video standardization. The PI sequence represents typical web-conferencing content captured with a low-end desktop camera. It contains two people talking and gesticulating relatively close to the camera. See Appendix A for sample frames of all test sequences used in this thesis.

Sequence	Format	Frame Rate	No. Frames Coded
Container Ship	QCIF	10 fps	100
Foreman	QCIF	10 fps	100
News	QCIF	10 fps	100
Silent Voice	QCIF	15 fps	150
PI	320x240	15 fps	400
Paris	CIF	15 fps	150
Silent Voice	CIF	15 fps	150
Bus	CIF	30 fps	150
Coastguard	CIF	30 fps	300
Flowergarden	CIF	30 fps	250
Foreman	CIF	30 fps	300
Mobile & Calendar	CIF	30 fps	300
Tempete	CIF	30 fps	260
Trailblazers	CIF	30 fps	300

Table 5-1: Video sequences used in analysis of H.26L coding features.

5.2 Motion Compensation Block Sizes

For the purpose of motion compensation, each 16x16 macroblock can be partitioned in one of seven ways in H.26L, from using a single motion vector for the entire block to using sixteen individual motion vectors for each of the 4x4 blocks that compose a macroblock (refer to Figure 3-2). Smaller blocks are intended to improve motion representation, especially in areas with fine motion detail, at the cost of larger overhead for transmitting motion vectors in the bitstream.

Supporting a large number of different block sizes – particularly smaller block sizes – increases algorithmic and computational complexity in both the encoder and decoder. Modern processors can deal more efficiently with larger blocks because they allow for a larger amount of data to be processed in the same way, reducing the overhead for looping and branching. For example, the interpolation of half-pixel values for a macroblock coded in the 4x4 mode requires 66% more operations than if the macroblock is coded in the 16x16 mode, and 37% more operations than for the 8x8 mode [70]. Furthermore, support for a larger number of block sizes directly increases the computational complexity of the motion estimation process in the encoder by increasing the number of possible options that can be tested when searching for the best match for macroblock.

In our comparisons, smaller block sizes are added incrementally in logical groupings. We also include the combination of 16x16 and 8x8, which are the two sizes found in the currently popular H.263 and MPEG-4 standards. Details of the block size combinations used are shown in Table 5-2. Other encoder settings for this experiment include the use of five reference frames, UVLC entropy coding and quarter-pixel motion vector accuracy.

Code	**Block types used**
1	16x16
1, 4	16x16, 8x8
1 to 3	16x16, 16x8, 8x16
1 to 4	16x16, 16x8, 8x16, 8x8
1 to 6	16x16, 16x8, 8x16, 8x8, 8x4, 4x8
1 to 7	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4

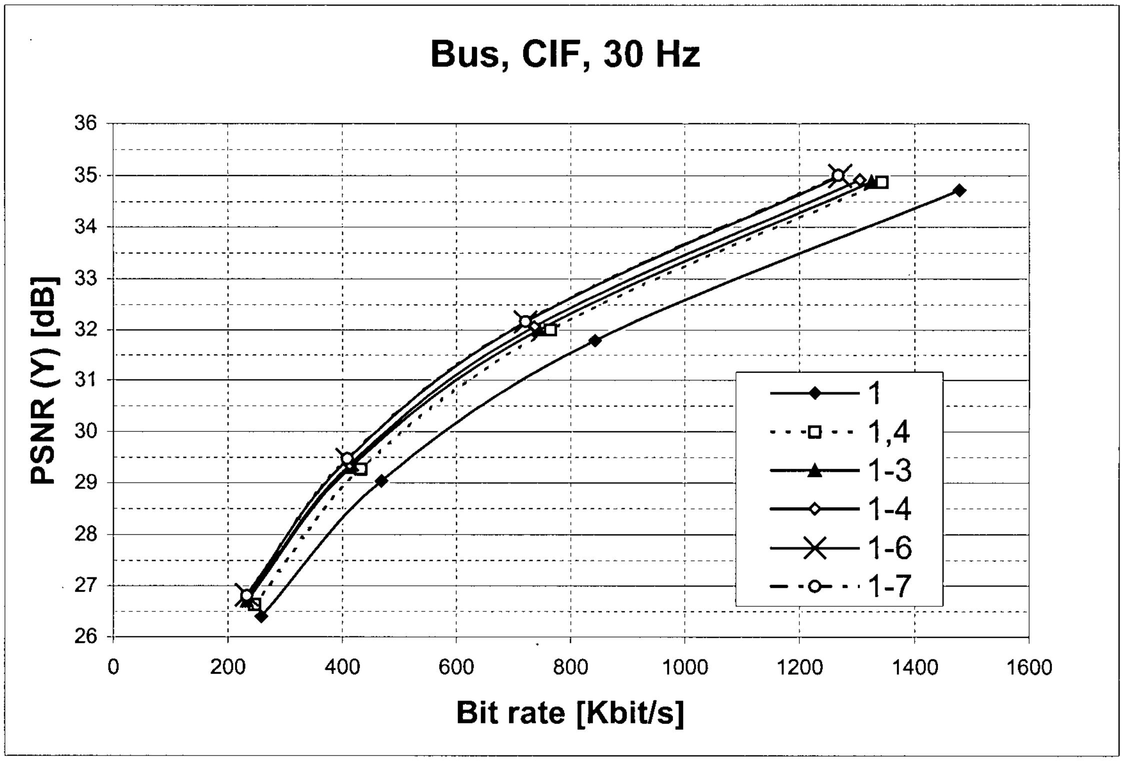
Table 5-2: Key to combinations of motion compensation block sizes.

Delta-SNR results comparing the benefit of the various combinations of block types versus using only 16x16 blocks for motion compensation only are presented in Table 5-3and sample rate-distortion curves are given in Figure 5-1.

			1, 4		**1 to 3**		**1 to 4**	
Res.	**Hz**	**Sequence**	**%Bits**	**PSNR**	**%Bits**	**PSNR**	**%Bits**	**PSNR**
QCIF	10	Container	13.60	0.699	15.76	0.807	16.79	0.872
		Foreman	8.94	0.505	12.22	0.682	13.49	0.774
		News	10.05	0.586	10.98	0.631	13.33	0.786
	15	Silent Voice	7.19	0.347	9.48	0.454	11.38	0.561
320x240	15	PI	12.44	0.703	13.61	0.758	16.27	0.925
CIF	15	Paris	12.75	0.677	13.44	0.700	15.61	0.831
		Silent Voice	4.56	0.191	7.73	0.325	8.34	0.354
	30	Bus	12.11	0.622	15.56	0.800	17.05	0.886
		Coastguard	6.72	0.243	9.67	0.351	10.67	0.391
		Flowergarden	14.39	0.804	16.26	0.906	17.43	0.980
		Foreman	7.69	0.374	12.00	0.589	12.50	0.620
		Mobile	10.28	0.500	11.55	0.561	13.04	0.641
		Tempete	9.20	0.395	12.19	0.529	12.91	0.565
		Trailblazers	8.39	0.400	11.13	0.522	12.75	0.605
		MEAN	**9.88**	**0.503**	**12.26**	**0.615**	**13.68**	**0.699**
		MIN	**4.56**	**0.191**	**7.73**	**0.325**	**8.34**	**0.354**
		MAX	**14.39**	**0.804**	**16.26**	**0.906**	**17.43**	**0.980**

			1 to 6		**1 to 7**	
Res.	**Hz**	**Sequence**	**%Bits**	**PSNR**	**%Bits**	**PSNR**
QCIF	10	Container	19.19	1.017	19.59	1.039
		Foreman	16.08	0.950	16.38	0.970
		News	16.12	0.984	16.42	1.013
	15	Silent	12.71	0.641	12.74	0.645
320x240	15	PI	19.21	1.130	19.51	1.153
CIF	15	Paris	19.02	1.048	19.21	1.069
		Silent	9.28	0.402	9.12	0.396
	30	Bus	19.80	1.056	20.00	1.072
		Coastguard	11.67	0.434	11.90	0.443
		Flowergarden	23.31	1.359	23.59	1.381
		Foreman	14.89	0.756	14.85	0.755
		Mobile	16.13	0.814	16.38	0.830
		Tempete	15.07	0.673	15.26	0.683
		Trailblazers	14.69	0.710	14.88	0.722
		MEAN	**16.23**	**0.855**	**16.42**	**0.869**
		MIN	**9.28**	**0.402**	**9.12**	**0.396**
		MAX	**23.31**	**1.359**	**23.59**	**1.381**

Table 5-3: Delta-SNR summary for motion compensation block sizes. Coding gains are relative to using the 16x16 motion compensation mode only.



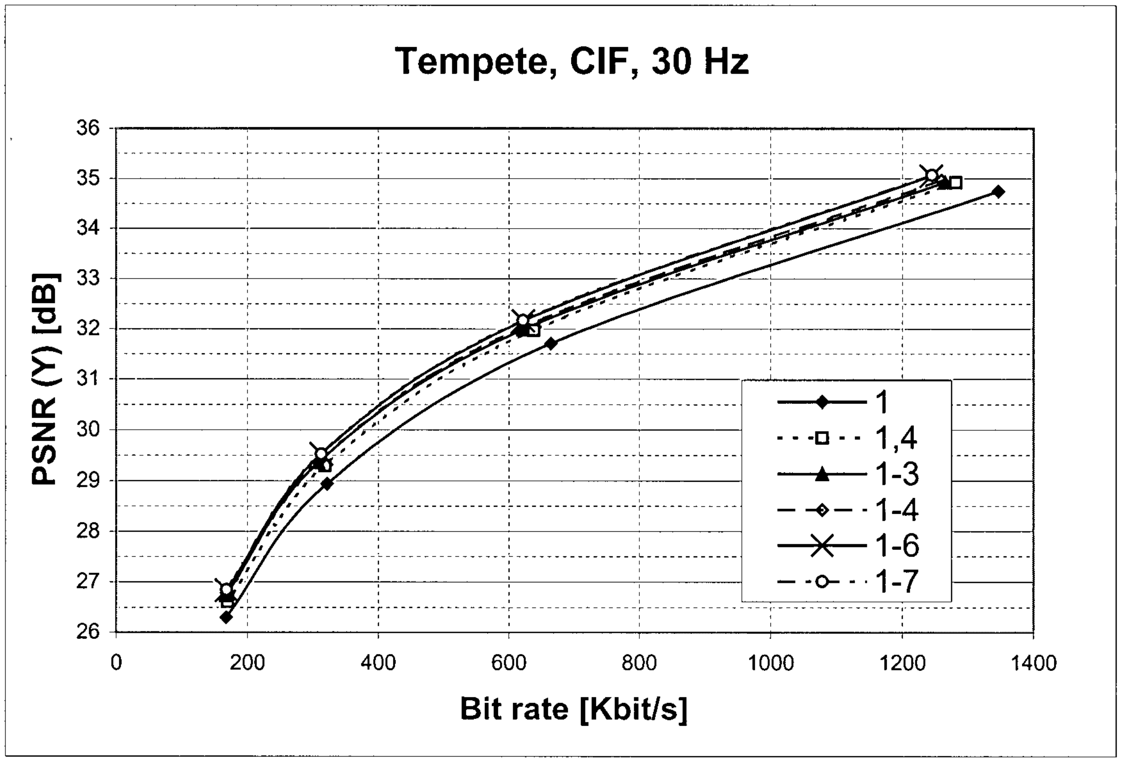


Figure 5-1: Sample rate-distortion curves illustrating the performance of different combinations of motion compensation block sizes in H.26L.

This experiment reveals that using all 7 motion compensation block types provides approximately 16% bit savings versus using 16x16 blocks only. However, we observe that most of this gain can be captured by using only the larger block sizes. By allowing only blocks that are 8x8 and larger (modes 1-4), greater than 80% of the bit savings realized by allowing all of the block sizes can be captured. And even if the 8x8 block type is not used so that only modes 1-3 are allowed, there is little loss in coding efficiency, particularly at lower bit rates. The block sizes smaller than 8x8 tend to be useful only at relatively high bit rates, and the 4x4 block size, which is the most computationally complex, provides minimal PSNR improvement for all of the tested content. Finally, our results suggest that smaller block sizes provide less benefit as the resolution of the source content is increased. In particular, the benefit of adding more motion compensation block types for the Silent Voice sequence is smaller at CIF resolution than at QCIF. In summary, the results of this test indicate that the performance versus complexity tradeoff in an H.26L encoder can be improved if the use of block sizes of 8x8 and smaller is limited.

5.3 Multiple Reference Frames

Allowing motion compensated predictions to come from more than only the most recent temporally previous reference frame has been shown to enable significantly improved coding efficiency for many types of content [16]. The idea is to add a temporal component to the motion vectors, permitting the selection of one of several reference frames at the macroblock level. In terms of complexity, memory requirements are increased in both the encoder and decoder in order to maintain a buffer of several previous reference frames. Furthermore, the computational complexity of the encoder's motion estimation process may increase significantly as additional reference frames are included in the search space.

In these tests, we only consider the performance of a sliding window reference frame buffer, in which only the N most recent reference frames are stored and available for use in motion compensated prediction. Support for an adaptive buffering mechanism, in which specific reference frames could be stored for an arbitrary amount of time (with a constraint on the total number of frames in the buffer) has been adopted in principle by VCEG for inclusion in H.26L, but is not included in the current draft or reference software.

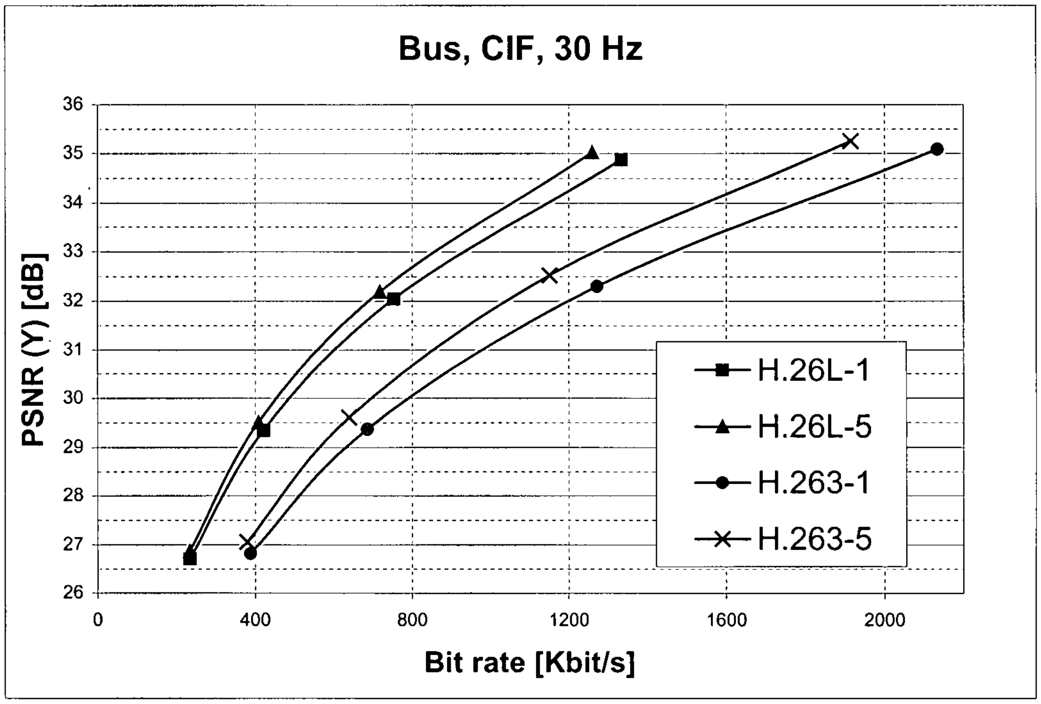
In this experiment, the sequences were coded once using only a single reference frame, and once with five reference frames. UVLC coding and quarter-pixel motion compensation were used and the search range was set to 32 pixels from the predictor. In an attempt to capture background uncovering, we performed some experiments in which the (0,0) motion vector was always included in the search area. However, with a large search range of 32 pixels, we found that this provided no significant benefit for this content.

This experiment also provides an opportunity to compare the benefit of using multiple reference frames in H.26L and H.263. For H.263 results, UBC's RD-optimized H.263 encoder [63] was used with Annexes D, F, I, J, T, and, optionally, Annex U. A full search range of 32 pixels was used for H.263 with quantization parameter values of 5, 8, 13 and 21. Summary Delta-SNR results for multi-frame prediction are presented in Table 5-4. Sample RD-curves are given in Figure 5-2.

			H.26L		H.263		H.26L/H.263	
Res.	**Hz**	**Sequence**	**%Bits**	**PSNR**	**%Bits**	**PSNR**	**%Bits**	**PSNR**
QCIF	10	Container	2.73	0.135	16.83	0.841	0.162	0.161
		Foreman	4.34	0.246	16.31	0.932	0.266	0.264
		News	0.95	0.055	2.06	0.107	0.461	0.514
	15	Silent Voice	4.07	0.199	7.58	0.362	0.537	0.550
320x240	15	PI	1.65	0.090	2.85	0.146	0.579	0.616
CIF	15	Paris	3.36	0.173	7.29	0.372	0.461	0.465
		Silent Voice	5.69	0.240	6.36	0.331	0.895	0.725
	30	Bus	6.76	0.336	11.82	0.628	0.572	0.535
		Coastguard	0.53	0.019	2.29	0.090	0.231	0.211
		Flowergarden	6.32	0.329	10.10	0.610	0.626	0.539
		Foreman	3.19	0.153	11.82	0.541	0.270	0.283
		Mobile	20.05	1.027	30.78	1.935	0.651	0.531
		Tempete	18.64	0.845	28.90	1.360	0.645	0.621
		Trailblazers	1.15	0.050	1.55	0.070	0.742	0.714
		MEAN	**5.67**	**0.278**	**11.18**	**0.595**	**0.507**	**0.481**
		MIN	**0.53**	**0.019**	**1.55**	**0.070**	**0.162**	**0.161**
		MAX	**20.05**	**1.027**	**30.78**	**1.935**	**0.895**	**0.725**

Table 5-4: Delta-SNR summary for multi-frame prediction in H.26L and H.263. The two right-most columns show the relative coding gain provided in H.26L compared to H.263.

These results illustrate that the performance of multi-frame prediction in H.26L is highly dependent on the characteristics of the source content. For the majority of sequences, average bit savings are less than 5%. Yet, two sequences – Mobile and Tempete – show savings around 20%, nearly 3 times greater than the gains shown for any of the other 12 sequences. This strong content dependency suggests that an intelligent encoder might benefit from recognizing content for which multiple reference frames show little benefit by limiting the searching of multiple reference frames in order to save processor cycles that could be better utilized for some other purpose. Or, perhaps a more complex adaptive buffering scheme should be implemented in place of the simple sliding window method in order to yield a greater benefit from multiple reference frame prediction.



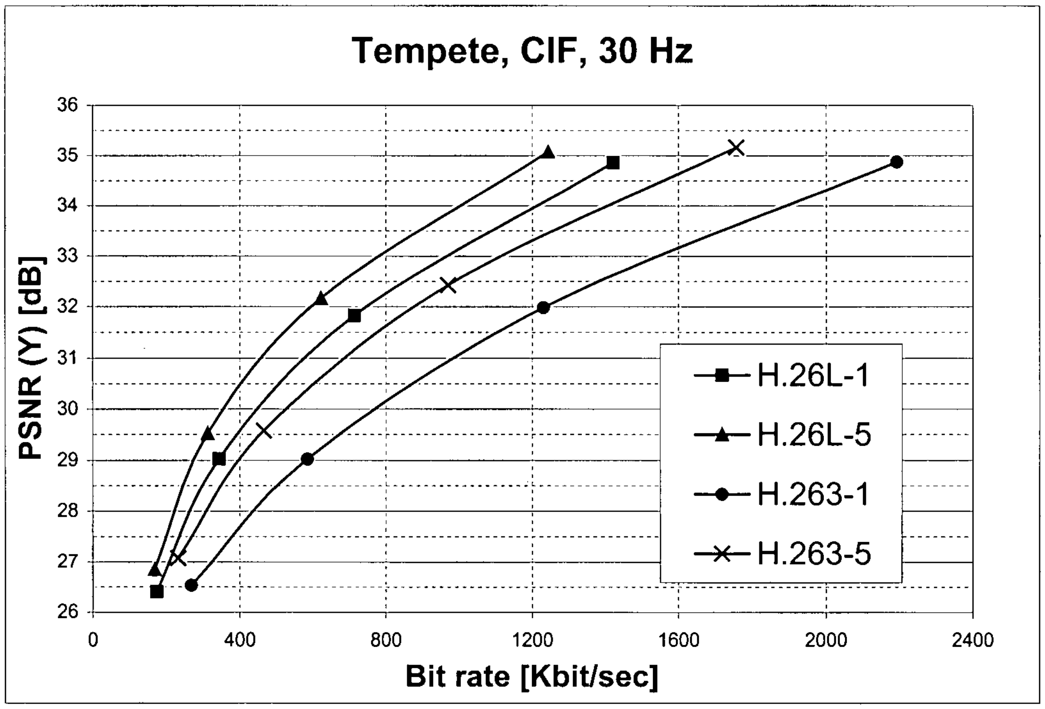


Figure 5-2: Sample rate-distortion curves illustrating the effect of using 5 previous reference frames in H.263 and H.26L.

Comparing H.26L and H.263, we observe that searching 5 frames in a sliding window buffer in H.26L provides approximately half of the gain that is provided by searching a similar buffer using Annex U in an H.263 encoder. The main reason for this difference is that other features of H.26L that are not in H.263, such as quarter-pel motion compensation and the availability of more block sizes, already capture some of the coding gain that Annex U can capture in H.263. Additionally, the macroblock level syntax for multi-frame prediction allows for less flexibility in H.26L, since the entire macroblock must be predicted from the same reference frame, whereas Annex U allows each 8x8 block to be predicted from a different frame. This may suggest that some improvements might be possible in the H.26L syntax to allow for more flexibility at the macroblock level.

One final interesting observation that can be drawn from the RD-curves is that H.26L outperforms H.263 significantly for all sequences in this set, with the notable exception of the Silent Voice, at both CIF and QCIF resolution. For this sequence, the RD-curves for the two standards almost overlap. Informal subjective testing has shown that H.26L offers a small subjective improvement over H.263 on this sequence, however. The main difference that is observed is that H.26L introduces less blurring, likely due to its improved deblocking filter, and perhaps the smaller block size used in the residual transform. Finally, we note that since the CABAC mode was not used for the H.26L results, further improvement in coding performance is achievable using H.26L.

5.4 B-Pictures

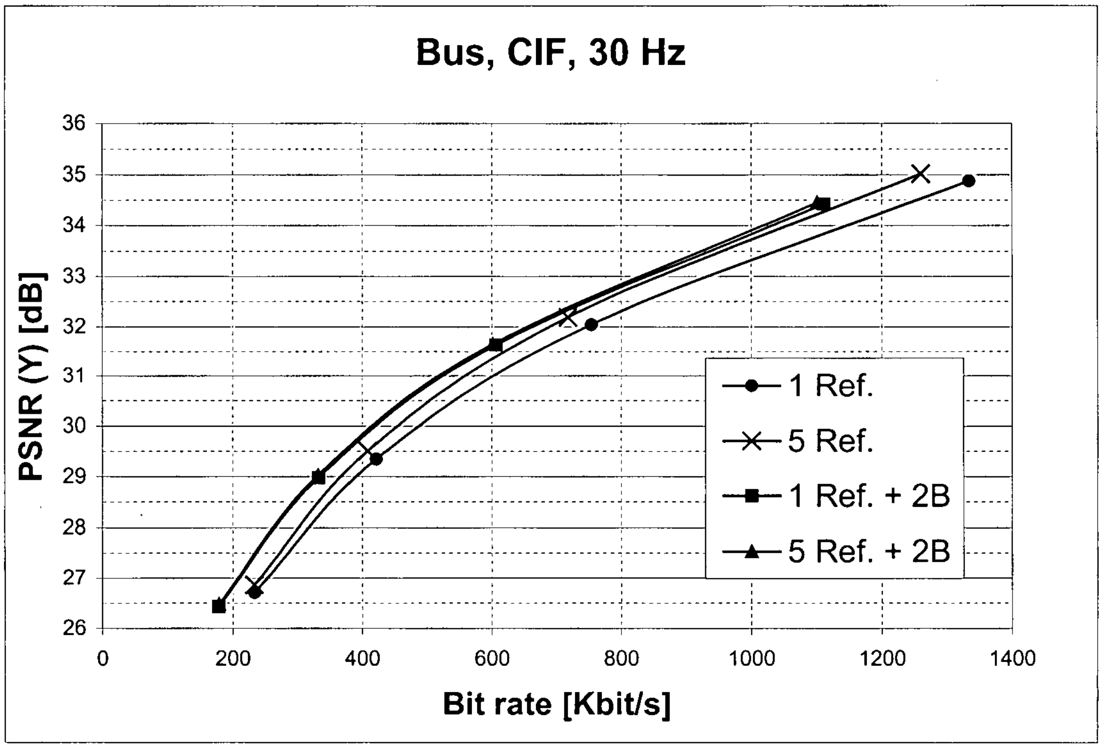
In B-picture encoding, motion compensated predictions can be derived from both temporally previous and temporally subsequent reference frames. This permits better predictions, leading to improved coding efficiency for most content. However, the re-ordering of frames in the bitstream that is necessary to enable bi-directional prediction introduces additional delay in the encoder-decoder loop, making B-pictures generally unacceptable in conversational video applications. Moreover, B-picture use increases memory requirements in the encoder and decoder in order to buffer the additional frames as they are encoded and decoded out of their original order. Motion estimation complexity is increased, due to the larger number of predictions that are possible when predictions are allowed in both temporal directions. Also, the motion compensation process that occurs in both the encoder and decoder is more computationally complex for certain coding modes within B-pictures that require the averaging of predicted blocks from two reference frames.

In this experiment, results are generated with 2 B-pictures inserted between each pair of reference frames (I- or P-pictures). The quantization parameter for the B-pictures is always one step larger than the parameter used for P-pictures, which results in a quantization step size that is approximately 12% larger in B-pictures than in P-pictures. Such a difference in quantization is typical when B-pictures are used, because slightly lower fidelity is acceptable for B-pictures since they are not used as reference frames. This experiment also provides an opportunity to assess any interactions that may exist between B-picture usage and multiple reference frame prediction. We expect some interaction because inserting B-pictures increases the temporal distance between reference frames. Thus, we expect less correlation to exist in the temporally distant reference frames when B-pictures are present. To perform this comparison, the sequences were coded once using only a single reference frame, and once with five reference frames. UVLC coding and quarter-pixel motion compensation were used and the search range was set to 32 pixels from the predictor.

Sample RD-curves for several sequences are given in Figure 5-3. Each plot contains 4 curves with B-pictures and multiple reference frames alternatively enabled and disabled. The Delta-SNR results showing the coding gain from inserting 2 B-pictures while using 1 and 5 reference frames are given in Table 5-5. Further results comparing the benefit of using 5 reference frames rather than only a single reference frame with and without 2 B-pictures already present in the bitstream are given in Table 5-6.

			2B with 1 Ref.		**2B with 5 Ref.**	
Res.	**Hz**	**Sequence**	**%Bits**	**PSNR**	**%Bits**	**PSNR**
QCIF		Container	24.49	1.37	21.84	1.214
		Foreman	-1.52	-0.08	-0.92	-0.049
		News	8.43	0.499	8.61	0.512
	15	Silent Voice	7.26	0.347	8.95	0.442
320x240	15	PI	0.81	0.045	2.16	0.118
CIF	15	Paris	4.55	0.236	6.62	0.344
		Silent Voice	6.81	0.286	8.77	0.375
	30	Bus	13.18	0.649	8.02	0.407
		Coastguard	15.90	0.579	15.70	0.580
		Flowergarden	16.52	0.846	13.45	0.708
		Foreman	12.70	0.615	10.96	0.577
		Mobile	34.61	1.813	23.63	1.261
		Tempete	28.23	1.273	20.56	0.936
		Trailblazers	-4.57	-0.196	-5.33	-0.229
		MEAN	**11.96**	**0.592**	**10.22**	**0.514**
		MIN	**-4.57**	**-0.196**	**-5.33**	**-0.229**
		MAX	**34.61**	**1.813**	**23.63**	**1.261**

Table 5-5: Delta-SNR summary of coding gain provided by adding 2 consecutive B-pictures to bitstreams using 1 and 5 prior reference frames.



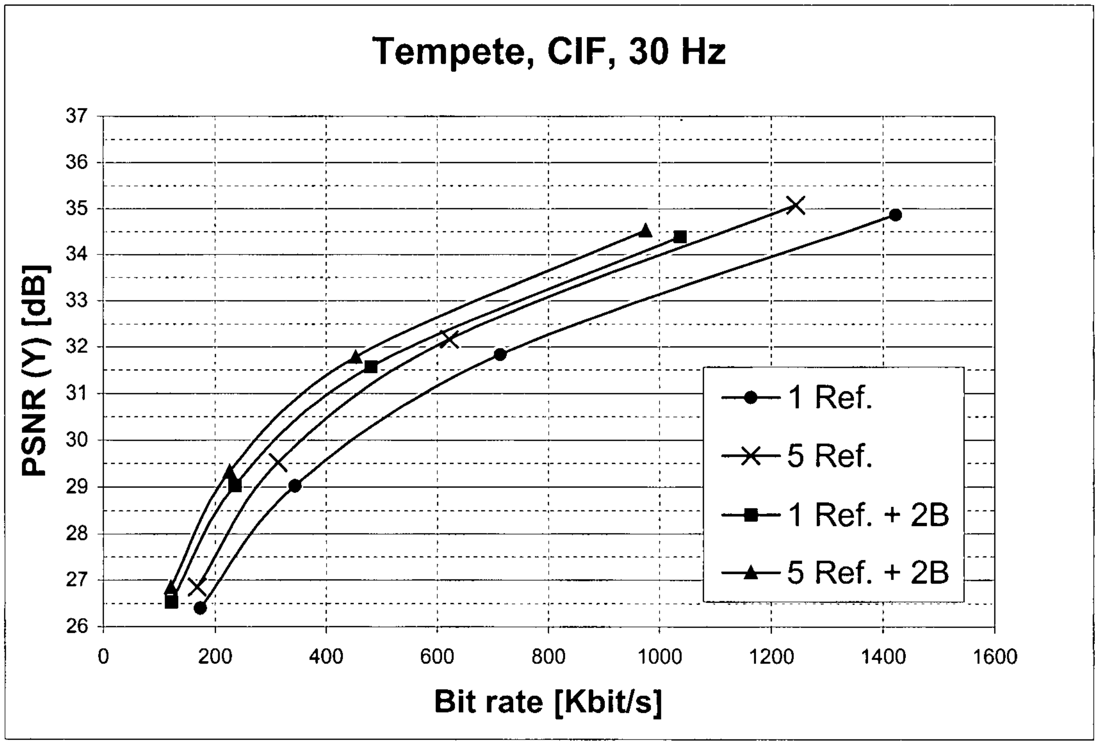


Figure 5-3: Sample rate-distortion curves illustrating the effect of inserting 2 consecutive B-pictures with and without multiple reference frames in H.26L.

Our results show that B-pictures yield significant gains in coding efficiency, with bit savings as high as 35%, but the gains are also strongly dependent on the source content. Based on these results and knowledge of the source content, we can conclude that B-pictures work best when the motion from one frame to the next is smooth and the difference between frames is not very large. They work poorly on sequences that have very fast changes and camera panning, such as Foreman at 10 Hz and Trailblazers. The average bit savings over the entire test set from adding 2 consecutive B-pictures are 12% with only a single reference frame and 10% when 5 previous reference frames are used. B-pictures likely provide less improvement when 5 reference frames are already being used because some of the benefit of bi-directional prediction is already captured by the availability of multiple temporally previous reference frames.

			5 Ref. with No B		5 Ref. with 2 B	
Res.	**Hz**	**Sequence**	**%Bits**	**PSNR**	**%Bits**	**PSNR**
QCIF		Container	2.73	0.135	-0.72	-0.032
		Foreman	4.34	0.246	4.80	0.265
		News	0.95	0.055	1.14	0.067
	15	Silent Voice	4.07	0.199	5.81	0.278
320x240	15	PI	1.65	0.090	2.98	0.160
CIF	15	Paris	3.36	0.173	5.43	0.285
		Silent Voice	5.69	0.240	7.66	0.324
	30	Bus	6.76	0.336	1.22	0.054
		Coastguard	0.53	0.019	0.43	0.013
		Flowergarden	6.32	0.329	2.86	0.132
		Foreman	3.19	0.153	0.76	0.032
		Mobile	20.05	1.027	7.50	0.320
		Tempete	18.64	0.845	10.53	0.409
		Trailblazers	1.15	0.050	0.37	0.016
		MEAN	**5.67**	**0.278**	**3.63**	**0.166**
		MIN	**0.53**	**0.019**	**-0.72**	**-0.032**
		MAX	**20.05**	**1.027**	**10.53**	**0.409**

Table 5-6: Coding improvement for using multiple reference frames with and without 2 consecutive B-pictures already present in the bitstream.

Some interesting observations can be drawn from Table 5-6, which compares the coding gain achieved by using 5 reference frames versus using only a single reference frame with and without B-pictures present in the bitstream. The general trend is that the gain that can be achieved from multiple reference frames are reduced when B-pictures are present. This effect is strongest on the sequences for which multiple reference frames provide the largest improvements, such as Mobile and Tempete. Over the entire test set, the average bit savings are reduced from 5.67% to 3.63% if B-pictures are already present, and the maximum savings are reduced by a factor of 2. However, on several sequences, the benefit from using 5 reference frames is actually larger when B-pictures are present than when they are not. These sequences include Paris, Silent (at QCIF and CIF), PI and News. This result goes against the expectation that the larger distance between reference frame will lead to less temporal correlation and, therefore, reduced coding gain from multiple reference frames. The characteristics of the sequences for which this effect is present suggest that the larger temporal distance between frames allows the encoder to detect a background area that was occluded for some period and then revealed. In these sequences, 5 frame intervals in not enough time to capture such activity, but the insertion of 2 B-pictures between each pair of reference frames allow the encoder to search a reference frame that is 15 frame intervals in the past. These results imply that multiple reference frame prediction could yield greater improvements in some cases if a larger number of reference frames is used, or if intelligent adaptive buffering methods were developed and used in place of the sliding window buffer.

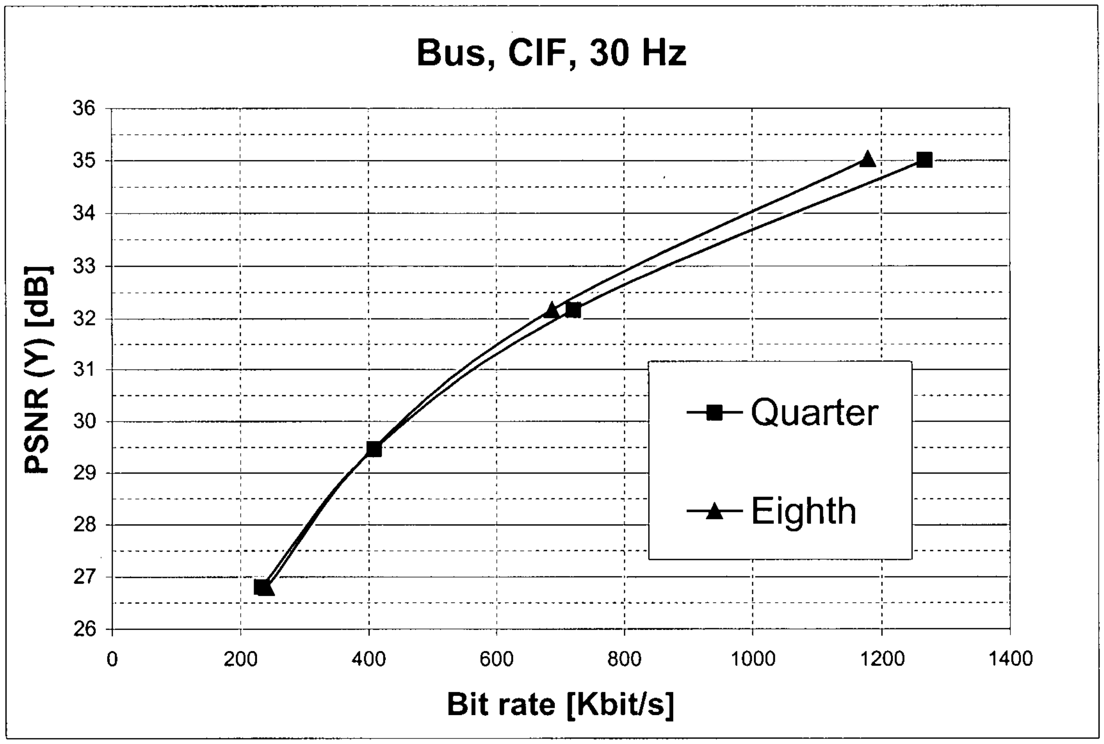
5.5 Spatial Accuracy for Motion Compensation

The current draft of H.26L specifies a default of quarter-pixel and includes an optional one-eighth-pixel accuracy mode for motion compensation [71]. Eighth-pixel accuracy increases the computational complexity of both the encoder and decoder due to the larger number of phases and taps necessary in the interpolation filter. The encoder must also perform an additional stage of motion vector refinement, after the best quarter-pixel accurate motion vector is found. Finally, since motion vectors must be transmitted with higher spatial accuracy, the overhead in the bitstream is essentially doubled for eighth-pixel accuracy.

This experiment assesses the benefit of increasing the spatial accuracy of motion vectors from quarter-pel to eighth-pel. Five reference frames, UVLC entropy coding, and a search range of 16 pixels were used. Delta-SNR results are presented in Table 5-7 and sample RD-curves are given in Figure 5-4.

Res.	Hz	Sequence	% Bits Savings	PSNR Gain
QCIF	10	Container Ship	0.57	0.032
		Foreman	-5.60	-0.314
		News	-3.47	-0.205
	15	Silent	-6.05	-0.285
320x240	15	PI	-4.66	-0.252
CIF	15	Paris	-1.85	-0.097
		Silent	-5.28	-0.217
	30	Bus	2.50	0.129
		Coastguard	-2.50	0.089
		Flowergarden	6.80	0.380
		Foreman	-5.22	-0.254
		Mobile	10.78	0.574
		Tempete	1.88	0.082
		Trailblazers	-5.99	-0.265
		MEAN	**-1.29**	**-0.043**
		MIN	**-6.05**	**-0.314**
		MAX	**10.78**	**0.574**

Table 5-7: Delta-SNR summary for eighth-pel motion vector accuracy.



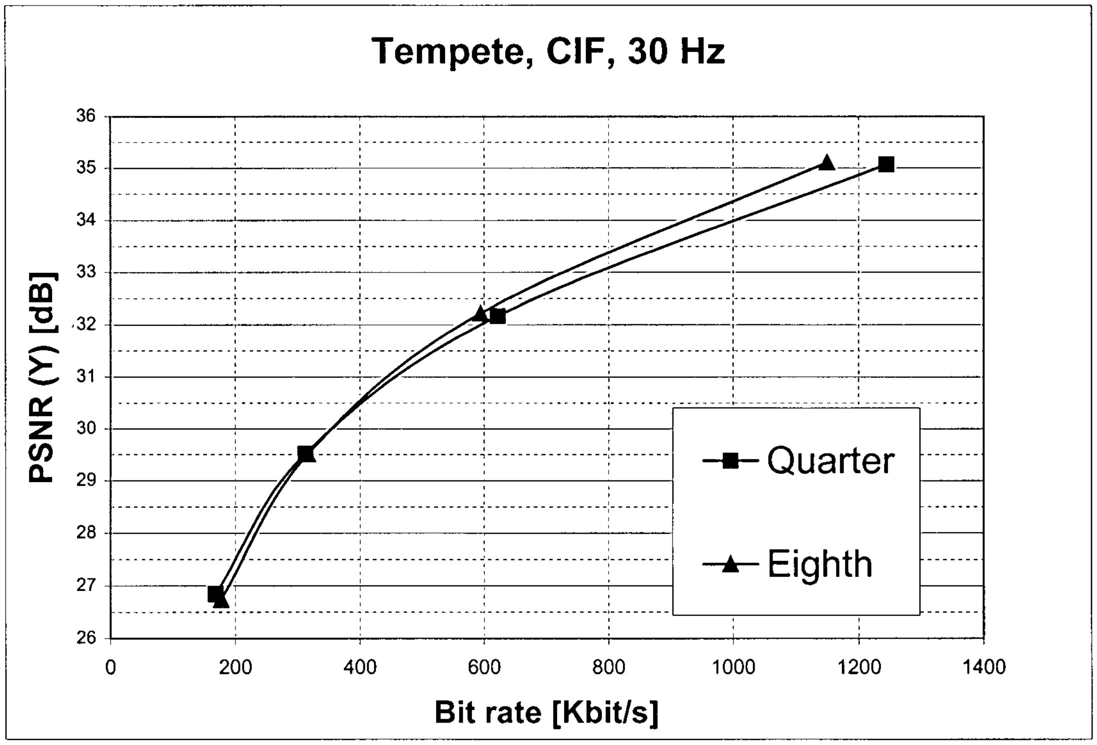


Figure 5-4: Sample rate-distortion curves illustrating the effect of the spatial motion vector accuracy in H.26L.

Initially, would seem from the Delta-SNR summary that eighth-pel accuracy provides little benefit, especially considering its added complexity. However, by looking at the RD-curves, it is clear that the benefit provided by using eighth-pel accuracy increases as the fidelity of the coded video is increased. The results also suggest that eighth-pel may be more beneficial at higher video resolutions. Therefore, eighth-pel accuracy would likely be beneficial in profiles meant to target applications that require high-quality video – digital cinema for example. The high complexity and minimal benefit at low bit rates and resolutions suggest that eighth-pel accuracy is unlikely to be used in applications requiring real-time encoding capability. Thus, only quarter-pel accuracy will be considered in the complexity reduction algorithms described in the following chapter, which target such real-time applications.

5.6 Entropy Coding Methods

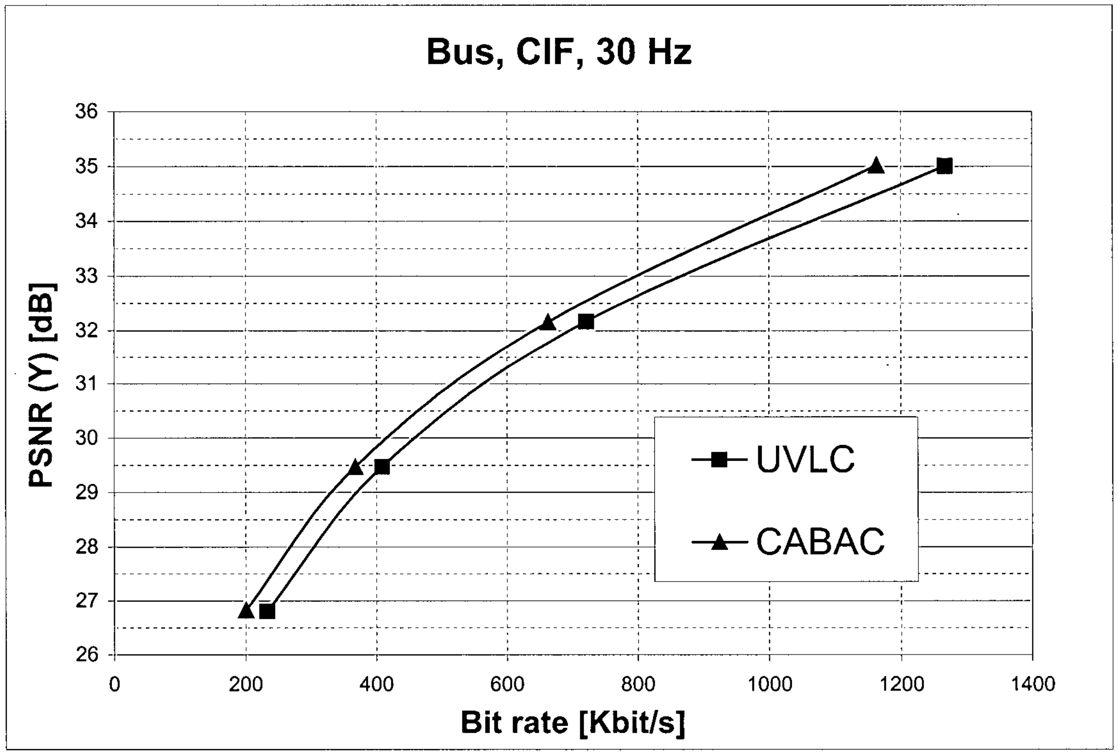
In our final experiment, we compare the rate-distortion performance of the two entropy coding methods that are available in H.26L. The UVLC method offers ease of implementation and computational simplicity by using a single, reversible variable-length code table for all syntax elements. The more efficient CABAC method is adaptive to content and offers near-optimal entropy coding over all different types of content and bit rates, but with complexity drawbacks. For further details and results, see [59].

In this final experiment, the CABAC entropy coding mode was turned on and compared to the UVLC coding mode. Quarter-pixel motion compensation and 5 reference frames were also used. The search range was 16 pixels from the predicted motion vector. Summary Delta-SNR results are presented in Table 5-8 and sample rate-distortion plots for the entropy coding modes are shown in Figure 5-5.

Res.	**Hz**	**Sequence**	**% Bits Savings**	**PSNR Gain**
QCIF	10	Container Ship	4.96	0.251
		Foreman	7.56	0.436
		News	4.98	0.295
	15	Silent	4.81	0.238
320x240	15	PI	6.35	0.355
CIF	15	Paris	5.47	0.288
		Silent	6.94	0.298
	30	Bus	9.49	0.485
		Coastguard	9.50	0.376
		Flowergarden	9.98	0.536
		Foreman	13.07	0.686
		Mobile	9.39	0.467
		Tempete	7.54	0.329
		Trailblazers	7.48	0.349
		MEAN	**7.68**	**0.385**
		MIN	**4.81**	**0.238**
		MAX	**13.07**	**0.686**

Table 5-8: Delta-SNR summary for CABAC entropy coding.

These results show that CABAC provides consistent gains in coding efficiency, typically between 5 and 10 percent, over a wide variety of content. CABAC seems to provide larger gains for the higher resolution sequences. From the RD-curves, we note that the coding efficiency gains are present at all bit rates, and tend to be largest at very low and very high bit rates.



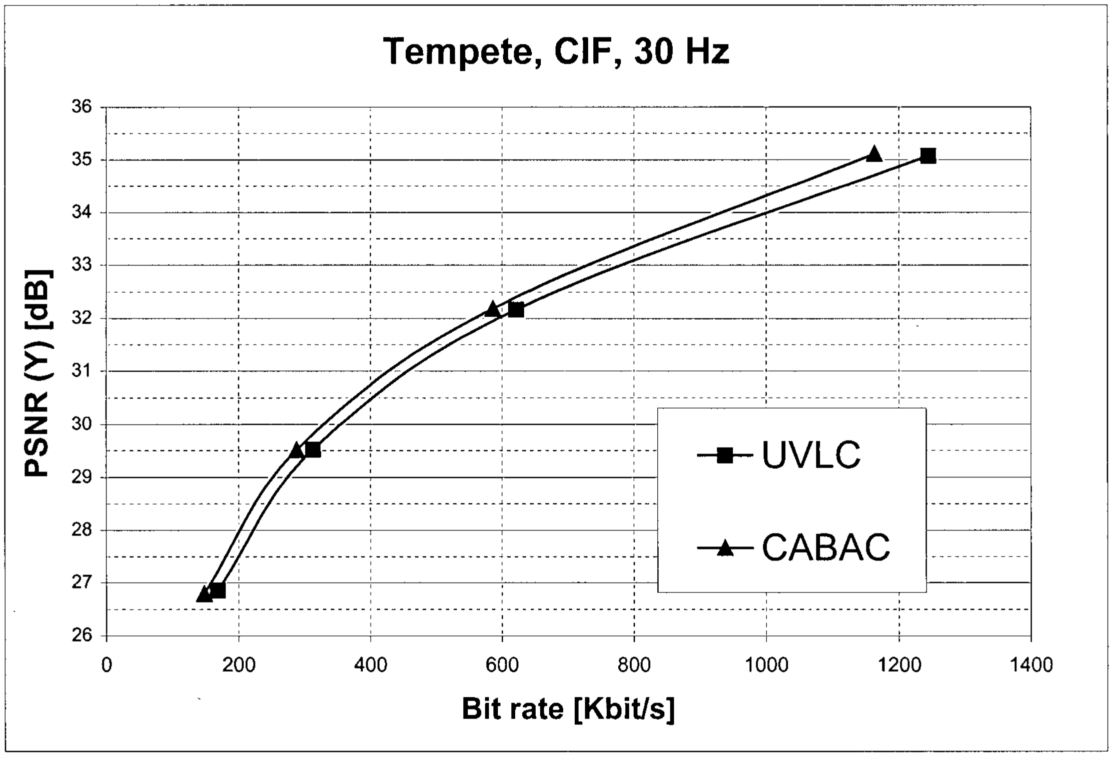


Figure 5-5: Sample rate-distortion curves for entropy coding modes in H.26L.

5.7 Summary

In this chapter, we presented a large set of data generated using the rate-distortion optimized mode of the H.26L TML-8.5 software in order to analyze the performance of several features of the current H.26L draft standard. The results indicate that the use of many different block sizes provides a consistent improvement, averaging 16% bit savings if all block types are used versus using the 16x16 mode only. However, using only 8x8 and larger blocks can capture most of the benefit of the different block sizes, although the smaller blocks become more beneficial as the bit rate is increased. Coding gains using multiple reference frame prediction seem to be highly dependent on source content. While large improvements are produced for 2 sequences, average bit savings are less than 5% for the majority of the 14 tested sequences. The benefit of multi-frame prediction is further reduced when B-pictures are present in the bitstream. B-pictures themselves produced relatively large gains, with the largest gains on sequences with smooth changes from one frame to the next. Our observations with regards to the benefit of eighth-pixel accuracy indicate that such is only beneficial at high resolutions and high bit rates, and also contain high spatial detail. Finally, the CABAC entropy coding mode provides a fairly consistent improvement in coding efficiency of between 5 and 10% over the UVLC mode. This seems to be largest for higher resolution sequences and at very low bit rates.

6 Algorithms for Real-Time H.26L Encoding

This chapter presents the final contribution of this thesis, in which we build upon the analysis of H.26L coding features from the previous chapter in order to develop reduced-complexity encoding algorithms that enable efficient real-time H.26L encoding. Several complexity bottlenecks in the current H.26L Test Model "low-complexity mode" motion estimation and mode decision processes are systematically replaced with reduced-complexity algorithms that sacrifice little or no coding efficiency. We will illustrate that real-time H.26L encoding is possible while achieving compression performance that is significantly better than even a fully RD-optimized H.263 or MPEG-4 encoder.

6.1 Introduction

Real-time video encoding capability is most important for conversational/interactive video applications, for which real-time encoding capability is a necessity. These applications represent the primary motivation for the development of real-time video encoding algorithms. Thus, results presented in this chapter will focus on such applications in terms of the content and H.26L coding features that are used. Still, many of the techniques described herein can also be used to reduce encoding complexity in other video applications, for which real-time encoding is not an absolute requirement.

The performance of our algorithms will be compared to that of the H.26L Test Model low-complexity mode. This mode does not use full Lagrangian-based rate-distortion optimization, as the high-complexity mode does, but it is much more realistic for any real-time application, since it does not require that each macroblock be fully encoded multiple times in different modes, as does the high-complexity mode. Coding performance will be illustrated in terms of rate-distortion curves that illustrate the drop in coding efficiency introduced by the fast algorithms. Reductions in complexity will be measured in terms of the number of fundamental operations that contribute to each bottleneck that we are addressing. We will also include results in terms of the improvement in overall encoder speed.

We select a set of H.26L coding features based on the requirements of conversational video applications and the results of the coding performance analysis from the previous chapter. Obviously, B-pictures will not be included, due to delay constraints. Furthermore, multiple reference frame prediction will not be included in our analysis. This decision is based on the relatively small gains provided by multi-frame prediction for most conversational content, its added computational complexity, and the extra dimension that it adds to all analysis of coding performance. Similarly, eighth-pel motion vector accuracy will not be used, since it does not seem to be beneficial, except at very high bit rates, which are not typically used in conversational applications. The CABAC entropy coding mode will be used for all H.26L results because of the consistent gains in coding efficiency that it provides, particularly at low bit rates, which are important in conversational applications. All motion compensation block sizes will be enabled.

6.2 H.26L Test Model Low-Complexity Mode

The low-complexity mode decision algorithm described in the H.26L test model [3] and implemented in the reference software proceeds to encode each macroblock by forming predictions for all available coding modes and then encoding the macroblock in the mode that yields the minimum distortion measure. Instead of simply using the SAD as the distortion measure, the test model software also uses the sum of absolute transformed difference (SATD) during sub-pixel accurate searching. Before the SAD is computed, a 4x4 Hadamard transform is performed on the block difference, and then the SAD of this result is used as the distortion measure. This improves the block matching performance because a transform will be performed on the block difference before transmission, resulting in a coding efficiency improvement of up to 0.3 dB. The Hadamard is chosen to approximate the DCT-like integer transform used in H.26L because of its low complexity. However, even the simple Hadamard transform adds a significant amount of complexity to the motion estimation process.

For a P-picture macroblock, predictions are formed for all 4x4 intra and 16x16 intra modes and the total SATD for the macroblock is computed. Throughout the search, the test model software also adds penalties to the computed SA(T)D values to account for the number of bits required to encode the motion vector at each position and the intra prediction mode. Following intra-mode prediction, a full integer-pixel search is performed on a rectangular search area centered upon the predicted motion vector. The TML-8 software includes an efficient full search that reuses SAD values for small block, beginning with 4x4 blocks, to efficiently compute the SAD for larger blocks. For each sub-block of the seven possible macroblock motion compensation modes, the motion vector with the minimum integer-pixel SAD is found and then refined to half-pixel and quarter-pixel accuracy. In each refinement stage, all 8 sub-pixel positions adjacent to the current best motion vector are searched, using the SATD as the distortion measure. The SATD is also computed at the best integer-pixel position during the half-pixel search. Finally, the best SATD of all sub-blocks are summed to generate a macroblock SATD for each of the inter modes. These are compared with the SATDs of the best intra mode, and the macroblock is encoded in the mode with the minimum SATD.

In order to determine the bottlenecks in the encoding process, the distribution of encoding time between different functional parts of the encoder was measured using Intel's VTUNE Performance Analyzer. With a search range of 16 pixels from the predicted motion vector (i.e. a 33x33 search window), the integer-pixel search consumes roughly half of the CPU cycles of the encoder. Sub-pixel refinement of motion vectors takes one-third of the time. The TML-8 software performs interpolation of all pixel values at sub-pixel locations at the start of coding each frame and this process takes 10% of the encoding time. Intra prediction occupies about 3%, with all remaining functions taking the remaining 4%. Clearly, the motion estimation process, which consumes almost 85% of the encoder's processor cycles when the integer and sub-pixel searches are combined, must be the focus of any complexity reduction efforts.

6.3 Fast Integer Pixel Motion Estimation

We will base our complexity analysis on the number of fundamental operations, such as additions, absolute difference calculations, and comparison operations, that are required by each algorithm. Of course, there is additional computational overhead in the encoder, such as memory access and loop overhead. However, we will primarily base our results on the number of fundamental operations, since these measures are readily available and provide a sufficient measure of the computational complexity.

6.3.1 Complexity of the TML Integer Search

The integer-pixel motion search implemented in the TML software performs a full search of all pixels positions in an NxN window of the previous reference frame, where N = 2*(search range) + 1. With a search range of 16, N = 33, and the window size is 33x33=1089. The full search computes the SAD for the entire 16x16 macroblock at each of the positions in the search window. However, the summations are separated into 4x4 sub-blocks, so that these summations can be combined to generate SAD values for larger sub-blocks. In this way, absolute pixel difference operations do not need to be repeated when performing motion estimation for multiple motion compensation block sizes. Therefore, the number of individual sum of absolute difference operations per macroblock is:

$$SAD\,Ops = (2 \cdot Search\,Range + 1)^2 \cdot 16^2 = N^2 \cdot 256$$

where each operation consists of one subtraction, one absolute difference operation, and one addition, performed on pixel data. The process of combining the SADs for small blocks into larger blocks from 4x4 requires a large number of addition operations. Specifically, 8 additions per search position are required to generate all of the 4x8 SADs from the 4x4 SADs. Continuing this process up to the entire 16x16 macroblock SAD, $25 \cdot N^2$ additions are required per macroblock. The last step in the integer-pixel motion estimation process is the search through all of these SAD values for each sub-block of each motion compensation mode in order to find the minimum SAD match. This requires one comparison with the current best SAD and also the calculation of a penalty cost that is based on the number of bits required to transmit the motion vector difference for each position. Each calculation of the motion vector cost requires several basic operations, including 3 multiplications, 3 array accesses, and 2 absolute value operations. The total number of sub-blocks if all 7 motion modes are searched is 41, so the total number of these motion vector comparisons required is $41 \cdot N^2$. The TML-8 encoder was instrumented to count the number of these three types of fundamental operations per macroblock, and the results are shown in Table 6-1. Our measured results match the theoretical values exactly.

Operation	Theoretical	Measured (Range = 16, N = 33)
SAD Operations	$256 \cdot N^2$	278 784
Block Additions	$25 \cdot N^2$	27 225
SAD Compares	$41 \cdot N^2$	44 649

Table 6-1: Number of operations required for integer-search specified for H.26L TML low-complexity mode.

6.3.2 Description of Proposed Fast-Search Algorithm

All of the proposed reduced-complexity algorithms were implemented in a new C-language code-base, designed with the flexibility required by the algorithms in mind. The new coding framework was designed based on the analysis of the previous chapter and the concept of providing early exits that stop the motion estimation process once a "good enough" match has been found. This will be addressed in greater detail later in this chapter.

Our proposed fast integer search algorithm merges the floating-center diamond search strategy described in [31] with the idea of combining SAD values from smaller sub-blocks to generate those for larger blocks in order to greatly reduce the number of SAD operations. We also make use of the result from the previous chapter that found that the block sizes smaller than 8x8 can only provide relatively small improvements in coding efficiency, and anticipate the insertion of early exits from motion estimation, which will be described shortly. The motion estimation algorithm consists of the following steps:

1. Perform a floating-center diamond search for the 16x16 macroblock mode. The search path and exit from the search are based on 16x16 SAD values. However, at each search position, the SAD values are initially accumulated for each of the 4 8x8 blocks within the macroblock. These 8x8 SAD values are stored for use in the search of the 16x8, 8x16 and 8x8 macroblock modes.
2. Search the 16x8, 8x16, and 8x8 motion compensation modes, re-using the SAD operations that were stored during the 16x16 search. No new SADs are computed. The SAD values that were saved during the 16x16 search are scanned for the best match for each sub-block. The SADs for 16x8 and 8x16 blocks are calculated by adding SADs from 2 8x8 blocks.
3. Perform floating-center diamond searches for the 4x8 and 8x4 block types, using the best motion vector for the co-located 8x8 block of each smaller block as the starting point for the search.

Throughout this process, full sub-pixel refinement is performed after each integer-pixel search for a motion vector. All 8 neighboring sub-pixel positions are searched at each stage, as in the TML algorithm. However, unlike the TML algorithm, the Hadamard transform is not included in the computation of the distortion measure in any part of our encoder, because it provided only a minimal quality improvement in our fast encoder but increased complexity significantly.

To reduce the complexity further, some additional modifications have been made to the TML algorithm. First, in our new coding framework, motion estimation is performed before the intra coding modes are tested. This permits the insertion of a lossless early-exit in prediction of the 4x4 intra mode, based on the minimum SAD for the macroblock found by searching the inter modes. Prediction of the 4x4 intra mode requires complete reconstruction of each 4x4 block in the macroblock before the next 4x4 block in raster scan order can be predicted and the prediction error computed. This is because each the prediction from each block is based on the reconstructed values of pixels in the surrounding blocks. Full reconstruction of a block requires computation of its forward spatial transform, quantization, inverse quantization, and the inverse transform. However, if after accumulating the prediction error for any 4x4 block, the SAD for the intra 4x4 mode already exceeds the minimum inter-mode SAD for the entire macroblock, the 4x4 intra prediction process can be stopped because we can be certain that this mode will not be selected.

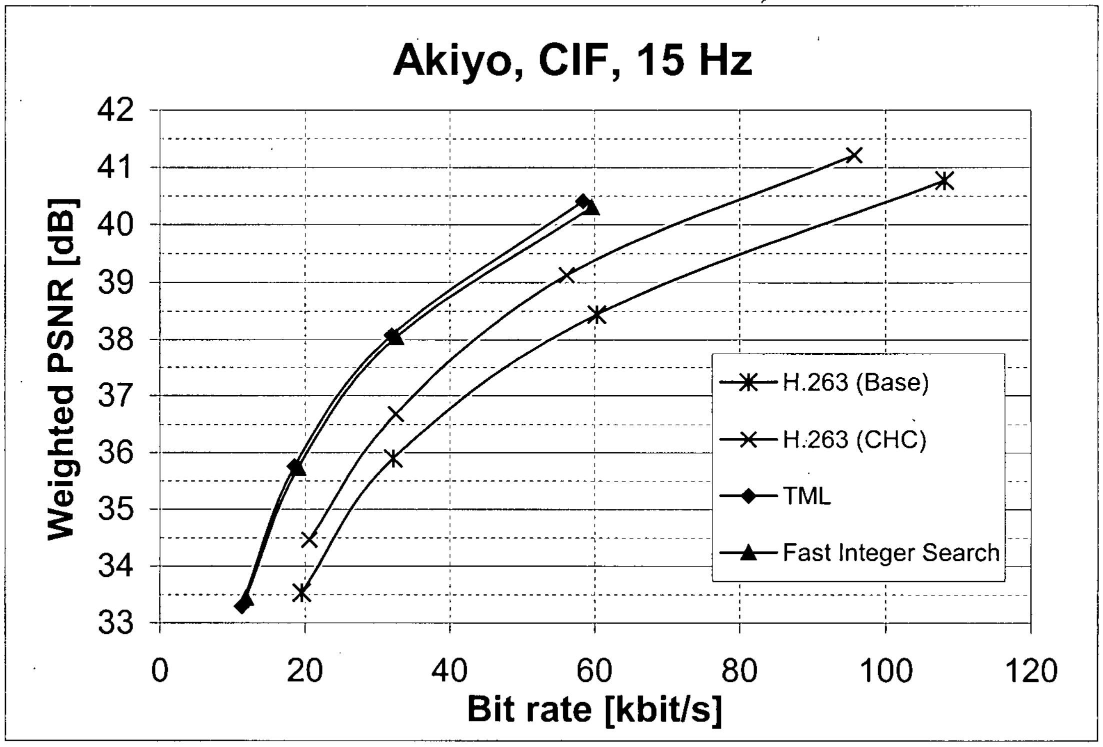
The substantial complexity reduction provided by our fast algorithms is summarized in Table 6-2 in terms of the number of operations per macroblock. The results are generated for Foreman encoded with QP=24, and Akiyo with QP=20. In both cases, the number of integer-pixel SAD operations – the most time-consuming operation in the TML encoder – is reduced by 97%. Other significant operations in the integer search are reduced by more than 99%. Additionally, our encoder uses no 4x4 Hadamard transforms, while the TML encoder averages over 2000 of these per macroblock. The early-exit from the intra 4x4 mode provides further complexity reduction in that area. The number of SAD operations during testing of the 4x4 prediction modes and the number of fully spatially transformed and reconstructed 4x4 blocks are reduced by 49-74% by the early exit from the intra 4x4 search. While direct comparison of the encoder speeds between the public software and our encoder can be somewhat misleading because some of our fast encoder's speed improvement is derived from its more efficient implementation, it is worth noting that the speed increases from 4 seconds per frame to 4 frames per second, or by a factor of 16, for encoding CIF content a 600 MHz Pentium III PC.

Foreman, CIF, 30 Hz, QP=24	**TML**	**Fast Integer Search**	**Difference**
Bit rate (kbit/s)	146.12	148.02	+1.03 %
Weighted PSNR (dB)	32.95	32.80	-0.15 dB
Integer-pel SAD Operations	277 855	8553	-96.9%
Integer-pel SAD Block Additions	27 134	45	-99.8%
Integer-pel SAD Compares	44 500	258	-99.4%
4x4 Hadamards	2054	0	-100%
Intra 4x4 SAD Operations	1510	770	-49.0%
Intra 4x4 Block Encodes	16	7.2	-55.0%

Akiyo, CIF, 15 Hz, QP=20	**TML**	**Fast Integer Search**	**Difference**
Bit rate (kbit/s)	31.95	32.56	+1.90 %
Weighted PSNR (dB)	38.08	38.04	-0.04 dB
Integer-pel SAD Operations	276 925	6094	-97.8%
Integer-pel SAD Block Additions	27 043	36	-99.9%
Integer-pel SAD Compares	44 351	188	-99.6%
4x4 Hadamards	2046	0	-100%
Intra 4x4 SAD Operations	1510	480	-68.2%
Intra 4x4 Block Encodes	16	4.16	-74.0%

Table 6-2: Comparison of coding performance and complexity for TML encoder and optimized software using floating-center diamond search, for Foreman and Akiyo.

Figure 6-1 compares the rate-distortion performance achieved by our encoder that implements these algorithms against that of the TML encoder in its low-complexity mode. We provide results for two sequences that represent typical conversational video, but vary greatly in the amount of motion and spatial detail: Akiyo and Foreman. The rate-distortion curves illustrate that the loss in coding efficiency introduced by our reduced-complexity methods is relatively small. The drop in the weighted PSNR is at most 0.2 dB for Akiyo and 0.3 dB for the more difficult Foreman sequence, with losses becoming slightly larger at higher bit rates. Also shown on the plots in Figure 6-1 are RD-curves generated using a fully RD-optimized H.263 encoder compliant with Baseline H.263 and the Conversational High Compression profile. These curves are included to illustrate the optimal compression performance that can be achieved using H.263, which is the standard that is currently used in industry for such applications. Our fast H.26L encoder consistently outperforms the fully RD-optimized H.263 CHC by at least 1 dB, and H.263 Baseline by 2 dB. Moreover, it should be noted that the RD-optimized H.263 encoder is far too complex for real-time applications. The performance of any real-time H.263 encoder would fall below the levels shown in the plots. We have measured the drop in coding performance of a real-time H.263 encoder to be in the range from 0.4 – 0.8 dB for conversational video content. This result was generated by configuring the UBC H.263 encoder to use a fast integer search and a reduced-complexity mode decision process, as would be necessary for a real-time implementation. Thus, the benefit of using H.26L over H.263 in real-time applications is even greater than what is shown in the figures presented throughout this chapter.



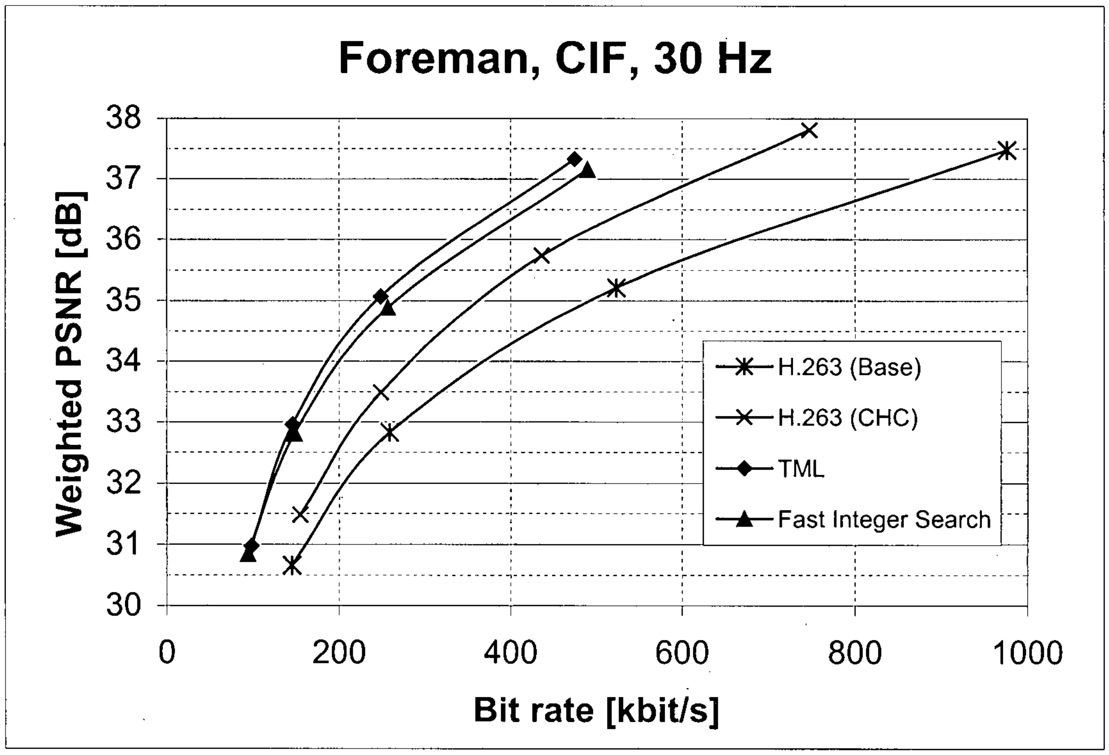


Figure 6-1: Rate-distortion performance of fast integer-pixel search.

6.4 Fast Sub-Pixel Refinement

Once the complexity of the integer-pixel search has been reduced by the methods described in the previous section, the most significant bottleneck in the encoder is, by far, the sub-pixel motion vector refinement process. In the TML software, 9 positions are tested in the half-pel stage, and 8 positions in the quarter-pel stage, for each motion vector. The additional position in the half-pel search is necessary because the Hadamard transform is also performed at the best integer-pel location. Additionally, the cost of interpolating the pixel values at the sub-pixel positions must be considered. In the TML software, all positions are interpolated only once and stored in memory. This saves repetitive interpolation of the same pixel position each time it is required during motion estimation at the cost of a very large amount of additional memory – 15 times the size of the original luminance frame. Interpolation of each half-pixel position using the 6-tap filter requires 3 multiplications, 6 additions, and one shift operation. Bilinear interpolation of quarter-pixel values from half-pixel values is less complex, requiring only one addition and one shift per pixel, although positions with more low-pass filtering (1 in 12 positions) require 4 additions and a shift. The number of operations required by the TML algorithm per P-picture macroblock when sub-pixel refinement is performed for all seven motion compensation modes is summarized in Table 6-3.

Operation	Number of Operations per Macroblock
Half-pel SAD Operations	9 x 7 x 256 = 16 128
Half-pel SAD Comparisons	9 x 41 = 369
Half-pel Interpolations	3 x 256 = 768
Quarter-pel SAD Operations	8 x 7 x 256 = 14 336
Quarter-pel SAD Comparisons	8 x 41 = 328
Quarter-pel Interpolations	12 x 256 = 3072

Table 6-3: Number of operations required for sub-pixel refinement by TML algorithm.

The sub-pixel search algorithms will be described with respect to Figure 6-2, in which upper-case letters represent integer-pixel positions, numbers represent half-pixel positions, and lower-case letters represent quarter-pixel positions.

```
A       B       C

    1   2   3

D   4 a E b 5   F
    c d e f g
    6 h 7 i 8
    j k l m n
G       H       I
```

Figure 6-2: Illustration of integer-pixel (A-I), half-pixel (1-8) and quarter-pixel (a-n) locations.

There have been methods proposed to reduce the complexity of half-pixel interpolation by making use of SAD values at integer-pixel positions that surround the best integer location and were calculated during the diamond search. These values can be used to predict the most promising half-pixel locations, and then the SAD is computed at only 0 to 4 of the most promising locations [36]. For example, assuming E is the best integer-pixel location after a diamond search, the SAD values at positions B, D, F and H would also have been computed during the integer-pixel search. One effective method suggests testing only the three half-pixel candidates that are adjacent to the integer position with the minimum SAD amongst these 4 candidates (i.e. if position H has the smallest SAD, only half-pel candidates 6, 7, and 8 are tested). Experiments have shown that it is possible to perform only 2-3 SADs while introducing only a small drop in coding efficiency.

For H.26L encoding, these algorithms must be modified in order to deal with two stages of sub-pixel refinement – half and quarter. Because there is a subsequent refinement stage, there is a need to be conservative in the half-pixel refinement. Based on these observations and our experiments, we propose the following algorithm: First, the integer position with worst SAD of the four positions surrounding the best integer position is found. Then, the three half-pixel candidates that are adjacent to this worst position are not tested, but the 5 remaining positions are checked. For example, if position B has the largest SAD of the four positions in the diamond around E, half-pixel positions 4 through 8 are tested. Once the best half-pixel position is determined, the minimum SAD amongst the adjacent integer-pixel and half-pixel positions that have already been calculated is found. Then, only the 3 or 5 quarter-pixel positions that are nearest to this position are searched, depending on the desired quality-complexity tradeoff. For example, if position 7 is the best half-pixel match, and position 8 has a smaller SAD than positions E, 6, and H, the quarter-pixel candidates `f`, `i`, and `m` are tested, with the possible addition of `e` and `l`. In another example of the search pattern, if position 6 is the best half-pixel location, then a decision is made between searching in the direction of position 4 and position 7. Assuming 7 has a lower SAD than 4, the quarter-pixel positions `d`, `h`, and `k` are searched, with the optional addition of `c` and `j`. The 2 extra positions can be tested if minimal loss in coding efficiency is more important than an additional 5-10% encoder speedup. Also, recalling that the gain from quarter-pixel motion compensation is largest at higher bit rates, we have performed experiments to determine that the quality improvement from searching the additional two points becomes most significant when the quantization parameter is less than or equal to 20. This adaptation method is included in our implementation.

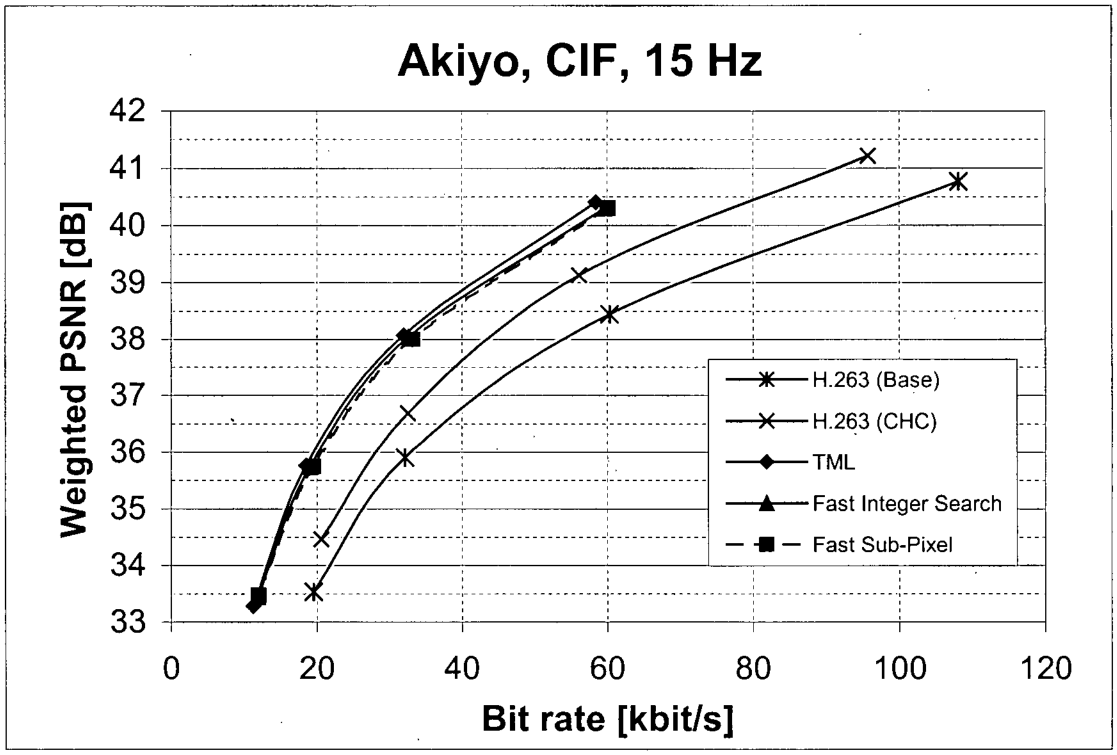
With the reduced number of quarter-pixel SAD operations that will be required by this algorithm, the large amount of memory required for interpolating all quarter-pixel positions in advance can be saved by instead interpolating only quarter-pixel values as needed. Although some interpolations may be repeated because the same positions are tested more than once, the reduction in memory requirements is substantial. With this method, only the half-pixel positions are interpolated in advance, and the additional memory requirement for storing interpolated frames is reduced from 15 additional luminance frames to only 3.

Figure 6-3 illustrates the rate-distortion performance of our encoding algorithm when this fast sub-pixel refinement algorithm is implemented. The reduction in PSNR is less than 0.1 dB. In Table 6-4, the computational complexity for sub-pixel refinement in our encoder that uses these algorithms is compared to that of the TML encoder and our fast-integer search encoder from the previous section. The increased number of quarter-pixel interpolations relative to the TML is a result of the memory savings produced by not pre-interpolating all quarter-pixel values and storing them in memory in order to reduce memory requirements. The fast sub-pixel search increases overall encoding speed by 30% when 3 quarter-pixel positions are tested for each motion vector, and 22.7% when 5 positions are tested for Akiyo. The number of half-pixel operations is reduced by 35% in each case, and the number of quarter-pixel operations is reduced by 57% and 37% for testing 3 and 5 quarter-pixel positions, respectively. With the combination of the fast integer and sub-pixel searches, encoding speed of 5.5 frames per second can be achieved on our test platform.

Foreman, CIF, 30 Hz, QP=24	TML	Fast Integer Search	Fast Integer and Fast Sub-pixel	Difference for Fast-Sub-pixel Search
Bit rate (kbit/s)	146.12	148.02	151.50	+2.35 %
Weighted PSNR (dB)	32.95	32.80	32.80	0 dB
Encoding Speed (fps)	-	4.20	5.46	+30.0 %
Half-pel SAD Operations	16 074	12 247	8029	-34.4 %
Half-pel SAD Comparisons	368	199	128	-35.7 %
Half-pel Interpolations	765	765	765	0
Quarter-pel SAD Operations	14 288	12 247	5217	-57.4 %
Quarter-pel SAD Comparisons	327	199	81.2	-59.2 %
Quarter-pel Interpolations	3072	12 373	5325	-56.7 %

Akiyo, CIF, 15Hz, QP = 20	TML	Fast Integer Search	Fast Integer and Fast Sub-pixel	Difference for Fast-Sub-pixel Search
Bit rate (kbit/s)	31.95	32.56	33.15	+1.81 %
Weighted PSNR (dB)	38.08	38.04	38.00	-0.04 dB
Encoding Speed (fps)	-	4.63	5.68	+22.7 %
Half-pel SAD Operations	16 020	12 206	7680	-37.1 %
Half-pel SAD Comparisons	367	199	124	-37.7 %
Half-pel Interpolations	763	763	763	0
Quarter-pel SAD Operations	14 240	12 206	7680	-37.1 %
Quarter-pel SAD Comparisons	326	199	124	-37.7 %
Quarter-pel Interpolations	3072	12 240	7713	-37.0 %

Table 6-4: Performance of fast sub-pixel refinement algorithm for Foreman and Akiyo.



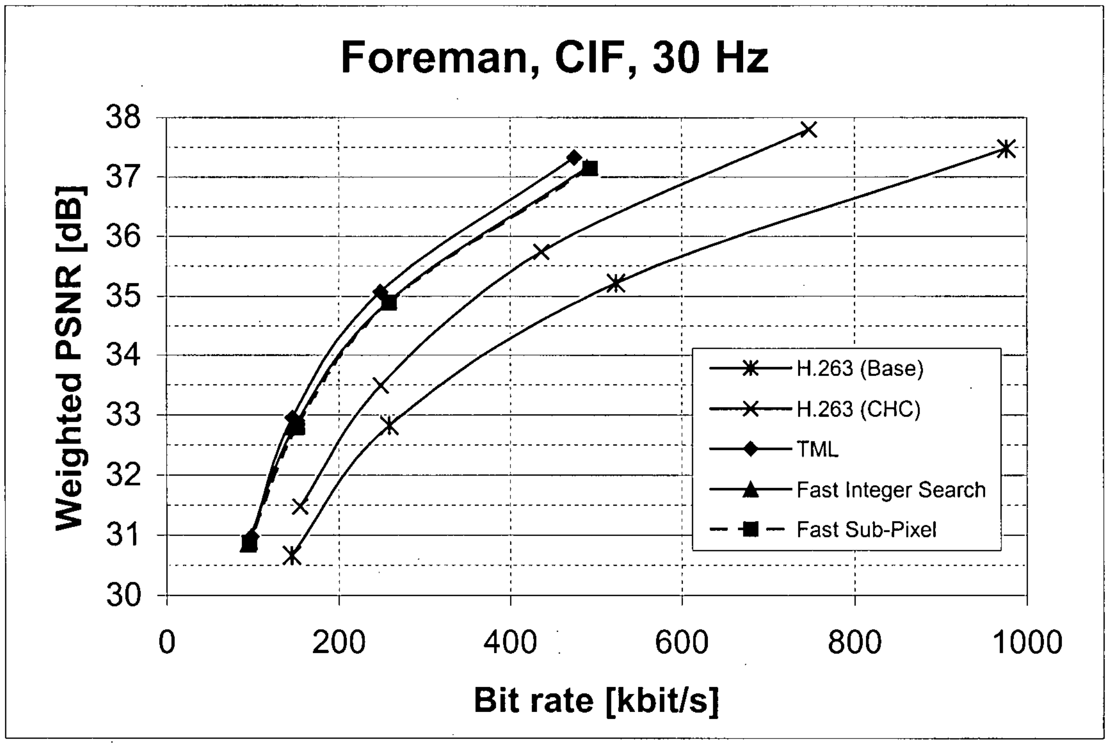


Figure 6-3: Rate distortion performance of fast sub-pixel refinement algorithm.

6.5 Threshold-Based Early Exits from Motion Estimation

Threshold-based early exits from the motion estimation process have been demonstrated to greatly improve encoding speed with minor losses in coding efficiency for H.263, MPEG-2 and MPEG-4 encoders [37] [33]. The basic idea of the early exit strategy is to terminate the motion estimation process for a macroblock once a "good enough" match has been found. H.26L is an excellent candidate for such optimization, because of the large number of possible coding modes that are available. For example, the 16x16 motion compensation mode is often determined to be the best mode after exhaustively testing all seven motion compensation modes and the two intra coding modes. Hence, if the 16x16 mode is chosen, the time spent searching all of the other modes might have been saved if the motion estimation process was ended after the 16x16 search. Threshold-based early exits terminate the motion search once a match for the macroblock has been found with a SAD below a specific threshold value. The algorithm assumes that the match is "good enough" and that little or no further improvement in the prediction will be achieved if the motion search continues.

After performing motion estimation in our H.26L encoder for each of the seven inter modes, the minimum SAD for the macroblock thus far is tested against a threshold value. If the SAD is below this threshold, the motion estimation process is stopped. Our analysis of H.26L coding features showed that most of the coding gain that can be realized by using several different motion compensation block sizes can be captured by using only the larger block sizes. Furthermore, the computational complexity of testing each motion compensation mode is largest for the modes with the smallest sub-block partitions. Since, the cost associated with transmitting all of the motion vectors for these small sub-block modes is also large, these modes will only be selected for macroblocks whose minimum SAD is large. These observations suggest that the threshold values should increase for smaller block sizes, so that the small block coding modes are only tested when the minimum SAD found so far is very large. Exits are also inserted before the testing of the two intra modes, so that these modes are not searched when one of the motion compensated modes has provided a suitable match.

In our work, we have developed a novel, content-adaptive approach to the selection of threshold values. In prior work on threshold-based exits, threshold values have been experimentally determined constants that are selected by analyzing the performance of the encoder over a wide variety of source content and bit rates. However, setting the thresholds to constant values has a significant drawback in that the performance of the early exits will vary greatly depending on the characteristics of the content and video bit rate. This is because the range of minimum SAD values for a macroblock varies depending on the content and bit rate. For example, SAD values are largest for content that contains high spatial detail. Also, SAD values become larger when the quantization parameter is increased. Thus, the consistency of the early exit method can be greatly improved if the threshold values are normalized based on the average minimum SAD value for the current content and bit rate. Essentially, normalization of the threshold values adapts the definition of a "good enough" match to the current characteristics of the video being encoded. In our proposed method, the average minimum SAD value found for all macroblocks of the previous frame is used to normalize the thresholds. For encoding the first frame of a sequence, or in the case that a scene change is detected, the threshold values are initialized to conservative constant values.

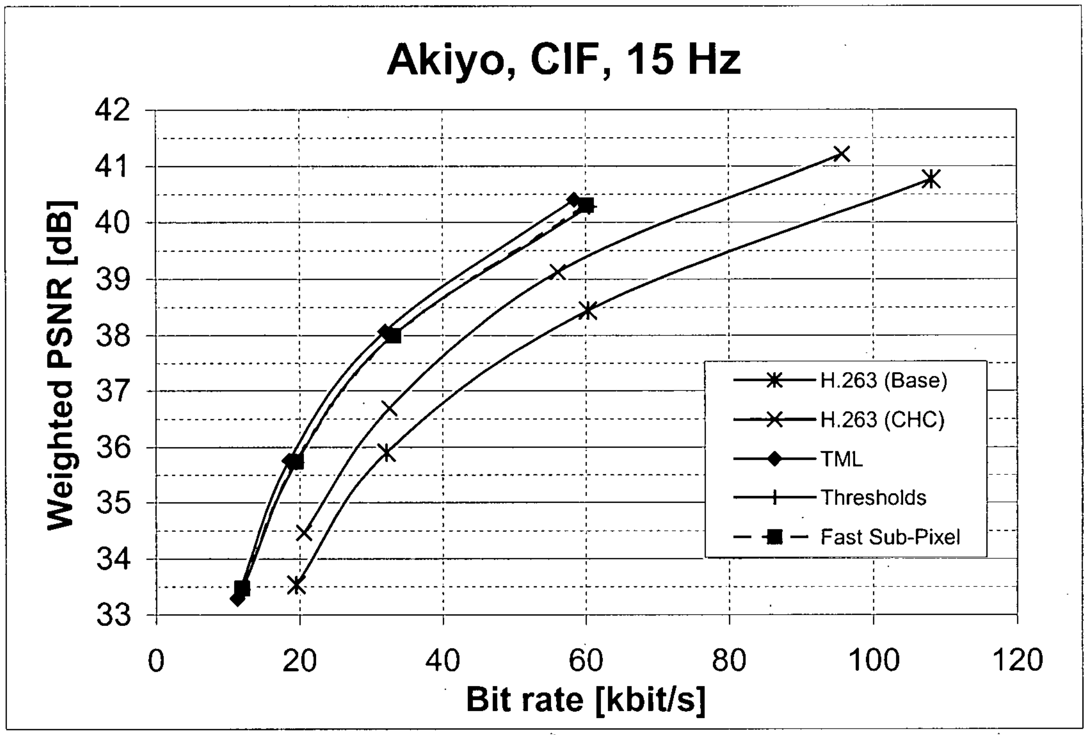
Rate-distortion curves for Akiyo and Foreman showing the performance measured after adding the thresholding methods to the fast integer and sub-pixel searches are given in Figure 6-4. The drop in coding performance from adding the early exits is less than 0.1 dB, with the largest degradation occurring at the highest bit rates. Simply scaling the threshold values based on the quantization parameter can reduce this effect by performing more conservative thresholding at high bit rates. Also note that, although the characteristics of the two test sequences are very different, with Foreman containing much more motion and spatial detail than Akiyo, the relative performance of the adaptive thresholding techniques is very similar for the two sequences.

Complexity analysis for the two test sequences is presented in Table 6-5. As can be seen, the number of operations required has been dramatically reduced by the introduction of threshold based early exits. Encoding speed is increased by 82% and 105% for the two sequences.

Foreman, CIF, 30 Hz, QP=24	TML	Fast Integer and Fast Sub-pixel	Fast Integer, Sub-pixel and Early Exits	Difference for Early Exits
Bit rate (kbit/s)	146.12	151.50	151.69	+0.13 %
Weighted PSNR (dB)	32.95	32.80	32.74	-0.06 dB
Encoding Speed (fps)	-	5.46	9.96	+82.4 %
Integer-pel SAD Operations	277 855	8553	4238	-50.5 %
Integer-pel SAD Compares	44 500	258	87.3	-66.2 %
Half-pel SAD Operations	16 074	8029	3877	-51.7 %
Half-pel SAD Comparisons	368	128	37.4	-70.8 %
Quarter-pel SAD Operations	14 288	5217	2562	-50.9 %
Quarter-pel SAD Comparisons	327	81.2	24.9	-69.3 %
Quarter-pel Interpolations	3072	5325	2666	-49.9 %
Intra 4x4 SAD Operations	1510	770	178	-76.9 %
Intra 4x4 Block Encodes	16	7.2	1.75	-75.7 %
Intra 16x16 SAD Operations	973	973	496	-49.0 %

Akiyo, CIF, 15 Hz, QP=20	TML	Fast Integer and Fast Sub-pixel	Fast Integer, Sub-pixel and Early Exits	Difference for Early Exits
Bit rate (kbit/s)	31.95	33.15	33.13	-0.00 %
Weighted PSNR (dB)	38.08	38.00	38.00	0.0 dB
Encoding Speed (fps)	-	5.68	11.64	+104.9 %
Integer-pel SAD Operations	276 925	6094	3019	-50.5 %
Integer-pel SAD Compares	44 351	188	51.9	-72.4 %
Half-pel SAD Operations	16 020	7680	2866	-62.7 %
Half-pel SAD Comparisons	367	124	27.0	-78.2 %
Quarter-pel SAD Operations	14 240	7680	2866	-62.7 %
Quarter-pel SAD Comparisons	326	124	27.0	-78.2 %
Quarter-pel Interpolations	3072	7713	2899	-62.4 %
Intra 4x4 SAD Operations	1510	480	130	-72.9 %
Intra 4x4 Block Encodes	16	4.16	1.18	-71.6 %
Intra 16x16 SAD Operations	973	973	465	-52.2 %

Table 6-5: Performance of adaptive threshold-based early exits for Foreman and Akiyo.



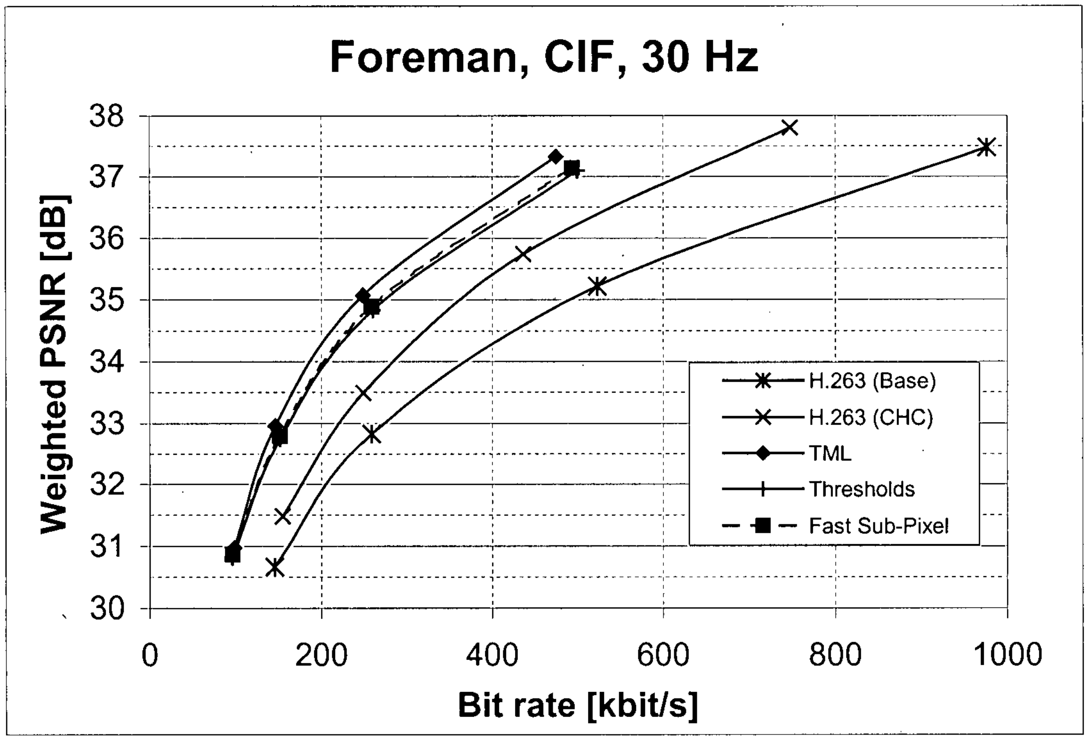


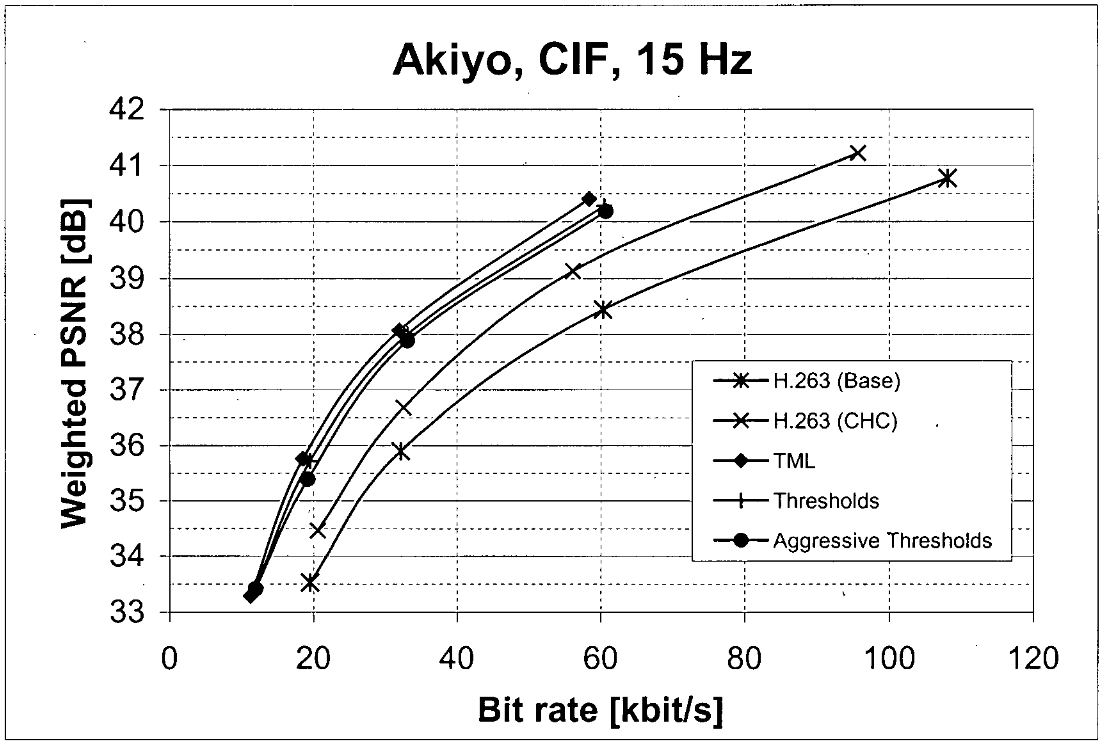
Figure 6-4: Rate-distortion performance of threshold-based early exits.

Another attractive feature of the early exit method is that it is parameterizable. The complexity-performance tradeoff can be easily modified by selecting different values for the thresholds. The results in the above table represent relatively conservative threshold values that assure minimal loss in coding performance. However, by scaling all of the threshold values upward by 30%, a new operating point is reached that results in further encoding speedup of more than 30% with a quality loss of approximately 0.1 dB. A comparison to the performance of the encoder with conservative and aggressive threshold settings is given in Table 6-6, and rate-distortion curves in Figure 6-5. Notice that the speed increase and quality degradation are again very similar for the two sequences due to the adaptive thresholding methods.

Foreman, CIF, 30 Hz, QP=24	Conservative Thresholds	Aggressive Thresholds	Difference for Early Exits
Bit rate (kbit/s)	151.69	150.90	-0.52 %
Weighted PSNR (dB)	32.74	32.63	-0.11 dB
Encoding Speed (fps)	9.96	13.10	+31.5 %

Akiyo, CIF, 15 Hz, QP=20	Conservative Thresholds	Aggressive Thresholds	Difference for Early Exits
Bit rate (kbit/s)	33.13	33.07	-0.18 %
Weighted PSNR (dB)	38.00	37.89	-0.11 dB
Encoding Speed (fps)	11.64	15.67	+34.6 %

Table 6-6: Performance comparison of using conservative and aggressive threshold values for Foreman and Akiyo.



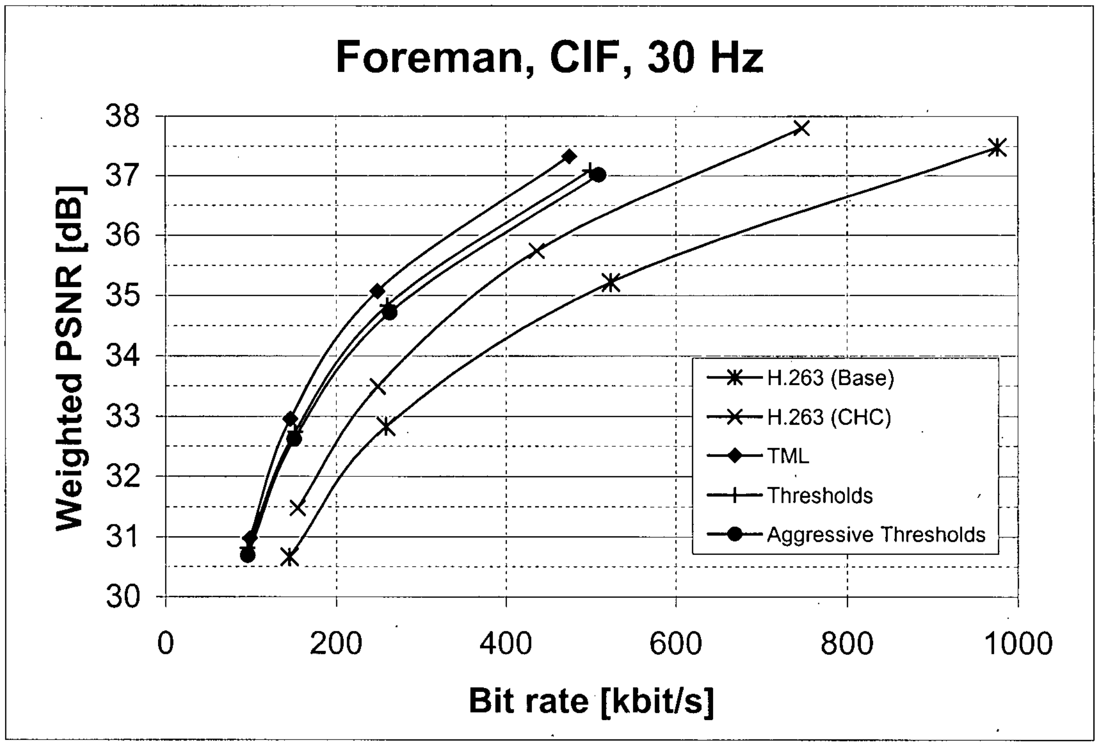


Figure 6-5: Comparison of rate-distortion performance of conservative and aggressive thresholds. Aggressive threshold produce encoding speeds more than 30% faster.

6.6 Summary

The combination of algorithms presented in this chapter when implemented in efficient C language, produce encoding speeds that are over 50 times of those speeds produced by the TML software with minimal loss in compression performance. The rate-distortion performance of the fast H.26L encoder is still generally more than 1 dB better than the RD-optimized H.263 CHC encoder and 2 dB better than the RD-optimized H.263 Baseline encoder. Furthermore, the algorithms are all computationally simple and platform-specific optimization (e.g. MMX) could readily be added to the C-language implementation of the encoder to further improve encoding speed significantly. The combination of such optimization with a faster processor than that used in generating these results, encoding of CIF resolution content at 30 frames per second should be easily achievable.

7 Conclusions and Future Research

In this thesis, we present a detailed analysis of the rate-distortion coding performance of the draft H.26L video coding standard and propose a set of efficient algorithms that enable real-time H.26L encoding for conversational video applications. Our application-based comparisons in Chapter 4 illustrate the significant improvement in coding efficiency that H.26L can provide relative to existing video coding standards for both interactive and streaming video applications. These comparisons, conducted within a rate-distortion optimized framework, also establish a limit of coding performance for each standard. We then perform a detailed analysis of the individual improvements in coding performance provided by several key features of the H.26L standard in Chapter 5. This analysis determines the characteristics of the coding gains provided by each of the features of interest and establishes a foundation for the development of our reduced-complexity encoding algorithms. These algorithms are described in Chapter 6. Specific algorithms include a fast integer-pixel search that is based on the floating-center diamond search, fast sub-pixel motion vector refinement, and content-adaptive threshold-based early exits from motion estimation. When used in combination, the set of proposed algorithms reduce the complexity of H.26L encoding my nearly 2 orders of magnitude compared to the TML "low-complexity mode", with only minor sacrifices in coding efficiency.

Our efficient C implementation of an H.26L encoder using these algorithms yields encoding speeds of up to 15 frames per second on a 600 MHz Pentium III PC for CIF resolution content, while maintaining coding performance significantly better than even the most highly RD-optimized H.263 implementation. Implementations of these algorithms that are optimized for specific powerful hardware platforms, such as Texas Instruments' C64 and Equator's MAP-CA, should easily yield encoding speeds of more than double that of our C implementation. Thus, by using the proposed algorithms, real-time H.26L encoding is within reach.

Our work suggests several interesting directions for future research on the topic of real-time video encoding algorithms. In a real-time application, the problem facing the encoder designer is not simply to encode as fast possible, but rather to encode with the best possible quality, given a fixed amount of time or processor cycles to encode each frame. While our goal in this thesis was to demonstrate that real-time encoding is achievable with H.26L, future research might address the problem of finding the best balance of coding features and algorithms to optimize the use of a fixed number of processor cycles in a real-time application. Furthermore, the development of reduced-complexity encoding algorithms specifically for offline encoding applications such as streaming and distribution, in which real-time is not a requirement, is another area for future research. While many of the concepts presented in this work would also be beneficial in such applications, a focused research geared towards such applications would lead to improved cost-performance tradeoffs.

Bibliography

[1] ITU-T and ISO/IEC JTC1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Recommendation H.262 – ISO/IEC 13818-2 (MPEG-2), November 1994.

[2] ITU-T, "Video coding for low bitrate communication," ITU-T Recommendation H.263; version 1, November 1995; version 2, January 1998.

[3] ITU-T/SG16/VCEG (Q.6), H.26L Test Model Long-Term Number 8 (TML-8), Document VCEG-N10, Video Coding Experts Group (VCEG) 14th meeting, Santa Barbara, CA, USA, 24-27 September, 2001.

[4] K. P. Rao and P. Yip, *Discrete Coding Transforms: Algorithms, Advantages, Applications.* New York: Academic Press, 1990.

[5] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding.* Upper Saddle River, NJ: Prentice-Hall, 1995.

[6] ITU-T and ISO/IEC JTC1, "Digital Compression and Coding of Continuous-Tone Still Images," ISO/IEC 10918-1 – ITU-T Recommendation T.81 (JPEG), September 1992.

[7] ISO/IEC JTC1, "Information Technology – JPEG 2000 image coding system – Part 1: Core coding system," ISO/IEC 15444-1, 2001.

[8] H.G. Musmann, P. Pirsch, and H.J. Gralleer, "Advances in Picture Coding", *Proceeding of the IEEE*, vol. 73, no. 4, pp. 523-548, April 1985.

[9] F. W. Mounts, "A video encoding system with conditional picture-element replenishment," *Bell Systems Technical Journal*, vol. 48, no. 7, pp. 2545-2554, September 1969.

[10] D. Hein, N. and Ahmed, "*Video Compression Using Conditional Replenishment and Motion Prediction*," IEEE Transactions on Electromagnetic Compatibility, vol. EMC-26, No. 3, August 1984.

[11] A. N. Netravali and J. D. Robbins. "Motion compensated television coding: Part 1," *Bell System Technical Journal*, vol. 58, pp. 631-670, 1979.

[12] D. E. Pearson, "Developments in model-based video coding," *Proceeding of the IEEE,* vol. 83, no. 6, pp. 892-906, June 1995.

[13] J.R. Jain and A.K. Jain, "Displacement measurement and its applications in intraframe image coding," *IEEE Transactions on Communications*, vol. 29, pp. 1799-1808, 1981.

[14] D. J. Vaisey and A. Gersho, "Variable block-size image coding," *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, pp. 25.1.1-25.1.4, April 1987.

[15] B. Girod, "Why B-pictures work: a theory of multi-hypothesis motion-compensated prediction," *Proc. IEEE Int. Conf. Image Processing (ICIP),* vol. II, pp. 213-217, Chicago, October 1998.

[16] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. on Circuits Systems for Video Technology*, vol. 9, pp. 70-84, February 1999.

[17] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. on Communications,* vol. 41, pp. 604-612, April 1993.

[18] J. Ribas-Corbera and D. L. Neuhoff, "Optimizing motion-vector accuracy in block-based video coding," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 11, pp. 497-511, April 2001.

[19] K.R. Rao and J.J. Hwang, *Techniques & Standards for Image Video & Audio Coding*, Upper Saddle River, NJ: Prentice-Hall, 1996.

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

[21] W. Chen, C. H. Smith, and S. Fralic, "A fast computational algorithm for the discrete cosine transform", *IEEE Trans. on Communication*, vol. 25, pp. 1004-1009, September 1977.

[22] A. Ortega and K. Ramchandran, "Rate-distortion method for image and video compression," *IEEE Signal Proc. Magazine,* vol. 15, pp. 23-50, November 1998.

[23] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, pp. 399-417, 1963.

[24] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, pp. 1445-1453, September 1988.

[25] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression", IEEE Signal Processing Magazine, pp. 74-90, November 1998.

[26] A. Hallapuro and M. Karczewicz, "Complexity Analysis of H.26L," Document VCEG-M50, ITU-T Video Coding Experts Group (VCEG) 12th Meeting, Austin, TX, USA, 2-4 April, 2001.

[27] S. Wenger, "H.26L Complexity Analysis according to VCEG-L36 section 2.1.4," ITU-T VCEG 12th Meeting, Austin, TX, USA, 2-4 April, 2001.

[28] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architecture.* Boston: Kluwer Academic Publishers, 1995.

[29] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. IEEE National Telecommunication Conference*, vol. 4, pp. G5.3.1-G5.3.5, November 1981.

[30] S. Zhu and K.K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *Proc. of Int. Conf. on Information, Communications and Signal Processing (ICASSP)*, vol. 1, pp. 292-296, 1997.

[31] M. Gallant, G. Côté, and F. Kossentini, "An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding," *IEEE Transaction on Image Processing,* vol. 8, pp. 1451-1455, October 1999.

[32] ITU-T/SG16/Q15 (presently Q6), "H.263 Test Model 11," Document Q15-G-16, ITU-T VCEG 7th meeting, Monterey, CA, USA, 16-19 February, 1999.

[33] ISO/IEC WG11 MPEG Video Group, "MPEG-4 Video Optimized Visual Reference Software," ISO/IEC JTC1/SC29/WG11 N4057, Singapore, March 2001.

[34] Y. Senda, H. Harasaki, and M. Yamo, "A simplified motion estimation using an approximation for the MPEG-2 real-time encoder," *Proc. of Int. Conf. on Information, Communications and Signal Processing (ICASSP)*, vol. 4, pp. 2273-2276, 1995.

[35] Y. Senda, "Approximate criteria for the MPEG-2 motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 10, pp. 490-497, April 2000.

[36] B. Erol, F. Kossentini, and H. Alnuweiri, "Efficient coding and mapping algorithms for software-only real-time video coding at low bit rates," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 10, pp. 843-856, September 2001.

[37] I. R. Ismaeil, A. Docef, F. Kossentini, and R. K. Ward, "A computation-distortion optimized framework for efficient DCT-based video coding," *IEEE Transactions on Multimedia,* vol. 3, pp. 298-310, September, 2001.

[38] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30-44, April 1991.

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

[40] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," *Proc. of IEEE Data Compression Conference*, March 2000.

[41] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103-1127, Nov. 2000.

[42] ITU-T, "Video codec for audiovisual services at p x 64 kbit/s," ITU-T Recommendation H.261; version 1, November 1990; version 2, March 1993.

[43] M. Liou, "Overview of the px64 kbit/s Video Coding Standard", *Communications of the ACM*, vol. 34, no. 4, April 1991.

[44] ISO/IEC JTC1, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video," ISO/IEC 11172-2 (MPEG-1), March 1993.

[45] J.L. Mitchell, W.B. Pennebaker, C.Fogg, and D.J. LeGall, *MPEG Video Compression Standard*, Chapman and Hall, New York, USA, 1997.

[46] B.G. Haskell, A. Puri, and A.N. Netravalli, *Digital Video: An Introduction to MPEG-2*, Chapman and Hall, NewYork, USA, 1997.

[47] B. Girod, E. Steinbach, and N. Farber, "Performance of the H.263 video compression standard," *J. VLSI Signal Processing (Special Issue on Recent Development in Video)*, vol. 17, no. 2-3, pp. 101-111, 1997.

[48] G. Côté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 8, no 7, pp. 849-866, November 1998.

[49] ITU-T, “Enhanced reference picture selection mode,” ITU-T Recommendation H.263 Annex U, November 2000.

[50] ITU-T, “Data partitioned slice mode,” ITU-T Recommendation H.263 Annex V, November 2000.

[51] ITU-T, “Additional supplemental enhancement information,” ITU-T Recommendation H.263 Annex W, November 2000.

[52] M. Horowitz, “Demonstration of H.263++ Annex U Performance,” Document Q15-J-11, VCEG 10th meeting, Osaka, Japan, 15-19 May, 2000.

[53] ITU-T, “Profiles and levels definition,” ITU-T Recommendation H.263 Annex X, April 2001.

[54] ISO/IEC JTC1, “Coding of audio-visual objects – Part 2: Visual,” ISO/IEC 14496-2 (MPEG-4 visual version 1), April 1999; Amendment 1 (version 2), February, 2000; Amendment 4 (streaming profile), January, 2001.

[55] B. Erol, A. Dumitras, and F. Kossentini, "*Emerging MPEG Standards: MPEG-4 and MPEG7*, " Handbook of Image and Video Processing, Ed. A. Bovik, Academic Press, Chap. 6.5, pp.615-16, 2000.

[56] ISO/IEC JTC1/SC 29/WG 11, “Overview of the MPEG-4 Standard (V.18),” ISO/IEC JTC1/SC 29/WG 11 N4030, March, 2001. *Current version available at* http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm.

[57] K. Hibi, “Report of the Ad Hoc Committee on H.26L Development,” Document Q15-H-07, VCEG 8th meeting, Berlin, Germany, 3-6 August, 1999.

[58] G. Bjontegaard, “Clarification of “Funny Position”,” Document Q15-K-27, VCEG 11^{th} meeting, Portland, OR, USA, 22-25 August, 2000.

[59] D. Marpe, G. Blättermann, G. Heising, and T. Wiegand, "Further Results for CABAC entropy coding scheme," Document VCEG-M59, VCEG 13th meeting, Austin, TX, USA, 2-4 April, 2001.

[60] I. H. Witten, R. M. Neal, and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM,* vol. 30, no. 6, 1987.

[61] Independent JPEG Group, "JPEG Software, release 6b," March, 1998. *ftp://ftp*.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz.

[62] MPEG Software Simulation Group, "mpeg2encode/mpeg2decode version 1.2," July, 1996. http://www.mpeg.org/MSSG/, July, 1996.

[63] Signal Processing & Multimedia Group, University of British Columbia, "ITU-T H.263 Research Library, version 0.3", April, 2001. http://spmg.ece.ubc.ca.

[64] ITU-T VCEG, "H.26L TML encoder/decoder software, release 8.5," October, 2001. ftp://standard.pictel.com/video-site/h26L/tml85.zip.

[65] P. Topiwala, G. Sullivan, A. Joch, and F. Kossentini, "Performance Evaluation of H.26L TML-8 versus H.263++ and MPEG-4", Document VCEG-N18, VCEG 14th meeting, Santa Barbara, CA, USA, 24-27 September, 2001.

[66] P. Topiwala, G. Sullivan, A. Joch, and F. Kossentini, "Overview and Performance Evaluation of the ITU-T Draft H.26L Video Coding Standard," Proc. SPIE, Appl. of Digital Image Processing, August 2001.

[67] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," Document VCEG-M33, VCEG 13th meeting, Austin, TX, USA, 2-4 April, 2001.

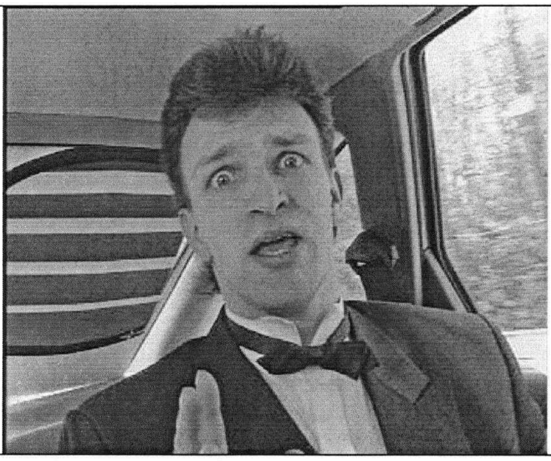
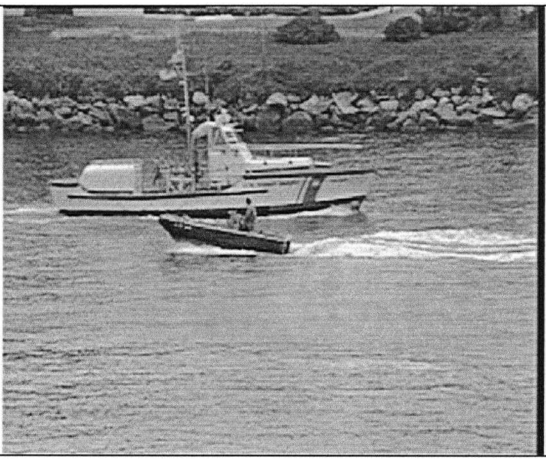
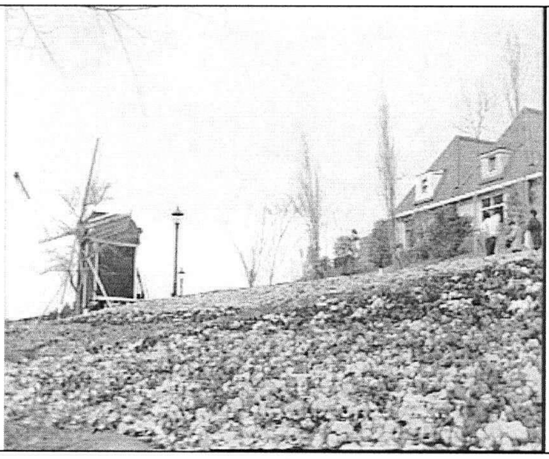
[68] A. Joch and F. Kossentini, "Performance analysis of H.26L coding features," Document VCEG-O42, VCEG 15th meeting, Pattaya, Thailand, 4-6 December, 2001.

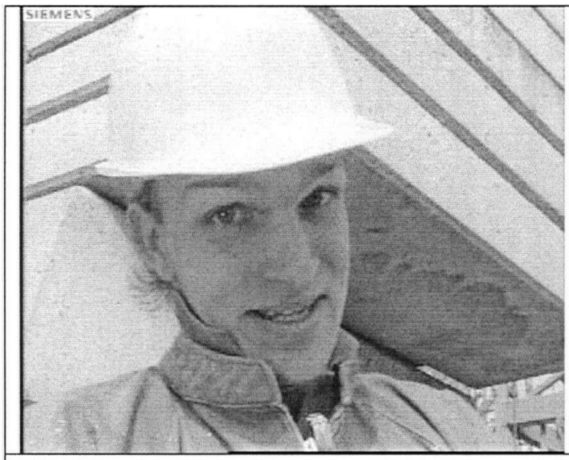
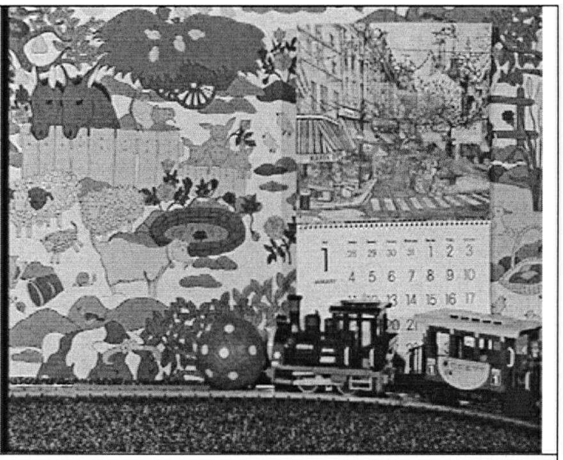
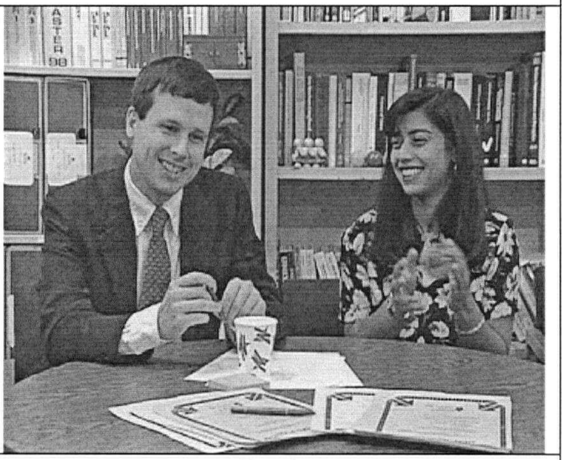
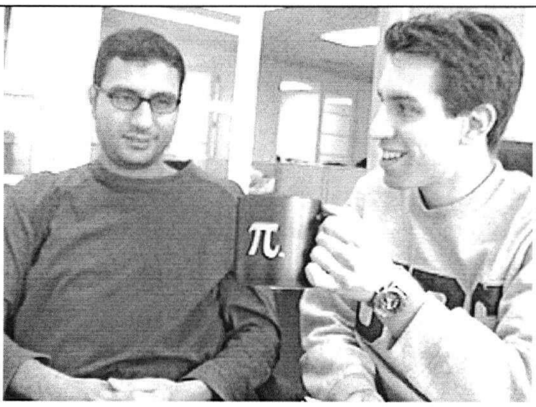
[69] G. Sullivan, "Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material," Document VCEG-N81, VCEG 14th meeting, Santa Barbara, CA, USA, 24-27 September, 2001.

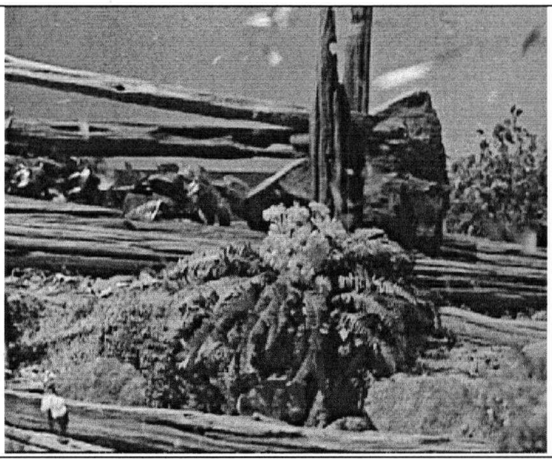
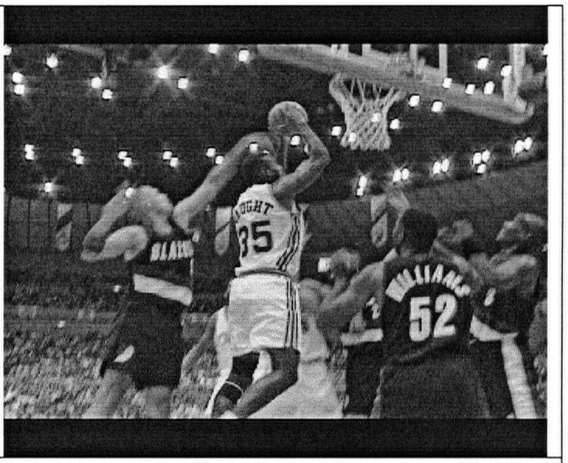
[70] M. Zhou, "Benchmark Analysis of H.26L Decoder Functional Block," Document VCEG-N23, VCEG 14th meeting, Santa Barbara, CA, USA, 24-27 September, 2001.

[71] Thomas Wedi, "1/8-pel Displacement Vector Resolution for TML-6," Document VCEG-M45, VCEG 13th meeting, Austin, TX, USA, 2-4 April, 2001.

Appendix A: Sample Frames from Video Test Sequences

Akiyo	**Bus**
	
Carphone	**Coast Guard**
	
Container (QCIF)	**Flower Garden**

	
Foreman	**Mobile and Calendar**
	
News (QCIF)	**Paris**
	
PI (320x240)	**Silent Voice**

	
Tempete	**Trailblazers (©NBA)**