LOW-COMPLEXITY MODE SELECTION FOR RATE-DISTORTION OPTIMAL VIDEO CODING

A Thesis
Presented to
The Academic Faculty

by

Hyungjoon Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2007

LOW-COMPLEXITY MODE SELECTION FOR RATE-DISTORTION OPTIMAL VIDEO CODING

To my loving wife, Sookyung

To my parents, Mom and Dad

ACKNOWLEDGEMENTS

I would like to thank to my advisor Dr. Yucel Altunbasak for the invaluable guidance, encouragement, and support during my doctoral studies. It has been a great pleasure to have him as a my advisor. My deepest thanks are also extended to Dr. Russell Mersereau, Dr. Ghassan Al-Regib, Dr. Allen Tannenbaum, and Dr. Xiaoming Huo for serving as committee member. I also thank to the members at EG Technology for helping me learning practical techniques. Special thanks go to Junfeng Bai, Sam John, Santhana Krishnamachari, John Lan, Joe Monaco, and Ramesh Panchagnula. Especially, I'm grateful to Dr. Kyeong-Ho Yang for numerous useful comments and suggestions on my research. Finally, I would like to express the deepest gratitude to my wife and my parents for giving me so much in everything so that I can concentrate on my research and successfully complete this thesis.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

SUMMARY xi

I INTRODUCTION 1

1.1 Objective and Contributions 1

1.2 Organization 5

II BACKGROUND 6

2.1 Motion Estimation 7

2.2 Coding Mode Selection 10

2.3 Transform 13

2.4 Quantization 14

2.5 Entropy Coding 16

2.6 Rate Control 17

III RATE-DISTORTION OPTIMAL MODE SELECTION 19

3.1 Rate-Distortion Model 19

3.2 Coding Mode Selection in MPEG-2 22

3.3 Joint Rate-Distortion Optimal Mode Selection and Motion Estimation . . 26

3.4 Complexity Analysis on Model-Based Mode Selection 28

3.5 Experimental Results 31

IV REAL-TIME VIDEO ENCODER IMPLEMENTATION ON DIGITAL SIGNAL PROCESSORS 43

4.1 Hierarchical Memory Structure 43

4.1.1 Level-1 Memory 44

4.1.2 Level-2 Memory 46

4.1.3 Enhanced DMA 47

4.2 Encoder Modules 48

4.2.1 Integer-Pixel Accuracy Motion Estimation 49
4.2.2 Macroblock Mode selection 51
4.2.3 Sub-Pixel Accuracy Motion Estimation 52
4.2.4 Others Modules 58
4.3 Overall structure 59
V CONCLUSION AND FUTURE WORK 65
5.1 Contributions 65
5.2 Suggestions for Further Research 65
APPENDIX A INTRINSICS FOR TMS320DM64X/TMS320C64X 68
REFERENCES 73

LIST OF TABLES

1 Key compression features in the video coding standards. Progr. and Interl. mean progressive and interlaced, respectively. 3

2 Macroblock modes for P- and B-type pictures in the MPEG-2 standard. MC and ME mean motion compensation and motion estimation, respectively. . 5

3 The average PSNR values for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed mode selection and motion estimation algorithm with ρ-rate control. 32

4 The average PSNR values when the two different γ adaptation method are used. The encoding parameters are the same as in Table 3. 33

5 The average PSNR values of the proposed algorithm for different number of λ values. The encoding parameters are the same as in Table 3 except M . . 34

6 Frame size and block size in hierarchical motion estimation for full D1 frame 49

7 Memory requirement for macroblock line based prefetch scheme 50

8 Number of calls for upsampling and cost calculation functions 64

LIST OF FIGURES

1 History of video coding standards 2

2 Hybrid encoder structure 6

3 TM5 mode selection for (a) intra-mode vs. inter-mode and (b) Motion Compensated (MC) mode vs. No MC mode 10

4 Various block shapes in H.264 13

5 Frame DCT type (left) and field DCT type (right) for luma component in the MPEG-2 standard 14

6 4x4 transform in H.264. E is a multiplication matrix and symbol $\otimes$ indicates scalar multiplication. In actual process, the scalar multiplication is incorporated into quantization and only matrix multiplication is performed in transform. 15

7 8x8 quantization matrix for (a) intra-mode and (b) inter-mode used in TM5 for MPEG-2 16

8 Scanning order for 8x8 block (a) zig-zag scan and (b) alternate scan 17

9 Experimental analysis of change of γ with respect to the rate for (a) intra-coding and for (b) inter-coding. The experimental values are obtained using a large set of randomly selected macroblocks. 21

10 RD optimal macroblock mode selection procedure. L is total number of possible macroblock modes. 25

11 RD optimal joint macroblock mode selection and motion estimation procedure. L is total number of possible macroblock modes, M is the number of λ values used for optimization. 27

12 The PSNR value at each frame for FOOTBALL for (i) TM5, (ii) TM5 with the ρ-domain rate control, and (iii) the proposed algorithm with the ρ-domain rate control. 34

13 The PSNR value at each frame for FLOWER GARDEN for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control. 35

14 The PSNR value at each frame for CAROUSEL for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control. 35

15 The PSNR value at each frame for SUSIE for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control. . 36

16 The PSNR value at each frame for NEWS for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control. 36

17 Reconstructed 318th frame of FOOTBALL when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 80112, 76608, and 42315 for (a), (b) and (c) respectively. The motion vector bits are 69969, 65657, and 13263 for (a), (b) and (c) respectively. The corresponding PSNR values are 31.9, 32.5, and 34.8 dB, respectively. 37

18 Reconstructed 320th frame of FLOWER GARDEN when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 48203, 52963 and 29204 for (a), (b) and (c) respectively. The motion vector bits are 38696, 37534, and 9509 for (a), (b) and (c) respectively. The corresponding PSNR values are 25.8, 27.0, and 29.3 dB, respectively. 38

19 Reconstructed 318th B-type frame of the CAROUSEL sequence for (a) TM5, (b) TM5 with ρ-rate control, and (c) proposed algorithm with ρ-rate control. The frame bit budget is 68242, 65607, and 43011 for (a), (b) and (c) respectively. The motion vector bits are 58172, 55554, and 12423 for (a), (b) and (c) respectively. The corresponding PSNR values are 27.7, 27.3, and 30.2 dB, respectively. 39

20 Reconstructed 171th frame of SUSIE when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 62895, 62511, and 34400 for (a), (b) and (c) respectively. The motion vector bits are 53100, 52958, and 10725 for (a), (b) and (c) respectively. The corresponding PSNR values are 34.6, 35.4, and 39.8 dB, respectively. 40

21 Reconstructed 123th frame of NEWS when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 61611, 60601, and 32083 for (a), (b) and (c) respectively. The motion vector bits are 42988, 43042, and 6562 for (a), (b) and (c) respectively. The corresponding PSNR values are 32.4, 35.1, and 36.3 dB, respectively. 41

22 The motion field of the 303th frame of FLOWER GARDEN when it is coded using (a) TM5 and (b) the proposed method. 42

23 Hierarchical memory structure 44

24 Sample encoder structure. M is the total number of macroblocks in a frame. 45

25 Hierarchical memory structure in TI's TMS320DM64x/TMS320C64x 46

26 Cache-SRAM configurations for L2 memory in TMS320DM642 47

27 Sample data transfer scheme 48

28 Three-step motion search at level-3 50

29 Required reference area at each level with macroblock line based prefetch . 51

30 Four cases in loading a reference block 53

31 Half-pixel accuracy interpolated reference image 54

32 Calculation of $b_{y,x}$ 54

33 Calculation of $c_{y,x}$ 55

34 Calculation of $d_{y,x}$ 55

35 Separate interpolated b-, c-, and d-images 56

36 SAD calculation for the b-image 57

37 MPEG-2 encoder structure 60

38 Frame and field prediction in the MPEG-2 coding standard 62

39 Calculation of frame SAD and four field SADs 63

40 Estimated rates vs. actual rates 67

41 **_subabs4** takes single cycle 69

42 **_dotpu4** takes four cycles 69

43 **_avgu4** takes two cycles 69

44 **_avg2** takes two cycles 69

45 **_shrmb** takes single cycle 70

46 **_shlmb** takes single cycle 70

47 **_add2** takes single cycle 70

48 **_sub2** takes single cycle 70

49 **_packl4** takes single cycle 71

50 **_packh4** takes single cycle 71

51 **_unpklu4** takes single cycle 71

52 **_unpkhu4** takes single cycle 71

53 **_lo** takes single cycle 72

54 **_hi** takes single cycle 72

55 **_mpyu4** takes four cycles 72

SUMMARY

The primary objective of this thesis is to provide a low-complexity rate-distortion optimal coding mode selection method in digital video encoding. To achieve optimal compression efficiency in the rate-distortion framework with low computational complexity, we first propose a rate-distortion model and then apply it to the coding mode selection problem. The computational complexity of the proposed method is very low compared to overall encoder complexity because the proposed method uses simple image properties such as variance that can be obtained easily. Also, the proposed method gives significant PSNR gains over the mode selection scheme used in TM5 for MPEG-2 because the rate-distortion model considers rate constraints of each mode as well as distortion. We extend the model-based mode selection approach to motion vector selection for further improvement of the coding efficiency.

In addition to our theoretical work, we present practical solutions to real-time implementation of encoder modules including our proposed mode selection method on digital signal processors. First, we investigate the features provided by most of the recent digital signal processors, for example, hierarchical memory structure and efficient data transfer between on-chip and off-chip memory, and then present practical approaches for real-time implementation of a video encoder system with efficient use of the features.

CHAPTER I

INTRODUCTION

1.1 Objective and Contributions

International video coding standards offer very high compression ratios, allowing the transmission, storage, and manipulation of visual information in various environments [9, 11, 12, 14, 15, 41]. They have a broad range of applications including digital TV, DVD, video conferencing, video phone, video game, education, etc. International standard organizations, such as International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) or International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), have continuously improved the video coding standards in the past two decades to achieve better compression efficiency in the various applications. For example, the ITU-T standards, called recommendations and denoted with H.26x, are usually optimized for real-time video communication while the ISO/IEC video coding standards, denoted with MPEG-x, are designed mainly for storage and broadcasting. For the most part, the two standardization committees have worked independently on the different standards except H.262/MPEG-2 and H.264/MPEG-4 Part 10 that were developed jointly by the two committees. Figure 1 shows the evolution of the ITU-T recommendations and ISO/IEC MPEG standards, and Table 1 summarizes key features of the standards.

H.261 was originally designed for video conferencing over Integrated Service Digital Network (ISDN) channels on which data rates are multiples of 64 kbit/s. The standard was quite successful and continued to be used in all subsequent video coding standards. In an attempt to improve on the compression performance of H.261, the ITU-T working group developed H.263, which was also aimed for a low-bit rate encoding for video conferencing based on experience from the H.261, MPEG-1, and MPEG-2 standards. It provides significantly better compression performance as well as greater flexibility. It was further enhanced in the projects known as H.263+ and H.263++. H.264, known as Advanced Video Coding

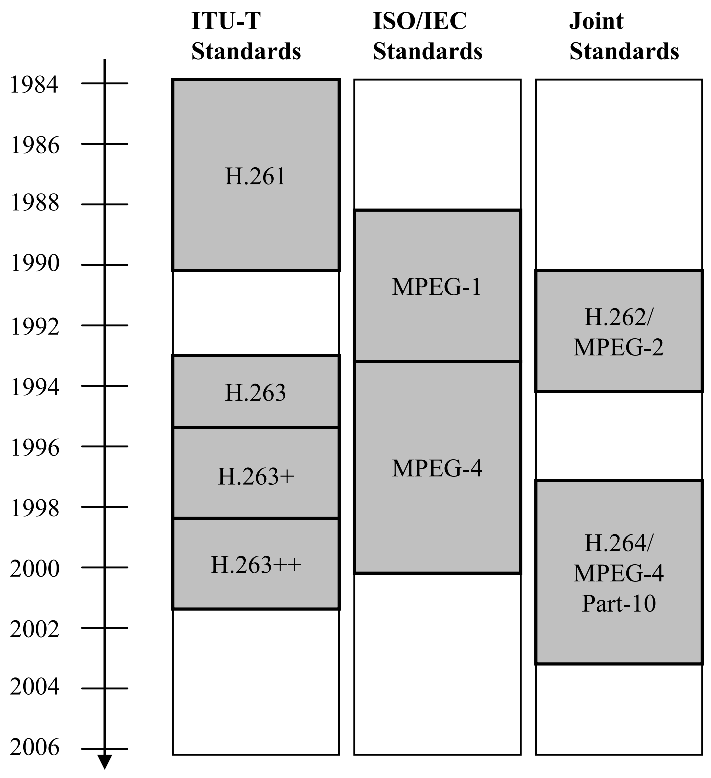


Figure 1: History of video coding standards

(AVC) standard or MPEG-4 part 10, was designed to improve compression efficiency by around 50% in comparison to any other existing video coding standards. An additional goal was to provide flexible encoding schemes that allow the standard to be applied to a broader range of applications. The video coding capabilities of H.264 cover from very low bit rate, low frame rate, and postage stamp size video for mobile and dial-up services, through to standard/high definition television services, high definition DVD, and beyond.

MPEG-1 video was originally designed for storage of CIF format (352x288) videos. It is used in the Video CD (VCD) format and its quality at standard VCD resolution and bit rate is roughly that of a Video Home System (VHS) tape. One critical disadvantage of MPEG-1 video is that it supports only progressive pictures. This deficiency helped prompt development of the more advanced video coding standard, MPEG-2. It is widely used around the world to specify the format of the digital TV signals that are broadcasted by terrestrial, cable, and direct broadcast satellite TV systems. It also specifies the format of

Table 1: Key compression features in the video coding standards. Progr. and Interl. mean progressive and interlaced, respectively.

Features	H.261	MPEG-1	MPEG-2	H.263	MPEG-4	H.264
Picture type	I, P	I, P, B	I, P, B	I, P, B	I, P, B	I, P, B
Entropy coding	VLC	VLC	VLC	VLC	VLC	CAVLC/ CABAC
MV resolution	Integer-pel	half-pel	half-pel	half-pel	quarter-pel	quarter-pel
Transform	8x8 DCT	8x8 DCT	8x8 DCT	8x8 DCT	8x8 DCT	4x4, 8x8 integer transform
Block shapes	16x16	16x16	16x8, 16x16	8x8, 16x16	8x8, 16x16	4x4, 4x8, 8x4, 8x8, 16x8, 8x16, 16x16
Intra prediction	No	No	No	No	No	Yes
Format	Progr.	Progr.	Progr./ Interl.	Progr.	Progr./ Interl.	Progr./ Interl.
Prediction mode	Frame	Frame	Field/ Frame	Frame	Field/ Frame	Field/ Frame
Deblocking filter	Yes	No	No	Yes	No	Yes

movies and other programs that are distributed on DVD and similar disks. The video part of MPEG-2 is similar to MPEG-1 but also provides support for interlaced video. MPEG-2 video is not optimized for low bit-rates, e.g., less than 1 Mbit/s, but outperforms MPEG-1 at 3 Mbit/s and above. MPEG-4 absorbs many of the features of MPEG-1, MPEG-2, and other related standards, adding new features such as extended Virtual Reality Modeling Language (VRML) support for 3D rendering, object-oriented composite files, support for externally-specified Digital Rights Management (DRM), and various types of interactivity. MPEG-4 Part-10 was written by the ITU-T together with the ISO/IEC as the product of a collective partnership effort known as the Joint Video Team (JVT). Thus, the ITU-T H.264 standard and the ISO/IEC MPEG-4 Part 10 are technically identical.

The main purpose of the video coding standards is to provide inter-operability between encoders and decoders built by different manufacturers. However, the coding standards do not necessarily represent the best solutions to implementation and compression, but rather attempt to achieve a compromise between the amount of the implementation flexibility and compression efficiency. Therefore, by specifying only decoder compatible bitstream syntax, the video coding standards can leave enough room for algorithm optimization and innovation, and also permit complexity reduction for practical implementation. In this thesis, we focus on the computational complexity reduction and compression efficiency only for encoder algorithms.

In general, a video encoder consists of non-normative modules, such as motion compensated prediction, coding mode selection, quantization, and rate control, and normative modules, such as transform, inverse quantization, run-length coding, and entropy coding. Among the non-normative modules, coding mode selection is one of the key modules for achieving high coding efficiency. Typical video sequences contain widely varying contents and motions that can be more effectively encoded if different strategies are permitted to encode different regions. Most standards address this problem by utilizing multiple modes of operation that are selected on a block-by-block basis. For example, although block-based motion compensation is generally regarded as an efficient means for coding a block, where there is a significant change between the current frame and the previous frames, it may be more efficient to encode the block without motion compensation. Table 2 shows the macroblock modes for P- and B-type pictures in the MPEG-2 standard.

To obtain high compression efficiency, optimal coding mode selection is achieved within a rate-distortion framework, which minimizes distortion given a rate constraint. In general, it is required to evaluate a cost function, which requires both actual rate and distortion, for each candidate mode, and then an optimal mode that produces minimal cost is selected in the rate-distortion sense. However, a direct calculation of rate and distortion for all coding modes causes prohibitively high computational complexity. In this thesis, we propose a rate-distortion model and apply it to the rate-distortion optimal mode selection problem. The proposed method requires low computational complexity because evaluating our cost function derived from the model needs only the simple statistics of residual image and some table lookups. On the other hand, the proposed approach can still provide good compression efficiency because our model considers rate constraints and has adaptability to varying video contents. The proposed method also can be easily extended to motion vector

Table 2: Macroblock modes for P- and B-type pictures in the MPEG-2 standard. MC and ME mean motion compensation and motion estimation, respectively.

P-Type	B-Type
Intra	Intra
Inter, no-MC	Inter, MC with frame ME-forward
Inter, MC with frame ME	Inter, MC with frame ME-backward
Inter, MC with field ME	Inter, MC with frame ME-interpolated
	Inter, MC with field ME-forward
	Inter, MC with field ME-backward
	Inter, MC with frame ME-interpolated

selection when multiple motion vectors are available in a coding mode.

In addition to our theoretical work, we provide practical solutions to real-time implementation of each encoder module as well as the proposed mode selection method on Digital Signal Processors (DSPs). First, we investigate the features provided by state-of-the-art DSPs. For instance, most of modern DSPs designed for digital media processing, e.g., Texas Instruments' TMS320DM64x or TMS320C64x DSPs, have hierarchical memory structure to minimize the time to access to external memory that operates usually much slower than CPUs, and also employ a peripheral that is dedicated to transfer data from one memory space to another. Then, we present some implementation approaches that make efficient use of these features for each encoder module and whole structure.

1.2 Organization

The organization of this thesis is as follows. Chapter 2 provides the backgrounds of hybrid video encoders and the previous researches. We also present the new features of H.264 and differences between H.264 and the old standards. Chapter 3 describes our proposed rate-distortion model and its application to coding mode selection. We also extend the method to motion vector selection. At the end of the chapter, a complexity analysis and experimental results are presented. In addition to the theoretical work, practical solutions to real-time implementation of an MPEG-2 encoder on DSPs are presented in Chapter 4. We investigate the features provided by most of modern DSPs, and discuss some practical issues on the real-time implementation of encoder algorithms with consideration of the features. In Chapter 5, we provide a conclusion and future work.

CHAPTER II

BACKGROUND

Most of video compression standards use the block-based hybrid motion compensated and transform coding method with scalar quantization. In a hybrid encoder, motion compensated prediction is employed to reduce temporal redundancy and then transform coding is applied to the corresponding difference image to reduce spatial redundancy. Figure 2 shows a block diagram of a typical hybrid encoder.

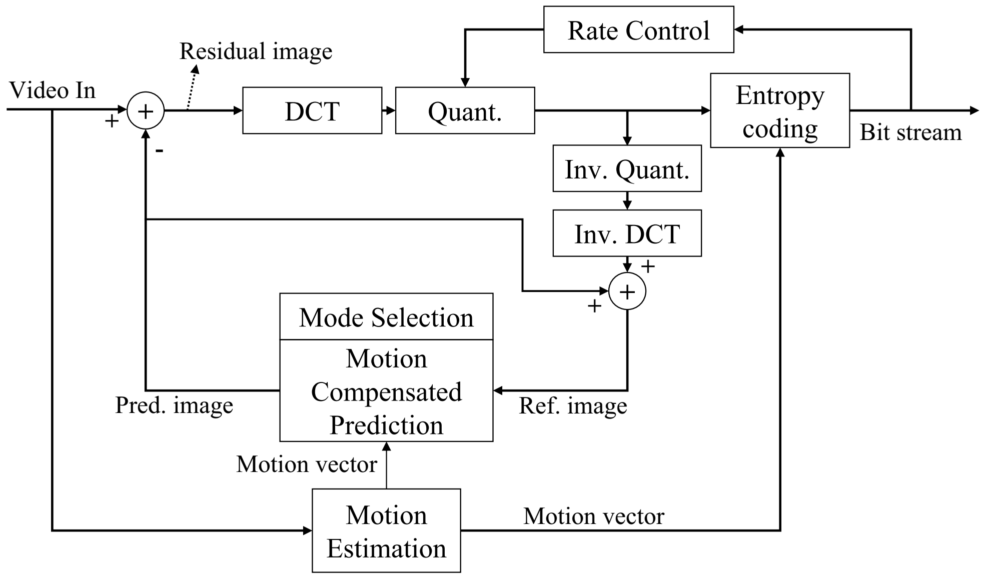


Figure 2: Hybrid encoder structure

In this chapter, we provide background information on each encoder module in typical hybrid encoders and a brief review of related works. In Section 2.1, we discuss general issues on motion estimation and provide a literature review on fast motion search algorithms. Section 2.2 presents a description on coding mode selection and a literature review on related research works. Other modules, transform, quantization, entropy coding, and rate control, are briefly described in Section 2.3, 2.4, 2.5, and 2.6, respectively. In each section, we also describe the new features introduced in the H.264 standard.

2.1 Motion Estimation

In block-based video coding, motion compensated prediction assumes that a block of pixels within the current image can be predicted from the previous encoded frames. Thus, most of motion estimation algorithms use block-matching techniques that obtain motion vectors by minimizing a cost function measuring the difference between a candidate block in the previous frames and the current original block. For comparison criteria, the Sum of Squared Difference (SSD) between the original and the reference block provides a good measure of the energy remaining in the difference block. However, Sum of Absolute Difference (SAD) is more commonly used than SSD because SAD can be easily calculated and also provide a good approximation of the residual energy.

The simplest, but the most computational intensive method, is the full search that evaluates the cost at every position in a search window. However, in most cases, search strategies faster than the full search are utilized although they lead to suboptimal solutions. Some examples of fast search algorithms include the three-step search, logarithmic search, cross search, nearest neighbors search [25, 26, 32, 38]. These fast search algorithms evaluate the criterion functions only at a predetermined subset of the candidate motion vector locations. Hierarchical representation of images in the form of a Laplacian pyramid may be used with the block-matching methods for fast motion estimation [23, 50]. The basic idea of hierarchical block-matching is to perform motion estimation at each level successively, starting with the lowest resolution level. The low resolution levels serve to determine a rough estimate of the displacement. The estimate of the motion vector at a low resolution is passed onto the next higher level as an initial estimate. The higher level serve to refine the motion vector estimate. At higher levels, relatively smaller search window can be used since motion searching starts with a good initial estimate.

A number of fast motion estimation techniques have been proposed to reduce computational complexity with minimizing degradation of prediction accuracy. The work by Chen and Willson [18] involves finding the optimal vector that minimizes a cost function that has both the distortion and the rate terms. They consider the inter-dependency between the macroblocks and generate a trellis of the candidate motion vectors and use the Viterbi algorithm to find the best motion vectors. The rate and distortion are explicitly computed for each candidate motion vector. Because of the high complexity of the above algorithm, they also proposed a faster but sub-optimal approach by choosing a finite depth and reduced state trellis. The PSNR gains in the range of 0.3-1.5 dB over TMN5 were reported with H.263 codec on CIF-format sequences at a rate of 10 Kbits/frame. In [19], Chen and Kung also considered the rate explicitly. Based on their observation that piecewise continuous motion field reduces the bit rate for differentially encoded motion vectors, a neighborhood relaxation method was proposed. They reported an average rate reduction of 13.9% when using a fixed quantization parameter with H.263 codec.

In [21], Chung *et al.* used parametric functions for the rate and the distortion to estimate the R-D curve and to solve for Lagrangian minimization problem. The functions are generated off-line, and as a result, the computational complexity is reduced. They suggested to estimate the Lagrangian multiplier by preprocessing some parts of the source. They also proposed a simplified mode decision. PSNR gains of 2.5 dB and 2.0 dB were reported with H.261 and Telenor H.263 codecs, respectively, for Miss America sequence.

In [56], Subramanian and Chan used the same parametric functions and chose a fixed value for the Lagrangian multiplier within a frame to solve for R-D optimal motion estimation. The Lagrangian multiplier was predicted from the previous frame. The proposed motion estimation algorithm was reported to achieve 0.6 dB and 1.1 dB PSNR gains over TMN5 algorithm of H.263 when fixed quantization parameters of 20 and 12, respectively, were used to encode Salesman sequence.

Shen and Chan suggested reduction in the number of the candidate motion vectors of the exhaustive R-D optimal motion search to achieve a faster but suboptimal motion estimation in [54]. They reported 0.4 - 0.6 dB PSNR gains at the expense of 100 % increase in computational complexity with Car Phone and Foreman sequences, respectively, compared to TMN5 H.263 encoder.

In [31], Hu *et al.* achieved 0.4-0.7 dB PSNR gains with their proposed model-based R-D optimal motion compensation approach on Car Phone and Foreman sequences, respectively. They used polynomials for rate and distortion as a function of the quantization parameter, and employed motion vector pruning for complexity reduction.

It is self-evident that actual scene motion has arbitrary accuracy and is oblivious to the pixel grid structure resulting from spatial sampling at the image acquisition stage, for example, charge-coupled device arrays or other analog to digital post-acquisition operations. A theoretical and experimental analysis has established that sub-pixel accuracy has a significant impact on motion compensated prediction performance for a wide range of natural moving scenes [27]. As a consequence, recent standardization efforts in video compression have embraced the principle of sub-pixel accuracy for motion estimation and motion compensated prediction. Since searching on a sub-pixel grid obviously requires more computation than the integer-pixel accuracy motion search, in practice, most block search algorithms first locate the best matching block with a resolution of one integer pixel, and then in a separate and subsequent step, calculate the best possible block with a sub-pixel resolution. Many fast schemes have been proposed to decrease computational complexity in [20, 24, 33, 34, 42, 52, 53].

To reduce temporal redundancy further, many advanced features for motion estimation are employed in H.264 [15]. The standard supports the use of multiple reference pictures from which inter-prediction of macroblocks and blocks can be made. Multiple reference pictures may help prediction of transitionally covered background and periodic non-transitional motion. For sub-pixel motion estimation, quarter-pixel accuracy motion compensation is used. Six-tap interpolation filtering for the half-pixel positions is followed by bi-linear interpolation to derive the quarter-pixel positions. By using various block shapes for motion compensation, H.264 can increase the accuracy of motion estimation significantly. According to the standard, a macroblock can be partitioned horizontally and/or vertically, resulting in 16x8, 8x16, and 8x8 block shapes, and also each 8x8 block may be divided further into 8x4, 4x8, and 4x4 block shapes. More details on block sizes and various coding modes associated with the block shapes are discussed in the next section.

2.2 Coding Mode Selection

After the motion estimation module in Figure 2 finds the best motion vector(s)[1] for each mode, mode selection/motion compensated prediction module receives motion vector information from the motion estimation module. After the best mode is determine in the module, the best prediction image associated with the mode is sent to the next module.

The mode selection method used in the Test Model 5 (TM5) for MPEG-2 [8] uses the SADs that are associated with the best motion vector(s), and the variances of the residual images. The mode selection rule is shown in Figure 3. Although this approach is very simple and gives reasonable performance at high bit-rate, its coding efficiency at low bit-rate is not acceptable because of a lack of consideration of rate constraints.

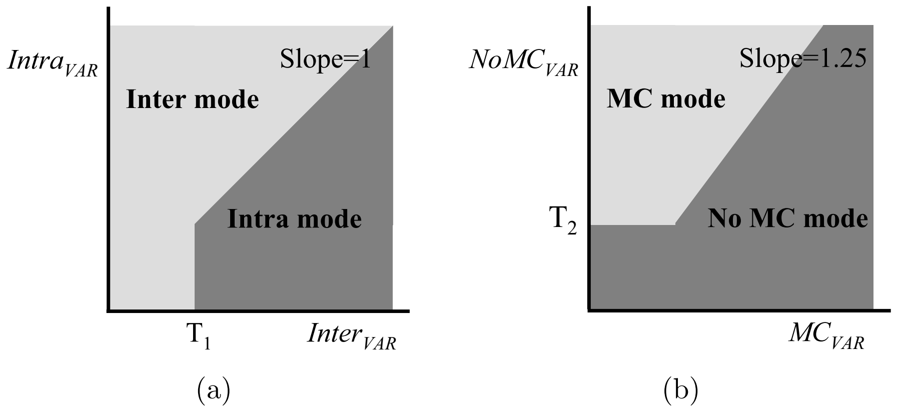


Figure 3: TM5 mode selection for (a) intra-mode vs. inter-mode and (b) Motion Compensated (MC) mode vs. No MC mode

A number of mode selection techniques have been proposed to improve compression efficiency. Recent developments [40, 45, 47, 48, 55, 57, 60] show that the coding efficiency of encoders can be improved significantly by applying rate-distortion theories to mode selection selection problem. In general, encoder performance is measured by its bit rate (R) and distortion (D) that is introduced by quantization and measured as the SSD between the original and the coded video frame. In a rate-distortion optimization framework, the goal is to minimize D subject to a rate constraint. This constrained problem reads as follows:

[1]The number of motion vectors for a coding mode varies depending on the video coding standards, coding structure, picture type, etc. For example, for an inter-mode in MPEG-2, there is one motion vector for progressive prediction and two for interlaced prediction in P-type pictures.

$$\min\{D(R)\} \quad \text{subject to} \quad R \le R^*, \text{ where } R^* \text{ is the allowed rate.} \tag{1}$$

In [60], Wiegand *et al.* provided a solution to Eq. 1 for a given group of blocks, where the quantization parameter and mode selection are jointly optimized. The search for the best combination of modes is viewed as an equivalent search for the best path in a trellis. This work was extended in [45], where a near-optimal M-best-search Viterbi-type algorithm was proposed to efficiently find a set of rate-distortion optimal modes. In the proposed algorithm, the rate and distortion are explicitly computed. The PSNR gains in the range of 0.5-1.0 dB over Test Model Near-term 5, version (TMN5) for H.263 [10] of H.263 were reported for the encoding rates of 10 to 50 Kbps.

Sun *et al.* proposed a heuristic approach for the mode selection in [57], wherein a spatial-masking-activity weighted quantizer scale was used as the distortion measure. The optimal mode was selected to minimize the overall rate subject to uniform distortion over the picture, given a set of motion vectors. A near-optimum algorithm was proposed for obtaining an upper bound to achievable performance, and a suboptimal practical algorithm was provided for efficient mode selection. PSNR gains in the range of 0.4-1.0 dB over TM5 were reported on Bicycle, Flower Garden, and Football sequences with the proposed suboptimal algorithm, compared to 0.5-1.2 dB gains with the near-optimum algorithm using the MPEG-2 encoder at 3 Mbps.

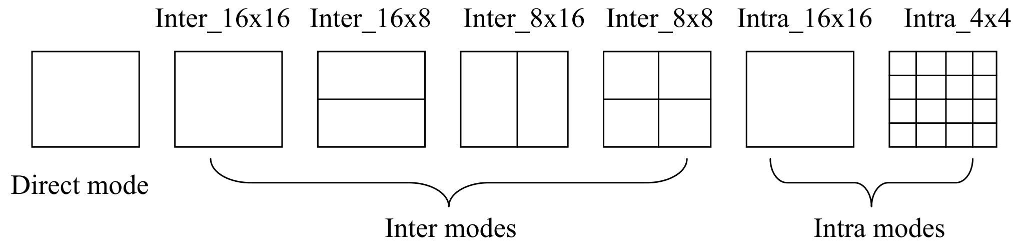
Peel *et al.* proposed the estimation of the Lagrangian multiplier as a function of the buffer fullness, and provided a locally optimal joint motion estimation and mode selection in [47]. PSNR gains up to 3.5 dB were reported at low bit rates for high-motion QCIF-format sequences.

Lee *et al.* presented a fast encoding method for MPEG-2 compliant encoding of interlaced video in [40]. They substituted SAD for distortion D and used a single adaptive λ based on the generated number of bits and/or quality constraint to reduce the calculation for rate-distortion costs. The proposed algorithm outperformed the TM5 coder by 1.5-2.0 dB, and total encoder computation can be reduced up to 10% of the total computation of TM5 by employing a fast motion estimation algorithm. However, the computational savings come from the fast motion estimation, and the proposed mode selection method requires higher computational complexity than TM5 because it needs to perform transform and quantization for entropy coding.

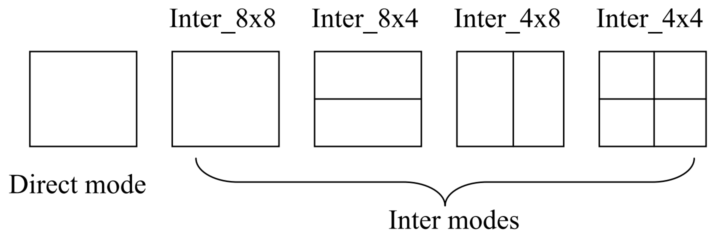
In Test Model Near-term 8 (TMN8) for H.263 [13], SAD and the number of bits only for motion and header information are used for D and R, respectively. Although this yields a performance level within an average of 0.5 dB of that of the rate-distortion optimized mode selection algorithm, it is computationally very efficient. Yang *et al.* also proposed a low-complexity mode selection technique using variance-rate and variance-distortion tables in [62]. With the table-lookup, the computation for obtaining R and D can be reduced dramatically. Combined with the trellis-based mode selection approach, the proposed method provides better PSNR performance by 0.8 dB for 16Kbps, 0.5 dB for 24 Kbps, and 0.3 dB for 32 Kbps over TMN8 for QCIF videos of 50 frames each.

As mentioned in Section 2.1, various block shapes are supported in H.264. All modes associated with the block shapes for luma component are illustrated in Figure 4. There are two intra-prediction modes, which are denoted as Intra_16x16 and Intra_4x4. The Intra_16x16 does spatial predictions of a 16x16 luma block and the Intra_4x4 consists of 16 4x4 luma blocks that are separately predicted. There are four spatial prediction modes in the Intra_16x16. In Intra_4x4, there are nine prediction modes for each 4x4 block. For inter-frame prediction, each macroblock mode corresponds to a specific partition of the macroblock. Five candidate modes are available including direct mode. For 8x8 inter-prediction mode, which is denoted as Inter_8x8, each of the four 8x8 blocks is split further in four ways. Figure 4(b) shows the five candidate modes for an 8x8 block in B-type frames.

In general, selecting a mode with a large partition size means that a small number of bits for motion information is required. However, motion estimation may not be accurate resulting in generating a large number of bits for sending transform coefficients. On the contrary, selecting a mode with a small partition size may require a small number of bits needed to signal residual information but produce a large number of bits for motion vectors and side information.



(a) Block shapes for 16x16



(b) Block shapes for Inter_8x8

Figure 4: Various block shapes in H.264

2.3 Transform

For highly correlated sources such as natural images, the compaction ability of Discrete Cosine Transform (DCT) is very close to that of the optimal transform, Karhunen-Loeve Transform (KLT). Moreover, DCT is data independent, unlike KLT that uses a set of data-dependent basis vectors. For these reasons, DCT is employed in all the current video coding standards to de-correlate blocks of original pixels or motion compensated difference pixels and compact their energy into a few coefficients as possible.

Besides its relatively high de-correlation and energy compaction capabilities, DCT is efficient and amenable to software and hardware implementations. The most common algorithm for implementing 8x8 DCT is 8-point DCT transformation of the rows followed by 8-point DCT transformation of the columns. Theoretically, exact reconstruction of the original data can be obtained through Inverse DCT (IDCT) operation. However, it is often not possible because of finite-precision arithmetic. Thus, the standards specify a minimum level of precision that IDCT errors must meet to avoid IDCT mismatch between the reconstruction frames at the encoder and decoder.

The MPEG-2 standard allows a frame or field DCT option for each macroblock in a frame picture[2]. Figure 5 shows residual macroblocks for frame and field DCT types. This allows computing DCT on a field-by-field basis for specific parts of a frame picture. For example, field DCT type may be chosen for macroblocks containing high motion, whereas frame DCT type may be appropriate for macroblocks with little or no motion but containing high spatial activity.

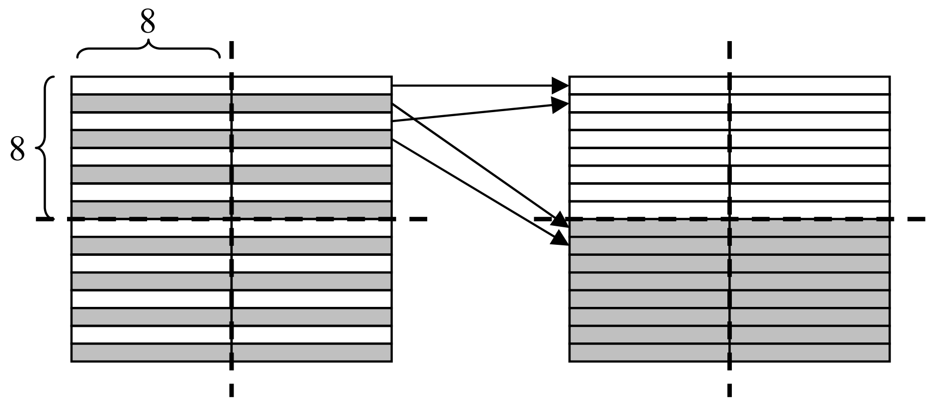


Figure 5: Frame DCT type (left) and field DCT type (right) for luma component in the MPEG-2 standard

In the H.264 standard, 8x8 transform is replaced with 4x4 integer transform that completely eliminates IDCT mismatch problem [43]. The 4x4 integer transform also has lower complexity because it allows computation of the forward or inverse transform with just additions and a minimal number of bit-shifts, but no multiplications. Figure 6 shows forward and inverse 4x4 transform.

2.4 Quantization

Transform is followed by coefficient quantization, the stage at which loss of video detail is traded-off against the video compression ratio. Because human eyes are less sensitive to reconstruction error related to high spatial frequencies than those related to low frequencies, quick high frequency changes can not often be seen and may be discarded while slow linear changes in intensity or color are important to the eyes. Thus, the basic idea of

[2]Interlaced video composed of a sequence of even and odd fields separated by a field period. MPEG-2 defines two picture types, frame picture and field picture. Frame pictures are obtained by interleaving lines of even and odd fields, and field pictures are simply the even and odd fields treated as separate pictures.

$$Y = C_f X C_f^T \otimes E_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} X \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

(a) Forward transform, a = 1/2, b = sqrt(2/5)

$$X' = C_i \left(Y \otimes E_i \right) C_i^T = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left(Y \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

(b) Inverse transform, a = 1/2, b = sqrt(2/5)

Figure 6: 4x4 transform in H.264. E is a multiplication matrix and symbol $\otimes$ indicates scalar multiplication. In actual process, the scalar multiplication is incorporated into quantization and only matrix multiplication is performed in transform.

the quantization is to eliminate as many of the nonzero DCT coefficients corresponding to high frequency component, resulting in a reduced variance of quantized DCT coefficients as compared to the variance of the original DCT coefficients as well as a reduction of the number of non-zero coefficients.

For each DCT coefficient matrix, the DCT coefficients are quantized using a quantization matrix, which contains the quantization step size for each DCT coefficient. The quantization matrix is obtained by multiplying a base matrix by a quantization parameter that typically controls the amount of compression and corresponding reduction in quality of the compressed video. The base matrix has small step sizes for low frequency components and large step sizes for high frequency components, which increases compression ratios with minimal visual effects. Each video coding standard specifies a default base matrix or an encoder can use a custom matrix specified in the sequence header. As an example, the base matrices for intra- and inter-modes used in TM5 are shown in Figure 7.

In the H.264 standard, quantization is combined with the normalization of the integer transform coefficients to avoid divisions at the encoder and to ensure 16-bit arithmetic data processing [43]. With the quantization scheme, a minimal computational complexity with no penalty in PSNR performance can be achieved.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

(a)

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

(b)

Figure 7: 8x8 quantization matrix for (a) intra-mode and (b) inter-mode used in TM5 for MPEG-2

2.5 Entropy Coding

Entropy coding reduces the number of bits required to represent the compressed video data through the use of means such as Variable Length Coding (VLC). The variable length codes are generated with Huffman codes such that shorter codewords are used to represent more frequently occurring symbols. In a typical hybrid encoder, the data to be coded by entropy coder falls into 3 categories: transform coefficients, motion vectors, and side information such as headers and synchronization marker. Usually, motion vectors are predicted by the already transmitted motion vectors of the neighboring blocks prior to entropy coding, and then only motion vector differences are encoded. For the quantized DCT coefficients, they are arranged into a one-dimension array by scanning them in zig-zag or alternate order prior to encoding. Figure 8 illustrates two scanning orders for an 8x8 quantized coefficient matrix. This arrangement places the DC coefficients first and the AC coefficients are ordered from low to high frequency and the array is coded using a three-dimensional run-length table, representing the triple, RUN, LEVEL, and LAST. The RUN means the distance between two nonzero coefficients in the array. The LEVEL is the nonzero value immediately following a sequence of zeros. The LAST is a flag that indicates whether the current code corresponds to the last coefficient in the array. This coding method increases the likelihood of grouping all nonzero coefficients together and facilitates run-length coding of the zero coefficients.

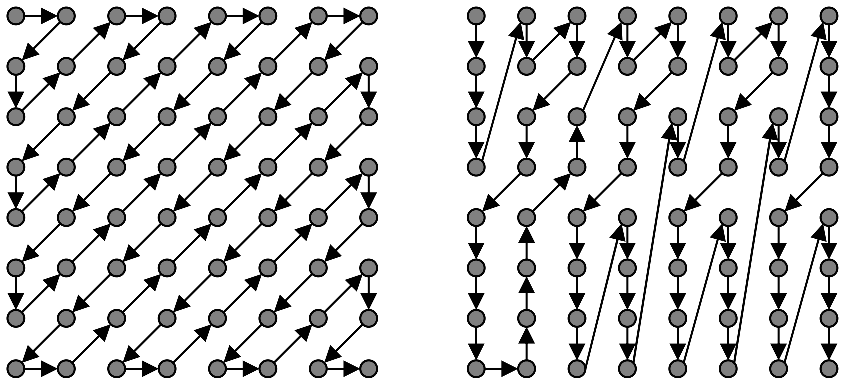


Figure 8: Scanning order for 8x8 block (a) zig-zag scan and (b) alternate scan

In the H.264 standard, Context Adaptive Binary Arithmetic Coding (CABAC) is introduced for higher performance [44]. An arithmetic code is more efficient than a variable length code for symbol probabilities that are much greater than 50% because it permits a symbol to be represented with less than one bit. Adaptivity also reduces the inefficiency of non-stationary symbol statistics caused by mismatch between static codeword length and actual symbol probabilities that change depending on bit-rate, type of motion, and other factors. The H.264 standard also introduces Context-adaptive Variable-Length Coding (CAVLC), which is a lower-complexity alternative to CABAC. Although CAVLC has lower complexity than CABAC, it is more elaborate and more efficient than the methods typically used in other prior standards.

2.6 Rate Control

In general, video encoders are designed to operate in Constant Bit-Rate (CBR) mode or Variable Bit-Rate (VBR) mode. A CBR bit stream is created by adapting quantization parameters to produce a constant bit-rate. On the other hand, in the VBR mode, the quantization parameters are nearly static to maintain uniform quality producing a variable bit-rate. In both cases, the goal of rate control module is to keep the output bit stream within constrained limits[3] while achieving maximally uniform video quality with minimal visual artifacts such as blocking, jitter, and flickering.

[3]There exists a maximum bandwidth even in the VBR mode.

The optimal solution to the rate control problem requires explicit knowledge of the video source properties. However, since this knowledge is not available before encoding, many conventional rate control algorithms use rate-distortion models that estimate the rate and quality of the output of the video coder. In [58], a theoretical rate-distortion function for a Gaussian random variable with a squared error distortion model is used to derive a logarithmic rate-quantization model. The proposed algorithm was reported to achieve 0.5 1.2-dB PSNR improvements over the TM5 rate control algorithm. The computational complexity of the proposed algorithm was reported to be 25% more than TM5.

In [49], a rate model derived from the entropy of a Laplacian distributed random variable and a distortion model derived from quantization error of a uniform random variable are used to solve for the optimal set of quantization parameters. This algorithm was reported to outperform the previous rate-control algorithms for H.263 video.

He *et al.* presented a rate-distortion model based on the fraction of zeros among the quantized DCT coefficients and a rate control method based on these models in [28, 29, 30]. The experimental results of the method was reported to perform better than the TMN8 rate control algorithm both in terms of its buffer regulation performance and the resulting visual quality. An average of 0.3-dB PSNR improvement over TMN8 was reported for various video sequences.

Kamaci *et al.* present rate-distortion models based on Cauchy density approximation to the DCT coefficient distribution in [35]. Using the model, they developed a frame bit allocation algorithm that is capable of achieving an average of 0.24-dB PSNR gain compared to JM 8.4 rate control algorithm and an average of 0.33-dB PSNR gain compared to an improved TM5-based frame bit-allocation algorithm proposed for H.264 video coder.

CHAPTER III

RATE-DISTORTION OPTIMAL MODE SELECTION

As mentioned in Section 2.2, the simple variance-based mode selection method used in TM5 gives very poor performance at low bit rate in the rate-distortion sense because there is no consideration of rate. On the other hand, the Lagrangian rate-distortion optimization technique with actual information on rate and distortion can achieve significant PSNR improvements. However, calculating Lagrangian costs consisting of the rate and distortion makes the technique inappropriate for power-limited or computation-limited applications such as cellular phone or hand-handled devices. Therefore, we propose a low-complexity model-based mode selection technique, which estimates distortion for a given rate and chooses a mode that produces minimum distortion. The computational complexity of the proposed technique is very low like the method in TM5 because the technique requires only variance and some information on rate for coding motion vectors and headers.

We also present joint coding mode and motion vector selection. In general, rate constrained motion estimation is necessary for low bit rate encoding. One of the more widely used methods is Lagrangian minimization, whose cost function consists of SAD and rate for motion vectors. For a given Lagrangian multiplier and a coding mode, the motion estimation module produces a motion vector that minimizes Lagrangian cost. In our work, multiple Lagrangian multipliers are used and therefore the motion estimation module generates multiple motion vectors. Then, we select one of the candidate motion vectors in the process of mode selection.

3.1 Rate-Distortion Model

Wiegand *et al.* applied the Lagrangian optimization technique to mode selection in [60]. The constrained problem in Eq. 1 can be converted to an unconstrained one by introducing a Lagrangian multiplier λ, as follows:

$$\min\left\{J(D,R)\right\}, \quad \text{where} \quad J \ = \ D+\lambda R. \tag{2}$$

The proposed Lagrangian optimization scheme for coding mode selection requires calculating rate and distortion, which means DCT, quantization, entropy coding, and all reconstruction processes should be performed to determine an optimal mode. This approach may not be appropriate for power-limited or computation-limited applications because of its computational complexity. In this thesis, instead of employing the Lagrangian optimization technique, a model-based mode selection approach is taken. We consider the rate-distortion functions of the form

$$D(R) = \eta\sigma^{\beta}e^{-\gamma R}, \tag{3}$$

where σ denotes the standard deviation of the source and η, β, and γ are unknown parameters. This form of rate-distortion function is chosen based on two observations:

- *ρ-domain analysis:* Based on the assumption that the AC coefficients of a natural image have a distribution that is best approximated by a generalized Gaussian distribution [39, 63], He *et al.* proposed two models for rate and distortion as a function of the fraction of the zeros among the DCT coefficients [28, 30]. The proposed rate model is simple, linear, and yet accurate:

$$R = \theta\left(1-\rho\right), \tag{4}$$

  where R is a normalized rate for quantized transform coefficients, θ is a model parameter, and ρ is a normalized number of zero quantized coefficients. Similarly, for the distortion, an exponential relationship as $D\left(\rho\right) = \sigma^2 e^{-\alpha(1-\rho)}$ is suggested, where σ^2 is the source variance and α is an unknown constant. It was also reported that α depends on the picture type and content. Combining both parametric equations, we reach the following rate-distortion function:

$$D\left(R\right) = \sigma^2 e^{-\gamma R}, \tag{5}$$

  where $\gamma = \frac{\alpha}{\theta}$. This simple model was proven to be effective for rate control in [30].

- *Actual encoding statistics:* By encoding various video sequences at different rates, we obtained actual rate-distortion curves. Based on the actual rate-distortion relations, we conclude that an exponential form of Eq. 3 is valid if model parameters are accurately estimated. Figure 9 shows the variation of γ as a function of the rate for intra- and inter-coding, respectively, when using the model of Eq. 3 with $\eta = 1$ and $\beta = 2$.

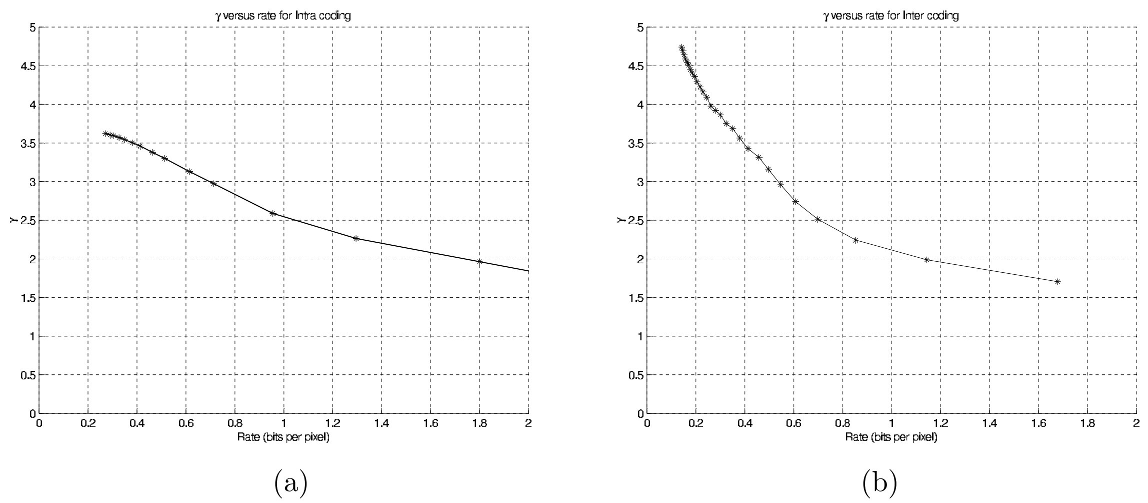


Figure 9: Experimental analysis of change of γ with respect to the rate for (a) intra-coding and for (b) inter-coding. The experimental values are obtained using a large set of randomly selected macroblocks.

Information-theoretic analysis also suggests further justification for this model. As mentioned above, the distribution of the AC coefficients of a natural image is best approximated by a generalized Gaussian distribution. For a Gaussian source, the rate-distortion function is given by [22]

$$R = \frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right), \tag{6}$$

where distortion, D, is a SSD between the original and the reconstructed image. It immediately follows that the distortion resulting from the quantization of the AC coefficients, D^{AC}, is given in terms of the rate to encode the AC coefficients, R^{AC}, as

$$D^{AC} = \sigma^2 e^{-2\ln(2)R^{AC}}. \tag{7}$$

This equation suggests that for a perfect Gaussian source, $\gamma = 2\ln(2)$. We use γ as a coding parameter to control accuracy since the actual AC coefficient distribution is not exactly Gaussian. We consider two methods for controlling the model parameter γ:

- *Adaptive estimation:* We can estimate the actual γ value for a frame by using the encoding statistics of previous frames of the same picture type. In this case, γ values for intra- and inter-coding are estimated using previous intra- and inter-coded macroblocks. Let $(D_{act,intra}, R_{act,intra}, \sigma^2_{act,intra})$ be the actual distortion, rate, and source variance of the last intra-coded frame, respectively. Similarly, let $(D_{act,inter}, R_{act,inter}, \sigma^2_{act,inter})$ be the actual distortion, rate, and source variance of the last inter-coded frame, respectively. Then, the estimates of γ are

$$\begin{aligned} \gamma_{intra} &= \frac{1}{R_{act,intra}} \ln\left(\frac{\sigma^2_{act,intra}}{D_{act,intra}}\right), \\ \gamma_{inter} &= \frac{1}{R_{act,inter}} \ln\left(\frac{\sigma^2_{act,inter}}{D_{act,inter}}\right). \end{aligned} \tag{8}$$

- *Rate-dependent lookup table approach:* Our experiments have shown that there is a strong relation between the coding rate and γ. Thus, instead of estimating γ for each frame adaptively, a simpler way is to use a lookup table to choose γ for a given frame bit budget. The values of γ in Figure 9 can be used for this approach.

By using the rate-distortion model in Eq. 3, we can obtain a computationally efficient method for rate-distortion optimized mode selection. We also include rate-distortion optimal motion vector selection in the process of mode selection.

3.2 Coding Mode Selection in MPEG-2

Consider a group of N macroblocks to be encoded in a frame. Let m_i be the coding mode of the i^{th} macroblock, $(i = 1, 2, ..., N)$, and let M_N be the set of the modes of all macroblocks. Then,

$$M_N = \{m_1, m_2, \cdots, m_N\} \tag{9}$$

The problem of finding the rate-distortion optimal set of the modes, M_N^*, for the group of N macroblocks can be formulated as

$$M_N^* = \arg\min_{M_N} D(M_N),$$

subject to

$$R(M_N) \leq R^{total}, \tag{10}$$

where $D(M_N)$ and $R(M_N)$ represent the sum of the distortions and the rates of N macroblocks, respectively. R^{total} is the available total bit budget to encode the set of N macroblocks. The bit budget is shared to encode the DCT coefficients, the motion vector, and the header information. Solving the constrained minimization problem of Eq. 10 using trellis-type algorithms will yield the set of rate-distortion optimal modes of the N macroblocks. However, the resulting computational complexity will be very high. To avoid such high computational workload, we assume that current macroblock mode, m_i, is independent of any of the other macroblocks. Thus, we can modify the constrained minimization problem of Eq. 10 as

$$M_N^* = \arg\min_{M_N} D(M_N) = \arg\min_{M_N} \sum_{i=1}^{N} D_i(m_i) = \sum_{i=1}^{N} \arg\min_{m_i} D_i(m_i),$$

subject to

$$\sum_{i=1}^{N} R_i(m_i) = \sum_{i=1}^{N} R_i^{mv}(m_i) + \sum_{i=1}^{N} R_i^{dct}(m_i) + \sum_{i=1}^{N} R_i^{hdr}(m_i) + R^{misc} \leq R^{total}, \tag{11}$$

where $D_i(m_i)$ denotes the distortion of the i^{th} macroblock determined by m_i. Similarly, $R_i(m_i)$ represents the rate of the macroblock in the mode m_i. $R_i^{mv}(m_i)$, $R_i^{dct}(m_i)$ and $R_i^{hdr}(m_i)$ denote rates for motion information, DCT coefficients, and header information, respectively, associated with the i^{th} macroblock when it is coded in mode m_i. R^{misc} represents the rate for coding other information that is not related to the macroblocks, e.g., the sequence/picture/slice headers.

Further simplification is possible if we assume that the target total number of bits for the i^{th} macroblock, R_i^{total}, is known. With this assumption, the rate constraint simplifies to

$$R_i^{mv}(m_i) + R_i^{dct}(m_i) + R_i^{hdr}(m_i) \leq R_i^{total}, \quad \forall i = 1, ..., N. \tag{12}$$

Thus, the best coding mode, m_i^*, of each macroblock is obtained by solving the following constrained minimization problem:

$$m_i^* \;=\; \arg\min_{m_i} D_i(m_i),$$

subject to

$$R_i^{mv}(m_i) + R_i^{dct}(m_i) + R_i^{hdr}(m_i) \leq R_i^{total}. \tag{13}$$

The optimal mode for the i^{th} macroblock, m_i^*, can be determined by explicitly computing the distortion for each mode by encoding the macroblock. However, computing distortion requires DCT, quantization, inverse quantization, and inverse DCT that are computationally expensive processes. To alleviate the problem of evaluating the distortion for each mode, we instead use a rate-distortion model. In general, the overall coding distortion of a macroblock is composed of the distortions because of the quantization of the DC coefficients and the AC coefficients. The reason for the separate treatment of the DC and AC coefficients is that encoding the DC coefficients in an intra-type macroblock is different than the rest of the DCT coefficients. For an intra-type macroblock, the DC coefficients of the macroblock are quantized with a fixed step size[1] and differentially encoded. Thus, we can write the macroblock distortion as

$$D_i(m_i) = D_i^{DC}(m_i) + D_i^{AC}(m_i), \tag{14}$$

where $D_i^{DC}(m_i)$ and $D_i^{AC}(m_i)$ are the distortions of DC components and AC components, respectively. Note that the actual value of $D_i^{DC}(m_i)$ can easily be calculated without going through the encoding process. For the distortion of the AC components, we use the rate-distortion relation presented in Section 3.1:

$$D_i^{AC}(m_i) = \sigma_i^2(m_i) e^{-\gamma(m_i) R_i^{AC}(m_i)}. \tag{15}$$

for a given mode m_i. In this equation, γ takes different values for intra- and inter-modes, and $\sigma_i^2(m_i)$ is the variance of the AC coefficients, which depends on the mode m_i. We finalize the rate-distortion formulation by substituting Eq. 15 in Eq. 14 to get

$$D_i(m_i) = D_i^{DC}(m_i) + \sigma_i^2(m_i) e^{-\gamma(m_i)(R_i^{total} - R_i^{mv}(m_i) - R_i^{hdr}(m_i) - R_i^{DC}(m_i))}, \tag{16}$$

assuming that $R_i^{AC}(m_i) = R_i^{total} - R_i^{mv}(m_i) - R_i^{hdr}(m_i) - R_i^{DC}(m_i)$.

[1]Either 1, 2, or 4 can be selected by the user depending on the DC precision

Using this model in the optimization problem formulated in Eq. 13, we get

$$m_i^* \quad = \quad \arg\min_{m_i} \left\{ D_i^{DC}(m_i) + \sigma_i^2(m_i) e^{-\gamma(m_i)\left[R_i^{total} - R_i^{mv}(m_i) - R_i^{hdr}(m_i) - R_i^{DC}(m_i)\right]} \right\}. \quad (17)$$

Equation 17 formulates the rule for choosing the best coding mode for the i^{th} macroblock. The scheme of this rate-distortion optimal mode selection method is shown in Figure 10 when $D_i^{DC}(m_i)$ is ignored. To solve Eq. 17, we need the distortion resulting from DC coefficient[2] quantization (D_i^{DC}), total macroblock bits (R_i^{total}), variance (σ_i^2), of the prediction residual, and the number of bits for encoding the motion vector(s), header, and the DC coefficient. We also need to estimate the model parameters $\gamma(m_i)$ for intra- and inter-coded macroblocks, respectively.

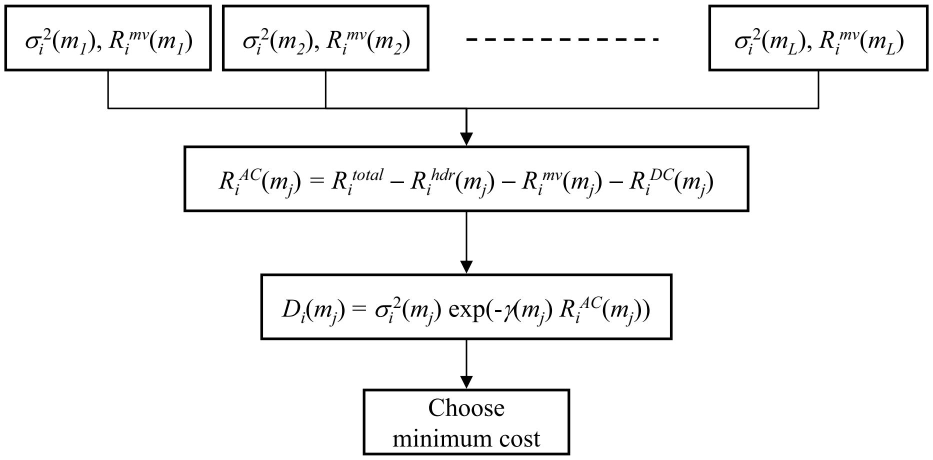

$\sigma_i^2(m_1), R_i^{mv}(m_1)$
$\sigma_i^2(m_2), R_i^{mv}(m_2)$
$\sigma_i^2(m_L), R_i^{mv}(m_L)$
$R_i^{AC}(m_j) = R_i^{total} - R_i^{hdr}(m_j) - R_i^{mv}(m_j) - R_i^{DC}(m_j)$
$D_i(m_j) = \sigma_i^2(m_j)\exp(-\gamma(m_j) R_i^{AC}(m_j))$
Choose minimum cost

Figure 10: RD optimal macroblock mode selection procedure. L is total number of possible macroblock modes.

At this point, we assume that motion estimation has already taken place before the mode selection step so that the motion vectors and the residual are determined already. Using the VLC tables, we can easily determine the number of bits for the motion vector. Similarly, the number of bits required to encode the header information can be estimated through lookup tables. Since the DC coefficient quantization does not depend on the choice of the quantization parameter, we can also determine the DC coefficient rate and distortion through simple calculations and lookup tables. The estimation of γ was already discussed

[2]Only intra-mode has DC coefficients.

in Section 3.1. Unfortunately, the total bit budget R_i^{total} is not known. One can estimate R_i^{total}. However, for the sake of simplicity, we assumed that the frame bit budget is equally distributed among all the macroblocks. Clearly, this is not a correct assumption. However, it still provides substantial gains within the context of the proposed mode selection algorithm, as we demonstrate in Section 3.5.

3.3 Joint Rate-Distortion Optimal Mode Selection and Motion Estimation

In Section 3.2, we assumed that the motion vectors are already determined by the motion estimation module. However, we can further improve the encoding performance by applying rate-distortion optimization to motion estimation in TM5. Let E_i be the SAD between the i^{th} original macroblock and the reference macroblocks. Clearly, E_i depends on the selected motion vector. Also, let R_i^{mv} be the rate for the selected motion vector. We can state the rate-distortion optimal motion estimation as estimating the motion vector that minimizes E_i, subject to a motion vector rate constraint. This constrained problem can be converted to an unconstrained problem using a Lagrangian multiplier. Thus, the rate-distortion optimal motion vector for the i^{th} macroblock is selected so as to minimize the Lagrangian cost function:

$$J_i = E_i + \lambda R_i^{mv}, \tag{18}$$

where λ is the unknown Lagrangian multiplier. λ can be viewed as a factor that determines the relative importance of the rate and the distortion terms. If $\lambda = 0$, then the rate constraint is ignored. Several techniques for Lagrangian rate-distortion optimal motion estimation and λ estimation were proposed in [21, 40, 56, 59].

The proposed approach is to use a set of λ values, $\lambda_1, \lambda_2, ..., \lambda_M$, and select motion vectors that minimize the cost J_i. That is, for each λ_k, $k = 1, 2, ..., M$, we can find a motion vector that minimizes the cost function $J_{i,k} = E_{i,k} + \lambda_k R_{i,k}^{mv}$, where $R_{i,k}^{mv}$ is the number of motion vector bits required to encode the selected motion vector for i^{th} macroblock. This process will result in a maximum of M candidate motion estimates. The motion vector estimate for different λ_k values can be identical. We can combine the mode selection and motion estimation by using all candidate motion vectors determined for each λ_k and optimize the macroblock mode and the motion vectors together by evaluating all possible cases by extending Eq. 17 as

$$\begin{aligned}(m_i^*, MV_i^*) &= \\ \arg\min_{m_i, MV_i} &\left\{ D_i^{DC}(m_i) + \sigma_i^2(m_i, MV_i) e^{-\gamma(m_i)\left[R_i^{total} - R_i^{mv}(m_i, MV_i) - R_i^{hdr}(m_i) - R_i^{DC}(m_i)\right]} \right\}, \quad (19)\end{aligned}$$

where MV_i and MV_i^* are a candidate motion vector and the best motion vector for i^{th} macroblock, respectively. Figure 11 shows the overall procedure for the rate-distortion optimized joint mode and motion vector selection when $D_i^{DC}(m_i)$ is ignored. The procedure is similar to the rate-distortion optimal mode selection case except that the number of candidates is increased. Producing a set of motion vectors (one for each λ) is not as costly an operation as it may appear at first. The main computational load in motion estimation is the calculation of E_i, which is independent of the λ value. This is made possible by the simplicity of the rate-distortion models that we use.

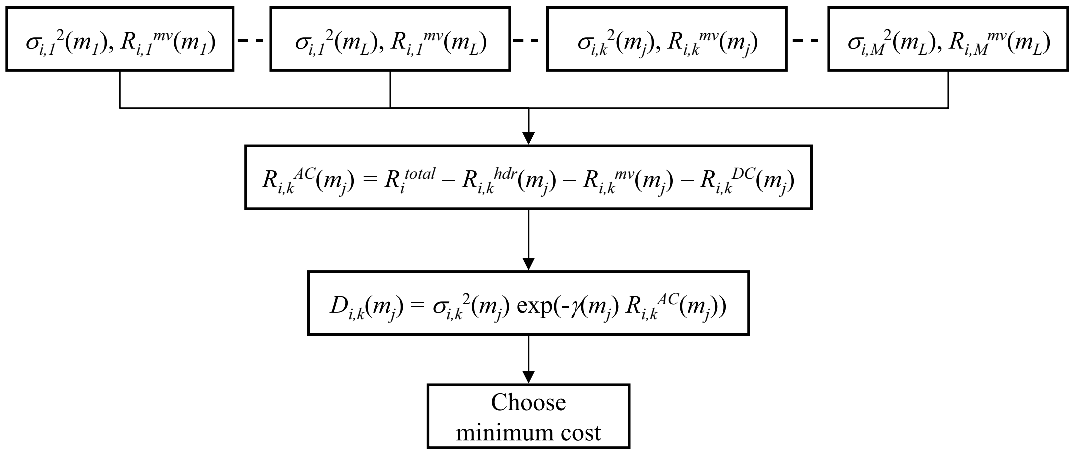

$\sigma_{i,1}{}^2(m_1), R_{i,1}{}^{mv}(m_1)$
$\sigma_{i,1}{}^2(m_L), R_{i,1}{}^{mv}(m_L)$
$\sigma_{i,k}{}^2(m_j), R_{i,k}{}^{mv}(m_j)$
$\sigma_{i,M}{}^2(m_L), R_{i,M}{}^{mv}(m_L)$
$R_{i,k}{}^{AC}(m_j) = R_i^{total} - R_{i,k}{}^{hdr}(m_j) - R_{i,k}{}^{mv}(m_j) - R_{i,k}{}^{DC}(m_j)$
$D_{i,k}(m_j) = \sigma_{i,k}{}^2(m_j)\exp(-\gamma(m_j) R_{i,k}{}^{AC}(m_j))$
Choose minimum cost

Figure 11: RD optimal joint macroblock mode selection and motion estimation procedure. L is total number of possible macroblock modes, M is the number of λ values used for optimization.

When N is the total number of macroblocks in a frame, e.g., $N = 1350$ for full D1 video resolution, and M is the number of λ values used in motion estimation, and L be the total number of possible macroblock modes, the optimal coding mode m_i^* and motion vector MV_i^* for the i^{th} macroblock, $i = 1, 2, ..., N$, are computed as follows:

1. Set macroblock index $i = 1$.

2. For each possible macroblock mode m_i, $m_i = 1, 2, ..., L$, repeat steps 2(a)-2(e):

   (a) If m_i is a mode that requires motion compensation, then perform a motion search. That is, estimate the best motion vector for each value of λ_k, $k = 1, 2, ..., M$, by minimizing $J_{i,k}(m_i) = E_{i,k} + \lambda_k R^{mv}_{i,k}(m_i)$. Compute the corresponding number of motion vector bits $R^{mv}_{i,k}(m_i)\ \ \forall k$, using the lookup tables. Go to step (c).

   (b) If m_i does not require any motion compensation (e.g., skip or intra-mode), then set $R^{mv}_{i,k}(m_i) = 0\ \ \forall k$.

   (c) Compute the residual variances $\sigma^2_{i,k}(m_i)$ for $\forall k$.

   (d) Compute the header bits, $R^{hdr}_{i,k}(m_i)$, and the DC bits, $R^{DC}_{i,k}(m_i)$, for $\forall k$, using the VLC tables.

   (e) Compute $D^{DC}_{i,k}(m_i)$ and estimate R^{total}_i, for $\forall k$, using the frame bit budget. (For simplicity, one can take $D^{DC}_{i,k}(m_i) = 0$ unless the bit-rate is very low.)

3. Compute the distortion $D_{i,k}(m_i) = D^{DC}_{i,k}(m_i) + \sigma^2_{i,k}(m_i)e^{-\gamma(m_i)R^{AC}_{i,k}(m_i)}$ for each possible mode and motion vector (only for MC case).

4. Choose the macroblock mode m^*_i and the motion vector MV^*_i (for MC mode only) that yield minimum $D_{i,k}(m_i)$:

$$\{m^*_i, MV^*_i\} = \underset{\{m_i, MV_{i,k}\}}{\arg\min}\ [D_{i,k}(m_i)]$$

5. Set $i = i + 1$. If $i \leq N$ go to step 2. Else stop.

3.4 Complexity Analysis on Model-Based Mode Selection

The computational complexity of the encoder is as important as its coding performance. In this section, we discuss the complexity of the proposed algorithm in Section 3.2. The proposed rate-distortion optimal mode selection and motion estimation algorithm does not consider a specific motion search method. The search can be full search, N-step search, or hierarchical search. The proposed algorithm requires additional computations such as computation of source variance and distortion, and determination of the rate terms. Since the exact computational complexity analysis depends on instruction sets of particular processor, here, we provide a simple theoretical analysis.

- **ρ -domain rate control complexity:** As discussed by He *et al.* in [30], the computational complexity of ρ-domain rate control is low. The distribution of DCT coefficients in current frame is needed at the beginning of the rate control. After each macroblock is coded, the distribution is updated using the coded macroblock statistics. The quantization parameter is determined using a simple rate model and a mapping between ρ and quantization step size. The model parameters are updated using actual coding bits and actual ρ. All of these operations require few calculations, and the resulting computational complexity is similar to that of TM5.

- **Calculation of number of bits for motion vectors:** In the MPEG-2 standard, motion vectors are coded differentially. So the number of bits can be easily obtained from a lookup table if the difference of motion vectors is available. Thus, we can ignore the computational complexity for this calculation.

- **Calculation of variances:** We need to compute the macroblock variance for each candidate mode and motion vector pair. For a 8×8 block, σ^2 is calculated using

$$\sigma^2 = \sum_{i=0}^{63} x_i^2 - \left(\frac{\sum\limits_{i=0}^{63} x_i}{8} \right)^2 , \tag{20}$$

  where x_i is the pixel value or the pixel difference value for intra-mode and inter-modes, respectively. To perform this computation, we need 66 multiplications and 127 additions per block for the intra-mode. In the case of a inter-mode, we need an extra 64 additions to compute the pixel differences, therefore, the total number of operations per block for one candidate mode is 66 multiplications and 191 additions for the inter-modes. For a 4:2:0-type macroblock, there are six blocks, thus the number of operations required per macroblock is 396 multiplications and 762 additions for an intra-mode, and 396 multiplications and 1146 additions for an inter-mode. For a P-type frame, we consider three inter-modes and one intra-mode, so the total number of operations per macroblock is 396×4 multiplications and $1146 \times 3 + 762$ additions. For a B-type frame, we consider six inter-modes and one intra-mode, so the total number of operations per macroblock is 396×7 multiplications and $1146 \times 4 + 762$ additions. If we use multiple λ values, then the number of operations per macroblock required to compute the variances associated with each mode and each motion vector candidate will be slightly higher. Finally, for P-type frames, if M number of λ values are used, there will be a total of $396 \times (3M + 1)$ multiplications and $3M \times 1146 + 762$ additions per macroblock. Similarly, for B-type frames, there will be a total of $396 \times (6M + 1)$ multiplications and $6M \times 1146 + 762$ additions per macroblock. In most of software or hardware implementations, all multiplications can be easily replaced with table lookup or highly optimized assembly functions.

- **Calculation of distortions and choosing a mode:** To calculate the distortion of an inter-mode using Eq. 16, we need two multiplications, four additions and one exponentiation. If there are M values of λ, the number of candidate modes per macroblock is $2M + 1$ and $6M + 1$ for P-type frames and B-type frames, respectively. Thus, we need a total of $7 \times (2M + 1)$ and $7 \times (6M + 1)$ operations per macroblock for P-type frame and B-type frame, respectively. This number is relatively small when compared to the number of operations required for computing the variance for each candidate macroblock mode and motion vector pair.

- **Estimation of the model parameter γ:** In the case where method 1 is used for rate-distortion relation, the model parameter γ is estimated using Eq. 8 after encoding a frame. This requires calculation of the frame distortion and frame variance. Assuming that only the luminance component is used, we need 256 multiplications and 511 additions per macroblock to calculate the frame distortion. For the frame variance, we can re-use the variances that are used in the mode decision process. If we use method 2, these calculations are avoided since γ is determined using a lookup table. All multiplications also can be replaced with table lookup.

We observe that the majority of the overall computational cost comes from the macroblock variance computation for each candidate mode and motion vector pair. A state-of-the-art hardware or software processor can easily realize this level of computational cost.

3.5 Experimental Results

To verify the effectiveness of the proposed mode selection and motion estimation algorithm, we incorporated it into the software TM5-based MPEG-2 encoder that was developed at The University of California, Berkeley [16]. We then compared its performance with that of TM5 and TM5 with the ρ-domain rate control algorithm proposed in [30]. In particular, a comparison with the ρ-domain rate control algorithm is included to demonstrate that (i) the proposed algorithm can work effectively with different rate control mechanisms and (ii) we can provide significant gains over one of the most-successful rate-control algorithms.

In the test set, we used five sequences with CCIR-601 parameters (720×480 pixels and a frame rate of 30 fps): Football, Flower Garden, Carousel, Susie, and News. In all experiments, the number of encoded frames was 103. We chose a GOP size of 15. We considered three bit rates of 2, 4, and 6 Mbps. The search range was selected as 63×63 for both P- and B-type frames, and the alternate scan option was turned on for all cases.

Table 3 shows the PSNR performance comparison for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed mode selection and motion estimation with ρ-domain rate control algorithm. In the ρ-domain rate control method, the bit-budget for each frame is allocated using TM5, but the allocation of the frame bit budget to macroblocks is achieved through a simple, but effective process as explained in [30].

In these tests, we used 10 different λ values for motion estimation, i.e., $M = 10$. The results indicate that significant PSNR gains can be obtained with the proposed mode selection and motion estimation algorithm. The modified TM5 algorithm with ρ-domain rate control achieves an average of 1.05 dB PSNR gain over TM5. The proposed algorithm achieves an additional average of 1.26 dB PSNR gain. Especially at 2 Mbps, the PSNR improvement of the proposed algorithm is quite substantial. The proposed algorithm achieves

Table 3: The average PSNR values for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed mode selection and motion estimation algorithm with ρ-rate control.

Sequence	Algorithm	2 Mbps	4 Mbps	6 Mbps
Carousel	TM5	28.35	33.35	35.44
	TM5+ρ	29.35 (+1.00)	34.72 (+1.37)	37.04 (+1.60)
	Proposed algorithm	32.10 (+3.75)	35.55 (+2.20)	37.61 (+2.17)
Flower Garden	TM5	26.38	30.17	32.15
	TM5+ρ	27.13 (+0.75)	31.28 (+1.11)	33.53(+1.38)
	Proposed algorithm	28.96 (+2.58)	32.01 (+1.84)	34.08 (+1.93)
Football	TM5	30.13	35.34	37.61
	TM5+ρ	30.47 (+0.34)	35.90 (+0.56)	38.27 (+0.66)
	Proposed algorithm	33.64 (+3.51)	36.90 (+1.56)	38.65 (+1.04)
Susie	TM5	37.09	40.41	41.45
	TM5+ρ	37.17 (+0.08)	41.20 (+0.79)	42.39(+0.94)
	Proposed algorithm	39.64 (+2.55)	41.64 (+1.23)	42.71 (+1.26)
News	TM5	33.10	35.94	37.27
	TM5+ρ	35.06 (+1.96)	37.62 (+1.68)	38.76 (+1.49)
	Proposed algorithm	36.38 (+3.28)	37.99 (+2.05)	39.07 (+1.80)

averages of 3.13 dB and 2.31 dB gains over TM5 and TM5 with ρ-domain rate control at 2 Mbps, respectively. Among the five sequences, the Carousel and Football sequences are richer in terms of motion and the PSNR gains for these sequences are higher. Overall, these results justify that the proposed rate-distortion optimization can provide significant benefits, especially at low rates.

In Table 4, we present the PSNR values when we used two different methods of γ estimation, as discussed in Section 3.1. In method 1, γ is estimated using the actual rate and distortion statistics of the previously encoded frame, whereas in method 2, γ is determined using a fixed lookup table. The results show that the performances of both methods are similar. Even though method 1 is slightly more complex, it can adapt to the sequence content. Method 2 does not provide such adaptation.

We also tested the performance of the proposed algorithm for various numbers of λs. Increasing the number of λs results in better encoding performance in terms of picture quality but at the cost of increased computational complexity. Table 5 shows the PSNR results for three cases. The results show an average of 0.13 dB PSNR loss when the number

Table 4: The average PSNR values when the two different γ adaptation method are used. The encoding parameters are the same as in Table 3.

Sequence	γ adaptation method	2 Mbps	4 Mbps	6 Mbps
CAROUSEL	Method 1	32.10	35.55	37.61
	Method 2	32.26	35.64	37.66
FLOWER GARDEN	Method 1	28.96	32.01	34.08
	Method 2	29.00	32.00	34.07
FOOTBALL	Method 1	33.54	36.90	38.65
	Method 2	33.48	36.97	38.69

of λs is decreased from 10 to 2, and another 0.11 dB loss when the number of λ values is decreased from 2 to 1.

Figures 12, 13, 14, 15, and 16 illustrate the PSNR values at each frame for the FOOTBALL, FLOWERGARDEN, CAROUSEL, SUSIE, and NEWS, respectively, where the bit rate is 2 Mbps, the GOP size is 15, the search range is 63×63 for both P- and B-type frames, and $M = 10$. In Figures 17, 18, 19, 20 and 21, one of the the reconstructed frames from FOOTBALL, FLOWERGARDEN, CAROUSEL, SUSIE and NEWS sequences are shown, respectively, for visual inspection. The proposed algorithm achieves remarkable visual quality improvement. Especially for the FOOTBALL sequence, the difference in the perceptual quality is significant. With the proposed algorithm, the motion vector bits are reduced by 50%- 85%. Finally, Figure 22 shows the resulting motion fields obtained by TM5 and the proposed algorithm for a frame of FLOWER GARDEN. As expected, the proposed algorithm produces smoother motion field.

Table 5: The average PSNR values of the proposed algorithm for different number of λ values. The encoding parameters are the same as in Table 3 except M

Sequence		2 Mbps	4 Mbps	6 Mbps
CAROUSEL	M=1	31.42	35.27	37.41
	M=2	31.97	35.45	37.52
	M=10	32.10	35.55	37.61
FLOWER GARDEN	M=1	28.19	31.68	33.83
	M=2	28.76	31.85	33.94
	M=10	28.96	32.01	34.08
FOOTBALL	M=1	33.48	36.97	38.69
	M=2	33.00	36.71	38.52
	M=10	33.64	36.90	38.65
SUSIE	M=1	39.28	41.47	42.60
	M=2	39.58	41.59	42.67
	M=10	39.64	41.64	42.71
NEWS	M=1	36.12	37.87	38.97
	M=2	36.35	37.96	39.05
	M=10	36.38	37.99	39.07

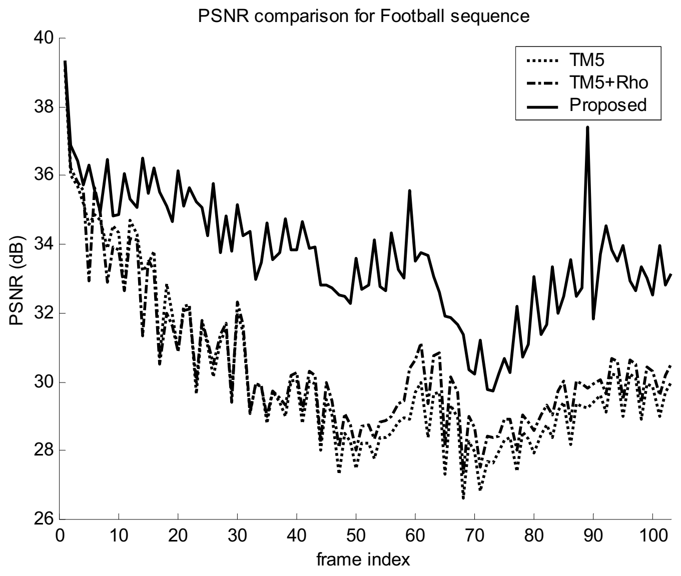


Figure 12: The PSNR value at each frame for FOOTBALL for (i) TM5, (ii) TM5 with the ρ-domain rate control, and (iii) the proposed algorithm with the ρ-domain rate control.

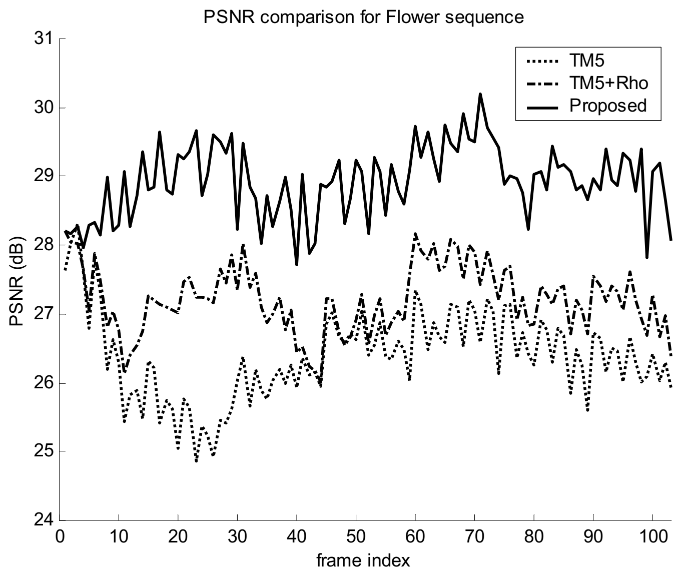


Figure 13: The PSNR value at each frame for FLOWER GARDEN for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control.

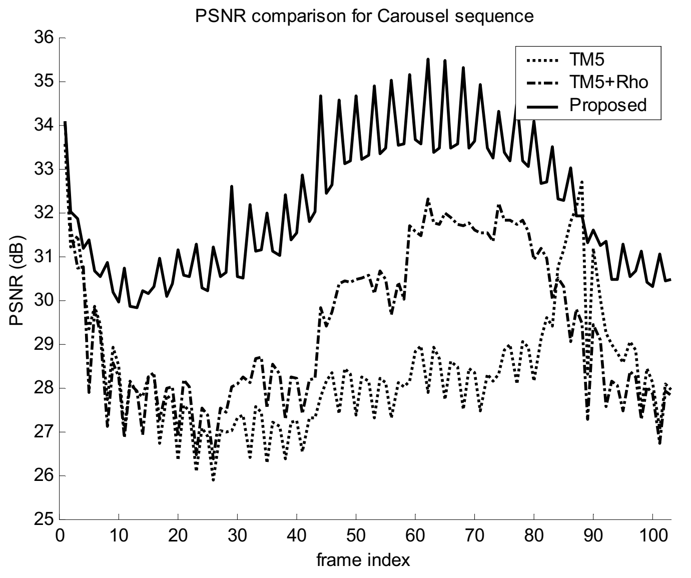


Figure 14: The PSNR value at each frame for CAROUSEL for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control.

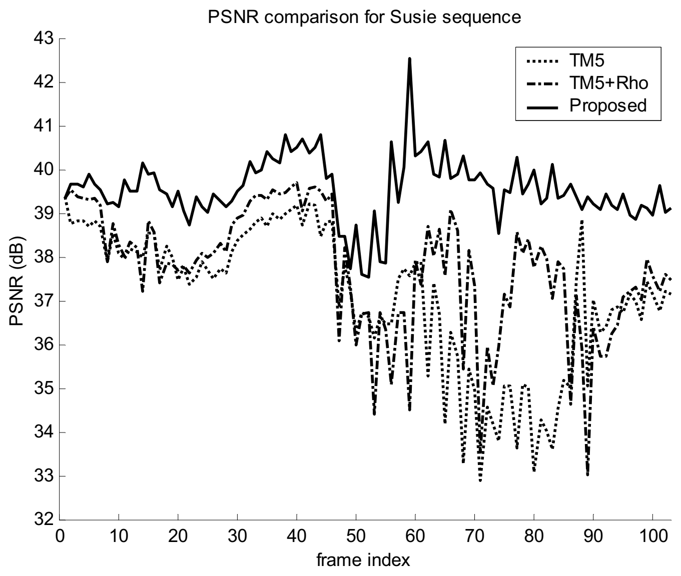


Figure 15: The PSNR value at each frame for Susie for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control.

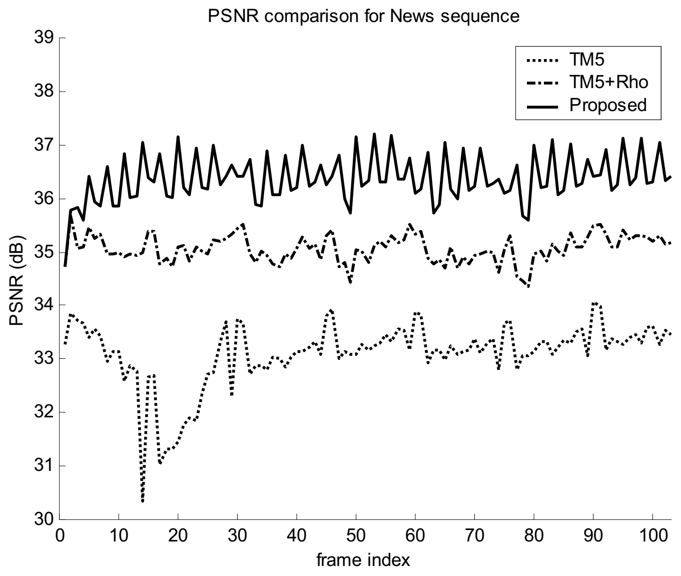
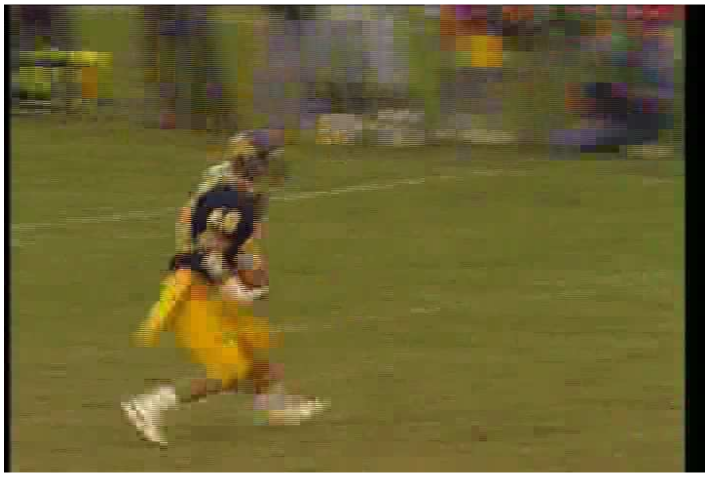
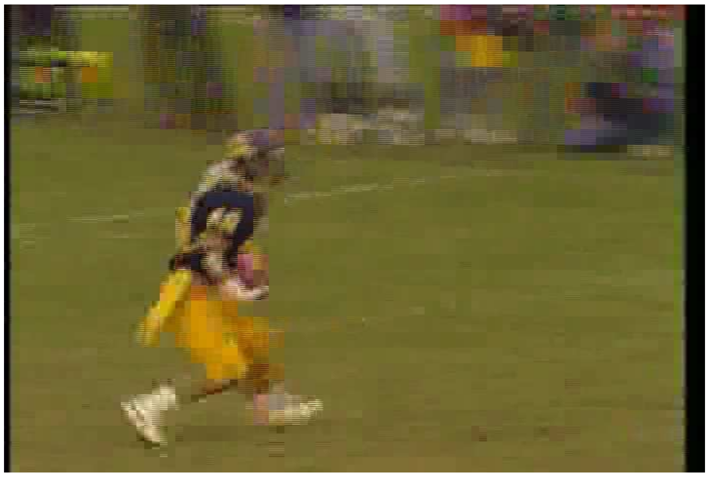


Figure 16: The PSNR value at each frame for News for (i) TM5, (ii) TM5 with ρ-domain rate control, and (iii) the proposed algorithm with ρ-domain rate control.


(a)


(b)

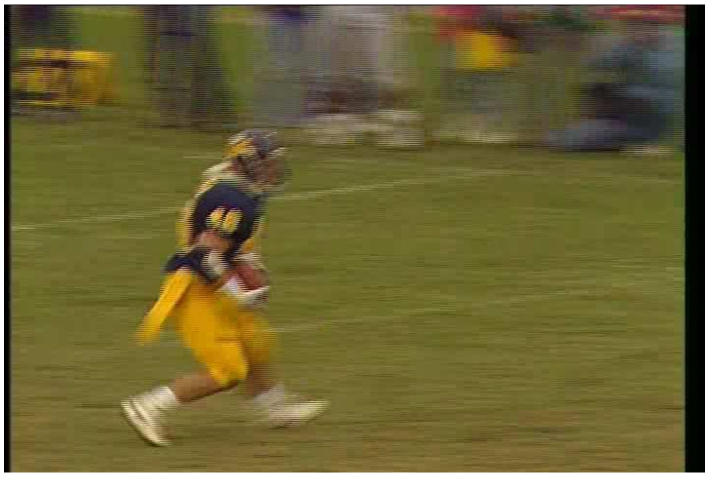
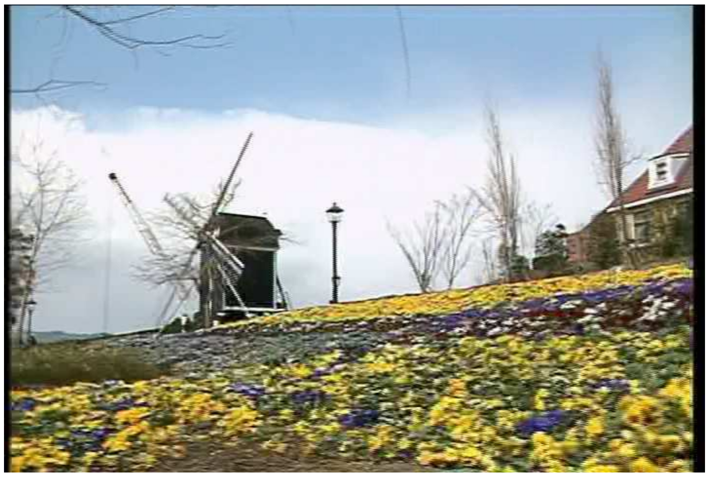
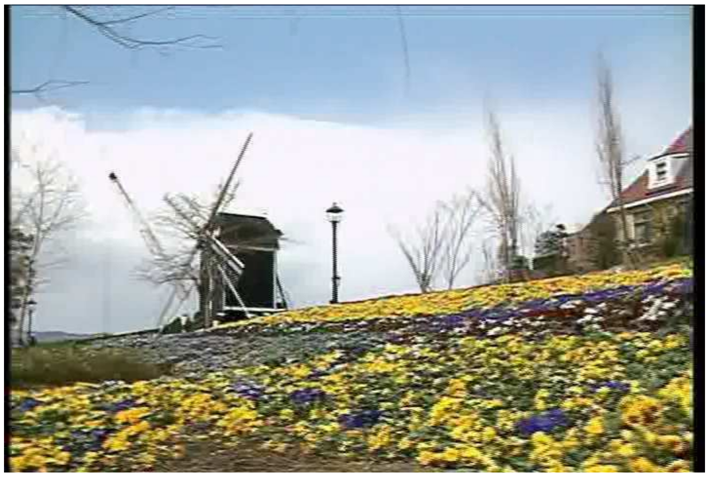

(c)

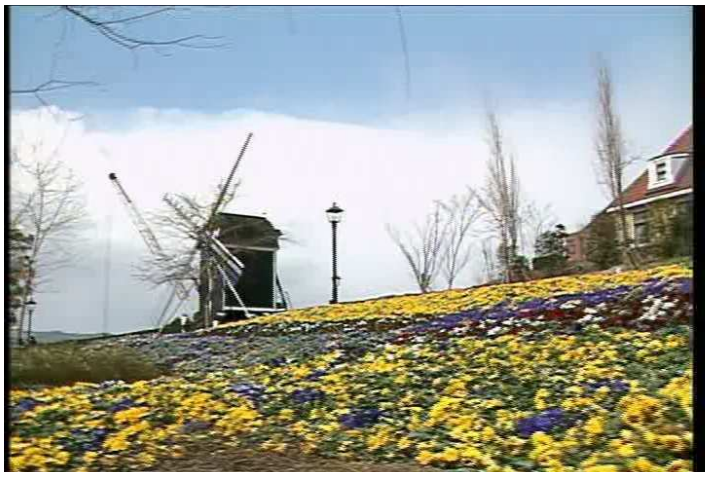
Figure 17: Reconstructed 318th frame of FOOTBALL when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 80112, 76608, and 42315 for (a), (b) and (c) respectively. The motion vector bits are 69969, 65657, and 13263 for (a), (b) and (c) respectively. The corresponding PSNR values are 31.9, 32.5, and 34.8 dB, respectively.



(a)

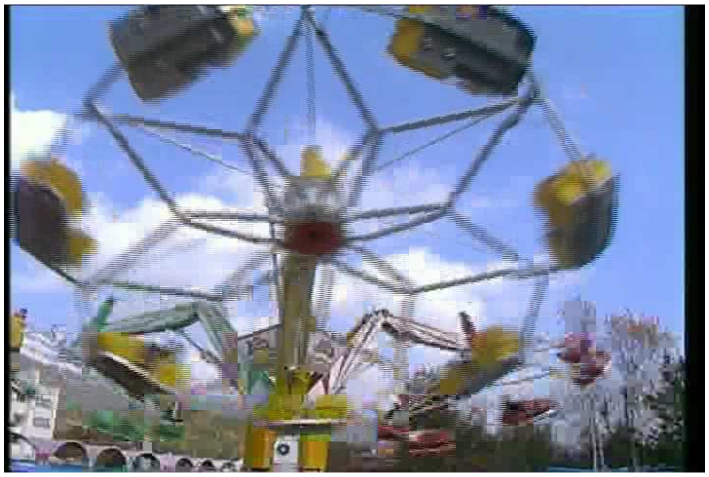


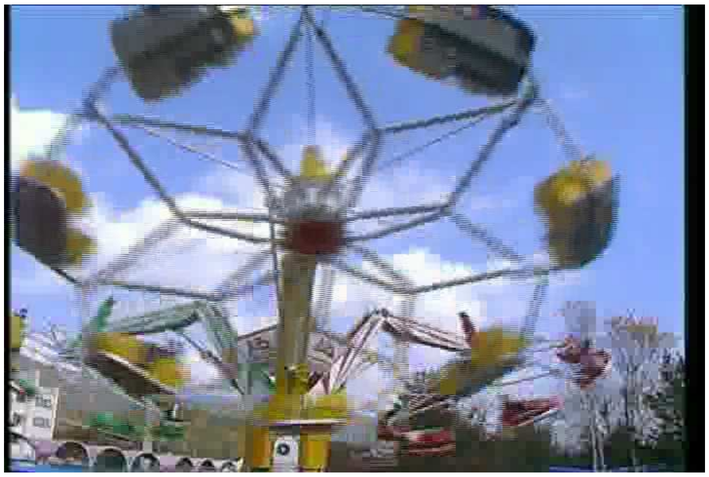
(b)



(c)

Figure 18: Reconstructed 320th frame of FLOWER GARDEN when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 48203, 52963 and 29204 for (a), (b) and (c) respectively. The motion vector bits are 38696, 37534, and 9509 for (a), (b) and (c) respectively. The corresponding PSNR values are 25.8, 27.0, and 29.3 dB, respectively.


(a)


(b)

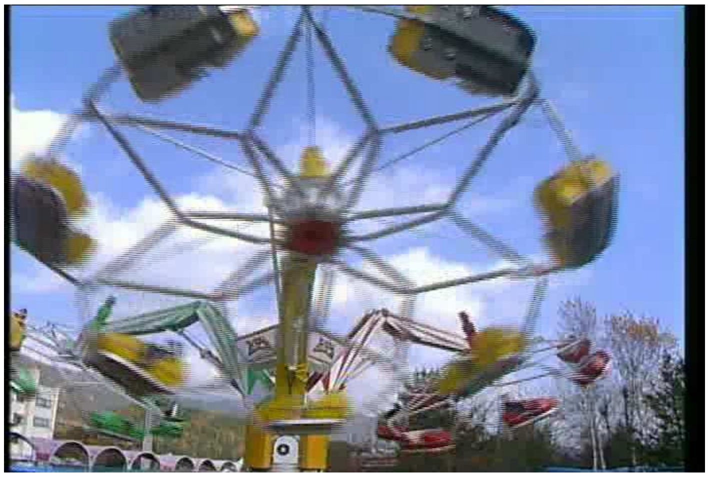
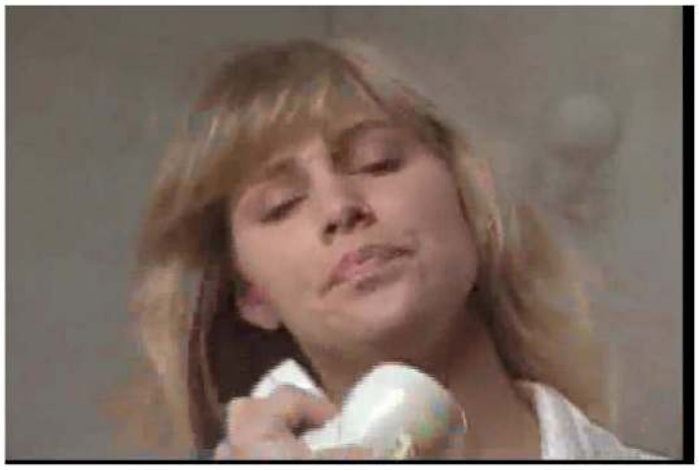
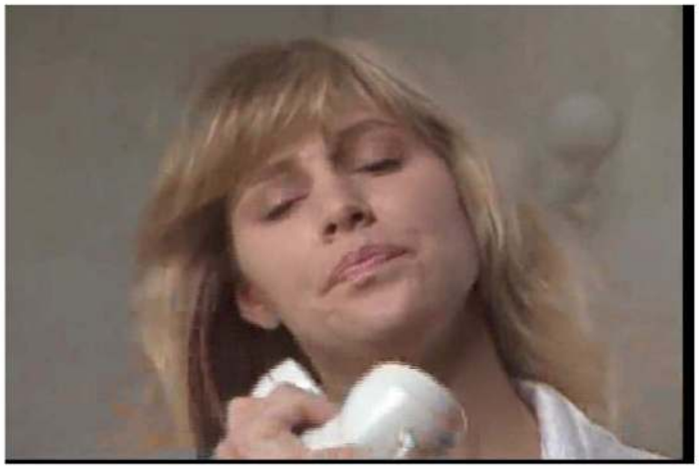

(c)

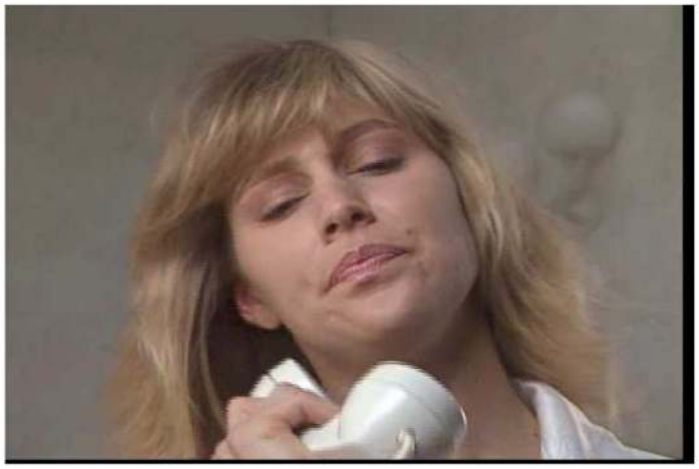
Figure 19: Reconstructed 318th B-type frame of the CAROUSEL sequence for (a) TM5, (b) TM5 with ρ-rate control, and (c) proposed algorithm with ρ-rate control. The frame bit budget is 68242, 65607, and 43011 for (a), (b) and (c) respectively. The motion vector bits are 58172, 55554, and 12423 for (a), (b) and (c) respectively. The corresponding PSNR values are 27.7, 27.3, and 30.2 dB, respectively.



(a)

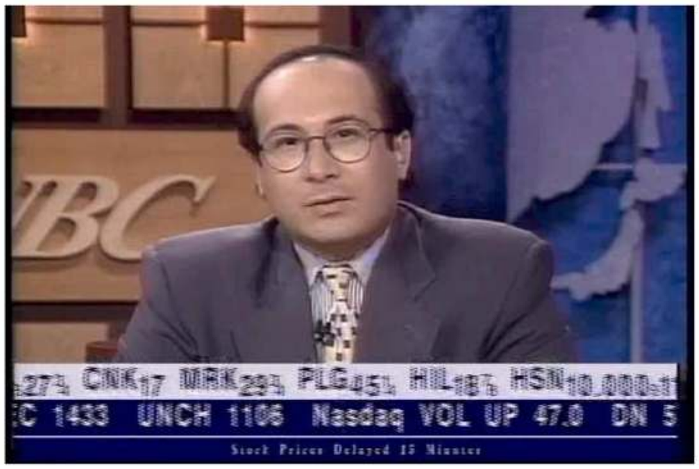


(b)

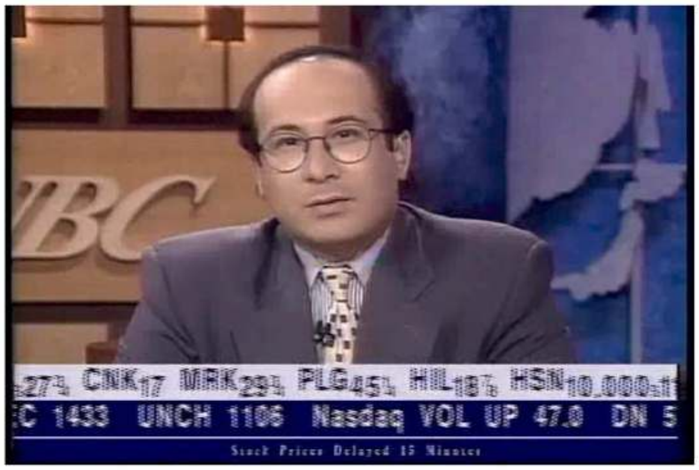


(c)

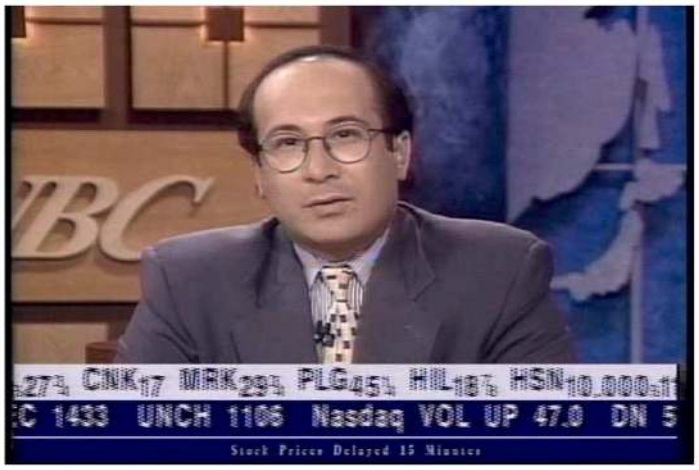
Figure 20: Reconstructed 171th frame of SUSIE when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 62895, 62511, and 34400 for (a), (b) and (c) respectively. The motion vector bits are 53100, 52958, and 10725 for (a), (b) and (c) respectively. The corresponding PSNR values are 34.6, 35.4, and 39.8 dB, respectively.



(a)

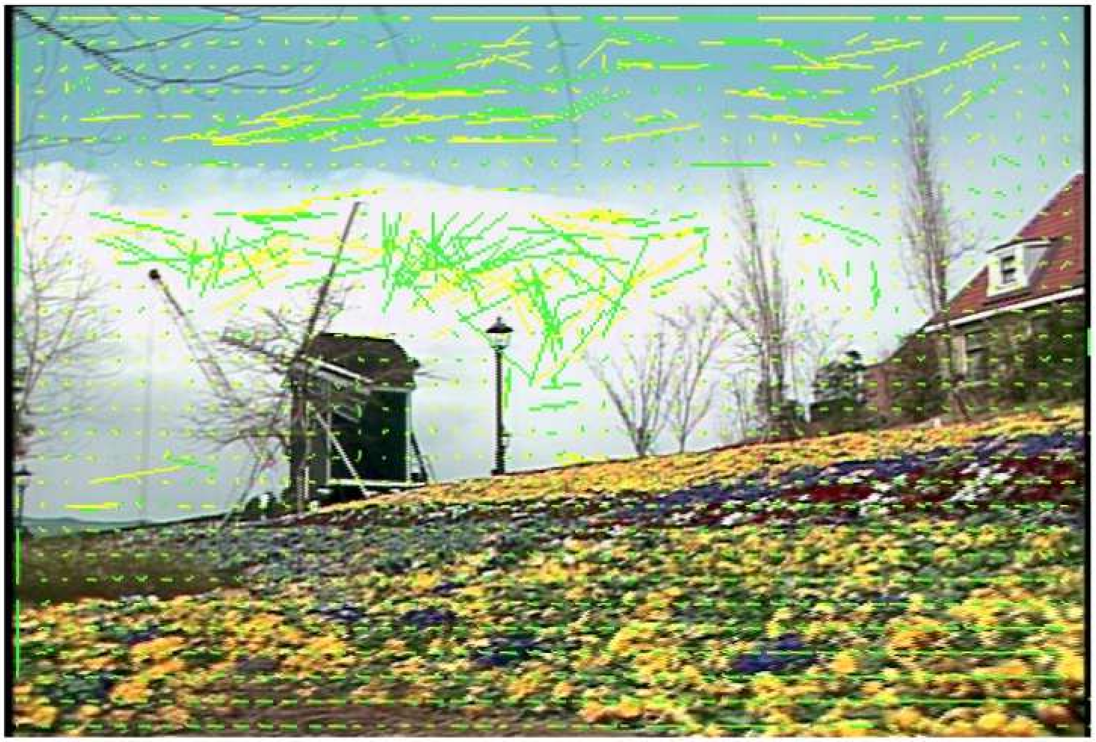


(b)

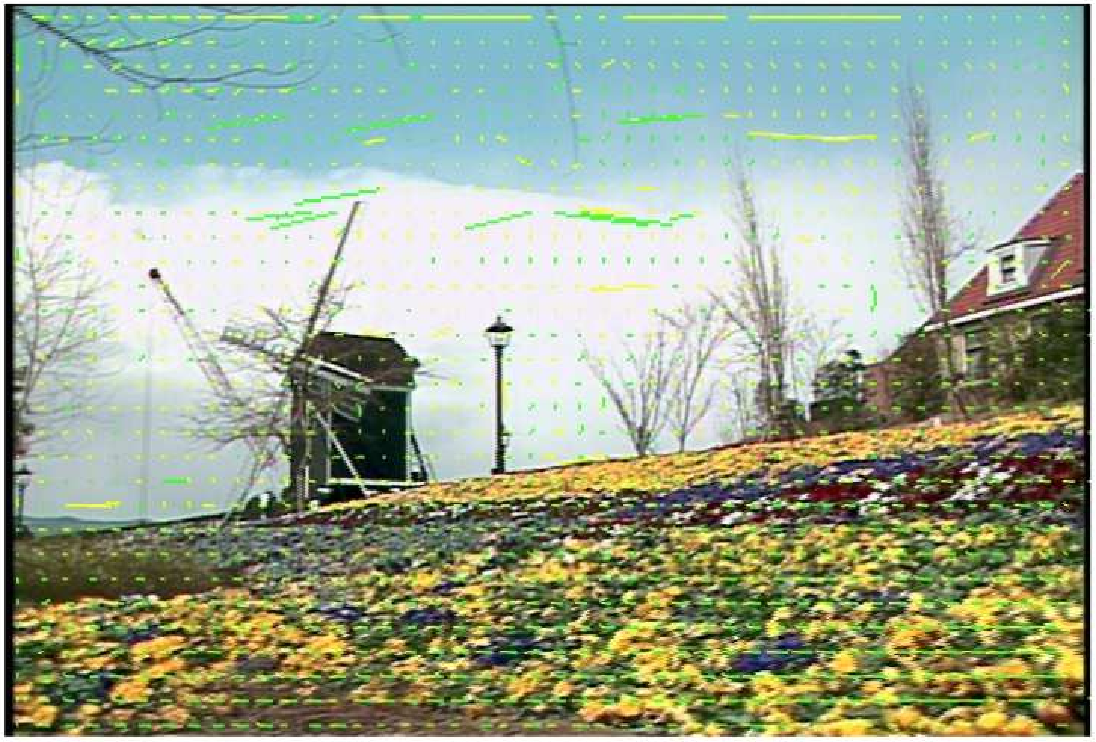


(c)

Figure 21: Reconstructed 123^{th} frame of News when it is coded using (a) TM5, (b) TM5 with ρ-domain rate control, and (c) the proposed algorithm with ρ-domain rate control. The frame bit budget is 61611, 60601, and 32083 for (a), (b) and (c) respectively. The motion vector bits are 42988, 43042, and 6562 for (a), (b) and (c) respectively. The corresponding PSNR values are 32.4, 35.1, and 36.3 dB, respectively.



(a)



(b)

Figure 22: The motion field of the 303th frame of Flower Garden when it is coded using (a) TM5 and (b) the proposed method.

CHAPTER IV

REAL-TIME VIDEO ENCODER IMPLEMENTATION ON DIGITAL SIGNAL PROCESSORS

Digital signal processors (DSPs) are widely used for digital audio, image, and video signal processing such as noise filtering, enhancement, compression/decompression, etc. because of their flexibility and re-configurability that are necessary for implementing complicated algorithms. As we discussed in Chapter 2, a digital video encoder consists of various algorithm modules, e.g., motion estimation, coding mode selection, DCT, quantization, entropy coding, etc. Therefore, a DSP platform can be one of the best solutions for real-time implementation of digital video encoders. However, to make maximum use of the advantages of DSPs, we need to understand the key features of the processors because the features essentially determine the architecture of a digital video encoder and the performance of the encoder. For example, it is almost impossible to implement a real-time video encoder on a DSP system without understanding special instructions for multiple data operations, hierarchical memory structure, or Direct Memory Access (DMA) devices provided by most of state-of-the-art DSPs.

The rest of this chapter is organized as follows. In the Section 4.1, we describe hierarchical memory structure that is one of the most important concepts in implementing real-time video encoders on DSP systems. Practical approaches for designing video encoder modules for MPEG-2 are provided in Section 4.2. Finally, Section 4.3 presents an overall structure for a real-time MPEG-2 video encoder on a DSP system.

4.1 Hierarchical Memory Structure

Most of state-of-the-art DSPs have a hierarchical memory structure, e.g., two-level internal cache based memory and external memory shown in Figure 23. Level-1 (L1) memory that is small but very fast is placed close to the CPU and can be accessed without CPU stalls. The next lower level memory, Level-2 (L2) memory, is much larger but slightly slower.

External memory that is located outside of the CPU is huge but very slow so that access to this memory causes CPU stalls. Through this type of architecture, the average memory access time can be closer to the access time of the fastest memory rather than the the access time of the slowest memory. In this section, we present some important features of memory structure in TI's TMS320DM64x/TMS320C64x device and how the features affect implementation of video encoders on the DSP systems [1, 2, 3].

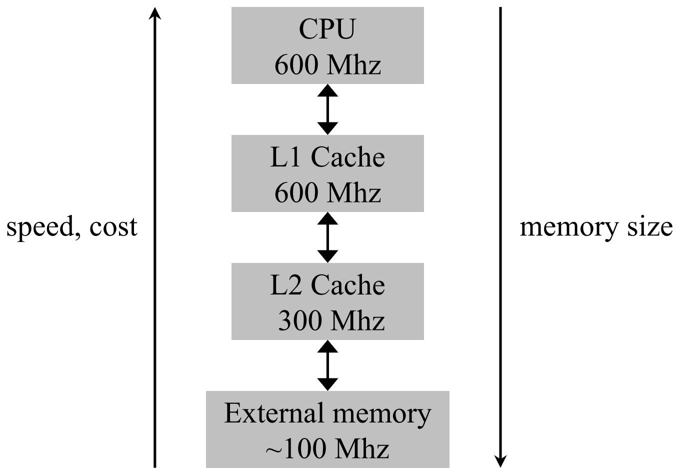


Figure 23: Hierarchical memory structure

4.1.1 Level-1 Memory

L1 memory is divided into program cache (L1P) and data cache (L1D). L1P that is a direct-mapped cache[1] services program fetches from the CPU. To minimize L1P cache miss, it is important that all the critical kernels including main control code can fit into the cache. Conventional implementation of a video encoder is based on macroblock-level processing, where the video encoder can process the next macroblock only after the current macroblock goes through all the processing modules shown in Figure 2. In this approach, because the overall program size, i.e. code size, is usually larger than the size of L1P, the code needs to swap between L1P and L2 cache every macroblock fetching period. This causes significant L1P cache miss penalty. To avoid the huge cache miss penalty, the encoder

[1]In a direct-mapped cache, each address in the lower-level memory is mapped to a single location in the cache. Multiple locations may map to the same location in the cache.

system needs to be split into several groups so that each of them can fit into L1P, and process N macroblocks at a time in each group instead of a single macroblock. Figure 24 illustrates an example encoder structure. In this structure, the encoder is divided into four loops, first loop is integer-pixel accuracy motion estimation, second loop includes sub-pixel accuracy motion estimation and mode selection, third loop includes transform, quantization and reconstruction, and fourth loop has entropy coding and rate control.

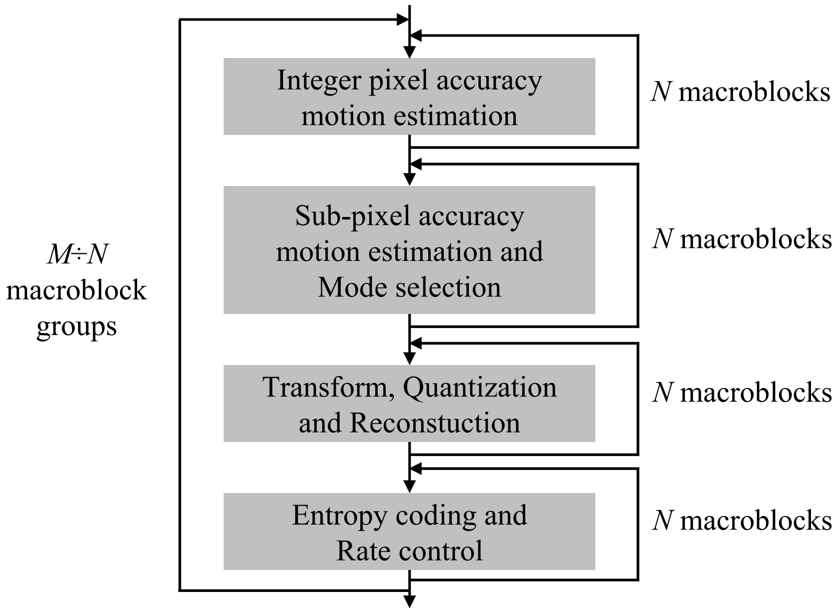

Integer pixel accuracy motion estimation
N macroblocks
Sub-pixel accuracy motion estimation and Mode selection
N macroblocks
M÷N macroblock groups
Transform, Quantization and Reconstuction
N macroblocks
Entropy coding and Rate control
N macroblocks

Figure 24: Sample encoder structure. M is the total number of macroblocks in a frame.

L1D that is a two-way set associative cache[2] services data access from the CPU. The size of a group of macroblocks, i.e., N, is determined by the available size of L1D in Figure 24. To minimize L1D cache miss, all data for N macroblocks should not be flushed out of L1D until a loop is finished. For example, in the third loop that includes transform, quantization, and reconstruction kernels, all required data for N macroblocks such as original image and prediction image is kept in L1D until the current loop is over. To determine N, it is necessary to know what kind of data is required and available L1D size for each loop. Using the larger N will reduce L1D cache miss and also increase the data transfer performance.

[2]In contrast to a direct-mapped cache, a multi-way set-associative cache selects a place for the data from a set of locations in the cache. A direct-mapped cache can be considered a single-way set-associative cache.

4.1.2 Level-2 Memory

When there is a large difference in memory size and access time between L1 cache and external memory, L2 memory is typically introduced to further reduce the number of accesses to the external memory. Figure 25 illustrates the hierarchical memory architecture in TI's TMS320DM64x/TMS320C64x.

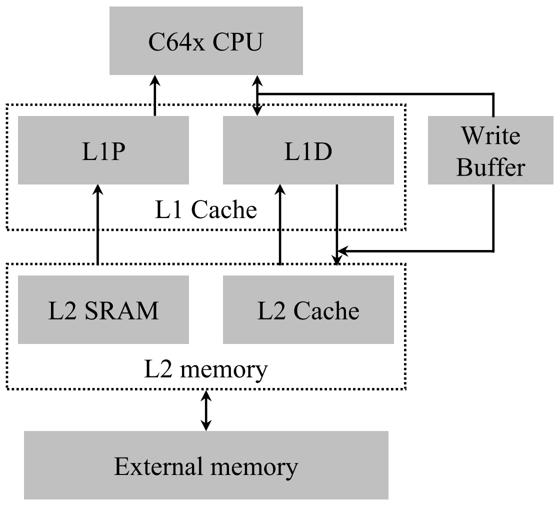


Figure 25: Hierarchical memory structure in TI's TMS320DM64x/TMS320C64x

In these DSP systems, L2 memory operates basically in the same manner for L1 cache but it has several different properties:

- It usually has larger capacity and slower. For example, TMS320DM642 has 16 Kbytes L1P, 16 Kbytes L1D, and 256 Kbytes L2 memory.
- L2 memory space can be split into an addressable internal memory (L2 SRAM) and cache portion (L2 cache). Five possible cache-SRAM configurations of TMS320DM642 are shown in Figure 26. The L2 cache-SRAM configuration is determined by the structure of an encoder, i.e. the size of data and code used in the encoder system.
- L2 cache is used to cache external memory addresses only while L1P and L1D can cache both L2 SRAM and external memory addresses.

One of the most important properties is the configurability of the L2 memory as SRAM

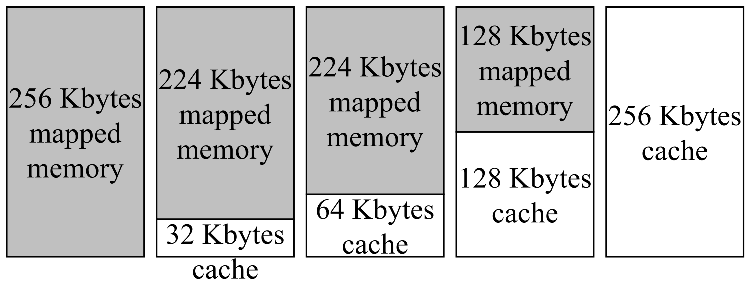


Figure 26: Cache-SRAM configurations for L2 memory in TMS320DM642

or cache, which allows designers to maximize the efficiency of their systems. For example, when we know exactly which data is required for the next process, the data can be copied from external memory to L2 SRAM in parallel to performing the current process by an enhanced DMA controller that operates independently without imposing extra burden on the CPU. Thus, after the current process is completed, the next process can be started without waiting for loading the data from external to L2 memory. Another example is that when a system needs data from a peripheral, if there is no L2 SRAM, this data has to be written into external memory. Then, when the CPU access the data, it will be brought into L2 cache. This setup is very inefficient. However, if part of the L2 memory can be used as memory-mapped RAM, we can store incoming data directly into L2 SRAM and process the data, rather than having to move it to external memory, cache it into L2, and then move it back external memory to send it out to the peripheral.

4.1.3 Enhanced DMA

An Enhanced Direct Memory Access (EDMA) controller is a peripheral that can be setup to transfer data or program from one place to another without the CPU's intervention. In EDMA communication, each data transfer is initiated by a transfer request that contains all the information required to perform the transfer, e.g., source address, destination address, transfer priority, element count, etc. Each submitted transfer request is serviced based on priority. EDMA ensures minimal CPU stalling cycles because the transfers are overlapping with processing. Figure 27 shows an example for a data transfer scheme between L2 SRAM and external memory. For the n^{th} macroblck, the first data, DataA(n), is required first to

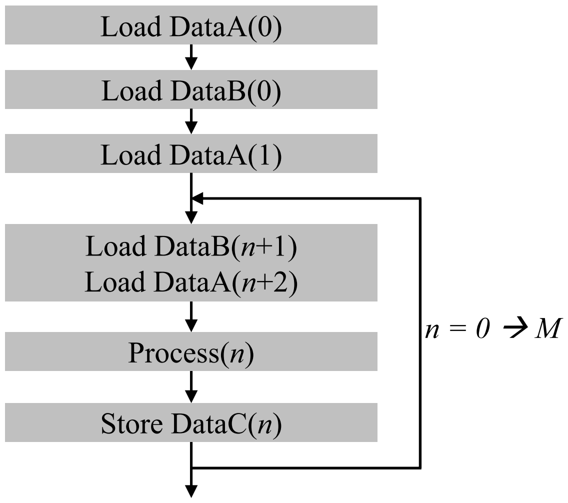


Figure 27: Sample data transfer scheme

determine the starting address of the second data, DataB(n). DataB(n), which is required to process the macroblck, is loaded from the external memory. The third data, DataC(n), is stored to the external memory while the next macroblock is processed. Usually, ping-pong memory buffer structure is used in this kind of scheme. For example, one buffer is used as an active buffer for data processing and the other one is used as a passive buffer for data loading or writing until processing the data in the active buffer is completed and the passive buffer is filled or emptied by EDMA. To minimize the waiting time for data transfer, we need to orchestrate the size of data to be transferred, transfer rate, transfer priority, processing time, etc.

4.2 Encoder Modules

Here, we propose various techniques for implementing encoder algorithms. In Subsection 4.2.1, we provide an analysis on memory requirements in a fast motion search scheme, which is a combination of hierarchical motion estimation and 3-step search. Subsection 4.2.2 presents implementation of our proposed mode selection. In Subsection 4.2.3, we describe an efficient reference area loading, implementation of interpolation, and SAD calculation in sub-pixel accuracy motion estimation. Finally, in Subsection 4.2.4, we briefly discuss miscellaneous implementation issues on transform and quantization.

4.2.1 Integer-Pixel Accuracy Motion Estimation

Video compression derives most of its coding efficiency from motion compensated prediction that removes huge temporal redundancy between frames. However, motion estimation process contributes the heaviest computational load in the whole video encoding. Thus, a good video encoder algorithm should keep a good balance between computational complexity and coding efficiency.

As described in Section 2.1, motion estimation consists of integer-pixel and sub-pixel accuracy motion estimation. Usually, first integer-pixel motion estimation is done and then sub-pixel motion estimation is performed around the estimated integer motion vector. For integer-pixel motion estimation, a real-world motion estimation generally uses a combination of fast motion search techniques. For example, a combination of hierarchical search and N-step search techniques can give good results with affordable computational cost. Here, memory requirement, which is one of the key factors in designing integer-pixel motion estimation module, is examined for 3-level hierarchical motion estimation method with 3-step search technique. All required data in a search window should be available in L2 SRAM to avoid access to external memory. Table 6 shows frame size and block size for luma component at each level for full D1 video (720x480).

Table 6: Frame size and block size in hierarchical motion estimation for full D1 frame

Level	Frame size	Block size
3	180*120 = 21600	4x4
2	360*240 = 86400	8x8
1	720*480 = 345600	16x16

As shown in the table, it may not be possible to put the whole reference frames in L2 SRAM. On the other hand, data prefetch on macroblock basis would be very inefficient because adjacent blocks have large overlapped reference area. Thus, a heuristic solution is to prefetch the reference area on macroblock group basis. As an example, we need 18x18 reference area at level-3 for a macroblock in 3-step motion search, which is depicted in Figure 28.

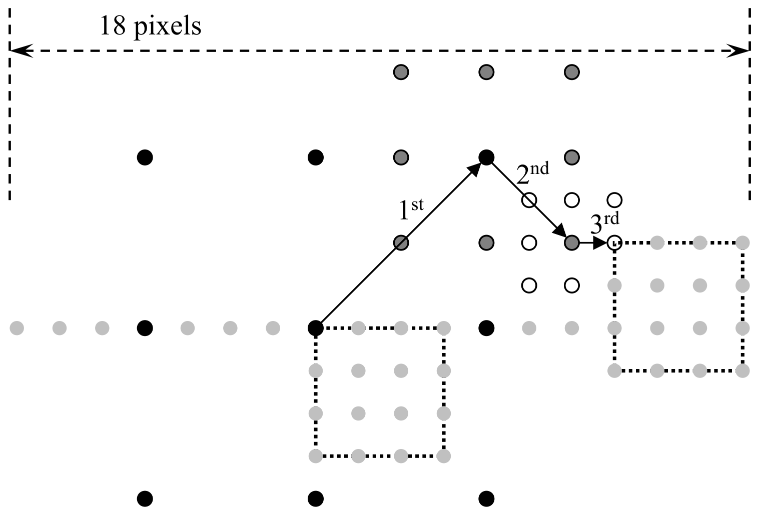


Figure 28: Three-step motion search at level-3

The required search area[3] at each level is shown in Figure 29. The maximum size of vertical displacement in one direction at level-3 is 7 that means 14 at level-2. Thus, the maximum vertical displacement at level-2 becomes 14+7 = 21. Similarly, 21 at level-2 means 42 at level-1, and the maximum vertical displacement at level-1 becomes 42+7 = 49. The amount of data required at each level is shown in Table 7.

Table 7: Memory requirement for macroblock line based prefetch scheme

Level	macroblock line size	Required memory size	ping-pong buffer
3	180x4 = 720	180 * (7+7+4) = 3240	3240*2 = 6480
2	360x8 = 2880	360 * (21+21+8) = 18000	18000*2 = 36000
1	720x16 = 11520	720 * (49+49+16)= 82080	82080*2 = 164160

Even though a ping-pong buffer structure is used, memory requirement for level-3 is quite small, and it may be possible to put the whole level-3 reference frame into L2 SRAM. Also, the required memory size at level-2, 36000 bytes, is not very large compared to the size of L2 memory provided by modern digital media processing purpose DSPs. However, the required memory size for level-1 motion estimation may not fit into L2 SRAM. In this case,

[3]The size of horizontal displacement does not need to be considered when reference area for one macroblock line is loaded.

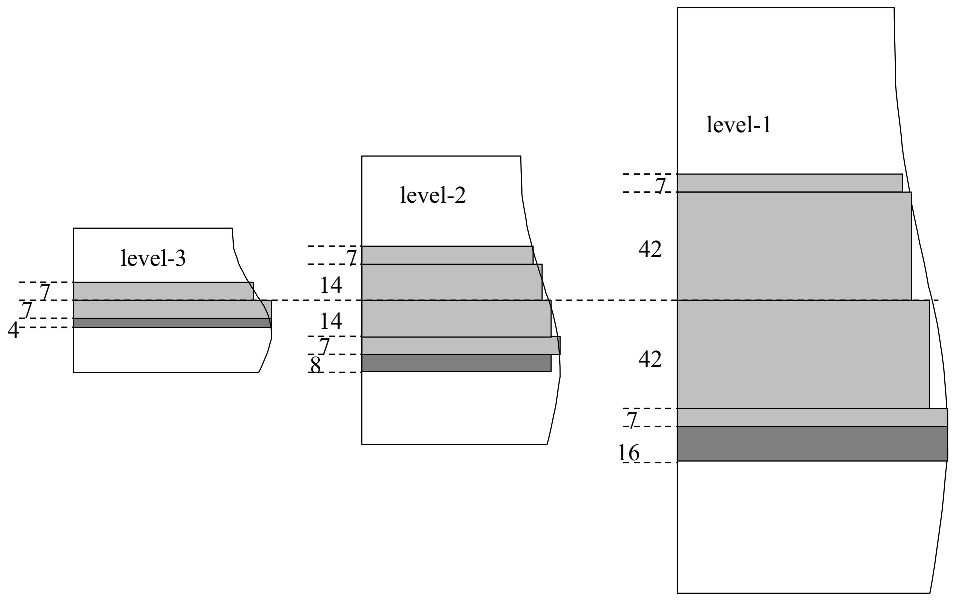


Figure 29: Required reference area at each level with macroblock line based prefetch

we may use smaller size of macroblock group or avoid using ping-pong buffer structure.

Here, we discuss the memory requirement for the combination of hierarchical motion search and 3-step search. Depending on the size of available memory or computational power, different search schemes can be employed at each level. For example, full search scheme may be used at level-3, and at each lower level, only +/- 1 pixels are searched around the coarse motion vector determined at higher level.

4.2.2 Macroblock Mode selection

In Section 3.4, we present theoretical analysis on computational complexity of our proposed mode selection method. Here, we address practical issues on calculating the required information such as variance and distortion for TI's DSPs.

Most of DSPs provide specialized C-functions, called intrinsics, that are mapped directly to functions in hardware by compilers [5]. By using the intrinsics, the full benefit of the hardware can be achieved since the compilers map an intrinsic to an extremely fast sequence of assembly instructions. As mentioned in Section 3.4, major computational cost comes from the variance calculation. The variance calculation can be very efficiently implemented by using two intrinsics[4], **_subabs4** and **_dotpu4**, that are highly optimized taking only 1 and 4 cycles, respectively. **_subabs4** calculates the absolute value of the differences between two packed 8-bit values, and **_dotpu4** returns a dot-product between two packed 8-bit values.

When x_i is a pixel difference between the original and prediction images, the SSD and sum of difference in Eq. 20 can be expressed as follows:

- $\sum_{i=0}^{63} x_i^2 = \sum_{k=0}^{15}$ **_dotpu4**$(abs\_sub_k,\ abs\_sub_k)$

  where $abs\_sub_k =$ **_subabs4**$(orig_k,\ pred_k)$, and $orig_k$ and $pred_k$ are 4-byte (32-bit) variables that contain k^{th} 4-byte blocks in the original and predicted images, respectively.

- $\sum_{i=0}^{63} x_i = \sum_{k=0}^{15}$ **_dotpu4**$(orig_k,\ \text{0x01010101}) - \sum_{k=0}^{15}$ **_dotpu4**$(pred_k,\ \text{0x01010101})$

The remaining calculation for σ^2 needs only one bit-shift, one scalar multiplication, and one subtraction. As mentioned in Section 3.4, a frame distortion, i.e. SSD between original and reconstructed frames, is required to update γ. Similarly, we can calculate the distortion by using two intrinsics.

4.2.3 Sub-Pixel Accuracy Motion Estimation

As mentioned in Section 2.1, sub-pixel accuracy motion estimation improves motion prediction accuracy significantly. Here, we discuss several implementation issues such as data loading from external to internal memory, pixel interpolation, etc.

To ensure an optimal data transfer performance, all data transfers should be 32-bit (4-byte) aligned. It means that both the starting address and transferring data size have to be a multiple of 4 bytes. Since a motion vector can point to any pixel position in the reference frame, the alignment of the reference block is not known. For example, when the reference block size is 10x10, we have to consider four cases shown in Figure 30. In this example, the starting address is same for all the four cases, and a 16x10 reference block has to be transferred with an offset. The offset of the transferred block need to be considered in the pixel interpolation step.

[4]Appendix A describes the intrinsics used in our algorithm implementation with illustrative figures.

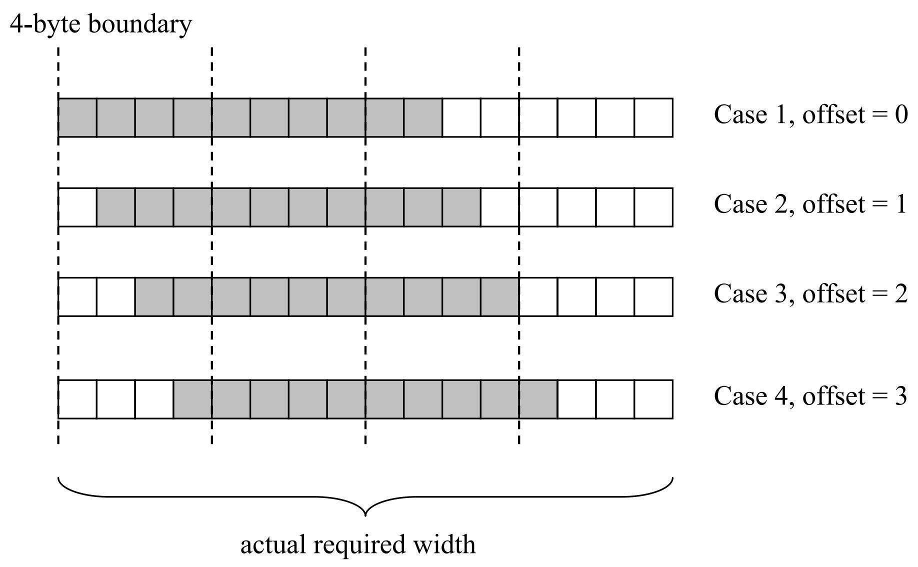


Figure 30: Four cases in loading a reference block

Pixel interpolation[5], which is a computationally intensive process, is achieved efficiently with intrinsics such as **_avg4**, **_avgu2**, and **_shrmb**, and shown in Figures 43, 44, and 45 in Appendix A, respectively. Figure 31 shows an interpolated image. $a_{y,x}$ is an integer pixel and the interpolated half-pixels are defined as:

- $b_{y,x} = (a_{y,x} + a_{y,x+1} + 1)/2$
- $c_{y,x} = (a_{y+1,x} + a_{y,x} + 1)/2$
- $d_{y,x} = (a_{y,x} + a_{y,x+1} + a_{y+1,x} + a_{y+1,x+1} + 2)/4$

The overall procedure for calculating $b_{y,x}$ is shown in Figure 32. The first step is to load 8 pixels of j^{th} row from L2 SRAM to register by using **_memd8** that loads or stores unaligned 8-byte data to register. In the second step, we separate the 8-byte pixels into upper and lower 4-byte pixels, $A4H(j)$ and $A4L(j)$, using **_hi** and **_lo**, and then generate another 4-byte data using **_shrmb**. **_shrmb** shifts the second register right by one byte and then the least significant byte of the first register is merged into the most significant byte position. At the last step, the average values can be obtained by using **_avgu4**. Similarly, we

[5]Here, we consider the pixel interpolation in the MPEG-2 standard.

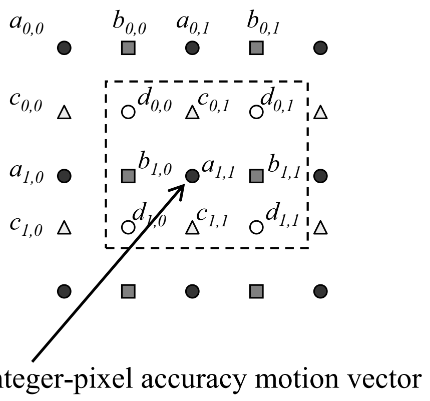
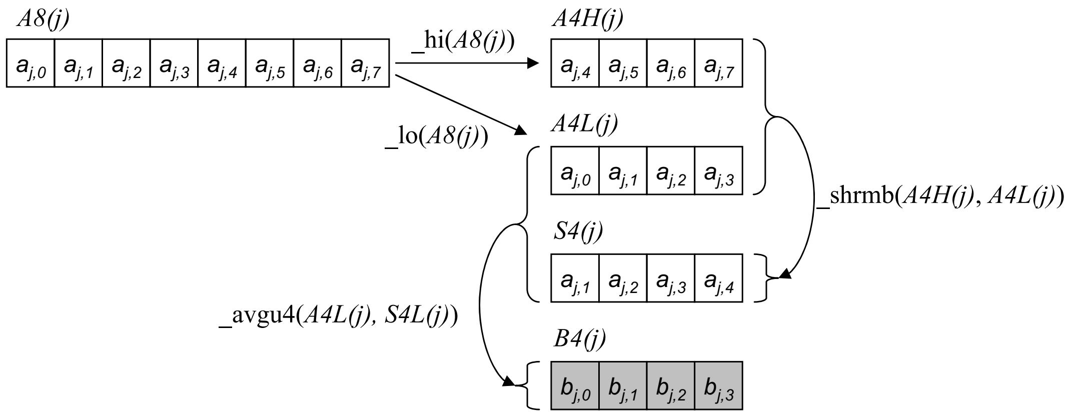


Figure 31: Half-pixel accuracy interpolated reference image



A8(j)
$a_{j,0}$ $a_{j,1}$ $a_{j,2}$ $a_{j,3}$ $a_{j,4}$ $a_{j,5}$ $a_{j,6}$ $a_{j,7}$
_hi(A8(j))
_lo(A8(j))
A4H(j)
$a_{j,4}$ $a_{j,5}$ $a_{j,6}$ $a_{j,7}$
A4L(j)
$a_{j,0}$ $a_{j,1}$ $a_{j,2}$ $a_{j,3}$
_shrmb(A4H(j), A4L(j))
S4(j)
$a_{j,1}$ $a_{j,2}$ $a_{j,3}$ $a_{j,4}$
_avgu4(A4L(j), S4L(j))
B4(j)
$b_{j,0}$ $b_{j,1}$ $b_{j,2}$ $b_{j,3}$

Figure 32: Calculation of $b_{y,x}$

can calculate $c_{y,x}$ using **_memd8**, **_hi**, **_lo** and **_avgu4**. The procedure for the calculation of $c_{y,x}$ shown in Figure 33. We first load 8 pixels of j^{th} row and 8 pixels of $(j+1)^{th}$ row, and then divide them and calculate average values.

The procedure for calculating $d_{y,x}$ is shown in Figure 34. Since we accumulate four pixel values, 2-byte memory space is required to store the accumulated results. The conversion from four 1-byte values to four 2-byte values can be done by using **_mpyu4**($A4L(j)$, 0x01010101). And then we can accumulate two 2-byte values using **_add2** 8 times after dividing 8-byte data into 4-byte data. The accumulated values can be divided using bit-shift operation, and **_packl4** converts from four 2-byte values to four 1-byte values.

As depicted in Figure 31, there are nine search positions, $a_{1,1}$, $b_{1,0}$, $b_{1,1}$, $c_{0,1}$, $c_{1,1}$,

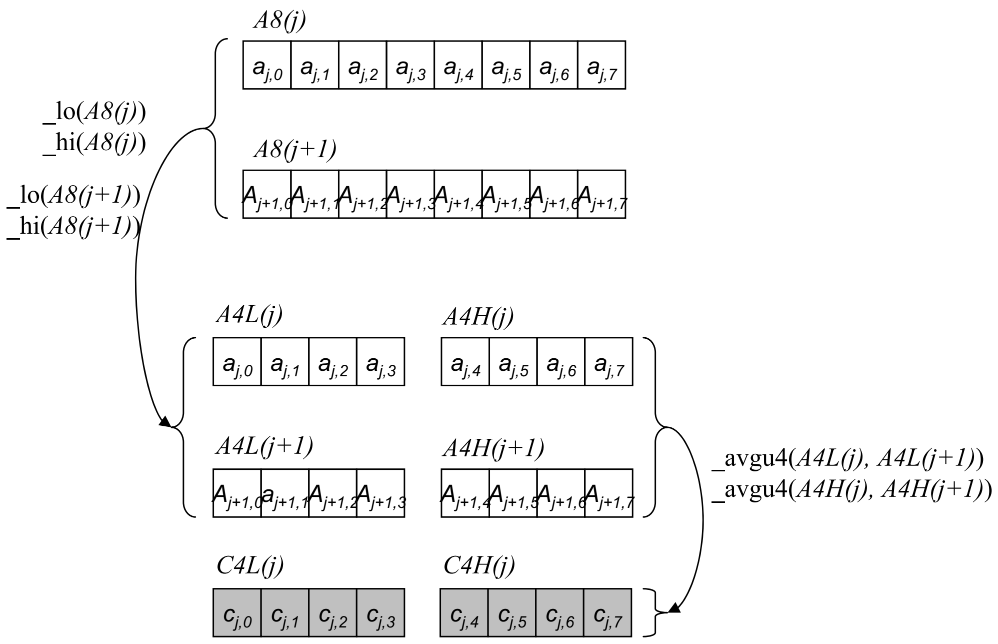

A8(j)
$a_{j,0}$ $a_{j,1}$ $a_{j,2}$ $a_{j,3}$ $a_{j,4}$ $a_{j,5}$ $a_{j,6}$ $a_{j,7}$
_lo(A8(j))
_hi(A8(j))
_lo(A8(j+1))
_hi(A8(j+1))
A8(j+1)
$A_{j+1,0}$ $A_{j+1,1}$ $A_{j+1,2}$ $A_{j+1,3}$ $A_{j+1,4}$ $A_{j+1,5}$ $A_{j+1,6}$ $A_{j+1,7}$
A4L(j)
A4H(j)
A4L(j+1)
A4H(j+1)
_avgu4(A4L(j), A4L(j+1))
_avgu4(A4H(j), A4H(j+1))
C4L(j)
C4H(j)
$c_{j,0}$ $c_{j,1}$ $c_{j,2}$ $c_{j,3}$ $c_{j,4}$ $c_{j,5}$ $c_{j,6}$ $c_{j,7}$

Figure 33: Calculation of $c_{y,x}$

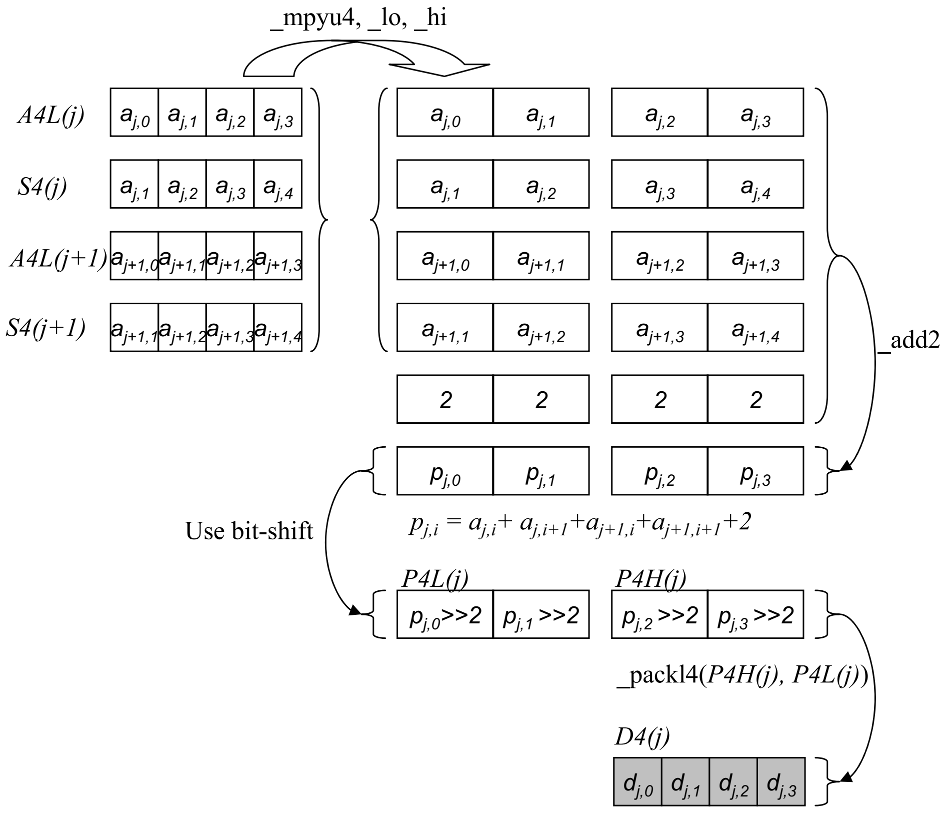

_mpyu4, _lo, _hi
A4L(j)
S4(j)
A4L(j+1)
S4(j+1)
_add2
2 2 2 2
$p_{j,0}$ $p_{j,1}$ $p_{j,2}$ $p_{j,3}$
Use bit-shift
$p_{j,i} = a_{j,i} + a_{j,i+1} + a_{j+1,i} + a_{j+1,i+1} + 2$
P4L(j)
P4H(j)
$p_{j,0}>>2$ $p_{j,1}>>2$ $p_{j,2}>>2$ $p_{j,3}>>2$
_packl4(P4H(j), P4L(j))
D4(j)
$d_{j,0}$ $d_{j,1}$ $d_{j,2}$ $d_{j,3}$

Figure 34: Calculation of $d_{y,x}$

$d_{0,0}$, $d_{0,1}$, $d_{1,0}$, and $d_{1,1}$, to be examined. When the interpolated pixels are stored into separated memory locations, the nine search points are shown in Figure 35. By storing the interpolated pixels separately, SAD calculation can be efficiently performed by utilizing fact that there are adjacent multiple search points in b-image, c-image, and d-image. For example, two SADs associated with the positions, $b_{1,0}$ and $b_{1,1}$, can be calculated with single data load to CPU registers. Let $SAD0$ and $SAD1$ be two SAD values for $b_{1,0}$ and $b_{1,1}$, respectively. Then, for an 8x8 block, we have $SAD0 = \sum_{i=0}^{7}\sum_{j=0}^{7} |b_{j+1,i} - o_{j,i}|$ and $SAD1 = \sum_{i=0}^{7}\sum_{j=0}^{7} |b_{j+1,i+1} - o_{j,i}|$, where $o_{j,i}$ is a pixel in the original image block.

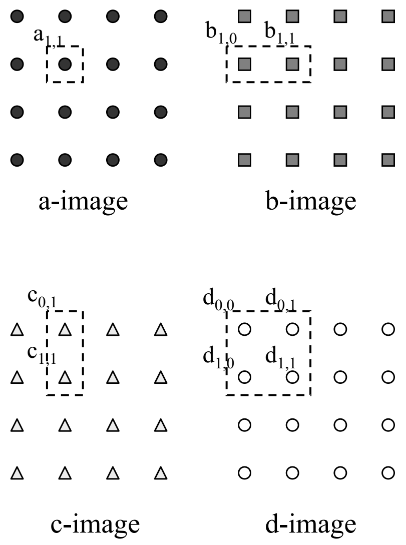
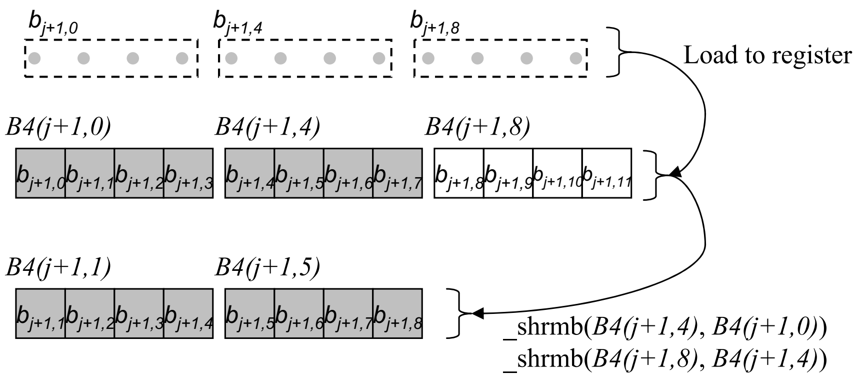


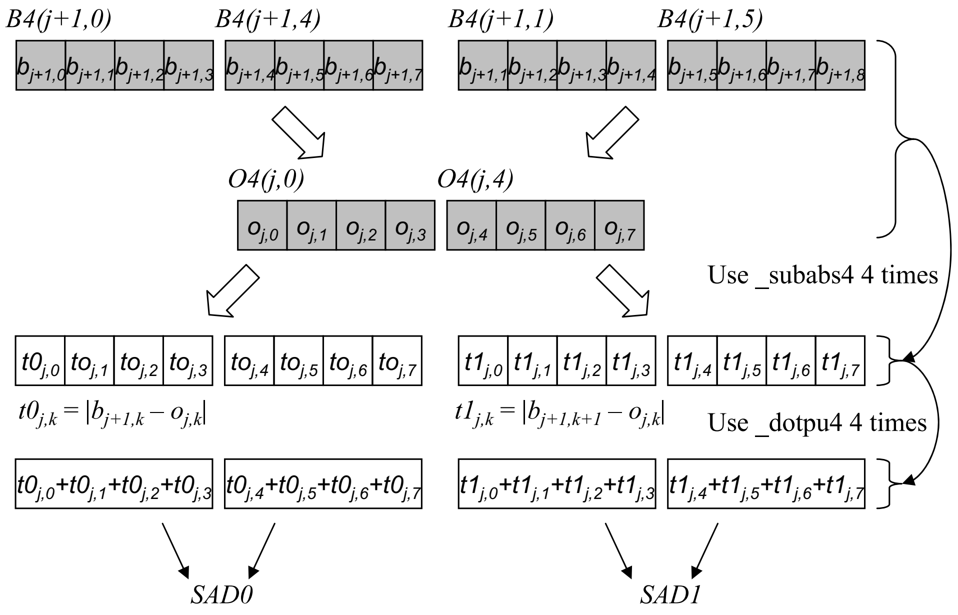
Figure 35: Separate interpolated b-, c-, and d-images

By using **_shrmb**, **_subabs4**, and **_dotpu4**, $SAD0$ and $SAD1$ can be calculated very efficiently. Figure 36 shows calculation of $SAD0$ and $SAD1$ with the intrinsics. First, we load three 4-byte data in the $(j+1)^{th}$ row in the b-image, $B4(j+1,0)$, $B4(j+1,4)$ and $B4(j+1,8)$, into the registers and generate two 1-byte shifted data, $B4(j+1,1)$ and $B4(j+1,5)$, using **_shrmb**. And then we can calculate SADs for 8 pixels using **_subabs4** and **_dotpu4**. With this approach, the number of data loads from L2 SRAM to the registers can be minimized, i.e. 3x8 4-byte data loads for the b-image and 2x8 4-byte data loads for the original image. Similarly, efficient implementation of SAD calculations for the c-image and d-image can be done.



$b_{j+1,0}$
$b_{j+1,4}$
$b_{j+1,8}$
Load to register
$B4(j+1,0)$
$B4(j+1,4)$
$B4(j+1,8)$
$b_{j+1,0}$ $b_{j+1,1}$ $b_{j+1,2}$ $b_{j+1,3}$
$b_{j+1,4}$ $b_{j+1,5}$ $b_{j+1,6}$ $b_{j+1,7}$
$b_{j+1,8}$ $b_{j+1,9}$ $b_{j+1,10}$ $b_{j+1,11}$
$B4(j+1,1)$
$B4(j+1,5)$
$b_{j+1,1}$ $b_{j+1,2}$ $b_{j+1,3}$ $b_{j+1,4}$
$b_{j+1,5}$ $b_{j+1,6}$ $b_{j+1,7}$ $b_{j+1,8}$
_shrmb($B4(j+1,4)$, $B4(j+1,0)$)
_shrmb($B4(j+1,8)$, $B4(j+1,4)$)

(a) Prepare four 4-byte data



$B4(j+1,0)$
$B4(j+1,4)$
$B4(j+1,1)$
$B4(j+1,5)$
$b_{j+1,0}$ $b_{j+1,1}$ $b_{j+1,2}$ $b_{j+1,3}$
$b_{j+1,4}$ $b_{j+1,5}$ $b_{j+1,6}$ $b_{j+1,7}$
$b_{j+1,1}$ $b_{j+1,2}$ $b_{j+1,3}$ $b_{j+1,4}$
$b_{j+1,5}$ $b_{j+1,6}$ $b_{j+1,7}$ $b_{j+1,8}$
$O4(j,0)$
$O4(j,4)$
$o_{j,0}$ $o_{j,1}$ $o_{j,2}$ $o_{j,3}$
$o_{j,4}$ $o_{j,5}$ $o_{j,6}$ $o_{j,7}$
Use _subabs4 4 times
$t0_{j,0}$ $to_{j,1}$ $to_{j,2}$ $to_{j,3}$
$to_{j,4}$ $to_{j,5}$ $to_{j,6}$ $to_{j,7}$
$t1_{j,0}$ $t1_{j,1}$ $t1_{j,2}$ $t1_{j,3}$
$t1_{j,4}$ $t1_{j,5}$ $t1_{j,6}$ $t1_{j,7}$
$t0_{j,k} = |b_{j+1,k} - o_{j,k}|$
$t1_{j,k} = |b_{j+1,k+1} - o_{j,k}|$
Use _dotpu4 4 times
$t0_{j,0}+t0_{j,1}+t0_{j,2}+t0_{j,3}$
$t0_{j,4}+t0_{j,5}+t0_{j,6}+t0_{j,7}$
$t1_{j,0}+t1_{j,1}+t1_{j,2}+t1_{j,3}$
$t1_{j,4}+t1_{j,5}+t1_{j,6}+t1_{j,7}$
$SAD0$
$SAD1$

(b) Calculate SADs

Figure 36: SAD calculation for the b-image

4.2.4 Others Modules

Many DSP manufactures provide highly optimized assembly source code or library functions for widely used but computationally intensive image and video processing algorithms such as convolution, 8x8 DCT, quantization etc. For example, TI provides a set of library functions that can be directly used for real-time applications where optimal execution speed is critical [4].

DCT can be implemented with **IMG_fdct_8x8** and **IMG_idct_8x8** that perform forward and inverse DCT, respectively. **IMG_idct_8x8** can perform rounding and saturation to signed 9-bit quantities as well as inverse DCT assuming the input coefficients are signed 12-bit values. In general, these functions can perform DCT for multiple 8x8 blocks per call, and usually performing DCT for multiple 8x8 blocks takes less number of cycles compared to performing DCT for a single 8x8 block.

Quantization can be done efficiently by using **IMG_quantize**. The function quantizes a list of blocks by multiplying their contents with a second block of values that contains reciprocals of the quantization items. Because the function does not consider dead-zone, we may need to modify the quantized coefficients to avoid performance loss. The function also can perform inverse quantization by using appropriate reciprocals. The reciprocals for forward and inverse quantization should be calculated at encoder initialization stage and loaded into internal memory before forward or inverse quantization processes. To calculate the reciprocals, we need to understand how **IMG_quantize** function works. For the k^{th} coefficient in an 8x8 coefficient matrix, the C version of the function is as follows:

```
round = $q\_pt$ ? $1 << (q\_pt - 1) : 0$;
quot = $coef_k * recip_k$ + round
level$_k$ = quot $>> q\_pt$
```

where $coef_k$ is the coefficient to be quantized, $recip_k$ is a reciprocal associated with the position of the coefficient, and q_pt is a accuracy control factor for the reciprocal that has a range of 0 to 31. For a given quantization parameter ($quant\_param$) and an appropriate

q_pt, the reciprocal for the k^{th} position ($recip_k$) in the forward quantization is calculated by $recip_k = (1 << q\_pt)$ / (quant_mat$_k$ * $quant\_param$), where quant_mat$_k$ is the k^{th} value in the base quantization matrix. For the inverse quantization, the reciprocal for k^{th} position is $recip_k$ = quant_mat$_k$ * $quant\_param$ with $q\_pt = 0$.

4.3 Overall structure

Designing a video encoder is affected by numerous hardware factors , e.g., speed of a processor, L1 cache size, L2 memory size, EDMA performance, etc., and application's requirements such as input frame size and frame rate. Here, we present a structure for a real-time full D1 MPEG-2 video encoder that can be implemented on a TI's TMS320DM642 system.

Figure 37 shows overall encoder structure that contains data transfer information and process flow[6]. MB_W and MB_H are the numbers of macroblocks in a row and column, respectively. MB_S is the total number of macroblocks in a frame, i.e. $MB\_W \times MB\_H$. For example, for NTSC full D1 video, $MB\_W = 45$, $MB\_H = 30$, and $MB\_S = 1350$. N is a number for a group of macroblocks and the number in the parenthesis for each module means the number of macroblocks that are processed in the module. For example, the level-3 integer motion estimation module finds motion vectors for all the macroblocks in a frame, and each module in the Loop 2 processes N macroblocks per call.

The first module, picture type decision and bit allocation, determines the picture type (I, P or B) and allocates a number of bits for the current frame. If the current picture type is I, it performs bit allocation for a Group Of Pictures (GOP). This module also manages reference frame buffer.

The sub-sampling module generates reduced size frames that are stored in the internal memory for hierarchical motion estimation. The level-3 motion estimation produces motion vectors for all the macroblocks and the coarse motion vectors are refined in the level-2 motion estimation module. For the level-3 and level-2 motion estimation, reference areas for a macroblock line are loaded into L2 SRAM with ping-pong buffer structure as described in 4.2.1. After finding all motion vectors, the current reduced frames are stored into the

[6]For I-type frames, integer- and sub-pixel accuracy motion estimation and mode selection are not required.

reference frame buffer by using DMA transfer.

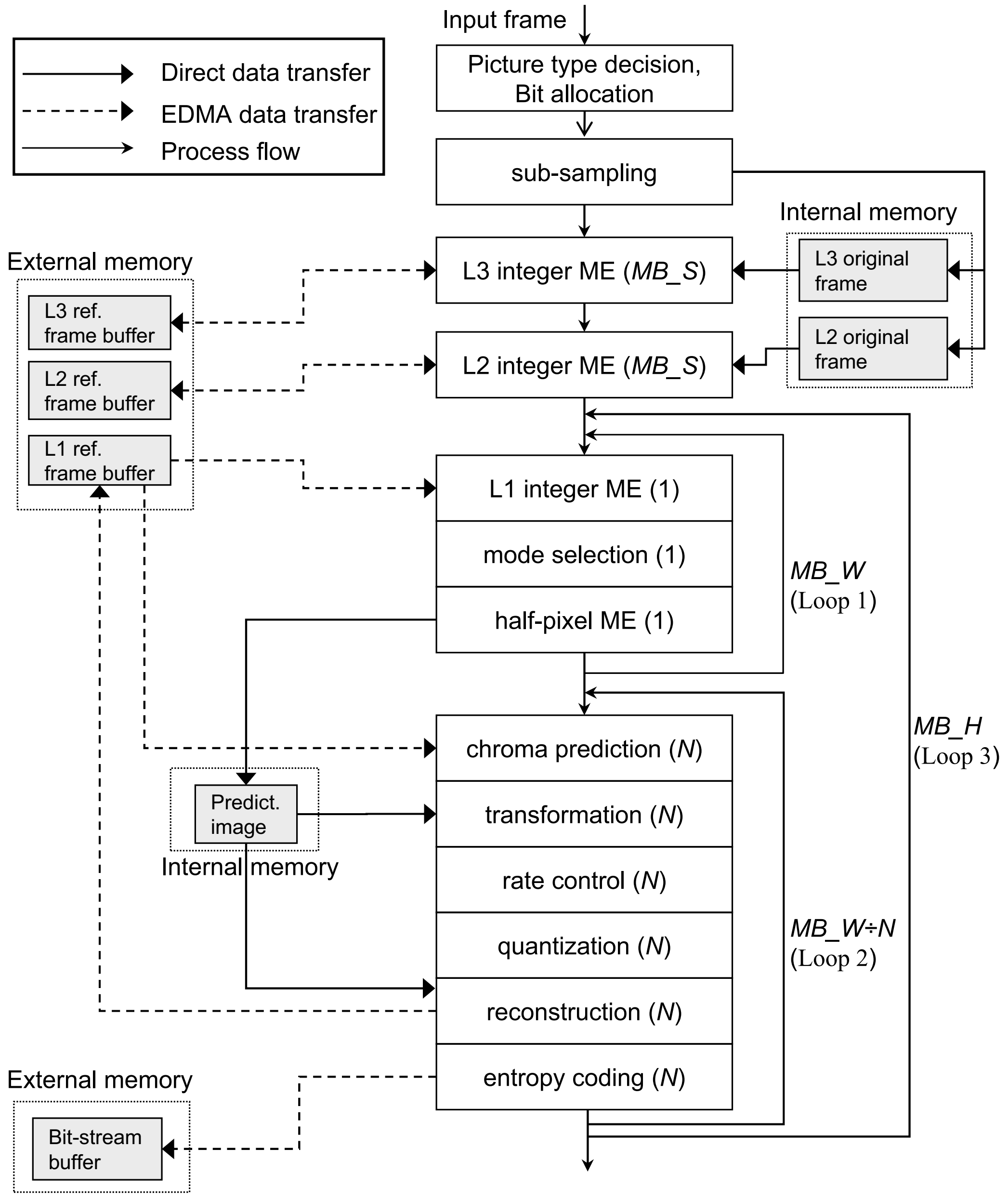


Figure 37: MPEG-2 encoder structure

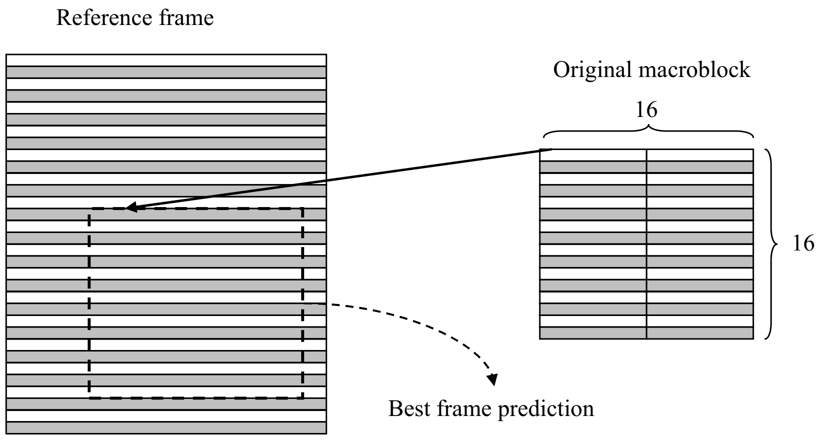
In the level-2 motion estimation module, the best cost for each macroblock and/or the accumulated cost are stored to detect scene change or motion change in the rate control module. The rate control module determines the complexity of the current frame by using the cost information. For example, if the accumulated cost for the current frame is much larger than that for the previous frame, the rate control module tries to minimize perceptual quality fluctuation by changing the frame type to I or performing appropriate frame bit allocation.

In the Loop 1, the level-1 motion estimation module produces frame, top field, and bottom field motion vectors for each reference direction. Before starting level-1 motion estimation for the first macroblock line, the corresponding reference area has to be loaded into L2 SRAM, which means the CPU has to wait until loading is done. However, from the second macroblock line, the reuqired reference area is loaded by EDMA while Loop 2 is processed. In this case, ping-pong buffer structure is not required to store the reference area.

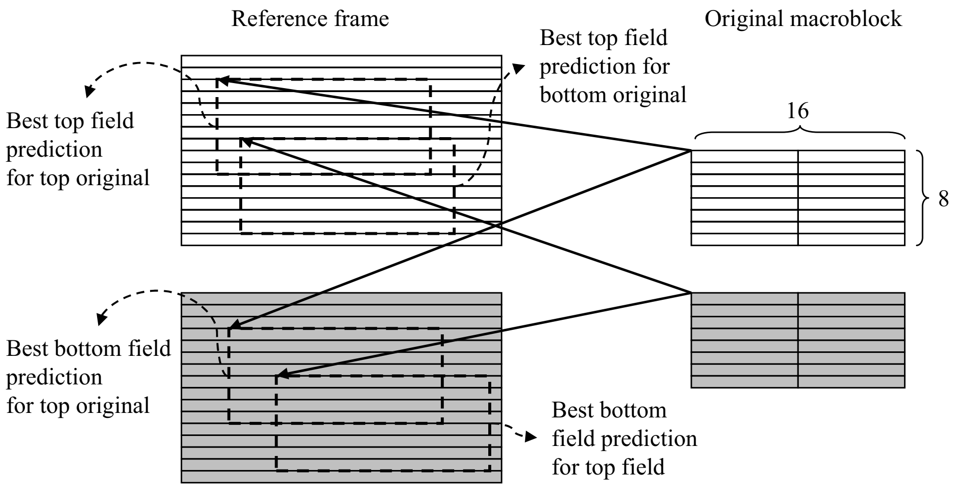
Figure 38 depicts frame and field prediction in the MPEG-2 standard. In the frame prediction, both reference area and original macroblock are considered as frame. On the other hand, in the field prediction, the original macroblock is decomposed into top and bottom fields, and the best prediction is obtained for each original field. The best prediction for the original top field can be found in either the reference top or bottom field. Similarly, the best prediction for the original bottom field can be obtained in the top or bottom field.

However, in practical implementation, it is not necessary to decompose a frame into two fields. Figure 39 shows how to calculate one SAD cost for frame prediction and four costs for field prediction for a given search point. $OT(j)$ and $OB(j)$ are the j^{th} rows of the original top field and bottom field, respectively. Similarly, $RT(j)$ and $RB(j)$ are the j^{th} rows of the reference top field and bottom field, respectively. Calculating four SADs for field predictions and one SAD for frame prediction is as follows:

- SAD between original top and reference top (SAD_TT) $= \sum_{j=0}^{7} \text{sad\_calc}(OT(j), RT(j))$
- SAD between original top and reference bottom (SAD_TB) $= \sum_{j=0}^{7} \text{sad\_calc}(OT(j), RB(j))$
- SAD between original bottom and reference top (SAD_BT) $= \sum_{j=0}^{7} \text{sad\_calc}(OB(j), RT(j))$
- SAD between original bottom and reference bottom (SAD_BB) $= \sum_{j=0}^{7} \text{sad\_calc}(OB(j), RB(j))$
- SAD for frame prediction $= SAD\_TT + SAD\_BB$



(a) Frame prediction


Reference frame
Original macroblock
Best top field prediction for bottom original
16
8
Best top field prediction for top original
Best bottom field prediction for top original
Best bottom field prediction for top field

(b) Field prediction

Figure 38: Frame and field prediction in the MPEG-2 coding standard

where calc_sad($row1$, $row2$) is a function that calculate SAD between two 16-pixel rows, $row1$ and $row2$.

In this approach, when two rows of the reference block, one from top and the other one from bottom field, are loaded from the internal memory to registers, all the five SADs for a given search points can be calculated, which minimizes redundant SAD calculations and data loading time.

After the level-1 motion estimation module generates motion vectors for a macroblock, the mode selection modules calculates costs for all candidate modes and decides the best

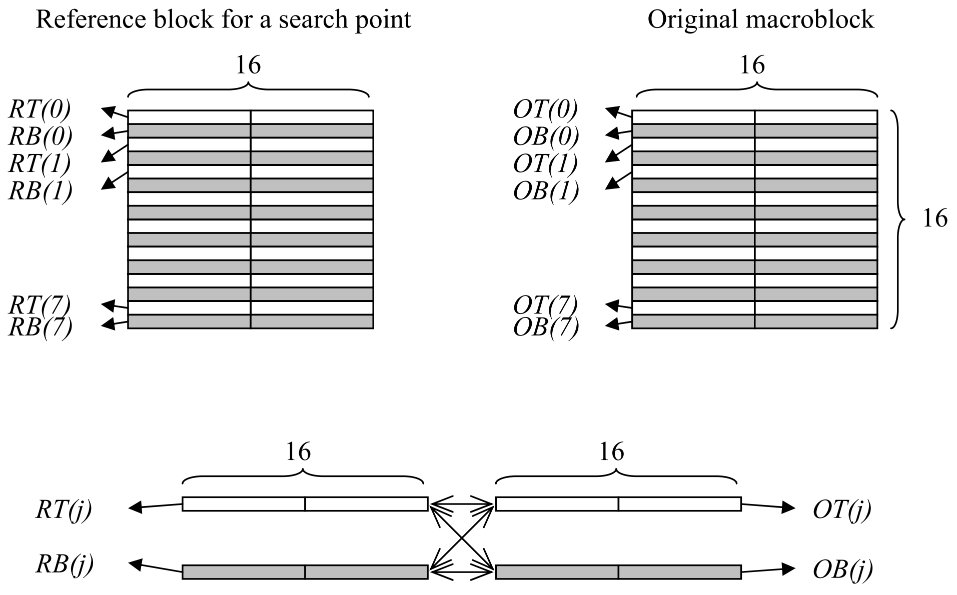


Figure 39: Calculation of frame SAD and four field SADs

mode. Decisions for prediction type (frame or field), prediction direction (forward, backward, or bi-direction) are also included in the mode selection module. For the variance calculation, there is no data transfer between internal and external memory because the reference area is already in the internal memory.

For the best mode, the sub-pixel accuracy motion estimation module performs upsampling and determines a search position that generates minimum costs as described in 4.2.3. Depending on the prediction type and prediction direction, the number of calls for upsampling and cost calculation functions is different. Table 8 shows the number for each case. The best luma prediction block[7] determined by the sub-pixel accuracy motion estimation module is directly copied into the internal memeory, and used to obtain residual and reconstruct images in the Loop 2.

The chroma prediction module in the Loop 2 generates prediction and residual blocks for the chroma components. If a motion vector points to a half pixel location, the module performs upsampling first, and then calculate residual blocks. Reference area loading with ping-pong buffer structure can be done by EDMA since the final motion vectors are already

[7]Integer-pixel motion estimation, mode selection, and sub-pixel motion estimation use only luma component.

Table 8: Number of calls for upsampling and cost calculation functions

	Forward	Backward	Bi-direction
Frame prediction	1	1	2
Field prediction	2	2	4

available.

Before performing forward transform, we need to decide DCT type and re-arrange the residual image depending on the DCT type. The frame and field correlation coefficients of the residual image are used in TM5 for the DCT type decision. Implementation of the correlation coefficient calculation is similar to that of the variance calculation in the mode selection since it involves only simple calculations such as accumulation and SSD.

Before quantization, the rate control module chooses a quantization parameter for N macroblocks. Depending on a rate control scheme, the rate control module may be included in the quantization module and determine a quantization parameter for each macroblock.

In the reconstruction module, inverse quantization, inverse DCT, and some miscellaneous processes such as addition and clipping are performed. In the MPEG-2 standard, reconstruction for B-type frames is not required for most of applications because the reconstructed frame is not used as a reference frame[8]. After the reconstruction process, the reconstructed images are stored into the reference frame buffer located in the external memory using EDMA.

The entropy coding module produces a bit-stream that is written into an external bit-stream buffer and returns information on the number of bits used to code DCT coefficients, motion vectors, and other components to the rate control module. In general, the tables for variable length coding always reside in L2 SRAM to avoid accessing to the external memory. The complexity of this module varies depending on bit-rate in constant bit-rate environment. Usually, high bit-rate coding takes more number of cycles because it produces more nonzero coefficients. However, the bit generation in variable bit-rate environment varies depending on the contents of input video.

[8]In the H.264 standard, reconstruction process may be required to perform accurate spatial intra-prediction.

CHAPTER V

CONCLUSION AND FUTURE WORK

5.1 Contributions

In this thesis, we propose a rate-distortion model and apply it successfully to mode selection problem. The proposed mode selection method has very low computational complexity since it requires simple statistics of the residual image that can be implemented easily on most of modern DSP systems. We are also able to improve the encoding performance further by extending the scheme to motion vector selection when multiple motion vectors are available in a mode. Our experimental results show that the proposed method can give significant PSNR and visual quality improvement over TM5 for MPEG-2.

In addition to our theoretical work, we discuss key implementation issues on each encoder module including the proposed mode selection method. First, we provide a brief background on major features of modern DSPs. Second, we present practical solutions to implementation of encoder modules such as integer-pixel, sub-pixel accuracy motion estimation, and our proposed mode selection method. Finally, we provide an overall structure for a real-time implementation of MPEG-2 encoder on a TI's DSP system, where we discuss overall data and process flows.

5.2 Suggestions for Further Research

Here, we discuss the coding mode selection in H.264 as a future work, and present several suggestions for computational complexity reduction. Basically, cost calculation process for the mode selection in H.264 is almost the same as in the previous standards. However, the total computational complexity for cost calculation increases in proportion to the number of candidate modes. In the literature, most of the algorithms focus on reducing the number of candidate block modes using the characteristics of the current macroblock, e.g., homogeneity, temporal stationarity, number of zero coefficients etc., to skip cost calculation for less probable candidate modes [17, 37, 61, 64, 65]. After removing some less probable modes, they apply Lagrangian optimization with actual information on rate and distortion only to the remaining modes. Clearly, with these approaches, total computational complexity can be reduced over the reference software. However, the approaches are not guaranteed to work in real-time environment because they use Lagrangian optimization method with actual information that is obtained by performing transform, quantization, reconstruction, and entropy coding. Therefore, we suggest several directions for real-time implementation of mode selection for future work.

- Using the proposed model-based mode selection method: This approach will reduce computational complexity significantly over the reference software and could be used in real-time environment without any further simplification. However, we need to determine whether performance degradation over the reference software is acceptable or not.

- Using Lagrangian technique with estimate rate and distortion: The direction of this approach is different than that of our proposed method. As described in [60], we could employ Lagrangian optimization with estimated rate and distortion. With this approach, we need to estimate distortion and rate without performing actual encoding processes. For the distortion estimation, distortion could be calculated in the transform domain. Because 4x4 integer transform and quantization is much simpler than 8x8-based DCT and quantization, calculating distortion in the transform domain would be much simpler in H.264 than in the previous standards. For rate estimation, exploiting the fact that the number of zero quantized transform coefficients can be easily obtained in the process of estimating distortion, we could use the information on the number of zeros to estimate the number of bits required for sending quantized transform coefficients. For effective and computationally simple rate estimation, Kim *et al.* employ the rate model in Eq. 4 explained in Section 3.1 [36]. The actual number of bits and the estimated number of bits for macroblocks are shown in Figure 40.

- Using our model-based mode selection method to find probable modes and applying Lagrangian technique with estimate rate and distortion only to the remaining modes:

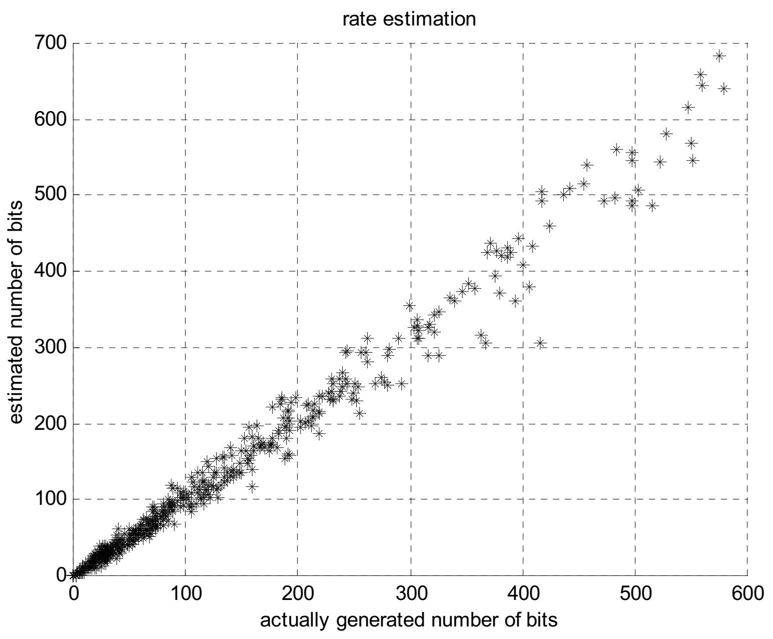


Figure 40: Estimated rates vs. actual rates

Even if the second approach could give much better performance than the first one, the second approach has higher computational complexity than the first one. Thus, we could find a good trade-off between performance and complexity by combining two methods. First, we could use our proposed mode selection method and remove some modes that have large costs, and then apply Lagrangian method with estimated rate and distortion only to the remaining modes.

APPENDIX A

INTRINSICS FOR TMS320DM64X/TMS320C64X

In algorithm implementation on DSP systems, the first optimization step that we can perform on C source code is to use intrinsic operators that correspond to the highly optimized assembly language instruction(s). Although TI provides various intrinsics for TMS320DM64x/TMS320C64x [5], here, we describe the intrinsics that are used in our algorithm implementation.

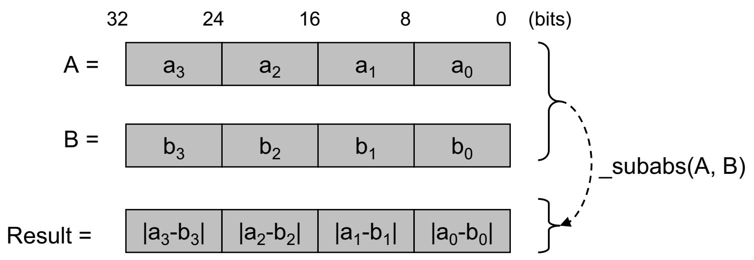


Figure 41: _subabs4 takes single cycle

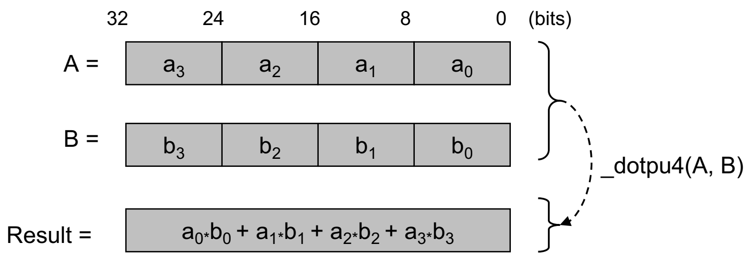


Figure 42: _dotpu4 takes four cycles

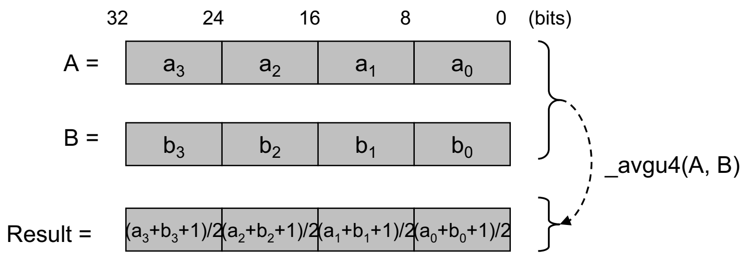


Figure 43: _avgu4 takes two cycles

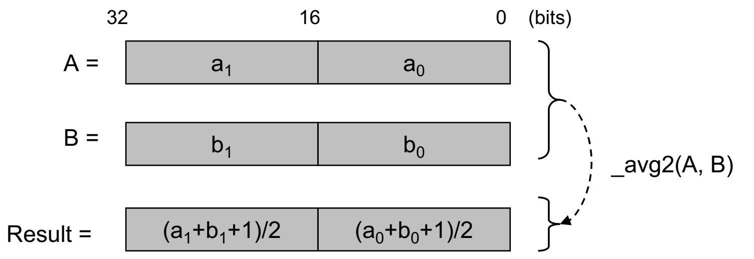


Figure 44: _avg2 takes two cycles

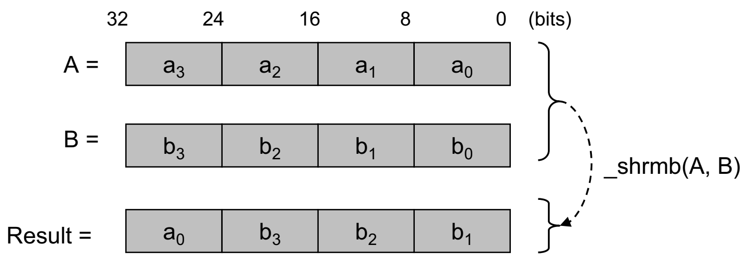


Figure 45: _shrmb takes single cycle

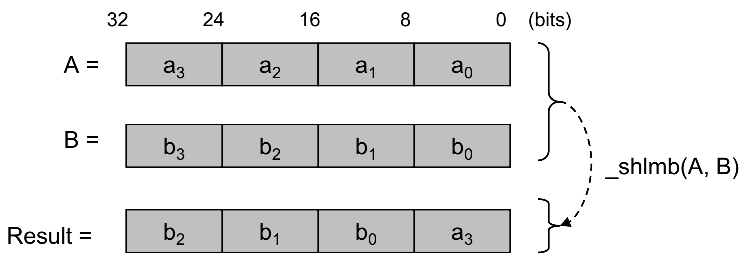


Figure 46: _shlmb takes single cycle

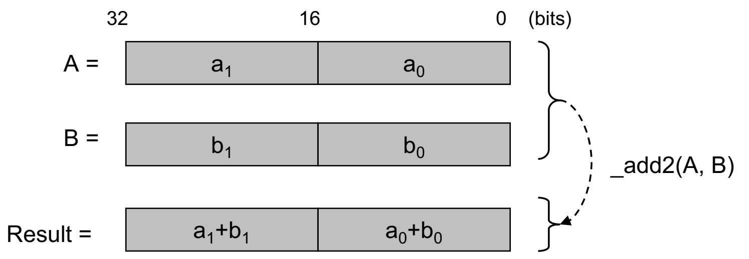


Figure 47: _add2 takes single cycle

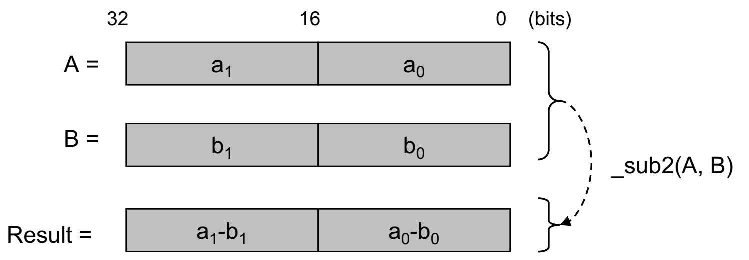


Figure 48: _sub2 takes single cycle

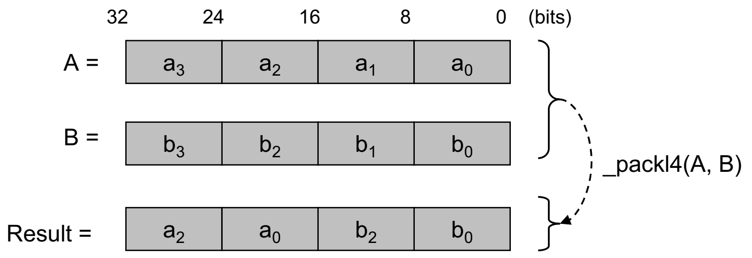


Figure 49: _packl4 takes single cycle

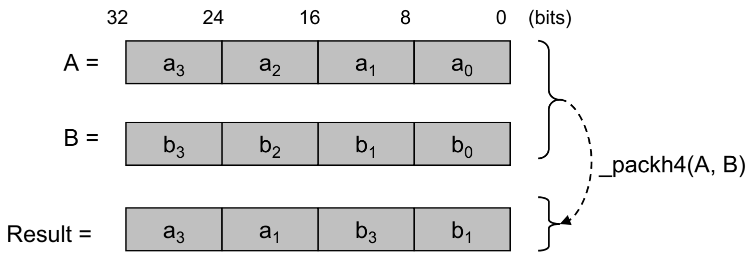


Figure 50: _packh4 takes single cycle

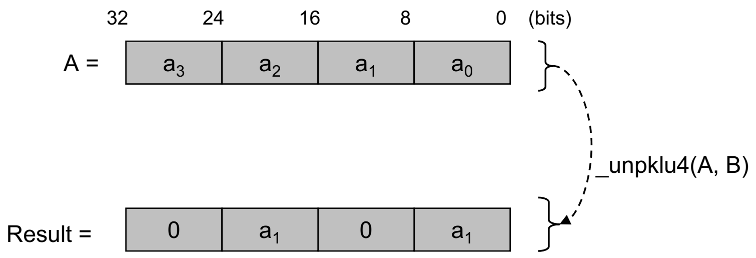


Figure 51: _unpklu4 takes single cycle

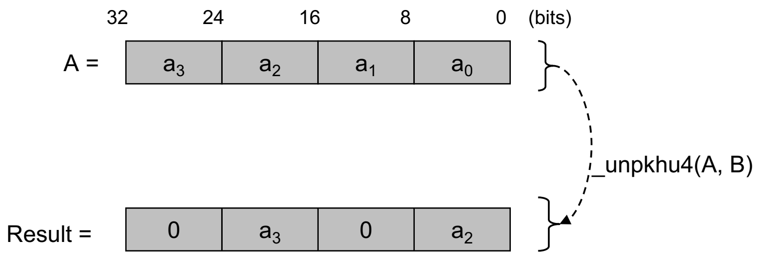


Figure 52: _unpkhu4 takes single cycle

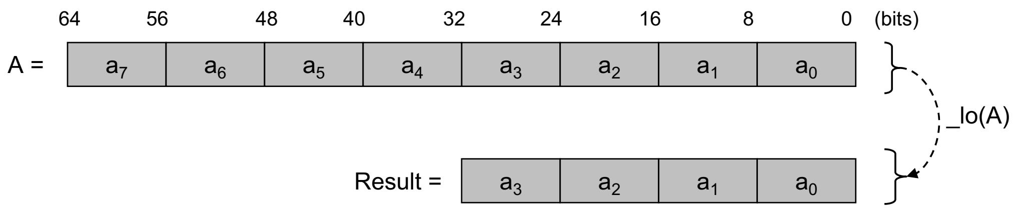


Figure 53: _lo takes single cycle

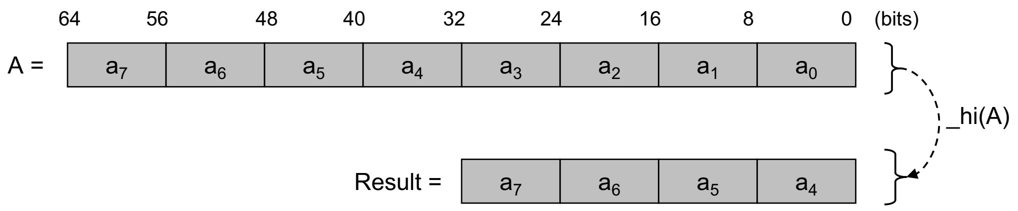


Figure 54: _hi takes single cycle

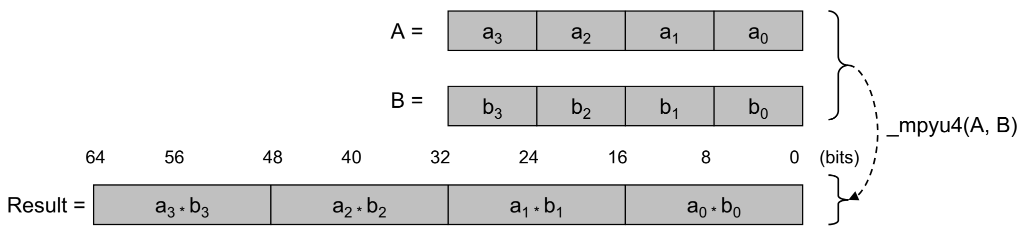


Figure 55: _mpyu4 takes four cycles

REFERENCES

[1] "TMS320C6000 CPU and instruction set reference guide," *Application Notes, Texas Instruments.*

[2] "TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) controller reference guide (rev. c)," *Application Notes, Texas Instruments.*

[3] "TMS320C64x DSP two-level internal memory reference guide," *Application Notes, Texas Instruments.*

[4] "TMS320C64x image/video processing library programmer's reference," *Application Notes, Texas Instruments.*

[5] "TMS320C64x/C64x+ DSP CPU and instruction set reference guide (rev. c)," *Application Notes, Texas Instruments.*

[6] "Video encoding optimization on TMS320DM64x/C64x," *Application Notes, Texas Instruments.*

[7] "Generalized lagrange multiplier for method for solving problems of optimum allocation of resources," *Operations Research*, vol. 36, pp. 1445–1453, 1988.

[8] "Coded representation of picture and audio information-MPEG-2 test model 5," *ISO-IEC AVC-491*, 1993.

[9] "Video coding for audiovisual services at $p \times 64$ kbits," *ITU-T, ITU-T Recommendation H.261*, Mar. 1993.

[10] "Video codec test model, near-term, version 5 (TMN5)," *ITU-T Study Group 15, Video Expert Group*, Jan. 1995.

[11] "Video coding for low bit rate communications," *ITU-T, ITU-T Recommendation H.263*, 1995.

[12] "ISO/IEC, reference number ISO/IEC 13818-2," *The MPEG-2 international standard*, 1996.

[13] "Video codec test model, near-term, version 8 (TMN8)," *ITU-T Study Group 16, Video Expert Group*, June 1997.

[14] "Motion pictures expert group - overview of the MPEG-4 standard," *ISO/IEC JTC1/SC29/WG11 N2459*, 1998.

[15] "Draft ITU-T recommendation and FDIS of joint video spec. (H.264—ISO/IEC 14496-10 AVC)," *JVT of MPEG and VCEG, Doc. JVT-G050R1*, May 2003.

[16] "http://bmrc.berkeley.edu/frame/research/mpeg/," *MPEG codec*, April 2007.

[17] Ahmad, A., Khan, N., Masud, S., and Maud, M., "Selection of variable block sizes in h.264," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 3, pp. 173–176, May 2004.

[18] Chen, M. and Willson, A., "Rate-distortion optimal motion estimation algorithm for video coding," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, pp. 2096–2099, 1996.

[19] Chen, Y.-K. and Kung, S. Y., "Rate optimization by true motion estimation," *Proc. IEEE Workshop on Multimedia Signal Processing*, Jun. 1997.

[20] Choi, K. T., Chan, S. C., and Ng, T. S., "A new fast motion estimation algorithm using hexagonal subsampling pattern and multiple candidates search," *ICIP'96*, vol. 2, pp. 497–500, 1996.

[21] Chung, W. C., Kossentini, F., and Smith, M. J. T., "An efficient motion estimation technique based on a rate-distortion criterion," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 43, pp. 2771–2783, Nov. 1995.

[22] Cover, T. M. and Thomas, J. A., "Elements of information theory," *Wiley, New York, NY*, 1991.

[23] Dekalp, A. M., *Digital Video Processing.* Prentice Hall, 1995.

[24] Du, C., He, Y., and Zheng, J., "Pphps: A parabolic predictionbased, fast half-pixel search algorithm for very low bit-rate movingpicture coding," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 6, no. 1, pp. 514–518, 2003.

[25] Gallant, M., Cote, G., and Kossentini, F., "An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding," *IEEE Trans. Image Proc.*, vol. 8, Dec 1999.

[26] Ghanbari, M., "The cross-search algorithm for motion estimation," *IEEE Trans. Comm.*, vol. 38, July 1990.

[27] Girod, B., "Motion-compensating prediction with fractional-peel accuracy," *IEEE Trans. Comm.*, vol. 41, no. 4, pp. 604–612, 1993.

[28] He, Z., Kim, Y., and Mitra, S. K., "A novel linear source model and a unified rate control algorithm for H.263/MPEG-2/MPEG-4," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Oct 1998.

[29] He, Z., Kim, Y. K., and Mitra, S. K., "Low delay rate control for dct video coding via ρ-domain source modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, Aug. 2001.

[30] He, Z. and Mitra, S. K., "A unified rate-distortion analysis framework for transform coding," *IEEE Trans. on Circuit Syst. for Video Technol.*, vol. 11, pp. 1221–1236, Dec. 2001.

[31] Hu, S. Y., Chen, M. C., and Willson, A. N., "A fast rate-distortion optimization algorithm for motion compensated video coding," *IEEE Int. Symp. on Circuits and Systems*, pp. 1349–1352, June 1997.

[32] Jain, J. R. and Jain, A. K., "Displacement measurement and its application in interframe image coding," *IEEE Trans. Comm.*, vol. 29, Dec 1981.

[33] Jeong, J., "Fast sub-pixel motion estimation having lower complexity," *ICCE. 2003 IEEE International Conference*, pp. 174–175, 2003.

[34] Jeong, J. and Ahn, W., "Subpixel-accuracy motion estimation using a model for motion compensated error," *PCS'93*, 1993.

[35] Kamaci, N., Altunbasak, Y., and Mersereau, R. M., "Frame bit allocation for the h.264/avc video coder via cauchy-density-based rate and distortion models," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 994–1006, Aug. 2005.

[36] Kim, H. and Altunbasak, Y., "Low-complexity macroblock mode selection for the H.264/AVC encoders," *IEEE Int. Conf. on Image Processing*, Oct. 2004.

[37] Kim, Y.-H., Yoo, J.-W., Lee, S.-W., Shin, J., Paik, J., and Jung, H.-K., "Adaptive mode decision for h.264 encoder," *Electronic Letters*, vol. 40, pp. 1172–1173, Sep. 2004.

[38] Koga, T. and Iinuma, K., "Motion compensated interframe coding for video conference," *Proc. NTC*, Nov 1991.

[39] Lam, E. Y. and Goodman, J. W., "A mathematical analysis of the dct coefficient distributions for images," *IEEE Trans. Image Processing*, vol. 9, pp. 1661–1666, Oct. 2000.

[40] Lee, Y. W., Kossentini, F., and Ward, R., "Efficient rd optimized macroblock coding mode selection for MPEG-2 video encoding," *Proc. Int. Conf. Image Processing*, vol. 6, pp. 168–181, Apr. 1996.

[41] LeGall, D., "MPEG: A video compression standard for multimedia application," *Commun., ACM*, vol. 34, pp. 46–58, Apr. 1991.

[42] Li, X. and Gonzales, C., "A locally quadratic model of the motion estimation error criterion function and its application to subpixel interpolations," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 6, pp. 118–122, Feb. 1996.

[43] Malvar, H. S., Hallapuro, A., Karczewicz, M., and Kerofsky, L., "Low-complexity transform and quantization in h.264/avc," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 598–603, July 2003.

[44] Marpe, D., Schwarz, H., and Wiegand, T., "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 620–636, July 2003.

[45] Mukherjee, D. and Mitra, S. K., "Combined mode selection and macroblock quantization step adaptation for the H.263 video encoder," *Proc. Int. Conf. Image Processing*, pp. 37–40, Aug. 1997.

[46] Ortega, A. and Ramchandran, K., "Rate-distortion methods for image and video compression : An overview," *IEEE Signal Processing Magazine*, pp. 23–50, Nov. 1998.

[47] Peel, C. B., Budge, S. E., Liang, K. M., and Huang, C.-M., "Locally optimal, buffer-constrained motion estimation and mode selection for video sequences," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 47, pp. 1718–1734, Jul. 2001.

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

[49] Ribas-Corbera, J. and Lei, S., "Rate control in dct video coding for low-delay communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 172–185, Feb. 1999.

[50] Richardson, I. E. G., *Video Codec Design.* John Wiley and Sons, 2002.

[51] Richardson, I. E. G., *H.264 and MPEG-4 Video Compression.* John Wiley and Sons, 2003.

[52] Senda, Y., Harasaki, H., and Yano, M., "A simplified motion estimation using an approximation for the mpeg-2 real-time encoder," *ICASSP'95*, vol. 4, pp. 2273–2276, 1995.

[53] Senda, Y., Harasaki, H., and Yano, M., "Theoretical background and improvement of a simplified half-pel motion estimation," *ICIP'96*, vol. 3, pp. 263–266, 1996.

[54] Shen, J. and Chan, W.-Y., "Fast rate-distortion optimisation algorithm for motion-compensated transform coding of video," *IEE Electronics Letters*, vol. 36, pp. 305–306, Feb. 2000.

[55] Shoham, Y. and Gersho, A., "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sep. 1988.

[56] Subramanian, P. and Chan, W.-Y., "Reduced-complexity rate-distortion optimization of multiresolution motion field and prediction residual," *Proc. Int. Conf. Image Processing*, vol. 2, pp. 799–802, Aug. 1997.

[57] Sun, H., Kwok, W., Chien, M., and Ju, C. H. J., "MPEG coding performance improvement by jointly optimizing coding mode decisions and rate control," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 449–458, Jun. 1997.

[58] Tao, B., Dickinson, B. W., and Peterson, H. A., "Adaptive model-driven bit allocation for mpeg video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 147–157, Feb. 2000.

[59] Wiegand, T. and Girod, B., "Lagrange multiplier selection in hybrid video coder control," *Proc. Int. Conf. Image Processing*, pp. 542–545, Oct. 2001.

[60] Wiegand, T., Lightstone, M., Mukherjee, D., Campbell, T. G., and Mitra, S. K., "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.

[61] Wu, D., Wu, S., Lim, K., Pan, F., Li, Z., and Lin, X., “Block inter mode decision for fast encoding of h.264,” *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 3, pp. 181–184, May 2004.

[62] Yang, K. and Jacquin, A., “Real-time implementation of rate-distortion optimized coding mode selection for h.263 video coders,” *Proc. Int. Conf. Image Processing*, 1998.

[63] Yavanof, G. S. and Liu, S., “Statistical analysis of the dct coefficients and their quantization error,” *Conf. Rec. 30th Asilomar Conf. Signals, Systems, Computers*, vol. 1, pp. 601–605, 1997.

[64] Yin, P., Tourapis, H. C., Tourapis, A. M., and Boyce, J., “Fast mode decision and motion estimation for jvt/h.264,” *Proc. Int. Conf. Image Processing*, vol. 3, pp. 853–856, Sep. 2003.

[65] Yu, A., “Efficient block-size selection algorithm for inter-frame coding in h.264/mpeg-4 avc,” *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 3, pp. 169–172, May 2004.