MULTIPLE GLOBAL ANE MOTION MODELS USED IN VIDEO CODING

A Thesis
Presented to
The Academic Faculty

by

Xiaohuan Li

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

MULTIPLE GLOBAL ANE MOTION MODELS USED IN VIDEO CODING

Approved by:

Professor Russell M. Mersereau,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Joel R. Jackson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Anthony Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David V. Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Hermann M. Fritz
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: 17 November 2006

To my parents, Jiesheng Li and Xiangmei Zeng;

To my husband, Ji Chen;

To my son, Dawson Chen.

ACKNOWLEDGEMENTS

I owe my achievement of Ph.D. greatly to my parents. Had it not been for their encouraging and cultivation during my teenage years, I may not even have pursued Ph.D. in my life.

I would like to thank my advisor Dr. Joel R. Jackson, for his invaluable support and guidance on my research and study. His intelligence and open-mindedness have helped to make my Ph.D. experience not only enlightening but also enjoyable. Dr. Jackson has not only tried his best to cultivate my academic and research capabilities, but also encouraged my collaboration with the industry during my graduate years.

I must also express my deepest gratitude to Dr. Russell M. Mersereau, who from time to time discusses with me all my technical confusions in the finest details, refers me to extensive literature in the field of image processing and reads every publication of mine with great patience.

I want to also thank Dr. Aggelos K. Katsagglos from Northwestern University, for his precious advices on my research goals and continuous support during my graduate studies.

I must also thank Dr. Anthony Yezzi, Dr. David V. Anderson and Dr. Hermann M. Fritz to serve on my reading committee.

I'd like to extend my gratitude to staff at Georgia Tech- Savannah as well as staff and students at Center of Signal and Image Processing. Mr. Page Siplon supported me through the port of Savannah since summer of 2004. Mr. Mike Potter, Ms. Pat Potter, Ms. Christy Ellis and Ms. Yvonne D. Bridges helped me on miscellaneous administrative items. Mr. Keith May fixed a dozen of my computer problems.

I'd also like to thank Hyungjoon Kim for sharing his expertise on H.264 video

coding with me.

Last but not least, I must thank my husband Ji, for his warm support and companionship during my graduate years. I'd also like to appreciate the arrival of my son Dawson, which gives me more strength and motivation to achieve my goals in life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The research presented in this dissertation explores a hybrid video codec's performance by simplifying its motion structure, instead of complicating it. This is in contrast to the latest compression standard H.264, and the majority of video researchers who are exploring more complex motion models. Specifically, we propose to use global motion models instead of local block-wise motion vectors to compress motion information between consecutive frames.

To cover the frequently occurring operations of rotation and zooming in global motion, a 6-D affine model is adopted instead of the more common 2-D translational one. To account for multiple motion objects in a video frame, motion segmentation is implemented based on the scalable motion field of an H.264 encoder. An affine model for each segment is estimated and used for global motion compensation of the corresponding areas. A warped reconstruction of the entire video frame is constructed using the segmentation map.

The multiple affine models are predicatively compressed with a specially designed vector quantizer, which consists of a long main dictionary stored off-line and a short cache word list on line. The cache word list is searched for a match each time an affine model is quantized. The main dictionary is checked only when a "miss" happens.

While reconstructing the current frame with multiple affine models, the proposed video codec system does not discard the classical block-matched reconstruction of each macroblock. Specifically, a macroblock can be reconstructed under any of the original H.264 inter or intra modes, or, with one of the affine models. Hence we add N affine modes to the original macroblock mode list of I4, I16, P16x16, P16x8, P8x16,

P8x8, P4x4 and DIRECT, where N is the number of countable motion objects in the frame. One of the new modes is chosen by Lagrange optimization. By elongating the mode list and spending moderately more bits on mode indication, we save the encoder the prohibitive effort of transmitting a segmentation map to the decoder.

Finally we present the experiment results of our system, in comparison with the latest published version of JM, the H.264 codec reference software. Our system manifests a notable gain (up to 0.8 dB) in rate-distortion performance when the video stream bit rate is below 100 kbps. 30%-70% of the macroblocks in a P-frame end up being encoded by the affine modes. The proposed system also shows many other advantages over traditional codecs, such as less pronounced blocking artifacts and more error resilience.

CHAPTER I

INTRODUCTION

1.1 Motivation

H.264, also known as MPEG-4 part 10, is the latest international video coding standard approved by ITU-T and ISO/IEC. It outperforms all its predecessors by a considerable margin in terms of its rate-distortion optimization. It introduces a variety of new features to enhance the encoder's ability to predict video content [2], at the price of increased complexity. The reforms brought by H.264 span the entire process of encoding a video sequence, including motion estimation, motion compensation, block mode selection, the DCT transform, de-blocking and entropy coding.

Among these new features, the motion-related techniques contribute most significantly to the gain in PSNR or reduction in bit rate. More precisely, H.264 provides a motion model that predicts the current frame more accurately. This is achieved by a motion structure that is much more complicated, memory demanding and computationally expensive than with prior algorithms. For example, the expansion of a single reference frame to multiple reference frames adds one more dimension to the original 2-D motion vector. The increased precision of motion vectors with quarter-pixel resolution and pointing across picture boundaries requires a longer encoding table and more bits for each item. Blocks of variable and smaller sizes demand on a larger number of motion vectors per frame, and additional bits to indicate the mode of each block.

While imposing a tremendous demand on memory allocation and computation complexity like some of the other new features, the motion related modifications also increase the portion of bitstream that must be spent on motion information.

When we review the evolution of video coding standards, we can see the history of the development of motion models. The evolution of motion compensation procedure has followed two routes: (1) refining the accuracy of the motion vector and (2) shrinking and varying the size of the compensated block. More specifically, displacement vectors evolved from integer-pixel in MPEG-1/H.261 to half-pixel in MPEG-2/H.263, and finally to quarter-pixel in H.264; block sizes have been refined from the solo pattern of 16x16 in H.261/263/MPEG-1/2 to include 8x8 in MPEG-4, and eventually 4x4 and even the non-square shapes of 16x8 and 8x16 in H.264.

While this more and more complicated motion structure successfully improves the prediction process and diminishes the number of non-zero coefficients in the DCT matrix, it increases the number of bits allocated to motion vectors. The strategy has so far been effective in shrinking the bitstream, because the increase in motion bits is more than offset by a significant decline in residual bits. However, this is only true when the refined residual errors are quantized with a small enough quantization step-size before they are transformed. In other words, a coarse quantization step-size will set the fine residual errors to zero anyway, and the advantage of a sophisticated and accurate motion compensation scheme is lost. Furthermore, since small quantization step-sizes are associated with high bit rate of transmission, the advanced motion models are only effective when abundant bandwidth is available. Most of the time, this is the case, given the fast development of communication network technologies today. However, there are still many important implementations with limited transmission bit rate constraints, such as wireless video conference, etc. The following is a study of the rate-distortion performance of MPEG-2, MPEG-4 and H.264.

We can see in Figure 1 that H.264 beats MPEG-4 and MPEG-2 in coding efficiency (in terms of the objective reconstruction quality measure PSNR) over almost the entire bandwidth, especially at the higher end when bit rate exceeds 100 kbps.

Figure 2 illustrates the percentage of bits allocated to motion related information

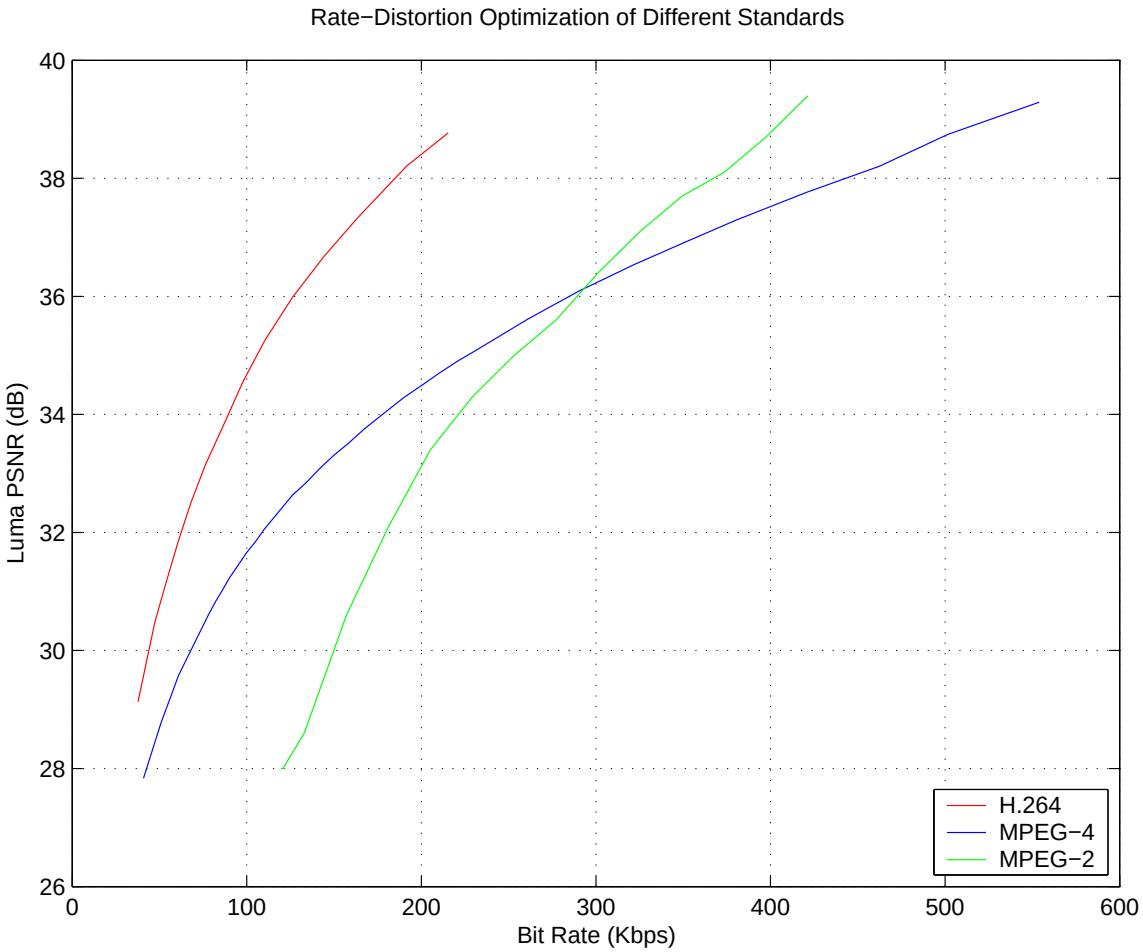


Figure 1: Rate distortion performance of MPEG-2 (simple profile) , MPEG-4 (baseline) and H.264 (baseline) standards, on the first 300 frames of *foreman* QCIF (video format with frame size 176×144 , 30 Hz. All with only the first frame as the intra frame.

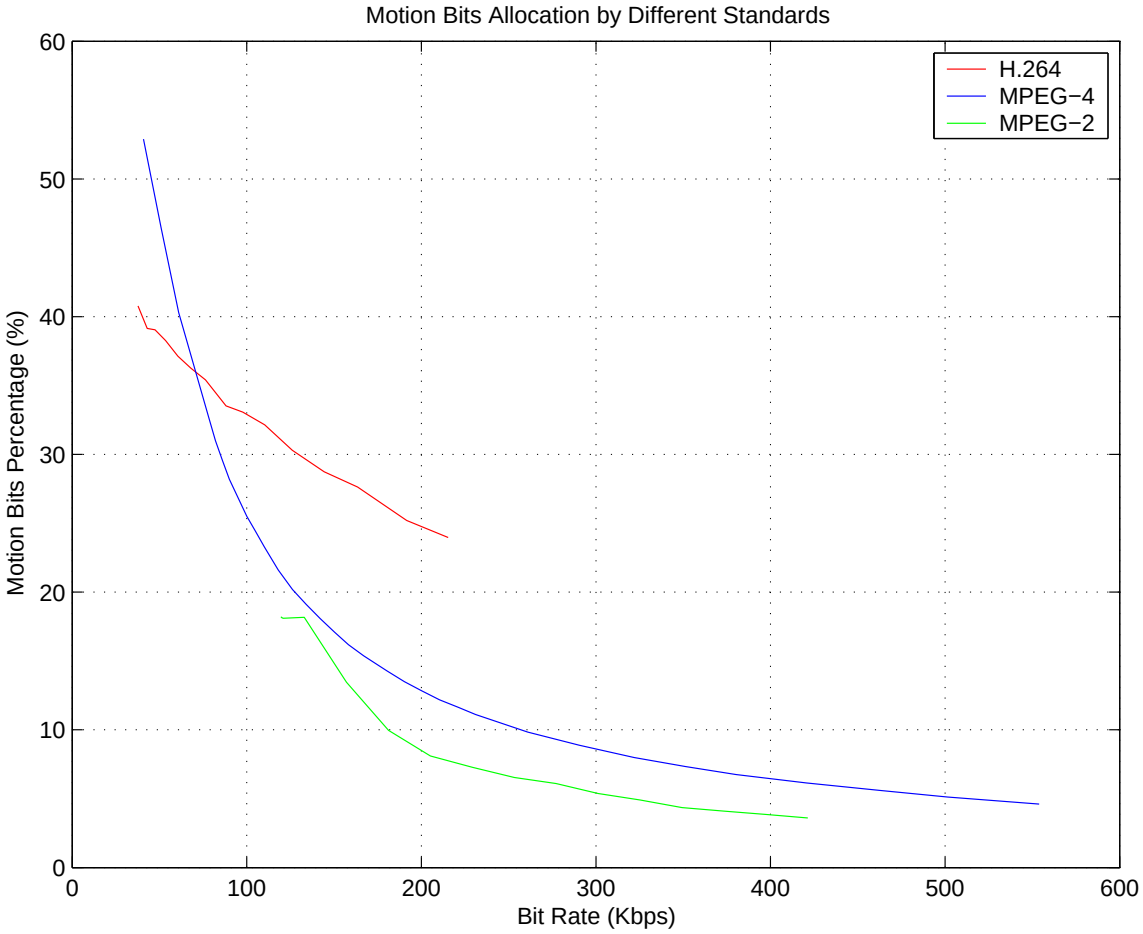


Figure 2: Motion bits distribution by MPEG-2 (simple profile) , MPEG-4 (baseline) and H.264 (baseline) standards, on the first 300 frames of *foreman* QCIF, 30 Hz. All with only the first frame as the intra frame.

(intra/inter mode, MB partition, motion vectors, etc.) because of rate-distortion optimization under a range of network bandwidths. For MPEG-2 and MPEG-4, as the available bit rate goes up, the total number of bits spent on motion vectors remains about the same, as a result of the straightforward motion estimation schemes embodied in these standards. However, in H.264 the motion vector selection is optimized by a Lagrange parameter similar to that of the mode selection. Therefore the number of H.264's motion bits increase at the higher end of the bandwidth. (This will be elaborated in the following chapters.) In both cases, the motion portion increase in the bitstream doesn't catch up with the increase in bits spent on residual errors (as a result of the Quantization Parameter(QP) decrease). That explains why the overall percentage of motion bits falls as the bandwidth escalates. Nevertheless, when the upper limit QP defined in the standard is reached, the residual portion of the bitstream stops expanding, and the motion portion approaches a limit consequently.

On the other hand, with the same bit rate, the percentage of bandwidth spent on the transmission of motion information goes up as the standards become more sophisticated, and as better and better PSNR is achieved, according to Figure 2. This leads to the conclusion that increase of expenditure on motion bits appears to always be effective in raising the coding efficiency. Each standard reaches its limit of motion calculation constrained by the computation power limits. While there will definitely be a limit on this path of video coding technology, we can't help asking: are there other approaches to improve video codecs when we are waiting for the next revolution in the chip industry to launch the one in the video coding industry? This is the question we are trying to answer with the work presented in this thesis.

1.2 Scope of Thesis

While a complete hybrid video codec with an H.264 structure has been constructed and modified to accommodate added affine motion estimation and compensation modules, the affine motion estimation techniques are the basis, but not the contribution and originality, of this thesis. However, different affine model estimation algorithms have been attempted for the purpose of motion compression and minor novel changes have been implemented on the existent methods.

1.3 Organization of Thesis

The thesis presentation is organized as follows. Chapter 2 is an introduction of the fundamental techniques upon which the thesis work is built. The most important include motion compression methods in the hybrid video coding standards and their development trend; the mathematics of affine motion model parametrization and vector quantization. Chapter 3 and chapter 4 are the two major chapters, offering a comprehensive coverage of the two proposed video codecs with global affine motion model for MPEG-4 sequences with the contour-defining alpha channel, and for H.264 sequences without an alpha channel, respectively. Chapter 5 is a probe into the error resilience of the proposed system. Chapter 6 concludes the thesis with simulation results, technical conclusions and a discussion of remaining issues on the system.

CHAPTER II

BACKGROUND

2.1 Motion Structure in Hybrid Video Codecs

2.1.1 Video Coding Standards and Motion Compensation Techniques

2.1.1.1 Video Standard History

All existing video coding standards are categorized as hybrid coding standards. They exploit both spatial and temporal redundancy in a sequence of video frame, in order to achieve a high compression ratio. Motion Compensation (MC), which represents a target block of pixels in the current frame by copying a similar block of pixels in a cached reference frame, and Motion Estimation (ME), which locates the block to be copied, are the major techniques to reduce temporal redundancy. Since the birth of the first hybrid coding standard H.120 version 2 in 1988, its successors H.261 (1991), MPEG-1 (1993), MPEG-2/H.262 (1994), H.263 (1996), MPEG-4(1999) and the latest H.264 (2003), have achieved greater and greater compression rates by incorporating new techniques and upgrading already built-in techniques. The most significant portion of the increase comes from the refinement of the MC and ME techniques.

2.1.1.2 Hybrid Video Codec Structure

MPEG video frames are categorized as intra coded (I-) frames, temporally predicted (P-) frames and bi-directionally (B-) predicted frames [1]. I-frames are allocated more bits than the others, because the DCT of the whole original frame is coded, i.e., only the spatial redundancy is exploited. For P- or B-frames, the DCT of the motion compensation error frame (which is usually significantly lower in intensity than the original frame) and the motion vectors are coded, i.e., both spatial and temporal redundancies are exploited. The existing MPEG standards have mainly used block

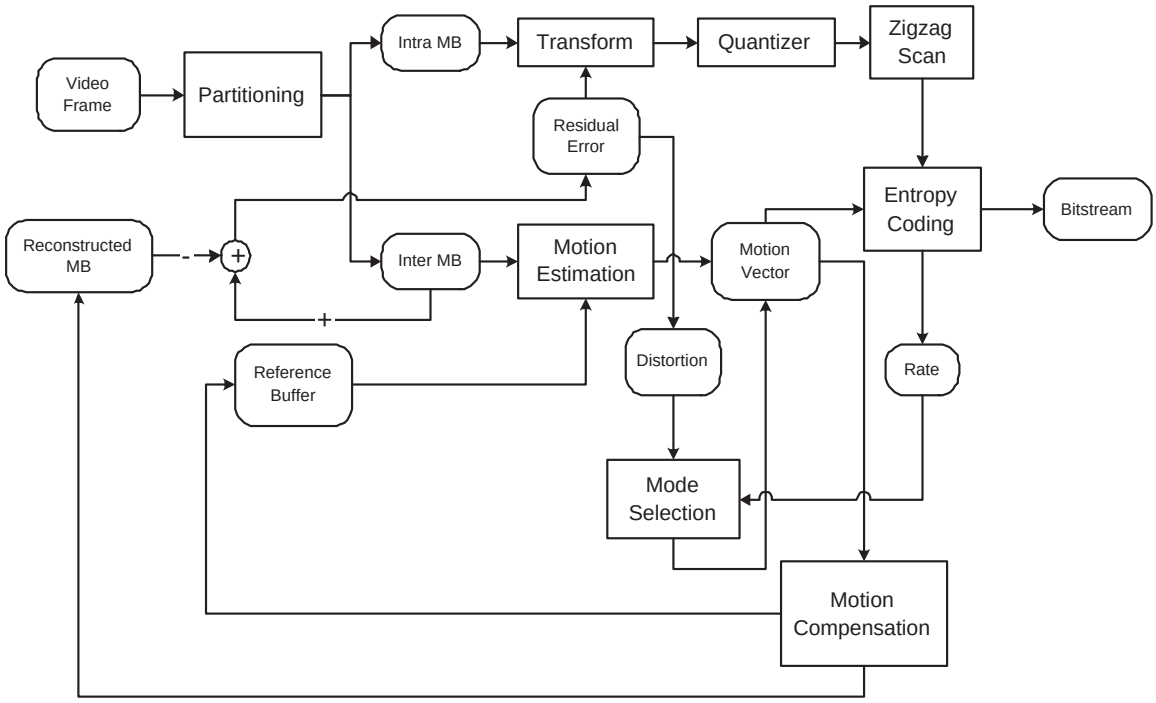


Figure 3: Block diagram of a hybrid video encoder.

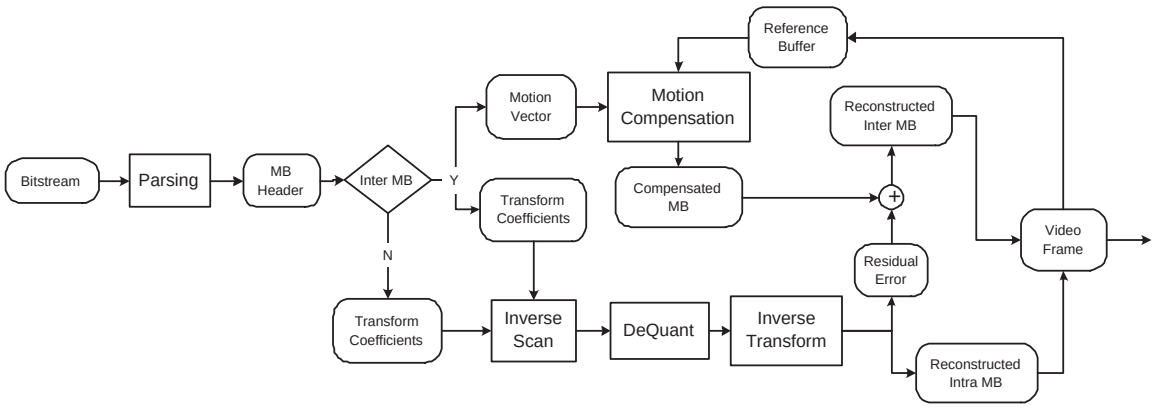


Figure 4: Block diagram of a hybrid video decoder.

matching for motion estimation. Every video frame to be coded is divided into squares of 16×16 or 8×8 pixels called macroblocks (MBs) or blocks. For each MB in the frame, a search is made in the reference frame over an area around the position of the target MB that allows for a certain range of translation specified by the coder. The result of the search is the MB with the least prediction error, usually a mean square error (MSE), or mean absolute difference (MAD). The estimated two-dimensional displacement of the MB is referred to as the motion vector.

Figures 3 and 4 together offer a overview of the basic functional modules in a hybrid video codec. The input to the video encoder is a sequence of video frames. All the MB's in an I-frame (and some of the MB's in a P-frame) are intra-MB's, which are coded directly without any prediction (intra-MB's are spatially predicted in H.264, which will be discussed in the following sections). Inter-MB's, however, need to be motion estimated and expressed as a group of MV's and a matrix of residual errors before they can be further processed. The residual errors of an inter-MB, as well as the intra coefficients of an intra-MB, are fed into the transformer, where the discrete cosine transform(DCT) expresses the spatial signals as a spectral distribution in the frequency domain. The DCT coefficients are then further trimmed by a matrix of quantizers. Since the energy distribution after the transform is concentrated in the low frequency areas, a special zigzag mapping matrix is designed to scan the coefficients so that the all-zero areas are grouped together. The zigzagged coefficients, as well as the motion vectors, are entropy coded.

Since different motion estimating block sizes and different prediction modes (intra or inter) have been allowed to increase the estimation accuracy, the encoder needs to make optimal selections in terms of coding efficiency and quality. The coded MV and residual errors for each partition size can be measured by their numbers of bits. The motion compensated MB's can be compared with the original MB to produce a PSNR that measures the reconstruction quality. The two parameters coding efficiency

and quality, are jointly considered in the mode selection module. Once an optimal prediction mode has been selected, the reconstructed MB is written to the reference frame buffer, while the coded bits are output to the bitstream.

The decoder structure is constructed from the inverse of each essential element of the encoder, except for the mode selection module, which is not needed in the decoder.

2.1.1.3 Motion Vector Accuracy and Motion Compensated Block Size

The evolution of MC has followed two routes: (1) refining the accuracy of the motion vector and (2) shrinking and varying the size of the compensated block. More specifically, displacement vectors have evolved from integer-pixel in MPEG-1/H.261 to half-pixel in MPEG-2/H.263, and finally to quarter-pixel in H.264; block sizes have been refined from the single 16×16 pattern in H.261/263/MPEG-1/2 to include 8×8 blocks in MPEG-4, and eventually 4×4 block and even the non-square 16×8 and 8×16 blocks in H.264. Although in theory the motion vector accuracy can be quantized still finer. G. Birod [7] in 1993 quantified the effect of fractional-pixel accuracy on the efficiency of a MC predictor in conjunction with various spatial interpolation filters, and concluded from experiments that for a block size of 16×16, quarter-pel resolution is sufficient for TV signals, and half-pel resolution is sufficient for videophone signals. Although block sizes have changed since these experiments, one can generalize that the compression rate of a hybrid coder cannot be indefinitely improved by unlimited refinement of the motion vector accuracy. On the other hand, a discrete block-size as low as 4×4, does not much room for further downsizing. In other words, if the hybrid video coder is to be improved in terms of its compression efficiency in the future, more MC techniques need to be investigated, instead pushing the above two refinements to further extremes.

2.1.1.4 Multiple Reference Frames

The essence of the efforts to adjust MV accuracy and block size is to provide more spatially distributed prediction options, so as to reduce the prediction errors. Another approach that shares the same philosophy is to allow for more prediction candidates in multiple reference frames [8], which was incorporated a standard in H.264. The magnified computation load for ME can be trimmed by non-exhaustive searching alternatives [9].

2.1.1.5 Multi-Hypothesis Motion Compensation

While modifying the parameters for ME in the standard block-matching paradigm is successfully reducing compensation errors, non-conventional MC and ME methods using spatial and temporal motion correlations are being explored also.

Overlapped block matching compensation (OBMC) OBMC estimates a pixel by superimposing predictions using not only the MV of the block that contains it, but also the MV's of neighboring blocks [10] [11]. Since the compensated pixel is a weighted average of several values across the block boundary, OBMC not only achieves a lower compensation error, but also reduces the blocking artifacts. As a result it was standardized in H.263 in 1995. However, since OBMC entails multiple MV's and each MV affects multiple blocks, the corresponding ME has to be performed in an iterative manner. The high computation cost often makes this technique prohibitive.

Bi-directional Prediction Applying the idea of OBMC to the temporal domain, the superimposition of blocks from different reference frames (the previous and the next as in MPEG and H.263) often allows a more accurate prediction than if only one block is used from the previous frame. In the so-called bi-directionally predicted frame (B-frame), each block is associated with two MV's and two error blocks. B-frames are particularly efficient in dealing with uncovering areas, for the uncovering

block content can often be found in the future frame if it is missing in the previous one. However, because of the use of a future reference, B-frames suffer from increased delay and the necessity of additional storage [12].

Multi-Hypothesis Motion Compensation Multi-hypothesis motion compensation, which is a generalization of both OBMC and B-frames, is realized by the introduction of multi-hypothesis-mode macroblocks in H.264 [13]. Under the multi-hypothesis-mode, superimposed prediction pairs can come from any reference frames in the buffer. The combination can be forward-backward (as in the B-frame case), as well as forward-forward, backward-backward, or even forward-current, backward-current and current-current (as in the OBMC case). Moreover, the number of past or future reference frames is not confined to two. The average of the compensation pairs provides a much closer prediction at the price of an increased computational load.

2.1.2 Search Strategies for Block-Matching ME

2.1.2.1 Principle of Block-Matching

While it is the most straightforward and intuitive approach for motion estimation, block-matching is also the most widely used motion search strategy. A comparison of the target block is made with every possible reference block in a certain temporal and spatial range, and a difference measure (e.g. mean square error) is computed for each of them. The displacement between the reference block associated with the smallest error measure is used as the MV for the target block. The several important parameters modifiable in a block-matching search include the step size (how closely distributed the MV's are, e.g. 1 for pixel-wise ME, 16 for macroblock-wise ME), matching window size (the number of pixels in the neighborhood that are compared for the error measure), search range (the maximum possible MV) and MV accuracy.

2.1.2.2 More Search Patterns

The full-search block-matching strategy exhausts every pixel in every window that is within the search range. For example, to get a block(8×8)-wise quarter-pel motion field for a QCIF(176×144) frame, with the usual search range of 16×16, a window size of 16×16 and an MSE error criterion, $16 \times 16 \times 64 \times 64 \times 22 \times 18 = 415M$ comparison operations, $415M$ summing operations and $415M$ multiplication operations would be required for each frame, which is enormous. In order to overcome this computational drawback, a large amount of work has been done to design alternative fast search algorithms for block-matching, such as 2-D logarithmic search [14], three-step search [15], conjugate direction search [16], cross search [17], four-step search [18], block-based gradient descent search [19], etc. While these approaches attempt to shorten the search path on a square shaped search window, diamond [20] and hexagon [21] shaped search patterns have also been found to further speed up the search job. Some of these fast algorithms are so simple and effective that they have been recommended, although not specified, by the standard.

2.1.2.3 Hierarchical Structure

A hierarchical motion structure is one of the many techniques designed to reduce the computational load. Exploiting the fact that neighboring motion vectors tend to similar, the current and the reference frames are filtered and sub-sampled to obtain two pyramids of frames, with the original full-size frames at the bottom, and their down-sized versions on top in a descending order. Conventional blocking-matching is performed on a top level, resulting in a small motion field that can be stretched into a rough estimate of the larger motion field for the level beneath. This multi-resolution structure not only reduces ME computation, but also avoids local optima, which lead to false MV's.

2.1.2.4 Rate-Constrained ME

Rate-constrained ME was first proposed by by Sullvan et. al. in 1991 [22], and included in the H.263/MPEG-4 standards [49]. Instead of minimizing the measured error only, rate-constrained ME seeks for an MV that minimizes the weighted sum of the error and the bits required for coding the MV itself, as in (1).

$$mv' = \arg \min_{mv \in M} (D_{DFD}(mv) + \lambda_{MOTION} R_{MOTION}(mv)) \tag{1}$$

where M is the set of motion vectors within the search range. The measured distortion associated with the motion vector is

$$D_{DFD}(mv) = \sum_{(x,y) \in A} |s[x, y, k] - s'[x - mv_x, y - mv_y, k - 1]|^p \tag{2}$$

where p=1 for sum of absolute difference (SAD) and p=2 for sum of squared errors (SSD). The choice of is made through experimental results and defined as $\lambda_{MOTION} = \sqrt{\lambda_{MODE}}$ for SAD and $\lambda_{MOTION} = \lambda_{MODE}$ for SSD, where is the Lagrange parameter for mode selection and obtained by $\lambda_{MODE} = 0.85 \times 2^{(Q_{N264}-12)/3}$ in H.264. As a consequence, a better performance in terms of rate-distortion optimization is accomplished.

2.1.3 Other ME Techniques

2.1.3.1 Gradient-Based Motion Estimation

If we model a video frame as a function of spatial and temporal variables as

$$I(x + mv_x dt, y + mv_y dt, t + dt) = I(x, y, t) \tag{3}$$

and assume that the intensity of each pixel remains the same in the two frames connected by the motion field, then if we furhter assume that there is a smooth change of pixel intensities over x, y and t, (3) can be expanded in a Taylor series as

$$I(x, y, t) + \frac{\partial I}{\partial x} mv_x dt + \frac{\partial I}{\partial y} mv_y dt + \frac{\partial I}{\partial t} dt + O(dt^2) = I(x, y, t) \tag{4}$$

Simplifying this equation gives the famous optical flow constraint equation

$$\frac{\partial I}{\partial x}mv_x + \frac{\partial I}{\partial y}mv_y + \frac{\partial I}{\partial t} = 0 \tag{5}$$

If we treat v_x and v_y as two unknowns, we will need a second equationin a dition to (5) to solve for them. Lucas and Kanade proposed applying (5) to a small region (with more than 2 pixels) of the image to obtain a local MV. To satisfy the smoothness condition, the spatial and temporal gradients are always low-pass filtered to combat high frequency noises. Although the gradient approaches require several iterations to reach an accurate estimate, still the computational load is far less than block-matching.

2.1.3.2 Phase Correlation Motion Estimation

According to the Fourier shift property, a displacement in the spatial domain corresponds to a phase factor in the frequency domain. Specifically, if we rewrite (3) as

$$f_2(x, y) = f_1(x - mv_x, y - mv_y) \tag{6}$$

and compute the Fourier transform of the two sides, we have

$$F_2(u, v) = F_1(u, v)exp(-i(u \cdot mv_x + v \cdot mv_y)) \tag{7}$$

The normalized cross power spectrum is then given by

$$\frac{F_2(u, v)F_1^*(u, v)}{|F_1(u, v)F_1^*(u, v)|} = exp(-i(u \cdot mv_x + v \cdot mv_y)) \tag{8}$$

By inverse Fourier transforming (8), we expect to obtain a Dirac-delta function centered at (mv_x, mv_y). Then the problem boils down to locating the peak in a 2-D domain [25]. Although the phase correlation technique is supposed to find the true motion for each transformed block, its high computational load has made it impractical in compression uses.

2.2 Motion advances in H.264

2.2.1 Overview of H.264 standard

H.264 is the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) [2]. The standardization of H.264/AVC announced a new era in the video coding development with significantly improved compression performance and provisions for a "network-friendly" video representation addressing "conversational" (video telephony) and "non-conversational" (storage, broadcast, or streaming) applications. H.264/AVC has achieved an unprecedented advance in rate-distortion efficiency relative to existing standards.

The success of H.264 is effected by the combination of a dozen new techniques. We shall only discuss the video coding structure related features, thus we shall skip the streaming related designs, which are exemplified by the parameter set structure, the NAL unit syntax structure, flexible slice size, flexible macroblock ordering (FMO), arbitrary slice ordering (ASO), redundant pictures and SP/SI synchronization/switching pictures, etc.

Among the features that enhance the coding efficiency, several motion-related ones enhance the ability to predict a video frame, and thus contribute the most in the gain in reconstruction quality. We will elaborate on this group of technical highlights in section 2.2.2. Other design features which also contributed to the enhancement, are briefly discussed in section 2.2.3.

2.2.2 Motion-related features of H.264

2.2.2.1 Variable block-size for motion compensation

H.264 provides for a more flexible selection of motion compensation block sizes and shapes than any previous standard, with a minimum luma motion compensation block size as small as 4×4. Such a small block has a greater chance of finding a match

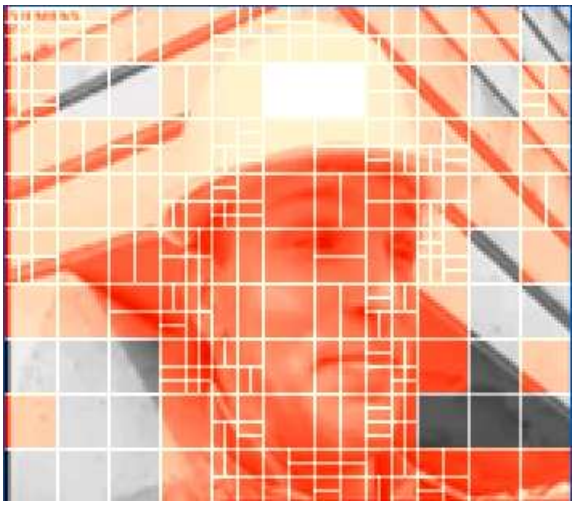


Figure 5: Optimum choice of partitions for the 3rd frame of the *foreman* sequence, from Vprove, a video compression analysis s)Aoftware.

in the reference frame than a larger block. However, a larger block may be still be used if it provides a close enough match (see Figure 5). That is why the variable block-size design decreases residual errors and the corresponding number of bits [5].

2.2.2.2 Quarter-pixel motion compensation accuracy

Most of the standards prior to H.264 enable half-pixel motion vector accuracy at most. The new standard improves upon this by adding quarter-pixel motion vector accuracy, as first found in an advanced profile of the MPEG-4 Visual (part 2) standard, but it further reduces the complexity of the interpolation processing compared to the prior design. Finer motion vectors cost (1) more bits to be entropy coded (2) more complex interpolation on the reference frame, but at the same time offer more compensation candidate blocks and reduce the residual errors.

2.2.2.3 Motion vectors over picture boundaries

While motion vectors in MPEG-2 and its predecessors were required to point only to areas within the previously-decoded reference picture, the picture boundary extrapolation technique first found as an optional feature in H.263 is included in H.264/AVC.

2.2.2.4 Multiple reference picture motion compensation

The new design extends upon the enhanced reference picture selection technique found in H.263++ to enable efficient coding by allowing an encoder to select, for motion compensation purposes, among a larger number of pictures that have been decoded and stored in the decoder; while predictively coded pictures (called "P" pictures) in MPEG-2 and its predecessors used only one previous picture to predict the values in an incoming picture. The same extension of referencing capability is also applied to motion-compensated bi-prediction, which is restricted in MPEG-2 to using two specific pictures only (one of these being the previous intra (I) or P picture in display order and the other being the next I or P picture in display order).

2.2.2.5 Decoupling of referencing order from display order

In all prior standards, the ordering of pictures for motion compensation in the reference frame buffer are strictly determined by the ordering of pictures in final display. In H.264/AVC, these restrictions are largely removed. The encoder can now choose the ordering of pictures for referencing and display purposes with a high degree of flexibility constrained only by a total memory capacity bound imposed to ensure decoding ability. Removal of the restriction also enables removing the extra delay previously associated with bi-predictive coding.

2.2.2.6 Decoupling of picture representation methods from picture referencing capability

In prior standards, pictures encoded using some encoding methods (namely bi-predictively-encoded pictures) could not be used as references for prediction of other pictures in the video sequence. By removing this restriction, the new standard provides the encoder more flexibility and, in many cases, an ability to use a picture for referencing that is a closer approximation to the picture being encoded, and therefore generate lower residual errors.

2.2.2.7 Weighted prediction

A new innovation in H.264/AVC allows the motion-compensated prediction signal to be weighted and offset by amounts specified by the encoder. This can dramatically improve coding efficiency for scenes containing fades, and can be used flexibly for other purposes as well.

2.2.2.8 Improved "skipped" and "direct" motion inference

In prior standards, a "skipped" area of a predictively-coded picture could not have any motion in the scene content. This had a detrimental effect when coding video containing global motion, so the new H.264/AVC design instead infers motion in "skipped" areas. [4] For bi-predictively coded areas (called B slices), H.264/AVC also includes an enhanced motion inference method known as "direct" motion compensation, which improves further on prior "direct" prediction designs found in H.263+ and MPEG-4 Visual.

2.2.2.9 Directional spatial prediction for intra coding

The previous standards apply transforms on intra blocks directly without any kind of prediction. The new standard exploits the spatial redundancy in the intra-coded areas by extrapolating the edges of the previously-decoded parts of the current picture. This predictor is applied in regions of pictures that are coded as intra (i.e., coded without reference to the content of some other picture). This improves the quality of the prediction signal, and also allows prediction from neighboring areas that were not coded using intra coding (something not enabled when using the transform-domain prediction method found in H.263+ and MPEG-4 Visual).

2.2.2.10 In-loop deblocking filtering

Block-based video coding produces artifacts known as blocking artifacts. These can originate from both the prediction and residual difference coding stages of the decoding process. Application of an adaptive deblocking filter is a well-known method of improving the resulting video quality, and when designed well, this can improve both objective and subjective video quality. Building further on a concept from an optional feature of H.263+, the deblocking filter in the H.264/AVC design is brought within the motion-compensated prediction loop, so that this improvement in quality can be used in inter-picture prediction to improve the ability to predict other pictures as well.

2.2.3 Other important features of H.264

2.2.3.1 Small block-size transform

All major prior video coding standards used a transform block size of 8x8, while the new H.264/AVC design is based primarily on a 4x4 transform. This allows the encoder to represent signals in a more locally adaptive fashion, which reduces artifacts known colloquially as "ringing". (The smaller block size is also justified partly by the advances in the ability to better predict the content of the video using the techniques noted above, and by the need to provide transform regions with boundaries that correspond to those of the smallest prediction regions.)

2.2.3.2 Hierarchical block transform

While in most cases, using the small 4x4 transform block size is perceptually beneficial, there are some signals that contain sufficient correlation to call for some method of using a representation with longer basis functions. The H.264/AVC standard enables this in two ways: 1) by using a hierarchical transform to extend the effective block size use for low frequency chroma information to an 8x8 array, and 2) by allowing the encoder to select a special coding type for intra coding, enabling extension of

the length of the luma transform for low-frequency information to a 16x16 block size in a manner very similar to that applied to the chroma.

2.2.3.3 Short word-length transform

All prior standard designs have effectively required encoders and decoders to use more complex processing for transform computation. While previous designs have generally required 32-bit processing, the H.264/AVC design requires only 16-bit arithmetic.

2.2.3.4 Exact-match inverse transform

In previous video coding standards, the transform used for representing the video was generally specified only within an error tolerance bound, due to the impracticality of obtaining an exact match to the ideal specified inverse transform. As a result, each decoder design would produce slightly different decoded video, causing a "drift" between encoder and decoder representation of the video and reducing effective video quality. Building on a path laid out as an optional feature in the H.263++ effort, H.264/AVC is the first standard to achieve exact equality of decoded video content from all decoders.

2.2.3.5 Arithmetic entropy coding

An advanced entropy coding method known as arithmetic coding is included in H.264/AVC. While arithmetic coding was previously found as an optional feature of H.263, a more effective use of this technique is found in H.264/AVC to create a very powerful entropy coding method known as CABAC (context-adaptive binary arithmetic coding).

2.2.3.6 Context-adaptive entropy coding

The two entropy coding methods applied in H.264/AVC, termed CAVLC (context-adaptive variable-length coding) and CABAC, both use context-based adaptivity to improve performance relative to prior standard designs.

2.3 Affine motion model and related work

2.3.1 Higher Order Motion Models

While block-matching is an efficient way of modeling much of the motion in a video frame, its model covers only translation, which limits its effectiveness when there is more complicated motion, such as rotation and zooming. Higher order motion models like affine models and perspective models can represent these motion types, albeit with more parameters.

If the translational motion model is expressed as in (9)

$$\begin{cases} u(x,y) = a_1 \\ v(x,y) = a_2 \end{cases}$$

higher-order models expressed on this pattern as follows, with the affine model in (9)

$$\begin{cases} u(x,y) = a_1 x + a_2 y + a_3 \\ v(x,y) = a_4 x + a_5 y + a_6 \end{cases}$$

the bilinear model in (9)

$$\begin{cases} u(x,y) = a_1 x + a_2 y + a_3 + a_4 xy \\ v(x,y) = a_4 x + a_5 y + a_6 + a_8 xy \end{cases}$$

and the perspective model in (9)

$$\begin{cases} u(x,y) = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} - x \\ v(x,y) = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} - y \end{cases}$$

2.3.2 Motion Elements

2.3.2.1 Translation

As the most common motion element in image sequences, translation refers to a rigid shift in either the x or y or both directions, i.e. (9)

$$\begin{cases} x' = x + a_1 \\ y' = y + a_2 \end{cases}$$

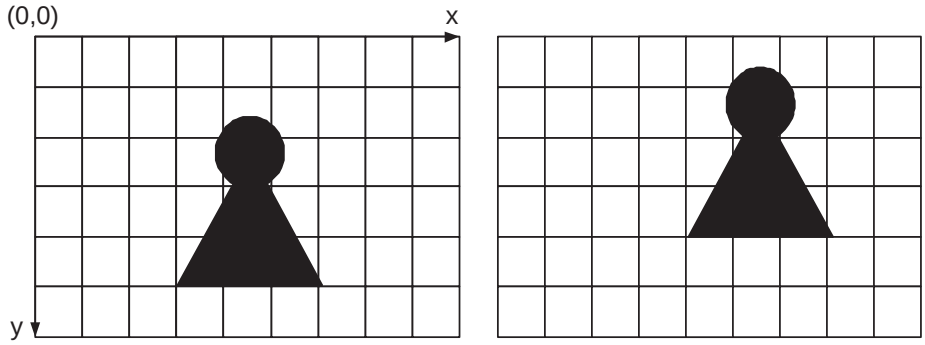


Figure 6: Translation of object, with $(a_1, a_2) = (1, -1)$.

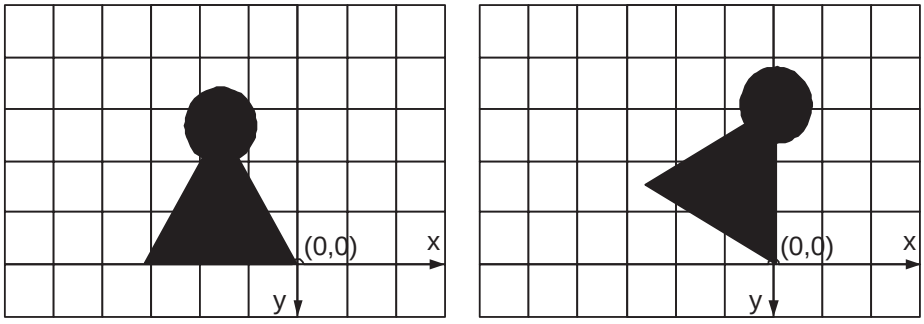


Figure 7: Rotation of object about the origin, with $\theta = 30°$.

In an affine model, a_3 and a_6 account for translations in the two dimensions.

2.3.2.2 Rotation

Rotation in a polar axis system is described as the shift on the angular axis with constant amplitude. A pure rotation through a clockwise angle is parameterized as (9).

$$\begin{cases} x' = \cos\theta \cdot x - \sin\theta \cdot y \\ y' = \sin\theta \cdot x + \cos\theta \cdot y \end{cases}$$

2.3.2.3 Zooming

Zooming, or scaling, is a general term encompassing both cases of magnifying/dilating and minifying/contracting the scene, which are very common in movie videos. A

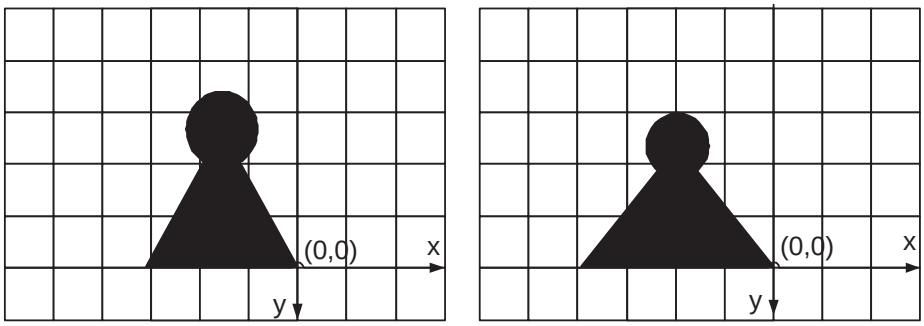


Figure 8: Zooming of object about the origin, with $(a_1, a_2) = (0.9, 1.3)$.

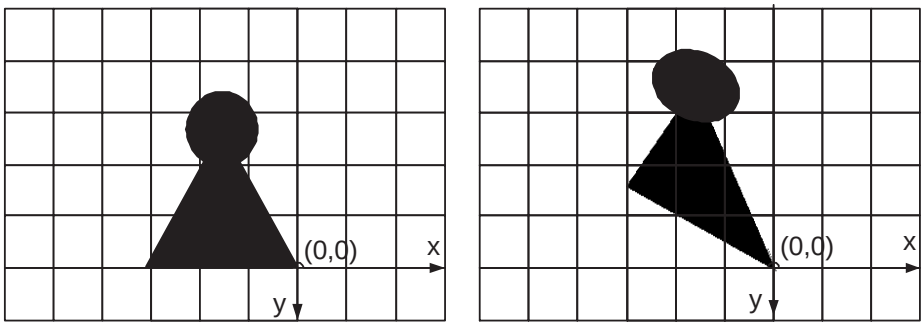


Figure 9: Vertical twist of object about the origin, with $a = -0.5$.

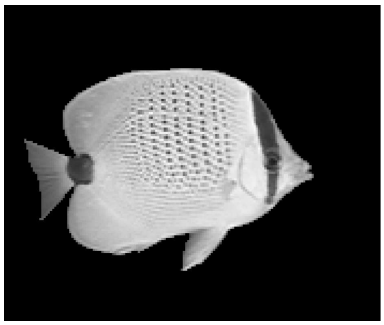
simple model describes a pure zooming as (9).

$$\begin{cases} x' = a_1 \cdot x \\ y' = a_2 \cdot y \end{cases}$$

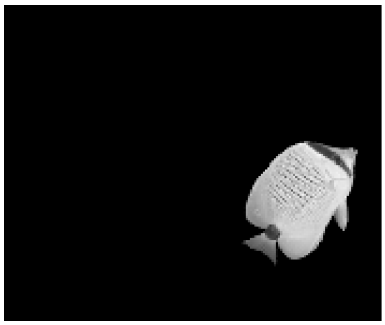
2.3.2.4 Shear

The effect of shear in a 2-D plane is that of stretching an object in the direction of one axis while maintaining it in the other, and parameterized as in (9) for a horizontal twist, or (9) for a vertical one. Though not usually found in natural video sequences' realistic motion, shear is very common in synthesized computer animations and graphics.

$$\begin{cases} x' = x + a \cdot y \\ y' = y \end{cases}$$



(a)



(b)

Figure 10: (a) is the original picture of frame0 in the *bream.qcif* sequence. By applying translation with $[a_1, a_2]' = [6, -250]$, rotation with $\theta = -45^o$ and $[a_1, a_2]' = [\frac{1}{2}, \frac{1}{2}]$ consecutively, (b) is obtained. The three motion elements can be concatenated and expressed as an affine transform operation in (9)

$$\begin{cases} x' = x \\ y' = a \cdot x + y \end{cases}$$

2.3.2.5 Panning/Tilting

Panning and tilting refer to the large motion of the camera in a certain direction, namely, a dominant global motion in the video frame. It can include translation as well as rotation. More importantly, it can be modeled more efficiently by a global motion model instead of a segmented local one.

2.3.2.6 Multiple Motion Elements in the Affine Model

As illustrated in (9)- (9), any motion element can be expressed as an operation on the 2-D coordinates of the corresponding pixel. A series of operations result in an affine motion model, in which a_3 and a_6 stand for translation, a_1, a_2, a_4 and a_5 all contain rotation, a_1 and a_5 cover zooming, and a_2 or a_4 accounts for shear. Figure (10) shows an example of the combined effect of several motion elements mentioned above. Specifically, the motion between Figure 10.a and 10.b consists of translation with $[a_1, a_2]' = [6, -250]$, rotation with $\theta = -45^o$ and $[a_1, a_2]' = [\frac{1}{2}, \frac{1}{2}]$. Plugging these

parameters in ((9)), (9) and (9), we get the overall affine motion model as

$$\overrightarrow{mv} = \begin{bmatrix} mv_x \\ mv_y \end{bmatrix} = \begin{bmatrix} 0.414 & -1.414 \\ 1.414 & 0.414 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 6 \\ -250 \end{bmatrix} \tag{9}$$

2.3.3 Affine Motion Model Estimation

2.3.3.1 Gradient-Based Method

As explained for the gradient-based motion estimation for the translational model, the gradient-based estimation for affine motion parameter set can be derived from the optical flow constraint equation (5). Except that at least 6 pixels would be needed in order to solve for a non-singular solution for the equation. After some simple manipulation, it can be seen that the affine model parameter can be formed by solving (10).

$$\begin{bmatrix} \sum xI_xxI_x & \sum yI_xxI_x & \sum I_xxI_x & \sum xI_yxI_x & \sum yI_yxI_x & \sum I_yxI_x \\ \sum xI_xyI_x & \sum yI_xyI_x & \sum I_xyI_x & \sum xI_yyI_x & \sum yI_yyI_x & \sum I_yyI_x \\ \sum xI_xI_x & \sum yI_xI_x & \sum I_xI_x & \sum xI_yI_x & \sum yI_yI_x & \sum I_yI_x \\ \sum xI_xyI_y & \sum yI_xyI_y & \sum I_xxI_y & \sum xI_yxI_y & \sum yI_yxI_y & \sum I_yxI_x \\ \sum xI_xyI_y & \sum yI_xyI_y & \sum I_xyI_y & \sum xI_yyI_y & \sum yI_yyI_y & \sum I_yyI_y \\ \sum xI_xI_y & \sum yI_xI_y & \sum I_xI_y & \sum xI_yI_y & \sum yI_yI_y & \sum I_yI_y \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} \sum -xI_xI_t \\ \sum -yI_xI_t \\ \sum -I_xI_t \\ \sum -xI_yI_t \\ \sum -yI_yI_t \\ \sum -I_yI_t \end{bmatrix} \tag{10}$$

where I_x, I_y and I_t are the frame's gradients in the horizontal, vertical and temporal directions respectively. All sums are done over the pixels covered by the affine model.

2.3.3.2 Fitting Translational Motion Field

If the MV's for the small partitions (e.g. 8×8 blocks) of a large region (e.g. a motion object) are known, assuming that the large region is affected by one affine transform, affine motion estimation is equivalent as fitting a 6-D point to the known MV's. More specifically, by plugging all the partitions' parameters into (11), the i and j in which

are vertical and horizontal coordinates for the pixels in the affine motion object.

$$\begin{bmatrix} i & j & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i & j & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} i + mv_x(i,j) \\ j + mv_y(i,j) \end{bmatrix} \tag{11}$$

and lining them up, we obtain a redundant equation (12)

$$A_{2N\times 6} x_{6\times 1} = b_{2N\times 1} \tag{12}$$

where N is the number of pixels in the segmentation. Taking the pseudo-inverse of A, we obtain the solution as in (13).

$$x = (A^T A)^{-1} \cdot A^T b \tag{13}$$

2.3.4 Video Compression Schemes Using An Affine Motion Model

2.3.4.1 Affine Motion Model in Place of Block-Based Translational Model

To overcome the limitation of a block-matching MC scheme, researchers have looked into the possibility of more complicated motion models in motion coding. [26], [27] and [28] are examples of such efforts. An affine model is introduced as a substitute for the translational model, applied to the same block. More bits are spent on motion information, in the hope of reducing the number of bits for the residual error.

2.3.4.2 Affine Motion Model in Conjunction with Block-Based Translational Model

[29] develops such an affine motion video codec by enlarging the block-size (to 32×32) for affine model estimation, while applying the affine transformation to the whole frame, which is implemented in series with the block-matching estimation. Remarkable progress in rate-distortion performance is reported, while a dramatic increase in computation load and codec structure complexity is observed.

2.3.4.3 Affine Motion Model for Frame-Wise Global Motion

Smolic et al proposes to use affine model on a frame-wise basis in cases of sheer camera panning [30]. Block-matched MVs are also estimated at each MB, and the choice between the global affine model and the local translational model is made by minimizing a Lagrange cost. Another significant feature of the system proposed in [30] is the use of a super-resolution (SR) technique in place of residual error coding when global motion is predominant. In order for SR be effective, long term reference frames become a must instead of an option. Due to the large overhead from long term affine models, this technique works much better with high resolution formats, e.g. 1280×720, than lower resolution formats like QCIF, CIF and SIF.

2.3.5 Sprite

The well-known technique of sprite coding was first specified in MPEG-4 in 1999 [31], in response to the need formore efficient coding for global motion, with the introduction of the concept of a video object plane and a video object layer [1]. A sprite consists of pixels that are present in the frame throughout the sequence, which are often background components. It's usually constructed by adding the occluded pixels in all following frames to the first frame (see Figure 11 for an example). Therefore the sprite contains all parts of the background that were at least visible once in an image sequence. It can be used for the reconstruction or the predictive coding of the background. A comparison of the sprite coding and the video coding method proposed by this thesis will be offered in later chapters.

According to the way its motion is transmitted in the bitstream, a sprite can be labeled as a static or a dynamic sprite.

2.3.5.1 Static Sprite

For static sprites, long-term continuous motion information needed throughout the sequence needs to be estimated off-line before the coding process starts. As a result,

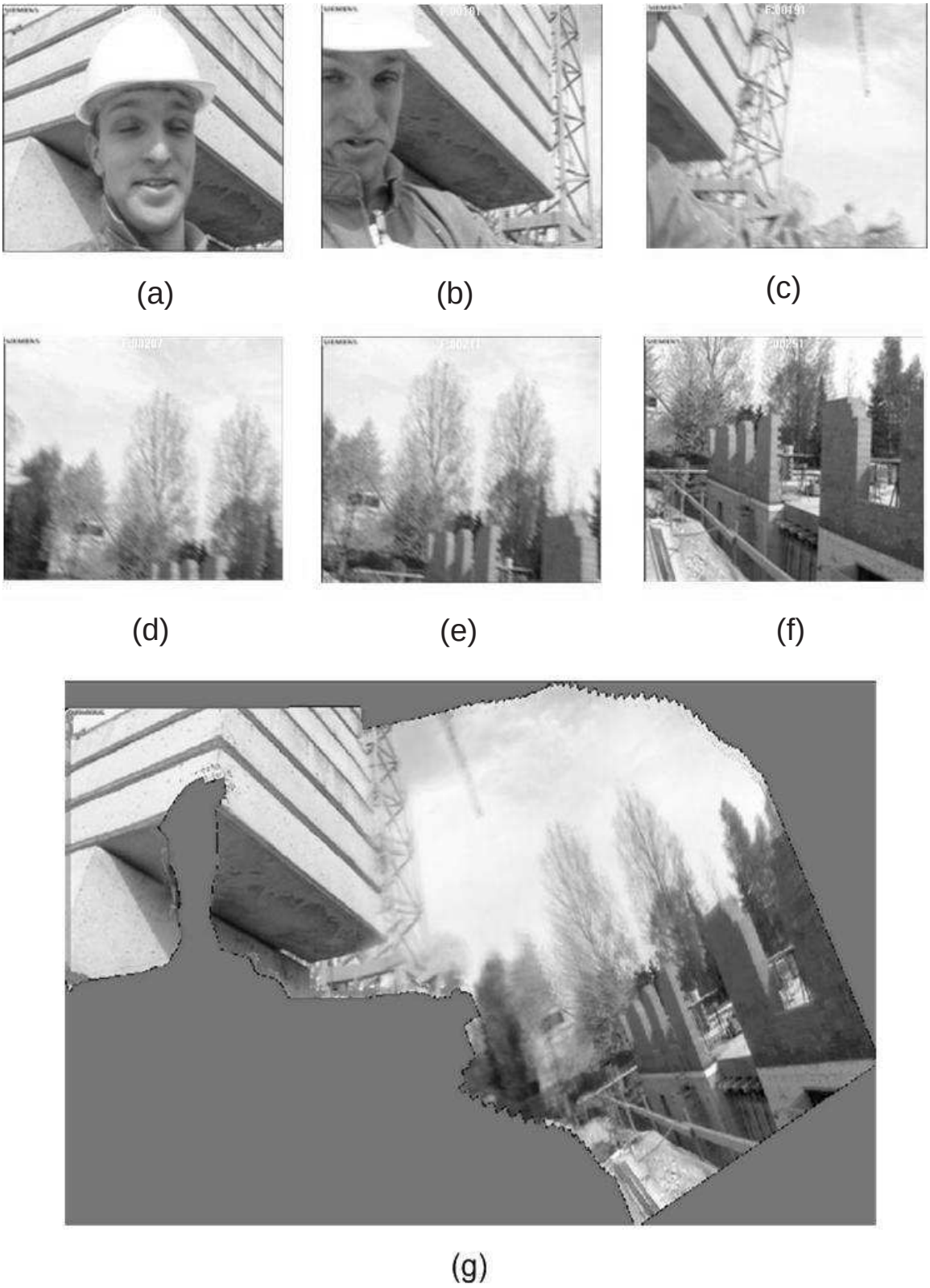


Figure 11: Generation of the sprite from the background of a video sequence. (a) (f) are frames 1, 181, 191, 207, 211, 232 of *foreman.qcif*. (g) is the generated sprite background from [32]

it is not suitable for on-line video communication purposes.

2.3.5.2 Dynamic Sprite

Dynamic sprites' motion information is coded and transmitted on a frame-wise basis, like the rest of the frame, together with the residual error, which will be added to the warped sprites at the decoder side.

One significant constraint of sprite coding, which also restricts the use of video objects in MPEG-4/7/21, is the fact that the whole infrastructure is constructed upon apriori segmentation maps, which must be very accurate.

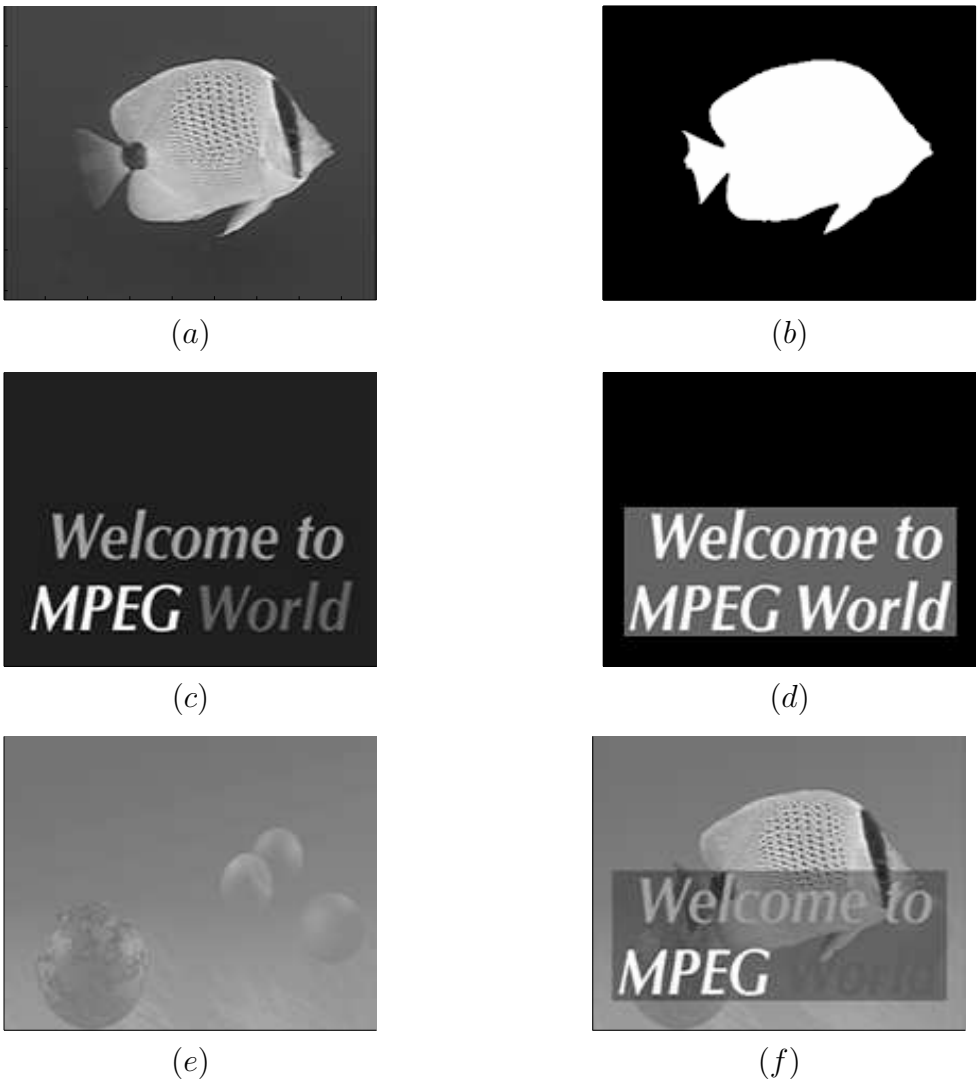


Figure 12: Synthesized video from 3 VOP's. (a) and (b) are the luma and α channels for VOP 1 (the fish) respectively. (c) and (d) are the luma and α channels for VOP 2 (the caption) respectively. (e) is the luma for VOP 0 (the background), and since the transparency for the background is always zero, no α channel is needed for VOP 0. Note that the α channels for VOP 1 and 2 are both grayscale, meaning that they define a transparency degree for each pixel in that VOP. A binary α channel would only define if a pixel is transparent or opaque. Finally, (f) is the video frame synthesized from the weighted sum of the 3 VOP's. All frames are from the 95th frame of bream.qcif.

CHAPTER III

GLOBAL AFFINE MODEL FOR MPEG P-VOP

3.1 Introduction

MPEG-4 and higher versions have introduced the concept of a Video Object Plane (VOP), defined by its shape and texture. Masked by shape (the α-plane), the VOP is often a contoured independent object, whose texture, motion and shape information are all transmitted in the bitstream, according to MPEG-4 visual [33]. The bitstreams of several VOP's and accompanying composition information can be multiplexed so that the decoder receives all the information to decode the VOP's and arrange them into one video scene, as illustrated in Figure 12. This results in interactivity and flexibility for standardized video and multimedia applications [34].

The shape of an object is defined as the α-plane, or the α-map, which specifies whether or not a pixel belongs to the corresponding VOPt. An α-plane can be either grayscale or binary. A grayscale α-plane has each of its pixel values ranging from 0 to 255. A binary α-plane has each of its pixel as either 0 or 255. α value 255 means the object is opaque at that pixel, namely, it's fully displayed; value 0 means the object is transparent, i.e. it's totally unseen in the synthesized image. Any in-between values define the transparency of the object pixel, the smaller the more transparent (see Figure 12 (b) and (d) for instance). When a VOP is a rigid moving object, which is very often the case, its motion is more likely to be global than the motion in a synthesized multiple-VOP frame.

On the other hand, in many cases, the block-based motion vectors are very similar in a neighborhood or even the whole frame. Putting the motion vectors directly into the bit stream is like writing the raw pixels of each frame to disk. The concept of

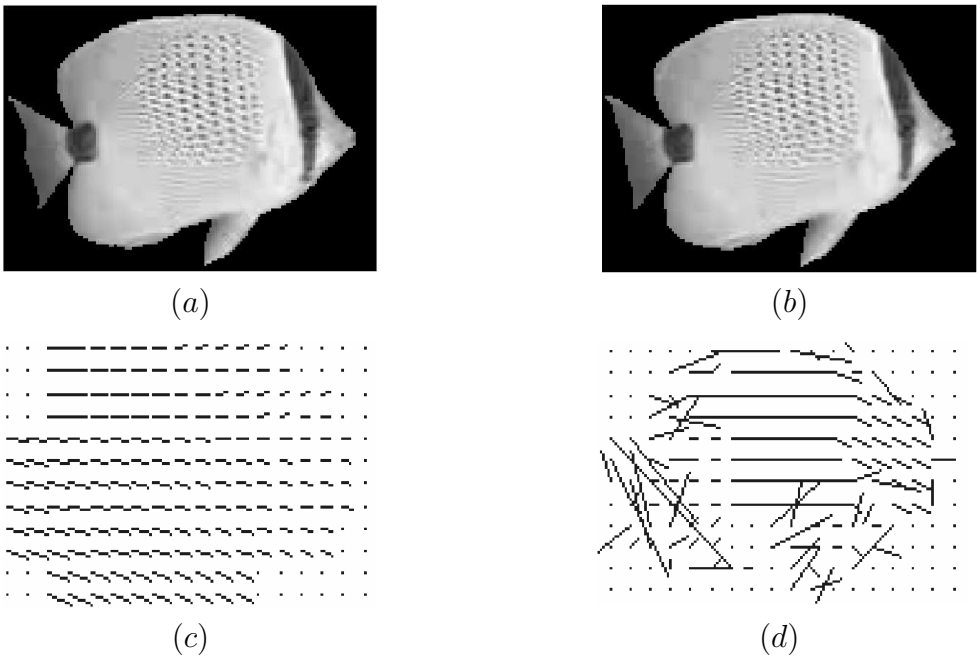


Figure 13: (a) and (b) are frame 3 and frame 4 of the sequence bream2-1.qcif respectively, with binary alpha plane defined; (c) and (d) are block-wise motion vector fields from affine and block-matching estimation approaches respectively. From subjective observation, one can judge that the motion between (a) and (b) is mainly global translation and rotation. The reason for some irregular motion vectors in (d) is that the optimization goal of block-matching method is MSE of the compensated block, instead of the real motion of the target block.

prediction of motion vectors has been included in MPEG to deal with this situation. Research of corresponding predicting algorithms has been carried out, e.g. [35], [36] and [37]. All of the approaches only decrease the intensity of the coded motion vectors, but not the number of motion vectors. For a 176×144 frame, there are $22 \times 18 = 396$ blocks and therefore 792 real numbers to be coded. This means either a large fraction of the bit stream must be occupied by motion information, or low accuracy is necessary for each motion vector, or both. In this case, it is reasonable to adopt an affine motion model other than block-wise translational motion models. See Figure 13 for an example. The affine motion model has been utilized by the MPEG standards, but only for coding the background sprite. [47] is an effort to introduce an affine motion model for coding motion of VOP's, however, it applies the affine

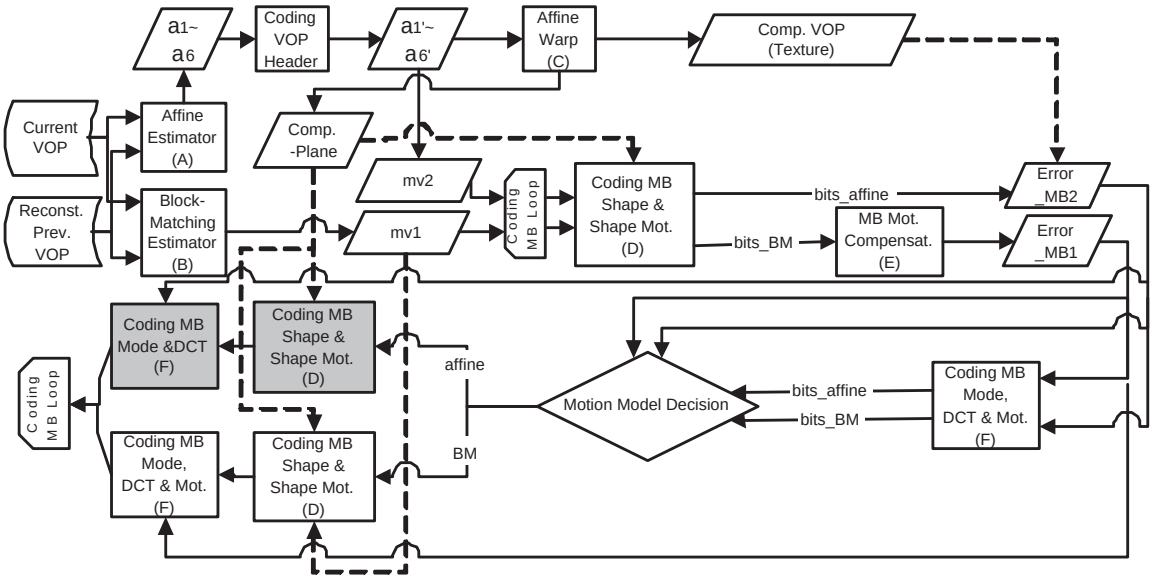


Figure 14: MB-wise multiple motion model encoder structure. Solid lines illustrates process of the program, dashed lines mean data transmission only. Gray blocks are modules that output to real bit stream if affine mode is selected.

motion model to each block instead of to the whole VOP, so it results in an even larger number of motion bits, this is traded off for a gain in motion accuracy in cases of rotation, divergence and shear. In this chapter, we study the efficiency of coding a sequence of VOP's of one single Video Object (VO), with the affine motion model. The factors that will affect the efficiencies are: (1) the number and intensity of the motion parameters and (2) the intensity of the compensation error VOP.

3.2 Proposed Adaptive Coding Scheme Using Affine Motion Models for MPEG P-VOP's

This work is based on the software of MoMuSys-OM-1.0-000706 coder-decoder. The adaptive coding scheme with affine motion models is created by modifications to the momusys system as described in the following section. Figure 14 is a flow chart of the core part of the modified encoder.

3.2.1 Affine Motion Estimator for Each P-VOP

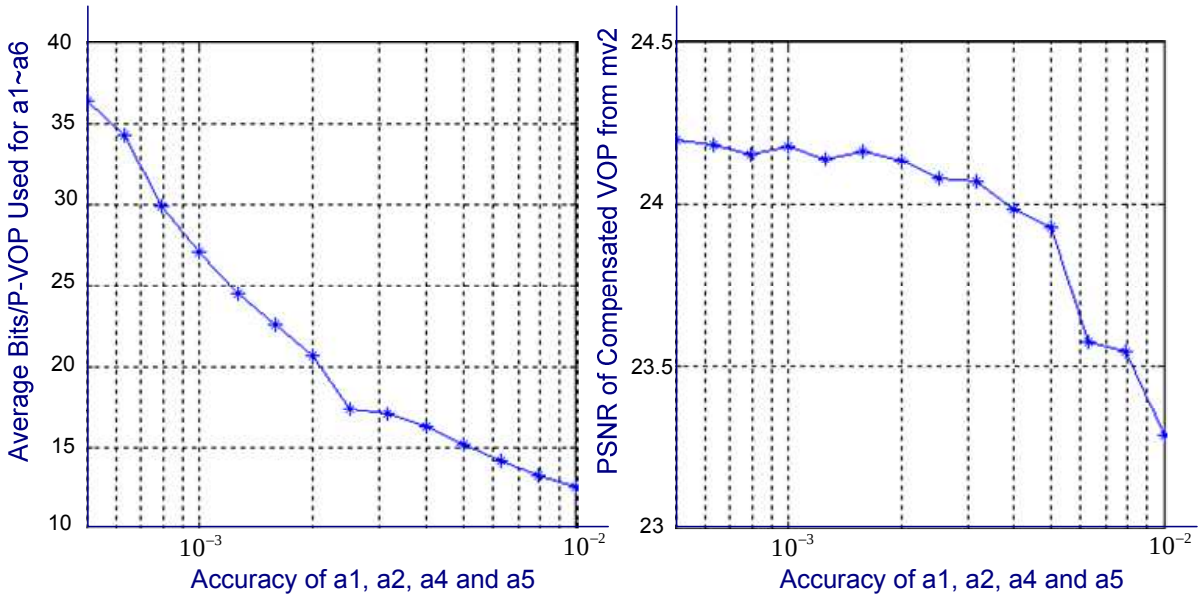
An affine motion estimator ((A) in Figure 14) is implemented as well as the original block-matching searching algorithm ((B) in Figure 14), prior to the MB coding loop. The affine motion estimation program used herein is hierarchical and iterative; it outputs the 6 affine parameters, $\alpha_1 \sim \alpha_6$, which are interpreted in terms of block-wise optical flow vectors (mv_2) by (14). Note that here i and j are indices of the MB, instead of the pixel.

$$\begin{bmatrix} mv2_x \\ mv2_y \end{bmatrix} (i,j) = \begin{bmatrix} a'_1 & a'_2 \\ a'_3 & a'_4 \end{bmatrix} \times \begin{bmatrix} (i-1) \times 16 + 8 \\ (j-1) \times 16 + 8 \end{bmatrix} + \begin{bmatrix} a'_3 \\ a'_6 \end{bmatrix} \tag{14}$$

$\alpha'_1 \sim \alpha'_6$ in (14) are decoded affine parameters from the variable length coder (VLC) approximations of $\alpha_1 \sim \alpha_6$. These values will be used by the decoder for further computation. The only use of $mv2$ is to provide prediction for shape motion estimation. They are neither used for texture compensation, nor coded to the bit-stream, as $mv1$ are in the MB coding loop. The MB loop creates two compensated MB's for each MB to be coded: one from $mv1$ via block-matching; another trimmed from the affine-warped VOP, which is generated before the loop starts.

3.3 VOP Header Modification

The resulting affine parameters $\alpha_1 \sim \alpha_6$ are predicatively coded by VLC to the VOP header. To parallel its MPEG-4 coder counterpart, a step size of 0.5 and a search range of $[-16, +16]$ were chosen for α_3 and α_6, which represent the shift components of the motion. The original VLC for MV difference (MVD) of the MPEG standards uses for α_3 and α_6. The rotation parameters α_1, α_2, α_4 and α_5 are more subtle and need greater accuracy. Assuming that the smaller amounts of motion occur most often, we can generate a VLC for the magnitude of α_1, α_2, α_4 and α_5 (another bit is allocated for the sign) as in Table 1. For the sequence of *bream*, the magnitudes of α_1, α_2, α_4 and α_5 are rarely greater than 0.1, so we set the range to be $[-0.25, +0.25]$.



(a) (b)

Figure 15: Choice of accuracy of affine parameters.

The choice of precision affects the length of the VLC. We experimented withaccuracies uniformly separated on the logarithmic scale of $[0.0005, 0.01]$, to obtain an optimal accuracy, trading off the number of bits used for the affine parameters and the PSNR of the affine warped VOP, as well as the ultimate number of bits required for the whole VOP. In Figure 15, we see that as the accuracy (increases in magnitude), bits used for the affine parameters decrease, the PSNR of the warped VOP does as well. However in the lower range of accuracy, the number of header bits (1) does not change as drastically as in the higher range, while the PSNR (2) drops almost linearly over the whole range. As a result, we choose the 0.001 for the accuracy, which achieves the lowest number of bits for the header, among those with the highest compensation PSNR for the affine estimator.

3.4 Block-wise Motion Model Choice Criteria

The choice between affine and block-matching (BM) motion models is made in the MB coding loop. Coding an MB consists mainly of 2 modules: (1) coding the MB

Table 1: The VLC table for α_1, α_2, α_4 and α_5. The magnitude is in units of accuracy, e.g. 0.0005. The last VLC contains 500 0's in a row.

Variable Length Code	Magnitude of Affine Parameter
1	0
01	1
001	2
0001	3
⋮	⋮
000...0001	500

shape and shape motion (as (D) in Figure 14); (2) coding the MB's motion and corresponding texture compensation DCT (as (F) in Figure 14). Both modules put bits into the bit-stream, which is the final output of the encoder. In order to compare the efficiency of the affine and BM models comprehensively, (1) and (2) are called 3 times. First, both models are for used for both modules (1) and (2). As the output bits are sent to two dummy bit-streams, the lengths of the two bit-streams are used for comparison. After the choice is made, (1) and (2) are called again, using the chosen motion model; this time, the resulting bits are put to the real bit-stream. The choice between affine and BM motion models has to take two factors into consideration: (1) the number of bits required and (2) the reconstructed PSNR. The significance of the bit length comparison is evident: the BM model needs some bits for motion vectors, which are totally saved by the affine model; while the BM compensated MB might be smaller in magnitude than the affine warped MB, and thus result in fewer bits for error DCT. In fact this is the case, because of the BM motion compensation's MSE target. Chances are that the affine model can beat the BM with a pretty large error MB, because of the absence of motion vectors. When bit length is the only concern, the two models' operating regions are divided as in Figure 14(a). However, a large error MB not only implies more bits for the DCT, but also a lower reconstruction PSNR. To prevent too much sacrifice of PSNR in bit saving, we expect the operating regions of the two motion models to be as in Figure 16(b), instead of Figure 16(a).

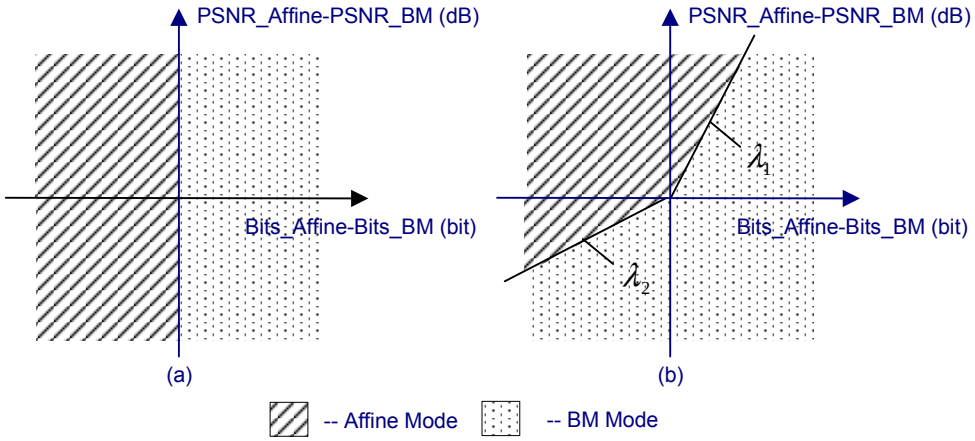


Figure 16: Region division for block-matching and affine motion models, with (a) taking only bit-rate into consideration, (b) selection by rate-distortion optimization.

In the upper left region, the affine model is preferred only when a modest loss of PSNR trades off a significant gain of bit rate. If a mediocre saving of bits is achieved by a gigantic loss of PSNR, then it is not a sensible choice. The ratio is heuristically set to 0.01. The loss of PSNR in the upper left region can eventually be compensated by reducing the DCT quantization step size in rate control when encoded bit rate is lower than the target bit rate. The motion mode selection between the affine and BM models borrows the same philosophy as the mode selection among different block sizes in the MPEG-4/H.263 (INTER8x8 and INTER16x16) and H.264 (INTER16x16, INTER16x8, INTER8x16 and INTER8x8) standards: the optimal coding mode finds a best tradeoff between the motion vector bit-rate and the residual error bit-rate. With the BM estimated motion vectors and the affine warp parameters, the MB modes are decided based on a Lagrangian cost function [29],

$$J_{MODE}(S_k, I_k|Q, \lambda_{MODE}) = D_{REC}(S_k, I_k|Q + \lambda_{MODE} R_{REC}(S_k, I_k|Q)) \qquad (15)$$

where the MB mode I_k is varied over the set of possible MB modes I=INTRA, SKIP, INTER-16x16, INTER-8x8, and AFFINE. (15) is minimized when the optimal rate-distortion combination is achieved. The distortion D_{REC} is measured as the sum of

squared differences between the reconstructed s' and the original frame s

$$D_{REC} = \sum_{(x,y)\in A} |s[x,y,t] - s'[x,y,t]|^2 \tag{16}$$

where A is the MB to be coded. The rate R_{REC} is the sum of the number bits for syntax, motion vectors, residual errors and shape information. The coefficient $\lambda_{MODE}$is chosen according to [29] as

$$\lambda_{MODE} = 0.85 \times Q^2 \tag{17}$$

where Q is the quantization step size for the current frame.

3.5 Complexity Analysis

The computational load of the proposed coding scheme consists of (1)affine model estimation (the warped VOP is generated as part of this calculation) and (2) calculation of the affine mode cost in BM mode selection. The temporal gradient I_t used in the determination of the affine model is obtained by simple pixel substraction with a cost of approximately $6 \times 9 \times 16 \times 16 = 13824$ additions, in the case of the frame0 of *bream.qcif*, whose VO is defined on a 6×9 rectangle. I_x and I_y require 12 additions and 7 multiplications for filtering and subtraction with a 9×9 kernel at each pixel. This results in a total cost for determining the spatial gradients is $12 \times 13824 \times 2$ additions and $7 \times 13824 \times 2$ multiplications. Calculating $x \cdot I_x$, $x \cdot I_y$, $y \cdot I_x$ and $y \cdot I_y$ costs 4 multiplications and filling the elements above the diagonal of H requires an additional 21 multiplications at each pixel. The load for constructing H with the given gradients is thus 25×13824 multiplications and 21×13823 additions. Likewise, B requires 6×13824 multiplications and 6×13823 additions. Adding the 301 multiplications and 250 additions to solve a 6×6 set of linear equations, using the 3-layer structure and assuming an average of 3 iterations per layer, the overall complexity for affine motion estimation is about $2.8M$ multiplications and $2.5M$ additions, per frame. For a half-pixel MSE BM scheme, with a search range of $[-16, 16]$, the worst case requires

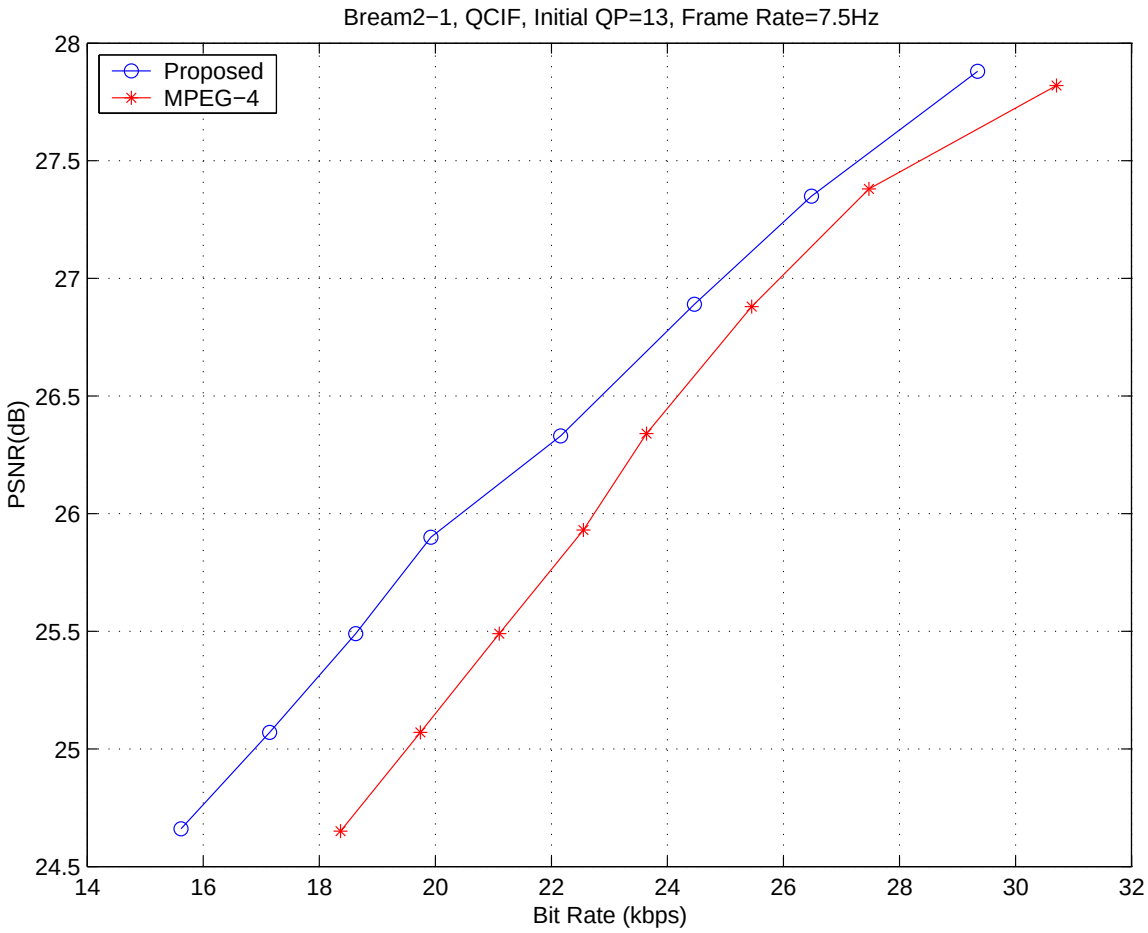


Figure 17: Rate-PSNR curves of BREAM sequence encoded with MPEG-4 and proposed encoders.

$(16 \times 16 + 16 \times 16 - 1) \times 64 \times 64$ additions and $16 \times 16 \times 64 \times 64$ multiplications to find the motion vector for one of the 6×9 MB's and $4 \times (8 \times 8 + 8 \times 8 - 1) \times 64 \times 64$ additions and $4 \times 8 \times 8 \times 64 \times 64$ multiplications for the MV's for the corresponding 4 blocks. This comes to $225M$ additions and $113M$ multiplications per frame. The computational load for the global affine motion estimation is approximately 1% of the full search BM case, and is comparable to most of the realistic partial search cases. The computational complexity for the affine mode in mode selection is comparable to that for the BM mode.

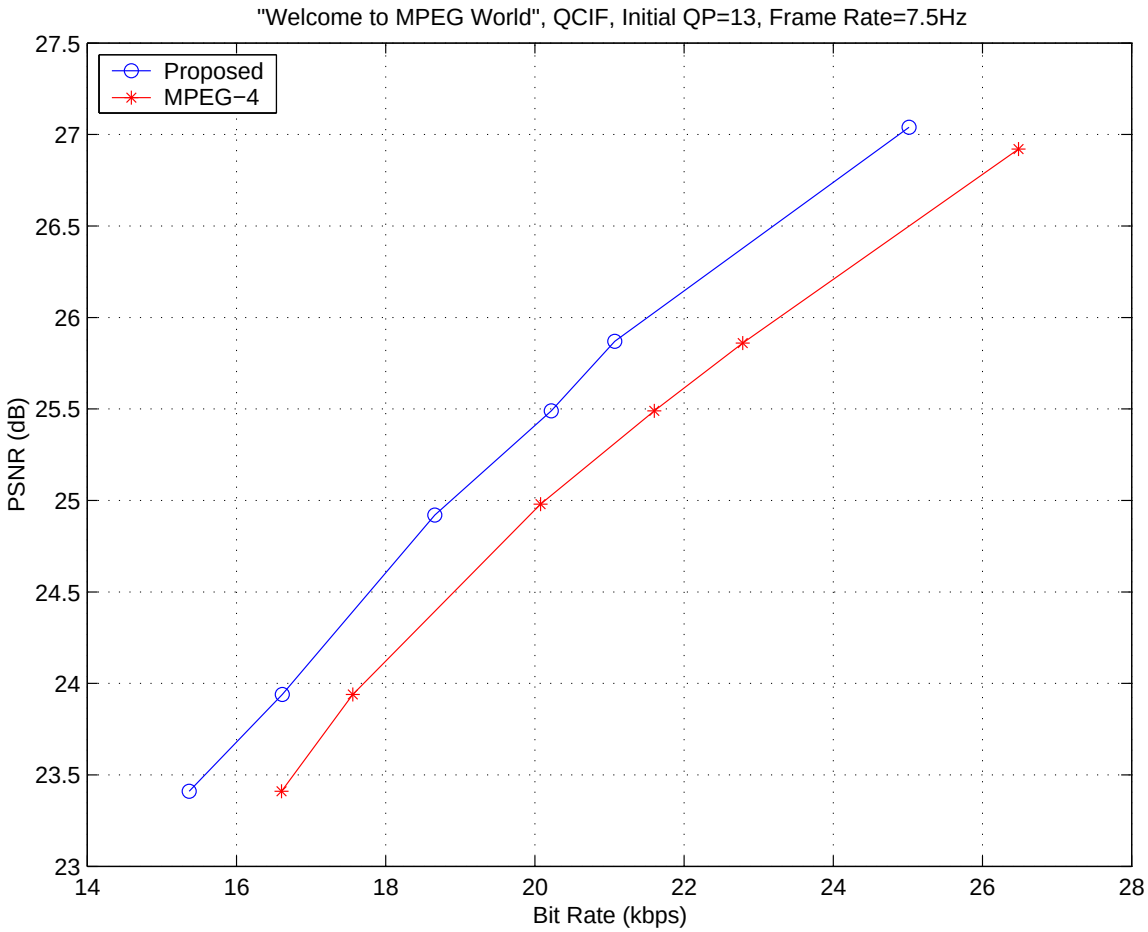


Figure 18: Rate-PSNR curves of "WELCOME TO MPEG4 WORLD" sequence encoded with MPEG-4 and proposed encoders.

3.6 Simulation Results

Simulations we carried out on the sequences *bream2_1.qcif* and *bream2_2.qcif*, under various target bit rates, with the original Momusys codec and the proposed multiple motion model code. We observe a notable decrease in the number of bits used for an average P-frame, especially when the transmission bit rate is very low (Figures 17 and 18). The system's relative improvement at low bit rates can be explained by the observation that affine model, in most cases, loses to the BM model in terms of compensation error, as a tradeoff for its compact motion representation. Also, due to BM model's MSE optimization target, even when it doesn't provide as accurate a motion vector as the affine model, it still often generates a smaller residual error. When the target bit rate for video communication is low, the quantization of the DCT is forced to be crude, and the subtle difference of the error block produced by the two motion models is diminished. In all cases, 50% to 70% of the inter-coded MB choose the affine mode over other modes.

The proposed codec can decrease the number of bits used for an average P-frame by 10% to 20% for a range of data transmission bit rates, compared to the existing MPEG codec. The practical significance of this system is that it requires only moderate modification to the standard, to achieve its coding gain, and therefore may well be added to an original MPEG-4 codec. Due to its low bit-rate adaptability, the proposed scheme is best utilized in video conference scenarios. Although simulation was carried out on sequences with pre-defined VOP's, the alpha channel is not necessarily a pre-requisite for using the affine model. Application of the proposed coding scheme for more general sequences is introduced in the next chapter.

CHAPTER IV

MULTIPLE GLOBAL AFFINE MODELS FOR GENERAL VIDEO FRAMES

4.1 Introduction

In this chapter, we propose to use multiple global affine motion models on a natural video frame, in place of the local block motion vectors, in lower transmission bandwidth scenarios.

In the latest video coding standard H.264, motion compensation partitions have been reduced to allow for blocks as small as 4×4. For a qcif (176×144) P-frame, assuming all MB's are coded as inter MB's, then motion estimation can generate between $11 \times 9 \times 2 = 198$ (in the case of all $INTER16 \times 16$ modes) to $44 \times 36 \times 2 = 3168$ (in the case of all $INTER4 \times 4$ modes) motion parameters. On the other hand, with affine models for each motion object (MO) used, only $N \times 6$ parameters are needed, where N stands for the number of MO's in the frame. In many applications, e.g. video conferencing, there are only two MO's, the background and the foreground. With $N = 2$, only 12 motion parameters are transmitted in the bitstream.

Meanwhile, in low bit rate network scenarios, a hybrid video coder (e.g. AVC/H.264) tends to allocate a greater portion of the bit budget for motion vectors, while saving bits on residual errors (Figure 2). With the objective of reducing the bit budget for motion information at lower end bitrates, we propose to use the global motion model in place of local block motion vectors, when the available bandwidth for video communication is lower than 100 kbps.

To detect multiple MO's in a frame, motion segmentation needs to be performed while the affine motion model for each MO is being computed. While pixel-wise

segmentation can offer the best segmentation accuracy, it is computationally expensive and not necessary. Instead, we perform the segmentation and estimation on a block base, to be compatible to the existing codec structure. This is sufficient for our accuracy needs.

The scaling portion of the affine model is coded with a $4-$dimensional vector-quantizer (VQ), the translational part with a classical motion vector. Both parts are transmitted in the frame header. Multiple new macroblock (MB) modes, associated with the affine motion models, are added to the original list of $I4$, $I16$, $P16 \times 16$, $P16 \times 8$, $P8 \times 16$, $P8 \times 8$ and $DIRECT$ etc. of the standard. An MB chooses one of the affine modes, if the corresponding MO affine motion model results in the lowest Lagrange cost.

This chapter is organized as follows. Section 4.2 introduces general backgrounds for motion segmentation, as well as the specific affine motion estimation process implemented in the proposed codec. Section 4.3 covers the vector quantizer used to quantize the affine model and the cache-memory design to accelerate the dictionary search. Section 4.4 discusses the effect of the global motion model on blocking artifacts and introduces a new subjective reconstruction quality measure, which favors the affine modes more than the traditional PSNR. Section 4.5 is a description of the mode selection module, which modifies the existing Lagrange optimizer with the new affine modes, as well as the new perceptual PSNR distortion metric. Section 4.6 presents results of simulations on coding efficiency, and subjective reconstruction quality.

4.2 Motion Segmentation

4.2.1 Motion Objects

4.2.1.1 Motion Object and Motion Boundary

A motion object refers to a set of pixels with the same coherent motion. It is more of a practical concept than a mathematical definition, for it is difficult to be defined

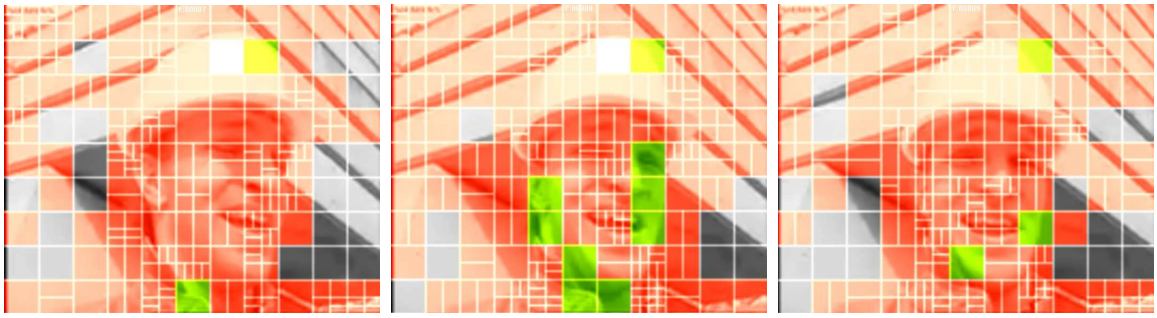
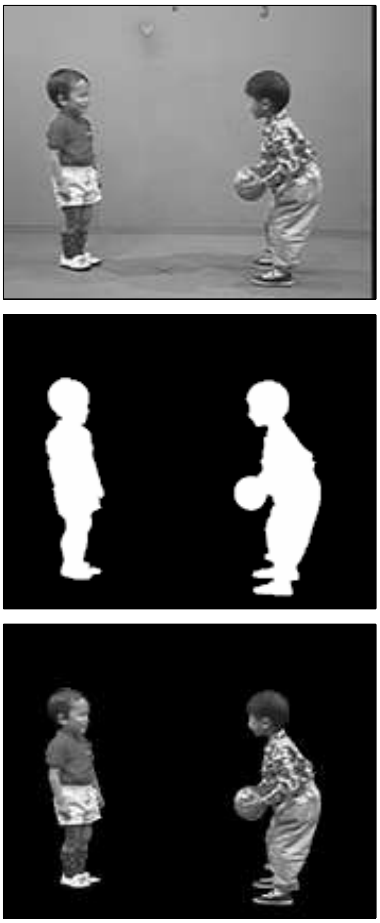


Figure 19: Distribution of intra MB's over the P-frames (frame 5-8), with yellow MB's as $INTRA4 \times 4$, green as $INTRA16 \times 16$, gray as $SKIP$ and orange as $INTER$. The intra MB's are mostly distributed around the human face, where occlusion happens frequently.

precisely. Realistically it could be a rolling ball, or a panning background, but it should not be a dancing human. The margin areas at the boundary of two or more motion objects are called motion boundaries, which may suffer from occlusion and uncovering and hence be elusive for motion compensation. The residual error for a motion boundary block is often notably higher than for an internal block of a motion object. When the intra-mode is selected in an inter-coded slice, it is often on the motion boundary. See Figure 19 for an example.

4.2.1.2 Video Object Plane

A Video Object Plane (VOP) is a concept first defined by MPEG-4, as a feature of object-based video coding [1]. It is introduced for the sake of multiple video object manipulation, rather than compression enhancement. A VOP can contain multiple motion objects, such as the two children and one ball in the standard sequence *children*. A VOP can (such as sequence bream) or can not (such as in the sequence *children* in Figure 20) be a motion object. It is after all a subjective definition. The support of a VOP is represented by a gray-scale or binary map of the same size as the luminance frame. This is called the alpha-channel; it is transmitted as a fourth channel together with the one luminance and two chrominance channels. A gray-scale alpha-map defines the transparency of each pixel, with 0 as transparent and 255 as



(a)

(b)

(c)

Figure 20: An example of a VOP layer that's not one motion object.(a) is the original scene for the VOP. (b) is the alpha channel for the VOP. (c) is the VOP masked by its alpha channel, ready to be synthesized with other VOP's for a new scene, as in Figure 13.f. The difference between the MO in Figure 13.a and the one in this figure is that the first one is a rigid MO, while the latter is not.

opaque. A binary alpha-map is the same, except that it omits the intermediate values. The alpha-channel can also be coded in intra or inter mode, with the assistance of arithmetic coding.

4.2.2 Motion Segmentation Problem Formulation

Motion segmentation groups similar pixels based on their motion consistency, given two consecutive frames of the same scene under motion. For example, if the scene consists of two different cars and the background, motion segmentation should output

three segments to represent the background environment, and the two cars respectively, based on their different movements [43]. In other words, the task of motion segmentation is to cluster all pixels of a video frame into partitions as close as to the real moving objects as possible. Since the objective is to determine the motion coherence of pixels, a pixel-wise optical flow field is needed as an input to the motion segmenter. This way, the task of grouping pixels is converted into the mission of grouping the similar motion parameters associated with the pixels.

4.2.3 Flow-Based Motion Segmentation Methods

4.2.3.1 Bayesian Expectation-Maximization (EM)

Assuming that the number of motion objects is known, the Bayesian method models the segmentation problem as a missing data problem and provides a theoretically elegant framework for the solution. For a given observation O, EM determines the segmentation X that maximizes $P(X|O)$, often referred to as the maximum a posteriori (MAP) estimation. In [44] the pre-computed optical flow is used as the observation O, the segmentation X is modeled as a Markov random field. With the knowledge of each pixel's segment identity, the maximum likelihood (ML) estimation can easily predict the probability density of each segment; on the other hand, based on MAP, the segmentation map can be estimated from the density parameters [43]. Thus the EM approach iterates between the estimation (E-step) and maximization (M-step), until it converges. Limitations of the EM methods include the requirement that the number of segments be known, which can practically be dealt with by cross-validation procedure. It also requires considerable computation when the dimension of X is high.

4.2.3.2 Clustering

K-means clustering vector quantization is an intuitive and is also the most popular method of motion segmentation. Though numerous implementations of K-means

motion segmentation have been proposed by researchers, the basic algorithm consists of the following steps.

First, an initial segmentation map is determined. This initialization can be random or preset. Yang et al proposed a motion segmentation scheme with adaptive number of segments [45], in which the initial number of segments is set to a large upper limit, e.g. 30, and after each round of clustering, any two segments with a distance below a predefined threshold is merged, reducing the number of segments by one.

Figure 21 demonstrates an example in which (a) and (b) are preset initial segmentation maps, which lead to fairly good convergence results;(f) is the preset color segmentation map, which converges to a more meaningful and neat segmentation [46]. In either case, the number of segments is preset to 2 in Figure 21.

Second, the affine model for each segment is computed from the MV's within that segment by (13). The MV's could be pixel-wise or block-wise, depending on the accuracy and computation load requirements.

Third, the MV's are reassigned to one of the N affine models, to minimize either the 2-D motion vector distortion as in (18) or compensated intensity distortion as in (19).

$$D_M = \sum_{(i,j)\in S_k} \|mv(i,j) - mv(A_k;(i,j))\|^2 \tag{18}$$

$$D_I = \sum_{(i,j)\in S_k} \|I'(i-mv_x(i,j), j-mv_y(i,j)) - I'((1-a_{1,k})\cdot i - a_{2,k}\cdot j - a_{3,k}, i+(1-a_{5,k})\cdot j - a_{6,k})\|^2 \tag{19}$$

While (19) provides a more physically meaningful criterion, (19) is more computationally efficient and often leads to a comparable result with (18).

Fourth, as in any classical K-means iteration, the cluster center, interpreted as the affine model, is updated. Instead of taking the centroid as the center, the motion segmentation process obtains the new center by (13) as in step one.

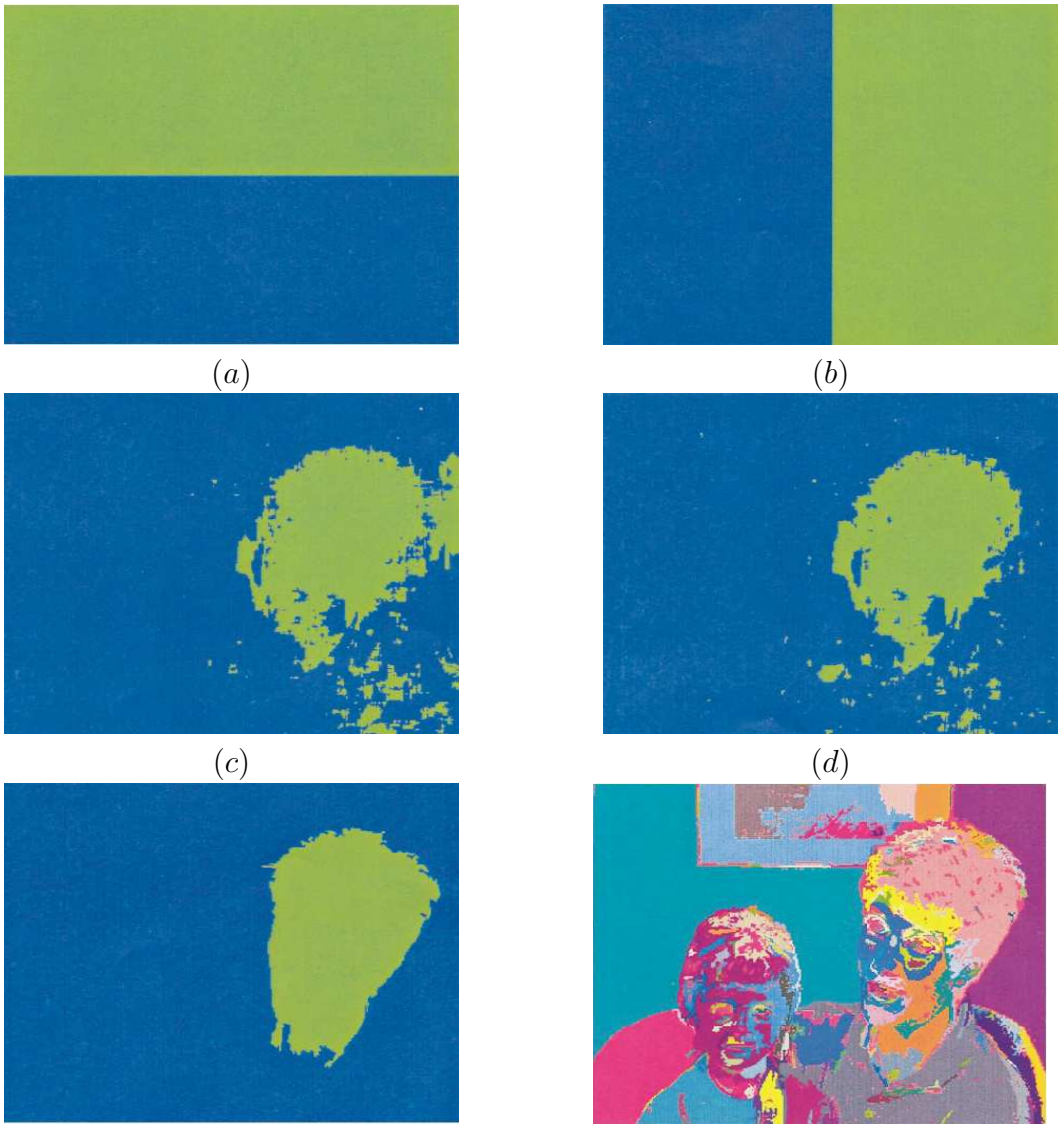


Figure 21: Effect of different initial segments on motion segmentation results for the standard sequence of "mother and daughter", from [46]. (a), (b) are random initial guess of the segmentation map. (f) is the region map obtained from color segmentation, which could be used as an initial map as well. (c) and (d) are intermediate results after 30 segmentation iterations. (e) is the final motion segmentation map.

Steps three and four are iterated until the number of pixels that change their segmentation assignment is below a threshold. In addition to this basic iteration structure, a motion segmentation process can be equipped with various functionalities to facilitate the accuracy and speed of the segmentation task, such as filtering small segments to enforce spatial continuity.

4.2.4 Motion Estimation in the Multiple Affine Model Video CODEC

In this section, the proposed novel hybrid video encoder, which models the frame motion information using the affine motion model for multiple motion objects in the frame, together with conventional block based motion vectors is discussed. Figure 23 is a block diagram of the proposed encoder. The proposed codec is constructed upon the JM 7.1 codec.

4.2.4.1 Block Matching Motion Estimation

In Figure 23, the block labelled "BM Estimator 1" is the JM block-matching motion estimator. It estimates quarter-pixel resolution motion vectors for all of the block sizes allowed in H.264, for one MB. "BM Estimator 0" is similiar to "BM Estimator 1", except that "BM Estimator 0" uses the original previous frame as its reference, in order model the optical flow as accurately as possible; whereas "BM estimator 1" uses the previous reconstructed frame, in order to mimic the behavior of the decoder. For the same reason, after the affine models have been calculated, the reconstructed previous frames are used to predict the current frame. Although BM estimators 0 and 1 share the same search range (e.g. 16×16) and step size (e.g. 4×4), their matching window sizes are significantly different (e.g. 32×32 for estimator 0 vs. 4×4 for estimator 1). This is to avoid pseudo motion vectors in areas of uniform texture areas.

As discussed in section 4.2.3.2, the MV's that are inputs to the segmentation module can either be pixel-wise or block-wise. We use a motion field of 4×4 block

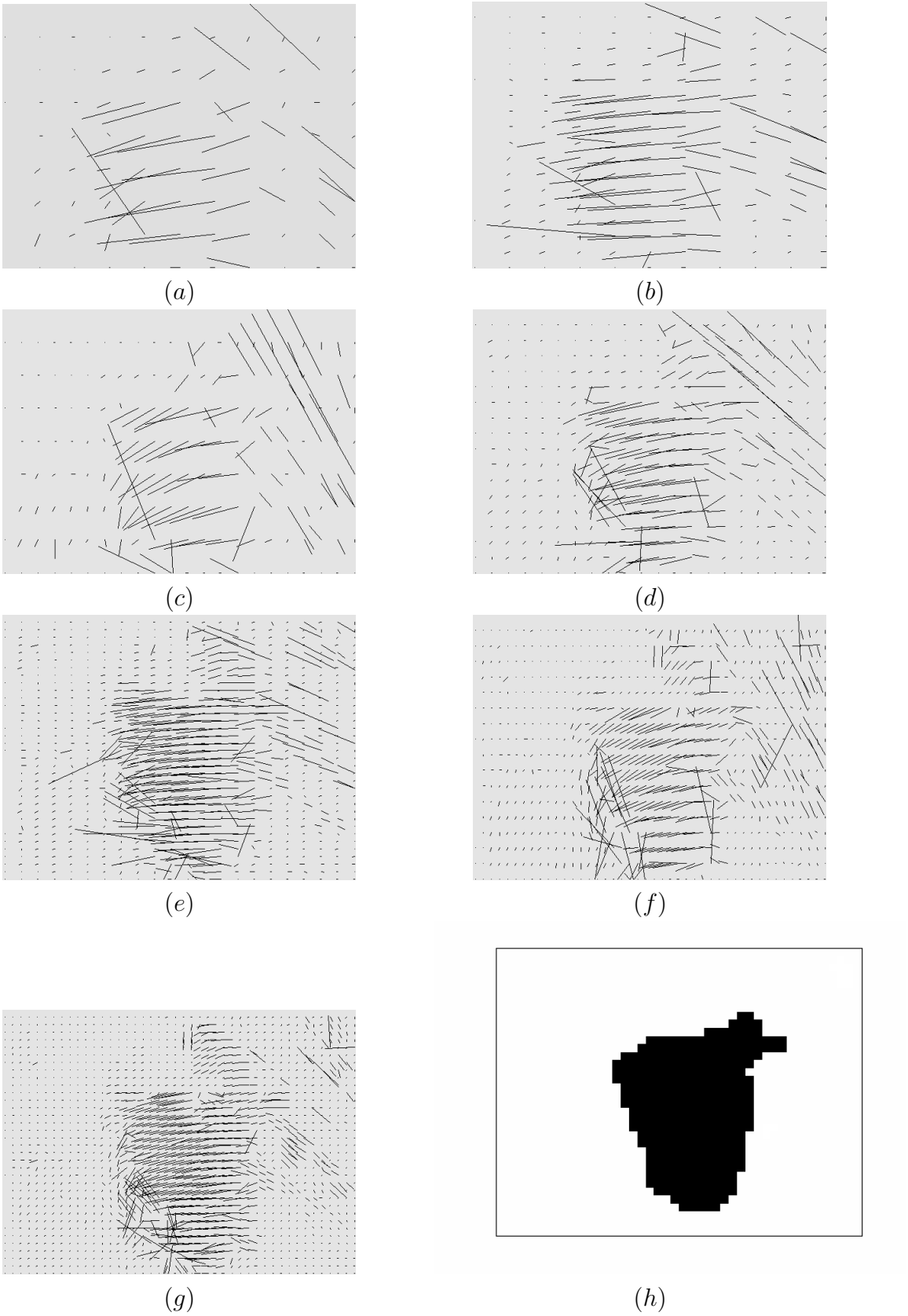


Figure 22: Block-matching motion fields of variable sizes ((a)-16 × 16, (b)-16 × 8, (c)-8 × 16, (d)-8 × 8, (e)-8 × 4, (f)-4 × 8, (g)-4 × 4) generated by the original JM encoder, and the motion segmentation map (h) generated from the 4 × 4 one, on foreman.QCIF frame 0 and 1.

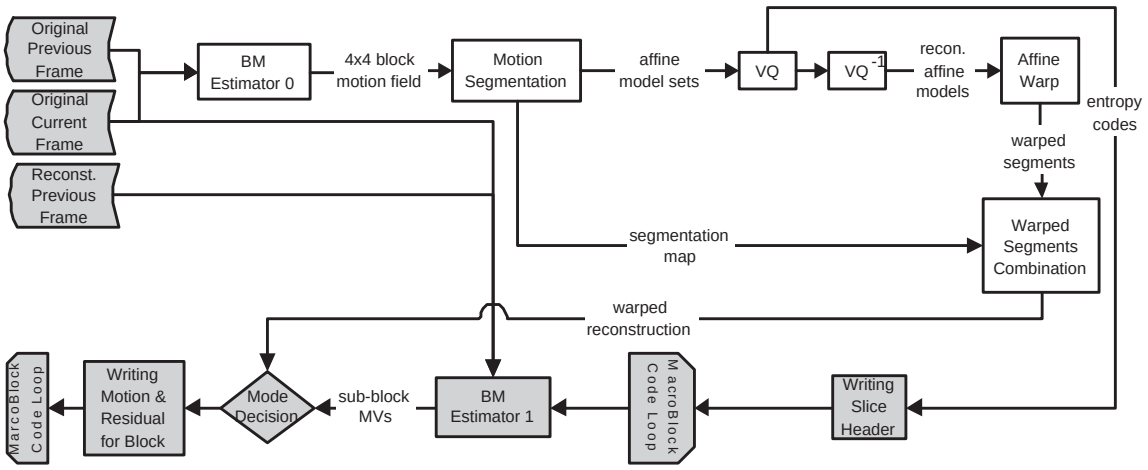


Figure 23: System block diagram of the proposed hybrid video encoder. The gray parts are the originals of the JM encoder.

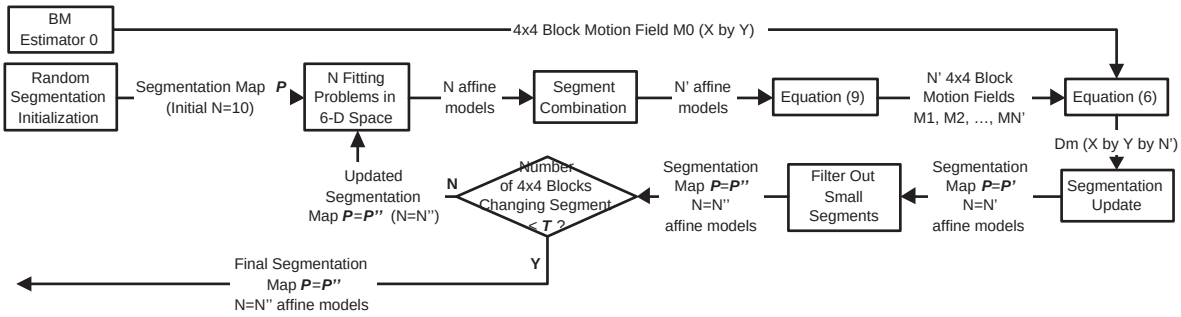


Figure 24: Flow chart of the "Motion Segmentation" module in Figure 23. This simultaneously performs segmentation and affine model estimation.

MV's, because (i) pixel-wise motion estimation is quite costly and not necessary, since the selection of the affine mode is made at the MB level; (ii) MB-wise motion fields are too crude for motion segmentation. The inclusion of too many points outside the motion object will affect the accuracy of the estimated affine model, given the fact that the affine motion estimation and the motion segmentation are coupled.

4.2.4.2 Motion Segmentation and Affine Model Estimation

The motion segmentation and affine model estimation processes are coupled, and both are performed by the subsystem labelled "Motion Segmentation" in Figure 23. For a segmentation task that ultimately aims at the construction of video objects as

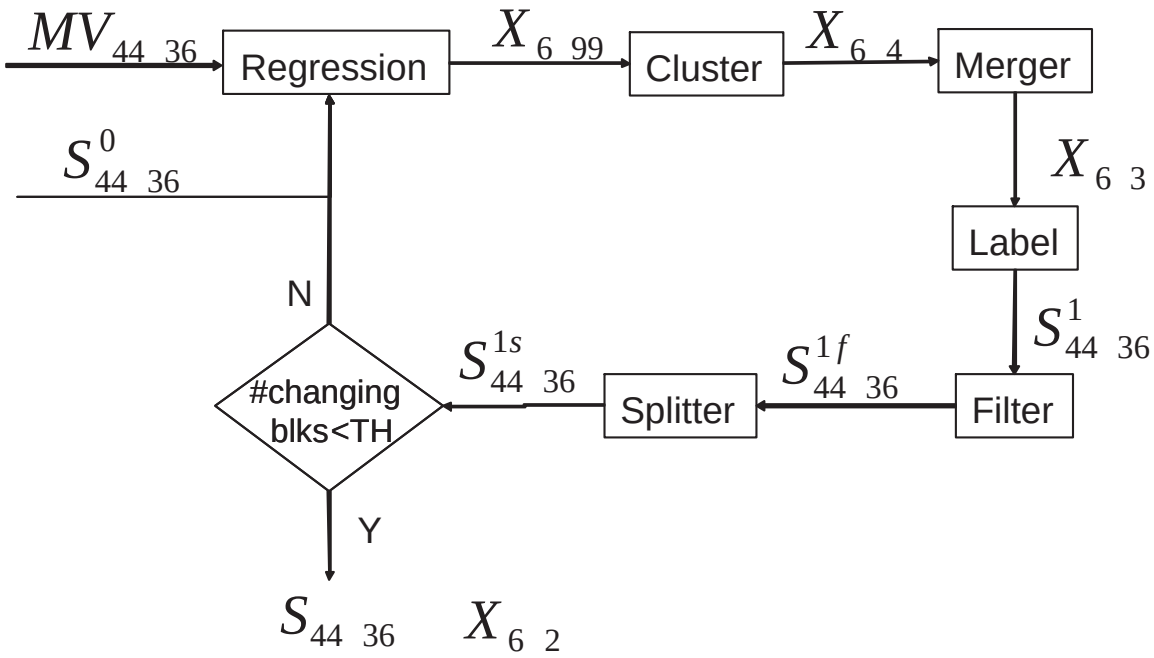


Figure 25: Block diagram for motion segmentation, with an initial random block-wise segmentation map (refer to Figure 26) and the motion field with the same block size base as the input, and the final segmentation map, as well as the affine models for each segment, as the output. $X_{6\times N}$ stands for the affine model matrix, where N is the number of models. $S_{44\times 36}$ is the label map for the 44×36 4×4 blocks, ranging from 1 to N.

in [46]; an accurate pixel-wise segmentation map is useful. To generate a pixel-wise segmentation map $\boldsymbol{P}$, however, for a QCIF frame, requires $176 \times 144 \times N$ labeling operations after the block-wise motion segmentation procedure. In addition, the affine model fitting equation of eq.(13) would require matrix A to be of size $2 \times 176 \times 144$ by 2. However, our experiments show that a block-wise final $\boldsymbol{P}$ (see Figure 24) achieves the same number of affine mode MB's and approximately the same final coding efficiency as a pixel-wise final $\boldsymbol{P}$. So we use the block-wise final P as shown in Figure 24, to reduce the computation. See Figure 22 for an example.

In addition to the clustering routines needed for motion segmentation, several other features are inculded in the segmentation module, to improve performance [44]. These are shown in (Figure 25).

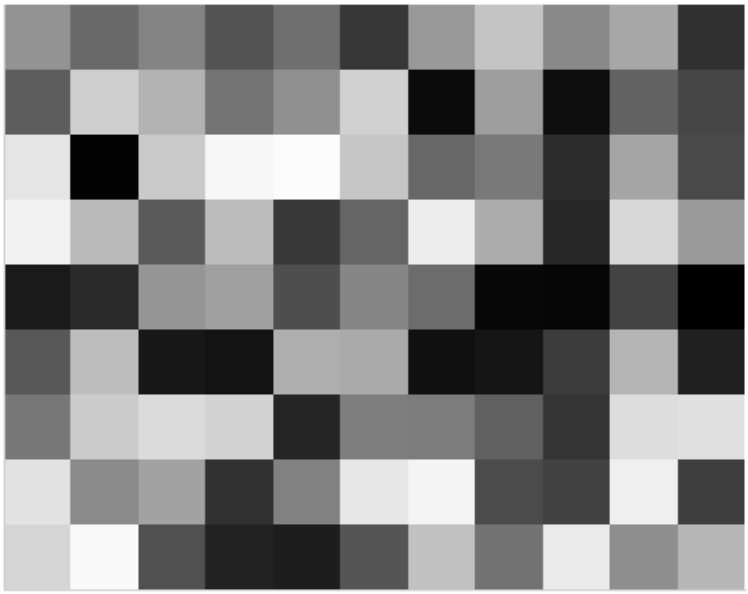


Figure 26: Initial random guess of the segmentation map, on a 16×16 base, for a QCIF frame.

To accommodate the initial assumption that each MB in a QCIF frame has a unique affine model, the regression parameter N is set to 99. After the regression process in Figure 25 exits with N affine models, each of the M initial 4×4 blocks ($M = 44 \times 36\times$ in Figure 26) has a label ranging from 1 to N. Clustering group these 99 classes in the 6-dimensional affine parameter space into a preset small number (heuristically 4) of clusters. Since the ultimate goal of the coder is to reduce the number of motion bits, generating too many affine models would add too much overhead for the Rate-Distortion performance to improve. Thus, the cluster block is necessary to limit the number of affine models to be transmitted. As N is reduced to 4, the M segment units are re-labeled with labels 1 to 4.

The clustering process presets the number of output segments to 4, but one frame doesn't necessarily contain 4 motion objects. In this case, some of the 4 segments that essentially belong to one MO will have very close affine model in the $4-D$ space.

They are merged by the "Merger" in Figure 25, and N is further reduced to, say, 3.

To enforce the local spatial connectivity of the motion regions, additional constraints need to be applied before the N models are re-estimated in the loop. Because the affine motion classification is performed at each small 4×4 block in the frame with models that apply to a much larger region, one model may coincidentally fit a small block in another region undergoing a different coherent motion. Also, since block-matching is a method that targets at the best match instead of the true motion, some small blocks may be associated with wrong motion vector and then labeled erroneously. As a result, the affine model may support sporadic 4×4 blocks that are disjoint from each other. The serial modules "Splitter" and "Filter" identify these blocks and filter them out. When several disjoint regions are supported by a single model, the region splitter separates them into independent regions and produces a new model for each. Then the "Filter" eliminates the small regions to keep the estimator stable. Finally eq.(13) creates an affine model for each reliably connected region. The ultimate output of the segmentation process is (i) the segmentation map $S_{44\times36}$ with values from 1 to N ($N = 2$ in the foreman frame example) and (ii) the N (e.g. $N = 2$) affine models $X_{6\times2}$.

Figure 27 presents an example of the segmentation results. Figure 28 shows the result of applying the segmentation map on the reference and the current frame. The four images in Figure 28 are used for affine model estimation in (10). As can been seen from the estimated affine models X_0 and X_1, The foreground motion X_0 is drastic because the head is turning, while the background reflects slight shift caused by the camera shaking. Figure 29 demonstrates how the warped MO's are synthesized into a complete motion compensated frame.

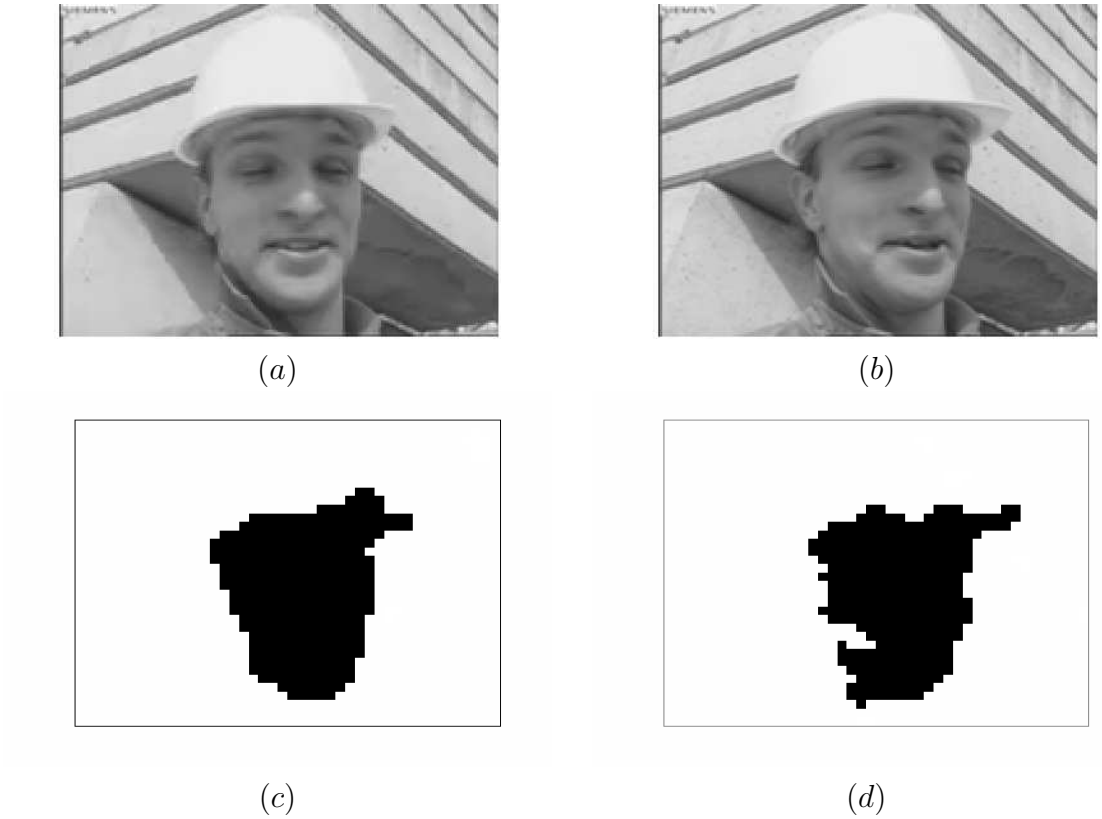


Figure 27: Example of segmentation results. (a) and (b) are the originals of frame 1 and frame 2 of foreman.QCIF; (c) and (d) are the binary segmentation map as a result of the segmentation process described in section 4.2.4.2.

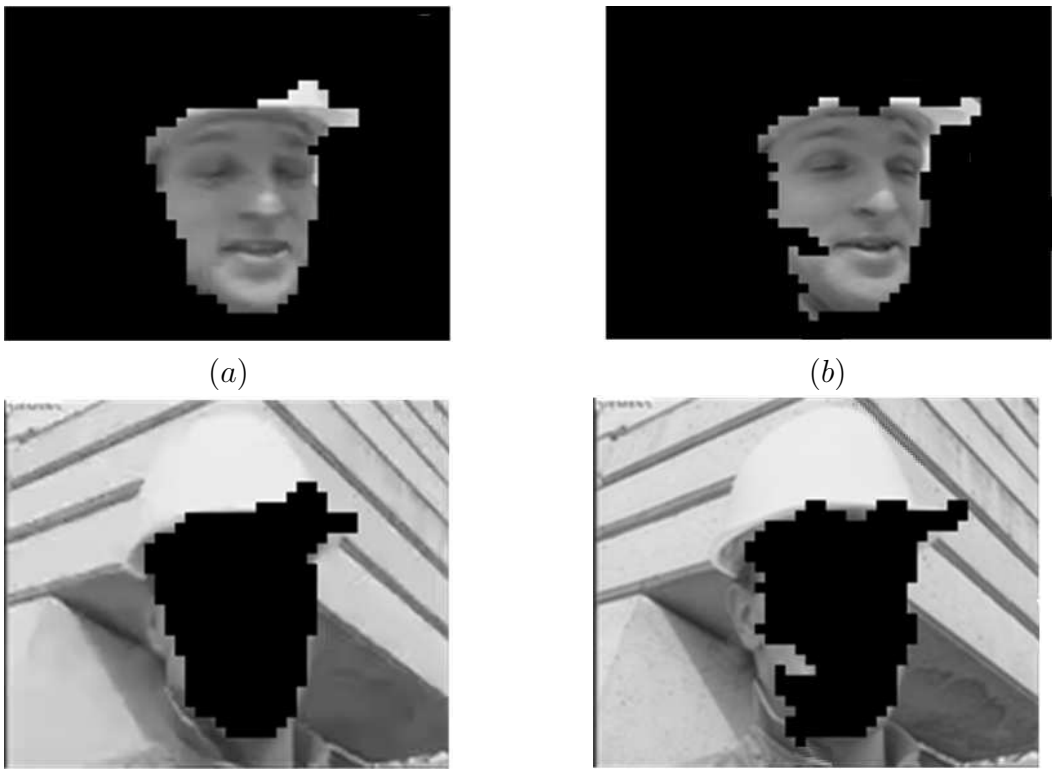


Figure 28: Figure 27.a is divided into 2 motion objects: foreground (a) and background (c), using the segmentation map obtained in Figure 27.c. Figure 27.b is divided into 2 motion objects: foreground (b) and background (d), using the segmentation map obtained in Figure 27.d. (a) and (b) are then used on (10) to estimate the affine motion model for the foreground object, while (c) and (d) are used for the background. $X_0 = [0.0080, -0.0228, -0.0030, -0.0096, 1.8517, -4.0776]^T$ and $X_1 = [-0.0013, -0.0020, 0.0010, -0.0139, 0.3334, -0.1161]^T$ are obtained for the foreground and background respectively.

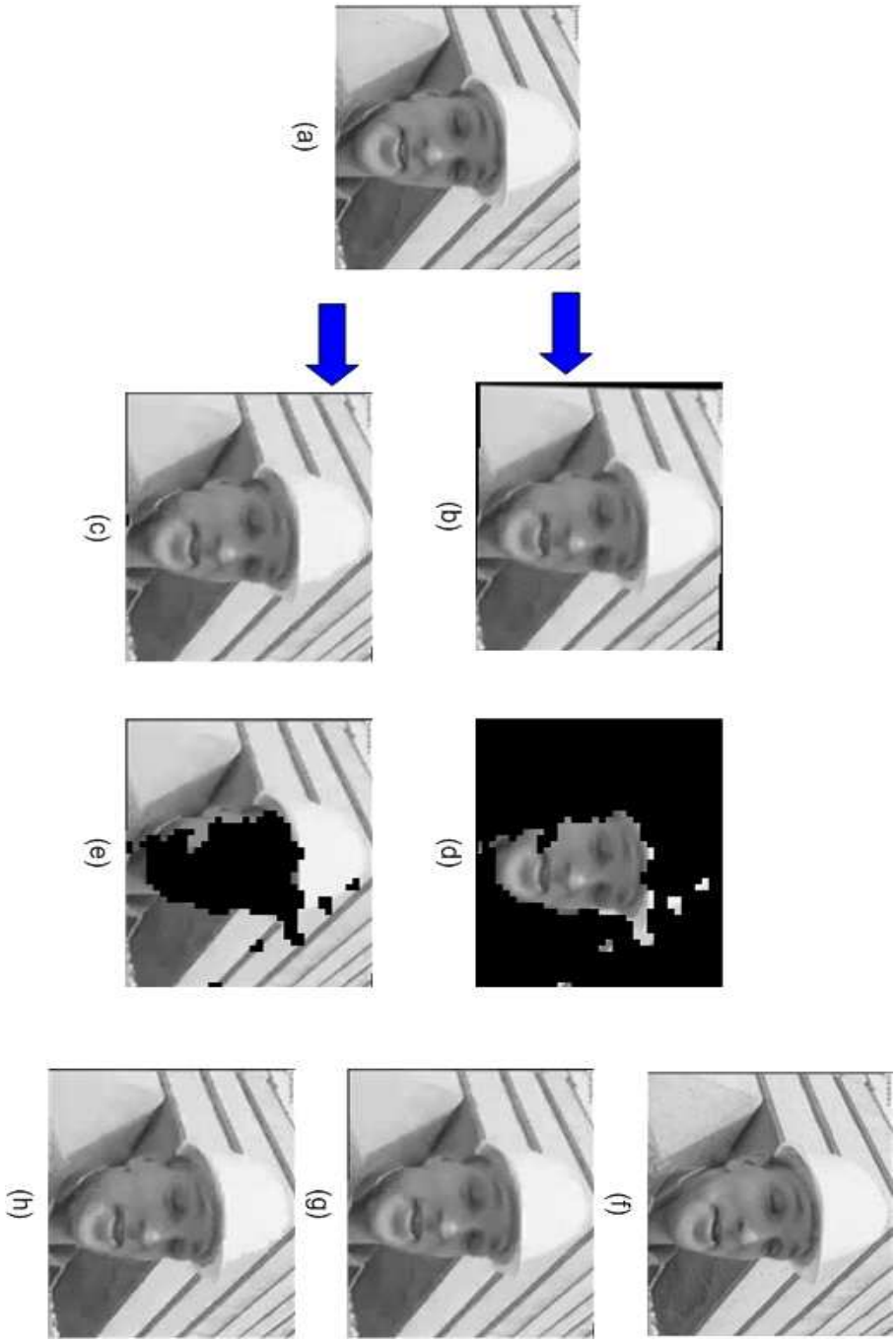


Figure 29: Synthesis of warped motion objects into a complete compensated frame. (a) is the reference frame, i.e. frame 1 of foreman.QCIF; (b) and (c) are (a) warped by X_0 and X_1 respectively; (d) is (b) masked by segmentation map Figure 27.d; (e) is (c) masked by the complimentary map. Adding (d) and (e) produces (g) the final affinely reconstructed image. For comparison, (f), the original of frame 2, and (h), the block-matched frame are listed. It can be observed that (h) resembles the original more on a rough scale and particularly on the face area, while (g) offers an equally good reconstruction on the rest of the frame and a more smooth and natural reconstruction on the face area.

4.2.4.3 Adaptive Number of Motion Objects

Since the number of motion objects, N, is usually no more than 3, the extra bits caused by the increase of mode indication complexity are trivial. When there are more than 3 MO's, too many affine models will result in too much overhead in the slice header, which contradicts the purpose of saving motion bits.

On the other hand, when local motion in a frame is drastic, the global motion model will fail. To avoid wasting bit-rate on the affine motion models in such cases, a metric S is obtained in the motion segmentation phase to measure the locality of the motion. If S is above a predefined threshold, then the affine motion models are skipped in construction of the slice header and the L global modes are all disabled.

4.3 Compression of the Affine Models

4.3.1 Quantization of the 6-D Affine Models

As discussed in Section 2.3.2.6, among the 6 parameters that constitute an affine model, a_1, a_2, a_4 and a_5 are the scaling factors while a_3 and a_6 are the shift factors. In order to compress the 6-D model most efficiently, we quantize $[a_1, a_2, a_4, a_5]^T$ as one vector A and $[a_3, a_6]^T$ as another, b. The 4-D A is quantized by the vector quantizer introduced in the following sections, while the 2-D b is quantized and coded as a regular motion vector. Both A and b are compressed and put in the slice header as in Figure 30. Meanwhile, for those MB's that are still coded using traditional inter modes, e.g. $INTER16 \times 16$, $INTER4 \times 4$, etc., the motion vectors are coded and put in the MB data packet.

4.3.2 Vector Quantizer

4.3.2.1 Quantization

Quantization is a vital step in any digital signal processing task. It represents an infinite number of values on a continuous/discrete range with a finite number of indices, with some acceptable loss of precision. Quantizers can be divided into two

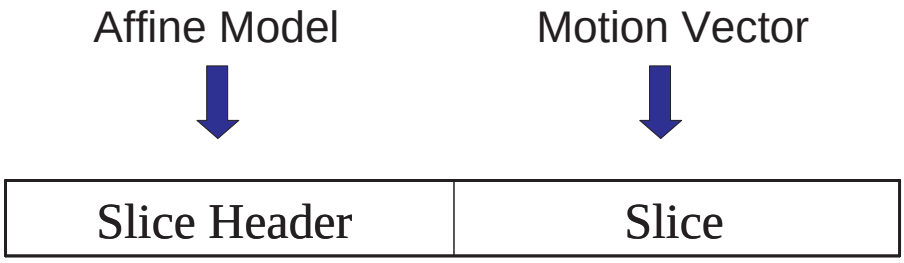


Figure 30: Position of the $6-D$ affine models in the bitstream.

groups: scalar quantizers and vector quantizers, each of which works efficiently in different applications.

4.3.2.2 Scalar Quantizer

A scalar quantizer divides a range into N steps, each represented an index. An input signal is labeled by the index of the step into which it falls. In de-quantization, the recovered value of the signal is the central value of the indexed step; the expected deviation from the original is determined by the width of the step.

A scalar quantizer can be uniform, in which all the steps have the same width, as well as non-uniform, in which they don't. Non-uniform quantizers are usually designed to minimize the de-quantized distortion for a given number of steps (e.g. a Lloyd-Max quantizer), or to minimize the distortion with an upper limit on the number of bits needed to represent the steps (entropy-constrained quantizer). Experiments show that in most cases, a simple uniform quantizer followed by an entropy coder performs similiarly to an entropy-constrained quantizer [12].

4.3.2.3 Basic Concept of Vector Quantizer

The basic idea behind a vector quantizer (VQ) is to divide a vector space into regions (analogous to the steps in a scalar quantizer), and replace any vector that falls within a region with the centroid. The centroids are often called codewords, that are of the region into which it falls. This is analogous to the step centers in a scalar quantizer. In order to define the regions, a finite number of vectors that are representative of

the statistical characteristics of the real data (referred to as the training set) are clustered using a clustering algorithm and an optimization criterion. The most well-known clustering algorithm is called the Linde-Buzo-Gray (LBG) algorithm or the generalized Lloyd algorithm[38], which starts with an initial selection of codewords and iterates between clustering the vectors in the training set based on the codebook and updating the code-book. It can fail because of a poor initial guess, and suffers computationally from the number of searching operations in the assigning phase. To address these problems, various fast approaches with a minor sacrifice of performance and significant gains in matching speed have been proposed, including the pair-wise nearest neighbor (PNN) algorithm [39] and the self-organizing feature map algorithm [40].

4.3.2.4 Cache-Memory Design

Although a vector quantizer can shrink the vector space to a code-book, still the code-book is too large to be searched for a match for real-time coding purposes. K. K. Truong et al applied the cache-memory technique from computer architecture design [41], to the code-book construction in VQ. This technique defines the full-size code-book obtained from clustering as the main code-book, which is stored off line. On the other hand, an on-line cache code-book which is far shorter than the main code-book is initially searched for VQ. Only when a vector cannot find a satisfactory match (distortion below a pre-defined threshold) in the cache code-book, is the main code-book searched and the cache code-book updated. Simulations demonstrate that the cache miss frequency is mostly below 5%.

While it dramatically cuts down the operations in VQ code-book searching, the cache-memory technique needs a carefully designed cache replacement rule to function optimally. Least recently used (LRU), for example, substitutes the new match found in the main code-book for the codeword with the oldest hit time record in the cache

code-book. In least frequently used (LFU), the codeword to be removed is the one with the lowest number of hit. Another popular rule is first-in-first-out (FIFO) which takes out the codeword that stayed longest in the stack.

Truong et al also discusses the choice of the cache size for the different update rules and concludes that the fixed sizes of 100, 200 and 25 for the LRU, LFU and FIFO respectively are the most efficient [41].

The adaptive working set model (AWSM) is also proposed in [41], for the design of the cache code-book. In this model, the cache is the updated under any replacement rule, but rather by the location of the coding process. In the case of image block vector coding, the cache is the set of codewords that have been used in a causal window located around the current block. The most prominent advantage of AWSM is the adaptation of the window size to the cache-miss frequency, which uses the cache source most efficiently.

4.3.2.5 Entropy Coding

After quantization, the matched codeword needs to be translated to a binary stream so as to ultimately realize the purpose of quantization. Entropy coding expresses a set of quantized vectors with a set of unique symbols, the average length of which equals the signal entropy, and therefore is the shortest possible, according to Shannon's conclusion. Using an entropy coding (e.g. Huffman coding) scheme for the cache index, a VQ can achieve an even better coding efficacy.

4.3.2.6 Vector Quantizer for Affine Models

The training set that we use to construct the main code-book is obtained from a variety of video sequences that are representative of common global motion elements. These include rotation (bream, "Welcome to MPEG4 World"), translation (*coastguard*, *container*, *mobile* and *calendar*), panning (*foreman*, *flower garden*), and zooming (*highway*). To maximize the accuracy of the affine models used for training,

we specify the number of MO's for every sequence, instead of adapting it to the segmentation process. For example, the 300-frame coastguard sequence would generate 2(the boat and the background)×300 sets of affine models, and the 300-frame mobile and calendar sequence will generate 4(calendar, wall, train and ball)×300 sets of affine models. Note that only the scaling parameters a_1, a_2, a_3 and a_4 are used with vector quantization. With the above sequences, we have more than $10,000$ 4-D vectors that are used to train 300 codewords in the main code-book $Q_m = \{A_1, A_2, \ldots, A_{300}\}$, where $A_i = (a_{i,1}, a_{i,2}, a_{i,4}, a_{i,5})^T$.

The cache code-book that we search for VQ is 16 items long in our codec, noted as $Q_i = \{A_1, A_2, \ldots, A_{16}\}$. After the "Motion Segmentation" module produces the N affine models, the N 4×1 scaling vectors are checked against Q_c for a closest match $A_I(1 \leq l \leq 16)$. If the quantization distortion D_v is greater than a predefined threshold, then a *cache-miss* happens and Q_c needs to be updated. Let $A_i(n)$ be the cached codeword whose last reference prior to frame n is the oldest. By the LRU rule, $A_i(n)$ is dismissed from the cache code-book. The main code-book is then searched, and the best match $A_I(17 \leq l \leq 300)$ is put into the cache code-book.

When the quantization distortion D_v is below the predefined threshold, a *cache-hit* is observed. In this case, Q_c keeps the same members but needs to be re-ordered. Figure 31 illustrates the updating process of Q_c .

Our experiments demonstrate that in 95% of the cases, no more than 10 bits are spent on the vector quantized affine model, which is less than 2% of the overhead of the frame bit-rate in our testing scenarios. $[a_3, a_6]$ of the affine model is coded predicatively as a conventional motion vector, except that the resulting VLC is put in the slice header.

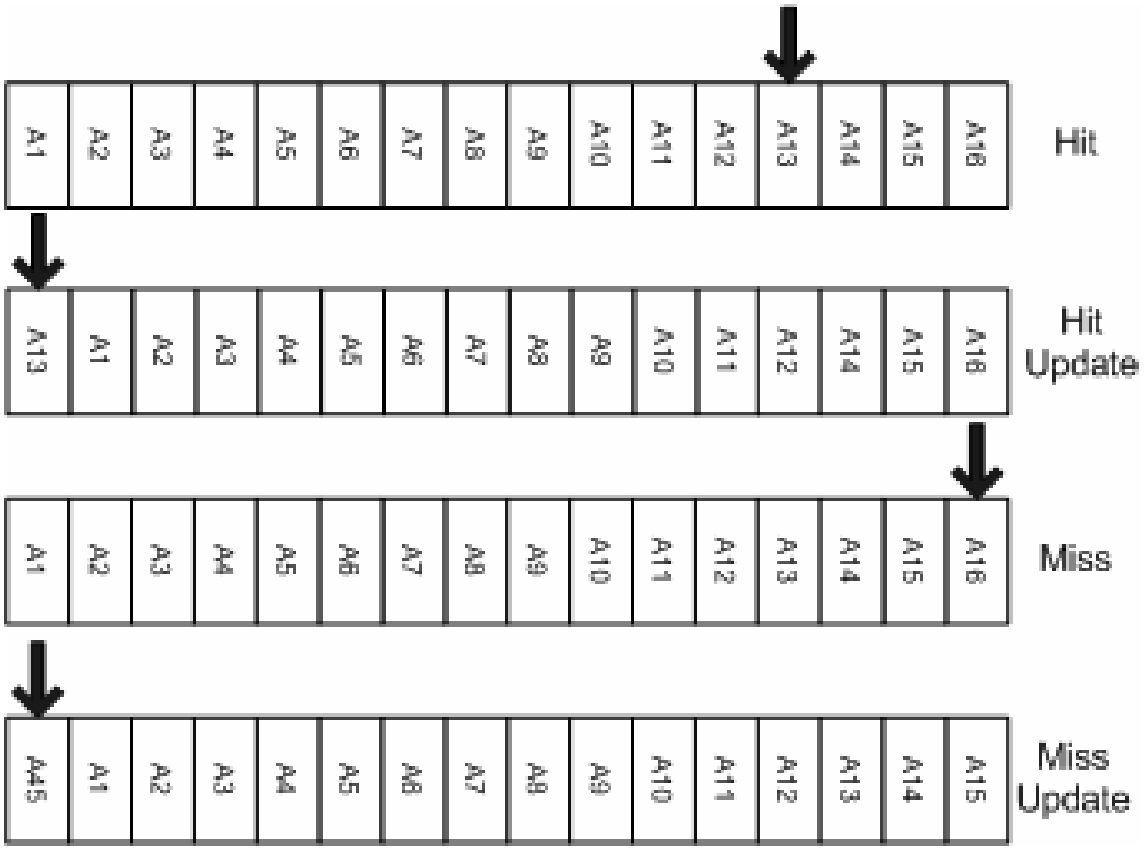


Figure 31: The updating processes of the LRU refreshment strategy during a cache-hit (top) and cache-miss (bottom).

4.4 Global Motion Model and Blocking Artifact

4.4.1 Blocking Artifacts

Blocking artifacts have been a major drawback of block-based hybrid video coding systems. These originate from two aspects of the encoder: (i) block-based motion estimation and (ii) block-based DCTs. Various error concealment approaches [55] [56] [57] have been proposed to reduce the artifacts at the boundaries of the neighboring MB's. H.264's introduction of in-loop filter smoothes out the blocking artifacts before a compensated frame is used as a reference, and reduces the temporal propagation of the artifacts. However, the in-loop filter operation is computationally expensive and erases some of the reconstruction texture details because of its low-pass nature.

The affine global motion model however, treats one motion object as a whole

and applies the same warping operation to all the MB's that belong to the MO. This way, the motion compensation works spatially continuously on the MB's and the reconstruction result appears smooth at the MB boundaries, even without any filtering operation. The low pass element in the interpolation step in the affine model estimation is not more detrimental to the texture details than the interpolation implemented to perform quarter-pixel block matching. However, since the MB's coded with affine mode still undergo the block-based DCT transformation, the in-loop filter remains necessary. Since the filtering condition is satisfied less frequently, the filtering operation is called less often and requires fewerf computation resources.

4.4.2 Perceptual PSNR

According to studies of the human vision neural system [53], human eyes are more sensitive to the inner area of a motion object than to the boundaries, more concerned about foreground objects than background ones and, more attentive to fast moving objects than relatively static ones. These properties can be well exploited by the proposed video codec with its segmentation motion structure. For example, the distortion of the interior MB's of a motion object can be weighted in the Lagrange formula more heavily than that of the boundary MB's [54]. This can drastically affect the choice of the mode, and will most likely result in more choices of the AFFINE mode. The modification of the mode selection criteria remains to be added to the current project. Hopefully, it will increase the subjective appearance of the decoded sequence, and hence the perceptual PNSR, as defined in [54].

To take the perceptual distortion into consideration, we simplified the pixel-wise weighted mean square error (MSE) defined in [54] to the MB-wise weighted MSE in

$$J_{MODE}(S_k, I_k|Q, \lambda_{MODE}) = C_{PER} D_{REC}(S_k, I_k|Q + \lambda_{MODE} R_{REC}(S_k, I_k|Q)) \quad (20)$$

which is a modified version of (15). Namely, for MB's with the same segmentation label for all its 16 4×4 sub-blocks, we assume it is the internal part of an MO.

According to [54], when the same MSE distortion occurs at the central and boundary regions, the perceptual distortion is larger in the central region. So we use a $C_{PER} > 1$ for such an MB. For an MB with different segmentation label, $C_{PER} = 1$ is used, and eq. (20) becomes a Lagrangian equation eq.(15) as in [23]. Because human vision is more sensitive to active regions than to static ones, we define

$$C_{PER} = 1 + \sigma \cdot \sqrt[2]{{a_3}^2 + {a_6}^2} \tag{21}$$

where a_3 and a_6 are the translational elements from the MO containing the current MB and σ is a heuristic coefficient.

The rate R_{REC} in (20) is the sum of the bits for syntax, motion vectors, residual errors and shape information. Coefficient λ_{MODE} is chosen according to [23] as (17), where Q is the quantization step size for the current frame.

The perceptual PSNR we use in the affine-model codec system is defined as

$$PSNR_{PER} = 10 \cdot \log_{10}(\frac{255^2}{\sum C_{PER} \cdot D_{REC}}) \tag{22}$$

where the sum is over the whole frame.

Figure 34 also presents the rate-distortion performances of the two encoders, when distortion is defined as the perceptual distortion by (11), (13) and (14). The proposed encoder still beats the JM7.3 encoder by a greater margin at the lower end of the bitrate spectrum, but by a better margin over the whole range compared to the case where the regular PSNR is used to measure distortion. Note that the calculation of perceptual PSNR for JM-created bitstream utilizes the segmentation map from the proposed encoder.

4.5 Macroblock Level Mode Selection

The choice between the N affine and 4 (16×16, 16×8, 8×16, 8×8) BM motion models is made in the MB coding loop. Two factors are taken into consideration: the coded bit length for the MB and the reconstructed PSNR. The BM model needs

some bits for motion vectors, which are totally avoided by the affine models; while the BM compensated MB might be smaller in magnitude than the affine warped MB, and thus result in fewer bits for the residual DCT, which is always the case, because of the BM motion compensation's mean square error (MSE) optimization target. Chances are that the affine model can beat the BM method for large error MB's, due to the absence of motion vectors. The motion mode selection between affine and BM models shares the same philosophy as the mode selection among different block sizes in the H.264 standard: the optimal coding mode finds a best tradeoff between motion vector bit-rate and residual error bit-rate.

While the BM compensated MB's and the corresponding residual errors are computed in the MB loop in the JM encoder, the added affine warped MB's are simply clipped from the synthesized warped reconstruction (Figure 23), which is obtained before the MB loop starts.

With the BM estimated motion vectors and the affine warp parameters, the MB modes are decided based on a Lagrangian cost function [49], which is minimized when the optimal rate-distortion combination is achieved as in (15), where the MB mode I_k is varied over the set of possible MB modes

$$I = \{I4, I16, SKIP, P16 \times 16, P16 \times 8, P8 \times 16, P8 \times 8, A_1, A_2, \ldots, A_N\} \quad (23)$$

The distortion D_REC is defined in (16), where A is the MB to be coded.

Figure 32 shows the distribution of MB mode selection in frame2 of *foreman.qcif* by the JM encoder, while Frame 33 shows the mode distribution with the proposed codec. It can be seen that half of the MB's use the affine modes in Figure 33.

4.6 Simulation Results

Simulation results demonstrate a notable decrease in the number of bits for an average coded P-frame for similar quality. Simulations are carried out on the 300 frames of *coastguard*, which is a typical scene composed of a focused front object over a moving

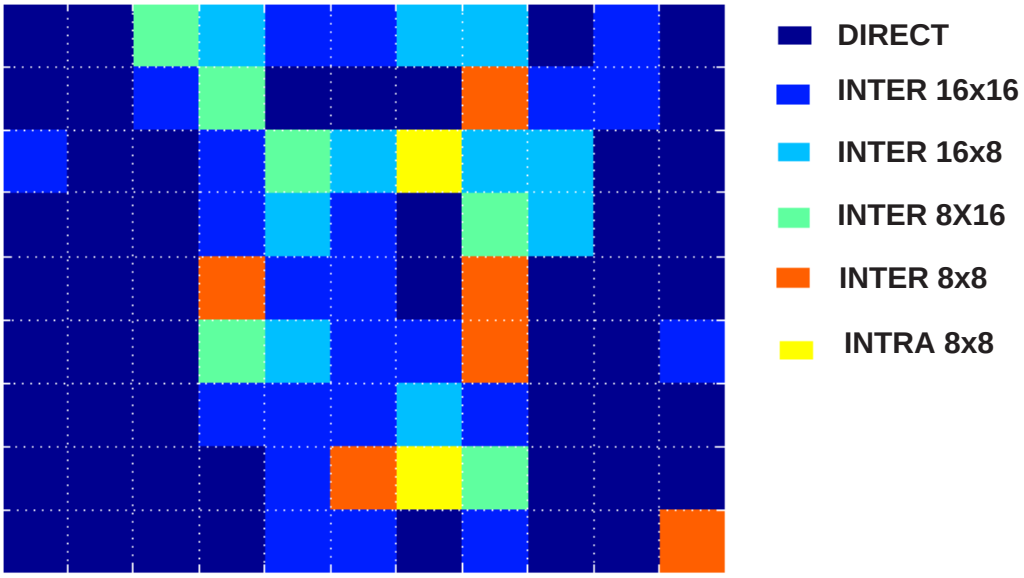


Figure 32: Mode distribution of the original H.264 encoder, on the second frame of foreman.qcif.

background; and the last 100 frames of foreman, which is a classical panning scene. Rate distortion performances of the proposed codec system and the original JM codec system have been reported and compared in Figure 34.

Bit rates shown in Figure 34 are for P-frames only. As the bit rate goes up, a greater portion of the bandwidth is allocated to the residual error, and savings from the motion vectors affect the whole bitstream marginally. Due to the the BM model's MSE optimization target, even if it doesn't give as accurate a motion vector as the affine model does in cases involving rotating and zooming, it might still generate a smaller residual error block. However, when the target bit rate for video communication is low, the quantization of DCT coefficients is forced to be crude, and the subtle differences of the error block produced by the two motion models is diminished.

It can be concluded from Figure 34 that at the lower end of the simulation range, the bit rate saving is about 18%, corresponding to a PSNR increase of about 1 dB.

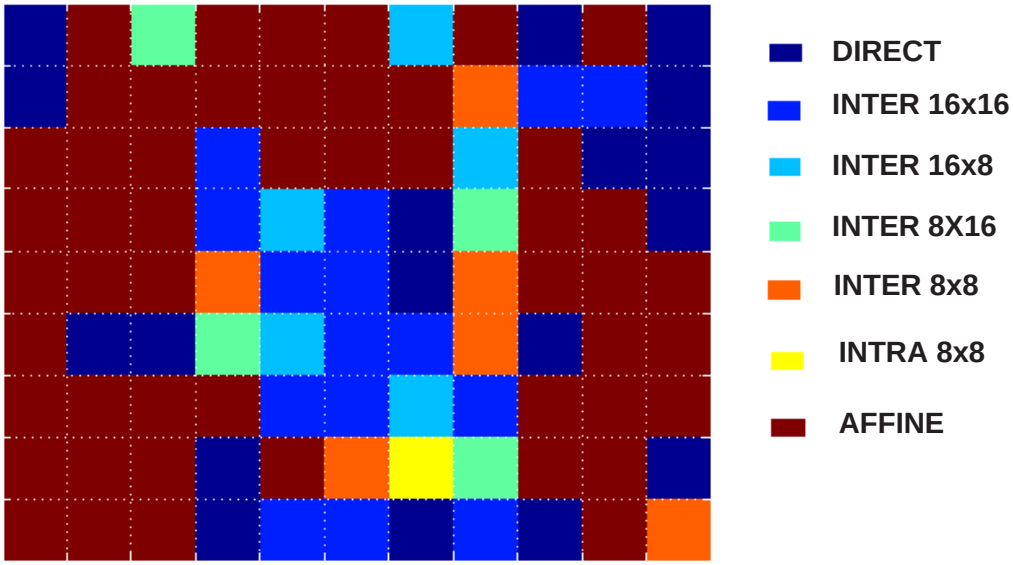


Figure 33: Mode distribution of the proposed encoder, on the second frame of foreman.qcif.

In all cases, 20% to 40% of the inter-coded MB's choose one of the affine modes.

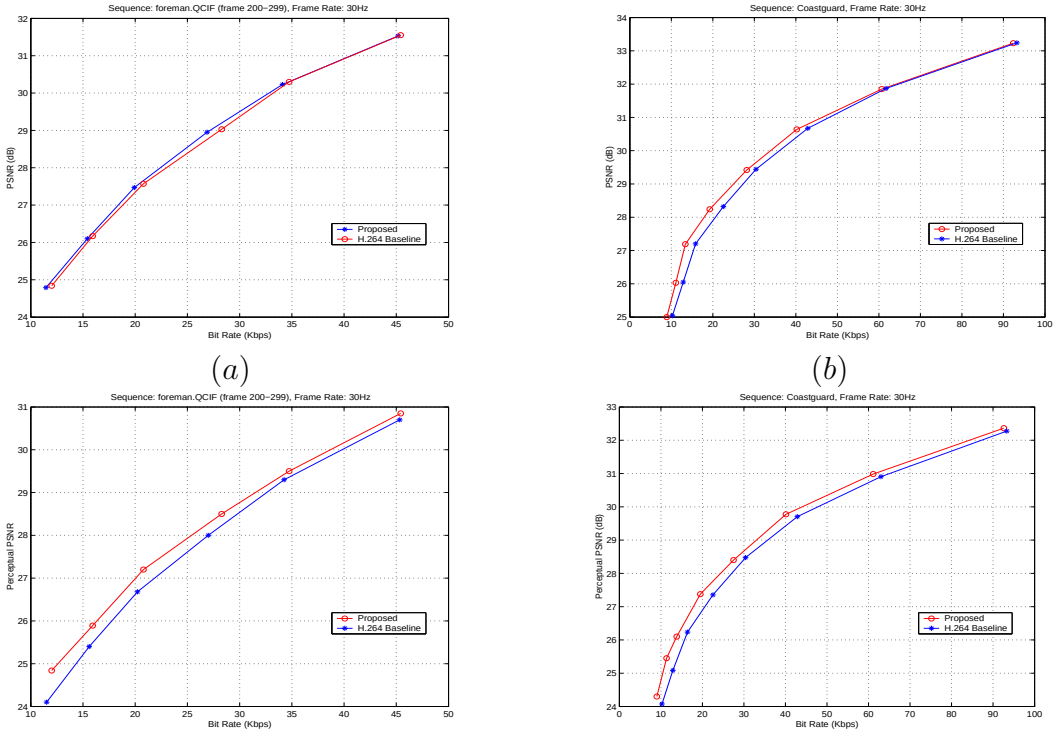


Figure 34: Rate distortion performance for sequence *foreman.qcif* (panning) and *coastguard.qcif* (global translation).Both PSNR and perceptual PSNR are used for measuring distortion. The upper left is the result of perceptual PSNR with sequence *coastguard*; the upper right of PSNR with *coastguard*; the lower left of perceptual PSNR with sequence foreman (last 100 frames); the lower right of PSNR with foreman

CHAPTER V

CONCLUSION AND FUTURE WORK

5.1 Simulation Results and Conclusion

For the sequences of *Foreman* and *Coastguard*, a bit-rate saving of $1 \sim 18\%$ compared to the H.264 baseline coder is observed with transmission bitrates ranging from 10kbps to 100kbps, with more than 50% of the MB's choosing one of the affine modes. Greater compression efficiency is achieved at the lower end of the bitrate range, as expected. Also, the larger the video frame size, the more notable the improvement over the H.264 baseline.

Besides increasing the coding efficiency, the global affine model manifests the following features that refine the compressed video quality. (i) When the number of slices per frame is more than 1, the global affine motion model can enhance the error-resilience of the video stream, because the affine motion parameters are duplicated in the headers of different slices over the same frame. (ii) The global motion model predicts a frame by warping the whole reference frame and this helps to decrease blocking artifacts in the compensation frame.

5.2 Future Work

5.2.1 Scalability in Motion Segmentation

As explained above, in the video coding scenario, accurately labeling the interior pixels/blocks of a motion object is much more important than actually depicting the contour of the object. The reason for this is that the boundary blocks will not select the AFFINE mode anyway. They will more likely choose a block-matching inter mode, or even an intra mode. The set of blocks that are eventually compensated by one of the AFFINE models is a subset of the corresponding motion object.

On the other hand, the pixels of a motion object in the boundary blocks may contribute favorably to the estimation of the affine model. To exploit the motion information at these locations, motion segmentation can be designed as a series of stages. First, MB-wise segmentation based on MB-wise motion vectors is performed. Only those boundary MB defined in the first stage are considered for re-labeling in the iterations of the second stage, in which the segmentation task is carried out on a block-wise basis. This process continues until a pixel-wise segmentation map is accomplished.

Whether a merely MB-wise motion segmentation or a refining segmentation is more efficient in modeling the affine motion parameters will need be decided by experiments and further study.

5.2.2 Error Concealment Algorithms

In a conventional hybrid video codec, the motion information is coded MB by MB. When a video packet is lost, the motion vectors for the MB's within are gone, together with the corresponding residual errors. In a partitioning mode, motion information is put in partition I, which has a higher priority of importance than residual errors, because without the accurate motion compensation, the residual error is nothing but noise. The proposed global motion modeling makes it possible to raise the protection priority of motion information to the same level as the frame header, in the hope that even if an MB has not selected the AFFINE mode by Lagrange optimization, it will achieve a reasonably good reconstruction, when the selected motion vector is lost in transmission. An error concealment infrastructure needs to be designed in accordance with the video stream carrying global, as well as local, motion information. When one frame is one slice, the header burden due to global motion model is trivial and can be used for error concealment when the data packet is lost. When there are multiple slices in a frame, the overhead volume will increase, and the maximum

number of motion objects needs to be adapted to the slice size. In this case, the affine models are simply duplicated in theheaders of slices that dwell in the same frame, and increased error resilience comes from redundancy. When a whole slice including the header is lost, its spatially neighboring slices can offer useful motion information for concealment.

A possible problem for the concealment scheme is the decision of motion object identity for a lost MB. Because although the global motion models are in the header, the object index for an MB is coded locally and subject to more contamination, when there are multiple motion objects in the frame. Future research needs to design a mechanism, to find out the most possible motion object index for a lost MB.

REFERENCES

[1] Text for ISO/IEC FDIS 14496-2 Visual, ISO/IEC JTC1/ SC29/ WG11 N2502, Vancouver, Canada, Nov. 1998.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "An Overview of the H.264/AVC video coding standard," in *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 560-576, vol. 13, Jul. 2003.

[3] T. Wedi and H. G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," in *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 577-586, vol. 13, Jul. 2003.

[4] M. Flierl and B. Girod, "Generalized B pictures and the draft H.264/AVC video-compression standard," in *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 587-597, vol. 13, Jul. 2003.

[5] W. Wien, "Variable block-size transforms for H.264/AVC," in *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 604-613, vol. 13, Jul. 2003.

[6] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," in *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 688-703, vol. 13, Jul. 2003.

[7] B. Girod, "Motion-Compensation Prediction with Fractional-Pel Accuracy," *IEEE Trans Communications*, vol. 41, pp. 604-612, Apr. 1993.

[8] T. Wiegand, X. Zhang, and B. Girod,"Long-Term Memory Motion-Compensated Prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 70-84, Feb. 1999.

[9] Y. W. Huang, B.Y. Hsieh, T.C. Wang, S.Y. Chien, S.Y. Ma, C.F. Shen, and L.G. Chen, "Analysis and Reduction of Reference Frames for Motion Estimation in MPEG-4 AVC/JVT/H.264," *Proc. International Conference on Multimedia and Expo 2003 (ICME2003)*, vol. 2, pp. 812-815, Baltimore, MD, U.S.A. Jun. 2003.

[10] S. Nogaki and M. Ohta, "An Overlapped Block Motion Compensation for High Quality Motion Picture Coding," *IEEE Int. Symp. Circuits Syst.*, pp. 184-187, May 1992

[11] M.T. Orchard and G. J. Sullivan, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach," *IEEE Trans. Image Processing*, vol. 3, pp. 693-699, Sep. 1994.

[12] A. M. Tekalp, *Digital Video Processing*. London, U.K.: Prentice-Hall, 1995.

[13] M. Flierl and B. Girod, "Generalized B Pictures and the Draft H.264/AVC Video-Compression Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 587-597, Jul. 2003.

[14] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COMM-29, pp.1799-1808, Dec. 1981.

[15] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 29-Dec. 3 1981, pp. G5.3.1-5.3.5.

[16] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COMM-33, pp. 888-896, Aug. 1985.

[17] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, pp. 950-953, July 1990.

[18] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*,vol. 6, pp. 313-317, June 1996.

[19] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 419-423, Aug. 1996.

[20] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast blockmatching motion estimation," *IEEE Trans. Image Processing*, vol. 9, pp. 287-290, Feb. 2000.

[21] C. Zhu, X. Lin and L.P. Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 349-355, Aug. 1996.

[22] G. J. Sullivan and R. L. Baker, "Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks," *Proc. GLOBECOM'91*, Phoenix, AZ, Dec. 1991, pp. 85-90.

[23] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, G.J.Sullivan, "Rate-constrained Coder Control and Comparison of Video Coding Standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 688-703, Jul. 2003.

[24] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. 7th Inl Joint Conf on Artificial Intelligence (IJCAI) 1981*, Vancouver, BC, Canada, pp. 674-679.

[25] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," *Proc. Int. Conf. Cybernetics Society*, 1975, pp. 163-165.

[26] M.C. Lee, W.G. Chen, C. B. Lin, C. Gu, T. Markoc, S. I. Zabinsky, and R. Szeliski, "A Layered Video Object Coding System Using Sprite and Affine Motion Model", *IEEE Trans. Circuits Syst. Video Technol.*, pp. 130-145, Vol. 7, No. 1, Feb., 1997.

[27] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H. Watanabe, "Two-stage motion compensation using adaptive global MC and local affine MC," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 75-85, Vol. 7, No. 1, Feb. 1997.

[28] K. Zhang, M. Bober, and J. Kitter, "Image sequence coding using multiple-level segmentation and affine motion estimation", *IEEE Journal on Selected Areas in Comm.*, pp. 1704-1713, Vol. 15 No. 9, Dec. 1997.

[29] E. Steinbach, T. Wiegand and B. Girod, "Using Multiple Global Motion Models for Improved Block-Based Video Coding," *Proc. Int. Conf. Image Processing 99 (ICIP99)*, Kobe, Japan, Oct. 1999, vol. 2, pp. 56-60.

[30] A. Smolic, Y. Vatis, H. Schwarz, and T. Wiegand, "Improved H.264/AVC coding using long-term global motion compensation," *Proc. VCIP 2004, SPIE Visual Communications and Image Processing*, San Jose, CA, USA, Jan. 2004.

[31] A. Smolic, T. Sikora, and J.-R. Ohm, "Long-Term Global Motion Estimation and its Application for Sprite Coding, Content Description and Segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 9, No. 8, pp. 1227-1242, Dec. 1999.

[32] Y. Lu, W. Gao, and F. Wu, "Efficient Background Video Coding With Static Sprite Generation and Arbitrary-Shape Spatial Prediction Techniques," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 13, No. 5, pp. 394-405, May. 2003.

[33] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann and G. M. Schuster, "MPEG-4 and Rate-Distortion-Based Shape-Coding Techniques," *Proc. IEEE*, Vol. 86, pp. 1126-1154, June. 1998.

[34] Y. Wang, G. Wen, S. Wenger, and A. K. Katsaggelos, "Review of Error Resilient Techniques for Video Communications," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61-82, July 2000.

[35] Ismaeil, I., Docef, A., Kossentini, F., and Ward, R., "Efficient Motion Estimation Using Spatial And Temporal Motion Vector Prediction," *Proc. International Conference on Image Processing*, Kobe, Japan, Vol. 1, pp. 70-74 , Dec, 1999

[36] M.G. Liu, C.H. Hou, "A Fast Block-Matching Motion Estimation Algorithm Based on Spatial-Temporal Motion Vector Correlation," *Proc. International Symposium on Intelligent Multimedia*, Hong Kong, pp. 498-501, May, 2001

[37] S.D. Kim, J.B. Ra, "An Efficient Motion Vector Coding Scheme Based on Minimum Bit Rate Prediction," *IEEE Transaction on Image Processing*, Vol. 8, pp-1117-1120, Aug, 1999.

[38] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantization design," *IEEE Trans. Commun.*, vol. 28, pp. 84-95, Jan. 1980

[39] W.H.Equitz, "New vector quantization clustering algorithm," *IEEE Trans Commun.* Vol.37, pp. 1568,-1575, Oct. 1989.

[40] K.K.Truong, "Multilayer Kohonen image codebooks with a logarithmic search complexity," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 4, pp. 2789-2792, May 1991.

[41] K.K. Truong, "Vector quantizer design for images and video based on hierarchical structures," Thesis (Ph.D.)–School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA, U.S.A. 1993.

[42] P.V.C. Hough, "Machine Analysis of Bubble Chamber Pictures," *Proc. of International Conference on High Energy Accelerators and Instrumentation, CERN*, 1959.

[43] Y. Wu, "Optical Flow and Motion Analysis," ECE510-Computer Vision Course Notes, Department of Electrical and Computer Engineering, Northwestern University, Winter 2001.

[44] J.Y.A.Wang and E.H.Adelson, "Representing moving images with layers," *IEEE Trans. Image Processing*, Vol. 3, No. 5, Sep. 1994.

[45] G.D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, "Motion segmentation by multistage affine classification," *IEEE Trans Image Processing*, Vol. 6, No. 11, Nov. 1997.

[46] Y.Altunbasak, P.E.Eren and A.M.Tekalp, "Region-based parametric motion segmentation using color information," *Graphic Models on Image Processing*, Vol. 60, No. 1, pp. 13-23, Jan. 1998.

[47] N. Grammalidis, D. Beletsiotis and M. G. Strintzis, "Sprite Generation and Coding in Multiview Image Sequences," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 2, Mar. 2000.

[48] X. Li, J.R.Jackson, A.K.Katsaggelos, R.M.Mersereau, "An adaptive coding scheme using affine motion model for MPEG P-VOP," *Proc. of International Conference on Acoustic and Speech Signal Processing (ICASSP04)*, Montreal, Canada, May 2004.

[49] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G. J. Sullivan, "Rate-constrained coder Control and Comparison of Video Coding Standard", *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, pp.688-703, Jul. 2003.

[50] X. Li, J.R.Jackson, A.K.Katsaggelos and R.M.Mersereau, "Multiple Global Affine Motion Model for H.264 Video Coding with Low Bit Rate," *Proc. of SPIE Conference on Image and Video Communications and Processing III*, San Jose, CA, U.S.A., Jan. 2005.

[51] E. Kowler, "Eye Movements and Their Role in Visual and Cognitive Processes," *New York: Elsevier Science*, 1990.

[52] J. K. O'Regan and A. Levy-Schoen, " Eye Movements: From Physiology to Cognition," *New York: Elsevier Science*, 1987.

[53] C. W. Oyster, "The human eye: structure and function," Sunderland, Mass. Sinauer Associates, 1999.

[54] C-W Wong, O. C. Au, B. Meng, H-K Lam, "Perceptual Rate Control for Low-Delay Video Communications," *Proc. of International Conference on Multimedia and Expo (ICME2003)*, Vol. 3, pp. 6-9, Jul. 2003.

[55] H. S. Malvar and D. H. Staelin, "The LOT: Transform Coding without Blocking Effect", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp.553-559, Apr. 1989.

[56] R. Rosenholtz and A. Zakhor, "Iterative Procedure for Reduction of Blocking Effects in Transform Image Coding", *IEEE Trans. Circuit Syst. Video Technol.*, vol. 2, pp.91-95, Mar. 1992.

[57] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized Reconstruction to Reduce Blocking Artifacts of Block Discrete Cosine Transform Compressed Image," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 3, pp.421-432, Dec. 1993.

VITA

Xiaohuan Li was born in the city of Chongqing, China in November 1976. She received her Bachelor of Engineering degree from the Automation Department of Tsinghua University, Beijing, China in June 2000. From September 2000 to December 2001, she studied in the Department of Electrical and Computer Engineering in Northwestern University, Evanston, IL, U.S.A. During the 4 academic quarters she spent in Northwestern, she served as a Teaching Assistant in the ECE department. Meanwhile, she conducted research in the field of error concealment in video shape coding, with the advisory of Dr. Aggelos K. Katsaggelos, director of the Image and Video Processing Laboratory. During the summer quarter of 2001, she worked as an exchange student in the Media Lab of Hochschule fr Technik (HSR), Rapperswil, Switzerland, where she initiated her research on video concealment with Dr. Guido M. Schuster. Xiaohuan received her Master of Science degree from Northwestern University in June 2002. From January 2002 to May 2005, she worked as a Graduate Research Assistant in the School of Electrical and Computer Engineering in Georgia Institute of Technology, Atlanta, GA. During her stay in the Center of Signal and Image Processing, she worked extensively in the field of video coding, with a focus on global motion models used in video compression, under Dr. Joel R. Jackson's advisory. From June 2005 on, she worked as a video architecture engineer in NVidia Corp. in Santa Clara, CA. Xiaohuan defended her Ph.D. thesis in May 2006 and received her Ph.D. degree in Electrical Engineering from Georgia Tech in July 2006.

Non-academic biographical highlights of her graduate years include her wedding to Dr. Ji Chen in August 2001 during her master study and her giving birth to her son Dawson Chen in January 2004 during her Ph.D. study.