Motion Estimation and Mode Decision for Low-Complexity H.264 Decoder

Yong Wang and Shih-Fu Chang
{ywang, sfchang}@ee.columbia.edu
Department of Electrical Engineering
Columbia University

Summary

Emerging video coding standards like H.264 achieves significant advances in improving video quality, reducing bandwidth, but at the cost of greatly increased computational complexity at both the encoder and the decoder. Playing encoded videos produced by such compression standards requires major computational resources and thus power on various handheld devices that are getting increasingly popular in mobile applications. Among the components in the decoding system, the interpolation procedure used in the motion compensation component consumes the largest computation (about 50%) due to the use of sub-pixel motion vectors. One way to reduce this major cost is to change the coding algorithm so that the generated compressed bit streams incur less interpolation operations. In this work, we propose a novel Complexity Adaptive Motion Estimation and mode Decision (CAMED) system to optimize the selection of the motion vectors and motion compensation block modes in order to significantly reduce the computational cost while keeping the video quality virtually unchanged. We accomplish this goal by (1) applying a rigorous methodology to extend the conventional rate-distortion optimization framework to include the computation term, (2) developing a complexity model that can reliably determine the appropriate parameter (i.e., Lagrange multiplier) needed for optimizing the rate-distortion-complexity tradeoff relationships, and (3) a complexity-control algorithm to meet any specified target complexity level while keeping the complexity as consistent as possible throughout the video sequence. Our method can be applied to any existing H.264 encoder system and is compatible with any standard-compliant decoder. Our extensive experiments with different video contents, bit rates, and complexity levels show very promising results in reducing the number of interpolation by up to 60% while keeping the video quality almost intact (quality difference less than 0.2dB). Since the interpolation operation constitutes the largest computational cost component at the decoder, our results have great potential for reducing the power consumption in any practical video decoding systems using the latest video coding standard such as MPEG-4, H.264 and Motion Compensated Embedded Zero Block Coding (MC-EZBC).

Innovation Claims:

1. An improved H.264 video encoder that generates video bit streams that have high quality but require much less computation (power) in any standard compatible decoder.

2. A joint rate-distortion-complexity (R-D-C) optimization framework and associated algorithms that allow for optimization of the tradeoff among video quality, bit rate, and computational complexity (described in Section 3.1)
3. Methods for estimating the computational complexity and hardware power consumption associated with each coding options using different motion vector type and block mode (described in Section 3.2)
4. A novel complexity control method that can achieve arbitrary target complexity levels by monitoring the complexity consumption status and effectively predicting the appropriate control parameter to be used in the R-D-C optimization procedure (described in Section 3.3)

Description

1. Introduction

Most of today's video coding systems encode the video bit streams to achieve the best video quality (e.g., the minimal signal distortion) while satisfying certain bitrate constraints. Specifically the following optimization problem formulation is often adopted.

$$\begin{aligned} &\min_{P} D(\mathbf{P}) \\ &s.t., R(\mathbf{P}) \le R_T \end{aligned} \tag{1}$$

where $\mathbf{P}$ represents the control variables (CV) which eventually determine the final video quality and bit rate. Typical CVs include quantization parameter (QP), motion vector, motion estimation block mode, etc. D is the distortion introduced by the encoding process. R is the bit rate of the encoded video and R_T is the target bit rate. The solution of the above problem aims at finding the optimal control variables for each coding unit in order to minimize the average distortion while satisfying the bit rate constraint. Though in practice, some comprised choices may be made for the control variables due to the resource limitations (e.g., memory and computational complexity), Equation (1) does not explicitly models the required complexity in video encoding or decoding. As a matter of fact, many recent advances in the coding efficiency are accomplished by using increasingly complex computational modules, such as sophisticated processes for motion estimation.

On the contrary, many media application devices such as mobile handheld devices are getting smaller and lighter. The computational resources available on the handheld devices become relatively scarce, given the increasing functionalities and complexity of applications running on the devices. Therefore, recently in the literature there is growing interest in complexity (power) aware video coding solutions. ARMS and National Semiconductor develop a systematic approach called PowerWise technology, which can efficiently reduce the power consumption of mobile multimedia applications through adaptive voltage scaling (AVS) [9]. Zhou *et al* implements an H.264 decoder based on Intel's single-instruction-multiple-data (SIMD) architecture that reduces the decoding complexity and improved the H.264 decoding speed by up to three times [8]. In [14] Ray and Radha propose a method to reduce the decoding complexity by selectively replacing the I-B-P Group of Pictures (GOP) structure with one using I-P only. Lengwehasatit and Ortega develop a method to reduce the decoding complexity by optimizing the Inverse DCT implementation [15]. He *et al* optimizes the power-rate-distortion performance by constraining the sum of absolute difference (SAD) operations during the motion estimation process at the encoder [16]. In addition, power aware joint source channel coding is also an active topic for mobile wireless video communication [10][11][12]. Unlike the conventional paradigm using complex encoding and light decoding, Girod *et al* proposes the distributed video coding system, which transfers the motion estimation process from the encoder to the decoder so that the encoding complexity can be greatly reduced [17].

In our work, we focus on an important aspect of the complexity minimization problem – how to develop an encoding algorithm that achieves both high video quality and low decoding complexity while satisfying the bit rate constraint. Our goal is to reduce the complexity requirement of emerging video codecs like H.264 on the resource-limited devices like handheld devices. Our work is different from the rest in that we modify the video encoding algorithm to minimize the required complexity at the decoder, not the encoder. Our approach does not require any change in the existing decoder implementations. Our method modifies the non-normative parts of the H.264 encoding algorithm to generate bit streams that can be decoded by standard-compliant decoders. In other words, we develop novel H.264 encoding algorithms that generate low-decoding-complexity and high-quality bit streams. Other techniques for the decoder power minimization, such as those in [8][9][14][15], are complementary and can be combined with our solution.

Specifically, when considering the decoder's complexity during video encoding, we reformulate the optimization problem as follows.

$$\begin{aligned} &\min_{P} D(\mathbf{P}) \\ &s.t., R(\mathbf{P}) \le R_T \\ &\quad C(\mathbf{P}) \le C_T \end{aligned} \tag{2}$$

where C is the computational complexity at the decoder. Compared with the problem defined in Equation (1), a constraint on computational complexity is explicitly added. The solution for in Equation (2) needs to determine the best control variables, P, for each coding unit. Similar to the case for Equation (1), the control variables include quantization parameter, block mode of the motion compensation process, and the associated motion vectors.

Among the control variables, the motion vectors have the largest impact on the decoding complexity. Motion vectors can be of integer or fractional values corresponding to a displacement distance of integral pixels or fractional pixels. When a motion vector is of a sub-pixel value, multi-tap filtering is required to compute interpolation to form a reference block that is needed in the motion compensation process in the decoder. Such interpolation filtering involves huge computational cost and typically significantly increases the overall decoding complexity. Fig. 1 shows the breakdown of the complexity of a typical H.264 decoder implementation [3]. It is clear that the interpolation component constitutes about 50% of the decoding complexity. Although for mobile multimedia applications there are other power consuming components like wireless communication, display, and memory access, the decoding process is typically a significant one. Therefore improving the cost associated with the interpolation process is important for achieving a low-power decoding system, either in hardware or software.

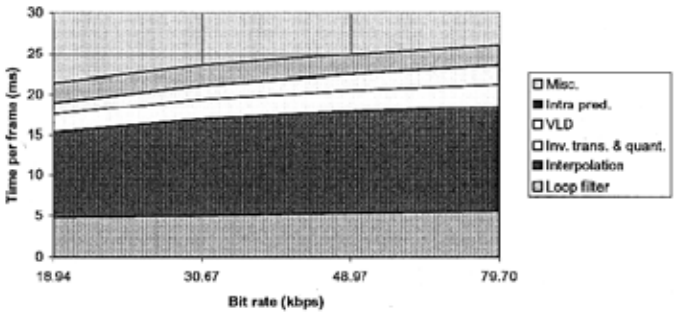


Fig. 1: Computational complexity distribution in decoding the *Foreman* video sequence of the QCIF resolution

In this work, we extend the conventional rate-distortion framework based on the Lagrange optimization method to incorporate the computational complexity. To estimate the complexity associated with different types of motion vectors, we develop models to approximate the implementation cost involved in the interpolation filtering process. In addition, we extend the rate control algorithm to handle the joint rate-complexity control issue so that both the targets of rate and complexity can be met. Our optimization method intelligently selects the block mode and motion vector type of each coding unit to achieve the highest video quality. When tested over a diverse set of video sequences over different bit rates, our solution achieves very significant complexity reduction (up to 60%) of the most complex component, interpolation filtering, while keeping the video quality almost intact (degradation within 0.2dB). When incorporated into the practical system, our solution has great potential in reducing the overall power consumption.

The rest of this paper is organized as follows. Section 2 includes reviews of principle components of a typical hybrid video coding system such as H.264. It describes the basic concepts of motion estimation, motion compensation, and their implication on the computational complexity. The rate-distortion optimization framework based on the Lagrange optimization method is reviewed. It also explains the process used to control the rate over frames to meet the overall target. In Section 3, we present the proposed CAMED method for generating low-complexity bit streams. Section 4 includes the experiment results. Conclusions and future work are described in Section 5.

2. Review of Typical Hybrid Video Coding Systems

Fig. 2 illustrates the system diagram for a typical hybrid motion compensation and block-transform video coding system. The darker box shows the decoding procedure, which is also simulated in the encoder system for rate control purpose. The basic decoding unit is a macroblock (MB). For each MB, the encoded bit stream first undergoes entropy decoding to obtain the syntax bits (not shown in the figure), motion vector $\mathbf{v}$, and quantized coefficients $\overline{d}_T(t)$, where t is the time index of the image frame. Typical entropy codecs include variable length coding (VLC) and adaptive arithmetical coding (AAC). Inverse quantization is then employed to obtain the transform coefficient $d_T(t)$, which is further fed to an inverse transform module to reconstruct the pixel value or

prediction error $d(t)$, depending on whether intro- or inter-coded mode is utilized during encoding. For inter-coding mode, motion compensation is applied to generate the reference image $\bar{P}_R(t)$ using motion vector $\mathbf{v}$ and previously decoded and buffered reference image $\bar{P}(t-1)$. We use motion compensation to refer to the process of compensating the image displacement due to motion across frames. When the motion vector is of a sub-pixel value, interpolation is needed to compute the reference image. Lastly, by combining the prediction error $d(t)$ and the reference image $\bar{P}_R(t)$ the decoded image of the current frame is output.

The computational complexity of each component varies. Some are relatively constant and independent of the encoded data while others heavily depend on the coding results. For example, the components of inverse quantization and inverse transform have nearly fixed computational cost per coding unit while the motion compensation component has variable complexity depending on the block mode and the type of motion vector. Furthermore, as shown in Fig. 1, the decoder complexity is dominated by the interpolation filtering process used in motion compensation if the motion vectors are sub-pixel. Other parts of the decoding system, like entropy decoding and inverse transform, do not incur significant computational cost when compared to the interpolation process.

Note motion estimation is usually the most computationally complex process since it involves searching over a large range of possible reference locations, each of which may require interpolation filtering. Recognizing this, many fast motion estimation algorithms such as those proposed in [18][19] have been developed to reduce the motion estimation complexity during encoding. Other work proposes scalable methods for motion estimation [20] to control the coding complexity. Nevertheless these methods all focused on the encoding complexity reduction instead of the decoding complexity.

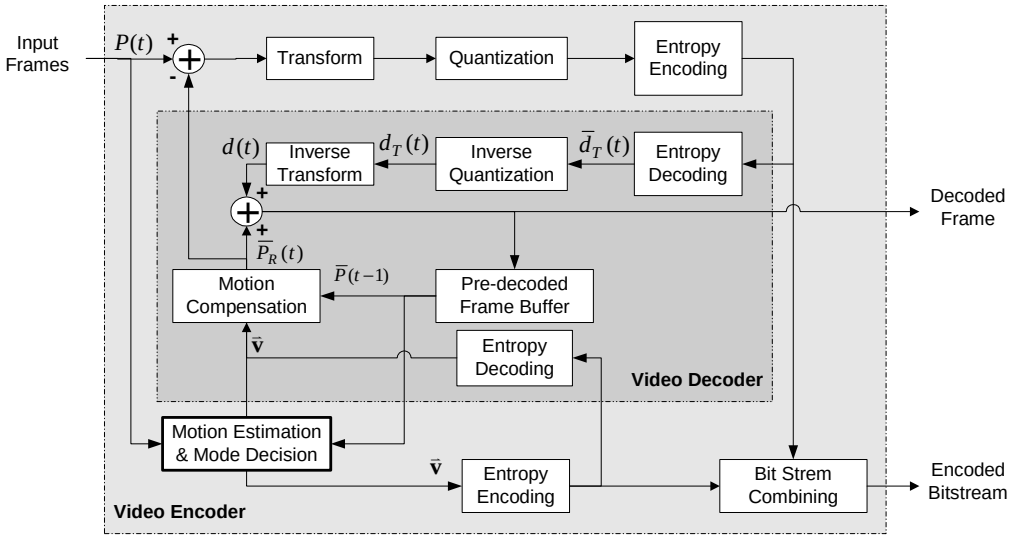


Fig. 2: Conceptual diagram for typical video coding systems

2.1 Sub-pixel interpolation

Motion estimation is one of the most important components, and also the most computationally complex part in any video coding systems. Motion estimation can be illustrated using Fig. 3. The basic idea is to search for an optimal block with similar values in previous coded frames as the reference signal for the block in current frame so that the encoding cost can be minimized. The optimal reference signal position is indicated by the displacement vector, called motion vector (denoted as V in Fig. 3). Motion estimation applies the basic idea of inter-frame predictive coding. Sometimes, multiple reference signals are used to form motion estimation, like the case for bi-directional inter-frame prediction. Motion vectors are entropy encoded in a differential and predictive manner [1]. Compared to motion estimation, motion compensation is the procedure by which the decoder extracts a reference signal from the location indicated by the motion vector. In reconstructing the reference signal, interpolation is a widely adopted technique to improve the compensation precision when the motion vector has a sub-pixel value. The effectiveness of the sub-pixel motion compensation has been verified in H.263 and subsequent coding standards, at the cost of increasing complexity (up to 50% referring to Fig. 1). Therefore reducing the motion compensation complexity becomes the most important target for improvement.

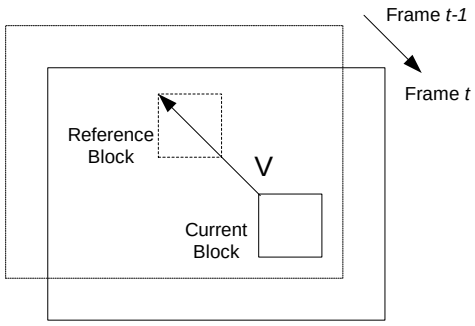


Fig. 3: Motion compensation between current and the reference frames

H.264 uses up to quarter pixel precision during interpolation [1][2]. Fig. 4 illustrates the details of this procedure, where gray blocks with capital letters indicate the integer locations and the white blocks with lowercase letters the sub pixels. All half-pixel locations undergo 6-tap FIR filtering horizontally and vertically, whenever any one applies. All quarter-pixel locations undergo 2-tap average filtering using integer and half pixels. For example, the following formulae are used to calculate sub pixel *b* and *e*:

$$b = \left(\left(E - 5F + 20G + 20H - 5I - J \right) + 16 \right) / 32$$

$$e = \left(b + h + 1 \right) / 2$$

The amount of filtering operations varies depending on the exact location of the pixel. Table 1 lists the possible interpolation operations and the associated complexity. It is

clear that different interpolation methods have quite different computing complexities. Some up-to-date video codecs may even have more complex interpolation. For example, in the recent 3D scalable video coding standard like MC-EZBC, an 8-tap floating filtering process is used to achieve high interpolation accuracy.

Given the information about the interpolation cost associated with each type of motion vectors, the basic idea of reducing the decoder complexity is to select motion vectors that involve less interpolation complexity while keeping the video quality high. Our empirical analysis of some H.264 statistical data shows that depending on the video content, 40% to 80% of motion vectors are located on sub pixels with different interpolation complexities. Therefore the principal approach to complexity reduction is to change motion vectors from high complexity sub pixel positions into the ones with low complexity, or even to integer-pixel positions.

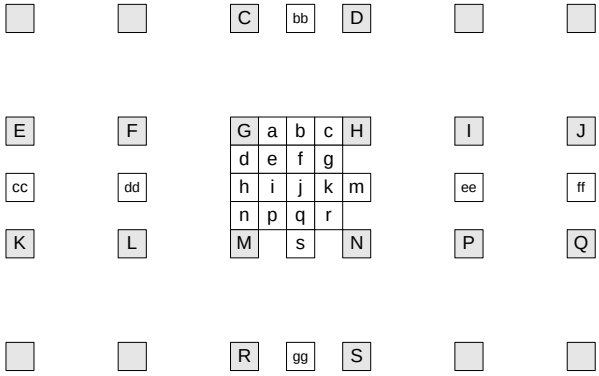


Fig. 4: Notations for sub-pixel locations in H.264

Table 1. Sub pixel locations and their interpolation complexities

Sub Pixel Type	Points	Interpolation
(0, 0)	G	No
(0, 1/2), (1/2, 0)	b, h	1 6-tap
(0, 1/4), (1/4, 0), (0, 3/4), (3/4, 0)	a, c, d, n	1 6-tap + 1 2-tap
(1/4, 1/4), (1/4, 3/4), (3/4, 1/4), (3/4, 3/4)	e, g, p, r	2 6-tap + 1 2-tap
(1/2), (1/2)	j	7 6-tap
(1/2, 1/4), (1/4, 1/2), (3/4, 1/2), (1/2, 3/4)	i, f, k, q	7 6-tap + 1 Bilinear

2.2 Block mode

In order to further reduce the temporal redundancy and improve the efficiency of motion estimation, H.264 defines a diverse set of block mode options. Besides the conventional modes of {intra, forward, backward, bi-directional}, two new important modes are introduced: variable block size and SKIP/DIRECT.

Firstly, unlike earlier coding standards using a fixed block size (usually 16x16 or 8x8) during motion estimation, H.264 allows to partition an MB into into several blocks with variable block size, ranging from 16 pixels to 4 pixels in each dimension. The possible modes of different block sizes are shown in Fig. 5. An MB can comprise up to 16 blocks.

Each block with reduced size can have its individual motion vectors to estimate the local motion at a finer granularity. Though such finer block sizes incur overhead such as extra computation for searching and extra bits for coding the motion vectors, they allow more accurate prediction in the motion compensation process and consequently the residual errors can be considerably reduced, which are usually favorable for the final rate-distortion performance.

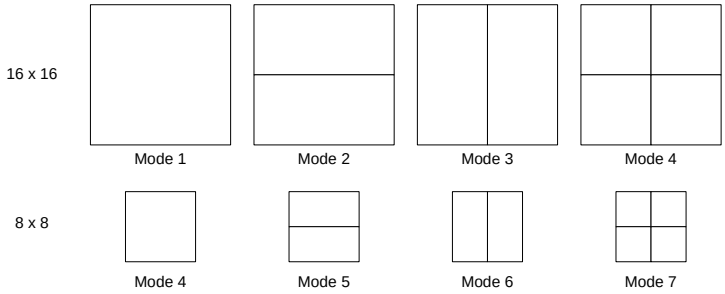
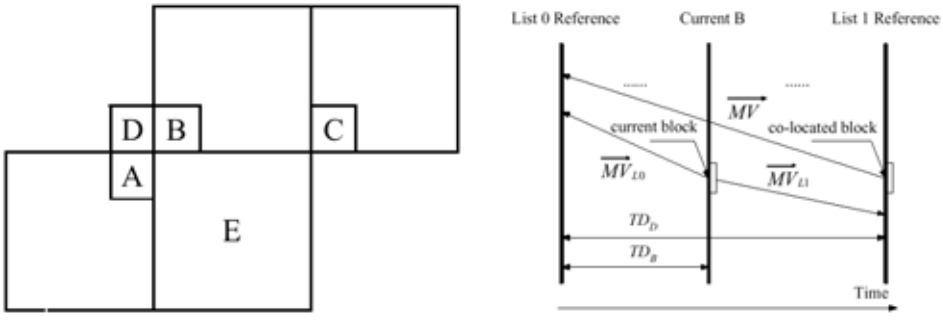


Fig. 5: Modes of variable block sizes in H.264

Secondly, the SKIP/DIRECT mode is utilized for the P/B frame in H.264 motion compensation to further increase the coding efficiency. The basic idea is to use the spatial/temporal neighbor motion vectors to predict the motion vector of the current block, without sending extra bits to encode the current motion vector. Fig. 6 (a) illustrates the SKIP mode, where the motion vectors of blocks A, B, C and D (if available) may be used to estimate the motion vector of MB E. In Fig. 6 (b) the motion vector of the current block in a B frame is interpolated from the motion vector of the co-located block from the adjacent frames, assuming a constant global motion. Details regarding the SKIP/DIRECT mode can be found in [1][4]. In our mode decision algorithm to be described later, both the variable-size block mode and the SKIP/DIRECT mode are considered during the search process.



(a) SKIP (b) DIRECCT

Fig. 6: the SKIP/DIRECT mode for the P/B frame in H.264

The selection of block mode has direct impact on the decoder computational complexity, because it determines what kind of motion vectors is recorded in the bit stream. Optimal

selection of the block mode and the associated motion vectors is the main problem addressed in our work, where a systematic solution is derived.

2.3 Motion vector searching and block mode selection

As introduced in Section 1, conventional video coding systems encode the video bit stream by solving the optimization problem defined in Equation (1). The main control variables **P** involved in this procedure include motion vector **V**, block mode M and quantization parameter QP. There is complex interaction between the choices of these variables and thus finding the optimal solution is difficult. In practice, compromised approaches are taken and approximate solutions are developed. For example, typically QP is determined through some empirical models and updated throughout the video sequence by some rate control algorithms. Given QP, the other variables, motion vector and block mode, are decided by applying some rate-distortion optimization process. An excellent survey of these procedures is described in [5]. We present a brief summary in the following.

Specifically, for each block B with a block mode M, the motion vector associated with the block is selected through a rate-distortion joint cost function [5]:

$$\mathbf{V}^*(B,M) = \arg\min_{\mathbf{V}\in\sup\{\mathbf{V}\}} J^{R,D}_{MOTION}(\mathbf{V}|B,M) = \arg\min_{\mathbf{V}\in\sup\{\mathbf{V}\}} \left\{ D_{DFD}(\mathbf{V}|B,M) + \lambda_{MOTION} R_{MOTION}(\mathbf{V}|B,M) \right\} \quad (3)$$

where $\mathbf{V}^*$ is the optimal motion vector, sup{**V**} defines the search space, whose dimensions include the prediction direction, the reference frame list and the search range. R_{MOTION} is the estimated bit rate to record the motion vector. D_{DFD} represents the prediction error between the current block and the reference block. Usually the sum of absolute difference (SAD) is adopted because the search space of motion vector is much larger than that of mode and SAD has lighter computation cost compared with the sum of squared difference (SSD). $J^{R,D}_{MOTION}(\mathbf{V})$ is the rate-distortion joint cost comprising of R_{MOTION} and D_{DFD}. λ_{MOTION} is the Lagrange multiplier to control the weight of the bit rate cost, relative to the signal distortion caused by the prediction error.

In a similar manner the block mode M for an MB is decided by the following.

$$M^*(MB,QP) = \arg\min_{M\in\sup\{M\}} J^{R,D}_{MODE}(M|MB,QP) = \arg\min_{M\in\sup\{M\}} \left\{ D_{REC}(M|MB,QP) + \lambda_{MODE} R_{REC}(M|MB,QP) \right\} \quad (4)$$

where M^* is the optimal block mode, and $\sup\{M\}$ is the set of block mode options (such as INTRA, SKIP, DIRECT, FORWARD, BACKWARD, BIDIRECTION, etc). A full list of block mode options in H.264 can be found in [4]. D_{REC} is the SSD between the current MB and the reconstructed one through motion compensation. R_{REC} is the estimated bit rate associated with mode M. $J^{R,D}_{MODE}(M)$ is the joint cost comprising of rate R_M and

distortion D_M, and λ_{MODE} is the Lagrange multiplier. The motion vectors associated with the optimal block mode $\mathbf{V}^*\left(B, M^*\right)$ will be the final coded data recorded in the bit stream.

The Lagrange multipliers used in the above two cost functions determine the relative weights between signal quality and bit rate. To simply the search process, an empirically derived relationship as the following is typically used in practice. The square root relationship is partly due to the fact that SAD is used in modeling D_{DFD} while SSD is used for D_{REC}.

$$\lambda_{MOTION} = \sqrt{\lambda_{MODE}} \tag{5}$$

2.4 Rate control

Rate control (RC) is the procedure of adjusting CVs so that the target rate requirement can be achieved while optimizing the overall video quality. Given a target bit rate, we can compute the average allocated bit rate for each basic coding unit. Then we can use the Lagrange optimization method to find the optimal set of control variables. However, searching over the entire variable space is very complex. In practice, most implementations use empirical models to restrict the search space. For example, a popular method, called rate-quantization modeling, maps the target bit rate to the quantization parameter, from which the Lagrange multipliers are decided. In addition, since coding of a data unit may not result in a bit rate that exactly matches the target, a separate process, called buffer management, is used to monitor the available bit rate budget for the remaining data units and thus update the allocated recourse. We briefly review these processes in the following.

Rate-Quantization (RQ) model describes the relationship between QP and the bit rate. A widely adopted quadratic RQ model is [6]:

$$R = D\left(P_1 \cdot QP^{-1} + P_2 \cdot QP^{-2}\right) \tag{6}$$

where D is the source complexity of the video signal, and usually measured using the motion estimation prediction errors (such as SAD), and $\{P_1, P_2\}$ are model parameters. Some systems use $P_2 = 0$ for simplicity. A typical RQ modeling procedure involves two major steps: model estimation and QP prediction. Fig. 7 shows a conceptual illustration of these procedures. Firstly several basic coding units are coded using some preset QP values. The coding units may include a certain number of MBs or one whole frame. The resulting rate-quantization-distortion (R-Q-D) points are collected, as indicated by the gray circles in Fig. 7. The model in Equation (6) is then estimated based on the observations. The estimated model is indicated by the multiple curves shown in Fig. 7. The estimated model can then be used to determine the QP value for the next coding unit based on the target bit rate R_t and source complexity D_t for the new unit. The former is decided by the buffer management process to be described below, and the latter is predicted using previous observations of the source complexity. Usually the source

complexity is assumed to vary gradually and can be estimated using some simple relationship such as a linear model. Once coding of the new unit is completed, new observations of the R-Q-D points are collected and used to update the estimation of the RQ model in a sliding window manner. Namely, the oldest R-Q-D point is purged and the latest point is added to update the model.

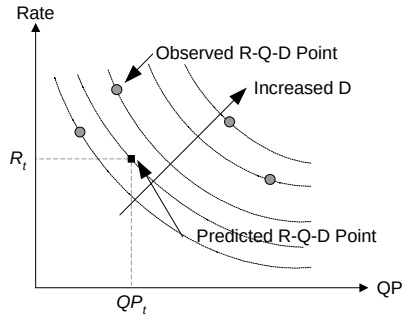


Fig. 7: Rate-Quantization model estimation and QP prediction in the rate control process

The buffer management employs a virtual buffer to simulate the behavior of the data buffer on the decoder side. It is an important component in rate control in order to adjust the target bit rate for each coding unit and avoid the problem of buffer overflow or underflow. For example, given a target bit rate for the video sequence, the average bit rate allocation for each Group of Pictures (GOP) can be computed, and the allocated bit rate, R_t, for a new frame to be coded (such as P frame) can be determined by monitoring the actual number of bits spent on the previous frames.

In H.264, given the target rate and QP for the coding unit, the following empirical relationship is often used to determine the Lagrange multiplier needed in the rate-distortion tradeoff optimization.

$$\lambda_{MODE} = 0.85 \times 2^{\frac{QP-12}{3}} \tag{7}$$

The validity of such a model is justified by empirical simulations, though some analytical explanations have been offered in the literature such as [5]. Such an empirical model is very useful in simplify the search process in the Lagrange optimization method, while practical implementations have often shown satisfactory performance. Other parameters like λ_{MOTION} can also be found according to Equation (5).

3. Complexity adaptive motion estimation and mode dcecision

We propose a new system for Complexity-Adaptive Motion Estimation and mode Decision (CAMED). Given defined metrics for signal distortion and computational complexity, the CAMED method explores the tradeoff between video quality and resource consumption (both bit rate and computational complexity) to approximate the optimal motion vectors and block mode used in the motion compensation process in the decoder. The CAMED system consists of several components: the rate-distortion-

complexity (R-D-C) joint optimization framework, the complexity cost function, and the complexity control algorithm. The R-D-C framework extends the previously discussed Lagrange optimization framework to incorporate the complexity term. The complexity cost function provides quantitative measurements of the required computation for each motion vector type. The complexity control algorithm is used to control the complexity over different coding units to meet the overall target complexity. We discuss each of them in the following.

3.1 The Rate-Distortion-Complexity optimization frameowrk

Our proposed CAMED system basically tries to solve the problem defined in Equation (2), with an explicit Lagrange term to model the complexity cost. Therefore, the motion vectors are selected through a rate-distortion-complexity joint cost function:

$$\mathbf{V}_C^*(B,M) = \arg\min_{\mathbf{V}\in\sup\{\mathbf{V}\}} J_{MOTION}^{R,D,C}(\mathbf{V}|B,M) = \arg\min_{\mathbf{V}\in\sup\{\mathbf{V}\}} \left\{ J_{MOTION}^{R,D}(\mathbf{V}|B,M) + \gamma_{MOTION} C_{MOTION}(\mathbf{V}|B,M) \right\} \quad (8)$$

where C_{MOTION} is the complexity cost function associated with the selected motion vector $(\mathbf{V}|B,M)$, γ_{MOTION} is the Lagrange multiplier for the complexity term, $J_{MOTION}^{R,D}(\mathbf{V})$ is the rate-distortion joint cost function defined in Equation (3), and $J_{MOTION}^{R,D,C}(\mathbf{V})$ is the rate-distortion-complexity joint cost function.

Similar to the earlier case described in Equation (4), the block mode search process is guided by the following.

$$M_C^*(MB,QP) = \arg\min_{M\in\sup\{M\}} J_{MODE}^{R,D,C}(M|MB,QP) = \arg\min_{M\in\sup\{M\}} \left\{ J_{MODE}^{R,D}(M|MB,QP) + \gamma_{MODE} C_{MODE}(M|MB) \right\} \quad (9)$$

where C_{MODE} is the complexity cost function associated with the block mode, γ_{MODE} is the Lagrange multiplier, $J_{MODE}^{R,D}(\mathbf{V})$ is the rate-distortion joint cost function defined in (4), and $J_{MODE}^{R,D,C}(\mathbf{V})$ is the rate-distortion-complexity joint cost function.

Now consider two extreme cases. When $\gamma_{MODE} = 0$, the solutions of Equations (8) and (9) are identical with the ones in Equations (3) and (4), namely no consideration was given to the complexity constraint and many motion vectors may be of sub-pixel values in order to minimize the distortion. When $\gamma_{MODE} = \infty$, all motion vectors are forced to integer pixel locations in order to minimize the complexity involved in interpolation for sub-pixel locations. Clearly there is a tradeoff between these two extremes to balance the performance in terms of quality and complexity.

For simplification, we can adopt restrictions like that described in Equation (5) to limit the search space. For example, in our experiments to be described later, we use the following relationship to link γ_{MODE} and γ_{MOTION}.

$$\gamma_{MOTION} = \sqrt{\gamma_{MODE}} \tag{10}$$

3.2 Complexity cost function

In the joint cost function described above, we need a quantitative model to estimate the complexity associated with each candidate motion vector and block mode. As discussed in Section 2.1, the computational complexity is heavily influenced by the type of the motion vector (integer, half-pixel, or quarter-pixel) and the interpolation filters used in the motion compensation process. If we just focus on the interpolation filtering cost, quantitative estimates of such complexities can be approximated by the number of filtering operations needed in interpolation, such as those listed in Table 1. For example, using the same 6-tap filter and 2-tap filter implementations, the complexity of each motion vector type is as follows.

$$c_B(\mathbf{V}) = N_B \cdot c_P(\mathbf{V}) \tag{11}$$

$$c_P(\mathbf{V}) = \begin{cases} 0 & V \text{ is integer MV} \\ e_6 & V \text{ is subpixel b,h} \\ e_6 + e_2 & V \text{ is subpixel a,c,d,n} \\ 2e_6 + e_2 & V \text{ is subpixel e,g,p,r} \\ 7e_6 & V \text{ is subpixel j} \\ 7e_6 + e_2 & V \text{ is subpixel i,f,k,q} \end{cases} \tag{12}$$

where $c_B(\mathbf{V})$ is the computational cost for the current coding block, V is the motion vector, $c_P(\mathbf{V})$ is the computational complexity required for calculating a reference pixel, N_B is the number of pixels in the current coding block, and $\{e_6, e_2\}$ are the estimated complexities for 6-tap and 2-tap interpolation respectively. Our experiment later will show that a simplified model ignoring the 2-tap interpolation will mostly result in the same selection of the motion vectors. With such simplification, the above model becomes the following with a common factor e_6 removed.

$$c_P(\mathbf{V}) = \begin{cases} 0 & \text{integer MV} \\ 1 & \text{subpixel a,b,c,d,h \& n} \\ 2 & \text{subpixel e,g,p,r} \\ 7 & \text{subpixel i,j,f,k,q} \end{cases} \tag{13}$$

The estimated cost of interpolation can be derived from the specific software or hardware implementations. For example, each interpolation operation can be divided into a number of basic operators such as addition, shifts, and/or multiplications. In this case, $\{e_6, e_2\}$ can be modeled with more details such as the following.

$$e_i = \sum_j \rho N(o_j) P(o_j), i = 2, 6 \tag{14}$$

where o_j is the basic operator involved in the interpolation implementation, $N(o_j)$ is the required number of operator o_j, $P(o_j)$ is the power consumption of operator o_j, and $\rho \geq 1$ is the adjustment factor to consider additional power cost such as memory access. For instance, Fig. 8 shows a hardware implementation of interpolation that was introduced in [7]. Its estimated complexity is

$$e_6 = \rho\left(6P_{add} + 2P_{shift}\right) \tag{15}$$

where P_{add}, P_{shift} are the power consumption for the addition operator and the 2-bit shift operator respectively.

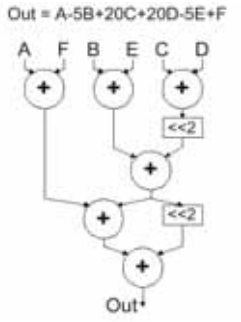


Fig. 8: A hardware implementation for the interpolation unit

Each block may be associated with multiple reference blocks, each of which needs a motion vector. For example, for bi-directional prediction, each block may need two motion vectors for forward and backward prediction respectively. Thus, the computational cost for a block B with the block mode *M* is calculated as:

$$C_{MOTION}\left(\mathbf{V}|B,M\right) = \sum_j \left(c_B\left(\mathbf{V}_j, M, B\right)\right) \tag{16}$$

where the summation is over each reference block.

Each MB may consist of several smaller blocks, depending on the block mode, *M*. The overall computational cost associated with a MB and a block mode can be calculated as:

$$C_{MODE}\left(M|MB\right) = \sum_i \sum_j \left(c_B\left(B_i, \mathbf{V}_j, MB\right)\right) \tag{17}$$

where *i* is the index of the individual blocks contained in the MB, and *j* is the index for multiple motion vectors associated with a single block. Equation (16) and (17) are generic and applicable to all inter-coded block modes, including foreword/backward/bi-directional motion compensation and SKIP/DIRECT.

3.3 Complexity control

Complexity control, analogous to the rate control process described in Section 2.3, is a process to allocate the complexity resource among the coding units and to determine parameters like Lagrange multiplier γ_{MODE} to be used in the optimization procedure. In Section 2.3, the allocated bit rate is mapped to the quantization parameter, which in term is used to find the Lagrange multiplier λ_{MODE}. In this section, we describe two components of the complexity control algorithm – the complexity modeling and the buffer management. The former is used to characterize the relationship between the target complexity and the Lagrange multiplier γ_{MODE}. The latter is for monitoring the complexity usage and updating the available computational resource for each new data unit.

3.3.1 Complexity modeling

In complexity control a feasible modeling of complexity and control parameter (γ_{MODE} in our case) is necessary. So far there is very little knowledge regarding the statistical properties of the computational complexity in H.264. Therefore, similar to the practical solutions used in most rate control algorithms, we resort to empirical observations from experimental simulations.

One of the objectives is to find the relationship between the target complexity and the optimization control parameter, γ_{MODE}. Fig. 9 shows some simulations results revealing such relationship. The details of the experiment will be described later in this paper. The results indicate there is an approximately linear relationship between the complexity value and log of the Lagrange multiplier. It is also clear the type of the frame (B or P) influences greatly the relationship.

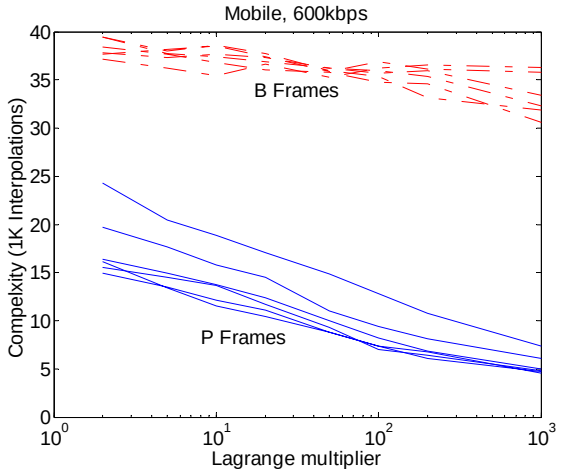


Fig. 9: Relationship between Lagrange multiplier and the resulting complexity (top: B frames, bottom: P frames)

A reasonable model based on the above observations is as follows.

$$C(\gamma_{MODE}) = D(K_1 \ln(\gamma_{MODE}) + K_0) \tag{18}$$

where C is the complexity, D is a factor measuring the video source complexity similar to that used in Equation (6) for rate control. K_1, K_0 are the model parameters that needed to be learned during the coding procedure. Due to different coding mechanism, P and B frames will have distinguished model parameters and need to be handled separately.

Though lacking a theoretical explanation, the above model is driven by the empirical simulation observations. The linear dependence of the computational complexity on the signal source complexity is also intuitive – the more complex the signal source is, the higher accuracy is needed in estimating the motion vector and thus there is a larger gain in using sub-pixel motion vectors, resulting in an increased computational cost. Fig. 10 shows the approximate linear relationship between the computational complexity and the mean prediction error (measured in mean absolute difference, MAD) from our simulation (details to be described in later sections). Like the previous case of rate control, the MAD measure can be considered as approximate estimation of the signal source complexity.

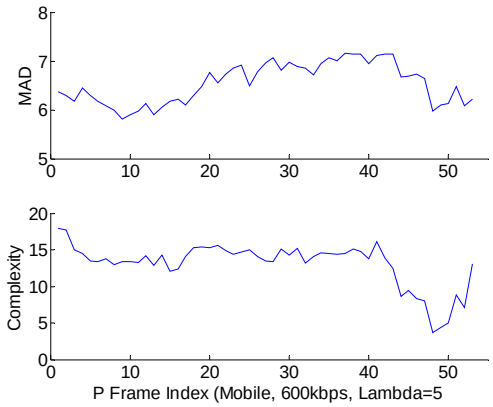


Fig. 10: The relationship between the computational complexity and the signal source complexity

Using the above model, the Lagrange multiplier $\gamma_{MODE}(t)$ for the current coding unit t can be determined by the following.

$$\gamma_{MODE}(t) = \exp\left\{\frac{C(t) - K_0 D(t)}{K_1 D(t)}\right\} \tag{19}$$

where *C(t)* is the allocated computational budget and $D(t)$ is the predicted complexity measurement for unit t. In practice, in order to avoid large quality fluctuation, the change rate of $\gamma_{MODE}(t)$ is bounded by some thresholds.

3.3.2 Buffer Management

Complexity buffer is a virtual buffer to simulate the complexity usage status on the decoder side. It is analogous to the rate buffer used in the rate control to update the estimation of available resource and avoid issues of buffer overflow or underflow. Denote C_{GOP} the remaining complexity budget in one GOP, N_P , N_B the remaining numbers of P, B frames respectively, and η the complexity ratio between P and B, which is updated along the video coding. The target complexity levels for P, B frame C_P, C_B are calculated by solving the following equations:

$$\frac{C_B}{C_P} = \eta \tag{20}$$

$$N_P C_P + N_B C_B = C_{GOP} \tag{21}$$

Once C_P, C_B are available, $\gamma_{MODE}(t)$ is determined using the model described in the previous subsection. The formulations in Equation (20) and (21) assume the basic coding unit as one frame. It can be easily extended to smaller units for a finer granularity.

4. Experiment Results

4.1 Experiment environment

Table 2 lists the experiment environment used in our simulation. Four standard test video sequences were chosen and they had distinguished characteristics in motion intensity and texture complexity, two crucial factors influencing the motion estimation performance. H.264 reference codec of version JM82 was used. We use equation (13) to calculate the complexity cost function.

Table 2. Experiment Environment

Sequence Information	
Sequence Name	*Akiyo, Foreman, Mobile, Stefan*
Image Format	CIF (352×288 pixels)
Video Format	30 frame per second, GOP size = 15, sub GOP size = 3
IBP structure	IBBPBBPBBP…
Simulation Parameters	
Bit rate	100, 200, 400, 600, 800, 1000, 1500 *Kbps*
γ_{MODE} values	From 0 to 500 (Lambda in the figure)
H.264 Configuration (selected)	
Profile	Main
Search Range	32
Inter Search Block Mode	All On
Use Fast Motion Estimation	Off
Frame / Slice Mode	Frame Mode
Direct Mode Type	Temporal
Basic Rate Control Unit	11 Macroblocks
S-P Frame	No

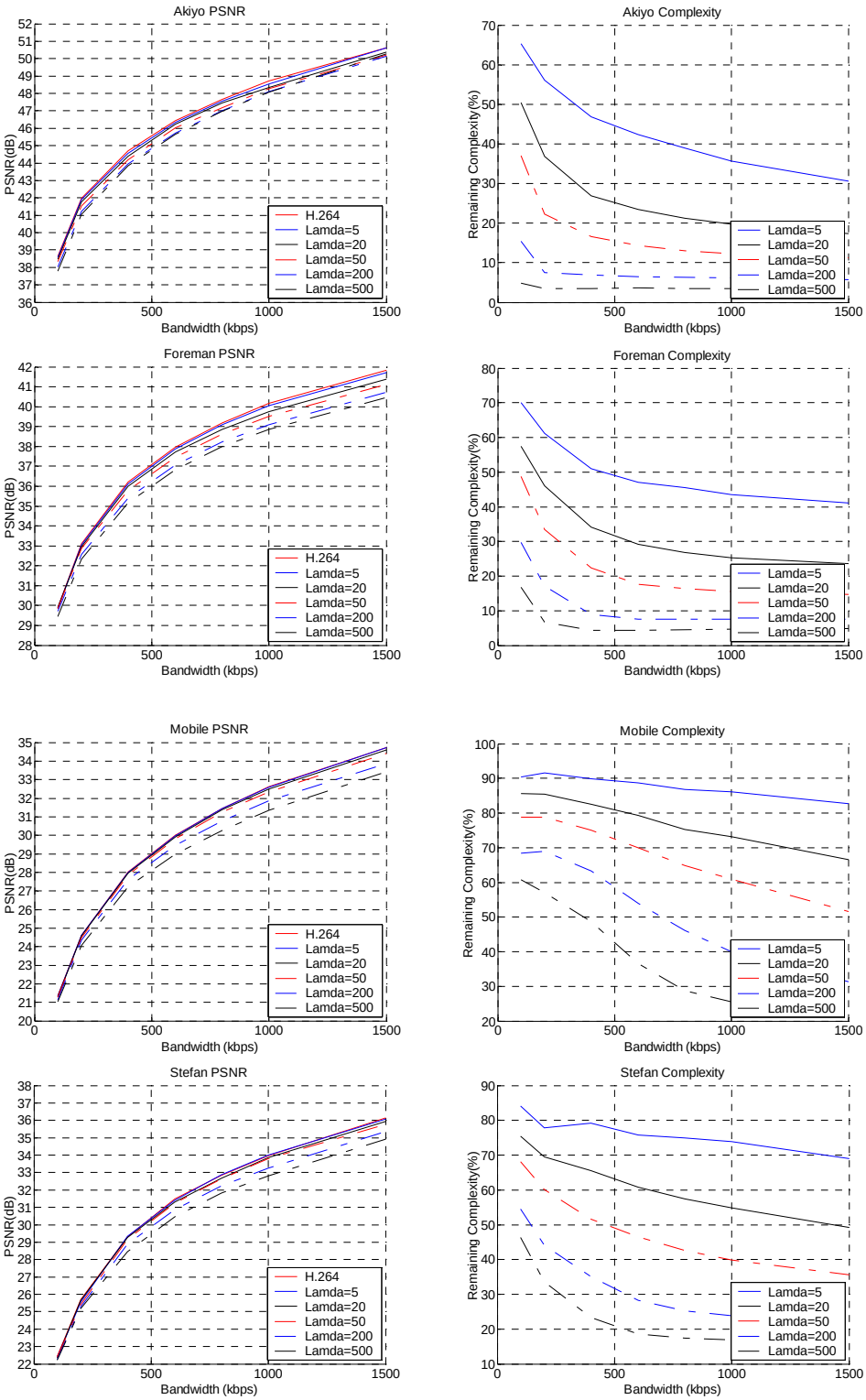


Fig. 11: Performance of rate-distortion and rate-complexity using the proposed CAMED system

4.2 Rate-distortion-complexity performance in CAMED

Fig. 11 lists the rate-distortion performance together with rate-complexity results by different γ_{MODE} values. The latter is measured in terms of the ratio of the remaining complexity when applying the proposed CAMED method to the original complexity when using the H.264 JM82 codec (i.e., $\gamma_{MODE} = 0$). Note the complexity includes only the computation required for reconstructing the reference signals in the decoder, which is the most demanding component in the decoding process.

Several important findings are in order. Firstly, adjusting γ_{MODE} is an efficient way to control the computational complexity. Up to 95% of the interpolation cost can be removed within a relatively small range of γ_{MODE} (see *Foreman* at 1000kbps with γ_{MODE} =500). Secondly, the video quality is well maintained when reducing the complexity. If we use 0.5dB as the perceptual quality difference threshold, up to 60% of the computational cost can be saved without visible impairment (see *Stefan* at 1000kbps with γ_{MODE} =50 and PSNR drop of 0.197dB). Fig. 12 further shows the frame-to-frame quality and complexity over the entire video sequence *Stefan*. In fact, for all sequences, 30~50% cost saving can be obtained within 0.1dB quality loss. According to the benchmark provided in [3], this can be translated into an overall decoding complexity saving up to 30%.

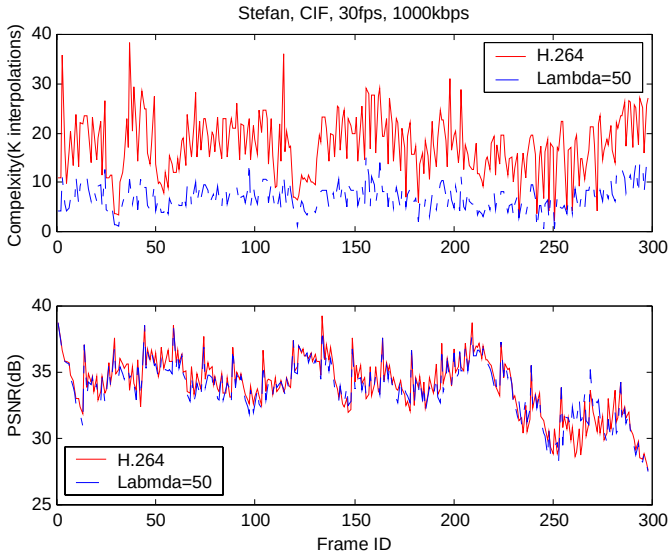


Fig. 12: Frame-to-frame video quality and computational complexity comparison. The video quality is well maintained though the complexity is greatly reduced.

The reason of the above excellent performance probably can be attributed to the statistical characteristic of video signals. Without theoretical explanation, our intuitive conjecture is that not every sub-pixel motion vector is equally important in predicting the reference signal required in the motion compensation process. Moving less critical ones

from the sub-pixel resolution to the integer resolution will not dramatically increase the prediction error, but will help significantly in reducing the computational cost at the decoder. Fig. 13 shows one example comparing the motion vector distribution with and without applying the proposed CAMED method. It is obvious that many motion vectors shift to the locations with lower interpolation complexities (integer or half-pixel locations).

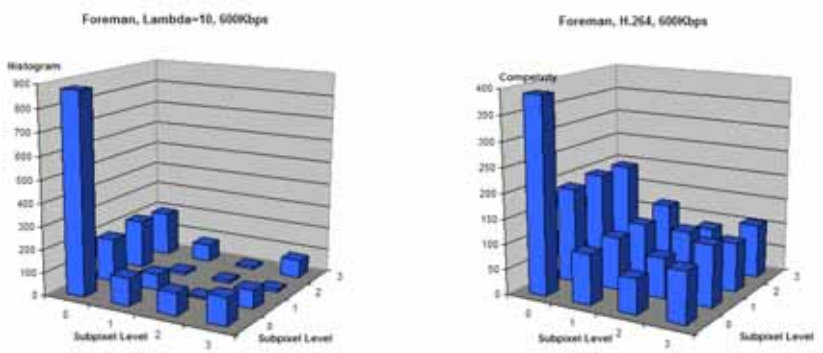


Fig. 13: Subpixel motion vector distribution with and without the proposed CAMED method

4.3 Complexity Control

During complexity control, the basic operations are to adjust γ_{MODE} and manage the complexity buffer. First of all, the complexity model presented in Equation (18) need to be verified. Fig. 9 illustrates the relationship between Lagrange multiplier and resulting complexity for the sequence *Mobile* at 600kbps. Each curve is for one P or B frame. The linear relationship between the complexity and logarithmic γ_{MODE} is evident, especially for P frames. B frames usually have larger complexity because of bi-directional prediction. For the same frame type, we hypothesize the variation is caused by different content complexity in each frame. Like the empirical approach used in the conventional rate control process, we use MAD as an approximate measure of the frame content complexity. Fig. 14 compares the frame-to-frame evolution of computational complexity with some major coding parameters for the P frames in the *Mobile* sequence at 600kbps with $\gamma_{MODE} = 5$. Compared to other options (such as quantization parameter), MAD appears to demonstrate the closest correlation with the computational though some variance is still noticeable. Study of improved measures capturing the statistical properties of the computational complexity is an interesting topic for future research.

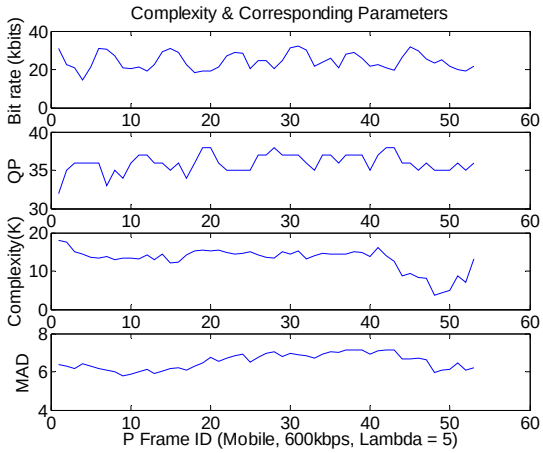


Fig. 14: Relationship between computational complexity and major coding parameters

Table 3 lists the main parameters used in the complexity control experiment. Some parameters may be further fine-tuned in order to find better tradeoff between quality fluctuation and control efficiency. We leave such potential improvement in future studies.

Table 3: Parameters used in complexity control

Basic control unit	One frame
Frame complexity prediction window size	6
Frame complexity prediction model	Linear
γ_{MODE} prediction window size	6
γ_{MODE} prediction model	Equation (19)
γ_{MODE} range	[0 3000]
Maximum λ_M^C change magnitude	80
Initial ratio of B/P frame complexity	1.6
Initial λ_M^C value	10
Target complexity per GOP (amount of interpolations, defined in Equation (13))	50K ~ 250K

Fig. 15 shows the detailed complexity control results for the sequence *Foreman* at 1000kbps with different target complexity levels. The complexity of the baseline H.264 JM82 result is also shown for comparison. The results are very promising. Though in all cases the initial γ_{MODE} is set to the same value 10, it can be adaptively adjusted and the target complexity level is consistently accomplished, except for the case aggressively reducing the complexity from about 250K to 50K (80% reduction). The latter aggressive case will not be achievable without sacrificing greatly the video quality. For other cases, some fluctuation can still be seen at the end of the sequence starting from 14th GOP, where the sequence contains rapid camera panning. Our complexity control method cannot completely smooth this huge complexity increase because we bound the maximum magnitude and the maximum change rate of the γ_{MODE} parameter in order to avoid excessive quality loss and quality fluctuation. In other words, we try to maintain some consistence in video quality throughout the entire video sequence as well.

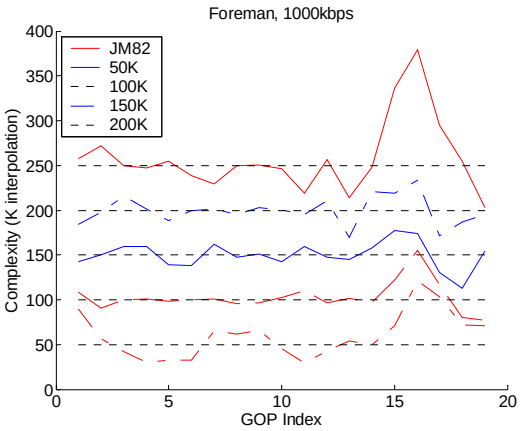


Fig. 15: Complexity control performance

Table 4 summarizes the complexity control performance at 1000kbps. Complexity control error is calculated as the difference between the actual resulting complexity and the target complexity, normalized by the target complexity. Complexity Saving is the percentage of the original computational cost that has been removed. Quality Degradation is the quality difference (in PSNR) between the bit stream generated using original H.264 and the one using our complexity control method. These results confirm that large savings of the computational complexity (30% to 60%) can be achieved with small quality degradation (0.3dB). Improvements from different video clips are different depending on the type of the content and the complexity of the signal. The most challenging case is the *Mobile* sequence, which has a steady camera motion (slowly panning left) and thus the SKIP/DIRECT mode is frequently used. It is difficult for the proposed CAMED method to change the motion vector to the integer one without incurring significant increase of bit rate. This is partly because that the SKIP/DIRECT mode is already a very efficient coding mode – very few bits are needed in coding the mode information and changing the mode will not improve much the prediction accuracy due to the motion in the video content. However, even for such a challenging case, our proposed CAMED method can still achieve about 33% complexity saving in order to keep the video quality more or less intact.

Table 4: Complexity control performance (at 1000kbps)

Target Complexity		50K	100K	150K	200K	250K
Foreman	Complexity control error	19.78%	2.81%	0.09%	0.2%	1.4%
	Complexity Saving	76.78%	60.15%	41.80%	22.63%	4.45%
	Quality Degradation (dB)	0.60	0.28	0.13	0.06	0.02
Stefan	Complexity control error	61.43%	1.73%	1.16%	0.14%	4.37%
	Complexity Saving	67.68%	59.27%	39.25%	20.04%	4.28%
	Quality Degradation (dB)	0.78	0.60	0.35	0.05	-0.09
Mobile	Complexity control error	195.8%	47.92%	1.37%	5.24%	11.59%
	Complexity Saving	47.92%	47.92%	47.91%	33.27%	22.18%
	Quality Degradation (dB)	0.63	0.63	0.63	0.31	0.13

5. Conclusions and Future Work

We introduce a novel complexity adaptive motion estimation and mode decision method (called CAMED) to improve the emerging hybrid video encoding system like H.264 so that the computational resource required at the decoder can be greatly reduced while the video quality is maintained with very little degradation. Such results are very useful for the increasingly popular handheld devices in many mobile applications.

We first analyze the decoding complexity behavior and identify the most critical components, i.e., the motion vector and the block mode that affect the cost of the interpolation process in motion compensation. We develop simple but practical cost functions to estimate the required computation for each motion vector and block mode. Then we extend the conventional rate-distortion optimization framework based on the Lagrange multiplier method to explicitly handle the computational complexity. In addition, for complexity control, we propose an effective logarithmic-linear model to predict the relationship between the target complexity and the Lagrange multiplier. The joint rate-distortion-complexity framework together with the complexity control algorithm provides an effective solution for optimizing the tradeoff between video quality and resources, including both bit rate and computational complexity. The proposed system can be easily embedded into existing video coding systems and will work with any standard compatible decoder.

Our extensive experiments using H.264 codec over different video sequences, different bit rates, and different complexity levels demonstrate the power of the proposed system. Up to 60% of the interpolation complexity can be saved at the decoder without incurring noticeable quality loss (within 0.2 dB). Even for challenging video clips such as *Mobile*, 33% of the complexity can be reduced with quality difference less than 0.3dB. The proposed complexity control scheme can reliably meet the target complexity requirement for a wide range of video content.

The proposed system has great potential in realizing an efficient low-power video decoder product. There are several interesting topics that will benefit further investigation. First, in practice, many video encoder implementations utilize some fast motion estimation procedures to reduce the power consumption on the encoder side. An interesting topic is to study how the proposed technique will affect the video quality and computational complexity when such fast encoder implementations are applied. Secondly, the computational complexity may not provide a reliable estimate about the final power consumption of the entire decoder system. There are other factors involved, such as chip-level issues, hardware architecture, memory access, etc. Although the interpolation procedure has been identified to be the most significant one in the decoder, it will be important to conduct a more extensive study of the impact on the real power consumption. Using the limited models available in the literature, we conjecture the actual power consumption saving by our current method may be in the range between 10% and 30%. Thirdly, several components of the proposed framework, such as the complexity modeling, are not fully optimized and present interesting opportunities for further improvement.

Reference:

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard," IEEE Trans. Circuits Syst. Video Technol., vol 13, pp.560-576. Jul. 2003.

[2] T. Wedi; H.G. Musmann, Motion- and aliasing-compensated prediction for hybrid video codingPage(s): IEEE Trans. Circuits Syst. Video Technol., vol 13, pp.577- 586. Jul. 2003.

[3] V. Lappalainen, A. Hallapuro, and T.D. Hämäläinen, "Complexity of OptimizedH.26L Video Decoder Implementation," IEEE Trans. Circuits Syst. Video Technol., vol 13, pp. 717-725. Jul. 2003

[4] A. M. Tourapis, F. Wu, S. Li, "Direct mode coding for bi-predictive pictures in the JVT standard", ISCAS2003, vol. 2, 700-703, Thailand, 2003.

[5] G. J. Sullivan and T. Wiegand, Rate-Distortion Optimization for Video Compression IEEE Signal Processing Magazine, Vol. 15, Num. 6, pp. 74-90, Nov. 1998

[6] T. Chiang and Y.-Q. Zhang, "A New Rate Control Scheme Using Quadratic Rate Distortion Model," IEEE Trans. Circuits Syst. Video Technol., Vol. 7, pp. 246-250, Feb. 1997

[7] T.-C. Chen, Y.-C. Huang and L.-G. Chen, "Full Utilized and Resuable Architecture for Fractional Motion Estimation of H.264/AVC", ICASSP2004, Montreal, Canada, May 17-21, 2004

[8] X. Zhou, E. Li, and Y.-K. Chen, "Implementation of H.264 Decoder on General-Purpose Processors with Media Instructions", in Proc. of SPIE Visual Communications and Image Processing, Jan. 2003

[9] National's PowerWise™ technology. http://www.national.com/appinfo/power/powerwise.html

[10] Y. Eisenberg, C. E. Luna, T. N. Pappas, R. Berry, A.K. Katsaggelos, Joint source coding and transmission power management for energy efficient wireless video communications, CirSysVideo(12), No. 6, June 2002, pp. 411-424.

[11] Q. Zhang, W. Zhu, Zu Ji, and Y. Zhang, "A Power-Optimized Joint Source Channel Coding for Scalable Video Streaming over Wireless Channel", IEEE International Symposium on Circuits and Systems (ISCAS) 2001, May, 2001, Sydney, Australia.

[12] X. Lu, E. Erkip, Y. Wang and D. Goodman, "Power efficient multimedia communication over wireless channels", IEEE Journal on Selected Areas on Communications, Special Issue on Recent Advances in Wireless Multimedia, Vol. 21, No. 10, pp. 1738-1751, Dec., 2003

[13] H. Kim and Y. Altunbasak, "Low-complexity macroblock mode selection for the H.264/AVC encoders," IEEE Int. Conf. on Image Processing, Suntec City, Singapore, October 2004

[14] A. Ray and H. Radha, "Complexity-Distortion Analysis of H.264/JVT Decoder on Mobile Devices," Picture Coding Symposium (PCS), December 2004

[15] K. Lengwehasatit and A. Ortega, " Rate Complexity Distortion Optimization for Quadtree-Based DCT Coding ",ICIP 2000, Vancouver,BC, Canada, September 2000.

[16] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-Rate-Distortion Analysis for Wireless Video Communication under Energy Constraints," IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Integrated Multimedia Platforms, 2004.

[17] B. Girod, A. Aaron, S. Rane and D. Rebollo-Monedero , "Distributed video coding," *Proc. of the IEEE*, Special Issue on Video Coding and Delivery, 2005.

[18] A. M. Tourapis. "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation," Proceedings of Visual Communications and Image Processing 2002 (VCIP-2002), San Jose, CA, Jan 2002, pp. 1069-79.

[19] H.-Y. Cheong, A. M. Tourapis, "Fast Motion Estimation within the H.264 codec," in proceedings of ICME-2003, Baltimore, MD, July 6-9, 2003

[20] M. Schaar, H. Radha, Adaptive motion-compensation fine- granular-scalability (AMC-FGS) for wireless video, IEEE Trans. on CSVT, vol. 12, no. 6, 360-371, 2002.