COMPLEXITY ADAPTIVE H.264 ENCODING FOR LIGHT WEIGHT STREAMS

Yong Wang, Shih-Fu Chang

Digital Video | Multimedia Group, Dept. of Electrical Engineering, Columbia University
1312 Mudd, 500 W 120 Street New York, NY 10027

ABSTRACT

Emerging video coding standard H.264 achieves significant efficiency improvements, at the expense of greatly increased computational complexity at both the encoder and the decoder. Most prior works focus on reducing the encoder complexity only. In this paper, we develop a novel approach to reduce the decoder complexity without changing any implementation of standard-compliant decoders. Our solution, called complexity adaptive motion estimation and mode decision (CAMED), involves several core components, including a rigorous rate-distortion-complexity optimization framework, complexity cost modeling, and a complexity control algorithm. Such components are incorporated in the encoder side. Experiments over diverse videos and bit rates have shown that our method can achieve significant decoding complexity reduction (up to 60% of interpolation computation) with little degradation in video quality ($\leq$ 0.2dB) and very minor impact on the encoder complexity. Results also show that our method can be readily combined with other methods to reduce the complexity at both encoder and decoder.

1. INTRODUCTION

H.264 is the latest ITU-T/MPEG joint video coding standard [1]. It incorporates a set of advanced techniques, such as variable block size motion estimation and context adaptive binary arithmetic coding [1]. As a consequence H.264 achieves significant improvement in rate-distortion (R-D) efficiency compared to existing standards at the cost of considerably increased computational complexity [2]. To address this problem, many works have been proposed for complexity reduction, mostly focusing on either the encoder algorithm simplification or hardware design optimization. Among those, Tourapis extended the EPZS (Enhanced Predictive Zonal Search) algorithm to reduce the motion estimation (ME) cost [3]. Yin *et al* proposed a fast mode decision method to speed up ME procedure [4]. Zhou *et al* implemented H.264 decoders based on Intel's single-instruction-multiple-data (SIMD) architecture to reduce the decoding complexity of H.264 [5].

In this paper, we focus on algorithmic-level solutions to reduce the computational cost at the H.264 decoders. Our effort has been motivated by the fact that a large number of mobile or handheld devices begin to support video applications. However, playback of complex video streams (such as H.264) on such devices still requires a lot of computations [2]. Algorithmic solutions for reducing computational cost at the decoder will nicely complement other solutions for simplifying encoder-side cost and other solutions based on hardware-level approaches. Specifically, we define our problem as follows - how to modify the non-normative part of H.264 encoder algorithm so that the encoded video streams can be decoded with much less computational cost by any standard-compliant decoder, while keeping the video quality degradation (if any) minimal. Our solution, called complexity adaptive motion estimation and mode decision (CAMED), provides a systematic method to effectively reduce the cost of the most expensive component at the algorithm level. Our system includes several novel components: a rigorous rate-distortion-complexity (RDC) optimization framework, complexity cost modeling, and a complexity control algorithm. The experiments over different video contents and bit rates have confirmed the excellent performance of the proposed solution - saving of up to 60% of interpolation operations within 0.2dB of video quality difference. When combined with other methods intended for encoder cost reduction, our solution also demonstrates very promising performance.

The remainder of this paper is organized as follows. Section 2 provides an overview on motion estimation and mode decision in H.264. Section 3 introduces our proposed CAMED framework. Section 4 reports the experiment results and analysis. The conclusion is given in Section 5.

2. MOTION ESTIMATION AND MODE DECISION

2.1. Sub-pixel interpolation

We discuss the most complex operation, sub-pixel interpolation, in H.264 decoder in this section. H.264 applies up to quarter pixel precision to enhance the performance of the motion estimation (ME) and motion compensation (MC) processes. The reference blocks located at sub-pixels are obtained through interpolation [1]. Figure 1 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/or vertically. All quarter-pixel locations undergo 2-tap average filtering using integer and half pixels. 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 different computing complexities. According to the benchmark data in [2], interpolations dominate the computational complexity (up to 50%) of H.264 decoding. Therefore, reducing the interpolation amount (especially 6-tap filtering) can decrease the decoding complexity. Our statistical analysis shows that 40% to 80% of motion vectors are located on sub pixels with different interpolation complexities. Therefore one intuitive idea for 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. However, doing so may adversely affect the ME performance and thus the final video quality. Therefore, we need a systematic framework for intelligently determining the types of the motion vectors in order to optimize the tradeoff between quality, bitrate, and computational complexity.

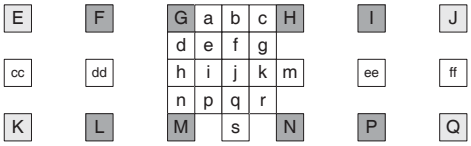

Fig. 1. Notations for sub-pixel locations in H.264.

Table 1. Sub pixel locations and their interpolation complexities

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

2.2. Block mode

H.264 defines a diverse set of block mode options. Besides the conventional modes of intra, forward, backward, bi-directional, two more important modes are introduced. First, H.264 allows to partition an macroblock (MB) into several blocks with variable block size, ranging from 16 pixels to 4 pixels in each dimension. An MB can comprise up to 16 blocks. Each block can have its individual motion vectors. Second, the SKIP/DIRECT mode is utilized for the P/B frame 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. Details regarding the SKIP/DIRECT mode can be found in [1].

Block mode has direct impact on the decoder computational complexity, because it determines what kind of motion vectors is recorded in the bit stream. R-D framework is usually conducted in motion estimation and mode decision [6]. 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.

3. COMPLEXITY ADAPTIVE MOTION ESTIMATION AND MODE DECISION

Given defined metrics for signal distortion and computational complexity, CAMED explores the tradeoff between video quality and resource consumption (both bit rate and computational complexity at the decoder) to determine the optimal motion vectors and block mode used in the MC process in the decoder. CAMED consists of several components: the RDC joint optimization framework, the complexity cost modeling, and the complexity control algorithm.

3.1. The Rate-Distortion-Complexity optimization framework

In CAMED, the optimal motion vector $\mathbf{V}^*$ for each block B is selected through a RDC joint cost function and the Lagrange multiplier method:

$$\mathbf{V}^*(B,M) = \arg\min_{\mathbf{V}}\{J^{R,D}_{MOTION}(\mathbf{V}|B,M) + \gamma_{MOTION}C_{MOTION}(\mathbf{V}|B,M)\} \quad (1)$$

where M indicates the current block mode, C_{MOTION} is the complexity cost associated with the selected motion vector $(\mathbf{V}|B,M)$, γ_{MOTION} is the Lagrange multiplier for the complexity term, $J^{R,D}_{MOTION}$ is the R-D joint cost function defined in [6].

Similarly the optimal block mode M^* is found by the following.

$$M^*(MB,QP) = \arg\min_{M}\{J^{R,D}_{MODE}(M|MB,QP) + \gamma_{MODE}C_{MODE}(M|MB,QP)\} \quad (2)$$

where C_{MODE} is the complexity cost associated with the block mode, γ_{MODE} is the Lagrange multiplier, $J^{R,D}_{MODE}$ is the R-D joint cost function defined in [6]. When $\gamma_{MODE}, \gamma_{MOTION} = 0$, the solutions of Equation (1) and Equation (2) are identical with conventional R-D framework. Considering the difference between distortion values used in Equation (1) and Equation (2) (mean of absolution difference or MAD, and mean of squared error or MSE respectively), the following relationship is justified empirically [6].

$$\gamma_{MOTION} = \sqrt{\gamma_{MODE}} \quad (3)$$

3.2. Complexity cost modeling

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. The computational complexity is heavily influenced by the type and the location of the motion vector and the interpolation filters used in the MC 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, as listed in Table 1. Furthermore, filtering of different lengths incur different computational costs. Considering 6-tap filtering is much more complex than 2-tap one, a simplified cost function model based on Table 1 is as follows:

$$c_P(\mathbf{V}) = \begin{cases} 0 & \mathbf{V} \text{ is integer MV} \\ 1 & \mathbf{V} \text{ is subpixel a, b, c, d, h and n} \\ 2 & \mathbf{V} \text{ is subpixel e, g, p, r} \\ 7 & \mathbf{V} \text{ is subpixel i, j, f, k, q} \end{cases} \quad (4)$$

Our optimization framework is general and other cost models can be easily incorporated into Equation (1) and (2). For example, a combined metric taking into account both numerical operations and memory access transactions may be used to more accurately model the power consumption condition in some specific mobile devices.

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:

$$C_{MOTION}(\mathbf{V}|B,M) = \sum_j \Big(c_B(\mathbf{V}_j, M, B)\Big) \quad (5)$$

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:

$$C_{MODE}(M|MB) = \sum_i \sum_j \Big(c_B(B_i, \mathbf{V}_j, MB)\Big) \quad (6)$$

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 (5) and Equation (6) are generic and applicable to all inter-coded block modes, including foreword/backward/bi-directional ME and SKIP/DIRECT mode.

3.3. Complexity control

The RDC framework above allows us to find optimal motion vectors and block modes with respect to the given constraints of bit rate and computational complexity. Given the overall computational resource, we also need a method 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. for each coding unit. Solutions to such problems are called complexity control methods, analogous to rate control methods used in conventional video coding. In this section, we describe two components of the complexity control algorithm - the complexity modeling and the buffer management.

3.3.1. Complexity modeling

In complexity control, we need a model for predicting the relation between the control parameter (γ_{MODE} in our case) and the resulting complexity. This is analogous to the rate modeling relationship used in practical video coding. However, so far there is very little knowledge available for such prediction model. Therefore, we resort to our empirical observations from experimental simulations. Specifically, we have found a reasonable log-linear model as follows.

$$C_{MODE} = D\Big(K_1 \ln(\gamma_{MODE}) + K_0\Big) \tag{7}$$

where C_{MODE} is the computational complexity, D is a factor characterizing the video content (e.g., scene complexity and motion activity). K_0, K_1 are the model parameters that needed to be determined empirically. Due to different coding mechanism, P and B frames will have distinguished model parameters and need to be handled separately. Using the above model, $\gamma_{MODE}(t)$ for the current coding unit t can be determined by the following.

$$\gamma_{MODE}(t) = \exp\left\{\frac{C_{MODE}(t) - K_0 D(t)}{K_1 D(t)}\right\} \tag{8}$$

where $C_{MODE}(t)$ is the allocated budget of computational resource and and $D(t)$ is the content feature extracted from the current coding unit. We have developed some content features (such as average motion vector magnitude and average DCT coefficient energy) approximating the content characteristics. Details about such features and fitting of the above complexity model can be found in [7].

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 complexity budget available for the current group of pictures (GOP), N_P, N_B the remaining numbers of P, B frames respectively, and η the complexity ratio between P and B. The target complexity levels for P, B frame C_P, C_B are calculated by solving the following equations:

$$\frac{C_B}{C_P} = \eta \tag{9}$$

$$N_P C_B + N_B C_B = C_{GOP} \tag{10}$$

Once C_P, C_B are available, $\gamma_{MODE}(t)$ is determined using Equation (8). The formulations in Equation (9) and Equation (10) assume the basic coding unit as one frame. It can be easily extended to smaller units for a finer granularity.

4. EXPERIMENT RESULTS

In our experiment H.264 reference codec of version JM82 was used. Both Main and Baseline profiles were evaluated. Unless specified elsewhere, full search is used in ME. The experiment was carried on for four sequences (*Akiyo*, *Stefan*, *Foreman* and *Mobile* at CIF resolution) and only the results for *Foreman* are presented here due to space limitation, though similar performance was observed for all of them.

4.1. Rate-distortion-complexity performance in CAMED

Figure 2 lists the rate-distortion performance together with rate-complexity results by different γ_{MODE} values using Main profile. The remaining complexity are relative to original JM82 results. The additional complexity in encoding (see Equation (1) and (2)) is trivial and will not be discussed here. Several important findings are in order. First, adjusting γ_{MODE} is an efficient way to control the computational complexity. A considerable ratio of interpolation computation can be saved with much smaller quality degradation. Secondly, the video quality is well maintained when the complexity is reduced. Up to 60% of the 6-tap interpolation computational cost can be saved without visible impairment (see $500Kbps$ with $\gamma_{MODE} = 20$ and PSNR drop about 0.2dB). Subjective perceptual quality evaluation also provided confirmative conclusion.

The reason of the above excellent performance can be attributed to the statistical characteristic of video signals. In ME not every sub-pixel motion vector is equally important in predicting the reference signal required in the motion compensation process. Moving less critical ones to the alternatives with reduced complexity will not dramatically increase the prediction error, but will help significantly in reducing the computational cost at the decoder. This important feature, however, could not be utilized using R-D framework. Our proposed CAMED can efficiently take advantage of this.

4.2. Compatibility with fast motion estimation

Fast motion estimation (FME) is widely used in video encoding applications in order to reduce encoding complexity. In FME not all pixels (sub pixels) are checked, which might have potential influence on CAMED in that the optimal motion vector with best RDC performance might be skipped. We also investigate the compatibility of our CAMED with FME. The experiment results in this section were obtained using the Baseline profile. FME was switched on for integer motion vectors, and subpixel ME use either 3-step full search (FS) [4] or fast fractional motion estimation (CBFPS, or center biased fractional pel search, the algorithm described in [8]).

Table 2 lists the result at $400Kbps$. Four options are compared: FS, CBFPS, FS+CAMED, and CBFPS+CAMED. Several conclusions can be drawn. First, though CBFPS can efficiently reduce the encoding complexity, it has small contribution to decoding complexity (e.g., we observed 2.0% saving for *Stefan* in our experiment). This is reasonable because CBFPS (and other FME algorithms) actually does not take into account the decoder complexity in its design. Second, the combination of CAMED with the encoder-side simplification method (CBFPS) can effectively reduce the complexity at both the encoder and the decoder (64.75% and 74.28% for en-

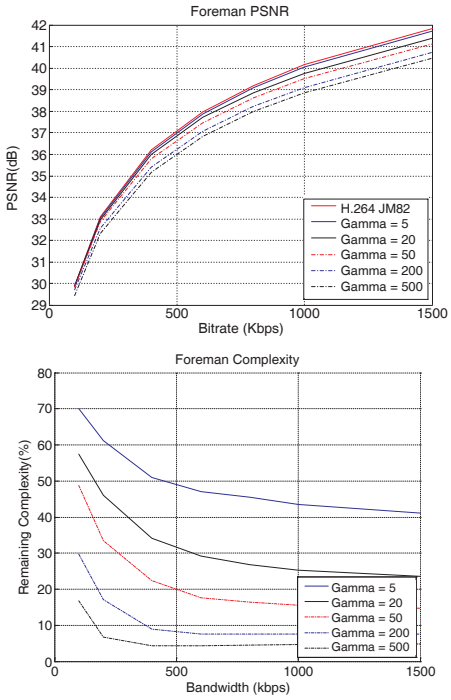

Fig. 2. Performance of rate-distortion and rate-complexity using the proposed CAMED system. Note the complexity includes only the 6-tap interpolation computation required for reconstructing the reference signals in the decoder, the most demanding component in decoding.

coding and decoding saving respectively). The quality degradation is very small (0.206dB). This demonstrates that CAMED has very good compatibility with FME algorithms. We believe this result is very significant for real applications in order to achieve encoding-decoding joint complexity reduction.

4.3. Complexity control

During complexity control, the basic operations are to adjust γ_{MODE} and manage the complexity buffer. In our experiment γ_{MODE} was initialized to 10 for all sequences and adjusted during complexity control. Table 3 summarizes the performance of complexity control at $100Kbps$ with baseline profile. These results confirm that large savings of the computational complexity can be achieved with small quality degradation when the complexity control process is incorporated. Complexity control error is calculated as the difference between the actual resulting complexity and the target complexity, normalized by the target complexity. Our results in Table 3 also show that the complexity control process is working effectively, adaptively updating the control parameter γ_{MODE} in allocating resource over multiple coding units and meeting the overall resource constraint.

5. CONCLUSIONS

We present a novel complexity adaptive motion estimation and mode decision (CAMED) method and use it to effectively reduce the decoder-side computational cost required at the H.264 decoder. Our extensive

Table 2. Evaluation of compatibility with Fast ME (FME): *Foreman*

	PSNR(dB)†	Encoding Saving♮	Decoding Saving♯
FS	36.024	0.00%	0.00%
CBFPS	-0.044	61.80%	17.39%
FS+CAMED $\gamma = 10$	-0.147	0.00%	70.08%
CBFPS+$\gamma = 10$	-0.206	64.75%	74.28%

† Except for FS, other values indicate the quality degradation

♮ Encoding saving is based on the amount of checked sub pixels

♯ Decoding saving is based on the amount of interpolations

Table 3. Complexity control performance ($100Kbps$)

Target Complexity (6-tap Interpolation)		10K	20K	30K
Foreman	Complexity control error (%)	5.00	1.50	3.38
	Complexity saving (%)	81.70	65.67	45.95
	Quality degradation (dB)	0.16	0.09	0.03

experiments using JM82 reference software 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 difference (within 0.2 dB). The proposed complexity control scheme can reliably meet the target complexity requirement for a wide range of video content. We have also shown that it can be effectively combined with other methods to jointly reduce the computational complexity at both the encoder and decoder sides.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and et al, "Overview of the h.264/avc video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, , no. 7, pp. 560–576, Jul. 2003.

[2] V. Lappalainen, A. Hallapuro, and T.D. Hamalainen, "Complexity of optimized h.26l video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, , no. 7, pp. 717–725, 2003.

[3] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proceedings of VCIP'02*, Jan 2002, pp. 1069–1079.

[4] P. Yin, A. M. Tourapis, H.-Y. Cheong, and J. Boyce, "Fast mode decision and motion estimation for jvt/h.264," in *Proc. ICIP'03*, 14-17 Sept. 2003, vol. 3, pp. 853–856.

[5] X. Zhou, E. Li, and Y.-K. Chen, "Implementation of h.264 decoder on general-purpose processors with media instructions," in *Proc. VCIP'03*, Jan. 2003.

[6] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[7] Y. Wang and S.-F. Chang, "Motion estimation and mode decision for low-complexity h.264 decoder," Tech. Rep. 210-2005-4, Columbia University DVMM Group, 2005.

[8] Z.B. Chen, P. Zhou, and Y. He, "Fast integer pel and fractional pel motion estimation for jvt," in *JVT of ISO/IEC MPEG and ITU-T VCEG, JVT-F017*, Awaji, Japan, 5-13 Dec. 2002.