Complexity Control of Motion Compensation for Video Decoding

Wei-Hsiang Chiou Chih-Hung Kuo and Yi-Shian Shie
National Cheng Kung University, Tainan, Taiwan
E-mail:{n26004799, chkuo, n26011445}@mail.ncku.edu.tw,

***Abstract*—This paper proposes a complexity control mechanism for the video encoder to generate a bitstream that fits the power constraint of the decoder. We combine the complexity term of motion compensation with the conventional rate-distortion optimization (RDO). The Lagrange multiplier is updated for each macroblock (MB) to meet the target computing complexity. Experimental show that the proposed method provides a good control accuracy of computing complexity. The whole average error of test sequences is 1.20% with constant bit rate constraints.**

I. Introduction

Modern video coding standards enhance the video quality by searching for the best combination from a large amount of prediction modes. The complexities both on the encoder and the decoder are highly increased comparing to prior coding standard. For encoders, motion estimation dominates the computation power, including different block sizes, multiple reference frames, and different precision of motion vectors (MVs). There have been many speed-up methods proposed to adjust the complexity such as diamond search, hexagon search, and fast mode decision for the inter predictions.

On the other hand, the decoder has less flexibility to adjust the computing complxity. The decoders are more strictly defined by the standards, and their tasks are simply to reconstruct the videos from bitstreams. They have less options to adjust the complexity. When an encoder generates a bitstream, it usually has no information on the capability of decoding platform. Although the decoder is less complex than the encoder, sometimes it is still necessary to restrict the computing power of the decoder. If the required computation power exceeds the capability of the decoding platform, the user may suffer from many unpleasant visual experiences. The limitation of the computing power becomes more realistic as the portable multimedia devices become increasingly popular. Different from traditional general-purpose computers, portable devices have many hardware constraints. Therefore, how to generate a decoder-friendly bitstream according to computing power of the decoder platform becomes more and more important.

For an H.264/AVC decoder, the operation of motion compensation usually dominates the computing power. This high complexity mainly comes from the needs of high precision for MVs. If the MVs are not integer, the 6-tap finite impulse filters are invoked to interpolate for the fraction pixel. There are two major methods to predict the complexity of the motion compensation according to whether it uses the trained model or not during the encoding precess. The former uses a linear model which is trained off-line to get the more precise decoding time. Lee *et al.* [1] propose a prediction model for the motion compensation as a linear function of cache misses, the number of interpolation, and the number of MV per MB. Then, this model is used in the rate-distortion optimization (RDO) process to control the complexity. If the complexity estimated from linear model is higher than the target complexity at motion estimation process, the mode decision process is skipped. Mehdi *et al.* [2] show that the motion compensation complexity is linearly related to the basic operations such as sum, multiply, and shift to interpolate one pixel. The latter is joining the Lagrangian based cost function with computing complexity. Ugur *et al.* [3] proposes the interpolation complexity based on the precision of MVs, and combines the complexity to the conventional Lagrange cost function in order to find the trade off between RD performance and computing complexity.

Lee *et al.* [4] propose that the complexity allocated for each frame should be the same. Theoretically, it has good complexity control when the complexity difference between one frame to another is similar. However, if the difference is huge, the remaining complexity from previous frame can not be used to later frames. This may waste the available computing power, and increase the error in complexity control. In this paper, we focus on adjustment in the motion estimation. With different MV precisions, the filters which are used to get the fraction pixels may be different. We combine the interpolation complexity model [1] with the H.264 convention Lagrangian cost function, and update the Lagrange multiplier to meet the target complexity constraint. This mechanism enables the encoder to generate the bitstream that meets the capability of the decoder platform.

The rest of the paper is organized as follows. Section II introduces how to incorporate the complexity control mechanism into the encoder, which generate bitstream that meets the complexity constraint. Section III shows the experimental results. Finally, Section IV concludes our work.

II. Complexity Control

In order to fit the hardware constraint of the decoder, we propose a complexity control mechanism for encoder to choose suitable prediction mode. The proposed mechanism is composed of two main processes as shown in Fig 1. In the RDO process, we combine the Lagrangian cost function with

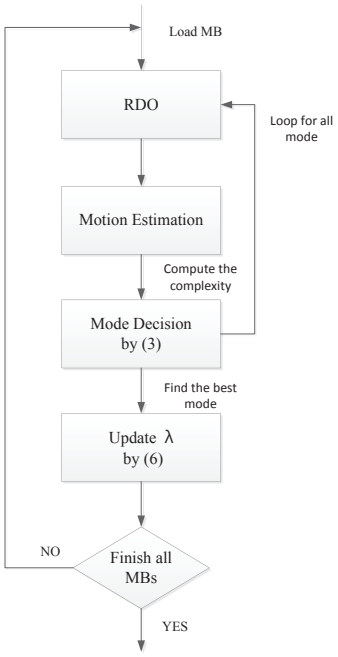

Fig. 1. The proposed complexity algorithm.

TABLE I
NUMBERS OF THE INTERPOLATION 6-TAP FILTER FOR MXN BLOCK

MV(x,y) Accuracy	N_x	N_y
(Int-pel,Int-pel)	0	0
(Sub-pel,Int-pel)	M·N	0
(Int-pel,Sub-pel)	0	M·N
(Half-pel,Sub-pel)	M·N	M·(N+5)
(Sub-pel,Half-pel)	M·(N+5)	M·N
(Quarter-pel,quarter-pel)	M·N	M·N

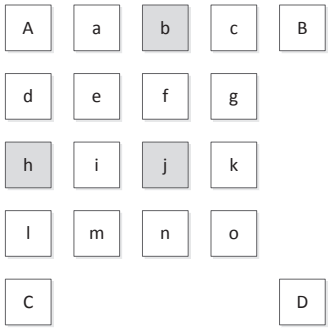

Fig. 2. Fraction pixel interpolation.

the complexity cost so that the prediction mode can obtain trade-offs between bit rate, quality and computing complexity. After finding the best prediction mode, we use the subgradient method to update the Lagrange multiplier in order to meet the target complexity constraint. In the following subsection, we will introduce how to measure the complexity.

A. Modeling Complexity of Interpolation

The H.264/AVC standard enables MVs with half-pixel and quarter-pixel precisions in the motion estimation precess to enhance the prediction accuracy. For different precisions of MVs, it uses different methods to interpolate the sub-pixels. For half-pixels, 6-tap finite impulse filters are used. For quarter-pixels, two nearest half- or integral-pixels are used for interpolations.

For example, in Fig 2, A,B,C,D are interger-pixels, and a through p are fraction-pixels. The half-pixel j should be interpolated in x-direction and y-direction. In order to use 6-tap filters to interpolate j, the extra sub-pixels need to be interpolated in advance. Due to these additional computations, the half-pixel j is the most complex in all sub-pixels. Besides, the quarter-pixels f,i,k,n should use the j value, so these pixels are most complex in all quarter-pixels. Lee *et al.* [1] proposed a model that can compute the complexity by counting the number of the 6-tap filters, and Table I summarizes the number of filters for diffrent accuracy of MVs.

B. Cost Function for Complexity

In order to generate bitstream that meets the target complexity, we add a complexity term in the conventional H.264/AVC RDO process. The RDO is used to trade off between bit rate and video quality. It consists of two steps: the motion estimation and the mode decision. The former finds the best MV that minimize the Lagrange cost in the search range among all reference frames. The latter finds the best prediction mode from all candidate modes to minimize the Lagrange cost.

At the motion estimation step, the reference frames are searched for each inter prediction mode, and the MV that has minimal Lagrangian cost is chosen. The convention H.264/AVC motion estimation Lagrangian cost is given by

$$J(MV, \lambda_{motion}) = SAD(MV) + \lambda_{motion} R(MV) \quad (1)$$

where MV is the motion vector, λ_{motion} is the Lagrangian multiplier, SAD is the sum of absolute difference between original and reference blocks, R is the bits to code the MV. After finding the best MV, we compute the complexity of interpolation according to the precision of MV, and combine it in the later steps.

At the mode decision step, the best mode that has minimal Lagrangian cost is chosen from all candidate prediction modes. The convention H.264/AVC mode decision Lagrangian cost is represented by

$$J(mode, \lambda_{mode}) = SSD(mode) + \lambda_{mode} R(mode) \quad (2)$$

where R is the bits generated by the entropy coding, λ_{mode} is the Lagrangian multiplier, SSD is the sum of square difference between reconstruct and original blocks.

Taking the computing complexity into consideration, we modify the equation (2). The equation becomes

$$J(mode, \lambda_{mode}, \lambda_c) = SSD(mode) + \lambda_{mode} R(mode) + \lambda_c C_{mode} \quad (3)$$

where λ_c is the Lagrangian multiplier associated with the computing complexity. C_{mode} represents the interpolation complexity counted as in Table I, and it is accumulated of all involved in one MB. It can be written as

$$C_{mode} = \sum_{i=1}^{N} C_{intp}(i) \tag{4}$$

where N is the number of MVs in a MB, and C_{intp} is the number of 6-tap filters which is used for each partition. For example, if the MB is coded as inter-8×8 mode, there are four motion vectors in a MB. The value of N is 4. After the mode decision step, the best mode which is chosen from Equation (3) obtains a trade-off between the video quality, the bit rate and the computing complexity. In the next subsection, we proposed an algorithm for controling the computing complexity.

C. Complexity Control

The procedure of the complexity control is similar to that of the rate control. It uses parameter λ_c to allocate the complexity cost for each coding units, and finally meets the target complexity constraint. For complexity control, we need to decide λ_c which controls the trade-off between the R-D performance and computing complexity. The value of λ_c is usually a constant value [3]. However, the complexity of each MB (or each frame) is different, and λ_c should be dynamically adjusted which depends on the complexity in previous MBs. In this work, we use the subgradient method to adjust λ_c. The coding unit is set to be a MB. For the $(k+1)$th MB, λ_c should be adjusted according to the previous MB by

$$\lambda_c^{(k+1)} = \lambda_c^{(k)} - \alpha_k g^{(k)} \tag{5}$$

where $g^{(k)}$ is a subgradient of the equation (3), α is a positive step size. In order to simplify the computation, we use the difference between the $C(mode)$ and the target decoding complexity C_{target} as the subgradient g^k. The equation (5) is rewritten as

$$\lambda_c^{(k+1)} = \lambda_c^{(k)} + \alpha_k (C_{mode}^{(k)} - C_{target}). \tag{6}$$

After the RDO process, we find the best prediction mode, and the complexity of interpolation C_{mode}. Then we use Equation (6) to get a new λ for next MB to use. For example, when the C_{mode} of the current MB is higher than the target C_{target}, it means that we have used more complexity than expected, and we should decrease the complexity for next MB. Hence, λ_c^{k+1} increases slightly and less complexity is used for the next MB.

III. Experimental Results

The proposed algorithms are implemented on the H.264/AVC software reference JM18.0. The search range is 32x32 pixels, and the number of reference frames is 5. Totally 6 video sequences are tested, each has 300 frames. The QP value ranges from 20 to 36. In order to emulate different decoding platforms, we set different complexity ratios from 0.5 to 0.8. For example, if the ratio is 0.8, we

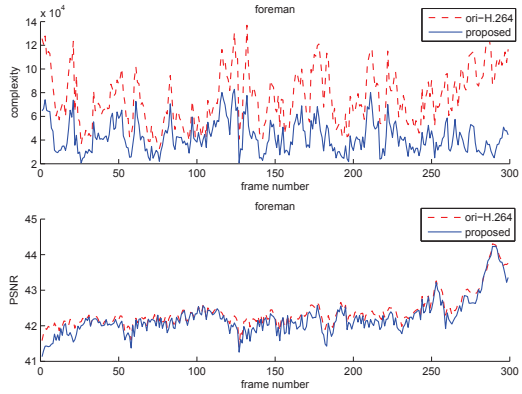

Fig. 3. Frame-to-frame complexity and PSNR, target complexity set to 60%.

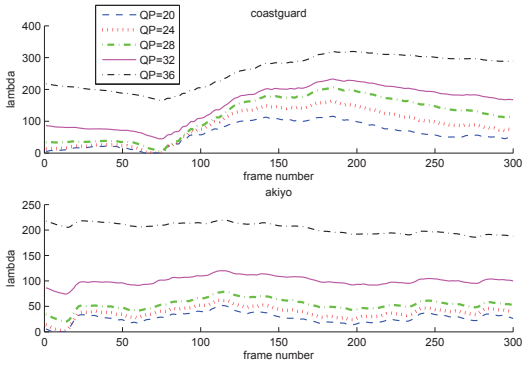

Fig. 4. The values of λ_c for various QP.

expect that the number of filters to be used to interpolate fraction pixels is only 80 percent of the original. Fig. 3 compares frame-to-frame complexities and PSNR. Our proposed algorithm can allocate complexity according to each frame and the distribution of complexity is similar to original H.264 reference software. Fig. 4 shows the evolution of λ_c as the frame index grows in test sequences 'coastguard' and 'akiyo'. The former is a high motion sequence, and the latter is a low motion sequence. In the test sequence 'coastguard', we can see that λ_c is very close to zero (but not exactly zero) from the 40th frame to 75th frame. During these frames, the complexity is lower than target complexity, so the value of λ_c decreases in order to allocate more complexity to meet the target complexity constraint. However, the sequence 'akiyo' is more static, and the motion is lower. Hence, the values of λ_c fluctuate less than the other.

The performance at different complexity is summarized in Table II, where we list the differences in PSNR loss between original H.264 and our algorithm. The result shows 60% complexity is achieved at the cost of a small bit rate increase, as well as a small PSNR loss. The above simulations do not impose any bit rate constraint. The bit rate varies for different test sequences and different complexities. For more practical simulations, we combine our complexity algorithm with the H.264/AVC rate control. The performance is summarized in

TABLE II
R-D-C PERFORMANCE UNDER COMPLEXITY CONSTRAINT

sequence	Complexity	PSNR loss (dB)	bit rate increase(%)
akiyo	0.8	0.038	-1.88
	0.6	0.089	-1.77
silent	0.8	0.022	-0.32
	0.6	0.045	0.35
foreman	0.8	0.050	-0.32
	0.6	0.100	0.95
stefan	0.8	0.075	-0.08
	0.6	0.145	0.86
coastguard	0.8	0.125	-0.95
	0.6	0.217	0.43
mobile	0.8	0.082	-0.32
	0.6	0.244	0.23
Average	0.8	0.065	-0.65
	0.6	0.14	0.18

TABLE III
R-D-C PERFORMANCE UNDER COMPLEXITY AND BIT RATE CONSTRAINT

sequence	Target bit rate(kbit/s)	PSNR loss(dB)	Rate (kbit/s)	Complexity error(%)
akiyo	400	0.414	399.47	0.18
	1200	0.273	1200.48	0.05
silent	400	0.136	399.95	0.35
	1200	0.190	1201.13	2.69
foreman	400	0.258	400	1.01
	1200	0.296	1199.69	0.42
stefan	400	0.298	400.47	0.13
	1200	0.379	1199.51	0.32
coastguard	400	0.196	399.82	0.75
	1200	0.304	1200.66	0.4
mobile	400	0.305	400.65	0.54
	1200	0.371	1201.21	0.01
Average	400	0.268	400.06	0.49
	1200	0.302	1200.45	0.65

Table III. The results show that 50% of the complexity is reduced under different bit rate constraints. This demonstrates that our algorithm has a good control accuracy under different bit rate constraints.

We compare the proposed algorithm with the reference work [4], in which Lee *et al.* propose that the complexity allocated to each frame should be the same. However, this is not an efficient way. Fig. 5 shows the frame-by-frame complexity for different algorithms. Lee's algorithm does not adjust the complexity according to different frames. If there exists any extra computing power in the previous frame, it can not be utilized in the current frame. This is the reason that the control error is larger than our algorithm. Fig. 6 compares the error rate for different video sequences under different complexity constraint. For each complexity, we average 4 control errors from 4 different bit rate constraints. Different from the scheme proposed by Lee *et al.*, the control error is more stable in various test sequences and the whole average control error is 1.20% approximately in our algorithm. The result shows that our algorithm more accurately controls the complexity.

IV. CONCLUSION

In this work, we propose a new approach to generate decoder-friendly bitstream. This algorithm helps the encoder to choose the suitable prediction modes, and generate a bitstream that meets the complexity constraints for different decoding platforms. As a result, the bitstream has a balance between RD performance and computing complexity. Our mechanism is verified experimentally. The whole average control error of all test sequences is 1.20% with the bit rate constraint.

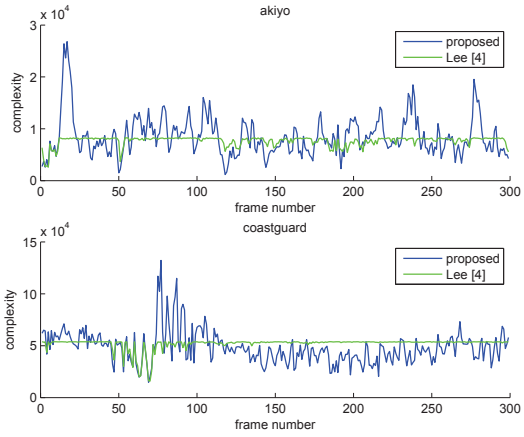

Fig. 5. Frame-by-frame complexity.

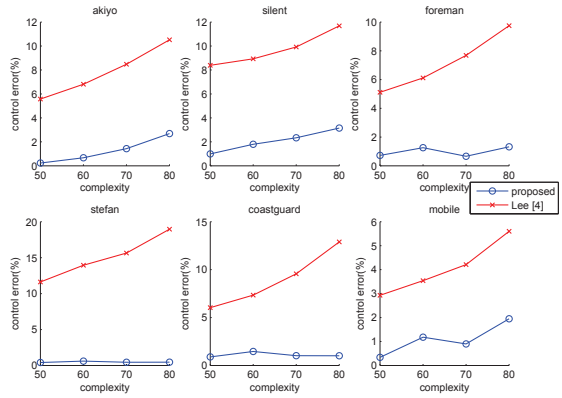

Fig. 6. The complexity error for different algorithms.

V. ACKNOWLEDGMENT

The work is supported in part by National Science Council (NSC) of Taiwan under Grant NSC101-2220-E-006-007.

REFERENCES

[1] S. W. Lee and C. -C. Jay Kuo, "Complexity Modeling for Motion Compensation in H.264/AVC Decoder," *IEEE International Image Processing*, vol. 5, pp. 313-316, Oct. 2007.

[2] Mehdi Semsarzadeh , Mohsen Jamali Langroodi , Mahmoudreza Reza Hashemi and Shervin Shirmohammadi, "Complexity Modeling of the Motion Compensation Process of the H.264/AVC Video Coding Standard," *IEEE Multimedia and Expo*, pp. 925-930, Oct. 2012.

[3] Kemal Ugur, Jani Lainema, Antti Hallapuro, and Moncef Gabbouj "Generating H.264/AVC Compliant Bitstreams for Lightweight Decoding Operation Suitable for Mobile Multimedia Systems ," *ICASSP*, May 2006.

[4] S. W. Lee and C. -C. Jay Kuo, "Motion Compensation Complexity for Decoder-Friendly H.264 System Design," *IEEE Multimedia Signal Processing*, pp. 119-122, Oct. 2007.