GENERATING H.264/AVC COMPLIANT BITSTREAMS FOR LIGHTWEIGHT DECODING OPERATION SUITABLE FOR MOBILE MULTIMEDIA SYSTEMS

Kemal Ugur[1], Jani Lainema[1], Antti Hallapuro[1], Moncef Gabbouj[2]
[1]Nokia Research Center, Tampere, Finland
[2] Tampere University of Technology, Tampere, Finland

ABSTRACT

In this work, we propose novel encoder algorithms for the state-of-the-art video coding standard H.264, to generate decoder friendly video bitstreams. Using the proposed algorithms, it is possible to generate bitstreams requiring significantly less decoding complexity, with negligible effect on picture quality. This is achieved by using novel algorithms for mode decision and motion estimation that bias easy-to-decode motion vectors in a Rate-Distortion optimized fashion. Experimental results show that, more than 15% decoding complexity reduction is achieved with less than a 0.1 dB penalty on the average video quality. We believe that this approach has potential in various use cases especially in mobile multimedia systems, where the video decoder operation is often dominating the handsets power consumption.

1. INTRODUCTION

H.264/AVC is the state-of-the art video coding standard that is jointly developed by ISO/MPEG and ITU-T/VCEG study groups. When compared to the earlier video coding standards, H.264/AVC achieves significantly better video quality at similar bitrates. Due to its high compression efficiency and network friendly design, H.264/AVC is gaining momentum in industry ranging from third generation mobile multimedia services, digital video broadcasting to handheld (DVB-H) to high definition digital versatile discs (HD-DVD). H.264 achieves increased compression efficiency with the expense of increased complexity for both the encoders and the decoders. Similar to previous standards, the complexity of the H.264 encoder is typically much higher than that of the decoder. There exist several tools that significantly reduce the encoding complexity by resulting slightly lower video quality. These tools include for example fast motion estimation [1], fast mode decision [2] and disabling the use of some of the motion modes etc. The commonality of all these is, they achieve complexity reduction by performing a complexity-video quality trade-off.

For decoders, the situation is completely different. Since the decoder is strictly defined by the standard, it is not possible to have lower complexity decoding by performing a similar video quality-decoding complexity trade-off. Even though the decoder has less complexity than the encoder, the importance of low complexity decoding is equally important as low complexity encoding, if not more. The reason for that is, in many applications such as DVD players, digital TV receivers etc, the end-user equipment has only the decoder implemented and the decoder block is the only codec related functional block adding complexity to the system. Even if the encoder and decoder co-exist in a system, such as in a video conferencing application, the decoding complexity could become very important if the encoder and decoder are running on different hardware platforms.

In this work, we propose several encoding algorithms that can be used to generate fully H.264 compliant *low-complexity bitstreams,* which require significantly less decoding complexity than bitstreams created with traditional encoding algorithms. More specifically, the focus of this work is to generate bitstreams that would require less amount of half-pixel and quarter pixel interpolations at the decoder, as interpolation step consumes most of the decoding processor cycles [3]. This is achieved in two stages. At the motion estimation stage, the candidate motion vectors having less decoding complexity are biased using a Lagrangian based cost function. At the mode decision stage, the decoding complexity of each mode is estimated and the modes with less decoding complexity are favoured using a similar cost function. Using the proposed methods, the encoder can generate *low-complexity bitstreams* having over 15% less decoding complexity with less than a 0.1 dB penalty on the video quality on average.

This paper is organized as follows; Section 2 provides a brief analysis of the H.264 decoder complexity and the interpolation scheme. Section 3 presents the proposed motion estimation and mode decision algorithms to generate the low complexity bitstreams. Section 4 presents the simulation environment and the experimental results. Conclusions and discussions are presented in Section 5.

2. H.264 DECODER COMPLEXITY ANALYSIS

Previous analyses on H.264 decoder complexity show that the motion compensation is the most computationally complex step at the decoder, followed by the deblocking filter process [3][4]. The high complexity in motion compensation is due to interpolation needed to decode motion vectors with half or quarter pixel accuracy. It was shown that, for a baseline H.264 decoder, this interpolation step takes around 39% of the execution time on average, and it can go up to 44% for some sequences. In the next subsection, we first analyze the details of the H.264's interpolation scheme and present source of its complexity.

2.1 Half-pixel and Quarter-pixel Interpolation

H.264 allows usage of motion vectors with quarter and half pel precision to increase the accuracy of the motion prediction. For the case of integer motion vectors, the prediction signal contains the original values of the reference picture; otherwise the values at non-integer positions need to be in-

terpolated from the original pixels at integer positions. In Figure 2, the original pixels at integer locations are labelled by upper-case letters within shaded boxes, other symbols represent other locations to be interpolated. The samples at half-pixel locations, *b*, *h*, *m* and *s*, are interpolated from integer samples by applying a one dimensional 6-tap FIR filter on the integer sampled pixels. For example, the sample at half-pixel location *b* is obtained from the samples at integer locations E, F, G, H, I and J given in Figure 1. More specifically, *b* is given as

$$b = \frac{((E - 5.F + 20.G + 20.H - 5.I + J) + 16)}{32} \quad (1)$$

The half-pel sample at position *j* is interpolated by applying the 6-tap filter on the 6 samples at half pixel locations either in vertical or horizontal direction. In other words, in order to interpolate *j*, the samples at locations *cc*, *dd*„ *h, m, ee* and *ff* must be interpolated first (alternatively, *j* is obtained using the samples at locations *aa, bb, b, s*, gg and *hh,* which would yield the same value). Due to the additional need of interpolating other half-sample locations, the interpolation of sample at position *j* has the most complexity among other half-pixel samples.

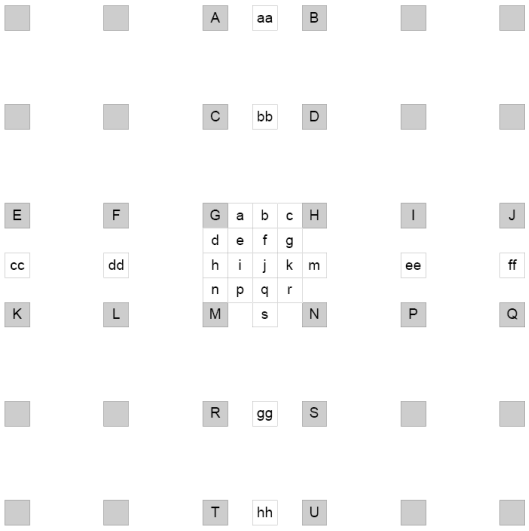

Figure 1 H.264 Sub-Pixel Interpolation

The samples at quarter pixel positions are obtained by averaging two nearest samples at half or integer positions. Figure 2 illustrates the details of the quarter pixel interpolation scheme employed in the H.264 standard. For each quarter pixel, two values are averaged by up-rounding. In Figure 2, quarter pixels are denoted by letters *a, c, d, e, f, g, i, k, n, p, q, r* and are placed within shaded boxes. Each quarter-pixel is connected to two other half or integer pixels that will be used to calculate the corresponding quarter pixel. For example, in order to interpolate quarter-pixel *d*, the integer pixel *G* and the half pixel *h* must be averaged. So, the half-pixel *h* must be interpolated first using the 6-tap filter, in order to calculate the value of quarter pixel *d*. Therefore, the quarter pixels, for which the half pixel *j* is used for averaging, have the most complexity among other quarter pixel locations (i.e. the quarter pixels at locations *f, i, k, q* have the highest interpolation complexity).

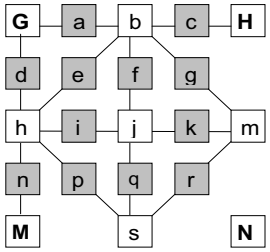

Figure 2 Quarter Pixel Interpolation Scheme

3. ENCODING FOR LOW COMPLEXITY DECODING

In the previous section, it was shown that the interpolation complexities of different quarter and half pixel locations are not the same. Therefore, the decoding complexity depends on both the horizontal and vertical sub-pixel components of the motion vector. For example, if a motion vector has integer values for both its horizontal and vertical components, the decoder does not need to perform any interpolation to obtain the prediction signal, and hence have minimal complexity. However, the complexity would increase, if the motion vector points to locations that are difficult to interpolate, such as the half pixel location *j*, or any quarter-pixel location that requires the value of *j*.

We first analyzed the amount of operations needed to interpolate each sub-pixel location, using a highly optimized H.264 decoder implementation [4]. Using this data, we approximated the interpolation complexity of each motion vector with different horizontal and vertical sub-pixel accuracies. Figure 3 presents the approximate interpolation complexities required to decode motion vectors with different accuracies. The reader is referred to [4][5] for a detailed analysis of interpolation complexity.

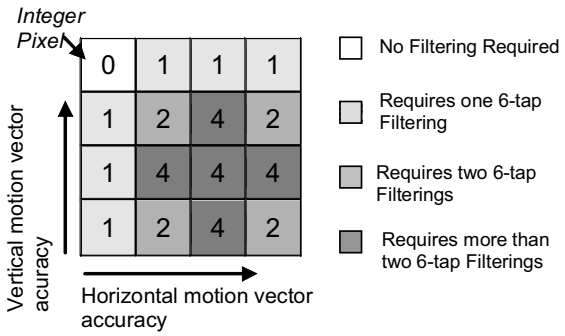

Figure 3 Approximate Interpolation Complexities of Motion Vectors with different sub-pixel components.

In Figure 3, each location is represented by a box and the numbers in each box indicate the approximate interpolation complexity required to decode the motion vector, larger number indicating a higher complexity. For example, if the motion vector has integer values at both directions, the interpolation complexity is zero. However, if the motion vec-

tor points to the half-pixel location just next to the integer pixel in horizontal direction (indicated as *b* in Figure 2), the interpolation complexity increased to 1, as the decoder needs to perform an additional 6-tap filtering.

We use this approximate complexity data to generate bitstreams that have more motion vectors pointing to easy-to-interpolate locations, in order to reduce the decoding complexity. This is achieved in two stages. At the motion estimation step, the candidate motion vectors having less decoding complexity are biased using a Lagrangian based cost function. At the mode decision step, the decoding complexity of each mode is estimated and the modes with less decoding complexity are favoured using a similar cost function. The details of these stages are presented in the next subsections.

3.1 Motion Estimation

At the motion estimation stage, the reference picture is searched for the candidate motion vectors, and the motion vector that results in the best prediction is chosen. The conventional motion estimation that is implemented in the current H.264 test model chooses the motion vector that minimizes the following cost function.

$$J(\mathbf{m}, \lambda_{motion}) = SAD(s, c(\mathbf{m})) + \lambda_{motion}.R(\mathbf{m}) \qquad (2)$$

with $\mathbf{m} = (m_x, m_y)^T$ being the motion vector, and λ_{motion} being the Lagrange multiplier The first term of the above cost function is the distortion term and it is given as the Sum of Absolute Difference (SAD) between the original signal *s,* and the reference signal *c(**m**)*. The rate term $R(\mathbf{m})$ represents the number of bits that would be used to code the motion vector **m**.

In order to favour motion vectors with less interpolation complexity and penalize ones with higher complexity, we modify the conventional cost function, and use the one indicated as in Equation 3.

$$J'(\mathbf{m}, \lambda_{motion}, \lambda_{ME}) = J(\mathbf{m}, \lambda_{motion}) + \lambda_{MEC}.C_{ME}(\mathbf{m}) \quad (3)$$

The proposed cost function has an additional term, $C_{ME}(\mathbf{m})$, that represents the decoding complexity of the candidate motion vector, **m**. $C_{ME}(\mathbf{m})$ is basically a two dimensional array and it is calculated using the data given in Figure 3. For example, if the candidate motion vector has integer-pixel accuracy in both horizontal and vertical directions, then $C_{ME}(\mathbf{m})$ is zero. Similarly, if **m** points to location *j*, the value of $C_{ME}(\mathbf{m})$ is 4. The value of $C_{ME}(\mathbf{m})$ is further multiplied by the Lagrangian term λ_{MEC}, to adjust the complexity–video quality trade-off. Larger the value of λ_{MEC}, less decoding complexity the resulting bitstream has, with a higher penalty on the coding efficiency.

3.2 Mode Decision

After the motion estimation is performed for all candidate INTER modes, the coding results of the modes are compared and the one that minimizes the following Lagrangian cost function is chosen.

$$J(M, \lambda_{MODE}) = SSD(s, r) + \lambda_{MODE}.R(s, c, M) \qquad (4)$$

with *r* being the reconstruction signal for the given mode, *M*, and λ_{MODE} is the Lagrangian multiplier. M is referring to one of the candidate INTER modes as illustrated in Figure 4.

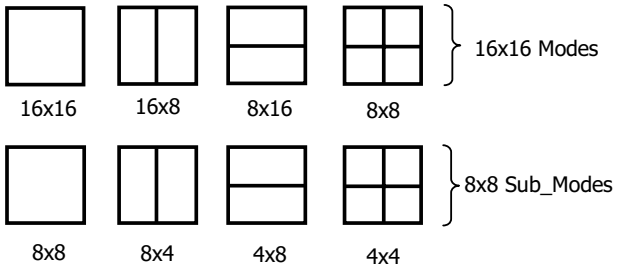

Figure 4 INTER modes supported by H.264 standard

The first term of the above cost function is the distortion term, and it is given as the Sum of Square Difference (SSD) between the original and the reconstructed signal. *R(s,c,M)* is the rate term that represents the number of bits used to code the mode, *M*. In order to favour the INTER modes with less interpolation complexity, we modify Equation 4, and use the following cost function:

$$J'(M, \lambda_{MODE}) = J(M, \lambda_{MODE}) + \lambda_{MDC}.C_{MODE}(M) \quad (5)$$

Similar to the proposed cost function used in Motion Estimation, Equation 5 has the additional term $C_{MODE}(M)$ representing the decoding complexities of candidate modes. $C_{MODE}(M)$ is the sum of all the interpolation complexities for all motion vectors involved in the candidate mode, and is illustrated below in Equation 6.

$$C_{MODE}(M) = \sum_{i=1}^{num\_of\_MVs} C_{ME}(\mathbf{m}_i) \qquad (6)$$

where *num_of_MVs* refers to the number of motion vectors used in the given mode. For example, if the candidate mode is INTER_8x8, in which the macroblock is divided into four blocks and each block has its own motion vector, the value of *num_of_MVs* is four. $C_{MODE}(M)$ is multiplied by λ_{MDC} to adjust the complexity-quality trade-off at the mode decision stage. The values for λ_{MDC} and λ_{MEC} affect the encoding performance and should be careful selected. Results of our simulations suggest that values 2, 50 for λ_{MDC} and λ_{MEC} respectively result in the best complexity-quality trade-off. The effect of changing the values of Lagrangian parameters, λ_{MEC} and λ_{MDC}, could be observed more in the experimental results.

It should be noted that, conventional cost function is used in the proposed method when deciding between INTER and INTRA modes.

4. EXPERIMENTAL RESULTS

In order to test the performance of our scheme, we first generated the *low-complexity* bitstreams for different test sequences, using the methods described above. Then using a highly optimized H.264 decoder implementation on an ARM-11 platform, we decoded all the bitstreams and recorded the required number of processor cycles for each one. We repeated the same process using the conventional encoder, with the same encoder settings, to get the reference *RD Optimized* bitstreams.

Figure 5 presents the results for the test sequence Foreman. When compared to the *RD Optimized* bitstream, the *low complexity* bitstream has 14% less decoding complexity. The

penalty on video quality, to achieve lower complexity decoding is 0.1 dB. We also present the decoded pictures from both the bitstreams in Figure 6. It is observed that the *low complexity* bitstream has practically the same visual quality as the *RD Optimized* one.

As mentioned before, the values of λ_{MDC} and λ_{MEC} are chosen as 2, 50 respectively. However, our method allows one to decrease the decoding complexity more by allowing a higher penalty on the coding efficiency. This could be achieved by using larger values for λ_{MDC} and λ_{MEC}. In order to demonstrate the effect of changing those values, we repeated the above experiment with λ_{MDC} being 4 instead of 2, to favour modes having less decoding complexity, and generated the *lower complexity* bitstream. As it is seen in Figure 7, the decoding complexity is decreased even more, with slightly less coding efficiency.

5. CONCLUSIONS

In this work, we proposed novel encoder algorithms that are used to generate H.264 compliant decoder-friendly bitstreams. Using the proposed algorithms, it is possible to generate bitstreams requiring significantly less decoding resource consumption, with negligible effect on picture quality. This is achieved by using novel algorithms for mode decision and motion estimation that bias easy-to-decode motion vectors in a Rate-Distortion optimized fashion. It was shown that, more than 15% decoding complexity reduction is achieved with less than a 0.1 dB penalty on average video quality. It was also shown that proposed methods allow further complexity reduction, by having slightly less picture quality.

It is believed that, this approach has many important use cases in mobile multimedia systems, where the video decoder operation is often dominating the handsets power consumption.

REFERENCES

[1] Peng Yin; Tourapis, H.-Y.C.; Tourapis, A.M.; Boyce, J. , "Fast mode decision and motion estimation for JVT/H.264", in ICIP 2003, vol.3, pg: 853-856, 4-17 Sept. 2003

[2] J. Lee; B. Jeon, "Fast mode decision for H.264," IEEE International Conference on Multimedia and Expo, 2004., vol.2, no.pp. 1131- 1134 Vol.2, 27-30 June 2004

[3] Horowitz, M.; Joch, A.; Kossentini, F.; Hallapuro, A., "H.264/AVC baseline profile decoder complexity analysis", in CSVT, IEEE Trans. on, vol.13, pg: 704- 716, July 2003

[4] Lappalainen, V.; Hallapuro, A.; Hamalainen, T.D., "Complexity of optimized H.26L video decoder implementation", CSVT, IEEE Trans. on, vol.13, pg: 717- 725, July 2003

[5] Karczewicz, M.; Hallapuro, A.; "Interpolation solution with low encoder memory requirements and low decoder complexity.", VCEG-N31, 24-27 Sept. 2001.

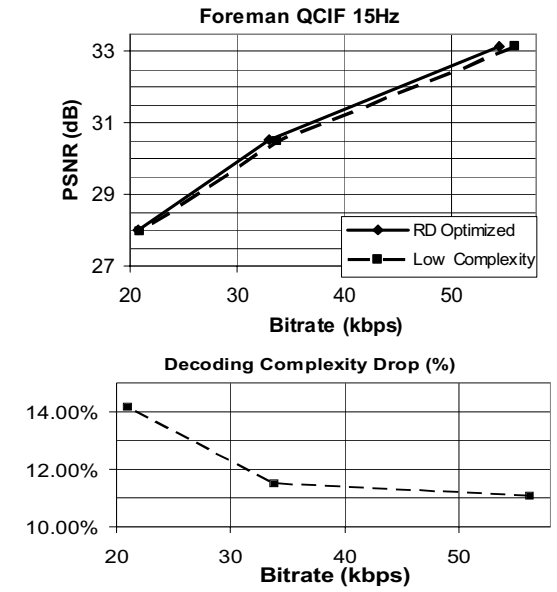

Figure 5 Analysis of the proposed method

Figure 6 Reconstructed Frame # 89 Foreman a: RD Optimized Reference b: Low Complexity

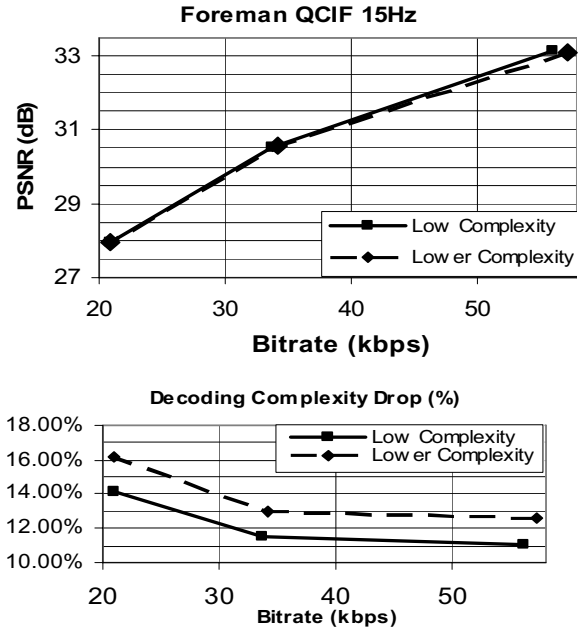

Figure 7 Effect of Changing λ_{MDC}