Complexity Modeling for Motion Compensation in H.264/AVC Decoder

Szu-Wei Lee and C.-C. Jay Kuo

Ming Hsieh Department of Electrical Engineering and Signal and Image Processing Institute
University of Southern California, Los Angeles, CA 90089, USA
E-mails: szuwei.lee@usc.edu and cckuo@sipi.usc.edu

ABSTRACT

A complexity model for motion compensation in the H.264/AVC decoder is proposed. This model considers a rich set of inter prediction modes allowed by H.264 as well as the relationship of motion vectors (MVs), which are related to cache miss performance. This model is verified by experimental results. Possible applications of the complexity model are also described. One application scenario is to integrate it in an H.264 encoder so that the encoder can estimate the decoding complexity and choose the best inter prediction mode to meet the complexity constraint of the target decoding platform.

Index Terms— **H.264/AVC, motion compensation, decoder complexity, rate-distortion optimization (RDO)**

1. INTRODUCTION

H.264/AVC [1] is the latest video coding standard proposed by ITU-T and ISO/IEC. It has been selected as the video coding tool in HD-DVD and Blue-ray specifications. H.264 provides various inter prediction modes to improve the coding gain. That is, its encoder may search all possible inter prediction modes and choose the best one that minimizes the rate-distortion (RD) cost. Due to the use of a larger set of inter prediction modes, the H.264 decoding complexity becomes higher, too. For example, it was reported in [6] that its complexity is about 2.1 to 2.9 times more than the H.263 decoder. For some applications, the coded video bit stream will be decoded in portable consumer electronics devices. Under such a scenario, the reduction in decoding complexity so as to save the power becomes a critical issue.

To achieve the power saving purpose of the H.264 decoder, one solution is to allow the H.264 encoder to generate a decoder-friendly bit stream. That is, the H.264 encoder has a target decoding platform in mind and yields a bit stream that is easy to decode in that platform. This motivates us to study the complexity model for the H.264 decoder. Once the model is available, it can be used by the H.264 encoder to estimate the decoding complexity for various inter prediction modes and then select the best one to balance the tradeoff between rate-distortion (RD) and decoding complexity. Since a general decoding complexity model is too broad to cover, we are focused on the complexity model for motion compensation (MCP, i.e. the decoding process for inter prediction) operation in H.264 in this work.

The decoding complexity models have been studied in the past [2]-[5]. A MPEG-4 video complexity verifier (VCV) model was described in [2], where the numbers of boundary macro-blocks (MBs) and non-boundary MBs decoded per second are estimated and the decoding complexity can be modeled as a function of these two numbers. However, the decoding complexities of the MBs coded by different inter prediction modes in MPEG-4 can be different so that the VCV model is not very accurate. To address this shortcoming, Valentim *et al.* [3][4] proposed an enhanced complexity model of MCP operation for MPEG-4 video. By considering the fact that MBs coded with different inter prediction modes have different decoding complexities, they use the maximal decoding time to measure the decoding complexity of MBs coded by different inter prediction modes individually. Then, the total decoding complexity of this bit stream is the sum of each individual MB's complexity. More recently, it was reported in [5] that the complexity model for MCP can be simplified and the decoding complexity of MB is proportional to the number of motion vectors (MVs). In other words, the MB with more MVs should have higher decoding complexity than that with fewer MVs.

The above decoding complexity models are however not suitable for H.264 for two reasons. First, H.264 provides a richer set of inter prediction modes that cannot be well handled by the existing models. Second, these models do not take the relationship of MVs into account. Since the relationship of MVs is related to the efficiency of cache management, it plays an important role in CPU performance and therefore decoding complexity. A new complexity model for MCP is provided to address these two issues in this work.

The rest of this paper is organized as follows. An H.264 decoding complexity model for MCP operation is proposed in Sec. 2. The proposed decoding complexity model is verified experimentally in Sec. 3. The applications of the decoding complexity model and its integration with an H.264 encoder are discussed in Sec. 4. Finally, concluding remarks are given in Sec. 5.

2. PROPOSED COMPLEXITY MODEL

2.1 Complexity model

H.264 provides various block sizes for inter prediction to improve the coding gain. One 16x16 MB can be partitioned into one 16x16 block, two 16x8 or 8x16 blocks or four 8x8 blocks. Each 8x8 blocks can be further partitioned into two 8x4 or 4x8 blocks or four 4x4 blocks. As a result, there are totally 19 modes to encode one 16x16 MB. Moreover, each block whose size is larger than 8x8 can be predicted using different reference frames. In addition, H.264 provides fractional samples of motion estimation (ME). The MV can be of half- or quarter-pel resolution. The half-pel value is obtained by applying a one-dimension 6-tap FIR interpolation filter horizontally (the x-direction) or vertically (the y-direction). The quarter-pel value is obtained by the average of two nearest half-pel values. This flexibility makes the decoding complexity modeling more challenging as compared with that in [3],[4].

In this work, the decoding complexity is modeled as a function of the number of cache misses N_c, the number of y-direction interpolation filters N_y, the number of x-direction interpolation filters N_x and the number of MVs per MB N_v. Mathematically, it can be written as

$$\alpha \cdot N_c + \beta \cdot N_y + \gamma \cdot N_x + \mu \cdot N_v, \tag{1}$$

where $\alpha, \beta, \gamma, \mu$ are weight coefficients of these four decoding complexity coefficients, respectively. As done in [5], our model includes the number of MVs per MB. The numbers of the x-direction and the y-direction interpolation filters are used to model the decoding complexity of interpolation filter if the MV is of half- or quarter-pel accuracy. Since the decoder may have different implementations of interpolation filters along the x- and y-directions, two different terms are used in the proposed decoding complexity model.

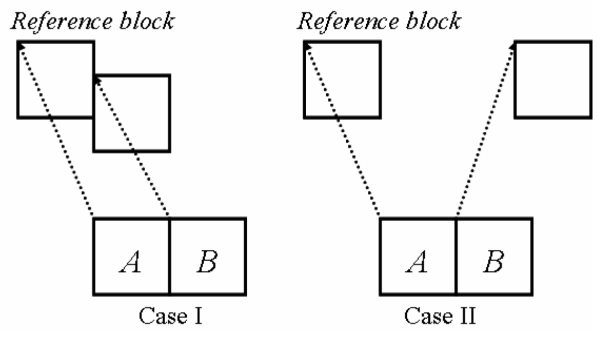

Figure 1. Inter prediction of two successive blocks

Furthermore, we include the number of cache misses to deal with two cases shown in Fig. 1, where two MVs point to two reference blocks with a different spatial relationship. The two reference blocks are closer to each other in Case I but far away in Case II. The decoding complexities of the two cases are different since most data required for block B in Case I could be obtained in the CPU cache or internal registers after the decoding of block A is done. Intuitively, the number of cache misses for Case I is fewer than that of Case II, and the decoding complexity for block B in Case I should be lower than that of Case II.

Two experiments are conducted on Pentium 1.7 Ghz mobile CPU platform to verify this conjecture. We modify the H.264 encoder so that the encoder can use a specific inter prediction mode to encode MBs. Two inter prediction modes, *i.e.* P16x8 and P8x16, are selected to encode Foreman sequence and generate two bit streams where MV is only allowed in the integer-pel position. It is observed by Intel Vtune that MCP decoding complexities of these two bit streams are quite different. The MCP decoding complexity of the bit stream encoded by P16x8 takes about 64.85 milli-seconds while that encoded by P8x16 takes about 100.19 milli-seconds. This implies that the number of MVs is not the only factor to decide the MCP decoding complexity. In addition, the P4x4 mode is selected to encode Foreman and Container sequences where MV is only allowed in the integer-pel position again. It is also observed by Intel Vtune that their MCP decoding complexities are quite different, too. The MCP decoding complexities of Foreman and Container P4x4 bit streams take about 272.19 and 208.58 milli-seconds, respectively. The reason of demanding less MCP decoding complexity for the Container bit stream is that it contains a large smoother area and its relationship of MVs is similar to that in Case I while Foreman has a lot of motion and its relationship of MVs is similar to that in Case II. This demonstrates that the relationship of MVs plays an important role in the decoding complexity model.

The weight coefficients in Eq. (1) can be obtained by the following steps. First, several pre-encoded bit streams are selected and the Intel Vtune performance analyzer is used to measure the number of clock ticks spent by the MCP operation. Second, the numbers of the decoding complexity coefficients are counted individually for these pre-encoded bit streams. Finally, the constrained least square approach is used to find the best fitting of the weight coefficients.

For simplicity, the decoding complexities of MB with a half- and a quarter-pel MV in our complexity model are assumed to be the same since the operation that takes the average of two nearest half-pixels requires only two addition and one shift instructions. As compared to other operations, the cost of these operations is low and negligible in most platforms.

2.2 Computation of cache m*isses*

As mentioned in the previous section, the relationship of MVs of successive blocks can lead to different decoding complexities. To deal with the two cases as shown in Fig. 1, a simple cache model is used to compute the number of cache misses. The simple cache has *T* entries and each entry has *L* bytes. If the required data for the MCP operation from address $addr_i$ with data size *S* cannot be found in the cache, the number of cache misses is added by one and then the cache entries are updated.

The MCP operation consists of the luminance and the chrominance parts. Since the luminance MCP operation is similar to the chrominance one, the decoding complexity of the whole MCP operation is in proportion to that of the luminance part. Thus, only luminance MCP is considered for simplicity, and the starting address and the size of memory access for luminance MCP are examined to compute the number of cache misses.

The luminance MCP for an MxN integer-pixel block can be viewed as the access to a 2D array with *N* rows and *M* columns, and it takes *N* times of memory accesses where each memory access is *M* bytes because each luminance component of pixel is usually represented as 8-bit integer. However, if the y-direction MV points to a non-integer pixel position, two additional rows in the top of the MxN block and three additional rows in the bottom of this block are needed for the y-direction interpolation filters. Similarly, if the x-direction MV is at a non-integer pixel position, two additional columns to the left and three additional columns to the right of the block are needed for the x-direction interpolation filters. Table 1 lists the number of memory accesses and its size for different MV positions.

TABEL I. NUMBERS OF MEMORY ACCESSES, MEMORY ACCESS SIZES AND STARTING ADDRESSES OF THE FIRST MEMORY ACCESS.

Position of MV=(x, y)	Number of memory accesses	Size per memory access	Starting address of the 1st memory access
(integer, integer)	N	M	addr
(non-integer, integer)	N	M+5	addr-2
(integer, non-integer)	N+5	M	addr-2*pic_width
(non-integer, non-integer)	N+5	M+5	addr-2*pic_width-2

For an integer-pixel block, the starting address of the i-th memory access can be easily computed from the MV, the reference frame index and the decoded frame resolution. Given *MV(x,y)* and the reference frame index, denoted by *ref_index*, the starting address of the first memory access, denoted by $addr_1$, can be computed by $addr_1 = addr$ and

$$addr = ref\_index \cdot width \cdot height + y \cdot width + x, \quad (2)$$

where *width* and *height* are the decoded frame resolution. Generally speaking, the starting address of the i-th memory access can be obtained by

$$addr_i = addr_1 + (i-1) \cdot width, i = 1,2,\dots .$$

The starting address of the first memory access is slightly different if MV points to a non-integer pixel position. If the y-direction MV points to a non-integer position, the starting address of the first memory access becomes

$$addr_1 = addr - 2 \cdot width ,$$

because two additional rows in the top of the block are needed for interpolation filters. Similarly, the starting address of the first memory access becomes

$$addr_1 = addr - 2 ,$$

if the x-direction MV points to a non-integer position. The starting address of the first memory access for different MV positions is summarized in Table 1.

The memory access whose size is greater than the size of cache entry can be treated as multiple memory accesses, and each of them has the memory access size equal to the size of cache entry. If required data cannot be found in the cache, then the number of cache misses is added by one and then cache entries are updated.

2.3 Computation of other relevant quantities

Similar to [5], the proposed decoding complexity model consists of a term which is proportional to the number of MVs. This quantity can be obtained according to the inter prediction mode. For example, P16x16 and Skip modes have only one MV but P16x8, P8x16 have two MVs. The maximal MV number per MB is 16, which is the case one 16x16 MB is portioned into 16 4x4 blocks.

The number of the x-direction (or the y-direction) interpolation filters can be computed as follows. For an MxN block, the number of the x-direction (or the y-direction) interpolation filters is $M \cdot N$ if only the MV along the x-direction (or the y-direction) is of subpel accuracy. However, if both x- and y-direction MVs are of subpel accuracy, the number of the x-direction (and the y-direction) interpolation filters is $(M+5)(N+5)$.

3. EXPERIMENTAL MODEL VERIFICATION

We conducted experiments to verify the proposed decoding complexity model given in (1) on the PC platform. The CPU was Pentium mobile 1.7 GHz CPU with 512 Mb RAM and the operating system was Windows XP. The reference JM9.4 decoder was optimized by the Intel MMX technology. We selected Foreman and Mobile sequences as training sequences and pre-encoded 70 training bit streams. Each bitstream file contained 270 frames. Among them, 42 sequences were coded with the integer-pel ME mode while 28 sequences were coded with the subpel ME mode. The Intel Vtune performance analyzer 8.0 was used to measure the MCP decoding complexities for all pre-encoded bit streams.

In our experiments, there were two different cache models for all inter prediction modes. For P4x8 and P4x4 blocks, the number of cache entries was 64 and the size of cache entry was 8 bytes. The number of cache misses was simply the number of memory accesses to blocks rather than P4x8 and P4x4, which is the case that the cache size is so small such that the required data of memory access cannot be obtained from the cache. This setting can result in good estimation for pre-encoded bit streams as well as others. The number of clock-ticks spent by the MCP operation was measured by the Intel Vtune and then the number of clock-ticks was divided by $1.7 \cdot 10^7$ to get the decoding time of MCP in milli-seconds. The proposed complexity model computed the number of cache misses, the number of x-direction and y-direction interpolation filters as well as the number of MVs per MB for those pre-encoded bit streams. Finally, the information was used to train the weight coefficients, i.e. $\alpha, \beta, \gamma,$ and μ . The constrained least square method was used to determine weight coefficients.

TABLE II. DECODING COMPEXITY (DECODING TIME IN MILLI-SECONDS) COMPARSION FOR CONTAINER CIF VIDEO SEQUENCE

Container (QP)	Actual complexity	Estimated complexity	Error (%)
1	187.42	174.58	6.85 %
5	172.39	164.09	4.81 %
10	153.89	148.54	3.48 %
15	117.42	116.14	1.09 %
20	98.45	96.66	1.82 %
25	78.39	73.02	6.86 %
30	54.12	52.10	3.73 %
35	44.52	40.45	9.14 %

TABLE III. DECODING COMPEXITY (DECODING TIME IN MILLI-SECONDS) COMPARSION FOR MOBILE CIF VIDEO SEQUENCE

Mobile (QP)	Actual complexity	Estimated complexity	Error (%)
1	251.22	232.97	7.27 %
5	240.88	225.00	6.59 %
10	238.70	224.31	6.03 %
15	232.42	218.6.0	5.95 %
20	219.40	215.82	1.63 %
25	211.31	208.64	1.26 %
30	203.42	199.80	1.78 %
35	185.94	181.80	2.23 %

TABLE IV. DECODING COMPEXITY (DECODING TIME IN MILLI-SECONDS) COMPARSION FOR SILENT CIF VIDEO SEQUENCE

Silent(QP)	Actual complexity	Estimated complexity	Error (%)
1	197.88	187.83	5.03 %
5	197.37	191.70	2.87 %
10	183.80	176.31	4.08 %
15	128.43	139.00	8.23 %
20	116.84	127.02	8.71 %
25	103.13	113.01	9.57 %
30	93.52	101.02	7.99 %
35	81.36	89.01	9.39 %

The weight coefficients are:

$$\alpha = 2.412 \times 10^{-5} ,\ \beta = 4.861 \times 10^{-6} ,$$

$$\gamma = 2.675 \times 10^{-7} ,\ \mu = 9.656 \times 10^{-5} .$$

Next, these weight coefficients trained by Foreman and Mobile sequences were adopted by the proposed complexity model to estimate the decoding complexities of the following test bit streams: three CIF video sequences, Container, Mobile and Silent, and each of them was encoded with a set of quantization parameters (QP), QP=1, 5, 10, 15, 20, 25, 30 and 35, where the non-integer pixel ME was enabled for these test sequences. Tables II, III and IV show the comparisons between the estimated decoding complexity based on the proposed complexity model and the actual decoding complexity measured by the Intel Vtune. We see that the proposed decoding complexity model provides good estimation results for the test bit streams. The errors are within 10%.

4. APPLICATIONS OF PROPOSED COMPLEXITY MODEL

The application of the decoding complexity model is briefly discussed in this section. A more thorough treatment will be given in our future work. Consider the scenario that an H.264 encoder generates a single bit stream for different decoding platforms without the use of any decoding complexity model. In contrast, the encoder may generate several bit streams for different decoding platforms separately according to their computational powers so that the resultant bit stream is easy to be decoded for a particular platform. For the latter case, the decoding complexity models have to be integrated into the H.264 encoder so that the encoder can estimate the possible decoding complexity and then generate decoder-friendly bit streams.

In the conventional H.264 encoder, the rate-distortion optimization (RDO) process is used to decide the optimal inter prediction mode that minimizes the RD cost for the MB as shown in Fig. 2 (a). It consists of three steps. First, since different inter prediction modes have a different number of MVs, the RDO process first finds the best MV for a specific inter prediction mode. Second, the RDO process performs the encoding processes, which include the spatial domain transform, quantization and entropy encoding, and then performs the decoding processes to get reconstructed video frame so that the distortion and the bit rate can be obtained. Finally, the RDO process finds the best inter prediction mode that yields the minimal RD cost for the MB.

Since the proposed complexity model requires MV only for a specific inter prediction mode, the estimated decoding complexity can be computed if the weight coefficients are given. The proposed complexity model can be integrated in Step (2) in the RDO process as shown in Fig. 2 (b). The estimated decoding complexity can be computed and the RDO process can skip those inter prediction modes whose decoding complexities are higher than that can be offered by the target decoding platform. In other words, the RDO process shown in Fig. 2 (b) searches the best mode among a smaller set of inter prediction modes whose decoding complexities are less than the decoding complexity constraint. As a result, the RDO process with the decoding complexity model can select the best inter prediction mode that minimizes the RD cost as well as meets the decoding complexity constraint for the target platform. Another advantage for the RDO process shown in Fig. 2 (b) is that the encoding complexity can be saved since the encoder does not have to perform the encoding and the decoding processes for all inter prediction modes. Please note that the weight coefficients are different among different platforms and, therefore, they have to be measured for the target decoding platform first so that the RDO process with the decoding complexity model can estimate the decoding complexity for a given inter prediction mode and MV and then select the optimal inter prediction mode for the MB.

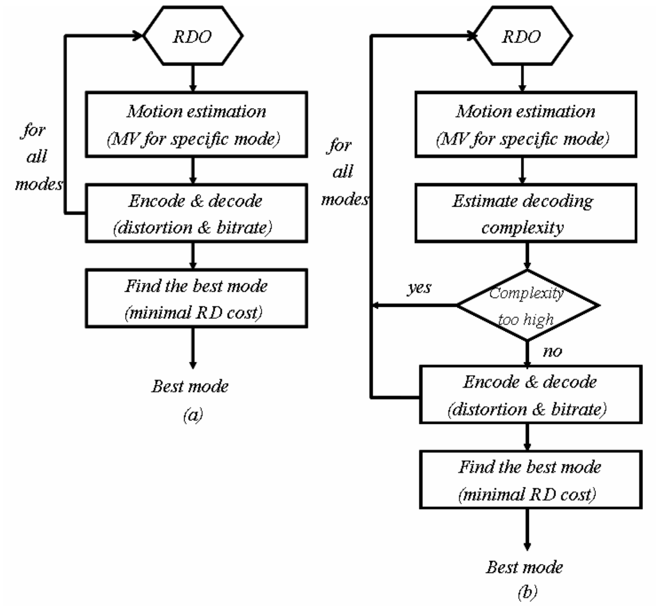
RDO
Motion estimation (MV for specific mode)
Encode & decode (distortion & bitrate)
Find the best mode (minimal RD cost)
for all modes
Best mode
(a)
RDO
Motion estimation (MV for specific mode)
Estimate decoding complexity
Complexity too high
yes
no
Encode & decode (distortion & bitrate)
Find the best mode (minimal RD cost)
for all modes
Best mode
(b)

Figure 2. RDO process and RDO process with integrated decoding complexity model.

5. CONCLUSION

The complexity model for motion compensation in the H.264/AVC decoder was examined in this work. This model helps the encoder select proper inter prediction modes and then generate a video bit stream that is most suitable for a particular platform with certain hardware constraint. As a result, the coded bit stream can balance the tradeoff between the RD requirement as well as the computational power of the target decoding platform. The proposed decoding complexity model was verified experimentally. We see that the model provides a fairly good estimation results for various test bit streams. Finally, possible applications of the decoding complexity model, including its integration with the H.264 encoder, were discussed.

6. REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra," Overview of the H.264/AVC coding standard," IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[2] "Information technology – coding of audiovisual objects – Part 2: Visual," Dec. 1999.

[3] J. Valentim, P. Nunes and F. Pereia, "An alternative complexity model for the MPEG-4 video verifier mechanism," IEEE International Conference on Image Processing 2001, pp. 461-464.

[4] J. Valentim, P. Nunes and F. Pereia, "Evaluating MPEG-4 video decoding complexity for an alternative video complexity verifier model," IEEE Trans. on Circuits and Systems for Video Technology, pp. 1034-1044, vol. 12, no. 11, Dec. 2002.

[5] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," IEEE Trans. on Multimedia, pp. 471-479, vol. 7, no. 3, June 2005.

[6] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 7, pp.704-716, July 2003.