# An Adaptive Motion Data Storage Reduction Method for Temporal Predictor

Ruobing Zou, Oscar C. Au, Lin Sun, Sijin Li, and Wei Dai

Department of Electronic and Computer Engineering,
Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
{zouruobing,eeau,lsunece,sliae,weidai}@ust.hk

**Abstract.** In the state-of-art video coding standard HEVC, temporal motion vector (MV) predictor is adopted in order to improve coding efficiency. However, motion vector information in reference frames, which is used by temporal MV predictor, takes significant amount of bits in memory storage. Therefore motion data needs to be compressed before storing into buffer. In this paper we propose an adaptive motion data storage reduction method. First, it divides the current 16x16 block in the reference frame into four partitions. One MV is sampled from each partition and all sampled MVs form a MV candidate set. Then it judges if one or two MVs should be stored into the MV buffer by checking the maximum distance between any two of the MVs in the candidate set. If the maximum distance is greater than a certain threshold, the motion data of the two MVs that have maximum distance are put into memory; otherwise the motion data of the upper left block is stored. The basic goal of the proposed method is to improve the accuracy of temporal MV predictor at the same time reducing motion data memory size. Simulation results show that compared to the original HEVC MV memory compression method in the 4th JCT-VC meeting, the proposed scheme achieves a coding gain of 0.5%~0.6%; and the memory size is reduced by more than 87.5% comparing to without using motion data compression.

**Keywords:** HEVC, MV data storage, MV compression, temporal predictor.

## 1 Introduction

The Motion Pictures Experts Group (MPEG) and the International Telecommunications Union's Telecommunication Standardization Sector (ITU-T) recently have formed a joint collaborative team on video coding (JCT-VC). The JCT-VC is aiming at designing a next generation video coding standard. This standard, currently named as High Efficiency Video Coding (HEVC), targets substantial coding efficiency improvement compared to state-of-the-art video coding standards such as MPEG-4 and H.264/AVC [1]. In order to evaluate different coding techniques, a software platform called HEVC Test Model (HM) is developed, which contains selected coding tools. In HEVC, the MVs of reference frames are used in the prediction process of a current

Y.-S. Ho (Ed.): PSIVT 2011, Part II, LNCS 7088, pp. 48–59, 2011.

frame as temporal MV candidates, such as in Advanced Motion Vector Prediction (AMVP) and in PU-level Merge mode [2]. Therefore, temporal information needs to be stored in a buffer, which occupies a portion of memory size. In order to save the memory for storing MV data of the reference frames, a MV memory compression method was adopted by the 4th JCT-VC meeting and written into the HEVC working draft [3]. However, as reported in [2], this MV memory compression method results in a coding loss of 0.6%~0.8% in terms of BD Bit Rate (BD-rate) [4], compared with turning off MV compression tool. JCT-VC recently formed a new Core Experiment to study the problem of motion data storage. Under this background, we propose a novel motion data storage reduction scheme which can reduce the memory size while improving coding efficiency.

The remainder of the paper is organized as follows. In Section 2, we generally describe inter coding tools in HEVC, to which temporal predictor is applied, and then we explain how MV data is compressed and stored. In Section 3, the proposed method is introduced. Results on HEVC Test Model HM2.0 software are presented in section 4. Conclusions are drawn in section 5.

## 2 Motion Data Storage Reduction

### 2.1 Prediction Unit (PU) Merge Mode and Advanced Motion Vector Prediction (AMVP)

In terms of HEVC, two different motion information prediction schemes exist in the inter frame coding. One of the motion information prediction schemes is merge mode. Merge mode is performed on PU level, where PU is the basic unit used for carrying the information that is related to the prediction processes. The merge mode infers that the inter prediction direction and the reference frame index of the current PU are the same as one of the neighbouring blocks and uses their MVs as predictors. Merge candidates are four spatial predictors and one temporal predictor, as shown in Fig. 1(a). Temporal predictor is the MV of the co-located PU partition, labeled "Col" in Fig. 1 (a) and (b). Derivation of the temporal predictor will be explained in detail later.

The other prediction scheme is AMVP which uses MVs from the neighbouring PUs as MV predictors [5]. There are totally three possible MV predictors: two spatial predictors "Left" and "Above", as well as a temporal predictor. AMVP "Left" predictor is the first 4x4 neighbour found along the left edge of the current PU, while the "Above" predictor is the first available 4x4 neighbour found along the top edge, as illustrated by Fig. 1(b). We will further describe the temporal MV predictor in below.

### 2.2 Temporal Motion Vector Predictor

In both the merge mode and AMVP, the MVs of reference frames are used as the temporal predictors to improve coding efficiency. As determined in the 4th HEVC

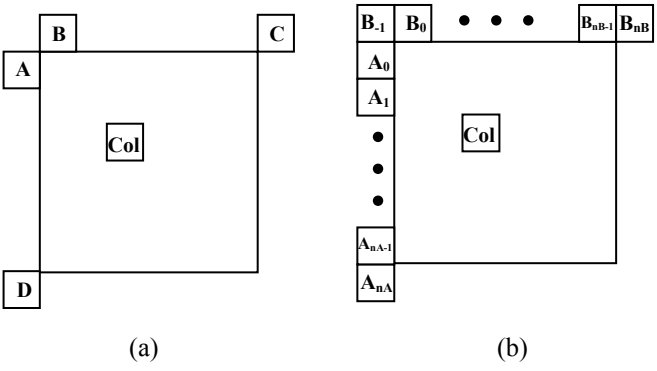


**Fig. 1.** Predictors for Merge mode and AMVP. (a) Neighboring blocks checked by merge mode, including the temporal predictor (Col). (b)Search range for left predictor ($A_0$ to $A_{nA}$), above predictor ($B_{-1}$ to $B_{nB}$) and co-located temporal predictor (Col).

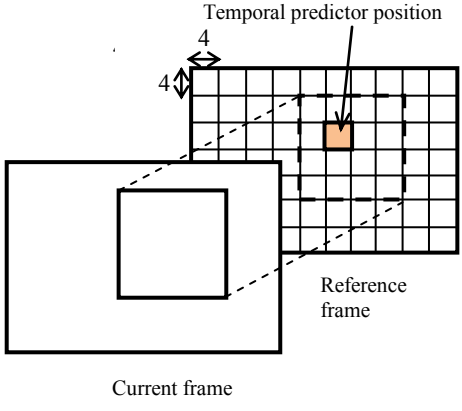


**Fig. 2.** Position of the center temporal predictor

meeting, the temporal predictor is found by mapping the upper left position of the center of the current partition to a co-located block in reference frame, and the MV of the corresponding 4x4 block in the co-located block is used as a temporal MV predictor [6]. By using the temporal MV predictor, it shows that the average 2.2% of BD-rate saving is achieved for random access and low delay test conditions [7].

### 2.3 Reduced Resolution Storage of Motion Vector Data

Despite of the BD-rate saving brought by the temporal MV predictor, the usage of the temporal MV leads to the need of additional memory requirement in inter prediction because the MV data information of the reference frames needs to be stored. Without

compression, each 4x4 block has its own MV, which needs significant buffer size, considering the granularity of motion representation and considering that there are two vectors per block for B slice. To give an example, it is currently estimated the buffer size to be approximately 26Mbits/frame for 4kx2x application, though this size will ultimately depend on the precision and maximum MVs supported [8]. Large amount of MVs to be stored causes the internal memory size to increase, which may result in increasing hardware cost and power consumption [9].

In order to reduce the motion data storage, a memory compression method in [8] was adopted at 4th JCT-VC meeting, in which a lower resolution MV buffer is used. It means that within a predefined 16x16 block, only one MV in the top left 4x4 block is saved into the buffer, instead of sixteen different MVs, as illustrated by Fig. 3 (a) and (b). Indeed not only MV fields but also reference frame indices and modes of the reference frame (inter/intra) are used in temporal MV prediction and thus they are compressed in the same method as well. The saved MV is used as a representative MV of the current 16x16 block, and when a reference frame is referred as a co-located block for the temporal predictor, the representative MV is read from the buffer and then assigned to all 4x4 blocks within the current 16x16 block (Fig. 3(c)). Therefore, both the required memory size and the memory access bandwidth for the temporal MV data are reduced to 1/16.

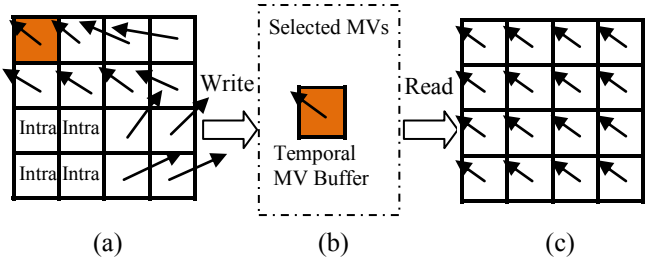


**Fig. 3.** Illustration of the MV compression in HEVC. (a) Motion vectors before compressed. (b) After MV compression. (c) In referring as co-located block for temporal predictor. [10]

## 3 Proposed Scheme

In this section, we first analyze the HEVC MV compression scheme introduced before and show its disadvantage in some cases. Then we propose an adaptive MV compression scheme.

### 3.1 Analysis of the Limitations of Storing Upper Left Block MV Information

To illustrate the limitation of the original memory compression method in HEVC, we examine the sixteen MVs in a current 16x16 block before compression. Suppose that

the current block lies inside an object which has translational motion. Under this situation nearly all 4x4 blocks have similar MV, therefore, MV of the upper left 4x4 block (denoted by $mv_0$) works well as a representative of the current block. However, the current scheme ignores that there could be two or more objects located inside a 16x16 block. If these two objects have different movement speed and directions, then simply assigning $mv_0$ to all blocks may result in a large deviation from a block's real MV, consequently causing reliability degradation of the temporal predictor and introducing coding loss.

### 3.2 Proposed Motion Data Storage Reduction Scheme

In order to reduce the coding loss, it is natural to consider storing more MVs into memory. However, increasing data in the storage inevitably leads to lower compression rate. Therefore it is crucial to achieve a good trade-off between these two factors. Based on this idea, we propose a novel method which reduces the MV buffer size by adaptively choosing to store either one or two MVs from the sixteen MVs of the current block. The details of the proposed algorithm can be depicted as follows.

The proposed scheme first divides the current 16x16 block into four partitions. As depicted in Fig. 4, partition 1 to 4 is shown with horizontal shading, down diagonal shading, up diagonal shading and vertical shading separately. Each partition contains four 4x4 blocks and thus it has four MVs. We denote the MV of block $n$ as $mv_n$, $\forall n \in \{0,\ldots,16\}$. The MV set of the partition $k$ is denoted as $N_k$, $\forall k \in \{1,\ldots,4\}$. If a block's MV is not available, when the block is in intra mode or out of the slice boundary, then its MV is set to be (0,0).

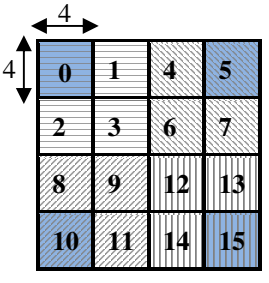


**Fig. 4.** Partitions and sampled MVs in proposed scheme. Partition 1 contains block 0, 1, 2 and 3; partition 2 contains block 4, 5, 6 and 7; partition 3 contains block 8, 9, 10 and 11; partition 4 contains block 12, 13, 14 and 15.

Following the partitioning, a sampling is conducted on every MV set $N_k$. One MV is sampled out of four and becomes a member of set $P$, which is a set of MV candidates that might be put into the buffer. All of the sampled MVs are on the corner of the current 16x16 block, and thus the sampled MV of partition $k$ is $mv_{5(k-1)}$, $k \in$

{1,…,4}, as shown in dark colour shading in Fig. 4. In this way, the MV candidate set $P = \{ mv_0, mv_5, mv_{10}, mv_{15} \}$ is obtained. The reason why blocks on the corner are chosen is because these blocks are far from each other thus they are relatively with low correlation. This is particularly true when two or more objects exist, whose MVs pointing to different directions. It ensures that different movement conditions are taken into consideration simultaneously.

With the candidate set $P$, Euclidean distance among any two MVs in $P$ is calculated as follows:

$$d(i, j) = \| mv_i - mv_j \|, \tag{1}$$

Where $mv_i, mv_j \in P$, $i \neq j$ and $i, j \in \{0, 5, 10, 15\}$. We calculate the Euclidean distance because x-component and y-component of vectors are considered together. The maximum distance $d_{max}$ is used in checking whether to store one or two set(s) of MV data as follows. Suppose $mv_a, mv_b \in P$ satisfy:

$$d(a, b) = \| mv_a - mv_b \| = d_{max},$$

Where $mv_a$ is the sampled MV of partition A, $mv_b$ is the sampled MV of partition B, $a \neq b$ and A, B $\in \{1,\ldots,4\}$. The basic rule can be described as:

$$\text{MVs stored} = \begin{cases} mv_0, & \text{if } d_{max} \leq T, \\ mv_a \text{ and } mv_b, & \text{if } d_{max} > \text{T}, \end{cases} \tag{2}$$

If $d_{max}$ is smaller than a certain threshold T, MVs in $P$ are considered to be similar. In this case, using only one MV can well represent all MVs in the entire 16x16 block. Hence, MV information of the upper left block, namely $mv_0$, frame indices and modes, is stored into the MV buffer, which is basically the same as the memory compression method in HEVC. Besides, a one-bit MV_num_flag is set to 1 as a mark of putting only one MV into buffer.

On the other hand, if $d_{max}$ is greater than T, it means that at least one partition has MV deviating from others, so that the algorithm adaptively stores both $mv_a$ and $mv_b$ into the MV buffer, along with their frame indices and modes. Similarly, the MV_num_flag is set to 0 indicating that two MVs are selected. In addition, we store a one-bit flag for every partition so as to indicate that it uses either $mv_a$ or $mv_b$ as its representative. Specifically, we set the flag of partition A (flag_A) to be 1, indicating that it uses $mv_a$ as representative; and flag of partition B (flag_B) to be 0. With regard to the other two partitions, each of them also needs to select a representative MV from the stored MVs in the following method. Suppose these two partitions are partition C and D, with sampled MV $mv_c$ and $mv_d$, where $mv_c, mv_d \in P$, $a \neq b \neq c \neq d$ and C, D $\in \{1\ldots4\}$. For each of $mv_c$ and $mv_d$, we check its distance to $mv_a$ and $mv_b$. Take partition C as an example, the representative MV (rMV) of partition C equals to:

$$\text{rMV} = \begin{cases} mv_a, & \text{if } d(c,a) < d(c,b), \\ mv_b, & \text{if } d(c,a) \geq d(c,b), \end{cases} \tag{3}$$

Meanwhile, a flag is set to indicate which stored MV is used as representative of partition C. To be specific, if $mv_a$ is selected as the representative, the flag of partition C (flag_C) is set to be 1; otherwise the flag equals to 0. The same process is performed on $mv_d$ according to equation (3). A flowchart of the whole process of motion data compression is shown in Fig. 6.

So far, we have introduced how motion data is adaptively selected and compressed into the MV buffer. After that, when the 16x16 block is referred as co-located block in temporal prediction, the MV(s) stored will be read from buffer. If two MVs are stored, they are assigned to the partitions as representative according to the flags we previously stored; otherwise the single stored MV is assigned to all partitions. Fig. 5 (a) ~ (e) shows an example of how to store MV data adaptively based on different situation and how to assign MV(s) that has/have been read from buffer.

As a summary, the memory size of the proposed method is bigger than HEVC's method, whose memory size is reduced to 6.25% and smaller than constantly storing two MVs, whose memory size is 12.5%. So the total reduction of the memory size is more than 87.5% comparing to not using MV compression.

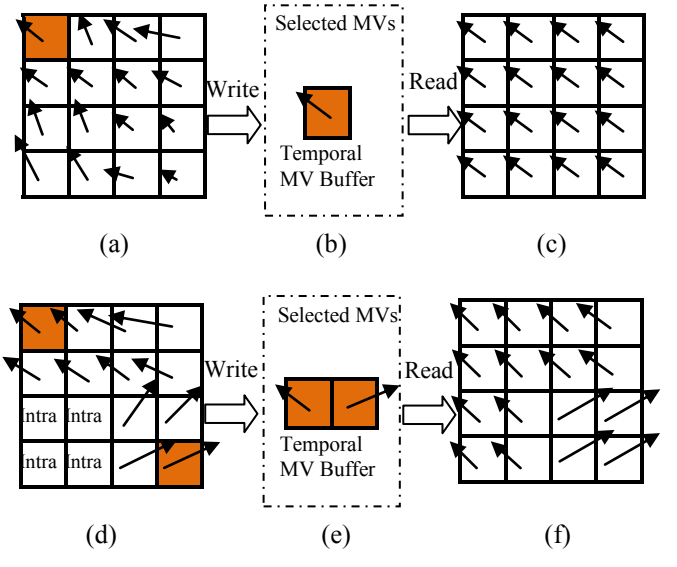


**Fig. 5.** Illustration of proposed motion data compression scheme. (a) ~ (c) Illustrates the case of storing one MV, (d) ~ (f) Illustrates the case of storing two MVs. (a) (d) Motion vectors before compressed. (b) (e) After motion data compression. (c) (f) In referring as co-located block for temporal predictor.

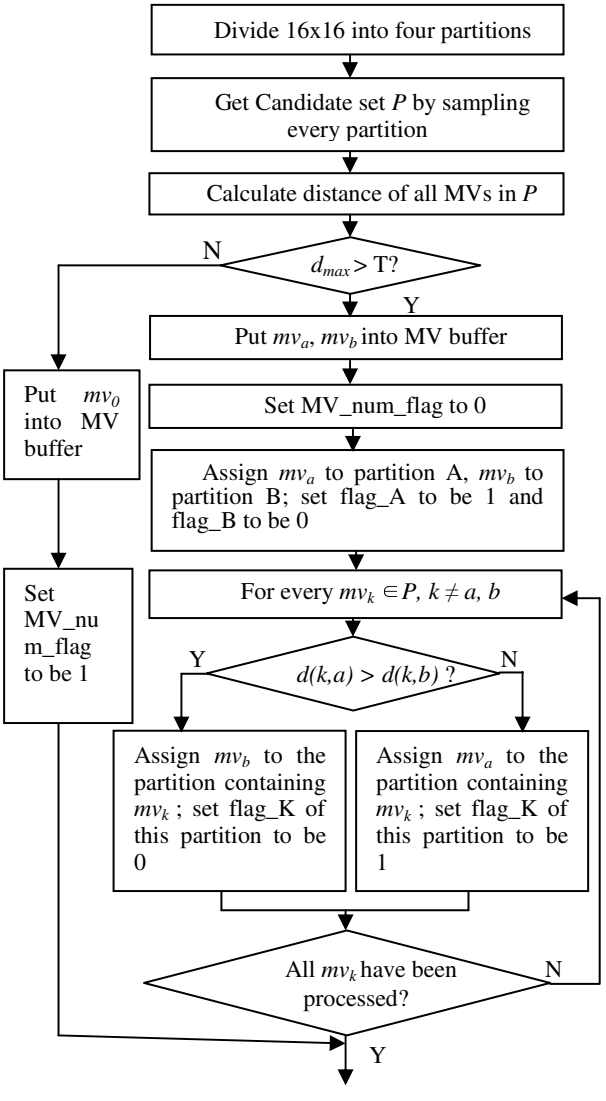


**Fig. 6.** Detailed process of motion data compression

## 4 Experimental Results

To measure the performance of the proposed motion data compression technique, we compare the performance of proposed scheme with HEVC MV compression scheme described in Section 2.

### 4.1 Test Conditions

We execute experiments on HM software version 2.0, which is the latest version of HEVC Test Model created following the decisions taken at the 4th meeting of JCT-VC in Daegu in January 2011. The platform of experiment is Intel i7 CPU 860 @2.80GHz with 16GB RAM. Video sequences to be tested are provided by JCT-VC, including five classes of sequence: Class A, Class B, Class C, Class D and Class E, with resolution 2560x1600, 1920x1080, 832x480, 416x240 and 1280x720 pixels. For each video sequence four quantization parameter values are used: 22, 27, 32 and 37.

Our experiment is executed with common test conditions in HEVC to make sure of conducting experiments in a well-defined environment and to ease the comparison of the outcome of experiments [11]. Four common test conditions are used in inter prediction, reflecting a combination of high efficiency (HE) and low complexity (LC), and of random-access (RA), and low-delay (LD) settings:

- Random access, high efficiency (RA-HE)
- Random access, low complexity (RA-LC)
- Low delay, high efficiency (LD-HE)
- Low delay, low complexity (LD-LC)

Noted that when testing MV compression tools, low-delay configurations should be skipped for class A, and random-access configurations should be skipped for class E. Table 1 gives the specifications of sequences used for random-access, and low-delay conditions.

**Table 1.** Basic information of the test sequences

| Class | Sequence name | Frame count | Frame rate | RA | LD |
|---|---|---|---|---|---|
| A (4k) | Traffic | 150 | 30fps | Y | N/A |
| | PeopleOnStreet | 150 | 30fps | Y | N/A |
| B (1080p) | ParkScene | 240 | 24fps | Y | Y |
| | Cactus | 500 | 50fps | Y | Y |
| | Kimono | 240 | 24fps | Y | Y |
| | BasketballDrive | 500 | 50fps | Y | Y |
| | BQTerrace | 600 | 60fps | Y | Y |
| C (WVGA) | RaceHorses | 300 | 30fps | Y | Y |
| | BasketballDrill | 500 | 50fps | Y | Y |
| C (WVGA) | BQMall | 600 | 60fps | Y | Y |
| | PartyScene | 500 | 50fps | Y | Y |
| D (WQVGA) | RaceHorses | 300 | 30fps | Y | Y |
| | BasketballPass | 500 | 50fps | Y | Y |
| | BQSquare | 600 | 60fps | Y | Y |
| | BlowingBubbles | 500 | 50fps | Y | Y |
| E (WQVGA) | Vidyo1 | 600 | 60fps | N/A | Y |
| | Vidyo3 | 600 | 60fps | N/A | Y |
| | Vidyo4 | 600 | 60fps | N/A | Y |

Tests are executed by separately turning on a tool of HEVC MV compression, and turning on a tool of the proposed method. When implementing the latter experiment, we use the square of MV distance instead of MV distance because calculating square root increases complexity. As for choosing the threshold T, when the distance between two vectors is more than the maximum distance in a 4x4 block, we take them as deviating from each other and store both of them. So the square of T is set to be 32, which is the largest distance within a 4x4 block. Also, noted that the HEVC working draft stated a bug in HM 2.0 [12], however, the current software does not exactly implement this. We fixed this bug in our experiment, i.e. reference index and mode is also decimated by taking the values that apply to the top left pixel position of each 16x16 block, as motion vector field does.

**Table 2.** Coding performance of the proposed scheme

| Class | RA-HE | | | RA-LC | | |
|---|---|---|---|---|---|---|
| | BD-rate | | | BD-rate | | |
| | Y | U | V | Y | U | V |
| A | -0.7 | -0.6 | -0.5 | -0.6 | -0.6 | -0.6 |
| B | -0.2 | -0.2 | -0.2 | -0.2 | -0.1 | -0.1 |
| C | -0.4 | -0.4 | -0.4 | -0.4 | -0.3 | -0.4 |
| D | -0.6 | -0.6 | -0.6 | -0.8 | -0.6 | -0.6 |
| E | | | | | | |
| All | -0.5 | -0.5 | -0.4 | -0.5 | -0.3 | -0.4 |
| Enc Time[%] | 102 | | | 101 | | |
| Dec Time[%] | 101 | | | 100 | | |

| Class | LD-HE | | | LD-LC | | |
|---|---|---|---|---|---|---|
| | BD-rate | | | BD-rate | | |
| | Y | U | V | Y | U | V |
| A | | | | | | |
| B | -0.3 | -0.2 | -0.3 | -0.4 | -0.4 | -0.3 |
| C | -0.6 | -0.4 | -0.6 | -0.7 | -0.7 | -0.7 |
| D | -0.6 | -0.3 | -0.8 | -0.6 | -0.6 | -0.5 |
| E | -0.7 | -0.6 | -0.2 | -0.5 | -0.5 | -0.9 |
| All | -0.6 | -0.4 | -0.5 | -0.6 | -0.4 | -0.7 |
| Enc Time[%] | 102 | | | 101 | | |
| Dec Time[%] | 101 | | | 100 | | |

### 4.2 Results and Analysis

After the performance of MV compression method in HEVC and that of the proposed method are both tested, their test results are compared in terms of BD-rate [4], which is computed by Excel sheet provided by JCT-VC. The negative value of BD-rate

indicates a coding gain of the proposed method, comparing with the HEVC MV compression method in HM 2.0.

Experimental results are shown in Table 2; it is observed that the proposed method shows 0.5%, 0.5%, 0.6% and 0.6% BD-rate gain, which indicates consistently improvement in all test conditions. In random access cases, the proposed method shows 0.5% of BD-rate saving than the HEVC method for high efficiency, and 0.5% of BD-rate saving than HEVC method for low complexity on average. While in low delay case, the proposed method saves 0.6% of BD-rate saving for high efficiency and 0.6% for low complexity on average. Meanwhile, the difference in encoding and decoding time is relatively small.

In order to show the gap between with and without MV compression, we also cite the experimental results in [2], which tested on the HM2.0 software in two conditions: turning on and turning off with HEVC MV compression tool. The results show that without MV compression, it achieves 0.6%, 0.7%, 0.8% and 0.8% BD-rate gain for RA-HE, RA-LC, LD-HE and LD-LC configuration, which means that turning on the tool brings a coding loss of 0.6%, 0.7%, 0.8% and 0.8% BD-rate. It can be seen from all above results that the performance loss taken by the MV compression in HEVC can be mostly compensated by using the proposed MV data storage scheme. The proposed motion data storage reduction scheme keeps the loss to 0.1%~0.2%.

The advantage of the proposed method over the original HEVC MV compression method is that it adds only one MV to temporal MV buffer adaptively in certain cases, but it in turn brings relatively high coding gain than the original method. The gain comes mainly from the fact that our method effectively reduced the inaccuracy of the temporal predictor due to MV compression.

## 5 Conclusion

In this paper, a novel motion data storage reduction method is proposed. In the proposed method, for every 16x16 block in a reference frame, it judges if an additional memory area should be used for motion data compression by checking the maximum MV distance within the current block, and then it adaptively selects one or two MVs to store into the MV buffer. By doing so, the proposed method avoids a waste of memory caused by always storing two MVs; at the same time, it also reduced the inaccuracy of temporal MV predictor caused by constantly storing one MV. Compared with the memory compression method in HEVC, coding gain can be observed in terms of BD-rate saving, which is 0.5%, 0.5%, 0.6% and 0.6% for RA-HE, RA-LC, LD-HE and LD-LC configuration. In other words, the proposed scheme improves coding efficiency for various video sequences than the MV compression method in HEVC, while reducing motion data memory size by more than 87.5% comparing to without motion data compression. In the recently released software HM 3.0, the position of temporal MV predictor was changed to the right corner of the current PU. Our further research will include finding an effective MV data storage method in the new condition.

## References


1. Segall, A., Zhao, J., Yamamoto, T.: Parallel Intra Prediction for Video Coding. In: Picture Coding Symposium, Nagoya, Japan, pp. 310–313 (2010)
2. Guo, X., Lin, J., Huang, Y.W., Lei, S.: Motion Vector Decimation for Temporal Prediction. In: JCT-VC 5th Meeting, JCTVC-E092, Geneva (2011)
3. WD2: Working Draft 2 of High-Efficiency Video Coding. In: JCT-VC 4th Meeting, CTVC-D503, Daegu, Korea (2011)
4. Bjontegaard, G.: Calculation of Average PSNR Differences Between RD Curves. In: ITU-T SC16/Q6, VCEG-M33, Austin, USA (2004)
5. Yeo, C., Tan, Y.H., Li, Z.: Simplified AMVP Candidate Derivation For Inter and Merge Modes. In: JCT-VC 5th Meeting, JCTVC-E101, Geneva (2011)
6. Park, S., Park, J., Jeon, B.: Modifications of Temporal MV Compression and Temporal MV Predictor. In: JCT-VC 5th Meeting, JCTVC-E059, Geneva (2011)
7. Lim, S.C., Kim, H.Y., Kim, J., Lee, J., Choi, J.S.: Dynamic Range Restriction of Temporal Motion Vector. In: JCT-VC 5th Meeting, JCTVC-E142, Geneva (2011)
8. Su, Y., Segall, A.: CE9: Reduced Resolution Storage of Motion Vector Data. In: JCT-VC 4th Meeting, JCTVC-D072, Daegu, Korea (2011)
9. Choi, J., Kim, J.: Motion Vector Memory Reduction Scheme for Scalable Motion Estimation. Optical Engineering 48(9) (2009)
10. Fujibayashi, K., Bossen, F.: CE9 3.2d Modified Motion Vector Compression Method. In: JCT-VC 5th Meeting, JCTVC-E231, Geneva (2011)
11. Bossen, F.: Common Test Conditions and Software Reference Configurations. In: JCT-VC 4th Meeting, JCTVC-D600, Daegu, Korea (2011)
12. Meeting Report of The Fifth Meeting of The Joint Collaborative Team on Video Coding. In: JCT-VC 5th Meeting, Geneva (2011)
13. Wang, R., Li, J., Huang, C.: Motion Compensation Memory Access Optimization Strategies For H.264/AVC Decoder. In: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, vol. 5, pp. 97–100 (2005)