PROGRESSIVE MOTION VECTOR RESOLUTION FOR HEVC

Juncheng Ma[1], Jicheng An[2], Kai Zhang[2], Siwei Ma[1], Shawmin Lei[2]

{[1]jcma, [1]swma}@pku.edu.cn
[1]Institute of Digital Media, Peking University, Beijing 100871, China
{[2]jicheng.an, [2]kai.zhang, [2]shawmin.lei}@mediatek.com
[2]North Building 10F, Raycom Infotech Park Tower C, No. 2 Kexueyuan South Rd., Haidian District, Beijing 100190, China

ABSTRACT

This paper proposes a progressive motion vector resolution (PMVR) method for High Efficiency Video Coding (HEVC). In the proposed scheme, high motion vector (MV) resolutions, e.g. 1/4 or 1/8 pixel resolution, are employed for MVs near to the motion vector predictor (MVP) and low MV resolutions are employed for MVs far from the MVP. The range of each MV resolution is indicated by a threshold parameter. And a new motion vector difference (MVD) derivation method is designed to encode MVD efficiently. Experimental results show that PMVR with 1/8 pixel motion search can achieve a BD-rate gain up to 16% with almost the same coding time with HM8.0, and for PMVR without 1/8 pixel motion search, up to 6.1% BD-rate gain can be achieved with 9% encoding time saving on average.

Index Terms— video coding, HEVC, MV resolution, MVD

1. INTRODUCTION

During the past decade, the rapid development of digital video compression technology has promoted the prosperity of digital video industry, including digital TVs, digital cameras, etc. The latest video coding standard HEVC [1] is developed by the Joint Collaborative Team on Video Coding (JCT-VC), which is formed by the ISO/IEC MPEG and ITU-T VCEG standardization organizations. So far, HEVC has been released, and it can achieve 50% or even more bits saving compared to H.264/AVC [2] with comparable visual quality.

Motion prediction and motion vector (MV) coding plays a key role in video coding process. Usually, higher resolution MV can achieve more accurate motion compensation, but the problem is high resolution MV will also increase the coding overhead of MV. So it is very important to achieve a good balance between the motion vector resolution and the coding overhead. In the state-of-the-art video coding standards, usually 1/4 pixel MV resolution is used for luma prediction. Moreover, to reduce the bits used for MV coding, a motion vector predictor (MVP) is derived from the spatial and temporal neighboring blocks, and MV is actually coded as the difference between the MV and MVP. In HEVC, an advanced motion vector prediction (AMVP) scheme is designed, in which a motion vector candidates set is constructed from the spatial neighboring blocks and the temporal co-located block in the reference picture, and the best MVP is selected with the optimal rate distortion cost. Moreover, a "merge" mode is added for inter prediction, allowing the inheritance of MVs from the neighboring prediction units (PUs) [3][4], and skip mode is actually a special case of merge mode when the residuals of current block are all equal to zero.

In the standardization process of HEVC, the MV coding has been widely studied. In JCTVC-A121 [5], an adaptive motion vector resolution (AMVR) method is proposed, in which MV resolution is adaptively selected between 1/4 and 1/8 pixel for each block. To signal the MV resolution to the decoder, additional parameters have to be coded at block level. Therefore, the performance of AMVR is degraded due to the increasing overhead.

In this paper, we propose a progressive motion vector resolution (PMVR) method for HEVC. In PMVR, the MV resolution can be progressively decreased with the increasing magnitude of the MVD. The range of higher MV resolution is controlled by thresholds signaled at slice level, which can reduce the overhead coding compared with AMVR. Moreover, an efficient MVD coding method is proposed, to improve the coding efficiency further.

The rest of the paper is organized as follows. Section 2 presents an overview of motion vector coding techniques in HEVC. Section 3 describes the proposed PMVR method. Experimental results and analysis are presented in Section 4. Finally Section 5 concludes the paper.

2. OVERVIEW OF MOTION VECTOR CODING TECHNIQUES IN HEVC

In HEVC, two inter-prediction modes are employed for inter coding blocks: inter mode and merge mode. For inter mode, the advanced motion vector prediction (AMVP)

method is used for MVP derivation. The AMVP candidates include two spatial MV candidates and one temporal MV candidate in each reference frame, and the best one is selected with rate-distortion optimization (RDO) decision [6]. The index of the optimal MV candidate and the reference picture are explicitly transmitted to the decoder. Generally, motion estimation is performed around the selected MVP, and the difference between the MVP and the actual MV is also signaled in the bit stream. For merge mode, however, the motion information is directly inherited from one of the spatial neighboring blocks or a temporal co-located block on the reference picture with minimum POC difference from current picture within reference picture list 0 or list 1. The reference picture list flag is signaled in the slice header. Therefore, no motion information is needed for the decoder but a "merge index" is coded to indicate which candidate to be used. Particularly, the skip mode is actually a special case of merge mode when the residuals of current block are all equal to zero.

Given the reference picture list and index, the spatial and temporal MV prediction candidates are shown in Fig. 1. For AMVP, the two spatial candidates are derived based on the five spatial neighbor blocks (A0, A1, B0, B1 and B2). The first MV candidate is selected from A0 and A1 in order, and the second one from B0, B1 and B2 in order. If a neighbor block is unavailable or intra coded, then the MV candidate corresponding to this block is ignored. The temporal MV candidate is scaled from the MV of the co-located block based on the POC distances, as shown in Fig. 2. The co-located block is T0 by default if it is available, inside the current coding tree unit (CTU) and not intra coded, otherwise T1 will be selected. Finally, if the number of the selected MV candidates is less than two, zero MV candidates will be added to make sure it equals to two.

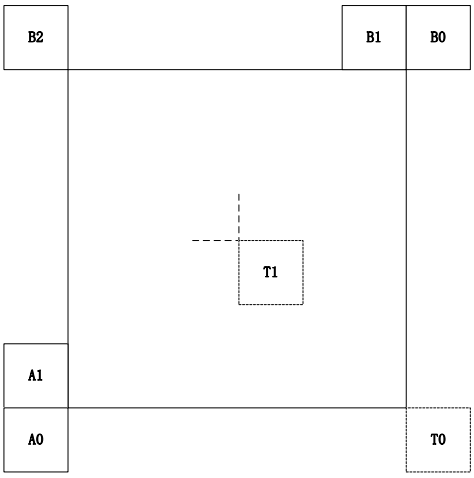

Fig.1. MV candidate set for inter, merge and skip mode

For merge and skip mode, they share the same candidate set with AMVP except for different derivation method. Firstly, for spatial merge candidates, the five neighbor blocks are checked in order A1, B1, B0, A0 and B2, and B2 is considered only when any of the four previous blocks is unavailable or coded in intra mode. Secondly, the temporal merge candidate derivation is the same as AMVP, except that no reference picture index is signaled to decoder. Finally, if the number of merge candidates doesn't reach the maximum number of merge candidates (signaled in the slice header), combined bi-predictive merge candidates using two candidates in different reference list found during the first two steps (only for B slices) and zero MV candidates are added until the condition is met.

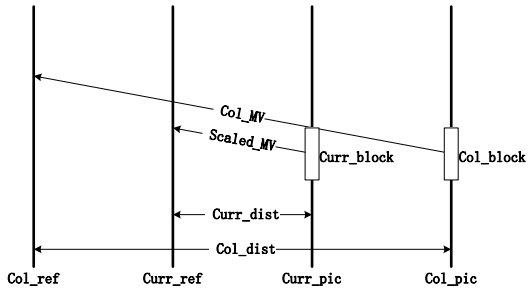

Fig.2. MV scaling for temporal MV candidate

3. PROPOSED PMVR METHOD

In both H.264/AVC and HEVC, it is observed that MV close to the MVP is more likely to be optimal in the rate distortion sense. For pixel positions closer to the MVP, it is worth searching MV with higher accuracy. Otherwise, for pixel positions farther from the MVP, which are less likely to be optimal, lower accuracy is enough for MV search. Therefore, it is proposed to employ different MV resolutions in a progressive manner. That is, using higher MV resolution for MVs near to the MVP and lower MV resolution for MVs far from the MVP. As we know, the MV resolution is always set to 1/4 pixel in HEVC. In principle, PMVR can support any number of resolutions. However, only up to three sub-pixel resolutions (1/8, 1/4, and 1/2 pixel) are supported as a tradeoff between the coding performance and complexity.

3.1. Limited 1/4 pixel and 1/8 pixel positions

First, to support 1/8 pixel resolution, 1/8 pixel luma interpolation filter and 1/16 pixel chroma interpolation filter are derived using the DCT-IF method [7], as shown in Table 1.

Table 1. Proposed interpolation filter
(a) 8-tap 6-bit 1/8 pixel luma interpolation filter

Position	Filter Coefficients
1/8	{-1, 3, -6, 62, 9, -4, 2, -1}
3/8	{-2, 5, -12, 50, 30, -10, 4, -1}

(b) 4-tap 6-bit 1/16 pixel chroma interpolation filter

Position	Filter Coefficients			
1/16	{-2,	63,	4	-1}
3/16	{-5,	59,	13	-3}
5/16	{-6,	52,	23,	-5}
7/16	{-7,	43,	34,	-6}

The 1/4 (1/8) pixel MV positions are disabled when they are outside of the specific 1/4 (1/8) pixel range. Fig. 3 illustrates the MV resolution restriction, in which the red square indicates the 1/4 pixel range and the blue square indicates the 1/8 pixel range.

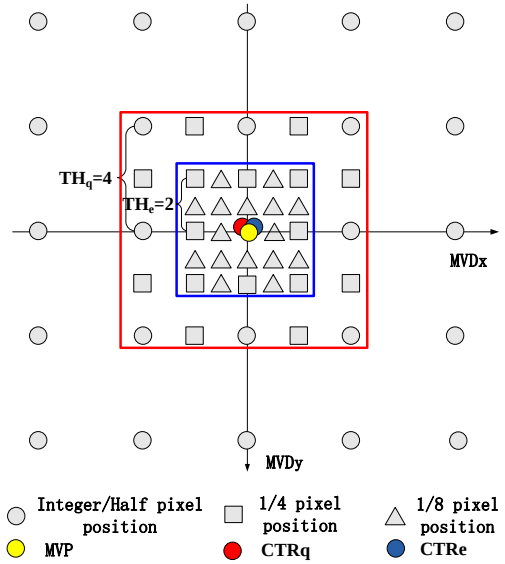

Fig. 3. MV resolution range in PMVR when the MVP is at half pixel position $((TH_q, TH_e) = (4, 2))$

The variables TH_q and TH_e are two thresholds (in units of 1/8 pixel) to define the square range size of 1/4 pixel position and 1/8 pixel position respectively. TH_q and TH_e are restricted to be exact multiples of 2 and 4, respectively. Furthermore, TH_q should not be less than TH_e.

CTRq ($CTRq_x$, $CTRq_y$) and CTRe ($CTRe_x$, $CTRe_y$) are the center of the range of 1/4 pixel and 1/8 pixel positions, and they are derived from the MVP (MVP_x, MVP_y) as follows:

$$CTRe_x = (MVP_x >> 1) << 1 \quad (1)$$
$$CTRe_y = (MVP_y >> 1) << 1 \quad (2)$$
$$CTRq_x = ((MVP_x+1) >> 2) << 2 \quad (3)$$
$$CTRq_y = ((MVP_y+1) >> 2) << 2 \quad (4)$$

It can be seen from (1)-(4) that CTRe is the nearest quarter pixel position to MVP and CTRq is the nearest half pixel position to MVP. If the MVP is just at integer or half pixel positions, then the MVP, CTRq and CTRe coincide as shown in Fig. 3; If the MVP is at 1/4 pixel position, the MVP and CTRe coincide, but the CTRq is converted to integer or half pixel position from the MVP as shown in (3)-(4); If the MVP is at 1/8 pixel position as shown in Fig. 4, both the CTRq and CTRe are converted to integer or half pixel position from the MVP as shown in (1)-(4).

For an extreme case, when $TH_e = 0$, the 1/8 pixel MV resolution is actually disabled except for skip and merge mode. In this case, the MVP of inter mode except skip and merge mode has to be converted to 1/4 pixel accuracy as follows:

$$MVP_x = (MVP_x >> 1) << 1 \quad (5)$$
$$MVP_y = (MVP_y >> 1) << 1 \quad (6)$$

Furthermore, when $TH_e = TH_q = 0$, even the 1/4 pixel MV resolution is disabled. In this case, the MVP of inter mode except skip and merge mode should be converted to half pixel accuracy as follows:

$$MVP_x = (MVP_x >> 2) << 2 \quad (7)$$
$$MVP_y = (MVP_y >> 2) << 2 \quad (8)$$

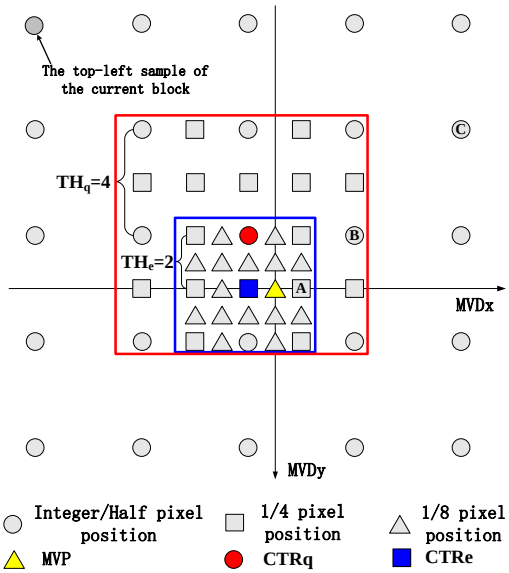

Fig. 4. MV resolution range in PMVR when the MVP is at 1/8 pixel position $((TH_q, TH_e) = (4, 2))$

With the proposed limited 1/4 pixel and 1/8 pixel MV positions, the motion estimation can be simplified. For each 1/4 pixel MV candidate MVC (MVC_x, MVC_y), if $|MVC_x - CTR_x| > TH_q$ or $|MVC_y - CTR_y| > TH_q$, then the MVC is skipped. For each 1/8 pixel MV candidate MVC, if $|MVC_x - CTR_x| > TH_e$ or $|MVC_y - CTR_y| > TH_e$, then the MVC is skipped.

3.2. Efficient MVD representation

With the 1/4 pixel range and 1/8 pixel range described in the previous section, actually there are three cases of MV position:

- Inside the 1/8 pixel range (e.g. A in Fig. 4).
- Inside the 1/4 pixel range and outside the 1/8 pixel range (e.g. B in Fig. 4).

- Outside the 1/4 pixel range (e.g. C in Fig. 4).

For the second and third case, 1/4 pixel accuracy and 1/2 pixel accuracy are enough for MV representation, respectively. However, the original MVD which is directly equal to MV minus MVP, has to be in 1/8 pixel accuracy because the MVP may be in any position. Therefore, we use a piecewise compression method to reduce the MVD value. To be specific, for the first case of MV position, the MV is kept unchanged; for the second case, the part of MV exceeding 1/8 pixel range (limited by TH_e) is reduced to half of its original size; otherwise the part of MV exceeding 1/8 pixel range and 1/4 pixel range (limited by TH_q) are reduced to half and one fourth of its original size, respectively. Then the MVD is equal to the difference between the "new" MV and the MVP as before. The detailed derivation method is provided as Algorithm 1 below. For example, the MVD value (MVD_x, MVD_y) of typical position A, B and C in Fig. 4 before and after conversion are listed in Table 2, with the MVP, MV, CTRe and CTRq as input. By this means fewer bits are used for MVD coding and no extra flag has to be signaled at block level to indicate the MV resolution because the MV position information is kept in MVD with piecewise method. After that, the converted MVD is coded with CABAC in the same way as HEVC.

Table 2. Example of MVD conversion (the top-left sample of the current block is pointed out in Fig. 4)

Position	Input				Ouput	
	MVP	MV	CTRe	CTRq	Original MVD	Proposed MVD
A	(9, 10)	(10, 10)	(8, 10)	(8, 8)	(1, 0)	(1, 0)
B	(9, 10)	(12, 8)	(8, 10)	(8, 8)	(3, -2)	(2, -1)
C	(9, 10)	(16, 4)	(8, 10)	(8, 8)	(7, -6)	(3, -1)

Algorithm 1: MVD derivation at the encoder

Input : MV, MVP, CTRq, CTRe
Ouput : MVD
Begin

- If $|MV_x - CTRq_x| > TH_q$
  - $S = sign(MV_x - CTRq_x)$
  - $MVD_x = CTRe_x + S*TH_e + ((CTRq_x + S*TH_q) - (CTRe_x + S*TH_e)) / 2 + (MV_x - (CTRq_x + S*TH_q)) / 4 - MVP_x$
  - $MVD_y = (MV_y - CTRq_y) / 4$
- Elseif $|MV_y - CTRq_y| > TH_q$
  - Similar to x component above
- Elseif $|MV_x - CTRe_x| > TH_e$
  - $S = sign(MV_x - CTRe_x)$
  - $MVD_x = CTRe_x + S*TH_e + (MV_x - (CTRe_x + S*TH_e)) / 2 - MVP_x$
  - $MVD_y = (MV_y - CTRe_y) / 2$
- Elseif $|MV_y - CTRe_y| > TH_e$
  - Similar to x component above
- Else
  - $MVD_x = MV_x - MVP_x$
  - $MVD_y = MV_y - MVP_y$

End

At the decoder, the MV is derived in reverse from the formulas which are detailed in Algorithm 1, according to its position. The MV position can be judged from the relation between the MVD, MVP, TH_q and TH_e. The derivation method is detailed in Algorithm 2, with the MVD, MVP, CTRq and CTRe as input.

Algorithm 2: MV derivation at the decoder

Input : MVD, MVP, CTRq, CTRe
Ouput : MV
Begin

- $CTRavg_x = (CTRq_x + CTRe_x) / 2$
- $CTRavg_y = (CTRq_y + CTRe_y) / 2$
- $THavg = (TH_q + TH_e) / 2$
- $TMV_x = MVD_x + MVP_x$
- $TMV_y = MVD_y + MVP_y$
- If $|TMV_x - CTRavg_x| > THavg$
  - $S = sign(TMV_x - CTRavg_x)$
  - $MV_x = TMV_x * 4 - (CTRe_x + S*TH_e) * 4 - ((CTRq_x + S*TH_q) - (CTRe_x + S*TH_e)) * 2 + (CTRq_x + S*TH_q)$
  - $MV_y = MVD_y * 4 + CTRq_y$
- Elseif $|TMV_y - CTRavg_y| > THavg$
  - Similar to x component above
- Elseif $|TMV_x - CTRe_x| > TH_e$
  - $S = sign(TMV_x - CTRe_x)$
  - $MV_x = TMV_x * 2 - (CTRe_x + S*TH_e) * 2 + (CTRe_x + S*TH_e)$
  - $MV_y = MVD_y * 2 + CTRe_y$
- Elseif $|TMV_y - CTRe_y| > TH_e$
  - Similar to x component above
- Else
  - $MV_x = MVD_x + MVP_x$
  - $MV_y = MVD_y + MVP_y$

End

3.3. Threshold selection

The values of the threshold TH_q and TH_e have a great effect on the coding performance and complexity. The smaller the threshold, the more MV positions can be skipped during ME, thereby more time saving can be achieved at the encoder and the distortion may get larger since the MV accuracy get lower. Meanwhile, fewer coding bits are used for MVD coding. The opposite result is got with the threshold set larger. Experientially, TH_q = 4 and TH_e = 2 can achieve the best RD performance with almost the same encoding and decoding time. While TH_q = 4 and TH_e = 0 can achieve both encoding time reduction and RD performance improvement.

4. EXPERIMENTAL RESULTS

To verify the performance of the proposed PMVR method, it has been implemented into HEVC reference software HM8.0. Simulations are conducted for test sequences *PeopleOnStreet, Kimono, BasketballDrive, BasketballDrill, BQMall, PartyScene, BQSquare and BlowingBubbles* under common test conditions defined in [8]. Two cases (TH_q, TH_e) = (4, 2) and (TH_q, TH_e) = (4, 0) are tested and the experimental results are shown in the following two tables.

From Table 3, it can be seen that for (TH_q, TH_e) = (4, 2) case, proposed PMVR can provide 1.7%, 1,4% and 3.8% BD-rate gain on average for RA-Main, LB-Main and LP-Main respectively with almost the same time cost as HM8.0 Anchor. Specially, the maximum BD-rate gain is 16% for sequence BQSquare. The reason is that BQsquare is a sequence with a slow-moving shot far away and lots of scenes of water wave, which needs high MV resolution (1/8 or higher) for slowly zooming and moving scene as well as waving water. Therefore, PMVR with 1/8 pixel resolution is particularly useful for full-view sequences with slow scene moving because it can provide both more accurate motion compensation and fewer MVD coding bits than HM8.0 Anchor.

Table 3. Performance of PMVR with (TH_q, TH_e)=(4, 2)

vs HM8.0 Anchor									
	Random Access main			Low delay B main			Low delay P main		
	Y	U	V	Y	U	V	Y	U	V
PeopleOnStreet	0.0%	-1.0%	-0.7%	0.2%	-1.0%	-0.5%	0.6%	-0.4%	-0.2%
Kimono	-0.3%	-0.2%	-0.5%	0.0%	0.3%	-0.3%	0.0%	0.0%	0.0%
BasketballDrive	-0.4%	-1.1%	-1.2%	-0.3%	-1.1%	-1.0%	-0.7%	-1.3%	-1.0%
BasketballDrill	-1.7%	-2.3%	-2.0%	-1.0%	-5.3%	-4.7%	-1.4%	-4.7%	-3.9%
BQMall	-0.9%	-1.1%	-1.0%	-0.7%	-1.1%	-2.1%	-2.3%	-3.0%	-2.7%
PartyScene	-2.8%	-3.1%	-2.8%	-1.9%	-3.1%	-3.4%	-6.4%	-6.4%	-6.5%
BQSquare	-6.2%	-3.5%	-5.0%	-5.8%	-4.5%	-9.7%	**-16.0%**	-12.4%	-17.8%
BlowingBubbles	-1.7%	-2.2%	-2.6%	-1.8%	-3.4%	-2.3%	-4.3%	-4.5%	-5.1%
Average	-1.7%	-1.8%	-2.0%	-1.4%	-2.4%	-3.0%	-3.8%	-4.1%	-4.7%
Enc Time [%]	101%			97%			103%		
Dec Time [%]	100%			101%			101%		

As an extreme case with TH_e = 0, 1/8 pixel positions are actually disabled in motion search, which keeps low complexity for the encoder. However, 1/8 pixel MVs can still be derived by MV scaling in skip or merge mode, and thus coding performance can still be improved in this case. From Table 4, it can be seen that for (TH_q, TH_e) = (4, 0) case, it can provide 1.4%, 1.0% and 1.7% BD-rate gain on average for RA-Main, LB-Main and LP-Main respectively with around 9% encoding time reduction.

To further compare the proposed algorithm with the Anchor, the Rate-Distortion performance curves of some typical test sequences are shown in Fig. 5. It can be intuitively seen that, PMVR achieves better R-D performance than HM8.0.

Table 4. Performance of PMVR with (TH_q, TH_e)=(4, 0)

vs HM8.0 Anchor									
	Random Access main			Low delay B main			Low delay P main		
	Y	U	V	Y	U	V	Y	U	V
PeopleOnStreet	-0.7%	-1.6%	-1.4%	-0.5%	-1.2%	-0.7%	-0.5%	-1.2%	-0.7%
Kimono	-0.8%	-0.5%	-0.5%	-0.5%	-0.2%	-0.5%	-0.1%	-0.6%	0.1%
BasketballDrive	-0.6%	-1.0%	-0.9%	-0.5%	-0.4%	-0.4%	-0.1%	-0.3%	-0.1%
BasketballDrill	-1.2%	-1.5%	-1.2%	-0.8%	-2.1%	-2.4%	-0.2%	-1.7%	-1.3%
BQMall	-0.8%	-0.9%	-0.7%	-0.7%	-0.6%	-0.4%	-1.0%	-1.1%	-0.7%
PartyScene	-1.9%	-2.0%	-1.9%	-1.1%	-0.9%	-0.7%	-3.2%	-2.5%	-2.2%
BQSquare	-3.7%	-2.0%	-2.6%	-2.5%	1.2%	-1.7%	-6.1%	-1.1%	-5.3%
BlowingBubbles	-1.4%	-1.4%	-1.4%	-1.5%	-0.7%	0.1%	-2.7%	-1.2%	-1.0%
Average	-1.4%	-1.4%	-1.3%	-1.0%	-0.6%	-0.8%	-1.7%	-1.2%	-1.4%
Enc Time [%]	94%			88%			91%		
Dec Time [%]	101%			101%			100%		

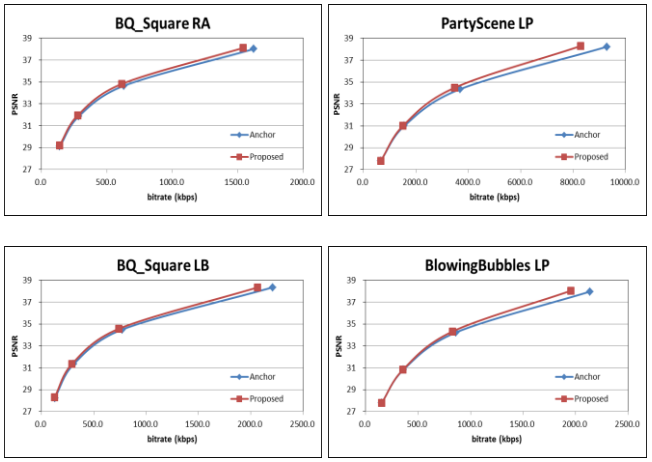

Fig. 5. The rate-distortion curves of PMVR ((TH_q, TH_e)=(4, 2)) and the anchor

5. CONCLUSION

This paper proposes a progressive motion vector resolution adaptation method for HEVC. The novelty of this scheme lies in that we progressively adjust the MV resolution based on the distance between the MV and MVP and restrict the resolution ranges by simple thresholds. Moreover, a piecewise MVD derivation method is applied to code MVD efficiently. Experimental results show that PMVR can

achieve significant BD-rate gain with almost the same or less time cost as HM8.0.

6. ACKNOWLEDGEMENT

This research is supported by the 863 program (2012AA011505) and the National Science Foundation of China (61121002, 61103088), which are gratefully acknowledged.

7. REFERENCES

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 8," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, document JCTVC-J1003, Stockholm, Sweden, July, 2012.

[2] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification ITU-T Rec. H.264/ISO/IEC 14996-10 AVC), Mar. 2003.

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits and Systems for Video Tech., this issue.

[4] P. Helle, S. Oudin, B. Bross, et al, "Block Merging for Quadtree-Based Partitioning in HEVC". IEEE Trans. Circuits and Systems for Video Tech., 22(12): 1720-1731, Dec. 2012.

[5] M. Karczewicz, P. Chen, R. Joshi, X. Wang, W.-J. Chien, and R. Panchal, "Video coding technology proposal by Qualcomm Inc", JCT-VC meeting contribution JCTVC-A121, Dresden, DE, April, 2010.

[6] G. Laroche, J. Jung, B. Pesquet-Popescu, "RD Optimized Coding for Motion Vector Predictor Selection". IEEE Trans. Circuits and Systems for Video Tech., 18(9): 1247-1257, Sept. 2008.

[7] K. McCann, W. -J. Han and I.-K. Kim, "Samsung's Response to the Call for Proposals on Video Compression Technology," JCTVC-A124, Dresden, Germany, April 2010.

[8] F. Bossen, "HM 8 Common Test Conditions and Software Reference Configurations," ITU-T SG16 Contribution, JCTVC-J1100, Stockholm, July. 2012.