|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  7th Meeting: Geneva, CH, 21-30 November, 2011 | Document: JCTVC-G277 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE3: Progressive Motion Vector Resolution** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
|  |  | | |
| *Author(s) or Contact(s):* | \*Jicheng An, \*Xiang Li, \*Xun Guo and  #Shawmin Lei  \*North Building 10F, Raycom Infotech Park Tower C, No. 2 Kexueyuan South Rd., Haidian District, Beijing, China 100190  #No. 1, Dusing Rd. 1, Hsinchu Science Park, Hsinchu, Taiwan 30078 | Email: | {jicheng.an, x.li, xun.guo, shawmin.lei}@mediatek.com | |
| *Source:* | MediaTek Inc. | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

In JCTVC-F125, a progressive MV resolution (PMVR) method was proposed, which uses higher MV resolution near to MV predictor (MVP) and lower MV resolution far from MVP. Thresholds for 1/4 and 1/8 accuracy were used to indicate the range of corresponding MV resolutions. This contribution presents the results of PMVR method on top of HM4.0. For PMVR without 1/8 accuracy, it is reported that by using different thresholds, average BD-Rate reduction of 0.2% can be achieved with around 9% encoding time decrease for RA and LB cases, and average BD-Rate reduction of 0.1% can be achieved with around 4% encoding time decrease for LP case. For PMVR with 1/8 accuracy, it is reported that by using different thresholds, average BD-Rate reduction of 0.5% with around 8% encoding time decrease can be achieved for RA and LB cases, and average BD-Rate reduction of 2.6% with 5% encoding time increase can be achieved for LP case.

# Introduction

In H.264/AVC [1] and current HEVC [2], MV resolution is always set to 1/4 pixel. In JCTVC-F125 [3], a PMVR method was first proposed. It is observed that MV near to MVP is more likely to be rate-distortion (R-D) optimal. Therefore, it is proposed to employ different MV resolutions in a progressive manner. That is, using higher MV resolution when MV is near to MVP and lower MV resolution when MV is far from MVP.

In this contribution, the same idea proposed in JCTVC-F125 is tested based on HM4.0. Moreover, the MV resolution threshold index is transmitted in SPS to indicate the range of the MV resolution. In section 2, the PMVR without 1/8 pixel MV resolution is described, which is actually a simplification of HM4.0. In section 3, the PMVR with 1/8 pixel MV resolution is described. Section 4 shows experimental results, and finally Section 5 gives conclusion.

# PMVR without 1/8 MV resolution

## Algorithm Description

This method is actually a simplification of HM4.0 since it disables 1/4 pixel MV positions when they are out of the range of 1/4 threshold. Fig. 1 illustrates the MV resolution when the threshold is 2.



(a) MVP in integer or half pixel position. (b) MVP in quarter pixel position.

Fig. 1 PMVR without 1/8 pixel resolution with TH=2.

In Fig. 1, the TH is the threshold parameter used to control the range of 1/4 pixel position. 1/4 pixel MV positions are only allowed within a limited range around MVP (within the red square). TH should be larger than or equal to zero and should be an exact multiple of 2. The CTR is the center of 1/4-pixel position, which is derived as follows:

CTRx = MVPx>>1<<1, (1)

CTRy = MVPy>>1<<1. (2)

When MVP is in half or integer pixel positions, MVP and CTR coincide as shown in Fig. 1 (a). When MVP is in quarter pixel position, MVP and CTR don’t coincide as shown in Fig. 1(b).

For an extreme case, when TH is set to zero, 1/2 pixel resolution is always used for inter mode except skip and merge modes. In this case, the MVP of inter mode other than skip and merge modes has to be converted to 1/2 pixel accuracy as follows:

MVPx = MVPx >>1<<1, (3)

MVPy = MVPy >>1<<1. (4)

Since not all the 1/4 pixel MV positions are needed in PMVR, the MVD derivation, which is equal to MV minus MVP in HM4.0, can be modified to a more efficient (or dense) manner as shown in Algorithm 1. After that, MVD is coded with CABAC or CAVLC in the same way as MVD coding in HEVC.

|  |
| --- |
| **Algorithm 1: MVD derivation** |
| **input** **: MV, MVP, TH**  **ouput** **: MVD**  **begin**   * If |MVx-CTRx| > TH   + MVDx = (MVx + CTRx + sign(MVx-CTRx)\*TH) / 2 – MVPx   + MVDy = (Mvy – CTRy) / 2 * Elseif |MVy-CTRy| > TH   + MVDx = (MVx – CTRx) / 2   + MVDy = (MVy + CTRy + sign(MVy-CTRy)\*TH) / 2 – MVPy * Else   + MVDx = MVx – MVPx   + MVDy = MVy – MVPy |
| **end** |

For example, the MV positions A, B, and C in Fig. 1 (a) has MVD values listed in Table 1:

Table 1 MVD values in HM4.0 and in PMVR

|  |  |  |  |
| --- | --- | --- | --- |
| MV positions | A | B | C |
| (MVDx, MVDy) in HM4.0 | (2, 0) | (4, -2) | (4, -4) |
| (MVDx, MVDy) in PMVR | (2, 0) | (3, -1) | (3, -2) |

It can be seen that the MVD coding bits can be reduced by PMVR.

Inversely, the MV derivation according to MVP and MVD at decoder is shown in Algorithm 2 as follows:

|  |
| --- |
| **Algorithm 2: MV derivation** |
| **input** **: MVD, MVP, TH**  **ouput** **: MV**  **begin**   * CTRDx = CTRx - MVPx * CTRDy = CTRy - MVPy * If |MVDx-CTRDx| > TH   + MVx = MVPx + MVDx\*2 - CTRDx - sign(MVDx-CTRDx)\*TH   + MVy = MVPy + MVDy\*2 + CTRDy * Elseif |MVDy-CTRDy| > TH   + MVx = MVPx + MVDx\*2 + CTRDx   + MVy = MVPy + MVDy\*2 - CTRDy - sign(MVDy-CTRDy)\*TH * Else   + MVx = MVDx + MVPx   + MVy = MVDy + MVPy |
| **end** |

* 1. ***Threshold Transmission***

To make the proposed method more flexible, it is proposed to signal the threshold TH in bitstreams. In specific, the index of threshold values is added in SPS. Table 2 shows the mapping between TH index and TH value, in which “0” means pure 1/2 MV resolution and “+∞” means pure 1/4 MV resolution.

Table 2 TH index and value

|  |  |
| --- | --- |
| TH index | TH |
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | +∞ |

# PMVR with 1/8 MV resolution

## Algorithm Description

First, To support 1/8 pixel resolution, 1/8 pixel interpolation filter for luma and 1/16 pixel interpolation filter for chroma derived using DCT-IF method are proposed as shown in Table 2 and Table 3.

Table 3 Proposed 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} |

Table 4 Proposed 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} |

Then, as shown in Fig. 2, the 1/4 pixel positions are only allowed within the red square, and 1/8 pixel positions are only allowed within the blue square. The variables THq and THe are two thresholds to control the ranges of 1/4 pixel position and 1/8 pixel position respectively. THq and THe are restricted to exact multiples of 2 and 4, respectively, and THq should be larger than THe except when THq=THe=0.



(a) MVP in integer or 1/2 pixel position. (b) MVP in 1/8 pixel position.

Fig. 2 PMVR with 1/8 pixel resolution with (THq, THe)=(4, 2).

CTRq and CTRe are the centers of 1/4 pixel and 1/8 pixel positions respectively, which are derived using the following equations:

CTRe = MVPx>>1<<1, (5)

CTRe = MVPy>>1<<1, (6)

CTRq = (MVPx+1)>>2<<2, (7)

CTRq = (MVPy+1)>>2<<2. (8)

When MVP is in half or integer pixel position, the MVP, CTRq, and CTRe coincide as shown in Fig. 2(a). When MVP is in 1/8 pixel position, the MVP, CTRq and CTRe don’t coincide as shown in Fig. 2(b).

When THe=0, 1/4 pixel resolution is always used for inter mode except skip and merge modes. In this case, the MVP of inter mode except skip and merge modes has to be converted to 1/4 pixel accuracy as follows:

MVPx = MVPx >>1<<1, (9)

MVPy = MVPy >>1<<1. (10)

When THq=0, THe also must be equal to 0, 1/2 pixel MV resolution is always used for inter mode except skip and merge mode. So the MVP of inter mode except skip and merge mode has to be converted to 1/2 pixel position as:

MVPx = (MVPx+1) >>2<<2 (11)

MVPy = (MVPy+1) >>2<<2 (12)

Since not all the 1/4 pixel and 1/8 pixel MV positions are needed in PMVR. The MVD derivation, which is equal to MV minus MVP in HM4.0, can be modified to a more efficient (or dense) manner as shown in Algorithm 3. After that, MVD is coded with CABAC or CAVLC in the same way as MVD coding in HEVC.

|  |
| --- |
| **Algorithm 3: MVD derivation** |
| **input** **: MV, MVP, THq, THe**  **ouput** **: MVD**  **begin**   * If |MVx-CTRqx| >THq   + S = sign(MVx-CTRqx)   + MVDx = (MVx + 2(CTRex+S\*THe) + CTRqx + S\*THq)/4 - MVPx   + MVDy = (MVy – CTRqy) / 4 * Elseif |MVy-CTRqy| >THq   + S = sign(Mvy-CTRqy)   + MVDx = (MVx – CTRqx) / 4   + MVDy = (MVy + 2(CTRey+S\*THe) + CTRqy + S\*THq)/4 – MVPy * Elseif |MVx-CTRex| > THe   + MVDx = (MVx + CTRex + sign(MVx-CTRex)\*THe)/2 – MVPx   + MVDy = (MVy – CTRey)/2 * Elseif |MVy-CTRey| > THe   + MVDx = (MVx – CTRex)/2   + MVDy = (MVy + CTRey + sign(MVy-CTRey)\*THe)/2 – MVPy * Else   + MVDx = MVx – MVPx   + MVDy = MVy – MVPy |
| **End** |

For example, the MV positions A, B, and C in Fig. 2 (b) has MVD values listed in Table 4.

Table 5 MVD values in uniform 1/8 pixel resolution and in PMVR

|  |  |  |  |
| --- | --- | --- | --- |
| MV positions | A | B | C |
| (MVDx, MVDy) in uniform 1/8 pixel resolution | (1, 0) | (3, -2) | (7, -6) |
| (MVDx, MVDy) in PMVR | (1, 0) | (2, -1) | (3, -1) |

It can be seen that the MVD coding bits can be reduced by PMVR.

Inversely, the MV derivation according to MVP and MVD in decoder is shown in Algorithm 4 as follows:

|  |
| --- |
| **Algorithm 4: MV derivation** |
| **input** **: MVD, MVP, THq, THe**  **ouput** **: MV**  **begin**   * CTRDex = CTRex - MVPx * CTRDey = CTRey – MVPy * CTRDqx = CTRqx - MVPx * CTRDqy = CTRqy - MVPy * CTRDavgx = (CTRDex + CTRDqx)>>1 * CTRDavgy = (CTRDey + CTRDqy)>>1 * THavg = (THe + THq)>>1 * If |MVDx-CTRDavgx| > THavg   + S = sign(MVDx-CTRDavgx)   + MVx = MVPx + MVDx\*4 – (CTRDex + S\*THe)\*2 – CTRDqx – S\*THq   + MVy = MVPy + MVDy\*4 + CTRDqy * Elseif |MVDy-CTRDavgy| > THavg   + S = sign(MVDy-CTRDavgy)   + MVx = MVPx + MVDx\*4 + CTRDqx   + MVy = MVPy + MVDy\*4 – (CTRDey + S\*THe)\*2 – CTRDqy – S\*THq * Elseif |MVDx-CTRex| > THe   + MVx = MVPx + MVDx\*2 – CTRDex – sign(MVDx-CTRex)\*THe   + MVy = MVPy + MVDy\*2 + CTRDey * Elseif |MVDy-CTRey| > THe   + MVx = MVPx + MVDx\*2 + CTRDex   + MVy = MVPy + MVDy\*2 – CTRDey – sign(MVDy-CTRey)\*THe * Else   + MVx = MVDx + MVPx   + MVy = MVDy + MVPy |
| **End** |
|  |

* 1. ***Threshold Transmission***

Similar to Section 2.2, it is also proposed to signal the index of threshold pairs in bitstreams. Table 6 shows the mapping between TH pair index and threshold values, in which (+∞, +∞) means pure 1/8 MV resolution, and (0, 0) means pure 1/2 MV resolution.

Table 6 TH pair index and value

|  |  |
| --- | --- |
| TH pair index | (THq, THe) |
| 0 | (0, 0) |
| 1 | (4, 0) |
| 2 | (4, 2) |
| 3 | (+∞, +∞) |

# 

# Experimental Results

To evaluate the performance of PMVR, simulations are conducted under conditions defined in JCTVC-F900 [4]. For the PMVR without 1/8 pixel presented in Section 2, the TH=2 and TH=4 are tested. The results are shown in Table 7 and Table 8 respectively.

It can be seen from Table 7 that the TH=2 case can provide around 0.2% BD-rate gain with 9~10% encoding time reduction for RA and LB configurations, and less than 0.1% BD-rate loss with 6~7% encoding time reduction for LP configurations. From Table 8, the TH=4 case can provide less than 0.2% BD-rate gain with 6% encoding time reduction for RA and LB, and less than 0.1% BD-rate gain with 4% encoding time reduction for LP. In all, the TH=2 case has more BD-rate gain and complexity reduction for B slices, and TH=4 case has better RD performance for P slices. Therefore, it is proposed to set TH=2 for B slice and TH=4 for P slice as default.

For the PMVR with 1/8-pixel presented in Section 3, the threshold pair (THq, THe) equal to (4, 2) and (4, 0) are tested, and the results are shown in Table 9 and Table 10 respectively. It can be seen from Table 9 and Table 10 that the (4, 0) case can provide around 0.3%~0.7% BD-rate gain with 8% encoding time reduction for RA and LB configurations, and (4, 2) case can provide around 2.6% BD-rate gain with 7% encoding time increase for LP configurations. Therefore, it is proposed to set (THq, THe)=(4, 0) for B slices and (4, 2) for P slices as default.

Table 7 Performance of PMVR without 1/8 pixel with TH=2



Table 8 performance of PMVR without 1/8 pixel with TH=4



Table 9 Performance of PMVR with 1/8 pixel with (THq, THe)=(4, 2)



Table 10. Performance of PMVR with 1/8 pixel with (THq, THe)=(4, 0)



# Conclusion

In this proposal, two PMVR variants (with 1/8-pixel and without 1/8-pixel) are tested based on HM4.0.

The PMVR without 1/8 pixel case is a simplification of HM4.0, which disables the ¼ pixel MV positions that are far from MVP by a threshold parameter TH. Consequently, the complexity of motion estimation and interpolation can be reduced and BD-rate gain can also be achieved (For RA and LB, around 9% encoding time reduction and 0.2% BD-rate gain. For LP, around 4% encoding time reduction and 0.05% BD-rate gain). Based on the experimental results, it is suggested to set TH=2 for B slice and TH=4 for P slice as default and send the TH index in SPS to enable more flexibility.

The PMVR with 1/8-pixel case disables the 1/4-pixel MV positions which are far away from MVP by a threshold parameter THq and enables the 1/8-pixel MV positions which are close to MVP by a threshold parameter THe. For RA and LB configurations, around 0.3%~0.7% BD-rate gain with 8% encoding time reduction can be achieved by setting the (THq, THe)=(4, 0), and for LP configurations, around 2.6% BD-rate gain with 7% encoding time increase can be achieved by setting the (THq, THe)=(4, 2). Therefore, it is suggested to set the (THq, THe)=(4, 0) for B slices and (4, 2) for P slices as default and send the TH pair index in SPS to enable more flexibility.

# References

1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification ITU-T Rec. H.264/ISO/IEC 14496-10 AVC), Mar. 2003.
2. T. Wiegand, W.J. Han, B. Bross, J.-R. Ohm, and G. J. Sullivan, “WD4: Working Draft 4 of High-Efficiency Video Coding”, JCTVC-F803, Torino, IT, July, 2011.
3. J. An, X. Li, X. Guo, and S. Lei, “Progressive Motion Vector Resolution”, JCTVC-F125, Torino, IT, July, 2011.
4. F. Bossen, “Common test conditions and software reference configurations”, JCTVC-F900, Torino, IT, July, 2011.

# Patent rights declaration(s)

**MediaTek Inc. may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**