|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extension Development**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  2nd Meeting: Shanghai, SE, 13–19 Oct 2012 | Document: JCT2-B0089 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Improvement on MV candidates for 3DVC** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Kai Zhang1, Jicheng An1, Jian-Liang Lin2, Yu-Lin Chang2, Yu-Wen Huang2, Shawmin Lei2  1 MediaTek Beijing North Building 10F, Raycom Infotech Park Tower C, No. 2 Kexueyuan South Rd., Haidian District, Beijing, China 100190  2 MediaTek Hsinchu No. 1, Dusing Rd. 1, Hsinchu Science Park, Hsinchu, Taiwan 30078 | Tel:  Email: | Shawmin Lei +886-3-5670766 ext. 25555 {kai.zhang, jicheng.an, jl.lin, yulin.chang, yuwen.huang, shawmin.lei}@mediatek.com |
| *Source:* | MediaTek Inc. | | |

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

# Abstract

In the current HTM, the inter-view candidate is introduced into the MV candidate list for AMVP and merge blocks. To get the inter-view candidate, an estimated DV is obtained first. If no estimated DV can be found, a default zero-DV is used. Then an inter-view motion predictor or a disparity vector predictor is given as the inter-view candidate, based on the estimated DV and the current reference picture. In this contribution, we propose three simple modifications to further improve MV candidates in 3DVC. First, the default zero-DV is forbidden in the DV estimation process. The inter-view candidate will not be used if the estimated DV is unavailable. Second, AMVP candidate filling process will be simplified by eliminating redundancy neighboring checking for DVs, Finally merge candidate list will be more reasonable by moving back the inter-view candidate if it turns out to be a disparity vector predictor instead of an inter-view motion predictor. Experimental results show that, the proposed solution achieves about 0.5% BD-rate gain on video 1/video 2 and 0.2% BD-rate gain on the coded & synthesized view respectively, without increasing the computational complexity.

# Introduction

3D video coding is developed for video and depth data of multiple views simultaneously captured by several cameras. Since cameras capture the same scene from different viewpoints, multi-view video data contains a large amount of inter-view redundancy. To utilize the inter-view redundancy, inter-view reference and inter-view motion prediction are introduced on the dependent views in the current HTM [1].

As depicted in Figure 1, inter-view reference picture is inserted into the reference lists as a common temporal reference picture. Since there is no identification for a reference picture to show whether it is an inter-view reference or not, the Disparity-Compensated Prediction (DCP) is realized just as an alternative to motion-compensated prediction (MCP), by choosing the appropriate reference index.

In the MCP case, inter-view motion prediction is a powerful tool as shown in Figure 2. First, an estimated Disparity Vector (DV) for the current block is derived. Second, the reference block in the base-view picture is located by adding the position of the current block and the DV. Third, an existence condition is checked whether the reference block is inter-coded and its Motion Vector (MV) refers to a reference whose POC can be covered by the reference lists of the current block. If the existence condition turns out to be true, the MV of the reference block will be provided as the inter-view motion predictor for the current block. Otherwise, the estimated DV itself (with vertical component set to zero) can be regarded as a ‘Motion Vector Predictor (MVP)’, which is a DV Predictor (DVP) actually.

As we can see, the estimation of DV plays a critical role in the process of inter-view motion prediction. In the current HTM, DV is derived by checking whether spatial or temporal neighboring blocks possess a DV, which will be used as the estimated DV for the current block [2]. As a complement, DV-MCP [3] is adopted to provide an estimated DV when none of the neighboring blocks has a DV. A default zero DV will be used if DV-MCP also fails to find an estimated DV.

In general, there are two types of inter-blocks: merge and non-merge. In non-merge inter-blocks, Adaptive Motion Vector Prediction (AMVP) is utilized. An MV candidates list is constructed in two different ways to provide possible MVPs for merge or AMVP blocks. In 3DVC, the inter-view candidate is introduced into this list. The inter-view candidate can be an inter-view motion predictor or a DVP. It depends on the aforementioned existence condition in merge blocks and the target reference in AMVP blocks. The inter-view candidate takes the first candidate position (position 0) in merge blocks and the third candidate position (position 2) in AMVP blocks. Because the inter-view reference and the temporal reference are treated in an indiscriminate way, there are some problems both in AMVP and merge blocks.

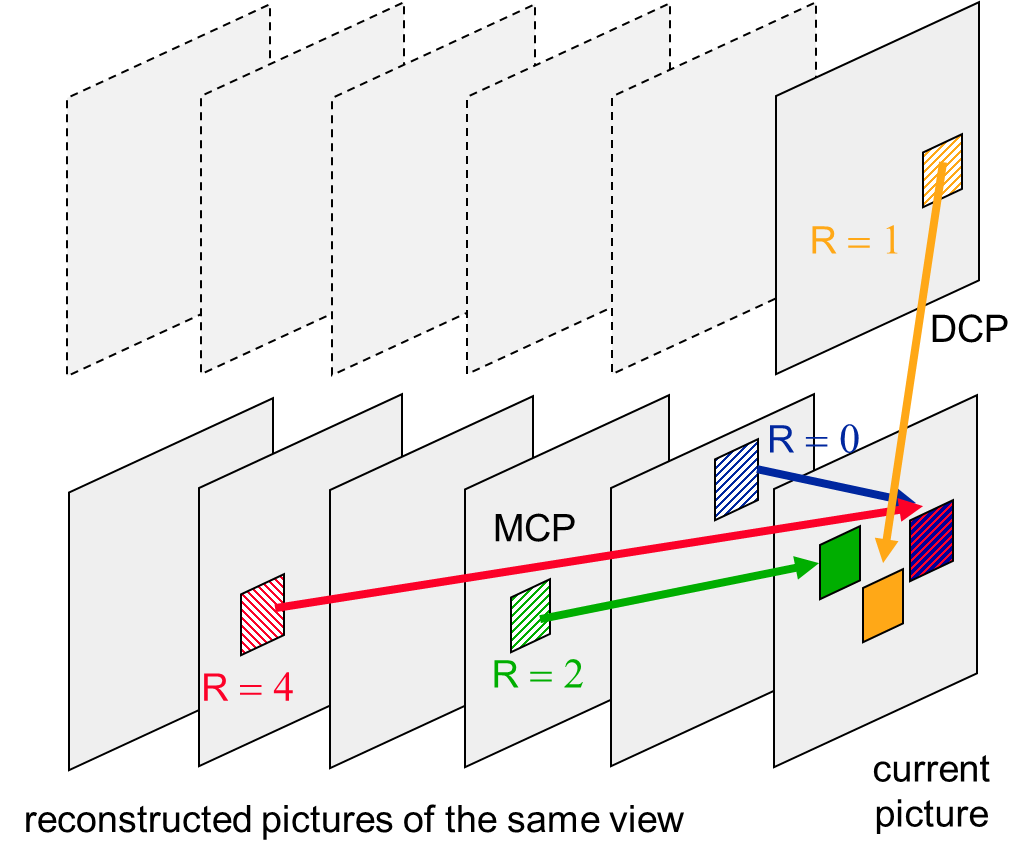


Figure 1. Disparity-compensated prediction as an alternative to motion-compensated prediction.



Figure 2 Inter-view motion prediction.

In AMVP blocks, the target reference picture is specified explicitly. The AMVP procedure fills up the candidate list. First, the DV estimation process is invoked to find an estimated DV. Second, the spatial neighboring blocks are checked to determine whether they possess MVs that refer to the target reference picture, or can be scaled to it. Third, the inter-view candidate is checked to identify if it refers to the target reference picture. Finally, temporal neighboring blocks are checked. shows a simplified flowchart of the AMVP candidate filling process. The POC scaling checking is omitted in the following discussion.



Figure 3 AMVP candidate filling flowchart.



Figure 4 DV estimation flowchart.

Figure 4 depicts the DV estimation process [2][3]. Neighboring blocks are checked one by one to determine whether they possess DVs, *i.e.*, ‘MV’s referring to the inter-view reference. The first found neighboring DV is returned as the estimated DV of the current block. DV-MCP can provide a backup estimated DV in some cases when none of the neighboring blocks has a DV.

If the target reference of AMVP is just an inter-view reference, there might be redundancy between the neighboring checking in the DV estimation process and that in the AMVP candidate filling process. They both try to find if there is a ‘MV’ referring to the inter-view reference in the neighboring blocks. In the worst case, while none of the neighboring blocks has a DV, all the neighboring blocks will be checked for the second time in different order, as depicted in Figure 5. Moreover, the inter-view candidate becomes a DVP instead of a candidate of the inter-view motion prediction while the target reference is just the inter-view reference. Because this candidate also comes from neighboring DVs, the main difference to other candidates only lies in its zero vertical component. It may not be so efficient while using this candidate.

In Merge blocks, inter-view candidate holds the first candidate position. As we described before, inter-view candidate can be used in the inter-view motion prediction or used as a DVP, depending on the existence condition. In the former case, the inter-view candidate refers to a temporal reference, and in the latter case, it refers to an inter-view reference. It may be more efficient to put this candidate at different positions in these two different cases.



Figure Flowchart of AMVP candidate filling in the worst case.

# Proposed methods

## Modification in DV estimation process

In the DV estimation procedure, we simply forbid the use of the default zero-DV when the DV estimation process fails to find an estimated DV. The inter-view candidate will not be used if the estimated DV is unavailable. Unlike MV, zero is not a good default value for DV, thus the inter-view candidate is inefficient when the estimated DV is unavailable. By forbidding the use of the default zero-DV, we can make MV candidates more efficient.

## Modification in AMVP blocks

In AMVP blocks, we simply utilize the DV estimation process to fill up the AMVP candidates list if the target reference is the inter-view reference as shown in Figure 6. The DV estimation process in this case will not return the first found neighboring DV as before but fill the found neighboring DVs into the AMVP candidates list. There are three benefits of the proposed modification. First, the neighboring checking redundancy is avoided. Second, the inefficient DV candidate with zero vertical component is removed. Third, DV-MCP can be used as an AMVP candidate which may give more coding gain. For example, if only the above-left neighboring block has a DV *d*, then there will be only two non-zero candidates: *d* and (*dx*, 0) in the current HTM. In the proposed method, (*dx*, 0) will be removed due to its redundancy, and one or more DV-MCP candidates will be added into the candidates list. Generally speaking, the proposed method eliminates useless candidate checking while adds useful candidates.

## 

Figure Proposed AMVP candidate filling process if the target reference is the inter-view reference. It should be noted that, the ‘Check DV-MCP’ block means that one or more DV-MCP candidates will be used to fill AMVP candidate lists.

## Modification in merge blocks

In merge blocks, we also make a small modification according to the reference type just like in AMVP blocks. If the inter-view candidate refers to an inter-view reference, it is a DVP actually. It is not an advantageous candidate so that it should not take the first candidate position. In our solution, it is placed at the 4th position. In other case, while the inter-view candidate refers to a temporal reference, the inter-view motion prediction is carried out. It is still at the first candidate position.

# Experimental results

The proposed methods are integrated on HTM-4.0.1. Four tests are conducted under common test condition [5]: 1) DV estimation modification only, 2) AMVP modification only, 3) merge modification only 4) the combination of three modifications. The results are listed in Table 1-Table 4 respectively. It can be seen that there are coding gains about -0.1% on video 1/video 2 when only DV estimation is modified. In addition, we can get a gain about -0.2% on video 1/video 2 when only AMVP or merge is modified. As a whole, the gain grows to -0.5% on video 1/video 2 and -0.2% on coded and synthesized views respectively. Moreover, the encoding/decoding time has no obvious change in all these tests.

Table 1 Performance of test 1 based on HTM4.0.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **video 0** | **video 1** | **video 2** | **video only** | **synthesized only** | **coded & synthesized** | **enc time** | **dec time** | **ren time** |
| Balloons | 0.0% | 0.0% | -0.2% | 0.0% | 0.0% | 0.0% | 91.2% | 97.0% | 95.4% |
| Kendo | 0.0% | -0.2% | 0.0% | 0.0% | -0.1% | -0.1% | 92.5% | 101.6% | 89.7% |
| Newspapercc | 0.0% | -0.2% | -0.3% | -0.1% | -0.1% | -0.1% | 97.7% | 97.7% | 100.7% |
| GhostTownFly | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 95.9% | 82.4% | 83.0% |
| PoznanHall2 | 0.0% | 0.1% | 0.2% | 0.0% | 0.1% | 0.1% | 106.3% | 102.4% | 107.2% |
| PoznanStreet | 0.0% | -0.2% | -0.1% | -0.1% | 0.0% | 0.0% | 96.9% | 101.7% | 99.6% |
| UndoDancer | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 92.1% | 96.9% | 88.4% |
| 1024x768 | 0.0% | -0.1% | -0.2% | -0.1% | -0.1% | -0.1% | 93.8% | 98.7% | 95.2% |
| 1920x1088 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 97.6% | 95.5% | 94.1% |
| average | **0.0%** | **-0.1%** | **-0.1%** | **0.0%** | **0.0%** | **0.0%** | **96.0%** | **96.9%** | **94.6%** |

Table 2 Performance of test 2 based on HTM4.0.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **video 0** | **video 1** | **video 2** | **video only** | **synthesized only** | **coded & synthesized** | **enc time** | **dec time** | **ren time** |
| Balloons | 0.0% | -0.4% | -0.2% | -0.1% | -0.1% | -0.1% | 87.1% | 96.1% | 92.0% |
| Kendo | 0.0% | -0.3% | -0.1% | -0.1% | 0.0% | -0.1% | 96.9% | 101.6% | 93.9% |
| Newspapercc | 0.0% | -0.1% | -0.2% | -0.1% | -0.1% | -0.1% | 98.3% | 96.3% | 105.2% |
| GhostTownFly | 0.0% | -0.1% | 0.1% | 0.0% | 0.0% | 0.0% | 92.5% | 77.9% | 81.4% |
| PoznanHall2 | 0.0% | 0.3% | 0.2% | 0.1% | 0.1% | 0.1% | 98.3% | 97.1% | 101.0% |
| PoznanStreet | 0.0% | -0.6% | -0.4% | -0.2% | -0.2% | -0.2% | 97.7% | 94.1% | 100.1% |
| UndoDancer | 0.0% | -0.3% | -0.4% | -0.1% | -0.2% | -0.2% | 90.7% | 92.2% | 87.0% |
| 1024x768 | 0.0% | -0.2% | -0.2% | -0.1% | -0.1% | -0.1% | 93.9% | 98.0% | 96.9% |
| 1920x1088 | 0.0% | -0.2% | -0.1% | -0.1% | -0.1% | -0.1% | 94.7% | 90.0% | 92.0% |
| average | **0.0%** | **-0.2%** | **-0.2%** | **-0.1%** | **-0.1%** | **-0.1%** | **94.4%** | **93.3%** | **94.1%** |

Table 3 Performance of test 3 based on HTM4.0.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **video 0** | **video 1** | **video 2** | **video only** | **synthesized only** | **coded & synthesized** | **enc time** | **dec time** | **ren time** |
| Balloons | 0.0% | -0.2% | -0.2% | -0.1% | -0.1% | -0.1% | 97.5% | 99.2% | 100.7% |
| Kendo | 0.0% | 0.0% | 0.3% | 0.0% | 0.0% | 0.0% | 101.1% | 100.1% | 100.1% |
| Newspapercc | 0.0% | -0.5% | -0.5% | -0.2% | -0.2% | -0.2% | 103.7% | 98.5% | 100.2% |
| GhostTownFly | 0.0% | 0.1% | -0.1% | 0.0% | 0.0% | 0.0% | 100.0% | 80.7% | 81.0% |
| PoznanHall2 | 0.0% | 0.0% | -0.2% | -0.1% | -0.1% | -0.1% | 94.8% | 96.0% | 117.8% |
| PoznanStreet | 0.0% | -0.9% | -0.9% | -0.3% | -0.3% | -0.3% | 96.5% | 95.6% | 94.6% |
| UndoDancer | 0.0% | -0.1% | 0.1% | 0.0% | 0.0% | 0.0% | 104.4% | 94.6% | 96.7% |
| 1024x768 | 0.0% | -0.2% | -0.1% | -0.1% | -0.1% | -0.1% | 100.7% | 99.3% | 100.4% |
| 1920x1088 | 0.0% | -0.2% | -0.3% | -0.1% | -0.1% | -0.1% | 98.8% | 91.5% | 96.7% |
| average | **0.0%** | **-0.2%** | **-0.2%** | **-0.1%** | **-0.1%** | **-0.1%** | **99.7%** | **94.7%** | **98.2%** |

Table 4 Performance of test 4 based on HTM4.0.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **video 0** | **video 1** | **video 2** | **video only** | **synthesized only** | **coded & synthesized** | **enc time** | **dec time** | **ren time** |
| Balloons | 0.0% | -0.4% | -0.4% | -0.2% | -0.2% | -0.2% | 92.6% | 93.2% | 96.5% |
| Kendo | 0.0% | -0.2% | 0.0% | -0.1% | 0.0% | 0.0% | 97.8% | 97.5% | 92.4% |
| Newspapercc | 0.0% | -0.7% | -0.7% | -0.3% | -0.3% | -0.3% | 107.2% | 96.7% | 105.5% |
| GhostTownFly | 0.0% | -0.3% | -0.2% | -0.1% | -0.1% | -0.1% | 94.1% | 83.4% | 81.7% |
| PoznanHall2 | 0.0% | -0.3% | -0.4% | -0.2% | -0.3% | -0.3% | 98.4% | 97.8% | 100.2% |
| PoznanStreet | 0.0% | -1.1% | -1.2% | -0.4% | -0.4% | -0.4% | 99.4% | 98.1% | 99.9% |
| UndoDancer | 0.0% | -0.4% | -0.5% | -0.2% | -0.2% | -0.2% | 91.9% | 87.7% | 89.8% |
| 1024x768 | 0.0% | -0.5% | -0.4% | -0.2% | -0.2% | -0.2% | 99.0% | 95.8% | 98.0% |
| 1920x1088 | 0.0% | -0.5% | -0.6% | -0.2% | -0.2% | -0.2% | 95.9% | 91.5% | 92.6% |
| average | **0.0%** | **-0.5%** | **-0.5%** | **-0.2%** | **-0.2%** | **-0.2%** | **97.2%** | **93.3%** | **94.9%** |

# Working draft modifications

## Modification in DV estimation process

*In subclause* *G8.5.2.1.11,* *the original text is,*

The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.

The flag availableFlagLXInterView is set equal to 1 and the motion vector candidate mvLXInterView is derived by

…

*Shoud be changed to,*

The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.

If availableDV equals to 0, availableFlagLXInterView is set equal to 0.

Otherwise, the flag availableFlagLXInterView is set equal to 1 and the motion vector candidate mvLXInterView is derived by

…

*In subclause G8.5.2.1.11, the original text is,*

The derivation process as specified in subclause is invoked with the luma location ( xP, yP ), and the variables nPSW, nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.

The reference layer luma location ( xRef, yRef ) is derived by

…

*Should be changed to*

The derivation process as specified in subclause is invoked with the luma location ( xP, yP ), and the variables nPSW, nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.

If availableDV equals to 0, availableFlagLXInterView is set equal to 0, and this process is terminated.

Otherwise, the reference layer luma location ( xRef, yRef ) is derived by

…

*In subclause G8.5.2.1.15, the original text is,*

The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), and the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp and as the outputs.

The reference layer luma location ( xRef, yRef ) is derived by

…

*Should be changed to*

The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), and the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp and as the outputs.

If availableDV equals to 0, availableFlagLXInterView is set equal to 0, and this process is terminated.

Otherwise, the reference layer luma location ( xRef, yRef ) is derived by

…

*In subclause G8.5.2.1.17**, the original text is,*

1. The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.
2. The motion vector mvLXInterView and the reference index RefPicListLX are derived by

…

*Should be changed to*

1. The derivation process for a disparity vector as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, as the inputs and a flag availableDV and a disparity vector mvDisp as the outputs.
2. If availableDV equals to 0, availableFlagLXInterView is set equal to 0, and this process is terminated. Otherwise, the motion vector mvLXInterView and the reference index RefPicListLX are derived by

…

## Modification in AMVP blocks

*In subclause* *G8.5.2.1.5, the original text is,*

1. Depending on multi\_view\_mv\_pred\_flag, the following applies:

– If multi\_view\_mv\_pred\_flag is equal to 0, the variable availableFlagLXInterView is set equal to 0.

– Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view motion vector predictor candidate as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPbW and nPbH, the reference index refIdxLX and the reference view identifier refViewIdx as the inputs and the outputs are the flag availableFlagLXInterView and the motion vector candidate mvLXInterView.

…

*Should be changed to*

1. Depending on multi\_view\_mv\_pred\_flag, the following applies:

* If multi\_view\_mv\_pred\_flag is equal to 0, the variable availableFlagLXInterView is set equal to 0.
* Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), if the ViewIdx of RefPicListX [refIdxLX] equals to refViewIdx, the derivation process for disparity vector prediction as specified in subclause G.8.5.2.1.18 is invoked with with the luma location ( xP, yP ), the variables nPbW and nPbH, the reference index refIdxLX and the reference view identifier refViewIdx as the inputs and the outputs are motion vector predictor candidate list, mvpListLX (with X being equal to 0 or 1) and the number of MVP candidates, numMVPCandLX (with X being equal to 0 or 1). Then the process goes to step 7. Otherwise (the ViewIdx of RefPicListX [refIdxLX] does not equal to refViewIdx), the derivation process for the inter-view motion vector predictor candidate as specified in subclause is invoked with the luma location ( xP, yP ), the variables nPbW and nPbH, the reference index refIdxLX and the reference view identifier refViewIdx as the inputs and the outputs are the flag availableFlagLXInterView and the motion vector candidate mvLXInterView.

…

*A new subclause G8.5.2.1.18 is appended as,*

G8.5.2.1.18 Derivation process for a disparity vector prediction

Inputs to this process are:

* a luma location ( xP, yP ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit
* a view identifier refViewIdx specifying a reference view.

Outputs of this process are:

– a disparity vector predictor candidate list, dvpList,

– the number of DVP candidates, numDVPCand.

For N being A0, A1, B0, B1, or B2, and X being 0 or 1, flags availableFlagIvpMvNX and the components of disparity vectors ivpMvDispNX are set equal to 0. availableDV is set equal to 0.

The motion vector mvDisp and the availability flag availableDV are derived in the following ordered steps and the whole decoding process of this sub-clause terminates once availableDV is set to 1.

1. The variable maxNumMVPCand is set equal to 3 and numDVPCand is set to 0.
2. For each N being A1, B1, A0, B0, or B2, and ( xN, yN ) being ( xP − 1,  yP + nPSH − 1 ), ( xP + nPSW − 1,  yP − 1 ), ( xP + nPSW,  yP − 1 ), ( xP − 1,  yP + nPSH  ), or ( xP − 1,  yP − 1 ), the following applies.
3. When yP−1 is less than (( yC >> Log2CtbSizeY ) << Log2CtbSizeY), the following applies.

xB0 = ((xB0>>3)<<3) + ((xB0>>3)&1)\*7 (G‑91)  
xB1 = ((xB1>>3)<<3) + ((xB1>>3)&1)\*7 (G‑92)  
xB2 = ((xB2>>3)<<3) + ((xB2>>3)&1)\*7 (G‑93)

1. For each X from 0 to 1, the following applies, if availableN is equal to 1 and PredFlagLX[ xN ][ yN ] is equal to 1

[Ed.(GT): Derivation of availableN is missing.]

* + If RefPicListX[ RefIdxLX[ xN ][ yN ] ] is an inter-view reference picture with ViewOrderIndex equal to refViewIdx, dvpList[ numDVPCand ] is set equal to mvLXN[ xN ][ yN ] and numDVPCand  is incremented by 1.
  + Otherwise, if PredMode[ xN ][ yN ] is equal to MODE\_SKIP and IvpMvFlagLX[ xN, yN ] is equal to 1, the following applies:

ivpMvDispNX = IvpMvDispLX[ xN, yN ] (G‑96)  
availableFlagIvpMvNX = 1 (G‑97)

1. The derivation process for a disparity vector from temporal neighbour block as specified in is invoked with a luma location ( xP, yP ), variables nPSW and nPSH as inputs, and flag availableDV, disparity vector mvDisp, availableFlagIvpMvColX, and ivpMvDispColX as outputs. If availableDV equals to 1, dvpList[ numDVPCand ] is set equal to mvDisp and numDVPCand  is incremented by 1.
2. For each N being A1, B1, A0, B0, B2, and Col, the following applies.
   * For each X from 0 to 1, the following applies.
   * If availableFlagIvpMvNX is equal to 1, dvpList[ numDVPCand ] is set equal to ivpMvDispNX and numDVPCand  is incremented by 1.

## Modification in merge blocks

*In subclause G8.5.2.1.1,* *the original text is,*

1. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps:.
2. The variable numMergeCand is set equal to 0.
3. When availableFlagInterView is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to InterView and the variable numMergeCand is increased by 1.
4. When availableFlagA1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.
5. When availableFlagB1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.
6. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
7. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
8. When availableFlagB2 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
9. When availableFlagCol is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.

*Should be changed to,*

1. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps:.
2. The variable numMergeCand is set equal to 0.
3. When availableFlagInterView is equal to 1, and none of RefPicListX [refIdxLXInterView] is an inter-view reference picture (with X being equal to 0 or 1), the entry mergeCandList [ numMergeCand ] is set equal to InterView and the variable numMergeCand is increased by 1.
4. When availableFlagA1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.
5. When availableFlagB1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.
6. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
7. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
8. When availableFlagInterView is equal to 1, and RefPicList0[refIdxL0InterView] or RefPicList1[refIdxL1InterView] is an inter-view reference picture, the entry mergeCandList [ numMergeCand ] is set equal to InterView and the variable numMergeCand is increased by 1.
9. When availableFlagB2 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
10. When availableFlagCol is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.

# Conclusion

In this proposal, three simple modifications on the construction of MV candidate list in AMVP and merge blocks are proposed. The proposed method can make the candidate filling process more reasonable and efficient by distinguishing the inter-view reference. On the one hand, redundancy neighboring checking can be avoided; On the other hand, some coding gain is achieved without obvious increase of computational complexity. Experimental results demonstrate that coding gains about -0.5% on video 1/video 2 and -0.2% on the coded & synthesized view can be achieved respectively.

# 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).**

# References

1. H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Muller, H. Rhee, G. Tech, M. Winken, T. Wiegand, “Description of 3D video technology proposal by Fraunhofer HHI (HEVC compatible; configuration B),” Document of ISO/IEC JTC 1/SC 29/WG 11, m22571, November, 2011.
2. L. Zhang, Y. Chen, M. Karczewicz, “3D-CE5.h related: Disparity vector derivation for multiview video and 3DV,” Document of ISO/IEC JTC 1/SC 29/WG 11, m24937, April, 2012.
3. J. Sung, M. Koo, S. Yea, “3D-CE5.h: Simplification of disparity vector derivation for HEVC-based 3D video coding,” Document of Joint Collaborative Team on 3D Video Coding Extension Development, JCT3V-A0126, July, 2012.
4. HTM-4.0.1, <https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-4.0.1/>
5. D. Rusanovskyy, K. Müller, A. Vetro, “Common test conditions of 3DV Core Experiments,” Document of Joint Collaborative Team on 3D Video Coding Extension Development, JCT3V-A1100, July, 2012.