|  |  |
| --- | --- |
| **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, CN, 13–19 Oct. 2012 | Document: JCT3V-B0047 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **3D-CE5.h related: Improvements for disparity vector derivation** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Jewon Kang Ying Chen Li Zhang Marta Karczewicz  5775 Morehouse Drive San Diego, CA 92121 USA | Tel: Email: | 1-858-651-8457 [jewonk@qti.qualcomm.com](mailto:jewonk@qti.qualcomm.com)  1-858-845-6589  [cheny@qti.qualcomm.com](mailto:cheny@qti.qualcomm.com)  +1-858-651-6660 [lizhang@qti.qualcomm.com](mailto:lizhang@qti.qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

This contribution presents modifications for disparity vector derivation in the current 3D-HEVC. In the current 3D-HEVC design, a disparity vector is estimated with searching disparity motion vectors (DMV) of coded blocks in spatial and temporal neighbors as well as derived disparity vectors (DDV) which may be stored to each block. However, using DDV requires large memory bandwidth not only for accessing the reference pictures but also for storing information related to the DDV since the DDV needs to be present for each block in all the pictures of the decoded picture buffer (DPB). Thus, in this contribution, it is proposed that DDV is stored only in the current view component to reduce the memory size, and not in a reference picture. On top of that, the contribution proposes a temporal picture selection method that allows up to two candidates (i.e., the same number with the current design). In summary, the proposal reduces the memory size as much as for storing the DDV in the DPB, and achieves a coding gain about 0.1% bit-rate saving for texture views.

# Introduction

In the current 3D-HEVC [1], a disparity vector is derived using disparity motion vectors in spatial and temporal neighboring blocks. Several block positions are pre-defined, and each of them is examined in order. Once a disparity motion vector is found in a neighboring block, the disparity motion vector becomes the output (disparity vector) of the derivation process, and the process is finished. For the block positions, we use five spatial neighboring blocks, i.e., the below-left, left, above-left, above, and above-right [2], and two temporal neighboring blocks, i.e., blocks at the center and the bottom-right positions of the co-located PU. Up to two temporal reference frame candidates are checked for the temporal neighboring blocks.

A derived disparity vector (DDV) is generated when a PU employs inter-view motion vector prediction, and stored to the PU for the purpose of a predictor [3]. This disparity information can be used for the disparity vector derivation process when the current PU does not contain a disparity motion vector from temporal and spatial neighbors. Specifically, during the process, if no disparity vector is found from the positions aforementioned, the DDVs are successively checked in the five neighboring blocks in spatial and the center point in temporal. The DDVs are signaled in every block with a flag indicating presence of the DDV and vector itself for the future derivation process.

However, the DDV requires large memory for storing the flag and the vector itself to every block. Unlike the disparity motion vectors stored in an existing motion field of a picture, the DDV needs to reserve additional memory bandwidth that can be approximately equal to a half of the memory size for the motion fields for a B picture in the worst case. That is, in a practical aspect, removing DDV from temporal neighbours can take an advantage in a system because the entire memory size of the DPB can be substantially decreased. Therefore, this proposal simplifies the DDV to reduce the memory size, and propose a method for selecting a temporal picture candidate to compensate a coding loss which may be caused by the lack of the temporal DDV.

# Proposal

The proposed method has two folds. First, this contribution proposes a simplification of the disparity vector derivation process by removing the DDV candidates from the temporal reference pictures, to reduce the storage required for DPB. Second, this contribution proposes an algorithm to select the second temporal candidate picture during the derivation process in addition to the co-located picture as the first temporal candidate picture. With substantial memory reduction, the proposed method provides about 0.1% BD rate saving for multiview video coding (texture only).

## Temporal candidate picture selection

The disparity derivation from temporal block candidates has a two-step process as HTM4.0. Temporal candidate pictures are selected, and then a regional searching is performed. HTM4.0 uses maximally two reference pictures from reference frame lists, i.e., RefPicList0 and RefPicList1 (if available), for the derivation.

In the proposed method, up to 2 temporal candidate pictures may be put into a candidate list. The first candidate picture is the co-located picture as used for Temporal Motion Vector Prediction (TMVP) in HEVC without low delay check. The co-located picture is indicated in a slice header. The second picture is derived in the reference picture lists with the ascending order of reference picture indices, and added into the candidate list, given as follows:

1. A random access point (RAP) is searched in the reference picture lists. If found, the RAP is placed into the candidate list for the second picture and the derivation process is completed. In a case that the RAP is not available for the current picture, go to step (2).
2. A picture with the lowest temporalID (TID) is searched out and placed into the candidate list of the temporal pictures as the second entry.
3. If multiple pictures with the same lowest TID exist, a picture of less POC difference with the current picture is chosen.

As shown in the above description, the second temporal candidate picture is chosen in a way that disparity motion vectors can have more chance to be present in the picture.

This process can be done in the slice level and be invoked only once per slice.

## Storing of DDV only for the current picture

In HTM 4.0, there are six candidate blocks for the DDVs that are checked for the final disparity vector, depending on their presence. These blocks are five spatial neighboring blocks, i.e., the below-left, left, above-left, above, and above-right, and from one temporal neighboring block.

As emphasized for the memory problem of the DDV, it is proposed that the DDV doesn’t need to be stored for a decoded view component. Thus, the five spatial neighboring blocks are checked for the DDV. In other words, the DDVs are stored only for the current view component. After a view component is decoded, the DDVs of the view component are removed from the memory.

Note, DDV could be stored only for the above several lines that cover the above-left, above, and above-right neighbouring blocks.

# Experimental results

Simulation results of the proposal are shown in Table 1. The implementation was based on HTM4.0 [1]. Simulations are done under common test conditions [4].

As shown in Table 1, the proposed algorithm achieves small coding gain, i.e., 0.1% BD-rate reduction for video-only, synthesized-only, and video/synthesized combined. The encoding and decoding measurement time may be not accurate because of hybrid CPUs used for the simulations.

**Table 1: Proposed algorithm VS HTM4.0.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Video 0 | Video 1 | Video 2 | Video only | Synthesized only | Coded & synthesized | Enc time | Dec time | Ren  time |
| Balloons | 0.0% | -0.1% | -0.8% | -0.2% | -0.1% | -0.1% | 103.3% | 99.3% | 109.4% |
| Kendo | 0.0% | -0.4% | -0.3% | -0.1% | -0.1% | -0.1% | 101.6% | 101.0% | 106.2% |
| Newspapercc | 0.0% | -0.4% | -0.4% | -0.2% | -0.2% | -0.1% | 101.7% | 100.3% | 109.5% |
| GhostTownFly | 0.0% | -0.1% | -0.2% | 0.0% | 0.0% | 0.0% | 96.3% | 100.5% | 107.1% |
| PoznanHall2 | 0.0% | -0.1% | -0.8% | -0.2% | -0.2% | -0.2% | 103.1% | 100.9% | 109.3% |
| PoznanStreet | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 89.0% | 100.7% | 107.1% |
| UndoDancer | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.0% | 93.9% | 100.4% | 105.3% |
| 1024x768 | 0.0% | -0.3% | -0.5% | -0.2% | -0.1% | -0.1% | 102.2% | 100.2% | 108.4% |
| 1920x1088 | 0.0% | -0.1% | -0.2% | -0.1% | 0.0% | 0.0% | 95.4% | 100.6% | 107.2% |
| **average** | **0.0%** | **-0.2%** | **-0.3%** | **-0.1%** | **-0.1%** | **-0.1%** | **98.3%** | **100.4%** | **107.7%** |

It is noticed that in HTM 4.0, temporal scalability is not supported in the encoder, thus all pictures have temporalId set equal to 0. As HTM will be aligned to HM later versions wherein temporal scalability is supported, results on turning off temporal scalability in the anchor are also provided, as shown in Table 2. Similar cording performance can be observed in this scenario. The encoding and decoding measurement time may be not accurate because of hybrid CPUs used for the simulations.

**Table 2: Proposed algorithm VS HTM4.0 (both with temporal scalability supported).**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Video 0 | Video 1 | Video 2 | Video only | Synthesized only | Coded & synthesized | Enc time | Dec time | Ren  time |
| Balloons | 0.0% | -0.3% | -0.7% | -0.2% | -0.1% | -0.1% | 104.2% | 100.9% | 94.8% |
| Kendo | 0.0% | -0.5% | -0.3% | -0.2% | -0.1% | -0.1% | 105.2% | 93.9% | 90.6% |
| Newspapercc | 0.0% | -0.5% | -0.5% | -0.2% | -0.2% | -0.2% | 103.4% | 97.0% | 94.9% |
| GhostTownFly | 0.0% | 0.0% | -0.2% | 0.0% | 0.0% | 0.0% | 98.7% | 101.4% | 89.0% |
| PoznanHall2 | 0.0% | 0.1% | -0.4% | -0.1% | 0.0% | -0.1% | 102.4% | 98.2% | 99.4% |
| PoznanStreet | 0.0% | -0.1% | -0.4% | -0.1% | 0.0% | -0.1% | 102.2% | 89.7% | 84.5% |
| UndoDancer | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 103.7% | 92.0% | 86.0% |
| 1024x768 | 0.0% | -0.4% | -0.5% | -0.2% | -0.1% | -0.1% | 104.3% | 97.2% | 93.4% |
| 1920x1088 | 0.0% | 0.0% | -0.3% | -0.1% | 0.0% | 0.0% | 101.7% | 95.2% | 89.6% |
| **average** | **0.0%** | **-0.2%** | **-0.4%** | **-0.1%** | **-0.1%** | **-0.1%** | **102.8%** | **96.1%** | **91.2%** |

# Conclusion

The proposed algorithm stores derived disparity vectors (DDV) only for blocks in the same view component and removes the DDV storage from the DPB to reduce the memory consumption. The selection of temporal candidate pictures is also modified to prefer pictures that have more chances to contain disparity motion vectors. While significantly reducing the memory consumption, this method provides a coding gain about 0.1% BD-rate reduction.

# References

1. G. Tech, K. Wegner, Y. Chen, S. Yea, “3D-HEVC Test Model 1”, JCT3VC-A1005, Jul. 2012.
2. L. Zhang, Y. Chen, M. Karczewicz, “3D-CE5.h: Disparity vector generation results,” JCT-3DC-A0097, Jul. 2012.
3. J. Sung, M. Koo, S. Yea, "3D-CE5.h: Simplification of disparity vector derivation for HEVC-based 3D video coding," JCT3V-A0126, Jul. 2012.
4. D. Rusanovskyy, K. Müller, A. Vetro, “Common Test Conditions of 3DV Core Experiments,” JCT3V-A1100, Jul. 2012.

# Patent rights declaration(s)

**Qualcomm Incorporated 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).**