|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  6th Meeting: Geneva, CH, 25 Oct. – 1 Nov. 2013 | Document: JCT3V-F0103 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Definite and sparse depth access for DoNBDV and VSP** | | |
| *Status:* | Input Document to JCT-3V | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Tomohiro Ikai  1-9-2 Nakase, Mihama-ku, Chiba-shi, Chiba 261-8520  JAPAN | Tel: Email: | +81-43-299-8526 [ikai.tomohiro@sharp.co.jp](mailto:ikai.tomohiro@sharp.co.jp) |
| *Source:* | Sharp Corporation | | |

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

# Abstract

This proposal presents a depth access restriction to avoid arbitrary depth access. The depth access is carried out for depth refinement DV(DoNBDV) or view synthesis prediction. Although the depth access is sparsely accessed in each block thanks to only corner point access but the top-left position of the depth access is not restricted. Therefore it is proposed to restrict the vertical position of the depth access to be zero (Proposal 1). The depth access is considered as a kind of parameter access since that is used for motion parameter derivation and needs to be completed before motion compensation process and the other parameter access is 4x4 block basis at the minimum. Therefore it is proposed to set corner position of vertical position to a multiple of 4 (Proposal 2) or 8 (Proposal 3) to reduce depth access further.

The experiment results show 0.0 %, 0.1 % and 0.0 % coding gain in synthesis in Proposal1, Proposal2 and Proposal3 respectively.

# Introduction

The process of depth based parameter derivation such as DoNBDV and VSPshould be minimized since that is additional process to HEVC version1 and needs to be finished before motion compensation. If the practical codec utilizes pipeline system or micro parallelism and this additional parameter derivation process requires much more time, it is needed to re-design / re-schedule the pipeline or parallelism and this is a burden and needs cost. If the codec doesn’t utilize that things, or the codec needs to do the process linearly, the process rather have to be minimized because that process is not pixel access where some plural pixels of four or eight can be processed at one time and is already cached, but parameter-like access where is one by one and some extent of arbitrary access are inevitable.

Therefore it is proposed to restrict vertical position to be zero as for left-top position of depth block relative to texture block (proposal 1) and it is proposed to only access 4 or 8 lines (proposal2 and 3).

Proposal 2 and 3 include Proposal 2

Proposal 3 includes Proposal 2 and VSP partitioning change for AMP as the same as JCT3V-F0102.

# Proposal

## Proposal1: Restrict vertical disparity vector to avoid arbitrary depth access

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION 1

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION\_PLUS 0

#define SHARP\_VSP\_AMP 0

Restrict the vertical position of the depth access to be zero (highlighted as Yellow)



Table 1 Summary of proposal 1 (Test1)

|  |  |  |
| --- | --- | --- |
|  | HTM8 | Proposal 1 |
| (A)Relative top-left position | yTL = yP + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) | yTL = yP |

## Proposal2: Modify corner position to achieve sparse depth access (4 pixel line only)

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION 1

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION\_PLUS 1

#define SHARP\_VSP\_AMP 0

In addition to Proposal 1, the following modification on vertical position is proposed.

1. Define the bottom position as yTL + ySubB + nSubSubBlkH in VSP split decision and the DepthRefinement case (splitFlag is equal to 0)
2. Define the bottom position as equal to the top position when splitFlag is equal to 1



1. Define the bottom position as yTL + ySubB + nSubSubBlkH ~~– 1~~



1. Define the bottom position as equal to the top position

Table 2 Summary of proposal 2 (Test2)

|  |  |  |
| --- | --- | --- |
|  | HTM8 | Proposal 2.b |
| 1. Relative top-left position | yTL = yP + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) | yTL = yP |
| 1. Bottom address in VSP split decision | yTL + ySubB + nSubSubBlkH – 1 | yTL + ySubB + nSubSubBlkH ~~– 1~~ |
| 1. Bottom address for max of four corner | yTL + ySubB + nSubSubBlkH | splitFlag ? yD0 :  (yTL + ySubB + nSubSubBlkH  ~~– 1~~)) |

## Proposal3: Modify corner position amd AMP partitioning to achieve sparse depth access (8 pixel line only)

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION 1

#define SHARP\_DEPTH\_ACCESS\_RESTRICTION\_PLUS 2

#define SHARP\_VSP\_AMP 1

In addition to Proposal 1, the following modification on vertical position is proposed.

1. Define the bottom position as yTL + ySubB + nSubSubBlkH in VSP split decision and the DepthRefinement case (splitFlag is equal to 0) same as Proposal2
2. Define the bottom position as equal to the top position when splitFlag is equal to 1 same as Proposal2
3. Define the top position as multiple of 8 with rounding
4. When AMP, 8x4 or 4x8 is constantly used when heigh % 8 != 0 or width % 8 != 0 same as JCT3V-F0102

able 2 Summary of proposal 2 (Test2)

|  |  |  |
| --- | --- | --- |
|  | HTM8 | Proposal 2.b |
| 1. Relative top-left position | yTL = yP + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) | yTL = yP |
| 1. Bottom address in VSP split decision | yTL + ySubB + nSubSubBlkH – 1 | yTL + ySubB + nSubSubBlkH ~~– 1~~ |
| 1. Bottom address for max of four corner | yTL + ySubB + nSubSubBlkH | splitFlag ? yD0 :  (yTL + ySubB + nSubSubBlkH  ~~– 1~~)) |
| 1. Top address for max of four corner | yTL + ySubB | ((yTL + ySubB + 4) >> 3) << 3 |
| 1. AMP change | When height < 8 || width < 8, no split | When height %8 || width % 8, constant split |

# Proposed Text

Proposal1

**H.8.5.5.2 Derivation process for a disparity sample array**

Inputs to this process are:

* a luma location ( xP, yP ) relative to the top-left luma sample of the current picture,
* a disparity vector mvDisp,
* a view order index refViewIdx specifying a reference view,
* a view order index depthViewIdx specifying the view the depth should be derived from
* variables nPSW and nPSH specifying a width and a height, respectively
* a variable splitFlag.

Outputs of this process are:

* a (nPSW)x(nPSH) array disparitySamples of disparities values.

Let refDepPic the picture in the current access unit with ViewIdx( refDepPic ) equal to ViewIdx and DepthFlag( refDepPic ) equal to 1.

Let refDepPels be an array of reconstructed depth samples refDepPic. The luma location (xTL, yTL) of top-left luma sample of a block in refDepPels is derived by

* 1. xTL = xP + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) (H‑) (H‑)
  2. yTL = yP ~~+ ( ( mvDisp[ 1 ] + 2 ) >> 2 )~~ (H‑) // (A)

1. The variables nSubBlkW and nSubBlkH are derived as specified in the following:
   1. nSubBlkW = splitFlag ? 8 : nPSW (H‑257)
   2. nSubBlkH = splitFlag ? 8 : nPSH (H‑258)

The array disparitySamples of size (nPSW)x(nPSH) is derived as specified in the following:

* For sBy in the range of 0 to ( ( nPSH / nSubBlkH) – 1 ), inclusive, the following applies:
  + For sBx in the range of 0 to ( ( nPSW / nSubBlkW) – 1 ), inclusive, the following applies:
    - The variables xB, yB, xP0, yP0, xP1, yP1, are derived as specified in the following:

xB = sBx \* nSubBlkW  
 yB = sBy \* nSubBlkH  
 xP0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB )  
 yP0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB )  
 xP1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB + nSubBlkW – 1 )  
 yP1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB + nSubBlkH – 1 )

* + - The variable nSubSubBlkW is set equal to nSubBlkW and the variable nSubSubBlkH is set equal to nSubBlkH.
    - When splitFlag is equal to 1, nSubSubBlkW and nSubSubBlkW are modified as follows:
      * The variable horSplitFlag is derived as specified in the following:
        1. horSplitFlag = ( refDepPels[ xP0 ][ yP0 ] > refDepPels[ xP1 ][ yP1 ] )  
            = = ( refDepPels[ xP1 ][ yP0 ] > refDepPels[ xP0 ][ yP1] ) ) (H‑259)
        2. nSubSubBlkW = horSplitFlag ? nSubSubBlkW : ( nSubSubBlkW >> 1 ) (H‑260)
        3. nSubSubBlkH = horSplitFlag ? ( nSubSubBlkH >> 1 ) : nSubSubBlkW (H‑261)
    - The derivation process for a disparity sample block as specified in subclause H.8.5.5.2.1 is invoked with the luma location ( xB, yB ), variables nSubBlkW and nSubBlkH, the array of reconstructed depth samples refDepPels, the luma location ( xTL, yTL ), the variables nSubSubBlkW and nSubSubBlkW, the view order index refViewIdx, and the array disparitySamples as the inputs, and the output is the modified array disparitySamples.

**Proposal2**

**H.8.5.4.2 Derivation process for a disparity sample array**

Inputs to this process are:

* a luma location ( xP, yP ) relative to the top-left luma sample of the current picture,
* a disparity vector mvDisp,
* a view order index refViewIdx specifying a reference view,
* a view order index depthViewIdx specifying the view the depth should be derived from
* variables nPSW and nPSH specifying a width and a height, respectively
* a variable splitFlag.

Outputs of this process are:

* a (nPSW)x(nPSH) array disparitySamples of disparities values.

Let refDepPic the picture in the current access unit with ViewIdx( refDepPic ) equal to ViewIdx and DepthFlag( refDepPic ) equal to 1.

Let refDepPels be an array of reconstructed depth samples refDepPic. The luma location (xTL, yTL) of top-left luma sample of a block in refDepPels is derived by

* 1. xTL = xP + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) (H‑) (H‑)
  2. yTL = yP ~~+ ( ( mvDisp[ 1 ] + 2 ) >> 2 )~~ (H‑) // (A)

1. The variables nSubBlkW and nSubBlkH are derived as specified in the following:
   1. nSubBlkW = splitFlag ? 8 : nPSW (H‑)
   2. nSubBlkH = splitFlag ? 8 : nPSH (H‑)

The array disparitySamples of size (nPSW)x(nPSH) is derived as specified in the following:

* For sBy in the range of 0 to ( ( nPSH / nSubBlkH) – 1 ), inclusive, the following applies:
  + For sBx in the range of 0 to ( ( nPSW / nSubBlkW) – 1 ), inclusive, the following applies:
    - The variables xB, yB, xP0, yP0, xP1, yP1, are derived as specified in the following:

xB = sBx \* nSubBlkW  
 yB = sBy \* nSubBlkH  
 xP0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB )  
 yP0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB )  
 xP1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB + nSubBlkW – 1 )  
 yP1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB + nSubBlkH ~~– 1~~) ) // (B)

* + - The variable nSubSubBlkW is set equal to nSubBlkW and the variable nSubSubBlkH is set equal to nSubBlkH.
    - When splitFlag is equal to 1, nSubSubBlkW and nSubSubBlkW are modified as follows:
      * The variable horSplitFlag is derived as specified in the following:
        1. horSplitFlag = ( refDepPels[ xP0 ][ yP0 ] > refDepPels[ xP1 ][ yP1 ] )  
            = = ( refDepPels[ xP1 ][ yP0 ] > refDepPels[ xP0 ][ yP1] ) ) (H‑)
        2. nSubSubBlkW = horSplitFlag ? nSubSubBlkW : ( nSubSubBlkW >> 1 ) (H‑)
        3. nSubSubBlkH = horSplitFlag ? ( nSubSubBlkH >> 1 ) : nSubSubBlkW (H‑)
    - The derivation process for a disparity sample block as specified in subclause H.8.5.5.2.1 is invoked with the luma location ( xB, yB ), variables nSubBlkW and nSubBlkH, the array of reconstructed depth samples refDepPels, the luma location ( xTL, yTL ), the variables nSubSubBlkW and nSubSubBlkW, the view order index refViewIdx, and the array disparitySamples as the inputs, and the output is the modified array disparitySamples.

**H.8.5.4.2.1 Derivation process for a disparity sample sub block**

…  
 xD0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB )

yD0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + ySubB )

xD1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB + nSubSubBlkW – 1)

yD1 = splitFlag ? yD0 : Clip3( 0, pic\_height\_in\_luma\_samples – 1, (yTL + ySubB + nSubSubBlkH)) .// (C)

~~yD1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, (y~~~~TL~~ ~~+ ySubB + nSubSubBlkH – 1)) // HTM80~~

maxDep = Max( maxDep, refDepPels[ xD0 ][ yD0 ] )  
 maxDep = Max( maxDep, refDepPels[ xD0 ][ yD1] )  
 maxDep = Max( maxDep, refDepPels[ xD1][ yD0 ] )  
 maxDep = Max( maxDep, refDepPels[ xD1][ yD1 ] )

**Proposal3**

(Proposal) The derivation process for a disparity sample array as specified in section H.8.5.5.2 is invoked with the luma location ( xP, yP ), the disparity vector mvDisp, the variable refViewIdx, the variable depthViewIdx, the variable nPSW equal to nPbW, the variable nPSH equal to nPbH, and the variable splitFlag equal to 1 as the inputs, and the output is the array disparitySamples of size (nPSW)x(nPSH).

**H.8.5.4.2 Derivation process for a disparity sample array**

Inputs to this process are:

* a luma location ( xP, yP ) relative to the top-left luma sample of the current picture,
* a disparity vector mvDisp,
* a view order index refViewIdx specifying a reference view,
* a view order index depthViewIdx specifying the view the depth should be derived from
* variables nPSW and nPSH specifying a width and a height, respectively
* a variable splitFlag.

Outputs of this process are:

* a (nPSW)x(nPSH) array disparitySamples of disparities values.

Let refDepPic the picture in the current access unit with ViewIdx( refDepPic ) equal to ViewIdx and DepthFlag( refDepPic ) equal to 1.

Let refDepPels be an array of reconstructed depth samples refDepPic. The luma location (xTL, yTL) of top-left luma sample of a block in refDepPels is derived by

* 1. xTL = xP + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) (H‑)

1. yTL = yP ~~+ ( ( mvDisp[ 1 ] + 2 ) >> 2 )~~ // (A) (H‑The variables nSubBlkW and nSubBlkH are derived as specified in the following:
   1. nSubBlkW = (splitFlag && !(nPSH % 8) && !( nPSH % 8) ) ? 8 : nPSW (H‑) // (E)
   2. nSubBlkH = (splitFlag && !(nPSH % 8) && !( nPSH % 8) ) ? 8 : nPSH (H‑) // (E)

The array disparitySamples of size (nPSW)x(nPSH) is derived as specified in the following:

* For sBy in the range of 0 to ( ( nPSH / nSubBlkH) – 1 ), inclusive, the following applies:
  + For sBx in the range of 0 to ( ( nPSW / nSubBlkW) – 1 ), inclusive, the following applies:
    - The variables xB, yB, xP0, yP0, xP1, yP1, are derived as specified in the following:

xB = sBx \* nSubBlkW  
 yB = sBy \* nSubBlkH  
 xP0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB )  
 yP0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB )  
 xP1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB + nSubBlkW – 1 )  
 yP1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB + nSubBlkH ~~– 1~~) // (B)

* + - The variable nSubSubBlkW is set equal to nSubBlkW and the variable nSubSubBlkH is set equal to nSubBlkH.
    - When nPSH % 8 is not equal to 0, nSubSubBlkW and nSubSubBlkH are modified as follows: // (E)

nSubSubBlkW = 8 (‑240)

nSubSubBlkH = 4 (‑241)

* + - Otherwise, when nPSW % 8 is not equal to 0, nSubSubBlkW and nSubSubBlkH are modified as follows:

nSubSubBlkW = 4 (‑240)

nSubSubBlkH = 8 (‑241)

* + - Otherwise, When splitFlag is equal to 1, nSubSubBlkW and nSubSubBlkW are modified as follows:
      * The variable horSplitFlag is derived as specified in the following:
        1. horSplitFlag = ( refDepPels[ xP0 ][ yP0 ] > refDepPels[ xP1 ][ yP1 ] )  
            = = ( refDepPels[ xP1 ][ yP0 ] > refDepPels[ xP0 ][ yP1] ) ) (H‑)
        2. nSubSubBlkW = horSplitFlag ? nSubSubBlkW : ( nSubSubBlkW >> 1 ) (H‑)
        3. nSubSubBlkH = horSplitFlag ? ( nSubSubBlkH >> 1 ) : nSubSubBlkW (H‑)

The derivation process for a disparity sample block as specified in subclause H.8.5.5.2.1 is invoked with the luma location ( xB, yB ), variables nSubBlkW and nSubBlkH, the array of reconstructed depth samples refDepPels, the luma location ( xTL, yTL ), the variables nSubSubBlkW and nSubSubBlkW, the view order index refViewIdx, and the array disparitySamples as the inputs, and the output is the modified array disparitySamples.

**H.8.5.4.2.1 Derivation process for a disparity sample sub block**

…  
 xD0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB )

yD0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, ((yTL + ySubB + 4) >> 3) <<3 ) // (D)

xD1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB + nSubSubBlkW – 1)

yD1 = splitFlag ? yD0 : Clip3( 0, pic\_height\_in\_luma\_samples – 1, (yTL + ySubB + nSubSubBlkH)) // (C)

~~yD1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, (y~~~~TL~~ ~~+ ySubB + nSubSubBlkH – 1)) // HTM80~~

maxDep = Max( maxDep, refDepPels[ xD0 ][ yD0 ] )  
 maxDep = Max( maxDep, refDepPels[ xD0 ][ yD1] )  
 maxDep = Max( maxDep, refDepPels[ xD1][ yD0 ] )  
 maxDep = Max( maxDep, refDepPels[ xD1][ yD1 ] )

# Simulation results

Proposal 1: Relative vertical position is always zero (Test1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video PSNR / video bitrate | video PSNR / total bitrate | synth PSNR / total bitrate | enc time | dec time | ren time |
| Balloons | 0.0% | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 99.7% | 98.5% | 97.7% |
| Kendo | 0.0% | 0.2% | 0.0% | 0.0% | 0.0% | 0.0% | 99.5% | 95.6% | 96.3% |
| Newspaper\_CC | 0.0% | 0.0% | 0.1% | 0.0% | 0.0% | 0.1% | 99.8% | 99.2% | 98.4% |
| GT\_Fly | 0.0% | 0.1% | -0.1% | 0.0% | 0.0% | 0.0% | 99.6% | 100.1% | 100.0% |
| Poznan\_Hall2 | 0.0% | 0.1% | -0.1% | 0.0% | 0.0% | 0.0% | 100.0% | 99.4% | 100.0% |
| Poznan\_Street | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 100.0% | 99.0% | 99.6% |
| Undo\_Dancer | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 0.1% | 100.0% | 103.5% | 100.1% |
| 1024x768 | 0.0% | 0.1% | 0.1% | 0.0% | 0.0% | 0.0% | 99.6% | 97.8% | 97.5% |
| 1920x1088 | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 99.9% | 100.5% | 99.9% |
| **average** | **0.0%** | **0.1%** | **0.0%** | **0.0%** | **0.0%** | **0.0%** | **99.8%** | **99.3%** | **98.9%** |
|  |  |  |  |  |  |  |  |  |  |
| Shark | 0.0% | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 99.8% | 100.8% | 101.8% |

Proposal 2: 4 line base access (Test2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video PSNR / video bitrate | video PSNR / total bitrate | synth PSNR / total bitrate | enc time | dec time | ren time |
| Balloons | 0.0% | -0.2% | 0.1% | 0.0% | 0.0% | 0.0% | 100.2% | 100.5% | 100.2% |
| Kendo | 0.0% | 0.0% | -0.2% | 0.0% | 0.0% | -0.1% | 100.2% | 100.0% | 99.6% |
| Newspaper\_CC | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 100.1% | 100.3% | 99.7% |
| GT\_Fly | 0.0% | 0.7% | 0.6% | 0.1% | 0.1% | 0.0% | 99.9% | 101.1% | 100.0% |
| Poznan\_Hall2 | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | -0.1% | 99.7% | 100.9% | 100.1% |
| Poznan\_Street | 0.0% | 0.2% | 0.0% | 0.0% | 0.0% | 0.0% | 100.2% | 98.8% | 99.6% |
| Undo\_Dancer | 0.0% | -0.4% | -0.6% | -0.2% | -0.2% | -0.1% | 99.7% | 99.6% | 100.2% |
| 1024x768 | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 100.2% | 100.3% | 99.8% |
| 1920x1088 | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | -0.1% | 99.9% | 100.1% | 100.0% |
| **average** | **0.0%** | **0.0%** | **0.0%** | **0.0%** | **0.0%** | **-0.1%** | **100.0%** | **100.2%** | **99.9%** |
|  |  |  |  |  |  |  |  |  |  |
| Shark | 0.0% | -0.4% | -0.3% | -0.1% | -0.1% | -0.1% | 99.9% | 97.8% | 98.2% |

Proposal 2: 8 line base access (Test2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video PSNR / video bitrate | video PSNR / total bitrate | synth PSNR / total bitrate | enc time | dec time | ren time |
| Balloons | 0.0% | -0.2% | -0.1% | 0.0% | 0.0% | 0.0% | 100.4% | 102.5% | 101.6% |
| Kendo | 0.0% | 0.0% | -0.2% | 0.0% | 0.0% | -0.1% | 100.4% | 101.6% | 101.1% |
| Newspaper\_CC | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 100.2% | 99.5% | 100.6% |
| GT\_Fly | 0.0% | 0.5% | 0.4% | 0.1% | 0.1% | 0.1% | 99.8% | 100.9% | 99.9% |
| Poznan\_Hall2 | 0.0% | 0.1% | -0.4% | -0.1% | -0.1% | -0.2% | 100.0% | 99.2% | 99.7% |
| Poznan\_Street | 0.0% | 0.2% | 0.1% | 0.1% | 0.0% | 0.0% | 100.1% | 99.0% | 99.8% |
| Undo\_Dancer | 0.0% | 0.5% | 0.8% | 0.1% | 0.1% | 0.1% | 100.1% | 100.8% | 100.1% |
| 1024x768 | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | -0.1% | 100.3% | 101.2% | 101.1% |
| 1920x1088 | 0.0% | 0.3% | 0.2% | 0.0% | 0.0% | 0.0% | 100.0% | 100.0% | 99.9% |
| **average** | **0.0%** | **0.2%** | **0.1%** | **0.0%** | **0.0%** | **0.0%** | **100.1%** | **100.5%** | **100.4%** |
|  |  |  |  |  |  |  |  |  |  |
| Shark | 0.0% | 0.1% | 0.2% | 0.0% | 0.0% | 0.0% | 100.1% | 96.0% | 96.6% |

# Conclusion

It is proposed to restrict the vertical position of the depth access to be zero (Proposal1: option1) or a multiple of 4 (Proposal2) or 8 (Proposal 3) to minimize the depth access cost which is required for the additional parameter derivation process before motion compensation process. It is recommended to adopt Proposal2 or Proposal 3 (Proposal 1 is included in both cases).

# Patent rights declaration(s)

**Sharp Corporation 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).**