|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  16th Meeting: San José, US, 9–17 Jan. 2014 | Document: JCTVC-P0063\_r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-SCE1: Asymmetric 3D LUT for Color Gamut Scalability** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Xiang Li  Jianle Chen  Marta Karczewicz  5775 Morehouse Drive San Diego, CA 92121, USA | Tel: Email: | +1 858 658 3923 [lxiang@qti.qualcomm.com](mailto:lxiang@qti.qualcomm.com)  [cjianle@qti.qualcomm.com](mailto:cjianle@qti.qualcomm.com)  [martak@qti.qualcomm.com](mailto:martak@qti.qualcomm.com) |
| *Source:* | Qualcomm Inc. | | |

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

# Abstract

In this proposal, a method based on asymmetric 3D lookup table (up to 384 entries) is proposed for color gamut scalability. It is reported that on average 8.2% (AI-10bit), 8.2% (AI-8bit), 6.3% (RA-10bit) and 6.2% (RA-8bit) luma BD rate reduction was achieved over SCE-1 use case 1 anchor , and 8.4% (AI-10bit), 8.4% (AI-8bit), 6.6% (RA-10bit) and 6.4% (RA-8bit) luma BD rate reduction over SCE-1 use case 2 anchor. Note that the SCE-1 anchors employ weighted prediction to compensate color gamut difference between layers.

# Introduction

In SHVC, when the color gamut of base and enhancement layers is different, the efficiency of inter-layer prediction is rather low. To address this problem, 3D lookup table (3D LUT) based color prediction was proposed in [1] (9x9x9x3 table) and [2] (17x17x17x3 table). Although good coding efficiency was achieved, the size of the 3D LUT is too big for hardware implementations, especially for mobile devices. To reduce the table size while keeping good coding efficiency, a method based on asymmetric 3D LUT is proposed in this contribution, where up to 384 entries is employed for the 3D LUT. Simulations show that significant luma BD rate reduction was achieved on average over SCE 1 anchors, where weighted prediction is used to compensate the color gamut difference between layers.

# Proposed method

## Asymmetric 3D LUT

In [1] and [2], input Y-U-V space is evenly split into same-size cubic partitions so that for each partition the conversion between two color gamut can be approximated by 3-D linear prediction, namely

,

where *YE*, *UE* and *VE* represent original enhancement layer picture samples, *YB*, *UB*, *VB* denote samples of upsampled (if necessary) base layer reconstructed picture, *a, b, c* and *d* are coefficients of the color prediction model. In [1] and [2], the color prediction coefficients *a, b, c* and *d* are first converted to vertexes of a cubic partition and then signaled in bitstream.

To obtain high coding efficiency, 9x9x9x3 (2187 entries, 8x8x8 partitions for each component) and 17x17x17x3 (14739 entries, 16x16x16 partitions for each component) were employed in [1] and [2], respectively. Such large non-fixed tables could cause a serious complexity issue, for both hardware and software implementations, especially for mobile devices.

To reduce the table size while keeping the coding efficiency, it is proposed to use asymmetric look up table. That is, more partitions are used along luma component direction while less partitions are made along chroma component direction. In the current test, up to 8x2x2 partitions are employed. For each partition, information of color prediction, such as *a, b, c, d* coefficients or converted vertexes, are signaled. Therefore, the proposed asymmetric table has up to 8x2x2x4x3=384 entries.

## Derivation of color prediction coefficients

Color prediction coefficients are derived for each partition by minimizing the distortion between converted samples and original enhancement signal, namely

.

Then the color prediction coefficients are converted to the vertexes of the corresponding cuboid based on the position of the cuboid.

When the color prediction process is conducted before up-sampling, the color prediction coefficients are derived by minimizing the distortion between converted samples and the down-sampled enhancement signal.

## Update 3D LUT at picture level

The 3D LUT is signaled in PPS and may be updated in slice segment header when necessary. When the 3D LUT is updated in a slice segment header, the updated 3D LUT is applied for the related picture.

At the encoder side, the optimal 3D LUT is derived for each enhancement picture. First, the 3D LUT size is determined based on the bit cost of 3D LUT of previously coded frames. Then the 3D LUT is derived by minimizing the distortion between converted samples and original enhancement signal for each cuboid. Subsequently, the derived 3D LUT is compared to the one signaled in PPS and the one leading to smaller distortion is selected for coding the current picture.

# Proposed text

## Syntax and semantics

|  |  |
| --- | --- |
| pic\_parameter\_set\_rbsp( ) { | Descriptor |
| … |  |
| **slice\_segment\_header\_extension\_present\_flag** | u(1) |
| if(nuh\_layer\_id > 0) { |  |
| **pps\_cgs\_idc** | ue(v) |
| if(pps\_cgs\_idc==1 || pps\_cgs\_idc==3) |  |
| 3D\_LUT\_color\_data() |  |
| } |  |
| … |  |
| } |  |

**pps\_cgs\_idc** specifies that how CGS lookup table is signaled. pps\_cgs\_idc shall be in the range of 0 and 3, inclusively. pps\_cgs\_idc equal to 0 specifies that CGS table is not signaled. pps\_cgs\_idc equal to 1 specifies that CGS table is only signaled in PPS. pps\_cgs\_idc equal to 2 specifies that CGS table will be only signaled in slice header. pps\_cgs\_idc equal to 3 specifies that CGS table will be signaled in PPS and may be updated in slice header. When not present, pps\_cgs\_idc is inferred to be equal to 0.

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | Descriptor |
| … |  |
| if(nul\_layer\_id > 0) { |  |
| if(pps\_cgs\_idc==3) |  |
| **slice\_cgs\_overwrite\_pps\_flag** | u(1) |
| if(pps\_cgs\_idc==2 || slice\_cgs\_overwrite\_pps) |  |
| 3D\_ LUT\_ color\_data() |  |
| } |  |
| byte\_alignment( ) |  |
| } |  |

**slice\_cgs\_overwrite\_pps\_flag** equal to 1 specifies that CGS table will be updated in the current slice header. slice\_cgs\_overwrite\_pps\_flag equal to 0 specifies that CGS table will not be updated in the current slice. When not present, slice\_cgs\_overwrite\_pps\_flag is inferred to be equal to 0.

|  |  |
| --- | --- |
| 3D\_ LUT\_ color\_data ( ) { | **Descriptor** |
| **cur\_octant\_depth** | u(3) |
| **cur\_y\_part\_num\_log2** | u(2) |
| **input\_bit\_depth\_minus8** | u(3) |
| **output\_bit\_depth\_minus8** | u(3) |
| **res\_quant\_bit** | u(2) |
| coding\_octant( 0, 0, 0, 0, 1 << InputBitDepth) |  |
| } |  |

**cur\_octant\_depth** indicates the maximal split depth for the Y-U-V space of the current table.

**cur\_y\_part\_num\_log2** specifies the number of Y partitions for the smallest cubic partition. Variable YPartNum is derived as follows.

YPartNum = 1 << cur\_y\_part\_num\_log2

**input\_bit\_depth\_minus8** specifies the bit depth of the LUTs entries. Variable InputBitDepth is derived as follows:

InputBitDepth = 8 + input\_bit\_depth\_minus8

**output\_bit\_depth\_minus8** specifies the bit depth of the LUT output. Variable OutputBitDepth is derived as follows:

OutputBitDepth = 8 + output\_bit\_depth\_minus8

**res\_quant\_bit** specifies that the number of bits used in quantizing vertex residues. The quantization is achieved by right shifting vertex residues by res\_quant\_bit.

|  |  |
| --- | --- |
| coding\_octant (depth, y,u,v,length) { | **Descriptor** |
| if ( depth < cur\_octant\_depth ) |  |
| **split\_octant\_flag** | u(1) |
| if ( split\_octant\_flag ) { |  |
| for( l = 0 ; l < 2 ; l++ ) |  |
| for( m = 0 ; m < 2 ; m++ ) |  |
| for( n = 0 ; n < 2 ; n++ ) |  |
| coding\_octant ( depth+1, y+l\*length/2, u+m\*length/2,v+n\*length/2, length/2) |  |
| } |  |
| else { |  |
| for( i = 0 ; i < YPartNum ; i++ ) |  |
| for( vertex = 0 ; vertex < 4 ; vertex++ ) { |  |
| **encoded\_vertex\_flag** | u(1) |
| if( encoded\_vertex\_flag ) { |  |
| **resY**[yIdx][uIdx][vIdx][vertex] | se(v) |
| **resU**[yIdx][uIdx][vIdx][vertex] | se(v) |
| **resV**[yIdx][uIdx][vIdx][vertex] | se(v) |
| **}** |  |
| } |  |
| } |  |
| } |  |

**split\_octant\_flag** equal to 1 specifies that an octant is further split into eight octants with half length in all directions for the purpose of vertices residuals octant coding. When not present, it is inferred to be equal to 0.

**encoded\_vertex\_flag** equal to 1 specifies that the residues of the vertex with index [yIdx][uIdx][vIdx][vertex] are present. encoded\_vertex\_flag equal to 0 specifies that the residues for the vertex are not present. When not present, the flag is inferred to be equal to zero.

Variable yIdx is derived as

yIdx=(y+i\*(length>>cur\_y\_part\_num\_log2))>>(InputBitDepth-cur\_octant\_depth-cur\_y\_part\_num\_log2)

Variable uIdx is derived as

uIdx=u>>(InputBitDepth-cur\_octant\_depth)

Variable vIdx is derived as

vIdx=v>>(InputBitDepth-cur\_octant\_depth)

**resY**[yIdx][uIdx][vIdx][vertex]**, resU**[yIdx][uIdx][vIdx][vertex]**, resV**[yIdx][uIdx][vIdx][vertex] are the difference between the Y, U, and V components of the vertex with index [yIdx][uIdx][vIdx][vertex] and the predicted Y, U, and V component values for this vertex respectively. When not present, they are inferred to be equal to 0.

Please note that the above residues are coded with 3rd-order exp-golomb coding.

## Process of 3D LUT reconstruction

The proposed asymmetric 3D LUT is reconstructed by signaled residual table resX and the predicted table predX as follows, where X indicates Y, U, and V.

lutX[yIdx][uIdx][vIdx][vertex]=(resX[yIdx][uIdx][vIdx][vertex]<<res\_quant\_bit)

+predX[yIdx][uIdx][vIdx][vertex]

predX[yIdx][uIdx][vIdx][vertex] is derived as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [yIdx][uIdx][vIdx][vertex] | vertex=0 | vertex=1 | vertex=2 | vertex=3 |
| predY[yIdx][uIdx][vIdx][vertex] | yIdx<<yoShift | yIdx<<yoShift | yIdx<<yoShift | (yIdx+1)<<yoShift |
| predU[yIdx][uIdx][vIdx][vertex] | uIdx<<uoShift | (uIdx+1)<<uoShift | (uIdx+1)<<uoShift | (uIdx+1)<<uoShift |
| predV[yIdx][uIdx][vIdx][vertex] | vIdx<<voShift | vIdx<<voShift | (vIdx+1)<<voShift | (vIdx+1)<<voShift |

where yoShift = OutputBitDepth - cur\_octant\_depth - cur\_y\_part\_num\_log2, and

uoShift = voShift = OutputBitDepth - cur\_octant\_depth.

## Decoding process

The input to this process is a (y,u,v) triplet in one color gamut. The output of the process is the triplet (Y,U,V) in another color gamut.

First, intermediate variales are derived as follows.

yShift2Idx = InputBitDepth - cur\_octant\_depth - cur\_y\_part\_num\_log2

uShift2Idx = InputBitDepth - cur\_octant\_depth

vShift2Idx = InputBitDepth - cur\_octant\_depth

nMappingShift = yShift2Idx + uShift2Idx

nMappingOffset = 1 << (nMappingShift - 1)

Then for each (y,u,v) triplet in up-sampled picture following process is applied

yIdx = y >> yShift2Idx

uIdx = u >> uShift2Idx

vIdx = v >> vShift2Idx

deltaY = y – (yIdx << yShift2Idx)

deltaU = u – (uIdx << uShift2Idx)

deltaV = v – (vIdx << vShift2Idx)

Y = lutY[yIdx][uIdx][vIdx][0] + (((((deltaU - deltaV) \* lutY[yIdx][uIdx][vIdx][1] + deltaV \* lutY[yIdx][uIdx][vIdx][2] - deltaU \* lutY[yIdx][uIdx][vIdx][0]) << yShift2Idx) - (deltaY << uShift2Idx) \* (lutY[yIdx][uIdx][vIdx][2] - lutY[yIdx][uIdx][vIdx][3]) + nMappingOffset) >> nMappingShift)

U = lutU[yIdx][uIdx][vIdx][0] + (((((deltaU - deltaV) \* lutU[yIdx][uIdx][vIdx][1] + deltaV \* lutU[yIdx][uIdx][vIdx][2] - deltaU \* lutU[yIdx][uIdx][vIdx][0]) << yShift2Idx) - (deltaY << uShift2Idx) \* (lutU[yIdx][uIdx][vIdx][2] - lutU[yIdx][uIdx][vIdx][3]) + nMappingOffset) >> nMappingShift )

V = lutV[yIdx][uIdx][vIdx][0] + (((((deltaU - deltaV) \* lutV[yIdx][uIdx][vIdx][1] + deltaV \* lutV[yIdx][uIdx][vIdx][2] - deltaU \* lutV[yIdx][uIdx][vIdx][0]) << yShift2Idx ) - (deltaY << uShift2Idx) \* (lutV[yIdx][uIdx][vIdx][2] - lutV[yIdx][uIdx][vIdx][3]) + nMappingOffset) >> nMappingShift )

# Simulation results and discussions

The proposed method was implemented on top of SCE-1 anchor [3]. Two cases are tested.

## SpsPpsPeriod=0



## SpsPpsPeriod=1



# Conclusions

In this contribution, a color gamut prediction method based on asymmetric 3D LUT is proposed. Up to 384 entries are employed for the table. Simulations show that on average 8.2% (AI-10bit), 8.2% (AI-8bit), 6.3% (RA-10bit) and 6.2% (RA-8bit) luma BD rate reduction was achieved over SCE-1 use case 1 anchor, and 8.4% (AI-10bit), 8.4% (AI-8bit), 6.6% (RA-10bit) and 6.4% (RA-8bit) luma BD rate reduction over SCE-1 use case 2 anchor.

# References

1. [P.Bordes](mailto:philippe.bordes@technicolor.comp), [P.Andrivon](mailto:pierre.andrivon@technicolor.comp), [F.Hiron](mailto:franck.hiron@technicolor.comp), [P.Salmon](mailto:philippe.salmon@technicolor.comp), [R.Boitard](mailto:ronan.boitard@technicolor.comp), “SCE4: Results on 5.3-test1 and 5.3-test2”, JCTVC-O0159, October 2013, Geneva, CH.

1. [Y. He](mailto:yuwen.he@interdigital.com), [Y. Ye](mailto:yan.ye@interdigital.com), [J. Dong](mailto:jie.dong@interdigital.com), “Non-SCE4/AHG14: Combined bit-depth and color gamut conversion with 3D LUT for SHVC color gamut scalability”, JCTVC-O0161, October 2013, Geneva, CH.
2. P. Bordes, Y. Ye, E. Alshina, X. Li, S.-H. Kim, A. Duenas, K. Ugur, K. Sato, “Description of HEVC Scalable Extensions Core Experiment SCE1: Color Gamut and Bit-Depth Scalability”, JCTVC-O1101, October 2013, Geneva, CH.

# Patent rights declaration(s)

**Qualcomm 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).**