Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11
15th Meeting: Geneva, CH, 23 Oct. – 1 Nov. 2013

Document: JCTVC-O0159

Title:	**SCE4: Results on 5.3-test1 and 5.3-test2**
Status:	Input Document to JCT-VC
Purpose:	Proposal

Author(s) or Contact(s):
Philippe Bordes,
Pierre Andrivon,
Franck Hiron,
Philippe Salmon,
Ronan Boitard
975 avenue des champs blancs
CS 17616, 35576, Cesson-Sévigné Cedex, France

Tel: +33-2-99-27-32-42
Email: philippe.bordes@technicolor.com

Source: Technicolor

Abstract

This contribution reports the performance analysis of SCE4 "5.3-test1" and "5.3-test2" on Color Gamut and Bit-Depth Scalability, based on the use of 3D color Look-Up Tables (LUT) to perform inter-layer prediction.

It is reported that compared with the SCE4 anchor, for 5.3-test1 (8-bit BL, 10-bit EL) in AI configuration, the proposed method achieves an average BD rate of -12.3%, -9.9%, -16.0% for Y, U, V, respectively. For RA configuration, the proposed method achieves an average BD rate of -8.2%, -3.0%, -9.9% for Y, U, V, respectively.

It is reported that compared with the SCE4 anchor, for 5.3-test2 (10-bit BL, 10-bit EL) in AI configuration, the proposed method achieves an average BD rate of -12.2%, -9.6%, -14.9% for Y, U, V, respectively. For RA configuration, the proposed method achieves an average BD rate of -8.5%, -3.4%, -10.1% for Y, U, V, respectively.

1 Introduction - Problem Statement

Color Gamut Scalability has been identified as one requirement of Scalable Coding Extension of HEVC [1]. It allows addressing the cases the original Enhancement Layer uses a different color gamut than the Base Layer. This can be useful for instance in case of deployment of UHD services compatible with legacy HD devices: HD is using the Rec.709 [2], while UHD is likely to use some of the parameters defined in the Rec.2020 [3].

The general diagram of a scalable video encoder including a prediction tool for color differences between the base layer (BL) and enhancement layer (EL) is shown in Figure 1.

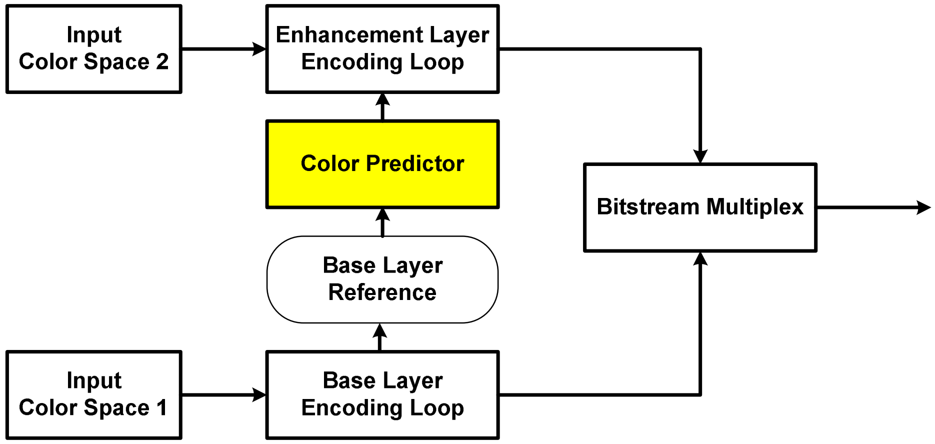

Figure 1: Color Space Scalable Encoder (courtesy from Sharp).

Basically, the role of the Color Predictor module is to predict the EL color samples from the collocated BL color samples. However, for a given pair of BL and EL video sequence, the determination of this color transfer function is not straightforward because the content creation workflows may include deterministic processing (Color Space 1 vs Color Space 2 conversion) but also non-deterministic operations, for the reasons explained below:

- The last digital cameras used in Digital Cinema (DC) allow to capture video signal with wide color and luminance range. The output raw data are Wide Color Gamut (WCG) that may be beyond DCI-P3-gamut, with potentially extended dynamic range (more than 14 f-stops). The floating point or 16 bits raw data represent much more information than what will be distributed and displayed finally.

  This trends will probably increase in the coming years, the new captors being able to capture several exposure directly for instance, and with the new MPEG AHG on Support of XYZ Color Space for Full Gamut Content Distribution that will facilitate the deployment of such content.

- Bit-depth scaling / tone mapping: the choice of the luminance range mapping versus the Reference Output Display characteristics (ODT) (ex: 16 bits vs 8 or 10 bits) is made by human operator depending on artistic intents. Figure 2 illustrates the tone mapping of the picture "Tree.hdr' [8] to address 2 different reference displays (an 8-bit and a 10-bit display).

Figure 2: grading of "Tree.hdr" [8]. Left: tone mapping for 8-bit LDR display, right: tone mapping for 10-bit HDR display (Prepared for 8-bit display).

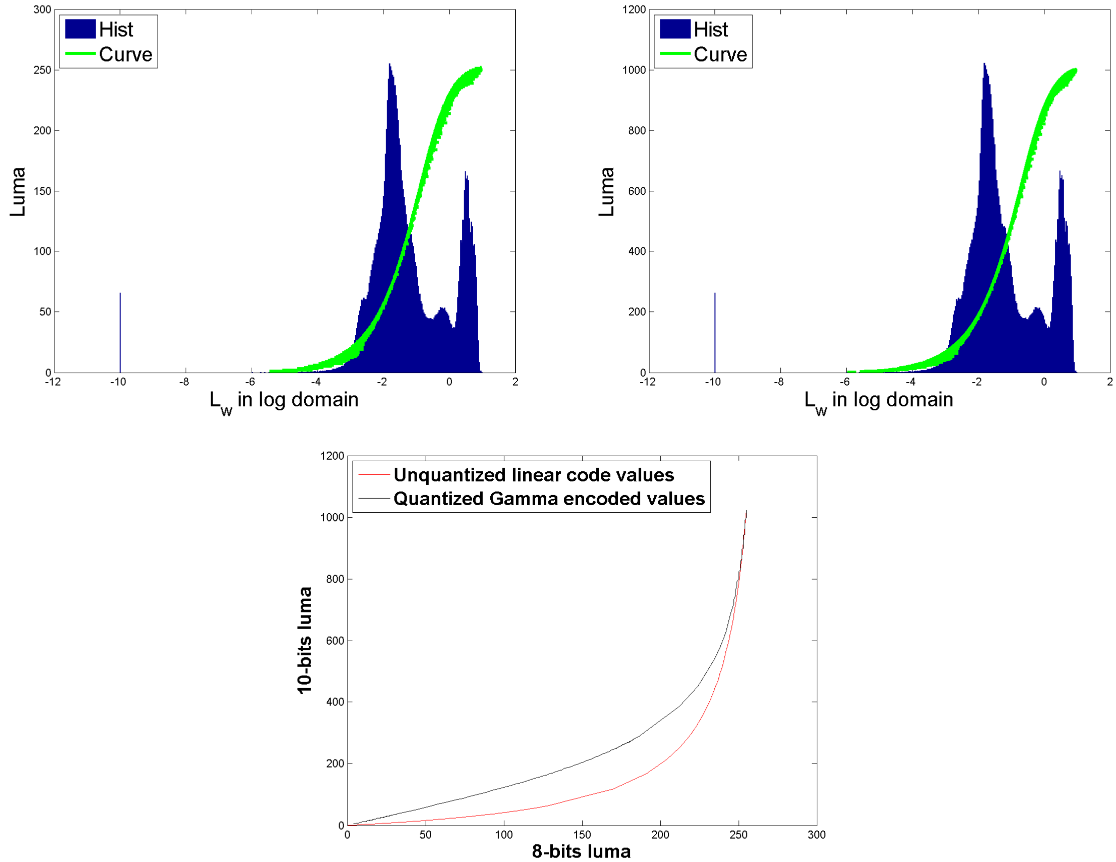

Figure 3: Tone mapping curve selected for grading picture "Tree" [8] using 8-bit LDR (up-left) and 10-bit HDR (up-right) reference displays. Bottom: relation between tone-mapped 10-bit and tone mapped 8-bit signal, w/o gamma (black/red).

The Figure 3 represents the two different sigmoids used to perform the tone mapping operation (using Reinhard et al. global TMO [4]) and the relation in between the two tone mapping functions which is clearly non-linear. This relation would be better approximated by a LUT rather than using linear based model.

- Color grading (artistic intent): color balancing is a key feature in content creation, because it traduces the artistic intent of the film director and has major impact on the final rendering. It is performed by high-skilled graphic operators (colorists), using a reference display. Then, if two targeted displays with different characteristics are used (e.g. DCI projectors for DC and Rec.709 TV for HDTV), the artistic intent may be different and the color grading may be different too. These graphic designers use special color authoring tools using 3D LUTs to represent/output their color processing operations.
- Color Space conversion: currently and pragmatically Rec.709-Rec.2020 has been identified as a probable color space conversion use case (e.g. scalable HDTV and UHDTV). This conversion is basically not linear. Besides, new video signal definitions may appear in the coming decade, fueled by the increasing display technology capabilities (e.g. OLED displays…). Some applications should require to adapt content to the end-device rendering characteristics/capabilities consequently.

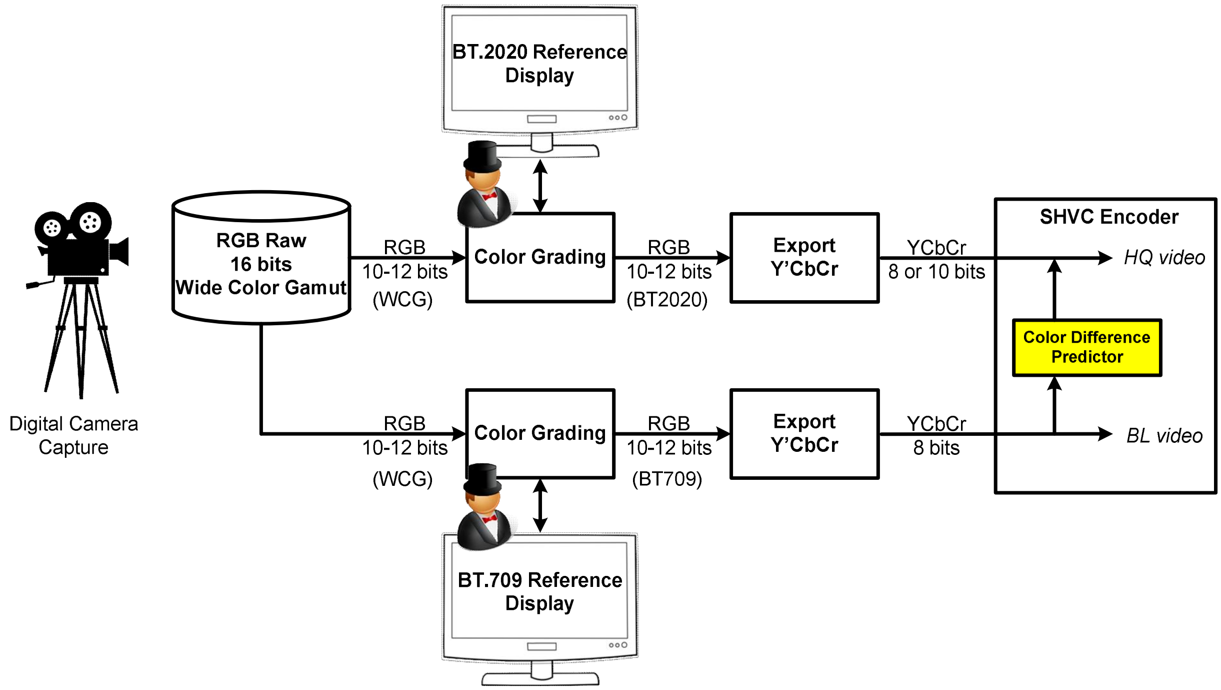

Figure 4: Hypothetical scalable HD/UHD processing workflow, inspired from (simplified) Digital Cinema workflow.

An hypothetical scalable HD/UHD processing workflow is depicted in Figure 4. A more precise definition of a digital motion picture workflow is proposed by the Academy Color Encoding System (ACES). It may be used to create content for movie theater or for physical media distribution such as DVD or Blu-ray disc typically.

Consequently, the "Color Difference Predictor" function model is highly unpredictable and may have very various shape. This justifies to describe it based on a generic and flexible model.

2 Technical description

In order to be able to address a wide range of Color Gamut Scalability (CGS) applications, without any a-priori on the "Color Predictor" model (Figure 5), CGS using 3D color Look-Up Tables (LUT) to perform inter-layer prediction has been proposed in JCTVC-M0197 [5] and JCTVC-N0168 [6].

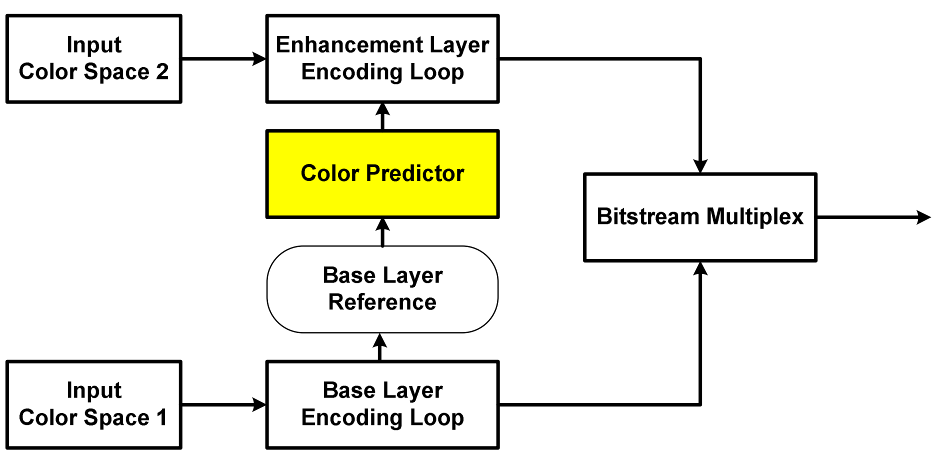

Figure 5: Principle of Color Space Scalable Encoder.

The principle of the 3D LUT is depicted in Figure 6: the 3D LUT can be considered as a sub-sampling of the 3D color space 1, where each vertex is associated with a color triplet corresponding to the color space 2 (predicted) values. For a given BL color sample (color space 1), the computation of its prediction in (EL) color space 2 is made using tetrahedral interpolation of the LUT.

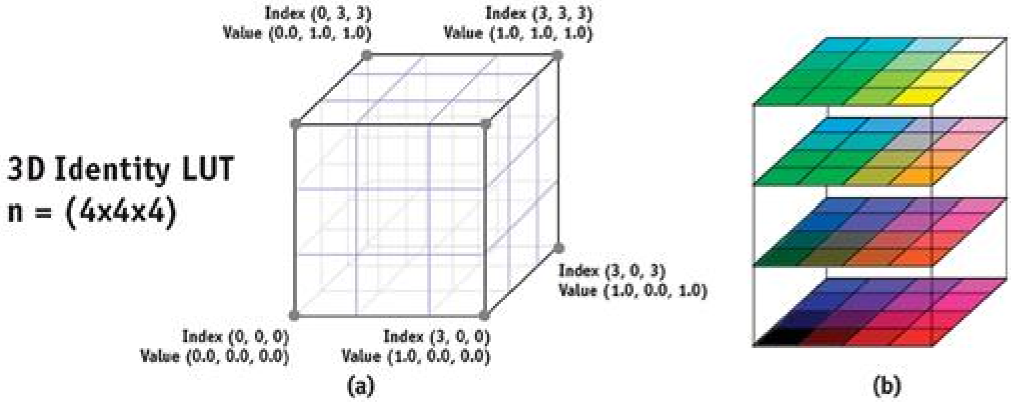

Figure 6: Principle of the 3D Color Look-Up Table (LUT).

In order to encode the 3D LUT data efficiently, each color component of a vertex is encoded with previously encoded color components of neighboring vertices. We propose also to use an octree based description of the 3D LUT in such a way the unused (or less used) 3D color space regions are encoded with coarsely lattice size as depicted in Figure 7.

At the decoder side, all the not coded vertices inside one octant are interpolated to reconstruct the full definition 3D LUT.

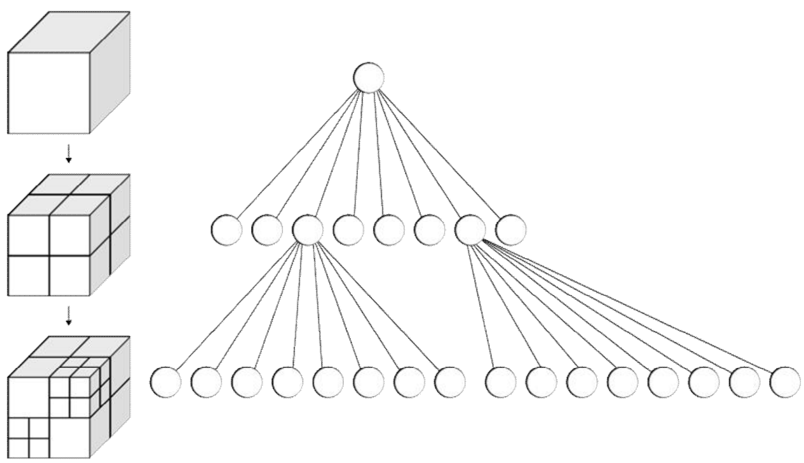

Figure 7: Octree based 3D LUT: each octant is encoded with 8 vertices at most.

This approach has several advantages:

- Many Color processing tools uses 3D LUTs to represent and save their intermediate and final color grading operations. In these cases, the 3D LUT information can be made available to the encoder easily.
- We propose that the size of the 3D LUT (number of vertices in one direction), is a parameter read in the bit-stream. In that way, the encoder may choose the best trade-off between “Color Predictor” module accuracy and encoding cost.
- At last, 3D color LUT interpolation module for color conversion is implemented in many STBs and display devices (graphics card).

The Color conversion, the bit-depth increase and the up-sampling processing order is depicted in Figure 8. We also provide simulation results for processing order as depicted in Figure 9.

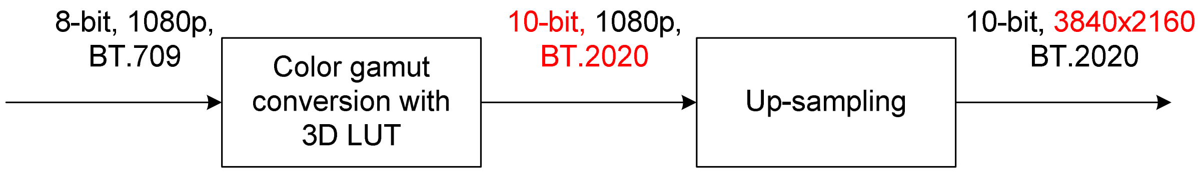

Figure 8: Color conversion first, bit-depth increase and up-sampling processing order.

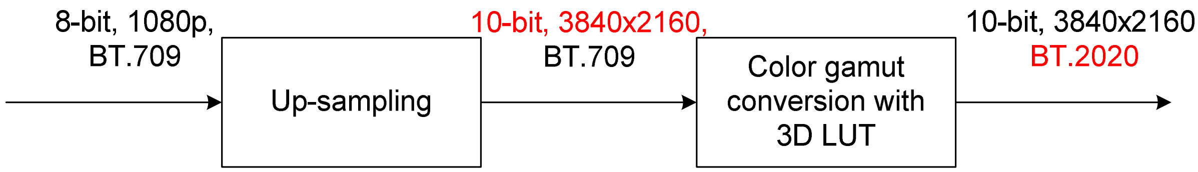

Figure 9: Up-sampling first, bit-depth increase and Color conversion processing order.

3 Test results

We used the content provided by the Ad hoc Group 14 on Color Gamut Scalability. The 3D LUTs used in our simulations have been trained offline, using uncompressed BT.709 and BT.2020 sequences and Least Square minimization method. The test conditions as described in SCE4 description [7] correspond to 2x scalability with the following contents:

- Test 1: Enhancement layer: 3840x2160 resolution, 10-bit, BT.2020 gamut / Baselayer layer: 1920x1080p, BT.709, 8-bit.
- Test 2: Enhancement layer: 3840x2160 resolution, 10-bit, BT.2020 gamut / Baselayer layer: 1920x1080p, BT.709, 10-bit.

The common SHVC test conditions (QPs) are used for AI and RA configurations, 2x scalability. The averaged results are depicted in Table 1 for processing order depicted in Figure 8. The results for processing order as depicted in Figure 9 achieves almost identical results but the encoding and decoding times are increased.

For test-1 (8-bit BL, 10-bit EL) in All Intra, the proposed method achieves an average BD rate of -12.3%, -9.9%, -16.0% for Y, U, V, respectively. For Random Access test case, the proposed method achieves an average BD rate of -8.2%, -3.0%, -9.9% for Y, U, V, respectively.

For test-2 (10-bit BL, 10-bit EL) in All Intra, the proposed method achieves an average BD rate of -12.2%, -9.6%, -14.9% for Y, U, V, respectively. For Random Access test case, the proposed method achieves an average BD rate of -8.5%, -3.4%, -10.1% for Y, U, V, respectively.

It is worthwhile to note the achieved BD rate gains compared to simulcast (Overall Test vs single layer) in 8-bit base and 10-bit base, both in AI-2x and RA-2x, are equivalent to the BD rate gains of SHM-3.0.1 in AI-2x (12.8%, 14.9%, 14.6%) and RA-2x (19.0%, 33.1%, 31.9%) obtained with classes A and B in average.

Then, the proposed method allows to encompass the color gamut dissimilarity between base layer and enhancement layer since same level of scalable coding performance are achieved as with the regular content where BL and EL video have same color gamut.

Table 1: BD-rate gains of SCE4 5.3-test1 (8-bit base) and 5.3-test2 (10-bit base) compared with SHM-3.0.1-SCE4 anchors.

	AI HEVC 2x 10-bit base			AI HEVC 2x 8-bit base		
	Y	U	V	Y	U	V
Class A+	-12.2%	-9.6%	-14.9%	-12.3%	-9.9%	-16.0%
Overall (Test vs Ref)	-12.2%	-9.6%	-14.9%	-12.3%	-9.9%	-16.0%
Overall (Test vs single layer)	11.2%	16.8%	9.8%	13.7%	18.8%	11.0%

Overall (Ref vs single layer)	26.8%	29.4%	28.9%	29.8%	32.1%	32.1%
Overall (Test EL+BL vs single EL+BL)	-27.1%	-23.5%	-28.5%	-25.4%	-22.3%	-27.8%
EL only (Test vs Ref)	-22.3%	-19.6%	-24.5%	-22.7%	-20.2%	-25.8%
Enc Time[%]		97.9%			98.0%	
Dec Time[%]		95.8%			90.1%	

	RA HEVC 2x 10-bit base			**RA HEVC 2x 8-bit base**		
	Y	U	V	Y	U	V
Class A+	-8.5%	-3.4%	-10.1%	-8.2%	-3.0%	-9.9%
Overall (Test vs Ref)	-8.5%	-3.4%	-10.1%	-8.2%	-3.0%	-9.9%
Overall (Test vs single layer)	20.9%	31.0%	19.3%	22.7%	32.5%	20.3%
Overall (Ref vs single layer)	32.3%	35.5%	33.0%	33.8%	36.4%	33.9%
Overall (Test EL+BL vs single EL+BL)	-19.4%	-12.1%	-19.8%	-18.3%	-11.5%	-19.5%
EL only (Test vs Ref)	-15.2%	-9.7%	-16.2%	-15.0%	-9.4%	-16.0%
Enc Time[%]		98.7%			98.9%	
Dec Time[%]		101.9%			91.8%	

4 References

[1] Ajay Luthra, Jens-Rainer Ohm, Jörn Ostermann, " Requirements of the scalable enhancement of HEVC," WG11 Requirements and Video, ISO/IEC JTC1/SC29/WG11 N12783, May 2012, Geneva, Switzerland.

[2] ITU-R Recommendation BT.709 "Parameter values for the HDTV standards for production and international programme exchange" Dec. 2010.

[3] ITU-R Recommendation BT.2020 "Parameter values for UHDTV systems for production and international programme exchange" April 2012.

[4] Reinhard, E., Stark, M., Shirley, P., & Ferwerda, J. (2002). Photographic tone reproduction for digital images. *ACM Transactions on Graphics*, *21*(3).

[5] Philippe Bordes, Pierre Andrivon, Roshanak Zakizadeh, « AHG14: Color Gamut Scalable Video Coding using 3D LUT," JCTVC-M0197, 13th Meeting: Incheon, KR, 18–26 Apr. 2013.

[6] Philippe Bordes, Pierre Andrivon, Patrick Lopez, Franck Hiron, " AHG14: Color Gamut Scalable Video Coding using 3D LUT: New Results," JCTVC-N0168, 14th Meeting: Vienna, AT, 25 July – 2 Aug. 2013.

[7] Andrew Segall, Philippe Bordes, Cheung Auyeung, Xiang Li, Elena Alshina, Alberto Duenas, "Description of Core Experiment SCE4: Color Gamut and Bit-Depth Scalability", JCTVC-N1104, 14th Meeting: Vienna, Austria, AT, July 29 – Aug 2, 2013.

[8] Greg Ward, "Tree.hdr" http://www.anyhere.com/gward/hdrenc/pages/img/Tree_oAC1.hdr.

5 Patent rights declaration

Technicolor 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).

6 Annex: Specification text for the proposed color gamut scalability

7.3.2.3 Picture parameter set RBSP syntax

pic_parameter_set_rbsp( ) {	**Descriptor**
…	
use_color_prediction_flag	u(1)
if (use_color_prediction_flag)	
3D_ LUT_ color_data ( )	
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

use_color_prediction_flag equal to 1 specifies that color prediction process is applied to the decoded reference layer picture samples. use_color_prediction_flag equal to 0 specifies that color prediction process is not applied to the decoded reference layer picture samples.

7.3.2.4 Color LUT parameters syntax

3D_ LUT_ color_data ( ) {	**Descriptor**
nbp_code	u(3)
lut_bit_depth_minus8	u(4)
coding_octant(0, 0, 0, 0)	
}	

coding_octant (layer, y,u,v) {	**Descriptor**
for(i = 0; i < 8 ; i++) {	
n = getVertex(y, u, v, i)	
if (!coded_flag[n]) {	
encoded_vertex_flag[i]	u(1)
if (encoded_vertex_flag[i]) {	
resY[i]	ue(v)
resU[i]	ue(v)
resV[i]	ue(v)
}	
coded_flag[n] = true	
}	
}	
if (layer < nbp_code) {	
split_octant_flag	u(1)
if (split_octant_flag) {	
for(i = 0; i < 8 ; i++) {	
coding_octant (layer+1, y+dy[i],u+du[i],v+dv[i])	
}	
}	
}	
}	

7.3.2.4 Color LUT parameters semantics

nbp_code indicates the three dimensional LUTs size nbp. nbp is equal to 1+(1<<(nbp_code-1)).

lut_bit_depth_minus8 specifies the bit depth of the LUTs samples LutBitDepth as follows:

LutBitDepth = 8 + lut_bit_depth_minus8 (7 4)

encoded_vertex_flag[i] equal to 1 specifies that the residuals for the i^{th} vertex of the octant(layer,y,u,v) are present. encoded_vertex_flag[i] equal to 0 specifies that the residuals for the i^{th} vertex of the octant(layer,y,u,v) are not present and are inferred to be equal to zero.

resY[i], resU[i], resV[i] are the difference of the luma, chroma1 and chroma2 components of the vertex (y+dy[i], u+du[i], v+dv[i]) with the predicted luma, chroma1 and chroma2 component values for this vertex respectively. The derivation of the predicted component values for this vertex and the decoding of the color LUTs is specified in the decoding process for color LUT in 8.4.

split_octant_flag specifies whether an octant is split into four octants with half size in all directions for the purpose of vertices residuals octant coding.

8.4 Decoding process for color LUT

Inputs to this process are:

- The residuals values resY[i], resU[i], resV[i] of octant (layer, y, u, v).

Outputs to this process are :

- The three arrays $LUT_Y[i_L][i_{C1}][i_{C2}]$, $LUT_{C1}[i_L][i_{C1}][i_{C2}]$, $LUT_{C2}[i_L][i_{C1}][i_{C2}]$. The array indices i_L, i_{C1}, i_{C2} specify the reference layer picture color space components sub-sampled ranging from 0 to (1+1<<(nbp_code-1)).

The decoding of the residual vertices of an octant(layer, y,u,v) is a recursive process. Each octant is composed of 8 vertices associated with a flag (encoded_vertex_flag[i]) indicating whether the residual components values are encoded or all inferred to be zero. The component values are reconstructed by adding the residuals to the prediction $pred_X[i]_{X=Y,C1,C2}$ of the components values of the i^{th} vertex as follows:

LUT_X[y+dy[i]][u+du[i]][v+dv[i]] = res_X[i] + $pred_X$[i] (7 5)

where the values of dy[i], du[i], dv[i] are given in Table 2.

Once reconstructed, a vertex n is marked as reconstructed (coded_flag[n]=true). The predicted component $pred_X[i]$ of the vertex (y+dy[i], u+du[i], v+dv[i]) with X=Y, C1, C2, is obtained using trilinear interpolation of the neighboring vertices of the upper layer as follows:

- If layer equal to 0, then $pred_X[i]$ is given by Table 3.
- Otherwise the value of $pred_X[i]$ is obtained as follows:

  $pred_X[i]$ = (Ai,0 + Ai,1 + Ai,2 + Ai,3 + Ai,4 + Ai,5 + Ai,6 + Ai,7 + (1<<(shift3-1))) >> shift3

  where the values of $(A_{i,k})_{i=0,7\ k=0,7}$ and shift3 are derived as follows:

  shift3 = 3 * (nbp_code – layer)

  $A_{i,k} = w_{i,k}$ * LUT_X[yr+$dy_{layer-1}$[k]][ur+$du_{layer-1}$[k]][vr+$dv_{layer-1}$[k]]

  Where the values of $w_{i,k}$, yr, ur, vr are derived as follows:

  yr = ((y >> (nbp_code - layer)) << (nbp_code - layer))

  ur = ((u >> (nbp_code - layer)) << (nbp_code - layer))

  vr = ((v >> (nbp_code - layer)) << (nbp_code - layer))

  if (yr == (nbp-1)) yr = yr - ((nbp-1) >> (layer-1))

  if (ur == (nbp-1)) ur = ur - ((nbp-1) >> (layer-1))

  if (vr == (nbp-1)) vr = vr - ((nbp-1) >> (layer-1))

  $w_{i,k}$ = sy(i,k) * su(i,k) * sv(i,k)

  where the values of sy(i,k), su(i,k), sv(i,k) are derived as follows:

  sy(i,k) = ((nbp-1) >> (layer-1)) - ABS((yr+$dy_{layer-1}$[k]) – (y+dy_{layer}[i]))

  su(i,k) = ((nbp-1) >> (layer-1)) - ABS((ur+$du_{layer-1}$[k]) – (u+du_{layer}[i]))

  sv(i,k) = ((nbp-1) >> (layer-1)) - ABS((vr+$dv_{layer-1}$[k]) – (v+dv_{layer}[i]))

Table 2: values dy[i],du[i] and dv[i] in function of index I, for vertices belonging to layer = layer_id.

i	dy_{layer_id}[i]	du_{layer_id}[i]	dv_{layer_id}[i]
0	0	0	0
1	0	0	(nbp-1) >> layer_id
2	0	(nbp-1) >> layer_id	0
3	0	(nbp-1) >> layer_id	(nbp-1) >> layer_id
4	(nbp-1) >> layer_id	0	0
5	(nbp-1) >> layer_id	0	(nbp-1) >> layer_id
6	(nbp-1) >> layer_id	(nbp-1) >> layer_id	0
7	(nbp-1) >> layer_id	(nbp-1) >> layer_id	(nbp-1) >> layer_id

Table 3: prediction values used for the 3 components of the 8 vertices belonging to the first layer (layer_id=0)(max=(1<< (bit_depth_lut_minus8+8)-1)).

i	$pred_Y$[i]	$pred_{C1}$[i]	$pred_{C2}$[i]
0	0	0	0
1	0	0	max
2	0	max	0

3	0	max	max
4	max	0	0
5	max	0	max
6	max	max	0
7	max	max	max

H.8.1.4.1.1 Color prediction process of luma and chroma sample values

- Input to this process is the reference luma sample array $rlPicSample_L$ and reference chroma sample arrays $rlPicSample_{C1}$ and $rlPicSample_{C2}$.
- Output of this process is the predicted color sample array $predColorSample_X$, with X=L, C1 or C2.

The variables shift, shift_out, i_X, iEr_X with (X=L,C1,C2) are derived as follows:

$shift = 9 - nbpCode + (LutBitDepth - 8)$

$shift\_out = shift - BitDepthEL_X + LutBitDepth$

$i_L = rlPicSample_L[xP_L,yP_L] >> shift$

$i_{C1} = rlPicSample_{C1}[xP_C,yP_C] >> shift$

$i_{C2} = rlPicSample_{C2}[xP_C,yP_C] >> shift$

$iEr_L = rlPicSample_L[xP_L,yP_L] - (i_L << shift)$

$iEr_{C1} = rlPicSample_{C1}[xP_L,yP_L] - (i_{C1} << shift)$

$iEr_{C2} = rlPicSample_{C2}[xP_L,yP_L] - (i_{C2} << shift)$

The sample value tempArray[n] with n = 0..7, is derived as follows:

$tempArray_X[0] = LUT_X[i_L][i_{C1}][i_{C2}]$

$tempArray_X[1] = LUT_X[i_L][i_{C1}][i_{C2}+1]$

$tempArray_X[2] = LUT_X[i_L][i_{C1}+1][i_{C2}]$

$tempArray_X[3] = LUT_X[i_L][i_{C1}+1][i_{C2}+1]$

$tempArray_X[4] = LUT_X[i_L+1][i_{C1}][i_{C2}]$

$tempArray_X[5] = LUT_X[i_L+1][i_{C1}][i_{C2}+1]$

$tempArray_X[6] = LUT_X[i_L+1][i_{C1}+1][i_{C2}]$

$tempArray_X[7] = LUT_X[i_L+1][i_{C1}+1][i_{C2}+1]$

- If ($iEr_L >= iEr_{C1}$) and ($iEr_{C1} >= iEr_{C2}$) then the interpolated sample value intSample is derived as follows:

$intSample = (tempArray_X[0] << shift) +$

$iEr_L * (tempArray_X[4] - tempArray_X[0]) +$

$iEr_{C1} *(tempArray_X[6] - tempArray_X[4])+$

$iEr_{C2} *(tempArray_X[7] - tempArray_X[6])$

- Otherwise if ($iEr_L > iEr_{C2}$) and ($iEr_{C2} >= iEr_{C1}$) then the interpolated sample value intSample is derived as follows:

$intSample = (tempArray_X[0] << shift) +$

$iEr_L * (tempArray_X[4] - tempArray_X[0]) +$

$iEr_{C1} *(tempArray_X[7] - tempArray_X[5])+$

iEr_{C2} *($tempArray_X[5]$ - $tempArray_X[4]$)

- Otherwise if (iEr_{C2} >= iEr_L) and (iEr_L > iEr_{C1}) then the interpolated sample value intSample is derived as follows:

  intSample = ($tempArray_X[0]$ << shift) +

  iEr_L * ($tempArray_X[5]$ - $tempArray_X[1]$) +

  iEr_{C1} *($tempArray_X[7]$ - $tempArray_X[5]$)+

  iEr_{C2} *($tempArray_X[1]$ - $tempArray_X[0]$)

- Otherwise if (iEr_{C1} > iEr_L) and (iEr_L >= iEr_{C2}) then the interpolated sample value intSample is derived as follows:

  intSample = ($tempArray_X[0]$ << shift) +

  iEr_L * ($tempArray_X[6]$ - $tempArray_X[2]$) +

  iEr_{C1} *($tempArray_X[2]$ - $tempArray_X[0]$)+

  iEr_{C2} *($tempArray_X[7]$ - $tempArray_X[6]$)

- Otherwise if (iEr_{C1} > iEr_{C2}) and (iEr_{C2} > iEr_L) then the interpolated sample value intSample is derived as follows:

  intSample = ($tempArray_X[0]$ << shift) +

  iEr_L * ($tempArray_X[7]$ - $tempArray_X[3]$) +

  iEr_{C1} *($tempArray_X[2]$ - $tempArray_X[0]$)+

  iEr_{C2} *($tempArray_X[3]$ - $tempArray_X[2]$)

- Otherwise if (iEr_{C2} >= iEr_{C1}) and (iEr_{C1} >= iEr_L) then the interpolated sample value intSample is derived as follows:

  intSample = ($tempArray_X[0]$ << shift) +

  iEr_L * ($tempArray_X[7]$ - $tempArray_X[3]$) +

  iEr_{C1} *($tempArray_X[3]$ - $tempArray_X[1]$)+

  iEr_{C2} *($tempArray_X[1]$ - $tempArray_X[0]$)

- If X=L, then the predicted color gamut sample is derived as follows:

  $predColorSample_L[xP_L,yP_L]$ = (intSample + (1 << (shift_out-1))) >> shift_out

- Otherwise, if X=C, then the predicted color sample is derived as follows:

  $predColorSample_C[xP_C,yP_C]$ = (intSample + (1 << (shift_out-1))) >> shift_out