INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 MPEG2015/M37400
October 2015, Geneva, Switzerland

Source **Technicolor**
Status **Proposal**
Title **Test model draft for SDR backward compatibility from CE2.2.a proposal**
Author E. François, F. Galpin, S. Lasserre, F. Le Leannec, T. Poirier

Abstract

This contribution presents syntax and text modifications relative to current HEVC standard that correspond to the processes proposed in CE2.2.a. In particular, it describes the mapping process and color reconstruction process.

1. Introduction

The following provides the syntax and semantics for the HDR reconstruction process applied in CE2.2.a after decoding. Note that the operations applying after 4:2:0-to-4:4:4 conversion are based on conventional processes. The operations preceding 4:2:0-to-4:4:4 conversion correspond to HDR samples reconstruction from the decoded samples. The HDR samples are generated in YCbCr, 10bit PQ, 4:2:0, CL format. This part correspond to the provided text specification.

The decoded signal is an SDR content that can be used as is by SDR rendering devices. The HDR signal after applying the specified reconstruction process is in YCbCr PQ CL 4:2:0 format.

The process is made of two steps:

- Luma and chroma mapping, based on two 1D-look-up-tables or piece-wise linear model (2 versions of the spec are given below, for each one of these models).
- Color reconstruction process to generate the final output YCbCr PQ CL 4:2:0 content.

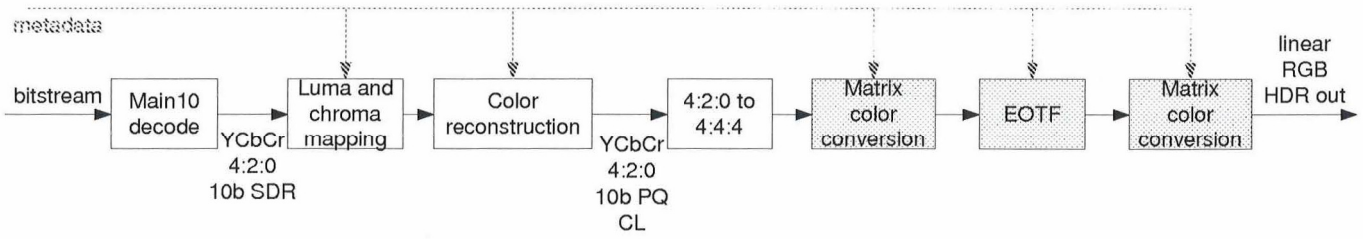


Figure 1: block diagram of the CE2.2.1 HDR reconstruction process.

BNSDOCID: <XP_____30065768A__I_>

Spec1: Text specification using 1D-look-up-tables for mapping process

Syntax changes

In 7.3.2.2.1 (General sequence parameter set RBSP syntax), replace the syntax table with the following:

seq_parameter_set_rbsp( ) {	Descriptor
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data( )	
}	
amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)

BNSDOCID: <XP_____30065768A__I_>

pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)
log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
strong_intra_smoothing_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_present_flag	u(1)
if(sps_extension_present_flag) {	
sps_range_extension_flag	u(1)
sps_multilayer_extension_flag	u(1)
sps_3d_extension_flag	u(1)
sps_scc_extension_flag	u(1)
sps_hdr_extension_flag	u(1)
sps_extension_3bits	u(3)
}	
if(sps_range_extension_flag)	
sps_range_extension( )	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension( ) /* specified in Annex **Erreur ! Source du renvoi introuvable.** */	
if(sps_3d_extension_flag)	
sps_3d_extension( ) /* specified in Annex **Erreur ! Source du renvoi introuvable.**/	
if(sps_scc_extension_flag)	
sps_scc_extension( )	
if(sps_hdr_extension_flag)	
sps_hdr_extension( )	
if(sps_extension_3bits)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

BNSDOCID: <XP_____30065768A__I_>

BNSDOCID: <XP_____30065768A__I_>

Add new section 7.3.2.2.6 Sequence parameter HDR coding extension syntax

sps_hdr_extension( ) {	Descriptor
hdr_recon_video_signal_info_present_flag	u(1)
if(hdr_recon_video_signal_info_present_flag) {	
hdr_recon_full_range_flag	u(8)
hdr_recon_colour_primaries	u(8)
hdr_recon_transfer_characteristics	u(8)
hdr_recon_matrix_coeffs	u(8)
hdr_max_display_mastering_luminance	u(32)
}	
for(i=0; i<3; i++)	
hdr_recon_correction_coef[c]	u(10)
}	

In 7.3.2.3.1 (General picture parameter set RBSP syntax), replace the syntax table with the following:

pic_parameter_set_rbsp( ) {	Descriptor
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
dependent_slice_segments_enabled_flag	u(1)
output_flag_present_flag	u(1)
num_extra_slice_header_bits	u(3)
sign_data_hiding_enabled_flag	u(1)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
transform_skip_enabled_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
if(cu_qp_delta_enabled_flag)	
diff_cu_qp_delta_depth	ue(v)
pps_cb_qp_offset	se(v)
pps_cr_qp_offset	se(v)
pps_slice_chroma_qp_offsets_present_flag	u(1)
weighted_pred_flag	u(1)
weighted_bipred_flag	u(1)
transquant_bypass_enabled_flag	u(1)
tiles_enabled_flag	u(1)
entropy_coding_sync_enabled_flag	u(1)
if(tiles_enabled_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)

BNSDOCID: <XP_____30065768A__I_>

for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
pps_loop_filter_across_slices_enabled_flag	u(1)
deblocking_filter_control_present_flag	u(1)
if(deblocking_filter_control_present_flag) {	
deblocking_filter_override_enabled_flag	u(1)
pps_deblocking_filter_disabled_flag	u(1)
if(!pps_deblocking_filter_disabled_flag) {	
pps_beta_offset_div2	se(v)
pps_tc_offset_div2	se(v)
}	
}	
pps_scaling_list_data_present_flag	u(1)
if(pps_scaling_list_data_present_flag)	
scaling_list_data( )	
lists_modification_present_flag	u(1)
log2_parallel_merge_level_minus2	ue(v)
slice_segment_header_extension_present_flag	u(1)
pps_extension_present_flag	u(1)
if(pps_extension_present_flag) {	
pps_range_extension_flag	u(1)
pps_multilayer_extension_flag	u(1)
pps_3d_extension_flag	u(1)
pps_scc_extension_flag	u(1)
pps_hdr_extension_flag	u(1)
pps_extension_3bits	u(3)
}	
if(pps_range_extension_flag)	
pps_range_extension( )	
if(pps_multilayer_extension_flag)	
pps_multilayer_extension( ) /* specified in Annex F */	
if(pps_3d_extension_flag)	
pps_3d_extension( ) /* specified in Annex I */	
if(pps_scc_extension_flag)	
pps_scc_extension( )	
if(pps_hdr_extension_flag)	
pps_hdr_extension( )	
if(pps_extension_3bits)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

BNSDOCID: <XP_____30065768A__I_>

Add new section 7.3.2.3.9 **Picture parameter HDR coding extension syntax**

pps_hdr_extension( ) {	**Descriptor**
hdr_colour_recon_lut_size	u(12)
for(i = 0; i < colour_correction_lut_size ; i++)	
hdr_colour_recon_lut_value[i]	u(16)
hdr_luma_mapping_lut_size	u(12)
for(i = 0; i < mappedY_lut_size ; i++)	
hdr_luma_mapping_lut_value[i]	u(10)
}	

BNSDOCID: <XP_____30065768A__I_>

Semantics changes

In 7.4.3.2.1 (General sequence parameter set RBSP semantics), add the following semantics:

sps_hdr_extension_flag equal to 1 specifies that the sps_hdr_extension( ) syntax structure is present in the SPS RBSP syntax structure. sps_hdr_extension_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of sps_hdr_extension_flag is inferred to be equal to 0.

sps_extension_3bits equal to 0 specifies that no sps_extension_data_flag syntax elements are present in the SPS RBSP syntax structure. When present, sps_extension_3bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of sps_extension_3bits not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of sps_extension_3bits to be not equal to 0 and shall ignore all sps_extension_data_flag syntax elements in an SPS NAL unit. When not present, the value of sps_extension_3bits is inferred to be equal to 0.

Add a new semantics section 7.4.3.2.6 (Sequence parameter set HDR extension semantic)

hdr_recon_video_signal_info_present_flag equal to 1 specifies that syntax elements hdr_recon_full_range_flag, hdr_recon_primaries, hdr_recon_transfer_function and hdr_recon_matrix_coefficients are present, hdr_recon_video_signal_info_present_flag equal to 0 specifies that syntax elements hdr_recon_full_range_flag, hdr_recon_primaries, hdr_recon_transfer_function and hdr_recon_matrix_coefficients are not present.

hdr_recon_full_range_flag has the same semantics as specified in clause E.3.1 for the video_full_range_flag syntax element, except that hdr_recon_full_range_flag specifies the colour space of the reconstructed HDR picture, rather than the colour space used for the CLVS. When not present, the value of hdr_recon_full_range_flag is inferred to be equal to the value of video_full_range_flag.

hdr_recon_colour_primaries has the same semantics as specified in clause E.3.1 for the colour_primaries syntax element, except that hdr_recon_colour_primaries specifies the colour space of the reconstructed HDR picture, rather than the colour space used for the CLVS. When not present, the value of hdr_recon_colour_primaries is inferred to be equal to the value of colour_primaries.

hdr_recon_transfer_characteristics has the same semantics as specified in clause E.3.1 for the transfer_characteristics syntax element, except that hdr_recon_transfer_characteristics specifies the colour space of the reconstructed HDR picture, rather than the colour space used for the CLVS. When not present, the value of hdr_recon_transfer_characteristics is inferred to be equal to the value of transfer_characteristics.

hdr_recon_matrix_coeffs has the same semantics as specified in clause E.3.1 for the matrix_coeffs syntax element, except that hdr_recon_matrix_coeffs specifies the colour space of the reconstructed HDR picture, rather than the colour space used for the CLVS. When not present, the value of hdr_recon_matrix_coeffs is inferred to be equal to the value of matrix_coeffs.

hdr_max_display_mastering_luminance specifies the nominal maximum display luminance of the mastering display in units of 0.0001 candelas per square metre.

The value hdrReconRenormalizing is derived as follows:

$$\text{hdrReconRenormalizing} = \text{Ceil}(\text{sqrt}(\text{hdr\_max\_display\_mastering\_luminance} * 256 / 10000)) \qquad (8\text{-xx})$$

Add a new semantics section 7.4.3.3.8 (Picture parameter set HDR extension semantics)

hdr_colour_recon_lut_size specifies the size of the HDR colour reconstruction look-up-table.

hdr_colour_recon_lut_value[i] specifies the i-th value of the HDR colour reconstruction lookup table.

hdr_luma_mapping_lut_size specifies the size of the HDR luma mapping look-up-table.

hdr_luma_mapping_lut_value[i] specifies the i-th value of the HDR luma mapping lookup table.

Add the following text at the end of 8.1.3 (Decoding process for a coded picture with nuh_layer_id equal to 0)

At the end of the clause, add item 5 as follows:

5. When sps_hdr_extension_flag is equal to 1, the HDR reconstruction process specified in clause 8.8 is applied.

BNSDOCID: <XP_____30065768A__I_>

Add the following new section 8.8 (HDR reconstruction process)

8.8 HDR reconstruction process

8.8.1 General derivation process for the reconstructed HDR picture

This clause specifies the application of the HDR reconstruction process. The HDR reconstruction process comprises two steps, namely HDR mapping process, and when ChromaArrayType is not equal to 0, HDR colour reconstruction process.

The process is applied as specified by the following ordered steps:

1. The HDR mapping process applies as follows:
   - The HDR mapping process as specified in clause 8.8.2 is invoked with the reconstructed picture sample array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} after HDR mapping as outputs.
   - The array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} are assigned to the array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.
2. When ChromaArrayType is not equal to 0, the HDR colour reconstruction process applies as follows:
   - The HDR colour reconstruction process as specified in clause 8.8.3 is invoked with the modified reconstructed picture sample arrays S_L, S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample arrays S'_L, S'_{Cb} and S'_{Cr} after colour reconstruction as outputs.
   - The arrays S'_L, S'_{Cb} and S'_{Cr} are assigned to the arrays S_L, S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.

8.8.2 HDR mapping process

Inputs to this process are the reconstructed picture sample array prior to HDR mapping process $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

Outputs of this process are the modified reconstructed picture sample array after HDR mapping process $mappedPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $mappedPicture_{Cb}$ and $mappedPicture_{Cr}$.

The mapped luma sample $mappedPicture_L[xP][yP]$ with $xP = 0..PicWidthInSamplesY - 1$, $yP = 0..PicHeightInSamplesY - 1$ is derived by invoking the HDR mapping process of luma sample values as specified in clause 8.8.2.1 with luma sample location (xP, yP) and sample array $recPicture_L$ as inputs.

The mapped chroma sample $mappedPicture_{Cb}[xC][yC]$ with $xC = 0..PicWidthInSamplesC - 1$, $yC = 0..PicHeightInSamplesC - 1$ is derived by invoking the HDR mapping process of chroma sample values as specified in clause 8.8.2.2 with chroma sample location (xC, yC), sample arrays $recPicture_L$ and $recPicture_{Cb}$ as inputs.

The mapped chroma sample $mappedPicture_{Cr}[xC][yC]$ with $xC = 0..PicWidthInSamplesC - 1$, $yC = 0..PicHeightInSamplesC - 1$ is derived by invoking the HDR mapping process of chroma sample values as specified in clause 8.8.2.2 with chroma sample location (xC, yC), sample arrays $recPicture_L$ and $recPicture_{Cr}$ as inputs.

8.8.2.1 HDR mapping process of luma sample values

Inputs to this process are a luma location (xP, yP) specifying the luma sample location relative to the top-left luma sample of the current picture and a luma sample array $inPicture_L$.

Outputs of this process is a HDR mapped luma sample value mappedLumaSample.

The value of mappedLumaSample is derived by applying the following ordered steps:

1. The array mappingLut is derived as by invoking clause 8.8.2.3 with hdr_luma_mapping_lut_size and the array hdr_luma_mapping_lut_value as inputs and mappingLut as output.
2. The value of mappedLumaSample is derived as follows:

$$mappedLumaSample = mappingLut[inPicture_L[xP][yP]] \quad (8\text{-}xx)$$

8.8.2.2 HDR mapping process of chroma sample values

Inputs to this process are a luma location (xP_C, yP_C) specifying the chroma sample location relative to the top-left chroma sample of the current picture, a luma sample array $inPicture_L$ and a chroma sample array $inPicture_C$.

BNSDOCID: <XP_____30065768A__I_>

Output of this process is a hdr mapped chroma sample value mapChrSample.

The value of mapChrSample is derived by applying the following ordered steps:

1. The value of Ycoloc is derived as follows:

$$Ycoloc = inPicture_L[\ SubWidthC * xP_C\ ][\ SubHeightC * yP_C\ ] \quad (8\text{-}xx)$$

2. The array colorReconLut is derived as by invoking clause 8.8.2.3 with hdr_luma_colour_recon_lut_size and the array hdr_colour_recon_lut_value as inputs and correctionLut as output.

3. The value of chrScaling, bitShift, offset0, offset1 and offset2 are derived as follows:

$$chrScaling = (\ hdrReconRenormalizing * colorReconLut[\ Ycoloc\ ] + 128\ ) >> 8 \quad (8\text{-}xx)$$
$$offset0 = 1 << (\ BitDepthC - 1\ ) \quad (8\text{-}xx)$$
$$bitShift = 12 \quad (8\text{-}xx)$$
$$offset1 = 1 << (\ bitShift - 1\ ) \quad (8\text{-}xx)$$
$$offset2 = 1 << 14 \quad (8\text{-}xx)$$

4. The value of mapChrSample is derived as follows:

$$mapChrSample = (\ chrScaling * (\ inPicture_C[\ xP_C\ ][\ yP_C\ ] - offset0\ ) + offset1\ ) >> bitShift + offset2 \quad (8\text{-}xx)$$

5. The value of mapChrSample is modified as follows:

$$mapChrSample = clip3(\ 0, (\ 1 << 16\ ) - 1, mapChrSample\ ) \quad (8\text{-}xx)$$

8.8.2.3 Mapping Look-up-table derivation

Inputs to this process are an value in_lut_size and an array in_lut_value[i], i=0..(in_lut_size – 1)

Output of this process is an array out_lut_value[i], i=0..(in_lut_size – 1).

For i=0..(in_lut_size – 1), the value out_lut_value[i] is derived as follows:

$$out\_lut\_value[\ i\ ] = in\_lut\_value[\ i\ ] \quad (8\text{-}xx)$$

8.8.3 HDR colour reconstruction process

Inputs to this process are the modified reconstructed picture sample arrays after the HDR mapping process $mapPicture_L$, $mapPicture_{Cb}$ and $mapPicture_{Cr}$.

Outputs of this process are the modified reconstructed picture sample arrays after HDR colour reconstruction process $corrPicture_{Cb}$ and $corrPicture_{Cr}$.

The colour corrected chroma samples $corrPicture_{Cb}[\ xC\ ][\ yC\ ]$ and $corrPicture_{Cr}[\ xC\ ][\ yC\ ]$, with xC = 0..PicWidthInSamplesC − 1, yC = 0..PicHeightInSamplesC – 1, are derived by applying the following ordered steps.

1. The value of Ycoloc is derived as follows:

$$Ycoloc = mapPicture_L[\ SubWidthC * xP_C\ ][\ SubHeightC * yP_C\ ] \quad (8\text{-}xx)$$

2. The values of U1 and V1 are derived as follows:

$$U1 = mapPicture_{Cb}[\ xC\ ][\ yC\ ] - (\ 1 << 14\ )$$
$$V1 = mapPicture_{Cr}[\ xC\ ][\ yC\ ] - (\ 1 << 14\ ) \quad (8\text{-}xx)$$

3. Chroma samples conversion is applied by invoking section 8.8.3.1 with U1 and V1 as inputs, R and B as outputs.

4. Color samples mapping process is applied by invoking section 8.8.3.2 with R, B and Ycoloc as inputs, R' and B' as outputs.

5. The conversion to C0C1 constant luminance samples is applied by invoking section 8.8.4.3 with R', B' and Ycoloc as inputs, $corrPicture_{Cb}[\ xC\ ][\ yC\ ]$ and $corrPicture_{Cr}[\ xC\ ][\ yC\ ]$ as outputs.

8.8.3.1 Chroma samples conversion

Inputs to this process are chroma sample values U and V.

Output of this process are converted values R and B.

BNSDOCID: <XP_____30065768A__I_>

The conversion process of samples U, V into samples R, B applies as follows.

1. The array LutTtoS is derived by invoking section 8.8.5.1.
2. The array LutTtoUVfactor is derived by invoking section 8.8.5.2.
3. The parameters c1, c2, and c3 are set equal to hdr_recon_correction_coef[0], hdr_recon_correction_coef[1], and hdr_recon_correction_coef[2], respectively.
4. The parameters mR and mB are derived as indicated in Table xx.

Table xx – Derivation of mR and mB from hdr_input_colour_primaries.

hdr_recon_colour_primaries	mR	mB
1	25802	30403
9	24160	30825

5. The values of U2, V2 and UV are derived as follows:

$$U2 = (U * U + 128) >> 8$$
$$V2 = (V * V + 128) >> 8$$
$$UV = (U * V + 128) >> 8 \quad (8\text{-xx})$$

6. The value of T is derived as follows:

$$T = \max(0, ((c1 * U2 + c2 * V2 + c3 * UV) >> 12)) \quad (8\text{-xx})$$

7. The following process applies to derive the value Y:
   - If T is greater than (1 << 10), the following applies:

$$Y = 0 \quad (8\text{-xx})$$

$$U = (U * LutTtoUVfactor[T] + (1 << 14)) >> 15 \quad (8\text{-xx})$$
$$V = (V * LutTtoUVfactor[T] + (1 << 14)) >> 15 \quad (8\text{-xx})$$

   - Otherwise (T is lower than or equal to (1 << 10)), the following applies:

$$Y = LutTtoS[T] \quad (8\text{-xx})$$

8. The values R and B are derived as follows:

$$R = (16384 * Y + mR * V + 1024) >> 11$$
$$B = (16384 * Y + mB * U + 1024) >> 11 \quad (8\text{-xx})$$

8.8.3.2 Color samples mapping

Inputs to this process are color sample values R and B and the luma sample value Y.

Output of this process are mapped color sample values R' and B'.

The conversion process of samples R, B into samples R', B' applies as follows.

1. The array LutOetf derived by invoking section 8.8.5.3.
2. The array LutEotf is derived by invoking section 8.8.5.4.
3. The values R1 and G1 are derived as follows.

$$R1 = (R * R + 8192) >> 14$$
$$B1 = (B * B + 8192) >> 14 \quad (8\text{-xx})$$

4. The luminance value L is derived as follows:

$$L = \min(LutOetf[Y], (hdr\_max\_display\_mastering\_luminance * (1 << 28) + 1) >> 1) \quad (8\text{-xx})$$

5. The values R2 and G2 are derived as follows.

$$R2 = \max(0, (R1 * L + 2048) >> 12)$$
$$B2 = \max(0, (B1 * L + 2048) >> 12) \quad (8\text{-xx})$$

6. For X being set to R and G, the mapped value X' is derived by applying the following pseudo-code:
   - The value e is set equal Log2(x >> 8)

BNSDOCID: <XP_____30065768A__I_>

– idx is set equal to (X >> e)

– X' is set equal to LutEotf[e][idx]

8.8.4 Conversion to CbCr constant luminance samples

Inputs to this process are color sample values R' and B' and the luma sample value Y.

Output of this process are mapped color sample values Cb and Cr.

The conversion process of samples R' and B' and the luma sample value Y into samples Cb and Cr applies as follows.

1. The value diff1 is set equal to (B' – Y).
2. The value U is derived as follows:
   – If diff1 is lower than 0, the following applies

   $$U = (diff1 * 527 + 512) >> 10 \quad (8\text{-}xx)$$

   – Otherwise (diff1 is greater than or equal to 0) , the following applies

   $$U = (diff1 * 647 + 512) >> 10 \quad (8\text{-}xx)$$
3. The value diff2 is set equal to (R' – Y).
4. The value V is derived as follows:
   – If diff2 is lower than 0, the following applies

   $$V = (diff2 * 595 + 512) >> 10 \quad (8\text{-}xx)$$

   – Otherwise (diff2 is greater than or equal to 0) , the following applies

   $$V = (diff2 * 1030 + 512) >> 10 \quad (8\text{-}xx)$$
5. The value Cb and Cr are derived as follows:

   $$Cb = Clip3(0, (1 << BitDepthC), U)$$
   $$Cr = Clip3(0, (1 << BitDepthC), V) \quad (8\text{-}xx)$$

8.8.5 Generation of the colour conversion look-up-tables

8.8.5.1 Generation of the look-up-table LutTtoS

The outut of this process is the array LutTtoS[i], for i=0..(1 << bitDepthC) – 1.

The array values LutTtoS[i], for i=0..(1 << bitDepthC) – 1, are generated as follows:

$$LutTtoS[i] = Ceil(sqrt(1 – (i ÷ ((1 << bitDepthC) – 1))) * (1 << bitDepthC)) \quad (8\text{-}xx)$$

8.8.5.2 Generation of the look-up-table LutTtoUVfactor

The outut of this process is the array LutTtoUVfactor[i], for i=0..(1 << bitDepthC) – 1.

The array values LutTtoUVfactor[i], for i=0..(1 << bitDepthC) – 1, are generated as follows:

$$LutTtoUVfactor[i] = Min((1 << 15), Ceil((1 << 15) ÷ sqrt(1 + i ÷ 16))) \quad (8\text{-}xx)$$

8.8.5.3 Generation of the look-up-table LutOetf

The outut of this process is the output array LutOetf[i], for i=0..(1 << bitDepthC) – 1.

The array values LutOetf[i], for i=0..(1 << bitDepthC) – 1, are generated as follows:

– The values c1, c2, c3, m1 and m2 are derived as follows:

$$c1 = 3424 ÷ 4096 \quad (8\text{-}xx)$$
$$c2 = 32 * 2413 ÷ 4096 \quad (8\text{-}xx)$$
$$c3 = 32 * 2392 ÷ 4096 \quad (8\text{-}xx)$$
$$m1 = 1 ÷ (128 * 2523 ÷ 4096) \quad (8\text{-}xx)$$
$$m2 = 1 ÷ (0.25 * 2610 ÷ 4096) \quad (8\text{-}xx)$$

– The value y is derived as follows:

BNSDOCID: <XP_____30065768A__I_>

$y = i \div (1 << bitDepthC)$ (8-xx)

– The value t is modified as follows:

$t = Max(0, y^{m2} - c1)$ (8-xx)

– The value LutOetf[i] is derived as follows:

$LutOetf[i] = Ceil((1 << 27) * (t / (c2 - c3 * t))^{m1})$ (8-xx)

8.8.5.4 Generation of the look-up-table LutEotf

The output of this process is the array LutEotf[i][j], for i=0..19, j=0..255.

The array values LutEotf[i][j], for i=0..19, j=0..255, are generated as follows:

– The values c1, c2, c3, m1 and m2 are derived as follows:

$c1 = 3424 \div 4096$ (8-xx)
$c2 = 32 * 2413 \div 4096$ (8-xx)
$c3 = 32 * 2392 \div 4096$ (8-xx)
$m1 = 128 * 2523 \div 4096$ (8-xx)
$m2 = 0.25 * 2610 \div 4096$ (8-xx)

– The value Lf is derived as follows:

$Lf = ((1 << i) * m) \div (1 << 27)$ (8-xx)

– The value t is modified as follows:

$t = Lf^{m1}$ (8-xx)

– The value LutEotf[i][j] is derived as follows:

$LutEotf[i][j] = Ceil((1 << bitDepthC) * ((c2 * t + c1) \div (1 + c3 * t))^{m2})$ (8-xx)

BNSDOCID: <XP_____30065768A__I_>

Spec2: Text specification using piece-wise linear model for mapping process

Syntax changes

Same changes in 7.3.2.2.1 (General sequence parameter set RBSP syntax) as in Spec 1

Same new section 7.3.2.2.6 Sequence parameter HDR coding extension syntax as in Spec 1

Same changes in 7.3.2.3.1 (General picture parameter set RBSP syntax) as in Spec 1

Add new section 7.3.2.3.9 Picture parameter HDR coding extension syntax

pps_hdr_extension( ) {	Descriptor
hdr_recon_num_comps_minus2	**u(1)**
hdr_recon_scale_bit_depth	
hdr_recon_offset_bit_depth	
hdr_recon_scale_frac_bit_depth	**u(4)**
hdr_recon_offset_frac_bit_depth	**u(4)**
for(c = 0; c <= hdr_recon_num_comps_minus2 + 1; c++) {	**u(4)**
hdr_recon_num_ranges[c]	
hdr_recon_equal_len_ranges_flag[c]	
hdr_recon_zero_offset_val[c]	**ue(v)**
for(i = 0; i < hdr_recon_num_ranges[c]; i++)	**u(1)**
hdr_recon_scale_val[c][**i**]	**u(v)**
if(! hdr_recon_equal_len_ranges_flag[c])	
for(i = 0; i < hdr_recon_num_ranges[c]; i++)	**u(v)**
hdr_recon_range_val[c][**i**]	
}	
}	

Semantics changes

Same changes in 7.4.3.2.1 (General sequence parameter set RBSP semantics) as in Spec 1

Same new semantics section 7.4.3.2.6 (Sequence parameter set HDR extension semantic) as in Spec 1

Add a new semantics section 7.4.3.3.8 (Picture parameter set HDR extension semantics)

hdr_recon_num_comps_minus2 plus 2 specifies the number of color components for which the mapping function is specified. hdr_recon_num_comps_minus2 shall be in the range of 0 to 1, inclusive. When not present, hdr_recon_num_comps_minus2 is set equal to 0.

hdr_recon_scale_bit_depth specifies the number of MSBs used to signal the integer part of syntax element hdr_recon_scale_val[c][i]. The value of hdr_recon_scale_bit_depth shall be in the range of 0 to 15, inclusive. When not present, the value of hdr_recon_scale_bit_depth is inferred to be equal to the value of hdr_recon_scale_bit_depth.

hdr_recon_offset_bit_depth specifies the number of MSBs used to signal the integer part of syntax elements hdr_recon_global_offset_val[c] and hdr_recon_offset_val[c][i]. The value of hdr_recon_offset_bit_depth shall be in the range of 0 to 15, inclusive. When not present, the value of hdr_recon_scale_bit_depth is inferred to be equal to the value of hdr_recon_scale_bit_depth.

BNSDOCID: <XP_____30065768A__I_>

hdr_recon_scale_frac_bit_depth specifies the number of LSBs used to indicate the fractional part of the scale parameter of the i-th partition of the c-th component. The value of hdr_recon_scale_frac_bit_depth shall be in the range of 0 to 15, inclusive. When not present, the value of hdr_recon_scale_frac_bit_depth is inferred to be equal to the value of hdr_recon_scale_frac_bit_depth.

The total number of bits scaleBitDepth to represent hdr_recon_scale_val[c][i] is defined as:

scaleBitDepth = hdr_recon_scale_bit_depth + hdr_recon_scale_frac_bit_depth (8-xx)

hdr_recon_offset_frac_bit_depth specifies the number of LSBs used to indicate the fractional part of the offset parameter of the i-th partition of the c-th component and global offset of the c-th component. The value of hdr_recon_offset_frac_bit_depth shall be in the range of 0 to 15, inclusive. When not present, the value of hdr_recon_offset_frac_bit_depth is inferred to be equal to the value of hdr_recon_offset_frac_bit_depth.

The total number of bits offsetBitDepth to represent hdr_recon_global_offset_val[c] and hdr_recon_offset_val[c][i] is defined as:

offsetBitDepth = hdr_recon_offset_bit_depth + hdr_recon_offset_frac_bit_depth (8-xx)

hdr_recon_num_ranges[c] specifies the number of ranges, for the c-th component, in to which the output sample range of the colour piece-wise model is partitioned in to. The value of hdr_recon_num_ranges[c] shall be in the range of 0 to 63(32), inclusive

hdr_recon_equal_len_ranges_flag[c] equal to 1 indicates that output sample range of the colour piece-wise model for the c-th component is partitioned into hdr_recon_num_ranges[c] nearly equal partitions, and the partition widths are not explicitly signalled. hdr_recon_equal_ranges_flag[c] equal to 0 indicates that that output sample range of the colour piece-wise model may be partitioned into hdr_recon_num_ranges[c] partitions not all of which are of the same size, and the partitions widths are explicitly signalled.

hdr_recon_global_offset_val[c] is used to derive the offset value that is used to map the smallest value of the valid input data range for the colour piece-wise model for the c-th component. The bit-depth of hdr_recon_global_offset_val is equal to colourOffsetBitDepth.

hdr_recon_scale_val[c][i] is used to derive the offset value that is used to derive the width of the of the i-th partition of the colour piece-wise model for the c-th component. The bit-depth of hdr_recon_scale_val[c][i] is equal to colourScaleBitDepth.

The variable ScaleValue[c][i] is derived as follows :

ScaleValue[c][i] = (hdr_recon_scale_val[c][i] – (1 << (scaleBitDepth – 1)) (8-xx)

hdr_recon_range_val[c][i] is used to derive the width of the of the i-th dynamic range partition of the colour piece-wise model that is used to derive the offset value for the i-th dynamic range partition for the c-th component. The bit-depth of hdr_recon_range_val[c] is offsetBitDepth bits.

The variable RangeValue[c][i] is derived as follows :

RangeValue[c][i] = (hdr_recon_range_val[c][i] – (1 << (offsetBitDepth – 1)) (8-xx)

Same added text at the end of 8.1.3 (Decoding process for a coded picture with nuh_layer_id equal to 0) as in Spec 1

Add the following new section 8.8 (HDR reconstruction process)

8.8 HDR reconstruction process

8.8.6 General derivation process for the reconstructed HDR picture

Same as in Spec 1

8.8.7 HDR mapping process

Same as in Spec 1

8.8.2.1 HDR mapping process of luma sample values

Inputs to this process are a luma location (xP, yP) specifying the luma sample location relative to the top-left luma sample of the current picture and a luma sample array $inPicture_L$.

BNSDOCID: <XP_____30065768A__I_>

Outputs of this process is a HDR mapped luma sample value mappedLumaSample.

The value of mappedLumaSample is derived by applying the following ordered steps:

1. The array mappingLut is derived as by invoking clause 8.8.2.3 with a value cIdx set equal to 0 as input and the array mappingLut as output.

2. The value of mappedLumaSample is derived as follows:

$$mappedLumaSample = mappingLut[\ inPicture_L[\ xP\ ][\ yP\ ]\ ] \quad (8\text{-}xx)$$

8.8.2.2 HDR mapping process of chroma sample values

Inputs to this process are a luma location (xP_C, yP_C) specifying the chroma sample location relative to the top-left chroma sample of the current picture, a luma sample array $inPicture_L$, a chroma sample array $inPicture_C$ and a value cIdx indicating the index of the color component.

Output of this process is a hdr mapped chroma sample value mapChrSample.

The value of mapChrSample is derived by applying the following ordered steps:

1. The value of Ycoloc is derived as follows:

$$Ycoloc = inPicture_L[\ SubWidthC * xP_C\ ][\ SubHeightC * yP_C\ ] \quad (8\text{-}xx)$$

2. The value of c is set equal to Min(cIdx, hdr_recon_num_comps_minus2 – 1)
3. The array colorReconLut is derived as by invoking clause 8.8.2.3 with with the value cIdx as input and the array correctionLut as output.

4. The value of chrScaling, bitShift, offset0, offset1 and offset2 are derived as follows:

$$chrScaling = (\ hdrReconRenormalizing * colorReconLut[\ Ycoloc\ ] + 128\ ) >> 8 \quad (8\text{-}xx)$$
$$offset0 = 1 << (\ BitDepthC - 1\ ) \quad (8\text{-}xx)$$
$$bitShift = 12 \quad (8\text{-}xx)$$
$$offset1 = 1 << (\ bitShift - 1\ ) \quad (8\text{-}xx)$$
$$offset2 = 1 << 14 \quad (8\text{-}xx)$$

5. The value of mapChrSample is derived as follows:

$$mapChrSample = (\ chrScaling * (\ inPicture_C[\ xP_C\ ][\ yP_C\ ] - offset0\ ) + offset1\ ) >> bitShift + offset2 \quad (8\text{-}xx)$$

6. The value of mapChrSample is modified as follows:

$$mapChrSample = clip3(\ 0, (\ 1 << 16\ ) - 1, mapChrSample\ ) \quad (8\text{-}xx)$$

8.8.2.3 Mapping Look-up-table derivation

Inputs to this process is avalu cIdx.

Output of this process is an array out_lut_value[i], i=0..(in_lut_size – 1).

For Y=0..(in_lut_size – 1), the value out_lut_value[Y] is derived as follows:

– The values shiftIn, shiftOut, OffIn and OffOut are defined as follows:

$$shiftIn = rangeBitDepth - BitDepth_L \quad (8\text{-}xx)$$
$$offIn = 1 << (\ shitfIn - 1\ ) \quad (8\text{-}xx)$$
$$shiftOut = rangeBitDepth + scaleBitDepth - 16 \quad (8\text{-}xx)$$
$$offOut = 1 << (\ shitfOut - 1\ ) \quad (8\text{-}xx)$$

– The values InRanges[c][i] and OutRanges[c][i], for i = 0..hdr_recon_num_ranges – 1, are derived as follows:
  – If i is equal to 0, the following applies:

$$InRanges[\ c\ ][\ 0\ ] = 0 \quad (8\text{-}xx)$$
$$OutRanges[\ c\ ][\ 0\ ] = (\ hdr\_recon\_global\_offset\_val[\ c\ ] << scaleBitDepth + offOut\ ) >> shitfOut \quad (8\text{-}xx)$$

BNSDOCID: <XP_____30065768A__I_>

- Otherwise (i is not equal to 0), the following applies:

```
InRanges[ c ][ i ] = InRanges[ c ][ i − 1 ] + ( RangeValue[ c ][ i − 1 ] + offIn ) >> shiftIn                (8-xx)
OutRanges[ c ][ i ] = OutRanges[ c ][ i − 1 ] + ( RangeValue[ c ][ i − 1 ] *
                                ScaleValue[ c ][ i − 1 ] + offOut ) >> shitfOut                              (8-xx)
```

- The values OffsetValue[c][i], for i = 0..hdr_recon_num_ranges[c] − 1, are derived as follows:

```
OffsetValue[ c ][ i ] = OutRanges[ c ][ i + 1 ] − ( InRanges[ c ][ i + 1 ] *
                                ScaleValue[ c ][ i − 1 ] + offOut ) >> shiftOut                              (8-xx)
```

- The value out_lut_value[Y] is derived as follows:
  - The values shiftDest1, offDest1, shiftDest2 and offDest2 are defined as follows:

```
shiftDest1 = BitDepthL + scaleBitDepth - 16(8-xx)
offDest1 = 1 << ( shiftDest1 − 1 )                                                                           (8-xx)
shiftDest2 = rangeBitDepth + scaleBitDepth - 16                                                              (8-xx)
offDest2 = 2 << ( shiftDest2 − 1 )                                                                           (8-xx)
numComp = hdr_recon_num_ranges[ c ]                                                                          (8-xx)
```

  - The value skipLoop is set equal to 0
  - The following applies:

```
for( i = 0; i < hdr_recon_num_ranges[ c ] && skipLoop = = 0; i++ )
    if( Y  <= InRanges[ c ][ i + 1 ] ) {
        out_lut_value[ Y ] = ( Y * ScaleValue[ c ][ i ] + offDest1 ) >> shitfDest1 +
                    OffsetValue[ c ][ i ]                                                                    (8-xx)
        skipLoop = 1
    }
```

  - If skipLoop is equal to 0, the following applies:

```
out_lut_value[ Y ] = ( InRanges[ c ][ numComp ] * ScaleValue[ c ][ numComp − 1 ] + offDest2 )
        >> shitfDest2  + OffsetValue[ c ][ numComp − 1 ]                                                     (8-xx)
```

8.8.8 HDR colour reconstruction process

1 Same as in Spec 1

BNSDOCID: <XP_____30065768A__I_>

Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11
23rd Meeting: San Diego, USA, 19–26 February 2016

Document: JCTVC-W0059 / m37693

Title: **HDR CE2: CE6-4.6b report on usage of SEI for Test Model**

Status: Input Document to JCT-VC

Purpose: Proposal

Author(s) or Contact(s): E. François, C. Chevance, Y. Olivier

Tel: Edouard.francois@technicolor.com
Email: Christophe.chevance@technicolor.com
Yannick.olivier@technicolor.com

Source: Technicolor

Abstract

This document reports results from experiment HDR CE6-4.6b. The experiment aims at replacing the SPS/PPS signalling used in ETM to convey the reshaper metadata, by an SEI container. The tests are made in reshape setting mode 1 focused on SDR backward compatibility, with same algorithm as in CE2.b-1. It is reported that same performance are obtained as CE2.b-1.

1 Introduction

The principle of the Exploratory Test Model [1,2] for HDR video coding is to apply prior to the encoding a pre-processing step, named reshaping, that converts the input HDR content into a reshaped version aiming at improving the compression performance and/or providing an SDR backward compatible content. The reshaper also generates reshaping metadata. After decoding, the video has to be post-processed (by the inverse reshaper) using the reshaping metadata to reconstruct the HDR video. The reshaper can work in two main modes, HDR-only mode targeting improved compression performance, and SDR backward compatible mode targeting the compatibility with legacy SDR devices while maintaining high compression performance.

The reshaping metadata can be conveyed in SPS/PPS or in an SEI container. In ETM, SPS/PPS is used. SPS/PPS changes may raise some issues, in particular regarding conformance. The SEI option is a solution to address this point. As the SEI is in principle not mandatory (the decision to make it mandatory or not is usually rather up to the application standards), there may be risks that it is not properly conveyed. In case of HDR-only mode, the loss of reshaping metadata is highly prejudicial since the reshaped signal is not supposed to be viewable as is on an HDR display (and even less on an SDR display). There is a high risk that the decoded signal cannot be correctly rendered. In case of SDR backward compatible mode, the signal can still be rendered, without reshaping, on an SDR and even on an HDR device, as soon as it understands the SDR format signalled in the VUI.

The purpose of this experiment is to move the reshaping data into a new SEI message, for the SDR backward compatible mode. The experiments are performed for reshape mode 1, using the tuning algorithm delivered in CE2.b-1.

2 ETM_RC_r1 process overview for reshapeSetting 1 configuration

Reshaper setting 1 configuration applies cross-plane mode for chroma reshaping, using two piece-wise-linear (PWL) modelled look-up-tables (LUTs) to control the chroma components. The luma component is reshaped using an intra-plane piece-wise-polynomial (PWP) modelled LUT. In the current ETM settings, 8 pieces are used both for luma PWP and for chroma PWL models.

The reshaping process works in four main steps, as shown in Figure 1:

BNSDOCID: <XP_____30117831A__1_>

1) Conversion of input HDR signal into Y'CbCr 10b 4:2:0 with inverse ST.2084 (PQ) EOTF;
2) Reshaping of 10b luma PQ to get an SDR-like 10b luma;
3) Automatic generation of an intermediate SDR 4:2:0 Y'CbCr picture;
4) Derivation of cross-plane chroma PWL models parameters to fit as much as possible the chroma components of the intermediate 4:2:0 Y'CbCr SDR picture.

In the following sub-sections, short description of these processes is provided. More details on the reshaping algorithms in ETM_RC_r1 are provided in document JCTVC-W0031 [1].

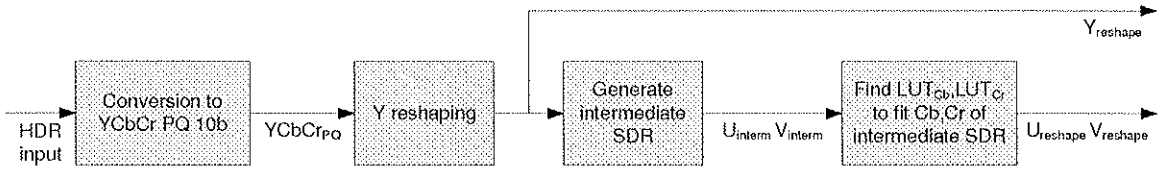


Figure 1: simplified block diagram of ETM_RC_r1 process in reshapeSetting 1 configuration.

2.1 Y reshaping

Y reshaping is achieved using the following function

$$F(L) = \log(1. + (L / Ba)^{g}) / \log(1. + (P / Ba)^{g})$$

controlled by two adaptive tuning parameters Ba, g, and by the peak luminance P. In ETM_RC_r1, g is set to 2.8, and Ba varies depending on the content. The resulting function is modeled by a PWP 8 pieces model. Y is then scaled to and quantized in Full or Standard Range signal.

2.2 Intermediate SDR generation

To generate the intermediate SDR ($Y_{interm}U_{interm}V_{interm}$), the following steps apply:

1) Generate HDR luminance L_{hdr} from input linear-light HDR R,G,B samples
2) Convert $Y_{reshape}$ to linear-light $L_{reshape}$ as follows

$$L_{reshape} = ((Y_{reshape} - off) / scale)^{(1 / 0.45)}$$

where off and scale are the parameters for Full or Standard Range normalization of $Y_{reshape}$

3) Scale linear-light HDR R,G,B samples by ratio $w = L_{reshape} / L_{hdr}$
4) Convert scaled R,G,B to SDR R',G',B' as follows

$$R' = 1023 * (R_{scaled})^{0.45} \qquad G' = 1023 * (G_{scaled})^{0.45} \qquad B' = 1023 * (B_{scaled})^{0.45}$$

5) Convert SDR R',G',B' to 10b Y'CbCr then downsample chroma to get 10b 4:2:0 signal $Y_{interm}U_{interm}V_{interm}$

2.3 Chroma LUTs fitting

The chroma LUTs fitting computes the two LUTs LUT_{cb} and LUT_{cr} in order to fit as much as possible the chroma components of the intermediate SDR picture. The LUTs are derived by least mean squares optimization, with minimization of the following functions:

$$\sum_{p=(x,y)} \left(\frac{(U_{PQ}(p) - O_C)}{LUT_{cb}[Y_{reshape}(p)]} - (U_{interm}(p) - O_C) \right)^2$$

$$\sum_{p=(x,y)} \left(\frac{(V_{PQ}(p) - O_C)}{LUT_{cr}[Y_{reshape}(p)]} - (V_{interm}(p) - O_C) \right)^2$$

BNSDOCID: <XP.....30117831A...1..>

with O_C equal to 512 (2^9) for 10 bits content.

More details on the cross-plane LUT optimization for reshape setting 1 are provided in document JCTVC-W0031 [1].

3 Proposed modifications

3.1 Reshaper for reshapeSetting 1 configuration

The main changes are shown in shown in Figure 2. First modification is the addition of a fitting process for the luma component, in order to fit as much as possible the intermediate SDR luma. This re-tuning of luma aims at improving the perceived saturation of the output SDR version resulting from the reshaping.

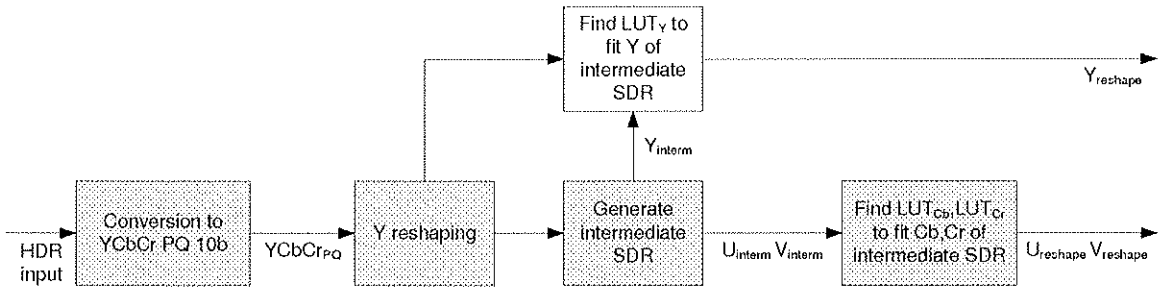


Figure 2: simplified block diagram of CE2.b-1 (and CE6-4.6b) reshaping algorithm.

The luma LUT is derived as follows:

$$LUTy[\,y] = \sum_{\substack{p=(x,y)\ such \\ that\ Y_{pq}(p)=y}} \left[w\left(y, Y_{interm}(p)\right) * Y_{interm}(p)\right]$$

with $w(y, Y_{interm})$ a weighting function defined as

$$w(y, Y_{interm}) = A/(A + \max(0, Y - Y_{interm}))$$

The function intends to favour or not the low luma levels from the reshaping. When A is small (we use 0.01 by default), the weight will favour low luma values (and therefore more intense colors).

In addition, the following changes have been made:

- In the Y reshaping block, to get more contrasted pictures, the parameter g is set to 2.4 instead of 2.8.
- In the Intermediate SDR generation block, a smooth clipping applies once the R,G,B samples are scaled (after step 3 in process described in section **Erreur ! Source du renvoi introuvable.**).
- In the PWL modelling of chroma LUTs, 32 pieces are used instead of 8. This allows finer matching to the intermediate SDR colors and results in a noticeable impact on the SDR rendering.

3.2 Changes in the HM

The HM has been modified to move all the ETM SPS/PPS reshaper-related syntax into a new SEI.

This corresponds to the following syntax changes:

BNSDOCID: <XP_____30117831A__1_>

sei_payload(payloadType, payloadSize) {	**Descriptor**
if(nal_unit_type == PREFIX_SEI_NUT)	
...	
else if(payloadType == 148)	
hdr_reconstruction_info(payloadSize)	
...	
}	

BNSDOCID: <XP_____30117831A__I_>

hdr_reconstruction_info(payloadSize) {	**Descriptor**
reshape_present_flag	ue(v)
if(reshape_present_flag) {	
reshape_chroma_crosschannel_flag	u(1)
reshape_input_luma_bit_depth_minus8	ue(v)
reshape_input_chroma_bit_depth_minus8	ue(v)
reshape_output_luma_bit_depth_minus8	ue(v)
reshape_output_chroma_bit_depth_minus8	ue(v)
reshape_default_flag	u(1)
if(!reshape_default_flag) {	
reshape_scale_int_bit_depth	u(4)
reshape_offset_int_bit_depth	u(4)
reshape_scale_frac_bit_depth	u(4)
reshape_offset_frac_bit_depth	u(4)
reshape_negative_scales_present_flag	u(1)
}	
reshape_num_comps_minus2	ue(v)
for(c = 0; c < reshape_num_comps_minus2 + 2; c++) {	
reshape_num_ranges_minus1[c]	ue(v)
reshape_equal_ranges_flag[c]	u(1)
reshape_global_offset_val[c]	u(v)
if(!reshape_equal_ranges_flag[c])	
for (i = 0; i < reshape_num_ranges_minus1[c] + 1; i++)	
reshape_range_val[c][i]	u(v)
if (c = = 0) {	
coeff_log2_offset_minus2	ue(v)
reshape_continuity_flag	u(1)
for(i = 0; i < reshape_num_ranges_minus1[c] + 2; i++) {	
poly_coef_int[i][0]	ue(v)
poly_coef_frac[i][0]	u(v)
}	
if(reshape_continuity_flag = = 1) {	
poly_coef_int[0][1]	se(v)
poly_coef_frac[0][1]	u(v)
}	
} else {	
for(i = 0; i < reshape_num_ranges_minus1[c] + 1; i++) {	
reshape_scale_int_val[c – 1][i]	u(v)
reshape_scale_fract_val[c – 1][i]	u(v)
}	
if (reshape_chroma_crosschannel_flag) {	
reshape_offset1[c – 1]	u(16)
reshape_offset2[c – 1]	u(16)
}	
}	
}	
}	

BNSDOCID: <XP_____30117831A__I_>

}	

The semantics and reconstruction process are the same as for the ETM [1].

4 Objective metrics

To be completed

The results are fully matching the results provided in CE2.b-1.

5 References

[1] "Proposed initial Test Model for HEVC HDR extension," R. Brondijk, R. Goris, R. van der Vleuten, L. van de Kerkhof, D. Rusanovskyy, A. Ramasubramonian, D. Bugdayci, S. Lee, J. Sole, M. Karczewicz, F. Galpin, S. Lasserre, F. Le Leannec, T. Poirier, E. François, MPEG document M37285, Geneva, Switzerland, Oct. 2015.

[2] "Description of the reshaper parameters derivation process in ETM reference software," K. Minoo, T. Lu, P. Yin, L. Kerofsky, D. Rusanovskyy, E. François, JCTVC-W0031/MPEG M37536, San Diego, USA, Feb. 2016.

[3] "HDR CE2: report of experiment CE2.b-1 on SDR-backward compatible configuration (reshape setting 1)," E. François, Y. Olivier, C. Chevance, JCTVC-W0033/MPEG M37691, San Diego, USA, Feb. 2016.

6 Patent rights declaration(s)

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).

BNSDOCID: <XP_____30117831A___I_>