|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **International Telecommunication Union** | | |
|  | | DRAFT | | |
| **ITU-T** | **Technical Paper** | | |
| TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU | | (December 2015) | |
|  | SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving video | | | |
|  | **HSTP-CCHDR Conversion and Compression of High Dynamic Range video** | | | |
|  |  | | | |

Summary

This document provides guidance on processing of high dynamic range (HDR) video. The purpose of this document is to provide a reference for best-practice operation of AVC or HEVC when used for compressing HDR video. This document includes a description of processing steps for converting linear light, RGB, 4:4:4 video into ST 2084, Y’CbCr, 4:2:0 video before encoding. This document also includes a description of processing steps for converting ST 2084, Y’CbCr, 4:2:0 to linear light, RGB, 4:4:4 after decoding. Some high level recommendations for compression with HEVC and AVC are also included in this document.

Keywords

HDR, High Dynamic Range, Video Coding

Change Log

The current version is “draft2” which includes modifications suggested on email on December 17.

The “draf1” version was the very first draft provided by Jonatan Samuelsson ([jonatan.samuelsson@ericsson.com](mailto:jonatan.samuelsson@ericsson.com)) for review by JCT-VC experts.

|  |  |  |
| --- | --- | --- |
| **Editor:** | - | - |

**CONTENTS**

| Page |
| --- |
| 1 Scope 4  2 References 4  3 Definitions 4  4 Abbreviations and acronyms 5  5 Mathematical functions and operations 5  6 Constants 6  7 Introduction 6  8 Pre-encoding processing 7  8.1 Process for applying the inverse transfer function 8  8.2 Process for calculating Cb 9  8.3 Process for calculating Cr 9  8.4 Chroma quantization process 9  8.5 Chroma subsampling process 9  8.6 Chroma upsampling process 9  8.7 Inverse chroma quantization process 10  8.8 Process for calculating luminance 10  8.9 Process for luma adjustment 10  8.10 Process for calculating luminance from luma and chroma 11  8.11 Process for calculating RGB 11  8.12 Process for applying the transfer function 11  9 Encoding process 12  9.1 HEVC encoding 12  9.2 AVC encoding 13  10 Decoding process 14  11 Post-decoding processing 14  11.1 Inverse luma quantization process 15 |

ITU-T Technical Paper HSTP-CCHDR

Conversion and Compression of High Dynamic Range video

# Scope

This document provides guidance on processing of high dynamic range video including conversions steps for going from a linear light RGB representation with BT.2020 colour primaries to 10 bit, ST 2084, 4:2:0, non-constant luminance Y’CbCr representation. The scope of this document is illustrated in Figure 1.



Figure 1. Illustration of the scope of this document.

It should be noted that the content preparation step as well as the display adaptation step are considered to be out of scope of this document. The processing steps described in this document are made available for reference only and does not contain any elements of normative nature. It is possible to replace one or more of the processing steps described in this document for example in order to reduce computational complexity or to improve fidelity. The intention with this document is to provide a best-practice guideline for operating an HDR video system that is constrained to use a 10 bit, ST 2084, 4:2:0, non-constant luminance Y’CbCr representation. This configuration is aligned with the HDR10 media profile defined in [CFFMF2.1]. The processing steps in this document are optimized with the intention of providing the best possible result when the same Hypothetical Reference Viewing Environment (HRVE) is used before the HDR system as after the HDR system. This document does not account for when different viewing environments are used before and after the HDR system.

NOTE 1 – This document does not provide any description of any preferred HRVE but acknowledges the fact that in many applications of HDR video it might be desirable with a well-defined HRVE description to ensure alignment between content preparation and content consumption.

# References

[AVC] Recommendation ITU-T H.264 (V10) (2015), *Advanced video coding for generic audiovisual services*. Also available as ISO/IEC 14496-10.

[BT.709] Recommendation ITU-R BT.709-6 (2015), *Parameter values for the HDTV standards for production and international programme exchange.*

[BT.2020] Recommendation ITU-R BT.2020-2 (2015), *Parameter values for ultra-high definition television systems for production and international programme exchange.*

[CFFMF2.1] DECE, *Common File Format & Media Formats Specification Version 2.1.*

[HEVC] Recommendation ITU-T H.265 (V3) (2015), *High Efficiency Video Coding.* Also available as ISO/IEC 23008-3.

[RP 431-2] SMPTE Standard RP 431-2 (2011), *D-cinema Quality - Reference Projector and Environment*.

[ST 2084] SMPTE Standard ST 2084 (2014), *High Dynamic Range Electro-Optical Transfer Function for Mastering Reference Display*.

# Definitions

This document defines the following terms:

**3.1 Inverse transfer function**: The function used in the pre-encoding processing to convert from a linear representation to non-linear representation. The function is the inverse of the transfer function used in the post-decoding processing. In applications where the transfer function is called Electro-Optical Transfer Function (EOTF), the inverse transfer function may be called inverse-EOTF.

**3.2 Random Access Point Access Unit (RAPAU)**: An access unit in the bitstream containing an intra coded picture with the property that all pictures following the intra coded picture in output order can be correctly decoded without using any information preceding the Random Access Point Access Unit in the bitstream.

**3.1 Transfer function**: The function used in the post-decoding processing to convert from a non-linear representation to linear representation. In applications where no further processing or display adaptation is performed, this function is sometimes called Electro-Optical Transfer Function (EOTF).

# Abbreviations and acronyms

This document uses the following abbreviations and acronyms:

AVC Advanced Video Coding

EOTF Electro-Optical Transfer Function

HDR High Dynamic Range

HEVC High Efficiency Video Coding

HRVE Hypothetical Reference Viewing Environment

QP Quantization Parameter

RAPAU Random Access Point Access Unit

RGB Red Green Blue

SDR Standard Dynamic Range

SEI Supplemental Enhancement Information

WCG Wide Colour Gamut

# Mathematical functions and operations

/ Integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.

 Used to denote division in mathematical equations where no truncation or rounding is intended. For example, 7 4 = 1.75.

Used to denote division in mathematical equations where no truncation or rounding is intended. For example, = 1.75.

Abs( x ) 



Clip3( x, y, z ) =



Floor( x ) the largest integer less than or equal to x.

Round( x ) = Sign( x ) \* Floor( Abs( x ) + 0.5 )

Sign( x ) =



# Constants

The following variables are used as constants throughout this document. For some of the variables it is indicated within parenthesis what the variable represents an approxiamtion of.

m1 = 0.1593017578 (represents )

m2 = 78.84375

c1 = 0.8359375

c2 = 18.8515625

c3 = 18.6875

wR = 0.2627

wG = 0.6780

wB = 0.0593

a13 = 1.4746 (represents 2\*(1-wR))

a32 = 1.8814 (represents 2\*(1-wB))

a22 = 0.16455 (represents )

a23 = 0.57135 (represents )

b21 = 0.139630 (represents )

b22 = 0.360370 (represents )

b32 = 0.459786 (represents )

b33 = 0.040214 (represents )

# Introduction

The HDR System described in this document consist of four major processes; Pre-encoding processing (clause 8), Encoding process (clause 9), Decoding process (clause 10), and Post-decoding processing (clause 11). These four processes are applied sequentially with the output of one process being used as input to the next process according to the above-mentioned order.

It is assumed that the input to the HDR System is linear light, RGB, 4:4:4 video and the output of the system is also linear light, RGB, 4:4:4 video, targeted to resemble the input video as closely as possible. Other video formats can be input to the HDR System by first converting them to linear light, RGB, 4:4:4 video. The HDR System described in this document is in practice a system for both HDR and Wide Colour Gamut (WCG) video since it is assumed that the input video is represented with colour primaries in accordance with [BT.2020].

NOTE 1 – For a fixed point linear representation of HDR video, approximately a 28 bit representation would be required to avoid introducing quantization errors. In practice, the input to the HDR System will typically be in a non-linear representation that could either first be converted to linear light or be directly converted to ST 2084.

It is assumed that encoding and decoding is performed in 4:2:0, 10 bit representation.

NOTE 2 – The assumption of 4:2:0 and 10 bit representation is made with focus on consumer and direct-to-home applications. Processes similar to the ones described in this document can be used for conversion and compression of 4:2:2 video and/or video with bit-depth higher than 10.

The post-decoding processing steps are aligned with what is commonly referred to as non-constant luminance in which colour conversion (to R’G’B’) is applied before applying the transfer function.

The processes described in this document are applied to one or more pictures with a width equal to PicWidthInSamples and a height equal to PicHeightInSamples. The variables PicWidthInHalf, and PicHeightInHalf are used to represent PicWidthInSamples / 2 and PicHeightInSamples / 2, respectively.

There is no specific or minimum bitdepth required for performing the operations described in the pre-encoding processing and the post-encoding processing. Using the precision associated with 64 bit floating point operations will give high accuracy, but it is also possible to use fixed point arithmetic and/or floating point operation with precision lower than 64 bit. Using too low precision should be avoided since it could lead to loss of precision in the output video. The input to the encoding step and the output of the decoding step is 10 bit integer representations.

# Pre-encoding processing

Inputs to this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleR of floating point red samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleG of floating point green samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleB of floating point blue samples in the range of 0.0 to 1.0, inclusive,

NOTE 1 – The processes described in this document have been designed to work optimally when the input samples represent light according to a linear function such that the value 0.0 corresponds to 0 cd/m2 and the value 1.0 corresponds to 10,000 cd/m2.

Outputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of integer luma samples in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCb of integer chroma samples of the component Cb in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCr of integer chroma samples of the component Cr in the range of 0 to 1023, inclusive.

A schematic overview of the pre-encoding processing is illustrated in Figure 2. The input (R, G, B) corresponds to PicSampleR, PicSampleG, and PicSampleB, respectively. The output (Y’, Cb, Cr) corresponds to PicSampleL, PicSampleSubCb, and PicSampleSubCr, respectively.



Figure 2. Overview of the pre-encoding processing

The pre-encoding processing consists of the following ordered steps:

1. A (PicWidthInSamples)x(PicHeightInSamples) array of non-linear red samples PicSampleNonLinearR is derived by invoking the process for applying the inverse transfer function specified in clause 8.1, with PicSampleR[ xP ][ yP ] as input and with the output assigned to PicSampleNonLinearR[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
2. A (PicWidthInSamples)x(PicHeightInSamples) array of non-linear green samples PicSampleNonLinearG is derived by invoking the process for applying the inverse transfer function specified in clause 8.1, with PicSampleG[ xP ][ yP ] as input and with the output assigned to PicSampleNonLinearG[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
3. A (PicWidthInSamples)x(PicHeightInSamples) array of non-linear blue samples PicSampleNonLinearB is derived by invoking the process for applying the inverse transfer function specified in clause 8.1, with PicSampleB[ xP ][ yP ] as input and with the output assigned to PicSampleNonLinearB[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
4. A (PicWidthInSamples)x(PicHeightInSamples) array of Cb samples PicSampleCb is derived by invoking the process for calculating Cb specified in clause 8.2, with PicSampleNonLinearR[ xP ][ yP ], PicSampleNonLinearG[ xP ][ yP ] and PicSampleNonLinearB[ xP ][ yP ] as input and with the output assigned to PicSampleCb[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
5. A (PicWidthInSamples)x(PicHeightInSamples) array of Cr samples PicSampleCr is derived by invoking the process for calculating Cr specified in clause 8.3, with PicSampleNonLinearR[ xP ][ yP ], PicSampleNonLinearG[ xP ][ yP ] and PicSampleNonLinearB[ xP ][ yP ] as input and with the output assigned to PicSampleCr[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
6. A modified chroma sample value PicSamplebCb[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cb is derived by invoking the chroma quantization process specified in clause 8.4 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleCb given as inputs.
7. A modified chroma sample value PicSamplebCr[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cr is derived by invoking the chroma quantization process specified in clause 8.4 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleCr given as inputs.
8. The subsampled chroma sample value PicSampleSubCb[ xPC ][ yPC ], with xPC = 0..PicWidthInHalf − 1, yPC = 0..PicHeightInHalf − 1, of the chroma component Cb is derived by invoking the chroma subsampling process specified in clause 8.5 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleCb given as inputs.
9. The subsampled chroma sample value PicSampleSubCr[ xPC ][ yPC ], with xPC = 0..PicWidthInHalf − 1, yPC = 0..PicHeightInHalf  − 1, of the chroma component Cr is derived by invoking the chroma subsampling process specified in clause 8.5 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleCr given as inputs.
10. A modified chroma sample value PicSamplebCb[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cb is derived by invoking the chroma upsampling process specified in clause 8.6 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleSubCb given as inputs.
11. A modified chroma sample value PicSamplebCr[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cr is derived by invoking the chroma upsampling process specified in clause 8.6 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleSubCr given as inputs.
12. A modified chroma sample value PicSamplebCb[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cb is derived by invoking the inverse chroma quantization process specified in clause 8.7 with PicSampleCb[ xPC ][ yPC ] given as input.
13. A modified chroma sample value PicSamplebCr[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cr is derived by invoking the inverse chroma quantization process specified in clause 8.7 with PicSampleCr[ xPC ][ yPC ] given as input.
14. A (PicWidthInSamples)x(PicHeightInSamples) array of target luminance samples YTarget[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1, is derived by invoking the process for calculating luminance specified in clause 8.8 with PicSampleR[ xP ][ yP ], PicSampleG[ xP ][ yP ] and PicSampleB[ xP ][ yP ] given as inputs.
15. The sample array PicSampleL is derived by invoking the luma adjustment process specified in clause 8.9 with sample arrays YTarget, PicSampleCb and PicSampleCr given as inputs.

NOTE 2 – Some of the pre-encoding processing steps can be performed in parallel as can be seen in Figure 2.

## Process for applying the inverse transfer function

Input to this process is one variable LinearVal.

Output of this process is one variable NonLinearVal.

The variable NonLinearVal is derived as follows:



NOTE 1 – The inverse transfer function in this document is functionally identical to the Inverse-EOTF in [ST 2084].

## Process for calculating Cb

Inputs to this process are three variables RedVal, GreenVal and BlueVal.

Output of this process is one variable CbVal.

The variable CbVal is derived as follows:

CbVal = Clip3(–0.5, 0.5, –b21 \* RedVal – b22 \* GreenVal + 0.5 \* BlueVal)

NOTE 1 – The equation for calculating Cb is functionally identical to the one described in [BT.2020].

## Process for calculating Cr

Inputs to this process are three variables RedVal, GreenVal and BlueVal.

Output of this process is one variable CrVal.

The variable CrVal is derived as follows:

CrVal = Clip3(–0.5, 0.5, 0.5 \* RedVal – b32 \* GreenVal – b33 \* BlueVal)

NOTE 1 – The equation for calculating Cr is functionally identical to the one described in [BT.2020].

## Chroma quantization process

Input to this process is a variable BeforeQuant.

Output of this process is a variable AfterQuant.

The variable AfterQuant is derived as follows:

AfterQuant = Clip3(0, 1023, Round(BeforeQuant \* 896 + 512))

## Chroma subsampling process

Inputs to this process are:

– a chroma sample location ( xPC, yPC ) relative to the top-left chroma sample,

– a chroma sample array PicSampleC.

Output of this process is the subsampled chroma sample value ChromaSample.

The value of the subsampled chroma sample value ChromaSample is derived by applying the following ordered steps:

1. The sample value tempArray[ n ] with n = 0..2, is derived as follows:

yPos = Clip3( 0, PicHeightInSamples − 1, (yPC << 1)+ n − 1 )

tempArray[ n ] = PicSampleC[ Clip3( 0, PicWidthInSamples − 1, (xPC << 1) − 1 ), yPos ] +  
 6 \* PicSampleC[ Clip3( 0, PicWidthInSamples − 1, (xPC << 1)), yPos ] +   
 PicSampleC[ Clip3( 0, PicWidthInSamples − 1, (xPC << 1) + 1 ), yPos ]

1. The resampled chroma sample value ChromaSample is derived as follows:

ChromaSample = ( tempArray[ 0 ] +  
 6 \* tempArray[ 1 ] +   
 tempArray[ 2 ] + 32 )  >>  6

## Chroma upsampling process

Inputs to this process are:

– a chroma sample location ( xPC, yPC ) relative to the top-left chroma sample,

– the chroma reference sample array PicSampleC.

Output of this process is the upsampled chroma sample value ChromaSample.

Table 1 specifies the 4-tap filter coefficients fC[ p, x ] with p = 0..1 and x = 0..3 used for the chroma upsampling process.

Table 1– 2-phase chroma resampling filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase p** | **Interpolation filter coefficients** | | | |
| fC[ p, 0 ] | fC[ p, 1 ] | fC[ p, 2 ] | fC[ p, 3 ] |
| 0 | 0 | 16 | 0 | 0 |
| 1 | −1 | 9 | 9 | −1 |

The value of the upsampled chroma sample value ChromaSample is derived by applying the following ordered steps:

1. The variables xRef and xPhase are derived as follows:

xRef =   xPC >>  1

xPhase = xPC % 2

1. The variables yRef and yPhase are derived as follows:

yRef =  yPC >>  1

yPhase = yPC % 2

1. The sample value tempArray[ n ] with n = 0..3, is derived as follows:

yPosRL = Clip3( 0, PicHeightInHalf − 1, yRef + n − 1 )

tempArray[ n ] = fC[ xPhase, 0 ] \* PicSampleC[ Clip3( 0, PicWidthInHalf − 1, xRef − 1 ), yPosRL ] +  
 fC[ xPhase, 1 ] \* PicSampleC[ Clip3( 0, PicWidthInHalf − 1, xRef ), yPosRL ] +   
 fC[ xPhase, 2 ] \* PicSampleC[ Clip3( 0, PicWidthInHalf − 1, xRef + 1 ), yPosRL ] +  
 fC[ xPhase, 3 ] \* PicSampleC[ Clip3( 0, PicWidthInHalf − 1, xRef + 2 ), yPosRL ]

1. The upsampled chroma sample value ChromaSample is derived as follows:

ChromaSample = Clip3( 0, 1023, ( fC[ yPhase, 0 ] \* tempArray[ 0 ] +  
 fC[ yPhase, 1 ] \* tempArray[ 1 ] +  
 fC[ yPhase, 2 ] \* tempArray[ 2 ] +   
 fC[ yPhase, 3 ] \* tempArray[ 3 ] + 128 )  >>  8 )

## Inverse chroma quantization process

Input to this process is a variable BeforeInverseQuant.

Output of this process is a variable AfterInverseQuant.

The variable AfterInverseQuant is derived as follows:

AfterInverseQuant = Clip3(-0.5, 0,5, (BeforeInverseQuant – 512)  896)

## Process for calculating luminance

Inputs to this process are three variables RedVal, GreenVal and BlueVal.

Output of this process is one variable LuminanceVal.

The variable LuminanceVal is derived as follows:

LuminanceVal = Clip3(0.0, 1.0, wR\* RedVal – wG \* GreenVal – wB \* BlueVal)

NOTE 1 – The equation for calculating the luminance is functionally identical to the one described in [BT.2020].

## Process for luma adjustment

Inputs to this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array YTarget of samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleCb of samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleCr of samples in the range of 0.0 to 1.0, inclusive,

Outputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of luma samples,

For each xP = 0.. PicWidthInSamples − 1, yP = 0.. PicHeightInSamples – 1 the following ordered steps are performed:

* The variable YpCandidateLow is set equal to 64
* The variable YpCandidateHigh is set equal to 940
* While YpCandidateLow+1 is not equal to YpCandidateHigh the following ordered steps are performed:

1. The YpCandidateMid is set equal to ( YpCandidateLow + YpCandidateHigh ) >> 1
2. The variable Ypn is derived as follows:
   * Ypn = (YpCandidateMid – 64) 
3. YTest is derived by invoking the process for calculating luminance from luma and chroma as specified in clause 8.10, with Ypn, PicSampleCb[ xP ][ yP ] and PicSampleCr[ xP ][ yP ] as input.
4. If YTest is less than YTarget[ xP ][ yP ] then YCandidateLow is set equal to YCandidateMid, otherwise (YTest is greater than or equal to YTarget[ xP ][ yP ]) then YCandidateHigh is set equal to YCandidateMid.

* The variable YTestLow is derived by invoking the process for calculating luminance from luma and chroma as specified in clause 8.10, with ((YpCandidateMid – 64), PicSampleCb[ xP ][ yP ] and PicSampleCr[ xP ][ yP ] as input.
* The variable YTestHigh is derived by invoking the process for calculating luminance from luma and chroma as specified in clause 8.10, with ((YpCandidateMid – 63), PicSampleCb[ xP ][ yP ] and PicSampleCr[ xP ][ yP ] as input.
* If Abs(YTestLow – YTarget[ xP ][ yP ]) < Abs(YTestHigh – YTarget[ xP ][ yP ]), PicSampleL[ xP ][ yP ] is set equal to YCandidateLow. Otherwise it is set equal to YCandidateLow+1.

## Process for calculating luminance from luma and chroma

Inputs to this process are three variables Ypn, CbVal and CrVal.

Output of this process is one variable YLinear.

The variable YLinear is derived by applying the following ordered steps:

1. The variables Redp, Greenp, Bluep are derived by invoking the process for calculating RGB as specified in clause 8.11, with Ypn, CbVal and CrVal as input, respectively.
2. The variables RedLinear, GreenLinear and BlueLinear are derived by invoking the process for applying the transfer function as specified in clause 8.12, with Redp, Greenp and Bluep as input, respectively.
3. YLinear is derived by invoking the process for calculating luminance as specified in clause 8.7, with RedLinear, GreenLinear and BlueLinear as input.

## Process for calculating RGB

Inputs to this process are three variables Yp, Cbp, Crp.

Output of this process are three variables Redp, Greenp, Bluep.

The variables Redp, Greenp, Bluep are derived as follows:

* + Redp = Clip3(0.0, 1.0, Yp + a13 \* Crp)
  + Greenp = Clip3(0.0, 1.0, Yp – a22 \* Cbp – a23 \* Crp)
  + Bluep = Clip3(0.0, 1.0, Yp + a32 \* Cbp)

## Process for applying the transfer function

Input to this process is one variable NonLinearVal.

Output of this process is one variable LinearVal.

The variable LinearVal is derived as follows:



NOTE 1 – The transfer function in this document is functionally identical to the EOTF in [ST 2084].

# Encoding process

Inputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of integer luma samples in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCb of integer chroma samples of the component Cb in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCr of integer chroma samples of the component Cr in the range of 0 to 1023, inclusive.

In addition, mastering display colour volume information is optional input to the encoding process.

Output of this process is a bitstream.

This document does not provide a detailed description of the encoding process or the bitstream format. Clause 9.1 provides information on suitable settings of syntax elements when creating bitstreams in accordance with [HEVC]. Clause 9.2 provides information on suitable settings of syntax elements when creating bitstreams in accordance with [AVC].

If mastering display colour volume information is input to the encoder it is recommended that Mastering display colour volume SEI messages are included at least at each Random Access Point Access Unit (RAPAU). The information provided in the mastering display colour volume information SEI message shall apply until, but not necessarily including, the next RAPAU. If multiple mastering display colour volume SEI messages are included in the bitstream between the start of two RAPAUs then those SEI messages shall have the same content. Table 2 shows an example of what values the Master display colour volume SEI message would contain in case the mastering display uses P3 colour primaries [RP 431-2], D65 white point and luminance range of 0 cd/m2 to 2000 cd/m2, inclusive.

Table 2 – Example Mastering display colour volume SEI message representing P3 colour primaries [RP 431-2], D65 white point and luminace range of 0 cd/m2 to 2000 cd/m2, inclusive.

|  |  |
| --- | --- |
| **Syntax element** | **Example value** |
| **display\_primaries\_x[0]** | 13250 |
| **display\_primaries\_y[0]** | 34500 |
| **display\_primaries\_x[1]** | 7500 |
| **display\_primaries\_y[1]** | 3000 |
| **display\_primaries\_x[2]** | 34000 |
| **display\_primaries\_y[2]** | 16000 |
| **white\_point\_x** | 15635 |
| **white\_point\_y** | 16450 |
| **max\_display\_mastering\_luminance** | 20000000 |
| **min\_display\_mastering\_luminance** | 0 |

## HEVC encoding

When creating the HEVC bitstream it is recommended to set the syntax elements listed in Table 3 to the values listed in Table 1 in each Sequence Parameter Set in the bitstream.

Table 3 – Recommended settings for HEVC encoding

|  |  |
| --- | --- |
| **Syntax element** | **Recommended value** |
| **general\_profile\_space** | 0 |
| **general\_profile\_idc** | 2 |
| **video\_full\_range\_flag** | 0 |
| **colour\_primaries** | 9 |
| **transfer\_characteristics** | 16 |
| **matrix\_coeffs** | 9 |
| **chroma\_sample\_loc\_type\_top\_field** | 2 |
| **chroma\_sample\_loc\_type\_bottom\_field** | 2 |

For HDR content represented with the colour primaries of [BT.2020] and the transfer function of [ST 2084], the video characteristics is typically different compared to the video characteristics of SDR content represented with [BT.709] colour primaries and [BT.709] transfer function. It is recommended to adjust the bit-distribution between chroma and luma for example by setting chroma QP offset (controlled by the syntax elements pps\_cb\_qp\_offset, slice\_cb\_qp\_offset, pps\_cr\_qp\_offset and slice\_cr\_qp\_offset) such that a small negative offset is used for low luma QP values and a large negative offset is used for high luma QP values. It is further recommended to adjust the bit-distribution between dark samples and bright samples for example by setting delta QP (controlled by the syntax elements cu\_qp\_delta\_abs and cu\_qp\_delta\_sign\_flag) such that blocks with a high averge luma value are assigned lower QP than blocks with a low average luma value.

## AVC encoding

When creating the AVC bitstream it is recommended to set the syntax elements listed in Table 4 to the values listed in Table 4 in each Sequence Parameter Set in the bitstream.

Table 4 – Recommended settings for AVC encoding

|  |  |
| --- | --- |
| **Syntax element** | **Recommended value** |
| **profile\_idc** | 110 |
| **video\_full\_range\_flag** | 0 |
| **colour\_primaries** | 9 |
| **transfer\_characteristics** | 16 |
| **matrix\_coeffs** | 9 |
| **chroma\_sample\_loc\_type\_top\_field** | 2 |
| **chroma\_sample\_loc\_type\_bottom\_field** | 2 |

For HDR content represented with the colour primaries of [BT.2020] and the transfer function of [ST 2084], the video characteristics is typically different compared to the video characteristics of SDR content represented with [BT.709] colour primaries and [BT.709] transfer function. It is recommended to adjust the bit-distribution between chroma and luma for example by setting chroma QP offset (controlled by the syntax elements chroma\_qp\_index\_offset and second\_chroma\_qp\_index\_offset) such that a small negative offset is used for low luma QP values and a large negative offset is used for high luma QP values. It is further recommended to adjust the bit-distribution between dark samples and bright samples for example by setting delta QP (controlled by the syntax element mb\_qp\_delta) such that blocks with a high averge luma value are assigned lower QP than blocks with a low average luma value.

# Decoding process

Input to this process is a bitstream.

Outputs of this process process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of integer luma samples in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCb of integer chroma samples of the component Cb in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCr of integer chroma samples of the component Cr in the range of 0 to 1023, inclusive.

When the bitstream is an HEVC bitstream the decoding process in [HEVC] is performed with the sample values of each output picture iteratively assigned to PicSampleL, PicSampleCb, and PicSampleCr, respectively.

When the bitstream is an AVC bitstream the decoding process in [AVC] is performed with the sample values of each output picture iteratively assigned to PicSampleL, PicSampleCb, and PicSampleCr, respectively.

If the decoder contains an interface for output of mastering display colour volume information and the bitstream contains mastering display colour volume SEI messages it is recommended that the decoder outputs mastering display colour volume information synchronously to the first picture for which the SEI message applies.

# Post-decoding processing

Inputs to this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleDecL of integer luma samples, in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleDecCb of integer chroma samples of the component Cb, in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleDecCr of integer chroma samples of the component Cr, in the range of 0 to 1023, inclusive.

Outputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleOutR of floating point red samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleOutG of floating point green samples in the range of 0.0 to 1.0, inclusive,

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleOutB of floating point blue samples in the range of 0.0 to 1.0, inclusive,

A schematic overview of the post-decoding processing is illustrated in Figure 3. The input (Y’, Cb, Cr) corresponds to PicSampleDecL, PicSampleDecCb, and PicSampleCr, respectively. The output (R, G, B) corresponds to PicSampleOutR, PicSampleOutG, and PicSampleOutB, respectively.



Figure 3. Overview of the post-decoding processing. The box marked Inverse quant. constitutes inverse luma quantization and inverse chroma quantization. The box marked “Conversion to R’G’B’ ” is realized in this document by the process for calculating RGB.

The post-decoding process consists of the following ordered steps:

1. The upsampled chroma sample value PicSampleUpCb[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cb is derived by invoking the chroma upsampling process specified in clause 8.6 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleDecCb given as inputs.
2. The upsampled chroma sample value PicSampleUpCr[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cr is derived by invoking the chroma upsampling process specified in clause 8.6 with chroma sample location ( xPC, yPC ) and the chroma sample array PicSampleDecCr given as inputs.
3. The upsampled chroma sample value PicSampleUpCb[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cb is modified by invoking the inverse chroma quantization process specified in clause 8.7 with PicSampleUpCb[ xPC ][ yPC ] as input.
4. The upsampled chroma sample value PicSampleUpCr[ xPC ][ yPC ], with xPC = 0..PicWidthInSamples − 1, yPC = 0..PicHeightInSamples − 1, of the chroma component Cr is modified by invoking the inverse chroma quantization process specified in clause 8.7 with PicSampleUpCr[ xPC ][ yPC ] as input.
5. The luma sample value PicSampleDecY[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1, of the luma component Y is modified by invoking the inverse luma quantization process specified in clause 11.1 with PicSampleDecY[ xP ][ yP ] as input.
6. Three (PicWidthInSamples)x(PicHeightInSamples) arrays of non-linear samples PicSampleNonLinearR, PicSampleNonLinearG and PicSampleNonLinearB are derived by invoking the process for calculating RGB specified in clause 8.11, with PicSampleDecY[ xP ][ yP ], PicSampleUpCb[ xP ][ yP ] and PicSampleUpCr[ xP ][ yP ] as input and with the output assigned to PicSampleNonLinearR[ xP ][ yP ], PicSampleNonLinearG[ xP ][ yP ] and PicSampleNonLinearB[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
7. The output samples PicSampleOutR is derived by invoking the process for applying the transfer function specified in clause 8.12, with PicSampleNonLinearR[ xP ][ yP ] as input and with the output assigned to PicSampleOutR[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
8. The output samples PicSampleOutG is derived by invoking the process for applying the transfer function specified in clause 8.12, with PicSampleNonLinearG[ xP ][ yP ] as input and with the output assigned to PicSampleOutG[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.
9. The output samples PicSampleOutB is derived by invoking the process for applying the transfer function specified in clause 8.12, with PicSampleNonLinearB[ xP ][ yP ] as input and with the output assigned to PicSampleOutB[ xP ][ yP ], with xP = 0..PicWidthInSamples − 1, yP = 0..PicHeightInSamples − 1.

## Inverse luma quantization process

Input to this process is a variable BeforeInverseQuant.

Output of this process is a variable AfterInverseQuant.

The variable AfterInverseQuant is derived as follows:

AfterInverseQuant = Clip3(0.0, 1.0, (BeforeInverseQuant – 64)  876)