|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  24th Meeting: Geneva, CH, 26 May – 1 June 2016 | Document: JCTVC-X1017 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | Draft 2 of Conversion and Coding Practices for HDR/WCG Y′CbCr 4:2:0 Video with PQ Transfer Characteristics | | |
| *Status:* | Output document | | |
| *Purpose:* | Output document | | |
| *Author(s) or Contact(s):* | Jonatan Samuelsson  Chad Fogg  Andrey Norkin  Andrew Segall  Jacob Ström  Gary Sullivan  Alexis Tourapis | Email: | jonatan.samuelsson@ericsson.com |
| *Source:* |  | | |

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

# Abstract

This document contains a draft of the conversion and coding practices technical report on HDR/WCG video. Summary

This document provides guidance on processing of High Dynamic Range (HDR) and Wide Colour Gamut (WCG) video. The purpose of this document is to provide a set of publicly referencable recommended guidelines for the operation of AVC or HEVC systems adapted for compressing HDR/WCG video. This document includes a description of processing steps for converting linear light, 4:4:4, RGB video signals into Non Constant Luminance (NCL) Y′CbCr video signals that use the ST 2084 Perceptual Quantizer (PQ) transfer function. Although the focus of this document is primarily on 4:2:0, Y′CbCr, 10 bit representations, these guidelines may also apply to other higher bitdepth and/or colour format representations such as 4:4:4, Y′CbCr, 12 bit video. In addition, this document provides some high level recommendations for compressing these signals using either the AVC or HEVC video coding standards. Description of post-decoding processing steps for converting these NCL Y′CbCr signals back to a linear light, 4:4:4, RGB representation is also included.

**CONTENTS**

| Page |
| --- |
| Abstract i  1 Scope 5  2 References 6  3 Definitions 6  4 Abbreviations and acronyms 7  5 Conventions 8  5.1 General 8  5.2 Arithmetic operators 8  5.3 Bit-wise operators 8  5.4 Assignment operators 9  5.5 Mathematical functions 9  5.6 Order of operations 9  6 Introduction 10  7 Pre-encoding process 11  7.1 Pre-encoding processes 12  7.1.1 Conversion from a linear to a non-linear light representation: RGB to R′G′B′ 12  7.1.2 Colour representation conversion: R′G′B′ to Non Constant Luminance Y′CbCr 13  7.1.3 Chroma down-conversion 14  7.1.4 Floating-point to fixed-point (narrow range) 10 bit conversion 15  7.2 Closed loop pre-encoding conversion – Luma Adjustment 16  7.2.1 Luma adjustment – Iterative approach 17  7.2.2 Luma adjustment – closed form solution 19  8 Encoding process 21  8.1 Perceptual Luma Quantization 21  8.1.1 Brightness Dependent Adaptive Quantization – An Example 22  8.2 Chroma QP offset 23  8.2.1 Example of Chroma QP offset settings. 24  8.3 Other encoding aspects 25  8.4 HEVC encoding 25  8.5 AVC encoding 26  9 Decoding process 26  10 Post-decoding processes 27  10.1 Post-decoding processes 27  10.1.1 Conversion from a fixed-point to a floating-point representation 27  10.1.2 Chroma up-sampling 27  10.1.3 Colour representation conversion: Non Constant Luminance Y′CbCr to R′G′B ′ 28  10.1.4 Conversion from a non-linear to a linear light representation: R′G′B′ to RGB 28  Annex A. VUI Supplemental enhancement information (SEI) messages. 30  A.1 Mastering display colour volume SEI message 30  A.2 Content Light Level information SEI message 30  Annex B. Model A and Model B configuration settings. 31  B.1 Model A configuration settings 31  B.2 Model B configuration settings 31 |

Conversion and Coding Practices for HDR/WCG Video

# Scope

High Dynamic Range (HDR) video is a type of video in which the sample values span a larger luminance range than conventional Standard Dynamic Range (SDR) video, for example reaching significantly beyond the lumince ranges used in BT.2350. HDR video has the ability to provide an enhanced viewer experience and more accurately reproduce scenes that at the same time includes dark areas and bright highlights such as emmisive light sources and reflections. Wide Colour Gamut (WCG) video, on the other hand, is video characterized by a wider spectrum of colors that has been commonly available in conventional video. Recent advances in capture and display technology have enabled consumer distribution of HDR and WCG content. However, given the characteristics of such content, special considerations may need to be made in terms of both processing and compression, compared to conventional content.

This document provides a set of recommended guidelines on processing of consumer distribution HDR/WCG video including conversions steps for going from a linear light RGB representation with BT.2020 colour primaries to a 10-bit, narrow range, ST 2084 (PQ), 4:2:0, non-constant luminance Y′CbCr representation. These guidelines may also apply to other higher bitdepth and/or colour format representations such as 12 bit 4:4:4. The scope of this document is illustrated in Figure 1‑1.



Figure 1‑1 – 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, however metadata generated during the content preparation may be passed through the encoder-decoder chain that significantly affect display adaptation. The content preparation step may include filtering and image enhancement processing such as de-noising, colour correction, and sharpening filtering among others. Such methods are deliberately not described in this document. The processing steps described in this document are made available for reference only and the document 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. This document’s intention is to provide some guidelines for operating an HDR/WCG video system that is constrained, but not restricted, to code a 10-bit, ST 2084 [11], 4:2:0, non-constant luminance Y′CbCr signal representation [6]. This configuration is aligned with the HDR10 media profile defined in [8], the interface defined in [9], and the restrictions in [15]. 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 and after the HDR/WCG system. This document does not account for when different viewing environments are used before and after the HDR/WCG system. In particular, display adaptation, such as the techniques described in the SMPTE ST 2094 [7] standards, are not considered in this document. BT.2390 [7] contains additional information on viewing environments and examples of parameters that may be appropriate to apply for a practical HDR systems.

[Ed. (JS+AT): Is the paragraph above sufficient? Is any further clarification needed?]

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

The recommended practices described in this document are based on investigative testing, conducted within the JCT and its parent organizations (ISO/IEC JCT1 SC29 WG11 (MPEG) and ITU-T SG 16 Q.6 (VCEG)), for HEVC coding of HDR/WCG content available at the time.

# References

[Ed. (AT): Should we move this to the end and rename this as bibliography? That seems to be the common practice with other specs (see HEVC and CICP).]

1. Recommendation ITU-T H.264 (V10) (2015), *Advanced video coding for generic audiovisual services*. Also available as ISO/IEC 14496-10.
2. Recommendation ITU-T H.265 (V3) (2015), *High Efficiency Video Coding.* Also available as ISO/IEC 23008-2.
3. Recommendation ITU-R BT.709-6 (2015), *Parameter values for the HDTV standards for production and international programme exchange.*
4. Recommendation ITU-R BT.1886 (2011), *Reference electro-optical transfer function for flat panel displays used in HDTV studio production*.
5. Recommendation ITU-R BT.2020-2 (2015), *Parameter values for ultra-high definition television systems for production and international programme exchange*.
6. Recommendation ITU-R BT.2100 (2016), *Image parameter values for high dynamic range television for use in production and international programme exchange.*
7. Report ITU-R BT.2390 (2016), *High dynamic range television for production and international programme exchange.*
8. DECE, *Common File Format & Media Formats Specification Version 2.1.*
9. CTA 861-G, *A DTV Profile for Uncompressed High Speed Digital Interfaces.*
10. SMPTE Standard RP 431-2 (2011), *D-cinema Quality - Reference Projector and Environment*.
11. SMPTE Standard ST 2084 (2014), *High Dynamic Range Electro-Optical Transfer Function for Mastering Reference Display*.
12. SMPTE Standard ST 2094 (2016), *Dynamic Metadata for Color Transforms of HDR and WCG Images*.
13. C. Poynton, A Technical Introduction to Digital Video (New York: John Wiley & Sons, 1996).
14. J. Ström, J. Samuelsson and K. Dovstam, ”Luma Adjustment for High Dynamic Range Video”, Proceedings of the IEEE Data Compression Conference (DCC), Snowbird, 2016.
15. Blu-Ray Disc Association, “BD-ROM: Audio Visual Application Format Specifications version 3” (July 2015)
16. A. Luthra, E. Francois, W. Husak, “Call for Evidence (CfE) for HDR and WCG Video Coding”, MPEG2014/N15083, 110th MPEG Meeting, Geneva, 2015
17. V. Baroncini, K. Andersson, A. K. Ramasubramonian, G. J. Sullivan, “Verification Test Report for HDR/WCG Video Coding Using HEVC Main 10 Profile”, 2016

# Definitions

This document defines the following terms:

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

NOTE 1 – In this document the pre-encoding process is assumed to operate on HDR/WCG video content that has been prepared using a hypothetical reference viewing environment as shown in Figure 1‑1. The content preparation step may contain processing such as applying a system gamma, in which the HDR/WCG video is converted from one linear representation (corresponding to the scene) to another linear representation (corresponding to the display). In systems were no such system gamma is applied in the content preparation step, the process of converting from a linear representation (corresponding to the scene) to a non-linear representation is typically called Opto-Electrical Transfer Function.

* 1. **Narrow range:** A range in a fixed-point (integer) representation that does not span the full range of values that could be expressed with that bit depth. In this document the range from 64 (black) to 940 (peak white) is used for Y′, and the range from 64 to 960 is used for Cb and Cr for 10 bit representations, which is aligned with [6]. Narrow range is in some applications called by synonyms such as: “limited range”, “video range”, “legal range”, “SMPTE range” or “standard range.”
  2. **Full range:** a range in a fixed-point (integer) representation that spans the full range of values that could be expressed with that bit depth. For 10-bit signals, black corresponds to code 0 and peak white corresponds to code 1023 [6].
  3. **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.
  4. **Transfer function:** The function used in the post-decoding process to convert from a non-linear representation to a 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

CL Constant Luminance

EOTF Electro-Optical Transfer Function

FIR Finite Impulse Response

HD High Definition

HDR High Dynamic Range

HEVC High Efficiency Video Coding

HRVE Hypothetical Reference Viewing Environment

HVS Human Visual System

NCL Non Constant Luminance

PQ Perceptual Quantizer as defined in BT.2100.

QP Quantization Parameter

RAPAU Random Access Point Access Unit

RGB Colour System using Red, Green, and Blue components

SDR Standard Dynamic Range

SEI Supplemental Enhancement Information

OETF Opto-Electrical Transfer Function

TF Transfer Function

VUI Video Usability Information

WCG Wide Colour Gamut

XYZ The CIE 1931 colour space. Y corresponds to the luminance signal.

Y′CbCr Colour space representation commonly used for video/image distribution as a way of encoding RGB information. Also commonly expressed as YCbCr or Y′C′BC′R. The relationship between Y′CbCr and RGB is dictated by certain signal parameters, such as colour primaries, transfer characteristics, and matrix coefficients. Unlike the (constant luminance) Y component in the XYZ representation, Y′ in this representation might not be representing the same quantity. Y′ is commonly referred to as “luma”. Cb and Cr are commonly referred to as “chroma”.

# Conventions

## General

NOTE – The mathematical operators used in this document are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

## Arithmetic operators

The following arithmetic operators are defined as follows:

|  |  |
| --- | --- |
|  | Addition |
| − | Subtraction (as a two-argument operator) or negation (as a unary prefix operator) |
| \* | Multiplication, including matrix multiplication |
| xy | Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation. |
| / | 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. |
|  | Used to denote division in mathematical equations where no truncation or rounding is intended. |
|  | The summation of f( i ) with i taking all integer values from x up to and including y. |
| x % y | Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0. |

## Bit-wise operators

The following bit-wise operators are defined as follows:

|  |  |
| --- | --- |
| & | Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| | | Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| ^ | Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| x >> y | Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation. |
| x << y | Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0. |

## Assignment operators

The following arithmetic operators are defined as follows:

|  |  |
| --- | --- |
|  | Assignment operator |
|   | Increment, i.e., x  is equivalent to x  x  1; when used in an array index, evaluates to the value of the variable prior to the increment operation. |
| − − | Decrement, i.e., x− − is equivalent to x  x − 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation. |
| += | Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3). |
| −= | Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3). |

## Mathematical functions

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

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

## Order of operations

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– Operations of a higher precedence are evaluated before any operation of a lower precedence.

– Operations of the same precedence are evaluated sequentially from left to right.

Table 5‑1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this document is the same as used in the C programming language.

Table 5‑1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

|  |
| --- |
| **Operations (with operands x, y, and z)** |
| "x++", "x− −" |
| "!x", "−x" (as a unary prefix operator) |
| "xy" |
| "x \* y", "x / y", "x  y""", "x % y" |
| "x + y", "x − y" (as a two-argument operator), "" |
| "x  <<  y", "x  >>  y" |
| "x < y", "x  <=  y", "x > y", "x  >=  y" |
| "x  = =  y", "x  !=  y" |
| "x & y" |
| "x | y" |
| "x  &&  y" |
| "x  | |  y" |
| "x ? y : z" |
| "x..y" |
| "x = y", "x  +=  y", "x  −=  y" |

# Introduction

The HDR/WCG System described in this document consists of four major stages:

* a pre-encoding stage consisting of several pre-processing processes (Section 7),
* an encoding stage (Section 8),
* a decoding stage (Section 9), and
* a post-decoding stage, also consisting of several post-processing processes (Section 10).

These four stages are applied sequentially with the output of one stage being used as input to the next stage according to the above-mentioned order.

It is assumed that both the input to and the output of the HDR/WCG System are 4:4:4, linear light, floating-point signals, in an RGB colour representation using the same colour primaries. The output signal is targeted to resemble the input video signal as closely as possible. Other video formats can be used as input to the HDR/WCG System by first converting them to the above defined input signal representation. The HDR/WCG System described in this document is, in practice, a system for both HDR and WCG video since it is assumed that the input video is represented with colour primaries in accordance with BT.2020 [5] and BT.2100 [6].

Two different models, Model A and Model B, are described in this document for the Pre-encoding and Encoding processes, which correspond to the two different configuration settings described in Annex B. For the Decoding process and Post-decoding processes, a single model is described.

|Ed. (AT): Consider to use more descriptive names of the models such as Baseline and Main]

The primary purpose of the pre-encoding process is to convert the video input from its 4:4:4 RGB linear light, floating-point signal representation to a signal that is suitable for a video encoder. The conversion to a non-linear representation is performed in an attempt to exploit the characteristics of the human visual system (HVS) that could allow the re-quantization of the signal at a limited precision.

NOTE 1 – For a fixed-point linear HDR/WCG video representation, approximately a 28-bit integer representation would be required to avoid introducing visible quantization/banding errors due to the 28 f-stop linear light dynamic range (0.00005 to 10,000 nits) that is spanned by the 10-bit ST 2084 transfer function. In practice, the input to the HDR/WCG System will typically be in a non-linear representation that could either have to be first converted to linear light data or be directly converted to ST 2084.

It is assumed that encoding and decoding is performed in a 4:2:0, 10-bit representation. An encoder is expected to make the best use of the encoding tools available according to a particular specification, profile, and level, given also the characteristics of the content and the limitations of the intended application and implementation. In particular, different encoding algorithms, such as algorithms for motion estimation, mode decision, rate allocation, rate control, and post-filtering control among other aspects, may have to be considered when encoding HDR/WCG material, in a given representation, compared to SDR material. The decoding process on the other hand is fully described in the respective HEVC and AVC decoder standards, where a decoder must fully comply to the intended profile and level to output precisely reconstructed video samples from a given input bistream. This is done according to a deterministic decoding process, nominally over a time window indicated in the bitstream.

NOTE 2 – The focus of this document is on consumer and direct-to-home applications, which are expected to be, at least in the near term future, in a 4:2:0 10-bit format. Processes similar to the ones described in this document can be used for conversion and compression of other formats, such as 4:2:2 and 4:4:4 chroma formats and/or video with bit-depth higher than 10.

The steps in the post-decoding process are aligned with what is commonly referred to as the non-constant luminance representation (NCL) in which colour conversion, to R′G′B′, is performed prior to applying the transfer function to produce linear RGB.

There is no specific or minimum bitdepth required for performing the operations described in the pre-encoding process and the post-encoding process. 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 operations with precision lower than 64 bits. Using too low precision should be avoided since it could potentially lead to loss of precision in the output video. The input to the encoding step and the output of the decoding step are 10-bit integer representations.

# Pre-encoding process

The pre-encoding process described in this document includes the following components:

1. a conversion component from a linear data representation to a non-linear data representation using the appropriate transfer function,
2. a colour format conversion component that converts data to the Non-Constant Luminance Y′CbCr representation,
3. a conversion component that converts a floating-point representation to a fixed-point representation (i.e. 10 bits),and
4. a chroma down-conversion component that converts data from 4:4:4 to 4:2:0.

NOTE 3 – Scaling may also be a vital component of the pre-encoding process, for example if the target system requires a particular image resolution be delivered to the decoder. It may be desirable, for example, to rescale a source from a 1920x1080 to a 3840x2160 resolution, or vice versa. Scaling is not included in the scope of this document. However, it is common practice that, for improved performance even for SDR material, rescaling is performed in either the linear domain or at least using the R′G′B′ representation, instead of using the Non-Constant Luminance Y′CbCr representation.

Figure 7‑1 presents a diagram of how these components are combined in Model A to generate the desirable outcome, in a conventional manner. In this model, all blocks work independently, whereas chroma subsampling is performed using fixed-point arithmetic and at the same precision as the target outcome.



Figure 7‑1 – Conventional pre-encoding process system diagram.

Although this combination could be the most appropriate for some implementations, it has several limitations that can affect both its performance and implementation complexity. In this section the pre-encoding process components are first introduced in more detail, and then the alternative configuration corresponding to Model B, is presented in section 7.2. Recommendations on how to best utilize some of the conversion components are also presented.

## Pre-encoding processes

### Conversion from a linear to a non-linear light representation: RGB to R′G′B′

Conversion from a linear to a non-linear light representation is performed using an inverse transfer function, or as is commonly referred to other specifications, an opto-electrical transfer function (OETF). In this document, the ST.2084 transfer function, also referred to as PQ, is used.

More specifically, the non-linear light representation V of a linear light intensity signal Lo, that takes values normalized to the range [0, 1], can be computed as:

V = iTF(Lo) = ( ( c1 + c2 \* Lon ) ÷ ( 1 + c3 \* Lon ) )m (7‑1)

where c1, c2, c3, m, and n are constants, which are defined as follows:

c1 = c3 − c2 + 1 = 3424 ÷ 4096 = 0.835 937 5 (7‑2)

c2 = 2413 ÷ 128 = 18.851 562 5 (7‑3)

c3 = 299 ÷ 16 = 18.687 5 (7‑4)

m = 2523 ÷ 32 = 78.843 75 (7‑5)

n = 1305 ÷ 8192 = 0.159 301 757 812 5 (7‑6)

The peak value of 1 for Lo is ordinarily intended to correspond to an intensity level of 10 000 candelas per square metre, while the value of 0 for Lo is ordinarily intended to correspond to an intensity level of 0 candelas per square metre. The behaviour of the inverse ST 2084 [11] transfer function in relationship to the BT.709 [3][5] and the inverse of the BT.1886 [4] transfer functions is shown in Figure 7‑2.

NOTE  – It should be noted that a direct comparison of the ST 2084 transfer function with BT.709 might not be appropriate since BT.709 may assume the use of a system gamma during decoding.



Figure 7‑2 – Behaviour of the inverse of ST 2084 (PQ), the inverse of BT.1886, and BT.709 transfer functions.

This process is applied to all R, G, and B linear light samples, where each component is a number between 0.0 (representing no light) and 1.0 (representing 10 000 candelas per square metre). This results in their non-linear counterparts R′, G′, and B′ as follows.

R′ = iTF( R ) (7‑7)

G′ = iTF( G ) (7‑8)

B′ = iTF( B ) (7‑9)

Although it is, in general, recommended to perform this conversion process using Equation 7‑1 directly, this, however, may not be possible in some implementations given the complexity of the computation. Instead, look-up tables (LUT) may be preferred. Due to the characteristics of the conversion and the desire to achieve high precision for both low/dark and high values, it is highly recommended that, in such scenario, a non-uniformly indexed LUT interpolator is used. Such schemes can achieve relatively high accuracy/minimum approximation error for the conversion, while achieving considerable memory savings.

### Colour representation conversion: R′G′B′ to Non Constant Luminance Y′CbCr

Conversion from the R′G′B′ to the Non Constant Luminance Y′CbCr representation is commonly performed using a 3x3 matrix conversion process of the form:

(7‑10)

where *wYR*, *wYG*, *wYB*, *wCbR*, *wCbG*, *wCbB*, *wCbR*, *wCbG*, and *wCbB* are constants. The values for *wYR*, *wYG*, and *wYB*, are set to exactly the same values used to convert R, G, and B data to the CIE 1931 Y (luminance) signal. For BT.2020 colour primaries these are defined as follows:

(7‑11)

(7‑12)

(7‑13)

NOTE  – These constants are commonly specified using precision limited to 16-bits or less. It may be desirable and beneficial that these are used with a higher precision representation, such as 64-bit float, by computing the appropriate parameters using the primary coordinates.

On the other hand, the values of the constants *wCbR*, *wCbG*, *wCbB*, *wCrR*, *wCrG*, and *wCrB* are computed in a manner that the resulting Cb and Cr components are always within the [-0.5, 0.5] range. This essentially results in the following values:

(7‑14)

(7‑15)

(7‑16)

(7‑17)

(7‑18)

(7‑19)

NOTE  – In some implementations, *wCbR*, *wCbG*, *wCrG*, and *wCrB* may be further restricted to a fewer number of decimal points. In particular, values *wCbR* = -0.13963, *wCbG* = -0.36370, *wCrG* = -0.459786, and *wCrB* = -0.040214 may be used. However, using the above specified higher precision values may be beneficial.

NOTE 7 – The models described in Annex B use 6 decimal point precision.

An alternative method to perform the same conversion process is presented in [5][6], where the chroma components are computed after the conversion of the luma component according to equation 7‑10 as follows:

(7‑20)

(7‑21)

with and .

This can be seen as equivalent to the matrix presented in equation (7-10).

The inverse process, i.e. converting Y′, Cb, and Cr data back to R′, G′, and B′ data, is specified in Section 10.1.3.

### Chroma down-conversion

Converting the HDR/WCG video data from a 4:4:4 representation to a 4:2:0 representation nominally involves filtering and down-converting/subsampling the two chroma planes in both the horizontal and vertical directions, though it is possible to apply more complex methods that preserve edges and thus reduce the impact of interpolated colour values that did not exist in the local neighborhood of the original 4:4:4 pixel. It is also a requirement, according to both [5][6], that the resulting chroma samples are co-sited with those of luma at even horizontal and vertical positions (Figure 7‑3) where the first respective sample and line are counted starting from zero.



Figure 7‑3 – Chroma and Luma sample location relationship.

It is anticipated that a considerable amount of consumer electronics conversion systems would use 2-D separable finite impulse response (FIR) linear filters for low-pass filtering the chroma data before subsampling (2:1 decimation step). Such filters would basically be of the form:

(7‑22)

where x[n] is the input chroma signal, y[n] is the filtered output chroma signal, (2 \* N) corresponds to the filter order or, equivalently, (2 \* N + 1) corresponds to the number of taps of the filter, and bi corresponds to the coefficient of the filter at position i.

It has been observed that, especially due to the step transfer slope characteristics and quantization behaviour of ST.2084, special caution needs to be made when selecting the filter coefficients of such a resampling filter, in order to mitigate chroma leakage. Conventional filters, such as linear filters, that are commonly used for down-conversion of SDR chroma signals may potentially result in visual artefacts when applied to HDR/WCG signals. This document however only describes linear FIR filters. In particular, the following FIR filters may result in less objectionable artefacts. Such filters can be utilized for both vertical and horizontal filtering of the chroma samples. Model A and Model B of Annex B both uses Filter f0:

Table 7‑1 – Suggested filters for chroma downsampling

|  |  |  |  |
| --- | --- | --- | --- |
| **Filter** | **Filter coefficients** | | |
| b-1 | b0 | b1 |
| f0 | 1/8 | 6/8 | 1/8 |
| f1 | 1/4 | 2/4 | 1/4 |

The characteristics (magnitude and phase) of these filters are shown in Figure 7‑4. Filter f1 has a stronger attenuation that is equal to -6dB at 0.5π rad/s, whereas filter f0 could potentially cause some aliasing artifacts due to a significant amount of energy left in its stop-band. However, filter f0 may help in reducing chroma leakage since edges passed through this filter could output less severe transients.

|  |
| --- |
|  |

Figure 7‑4 – Frequency response of filters f0 and f1 for chroma downsampling

[Ed. (JS): Update Figure to say f0 and f1.]

The upconversion process, i.e. from a 4:2:0 representation back to a 4:4:4 representation, is discussed in section 10.1.2.

### Floating-point to fixed-point (narrow range) 10 bit conversion

A key component of the pre-encoding process is the conversion from a floating-point to a fixed-point, narrow range, 10-bit representation. This process is essentially a quantization step that would introduce some distortion. In general, the conversion process can be expressed as:

(7‑23)

or equivelantly:

(7‑24)

[Ed. (AT): Adding definitions of digital values of Y, Cb and Cr could be useful for making the luma adjustment and quantization sections more clear.]

where is the floating-point representation of a particular component and D′ is the resulting quantized value at a precision of b bits. In this document, b = 10. The scale and offset constants depend on the target range (narrow versus full range video) and the component type (luma, chroma, or primary components). More specifically, for the narrow range NCL representation, the scale and offset for the luma component are set as:

(7‑24)

(7‑25)

(7‑27)

On the other hand, the fixed precision narrow range representation for the two chroma components can be computed as follows:

(7‑26)

(7‑27)

and

(7‑30)

(7‑31)

NOTE  – If a full range is desired for some applications, the recommendation in [6] is to use for all components, for the chroma components, and offset = 0 in all other cases. An alternative full range representation is used in MPEG-4 AVC/H.264 [1] and HEVC [2] for some other of applications using different transfer functions other than ST 2084 (PQ) or HLG.

Presented in Figure 7‑5: the mapping of non-normalized, gray (R=G=B) linear light values to a non-linear representation according to respective HDR (ST 2084) and SDR (gamma 2.4 for BT.709 / BT.2020, assuming the use of the BT.1886 forward transfer function during display).



Figure 7‑5 – Mapping of “gray” linear light values to quantized 8 (SDR only) and 10 bit (SDR and HDR) values

The inverse conversion process, i.e. converting from a fixed-point representation back to a floating-point representation, is discussed in detail in Section 10.1.1.

## Closed loop pre-encoding conversion – Luma Adjustment

As mentioned in section 7.1.3, chroma leakage may occur in the NCL representation, primarily due to chroma down‑sampling, potentially resulting in objectionable artifacts.

This section presents an alternative conversion method, which can considerably alleviate this problem. This method, which is pixel-wise independent, is called the Luma Adjustment method, and it is basically a closed loop conversion process where the impact of chroma down‑sampling, quantization, inverse quantization, and up‑sampling, is accounted for during the luma conversion process. An example schematic diagram of such a system is presented in Figure 7‑6. The Luma Adjustment method presented in section 7.2.1 is used in Model B of Annex B. An alternative approach that approximates the Luma Adjustment method, is presented in Section 7.2.2.



Figure 7‑6 – Example schematic diagram of a closed loop pre-encoding conversion system

### Luma adjustment – Iterative approach

In section 7.1.2 and Equation 7‑10, we presented the conversion process from R′G′B′ to the Y′CbCr NCL representation. Given this process the Cb and Cr components can be computed as follows:

(7‑28)

(7‑29)

If we first convert these two components to their target resolution, i.e. using the steps specified in Section 7.1.3, followed by conversion back to the original representation resolution, as presented in Section 10.1.2, we can have the opportunity to analyze the error introduced into the signal and potentially compensate for it. More specifically, performing quantization, downscaling (QD), and subsequently up-scaling and inverse quantization (IQU) onto these components would result into the reconstructed and components, which are defined as follows:

(7‑30)

(7‑31)

Luminance (Y), unlike luma (Y′), is computed given the linear R, G, and B component values using a formulation of the form:

(7‑32)

Since , this can be rewritten as:

(7‑33)

However, since the reconstructed and values will likely differ from the original values and values, the reconstructed , , and values will also differ from the original R′, G′, and B′ values. Therefore, the reconstructed luminance , which is equal to:

(7‑34)

will also differ from the original luminance Y. Using Equation 10‑2 it can be computed that:

. (7‑35)

Chroma component dependent factors can then be defined as follows:

(7‑36)

(7‑37)

(7‑38)

resulting in the following formulation for Y*rec*:

(7‑39)

The intent of the luma adjustment method is to try and locate the value of Y′ that would minimize distortion for Yrec compared to the original luminance value Y. Unfortunately, due to the non-linear characteristics of the transfer function, solving for Y′ given Y and the values of Crfactor, Gfactor, and Cbfactor is not a straightforward process. However, root-finding numerical methods, such as the bisection method, can be used instead. Performance, and more specifically the convergence speed and accuracy of these methods, is though considerably impacted by the selection of the initial interval as well as the computations performed during the search.

NOTE  – Alternative methods that try to approximate the impact on Y and/or methods that evaluate impact on other components have also been suggested and are currently under evaluation.

The target of the luma adjustment process is to minimize luminance distortion, which can be realized through the following ordered steps:

1. Calculate the luminance value Y from the original R, G, and B, e.g., using Equation 7‑32. This will be referred to as the value.
2. Convert the R, G, and B data to their R′, G′, and B′ representation.
3. Given the R′, G′, and B′ planes generate the Cb and Cr chroma planes.
4. Downscale and quantize the chroma planes to their target representation.
5. De-quantize and up-convert the chroma planes back to their original representation, i.e. and
6. Calculate Crfactor, Gfactor, and Cbfactor from and .
7. Given the reconstructed chroma planes, try to find for each luma position an appropriate Y′ value, i.e. bestY′, that would potentially result in a minimum distortion for a particular aspect of the signal, i.e. in this case minimum luminance distortion. The value of bestY′ at each luma position would be the value used for the encoding of the luma signal. In particular, bestY′ would be the solution to the equation:  
    (7‑40)

#### Bisection search

The Bisection search method is an iterative technique that is commonly used to derive the roots of an equation f(x) = 0. The function f is assumed to be continuous, and defined over an interval [a, b], where f(a) and f(b) need to have opposite signs. For this method to work, the behaviour of f(x) within this interval needs to also be monotonic (i.e. consistently increasing or decreasing). At each iteration of the search the interval is divided by two, i.e. at the midpoint c = (a + b) / 2). Then the value of the function f(c) is computed at this point. Depending on the value of f(c) and its relationship with f(a) and f(b) a new smaller interval is defined that satisfies the opposite sign condition. This search is repeated until a root is found, the interval is sufficiently small, or if a certain maximum number of iterations has been achieved.

This method can be used to find the bestY′ value for luma, as discussed in the previous section, in the following way:

Let be the value that represents the quantized representation of the luma component, as defined in Section 7.1.1. Furthermore, let be the dequantization function (Section 10.1.4), that maps the value of *x* back to its original representation and essentially a value between [0, 1]. Then, the narrow range, 10-bit representation can be computed as:

. (7‑41)

Now let be the function:

(7‑42)

The initial interval can be set as the entire range, i.e. [a, b] = [64, 940]. Given the characteristics of the ST 2084 transfer function it is expected that f(a) ≤ 0 and f(b) ≥ 0, which satisfy the bisection conditions. The value of f(*x*) at position *x* can be computed as *x* = (a + b) / 2 = 502 and adjust the interval can then be adjusted accordingly. If, for example, f(502) > 0 then the interval will be adjusted to [64,502]. The next evaluation point will then be the middle of this new interval, i.e. *x* = (64 + 502) / 2 = 283. At this point if now f(283) < 0 then the interval will be adjusted to [283,502]. The process can continue until either a value is found that satisfies f(x) = 0, or the interval is of the form [k, k+1], e.g. [343, 344]. In this case both cases can be evaluated and the one resulting in the smallest distortion for Y or iTF(Y) can be used.

A critical component of this method is the selection of the initial interval. The brute force approach is to use the entire valid range as the initial range, e.g. for the 10-bit narrow representation the range of [64, 940] as in the previous example. This, though, will require a number of iterations equal to the target bitdepth of the content to reach an interval of size one. However, using information about the original pixel and the chroma values , upper and lower bounds can be found for the value of bestY′, greatly reducing the size of the initial interval. This can have a direct impact on the number of computations performed, and thus the complexity of the process. For example, in the 10-bit case, the number of average iterations may be reduced from around ten to less than two. Three such bounds are preferably used. The first is described in [14], and uses the following definitions:

(7‑43)

(7‑44)

. (7‑45)

will always be in the interval The proof for this is out of the scope of this document, but is presented in [11]. The second bound uses the fact that all three variables R′bound, G′bound, and B′bound are smaller than 1, which leads to a tighter upper bound, i.e. . This makes use of the fact that the transfer function is convex [14]. The third bound relies on the fact that all three of the reconstructed colour components , , and cannot simultaneously be smaller (or all simultaneously larger) than the original colour components R′, G′, and B′for the computation of bestY′. By using the following definitions:

(7‑46)

(7‑47)

(7‑48)

it can be derived that bestY′ must be in the interval:

(7‑49)

By combining all three bounds together the minimum and maximum bounds for Y′ can be computed as:

(7‑50)

(7‑51)

Finally the initial interval [*a*, *b*] for x is calculated as and where is the inverse of (7‑41).

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| (a) | (b) | (c) |

Figure 7‑7  – Tone-mapped examples showing improvements of the luma adjustment method.   
a) Originals in 4:4:4. b) Traditional subsampling. c) Luma Adjustment based subsampling .

Figure 7‑7 shows an example of what the difference can be when performing traditional subsampling as described in Section 7.1.3 and using the luma component unchanged (Figure 7‑7b) compared to the luma adjustment method described in this section (Figure 7‑7c). These images were processed in the ITU-R BT.709 colour space to more easily demonstrate the differences. Since the printed medium cannot reproduce HDR images, tone-mapped versions are calculated using , where clip( ) clips to the interval [0, 255], =2.2, and c is an exposure parameter set to make the SDR image look similar to the HDR image. Note how the artifacts are clearly more visible in the traditional subsampling case compared to when using the Luma Adjustment method.

### Luma adjustment – closed form solution

The bisection search method described in the previous section, and for a 10 bit data representation has a worst case complexity of ten iterations. This might be a problem for certain implementations, and in particular hardware implementations. Quite often, hardware systems are designed by taking in account the worst-case scenario, and assume that the same number of processing steps are required for each block or sample. Even though the complexity of the bisection method could be further bounded by limiting the maximum number of iterations, it may be beneficial for such applications to use a closed-form solution that is able to determine an appropriate luma value in a single step.

A closed-form solution can be found as follows. First, downsampled chroma samples are obtained. Then, chroma is upsampled to the original (luma) resolution by applying a chosen upsampling filter. Then, for every pixel, the algorithm estimates a luma value Y' that, in combination with the upsampled and values, will result in a reconstructed pixel with colour coordinates {Rnew, Gnew, Bnew}. It is highly desirable that is as close as possible to the original linear light value according to a chosen distance metric.[Ed (AT): The use of coordinates seems not necessary here and is quite inconsistent with how the previous sections were written. They also do not seem to provide any useful information. I would suggest removing them from the text. Yes, I could assume that D may have been a distortion process that also takes in account x and y, but not in its current form, since and y are not passed as parameters to D.][Ed. (AT): We seem to be redefining new symbols for the same variables that were defined in the previous section. Lets reuse those. Rnew is equivalent to for example] The difference between the two RGB values is denoted as:

(7‑52)

Depending on the chosen distance metric *Diff* (.), different closed-form solutions to the optimization problem can be obtained. In the following, a solution based on the weighted sum of the differences of linear R, G, and B data is described.

In particular, the square of the sum of the weighted differences between the individual R, G, and B components can be computed as:

*D* = ( wR \* ( *R*new – *R*org) + wG \* (*G*new – *G*org) + wB \* (*B*new – *B*org) )2. (7‑53)

This is also equivalent to the equation

*D* = ( wR \* ( *f* (R'new) – *f* (R'org)) + wR \* (*f* (G'new)– *f* (G'org)) + wB \* ( *f* (B'new) – *f* (B'org)))2, (7‑54)

where *f* ( *X* )is the forward transfer function, for example the PQ transfer function. If weights wR, wG, and wB are set equal to the contribution of the linear R, G, and B components to the luminance component, this cost function would be minimizing the squared difference between the luminance values.

NOTE  – It is also possible to use other error functions, such as the sum of the R, G, and B component squared errors, which may result in different solutions.

Finding a closed-form solution for Y' may be difficult because of the non-trivial form of the PQ transfer function. In order to obtain a closed form solution, the forward transfer function *f* ( X ) is approximated with a first degree polynomial using the truncated Taylor series expansion:

*f* (X*i* + ∆) = *f* (X *i*) + *f* *'* (X*i*) \* ∆, (7‑55)

where *f* *'* (X*i*) is the value of the derivative of *f* (X) with respect to X at point X*i*. Substituting (7‑55) into (7‑54), the cost function is approximated as follows:

*D* = ( wR \* *f* *'* (*R'*org) \* ∆R + wG \* *f* *'* (*G'*org) \* ∆G + wB \* *f* *'* (*B'*org) \* ∆B )2.  (7‑56)

Colour component values *R', G',* and *B'* in the transfer function domaincan be obtained from Y', Cb, and Cr data using the inverse colour transformation specified in Section 10.1.3 using equation (10‑2). [ED. (AT): The text was redefining the equations specified in section 10.1.3. If we wish to follow existing practice, we may wish to avoid redefining the same information again]

After substituting (7‑56) into (10‑2), we get :

*D* = ( wR \* *f* *'* (*R'*org) (*aRY* \* Y*'*new + *aRCb* \* + *aRCr* \* – (*aRY* \* Y*'*org + *aRCb* \* Cborg + *aRCr* \* Crorg ) )   
 + wG \* *f* *'* (*G'*org) (*aGY* \* Y*'*new + *aGCb* \* + *aGCr* \* – (*aGY* \* Y*'*org + *aGCb* \* Cborg + *aGCr* \* Crorg ) )   
 + wB \* *f* *'* (*B'*org) (*aBY* \* Y*'*new + *aBCb* \* + *aBCr* \* – (*aBY* \* Y*'*org + *aBCb* \* Cborg + *aBCr* \* Crorg ) ))2. (7‑57)

[ED. (AT): We may wish to change this into an equation form. TBD]

[ED. (AT): We need to reuse variables defined earlier to avoid confusion and be a bit more consistent. TBD]

Sorting the expressions inside the brackets and substituting with their numerical values those coefficients in matrix A which are equal to 0 and 1, we get

*D* = ( wR \* *f* *'* (*R'*org) \* ( Y*'*new – *eR* ) ) + wG \* *f* *'* (*G'*org) \* ( Y*'*new – *eG*) + wB \* *f* *'* (*B'*org) \* (Y*'*new – *eG* ) )2,(7‑58)

where *eR*, *eG*, and *eB* are defined as follows

*e*R = Y'org – ( – Crorg) \* *aRCr*,  
 *e*G = Y'org – ( – Cborg) \* *aGCb* – ( – Crorg) \* *aGCr* ,  
 *e*B = Y'org – ( – Cborg) \* *aBCb*, (7‑59)

Then, in order to find the local minimum, we differentiate *D* with respect to Y' (note that *eR*, *eG*, and *eB* do not depend on *D*), set the derivative equal to zero, and solve the resulting equation with respect to Y'. The value of Y' is then equal to:

(7‑60)



Using equations (7‑59) and (7‑60), it is possible to obtain the results in a single step. This approach can be adopted by applications that would benefit from or require lower complexity and a fixed number of operations per pixel. The algorithm presented above has a small performance difference in terms of objective metric performance compared to the bisection search. This is mostly due to the approximation of the transfer function with its tangent. However, it has been observed that the results of the algorithms look visually identical on the content studied in the experiments. [ED. (AT): Is this statement needed? We never performed a careful experiment to verify this. Also “visually identical” seems to be a rather strong term. I would suggest removing it or toning this down a bit. TBD]

The derivative *f* *'* (X) in equation (7‑60) can be computed using a formula obtained by the differentiation of the forward transfer funtion or by the definition of a derivative, i.e. by dividing the change in the function value by the increment of the function argument. Alternatively, the derivative values can be pre-computed and stored in a look-up table.

As mentioned earlier, the weights wR, wG, and wB can be chosen based on the desired precision or importance of each component. For example, they can be set equal to 1 or based on the contribution of each colour component to the luminance.

To summarize the algorithm described above, the following steps are to be applied:

1. Convert the Rorg, Gorg, and Borg data to their R′org, G′org, and B′org representation, if needed.
2. Given the R′org, G′org, and B′org planes generate Cborg and Crorg chroma planes.
3. Downscale chroma planes to 4:2:0 or 4:2:2 representation and quantize the samples.
4. De-quantize and up-convert the chroma back to their original resolution to obtain Cbnew and Crnew.
5. For each luma sample, calculate *e*R, *e*G, *e*B based on (7‑59).
6. Calculate Y*'*new based on (7‑60)

[ED. (AT): Lets reuse variable names as in the previous section. This avoids confusion]

NOTE  – the equations described above do not take into account the effects of clipping of the R, G, and B data, within the range of 0 to 10,000 cd/m2, when applying the colour transformation from Y*'*CbCr to RGB . This may decrease the precision of the Y*'*new estimation when R, G, and B values are close to their upper limit of 10,000 cd/m2. This effect of the clipping can be mitigated by modifying equations (7‑61) when one or more of the R, G, and B are clipped at 10,000 nits. The details of such a modification are considered as being out of the scope for this report. It can be argued that this effect would not be significant for most of the currently available HDR/WCG content, given that due to limitations of existing displays, HDR/WCG content are rarely mastered with a peak luminance value close to 10,000 cd/m2. Therefore, the results obtained with the solution described above would usually be subjectively identical to the bisection search. [ED. (AT): The word identical, even if in terms of subjective performance, seems too strong. This should be adjusted]

NOTE  – Several other methods for performing the luma adjustment process using a closed form process, have also been presented.

# Encoding process

After preprocessing, the data is ready for compression. The HDR/WCG data coming out of the preprocessing step will exhibit slightly different characteristics than typical, standard dynamic range (SDR) data. This means that it may be possible to increase perceptual/subjective quality if the encoder is configured in slightly different manner compared to when compressing SDR data. This section presents two such differences in data characteristics and give guidence on how an encoder may be configured to better exploit these differences.

## Perceptual Luma Quantization

When processing SDR data, a power law transfer function such as the one described in BT.709 is typically used. As is described above, the HDR/WCG data has instead undergone processing using the SMPTE ST 2084 (PQ) transfer function. This will in itself give a different characteristic of the processed data. One way to see this is to preprocess the same SDR data using both the BT.709 transfer function and the PQ transfer function. For a 10 bit representation, if we assume that the original data has a peak brightness of 100 cd/m2, the luma component will occupy all code levels from 64 to 940 if the BT.709 transfer function is used. However, only code levels from 64 to 509 will be used in the case of the PQ transfer function. Since the step sizes are different in the two cases, a pertubation of +/- 1 code level around code level 509 (100 cd/m2) in the PQ case will be equivalent of roughly +/- 4 code levels around code level 940 (also 100 cd/m2) in the BT.709 case. At the same time, a pertubation of +/- 1 code level around code level 80 (0.01 cd/m2) in the PQ case will be roughly equivalent to a pertubation of +/- 1 code level around code level 80 (0.01 cd/m2) in the BT.709 case. Thus, if an encoder is wired to treat an error of one code level the same way regardless if it is at level 80 or 509, such an encoder will allow errors that are four times larger in the bright areas (at around 100 cd/m2) if it uses the PQ transfer function compared to the use of BT.709. In other words, by switching from a BT.709 transfer function to SMTPE ST 2084, a lot of bits will be redistributed from the bright areas of the image to the dark areas.

Thus, if an encoder with a certain setting has achieved a good balance between bright and dark areas for the BT.709 transfer function, using the encoder with the same settings for PQ may produce images in which bright and dark areas are allocated too few and too many bits, respectively. This may result in more objectionable compression artifacts in the bright areas, while no perceivable improvement may be observed in the dark areas. For HDR/WCG data this effect can be even more pronounced; the luminance increase from a code level increase is even higher at, for example 4000 cd/m2 than it is at 100 cd/m2. Furthermore, it might be the case that the HDR/WCG content contains considerable amounts of noise in the dark areas which may further perturbate this behaviour.

One way to ameliorate this effect is for the encoder to calculate the average luma value in a block and, using this value, adaptively adjust the block’s quantization (QP) parameter. In particular, an encoder may increase or decrease the QP for the block if it is classified as a *dark* or a *bright* block respectively. In this way, it may be possible to shift bits back from dark regions to bright regions and potentially achieve a result that may be perceptually more pleasing.

[Ed. (AN): the algorithm described above partly undoes the effect of applying inverse PQ to the sample values. It would be useful to explain why it helps subjectively, as the motivation for using PQ is exatly to shift bits to the dark areas (as the human eye is more sensitive to the errors in the dark areas). Could it possibly be related to bright areas in the frame masking errors in the dark areas as a result of eye adaptation to bright areas?]

[Ed. (AT) Let’s revisit during the meeting. In general, adaptive rate allocation is used even in SDR systems. This is not an effect of HDR systems.]

Many existing coders already use some form of adaptive QP method. As an example, such methods can be used to increase the QP in areas of very high variance (where it is perceptually hard to see errors) and decrease the QP in areas of lower variance (where errors are typically more visible). In some other systems, brightness, motion, as well as other features, may also be considered. However, these methods are likely to have been designed based on SDR content characteristics. Given the above observations regarding the transfer function relationships, it is advised that, when compressing PQ encoded data, a QP adaptation method is considered that also takes into account these relationships. Other characteristics, such as colour, could also be considered.

A simple example QP adaptation method, which is used in Model B, of Annex B, is presented below. This method was found to result in better subjective as well as objective performance compared to the fixed QP coding configuration that is used in Model A.

### Brightness/Luma Dependent Adaptive Quantization – An Example

The purpose of this approach is to try and match a similar level of distortion to a particular, gray level, luminance value when either the power law transfer function of BT.709 or the PQ transfer function are used, in combination with 10 bit quantization as well as a codec’s quantization level. More specifically, it is highly desirable to dermine the QP value QPPQ that should be used with a PQ encoded value x, that would result in the same or similar distortion if that same value was encoded using the BT.709 transfer function and a known QP value QP709­. That is:

(8‑1)

The linear characteristics of the transformations employed on residual data in codecs such as MPEG-4 AVC/H.264 and HEVC/H.265 enable the consideration of these formulations even after such transformations are performed. However, these also limit the consideration of such an optimization at a block level. Based on the characteristics of the BT.709 and PQ transfer function and equation (8‑1) we can compute an approximate relationship between QPPQ and QP709 as: (8‑2)

This relationship is depicted in Table 8‑1, as well as in Figure 8‑1, with intL replacing the value of x. More specifically, in a particular implementation, intL is computed by obtained the average luma value of a 64x64 CTU block, Laverage, and then rounding this quantity, i.e. intL= round(Laverage). Based on this relationship, for every CTU, the QP will be adjusted according to its brightness by this dQP value.

Table 8‑1 –Look-up table of the dQP value from the average of the luma value.   
As an example, an average luma value of 503 in a 64x64 block would result in a dQP value of -1.

|  |  |
| --- | --- |
| **luma intL range** | **dQP** |
| intL < 301 | 3 |
| 301 ≤ intL < 367 | 2 |
| 367 ≤ intL < 434 | 1 |
| 434 ≤ intL < 501 | 0 |
| 501 ≤ intL < 567 | -1 |
| 567 ≤ intL < 634 | -2 |
| 634 <= intL < 701 | -3 |
| 701 <= intL < 767 | -4 |
| 767 <= intL < 834 | -5 |
| intL >= 834 | -6 |



Figure 8‑1 – Change in QP value as a function of the average luma value in the 64x64 pixel block.

## Chroma QP offset

Another major difference between HDR/WCG and SDR data has been observed in the characteristics of the chroma channels Cb and Cr. For 10 bit SDR content encoded using the BT.709 transfer function and the BT.709 colour space, typically all three components Y’, Cb, and Cr use the entire allowed range, i.e., Y’ will use up most of the range [64, 940] and Cb and Cr will populate most of [64, 960]. However, for HDR/WCG data using the BT.2020 colour space and the PQ transfer function, the Cb and Cr distributions will be clustered closer to the mid point of 512, which represents a value of Cb and Cr equal to 0. On the other hand, the Y’ component may still populate most of its allowed range. Furthermore, if the content does not excercise the entire BT.2020 colour space, the Cb and Cr distributions will be even more tightly clustered around 0. In particular, if SDR content is instead represented using the PQ transfer function and the BT.2020 colour space, the distribution of the Cb and Cr will be reduced substantially compared to its original BT.709 representation. However, the luminance distribution may not be as affected.

The above observations may have a considerable impact on the encoding process. An existing encoder setting may have been able to achieve a good balance between luma and chroma for SDR content in a BT.709 representation. However, the same encoder with the same settings will likely not achieve the same performance for the same content if the content is represented using the PQ transfer function and the BT.2020 colour space. Given the characteristics of the new representation will result in a bitrate allocation shift from chroma to luma. However, if chroma is not allocated enough bits, this may give rise to visible chroma artifacts. These artifacts may for example appear in white areas, where miscolourations in the direction of cyan and magenta can become visible, as seen in Figure 8‑2a.

One way to ameliorate this is for the encoder to apply a negative chroma QP offset value. This will lower the QP value used for quantizing the chroma coefficients and has an effect similar to stretching out the Cb and Cr distributions. This effectively shifts bits back from luma to chroma and thus allowing the encoder to achieve a better balance between chroma and luma quality.

Since chroma artifacts typically become more visible at low bit rates, applying a large negative chroma QP offset at such rates can potentially help reduce these artifacts significantly. However, after a certain rate point, chroma quality may be considered as being good enough. At this point it may no longer be necessary to shift bits from luma to chroma. Thus, at higher rates the chroma QP offset can be smaller or even set to zero.

A special case occurs when it is known that the content is in a restricted subset of the BT.2020 colour space. As an example, if a mastering display limited within the P3D65 colour space was used to grade the content, then it is likely that the content does also not venture outside of this space. In this case, it might be known in advance that the chroma values will never go outside a certain interval that is much smaller than the allowed [64, 960] range. Under such circumstances it may be advantageous to use a larger negative chroma QP offset compared to the QP offset used for content that makes use of the entire BT.2020 colour space.

### Example of Chroma QP offset settings.

In the following example it is assumed that the colour space of the mastering display/capture device is known.

Based on this knowledge a model is used to assign QP offsets for Cb and Cr based on the luma QP and a factor based on the capture and representation colour spaces. The model is expressed as:

QPoffset\_Cb = clip(round(ccb \* (k \* QP+l)), -12, 0) (8‑3)

QPoffset\_Cr = clip(round(ccr \* (k \* QP+l)), -12, 0), (8‑4)

where ccb = 1 if the capture colour space is the same as the representation colour space, ccb=1.04 if the capture colour space is P3D65 and the representation colour space is BT.2020, and ccb=1.14 if the capture colour space is BT.709 and the representation space is BT.2020.

Similarly, ccr = 1 if the capture colour space is the same as the representation colour space, ccr=1.39 if the capture colour space is P3D65 and the representation colour space is BT.2020, and ccr=1.78 if the capture colour space is BT.709 and the representation space is BT.2020.

Finally, k = -0.46 and l = 9.26.

[Ed. (AN): Should it be mentioned how the constants above were obtained?]

[Ed. (AT): I agree. Basically these were computed based on the coverage range of these smaller ranges compared to BT.2020. Maybe add a figure to illustrate how large part of BT.2020 the BT.709 colours occupy.]

For HEVC, if no other chroma QP offset is desired on a picture level, the syntax elements pps\_cb\_qp\_offset and pps\_cr\_qp\_offset can be set equal to QPoffset\_Cb and QPoffset\_Cr, respectively. Finer control of the chroma QP offset can be achieved at the slice level.

For AVC, if no other chroma QP offset is desired on a picture level, the syntax elements chroma\_qp\_index\_offset and second\_chroma\_qp\_index\_offset can be set equal to QPoffset\_Cb and QPoffset\_Cr, respectively.

An example of the effect of this method is shown in Figure 8‑2. Figure 8‑2a shows a tonemapped segment of an HDR/WCG image that was compressed without using neither the luma QP nor the chroma QP offset modifications described above. Figure 8‑2b, on the other hand shows the same segment compressed at the same bit rate using both of these modifications. It should be noted that the large chroma artifacts, especially on the white window shutter and on the inside of the umbrella, have been ameliorated. Furthermore luma, especially in the wall areas, has also been improved.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 8‑2 – a) Image quality without the enhancements. b) Image quality with the enhancements.

## Other encoding aspects

Apart from modifying the QP allocation in the encoder, it may also be desirable for an encoder manufacturer to adjust other non-normative encoding processes in their encoders, such as the motion estimation, intra and inter mode decision, trellis quantization, and rate control among others. These processes commonly consider simple distortion metrics such as Mean Absolute Difference (MAD), or Sum of Square Errors (SSE), for making a variety of decisions for the decision process, and may have been tuned based on SDR content characteristics. Given, however, the earlier observations about the differences in the characteristics between SDR and HDR/WCG content, these processes may also need to be appropriately adjusted. Furthermore, other metrics may also be more appropriate for these encoding decisions. These aspects are not explored in the context of this document or in any of the models defined in Annex B.

## HEVC encoding

When creating the HEVC bitstream it is recommended to set the syntax elements listed in Table 8‑2 to the values listed in Table 8‑2 in each Sequence Parameter Set (SPS) in the bitstream. The syntax elements in Table 8-1 below are conveyed in the Video Usability Information syntax branch of the SPS defined in Annex E of the HEVC specification. They may also be duplicated and carried in various application layer headers. The equivalent parameters in the HEVC HM reference software are also presented.

Table 8‑2 – Recommended settings for HEVC encoding

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **Location** | **HM parameter name** | **Recommended  value** |
| general\_profile\_space | profile\_tier\_level() | N/A | 0 |
| general\_profile\_idc | profile\_tier\_level() | Profile | 2/main10 |
| vui\_parameters\_present\_flag | seq\_parameter\_set\_rbsp( ) | VuiParametersPresent | 1 |
| video\_signal\_type\_present\_flag | vui\_parameters( ) | VideoSignalTypePresent | 1 |
| video\_full\_range\_flag | vui\_parameters( ) | VideoFullRange | 0 |
| colour\_description\_present\_flag | vui\_parameters( ) | ColourDescriptionPresent | 1 |
| colour\_primaries | vui\_parameters( ) | ColourPrimaries | 9 |
| transfer\_characteristics | vui\_parameters( ) | TransferCharacteristics | 16 |
| matrix\_coeffs | vui\_parameters( ) | MatrixCoefficients | 9 |
| chroma\_loc\_info\_present\_flag | vui\_parameters( ) | ChromaLocInfoPresent | 1 |
| chroma\_sample\_loc\_type\_top\_field | vui\_parameters( ) | ChromaSampleLocTypeTopField | 2 |
| chroma\_sample\_loc\_type\_bottom\_field | vui\_parameters( ) | ChromaSampleLocTypeBottomField | 2 |

For HDR/WCG content represented with the colour primaries of BT.2020 [5] and the transfer function of ST 2084 [11], the video characteristics is typically different compared to the video characteristics of SDR content represented with BT.709 colour primaries and BT.709 OETF (BT.1886 EOTF) transfer function [3]. It is recommended to adjust the distribution of bits between chroma and luma for example by setting chroma QP offset (controlled by the HEVC 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 distribution of bits 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. It is also recommended to take into account the activity (variance) of a block when setting delta QP for the block.

## AVC encoding

When creating the AVC bitstream it is recommended to set the syntax elements listed in Table 8‑3 to the values listed in Table 8‑3 in each Sequence Parameter Set in the bitstream. The syntax elements in Table 8-2 below are conveyed in the Video Usability Information syntax branch of the SPS defined in Annex E of the AVC specification. They may also be duplicated and carried in various application layer headers.

Table 8‑3 – Recommended settings for AVC encoding

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **Location** | **JM parameter Name** | **Recommended**  **value** |
| profile\_idc | seq\_parameter\_set\_data( ) | ProfileIDC | 110 |
| vui\_parameters\_present\_flag | seq\_parameter\_set\_data( ) | EnableVUISupport | 1 |
| video\_signal\_type\_present\_flag | vui\_parameters( ) | VUI\_video\_signal\_type\_present\_flag | 1 |
| video\_full\_range\_flag | vui\_parameters( ) | VUI\_video\_full\_range\_flag | 0 |
| colour\_description\_present\_flag | vui\_parameters( ) | VUI\_colour\_description\_present\_flag | 1 |
| colour\_primaries | vui\_parameters( ) | VUI\_colour\_primaries | 9 |
| transfer\_characteristics | vui\_parameters( ) | VUI\_transfer\_characteristics | 16 |
| matrix\_coeffs | vui\_parameters( ) | VUI\_matrix\_coefficients | 9 |
| chroma\_loc\_info\_present\_flag | vui\_parameters( ) | VUI\_chroma\_location\_info\_present\_flag | 1 |
| chroma\_sample\_loc\_type\_top\_field | vui\_parameters( ) | VUI\_chroma\_sample\_loc\_type\_top\_field | 2 |
| chroma\_sample\_loc\_type\_bottom\_field | vui\_parameters( ) | VUI\_chroma\_sample\_loc\_type\_bottom\_field | 2 |

(Ed. Note (AT): the JM currently does not support the PQ entry)

For HDR/WCG content represented with the colour primaries of BT.2020 [5] and the transfer function of ST 2084 [11], 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 [3]. It is recommended to adjust the distribution of bits 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 distribution of bits 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. It is also recommended to take into account the activity (variance) of a block when setting delta QP for the block.

# Decoding process

When the bitstream is an HEVC/H.264 bitstream the decoding process as in the HEVC specification [2] is performed.

When the bitstream is an MPEG-4 AVC/H.264 bitstream the decoding process in the MPEG-4 AVC/H.264 specification [1] is performed.

NOTE 13 – The decoding process for HDR/WCG video is no different from the decoding process of SDR video.

# Post-decoding processes

The post-decoding stage described in this document includes the following components:

1. a conversion component that converts a fixed-point representation, i.e. 10 bits, to a floating-point representation,
2. a chroma up-conversion component that converts data from 4:2:0 to 4:4:4,
3. a colour format conversion component that converts data from the Non-Constant Luminance Y′CbCr representation back to a the non-linear R′G′B′ representation, and
4. a conversion component from the non-linear data representation back to a linear data representation.



NOTE  – As was also in the case of pre-encoding, image scaling might also be desirable during this stage. For example, up‑scaling the decoded data from a 1920x1080 to a 3840x2160 resolution is highly likely given the prevalence of 4K HDR/WCG displays. Again, we will not elaborate on this topic, however, we would suggest that rescaling is performed in either the linear domain, or at least using the R′G′B′ representation.   
[Ed. (JS): If the suggestion should be kept in this note, there needs to be results to back up the suggestion.]

Figure 10‑1 presents a diagram of how these components could potentially be combined together to generate the desirable outcome, in a conventional manner. In this system, all blocks work independently, whereas chroma up‑sampling is performed using fixed-point arithmetic and its outcome is at the same precision as the input signal.

Figure 10‑1 – Conventional post-decoding process system diagram.

The various components of this stage are better introduced in the Section 10.1. More specifically, sub-section 10.1.1 describes the conversion process from a fixed-point representation back to a floating point representation, sub-section 10.1.2 discusses chroma up-conversion, sub-section 10.1.3 describes the colour representation conversion, i.e. from Y′CbCr back to R′G′B′ and sub-section 10.1.4 describes the conversion steps from a non-linear representation back to a linear one. Other configurations than the one depicted in Figure 7‑1, that can provide different performance/complexity trade-offs, could also be used.

## Post-decoding processes

### Conversion from a fixed-point to a floating-point representation

This process can be seen as the exact inverse of the process presented in Section 7.1.4. In particular, a fixed-point precision value can be converted to a floating-point precision value using the following formula:

(10‑1)

The exact same values for scale and offset as in Section 7.1.4 are used according to the component type, whereas minE and maxE are equal to -0.5 and 0.5 for the chroma components respectively, and equal to 0 and 1.0 for all other colour components.

### Chroma up-sampling

Chroma plane interpolation both vertically and horizontally is performed to convert the 4:2:0 NCL Y′CbCr signal to a 4:4:4 representation. Similar to the down-conversion process in Section 7.1.3, this step needs to again account for the siting of the chroma components compared to those of the luma (Figure 7‑3).

It is quite likely that FIR linear filters would be used by many implementations for this process. Similar to the down-conversion case, it has been reported that non-linear and/or adaptive filters could result in improved performance, but this work is still under investigation. Regardless, it has been observed that the 2-phase resampling filter shown in Table 10‑1 could provide relatively reasonable performance results on highly compressed HDR/WCG material. The same filter is applied both vertically and horizontally.

NOTE  – This is essentially a Lanczos 2 filter. Higher precision and order filters could potentially be used when up‑sampling content of very high quality/no compression.

Table 10‑1 – 2-phase chroma resampling filter

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

### Colour representation conversion: Non Constant Luminance Y′CbCr to R′G′B ′

Conversion from the non-constant luminance Y′CbCr representation back to the R′G′B′ representation can be performed using the following formula:

(10‑2)

Since , all the matrix coefficients in **A** can be calculated directly from *wYR*, *wYG*, *wYB*. In particular, given the characteristics and precision of the coefficients used in Section 7.1.2, it is possible to compute that:

, (10‑3)

(10‑4)

(10‑5)

(10‑6)

(10‑7)

(10‑8)

Basically, if high precision is possible, it is recommended that the following matrix should be used:

(10‑9)

For systems with limited precision, a lower precision representation of the above matrix, that retains only 6 digits of precision, could be used instead.

### Conversion from a non-linear to a linear light representation: R′G′B′ to RGB

Conversion from a non-linear to a linear light representation is performed using a forward transfer function, or as is commonly referred to other specifications, an electrical-optical transfer function (EOTF). In this document, the exact inverse of the process described in Section 7.1.1 is used.

More specifically, the linear light intensity signal Lo can be computed from the non-linear representation V, which takes values in the range [0, 1], as follows:

(10‑10)

where c1, c2, c3, m, and n were defined in Section 7.1.1. In Figure 10-2 below, notable points on the plot include: non-linear real-value signal value 0.5 (corresponding to 10-bit narrow code level 502) maps to 0.01 real-value signal output (0.01 \* 10 000 = 100 nits), while input value 0.9 (10-bit code level 853) maps to 0.4 (4000 nits).



Figure 10‑2 – Behaviour of the ST 2084 (PQ) forward transfer function.

This process is applied to all R′, G′, and B′ non-linear representations, resulting in their linear light counterparts R, G, and B as follows.

R = TF( R′ ) (10‑11)

G = TF( G′ ) (10‑12)

B = TF( B′ ) (10‑13)

LUTs, as was also the case for the linear to non-linear conversion, could also be used instead of using directly Equation (10‑10). In that scenario, a non-uniform LUT interpolator is also recommended.

Annex A. VUI Supplemental enhancement information (SEI) messages.

This annex provides short descriptions of SEI messages that can be used together with HDR/WCG video.

* 1. Mastering display colour volume SEI message

If mastering display colour volume information is included, 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 A.1­ 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 A.1­ – Example Mastering display colour volume SEI message representing P3 colour primaries [RP 431-2], D65 white point and luminance range of 0 cd/m2 to 2000 cd/m2, inclusive.

|  |  |  |
| --- | --- | --- |
| **Syntax element** | **HM parameter** | **Example value** |
| display\_primaries\_x[0] | SEIMasteringDisplayPrimaries | 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 | SEIMasteringDisplayWhitePoint | 15635 |
| white\_point\_y | 16450 |
| max\_display\_mastering\_luminance | SEIMasteringDisplayMaxLuminance | 20000000 |
| min\_display\_mastering\_luminance | SEIMasteringDisplayMinLuminance | 0 |

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 apply.

* 1. Content Light Level information SEI message

[Ed. (JS): Add description and example.]

Blu-Ray Disc Association BD-ROM 3.0 UltraHD specification [15] provides guidelines that MaxFALL not exceed 400 nits, and that only a small percentage of pixels, such as specular highlights, exceed 1000 nits. For the BD-ROM 3.0 application, these measures are taken off-line during a complete pass through the content before the final MaxFALL and MaxCLL calculation is made. Live or broadcast services may set these values to pre-determined levels without actual measurement on content.

Annex B. Model A and Model B configuration settings.

This annex describes the configuration parameter settings for the reference software accompanying this document, known as HDRTools, as well as configuration parameter settings for the HEVC reference software, HM. Two different configurations are presented; *Model A* that corresponds to the setup which was used as reference in MPEGs Call for Evidence [16] and *Model B* that corresponds to an improved reference used in subsequent evaluations. Eqvivalent configuration parameter settings were tested in the verification test for HDR/WCG video coding using HEVC Main 10 Profile [17].

* 1. Model A configuration settings

HDRTools version 0.12 is used for conversion from EXR2020, BT2020Tiff or P3D65Tiff and YCbCr420 with the following changes to the default parameters found in the CfE\_cfgFiles folder:

EnableTFunctionLUT = 0 # Use LUTs for TF computations

EnableTFDerivLUT = 0 # Use LUTs for TF derivative computation

ClosedLoopConversion = 0

HM version 16.2 [ref] is used with the following additions to encoder\_random\_access\_main10.cfg:

Level : 4.1

LumaLevelToDeltaQPMode : 0 # Change luma deltaQP based on average luma

WCGPPSEnable : 0

HDRTools version 0.12 is used for conversion from YCbCr420 to EXR2020, BT2020Tiff an P3D65Tiff.

* 1. Model B configuration settings

HDRTools version 0.12 is used for conversion from EXR2020, BT2020Tiff or P3D65Tiff to YCbCr420 with the default config files found in the CfE\_cfgFiles folder which includes the following parameter setting:

EnableTFunctionLUT = 1 # Use LUTs for TF computations

EnableTFDerivLUT = 1 # Use LUTs for TF derivative computation

ClosedLoopConversion = 8

For the approximative luma adjustment method presented in Section 7.2.2 the value XX should be used for ClosedLoopConversion.

HM version 16.2 is used with the following additions to encoder\_random\_access\_main10.cfg:

Level : 4.1

LumaLevelToDeltaQPMode : 1 # Change luma deltaQP based on average luma

WCGPPSEnable : 1

WCGPPSChromaQpScale : -0.46 # Linear chroma QP offset mapping (scale)

WCGPPSChromaQpOffset : 9.26 # Linear chroma QP offset mapping (offset)

WCGPPSCbQpScale : 1.14 # Scale factor depending on capture and

# representation colour space (with BT.2020

# container use 1.14 for BT.709 material

# and 1.04 for P3 material)

WCGPPSCrQpScale : 1.79 # Scale factor depending on capture and

# representation colour space (with BT.2020

# container use 1.79 for BT.709 material

# and 1.39 for P3 material)

HDRTools version 0.12 is used for conversion from YCbCr420 to EXR2020, BT2020Tiff, and P3D65Tiff.