Luma Adjustment for High Dynamic Range Video

Jacob Ström, Jonatan Samuelsson, and Kristofer Dovstam

Ericsson Research
Färögatan 6
164 80 Stockholm, Sweden
{jacob.strom,jonatan.samuelsson,kristofer.dovstam}@ericsson.com

Abstract

In this paper we present a solution to a luminance artifact problem that occurs when conventional non-constant luminance $Y'CbCr$ and 4:2:0 subsampling is combined with the type of highly non-linear transfer functions typically used for High Dynamic Range (HDR) video. These luminance artifacts can be avoided by selecting a luma code value that minimizes the luminance error. Subjectively, the quality improvement is clearly visible even for uncompressed video. Improvements in tPSNR-Y of up to 20 dB have been observed, compared to conventional subsampling. Crucially, no change in the decoder is needed.

Introduction

Recently, a tremendous increase in quality has been achieved in digital video by increasing resolution, going from standard definition via high definition to 4k. High dynamic range (HDR) video uses another way to increase perceived image quality, namely by increasing contrast. The conventional TV system was built for luminances between 0.1 candela per square meter (cd/m^2) and 100 cd/m^2, or about ten doublings of luminance [5]. We will refer to this as standard dynamic range (SDR) video. As a comparison, some HDR monitors are capable of displaying a range from 0.01 cd/m^2 to 4000 cd/m^2, i.e., over 18 doublings.

Conventional SDR Processing

Typical SDR systems such as TVs or computer monitors often use an eight bit representation where 0 represents dark and 255 bright[1]. Just linearly scaling the code value range $[0, 255]$ to the luminance range $[0.1, 100]$ cd/m^2 mentioned above would not be ideal: The first two code words 0 and 1 would be mapped to 0.1 cd/m^2 and 0.49 cd/m^2 respectively, a relative difference of $\frac{0.49-0.1}{0.1} = 390\%$. The last two code words 254 and 255 on the other hand, would be mapped to 99.61 cd/m^2 and 100 cd/m^2 respectively, a relative difference of only $\frac{100-99.61}{99.61} = 0.3\%$. To avoid this large difference in relative step sizes, SDR systems include an electro-optical transfer function (EOTF) which maps code values to luminances in a non-linear way. As an example, the red component is first divided by 255 to get a value $R'_{01} \in [0, 1]$ which is then fed through a power function

$$R_{01} = (R'_{01})^{\gamma}. \tag{1}$$

[1]Some systems use a restricted range from 16 to 235, but we disregard this here for simplicity.

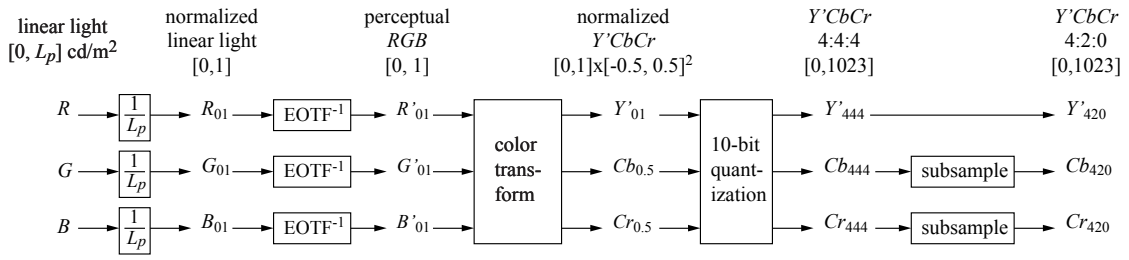

Figure 1: Going from linear light to $Y'CbCr$.

Finally R_{01} is scaled to the range $[0.1, 100]$ to get the light representation R in cd/m^2. The green and blue components are handled in the same way. By selecting $\gamma = 2.4$, the relative difference between the first two code words becomes 0.16% and ditto for the last two code words becomes 0.95%, which is much more balanced.

SDR acquisition process

For video, the acquisition process can be modelled according to Figure 1. Assuming the camera sensor measures linear light (R, G, B) in cd/m^2, the first step is to divide by the peak brightness to get to normalized linear light (R_{01}, G_{01}, B_{01}). Then the inverse of the EOTF is applied[2] $R'_{01} = (R_{01})^{\frac{1}{\gamma}}$, and likewise for green and blue. To decorrelate the color components, the transform

$$\begin{bmatrix} Y'_{01} \\ Cb_{0.5} \\ Cr_{0.5} \end{bmatrix} = \begin{bmatrix} 0.2627 & 0.6780 & 0.0593 \\ -0.1396 & -0.3604 & 0.5000 \\ 0.5000 & -0.4598 & -0.0402 \end{bmatrix} \begin{bmatrix} R'_{01} \\ G_{01} \\ B_{01} \end{bmatrix}, \tag{2}$$

is applied. The matrix coefficients depend on the color space, here we have assumed that (R, G, B) is in the BT.2020 color space. The 0.5 subscript of $Cb_{0.5}$ and $Cr_{0.5}$, is to indicate that they vary between $[-0.5, 0.5]$ rather than between $[0, 1]$.

The next step is to quantize the data. In this example we quantize to 10 bits, yielding components $(Y'_{444}, Cb_{444}, Cr_{444})$ that vary from 0 to 1023. Finally, the last two components are subsampled. We have followed the subsampling procedure described by Luthra et al. [3]. The data can now be sent to a video encoder such as HEVC [7].

Display of SDR data

On the receiver side, the HEVC bitstream is decoded to recover $\hat{Y}'_{420}$, $\hat{Cb}_{420}$ and $\hat{Cr}_{420}$. The hats are used to indicate that these values may differ from Y'_{420}, Cb_{420} and Cr_{420} due to the fact that HEVC is a lossy encoder. The signal is then processed in reverse according to Figure 2. The end result is the linear light representation $(\hat{R}, \hat{G}, \hat{B})$ which is displayed.

[2]Sometimes it may be advantageous to use a function that is not the inverse of the EOTF, but we disregard this case here for simplicity.

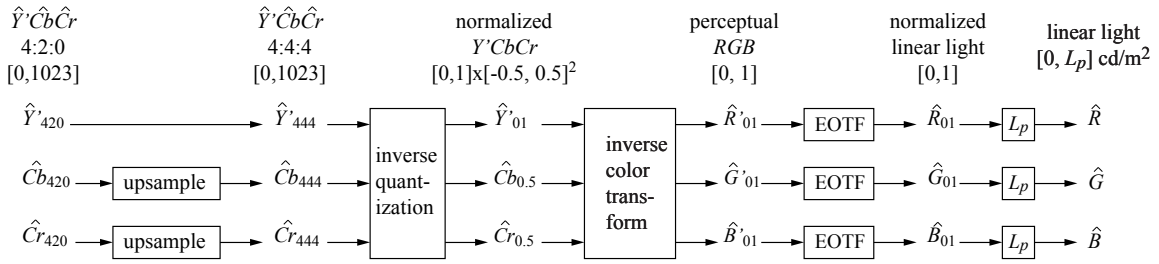

Figure 2: Going from $\hat{Y}'\hat{C}b\hat{C}r$ 4:2:0 to linear light.

HDR processing

For HDR data, which may include luminances of up to $10,000$ cd/m^2, a simple power function is not a good fit to the contrast sensitivity of the human eye over the entire range of luminances. Any fixed value of γ will result in too coarse a quantization either in the dark tones, the bright tones, or the mid tones. To solve this problem, Miller et al. introduce the PQ-EOTF [1], changing the EOTF box in Figure 2 to

$$R_{01} = \left(\frac{(R'_{01})^{\frac{1}{m}} - c_1}{c_2 - c_3 (R'_{01})^{\frac{1}{m}}} \right)^{\frac{1}{n}}, \tag{3}$$

where m = 78.8438, n = 0.1593, c1 = 0.8359, c2 = 18.8516, and c3 = 18.6875. The peak luminance L_p is also changed from 100 to $10,000$. Likewise the EOTF^{-1} box in Figure 1 is replaced by the inverse of Equation 3.

Problem

If applying the processing outlined in Figures 1 and 2 (with the new EOTF and EOTF^{-1} and $L_p = 10,000$), something unexpected occurs. As is shown in the first two rows of Figure 5, artifacts appear. Since the printed medium cannot reproduce HDR images, tone-mapped versions are calculated using $R_{SDR} = \text{clamp}\left(255 \times (R \times 2^c)^{\frac{1}{\gamma}}, 0, 255\right)$. Here clamp (x, a, b) clamps the value x to the interval $[a, b]$, $\gamma = 2.22$, and the exposure value c varies for the different images. The green and blue components are treated similarly. In the left column of Figure 5 we can see (the tonemapped version of) the original data (R, G, B). In the middle column we can see (the tonemapped version of) the end result $\left(\hat{R}, \hat{G}, \hat{B}\right)$ after going through the processing outlined in Figure 1 followed by the processing in Figure 2. Note that for the first two rows of Figure 5, no compression has taken place other than subsampling and quantizing to 10 bits. Yet disturbing artifacts occur. This problem was pointed out and illustrated by François at the 110^{th} MPEG meeting in Strasbourg, 2014 [2].

Analysis

Assume that the following two pixels are next to each other in an image:

$$RGB1 = (1000, 0, 100), \quad \text{and} \tag{4}$$
$$RGB2 = (1000, 4, 100) \tag{5}$$

Note that these colors are quite similar. However, the first four steps of Figure 1 yield

$$Y'_{444}Cb_{444}Cr_{444}1 = (263, 646, 831) \quad \text{and} \tag{6}$$
$$Y'_{444}Cb_{444}Cr_{444}2 = (401, 571, 735) \tag{7}$$

which are quite different from each other. The average of these two values is $Y'CbCr = (332, 608.5, 783)$. Now if we would go backwards in the processing chain to see what linear RGB value this represents, we get $RGB = (1001, 0.48, 100.5)$, which is quite close to both $RGB1$ and $RGB2$. Thus, just averaging all three components is not a problem. A larger problem arises when only Cb and Cr are interpolated, and we use the Y' values from the pixels without interpolation. This is what is done in conventional chroma subsampling which is performed in order to create a 4:2:0 representation. An example is the anchor generation process described by Luthra et al. [3]. For instance, taking Y' from the first pixel in Equation 6, i.e., $Y'CbCr = (263, 608.5, 783)$ represents a linear RGB color of $(484, 0.03, 45)$, which is much too dark. Similarly, taking Y' from the second pixel, in Equation 7, i.e., $Y'CbCr = (401, 608.5, 783)$ gives an RGB value of $(2061, 2.2, 216)$, which is too bright.

Possible Workarounds

Consider adding a third pixel to the example,

$$RGB3 = (1000, 8, 100). \tag{8}$$

If we convert these linear inputs to $R'_{01}G'_{01}B'_{01}$ we get

$$R'_{01}G'_{01}B'_{01}1 = (0.7518, 0.0000, 0.5081) \tag{9}$$
$$R'_{01}G'_{01}B'_{01}2 = (0.7518, 0.2324, 0.5081) \tag{10}$$
$$R'_{01}G'_{01}B'_{01}3 = (0.7518, 0.2824, 0.5081). \tag{11}$$

Clearly, the jump in G'_{01} is bigger between the first and second pixel although the linear G changes in equal steps of 4. Likewise, the difference between the $Y'CbCr$ coordinates will be bigger between the first two pixels than the last two. Hence, the effect will be biggest when one or two of the components are close to zero in linear light, i.e., when the color is close to the edge of the color gamut, something that was also pointed out by François [2]. Thus one way to avoid the artifacts can be to just avoid saturated colors. However, the larger color space of BT.2020 was introduced specifically to allow for more saturated colors, so that solution is not desirable.

This highlights another issue: Much test content is shot in Rec.709, and after conversion to BT.2020, none of the colors will be fully saturated and thus the artifacts

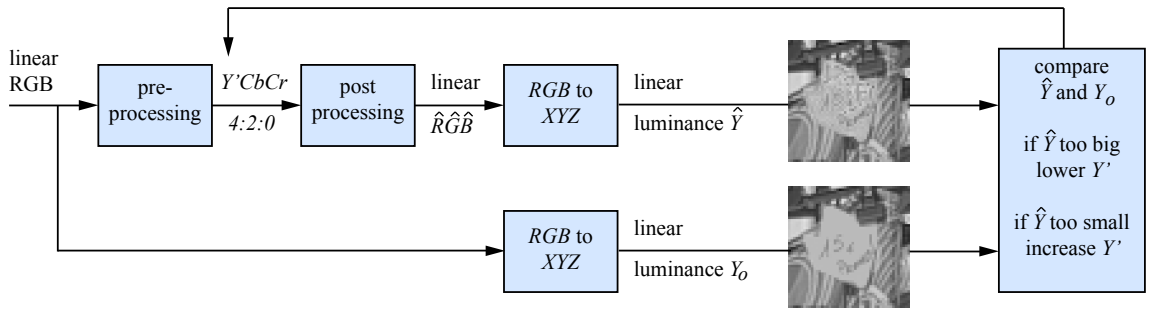

Figure 3: By changing the Y' value in an individual pixel, it is possible to reach a linear luminance $\hat{Y}$ that matches the desired linear luminance Y_o.

will be small. As an example, a pixel acquired in Rec.709, e.g., RGB709 $= (0, 500, 0)$ will after conversion to BT.2020 no longer have any zero components, RGB2020 $= (165, 460, 44)$. Later on, when cameras are capable of recording in BT.2020, much stronger artifacts will appear. To emulate the effect of BT.2020 content in a BT.2020 container, we have therefore used Rec.709 material in a Rec.709 container for the processing of the figures in this document, such as for Figure 5. Mathematically however, there is no difference, since the coordinates $R_{01}, G_{01}B_{01}$ will span the full range of $[0, 1]$ in both cases.

Another workaround is to use *constant luminance processing* (CL), as described in ITU-R Rec. BT.2020 [6]. In CL, all of the luminance is carried in Y', as opposed to only most of the luminance being carried in the luma Y' of Figure 1, which is referred to as *non-constant luminance processing* (NCL). However, one problem with CL is that it affects the entire chain; converting back and forth between a 4:2:0/4:2:2 CL representation and a 4:2:0/4:2:2 NCL representation endangers introducing artifacts in every conversion step. In practice it has therefore been difficult to convert entire industries from the conventional NCL to CL.

Proposed Solution: Luma Adjustment

The basic idea is to make sure that the resulting luminance matches the desired one. With luminance, we mean the Y component of the (linear) CIE1931 XYZ color space [4]. This Y is different from the luma Y' of Figure 1 since Y is calculated from the linear R G B values

$$Y = w_R R + w_G G + w_B B, \tag{12}$$

where $w_R = 0.2627$, $w_G = 0.6780$ and $w_B = 0.0593$. The luminance Y corresponds well to how the human visual system appreciates brightness, so it is interesting to preserve it well. This is shown in Figure 3 where both the processed signal (top) and the original signal (bottom) is converted to linear XYZ. Then the Y components are quite different as can be seen in the figure. The key insight is that the luma value Y' can be changed independently in each pixel, and therefore it is possible to arrive at the desired, or original, linear luminance Y_o by changing Y' until $\hat{Y}$ equals Y_o, as is shown in Figure 3. It is also the case that $\hat{Y}$ increases monotonically with Y', which means that it is possible to know the direction in which Y' should be changed.

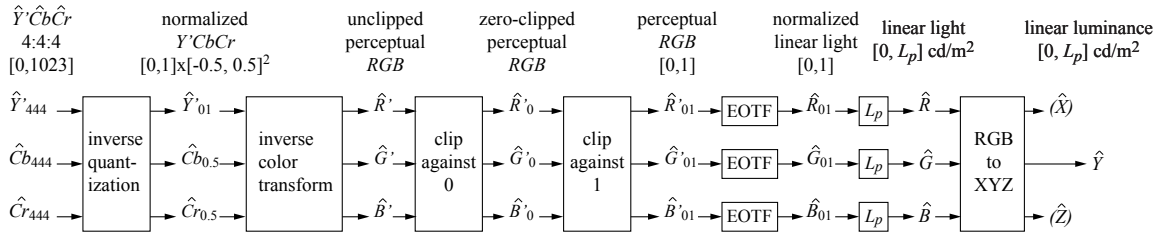

Figure 4: How $\hat{Y}$ is calculated including details on clipping.

Therefore simple methods such as interval halving can be used to find the optimal Y', in at most ten steps for 10 bit quantization. If a one-step solution is preferred, it is possible to use a 3D look-up table that takes in Cb, Cr and the desired linear luminance Y_o and delivers Y'.

Implementational aspects

The technique can be implemented efficiently in the following way: First, the desired, or original luminance Y_o for each pixel is obtained by applying Equation 12 to the original (R, G, B) values of each pixel. Second, the entire chain from (R, G, B) in Figure 1 to $\left(\hat{Y}'_{01}, \hat{C}b_{0.5}, \hat{C}r_{0.5}\right)$ in Figure 2 is carried out. Then, for each pixel, a starting interval of $[0, 1023]$ is set. Next, the candidate value $\hat{Y}'_{444} = 512$ is tried. $\hat{Y}'_{01}$ is calculated from the candidate value, and using the previously calculated $\left(\hat{C}b_{0.5}, \hat{C}r_{0.5}\right)$ it is possible to go through the last few steps of Figure 2, yielding $\left(\hat{R}, \hat{G}, \hat{B}\right)$. This is now fed into Equation 12 to get the candidate luminance $\hat{Y}$. For a given pixel, if $\hat{Y} < Y_o$, this means that the candidate value $\hat{Y}'_{444}$ was too small, and that the correct luma value must be in the interval $[512, 1023]$. Likewise if $\hat{Y} > Y_o$, the correct luma value must be in the interval $[0, 512]$. The process is now repeated, and after ten iterations the interval contains two neighboring values such as $[218, 219]$. At this stage, both of the two values are tried, and the one that produces the smallest error $\left(\hat{Y} - Y_o\right)^2$ is selected. We call this way of finding the best luma value "luma adjustment".

Mathematical Bounds

This section will describe some mathematical bounds on the optimal $\hat{Y}'_{444}$ that can be used to lower the number of needed iterations compared to if the entire interval $[0, 1023]$ is used. Figure 4 describes the calculation from $\hat{Y}'_{444}$ to $\hat{Y}$. This figure is more detailed than Figure 2; it also describes the clipping of $\hat{R}'$, $\hat{G}'$ and $\hat{B}'$ that is needed due to the fact that the inverse color transform may result in colors outside

the interval $[0, 1]$. Starting with Equation 12, and following Figure 4 backwards gives

$$\hat{Y} = w_R\hat{R} + w_G\hat{G} + w_B\hat{B} \tag{13}$$
$$= w_R L_p\hat{R}_{01} + w_G L_p\hat{G}_{01} + w_B L_p\hat{B}_{01} \tag{14}$$
$$= w_R L_p\mathbf{tf}\left(\hat{R}'_{01}\right) + w_G L_p\mathbf{tf}\left(\hat{G}'_{01}\right) + w_B L_p\mathbf{tf}\left(\hat{B}'_{01}\right), \tag{15}$$

where $\mathbf{tf}(\cdot)$ is the EOTF of Equation 3. Now let $\hat{M}'_{01} = \max\{\hat{R}'_{01}, \hat{G}'_{01}, \hat{B}'_{01}\}$. Since $\mathbf{tf}(\cdot)$ is monotonically increasing, it follows that $\mathbf{tf}\left(\hat{R}'_{01}\right) \leq \mathbf{tf}\left(\hat{M}'_{01}\right)$, and the same is true for green and blue. Hence

$$\hat{Y} \leq w_R L_p\mathbf{tf}\left(\hat{M}'_{01}\right) + w_G L_p\mathbf{tf}\left(\hat{M}'_{01}\right) + w_B L_p\mathbf{tf}\left(\hat{M}'_{01}\right) \tag{16}$$
$$= (w_R + w_G + w_B)L_p\mathbf{tf}\left(\hat{M}'_{01}\right) \tag{17}$$
$$= L_p\mathbf{tf}\left(\max\{\hat{R}'_{01}, \hat{G}'_{01}, \hat{B}'_{01}\}\right) \tag{18}$$
$$\leq L_p\mathbf{tf}\left(\max\{\hat{R}'_0, \hat{G}'_0, \hat{B}'_0\}\right), \tag{19}$$

since $w_R + w_G + w_B = 1$. The last step is due to the fact that clipping against 1 can never make a value larger. We now make the crucial observation that all three variables $\left(\hat{R}', \hat{G}', \hat{B}'\right)$ cannot be negative at the same time. They are calculated as

$$\begin{array}{lllll} \hat{R}' = \hat{Y}'_{01} & & & +a_{13}\hat{C}r_{0.5} \\ \hat{G}' = \hat{Y}'_{01} & -a_{22}\hat{C}b_{0.5} & & -a_{23}\hat{C}r_{0.5} \\ \hat{B}' = \hat{Y}'_{01} & +a_{32}\hat{C}b_{0.5} & & \end{array} \tag{20}$$

where all coefficients $\{a_{ij}\} > 0$. (The relation in Equation 2 is the inverse of this relation.) For both $\hat{R}'$ and $\hat{B}'$ to be smaller than zero, both $\hat{C}b_{0.5}$ and $\hat{C}r_{0.5}$ must be negative, since $\hat{Y}'_{01} \geq 0$. But in that case $\hat{G}'$ must be positive. Hence $\max\{\hat{R}', \hat{G}', \hat{B}'\} \geq 0$, which means that $\max\{\hat{R}', \hat{G}', \hat{B}'\} = \max\{\hat{R}'_0, \hat{G}'_0, \hat{B}'_0\}$. We can therefore write

$$\hat{Y} \leq L_p\mathbf{tf}\left(\max\{\hat{R}', \hat{G}', \hat{B}'\}\right). \tag{21}$$

Now assume $\hat{R}'$ is the largest of $\hat{R}'$, $\hat{G}'$ and $\hat{B}'$. We then have

$$\hat{Y}_{red-is-biggest} \leq L_p\mathbf{tf}\left(\hat{R}'\right) \tag{22}$$

which can be inverted to

$$\mathbf{tf}^{-1}\left(\hat{Y}_{red-is-biggest}/L_p\right) \leq \hat{Y}'_{01} + a_{13}\hat{C}r_{0.5} \tag{23}$$

where we have used Equation 20 to replace $\hat{R}'$. Thus, if red happens to be the biggest color component, we have a bound on the optimal $\hat{Y}'_{01}$,

$$\hat{Y}'_{01} \geq \mathbf{tf}^{-1}\left(Y_o/L_p\right) - a_{13}\hat{C}r_{0.5}, \tag{24}$$

where Y_o is our desired luminance, i.e., the luminance of the original. Similarly, if green or blue happens to be the biggest color component, we have two other bounds:

$$\hat{Y}'_{01} \geq \mathbf{tf}^{-1}\left(Y_o/L_p\right) + a_{22}\hat{C}b_{0.5} + a_{23}\hat{C}r_{0.5} \tag{25}$$

$$\hat{Y}'_{01} \geq \mathbf{tf}^{-1}\left(Y_o/L_p\right) - a_{32}\hat{C}b_{0.5} \tag{26}$$

One of these three bounds must be the correct one, so we can simply take the most conservative bound. Hence we get

$$\hat{Y}'_{01} \geq \hat{Y}'_{lower} = \mathbf{tf}^{-1}\left(Y_o/L_p\right) + r, \tag{27}$$

where $r = \min\{-a_{13}\hat{C}r_{0.5},\quad a_{22}\hat{C}b_{0.5} + a_{23}\hat{C}r_{0.5},\quad -a_{32}\hat{C}b_{0.5}\}$. In a similar fashion, it is possible to calculate an upper bound for $\hat{Y}'_{01}$, namely

$$\hat{Y}'_{01} \leq \hat{Y}'_{upper} = \mathbf{tf}^{-1}\left(Y_o/L_p\right) + s, \tag{28}$$

where $s = \max\{-a_{13}\hat{C}r_{0.5},\quad a_{22}\hat{C}b_{0.5} + a_{23}\hat{C}r_{0.5},\quad -a_{32}\hat{C}b_{0.5}\}$. Finally, $\hat{Y}'_{lower}$ and $\hat{Y}'_{upper}$ can be multiplied by 1023 to get bounds on $\hat{Y}'_{444}$ instead of $\hat{Y}'_{01}$.

Tighter Upper Bound

A tighter upper bound can be found using the fact that the EOTF in Equation 3 is a convex function: From Equation 15 we get

$$\hat{Y}/L_p = w_R\mathbf{tf}\left(\hat{R}'_{01}\right) + w_G\mathbf{tf}\left(\hat{G}'_{01}\right) + w_B\mathbf{tf}\left(\hat{B}'_{01}\right). \tag{29}$$

For a convex function $f(x)$, the following inequality holds if $\sum_k w_k = 1$,

$$w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) \geq f(w_1 x_1 + w_2 x_2 + w_3 x_3), \tag{30}$$

Thus

$$\hat{Y}/L_p \geq \mathbf{tf}\left(w_R\hat{R}'_{01} + w_G\hat{G}'_{01} + w_B\hat{B}'_{01}\right). \tag{31}$$

If none of the variables clip, this is equal to

$$\hat{Y}/L_p \geq \mathbf{tf}\left(w_R\hat{R}' + w_G\hat{G}' + w_B\hat{B}'\right). \tag{32}$$

Taking the inverse of Equation 20 gives

$$\begin{bmatrix} \hat{Y}'_{01} \\ \hat{C}b_{0.5} \\ \hat{C}r_{0.5} \end{bmatrix} = \begin{bmatrix} 0.2627 & 0.6780 & 0.0593 \\ -0.1396 & -0.3604 & 0.5000 \\ 0.5000 & -0.4598 & -0.0402 \end{bmatrix} \begin{bmatrix} \hat{R}' \\ \hat{G}' \\ \hat{B}' \end{bmatrix}, \tag{33}$$

and we can see that the expression in Equation 32 exactly matches the first row, giving

$$\hat{Y}/L_p \geq \mathbf{tf}\left(\hat{Y}'_{01}\right). \tag{34}$$

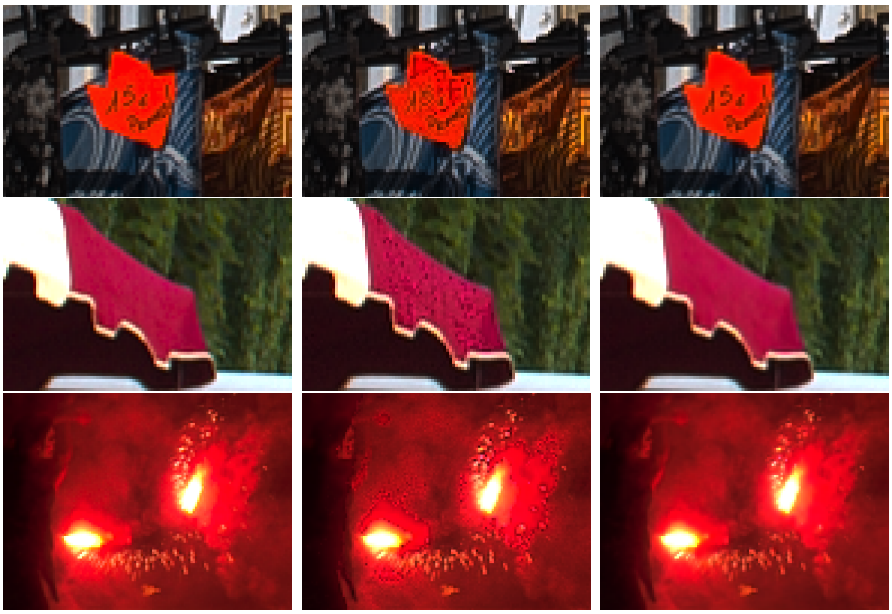

Figure 5: Left: Original 4:4:4. Middle: Conventional processing uncompressed (top two images) and compressed to 20835 kBps (bottom image). Right: Proposed method uncompressed (top two images) and compressed to 17759 kBps (bottom image). Sequences courtesy of Technicolor and the NevEx project.

This can be inverted to get

$$\hat{Y}'_{01} \leq \hat{Y}'_{upper-tight} = \mathbf{tf}^{-1}\left(Y_o/L_p\right). \tag{35}$$

Since we have disregarded the clipping, this bound is not guaranteed to hold. In practice however, the bound $\hat{Y}'_{upper-tight}$ gives a good end result if none of the following variables R'_{test}, G'_{test} or B'_{test} overflows, i.e., exceeds 1.0:

$$R'_{test} = \mathbf{tf}^{-1}\left(Y_o/L_p\right) + a_{13}\hat{C}r_{0.5} \tag{36}$$

$$G'_{test} = \mathbf{tf}^{-1}\left(Y_o/L_p\right) - a_{22}\hat{C}b_{0.5} - a_{23}\hat{C}r_{0.5} \tag{37}$$

$$B'_{test} = \mathbf{tf}^{-1}\left(Y_o/L_p\right) + a_{32}\hat{C}b_{0.5}. \tag{38}$$

If any of the variables exceed 1.0, the bound $\hat{Y}'_{upper}$ can be used instead.

Results

We implemented the conventional processing chain that is used for creating the anchors in [3] and compared this to our chain, which includes the luma adjustment step, but keeps the decoder the same. The first two rows of Figure 5 show results without compression. Here, both the conventional processing chain and our processing chain converts to $Y'CbCr$ 4:2:0 and then back to linear RGB. The bottom row shows compressed results. Note how artifacts are considerably reduced for the proposed method. Total encoding time (color conversion plus HM compression) increases about 3% compared to traditional processing. Measured over only the color conversion, execution time increases around 30% compared with the color conversion process from [3].

Table 1: tPSNR-Y and deltaE increase (dB) Rec.709 container.

class	sequence	tPSNR-Y	deltaE
class A'	FireEaterClip4000r1	13.81	2.23
	Tibul2Clip4000r1	18.01	3.85
	Market3Clip4000r2	20.30	0.15
	Overall	**17.37**	**2.08**

Table 2: tPSNR-Y and deltaE increase (dB) for BT.2020 container.

class	sequence	tPSNR-Y	deltaE
class A	FireEaterClip4000r1	5.88	0.73
	Market3Clip4000r2	10.17	0.95
	Tibul2Clip4000r1	7.60	0.02
class B	AutoWelding	11.25	0.12
	BikeSparklers	11.33	0.02
class C	ShowGirl2Teaser	6.28	0.05
class D	StEM_MagicHour	7.22	0.03
	StEM_WarmNight	8.53	0.04
class G	BalloonFestival	7.71	0.05
	Overall	**8.44**	**0.22**

For HDR material, no single metric has a role similar to PSNR for SDR content. Instead we report two metrics from Luthra et al.; tPSNR-Y for luminance and deltaE for chrominance. In Table 1 the uncompressed results for BT.709 material in a BT.709 container is shown. Here we see a large increase in luminance quality measured as tPSNR-Y of over 17 dB on average, and over 20 dB for one sequence. Also the deltaE result is improving. Table 2 shows the uncompressed results for BT.709 material or P3 material in a BT.2020 container. Here the gains are less pronounced, since no colors directly on the gamut edge are available, but the tPSNR-Y improvement is still 8 dB on average and over 11 dB for some sequences. The deltaE measure improves marginally. Note that with true BT.2020 material, we expect the gains to be more similar to those in Table 1.

References

[1] S. Miller, M. Nezamabadi and S. DalyJ, "Perceptual Signal Coding for More Efficient Usage of Bit Codes," *Motion Imaging Journal*, 122:52–59, 2013.

[2] E. Francois, Not public: "MPEG HDR AhG: about using a BT.2020 container for BT.709 content," *110th MPEG meeting in Strasbourg* Compression Conference, Strasbourg, France, October 2014.

[3] A. Luthra, E. François, W. Husak, "Call for Evidence (CfE) for HDR and WCG Video Coding", MPEG2014/N15083, 110th MPEG Meeting, Geneva, 2015.

[4] CIE (1932), "Commission internationale de l'Eclairage proceedings," 1931. Cambridge: Cambridge University Press.

[5] ITU-R, "Reference electro-optical transfer function for flat panel displays used in HDTV studio production," Recommendation ITU-R BT.1886, 03/2011

[6] ITU-R, "Parameter values for ultra-high definition television systems for production and international programme exchange," Recommendation ITU-R BT.2020-2, 10/2015

[7] ISO/IEC 23008-2:2015, "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding", 2015