|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  26th Meeting: Geneva, CH, 12–20 January 2017 | Document: JCTVC-Z0022 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AhG on HDR and WCG: Chroma Adjustment for HDR Video** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Information | | |
| *Author(s) or Contact(s):* | Jacob Ström, Per Wennersten,  Kenneth Andersson,  Rickard Sjöberg, Martin Pettersson Färögatan 6 184 60 Stockholm Sweden | Tel: Email: | +46720672192 jacob.strom@ericsson.com |
| *Source:* | Ericsson | | |

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

# Abstract

This contribution presents a preprocessing method for HDR video that is going to be compressed using a non-constant luminance 4:2:0 Y’CbCr representation and the PQ transfer function. The input and output to the preprocessing is linear RGB 4:4:4 and the processing takes place prior to subsequent processing such as subsampling. Each linear channel R, G and B is low-pass filtered in a way so that the chromaticity (in CIE coordinates) and the luminance (in ) of a processed pixel never deviates more than a fixed amount from that of the original pixel. By setting the allowed deviation small enough, the contribution states that the resulting 4:4:4 image will look perceptually equivalent to the original image, yet give rise to a smoother representation in Y’CbCr, improving compression performance for saturated colors. The contribution also claims that there are indications that this preprocessing step also helps perceptual quality of compressed material in such areas, as well as helping in the case of mismatched upsampling filters between the luma adjustment and the decoder in such areas.

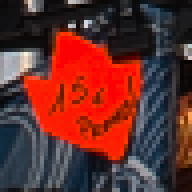
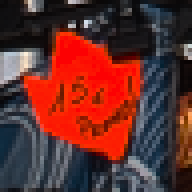
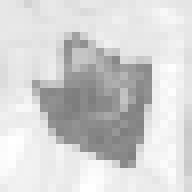
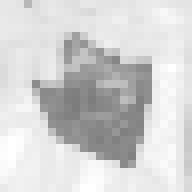
# Introduction

As noted by François et al. at the Strasbourg meeting [1], traditional subsampling of colors close to the color gamut edge can result in luminance artifacts. Luma adjustment, introduced at the Warsaw meeting [2], can help ameliorate this problem, and implementations and variants have been presented at subsequent meetings and conferences [3-15]; for an introduction, see [4].

While the main goal of the luma adjustment method is to improve the decoded luminance, it often has the beneficial side effect of making the luma smoother in areas of saturated color. An example is shown in Figure 1 where the two leftmost two images show the luma for two consecutive frames processed according to the Call for Evidence [16] (“Anchor 1.0”). The two rightmost images show the luma processed according to the enhanced reference model in [17] (“Anchor 3.2”), which uses luma adjustment. As can be seen the luma is smoother in the second case, which should make it cheaper to encode and easier to predict.

  
 Y’ frame 389 Y’ frame 390 Y’ frame 389 Y’ frame 390  
*Figure 1: Luma (Y’) of two consecutive frames. Left: Anchor 1.0. Right Anchor 3.2. Content is BT.709 in a BT.709 container.*

However, the luma adjustment does not change the chroma, and hence will not improve smoothness in the chroma components. As can be seen in Figure 2, the chroma components can be quite different from frame to frame even though the image itself does not change much perceptually.

  
 Cb frame 389 Cb frame 390 RGB444 frame 389 RGB444 frame 390

*Figure 2: The Cb (as well as Cr) can change quite a lot even though the RGB444 image looks similar across frames. RGB444 frames are tonemapped but appear very similar on an HDR display. Content is BT.709 in BT.709 container.*

The idea behind this contribution is to attempt at changing the RGB444 representation so that after conversion to Y’CbCr, the chroma components are more stable and easier to predict between frames. The goal is to do that without affecting the perceptual look of the RGB444 frame. After that, Anchor 3.2 processing (including luma adjustment) can take place.

# Concept

Assume we have the two colors RGB1 = (4000, 0, 100) and RGB2 = (4000, 4, 100) in linear ITU-R BT.2020 primaries. How different are these two colors to the human eye? Their luminance does not differ much; converting to XYZ gives Y1 = 1057 and Y2 = 1059. This is in fact smaller than a quarter of one luma code value when using 10-bit PQ: -0.24, which means that this is a luminance four times smaller than what we can represent between two white colors using 10-bit PQ.

The second question is how different the chromaticity of the two colors is. Here it is reasonable to assume that the small change from 0 to 4 is drowned out by the red component of 4000. In more detail, according to Larson [18], if the change in chromaticity as measured in the CIE representation is smaller than 0.5/410, it is well below the visual threshold. In our case the difference in is about 0.45/410 and the difference in is about 0.07/410, thus neither should be visible.

We now look at what happens to the two colors during subsampling. First we convert these two colors to 10-bit Y’CbCr: YCbCr1 = (298, 627, 898) and Y’CbCr2 = (436, 552, 802). In traditional subsampling, Cb and Cr are averaged while Y’ is kept intact. This would yield YCbCr1 = (298, (627+552)/2, (898+802)/2) = (298, 585.5, 850) for the first color and Y’CbCr = (436, 585.5, 850) for the second color. When converting back to RGB we see that the colors have changed dramatically in brightness; RGB1rec = (1927, 0.03, 45) and RGB2rec = (8339, 2.2, 216). This change from RGB1 to RGB1rec is clearly visible, and so is the one from RGB2 to RGB2rec. There is also an error in chromaticity; the biggest deviation of is in the of the first pixel, this error is 0.25/410.

Yet since we could not tell the original two colors from each other, we could just as well have used the same value for both pixels, such as RGB1. In this case, the Y’CbCr representation prior to subsampling would be the same; YCbCr1 = (298, 627, 898) = YCbCr2 which means that the subsampling/upsampling step would not affect it. The reconstructed colors RGB1 and RGB2 would be equal to RGB1 and thus indistinguishable from the originals and no luminance artifacts would occur.

Better still, we could set both colors to be the average (RGB1+RGB2)/2 = (4000, 2, 100), so that not all of the error is in RGB2. In this case the worst chromaticity error is for the coordinate of the second pixel, it is 0.23/410. Note that this is smaller than what should be visible and also smaller than the chromaticity error from traditional subsampling of 0.25/410.

In summary, by averaging the two pixels prior to subsampling, the error in luminance has been eliminated while the error introduced in chromaticity is still below what is visible, and even smaller than what would have been the result from regular processing.

Just averaging all pixel pairs in an image would not be a good idea since that would just be equivalent of lowering the resolution. However, as can be seen in the above example, there are cases where averaging helps finding a subsampled representation that is better at preserving both the luminance and the chrominance of the original signal. In this contribution we present a method for doing so in a systematic way without the drawback of lowered resolution.

# Detailed Description

Assume we have a color . How much can we change, say, the green component, without the human visual system noticing that it is a different color? One way to answer that is to ask the question, what is the lowest and highest values and of the green component G that we can choose so that the new color (4000, G, 100) has a chromaticity that deviates from that of the original color with less than 0.5/410?

Thankfully, it is possible to calculate these two values analytically from the representation of the original color, as can be seen in Appendix A.

We also want to make sure that the luminance is not affected too much. Here we can ask, what is the highest and lowest values and of the green component G that we can choose so that the new color (4000, G, 100) has luminance which does not alter the value of more than 0.5/(940-64) = 0.5/876? Also these two limits are possible to calculate analytically, as is shown in Appendix B.

By combining these restrictions using

we get for each pixel in the image an interval so that if we keep the green component inside this interval, the pixel will have a chromaticity and a luminance that is indistinguishable from that of the original pixel.

The first step of the method is therefore to calculate and as described above and in Appendix A and B. The second step is to low-pass filter the image in the x- and y- direction. We have used a 5 tap box filter, [1 1 1 1 1]/5. After this, the green component of each pixel is clamped (clipped) to its allowed interval . The image is again low-pass filtered with the 5-tap box filter and then clipped a second time, using the same interval as the first time.

After this follows an equivalent treatment of the blue and red components.

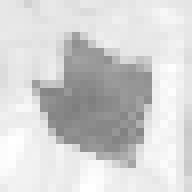
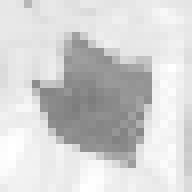
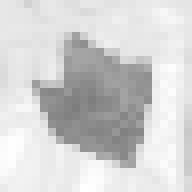
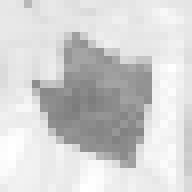
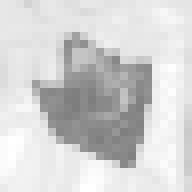
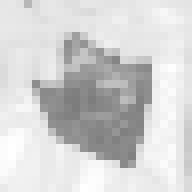
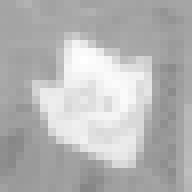
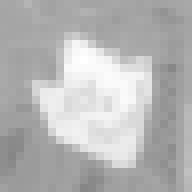
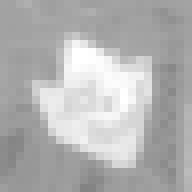
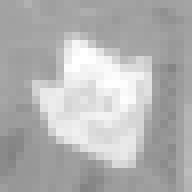
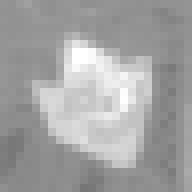
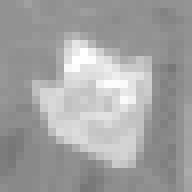
As a last step, the luminance of the original pixel is transferred to the processed pixel. This is done by converting the processed pixel to CIE1931 coordinates , and then combine this with the original luminance to get the representation of the final pixel,

The final pixel is then converted to RGB. After this, the RGB444 output image is processed as normally using Anchor 3.2, i.e., the subsequent step is luma adjustment.

# Results

## Chroma

In Figure 3 we show the Cb and Cr components of two consecutive frames, first using Anchor 3.2 processing (left) which is equivalent to Anchor 1.0 processing with respect to chroma. The next chroma pair has been processed using the method presented here, which we call chroma adjustment, followed by Anchor 3.2 processing (middle). The allowed deviation for and was set to , and the allowed deviation in was set to . This means that the RGB444 image before and after chroma adjustment should be perceptually identical.

  
  
 frame 389 frame 390 frame 389 frame 390 frame 389 frame 390  
*Figure 3. Top row: Cb, Bottom row Cr. Left: Anchor 1.0/3.2 Middle: Chroma adjustment with =0.5/410 and =0.5/876. Right: Chroma adjustment with =2/410 and =1/876. Content is BT.709 in a BT.709 container.*

As can be seen in the middle part of Figure 3, the chroma components are smoother and more similar across frames, which should help prediction.

In practice it seems it is possible to allow slightly higher values of and . In the rightmost part of Figure 3 we have used a value of of 2/410 and a value of of 1/876. This seems to give even better results without any visible artifacts.

## Luma

Since the chroma becomes smoother due to the processing, subsampling and upsampling will affect it less. This means that the luma adjustment method needs to work less hard to make sure that the luminance is correct, which in turn brings a smoother luma in areas of saturated colors. This can be seen in Figure 4. In the left part of the image, luma is shown for two frames processed using Anchor 1.0, i.e., without either luma adjustment or chroma adjustment. As can be seen the luma is very unsmooth. The middle shows processing using Anchor 3.2, i.e., using luma adjustment but not chroma adjustment. As can be seen this provides a smoother luma, however the smoothest version is when both chroma adjustment and luma adjustment are used (right).

  
 Anchor 1.0 (no adjustment) Anchor 3.2 (luma adjustment) chroma adjustment + luma adjustment

*Figure 4. Luma (Y’) of two consecutive frames. Left: Anchor 1.0. Middle: Anchor 3.2. Right: Chroma adjustment (=2/410 and =1/876) followed by Anchor 3.2 processing. Content is BT.709 in a BT.709 container.*

## Visual quality after compression

The effect of the smoother and more consistent chroma and luma should help lower the bit rate, especially if there are lots of areas with colors close to the gamut edge. It also increases the perceptual quality in these areas, as can be seen in Figure 5.

  
 original RGB444 anchor 1.0, 10340 kbps anchor 3.2, 9936 kbps chroma adj. + luma adj., 9422 kbps

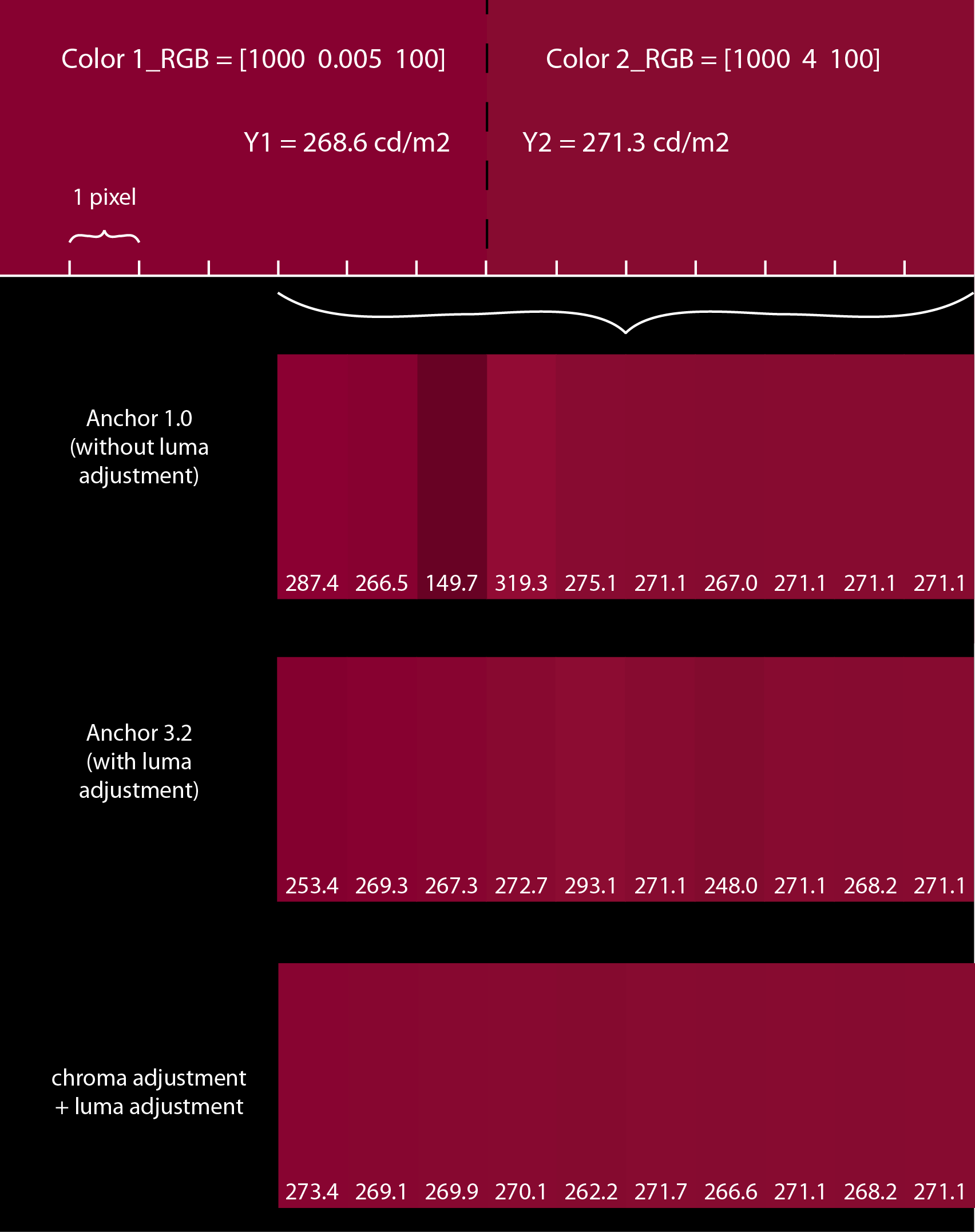
*Figure 5: Market sequence in BT.709 container, compressed using QP 21. BT.709 content in BT.709 container.*

## Behavior when filters are mismatching

Pu et al. points out that mismatching filters can diminish the effect of luma adjustment [19]. One step in the luma adjustment process is to upsample the chroma components to mimic what happens in the decoder. In many cases it cannot be known what filter is going to be used in the decoder, and if the filter used in the luma adjustment step differs from what is actually used, the method is not going to be able to fully adjust the luma to the correct luminance.

We have replicated the experiment from [19], and while individual pixels may get a luminance that is worse using luma adjustment than with Anchor 1.0, the luminance error is still reduced considerably in the pixels with the worst luminance errors, as can be seen in Figure 6 and 7. Just as in [19] we have used the TM filter ([21, -52, 159, 159, -52, 21]/256) in the luma adjustment step whereas the decoder uses the regular Lanczos-2 filter ([-1 9 9 -1]/16).

Since chroma adjustment will make neighboring chroma values more similar to each other for saturated colors, the choice of upsampling filter will matter less. Thus the problem with mismatching filters, which we believe is small to begin with, is further reduced, which is also shown in Figure 6. Figure 7 shows an example of real image data with the same type of mismatching filters (TM in the luma adjustment step and Lanczos-2 in the decoder).

  
*Figure 6. Replication of the experiment in [19]. Note how that even though the mismatching filter causes some imperfections for Anchor 3.2, it is still much better than not using luma adjustment. Chroma adjustment plus luma adjustment reduces these imperfections further. Values are in BT.2020 color space.*

  
 original RGB444 Anchor 1.0 Anchor 3.2 (luma adj.) chroma adj. + luma adj.  
 with mismatching filters with mismatching filters

*Figure 7. From left to right: Original RGB444, Anchor 1.0, Anchor 3.2 with TM in luma adjustment and Lanczos-2 in decoder, Chroma adjustment plus Luma Adjustment. Tonemapping has been performed on all images. Content is BT.709 in a BT.709 container.*

## Objective Results

For the objective results, chroma adjustment has first been used with =0.5/410 and =0.5/876. Both the regular sequences (in BT.2020 container) and A’ sequences (in BT.709 container) are used. For the BT.2020 container sequences, the changes are small, which is to be expected when all colors are far from the color gamut boundary.

***Table 1: BT.2020 container, 1 6 1 downsampling filters, =0.5/410 and =0.5/876***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sequence | tPSNRX | tPSNRY | tPSNRZ | tPSNRXYZ | OSNRXYZ | DE100 | MD100 | L100 |
| FireEater | -0.2% | -0.1% | 0.7% | 0.1% | 0.5% | 0.1% | 1.1% | 0.2% |
| Market | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 1.0% | -41.7% | 0.0% |
| SunRise | -0.1% | 0.0% | 1.0% | 0.3% | 0.7% | 0.3% | 20.8% | 0.1% |
| BikeSp 1 | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 1.1% | -15.1% | -0.1% |
| BikeSp 2 | -0.1% | -0.1% | 0.3% | 0.0% | 0.0% | 0.7% | -4.3% | -0.2% |
| GarageExit | -0.2% | -0.2% | 0.5% | 0.1% | 0.2% | 1.2% | 6.9% | -0.1% |
| ShowGirl | -0.2% | -0.1% | 0.3% | 0.0% | -0.8% | -0.1% | 12.7% | -0.1% |
| MagicHour1 | -0.1% | -0.1% | -0.2% | -0.1% | -0.1% | -0.1% | -19.5% | -0.1% |
| MagicHour2 | 0.0% | 0.0% | -0.1% | -0.1% | 0.1% | 0.1% | -4.9% | -0.1% |
| MagicHour3 | -0.2% | -0.1% | 0.0% | -0.1% | 0.1% | -0.5% | -0.6% | -0.1% |
| WarmNight1 | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 0.5% | 6.2% | -0.1% |
| WarmNight2 | 0.1% | 0.0% | 0.2% | 0.1% | 0.2% | 0.5% | -1.3% | 0.0% |
| BalloonFest | -0.1% | -0.1% | 0.7% | 0.2% | 0.6% | 1.5% | -22.6% | -0.1% |
| Hurdles | 0.0% | -0.1% | 0.0% | -0.1% | 0.0% | 2.9% | 4.9% | -0.1% |
| Start | -0.3% | -0.3% | 0.1% | -0.2% | -0.1% | 0.9% | -8.1% | -0.4% |
| **All** | **-0.1%** | **-0.1%** | **0.2%** | **0.0%** | **0.1%** | **0.7%** | **-4.4%** | **-0.1%** |

For the BT.709 container sequences, the changes are bigger as shown in the below diagram. Here consistent gains of -0.5% or better are obtained for all metrics.

***Table 2: BT.709 container, 1 6 1 downsampling filters, =2/410 and =1/876***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sequence | tPSNRX | tPSNRY | tPSNRZ | tPSNRXYZ | OSNRXYZ | DE100 | MD100 | L100 |
| FireEater | -1.7% | -1.6% | -1.2% | -1.6% | -1.3% | -1.4% | -0.9% | -1.0% |
| Tibul | -0.4% | -0.2% | -0.3% | -0.3% | -0.5% | -0.2% | -2.9% | 0.1% |
| Market | -1.2% | -1.2% | -1.2% | -1.2% | -1.1% | -0.6% | -11.3% | -1.1% |
| **All** | **-0.8%** | **-0.7%** | **-0.7%** | **-0.8%** | **-0.7%** | **-0.5%** | **-3.8%** | **-0.5%** |

The best visual gains for the BT.709 container case however are obtained when using stronger filtering; =2/410 and =1/876 combined with 1 2 1 downsampling. This also gives considerably larger objective gains for the BT.709 container case, especially for luma.

***Table 2: BT.709 container, 1 2 1 downsampling filters, =2/410 and =1/876***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sequence | tPSNRX | tPSNRY | tPSNRZ | tPSNRXYZ | OSNRXYZ | DE100 | MD100 | L100 |
| FireEater | -4.3% | -4.1% | -1.0% | -3.6% | -2.3% | -3.1% | -2.7% | -2.9% |
| Tibul | -1.1% | -0.5% | 0.1% | -0.7% | -0.8% | -0.1% | -4.3% | 0.6% |
| Market | -4.7% | -4.8% | -1.7% | -3.6% | -3.3% | 2.1% | 95.0% | -4.7% |
| **All** | **-2.5%** | **-2.4%** | **-0.6%** | **-2.0%** | **-1.6%** | **-0.3%** | **22.0%** | **-1.7%** |

# Conclusion

This contribution has presented a pre-processing step called chroma adjustment that can be applied before the Anchor 3.2 processing chain. It has the effect of making both luma and chroma smoother and therefore less costly to encode in terms of bit rate. If the filtering parameters are set conservatively, i.e., =0.5/410 and =0.5/876, it should not be possible even in principle to see any degradation of the image due to the chroma adjustment step. In practice, if the content reaches closer to the color gamut (such as BT.2020 content in a BT.2020 container or BT.709 in a BT.709 container), stronger filtering such as =2/410 and =1/876 gives better objective results, and also looks more subjectively pleasing. The filtering of a color component is carried out by first calculating boundaries. This can be done in an analytic manner. The second step is a smoothing of the (floating point) data while clamping to the calculated boundaries. It should be noted that the example images in this contributions are BT.709 in a BT.709 container. This is very similar to what happens when BT.2020 material fills the BT.2020 container. Taking the market sequence and just assuming the RGB samples are in BT.2020 confirms that this is indeed the case.

# References

[1] E. François, P. Lopez, Y. Olivier, “About using a BT.2020 container for BT.709 content”, ISO/IEC (MPEG) document m35255, Strasbourg, Oct 2014

[2] J. Ström, J. Samuelsson, M. Pettersson, K. Andersson, P. Wennersten, R. Sjöberg, “Ericsson's response to CfE for HDR and WCG”, ISO/IEC (MPEG) document m36184, Warsaw, Poland, June 2015.

[3]A. M. Tourapis, Y. Su, D. Singer, “Enhanced Luma Adjustment Methods”, JCT-VC JCTVC-W0052, San Diego, Feb 2016.

[4] C. Rosewarne and V Kolesnikov, “CE1-related: LUT-based luma sample adjustment”, JCT-VC, JCTVC-W0056, San Diego 2016.

[5] A. Norkin, “Closed form HDR 4:2:0 chroma subsampling (HDR CE1 and AHG5 related)”, JCT-VC JCTVC-W0107, San Diego, Feb 2016.

[6] J. Ström, J. Samuelsson and K. Dovstam, ”Luma Adjustment for High Dynamic Range Video”, Proceedings of the IEEE Data Compression Conference (DCC), Snowbird, March 2016.

[7] A. Norkin, “Fast algorithm for HDR color conversion”, Proceedings of the IEEE Data Compression Conference (DCC), Snowbird, March 2016.

[8] J. Ström, J. Samuelsson, K. Andersson, P. Hermansson, “Modified Linearization of Luma Adjustment”, JCT-VC, JCTVC-X0036, Geneva, May 2016.

[9] C. Rosewarne and V Kolesnikov, “AHG13: Further results for LUT-based luma sample adjustment”, JCT-VC, JCTVC-X0054, Geneva, May 2016.

[10] A. Norkin, “On closed form HDR 4:2:0 chroma subsampling (AHG13 related)”, JCT-VC, JCTVC-X0072, Geneva, May 2016.

[11] J. Ström, “AhG-13 related: Multi-LUT Luma Adjustment Implementation”, JCT-VC, JCTVC-Y0030, Chengdu, Oct 2016.

[12] A. Norkin, “On HDR 4:2:0 chroma subsampling (AHG13 related)”, JCT-VC, JCTVC-Y0033, Chengdu, Oct 2016.

[13] C. Rosewarne and V Kolesnikov, “AHG13: Further results for luma sample adjustment”, JCT-VC, JCTVC-Y0034, Chengdu, Oct 2016.

[14] J. Ström, K. Andersson, M. Pettersson, P. Hermansson, J. Samuelsson, A. Segall, J. Zhao, S-H. Kim, K. Misra, A. M. Tourapis, Y. Su and D. Singer, “High Quality HDR Video Compression using HEVC Main 10 Profile”, Proceedings of the IEEE Picture Coding Symposium (PCS), Nuremberg, Dec 2016.

[15] A. Norkin, “Fast algorithm for HDR video pre-processing”, Proceedings of the IEEE Picture Coding Symposium (PCS), Nuremberg, Dec 2016.

[16] Call for Evidence (CfE) for HDR and WCG Video Coding, ISO/IEC (MPEG), <http://mpeg.chiariglione.org/standards/exploration/high-dynamic-range-and-wide-colour-gamut-content-distribution/call-evidence>.

[17] J. Samuelsson, C. Fogg, A. Norkin, A. Segall, J. Ström, G. J. Sullivan, P. Topiwala, A. Tourapis, “Conversion and Coding Practices for HDR/WCG Y′CbCr 4:2:0 Video with PQ Transfer Characteristics Draft 3”, JCT-VC, JCTVC-Y1017, Chengdu, Oct 2016.

[18] G. W. Larson, “The LogLuv Encoding for Full Gamut, High Dynamic Range Images”, Journal of Graphics Tools, 3(1):15-31, 1998.

[19] F. Pu, T. Lu, P. Yin, T. Chen, W. Husak, “AHG13: on Luma Adjustment”, JCT-VC, JCTVC-X0043, Geneva, May 2016.

# Appendix A

Assume we have a color . How much can we change, say, the green component, without the human visual system noticing that it is a different color? One way to answer that is to say that we can change the green with as long as the new color (4000, 2+, 100) has a chromaticity that deviates from that of the original color with less than 0.5/410. To make it more general, assume we have already changed red and blue so that we are interested in for what values of the color we will still get a chromaticity that deviates less than 0.5/410 from the chromaticity of the original color in both and ?

The coordinates and are calculated from , so we first convert the original color to :

where is the 3x3 matrix used to calculate from BT.2020. (If a BT.709 container is used, the corresponding matrix from BT.709 to XYZ is instead used.) The original coordinate is calculated as

Let us denote by the -component of the resulting color . It is allowed to differ from by , and can be calculated as

We can calculate as

Hence we can rewrite the expression for as

which equals

which can in turn be written as

where , , and are constants that does not depend on the pixel and where . It is now possible to solve for , giving

Assume we allow a deviation of of either or . We can now calculate the two that will give these two devations:

The smallest allowed value of green is now and the largest allowed value of green is , where G is the green component of the original color.

A similar expression is possible for

which brings a new restriction on to be in the interval ]. To make sure that both and are in the allowed range we have to take the most conservative value:

# Appendix B

Assume that we have an original color and an already processed color where the red and blue components are different , . What values of can we use so that the luminance appears indistinguishable from the original?

In this case, we say that two colors with luminances and are indistinguishable luminance-wise if .

The value to match is therefore , where is the luminance of the original pixel, , and where , and are constants from the transformation matrix from to in Appendix A. We thus have

and likewise for the luminance of the processed pixel

We set which gives

which equals

where is a helper variable equal to . We now investigate when the difference between and is equal to the maximally allowed . This happens when

which is equal to

Taking of both sides gives

which equals

We can now solve for

Similarly we have

and we can finally calculate the limits for the green component using and .

Note that if a fixed is used for all pixels, is a one-dimensional function of which can be efficiently implemented as a look-up table (LUT).

# Patent rights declaration(s)

**Ericsson may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**