Hue-Preserving Color Transforms for LED Wall Virtual Production Workflows

Michael D. Smith, Engineer

Wavelet Consulting LLC, 321 N Pass Ave #145, Burbank, CA 91505, miksmith@attglobal.net

Michael Zink, Vice-President, Emerging & Creative Technology

Warner Bros., 4000 Warner Blvd, Burbank, CA, 91522, Michael.Zink@warnerbros.com

Written for presentation at the

SMPTE 2021 Annual Technical Conference & Exhibition

Abstract. *Virtual Production using LED wall display technology is gaining popularity in the entertainment industry to produce motion pictures, episodic television, live broadcast and esports content. This new paradigm typically uses one or more large LED displays that contains millions of individual Light Emitting Diodes (LED) that are used to display a virtual background and/or foreground objects from a virtual scene that is simultaneously captured by the digital photographic camera on set, resulting in so-called "in-camera visual effects". In a virtual production workflow, digital cameras capture the actors and objects on the set in the foreground while simultaneously capturing the image shown on the LED wall behind the actors. Realtime video game rendering engines update the image shown on the LED wall image to compensate for changes in camera location, camera pose and focal length. Modern digital camera workflows typically include color transforms that were not designed to accurately render the large areas of saturated colors that are possible when capturing the image shown on an LED wall. Some examples of a hue shift that can occur in typical workflows are blue to cyan, red to pink, red to orange and green to yellow.*

We found these hue shifts can occur dynamically while racking focus to and from the LED wall, and also statically when the LED wall is kept out of focus, which is a common technique that is used to minimize moiré artifacts. This paper explores a simple modification of these common color transforms to preserve the hue of the scene while also creating a similar Look of the existing color transforms.

Keywords. *LED Wall, Virtual Production, Color Transform, Lookup Table, LUT, hue shift, hue-preserving, rendering.*

(The SMPTE disclaimer is on a footer on this page, and will show in Print Preview or Page Layout view.)

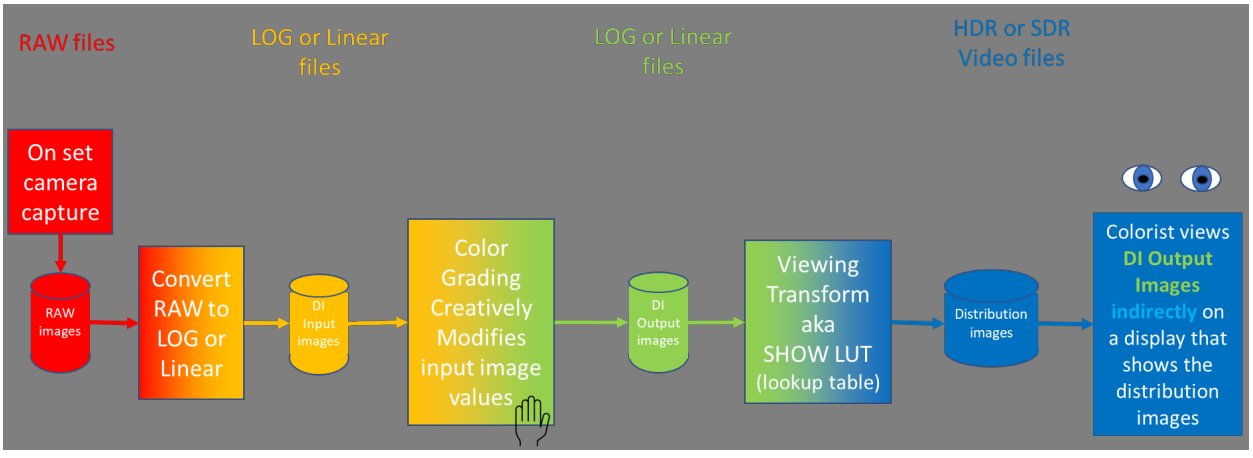
Introduction

Color transforms used in modern digital camera workflows do not typically preserve the hue of the scene while rendering from the captured scene-referred RAW or LOG formats to video distribution formats like BT.709 or BT.2100. These hue shifts are most noticeable when the imagery from the scene contains bright saturated colors. In traditional production, bright saturated colors are not commonly captured by cameras, except for neon and other practical lights, such as car brake lights or traffic lights in outdoor scenes. If green-screens are used to create virtual backgrounds after camera capture, the green-screen imagery is replaced with VFX imagery that is adjusted so the composite ("comp") looks as intended through a viewing LUT, allowing the compositor to artistically compensate for any hue-shifting inherent in the viewing transform. These are some reasons that the lack of hue-preservation has gone unaddressed. In contrast to these legacy workflows, it is very easy to capture large areas of bright saturated colors when using a LED wall in a virtual production workflow. The bright saturated colors that can be emitted from LED walls can stimulate hue-shifts when processing the footage through common digital camera viewing LUTs.

Common rendering color pipelines do not preserve hue

Common camera rendering color pipelines, like ARRI has documented in SMPTE RDD 31 Annex B [1], are not typically hue-preserving. Transforms that do not preserve hue have a long history in our industry and have been the basis of an accustomed look of photography with roots in the legacy of photochemical film reproduction. Many digital camera rendering algorithms were designed to reproduce the look that was achieved by the photochemical film cameras that those digital cameras were designed to replace. These non-hue preserving digital camera rendering algorithms have of course been used to produce beautiful award-winnning motion pictures and episodic television that have been enjoyed worldwide.

While such rendering transforms that do not preserve hue have been widely accepted by the industry, if they are used to render images captured using virtual production stages with LED wall backgrounds, there is the potential for strong unexpected hue shifts. Our current focus is on the changes in hue that occur in the viewing transform, which is just one part of multi-step modern color pipeline illustrated in Figure 1.



Viewing transforms

Viewing transforms usually include a few different components that are usually consolidated into a single 3D lookup table (LUT) that performs all the operations simultaneously. The important components in a viewing transform are:

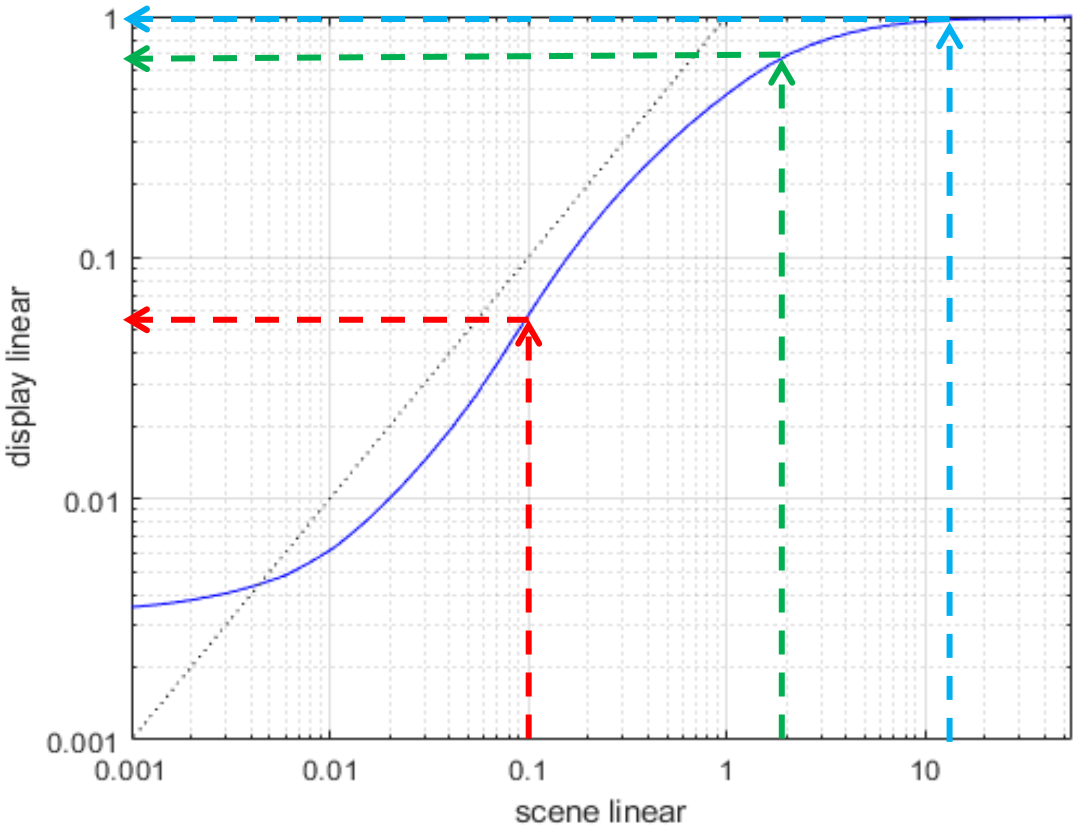
1. Creative Look – global adjustments to scene-referred imagery to achieve creative goals like changes to saturation, contrast, gain, etc.
2. Rendering – scene-referred to display-referred format conversion usually includes tone-curve processing that boosts shadow detail, increases midtone contrast and rolls off highlight details.
3. Display Transform - prepares rendered imagery for display on a specific type of video device and usually includes encoding primary conversion (rendering primaries to display encoding primaries), formatting for a video interface or distribution-file includes inverse-EOTF aka “gamma correction” and full-to-legal conversion if necessary. The display transform can also include steps to compensate for viewing environments or display gamut reduction.

How is hue altered by viewing transforms?

We have identified two high-level causes of hue shifts that happen viewing transforms:

1. Applying the rendering tone-curve independently to R, G and B channels
2. Clamping out-of-gamut colors to the display gamut boundary

When the rendering tone curve is applied independently to R, G and B channels, the RGB color ratios can change, which modifies the hue and saturation of the output. Figure 2 shows an example tone curve with red, green and blue dotted lines representing the independent R, G, B channel processing of an example bright blue input color (R,G,B)=(0.2, 2.0, 12.0). The resulting output color is cyan (R,G,B)=(0.06,0.70,0.96).



In this example, the R/G ratio changes from 0.2/2.0 = 0.100 to 0.06/0.70=0.085, the G/B ratio changes from 2.0/12.0=0.167 to 0.70/0.96=0.729 and the R/B ratio changes from 0.2/12.0=0.017 to 0.06/0.96=0.063. The altered RGB ratios can also be expressed as simple red-green chromaticities (r,g) = R/(R+G+B), G/(R+G+B), for example the input color has (r,g) = (0.007,0.142) and the output color has (r,g) = (0.035,0.407). If we associated specific (x,y) chromaticities to the RGB primaries and white-point used in this example, we could compute the (x,y) chromaticity of the input and output colors and also compute a perceptual color difference like ΔE.

Clamping out-of-gamut colors to the display gamut boundary can also cause hue shifts in viewing transforms. Clamping is typically applied independently to R, G and B channels and forces each R, G, and B value to be in the range 0.0-1.0. Clamping is a common step before inverse-EOTF encoding that involve power functions that are undefined for negative input values when the power function's exponent is not an integer, for example, pow(-0.1,1/2.4) is undefined on many platforms. Additionally, many LUT processors don't support input values outside the range 0.0-1.0 and use a clamp() operator on the input R, G, B before doing the LUT interpolation.

The following example illustrates how a hue-shift can occur due to the clamp() operation applied after an encoding primary conversion. Suppose an orange input color (R,G,B)=(0.85, 0.50,

0.03) is processed by an Arri Wide Gamut (AWG) to BT.709 encoding primary conversion according to the following equations:

$$Rout = 1.62 * R + -0.54 * G + -0.08 * B$$

$$Gout = -0.07 * R + 1.33 * G + -0.26 * B$$

$$Bout = -0.02 * R + -0.23 * G + 1.25 * B$$

This results in BT.709 RGB values (1.10, 0.60, -0.09), the presence of RGB values outside the range 0.0-1.0 indicates the input AWG color is out-of-gamut in BT.709. If a clamp() operator is applied, the result is (1.0, 0.60, 0.00). In this example, the rg-chromaticity is altered by clamp(). The unclamped BT.709 color has (r,g)=(R / [abs(R)+abs(G)+abs(B)], G / [abs(R)+abs(G)+abs(B)]) = (0.615, 0.335) while the clamped BT.709 color has (r,g)= (0.625, 0.375).

Modification to color pipelines to preserve hue

The non-hue-preserving color transform described in SMPTE RDD 31 Annex B has three basic steps:

1. apply a sigmodal tonecurve independently to each R, G and B ARRI Wide Gamut LogC encoded data. The tonecurve is shown in Figure 3.
2. Apply a 3x3 matrix to convert AWG primaries to BT.709 primaries with some additional desaturation.
3. Apply inverse-EOTF “gamma-correction” to convert the BT.709 display linear signal to a BT.709 video non-linear signal.

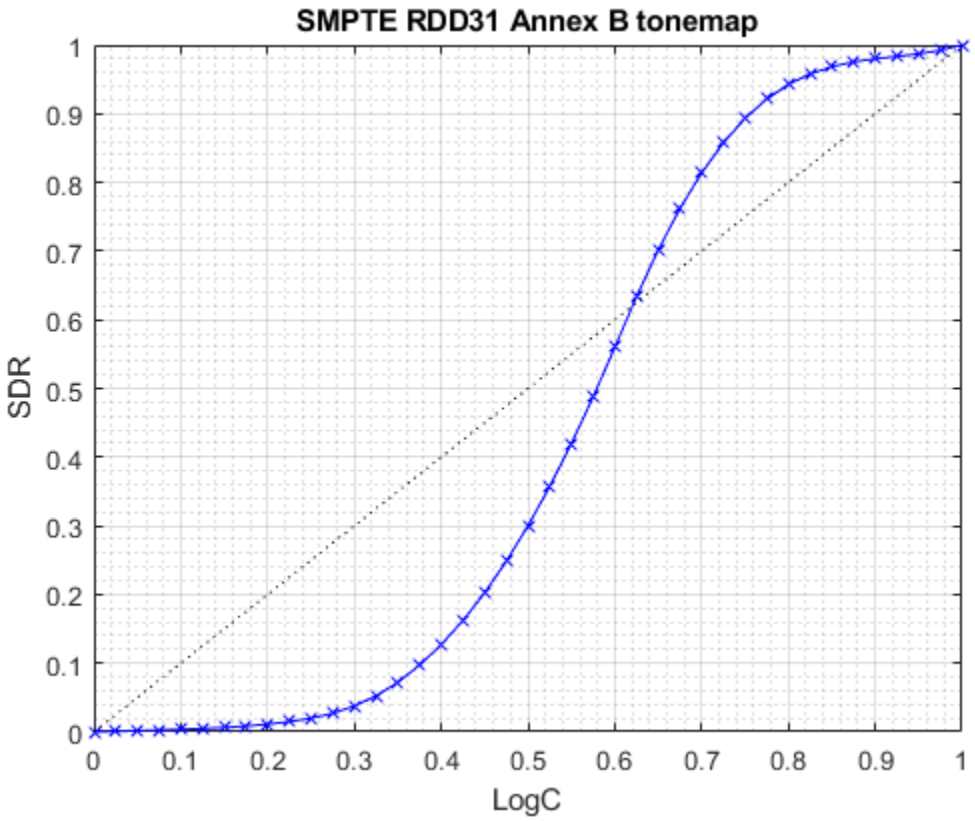


Figure 3 - sigmodal tonemapping curve

The RDD 31 rendering algorithm can be modified to preserve hue using norm-based tone curve processing [2] by following these steps:

1a) Compute norm maxRGB_logc = max(R_logc, G_logc, B_logc)

1b) Apply tonemap to maxRGB_logc

maxRGB_display_linear_awg = tonemap(maxRGB_logc)

1c) Compute scaling_factor = maxRGB_display_linear_awg / linearize(maxRGB_logc) where linearize() converts from LogC to scene-linear according to equation 5.6 in RDD31.

1d) Compute linear AWG values

R_awg=linearize(R_logC)

G_awg=linearize(G_logC)

B_awg=linearize(B_logC)

1e) Multiply AWG red, green, and blue values by scaling_factor

R_display_linear_awg = scaling_factor * R_awg

G_display_linear_awg = scaling_factor * G_awg

B_display_linear_awg = scaling_factor * B_awg

After step 1a-1e are complete, the same steps 2 and 3 as described above can be followed.

Examples of Hue-Preserving Color Transforms using imagery from LED Wall Virtual Production stage

A test scene was configured with colorful foreground objects, this paper's co-author Michael Zink sitting next to a gray card and color checker chart in front of an LED wall. The LED Wall in the background is lighting the scene from both the back and the side. A key light was also configured to light the scene from the front on camera-left. The LED wall luminance was set to 360 nits and the wall was set to operate with its native primaries in Acheivable Gamut mode. An Arri Alexa LF camera was used to capture footage of the test scene and was set to ISO 800 and shutter angle 180-degrees. The camera was configured with an ARRI prime with aperature T2.8.

Figure 4 illustrates the differences that result from using the default ARRI LUT described in RDD 31 Annex B and the hue-preserving modification described in the previous section. The hue-shift that results from the RDD 31 Annex B viewing transform is most visible with the bright red and blue backgrounds that change to pink and cyan respectively.

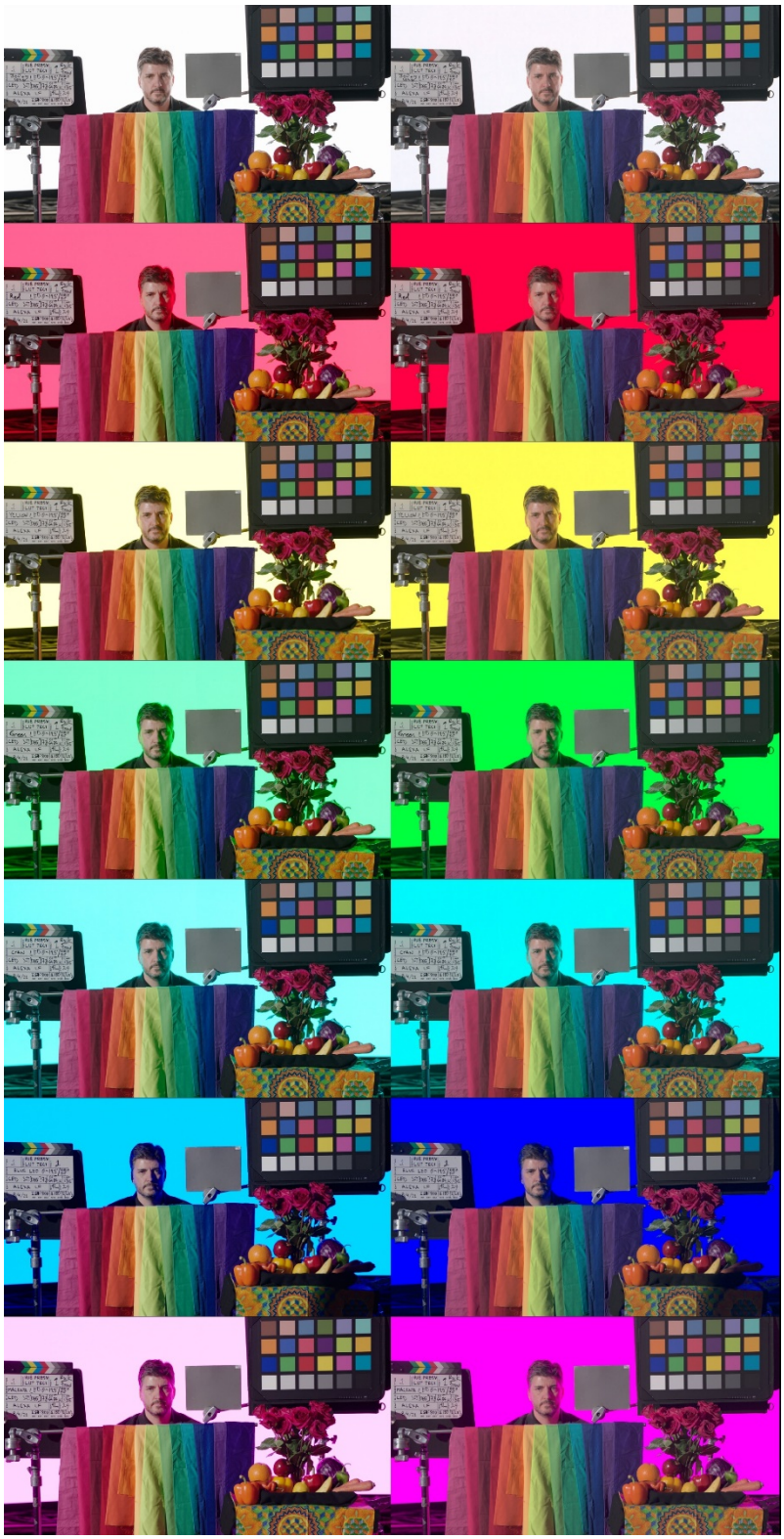


Figure 4 - Comparison of two different viewing transforms applied to the same LogC image data, the left side corresponds to SMPTE RDD31 Annex B and is not hue-perserving. The right side is hue-preserving and implements the modified rendering process described in this paper.

Figure 5 - cropped section of images showing face lit from the side with light from LED wall with SMPTE RDD31 Annex B viewing transform

Figure 6 - cropped section of images showing face lit from the side with light from LED wall with MaxRGB hue-preserving viewing transform

LED walls are also often used for lighting and reflections in Virtual Production. Preserving hue between the background and objects lit with colored light from the LED wall helps reduce discrepancies that break the consistency between visual effects and photography of real objects. An example illustrating this is the colored lighting on the right side of the faces shown in Figure 5 and Figure 6. With the non-hue preserving viewing transform in Figure 5 there is a discrepancy between the hue on the side of the face in colored lighting and the background, that is especially notable in the red and blue backgrounds that appear pink and cyan while the face appears red and blue. In Figure 6 with the hue-preserving transform, both the face in colored lighting and the background appear have consistent color.

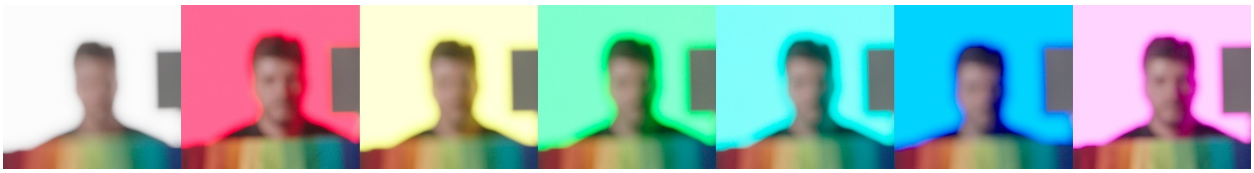


Figure 7 – cropped area of images showing face with RDD 31 Annex B viewing transform, out of focus

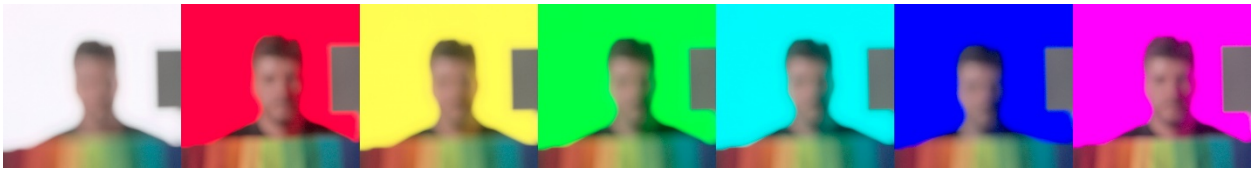


Figure 8 - cropped area of images showing face with MaxRGB hue-perserving viewing transform, out of focs

In regions that are out of focus, the light levels of an area become blurred together. In the independent R, G, B tone curve rendering that is not hue preserving, the hue changes with the intensity of the area. Bright areas have bigger hue shifts than darker areas, and therefore the areas that have large contrasts in light level, for example the bright background versus the dimmer foreground show hue shifts in the out of focus bokeh. This hue shift appears as a halo around foreground objects. For example, in Figure 7 with the non-hue-preserving viewing transform, the head is surrounded by a red halo that transitions to a pink background. The hue-perserving norm-based tone curve rendering shown in Figure 8 does not change the hue

dependent on the intensity, thus when it goes out of focus and bright and dim areas blur together, the hue in those areas do not change.

Hue-perserving Viewing Transform Alternatives

While we have focused on a hue-perserving viewing transform using a max() norm, there are other alternatives, as identifed in [2]. The results of several alternatives are shown in Figure 9.

Figure 9 - Examples of hue-preserving viewing transfrom alternatives, from top to bottom, RDD 31, RDD 31 with hue_restore_dw3() [3] applied after the independent RGB tone curve and again after the AWG to BT.709 primary conversion, MaxRGB-norm, MaxRGB-norm with hue_restore_dw3() applied after AWG to BT.709 primary conversion, Power-norm (5ths over 4ths), Power-norm (5ths over 4ths) with hue_restore_dw3() applied after AWG to BT.709 primary conversion

Conclusion

Given that many LED walls are able to show very bright and saturated colors and the new paradigm of LED-based Virtual Production dicates that the camera directly capture the image shown on the LED wall, it is expected that the characteristics of the captured image will be different than images captured on traditional physical sets. We studied the performance of viewing transforms when handling bright saturated colors from LED walls and how a commonly used viewing transform results in undesirable hue shifts. We confirmed that when a viewing transform was specifically designed to preserve hue, it handles the bright saturated color in a

more predicable way. Thus we expect hue-perserving viewing transforms to be relevant in this new workflow paradigm.

References

[1] "RDD 31:2014 - SMPTE Registered Disclosure Doc - Deferred Demosaicing of an ARRIRAW Image File to a Wide-Gamut Logarithmic Encoding," in RDD 31:2014 , vol., no., pp.1-17, 26 Sept. 2014, doi: 10.5594/SMPTE.RDD31.2014.

[2] G. Demos and D. Walker, "Core Color Rendering Algorithms for High Dynamic Range Display," in SMPTE Motion Imaging Journal, vol. 127, no. 9, pp. 1-9, Oct. 2018, doi: 10.5594/JMI.2018.2810022.

[3] restore_hue_dw3() function – on lines 26-84 of file /v0.7.1/transforms/ctl/utilities/ transforms-common.ctl, available in ACES 0.71 – accessed online https://github.com/ampas/aces-dev/blob/v0.7.1/transforms/ctl/utilities/transforms-common.ctl