Sculpting Color Spaces

Yanli Zhao
MPC R&D
yanli-z@mpcfilm.com

Darryl Gouder
MPC R&D
darryl-g@mpcfilm.com

Rob Pieké
MPC R&D
rob-p@mpcfilm.com

(a) Despilling (b) Before (c) After

Figure 1: Despilling. (a) shows a deformed 3D color space via sculpting manipulation; (b) and (c) demonstrate the result.

ABSTRACT

Color correction with a long chain of keyers and math operators, even in the hands of experienced artists, often induces artifacts such as muddy colors or malformed edges. Inspired by tools which display a 3D color histogram [COL 2007], and Flame's Color Wrapper [WRA 2019], we embarked on building a user-friendly 3D color space sculpting toolset which allows a user to make complex and elegant color alterations quickly, intuitively and effectively.

In this paper, we will show how smooth transitions can be achieved by our tool through a combination of soft-selection, LUT auto-filling, and tetrahedral interpolation. Multiple approaches for interactive selection and highlighting were introduced to overcome the inaccurate and esoteric manipulation when applying similar 3D based tools to point clouds of colors. Our tool has been robustly integrated with the color correction pipeline through the ability to import and export industry standard 3D LUT files. We have found success for our tool in a number of color-manipulation-related tasks, such as: denoising, despilling, and standard grading.

CCS CONCEPTS

• **Computing methodologies** → **Image processing**; *Volumetric models*; • **Human-centered computing** → *Graphical user interfaces.*

KEYWORDS

color correction, image processing, human-computer interaction

SIGGRAPH '19 Talks, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6317-4/19/07.
https://doi.org/10.1145/3306307.3328169

ACM Reference Format:
Yanli Zhao, Darryl Gouder, and Rob Pieké. 2019. Sculpting Color Spaces. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328169

1 CONTEXT

Color grading film projects via node graphs with numerous keyers and math operators results in enormous labor costs, and requires advanced skills to meet the ever increasing needs of photo-realism. Unavoidable conflicts from different keyers introduce edges and muddy colors. While some 3D color histogram based tools have support for interaction, we've observed them to be esoteric to users, making it difficult to achieve complex effects.

2 IMPLEMENTATION

Our tool begins by transforming a 2D image into a 3D color histogram in RGB or HSV coordinates (referred to in this paper as a *cloud*). As high dynamic range (HDR) is widely used in the VFX industry, we provide control to either interpret color values as raw linear data, or can apply tone mapping operations (for example converting to Rec709) first. The ultimate aim is to provide a cloud whose shape is practical for artists to work with.

From this cloud, a sparse bounding hexahedral geometry (referred to in this paper as a *shell*) is generated. Similar to other particle-grid techniques in VFX (such as fluid simulation), we maintain a relationship between the individual points in the cloud and their location in the shell. This allows us to leverage existing deformation tools which rely on traditional meshes with explicit topology.

2.1 Smooth Transition

As alluded to, manipulations of the shell drive changes in the positions of the points in the cloud and, as an indirect result, the corresponding colors in the image. Although each point in the cloud originates from an individual pixel (more specifically, its

color value), we do not directly use the original mapping to change pixel colors, as we found it difficult to maintain subtle transitions without visual artifacts when doing so.

Inspired by OpenColorIO [OCI 2003], we instead treat the manipulated shell as a new color space from which a 3D LUT is generated. Since most industry-standard formats for LUTs require a dense grid of color space data, the values of "non-existing points/cells" in our sparse shell are determined from a weighted-average interpolation of the k-nearest neighbors (referred to as *KNN interpolation*). This effectively works like a spring to distribute the manipulation to the whole LUT. The LUT is stored as a dense 3D array and applied to the image with tetrahedral interpolation [M. Kasson et al. 1993].

2.2 Interaction

To support accurate and interactive manipulation, we enable artists to make component selection on all three of: the shell, the cloud, and the original 2D image. When pixels from the image are selected, the corresponding points and cells will be selected automatically, ready for manipulation. Similarly, when points or cells are selected, the corresponding pixels in the image will be highlighted, giving the user can an instant and obvious preview of the areas which may be affected.

We also support masking to allow for further control and localization of the editing. For example, the user can change the color of one green tree in the image and leave the other similar colored trees as they are.

2.3 Collaboration

As mentioned in 2.1, manipulations are translated to a 3D LUT. This LUT can be exported to an industry standard color space format file and then used widely by traditional tools for other images, shots or projects. This promotes a workflow of performing manipulations on a handful of "key images" and applying the results broadly.

For further refinement, we support ingesting LUT files (irregardless of their origin - i.e., they need not have been created using our tool). From these, we can restore the deformed cloud and shell and continue to apply more manipulations.

3 APPLICATIONS

The above implementation clarifies how we build the shell and cloud. In addition to standard color grading, we outline two other workflows below.

3.1 Denoise & Despill

In the VFX industry, real shots frequently consist of noise and spilling blue/green color which brings difficulty for chroma-keying.

When considering the cloud, of such an image, the noise is often well localised in the color space, making it easy to isolate and operate on. Figure 2b exemplifies a noisy blue screen shot, which can be rectified with a simple scaling down along the red-green axis as shown in Figure 2a. This reduces the noise dramatically, as seen in Figure 2c. Note that the details of the transparent objects are intact. This denoised image is much easier to extract a key from.

After noise removal, blue color spilling in the foreground is still an issue as seen in Figure 1b. We can do "anti-blue" by translating blue points in the yellow direction along the blue-yellow axis as shown in Figure 1a. Similar to the noise, blue spilling is reduced in Figure 1c, without breaking detail and photo-realism. In production, anti-blue, -green and -purple is common and easy achieved by our tool.

(a) Denoising **(b) Before** **(c) After**

Figure 2: Denoise. (a) shows the 3D color space sculpting manipulation for denoising; (b) and (c) show the result.

3.2 Live action integration

A common challenge in VFX production is ensuring a visually-seamless integration of computer-generated content with real (i.e., filmed) images. It is not uncommon for said integration to look obviously "fake" or "wrong", without having an obvious explanation for *why* (and, by extension, how to fix it).

While in the previous section (3.1) we typically operate in a tone-mapped color space, we've found that comparing raw linear color space shells and clouds from CG content to those from real content can provide great intuition into the challenge just mentioned. The shapes and sizes of the respective clouds can identify mismatches in color balance, highlight intensities, etc. Further, by treating the shell of the real content as our "ground truth", we can warp and align our CG content manually or via deformers to achieve a more believable final composited image.

4 IMPACT AND FURTHER THOUGHTS

Our color space sculpting tool is a strong supplement for sophisticated image processing tasks, as it ensures exquisite transitions, simplifies node graph and reduces keyers' conflicts significantly.

We have recently shifted our focus to performance, observing that the performance of the KNN interpolation step in our process drops significantly if the resolution of our shell is very high. We are considering investigation of GPU implementations and/or other interpolation schemes.

ACKNOWLEDGMENTS

Special thanks go to Adam Valdez for not only his creative vision for this tool, but also for providing continuous artistic support and advice on production usage.

REFERENCES

2003. OpenColorIO. http://opencolorio.org.
2007. 3D Color Inspector/Color Histogram. https://imagej.nih.gov/ij/plugins/color-inspector.html.
2019. Color Wrapper Flame Autodesk Inc. https://knowledge.autodesk.com/support/flame-products/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Flame-ColourCorrecting/files/GUID-260FAB6D-E4B6-456C-B3B2-78103AE05CAC-htm.html.
James M. Kasson, Wil Plouffe, and Sigfredo I. Nin. 1993. Tetrahedral interpolation technique for color space conversion. *Proc. SPIE* 1909, 127–138. https://doi.org/10.1117/12.149035