Implementing the SGCK Gamut Compression

Shankhya Debnath

August 23, 2024

1 Introduction

SGCK (Spatial Gamut Compression for Key Color) is a gamut compression algorithm designed to fit colors within a target gamut, such as AdobeRGB, while maintaining perceptual quality. This document explains the mathematical background and implementation of the SGCK compression algorithm, along with an analysis of the results obtained from compressing the color gamut of an input image.

2 Mathematical Background

The SGCK algorithm compresses the color gamut by adjusting the lightness and chroma of colors in a controlled manner. The process can be summarized as follows:

2.1 Lightness and Chroma

In the CIELAB color space, a color is represented by the triplet (L^*, a^*, b^*), where L^* denotes lightness, and a^* and b^* represent the color-opponent dimensions. The chroma C^* and hue angle h are defined as:

$$C^* = \sqrt{a^{*2} + b^{*2}} \tag{1}$$

$$h = \arctan\left(\frac{b^*}{a^*}\right) \tag{2}$$

2.2 Non-linear Lightness Mapping

The lightness of each color is scaled by a compression factor, α, to reduce the lightness of out-of-gamut colors:

$$L^*_m = L^* \times \alpha \tag{3}$$

If the scaled lightness exceeds the cusp lightness, it is clamped to the cusp lightness value:

$$L^*_m = \min(L^* \times \alpha, L^*_{cusp}) \tag{4}$$

2.3 Knee Scaling of Chroma

Chroma is similarly scaled by the compression factor:

$$C_m^* = C^* \times \alpha \tag{5}$$

The compressed LAB values are then calculated using the original hue angle h:

$$a_m^* = C_m^* \times \cos(h) \tag{6}$$

$$b_m^* = C_m^* \times \sin(h) \tag{7}$$

Thus, the compressed color (L_m^*, a_m^*, b_m^*) is obtained.

3 Implementation

The Python code implements the SGCK algorithm to compress the gamut of an input image. The code follows these steps:

1. Generate LAB points for the target AdobeRGB gamut using the ICC profile.
2. Generate a Gamut Boundary Descriptor (GBD) using the Convex Hull method.
3. Estimate the cusp lightness, L_{cusp}^*, from the GBD.
4. For each pixel in the input image:
   - Apply non-linear lightness mapping to compress the lightness.
   - Apply knee scaling to compress the chroma.
   - Recalculate the a^* and b^* components based on the compressed chroma and original hue angle.
5. Convert the compressed LAB image back to the RGB color space.
6. Plot and compare the original and compressed gamuts in the L/C plane.
7. Display the original and compressed images side by side.

4 Observations

The SGCK algorithm effectively compresses the color gamut of the input image, fitting it within the AdobeRGB gamut. As seen in the compressed image, the colors have been adjusted to avoid clipping, with a noticeable reduction in lightness and chroma. The compressed gamut in the L/C plane shows a significant reduction in chroma for higher lightness values, which aligns with the intended behavior of the SGCK algorithm.

While the compression preserves the overall hue and appearance of the image, there is a slight loss of vividness due to the reduction in chroma. This is a trade-off inherent in gamut compression, where the goal is to fit all colors within the target gamut while minimizing perceptual differences.

5 References

1. Xu, L., Zhao, B., & Luo, M. R. (2018). Colour gamut mapping between small and large colour gamuts: Part I. gamut compression. Optics express, 26(9), 11481-11495.

2. Morovic, J. (2003). Guidelines for the evaluation of gamut mapping algorithms. PUBLICATIONS-COMMISSION INTERNATIONALE DE L ECLAIRAGE CIE, 153, D8-6.