

Stable Farthest-Color Lookups via Smoother Perceptual Color Gamuts

by Lana Yang-Maccini

Advisor: James Tompkin

Reader: Daniel Ritchie

A Thesis submitted in partial fulfillment of the requirements for Honors in the Department of Computer Science at
Brown University



Department of Computer Science
Brown University
Providence, RI
April 2026

Abstract

Farthest-color lookups, which map an input color to its most perceptually distant color in a target color space, are an important tool in visualization and graphics for augmented reality overlays, palette design, and more. The natural formulation of such a lookup is discontinuous: the argmax over a perceptually uniform color gamut is attained at a set of extrema, so small input perturbations flip the output between distant colors. In time-varying inputs, like augmented reality, this causes flicker, and in other settings it creates outputs that are unstable under small edits. We identify three compounding sources for this instability: the intermediate color space used for distance computations, the boundary of the gamut of that color space, and the distance measure itself. We investigate each of these sources and propose a pipeline that addresses them by using CIELAB as the intermediate color space, a neural-bounding-based inner smoothing of the intermediate color gamut, and regularized geodesic distances on the bounded surface. The result is a lookup table of size $256 \times 256 \times 256$ that maps each 8-bit sRGB input color to a perceptually distant color and reduces discontinuities while maintaining high saturation and visual contrast. As a demonstrative task, we color active augmented reality area labels in dynamic environments and reduce the maximum label-color gradient by approximately $5\times$ over a prior pipeline.

Acknowledgments

I would first like to thank my thesis advisor, James Tompkin. James' guidance made this thesis possible, and I am so grateful for the support he provided me both while writing this thesis and throughout my research career at Brown. Doing visual computing research has been endlessly interesting and exciting. Thank you for your patience as I learned and grew throughout this process!

I would also like to thank my reader, Daniel Ritchie, for his support in writing this thesis and for teaching me so much about computer graphics and geometry, much of which inspired this thesis. Daniel's availability to students and his teaching over the past few years have allowed me to explore graphics at a much deeper level.

Next, I want to thank Andries van Dam, who was the first person to teach me computer science and whose work in graphics has inspired me for years. Andy's class was what first made me feel welcome in computer science, and since then that feeling has only grown.

Finally, I want to thank my family and friends for listening to me talk about all aspects of this work many, many times over the past two semesters, and for their support and encouragement throughout. I could not have done this without any one of these people, and for them I am immensely grateful.

Contents

Abstract	i
Acknowledgments	ii
1 Introduction	1
1.1 Background	1
1.1.1 Human Color Vision and Color Spaces	2
1.1.2 Color Terminology	4
1.1.3 Color Distance Measures	4
1.2 Augmented Reality Overlays	5
1.3 Related Works	6
2 Problem Overview	9
3 Methodology	10
3.1 Overview	10
3.2 Color Spaces	10
3.2.1 sRGB	11
3.2.2 CIELAB	11
3.2.3 Oklab	12
3.3 Geometry Smoothing	12
3.3.1 Gaussian Filter	13
3.3.2 Neural Bounding	14
3.4 Distance Measures	15
3.4.1 Euclidean Distances	15
3.4.2 Geodesic Distances	15
4 Results and Discussion	17
4.1 Lookup Table: Qualitative Results	17
4.1.1 Color Spaces	17
4.1.2 Smoothing and Bounding	18
4.1.3 Distance Measures	20
4.2 Lookup Table: Quantitative Results	21
4.3 AR Scene Results	26

5	Conclusion	30
5.1	Summary of Contributions	30
5.2	Future Work	30
	References	32
A	Supplementary Code and Equations	34

1 Introduction

1.1 Background

In computer visualization and graphics, mapping an input color to the most perceptually distant color from it, or its farthest color, is often an important task. For example, augmented reality overlays must be colored so that they are perceptually distinct from the underlying environment, and designing a palette of colors for a map or chart requires a set of colors that are perceptually distinct from each other and their background. For ease of use and visualization purposes, we precompute the farthest color mapping for each 8-bit sRGB color and place each farthest color in a $256 \times 256 \times 256$ lookup table indexed by the R, G, and B values of the input color. However, the natural formulation of such a mapping is discontinuous, and the resulting lookup table has sharp boundaries and quick shifts between colors. To illustrate why this might happen, we consider a 2D example in Figure 1.1. Suppose that the output space of possible farthest colors was the 2D space in blue in the figure, with sharp corners at points C and D . If we compute the farthest point in this space from an input color A with Euclidean distances, then we will find that the green C is the farthest point. However, if we evaluate the distances from input color B , which is very close to point A and therefore represents a very similar color, we find that it is far from both points C and D , but farthest from D . The output farthest color will then jump from C to D as we move from A to B , and will never be any of the colors between C and D because C and D , as sharp corners in the space, are further out than any neighbors.

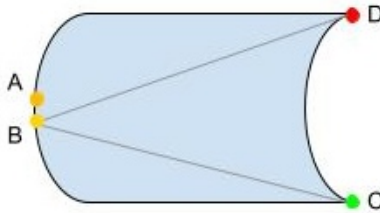


Figure 1.1: 2D demonstration of the cause of sharp boundaries on the lookup table.

In practice, these discontinuities flip the output between distinct colors with small input perturbations, like the shift from A to B . With time-varying inputs, like augmented reality, this causes a noticeable flicker. In other settings, this makes outputs unstable under small edits.

Smoothing a discontinuous lookup table itself removes harsh boundaries, but averaging perceptually distinct colors tends to produce dull or muddy colors, reducing the contrast that the lookup table intends to map. Therefore, we desire a new formulation of the lookup table that removes the large discontinuities to smooth the output and provide for a more stable color mapping, but also

that maintains high visual contrast between input and output colors.

1.1.1 Human Color Vision and Color Spaces

A color space is an organization system of colors for use in color representations and reproductions, often with numerical coordinates for each color. The sRGB color space, for example, is one of the most well-known color spaces, used widely on computer monitors and printers. This color space assigns each color a red, green, and blue value (R, G, and B). However, the sRGB color space relies on a linear model of intensity based on the physical qualities of light that does not actually match how human color vision responds to color differences. The human visual color system is nonlinear, meaning that our perception of color does not respond linearly to physical light intensity. Consequently, two pairs of colors with similar distances in the sRGB color space can have different perceptual distances to the human eye. Consider some function $f(x, y)$ that takes in two colors in the sRGB space and outputs a numerical representation of the difference in human perception between them, and also a function $\Delta(x, y)$ that takes in two colors in the sRGB space and outputs the physical distance between them in the space. The nonlinearity of human color vision means that for two pairs of colors (x_1, y_1) and (x_2, y_2) in sRGB space, it is often true that $\Delta(x_1, y_1) = \Delta(x_2, y_2)$ but $f(x_1, y_1) < f(x_2, y_2)$. This is undesirable when computing in color spaces, because the distance measure does not represent a consistent quality about color difference to the human eye.

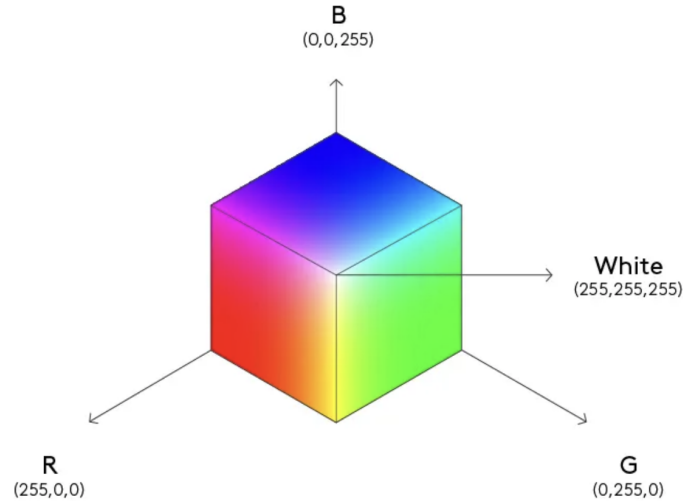


Figure 1.2: The sRGB color model with red, green, and blue intensity axes. [1]

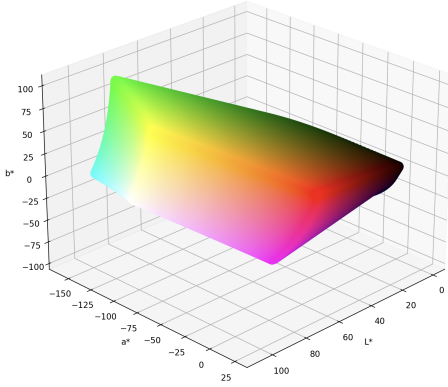
In an effort to resolve these issues with the sRGB color space, perceptually uniform color spaces were developed, with the aim of creating a color space where distances between colors are closer

to the perceptual distances between those colors. CIELAB, a popular color space developed in 1976, quantifies colors with a lightness value (L) and two values representing distance along the red-green and blue-yellow axes, respectively (a and b). It was constructed by placing colors so that the Euclidean distances between them in the space roughly corresponded to human perceptual differences. Oklab, developed in 2020, is another perceptually uniform color space based on the values L , a , and b . It was intended to fix some of the issues with the CIELAB color space, especially for better perceptual consistency across gradients and interpolation. In contrast, in the CIELAB space, changing the lightness or saturation of a color can cause the hue of the color to shift, which is undesirable for many applications, such as creating a color gradient that feels sensible to the human eye (Figure 1.3).

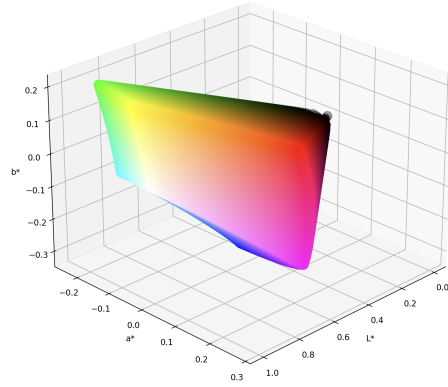


Figure 1.3: A comparison of gradients formed by blending white and blue in the Oklab and CIELAB color spaces [2]. Note that the CIELAB color gradient changes the hue of the blue to purple as it blends, but the Oklab color gradient does not.

To find the farthest color from an input color in some color space, we will need to examine the space of all colors in that color space that could be used as a farthest color. This is called the color gamut of a color space, and it represents all the reproducible colors on a specific device, which is a computer screen for our purposes. For example, the gamut of the sRGB space is the cube made up of each combination of integer values of R, G, and B from 0-255 (8-bit integers). See Appendix A, Listing A.1 and Listing A.2, for more details on finding the CIELAB or Oklab coordinates of an sRGB color.



(a) CIELAB color gamut in 3D space, found by plotting the CIELAB value of each 8-bit sRGB color.



(b) Oklab color gamut in 3D space, found by plotting the Oklab value of each 8-bit sRGB color.

Figure 1.4: Various color space gamuts, plotted in 3D.

1.1.2 Color Terminology

We also define some relevant color terms. The hue of a color represents the type of color it is, such as red, blue, or yellow. The saturation of a color represents how vivid or strong the color is, where colors with low saturation look dull or gray. The value of a color represents how light or dark the color is. These three values define another color space, HSV (hue saturation value), which provides a more intuitive representation of color than LAB-based color spaces. Beyond this, we can use these values to quantify intuitive qualities about color.

1.1.3 Color Distance Measures

Even with perceptually uniform color spaces, quantifying the perceptual distance between two colors is notoriously difficult. Perceptual color space cannot be perfectly modeled as a Euclidean space [3] because human color judgments are not consistent with a Euclidean geometry. Perceptual color space can be better modeled by a 3D Riemannian space, but this approach is not perfect. Riemannian spaces cannot account for the diminishing returns in human color vision [4], which is the fact that the sum of many small differences between colors is perceived as greater than the total, larger difference between colors. Nonetheless, 3D Riemannian spaces are generally considered a better approximation for perceptual color space than 3D Euclidean spaces [5].

In a Riemannian space, the shortest path between two points is represented by a curved path, the length of which is called a geodesic distance. In comparison, in a Euclidean space, the shortest path between two points is represented by a straight line, the length of which is the traditional Euclidean distance.

Due to the CIELAB space's popularity and longevity, there are other distance measures that

have been designed for it to better approximate perceptual distance. The ΔE_{76} distance measure represents a simple Euclidean distance, where the distance between two colors (L_1, a_1, b_1) and (L_2, a_2, b_2) is $\Delta E_{76} = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2}$. The ΔE_{00} distance measure, designed in 2000, adds a number of corrections and compensations to this to make a much more complicated, but more perceptually accurate, distance formula (see Appendix A, Listing A.3 for more details). Notably, ΔE_{00} distance is not continuous.

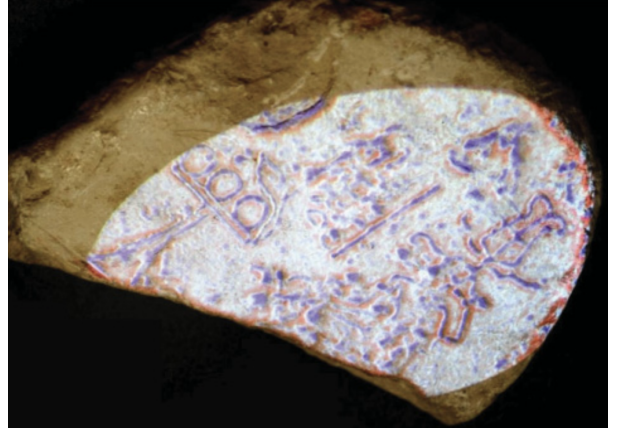
It is important to note that no color space or distance measure found so far perfectly represents human color perception, especially for large color differences [6], [4]. As such, any rigorous mathematical definition of a space or distance measure must be, at best, an approximation of properties of human color perception. However, we can use mathematical notions that will get as close as possible to true color perception through perceptually uniform color spaces like CIELAB. In this work, we employ a number of distance measure formulas that estimate perceptual differences to see which can best induce smoothness on farthest-color mappings. None of these distance formulas can reliably find the absolute farthest color to the human eye, but we use them as natural approximations to find a perceptually distant color from an input while removing discontinuities.

1.2 Augmented Reality Overlays

We use active AR area labels as a motivating example for stable lookup tables, and apply our new lookup table formulation to this task to test the efficacy of our solution in a real application. Augmented reality (AR) is a form of human-computer interaction that overlays digital cues onto our 3D physical world to provide information or add interactivity in real-time. In contrast, in the similarly popular virtual reality (VR), computers simulate the user’s entire environment. An essential part of AR systems are AR area labels. Area labels are AR shapes overlaid onto a real-life environment that can be used to highlight parts of the scene, to visualize invisible features, or to serve as backgrounds for text. For example, AR overlays and area labels can be used to take medical images to reduce radiation exposure [7] (Figure 1.5a) or to reveal details in cultural artifacts that are otherwise too eroded to see [8] (Figure 1.5b).



(a) Operator view of an AR-overlay-based system used for magnetic resonance guided injection of the hip.



(b) AR techniques used in an overlay to reveal details previously lost to aging effects in a cultural artifact.

Figure 1.5: Various uses of AR overlays.

Dynamic real-life environments can change quickly. To handle this, we use active AR area labels, which update their appearance to reflect changing environment conditions, like varying lighting or complexity of environment. Active area labels then allow us to maintain high visual contrast between the scene around the label and the label itself by shifting in real-time.

To find an appropriate area label color, we must find a color that is perceptually distinct from the scene underlying and surrounding the label position, and do so using a precomputed farthest-color lookup table to achieve real-time performance. However, an unstable lookup table causes a significant flicker. Flicker is visually exhausting [9] and can be distracting, so we desire an AR system that can color area labels that are both visually distinct in each frame and smooth frame-by-frame to avoid flicker. Therefore, color generation for active AR area labels is an ideal context in which to test the stability of a lookup table and see how well the lookup table maintains high visual contrast between input and output colors while reducing flicker.

1.3 Related Works

Kwon et al. [10] propose a method to generate lookup tables via a pipeline for AR area label colors. They begin by sorting all pixel values under the area label in a scene into 27 equally sized bins in sRGB space and then use the average pixel color of the bin with the most pixels as a characteristic environment color for the label. They then use the ΔE_{00} distance measure to find the farthest color in CIELAB space from the characteristic environment color, and this becomes the area label color. Once computed for all 256^3 sRGB colors, they put these into a lookup table (Figure 1.6) for use for querying in real-time. Although this method achieves visually distinct area labels, they note that it causes flickering in the label color due to discontinuities in the lookup table.

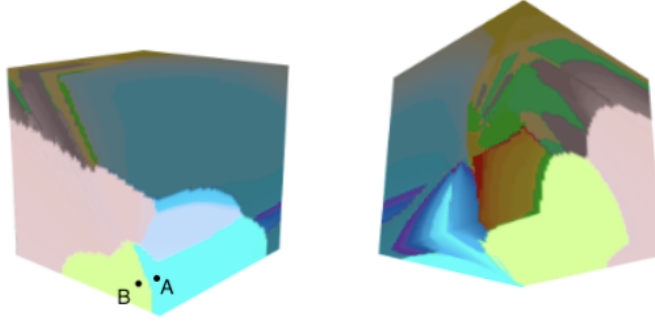


Figure 1.6: Two views of a lookup table generated using the method of Kwon et al. for real-time color lookup of perceptually distant AR area label colors, with possible characteristic environment (input) color locations for the environment in Figure 1.7 labeled.

Figure 1.7 displays neighboring frames in a video of a skyline and captures a moment of flicker. Even though the underlying scene on both is perceptually similar, the overlay color, which comes from the lookup table, is perceptually different. The two points labeled *A* and *B* on Figure 1.6 could represent the two input color locations on the lookup table for these two frames, and illustrates how moving the input from point *A* to point *B*, which is a small change in input color, causes a jump in the output color from light blue to green.

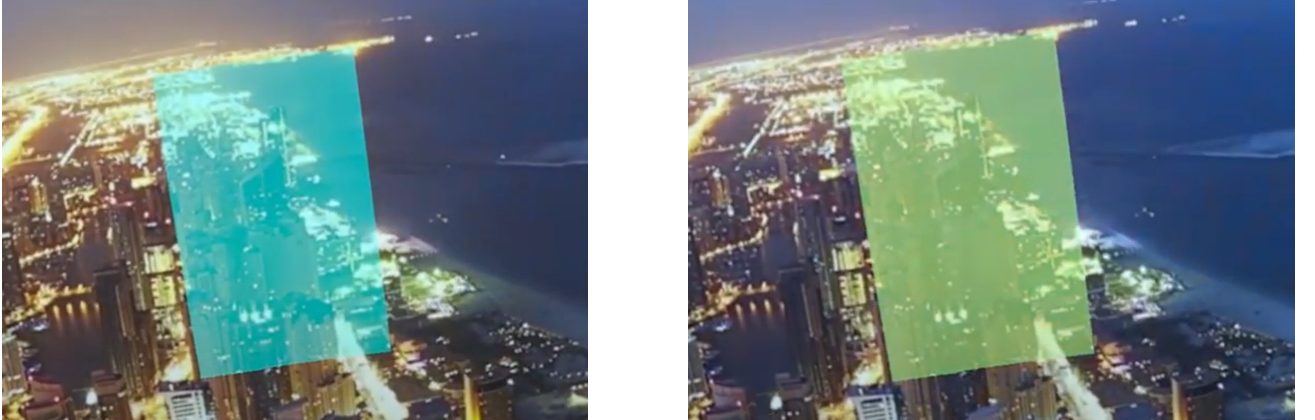


Figure 1.7: Flicker captured in nearby frames in a video of a skyline as the sun rises. The slight lightening of the sky from the left frame to the right frame triggers a drastic change in the label color.

Much work has been done on various types of smoothing, from distances to geometries, that we use in our method. First, Liu et al. [11] introduce a method for generating smooth bounding boxes around or inside 3D objects called neural bounding, and we adopt this method for geometry smoothing to remove or reduce extrema. Further, Edelstein et al. [12] propose a regularizer for geodesic distances to induce smoothness that we use in computing farthest colors.

In color spaces, Wuerger et al. [3] show that perceptual color space cannot be modeled as a

Euclidean space, while Bujack et al. [4] also show that perceptual color space does not perfectly fit a Riemannian space because of diminishing returns. Pant et al. (2012) [5] argue that 3D Riemannian spaces are a better approximation for perceptual color spaces than 3D Euclidean spaces. Pant et al. (2013) [6] note that no current color space or distance measure perfectly fits human color perception and the complexities of color vision.

In augmented reality, Marino et al. [13] show that AR area labels colored with disharmonic strategies (maximizing visual contrast) have better visibility against their environment. Lindquist et al. [9] show that high visual flicker in colored patterns causes visual discomfort, demonstrating the need for stable farthest-color queries.

2 Problem Overview

Let $G \subset \mathbb{R}^3$ be a bounded perceptually-uniform color gamut, let $\Delta : G \times G \rightarrow \mathbb{R}_{\geq 0}$ be a perceptual distance, and define the farthest-color map

$$h(c) = \arg \max_{c' \in G} \Delta(c, c'). \quad (2.1)$$

To produce a stable mapping, we must bound $\|\nabla h\|$, and do so by targeting a smoothed surrogate $\tilde{h}$. $\tilde{h}$ must maintain two properties: (i) $\Delta(c, \tilde{h}(c))$ is close to $\max_{c' \in G} \Delta(c, c')$ (the returned color is still perceptually far), and (ii) $\|\nabla \tilde{h}\| < \|\nabla h\|$ ($\tilde{h}$ is smoother than h).

Primarily, sharp boundaries in h that increase $\|\nabla h\|$ occur when G has sharp extrema. The $\arg \max$ function concentrates at these extrema for large sets of inputs, so small perturbations can trigger a jump in the output between extrema. To reduce this, we focus on the selection of G and Δ , as well as reducing the sharpness of extrema in G itself.

3 Methodology

3.1 Overview

To explore the effects of various color spaces, smoothing color gamut geometry, and various distance measures on the lookup table formulation, we construct the following pipeline to produce a lookup table, given the gamut of the 8-bit colors in sRGB space G_{RGB} , an intermediate color space with gamut G that distance computations are performed in, a smoothing method S , and a distance measure Δ . Although it is not necessary for color spaces to be 3D, in practice almost all color spaces are 3D because human color vision is trichromatic (perceived through three cones in the eye), so we do not consider color spaces that are not 3D in this work.

1. Let f map a color in sRGB space to a point in the intermediate color space. To find the boundary of G , we map each color in G_{RGB} to the intermediate color space, so $G = \text{Im}(f)$, the image of the function f in the intermediate color space.
2. Wrap a triangle mesh T around the boundary of G . This triangle mesh should be finely meshed and triangles should be as consistent as possible in total area and angle measurements. When calculating geodesic distances, very irregular meshings of the boundary points can cause inaccurate distance calculations across the mesh because the path across the surface traverses inconsistent triangles in unpredictable ways.
3. Smooth T with the smoothing method S to form a new triangle mesh $\tilde{T}$.
4. For every point c in G , find the nearest vertex v_c on $\tilde{T}$. Calculate $\arg \max_{v \in V(T_S)} \Delta(v_c, v)$ to find the farthest vertex v'_c . Find the nearest point c' in G to v'_c to find the farthest color from c by Δ . Repeating this across all points in G produces G' , the array of all farthest colors in the intermediate color space.
5. Map each color in G' back to sRGB space using f^{-1} to get G'_{RGB} , the collection of the farthest colors from each original G_{RGB} point using Δ and S . These points can now be used as a 3D lookup table, organized by the R, G, and B values of each original color.

3.2 Color Spaces

We explore first whether changing the color space that distance calculations are performed in affects the stability of the final lookup table. We call this color space the intermediate color space, as colors are transformed from sRGB space to this new color space and then back to sRGB space throughout the lookup table generation process. In particular, we investigate the uses of the sRGB, CIELAB, and Oklab color spaces as the intermediate color space.

3.2.1 sRGB

To demonstrate the need for an intermediate color space, we explore the sRGB color space itself as the intermediate color space, which simply keeps all computations throughout the process in the original sRGB color space. As mentioned in section 1.1.1, the sRGB color space has issues with perceptual uniformity due to the nonlinearity of human color vision, so it is not an ideal candidate because distance measures are not reflective of true color perception to begin with. However, the geometry of its gamut is also suboptimal. The sRGB gamut in 3D is simply a cube, which has sharp peaks at all of its corners. The farthest point calculation for each color is greatly simplified, as the farthest point is equivalent to whichever corner is farthest away. Using Euclidean distances on the sRGB space then produces a lookup table (Figure 3.1) with very low color diversity (only 8 colors). This then leads to low diversity of label colors and extremely sharp edges, which are highly unstable to input changes across any boundaries.

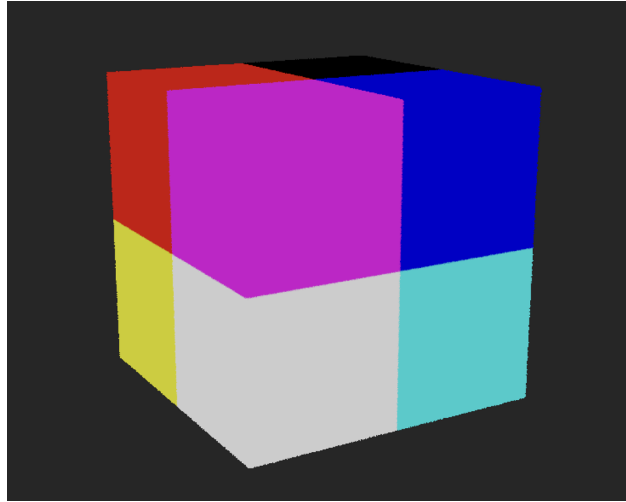


Figure 3.1: A lookup table generated using sRGB as the intermediate color space, with Euclidean distances.

As a result, we look to explore other, perceptually uniform color spaces, ideally with more complex gamut geometry than the cube offered by sRGB color space, to promote greater color diversity and smoother color gradients across the lookup table.

3.2.2 CIELAB

As seen in Figure 1.4a, the gamut of the CIELAB color space has a more irregular shape than the gamut of the sRGB color space. The curves in this shape can help create softer boundaries and a higher number of colors across the lookup table due to more complex distance variations across the shape. Further, CIELAB has some additional predefined distance measures specifically designed for the CIELAB space that can be more perceptually accurate, like ΔE_{00} , and the space itself is

more perceptually uniform than sRGB. These attributes motivate the use of this space in distance calculations, especially over sRGB.

3.2.3 Oklab

The Oklab color space was designed to improve on the CIELAB color space’s unintended effects on hue while maintaining perceptual uniformity. It is particularly good for forming gradients with constant hue or lightness, so a path in Oklab may have more consistent hue or lightness than a path in CIELAB, seeming more sensible to the human eye. We explore Oklab as the final intermediate color space option to see if these improved qualities of the space help at all with generating a lookup table that is smooth and sensible to the human eye.

3.3 Geometry Smoothing

Although changing color space has an impact on the lookup table result that may help with stability, the ultimate cause of discontinuity is sharp extrema, which are present across the gamuts of many color spaces. Thus, we investigate smoothing the geometry of the gamut boundary to remove or reduce these extrema.

We require an optimization to smooth the original triangle mesh T of the input points in the intermediate color space, from item 2 in Section 3.1, into a new, smoothed mesh $\tilde{T}$. This optimization must prioritize the following:

1. Containment: the optimized triangle mesh must be completely contained within the original triangle mesh. Because a color gamut represents the reproducible colors in that color space as found from 8-bit sRGB colors, any point outside the original mesh is not reproducible and cannot be used as a farthest color.
2. Curvature minimization: the overall objective of the smoothing is minimized curvature across the triangle mesh, which will remove significant peaks or corners.
3. Volume maximization: we also want to maximize volume within the optimized triangle mesh, which helps maximize distances between points within the mesh and, therefore, maximize final perceptual distances between input and output colors.

We explore a number of smoothing methods, including multidimensional Gaussian filtering and neural bounding.

We use a simple sphere bounding to demonstrate the issues with over-smoothing, which reduces the volume of the bounded mesh $\tilde{T}$ and the relative differences between colors.

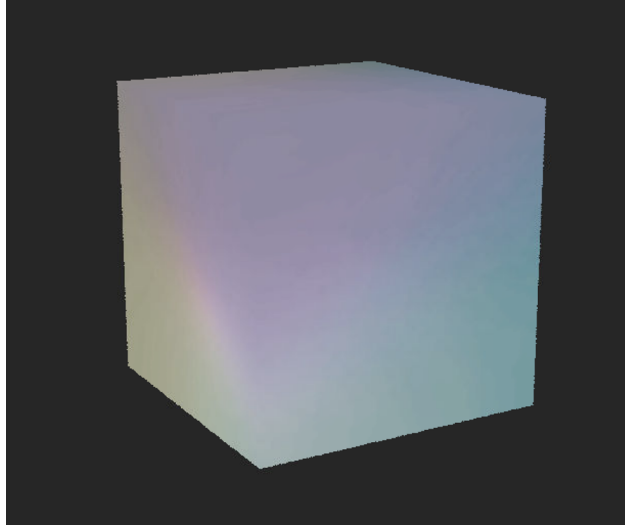


Figure 3.2: A lookup table generated using a sphere bounding method on the CIELAB gamut.

In Figure 3.2, we bound the colors in the CIELAB gamut to a sphere inscribed within the gamut, using the shortest distance of all boundary CIELAB points to the center of the gamut as the radius of the sphere to keep all points within the space of original 8-bit sRGB gamut. However, the CIELAB gamut is irregular, so this radial distance is low, as is the maximum distance between any two colors in the bounded space. This results in a pale lookup table without much color variation. Although it is very smooth, eliminating flicker, it does not maintain high visual contrast with input colors. Instead of using a simple sphere, we explore other methods that do not reduce the volume so severely.

3.3.1 Gaussian Filter

A Gaussian filter is an operation that uses a Gaussian kernel, or a mathematical function that weighs points based on their distance from the center, to blur or smooth data by taking a weighted average of surrounding data for each point. The Gaussian kernel used for filtering is determined by σ , and a larger σ causes more blurring because it creates a wider filter. Although Gaussian filtering is most common in image processing, used to blur images in 2D, it can also be applied to 1D or 3D data. Because we want to pull sharp extrema in towards the center, we can apply a 3D Gaussian filter on the 3D color gamut data. This will move any points in an extrema in towards their less extreme neighbors, smoothing out sharp peaks. The CIELAB and Oklab gamuts also tend not to have sharp valleys towards the center in 3D representations; instead, specific extrema point outwards. So using a Gaussian filter does not violate the containment requirement for smoothing as long as σ remains low enough to prevent it from smoothing larger areas of concavity outward.

The Gaussian filter operates on discrete data, so we apply the filter to the discrete points in the gamut of the intermediate color space, found by transforming the 8-bit sRGB points into the

selected color space and then forming a voxel representation of these points, where each point corresponds to a voxel with a value of 1 and empty space corresponds to voxels with a value of 0. We then create a boundary triangle mesh around these points (Figure 3.3), varying the value of σ to achieve different levels of smoothing.



Figure 3.3: Triangle mesh of the CIELAB gamut after Gaussian filtering with various values of σ .

To maintain a maximal volume, it is also beneficial not to make σ too large, as smoothing brings the mesh inwards. We hope to use a value of σ high enough to smooth the final lookup table and no higher to maximize visual contrast between input and output colors and ensure containment.

3.3.2 Neural Bounding

Neural bounding is a method of finding bounding boxes around shapes using neural networks introduced by Liu et al. [11]. In this framework, a neural network creates a bounding volume by learning to classify space as free or occupied, using a weight α as the cost for false negatives (parts of the original object outside the bounding volume) and another weight β as the cost for false positives (parts outside the original object are included in the bounding volume). To create a traditional bounding volume around an object, α must be high and β can be lower, but to create a bounding volume that should be entirely inside the object, but also contains as much of the object as possible, Liu et al. show that we can flip these weights to make β high and α comparatively lower. This kind of bounding volume, with flipped weights, produces a smoothed geometry limited to be within the original gamut, as we desire (Figure 3.4).



Figure 3.4: Neural bounded version of the CIELAB gamut, found by using a high β weight and then using a triangle mesh wrapped around the resulting volume.

3.4 Distance Measures

Finally, we investigate the effect of various distance measures as an approximation of color difference. We desire to find the farthest color from each input color to serve as a perceptually distant label color, so quantifying the distance between that color and all other colors is a necessary step. However, discontinuities and harsh boundaries can happen on the resulting lookup table due to jumps in distance measure results, so using different distance measures may help avoid such jumps. Further, finding a distance measure that is more accurate to human color perception helps with visual contrast of label colors.

3.4.1 Euclidean Distances

We begin by using Euclidean distances as a simple measure of the most perceptually different color. We see that this often creates lookup tables with low color diversity and harsh boundaries (Figure 3.1), so we hope that another distance measure can improve these issues.

3.4.2 Geodesic Distances

We then consider geodesic distances as an approximation of perceptual distance. Because geodesic distances are only defined on the surface, to find the distance using this measure for points within the gamut, we project points inside the gamut to the nearest point on the surface of the gamut, and then find the farthest geodesic distance between that projected point and all other points in the gamut. Although the geodesic distance cannot be defined for points within the gamut for comparison with the farthest geodesic distance, the farthest point from an input point in some gamut must be on the boundary of the gamut, so we only need to consider points on the boundary.

As noted in section 1.1.3, perceptual color space is not a perfect match for either Riemannian or Euclidean space, but is often approximated as a Riemannian space. In a Riemannian space, distance is found via the geodesic (path along the surface) between two points, instead of via the

straight line between them. No current formulation of a color difference measure perfectly reflects perceptual distance, especially over larger distances, so we use geodesic distances as a natural approximation.

Further, because geodesic distances represent distances across a surface, they may help create a wider variety of colors on the lookup table. The varying paths across the surface may better reflect the complexities of the surface, especially when compared to Euclidean distances, which would in most cases not take into account the particularities of the surface. To ensure smoothness of the distance measure to avoid boundaries in the lookup table, we use the regularized geodesic distances introduced by Edelstein et al. [12]. Their method calculates geodesic distances across the surface of a mesh using a regularizer to promote smoothness. This regularizer is controlled by a smoothing weight $\hat{\alpha}$. Using this distance measure then allows us to directly increase the smoothing on the distance calculation by increasing $\hat{\alpha}$, further smoothing out harsh boundaries on the lookup table.

4 Results and Discussion

We will next explore the results of the various intermediate color spaces, geometry smoothing techniques, and distance measures to show an ideal method for generating a lookup table for area label color assignment. Increasing smoothness is the primary concern. To achieve this, we desire a lower maximum and average gradient across the new lookup table. A higher diversity of colors across the lookup table is also helpful in ensuring a more stable lookup table so that there are sufficient colors to create smoother transitions. We also want to maintain high visual contrast between input and output colors, so we hope to increase saturation and intensity of colors in the lookup table because more saturated, bright colors are better to attract attention [14]. Our aim is to find the best method to generate a lookup table that is smoother than the original (with fewer sharp boundaries and instead slower transitions between perceptually disparate colors) and that maintains or increases visual contrast of labels. We can measure the success of this by examining the diversity of color, color gradients, and color richness across the lookup table, and hope for higher diversity of color and color richness but lower maximum and average color gradients, as a high gradient in the lookup table represents a potential input color change that would cause a jump in the output.

4.1 Lookup Table: Qualitative Results

4.1.1 Color Spaces

As expected, using sRGB color space as the intermediate color space does not produce lookup tables with the desired qualities of color diversity (for visual contrast and smoothness) and few harsh boundaries, so we examine more closely the differences between lookup tables produced using the CIELAB and Oklab spaces.

In general, using the CIELAB color space as the intermediate color space produced the best visual results. The Oklab space produces lookup tables with a few harsh boundaries that tend not to soften much (Figure 4.1a). This difference is present across various types of smoothing and parameter values in smoothing and distance calculations, so we find that CIELAB is a better space to do intermediate distance computations in.

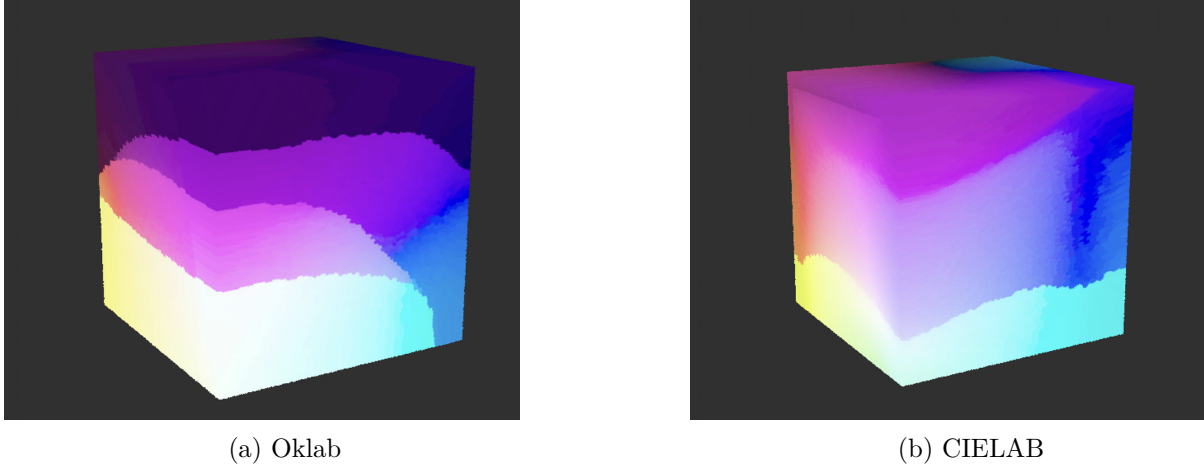


Figure 4.1: Lookup tables generated using CIELAB and Oklab as intermediate color spaces, both using no geometry smoothing and regularized geodesic distances with $\hat{\alpha} = 0.05$.

4.1.2 Smoothing and Bounding

When compared to lookup tables generated with no geometry smoothing, both Gaussian filtering and neural bounding produce visually smoother lookup tables with softened boundaries. Increasing the σ parameter in Gaussian filtering also improves the smoothing result, as shown in Figure 4.2.

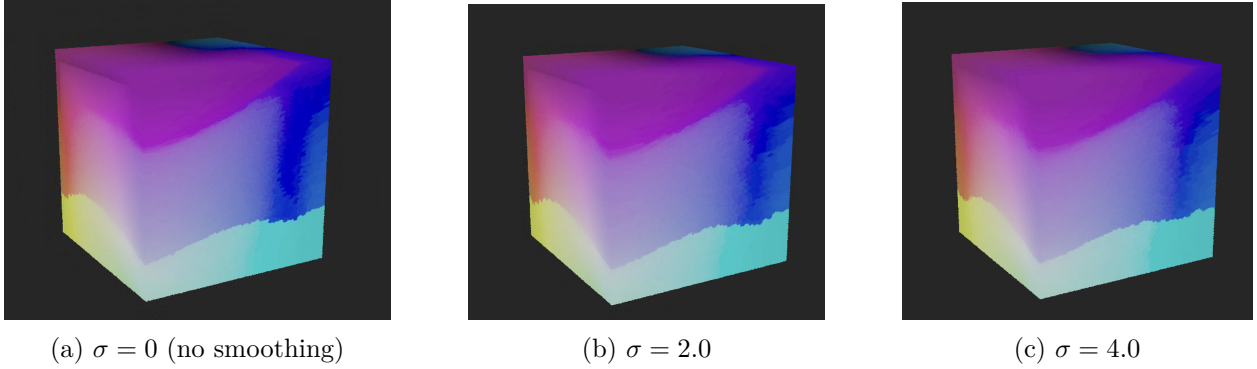


Figure 4.2: Lookup tables generated with Gaussian filter smoothing of the CIELAB gamut with various values of σ , all using regularized geodesic distances with $\hat{\alpha} = 0.05$.

The effect of increasing σ can be seen more clearly when more zoomed in (Figure 4.3).

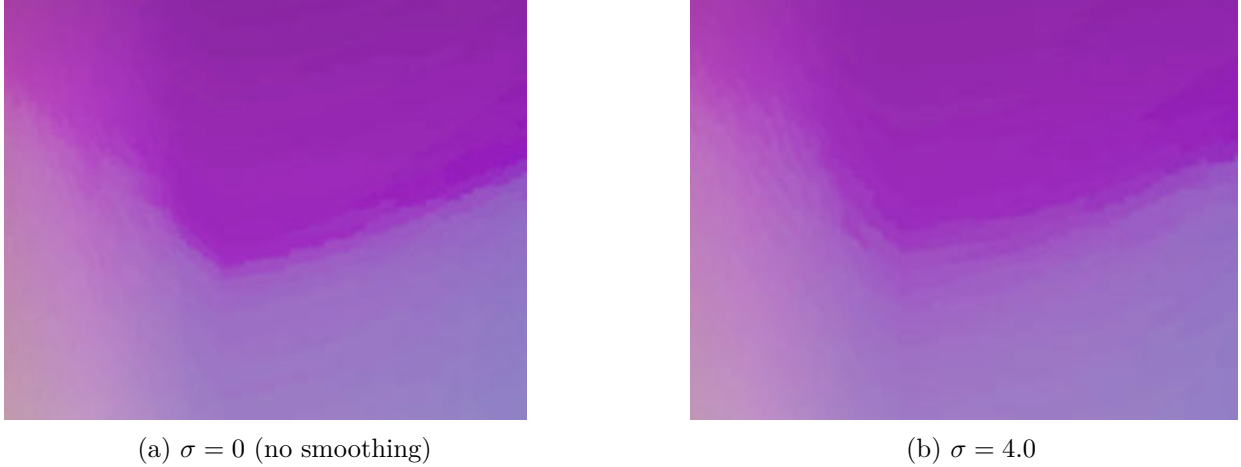


Figure 4.3: Zoom-in on a specific boundary in the lookup table generated with Gaussian filter smoothing across various values of σ , both using regularized geodesic distances with $\hat{\alpha} = 0.05$.

We then compare these results to the lookup table produced by the neural bounding technique with swapped weights to form a bounding volume inside the original CIELAB space (Figure 4.4).

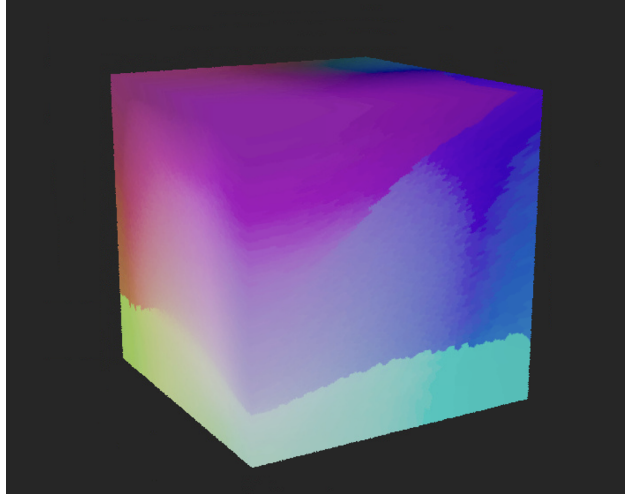


Figure 4.4: Lookup table generated with neural-bounding-based smoothing of the CIELAB gamut, using regularized geodesic distances with $\hat{\alpha} = 0.05$.

Although the results are similar for Gaussian filtering with $\sigma = 4.0$ and the neural-bounding-based method, we note that the neural-bounding-based method promotes slightly more visual smoothness (as shown in Figure 4.5 on a zoomed-in boundary), so we use geometry smoothed with neural bounding in our final method.

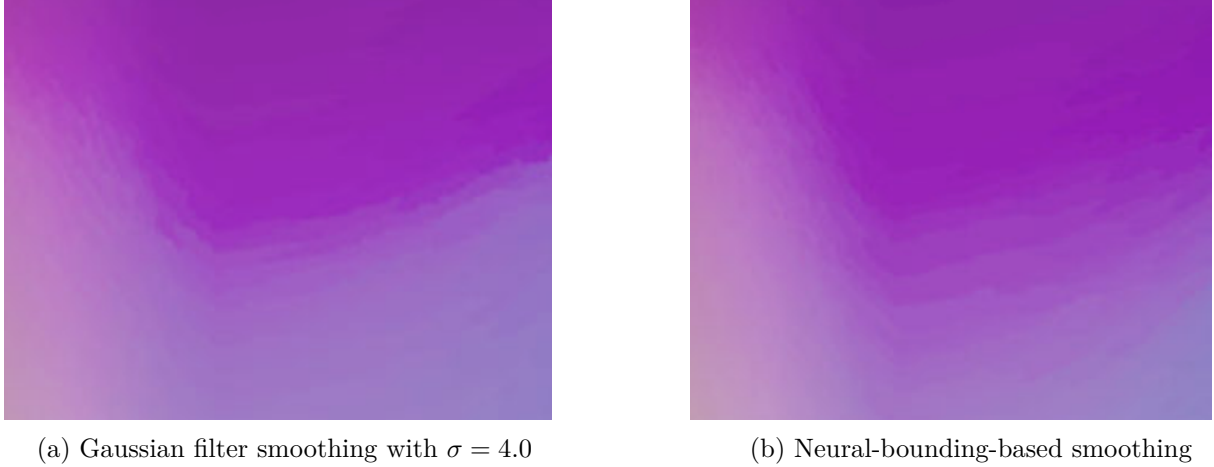


Figure 4.5: Zoom-in comparison on a specific boundary in the lookup table generated with Gaussian filter smoothing and neural-bounding-based smoothing, both using regularized geodesic distances with $\hat{\alpha} = 0.05$.

4.1.3 Distance Measures

Finally, we explore the impact of using regularized geodesic distances instead of Euclidean or ΔE_{00} distances. Regularized geodesic distances, even at a low $\hat{\alpha}$ value, show a significant improvement over Euclidean or ΔE_{00} distances (Figure 4.6), both in terms of color variety and richness, and perceptual smoothness of boundaries.

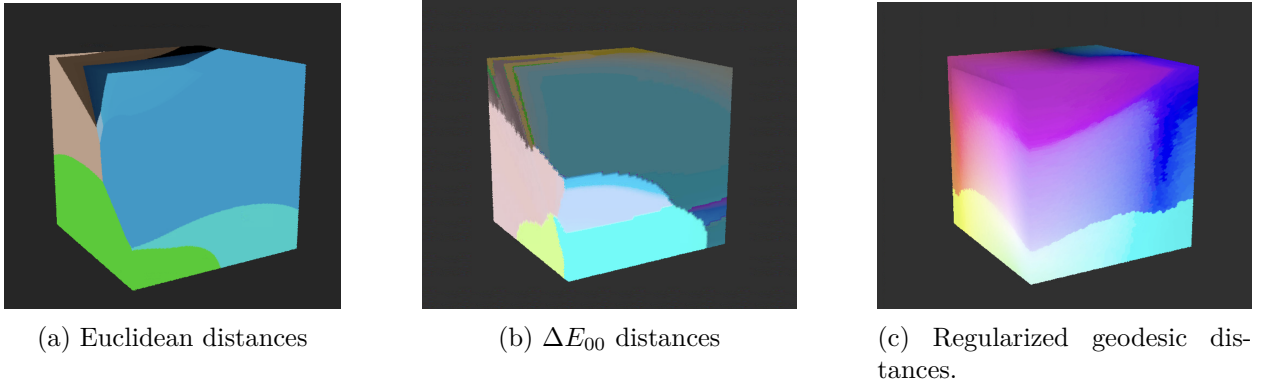


Figure 4.6: Lookup tables generated using different distance measures in the CIELAB space, with no geometry smoothing.

In addition, we can increase the value of $\hat{\alpha}$ to increase the amount of visual smoothing across the lookup table (Figure 4.7), but these improvements decrease as $\hat{\alpha}$ increases, with the largest improvements roughly between $\hat{\alpha} = 0.05$ and $\hat{\alpha} = 0.5$.

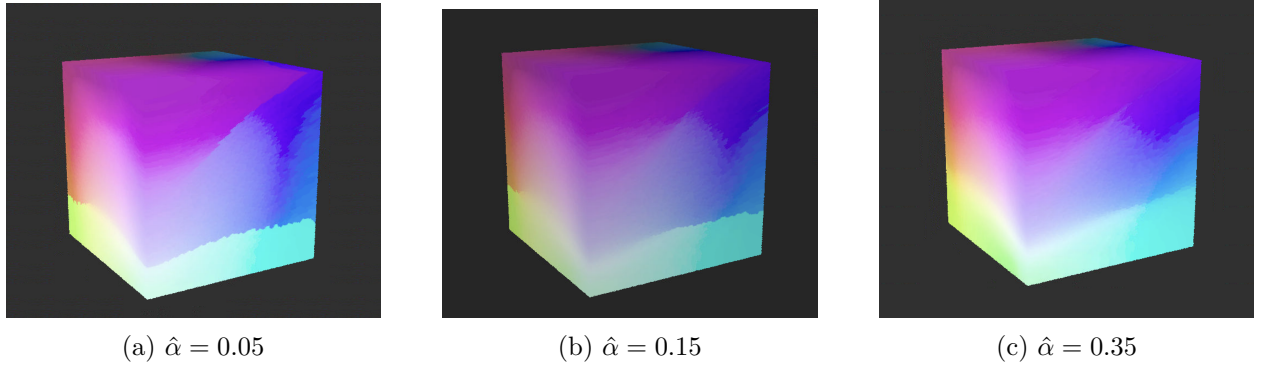


Figure 4.7: Lookup tables generated using different values of $\hat{\alpha}$ for regularized geodesic distances in the CIELAB space, with neural-bounding-based geometry smoothing.

We can see these effects more clearly on a zoomed-in portion of the boundaries of the lookup tables generated with different values of $\hat{\alpha}$.

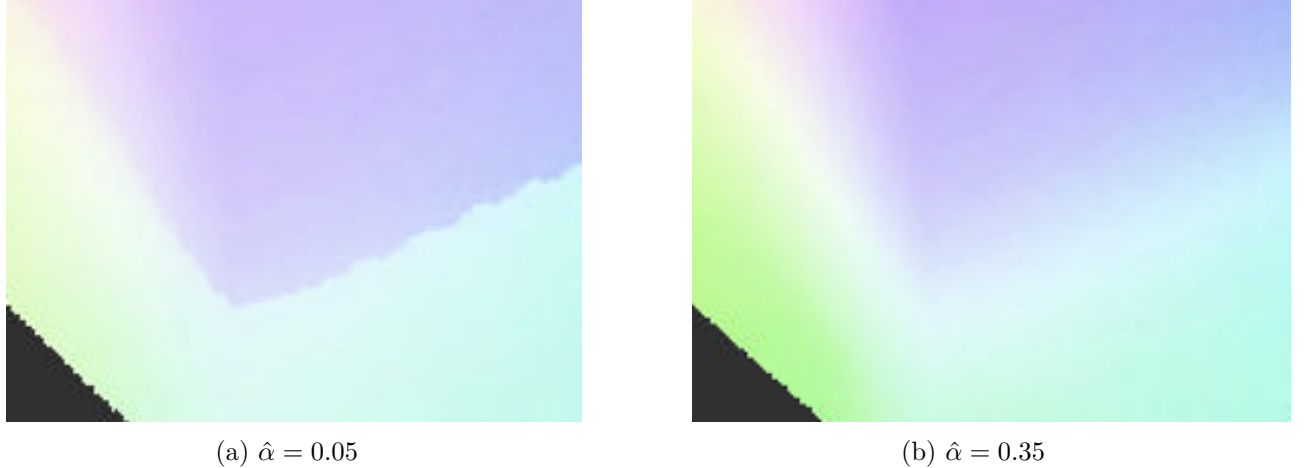


Figure 4.8: Zoom-in comparison on a specific boundary in the lookup tables generated with regularized geodesic distances with different values of $\hat{\alpha}$.

4.2 Lookup Table: Quantitative Results

We evaluate the efficacy of the different techniques also with quantitative methods, including gradients, intensity and saturation values, and color distributions. We compare new lookup tables generated with various techniques to each other and also to the original result from Kwon et al., marked from here forward as "Original", which uses ΔE_{00} distance in the CIELAB space and no geometry smoothing.

In particular, we calculate the gradient maximum and average gradient across a lookup table to approximate perceptual smoothness. We want a lower gradient across the cube to promote smoothness between colors, but note that increasing the diversity and richness of colors can increase gradients. We then examine the average intensity (derived from value) and saturation across each

lookup table to provide metrics on the richness of the colors in the lookup table, and we want higher intensity and saturation values to promote visual contrast and attract attention.

We first compare these metrics across techniques, specifically comparing across intermediate color spaces, geometry smoothing methods, and distance measures.

Method	Gradient Maximum	Gradient Average	Saturation Average	Intensity Average
Oklab	279.242	5.088	0.833	0.600
CIELAB	238.825	5.549	0.845	0.756

Table 4.1: Comparison of quantitative results on various intermediate color spaces, with Gaussian filtering and regularized geodesic distances.

Method	Gradient Maximum	Gradient Average	Saturation Average	Intensity Average
None	278.660	5.705	0.852	0.766
$\sigma = 2.0$	251.072	5.657	0.847	0.760
$\sigma = 4.0$	238.825	5.549	0.845	0.756
NB	219.049	5.404	0.835	0.756

Table 4.2: Comparison of quantitative results on various geometry smoothing methods, using regularized geodesic distances and the CIELAB color gamut. Methods marked with a σ value represent results obtained using Gaussian-filter-based smoothing with a smoothing parameter of σ , and the method marked with NB represents results obtained using neural-bounding-based smoothing.

Method	Gradient Maximum	Gradient Average	Saturation Average	Intensity Average
ΔE_{00}	163.267	7.624	0.657	0.500
Euclidean	312.310	3.440	0.923	0.904
RGD	212.176	4.978	0.861	0.786

Table 4.3: Comparison of quantitative results on various distance measures, with no geometry smoothing and using the CIELAB space. RGD = regularized geodesic distances, ΔE_{00} = original method (Kwon et al.).

Any type of geometry smoothing will reduce the space inwards and lower saturation and intensity when compared to results produced with no smoothing, so it is expected that no smoothing will have the highest saturation and intensity values. We then accept some reduction in intensity and saturation to increase smoothness, but try to keep intensity and saturation as high as possible.

We see here that using regularized geodesic distances minimizes the impact of smoothing on intensity and saturation to keep the color richness much higher than using ΔE_{00} distances. Because there are more saturated colors and more colors total in the new lookup tables, the gradients have

the potential to be larger. However, average gradients still decrease with regularized geodesic distances in comparison to ΔE_{00} , so there are fewer very high gradients.

With these results, we can support the observational results that CIELAB is the preferred intermediate color space and that neural-bounding-based geometry smoothing is most effective for reducing gradients. Although it does not necessarily score the highest in metrics, we also select using regularized geodesic distances as the preferred distance method because it provides a more balanced approach versus the other distance measures, each of which excels in some categories but has significant issues in other categories. In particular, ΔE_{00} distances achieve a low gradient maximum, but this is because the colors in its lookup table are much less saturated or bright. They also have the highest average gradient, indicating that there are many boundaries. Euclidean distances produce a saturated lookup table that has a low gradient average, but this is because large swaths of the lookup table are the same color (leading to a gradient of 0 across these areas). Boundaries between these areas have very high gradients, as shown by the problematically high gradient maximum, and this is what causes instability. In contrast, regularized geodesic distances do not have the worst score in any category, unlike the other two options.

We can then examine the effect of changing the $\hat{\alpha}$ value in regularized geodesic distances, summarized across the four metrics in the plots in Figure 4.9.

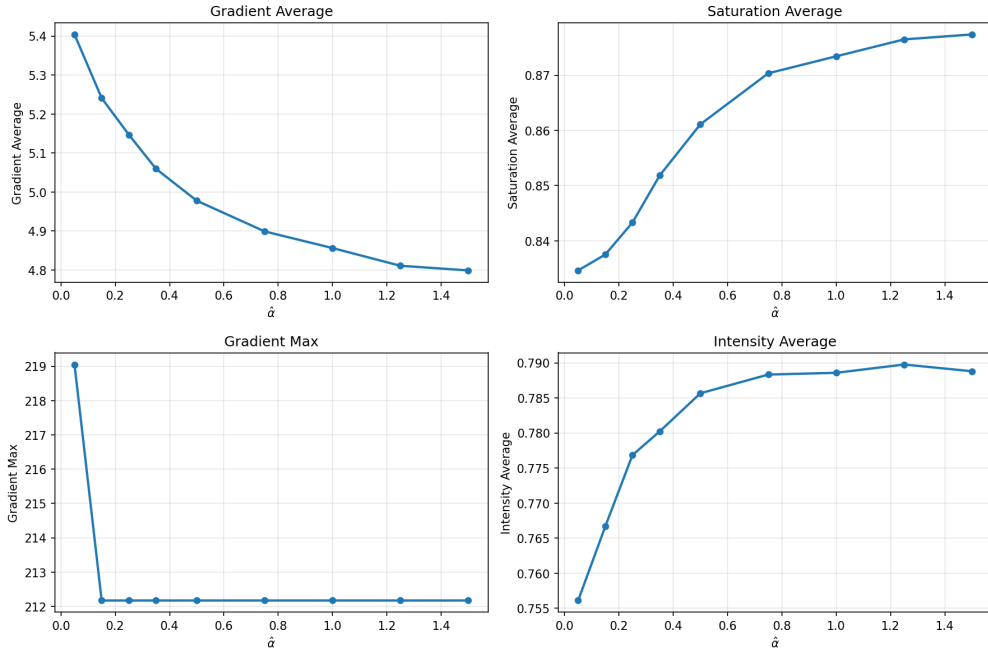


Figure 4.9: Plots of $\hat{\alpha}$ versus each of the metrics examined above.

From this, we can see that although initial improvements are large when increasing $\hat{\alpha}$, these improvements decrease quickly and then stop for each metric by around $\hat{\alpha} = 1.25$, so we continue with this as the selected value of $\hat{\alpha}$.

We can also compare these results with a much simpler and more naive method, Gaussian filtering the original lookup table from Kwon et al. (Figure 4.10).

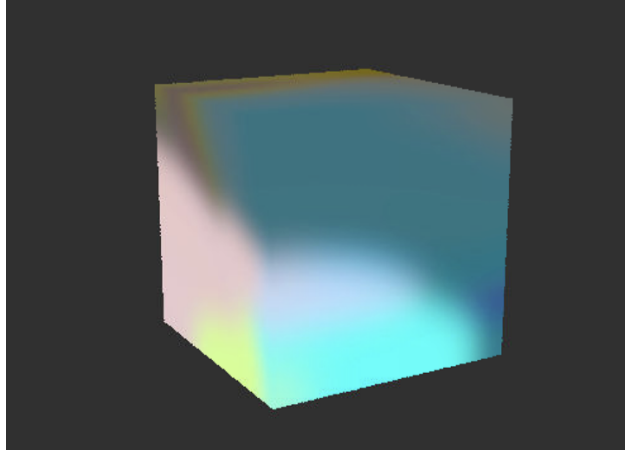


Figure 4.10: Lookup table found by simply smoothing the original lookup table.

Method	Gradient Maximum	Gradient Average	Saturation Average	Intensity Average
Original	163.267	7.624	0.659	0.502
Smoothed	5.685	2.965	0.590	0.481
New	212.176	4.799	0.878	0.789

Table 4.4: Comparison of the original and new lookup tables over time, alongside the Gaussian-filtered lookup table from Figure 4.10 ("Smoothed").

Smoothing the lookup table directly is a problematic approach for a number of reasons, which become clear in Figure 4.10 and Table 4.4. First, this method can only produce new colors on the lookup table that are between two neighboring colors on the original lookup table, limiting the possibility of many new colors. Further, taking the mathematical average of two colors that are perceptually distinct can produce a much duller color, with many averages tending towards browns or grays with low saturation and intensity, that is no longer perceptually distinct from the input. Overall, smoothing the whole lookup table as a final step smooths it out, but reduces visual contrast and results in dull colors. We instead use our method to smooth the lookup table while increasing visual contrast by expanding the search for smoother colors across the whole color space, instead of being limited to the interpolation between two colors.

As we can see in Table 4.4, the original lookup table has low saturation and intensity, but a much higher average gradient than the new lookup table, indicating the presence of many harsh boundaries that the new table lacks. Although the maximum gradient increases with our new method, this is in part because saturation and intensity increase across the lookup table, so the potential for gradient color changes is higher. In addition, the lower average indicates that these

discontinuous boundaries are much less common in our new formulation of the lookup table.

Finally, we can further examine the color distributions across the lookup tables using histograms of the hue of each of the pixel colors represented in the final lookup table. Each pixel color in a lookup table represents the output farthest color for one 8-bit sRGB color. In these, we can see clearly how Euclidean or ΔE_{00} distances cause severe clustering of colors and low diversity across the whole lookup table (Figure 4.11, Figure 4.12). In turn, this causes jumps and instability in the output farthest colors because there are no transition colors possible between the colors of high occurrence in the table, so it is impossible to have a smooth change between label colors. This makes clear that a stable lookup table must have a more even distribution across the histogram. In particular, the histogram should not have significant gaps along the hue axis, as this reduces the colors available to provide a smooth gradient.

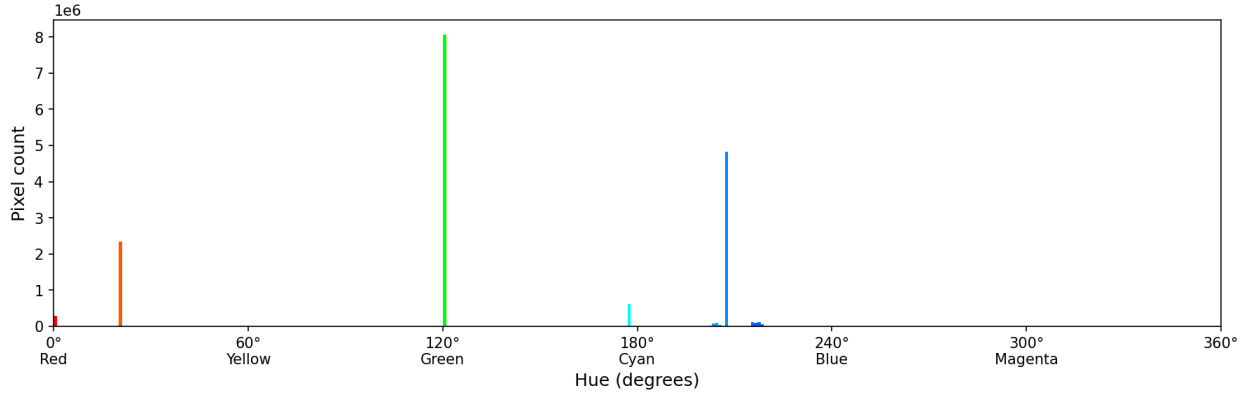


Figure 4.11: Hue distribution of pixel colors in the lookup table using Euclidean distances in the CIELAB space.

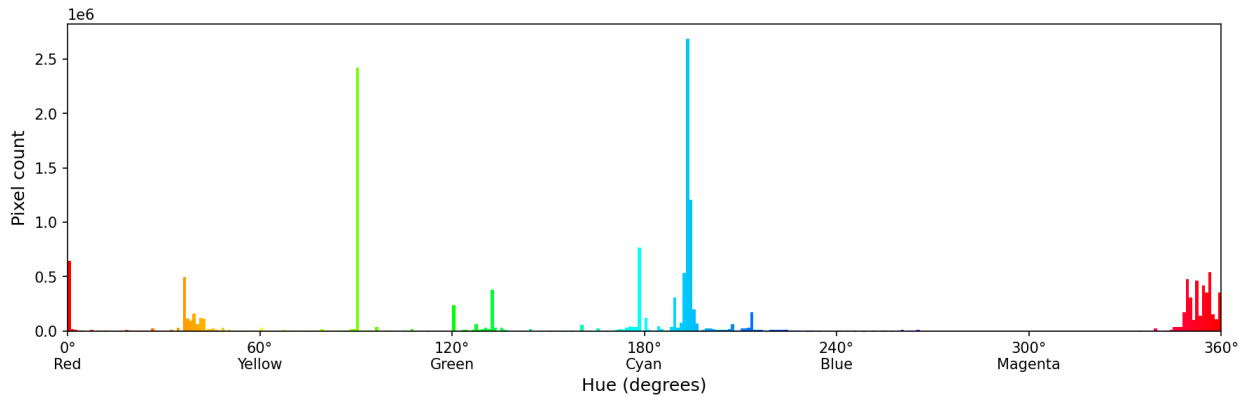


Figure 4.12: Hue distribution of pixel colors in the lookup table using ΔE_{00} distances in the CIELAB space (original method of Kwon et al.).

In comparison, our method creates much more diversity of color to create smoother boundary transitions, which can be seen in the hue histogram (Figure 4.13).

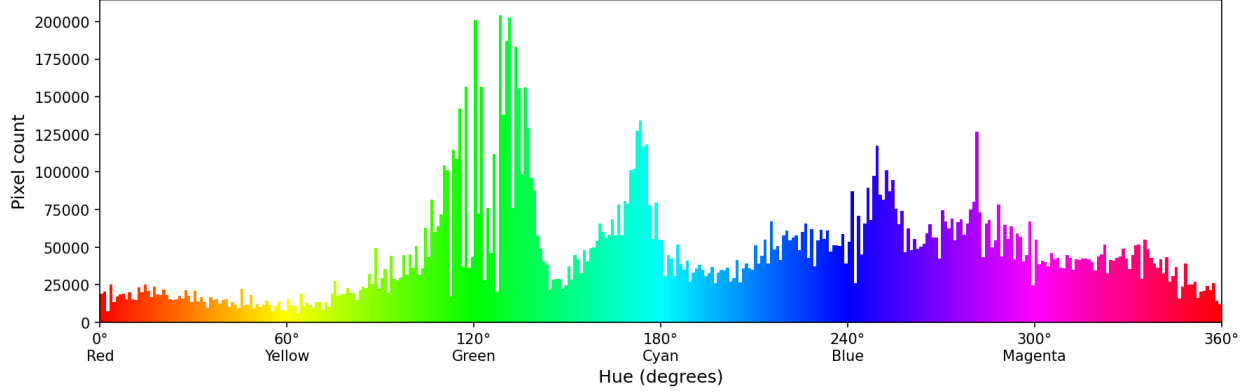


Figure 4.13: Hue distribution of pixel colors in lookup table using regularized geodesic distances and neural-bounding-based smoothing in the CIELAB space.

With this input, our final method generates a lookup table using neural-bounding-based smoothing on the CIELAB color gamut, with regularized geodesic distances with $\hat{\alpha} = 1.25$.

4.3 AR Scene Results

With our new lookup table, we then test our method on dynamic AR scenes, using the overall method of Kwon et al. and comparing the results with area label overlays generated using their lookup table. We can then calculate the gradient of the color of the area label over time (across each frame) to analyze the efficacy of the new lookup table in reducing flicker in the label color and calculate the intensity and saturation of the area label colors. In particular, we focus on the maximum gradient across each method, because there is a high gradient at a moment of flicker. We also compare against the naively smoothed lookup table from Figure 4.10.

Method	Gradient Max.	Gradient Avg.	Gradient SD	Saturation Avg.	Intensity Avg.
Original	217.954	2.806	23.404	0.763	1.0
Smoothed	74.330	3.956	10.490	0.676	0.987
New	41.605	3.760	6.786	0.423	0.977

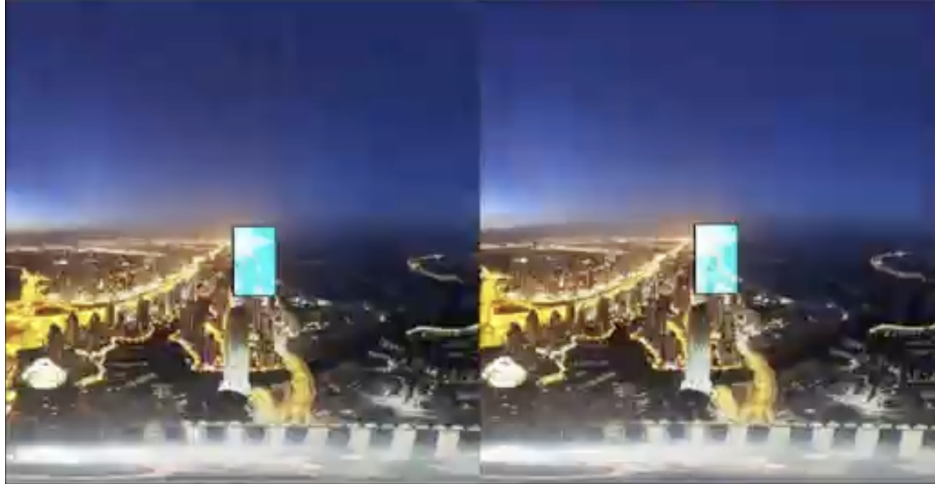
Table 4.5: Comparison of the *area label colors* generated using the original and new lookup tables over time, alongside the Gaussian-filtered lookup table from Figure 4.10 ("Smoothed"). SD = standard deviation.

In Table 4.5, the maximum gradient is greatly reduced ($5\times$) with the new method over the original. The gradient average goes up slightly, but only high gradients lead to flicker, as some change in the label color is necessary to maintain visual contrast in a dynamic environment. Our new lookup table is further supported by the lower standard deviation in the gradient. This indicates that gradients across each frame are mostly low (around 3.760), whereas gradients from the original lookup table often go much higher, as reflected in the much higher standard deviation.

The Gaussian-filtered version of the lookup table (Figure 4.10) reduces the gradient from the original, but does not do as well as the full new method. More significantly, color diversity and richness remain low, as with the original table, and visual contrast starts to decrease. This can be seen in the muddled borders between colors and the decreased saturation and intensity in these area label colors, as shown in Table 4.5, and across the whole lookup table, as shown in Table 4.4.

The new lookup table itself, as shown by Table 4.4, increases saturation and intensity over the original lookup table from Kwon et al., but this scene in particular uses a section of the original lookup table that had a high intensity and saturation to begin with. Our lookup table increases areas of very low saturation and intensity, but in smoothing the whole table also reduces saturation in very high areas, which is what we see here. In this instance, saturation and intensity decrease in the area labels with the significant reduction in flicker, but in other parts of the lookup table, and with other example scenes with different characteristic environment colors, saturation and intensity increase, as evidenced by the increased average across the lookup table.

We can also visualize the reduced flicker with the new method in specific instances of flicker in results generated with the original lookup table. We select 6 frames around the moment of flicker to demonstrate how the new lookup table changes the smoothness of the label color (Figure 4.14, Figure 4.15). In particular, we note that the area label colors across the two videos generated with the new and original methods are very similar in the first and final frames, but vary in the intervening frames. The colors are much closer frame-to-frame on the right (new method) than on the left (original method), in particular between frames 2 and 3.



(a) Frame 1

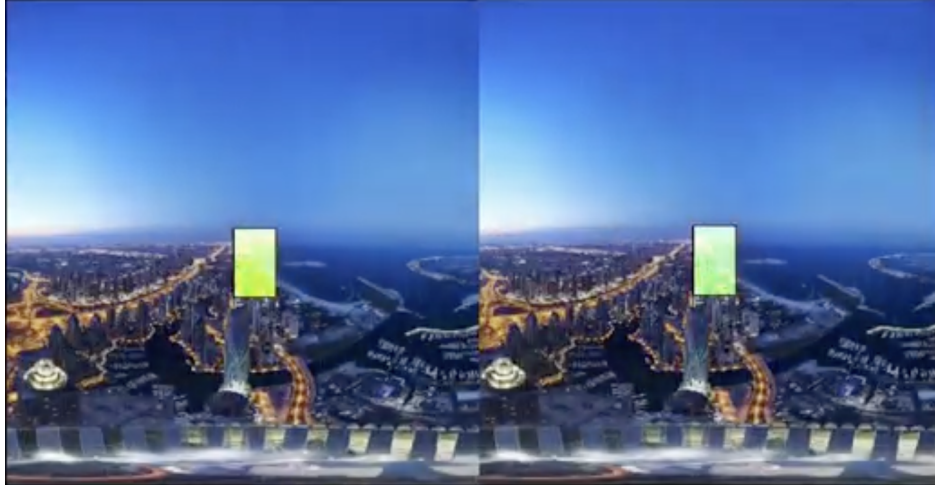


(b) Frame 2

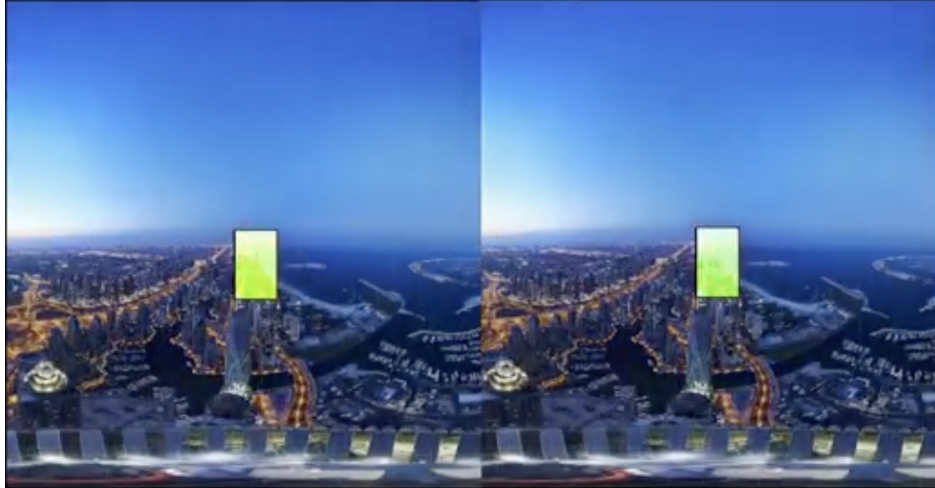


(c) Frame 3

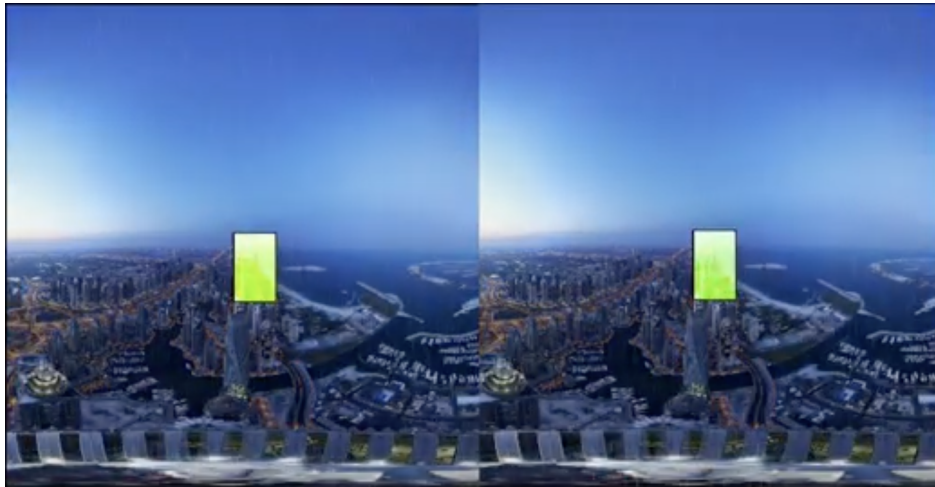
Figure 4.14: First three frames surrounding the moment of flicker between frames 2 and 3 in the original video. Left: video generated with original lookup table. Right: video generated with new lookup table.



(a) Frame 1



(b) Frame 2



(c) Frame 3

Figure 4.15: Second three frames, after the moment of flicker (see Figure 4.14 for moment of flicker). Left: video generated with original lookup table. Right: video generated with new lookup table.

5 Conclusion

5.1 Summary of Contributions

We identify three sources of discontinuity in the formulation of a farthest-color lookup table and smooth each to produce a lookup table that is stable under small input changes but maintains perceptual distance between input and output colors. We formulate a new method for forming lookup tables, using neural-bounding-based smoothing and regularized geodesic distances in the CIELAB color space, and show increased smoothness and saturation when compared to existing methods. When using our lookup table to color augmented reality area labels, we show a $5\times$ reduction in maximum gradient to greatly reduce flicker in dynamic environments. We also investigate a number of other methods for generating lookup tables, including using other intermediate color spaces, using Euclidean or ΔE_{00} distance measures, and using Gaussian filter smoothing on the color space, and their effects on the lookup table. We hope that these explorations and findings will be helpful in making graphics and visualizations that require a farthest-color lookup more accessible and easier to use for the general population, across a variety of applications.

5.2 Future Work

First, we focus in this work on smoothing the lookup table to remove discontinuities in the mapping from input to output colors, but to stabilize this mapping we could also smooth the input colors. This intervention would be application-dependent because input colors are determined differently based on applications. For example, we could modify the process to find characteristic environment colors to make it more robust to moving objects. This could help stabilize the output to a degree, but discontinuities would still exist in the lookup table itself.

We investigate three sources of discontinuity in the farthest-color lookup table, but there are many other potential areas of improvement. For one, we select the output color to be the absolute farthest color by the chosen distance measure from the input color, but selecting from a pool of the top n farthest colors from the input color with knowledge of surrounding farthest colors in the lookup table could further smooth discontinuities without sacrificing much perceptual distance. There are also many other techniques to modify these sources of discontinuity that could help further improve the results. For example, in gamut smoothing, Laplacian smoothing on the triangle mesh wrapped around the gamut of the intermediate color space might be effective. Further, more complicated notions of color difference and color space, like neural-network-based metrics for color differences in photographs [15], might be usable as distance measures that are more perceptually accurate than current methods. A user study on our results would also better establish their efficacy.

Finally, using a lookup table to precompute farthest colors with time-varying input data disallows using the time-varying data to improve stability of farthest colors. The precomputed lookup table will always produce the same farthest color from an input color, regardless of colors at nearby

times. To incorporate the added information time-varying data provides, further work could also explore integrating one or more previous farthest colors to form the next farthest color from a series of input colors. For example, to maintain real-time or close performance, we could precompute a 4-dimensional lookup table that stores the top n farthest colors from an input instead of just one farthest color, which would allow for a real-time algorithm to use past farthest color data to select from the top n farthest colors to find the one that will minimize the gradient from the previous farthest color to the current farthest color.

References

- [1] S. Levy, *An intro to color spaces (and how I used them for my game)*, <https://medium.com/lightricks-tech-blog/an-intro-to-color-spaces-and-how-i-used-them-for-my-game-f712153b114>, Lightricks Tech Blog, Medium, Feb. 2022. Accessed: Apr. 6, 2026.
- [2] B. Ottosson, *A perceptual color space for image processing*, <https://bottosson.github.io/posts/oklab/>, Jan. 2021.
- [3] S. M. Wuerger, L. T. Maloney, and J. Krauskopf, “Proximity judgments in color space: Tests of a euclidean color geometry.,” *Vision research*, vol. 35 6, pp. 827–35, 1995. [Online]. Available: <https://api.semanticscholar.org/CorpusID:7890608>.
- [4] R. Bujack, E. Teti, J. Miller, E. Caffrey, and T. L. Turton, “The non-riemannian nature of perceptual color space,” *Proceedings of the National Academy of Sciences*, vol. 119, no. 18, e2119753119, 2022. DOI: [10.1073/pnas.2119753119](https://doi.org/10.1073/pnas.2119753119). eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.2119753119>. [Online]. Available: <https://www.pnas.org/doi/abs/10.1073/pnas.2119753119>.
- [5] D. R. Pant and I. Farup, “Riemannian formulation and comparison of color difference formulas,” *Color Research and Application*, vol. 37, pp. 429–440, 2012. [Online]. Available: <https://api.semanticscholar.org/CorpusID:59044042>.
- [6] D. R. Pant and I. Farup, “Geodesic calculation of color difference formulas and comparison with the munsell color order system,” *Color Research & Application*, vol. 38, no. 4, pp. 259–266, 2013. DOI: <https://doi.org/10.1002/col.20751>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/col.20751>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/col.20751>.
- [7] J. Fritz et al., “Augmented reality visualization with use of image overlay technology for mr imaging-guided interventions: Assessment of performance in cadaveric shoulder and hip arthrography at 1.5 t,” *Radiology*, vol. 265, no. 1, pp. 254–259, 2012, PMID: 22843764. DOI: [10.1148/radiol.12112640](https://doi.org/10.1148/radiol.12112640). eprint: <https://doi.org/10.1148/radiol.12112640>. [Online]. Available: <https://doi.org/10.1148/radiol.12112640>.
- [8] B. Ridel, P. Reuter, J. Laviole, N. Mellado, N. Couture, and X. Granier, “The revealing flashlight: Interactive spatial augmented reality for detail exploration of cultural heritage artifacts,” *J. Comput. Cult. Herit.*, vol. 7, no. 2, Jun. 2014, ISSN: 1556-4673. DOI: [10.1145/2611376](https://doi.org/10.1145/2611376). [Online]. Available: <https://doi.org/10.1145/2611376>.
- [9] L. C. Lindquist, G. R. McIntire, and S. M. Haigh, “The effects of visual discomfort and chromaticity separation on neural processing during a visual task,” *Vision Research*, vol. 182, pp. 27–35, 2021. DOI: <https://doi.org/10.1016/j.visres.2021.01.007>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0042698921000249>.

-
- [10] H. Kwon, Y. Li, X. Ye, P. Muna-McQuay, L. Yin, and J. Tompkin, “Active appearance and spatial variation can improve visibility in area labels for augmented reality,” in *Proc. VIS*, 2024, pp. 206–210. DOI: [10.1109/VIS55277.2024.00049](https://doi.org/10.1109/VIS55277.2024.00049).
 - [11] S. W. Liu, M. Fischer, P. D. Yoo, and T. Ritschel, *Neural bounding*, 2024. arXiv: [2310.06822](https://arxiv.org/abs/2310.06822) [cs.GR]. [Online]. Available: <https://arxiv.org/abs/2310.06822>.
 - [12] M. Edelstein, N. Guillen, J. Solomon, and M. Ben-Chen, “A convex optimization framework for regularized geodesic distances,” in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, ser. SIGGRAPH ’23, ACM, Jul. 2023, pp. 1–11. DOI: [10.1145/3588432.3591523](https://doi.org/10.1145/3588432.3591523). [Online]. Available: <http://dx.doi.org/10.1145/3588432.3591523>.
 - [13] E. Marino, F. Bruno, L. Barbieri, M. Muzzupappa, and F. Liarokapis, “Background-aware colorization technique for augmented reality applications,” *IEEE Access*, vol. 9, pp. 161 761–161 772, 2021. DOI: [10.1109/ACCESS.2021.3129076](https://doi.org/10.1109/ACCESS.2021.3129076).
 - [14] H. Hagtvedt and S. A. Brasel, “Color saturation increases perceived product size,” *Journal of Consumer Research*, vol. 44, no. 2, pp. 396–413, None 2017. DOI: [None](https://ideas.repec.org/a/oup/jconrs/v44y2017i2p396-413..html). [Online]. Available: <https://ideas.repec.org/a/oup/jconrs/v44y2017i2p396-413..html>.
 - [15] H. Chen, Z. Wang, Y. Yang, Q. Sun, and K. Ma, “Learning a deep color difference metric for photographic images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 242–22 251.

A Supplementary Code and Equations

```
1 import numpy as np
2
3 def RGBtoLAB(RGB):
4     RGB = np.array(RGB) / 255.0
5     mask = RGB > 0.04045
6
7     RGB[mask] = ((RGB[mask] + 0.055) / 1.055) ** 2.4
8     RGB[~mask] /= 12.92
9     RGB *= 100
10
11
12     XYZ = np.dot(RGB, np.array([[0.4124, 0.3576, 0.1805],
13                                   [0.2126, 0.7152, 0.0722],
14                                   [0.0193, 0.1192, 0.9505]]))
15
16     XYZ /= np.array([95.047, 100.0, 108.883])
17     mask = XYZ > 0.008856
18     XYZ[mask] = XYZ[mask] ** (1/3)
19     XYZ[~mask] = (7.787 * XYZ[~mask]) + (16/116)
20
21     L = (116 * XYZ[:, 1]) - 16
22     a = 500 * (XYZ[:, 0] - XYZ[:, 1])
23     b = 200 * (XYZ[:, 1] - XYZ[:, 2])
24
25     return np.stack([L, a, b], axis=1)
```

Listing A.1: Algorithm to find CIELAB values for RGB color inputs.

```
1 import numpy as np
2
3 def RGBtoOKLAB(RGB):
4     RGB = np.array(RGB) / 255.0
5     mask = RGB > 0.04045
6
7     RGB[mask] = ((RGB[mask] + 0.055) / 1.055) ** 2.4
8     RGB[~mask] /= 12.92
9
10     l = 0.4122214708 * RGB[:,0] + 0.5363325363 * RGB[:,1] + 0.0514459929 *
11         RGB[:,2]
12     m = 0.2119034982 * RGB[:,0] + 0.6806995451 * RGB[:,1] + 0.1073969566 *
13         RGB[:,2]
```

```
12     s = 0.0883024619 * RGB[:,0] + 0.2817188376 * RGB[:,1] + 0.6299787005 *  
13         RGB[:,2]  
14  
14     l_ = np.cbrt(l)  
15     m_ = np.cbrt(m)  
16     s_ = np.cbrt(s)  
17  
18     RGB[:,0] = 0.2104542553 * l_ + 0.7936177850 * m_ - 0.0040720468 * s_  
19     RGB[:,1] = 1.9779984951 * l_ - 2.4285922050 * m_ + 0.4505937099 * s_  
20     RGB[:,2] = 0.0259040371 * l_ + 0.7827717662 * m_ - 0.8086757660 * s_  
21  
22     return RGB
```

Listing A.2: Algorithm to find Oklab values for RGB color inputs.

$$\Delta L' = L_2 - L_1 \quad (\text{A.1})$$

$$C_1 = \sqrt{a_1^2 + b_1^2}, C_2 = \sqrt{a_2^2 + b_2^2} \quad (\text{A.2})$$

$$\bar{L} = \frac{L_1 + L_2}{2}, \bar{C} = \frac{C_1 + C_2}{2} \quad (\text{A.3})$$

$$a'_1 = a_1 + \frac{a_1}{2}(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}), a'_2 = a_2 + \frac{a_2}{2}(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}) \quad (\text{A.4})$$

$$C'_1 = \sqrt{a'^2_1 + b_1^2}, C'_2 = \sqrt{a'^2_2 + b_2^2} \quad (\text{A.5})$$

$$\bar{C}' = \frac{C'_1 + C'_2}{2}, \Delta C' = C'_2 - C'_1 \quad (\text{A.6})$$

$$h'_1 = \text{atan2}(b_1, a'_1) \mod 360^\circ, h'_2 = \text{atan2}(b_2, a'_2) \mod 360^\circ \quad (\text{A.7})$$

$$\Delta h' = \begin{cases} h'_2 - h'_1 & \text{if } |h'_2 - h'_1| < 180^\circ \\ (h'_2 - h'_1) - 360^\circ & \text{if } (h'_2 - h'_1) > 180^\circ \\ (h'_2 - h'_1) + 360^\circ & \text{if } (h'_2 - h'_1) < -180^\circ \end{cases} \quad (\text{A.8})$$

$$\bar{h}' = \begin{cases} (\frac{h'_1 + h'_2}{2}) & \text{if } |h'_1 - h'_2| \leq 180^\circ \\ (\frac{h'_1 + h'_2 + 360^\circ}{2}) & \text{if } |h'_1 - h'_2| > 180^\circ, h'_1 + h'_2 < 360^\circ \\ (\frac{h'_1 + h'_2 - 360^\circ}{2}) & \text{if } |h'_1 - h'_2| > 180^\circ, h'_1 + h'_2 \geq 360^\circ \end{cases} \quad (\text{A.9})$$

$$\Delta H' = 2\sqrt{C'_1 C'_2} \sin(\frac{\Delta h'}{2}) \quad (\text{A.10})$$

$$T = 1 - 0.17\cos(\bar{h}' - 30^\circ) + 0.24\cos(2\bar{h}') + 0.32\cos(3\bar{h}' + 6^\circ) - 0.20\cos(4\bar{h}' - 63^\circ) \quad (\text{A.11})$$

$$S_L = 1 + \frac{0.015(\bar{L} - 50)^2}{\sqrt{20 + (\bar{L} - 50)^2}}, S_C = 1 + 0.045\bar{C}', S_H = 1 + 0.015\bar{C}'T \quad (\text{A.12})$$

$$R_T = -2\sqrt{\frac{\bar{C}'^7}{\bar{C}'^7 + 25^7}} \sin(60^\circ \cdot \exp(-(\frac{\bar{h}' - 275^\circ}{25^\circ})^2)) \quad (\text{A.13})$$

$$\Delta E_{00} = \sqrt{(\frac{\Delta L'}{k_L S_L})^2 + (\frac{\Delta C'}{k_C S_C})^2 + (\frac{\Delta H'}{k_H S_H})^2 + R_T \frac{\Delta C'}{k_C S_C} \frac{\Delta H'}{k_H S_H}} \quad (\text{A.14})$$

Listing A.3: CIELAB ΔE_{00} distance between two CIELAB colors (L_1, a_1, b_1) and (L_2, a_2, b_2) with parametric weighing factors k_L, k_C, k_H .