Color Classification in CIELAB Color Space: Algorithmic Design and Explanations

Yanik Wettstein, OpenAI ChatGPT

Abstract

We present a structured description of a color classification pipeline implemented in Python. The algorithm operates in the CIE L*a*b* color space [1] to provide a perceptually uniform representation of color, then separates chromatic and achromatic pixels, quantizes color values into bins, and optionally clusters these bins. Mathematical formulas and pseudocode are provided to elucidate the steps. Variables and parameters (thresholds, step sizes) are explained. For further reading, see the referenced Wikipedia articles.

Disclaimer

This project represents the outcome of a voluntary research initiative and should not be construed as formal scientific work. The methodologies employed have not undergone rigorous scientific validation procedures, and the testing has been conducted on a limited and non-comprehensive dataset. As such, the findings and conclusions presented herein should be interpreted with appropriate caution and are intended primarily for exploratory and educational purposes.

The complete source code for this project is publicly available and can be accessed via the project's GitHub repository at
GitHub. Users are encouraged to review, test, and contribute to the codebase as part of the open-source community.

For inquiries regarding efficiency improvements, accuracy recommendations, or any other questions and remarks pertaining to this project, please direct your correspondence to Yanik Wettstein at yanik.wettstein@gmail.com.

Due to time constraints and in the interest of efficient documentation, portions of the following text have been automatically generated using OpenAI's ChatGPT-5 language model, working from the raw Python source code. However, all generated content has been thoroughly reviewed and approved by Yanik Wettstein to ensure accuracy and appropriateness. Any errors or inconsistencies remain the responsibility of the project author.

1 Introduction

Color analysis in images often benefits from using a perceptually uniform color space. The CIE L*a*b* space (CIELAB) represents color with one lightness coordinate (L^*) and two chromatic coordinates (a^*, b^*) [1]. The lightness L^* ranges from 0 (black) to 100 (white), while a^* and b^* capture opponent color dimensions (green–red and blue–yellow, respectively) [1]. A chromatic color has significant a^* or b^* values, whereas an achromatic (gray-scale) color has negligible chromatic values. Working in Lab space makes Euclidean distance roughly correspond to perceptual color difference [2], which is advantageous for clustering and quantization.

2 Color Space Conversion

The pipeline begins by loading an image in RGB and converting each pixel to CIE L*a*b*. Standard conversion (implemented via scikit-image or similar) first linearizes RGB values, transforms to CIE XYZ, and then applies the nonlinear Lab transformation [1]. In formula form, given normalized RGB, one computes tristimulus values (X, Y, Z) and then:

$$L^* = 116\, f(Y/Y_n) - 16, \quad a^* = 500\big[f(X/X_n) - f(Y/Y_n)\big], \quad b^* = 200\big[f(Y/Y_n) - f(Z/Z_n)\big],$$

where $f(t) = t^{1/3}$ for $t > (6/29)^3$ (otherwise a linear expression) and (X_n, Y_n, Z_n) is the reference white [3]. In practice, libraries provide this conversion directly. The output is an $N \times 3$ array of type float, where N is the total number of pixels.

3 Chromatic vs. Achromatic Split

Each pixel is classified as *chromatic* or *achromatic* (nearly gray) using thresholds on lightness and chroma. Let L_i, a_i, b_i be the Lab channels of pixel i, and define chroma $C_i = \sqrt{a_i^2 + b_i^2}$. A pixel is marked achromatic if either

$$L_i < L_{\text{thresh}} \quad \text{or} \quad C_i < C_{\text{thresh}}\,,$$

and chromatic otherwise [2]. Here L_{thresh} and C_{thresh} are user-defined parameters (e.g. 10 and 6). In code, this produces boolean masks $\mathbf{M}_{\text{achro}}$ and $\mathbf{M}_{\text{chrom}}$ of length N:

$$M_{\text{achro},i} = (L_i < L_{\text{thresh}}) \ \vee \ (C_i < C_{\text{thresh}}), \quad M_{\text{chrom},i} = \neg M_{\text{achro},i}.$$

All pixels with M_{achro} true are grouped as achromatic; the rest are chromatic. This ensures very dark or desaturated pixels are treated as gray-scale [2].

4 Quantization of Chromatic Components

Within each group (chromatic or achromatic), we quantize the chromatic channels (a^*, b^*) into discrete bins. Specifically, let Δ_{ab} be the quantization step. We compute integer bin indices by:

$$a_{\text{bin},i} = \text{round}(a_i/\Delta_{ab}), \quad b_{\text{bin},i} = \text{round}(b_i/\Delta_{ab}).$$

For example, $\Delta_{ab} = 1$ maps each pixel's a^* and b^* to the nearest integer. This quantization groups similar hues together [4].

4.1 Stacking Pixels by Bin

After quantization, pixels are grouped by their $(a_{\text{bin}}, b_{\text{bin}})$. Let the chromatic group have N_{chrom} pixels and the achromatic group N_{achro}. For each group, form the set of pixels' Lab values. Identify the unique bin coordinates $\{(a_{\text{bin}}^{(k)}, b_{\text{bin}}^{(k)})\}_{k=1}^{K}$ that occur. For each bin k, let I_k be the set of indices of pixels falling into that bin. Define:

$$\begin{aligned}
\text{count}_k &= |I_k|, \\
L_{\text{rep}}^{(k)} &= \text{median}\{L_i : i \in I_k\}, \\
a_{\text{rep}}^{(k)} &= \frac{1}{|I_k|} \sum_{i \in I_k} a_i, \\
b_{\text{rep}}^{(k)} &= \frac{1}{|I_k|} \sum_{i \in I_k} b_i, \\
C_{\text{rep}}^{(k)} &= \text{median}\{\sqrt{a_i^2 + b_i^2} : i \in I_k\}.
\end{aligned}$$

Thus each bin is represented by its pixel count and typical Lab values [1, 4].

4.2 Pseudocode for Binning

5 K-Means Clustering of Bins (Optional)

To further simplify the palette, the representative Lab colors of all bins can be clustered. Apply K-means clustering [5] on the matrix of bin representatives $\{(L_{\text{rep}}^{(k)}, a_{\text{rep}}^{(k)}, b_{\text{rep}}^{(k)})\}$ with a chosen number of clusters K_{cluster}. K-means partitions the points into K_{cluster} sets $\{S_j\}$ by minimizing:

$$\min_{\{S_j\},\{\mu_j\}} \sum_{j=1}^{K_{\text{cluster}}} \sum_{x_i \in S_j} \|x_i - \mu_j\|^2.$$

The algorithm iterates: assign each point to the nearest centroid, then recompute each centroid as the mean of its assigned points [5].

Algorithm 1 Pixel Stacking by (a^*, b^*) Bins

Input: Lab pixels $\{(L_i, a_i, b_i)\}_{i=1}^{N}$, step Δ_{ab}
for $i = 1$ to N **do**
    $a_{\text{bin},i} \leftarrow \text{round}(a_i/\Delta_{ab})$
    $b_{\text{bin},i} \leftarrow \text{round}(b_i/\Delta_{ab})$
end for
Identify unique bins $\{(a_{\text{bin}}^{(k)}, b_{\text{bin}}^{(k)})\}_{k=1}^{K}$
for $k = 1$ to K **do**
    $I_k \leftarrow \{i : (a_{\text{bin},i}, b_{\text{bin},i}) = (a_{\text{bin}}^{(k)}, b_{\text{bin}}^{(k)})\}$
    $\text{count}_k \leftarrow |I_k|$
    $L_{\text{rep}}^{(k)} \leftarrow \text{median}\{L_i : i \in I_k\}$
    $a_{\text{rep}}^{(k)} \leftarrow \frac{1}{|I_k|} \sum_{i \in I_k} a_i$
    $b_{\text{rep}}^{(k)} \leftarrow \frac{1}{|I_k|} \sum_{i \in I_k} b_i$
    $C_{\text{rep}}^{(k)} \leftarrow \text{median}\{\sqrt{a_i^2 + b_i^2} : i \in I_k\}$
end for
Output: Arrays $L_{\text{rep}}, a_{\text{rep}}, b_{\text{rep}}, C_{\text{rep}}, \text{count}$

6 Conclusions

We have detailed the steps of a color classification algorithm that transforms RGB images into CIELAB, splits pixels by chromaticity, bins by quantized chromatic coordinates, and optionally clusters the results. This document was automatically generated by ChatGPT based on the code; for more background see the Wikipedia articles on *CIELAB color space* [1] and *k-means clustering* [5].

References

[1] Wikipedia, "CIELAB color space," `https://en.wikipedia.org/wiki/CIELAB_color_space`.

[2] Wikipedia, "Color difference," `https://en.wikipedia.org/wiki/Color_difference`.

[3] Wikipedia, "CIE 1931 color space," `https://en.wikipedia.org/wiki/CIE_1931_color_space`.

[4] Wikipedia, "Quantization (image processing)," `https://en.wikipedia.org/wiki/Quantization_(image_processing)`.

[5] Wikipedia, "K-means clustering," `https://en.wikipedia.org/wiki/K-means_clustering`.