An Analysis of the Efficacy of Different Color-Spaces in Extracting Color-Schemes From Images

Introduction:

The process of extracting a color-scheme from an image is functionally the same as clustering the various RGB values within an image using some clustering algorithm such as K-means. Most of the work done in this area by open-source developers seem to focus exclusively on applying this clustering to color values in the RGB space rather than alternatives such as CIERGB or XYZ. The purpose of my work is to investigate how changing the color space used for clustering may produce different color palettes for the same image, and which color space is qualitatively the best at certain tasks.

Related Works

- Dominant colors in an image using k-means clustering
    - https://buzzrobot.com/dominant-colors-in-an-image-using-k-means-clustering-3c7af4622036
    - This article showcases the barebones foundation of the process typically used to cluster, and thereby generate, color-schemes from images. The code I used to render generated color-schemes using a histogram is taken directly from his source code.
- HSP Color Model - Alternative to HSV (HSB) and HSL
    - http://alienryderflex.com/hsp.html
    - The original code that rendered a histogram of colors ordered the colors by the frequency with which they appeared in the input image. I decided to investigate some other ways of ordering these images, the first of which was brightness. The equation I used to compute brightness was taken from this website.
- The incredibly challenging task of sorting colours
    - https://www.alanzucconi.com/2015/09/30/colour-sorting/
    - This article talks about the many different ways to sort colors, and how all of them are in some way inadequate. The code I used from this website was applied to my histogram_hsv() function.
- RGB To Grayscale
    - https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_gray.html
    - How to obtain a single black and white value given RGB channel values. I used this in my posterization method to turn certain regions grayscale.

Procedure

The first step in computing these color palettes is the loading of the image and the conversion of it from the native RGB space to any of the other spaces; this step is omitted for the RGB example. The library used for this conversion is SCI-KIT Color. Both before and after the conversion, the dimensionality of the image is I x J x 3, where I and J are the dimensions of the image and the 3 correspond to three channels, which is the universal number of channels for all color spaces used. This array is then flattened into an array of size IJ x 3 where IJ is the product of I and J and fed into a KMeans model. KMeans is an iterative approach to clustering that produces a user-specified number of clusters, and cluster centers. The initialization step involves assigning cluster centers to random points sampled from the input data, which in this case are colors. Then the algorithm iterates through each point in the dataset and assigns that point to the cluster center that minimizes the geometric distance between it and the cluster center. Once this is done, cluster centers are recomputed using all the newly updated points in that cluster, and the cycle continues. The algorithm terminates when cluster centers and point assignments stop updating. The computed cluster centers are then returned to the user as a list, constituting the color palette of the image.

Because of the random nature of KMeans, cluster center 1 in one run of the algorithm on the image may be red but there is no guarantee that future runs will still maintain this association. As such the order with which these colors are displayed to the user is of great importance. The first color histogram method orders the colors from most frequent to least frequent. The second orders the colors by bright to dark. The third orders them by their position in the hue wheel. I was unable to experiment with this more due to time constraints and lack of benefit, as it seems frequency is considered by most people I talked to to be ideal.

In addition to just presenting the user with these extracted colors, I have also implemented an algorithm that “posterizes” the input image by remapping each pixel’s color value to a color in the color-scheme that minimizes the geometric distance between the two. As the professor suggested, displaying the image exclusively using the computed color-scheme may help users distinguish between different color spaces.

Notebooks:

Demo:

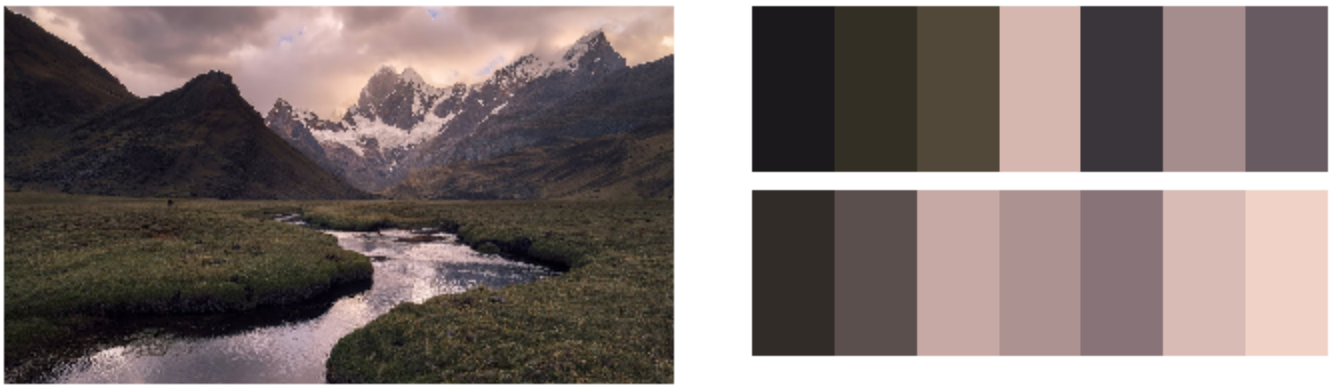
A collection of photographs and digital art obtained from Google Images is processed with all the different color spaces I’ve implemented and the generated color-schemes for each image are displayed to the user ordered by frequency. The purpose of this notebook is to give the reader a general overview of the unique characteristics inherent to each color space.

I asked several of my friends to look at these and the general consensus was LAB, CIERGB and XYZ are the most appealing. Some interesting comments I remember include how they liked

LAB because “the colors seem more evenly spaced and cohesive”. I believe the reason for this is precisely the fact that LAB is a perceptual color space, and thus a geometric distance computed in this space is more representative of the human visual system’s perception of difference between the two colors. However, these benefits seem to only apply when the image has several smooth gradients between dominant colors and seems to struggle in high contrast images.

RGB (top) LAB(Bottom)

The color-schemes produced by XYZ have been said by several people to be initially more appealing, cohesive, and “aesthetic” than the ones generated by other color schemes. However, the cost of this is that qualitatively speaking, XYZ produces colors that often don’t seem to be particularly representative of the original image. It seems to often omit strong and vibrant colors, and lacks “dark darks” and “bright brights”, producing a color scheme with little contrast.



LAB(top) XYZ(bottom)

RGBCIE generally outperforms RGB and doesn't seem to be particularly bad at anything, but also not particularly pleasing to look at. Though it often produces a different resultant color scheme than RGB, it also sometimes returns the exact same colors in a different order.

RGB(top) CIERGB(bottom)

HSV generally performed the poorest and seems to often produce unexpected colors that don't appear to be immediately visible in the image. This makes the produced color scheme seem "unharmonious" or "discordant", as some of my friends put it.

LAB(top) HSV(bottom)

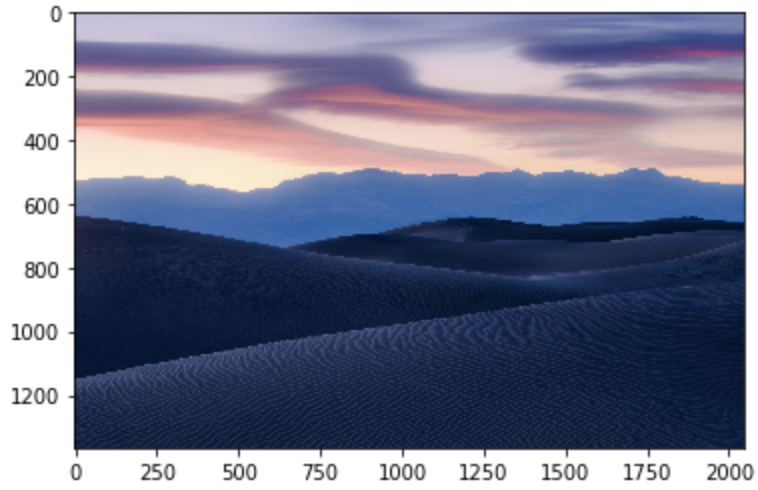
Interactive Posterization:

In a conversation with the professor, he suggested to me that the users might better be able to judge whether a color scheme was desirable by altering the image to only use those computed colors in a posterization process. At this point it became clear to me that there was a functional

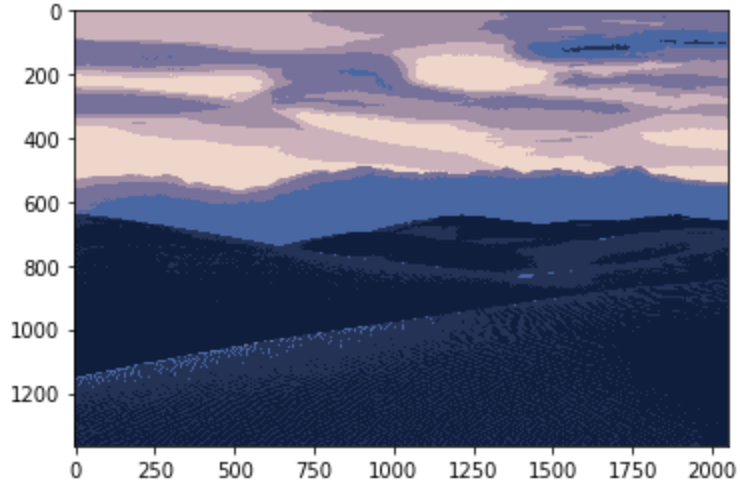
difference between a “pretty” color scheme and a color scheme that could accurately capture the appearance of the original image. This notebook allows the user to select the desired number of clusters, or colors, and which of them should appear in the posterization. If the user omits color 1, for example, then all parts of the image that have been clustered to color 1 will be made grayscale.

Unsurprisingly, qualitatively evaluation of the efficacy of generated color-schemes at reproducing an image accurately is not correlated with how aesthetic the color scheme is.

Original

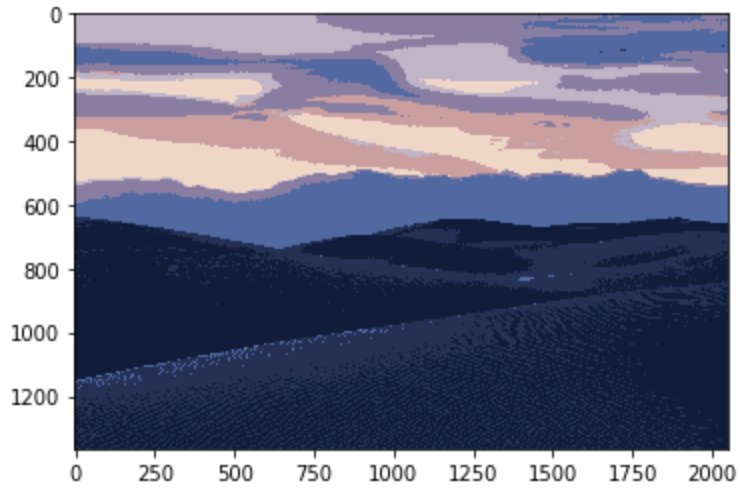


RGB



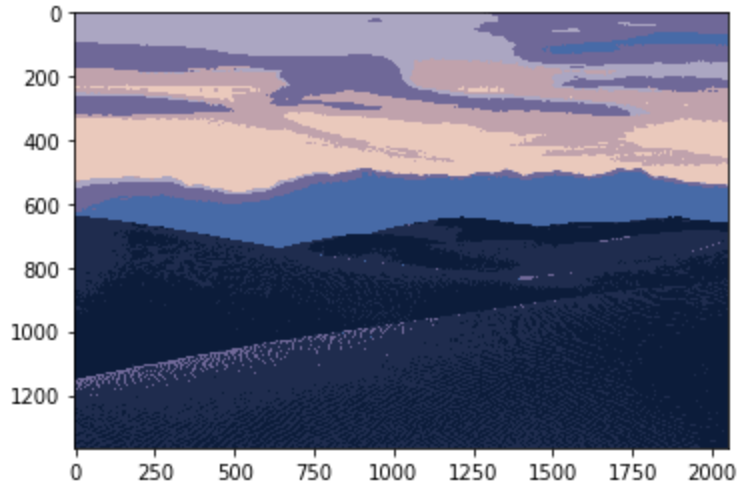
Though the texture and detailings of the sand have been well captured, the coloration of the clouds is starkly unlike the original input image and thus changes the tone of the original image.

LAB



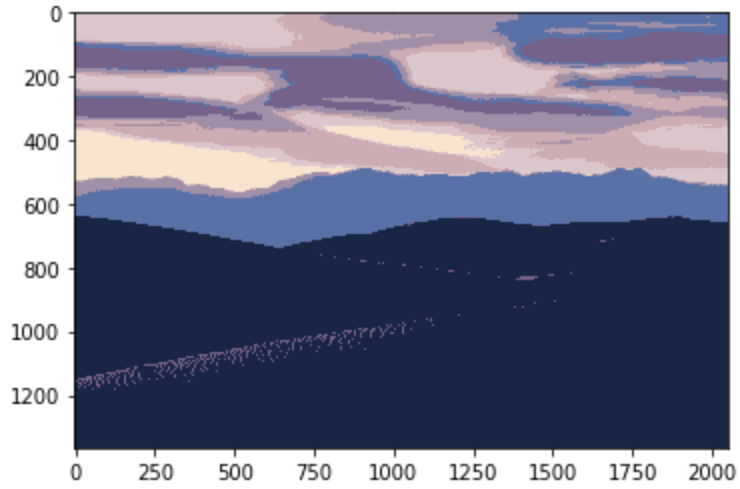
The texture and detailings of the sand are still well preserved but the coloration of the sky and the background mountains seems to be more accurate to our perception of the original image. This advantage likely stems from LAB being a perceptual color space.

HSV

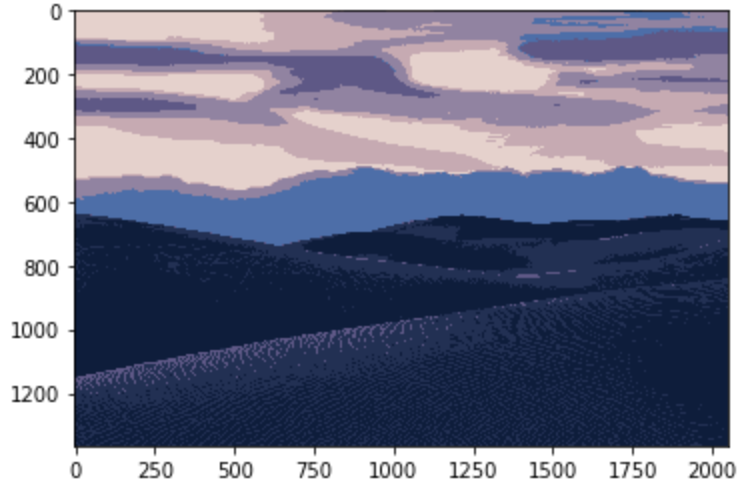


HSV produces colors that are vibrant and “sharp”. Additional color highlights such as the slight purple near the top of the left-lower sand dune are introduced that give the image an interesting stylization to it. However, it does not resemble the original image well, and the clouds no longer look like clouds.

XYZ



The detailings of the sand and mountains are more or less completely lost and produces a uniquely stylized representation of the original image that still allows one to generally understand the tone and subject matter of the original. The output in my opinion looks nice but not particularly good at re-creating the original image. XYZ seems particularly good at recreating the coloration of the clouds and sky.



CIERGB seems to perform a bit better than RGB with the additional purple sand dune coloration that helps capturre the essence of the original. However, the clouds suffer from the same issue as RGB.

Conclusion:

By projecting an image to various different color-schemes it becomes clear to me that each of these compute a color palette from the input image that have substantive qualitative and

quantitative differences from one another. LAB, XYZ and RGBCIE generally seem to perform better and generating pleasing color schemes. LAB and RGBCIE are best when trying to minimize the reconstruction difference between the image and an image using only colors generated with that respective color space.