The Digital Color Analyst

Daniel Munera Martinelli [2049054], Annika Unmüßig [2046421]

June 3, 2024

Contents

Chapter 1

Introduction

1.1 The idea

“Which colors suit me?“, “Why does this color make me look so pale?“, “Is this bright red jacket maybe too much for me?“

Many of us have likely asked ourselves, whether in front of a mirror or while shopping for clothes, which colors suit us best. This question can be particularly vexing for those who are passionate about fashion and dressing well.
Is there a way to determine which colors complement us and which ones to avoid? The answer lies in ”Color Analysis,” also known as ”Seasonal Color Analysis.”
Color Analysis is a method frequently employed in the cosmetics and fashion industries to identify the colors of clothing, makeup, and hairstyles that harmonize with an individual’s skin complexion, eye color, and hair color. This technique is invaluable for wardrobe planning and style consulting.
In 2020, the concept of Color Analysis gained renewed popularity, becoming a viral trend on the social media platform TikTok.

Figure 1.1: Color Analysis service pricing

1.2 Color Analysis

People calling themselves color analysts made themselves a business offering expensive “color consultation” to determine their clients personal “season”.
But what are those “seasons”?
In our project we used a color guide document by the ”Style Coaching Institute” and used their definition of a season.
They define 12 “Seasons”: “Light Spring”, “Bright Spring”, “Warm Spring”, “Light Summer”, “Muted Summer”, “Cool Summer”, “Dark Autumn”, “Muted Autumn”, “Warm Autumn”, “Dark Winter”, “Bright Winter” and lastly “Cool Winter”.

1.2.1 Brief Explanation of the Method

The color analysis method works like this: it examines the hue, value, and chroma of the coloring in your skin, eyes, and hair. Then, it helps you identify your seasonal color palette (summer, spring, autumn, or winter) and your dominant (light, bright, warm, cool, muted), which relates to the combination of your features.
The results are categorized as seasons to simplify understanding, since we already tend to associate certain colors with specific seasons. For example, white and cool blues for winter, and muted orange and red for autumn.

Figure 1.2a and Figure 1.2b show respectively “Bright Spring” featuers and ”Wow colors”.

(a) Features

(b) Wow colors

Figure 1.2: Bright Spring Season

1.3 Premise

If the color analysis is really worth it or if it really works is subjective. But it is a fact that it got a lot of hype on Social Media and a lot of people spend money rather online or in person for determining their "Wow colors".

We, Daniel and Annika, in this project made a simplified version of the Color Analysis based on the document of the "Style Coaching Institute". Our program takes a portrait picture as an input, and determines the "season" of that person by scanning the colors of the hair, eyes and skin.

Chapter 2

Method

2.1 Structure

We structured our program in three main parts:

1. Face Feature Detection: in this part the input portrait gets processed. With the help of a predefined "Face Landmark" library by Dlib, the face gets recognized and landmarks are getting placed. Then the function uses these landmarks to get the average eye and skin colors. In the end we used a hair segmentation tool by Google Mediapipe to extract the average hair color.

2. Getting the dataset for the "seasons": For this step we used the document, mentioned before and extracted the train colors for the eyes, hair and skin that are going to be used in the last step.

3. Matching "season" with the input portrait: This function calculates the closest values for hair, eyes and skin for every season and then calculates which season has the lowest sum of these distance values. The function then returns our final result: the closest colors found to the hair, eyes and skin color, the "season" of the person on the portrait and the "Wow" colors corresponding to the season.

2.2 Face Feature Detection

2.2.1 Generating Landmarks

Our main goal for the first part was to write a function that, receiving a portrait image of a person as input, returns the persons eyes, skin and hair colors as tuples of RGB values, as precisely as possible. For this, we worked with a pre-trained model which provides a facial landmarks detector, in the DLIB library. It estimates the location of 68 (x, y)-coordinates that map to facial structures on a face. Detecting facial landmarks is a two step process:

1. localize the face in the image;
2. detecting the key facial structures on the face.

(a) Input image (b) Landmarked image

Figure 2.1: Example of face feature extraction

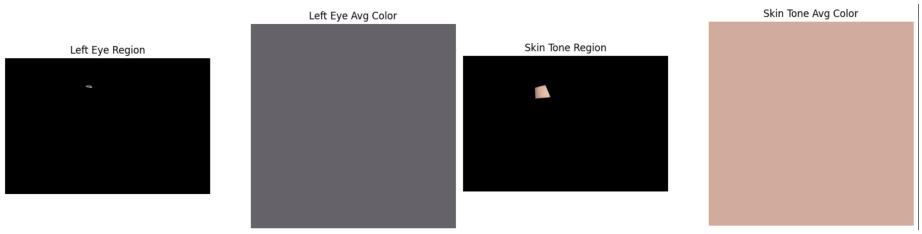
2.2.2 Color extraction

Let's start from the eyes color. First step is taking the landmarks that surround the left eye. After that we create a mask out of the region of interest (ROI) delimited by the landmarks, and apply it to the original image. Lastly we use K-Means algorithm on the masked image to get the average color of the iris. Since the sclera and pupil do not contribute to the eye color, we performed K-Means with 3 centroids: one for whitish details, one for black and one for the characteristic color of the eye. In the end the color closer to black, and the color closer to white are dropped, and only the iris color is kept.

For skin tone the process is similar. We take an area right under the left eye, with the help of landmarks placed on the nose, on the side of the face and under the eye. We apply to the original image the mask we get out from the new ROI, and use k-Means to get the average skin color. This time 2 centroids are enough: one for the black part of the masked image and one for the ROI.

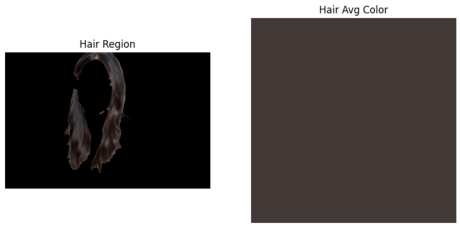
For the hair, the task could not be solved that easily (we tried a landmarks-based approach and also simple masks, unsuccessfully), so we opted for another pre-trained model: Google Mediapipe Image Segmenter. Their Image Segmenter makes it possible to divide images into regions based on predefined categories, in our case the hair. It locates the hair on the head, and outputs an image segmentation map for their hair. And from there we can use K-Means again to compute the hair average color.

Note: we talk about "average color" since it is the easiest way to summarize the complexity of face features. Also, we assume the eyes of the person in the input image are the same color.



(a) Avg. eyes color

(b) Avg. skin color



(c) Avg. hair color

Figure 2.2: Color extraction outputs

2.3 The Dataset

In order to compare the extracted data of the input picture we need a dataset that defines hair, skin and eye color for every season. For this we used the "Style Coaching Institute" definition. In the document there are several different colors for each season and kind of face feature [Fig. 2.3] and our goal was to have a dataset with RGB values for each season divided in categories: skin, hair, eye color and "wow colors".

(a) Eye sample (b) Hair sample (c) Palette of skin tones

Figure 2.3: Samples provided by "*Style Coaching Institute*"

We created three functions that aim to extract the colors of the eyes and hair in a similar way that we do it in our function that extracts the colors for the portrait image, so that in the end we get as accurate results possible.

Starting from the eyes, we took the sample eyes [Fig. 2.3a] of the document and created a collection out of them, naming them the same as in the document, to not create confusion. For each eye sample, the region of interest is isolated from the image, by converting the picture to grayscale, applying blurring to reduce noise, using edge detection and finally performing circle detection to mask the iris and the pupil. Then K-Means clustering is applied to the masked eye region, to identify the dominant colors. Since black details are prevalent in eye images, the "outlier colors" are discarded in order to focus on the actual eye color. In the end, a dictionary gets created, with the name of the color as a key and the corresponding RGB tuple as value. Unfortunately the algorithm could not recognize all of the colors perfectly and we deleted the ones that produced results very different from what we expected.

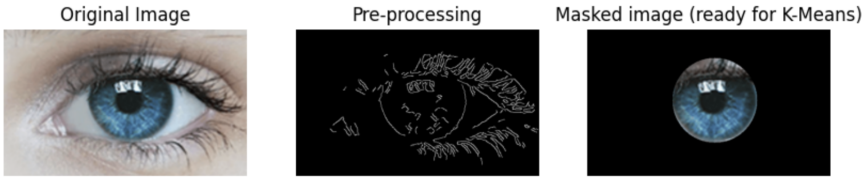


Figure 2.4: Extracting eye color from the sample images

For the hair colors [Fig. 2.3b] we did a similar thing. The region of interest was the entire photo of the hair so it was sufficient to use K-Means clustering to get the average color of the hair. Also in this case we saved our results into a hair dictionary with the name of the hair color as a key and the RGB tuple as a value.

The skin colors were already given in a palette format [Fig. 2.3b] but sometimes there were fine white lines separating similar colors so we created a function that returns the n most frequent colors in the color palette and saves them to a list. (n chosen based on the number of colors in the palette).

In order to create the final dataset, we created a dictionary with the season names as keys and another dictionary as value. This "value dictionary" contains

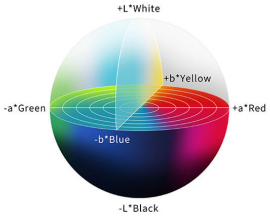
skin, eyes and hair palettes for the corresponding season. In the end using the pandas library we transpose the dictionary to a dataframe with seasons as rows and skin, hair and eyes as columns. Lastly, a fourth column is added, containing the "Wow colors" of each season.
With this data frame we can move on to the final step of our project: the matching of extracted values from input image with one of the seasons in the created dataframe.

2.4 Season Matching

Our program is able to associate the extracted colors from the input image to one of the seasons thanks to a "Match Score". More precisely, a "Match Score" is computed for each season. The matching season to our input will be the one with the lowest score. How is the "Match Score" computed?

The program loops through every season and uses a function to calculate the differences between the extracted eyes, skin, hair colors and all possible eyes, skin, hair colors of the season. We quickly found out, that simply taking the Euclidean distance of two RGB tuples would not lead us to satisfying results since the color space is not uniform. This is where we found out about the CIE L*a*b color space [Fig.2.5a] that is designed for the specific purpose of measuring the difference/similarity between colors.

The benefit of working with this color format is, that it follows the standard of CIE (International Commission of Illumination) which defined a formula to calculate the distance between two L*a*b colors [Fig.2.5b]. For this, there is a function in the skimage module called "deltaE-ciede2000" which will calculate our desired similarity for our colors. So for every season we calculate the closest skin, hair and eye colors to the input portrait image. We defined a "match score" for every season, which is the sum of the differences of the closest features to the portrait. The season corresponding to the lowest "match score", will be the matching season to our input image. In the end of the program, we return the season's name and "Wow colors" and the closest colors found to our skin, hair and eyes colors, so we can see how accurate the match is.



(a) LAB Colorspace

$$\Delta E_{00}^{*} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \frac{\Delta C'}{k_C S_C} \frac{\Delta H'}{k_H S_H}}$$

(b) CIELAB ΔE

Chapter 3

Conclusion

3.1 Results

Unfortunately with the "season" classification there is no scientific way to prove, one's matching season is actually the one the program outputs. We mainly focused on how accurate the face feature extraction works and how well it recognizes different skin, hair and eye colors. Here are some results:

(a) Matching

(b) Result

(a) Matching

(b) Result

3.2 Limitations and Problems

1. **Lightning and bad quality of the picture**: one of the biggest limitations of the project is that we can say that in bad lightning conditions or bad camera quality, the color detection will not be very precise. For example, if there is a dark shadow on the person's face, the program will perceive the hair/eye/skin color, to be a lot darker than it actually is and therefore extract wrong values for the features. Same is with bad picture quality. It is harder to extract an accurate color if the picture is blurred for example.

2. . **Faces are different**: We didn't consider, for example the cases, where it is not possible to extract a hair color because the person in the picture is bald or if a person has major skin imperfections (strong acne for example). Our model cannot make these considerations, therefore we cannot guarantee an accurate feature extraction in such scenarios.

3. **Dataset**: While trying to extract the eyes and hair colors of the samples given in the document, in some cases our function did not recognize the region of interest correctly [Fig.3.3] and therefore gave us the wrong color. This happened to us a lot with any variation of the eye color "hazel". As a solution we exchanged these colors with similar colors that we already retrieved correctly or deleted the color completely if there was already a similar color inside the according list.

Figure 3.3: Bad recognition of the pupil

4. As mentioned before the “color analysis” is no scientifically proven method. Yes, it is based on color values and color toning, and finding the closest, so a bit of math is always part of it, but which colors suit one good or not in the end is a subjective perception. The “color analysis” can be a fun tool to experiment and to even discover new colors for yourself. It should be seen as no rule but more an inspiration to try out something new.

We personally would not pay for an expensive professional color analysis, if it is that easy (understatement) to just create your own AI that does a perfect (exaggeration) analysis.

Chapter 4

References

1. **Style Coaching Institute**: link
2. **Color Analysis Cheat-sheet**: link
3. **DLIB 68 facial landmarks**: link
4. **Google Mediapipe Image Segmenter**: link
5. **CIELAB ΔE formula**: link