# Color Palettes: Pattern Recognition and Classification of Images

## Introduction

The introduction is a crucial part of your report and sets the stage as well as motivates the work you have completed. It briefly introduces the problem, emphasizes its importance and shows to the reader what you did in your work. That is, you show the reader that your problem is fundamental and that it needs to be solved as it fills a gap in the known literature and prior work.

The introduction starts by addressing the problem in general and points the reader in the direction how you want to solve your problem / research question. As a metaphor, John Swales developed a three-stage model for research introductions:

[Move 1:] Establish a territory (claim centrality of topic)

[Move 2:] Establish a niche (indicate gap)

[move 3:] Occupy the niche (outline purpose and indicate research structure/methods).

You may follow that model while writing the introduction. In the first part of the introduction or in a separate section, you are expected to mention and organize supporting literature where you outline the {\em state of the art} of your research question.

At the end of the introduction you may include a short summary of the structure of the rest of the paper. The introduction together with the conclusion should give a complete short version of your work and what you have achieved.

## Related work

After the introduction there follows the related work section. Try to classify the related methods in the literature with respect to the problems and solutions in your work. Establish the position of your work with respect to those competing on the same turf. Summarize the core contributions of the most important related prior solutions within the context of your work: if different from yours, clearly indicate the limitations that your solution improves upon; if similar to yours, state its advantage over other methods and thus indirectly justify your method. At the end, it should be clear why a new solution is required to solve the same problem and how your method is different from other related work.

You should know your literature well in order to find the missing link, or the most important unsolved problem, understand your contribution to the field, place your contributions in the midst of other works, and to sell your work.

Select references carefully and organize them into related groups. Within one topic you can organize references from old (first seminal) paper to newest achievements, and if too many related solutions exist, restrict to the last few most important papers if necessary.

## Problem statement

The technical sections introduce and motivate the proposed solution in the light of the problem, which requires a precise problem statement together with any assumptions and requirements. Where algorithmic or mathematical descriptions are not appropriate, other technical and implementation problems can be stated that are to be solved in this work.

## Technical solution

One or more sections should be directed towards the detailed description of the proposed solution, including technical details about the used data structures, algorithms and mathematical methods. The technical description should allow the resourceful and interested reader to reproduce and verify your work, together with the implementation information given in a later section.

This is the most important part of the report that should answer every little technical question that arises in the reader's mind. Your algorithm might be a puzzle with many pieces which are described in a linear order in the report -- and many such orders may be possible. A good order to describe the different components of your solution is one that allows to clearly explain one component after another based exclusively on what the reader has already seen in any previous sections, thus minimizing any forward references.

This core part of the report should be organized into coherent subsections, giving an overview and introducing formalism first. Following an overview of the necessary steps of the entire method, each step can then be described elaborately in each subsection.

Note that implementation details should be avoided as much as possible, and the focus should be on the formal and algorithmic solution; the implementation section is specifically targeted to explain programming details.

## Implementation

The implementation section focuses on the programming problems and details such as the organization of the source code, the dependencies of the different modules etc.

## Experimental results

The evaluation discusses the proposed and competing solutions in the light of the initially stated problem requirements and limitations. This typically involves some sort of experimental evaluation which leads to some type of qualitative or quantitative results.

Quantitative results include observed numbers indicating performance timings (speed) or accuracy measures of the given implementation and test datasets. If possible, statistical tests and analysis should be given, or where applicable formal proofs. Meaningful and informative numerical results must be complete and unambiguous. Explain in detail how the evaluation has been designed, as well as the experimental setup and test cases. This includes accurate description of the test data (type, properties, size etc.) as well as the test setup (e.g. view settings, screen resolution etc.) and the measured variables (frame rate, throughput, accuracy etc.).

Qualitative results may be reported if clear quantitative measures are not feasible or applicable. Qualitative results clearly show the features and functionality of the completed work, indicating if and how they are novel or different from prior work. Qualitative results are especially suitable if something {\em new} has been achieved that no-one has done before in the same way.

Essentially, the goal of the experimental results is to convince the reader by numbers, tests and images (and maybe user studies), giving some sort of proof why the proposed solution is good, different and/or better than other solutions.

## Discussion

Puts the results in perspective, discussing it in relationship to other related work. Indicate possible (side-)effects and eventual limitations due to the evaluation. State the {\em take home message} of the paper that the reader should remember and provide an outlook on possible future work that extends the given solution or fixes specific limitations. Close with a brief description (that is different from the Abstract) of the proposed solution.

Summarize your main findings in one or maximum two pages. Try to keep yourself short and clear. Give a short discussion about your results where you focus on what your findings mean. E.g., show how your results and interpretations agree with the original question and with other published work or if there are any possible practical applications for your work. At the end, give hints on further improvements or development directions / areas.

## Conclusion

##################################################################################

**Tasks’ workflow**

**1. Obtain a dictionary of colors → en, en extended by google, en extended the user (LAB space)**

**2. Classification (KNN) → hard borders, soft borders  (LAB space)**

**3. Visualization of the results**

**4. Query**

This research project lies at the intersection of computer science (particularly artificial intelligence) and the arts (particularly film and color theory). My contributions have been to design efficient mechanisms for color palette search and comparison (image processing) and to leverage machine learning and search methods in application to web tool design. I developed innovative computer vision / machine learning systems in the area of visual search. I’m involved in data analysis, prototyping, testing and deployment.

Isaac Newton

Johannes Itten

Goethe

Munsell

Color gamut = mathematical definition of color in a color space

<https://www.december.com/html/spec/colorlinks.html>

<https://people.csail.mit.edu/jaffer/Color/>

The project consists of the following main tasks:

**Task 1: Given a hierarchically structured color palette, classify the contained colors such that a palette is searchable using categorical names.**

The FilmColors project’s Visual Annotation webtool (VIAN) predefines categorical color names such as “copper”, “orange” and “mustard” that can be used for the search query. A function in VIAN is able to extract a color palette from an image in hierarchical order, i.e. get n-most salient color patches at the lowest image level and hierarchically merge them together in a tree until the highest level is reached with only one color that is the average color of the image. The task is to make the colors in these color palettes searchable based on the color name options in a search query.

TODO: Regarding a query for "red" also retrieving "ocra", "blood-red" etc. this is an important feature, but also one should be able to only retrieve the typical "red" when searching for it specifically. ??? User needs to be able to search ‘ocra’ not in VIAN colors, too… ?

Categorical color dictionaries: EPFL Color Thesaurus, others … , (allow user to select dictionary)

He could query “ocra” (not VIAN color): he gets only ocra – NOT DONE

He could query “red” (not VIAN color): he gets only red – NOT DONE

He could query “red” (VIAN color): he gets red, ocra, blood red - DONE

Query category “red” or exactly color “red”

TODO: accuracy distribution of “red” (probability distribution) of many values

English world color culture only, but any other dictionary (Japanese) with name, language and value should be parsable for any dictionary for later use

Color space: LAB only

Give user he option to decide between hard and soft color border

Distance: Euclidean only because LAB is perceptually uniform, search justification for this

DESIRABLE: color wheel picker, searchable by color value, 3d Voronoi tessellation (knn k=1)

1. Input: Color name

On VIAN, the user should be able to search for a VIAN color name and get color palettes back that contain that color name. The VIAN colors are in alphabetical order: amber, apricot, beige, black, blue, bronze, brown, copper, coral, cream, cyan, gold, green, grey, lavender, magenta, mustard, orange, peach, pink, purple, red, rust, sepia, silver, ultramarine, white, and yellow.[[1]](#footnote-1) An extension to this are the 950 English color names used in the Color Survey - or XKCD study - on color names.[[2]](#footnote-2)

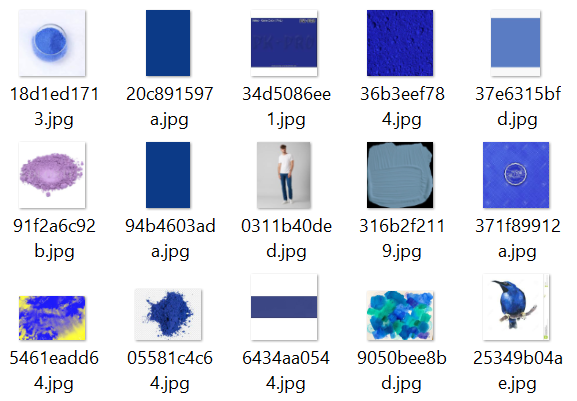
For these, we need to define the color value of the VIAN colors in some chosen color space. The chosen color space is CIE-Lab. Because it is larger than all other color spaces, users can search more specifically about a color. CIE-Lab values of colors can contain more information about color.

Albrecht Lindner et al. (2012) at EPFL have created a Color Thesaurus found on <https://colorthesaurus.epfl.ch/> which translate a color value for a color name in a given language into another color value in another language for the same color name and displays the respective color and closest neighboring colors in a web tool. Color names can be searched in 9 different languages – Chinese, French, German, Italian, Japanese, Korean, Portuguese, Russian, and Spanish to yield different color values in one language compared to another for each translation-equivalent color name. Neighboring color names are given when tweaking the parameter on the HSV channels as well as an sRGB, RGB HEX and CIE-Lab color value. A color wheel picker helps the user to determine the right color.

The original data set has 6’167 rows of colors in 9 different languages, their English names, sRGB and CIE-Lab color values. For example, for “baby blue” a discrete color value in sRGB or CIE-Lab is returned. These color names can be categorized into VIAN colors. Thus, we process this csv-data set in Excel by removing all non-English color names. There remain 720 color instances. We add a new column “VIAN color category” to the data set and hand-categorize all color names from the Color Thesaurus into one of the VIAN colors. For two-named colors such as “purple blue”, the color was categorized into “blue”. In the more obvious case, such as “apple green” the color was categorized into “green”. All colors that were not derivable by a second word match or identical match were hand-categorized into VIAN colors (Python script). We add two additional columns “srgb” and “lab” by concatenating “r”,”g”,”b” and “l”,”a”,”b”.

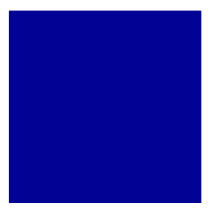
Except for the VIAN color “ultramarine” all VIAN colors were classifiable from the Color Thesaurus’ list of colors. Some VIAN colors can be mapped to several Thesaurus color names, while others have at least one color name translation. This is why we need an additional approach to define the ultramarine color values. To make things comparable, we follow in the line of the Color Thesaurus project and use a home-bread script to login to Google Image Search to fetch the first one hundred images for color name “ultramarine” in English. The search query as described by Lindner et al.[[3]](#footnote-3) was the “color name” and the word “color” in English.[[4]](#footnote-4) For “ultramarine” this would mean to key in “ultramarine color” in the search bar. Below are sample images for “ultramarine”. It can be adapted for any other color, if the set of VIAN colors should ever be adapted to include new color names.

First 15 out of 100 first Google Image Search Results for Color “ultramarine”

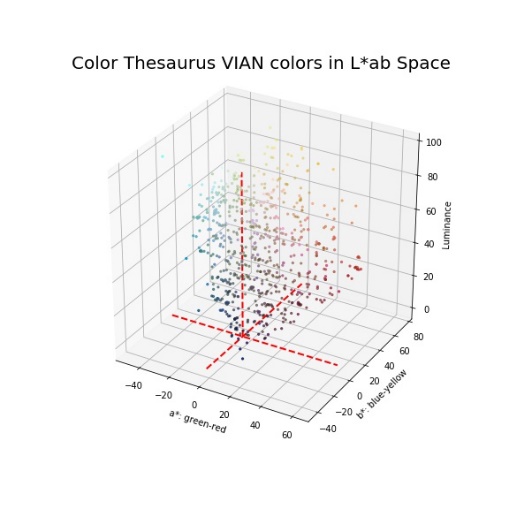
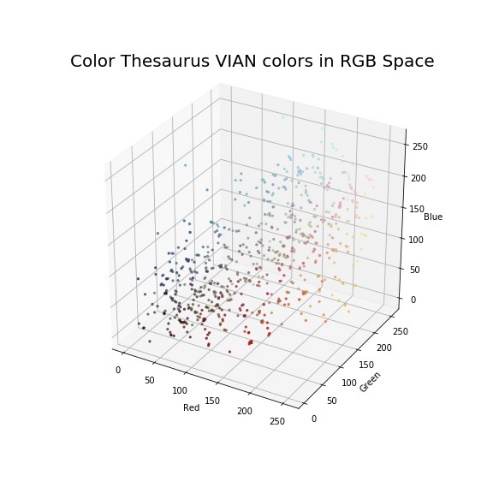


We crop the image where possible to extract the center image – this makes sense, because most of the images have white surroundings. When calculating the average color across all images, the white would tint the ultramarine into a diluted color value – a bias that needs to be avoided. As a result, we get ultramarine’s RGB color value: (2,3,150).

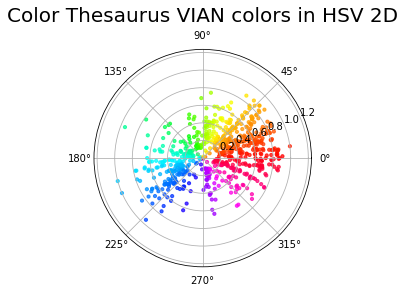
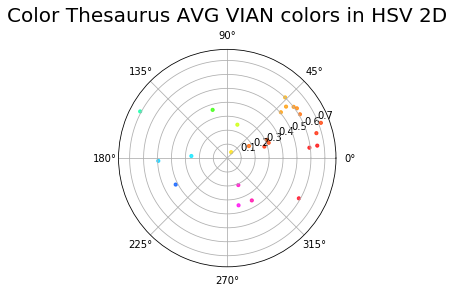
Ultramarine – RGB(2,3,150)



TODO: compute the mean of the colors in lab not in rgb

Next, we plot all English VIAN colors for equal values in the Color Thesaurus in RGB, HSV and Lab space:

In HSV space the colors are situated on a hue circle because the HSV cone has a circular form at the top. Plotting them in 2D can give a better overview of color hues in the VIAN colors. To the left all Color Thesaurus values are plotted for all VIAN colors while to the right the Color Thesaurus values are averaged to yield a unique value per VIAN color.



The VIAN colors are biased towards orange and red color tones on the color wheel. These are predominantly warm colors made available for the user to search. If we take the averages of all CIE-Lab VIAN color values for a VIAN color category and plot the resulting color class centers, we find mostly earthy warm color hues in the color scheme.

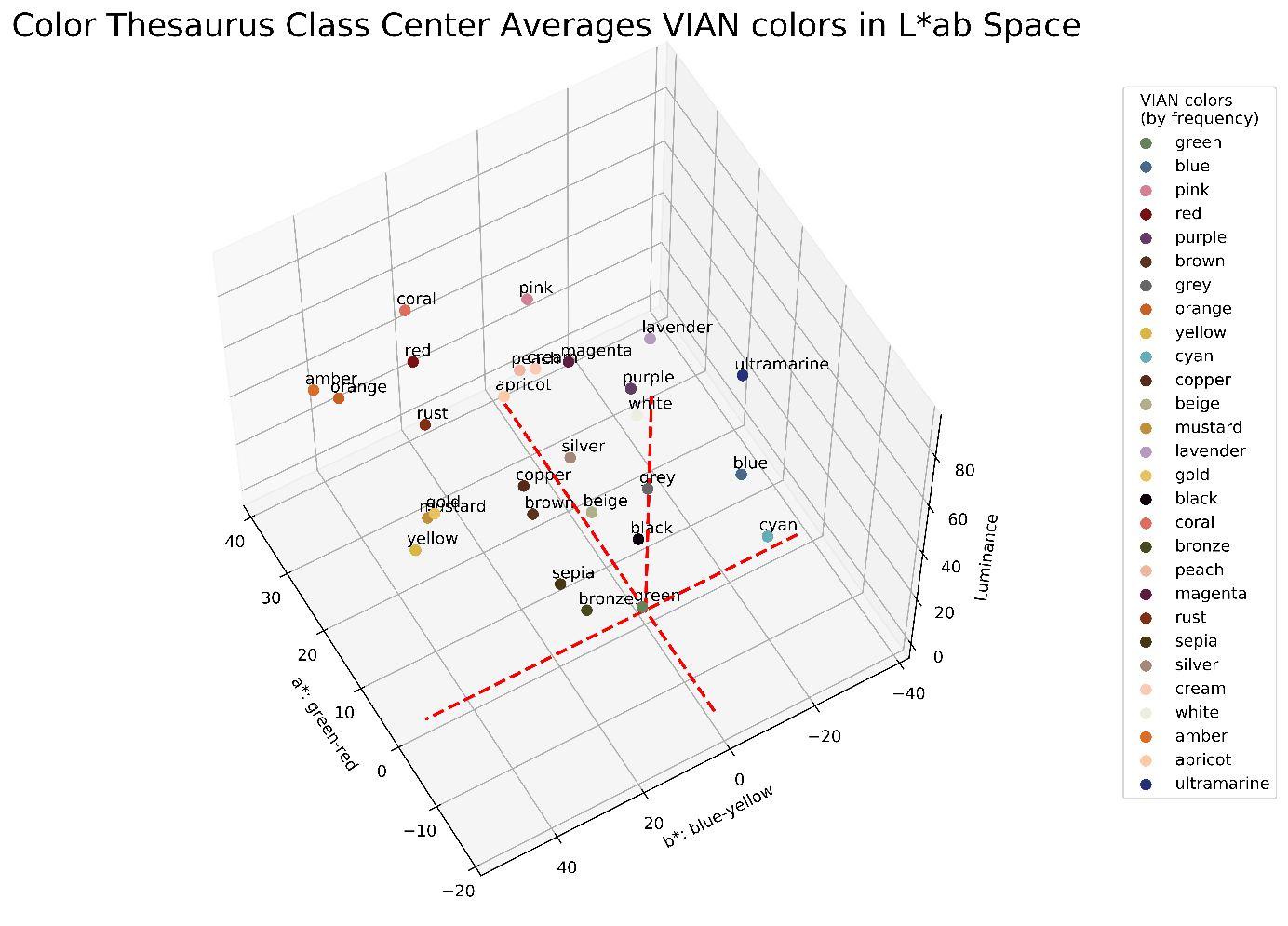
Taking the CIE-Lab average of all color thesaurus values for a VIAN color category, we get a unique value for all 28 VIAN colors. We sort them by their luminance value from dark to light.

Color Thesaurus Class Center Averages VIAN Colors – Color Palette



Some colors like sepia, bronze or copper and brown or yellow and gold or peach, apricot and cream are very close to each other – almost indistinguishable to the human eye. Thus, it is advised to extend the VIAN colors to another set of colors (TODO: WHICH? Include basic colors) so that the user has a larger variety of colors to choose from for the search query.

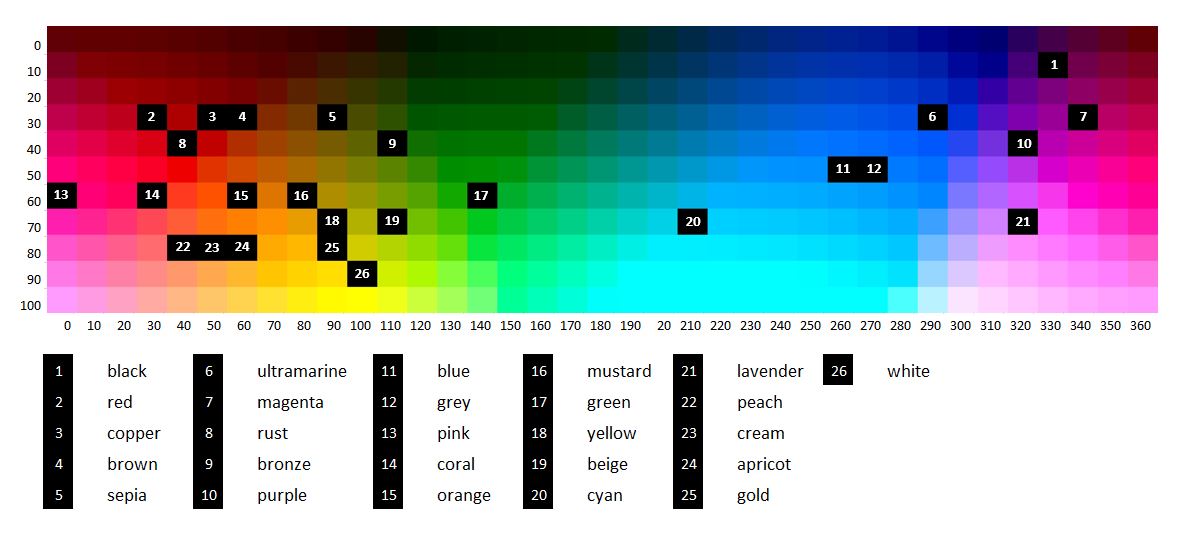
We plot these VIAN colors in CIE-Lab space as well, to get a better idea of their position in 3D.



TODO: increase font size

If we project these VIAN color points to a constant chroma of 100, that is the highest saturation for a color, convert the lab colors to lch, and round the lch values off to the nearest 10, colors orange, amber and silver become identical in value. Since orange seems to be a good basic color, orange was left in the data. The lch color space is divided into 10 luminance levels ranging from 0 (dark) to 100 (light) and 36 hue angles of 10°-steps around the color wheel.

Color Thesaurus Class Center Averages VIAN colors in LCH 2D Space



The VIAN colors in lch space are heavily clustered around warm, earthy color tones, i.e. browns, reds, oranges, and yellows. This is best seen in peach, cream and apricot – VIAN colors that follow each other closely. Only nine out of 26 different VIAN colors (34.6%) include cold, clear color tones, i.e. greens, blues, purples, and magentas. Thus, it is strongly advised to balance off the predetermined set of VIAN colors to incorporate a more equal distribution across the lch color space.

Next, we partition this space into regions where each VIAN color forms the center of the Color Thesaurus color values. For a given new color value it should be possible to determine which VIAN color it belongs to by looking at the region it is situated in. The decision boundaries between each VIAN color center helps to delineate these regions.

A short word about decision boundaries: at the border between yellow and green, it is typically difficult to decide to which the yellow-green color belongs. Either, the user is forced to make a decision where the border is situated between the most greenish yellow color and the most yellowish green color. This hard decision boundary needs to be fixed. But in the process of fixing, the border itself is a color calculated of these two colors, which leave the user to decide about the color class again to infinite precision. Another approach would be to allow for single colors at the border to be classified into both yellow and green. This could be extended to a bandwidth of colors at the border, classified to both yellow and green. This multi-label approach would require a multi-label classification during the supervised learning process of classification.

TODO: provide the user with both alternatives, clear-cut border or soft border (Hierarchical – interconnected)

TODO: write about that lab colors can only be seen as rgb, lab colors can also be outside of the visible light, opencv clamps all the lab colors into rgb colors

TODO: sample rgb color space, then convert them into lab and then classify it there in lab, convert back to rgb to display it

* Triangulation Approach

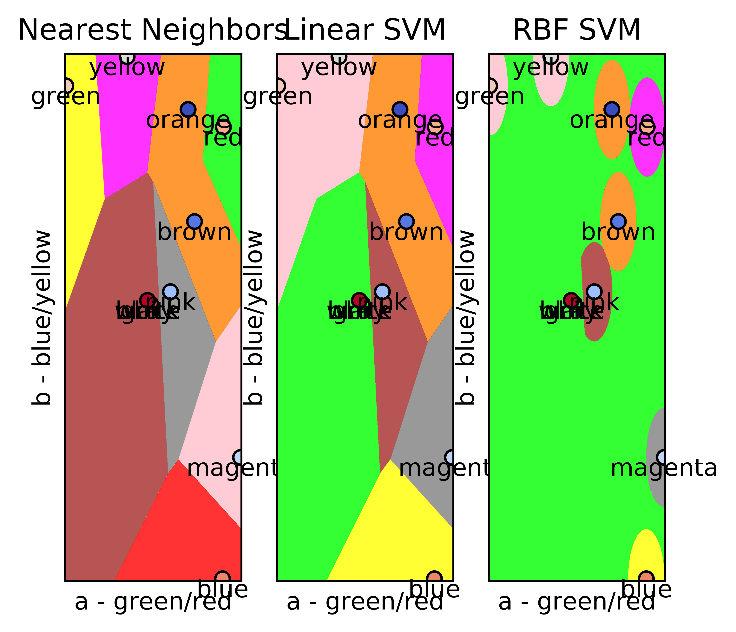
Taking two VIAN colors such as “cyan” and “blue” we take the perpendicular at the middle of the distance between these two points to define the decision boundary. This 3D Voronoi tessellation creates a decision boundary for each VIAN color.

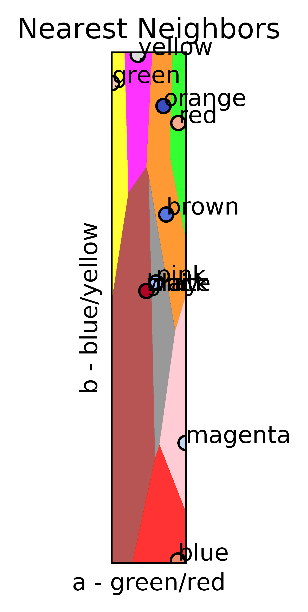
Knn= 1

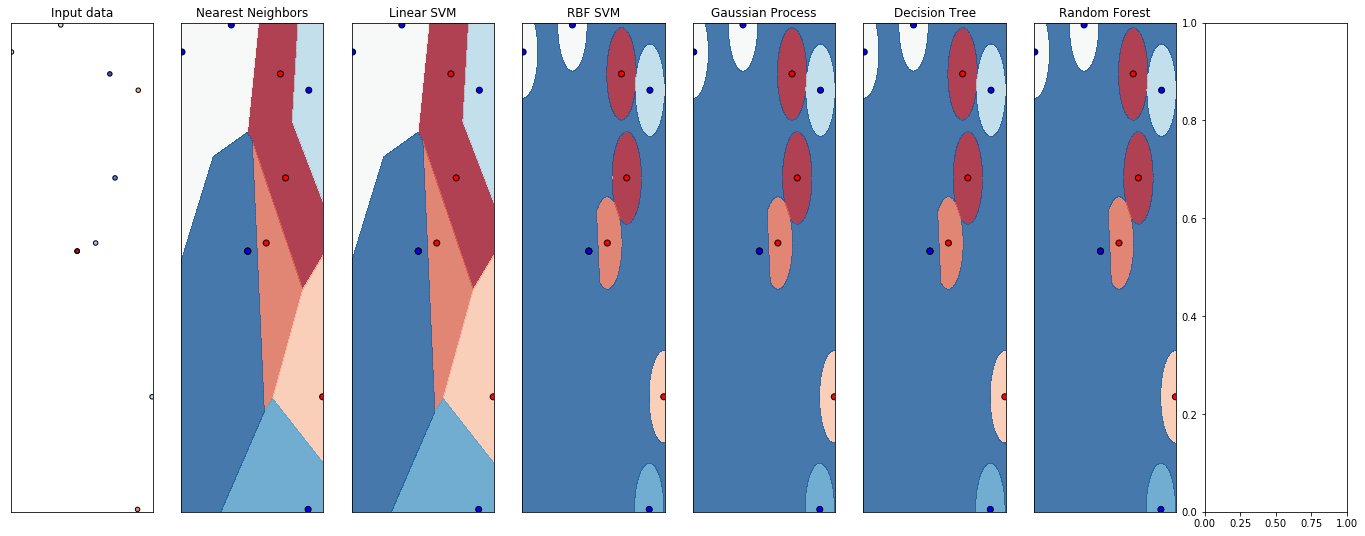
* Machine Learning Approach

We use a supervised machine learning approach for partitioning the LAB space into regions which allows for multiclass classification. (TODO: multilabel?) We choose the CIE-Lab space, because colors there are represented as perceptually uniform. From 10 machine learning classifiers, only two were yielding good results: K-Nearest Neighbor Classifier and Linear Support Vector Classifier.

The other classifiers which were tested for their usability were Gaussian Process, Decision Tree, Random Forest, AdaBoost, Naïve Bayes, Neural Net and QDA. The last two classifiers needed to be removed because there was only one sample case available for a color category (apricot) which is not enough data for these kinds of machine learning frameworks to work. The error would lie in the covariance being ill defined. The other classifiers simply made out a circular region among most VIAN colors, leaving out the larger space around the circles to a predominant VIAN color. This imbalance was the reason why all these classifiers were removed.

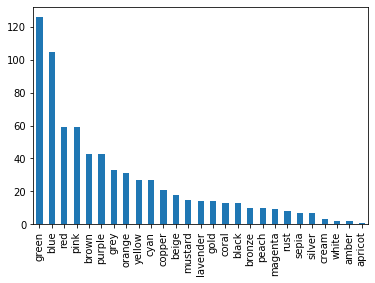
TODO: VIAN, not 11hues





We trained these two machine learning classifiers on the Color Thesaurus / VIAN dataset taking the CIE-Lab color values as feature and the 27 VIAN color category or all 720 Color Thesaurus color names as label. The VIAN color categories are very imbalanced – green has more than 120 discrete CIE-Lab values while apricot has only 1 value.

|  |  |
| --- | --- |
| VIAN color | Count |
| green | 126 |
| blue | 105 |
| red | 59 |
| pink | 59 |
| … | … |
| cream | 3 |
| white | 2 |
| amber | 2 |
| apricot | 1 |



Making the top and bottom of the list are green, blue, and red – the RGB colors – and white, amber and apricot. The latter colors were specific colors, while the former encompass a wider region of discrete color values.

For training the machine learning classifier, the entire dataset was used. For testing, we sample from the entire CIE-Lab space by taking testpoints from a grid of equal step size for each channel. For example, if 3 steps are chosen, the luminance channel ranging from 0-100 yields testpoints 0, 50 and 100. These CIE-Lab testpoints are visualized as color patches that are compared against the color patch of the classifier-predicted VIAN color label. This allows for a quick manual check whether the classification was correct or not. The resulting accuracy and error was improved over several iterations and parameter combinations. The top 3 results:

TODO: to redo with 28 vian colors with ultramarine

TODO: try other parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Training Data | ML Classifier | Parameters | Test Data | Accuracy / Error |
| 1 | Rows: 720 original data set + 216 testpoints = 936  Feature: CIE-Lab color values  Label: 27 VIAN colors | KNN | Number of nearest neighbors = 5,  P=2 (Euclidean) | Feature: 27 CIE-Lab color values  Predicted Label: classified 27 VIAN colors  Real Label: manually-labelled (training data) | 100% / 0% |
| 2 | Rows: 720 original data set + 1331 testpoints = 2051  Feature: CIE-Lab color values  Label: 27 VIAN colors | KNN | Number of nearest neighbors = 5,  P=2 (Euclidean) | Feature: 125 CIE-Lab color values  Predicted Label: classified 27 VIAN colors  Real Label: manually-labelled (training data) | 96.8% / 3.2% |
| 3 | Rows: 720 original data set + 1331 testpoints = 2051  Feature: CIE-Lab color values  Label: 27 VIAN colors | KNN | Number of nearest neighbors = 5,  P=2 (Euclidean) | Feature: 27 CIE-Lab color values  Predicted Label: classified 27 VIAN colors  Real Label: manually-labelled (training data) | 96.3% / 3.7% |

1. The K-Nearest Neighbor classifier is a lazy learning algorithm that computes for each new test point its classification by taking a simple majority vote of its labeled k nearest neighbors where k is the number of neighbors it checks and p is the distance metric calculated between the new test point and the nearest training point neighbor. The algorithm is robust to noisy training data. But, k and p need to be determined and computation cost could be high compared to other machine learning classifiers.
2. The Support Vector classifier separates the training data cloud linearly based on their clusters in space. By computing a hyperplane between the clusters, a decision boundary is created that makes classification of new points into each region possible. To calculate the hyperplane, it maximizes the margin between the two support vectors delineating two adjacent clusters of points. Each cluster belongs to another class.

Confusion matrix:

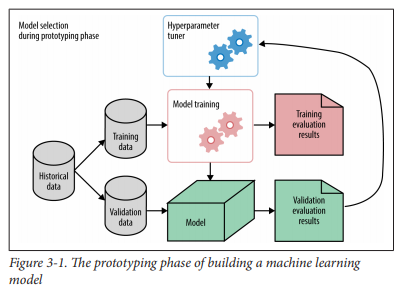
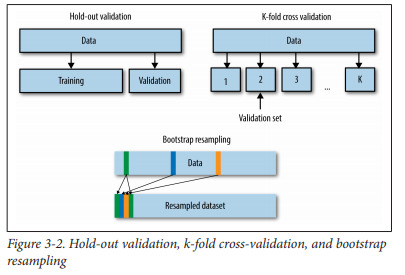
|  |  |  |
| --- | --- | --- |
| **Real Value** | **Predicted Value** | **Accuracy** |
| Green | Not green | 0 |
| Green | Green | 1 |

TODO: F-1 score /ROC

TODO: Performance timeit of both approaches

TODO: try other color space

We select the best model using validation results.

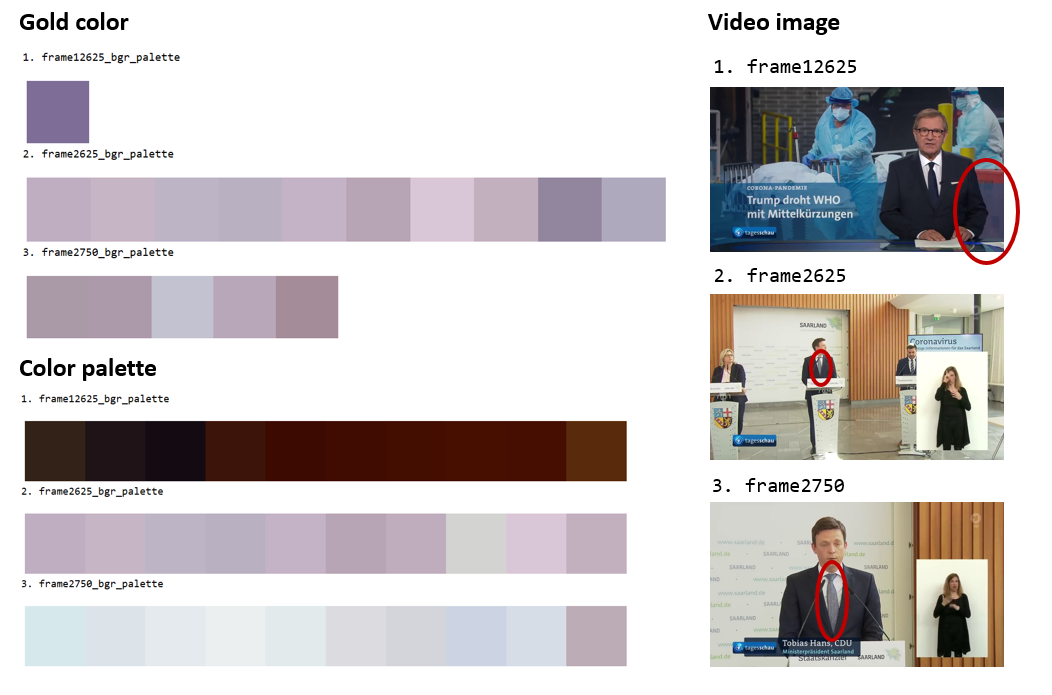
 

The user searches “sepia”, find a match of that with color palette’s colors

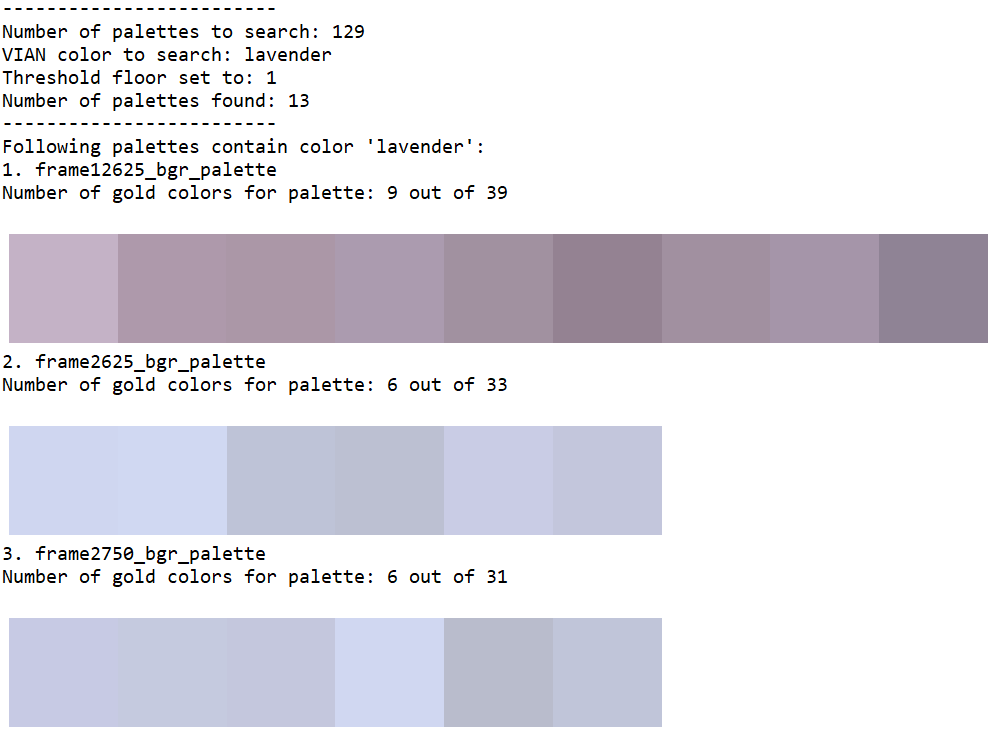
1. Take all colors inside color palette, run it through ML model to get classification of colors into VIAN color categories, if one of them is also “sepia”, color space: HSV
2. Take “sepia” color values range, take color values of each color in color palette and see whether it is within the color values range of “sepia”, if yes, display this color palette in, each palette’s colors has a VIAN color tag that is the result of the machine learning prediction, color space = LAB

For sample color palettes, we subsample a video footage of “Die Tagesschau” into images from which one color palette per image is extracted for all video frames to get a pool of color palettes. The extraction is hierarchical – the lowest row contains the highest amount of different colors for a video image. The user can set which row to fix, we fix it to the lowest row ‘row 20’ for now. In general, it is better to choose a low level of hierarchy for a color palette, because for the highest level only one color will be available for an image. Then, all BGR colors of the lowest row are converted into LAB. Then, they are classified into one out of 28 VIAN colors using above mentioned top-rated machine learning classifier KNN (n\_neighbors = 5, p=2) which was trained on 2051 test data. The user’s VIAN color search key is matched to the predicted VIAN colors of each palette. The number and names of all successfully matched color palettes (gold palettes) are shown as a result of the query. (The gold color palettes first n-to-all color patches can be displayed. )

For example, the user can search for VIAN color ‘lavender’ in 129 color palettes. If one of these color palettes contain ‘lavender’ the color palette will be shown to the user. The number of palettes that were found containing the color ‘lavender’ are 33. The first three search results are displayed in the following:

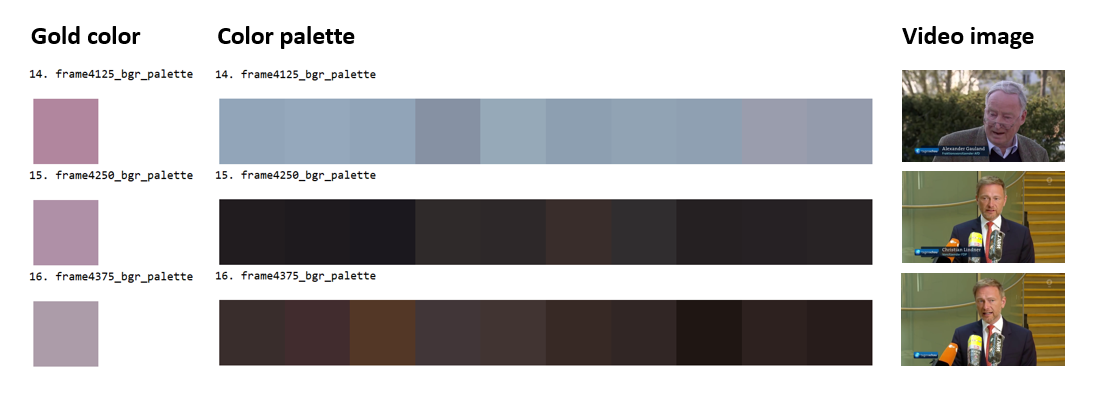
TODO: sort color palette by ratio\_width

Adding to this, the user can set a threshold floor for the search key. Because ‘lavender’ in an image takes up only a certain percentage of the total image sometimes it is not useful to the user to get all images where ‘lavender’ makes up only a small portion in the image. Setting a threshold will help the user to sift such images out of the result. The user would have then all color palettes for images where ‘lavender’ is present with some dominance. At the lowest level of an image’s color palette, all 101 colors have a certain ratio width, typically, that fluctuates around 1 percent from the total width. Fixing the threshold to 1%, we get only 13 color palettes back for ‘lavender’.



The first 13 color palettes with a threshold floor of 0 or 1 are the same. The remaining 20 color palettes for a threshold of 0 have significantly less ‘lavender’ in the original video image, because these ‘lavender’ colors make up less than 1 % of the image. In the below image, the next 14 up gold colors for ‘lavender’ are shown with their original color palette (10 out of 101 color patches) and the color palette’s original image.

TODO: sort color palettes (showing 10 colors out of all colors in color palette)



Python Scripts:

1. ColorClassCenter.py
2. ColorClassCenterVisualization.py
3. ColorPaletteSearchColor.py
4. ColorMLClassification.py
5. ColorConversion.py
6. ColorMLClassificationVisualization.py
7. ColorPaletteExtraction.py
8. ColorPaletteInHSVSpace.py
9. ColorPaletteSort.py
10. ColorPaletteVisualization.py
11. ColorSpaceVisualization.py
12. ColorThesaurusPreprocessing.py
13. ColorThesaurus2VIANClassification.py
14. ColorThesaurusVIANInSpace.py
15. ColorVisualization.py
16. ImageDownloadGoogle.py
17. Images2AVGColor.py
18. MovieToImage.py

**Task 2: Implement a method to compute the distance between hierarchically structured color palettes**

The FilmColors project’s Visual Annotation webtool (VIAN) features a function that is able to extract a color palette from an image in hierarchical order. At the lowest level are the most colors extracted for the color palette. At the highest level all colors were merged to form a single color average for the image. The task is to get one such color palette and compare it to other color palettes based on their similarity to the first color palette. For the first color palette, the distance is calculated to all other color palettes to determine the top-k nearest color palettes.

Similarity metric

Distance Euclidean only in LAB space only

Display cluster of points for both cps in LAB space

TODO: Make histogram (color distribution, probability)

Y- axis: ratio width, x-axis : colors (sorted by ratio width) or put it in lab color space histogram a, b, l channels in 3d axis, and bin frequency, no need to see the histogram

Compare multidim histograms (opencv, cv2.comparehist() look for similarity metrics )

Color names in the histogram close to each other are also closer together in another color palette histogram

Preprocessing

Each color palette comes with information about its colors: the bgr values of each color and the ratio width of each color. First, we extend this information about colors by converting the bgr values to lab values for each color and predicting the categorical color names for each color using the top-rated machine learning model features in Task 1.

Method

1. Aggregate Comparison

On color palette level: aggregate individual bgr colors by averaging bgr to a CP metric and compare CP metrics with each other (highest row in hierarchy), summing, taking the closest (Min) or furthers (Max) can distort the results we are trying to get, image gets divided into superpixels and the superpixels are aggregated until only one superpixel is left

One image can only have one color palette of at most 101 colors. At the lowest level, all image’s color palette have at most 101 colors

TODO: Agglomerative clustering to show and skips to show

Display math formula – no!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CP1 | | CP2 | | CP1-CP2 |
| Color | Ratio\_  width | Color | Ratio\_  width | Distance (eucl, jaccard, manhattan, cosine etc.) |
| Bgr1 to Lab1 or catcol1 | Ratio\_width1 | Bgr1 to Lab1 or catcol1 | Ratio\_width1 | Dist(CP1.Lab1\* Ratio\_width1),(CP2.Lab1\* Ratio\_width1)) |
| Bgr2 to Lab2 or catcol2 | Ratio\_width2 | Bgr2 to Lab2 or catcol2 | Ratio\_width2 | Dist(CP1.Lab2\* Ratio\_width2),(CP2.Lab2\* Ratio\_width2)) |
|  |  |  |  | Argmin(Dist(CP1.LabID\*ratio\_widthID),(CP2.LabID\*ratio\_widthID)))  Min(Distance) |

1. Distributive Comparison

On bgr level: calculate all combinations of pairwise distances between all bgr colors in LAB space (due to perceptual uniformity in that space) and find minimum distance between cp1bgr1 and cp2bgr\*, make metric cp1-cp2, do with other cps, lowest metric pair is closest neighbor (lowest row in hierarchy), include weights for each color by looking at ratio\_width, at highest resolution has 101 colors

All color palettes are loaded into a pool of color palettes. To find the most similar color palettes inside a pool of color palettes, we convert all palette colors to lab values first. The LAB space is chosen because its advantage lies in the perceptual uniformity of color in that space. Then, we build color-palette pairs by calculating all pairwise combinations (with withdrawal) possible between the color palettes in the pool. Each color in one color palette is matched to a color from the other color palette in such a way to get the minimum distance. In this instance, the Euclidean distance metric is chosen. For a color-palette pair, we take the average of all minimum distances that is each color-palette pair’s bondage index. The smaller the bondage, the more similar the color-palette pair.

For 129 color palettes in the pool, the n-closest possible color-palette pair were found (through recursion).

Top-1: Palettes “frame1250”, “frame1375” are the closest to each other.

Top-2: Palettes “frame1125”, “frame1375” are 2. closest to each other.

Top-3: Palettes “frame1125”, “frame1250” are 3. closest to each other.

These color palettes’ original image frame is displayed in pairs to see why they are the closest to each other. While top-1 “frame1250” and “frame1375” are exactly the same, “frame1125”paired up with each of the top-1 image frame are almost the same: the number of deaths 1.861 is not displayed in the former frame image frame. The result is very convincing.

Top 1: Image frames “frame1250”, “frame1375”

Top 2: Image frames “frame1125”, “frame1375”



Top 3: Image frames “frame1125”, “frame1250”



The images were taken from the following video footage sequence:



For finding the top-10 most similar color palettes for given image frame “frame3500\_bgr\_palette” the asymmetry in the pairs needs to be considered when finding the result. Also each color in the color palette has an unequal importance distribution. The weights were set to be equal to the ratio\_width of each color.

The top-3 most similar color palettes are top-1: “frame3625\_bgr\_palette”, top-2: “frame3750\_bgr\_palette”, and top-3: “frame3375\_bgr\_palette”. Had the weights not been calculated, “frame3125\_bgr\_palette” would have figures at the 100th position instead of XX..

Distance metric = Euclidean, jaccard …

Python Scripts:

1. ColorPaletteDistCalc.py
2. Classify the patterns within the color palettes in different types of color contrasts defined by the ERC FilmColors Project

pattern = color contrast Ðiven a color palette (first: 5 rgb-valued patches, then scale it up), classify them into one of the color contrasts categories

1. Implement a web-based user interface that allows the user to select or create a palette and generate a list of similar color palettes

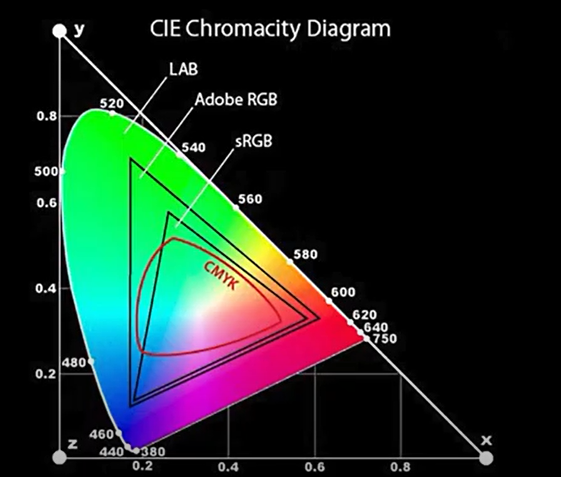
user defines number of patches to fill with colors that the user wants and specifies the proportion of them 0-100%, result: top-k nearest color palettes from a given list of CPs

**Appendix**

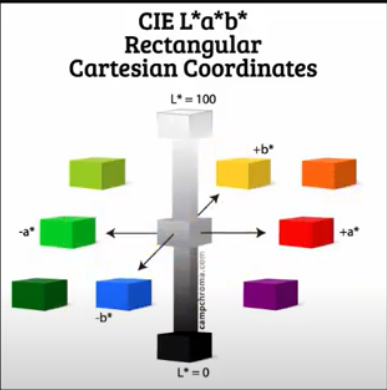
Color Basics

A color is just a reflection of different wavelengths. The visible lights is between 390 and 450 nanometers. We perceive a ray of light through the human visual system.

Color Spaces

All these different color gamuts – LAB, (English world: Adobe RGB or European world: ICE RGB), sRGB and CMYK have different uses.

CMYK has three primary colors: cyan, magenta and yellow. CMYK is used at your printer.

LAB is much larger than all the other color spaces: It encompasses many more colors and more intensities of colors. LAB has four primary colors, two of which are complementary: green-magenta and blue-yellow. Pulling into minus on the a-channel the image turns into a cold green color, in the other plus direction magenta is added to the image. Same goes for the b-channel, because blue is cold color, the slider needs into the minus and for warm yellow will be shown. In LAB, more information about color is delivered in this color space. LAB colors will be the same across different PCs.

RGB has three primary colors: red, green and blue. RGB is used on the web, your PC monitor and camera. The RGB is interpreted in different ways depending on the PC.

Adobe RGB has a larger color gamut than sRGB by around 35% for blues and greens.

Color Conversion

The preferred websites for converting color values to different color spaces are

* Nixsensor.org
* EasyRGB: categorical color names
* colorizer.org
* Rapidtables
* Colorthesaurus

Some of the websites display the color, some only deal with color values.

1. Found on www.vian.app/keywords, email = [beta-tester@test.uzh.ch](mailto:beta-tester@test.uzh.ch), pw = "vian-beta-tester20", in Global, Hues. All hues except for names that mean a collection of hues such as “desaturated”. [↑](#footnote-ref-1)
2. <https://blog.xkcd.com/2010/05/03/color-survey-results/> [↑](#footnote-ref-2)
3. <https://www.epfl.ch/labs/ivrl/research/image-mining/multi-lingual-color-thesaurus/> [↑](#footnote-ref-3)
4. Albrecht Lindner, Bryan Zhi Li, Nicolas Bonnier, and Sabine Süsstrunk, “A large-scale multi-lingual color thesaurus,” IS&T Color and Imaging Conference, 2012. [↑](#footnote-ref-4)