

# COLOUR EXTRACTION FROM CATALOGUE IMAGES

**A project report submitted for the partial fulfillment of the Bachelor of Technology Degree in Electronics & Communication Engineering under Maulana Abul Kalam Azad University of Technology**

BY  
**RAHUL KUMAR CHAURASIA**

ROLL NO : 10400314115 , REGISTRATION NO : 141040110448 OF 2014-2015

Under the Guidance of:

**Prof. ARUNAVA MUKHOPADHYAY**

Department of Electronics & Communication Engineering

For the Academic Year 2017-2018



Institute of Engineering & Management  
Y-12, Salt Lake, Sector-V, Kolkata-700091

Affiliated To:



Maulana Abul Kalam Azad University of Technology

BF-142, Salt Lake, Sector I, Kolkata-700064

# CERTIFICATE

## TO WHOM IT MAY CONCERN

This is to certify that the project report entitled “**COLOUR EXTRACTION FROM CATALOGUE IMAGES**”, submitted by

**RAHUL KUMAR CHAURASIA**

**Registration No.** 141040110448 of 2014-15 **Roll no.** 10400314115

Students of **INSTITUTE OF ENGINEERING & MANAGEMENT**, in partial fulfilment of requirements for the award of the degree of **Bachelor of Technology in Electronics & Communication Engineering**, is a bona fide work carried out under the supervision and guidance of **Prof. ARUNAVA MUKHOPADHYAY** during the final year of the academic session of 2014-2018. The content of this report has not been submitted to any other University or Institute for the award of any other degree.

It is further certified that work is entirely original and its performance has been found to be quite satisfactory.

---

Prof. ARUNAVA MUKHOPADHYAY  
Mentor  
Dept. of Electronics & Communication  
Engineering  
Institute of Engineering & Management

---

Prof. Dr. MALAY GANGOPADHYAY  
H.O.D  
Dept. of Electronics & Communication  
Engineering  
Institute of Engineering & Management

Prof. Dr. A. K. NAYAK

Principal

Institute of Engineering & Management

Sector-V, Salt Lake Electronics Complex, Kolkata-700091

# ACKNOWLEDGEMENT

We should like to take this opportunity to extend our gratitude to the following revered persons without whose immense support, completion of this project wouldn't have been possible.

We are sincerely grateful to our advisor and mentor Prof. **ARUNAVA MUKHOPADHYAY**, Dept. of **Electronics & Communication Engineering**, IEM Kolkata, for her constant support, significant insights and for generating in us a profound interest for this subject that kept us motivated during the entire duration of this project.

We would also like to express our sincere gratitude to **Prof. Dr. Satyajit Chakrabarti** (Director,IEM), **Prof. Dr. Amlan Kusum Nayak** (Principal,IEM) and **Prof. Dr. Malay Gangopadhyay**, HOD of **Electronics & Communication Engineering** and other faculties of Institute of Engineering & Management, for their assistance and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

**RAHUL KUMAR CHAURASIA**

Roll No: 10400314115

Reg. No: 141040110448 of 2014-15

Dept. of Electronics & Communication Engineering

Institute of Engineering & Management, Kolkata

# ABSTRACT

The goal we set for this computer vision experiment was to detect the color of the dominant object of a picture. Searching by color is a common occurrence when it comes to e-commerce websites, and while the nature of the object is usually clear in the description, its color is often more of a problem. Colour is one of the most difficult fields to normalise. To find the colors of an image we use clustering on an  $N \times 3$  matrix where each row of the matrix represents an RGB pixel. The main colors of an image are deemed to be the cluster centers. Instead of using a fixed cluster size, we used an algorithm that dynamically determine the number of clusters (see mean-shift or DBSCAN in [sklearn](#)) because a fixed number of clusters tended to result in muddy colors. Once we extracted the colors, the same needs to be mapped to fixed palette of 22 colours namely Red, Yellow, Blue, Cyan, Teal, Navy etc.

For mapping the extracted colors to palette we used CIELAB Delta E\* equations to calculate the distance between two colors as a metric.

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>5</b>
<b>INTRODUCTION</b>	<b>6</b>
Motivation	6
Objective	7
Organization of the report	9
<b>BACKGROUND</b>	<b>10</b>
<b>PROPOSED STRATEGY</b>	<b>11</b>
DATA PROCESSING PIPELINE	12
<b>EXPERIMENTAL SETUP AND PRACTICAL IMPLEMENTATION</b>	<b>13</b>
HARDWARE/ SERVER SETUP:	13
PREPROCESSOR	15
FEATURE EXTRACTOR:	19
COLOR CLUSTERING MODEL:	20
Parameters:	20
Parameter estimation:	20
Why should I use DBSCAN?	22
Abstract Algorithm	22
DOMINANT COLOR EXTRACTION	25
PALETTE MAPPING	26
Color distance	27
Delta E 1976	29
Delta E 2000	29
<b>RESULTS:</b>	<b>32</b>
<b>CONCLUSION</b>	<b>34</b>
<b>REFERENCES</b>	<b>35</b>

# INTRODUCTION

## Motivation

One of the main problems of this and the coming decade is to make sense of the data we have. This argument has led to the rise of fields like big data analytics and data mining to make organize semi-structured data. Their goal is to transform the streams of semi-structured data into datasets for analysis and to build A.I. applications. Colour is one of the most difficult fields to normalise.

Searching by color is a common occurrence when it comes to e-commerce websites, and while the nature of the object is usually clear in the description, its color is often more of a problem. Some of the colors can be deciphered by a human but all are very difficult for a computer. Even simple colors such as snakeskin or periwinkle can be hard to process automatically. In the past, organizations used a range of techniques to map retailer colors to an organization specific palette colors, from simple keyword matching to complex ML models. We found that these methods produced unsatisfactory results due to complexity of color names used by retailers. The only way we've found to accurately determine product color is via the product images.

## Objective

The objective is to extract the dominant colours of catalogue images , map the extracted colours to a standard palette of 22 colours and then output a dictionary in following format.

```
{  
  "filename": <filename>,  
  "colors": [  
    {  
      "base_color_hex": <hexcode of the mapped color>,  
      "dominance_level": <percentage of color present in the foreground>,  
      "Base_color": <name of the base colour>  
    }  
  ]  
}
```

*Example Input / Output:*



```
{  
  "filename": "2.png",  
  "colors": [  

```

```
{
  "base_color_hex": "#800000",
  "dominance_level": "0.927237515562",
  "base_color": "maroon"
},
{
  "base_color_hex": "#a52a2a",
  "dominance_level": "0.0453036381242",
  "base_color": "brown"
},
{
  "base_color_hex": "#ffc0cb",
  "dominance_level": "0.0126573523309",
  "base_color": "pink"
},
{
  "base_color_hex": "#808000",
  "dominance_level": "0.0107207082584",
  "base_color": "olive"
}
]
```



## Organization of the report

The body of the report begins with a brief introduction of the project by stating the initial motivation to start the same along with the objective and the structure of the report.

Then we talk about the ‘Background’ and similar work related to our objective and its Pros and Cons. We then look at our proposed strategy keeping in mind the previous works and there cons. We also look at a the flow chart and how the processing pipeline is divided into small modules.

In the Experimental Setup and Implementation section , we dive into to the hardware and software requirements and all the knitty gritty of various processing modules. We also look at the source code followed by the ‘Results’ section containing sample input-output from our pipeline.

We then conclude the report by stating all the constraints and the bottlenecks we of the pipeline followed by a list of references without which the successful completion of the project could not have been possible

# BACKGROUND

We are obviously not the first to attempt to address this problem, but we've taken a slightly different approach from the current solutions in the market. We can divide the applications trying to solve this problem into two different groups that provide different experiences:

- Commercial applications like [vue.ai](https://vue.ai)
- Open source scripts like [josip/node-colour-extractor](https://github.com/josip/node-colour-extractor) or [lovesh/color-thief](https://github.com/lovesh/color-thief)

The open-source scripts listed are going for a different and simpler problem. They extract the palette composed of the principal colors of an image. While they do the job well, they can't be applied out of the box to fashion images because we are not interested in the colors in the entire image. Using them directly would result in the color of the background being detected as the main color of the given image.

Moreover they do not provide a *word* for the extracted colors, and indexing RGB values would not improve the search experience.

Vue.ai showcases some impressive results and provides many more features than our experiment. We wanted to provide a quick, easy way for users to enrich their records a bit and improve their ranking, especially if they have to deal with poor descriptions in their records.

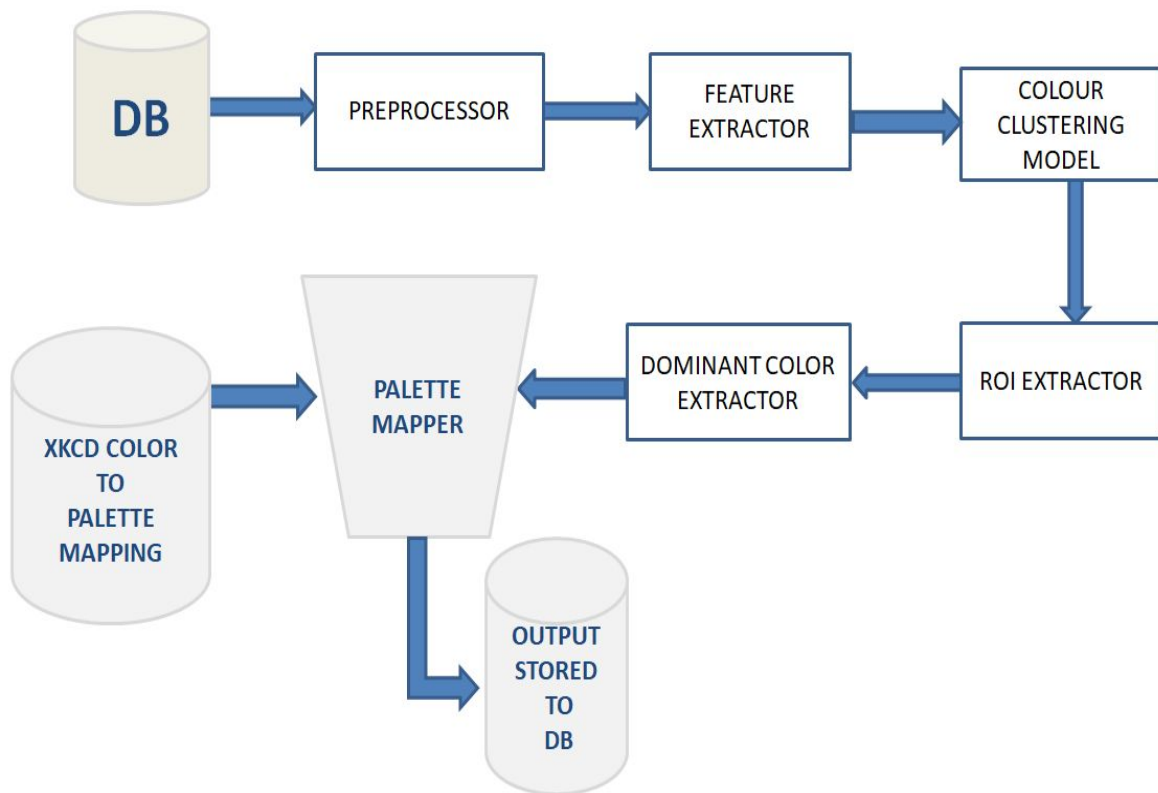
## PROPOSED STRATEGY

The problem we try to address is a relatively complex one and can be solved by a number of different ways. Most state-of-the-art computer vision frameworks these days adopt the Deep Learning path by using techniques such as Convolutional Neural Network to classify images. This is a path we decided not to take. Neural networks offer astounding results, but to do so they also need large datasets in addition to dedicated hardware which shortens the computation time. We started this project as an experiment and we began with simpler methods to see where those would take us.

The sequence of algorithms we chose was able to process images reasonably fast on a machine with a classic CPU and can be installed as simply as any other python package.

The following flow represents our current strategy. The idea is create a data processing pipeline as distributed acyclic graph (DAG) where each nodes represent a processing module acting as a data transformer and edges represents input data in various forms.

## DATA PROCESSING PIPELINE



# EXPERIMENTAL SETUP AND PRACTICAL IMPLEMENTATION

## HARDWARE/ SERVER SETUP:

*#assuming the server is an ubuntu instance.*

#####

steps:

1) Install python3.6

sudo add-apt-repository ppa:jonathonf/python-3.6

sudo apt-get update

sudo apt-get install python3.6

#####

2) Create python virtual env

sudo pip3 install virtualenv

virtualenv -p /usr/bin/python3.6 homecanvas

#####

4) Activate homecanvas virtual environment **and** install dependencies

source homecanvas/bin/activate

INSTALL THE FOLLOWING DEPENDENCIES:

colormath==3.0.0

cycler==0.10.0

decorator==4.3.0

kiwisolver==1.0.1

```
matplotlib==2.2.2
networkx==2.1
numpy==1.14.3
opencv-python==3.4.0.12
pandas==0.22.0
Pillow==5.1.0
pyparsing==2.2.0
python-dateutil==2.7.2
pytz==2018.4
PyWavelets==0.5.2
scikit-image==0.13.1
scikit-learn==0.19.1
scipy==1.0.1
six==1.11.0
sklearn==0.0
```

BY USING : `pip -r homecanvas_delivery/requirements.txt`

5) Run code:

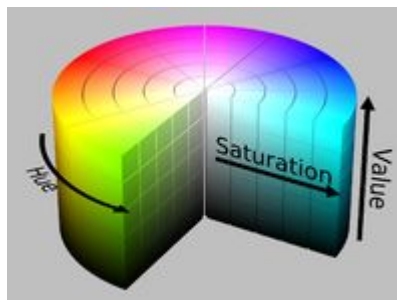
```
source homecanvas/bin/activate
```

```
python driver.py <input_image_folder> <output_folder>
```

## PREPROCESSOR

The objective is to apply a series of operations to achieve the following:

1. Reading the input image in BGR format and create a thumbnail version of the same to speedup the processing.
2. Foreground extraction :
  - a. Convert the BGR image to HSV format and get the S channel.



*Unlike RGB and CMYK, which are defined in relation to primary colors, HSV is defined in a way that is similar to how humans perceive color.*

*HSV is named as such for three values: hue, saturation, and value.*

*This color space describes [colors](#) (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness value.*

We used 'S' channel because most of the input images have white as background colour and white has 'S' value equal to 0.

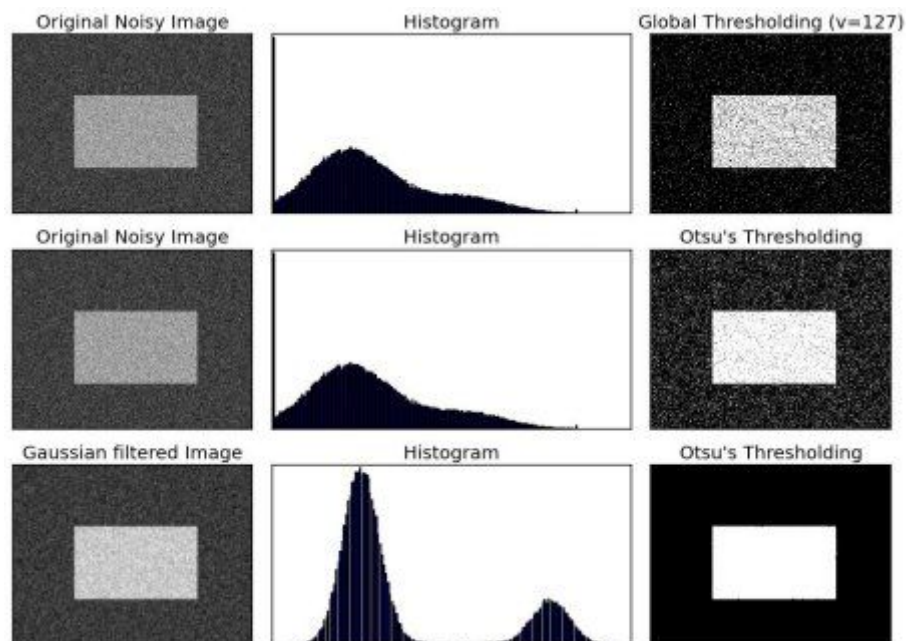
So it was very easy to create a binary mask and separate background and foreground

- b. Otsu's Binarization:

One we extracted the S channel, we need a way to binarize the images

in-order to create the mask which can be generalised over to a large subset of input data.

*Consider a bimodal image (In simple words, bimodal image is an image whose histogram has two peaks). For that image, we can approximately take a value in the middle of those peaks as threshold value, right ? That is what Otsu binarization does. So in simple words, it automatically calculates a threshold value from image histogram for a bimodal image. (For images which are not bimodal, binarization won't be accurate.)*





OUTPUT FROM BINARIZATION:



INPUT IMAGE



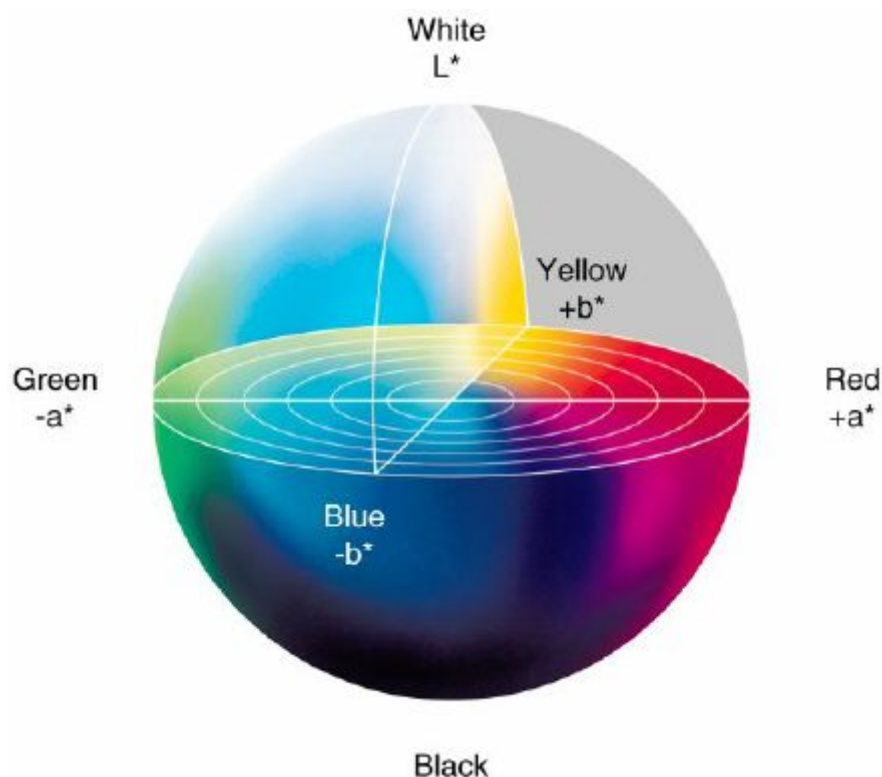
OUTPUT IMAGE

### 3. Convert the image to RGB and LAB space

#### *LAB COLOR SPACE:*

The **CIELAB color space** (also known as **CIE  $L^*a^*b^*$**  or sometimes abbreviated as simply "Lab" color space) is a [color space](#) defined by the [International Commission on Illumination](#) (CIE) in 1976. It expresses color as three numerical values,  $L^*$  for the lightness and  $a^*$  and  $b^*$  for the green–red and blue–yellow color components. CIELAB was designed to be **perceptually uniform** with respect to human color vision, meaning that the same amount of numerical change in these values corresponds to about the same amount of visually perceived change.

This space unlike other RGB allows change in color as function of euclidean distance and thus is much more useful during clustering operations



## FEATURE EXTRACTOR:

This module convert an image whose initial form is a  $N \times M \times 3$  (where  $N$  is height and  $M$  is the width) matrix in RGB or LAB space to  $NM \times 5$  by the following steps:

1. We compute an array where the subarrays contain index values  $0,1,\dots$  varying only along the corresponding axis ( $x$  and  $y$ ) for each of the pixels .
2. Next step is to concatenate the above calculated values to their corresponding pixels such that :  
each pixel represents an array of 5 elements  $\rightarrow [L, A, B, X \text{ index}, Y \text{ index}]$

This would enhance the output of clustering by recognizing shapes along with colors.

## COLOR CLUSTERING MODEL:

The main colors of an image are deemed to be the cluster centers. Our use of clustering was inspired by this [article from Charles Leifer](#). Instead of using a fixed cluster size, we used an algorithm that dynamically determine the number of clusters (DBSCAN in [sklearn](#)) because a fixed number of clusters tended to result in muddy colors.

**DBSCAN** is a well-known data clustering algorithm that is commonly used in data mining and machine learning.

Based on a set of points (let's think in a bidimensional space as exemplified in the figure), DBSCAN groups together points that are close to each other based on a distance measurement (usually Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions.

### Parameters:

The DBSCAN algorithm basically requires 2 parameters:

**eps:** the minimum distance between two points. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.

**minPoints:** the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

### Parameter estimation:

The parameter estimation is a problem for every data mining task. To choose good parameters we need to understand how they are used and have at least a basic previous knowledge about the data set that will be used.

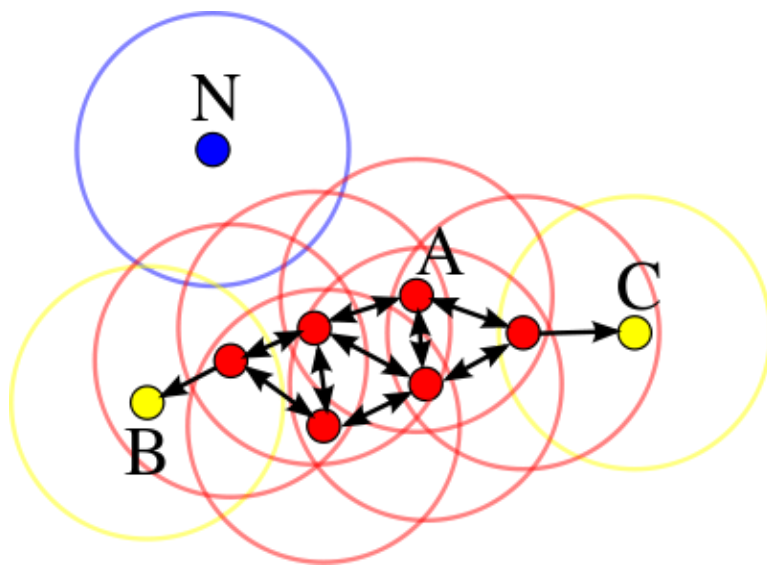
**eps:** if the eps value chosen is too small, a large part of the data will not be clustered.

It will be considered outliers because don't satisfy the number of points to create a dense region. On the other hand, if the value that was chosen is too high, clusters will merge and the majority of objects will be in the same cluster. The eps should be

chosen based on the distance of the dataset (we can use a k-distance graph to find it), but in general small eps values are preferable.

**minPoints:** As a general rule, a minimum minPoints can be derived from a number of dimensions (D) in the data set, as  $\text{minPoints} \geq D + 1$ . Larger values are usually better for data sets with noise and will form more significant clusters. The minimum value for the minPoints must be 3, but the larger the data set, the larger the minPoints value that should be chosen.

You can find more about parameter estimation [here](#).



*In this diagram,  $\text{minPts} = 4$ . Point A and the other red points are core points, because the area surrounding these points in an  $\epsilon$  radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.*

### Why should I use DBSCAN?

The DBSCAN algorithm should be used to find associations and structures in data that are hard to find manually but that can be relevant and useful to find patterns and predict trends.

Clustering methods are usually used in biology, medicine, social sciences, archaeology, marketing, characters recognition, management systems and so on.

After a lot of trial and error and exploratory data analysis the following Configuration was used for DBSCAN:

eps=7  
min\_POINTS=5

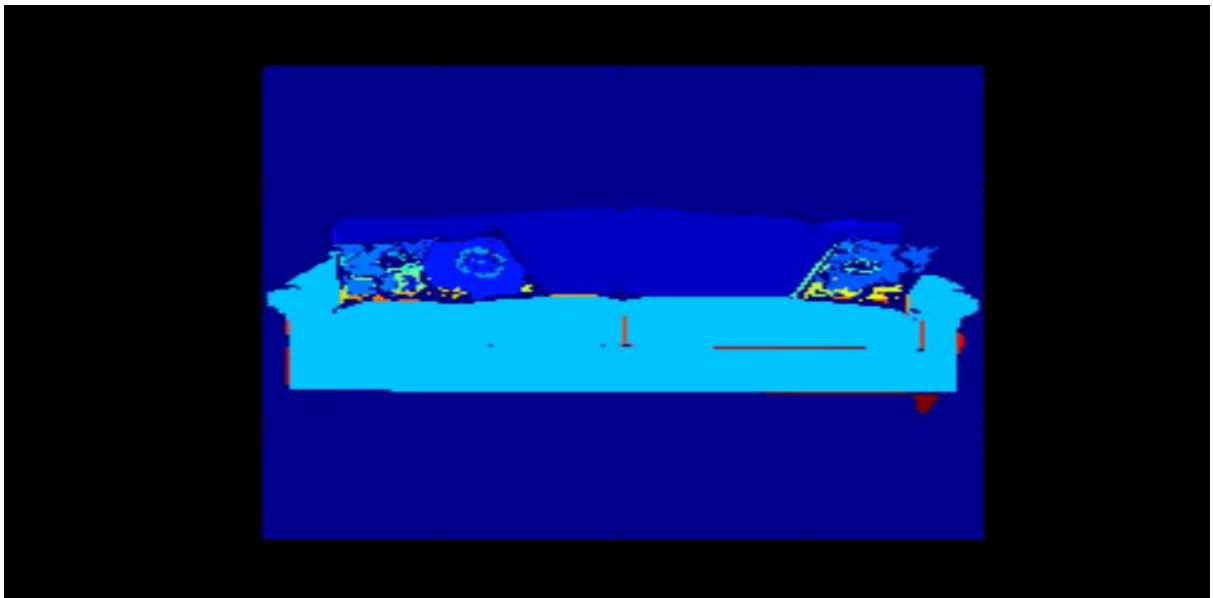
### Abstract Algorithm

The DBSCAN algorithm can be abstracted into the following steps:<sup>[5]</sup>

1. Find the  $\epsilon$  (eps) neighbors of every point, and identify the core points with more than minPts neighbors.
2. Find the [connected components](#) of *core* points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an  $\epsilon$  (eps) neighbor, otherwise assign it to noise.

A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory. The original DBSCAN algorithm does not require this by performing these steps for one point at a time.

SAMPLE OUT REPRESENTING CLUSTER LABELS FROM DBSCAN :



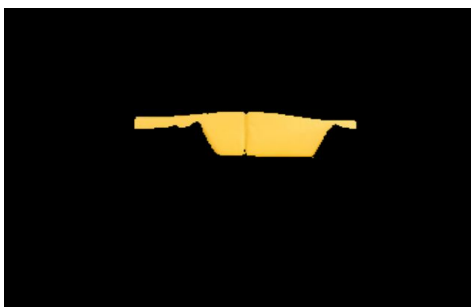
## ROI EXTRACTION

The next step is create separate copies of the image where each image is corresponds to a cluster as a foreground

Steps:

1. Loop over the label\_map extracted from DBSCAN output and create binary mask for every cluster.
2. Using the above generated label\_masks extract the foreground

Example of ROI\_MAPS:





## DOMINANT COLOR EXTRACTION

In this module we take each of the roi\_maps and apply the following steps:

1. Calculate the dominance level:

$$\text{dominance\_level} = \text{sum}(\text{label\_masks}) / (\text{sum}(\text{binary mask}))$$

2. Flatten the image such that  $N \times M \times 3 \rightarrow NM \times 3$  and exclude all the background pixels i.e black { rgb(0,0,0) }
3. Convert the pixels from RGB to HSV and split H,S,V channels

4. To get the dominant color we take :

h= most frequent value of H ( by calculating its histogram)

s= maximum of S (to get the most amount of color)

v= average of S ( to get the average lighting condition )

pixel= [h, s,v] , in hsv format.

Convert pixel to rgb hsv

## PALETTE MAPPING

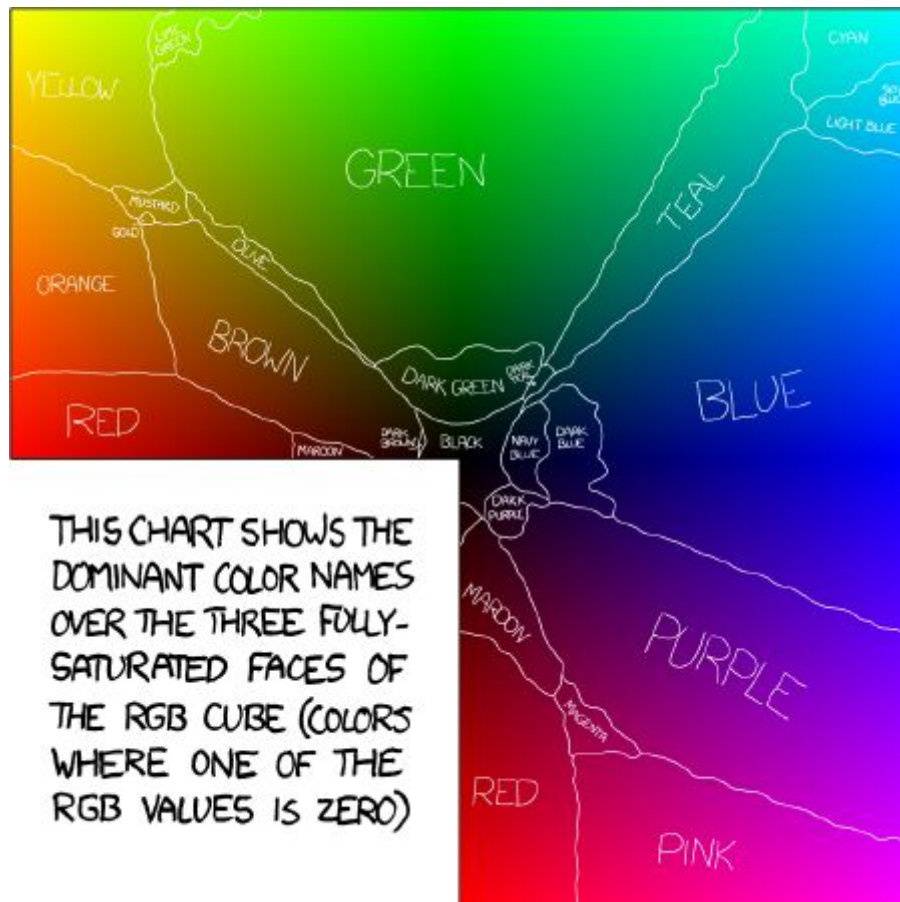
After extracting rgb and hex values of the dominant colors from the image , its time to map the same to the standard color palette.

We wanted to to map the extracted colors to the following palette:

	Hex	RGB
Red	#e6194b	(230, 25, 75)
Yellow	#ffe119	(255, 225, 25)
Blue	#0082c8	(0, 130, 200)
Cyan	#46f0f0	(70, 240, 240)
Teal	#008080	(0, 128, 128)
Navy	#000080	(0, 0, 128)
Green	#3cb44b	(60, 180, 75)
Mint	#aaffc3	(170, 255, 195)
Olive	#808000	(128, 128, 0)
Lime	#d2f53c	(210, 245, 60)
Coral	#ffd8b1	(255, 215, 180)
Orange	#f58231	(245, 130, 48)
Purple	#911eb4	(145, 30, 180)
Magenta	#f032e6	(240, 50, 230)
Pink	#fabebe	(250, 190, 190)
Lavender	#e6beff	(230, 190, 255)
Brown	#aa6e28	(170, 110, 40)
Beige	#fffac8	(255, 250, 200)
Maroon	#800000	(128, 0, 0)
Grey	#808080	(128, 128, 128)
White	#FFFFFF	(255, 255, 255)
Black	#000000	(0, 0, 0)

But mapping extracted colors which is a subset of  $256^3$  to 22 colors is a bad idea as  $22/256^3 = 0.0000013113$ .

Mapping from a hex value to a color name is more complex than it seems; for instance, when is a red considered pink or when does grey become black. The solution to this problem came from Randall Munroe of xkcd fame and his Color Survey.



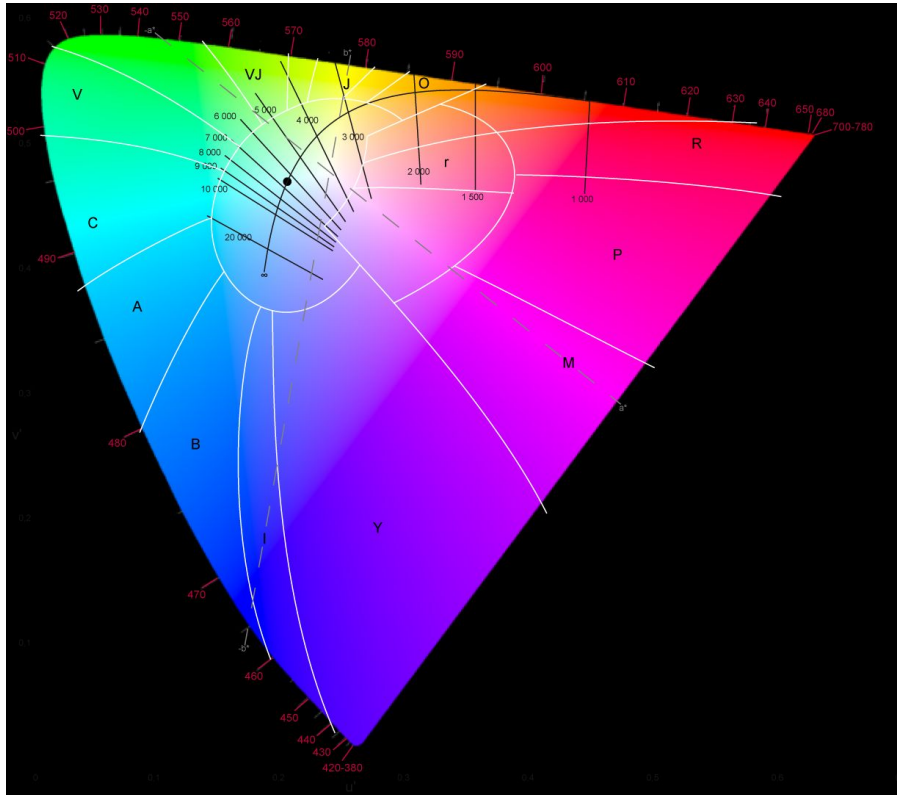
he survey consisted of asking his readers to identify hex colors by name. The result is a list of 200k RGB values and names for those colors picked from a small set.

## Color distance

The data from xkcd is not complete so we need some way of finding similar colors.

To define a metric of color difference we first need a space in which to measure distance. The RGB space is not suited to measuring color difference because distance magnitudes in the color space do not necessarily correspond to the magnitude of color

difference as perceived by humans. To rectify this deficiency the International Commission on Illumination (CIE) defined the [Lab color space](#) which aims to attain so called perceptual uniformity.



CIE has defined a number of [color difference](#) functions. The simplest being CIE1976 which is euclidean distance on the Lab colorspace. The latest, CIE2000, attempts to rectify issues that CIE1976 has with perceptual uniformity as shown in the figure below. Under CIE1976, the two shades of green are considered further apart than blue and purple but under CIE2000 this is no longer a problem (thanks to [Rok Carl](#) for the image).

## Delta E 1976

$$\Delta E_{ab}^* = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2}$$

## Delta E 2000

The CIE organization decided to fix the lightness inaccuracies by introducing dE00.

It's currently the most complicated, yet most accurate, CIE color difference algorithm available.

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

$$\Delta L' = L_2^* - L_1^*$$

$$\bar{L} = \frac{L_1^* + L_2^*}{2} \quad \bar{C} = \frac{C_1^* + C_2^*}{2}$$

$$a'_1 = a_1^* + \frac{a_1^*}{2} \left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right) \quad a'_2 = a_2^* + \frac{a_2^*}{2} \left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)$$

$$\bar{C}' = \frac{C'_1 + C'_2}{2} \text{ and } \Delta C' = C'_2 - C'_1 \quad \text{where } C'_1 = \sqrt{a'^2_1 + b'^2_1} \quad C'_2 = \sqrt{a'^2_2 + b'^2_2}$$

$$h'_1 = \text{atan2}(b'_1, a'_1) \mod 360^\circ, \quad h'_2 = \text{atan2}(b'_2, a'_2) \mod 360^\circ$$

$$\Delta h' = \begin{cases} h'_2 - h'_1 & |h'_1 - h'_2| \leq 180^\circ \\ h'_2 - h'_1 + 360^\circ & |h'_1 - h'_2| > 180^\circ, h'_2 \leq h'_1 \\ h'_2 - h'_1 - 360^\circ & |h'_1 - h'_2| > 180^\circ, h'_2 > h'_1 \end{cases}$$





$$\Delta H' = 2\sqrt{C'_1 C'_2} \sin(\Delta h'/2), \quad \bar{H}' = \begin{cases} (h'_1 + h'_2 + 360^\circ)/2 & |h'_1 - h'_2| > 180^\circ \\ (h'_1 + h'_2)/2 & |h'_1 - h'_2| \leq 180^\circ \end{cases}$$

$$T = 1 - 0.17 \cos(\bar{H}' - 30^\circ) + 0.24 \cos(2\bar{H}') + 0.32 \cos(3\bar{H}' + 6^\circ) - 0.20 \cos(4\bar{H}' - 63^\circ)$$

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

$$R_T = -2\sqrt{\frac{\bar{C}'^7}{\bar{C}'^7 + 25^7}} \sin \left[ 60^\circ \cdot \exp \left( - \left[ \frac{\bar{H}' - 275^\circ}{25^\circ} \right]^2 \right) \right]$$

## CIE1976

COLOR 1	COLOR 2	DISTANCE
		25.94
		21.40

## CIE2000

COLOR 1	COLOR 2	DISTANCE
		5.92
		9.33

- Xkcd has a dataset which contains 955 hex values and their color names.
- We used this mapping and our standard 22 color palette to create the following mapping using CIE 2000 and some manual annotation:

base_color	base_color_hex	color_name	hex	lab	rgb
lavender	#e6e6fa	cloudy blue	#acc2d9	['77.4971556749', '-2.56399231616', '-13.9724042775']	['172', '194', '217']
mint	#3eb489	dark pastel green	#56ae57	['64.1101325495', '-44.5110918113', '36.5039195112']	['86', '174', '87']
olive	#808000	dust	#b2996e	['64.4631133314', '3.05166844388', '26.0453429865']	['178', '153', '110']
lime	#bfff00	electric lime	#a8ff04	['91.6185819116', '-55.116854953', '87.5958201679']	['168', '255', '4']
green	#00ff00	fresh green	#69d84f	['77.6842154879', '-56.7324704498', '56.2325476156']	['105', '216', '79']

- Now we take the extracted pixel and use color distance to map it to one of the 955 colors from the xkcd
- We can then use the above table to map it against the standard palette using the Base colors.

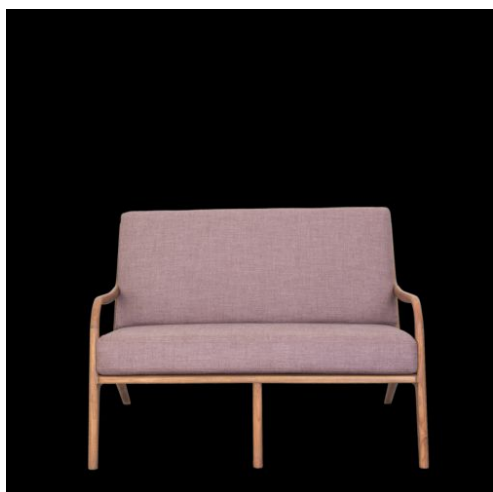


Color Mapping Flow

## RESULTS:

	<pre>{   "base_color_hex": "#ffa500",   "dominance_level": "0.872777",   "base_color": "orange"  }</pre> 
	<pre>{   "base_color_hex": "#800000",   "dominance_level": "0.9272375",   "base_color": "maroon"  }</pre> 
	<pre>{   "base_color_hex": "#3eb489",   "dominance_level": "0.8780401",   "base_color": "mint"  }</pre> 





```
{  
  "base_color_hex": "#ffc0cb",  
  "dominance_level": "0.194775",  
  "base_color": "pink"  
}
```



```
}
```



```
{  
  "base_color_hex": "#ff0000",  
  "dominance_level": "0.95213",  
  "base_color": "red"  
}
```



```
}
```



```
{  
  "base_color_hex": "#008080",  
  "dominance_level": "0.88415",  
  "base_color": "teal"  
}
```



```
}
```

# CONCLUSION

- Foreground extraction method used here is effective only with white/black backgrounds as most of the catalogue images with this configuration. Complex backgrounds like a scenery or gradient needs more complex algorithm and is out of scope for our context.
- Our 'Feature Extraction' and 'Clustering' technique is consistent in giving accurate results.
- 'Dominant Color Extraction' technique is state of the art and has been applied after lots of trial and error and lots of intuition.
- Our 'Palette Mapping' technique took most of the time .The 'XKCD TO PALETTE MAPPING' required lots of manual annotation and still produces some error.
- The 'Delta E CIE' equation which is used to calculate color distance still is not very accurate .

## REFERENCES

- A Similar approach proposed by an upcoming european ecommerce platform:

<https://making.lyst.com/2014/02/22/color-detection/>

- Brief introduction of the DBSCAN CLUSTERING ALGORITHM :

<https://en.wikipedia.org/wiki/DBSCAN>

- Link to list of 954 colours and corresponding hex values sampled from survey:

<https://xkcd.com/color/rgb/>

- Gist from the XKCD Survey about the colours and human perceptual uniformity:

<https://blog.xkcd.com/2010/05/03/color-survey-results/>

- DBSCAN API References:

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

- LAB Colorspace and color distance:

[https://en.wikipedia.org/wiki/CIELAB\\_color\\_space](https://en.wikipedia.org/wiki/CIELAB_color_space)

