NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY
[IJIERT] ISSN: 2394-3696
Website: ijiert.org
VOLUME 6, ISSUE 11, Nov. -2019

IMAGE COLOR SEGMENTATION USING K-MEANS CLUSTERING ALGORITHM

Chaitanya Krishna Surryadevara
Department of Information Technology
cssuryadevara001@my.wilmu.edu

ABSTRACT

Working with images can be a very time-consuming task, especially if you have many images to work on. Machine learning can thus be a great time-saver for various image analysis and editing tasks, such as finding the dominant colors of an image thanks to the K-means clustering algorithm. The K-means clustering algorithm defines a number K of clusters and the best "centroids" to cluster the data around. When applied to images, it allows extracting the k dominant colors in an image to be used for other purposes.

Keywords: ML, AI, image analysis, K-means clustering, clustering-algorithm, extract, dominant-color, percentage, editing, algorithm, finding.

INTRODUCTION

I wanted to write some software that would allow me to extract a set of colours from an image, and do it in a way that seems natural and takes human perception into consideration. A colour scheme can often sum up the 'vibe' of an entire image, and so I thought it would be a useful thing to be able to do.

So… I spent some time thinking of some ways that I could do this. I devised some fairly simple algorithms that would, for example, chop the image regularly into chunks and output the mean colour of each of these parts. Maybe extra layers could be added where the chunks are compared to each other and joined into groups, and maybe each colour could be recursively combined with another until the desired number of colours is reached. I quickly realised, though, that this problem had already been solved in the general case, and in a way that will work quite nicely.

The solution

K-means clustering is a method through which a set of data points can be partitioned into several disjoint subsets where the points in each subset are deemed to be 'close' to each other (according to some metric). A common metric, at least when the points can be geometrically represented, is your bog standard euclidean distance function. The 'k' just refers to the number of subsets desired in the final output. It turns out that this approach is exactly what we need to divide our image into a set of colours.

METHODOLOGY

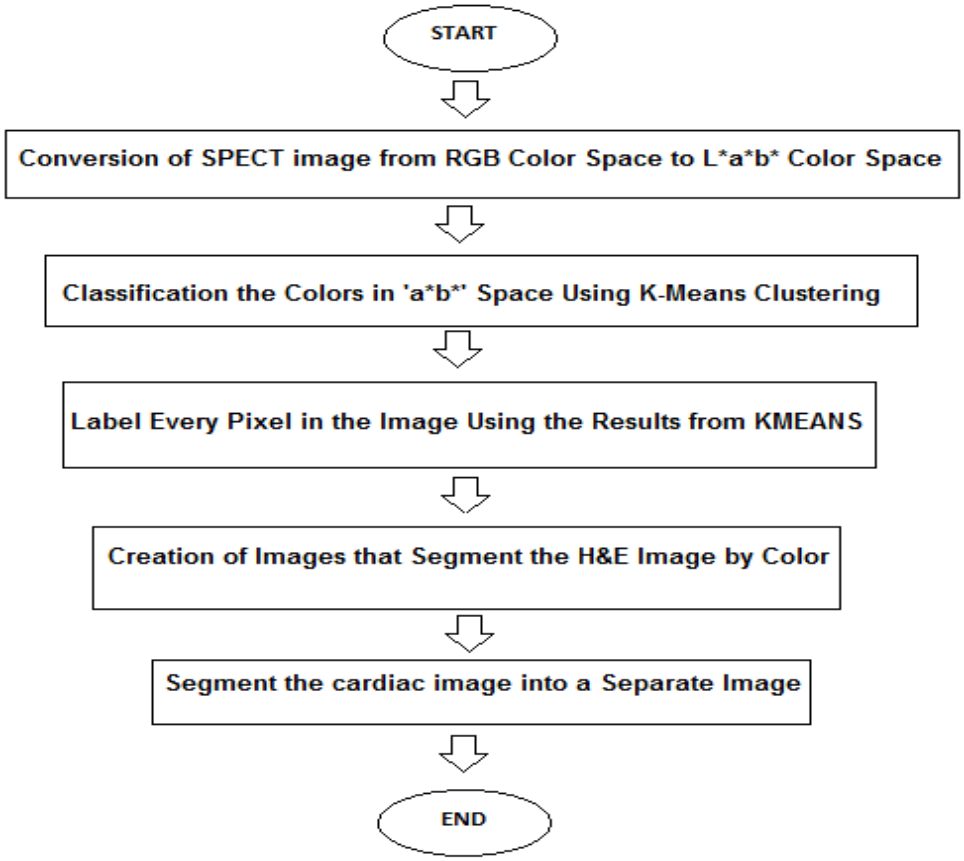


Fig1. Steps of Colour-Based Segmentation Using K-Means Clustering

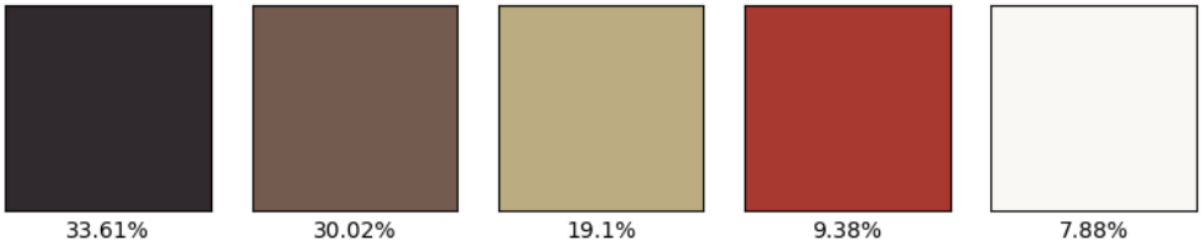
- Line 1-5 – Importing packages required to find most dominant colors in an image.
- Line 7 – Defining the no. of clusters for the KMeans algorithm.
- Line 9 – Reading our input image.
- Line 10 – Keeping a copy of it for future use.
- Line 11 – Printing its shape.
- Line 13 – Resizing our image to get results fast.
- Line 14 – Printing resized image shape.
- Line 16 – Flattening the image. In this step, we are just keeping all the columns of the image after each other to make just one column out of it. After this step, we will be left with just 1 column and rows equal to the no. of pixels in the image.
- Line 17 – Let's check the shape of the flattened image now.

```
Org image shape -->  (1932, 2835, 3)
After resizing shape -->  (200, 293, 3)
After Flattening shape -->  (58600, 3)
```

Flattened shape becomes 200*293 = 58600

- Line 19 – Making a KMeans Clustering object with n_clusters set to 5 as declared in Line 7.

- Line 20 – Fit our image in Kmeans Clustering Algorithm. In this step, the flattened image is working as an array containing all the pixel colors of the image. These pixel colors will now be clustered into 5 groups. These groups will have some centroids which we can think of as the major color of the cluster (In Layman's terms we can think of it as the boss of the cluster).
- Line 22 – We are extracting these cluster centers. Now we know that these 5 colors are the dominant colors of the image but still, we don't know the extent of each color's dominance.
- Line 24 – We are calculating the dominance of each dominant color. ***np.unique(kmeans.labels_,return_counts=True)***, this statement will return an array with 2 parts, first part will be the predictions like [2,1,0,1,4,3,2,3,4…], means to which cluster that pixel belongs and the second part will contain the counts like [100,110,310,80,400] where 100 depicts the no. of pixels belonging to class 0 or cluster 0(our indexing starts from 0), and so on, and then we are simply dividing that array by the total no. of pixels, 1000 in the above case, so the percentage array becomes [0.1,0.11,0.31,0.08,0.4]
- Line 25 – We are zipping percentages and colors together like, **[(0.1,(120,0,150)), (0.11,(230,225,34)), …]**. It will consist of 5 tuples. First tuple is **(0.1,(120,0,150))** where first part of the tuple **(0.1) is the percentage and (120,0,150) is the color**.
- Line 26 – Sort this zip object in descending order. Now the first element in this sorted object will be the percentage of the most dominant colors in the image and the color itself.
- Line 28-36 – We are plotting blocks of dominant colors.



Blocks of 5 dominant colors

- Line 38-54 – We are plotting the following bar.

The bar represents the proportions of dominant colors

- Line 56-72 – We are creating the final result.
- Line 79- We are saving the output image.

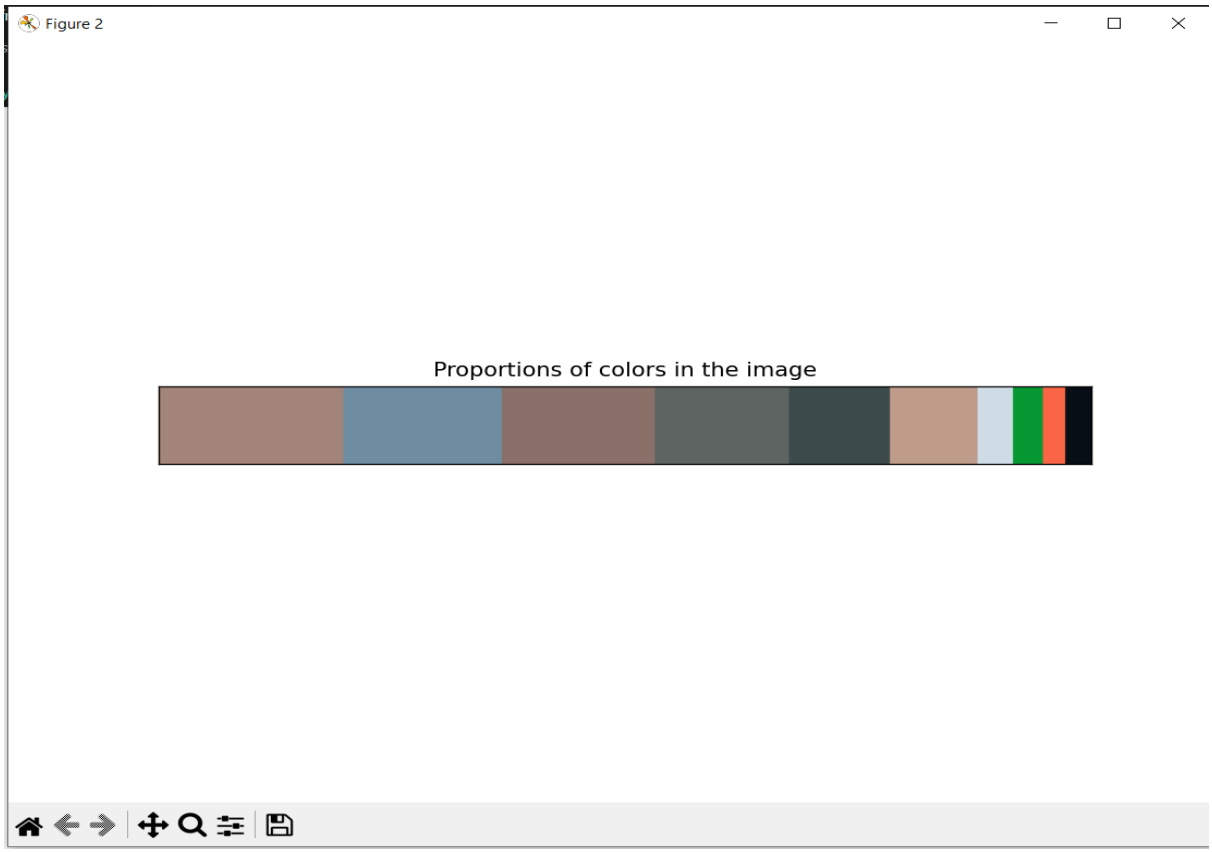
Final results

OUTPUTS

Img1. Input Image

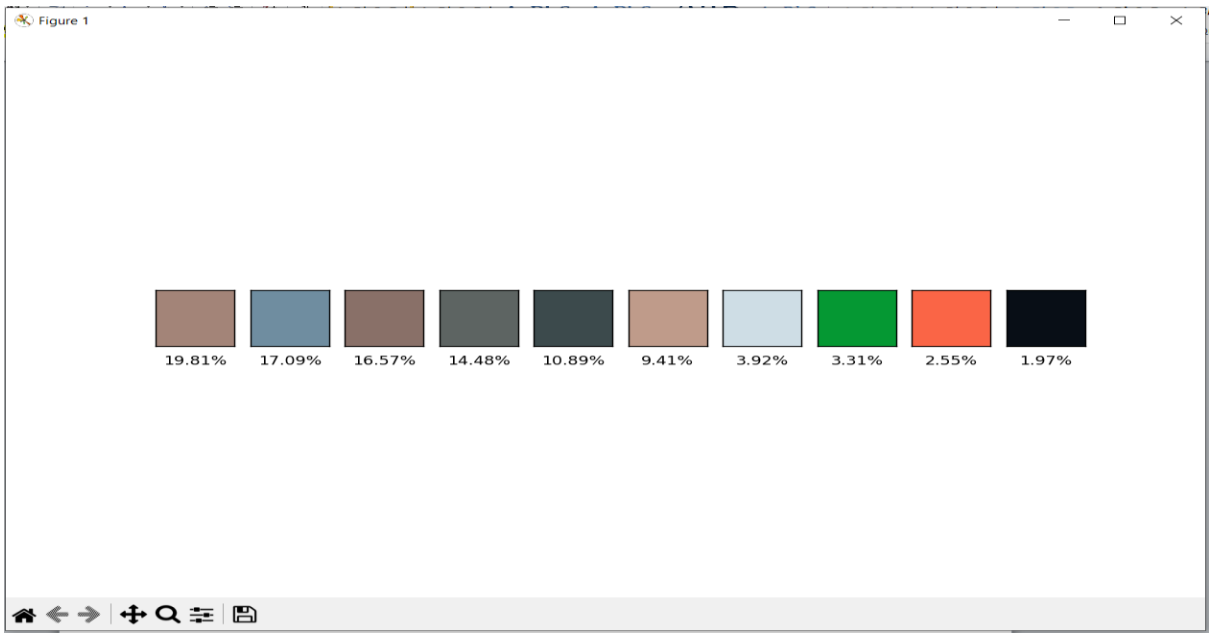
K-means is a clustering algorithm that can be used to classify colors in an image. The basic idea is to convert the image into a feature space (such as RGB or HSV), then use k-means to cluster the pixels into k clusters, where each cluster represents a different color

OUTPUT



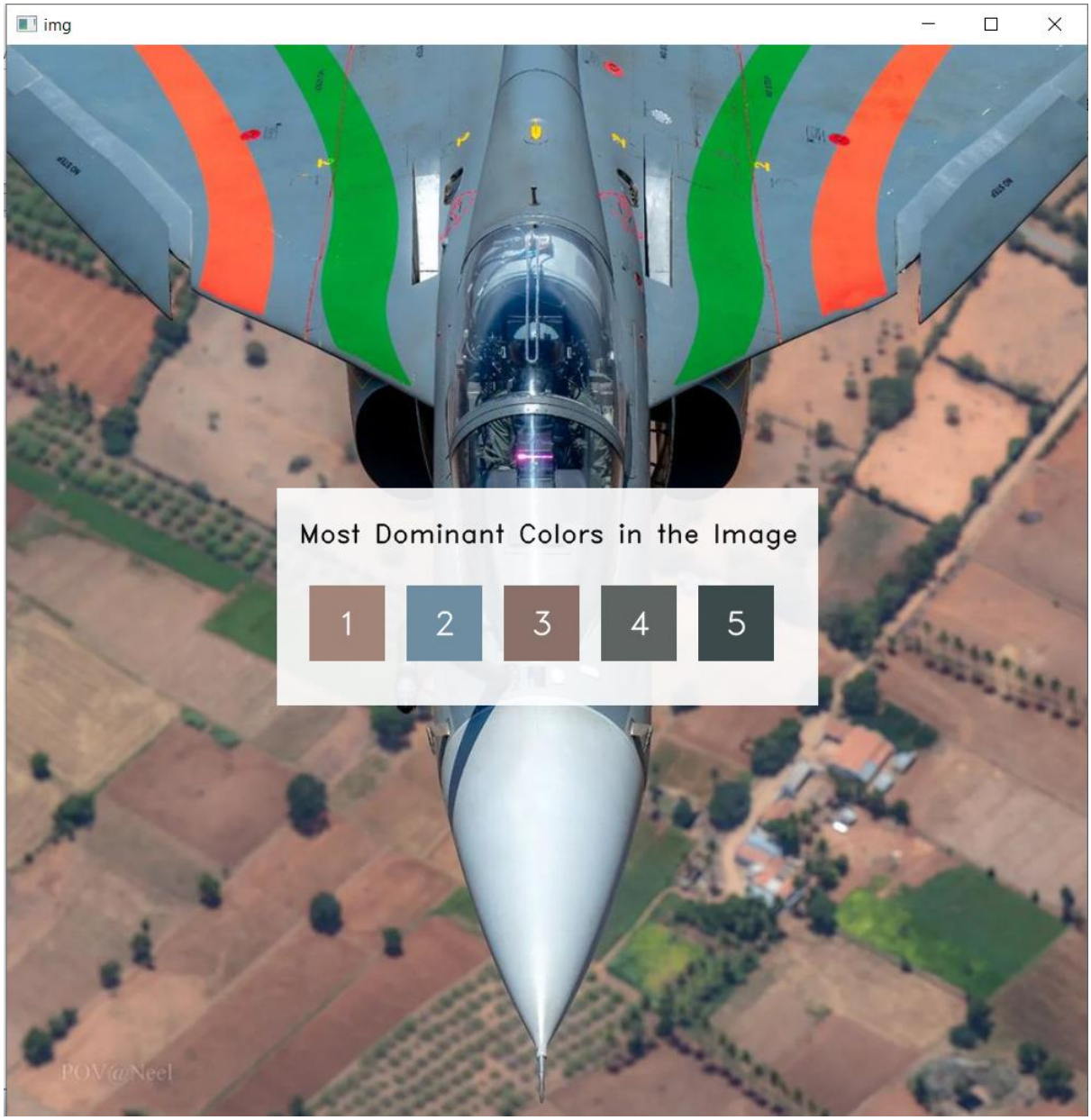
Img2. Top 10 Dominating Colors

The software extracts to 10 most dominated colour from the image. These colours are then displayed to the user. Each colour is in block to notify how much of it is present in the picture



Img3. Percentage of Top 10 Dominating Colors

The software extracts to 10 most dominated colour from the image. These colours are then displayed to the user. Each colour is in block to notify how much of it is present in the picture. This output shows what percentage of colour is present in the image.



Img4. Top 5 Most Dominating Colors

This output shows the top 5 most dominating colour in the image. The software also shows the input image in the background.

RESULTS (summary)

The software takes input as an image runs K-mean algorithm on it and gives output as the most domination colours in the image. The software also shows the percentage of colour present in the image. The software worked for each image input and showed the most dominating colour successfully each time.

CONCLUSION

The use of K-means clustering for color segmentation can be a powerful tool for identifying and quantifying objects in an image based on their colors. In this tutorial, we demonstrated how to use the K-means algorithm, along with OpenCV and scikit-learn, to perform color segmentation and count the number of objects of each color in an image. This technique can be applied to a variety of scenarios where it is necessary to analyze and classify objects in an image based on their colors.

FUTURE SCOPE

The future scope of a "Color Classification from Image using K-Means" project could include several possibilities such as:

- Incorporating additional image processing techniques to improve the accuracy of color classification.
- Using the color classification results to perform other image processing tasks, such as object recognition or image segmentation.
- Incorporating the project into a larger image processing pipeline, such as an image search engine or an image editing application.
- Integrating the project with deep learning for more accurate color classification.
- Implementing it in real-time applications like Surveillance systems, Autonomous cars, Robotics etc.

REFERENCES

[1]. R. Yager and D. Filev, Generation of Fuzzy Rules in Mountain Clustering, In Journal of Intelligent and Fuzzy System, vol. 2(3), pp. 209-219, (1992).

[2]. Xie, K. & Zhang, T. & Xi, L. & Li, W.-X & Ping, X.-J. (2014). Steganalysis of heterogeneous images using k-means clustering. Yingyong Kexue Xuebao/Journal of Applied Sciences. 32. 543-550. 10.3969/j.issn.0255-8297.2014.05.017.

[3]. Yang, Kang & Liu, Changhua & Wu, Xiaoming & Li, Hao. Segmentation of Rapeseed Color Drone Images Using K-Means Clustering. CSAE : Proceedings of the 3rd International Conference on Computer Science and Application Engineering. 1-4. 10.1145/3331453.3361297.

[4]. K. M. Bataineh, M. Naji and M. Saqer, A Comparison Study between Various Fuzzy Clustering Algorithm, In Jordan Journal of Mechanical and Industrial Engineering, vol. 5, no. 4, August (2011).

[5]. Maw, Swe & Zin, T.T. & Yokota, M. & Min, E.P.. (2018). Classification of shape images using K-means clustering and deep learning. ICIC Express Letters. 12. 1017-1023. 10.24507/icicel.12.10.1017.

[6] Koheri Arai and Ali Ridho Barakbah, Heirarchical K -means: An algorithm for Centroid initialization for K -means, Saga University, (2007).

[7] D. L. Pham, C.Y. Xu, J.L. Prince, A Survey of Current Methods in Medical Image Segmentation, In Annual Review of Biomedical Engineer, (2000).

[8] Pallavi Purohit and Ritesh Joshi, A New Efficient Approach towards k-means Clustering Algorithm, In International Journal of Computer Applications, (0975-8887), vol. 65, no. 11, March (2013).

[9] Alan Jose, S. Ravi and M. Sambath, Brain Tumor Segmentation using K -means Clustering and Fuzzy C-means Algorithm and its Area Calculation. In International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, issue 2, March (2014).

[10] Madhu Yedla, Srinivasa Rao Pathakota and T. M. Srinivasa , Enhanced K -means Clustering Algorithm with Improved Initial Center, In International Journal of Science and Information Technologies, vol. 1(2), pp. 121-125, (2010).
[11] K. A. Abdul Nazeer and M. P. Sebastian, Improving the Accuracy and Efficiency of the k-means Clustering Algorithm, In Proceedings of the World Congress on Engineering, London, WCE, vol. 1, July (2001).
[12] A. N. Aimi Salihah, M.Y. Mashor, N.H. Harun and H. Rosline, Colour Image Enhancement Technique for Acute Leukaemia Blood Cell Morphological Feature, In IEEE International Conference on System, Man and Cybernatic, pp. 3677-3682, (2010).

CODE

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import imutils
clusters = 10 # try changing it                #////////////////////////////////////////
img = cv2.imread('C://Users//ASUS//Desktop//AI Project//tejas1.jfif')
org_img = img.copy()
print('Org image shape --> ',img.shape)
img = imutils.resize(img,height=200)
print('After resizing shape --> ',img.shape)
flat_img = np.reshape(img,(-1,3))
print('After Flattening shape --> ',flat_img.shape)
kmeans = KMeans(n_clusters=clusters,random_state=0)
kmeans.fit(flat_img)
dominant_colors = np.array(kmeans.cluster_centers_,dtype='uint')
percentages = (np.unique(kmeans.labels_,return_counts=True)[1])/flat_img.shape[0]
p_and_c = zip(percentages,dominant_colors)
p_and_c = sorted(p_and_c,reverse=True)
block = np.ones((50,50,3),dtype='uint')
plt.figure(figsize=(12,8))
for i in range(clusters):
    plt.subplot(1,clusters,i+1)
    block[:] = p_and_c[i][1][::-1] # we have done this to convert bgr(opencv) to rgb(matplotlib)
    plt.imshow(block)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel(str(round(p_and_c[i][0]*100,2))+'%')
bar = np.ones((50,500,3),dtype='uint')
plt.figure(figsize=(12,8))
plt.title('Proportions of colors in the image')
```

```
start = 0
i = 1
for p,c in p_and_c:
    end = start+int(p*bar.shape[1])
    if i==clusters:
        bar[:,start:] = c[::-1]
    else:
        bar[:,start:end] = c[::-1]
    start = end
    i+=1
plt.imshow(bar)
plt.xticks([])
plt.yticks([])
rows = 1000
cols = int((org_img.shape[0]/org_img.shape[1])*rows)
img = cv2.resize(org_img,dsize=(rows,cols),interpolation=cv2.INTER_LINEAR)
copy = img.copy()
cv2.rectangle(copy,(rows//2-250,cols//2-90),(rows//2+250,cols//2+110),(255,255,255),-1)
final = cv2.addWeighted(img,0.1,copy,0.9,0)
cv2.putText(final,'Most Dominant Colors in the Image',(rows//2-230,cols//2-
40),cv2.FONT_HERSHEY_DUPLEX,0.8,(0,0,0),1,cv2.LINE_AA)
start = rows//2-220
for i in range(5):      #////////////////////////////////////////
    end = start+70
    final[cols//2:cols//2+70,start:end] = p_and_c[i][1]
     cv2.putText(final,str(i+1),(start+25,cols//2+45),cv2.FONT_HERSHEY_DUPLEX,1,(255,255,255),1,cv2.
LINE_AA)
    start = end+20
plt.show()
cv2.imshow('img',final)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('output.png',final)
```