**Finding and Using Images' Dominant Colors using python &OpenCv**

**END TERM REPORT**

***BY***



**Department of Intelligent Systems**

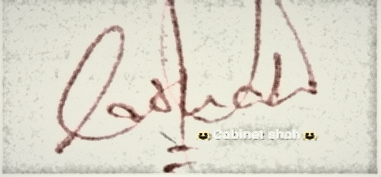
**School of Computer Science Engineering**

**Lovely Professional University**

04-2020

**Student Declaration**

This is to declare that this report has been written by cabinet kumar shah. No part of the report is copied from other sources. All information including from other sources have been duly acknowledged. I aver that if any part of the report is found to be copied, I shall take full responsibility for it.

****

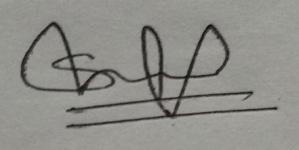
**Cabinet Kumar Shah**

**11812420** **B59**

sign.jpg

**Rajesh Durgam**

**11812522** **B55**



**Senate Shah**

**11812422** **B33**

Lovely professional university

BH-4 (Hostel)

Date : 06/04/2020

**TABLE OF CONTENTS**

**TITLE PAGE NO.**

**BONAFIDE CERTIFICATE 1**

**INTRODUCTION 2**

**AIM/OBJECTIVE OF THE PROJECT 2**

**K-MEANS CLUSTERING 2**

**PROJECT DETAILS 2-3**

**IMAGE RECOGNIZATION CODE 4**

**IMAGE RECOGNIZATION OUTPUT SAMPLE 5**

**WORK CONTRIBUTION DETAILS 6**

**PROJECT CONTRIBUTION CHART 6**

<https://github.com/cabinetshah2710/AI-Major-Project> ---link for project on github.

**BONAFIDE CERTIFICATE**

Certified that this project report “Finding and Using Images' Dominant Colors using python &Open CV” is the bonafidework of“Mr. cabinet kumar shah” who carried out the project work under my supervision.

Sagar Pande

Assistant Professor

23754

Artificial Intelligence

**INTRODUCTION**

When I started to learn image recognition, this was the first real project I have ever done, and it was fun. Therefore, I decided to start my first machine learning post with this project. The project aims to find the dominant color of an image. But you must be thinking, why would we need this?

Consider that you need to summarize the most important moments of a football match and you are a computer scientist, how would you solve that problem? If you like to watch football you may prefer to watch and decide which moments are most important by yourself. But what if you don’t like football? :) Yes, I don’t like football and I’m a computer scientist. Yay! All I have to do is to find the dominant color for each frame of video. And you can basically presume that if the moment is important you camera will leave the playground and focus to football player. So, if green color is dominant it means that it’s not the important frame and you can eliminate it. Of course, it may not be that simple but this project should be a basic project to learn K-Means Clustering.

In this project, I have written code using python3 with OpenCV. OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. I used k-means clustering to cluster the colors instead of using histogram for each color of pixel.

**AIM OF THE PROJECT**

In this project major aim is to find dominant color in picture via opencv with python lang. To be able to find dominant color I’ve used K-means clustering instead of try to find histogram for each pixel. In dominant\_color.py I’ve just used numpy and sklearn libraries for both histogram and K-means. I didn’t use library; wrote K-means and Histogram code by manually instead in dominant\_color\_man.py.

**K-Means Clustering:**

K-Means Clustering is an unsupervised learning method. If data set samples have labels, we prefer to use supervised method, but in real world mostly we don’t have labels and that’s why we prefer clustering methods which are known as unsupervised methods. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data.
2. Labels for the training data (each data point is assigned to a single cluster)

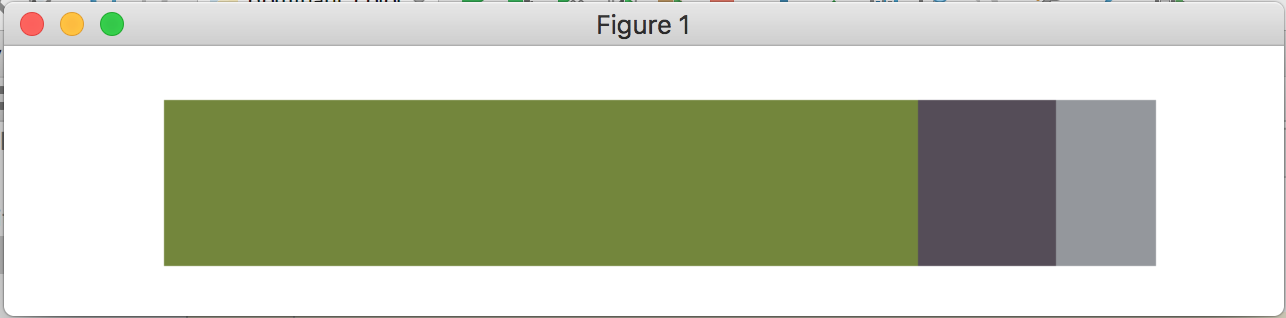
Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have been formed organically. The “Choosing K” section below describes how the number of groups can be determined.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

**Project Details**

As, I have mentioned previously I used OpenCV library for image processing with Python3 for this project. The other libraries that I have used NumPy for numerical arrays, Matplotlib for visualizing my result, Sklearn for machine learning.

First we will read image data using **cv2.imread()** function from OpenCV as cv2. Once the image is read using cv2, our image color channel comes to us as **Blue-Green-Red**. But we want **Red-Green-Blue** as our image color channel, so we convert it to the required channel using **cv2.cvtcolor()** function. Now we have 3-D parameters in our image data: row number X column number X colour channel number. But, we won’t need row and column information separately. Besides, it is hard to deal with **3-D Matrix** that’s why we **reshape()** image and make it **2-D Matrix** data. This was the preparation part for images and now we are ready to go on with clustering. Since we will import K-Means from top of our code, we can easily use it by only giving n\_clusters, which represents cluster number originally. After that we will use **fit()** function to apply K-Means Clustering algorithm on our pre-processed image data and result will come back to **clt**objects. We use**find\_histogram()** function to limit the number of histograms to the desired number of clusters. You can find the full details about**find\_histogram()** and **plot\_colors2()**function in this article below. As we don’t want to find histogram for all pixel, and all the color palette, so we would like to limit it to the desired number of clusters. The **plot\_colors2()** function will prepare a bar, and put colors on this bar as shown in the image below:



A **histogram** is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. When we come to **find\_histogram()** function it takes one parameter which is trained clt object and returns histogram information. **np. arange(0, len(np.unique(clt.labels\_))+1)** function will return evenly spaced values within a given interval. For our example, because we have 3 clusters numLabels’ value will be [0 1 2 3]. After that, we will calculate the histogram according to numberLabels and cluster’s label values. **clt.labels\_** is an array which includes cluster labels for each pixel. After calculating the histogram value, we will calculate the ratio of the histogram values we obtained because it is more difficult to work with numbers than to work with ratios.

After calculating the histogram ratios, we will jump in to **plot\_colors2()**function. This function will plot a bar which shows colours according to clusters and histogram. First we prepare a 3-D matrix, all values are zero. Then we plot the relative percentage of each cluster. **cv2.rectangle()**function draws a simple, thick, or filled up-right rectangle. We are filling a rectangle with colors in terms of histogram ratios.

**IMAGE RECOGNIZATION CODE:**

import cv2

importnumpy as np

importmatplotlib.pyplot as plt

fromsklearn.cluster import KMeans

deffind\_histogram(clt):

"""

create a histogram with k clusters

:param: clt

:return:hist

"""

numLabels = np.arange(0, len(np.unique(clt.labels\_)) + 1)

(hist, \_) = np.histogram(clt.labels\_, bins=numLabels)

hist = hist.astype("float")

hist /= hist.sum()

returnhist

def plot\_colors2(hist, centroids):

bar = np.zeros((50, 300, 3), dtype="uint8")

startX = 0

for (percent, color) in zip(hist, centroids):

# plot the relative percentage of each cluster

endX = startX + (percent \* 300)

cv2.rectangle(bar, (int(startX), 0), (int(endX), 50),

color.astype("uint8").tolist(), -1)

startX = endX

# return the bar chart

return bar

img = cv2.imread("pic/img7.jpeg")

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img = img.reshape((img.shape[0] \* img.shape[1],3)) #represent as row\*column,channel number

clt = KMeans(n\_clusters=3) #cluster number

clt.fit(img)

hist = find\_histogram(clt)

bar = plot\_colors2(hist, clt.cluster\_centers\_)

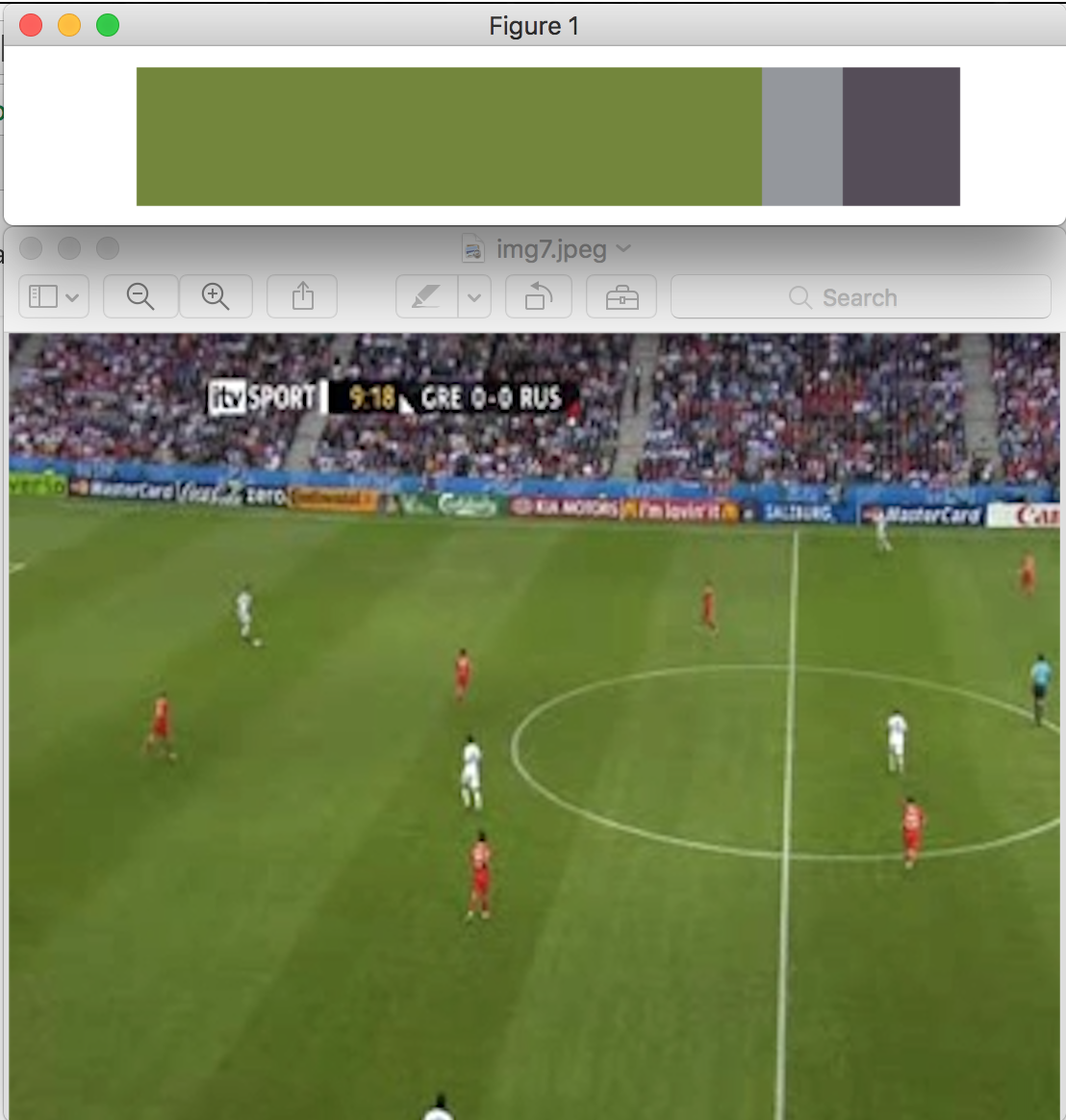
plt.axis("off")

plt.imshow(bar)

plt.show()

**IMAGE RECOGNIZATION OUTPUT SAMPLE:**

In the end, we prepare an image processing application which aims to find the dominant color for an image. We have used K-Means Clustering and calculate histogram to find the following result:



**WORK CONTRIBUTION:**

**Cabinet Shah:-**Worked on K-Means, Worked with Matplotlib library, created Means\_Clustering(), Random\_initialise\_cluster(), created a class named:- KMeansClustering(), k\_means(),Synchronised or combined all the code crerated individually.

**Senate Shah:-**worked on Histogram, Worked with Numpy library, created Update\_centroid(), find\_clusters(),k\_means\_cluster().

**Rajesh Durgam:-**Createdmodules : Find\_histogram(), defPlot\_colors(), Plot\_colors2(), calculate\_euclidean\_distance(), Worked with Numpy library, prepared the final report on project.

**PROJECT CONTRIBUTION CHART:**