PROJECT 3: COLOR DETECTION IN IMAGES

1. Introduction:

All pictures are filled with beautiful, mesmerizing colors. Most of the color names are unknown to ask, we assign them in the form of groups like all shades of blue gets summed into a category of blue, similarly with red and green. Some of us, would like to actually know the exact color names so this tool helps identify those colors. It is used to recognize objects, it is also used as a tool in various image editing and drawing apps. Color detection is the process of detecting the name of any color, distinguishing is easy for humans, whereas it isn't as straightforward for computers.

Human eyes and brains work together to translate light into color. Light receptors that are present in our eyes transmit the signal to the brain. Our brain then recognizes the color. Since childhood, we have mapped certain lights with their color names. We will be using the somewhat same strategy to detect color names.

In this project, we are going to build an application through which you can automatically get the name of the color just by clicking on them. So for this, we will have a data file that contains the color name and its R, G, B (i.e. Red, Green, and Blue) values. Then we will calculate the distance from each color and find the shortest one which will be used to get the almost accurate color name.

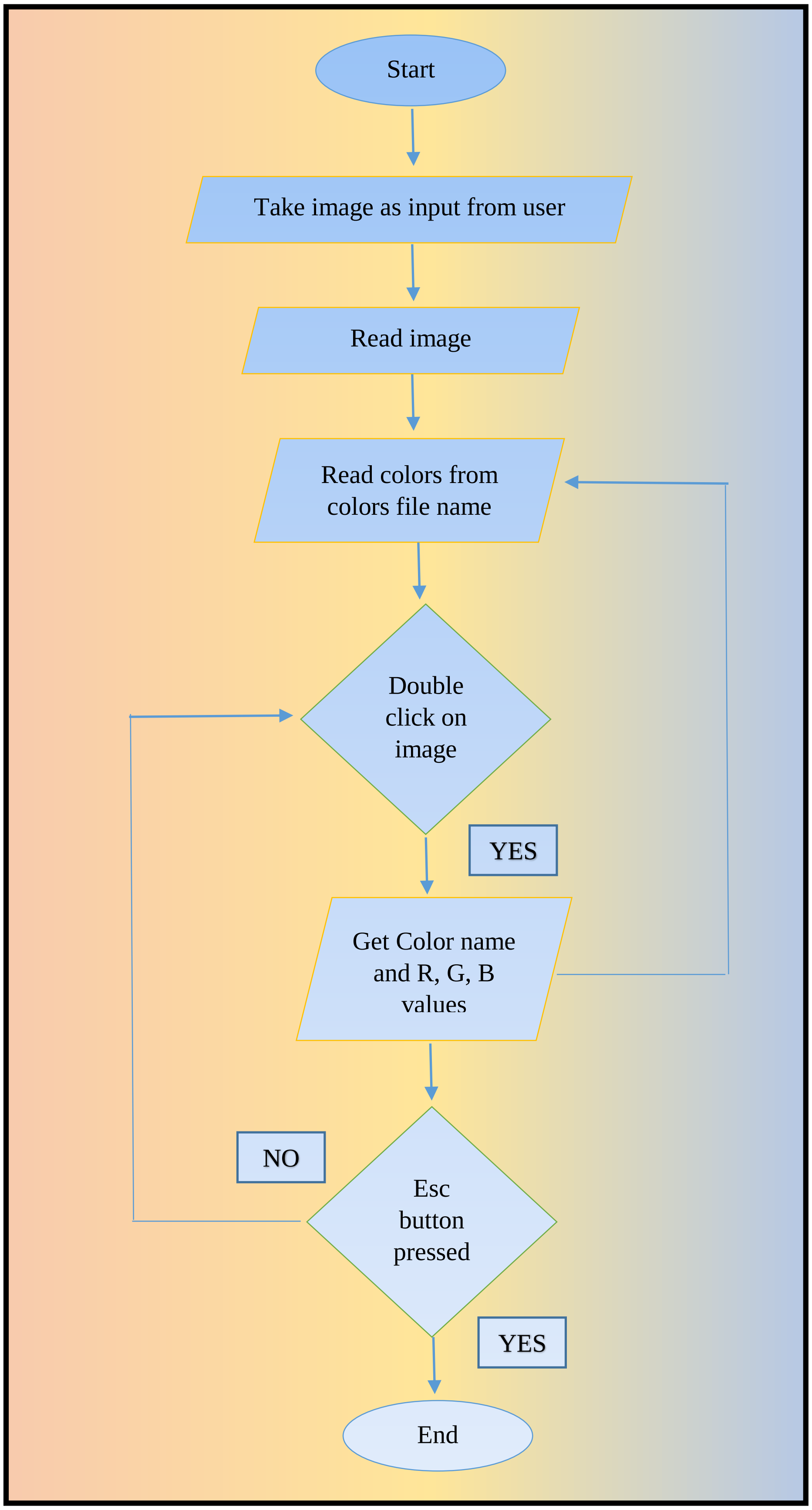
The Dataset

Colors are made up of 3 primary colors; red, green, and blue. In computers, we define each color value within a range of 0 to 255. So we can define a color is 256*256*256 = 16,581,375 ways. There are approximately 16.5 million different ways to represent a color. In our dataset, we need to map each color's values with their corresponding names. But don't worry, we don't need to map all the values. We will be using a dataset that contains RGB values with their corresponding names. The CSV file for our dataset has been taken from this link: https://github.com/codebrainz/color-names/blob/master/output/colors.csv

2. Technology Used:

OpenCV, Pandas, and numpy are the Python packages that were used for this project
OpenCV (Open Source Computer Vision) is a library for computer vision that includes numerous highly optimized algorithms. The library has more than 2500 algorithms and is capable of processing images and videos to detect faces, identify objects, classify human actions, and track moving objects, color detection, pattern recognition and many more.
Pandas, is used to handle and manage the data using data frames which makes it easy to manipulate the data in various ways, it also helps in Alignment and indexing of the data, and various other tasks.
When working with images, and OpenCV, the images are stored in a **numpy** ndarray, it helps in getting the image size, dimensions, and the number of channels for each pixels, height, width and other information about the image, which is used in the program to understand the image and get the color information.

3. Flowchart:

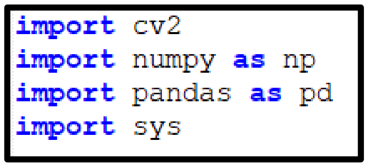

Start
Take image as input from user
Read image
Read colors from colors file name
Double click on image
YES
Get Color name and R, G, B values
NO
Esc button pressed
YES
End

4. Code for the project:

https://github.com/mokshadashah/color_detection

5. Explanation of the code:

a. Import all required libraries to run the program:

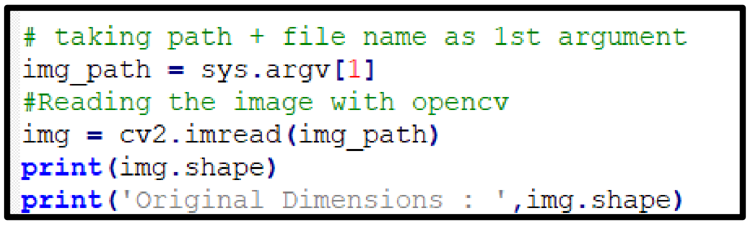


```
import cv2
import numpy as np
import pandas as pd
import sys
```

b. Taking image path and image name as input from user:

We create an argument parser to take the path of the image and the name of image from the command line. Such as: python file_name.py <path_to_image_and_image_name_with_its_extension>

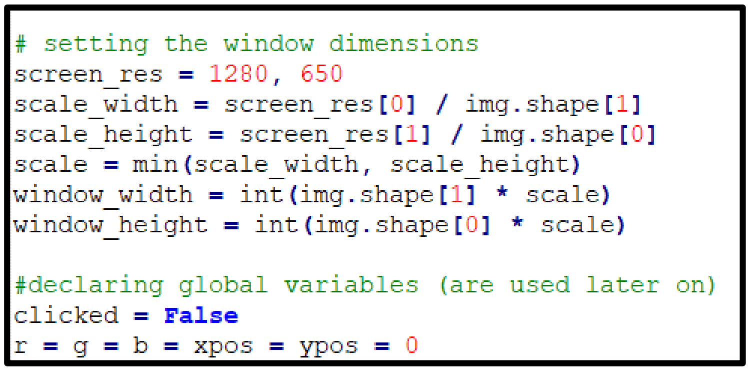
And then reading the image from its path.



```
# taking path + file name as 1st argument
img_path = sys.argv[1]
#Reading the image with opencv
img = cv2.imread(img_path)
print(img.shape)
print('Original Dimensions : ',img.shape)
```

c. Setting the window dimensions and scale:

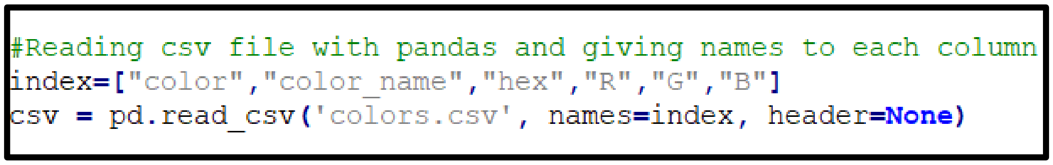
The window needs to match the image, so using the image dimensions we adjust the window height, width, and the scale of the window. Along with that, we declare global variable to use further in the program.



```
# setting the window dimensions
screen_res = 1280, 650
scale_width = screen_res[0] / img.shape[1]
scale_height = screen_res[1] / img.shape[0]
scale = min(scale_width, scale_height)
window_width = int(img.shape[1] * scale)
window_height = int(img.shape[0] * scale)

#declaring global variables (are used later on)
clicked = False
r = g = b = xpos = ypos = 0
```

d. **Reading colors.csv file and giving column names:**

Here, we read the colors.csv file and give headings to the columns to distinguish and use them ahead.



```
#Reading csv file with pandas and giving names to each column
index=["color","color_name","hex","R","G","B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

e. **Defining a function to get color names from the image:**

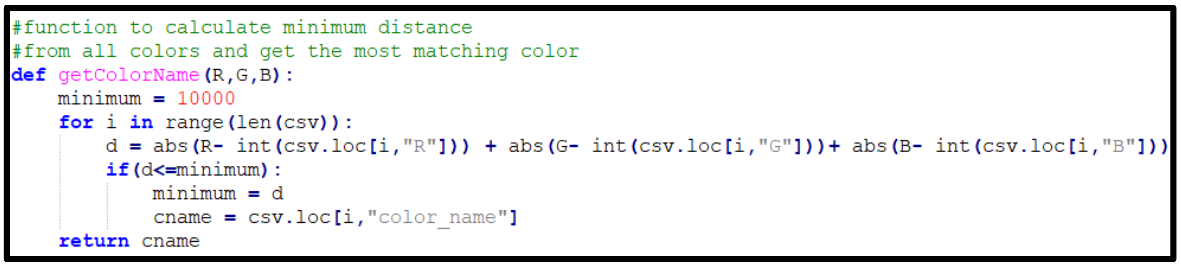
We create a function to retrieve the color names by calculating the minimum distance from the colors to get the most matching one.

Using the R, G, B values we find the distance and determine the color.

To get the color name, we calculate a distance (d) which tells us how close we are to color and choose the one having minimum distance.

Our distance is calculated by this formula:

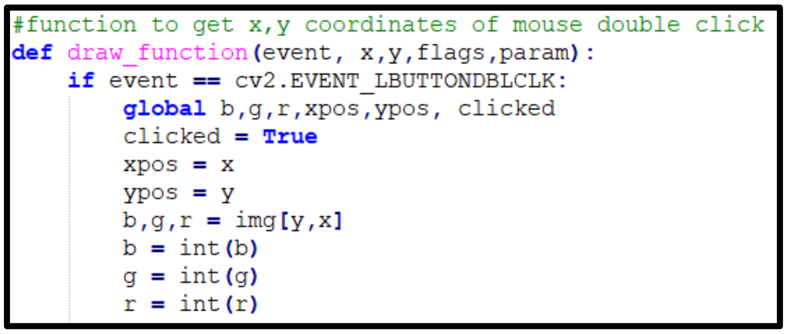
d = abs(Red – ithRedColor) + (Green – ithGreenColor) + (Blue – ithBlueColor



```
#function to calculate minimum distance
#from all colors and get the most matching color
def getColorName(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+ abs(B- int(csv.loc[i,"B"]))
        if(d<=minimum):
            minimum = d
            cname = csv.loc[i,"color_name"]
    return cname
```

f. **Defining a function to get the x and y coordinates on the screen:**

We, then create an event on which, when the mouse is double clicked gives the R, G and B integer value and then the above declared function finds the most minimum distance to it and decides the color name.

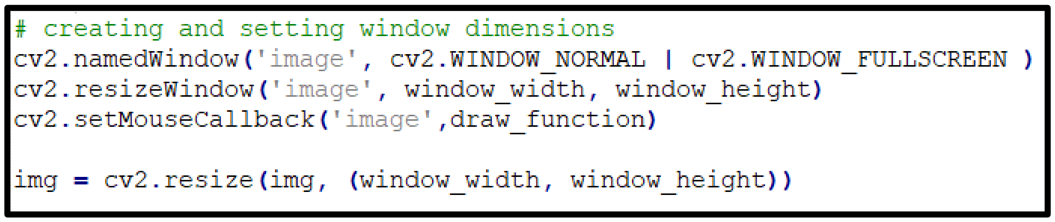


```
#function to get x,y coordinates of mouse double click
def draw_function(event, x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)
```

g. Creating and setting dimensions of the window to fit the image:

We give name to the window and declare its type and resize it according to the above declared window width and height.

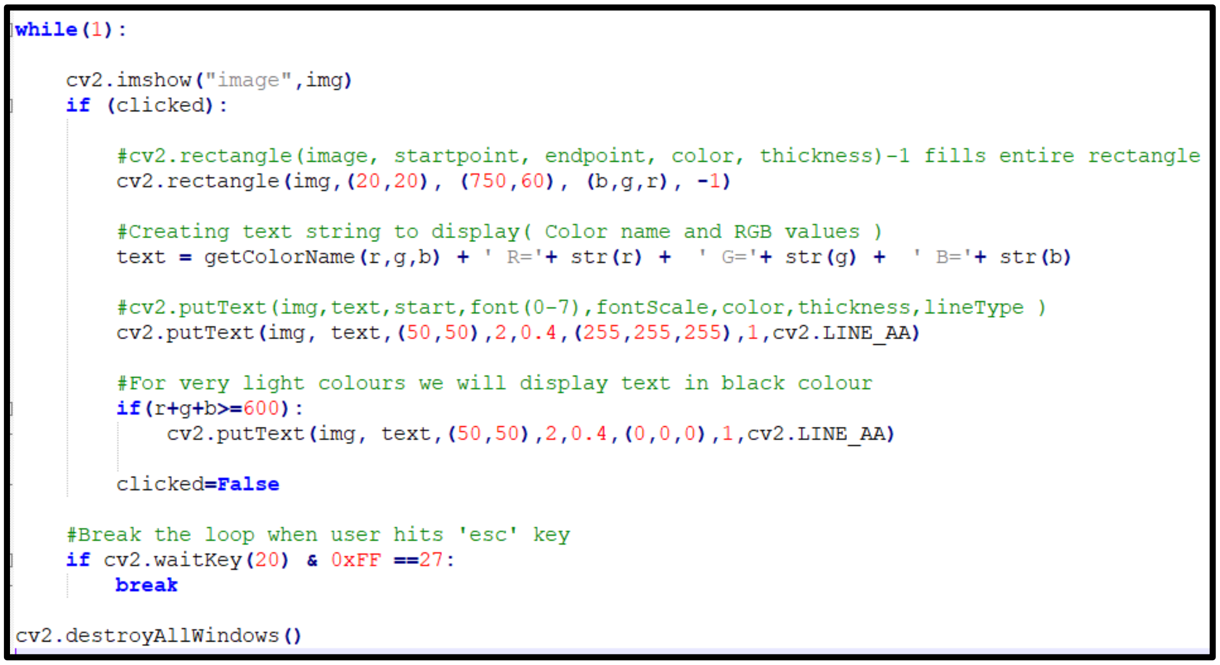
Also, we resize the image according to the window height and weight to match it.



```python
# creating and setting window dimensions
cv2.namedWindow('image', cv2.WINDOW_NORMAL | cv2.WINDOW_FULLSCREEN )
cv2.resizeWindow('image', window_width, window_height)
cv2.setMouseCallback('image',draw_function)

img = cv2.resize(img, (window_width, window_height))
```

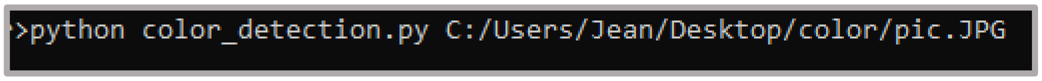
h. When the image is clicked, loop to get info about the color selected on the image:

While true, the image will be displayed and if double clicked, a rectangle is created, text is created and they are displayed, which shows the color name and the R, G, and B values. If the color is light, we display the text in a darker color so that the text is visible. And the same process is repeated for the next double click to check the color. The loop will break i.e. the window will only be excited when pressed the escape key.



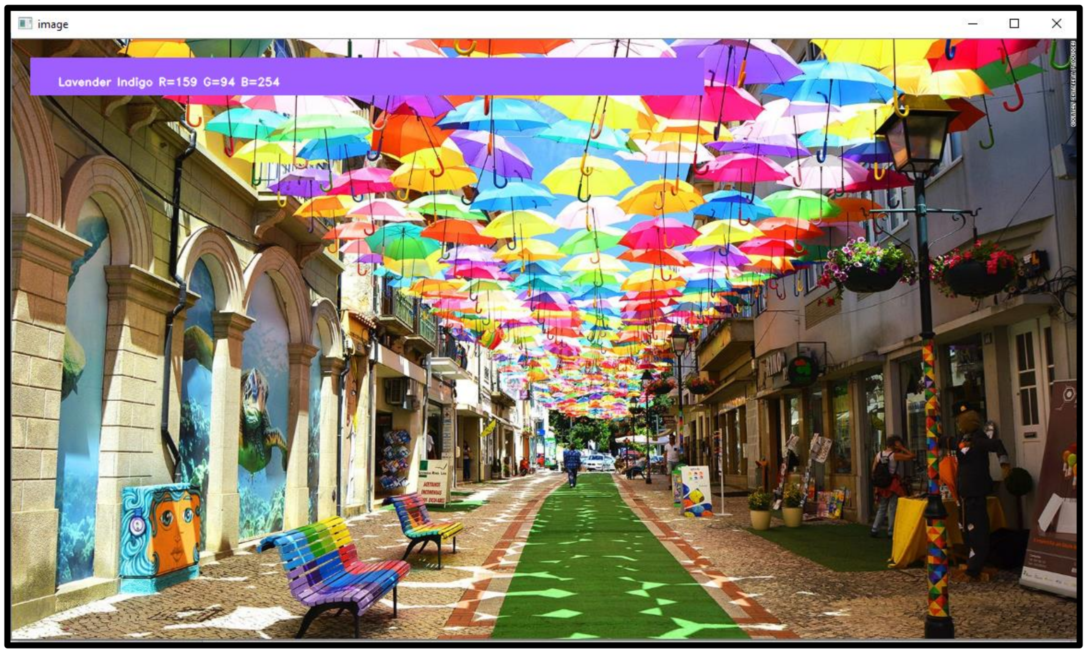
```python
while(1):

    cv2.imshow("image",img)
    if (clicked):

        #cv2.rectangle(image, startpoint, endpoint, color, thickness)-1 fills entire rectangle
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) +  ' G='+ str(g) +  ' B='+ str(b)

        #cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
        cv2.putText(img, text,(50,50),2,0.4,(255,255,255),1,cv2.LINE_AA)

        #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.4,(0,0,0),1,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

i. **Run the code:**

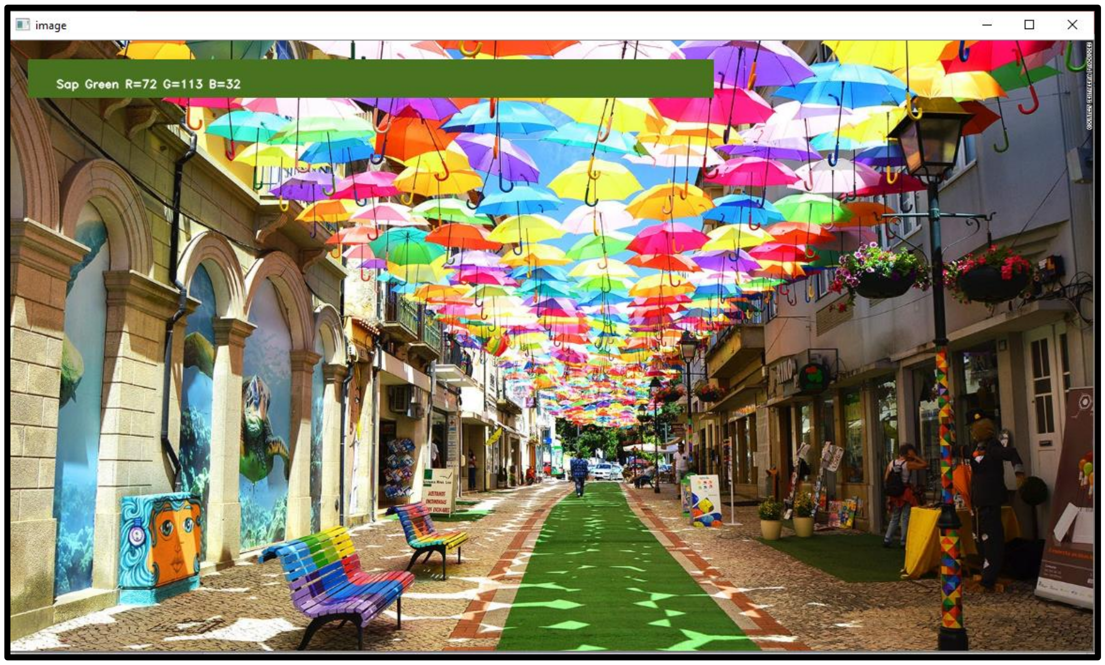


```
>python color_detection.py C:/Users/Jean/Desktop/color/pic.JPG
```

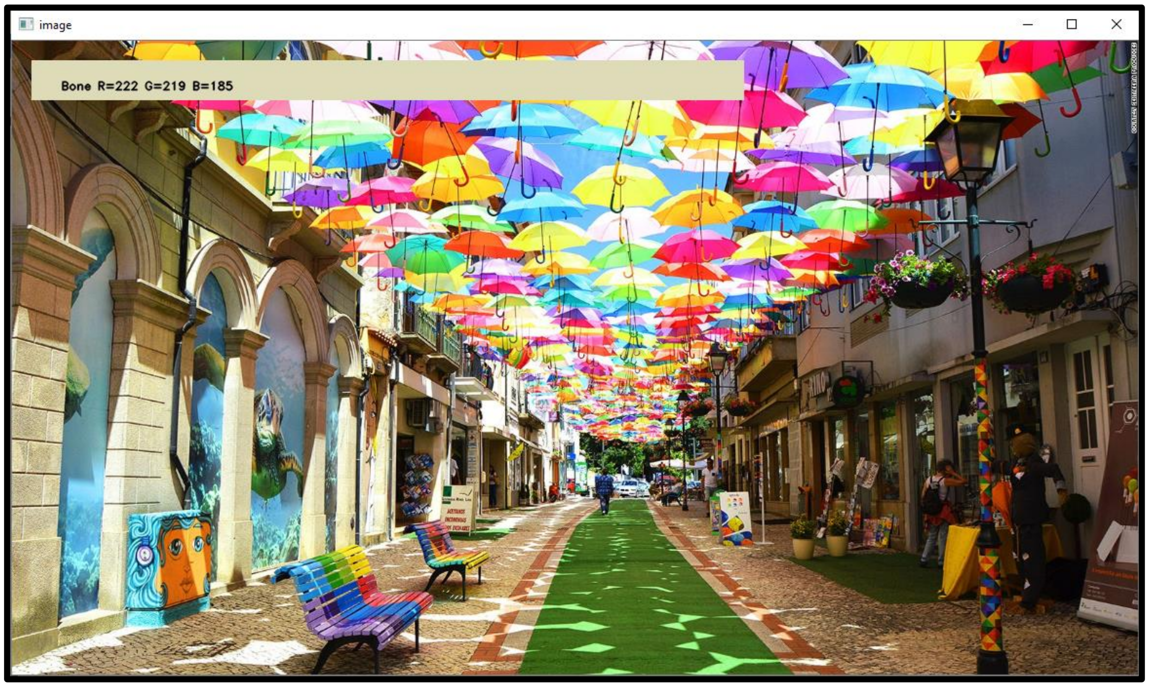
j. **Final Output:**



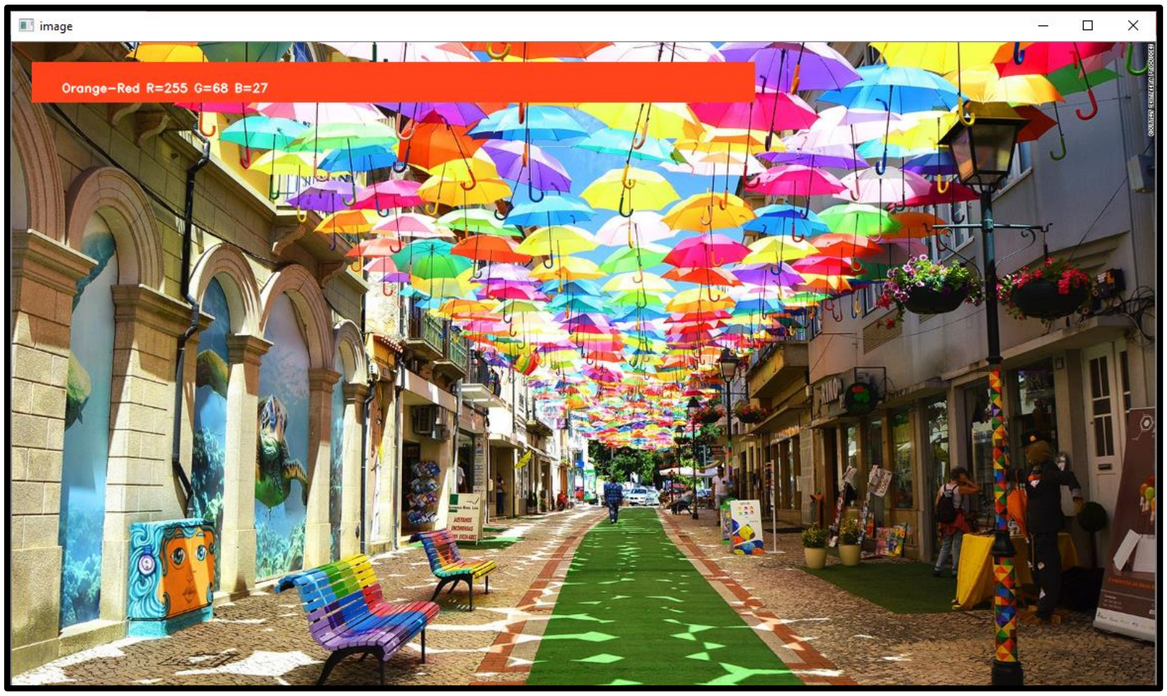
```
Lavender Indigo R=159 G=94 B=254
```



```
Sap Green R=72 G=113 B=32
```



```
Bone R=222 G=219 B=185
```



```
Orange-Red R=255 G=68 B=27
```