A Project Report

On

Colour Detection using Python

S.No	Enrollment Number	Admission Number	Student Name	Degree / Branch	Sem
1	18021011726	18SCSE1010216	UTTAM DUTTA	B.Tech/CSE	VII
2	18021011698	18SCSE1010470	ADARSH RAJ	B.Tech/CSE	VII

Under the Supervision of

Dr. Shrddha Sagar
Assistant Professor

GALGOTIAS
UNIVERSITY

School of Computing Science and Engineering
Greater Noida, Uttar Pradesh
Fall 2021 – 2022

CANDIDATE’S DECLARATION

We hereby certify that the work which is being presented in the project, entitled “**COLOR DETECTION USING OPENCV**” in partial fulfilment of the requirements for the award of the B.tech computer science submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **July 2021 to December 2021**, under the supervision of **Dr. Shrddha Sagar Assistant Professor**, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

UTTAM DUTTA 18SCSE1010498

ADARSH RAJ 18SCSE1010470

Supervisor Name

CERTIFICATE

The Final Project Viva-Voce examination of Uttam Dutta 18SCSE1010498 and Adarsh Raj 18SCSE1010470 has been held on _________________ and their work is recommended for the award of B.Tech in Computer Science Engineering.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date : December, 2021

Place : Greater Noida

TABLE OF CONTENTS

Abstract

Color detection is necessary to recognize objects, it is also used as a tool in various image editing and drawing apps. It is the process of detecting the name of any color. Well, for humans this is an extremely easy task but for computers, it is not straightforward. 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. Hence the problem that arises in front of us is how to make a computer understand or recognize colors , so we are going to solve this problem. So basically in this project using python we need 3 different parts to be used. Python code which will be recognizing color, Image that will be used for testing the color recognition, a .csv file that will be containing the colors

as dataset. Hence the above 3 modules will help us in achieving our aim that is detecting the colors in an image using python.

Literature Review

[1] Color can be identified from the sensory optic nerves of the eyes. Color can only be seen or identified when a source of light is applied to an object. Color blindness can be termed as inability of the differentiation between colors. It is incurable disease that can be termed as lifelong disease. Edges can be very helpful in color differentiation boundary.

[2] Color detection model can be used in mixing of colors especially in paints, dyes and color pigments. It can be also very helpful in to differentiating colors that are used in robotics and in other medical fields. It can also be used in Graphic Arts Industry. Other implementations can

also be used in agricultural industry like especially detection of quality of soil.

[3] Color Detection can be used in agriculture industry to find the weeds the along with the crops. Via color detection weeds can be identified and destroyed and the crops can be saved. It can be also used in medical industries to detect the disease and other disorders especially in face and other internal diseases like cancers.

[4] The main aim of computer vision is to analyze the behavior of human eye and the reduction of human effort. Through computer vision various task can be done that is done by human eye, whether to detect the object or identify its color. By this method it is very helpful to detect the symptoms of the disease and the other applications in other industries like agriculture.

Due to its powerful learning ability and advantages in dealing with occlusion, scale transformation and background switches, deep learning based object detection has been a research hotspot in recent years. This paper provides a detailed review on deep learning based object detection frameworks which handle different subproblems, such as occlusion, clutter and low resolution, with different degrees of modifications on R-CNN. The review starts on generic object detection pipelines which provide base architectures for other related tasks.

Then, three other common tasks, namely salient object detection, face detection and pedestrian detection, are also briefly reviewed. Finally, we propose several promising future directions to gain a thorough understanding of the object detection landscape.

This review is also meaningful for the developments in neural networks and related learning systems, which provides valuable insights and guidelines for future progress.

colour detection is the process of detecting the name of any color. Simple isn’t it? Well, for humans this is an extremely easy task but for computers, it is not straightforward.

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.

Problem Formulation

“It is said that a problem well defined is a problem half solved.”

So here we are talking about a very common disease we come across nowadays “Color Blindness”.

This is a genetic disease and cannot be easily cured. In this disease the person suffering with it cannot differentiate between colors. So with the help of technology .

we can help the person suffering from Color Blindness without physically operating or doing anything to the person’s eyes.

Now let us formulate it.

- Import the modules necessary in solving the problem.
- Take the image path of which you want to find the colors in.
- Read the csv file in the hex code of colors is stored.
- Now calculate the minimum distance from all the color and get the most matching color using the get_color_name function.
- Make a function to get the x, y coordinates of the click point of mouse.
- Display image on the window with a pointer to select any point on image.

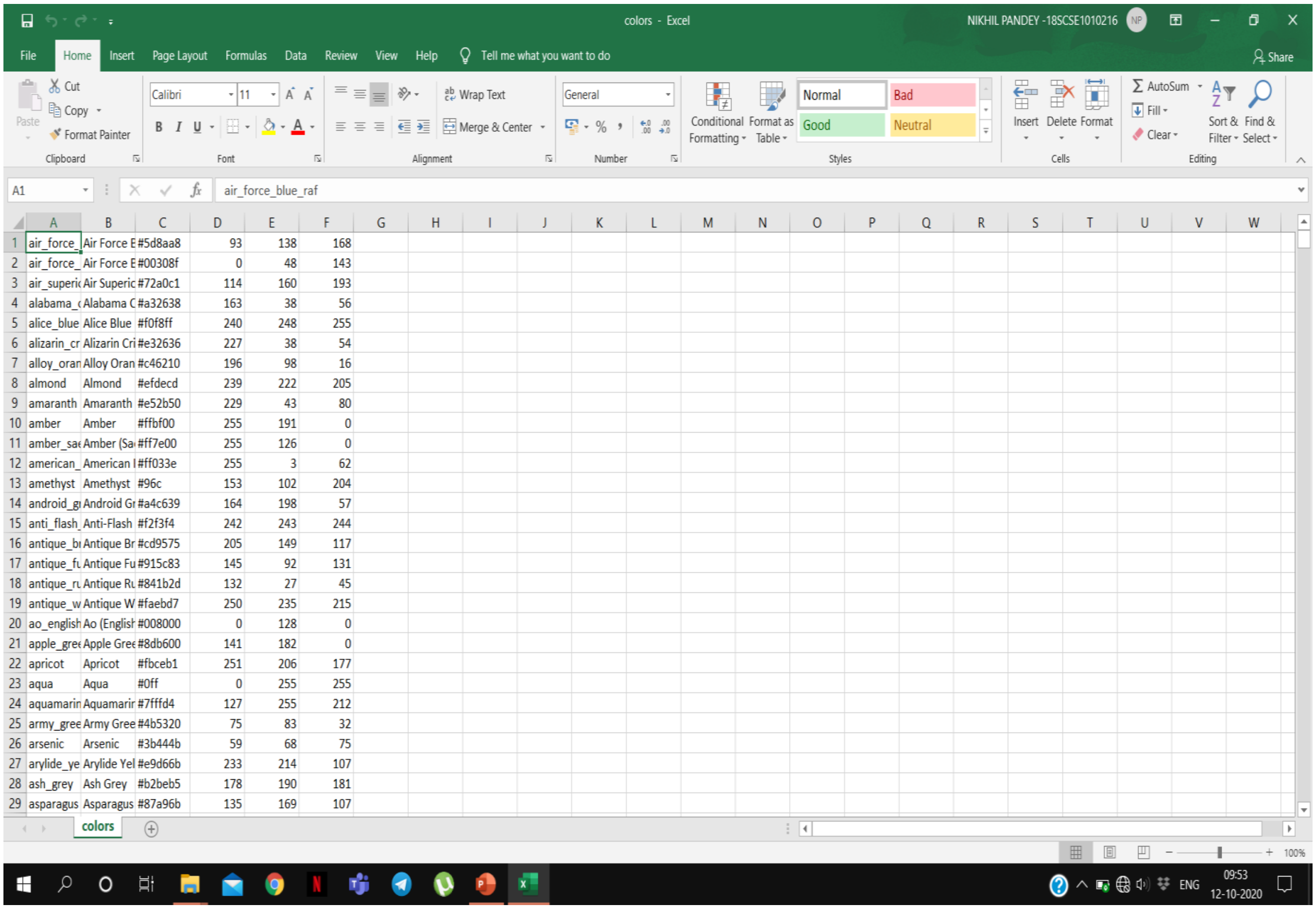
- Display the color name of the point along with the R, G, B values

Dataset of colors

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 in how many ways we can define a color? The answer is 256*256*256 = 16,581,375.

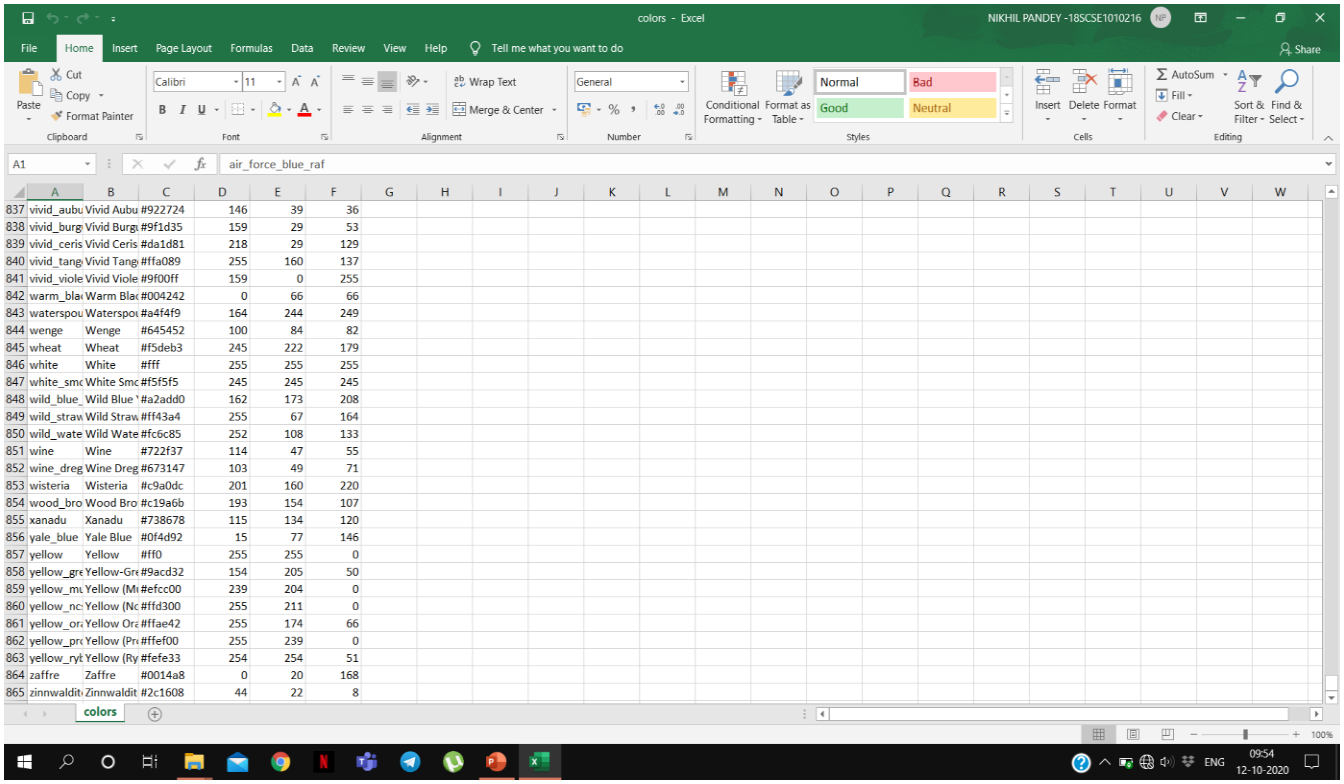
There are approximately 16.5 million different ways to represent a color. In our dataset, we need to map each color's values with their corresponding names. But don't worry, we don't need to map all the values.

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:



A1 | air_force_blue_raf

	A	B	C	D	E	F
1	air_force_	Air Force E	#5d8aa8	93	138	168
2	air_force_	Air Force E	#00308f	0	48	143
3	air_superi	Air Superic	#72a0c1	114	160	193
4	alabama_	Alabama C	#a32638	163	38	56
5	alice_blue	Alice Blue	#f0f8ff	240	248	255
6	alizarin_cr	Alizarin Cri	#e32636	227	38	54
7	alloy_oran	Alloy Oran	#c46210	196	98	16
8	almond	Almond	#efdecd	239	222	205
9	amaranth	Amaranth	#e52b50	229	43	80
10	amber	Amber	#ffbf00	255	191	0
11	amber_sa	Amber (Sa	#ff7e00	255	126	0
12	american_	American I	#ff033e	255	3	62
13	amethyst	Amethyst	#96c	153	102	204
14	android_g	Android Gr	#a4c639	164	198	57
15	anti_flash_	Anti-Flash	#f2f3f4	242	243	244
16	antique_b	Antique Br	#cd9575	205	149	117
17	antique_fu	Antique Fu	#915c83	145	92	131
18	antique_ru	Antique Ru	#841b2d	132	27	45
19	antique_w	Antique W	#faebd7	250	235	215
20	ao_english	Ao (English	#008000	0	128	0
21	apple_gre	Apple Gree	#8db600	141	182	0
22	apricot	Apricot	#fbceb1	251	206	177
23	aqua	Aqua	#0ff	0	255	255
24	aquamarin	Aquamarir	#7fffd4	127	255	212
25	army_gree	Army Gree	#4b5320	75	83	32
26	arsenic	Arsenic	#3b444b	59	68	75
27	arylide_ye	Arylide Yel	#e9d66b	233	214	107
28	ash_grey	Ash Grey	#b2beb5	178	190	181
29	asparagus	Asparagus	#87a96b	135	169	107



A1 | air_force_blue_raf

	A	B	C	D	E	F
837	vivid_aubu	Vivid Aubu	#922724	146	39	36
838	vivid_burg	Vivid Burgu	#9f1d35	159	29	53
839	vivid_ceris	Vivid Ceris	#da1d81	218	29	129
840	vivid_tang	Vivid Tang	#ffa089	255	160	137
841	vivid_viole	Vivid Viole	#9f00ff	159	0	255
842	warm_bla	Warm Bla	#004242	0	66	66
843	waterspou	Waterspou	#a4f4f9	164	244	249
844	wenge	Wenge	#645452	100	84	82
845	wheat	Wheat	#f5deb3	245	222	179
846	white	White	#fff	255	255	255
847	white_smo	White Smo	#f5f5f5	245	245	245
848	wild_blue_	Wild Blue '	#a2add0	162	173	208
849	wild_straw	Wild Straw	#ff43a4	255	67	164
850	wild_wate	Wild Wate	#fc6c85	252	108	133
851	wine	Wine	#722f37	114	47	55
852	wine_dreg	Wine Dreg	#673147	103	49	71
853	wisteria	Wisteria	#c9a0dc	201	160	220
854	wood_bro	Wood Bro	#c19a6b	193	154	107
855	xanadu	Xanadu	#738678	115	134	120
856	yale_blue	Yale Blue	#0f4d92	15	77	146
857	yellow	Yellow	#ff0	255	255	0
858	yellow_gre	Yellow-Gre	#9acd32	154	205	50
859	yellow_mu	Yellow (Mu	#efcc00	239	204	0
860	yellow_nc	Yellow (Nc	#ffd300	255	211	0
861	yellow_ora	Yellow Ora	#ffae42	255	174	66
862	yellow_pro	Yellow (Pro	#ffef00	255	239	0
863	yellow_ryb	Yellow (Ry	#fefe33	254	254	51
864	zaffre	Zaffre	#0014a8	0	20	168
865	zinnwaldit	Zinnwaldit	#2c1608	44	22	8

The above dataset contains 865 colors along with their R, G, B values we will be using this dataset only to get the colors for the mouse clicks.

Calculating Distance

Now as we have already shown in the screenshot that we have the R, G, B values of the colors that we have attached in the dataset.

Explanation of Code:

- **Camera Settings:** In order to perform runtime operations, the device’s web-camera is used. To capture a video, we need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. The device index is just the number to specify which camera. Normally one camera will be connected, so we simply pass 0. You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But in the end, don't forget to release the capture. Moreover, if anyone wants to apply this colour detection technique on any image it can be done with little modifications in the code which I'll discuss later.

- **Capturing frames:** The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program.
  After capturing the live stream frame by frame we are converting each frame in BGR color space(the default one) to HSV color space. There are more than 150 color-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV. For color conversion, we use the function cv2.cvtColor(input_image, flag) where flag determines the type of conversion. For BGR to HSV, we use the flag cv2.COLOR_BGR2HSV. Now we know how to convert BGR images to HSV, we can use this to extract a colored object. In HSV, it is more easier to represent a color than RGB color-space.
  In specifying the range , we have specified the range of blue color. Whereas you can enter the range of any colour you wish.

- **Masking technique:** The mask is basically creating some specific region of the image following certain rules. Here we are creating a mask that comprises of an object in blue color. After that, I have used a bitwise_and on the input image and the threshold image so that only the blue coloured objects are

highlighted and stored in res.
We then display the frame, res, and mask on 3 separate windows using imshow function.

- **Display the frame:** As imshow() is a function of HighGui it is required to call waitKey regularly, in order to process its event loop.
  The function waitKey() waits for key event for a “delay” (here, 5 milliseconds). If you don’t call waitKey, HighGui cannot process windows events like redraw, resizing, input event etc. So just call it, even with a 1ms delay .

- **Summarizing the process**:
  1. Take each frame of the video.
  2. Convert each frame from BGR to HSV color-space.
  3. Threshold the HSV image for a range of blue color.

We need a function in the program that is going to return the color of the point where the mouse is clicked. And as we know for getting the color we need to get the distance and then compare it with the dataset.

Calculating the distance is done by the formulae given below :-

D= abs (Red-ithRedColor) + abs (Green-ithGreenColor) + abs (Blue-ithBlueColor)

(ithRedColor, ithGreenColor, ithBlueColor are the colors R,G,B values from the dataset)

Required tools

OpenCV:

OpenCV (Open Source Computer Vision) library aims at real time Computer Vision. It is mainly used to do all the operations related to images.

Pandas:

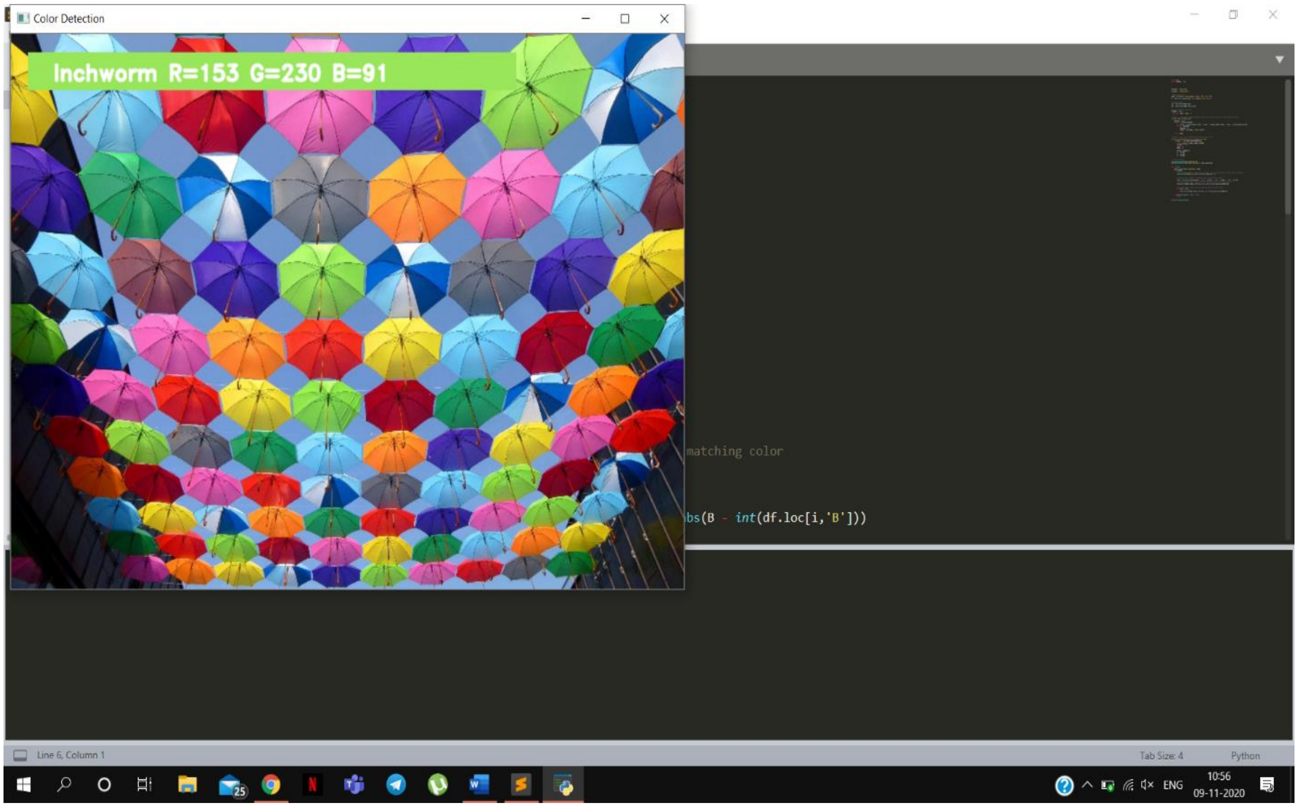
Pandas is Python Package which stands for Python and data analysis. This library helps in data manipulation and analysis.

Image Processing:

Image Processing technique is used to perform some certain operations on an image, in order to get an enhanced image as an output or to extract some useful information from the image.

It acts as a type signal processing in which input is an image and output may be an image or characteristics/features associated with that image (we have used it for resizing the image) .

Output

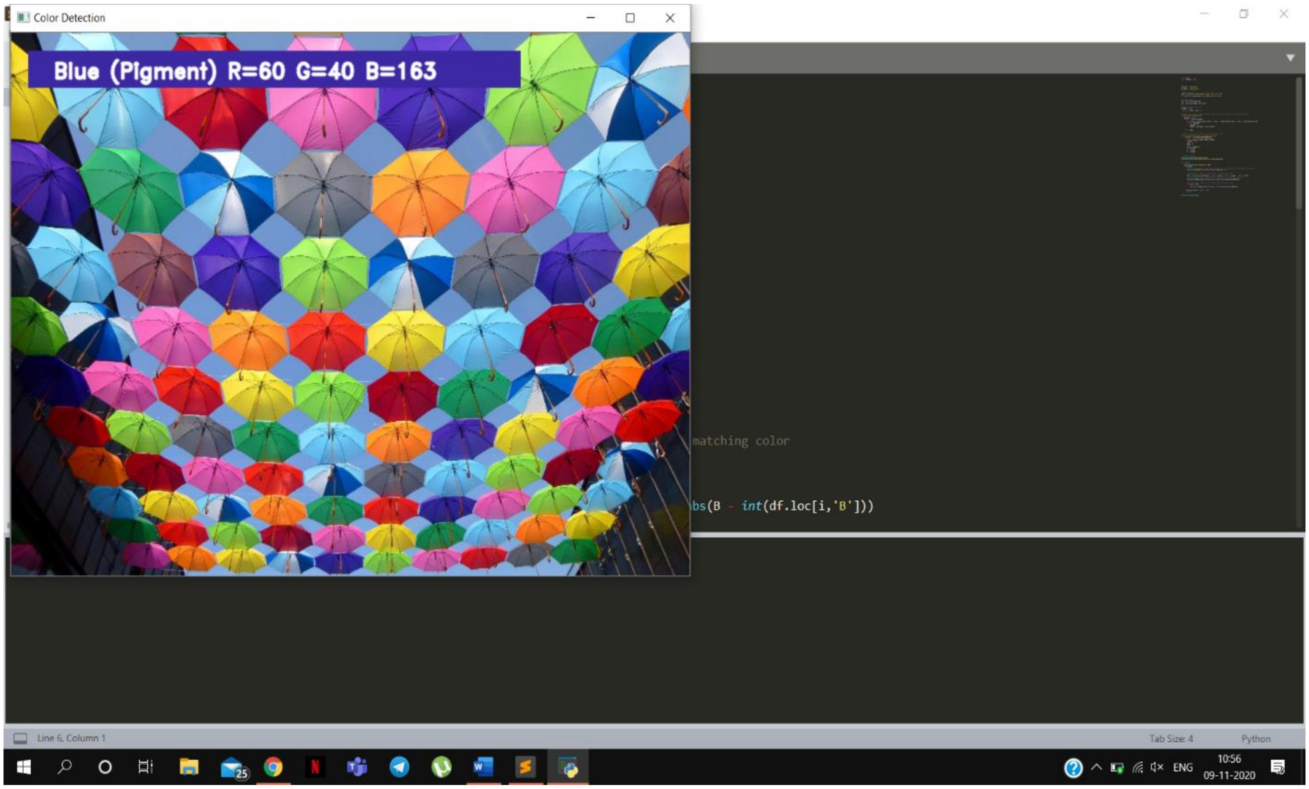


Color Detection

```
Inchworm R=153 G=230 B=91
```

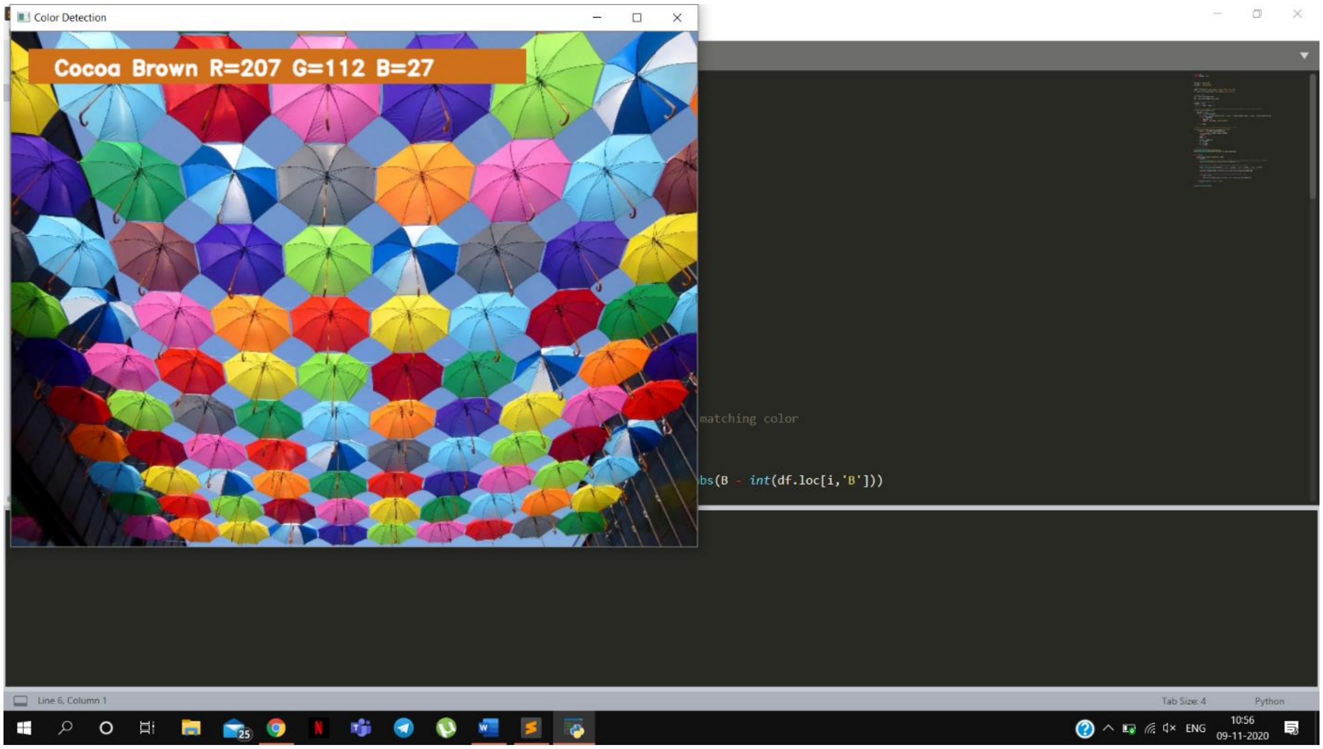
```
matching color

bs(B - int(df.loc[i,'B']))
```



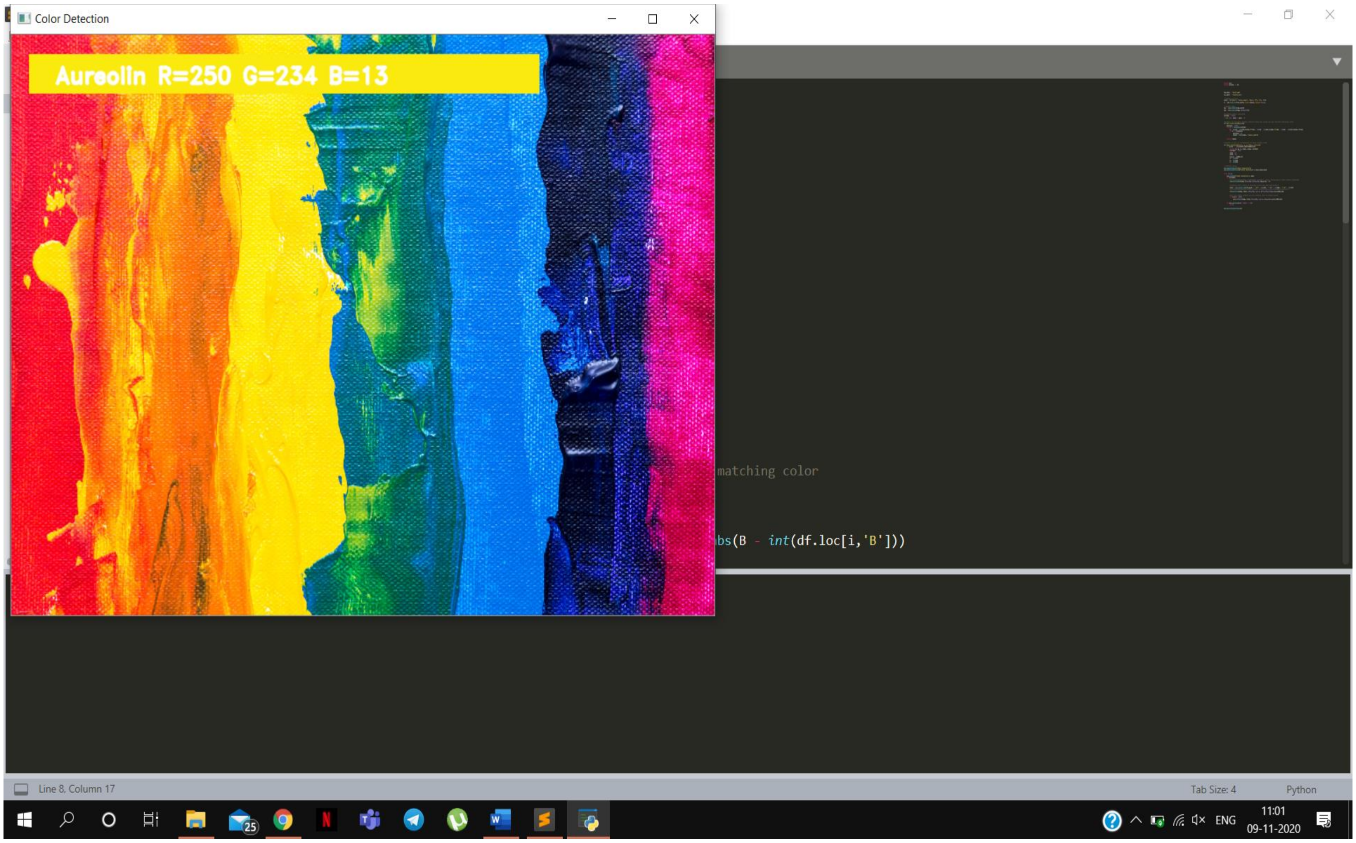
```
Blue (Pigment) R=60 G=40 B=163
```

```
matching color
bs(B - int(df.loc[i,'B']))
```



```
Cocoa Brown R=207 G=112 B=27
```

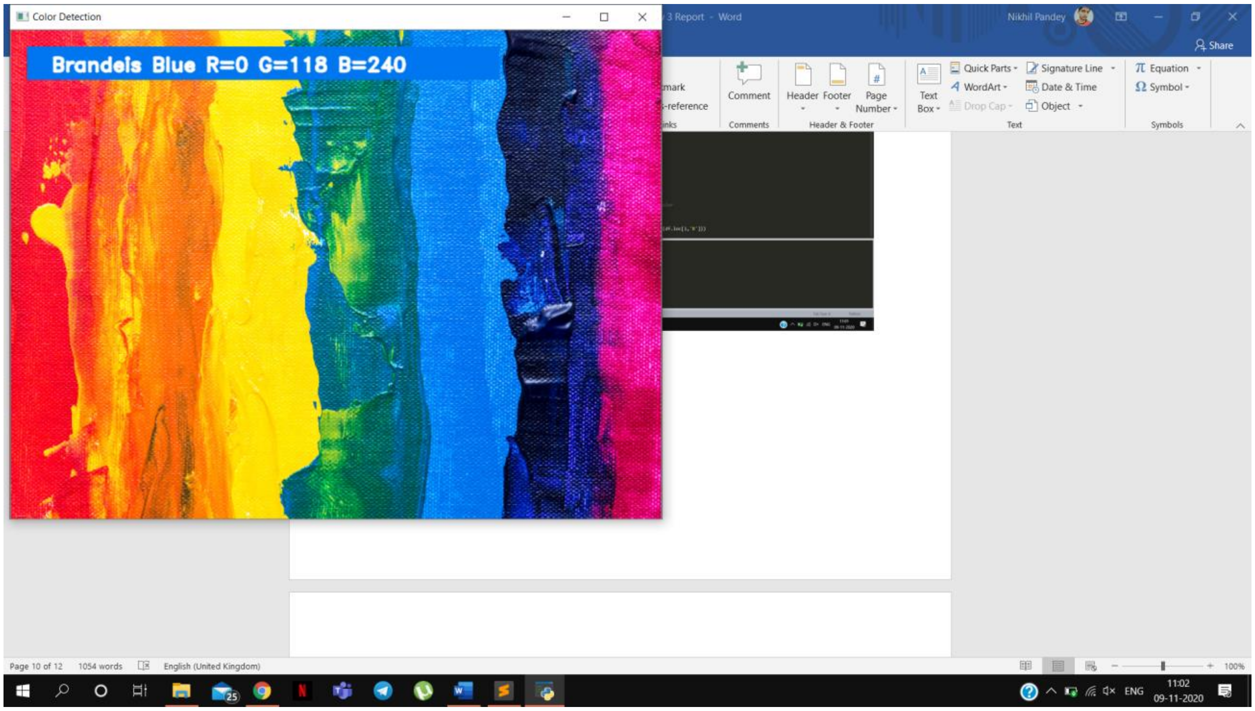
```
matching color
bs(B - int(df.loc[i,'B']))
```



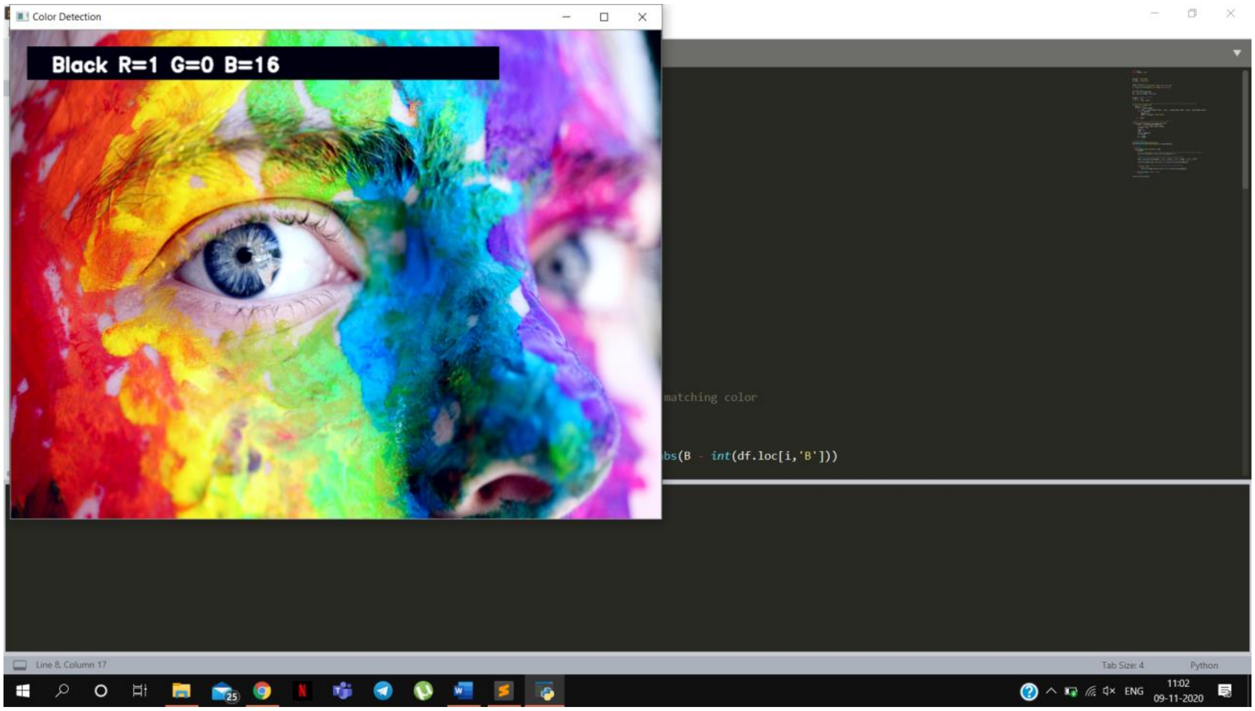
Color Detection

```
Aureolin R=250 G=234 B=13
```

```
matching color
bs(B - int(df.loc[i,'B']))
```

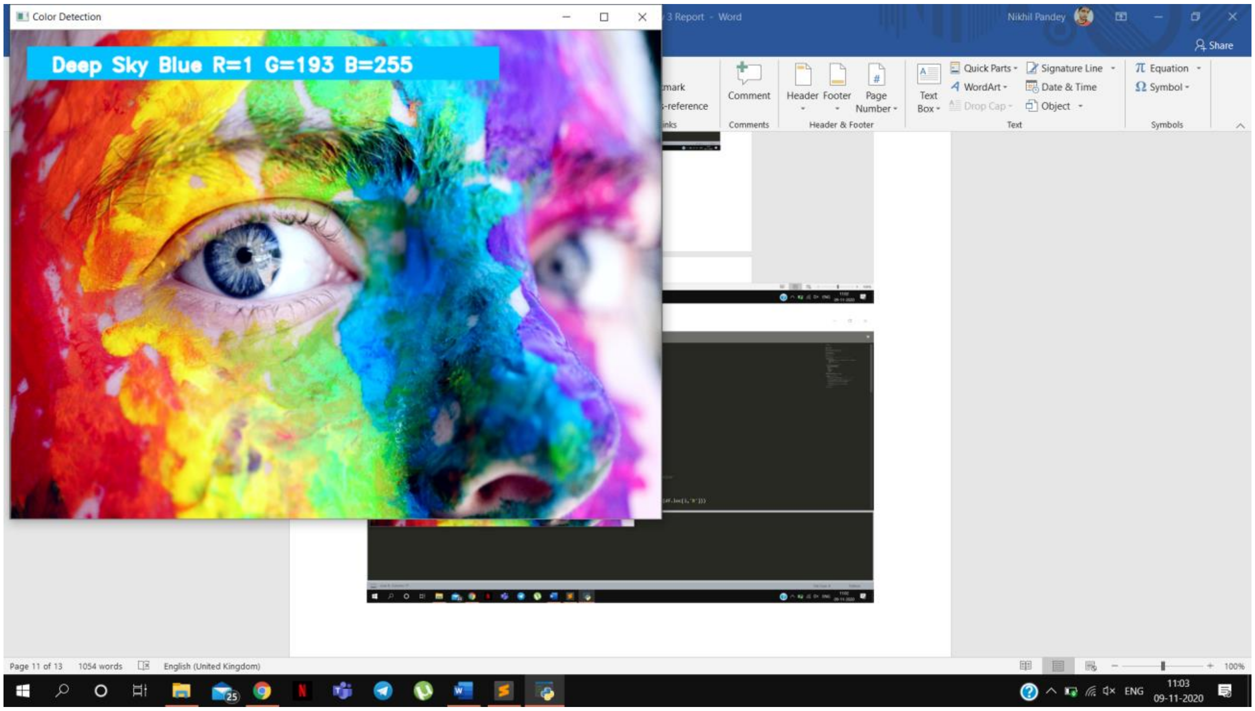


Brandeis Blue R=0 G=118 B=240

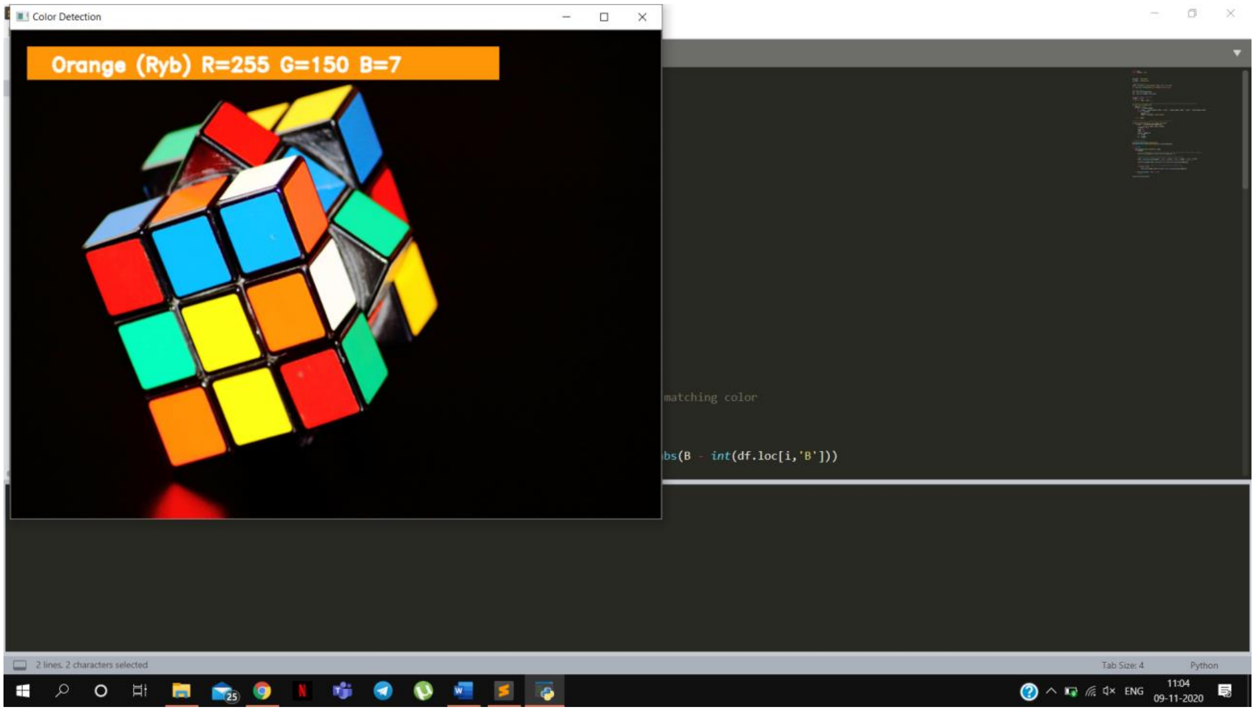


Black R=1 G=0 B=16

```
matching color
bs(B - int(df.loc[i,'B']))
```



Deep Sky Blue R=1 G=193 B=255

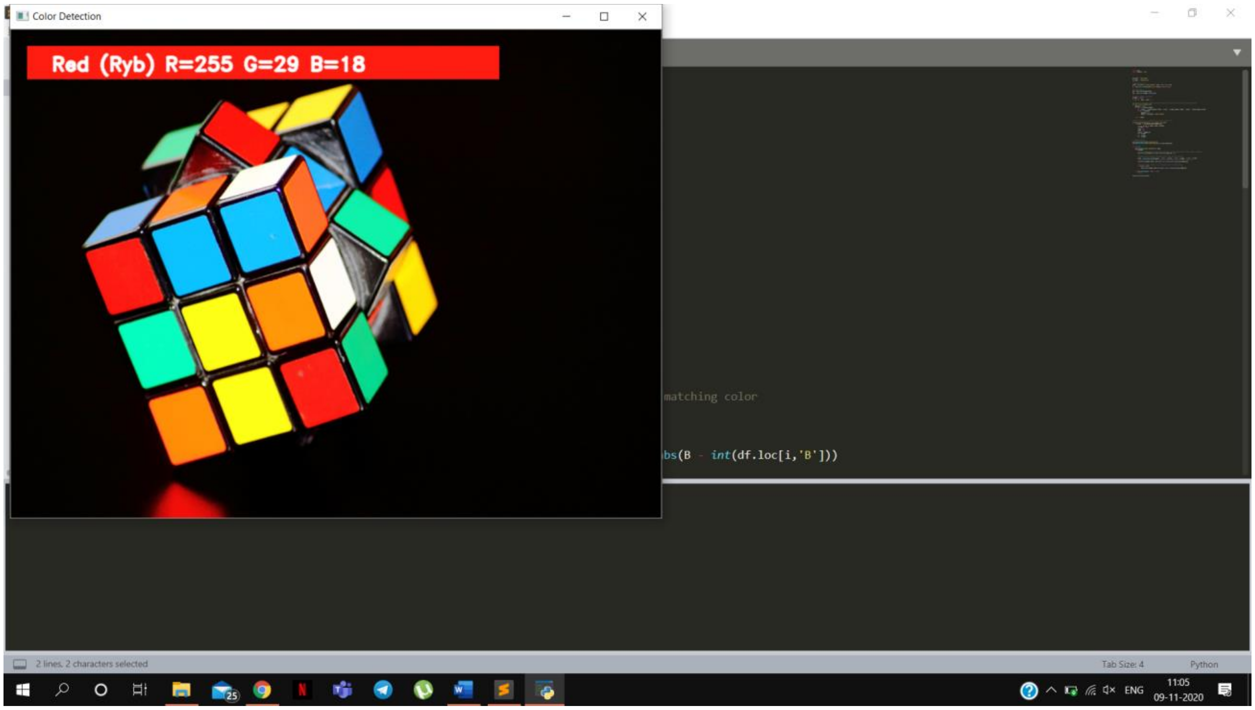


Orange (Ryb) R=255 G=150 B=7

```
matching color

bs(B - int(df.loc[i,'B']))
```

2 lines, 2 characters selected



```
Red (Ryb) R=255 G=29 B=18
```

```
matching color

bs(B - int(df.loc[i,'B']))
```

Feasibility Analysis

Technical Feasibility:

The software is offline and totally based on python so it doesn’t requires any kind of internet and there can’t be any issue regarding component not working correctly.

It is compatible for Windows, Linux and Mac.

Resource and Time feasibility:

Resources that are required for the project,

- Programming Device
- Programming Tool (freely available)
- Programming individuals

So this project has the following required resource.

References

- Color Blindness

  https://www.sciencedirect.com/science/article/abs/pii/S0033350698005903

- Pandas documentation

  https://pandas.pydata.org/docs/

- OpenCv Documentation

  https://opencv.org/