Published in Towards Data Science

Behic Guven Follow

Sep 12, 2020 · 6 min read · Member-only · Listen

Building a Color Recognizer in Python

Simple and hands-on machine learning app using OpenCV

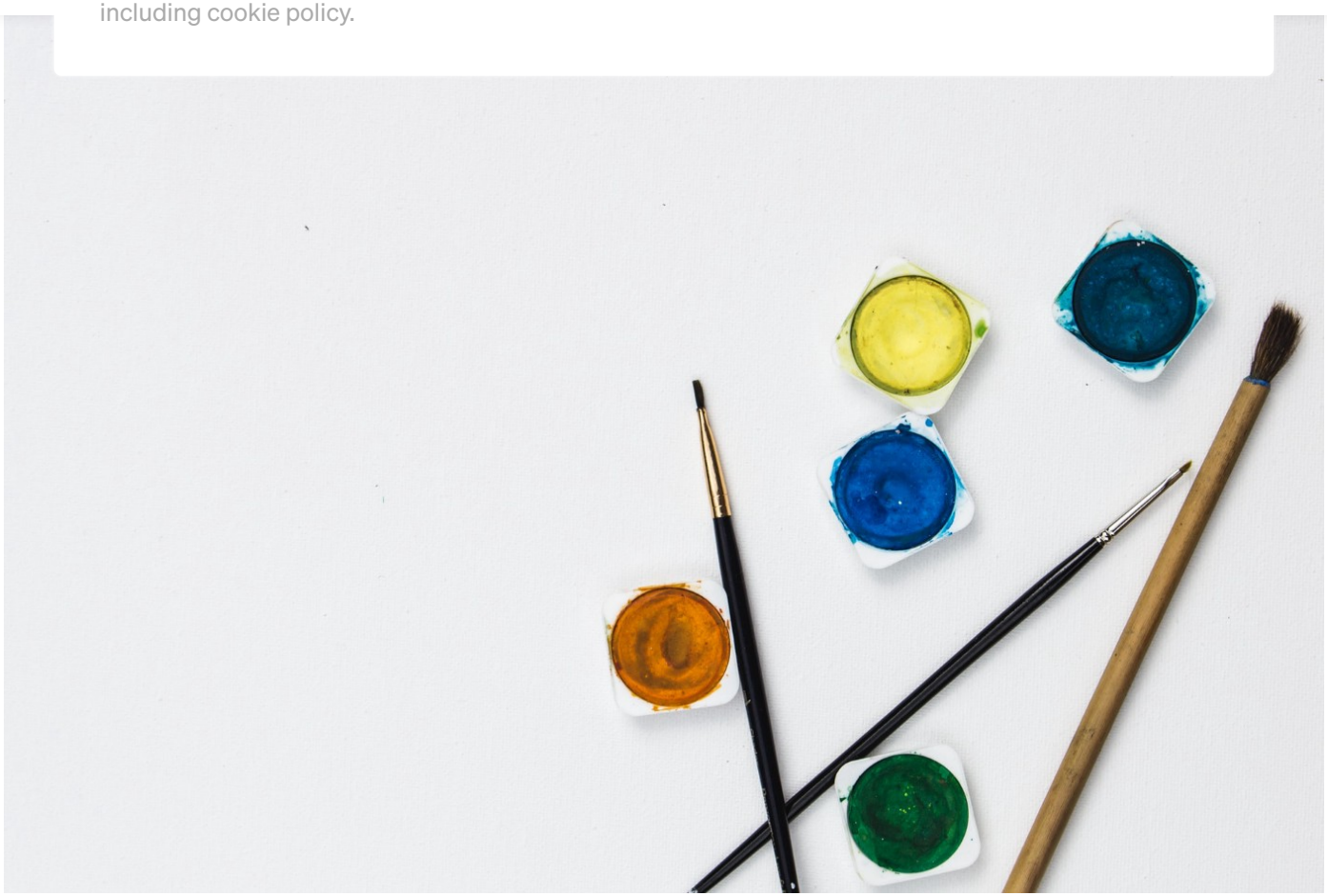


Photo by Kelli Tungay on Unsplash

In this post, I will show you how to build your own color recognizer using Python. This process is also known as “Color Detection”. We will create a

hel

Computer Vision.

If you haven't heard of Computer Vision before, this is the best time to learn about it. Most of the machine learning and artificial intelligence fields are strongly connected to Computer Vision. As we are growing and exploring, seeing the outside world has a big impact on our development. This goes the same for the machines, they see the outside world using images, and those images are turned into data values that computers can understand.

In previous posts, I showed how to detect faces and also how to recognize faces in an image, these are great projects to practice python in artificial intelligence and computer vision. Let's do some work!

Table of Contents

Getting Started

We will use three main modules for this project. They are NumPy, Pandas and OpenCv. OpenCv is a highly optimized library with a focus on real-time applications.

> *OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.*
>
> *Source: https://opencv.org*

Libraries

As mention earlier, there are three modules we will use for this project. To use these modules we have to install the necessary libraries. Library installation is a very easy step using pip. Pip is a package management tool. We will do the installation using the command-line interface. Here is the line to install all 3 libraries at once:

```
pip install numpy pandas opencv-python
```

After the installation is completed, we have to import them to our program. Open a new file in your favorite code editor. Here is the code on how to import the installed libraries:

```
j
```

OpenCv is imported as cv2. And for other libraries, we imported them “as” so that it is easier to call them in the program.

Perfect! Now, we can move to our next step, where we will define the image we want to use to test our color recognizer application.

Define Image

You can choose any image you want. I will save my image in the same folder as my program, which makes it easier to find and import.

```
img = cv2.imread("color_image.jpg")
```

To give you some idea, here is the image I will use for this project:

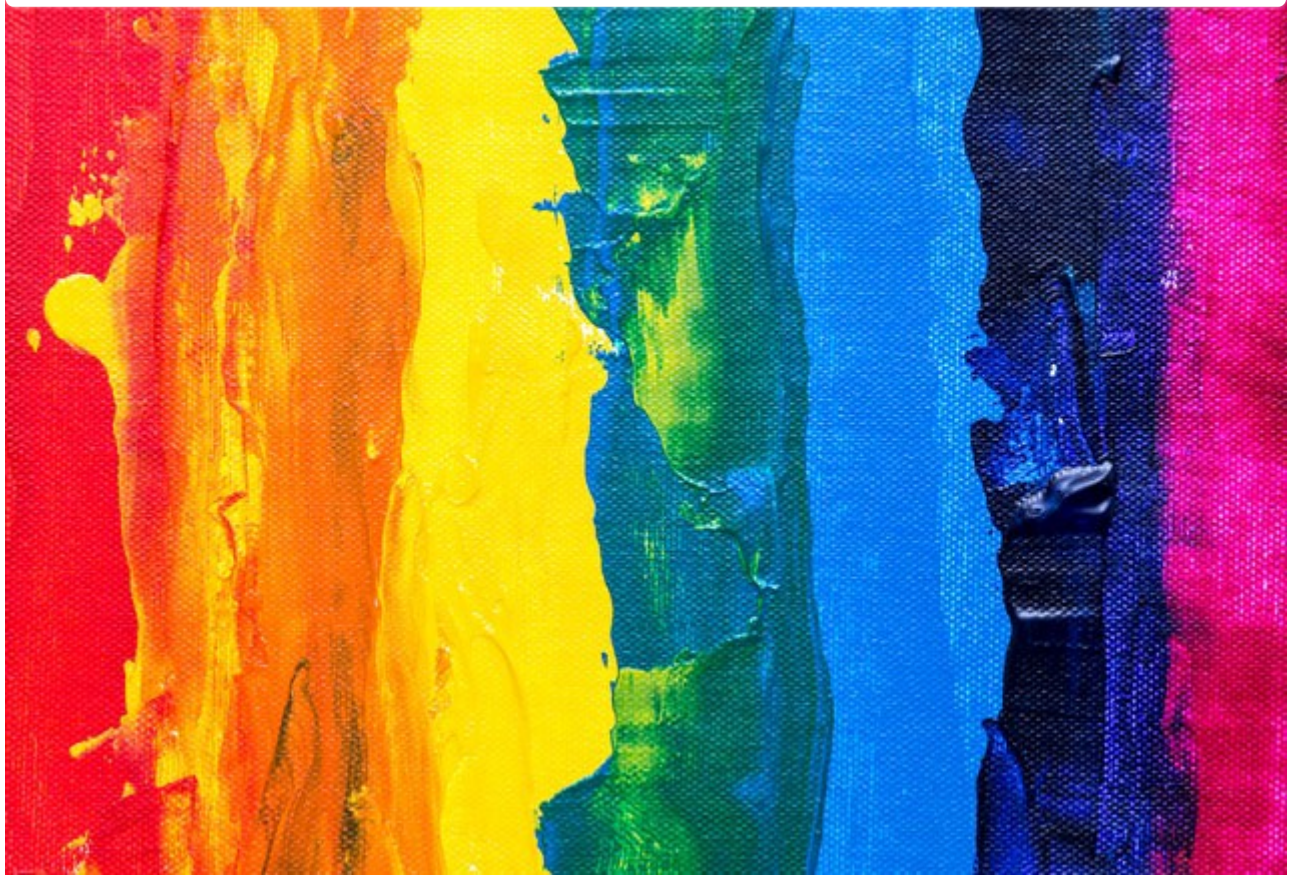



Photo by Steve Johnson from Pexels

Great! Are you ready for some programming? Without losing any time let's move to the next step.

Co

Let me ask you a nice question. *Did you know that machines are so pure?* Well, I think they are because they learn whatever you teach them. They are like a big white canvas. And your program is your brush :)

Teaching the Colors

First, we have to teach them the colors. To do that we need data that includes color names and some values to match with those colors. Since most of the colors can be defined using Red, Green, and Blue. That's why we will use the RGB format as our data points. I found a ready csv file with around 1000 color names and the RGB values. Here is the GitHub link. We will use this csv file in our program. The screenshot of the file to give you some idea:

alice_blue	Alice Blue	#f0f8ff	240	248	255
alizarin_crimson	Alizarin Crimson	#e32636	227	38	54
alloy_orange	Alloy Orange	#c46210	196	98	16
almond	Almond	#efdecd	239	222	205
amaranth	Amaranth	#e52b50	229	43	80
amber	Amber	#ffbf00	255	191	0

Let

file we downloaded doesn't have column names, I will be defining them in the program. This process is known as data manipulation.

```
index=["color", "color_name", "hex", "R", "G", "B"]

csv = pd.read_csv('colors.csv', names=index, header=None)
```

Global Variables

In the following steps, we will define two functions. To make the application work smoothly, we need some global variables. You will know how global variables can be helpful when working with functions.

```
clicked = False
r = g = b = xpos = ypos = 0
```

Color Recognition Function


hap

```
def recognize_color(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
```

Mouse Click Function

This function is used to define our double click process. We will need it when creating our application part.

```
def mouse_click(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
```


I h[illegible] t

writing them in your editor, the big picture will come to life. I do my best to keep things simple and easy to understand. I will add my contact info at the end of this article, reach me if you need any help.

Application

I am glad you made it to this step. In this step, we will open the image as a new window using OpenCV methods. And in that window, we will use the functions we defined earlier. The application is so simple, it returns the color name and color values when you double click on a certain area on the image.

Application Window

First things first, let me show you how to open the image file as a new window using OpenCV.

```
cv2.namedWindow('Color Recognition App')
```

Sec

functionality to our application.

```
cv2.setMouseCallback('Color Recognition App', mouse_click)
```

The Application

Here is the while loop to start our application window working.

```
while(1):

cv2.imshow("Color Recognition App",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 = recognize_color(r,g,b) + ' R='+ str(r) +  ' G='+ str(g)
+  ' B='+ str(b)
```


```
#
        cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False
```

Close the Application

If you worked with OpenCV projects, you may be familiar with this step. We have to define how to end and close the application window. Otherwise, it will run forever since we used *while(1)* to start the application. Adding the following lines is a good practice for your future projects.

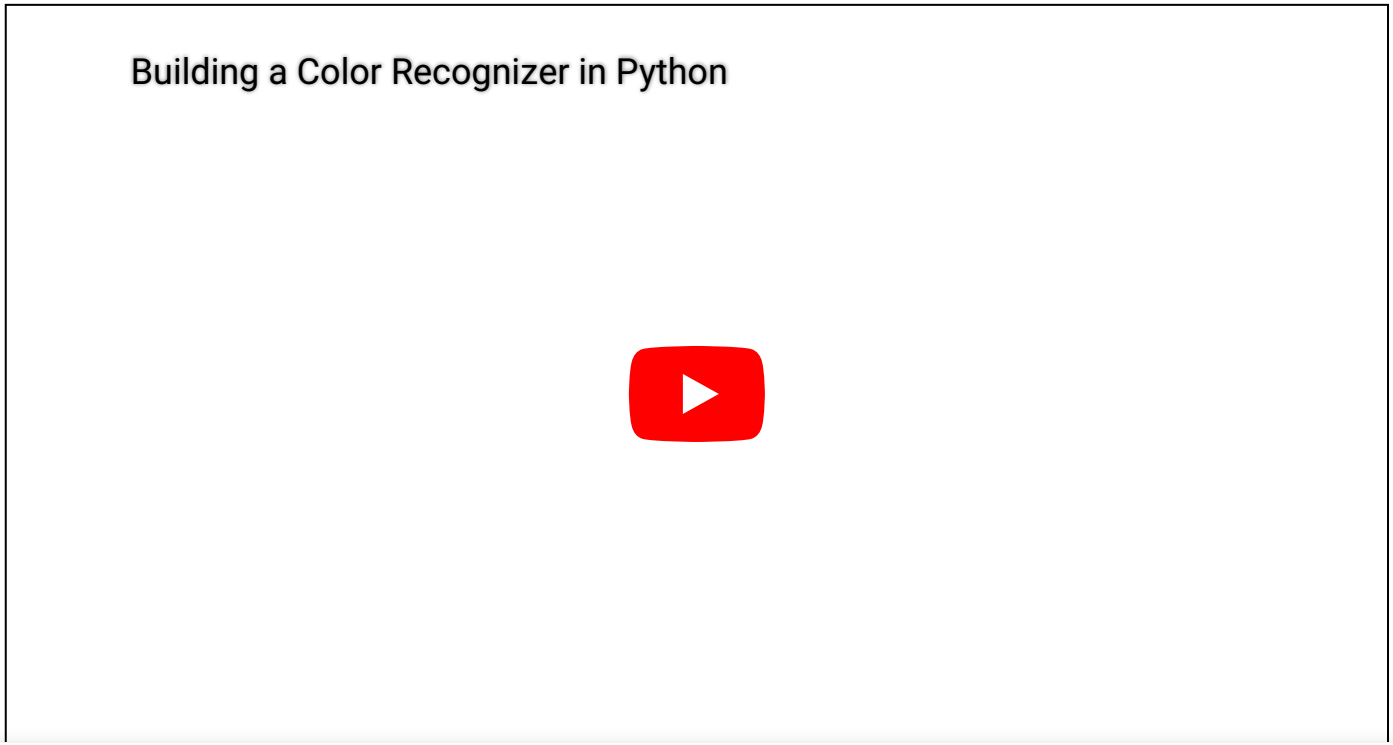
```
#Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```


Cu

many positive feedbacks about the demonstration videos, it gives a better understanding of the program and application process. The video will be published on my **youtube channel**.

(Update: Video is ready and available below. Thanks 😊)



Congrats!! You have created a cool computer vision application that recognizes colors in an image. Now, you have some idea of how to use computer vision in a real project. Hoping that you enjoyed reading my article. I will be so happy if you learned something new today. Working on hands-on programming projects like this one is the best way to sharpen your coding skills.

Feel free to contact me if you have any questions while implementing the code.

> *Follow my blog and Towards Data Science to stay inspired. Thank you,*

More Computer Vision Projects:

Rendering Text on Video using Python

Hands-on machine learning project using moviePy library

towardsdatascience.com

-

Step-by-step guide to face recognition in real-time using OpenCv library

towardsdatascience.com