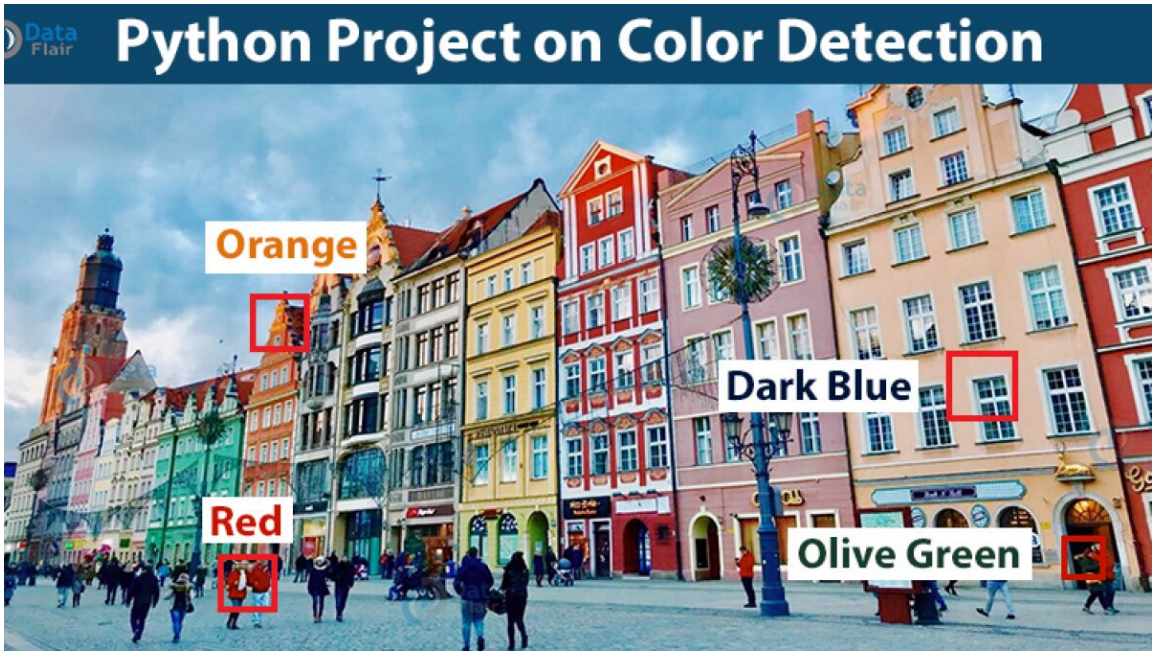
Python Project in Image Detection, Recognition, and Classification

Reading the colors of an image and turning selfies to drawings

By Cai Lin, Project for CISC4900



Abstract

My project is the development of applications related to Computer Vision, which is really important in our development of Artificial Intelligence. The specific project I am doing is using Python to develop an image detection/recognition and classification application that can aid in converting a selfie you upload into a pencil like drawing. Another function of the application is to automatically detect a color that your mouse clicks on in the image.

Project Details

My Motivation for the project

Last Semester I was part of Cuny Tech Prep's Data Science and Web development Fellowship. For the final project we had to complete a data science project using everything we learned from our time with our mentors. For my project I was able to complete a Machine learning project that detected patterns in trending video statistics. However, during the final week of presentation I saw my classmates doing a project in image recognition and it suddenly inspired me to do the same. So, I started doing research and decided on doing my project on Image recognition and classification as well.

While I was doing the research, I actually happened to encounter a problem that is specifically related to my project. So as a side hustle, I run an ecommerce store Mysheeb.com with my family, which sells small craft stuff. Earlier this year, there was this order that asked to make a selfie of the customer's cat into a sticker, and it just clicked with me that I should complete an application for my website that will automatically turn selfies into drawings. Halfway through finishing my project, I discovered that people have a hard time telling the colors of an image. Which is quite important since we don't want to make an order but turns out the customer had another color in mind. So I decided to add a color classification application as well!

Detailed Description of Project

In this project I will be creating an application that will aid in turning a selfie into a drawing as well as detecting the specific colors of an image. I will be coding in python using an extremely useful library called OpenCv that helps with implementing Image recognition applications. In the selfie-to-drawing application, the user can upload a selfie of themselves and the application will then turn the selfie image into a drawing image. This will help my customers from my ecommerce store in getting their selfies turned into handicraft items such as stickers that they can purchase and customize!

In the application for color recognition, it is actually even more useful! In this application, the user will select an image and click on any position in the image. The application will then be able to tell the specific color of position that the mouse clicked. This is so useful because as someone like me who is basically color blind(not actually, but I just can't tell the difference in the name for colors.) There is just way too many different types of Pink. With this application, customers will then be able to tell us the actual color they want, instead of us trying to figure it out. Another important fact is that some monitors show color differently, so sometimes it is confusing for the naked eye to actually detect the specific color.

Our ecommerce store does a lot of big orders from manufacturers in China via Aliexpress. In the orders we will usually send them a drawing of the product we want. However, there is a major issue in that sometimes the color we want is miscommunicated. Usually there is this big color book that has all these different colors and their name, and that book is then used to communicate with manufacturers on the actual color we want. It seems really outdated, so my application will hopefully help resolve this issue for manufacturers and other small hand craft ecommerce store owners such as us. It can also be very helpful for other graphic designers and web designers to have an application for color recognizer.

Technical overview of the project

Selfie-Image to Drawing

There are two main steps that I had to do to convert a selfie-image into a drawing, which is detecting the edges and smoothing region. Since a drawing usually have well-defined edges, I can then use code to capture the edges of the image to emphasize and define them. Then region smoothing will help remove insignificant color boundaries as well as reduce the noise or grain of the image(making the pixel more clear). The above are done via the library from OpenCv, which is amazing on what it can accomplish.

First I created a function that will handle all the work to convert an image into a drawing. Then with this function cvtColor() from OpenCv I then converted the image into a grayscale image.

```
#conver the image into a grayscale image via cvtColor from opencv
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Now to convert the image into a Drawing, we have to apply GaussianBlur() from OpenCv and set a scale kernel for the user to adjust the blurriness of the grayscale image. The greater kernel value the user selected, the larger the standard deviation and would in turn increase the blurring effect. By blurring the image, we are actually able to reduce the noise of the image and allow us to detect edges of the image. Below is an example of a blurring effect based on different settings.

Kernel size = 10

Kernel Size = 60

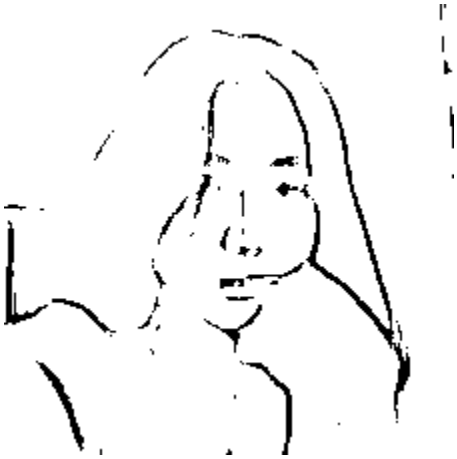
Kernel Value = 90

Next we will have to divide the original grayscale image with the blurred grayscale image. This gives us a ratio of change between each pixel of two images. The stronger the blurred effects, the more value of each pixel changes with respect to its origin and gives us a sharper sketch.

Detecting edges

```
gray = cv2.medianBlur(gray, kernel)
edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                              cv2.THRESH_BINARY, 9, 9)
```

The next step is to detect the edges of the image, which is done by using the cv2.adaptiveThreshold function from OpenCV. Here we have to reduce the noise from the blurred grayscale image again using cv2.medianBlur. The larger the blur value means fewer black noises in the image. The adaptiveThreshold function will then define the edges. The larger value defined meant thicker edges that will be emphasized in the image. Below is an example I created.



Next we have to reduce the number of colors in the drawing because a photo usually has a lot more colors. We can use a K-Means clustering algorithm provided by OpenCv to apply color quantization to the image

```
# Transform the image
data = np.float32(img).reshape((-1, 3))

# Determine criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 0.001)

# Implementing K-Means
ret, label, center = cv2.kmeans(data, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
center = np.uint8(center)
result = center[label.flatten()]
result = result.reshape(img.shape)
```

The User can then adjust the k value to determine the number of colors they want to apply to the image. Below is an example using a 10 as K value for the image.

Bilateral Filter

Then we use a Bilateral Filter to reduce the noise in the image as well as give the image a bit blurred and sharpness-reducing effect to the image.
K means algorithm in this project to extract RGB values

```
gray = cv2.medianBlur(gray, kernel)
edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                              cv2.THRESH_BINARY, 9, 9)

color = cv2.bilateralFilter(img, smooth, edge_preserve, smooth)
Drawing = cv2.bitwise_and(color, color, mask=edges)
```

In the example, you can change the smooth, edge_preserve value using the Kernel. Below is an example of the final image.

Smooth - It is the diameter of each pixel neighborhood, the higher the value the smoother the image

Edge_preserve - It is the value the tunes the color averaging effect

Color detection and categorization

The second part of the application is to detect colors in an image. The program will return the RGB values of the color as well as the name of the color associated with them. The main Python libraries used for this application are NumPy, Pandas, and OpenCV.

Importing the necessary files

The first part of the application is to use the cv2.imread function of OpenCv and have an image uploaded into the application. Below is the sample image I used for the project.

Next we will have to teach or feed our application with data in color names and their matching values. In this application I will be using the RGB format. The data I will be using from the Github link below, with an example of the data.
https://github.com/codebrainz/color-names/blob/master/output/colors.csv

air_force_blue_raf	Air Force Blue (Raf)	#5d8aa8	93	138	168
air_force_blue_usaf	Air Force Blue (Usaf)	#00308f	0	48	143
air_superiority_blue	Air Superiority Blue	#72a0c1	114	160	193
alabama_crimson	Alabama Crimson	#a32638	163	38	56
alice_blue	Alice Blue	#f0f8ff	240	248	255
alizarin_crimson	Alizarin Crimson	#e32636	227	38	54
alloy_orange	Alloy Orange	#c46210	196	98	16

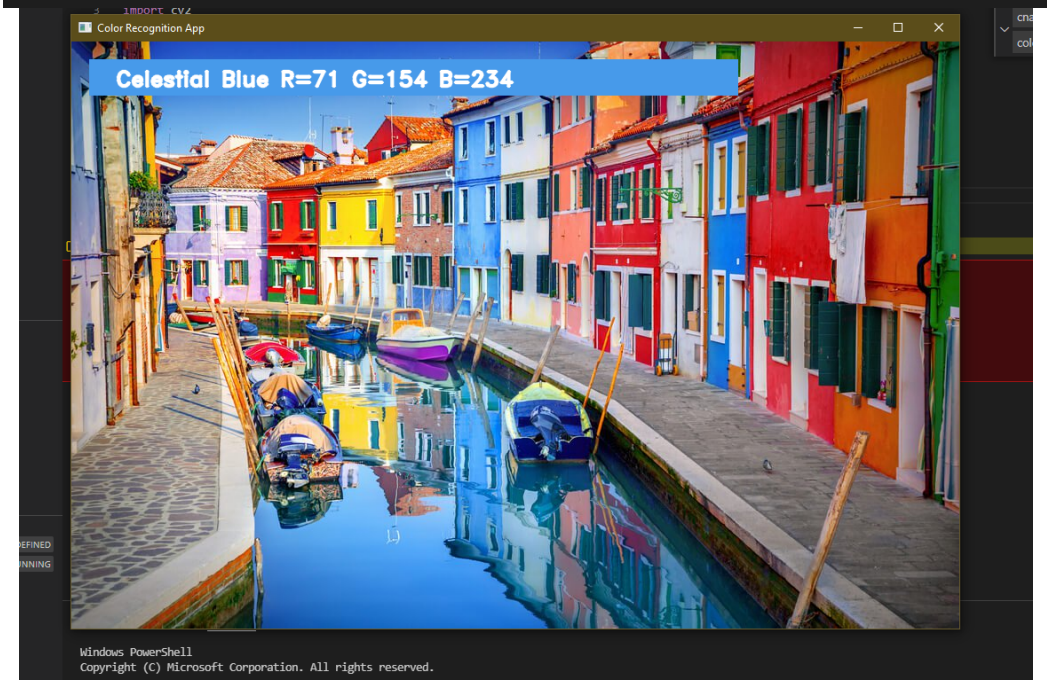
Then we can import the colors.csv file using the read_csv function from Pandas.

```
index=["color", "color_name", "hex", "R", "G", "B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

Color Recognition Function

The next step is to implement a color recognition function using the K-mean algorithm. The function will be called whenever the user double clicks on anywhere on the image, and it will then return the color name as well as its RGB values on the location of the mouse click.

```
def color_recognition(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        x = 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 = x
            colorName = csv.loc[i,"color_name"]
    return colorName
```



Mouse Click Function

Then we just have to add a double click function that will allow users to select the area of image they want the color from.

```
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
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)
cv2.namedWindow('Color Recognition App')
cv2.setMouseCallback('Color Recognition App', mouse_click)
```

Converting the code into an application

Now we just have to convert the codes into an application. For converting selfies into drawings, I had to make it into a web application as part of my website. So I used the Python library Streamlit to help make the webapp. I first created a widget that allows users to upload their own image using streamlit.file_uploader() function.

```
file = st.sidebar.file_uploader("Please upload an image file", type=["jpg", "png"])

if file is None:
    st.text("You haven't uploaded an image file")
else:
    image = Image.open(file)
    img = np.array(image)
```

Then we just have to add a slider for the users to adjust each parameter value by using the function streamlit.slider() from streamlit.

```
value = st.sidebar.slider('Tune the brightness of your sketch (the higher the value, the brighter your sketch)', 0.0, 300.0, 250.0)
kernel = st.sidebar.slider('Tune the boldness of the edges of your sketch (the higher the value, the bolder the edges)', 1, 99, 25, step=2)
```

Background

I took Machine learning last semester as an elective as well as participated in Cuny Tech Prep fellowship, where I learned a lot about machine learning. It is the first time I touched upon using Python and machine learning, and it just seems so interesting. In my machine learning class, I learned about quantizing data using algorithms such as K-means, and I actually had to use it for my project! There is just so much to learn in machine learning and Artificial intelligence. I consider myself to be a novice and there is just so much more to learn on the topic.

In this project, I had partially used knowledge from the final project I competed as a fellow for CTP(Cuny Tech Prep) as well as my work on my family's ecommerce store. For my CTP final project, I had to use machine learning to train and predict the pattern in trending videos. So it had familiarized myself with a lot of the coding and research I did in this project. For example, I had already known how to use all the software applications and Python Libraries I used in this project. My Mentor and other fellow members also taught me many powerful websites where I was able to learn and use technology from. For example one of my fellow members did his project on image recognition on a webcam, which I learned a great deal from him. He used resources from this github repos and I think it was really useful. https://github.com/sudonitin/live-color-picker. I also used a lot of resources from this github repo to learn the basics of implementing image to drawing applications. (https://github.com/SerialLain3170/AwesomeAnimeResearch)

Literature Review

https://github.com/sudonitin/live-color-picker.
https://github.com/SerialLain3170/AwesomeAnimeResearch
https://github.com/SystemErrorWang/White-box-Cartoonization
(Initial research)

As I mentioned above, I got most of my idea on creating my project from Github research and resources provided by CTP. After Completing the project, I want to expand upon my work and create more features of my application. Right now it is just the basic sketch and cartoonish art style. I want to expand that and generate anime or comic-like features as well in the future.

Project Objectives and Expected Outcomes

Objectives

1. Make an application for converting selfies into drawings
2. Make an application that allows user to detect colors of an image
3. Make the selfie-to-drawing application into an webapp

Expected Outcomes

1. User will be able to upload an selfie and have it converted into a sketch drawing
2. User will be able to change the sketch drawing into a more cartoonish drawing
3. User will be able to change the parameters on how their image looks using a slider
4. Color Detection: User will be able to choose an image of their choice and detect colors by left clicking twice on an area of the image
5. The application will then return the color name and the RGB value

Obstacles & Risks

I had to learn how to use new Python libraries such as OpenCV and streamlit. When I was initially doing the project, I had used Anaconda, Jupitor notebook, and google colabs to code my work initially. However, I had an issue of Anaconda not seeing OpenCv's library. So I had to uninstall and reinstall my Python folder and even install VS Code to complete my project. It was interesting, but I later figured out that I just had to change my anaconda Python environment to resolve this issue.

Another few challenges I faced are that I did not have enough time to add more features into my selfie-to-drawing application. I wanted to add an anime-like feature onto the application, but unfortunately there is a lot of data training to do and it will take me some time to figure it out. Hopefully I will be able to add more features to my application in the summer.

Software and Hardware Technology Requirements

Software Requirements

Describe any software that is required to complete your project objective(s).

1. VS code - https://code.visualstudio.com/
2. Anaconda/Jupyter Notebook - https://www.anaconda.com/
3. Google Colabs - https://colab.research.google.com/drive/1lV5oJ_hI8PsSV1WDVWWfL18-tMm4vnxe?usp=sharing#scrollTo=cyBGM-vuFNrE
4. Python - https://www.python.org/
5. Python Library
    a. Bottleneck
    b. nltk
    c. nose
    d. notebook
    e. numba
    f. numexpr
    g. numpy
    h. numpydoc
    i. opencv-python
    j. openpyxl
    k. opt-einsum
    l. packaging
    m. pep8
    n. pexpect
    o. pickleshare
    p. Pillow
    q. pkginfo
    r. pluggy
    s. ply
    t. PyYAML
    u. pyzmq
    v. SQLAlchemy

w. statsmodels
x. streamlit
y. sympy
z. Tqdm

6. Github
    a. Colors.csv - https://github.com/codebrainz/color-names/blob/master/output/colors.csv
    b. https://github.com/sudonitin/live-color-picker.
    c. https://github.com/SerialLain3170/AwesomeAnimeResearch
    d. https://github.com/SystemErrorWang/White-box-Cartoonization

Architecture

Selfie-into-drawing - user will upload an image that will then be trained and converted into an grayscale image, resulting in a sketch like drawing. The grayscale image will then have its edges emphasized via gray_blur. Then apply a color quantization and will result in a cartoonish drawing.

Color recognition - users will upload an image into the application that will then be trained and analyzed. The user can then left click twice with their mouse anywhere on the image to have its color and RGB value returned.

Project Log

1. Weekly Log

Weeks	Duration	Description of completed work	Challenges or Next steps
1	15 hours	Reading the syllabus Attending the orientation Doing some research via student portal on list of potential projects	Finding a topic or a project I will be interested in
2	20 hours	Continuation of research on list of potential projects	
3	15 hours	Contacted some of the supervisors suggested by the portal	
4	18 Hours	Contacted my project supervisor professor Katherine chuange and had a discussion with her on my project proposal	Decided I will continue my work done for CTP
5	25 Hours	Had an overview of my final	Deciding on how to improve my

		project done for CTP Rewatch the final presentations,	project
6	15 Hours	Decided to do an image recognition project, conducting initial research	
7	21 hours	Narrow down my project to a sign language reading project, color recognition project, and image to anime project After talking with my mentor we decided to do both color recognition project as well as image to drawing application	Next I Conduct research on what resources are needed to finish the project
8	15 Hours	Conducting research on initial resources needed for project	
9	15 Hours	Learned how to use new libraries	Next to start coding for the

		that will be used for the project	project
10	17 Hours	Encounter an issue on Jupiter Notebook Attempted to resolve issue Conduct research on why there was an issue	Reinstalled my software to attempt to fix the issue
11	15 Hours	Finish my selfie to image project features and training on Google Colabs	Next to convert the project into an webapp
12	22 Hours	Finish selfie to image project on VS code and added streamlit library to make it into a web app	
13	30 Hours	Finish the image recognition project on Jupyter Notebook	
14	20 Hours	Completed Final report	
15	15 Hours	Complete presentation	

Bibliography

1. https://github.com/sudonitin/live-color-picker.
2. https://github.com/SerialLain3170/AwesomeAnimeResearch
3. https://github.com/SystemErrorWang/White-box-Cartoonization
4. VS code - https://code.visualstudio.com/
5. Anaconda/Jupyter Notebook - https://www.anaconda.com/
6. Google Colabs - https://colab.research.google.com/drive/1lV5oJ_hI8PsSV1WDVWWfL18-tMm4vnxe?usp=sharing#scrollTo=cyBGM-vuFNrE
7. Python - https://www.python.org/
8. Python Library
    a. Bottleneck
    b. nltk
    c. nose
    d. notebook
    e. numba
    f. numexpr
    g. numpy
    h. numpydoc
    i. opencv-python
    j. openpyxl
    k. opt-einsum
    l. packaging
    m. pep8
    n. pexpect
    o. pickleshare
    p. Pillow
    q. pkginfo

r. pluggy
s. ply
t. PyYAML
u. pyzmq
v. SQLAlchemy
w. statsmodels
x. streamlit
y. sympy
z. Tqdm

9. Github
    a. Colors.csv - https://github.com/codebrainz/color-names/blob/master/output/colors.csv
    b. https://github.com/sudonitin/live-color-picker.
    c. https://github.com/SerialLain3170/AwesomeAnimeResearch
    d. https://github.com/SystemErrorWang/White-box-Cartoonization
10. Other research
    a. https://towardsdatascience.com/turn-photos-into-cartoons-using-python-bb1a9f578a7e
    b. https://towardsai.net/p/deep-learning/an-insiders-guide-to-cartoonization-using-machine-learning
    c. https://towardsdatascience.com/building-an-image-cartoonization-web-app-with-python-382c7c143b0d
    d. https://data-flair.training/blogs/python-deep-learning-project-handwritten-digit-recognition/