Assignment 1 Q1

Mounting

```
from google.colab import drive
drive.mount('/content/gdrive')
drive_path = '/content/gdrive/MyDrive/Multimedia_Compression/HW1/Color squares/'
```

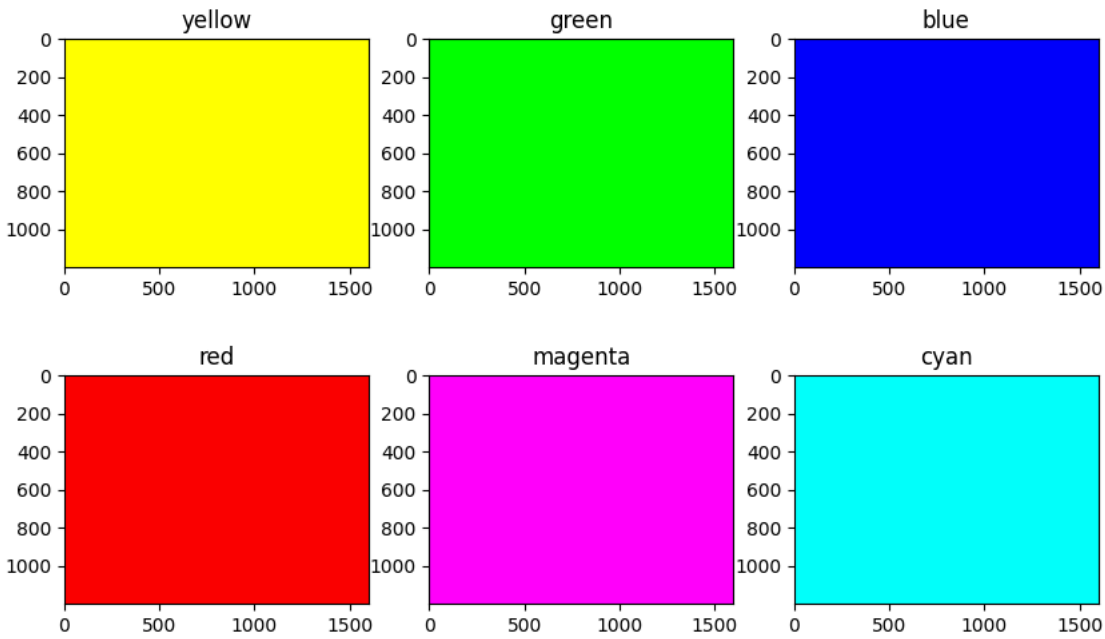
```
Mounted at /content/gdrive
```

Libraries

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

1. We will draw rectangles in the colors pure red (R = 0xff, G = 0x00, B = 0x00), pure green (0x00ff00), pure blue (0x0000ff), cyan (0x00ffff), magenta (0xff00ff), and yellow (0xffff00) and represent the computer.

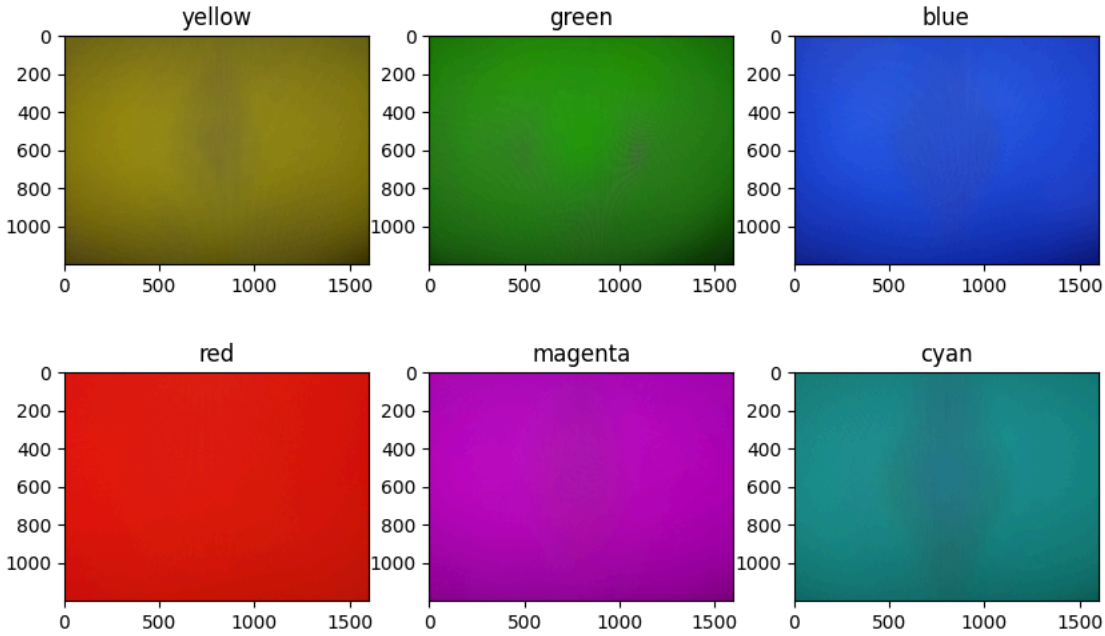
```
yellow = cv2.cvtColor(cv2.imread(drive_path + 'pure yellow.jpeg'), cv2.COLOR_BGR2RGB)
green = cv2.cvtColor(cv2.imread(drive_path + 'pure green.jpeg'), cv2.COLOR_BGR2RGB)
blue = cv2.cvtColor(cv2.imread(drive_path + 'pure blue.jpeg'), cv2.COLOR_BGR2RGB)
red = cv2.cvtColor(cv2.imread(drive_path + 'pure red.jpeg'), cv2.COLOR_BGR2RGB)
magenta = cv2.cvtColor(cv2.imread(drive_path + 'pure magenta.jpeg'), cv2.COLOR_BGR2RGB)
cyan = cv2.cvtColor(cv2.imread(drive_path + 'pure cyan.jpeg'), cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10, 6))
plt.subplot(2, 3, 1)
plt.title('yellow')
plt.imshow(yellow)
plt.subplot(2, 3, 2)
plt.title('green')
plt.imshow(green)
plt.subplot(2, 3, 3)
plt.title('blue')
plt.imshow(blue)
plt.subplot(2, 3, 4)
plt.title('red')
plt.imshow(red)
plt.subplot(2, 3, 5)
plt.title('magenta')
plt.imshow(magenta)
plt.subplot(2, 3, 6)
plt.title('cyan')
plt.imshow(cyan)
plt.show()
```



2. and 3. We took Images of the paintings using the cell phone camera and will present them using a code. We covered at least 80% of the area in order to create a large, uniform area in color.

We convert the Images to in RBG.

```
yellow = cv2.cvtColor(cv2.imread(drive_path + 'yellow.jpeg'), cv2.COLOR_BGR2RGB)
green = cv2.cvtColor(cv2.imread(drive_path + 'green.jpeg'), cv2.COLOR_BGR2RGB)
blue = cv2.cvtColor(cv2.imread(drive_path + 'blue.jpeg'), cv2.COLOR_BGR2RGB)
red = cv2.cvtColor(cv2.imread(drive_path + 'red.jpeg'), cv2.COLOR_BGR2RGB)
magenta = cv2.cvtColor(cv2.imread(drive_path + 'magenta.jpeg'), cv2.COLOR_BGR2RGB)
cyan = cv2.cvtColor(cv2.imread(drive_path + 'cyan.jpeg'), cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10, 6))
plt.subplot(2, 3, 1)
plt.title('yellow')
plt.imshow(yellow)
plt.subplot(2, 3, 2)
plt.title('green')
plt.imshow(green)
plt.subplot(2, 3, 3)
plt.title('blue')
plt.imshow(blue)
plt.subplot(2, 3, 4)
plt.title('red')
plt.imshow(red)
plt.subplot(2, 3, 5)
plt.title('magenta')
plt.imshow(magenta)
plt.subplot(2, 3, 6)
plt.title('cyan')
plt.imshow(cyan)
plt.show()
```



4. We found and calculated the average RGB values of each rectangle, including the standard deviations of the RGB values. We will present the results in a table.

Mean for every image

```
yellow_mean = np.mean(yellow, axis=(0, 1))
green_mean = np.mean(green, axis=(0, 1))
blue_mean = np.mean(blue, axis=(0, 1))
red_mean = np.mean(red, axis=(0, 1))
magenta_mean = np.mean(magenta, axis=(0, 1))
cyan_mean = np.mean(cyan, axis=(0, 1))

print('Average RGB values channle:')
print(f'For yellow:  {yellow_mean}')
print(f'For green: {green_mean}')
print(f'For blue: {blue_mean}')
print(f'For red: {red_mean}')
```

```
print(f'For magenta: {magenta_mean}')
print(f'For cyan: {cyan_mean}')
```

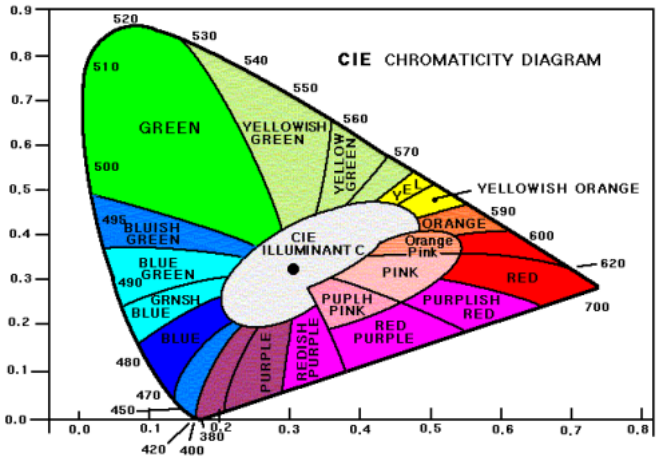
```
Average RGB values channle:
For yellow:  [127.11357552 116.85259844  22.5635224 ]
For green: [ 38.16961771 124.90157083  21.35739271]
For blue: [ 33.45304583  71.15809687 202.66508802]
For red: [216.16490521  25.11568021  13.0965375 ]
For magenta: [171.2661349    9.8273474  177.29959687]
For cyan: [ 27.93967604 128.6245125  128.35990729]
```

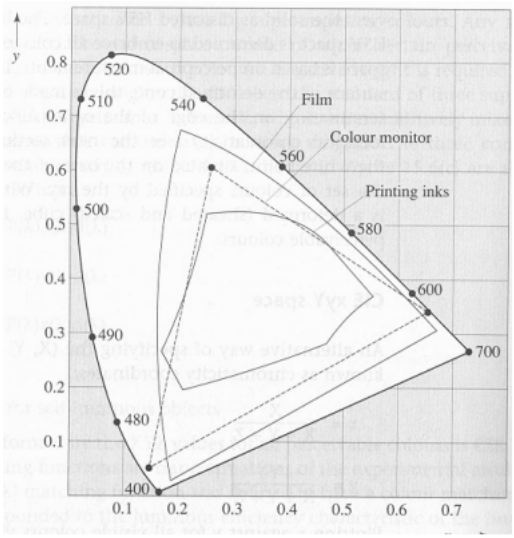
Calculating the standard deviation

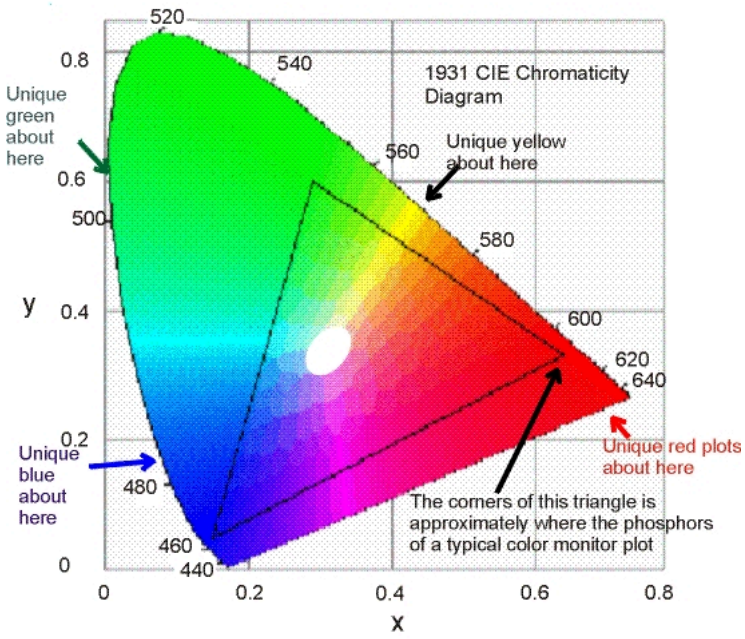
```
yellow_std =np.sqrt((255-yellow_mean[0]**2) + (255-yellow_mean[1])**2 + (0-yellow_mean[2])**2)
green_std =np.sqrt((0-green_mean[0])**2 + (255-green_mean[1])**2 + (0-green_mean[2])**2)
blue_std =np.sqrt((0-blue_mean[0])**2 + (0-blue_mean[1])**2 + (255-blue_mean[2])**2)
red_std =np.sqrt((255-red_mean[0])**2 + (0-red_mean[1])**2 + (0-red_mean[2])**2)
magenta_std =np.sqrt((255-magenta_mean[0])**2 + (0-magenta_mean[1])**2 + (255-magenta_mean[2])**2)
cyan_std =np.sqrt((0-cyan_mean[0])**2 + (255-cyan_mean[1])**2 + (255-cyan_mean[2])**2)

print('Standard deviation RGB values channle:')
print(f'For yellow:  {yellow_std}')
print(f'For green: {green_std}')
print(f'For blue: {blue_std}')
print(f'For red: {red_std}')
print(f'For magenta: {magenta_std}')
print(f'For cyan: {cyan_std}')
```

```
Standard deviation RGB values channle:
For yellow:  60.75323875882912
For green: 137.2539952457941
For blue: 94.45381960654692
For red: 48.067465860953284
For magenta: 114.65290911227976
For cyan: 181.07761435183312
```

Color	Average R Value	Average G Value	Average B Value	Standard Deviation
Yellow	127.11	116.85	22.56	60.75
Green	38.17	124.90	21.36	137.25
Blue	33.45	71.16	202.67	94.45
Red	216.16	25.12	13.10	48.07
Magenta	171.27	9.83	177.30	114.65
Cyan	27.94	128.62	128.36	181.08







4a. Which color varies the most? Explanation:

Cyan

As we can see in the diagrams from the lecture, the range of colors that can be displayed on electronic devices is limited. In addition, in diagram number 3, the triangle represents the range of colors that the monitor can display. The relative proportion of cyan within the triangle is the lowest out of the six colors we tested. This leads to a large variance.

4b. Which color changes the least? Explanation:

Red

For the same reason mentioned above, red is the color that changes the least. As we can see, the relative proportion of red within the triangle is the highest out of the six colors we tested. Therefore, the variance of the red color is the lowest among these six colors.

5. We calculate the pixel size in inches of an image that is 1024 pixels long and 1024 pixels wide, with a screen width of 16 inches.

$$PPI(\text{Pixels per Inch}) = \frac{1024}{16} = 64\,\text{pixels per inch}$$

$$\text{Pixel Size (in inches)} = \frac{1}{\text{PPI}} = \frac{1}{64} \approx 0.015625\ \text{inches}$$

The viewing distance given the resolution of 0.6arcmin calculated by the formula:

$$\text{Viewing Distance (in inches)} = \frac{\text{Pixel Size (in inches)}}{tan(\theta)}$$

In addition given 0.6arcmin we know that:

$$0.6 \text{ arcmin} = 0.6\frac{1}{60} \text{ degrees} = 0.01 \text{ degrees} = 0.01\frac{\pi}{180} \text{ Radians} \approx 0.0001745329 = \theta$$

Therfore:

$$\text{Viewing Distance (in inches)} = \frac{\frac{1}{64}}{tan(0.0001745329 \text{ radians})} \approx 89.52466 \approx 90 \text{ inches}$$

6. A 10-second video with 1000 frames calculates FPS We compressed the video and now there are only 760 frames. How does this affect the file size of the video and why?

Calculates FPS:

$$\text{FPS} = \frac{\text{Number of Frames}}{\text{Duration}} = \frac{1000}{10} = 100\text{FPS}$$

How does this affect the file size of the video?

Each frame contains a certain amount of information. When we compressed the video and reduced the number of frames from 1000 to 760, this means that the video contains less data that needs to be stored in it, which means the file's size will decrease.

Note that the amount of data in the video decreases due to a $(1 - \frac{760}{1000}) = 24\%$ decrease in the number of frames.

And why?

This means that less space is needed to store all the information in the video, and thus the file's size is reduced