Image Gray scaling

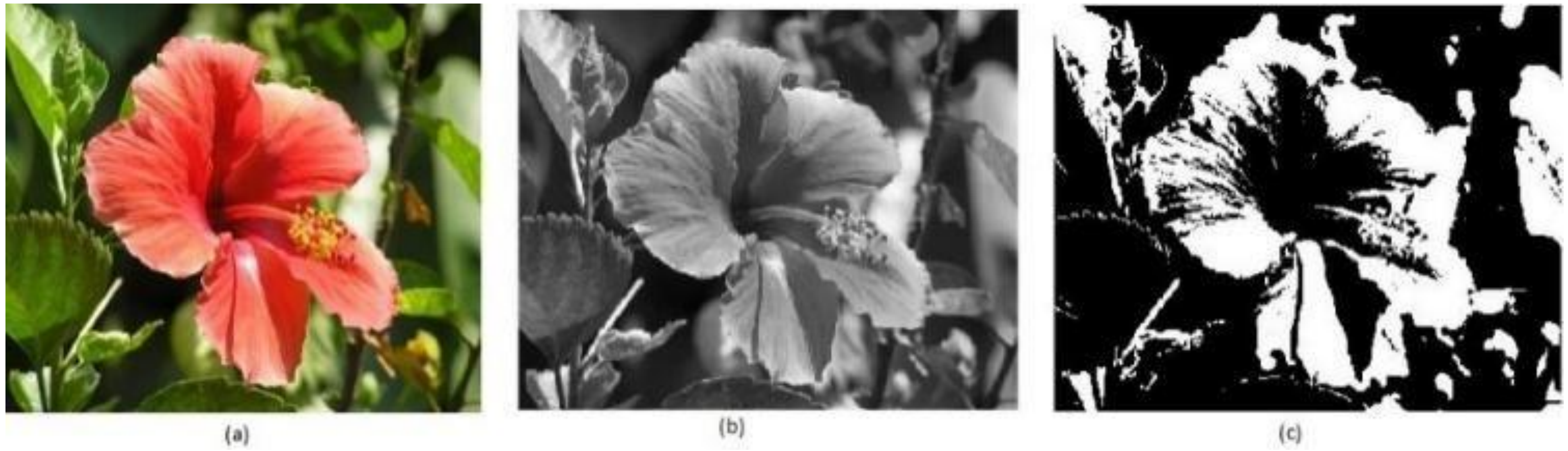
Abhay Magar(23MDS001)
Vaibhav Raval(23MDS010)
Aashita Jain(23MDS015)

Types of Images:

1. Color Image

2. Grayscale Image

3. Binary Image



Gray Scale Image:

1. A gray-scale (or gray level) image is simply one in which the only colors are shades of gray. Unlike color images, which contain multiple color channels (typically red, green, and blue), grayscale images contain only one channel representing the intensity of light at each pixel.
2. In a gray-scale image, each pixel has a value between 0 and 255, where zero corresponds to “black” and 255 corresponds to “white”.
3. The values between 0 and 255 are varying shades of gray, where values closer to 0 are darker and values closer to 255 are lighter.
4. For RGB (Red, Green, Blue) color images, typically each color channel (Red, Green, Blue) is represented using 8 bits, giving a total of 24 bits for the three channels combined. Grayscale images typically use only one channel to represent the intensity of light at each pixel. Hence, it is represented only using 8-bit.

Why is grayscale needed for image processing?

1. It helps in simplifying algorithms and as well eliminates the complexities related to computational requirements.
2. It makes room for easier learning for those who are new to image processing.
3. It enhances easy visualization. It differentiates between the shadow details and the highlights of an image because it is mainly in 2 spatial dimensions (2D) rather than 3D.
4. Color complexity is also reduced.

Methods available to convert an RGB image to grayscale

1. Luminosity Method:

   I = 0.21 * R + 0.72 * G + 0.07 * B

2. Average Method:

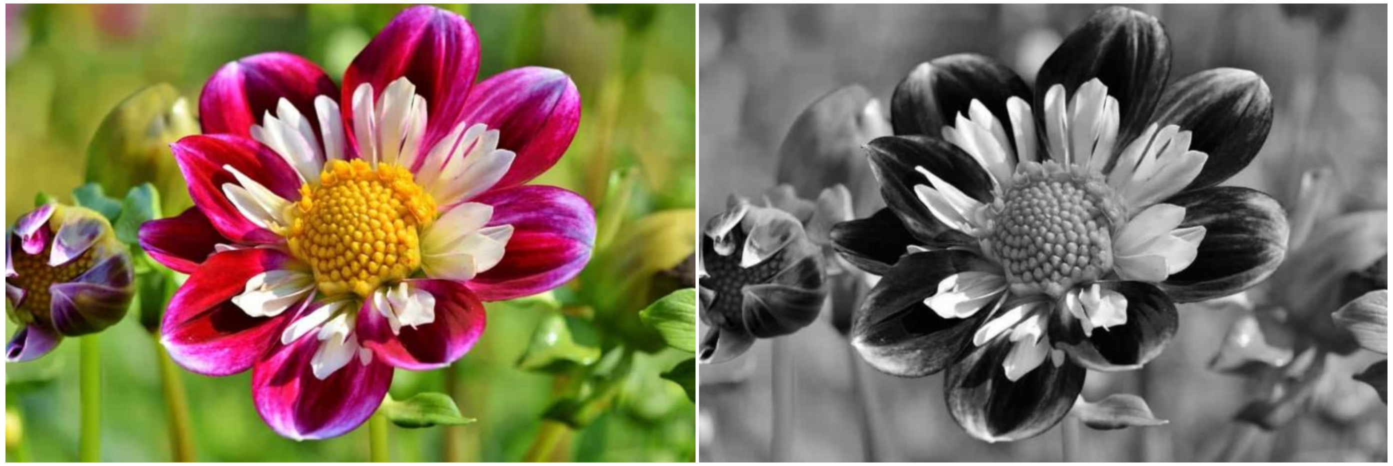
   The grayscale value of each pixel is computed as the average of its red, green, and blue components:
   I = (R + G + B) / 3

3. Single channel extraction:

   Simply take one of the RGB channels (usually green, as the human eye is most sensitive to green) as the grayscale image.

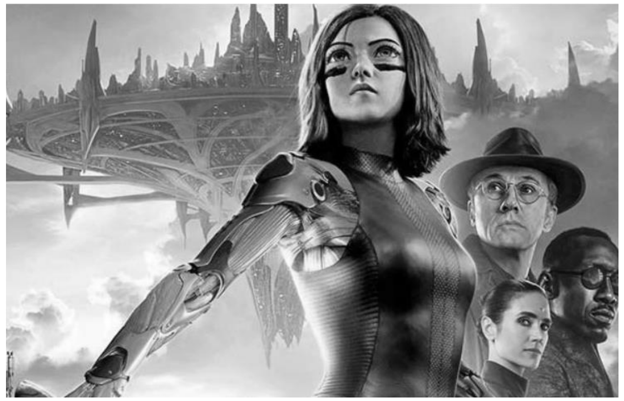
Luminosity Method

```
import numpy as np
import cv2

def luminosity_method(img):
    # Extracting the Red, Green, and Blue channels
    R = img[:,:,0]
    G = img[:,:,1]
    B = img[:,:,2]

    # Applying the luminosity method formula
    grayscale_img = 0.21 * R + 0.72 * G + 0.07 * B

    # Convert the grayscale image to uint8 datatype
    grayscale_img = np.uint8(grayscale_img)

    return grayscale_img

color_img = cv2.imread('flower1.jpg')

# Convert the color image to grayscale using the luminosity method
grayscale_img_lum = luminosity_method(color_img)

# Display the original and grayscale images
cv2.imshow('Color Image', color_img)
cv2.imshow('Grayscale Image (Luminosity Method)', grayscale_img_lum)
```



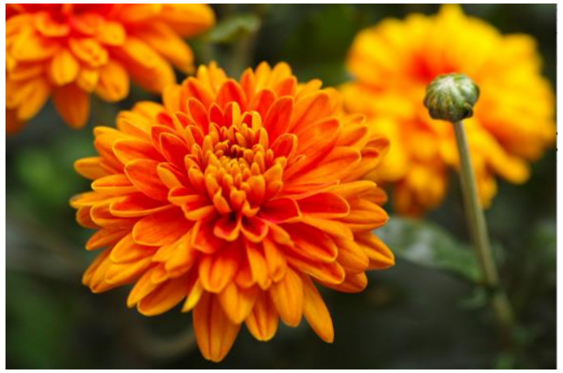
Average Method:

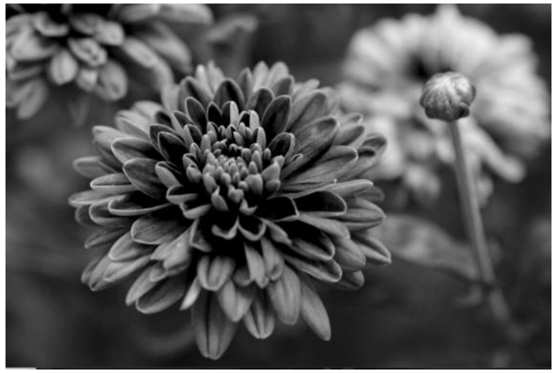
```
import numpy as np
import cv2

def average_method(img):
    # Calculate the average of RGB channels
    grayscale_img = np.mean(img, axis=2)

    # Convert the grayscale image to uint8 datatype
    grayscale_img = np.uint8(grayscale_img)

    return grayscale_img

# Read the color image
color_img = cv2.imread('animated1.jpg')

# Convert the color image to grayscale using the average method
grayscale_img_avg = average_method(color_img)

# Display the original and grayscale images
cv2.imshow('Color Image', color_img)
cv2.imshow('Grayscale Image (Average Method)', grayscale_img_avg)
```

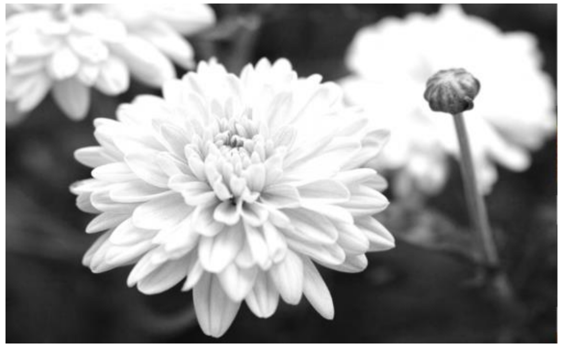


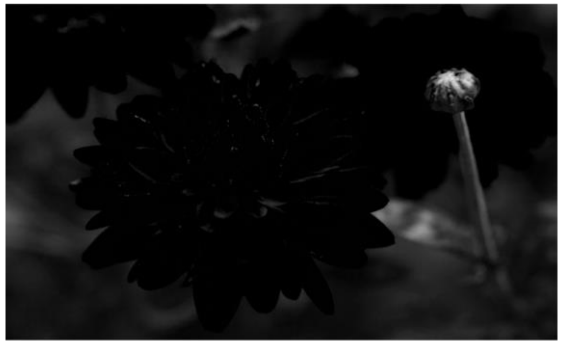
Single Channel Method

```
import cv2

# Read the color image
color_img = cv2.imread('flower.jpg')

green_channel = color_img[:,:,1]

# Extract the red channel
red_channel = color_img[:,:,2]

# Extract the blue channel
blue_channel = color_img[:,:,0]

# Display the original and grayscale images
cv2.imshow('Color Image', color_img)
cv2.imshow('Grayscale Image (Single Channel - Green)',
green_channel)
cv2.imshow('Grayscale Image (Single Channel - Red)',
red_channel)
cv2.imshow('Grayscale Image (Single Channel - Blue)',
blue_channel)
```


Color Image

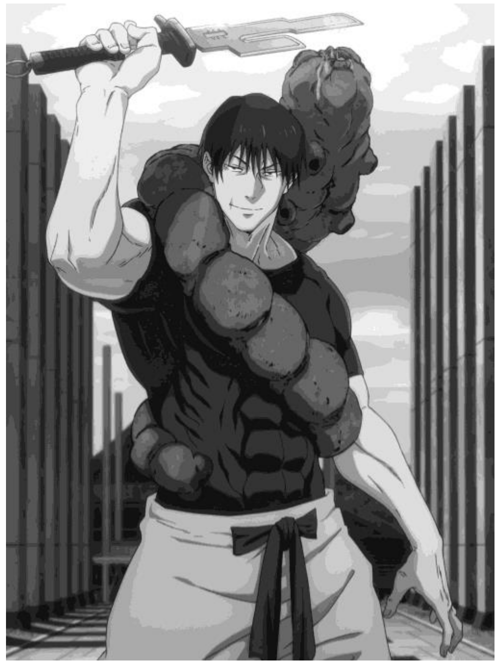

Green Channel


Red Channel


Blue Channel

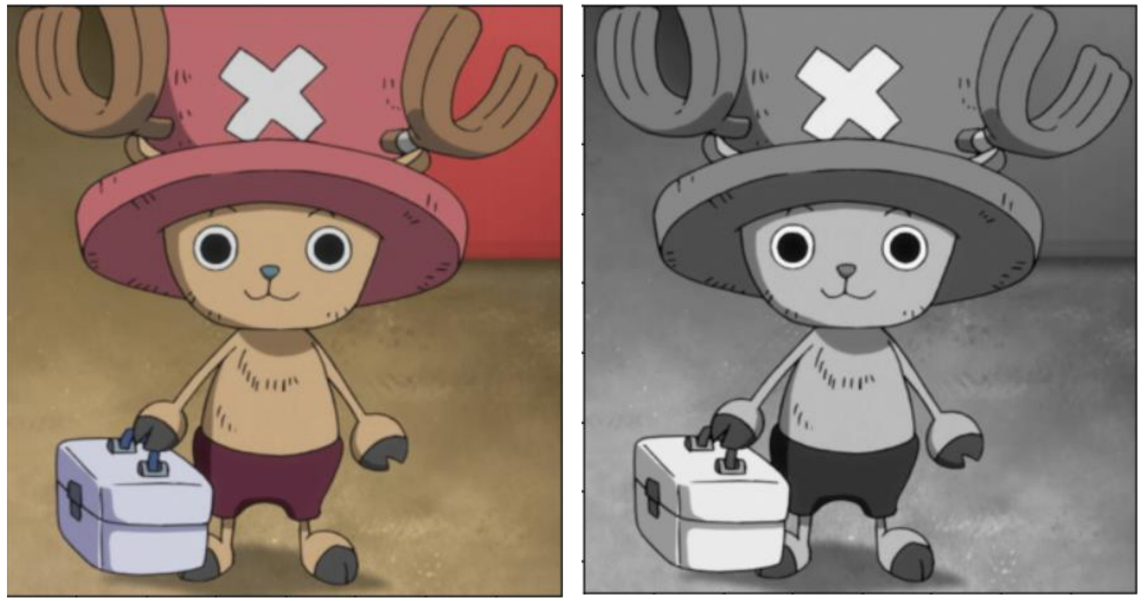
Pillow:

```
from PIL import Image
img = Image.open('toji.webp').convert('L')
img.save('greyscale.png')
```



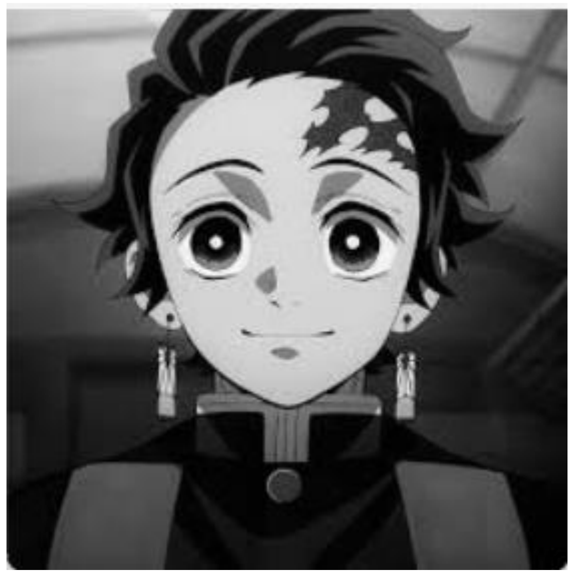
Using Matplotlib and Scikit Learn

```
import matplotlib.pyplot as plt
from skimage import io
from skimage import data
from skimage.color import rgb2gray

rgb_img  = io.imread('chopper.webp’)

gray_img  = rgb2gray(rgb_img)
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].imshow(rgb_img)
ax[0].set_title("Original image")
ax[1].imshow(gray_img, cmap=plt.cm.gray)
ax[1].set_title("Processed image")

fig.tight_layout()
plt.show()
```



Using cv2

```
import cv2

# Load the input image
image = cv2.imread('tanjiro.jfif')
cv2.imshow('Original', image)
cv2.waitKey(0)

# Use the cvtColor() function to grayscale the image
gray_image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)

cv2.imshow('Grayscale', gray_image)
cv2.waitKey(0)

# Window shown waits for any key pressing event
cv2.destroyAllWindows()
```



Applications of Grayscale Images:

- **Medical Imaging:** Grayscale images are extensively used in medical imaging for tasks such as X-rays, MRIs, CT scans, and ultrasound imaging.

- **Document Processing:** Grayscale images are often used in document processing applications such as scanning and OCR (Optical Character Recognition). They can help in extracting text and important features from documents.

- **Image Processing and Computer Vision:** Grayscale images are widely used in image processing and computer vision tasks such as object detection, image segmentation, object recognition, and feature extraction. Their simplicity makes them computationally less expensive to work with compared to full-color images.

THANK YOU