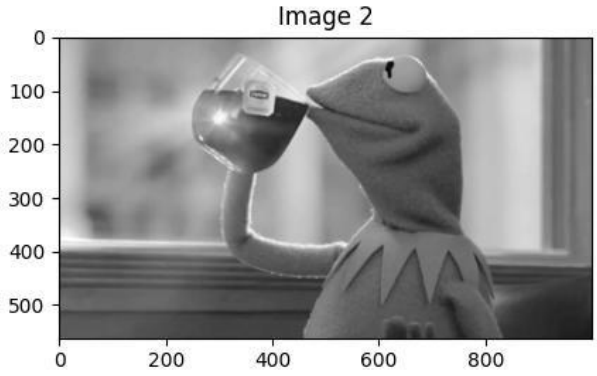
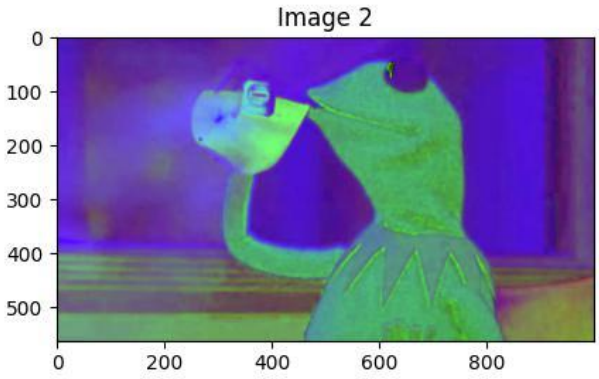
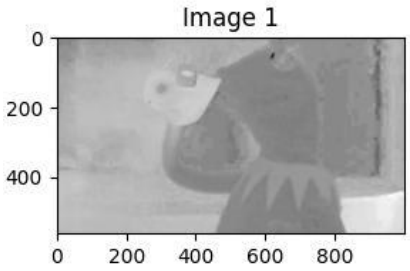
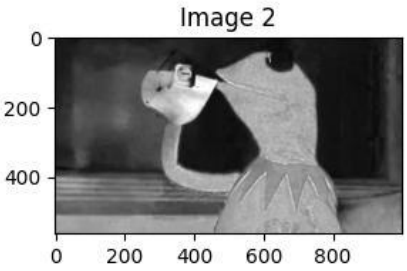
```python
import numpy as np
import pandas as pd
import cv2
from matplotlib import pyplot as plt
from pylab import imread
from skimage.color import rgb2gray
def imshows(ImageData, LabelData, rows, cols, gridType = False):

    ImageArray = list(ImageData)
    LabelArray = list(LabelData)

    if(rows == 1 & cols == 1):
        fig = plt.figure(figsize=(20,20))
    else:
        fig = plt.figure(figsize=(cols*8,rows*5))

    for i in range(1, cols * rows + 1):
        fig.add_subplot(rows, cols, i)
        image = ImageArray[i - 1]
        if (len(image.shape) < 3):
            plt.imshow(image, plt.cm.gray)
            plt.grid(gridType)
        else:
            plt.imshow(image)
            plt.grid(gridType)
        plt.title(LabelArray[i - 1])

    plt.show()
def ShowThreeImages(IM1, IM2, IM3):
    imshows([IM1, IM2, IM3], ["Image 1","Image 2", "Image 3"], 1, 3)
def ShowTwoImages(IM1, IM2):
    imshows([IM1, IM2], ["Image 1","Image 2"], 1, 2)
# Read Image
image_color = imread("C:/Users/Ruby/Downloads/Digital Image Processing/Lab01/Sample01/kermit.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)
# Display Image
ShowTwoImages(image_color, image_gray)
```

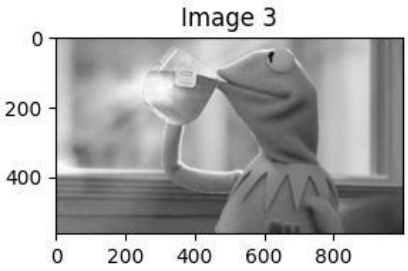


```python
# Convert Image into HSV color spaces
image_hsv = cv2.cvtColor(image_color, cv2.COLOR_BGR2HSV)
ShowTwoImages(image_color, image_hsv)
# Show each channel H , S and V
ShowThreeImages(image_hsv[:,:,0],image_hsv[:,:,1],image_hsv[:,:,2])
```

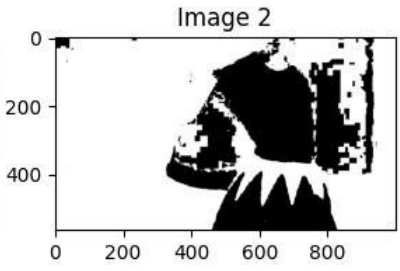


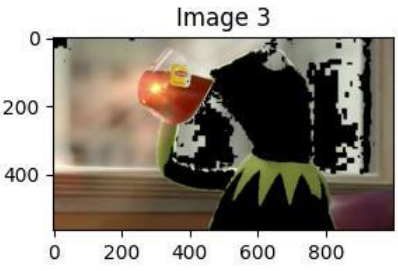




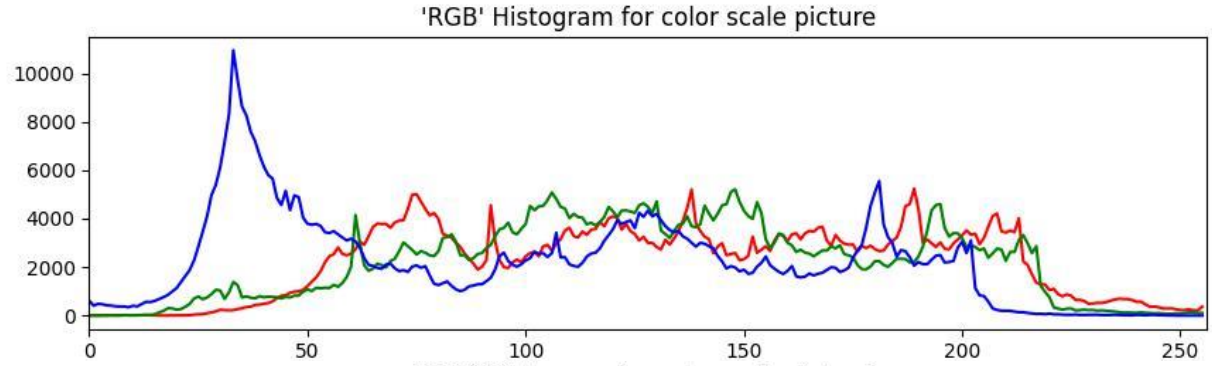


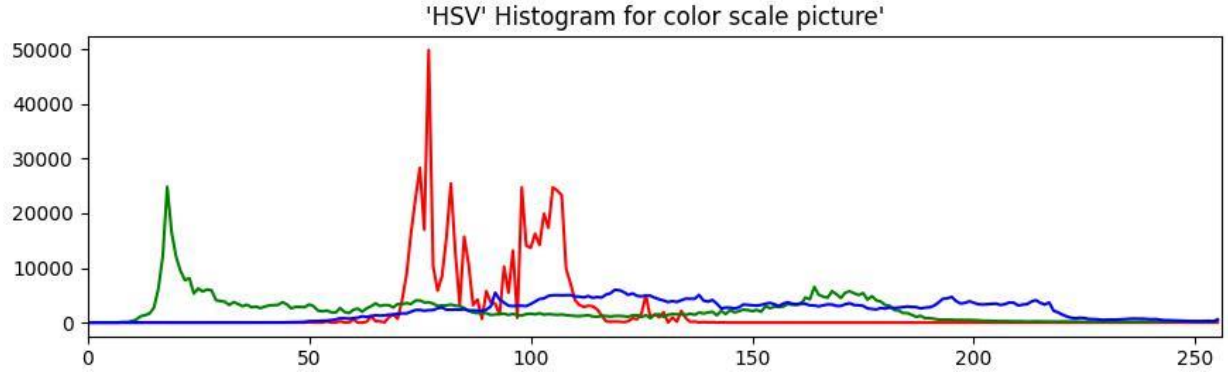

```python
hue_img = image_hsv[:,:,0]
hue_threshold = 80

hue_binary = hue_img > hue_threshold

def SegmentColorImageByMask(IM, Mask):
    Mask = Mask.astype(np.uint8)
    result = cv2.bitwise_and(IM, IM, mask = Mask)
    return result

hue_binary01_rgb = SegmentColorImageByMask(image_color, hue_binary01)
ShowThreeImages(image_color, hue_binary, hue_binary_rgb)
```



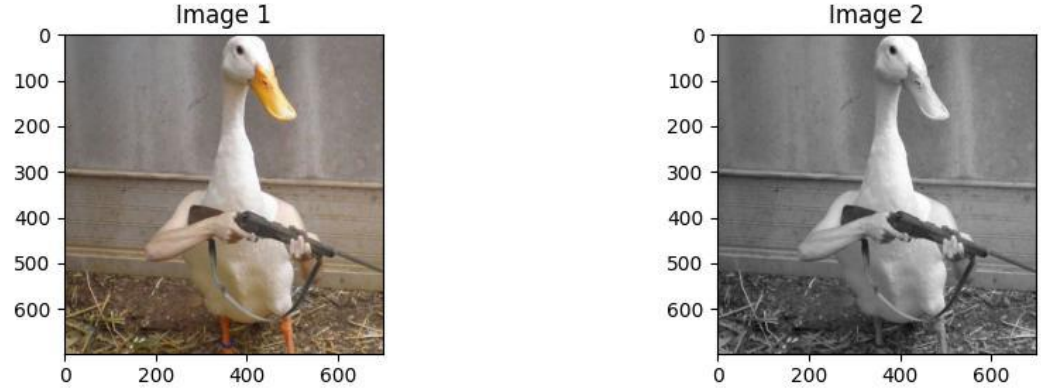


```
    color = ('r', 'g', 'b')
for channel,col in enumerate(color):
    histr = cv2.calcHist([image_color],[channel],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.title("'RGB' Histogram for color scale picture")
plt.show()

color = ('r', 'g', 'b')
for channel,col in enumerate(color):
    histr = cv2.calcHist([image_hsv],[channel],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.title("'HSV' Histogram for color scale picture'")
plt.show()
```

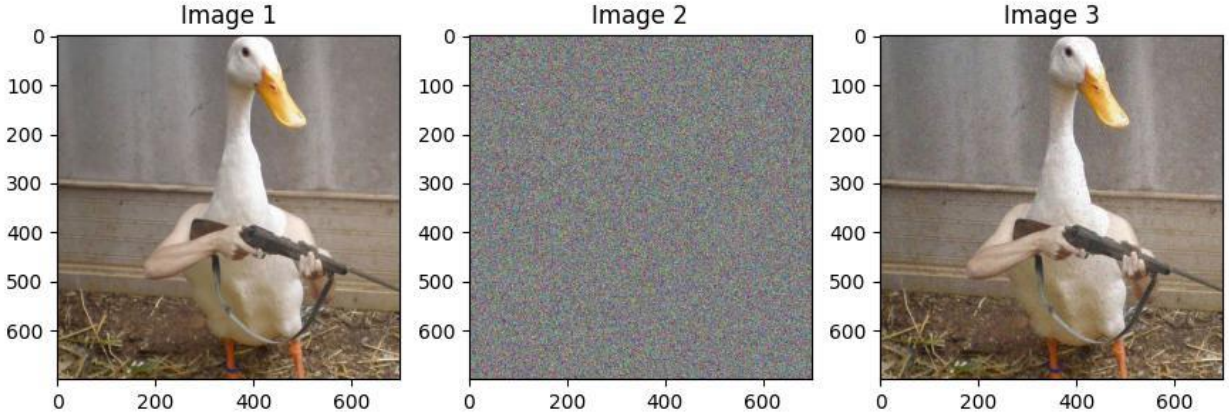




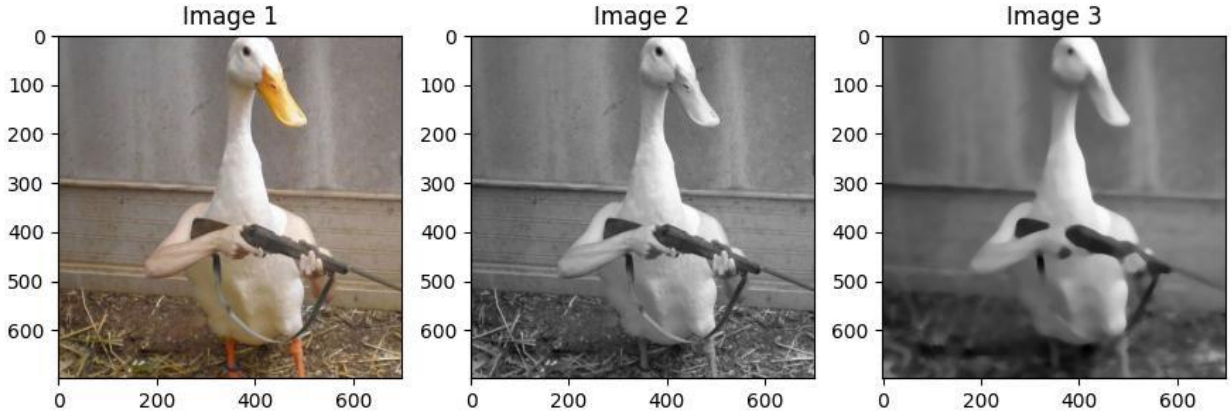
```
# Read Image
image_color = imread("C:/Users/Ruby/Downloads/Digital Image Processing/Lab01/Sample01/duck.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)
# Display Image
ShowTwoImages(image_color, image_gray)
```



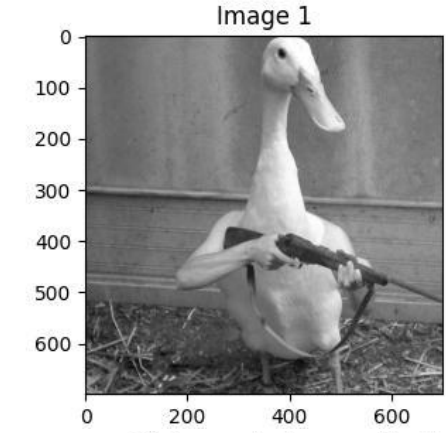

```
# Create Noise Image
noise = np.random.random(image_color.shape)
image_noise = image_color.copy()
image_noise[noise > 0.99] = 255
image_noise[noise < 0.01] = 0
ShowThreeImages(image_color, noise, image_noise)
```

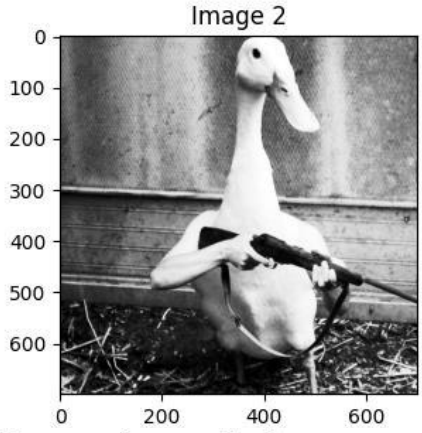


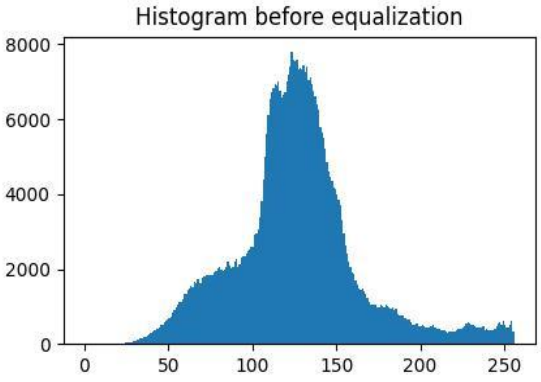

```
# Create Blurred Image
from skimage.filters.rank import median
from skimage.morphology import disk
image_blurred = median(image_gray, disk(10))
ShowThreeImages(image_color, image_gray, image_blurred)
```

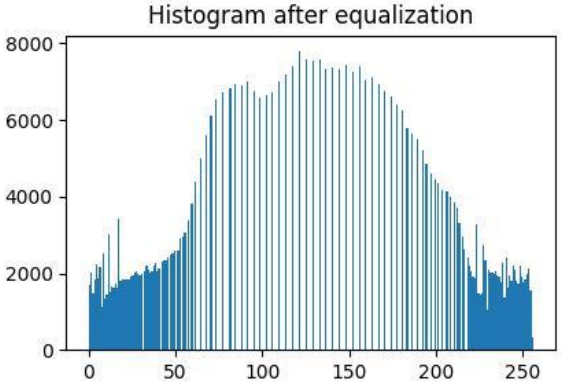


```
117     from skimage import data, exposure
118     image_equalization = exposure.equalize_hist(image_gray)
119     image_equalization = np.float32(image_equalization * 255)
120     ShowTwoImages(image_gray, image_equalization)
121     hist = cv2.calcHist([image_gray],[0],None,[256],[0,256])
122     plt.hist(image_gray.ravel(),256,[0,256])
123     plt.title('Histogram before equalization')
124     plt.show()
125     hist = cv2.calcHist([image_equalization],[0],None,[256],[0,256])
126     plt.hist(image_equalization.ravel(),256,[0,256])
127     plt.title('Histogram after equalization')
128     plt.show()
```

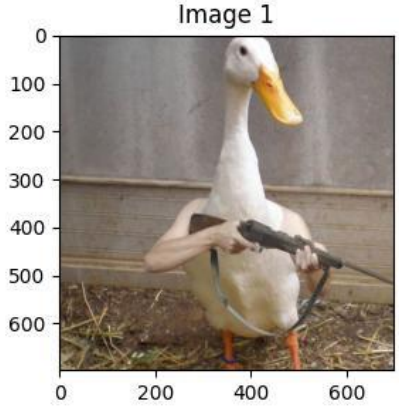


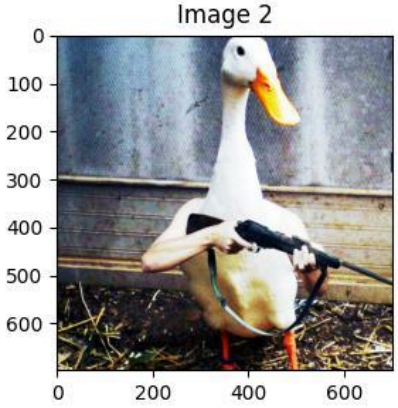




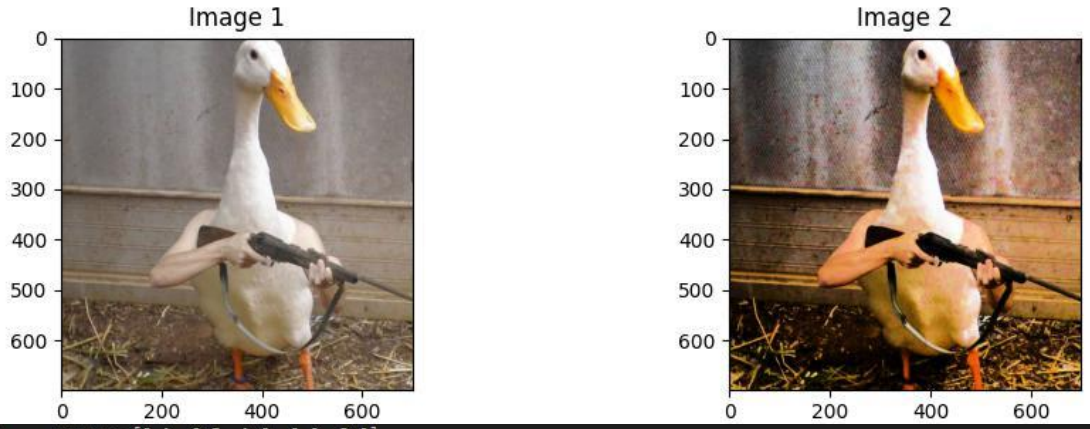



```
130     def histogram_equalize(img):
131         r, g, b = cv2.split(img)
132         red = cv2.equalizeHist(r)
133         green = cv2.equalizeHist(g)
134         blue = cv2.equalizeHist(b)
135         return cv2.merge((red, green, blue))
136     image_equalization_color = histogram_equalize(image_color)
137     ShowTwoImages(image_color, image_equalization_color)
```

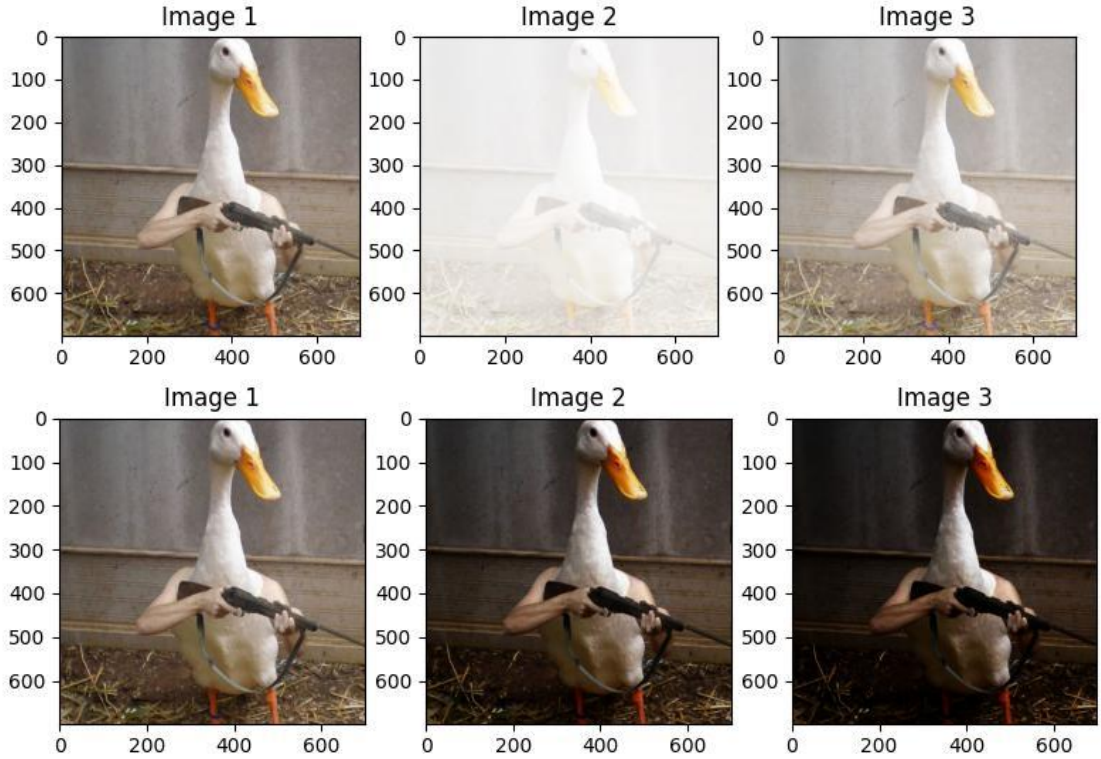




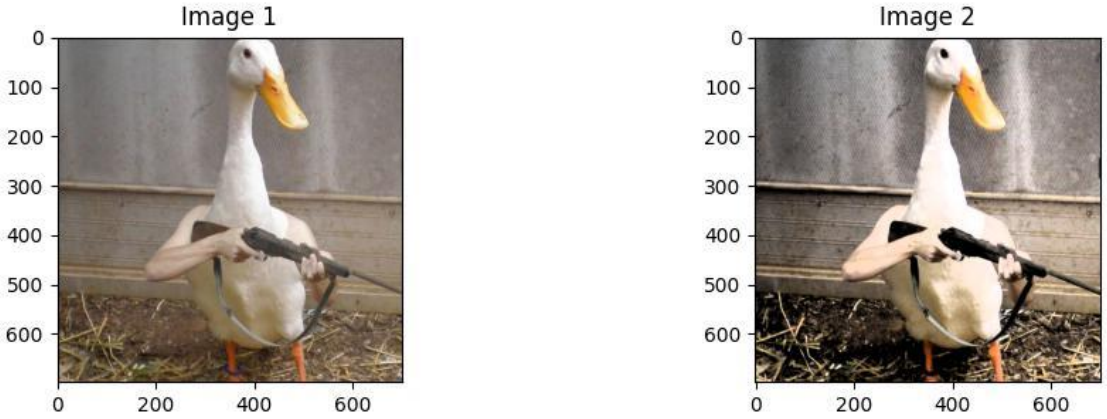
```
    # Convert Image into HSV color spaces
    image_hsv = cv2.cvtColor(image_color, cv2.COLOR_RGB2HSV)
    # Apply histogram equalization
    channel = 1
    image_hsv[:, :, channel] = cv2.equalizeHist(image_hsv[:, :, channel])
    channel = 2
    image_hsv[:, :, channel] = cv2.equalizeHist(image_hsv[:, :, channel])
    # Convert to RGB
    image_enhanced = cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB)
    ShowTwoImages(image_color, image_enhanced)
```




```
    gamma = [0.1, 0.5, 1.2, 2.2, 3.2]
    gamma_corrected_01 = np.array(255*(image_color / 255) ** gamma[0], dtype = 'uint8')
    gamma_corrected_02 = np.array(255*(image_color / 255) ** gamma[1], dtype = 'uint8')
    gamma_corrected_03 = np.array(255*(image_color / 255) ** gamma[2], dtype = 'uint8')
    gamma_corrected_04 = np.array(255*(image_color / 255) ** gamma[3], dtype = 'uint8')
    gamma_corrected_05 = np.array(255*(image_color / 255) ** gamma[4], dtype = 'uint8')
    ShowThreeImages(image_color, gamma_corrected_01, gamma_corrected_02)
    ShowThreeImages(gamma_corrected_03, gamma_corrected_04, gamma_corrected_05)
```



```python
# Function to map each intensity level to output intensity level.
def pixelValTransformation(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2

image_hsv = cv2.cvtColor(image_color, cv2.COLOR_RGB2HSV)
image_hsv_value = image_hsv[:,:,2]
# Define parameters.
r1 = 80
s1 = 0
r2 = 180
s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelValTransformation)
# Apply contrast stretching.
contrast_stretched = pixelVal_vec(image_hsv_value, r1, s1, r2, s2)

image_hsv[:,:,2] = contrast_stretched
image_enhanced = cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB)
ShowTwoImages(image_color, image_enhanced)
```




```python
from skimage import feature
image_edges = feature.canny(image_gray)
ShowTwoImages(image_gray, image_edges)
```