Image Histograms

Concept

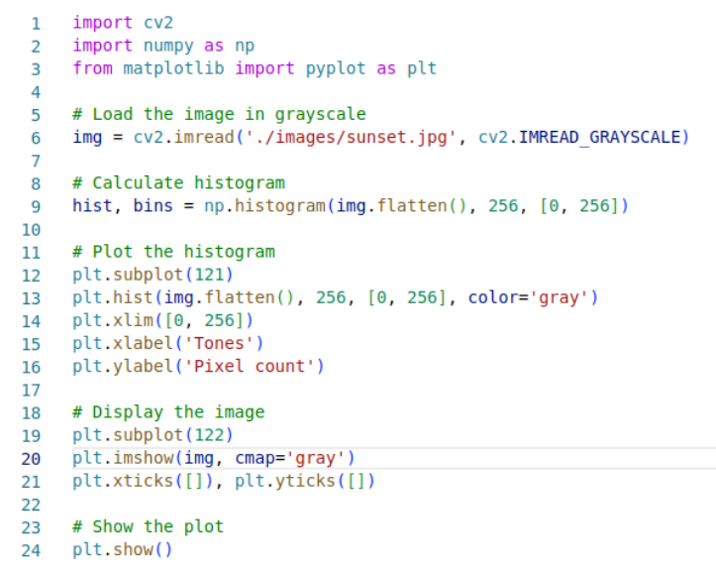
An image histogram is a graphical representation of the distribution of pixel intensity values in an image. It displays the number of pixels at each intensity level, helping to visualize the contrast, brightness, and overall tonal range of an image.

Histograms are utilized to present the statistical information of an image in a visually interpretable format. With the aid of a histogram, specific types of issues in an image can be identified.

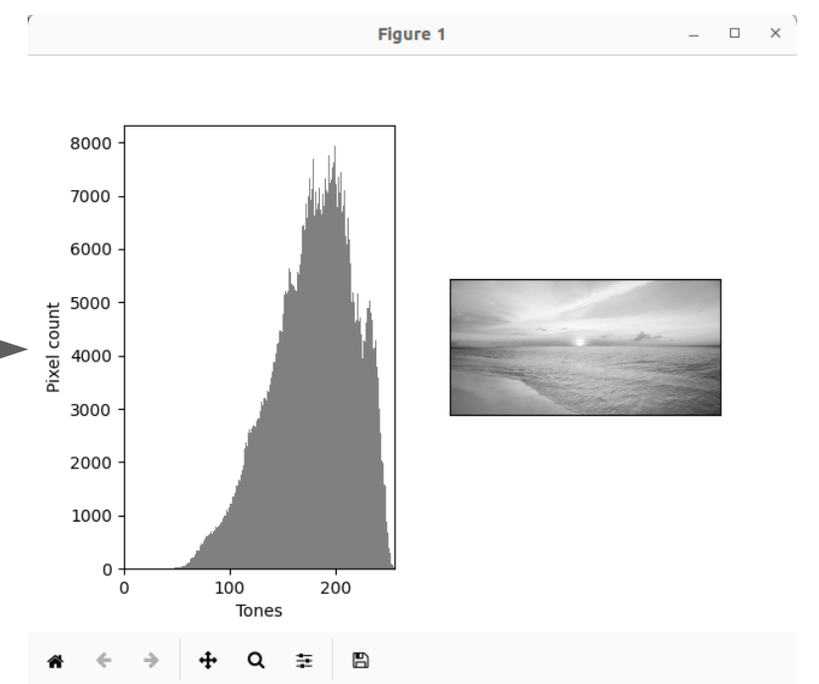
It shows how the shades of gray are distributed in the image and can help to see if the radiometric range of the image is being used in the best possible way. It's like a picture of the distribution of gray shades, which helps to identify where the lightest and darkest points of the image are.

By looking at the histogram and observing the spatial distribution of values, you can get a general idea of the brightness of an image. If the histogram values are concentrated to the left, the image will be darker. If they are concentrated to the right, the image will be brighter.

It's important to note that a histogram only contains statistical information about the distribution of pixels and does not provide spatial information about where those pixels are located within the image. Therefore, it is not possible to reconstruct an image solely from its histogram.



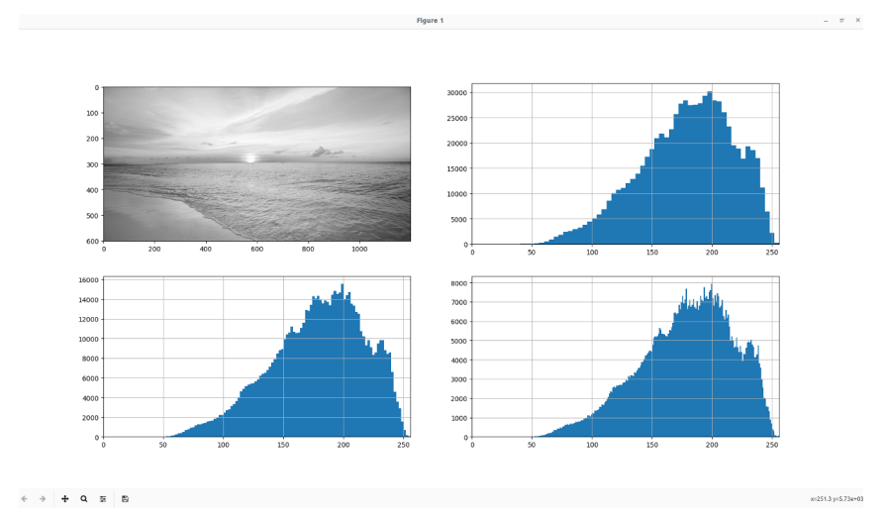
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the image in grayscale
img = cv2.imread('./images/sunset.jpg', cv2.IMREAD_GRAYSCALE)

# Calculate histogram
hist, bins = np.histogram(img.flatten(), 256, [0, 256])

# Plot the histogram
plt.subplot(121)
plt.hist(img.flatten(), 256, [0, 256], color='gray')
plt.xlim([0, 256])
plt.xlabel('Tones')
plt.ylabel('Pixel count')

# Display the image
plt.subplot(122)
plt.imshow(img, cmap='gray')
plt.xticks([]), plt.yticks([])

# Show the plot
plt.show()
```



Terminology

The histogram shows the number of pixels for each pixel value, from 0 to 255. The histogram is divided into subparts, each called a "bin", and the value of each bin is the sum of all pixel counts in it. BINS is represented by the histSize term in OpenCV documentation. DIMS refers to the number of parameters for which we collected data, in this case, only intensity values. RANGE is the range of intensity values to be measured, usually [0, 256], meaning all intensity values.

Reducing BINS (compartiments) can be used for image comparisson` and histogram simplifying

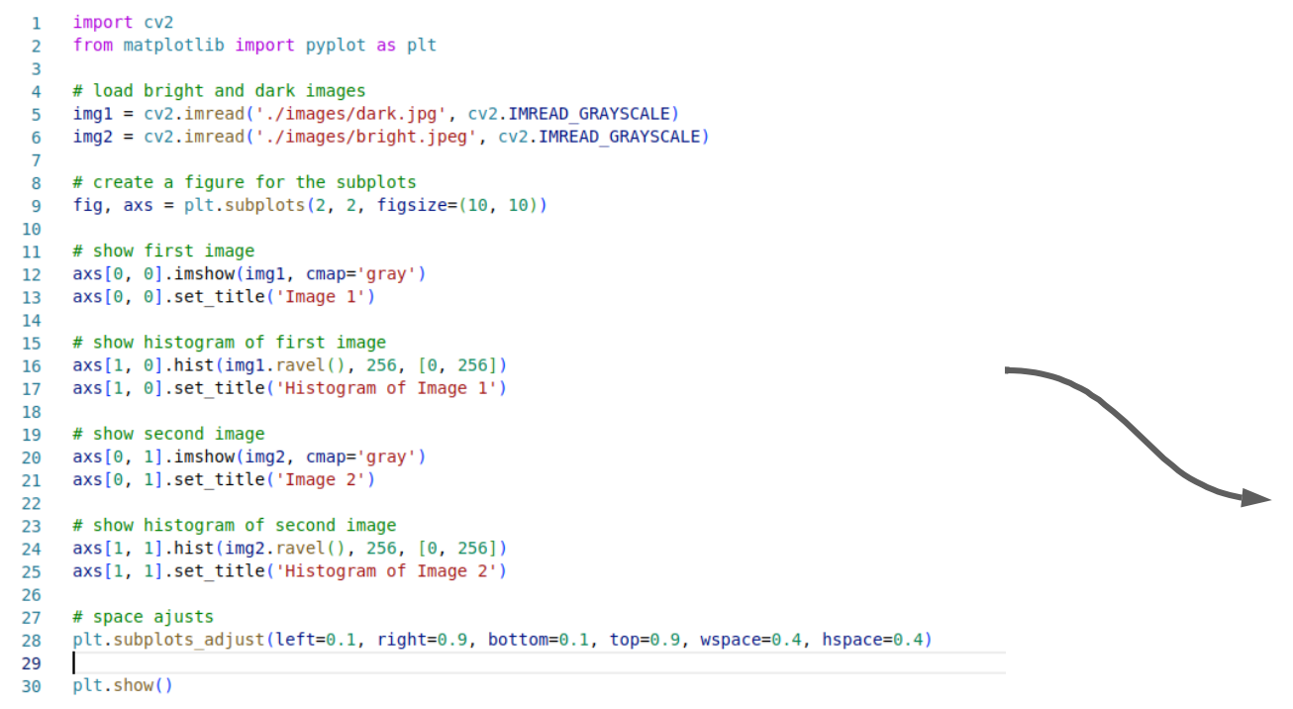


Contrast and Dynamic Range

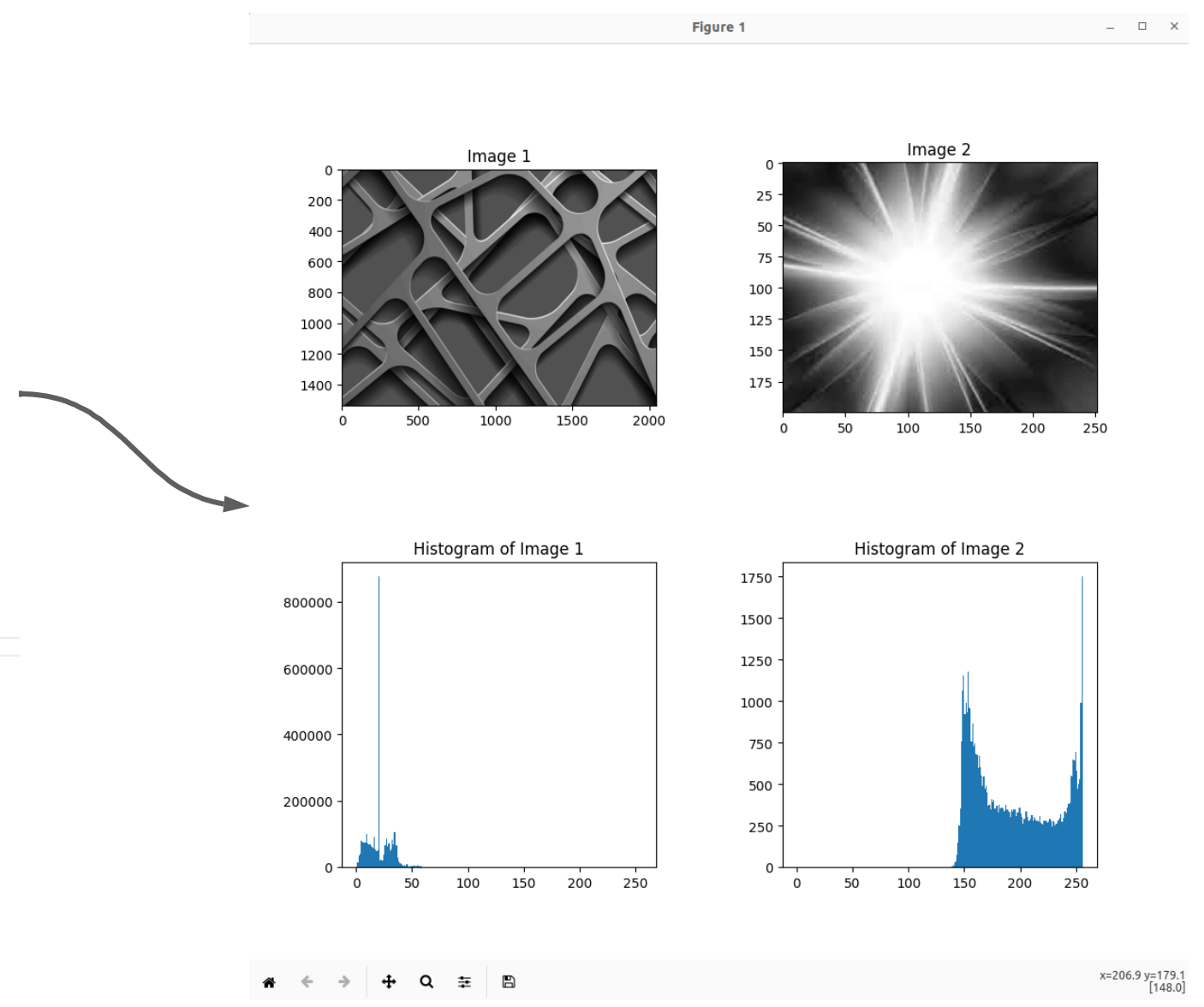
Contrast refers to the range of intensity values that are used in a given image. It is the difference between the minimum and maximum pixel values of the image. A fully contrasted image makes efficient use of the entire range of intensity values available (from black to white). The dynamic range of an image is the number of distinct pixel values in the image. Ideally, the dynamic range spans all usable pixel values, in which case the full range is utilized. When an image has a contrast range available between low and high values, the maximum dynamic range is achieved when all intensity values within this range are used.

Photography

Professional photographers who use high-end cameras always have access to a live histogram of the image being captured. This helps them to adjust the camera settings and utilize the color or grayscale channel in the best possible way to enhance the image display. The histogram provides imaging acquisition technicians with a view of how the light entering the camera was captured by the camera sensor. Is the image to bright or dark? for example

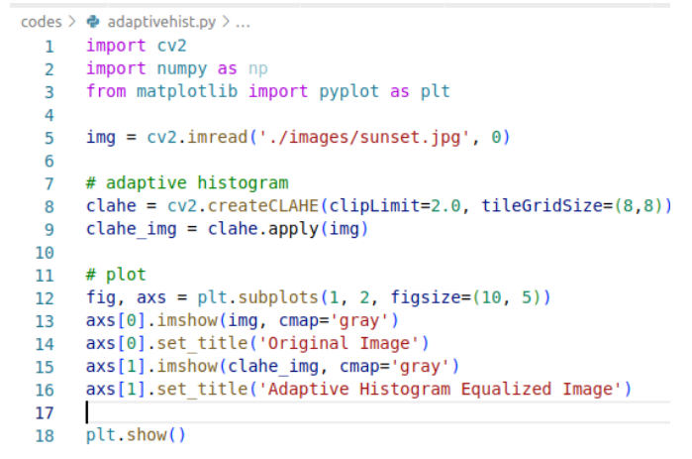


```python
import cv2
from matplotlib import pyplot as plt

# load bright and dark images
img1 = cv2.imread('./images/dark.jpg', cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('./images/bright.jpeg', cv2.IMREAD_GRAYSCALE)

# create a figure for the subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# show first image
axs[0, 0].imshow(img1, cmap='gray')
axs[0, 0].set_title('Image 1')

# show histogram of first image
axs[1, 0].hist(img1.ravel(), 256, [0, 256])
axs[1, 0].set_title('Histogram of Image 1')

# show second image
axs[0, 1].imshow(img2, cmap='gray')
axs[0, 1].set_title('Image 2')

# show histogram of second image
axs[1, 1].hist(img2.ravel(), 256, [0, 256])
axs[1, 1].set_title('Histogram of Image 2')

# space ajusts
plt.subplots_adjust(left=0.1, right=0.9, bottom=0.1, top=0.9, wspace=0.4, hspace=0.4)

plt.show()
```

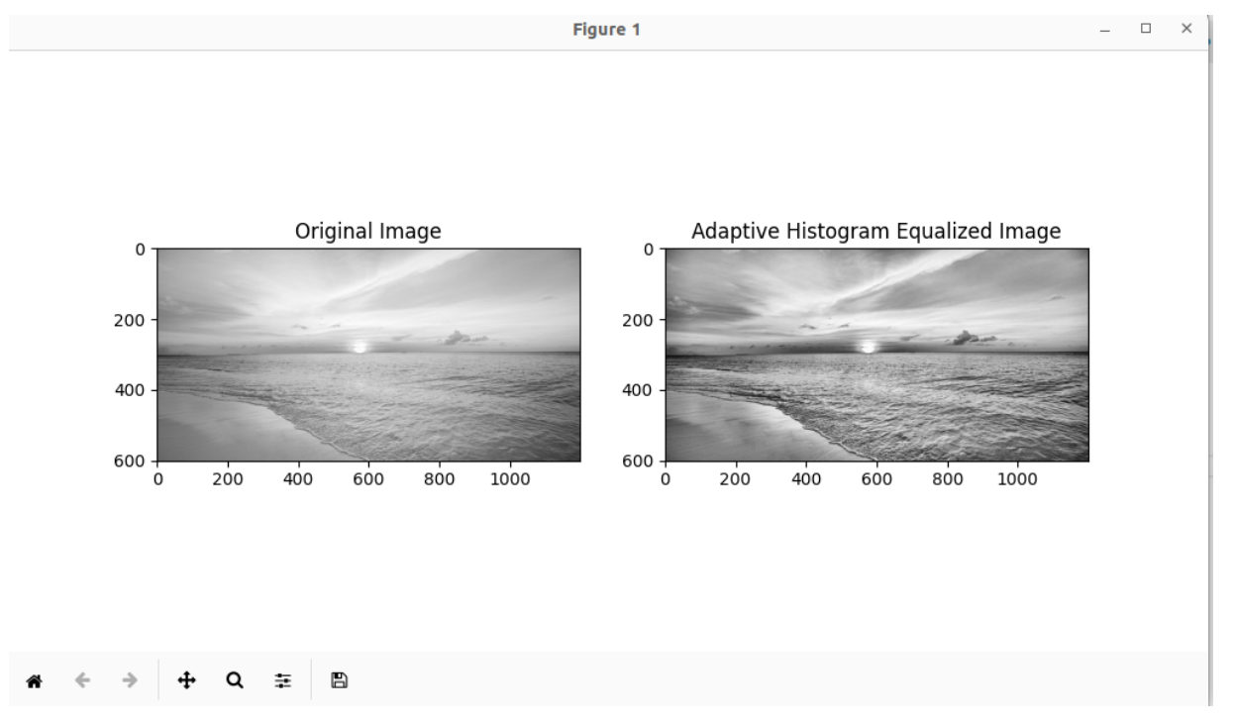


Adaptive Histogram

Adaptive histogram is a technique of image processing that aims to improve the contrast and brightness of an image by adjusting the histogram in a local manner. Unlike global histogram equalization, adaptive histogram equalization divides the image into smaller regions and applies histogram equalization to each region independently, allowing for a better preservation of local contrast and avoiding the over-amplification of noise that can occur in the global method. This technique is particularly useful for images with non-uniform lighting or with regions of significantly different illumination.



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('./images/sunset.jpg', 0)

# adaptive histogram
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
clahe_img = clahe.apply(img)

# plot
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(img, cmap='gray')
axs[0].set_title('Original Image')
axs[1].imshow(clahe_img, cmap='gray')
axs[1].set_title('Adaptive Histogram Equalized Image')

plt.show()
```



HIstograms with more then 1 bye per pixel

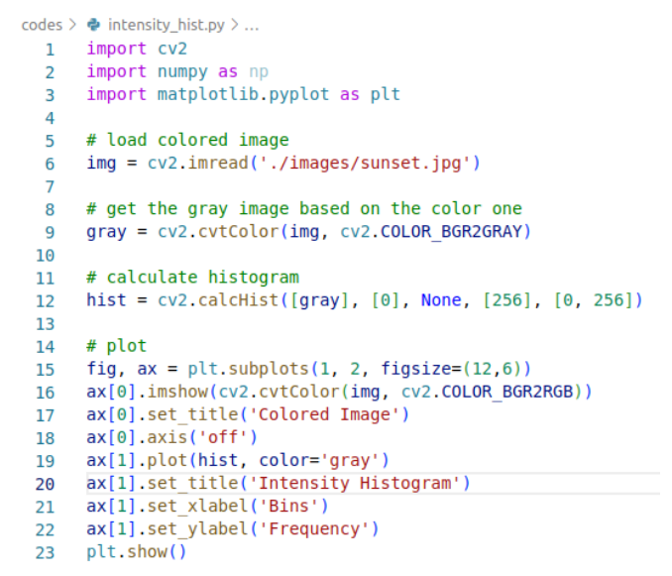
While some modern scanners can use 12 or 16 bits to capture more precise intensity values, the benefits of higher bit depths are limited and can lead to compatibility issues with some software and file formats. Additionally, human eyes cannot discern more than 256 levels of gray on a computer screen, so using higher bit depths does not necessarily result in better image quality.

While increased depth may not be necessary for everyday applications, there are certain fields such as medicine and astronomy where it is crucial. In these specialized areas, greater depth may lead to more precise measurements and analyses, ultimately leading to more accurate results and potentially life-saving discoveries.

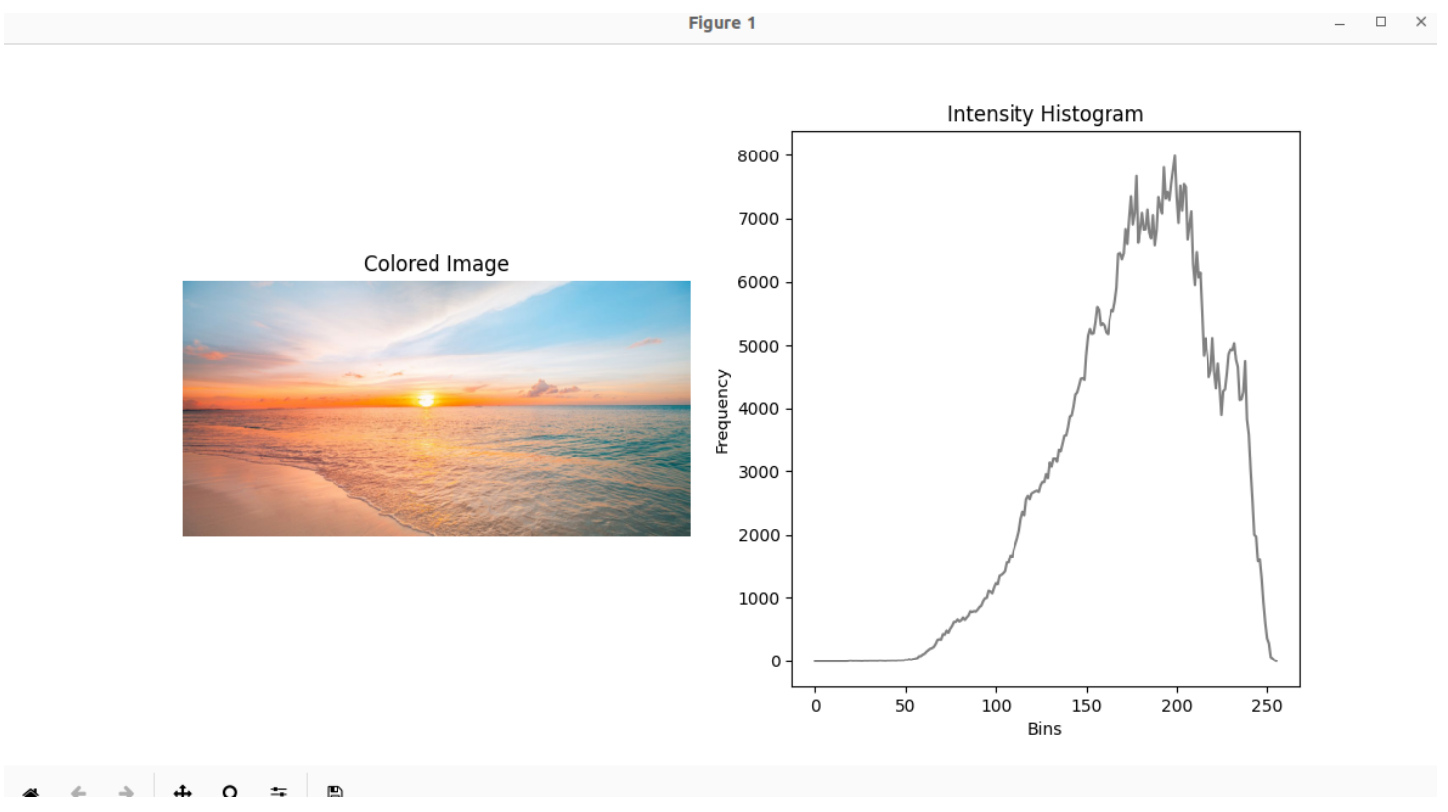
Histograms of Color Images

Intensity Histograms

In summary, the intensity or luminance histogram of a colored image is the same as the histogram of the corresponding grayscale image. The grayscale image is obtained by calculating the luminance of the individual channels of the colored image using a weighted sum that takes into account the theory of color perception.



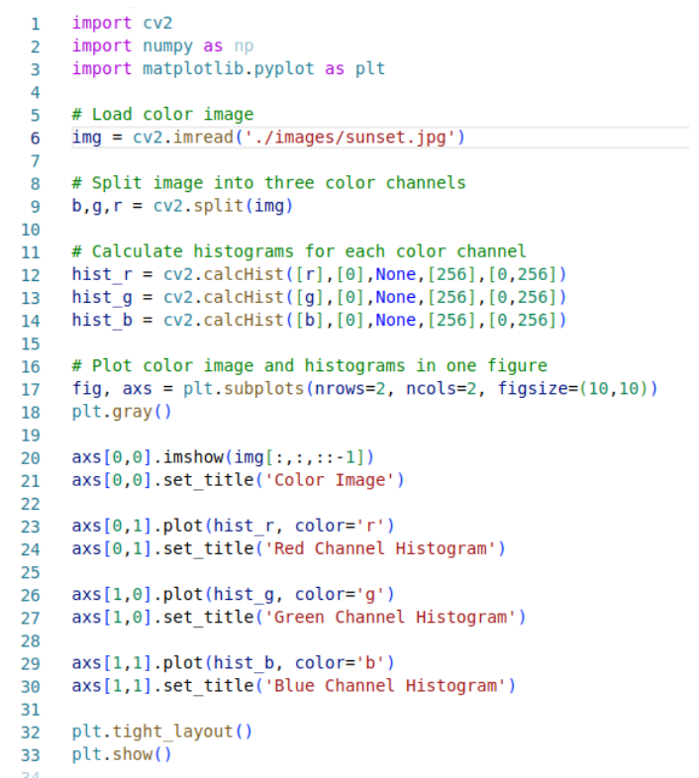
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# load colored image
img = cv2.imread('./images/sunset.jpg')

# get the gray image based on the color one
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# calculate histogram
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])

# plot
fig, ax = plt.subplots(1, 2, figsize=(12,6))
ax[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
ax[0].set_title('Colored Image')
ax[0].axis('off')
ax[1].plot(hist, color='gray')
ax[1].set_title('Intensity Histogram')
ax[1].set_xlabel('Bins')
ax[1].set_ylabel('Frequency')
plt.show()
```



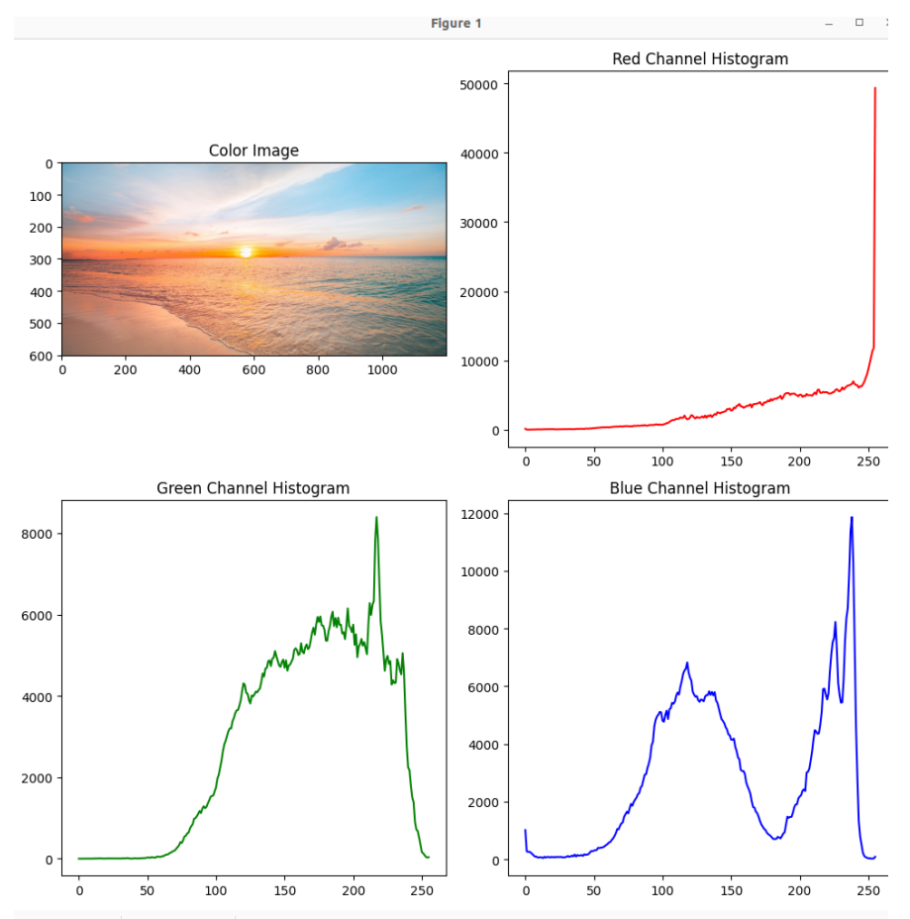
Individual Color Channel Histograms

Even though the luminance histogram takes into account all color channels, image errors that appear in single channels can remain unnoticed.

Luminance histograms and component histograms provide useful information about the lighting, contrast, dynamic range, and saturation effects related to individual color components. However, it's important to remember that they do not provide information about the actual color distribution in the image, since they are based on individual color channels and not the combination of individual channels that form the color of an individual pixel.

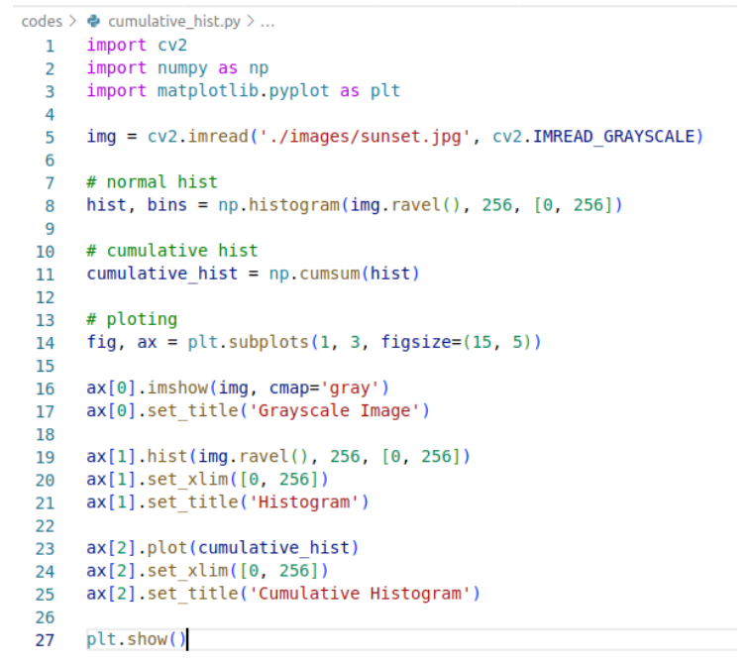


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load color image
img = cv2.imread('./images/sunset.jpg')

# Split image into three color channels
b,g,r = cv2.split(img)

# Calculate histograms for each color channel
hist_r = cv2.calcHist([r],[0],None,[256],[0,256])
hist_g = cv2.calcHist([g],[0],None,[256],[0,256])
hist_b = cv2.calcHist([b],[0],None,[256],[0,256])

# Plot color image and histograms in one figure
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(10,10))
plt.gray()

axs[0,0].imshow(img[:,:,::-1])
axs[0,0].set_title('Color Image')

axs[0,1].plot(hist_r, color='r')
axs[0,1].set_title('Red Channel Histogram')

axs[1,0].plot(hist_g, color='g')
axs[1,0].set_title('Green Channel Histogram')

axs[1,1].plot(hist_b, color='b')
axs[1,1].set_title('Blue Channel Histogram')

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

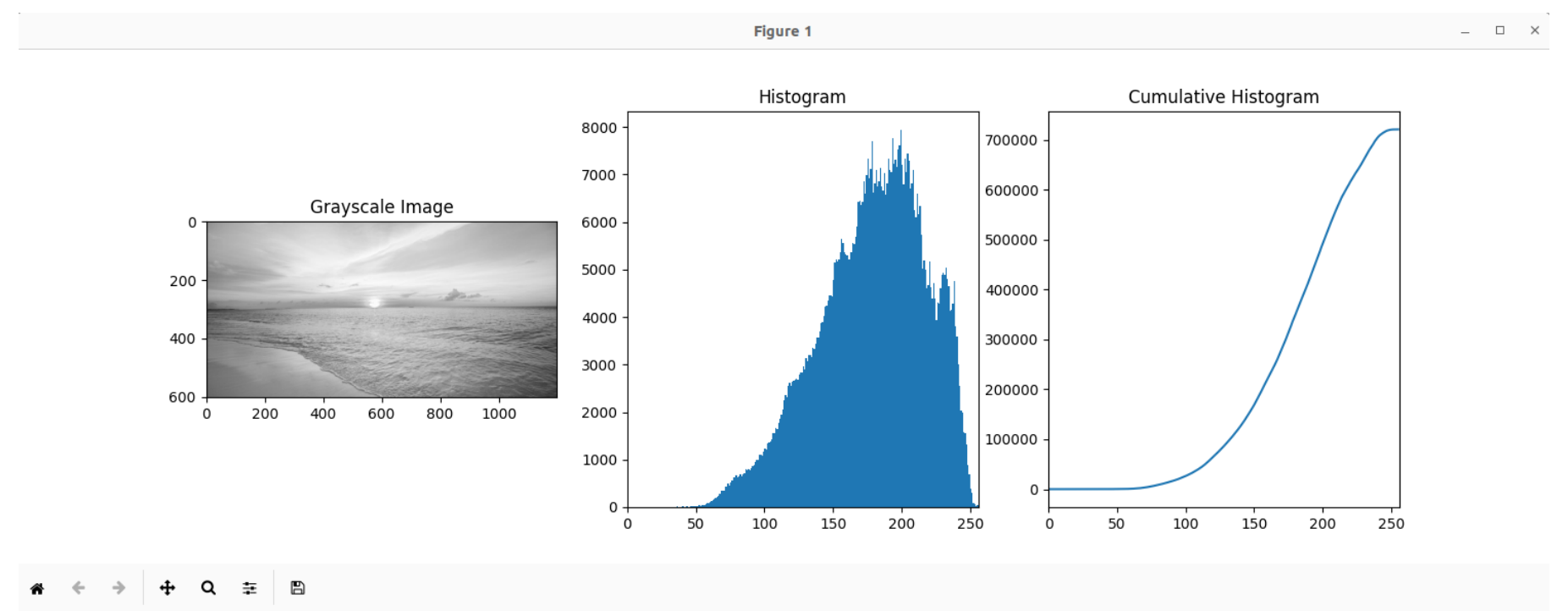


Cumulative Histogram

The cumulative histogram, which is derived from the regular histogram, is useful when performing certain image operations involving histograms, such as histogram equalization. The cumulative histogram is defined as the sum of all histogram values up to a given intensity level. It is denoted by H(i), where i is the intensity level and H(j) is the histogram value for intensity level j. The cumulative histogram is a useful tool in image processing for enhancing contrast and adjusting brightness.

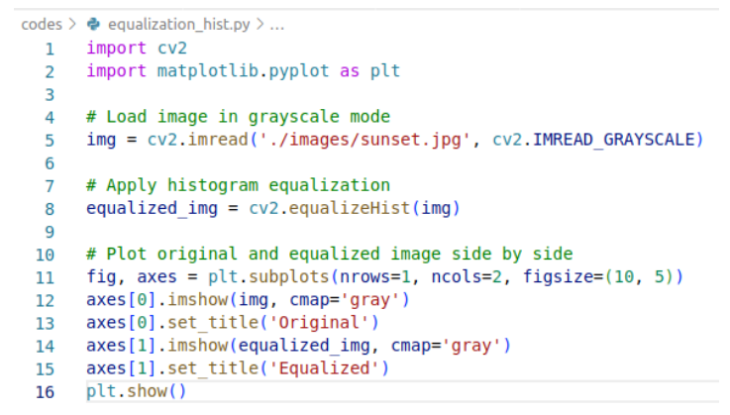


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('./images/sunset.jpg', cv2.IMREAD_GRAYSCALE)

# normal hist
hist, bins = np.histogram(img.ravel(), 256, [0, 256])

# cumulative hist
cumulative_hist = np.cumsum(hist)

# ploting
fig, ax = plt.subplots(1, 3, figsize=(15, 5))

ax[0].imshow(img, cmap='gray')
ax[0].set_title('Grayscale Image')

ax[1].hist(img.ravel(), 256, [0, 256])
ax[1].set_xlim([0, 256])
ax[1].set_title('Histogram')

ax[2].plot(cumulative_hist)
ax[2].set_xlim([0, 256])
ax[2].set_title('Cumulative Histogram')

plt.show()
```

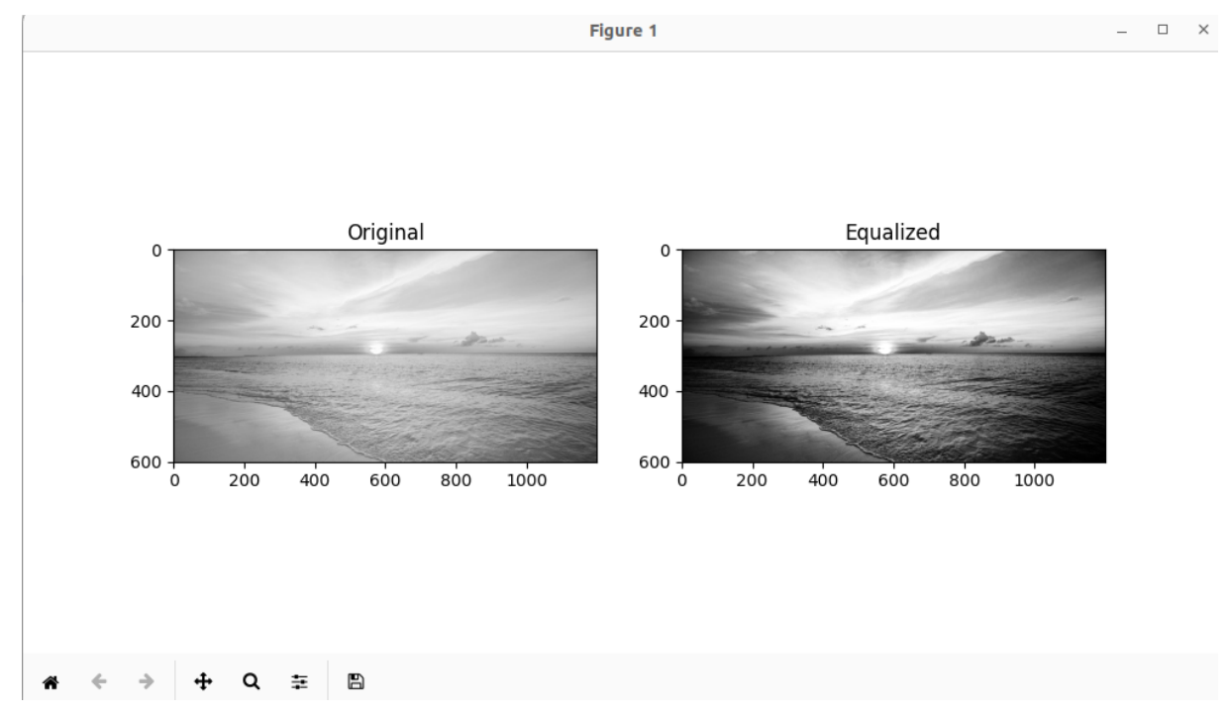


Histogram Equalization

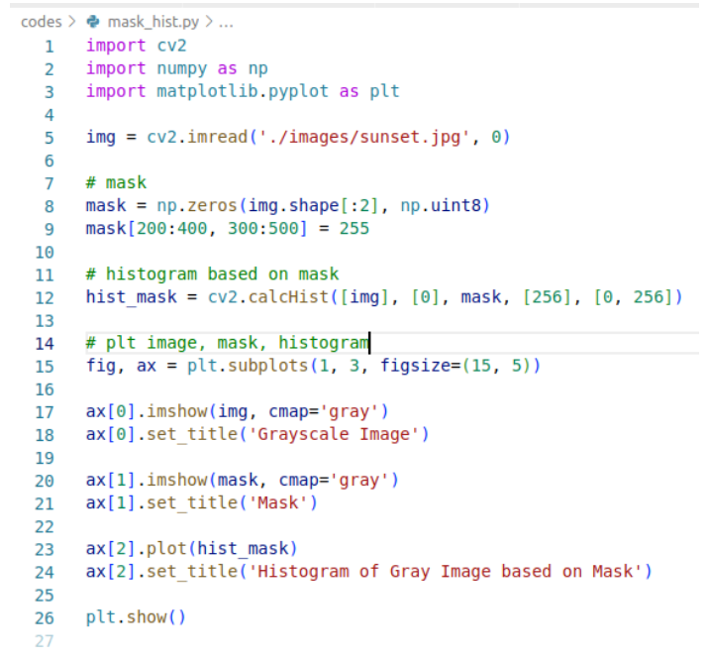
Histogram equalization is a technique used in image processing to enhance the contrast of an image by adjusting the intensities of its pixels. The main goal of histogram equalization is to obtain a more uniform distribution of pixel intensities in the image, which can result in an improvement in the overall image quality and make details that were previously difficult to see more visible.



```python
import cv2
import matplotlib.pyplot as plt

# Load image in grayscale mode
img = cv2.imread('./images/sunset.jpg', cv2.IMREAD_GRAYSCALE)

# Apply histogram equalization
equalized_img = cv2.equalizeHist(img)

# Plot original and equalized image side by side
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
axes[0].imshow(img, cmap='gray')
axes[0].set_title('Original')
axes[1].imshow(equalized_img, cmap='gray')
axes[1].set_title('Equalized')
plt.show()
```



The histogram of a particular mask or region of interest (ROI) can be useful in some cases to evaluate the local histogram in a given area.



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

img = cv2.imread('./images/sunset.jpg', 0)

# mask
mask = np.zeros(img.shape[:2], np.uint8)
mask[200:400, 300:500] = 255

# histogram based on mask
hist_mask = cv2.calcHist([img], [0], mask, [256], [0, 256])

# plt image, mask, histogram
fig, ax = plt.subplots(1, 3, figsize=(15, 5))

ax[0].imshow(img, cmap='gray')
ax[0].set_title('Grayscale Image')

ax[1].imshow(mask, cmap='gray')
ax[1].set_title('Mask')

ax[2].plot(hist_mask)
ax[2].set_title('Histogram of Gray Image based on Mask')

plt.show()
```

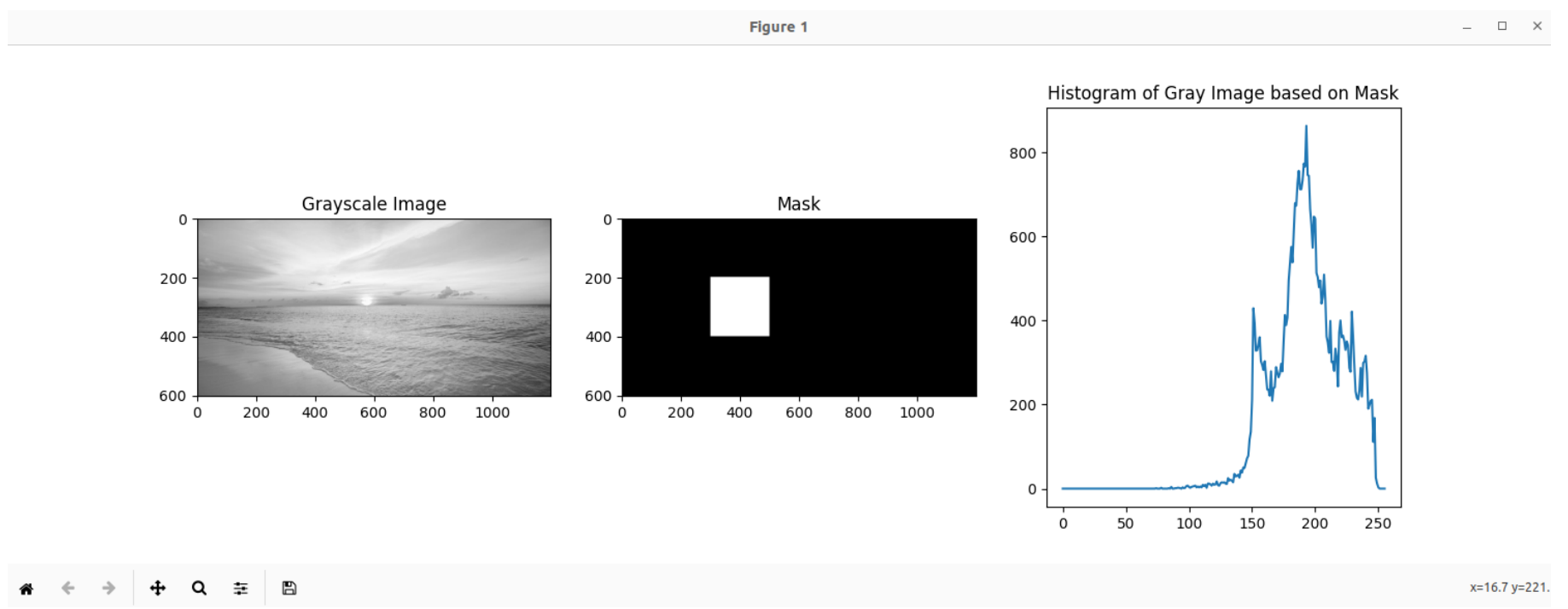


Figure 1
Grayscale Image
Mask
Histogram of Gray Image based on Mask