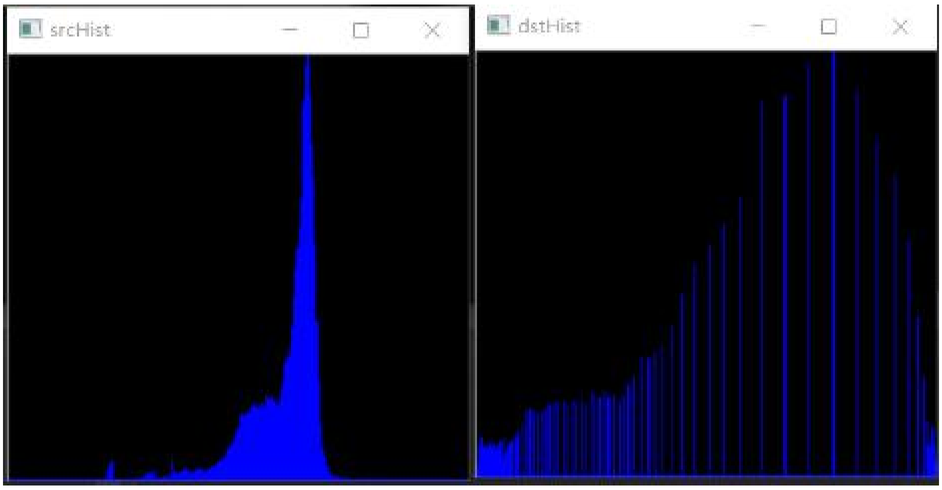
1.4.2 Histogram equalization

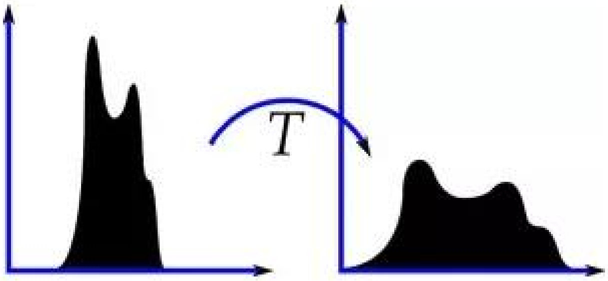
The spatial processing of images is mainly divided into two categories: grayscale transformation and spatial filtering. Histogram equalization is a commonly used grayscale transformation method. Generally, the components of the dark image histogram are concentrated at the lower end of the gray level, the components of the bright image histogram are biased toward the higher end of the gray level.



【left is the histogram, right is the histogram equalization effect】

Histogram equalization is to stretch the original histogram, which can make it is evenly distributed in the entire gray range, enhancing the contrast of the image.

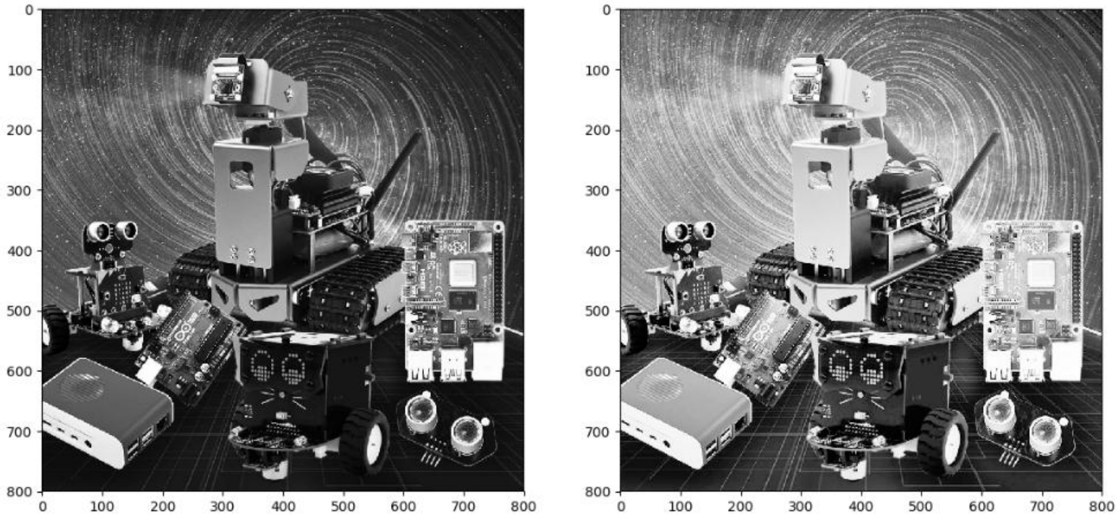
Function: **cv2.equalizeHist()**



```
# Grayscale histogram equalization

import cv2

import numpy as np

import matplotlib.pyplot as plt
```

```
img = cv2.imread('yahboom.jpg',1)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#cv2.imshow('src',gray)
dst = cv2.equalizeHist(gray)
#cv2.imshow('dst',dst)
#cv2.waitKey(0)


img = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
# plt draw two comparison pictures before and after processing
# original picture
plt.figure(figsize=(14, 9), dpi=100)# Set the size and pixels of the drawing area
plt.subplot(121)    # The first in a row and two columns
plt.imshow(img)


# Grayscale histogram equalization
plt.subplot(122)    # The second in a row and two columns
plt.imshow(dst)
plt.show()
```

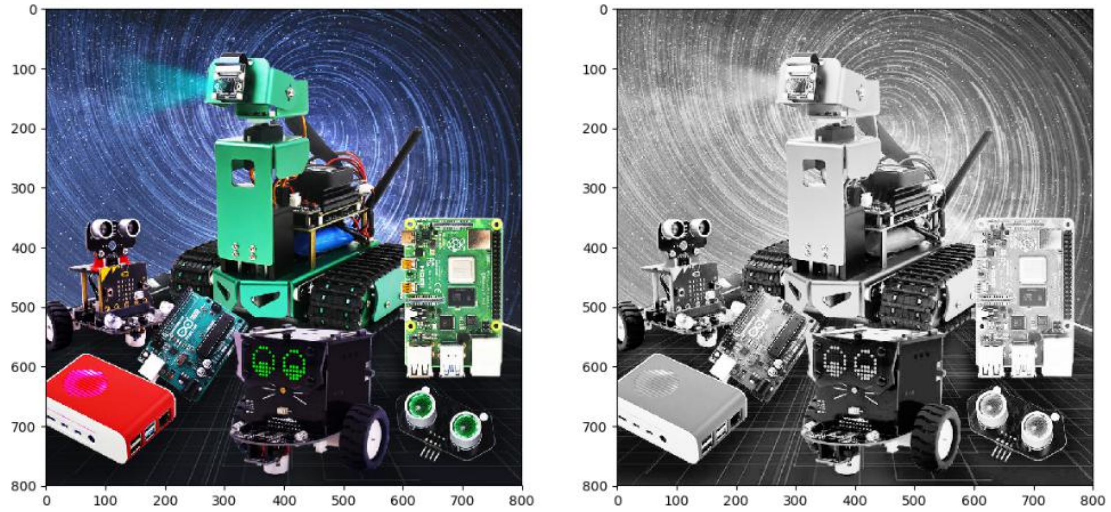
After running the following program, two pictures will be displayed in the jupyterLab control interface, as shown below.



```
# Grayscale histogram equalization
import cv2
import numpy as np
img = cv2.imread('yahboom.jpg',1)
# cv2.imshow('src',img)
(b,g,r) = cv2.split(img) # Channel decomposition
bH = cv2.equalizeHist(b)
gH = cv2.equalizeHist(g)
rH = cv2.equalizeHist(r)
result = cv2.merge((bH,gH,rH))# Channel synthesis
# cv2.imshow('dst',result)
# cv2.waitKey(0)


img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
```

```
# plt draw two comparison pictures before and after processing

plt.figure(figsize=(14, 9), dpi=100) # Set the size and pixels of the drawing area

plt.subplot(121)    # The first in a row and two columns

plt.imshow(img)

plt.subplot(122)    # The second in a row and two columns

# Color histogram equalization

plt.imshow(dst)

plt.show()
```

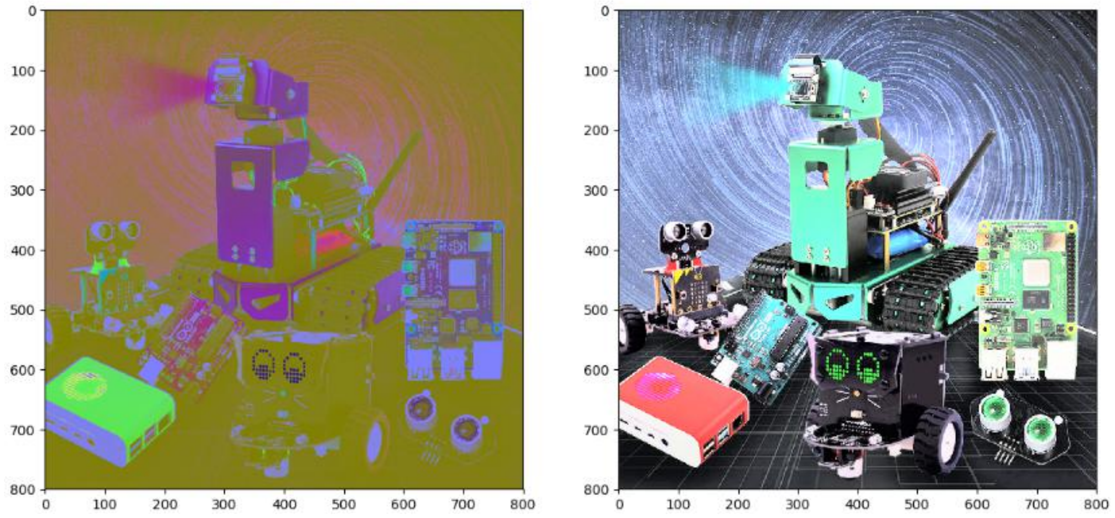
After running the following program, two pictures will be displayed in the jupyterLab control interface, as shown below.



```
# YUV histogram equalization

import cv2

import numpy as np

img = cv2.imread('yahboom.jpg',1)

imgYUV = cv2.cvtColor(img,cv2.COLOR_BGR2YCrCb)
```

```
# cv2.imshow('src',img)
channelYUV = cv2.split(imgYUV)
channelYUV[0] = cv2.equalizeHist(channelYUV[0])
channels = cv2.merge(channelYUV)
result = cv2.cvtColor(channels,cv2.COLOR_YCrCb2BGR)
# cv2.imshow('dst',result)
# cv2.waitKey(0)

imgYUV = cv2.cvtColor(imgYUV, cv2.COLOR_BGR2RGB)
result = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
#plt draw two comparison pictures before and after processing
plt.figure(figsize=(14, 9), dpi=100)# Set the size and pixels of the drawing area
plt.subplot(121)    # The first in a row and two columns
plt.imshow(imgYUV)
plt.subplot(122)    # The second in a row and two columns
#Color histogram equalization
plt.imshow(result)
plt.show()
```

After running the following program, two pictures will be displayed in the jupyterLab control interface, as shown below.


0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800