1.3.1 Grayscale processing

The process of converting a color picture into a grayscale picture is the grayscale processing of the picture. Each pixel of color picture in a color image is determined by the three components R, G, and B, each component range 0-255, so that a pixel can have more than 16 million (256*256*256 = 1677256) colors.

The grayscale image is a special color image with the same three components of R, G, and B, the range of one pixel is 256 kinds.
Therefore, in digital image processing, we will convert images of various formats into grayscale images to reduce the amount of calculation.
The grayscale image can still reflect the overall and partial distribution of chroma and highlight levels of the picture.

The description of the gray-scale image, like the color image, still reflects the distribution and characteristics of the overall and partial chroma and highlight levels of the entire image.

Image graying

Grayscale processing is the process of converting a color image into a grayscale image. The color image is divided into three components, R, G, and B, and displays various colors such as red, green, and blue. Pixels with larger gray values are brighter (maximum pixel value is 255, white), and pixels with smaller gray values are darker (minimum pixel value is 0, black).

The core idea of image graying is R = G = B, this value is also called gray value

Image graying algorithm

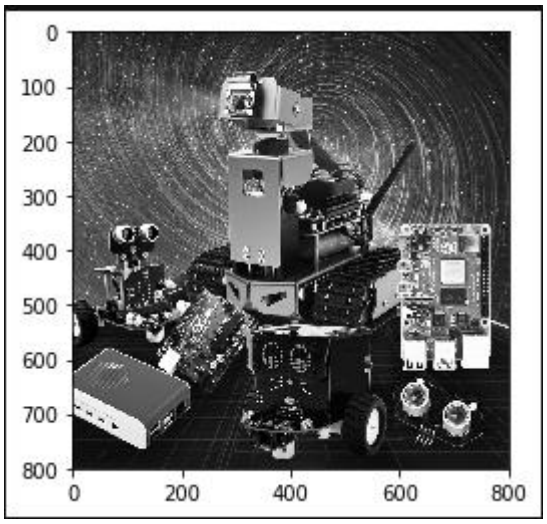
1) Maximum value method: make the value of R, G, B after conversion equal to the largest one of the three values before conversion, that is: R=G=B=max (R, G, B). The brightness of grayscale image converted by this method is very high.
2) Average method: R, G, and B values after conversion are the average values of R, G, and B before conversion. That is: R=G=B=(R + G + B)/3. The grayscale image produced by this method is relatively soft.
3) Weighted average method: according to a certain weight, the weighted average of the values of R, G, and B, that is, $R=G=B=(\omega_R R+\omega_G G+\omega_B B)/3$ is the weights of R, G, and B. We can take different values to form different gray images. Since the human eye is most sensitive to green, red, and has the lowest sensitivity to blue, we can get a more easily recognizable grayscale image. This method produces the best grayscale images.

The following code contains four methods to grayscale the picture,
Code path:

/home/pi/Yahboom_Project/1.OpenCV_course/03IP_Draw_text_line_segments/06_Picture_Rotation.ipynb

```
# Method 1 imread
# Note: Sometimes the picture will display after running the program for the first time, it will display after running the program for the second time

import cv2
import matplotlib.pyplot as plt

img0 = cv2.imread('yahboom.jpg',0)
img1 = cv2.imread('yahboom.jpg',1)
# print(img0.shape)
# print(img1.shape)
# cv2.imshow('src',img0)
# cv2.waitKey(0)

# Original picture
# img_bgr2rgb1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
# plt.imshow(img_bgr2rgb1)

# Grayscale picture
img_bgr2rgb0 = cv2.cvtColor(img0, cv2.COLOR_BGR2RGB)
        plt.imshow(img_bgr2rgb0)
        plt.show()
```

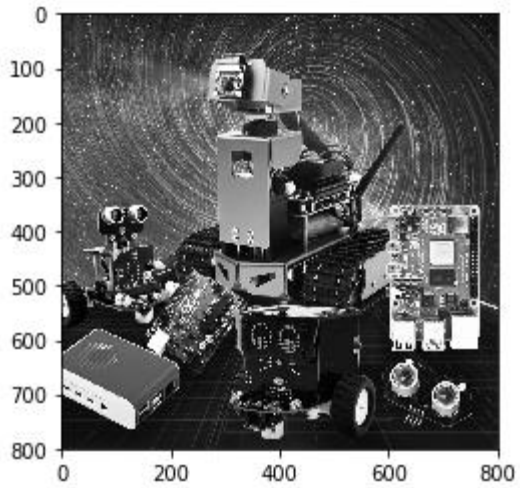
After running the following program, grayscale picture will be displayed in the jupyterLab control interface, as shown below.




```
# Method 2 cvtColor
# Note: Sometimes the picture will display after running the program for the first
```

```
time, it will display after running the program for the second time
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('image0.jpg',1)
dst = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)# Color space conversion 1 data 2 BGR
gray
#cv2.imshow('dst',dst)
#cv2.waitKey(0)

# Original picture
# img_bgr2rgb1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# plt.imshow(img_bgr2rgb1)

# Grayscale picture
img_bgr2rgb0 = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
      plt.imshow(img_bgr2rgb0)
      plt.show()
```

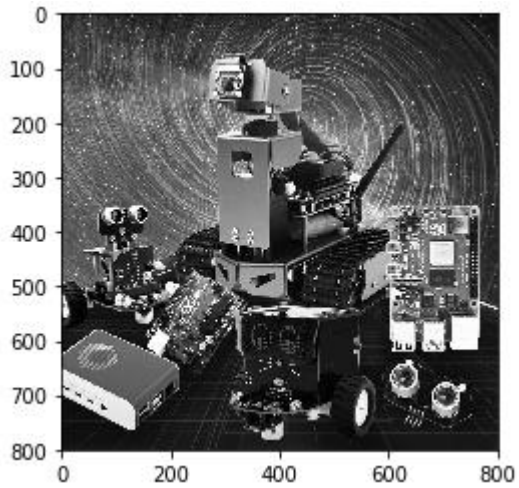
After running the following program, grayscale picture will be displayed in the jupyterLab control interface, as shown below.




```
# Method 3 Average
import cv2
import numpy as np
import matplotlib.pyplot as plt


img = cv2.imread('yahboom.jpg',1)
imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
```

```
# RGB R=G=B = gray    (R+G+B)/3
dst = np.zeros((height,width,3),np.uint8)
for i in range(0,height):
        for j in range(0,width):
                (b,g,r) = img[i,j]
                gray = (int(b)+int(g)+int(r))/3
                dst[i,j] = np.uint8(gray)
#cv2.imshow('dst',dst)
#cv2.waitKey(0)

# Original picture
# img_bgr2rgb1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# plt.imshow(img_bgr2rgb1)

# Grayscale picture
img_bgr2rgb0 = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
plt.imshow(img_bgr2rgb0)
        plt.show()
```

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




```
# Method 4 Weighted average
# gray = r*0.299+g*0.587+b*0.114
import cv2
import numpy as np
img = cv2.imread('yahboom.jpg',1)
imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
dst = np.zeros((height,width,3),np.uint8)
for i in range(0,height):
```

```
        for j in range(0,width):
                (b,g,r) = img[i,j]
                b = int(b)
                g = int(g)
                r = int(r)
                gray = r*0.299+g*0.587+b*0.114
                dst[i,j] = np.uint8(gray)
#cv2.imshow('dst',dst)
#cv2.waitKey(0)

# Original picture
# img_bgr2rgb1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# plt.imshow(img_bgr2rgb1)

# Grayscale picture
img_bgr2rgb0 = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
plt.imshow(img_bgr2rgb0)
        plt.show()
```

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