1.3.2 Image Binarization

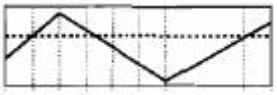
For binarization, we need to set a threshold. Those greater than the threshold are 0 (black) or 255 (white), the picture will be called black-white picture. The threshold can be fixed or it can be an adaptive threshold.

In generally, the adaptive threshold is the ratio of the average value of pixels in the area in this order or the weighted sum of Gaussian distribution and pixel at this point.

Global threshold

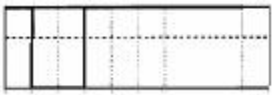
Python-OpenCV provide threshold function:

cv2.threshold(src, threshold, maxValue, method)

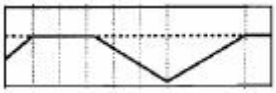


src Original picture: The broken line is the value to be thresholded; the dashed line is the threshold.

cv2.THRESH_BINARY: Set the gray value of the pixel point whose gray value is greater than the threshold to maxValue (for example, the maximum 8-bit gray value is 255), and set the gray value of the pixel point whose gray value is less than the threshold to 0.

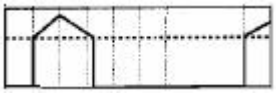


cv2.THRESH_BINARY_INV: Set the gray value of pixels greater than the threshold to 0, and set the gray value of pixels less than the threshold to maxValue.



cv2.THRESH_TRUNC: If the gray value of the pixel is less than the threshold, it will remain unchanged; if the pixel of the gray value of the pixel is greater than the gray value of the threshold, the pixel of the gray value will be set as the threshold.

cv2.THRESH_TOZERO: If the gray value of the pixel is less than the threshold, it will remain unchanged; if the gray value of the pixel is greater than the threshold, we need to set all gray values to 0.



cv2.THRESH_TOZERO_INV: If the gray value of the pixel is greater than the threshold, it will remain unchanged; if the gray value of the pixel is less than the threshold, we need to set its gray value to 0.

Path: /home/dofbot/Dofbot\4.opencv\4.image_beautification\02_Histogram_equalization.ipynb

Code as shown below.

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

img = cv2.imread('yahboom.jpg',1)
GrayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)    #Convert to grayscale image

ret,thresh1=cv2.threshold(GrayImage,10,255,cv2.THRESH_BINARY)
ret,thresh2=cv2.threshold(GrayImage,10,255,cv2.THRESH_BINARY_INV)
ret,thresh3=cv2.threshold(GrayImage,10,255,cv2.THRESH_TRUNC)
ret,thresh4=cv2.threshold(GrayImage,10,255,cv2.THRESH_TOZERO)
ret,thresh5=cv2.threshold(GrayImage,10,255,cv2.THRESH_TOZERO_INV)
titles = ['Gray Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [GrayImage, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

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