# 数字图像处理作业报告二

学号：71194506019 姓名：姜志刚 专业：计算机技术

## 题目

打开一副低对比度图像，拉伸其图像，直方图均衡。

## 推导

冈萨雷斯的书里给了几个公式，书中的式3.3-3一直不明白是如何得出的。

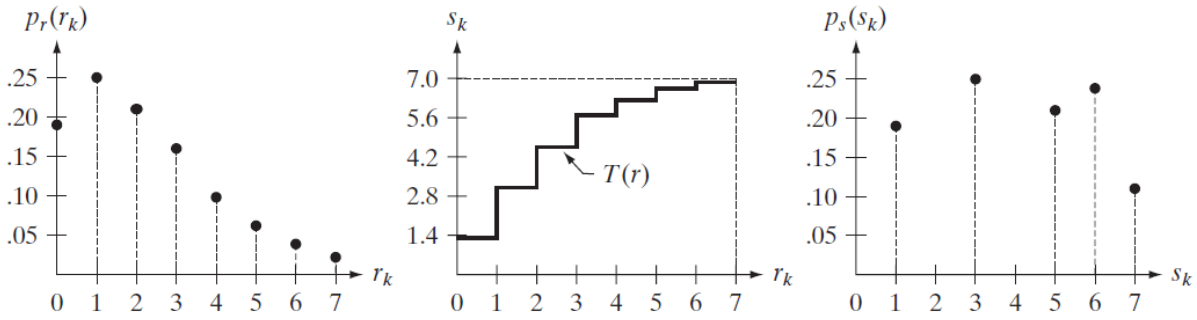
$$p_s(s)ds = p_r(r)dr$$

下面参考一些文章加上自己理解进行的推导：

$$s = T(r)$$

$r$为归一化后的颜色值，$s$为经过$T$变换的值，且$T(r)$在区间上为单调递增函数，即变换后的s，也是从黑到白，反函数：

$$r = T^{-1}(s)$$




a b c

**FIGURE 3.19** Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

设$F_S(s)$、$F_R(r)$分别为$S$和$R$的分布函数，则有：

$$F_S(s) = p\{S \leq s\} = p\{T(R) \leq s\} = p\{R \leq T^{-1}(s)\} = F_R(T^{-1}(s))$$

对$F_S(s) = F_R(T^{-1}(s)) = F_R(r)|_{r=T^{-1}(s)}$左右求导，$p_s(s)$、$p_r(r)$为概率密度函数：

$$p_s(s) = p_r(r)\frac{dr}{ds}|_{r=T^{-1}(s)}$$

$$\frac{ds}{dr} = \frac{p_r(r)}{p_s(s)}$$

即为所求。
均衡化后可知$p_s(s) = \frac{1}{L-1}$，对$\frac{ds}{dr} = \frac{p_r(r)}{p_s(s)}$进行积分，则：

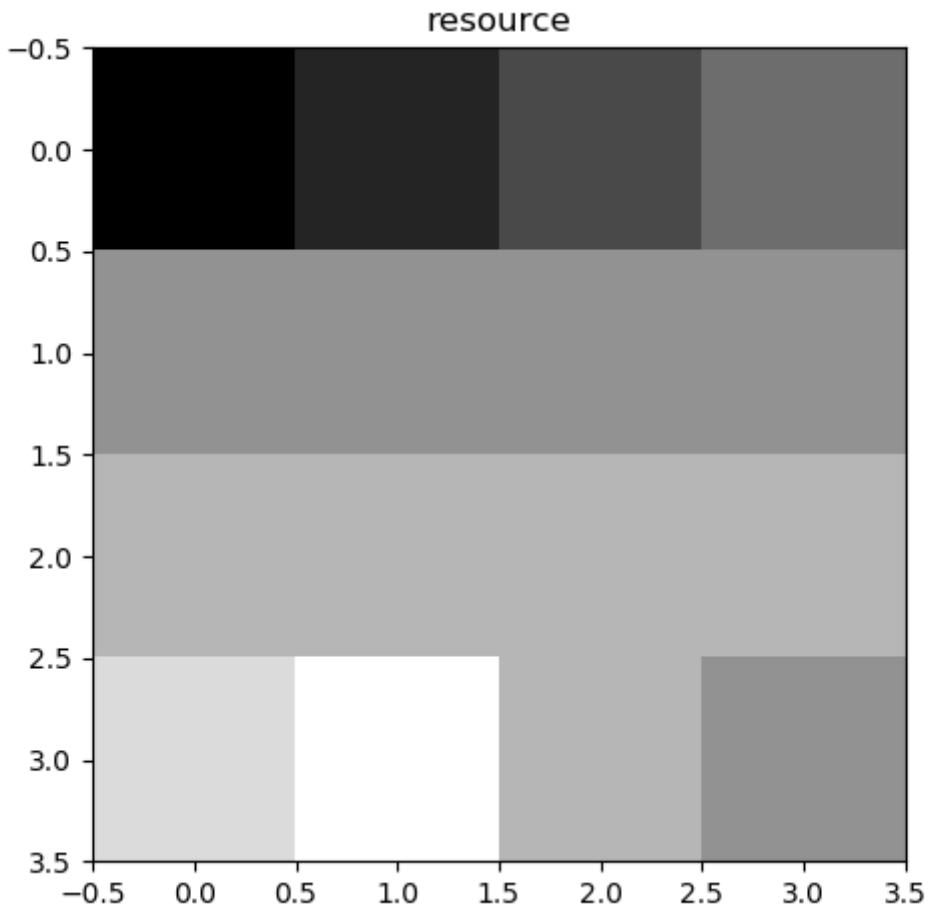
$$s = (L-1)\int_0^r p_r(w)dw = T(r)$$

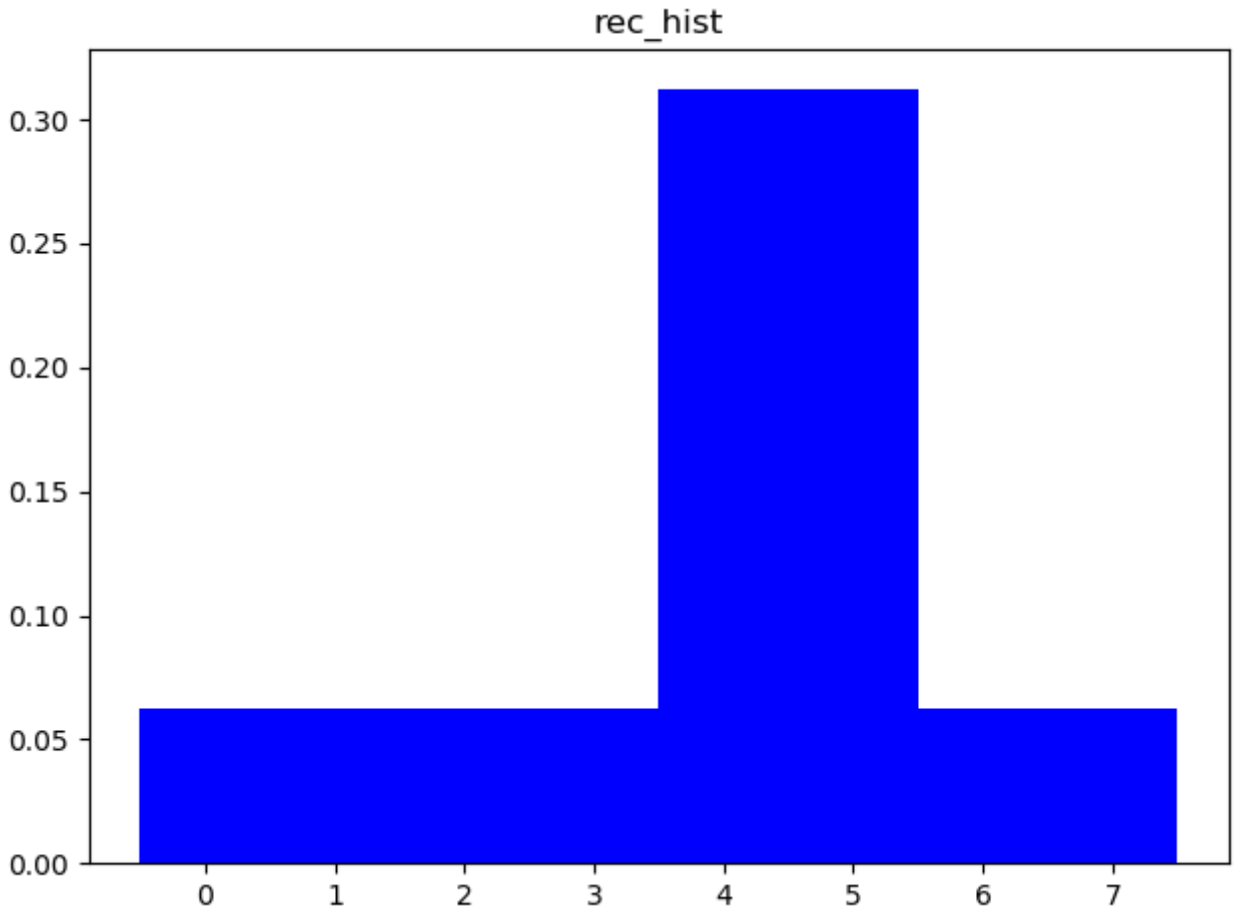
冈萨雷斯是先给出此公式，说这是**图像处理中特别重要的变换函数**，然后求得用这公式可以得到$p_s(s) = \frac{1}{L-1}$。 可是我不知道怎么来的，所以这里我先用结果$p_s(s) = \frac{1}{L-1}$，带入求得此公式。
此公式也就解释了为什么累计概率$(L-1)\int_0^r p_r(w)dw$就是$T(r)$，即$s$。

# 直方图均衡化

# 输入

假设有一幅4*4 大小8灰度级的图：



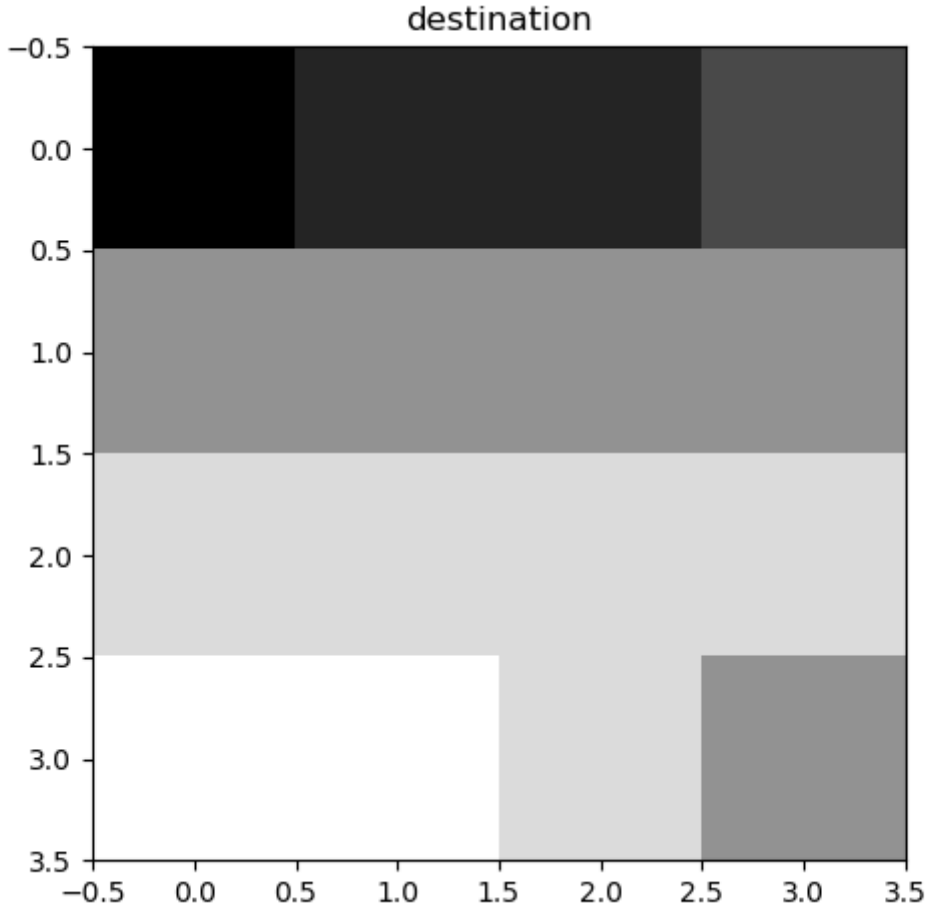
| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 |
| 6 | 7 | 5 | 6 |

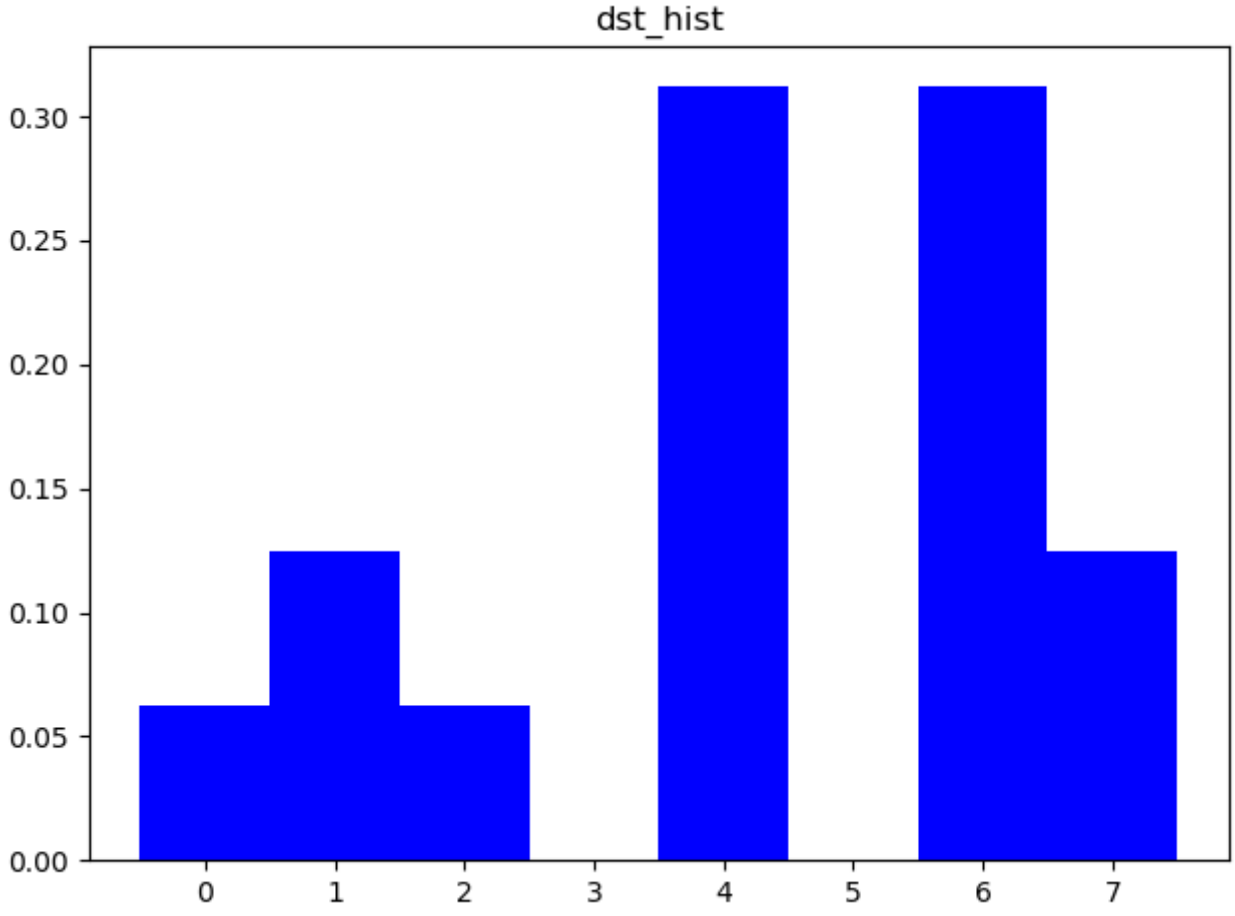
## 运算过程

*取整: $s_{k1} = int(s_k + 0.5)$

| $r_k$ | $n$ | $p_r(r_k)$ | $\sum p_r(r)$ | $s_{k1}$ | $p_s(r_k)$ |
|---|---|---|---|---|---|
| 0 | 1 | 1/16 | 1/16 = 0.06 | 0 | 1/16 |
| 1 | 1 | 1/16 | 2/16 = 0.12 | 1 | 2/16 |
| 2 | 1 | 1/16 | 3/16 = 0.19 | 1 | 1/16 |
| 3 | 1 | 1/16 | 4/16 = 0.25 | 2 | 0 |
| 4 | 5 | 5/16 | 9/16 = 0.56 | 4 | 5/16 |
| 5 | 5 | 5/16 | 14/16 = 0.88 | 6 | 0 |
| 6 | 1 | 1/16 | 15/16 = 0.93 | 7 | 5/16 |
| 7 | 1 | 1/16 | 1 | 7 | 2/16 |

## 结果


destination
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


dst_hist
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
6
7

# 代码实现

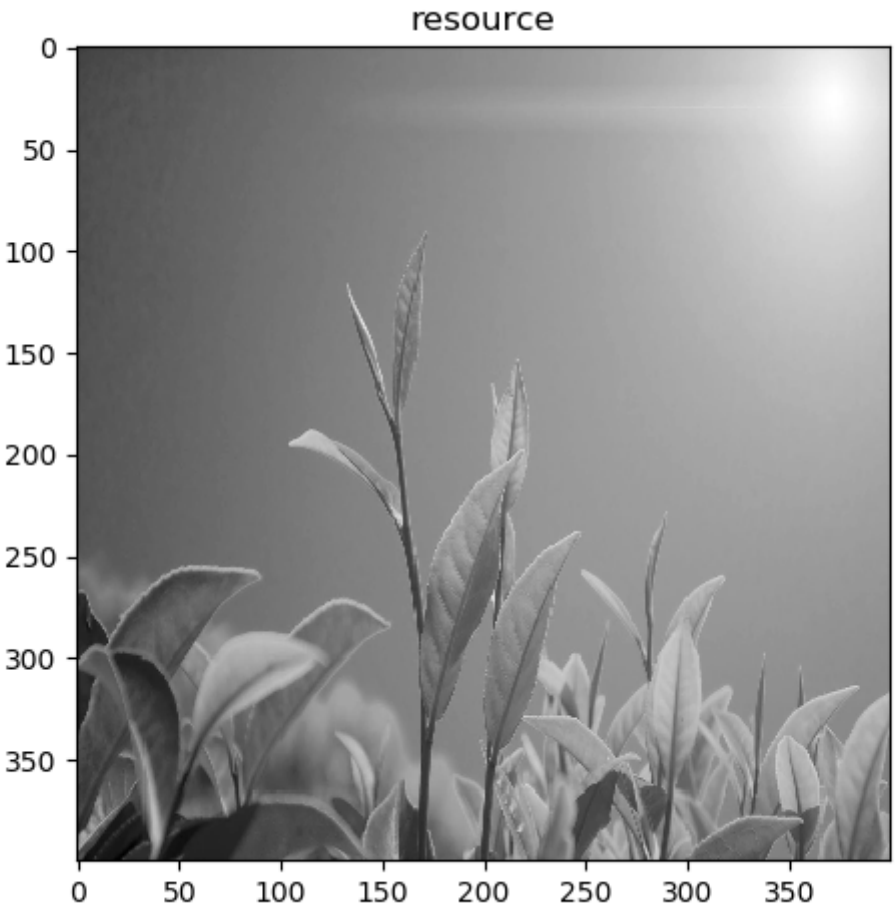
用python实现，关键代码：

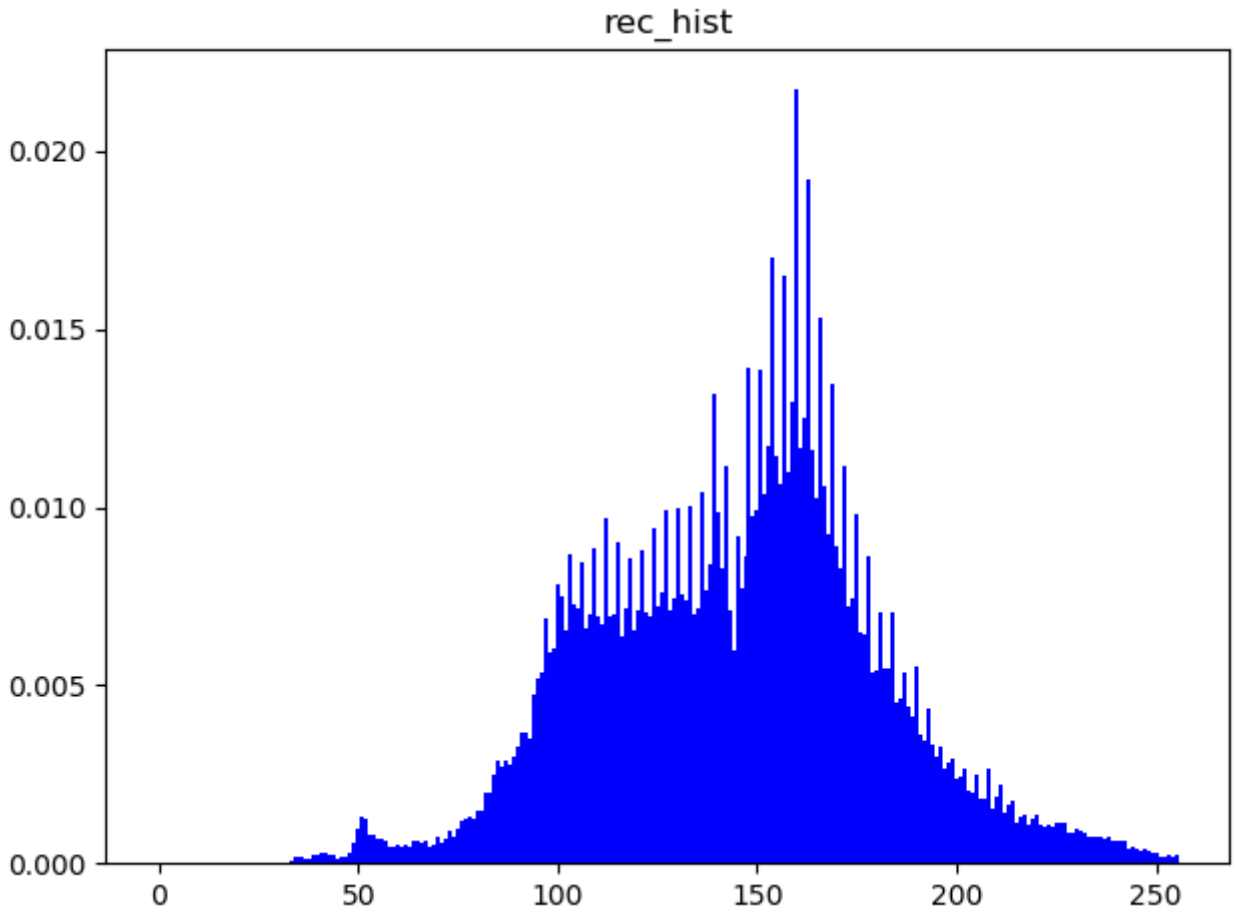
```
def equalize_hist(img, level):
    img = cv2.copyTo(img, None)
    height = img.shape[0]
    width = img.shape[1]

    color_gray = np.zeros(level, np.float)
    # n
    for i in range(height):
        for j in range(width):
            color_gray[img[i, j]] += 1
    # pr
    are = height * width
    for i in range(level):
        color_gray[i] /= are
    # sum(pr)
    for i in range(1, level):
        color_gray[i] += color_gray[i - 1]
    # sk
    for i in range(level):
        color_gray[i] = color_gray[i] * (level - 1) + 0.5
    # r -> s
    for i in range(height):
        for j in range(width):
            img[i, j] = color_gray[img[i, j]]
    return img
```

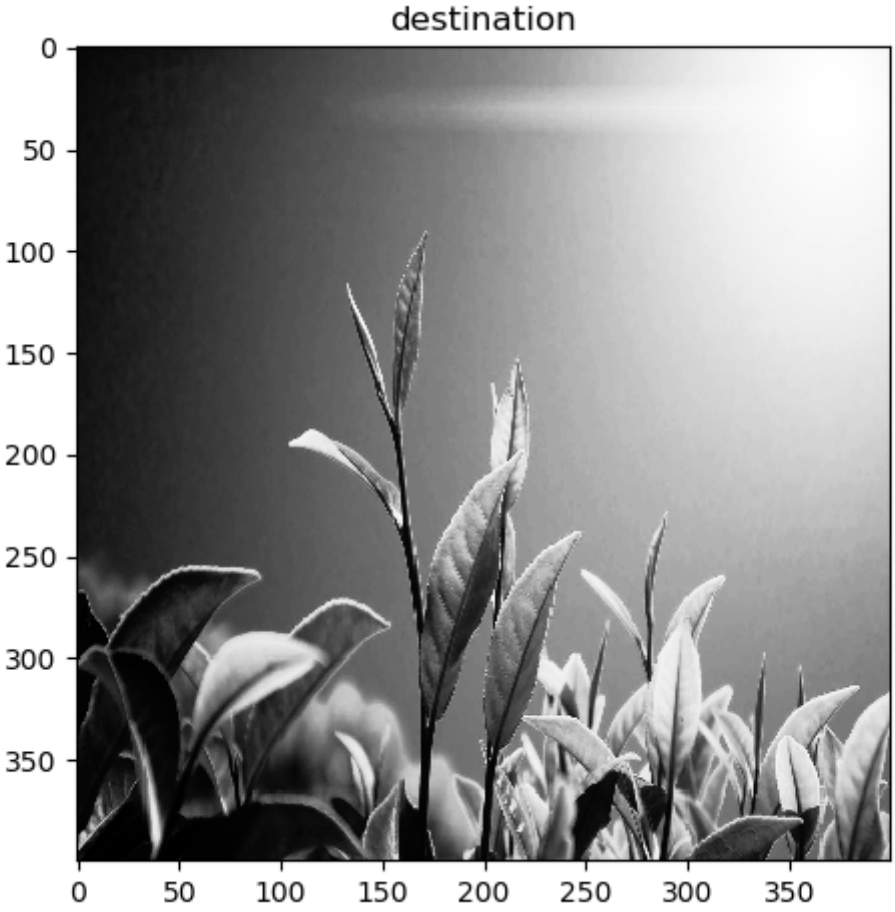
# 结果

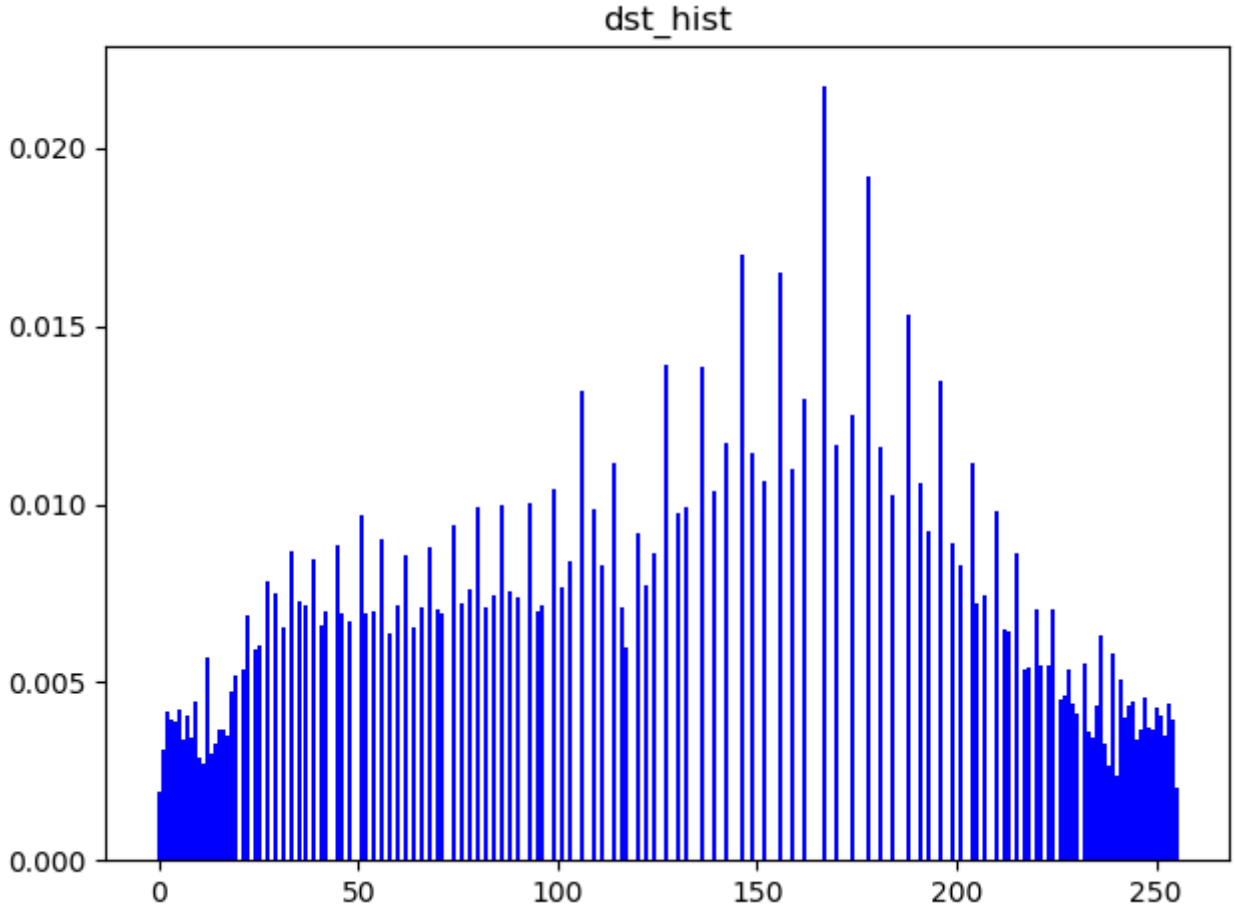
## 原图


resource
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

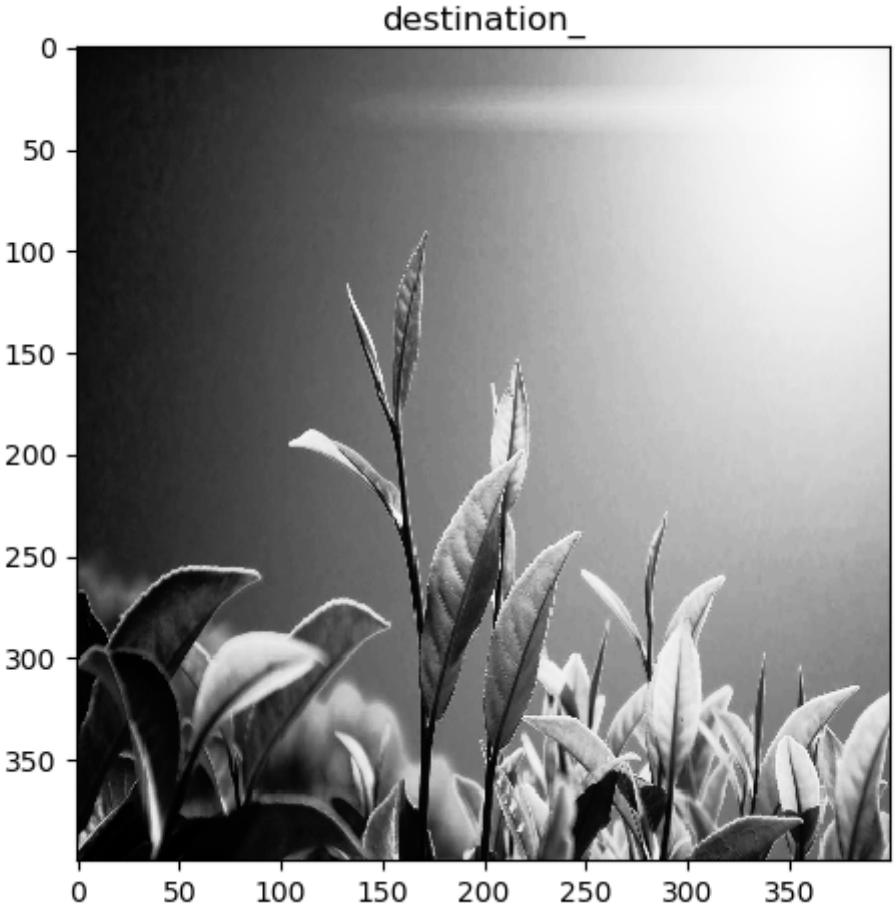

rec_hist
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250

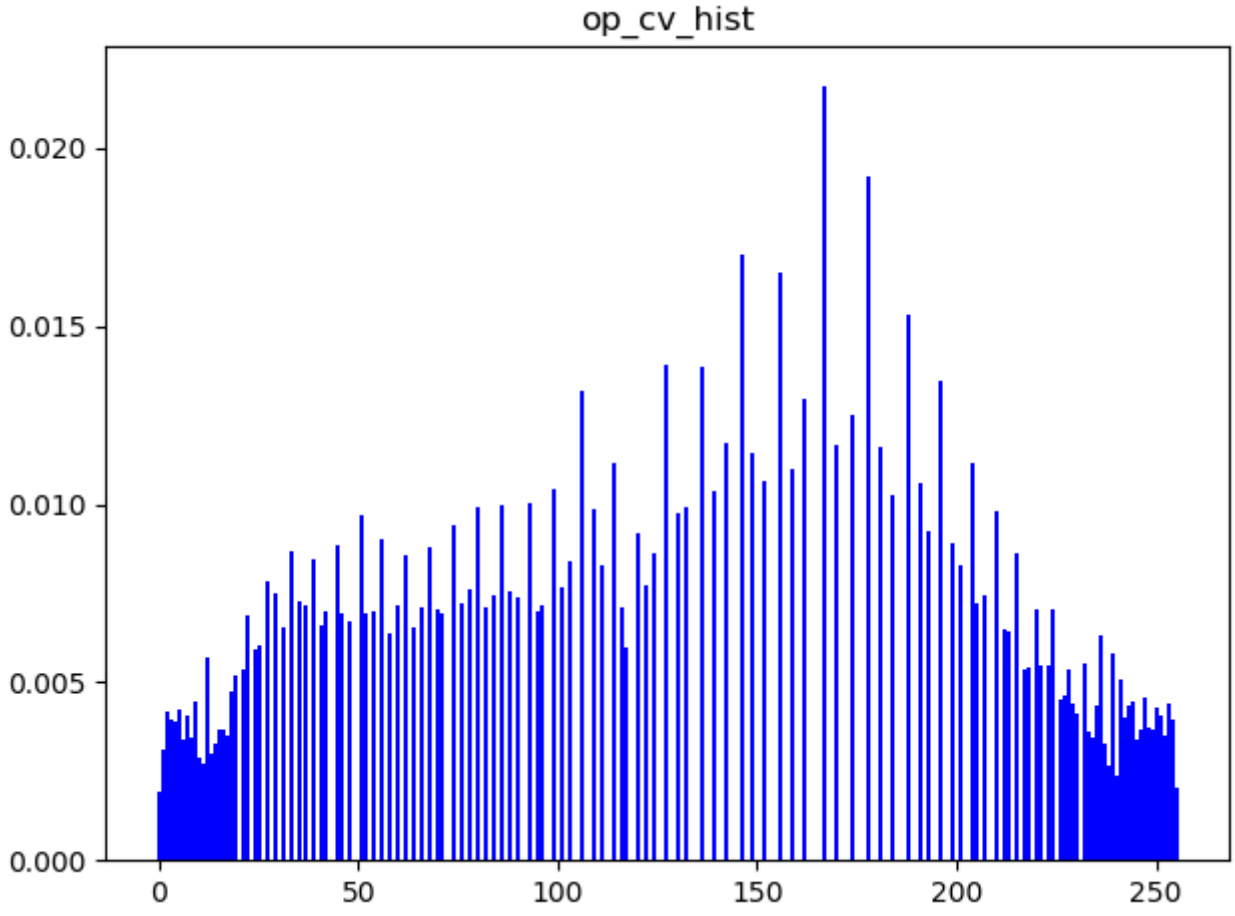
**实现均衡化**




**opencv实现的效果**


destination_
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


op_cv_hist
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250