

# 直方图均衡处理实验报告

蒋颜丞，自动化(电气)1903，3190102563

## 1 实验内容与要求

自选一张灰度图像，编程实现直方图均衡处理。

要求：

- 1.基于累积分布函数实现直方图均衡部分需要自写代码；
- 2.以实验报告形式提交结果，报告内容应包括自写源码、直方图均衡处理前后的图像和直方图。

本次实验将以下图为例，进行直方图均衡处理



## 2 实验原理

### 2.1 概念

**直方图均衡化 (Histogram Equalization)** 是指把一个已知灰度概率密度分布的图像经过一种变换，使之演变为一幅具有**均匀**灰度概率密度分布的新图像，是以累积分布函数变换法为基础的直方图修正法。

### 2.2 算法步骤

(1) **计算累计直方图**:  $p_r(r_k) = \frac{n_k}{n}, 0 \leq r_k \leq 1, k = 0, 1, \dots, l-1$ , 式中,  $l$  是灰度级的总数目,  $p_r(r_k)$  是取第  $k$  级灰度值的概率,  $n_k$  是图像中出现第  $k$  级灰度的次数,  $n$  是图像中像素总数;

(2) **构造变换函数**:  $s_k = T(r_k) = \sum_{j=0}^k \frac{n_j}{n} = \sum_{j=0}^k p_r(r_j), 0 \leq r_j \leq 1, k = 0, 1, \dots, l-1$

- (3) 对累计直方图进行灰度转换:  $s_k = T(r_k), k = 0, 1, \dots, l - 1$ ;
- (4) 对转换结果取最近的灰度级;
- (5) 输出新图像。

## 2.3 伪代码

### Image Histogram Equalization Algorithm

---

**Input** : *image*

**Result** : *new\_image*

*range*  $\leftarrow$  255

*number*  $\leftarrow$  *rows*(*image*) \* *cols*(*image*)

**for** *bright* = 0 to 255 :

*pixels.at.level*<sub>*bright*</sub>  $\leftarrow$  0

**for** *x* = 0 to *rows*(*image*) - 1 :

**for** *y* = 0 to *cols*(*image*) - 1 :

*pixels.at.level*<sub>*image*<sub>*x,y*</sub></sub>  $\leftarrow$  *pixels.at.level*<sub>*image*<sub>*x,y*</sub></sub> + 1

*sum*  $\leftarrow$  0

**for** *level* = 0 to 255 :

*sum*  $\leftarrow$  *sum* + *pixels.at.level*<sub>*level*</sub>

*hist*<sub>*level*</sub>  $\leftarrow$  uint8  $\left[ \text{round} \left( \frac{\text{range}}{\text{number}} * \text{sum} \right) \right]$

**for** *x* = 0 to *rows*(*image*) - 1 :

**for** *y* = 0 to *cols*(*image*) - 1 :

*new\_image*<sub>*x,y*</sub>  $\leftarrow$  *hist*<sub>*image*<sub>*x,y*</sub></sub>

**return** *new\_image*

## 3 源代码

```

1  import cv2
2  from PIL import Image, ImageDraw, ImageFont
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  def Img_Hist(img):
7      '''
8          计算图像的直方图

```

```

9      输入：灰度图像
10     输出：以一维向量保存的直方图数据
11     '''
12     hist=np.zeros(256,dtype=int) # 初始化直方图数据
13
14     # 统计直方图数据
15     for i in range(img.shape[0]):
16         for j in range(img.shape[1]):
17             k = img[i][j]
18             hist[k] += 1
19
20     return hist
21
22 def Equal_Hist(hist, img):
23     '''
24     直方图均衡处理
25     输入：原始图像直方图数据，原始图像
26     输出：处理后图像
27     '''
28     Tr = Cal_Tr(hist, img) # 计算Tr
29
30     new_img = np.zeros(shape=(img.shape[0],img.shape[1]),dtype=np.uint8)
31
32     # 对原图像进行映射
33     for k in range(img.shape[0]):
34         for l in range(img.shape[1]):
35             new_img[k][l] = Tr[img[k][l]]
36
37     return new_img
38
39 def Cal_Tr(hist, img):
40     '''
41     计算Tr
42     输入：原始图像直方图数据，原始图像
43     输出：Tr一维数组
44     '''
45     Pr=np.zeros(256,dtype=float) # 初始化频率分布
46     N = img.shape[0]*img.shape[1]
47     for i in range(256):
48         Pr[i]=hist[i]/N # 统计频率
49
50     # 计算Tr
51     Tr=np.zeros(256,dtype=float) # 初始化Tr
52     temp = 0
53     for m in range(256):
54         temp += 255*Pr[m]
55         Tr[m] = np.uint8(round(temp))
56
57     return Tr

```

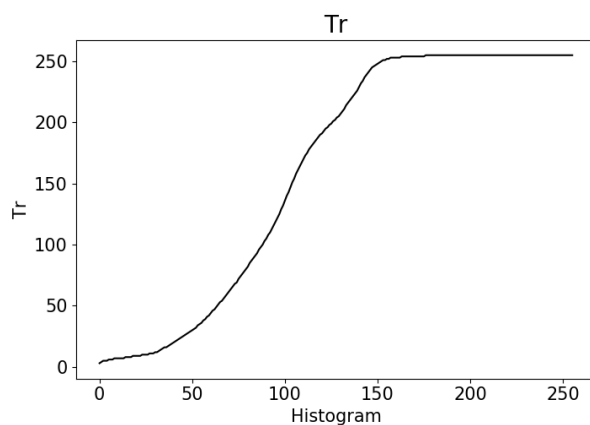
```

58
59 if __name__ == '__main__':
60
61     # 读取原始图像
62     img = cv2.imread('GrayPhoto.jpg', cv2.IMREAD_GRAYSCALE)
63
64     #计算原图的直方图
65     origin_hist = Img_Hist(img)
66
67     # 直方图均衡化
68     new_img = Equal_Hist(origin_hist,img)
69
70     # 计算处理后图像的直方图
71     new_hist = Img_Hist(new_img)
72
73     cv2.imshow('origin_image',img)
74     cv2.imshow('new_image',new_img)
75     cv2.imwrite('new_image.jpg', new_img)
76
77     # 绘制灰度直方图
78     plt.figure(1)
79     plt.subplot( 1, 2, 1 )
80     plt.plot(np.arange(256), origin_hist, 'r', linewidth=1.5, c='black')
81     plt.tick_params(labelsize=15)
82     plt.title("Origin",fontdict={'weight':'normal','size': 20})
83     plt.xlabel("Histogram",fontdict={'weight':'normal','size': 15})
84     plt.ylabel("Number of Pixels",fontdict={'weight':'normal','size': 15})
85
86     plt.subplot( 1, 2, 2 )
87     plt.plot(np.arange(256), new_hist, 'r', linewidth=1.5, c='black')
88     plt.tick_params(labelsize=15)
89     plt.title("New",fontdict={'weight':'normal','size': 20})
90     plt.xlabel("Histogram",fontdict={'weight':'normal','size': 15})
91     plt.ylabel("Number of Pixels",fontdict={'weight':'normal','size': 15})
92
93     plt.show()
94
95     print('Equalization has done.')
96
97     K = cv2.waitKey(0)

```

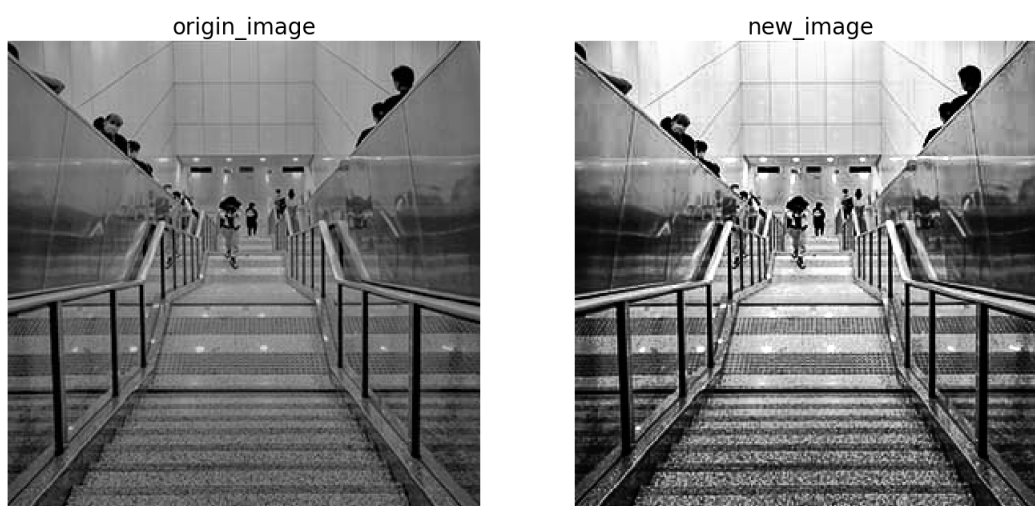
## 4 实验结果与分析

## 4.1 变换函数



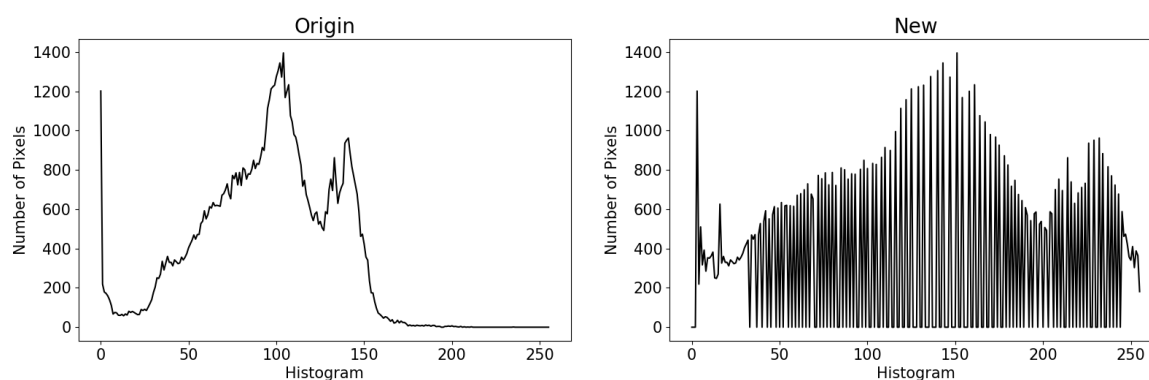
从变换函数  $s_k = T(r_k)$  来看，其符合变换函数的基本要求（递增性和有界性），同时，在灰度值较高的区域（165~255），变换函数对灰度值的提升较为显著，这也与后文所展示的直方图的变化相一致。

## 4.2 图像



从图像中可以看出，原来的图像画面比较平淡，整体的灰度较为一致；而直方图均衡化处理后的图像对比度更高，画面更有层次感，在视觉上更清晰。

## 4.3 直方图



从直方图中可以看出，原来的图像灰度值主要集中在中段（75~155），且灰度直方图曲线连续；而直方图均衡化处理后的图像：灰度值更为均匀，尤其是在灰度值较高的区域（165~255），其像素数量明显增多,但灰度直方图曲线变得离散，图像损失了一些细节。

## 5 结论

直方图均衡处理可以让图像的灰度级分布更为均匀，使图像具有高对比度和多变的灰色色调，观感更好，但同时可能会损失一些细节。