**北京邮电大学软件学院**

**2022-2023学年第二学期实验报告**

**课程名称：** 多媒体技术与应用

**项目名称： 实验三 直方图规定化**

**项目完成人：**

**姓名： 王宇涵 学号： 2020211730**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

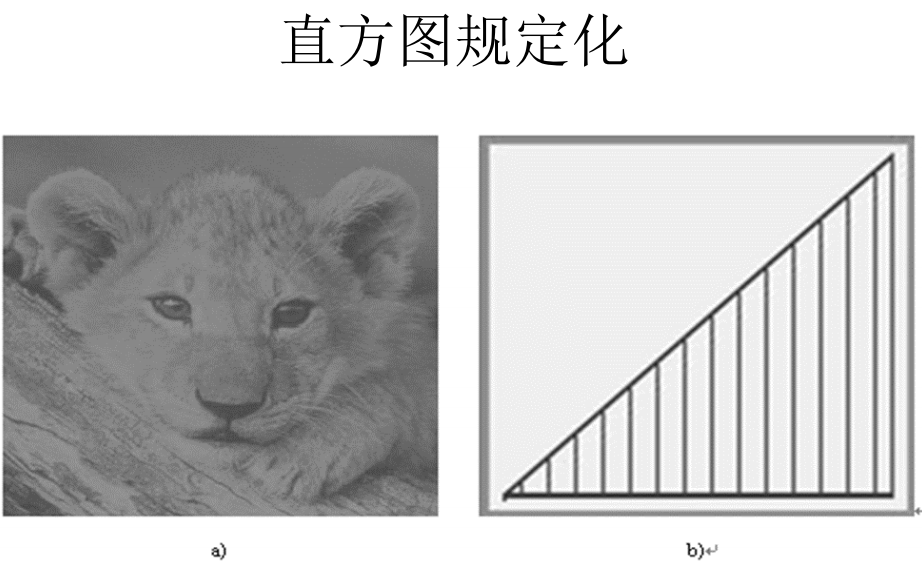
**指导教师：**\_\_\_\_\_\_\_\_\_\_\_李朝晖\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2023年 4 月 20 日**

1. **实验目的**

掌握图像直方图规定化

1. **实验内容**
2. **直方图规定化**



1. **实验环境**

Windows10操作系统

Pycharm(编译器)

Python3.8

1. **实验结果**

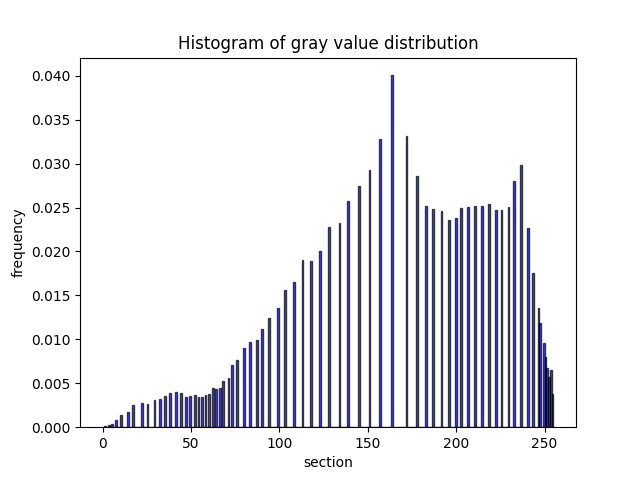
## 直方图规定化

转换前:



转换后:





1. **附录**

## 直方图规定化及分析

1. 问题分析

直方图规定化可以使图像按照某一特定的直方图形状进行均衡化，能够有目的的增强某个灰度区间。进行直方图规定化，首先要对原始图像的灰度直方图进行均衡化，再对规定的直方图进行均衡化，把均衡化作为中间结果，得到规定化后像素之间的映射关系。

1. 设计方案

首先读取图像，计算其灰度直方图，然后生成题目中所给的三角形直方图，再比较每个灰度级，得到两个累积直方图距离最近的点，得到映射关系，进行映射。输出图像。

具体步骤：获取原图和规定化直方图的均衡化结果→将原图的均衡化结果的取值集按升序排序，得到可匹配值数组→将可匹配数组的每一个值就近找到规定化直方图均衡化后的值的对应值→得到灰度对应数组，生成规定化图像

1. 算法

对原始图像进行规定化处理，即04，通过求差找到原始图像与标准图的累计灰度直方图的差的最小值，即找到两图距离最近的点，算法如下：

# 直方图规定化函数，传入的参数为源图像和目标图像的累计直方图（数组）  
def HistogramMatching(image, refTotal):  
 # 获取像素值  
 pixel\_map = image.load()  
 # 获取图像的宽和高  
 width, height = image.size  
 # 获取图像的模式  
 mode = image.mode  
 # 计算图像的像素总数  
 size = height \* width  
  
 # map 源图像的直方图  
 # 创建一个长度为 256 的数组，用于记录每个灰度级出现的次数  
 map = [0] \* 256  
 for x in range(width):  
 for y in range(height):  
 # 获取像素值  
 pixel = pixel\_map[x, y]  
 # 统计该灰度级出现的次数  
 map[pixel] = map[pixel] + 1  
  
 # totalmap 原始图像的累计直方图  
 # 创建一个长度为 256 的数组，用于记录每个灰度级在图像中出现的像素总数  
 mapTotal = [0] \* 256  
 for i in range(1, 256):  
 # 计算累计像素总数  
 mapTotal[i] = map[i] + mapTotal[i - 1]  
  
 # new数组用来对应源图像的灰度值（0-255）到处理后图像的灰度值的映射关系  
 # 创建一个长度为 256 的数组，用于记录每个源图像灰度级到目标图像灰度级的映射关系  
 new = [0]  
 index = -1  
  
 # 对于每一个灰度值（i），找到规定直方图中对应的最接近的灰度值（j）  
 for i in range(0, 256):  
 ans = 1000  
 for j in range(0, 256):  
 # 计算源图像的累计直方图和目标图像的累计直方图之间的差距  
 cur = abs(mapTotal[i] / size - refTotal[j] / size\_ref)  
 if (cur < ans):  
 ans = cur  
 index = j  
 # 将映射关系添加到 new 数组中  
 new.append(index)  
  
 # 创建新图像，一一对应  
 # 创建一个新图像，宽度和高度与源图像相同，模式为灰度图像  
 new\_image = Image.new(mode, (width, height))  
 new\_map = new\_image.load()  
 for x in range(width):  
 for y in range(height):  
 # 获取源图像的灰度值，根据 new 数组中的映射关系将其转换为目标图像的灰度值  
 new\_pixel = new[(pixel\_map[x, y])]  
 # 将目标图像的灰度值赋值给新图像  
 new\_map[x, y] = (new\_pixel)  
  
 return new\_image

首先获取源图像的像素值、宽度、高度和模式，然后计算源图像的直方图和累计直方图。接下来，根据目标图像的累计直方图和源图像的累计直方图，将源图像中每个灰度级映射到目标图像中的最接近灰度级，并将映射关系记录在数组 new 中。最后，将新的灰度级赋值给新的像素，得到处理后的图像。

**绘制直方图函数**

**同上次实验**

1. 程序

from typing import Match  
import matplotlib.pyplot as plt  
from PIL import Image, ImageOps  
import numpy as np  
import math  
  
# 直方图规定化函数，传入的参数为源图像和目标图像的累计直方图（数组）  
def HistogramMatching(image, refTotal):  
 # 获取像素值  
 pixel\_map = image.load()  
 # 获取图像的宽和高  
 width, height = image.size  
 # 获取图像的模式  
 mode = image.mode  
 # 计算图像的像素总数  
 size = height \* width  
  
 # map 源图像的直方图  
 # 创建一个长度为 256 的数组，用于记录每个灰度级出现的次数  
 map = [0] \* 256  
 for x in range(width):  
 for y in range(height):  
 # 获取像素值  
 pixel = pixel\_map[x, y]  
 # 统计该灰度级出现的次数  
 map[pixel] = map[pixel] + 1  
  
 # totalmap 原始图像的累计直方图  
 # 创建一个长度为 256 的数组，用于记录每个灰度级在图像中出现的像素总数  
 mapTotal = [0] \* 256  
 for i in range(1, 256):  
 # 计算累计像素总数  
 mapTotal[i] = map[i] + mapTotal[i - 1]  
  
 # new数组用来对应源图像的灰度值（0-255）到处理后图像的灰度值的映射关系  
 # 创建一个长度为 256 的数组，用于记录每个源图像灰度级到目标图像灰度级的映射关系  
 new = [0]  
 index = -1  
  
 # 对于每一个灰度值（i），找到规定直方图中对应的最接近的灰度值（j）  
 for i in range(0, 256):  
 ans = 1000  
 for j in range(0, 256):  
 # 计算源图像的累计直方图和目标图像的累计直方图之间的差距  
 cur = abs(mapTotal[i] / size - refTotal[j] / size\_ref)  
 if (cur < ans):  
 ans = cur  
 index = j  
 # 将映射关系添加到 new 数组中  
 new.append(index)  
  
 # 创建新图像，一一对应  
 # 创建一个新图像，宽度和高度与源图像相同，模式为灰度图像  
 new\_image = Image.new(mode, (width, height))  
 new\_map = new\_image.load()  
 for x in range(width):  
 for y in range(height):  
 # 获取源图像的灰度值，根据 new 数组中的映射关系将其转换为目标图像的灰度值  
 new\_pixel = new[(pixel\_map[x, y])]  
 # 将目标图像的灰度值赋值给新图像  
 new\_map[x, y] = (new\_pixel)  
  
 return new\_image  
  
  
size\_ref = 32640  
refTotal = [i for i in range(256)]  
for i in range(1, 256):  
 # ref 参考图像累计直方图  
 refTotal[i] += refTotal[i - 1]  
  
# 绘制灰度图的直方图  
def drawHistogram(image):  
 width, height = image.size  
 data = []  
 for x in range(width):  
 for y in range(height):  
 gray\_pixel\_map = image.load()  
 gray\_pixel = gray\_pixel\_map[x, y]  
 # print(type(gray\_pixel))  
 pixel = gray\_pixel  
 data.append(pixel)  
  
 plt.hist(data, bins=256, density=0.3, facecolor="blue", edgecolor="black", alpha=0.7, stacked=True)  
 # 显示横轴标签  
 plt.xlabel("section")  
 # 显示纵轴标签  
 plt.ylabel("frequency")  
 # 显示图标题  
 plt.title("Histogram of gray value distribution")  
 plt.show()  
image = Image.open(r'E:\大三\{选}多媒体技术与应用\2020实验\实验3\cat.JPG')  
  
# 方便起见,调用库函数转化为灰度图  
image = ImageOps.grayscale(image)  
drawHistogram(image)  
image.show()  
new\_image = HistogramMatching(image, refTotal)  
drawHistogram(new\_image)  
new\_image.show()  
# # 给出图像规定化后的累计直方图  
# new\_image = np.array(new\_image)  
# new\_image.show()

**实验心得**

直方图匹配是一种图像处理方法，可以通过调整图像的灰度分布来增强图像的对比度和清晰度。在进行直方图匹配时，我们需要先准备一个目标图像的累计直方图，然后根据这个累计直方图来对原始图像进行直方图均衡化，从而得到一个灰度分布与目标图像类似的新图像。

在进行直方图匹配的实验过程中，我们可以采用Python编程语言和PIL库来实现。具体来说，我们可以编写一个名为“HistogramMatching”的函数，该函数需要传入原始图像和目标图像的累计直方图作为参数，并返回一张经过直方图匹配处理后的新图像。

在编写“HistogramMatching”函数时，我们需要先获取原始图像的像素值、宽度、高度和模式，然后计算出原始图像中每个灰度级出现的次数，以及每个灰度级在原始图像中出现的像素总数。接着，我们需要根据目标图像的累计直方图来计算出每个源图像灰度级到目标图像灰度级的映射关系，并将这些映射关系存储到一个名为“new”的数组中。最后，我们可以创建一个新图像，并根据“new”数组中的映射关系来对原始图像进行像素值的映射，从而得到一张灰度分布与目标图像类似的新图像。

总的来说，直方图匹配是一种简单而有效的图像处理方法，可以帮助我们增强图像的对比度和清晰度。通过编写“HistogramMatching”函数，我们可以更加深入地理解直方图匹配的原理和实现方法，并将其应用到实际的图像处理任务中。