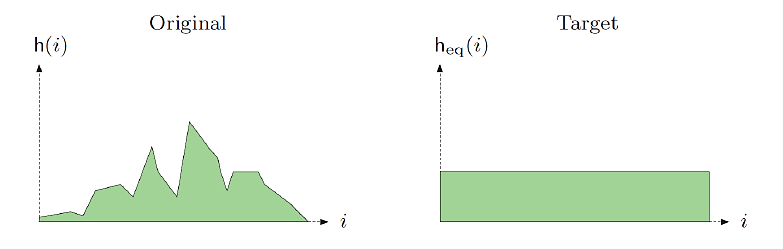
实验四 直方图调整

1. 知识点回顾

直方图反映了图像中每一灰度级与其出现的**频率/频数/概率**间的关系。本质上是灰度分布的概率密度函数，通过**调整直方图的形状**来**增强图像对比度**，是一种实用而有效的处理技术。

1. 直方图均衡化

基本思想是对原始图像的像素灰度做某种映射变换，使变换后图像灰度的概率密度近似均匀分布，即变换后图像的灰度级均匀分布。

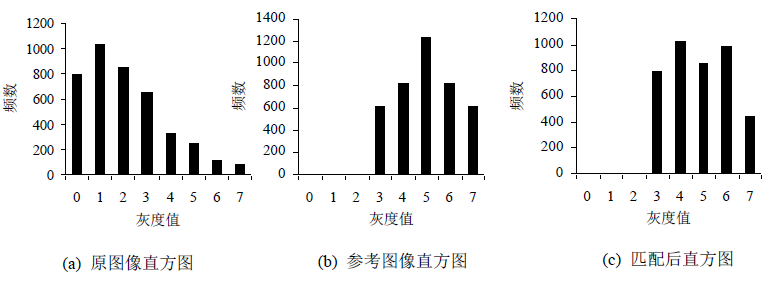


1. 理想情况下，直方图均衡化前后概率密度函数图

直方图均衡化的目的：保证每个灰度级的概率密度相等，也就是使灰度级的累积分布函数为线性形状；**提高图像细节的解译效果**。

1. 直方图规定化

通过将图像的直方图变成规定形状的直方图而对图像进行转换的增强方法，利于**图像融合、匹配及变化检测**等。



1. 原图像直方图、参考图像直方图及直方图规定化后直方图
2. 直方图显示

在Python中，显示直方图需要借助第三方库来完成，常用的第三方库有cv2（见实验02）、matplotlib和numpy。

若不借助第三方库，通过循环统计的方式也可以绘制出直方图。

* 1. 循环统计并绘制图像直方图

实现步骤包括3步：

1. 定义一个1\*256的数组存放每个灰度值出现的频次（对于8位图像）

代码：定义数组

array = np.arange(1,10,2)

#[1 3 5 7 9]

arange函数参数为：起点（可省略，默认0）、终点、步长（默认1，可谓小数）

1. 遍历图像，对每个灰度值进行统计
2. 以柱状图的方式绘制直方图：matplotlib.pyplot.bar()

代码：绘制柱状图

plt.bar(x, height, width=0.8, bottom=None, \*, align=‘center’, data=None)

plt.xlim(xmin, xmax)

plt.ylim(ymin, ymax)

bar函数参数为：x轴（数组）、柱高、柱宽（默认值为0.8）、柱高的下底部的高度（可在垂直方向叠加柱状图）、柱状图在x轴上的位置偏向（‘center’, ‘edge’）…

xlim函数参数为：x轴最小值、x轴最大值

ylim函数参数为：y轴最小值、y轴最大值

代码：统计并绘制图像直方图

def statHist(src):

rows = src.shape[0]

cols = src.shape[1]

# 定义1\*256矩阵，存储直方图统计量

hist = np.zeros(256, dtype = np.float32)

# 遍历像元值，统计每个DN值出现频次

for i in range(rows):

for j in range(cols):

dn = src[i, j]

hist[dn] += 1

# 绘制直方图，整饰

# 产生0.5的偏移量，使数值正好在条形的正中央

DN = np.arange(256) - 0.5

# width = 1 不可少，否则是条形图而不是直方图

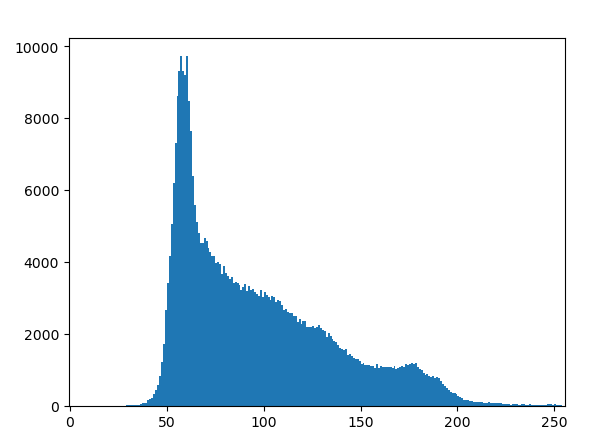
plt.bar(DN, hist, width = 1)

# 产生0.5的偏移量，使数值正好在条形的正中央

plt.xlim([0-0.5, 255.5])

plt.show()

return

1. 原图像(左)及其直方图(右)

时间较久，处理大小仅为640×640的影像就花费了几秒钟。因此，在了解其原理后，使用库中封装好的函数效率更高。

* 1. cv2.calcHist()方法

代码：

hist = cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

参数包括：

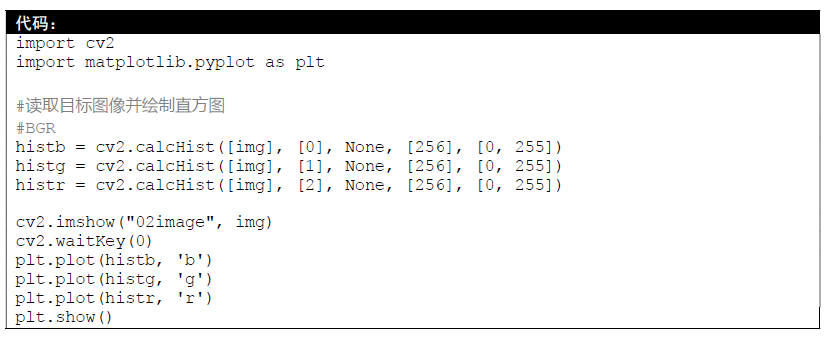
images 待统计直方图的图像

channels 图像的颜色通道，灰度图像为[0]，彩色图像的B、G、R通道分别对应[0]、[1]、[2]

mask 掩模，没有则为None

histSize 图像的量化级数，8bit为[256]

ranges 图像灰度的变化范围，8bit图像为[0, 255]



1. 实验02中绘制三通道图像直方图代码

对代码进行优化：

代码：

enume = enumerate(sequence, [start=0])

参数包括：

sequence 一个序列、迭代器或其他支持迭代对象

start 下标起始位置，默认为0

返回值包括：

enume 枚举对象

代码：统计并绘制三通道图像直方图

# 定义三个通道直方图的颜色

color = ['b', 'g', 'r']

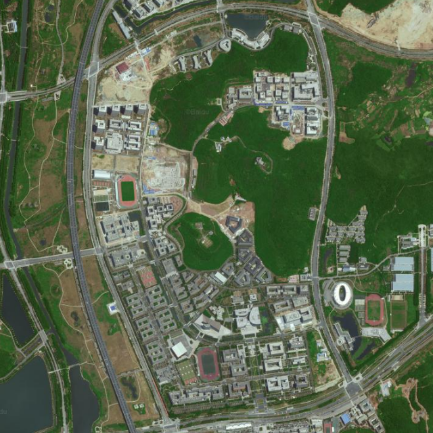
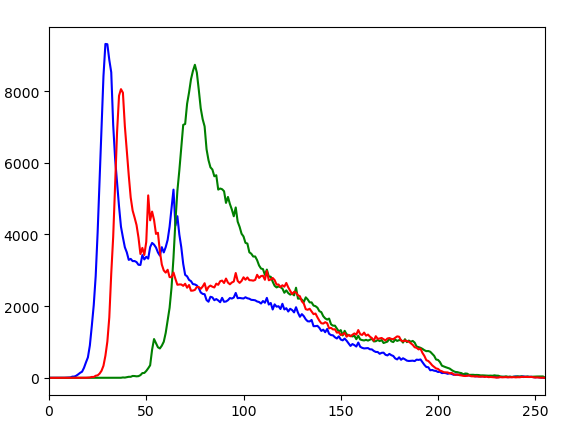
for i, c in enumerate(color): # [(0, 'b'), (1, 'g'), (2, 'r')]

hist = cv2.calcHist([img], [i], None, [256], [0, 255]) # 统计单个通道的直方图

plt.plot(hist, color = c) # 绘制直方图，并指定颜色

plt.xlim([0, 255])

plt.show()

1. 彩色图像(左)与其R、G、B通道直方图(右)
   1. matplotlib.pyplot方法

代码：

plt.hist(x, bins=10, range=None, ...)

参数包括：

x 一维数组。可用ravel()或flatten()来转换成一维数组

bins 将x分的份数。如果图像是8bit，则图像分成256类，即每一灰度级为一类，也即实现了图像灰度级的直方图统计

range 灰度级的范围，8bit为[0, 255]

返回值包括：

n 每份的值

bins 每份的坐标，size比n多1

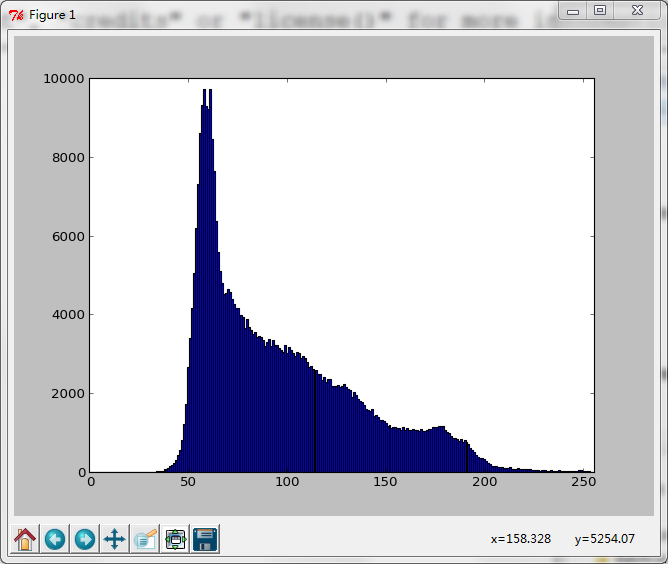
patches list中元素个数

代码：统计并绘制图像直方图

n, bins, patches = plt.hist(img.ravel(), 256, [0, 256])

plt.xlim(0, 255)

plt.show()

1. 灰度图像(左)与其直方图(右)
   1. numpy.histogram()方法

代码：统计图像直方图

hist, bins = np.histogram(a, bins=10, range=None, normed=False, weights=None, density=None)

参数包括：

a 一维数组；

bins 将a所分的份数，8bit图像为256；

range a数值变化范围，8bit图像为[0, 256]；

返回值包括：

hist 直方图统计结果，一维数组

bins 每份对应的坐标值，size比hist多1

代码：统计并绘制图像直方图

# 统计直方图

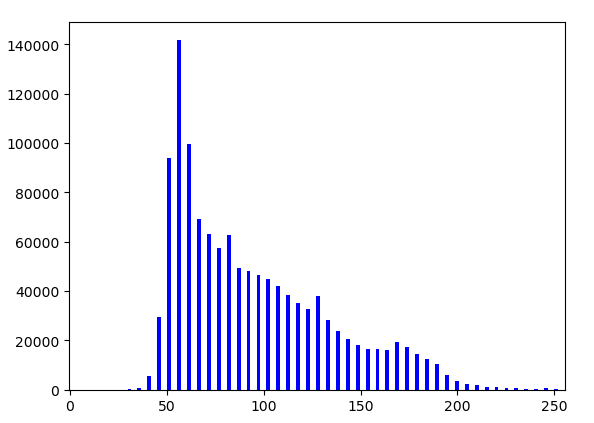
hist, bins = np.histogram(img.flatten(), 50, [0, 256])

# 绘制直方图

plt.bar(bins[:-1], hist, width = 2, color = 'b')

plt.xlim([0-0.5, 256-0.5])

plt.show()

1. 灰度图像(左)与其直方图(右)
2. 直方图调整
   1. 直方图均衡化

根据直方图均衡化的原理，具体步骤如下：

(1) 统计原图像每一级灰度的像素数和累积像素数；

(2) 计算每一级灰度值均衡化后对应的新值，并对其四舍五入取整，得到新灰度级；

(3) 以新值替代原灰度级，形成均衡化后的新图像；

(4) 根据原图像像素统计值对应找到新图像像素统计值，做出新直方图。

表1 直方图均衡化计算实例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 原灰度级 | 像素数 | 累积像素统计值 | 变换后值 | 新灰度级 | 新像素数 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 2 | 0.6 | 1 | 4 |
| 2 | 2 | 4 | 1.3 | 1 |
| 3 | 1 | 5 | 1.6 | 2 | 1 |
| 4 | 3 | 8 | 2.6 | 3 | 3 |
| 5 | 2 | 12 | 3.9 | 4 | 2 |
| 6 | 3 | 15 | 4.9 | 5 | 3 |
| 7 | 4 | 19 | 6.2 | 6 | 4 |
| 8 | 6 | 25 | 8.2 | 8 | 6 |
| 9 | 5 | 30 | 9.8 | 10 | 5 |
| 10 | 5 | 33 | 10.8 | 11 | 5 |
| 11 | 4 | 37 | 12.1 | 12 | 4 |
| 12 | 3 | 40 | 13.1 | 13 | 3 |
| 13 | 3 | 43 | 14.0 | 14 | 3 |
| 14 | 2 | 45 | 14.7 | 15 | 4 |
| 15 | 2 | 47 | 15.3 | 15 |
| 16 | 2 | 49 | 16 | 16 | 2 |

代码：实现图像的直方图均衡化

# 定义函数，统计和绘制图像直方图和图像累积直方图

def histPlot(src):

row = src.shape[0]

col = src.shape[1]

# 统计图像直方图

hist = np.zeros(256, dtype=np.float32)

for i in range(row):

for j in range(col):

index = src[i, j]

hist[index] += 1

# 统计图像累积直方图

cumHist = np.zeros(256, dtype=np.float32)

cumHist[0] = hist[0]

for i in range(1, 256):

cumHist[i] = cumHist[i - 1] + hist[i]

DN = range(256)

#双Y轴展示直方图和积累图

fig = plt.figure()

ax1 = fig.add\_subplot(111)

ax1.plot(DN, hist, color='b', linewidth=2, label = 'Histogram')

ax1.set\_ylabel('Histogram')

ax2 = ax1.twinx() # important function

ax2.plot(DN, cumHist, color='r', linewidth = 2, label = 'Cumulative Histogram')

ax2.set\_ylabel('Cumulative Histogram')

#整饰

fig.legend(loc=1, bbox\_to\_anchor=(1,1), bbox\_transform = ax1.transAxes)

fig.tight\_layout()

plt.xlim([0, 256])

plt.show()

return cumHist

# 定义函数，实现图像的直方图均衡化

def histEqual(src):

row = src.shape[0]

col = src.shape[1]

cumHist = histPlot(src)

# 生成查找表，利用查找表进行图像直方图均衡化

lut = np.zeros(256, dtype=np.float32)

pixelnum = row \* col

for i in range(256):

lut[i] = 255.0 / pixelnum \* cumHist[i]

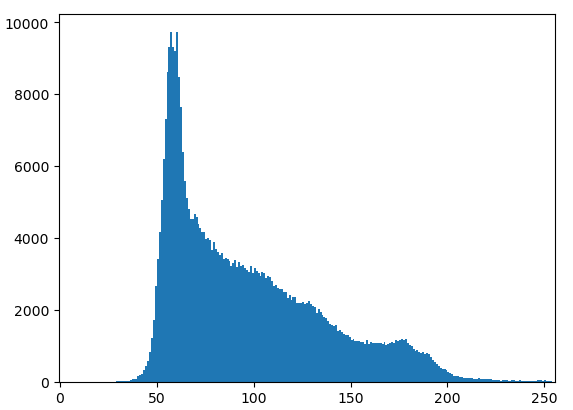
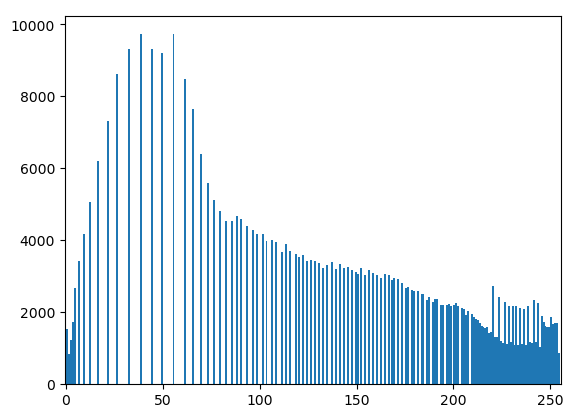
lut = np.uint8(lut + 0.5)

dst = cv2.LUT(src, lut) # 查找表变换

return dst

1. 原图像(左)与直方图均衡化后的图像(右)

1. 图像直方图均衡化前(左)和之后(右)的直方图

**思考：对彩色图像进行直方图均衡化会出现什么情况？**

* 1. 直方图规定化/直方图匹配

直方图规定化的具体步骤如下：

(1) 做出原图像直方图及累积直方图

(2) 做出参考图像的直方图及累积直方图

(3) 对于原图像中的每一灰度级的累积值，在参考直方图中找到对应的累积值；如果为数学公式可以直接计算求值，得到对应的新灰度值

(4) 以新值替代原灰度值，形成均衡化后的新图像

(5) 根据原图像像素统计值对应找到新图像像素统计值，做出新直方图

表2 参考图像直方图统计表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 灰度级 | 9/16 | 10/16 | 11/16 | 12/16 | 13/16 | 14/16 | 15/16 | 16/16 |
| 像素数 | 3 | 6 | 7 | 9 | 8 | 7 | 5 | 4 |
| 累积像素数 | 0.06 | 0.18 | 0.33 | 0.51 | 0.67 | 0.82 | 0.92 | 1.00 |

表3 直方图规定化计算实例

| 原灰度级 | 像素数 | 累积像素数 | 对应的参考累积像素数 | 新灰度级 | 新像素数 |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1/16 | 2 | 0.04 | 0.06 | 9/16 | 5 |
| 2/16 | 2 | 0.08 | 0.06 |
| 3/16 | 1 | 0.10 | 0.06 |
| 4/16 | 3 | 0.16 | 0.18 | 10/16 | 5 |
| 5/16 | 2 | 0.24 | 0.18 |
| 6/16 | 3 | 0.31 | 0.33 | 11/16 | 7 |
| 7/16 | 4 | 0.39 | 0.33 |
| 8/16 | 6 | 0.51 | 0.51 | 12/16 | 6 |
| 9/16 | 5 | 0.61 | 0.67 | 13/16 | 10 |
| 10/16 | 5 | 0.67 | 0.67 |
| 11/16 | 4 | 0.76 | 0.82 | 14/16 | 7 |
| 12/16 | 3 | 0.82 | 0.82 |
| 13/16 | 3 | 0.88 | 0.92 | 15/16 | 5 |
| 14/16 | 2 | 0.92 | 0.92 |
| 15/16 | 2 | 0.96 | 1.00 | 16/16 | 4 |
| 16/16 | 2 | 1.00 | 1.00 |

代码：实现图像的直方图匹配

#统计积累直方图

def cumuFre(src):

row = src.shape[0]

col = src.shape[1]

hist = np.zeros(256, dtype = np.float32)

cumuHist = np.zeros(256, dtype = np.float32)

for i in range(row):

for j in range(col):

index = src[i, j]

hist[index] += 1

cumuHist[0] = hist[0]

for i in range(1, 256):

cumuHist[i] = cumuHist[i-1] + hist[i]

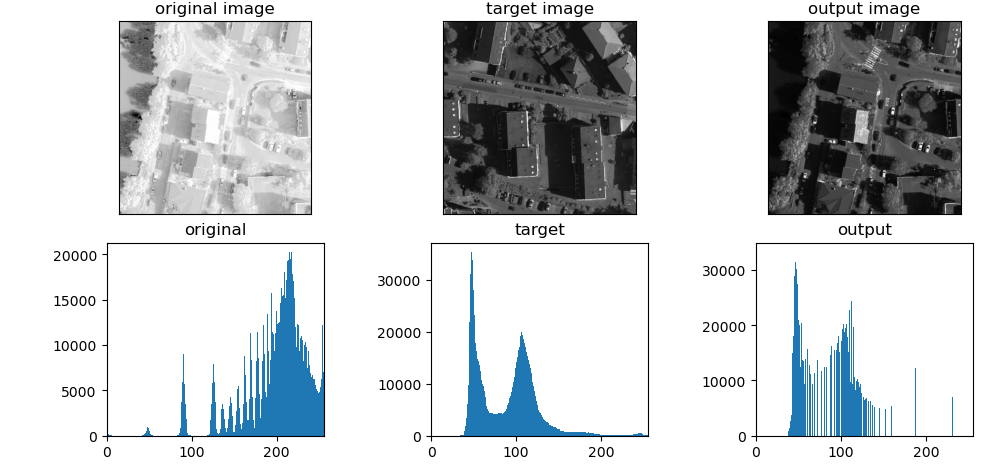
cumuHist = cumuHist / np.float32(row\*col)

return cumuHist

#直方图匹配

def histMatching(oriImg, tarImg):

作业：补全此功能



1. 输出结果