实验二 数字图像的图像增强和图像去雾

一、实验目的

1. 掌握基于灰度变换的图像增强方法；

2. 掌握基于直方图均衡化的图像增强方法。

二、实验内容

（1）用 OpenCV 编写一个程序实现图像的灰度变换，并分析灰度变换中参数值不同时，对图像增强效果的影响；

（2）用 OpenCV 编写一个程序实现基于直方图均衡化的图像增强，包括灰度图像和彩色图像；

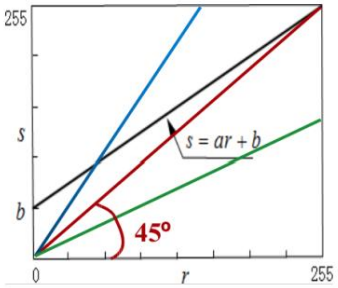
（3）用 OpenCV 编写一个程序实现局部直方图均衡化，增强图像的某些局部区域的细节。

三、实验过程

（1）基于灰度变换的图像增强

- 线性点运算

  使用 s = ar+b 的形式对图像进行线性变换



线性点运算

黑线：$0 < a < 1, b > 0$　　输出灰度压缩

红线：$a = 1, b = 1$　　输出灰度不变

蓝线：$a > 1, b = 0$　　输出灰度扩展，整体变亮

绿线：$0 < a < 1, b = 0$　　输出灰度压缩，整体变暗

- 灰度反转（a=-1, b=0）
- 灰度上移（a=1, b=50）

代码实现：

```python
# -- coding: utf-8 --
import cv2
import matplotlib.pyplot as plt
import numpy as np
img=cv2.imread(r'D:\i_building.jpg',0)

# 获取图像高度和宽度
height = img.shape[0]
width = img.shape[1]

#创建一幅图像， uint8是专门用于存储各种图像的（包括RGB，灰度图像等），范围是从0 - 255
result = np.zeros((height, width), np.uint8)
result1 = np.zeros((height, width), np.uint8)

# 图像灰度反转
for i in range(height):
    for j in range(width):
        gray = -(img[i, j])+255
        result[i, j] = np.uint8(gray)

# 图像灰度上移
for i in range(height):
    for j in range(width):
        if (img[i, j])+50>255:
            gray = 255
        else:
            gray = (img[i, j])+50
        result1[i, j] = np.uint8(gray)

# 显示图形
plt.figure(num='comparison')
titles = ['gray Image', 'gray scale inversion', 'brightness increased']
images = [img, result, result1]
for i in range(3):
    plt.subplot(1, 3, i + 1)
    plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

比较不同 a 值、b 值下图像增强的效果

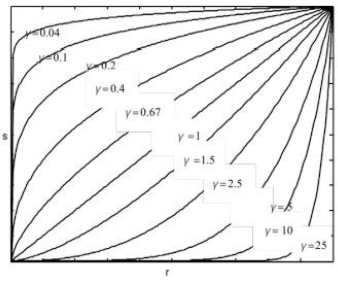
- 幂次变换

幂次变换的一般形式为：$s = c\ r^{\gamma}$

其中 c 和 γ 为正常数。

$0 < \gamma < 1$

加亮、减暗图像



$\gamma > 1$

加暗、减亮图像

采用不同的γ对输入图像进行幂次变换，可对原图像的对比度进行调整，获得不同的清晰度感觉。因此可以在主观经验和感受的技术上，选择适当的γ，来增强图像的清晰度。

代码实现：

```
# -- coding: utf-8 --
import cv2
import matplotlib.pyplot as plt
import numpy as np
from math import log10
img=cv2.imread(r'D:\i_building.jpg',0)

#r=0.5 , c=1
img=np.double(img)
result1=img**0.5
result1= np.uint8(result1*255/np.max(result1))

#r=2 , c=1
img=np.double(img)
result2=img**2
result2= np.uint8(result2*255/np.max(result2))


# 显示图形
plt.figure(num='comparison')
titles = ['gray Image', 'r=0.5', 'r=2']
images = [img, result1, result2]
for i in range(3):
    plt.subplot(1, 3, i + 1)
    plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

比较不同γ值下图像增强的效果。

（2）直方图均衡化

直方图均衡化的中心思想是把原始图像的灰度直方图从比较集中的某个灰度区间转变为全范围均匀分布的灰度区间，通过该处理，增加了像素灰度值的动态范围，从而达到增强图像整体对比度的效果。直方图均衡化包括三个核心步骤：
（a）统计直方图中每个灰度级出现的次数；
（b）计算累计归一化直方图；
（c）重新计算像素点的像素值。

通过对原始图像进行直方图均衡化处理，可得到色彩更均衡的图像。

opencv 提供了 calcHist 函数来计算图像直方图，该函数的变量列表如下：

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

参数解析：
hist：输出直方图。举个例子，对于灰度图来说，有 256 个不同的像素值，因此 hist 是一个 256 行，1 列的矩阵，每行中的数值表示图像中对应这个像素值的像素点数量。
images：源图像，应该是把图像放在[]中如：[img]
channels：用于计算直方图的 dims 通道列表。也以方括号给出。它是我们计算直方图的通道的索引。例如，如果输入为灰度图像，则其值为[0]。对于彩色图像，您可以传递[0]，[1]或[2]分别计算蓝色，绿色或红色通道的直方图。
mask：老生常谈，掩膜。需要需找图像特定区域的直方图时才建。
histSize：BIN 计数，需要放在[]中，对于全尺寸为[256]。
ranges：通常为[0,256]。
accumulate：累加标志。如果设置了，则在分配直方图时，直方图在开始时不会被清除。此功能使您能够从多组数组中计算单个直方图，或者及时更新直方图。
代码实现：

```python
# -*- coding: utf-8 -*-
import cv2
import numpy as np
import matplotlib.pyplot as plt
#读取图片
img = cv2.imread('D:/Pic/lena.bmp')
#灰度转换
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#直方图均衡化处理
result = cv2.equalizeHist(gray)
#显示图像
plt.subplot(221)
plt.imshow(gray, cmap=plt.cm.gray), plt.axis("off"), plt.title('(a)')
plt.subplot(222)
plt.imshow(result, cmap=plt.cm.gray), plt.axis("off"), plt.title('(b)')
plt.subplot(223)
plt.hist(img.ravel(), 256), plt.title('(c)')
plt.subplot(224)
plt.hist(result.ravel(), 256), plt.title('(d)')
plt.show()
```

思考：彩色图像的直方图均衡化如何实现？

（3）局部直方图均衡化

通过调用 OpenCV 中的 equalizeHist()函数实现直方图均衡化处理，该方法简单高校，但是它是一种全局意义上的均衡化处理，很多时候这种操作会把某些不该调整的部分给均衡处理了。因此，常常需要增强图像的某些局部区域细节。

在 OpenCV 中，调用函数 createCLAHE()实现对比度受限的局部直方图均衡化。它将整个图像分成许多小块，那么对每个小块进行均衡化。这种方法主要对于图像直方图不是那么单一的（比如存在多峰情况）图像比较实用。其函数原型如下所示：

retval=createCLAHE([,clipLimit[,tileGridSize]])

其中，clipLimit 参数表示对比度的大小，tileGridSize 表示每次处理块的大小

代码演示：

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
img = cv2.imread('D:/Pic/lena.bmp')

#灰度转换
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#局部直方图均衡化处理
clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(10,10))

#将灰度图像和局部直方图相关联，把直方图均衡化应用到灰度图
result = clahe.apply(gray)

#显示图像
plt.subplot(221)
plt.imshow(gray, cmap=plt.cm.gray), plt.axis("off"), plt.title('(a)')
plt.subplot(222)
plt.imshow(result, cmap=plt.cm.gray), plt.axis("off"), plt.title('(b)')
plt.subplot(223)
plt.hist(img.ravel(), 256), plt.title('(c)')
plt.subplot(224)
plt.hist(result.ravel(), 256), plt.title('(d)')
plt.show()
```

对比分析直方图均衡化和局部直方图均衡化的输出效果。