# 实验三 对比度增强

## 1. 课前回顾

### 1.1 cv2 图像操作

(1) 读取影像

img = cv2.imread('读取的文件名', cv2.IMREAD_GRAYSCALE)

第 1 个参数为读取的文件名及路径，第 2 个参数表示以何种方式显示，如灰度，彩色等

返回值为暂存于内存中的影像（img）

(2) 显示影像

cv2.imshow('输出的窗口名称'，img)

第 1 个参数为输出的窗口名称，第 2 个参数表示要输出的图像

(3) 保存影像

cv2.imwrite('输出文件名', img)

### 1.2 matplotlib 绘图

plt.plot(x, y, format_string, **kwargs)

参数分别为 x 轴数据，y 轴数据，format_string 控制曲线的格式字串

plt.show()

将图像显示出来

## 2. 线性变换

### 2.1 反转变换

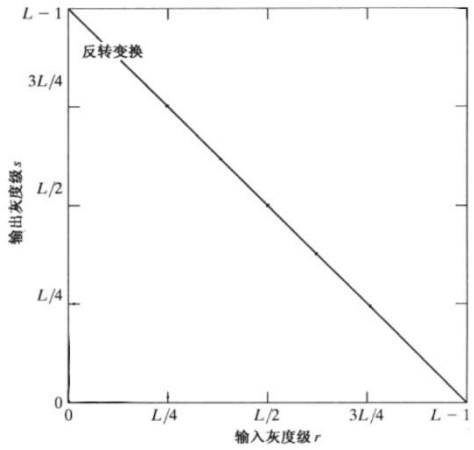


图1. 反转变换函数

使用图 1 中的函数，可以得到灰度级范围为[0，L-1]的一幅图像的反转图像，以增强嵌入在一幅图像的暗区域中的白色或者灰色的细节。其变换公式为：

$$s = L - 1 - r$$

其中：L 为灰度量化级数，r 为输入灰度值，s 为输出灰度级。

图2. 原图像(左)与反转后图像(右)

代码：

```
# coding: utf-8
import cv2


def inverseImg(src):
    dst = 255 - src
    return dst


img = cv2.imread(r'coin.jpg', 0)
cv2.imshow('input', img)


img2 = inverseImg(img)
cv2.imshow('output', img2)
cv2.imwrite(r'coin_out.jpg', img2)
```

```
cv2.waitKey()
cv2.destroyAllWindows()      #注意不要写成 destory
```

## 2.2 最小值-最大值拉伸

假设原图像的对比度较差，其亮度区间为[a, b]，经过线性变换后其亮度区间扩展为[0, 255]，对比度得到增强。8bit 图像的最小-最大值拉伸公式为：

$$g(i,j) = \frac{[f(i,j) - min_{ori}]}{max_{ori} - min_{ori}} \times 255$$

其中： $f(i,j)$是原图像的亮度值，$g(i,j)$是最小-最大值拉伸后的亮度值，$max_{ori}$和$min_{ori}$是原图像的亮度值中的最大值和最小值。

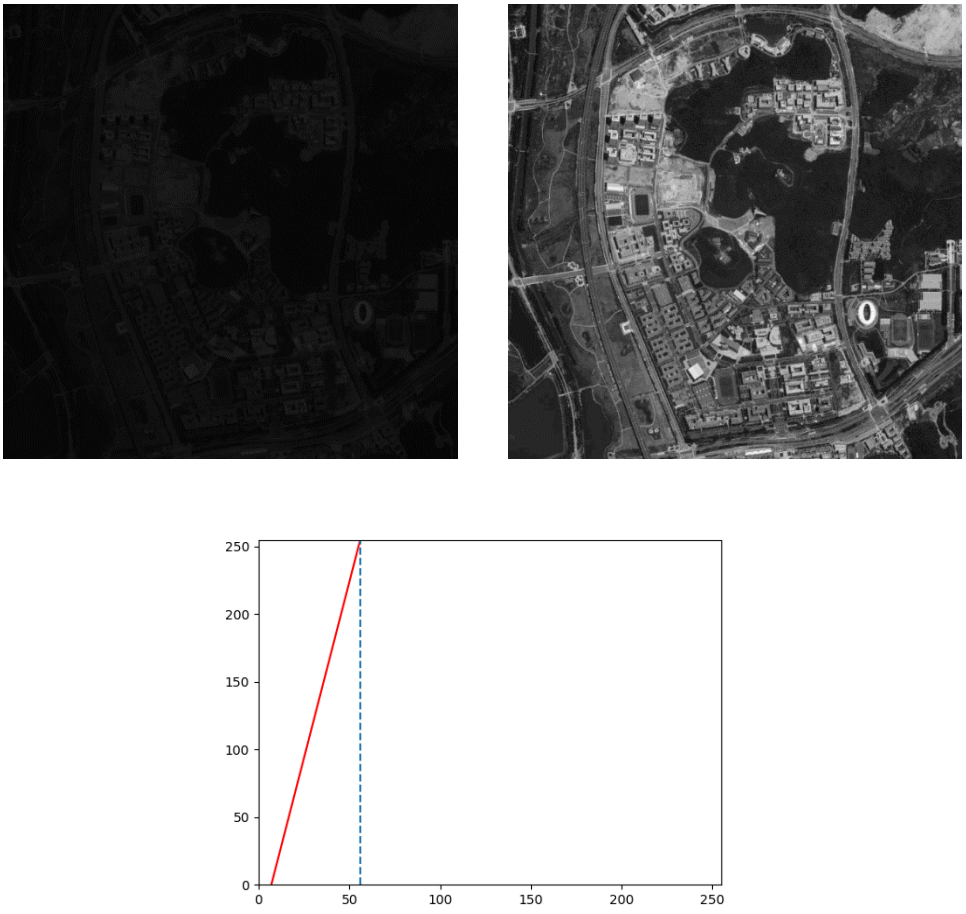


图3. 原图像(左上)、最小-最大值拉伸后图像(右上)及变换函数

代码：

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
#绘制变换函数图像
def linearPlot(x1, x2):
    plt.plot([x1, x2], [0, 255], 'r')
    # 第一个方框代表 x 的各个标记点，第二个方框代表 y 的各个标记点，第三个显示风格
    plt.plot([x2, x2], [0, 255], '--')
    plt.xlim(0, 255), plt.ylim(0, 255)
    plt.show()

def linearStretch(src):
    smax = np.max(src)
    smin = np.min(src)
    linearPlot(smin, smax)   #调用绘图函数
    dst = 255.0 * (src - smin) / (smax - smin)
    dst = np.uint8(dst + 0.5)
    return dst

img = cv2.imread(r'NJU_gray.tif', 0)
cv2.imshow('input', img)
cv2.waitKey(0)
# waitKey() 函数的功能是不断刷新图像 , 频率时间为 delay , 单位为 ms；
# waitKey(0) , 则表示程序会无限制的等待用户的按键事件

img2 = linearStretch(img)
cv2.imshow('output', img2)
cv2.waitKey(0)

cv2.imwrite(r'NJU_output.tif', img2)
cv2.waitKey()
cv2.destroyAllWindows() # 删除所有窗口；若要删除特定的窗口，往输入特定的窗口值
print('done!')
```

## 2.3 裁剪百分比拉伸

截去图像最小值以上和最大值以下指定百分比范围内的像元值，其他部分拉伸到全局范围，实现图像的增强。

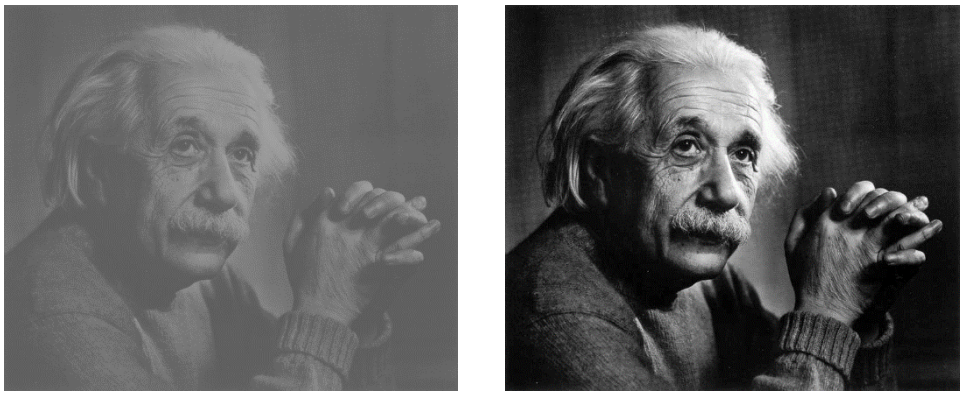

图4. 2%线性拉伸前(左)和拉伸后(右)的爱因斯坦相片

代码：

```
import cv2
import numpy as np

def linearPercentStretch(src, percent):
    src = np.float64(src)
    cut = np.floor(256 * percent + 0.5)  #计算需要裁减的数量；floor()向下取整
    minvalue = np.min(src)
    maxvalue = np.max(src)

    newminvalue = minvalue + cut
    newmaxvalue = maxvalue - cut

    row, col = src.shape
    ans = np.array((row, col), dtype = np.float64)
    ans = 255.0 * (src - newminvalue) / (newmaxvalue - newminvalue)
    ans = ans + 0.5
    ans[ans > 255] = 255
    ans[ans < 0] = 0
    ans = np.uint8(ans)
    return ans

img = cv2.imread(r'Einstein.tif', 0)
cv2.imshow('input', img)

img2 = linearPercentStretch(img, 0.02)

cv2.imshow('output', img2)
cv2.imwrite(r'Einstein_out.tif', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

## 2.4 分段线性变换

分段线性变换的作用是对不同亮度值范围分别进行对比度调整。用分段线性函数将原图像亮度值拉伸到指定亮度范围。使用分段线性变换前提是必须知道地物与直方图的对应关系。

图5. 分段变换前(左)和变换后(右)的图像

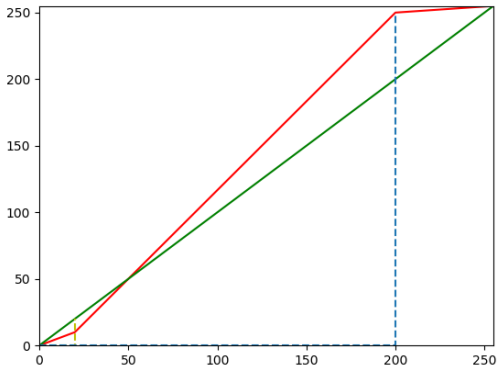


图6. 分段线性变换函数

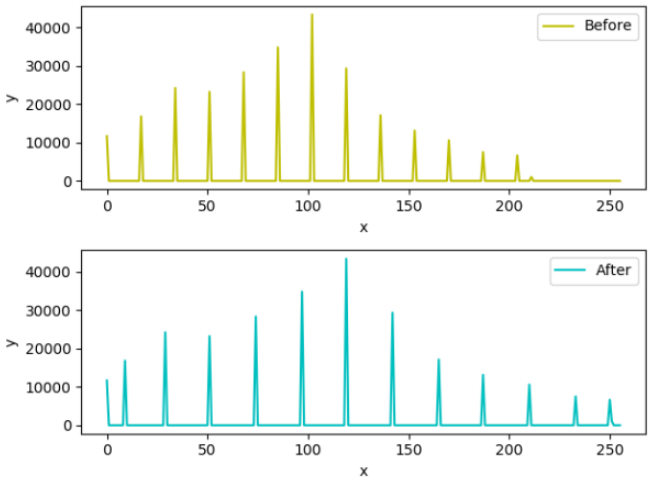

图 7. 变化前后图像直方图对比

代码：分段线性变换

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

# 分段线性拉伸
def LDSPlot(x1, y1, x2, y2):
    plt.plot([0, x1, x2, 255], [0, y1, y2, 255], 'r' , label = 'Targetline',
linewidth=2.5) # 红色实线，定义 label，线宽
    plt.plot([0, 255], [0, 255], 'g' , label = 'Guideline', linewidth=2.5)
    plt.plot([0, x1, x1], [0, 0, x1], 'y--') # 黄色虚线
    plt.plot([0, x2, x2], [0, 0, y2], '--') # 虚线
    plt.xlim(0, 255), plt.ylim(0, 255)
    plt.legend() # 显示图例
    plt.show()

def linearDividedStretch(src, x1, y1, x2, y2):
    #以查找表的方式做分段线性变换
    LDSPlot(x1, y1, x2, y2)
    x1, x2, y1, y2 = np.float(x1), np.float(x2), np.float(y1), np.float(y2)

    x = np.arange(256) # 列数为 256 的一维数组
    lut = np.zeros(256, dtype = np.float64)  #定义一个空表作为容器，储存每一灰度值对应变换
后的值

    #填写内容
    for i in x:
        if(i < x1):
            lut[i] = (y1 * 1.0 / x1) * i
        elif(i < x2):
            lut[i] = (y2 - y1) / (x2 - x1) * (i - x1) + y1
        else:
            lut[i] = (255 - y2) / (255 - x2) * (i - x2) + y2

    lut = np.uint8(lut + 0.5)
    dst = cv2.LUT(src, lut)
    return dst

img = cv2.imread(r'city_gamma.tif', 0)
cv2.imshow('input', img)

img2 = linearDividedStretch(img, 20, 10, 200, 250)
```

```
cv2.imshow('output', img2)
cv2.imwrite(r'city_out.tif', img2)
cv2.waitKey()

# 计算直方图，比较拉伸前后直方图变化
histb = cv2.calcHist([img], [0], None, [256], [0, 255])
hista = cv2.calcHist([img2], [0], None, [256], [0, 255])

# 对第 1 子图进行设定
plt.subplot(2, 1, 1)
plt.plot(histb, 'y', label = 'Before')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
# 对第 2 子图进行设定
plt.subplot(2, 1, 2)
plt.plot(hista, 'c', label = 'After')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()

cv2.destroyAllWindows()
print('done!')
```

查找表：cv2.LUT (InputArray src, InputArray lut, OutputArray dst)

• src 表示的是 8bit 输入图像(可以是单通道也可是 3 通道)；

• lut 表示包含 256 个元素查找表。查找表也可以是单通道，也可以是 3 通道。如果输入图像为单通道，那查找表必须为单通道；若输入图像为 3 通道，查找表可以为单通道，也可以为 3 通道。若为单通道则表示对图像 3 个通道都应用这个表，若为 3 通道则分别应用；

• dst 表示输出图像，大小、通道数、深度与 src、lut 相同。

**参考网址：**

https://docs.opencv.org/master/d2/de8/group__core__array.html#gab55b8d062b7f5587720ede032d34156f

## 2.5 削波

# 削　波

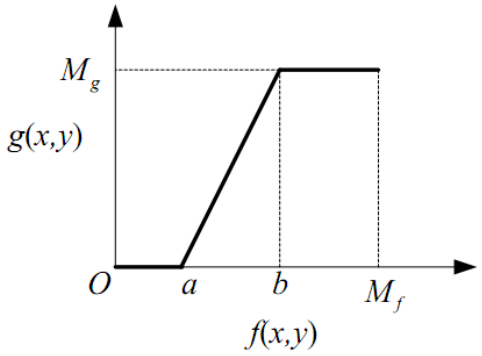


图8.　原图像(左)与削波后图像(右)

代码：

```
# coding: utf-8

import cv2
import numpy as np

#削波
def clipStretch(src, x1, x2):
    src = np.float32(src)
    dst = 255.0 * (src - x1) / (x2 - x1)
    dst[dst > 255] = 255
    dst[dst < 0] = 0
    dst = np.uint8(dst + 0.5)
    return dst

img = cv2.imread(r'Mary.jpg', 0)
cv2.imshow('input', img)
```

```
img2 = clipStretch(img, 50, 150)
cv2.imshow('output', img2)
cv2.imwrite(r'Mary_out.jpg', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

## 2.6 阈值化

图9. 原图像(左)与阈值化后图像(右)

代码：

```
# coding: utf-8
import cv2
import numpy as np

def thresholdImage(src, thres):
    src[src > thres] = 255
    src[src <= thres] = 0
    dst = src
    return dst

img = cv2.imread(r'NJU_gray_1.tif', 0)
cv2.imshow('input', img)



img2 = thresholdImage(img, 70)

cv2.imshow('output', img2)
cv2.imwrite(r'NJU_threshold_out.jpg', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

# 3. 非线性变换

## 3.1 对数变换

对数变换的通用形式为：

$$g(i,j) = c \times \log[1 + f(i,j)]$$

其中$c$是一个常数，并假设$c \geq 0$。该变换将输入中范围较窄的低灰度值映射为输出中较宽范围的灰度值，将输入中范围较宽的高灰度值映射为输出中较窄范围的灰度值。我们使用这种类型的变换来扩展图像中暗像素的值，压缩更高灰度级的值。

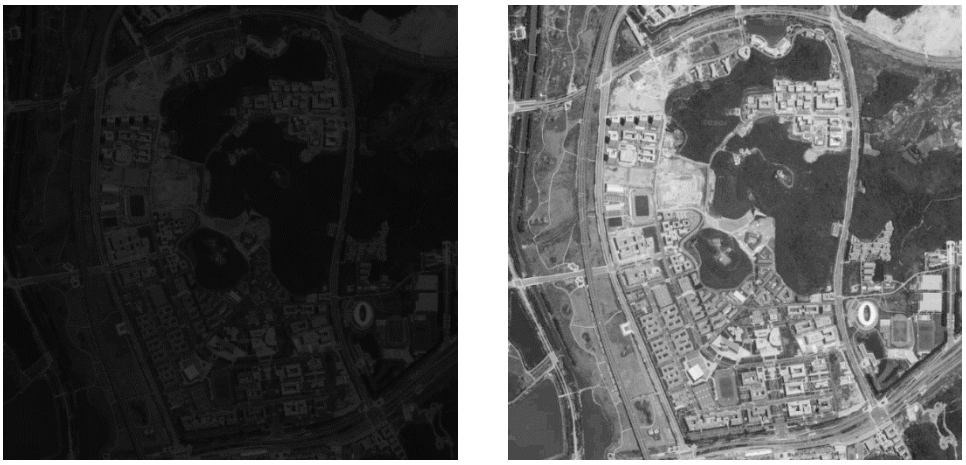

图10. 原图像(左)与对数变换后图像(右)

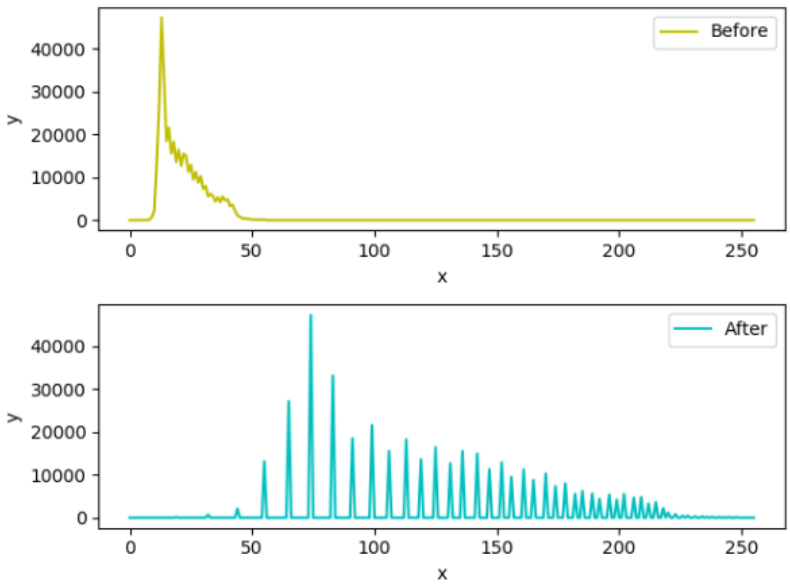


图11. 变换前后直方图对比图

代码：

```
# coding: utf-8
import cv2
```

```
import numpy as np
import matplotlib.pyplot as plt

#对数变换
def logStretch(src, a, b):
    src = np.float32(src)
    dst = a * np.log(1 + src) + b # 四舍五入
    dst = np.uint8(dst + 0.5) # 强制转到[0-255]
    return dst

img = cv2.imread('NJU_gray.tif', cv2.IMREAD_GRAYSCALE)
cv2.imshow('input', img)

img2 = logStretch(img, 125, 0.5)   #设计一个函数使图像灰度值拉伸到 0-255
cv2.imshow('output', img2)
cv2.imwrite('NJU_gray_2.tif', img2)
cv2.waitKey(0)

# 计算直方图
histb = cv2.calcHist([img], [0], None, [256], [0, 255])
hista = cv2.calcHist([img2], [0], None, [256], [0, 255])

# 对第 1 子图进行设定
plt.subplot(2, 1, 1)
plt.plot(histb, 'y', label='Before')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
# 对第 2 子图进行设定
plt.subplot(2, 1, 2)
plt.plot(hista, 'c', label='After')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()

cv2.destroyAllWindows()
```

**问题：对数变换中常数 c 的取值大小影响什么？**