该系列文章是讲解Python OpenCV图像处理知识，前期主要讲解图像入门、OpenCV基础用法，中期讲解图像处理的各种算法，包括图像锐化算子、图像增强技术、图像分割等，后期结合深度学习研究图像识别、图像分类应用。希望文章对您有所帮助，如果有不足之处，还请海涵~

该系列在github所有源代码：https://github.com/eastmountyxz/ImageProcessing-Python
PS：请求帮忙点个Star，哈哈，第一次使用Github，以后会分享更多代码，一起加油。

同时推荐作者的C++图像系列知识：
[数字图像处理] 一.MFC详解显示BMP格式图片
[数字图像处理] 二.MFC单文档分割窗口显示图片
[数字图像处理] 三.MFC实现图像灰度、采样和量化功能详解
[数字图像处理] 四.MFC对话框绘制灰度直方图
[数字图像处理] 五.MFC图像点运算之灰度线性变化、灰度非线性变化、阈值化和均衡化处理详解
[数字图像处理] 六.MFC空间几何变换之图像平移、镜像、旋转、缩放详解
[数字图像处理] 七.MFC图像增强之图像普通平滑、高斯平滑、Laplacian、Sobel、Prewitt锐化详解

前文参考：
[Python图像处理] 一.图像处理基础知识及OpenCV入门函数
[Python图像处理] 二.OpenCV+Numpy库读取与修改像素
[Python图像处理] 三.获取图像属性、兴趣ROI区域及通道处理
[Python图像处理] 四.图像平滑之均值滤波、方框滤波、高斯滤波及中值滤波
[Python图像处理] 五.图像融合、加法运算及图像类型转换
[Python图像处理] 六.图像缩放、图像旋转、图像翻转与图像平移
[Python图像处理] 七.图像阈值化处理及算法对比
[Python图像处理] 八.图像腐蚀与图像膨胀
[Python图像处理] 九.形态学之图像开运算、闭运算、梯度运算
[Python图像处理] 十.形态学之图像顶帽运算和黑帽运算
[Python图像处理] 十一.灰度直方图概念及OpenCV绘制直方图
[Python图像处理] 十二.图像几何变换之图像仿射变换、图像透视变换和图像校正
[Python图像处理] 十三.基于灰度三维图的图像顶帽运算和黑帽运算
[Python图像处理] 十四.基于OpenCV和像素处理的图像灰度化处理
[Python图像处理] 十五.图像的灰度线性变换

前一篇文章讲解了图像灰度化处理及线性变换知识，结合OpenCV调用cv2.cvtColor()函数实现图像灰度操作，本篇文章主要讲解非线性变换，使用自定义方法对图像进行灰度化处理，包括对数变换和伽马变换。本文主要讲解灰度线性变换，基础性知识希望对您有所帮助。

1.图像灰度非线性变换：DB=DA×DA/255

2.图像灰度对数变换
3.图像灰度伽玛变换

PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时参考如下文献：
杨秀璋等. 基于苗族服饰的图像锐化和边缘提取技术研究[J]. 现代计算机，2018(10).
《数字图像处理》（第3版），冈萨雷斯著，阮秋琦译，电子工业出版社，2013年.
《数字图像处理学》（第3版），阮秋琦，电子工业出版社，2008年，北京.
《OpenCV3编程入门》，毛星云，冷雪飞，电子工业出版社，2015.
[数字图像处理] 五.MFC图像点运算之灰度线性变化、灰度非线性变化、阈值化和均衡化处理详解
python+opencv+图像特效（图像灰度处理、颜色翻转、图片融合，边缘检测，浮雕效果）
数字图像处理-空间域处理-灰度变换-基本灰度变换函数
OpenCV图像增强算法实现（直方图均衡化、拉普拉斯、Log、Gamma）

一.图像灰度非线性变换：DB=DA×DA/255

图像的灰度非线性变换主要包括对数变换、幂次变换、指数变换、分段函数变换，通过非线性关系对图像进行灰度处理，下面主要讲解三种常见类型的灰度非线性变换。

原始图像的灰度值按照DB=DA×DA/255的公式进行非线性变换，其代码如下：

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

#读取原始图像
img = cv2.imread('miao.png')

#图像灰度转换
grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

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

#创建一幅图像
result = np.zeros((height, width), np.uint8)

#图像灰度非线性变换：DB=DA×DA/255
for i in range(height):
    for j in range(width):
```

```
            gray = int(grayImage[i,j])*int(grayImage[i,j]) / 255
            result[i,j] = np.uint8(gray)

    #显示图像
    cv2.imshow("Gray Image", grayImage)
    cv2.imshow("Result", result)

    #等待显示
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

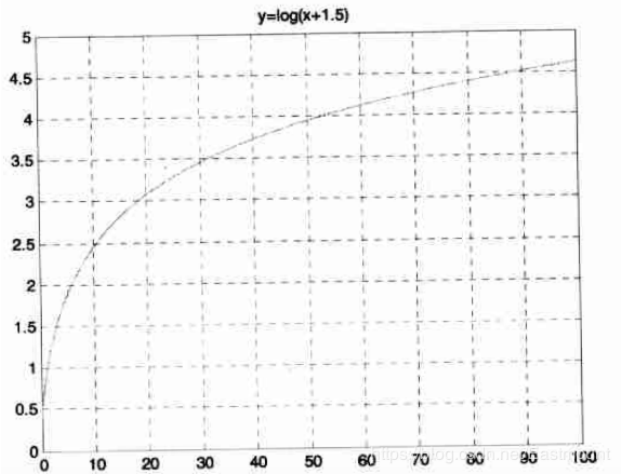
图像灰度非线性变换的输出结果下图所示：

二.图像灰度对数变换

图像灰度的对数变换一般表示如公式所示：

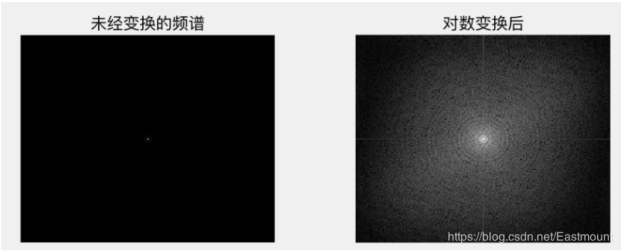
$$D_B = c \times \log(1 + D_A)$$

其中c为尺度比较常数，DA为原始图像灰度值，DB为变换后的目标灰度值。如下图所示，它表示对数曲线下的灰度值变化情况。



由于对数曲线在像素值较低的区域斜率大，在像素值较高的区域斜率较小，所以图像经过对数变换后，较暗区域的对比度将有所提升。这种变换可用于增强图像的暗部细节，从而用来扩展被压缩的高值图像中的较暗像素。

对数变换实现了扩展低灰度值而压缩高灰度值的效果，被广泛地应用于频谱图像的显示中。一个典型的应用是傅立叶频谱，其动态范围可能宽达0～106直接显示频谱时，图像显示设备的动态范围往往不能满足要求，从而丢失大量的暗部细节；而在使用对数变换之后，图像的动态范围被合理地非线性压缩，从而可以清晰地显示。在下图中，未经变换的频谱经过对数变换后，增加了低灰度区域的对比度，从而增强暗部的细节。

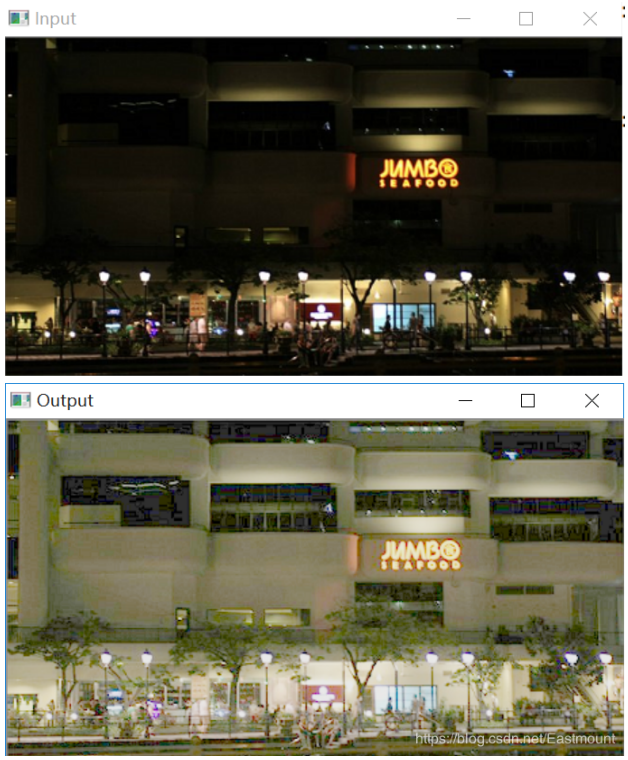


下面的代码实现了图像灰度的对数变换。

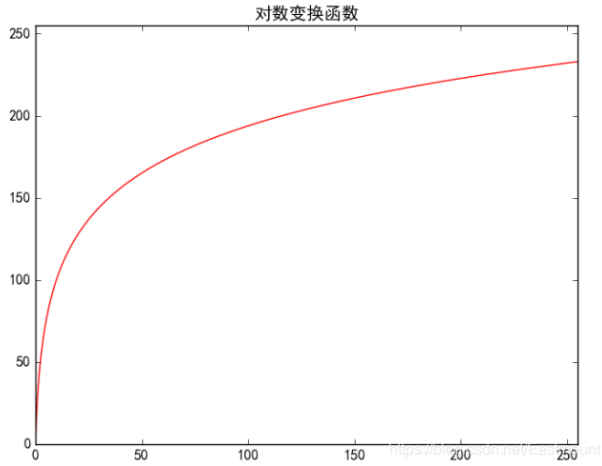
```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
import cv2

#绘制曲线
def log_plot(c):
    x = np.arange(0, 256, 0.01)
    y = c * np.log(1 + x)
    plt.plot(x, y, 'r', linewidth=1)
    plt.rcParams['font.sans-serif']=['SimHei'] #正常显示中文标签
    plt.title(u'对数变换函数')
```

```
    plt.xlim(0, 255), plt.ylim(0, 255)
    plt.show()

#对数变换
def log(c, img):
    output = c * np.log(1.0 + img)
    output = np.uint8(output + 0.5)
    return output

#读取原始图像
img = cv2.imread('test.png')

#绘制对数变换曲线
log_plot(42)

#图像灰度对数变换
output = log(42, img)

#显示图像
cv2.imshow('Input', img)
cv2.imshow('Output', output)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

下图表示经过对数函数处理后的效果图，对数变换对于整体对比度偏低并且灰度值偏低的图像增强效果较好。



对应的对数函数曲线如图



三.图像灰度伽玛变换

伽玛变换又称为指数变换或幂次变换，是另一种常用的灰度非线性变换。图像灰度的伽玛变换一般表示如公式所示：

$$D_B = c \times D_A^{\gamma}$$

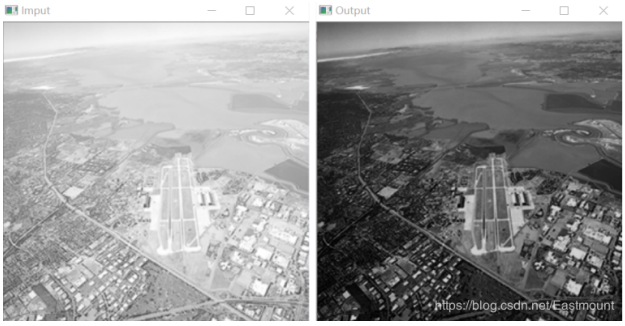
- 当γ>1时，会拉伸图像中灰度级较高的区域，压缩灰度级较低的部分。
- 当γ<1时，会拉伸图像中灰度级较低的区域，压缩灰度级较高的部分。
- 当γ=1时，该灰度变换是线性的，此时通过线性方式改变原图像。

Python实现图像灰度的伽玛变换代码如下，主要调用幂函数实现。

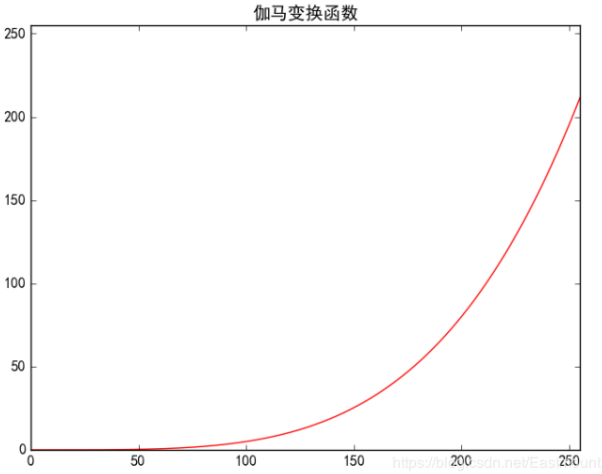
```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
import cv2

#绘制曲线
def gamma_plot(c, v):
    x = np.arange(0, 256, 0.01)
    y = c*x**v
    plt.plot(x, y, 'r', linewidth=1)
    plt.rcParams['font.sans-serif']=['SimHei'] #正常显示中文标签
    plt.title(u'伽马变换函数')
```

```
    plt.xlim([0, 255]), plt.ylim([0, 255])
    plt.show()

#伽玛变换
def gamma(img, c, v):
    lut = np.zeros(256, dtype=np.float32)
    for i in range(256):
        lut[i] = c * i ** v
    output_img = cv2.LUT(img, lut) #像素灰度值的映射
    output_img = np.uint8(output_img+0.5)
    return output_img

#读取原始图像
img = cv2.imread('test.png')

#绘制伽玛变换曲线
gamma_plot(0.00000005, 4.0)

#图像灰度伽玛变换
output = gamma(img, 0.00000005, 4.0)

#显示图像
cv2.imshow('Imput', img)
cv2.imshow('Output', output)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

下图表示经过伽玛变换处理后的效果图，伽马变换对于图像对比度偏低，并且整体亮度值偏高（或由于相机过曝）情况下的图像增强效果明显。



对应的幂律函数曲线如图所示。



文章周日写于钟书阁，女神伴于旁。希望文章对大家有所帮助，如果有错误或不足之处，还请海涵。最近连续奔波考博，经历的事情太多，有喜有悲，需要改变自己好好对女神，也希望读者与我一起加油。

（By：Eastmount 2019-03-31 深夜12点 https://blog.csdn.net/Eastmount/）