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

前面的第十篇文章讲解过图形形态学变换——顶帽运算和黑帽运算，本篇文章继续深入，结合灰度三维图像讲解图像顶帽运算和图像黑猫运算，通过Python调用OpenCV函数实。基础性知识希望对您有所帮助。

1.图像顶帽运算
2.图像黑帽运算
3.基于灰度三维图的顶帽黑帽运算

PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时，本篇文章涉及到《计算机图形学》基础知识，请大家下来补充。

参考文献：
《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
光照不均匀图像分割技巧2——顶帽变换和底帽变换
基于opencv绘制图片的三维空间显示图（python）

一.图像顶帽运算

图像顶帽运算（top-hat transformation）又称为图像礼帽运算，它是用原始图像减去图像开运算后的结果，常用于解决由于光照不均匀图像分割出错的问题。其公式定义如下：

$$T_{hat}(A) = A - (A \circ B)$$

图像顶帽运算是用一个结构元通过开运算从一幅图像中删除物体，校正不均匀光照的影响，其效果图如下图所示。

（a）原始图像　　（b）开运算　　（c）顶帽运算

在Python中，图像顶帽运算主要调用morphologyEx()实现，其中参数cv2.MORPH_TOPHAT表示顶帽处理，函数原型如下：

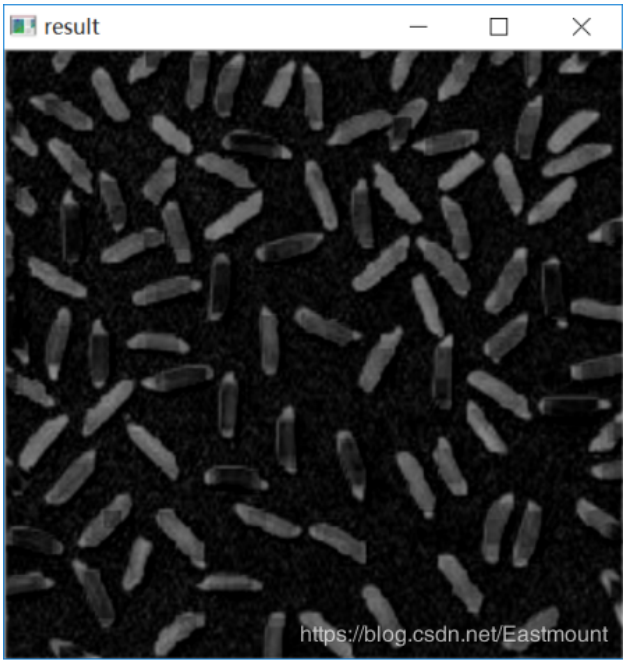
dst = cv2.morphologyEx(src, cv2.MORPH_TOPHAT, kernel)

- src表示原始图像
- cv2.MORPH_TOPHAT表示图像顶帽运算
- kernel表示卷积核，可以用numpy.ones()函数构建

假设存在一张光照不均匀的米粒图像，如图所示，我们需要调用图像顶帽运算解决光照不均匀的问题。其Python代码如下所示：


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

#读取图片
src = cv2.imread('test06.png', cv2.IMREAD_UNCHANGED)

#设置卷积核
kernel = np.ones((10,10), np.uint8)

#图像顶帽运算
result = cv2.morphologyEx(src, cv2.MORPH_TOPHAT, kernel)

#显示图像
cv2.imshow("src", src)
cv2.imshow("result", result)

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

其运行结果如下，它有效地将米粒与背景分离开来。



二.图像黑帽运算

图像底帽运算（bottom-hat transformation）又称为图像黑帽运算，它是用图像闭运算操作减去原始图像后的结果，从而获取图像内部的小孔或前景色中黑点，也常用于解决由于光照不均匀图像分割出错的问题。其公式定义如下：

$$\mathrm{B}_{\mathrm{hat}}(\mathrm{A}) = (A \bullet B) - A$$

图像底帽运算是用一个结构元通过闭运算从一幅图像中删除物体，常用于校正不均匀光照的影响。其效果图如下图所示。

（b）原始图像　（b）闭运算　（c）底帽运算

在Python中，图像底帽运算主要调用morphologyEx()实现，其中参数cv2.MORPH_BLACKHAT表示底帽或黑帽处理，函数原型如下：

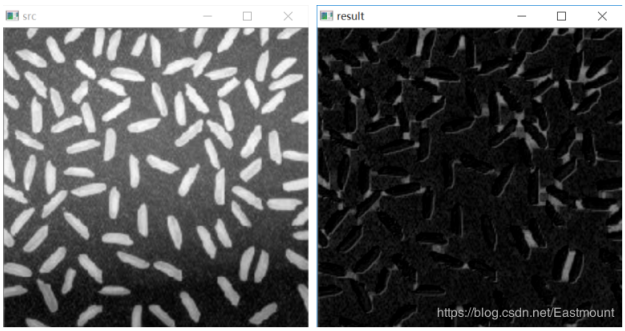
dst = cv2.morphologyEx(src, cv2.MORPH_BLACKHAT, kernel)

- src表示原始图像
- cv2.MORPH_BLACKHAT表示图像底帽或黑帽运算
- kernel表示卷积核，可以用numpy.ones()函数构建

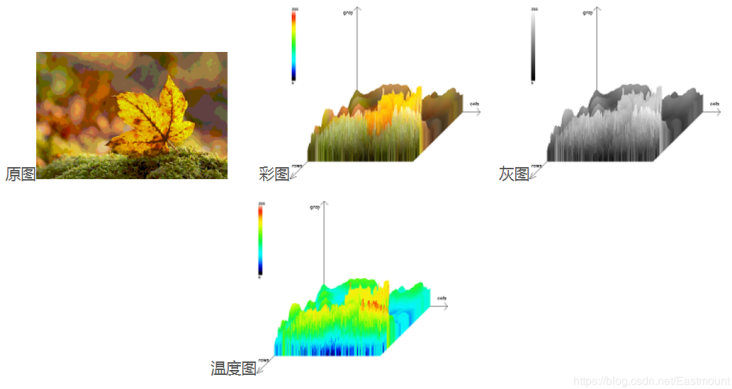
Python实现图像底帽运算的代码如下所示：

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

#读取图片
src = cv2.imread('test06.png', cv2.IMREAD_UNCHANGED)

#设置卷积核
kernel = np.ones((10, 10), np.uint8)

#图像黑帽运算
result = cv2.morphologyEx(src, cv2.MORPH_BLACKHAT, kernel)

#显示图像
cv2.imshow("src", src)
cv2.imshow("result", result)

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

其运行结果如图所示：



三.基于灰度三维图的顶帽黑帽运算

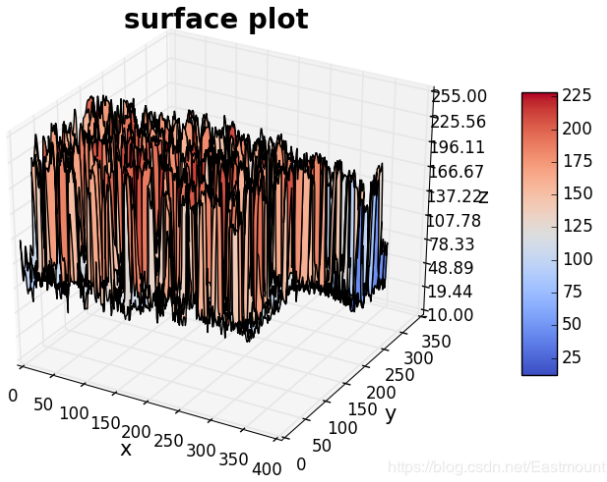


为什么图像顶帽运算会消除光照不均匀的效果呢？通常可以利用灰度三维图来进行解释该算法。灰度三维图主要调用Axes3D包实现，对原图绘制灰度三维图的代码如下：

```
# -*- coding: utf-8 -*-
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

#读取图像
img = cv.imread("test06.png")
img = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
imgd = np.array(img)      #image类转numpy

#准备数据
sp = img.shape
h = int(sp[0])         #图像高度(rows)
w = int(sp[1])        #图像宽度(colums) of image

#绘图初始处理
fig = plt.figure(figsize=(16,12))
ax = fig.gca(projection="3d")

x = np.arange(0, w, 1)
y = np.arange(0, h, 1)
x, y = np.meshgrid(x,y)
z = imgd
surf = ax.plot_surface(x, y, z, cmap=cm.coolwarm)

#自定义z轴
ax.set_zlim(-10, 255)
ax.zaxis.set_major_locator(LinearLocator(10))    #设置z轴网格线的疏密
#将z的value字符串转为float并保留2位小数
```

```
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# 设置坐标轴的label和标题
ax.set_xlabel('x', size=15)
ax.set_ylabel('y', size=15)
ax.set_zlabel('z', size=15)
ax.set_title("surface plot", weight='bold', size=20)

#添加右侧的色卡条
fig.colorbar(surf, shrink=0.6, aspect=8)
plt.show()
```

运行结果如下图所示：

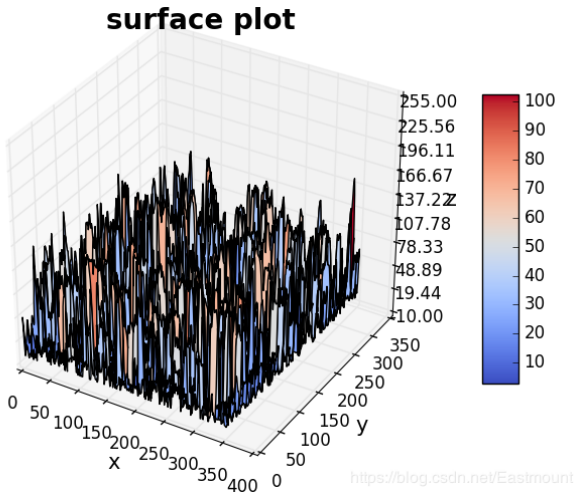


从图像中的像素走势显示了该图受各部分光照不均匀的影响，从而造成背景灰度不均现象，其中凹陷对应图像中灰度值比较小的区域。而通过图像白帽运算后的图像灰度三维图的代码如下：

```
# -*- coding: utf-8 -*-
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

#读取图像
img = cv.imread("test06.png")
img = cv.cvtColor(img,cv.COLOR_BGR2GRAY)

#图像黑帽运算
kernel = np.ones((10,10), np.uint8)
result = cv.morphologyEx(img, cv.MORPH_BLACKHAT, kernel)
```

```
#image类转numpy
imgd = np.array(result)

#准备数据
sp = result.shape
h = int(sp[0])         #图像高度(rows)
w = int(sp[1])        #图像宽度(colums) of image

#绘图初始处理
fig = plt.figure(figsize=(8,6))
ax = fig.gca(projection="3d")

x = np.arange(0, w, 1)
y = np.arange(0, h, 1)
x, y = np.meshgrid(x,y)
z = imgd
surf = ax.plot_surface(x, y, z, cmap=cm.coolwarm)

#自定义z轴
ax.set_zlim(-10, 255)
ax.zaxis.set_major_locator(LinearLocator(10))    #设置z轴网格线的疏密
#将z的value字符串转为float并保留2位小数
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

#  设置坐标轴的label和标题
ax.set_xlabel('x', size=15)
ax.set_ylabel('y', size=15)
ax.set_zlabel('z', size=15)
ax.set_title("surface plot", weight='bold', size=20)

#添加右侧的色卡条
fig.colorbar(surf, shrink=0.6, aspect=8)
plt.show()
```

效果图如下所示，对应的灰度更集中于10至100区间，由此证明了不均匀的背景被大致消除了，有利于后续的阈值分割或图像分割。



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家人，也希望读者与我一起加油。

（By：Eastmount 2019-03-21 下午6点 https://blog.csdn.net/Eastmount/）