[Python图像处理] 四十三.Python图像形态学处理万字详解（腐蚀膨胀、开闭运算、梯度顶帽黑帽运算）

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~

这篇文章是图像处理的最后一篇文章，后面我们将进入新的章节。主要包括图像识别、目标检测、图像分类、基于深度学习的图像处理等，感谢您一如既往的支持。

数学形态学是一门建立在格论和拓扑学基础之上的图像分析学科，是数学形态学图像处理的基本理论。其基本的运算包括：

- 腐蚀和膨胀
- 开运算和闭运算
- 图像顶帽运算和图像底帽运算
- 骨架抽取
- 形态学梯度
- Top-hat变换

万字长文整理，希望对您有所帮助。同时，该部分知识均为作者查阅资料撰写总结，并且开设成了收费专栏，为小宝赚点奶粉钱，感谢您的抬爱。如果有问题随时私聊我，只望您能从这个系列中学到知识，一起加油。代码下载地址（如果喜欢记得star，一定喔）：

- https://github.com/eastmountyxz/ImageProcessing-Python

文章目录

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
- [Python图像处理] 十六.图像的灰度非线性变换之对数变换、伽马变换
- [Python图像处理] 十七.图像锐化与边缘检测之Roberts算子、Prewitt算子、Sobel算子和Laplacian算子
- [Python图像处理] 十八.图像锐化与边缘检测之Scharr算子、Canny算子和LOG算子
- [Python图像处理] 十九.图像分割之基于K-Means聚类的区域分割
- [Python图像处理] 二十.图像量化处理和采样处理及局部马赛克特效
- [Python图像处理] 二十一.图像金字塔之图像向下取样和向上取样
- [Python图像处理] 二十二.Python图像傅里叶变换原理及实现
- [Python图像处理] 二十三.傅里叶变换之高通滤波和低通滤波
- [Python图像处理] 二十四.图像特效处理之毛玻璃、浮雕和油漆特效
- [Python图像处理] 二十五.图像特效处理之素描、怀旧、光照、流年以及滤镜特效
- [Python图像处理] 二十六.图像分类原理及基于KNN、朴素贝叶斯算法的图像分类案例
- [Python图像处理] 二十七.OpenGL入门及绘制基本图形（一）
- [Python图像处理] 二十八.OpenCV快速实现人脸检测及视频中的人脸
- [Python图像处理] 二十九.MoviePy视频编辑库实现抖音短视频剪切合并操作
- [Python图像处理] 三十.图像量化及采样处理万字详细总结（推荐）
- [Python图像处理] 三十一.图像点运算处理两万字详细总结（灰度化处理、阈值化处理）
- [Python图像处理] 三十二.傅里叶变换（图像去噪）与霍夫变换（特征识别）万字详细总结
- [Python图像处理] 三十三.图像各种特效处理及原理万字详解（毛玻璃、浮雕、素描、怀旧、流年、滤镜等）
- [Python图像处理] 三十四.数字图像处理基础与几何图形绘制万字详解（推荐）
- [Python图像处理] 三十五.OpenCV图像处理入门、算数逻辑运算与图像融合（推荐）
- [Python图像处理] 三十六.OpenCV图像几何变换万字详解（平移缩放旋转、镜像仿射透视）
- [Python图像处理] 三十七.OpenCV和Matplotlib绘制直方图万字详解（掩膜直方图、H-S直方图、黑夜白天判断）
- [Python图像处理] 三十八.OpenCV图像增强万字详解（直方图均衡化、局部直方图均衡化、自动色彩均衡化）
- [Python图像处理] 三十九.Python图像分类万字详解（贝叶斯图像分类、KNN图像分类、DNN图像分类）
- [Python图像处理] 四十.全网首发Python图像分割万字详解（阈值分割、边缘分割、纹理分割、分水岭算法、K-Means分割、漫水填充分割、区域定位）
- [Python图像处理] 四十一.Python图像平滑万字详解（均值滤波、方框滤波、高斯滤波、中值滤波、双边滤波）

- [Python图像处理] 四十二.Python图像锐化及边缘检测万字详解（Roberts、Prewitt、Sobel、Laplacian、Canny、LOG）
- [Python图像处理] 四十三.Python图像形态学处理万字详解（腐蚀膨胀、开闭运算、梯度顶帽黑帽运算）

一.数学形态学概述

数学形态学（Mathematical morphology） 是一种应用于图像处理和模式识别领域的新方法。数学形态学（也称图像代数）表示以形态为基础对图像进行分析的数学工具，其基本思想是用具有一定形态的结构元素去量度和提取图像中对应形状以达到对图像分析和识别的目的。数学形态学的应用可以简化图像数据，保持它们基本的形状特征，并出去不相干的结构。数学形态学的算法有天然的并行实现的结构，主要针对的是二值图像（0或1）。在图像处理方面，二值形态学经常应用到对图像进行分割、细化、抽取骨架、边缘提取、形状分析、角点检测，分水岭算法等。由于其算法简单，算法能够并行运算所以经常应用到硬件中。

常见的图像形态学运算包括腐蚀、膨胀、开运算、闭运算、梯度运算、顶帽运算和底帽运算等。主要通过MorphologyEx()函数实现，它能利用基本的膨胀和腐蚀技术，来执行更加高级形态学变换，如开闭运算、形态学梯度、顶帽、黑帽等，也可以实现最基本的图像膨胀和腐蚀。其函数原型如下：

- **dst = cv2.morphologyEx(src, model, kernel)**
  – src表示原始图像
  – model表示图像进行形态学处理，包括：
  (1) cv2.MORPH_OPEN：开运算（Opening Operation）
  (2) cv2.MORPH_CLOSE：闭运算（Closing Operation）
  (3) cv2.MORPH_GRADIENT：形态学梯度（Morphological Gradient）
  (4) cv2.MORPH_TOPHAT：顶帽运算（Top Hat）
  (5) cv2.MORPH_BLACKHAT：黑帽运算（Black Hat）
  – kernel表示卷积核，可以用numpy.ones()函数构建

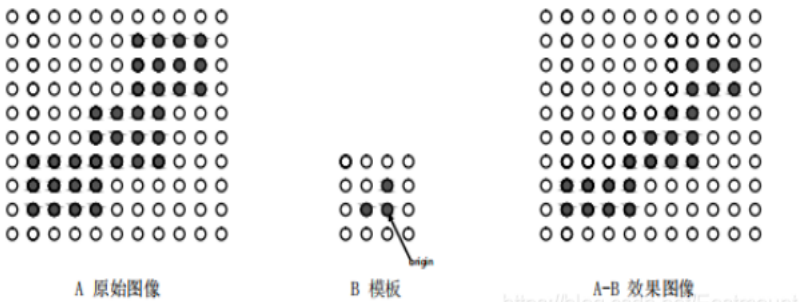
二.图像腐蚀

图像的腐蚀（Erosion）和膨胀（Dilation）是两种基本的形态学运算，主要用来寻找图像中的极小区域和极大区域。图像腐蚀类似于“领域被蚕食”，它将图像中的高亮区域或白色部分进行缩减细化，其运行结果比原图的高亮区域更小。

设A，B为集合，A被B的腐蚀，记为A - B，其定义为：

$$A - B = \{x \mid B_x \subseteq A\}$$

该公式表示图像A用卷积模板B来进行腐蚀处理，通过模板B与图像A进行卷积计算，得出B覆盖区域的像素点最小值，并用这个最小值来替代参考点的像素值。如图1所示，将左边的原始图像A腐蚀处理为右边的效果图A-B。



图像腐蚀主要包括二值图像和卷积核两个输入对象，卷积核是腐蚀中的关键数组，采用Numpy库可以生成。卷积核的中心点逐个像素扫描原始图像，被扫描到的原始图像中的像素点，只有当卷积核对应的元素值均为1时，其值才为1，否则将其像素值修改为0。

在Python中，主要调用OpenCV的erode()函数实现图像腐蚀。其函数原型如下：

- **dst = cv2.erode(src, kernel, iterations)**
  – src表示原始图像
  – kernel表示卷积核
  – iterations表示迭代次数，默认值为1，表示进行一次腐蚀操作

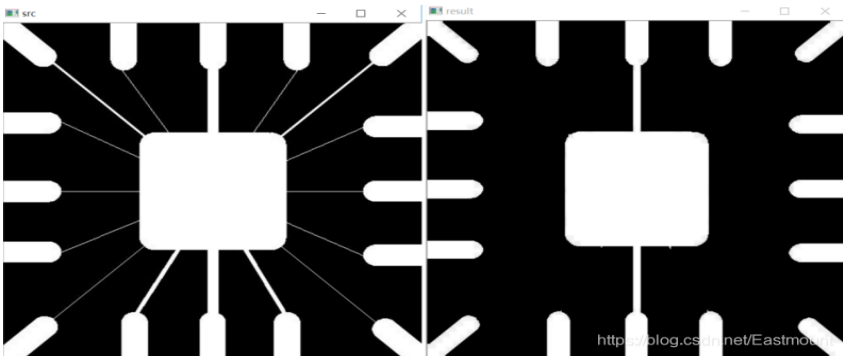
可以采用函数numpy.ones((5,5), numpy.uint8)创建5×5的卷积核，如下：

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

图像腐蚀操作的代码如下所示：

```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test01.jpg', cv2.IMREAD_UNCHANGED)

#设置卷积核
kernel = np.ones((5,5), np.uint8)

#图像腐蚀处理
erosion = cv2.erode(src, kernel, iterations=9)

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

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

输出结果如图2所示，左边表示原图，右边是腐蚀处理后的图像，可以发现图像中的干扰细线（噪声）被清洗干净。



如果腐蚀之后的图像仍然存在噪声，可以设置迭代次数进行多次腐蚀操作。比如进行9次腐蚀操作的核心代码如下：

- **erosion = cv2.erode(src, kernel,iterations=9)**

最终经过9次腐蚀处理的输出图像如图3所示。

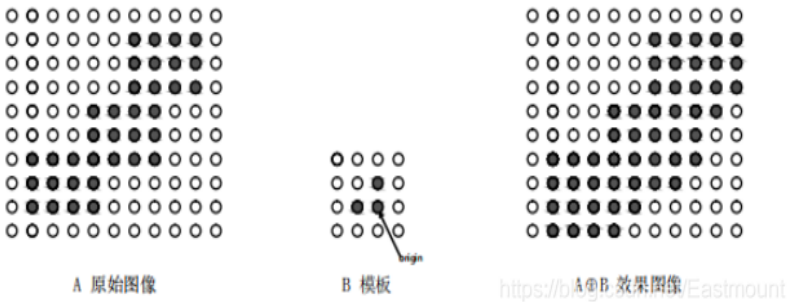
三.图像膨胀

图像膨胀是腐蚀操作的逆操作，类似于“领域扩张”，它将图像中的高亮区域或白色部分进行扩张，其运行结果比原图的高亮区域更大。

设A，B为集合，∅为空集，A被B的膨胀，记为A⊕B，其中⊕为膨胀算子，膨胀定义为：

$$A \oplus B = \{x \mid B_x \cap A \neq \phi\}$$

该公式表示用B来对图像A进行膨胀处理，其中B是一个卷积模板，其形状可以为正方形或圆形，通过模板B与图像A进行卷积计算，扫描图像中的每一个像素点，用模板元素与二值图像元素做“与”运算，如果都为0，那么目标像素点为0，否则为1。从而计算B覆盖区域的像素点最大值，并用该值替换参考点的像素值实现图像膨胀。图4是将左边的原始图像A膨胀处理为右边的效果图A⊕B。



图像被腐蚀处理后，它将去除噪声，但同时会压缩图像，而图像膨胀操作可以去除噪声并保持原有形状，如图5所示。

在Python中，主要调用OpenCV的dilate()函数实现图像腐蚀。其函数原型如下：

- **dst = cv2.dilate(src, kernel, iterations)**
  – src表示原始图像
  – kernel表示卷积核，可以用numpy.ones()函数构建

– iterations表示迭代次数，默认值为1，表示进行一次膨胀操作

图像膨胀操作的代码如下所示：

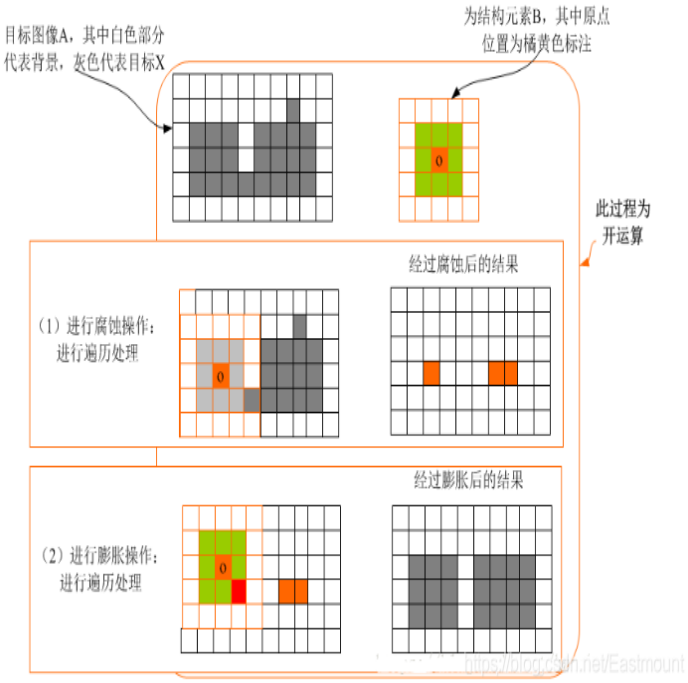
```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test02.png', cv2.IMREAD_UNCHANGED)

#设置卷积核
kernel = np.ones((5,5), np.uint8)

#图像膨胀处理
erosion = cv2.dilate(src, kernel)

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

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

输出结果如图6所示：

四.图像开运算

开运算一般能平滑图像的轮廓，削弱狭窄部分，去掉较细的突出。闭运算也是平滑图像的轮廓，与开运算相反，它一般熔合窄的缺口和细长的弯口，去掉小洞，填补轮廓上的缝隙。

图像开运算是图像依次经过腐蚀、膨胀处理的过程，图像被腐蚀后将去除噪声，但同时也压缩了图像，接着对腐蚀过的图像进行膨胀处理，可以在保留原有图像的基础上去除噪声。其原理如图7所示。



设A是原始图像，B是结构元素图像，则集合A被结构元素B做开运算，记为A∘B，其定义为：

$$\mathrm{A} \circ \mathrm{B}=(A-B) \oplus \mathrm{B}$$

换句话说，A被B开运算就是A被B腐蚀后的结果再被B膨胀。

图像开运算在OpenCV中主要使用函数morphologyEx()，它是形态学扩展的一组函数，其函数原型如下：

- **dst = cv2.morphologyEx(src, cv2.MORPH_OPEN, kernel)**
  – src表示原始图像
  – cv2.MORPH_OPEN表示图像进行开运算处理
  – kernel表示卷积核，可以用numpy.ones()函数构建

图像开运算的代码如下所示：

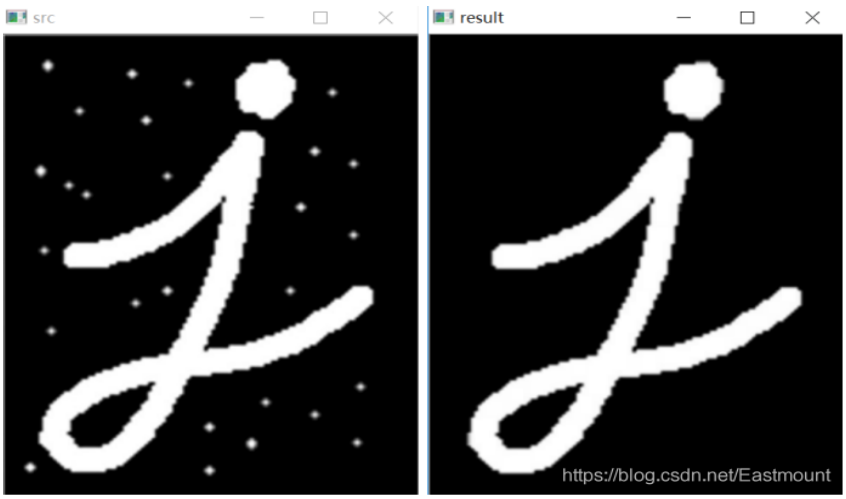
```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test01.png', cv2.IMREAD_UNCHANGED)

#设置卷积核
kernel = np.ones((5,5), np.uint8)

#图像开运算
result = cv2.morphologyEx(src, cv2.MORPH_OPEN, kernel)

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

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

输出结果如图8所示，左边为原始图像，右边为处理后的图像，可以看到原始图形中的噪声点被去除了部分。

但处理后的图像中仍然有部分噪声，如果想更彻底地去除，可以将卷积设置为10×10的模板，代码如下所示：

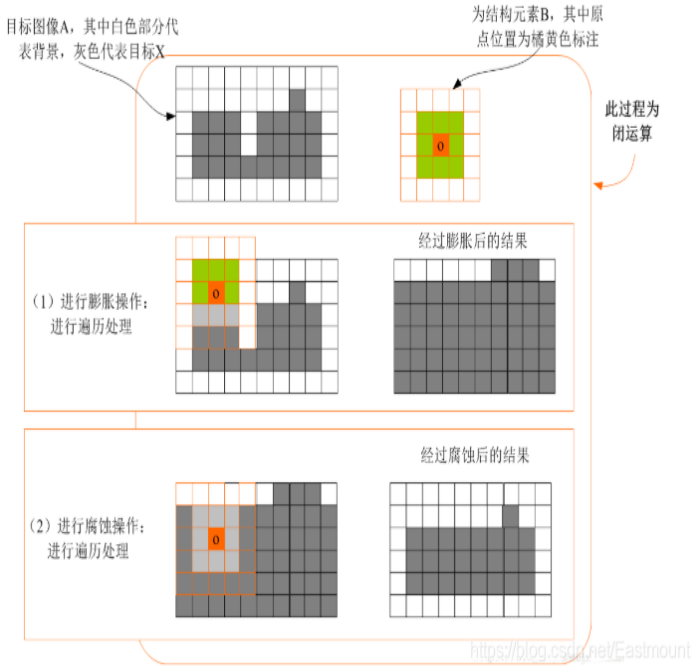
```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test01.png', cv2.IMREAD_UNCHANGED)

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

#图像开运算
result = cv2.morphologyEx(src, cv2.MORPH_OPEN, kernel)

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

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

运行结果如图9所示：



五.图像闭运算

图像闭运算是图像依次经过膨胀、腐蚀处理的过程，先膨胀后腐蚀有助于过滤前景物体内部的小孔或物体上的小黑点。其原理如图10所示：



设A是原始图像，B是结构元素图像，则集合A被结构元素B做开运算，记为A·B，其定义为：

$$A \bullet B = (A \oplus B) - B$$

换句话说，A被B闭运算就是A被B膨胀后的结果再被B腐蚀。

图像开运算在OpenCV中主要使用函数morphologyEx()，其函数原型如下：

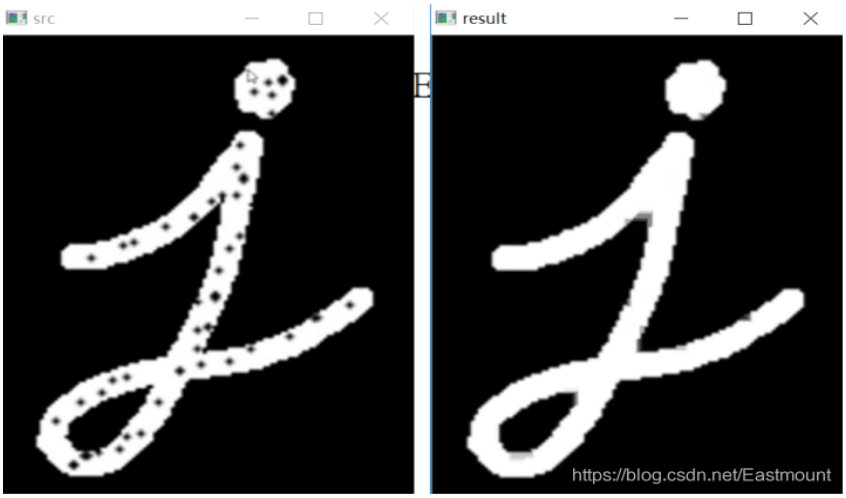
- **dst = cv2.morphologyEx(src, cv2.MORPH_CLOSE, kernel)**
  – src表示原始图像
  – cv2.MORPH_CLOSE表示图像进行闭运算处理
  – kernel表示卷积核，可以用numpy.ones()函数构建

图像开运算的代码如下所示：

```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test03.png', cv2.IMREAD_UNCHANGED)

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

#图像闭运算
result = cv2.morphologyEx(src, cv2.MORPH_CLOSE, kernel)

#显示图像
cv2.imshow("src", src)
```

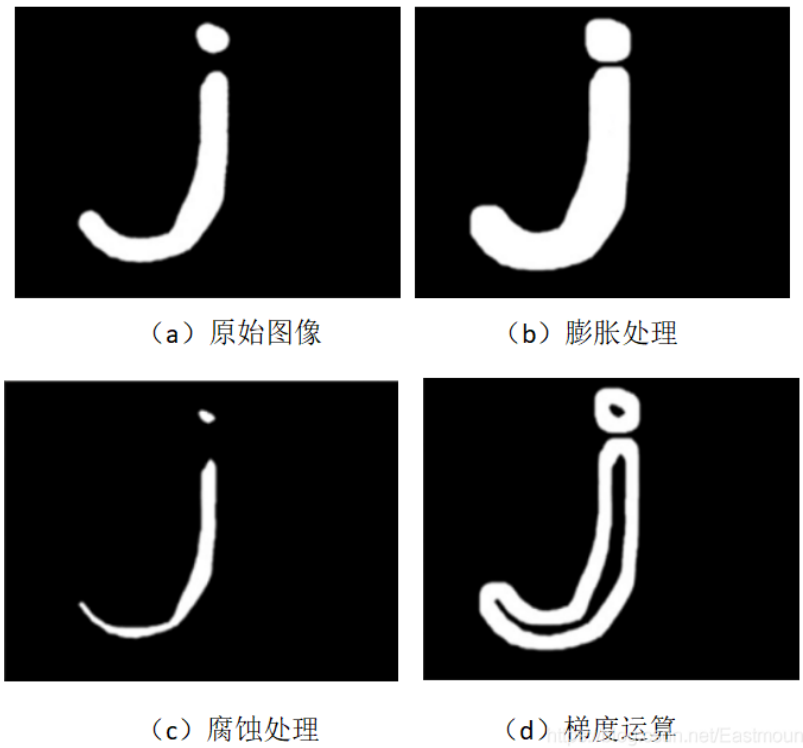
```
    cv2.imshow("result", result)

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

输出结果如图11所示，它有效地去除了图像中间的小黑点（噪声）。



六.图像梯度运算

图像梯度运算是图像膨胀处理减去图像腐蚀处理后的结果，从而得到图像的轮廓，其原理如图12所示，（a）表示原始图像，（b）表示膨胀处理后的图像，（c）表示腐蚀处理后的图像，（d）表示图像梯度运算的效果图。



在Python中，图像梯度运算主要调用morphologyEx()实现，其中参数cv2.MORPH_GRADIENT表示梯度处理，函数原型如下：

- **dst = cv2.morphologyEx(src, cv2.MORPH_GRADIENT, kernel)**

– src表示原始图像
– cv2.MORPH_GRADIENT表示图像进行梯度运算处理
– kernel表示卷积核，可以用numpy.ones()函数构建

图像梯度运算的实现代码详见如下。

```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np

#读取图片
src = cv2.imread('test04.png', cv2.IMREAD_UNCHANGED)

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

#图像梯度运算
result = cv2.morphologyEx(src, cv2.MORPH_GRADIENT, kernel)

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

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

图像梯度运算处理的结果如图13所示，左边为原始图像，右边为处理后的效果图。

七.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，顶帽运算用于暗背景上的亮物体，它的一个重要用途是校正不均匀光照的影响。其效果图如图14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15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
#By:Eastmount CSDN 2021-07-30
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()
```

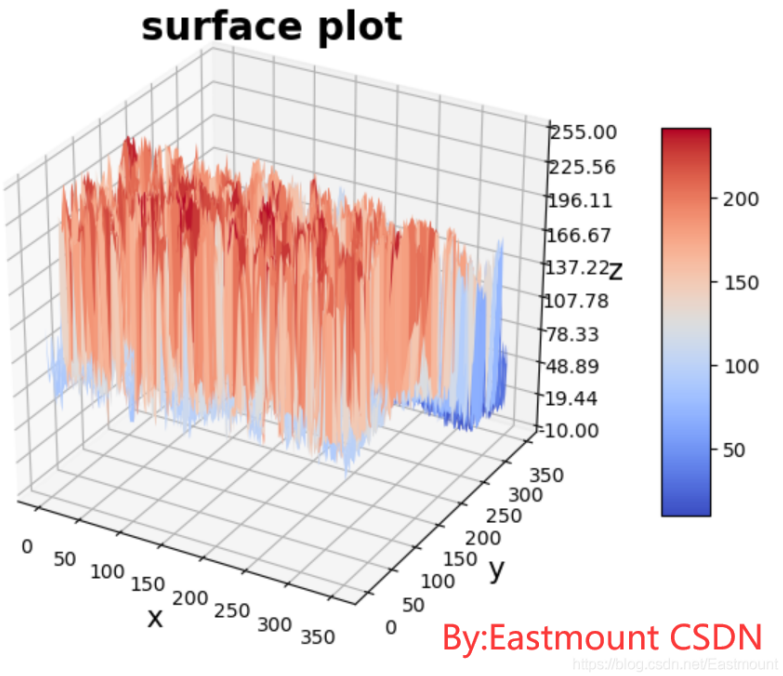
其运行结果如图16所示，可以看到顶帽运算后的图像删除了大部分非均匀背景，并将米粒与背景分离开来。

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：

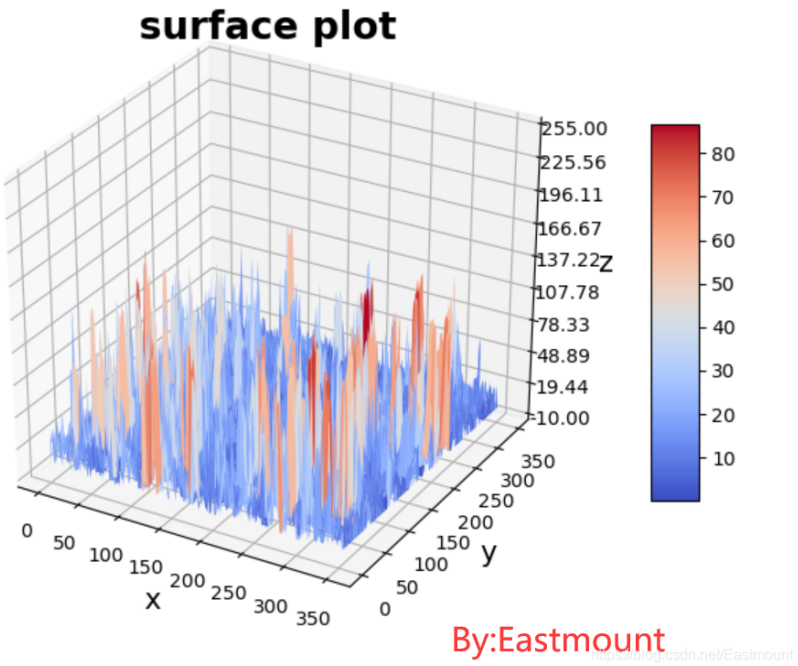
```
# -*- coding: utf-8 -*-
#By:Eastmount CSDN 2021-07-30
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

#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)   #img 原图

#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()
```

运行结果如图17所示，其中x表示原图像中的宽度坐标，y表示原图像中的高度坐标，z表示像素点(x, y)的灰度值。



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。通过图像白帽运算后的图像灰度三维图如图18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。

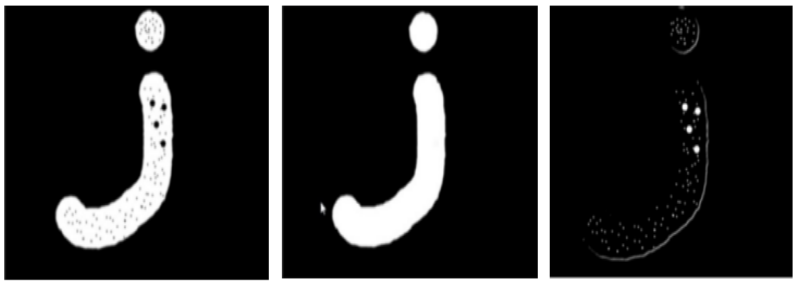


八.图像底帽运算

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：

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

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。其效果图如图19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
#By:Eastmount CSDN 2021-07-30
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20所示：

九.总结

本文主要讲解了图像数学形态学知识，结合原理和代码详细介绍了图像腐蚀、图像膨胀、图像开运算和闭运算、图像顶帽运算和图像底帽

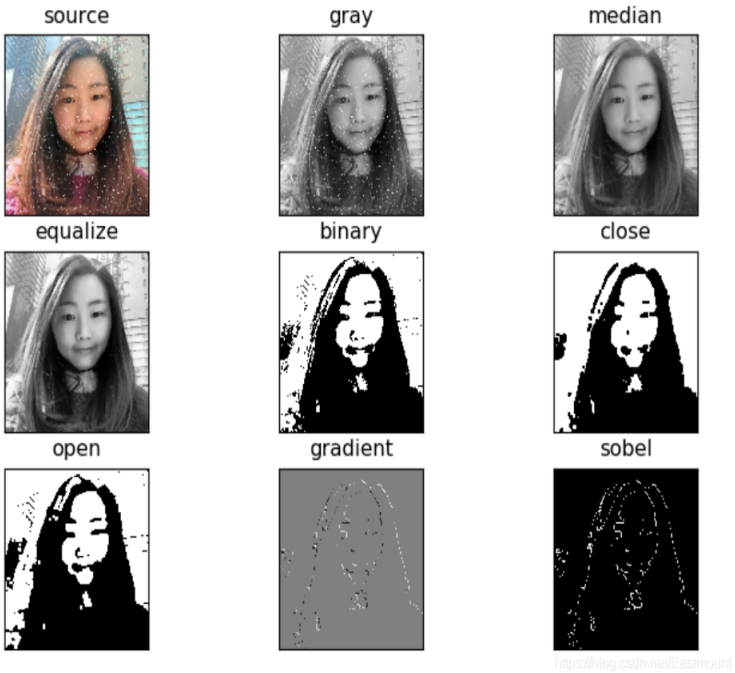
运算等操作，为后续的图像分割和图像识别提供有效支撑。

完整代码：

```
#encoding:utf-8
#By:Eastmount CSDN 2021-07-30
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
src = cv2.imread('test01_yn.png', cv2.IMREAD_UNCHANGED)
img = cv2.cvtColor(src,cv2.COLOR_BGR2RGB)

# 转化为灰度图
Grayimg = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

# 1、消除椒盐噪声:
# 中值滤波器
median = cv2.medianBlur(Grayimg, 5)
# 消除噪声图
cv2.imshow("median-image", median)

# 2、直方图均衡化:
equalize = cv2.equalizeHist(median)
cv2.imshow('hist', equalize)

# 3、二值化处理:
# 阈值为140
ret, binary = cv2.threshold(equalize, 127, 255,cv2.THRESH_BINARY)
cv2.imshow("binary-image",binary)
cv2.waitKey(0)

#设置卷积核
kernel = np.ones((10,10), np.uint8)
close = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)

#图像开运算
kernel = np.ones((10,10), np.uint8)
open1 = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

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

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

#图像开运算
kernel = np.ones((10,10), np.uint8)
gradient = cv2.morphologyEx(binary, cv2.MORPH_GRADIENT, kernel)

# Sobel算子 XY方向求梯度 cv2.CV_8U
x = cv2.Sobel(close, cv2.CV_32F, 1, 0, ksize = 3) #X方向
y = cv2.Sobel(close, cv2.CV_32F, 0, 1, ksize = 3) #Y方向
#absX = cv2.convertScaleAbs(x)   # 转回uint8
#absY = cv2.convertScaleAbs(y)
#Sobel = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)
gradient = cv2.subtract(x, y)
sobel = cv2.convertScaleAbs(gradient)
cv2.imshow('Sobel', sobel)
```

```
    cv2.waitKey(0)

    #循环显示图形
    titles = [ 'source', 'gray', 'median', 'equalize', 'binary', 'close', 'open', 'gradient', 'sobel']
    images = [img, Grayimg, median, equalize, binary, close, open1, gradient, sobel]
    for i in range(9):
       plt.subplot(3, 3, i+1), plt.imshow(images[i], 'gray')
       plt.title(titles[i])
       plt.xticks([]),plt.yticks([])
    plt.show()
```

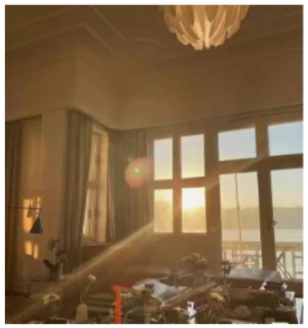
输出结果如下图所示：



源代码下载地址，记得帮忙点star和关注喔。

- https://github.com/eastmountyxz/ImageProcessing-Python

> 大学之道在明明德，
> 在亲民，在止于至善。
> 这周又回答了很多博友的问题，有大一学生的困惑，有论文的咨询，也有老乡和考博的疑问，还有无数博友奋斗路上的相互勉励。虽然自己早已忙成狗，但总忍不住去解答别人的问题。最后那一句感谢和祝福，永远是我最大的满足。虽然会花费我一些时间，但也挺好的，无所谓了，跟着心走。不负遇见，感恩同行。莫愁前路无知己，继续加油。晚安娜和珞。

> 很幸运在我的大学生活中，可以遇到这样一些带来光的人，真的很感谢也很幸运，可以以他们为榜样去学习，在我迷茫时我可以从中得到启发和帮助，接下来的日子，去追随光，靠近光，然后希望自己可以成为一个带来光的人
>
> 
>
>

参考文献：

- [1] 冈萨雷斯著，阮秋琦译. 数字图像处理（第3版）[M]. 北京：电子工业出版社，2013.
- [2] 阮秋琦. 数字图像处理学（第3版）[M]. 北京：电子工业出版社，2008.