OpenCV-python 学习笔记 OpenCV形态学转换

原始图像：

相关函数：

`cv2.erode()`

`cv2.dilate()`

`cv2.morphotogyEX()`

形态学转换原理：

一般情况下对二值化图像进行操作。需要两个参数，一个是原始图像，第二个被称为结构化元素或者核，它是用来决定操作的性质的。基本操作为腐蚀和膨胀，他们的变体构成了开运算，闭运算，梯度等。

1. 腐蚀

`erosion = cv2.erode(img,kernel,iterations=1)`

把前景物体的边界腐蚀掉，但是前景仍然是白色的。卷积核沿着图像滑动，如果与卷积核对应的原图像的所有像素值都是1，那么中心元素就保持原来的像素值，否则就变为零。根据卷积核的大小靠近前景的所有像素都会被腐蚀掉（变为0），所以前景物体会变小，整幅图像的白色区域会减少。这对于去除白噪音很有用，也可以用来断开两个连在一块的物体。

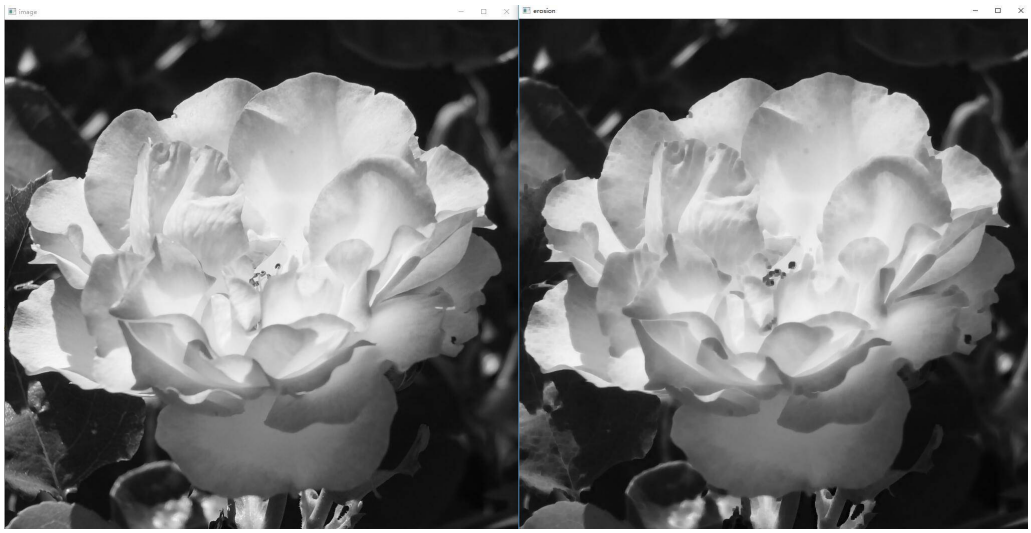
例如：

```
1 import cv2
```

```
import numpy as np

img = cv2.imread('1024.jpg',0)
kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(img,kernel,iterations=1)

while(1):
    cv2.imshow('image',img)
    cv2.imshow('erosion',erosion)
    k=cv2.waitKey(1)
    if k == ord('q'):#按q键退出
        break
cv2.destroyAllWindows()
```



2. 膨胀

`dilation = cv2.dilation(img,kernel,iterations=1)`

与腐蚀相反，与卷积核对应的原图像的像素值中只要有一个是1，中心元素的像素值就是1。所以这个操作会增加图像中白色区域（前景）。一般在去噪音时先腐蚀再膨胀，因为腐蚀再去掉白噪音的同时，也会使前景对象变小，所以我们再膨胀。这时噪音已经被去除，不会再回来了，但是前景还在并会增加，膨胀也可以用来连接两个分开的物体。

3. 开运算

`cv2.MORPH_OPEN`

先进行**腐蚀**再进行**膨胀**就叫做开运算。被用来去除噪音，函数可以使用`cv2.morphotogyEx()`

```
opening = cv2.morphotogyEx(img,cv2.MORPH_OPEN,kernel)
```

4. 闭运算

`cv2.MORPH_CLOSE`

先**膨胀**再**腐蚀**。被用来填充前景物体中的小洞，或者前景上的小黑点。

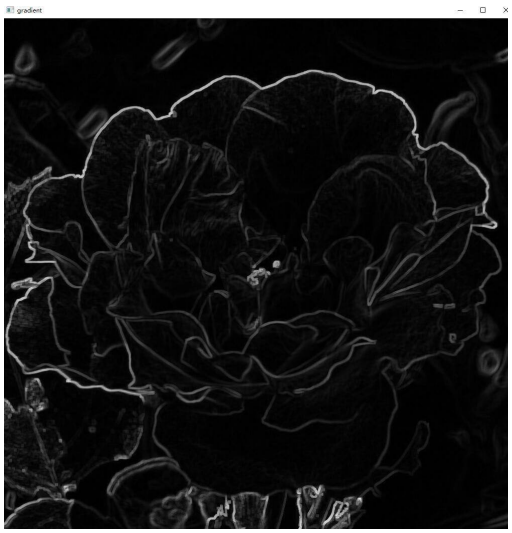
```
closing = cv2.morphotogyEx(img,cv2.MORPH_CLOSE,kernel)
```

5. 形态学梯度

`cv2.MORPH_GRADIENT`

一幅图像膨胀与腐蚀的**差别**。
结果看上去就像前景物体的轮廓。

```
gradient = cv2.morphotogyEx(img,cv2.MORPH_GRADIENT,kernel)
```



6. 礼帽

`cv2.MORPH_TOPHAT`

原始图像与进行**开运算**之后得到的图像的**差**.

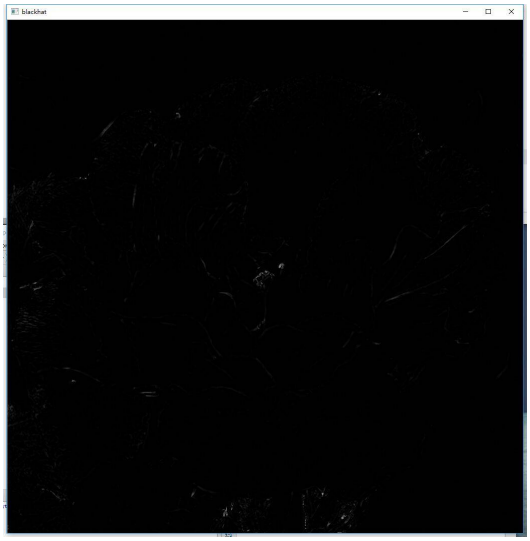
```
tophat = cv2.morphotogyEx(img,cv2.MORPH_TOPHAT,kernel)
```

7. 黑帽

`cv2.MORPH_BLACKHAT`

进行**闭运算**之后得到的图像与原始图像的**差**。

```
blackhat = cv2.morphotogyEx(img,cv2.MORPH_BLACKHAT,kernel)
```



8. 形态学操作之间关系及结构化元素

形态学操作之间的关系：

Opening:

$$dst = open(src, element) = dilate(erode(src, element), element)$$

Closing:

$$dst = close(src, element) = erode(dilate(src, element), element)$$

Morphological gradient:

$$dst = morph\_grad(src, element) = dilate(src, element) - erode(src, element)$$

“Top hat”:

$$dst = tophat(src, element) = src - open(src, element)$$

“Black hat”:

$$dst = blackhat(src, element) = close(src, element) - src$$

结构化元素：

之前的例子都是使用numpy构建了结构化元素，但是是正方形的，若需要构建椭圆或者圆形的核，可以使用OpenCV提供的函数`cv2.getStructuringElemenet()`，只需要告诉它你需要的核的形状和大小。（相当于自定义卷积核的形状核大小）

```
# Rectangular Kernel
>>> cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
array([[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]], dtype=uint8)

# Elliptical Kernel
>>> cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)

# Cross-shaped Kernel
>>> cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))
array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0]], dtype=uint8)
```