腐蚀与膨胀

电子信息工程系

袁羽

目录

CONTENTS

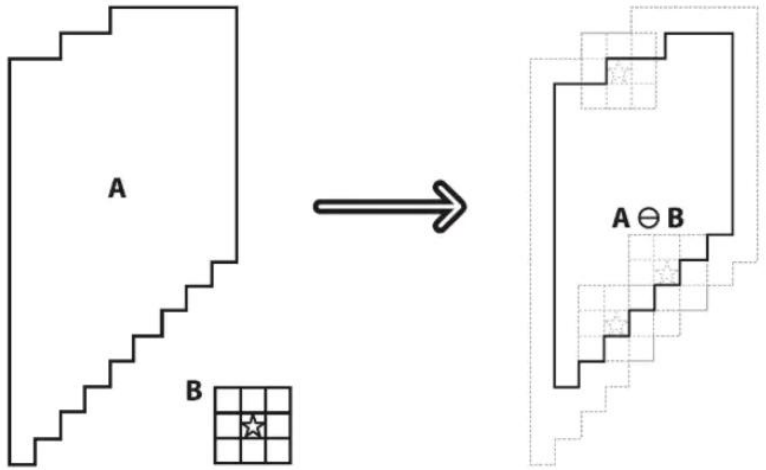
腐蚀和膨胀是图像形态学中的两种核心操作，通过这两种操作可以清除或强化图像中的细节。主要用于从图像中提取对于表达和描述区域形状有用的图像分量（特征分量），使后续的识别工作能够抓住目标对象最为本质的形状特征，如边界，骨架及凸壳。

膨胀是图像中的高亮部分进行膨胀，领域扩张，效果图拥有比原图更大的高亮区域，用来连接两个分开的物体以及修复内部空洞；腐蚀是图像中的高亮部分被腐蚀掉，领域缩减，效果图拥有比原图更小的高亮区域，用于拆分连在一起的物体；

合理使用腐蚀和膨胀，还可以实现图像开运算、闭运算、梯度运算、顶帽运算和黑帽运算等极具特点的操作。

一 腐蚀

腐蚀操作可以让图像沿着自己的边界向内收缩。OpenCV通过“核”来实现收缩计算。“核”的英文名为kernel，在形态学中可以理解为“由n个像素组成的像素块”，像素块包含一个核心（核心通常在中央位置，也可以定义在其他位置）。像素块在图像的边缘移动，在移动过程中，核会将图像边缘那些与核重合但又没有越过核心的像素点都抹除（集合运算），就像削土豆皮一样，将图像一层一层地“削薄”。



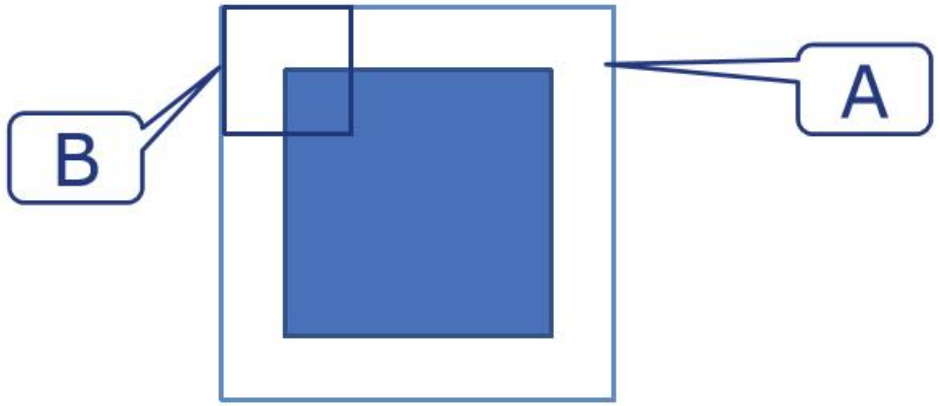
一 腐蚀

用结构元素B腐蚀A被定义为：

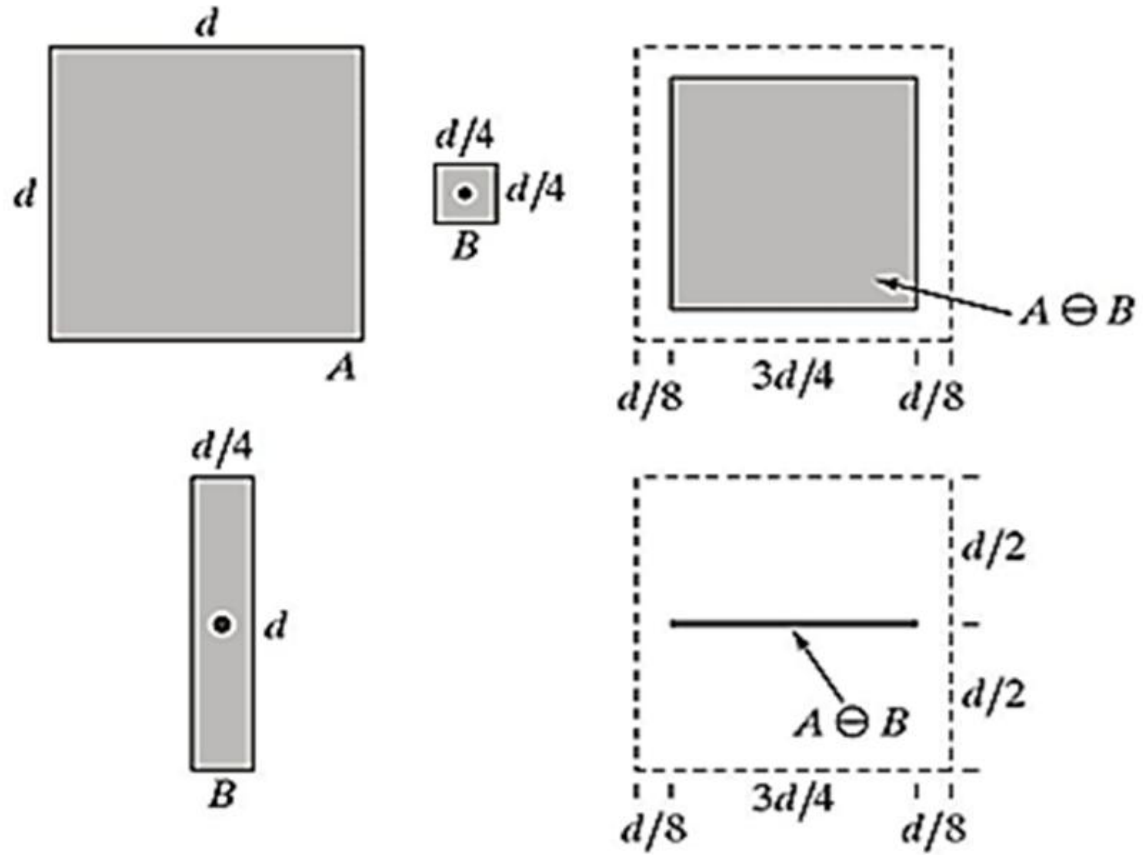
$$E = A\Theta B = \{x, y \mid B_{xy} \subseteq A\}$$

其中B_{xy}表示原点（中点）为(x，y)时的结构元素。

集合E是这样的点(x，y)组成的：B的原点位移到点(x，y)，B将完全包含于A中。



一 腐蚀



一 腐蚀

OpenCV将腐蚀操作封装成erode()方法，该方法的语法如下：

dst = cv2.erode(src, kernel, anchor, iterations, borderType, borderValue)

参数说明：

src：原始图像。

kernel：腐蚀使用的核。

anchor：可选参数，核的锚点位置。

iterations：可选参数，腐蚀操作的迭代次数，默认值为1。

borderType：可选参数，边界样式，建议默认。

borderValue：可选参数，边界值，建议默认。

返回值说明：dst：经过腐蚀之后的图像。

一 腐蚀

在OpenCV做腐蚀或其他形态学操作时，通常使用numpy模块来创建核数组，例如：

```
import numpy as np

k = np.ones((5, 5), np.uint8)
```

通过numpy模块的ones()方法创建了一个5行5列（简称5×5）、数字类型为无符号8位整数、每一个数字的值都是1的数组，这个数组作为erode()方法的核参数。除了5×5的结构，还可以使用3×3、9×9、11×11等结构，行列数越大，计算出的效果就越粗糙，行列数越小，计算出的效果就越精细。

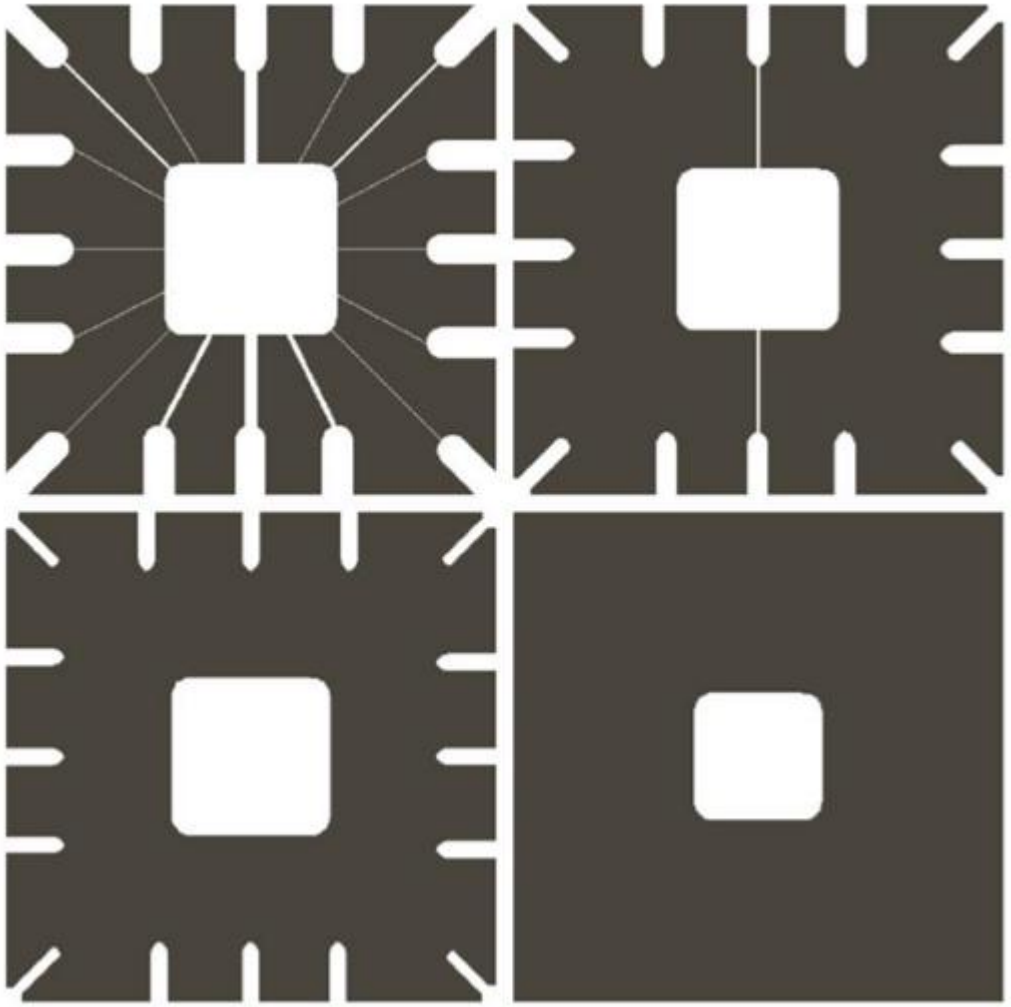
例：将仙人球图像中的刺抹除：使用3×3的核对仙人球图像进行腐蚀操作，可以将图像里的刺抹除。

例：将仙人球图像中的刺抹除：使用3×3的核对仙人球图像进行腐蚀操作，可以将图像里的刺抹除。

```
import cv2
import numpy as np
img = cv2.imread("cactus.jpg") # 读取原图
k = np.ones((3, 3), np.uint8) # 创建3*3的数组作为核
cv2.imshow("img", img) # 显示原图
dst = cv2.erode(img, k) # 腐蚀操作
cv2.imshow("dst", dst) # 显示腐蚀效果
cv2.waitKey() # 按下任何键盘按键后
cv2.destroyAllWindows() # 释放所有窗体。
```

例：比较不同大小的正方形核结构元素腐蚀的效果。

原图像



11*11

15*15

45*45

例：比较不同大小的正方形核结构元素腐蚀的效果。

```
import cv2
import numpy as np
img = cv2.imread("fs.png")  # 读取原图
cv2.imshow("img", img)  # 显示原图
k1 = np.ones((11, 11), np.uint8)  # 创建11*11的数组作为核
dst1 = cv2.erode(img, k1)  # 腐蚀操作
k2 = np.ones((15, 15), np.uint8)  # 创建15*15的数组作为核
dst2 = cv2.erode(img, k2)  # 腐蚀操作
k3 = np.ones((45, 45), np.uint8)  # 创建45*45的数组作为核
dst3 = cv2.erode(img, k3)  # 腐蚀操作
cv2.imshow("dst1", dst1)  # 显示腐蚀效果
cv2.imshow("dst2", dst2)  # 显示腐蚀效果
cv2.imshow("dst3", dst3)  # 显示腐蚀效果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体。
```

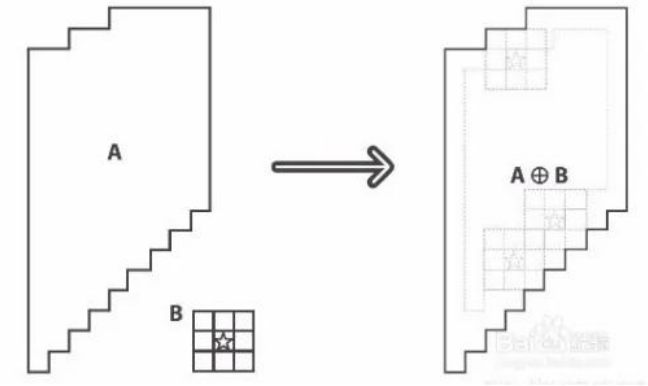
一 腐蚀

腐蚀是图像中的高亮部分被腐蚀掉，领域缩减，效果图拥有比原图更小的高亮区域，用于拆分连在一起的物体。

- 腐蚀能将连接的对象分开
- 腐蚀能去除喷溅突出
- 腐蚀会缩小图像

二 膨胀

膨胀操作与腐蚀操作相反，膨胀操作可以让图像沿着自己的边界向内扩张。同样是通过核来计算，当核在图像的边缘移动时，核会将图像边缘填补新的像素，就像在一面墙上反反复复地涂水泥，让墙变得越来越厚。

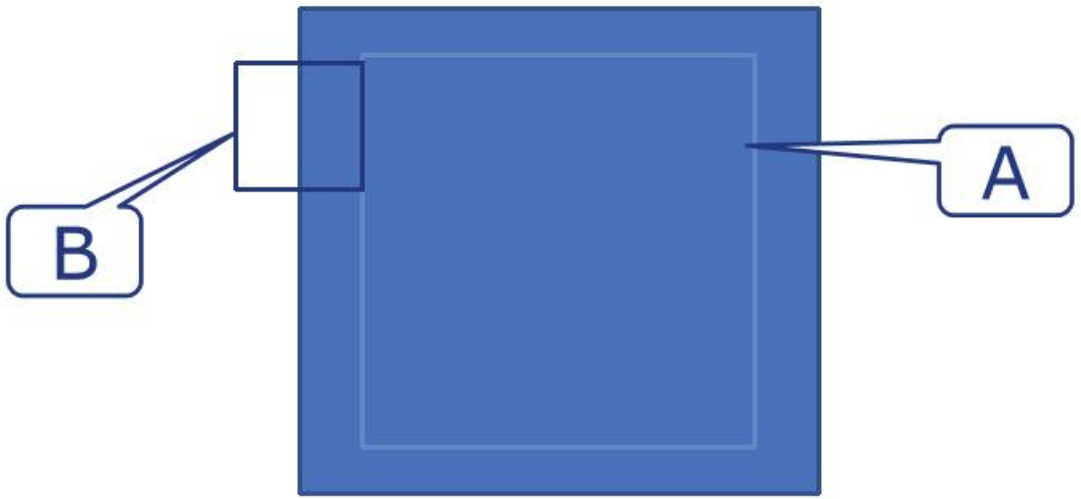


二 膨胀

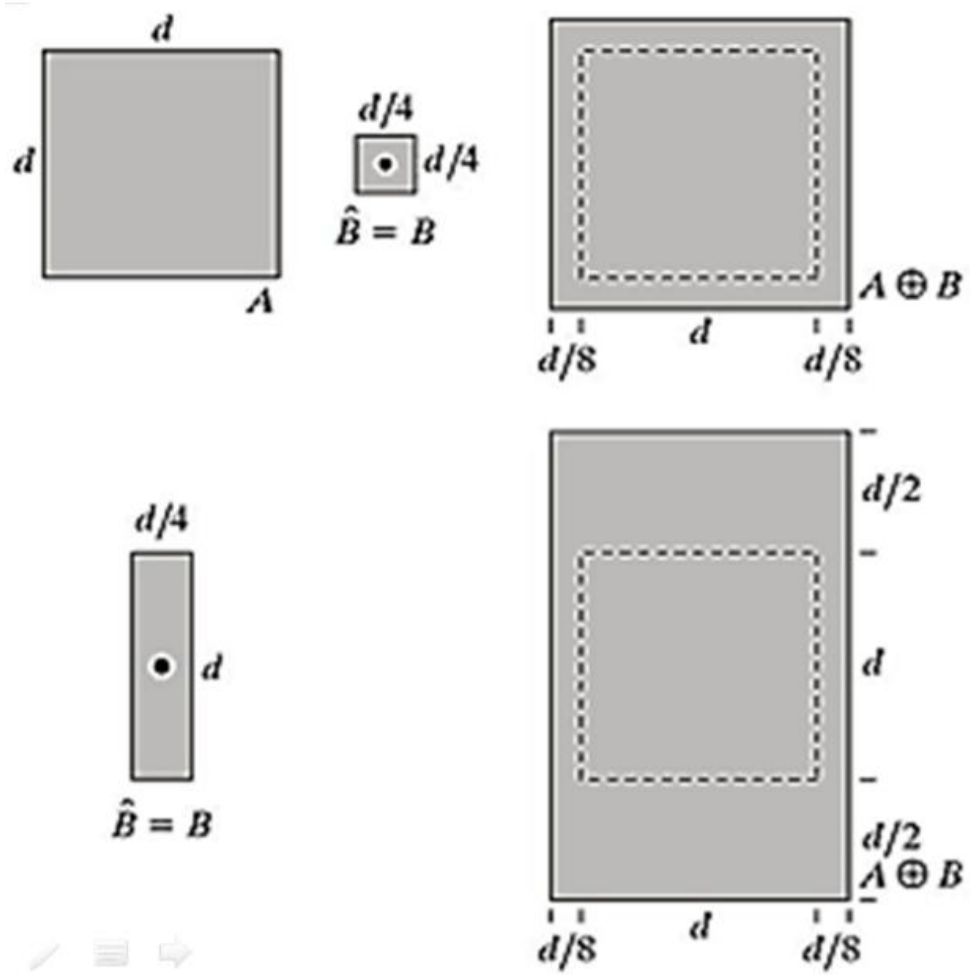
用结构元素B膨胀A被定义为：

$$D = A \oplus B = \{x, y \mid \hat{B}_{xy} \cap A \neq \varnothing\}$$

B对A膨胀产生的二值图像D是由这样的点（x，y）组成的集合：$\hat{B}$ 的原点位移到(x，y)，它与A的交集非空。



二 膨胀



二 膨胀

OpenCV将膨胀操作封装成dilate()方法，该方法的语法如下：

dst = cv2.dilate(src, kernel, anchor, iterations, borderType, borderValue)

参数说明：

src：原始图像。

kernel：膨胀使用的核。

anchor：可选参数，核的锚点位置。

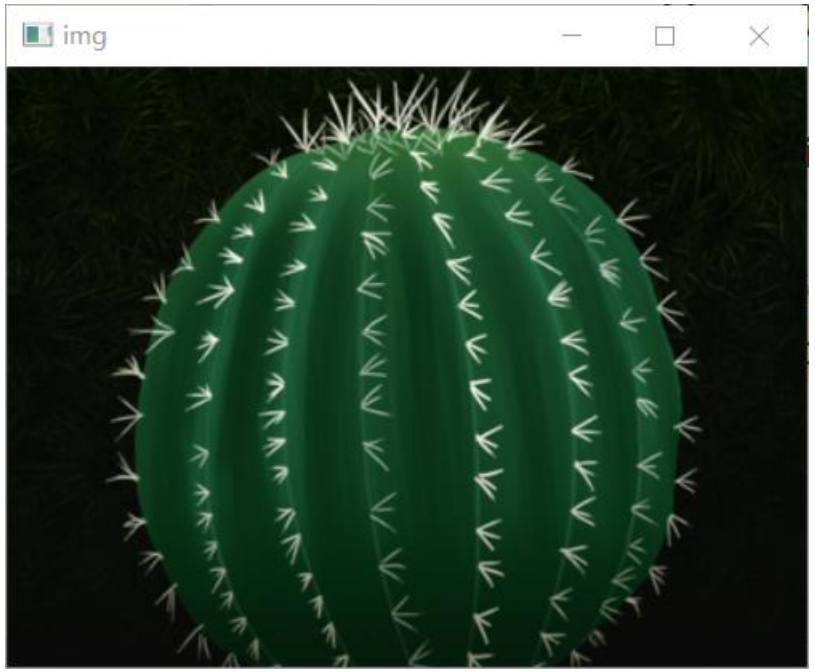
iterations：可选参数，腐蚀操作的迭代次数，默认值为1。

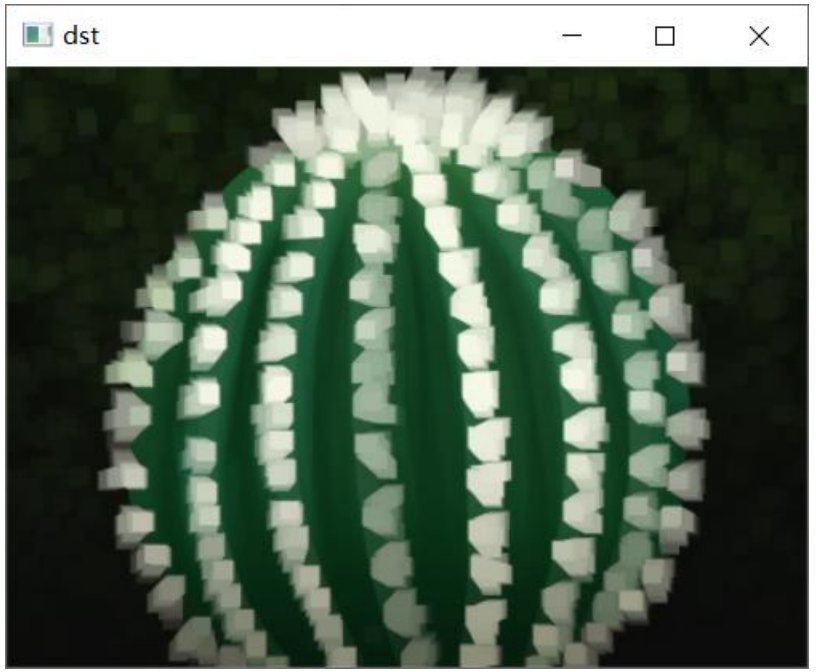
borderType：可选参数，边界样式，建议默认。

borderValue：可选参数，边界值，建议默认。

返回值说明：dst：经过膨胀之后的图像。

例：将图像加工成“近视眼”效果：利用膨胀操作可以将正常画面处理成近视眼看到的画面。

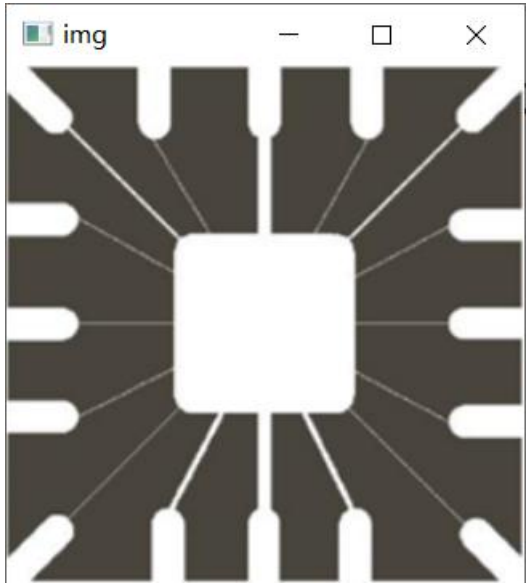


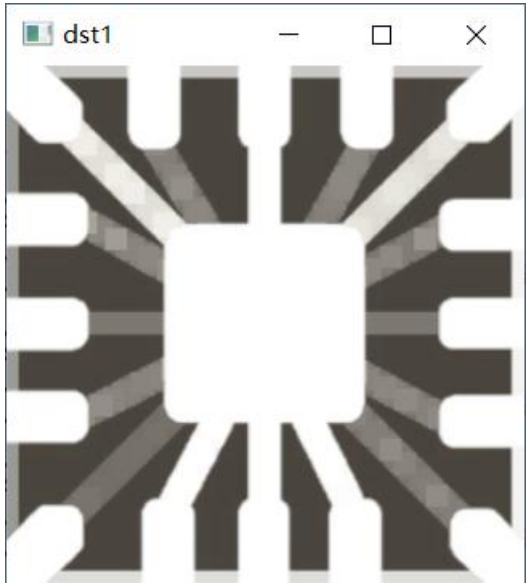


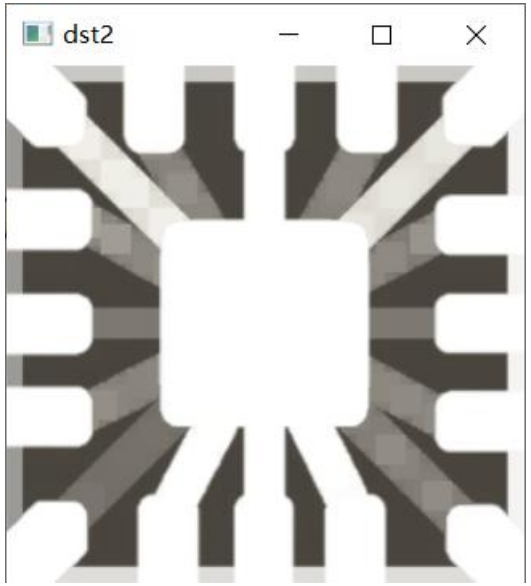
例：将图像加工成“近视眼”效果：利用膨胀操作可以将正常画面处理成近视眼看到的画面。

```
import cv2
import numpy as np
img = cv2.imread("cactus.jpg") # 读取原图
k = np.ones((9, 9), np.uint8) # 创建9*9的数组作为核
cv2.imshow("img", img) # 显示原图
dst = cv2.dilate(img, k) # 膨胀操作
cv2.imshow("dst", dst) # 显示膨胀效果
cv2.waitKey() # 按下任何键盘按键后
cv2.destroyAllWindows() # 释放所有窗体
```

例：比较不同大小的正方形核结构元素膨胀的效果。







例：比较不同大小的正方形核结构元素膨胀的效果。

```
import cv2
import numpy as np
img = cv2.imread("fs.png") # 读取原图
cv2.imshow("img", img) # 显示原图
k1 = np.ones((11, 11), np.uint8) # 创建11*11的数组作为核
dst1 = cv2.dilate(img, k1) # 膨胀操作
k2 = np.ones((15, 15), np.uint8) # 创建15*15的数组作为核
dst2 = cv2.dilate(img, k2) # 膨胀操作
k3 = np.ones((45, 45), np.uint8) # 创建45*45的数组作为核
dst3 = cv2.dilate(img, k3) # 膨胀操作
cv2.imshow("dst1", dst1) # 显示腐蚀效果
cv2.imshow("dst2", dst2) # 显示腐蚀效果
cv2.imshow("dst3", dst3) # 显示腐蚀效果
cv2.waitKey() # 按下任何键盘按键后
cv2.destroyAllWindows() # 释放所有窗体。
```

二 膨胀

图像膨胀是图像中的高亮部分进行膨胀，领域扩张，效果图拥有比原图更大的高亮区域，用来连接两个分开的物体以及修复内部空洞。

- 膨胀可修复图像断裂
- 膨胀可修复侵入突出
- 膨胀会放大

三 开运算

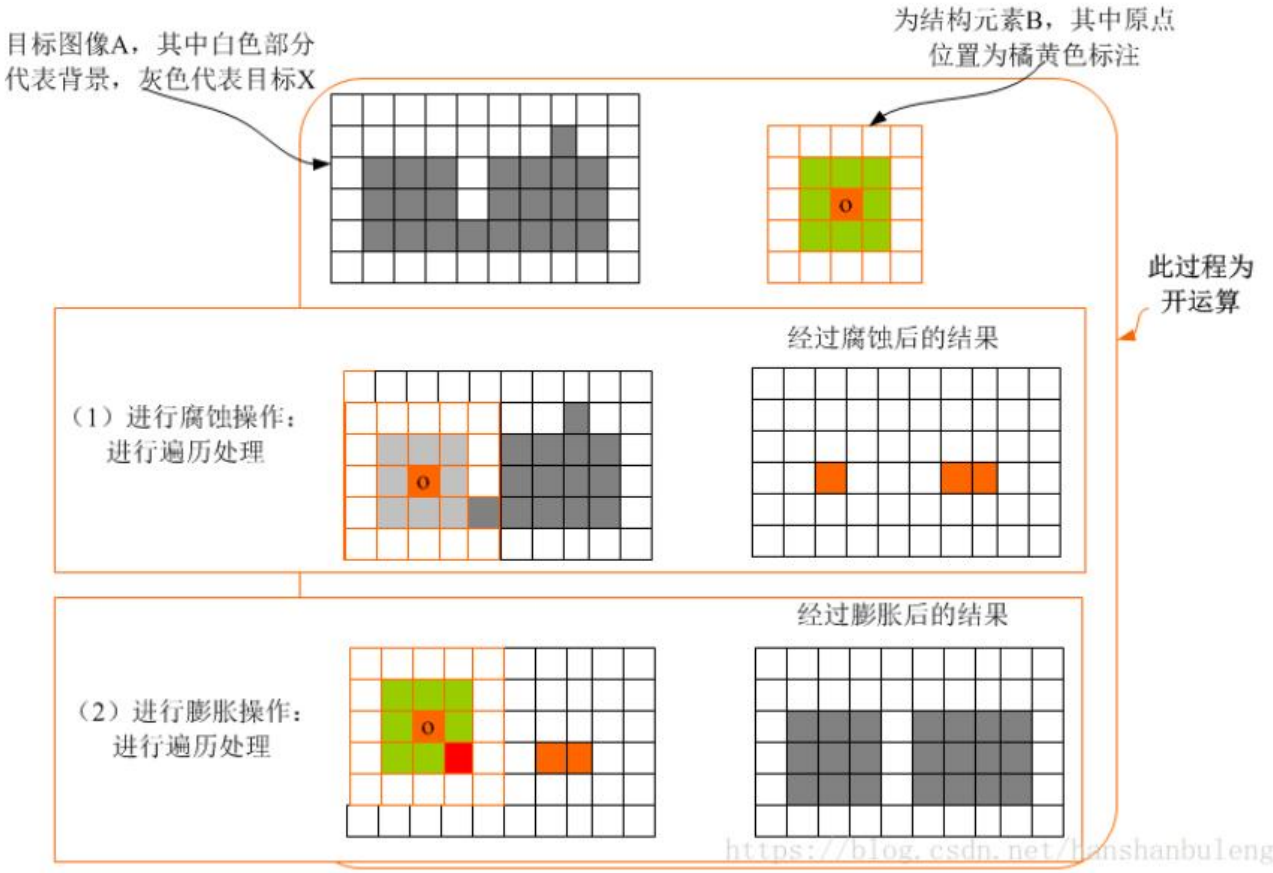
膨胀使图像变大，腐蚀使图像变小，为保持图像大小，膨胀与腐蚀通常成对出现。

开运算： 先腐蚀后膨胀的过程称为开运算。

$$A \circ B = (A \Theta B) \oplus B$$

开运算一般使对像的轮廓变得光滑，断开狭窄的间断和消除细小的突出物。

三 开运算


目标图像A，其中白色部分
代表背景，灰色代表目标X
为结构元素B，其中原点
位置为橘黄色标注
此过程为
开运算
经过腐蚀后的结果
（1）进行腐蚀操作：
进行遍历处理
经过膨胀后的结果
（2）进行膨胀操作：
进行遍历处理

例：抹除图像外部噪声：文字周围有很多噪点。要抹除这些孤立的小点，对图像进行开运算。

```
import cv2
import numpy as np
img = cv2.imread("w1.png")  # 读取原图
k = np.ones((9, 9),np.uint8)  # 创建9*9的数组作为核
cv2.imshow("img", img)  # 显示原图
dst = cv2.erode(img, k)  # 腐蚀操作
dst = cv2.dilate(dst, k)  # 膨胀操作
cv2.imshow("dst", dst)  # 显示开运算结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例：抹除图像外部噪声：文字周围有很多噪点。要抹除这些孤立的小点，对图像进行开运算。

例：抹除图像外部噪声：文字周围有很多噪点。要抹除这些孤立的小点，对图像进行开运算。

例：抹除图像外部噪声：对比滤波器和开运算。

例：抹除图像外部噪声：对比滤波器和开运算。

```
import cv2
import numpy as np
img = cv2.imread("w1.png")  # 读取原图
k = np.ones((9, 9),np.uint8)  # 创建9*9的数组作为核
cv2.imshow("img", img)  # 显示原图
dst1 = cv2.blur(img, (9, 9))  # 使用大小为9*9的滤波核进行均值滤波
dst = cv2.erode(img, k)  # 腐蚀操作
dst2 = cv2.dilate(dst, k)  # 膨胀操作
cv2.imshow("dst1", dst1)  # 显示均值滤波结果
cv2.imshow("dst2", dst2)  # 显示开运算结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例：抹除图像外部噪声：对比滤波器和开运算。

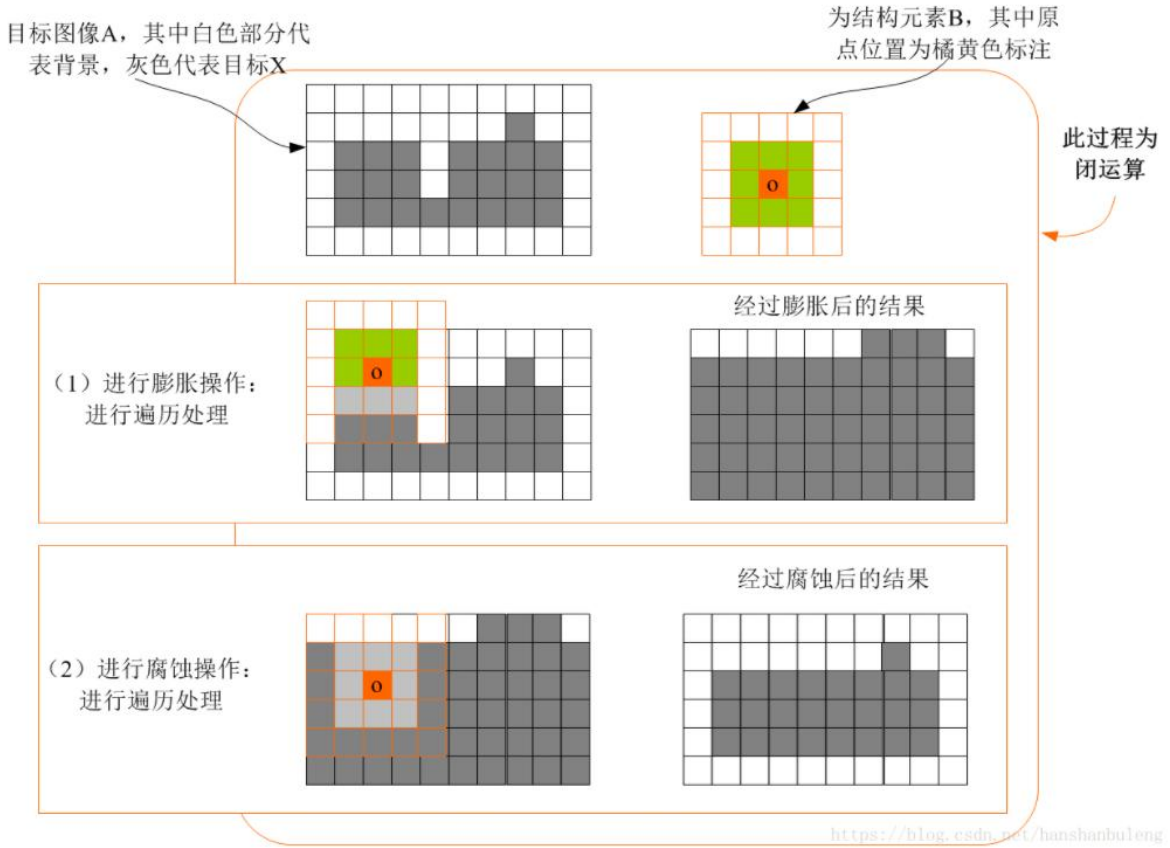
四 闭运算

闭运算： 先膨胀后腐蚀的过程称为闭运算。

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

- 闭运算具有填充物体内细小空洞、连接邻近物体、在不明显改变物体面积的情况下平滑其边界的作用。
- 闭运算会去除内尖角。

四 闭运算


目标图像A，其中白色部分代表背景，灰色代表目标X
为结构元素B，其中原点位置为橘黄色标注
此过程为闭运算
经过膨胀后的结果
（1）进行膨胀操作：进行遍历处理
经过腐蚀后的结果
（2）进行腐蚀操作：进行遍历处理

例：对汉字图片进行闭运算。

例：对汉字图片进行闭运算。

```
import cv2
import numpy as np
img = cv2.imread("w2.png")  # 读取原图
k = np.ones((10, 10), np.uint8)  # 创建10*10的数组作为核
cv2.imshow("img", img)  # 显示原图
dst = cv2.dilate(img, k)  # 膨胀操作
dst = cv2.erode(dst, k)  # 腐蚀操作
cv2.imshow("dst", dst)  # 显示闭运算结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例：对汉字图片进行闭运算。

作业：常见的卷积核为矩形，可用numpy模块创建卷积核。但是卷积核不止有矩形，还可以为其他形状，可以利用OpenCV中的内置函数cv2.getStructuringElement来创建。请大家查阅资料，学习此方法，对比不同形状卷积核的效果。

小结

1.腐蚀：erode()方法

2.膨胀：dilate()方法

3.开运算:先腐蚀后膨胀

4.闭运算:先膨胀后腐蚀

THANKS