腐蚀与膨胀

腐蚀与膨胀是最基本的形态学操作，主要实现的功能：1、消除噪声，2、分割出独立的图像元素，在图像中连接相邻的元素，3、寻找图像中的明显的极大值区域或极小值区域，4、求出图像的梯度。

膨胀对图像高亮部分进行“领域扩张”，效果图拥有比原图更大的高亮区域；腐蚀是原图中的高亮区域被蚕食，效果图拥有比原图更小的高亮区域。膨胀有放大的含义，腐蚀有缩小的含义。

腐蚀（erode）

腐蚀的基本思想，侵蚀前景物体的边界（总是试图保持前景为白色）；内核在图像中滑动（如在2D卷积中）。只有当内核下的所有像素都是1时，原始图像中的像素（1或0）才会被认为是1，否则它会被侵蚀（变为零）。

边界附近的所有像素都将被丢弃，具体取决于内核的大小.因此，前景对象的厚度或大小减小，或者图像中的白色区域减小.

它有助于消除小的白噪声，分离两个连接的对象

Imgproc.erode(Mat src, Mat dst, Mat kernel, Point anchor, int iterations, int borderType, Scalar borderValue)

参数说明： 1、src：源图像； 2、dst：目标图像； 3、kernel：膨胀操作的核，当为Null时，表示的是使用参考点位于中心的3x3的核。我们一般使用getStructuringElement配合这个参数使用。
4、anchor：锚的位置，默认值为（-1，-1），表示锚位于中心；
5、iterations：迭代使用膨胀的次数，默认为1；
6、borderType：推断外部像素的某种边界模式，默认值为BORDER_DEFAULT；
7、borderValue：当边界为常数时的边界值，有默认值，一般不去管它。

函数Imgproc.getStructuringElement(int shape, Size ksize, Point anchor)会返回指定形状或尺寸的内核矩阵。参数shape在opencv3.2.0中有多达11种取值，这里给出三种：Imgproc.MORPH_RECT（矩形）、Imgproc.MORPH_CROSS（交叉形）、Imgproc.MORPH_ELLIPSE（椭圆形）。ksize和anchor分别代表内核的尺寸和锚点位置。

膨胀（dilate）

膨胀恰好与侵蚀相反。 如果内核下的至少一个像素为“1”，则像素元素为“1”，因此它增加了图像中的白色区域或前景对象的大小增加。

通常，在去除噪音的情况下，腐蚀之后是膨胀，因为，侵蚀会消除白噪声，但它也会缩小我们的物体，所以我们膨胀它，由于噪音消失了，它们不会再回来，则我们的物体区域会增加。 它也可用于连接对象的破碎部分。

代码案例

案例1，腐蚀（erode）：

```
package com.what21.opencv01.demo07;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

/**
 * ?????Imgproc.erode?
 */
public class OpenCVErode {

    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    /**
     * VM
     * -Djava.library.path=D:/Apps/MyTool/opencv/build/java/
x64
     *
     * @param args
     */
    public static void main(String[] args) {
        Mat src = Imgcodecs.imread("D:/1.jpg");
        if (src.empty()) {
            return;
        }
```

```
        Mat dst = new Mat();
        Imgproc.erode(src, dst, new Mat());
        Imgcodecs.imwrite("D:/1.1.jpg", dst);
    }

}
```

1.jpg

案例2，膨胀（dilate）：

```
package com.what21.opencv01.demo07;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

/**
 * ?????Imgproc.dilate?
 */
public class OpenCVDilate {

    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }
```

```
    /**
     * VM
     * -Djava.library.path=D:/Apps/MyTool/opencv/build/java/
x64
     * @param args
     */
    public static void main(String[] args) {
        Mat src = Imgcodecs.imread("D:/1.jpg");
        Mat dst = Imgproc.getStructuringElement(Imgproc.MORP
H_RECT, new Size(30, 30));
        Imgproc.dilate(src, dst, new Mat());
        Imgcodecs.imwrite("D:/1!1.jpg", dst);
        Mat dst2 = Imgproc.getStructuringElement(Imgproc.MOR
PH_RECT, new Size(src.width(), src.height()));
        Imgproc.dilate(src, dst2, new Mat());
        Imgcodecs.imwrite("D:/1!2.jpg", dst2);
    }

}
```

1.jpg

1!1.jpg


1!2.jpg

案例3，腐蚀（erode）&膨胀（dilate）：

```
package com.what21.opencv01.demo07;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

/**
 * ????
 */
public class OpenCVErosionAndErode {

    static {
```

```
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    /**
     * VM
     * -Djava.library.path=D:/Apps/MyTool/opencv/build/java/
x64
     * @param args
     */
    public static void main(String[] args) {
        Mat src = Imgcodecs.imread("D:/1.jpg");
        Mat dilate = src.clone();
        Mat erode = src.clone();
        Mat element = Imgproc.getStructuringElement(Imgproc.
MORPH_RECT, new Size(3, 3));
        //??
        Imgproc.dilate(src, dilate, element, new Point(-1, -
1), 1);
        //??
        Imgproc.erode(src, erode, element, new Point(-1, -1)
, 1);
        Imgcodecs.imwrite("D:/1.dilate.jpg", dilate);
        Imgcodecs.imwrite("D:/1.erode.jpg", erode);
    }

}
```

1.jpg

1.dilate.jpg

1.erode.jpg