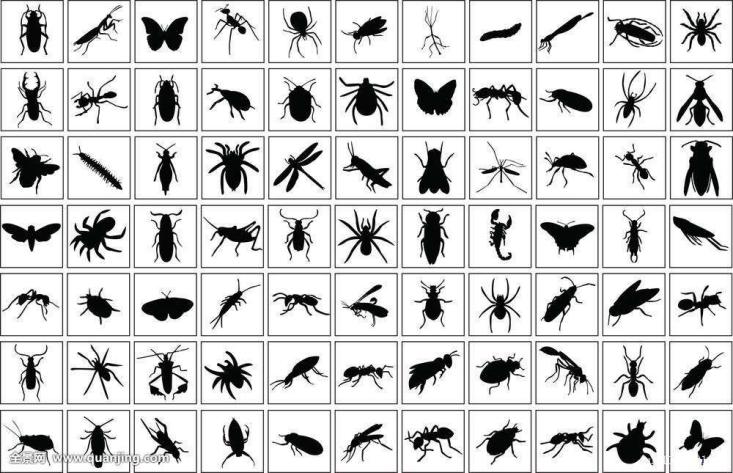
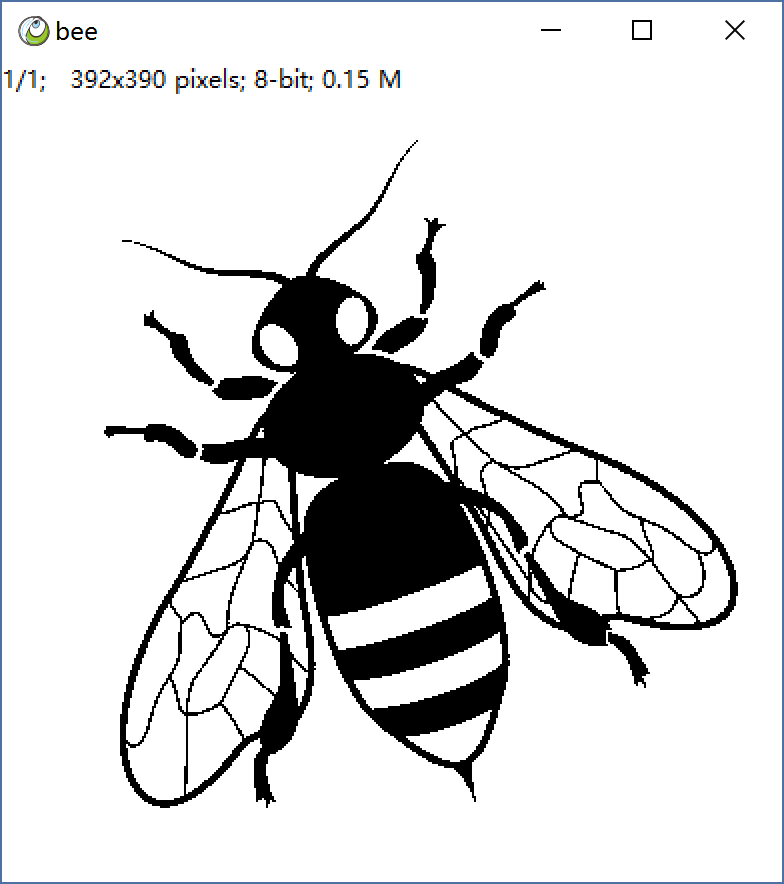
**第七章 二值图像**

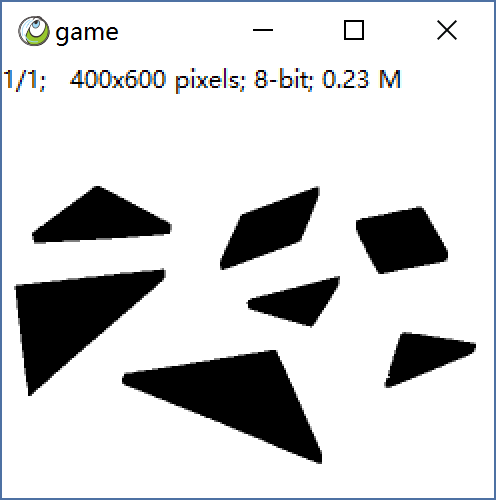
截至目前，我们讨论了图像像素运算，滤波器运算，以及一些在此基础上稍微复杂的特征检测，之前的图像多数是灰度的，也有少数彩色的，但是有这样一种图像，只有两种颜色，非黑即白，这种明确的关系，给我们一些工作带来了很大的方便，因而很多测量，计数等工作都是基于二值图像的，我们本章就来学习二值图像。

昆虫剪影 一只黄蜂

**二值图来源**

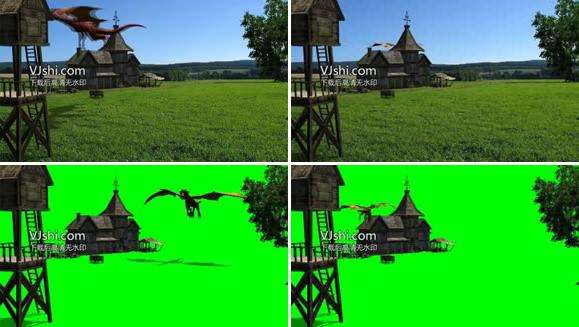
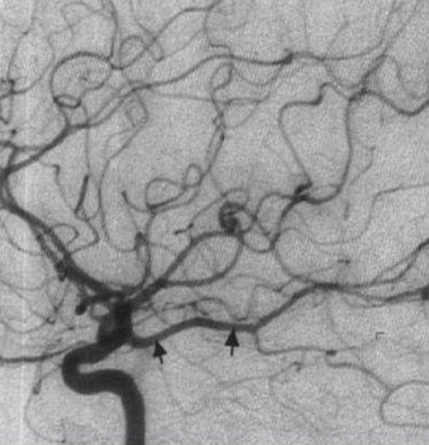
二值图像在自然界是不存在的，只是一种逻辑化的图像，它的获得，一种是本身就是人造的，另外一类是通过阈值化或者分割算法得到，也包括人工抠图的结果。

二维码 七巧板 阈值化

**观察描述：**二维码是人工生成的，从诞生一刻就是二值的。而其他一些非二值的，往往是用阈值化处理或其他的数字图像处理方法，转化成二值图像。

**小知识：**图像阈值化往往是一个非常复杂的课题，但有些时候，我们可以控制我们的拍照环境，比如选用合适的背景，合适的光照，从而使前景背景尽可能差异明显，从而简单的得到二值图像。

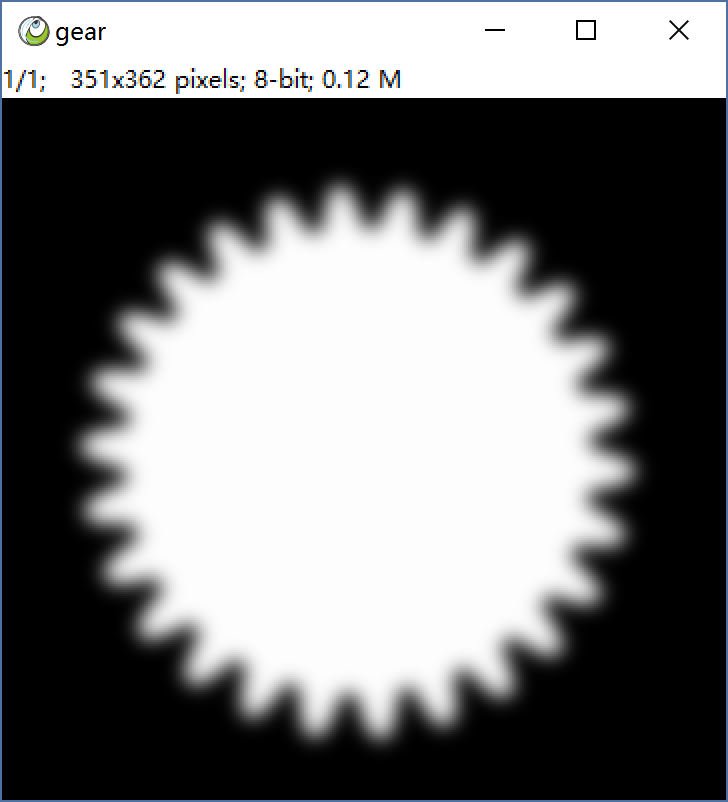
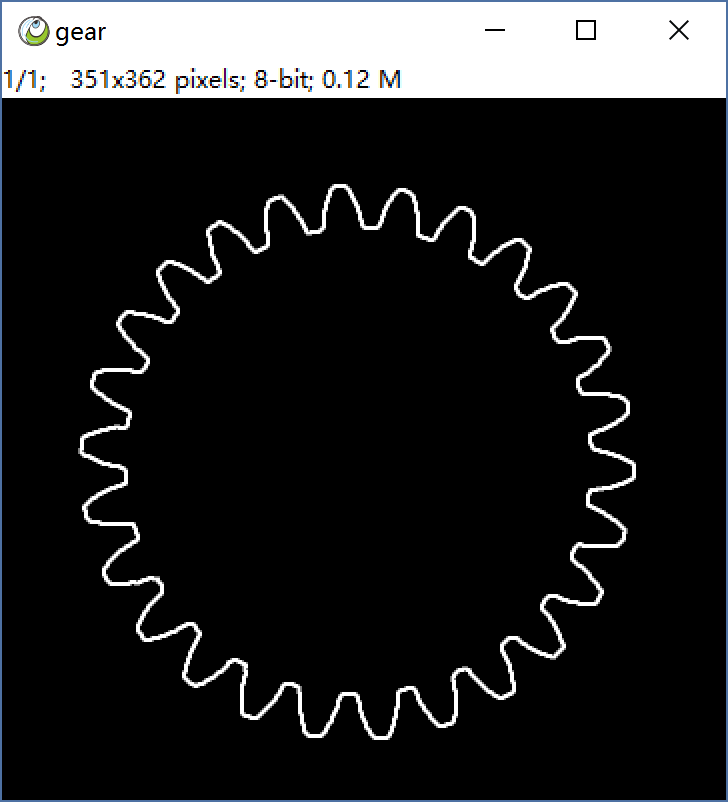
电影后期利用颜色反差完成抠图 注射造影剂凸显血管

**观察描述：**电影后期的景物叠加，就是选用对比强烈的幕布，让软件二值化变得容易，医学上也往往会在血液中注射造影剂协助血管成像，便于判读和诊断。可见如果环境受控，我们应尽量创造好的条件。

**卷积作用于二值图像**

我们第四章讨论了滤波器和卷积，并且认识了很多种不同的滤波器。其实二值图像使普通图像的一个特例，所以卷积运算同样可以作用在二值图像上。我们看看这些滤波器运用在二值图像上使怎样的效果？

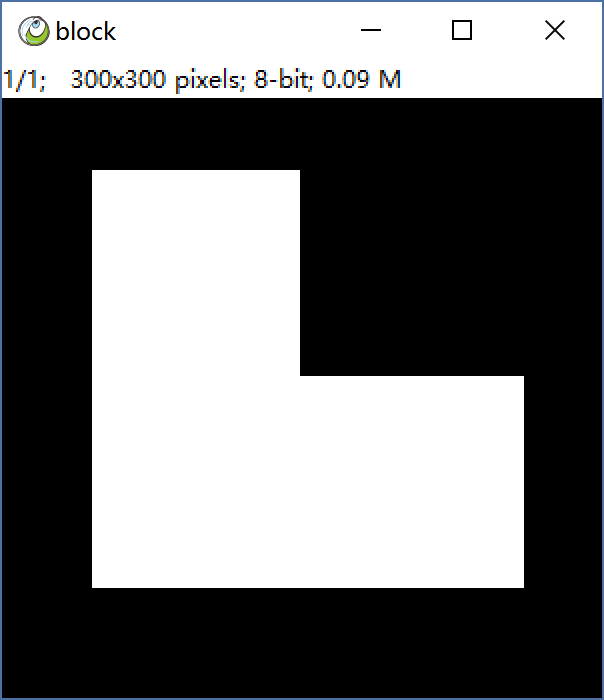
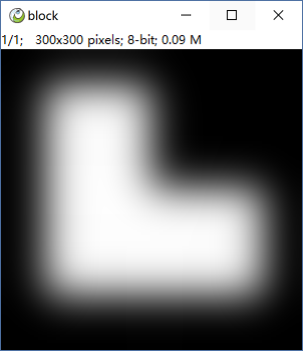
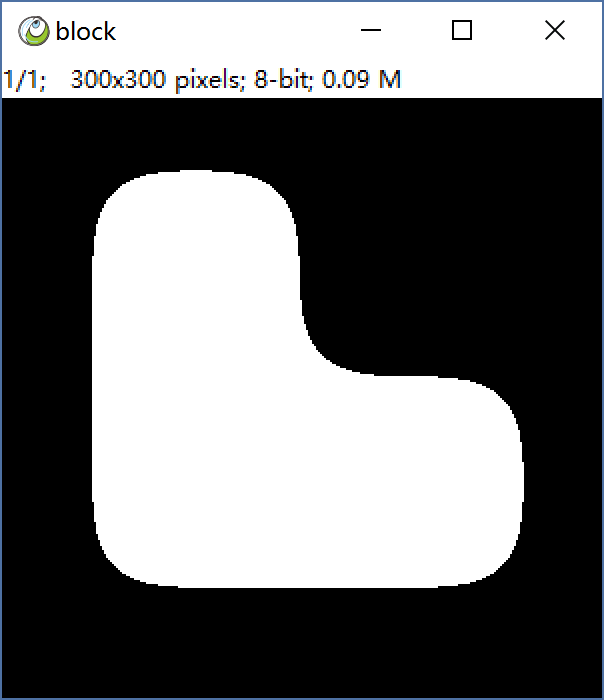
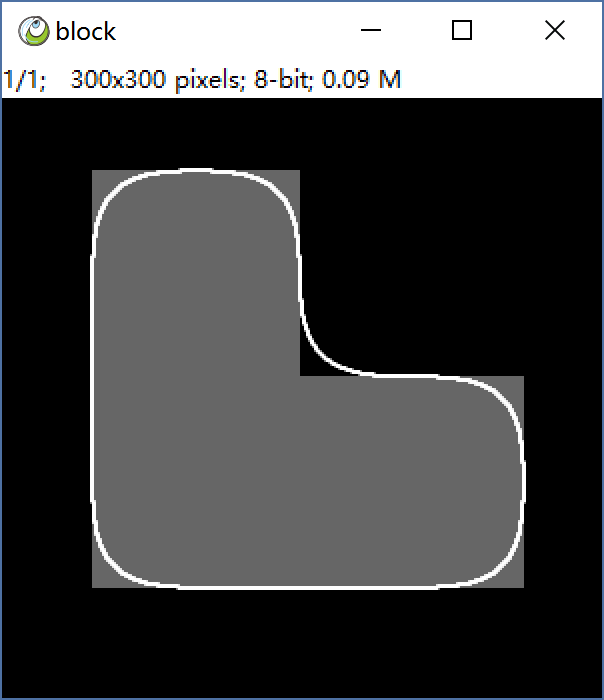
**高斯滤波与梯度滤波 $ IBook > Chapter7 Binary-Image > Show Filter On Binary**

齿轮剪影 gaussian sobel

**观察描述：**以上使一个二值图像的齿轮，对其进行高斯滤波，结果看上去齿轮不再尖锐，而sobel的结果得到了非常理想的边缘。

**高斯后再次阈值化 $ IBook > Chapter7 Binary-Image > Show Smooth The Rec**

二值图像 gaussian sigma=20 threshold thr=128 叠加效果

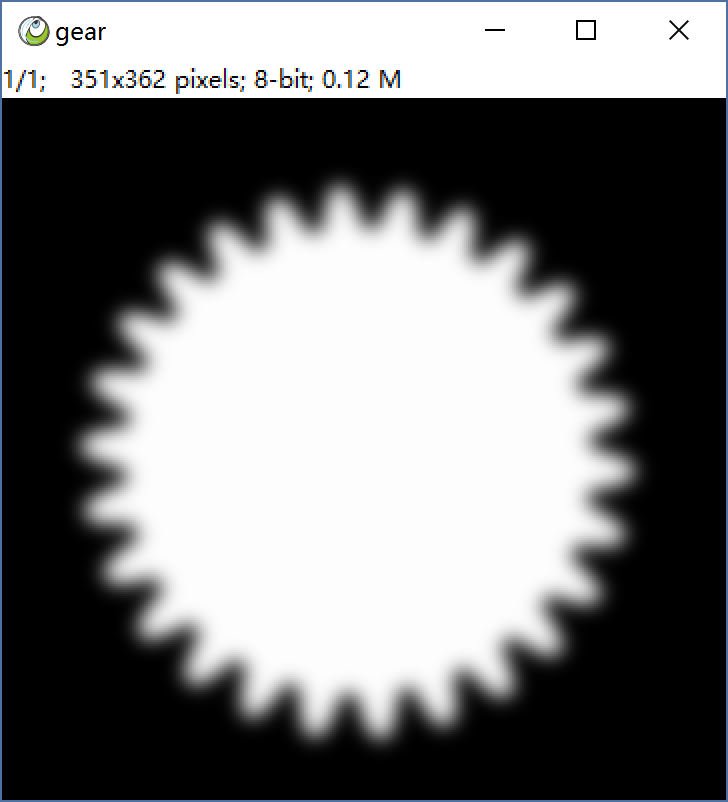
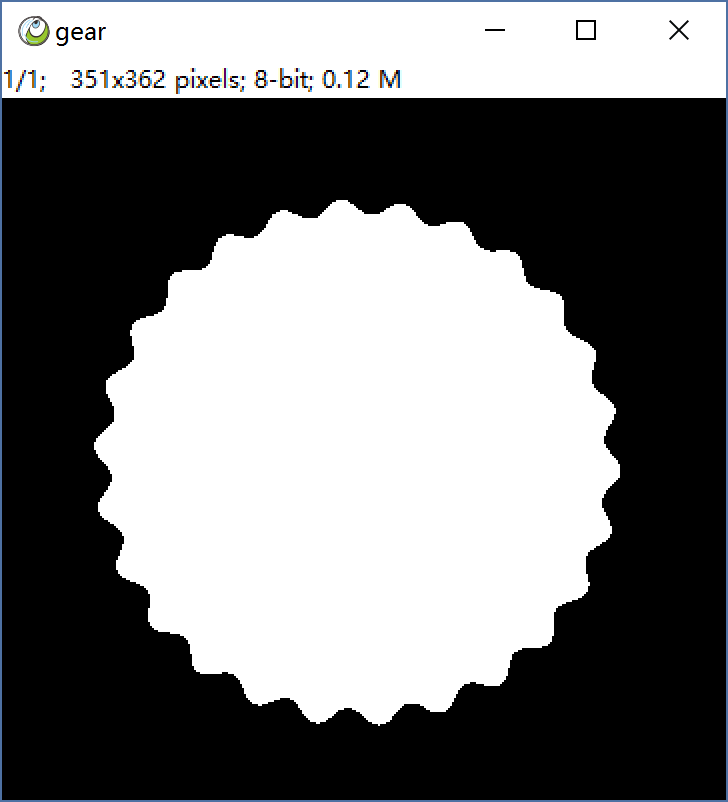
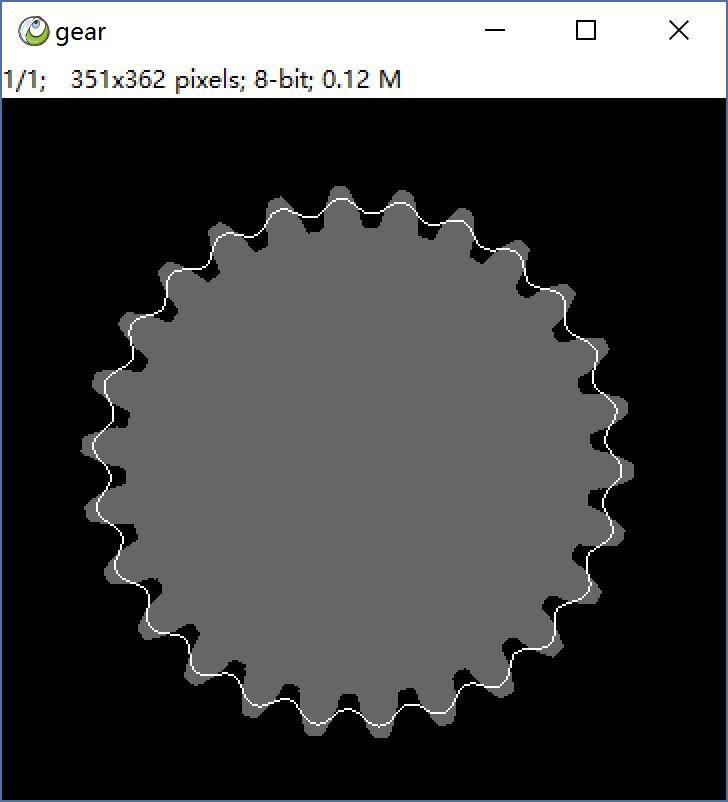
**观察描述：**上图使一个有棱有角的白色区域，接着对其做一个大尺度的模糊处理，然后用128做阈值运算，把二至运算的结果叠加会原图，发现直线部分和原图重合，而棱角处，外凸的被打磨，内凹的向外扩张了，很有趣的结果。

**为什么会有如此现象？为什么是128？**

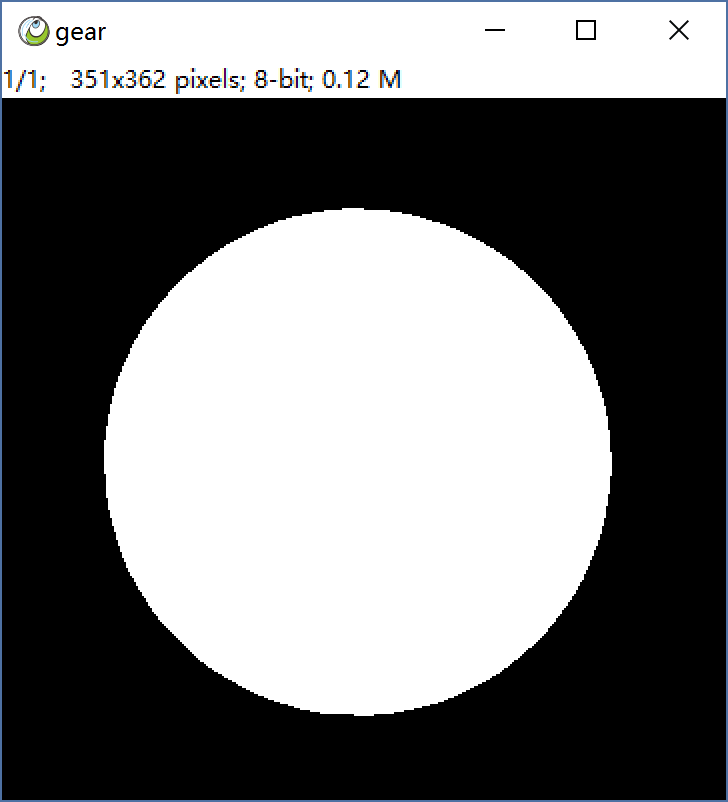
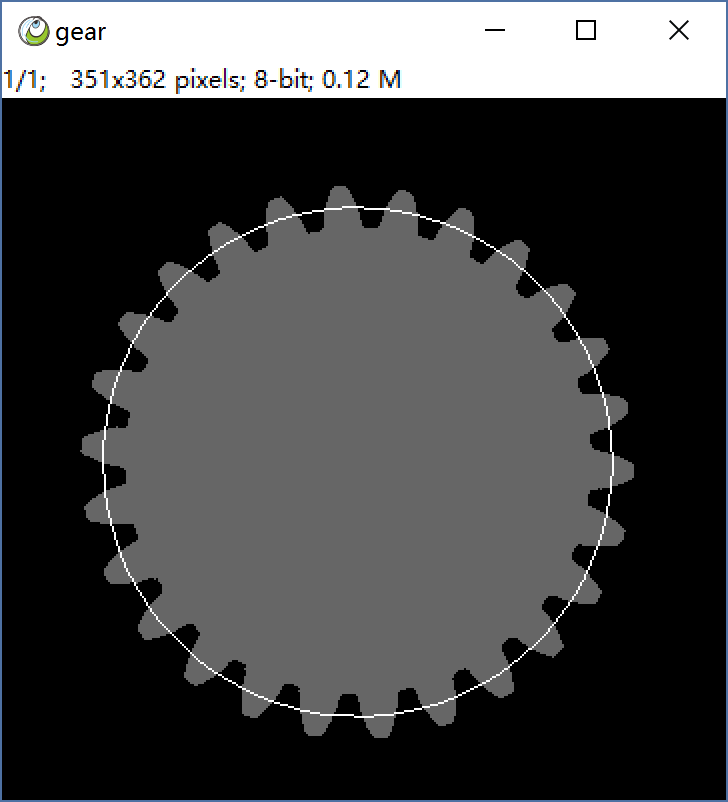
我们知道高斯模糊相当于当前像素与周围像素的融合，并且滤波窗口是对称，无方向性的，因而在直线边缘上，窗口内像素是黑白各半的，因而卷积结果是黑白的平均值，也就是256/2，恰好是128，而在突出区域，黑色像素占优势，因而目标被打磨，内凹部分，白色像素占优势，因而会扩张。

其实这个道理在自然界中很常见，风力侵蚀会产生削峰填谷的效果，因为露在外面，就更容易被风吹到，首当其冲，而凹地，由于背风，因而风沙落到凹地，就更容易沉淀下来。同样的道理可以解释为什么鹅卵石是圆的，不是它们天生是圆的，而是日积月累，不断打磨的概率结果。

**不同Sigma的作用结果 IBook > Chapter7 Binary-Image > Show Smooth The Gear**

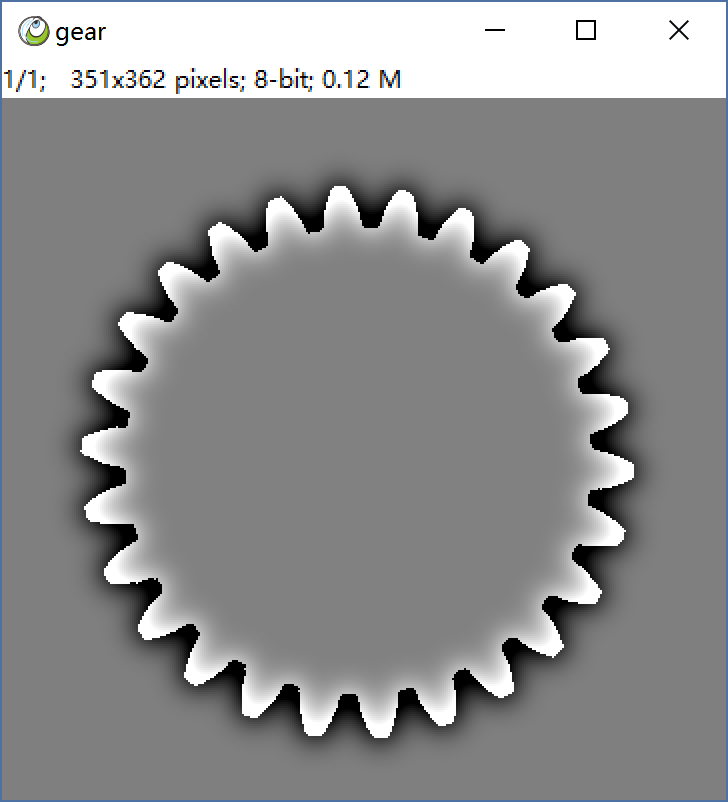
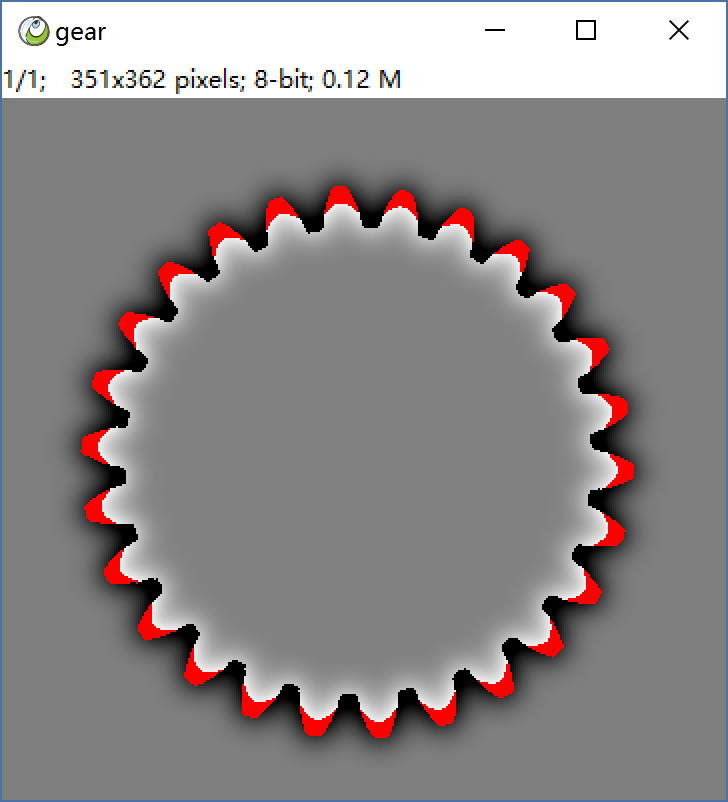
Gaussian sigma=8 threshold thr=128 叠加效果

Gaussian sigma=15 threshold thr=128 叠加效果

**观察描述：**sigma标识高斯窗口的大小，sigma越大，代表与周围的融合程度越高（也可以联想成这块鹅卵石被打磨的时间越久），我们看到随着sigma的增加，齿轮渐渐变得不尖锐，而当sigma增加到15时，已经看不到齿轮了，区域阈值化后，变成了一个圆。不错，这或许正是鹅卵石最终的归宿。

**差分高斯滤波 IBook > Chapter7 Binary-Image > Show Dog The Gear**

原图 - 高斯 sigma=8 阈值 结果

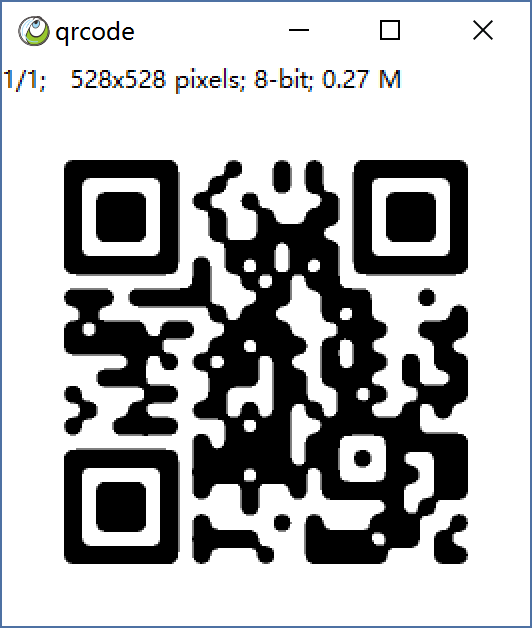
**观察描述：**DOG滤波的结果，齿轮突出部分最亮，而凹处最暗。我们已经学习了，DOG滤波器本质是原图减去一个高斯滤波的结果，原本突出部分被打磨掉了，因而差异呈高亮，而凹处，原本是黑色，但经过高斯被周围的白色融合，因而差异成暗色。

**经过DOG之后，二值图像中的突出部分呈现高亮，而低凹部分呈现最暗。（这种方法可以用于缺陷检测）**

**计算齿的个数**

经过上面的DOG滤波，接着对图像进行阈值处理，我们得到了每个齿轮突出部分的碎片，至此我们已经将齿轮有多少个齿，转化成了图像中有多少块白色区域的问题，这个问题我们在随后的学习中就会讨论。

**液态二维码 IBook > Chapter7 Binary-Image > Show QRCode Art**

二维码 高斯模糊 阈值

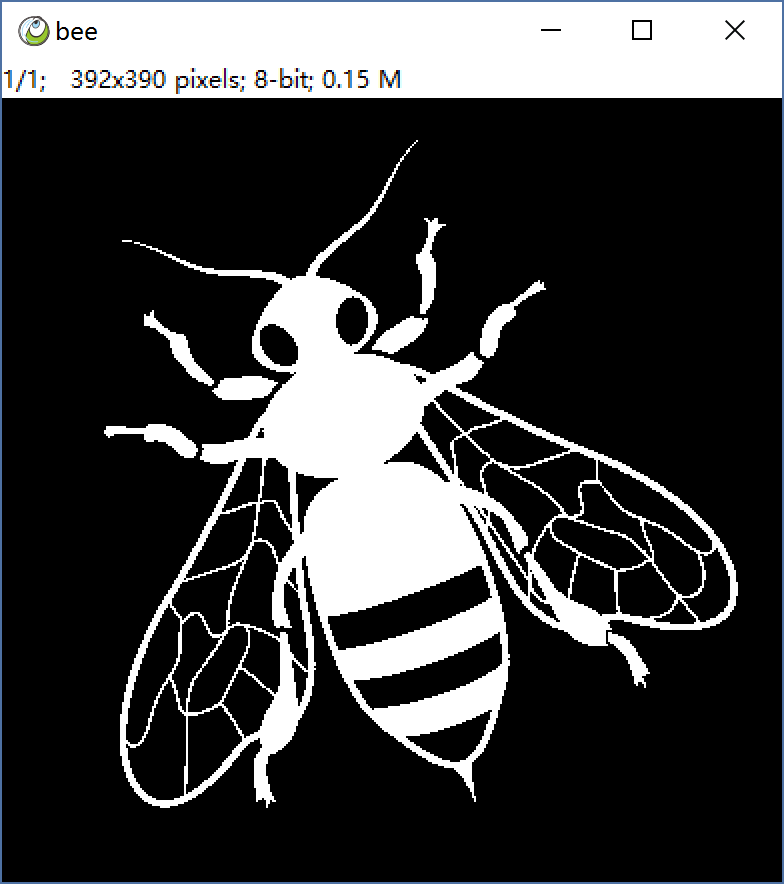
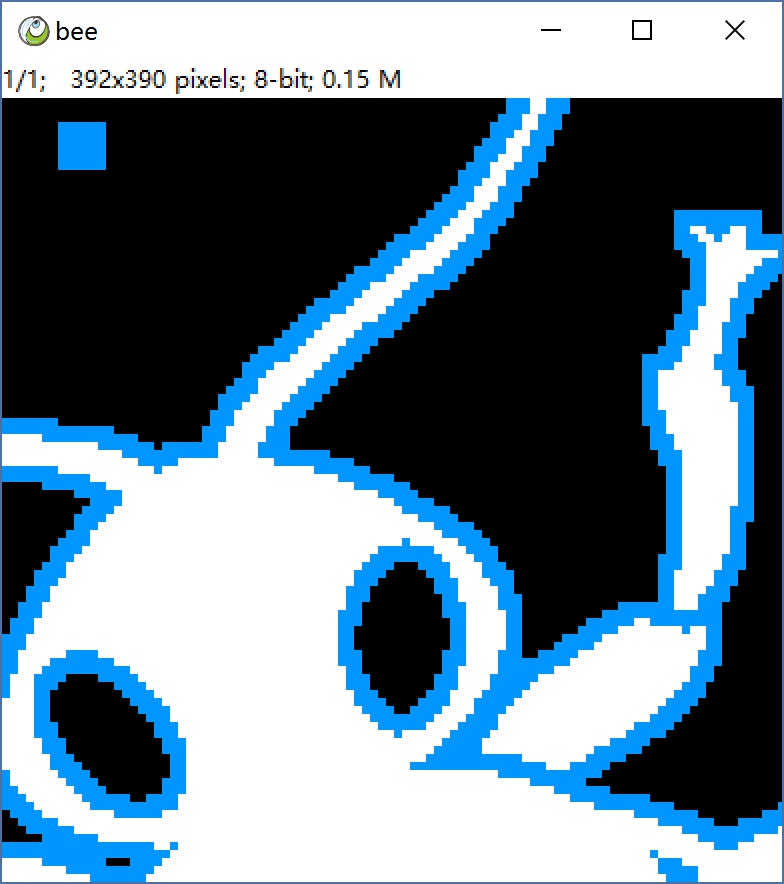
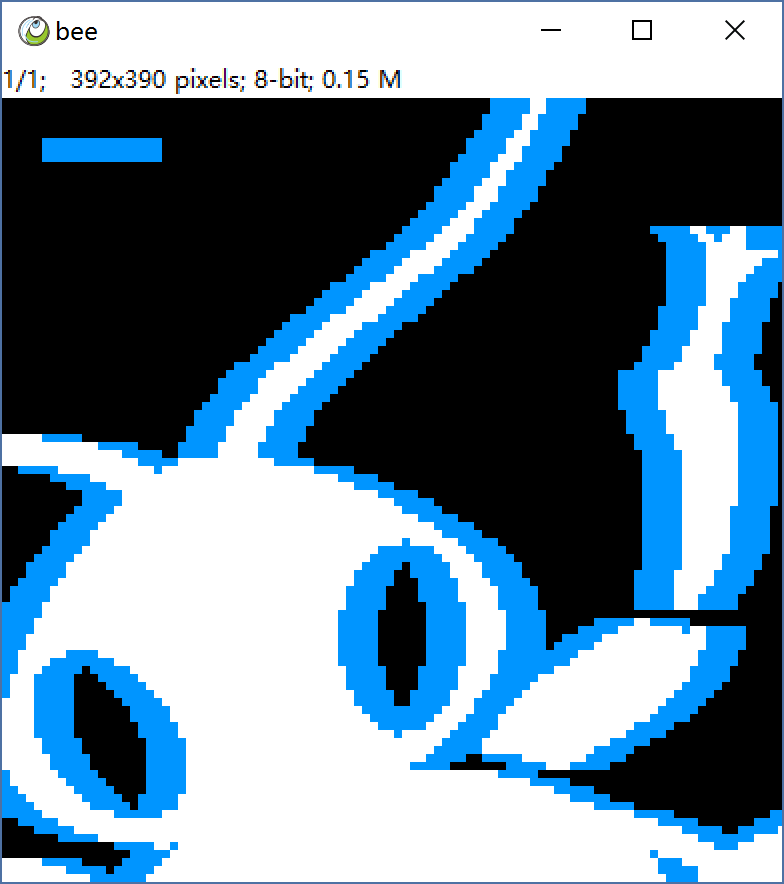
最后来一点趣味性的，我们经常看到的液态二维码，也可以利用我们刚才介绍的只是进行生成。

**形态学滤波**

的确，作为灰度图像的特例，卷积滤波可以用在二值图像上。然而还有一些作为二值图像特有的滤波方式，这些滤波器是非线性的，不能用卷积标识，并且其结果依旧是二值图像，唯一的作用是改变目标的形态，因而叫形态学滤波器。

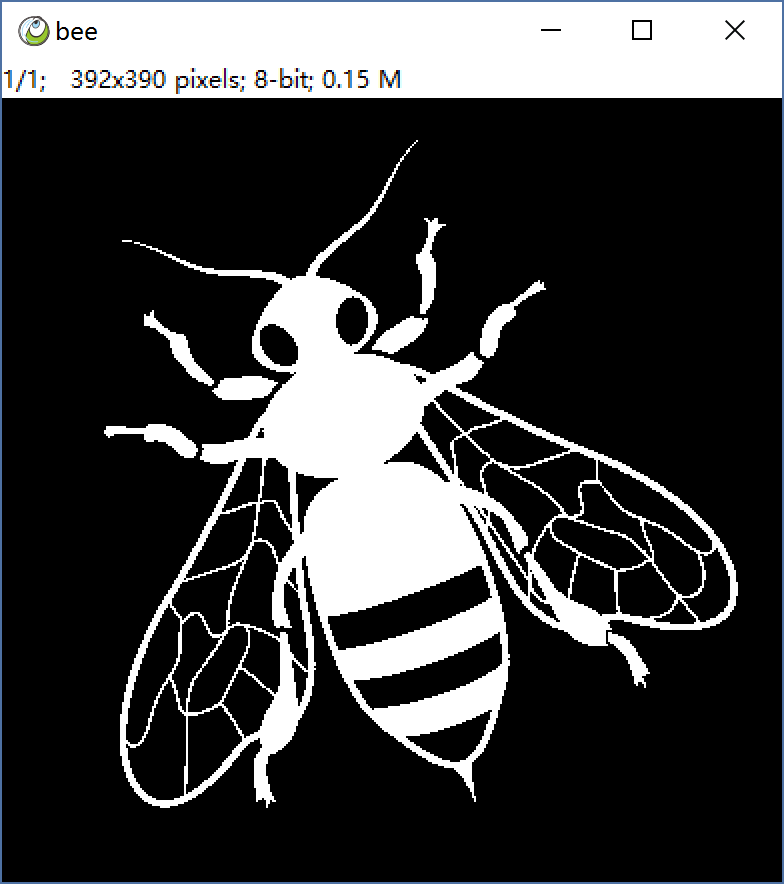
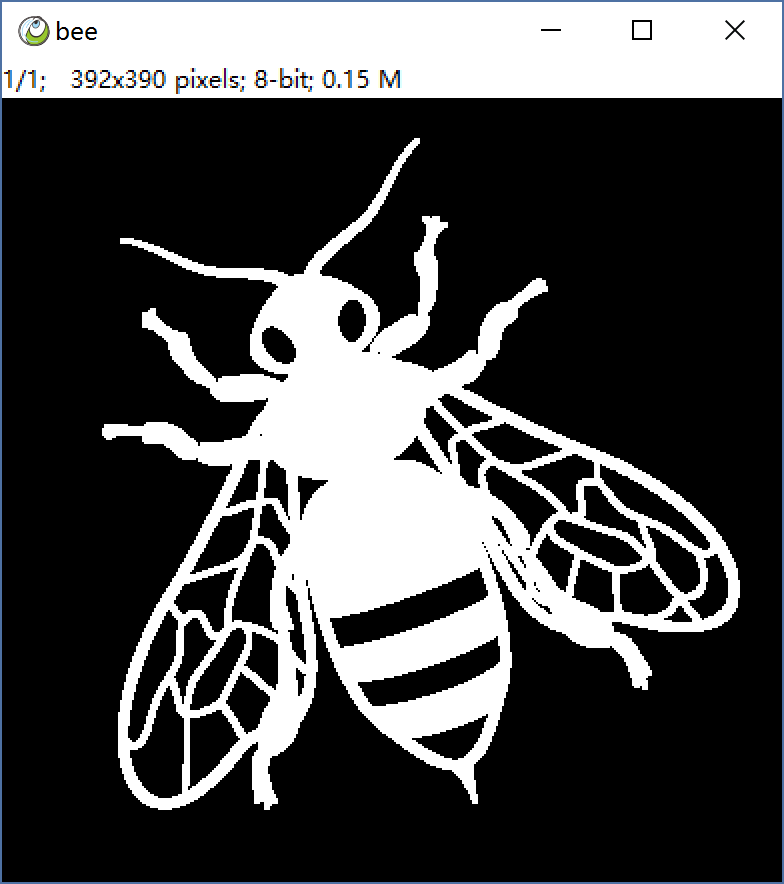
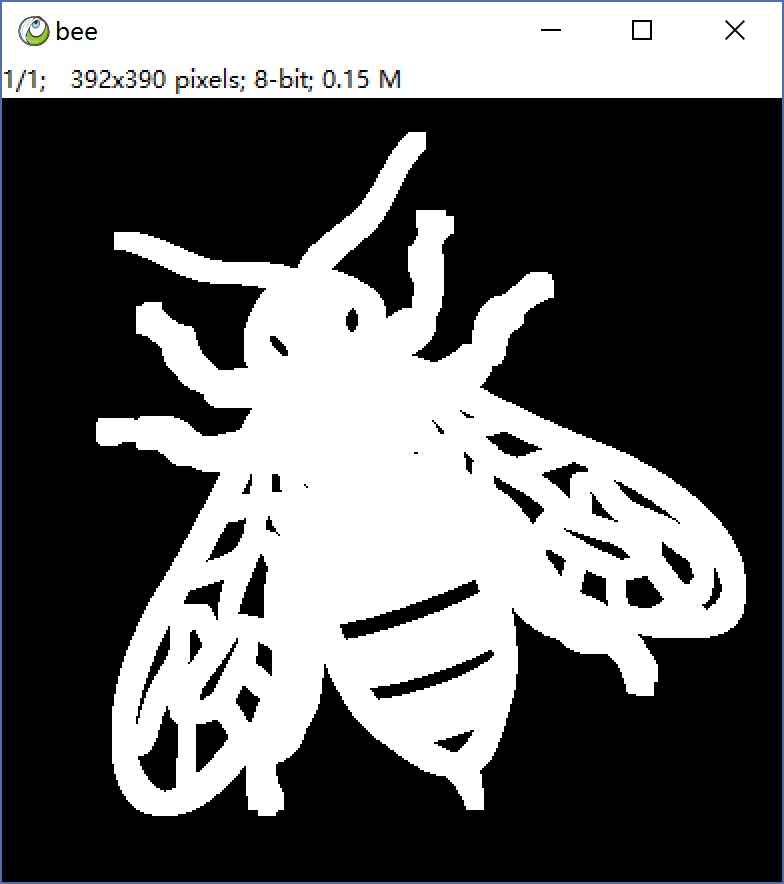
**膨胀 $ Process > Binary > Binary Dilation $ IBook > Chapter7 Binary-Image > Show Dilation**

膨胀是一个使原图变大的形态学运算，其直观理解是用画笔给二值图像描边，而画笔的形状就是滤波核，理论上可以用任意纹理的画笔，但通常我们实用一个矩形区域的画笔。

原图 膨胀5x5 膨胀 11\*1

**观察描述：**图2使用的是一个5x5的滤波核，因而视觉上图像是变大了一圈，而图3使用的是一个扁平的滤波核，因而图像只是在水平方向上变宽了。

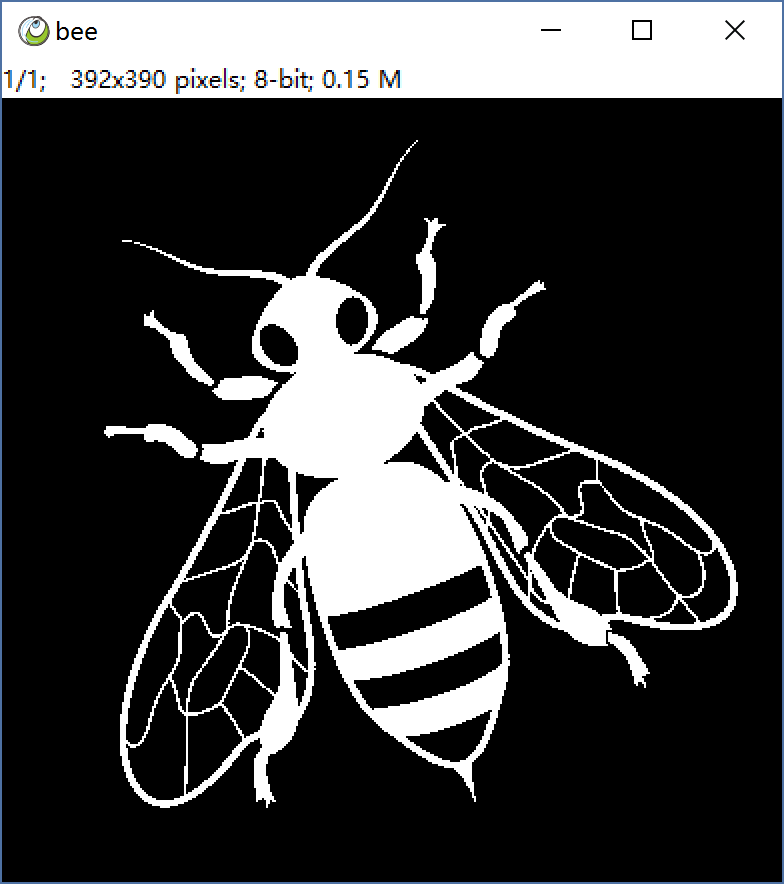
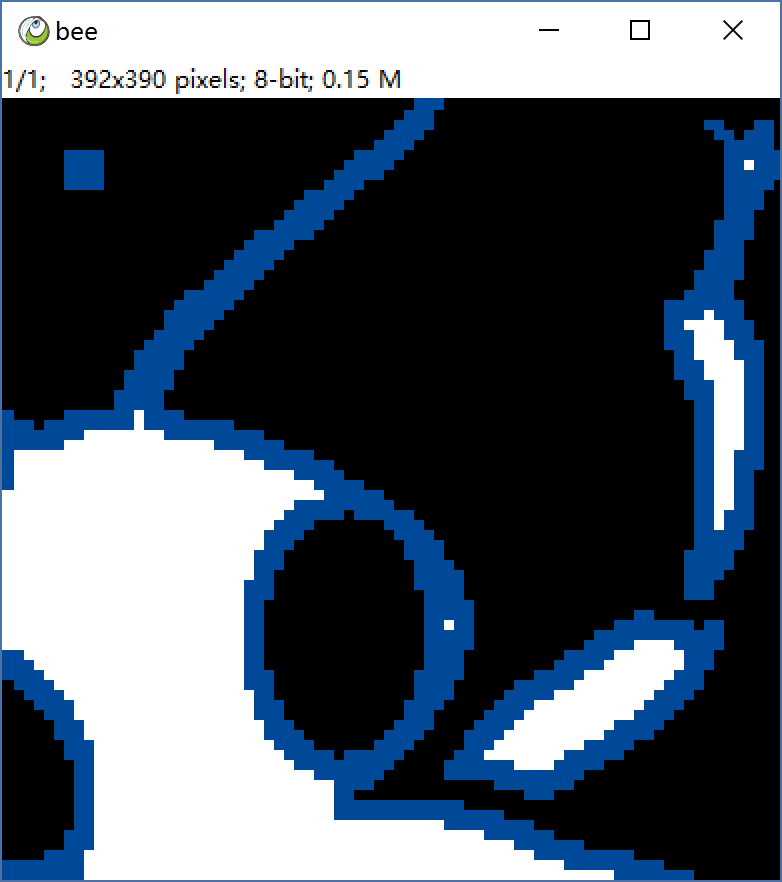
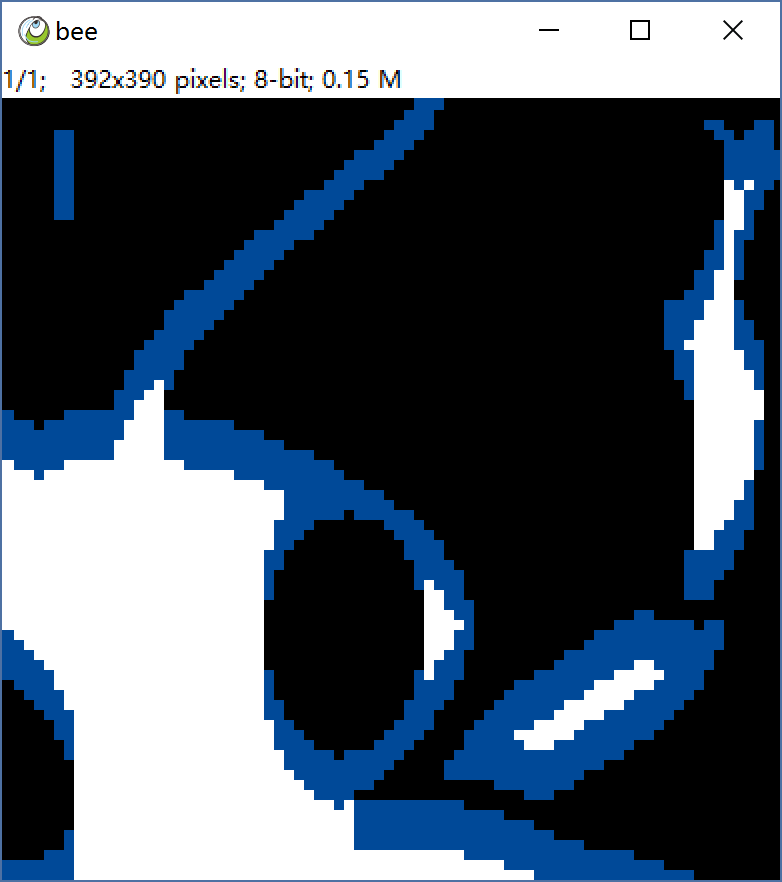
  

1x1（相当于原图） 5x5滤波核 9x9滤波核

**观察描述：**1x1的滤波模板相当于原图，滤波核越大，膨胀后的线条越粗。

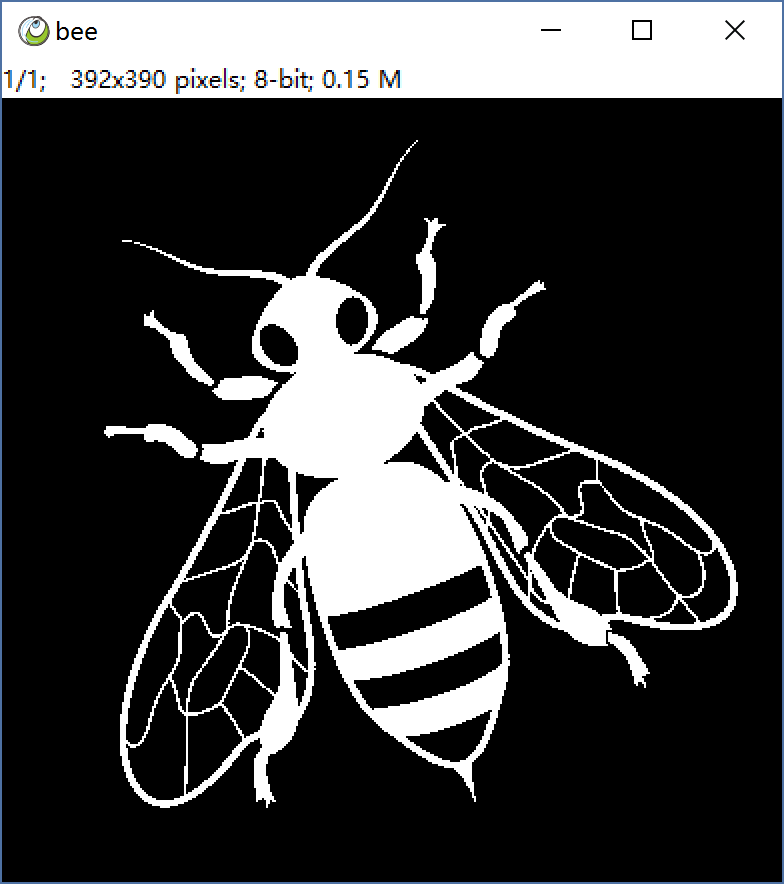
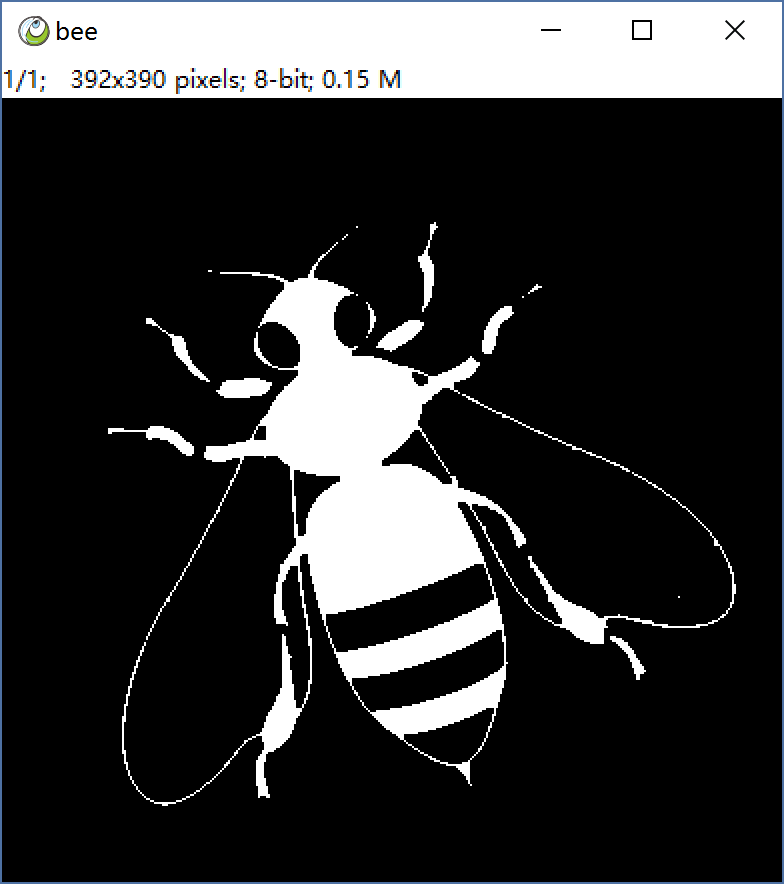
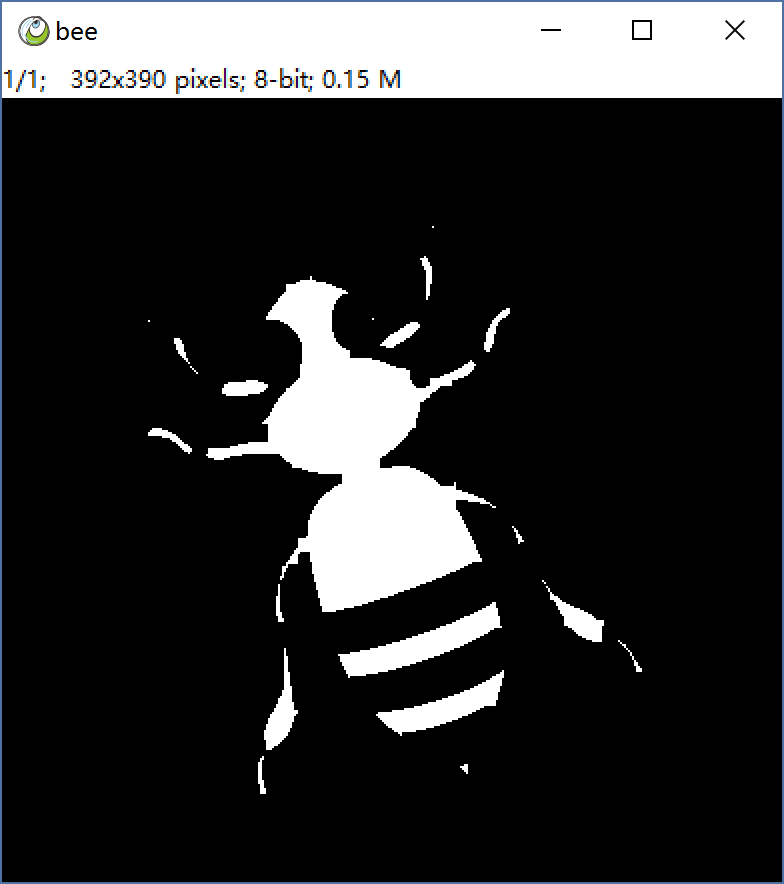
**腐蚀 $ Process > Binary > Binary Erosion $ IBook > Chapter7 Binary-Image > Show Erosion**

腐蚀运算与膨胀相反，是一个让图像变细的滤波器，其直观意义可以理解为用一块橡皮沿着二值图像的边缘擦除，同样橡皮的形状就是滤波核，理论上可以用任意纹理，但通常我们用一个矩形区域的橡皮。

原图 腐蚀5x5 腐蚀1\*11

**观察描述：**图2使用的是一个5x5的滤波核，因而视觉上图像缩小了一圈，特别细的部位消失了，图3使用的是一个1x11的滤波核，因而图像仅仅在纵向被腐蚀，横向得以保留。

1x1（相当于原图） 3x3滤波核 5x5滤波核

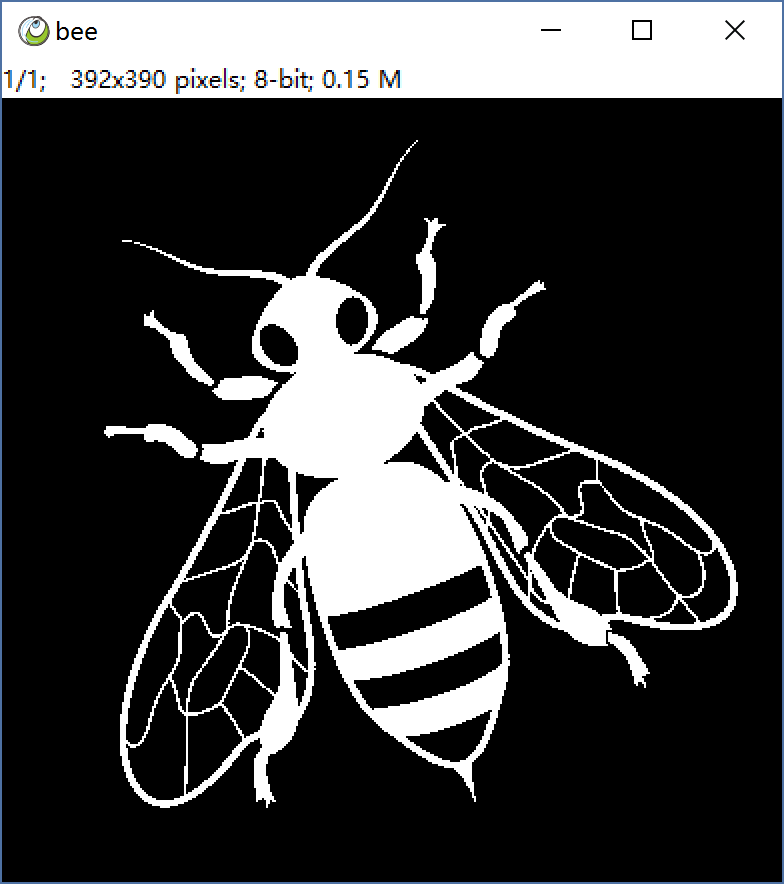
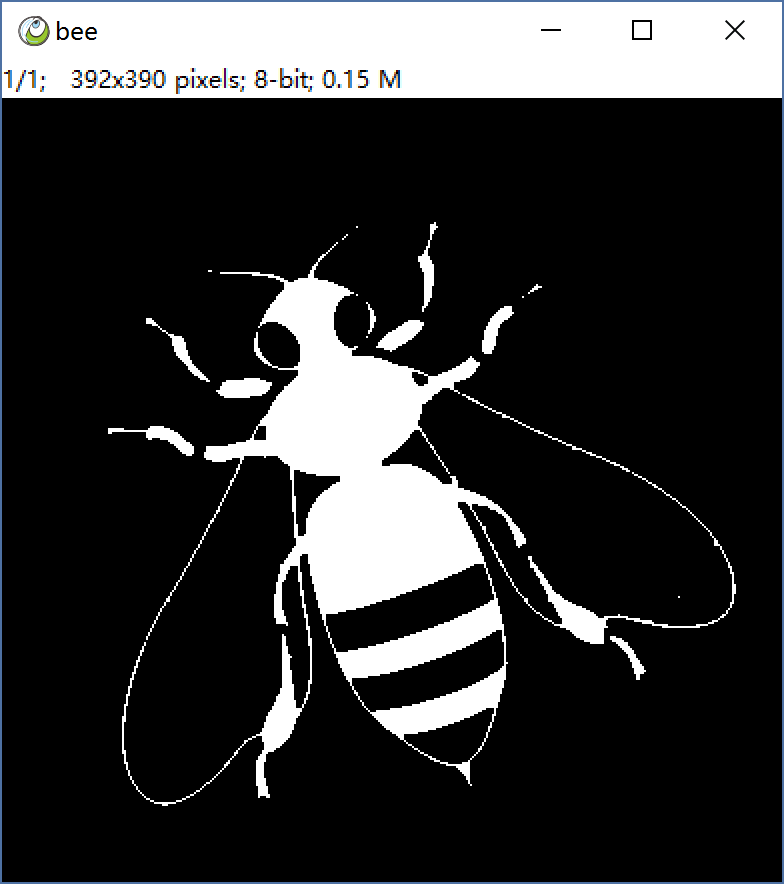
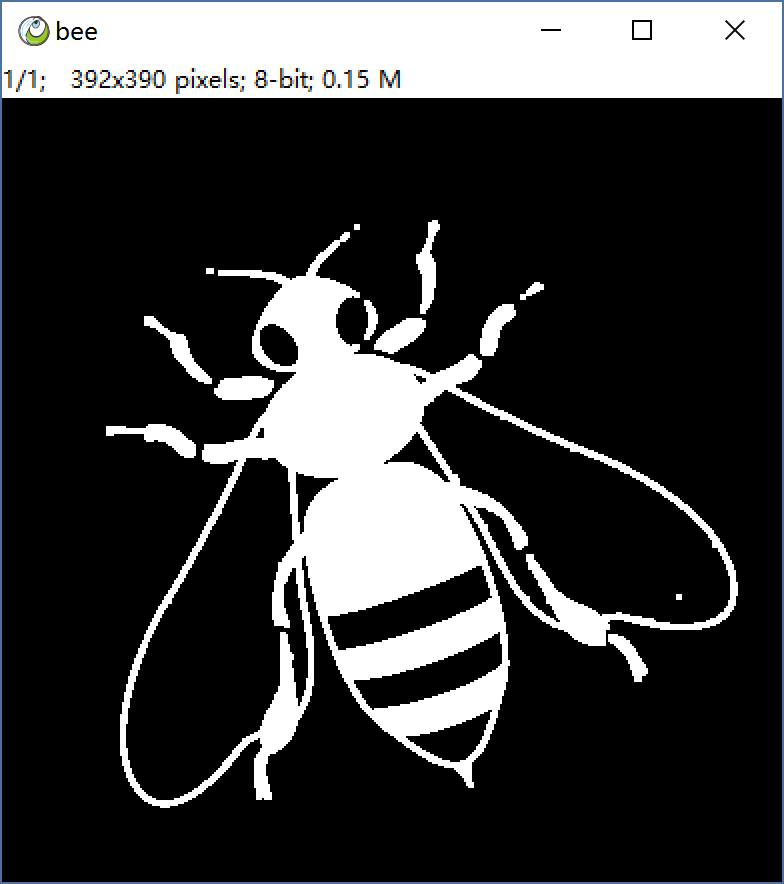
**观察描述：**1x1滤波核的腐蚀相当于原图，当使用3x3滤波核时，翅膀和触角等一些最细的线被腐蚀没有了，而当滤波核增加到5x5时，几乎只剩下躯体了。

**开运算**

**$ Process > Binary > Binary Opening**

**$ IBook > Chapter7 Binary-Image > Show Opening And Closing**

开运算其实是依次腐蚀，接着依次膨胀。腐蚀膨胀某种意义上是逆运算，线腐蚀后膨胀时为了确保不改变粗细，但开运算不是没有意义的。

原图 3x3腐蚀 3x3膨胀

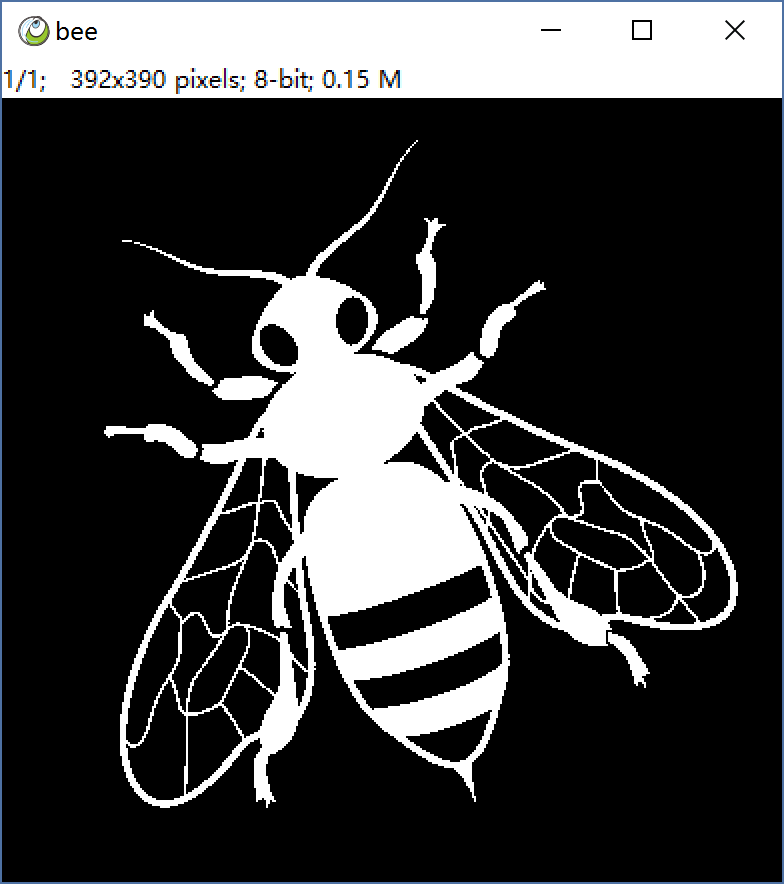
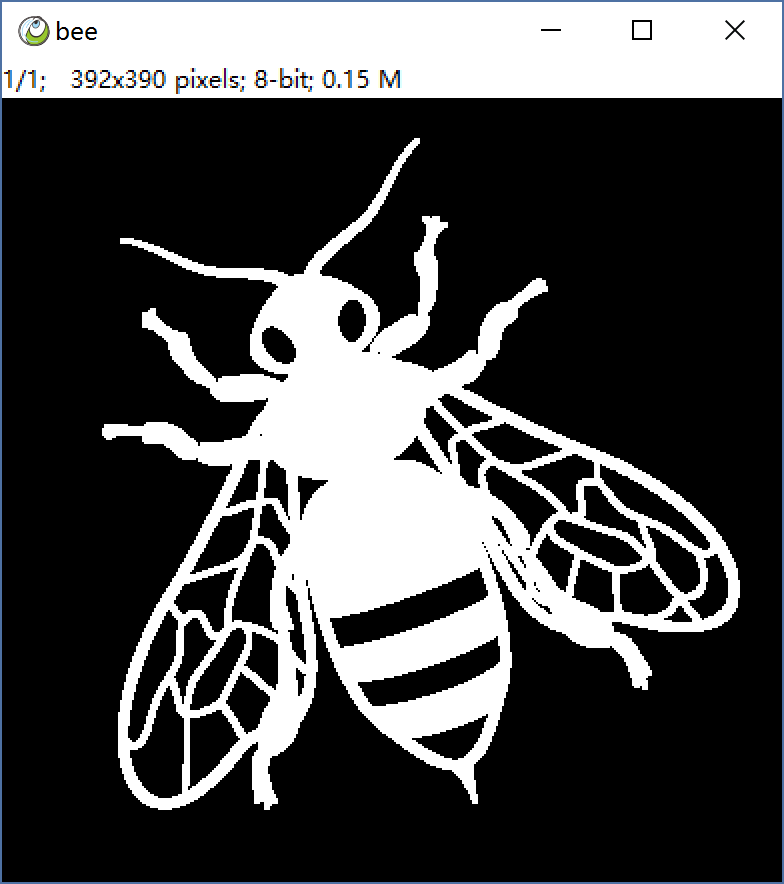
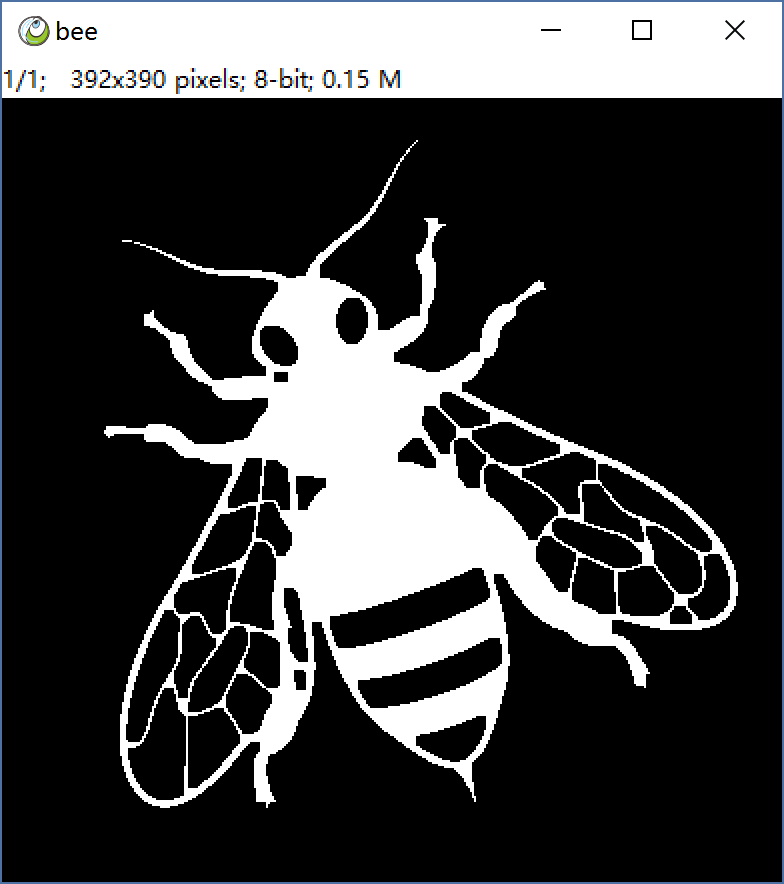
**观察描述：**经过3x3的腐蚀运算，翅膀里面最细的线条完全被腐蚀掉，其他部位基本得以保留，而再次膨胀后，线条粗细基本恢复到原图的样子，但仔细观察，除了翅膀上的线条，还有触角的尖端，我们发现图上一些特别细的线条消失了。**顾名思义，打开一些细小的连接，就是开运算。**

**闭运算**

**$ Process > Binary > Binary Closing**

**$ IBook > Chapter7 Binary-Image > Show Opening And Closing**

闭运算是一次膨胀接着一次腐蚀，同样可以让结果保持宽度，效果大家能否想象？

原图 5x5膨胀 5x5腐蚀

观察描述：经过5x5的膨胀运算，整体线条变宽，而再次腐蚀后，线条宽度恢复，所不同的是大腿和翅膀之间的一些空隙消失了，原本前足的上半截和下半截是不相连的，而闭运算后联为一体。**顾名思义，闭合一些细小的空隙，就是闭运算。**

**对偶性**

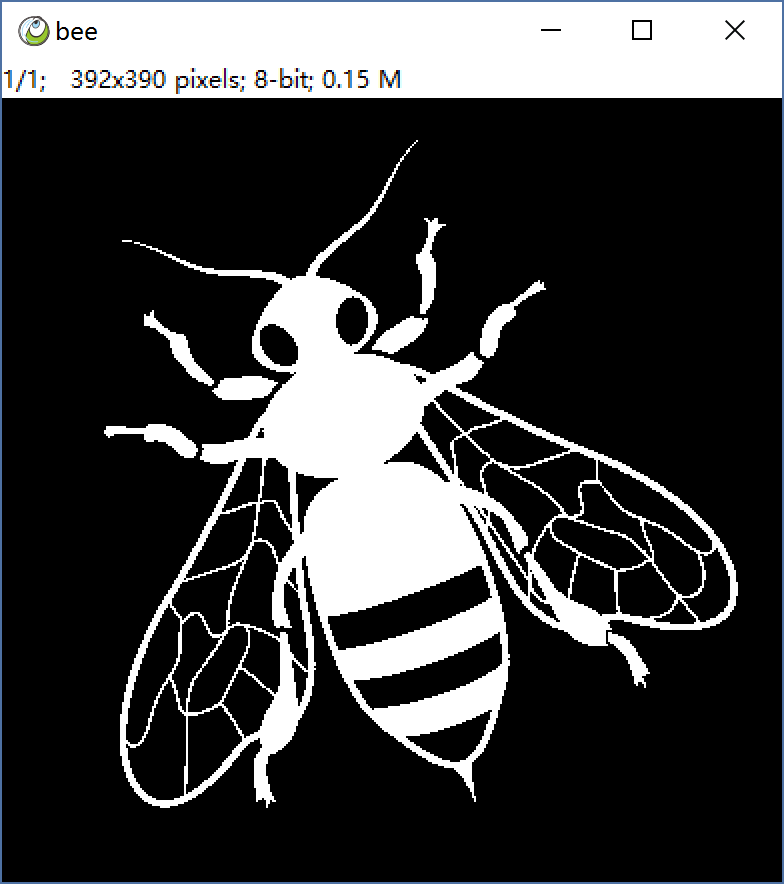
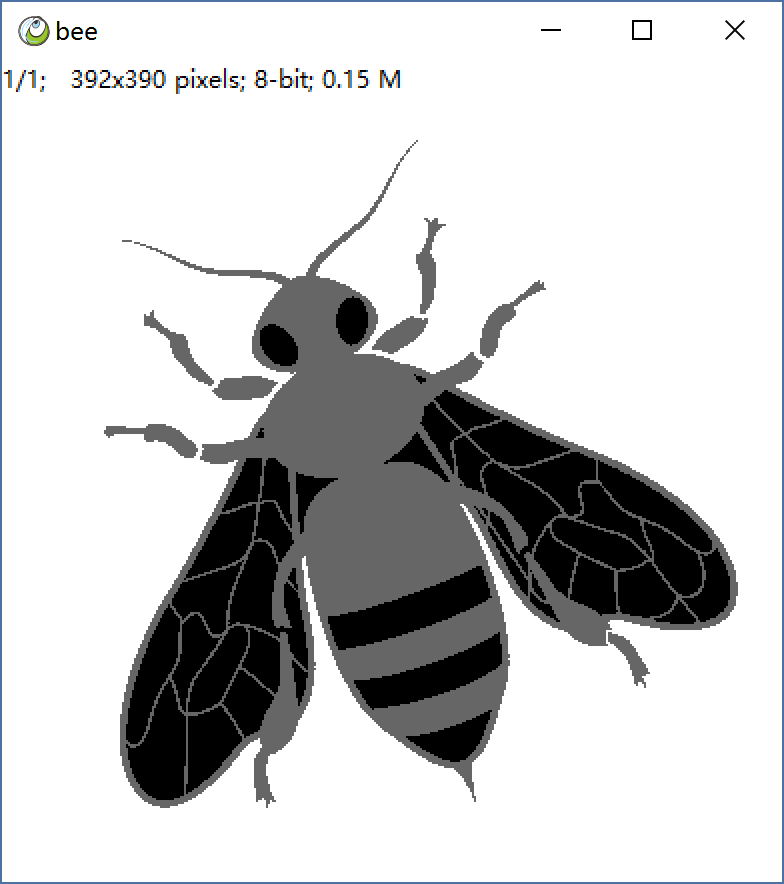
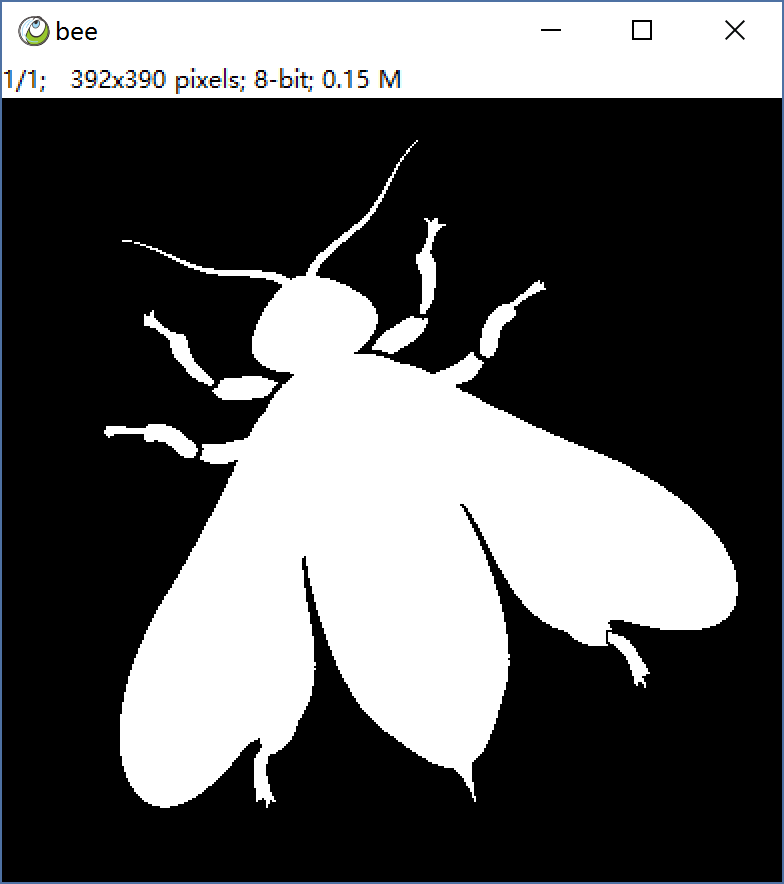
形态学运算具有对偶性，其实腐蚀和膨胀都可以理解为描边，只是用黑色的笔，还是用白色的笔，对白色区域的膨胀，可以理解为对黑色区域的腐蚀，对白色区域的腐蚀，可以理解为对黑色区域的膨胀。同理，白色区域的开，是黑色区域的闭，白色区域的闭是黑色区域的开。一个具体可操作的例子，膨胀可以先取反，然后腐蚀，之后再取反。

**形态运算**

以上的膨胀腐蚀，以及开闭运算，是一种非线性滤波器，具有类似卷积的滤波窗口，接下来我们看几个其他的形态运算。

**填充 $ Process > Binary > Fill Holes $ Chapter7 Binary-Image > Show Fill Holes**

将内部孤立的黑色区域填充，使得二值图像每块区域都是实心的。

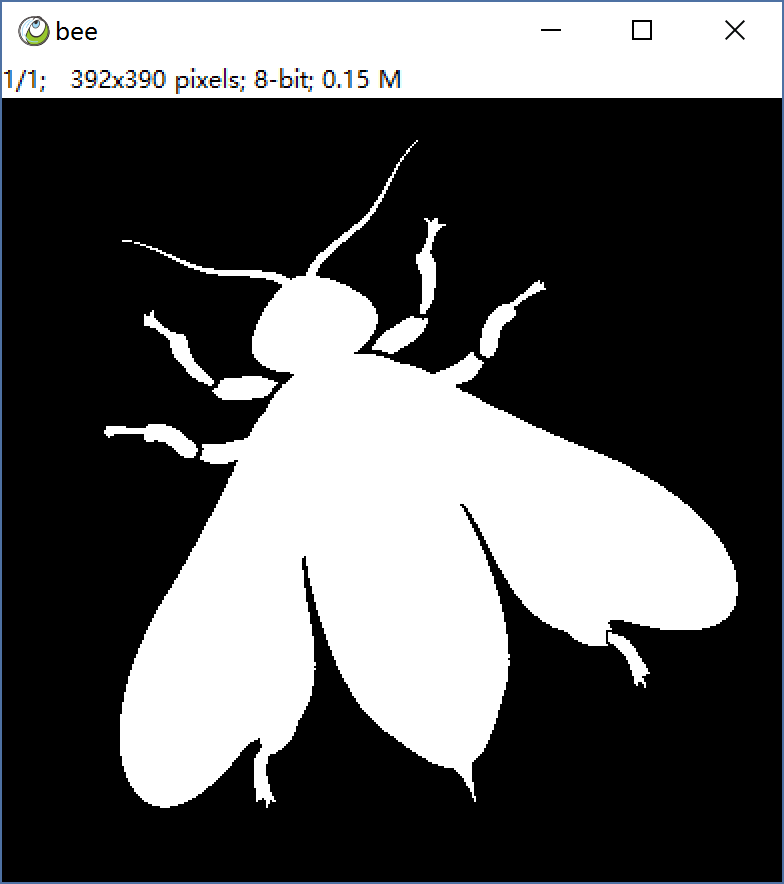
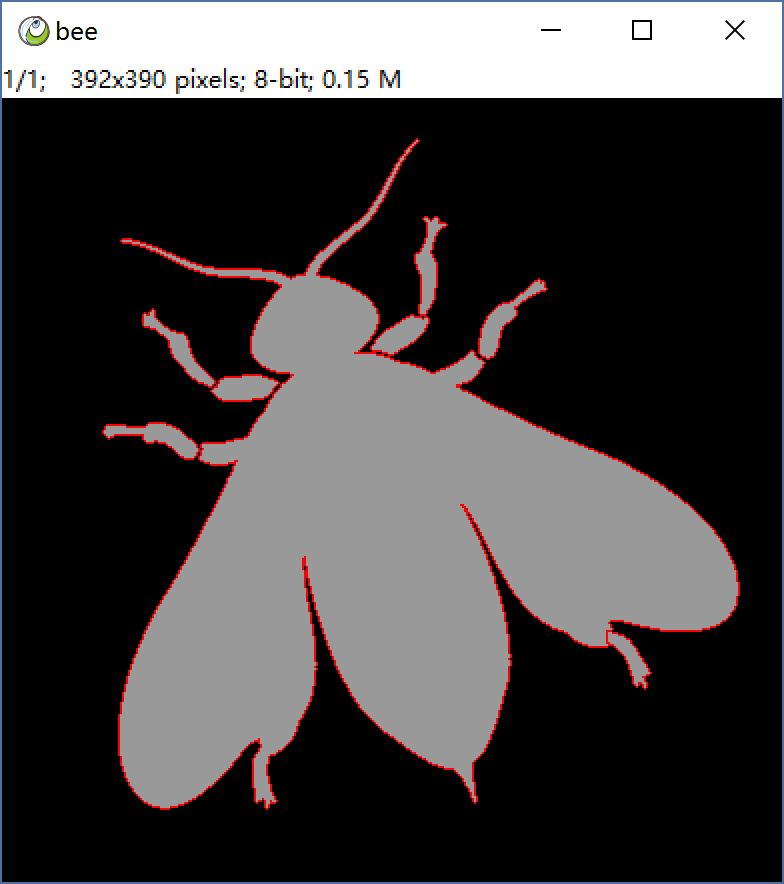
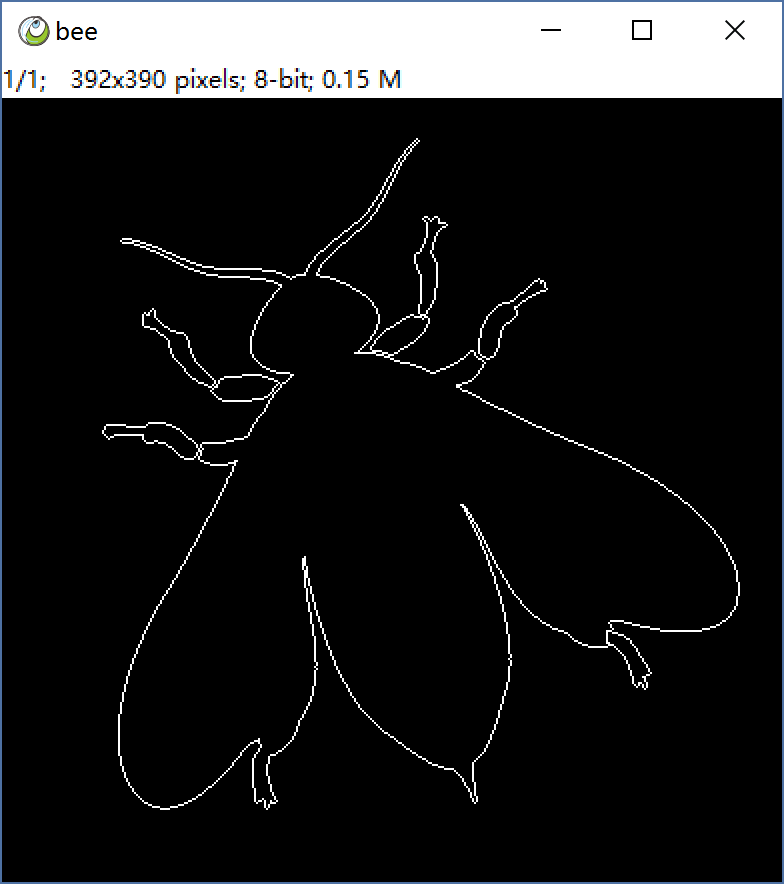
  

原图 填充外部 非外部区域

**观察描述：**内部的孔可能会有很多个，并且关系复杂，所以填充是一个很麻烦的工作，但好在，外部只有一个，所以，我们填充了外部，那么剩下的，就是内部。

**轮廓线 $ Process > Binary > Binary Outline $ Chapter7 Binary-Image > Show Outline**

我们已经学习了canny算子，并且也尝试过用sobel对二值图像滤波。但是对于二值图像而言，有一个更简单更精确的轮廓提取方法，用原图减去原图的腐蚀结果。

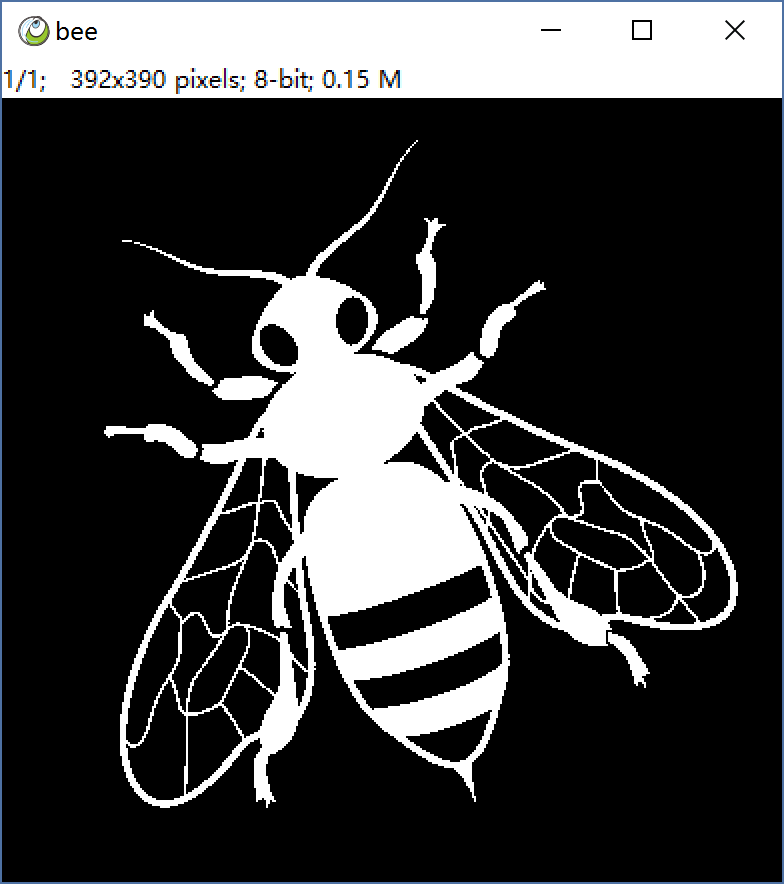
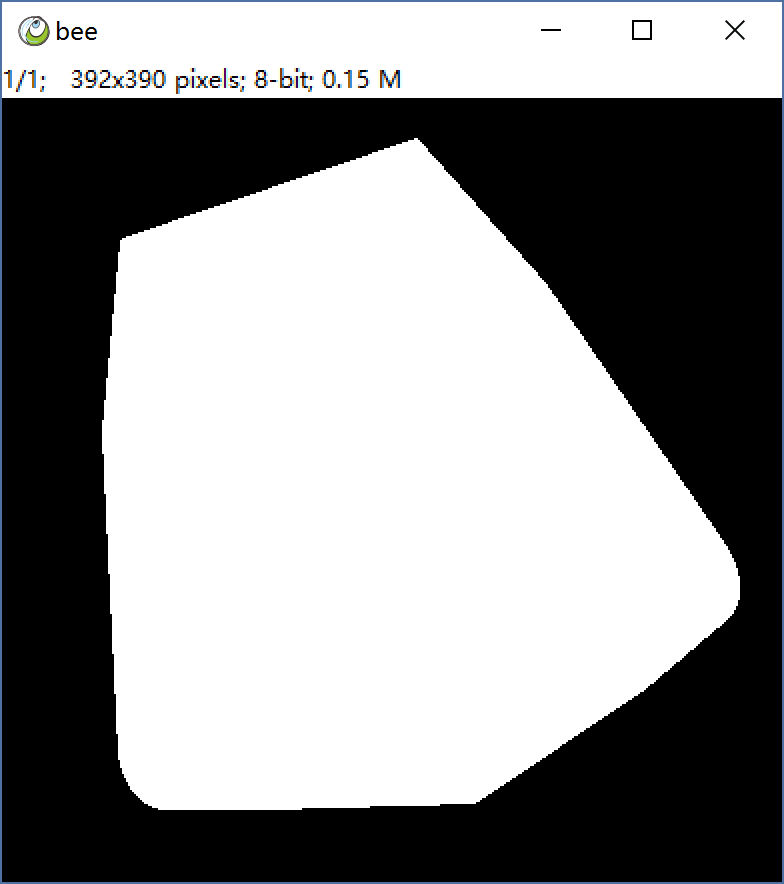
  

原图 减去其腐蚀结果 轮廓

**观察描述：**我们用填充后的结果作为原图，对其进行最小程度的腐蚀，腐蚀后，最外圈一层像素被去除，那么原图减去其腐蚀结果，可想而知，就是最外层像素。

**凸包 $ Process > Binary > ConvexHull $ Chapter7 Binary-Image > Show ConvexHull**

凸包的定义是，包含目标区域的最小凸多边形，感性认识可以理解为目标是一个刚体，我们用一个皮筋框在外面，一松手，啪！

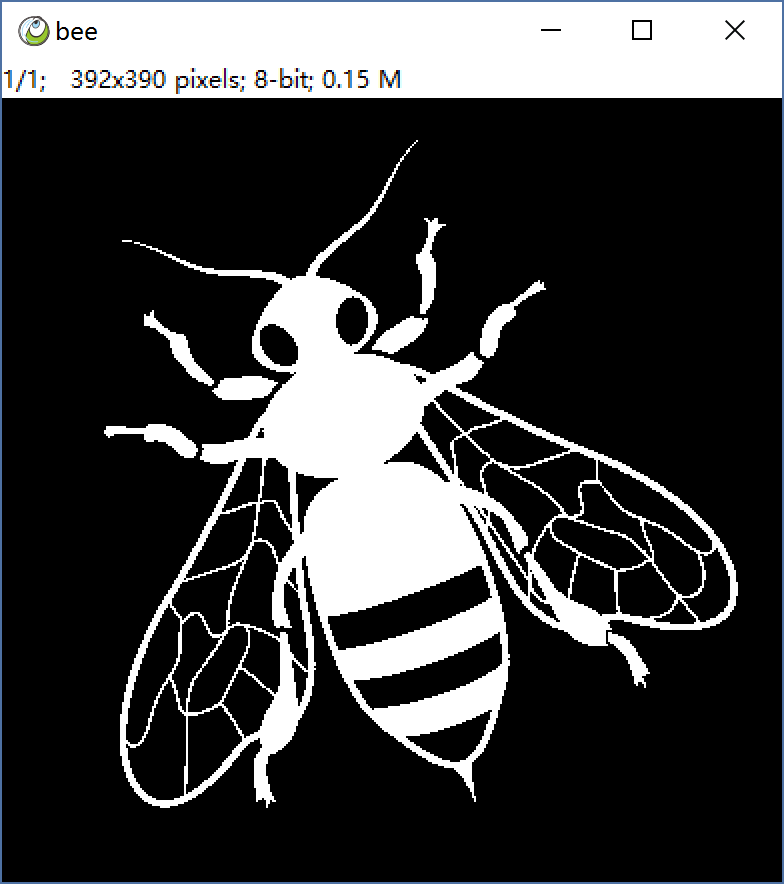
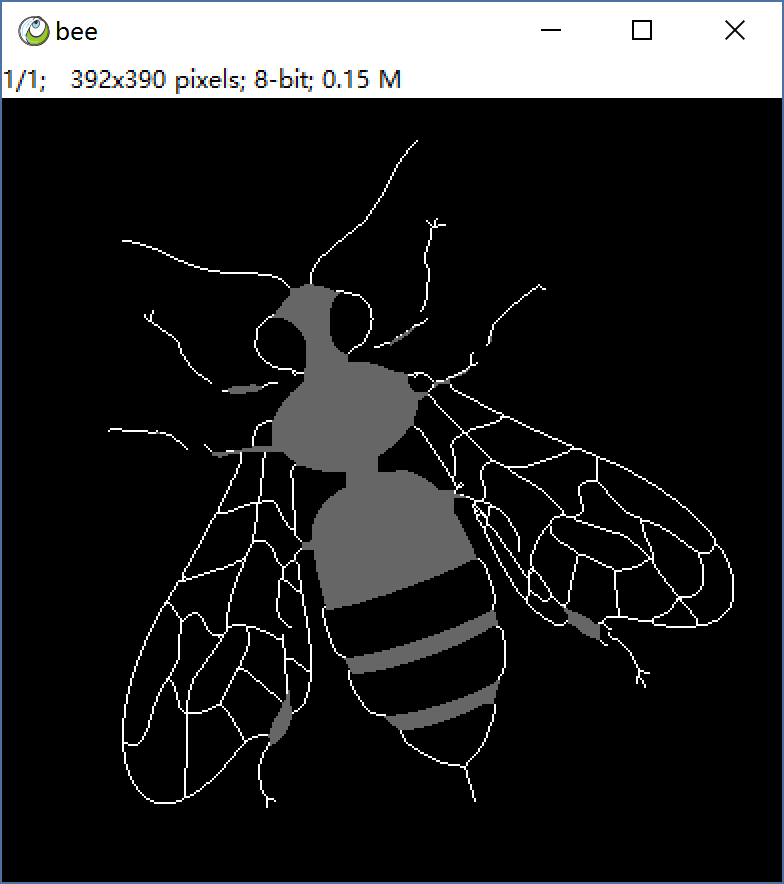
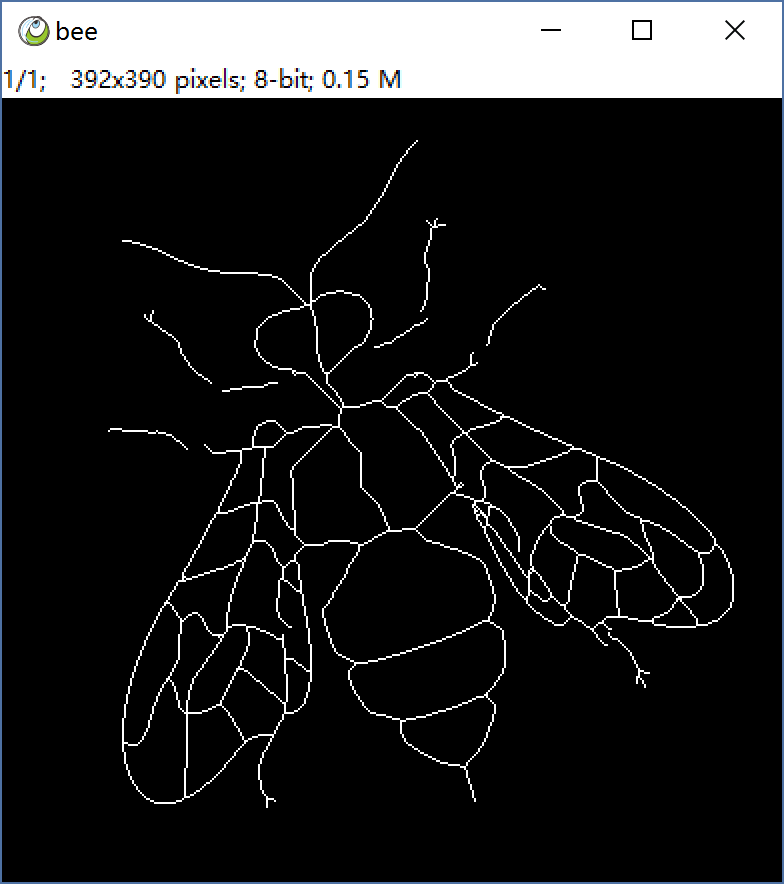
原图 凸包 叠加效果

**观察描述：**凸包像一块琥珀，将蜜蜂封住，蜜蜂的翅膀，尾针，触角等位置落在多边形上，这些点张成了这个凸包。（这里先做了一个闭运算，否则不是一个整体，会分别计算凸包）

**计算方法：**首先一个事实是，二值图像的凸包等价于轮廓的凸包，因为最终张成凸包的点，一定是轮廓上的点，所以我们只需要分析轮廓上的点就好，这样可以节省很多计算，凸包的计算并不复杂，但这属于计算几何的只是，这里不深入讨论，大概是通过对点按照横坐标排序，然后顺序扫描，判断这个点是否能把当前包张得更开。

**骨架 $ Process > Binary > Skeleton $ Chapter7 Binary-Image > Show Skeleton**

骨架是一种细化算法，是将二值图像变成单线，并保留拓扑结构的方法。所谓拓扑结构是指孔，节点的连接关系。

原图 逐层腐蚀 形成骨架

**观察描述：**骨架后的蜜蜂，依旧保留着原图的结构，所不同的是变成了单线，正像是骨头与肉的关系。

**计算方法**

细化本质上是连续的腐蚀，从外部一层一层的燃烧，但是遵循一定的规则终止。其实同样的规则我们再第五章山脊线提取时介绍过，这里再次提出。

1. 不能淹没终端像素
2. 不能将原本相联的区域分成两个孤岛。

判断上述原则是否成立，可以简单的用周围的3x3像素进行判别。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 8 | 0 | 4 | | 7 | 6 | 5 | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |

编号方式 可以去除 可以去除 末端原则 末端原则 切断原则 切断原则

**观察描述：**像腐蚀运算那样，我们从外层逐一去除像素，只是在这个过程中要根据上面原则判断像素是否可以去除，上图绿色的代表可以去除，红色的由于成为末端，或者导致切断，因而不可去除。以上情况只是举了几个具体的例子，实际一共8个邻居，每个格子有两个状态，理论上一共有2^8=256中情况，如果我们对邻居进行编号，那么判别可以通过查表来完成。

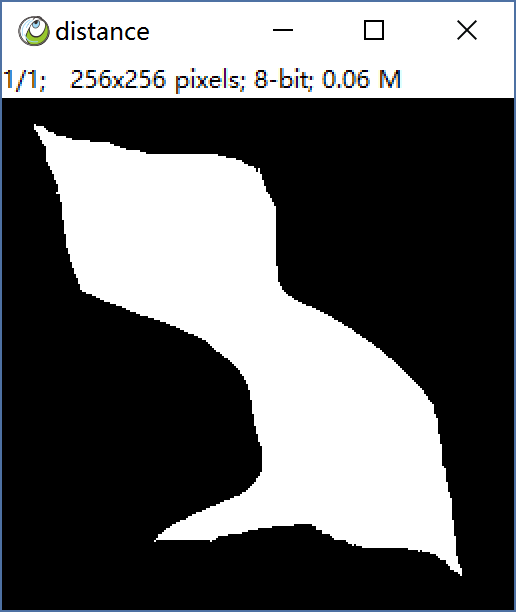
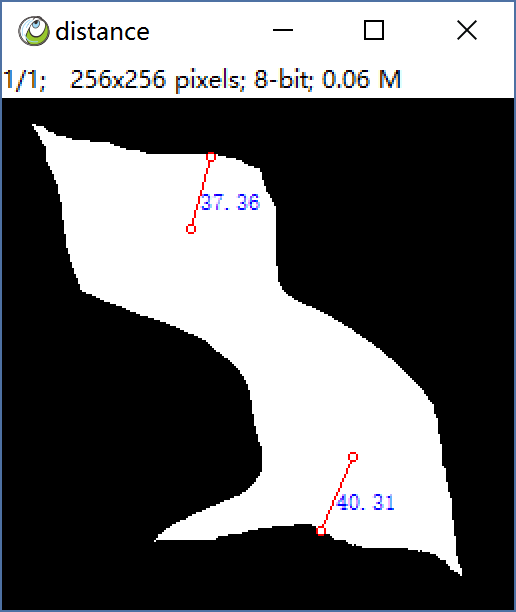
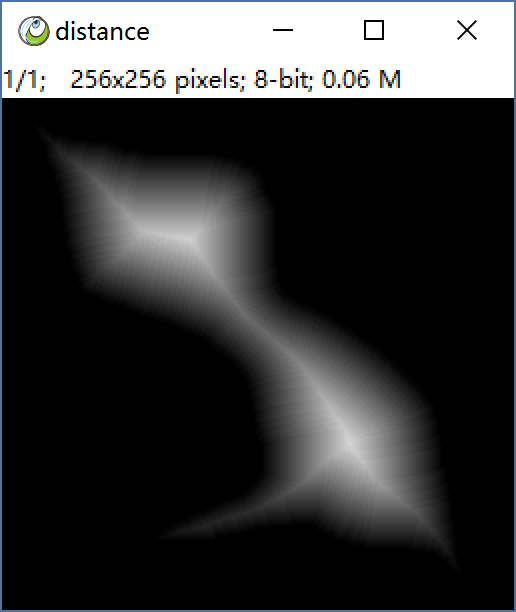
**距离变换**

截至目前，我们讨论了二值图像的线性滤波，形态学滤波，以及其他一些二值图像处理方法，而这里，我们讲解的距离变换，这一节的学习，请大家类比第五章，积极思考。

**距离变换的概念**

**$ Process > Binary > Distance Transform $ Chapter7 Binary-Image > Show Distance**

距离变换的结果是把二值图像转化为灰度图像，像素的亮度标识当前点到与之最近的背景点的距离。

二值区域 到背景的距离 距离变换

**观察描述：**二值区域，距离的意思是前景上的一点，到达背景的最短距离，用这个距离值当作该点的亮度，绘制出距离图，距离图中越亮，表示这里到达背景越远，也可以理解为这里比其他位置更丰满。

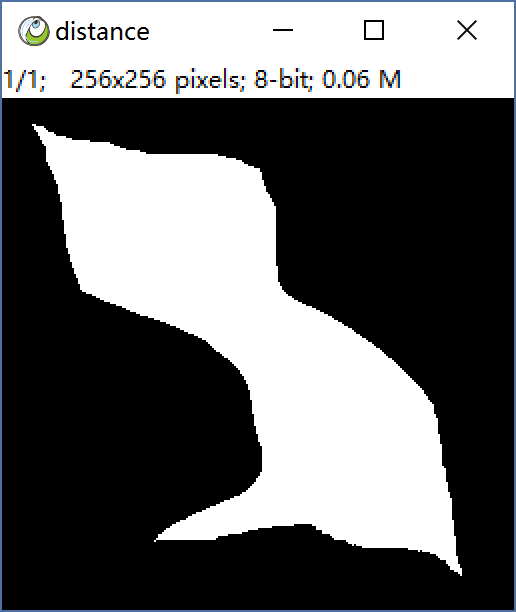
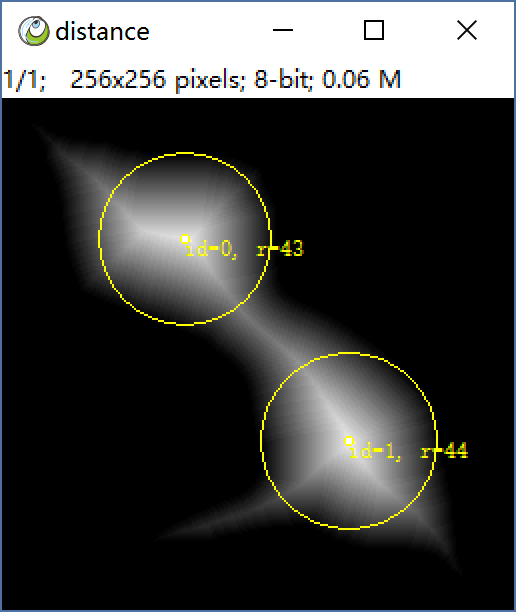
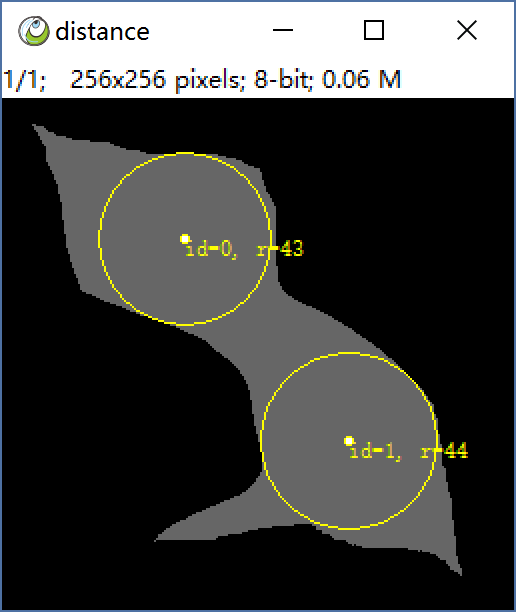
值得一提的是这样一来，二值图像变成了灰度图像，越靠近中心值越大，进而我们第五章的高程分析算法都可以使用。事实上第五章的DEM，正是用这种方式造出来的。

**图形的内切圆**

**$ Process > Hydrology > Find Maximum $ Analysis > Points Value**

**$ Chapter7 Binary-Image > Show Max Circle**

内切圆是指同时与多条边相切的圆，这里特制那种被图形卡住，动不了的，我们看图。

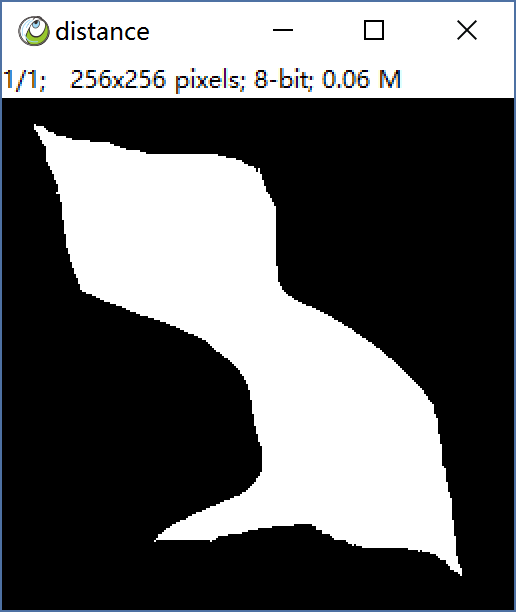
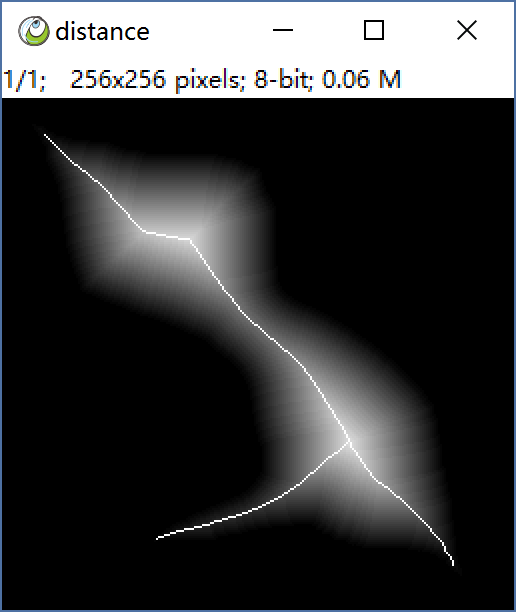
二值区域 制高点做圆 内切圆

**观察描述：**对二值区域进行距离变换，找到制高点作为圆形，像素值为半径做圆，我们可以得到图形的内切圆。想想为什么？那一点的距离本身就是指的半径，而制高点一定是同时从多个方向过来的距离相等，且无法进一步增大了。所以我们得到的就是图形的内切圆。

**中轴线的意义**

**$ Process > Binary > Medial Axis $ Chapter7 Binary-Image > Show Medial Axis**

中轴线是一种细化方法，它穿过物体的几何中心，两边相等。

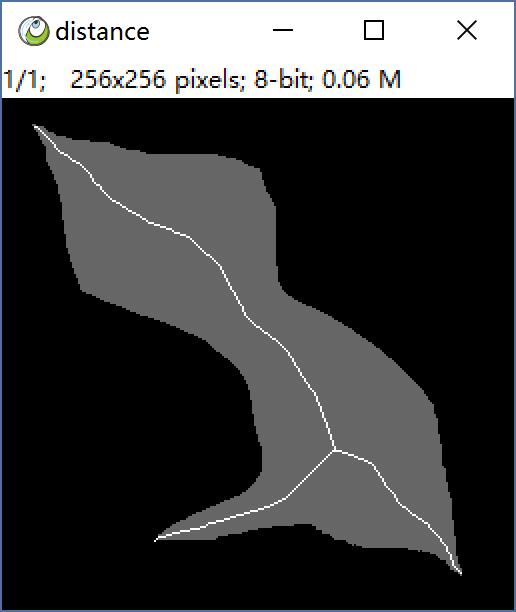
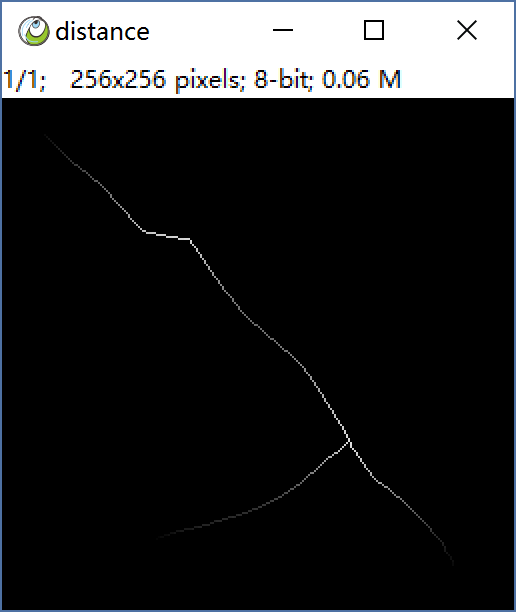
二值区域 几何中心 携带厚度信息

**观察描述：**对距离变换后的图进行山脊线提取，试想什么位置距离值最大，从一边开始，自然是远离边缘会更大，但是到了中心，两边距离一样，如果继续运动，则会靠近另一侧，所以中心也就是距离最大的地方，因而距离图的山脊线，就等价于二值图像的中轴线。

**中轴线与骨架的区别**

**$ Chapter7 Binary-Image > Show Skeleton And MidAix**

我们知道骨架和中轴线都是一种细化算法，那么二者区别在哪里？

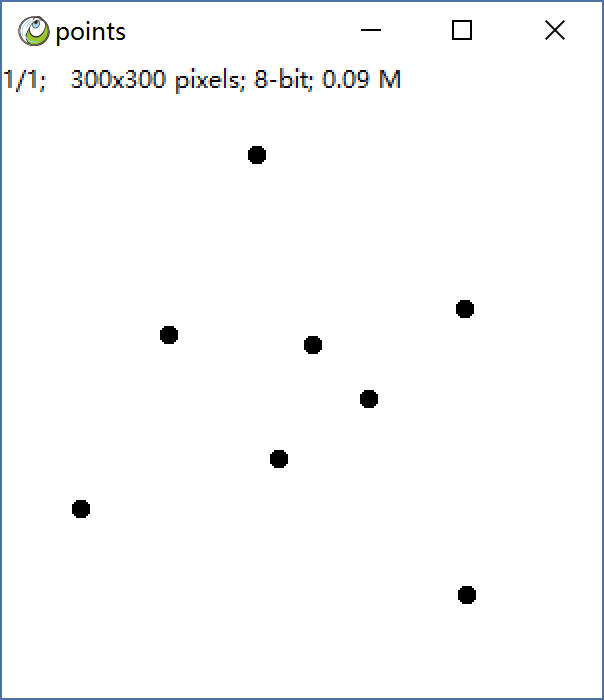
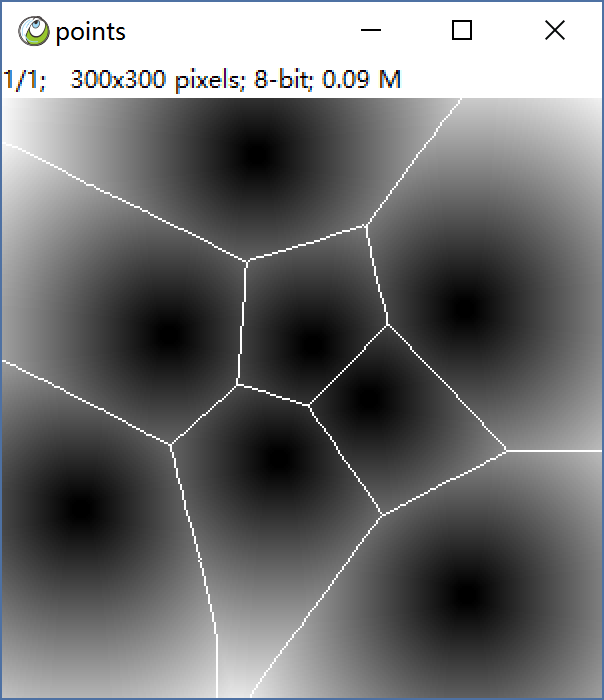
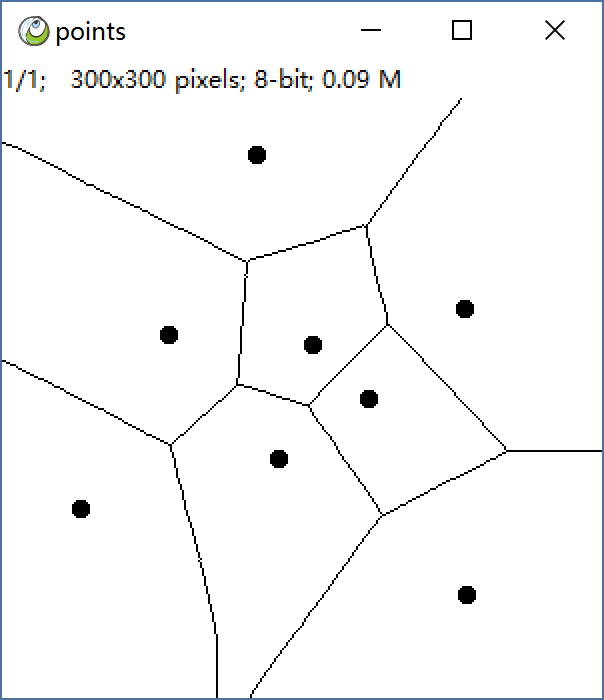
骨架 中轴线 带距离的中轴线

**观察描述：**观察三叉节点，骨架明显靠左一些，而中轴线保证了严格的几何中心性，导致这种结果的原因是，骨架算法是按照像素的逐层关系来去除的，而中轴线是严格按照距离排序进行像素去除的。这两者大致是一样的，但是又不一样，因为像素层数不等于距离。另外中轴变换还可以是带距离的，与二值结果不同，这样的中轴线，不仅告诉我们中轴线的位置，同时也告诉我们该处的粗细（内切圆半径）。

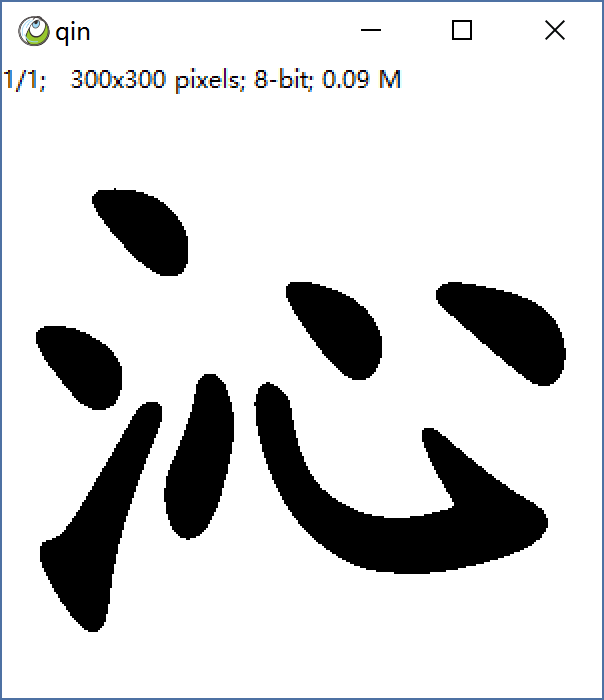
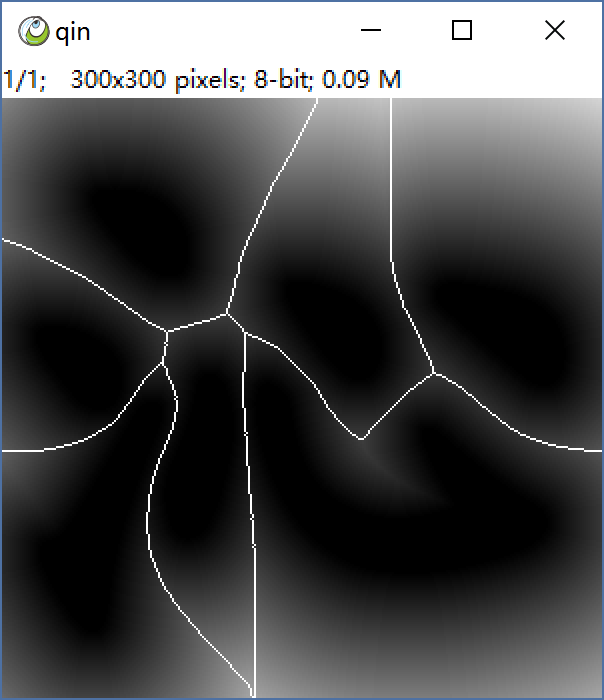
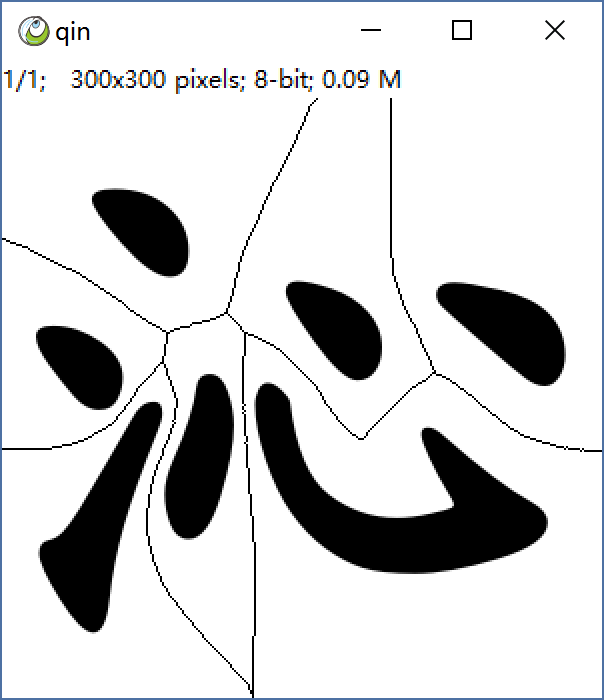
**瓜分地盘（Voronoi图）**

**$ Process > Binary > Binary Voronoi $ Chapter7 Binary-Image > Show Voronoi**

有这样一类问题，在一个地区，有若干邮局，而每个人需要根据自己的位置，选择最近的邮局，这就是著名的邮局问题。但是我觉得，把邮局当作主体，称之为瓜分地盘问题更贴切。

散点 距离变换+分水岭 叠加效果

汉字 距离变换+分水岭 叠加效果

**观察描述：**我们知道，对二值图像做距离变换，最大值出现在中间，所以以黑色区域位种子，进行涨水，最终结果就是分界线。众所周知，对于点来说，中垂线就是等距分割线，因而散点图的瓜分结果是直线段构成的，而对于复杂区域，是复杂曲线构成的。

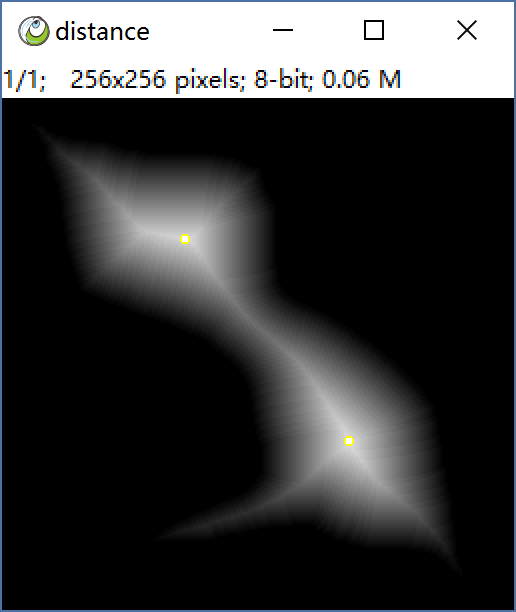
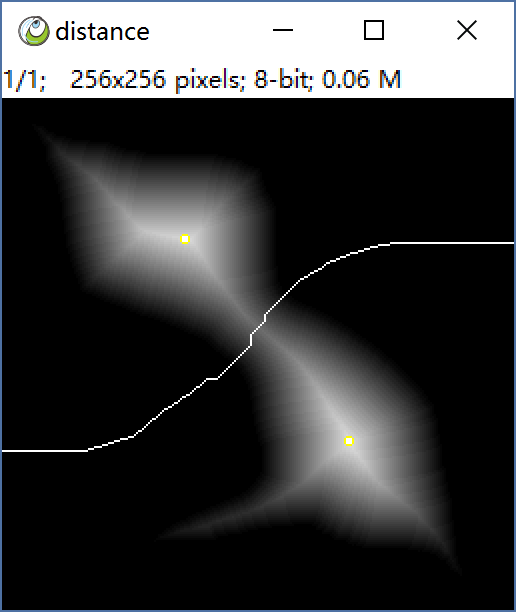
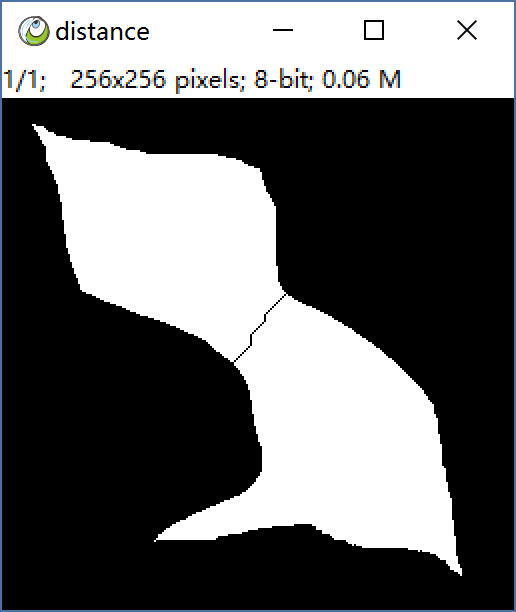
这类问题的算法在计算几何中成为Voronoi图，有相关的矢量算法，而我们这里介绍的是图像算法实现的。

**对偶性：**瓜分地盘问题某种意义上和中轴线是对偶问题，因为我们的分割线，正好是白色部分的中轴线，而所谓的瓜分，也的确是把前景区域从中轴线分开。

**二值图像的分割**

**$ Process > Binary > Binary Watershed $ Chapter7 Binary-Image > Show Binary Watershed**

二值图像没有灰度信息，因而所谓分割只能根据形态。最常用的是从比较细的地方分开。

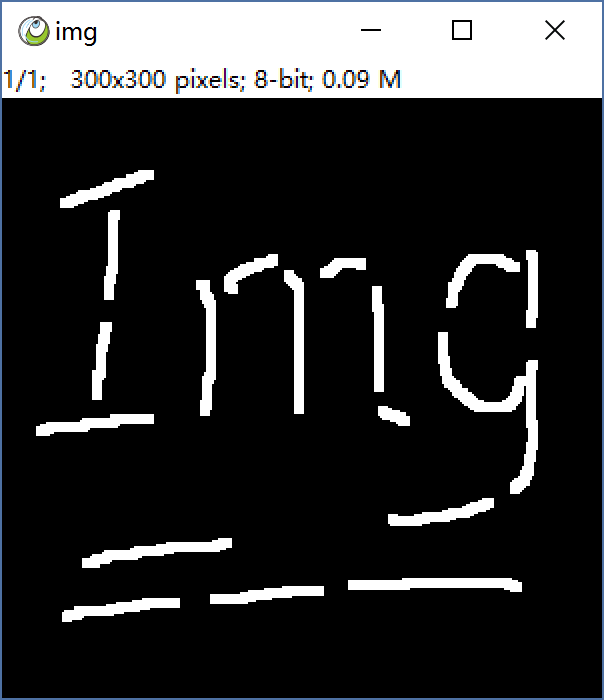
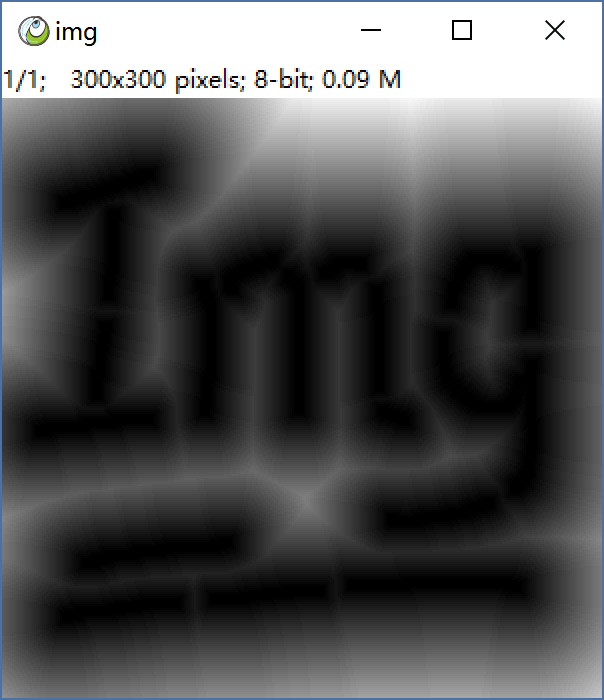
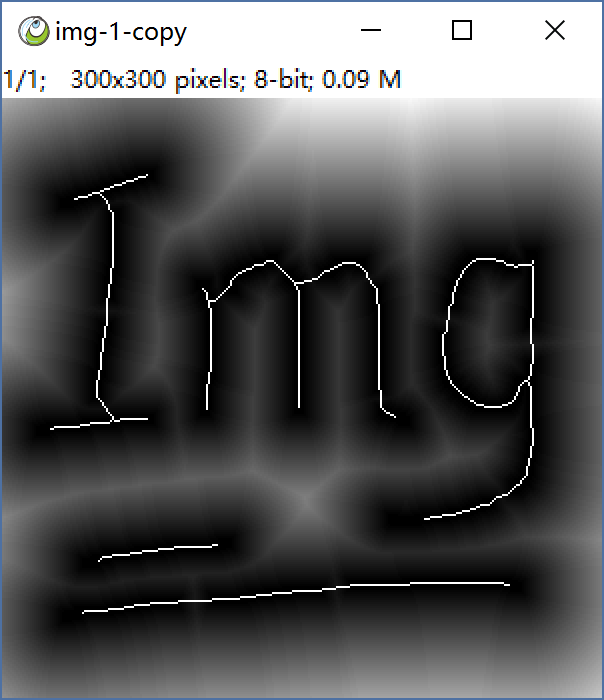
找峰值 分水岭 用分水线切割

**观察描述：**经过距离变换，越厚的地方值越大，而制高点基本处于某一个区域中心，在比较细的地方值比较小，用制高点做种子，进行分水岭，则分水线是通过图形比较细的部位，从而实现分割。这种方法能够把目标从凹的区域划断，因而可以用来处理一些突目标的粘连问题。

**修复断断续续的线**

**$ Process > Hydrology > Find Ridge $ Chapter7 Binary-Image > Show Repair Lines**

有时候我们希望把比较近的断开的线连起来，我们之前介绍过闭运算可以实现类似功能，但是形态学滤波结果的可控性不是很强。这里我们介绍通过距离变换和谷线提取的方法实现。

断断续续 反向做距离变换 提取山谷线

**观察描述：**原图一个断断续续的Img，取反，进行距离变换后，文字是黑色，周围是暗色，越远越亮，我们对原图进行谷线提取，于是我们得到了连续的Img.

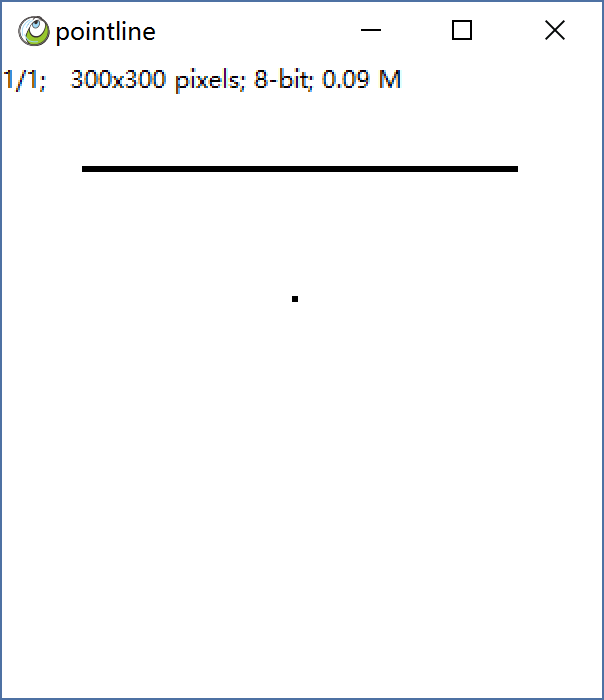
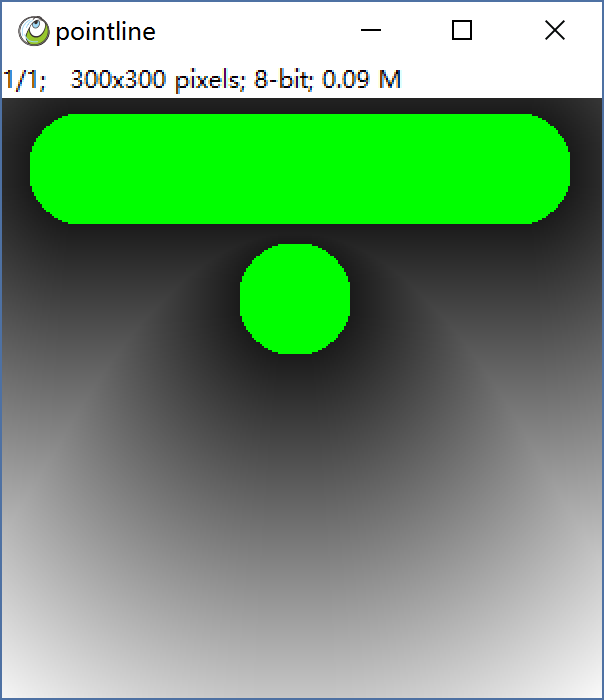
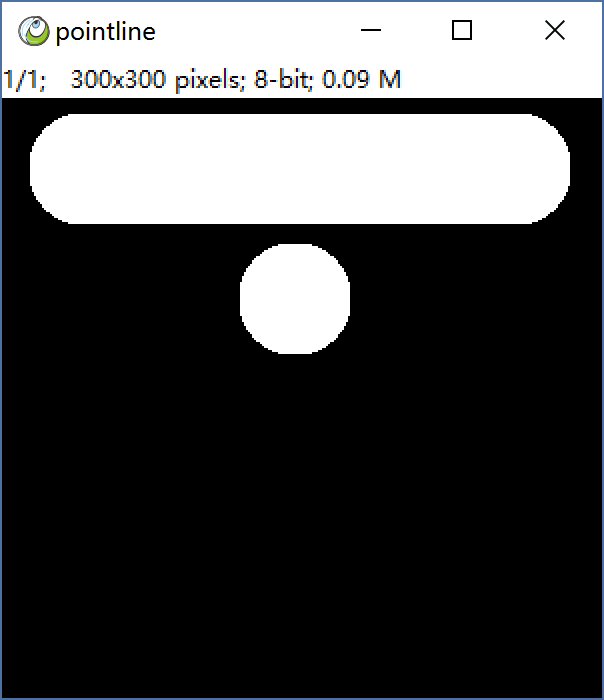
**对偶性：**这种前景修补方法，某种意义上与二值图像分割有对偶关系，一个简单的道理，对前景的修补，就是对背景的分割。我们把前景小的缝隙进行修补，也就是把背景细的地方划开。

**用距离变换做圆锥曲线**

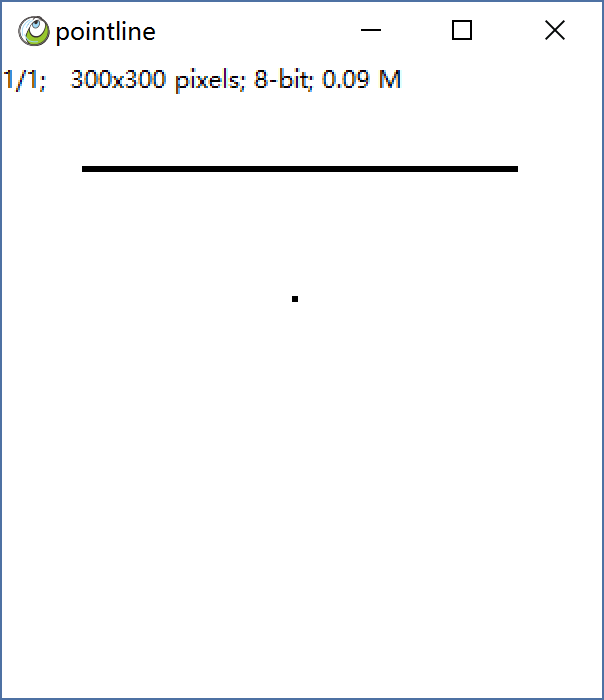
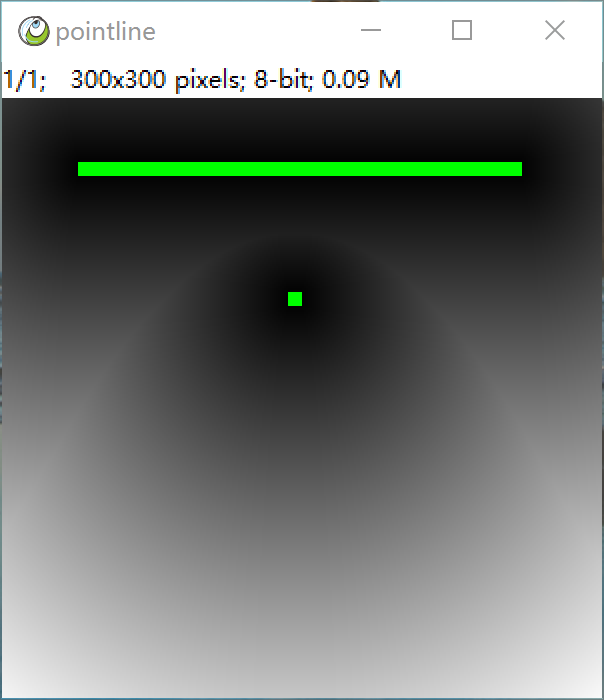
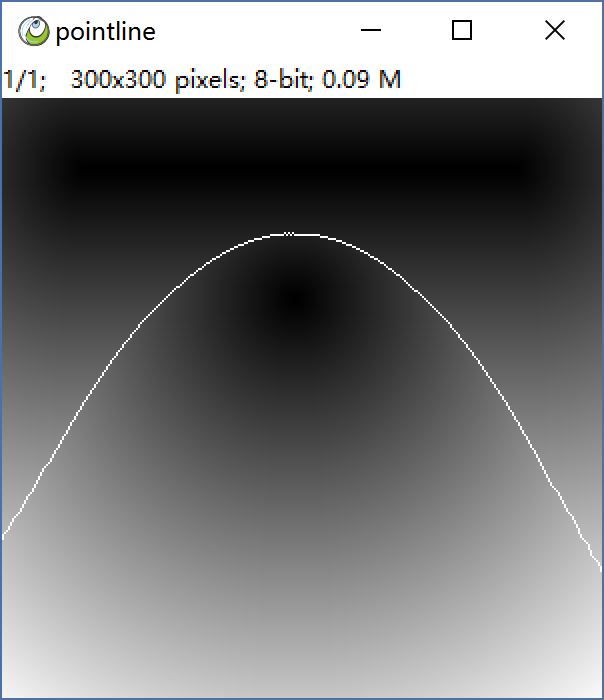
介绍了这么多有关距离变换的应用，最后我们看一个趣味性的问题，我们有没有办法做一些简单的几何图形，比如，圆，抛物线，椭圆。

**$ Chapter7 Binary-Image > Show Buffer By Distance**

**$ Chapter7 Binary-Image > Show Make Parabola**

点线图 距离变换并阈值 得到圆形和圆角线条

点线图 距离变换并分水岭 得到抛物线

**观察描述：**我们用距离变换之后，进行阈值处理，原先的线变成了有厚度的圆角长条，而之前点变成了一个圆形区域，这个结果相当于用一个圆形的笔迹对图像做了膨胀。另外，我们对距离变换结果进行分水岭，也就是之前我们讨论的瓜分地盘问题，得到了一条抛物线。想想抛物线的定义，到点与直线距离相等的点的集合，因此其分割线就是抛物线。

**思考：**那么能否做出椭圆呢？请大家自己动手尝试。提示以下，椭圆的定义是到亮点距离和相等的点的集合，既然是到两个点的距离和，那么试想我们是不是需要两张图？

**区域的特征**

**$ Analysis > Region Analysis > Geometry Analysis**

**$ Analysis > Region Analysis > Geometry Filter**

至此我们应该已经的对二值图像有了一定的认识，而其中联通的像素称之为一个区域，某种意义上你可以把区域理解成一个几何图形。本节我们来讨论区域的一些特征。

**四邻域和八邻域**

我们本节讲的区域是图像上联通的像素，那么一个关键问题就是如何定义联通？这看起来比较简单，不过有个细节问题，看下图，右边两种情况，橙色和绿色区域是否联通？

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |

四邻域 八邻域 一个对角接触 多个对角接触

**观察描述：**四邻域是用上下左右来定义联通，既一个前景点上下左右如果能触及另一个前景点，就判定联通。而八邻域则是用上下左右，以及对角线元素，一共八个元素进行判定。那么我们看后面的两个图，图三绿色和撑死区域只有一个像素以对角线方式触碰，因此，以四邻域定义，他们是不联通的，而以八邻域定义，他们是联通的。图四虽然多了几组对角元素，但是以四邻域定义，它依旧不联通。

**小技巧：**如果对单线（比如骨架或中轴线）做区域分析，那么请实用八邻域，否则你的目标会被切得七零八碎。相反，如果是对分水岭的结果进行区域分析，你需要实用四邻域，否则全部碎片将会联通起来。

**质心，面积，周长 $ Chapter7 Binary-Image > Show Base Analysis**

这几个是几何图形最基础的概念，其计算方法也非常简单，面积是像素的个数，质心就是像素坐标的平均，值得一提的是周长的概念，惆怅简单的说可以理解成轮廓像素的个数，然而实际上横向，纵向移动一个像素长度是1，而沿着对角线移动是根号2，ImagePy中的周长是考录了对角线的严格周长。



蜘蛛：l = 1907 s = 11135 蝎子：l = 1283.2 s = 8694 甲虫：l = 951.3 s = 11561

**观察描述：**蜘蛛的周长最大，甲虫的面积最大，蝎子虽然面积最小，但是周长缺比甲虫还要大。

**拓扑 $ Chapter7 Binary-Image > Show Holes Count**

拓扑是描述连接，包含，闭合关系的科学，我们这里主要描述集区域内的镂空。

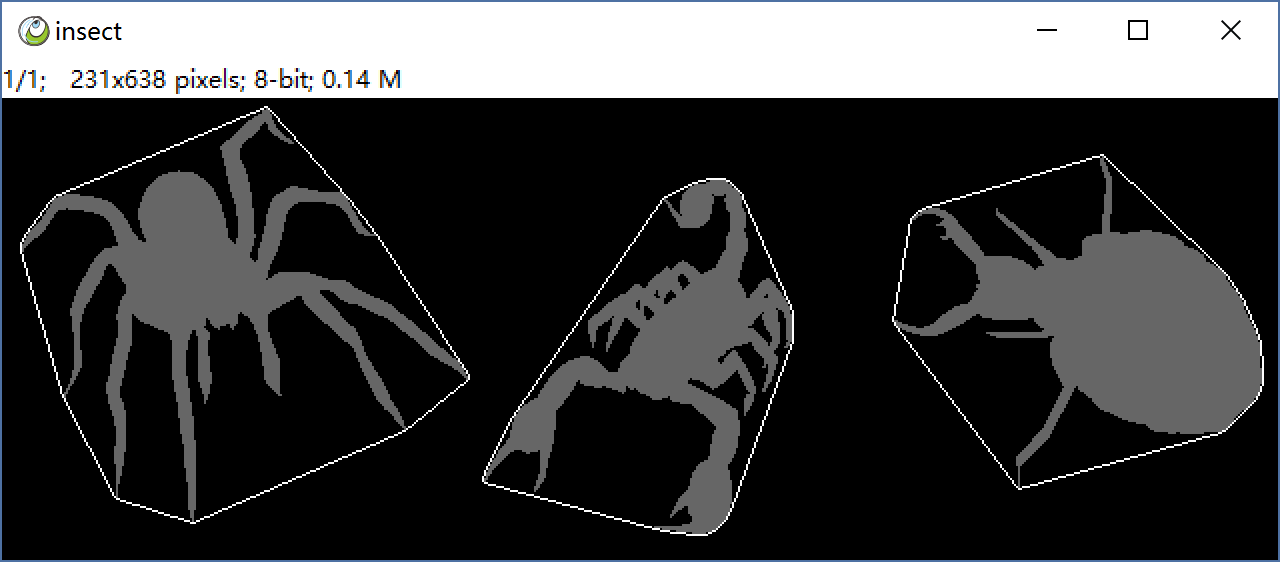


蜘蛛：无镂空 蝎子：7个镂空 甲虫无镂空

**观察描述：**蝎子的腿比较短，且弯曲，因而形成了很多镂空区域，拓扑分析可以计算出内部有多少镂空。

**饱满度 $ Chapter7 Binary-Image > Show Solidity**

饱满度不是一个很官方很严格的定义，这里定义为自身面积占据凸包面积的比例。简单说，就是越接近实心的，越饱满，张开的面积越大，越不饱满。圆的饱满度是1。



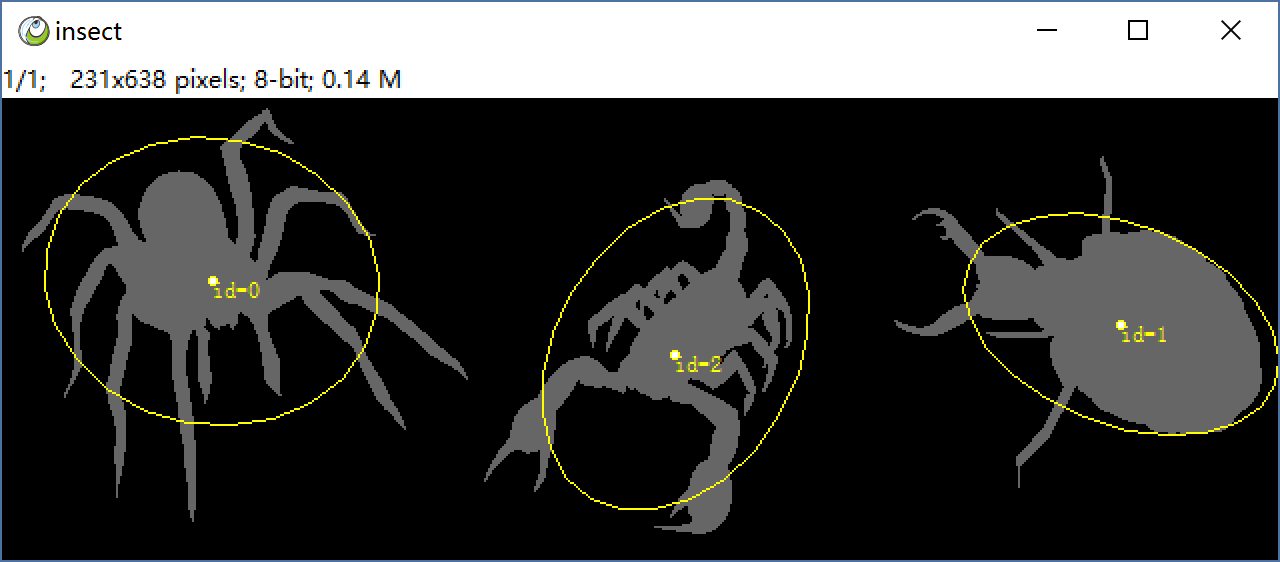
蜘蛛：0.33 蝎子：0.46 甲虫0.52

**观察描述：**蜘蛛由于腿特别长，因而张开的凸包很大，而自身面积有限，而蝎子的钳子，毒针也张成了一定面积，但并没有蜘蛛那么大，甲虫腿短身体厚重，因而饱满度最高。

**小知识：**大自然针对很奇妙，饱满一定程度上意味着防御力强，代价是移动缓慢，而不饱满往往动作敏捷，但防御力就要打折。蜘蛛向来是恶魔的化身，但遇到攻守更加平衡的蝎子，自己柔弱的身体往往禁不起蝎子的毒针，而蝎子纵使凶猛，但是也拿一只甲虫没有办法。这可以理解为生物形态学，想想水生植物的叶子，是不是往往更饱满一些，并且表面似乎有油？

**等效直径和等效椭圆 $ Chapter7 Binary-Image > Show Circle And Ellipse**

很多时候会遇到求区域的直径的问题，我们知道只有圆形才有直径，因而会对区域求等校圆，既质心重合，面积相等的圆。因而等效直径其实可以用面积计算出来，r = (s/pi)^0.5。这个很简单，我们不做介绍，这里说一下等效椭圆。等效椭圆是指质心重合，且在长短轴方向上有相同的方差投影。这个概念比较数学，我们直观的来感受就好。



蜘蛛：长轴:168.5 短轴:143.5 蝎子：长轴:167.0 短轴:120.4 甲虫：长轴:164.8 短轴:100.1

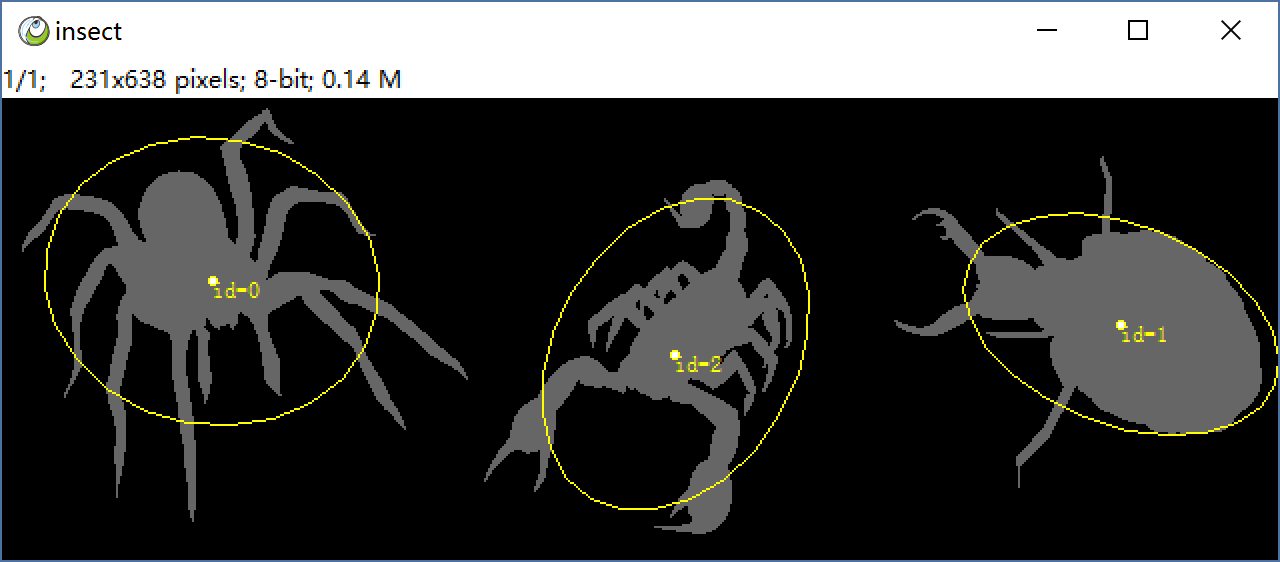
**观察描述：**我们看到椭圆中心落在区域质心，而椭圆基本上可以概括区域在两个方向上的纵深程度。与直观感受相似，蜘蛛的椭圆很接近于圆。

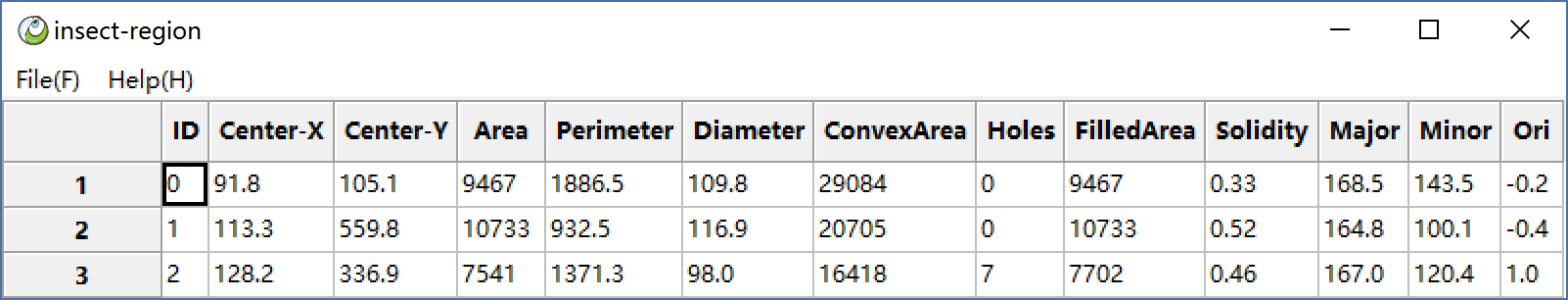
**派生指标：**有了等效椭圆，我们可以定义长轴的方向为区域方向，也可以定义区域的离心率，用长短轴之比，我们知道圆形的离心率为1，而目标越狭长，越接近于椭圆，离心率就越大。

**数学知识：**等效椭圆的求解，需要对区域的点求协方差矩阵，进而对协方差矩阵进行特征分解，得到两个特征值和两个特征向量。就是椭圆的长短轴，以及长度，这部分知识我们在Harris角点检测器中其实用到过，不过同样没有详细介绍，这里仍然不会介绍，不过可以反过来想想，梯度向量的分布能够张开成一个圆，那么就可以判定这个点为角点，对于边缘来说，只有一个方向的梯度向量，因而得到的等效椭圆很狭长，这样就可以感性理解为什么这两部分用了相同的数学知识了。

**属性计算与过滤 $ Chapter7 Binary-Image > Show Region Analysis**

在理解了概念的基础上，我们来计算这些指标。



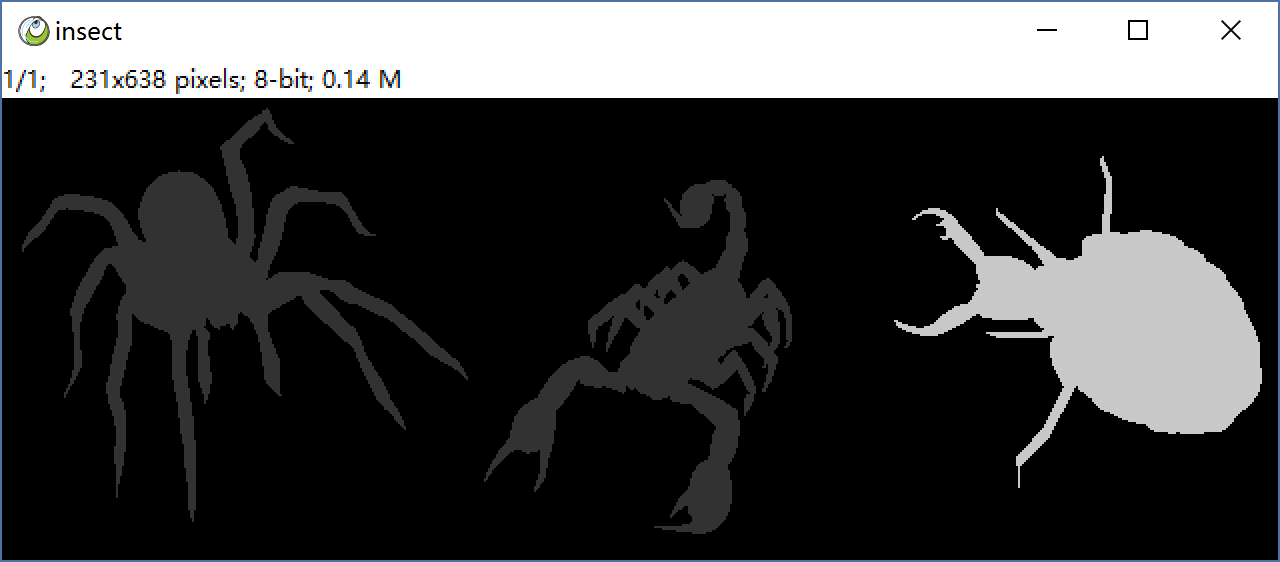
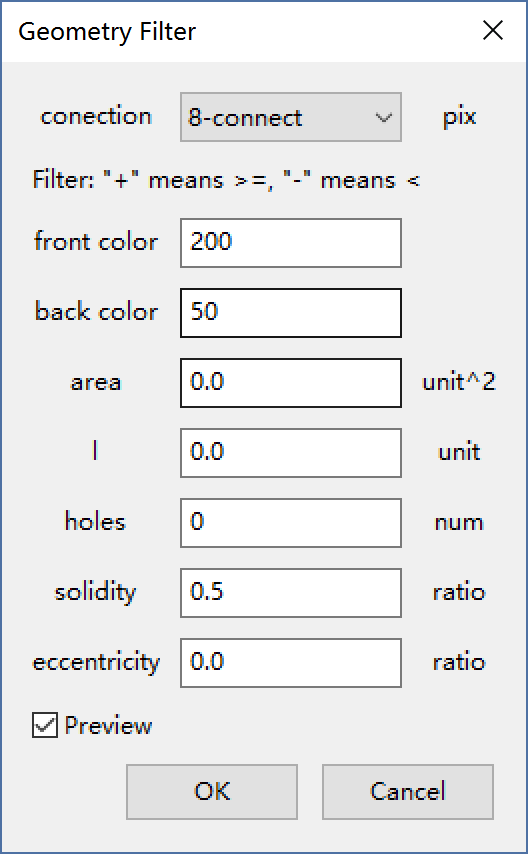


根据本节的知识，可以对照结果，体验并分析各个指标，加深认识。

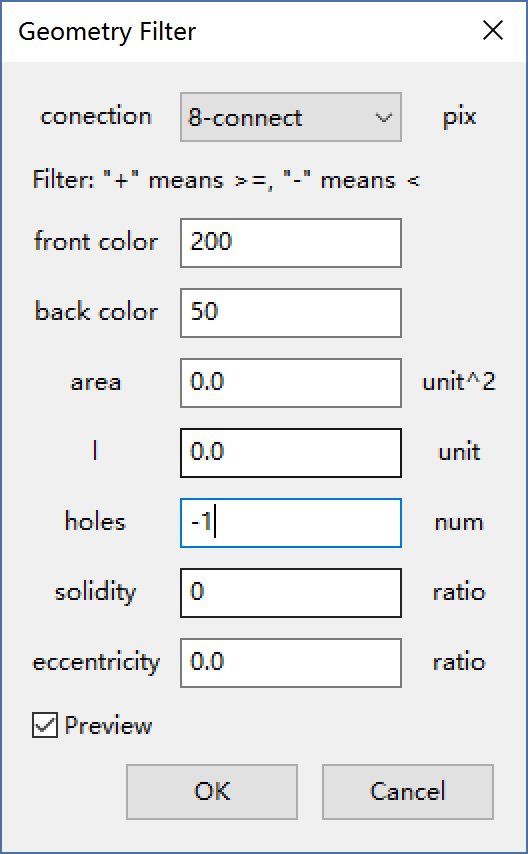
**$ Chapter7 Binary-Image > Show Region Filter**

最后我们来根据指标对区域进行一个筛选。填写正数表示选择大于等于该值的，负数代表小于该绝对值的。零表示不做限制，可以使用各种条件组合。如果通过所有约束，颜色设置为front ground，否则设置为back color。

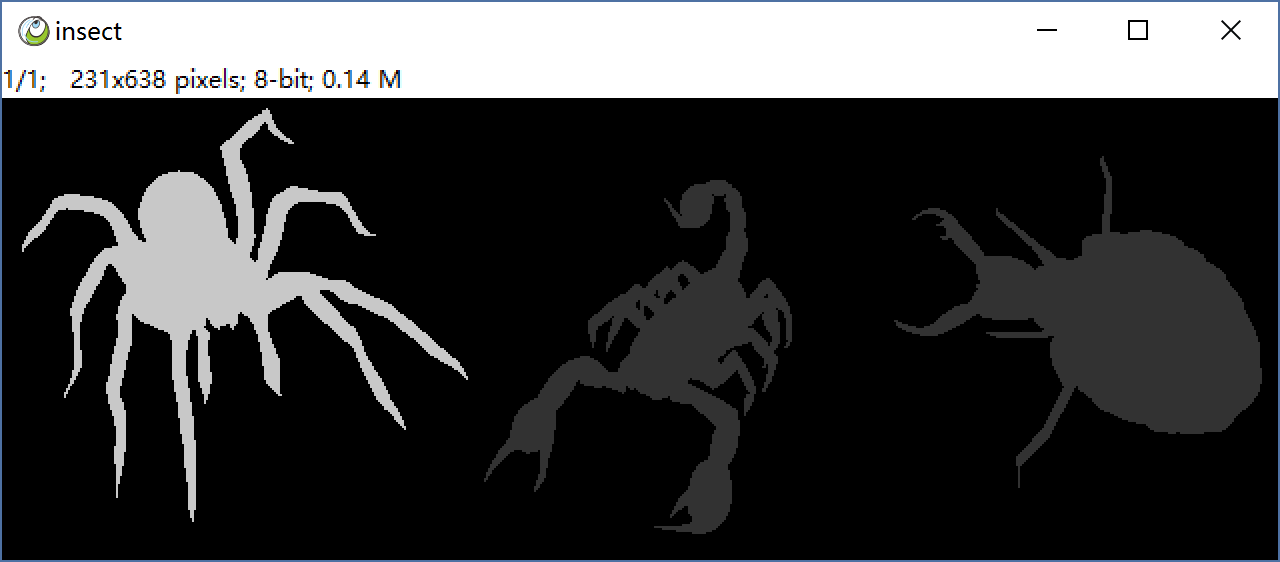
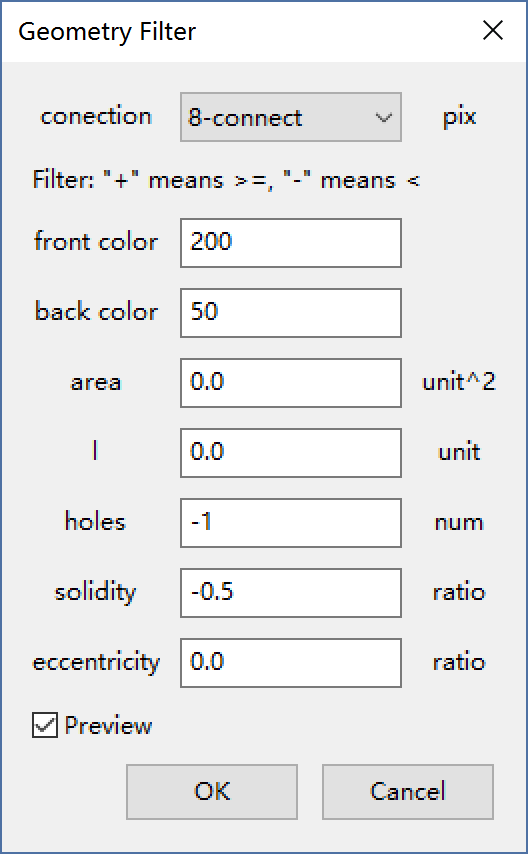
选取饱满度大于0.5的（solidity填0.5）

选取不带孔的区域（holes填 -1，表示镂空数小于1的区域）

选取饱满度不足0.5且没有孔的区域

**其他**

除了以上介绍的区域特征，还有很多，比如高阶矩，不变矩等，那些知识需要一定的数学基础才可以理解，不过本节介绍的是一些最基本，最实用的知识，掌握了这些，足以处理很多常见的问题，而更高阶段的学习，有兴趣的读者请自己查阅资料。

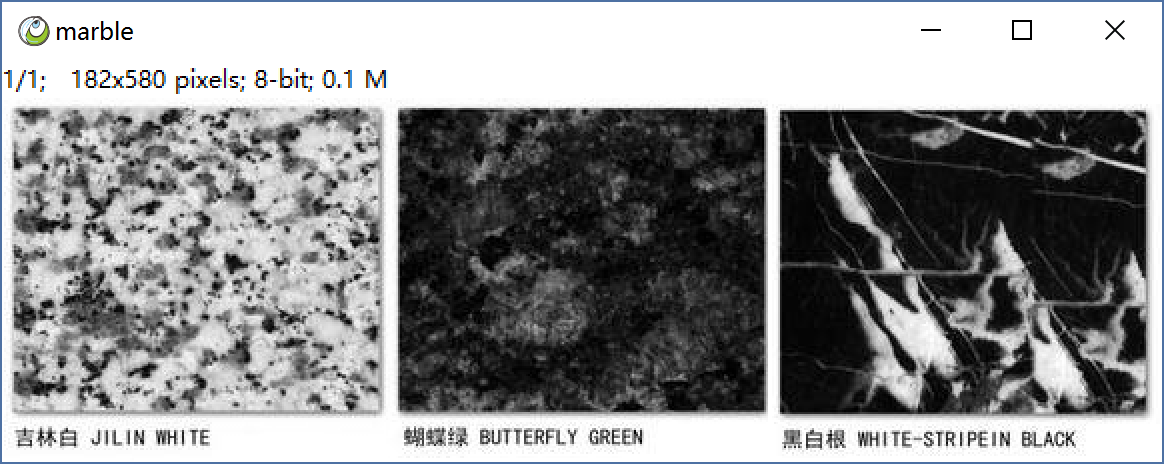
**对偶性**

我们已经学习通过几何特性对区域进行描述与筛选，假如问题是要填充一些满足一定特性的空隙呢？这个问题可以再次借助对偶思想解决，首先对图像进行求反，孔就成了区域，利用区域过滤方法过滤，得到结果后再次求反，就把区域过滤用在了填充空隙上了。

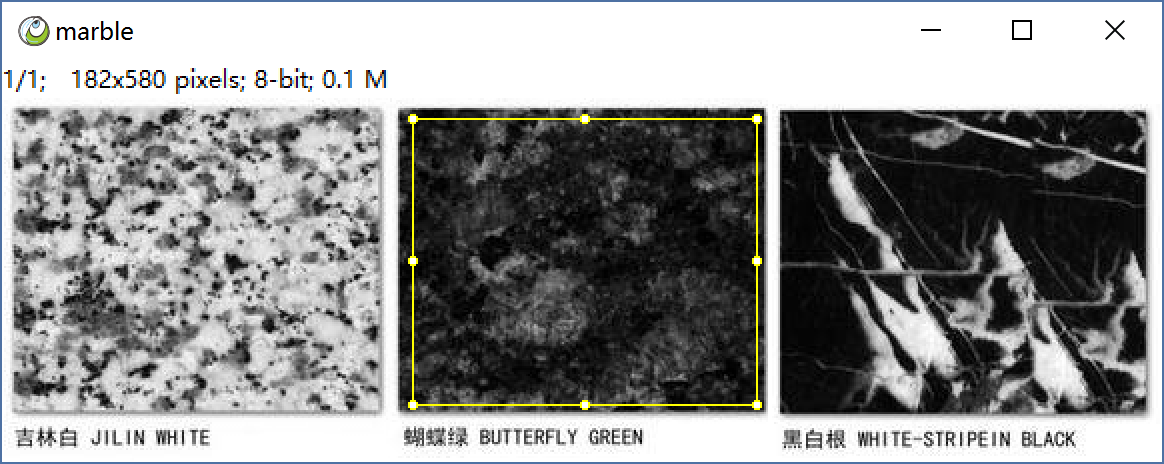
**标记与密度测量**

**像素统计 $ Analysis > Pixcel Statistic $ Chapter7 Binary-Image > Show Global Statistic**

我们在第三章学习了像素直方图，就是把每个像素进行频率统计，得到一个直方图，这里我们进行更深入的学习，在此基础上计算像素的最大值，最小值，平均值，方差，标准差。这些概念大家在数理统计中都接触过，最大值，最小值很容易理解，平均值反应了整体亮度，而方差反应内部变化，标准差时方差的开根号。



最小值：0 最大值：255 平均值：115.63 方差：8544.88， 标准差：92.44



最小值：0 最大值：176 平均值：37.22 方差：689.9 标准差：26.27

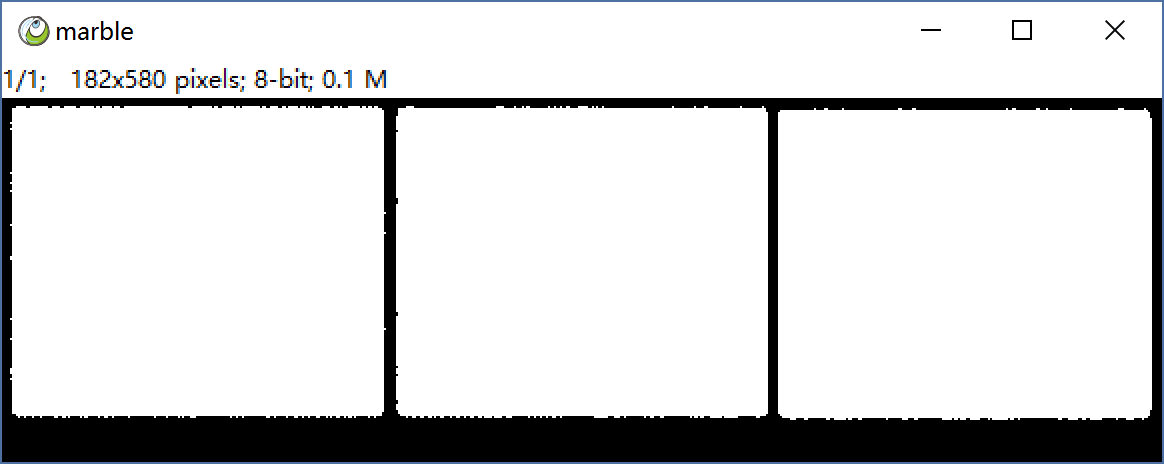
**观察描述：**整张图进行像素统计，图中内容复杂，黑白分布都很多，因而平均值适中，方差较大。而当我们选定中间的石头再次统计，就会得到选定区域的统计信息。选中的石头比较黑，内部纹理稀少。因而反映在统计结果就是最大值只有176，平均值37.2，标准差也只有26.27。

**分区统计**

现在我们有一个问题，如果要对每一块样本进行像素统计，该怎么做？当然一个方法时我们自己用选区工具框选，然后再进行统计。这样固然可行，但是如果目标很多的时候，工作量巨大，况且并非所有目标都是矩形的。所以我们要借助掩模的概念来处理。

**何为掩模？ $ Chapter7 Binary-Image > Show Mask**

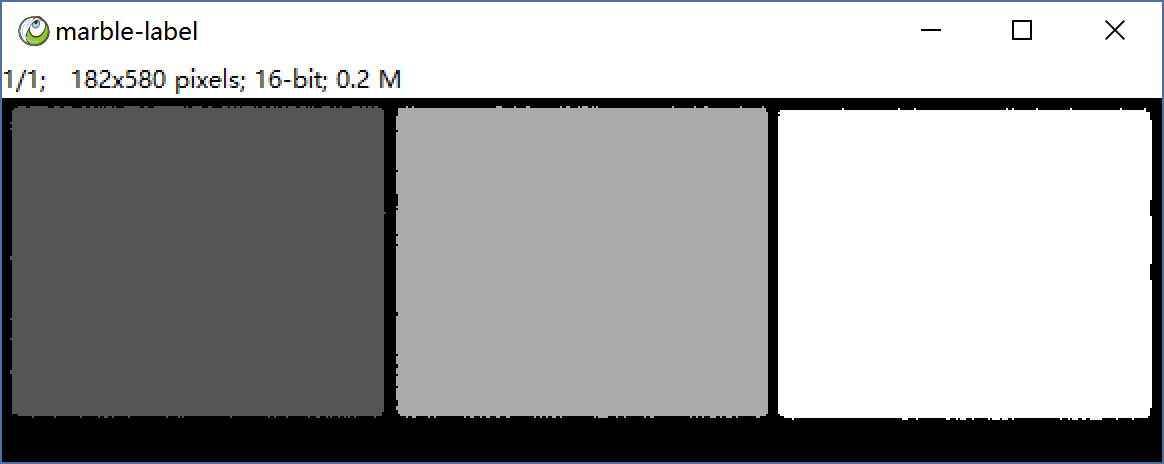
掩模其实就是一个二值图像，只是它的作用是去遮罩另一幅图像，把要测量的部分用一个个联通区域表示，我们看下图。



**观察描述：**掩模的每个区域对应待统计图像的一个目标。

**区域标记 $ Chapter7 Binary-Image > Show Label Mask**

这里要提到一个重要的过程叫区域标记，所谓区域标记，就是把联通的区域标记成唯一的值，背景像素为0。

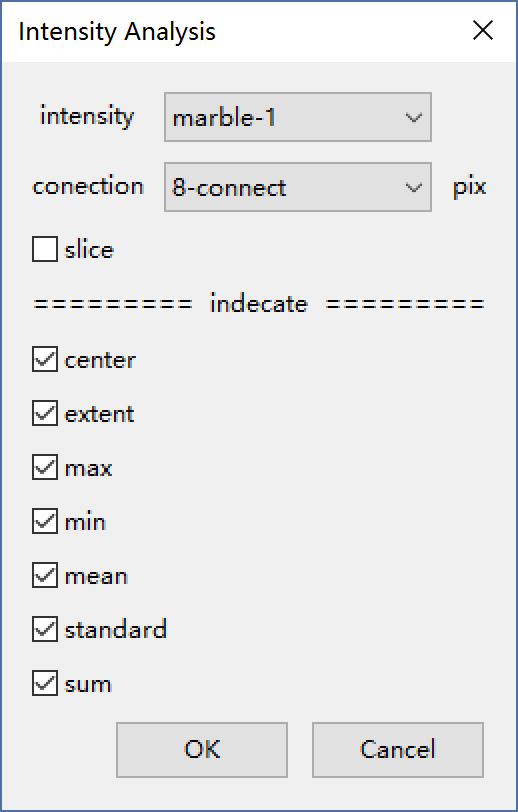


**观察描述：**我们看到背景像素为0，而每个区域被标记成唯一的值。

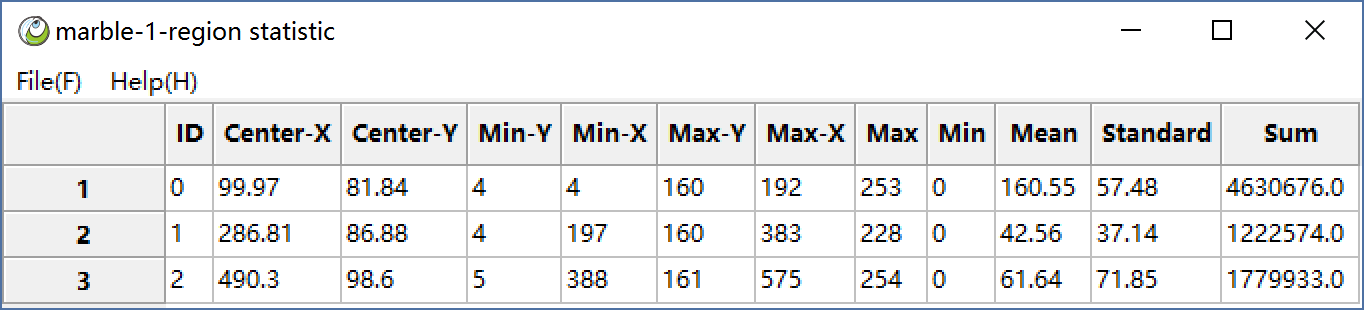
**掩模是如何起作用的：**这样一来，我们就可以通过标记值得到灰度图像对应像素所属的区域，以求和为例，我们对区域进行标记后，得知有三个区域，于是构建一个长度为3的数组[0, 0, 0]，而遍历标记图像，假设像素值为1，对应待测图的像素为80，则把像素值加给数组的对应位置，成了[80, 0, 0]，这样当完成像素的遍历，我们就得到了各个区域的像素和。（一个形象的形容就是我们有n个投票箱，待测量图像进行投票，对应的标记图像指明这一票投给第几个投票箱，0代表弃权，最后对投票箱进行汇总，得到和，其他统计量也可以用类似方法获得）

**值得一提的是，标记这项工作虽然很普遍，不过我们一般不需要主动操作，因为它已经被集成到一些功能的内部了。**

**分区密度统计 $ Chapter7 Binary-Image > Show Intensity Analysis**

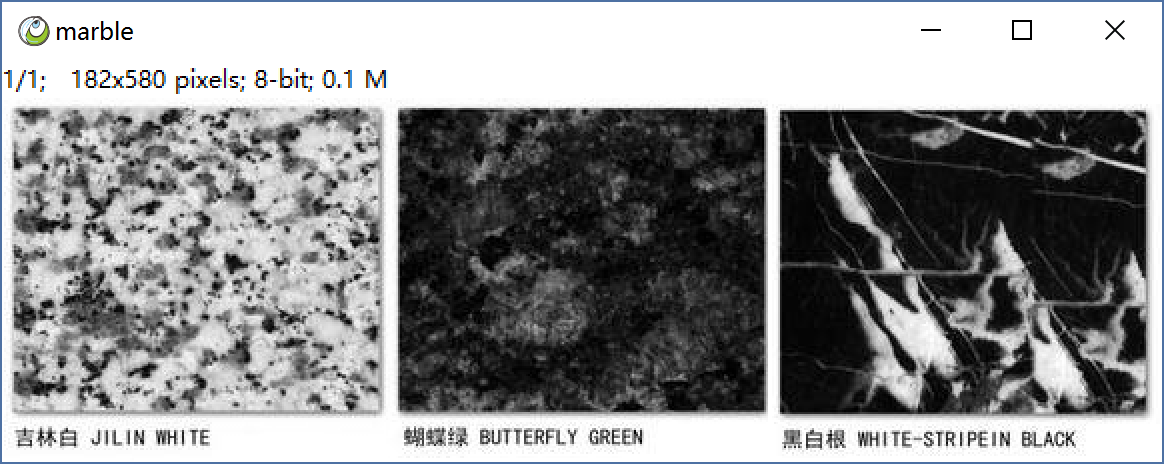
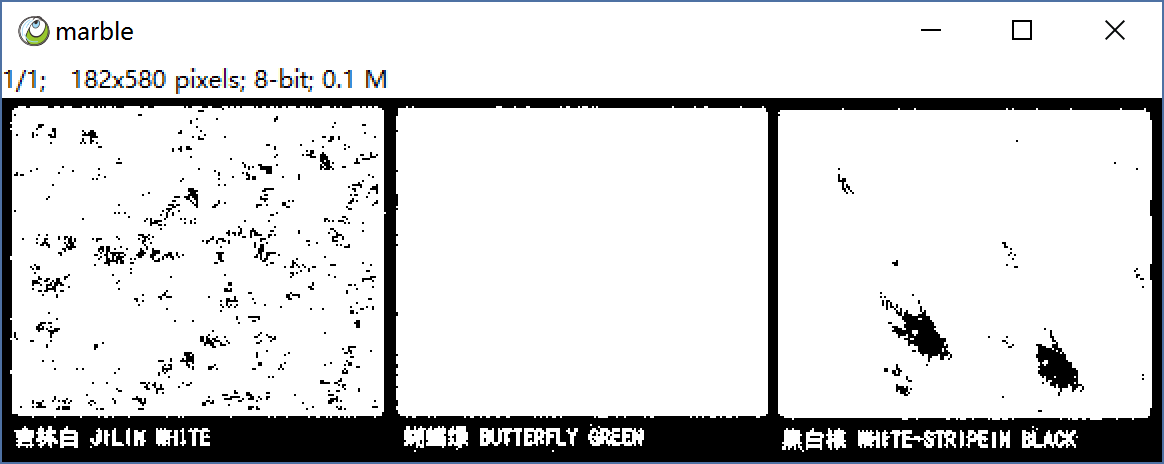
利用掩模对图像进行分区统计

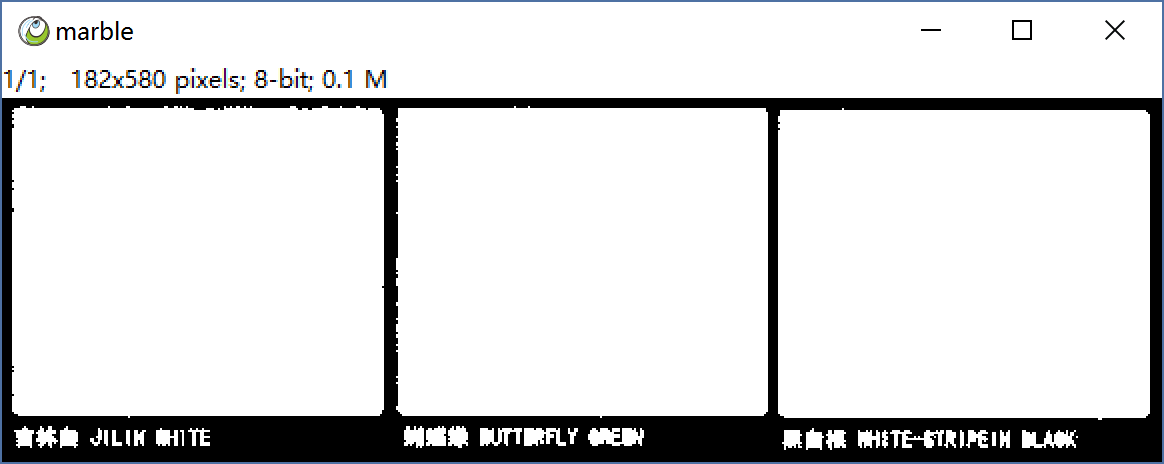
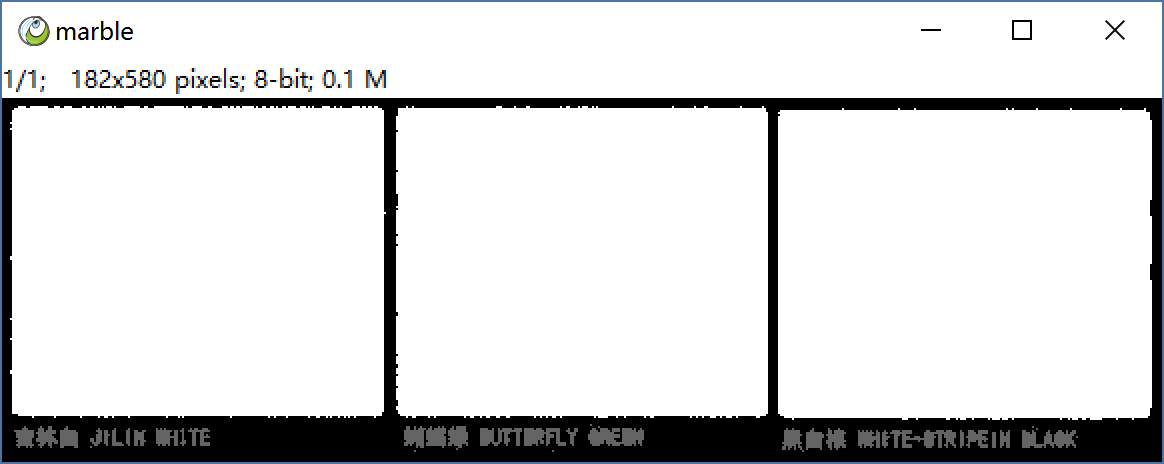


分区统计，得到了三个区域的统计信息，注意这里除了之前介绍的统计量之外，还有外接矩形和质心，我们观察第三块大理石质心的标记明显不在矩形中心，明显偏下，这是因为灰度图像的质心加权了像素值，白色纹理在右下方集中一些，因而质心被拉向右下方。

**如何得到掩模 $ Chapter7 Binary-Image > Show How To Get Mask**

我们知道只要获得掩模，就可以对图像进行分区域测量，但是目前为止我们还没有讨论掩模是如何产生的。有些时候掩模是给定的，但很多时候我们需要自己制作，其中一个方法是构造三个选区，然后进行填充，但当图像比较复杂，目标很多时，自己制作是一项昂贵的工作，一个经济的选择是我们通过原图的二值化，以及适当的处理得到。

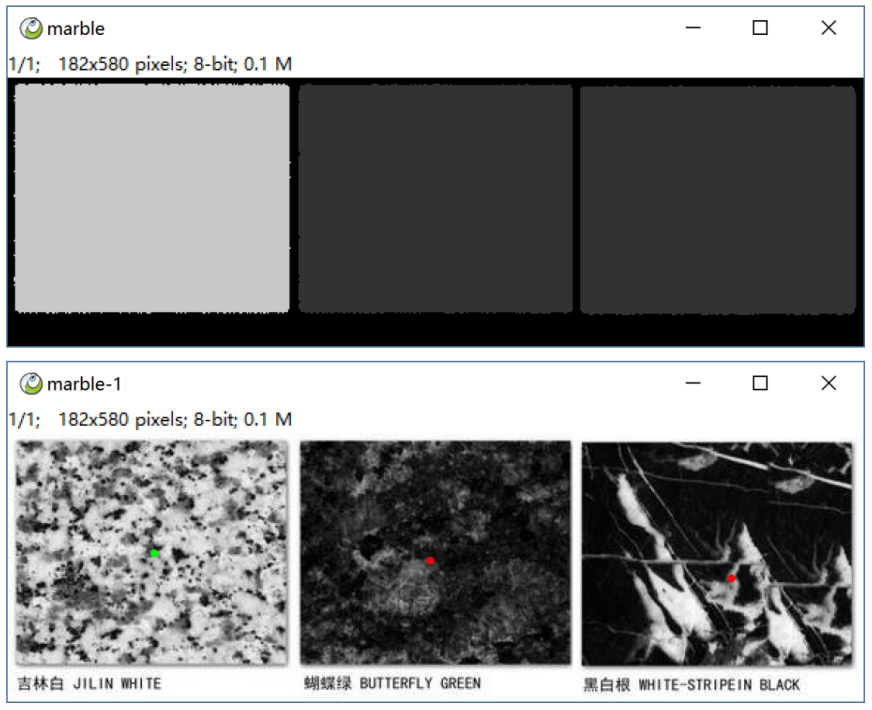
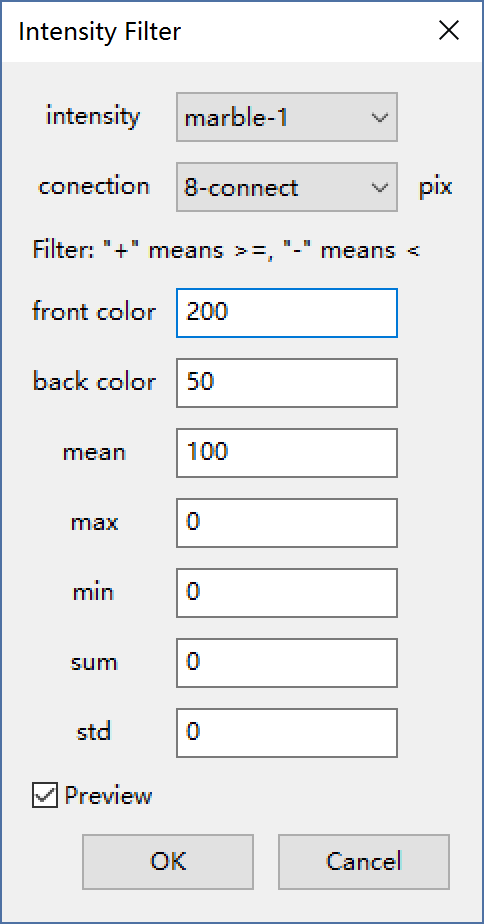
1. 原图 2）阈值化 3）填充空隙 4）区域过滤，去掉小面积区域。

**注意：**由于我们需要用掩模对原图进行分区域统计，因而最好将原图复制一份用于制作掩模。

**通过统计信息过滤 $ Analysis > Region Analysis > Intensity Filter**

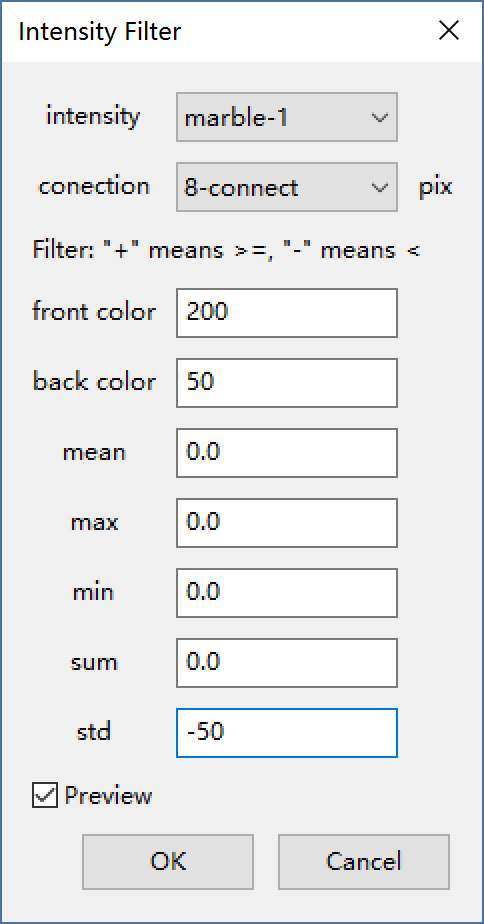
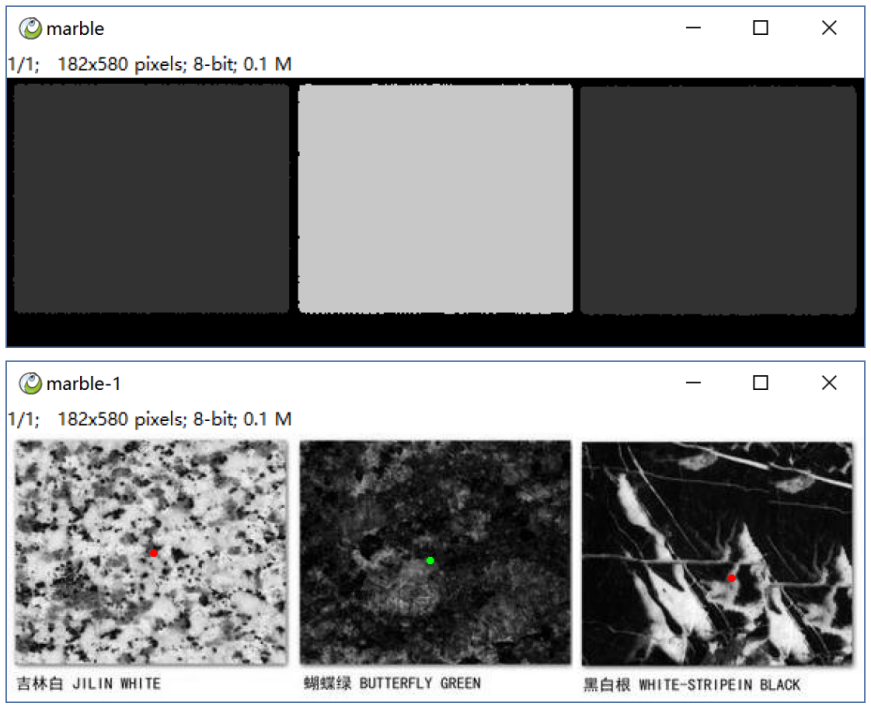
我们学习了通过掩模对图像进行分区统计，一个更进一步的工作是用这些统计量对掩模区域进行过滤。填写正数表示选择大于等于该值的，负数代表小于该绝对值的。零表示不做限制，可以使用各种条件组合。如果通过所有约束，颜色设置为front ground，否则设置为back color

**筛选出比较亮的一块 $ Chapter7 Binary-Image > Show Select The Lighter One**

筛选均值大于等于100的区域，我们看到两块深色的大理石对应的区域被过滤掉了。

**筛选没有花纹的 $ Chapter7 Binary-Image > Show Select The Pure One**



筛选均方差小于50的区域，我们看到第一块和第三块由于上面带有明显纹路，因而方差较大，而中间的样本接近于纯色，方差足够小，得以保留。

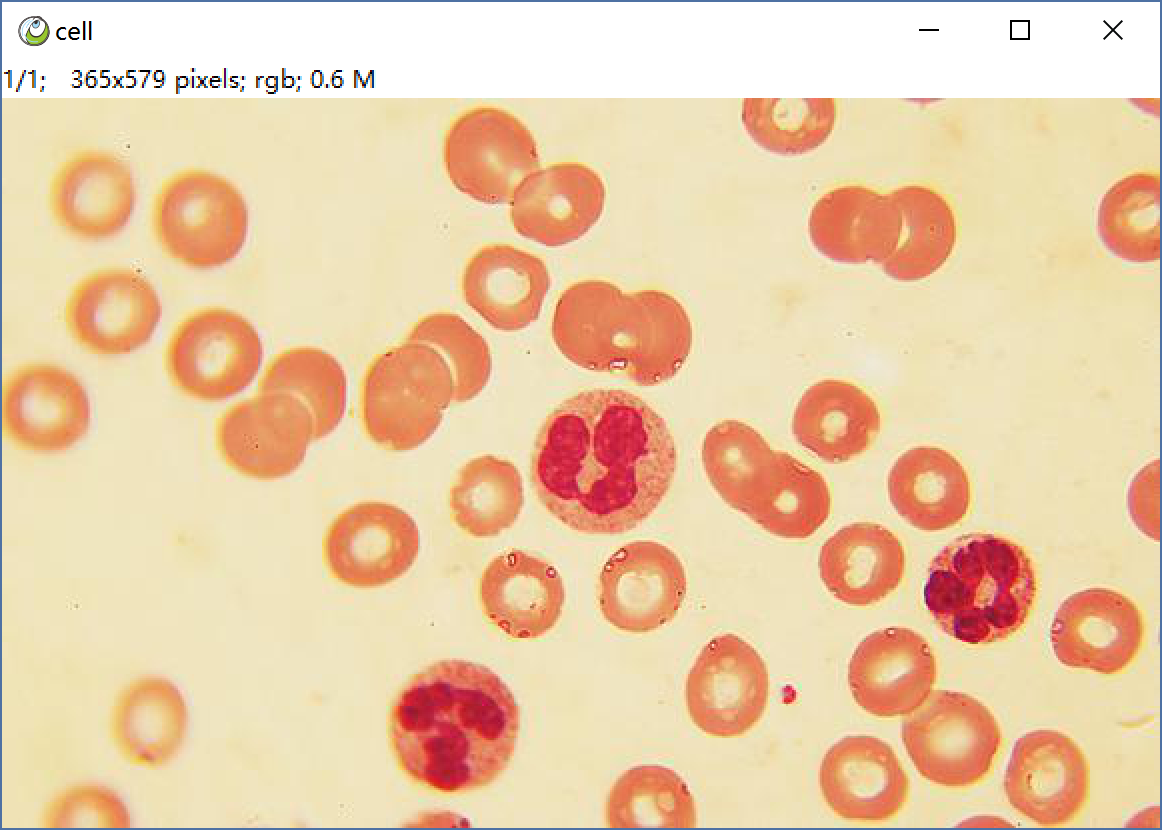
**一个细胞的例子**

至此本章的内容基本介绍完了。本章的知识非常重要，是测量，计数等操作的基础。由于本章信息量比较大，因而这里用一个综合性的例子加以巩固，会用到本章讲述的大多数功能，另外这里将一改之前理论讲解为主的编写风格，会详细介绍每一步的操作，用到的知识都是书中之前提到的，绝大多数是本章内容，请读者在阅读和操作同时，体会每一步操作的目的和意义，以及用到了哪些图像处理方法。

这是一张显微镜下的白细胞图片，白细胞是一类无色、球形、有核的血细胞，在血液中成球状，形态上可以分为有颗粒，无颗粒两大类。

1. 统计共多少个细胞
2. 对每个细胞进行密度测量，计算平均灰度，以及方差
3. 计算有颗粒和无颗粒细胞的数量及比例
4. 计算有几对粘连细胞。

**$ IBook > Image Referenced > cell** 打开这张显微镜下的细胞照片



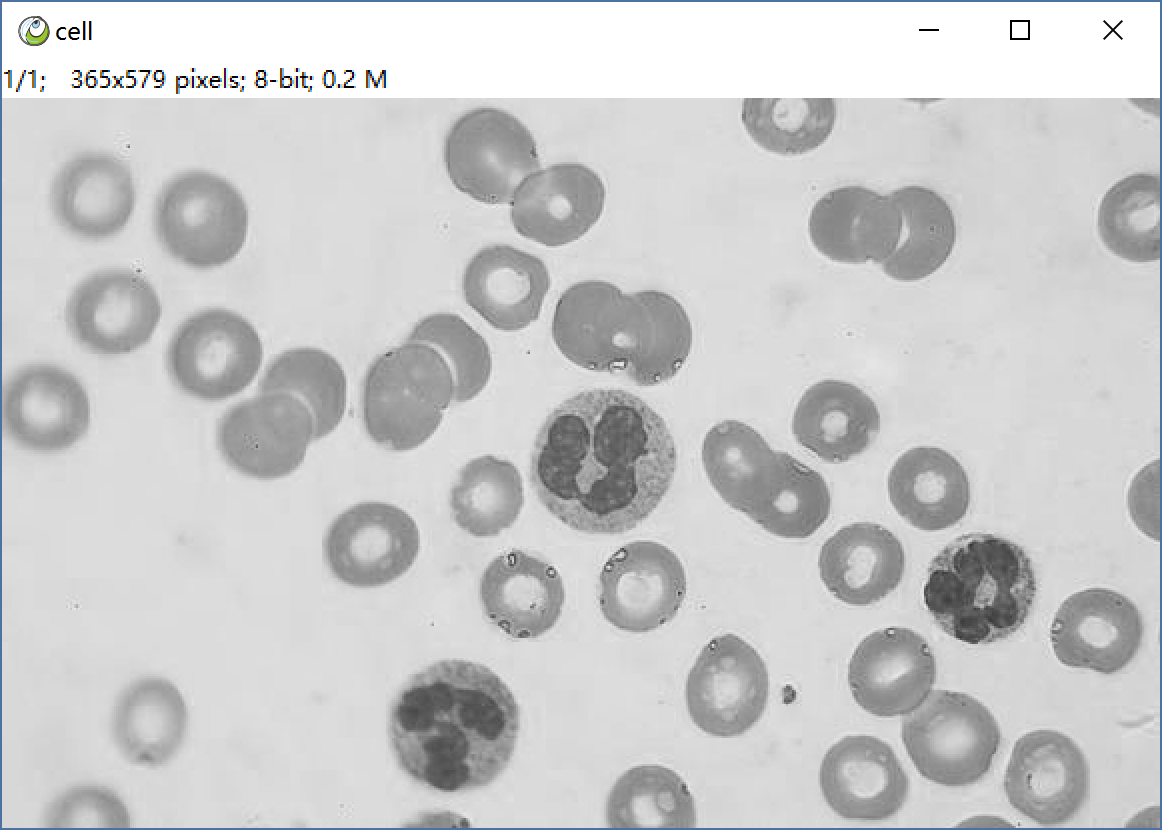
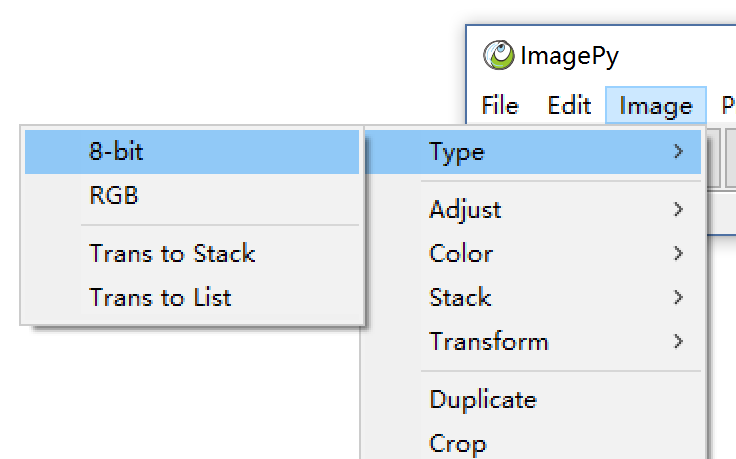
**分析：**我们要做的是一个计数，分区统计，以及筛选问题，重点是制作掩模，对于粘连细胞需要进行分离，然后根据区域形态或者密度进行目标筛选，统计。

**$ Chapter7 Binary-Image > Show Cell Analysis**

**$ Chapter7 Binary-Image > Show Cell Connected**

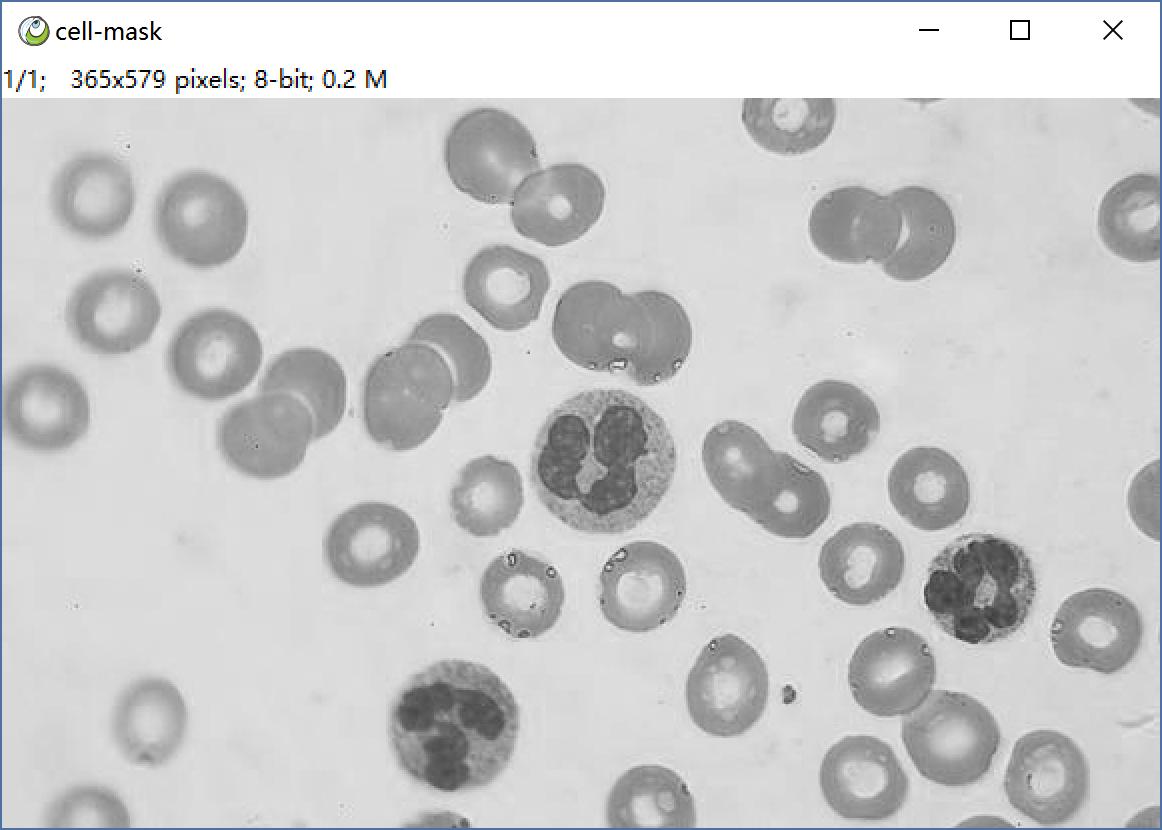
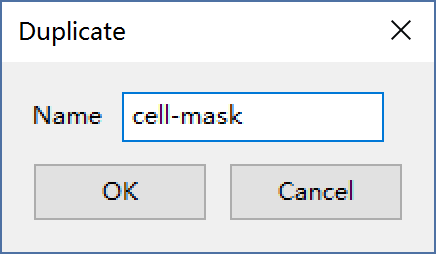
**$ Chapter7 Binary-Image > Show Cell With Granule**

**$ Image > Type > 8-bit** 转为灰度

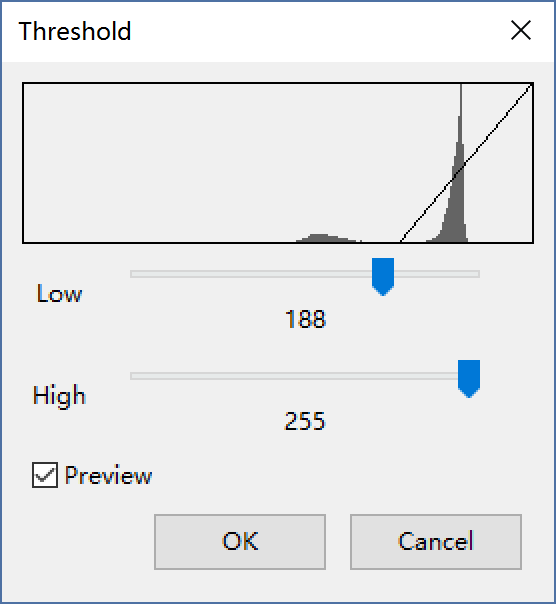
我们依然只保留灰度信息（不是说色彩信息没有用，而是简单够用就好）

**$ Image > Duplicate** 复制

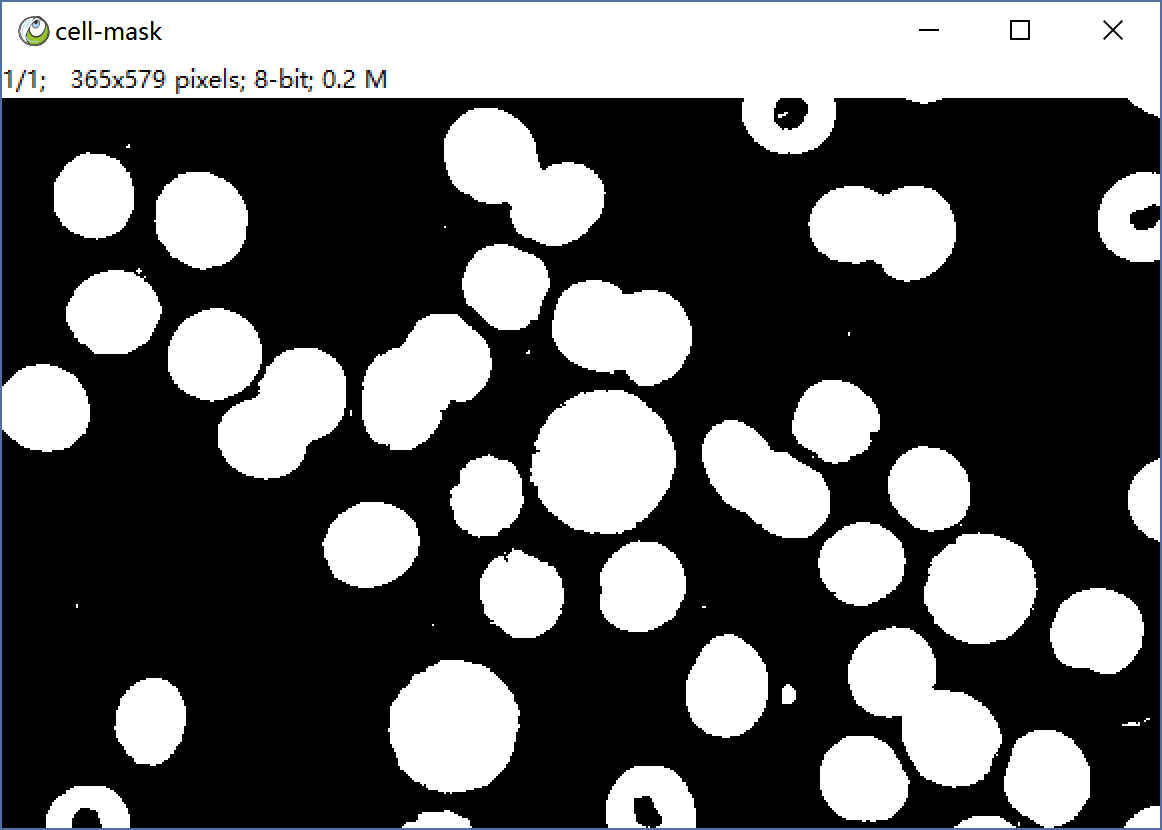
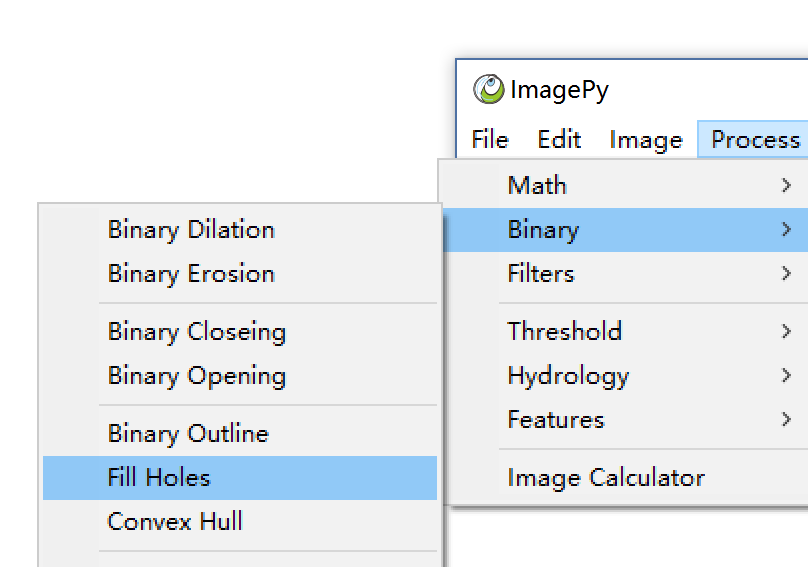
因为我们稍后要进行密度分析，因而掩模的制作在副本上进行

**$ Image > Adjust > Threshold**阈值处理

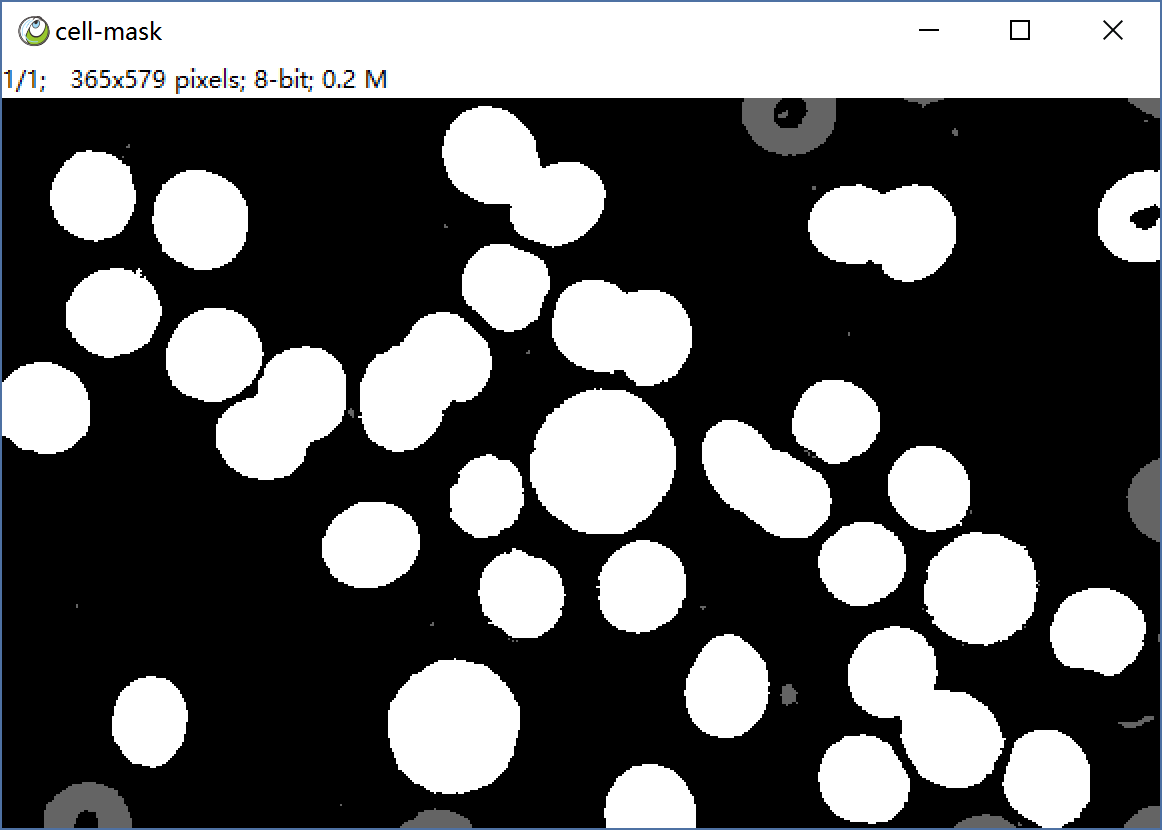
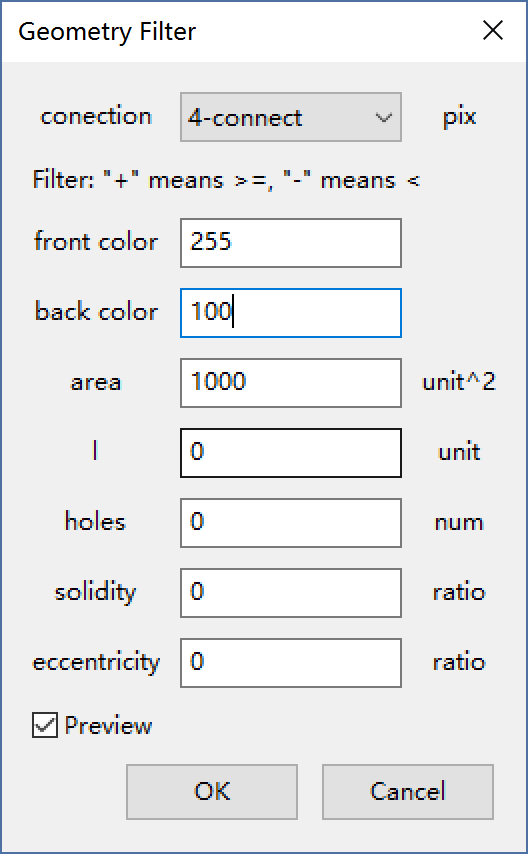
当然不要指望阈值之后就得到理想的掩模，具体阈值到什么程度，你要考虑后续如何可以修补掩模。

**$ Process > Binary > Fill Holes** 填充内部空隙

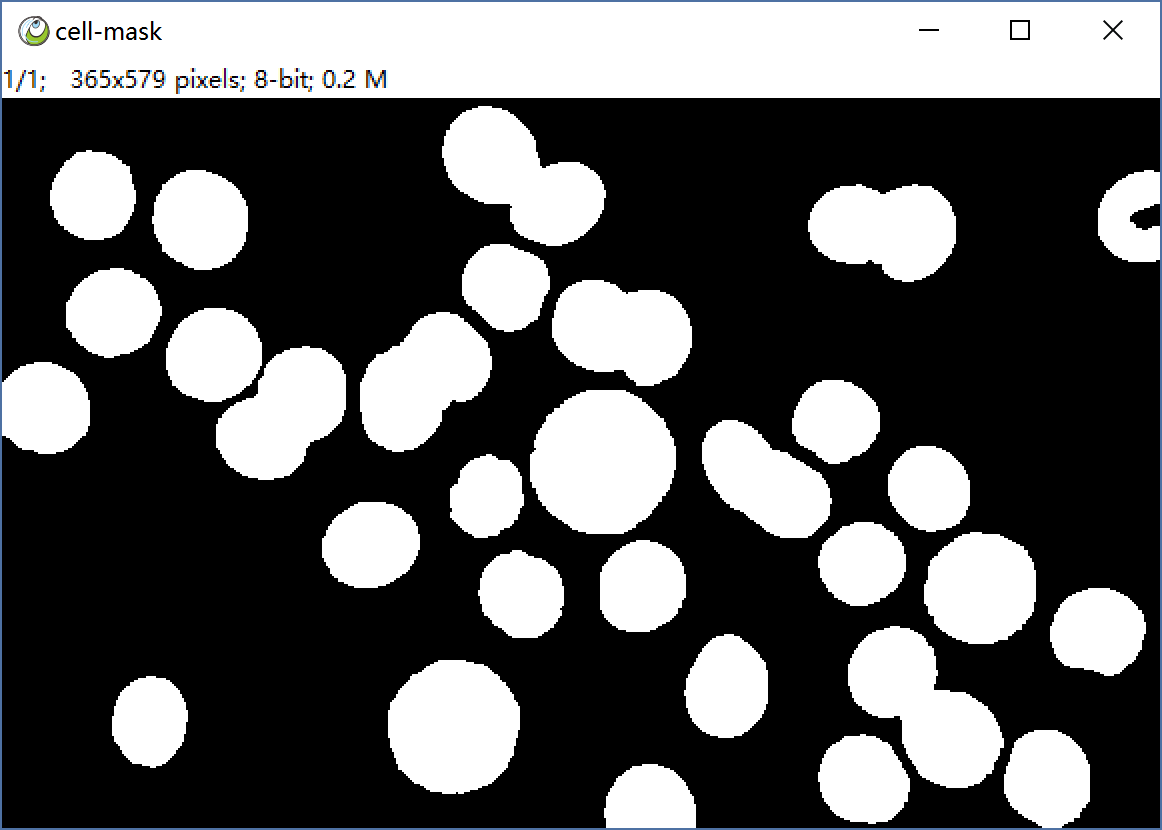
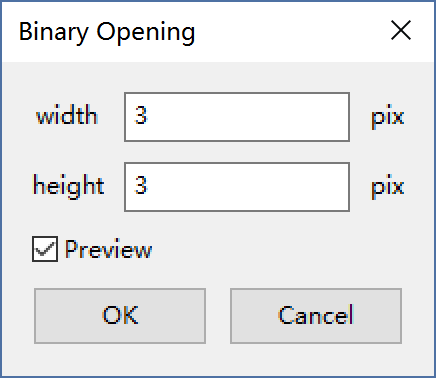
一些细胞中间是亮的，导致阈值之后是空心区域，我们要分析的是整个细胞，因此进行二值填充。

**$ Analysis > Region Analysis > Geometry Filter** 去除面积不足100的碎片。

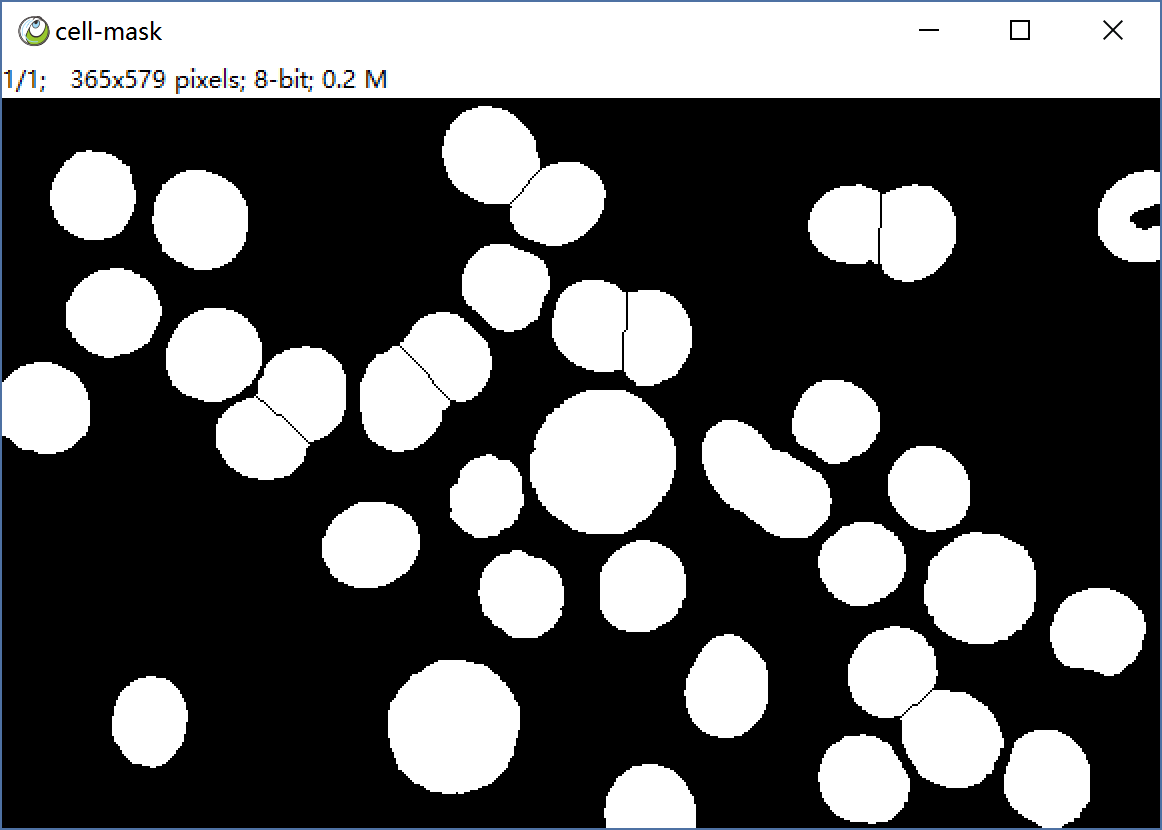
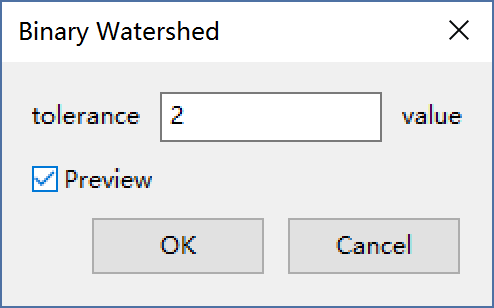
可以看到除了细胞还剩下一些小块碎片，那显然不是我们要分析的目标，用面积过滤器去除。

**$ Process > Binary > Binary Openint** 开运算对边缘进行修饰

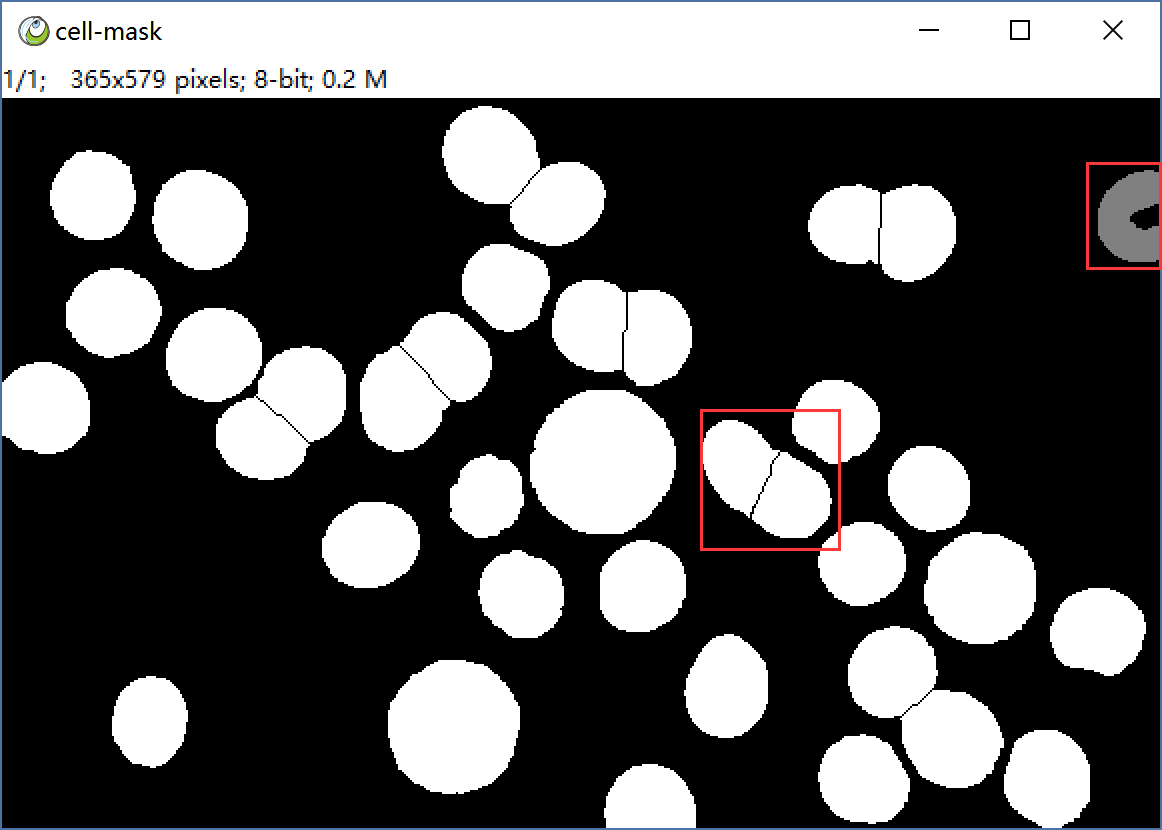
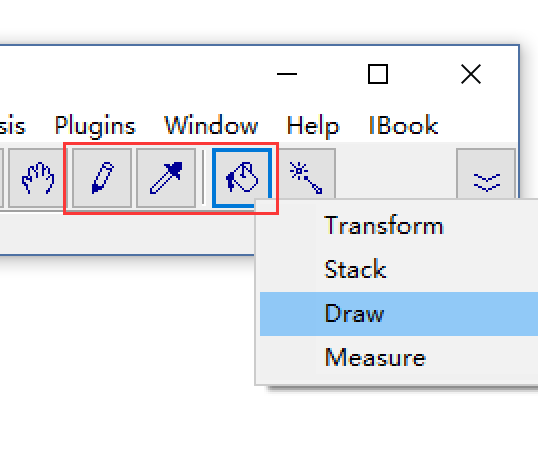
这一步只是把区域的毛边去掉，否则下一步分割有可能产生碎片

**$ Process > Binary > Binary Watershed** 二值分割

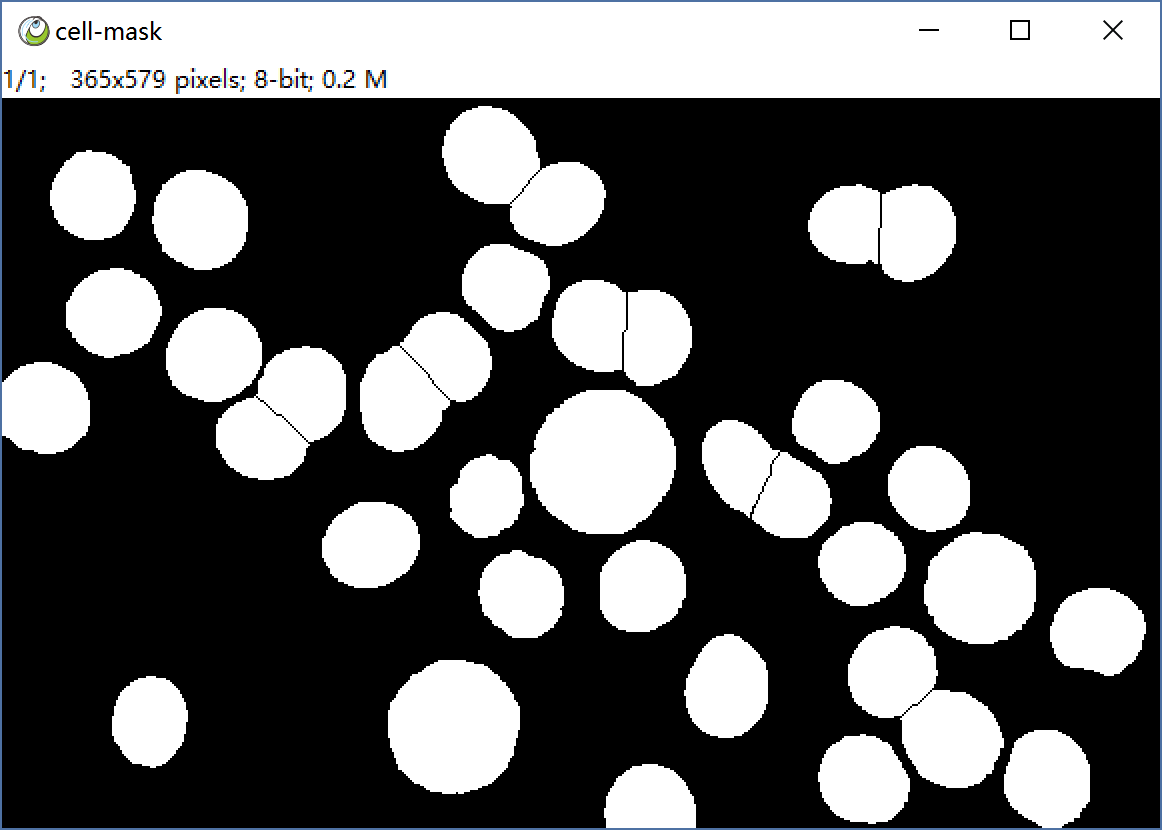
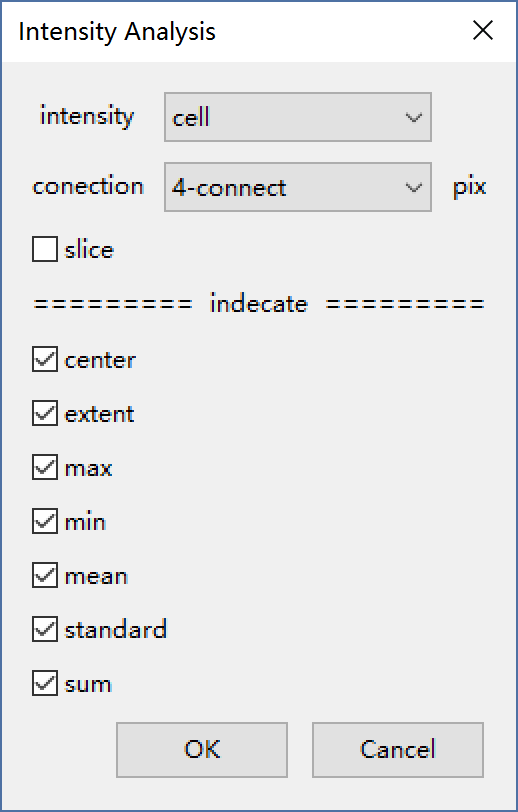
粘连细胞特征是中间有鞍部，因而使用二值分水岭分割方法打断。

**绘图工具**

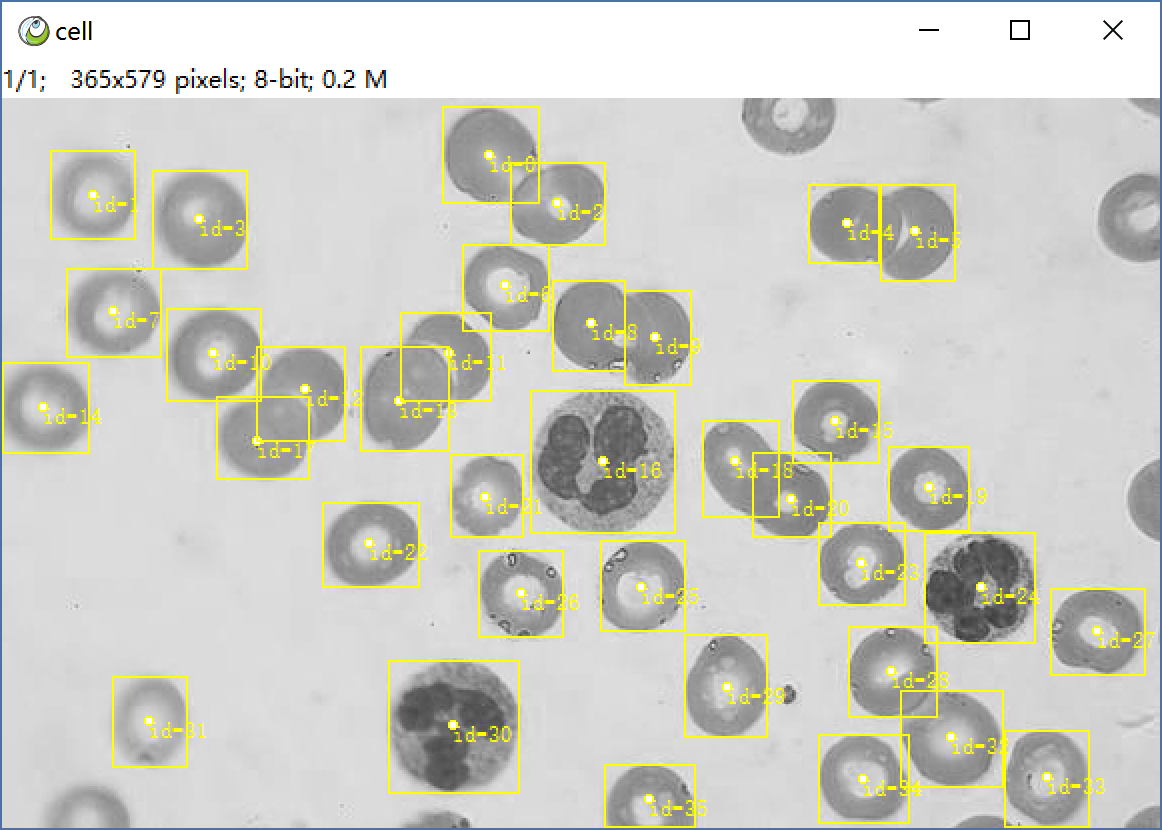
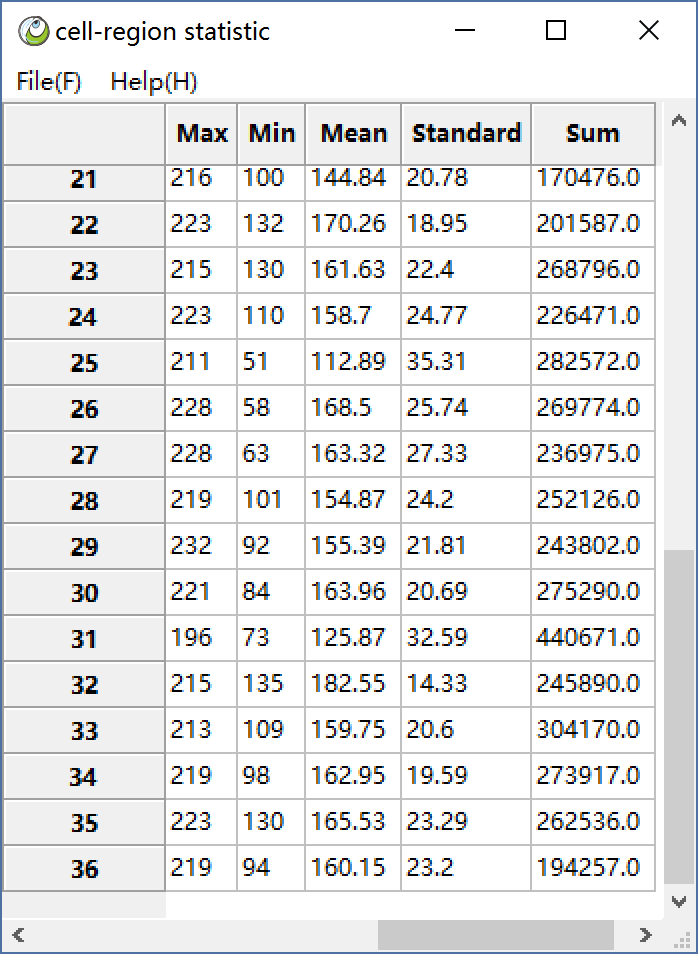
如果有不满意的地方，也可以手工修饰，但也不必太认真，因为个别的错误不太影响最后统计结果。

**$ Analysis > Region Analysis > Intensity Analysis 区域灰度统计**

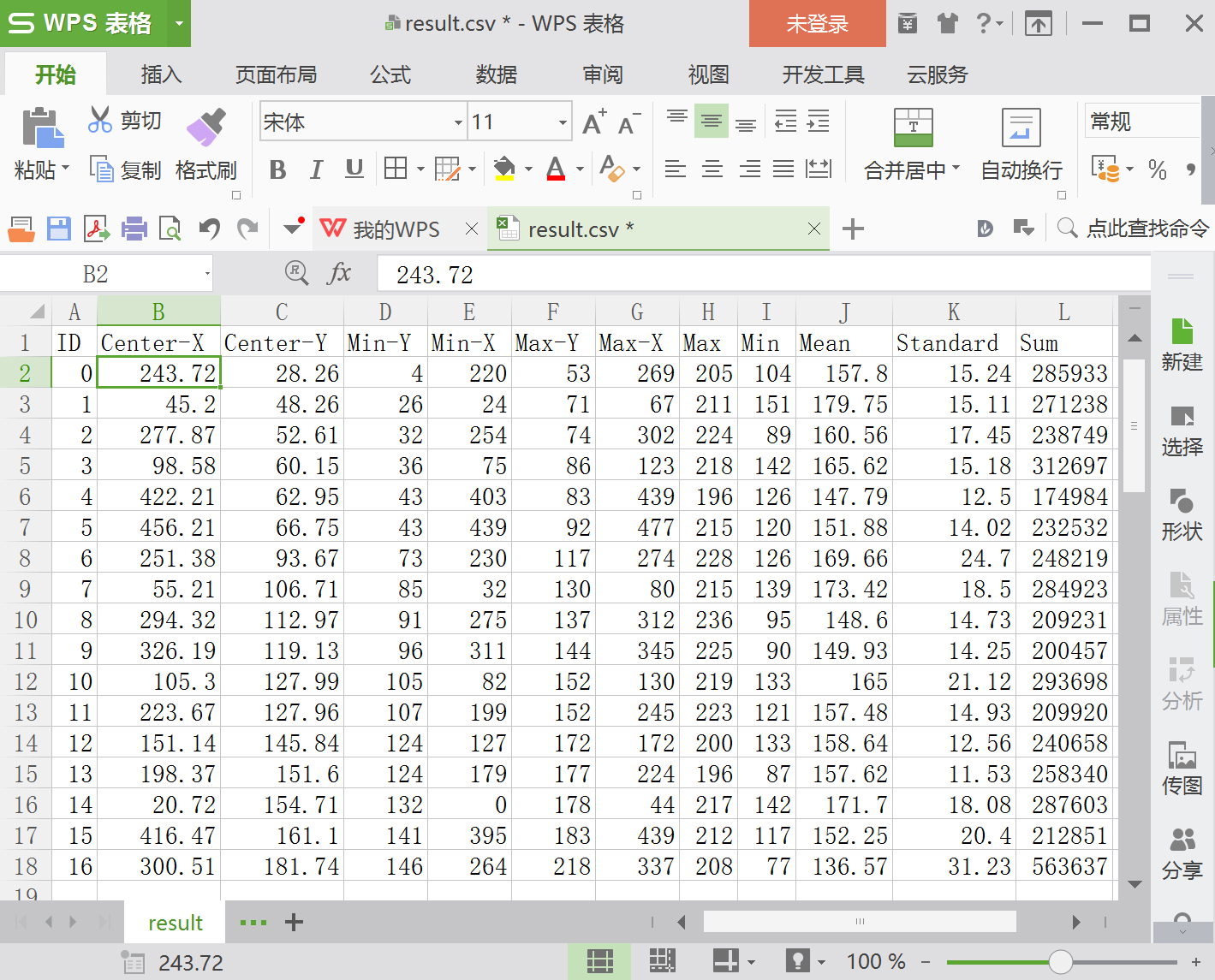
用图像当作掩模，选择cell为目标进行分析，注意选择四邻域，否则粘连细胞会被当成一个。

**得到统计结果**

得到结果报表，共计36个。

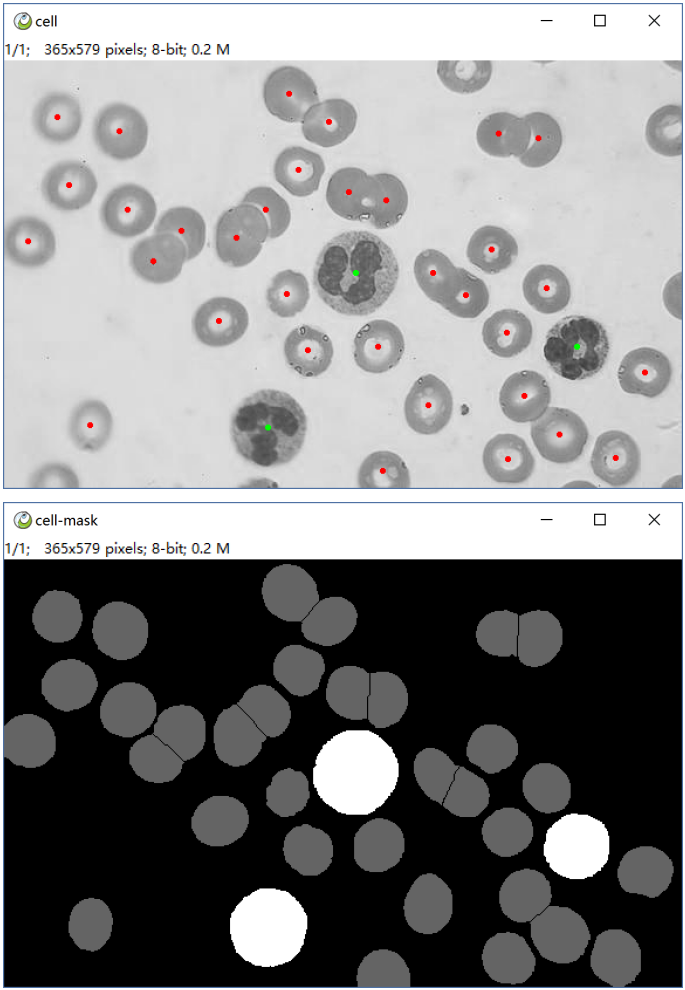
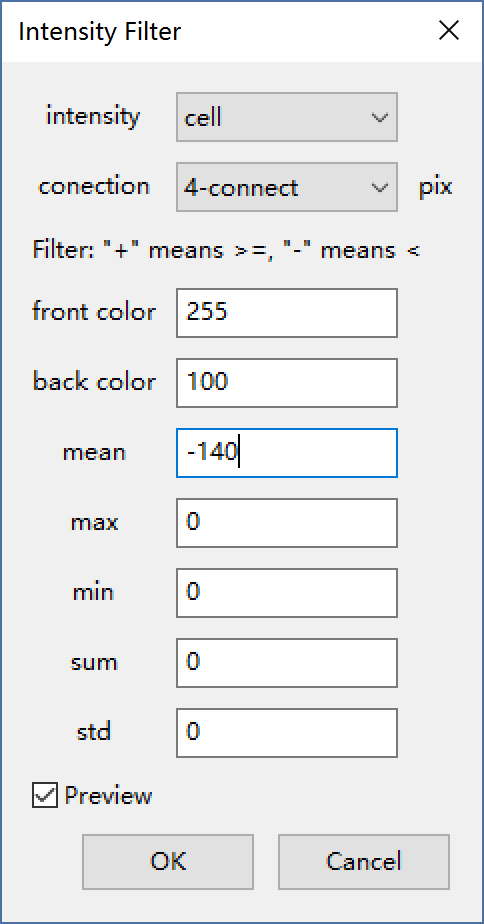
**报表窗口菜单File > Save As CSV**



如果需要，可以在File菜单，Save As成Excel，进行更进一步统计分析。比如做密度分布图。

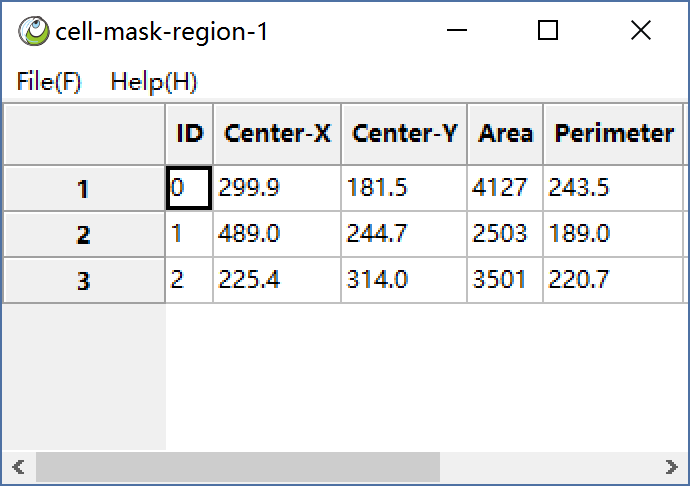
**小技巧：**我们在处理类似问题的时候，有一些中间过程是非常重要的，比如分割之后的掩模图像，这些中间结果“来之不易”，并且此后的操作都是基于它，所以保险起见，我们可以中途保存这些中间过程，以免后续由于操作不当，前功尽弃。值得一提的是，掩模图像请实用PNG，这样能获得最佳压缩效果，并且不会产生精度损失，万不可实用jpg进行存储，那样再次打开时，就不再是二值图像了。

**Analysis > Region Analysis > Intensity Filter**

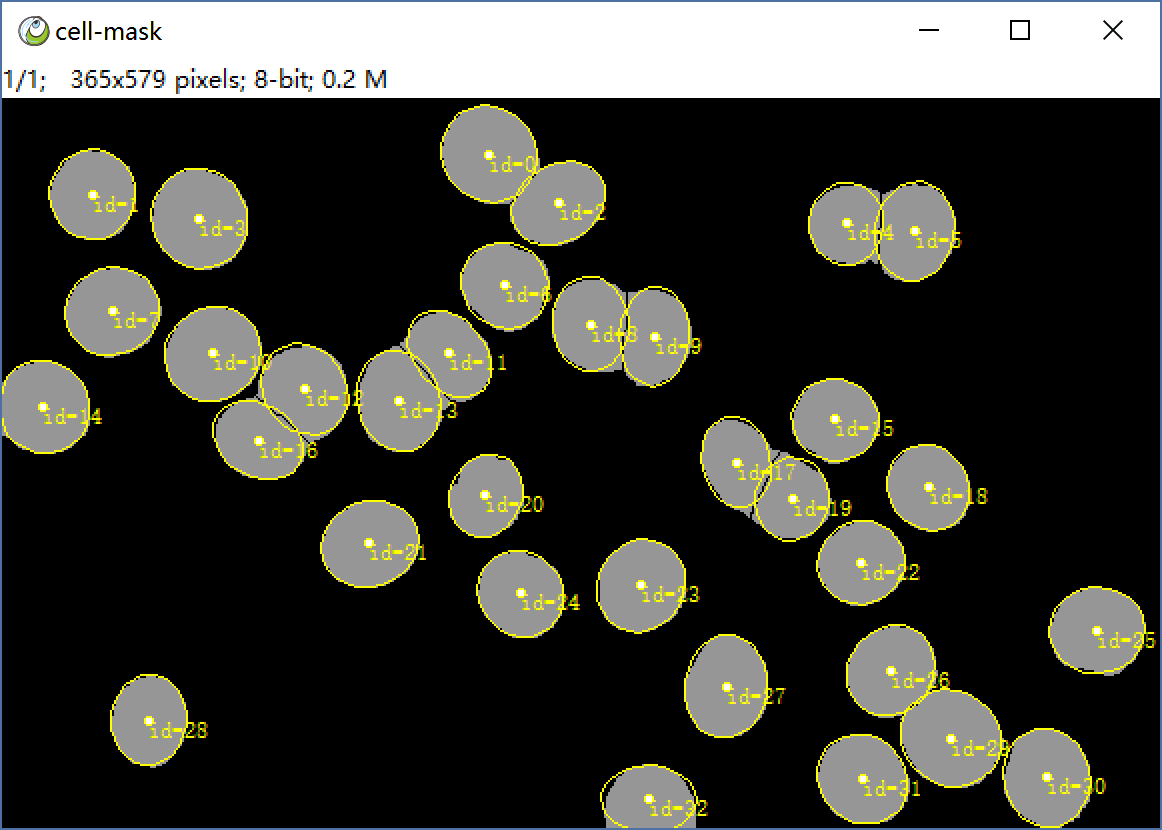
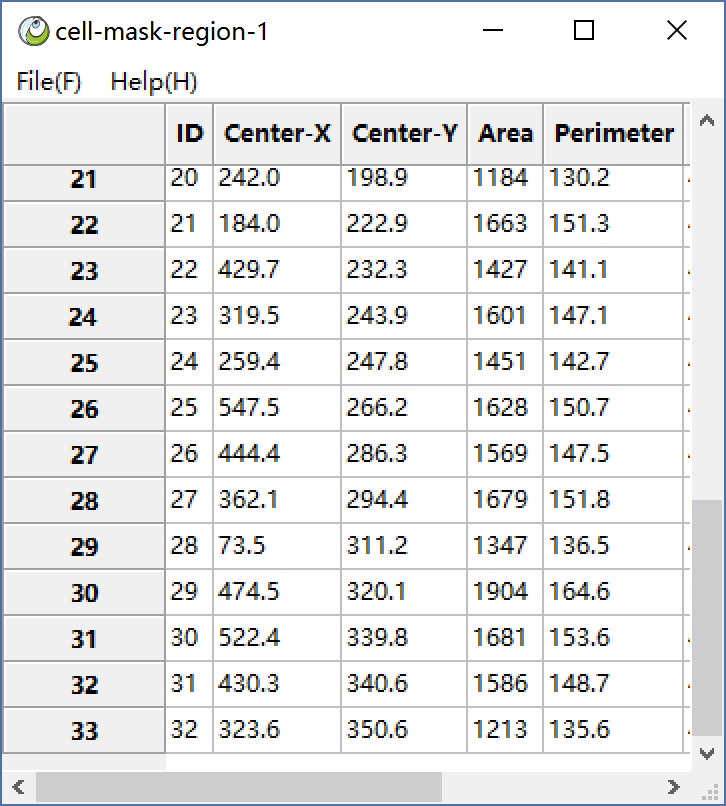
我们观察，带有颗粒的白细胞内部明显有几个暗核，所以会导致整体密度比无颗粒的低，因此我们用均值进行过滤，填写-140，表示选区小于140的区域。注意这里是为了演示，把被过滤掉的区域设置未100，读者自己操作，如果确认筛选成功，请将back color设置成0，否则依旧会纳入统计。

**Analysis > Region Analysis > Geometry Analysis**

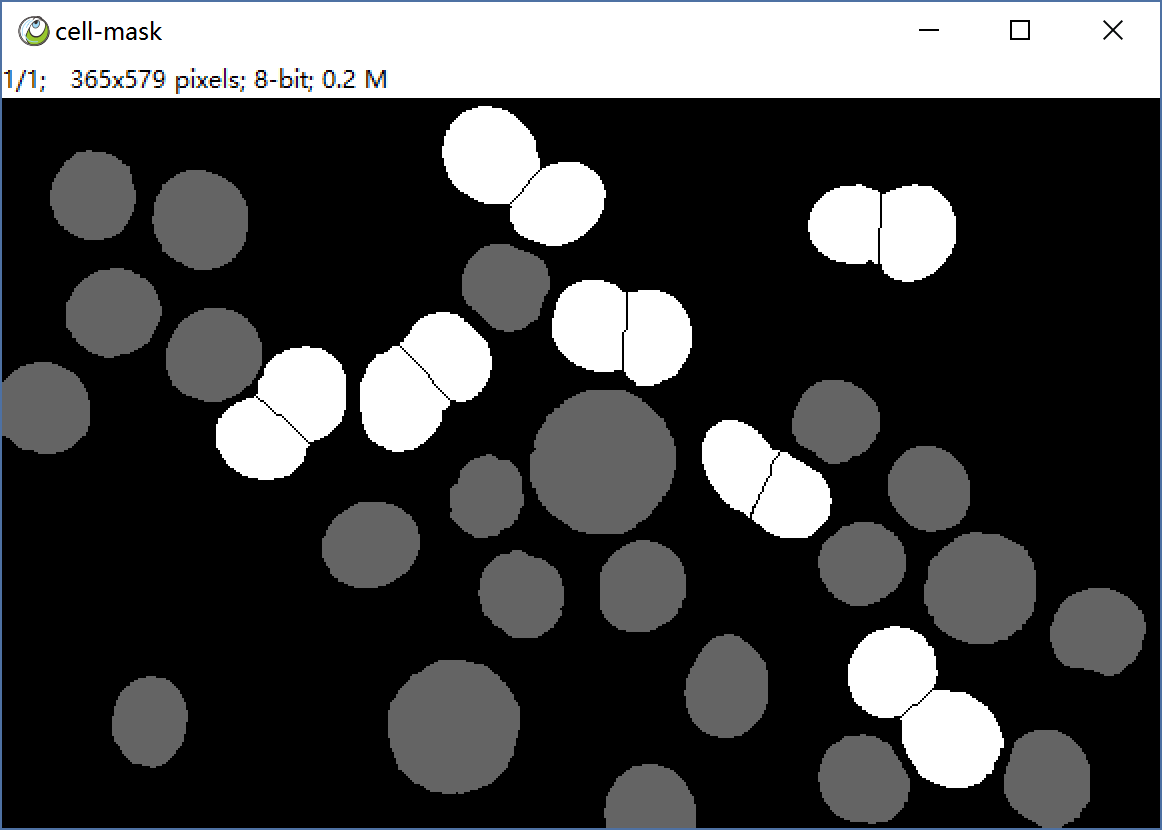
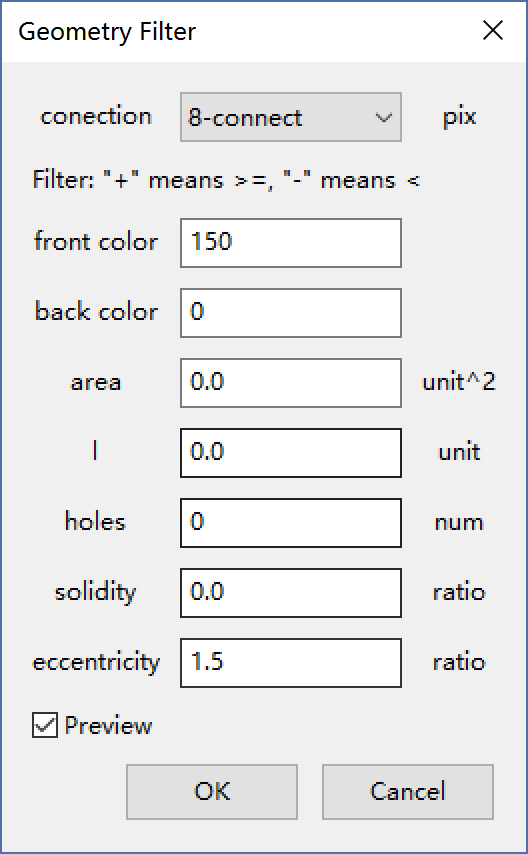
首先我们得到统计结果，3个，看起来有些画蛇添足，因为一眼就可以看出三个，但实际问题往往很多。

这里有一个小技巧，因为多数处理操作都是可以进行单步撤销的，所以，读者可以把掩模进行撤销，回到之前的状态，只要再次进行区密度筛选，mean填写140，去掉符号，我们就可以得到如下结果，只保留无颗粒的。

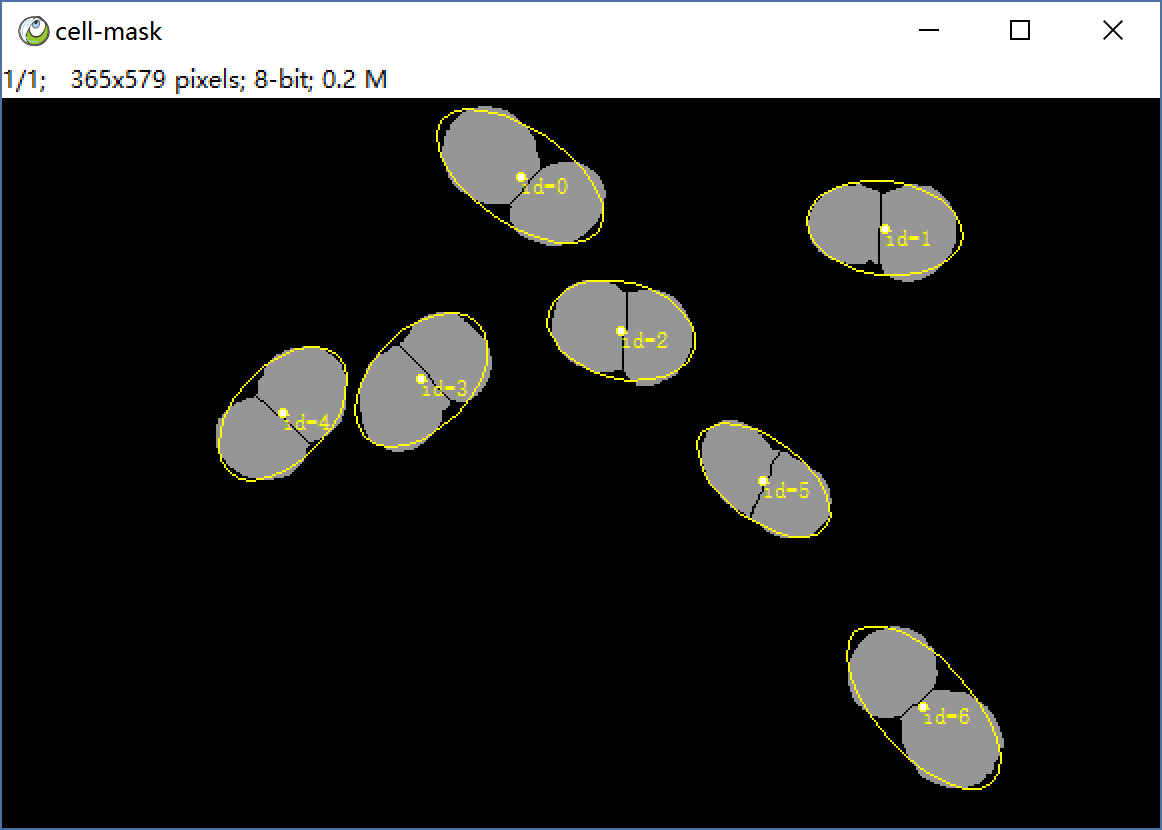
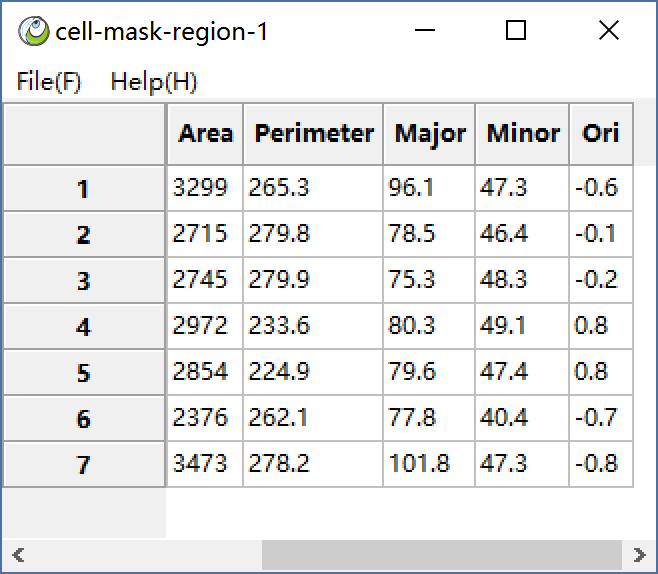
再次得到统计结果，33，接下来我们再次撤销掩模过滤，来统计粘连的有几对。

**$ Analysis > Region Analysis > Geometry Filter**

我们观察粘连的细胞都是一个很长的区域，用等效椭圆的偏心率来说，就是一个偏心率很大的椭圆，注意这里实用8邻域，因为这样会自动跨过分水线。

**Analysis > Region Analysis > Geometry Analysis**

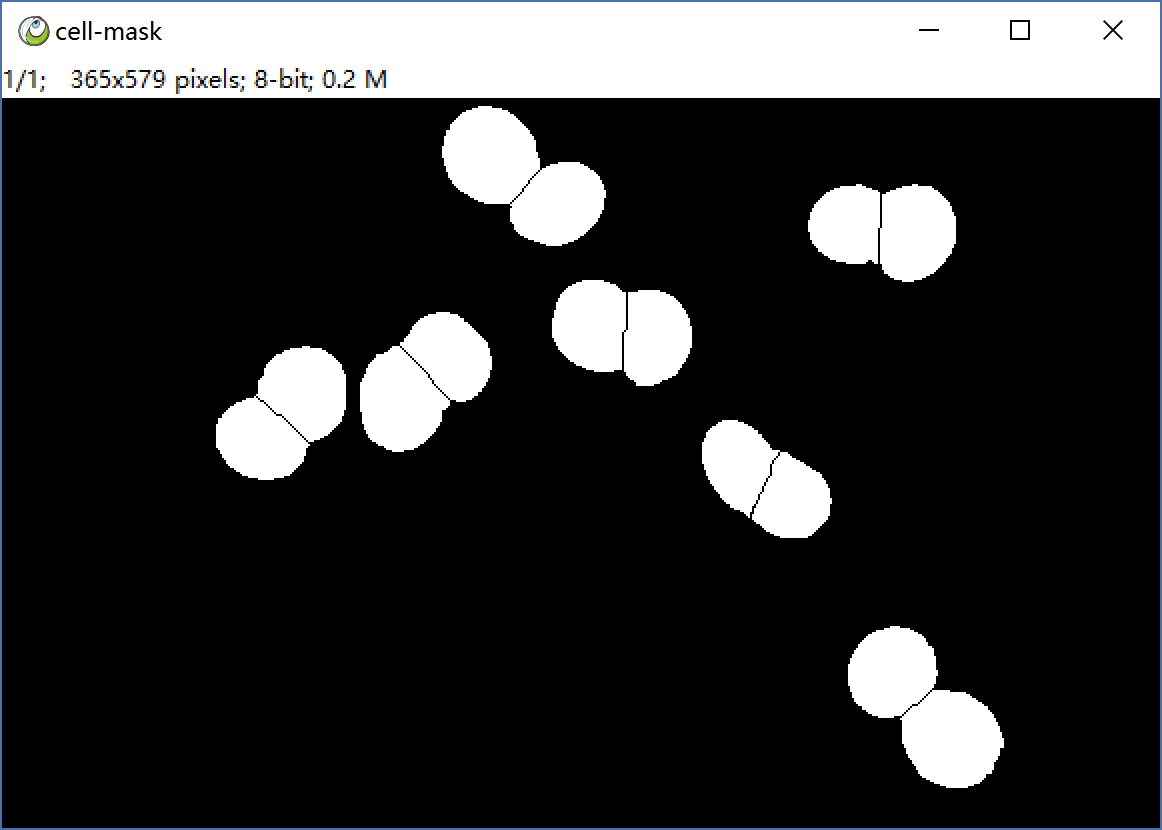
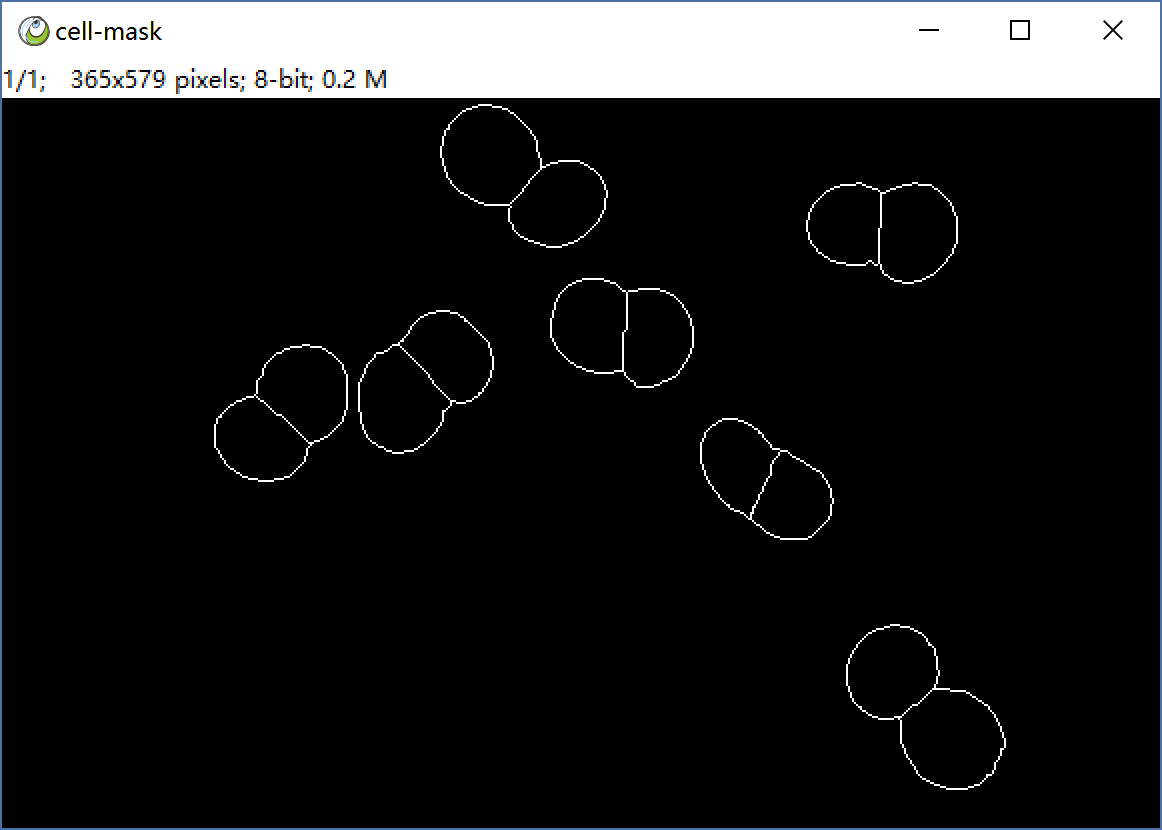
一共7对粘连细胞，至此我们全部问题都得到解答。

**视觉效果**

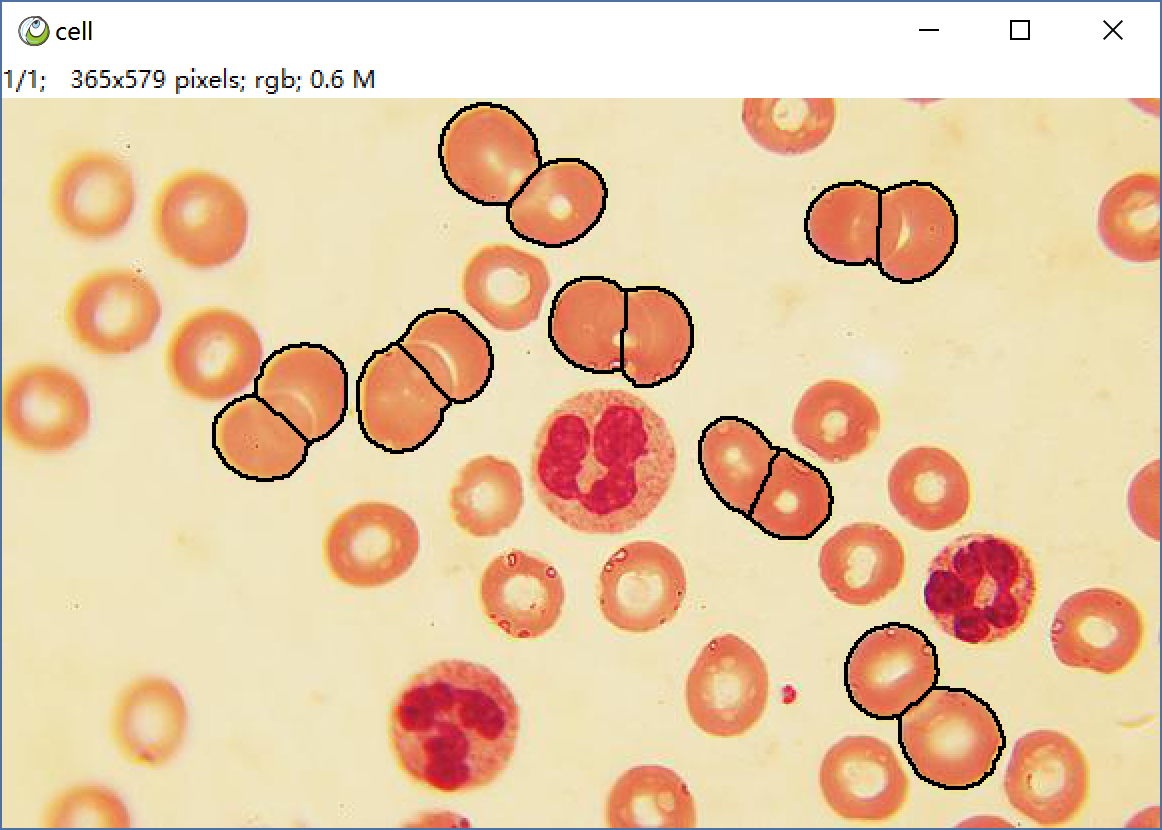
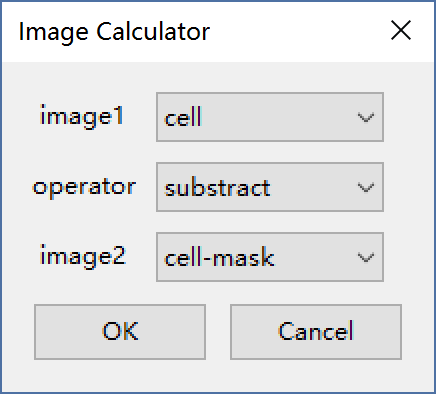
虽然我们全部问题都已经得解，但毕竟觉得数值过于抽象，俗话说有图有真相，效果好的图像可以给外行更多安全感，也可以为你的论文增色很多，我们最后做一些有利于展示的效果，这里我们做如下工作：

1. 我们来勾画粘连细胞
2. 我们对含有颗粒的细胞进行抠图

**$ Process > Binary > Binary Outline**

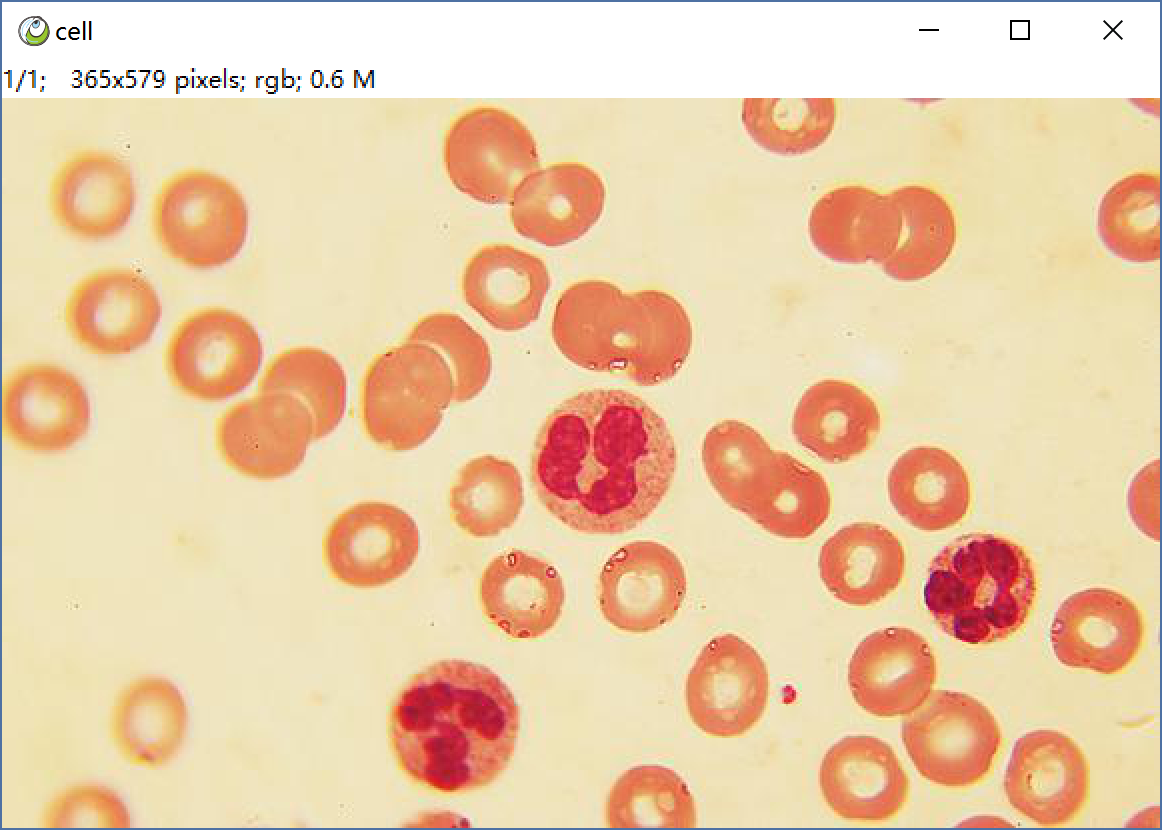
 

在之前的通过二值轮廓线，得到粘连细胞的轮廓线。不过这样一来，你将无法通过撤销找回完整的掩模，之前介绍的小技巧，存储中间结果只是预防重要信息的丢失，而这种，目测会破坏重要信息的操作，我们大可以操作之前duplicate一份，在副本上进行。

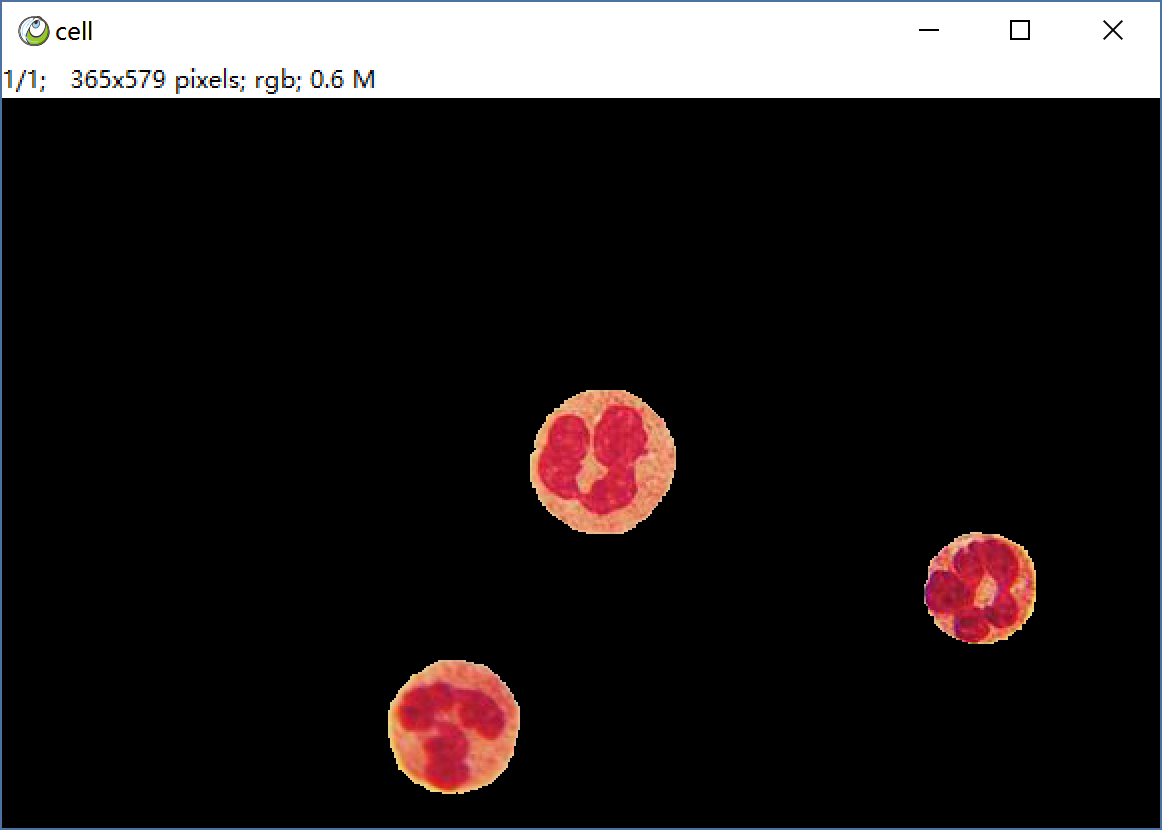
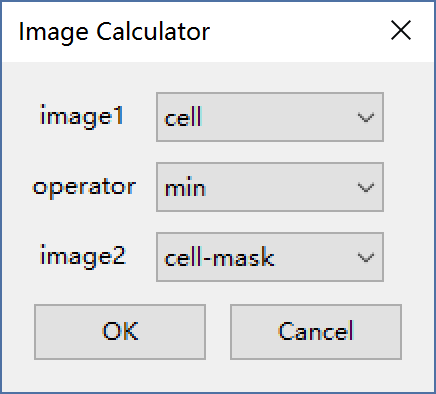
 

我们通过图像减法将彩色原图中粘连细胞的轮廓线进行勾勒，本质是因为和轮廓相减，像素值为零，当然，这里的轮廓线是为了视觉效果加粗过的，方法是在相减之前对二值轮廓线进行一个膨胀操作。

**小技巧：**虽说是为了视觉效果，但也不仅仅如此。重要的是这种叠加操作可以让人直观的感受到处理结果及精度，我们这里的问题是通过阈值二值化的，并且相对容易，但在一些诸如分水岭分割的问题时，我们如果不进行叠加，很难直观的评价分割结果如何。

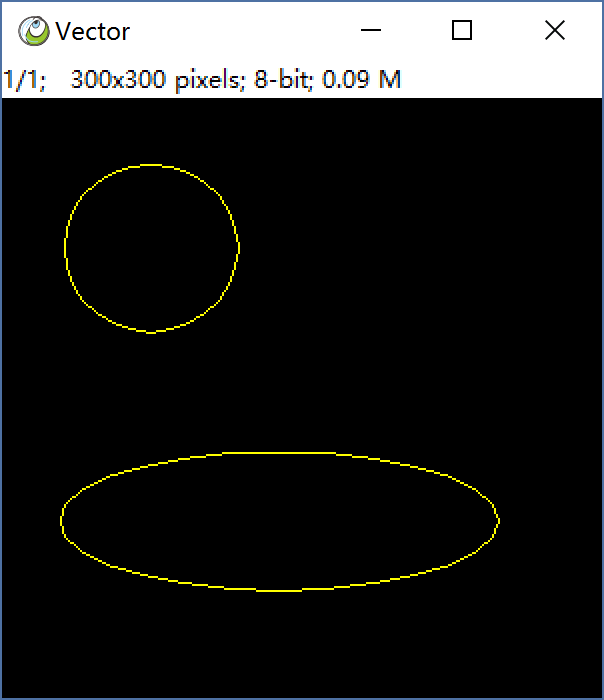
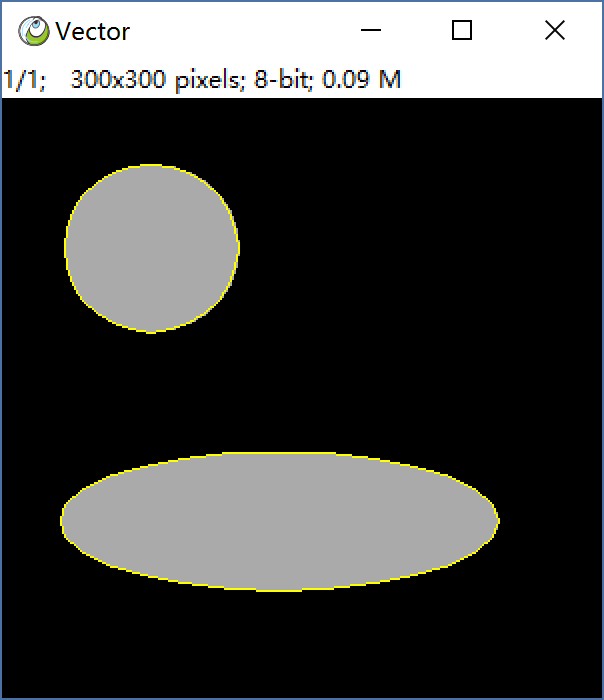
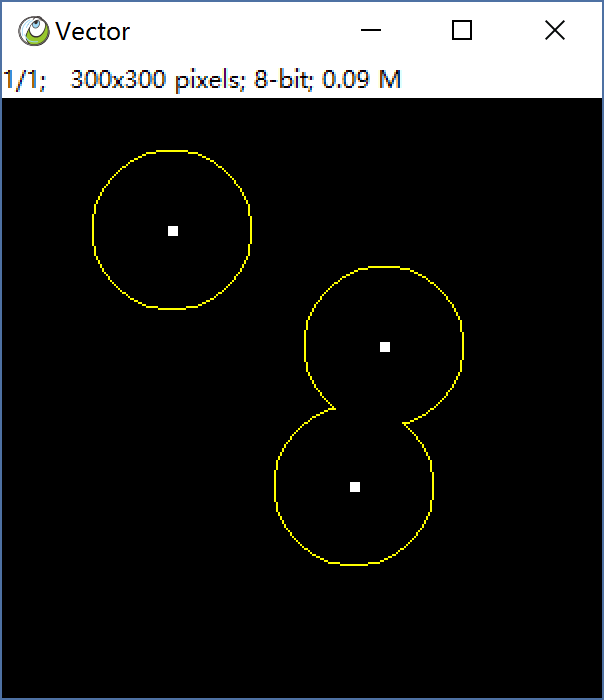
**$ Process > Image Calculator**

这次我们对掩模和彩色图像做min运算，既两幅图对应像素较小的。由于掩模的颗粒细胞区域是255，因而较小的一定是彩色图像，而其他区域掩模是0，因此会屏蔽原图其他部位，这样就完成了一个所谓的“抠图”工作。

**二值图像与矢量的关系**

本书主要介绍的是图像算法，但二值图像由于其明确性，某种意义上可以理解为矢量，关于矢量二值图像中的一块区域，可以用轮廓的坐标序列表示，这样就构成了一个矢量多边形，ImagePy重点不是矢量运算，但其中选区的确是基于矢量的。

两个椭圆ROI Image > Fill 填充 Selection > Inflate 扩张

**观察描述：**两个椭圆的roi本质是坐标序列，我们可以用填充方法把矢量转为图像，同理可以通过轮廓追溯再次得到矢量，我们的魔棒工具就是这个功能，而图3是点通过inflate扩张成圆，并且发生重叠。

矢量和二值图像是可以相互转换的，矢量转化图像的方法叫多边形填充，反过来叫轮廓追溯，矢量存储往往更节约空间，而所有的区域特性，包括质心，周长，误差椭圆等都有对应的矢量算法，膨胀腐蚀再矢量运算中叫buffer，地盘瓜分叫Voronoi，中轴线类似于Delaunay，关于矢量运算的学科叫计算几何，这些对于GIS专业应该有所了解，读者如有兴趣可以自己查阅相关资料。

**ImagePy 中的二值图像操作**

**二值操作，形态学滤波**

在ImagePy中，二值图像相关的操作集中在**Process > Binary**下，我们在距离变换一节介绍了二值图像的分割，瓜分地盘等问题，那些问题的原理的确是利用距离变换后的高程分析，不过我们使用不需要自己进行距离变换，再进行分水岭等操作，在**Process > Binary**下有专门为二值图像准备的**Binary Watershed**以及**Voronoi**方法。

**区域分析，测量**

测量相关的功能在**Analysis**菜单下，包括二值区域分析，过滤。也包括二值图像标记，全局灰度统计，利用掩模进行分区像素统计，以及利用密度进行区域过滤。

**本章小结**

本章是目前篇幅最多的一章，内容量很大，但是概念却相对简单。由于二值图像的是非分明特性，让一些操作变得简单可行，其实诸如测量，计数等工作，本质上是二值图像分析问题。

首先二值图像也是灰度图像的特例，因而之前学习的卷积等操作依旧适用，其次作为二值图像特有的形态学算法，可以感性的理解为用某种笔触去对原图进行描边，膨胀腐蚀只是用白色或黑色去描边的过程，接下来二值图像有凸包，填充，轮廓线等运算。距离变换是把二值图像通过距离变换转成灰度图像，借助高程分析算法，可以得到二值图像的中轴线，分割，完成地盘瓜分等工作。接下来区域是有特性的，我们学习了面积，周长，质心，饱满度，等效半径，等效椭圆，偏心率等特性，并可以根据特性对区域进行筛选，最后又学习了利用二值图像当作掩模，对另一幅图像进行分区域的像素统计，以及利用像素统计进行区域过滤。最后精心准备的例子，通过详细图文列举了每一步操作，几乎用到了本章的全部内容。请读者认真思考每一步操作的意义。

对偶特性是贯穿整个二值图像的，膨胀腐蚀部分，前景的膨胀等于背景的腐蚀，背景的膨胀等于前景的腐蚀，距离变换部分，前景的中轴线类似于背景瓜分地盘的分界线，前景的修补等于背景的分割，区域过滤部分，所有针对区域的方法，只有经过Invert，一样都可以作用于空隙。其实介绍每一部分时已经提到，这里再次拿出来强调，很多时候其实换一个角度看问题，就是我们熟悉的问题。