**数字图像处理 作业2 实验报告**

**姓名：周敏苑 学号：161220180 邮箱：[161220180@smail.nju.edu.cn](mailto:161220180@smail.nju.edu.cn)**

**实验要求**

**运用不同方法完成边缘检测和边缘链接函数，从图像中检测对象。**

**实验内容**

1. **边缘检测**

**边缘检测方法主要基于图像滤波，图像增强，图像检测三个步骤来实现的，在本次实验中，我才用高斯滤波函数来降低图像噪声的影响；而图像检测则采用一下几个常见的边缘检测算子实现的：**

**Roberts算子**

**Roberts算子是一种利用局部差分算子寻找边缘的算子，采用的是对角方向相邻两像素值之差，算子形式如下：**

**Gx = f(i,j) - f(i-1,j-1)**

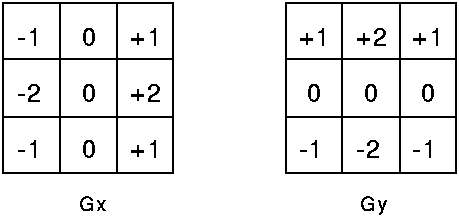
**Gy = f(i-1,j) - f(i,j-1)**

**|G(x,y)| = sprt(Gx^2-Gy^2)**

**找到适当的阈值τ（实际计算的时候经过尝试，阈值采用0.8的时候效果比较明显）,若 G (x,y)>τ，则 (i ,j)为边缘点，否则，判断 (i,j)为非边缘点。由此得到一个二值图像 {g(i,j)}，即边缘图像。**

**Sobel算子**

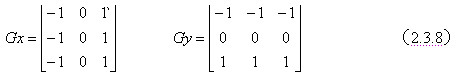
**索贝尔算子主要用作边缘检测，在技术上，它是一种离散性差分算子，用来运算图像亮度函数的灰度之近似值。在图像的任何一点使用此算子，将会产生对应的灰度矢量或是其法矢量，Sobel卷积因子为：**

****

**该算子包含两组3x3的矩阵，分别为横向及纵向，将之与图像作平面卷积，即可分别得出横向及纵向的亮度差分近似值**

**Prewitt算子**

**同 Sobel 算子相似，Prewitt 算子也是一种将方向的差分运算和局部平均相结合的方法，也是取水平和垂直两个卷积核来分别对图像中各个像素点做卷积运算，所不同的是，Sobel算子是先做加权平均然后再微分，Prewitt 算子是先平均后求微分，其对应的卷积模版为：**

****

**图像中的每个像素点和以上水平和垂直两个卷积算子做卷积运算后，再计算得到**

**梯度幅值 G (x,y)，然后选取适当的阈值τ（经过多次尝试，阈值采用0.4比较好）G(x,y)>τ，则 (i,j)为边缘点，否则，判断(i,j)为非边缘点。由此得到一个二值图像 {g(i,j)}，即边缘图像。**

**Laplacian算子**

**Laplace算子为二阶导数算子，一阶导数的局部最大值或鞍点对应着二阶导数的零交叉点故计算得到的近似零点就是一阶导数的最大值，判别为边缘，定义为：**

**图像算法 - 常用边缘检测算法对比分析 - dingmz_frc - dingmz_frc的博客**

**Canny算子**

**具体方法为：**

**（1）用高斯函数h(r)对图像进行平滑滤波，去除图像中的噪声。**

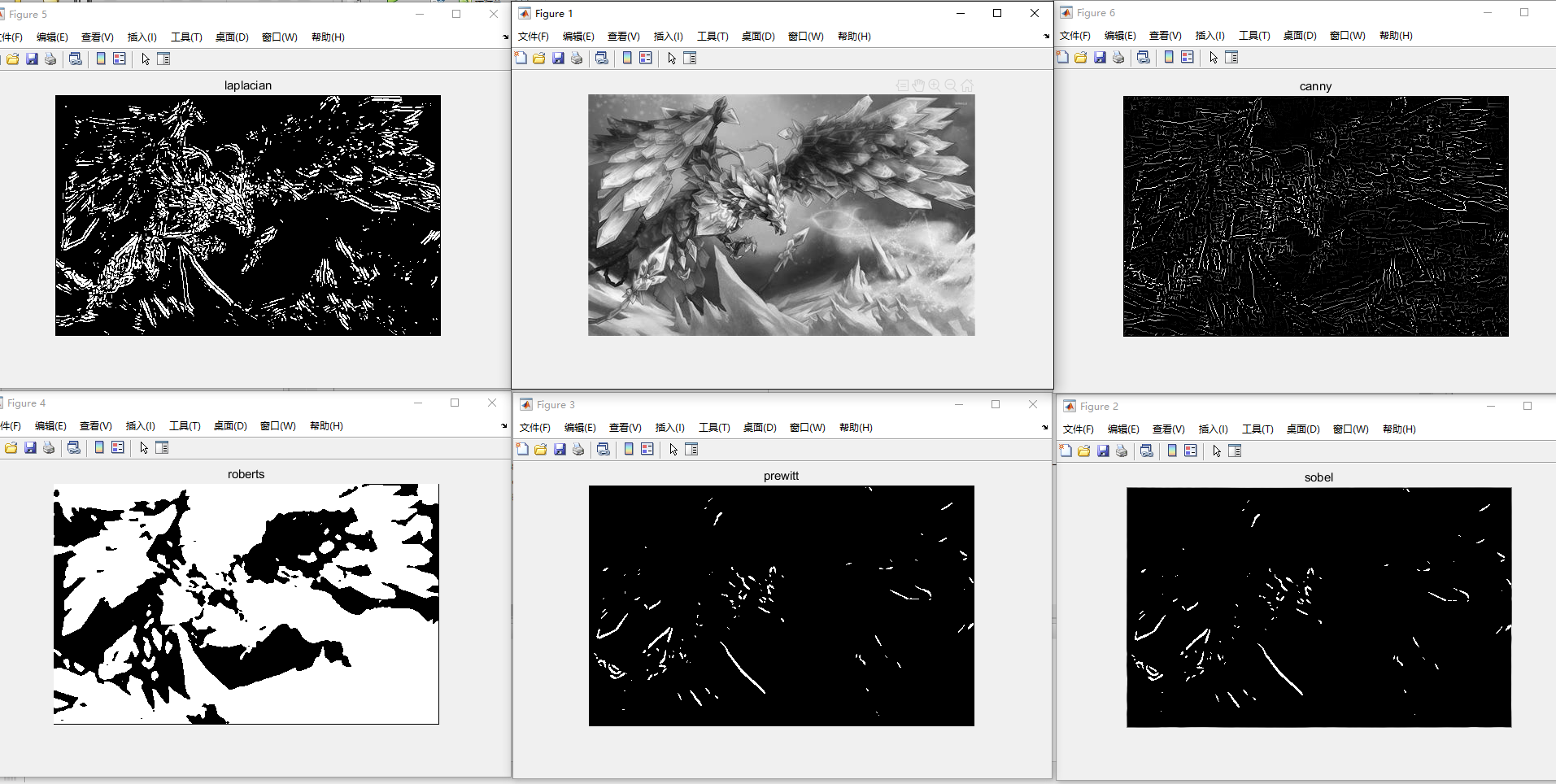
**（2）在每一点计算出局部梯度和边缘方向，可以利用Sobel算子、Roberts算子等来计算。边缘点定义为梯度方向上其强度局部最大的点。**

**（3）对梯度进行“非极大值抑制”。在第二步中确定的边缘点会导致梯度幅度图像中出现脊。然后用算法追踪所有脊的顶部，并将所有不在脊的顶部的像素设为零，以便在输出中给出一条细线。**

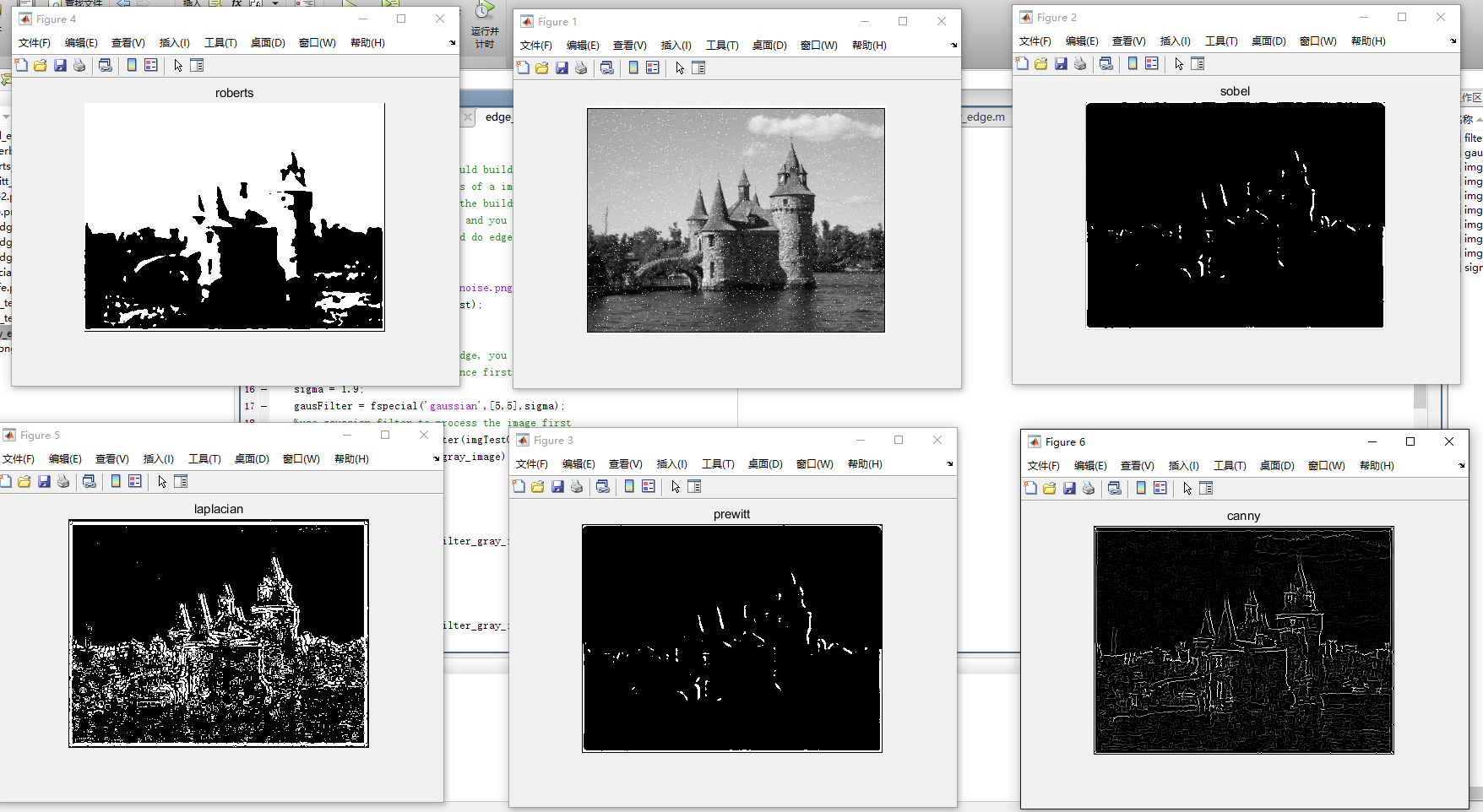
**（4）双阐值化和边缘连接。脊像素使用两个闽值Tl和竹做阂值处理，其中Tl<T2.值大于竹的脊像素称为强边缘像素，Tl和T2之间的脊像素称为弱边缘像素。由于边缘阵列孔是用高闽值得到的，因此它含有较少的假边缘，但同时也损失了一些有用的边缘信息。而边缘阵列Tl的闽值较低，保留了较多信息。因此，可以以边缘阵列几为基础，用边缘阵列Tl进行补充连接，最后得到边缘图像。**

**五种算子的比较**

**噪声较小图片：**



**噪声较大图片：**



**通过比较可以得出，无论噪声大与否，canny算子得到的边界图像最完整有效，具体各个算子的优劣为：**

**Roberts算子：**

**定位精度高，对于水平和垂直方向的边缘，检测效果较好，而对于有一定倾角的斜边缘，检测效果则不理想，存在着许多的漏检。另外，在含噪声的情况下，Roberts 算子不能有效的抑制噪声，容易产生一些伪边缘。**

**Sobel算子：**

**对噪声具有平滑作用，提供较为精确的边缘方向信息，边缘定位精度不够高。**

**Prewitt算子：**

**对噪声有一定抑制作用，抗噪性较好，但由于采用了局部灰度平均，因此容易检测出伪边缘，并且边缘定位精度较低。**

**Canny算子：**

**好的信噪比准则，即将非边缘点判为边缘点的概率要低，将边缘点判为非边缘点的概率要低;好的定位性能准则，即检测出的边缘点要尽可能在实际边缘的中心;** **单边缘响应准则即单一边缘具有唯一响应，单一边缘产生的多个响应的概率要低，并且对虚假边缘的响应应得到最大抑制。**

**Laplacian算子：**

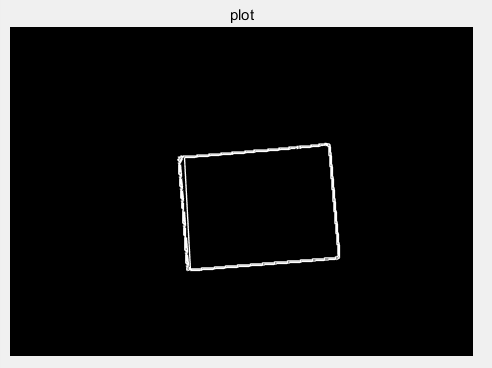
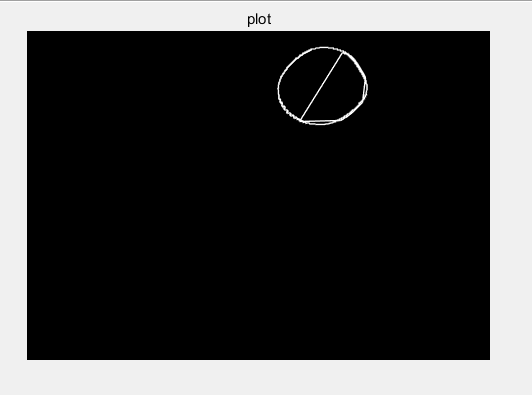
**Laplace算子对噪声具有无法接受的敏感性，因此在对图像做拉普拉斯边缘检测时，务必先滤波；Laplace算子的幅值产生双边元，这是复杂的分割不希望有的结果;最后，Laplace算子不能检测边缘的方向。**

**边缘链接**

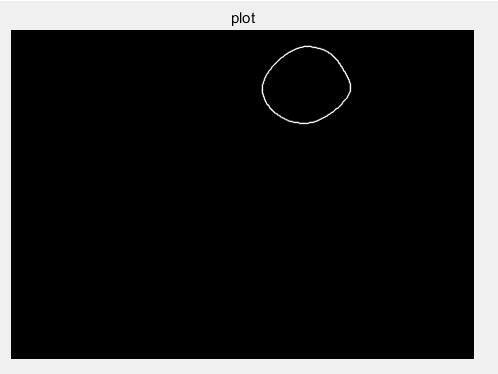
**在边缘检测部分，我使用了八连通和四连通两种方式，接下来着重介绍八连通的边缘链接方法：**

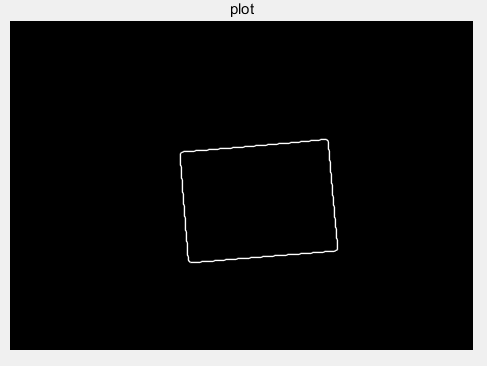
1. **首先通过imtool函数定位一个边界像素点；**
2. **并以这个点为起点，寻找周围的邻居点（邻居条件：在该点的邻域内，且之前没有访问过即：****）**
3. **找到的点都将它放入事先维护的一个栈（数组）中，每次取栈顶的元素重复上述动作，最后得到所有边界的像素点坐标；**

**做完上述步骤得到的结果是这样的：**



**而调用内部的bwtraceboundary函数得到的结果是这样的：**

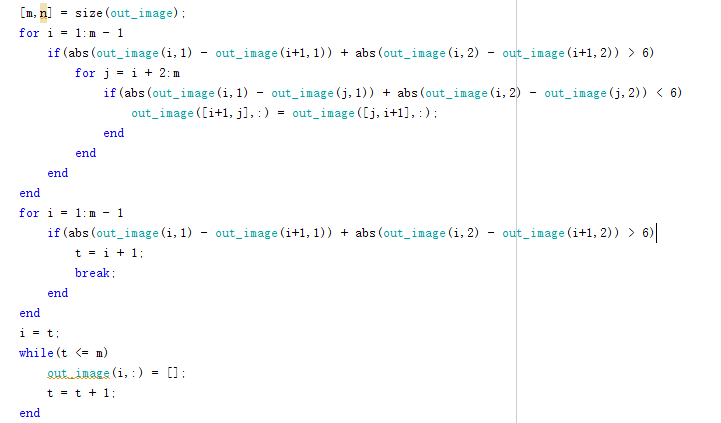




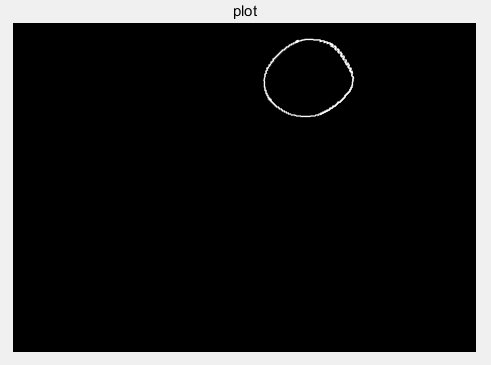
**很明显，边界内部的线是多余的，当我查看代码，终于找到了原因：plot函数是顺序连接每个像素点，但我返回的n\*2矩阵的像素点并不连续，于是就出现了边界以外的点；**

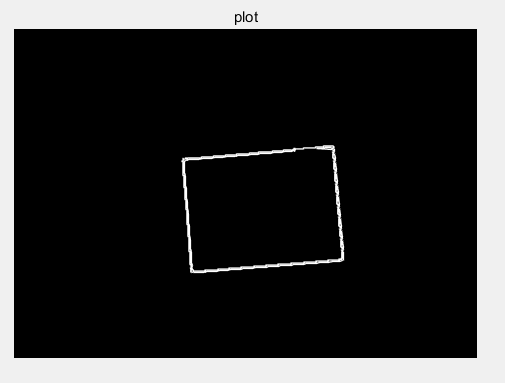
**知道了原因之后，我在my\_edgelinking函数返回结果之前，先对这个结果矩阵进行处理，也就是第4步。**

1. **处理返回的结果矩阵：处理的思路是这样的，遍历整个矩阵，如果出现前一个点和后一个点距离很远的情况，就从之后的点中找出一个与当前的点距离较近的点，交换这两个点即交换对应矩阵的两行；如果后面找不到一个与当前的点较近的点，那么说明后面的点全是边界某一块（之前已近形成这一块边界）的点，这样就把之后的点全部去除即可。代码：**



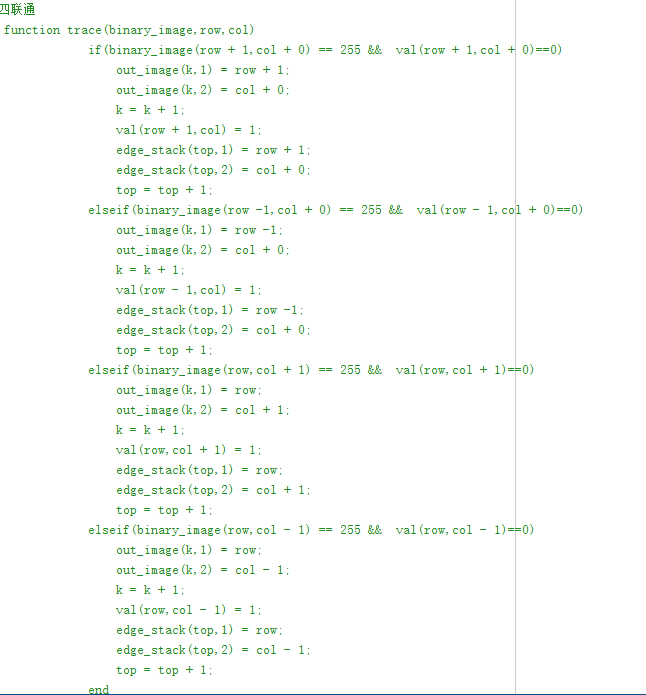
**这样处理下来得到的结果为：**





**令人欣慰的是，边界内部的线消失了。**

**四连通的方法就是将领域条件修改了一下：**



**实验心得**

**通过这次试验，尝试了多种边缘检测算子的实现，也自己实现了简单的边缘链接算法，对这一块内容有了更深的了解。同时，为了在边缘检测中更完美的做到边缘的精度也费了不小的努力，但最终的结果还是比较满意的，虽然对于不连续的顶点并不能完全的检测出来，但连通边界还是可以得到较为精确的结果。总的来说，这次的实验收获的东西还是很多的。**