**Canny算法实验报告**

报告人：何峙

**学号：**21215122

**专业：**大数据与人工智能

**实验题目**

* 对Code0代码进行封装 (学号尾数为2)；
* 增加一个函数：首先把相邻的边缘连成长的线条，并删除长度小于20的Edge。分析删除这些短Edge有何优缺点；
* 对算法的若干组参数，对所有测试图像进行测试，保存输出Canny算法每个主要步骤的结果图像, 并分析各参数对结果的影响；

**实验分析**

* Canny算法进行边缘检测的主要步骤

1. 生成灰度图像：

相关函数 void rgbToGray();

结果如下图：



1. 对灰度图进行高斯滤波，以去除图片中噪点对边缘检测的影响。

相关函数：void gaussian\_smoth(float sigma)

生成图像如下:



1. 用一阶有限差分计算梯度与幅值。

相关函数：void derivative\_x\_y()和void magnitude\_x\_y()

生成图像如下：



4．进行非极大值抑制。

相关函数：void non\_max\_supp()

生成图像：



5．最后，进行双阈值检测和连接边缘

相关函数：void apply\_hysteresis(float tlow, float thigh);

生成图像：



综上，使用Canny算法进行边缘检测结果如图：



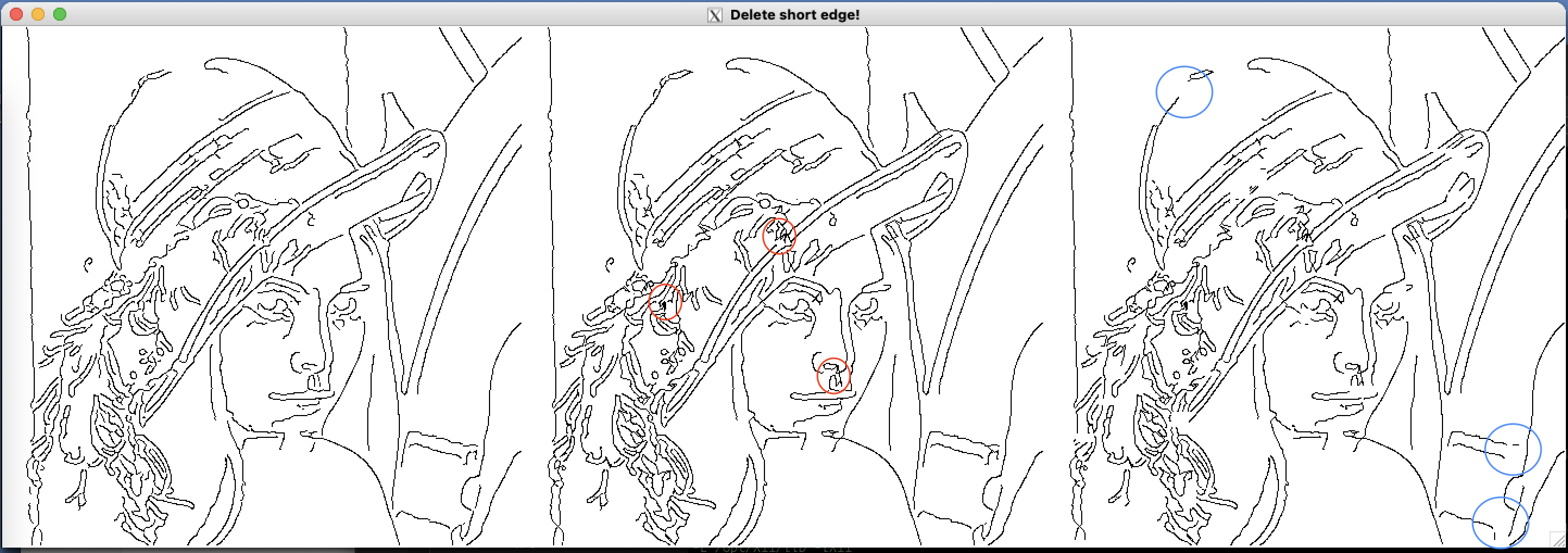
* 对相邻边缘连线后，再对删除长度小于20的短Edge的测试：

相关函数：

CImg<int> canny\_line(CImg<int> picture, int distance);

CImg<int> delete\_line(CImg<int> pirctur, int distance);

实验输出图如下：



其中左图是通常的Canny算法的输出图，中图是对相邻边连线之后的图，右图是删除长度小于20的短边后的图。

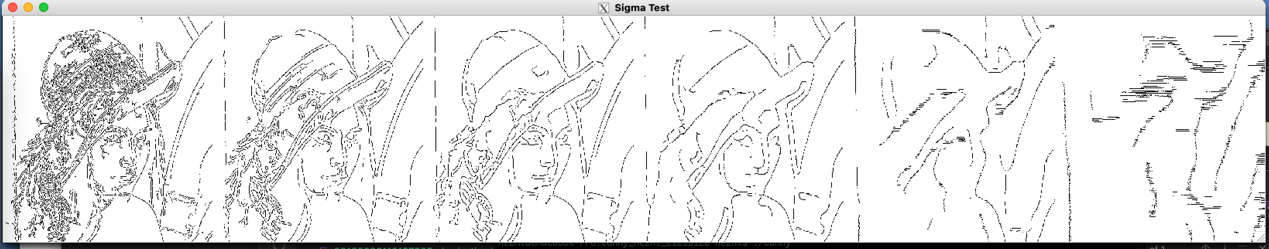
中图红色的标记是连线之后图部分的区别，可见相比于左图，边界区分更加明显，但随之而来也会带来很多不必要的噪点。

右图蓝色的标记是删除短边之后图部分的区别，明显可见渲染的像素更少，勾勒的线条会更加单一、直观，但可能会误删了一些图像的边界。

* 图像调参测试

1. 高斯滤波标准差调参测试

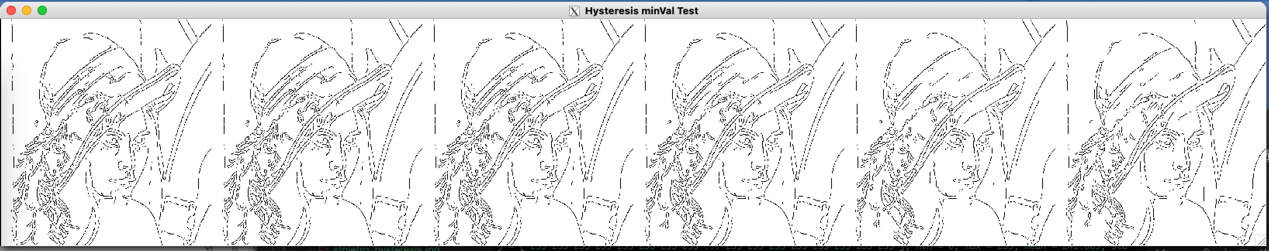
由左到右标准差分别为：1，2，3，5，10，20



分析：高斯标准差越小，对图像噪点滤波的效果就越差；如果标准差过大，模糊效果太强，边缘之间的很多细节又会给模糊掉，导致最终图片的边缘细节又不明显。

1. 低阈值调参测试

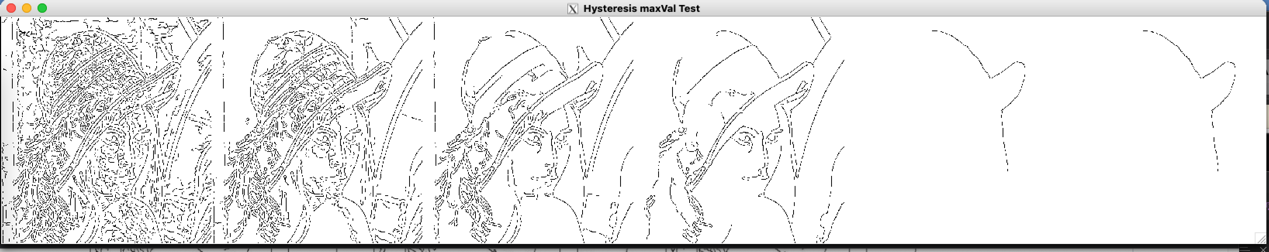
由左到右低阈值分别为：0.01，0.05，0.1，0.4，0.65，0.7



分析：低阈值越小，噪点越多；而低阈值增大，又会丢失很多强边缘像素。这是因为对于弱边缘像素（即低阈值和高阈值之间的像素）可以从真实边缘提取也可以是因噪声或颜色变化引起。为了获得更准确的效果，应该抑制后者引起的弱边缘像素。

1. 高阈值调参测试

由左到右高阈值参数分别为：0.4，0.6，0.8，0.9，1.0，2.0



分析：高阈值越小，就越多强边缘像素，因为大于选定这个阈值的像素点都将被确定为边缘；而高阈值越大，原来的强边缘像素大部分会转化为弱边缘像素，会丢失一部分像素点。这是因为被划分为强边缘的像素点已经被确定为边缘，它们是从图像中的真实边缘提取的。

*备注：由于图像过多，因此不在文档里一一列举生成的每个图像效果。图像详情可查看目录./resul\_Data/.*