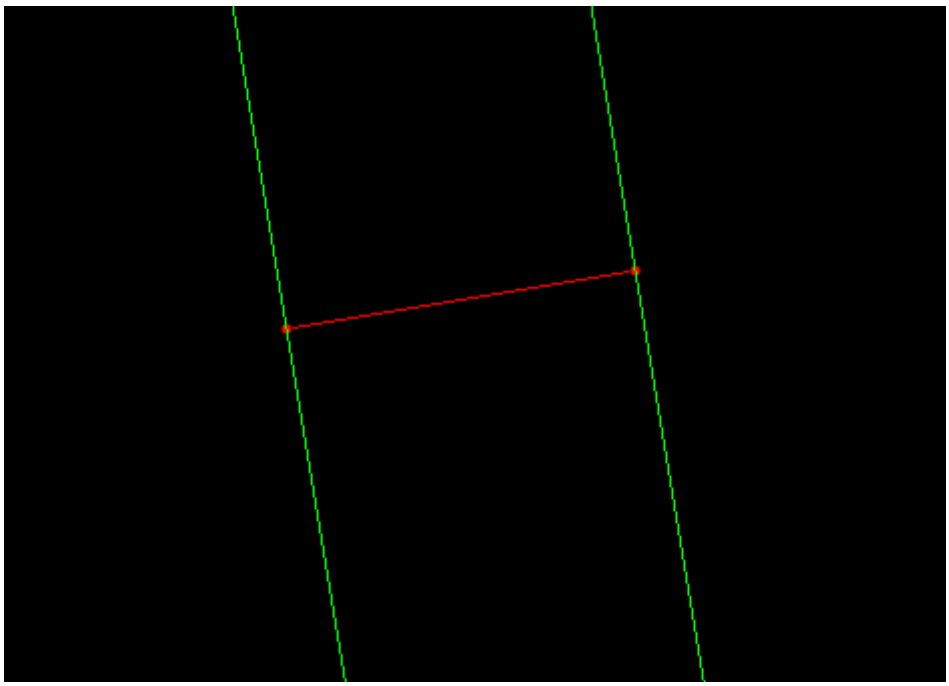
线段 Voronoi 图的扫描线算法的图形演示

一、 实验目标

本次实验我们用图形界面来演示如何使用扫描线算法来生成包含点和线段的 Voronoi 图。为了简化问题，我们对于线段加上了一个限制，即任何两条线段（包括点）都不在内部或者端点处相交。

二、 实验算法

本次实验我们采用 Steven Fortune 的扫描线算法。虽然上课时讲的是他的关于点的 Voronoi 图生成算法，但是他在论文中也提到了根据同样的原理，可以生成线段的 Voronoi 图，前提是线段不在内部相交。如果把线段看成是除去两个端点的一条只有内部的“开线段”，那么就可以和普通的点同样看待，如下图所示。我们在这次实验里采用了这种思想。在本文的剩下的部分，提到的“线段”如没有特殊说明，均指“开线段”。



图表 1 一条完整线段（红色）的 Voronoi 图

三、 数据结构

由于 DCEL 结构的良好的性质，我们在实验里继续使用 DCEL 结构来表达 Voronoi 图。

在本次实验的算法中，需要对 DCEL 结构进行操作的部分主要是处理基点事件和圆事件。在处理基点事件时，面和边的数目增加，顶点的数目保持不变；在处理圆事件时，顶点的数目增加，而面和边的数目保持不变。

四、 算法细节

虽然线段和点在原理上是相同的，但是在实际的实现过程中却比点要复杂得多。这主要体现在以下几个方面：

1、 线段形成的海岸线不是抛物线，而是折线。并且线段的海岸线之间的相交也比抛物线的相交复杂一些，因为折线与折线之间可能会有三个交点，也可能只有一个交点。不过大体上还是可以根据点的情况来进行处理。最终合理的交点只会有一个或两个，与抛物线的结果类似。

2、 线段的所谓的“外接圆圆心”，计算起来比较复杂，如果还有点的话，那么会出现多种情况，并且圆心通常不止一个。为了简化计算，这里把线段看成是直线。

a) 基点为三个点。圆心只有一个。
b) 基点为两个点和一条线段。圆心从 0 个到 2 个。
c) 基点为一个点和两条线段。圆心从 0 个到 2 个。
d) 基点为三条线段。圆心可能是 0 个，2 个或 4 个。

计算出的圆心中有的可能不合理，所以还需要验证。我们将这步验证和圆事件的验证放到了一起，具体做法是计算在圆事件的发生时间的时候，对应的海岸线的交点是否已经重合到了一起。

3、 点和线段之间的边也不是直线段，而是抛物线。在绘制边的时候需要注意，这条边是直线还是抛物线。

五、 遇到的问题

虽然通过课程的学习，我们对点的 Voronoi 图有了比较清楚的认识，但是在本次实验中，我们还是遇到了很多问题。主要是各种情况的处理问题。

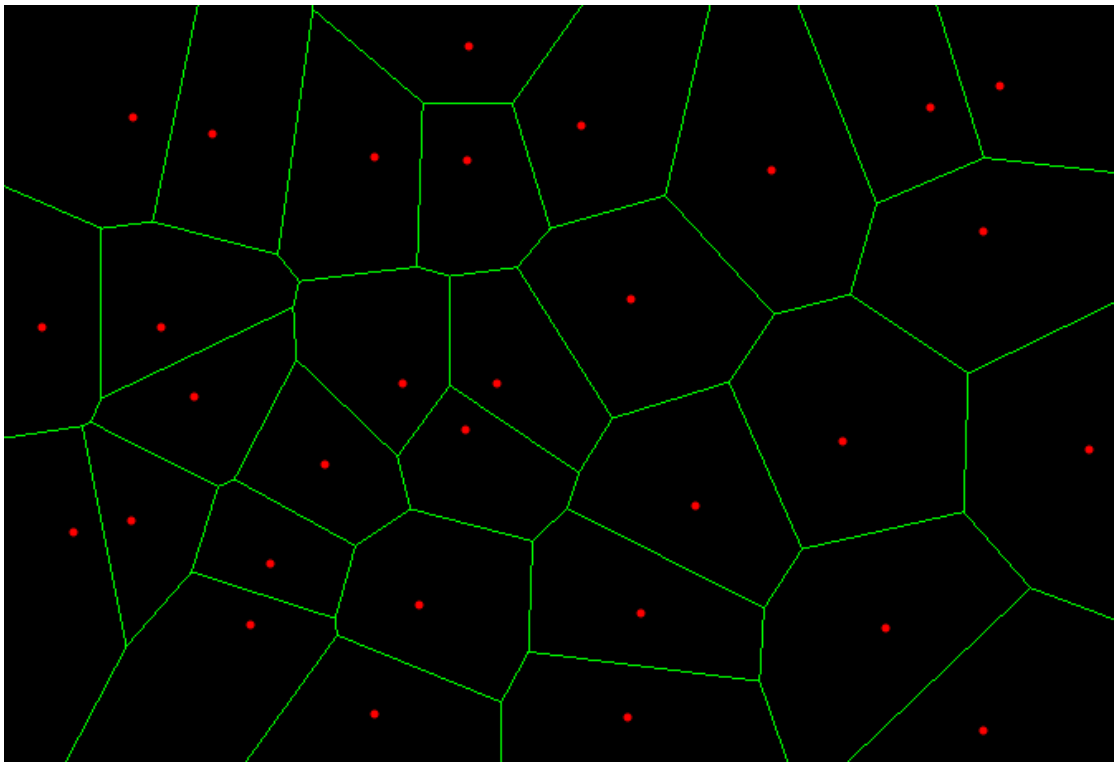
1、 运算精度问题。在程序中，我们使用双精度数进行运算。在判断两个数是否相等时，使用了容差进行判断。为了区别线段和他的两个端点，我们在添加线段时会将线段的两头进行适当的收缩，幅度在容差之内。在实际运算时，运算产生的误差可能会溢出容差。

2、 确定海岸线的交点。在程序中，需要求出某个海岸线上的边和下一条边的交点。根据两条边所在的海岸线可以求出它们所有的交点，问题在于确定是哪个交点。对于抛物线，一般有两个交点，可以根据两个基点的先后位置来判断；对于折线，因为可能有三个交点，所以有点复杂。在实验了多种方法之后，我们还是回到了最初的方法上来。因为线段是互不相交的，所以它们实际生成的海岸线，最多只能有两个交点，另外的交点不可能达到。因此，通过简单地判断线段的先后顺序，也就可以得到交点的位置。不过，在后来的测试中我们发现，某些情况下的处理应有问题，所以我们综合了线段间相互位置的判定，即线段在另一条线段的上方还是下方的方法，通过使用这两种方法，实验结果的出错程度也下降了。但是理论上缺乏论证，所以实现的健壮性仍有缺陷。

3、 退化情况等较特殊的情况。在本次实验的实现中，对退化情况的考虑不是很周全。因为主要集中于主要算法的理解和实现上，所以这方面的健壮性较差。

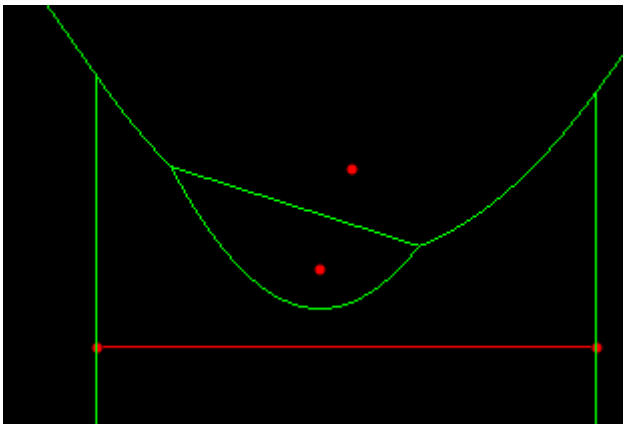
六、 实验结果和分析

1、 点的测试

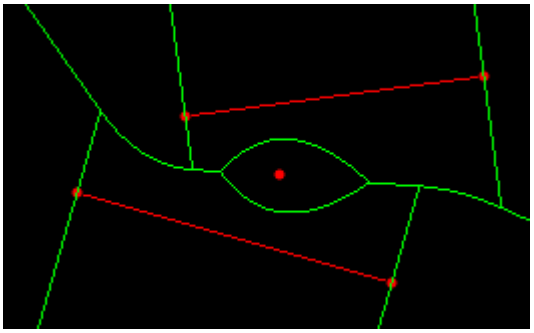


图表 2 点的 Voronoi 图

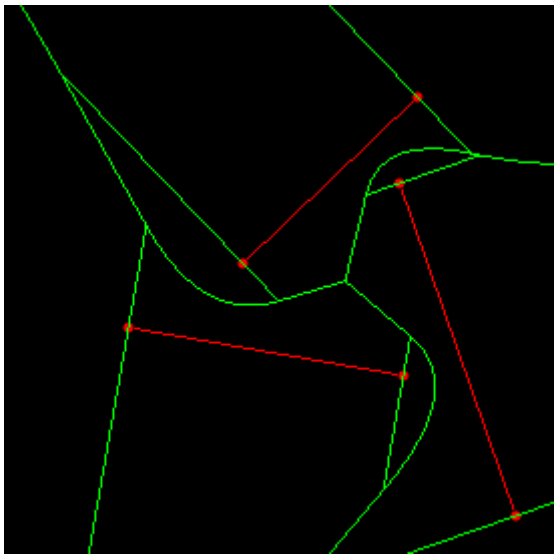
2、 圆心计算的测试



图表 3 两个点和一个线段

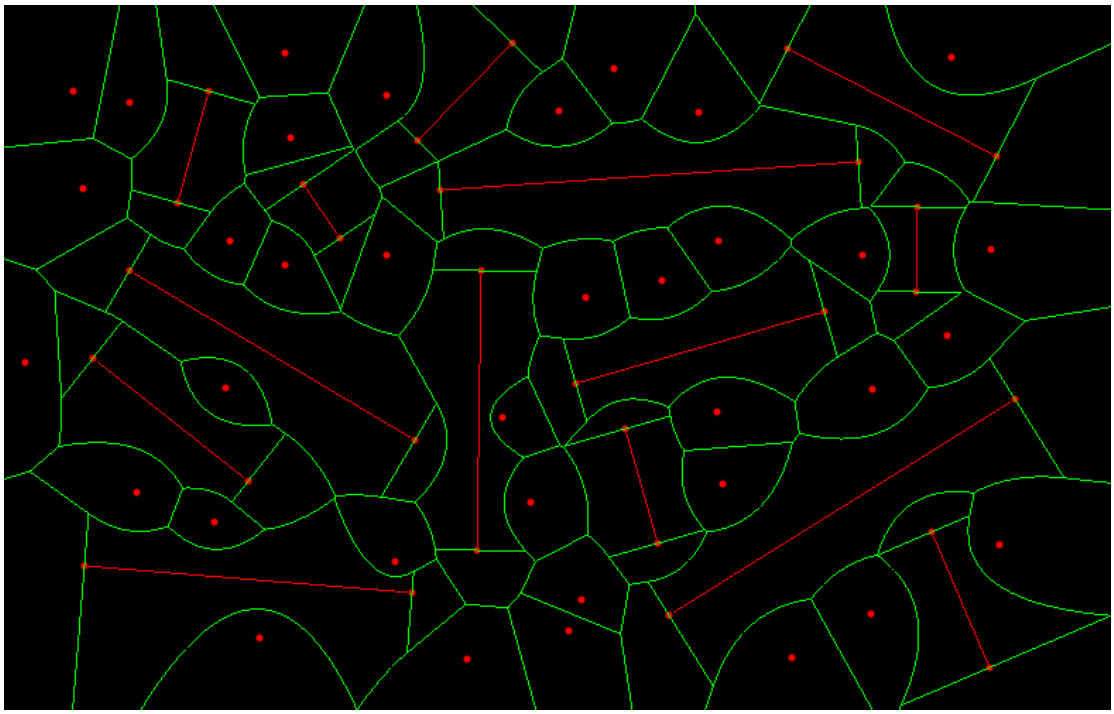


图表 4 两个线段和一个点

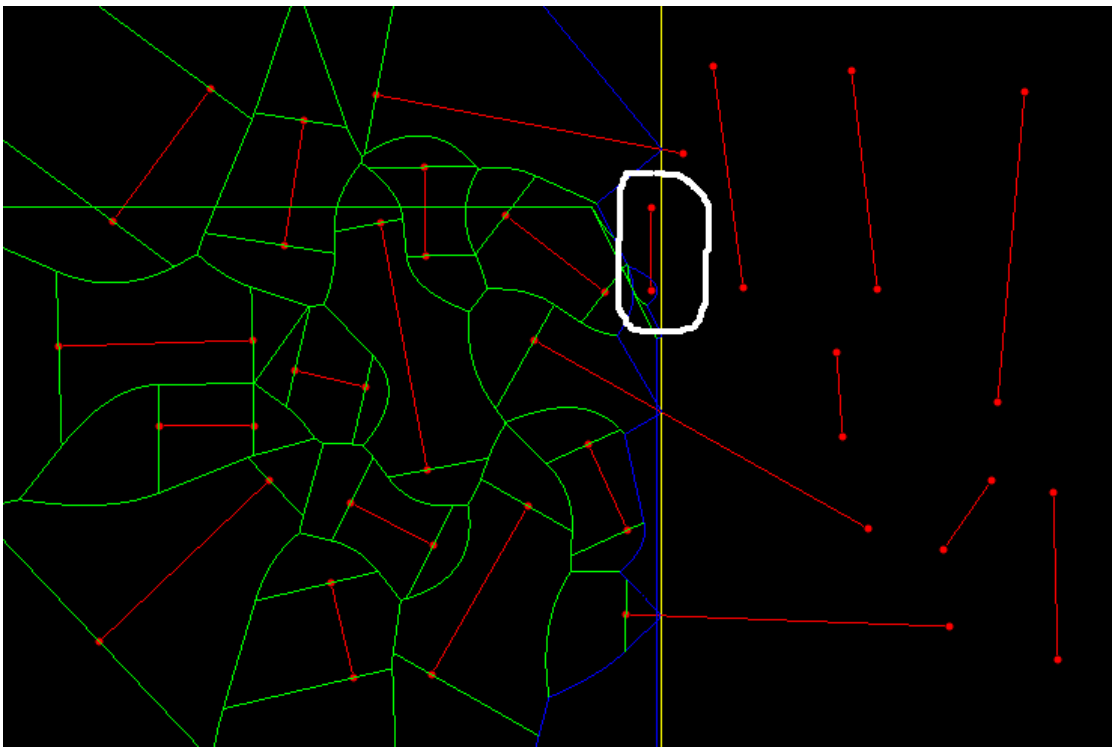


图表 5 三条线段

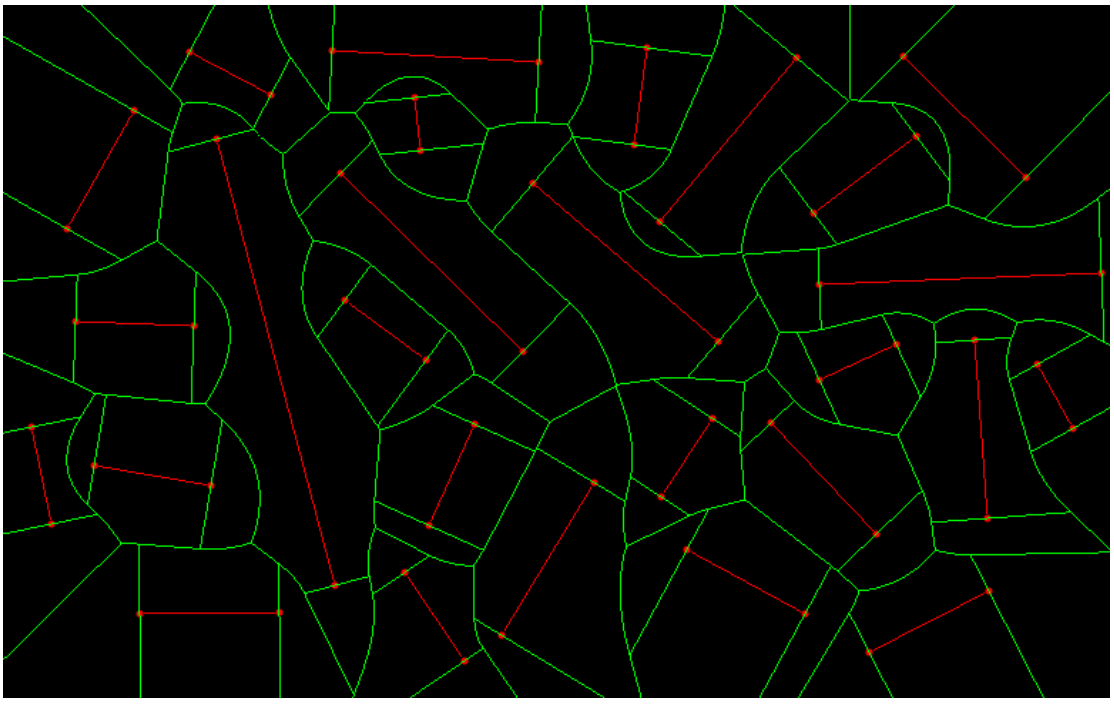
3、 健壮性测试



图表 6 点和线段混合



图表 7 竖直的线段（白色）容易发生错误



图表 8 全部是线段的 Voronoi 图

从实验结果可以看出，对于圆心的计算大致上是正确的，不过对于退化情况的处理还是不够周全。因为本程序的主要目的在于演示，并且文件的输入输出还没有实现，所以暂时不能进行大规模输入的测试。

七、 编译连接说明

本程序使用 Microsoft Visual Studio 2005 开发，打开工程文件 SegmentVoronoi.sln 后就可以生成程序了。

八、 参考文献

1、A Sweepline Algorithm for Voronoi Diagrams, Steven Fortune

2、Computational Geometry: Algorithms and Applications, Mark de Berg, et al.

3、海岸线的相交部分的有关抛物线的代码，我们借鉴了以前的张森，施侃乐等师兄的大作业的代码，再此向他们表示感谢。