平面点集的三角剖分

清华大学计算机科学与技术系
忻海 2009210965
张方略 2009310461
李冉 2009211014

实验内容

平面点集的三角剖分（Triangulation），是计算几何中一项十分重要的技术。我们实现三角剖分的主要步骤有：对读入的平面点集，基于分治策略（divide-and-conquer），以 zigzag 的方法完成该平面点集的三角剖分。而在剖分过程中，我们将整个结构中的顶点、边和三角片以 DCEL 结构的方式储存起来，从而完成对剖分所得到的三角网的一个完整的表示，并添加了可以展示 DCEL 结构优越性的实验内容。

三角剖分与 DCEL 结构

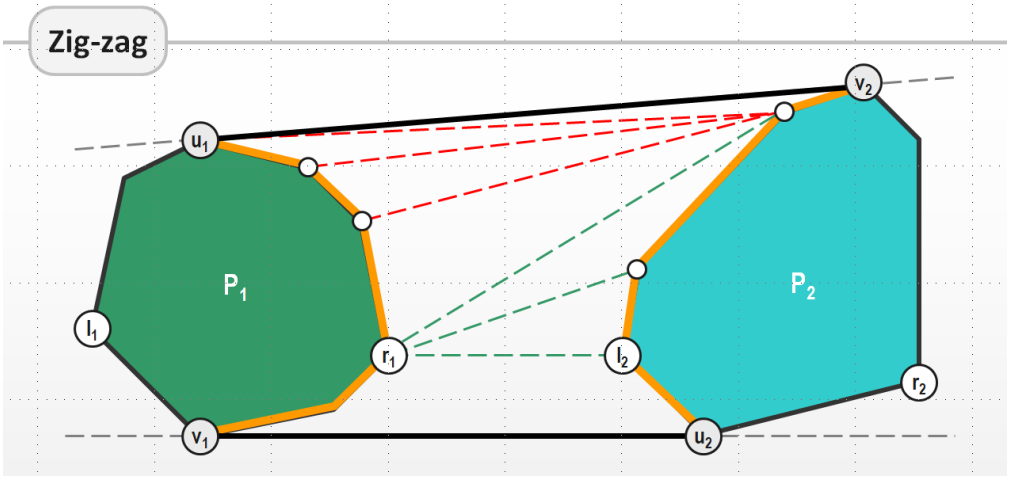
点集的三角剖分定义：

假设 V 是二维实数域上的有限点集，边 e 是由点集中的点作为端点构成的封闭线段, E 为 e 的集合。那么该点集 V 的一个三角剖分 T=(V,E)是一个平面图 G，该平面图满足条件：

- 除了端点，平面图中的边不包含点集中的任何点。
- 没有相交边。
- 平面图中所有的面都是三角面，且所有三角面的合集就是点集 V 的凸包。

算法描述

算法基于 divide and conquer 策略，每次将点集分为左右两部分，分别计算左右的三角剖分以及凸包。之后将左右部分用 zigzag 的方法组合起来，得到中间部分的三角剖分，并更新凸包。如图所示：



图：算法示例(摘自课件)

r_1 为左凸包的最右点，l_2 为右凸包的最左点。从这两点开始分别向上和向下做 zigzag。直到找到 u_1v_2 和 v_1u_2，并更新凸包。

复杂度分析

算法首先需要对点集排序，$O(n \log n)$。之后，每次合并左右两部分操作，最多为$O(n)$。这部分时间为：

$$\therefore T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

根据 Master Theme

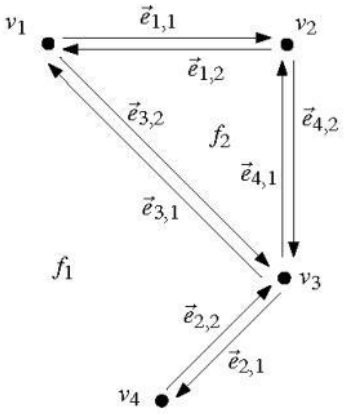
$$T(n) = O(n \log n)$$

所以总时间为$O(n \log n)$

三角剖分结果的 DCEL 结构表示

DCEL（Doubly-Connected Edge List），是一种有效的存储二维曲面拓扑信息的数据结构。

如图所示：



图：DCEL 结构（摘自《计算几何—算法与应用》）

在 DCEL 结构中，每个网格中的边，都是由两个被称作半边（HalfEdge）的有向边结构组成的，如（e31，e32）。整个网结构由顶点对象（Vertex），半边对象（HalfEdge）和面对象（Face）三种结构组合而成。

其中顶点对象要包含一个指针 leaving，指向一条以自身为起点的半边。

半边对象要包含如下几个指针：

- twin：指向和自己端点相同而方向相反的另一个半边，如（e31，e32）
- next：顺着自己的方向的以自身终点为起点的半边，如（e32 -> e41）
- face：按照自己方向的左侧对应的面，如（f2：e32，e41，e12）

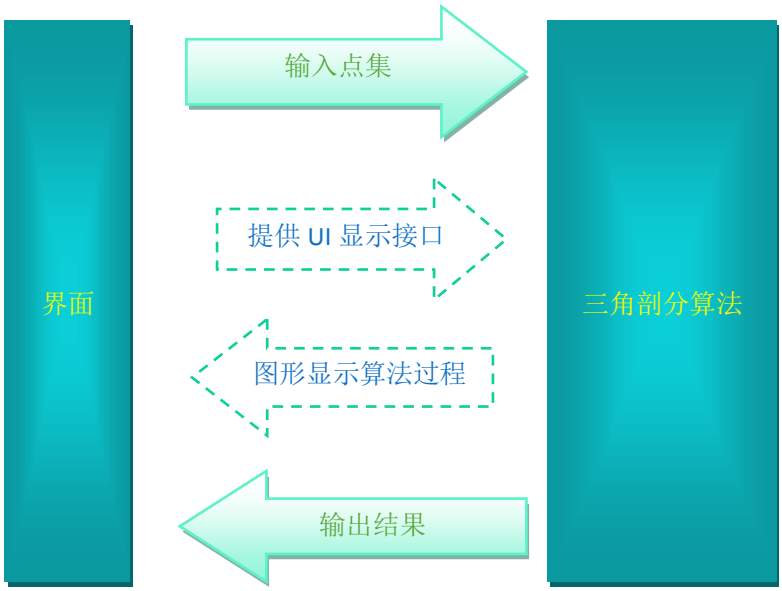
面对象包含指针 edge，指向以自己作为面的一条边。

三角剖分实验系统设计

我们的三角剖分实验及演示系统主要分为如下几个模块来设计实现的：

- 用户界面及交互模块
- 三角剖分核心方法

总体结构如下图：



图：程序构架

界面和算法独立。界面输入点集，并提供可选的图形显示接口（不提供显示接口则无图像输出）。算法部分输出最终剖分结果，并可提供算法运行过程动画。

换一套界面不影响算法的工作。

程序功能

为了使本实验系统的交互更加直观，同时也能更好的演示算法的鲁棒性与高效性，在设计中我们保留了两种用户交互方式：

1、 文件交互方式：

用户可以通过载入写有平面点数据的文本格式文件（*.txt），其格式如下例：

100;100

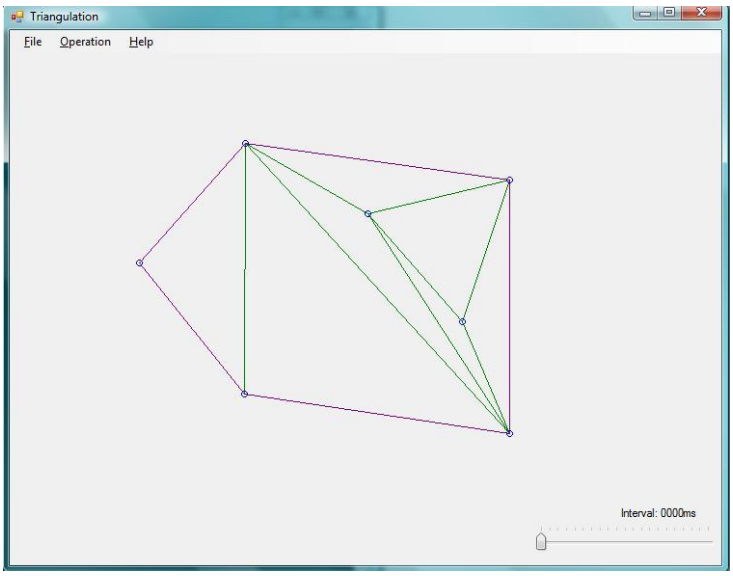
200;200

111;111

在菜单上点击 Triangulation 进行三角化，即可在屏幕上得到三角剖分后的图形结果。如需要将结果保存，可点击 Output 对分割结果进行存储

2、 图形交互方式：

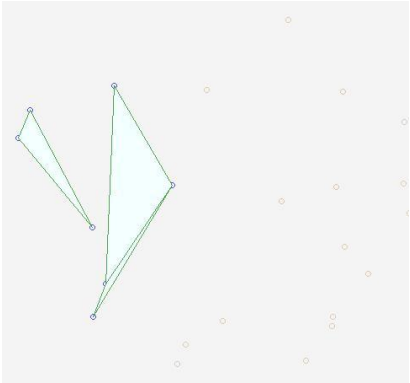
为了使用户可以自由地进行点数较少的实验，同时用户想要得到一个直观的三角剖分的感受，可以采用图形输入的方式。用户可利用鼠标在屏幕上点出若干个任意位置的点，然后点击 Triangulation，可直接观看剖分结果。用户也可以将自己所在屏幕上点出的点存储为相应的文本文件，方便下次演示时使用。



图：程序示例

动画显示算法过程

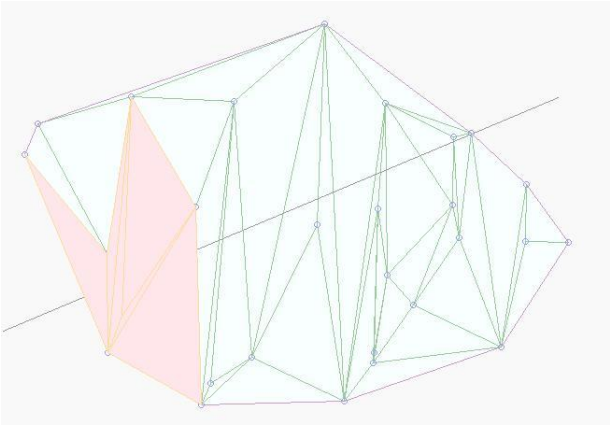
我们提供了剖分计算间隔（Interval）的设置，用户可以通过设置每个步骤的间隔来观看我们算法每一步的进行，从而对实验过程有更深刻，更直观的理解与感受。



图：程序动画示例

直线穿过三角面片的计算

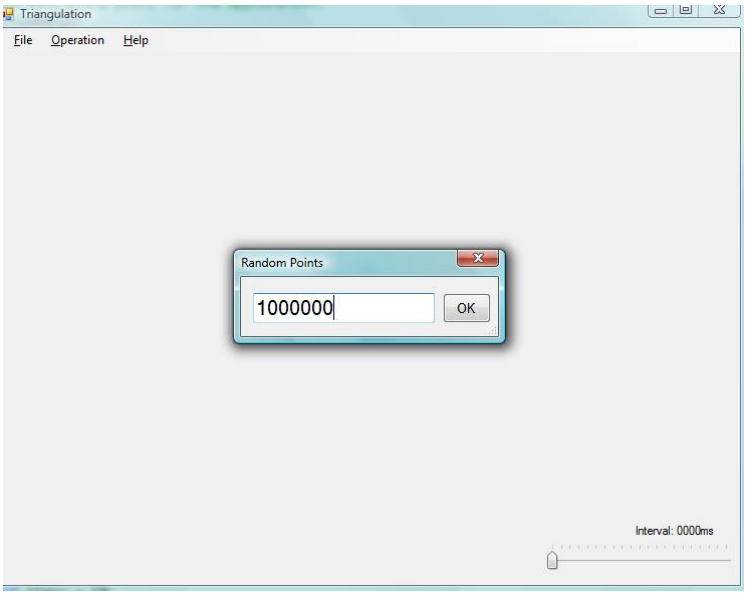
用户可以右键点击设定穿过的直线，程序给出利用 DCEL 结构查询直线穿过的三角面片的动画。



图：直线穿过三角面片的计算

随机点集测试速度

用户可以随机一定数量的点集，程序将会给出运算时间和文本结果，但不在界面上显示。



图：随机点示例

三角剖分算法设计

主体：

算法主体部分是一个递归过程：每次将输入的点集分为左右两部分，分别对左右部分作三角剖分。而后将左右部分的结果用 zigzag 方法合并，得到整体的三角剖分以及凸包，并返回。

当输入点集大小为 2 或者 3 时，直接返回两个半边或者一个三角形。

ZIGZAG 过程：

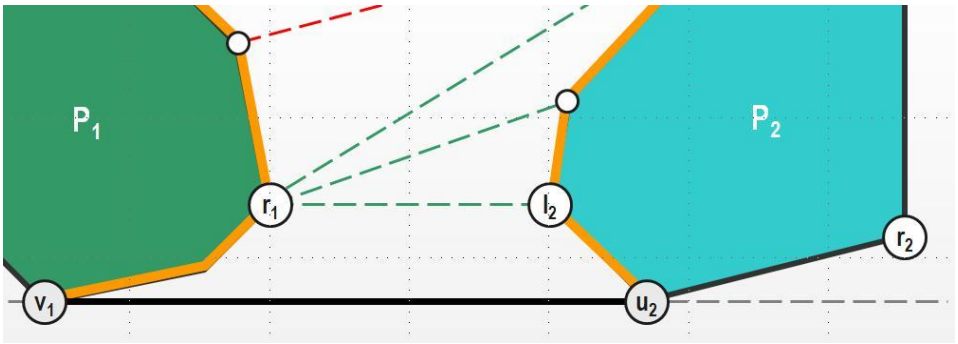


图 4：zigzag 过程示意

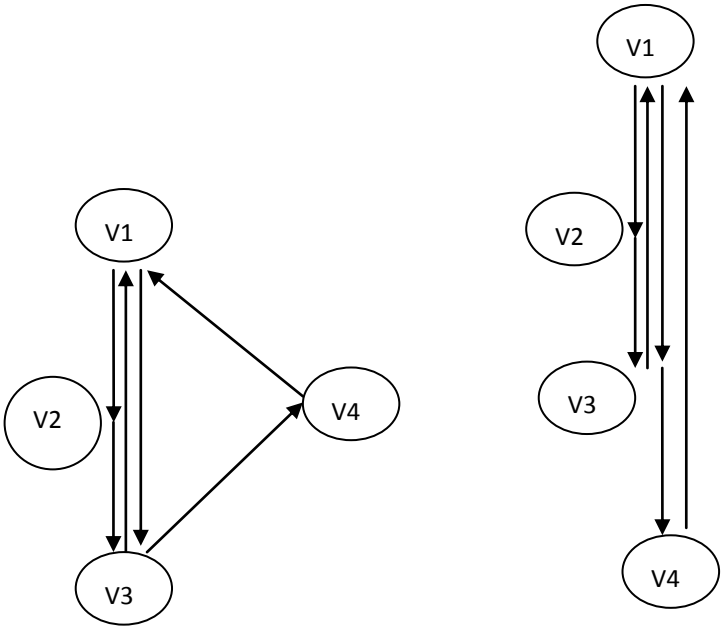
如图，从左右凸包的最靠近中间的点(r_1, l_2)上开始 zigzag。随意挑选左右某条边上邻近的第三个点(如 u_2)，并用 toLeft 方法判断是否可以构成三角形。是则加入新的三角面片(r_1, l_2, u_2)，并更新 zigzag 工作点(r_1, u_2)。否则换一条边，例如(r_1, u_2, r_2)不能构成三角形，则开始左边的探索。

当左右都到达临界状况时，如(v_1, u_2)，表明到达凸包边界，更新凸包。并回到(r_1, l_2)，开始向上做同样的 zigzag

退化情况处理：

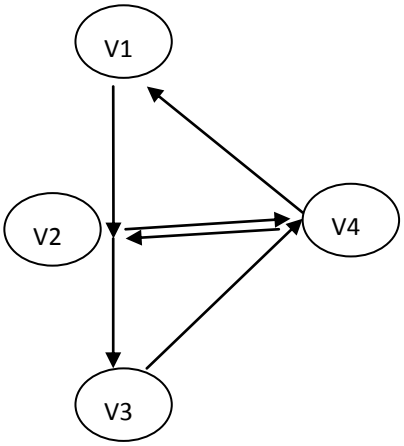
由于几何位置种类很多，直接处理很麻烦且容易出错，实现中选择延时处理。递归剖分过程中，认为共线的三点组成一个三角形，并记录在退化三角形中，其他处理完全同于非退化三角形。全图剖分完成后，集中处理退化三角形。分为两类

类型 1：



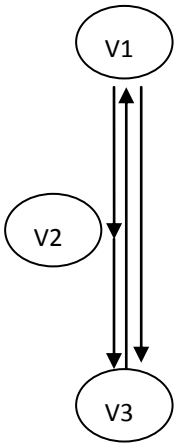
图：退化情况处理

如左图，(v1,v2,v3)组成退化三角形，同时(v1,v2,v4)组成三角形，这是必须的，否则会出现右图的情况。此时修改数据，将结果变为三角形(v1,v2,v4)和(v2,v3,v4)即可。

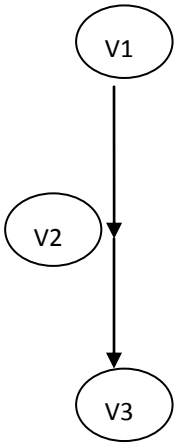


图：退化情况处理

类型 2：



如图，(v1,v2,v3)组成退化三角形，但 e13 是凸包上的边，即 e13 不组成三角形。此时应该移除半边 e13 和 e31，将 e12，e23 置为凸包的边。即：



图：退化情况处理

重复上述过程，直到不存在退化三角形。

DCEL 结构的存储与使用

算法中使用的 DCEL 结构如下：

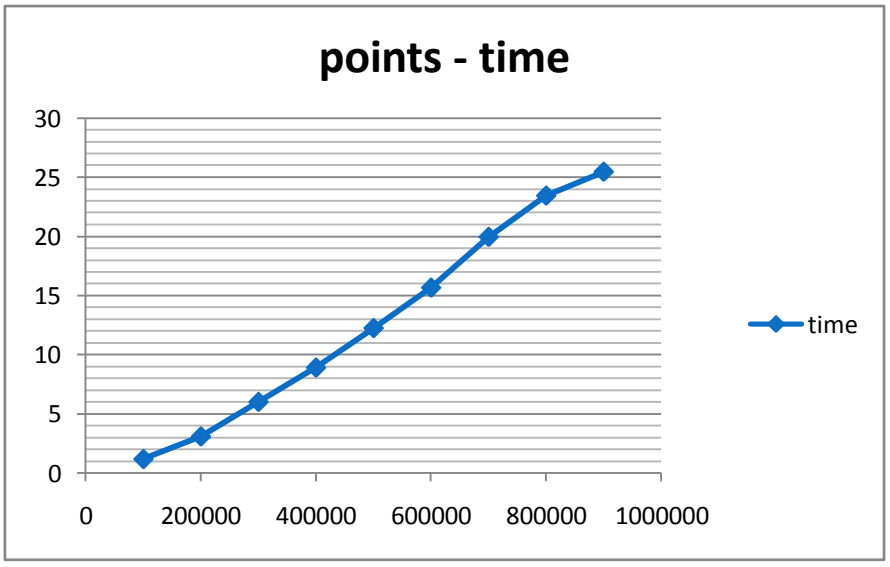
```
public class Face
{
    private bool _degeneration;
    public List<HalfEdge> edges;
}
public class HalfEdge
{
    public Vertex start;
    public Vertex end;
    internal HalfEdge twin;
    internal Face innerFace;

    internal HalfEdge next;
    internal HalfEdge pre;
}
```

每个 Face 中记录围成的 HalfEdge，并记录是否为退化三角形。

每个半边中记录起点 start，终点 end，twin，对应 face，后继边 next，前驱边 pre

系统测试结果

测试平台为 Vista，CPU: Q9400, 2.66G*4



图：点集大小与计算时间

如图，横坐标为点集大小，分别为 100,000 到 900,000。具体时间如下

points	time
100000	1.18
200000	3.091
300000	6.009
400000	8.911
500000	12.249
600000	15.665
700000	19.954
800000	23.44
900000	25.459

表格：点集大小和计算时间

可以看出时间小于$O(n^2)$。准确为$O(n \log n)$。

实验总结

通过这次实验，我们更好地掌握了三角剖分的原理与方法，并对 DCEL 结构的优点有了充分的认识。这次实验给我们计算几何课程的学习打下了一个好的基础，这使得我们在接下来的学习中有了更浓厚的兴趣与信心。