利用 Delaunay 三角剖分实现空间散乱数据点三维重构的区域增长算法

苏延超(2006310435)　杨欣(2006310441)　姚邦鹏(2006210773)

1. 引言

三角剖分是计算几何的研究重点之一。平面点集的三角剖分问题，是指在有限平面内点集内的点，按照一定的方式连接起来（连接两个顶点的边被称为对角线），成为互不交叉的三角网，同时要求在剖分完毕的图上不可能再增加新的对角线。扩展到三维的情况，由于四面体是三维空间的单纯形，相比其它类型的多面体，不仅描述方便，而且更适于对复杂问题作简化处理，因此，在计算机图形学、模式识别、曲面逼近、有限元网格生成等领域都得到了广泛应用。

在各种三角剖分算法中，**Delaunay** 三角剖分应用最为广泛。所谓 **Delaunay** 三角剖分，是指具有“三角剖分最小内角最大”性质的剖分。对于三维的情况，三维点集 S 的三角剖分应该是覆盖 S 的凸包的互不重叠的四面体的集合。可以形式化描述如下：

记二维点集 $S=\{s_1, s_2, \dots, s_n\}$，其凸包为 $CH(S)$，假设 $T=\{T_1, T_2, \dots, T_n\}$是经过三角剖分后生成的三角形集合，则 T 满足：

（1）T 中所有四面体的顶点都属于 S ；

（2）T 中任意两个四面体或者不相交，或者交集为一个顶点或一条边或一个面。

特别地，**Delaunay** 三角剖分满足“空园性质”，即：T 中任意一个三角形 T_i 的外接圆内不包含 S 中除 T_i 以外的任意一点。另外，早在 1985 年已经有人证明：如果 S 满足任意 4 点不共园，则由 **Delaunay** 三角剖分得到的 T 唯一（[1]）。

在教材（[2]）**Delaunay** 三角剖分的引言部分，用三维地表重构引出了 **Delaunay** 三角剖分。在地质勘测中，三维地表重构有着现实的应用意义。而另一方面，我们可以从 3D 扫描仪中获得实际物体表面的点云，如何从这些点云恢复出原来物体的形状也是一个热门问题。抽象而言，它们都属于三维散乱数据曲面的重构问题，现有的解决方法大致可以分为整体拟合和局部拟合两种。

整体拟合通过隐式或显式重构曲面方程得到曲面的解析表示；局部拟合从离散数据的

三角形网格出发，以三角形为单位分别构造出分片的插值曲面。相比之下，在对复杂曲面进行造型与加工时，前者缺乏柔性，而后者则十分适用。

局部拟合可以采用的方法有很多。一种是采用基于 三维 **Delaunay** 三角剖分的方法实现。基本做法是：首先对散乱点进行 **Delaunay** 三角剖分，然后从结果中分离出需要的三角形片，最后将三角形片连接在一起构成网格曲面。这类算法最明显的优点在于可以避免曲面自交，但存在不足：一方面，计算量较大（如相比区域增长算法）；另一方面，现有工作多关注于三角形片的选取，而忽视三角形片的拼接。

而对于某些性质较好的曲面，也可以将其投影到一个二维平面上，先在在二维平面中进行三角剖分，得到这些点在平面上的拓扑关系，从而恢复出它们在三维中的拓扑关系。这种算法把问题从三维降到了二维，实现较为简单，但是这种算法应用的范围有限，它要求对于输入三维点云，能够找到一个平面，使得该点云在这个平面上的投影的拓扑结构与原来的点云相同，对于某些数据（如三维地表）它可以很好的解决，但是对于一般物体表面则难以实现。

本实验从三维物体表面重构的实际应用出发，以空间散乱点的三维重构问题为背景，利用 **Delaunay** 三角剖分解决这一问题。本次实验我们采用的是一种区域增长算法—波前算法对空间点云实现三角剖分，从而恢复点云的拓扑结构。

2. 波前算法的定义以及原理

2.1 定义

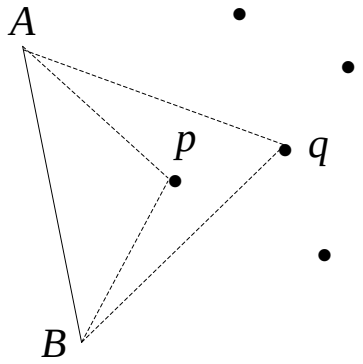
波前算法的目标是对点阵实现 Dealunay 三角剖分。该算法是指：通过任意一点开始，获得一个满足条件的初始三角形作为剖分的起点，以初始三角型的外环作为初始波前，寻找当前波前边的有效匹配点构成新的三角形，同时不断修正波前，并将当前波前边向后推移，直至所有点剖分完毕。

上面定义中的“波前”是指在剖分进行过程中的一系列的当前边，即等待剖分的边集；波前在扩展的过程中，每次只对一条边进行处理，这条边被称为“当前波前边”，所找到的点称为匹配点。

2.2 原理

波前算法的核心原理是 Delaunay 三角剖分的“最小内角最大”原则。波前算法在对每一条边进行扩展时，首先考查待匹配点的可行性（如该点是否已被扩展完毕、新三角形与当前边所在三角形的曲率变化等等），在可行的点中，按照“最小内角最大”的原则进行匹配。按照这一原则，保证匹配的所有点几乎都可以满足 Delaunay 三角剖分的条件。

但是，我们的算法中并没有严格按照“最小内角最大”的情况进行处理，而是要求匹配点与待扩展的当前边的 2 个端点构成的夹角应该尽量大。如下图所示，实线部分为当前波前边，图中的点是待匹配点。如果按照“最小内角最大”的原则，匹配点是 q 点。但在实际操作中，我们选中的是 p 点。尽管 ΔApB 的最小内角小于 ΔAqB 的最小内角，但∠ApB 大于∠AqB。实际上，如果先匹配了 q 点，由于 p 落在 ΔAqB 的内部，最终很有可能得到与先扩展 p 相同的结果。但先扩展 p，可以使波前沿着相同的方向前进（AB→p→q）；而若先扩展 q，则会使波前扩展的方向出现很多变化。



因此在波前扩展的过程中，我们并没有严格遵守波前算法的定义。但在绝大多数的情况下，该算法都可以得到和严格定义相同的剖分结果，但演示效果大大增强了。（PS：我们曾试图从数学上严格证明这两种匹配方式之间是否等价，但学识和精力所限，我们没有得出最终的结论）在下面的描述中，我们称这种原则为“扩展角最大”原则。

2.3 邻域搜索原则和算法复杂度分析

Delaunay 三角剖分的一般方法（课件上所述）的时间复杂度是 O(nlogn)。而波前算法需要扩充所有的点，并且在每匹配一个点（扩展一条边）时都需要考虑所有的点，因此波前算法的算法复杂度为 O(n2)。

为了提高算法的计算效率，我们对每一个数据点建立 k 邻域。对每一条边进行扩展时，

只考虑该边的两个端点的 k 邻域。另外，通过分块搜索，k 邻域也可在近似 O(n)时间内完成。这样，整个算法可在近似 O(n)时间内完成。

3. 数据结构及算法

与选题报告中不同，我们没有将三维数据投影到二维空间上，而是直接在三维空间中进行剖分操作。因此从数据结构到算法，在某些细节上都与选题报告中有所不同。因此此处详细介绍一下最终版本的程序中所选用的数据结构以及算法。

3.1 数据结构

我们使用了 Double-Connected Edge List (DCEL)结构来进行剖分操作以及结果存储、演示和图形的绘制。该结构由三个互相关联的类构成：

点结构（Vertex）。该结构存储了空间中一顶点在 x、y、z 方向上的坐标，以及一个链表，该链表中存储了所有的以该点为起点的边 incEdges。在初始情况下，对于所有的点，incEdges 均为空；随着算法的进行，新生成的边会加入到对应的点的 incEdges 中。

边结构（Edge）。DCEL 结构最大的特点是其中的边是有方向性的，这样每一条边都有一个孪生边 twinEdge。另外每一条边都对应一个三角形 nTri 以及起始点 oriVtx。在三角形内部，根据三条边之间的方向的关系，每一条边都有一个 prevEdge 以及 nextEdge。对于每一条边，用 isWF 标记当前边是否是波前。

三角形结构（Tiangle）。DCEL 结构中的每一个三角形只需要存储一条边即可，这条边的 prevEdge 以及 nextEdge 为该三角形的另两条边。

DCEL 结构的好处有：（1）对整张图，在满足拓扑连续性的条件下，通过 DCEL 可实现线性时间的遍历；（2）由于本算法在实现的时候需要考虑很多边界条件，在考虑这些条件时需要考查与当前边、当前顶点关联的所有点、边、三角形等。在 DCEL 的结构下可以很方便地找到与当前结构关联的其他数据结构，既实现了查找的方便性、又实现了查找的快速性。

另外，在 Java 下，所有的结构都只需要占据一份存储空间，其他与之相关的结构都只是存储的索引记录。因此，使用 DCEL 结构不会带来额外的空间开销。

3.2 算法流程

a. 数据预处理

对于输入的点集，我们需要对其中的每一个点都找到与其距离最近的 K 的点作为波前算法中的候选点。如果简单的采用全局搜索的方法，至少需要 O(n2)的时间复杂度。而一般点集的 K 近邻问题我们可以用 KD-Tree 的方法来实现。但是，在我们研究的问题中，我们的输入点集是有一定的特殊性的：所有点应该近似在一个二维流形上，而这些点的分布应该是大致均匀的。而对于每个点的 K 近邻的搜索应该在这个点的一个临域范围之内，因此，我们采用了基于空间分块的局部搜索算法来加速 K 近邻的搜索，具体算法如下：

Step1：统计各个点的坐标，求出包围它们的与坐标平面平行的最小长方体

Step2：根据 K 和点的个数将该长方体平分为若干个子立方体，并计算各个点所属的立方体。

对点集中的每一个点，初始化当前搜索的立方体为该点所处的子立方体

Step3：计算该点到当前搜索立方体的六个面的距离最小值 dshort

Step4：计算当前搜索立方体中所有点到该点的距离并找出最小的 k 个

Step5：如果已找到 k 个近邻并其中最大的距离小于 dshort，则当前点的 k 近邻已经找到，如果没有找到 k 个近邻，或 k 近邻中某些点到当前点的距离大于 dshort，则将当前搜索立方体和与其相邻的子立方体合并为一个新的搜索立方体，转 step

因为我们处理的点分布比较均匀，因此我们可以近似的保证每个子立方体内的点数为 O(1)，而对于每个点的搜索的范围也不会超过 O(1)个子立方体，所以这个算法在处理我们所需的点集时的时间复杂度是近似 O(n)的

b. 第一个三角形的生成

以导入文件的第一个点为起始点，寻找与该点距离最近的点，并将这两点的连线作为第一条波前边；寻找第一条边的有效匹配点构成初始三角形。初始波前即为该三角形的三条边。

此处的匹配由于没有其它点和边的干扰，只需要考虑“扩展角最大”这一个条件。另外，由于使用 DCEL 结构存储，每条边均有一条孪生边与其对应。初始波前为处始三角形边的外围，内部的孪生边已经不再是波前。下面的扩展正是从这三条边开始往外展开的。

c. 波前的扩展&匹配点的选择

所谓波前的扩展，是指对于当前波前边，找到一个与之相适应的最佳匹配点扩展新的三角形，并更新波前。这可以分为两部分：首先找到一个最佳的匹配点，然后更新波前。

为了实现 O(n)复杂度的算法，匹配点只能在当前波前边的两个顶点的 k 邻域内选择。匹配点的选择要综合考虑如下因素：（1）匹配点不能是当前波前边的两个端点，也不能位于当前波前边的孪生边所对应的三角形内。（2）匹配点不能和当前波前边位于同一条直线上。（3）新加入的点的引入，不可重新扩展已经处理完的边，也不可使得已经被处理完的边（已经位于两个三角形中）属于第三个新的三角形。（4）“扩展角最大”原则，即匹配点与当前波前边的两个端点构成的夹角应尽量大。（5）匹配后生成的新三角形相对于当前波前边的孪生边所属的三角形的法矢量变化不能太大。（6）如果有多个点同时满足上述条件，则在当前波前上、并且相邻于当前边的点有优先权。

对于上述原则中的某些方面，解释如下：

第（3）点：这一点可确保剖分的单调性，保证已经处理完毕的点不会被重新处理。另外，这一点的实现往往需要遍历波前和所有已经生成的边才可实现，效率不高。但在 DCEL 的结构下，只需要访问从待匹配点出发的边的状况即可。DCEL 结构在此处大大提高了计算效率。

第（4）点：算法在执行时，不但要求“最小内角最大”，并且所选出来的角也要大于一定的阈值方可扩展。

第（5）点：曲率变化大的地方往往是两个复杂平面的交界处或者是边界，在这些情况下停止搜索，然后继续下一条边的搜索。这样可以避免剖分中因为曲率变化很大所引起的不良情况。程序中通过对两边的夹角设置阈值来实现这一点。

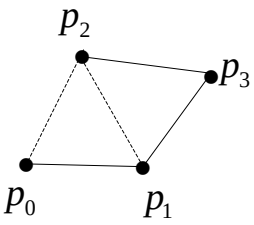
另外，在匹配时有可能遇到匹配点和当前波前边两个端点的连线均在波前中。这实际上是三个波前互相干涉的情况。此时，直接匹配该点，并将三个对应的边从标记为已经处理完毕。

d. 匹配点选择后的操作

匹配点选择完毕后，按照 DCEL 结构建立新的三角形和相应的其它数据结构，然后更新波前。更新波前的操作包括：将当前边标记为非波前，对于扩展的两条边，如果其孪生边不属于波前，则将其孪生边标记为波前。

另外，匹配的结果有可能使得某一条新边和波前中的另外两条边构成了三角形，如下

图所示：p0p1、p1p3、p3p2 均为波前，p0p1 为当前波前边，p2 是最佳匹配点。此时新扩展的边 p1p2 与波前中的另两条边构成了三角形。在波前的扩展时应对这种情况进行处理，即将 p1p2p3 加入三角形链表中，并将 p2p3、p1p3 标记为非波前。



3.3 算法总结

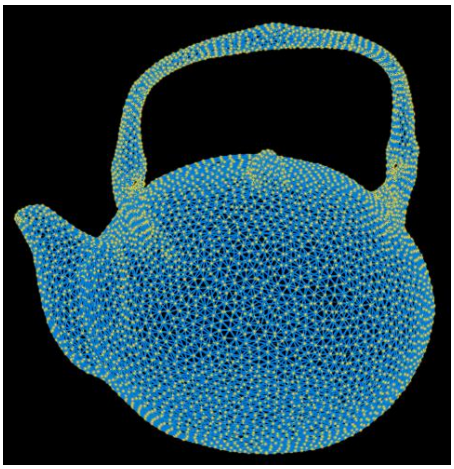
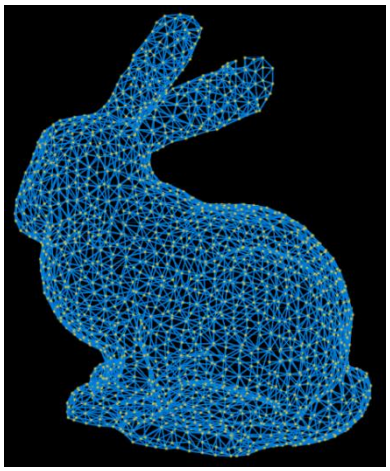
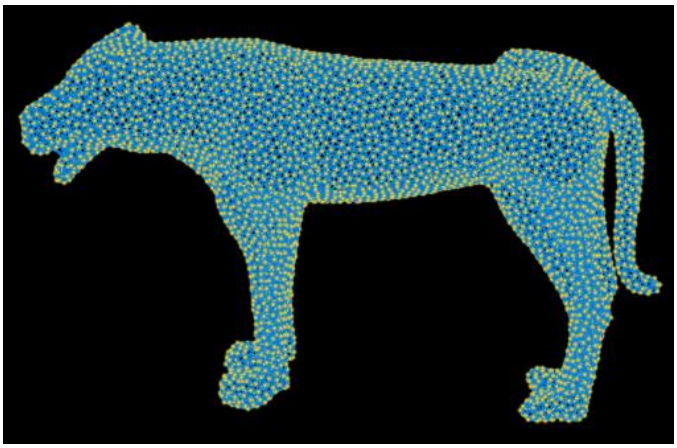
从上面的叙述可以看出，当前波前边无论扩展成功与否，都将其从波前中删除；对于新生成的边，如果满足一定的条件，则将其加入波前。在匹配点选择时综合考虑了多方面的情况，从而保证了在大多数情况下都可以使波前得到正确扩展。另外，该算法通过在 k 邻域内搜索的方法，实现了近似 O(n)复杂度的剖分。

波前算法与其他二维平面内的三角剖分方法相比，波前算法是一种区域增长算法，它在每一次增长的过程中新增的三角形都存在与最后的三角剖分之中。因此波前算法更加形象，更适合演示。而由于其局部搜索的特性，更容易扩展到 3 维中去，这也是我们采用波前算法作为解决三位表面重构的原因之一。

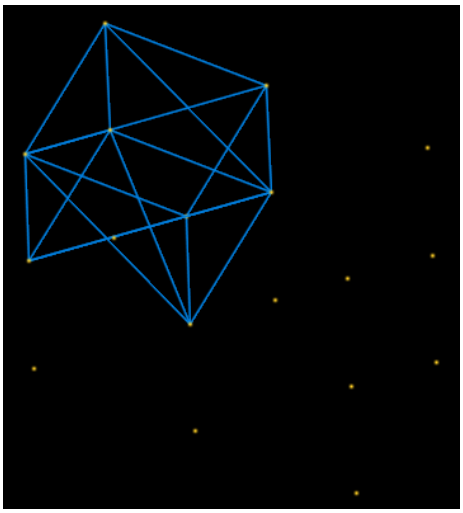
但是，我们的算法也存在很多不足之处，比如：对某些判断条件设置单一的、硬性的阈值，不够灵活（如两平面曲率变化容许范围的判断）、波前算法对某些情况无法处理、k 邻域选择不当导致某些点未被剖分等。

4. 示例

以下示例是在 10 近邻范围内搜索匹配点、并保证两平面曲率变化小于 90 度的条件下剖分得到的，实验环境为 jdk1.4.

对于波前算法来讲，如果波前提前闭合，将有一些点未被剖分。也就是说，波前算法本身并不能处理所有的情况，失败的一个例子如下：



参考文献

[1] Cavendish J.C., Field D.A., Frey W.H. An Approach to Automatic Three-Dimensional Finite Element Mesh Generation [J]. International Journal for Numerical Methods in Engineering, Vol. 21, 1985.

[2] （荷）德贝尔赫（Berg, M.）等著，邓俊辉译。计算几何——算法与应用（第 2 版），清华大学出版社。

[3] 卢海霞，杨耀权，苏杰，张文静，“基于 Java3D 实现散乱数据点三角剖分的算法”，华北电力大学学报，第 32 卷第 6 期，2005 年 11 月。

[4] 熊歆斌，宁涛，唐荣锡，“逆向工程中散乱数据点三角剖分的波前算法”，北京航空航天大学学报，第 30 卷第 4 期，2004 年 4 月。

附录：

程序结构

本次实验程序用 java 实现，程序结构如下：

```
┬  UI.java                    程序界面的实现
│
├──algorithms
│      KNN.java               K 近邻算法的实现
│      Triangulation.java     波前算法的实现
├──data                       存储测试数据的目录
├──data_structs               波前算法中所用的数据结果
└──scene                      三维显示相关的算法和数据结构
```

数据文件格式

点云数据文件以文本形式存储在 data 目录下，其格式如下：

[点的个数]

X1 Y1 Z1

X2 Y2 Z2

.

.

.

Xn Yn Zn

坐标之间以一个空格隔开

三维绘制算法

现存的一些基于 java 的 3D 绘制库，如 java3D，JOGL 等，由于采用了硬件加速，所以都需要安装特定的动态链接库才能使用。为了使我们的实验程序能够具有更好的平台无关性，我们实现了一个简单的 3D 模型绘制算法。算法大致如下：

根据当前的视角，将各个物体的坐标投影到图像平面上去

对于图像上的每一条扫描线 y=i

2.1 计算待绘制物体与扫描线的交点和相交区域

2.2 对扫描线上的每一点 x = j

2.2.1 找出该点处的所有待绘制的对象，并将它们按 z 坐标排序

2.2.2 计算这些对象在该点处的颜色和透明度，并按 z 坐标的顺序将颜色混合，得到图像上（i，j）点的颜色。

为了消除最终图像上的锯齿，在绘制点和直线的时候根据距离和一个高斯模型来决定该点的透明度，使得最后的图像较为美观。