Voronoi-Based Surface Reconstruction

李登高　刘建　岳利强

Abstract

基于 Nina Amenta, Marshall Bern[1]等人的论文，我们实现了一个从三维散乱点集重构出三维曲面的算法。该算法保证在采样密度满足一定要求的条件下，重构出来的三维曲面与原曲面是拓扑同构的，并且随着采样密度的增大，重构结果收敛到原曲面。输出的结果网格插值输入点集，而不是逼近输入点集。

该算法的基础是三维空间的 Voronoi 图，有了三维 Voronoi 图的有效算法，整个算法就很容易实现。我们采用的是 Clarkson[2]等人提出的随机增量算法，该算法通过求解 n+1 维的凸包来求解 n 维的 Delaunay 三角划分。对三维空间的 Delaunay 三角划分，该算法的时间复杂度和空间复杂度均为 $O(n^2)$。

通过对原算法一定程度上的优化与改进，我们的实现版本显示出了良好的效率。

Introduction

关于三维曲面的重构问题，有很多相关的文献。Hoppe[4]等人最早提出了这个问题，他们定义输入为三维空间 R^3 上的无组织点集，输出为多边形网格。而将 Voronoi 图和 Delaunay 三角划分引入到曲面重构中的思想由来已久，但是和其他的重构算法相比，Amenta 等人的算法具有下列优点：

1. 不需要实验确定参数，具有“自适应”的特性；
2. 重构结果插值输入点集，高频特性好，不会损失细节信息；
3. 在满足采样条件的基础上，正确性可以得到保证。

该算法生成输入采样点的 *crust*：即生成一系列的三角形，顶点均为采样点。实际上，crust 三角形总是会出现在采样点集的 Delaunay 三角划分中的。

所谓 crust，是指将采样点集的（部分）Voronoi 顶点加入到采样点集中，再做一次 Delaunay 三角划分，如果一个 Delaunay 单纯形中有三个点均属于采样点，则将这三个点构成的三角形加入到 crust 集合中。

实验结果表明，三维 crust 很好地再现了原曲面。

Delaunay Triangulation in 3D Space

在二维空间中，Delaunay 三角划分是指由采样点构成的三角形集合，其外接圆内部不包含其他采样点。推广到三维空间中，Delaunay 三角划分是指由采样点构成的四面体集合，其外接球内部不包含其他采样点。

我们知道，Delaunay 三角划分与 Voronoi 图是对偶的，一个 Voronoi 顶点对应一个 Delaunay 单纯形，求 Voronoi 图与求 Delaunay 三角划分是等价的。而 d 维的 Delaunay 三角划分可以转化为 d+1 维的凸包问题，其关键思想是将 d 维点集投影到 d+1 维广义抛物面上，求其凸包，然后再投影回去。

理解了上述基本问题，三维空间的 Delaunay 三角划分算法就归结到四维空间的凸包算法上了。幸运的是，现在关于高维凸包算法的研究成果非常丰富，一些算法模块可以从网上下载到。虽然如此，我们还是参考 Clarkson 的算法，写出了自己的实现版本。

Clarkson's Convex Hull Algorithm

Clarkson[2]的凸包算法是一种增量算法，适应任意维的凸包求解。为了阐述 Clarkson 算法，我们先引入高维空间中的下列概念（均不考虑退化的情况）：

超平面(Hyper plane)：超平面将 d 维空间分成两部分，超平面方程为一线性方程；二维空间中的直线，三维空间中的平面是超平面在二维、三维空间中的特例。
单纯形(Simplex)：d 维空间中 d+1 个点确定一个单纯形；二维空间中的三角形，三维空间中的四面体是单纯形在二维、三维空间中的特例。
面片(Facet)：d 维空间中 d 个点确定一个面片，即任意单纯形去掉一个顶点所剩下的部分；二维空间中的线段，三维空间中的三角形是面片在二维、三维空间中的特例。
脊(Ridge)：d 维空间中 d-1 个点确定一个脊；二维空间中的点，三维空间中的线段是脊在二维、三维空间中的特例。
邻接关系：当两个单纯形共用一个面片时，我们称这两个单纯形是邻接的。当两个面片共用一个脊时，我们称这两个面片是邻接的。

Clarkson 算法的基本思想是将点集的凸包用一系列单纯形的并集来表示。当加入一个点 x 时：如果 x 位于凸包内，忽略之；如果 x 位于凸包外，对于凸包上每个对 x 可见的面片 F，生成一个由点 x 和面片 F 构成的新单纯形，称 x 为该单纯形的**顶点(Peak Vertex)**，F 为该单纯形的**底面(Base Facet)**。通过巧妙设计的数据结构，对凸包的查询、更新操作能够以合理的时间复杂度来完成。

为了在数据结构中统一表示单纯形和凸包上的面片，引入无穷远点 O'的概念，将其看成是一个固有的点。将凸包上的面片也记成单纯形，其底面为该面片，顶点为 O'。以后不加说明，面片单纯形就是指这个面片对应的广义单纯形。

先建立起第一个单纯形，称之为**根**；和根共享底面的单纯形称为**反根**。在 d 维空间中，根有 d+1 个面片，从而有 d+1 个邻居（其中包含反根），这些邻居都是凸包上的面片和无穷远点 O'生成的广义单纯形。这样，我们就得到了 d+2 个初始单纯形。

记录单纯形的数据结构除了记录各个单纯形的顶点和底面之外，还记录各单纯形之间的邻接关系。这种数据结构保证了下列操作能在常数时间内完成： (1) 给定单纯形，找出其顶点与底面；(2) 给定单纯形的某点，找出与之相对的邻接单纯形和相对的点；(3) 给定单纯形的某邻接单纯形，找出与之相对的点。除此之外，我们还必须记录每个单纯形底面的平面方程，用于可见性判断，约定顶点总是位于底面的正半空间。

下面讨论每加入一个点 x 之后所进行的增量操作。

首先，需要确定 x 是否位于凸包外，如果是，必须找出凸包上所有对 x 可见的面片。通过从根和反根（单纯形）开始的深度优先搜索，可以找到凸包上某一个对 x 可见的面片。为了减少深度优先扩展的规模，对底面背向 x（即顶点与 x 位于底面的不同半空间）的单纯形不作扩展。同时，如果找到一个凸包上对 x 可见的面片，则停止搜索。如果搜索过程结束之后，仍然找不到凸包上对 x 可见的面片，可以断言 x 不在凸包外。

假设 x 位于凸包外，凸包上某一个对 x 可见的面片为 F。从 F 开始，做一次邻接搜索，可以枚举出凸包上所有对 x 可见的面片。在做深度优先搜索的过程中，不扩展与顶点 O’相对的邻居，因为这些邻居必然位于凸包内；除此之外的邻居必在凸包上，因为它们必然包含 O’点。

找到这些面片，我们需要创建一些新的单纯形，并且修改某些已有单纯形的信息和邻接关系。实际上，对于老凸包上对 x 可见的面片单纯形，只需将其顶点从无穷远点 O’替换成点 x 即可。在实际实现中，对于它们之间的邻接关系，不需要作出任何修改，因为它们之间原来的邻接关系正好反映了它们之间现在的邻接关系。

那些需要创建的单纯形对应于新的凸包面片，这些新的凸包面片总是由 x 和一个“过渡脊”构成的。所谓“过渡脊”，是指对 x 可见的凸包面片和对 x 不可见的凸包面片共有的脊。一个“过渡脊”对应于一个新创建的单纯形：由点 x、“过渡脊”和无穷远点 O’组成的单纯形，顶点为 O’。

接下来的工作就是更新这些新建单纯形的邻接关系了（包括它们之间以及它们和已有单纯形之间的邻接关系）。这个过程涉及到的已有单纯形就是定义“过渡脊”的两个面片单纯形，除此之外，该过程涉及到的邻接关系都是新建单纯形相互间的邻接关系。基于上述事实，可以将新建单纯形的创建工作与邻接关系的更新同时完成。大体思路是：从一个新建单纯形开始做邻接搜索，创建新的新建单纯形，同时创建它们之间的连接关系。

凸包的输出非常简单，只要输出凸包上所有对无穷远点 O’可见的面片即可，这个过程完全可以复用上面的代码。

上面粗略介绍了 Clarkson 算法的大致过程。关于该算法的复杂性分析，Clarkson[2]等人给出了严格的分析。我们只是引用他们的结论：

当 d=2,3 时，其时间复杂度为 O(nlogn)；当 d>3 时，其时间复杂度为 $O(n^{[d/2]})$。

Delaunay Triangulation by Convex Hull

前面已经提到 d 维的 Delaunay 三角划分可以通过 d+1 维的凸包算法来解决。设 d 维坐标为(x_1, x_2, …, x_d)，将其投影到 d+1 维广义抛物面上为(x_1, x_2, …, x_d, $x_1^2+x_2^2+\ldots+x_d^2$)。然后求这些新点集的凸包，再将朝“下”的凸包面投影回去，每个凸包面的投影对应于一个 Delaunay 三角划分中的“三角形”。这个转换的正确性是很容易证明的，只要注意到超平面与广义抛物面的交投影到 d 维空间必然是一个广义球。

尽管将 Delaunay 三角划分归结为凸包的过程是非常简洁的，我们还是据此对求凸包的算法做了一些改进：

(1) 由于投影到广义抛物面上的点必然位于凸包上，因此在增量过程中不会出现凸包内的点，对比上面确定点是否位于凸包外的算法，可以看到，有了这个假设，就可以节省很多的搜索时间；
(2) 由于只有朝“下”的面片对我们是有意义的，引入 d+1 维空间的“上”无穷远点(0, 0, …, +∞)作为第一个单纯形的顶点。这样整个凸包就变成了一个“柱状物”，大大减少了单纯形的数量，加快了算法的运行，同时又不影响算法的正确性；
(3) 注意到在(2)的改进中，所有与上无穷远点相关的凸包面去掉上无穷远点后剩下的脊正好对应 d 维凸包中的一个面片，因此，该算法除了输出 d 维点集的 Delaunay 三角划分之外，还可同时输出其凸包。我们注意到，这在下面的算法中将是非常有益的。

The Crust Algorithm

有了三维 Delaunay 三角划分的算法，我们来讨论 crust 算法在三维空间的实现以及和二维空间的异同。

为了叙述算法的内在思想，我们先引入**中轴**[1]的概念。在 d 维空间中，d-1 维曲面的中轴是指在曲面上拥有不止一个最近点的点的集合。形象的说，让一个广义球自由膨胀，保持内部为空，直到球上有两点与曲面相切，这个时候球心就在中轴上。

在二维空间中，crust 是定义在采样点集上的一个图[3]：一条边属于 crust 当且仅当这条边有一个外接圆内部既不包含采样点，又不包含 Voronoi 顶点。Amenta[3]等人证明了对于一段光滑曲线，如果采样间距小于 0.25 倍的点到中轴的距离时，crust 总是连接曲线上相邻的点。

二维空间 crust 算法非常简单，先计算出采样点集的 Voronoi 顶点，将其加入采样点集，然后计算整个点集的 Delaunay 三角划分，保留三角划分中连接两个采样点的线段，这些线段集就是所求的 crust。

实际上，在二维空间中采样密度足够时，采样点的 Voronoi 顶点集合总是很好地拟合了曲线的中轴，这也是二维 crust 算法正确性的保证。不幸的是，到了三维空间，这种完美的性质不复存在了。在二维空间中，如果三点几乎位于同一条直线上，则它们的外接圆的圆心必然远离该直线；而在三维空间中，如果四点几乎位于同一个平面上，它们的外界球的球心未必就远离该平面，甚至可能位于该平面上。由于这个原因，三维 crust 算法不能使用所有的 Voronoi 顶点，Amenta[1]等人采用了下述算法来“筛选”Voronoi

顶点，筛选出来的 Voronoi 顶点称为**极点**：

1. 计算采样点集 S 的 Voronoi 图；
2. 对每个采样点 s 应用下面的处理过程：
   a) 如果 s 不在凸包上，取 p^+为属于 s 的最远的 Voronoi 顶点。记 sp^+为 n^+。
   b) 如果 s 在凸包上，记 n^+为相邻凸包三角形外法向的平均值。
   c) 取 p^-为属于 s 的在 n^+上负向投影最大的 Voronoi 顶点。
3. 令 P 为所有极点 p^+与 p^-的集合。计算 $S \cup P$ 的 Delaunay 三角划分。
4. 保留所有顶点均属采样点 S 的三角形。

这就是三维空间 crust 算法的全过程。由于我们实现的 Clarkson 算法在输出三维 Delaunay 三角划分的过程中能够同时输出三维凸包，上述算法中所有的要点就得以解决。当然在实现中，我们还将面临其他问题。

Implementation Issues

首先介绍我们实现 Clarkson 凸包算法过程中遇到的问题，由于我们将凸包算法用来解决 Delaunay 三角划分，就带来了一个算法稳定性的问题：因为广义抛物面是非常“陡峭”的，任何数值上的误差不仅仅会影响到算法的精确性，甚至会造成整个算法的崩溃，因为误差可能会导致算法中的某些隐含条件被打破，使得算法流程出现错误（我们在算法实现过程中遇到了这个问题）。Clarkson 的解决办法是对误差进行严格控制，这在他们的另一篇论文中有所论述[5]，这使得他们的实现非常复杂，实际性能也因此受到影响。我们的解决办法是用整数运算代替浮点运算，因为整数运算是“精确”的，不会带来误差。实际实现中，我们发现 32 位整数的表数范围不能满足实际需求（在计算过程中可能溢出），因此我们使用了 64 位整数。

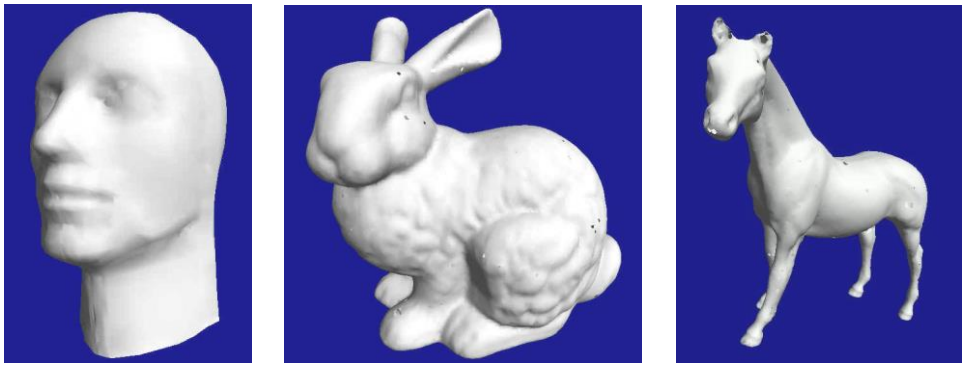
因为采样点数加上极点数不超过 3n，三维 Delaunay 三角划分的复杂度为 $O(n^2)$，所以整个算法的平均复杂度是 $O(n^2)$的。我们的实现版本在采样点数(Stanford Bunny)达到数万时，仍然能够在数分钟的时间内完成整个计算（硬件环境为 P4 1.4G，512M 内存）；而 Amenta[1]等人声称的耗时为 23 分钟（硬件环境为 SGI Onyx，512M 内存）。这部分得益于我们用整数精确运算代替了 Clarkson 繁琐的误差控制算法（Amenta 等人也注意到了这一点，并在[1]的最后部分有所提及），虽然这会损失一定的精确性，但是在实际应用中，我们并没有发现什么引人注意的误差。另外我们对上无穷远点的引入也带来了相当的性能提升。

Examples

我们利用 `gts.sourceforge.net` 网站上提供的若干例子作为我们程序的测试数据。尽管这些数据并不是专门用于三维重构的，程序运行的结果仍然令人非常满意。除了在一些特殊的地方出现了一些不应有的“洞”之外，重构结果基本反映了原曲面的信息。

仔细分析那些“洞”的形成原因，除了不满足采样密度要求之外，我们认为这正是由于三维空间与二维空间的区别造成的。仔细观察可以发现，有些“洞”的形成正是由于极点过于靠近原曲面造成的，在二维空间中，这种情况是不大可能发生的。解决这个问题

的办法之一就是舍去在求解 Voronoi 顶点的过程中系数矩阵的条件数“不好”的极点[1]，由于这需要人工确定阈值，我们没有应用到算法中去。



由散乱点集重构出来的曲面真实感图，数据来源 http://gts.sourceforge.net

Acknowledgments

我们要感谢邓俊辉老师，他将我们带入了计算几何的神奇世界，他宽松的教学方式使得我们能够“为所欲为”地钻研我们感兴趣的话题。

我们还要感谢教学实验室以及 Intel 机房的工作人员，他们为我们实现这个算法提供了近乎完美的上机条件。

还有八食堂的师傅们，每次算法出现挫折的时候，享受一顿八食堂的美食，立刻就能忘却烦恼，程序中的问题也就不攻自破了。

最后我还要感谢北大的姜晖、管雷同学（还有那台 512M 内存的 P4 电脑），他们是第一批欣赏我们程序的非清华人。

Reference

[1] Nina Amenta, Marshall Bern and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics Jul 19-24 1998 1998 ACM p 415-421*.

[2] K. Clarkson, K. Mehlhorn and R. Seidel. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications v 3 n 4 Sep 1993 p 185 0925-7721*.

[3] Nina Amenta, Marshall Bern and David Eppstein. The Crust and the β-Skeleton: Combinatorial Curve Reconstruction. *Graphical Models and Image Processing v 60 n 2 Mar 1998 Acad Press Inc p 125-135 1077-3169*.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH'92 Proceedings*, pages 71-78, July 1992.

[5] Kenneth L. Clarkson. Safe and Effective Determinant Evaluation. *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pp. 387-395, 1992.