Delaunay 三角网构建方法比较研究

余 杰 吕 品 郑昌文
(中国科学院软件研究所综合信息系统技术国家级重点实验室, 北京 100190) (中国科学院研究生院, 北京 100049)

摘 要 Delaunay 三角网构建是 3 维场景可视化领域的一个热点也是难点问题。归纳总结了现有 Delaunay 三角网构建研究中的 3 类方法——逐点插入法、三角网生长法和分治法,以及在各自原理框架下的不同实现算法;比较分析了 3 种不同方法的优缺点和各自代表性算法的时间复杂度,并详细讨论了 Delaunay 三角网构建方法在大规模场景渲染和地形可视化领域中未来 3 个研究方向:混合算法研究、算法支撑技术研究和分布式并行算法研究。

关键词 Delaunay 三角形 构网方法 构网效率 比较研究
中图法分类号: TP391 **文献标志码**: A **文章编号**: 1006-8961(2010)08-1158-10

A Comparative Research on Methods of Delaunay Triangulation

YU Jie, LÜ Pin, ZHENG Changwen
(*National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences, Beijing* 100190)
(*Graduate University, Chinese Academy of Sciences, Beijing* 100049)

Abstract Delaunay triangulation reconstruction is a hotspot but hard problem in 3D scene rendering and visualization field. In this paper, a review of Delaunay triangulation development is given, and then three current kinds of Delaunay triangulation methods are summarized: incremental method, triangle expanding method and divide-and-conquer method. Moreover, several kinds of algorithms under the frame of each triangulation method are compared in terms of advantage, disadvantage and complexity. Finally, directions of future work of Delaunay triangulation methods in large-scale scene rendering and terrain visualization field have been discussed, including research on algorithm combination, algorithm supporting technology and distributed parallel algorithm.

Keywords Delaunay triangle, triangulation method, triangulation efficiency, comparative research

0 引 言

在 3 维场景可视化研究领域中,构建大数据量、稳定的三角网一直是个热点问题。在众多三角网中,Delaunay 三角网所具有的空圆特性和最大最小角特性[1],保证了 Delaunay 三角网中不会出现过于狭长的三角形,使得三角网的构建更加合理与准确,从而具有极大的应用价值。同时,Miles 和 Lingas 已经证明了“在一般情况下,Delaunay 三角网是最优的”[2-3]。

在 20 世纪 70 年代末期到 80 年代末期,基于 Delaunay 三角剖分的三角网构建研究开始出现,很多学者和专家提出了许多具有实用价值的算法。进入 21 世纪,随着 Denlaunay 三角网应用领域的不断拓展以及应用需求的不断深入,特别是解决实时大规模场景渲染和地形可视化问题的迫切需要,对 Delaunay 三角网的构网效率、准确性和稳定性要求不断提高,这方面的研究已成为一大热点,出现了大量的研究成果。

本文归纳总结了现有 Delaunay 三角网构建研究中的 3 类方法——逐点插入法、三角网生长法和

基金项目:国家高技术研究发展计划(863)基金项目(2009AA01Z303)
收稿日期:2008-10-28;**改回日期**:2009-04-01

第一作者简介:余 杰(1986—),男。现为中国科学院软件研究所计算机应用技术专业在读硕士研究生。主要研究方向为地形可视化、空间数据分析。E-mail: yujie07@mails.gucas.ac.cn

分治法以及隶属于 3 类方法的各种不同实现算法。在此基础上,比较分析了 3 种不同方法的优缺点和各自代表性算法的时间复杂度,并结合 Delaunay 三角网构建最新的研究成果详细讨论了实时大规模场景渲染和地形可视化领域未来的研究动向。

1　Delaunay 三角网构建方法

根据 Delaunay 三角网构建过程的不同,把生成 Delaunay 三角网的算法分为逐点插入法、三角网生长法和分治法[4]。

1.1　逐点插入方法

逐点插入法于 1977 年由 Lawson 提出[5],随后 Bowyer,Watson 等众多学者对其进行了改进[6-11]。逐点插入方法的基本步骤如下:

1) 构建初始大三角形;

2) 随机排列点集 $\boldsymbol{P}$ 中的所有点 $P_0, P_1, \cdots, P_{n-1}$;

3) 对点集 $\boldsymbol{P}$ 中的点 P_r 按以下步骤执行插入操作:

(1) 定位包含点 P_r 的三角形 $P_iP_jP_k$;

(2) 如果 P_r 在三角形 $P_iP_jP_k$ 内部,则连接点 P_r 和三角形 $P_iP_jP_k$ 的各个顶点,将其剖分成三个子三角形;如果 P_r 在三角形 $P_iP_jP_k$ 的边 P_iP_j 上,则连接 P_r 与共边 P_iP_j 的两个三角形的第三个点,剖分形成四个子三角形;

(3) 通过边交换方法规格化三角剖分;

4) 移除包含大三角形任意顶点的所有三角形。

逐点插入法的各种实现算法的差别在于其构建初始多边形方法、搜索定位三角形和重构方法的不同上。

1) 初始多边形的构建

初始多边形可以是三角形,也可以是其他形状的多边形,采用最多的还是凸壳。凸壳是指包含 2 维平面点集内所有点的最小凸集[12]。如图 1 所示,由点 1,2,3,4,5,6 所构成的多边形就是该点集的凸壳。

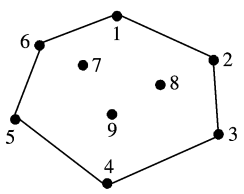

图 1　平面点集凸包

Fig. 1　The convex hull of a planar set of points

由平面点集生成凸壳的算法有多种,主要有:卷包裹法[13]、格雷厄姆法[14]、分治法[15]、增量法[16]。这些算法中,格雷厄姆法是一种常用的求点集凸壳的方法。其构造过程如图 2 所示。

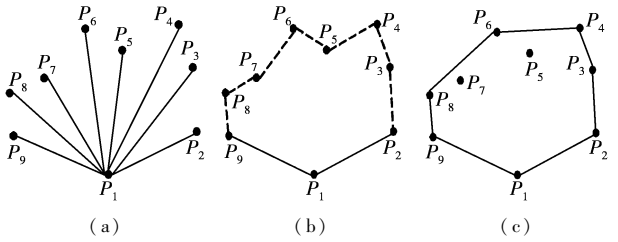

图 2　格雷厄姆方法

Fig. 2　Graham's method for determining the convex hull of finite planar set

依各点倾角及到 P_1 的距离排序(图 2(a));

按序连线成一多边形(图 2(b));

根据"凸多边形的各顶点必须在该多边形的任意一条边的同一侧"这一定理,删去非凸壳顶点,得到凸壳点集(图 2(c))。

Preparata 在文献[17]中已经证明,凸包问题的复杂度与排序问题的复杂度等价,都为 $O(n\log n)$。但是提高凸包计算速度的研究并没有中止。如 Zhang 提出的利用栅格为点建立所属范围及映射,当某个范围处在凸包内部时,利用索引剔除整个范围内所有点的方法[18]及 Golin 在文献[19]中提到的利用矩形和斜线方程合并的方法取得凸包内的一个极大四边形,从而可以大量剔除凸包内点的方法。

此外,也有很多学者希望利用并行计算的方法来提高凸包计算速度。Amato 在文献[20]中提到的并行算法已经将凸包计算的复杂度降低至 $O(\log n)$,并由 2 维扩展到了 3 维。数据结构方面,可以使用决策树或者 BSP[21]来记录并行算法的各个子结果。

2) 初始 Delaunay 三角网的形成

在凸包形成之后,初始 Delaunay 三角网的构建一般有 3 种方法:(1) 点插法:从点集的非凸包点中随意选取一点,与每一个凸包点相连形成初始三角网[1];(2) 环切边界法[22]:在凸壳链表中每次寻找一个由相邻两条凸壳边组成的三角形,此三角形的内部和边界上都不包含凸壳上的任何其他点;(3) 凸包点法:以凸包中某点为中心,一般取 Y 值最小的凸包点,与凸包中其他点构成初始 Delaunay 三角网[23]。

3）三角形的搜索及定位

根据逐点插入方法的步骤，初始 Delaunay 三角网建立完成之后，就要进行点的内插。当一个点内插到三角网中时，需要对该点所在的三角形进行定位。定位算法是提高逐点插入方法效率的关键所在之一。

快速定位点所在的三角形的一个重要方法就是通过对数据进行分块管理，先定位目标点所在的块，然后在该块管理的三角形中寻找目标三角形。这种方法极大地缩小了搜索目标三角形的范围，提高了定位速度[24]。

John 采用了基于面积坐标（或重心坐标）的关系来定位点的方法[25]，但是计算量比较大。蒲浩等人采用依据点与当前三角形重心与边的位置关系来定位点的判断标准[26]，从而大大减少了运算量，其搜索终止条件是插入点与重心相对于三角形的 3 条边均位于同侧。为了进一步减少运算量，刘少华等人采用的是点边关系方向定位算法[24]。该算法能快速找到点所在的三角形，但是其搜索路径不具有唯一性，因为可能存在目标点同时在一个三角形两条边的右边，当遇到这种情况时，只能选其中的一边进行定位，这样就不能保证定位路径为最优。为了解决搜索定位路径唯一性问题，刑建业等人提出一种改进算法——最速方向定位算法[27]，该算法可以保证定位方向是最优的。如图 3 所示，N 为待插入点，M 为首三角形重心，根据 MN 连线与三角形边相交的情况来选择相应的相邻三角形进行判断，直至找出 N 点所处的三角形。

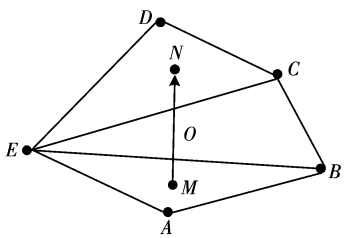

图 3 搜索点的示意图

Fig. 3 Point searching figure

但该算法没有考虑两种特殊情况：即方向线经过三角形的顶点及方向线与三角形的边重合。当遇到这两种情况时，算法有时会出现死循环情况。为了解决该问题，刘少华等人进而又提出了一种基于线—线关系的最速方向定位算法[24]，取得了比较好的效果。

4）点插入及 Delaunay 三角网的优化

当在三角网中新插入一个点 P 后，三角网重构一般有两种做法，一种是边交换方法：将 P 与其所在三角形的 3 个顶点连接起来形成新的三角形，再采用边交换进行重构优化；另一种是影响域凸包方法：确定 P 的影响域凸包，将 P 与所有影响域凸包顶点连接构成新三角形。

对于第 1 种方法，何时进行三角网的优化主要有两种方案[28]：一种方案是，每插入一个点，便对新的三角网进行优化；另一种方案是，三角网联结完毕后再进行优化。前者每次优化都不同程度地浪费了前面的优化时间，但三角形比较规整，重心分布均匀，易于三角形查找。后者在三角网未优化之前，存在大量狭长的、内角尖锐的小面积三角形，而内角尖锐的三角形存在着潜在的数字精度问题，同时狭长三角形太多，不利于三角形的查找。为了解决上述两种方法的不足，可以采用二次优化方案[28]：先插入总点数 n 中的部分点 $\sqrt{n}$，按一定的优化算法进行优化，在优化后的三角网中，再将剩下的所有点插入，最后再进行一次优化。

一般情况下相邻数据点是连续插入的，则每加入一点，该点附近三角形受影响的概率较大，需重构的三角形数较多。于是 Borouchaki 提出了先疏后密、随机插入的方法[29]，但这样难以保证插入点时，所破坏的单元数目较少。另外，Rebay 和 Anderson 则采用先加入位于边界上的点，然后再逐点加入区域内的点的方法[30-31]，取得了比较好的效果。为了进一步优化加入边界点的过程，邬吉明等人提出将加入的点按照均匀分布的原则排序，然后采用相邻排列、分级加入的思想来提高优化效率[32]。

在对复杂区域进行剖分，特别是边界尺度对比较大或是点集分布极为不规则时，Delaunay 算法判断一点在圆内还是圆外，有时会因浮点运算的舍入误差导致判断失误，以致程序非正常中断。徐明海等人建议采用双精度数据类型计算圆心及调整影响域空腔来解决此问题[33]，但似乎并不彻底。因此刘士和等人提出以排序过程代替以往算法中形成新三角形时部分复杂的搜索过程[34]，提高了计算效率的同时，也解决了由于判断误差导致“多米诺骨牌效应”以致程序非正常中断问题。Ely 和 Leclerc 则采用将不精确的点用小圆盘来代替的方法[35]。又因为不精确的点是由不精确的 x 坐标和 y 坐标导致，所以 Khanban 和 Edalat 用矩形来代替小圆盘以表示

两个方向的非精确度[36]。

1.2　三角网生长方法

生长算法[37]于 1978 由 Green,Sibson 提出,随后 Brassel,Reif 等众多学者对其进行了改进[38-40]。三角网生长算法的基本步骤如下:

1)任意选择点集 $\boldsymbol{P}$ 中的一点 P_1;

2)从点集 $\boldsymbol{P}$ 中选取距离 P_1 最近的点 P_2;

3)如果点 P_3 满足条件:三角形 $P_1P_2P_3$ 满足空圆特性且夹角 $P_1P_3P_2$ 最大,则选择点 P_3;

4)创建初始边列表和初始三角形列表;

5)如果边列表不为空,则执行以下步骤:

(1)从边列表中取出一条边;

(2)寻找满足空圆特性的点插入三角网中;

(3)与新插入的点构建新的边和三角形,并加入到边列表和三角形列表中;

(4)重复执行第 5 步,直至退出。

该类方法的各种实现算法的不同点多表现在搜寻“第 3 点”的策略上。

传统的三角网生长算法大部分时间花费在搜索符合要求的离散点中。可以引入直线分割式正负区的原理来查找第 3 点[41],每次只需查找部分离散点,从而大大提高构网的速度。如图 4 所示,可以求出由 $t_1[k]$ 和 $t_2[k]$ 构成的直线方程判别式,$F(x,y)=y-Ax-B$,将点坐标代入,若 $F(x,y)>0$,则点位于正区,若 $F(x,y)=0$,则点位于直线上,若 $F(x,y)<0$,则点位于负区。只有与 $t_3[k]$ 位于不同区的点才有可能成为扩展点,这样就不需要判断整个点集来找第 3 点,缩小了选点范围。

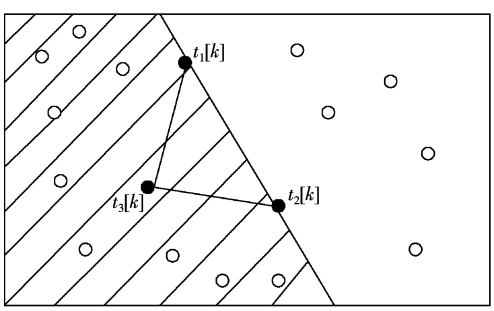

图 4　正负区判别将要搜索的点集

Fig. 4　Positive-negative-region judge point set

在三角网生长算法构网过程中,还可以利用拓扑关系——任意一个三角形的一条边最多只能和另外一个三角形公用(边缘三角形除外),来简化构造过程。

贺全兵等人在文献[42]中提到了封闭点的概念:点 P 是离散点集 $\boldsymbol{S}$ 中的一点,与 P 连接点的集合 $\boldsymbol{F}(P)=\{Q_1,Q_2,\cdots,Q_n\}$,如果在某一时刻有 $F(P)$ 不为空,且 $D(Q_1,P,Q_2)$,$D(Q_2,P,Q_3)$,…,$D(Q_n,P,Q_1)$ 均是已形成的三角形,则称点 P 在此时是封闭的,或称点 P 是封闭点。如图 5 所示,左图的 P 点是封闭点,而右图的 P 点则不是,因为还有一个三角形 PQ_4Q_5 没有生成。

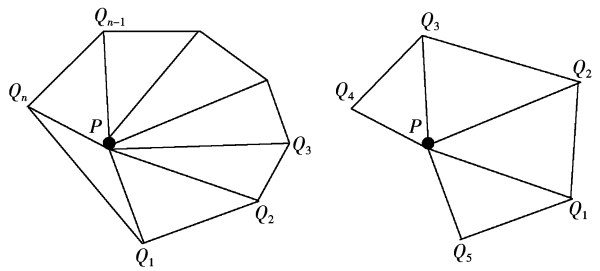

图 5　封闭点示意图

Fig. 5　Close point show figure

一旦一个点成为封闭点,那么它将不可能再与其他点构成满足剖分要求的三角形。所以在扩展三角形时,动态地排除封闭点将能加快构网速度。为了进一步加快选点的速度,可以在构网前对点集进行分区。而分区的策略谭仁春总结了两种方法:固定分区范围和固定选点数量[43]。

章孝灿等人在文献[44]中提到了利用拓扑关系和凸包技术来优化三角网生长算法。由于该算法在凸壳链表的维护和查找上降低了三角剖分算法的效率,陈学工等人在这一方面对其进行了改进[45]。另外,这类算法的关键是寻找凸包与凸包外一点的支撑线,蒋红斐等人提出了一种包围盒技术[46],该技术将凸包所在平面划分成 13 个区域,这样通过计算插入点所在的区域,就可判断出支撑点位于哪一段凸包边界上,从而提高支撑线的查找速度。

1.3　分治方法

分治算法思想[47]于 1975 年由 Shamos 和 Hoey 提出。并由 Lewis 和 Robinson 将此思想应用在 Delaunay 三角网构建中[48]。此后,Lee 和 Schachter 对其进行了改进[49]。

分治算法的基本步骤如下:

1)根据点坐标对点集进行排序;

2)将排序后的点集二分成两个点集 $\boldsymbol{V}_L$,$\boldsymbol{V}_R$;

3)如果点集 $\boldsymbol{V}_L$ 中点的数量大于 3,重复第 2 步;

4)如果点集 $\boldsymbol{V}_R$ 中点的数量大于 3,重复

第 2步;

5）合并点集 $\boldsymbol{V}_L$,$\boldsymbol{V}_R$。

该类方法的各种实现算法具有不同的点集划分策略、子三角网生成策略及合并策略。

1）点集的划分

根据点集分割方法的不同,可以分为单向划分、双向划分、四叉树划分等。Lee 和 Schachter 按照点集的 X 值进行单向排序、分割[49]。这样容易形成狭长的三角形,在合并进行 LOP 处理时,交换边的概率较大,严重影响整个构建三角网的效率。Dwyer 在此基础上进行了改进[50],他首先把点集分割成 $n/\log(n)$ 个子集,对每个子集构建 Delaunay 子三角网,然后对每行的子集进行横向、纵向合并,进而构建整个三角网。Katajainen 和 Koppinen 对 Dwyer 的算法又进行了改进,按照四叉树方式进行点集分割与子三角网合并[51]。但是由于四叉树的叶子节点内的点分布不均匀,可能出现 0 或 1 个点的情况。为了防止出现这种情况,胡金星等人采用了如下方法[52]:首先按照 X 方向、再 Y 方向进行固定大小的块分割;然后在固定块分割的基础上,再按照先 X 方向、后 Y 方向的顺序进行点集的排序、精细分割,如此反复进行,直至所有子集中的点数为少于 4 个点为止。而邹徐文等人则采用平衡二叉树代替四叉树的方法来解决点分配不均匀问题[53]。

上述分治算法都是将点集分割成易于生成三角网的小数据块集合,然后把子集中的三角网合并,经优化生成最终的三角网。如何确定块的大小是一个关键问题,它直接影响其后子集的合并效率。徐青等人认为可以将块大小设置成一个动态变化的值[54],即根据实际的需要,灵活确定一个阈值(每块的最小点数)来控制每块内点数的下限,从而使算法效率达到最优。

2）凸包合并

如何保证两个部分三角网合并后产生的三角网符合 Delaunay 准则,也是分治算法的关键。在相邻 Delaunay 子三角网的合并过程中,一般包括搜索合并起始线和终止线,删除不满足 Delaunay 准则的三角形,生成符合 Delaunay 准则的三角形,以及合并后新边界点的重构这 4 个重要部分[52],合并过程如图 6 所示。

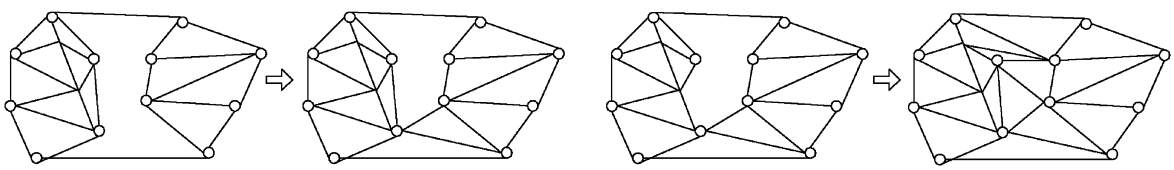

图 6 合并凸包示意图

Fig. 6 Convex hull combination figure

由于在点集分块时,可能某些块内的点数少于 3 个,这样,子集之间的合并分为边与 Delaunay 子三角网的合并、点与 Delaunay 子三角网的合并和 Delaunay 子三角网两两合并 3 种情况[52]。当然也可以将点少的块与邻近的块进行合并来省去这 3 种情况的判断。为了减少合并时内存的消耗,胡金星将三角形分成 4 种类型,根据对三角形类型的判断操作,把第三、四类三角形存入到空间数据库,而只需把一部分三角形驻入内存,从而提高了内存的使用效率[52]。

还有另外一类不同于上述实现的合并算法[55],这类算法是基于各子三角网的凸包边界来进行合并,这是因为两子网合并可进行扩展的边一定是各子网的凸包边,这样就不须搜寻整个子块内的点,从而也大大减少了点寻找的过程。当采用的是四叉树数据结构时,只需遍历此四叉树,将具有相同父亲节点的 4 个叶子节点的三角网合并,形成新的叶子节点,取代父亲节点,反复遍历直到根节点,从而形成一张完整的 Delaunay 三角网[56]。考虑到大量使用递归调用的程序运行效率是较低的,可在程序实现中将递归调用转化为迭代,来提高算法效率[57]。

2 3 类方法的比较

三角网生长方法由于搜寻第 3 点所消耗的时间过长,这种方法近年来已经很少用到,比较常用的是分治方法和逐点插入方法,而这两类方法又有其各自的优缺点。逐点插入方法实现过程相对简单,所

需内存较小,但它的时间复杂度相对较高。所以从时间复杂度方面看,分治算法是最好的。但由于分治算法中存在大量递归,所以实现起来需要较大内存空间,制约了其在大规模数据集上的应用。

逐点插入算法最坏情况下的时间复杂度是 $O(N^2)$。但是,若点是随机插入的话,其性能将能达到 $O(N\log N)$。但由于算法过程需要排序,其时间复杂度不能进一步改进。分治算法的效率很高,最坏情况下的时间复杂度都能达到 $O(N\log N)$,最好情况下其性能甚至能达到 $O(N\log\log N)$。生长算法的效率最差,性能最高也只能达到 $O(N^{3/2})$。

3 类方法各自典型的实现算法的时间复杂度如表 1 所示。

表 1　几种 Denaulay 三角网生成算法的时间复杂度

Tab.1　Run-time complexities for several Delaunay triangulation algorithm

算法	提出者	最好情况	最坏情况
逐点插入法	Lawson(1977)	$O(N^{4/3})$	$O(N^2)$
	Bowyer(1981)	$O(N^{3/2})$	$O(N^2)$
	Watson(1981)	$O(N^{3/2})$	$O(N^2)$
	Sloan(1987)	$O(N^{5/4})$	$O(N^2)$
	Guibas(1992)	$O(N\log N)$	$O(N^2)$
生长算法	Green 和 Sibson(1978)	$O(N^{3/2})$	$O(N^2)$
	Brassel 和 Reif(1979)	$O(N^{3/2})$	$O(N^2)$
	Macullagh 和 Ross(1980)	$O(N^{3/2})$	$O(N^2)$
	Mirante 和 Weigarten(1982)	$O(N^{3/2})$	$O(N^2)$
分治算法	Lewis 和 Robinson(1978)	$O(N\log N)$	$O(N^2)$
	Lee 和 Schachlter(1980)	$O(N\log N)$	$O(N\log N)$
	Guibas(1985)	$O(N\log N)$	$O(N\log N)$
	Dwyer(1987)	$O(N\log\log N)$	$O(N\log N)$
	Chew(1989)	$O(N\log N)$	$O(N\log N)$

近些年,众多学者对 Delaunay 三角网构建算法研究并没有很大程度地提高算法的时间复杂度。只是采用不同的数据结构或并行处理的思想来提高三角网的构建速度。

3　研究动向

国内外在 Delaunay 三角网构建方法上的已有研究主要集中在逐点插入算法搜索策略的改进和分治算法与逐点插入算法的合成以及分布式处理等方面。随着的研究的不断深入和需求的不断提升,特别是在大规模场景渲染和地形可视化等领域中,Delaunay 三角网构建在方法设计、理论研究和实际应用等方面仍有很多工作要做,需要解决的核心问题是如何同时达到时空性能最优、稳定性能最佳两个目标。结合 Delaunay 三角网构建方法最新的研究成果,对该领域的研究动向进行总结与展望。

1）混合 Delaunay 三角网构建方法

目前研究较多的合成算法是把逐点插入法植入到分治方法中[56]。但是由于三角网生长方法在点集较小时也具有比较好的性能,所以当前在子三角网构建时采用三角网生长方法的研究也开始出现[55]。与插入法和分治法结合相比,扫描线算法[58]除了具有良好的执行效率和稳定性外,也可以很好地解决插入和分治合成算法子模块的凸壳计算问题,所以郭兆胜等人提出了一种将扫描线算法与分治算法相结合的方法[59]。因此,如何将已有的方法进行合理的组合来提高构网效率,既是大规模数据场景网格生成领域中的研究热点,也是该领域下一步继续值得研究的方向。

除了传统的构建方法外,文献[60-62]使用了基于贪婪算法的表面重建方法,它们都是从一个已选的种子三角形开始逐渐增长来重构表面。Bernardini 等人提出的算法[60]由于不用对整个采样点集进行 Delaunay 三角剖分,所以速度很快。但是使用该算法重建表面的质量与用户自定义的与采样点密度相关的参数有关。当采样不均匀时,这些参数值的选取是非常复杂的。Boyer 和 Petitjean 提出的算法[61]可以支持不均匀采样,但是当遇到形成圆形的点集时,算法会失败。并且,该算法不保证重建表面拓扑结构的正确性。为了解决上述两个算法的不足,David 采用区域增长方式来构建三角网[62],所用的算法具有很强的健壮性且能够支持不均匀采样。但其拓扑性的正确性还没有得到很好的证明,有待进一步的研究。因此将智能方法引入大数据集 Delaunay 三角网构建过程进而来提高构网效率也是值得深入探讨的。

2）Delaunay 三角网构建方法的支撑技术

当前 Delaunay 三角网构建方法支撑技术的研究主要包括以下 3 个方面:数据预处理、算法稳定性以及构网算法优化过程的改进。

由于在大规模场景渲染和地形可视化中,地形数据量非常大,如何进行适当准确的重采样就显得

尤为重要。早在 2000 年时,Gopi 就提出了基于表面方向曲率的采样规则[63]:一个点沿着正切平面方向的采样密集度与那个点的曲率大小成反比。该规则的几何基础是“倘若方向曲率和表面弧长的乘积为常量,那么高斯球面上采样点集法向量的分布就是均衡的”。为了获得更多的地形细节信息,人们总是希望采样点尽可能地密集,这就导致了算法的输入点集包含了许多不必要点。于是,Gao 提出将输入点集根据局部特征大小进行非均匀重采样[64],也就是在具有更多细节的区域采样点密集,而在细节特征少的区域采样点稀少。这样就可以在不丢失所需要细节的前提下,尽可能大地提高重建速度。因此针对大规模场景渲染的应用,采用一种比较好的方法实现对数据尽可能少采样的同时又能真实地还原实际情况是值得进一步深入研究的方向。

在算法实现方面,由于浮点数精度的限制,使得判断一个点在圆内、圆外还是圆上是不精确的,甚至出现多点共圆导致程序死循环的问题[34],在大规模场景渲染,数据量较大的情况下这种问题显得尤为突出。因此如何在 Delaunay 三角网构建方法的具体实现过程中解决这些问题,从而使构建过程变得更加稳定也是需要继续研究的方向。

在三角网的构建过程中,优化是决定算法效率的关键步骤。尤其当构网数据量很大时,每次优化过程都会花费很长时间。在文献[65]中,Hale 认为可以通过设置极限角,然后在构网过程中保证三角形的任何角度都大于此角,进而就可以构建满足 Delaunay 性质的三角网。这一思路可以概括为在三角网的构建过程中同时考虑了优化过程,所以效率比较高,因此遵循这一思路进行进一步的研究也具有重大实际意义。

3) Delaunay 三角网分布并行构建方法

传统的 Delaunay 三角网构建为了获得更高的精确度,导致采样点往往很多,因而计算量大。所以面对当前大规模场景和地形可视化领域中越来越大的数据量的挑战,部分研究开始考虑在分治方法的基础上对构网过程进行并行处理,将大量的计算分给多个处理器来处理,从而达到提高算法运算速度的目的[66]。在文献[66]中,Blandford 使用缓冲技术来避免点信息的额外解码工作,建立共享队列来维持没有插入的点集。为了提高空间使用率,Blandford 将数据结构以压缩方式放在主存中,直到需要查询或更新时才进行解压。此外,Blandford 对每个点集合使用 OpenMP[67] 测试锁(不是等待锁)来解决并行计算中同一网格区域的数据冲突问题。由于传统的并行算法所采用的单粒度的处理办法并不能够充分利用多线程体系结构的优越性,且不具备可靠的稳定性,Christos 等人于 2005 年提出采用多粒度的并行算法[68],认为存在 3 种级别的粒度并行:域级别粗糙粒度并行,空腔级别中级粒度并行,元素级别精细粒度并行,并对这些粒度分级性能进行了分析。

然而,由于现有的分布并行构建算法存在多个起点,4 点共圆的情况可能会引起三角网构建混乱,且并行处理也存在边界任意性问题,另外还有诸如如何进行点数据的压缩存储、运用缓存技术,如何容错和负载平衡,如何管理共享工作队列以及如何提高分布式并行算法的效率和保证其稳定性方面还有很多问题亟待研究与解决。

4 结 语

Delaunay 三角网一直是大规模场景渲染和地形可视化领域的研究热点。尤其经过二十多年来的发展,其构网方法在速度上得到了很大的提高,且对存储空间的消耗也大大减少。但随着 Delaunay 三角网应用领域的不断拓展以及应用需求的不断深入,特别是解决实时大规模场景渲染和 3 维可视化数据量大、数据复杂等问题的迫切需要,对 Delaunay 三角网的构网效率、准确性和稳定性要求不断提高,因此需要在这方面继续开展更加深入的研究。

参考文献(References)

[1] Wang Jiayao. The Theory of Spatial Information System[M]. Beijing: Science Press, 2001: 172-184.[王家耀. 空间信息系统原理[M]. 北京: 科学出版社, 2001: 172-184.]

[2] Miles R E. Probability distribution of a network of triangles (a solution to problem 67-15)[J]. Society for Industrial and Applied Mathematics, 1969, 11(3): 399-402.

[3] Lingas A. The greedy and Delaunay triangulations are not bad in the average case[J]. Information Processing Letters, 1986, 22(1): 25-31.

[4] Wu Xiaobo, Wang Shixin, Xiao Chunsheng. A new study of Delaunay triangulation creation[J]. Acta Geodaetica et Cartographica Sinica, 1999, 28(1): 28-35.[武晓波, 王世新, 肖春生. Delaunay 三角网的生成算法研究[J]. 测绘学报, 1999, 28(1): 28-35.]

[5] Lawson C L. Software for C^1 surface interpolation[C]//Rice J. Mathematical Software III. Pasadena, California: California Institute of Technology, 1977: 161-194.

[6] Bowyer A. Computing dirichlet tessellations[J]. Computer Journal, 1981, 24(2): 162-166.

[7] Watson D F. Computing the n-dimension Delaunay tessellation with application to Voronoi polytopes[J]. Computer Journal, 1981, 24(2): 167-172.

[8] Sloan S W. A fast algorithm for constructing Delaunay triangulation in the plane[J]. Advances Engineering Software, 1987, 9(1): 34-55.

[9] Macedonia G, Pareschi M T. An algorithm for the triangulations of arbitrarily distributed points: applications to volume estimate and terrain fitting[J]. Computers and Geosciences, 1991, 17(7): 859-874.

[10] Floriani L D, Puppo E. An on-line algorithm for constrained Delaunay triangulation[J]. CVGIP: Graphical Models and Image Processing, 1992, 54(4): 290-300.

[11] Tsai V J D. Delaunay triangulations in TIN creation: an overview and a linear-time algorithm[J]. International Journal of Geographical Information Science, 1993, 7(6): 501-524.

[12] Barber C B, Dobkin D P, Huhdanpaa H. The quickhull algorithm for convex hulls[J]. ACM Transactions on Mathematical Software, 1996, 22(4): 469-483.

[13] Chand D R, Kapur S S. An algorithm for convex polytopes[J]. Journal of ACM, 1970, 17(1): 78-86.

[14] Graham R L. An Efficient Algorithm for determining the convex hull of a finite planar set[J]. Information Processing Letters, 1972, 1(4): 132-133.

[15] Preparata F P, Hong S J. Convex Hulls of Planar and Spatial Sets of Points, R-682[R]. Urbana, Illinois: Coordinated Science Laboratory, University of Illinois, 1975.

[16] Guibas L, Knuth D. Sharir M. Randomized incremental construction of Delaunay and Voronoi diagrams[J]. Algorithmica, 1992, 7(1): 381-413.

[17] Preparata F, Shames M. Computational Geometry[M]. New York: Springer, 1985: 103-105.

[18] Zhang Y P, Zhang Z Z, Chen Q. A new randomized parallel dynamic convex hull algorithm based on M2M model[C]// International Conference on Convergence Technology and Information Convergence. California: IEEE, 2007: 126-136.

[19] Golin M, Sedgewick R. Analysis of a simple yet efficient convex hull algorithm[C]//Proceedings of the Fourth Annual Symposium on Computational Geometry. New York: ACM, 1988: 153-163.

[20] Amato N M, Preparata F P. An NC parallel 3D convex hull algorithm[C]//Proceedings of the Ninth Annual Symposium on Computational Geometry. New York: ACM, 1993: 289-297.

[21] Goodrich M T. Randomized fully-scalable BSP techniques for multi-searching and convex hull construction[C]//Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete algorithms. Philadelphia: Society for Industrial and Applied Mathematics, 1997: 767-776.

[22] Tang Quan, Niu Zheng. An improved algorithm of Delaunay triangulation[J]. Journal of Computer Applications, 2007, 27(B06): 158-159, 212.[汤泉, 牛铮. 构建 Delaunay 三角网的改进算法[J]. 计算机应用, 2007, 27(B06): 158-159, 212.]

[23] Liu Peng, Fang Yong, Zhong Lianjiong, et al. Research on triangulation algorithm of discrete data on plane surface[J]. Geomatic Science and Engineering, 2004, 24(4): 8-10.[刘鹏, 方勇, 钟联炯, 等. 平面离散点集的不规则三角网自动生成算法的实现研究[J]. 测绘科学与工程, 2004, 24(4): 8-10.]

[24] Liu Shaohua, Wu Dongsheng, Luo Xiaolong, et al. Research on algorithms of point fast position in Delaunay triangular net[J]. Science of Surveying and Mapping, 2007, 32(2): 67-70, 113.[刘少华, 吴东胜, 罗小龙, 等. Delaunay 三角网中点目标快速定位算法研究[J]. 测绘科学, 2007, 32(2): 69-70, 113.]

[25] John B M. Transformation of trilinear and quadriplanar coordinates to and from Cartesian coordinates[J]. American Mineralogist, 1964, 49(7/8): 926-936.

[26] Pu Hao, Song Zhanfeng, Zhan Zhenyan. On the method for fast constructing Delaunay triangulation DTM[J]. China Railway Science, 2001, 22(12): 100-105.[蒲浩, 宋占峰, 詹振炎. 快速构建三角网数字地形模型方法的研究[J]. 中国铁道科学, 2001, 22(12): 100-105.]

[27] Xing Jianye, Cheng Ming. Particularize algorithm of the D-TIN generation[J]. Water Conservancy Science and Technology and Economy, 2007, 13(5): 287-288.[刑建业, 程铭. D-TIN 生成算法[J]. 水利科技与经济, 2007, 13(5): 287-288.]

[28] Yang Qiang, Huang Dilong, Zhang Jie. Algorithm of Delaunay triangulation generation based on convex shell and twice optimization[J]. Journal of Sichuan University of Science and Engineering(Natural Science Edition), 2007, 20(1): 94-98.[杨强, 黄地龙, 张洁. 基于凸壳和二次优化的三角网生成算法[J]. 四川理工学院学报, 2007, 20(1): 94-98.]

[29] Borouchaki H, George P L. Aspects of 2-D Delaunay mesh generation[J]. International Journal for numerical methods in engineering, 1997, 40(1): 1957-1975.

[30] Rebay S. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm[J]. Journal of Computational Physics, 1993, 106(1): 125-138.

[31] Anderson W K. A grid generation and flow solution method for the Euler equation on unstructured grids[J]. Journal of Computational Physics, 1994, 110(1): 23-38.

[32] Wu Jiming, Shen Longjun, Zhang Jinglin. A fast algorithm for generating Delaunay triangulation[J]. Journal on Numerical Methods and Computer Applications, 2001, 22(4): 267-275.[邬吉明, 沈隆钧, 张景琳. Delaunay 三角网格的一种快速生成法[J]. 数值计算与计算机应用, 2001, 22(4): 267-275.]

[33] Xu Minghai, Zhang Yanbin, Tao Wenquan. A modified Delaunay generation method for 2-D domain[J]. Journal of the University of Petroleum, China, 2001, 25(2): 100-105.[徐明海, 张俨彬,

陶文铨. 一种改进的 Delaunay 三角形化剖分方法[J]. 石油大学学报, 2001, 25(2): 100-105.]

[34] Liu Shihe, Luo Qiushi, Huang Wei. Generation of 2-D nonstructural grid by modified Delaunay method[J]. Engineering Journal of Wuhan University, 2005, 38(6): 1-5.[刘士和, 罗秋实, 黄伟. 用改进的 Delaunay 三角化方法生成 2 维非结构网络[J]. 武汉大学学报(工学版), 2005, 38(6): 1-5.]

[35] Ely S J, Leclerc A P. Correct Delaunay triangulation in the presence of inexact inputs and arithmetic[J]. Reliable Computing, 2000, 6(1): 23-38.

[36] Khanban A A, Edalat A. Computing Delaunay Triangulation with Imprecise Input Data[EB/OL]. (2004-08-23)[2008-10-21]. http://www.cccg.ca/proceedings/2003/38.pdf.

[37] Green P J, Sibson R. Computing Dirichlet tessellations in the plane[J]. The Computer Journal, 1978, 21(2): 168-173.

[38] Brassel K E, Reif D. A procedure to generate Thiessen polygons [J]. Geophysical Analysis, 1979, 2(11): 289-303.

[39] McCaullagh M T, Ross C G. Delaunay triangulation of a random data set isarithmic mapping[J]. Cartographic Journal, 1980, 17(2): 93-99.

[40] Maus A. Delaunay triangulation and the convex hull of n points in expected linear time[J]. BIT, 1984, 24(2): 151-163.

[41] Deng Shuguang, Liu Gang. An improved algorithm and implementation of TIN[J]. Computer Era, 2006, 2(1): 1-2.[邓曙光, 刘刚. 一种 TIN 生成算法的改进与实现[J]. 计算机时代, 2006, 2(1): 1-2.]

[42] He Quanbin, Li Guiyou, Wen Jin, et al. An improved algorithm for building Delaunay triangulation[J]. Computer and Digital Engineering, 2006, 34(5): 50-52, 64.[贺全兵, 黎贵友, 文进, 等. 生成 Delaunay 三角网的改进算法[J]. 计算机与数字工程, 2005, 34(5): 50-52, 64.]

[43] Tan Renchun, Yao Lan, Liu Min. An improved TIN algorithm based on auto-coupling triangle[J]. Science of Surveying and Mapping, 2007, 32(1): 75-76.[谭仁春, 姚岚, 刘敏. 一种改进的自动联结三角网算法[J]. 测绘科学, 2007, 32(1): 75-76.]

[44] Zhang Xiaocan, Huang Zhicai, Dai Qicheng, et al. An algorithm of speedily building TIN based on topological structure and convex shell in GIS[J]. Chinese Journal of Computers, 2002, 25(11): 1212-1218.[章孝灿, 黄智才, 戴企成, 等. GIS 中基于拓扑结构和凸壳技术的快速 TIN 生成算法[J]. 计算机学报, 2002, 25(11): 1212-1218.]

[45] Chen Xuegong, Chen Shuqiang, Wang Liqing. An algorithm of building Delaunay triangulation based on convex hull[J]. Computer Engineering and Application, 2006, 42(6): 27-29.[陈学工, 陈树强, 王丽青. 基于凸壳技术的 Delaunay 三角网生成算法[J]. 计算机工程与应用, 2006, 42(6): 27-29.]

[46] Jiang Hongfei, Tu Peng, Li Guozhong. Study on growth algorithm for generating constrained Deluanay triangulation[J]. Journal of Highway and Transportation Research and Development, 2004, 21(12): 38-41.[蒋红斐, 涂鹏, 李国忠. 基于生长算法构建 Delaunay 三角网的研究[J]. 公路交通科技, 2004, 21(12): 38-41.]

[47] Shamos M I, Hoey D. Closest-point problems[C]//Proceeding of the 16th Annual IEEE Symposium on Foundation of Computer Science. Los Angeles. California: IEEE, 1975: 151-162.

[48] Lewis B A, Robinson J S. Triangulation of planar regions with applications[J]. The Computer Journal, 1978, 21(4): 324-332.

[49] Lee D T, Schachter B J. Two algorithms for constructing a Delaunay triangulation[J]. International Journal of Computer and Information Science, 1980, 9(3): 219-242.

[50] Dwyer R A. A faster divide-and-conquer algorithm for constructing Delaunay triangulations[J]. Algorithmica, 1987, 2(2): 137-151.

[51] Katajainen J, Koppinen M. Constructing Delaunay triangulations by merging buckets in quad tree order[J]. Fundamenta Informaticae XI, 1988, 11(3): 275-288.

[52] Hu Jinxing, Ma Zhaoting, Wu Huanping, et al. Massive data Delaunay triangulation based on grid partition method[J]. Acta Geodaetica et Cartographica Sinica, 2004, 33(2): 163-167.[胡金星, 马照亭, 吴焕萍, 等. 基于格网划分的海量数据 Delaunay 三角剖分[J]. 测绘学报, 2004, 33(2): 163-167.]

[53] Zou Xuwen, Wu Baichao, Cui Jixian. Algorithm of Deluanay triangulation generation based on AVL tree[J]. Journal of Liaoning Technical University, 2007, 26(4): 513-516.[邹徐文, 武百超, 崔继宪. 基于平衡二叉树的三角网快速生成算法[J]. 辽宁工程技术大学学报, 2007, 26(4): 513-516.]

[54] Xu Qing, Chang Ge, Yang Li. The algorithm of TIN generation based on self-adapt clump organization[J]. Journal of Image and Graphics, 2000, 5A(6): 461-465.[徐青, 常歌, 杨力. 基于自适应分块的 TIN 三角网建立算法[J]. 中国图象图形学报, 2000, 5A(6): 461-465.]

[55] Dai Xiaoming, Zhu Ping. Divide algorithm realization of plane scattered data triangulation[J]. Computer Technology and Development, 2006, 16(1): 11-12, 40.[戴晓明, 朱萍. 平面散乱点三角剖分分治算法的实现[J]. 计算机技术与发展, 2006, 16(1): 11-12, 40.]

[56] Xiang Chuanjie, Zhu Yuwen. A practical algorithm of building Delaunay triangle meshes for terrain modeling[J]. Computer Applications, 2002, 22(11): 34-36, 39.[向传杰, 朱玉文. 一种高效的 Delaunay 三角网合并生成技术[J]. 计算机应用, 2002, 22(11): 34-36, 39.]

[57] Piegl L A, Richard A M. Algorithm and data structure for triangulating multiply connected polygonal domains[J]. Computers & Graphics, 1993, 17(5): 563-574.

[58] Fortune S. A sweepline algorithm for Voronoi diagrams[J]. Algorithmica, 1987, 2(1-4): 153-174.

[59] Guo Zhaosheng, Zhang Dengrong. An improved high-efficiency algorithm of Delaunay triangulation generation[J]. Remote Sensing Information, 2005, 3(1): 15-17.[郭兆胜, 张登荣. 一种改进的高效 Delaunay 三角网的生成算法[J]. 遥感信息, 2005, 3(1): 15-17.]

[60] Bernardini F, Mittleman J, Rushmeir H, et al. The ball-pivoting algorithm for surface reconstruction[J]. IEEE Transactions on Visualization and Computer Graphics, 1999, 5(4): 349-359.

[61] Petitjean S, Boyer E Regular and non-regular point sets: properties and reconstruction[J]. Computational Geometry Theory Application, 2001, 19(2/3): 101-126.

[62] David C S, Da F. A greedy delaunay-based surface reconstruction algorithm[J]. The Visual Computer, 2004, 20(1): 4 -16.

[63] Gopi M, Krishman S, Silva C T. Surface reconstruction based on lower dimensional localized Delaunay triangulation[J]. Computer Graphics Forum, 2000, 19(3): 467-478.

[64] Gao S, Lu H Q. A Fast Algorithm for Delaunay Based Surface Reconstruction[EB/OL]. (2003-05-08)[2008-10-21]. http://www. informatik. uni-trier. de/-ley/db/conf/wscg/wscg2003. html #GaoL03.

[65] Erten H, Ungor A. Triangulations with locally optimal steiner points[C]//Proceedings of the 5th Eurographics Symposium on Geometry Processing. Aire-la-Ville, Switzerland: Eurographics Association, 2007: 143-152.

[66] Blandford D K, Blelloch E G, Kadow C. Engineering a compact parallel Delaunay algorithm in 3D[C]//Proceedings of the 22th ACM Symposium on Computational Geometry. New York: ACM, 2006: 292-300.

[67] OpenMP[EB/OL]. (2008-09-12)[2008-10-21]. http://www. openmp. org.

[68] Christos D A, Ding X N, Andrey N C. Multigrain parallel Delaunay mesh generation: challenges and opportunities for multithreaded architectures[C]//Proceedings of Nineteenth ACM International Conference on Supercomputing. New York: ACM, 2005: 367-376.