ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, Supplement, December 2008, pp.1−9

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 LiDAR 点云数据的三角网构建算法*

刘晓平[1+], 朱晓强[1], 余 烨[1], 袁晓辉[2], Bill P. BUCKLES[2]

[1](合肥工业大学 计算机与信息学院 VCC 研究室,安徽 合肥 230009)

[2](Department of Computer Science and Engineering, University of North Texas, USA)

Building Algorithm of Triangulation Based on LiDAR Point Clouds

LIU Xiao-Ping[1+], ZHU Xiao-Qiang[1], YU Ye[1], YUAN Xiao-Hui[2], Bill P. BUCKLES[2]

[1](VCC Division, School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

[2](Department of Computer Science and Engineering, University of North Texas, USA)

+ Corresponding author: E-mail: lxp@vcc.hfut.edu.cn, http://cadcg.hfut.edu.cn

Liu XP, Zhu XQ, Yu Y, Yuan XH, Buckles BP. Building algorithm of triangulation based on LiDAR point clouds. *Journal of Software*, 2008,19(Suppl.):1−9. http://www.jos.org.cn/1000-9825/19/s1.htm

Abstract: Based on the existing Delaunay triangulation method, an algorithm of triangulation growth is presented. This algorithm divides the large-scale point clouds into uniform grids and determines the searching scope self-adaptively. During the process of building a triangulated irregular network (TIN) model, the generated base-lines in groups are grouped, and the close-points are removed dynamically, which improved the speed of reconstructing TIN in large-scale scenes dramatically. By searching the triangular vertices in the scope of the whole data set, this method avoided errors caused by interpolation and the process of stitching between grids. The efficiency and effectiveness of this algorithm are verified by using real world data to build TIN model with large scale LiDAR point clouds.

Key words: LiDAR; Delaunay triangulation; TIN model; triangulation growth; self-adaptive

摘 要: 在现有 Delaunay 三角网生长法的基础上进行改进,提出了一种三角网生长算法.该算法对大规模点云进行等格网分块,自适应确定搜索范围.通过在构建过程中对生成的基线进行分组和排序,动态删除封闭点,提高了构建三角网的速度;通过在整个点集范围内进行搜索,避免了通过插值所产生的误差和模块之间的拼接过程.利用此算法对大规模 LiDAR 点云数据进行构网,结果表明了该算法的有效性.

关键词: LiDAR;Delaunay 三角网;TIN 模型;三角网生长法;自适应

机载 LiDAR(light detect and ranging)获取的数据是分布于对象表面的三维点坐标,其数据集(点云)是对象的数字表面模型[1].由于机载 LiDAR 能够自动地获取高精度、高密度的地球表面 3D 坐标信息,已成为生成数字地面模型(DTM)和数字高层模型(DEM)的首选工具[2].利用点集构建不规则三角网(triangulated irregular

* Supported by the International (Regional) Cooperation and Exchange Project in National Natural Science Foundation of China under Grant No.60573174 (国家自然科学基金国际(地区)合作交流项目); the National Natural Science Foundation of China under Grant No.60673028 (国家自然科学基金); the Open Project Foundation in Key Laboratory of Special Display Technology for the Ministry of Education of China (特种显示技术教育部重点实验室开放课题)

Received 2008-04-20; Accepted 2008-11-14

networks,简称 TIN)的方法目前可分为 3 类:分治-合并算法、逐点插入算法和生长算法.

由于分治-合并算法时间效率高,目前使用最多,但其递归过程占用大量的内存空间,不适合大规模点云的三角化,胡金星等人在分治-合并算法的基础上对点集自适应划分,从而生成海量 DEM 数据[3],但是需要不断将数据从内存中换进换出.逐点插入法占用内存较少,但是时间效率较差,而且在大规模点云构网后的合并过程中,有时需要插值产生新点或者很难保证边界拼接时的 Delaunay 三角形(后面简称 D-三角形)性质[4].生长法算法简单,但时间效率也较差.Green 和 Sibson 首次实现了一种生长 Dirichlet 多边形图的生长算法.Brassel 和 Reif 稍后也发表了类似的算法.McCullaugh 和 Ross 通过把点集分块和排序改进了点搜索方法,减少了搜索时间.Maus[5]也给出了一种非常相似的算法[6].本文对现有生长算法进行改进,自适应确定搜索范围,在 LiDAR 点云数据疏密不均的情况下也能高效地进行构网.

本文第 1 节介绍生长法的思路和基本步骤.由于 LiDAR 点云的疏密不均(如水面部分点云特别稀疏)和数量的庞大,本文对现有生长法进行改进和优化.第 2 节介绍将三角形生长法用于海量数据的关键技术.首先根据点云的密集程度自适应判断搜索范围,对生成的基线进行排序,并在构网过程中动态删除封闭点和封闭块,从而加快构网速度并避免插值和模块的拼接过程;最后通过添加边界辅助点,避免出现比较狭长的三角形.第 3 节从理论上对本文算法进行复杂度分析,并通过第 4 节的实验来验证算法的有效性.

1 三角形生长算法

1.1 生长算法基本步骤

现有三角网生长算法的基本步骤是:

1) 以任一点为起始点;

2) 找出与起始点最近的数据点相互连接形成 D-三角形的一条边作为基线,按 D-三角网的判别法则,找出与基线构成 D-三角形的第 3 点;

3) 基线的两个端点与第 3 点相连,成为新的基线;

迭代 2)、3)两步直至所有基线都被处理.

上述过程表明,三角网生长算法的思路是,先找出点集中相距最近的两点连接成为一条 D-边,然后按 D-三角网的判别法则找出包含此边 D-三角形的另一端点,依次处理所有新生成的边,直至最终完成[6].本文则对生长法进行改进,使其在从海量 LiDAR 点云数据中重建三维模型的过程中得到应用.

1.2 本文算法

在三角网生长算法基本步骤的基础上,对其进行改进,使其能够在海量点云数据中快速构建 D-三角网,构建的主要步骤如下:

Step 1. 点云分块.

根据网格点的密度将点云分块,并对生成的基线进行分组,设点集分 $M\times N$ 块,分别存放于 $Grid[i,j]$ $(1\le i\le M,1\le j\le N)$ 中,则基线可分 M 组,分别存放于 $Base[i](1\le i\le M)$ 中.

Step 2. 构建第 1 条基线.

从最左下角的分块中任意找一点 P_1, 在以 P_1 所在分块 $Grid[i, j]$ 的临域分块中找出离 P_1 最近的点 P_2 ,则 P_1P_2 作为第 1 条基线.取 P_1,P_2 中 y 值较小的一个,不妨设为 P_1, 且 $P_1\in Grid[i,j]$, 则将基线 P_1P_2 存放于基线队列 $Base[i]$ 中.

Step 3. 扩展第 1 条基线的左侧.

对于第 1 条基线 P_1P_2, 先在其左侧搜索是否有满足 D-性质的点 Q.若存在,则 P_2Q 和 QP_1 为新的基线,并插入到相应的基线队列 $Base[k]$, 其中 k 的确定方法如同插入第 1 条基线 P_1P_2.

Step 4. 扩展基线的右侧.

基线右侧的扩展过程如图 1 所示.

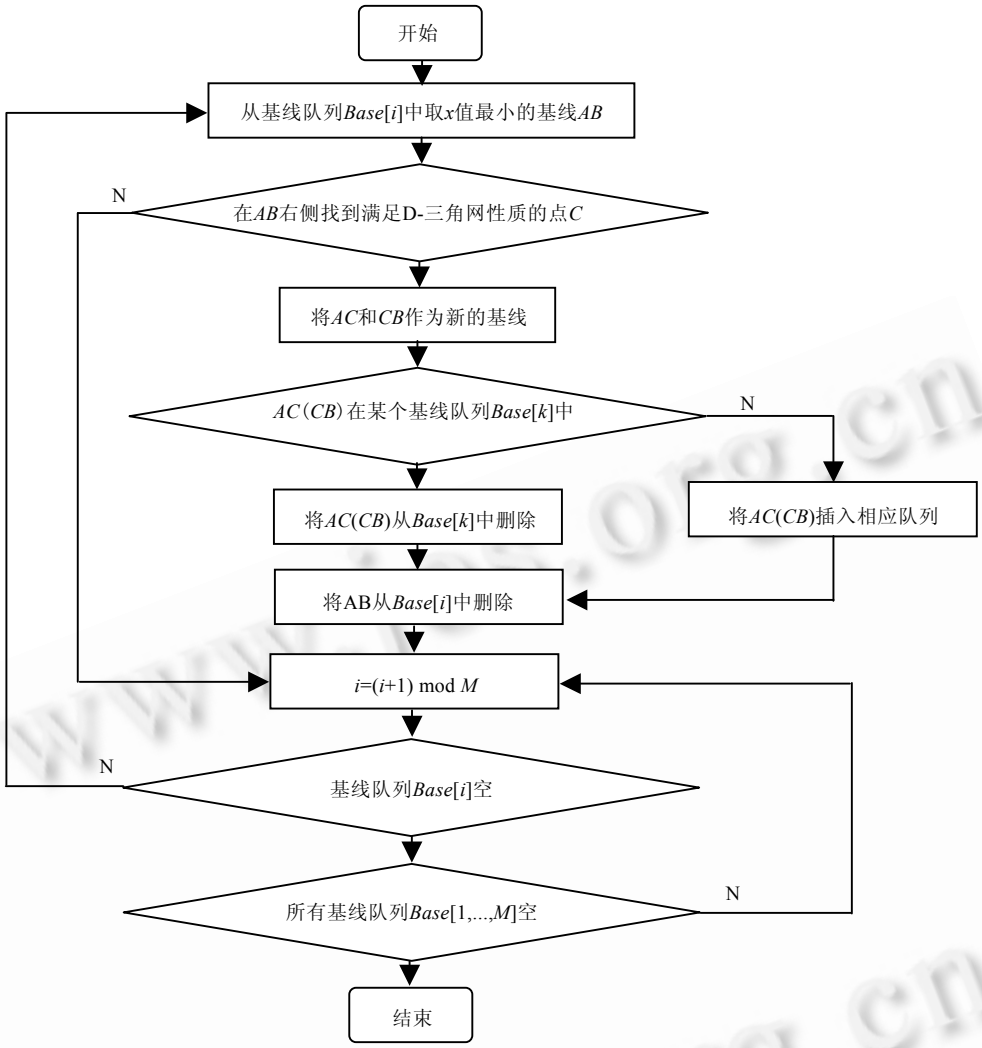

Fig.1 Flow char of extendibility of baseline's right

图 1 基线右侧扩展流程

2 关键技术

2.1 点云数据分块

三角网生长算法的核心就是求取新边构成三角网,大部分的计算都集中在从大量数据中搜寻最优的第 3 点,只要在搜索方法上加以改进就可以大大提高构网速度[7].

现有的数据分块方法有:均匀条带分割(如图 2 所示)、等格网分割(如图 3 所示)、四叉树分割(如图 4 所示)和自适应分块等[8].因为每一条基线第 3 点的查找范围是在基线的四周进行搜索,故图 2 中的均匀条带分割法依然会使搜索的范围相当大.此外,不仅要在包含基线的数据块内查找,同时也要在临近数据块内查找,故数据块的划分要易于查找或计算邻接关系,因此,本文采用图 3 中的等网格分割法.

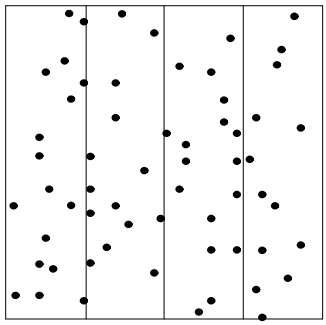

Fig.2 Uniform strip segmentation

图 2 均匀条带分割法

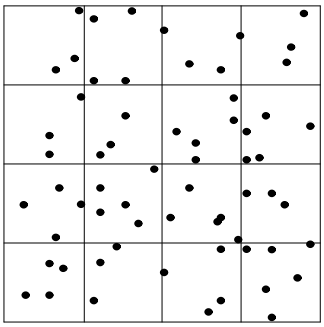

Fig.3 Uniform grid segmentation

图 3 等格网分割法

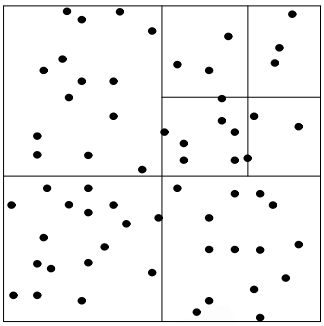

Fig.4 Quadtree segmentation

图 4 四叉树分割法

2.2 动态确定搜索范围

以往算法是在每条基线的两侧进行扩展,本文在新添加基线的时候控制新形成的三角形在当前基线的左侧(第 1 条基线除外,需单独处理),从而在每次扩展基线时只需对基线右侧进行扩展,缩小搜索范围.如图 5 所示,A,B,C,D 的坐标分别为 $(x_1,y_1),(x_2,y_2),(x_3,y_3),(x,y)$, 当前扩展的基线为 A,B,C 是其左侧三角形的另一顶点.其右侧搜索点 D 的范围应取关于直线 AB 与点 C 异侧的点,由于直线 AB 的方程为 $F(x,y)=(y_2-y_1)(x-x_1)-(x_2-x_1)(y-y_1)=0$, 当 $F(x,y)\cdot F(x_3,y_3)<0$ 时,即点 C 与 D 在直线 AB 的异侧,则该点可作为备选扩展点[9],另一种方法就是按照文献[10]中的叉积先判断 C 点与 AB 的哪一侧,然后利用叉积判断 D 在另一侧.将新生成的有向基线 AD 和 DB 加入基线队列,确保三角形 ABD 在基线 AD 和 DB 的左侧,从而无须再对 AD 和 DB 的左侧进行扩展.

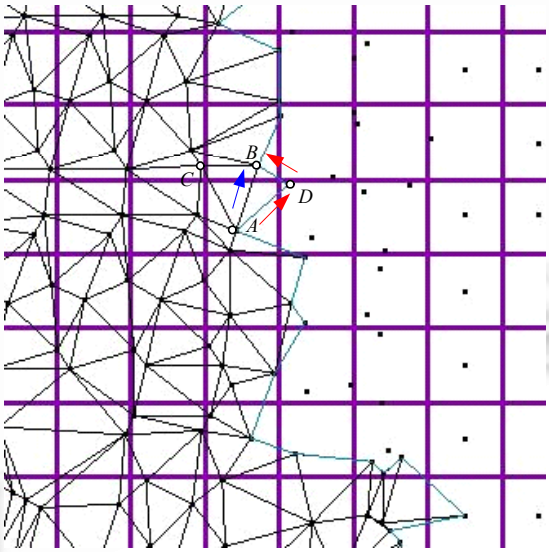

Fig.5 Newly added baseline

图 5 新添加的基线

为适应 LiDAR 点云疏密不均的情况,在构网过程中可以动态确定每条基线的扩展范围.对于待扩展的基线 AB(如图 6 所示),其扩展过程如下:

Step 1. 计算 AB 的中点 O;

Step 2. 在 AB 右侧的垂直平分线上截取 $OM=L$;

Step 3. 记过 A,B,M 三点的圆 *Circle* 在直线 AB 右侧的部分所在的分块集合为 *Blocks*;

Step 4. 在 *Blocks* 内搜索满足 D-三角网性质的点;

Step 5. 搜索到“有效”点则结束,否则令 $L=L+\Delta L_i$,此时如果 L 大于预设的阈值,则停止,否则返回 Step 2.

该方法既能有效缩小搜索范围,同时也能保证在点集疏密不均的情况下找到合适的候选点.

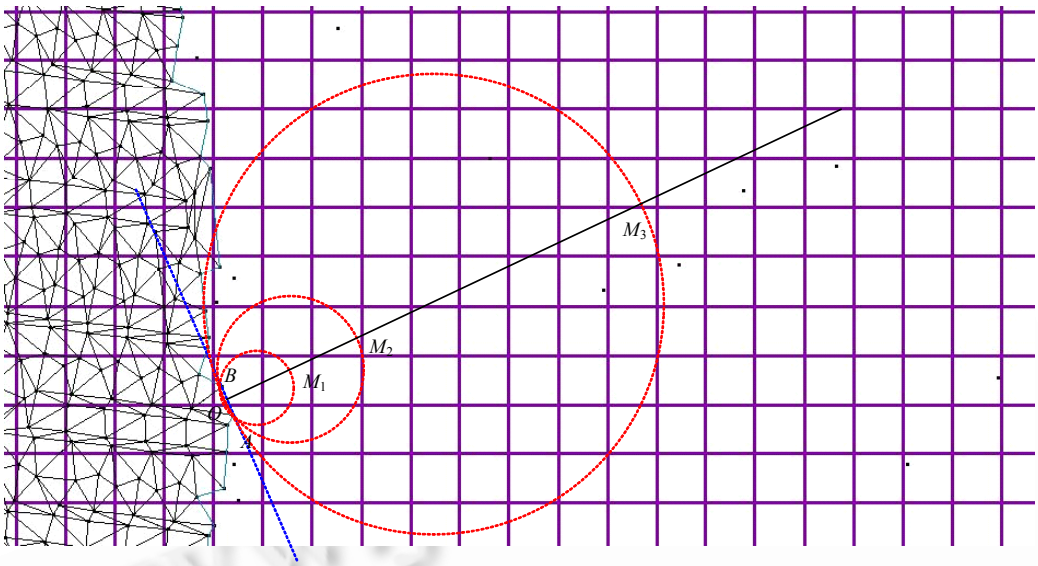

Fig.6 Scope of searching points

图 6 点的搜索范围

2.3 对基线队列排序

在将生成的新基线插入到相应队列时,都要遍历所要插入的整个队列,判断该基线是否已经存在于队列当中.因此,要对基线队列的长度进行适当的控制,可以通过将基线按照端点 x 的升序排列,同时在从基线队列 $Base[i]$ 中取基线进行扩展的时候,选取队列中的第 1 条基线,从而避免队列中的基线数目变得很大,极大地减少了内存占用和搜索时间(如图 7、图 8 所示).

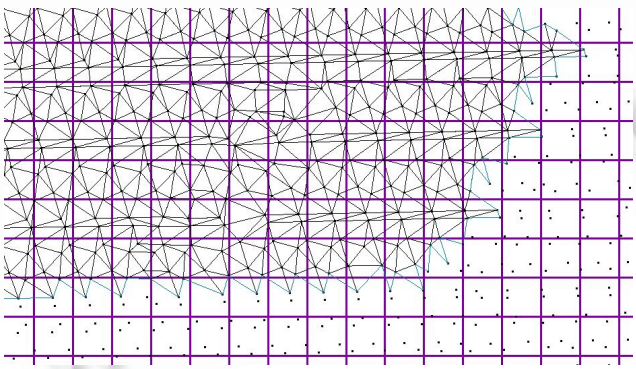

Fig.7 Before sorting

图 7 排序前

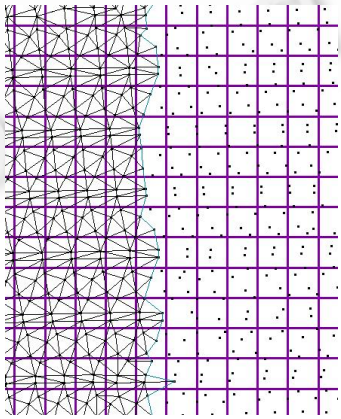

Fig.8 After sorting

图 8 排序后

2.4 动态排除封闭点和封闭块

封闭点[9]的定义:点 P 是离散点集 S 中一点,与 P 连接点的集合 $F(P)=\{Q_1,Q_2,...,Q_n\}$, 以 P 为端点的待扩展的基线条数为 $N\_Extend(P)$, 如果在某一时刻有 $F(P)\neq\varnothing$,且$N_Extend(P)=0$, 则称点 P 在此时是封闭的,或称

点 P 是封闭点.一旦一个点成为封闭点,它就将不可能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,应动态地将成为封闭点的点排出考虑的范围,从而减少搜索过程中的计算量[11].

算法中对每一个顶点 P 都用一个变量 $N\_Extend(P)$ 标记与该点相连的待扩展的基线数目, $N\_Extend(P)$ 初始化为−1,添加 P 的第 1 条基线时 $N\_Extend(P)$ 赋值 1,之后每添加一条新的基线时 $N\_Extend(P)$ 加 1,每删除一条基线时 $N\_Extend(P)$ 减 1,在扩展基线过程中若 $N\_Extend(P)$ 为 0,则表示该点已经封闭.

另一方面,每个分块都记录当前块内的总点数、未封闭的点个数,块的数据结构如下:

```
struct _BLOCK{
int nPoint;   //总点数
int nOpenPoint;   //尚未封闭的点个数
int* pPoint;   //点序号数组
…
};
```

若块内所有的点都已经封闭,即 *nPoint=nOpenPoint*,则在搜索时直接排除该分块而不是逐个点进行判断封闭性.

2.5 边界点的处理

在用插入法构造 D-三角形时一般是先构造点集的凸包,然后将剩余的点逐个插入到凸包中,这样可以很好地构建出点集的边界.在用三角网生成时一般不再构建凸包,这样,在处理边界点时,会出现一些很狭长的三角形,虽然它们也满足 D-三角形的条件,但是在实际应用当中是不需要这些三角形的.所以在本文中采用在点集边界的外围添加辅助点,先构建包含辅助点的 TIN 模型,然后从模型中删除以辅助点为顶点的三角形,通过该步骤能够很有效地避免边界的狭长三角形,而且能够很好地反映点集的边界形状.辅助点的添加方法为:若 $Grid[i,j]$ 内没有点而其 8-临域方向不全有点,则在 $Grid[i,j]$ 内添加一个辅助点,该辅助点的坐标为 $Grid[i,j]$ 的中心位置.具体添加辅助的情况和效果如图 9 所示.

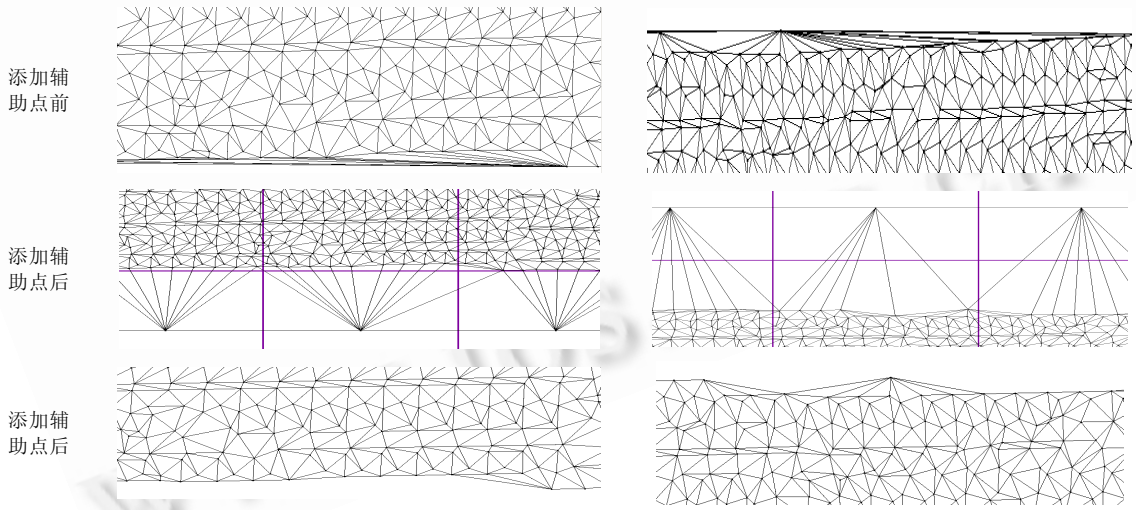

Fig.9 Process of boundary points

图 9 边界点的处理

3 算法复杂度分析

生长法构网的时间主要花费在扩展基线时候选点的搜索过程,按照生长法思想,对一定区域内的 n 个点,每查询一个“有效”点时,需要对这 n 个点逐个进行检索、计算,算法的平均复杂度为 $O(n^2)$. 当点个数比较庞大时,

每次查询都遍历所有的 n 个点势必会浪费很多时间.通过对点集进行合理的分块,可以利用查询附近区域内的部分点来代替查询全部 n 个点.

根据查询“有效”点的方式是否跨越分块,可以将基于分块的生长法构网分为两类.第 1 类是分别在每个分块内利用生长法构网,然后将每个分块构建的网格进行拼接而形成全局的网格模型[9];第 2 类是在当前基线所在分块及其邻域分块内进行查询“有效”点[7,11−12],从而避免了分块网格之间的拼接过程.对于第 1 类,网格的分块往往比较大,因为当分块较小时在点云稀疏的位置分块内可能就没有足够的点进行构网,且分块之间的拼接代价会很大,有时甚至需要添加过多的辅助点[4],从而增加冗余数据并降低网格精度,假设每个分块内点的个数约为 n_1,则算法复杂度计算为 $O(n\cdot n_1)$.对于第 2 类的情况,为保证每次都能找到“有效”点,有两种策略,一种是分块较大,每次在较少的邻域内查询;另一种是分块较小,每次在较多的邻域内查询,不妨设每个分块内有约 n_2 个点,每次在 q_2 个邻域分块内查询,则算法的复杂度可计算为 $O(n\cdot n_2\cdot q_2)$,虽然 n_2 或者 q_2 较小,但是 $n_2\cdot q_2$ 结果却比较大.

本文算法在第 2 类思想的前提下,由于采用了自适应确定查询范围的策略,可以使分块比较小,同时又能快速找到“有效”点.假设每个分块内有大约 n_3 个点,每次在 q_3 个分块内查询,则算法复杂度为 $O(n\cdot n_3\cdot q_3)$,其中 n_3 是一个相对于 n_1 小很多的数,同时 q_3 会随着点集密度的改变自适应变化,从而使 $n_3\cdot q_3$ 根据点云的密度的不同也自适应变化,避免了大量不必要的的查询.

另一方面,本文算法在扩展每条基线时仅在基线的一侧进行查询,该策略可以使查询次数平均减半,通过排除封闭点和封闭块也使查询次数极大地较少.经分析可知,本文算法不仅在点集密度均匀情况下快速构网,在像 LiDAR 点云数据这样密度变换剧烈的情况下,能够表现出更为突出的优势.

4 实验结果与分析

本文的实验数据来源于美国新奥尔良市区的 LiDAR 数据.由于 LiDAR 点云具有三维坐标,因此在构建 TIN 模型时暂不考虑其高度 Z 值,利用本文算法可以生成点云的二维网格(如图 10 所示),且自适应点集的疏密分布,在构建网格之后再将每一点作为空间点来显示.运行过程总共输入点个数 495 万,构建三角形约 867 万.图 11 为所构建模型的一部分.

从实验中我们能构看到,虽然对原始数据进行了分块,但是在后期不需要对分块进行合并,因此减少了合并分块数据所需时间;另一方面,我们在每一分块的边缘处不需要插值产生新的数据点,从而使模型中的每一个网格顶点都是机载 LiDAR 的实际采样点,能够真实地反映测量地区的实际表面形状.

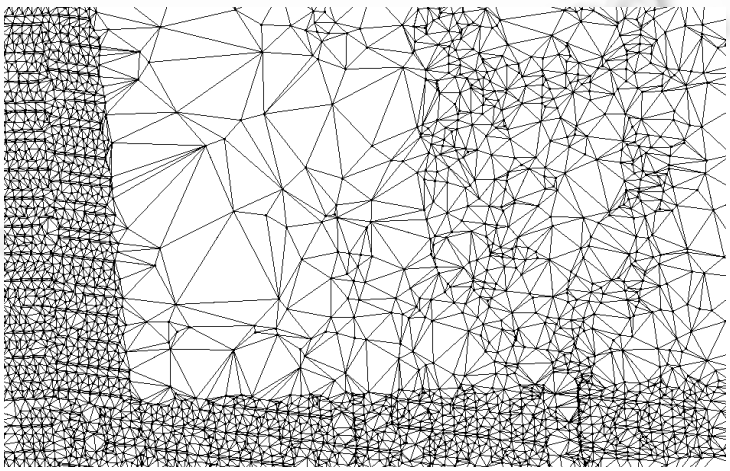

Fig.10 2D network of LiDAR-point

图 10 LiDAR 点云二维网格

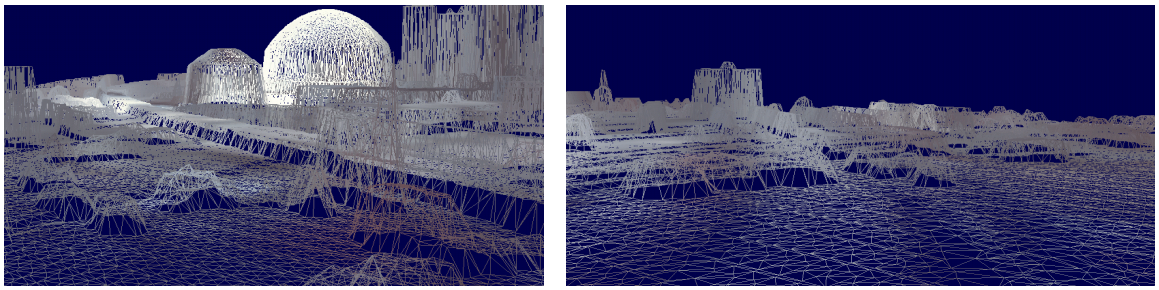

Fig.11 3D display of LiDAR-point TIN model

图11 LiDAR点云的TIN模型三维显示

为了进一步说明本文算法的优势和可行性,将本文算法与改进前等格网分块的生长算法、文献[4]中的逐点插入法进行比较,比较结果见表 1.从表 1 中可以看出,与改进前的等格网分块算法相比,本文算法的搜索策略能够大大加快构网速度,而且能够减少很多边界处的狭长三角形.在边界插值的逐点插入法由于要在分块的拼接处插入很多三角形,不仅浪费时间,而且增加了许多非必需三角形的数量.

Table 1 Comparision of algorithms

表1 算法比较结果

算法	结果		
	顶点个数	三角形个数	时间
边界插值的逐点插入法	106 075	203 731	54.035
等格网分块的生长法	79 079	160 153	96.255
本文算法	79 079	157 511	10.126

5 总结与展望

本文通过对 3 种主要的 TIN 模型的构建方法进行分析,在此基础之上针对三角形生长法进行扩展,使其能够很好地应用于大规模 LiDAR 点云的 TIN 模型构建,并且构建的结果完全使用原有的数据信息,无须进行插值和拼接过程,从而可以保证构建模型的精确性.为处理边界点,本算法通过添加边界辅助点避免了边界狭长三角形的生成.最后将该算法应用于美国新奥尔良市区的 LiDAR 数据进行构网,验证了算法的可行性.虽然在近些年内相对于逐点插入法和分治-合并法构网,生长法构网的研究较少,但本文的改进无疑是对生长法构网研究的一个有益的尝试和补充.

由于提高计算能力也是大规模点云快速构网的重要途径,将构网算法并行化已成为当前研究的热点[13−15],这也是我们下一步的研究方向.

References:

[1] Liang XL, Zhang JX, Li HT. Regular building model reconstruction from airborne laser scanning data. Journal of Image and Graphics, 2007,12(4):641−647 (in Chinese with English abstract).

[2] Sithole G, Vosselman G. Filtering of airborne laser scanner data based on segmented point clouds. In: Proc. of the Laser Scanning 2005. Enschede, 2005.

[3] Hu JX, Wu HP, Pan M, Ma ZT. Massive DEM creation with grid partitioning approach. Journal of Computer-Aided Design & Computer Graphics, 2004,16(1):41−44 (in Chinese with English abstract).

[4] Wu YG, Du Y, Wang XM, You X. Lod-Based algorithm of TIN model for large terrain simulation. Journal of System Simulation, 2005,17(3):665−669 (in Chinese with English abstract).

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

[6] Wu XB, Wang SX, Xiao CS. A new study of Delaunay triangulation creation. Acta Geodaetica et Cartographica Sinica, 1999,28(1): 28−35 (in Chinese with English abstract).

[7] He J, Dai H, Xie YQ, Liu BS. Fast improved Delaunay triangulation algorithm. Journal of System Simulation, 2006,18(11): 3055−3057 (in Chinese with English abstract).

[8] Luan XY. The method of building TIN and its three-dimensional display. Hydrographic Surveying and Charting, 2004,24(5): 39−41 (in Chinese with English abstract).

[9] Xu Q, Chang G, Yang L. The algorithm of TIN generation based on self-adapt clump organization. Journal of Image and Graphics, 2000,5(6):461−465 (in Chinese with English abstract).

[10] Jin WH, He T, Liu XP, Tang WQ, Tang RX. A fast convex hull algorithm of planar point set based on sorted simple polygon. Chinese Journal of Computers, 1998,21(6):533−539 (in Chinese with English abstract).

[11] Zhao WF. Improvement of generation algorithm of Delaunay triangulation network for the whole scatter points and software development. Engineering of Surveying and Mapping, 2003,12(4):22−25 (in Chinese with English abstract).

[12] Fang TP, Pieql LA. Delaunay triangulation using a uniform grid. IEEE Computer Graphics and Application, 1993,13(3):36−47.

[13] Chen MB, Chuang TR, Wu JJ. Parallel divide-and-conquer scheme for 2D Delaunay triangulation. Concurrency and Computation: Practice and Experience, 2006,18:1595−1612.

[14] Lee S, Park CI, Park CM. An improved parallel algorithm for Delauany triangulation on distribution memory parallel computers. In: Proc. of the 1997 Advances in Parallel and Distributed Computing Conf. (APDC'97). 1997. 131−138.

[15] Chen MB, Chuang TR, Wu JJ. A parallel divide-and-conquer scheme for Delaunay triangulation. In: Proc. of the 9th Int'l Conf. on Parallel and Distributed Systems (ICPADS). 2002. 571−576.

附中文参考文献:

[1] 梁欣廉,张继贤,李海涛.机载激光雷达数据的简单规则建筑物模型重建.中国图象图形学报,2007,12(4):641−647.

[3] 胡金星,吴焕萍,潘懋,马照亭.基于格网划分的海量 DEM 数据生成.计算机辅助设计与图形学学报,2004,16(1):41−44.

[4] 武玉国,杜莹,王晓明,游雄.大规模地形 TIN 模型的 LOD 算法设计与实现.系统仿真学报,2005,17(3):665−669.

[6] 武晓波,王世新,肖春生.Delaunay 三角网的生成算法研究.测绘学报,1999,28(1):28−35.

[7] 何俊,戴浩,谢永强,刘宝生.一种改进的快速 Delaunay 三角剖分算法.系统仿真学报,2006,18(11):3055−3057.

[8] 栾晓岩.一种 TIN 生成算法及其三维显示.海洋测绘,2004,24(5):39−41.

[9] 徐青,常歌,杨力.基于自适应分块的 TIN 三角网建立算法.中国图象图形学报,2000,5(6):461−465.

[10] 金文华,何涛,刘晓平,唐卫清,唐荣锡.基于有序简单多边形的平面点集凸包快速求取算法.计算机学报,1998,21(6):531−539.

[11] 赵文芳.离散点集 Delaunay 三角网生成算法改进与软件开发.测绘工程,2003,12(4):22−25.

刘晓平(1964－),男,山东济南人,博士,教授,博士生导师,主要研究领域为建模,仿真,协同计算.

袁晓辉(1973－),男,博士,助理教授,博士生导师,主要研究领域为计算机视觉,数据挖掘,人工智能.

朱晓强(1984－),男,硕士生,主要研究领域为计算机图形学,计算机图像处理.

Bill P. BUCKLES (1943－),男,博士,教授,博士生导师,主要研究领域为计算机视觉,数据挖掘,人工智能.

余烨(1982－),女,博士生,主要研究领域为建模,仿真,可视化.