**初稿**

Tinfour 图书馆的数据元素和算法

*用 Java 编写的不规则三角网 (TIN) 包*

G.W.卢卡斯 2021 年 2 月

版权所有 ©2016-2021 G.W.卢卡斯.允许制作和分发本文档的逐字副本，前提是其内容不以任何方式修改。保留所有其他权利。

i

**初稿**

**作者注**

本文件本质上是非正式的。其目的是为开发人员提供帮助。这不是一部学术著作。话虽这么说，我至少想把它做好。因此，如果您有任何更正或改进建议以使本文档更加有用，请随时告诉我。

**致谢**

我在宾夕法尼亚州立大学注册地理信息系统硕士课程时开始了 Tinfour 项目。该项目的灵感在很大程度上归功于 Justine Blanford，她的地理空间分析课程证明了该主题是多么有趣和有用。我还要感谢我的导师 Mike Renslow 和 Karen Schuckman，他们向我介绍了机载激光雷达，这项技术为 Tinfour 软件的性能考虑因素设定了如此高的标准。

我还要感谢我的朋友，已故的罗恩·巴顿 (Ron Patton)，是他首先让我对测深和水下环境建模的研究产生了兴趣。这些应用程序开启了我开发 Delaunay 三角测量软件的道路。我要感谢我的同事斯科特·马丁，他是我见过的最有才华的软件开发人员之一。

Scott 提供了许多使 Tinfour 成为可能的关键见解。

最后，我要感谢我的妻子 Kat，她容忍了我在创建这个软件时度过的许多个深夜和清晨。没有她的支持，我不可能完成实施。

ii

**初稿**

**目录**

1. 介绍 1
   1. 德劳内三角剖分 1
   2. 沃罗努图 4
2. 构建不规则三角网 (TIN) 6
   1. 性能和内存 6
      1. 构建网格的时间与样本大小的关系 6
   2. 构建三角形网格的算法和结构 8
      1. 通过增量插入构建网格 8
      2. 德劳内三角剖分 9
      3. 用于表示图的数据基元和结构 11
      4. 四边数据结构 11
      5. Ghost Vertex 和 Bootstrap 布局 13
      6. 边缘遍历和网格导航 16
      7. 四边结构的代码实现 17
      8. 顶点插入过程 19
         1. 通过边缘翻转进行简单插入 19
         2. 使用 Bowyer-Watson 算法提高性能 20
         3. Bowyer-Watson 插入概述 22
         4. 顶点位置 22
         5. 使用基于希尔伯特曲线的排序来减少步行长度 24
         6. 顶点唯一性 25
         7. 型腔创建 26
         8. 链接连接 26
      9. 坐标和数值问题 27
         1. 顶点的大数值坐标 27
         2. 地理坐标和投影坐标的特殊注意事项 29
      10. 约束 Delaunay 三角剖分 29
   3. 插值法 33

iii

**初稿**

* + 1. Tinfour 实施的技术 33
       1. 三角形面 (TriangleFacetInterpolator.java) 33
       2. 自然邻居（NaturalNeighborInterpolator.java） 33
       3. 地理加权回归 (GwrTinInterpolator.java) 33
    2. 交叉验证 34
    3. 地理加权回归 (GWR) 技术 35
       1. 表面模型 35
       2. 样本加权的带宽策略 38
       3. 解释结果 39
       4. 应用程序访问 GWR 结果 41
       5. GWR 实施状况 42
       6. GWR 技术的背景 43

测试和演示 44

* 1. 测试环境 44
     1. 命令行参数 44
  2. 应用示例 47
     1. 顶点文件中的高程和山体阴影网格示例 47
     2. 使用希尔伯特排序进行点细化 50
     3. 曲面插值的多个并发进程 50
  3. 测试应用 50
     1. 实施正确性的单一构建测试 50
     2. 性能评估的重复构建测试 52
     3. 用于调整性​​能和优化的孪生构建测试 52
     4. 由于样本大小，处理 TIN 的时间 54
  4. Tinfour 查看器 54

激光雷达数据样本 56

* 1. LAS 和 LAZ 格式 56
  2. 地面点和激光雷达数据分类 56
  3. 地理坐标 57

参考文献 58

iv

**初稿**

**图表**

图 1 – 非最佳三角剖分（左）和 Delaunay（右） 1

图 2 – 使用非结构化样本点创建的 Delaunay 三角剖分。 2

图 3 – 使用三角形网格从非结构化数据插值的表面。 3

图 4 – 全分辨率的山体阴影和高程编码激光雷达样本 4

图 5 – 来自非结构化样本的 Voronoi 图。 4

图 6 – 测量的网格构建时间 7

图 7 – 测得的网格构建时间（不包括对象分配） 8

图 8 – 使用增量插入构建网格 9

图 9 – 用于 Delaunay 三角剖分的外接圆准则 10

图 10 – 边 AB/BA 和邻居的四边数据结构 12

图 11 – 四边连接相邻三角形 ΔABC 和 ΔDBA 13

图 12 – 完成引导操作后的初始几何结构和链接 15

图 13 – 相邻边的遍历 16

图 14 – 通过翻转边缘恢复 Delaunay 属性 20

图 15 – Bowyer-Watson 插入的四个阶段 21

图 16 – 穿过三角形网格的“行走” 23

图 17 – 投影到希尔伯特空间填充曲线上的点 25

图 18 – 普通和约束 Delaunay 三角剖分 30

图 19 – 恢复最优性 31

图 20 – 普通 Delaunay、受约束的以及去除外部的受约束的。 31

图 21 – 使用约束区域功能渲染的图像 32

图 22 – 来自 Bear Mountain 激光雷达样本的山体阴影图像 49

图 23 – 根据康涅狄格州吉尔福德教堂街和 I-95 州际公路上的激光雷达数据构建的 Tinfour Viewer 图像（数据来源 NOAA，2011b） 54

图 24 – 沿海地区激光雷达样本的线框渲染（数据来源 NOAA，2011c） 55

v

**初稿**

**表格表**

表 1 – 四边 AB/BA 和邻居的链接 12

表 2 – 三角形 ΔABC 和 ΔDBA 的链接 13

表 3 – 引导操作完成后的链接 15

表 4 – 边遍历的伪代码 16

表 5 – QuadEdge 实例的内存使用情况 18

表 6 – 输入和输出选项 44

表 7 – 处理选项 45

表 8 – 随机样本生成选项 46

表 9——交叉验证应用的选项 47

vi

**初稿**

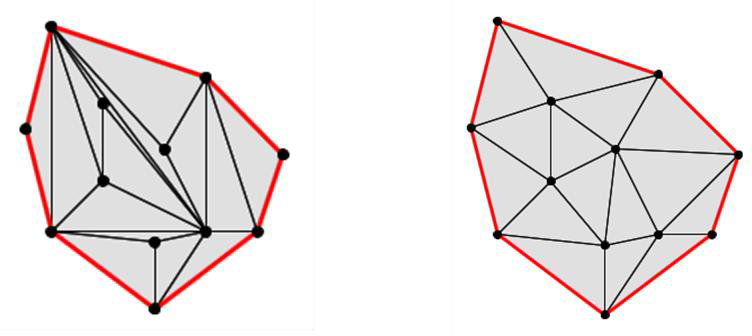
1. **介绍**

Tinfour 是一个基于 Java 的软件库，提供使用不规则三角网络 (TIN) 结构组织和建模非结构化数据的工具。

Tinfour 旨在成为一个免费的开源项目。因此，提供背景信息来帮助那些希望将代码用于自己的应用程序的人似乎是合适的。 《Tinfour》中没有任何东西可以用新颖来形容。它使用的技术直接源自计算几何的众所周知的结果。它的算法并不比你在大学编程课程中遇到的算法复杂。即便如此，构成软件基础的一些基本思想可能会被编码和实现细节所掩盖。考虑到这一点，我编写了这些注释，以帮助阐明其源代码中嵌入的逻辑和设计选择。 Tinfour 使用的算法取自已发表的作品，其中大部分可以追溯到 20 世纪 70 年代和 80 年代。为了方便读者，提供了引文。 Delaunay 三角剖分理论由 Boris Delaunay 于 1934 年提出。 Tinfour 中使用的许多概念都受到 Jonathan Shewchuk 作品的启发，特别是他的出色作品[三角形](http://www.cs.cmu.edu/~quake/triangle.html)图书馆（Shewchuk，1996）。

* 1. **德劳内三角剖分**

Tinfour 创建的主要结构是三角形网格，在 Delaunay 准则的意义上是最佳的。由于 Delaunay 三角剖分是图论中的一个众所周知的主题，并且在互联网上有广泛记录，因此简短的讨论就足够了。对于任何相当大的顶点集，有多种方法可以将它们连接到三角形网格中。但并非所有这些都可以很好地表示点之间的空间关系。 Delaunay 准则提供了一种选择在许多方面都最佳的网格的方法。为了说明这个想法，[图1](#_bookmark0)显示了当一组点以任意方式连接与遵守 Delaunay 准则的点连接时会发生什么情况的示例。



**图 1 – 非最佳三角剖分（左）和 Delaunay（右）**

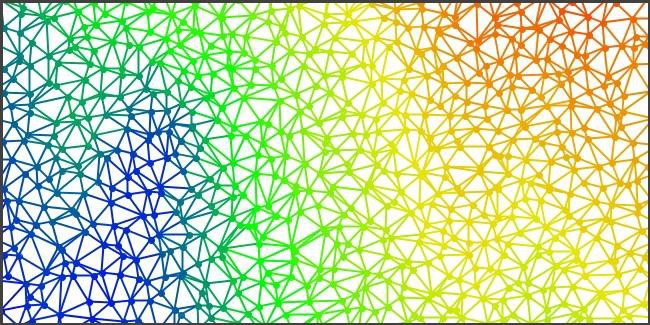
1

**初稿**

由于多种原因，Delaunay 三角剖分在对非结构化样本点集进行建模时具有优势。首先，它产生的三角形总体上比其他三角测量更接近等角。通过最小化“瘦”三角形的频率，Delaunay 网格提高了常用于曲面建模的浮点计算的数值稳定性。其次，Delaunay 确保表面上的绝大多数点将包含在由集合中最接近的样本点形成的三角形中。由于在执行插值时，最近的点通常是最相关的，因此 Delaunay 的这一属性会带来更准确的插值结果。最后，Delaunay 三角剖分提供了一种有效的工具，用于识别一组样本点内的特征和邻接关系。

[下图2](#_bookmark0)显示了如何将来自一组非结构化样本点的数据组织成三角网格的示例。该图的数据取自在康涅狄格州西北部熊山地区进行的基于激光雷达的高程调查（NOAA，2011a）。激光雷达系统使用低空飞行的飞机上的激光测距设备来获取表面高程数据。

这些系统提供高度准确和详细的表面视图，通常每平方公里收集数百万个样本。数据在[图2](#_bookmark0)占地 480 x 240 米。在原始激光雷达数据集中，该区域包含超过 133,000 个样本。出于描述目的，集合被二次采样到 675 个顶点。添加了颜色编码，蓝色色调分配给最低海拔，红色色调分配给最高海拔。

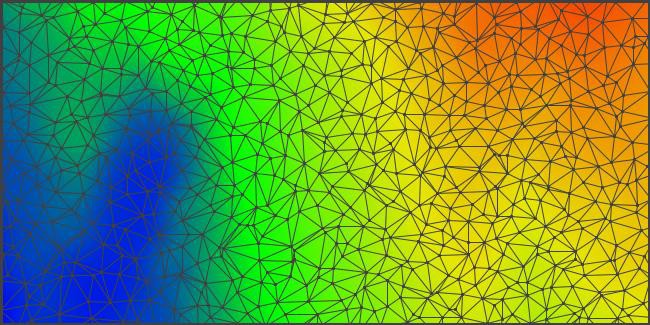


**图 2 – 使用非结构化样本点创建的 Delaunay 三角剖分。**

2

**初稿**

[图3](#_bookmark0)显示了通过使用称为自然邻点插值的技术从减少的采样点集导出的曲面。 Sibson (1981) 描述的该技术利用 Delaunay 三角剖分的特性，只需进行适度的处理即可产生令人愉悦的光滑表面。

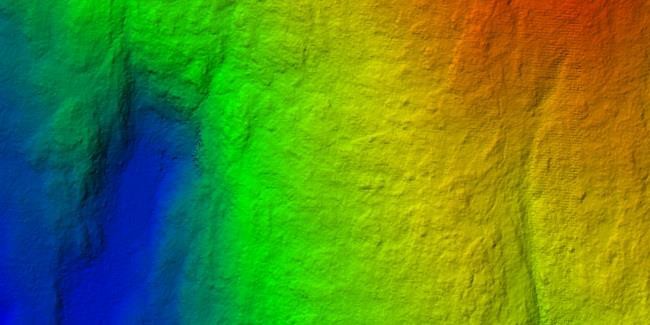


**图 3 – 使用三角形网格从非结构化数据插值的表面。**

最后，图 4 中的图像显示了使用上图所用的 Bear Mountain 激光雷达数据集全分辨率的效果。 Tinfour 用于从完整的样本集中构建三角形网格，为计算输出图像中每个位置的表面法线提供了一种有效的方法。一个简单的照明模型使用生成的表面法线来产生闪电和阴影效果。使用与上述两幅图像相同的调色板根据海拔分配颜色编码。图 4 中增强的详细程度展示了原始激光雷达调查中数据的质量。

3

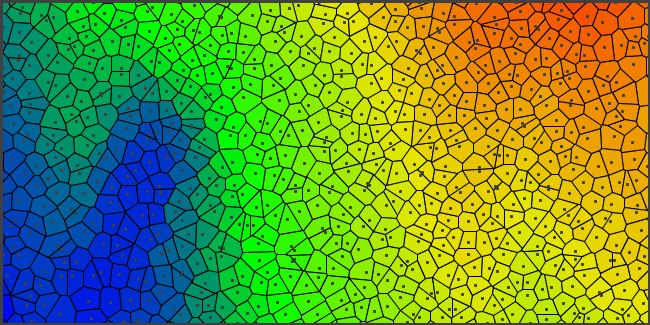
**初稿**



**图 4 – 全分辨率的山体阴影和高程编码激光雷达样本。**

**1.2 沃罗努图**

Delaunay三角剖分除了是计算几何中的一个重要结果之外，还与另一个重要的结构——Voronoi图密切相关。图[以下](#_bookmark0)显示了由与上图相同的采样点构建的 Voronoi 图。在图中，三角形网格的顶点被视为“种子”点，每个点都定义了平面的一个子区域。每个区域包含比数据集中的任何其他样本更接近其各自种子点的所有点的集合。



**图 5 – 来自非结构化样本的 Voronoi 图。**

4

与 Delaunay 三角剖分一样，Voronoi 图广泛用于分析表面上非结构化样本表示的数据。例如，考虑一些霉菌孢子随机分布在培养皿中的情况。随着孢子生长和繁殖，它们形成菌落，并以均匀的速度向外扩展，直到遇到其他菌落。菌落之间的边界构成了 Voronoi 图。或者，想象一种情况，我们希望构建机场服务区域的地图，以便飞行员在发生紧急情况时可以确定最近的机场。同样，这些区域的边界将构成一个 Voronoi 图。

Voronoi 图有时被称为 Delaunay 三角剖分的“对偶结构”。换句话说，对于每个 Voronoi 图，都有一个唯一的 Delaunay 三角剖分，反之亦然。这一特性使得可以通过最少的处理从另一种结构构造出一种结构（尽管有许多算法可以直接构造这些结构）。

尽管 Voronoi 图不是 Tinfour 项目的重点，但该库确实包含一个名为 BoundedVoronoi 的类，它将生成 Voronoi 图。通常，Voronoi 图是无界的。它的范围包括整个平面。出于软件目的，Tinfour 提供了一个 API，用于将 Vorionoi 结构限制在指定的（有界的）矩形区域中。

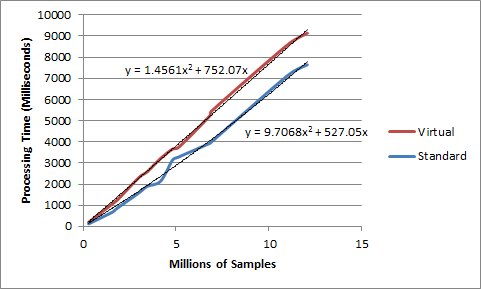
1. **构建不规则三角网 (TIN)**
   1. **性能和内存**
      1. 构建网格的时间与样本大小的关系

目前，Tinfour 网格构建实现有两个版本：标准版本和半虚拟版本。标准版本将所有边表示为内存中的对象。半虚拟版本将边维护为一组内存中的数据原语数组（整数、顶点引用等），并仅根据短期需要创建对象。标准版本比半虚拟版本更简单。它还往往比同类产品处理数据的速度更快。

不幸的是，标准版本比半虚拟边缘版本需要更多的内存：标准版本中每个样本 246 字节，而半虚拟版本中每个样本 120 字节。因此，虽然它不是真正的虚拟（核外）实现，但半虚拟版本避免了由于将边缘表示为持久对象而产生的大量内存开销。下面讨论这两种变体的细节。

图 6 中的图显示了使用俄勒冈州 Pole Creek 附近收集的激光雷达数据集在几次 Tinfour 运行中收集的计时值，该数据集包含超过 1200 万个样本，包括地面点和其他特征（USGS，2014）。使用名为 TimeDueToSampleSize 的应用程序处理数据，该应用程序是 Tinfour 软件发行版中包含的测试应用程序之一。为了测量样本大小对运行时间的影响，数据被随机二次采样为不同大小的较小数据集并由库进行处理。统计数据仅反映数据处理所花费的时间。读取输入数据文件和选择样本输入数据的时间被排除在测量之外。使用线性回归获得处理时间的趋势线和方程。

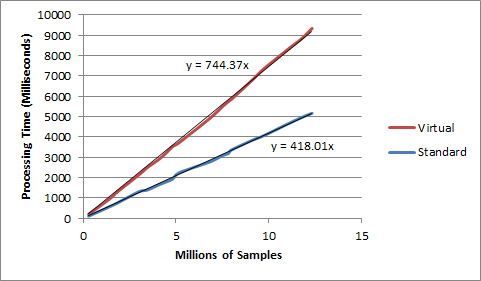
此测试的结果是使用在 Windows 7 主机上运行的 Oracle HotSpot Java 虚拟机 (JVM) 获得的，该主机配备 8 GB 内存、2.9 GHz CPU、512 KB L2 快速缓存内存和 4096 KB L3 内存。 Tinfour 构建过程是一个非并发（单线程）过程，因此处理器的数量很大程度上无关。当时没有其他主要进程正在运行。因此，此测试的结果取决于有利的条件，这些条件并不适用于所有情况，但在具有充足系统资源的生产环境中运行 Tinfour 时并非不现实。 在标准配置的情况下，该库平均每秒处理 186 万个样本，尽管它超过了较小样本量的处理速度。速度较慢但内存密集程度较低的半虚拟配置平均每秒处理 134 万个样本。



**图 6 – 测量的网格构建时间**

与大多数当代软件一样，Tinfour 库基于面向对象的方法。在标准实现中，三角形网格的元素（顶点和边）均由单独的对象表示。在标准实现中，源数据中的每个样本平均需要 7 个对象实例：每个顶点一个对象，6 个对象代表边。另一方面，半虚拟实现不使用显式对象来表示边缘，因此每个样本平均只需要 1.019 个对象：一个用于顶点，少量用于内部簿记。但即使由于半虚拟表示而有所减少，一组 1200 万个样本仍然需要大量的对象。

在大多数 Java 应用程序中，与构造对象相关的开销可以忽略不计。然而，传统的 Java 应用程序很少一次创建一百万个对象。当处理包含数百万个点的样本集时，对象的创建（以及随之而来的不可避免的垃圾收集）是处理成本的重要组成部分。[图7](#_bookmark0)显示了使用与上述相同的设置进行的计时测试的结果，但不包括对象分配的成本。图中的方程没有显示二次项，因为它们的量级非常小并且不具有统计显着性。请注意，半虚拟配置（为每个样本分配较少的对象）的性能变化比标准配置小得多。



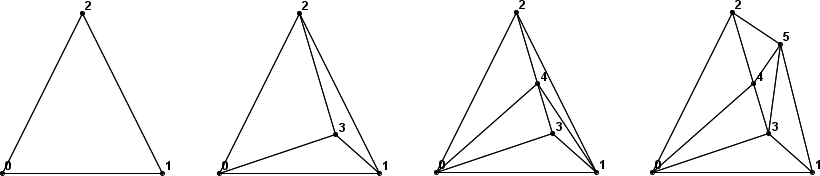
**图 7 – 测量的网格构建时间，不包括对象分配。**

Tinfour 确实允许应用程序在单独处理多个数据集的情况下降低对象分配和垃圾收集的成本。该库的主要类 - IncrementalTin 和 SemiVirtualIncrementalTin - 本质上是顶点的集合。当处理多个数据集时，可以通过在任务之间清除其内容来为每个数据集重用这些类的单个实例。清除内容时，不会删除为表示 TIN 中的边而构造的内部对象。相反，它们只是被标记为可重复使用。这样做既避免了分配新对象的成本，也避免了垃圾收集带来的开销。

* 1. **构建三角形网格的算法和结构**
     1. 通过增量插入构建网格

构造最佳三角网格的问题是计算几何中的一个重要课题，并已得到广泛研究。 Su 和 Drysdale (1996) 确定了构建三角网格的三大类算法：分治法、扫线法和增量插入法。 Tinfour 库使用增量插入算法。在此过程中，使用“引导”过程创建三个顶点的初始网格。一旦构建了初始网格，顶点就会一次添加到网格中。该过程如下图 8 所示。顶点 3 和 4 插入到现有网格的内部。顶点 5 延伸网格。请注意，每次添加都会改变三角形的结构，并有可能破坏先前存在的边（线段）。例如，插入顶点 4 具有破坏边 2-3 并用新边 3-4 和 2-4 替换它的效果。

**图 8 – 使用增量插入构建网格**



在构建包含大量折点的 TIN 的过程中，可能会多次构建并随后替换边。作为常规处理的一部分，Tinfour 会跟踪更换操作的数量。在使用来自 Bear Mountain 样本的激光雷达数据进行测试时，平均替换次数约为 6.5（对于一组超过 300 万条边）。该统计数据表明，处理边替换的有效方法是设计良好的 TIN 实现的要求。

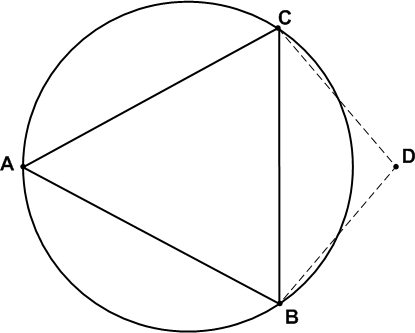
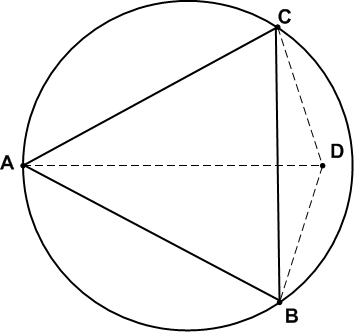
Tinfour 通过使用称为 EdgePool 集合的可重用对象池来实现这种效率。标准和半虚拟实现使用略有不同的 EdgePool 版本。

图[多于](#_bookmark0)还说明了 TIN 的一个显着特征。 TIN 的周长始终是凸多边形。

2.2.2 Delaunay三角剖分

如上所述，Tinfour 实现的基本产品是 Delaunay 三角剖分。 Delaunay 准则要求构造三角形网格，使得没有点位于它不属于的三角形的外接圆内。在左侧[下图9](#_bookmark1)，点 D 不在三角形 ΔABC 的外接圆内，因此一对三角形 ΔABC 和 ΔCBD 满足 Delaunay 准则。如果点 D 在外接圆内（如右图所示），则需要通过翻转边 BC 来重新组织三角剖分，使其连接边 AD 形成两个交替三角形 ΔABD 和 ΔDCA。请注意，在这两种情况下，始终给出点，以便它们按逆时针顺序指定三角形的边。

每次将新顶点插入三角形网格时，Tinfour 都会根据需要调整局部边缘，以确保遵守标准。因此，在施工的所有阶段，软件都会保持正确的 Delaunay 三角测量。

**图 9 – 用于 Delaunay 三角剖分的外接圆准则**

Cheng (2013, p. 57) 提供了一种计算方法，用于确定由坐标 (𝑑𝑥, 𝑑𝑦给出的点 D 是否在坐标为 (𝑎𝑥, 𝑎(𝑦、 (𝑏𝑥、𝑏𝑦和 (𝑐𝑥, 𝑐𝑦通过评估以下行列式 ：

𝑎 − 𝑑 𝑎 − 𝑑 (𝑎 − 𝑑 )2 + (𝑎 − 𝑑 2

𝑥 ) 𝑦 𝑦 𝑥 𝑥 𝑦 𝑦

内圆(a,b,c,d) =

𝑏 − 𝑑

𝑏 − 𝑑

(𝑏

− 𝑑

)2 + (𝑏

2

− 𝑑

| 𝑥

𝑐

𝑥

− 𝑑

𝑦 𝑦

𝑐 − 𝑑

𝑥

(𝑐

𝑥

− 𝑑

𝑦

)2 + (𝑐

− 𝑑

𝑦) |

2

𝑥 𝑥 𝑦 𝑦

𝑥 𝑥

𝑦 𝑦)

如果 InCircle(a,b,c,d) 的值大于零，则 D 位于 ΔABC 的外接圆内，并且违反 Delaunay 准则。为了恢复 Delaunay 属性，我们必须执行如上所述的边缘交换操作。如果该值小于零，则 D 在外接圆之外并且满足标准。如果该值恰好为零，则该点位于外接圆上，并且根据 Delaunay 准则，任何一种点排列都是可接受的。在这种不明确的情况下，必须应用一些其他标准来选择首选结构。

检查图纸时出现的一个问题[图9](#_bookmark1)是 D 点是否是

在 ΔABC 外接圆之外告诉我们可以确定 A 点在 ΔCBD 之外。稍微思考一下就会发现，InCircle(c,b,d,a) 的计算相当于交换 InCircle(a,b,c,d) 行列式中的行偶数次，根据 row 属性行列式，将产生与原始顺序相同的值。事实上，任何保留三角形顶点逆时针顺序的行排列总是需要偶数次交换。因此，只需评估一个行列式即可决定边缘翻转操作是否必要。

* + 1. 用于表示图的数据基元和结构

三角网格可以被视为由三个几何基元组成：

* + - 1. 顶点
      2. 边缘
      3. 三角形

Delaunay 证明，随着 Delaunay 三角剖分中顶点数量的增加，每种特征的数量接近以下值：

对于 N 个顶点：

* + - * + N×3 边
        + N×2 三角形
        + 每个顶点平均有 6 条边连接

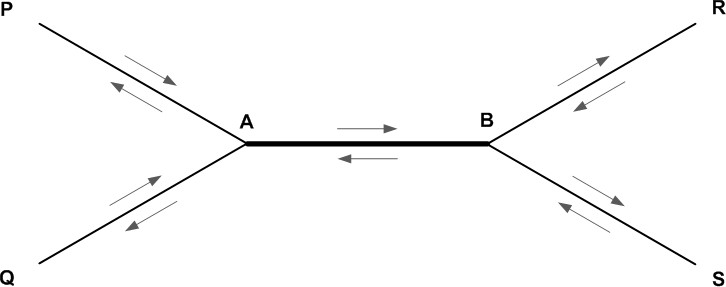
除外边界附近外，整个 TIN 的所有足够大的子区域都维持这些关系。

对于数据集中的每个样本，我们构造一个顶点。在激光雷达勘测等数据集中，样本数量通常为数百万，因此边和顶点的数量也同样很大。

* + 1. 四边数据结构

Tinfour 创建的三角形网格是由使用四边数据结构表示的链接边的集合构建的，该数据结构由 Guibas 和 Stolfi 在 1980 年代中期推广（Guibas，1985，第 74 页）。它适合构建许多不同类别的基于多边形的图，包括 Delaunay 三角剖分和 Voronoi 图。

四边结构的单个实例用于表示由一对顶点和链接到 4 个相邻边组成的单个边。如图所示[图10](#_bookmark0)下面，顶点 A 和 B 定义了线段 AB 及其“对偶”BA。 Tinfour 中的边始终被视为具有方向，并且每条边都有相反方向的对偶。边缘的链接取决于它们的方向。来自 AB 的前向链路将由顶点 BR 的第二个四边形表示。 AB 的反向边将是四边 PA。总的来说，这些四边形可以用来指示多边形的存在。在 TIN 中，所有此类多边形都是三角形，并且所有链接均已填充，但此限制不一定适用于其他类型的图形。边缘的链接在[表格1。](#_bookmark1)

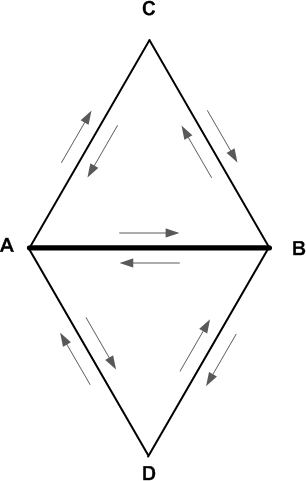


**图 10 – 边 AB/BA 和邻居的四边数据结构**

**表 1 – 四边 AB/BA 和邻居的链接**

|  |  |  |  |
| --- | --- | --- | --- |
| **边缘** | **向前** | **撤销** | **双重的** |
| AB | 巴西 | PA系列 | BA |
| BA | 空气质量 | 某人 | AB |
| PA系列 | AB | 未显示 | 美联社 |
| 美联社 | 未显示 | 质量保证 | PA系列 |
| 质量保证 | 美联社 | 未显示 | 空气质量 |
| 空气质量 | 未显示 | BA | 质量保证 |
| 巴西 | 未显示 | AB | RB系列 |
| RB系列 | 学士 | 未显示 | 巴西 |
| 学士 | 未显示 | RB系列 | 某人 |
| 某人 | BA | 未显示 | 学士 |

两个相邻的三角形将如下图 11 所示表示。



**图 11 – 四边连接相邻三角形 ΔABC 和 ΔDBA**

**表 2 – 三角形 ΔABC 和 ΔDBA 的链接**

|  |  |  |  |
| --- | --- | --- | --- |
| **边缘** | **向前** | **撤销** | **双重的** |
| AB | BC | CA | BA |
| BC | CA | AB | CB |
| CA | AB | BC | AC |
| BA | AD | DB | AB |
| AD | DB | BA | DA |
| DB | AB | AD | BD |

Tinfour 中的网格表示未将三角形指定为显式对象。三角形由与网格集合中的边集关联的链接暗示。表示顶点的数据对象不携带任何将它们显式绑定到边的信息。边知道顶点，顶点不知道边。因此，使用 Tinfour 的软件可以将顶点定义为不可变对象，或者只是将它们传递到库，而不必担心它们会被更改。

2.2.5 Ghost Vertex 和 Bootstrap 布局

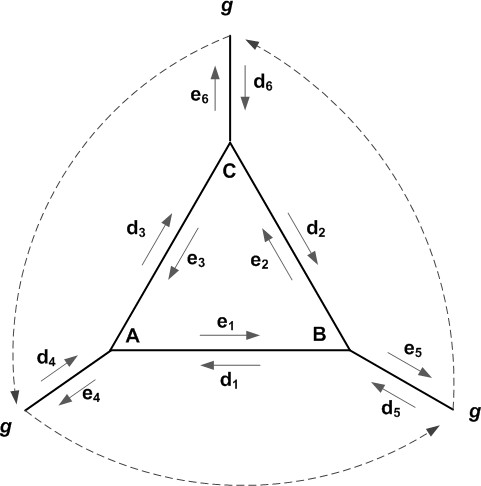
由于前面的两个示例都只是网格的片段，因此未记录某些链接。使用四边结构构造三角形网格的一个关键因素是规定结构中的每个链接都必须填充。这样做简化了许多编码问题，但确实需要特殊的逻辑来处理位于网格周边的边。

在不同的三角网格实现中，有不同的策略来避免或以其他方式管理空链接，Tinfour 依赖于称为“幽灵顶点”的概念（Cheng，2013，第 61 页）。

想象一个简单的三角形网格，包含一个具有三个周边的三角形。为了填充这些边的空链接，Tinfour 指定存在一个假想点（幻影顶点），该点连接到 TIN 周边上的每个顶点。通过这样做，它可以确保周界边缘的前向和反向链接都已填充。一些实现通过想象鬼点存在于网格中所有其他点的高阶维度中来为鬼点提供实际的几何规范。例如，在平面上组织的一组坐标的 2D 三角形网格中，鬼点可以被视为存在于三维中，在平面上方升高一定距离。 Tinfour 做了一些不同的事情，将幽灵顶点实现为空对象引用。

下面的图 12 说明了包含三个点的网格的链接，因为它将在初始引导操作后配置。图中，实线是实际的边，虚线表示连接，箭头表示链接方向。该网格由三个实际顶点（A、B 和 C）和一个虚拟顶点组成。尽管图中的幻影顶点显示在三个位置，但它是单个实体，因此始终标记为 g。

除了确保没有边具有空链接之外，引导操作还建立了将在所有后续点插入中维护的几何关系。特别是，三角形 ΔABC 内边的前向链接为三角形建立了逆时针顺序。 Tinfour 按逆时针顺序维护 TIN 内部的所有三角形。虽然外部链接没有真实的几何形状（因为幻影点为空），但会根据每个环所包含的周边边缘的方向对每个环施加一个顺序。



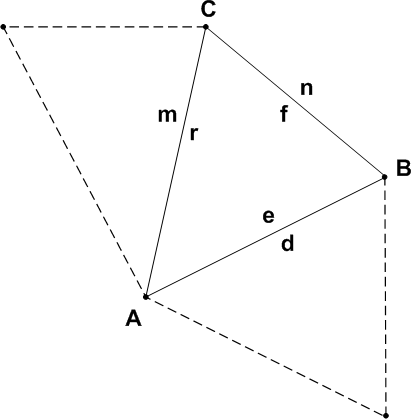
**图 12 – 引导操作完成后的初始几何结构和链接。**

**表 3 – 引导操作完成后的链接**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **边缘** | **起始顶点** | **结束顶点** | **向前** | **撤销** |
| e1 | A | B | e2 | e3 |
| e2 | B | C | e3 | e1 |
| e3 | C | A | e1 | e2 |
| e4 | A | g | d5 | d1 |
| e5 | B | g | d6 | d2 |
| e6 | C | g | d4 | d3 |
| d1 | B | A | e4 | d5 |
| d2 | C | B | e5 | d6 |
| d3 | A | C | e6 | d4 |
| d4 | g | A | d3 | e6 |
| d5 | g | B | d1 | e4 |
| d6 | g | C | d2 | e5 |

* + 1. 边缘遍历和网格导航

Tinfour 库中的许多操作都涉及某种从一条边到相邻边的遍历。例如，给定起始边，可以通过移动前向链接直到遍历返回到原始边来构造三角形。[下图13](#_bookmark0)显示从起始边 e 到其附近边的遍历。



**图 13 – 相邻边的遍历**

**表 4 – 边遍历的伪代码**

|  |  |  |
| --- | --- | --- |
| **边缘** | **顶点** | **链接** |
| e | AB | 起始边 |
| d | BA | e.getDual() |
| f | BC | e.getForward() |
| r | CA | e.getReverse()，还有 e.getForward().getForward()。 |
| n | BC | e.getDualFromForward() |
| m | AC | e.getDualFromReverse() |

如上所述，Tinfour 保持链接，以便形成网格的所有三角形在前向遍历下按逆时针顺序定向。因此，三个后续 getForward() 操作的结果会形成一个完整的三角形循环。

作为边缘遍历的最后一个说明，以下 Java 代码片段显示了一个绰号为“风车”的操作。该代码收集连接到中心的所有顶点的列表

通过一组连接边“锚定”顶点。在操作开始时，我们得到一条以锚点顶点 A 开头的边。该边的 getA() 方法将获取锚点顶点。 getB() 方法获取边另一端的顶点。在接下来的循环中， getDualFromReverse() 方法用于遍历连接到锚点的边，以便提取相邻顶点并将其添加到结果列表中。当遍历围绕锚点顶点完成完整循环并到达初始边缘时，收集工作终止。

IQuadEdge e； // 给定 e 从顶点 A 开始

ArrayList<顶点> 结果 = new ArrayList<>(); // 顶点集合

IQuadEdge 光标 = e;做{

顶点 b = 游标.getB();结果.add(b);

光标 = 光标.getDualFromReverse();

}while(!cursor.equals(e));

在 Tinfour 开发过程中，我们遇到了很多需要类似风车操作的网格处理应用程序，因此我们添加了一个便利功能来简化其使用。给定起始边 e，我们可以使用以下代码实现与上所示相同的结果：

for(IQuadEdge 光标: e.pinwheel()){ result.add(cursor.getB();

}

* + 1. 四边结构的代码实现

当我们考虑边缘遍历应用程序的实际问题时，我们经常发现导航三角形网格的代码需要了解边缘遍历的方向。在图论中，三角网格是无向图。但出于软件目的，为各个边缘提供方向感是很有用的。因此，如果我们希望使用 Java 对象表示边，那么有关方向的信息必须是 Java 类设计的一部分。 Tinfour 通过将每条边实现为一对链接对象来满足这一要求，每个链接对象对应一个遍历方向。实际上，它将四边结构分成两部分。每个部分都是一个单独的 Java 对象。每件作品都有其对偶的参考。两个部分同时实例化，并通过设置对其对应部分的双重引用将其联系在一起。

边缘表示的主类名为 QuadEdge。 QuadEdge 的每个实例都伴随有来自 QuadEdgePartner 类的伴生对象，该类派生自 QuadEdge。因此，TIN 中的每条边都有两个关联的对象。由于为 Delaunay 三角剖分中的每个顶点构建了大约 3 个边对，并且数据样本中的顶点数量可能达到数百万，因此完全填充的 TIN 中的对象实例数量可能会变得相当大。

因此，紧凑的类设计对于节省内存至关重要。例如，两个顶点定义一条线段，因此每条边都需要引用两个顶点对象。但 QuadEdge 实现仅实现了一个。由于 QuadEdge 对象始终与 QuadEdgePartner 对象关联，因此每个对象只需携带一个引用。该对任一侧的第二个顶点参考始终可以从其对应部分获得。

当在具有压缩引用选项的 HotSpot 虚拟机下运行时，QuadEdge 对象的每个实例需要 32 个字节。 QuadEdgePartner 也有同样的要求。[表5](#_bookmark1)显示类中元素的布局。因为每条边都需要一对对象，所以每条边需要 2×32=64 字节的内存。由于每个顶点有 3 个边对，因此 QuadEdge 表示的每个顶点内存使用总量为 3×64=196 字节。 Vertex 类本身的实例需要 40 个字节。因此，每个数据样本（包括边和顶点）的平均内存使用量为 196+40=226 字节。

由于 JVM 内存管理而产生的额外开销将该值提高到第 2.1 节性能和内存中引用的 246 字节。

**表 5 – QuadEdge 实例的内存使用情况**

|  |  |
| --- | --- |
| **元素** | **尺寸** |
| Java管理开销 | 8字节 |
| 参考Java类定义 | 4字节 |
| 参考双 | 4字节 |
| 参考顶点 A | 4字节 |
| 参考前缘（链接） | 4字节 |
| 参考反向边缘（链接） | 4字节 |
| 整数边索引（Tinfour记账） | 4字节 |
| **全部的** | **32字节** |

QuadEdge 类中的一些字段用于表示实际数据，但每个实例的 12 字节是由于 Java 中分配对象的不可避免的成本。这相当于每个边对 24 字节的开销，或者每个顶点平均 72 字节的开销。最重要的是，创建如此多的对象在构造和垃圾收集方面增加了相当大的处理器开销。

SemiVirtualEdge 实现并不将三角形网格中的边表示为实际对象，而是将边的元数据存储在传统数据基元中。当需要与边缘相关的对象时，它们会根据需要构造并快速丢弃。由于边不作为持久对象存在，因此半虚拟实现构造的对象数量减少了 1000 倍以上。因此，SemiVirtualEdge 实现将所有边的总成本降低到平均每个顶点 76.07，每个样本总共使用大约 120 字节，包括顶点内存和 JVM 开销。

由于 SemiVirtualEdge 实现减少了内存使用，因此它允许应用程序处理更多的顶点，而无需增加 JVM 的最大内存分配。此外，

通过减少构造的对象数量，它大大减轻了垃圾收集的负担。不幸的是，内存的减少有其自身的代价。访问 Java 数组以及解释正向和反向链接的整数索引所需的开销大大增加了运行时间。正如上面图 6（网格构建的测量构建时间）所示的两种变体的测量构建时间所示，SemiVirtualEdge 变体需要比标准 QuadEdge 实现多 40% 的时间来处理 100 万个点。

* + 1. 顶点插入过程

Tinfour 使用基于 Bowyer (1981) 和 Watson (1981) 两篇著名论文的算法将顶点插入网格中。这些论文之所以出名，是因为它们几乎在同一时间提交给同一期刊，并且都提出了重要且密切相关的结果。当《计算机杂志》的编辑收到这两篇论文时，他们选择在同一期并列发表它们。

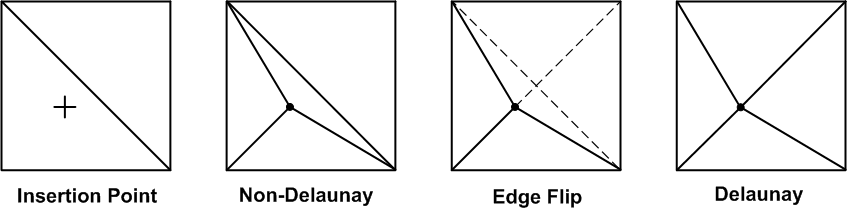
作为介绍，我将用一种更早且更简单的技术来讨论 Bowyer-Watson 算法，以说明其一些基本原理。 Lawson (1977) 描述的边缘翻转算法实际上是 Tinfour 实现的第一个算法。它具有代码紧凑且易于理解的吸引力。然而，当用 Bowyer-Watson 方法取代它时，构建 TIN 所需的时间减少了 50%。

* + - 1. ***通过边缘翻转进行简单插入***

Lawson 的原始算法使用简单的插入过程创建 Delaunay 网格。从三个点的初始网格（Tinfour 称之为“引导网格”）开始，该算法使用以下步骤插入每个顶点：

* + - * 1. 找到包含的三角形。
        2. 通过将顶点链接到现有三角形中的每个顶点，将顶点插入三角形中。
        3. 根据需要递归“翻转”边以恢复 Delaunay 属性。

劳森方法的关键是第三步。当将顶点插入到包含三角形中时，任何或所有生成的三角形都可能是非 Delaunay 的。如果不进行某种校正，结果将逐渐具有与上面图 1 中给出的非 Delaunay 网格与 Delaunay 三角剖分示例相同的次优外观。劳森通过测试每条新边来查看其相对边上的三角形是否满足德劳内标准，从而恢复了德劳内性质。如果没有，边缘将被“翻转”，从而产生一组备用三角形，如下图所示。



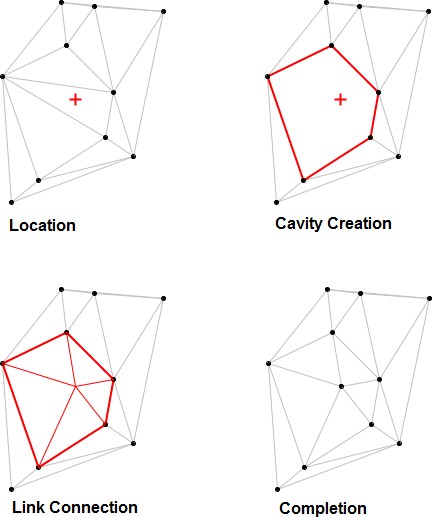
**图 14 – 通过翻转边缘恢复 Delaunay 属性**

不幸的是，当“非德劳内”边被翻转时，恢复德劳内最优性的工作不一定完成。当插入点位于紧邻三角形的外接圆内时，它也可能位于与邻居相邻的一个或多个三角形的外接圆内。因此，当插入逻辑检测到非 Delaunay 三角形时，它必须递归搜索“邻居的邻居”，寻找需要翻转的附加边以恢复 Delaunay 最优性。幸运的是，当搜索遇到“Delaunay 边”（不需要翻转的边）时，不需要继续超出该点。另外，如果搜索遇到周界边缘，则无需继续进一步。因此递归搜索将始终终止

即使保证终止，递归搜索也可能向外辐射并影响相邻三角形的几层。有几层？理论上，插入可以影响整个 TIN。在处理熊山样本时，早期的实现遇到了翻转操作向外辐射到周围43层三角形的情况。在实践中，Delaunay 特性的恢复通常涉及不超过两层（或六条边）。即便如此，与测试和修改边缘链接相关的开销足以保证采用替代方法。

* + - 1. ***使用 Bowyer-Watson 算法提高性能***

使用 Bowyer-Watson 算法插入顶点分 4 个阶段进行，如图所示[如下图 15。](#_bookmark2) 一旦找到包含三角形，该过程就会通过删除非 Delaunay 边在 TIN 中创建空腔。然后它将插入顶点连接到空腔的内边缘，恢复三角形网格。 Bowyer 和 Watson 的论文表明，生成的网格是 Delaunay 最优的。



**图 15 – Bowyer-Watson 插入的四个阶段**

作为进一步的改进，Tinfour 将型腔创建和链接连接步骤合并到单个操作中。这样做可以减少必须重新分配边缘链接的次数，从而提高插入例程的性能。然而，它确实使代码变得复杂。为了清楚起见，这些注释将插入算法描述为单独的步骤。对实际实现细节感兴趣的读者可以查看 IncrementalTin 类中 addWithInsertOrAppend() 方法的源代码。

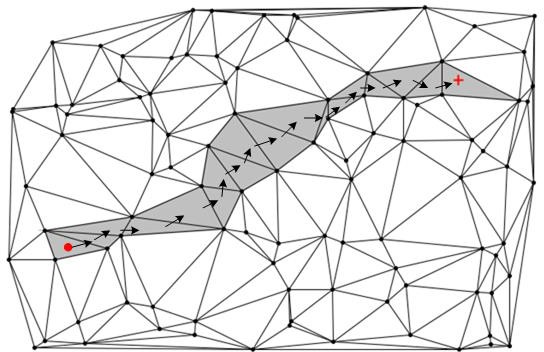
顺便提及，术语“顶点插入”也用于描述要添加的顶点位于TIN之外的情况。 Cheng (2013) 详细介绍了如何通过对下述整体逻辑进行微小改动来处理“幽灵三角形”（包括周边边缘和幽灵顶点）（第 59 页）。

* + - 1. ***Bowyer-Watson 插入概述***

一旦引导操作完成并且初始三角网格可用，BowyerWatson 算法将使用以下步骤将顶点插入网格中：

* + - * 1. 位置：对于要插入的每个顶点，确定包围的三角形。如果折点位于 TIN 外部，则定位幻影三角形，使三角形的周边最接近插入点。
        2. 唯一性：根据定义，TIN 中的每个折点都必须具有唯一的水平坐标。将顶点添加到 Tinfour 时，它会根据封闭三角形的三个顶点来测试插入顶点，以确定是否不同。如果插入折点不唯一，则不会将其添加到 TIN 中。相反，它与“顶点组”中预先存在的顶点组合。 TIN 的结构未更改。
        3. 插入：如果插入顶点是唯一的，则识别必须连接到插入顶点的网格顶点，根据需要删除边以确保网格保持 Delaunay 最佳状态（此步骤还包括当添加的顶点位于插入顶点的周长之外时扩展网格）纳税人识别号码）。
      1. ***顶点位置***

Tinfour 定位包含插入顶点的三角形的最直接方法是顺序搜索所有现有三角形，直到找到匹配项。不幸的是，这样的过程很慢，时间复杂度为 𝑂(𝑛2)，具体取决于输入集中的顶点数量。 Lawson (1977) 提出了一种使用“步行”算法的更快方法。[图16](#_bookmark3)说明了 Delaunay 三角剖分中两个三角形之间遍历的概念。只要算法能够识别合理的直接路径，遍历的步骤数就会大大少于网格中的顶点数。由于这样的路径很容易从 Delaunay 三角测量中获得，因此插入算法可以使用它来加快点定位过程。 Soukal (2012) 对步行算法进行了全面的讨论。



**图 16 – 穿过三角形网格的“行走”**

Tinfour 使用以下步骤执行顶点定位操作：

* + - * 1. 回想一下，网格中的所有三角形都按逆时针方向排列。因此，如果一个顶点包含在三角形中，则它将位于每个内边左侧的半平面内。
        2. 对于引导后的第一次插入，使用初始三角形的内侧之一选择“起始边”。对于所有后续搜索，从最近构建的三角形中选取起始边。
        3. 测试插入顶点是否位于起始边的左侧或左侧。如果是，则继续步骤 4。如果不是，则它将位于起始边对偶的左侧，因此转移到起始边的对偶。
        4. 重复以下步骤，直到找到包含的三角形或遍历转移到 TIN 的外部：

获取前缘。如果顶点位于前边的右侧，则转移到其对偶并继续步骤 5。

获取反向边缘。如果顶点位于反向边的右侧，则转移到其对偶并继续步骤 5。

如果顶点位于正向边和反向边的左侧，则它必须位于当前三角形的内部（或边上）。遍历结束。

* + - * 1. 搜索已转移到边的对偶，使得顶点位于该边的左侧。如果该边是内边，则从步骤 4 继续搜索。
        2. 如果该边是外部边，则通过移动到左或右周边边缘直到找到对向边来识别对向顶点的边。终止搜索。

上述步骤适用于独特、最佳的 Delaunay 三角剖分。不幸的是，非最佳网格可能包括游走算法落入循环路径且永远不会出现的区域。

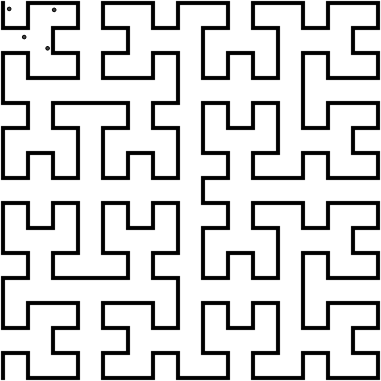
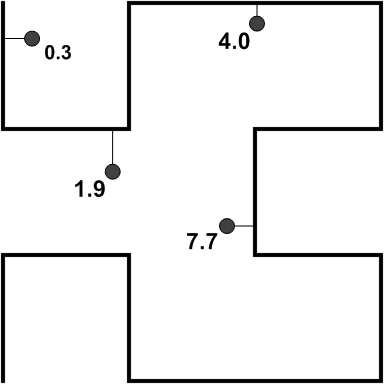
到达包含三角形。 Lawson 表明，可以通过随机交替步骤 4.a 和 4.b 中考虑前向或反向边缘的顺序来避免无限循环。即使遍历落在三角形跳跃的潜在循环序列中，如果遍历可以切换考虑相邻边的顺序，它最终也会转出循环。由于游走算法中的随机化元素，这种方法通常被称为“随机劳森游走”。

在每次操作之间，Tinfour 都会跟踪所谓的“起始边缘”，以便后续的每次步行都从前一次步行结束的地方开始。如果整个样本集中的两个后续顶点间隔很近（与其他顶点之间的距离相比），则步行操作所需的步数会减少。另一方面，如果样本集是随机定位的，则游走操作将倾向于在样本域中来回跳跃，从而导致游走操作的总长度增加。因此，当后续顶点往往比非后续顶点更靠近时，Tinfour 的行走操作往往会更有效。这样的数据集具有“高度的顺序空间自相关性”，可以比那些不具有这种特性的数据集更有效地处理。幸运的是，这正是典型激光雷达数据集中的情况。 由于激光雷达数据集中的点是使用扫描激光收集的，并且大多数激光雷达样本都是按照收集的顺序给出的，因此从激光雷达导出的顶点通常具有高度的顺序空间自相关性。对于 Bear Mountain 数据集，使用 Lawson 步行算法平均需要 3.38 步才能找到包含顶点的三角形（该值是使用下面描述的 SingleBuildTest 获得的）。

* + - 1. ***使用基于希尔伯特曲线的排序来减少步行长度***

有一个明显的情况，顺序空间自相关的假设不适用：随机样本。当在输入域中的随机位置给出样本时，一个样本不太可能位于其前一个样本附近。对于随机定位的样本，平均游走的长度往往与网格中点数的平方根成正比（例如，它与跨点集合的对角线长度成正比）。点位置的时间复杂度通过大量顶点添加进行整合，将接近 𝑂(𝑛3⁄2)。

为了减少插入一组顺序空间相关性较差的点所需的步骤数，Tinfour 库实现了一个类，用于使用基于希尔伯特空间填充曲线（Hilbert，1891）的排序方案对样本进行排序。样本中的每个点都投影到希尔伯特曲线的最近的一段上，并根据其沿曲线的距离分配一个排序键，如下所示[如下图17。](#_bookmark4) 由于希尔伯特曲线自行向后折叠，因此靠近的点往往具有相似的距离值。因此，排序确保了位置接近的点在生成的样本序列中彼此靠近。该操作极大地提高了样本集的顺序空间自相关性。因此，顶点定位过程的时间复杂度降低到 Java 排序本身的时间复杂度，通常优于 𝑂(𝑛 ∙ log 𝑛)。

**图 17 – 投影到希尔伯特空间填充曲线上的点**

尽管希尔伯特排序在处理自相关性较差的样本时可能很有用，但它并不适合所有数据集。例如，激光雷达样本很少需要希尔伯特排序，因为它们通常具有高度的顺序空间自相关性。事实上，这种排序可以通过添加额外的步骤来增加激光雷达样本的整体处理时间，该步骤本身会产生前期成本，并且只能适度减少顶点定位时间。例如，对 Bear Mountain 样本执行希尔伯特排序将平均遍历长度从 3.38 步减少到 3.12 步。当使用希尔伯特排序选项测试该样本时，与未排序的输入相比，构建 TIN 所花费的时间减少了 106 毫秒。不幸的是，排序本身花费了 236 毫秒。因此，在构建 TIN 之前对折点进行排序导致总体处理时间净增加 130 毫秒。 显然，熊山样本不是希尔伯特排序的良好候选者。但如果应用程序先验地知道样本具有较弱的顺序空间相关性，则它可以提高处理效率。该排序在诸如 Tinfour Viewer（如下所述）之类的应用程序中也很有用，其中同一数据集被处理多次（以便单次排序的成本在许多后续操作中分摊）。

计算希尔伯特“等级”的逻辑基于 Warren 中描述的 Lam & Shapiro 方法（2013 年，第 358 页）。

* + - 1. ***顶点唯一性***

Tinfour 测试每个插入顶点，以确保它基于最小距离标准是唯一的。如果顶点的水平坐标与现有顶点的水平坐标相同或几乎相同，则不会将其插入网格中。相反，Tinfour 创建了一个“顶点组”，将不不同的顶点视为单个实体。

VertexMergerGroup 类通过添加顶点列表作为其成员元素之一来扩展 Vertex。 Tinfour 第一次遇到插入顶点不唯一的情况时，它会将预先存在的顶点对象替换为用其水平构造的 VertexMergerGroup 实例

坐标。插入顶点和预先存在的顶点都被添加到组中。当应用程序需要顶点组的垂直 (z) 坐标时，Tinfour 会根据为 TIN 设置的访问选项来提取顶点的最小值、最大值或平均值。如果应用程序使用增量 TIN 类的访问器方法来请求网格中当前所有顶点的列表，则生成的顶点集合（Java 列表）包含顶点组作为元素。捆绑到组中的插入顶点不包含在结果中，但可以通过访问包含它们的组对象来获取。

* + - 1. ***型腔创建***

在接下来的过程中，当且仅当插入顶点位于位于边的相对侧的三角形的外接圆之外时，我们将边描述为“Delaunay”。该算法通过删除所有非 Delaunay 边来创建空腔。当边缘被移除时，相邻边缘的前向和反向链接被调整，使得空腔由一组正确链接的边缘界定。

型腔创建过程如下：

* + - * 1. 任意选择封闭三角形的一条边作为“起始边”。
        2. 将起始边的初始顶点指定为“起始顶点”。
        3. 将一个元素定义为“光标”边缘并将其设置为起始边缘。
        4. 如果光标边缘相对于插入顶点是 Delaunay，则不会将其删除。如果相对的顶点是鬼顶点，则不会删除该边，并且该边将被视为“有效”Delaunay。使用 InCircle 计算来确定光标边缘是否为 Delaunay。如果 InCircle 计算不明确，请将边视为 Delaunay。边缘是德劳内吗？

是：不要移除边缘。将光标转移到其自己的前边缘。

否：从网格中删除边，调整相邻边的链接以保持空腔多边形链接。将光标转移到其双轴的前边缘。

* + - * 1. 如果光标边缘的初始顶点是起始顶点，则型腔创建过程完成。否则，从步骤 4 开始重复。

封闭三角形的所有边都可能是正确的 Delaunay，并且“空腔”多边形将只是原始的封闭三角形。

* + - 1. ***链接连接***

生成的多边形可能是凸的或非凸的，但 Bowyer 和 Watson 的工作表明它将严格按逆时针顺序排序。此外，在插入顶点和多边形顶点之间构建的所有边都将是 Delaunay 最优的。链接连接过程很简单，生成的三角形将按逆时针顺序指定。此外，只要所有 InCircle 计算都明确，生成的网格将是 Delaunay 最优且唯一的。否则，它将“接近 Delaunay”且非唯一。尽管 Tinfour 可以实施附加规则来“消除歧义”，即 InCircle 计算给出不明确（零）结果的情况，但目前还没有任何规则。因此，同一组采样点可能会产生不同的 TIN，具体取决于它们添加到网格的顺序。

* + 1. 坐标和数值问题

计算几何应用因数值精度问题而臭名昭著。由于浮点运算的限制，基于具有精确代数解的表达式的计算经常会由于舍入或近似误差而失败。具体问题将在下面的讨论中出现，但有两个一般性考虑因素值得注意：

1. 几乎相同的顶点：Tinfour 中使用的三角剖分算法取决于网格中的每个顶点都是唯一的。如果折点太近，则组合它们的值的数值计算可能会导致 TIN 构建过程中出现错误。为了避免两个顶点间隔太近而导致计算失败的问题，Tinfour 必须定义一个阈值距离，用于将“几乎相同”的顶点视为同一点。
2. 需要扩展精度算术的情况：在某些情况下，Tinfour 将使用扩展精度算术来确定要素之间的几何关系（例如，顶点位于直线的哪一侧）。由于扩展算术比标准浮点计算需要更多处理，Tinfour 实现了阈值，以便当某些标准计算产生“接近零”的值时，可以采用替代扩展精度值计算。

阈值的分配取决于正在建模的数据的大小。用于对呼叫文化中营养物质分布进行建模的应用程序的坐标值与基于相距数百公里的天气观测的应用程序的坐标值有很大不同。

计算阈值时，增量 TIN 类的构造函数允许应用程序指定与要构建到 TIN 中的折点的平均间距相关的值。默认构造函数假定值为 1 个单位（米、英尺、秒差距等）。其他构造函数允许应用程序指定适当的值。

认为两个顶点相同的阈值是平均点间距的 1/10000第 。应用程序可以通过使用 Tinfour 项目中定义的 Thresholds 类来改变这一点。

* + - 1. ***顶点的大数值坐标***

尽管 Tinfour 绝不限于地理应用，但地球物理建模是不规则三角网络的主要用途，并且要求软件能够管理大幅度坐标。地球是一个很大的地方。我们的星球绕赤道一周 40081299 米。但地理应用程序经常处理一米是一个很大距离的情况。因此，在某些情况下，将坐标解析为最接近的米需要至少八位数字的精度。这样做需要使用 Java 的 8 字节双浮点数据类型，该类型提供大约 16 位十进制数字的精度。 4 字节浮点类型仅提供 7 位数字，通常足以满足垂直坐标（高程）的需要，但不足以满足水平坐标（x/y、纬度/经度等）的要求。

例如，用于测试 Tinfour 的数据集之一是在俄勒冈州 Pole Creek 附近进行的激光雷达调查。该区域包含约500万个地面点，平均间距约0.5米。水平坐标在投影坐标系中给出（x/y 值以米为单位，而不是

纬度/经度）。下面给出了前六个点的坐标。尽管水平坐标值很大，但后续值之间的差异很小。

|  |  |  |
| --- | --- | --- |
| X | Y | Z |
| 605075.31 | 4891924.60 | 1610.51 |
| 605074.19 | 4891924.54 | 1610.56 |
| 605072.07 | 4891924.84 | 1610.90 |
| 605071.97 | 4891925.62 | 1611.99 |
| 605067.98 | 4891925.29 | 1611.64 |
| 605055.42 | 4891925.26 | 1611.71 |

上面列出的 x 和 y 坐标分别需要 8 位和 9 位精度，并且都在 8 字节浮点值支持的范围内。但考虑一下当它们用于取决于这些坐标乘积的算术时会发生什么，例如众所周知的多边形面积公式中使用的算术。给定一组描述多边形的点 (𝑥1, 𝑦1), (𝑥2, 𝑦2), … , (𝑥𝑛, 𝑦𝑛)，下标 n+1 对应于点 1，多边形的面积 A 为：

𝑛

1

𝐴 = 2 ∑ 𝑥𝑖𝑦𝑖+1 − 𝑥𝑖+1𝑦𝑖

𝑖=1

x 和 y 的乘积所需的 17 位精度可能超过普通 8 字节浮点值的精度，因此低位数字会从表达式中消失。但由于激光雷达样本中的点靠得很近，因此低位数字只是我们感兴趣的数字。在获取这些术语的乘积和差异时，它们所携带的信息将会丢失。

一些 TIN 实现尝试通过标准化所有输入坐标（在构建 TIN 时将它们缩放到 0 到 1 的范围）来解决此问题。由于 Tinfour 的目标之一是保持数据元素的原始形式，因此它不会修改输入坐标。相反，它通过使用易受数值影响的表达式的替代代数形式来避免数值问题。例如，当多边形刚性地移动到另一个位置时，其面积不会改变。因此，如果我们通过减去固定偏移量来减少上面表达式中坐标的大小，我们就可以避免数字问题，而不会影响结果面积值。以下表达式显示了将多边形坐标平移到接近原点的位置的一种方法。

𝑛−1

1

𝐴 = 2 ∑(𝑥𝑖 − 𝑥1)(𝑦𝑖+1 − 𝑦1) − (𝑥𝑖+1 − 𝑥1)(𝑦𝑖 − 𝑦1)

𝑖=2

尽管面积示例相当人为，但上述 InCircle 函数是使用替代代数形式来避免有效数字丢失的一个很好的示例。不同作者

使用了不同形式的 InCircle 行列式，但 Cheng 给出的形式避免了精度问题。

有关更多详细信息，请参阅维基百科文章[失去意义。](https://en.wikipedia.org/wiki/Loss_of_significance)

* + - 1. ***地理坐标和投影坐标的特殊注意事项***

使用 TIN 进行处理时，为地球物理数据样本收集的数据存在一个特殊问题：地理坐标不是各向同性的。 X（经度）和 Y（纬度）方向给出的坐标没有一致的测量单位。通过想象经线（经线）在两极汇聚的方式，可以很容易地想象这种现象。全球各地的纬线（纬线）之间的距离是统一的，但在远离赤道的纬度上，子午线之间的距离会减小。

具有地理信息系统 (GIS) 经验的用户习惯于将数据从地理坐标转换为某些局部投影坐标系（基于地图的坐标系），以便可以使用具有一致单位的 2D 笛卡尔坐标系来处理要素各个方向的测量。许多地理计算都依赖于这种质量，包括 Tinfour 执行的 InCircle 和表面梯度计算。

由于地理坐标不是各向同性的，因此大多数激光雷达产品最初都是在投影坐标系中生产的。然而，互联网上的一些最佳数据源以地理坐标形式提供样本。例如，NOAA 数字海岸站点（NOAA，2015）上提供的所有数据在发布之前都会转换为地理坐标。 2011 年，美国农业部收集了康涅狄格州西北部利奇菲尔德县的激光雷达数据样本（NOAA，2011a）。样本由 1742 个一公里正方形组成。原始数据的坐标在通用横轴墨卡托 (UTM) 投影（18N 区）中给出。然而，在 NOAA 将数据发布到其 FTP 站点之前，他们将其转换为地理坐标系。在调查范围内，纬度1度代表距离约35.4公里，经度1度代表距离约26.4公里。 因此，该区域的数据表示明显是非各向同性的。

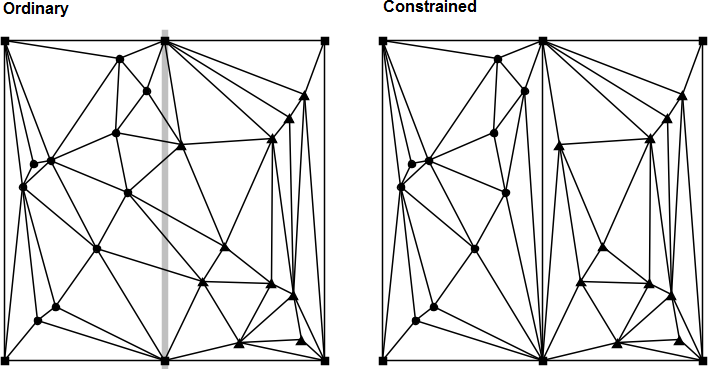
如果可能，在处理地理坐标中指定的数据时，将其转换为投影坐标系很有用。如果无法做到这一点，至少要确保在初始化 Tinfour 增量 TIN 类时为平均点间距指定一个合理的值。在上面的康涅狄格州数据中，平均点间距约为 1 米。因此，在地理坐标中，数据点之间的平均间隔约为千分之 35.4 度（如果垂直对齐）。

* + 1. 约束的 Delaunay 三角测量

上面描述的 Delaunay 技术基于以下假设：三角测量过程可以根据 Delaunay 准则自由关联相邻顶点。然而，在某些情况下，这样做并不一定是对数据的最佳处理方式。再次以高程数据为例，考虑感兴趣的陆地表面包括悬崖、道路切割、悬崖，甚至是

水体。连接此类边界相对两侧的顶点不一定是数据的最佳处理方式。

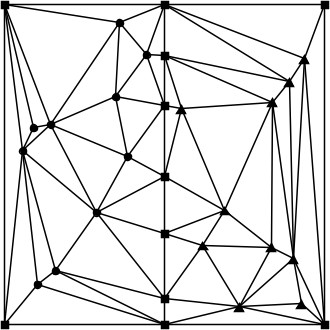
约束 Delaunay 三角剖分允许将一组边插入到三角形网格中，这些边取代了 Delaunay 准则并约束了顶点在网格中的连接方式。下图说明了这个概念。显示的数据出现在两个不同的区域中。普通的 Delaunay 随意在单独的顶点之间创建连接。受约束的 Delaunay 以定义数据区域限制的边的形式向系统添加更多信息。在图中，约束显示为三角剖分中心的垂直边缘。



**图 18 – 普通三角测量和约束 Delaunay 三角剖分**

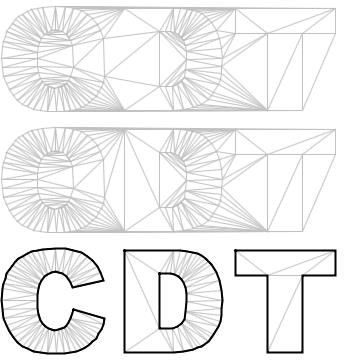
在三角剖分中添加约束的一个缺点是，网格中的所有三角形都不一定符合 Delaunay 准则。特别是，约束可能会产生“细”三角形，例如出现在图中约束边缘附近的三角形。当使用三角剖分法对值进行插值或对曲面进行建模时，通常不希望出现此类伪影。此外，许多应用程序利用了这样一个事实，即 Delaunay 三角测量很容易映射到另一个重要的图形结构，即 Voronoi 图。如果添加约束使三角剖分非 Delaunay，则它不再具有关联的 Voronoi 图。

Rognant等人（1999）描述了一种恢复Delaunay最优性的方法，他们也提供了该技术的简短数学证明。该技术沿约束边添加合成点，如下图所示。约束边被细分为更小的边，得到的三角形都符合 Delaunay 准则。最佳状态恢复。



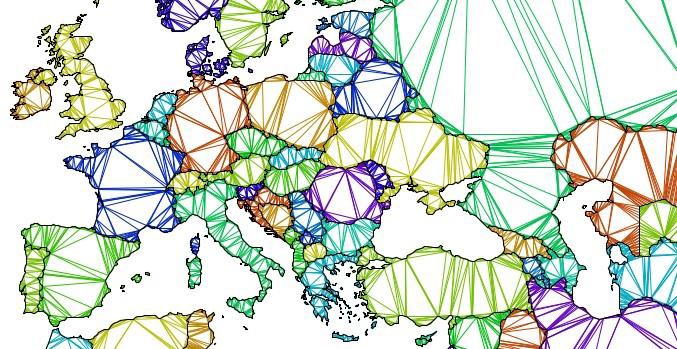
**图 19 – 恢复最优性**

CDT 的应用范围不仅限于地形建模，还延伸到数据建模的许多领域。与表面高程无关的 CDT 示例由 Tinfour 软件发行版中包含的 LogoCDT 应用程序提供。该应用程序的图像如下所示。



**图 20 – 普通的 Delaunay、约束和外部约束。**

Tinfour 实现了一个称为“约束区域”的概念，它允许基于多边形的约束来定义具有应用程序定义元数据的区域。此元数据通常以构造约束时添加到约束的 Java 对象的形式指定。例如，下图是使用来自公共领域的全球规模产品组成的[天然地球](http://www.naturalearthdata.com/)地图项目。每个国家/地区多边形都填充了一个 Java Color 对象。编写了一个测试应用程序，以使用注册到关联多边形的颜色从生成的 TIN 渲染内部边缘。



**图21 – 使用约束区域特征渲染的图像**

* 1. **插值**

插值可能是三角形网格最常见的应用。Tinfour 实现了三种不同的插值技术：三角面、自然邻居和地理加权回归多项式。由于这三种技术的实现使它们以只读方式访问 TIN，因此可以使用多线程方法并行操作任何 Tinfour 插值类的多个实例。

* + 1. Tinfour实施的技术

Tinfour实现的插值方法如下所述。

* + - 1. ***三角形小平面 （TriangularFacetInterpolator.java）***

三角形小平面技术通过使用包含该点的三角形的三个顶点来推导平面 z=f（x，y） 的方程，然后在指定坐标处求解 z 值，从而确定插值点 （x，y） 的值。如果该点位于 TIN 的边界之外，则将其投影到其下层边界边上，并使用两点插值分配一个值。

虽然这种方法快速且稳健，但它并不适合大多数应用，因为所得曲面的一阶导数在相邻三角形的边缘上不是连续的。

* + - 1. ***自然邻居 （NaturalNeighborInterpolator.java）***

Sibson 的自然邻居插值技术使用 （x，y） 邻域中顶点的加权和来计算 z 值。该技术快速、准确，并在 TIN 上提供一阶导数连续性，除了顶点（Sibson 提出了一种额外的方法，可以在任何地方提供一阶导数连续性，但尚未在 Tinfour 中实现）。自然邻域插值提供了良好的结果，是许多应用的绝佳选择。但是，它没有在 TIN 的边缘或外部定义。此外，用于计算曲面法线的 Tinfour 逻辑往往对斜率的局部变化没有反应，因此视觉结果似乎已经过滤掉了更精细的细节。

与许多自然邻居实现不同，Tinfour 算法在用于执行插值时不会改变 TIN。实际上，它以只读方式访问 TIN。因此，可以在通过 TIN 执行分析或其他操作的并发多线程应用程序中使用插值器。

* + - 1. ***地理加权回归 （GwrTinInterpolator.java）***

地理信息分析的基本思想之一是，在兴趣点附近收集的数据比从更远的地方收集的数据更相关。因此，如果我们使用回归技术在曲面上插值一个点，那么为附近的样本点分配比较远的样本点更大的权重似乎是很自然的。实际上，这就是地理加权回归 （GWR） 技术试图做到的。

GwrTinInterpolator 类在 （x，y） 附近选择一组有限的输入样本，并根据每个样本与插值点距离的倒数对每个样本应用加权因子。

尽管回归技术是 Tinfour 支持的方法中处理器最密集的方法，但它是唯一提供描述插值结果质量的统计数据的技术，包括其插值的标准差和预测区间。此外，由于此插值的结果是多项式（1、2 或 3 次），因此可用于查找 （x，y） 邻域中曲面的导数和二阶导数。此功能使回归技术在需要表面法线、斜率或曲率值的应用中非常有用。GWR 是 Tinfour 示例山体阴影实现的首选插值。

GWR 技术还具有与其他技术不同于三角网格的内在联系的特点。它只需要插值坐标附近的样本集合。因此，除了基于 TIN 的实现 GwrTinInterpolator 之外，Tinfour 还包括一个名为 GwrInterpolator 的类，该类接受一组采样点进行插值。应用程序可以使用任何适当的方法收集这些采样点，然后使用 GwrInterpolator 执行其插值。不需要 TIN。Tinfour 库还公开了一个内部类 SurfaceGWR，该类执行与 GWR 建模相关的大多数内部操作。

下面给出了有关 GWR 技术的更多详细信息。

* + 1. 交叉验证

地理空间分析的一个原则是，彼此靠近的要素往往具有相似的属性。我们预计，与相距较远的样本相比，两个靠近采集的高程样本更有可能具有相似的值。在地理信息分析中，这种想法通常被称为空间自相关。

考虑到这一原则，问题就出现了，即特定样本点的值可以被其邻居预测得有多好。交叉验证过程使用以下过程探索该问题：

1. 从一组 S 采样点构造 TIN。
2. 一次一个，暂时从 TIN 中删除一个采样点 si = （xi ， yi ， zi ），然后使用 TIN 在 （xi ， yi ） 处插入 z 的值。
3. 计算误差 ei = |z-zi |。
4. 将采样点 si 恢复回 TIN，并继续处理 TIN 中的其余样品。

误差项的平均值和方差为评估样本集的空间自相关程度和特定插值策略的相对成功率提供了一个度量。以下输出是使用 Tinfour 源代码分发中包含的示例应用程序之一 ExampleCrossValidation 获得的，用于处理 Bear Mountain 示例的一部分。

1036879顶点的测试663592

方法 均值 |err|标准开发 |err| 错误范围

|  |  |  |  |
| --- | --- | --- | --- |
| 三角形刻面 | 0.049917 | 0.047752 | -2.524 2.473 |
| 天然邻居 | 0.048697 | 0.046385 | -2.516 2.381 |
| GWR，固定带宽 0.86 | 0.048607 | 0.046385 | -2.395 3.586 |
| GWR，比例 0.45 | 0.048021 | 0.045532 | -2.447 2.318 |

本文显示了所有三个插值类的结果，其中包含地理加权回归方法的两种变体，下面将对此进行讨论。一般来说，所有产品都产生接近 5 厘米（约 1.96 英寸）的值。检查源数据表明，最严重的错误发生在具有非常陡峭或不连续表面的区域，例如悬崖或悬崖，这并不奇怪。

* + 1. 地理加权回归 （GWR） 技术

段落中介绍了地理加权回归技术[2.3.1.3](#_bookmark6) 以上。

地理加权回归 （GWR） 是一种统计技术，用于数据性质可能因地理区域而异的情况，这一特征称为“非

平稳性”。在 GWR 中，根据每个样本与感兴趣点的距离对其应用加权因子。因此，样本距离越远，它对回归估计的贡献就越少。虽然 GWR 是一种非常通用的技术，但 Tinfour 实现侧重于数据是具有各向同性坐标的实值表面（“场”）的特殊情况。这种处理方式非常适合地形分析和相关问题。可通过 GWR4 应用程序1 和 R 统计应用程序的模块在 Web 上获得更通用的实现。有关该技术的更多信息可在Wheeler和Páez（2010）的网站上获得。

Tinfour 的 GwrTinInterpolator 类提供了其他插值器不使用的可选插值参数：表面模型和带宽选择。曲面模型选项允许应用程序指定插值器用于对曲面进行建模的多项式的次数。应用程序可以指定平面模型、二次模型或三次模型。带宽选择选项允许应用程序指定不同的策略来为样本分配权重。

* + - 1. ***表面模型***

响应曲面模型选择而导出的多项式以下列形式给出，这些形式直接对应于 Java 枚举 SurfaceModel 中的状态，SurfaceModel 是 Tinfour 分析模块的一部分。插值系数 β0、β1、.、βn 是使用回归运算从输入样本中推导出来的。

平面的 𝑧1 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦

具有交叉项的平面

𝑧̂2 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦 + 𝛽3𝑥𝑦

二次 𝑧̂3 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦 + 𝛽3𝑥2 + 𝛽4𝑦2

带交叉项的二次

𝑧̂4 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦 + 𝛽3𝑥2 + 𝛽4𝑦2 + 𝛽5𝑥𝑦

（TF346型）不幸的是，在撰写本文时，我无法在网络上找到 GWR4 的副本，除非在一个想要推出广告软件的网站上。测试是使用不再可用的旧版本进行的。

四方 𝑧̂5 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦 + 𝛽3𝑥2 + 𝛽4𝑦2 + 𝛽5𝑥3 + 𝛽6𝑦3

带交叉项的三次

𝑧̂6 = 𝛽0 + 𝛽1𝑥 + 𝛽2𝑦 + 𝛽3𝑥2 + 𝛽4𝑦2 + 𝛽5𝑥𝑦 + 𝛽6𝑥2𝑦 + 𝛽7𝑥𝑦2 + 𝛽8𝑥3 + 𝛽9𝑦3

当计算坐标 （x， y） 的插值时，Tinfour 方法使用 TIN 来识别插值坐标邻域中的一组样本，选择适合曲面模型程度的多个输入样本。接下来，根据每个样本与插值点的距离为每个样本分配加权因子。最后，在执行回归之前，通过偏移量（-x，-y）调整输入样本的水平坐标。通过该调整，将插值点视为原点，并计算坐标 （0， 0） 的估计函数。因此，插值的值只是插值得到的估计值 β0 。这种方法的主要目的是在执行线性回归时减少数值问题的影响和精度损失。 但它也有一个副作用，即大大简化了插值坐标的计算，特别是在获得斜率、曲面法线或二阶导数时。

对于在坐标 （0,0） 处计算的所有 6 个模型，以下结果适用。

𝑧𝐼𝑛𝑡𝑒𝑟𝑝 = 𝛽0

𝜕𝑧

𝑧𝑥 = 𝜕𝑥 (0, 0) = 𝛽1

𝜕𝑧

𝑧𝑦 = 𝜕𝑦 (0, 0) = 𝛽2

在插值点处垂直于曲面的单位由下式给出

𝑛⃗ = (−𝑧𝑥,−𝑧𝑦,1) = (−𝛽1,−𝛽2,1)

√𝑧𝑥2+𝑧𝑦2+1

√𝛽12+𝛽22+1

插值点处斜坡处的曲面坡向定义为从 x 轴逆时针方向的最陡上升方向。计算公式为 θ = tan−1 （zy⁄zx），并根据两个输入参数的符号适当调整象限的角度。许多编程语言，包括 Java，都包含一个名为 atan2（） 的函数，该函数在计算切线时会自动进行此调整。使用由于将插值点视为原点而进行的简化，该方面由下式给出

𝜃 = 𝑎𝑡𝑎𝑛2(𝛽2, 𝛽1)

处理水文学的应用通常对最陡峭的下降方向更感兴趣，这当然会给出地表水流动的预期方向：

𝜃 = 𝑎𝑡𝑎𝑛2(−𝛽2, −𝛽1)

插值点处的斜率由梯度向量 |∇z|如

𝑠 = √𝑧𝑥2 + 𝑧𝑦2 = √𝛽12 + 𝛽22

二阶导数很容易计算二次模型和三次模型：

𝜕2𝑧

𝑧𝑥𝑥 = 𝜕𝑥2 (0,0) = 2𝛽3

𝜕2𝑧

𝑧𝑥𝑥 = 𝜕𝑦2 (0,0) = 2𝛽4

对于具有交叉项的二次模型和三次模型：

𝜕2𝑧

𝑧𝑥𝑦 = 𝜕𝑥𝜕𝑦 (0,0) = 𝛽5

Wilson 和 Gallant（2000 年，第 53 页）给出了以下剖面曲率的表达式，剖面曲率是沿指向下降方向的单位矢量测量的斜率变化率。

𝑧𝑥𝑥𝑧𝑥2 + 2𝑧𝑥𝑦𝑧𝑥𝑧𝑦 + 𝑧𝑦𝑦𝑧𝑦2

𝐾𝑝 = −

(𝑧 2 + 𝑧 2)(𝑧 2 + 𝑧 2 + 1 3⁄2

𝑥 ) 𝑥 𝑦

剖面曲率 Kp 和下面描述的其他曲率的值以每单位距离的弧度为单位。

根据 Peckham （2011， p 27） 的说法，分母中的额外因子 （zx2+ zy2+ 1）3⁄2 定义曲率“基于沿 3D 流线曲线（位于表面上）的微分运动”，而不是在水平面上。Wilson 和 Gallant 还为

等高线曲率（有时称为平面曲率），它描述了曲面上感兴趣点处等高线的曲率。等值曲率本质上描述了表面上的流道收敛（负）或发散（正）的速率：

𝑧𝑥𝑥𝑧𝑦2 − 2𝑧𝑥𝑦𝑧𝑥𝑧𝑦 + 𝑧𝑦𝑦𝑧𝑥2

𝐾𝑐 = −

(𝑧𝑥

2 + 𝑧𝑦2)

通过再次将额外系数添加到分母中，Peckham 获得了切向曲率，该曲率经过调整以匹配沿流线曲线而不是水平面的曲率。

𝑧𝑥𝑥𝑧𝑦2 − 2𝑧𝑥𝑦𝑧𝑥𝑧𝑦 + 𝑧𝑦𝑦𝑧𝑥2

𝐾𝑡 = −

(𝑧 2 + 𝑧 2)(𝑧 2 + 𝑧 2 + 1 3⁄2

𝑥 ) 𝑥 𝑦

Peckham 给出了流线曲率，它描述了流动方向的变化率（在水平面上测量）：

𝐾𝑠 = −

𝑧𝑥𝑧𝑦(𝑧𝑥𝑥 − 𝑧𝑦𝑦) + (𝑧𝑦2 − 𝑧𝑥2)𝑧𝑥𝑦

(𝑧 2 + 𝑧 2 3⁄2

𝑥 )

Peckham 对上面给出的所有表达式进行了清晰简洁的推导，并提供了将它们应用于表面流动和相关现象建模的有用见解。

最后，值得注意的是，上面给出的表达式可用于插值点以外的点，前提是将适当的偏移量添加到感兴趣的坐标中。虽然在计算 zx ， zy 等时不能使用上述简化，但代数很简单。

* + - 1. ***样本加权的带宽策略***

执行线性回归所需的最小样本数取决于目标多项式中的系数数。Tinfour 根据指定的模型自动从插值坐标的邻域中选择一组采样点。因此，在执行插值时，大阶多项式具有两个特征：它们需要更多的采样点，并且采样点与插值坐标的平均距离将增加，从而导致收集样本的总面积增加。由于地理空间分析的一个想法是附近的点比更远的点更相关，因此需要一些方法来解释扩展的数据集特征。

在地理加权回归中，相邻样本点的相对贡献通过反距离加权函数进行调整。Wheeler（2010 年，第 463 页）和其他作者已经确定了距离加权的几个函数。Tinfour使用高斯核

1 𝑑𝑖 2

𝑊𝑖 = 𝑒

−2( 𝛾 )

其中 di 给出第 i 个样本与插值坐标的距离，Wi 给出回归计算中用于该样本的加权因子。参数γ

上面的等式称为加权函数的“带宽”。它应该用与距离相同的测量单位表示。带宽的概念大致类似于通信理论中该术语的使用，因为它控制在执行回归时从远距离样本中接受多少信息。如果γ很大，则会减小距离项的大小并增加样本的总重量。因此，带宽值越大，距离越远的样本获得更大的权重，对整体插值的贡献就越大。对于足够大的带宽，距离项实际上为零，并且所有样本的权重接近于统一。在这种情况下，GWR 将等同于普通的最小二乘估计。如果带宽参数较小，则距离项变得更显著，并且更远距离样本的贡献会降低。

Tinfour提供了四种带宽选择方法，如下所述：

* + - * 1. 应用程序提供的固定带宽。
        2. 比例带宽选择为基于应用程序提供的参数的插值坐标中本地所选样本的平均距离的倍数。
        3. 自动选择带宽，其中根据赤池信息准则选择最佳带宽，并针对小样本量进行校正（AICc方法）。
        4. 普通最小二乘法（所有样本的均匀权重）。

如果应用程序具有特定于数据的信息或要求，并且需要强制执行特定条件，则首选固定带宽方法。它的缺点是应用程序必须实现自己的逻辑来建立带宽参数。此参数的派生没有固定的规则。对于上面关于交叉验证的讨论中所示的 Bear Mountain 样本，ExampleCrossValidation 应用程序仅使用整个样本集的平均点间距。

如果应用程序没有关于数据集的先验信息，并且希望应用与本地所选样本与插值坐标的平均距离成正比的带宽，则首选比例带宽。该方法的优点是可以调整带宽以满足样品的局部密度。

固定带宽和比例带宽方法的缺点是是任意设置，不是由数据的统计模型驱动的。相比之下，自动带宽方法基于公认的模型选择统计技术（带宽被视为模型的一个元素）。AICc 准则提供了一种判断特定样本集的表面模型和带宽值不同组合的相对性能的方法。在传统的回归应用中，该标准通常用于确定插值变量的哪个组合或子集导致响应曲面的最佳质量估计器。在自动选择过程中，Tinfour 使用标准来选择曲面模型选择和带宽组合，以在插值点附近生成良好的曲面表示。

不幸的是，AIC 分数的计算在计算上很昂贵。它需要比回归本身更多的处理。因此，自动模型和带宽选择的整个过程相当缓慢。通常每秒 100 到 200 次插值的处理速率。将 AICc 评分视为给定样本集带宽的函数通常会产生具有许多局部极值和拐点的高度复杂的曲线，这一事实加剧了这个问题。因此，像牛顿方法这样的简单数值技术不能用于找到最佳值。相反，必须采用处理器密集型搜索技术。

* + - 1. ***解释结果***

以下结果由交叉验证应用程序在启用自动模型和带宽选项的情况下执行。由于自动方法非常慢，因此只处理了激光雷达样本的一个子区域。对于此区域，总体误差计数比

上面的例子。以下详细信息还提供了所选带宽值的平均值和标准偏差，以及选择特定表面模型的次数。

已测试 9453 个顶点1036879（153.10/秒）

方法 均值 |err|标准开发 |err| 错误范围 总和错误

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 三角形刻面 | 0.052792 | 0.047142 | -0.334 0.533 | 2.242 |
| 天然邻居 | 0.051293 | 0.045680 | -0.342 0.414 | -3.003 |
| GWR，固定带宽 0.86 | 0.051572 | 0.045812 | -0.340 0.403 | 0.748 |
| GWR，比例 0.45 | 0.051091 | 0.045509 | -0.336 0.405 | -0.345 |
| GWR，自动 BW AICc | 0.052512 | 0.047208 | -0.346 0.377 | -1.037 |

自动选择的带宽值

意味 着： 0.728962 （0.400464 的平均值）

标准开发 0.133318 (0.071733)

最小值，最大值： 0.370358, 1.217216 （0.300， 0.650 平均值）

普通最小二乘法数目： 0

|  |  |
| --- | --- |
| 平面的 | 10 |
| PlanarWithCrossTerms（平面与交叉项） | 81 |
| 二次 | 1490 |
| QuadraticWithCrossTerms（二次与交叉项） | 814 |
| 四方 | 1360 |
| CubicWithCrossTerms（立方与交叉项） | 5698 |

考虑到自动带宽选择方法需要多少处理，它产生的所有回归方法中最大的平均绝对误差这一事实令人失望。人们可以合理地询问实现是否正确。虽然这仍然是一个悬而未决的问题（见[*GWR 实施状况*](#_bookmark7)下面），还有另一种解释。GWR 误差稍高的原因之一是 AICc 准则旨在选择为结果生成最佳整体插值多项式的配置。插值点的高程仅基于使用结果中的一个系数：β0。AICc 过程试图找到一组最优插值系数：β0、β1、.、βn。因此，产生强β0但高阶系数较弱的带宽和模型选择将具有相应的较差的AICc分数，并且不会比产生良好系数的选择更受欢迎。因此，如果应用程序对从表面导数（包括斜率、曲率和表面法线）得出的特征的估计同样感兴趣，则较弱的交叉验证结果将不太受关注。

此外，值得注意的是，GWR 在交叉验证技术中的成功是衡量样本点的值与其相邻点的值的预测程度。反过来，这又直接反映了预测区域中的地形类型。熊山样本表示崎岖的地形，有许多巨石、悬崖和陡峭的斜坡。来自康涅狄格州调查区域（20111218\_18TXM2835.las，主要是农田）的不同图块的更温和的地形产生了完全不同的结果。这些如下表所示。虽然自动带宽选择仍然具有最大的幅度误差，但偏差仅为 2.3 厘米小于两根手指宽度的值。这样的结果让人对激光雷达技术变得多么出色有了更深的了解。

测试了 12675 个顶点1193868（148.57/秒）

方法 均值 |err|标准开发 |err| 错误范围 总和错误

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 三角形刻面 | 0.022933 | 0.025198 | -0.281 0.265 | -0.083 |
| 天然邻居 | 0.022473 | 0.024584 | -0.320 0.249 | -2.920 |
| GWR，固定带宽，0.80 | 0.022708 | 0.024311 | -0.303 0.270 | -1.236 |
| GWR，比例 0.45 | 0.022473 | 0.024254 | -0.303 0.266 | -0.208 |
| GWR，自动 BW AICc | 0.023215 | 0.025326 | -0.343 0.312 | 1.819 |

* + - 1. ***应用程序访问 GWR 结果***

以下代码片段显示了如何获取坐标 （x，y） 处的插值结果并将其输出到 Java PrintStream （ps）。

在给定有效 TIN 的情况下对坐标 （x，y） 执行插值

使用指定的曲面模型和自动带宽选择 GwrTinInterpolator gwr = new GwrTinInterpolator（tin）;

double z = gwr.interpolate( SurfaceModel.Cubic,

BandwidthSelectionMethod.AutomaticBandwidth, 0, x, y, null);

获取最近插值的相关结果， double[] beta = gwr.getCoefficients（）;

double []predictionInterval = gwr.getPredictionInterval（0.05）;double zX = 贝塔[1];

双 zY = β[2];双 zXX = 2\*beta[3];double zYY = 2\*beta[4]；double zXY = β[4];

双方位角 = Math.atan2（zY， zX）;

双罗盘 = Math.toDegrees（Math.atan2（zX， zY））;如果（指南针<0）{

指南针+=360;

}

双年级 = Math.sqrt（zX\*zX+zY\*zY）;

double slope = Math.toDegrees(Math.atan(grade)); double kP = (zXX\*zX\*zX+2\*zXY\*zX\*zY + zYY\*zY\*zY) /

((zX\*zX+zY\*zY)\*Math.pow(zX\*zX+zY\*zY+1.0, 1.5));

ps.format（“估计 z： %12.5f\n“， z）;ps.format（“预测区间（95%% 置信度）：%12.5f 到 %6.5f （%f）\n”，

predictionInterval[0]、predictionInterval[1]、predictionInterval[1]-predictionInterval[0]）;

ps.format("Zx: %12.5f\n", beta[1]);

ps.format("Zy: %12.5f\n", 测试版[2]);

ps.format("方位角最陡上升 %12.5f\n", 方位角); ps.format("罗盘方位最陡上升 %05.1f°\n", 指南针); ps.format("成绩 %8.1f%%\n", 等级\*100);

ps.format("坡度: %8.1f°\n", 斜率);

ps.format("轮廓曲率: %12.5f\n", kP);

以下文本给出了上述代码在 Bear Mountain 示例（仅限地面点）的坐标 （627520， 4655800） 处的示例输出：

|  |  |
| --- | --- |
| 估计 z： | 612.54953 |
| 预测区间（95% 置信度）： | 612.37857 到 612.72049 （0.341922） |
| XXX公司 | -0.01685 |
| Zy: | 0.03897 |
| 方位角最陡峭的上升 | 1.97891 |
| 指南针方位最陡峭的上升 | 336.6° |
| 年级 | 4.2% |
| 坡： | 2.4° |
| 轮廓曲率： | 0.00833 |

* + - 1. ***GWR 实施状况***

GWR 的实施是 Tinfour 项目最能从专家关注中受益的领域。显然，自动带宽和模型选择选项对于批量生成插值来说太慢了，就像在生成高程或山体阴影格网时一样。在开发过程中，将 Tinfour 应用程序的结果与 GWR4 的结果进行比较，以验证实现的正确性。当 Tinfour 与应用指定的带宽一起使用时，插值和评估统计（方差、置信区间、AICc 分数）的结果会匹配。

但是，使用自动带宽选择时，用于测试的 GWR4 版本 （4.0.80） 无法成功执行，并且未报告任何有用的诊断信息。因此，自动带宽测试方法的结果尚未与其他应用进行比较。相反，测试是通过以下自检过程进行的：

* + - * 1. 对于选定的样本点集，使用“固定带宽”选项对 Tinfour 支持的范围内的大量带宽（样本与插值点的平均距离的 0.3 到 1.0 倍）执行回归。记录每个 AICc 分数。AICc得分最高的测试被认为是“最佳”带宽。
        2. 将从上述测试中获得的最佳带宽与在Tinfour实现中使用自动选择方法估计的带宽进行比较。
        3. 如果带宽选择在彼此的 5% 以内（计算方法是两个带宽之差除以评估范围），则认为测试成功。

由于Tinfour AICc计算是使用单独的软件包（GWR4）进行验证的，因此自检程序具有一定的优点。然而，由于GWR的AICc计算非常耗时，因此只考虑了几百个样本。因此，应谨慎看待这些结果。除了实施的细节之外，GWR的基本假设也值得关注。GWR 通过将远离兴趣点的样本的重要性视为小于附近样本的重要性来解决观测数据中的异方差性问题。虽然这个假设就目前而言是合理的，但它没有考虑样本之间的相互关联程度。考虑以下情况：两个相对相距较远的高程样本彼此靠近，因此捕获的要素（例如大巨石）与周围地形相比，高程有所增加。 虽然距离加权逻辑将减少每个样本的贡献，但数据集中仍有两个样本。

简单的 GWR 没有考虑到这样一个事实，即这两个样本加在一起过度代表了异方差特征。正如这个例子所显示的，解决样本之间的共线性和自相关是Tinfour实现的一个潜在改进领域。

* + - 1. ***GWR技术的背景***

关于GWR技术的一般性讨论很容易在互联网上找到（参见Wheeler 2010等）。

用于推导回归系数的计算改编自 Walpole 和 Myers（1989 年，第 401-442 页）第 10 章“多元线性回归”。Walpole 和 Myers 对多元线性回归问题、其评估统计量（特别是预测区间）及其使用进行了很好的介绍。加权回归的计算没有包含在他们的工作中，而是根据他们提供的信息得出的。由于这些计算不是来自已发表的文献，因此未经专家统计学家审查。

特定于加权回归的评估统计数据的详细信息摘自 Leung 等人（2000 年，第 9-32 页）。作者对提出无偏统计的数学合理公式和检测样本集中“非平稳性”（异方差性）的可靠方法特别感兴趣。

与应用于GWR的AICc标准相关的信息可在Charlton，Martin和Fotheringham，A（2009）中找到，这是从网上下载的白皮书。Brunsdon、Fotheringham 和 Charlton 的其他一些论文对同一材料提供了略有不同的观点，可以在网络上找到。

1. **测试和演示**

Tinfour 软件发行版包括一个名为“demo”的 Java 包（文件夹），其中包含演示如何使用该包的高级测试程序和示例应用程序。这些应用程序中的大多数都实现了从输入“顶点文件”构建 Delaunay 三角剖分的功能。顶点文件可以作为文本文件（提供 x、y 和 z 坐标的制表符、空格或逗号分隔文件）或行业标准激光雷达 LAS 文件的形式提供。一些应用程序测量软件性能或测试实现的正确性。其他人则演示了使用 Tinfour 对一组非结构化采样点进行分析。

下面描述的测试应用程序大多是小型的、非交互式的实现，用于演示或演示 Tinfour 项目的单个关键功能。但本节中的最终应用是完全不同的事情。Tinfour Viewer 是一个基于用户界面的应用程序，它提供图形显示和交互式功能，允许用户探索由非结构化数据构建的表面。从某种意义上说，这是Tinfour开发项目的高潮（至少到目前为止）。因此，这是本节讨论的最后一项。

* 1. **测试环境**

测试和示例应用程序都旨在供软件开发人员使用。当前所有测试应用程序都可以从集成开发环境 （IDE）（如 Netbeans 和 Eclipse）或命令行环境运行。运行测试的规范是从传递到 Java main 的参数中接受的。

* + 1. 命令行参数

Tinfour测试和演示应用的一般输入选项如下所述。如果应用程序具有唯一的命令行选项，则这些选项将在特定应用程序的使用文本或文档中给出。选项不区分大小写。布尔设置不采用参数，但根据需要以“-option”或“-noOption”的形式给出。

**表 6 – 输入和输出选项**

|  |  |  |
| --- | --- | --- |
| **选择** | **参数** | **描述** |
| -在 | 文件路径 | 要用作示例输入源的文件。可能是 .LAS（激光雷达）、.CSV（逗号分隔值）或字符分隔的文本文件（默认为空格或制表符）。当空格用作  分隔符，Tinfour 将多个空格视为单个元素。 |
| -定界符 | 字符 | 用于基于文本的输入文件的分隔符，如果  使用了空格以外的其他内容。 |
| -外 | 文件路径 | 生成文件输出的应用程序的输出路径规范。如果应用程序生成多个输出（例如  如果同时生成栅格高程格网文件和图像文件，则文件扩展名将根据需要替换。 |

**表7 – 处理选项**

|  |  |  |
| --- | --- | --- |
| **选择** | **参数** | **描述** |
| -细胞空间 | 浮点距离 | 对于生成输出网格的应用程序，网格单元之间的间距。间距应为在与源数据相同的单位制中指定的距离。生成网格的应用程序能够自动选择单元格间距，但通常首选显式设置  由大多数用户。 |
| -框架 | xMin xMax yMin Ymax | 对于可以专注于整个样品集的子区域的应用，即要处理的感兴趣区域。默认情况下，“帧”的坐标是输入数据集的整体边界。因此，除非指定了帧，否则将考虑整个数据集。frame 选项通常不会影响从源文件中读入哪些样本。相反，它提供  与样品处理方式相关的选项。 |
| -插值器 | 细绳 | 对于执行插值的应用程序，指示要使用的插值方法的字符串：   * NaturalNeighbor（默认） * 三角面 * 回归 |
| -激光雷达类 | 整数：0 到 255（对于特定分类）或 -1  来处理所有。 | 对于处理激光雷达数据的文件，要包含进行处理的激光雷达点的分类。山体阴影和高程相关的应用程序将默认值为 2（对于地面  点）。 |
| -激光雷达细化 | 浮点值大于零且小于  或等于一。 | 对于输入文件，表示要随机选择点的子集。通常在输入文件对于系统上的可用内存来说太大时使用。该选项的名称  是一个用词不当，因为它适用于所有输入格式。 |
| -n测试 | 整数：1 到最大整数  价值。 | 对于执行重复测试的应用程序，要执行的迭代次数。大于 3 的值是  受到推崇的。 |
| -调色板 | 命名调色板 | 提供 TIN​​ 颜色渲染的应用程序的调色板。调色板内置于测试应用程序中并使用以下名称：   * 蓝变黄 * 紫色调 * 红到黄到白 * 彩虹 * 黑到灰 |
| -预分配， | 无（布尔值） | 指示是否要在之前预先分配边缘的存储 |

|  |  |  |
| --- | --- | --- |
| -无预分配 |  | 构建 TIN。用于隔离对象的运行时成本  分配版本算法相关处理。 |
| -预排序，  -无预排序 | 非（布尔值） | 指示数据是否应使用  构建 TIN 之前的希尔伯特排序方法。 |
| -种子 | 整数 | 对于使用随机化方法来选择或生成输入样本点集的所有应用程序，该过程的随机种子。种子遵循总合同  java.util.Random 类。 |
| -锡类 | 给出完全限定的 Java 类名的字符串。 | 用于构建 TIN 的类。目前支持两种规格：  org.tinfour.standard.IncrementalTin org.tinfour.semivirtual.SemiVirtualIncrementalTin  规范遵循 Java 约定，并且是案例-  敏感的。作为快捷方式，您还可以使用“标准”和“半虚拟”作为参数。 |
| -A级  -B级 | 给出完全限定的 Java 类的字符串  姓名。 | 指定要为 TwinBuildTest 测试的类 |

一些测试实用程序可能会创建一组随机的非结构化输入样本点。少数使用网格，其中水平坐标受到小的随机调整的干扰，以测试 TIN 构建实施中的弱点。除了上面描述的 –seed 选项之外，此类应用程序还可以使用下面列出的选项：

**表 8 – 随机样本生成选项**

|  |  |  |
| --- | --- | --- |
| **选择** | **争论** | **描述** |
| -n列 | 整数：2到最大整数  尺寸 | 网格中的列数 |
| -n行 | 整数：2到最大整数  尺寸 | 网格中的行数 |
| -n顶点 | 整数：3 至  最大整数大小 | TIN 中包含的折点数量。 |

交叉验证测试应用程序实现了其他测试未使用的两个独特选项。这些选项与自动带宽选择的使用有关。此时，自动选择是处理器密集型的并且需要很长的运行时间。因此，ExampleCrossValidation 应用程序提供了一个选项来控制是否启用自动带宽选择。它还提供了一个选项，可以在应用程序执行测试时打印进度指示器（以及预计完成时间）。

**表 9——交叉验证应用的选项**

|  |  |  |
| --- | --- | --- |
| **选择** | **争论** | **描述** |
| -自动黑白，  -无自动BW | 无（布尔值）。  默认为 false。 | 启用带宽的自动计算 |
| -显示进度，  -noShow进度 | 无（布尔值）。  默认 false | 允许在测试期间打印期间进度报告  执行。 |

* 1. **应用示例**
     1. 顶点文件中的高程和山体阴影网格示例

Java 应用程序 ExampleGridAndHillshade 可以接受来自文本文件或 LAS 激光雷达文件的顶点，并生成高程网格和山体阴影图像，如[图22](#_bookmark8)。该图显示了从覆盖上述熊山地区附近稍大区域的激光雷达样本获得的山体阴影图像。该地区几乎全是树木繁茂，没有铺砌的道路。穿过图像的线性特征是一条未经改善的消防道路。这条路是通往熊山山顶的环形步道的一部分，也是通往位于图像稍东南部的阿巴拉契亚步道的通道。图像右下角隐约可见一条向北通向山顶的步道。山顶上有一个大石堆，清晰可见，如图右上角的“凸起”。

下图和上图 4 中的山体阴影图像是使用名为 ExampleGridFromFile 的示例应用程序生成的。演示程序执行以下操作：

* + - 1. 接受指定输入激光雷达 LAS 文件或文本文件的命令行参数。
      2. 接受命令行参数，指定扩展名为 .asc 的 Esri ASCII 栅格文件的输出文件路径。
      3. 从 LAS 文件中读取一组顶点（样本）。
      4. 根据估计的内存使用情况确定三角剖分类别，并选择标准或半虚拟变体来构建 Delaunay 三角剖分。
      5. 插入高程值网格并以 Esri ASCII 栅格格式存储信息，如下所示：
         1. 网格是根据输入数据的范围和指定为命令行参数的像元大小来计算的（如果省略，则使用样本的总体估计点间距）。在选择像元大小时，选择与指定数据一致的值。
         2. 使用可以在命令行上指定的三种插值方法之一来插值高程：

线性（简单三角面）

自然邻点插值（默认）

线性回归

* + - 1. 使用以下命令生成山体阴影图像并将其写入 PNG 文件：
         1. 创建一个 Java BufferedImage 实例，其大小根据上面计算的输出网格中的行数和列数确定。
         2. 如果命令行参数包含调色板规范，请使用它来对高程进行颜色编码。如果没有，则使用纯白色背景。[图22](#_bookmark8)是在不使用调色板的情况下生成的。图 4 使用“彩虹”调色板。
         3. 使用 Tinfour 的线性回归插值器创建描述每个网格点附近表面的三次方程。
         4. 求三次方程的偏导数，计算 X 轴和 Y 轴方向的斜率。从这些斜率中，获得每个网格点处垂直于表面的矢量。
         5. 使用简单的漫射闪电模型并假设点照明源，计算相对阴影。
         6. 根据光照值调整输出像素的亮度，并以PNG文件的形式输出图像。



**图 22 – 来自 Bear Mountain 激光雷达样本的山体阴影图像**

* + 1. 使用希尔伯特排序进行点细化

图 2 和图 3 均显示了使用 Bear Mountain 数据集的子样本创建的三角形网格。从样本列表中获取子集的最直接方法是循环遍历列表，以固定间隔选择样本。要将一组 1000 个样本减少到 100 个，只需选择每 10 个样本，依此类推。

这种简单方法的一个缺点是它不能保证完全覆盖采样点定义的区域。当样本以随机或半随机顺序且密度不均匀时，这种情况会加剧。

Tinfour 包括一个名为 ExampleWireframeWithThinning 的实验应用程序，用于生成图 2 和图 3 中的细化网格。该应用程序尝试通过在选择样本之前对数据执行希尔伯特排序来提高子样本的覆盖范围。在测试中，这种方法总是能取得良好的效果。然而，它代表着对简单选择的显着改进的想法仍然是推测的。

* + 1. 曲面插值的多个并发进程

Tinfour 使用不受益于并发处理的顺序过程构建 TIN。但是，由于插值和网格分析例程以只读方式在 TIN 上运行，因此它们适合使用 Java 的多线程功能进行并行处理。应用程序ExampleMultiThreadTest演示了如何通过使用多个并发线程处理数据来减少相对昂贵的表面插值过程所需的时间。

* 1. **测试应用**
     1. 实施正确性的单一构建测试

测试应用程序 SingleBuildTest 使用源文件的输入构建 TIN，并执行构建后测试以确保实现的正确性。它还打印有关内存使用的诊断信息和有关 TIN 构建过程的统计信息。

在构建结束时，SingleBuildTest 会执行“完整性测试”，以验证生成的 TIN 是否符合 Delaunay 标准以及所有数据元素是否已正确填充。实际上，这是实施正确性测试。

下面给出了示例输出。

TIN 类别： org.tinfour.semivirtual.SemiVirtualIncrementalTin 输入文件： C:\CT\_NW\_Lidar\A1\_Bear\bear\_all.las

文件中的顶点数（所有类）：4874727 要处理的顶点数：1036887

范围 x 值： 627000.000, 628000.010, (1000.010000)

y 值范围： 4654999.980, 4655999.990, (1000.010000)

z 值范围： 538.434, 709.032, (170.597961)

美东时间。样本间距： 0.862

预排序时间： 460.14

|  |  |
| --- | --- |
| 不使用预分配  开始插入 |  |
| 构建 TIN 的时间： | 2961.20 |
| TIN 的总时间：检查内存 | 2961.20 |
| 内存使用（字节/顶点） 所有对象： | 120.48 |
| 仅顶点： | 44.08 |
| 边缘和其他元素： | 76.40 |
| 申请总数（mb）： | 316.00 |

|  |  |  |  |
| --- | --- | --- | --- |
| 描述性数据 |  |  | |
| 插入的顶点数： | 1036887 |
| 重合顶点间距： | 0.000010 |
| 套： | 8 |
| 总数： | 16 |
| 周长上的边数： | 37 |
| 普通边数： | 3110597 |
| 幽灵边缘数量： | 37 |
| 编号边缘替换： | 5096883 | （平均： | 4.9) |
| Max Edge 由添加操作替换： | 70 |  | |
| 平均点间距： | 1.12 |  | |
| 应用的标称间距： | 1.00 |  | |
| 三角形数量： | 2073719 |  | |
| 三角形的平均面积： | 0.482 |  | |

桑普。区域标准差： 0.310

最小面积： 0.000300

最大面积： 14.024

总面积： 999993.4

|  |  |
| --- | --- |
| 施工统计 |  |
| SLW 步行次数： | 1036887 |
| 外部阶段： | 6947 |
| 测试： | 6364282 |
| 扩展： | 1124 |
| 平均完成步骤： | 3.12 |
| 内圆计算： | 13273748 |
| 扩展： | 14 |
| 冲突： | 0 |

|  |  |
| --- | --- |
| 边缘池诊断 |  |
| 分配的边： | 3110634 |
| 自由边缘： | 278 |
| 页数： | 3038 |
| 部分使用的页面： | 1 |
| 总分配操作： | 3173435 |
| 免费操作总数 | 62801 |

执行完整性检查 测试完成

* + 1. 性能评估的重复构建测试

重复构建提供了一种评估 Tinfour 实施性能的方法。为此，它会重复运行示例数据，测量构建所需的时间。在下面的示例输出文本中，请注意前两个测试的处理时间比后续测试的处理时间稍长。这种影响是由于 Java 类加载器和 JIT 编译器在执行初始阶段的成本造成的。经验表明，某些系统需要多达三次迭代才能使运行时间稳定到稳定状态。因此，前三个测试运行不包含在总体平均值中。

测试日期： 2016 年 2 月 24 日 02:09 世界标准时间

输入文件： C:\CT\_NW\_Lidar\A1\_Bear\bear\_partial.las TIN 类： org.tinfour.tin.IncrementalTin

预分类时间 0.0 顶点数 287889

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 跑步， | 建造， | 平均构建， | 总内存， | 分配时间 |
| 0, | 600.949, | 0.000, | 68.819, | 1314.329 |
| 1, | 475.069, | 0.000, | 68.825, | 205.250 |
| 2, | 415.067, | 0.000, | 68.825, | 130.842 |
| 3, | 428.102, | 428.102, | 68.825, | 122.653 |
| 4, | 424.691, | 426.397, | 68.825, | 129.509 |
| 5, | 406.460, | 419.751, | 68.825, | 77.295 |
| 6, | 403.428, | 415.670, | 68.825, | 61.259 |
| 7, | 413.498, | 415.236, | 68.825, | 119.233 |

移除平均最大值： 412.019

* + 1. 用于调整性​​能和优化的孪生构建测试

优化和性能调整可能是棘手的话题。有时，最有希望的概念最终却令人大失所望。看似“理所当然”的想法最终证明是严重的误解。在单元测试中成功的逻辑在集成到代码库中时实际上会减慢处理速度。

Tinfour 的软件开发工作包括使用名为 TwinBuildTest 的应用程序提高处理速度的不同想法和方法的大量实验。 TwinBuildTest 旨在克服现代操作系统是一个嘈杂的测试环境这一事实。很难控制甚至预测后台进程、安全软件或其他系统进程何时会与测试程序竞争资源。为了客观地衡量代码更改如何影响性能，开发过程依赖于“孪生构建测试”。

* + - 1. 实现同一网格构建类（或多个类）的两个版本，引入要在其中之一中进行测试的软件方法。
      2. 在同一个 Java 进程中重复运行两个版本，交替使用方法。记录每个的运行时间。
      3. 检查运行时间并观察趋势。比较每个处理所需的相对时间。如果统计数据包含明显的异常，则丢弃结果并再次运行测试。

以下文本显示了一项测试的输出，该测试比较了在资源和 CPU 能力有限的计算机上运行的包含约 29 万个点的激光雷达样本构建 TIN 的时间。在内部，Tinfour 使用缓冲区，允许在构建 TIN 时重用边。该测试评估了缓冲区是否真正有助于性能。 TwinBuildTest 用于将标准实现（A 类）与禁用缓冲区的版本（B 类）进行比较。

运行时间以毫秒为单位。与重复构建测试一样，前三个运行时间不包含在表格结果中。

孪生构建测试

日期： 2015年12月27日 16:57

A 类：org.tinfour.standard.IncrementalTin B 类：org.tinfour.standard.IncrementalTin1 预分配启用：true

示例数据：C:\CT\_NW\_Lidar\A1\_Bear\bear\_partial.las 顶点数 287889

|  |  |  |  |
| --- | --- | --- | --- |
| 世界标准时间 2015 年 12 月 27 日 16:57 |  | | |
| 跑步， 构建1， | 平均构建1， | 构建2， | 平均构建2 |
| 0, 3160.465, | 0.000, | 2308.458, | 0.000 |
| 1, 1607.773, | 0.000, | 1599.363, | 0.000 |
| 2, 747.325, | 0.000, | 805.106, | 0.000 |
| 3, 749.661, | 749.661, | 804.241, | 804.241 |
| 4, 764.466, | 757.064, | 812.882, | 808.562 |
| 5, 744.030, | 752.719, | 811.557, | 809.560 |
| 6, 749.223, | 751.845, | 815.042, | 810.931 |
| 7, 776.388, | 756.754, | 805.807, | 809.906 |
| 删除最大值后的平均值 | 751.845, |  | 808.622 |

比较时间法 a/b：0.9297858109283054 比较时间法 b/a：1.0755165203065342

标准实现的平均运行时间约为 750 毫秒，而修改后的实现的平均运行时间约为 800 毫秒。因此，带缓冲区的版本比不带缓冲区的版本运行速度快 7% 左右。结果证明缓冲区是有效的性能增强，应该保留在实现中。

向下查看值列，请注意两种方法的运行时间相当一致，表明测试期间系统环境相当稳定。有时，列中的条目会以明显随机的方式来回跳动，这表明与测试应用程序竞争资源的系统上可能发生了某些情况。在测试中，数据通常可能只包含一次运行，该运行比所有其他运行都要长得多。这种情况经常发生，以至于双构建测试应用程序在统计统计数据时自动忽略主系列的最差运行。但是，在运行时间明显不一致的情况下，最好简单地丢弃结果并再次运行测试。

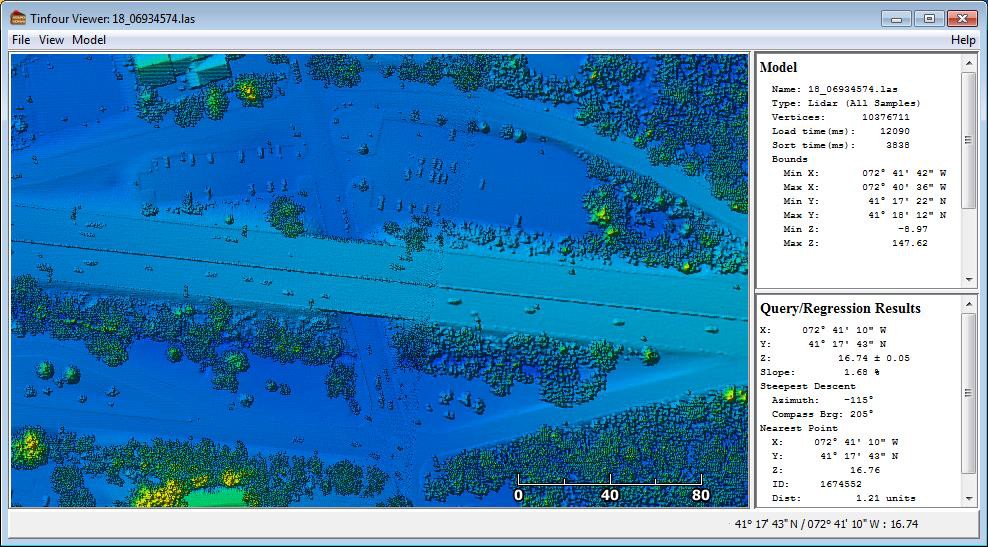
Twin Build Test 应用程序接受两个特殊的命令行参数，-classA 和 -classB，它们给出了要测试的类的完全限定名称。默认情况下，它使用标准 IncrementalTin 和半虚拟 SemiVirtualIncrementalTin。

* + 1. 由于样本大小，处理 TIN 的时间

TimeDueToSampleSize 测试执行重复的构建操作，其中它从样本集中随机选择输入数据的子集并测量构建它的时间。由于每个随机选择的大小各不相同，因此该测试应用程序提供了一种现实的方法来测量处理 TIN 的实际成本（作为输入折点数量的函数）。

* 1. **Tinfour 查看器**

当第一次考虑像 Tinfour 这样的软件包时，提出诸如“它有什么用？”之类的问题是合理的。以及“我能用它做什么？”这些注释至少部分是为了解决这些问题。因此，他们以一个讨论应用程序结束似乎很自然，该应用程序将 Tinfour 的主要功能集成到一个展示其功能的应用程序中。 Tinfour 查看器使用 Delaunay 三角剖分作为查看和分析由非结构化数据构建的曲面的工具。图[以下](#_bookmark9)显示了 Tinfour Viewer 的屏幕截图，该屏幕截图是在检查在康涅狄格州吉尔福德 I-95 州际公路穿过教堂街的高速公路立交桥附近收集的激光雷达样本时拍摄的。

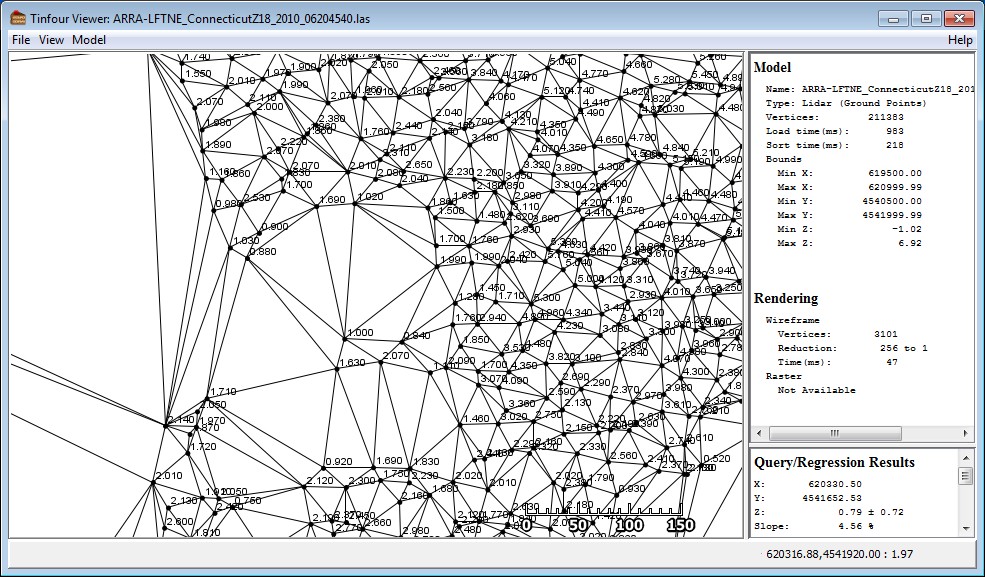


**图 23 – 根据康涅狄格州吉尔福德教堂街和 I-95 州际公路上的激光雷达数据构建的 Tinfour Viewer 图像（数据来源 NOAA，2011b）。**

Viewer 应用程序结合了上面示例应用程序中讨论的 Tinfour 的许多主要功能，包括：

1. 用于栅格渲染的高程建模和表面插值。
2. 山体阴影函数。
3. 点细化可生成复杂性降低的网格，以便在“线框”选项中进行检查。
4. 并发处理可加快光栅图像的生成速度。
5. 用于检查样品的交互式数据查询。

线框选项用于描绘 TIN 的结构。通过使用上述的点细化技术，它根据视图的比例来管理点的密度。图[以下](#_bookmark10) 是根据在康涅狄格州 Byram 附近的沿海地区收集的数据生成的，经过处理后，点的数量减少了 256 比 1。显示的一个有趣的特征是，整个收集区域的点密度并不均匀。传统的激光雷达系统使用通常被水面散射而不是反射的红外激光源。因此，样品密度在水中急剧下降是很常见的。图中[以下，](#_bookmark10)仅在该水域获得了少量数据样本。因此图中样品的分布反映了样品中不同的表面特征。



**图 24 – 沿海地区激光雷达样本的线框渲染（数据来源 NOAA，2011c）**

1. **激光雷达数据样本**

这些说明中提供的许多示例都使用激光雷达产品作为数据源。激光测量的高程数据集为 Tinfour 实现提供了一个很好的测试用例，因为它们很大，而且它们提供的真实数据描述了我们所有人都熟悉的特征。如果您希望下载并使用激光雷达产品供自己使用，则您应该熟悉一些问题。

* 1. **LAS 和 LAZ 格式**

LAS 是激光雷达数据交换的行业标准格式。它是由美国摄影测量和遥感协会 (ASPRS) 制定的非专有标准，旨在促进激光雷达信息共享。由于 LAS 是一种简单的二进制格式，遵循现代计算机数值数据表示标准，因此它可以促进高效处理和合理紧凑的数据存储。 LAS 文件还支持随机访问读取操作，该功能允许应用程序在处理激光雷达数据时更好地管理读入内存的数据量。最后，LAS 格式具有干净、简单的设计，可以轻松地为 Tinfour 库编写直接读取 LAS 文件的代码。

LAS 格式描述于<http://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf>

当然，即使 LAS 使用高效的数据二进制表示形式，激光雷达数据样本的庞大数量也会导致文件大小非常大。通过 Internet 下载数据时，数据大小尤其是一个问题，因为使用 ZIP 等传统工具无法很好地压缩 LAS 文件。

幸运的是，一位名叫 Martin Isenburg 的独立软件开发人员开发了一种名为 LAZ 的 LAS 变体，它将数据压缩嵌入到该格式中。由于压缩是专门为激光雷达数据的表示而定制的，因此它实现了出色的尺寸缩小率。此外，该实现做出了明智的选择，保留了原始版本的随机访问功能。最重要的是，Isenburg 先生将他的压缩实用程序作为开源项目发布，这导致 LAZ 作为行业标准得到广泛采用。

LAZ 格式使用的数据压缩算法相当复杂，最近才由 Reutegger (2016) 在 laszip2j 项目中用 Java 实现。 laszip2j 库当前由 Tinfour 测试和数据查看器应用程序使用。

* 1. **地面点和激光雷达数据分类**

当从飞机上收集激光雷达数据时，海拔测量传感器不知道反射激光源的特征是平坦的地面、树木、灌木丛还是人造结构。事实上，激光雷达探测鸟类或成群昆虫的情况并不罕见。因此，当激光雷达数据准备好分发时，通常会经历收集后的“分类”过程，其中根据产生样本的对象类型对每个测量结果进行分类。正如您可以想象的那样，此过程的计算量很大，并且通常涉及使用补充数据源，例如多光谱图像（航空照片）和地面调查信息。按照惯例，调查中的所有样本都包含在 LAS 数据集中。没有任何东西被丢弃。即使是那些

被拒绝为异常的数据被简单地标记为“未分类”或“保留”（一个人的异常数据是另一个人的“研究发现”）。

LAS 标准定义了多个数据类别，并为每个类别分配了一个数字代码。其中最重要的是“地面点”分类，其编号为 2。其他分类（包括地表水、道路、植被等）在 LAS 规范（ASPRS，2013）中进行了描述。 Tinfour 提供的示例应用程序主要关注地面点数据，尽管软件包中没有任何内容会限制其他数据分类的使用。

* 1. **地理坐标**

本地激光雷达数据很少使用地理坐标来收集。正如段落中所讨论的[*2.2.9 坐标和数值问题*](#_bookmark5)地理坐标不是各向同性的，因此无法很好地表示构成激光雷达勘测的地球表面小区域的 3D 特征。因此，为了促进 3D 数据样本的地理空间分析，激光雷达集合通常将水平坐标系基于使用成熟的绘图技术投影到平面的坐标。例如，上述在康涅狄格州采集的激光雷达样本最初是使用基于通用横轴墨卡托区 18N 地图投影的水平坐标收集的。

激光雷达数据的分发是另一回事。 NOAA 沿海激光雷达存储库是网络上最大的激光雷达数据集合之一。由于各种原因，NOAA 选择将其收集的所有数据转换为地理坐标。将原始激光雷达数据转换为地理坐标存在两个问题：

1. 数据不再是原来的格式，因为水平坐标（也许还有垂直坐标）已经经历了数学转换。
2. 由于数据是非各向同性的，因此需要将水平坐标投影到平坦的坐标平面上进行处理。

在 NOAA 的辩护中，他们选择将基于数十个不兼容的水平坐标系的数百个不同测量的激光雷达数据转换为一个统一的地理坐标系，这是非 GIS 从业人员所熟悉的一种。然而，这意味着在Tinfour分析数据之前，需要对其进行转换。

Tinfour 提供的 LAS 访问类自动执行此转换，尽管实现仍然有点粗糙并且缺乏重要功能（它不使用 LAS 文件中的元数据，但推断坐标是地理坐标并假设垂直单位以米为单位） ）。如果您需要对数据表示进行更多控制，las​​tools 库包含的工具允许您将 LAS 文件中的水平坐标系从地理值转换为投影值。

1. **参考文献**

阿德里安·鲍耶 (1981)。 “计算狄利克雷曲面细分”，《计算杂志》24(2)，第 14 页。 162-166。

郑少荣；戴伊，塔马尔·克里希纳；乔纳森·R·肖楚克 (2013)。 Delaunay 网格生成。佛罗里达州博卡拉顿：CRC Press。

*鲍里斯·德劳奈 (1934)。 “Sur la sphère vide”，Otdelenie Matematicheskikh i Estestvennykh Nauk 7，第 7 页。 793–800*

吉巴斯，L.，斯托尔菲，J (1985)。 “用于操作一般细分和计算 Voronoi 图的原语”，ACM Transactions on Graphics，第 4 卷，第 2 期，1985 年 4 月，第 74-123 页。

希尔伯特，D.（1891）。 “Ueber die stetige Abbildung einer Linie auf ein Flächenstück”，《数学年鉴》38 (1891)，459-460。访问自<http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002253135>

劳森，C.L. （1977）。 “C1 曲面插值软件”，数学软件 III，Rice，J.R. 编辑，第 161-194 页，学术出版社，纽约。

梁易、梅长林、张文秀（2000）。“基于地理加权回归模型的非平稳性统计检验”，环境与规划A，第32卷，2000年，第9-32页。

Natural Earth (2017)“1:10m、1:50m 和 1:110m 比例的免费矢量和栅格地图数据”。 2017 年 10 月访问自<http://www.naturalearthdata.com/>.

宾夕法尼亚州，联邦保护和自然资源部，地形和地质调查局（2006 年）。 PAMAP 程序 LAS 文件（宾夕法尼亚州 LiDAR 数据）。网站。 2015 年 12 月访问自[http://www.pasda.psu.edu](http://www.pasda.psu.edu/uci/MetadataDisplay.aspx?entry=PASDA&file=pamap_lidar_LAS.xml&dataset=1244)

佩卡姆，S.（2011）。 “轮廓、计划和流线曲率：简单的推导和应用”。电子文档于 2016 年 3 月下载于<http://geomorphometry.org/Peckham2011a>

马塞尔·鲁特格尔 (2016)。 “laszip4j – 移植到 Java 的 LASzip 库”。访问时间：2017 年 2 月<https://github.com/mreutegg/laszip4j>

罗格南特等人（1999）。 Delaunay 约束三角剖分：Delaunay 稳定算法。 “1999 年 IEEE 国际信息可视化会议”，第 17 页。 147-152。

乔纳森·R·肖楚克 (1996)。 “三角形：设计 2D 质量网格生成器和 Delaunay 三角测量器”，《应用计算几何：迈向几何工程》，编辑。 Lin, Min C 和 Manocha, Dinesh，计算机科学讲义，卷。 1148，第 203-222 页，Springer-Verlag，柏林，1996 年 5 月。2015 年 12 月下载于<http://www.cs.cmu.edu/~quake/tripaper/triangle0.html>

罗宾·西布森 (1981)。 “自然邻插值简介”。解释多元数据，第 21-36 页。埃德。 Barnett, V.，John Wiley & Sons, Inc. 奇切斯特 (1981)。

Soukal, R.、Málková, M.、Kolingerová, I. (2012)。 “TIN 模型中点定位的步行算法”，

*计算地球科学卷。 16，第 853-869 页。*

苏，P.，德赖斯代尔，R.（1996）。 “顺序 Delaunay 三角测量算法的比较”，

*计算几何 7 (1997) p。 361-385*

美国地质调查局 [USGS] (2014)。 “USGS 激光雷达点云 (LPC) OR\_PoleCreek\_2013\_000100 2014-09-26 LAS”。 2015 年 12 月下载于<https://www.sciencebase.gov/catalog/item/542fe8d4e4b092f17df623c7>

美国国家海洋和大气管理局 [NOAA] 海岸服务中心 (2011a)。 “2011年

美国农业部 - 自然资源保护局 (USDA-NRCS) 地形

激光雷达：康涅狄格州西北部”，文件 20111217\_18TXM2755.laz。 FTP 站点。 2015 年 12 月下载于<ftp://coast.noaa.gov/pub/DigitalCoast/lidar1_z/geoid12a/data/2597/>

美国国家海洋和大气管理局 [NOAA] 海岸服务中心 (2011b)。 “2011 FEMA 激光雷达：使用 GEOID12A 的带有正交垂直基准 NAVD88 的昆尼皮亚克河流域 (CT) 点云文件”，文件 18\_06934574.laz。 FTP 站点。 2015 年 12 月下载于<https://coast.noaa.gov/htdata/lidar1_z/geoid12a/data/1472/>

美国国家海洋和大气管理局 [NOAA] 海岸服务中心（2015 年）。 “NOAA Digital Coast 的激光雷达数据集”网站。 2015 年 12 月访问自<https://coast.noaa.gov/htdata/lidar1_z/>

*沃波尔，R. 和迈尔斯，R. (1989)。工程师和科学家的概率与统计（第四版）。*

麦克米伦出版公司，纽约市（1989）。

Warren, Henry S., Jr. (2013) 《黑客之乐》，第二版。培生教育公司。新泽西州上萨德尔河 (2013)。

大卫·F·沃森 (1981)。 “计算 n 维 Delaunay t 曲面细分并应用于 Voronoi 多变体”，《计算杂志》24(2)，第 11 页。 167-173。

大卫·C·惠勒、安东尼奥·佩斯 (2010)。 “地理加权回归”，应用空间分析手册：软件工具、方法和应用，第 461-486 页，Springer-Verlag，柏林 2010 年。Wilson, John P.、Gallant, John C. (2000)。地形分析：原理与应用。约翰威利父子公司，纽约（2000 年）。