并行自适应层网格剖分

2024 年 1 月 23 日

目录

第 1 章　层网格剖分

增材制造工艺仿真的残余应力和变形计算中，可以分为小尺度和构件尺度，小尺度考虑热循环细节，采用生死单元法，构件尺度忽略热循环细节，采用固有应变法。生死单元法中需要根据热源移动激活网格单元，采用静单元和激活单元的混合方法。热源是根据事先规划好的路径进行移动，路径是逐层规划的，因此对网格剖分提出了特殊的要求，网格单元需要在一层。固有应变法也需要对网格进行分块。因此我们针对增材制造特殊要求，整理计算几何中网格剖分算法，在现有层网格剖分软件基础上实现自适应和并行，并比较 Delaunay 网格和像素网格的效果。计算几何请参考 [1]、[2] 和 [3]，网格剖分请参考 [4] 和 [5]，计算机辅助设计（CAD）请参考 [6]。

网格剖分包括结构性网格和非结构性网格，结构性网格剖分主要包括 Algebraic Interpolation Method 和 PDE-based Methods，Algebraic Interpolation Method 通过映射将简单形状转成复杂形状，非结构性网格剖分主要包括 Spatial Decomposition Method、Advancing-front Method、Delaunay Technique，网格剖分还可以分为二维网格剖分、三维网格剖分和三维曲面网格剖分，三维曲面网格剖分可以采用结构性网格剖分映射类方法，也可以采用非结构性网格的剖分方法。增材制造层网格剖分由于网格逐层增加，和 Advancing-front Method 非常类似，可以通过设置推进距离满足层网格要求，区别在于三维实体模型是通过切片定义还是通过边界网格定义。根据切片定义，CNR IMATI 的团队采用了层二维剖分到三维，我们对该方法进行研究并和其他方法进行比较。路径规划方法可以借鉴网格剖分，分为二维、三维和三维曲面，分为等距线、等距面、截平面，二维等距线可以采用法向等距或者费马螺旋曲线等距，三维等距面可以采用法向等距，法向等距类似 Advancing-front Method，例如 infill 和 offset，三维曲面路径可以采用映射也可以采用测地线等距，或者截平面，测地线等距和截平面类似 Advancing-front Method，路径规划需要一些微分几何的知识。

首先看 Delaunay 网格剖分，狄利克雷镶嵌（Dirichlet Tessellation），沃罗诺伊图（Voronoi Diagram）和德劳内三角网格（Delaunay Triangulation）是网格剖分的基本概念。狄利克雷首先提出了可以将平面分割成凸单元，其次沃罗诺伊进行了进一步研究，并扩展到三维，最后德劳内验证了可以通过沃罗诺伊图的对偶获取三角网格，这种三角网格具有唯一性和很好的性质，最小角比其他存在的三角网格的最小角都大。沃罗诺伊图定义中单元和点集中某一点对应，单元中的点离该点距离比离点集其他点都近，如果是二维问题就是由连接两邻点直线的

垂直平分线围成的多边形。德劳内三角网格生成有不同方法，可以根据沃罗诺伊图对偶生成，比较常用的是递增法（Incremental Method），递增法是基于德劳内引理。德劳内引理证明了如果对于每对相邻单形都满足空外接圆准则，那么整个网格满足空外接圆准则并且是德劳内三角网格。基于德劳内引理，定义德劳内核，德劳内核原理是往旧网格中插入一点，如果该点在某一网格单元内，将该点和网格单元三个顶点连线，如果该点在网格单元某一边相邻网格单元外接圆内，则将该边进行翻转，从而获取新网格。插入点还有落在所有网格单元外的情况，为了避免这种情况，采用了一个技巧（Reduced Incremental Method），定义一个盒子包括了整个点集。我们对主要开源网格剖分软件中的数据结构和算法进行研究，包括 CGAL、Triangle、Netgen、Tetgen、Gmsh、OpenCASCADE，OpenCASCADE 是 CAD 软件也需要表面网格剖分，从而形成了非结构性层网格剖分框架。

1.1 非结构性网格

1.1.1 CGAL

1.1.1.1 数据结构和算法

CGAL 是一个重模板的软件，新版本程序全部都写在头文件中，因此不需要编译。CGAL 中包括二维和三维的点集生成三角网格（Triangulation），包括二维和三维的网格剖分（Mesh Generation），两者的区别是网格剖分是在点集生成的三角网格基础上根据网格质量准则进一步处理，比如进行德劳内加密。在三维网格剖分中有周期性网格剖分，后边将测试是否可以用于层网格剖分。CGAL 中二维点集生成三角网格有四个算法，分别为 Triangulation_2、Delaunay_triangulation_2、Constrained_triangulation_2、Constrained_Delaunay_triangulation_2，还有一些其他算法，程序在 Triangulation_2 文件夹中。第二个算法和第三个算法基于第一个算法，第四个算法基于第三个算法，第一个算法实现了点递增插入，第二个算法增加了翻转，第三个算法增加了约束，第四个算法增加了翻转和约束。通常来说网格数据结构包括顶点坐标和单元编号两部分，在这四个算法中默认定义的数据结构是 Triangulation_data_structure_2，程序在 TDS_2 文件夹中。Triangulation_data_structure_2 中采用了两个 Compact_container 数据结构分别保存顶点和单元，Compact_container 是 CGAL 自己定义的数据结构，程序在 STL_Extension 文件夹中。CGAL 中二维网格剖分算法有 Delaunay_mesher_2，程序在 Mesh_2 文件夹中，Delaunay_mesher_2 需要先提供一个已知点集生成的三角网格。CGAL 中三维点集生成四面体网格有三个算法，分别为 Triangulation_3、Delaunay_triangulation_3 和 Regular_triangulation_3，第二个算法和第三个算法基于第一个算法，程序在 Triangulation_3 文件夹中。数据结构采用了 Triangulation_data_structure_3，程序在 TDS_3 文件夹中，同样采用了 Compact_container 数据结构保存点和单元。CGAL 中三维网格剖分算法有 make_mesh_3，实现了德劳内加密，其中定义了 Mesh_complex_3_in_triangulation_3，并

依赖 Mesh_complex_3_in_triangulation_3_base，其中定义了点和单元数据结构，程序在 Mesh_3 文件夹中。

1.1.1.2 算例

当插入一个新点时，图 1.1是 Triangulation_2，因此没有发生翻转。图 1.2是 Delaunay_triangulation_2，发生了翻转。图 1.3是 Constrained_Delaunay_triangulation_2，对边进行约束，因此进行了约束下翻转。图 1.4是 Delaunay_mesher_2，对边进行了约束，因此进行了约束下翻转，并进行了德劳内加密。

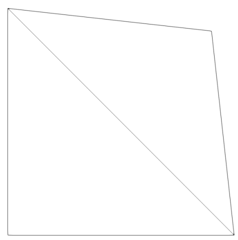

图 1.1: Triangulation_2

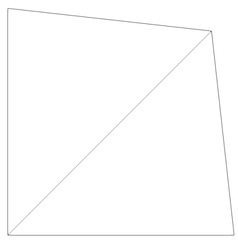

图 1.2: Delaunay_triangulation_2

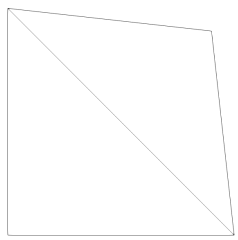

图 1.3: Constrained_Delaunay_triangulation_2

当插入一个新点时，图 1.5是 Triangulation_3，因此没有发生翻转。图 1.6是 Delaunay_triangulation_3，发生了翻转。图 1.7采用 OFF 文件定义了多面体，多面体的面只能为三角形，进行了网格剖分。

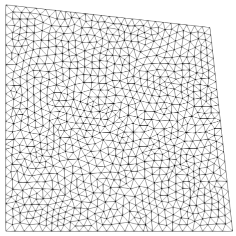

图 1.4: Delaunay_mesher_2

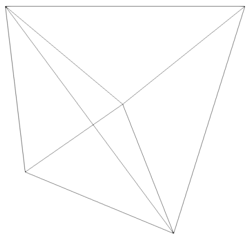

图 1.5: Triangulation_3

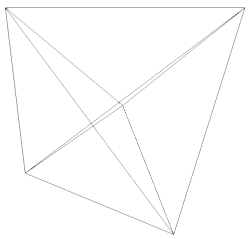

图 1.6: Delaunay_triangulation_3

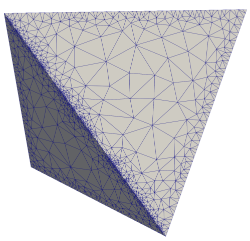

图 1.7: make_mesh_3

1.1.2 Triangle

Triangle 是一个二维 Delauany 网格剖分的软件，是 Jonathan Richard Shewchuk 开发的。软件包括四个文件，triangle.h 和 triangle.c，可视化用的 showme.c，以及一个 c 语言接口例子 tricall.c。很有意思的是 showme 是直接基于 x11 开发的。其中 triangle.c 中有 main 函数定义，可以直接从 triangle.c 编译成可执行程序，但是需要注释掉 TRILIBRARY 的定义，也可以将 triangle.c 编译成链接库，然后 tricall.c 调用链接库，将 tricall.c 编译成可执行程序，或者将 triangle.c 和 tricall.c 放在一起编译成可执行程序。Triangle 中给了一个例子，读取 A.poly 文件中数据进行网格剖分，结果会生成 A.1.* 的文件，然后用 showme 打开。在 CMakeLists 里将 var 设置成 ON，是将 triangle.c 编译成可执行程序，设置成 OFF，是将 tricall.c 编译成可执行程序。运行 install.sh 文件，编译安装后在 triangle_install/bin 文件夹中可以运行 triangle_run -p A 进行测试，用 showme A.poly 打开查看。

1.1.2.1 数据结构和算法

1.1.2.2 算例

1.1.3 Netgen

1.1.3.1 数据结构和算法

1.1.3.2 算例

1.1.4 Tetgen

1.1.4.1 数据结构和算法

1.1.4.2 算例

1.1.5 Gmsh

1.1.5.1 数据结构和算法

1.1.5.2 算例

1.1.6 OpenCASCADE

1.1.6.1 数据结构和算法

1.1.7 Slice2Mesh

1.1.7.1 数据结构和算法

1.1.7.2 算例

1.2 结构性网格

1.2.1 协调性网格

1.2.1.1 Clipper

1.2.1.2 RnD

1.2.1.3 数据结构和算法

1.2.1.4 算例

1.2.2 非协调性网格

1.2.2.1 数据结构和算法

1.2.2.2 算例

第 2 章　网格自适应性

第 3 章　网格并行

3.1　Metis

第 4 章　残余应力和变形

4.1 热弹塑性

4.2 小尺度热循环生死单元法

4.3 构件尺度固有应变法

参考文献

[1] Jean Daniel Boissonnat, Mariette Yvinec, and Herve Bronnimann. Algorithmic Geometry. Cambridge University Press, 2001.

[2] Jakob Andreas Bærentzen, Jens Gravesen, François Anton, and Henrik Aanæs. Guide to Computational Geometry Processing. Springer, 2012.

[3] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational Geometry Algorithms and Applications. Springer, 2012.

[4] Pascal Jean Frey and Paul-Louis George. Mesh Generation. HERMES Science Publishing, 2000.

[5] Paul-Louis George and Houman Borouchaki. Delaunay Triangulation and Meshing. HERMES Science Publishing, 1998.

[6] Erich Hartma. Geometry and Algorithms for COMPUTER AIDED DESIGN.