计算几何大实验报告

基于半边结构的二流形正则三角网格实体布尔运算

2012212888 赵鸿泽
2012212858 钱康来
2012212864 程显昱

项目概述

利用布尔运算可以将一些简单的形体构造出复杂的三维形体。

本学期计算几何大实验，小组采用半边结构，对三维的三角网格模型文件（.OBJ 类型文件）进行读取和拓扑重建，利用计算几何中讲到的面面求交、线面求交、线线求交、三角剖分、包含测试、点定位等，实现二流形正则实体的布尔运算（交、并、差），以及分步演示。

实验环境

本地操作系统：Windows 8 / Windows 7 64-bit

编辑软件：Visual Studio 2012

本地编译器：Visual Studio 2012

CPU：Intel(R) Core(TM) i5-2320 CPU @ 3.00GHz (4 CPUs), ~3.2GHz

内存：8.00GB RAM

测试模式：Release

布尔运算

布尔是英国的数学家，在 1847 年发明了处理二值之间关系的逻辑数学计算法，包括联合、相交、相减。在图形处理操作中引用了这种逻辑运算方法以使简单的基本图形组合产生新的形体。并由二维布尔运算发展到三维图形的布尔运算。[3]

三维形体的布尔运算通过对两个以上的物体进行并集、差集、交集的运算，从而得到新的形体。系统提供了 3 种布尔运算方式：Union（并 A∪B）、Intersection（交，A∩B）和 Subtraction（差，A-B）。

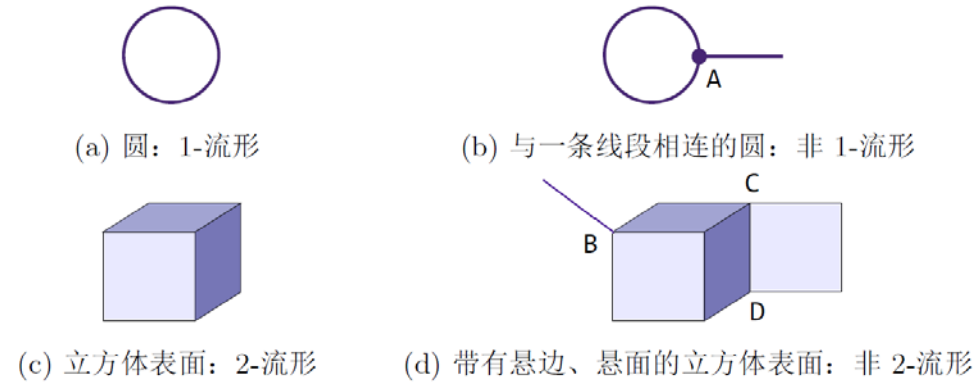
本次大实验系统中，形体 A 和 B 在进行布尔运算后形成新的形体 A，可以加入新的形体 B，继续参与布尔运算得到新的形体。可以选择不同的运算方式直接得到布尔运算结果，也可以选择分步演示。通过选择不同的选项修改图形的显示内容和效果，方便观察和调试。

流行和正则实体简介

流形(manifold) 是数学中的术语，由于其严格定义所需预备知识较多且较为抽象，下面仅直观地介绍必要的概念。更严密、更系统的介绍可参考书籍[1]或[2]等。

直观地说，流形是局部具欧氏空间性质的空间。更形式化地说，*n*-流形(*n*-manifold) 是一个局部同胚于欧氏空间 R^n 的拓扑空间，即其中任一一点都有一个邻域，该邻域能够被一个连续双射（逆也连续）映射到 R^n。

举例而言，圆是 1-流形的，而图 1(b)不是；立方体的表面是 2-流形的，而图 1(d)的表面不是。



(a) 圆：1-流形　(b) 与一条线段相连的圆：非 1-流形

(c) 立方体表面：2-流形　(d) 带有悬边、悬面的立方体表面：非 2-流形

图 1: 流形与非流形示例。点 A 处无法找到与 R1 同胚的邻域；点 B 和线段 CD 处无法找到与 R2 同胚的邻域。

以可定向的 2-流形为边界的形体表示是一类常见的表示方法，具有很多好的性质，如易于理解、表示效率高、操作效率好、数据结构紧致、算法简单。

为解决运算封闭性的问题，英国人 Requicha 提出了正则集合运算(regularized set operation)[4]，正则集合(regular set) 的在此运算下封闭。正则集合是这样一个集合，它等于它内部(interior)的闭包(closure)。正则集合运算首先对正则集合完成标准的布尔运算，然后对结果进行正则化(regularization)。直观而言，正则化的过程是“去掉边界”再“补上边界”的过程。对于三维正则形体，此过程将丢弃悬边、悬面和孤立点。

需要注意，正则和流形是两个不同的概念，而且三维正则形体的边界也不一定是 2-流形的，如图 2 所示。图中两个立方体共边，其边界在该边处不是 2-

流形的。但是，此形体内部的闭包与此形体相等，因而此形体是正则形体。此形体可通过两个立方体求并得到。

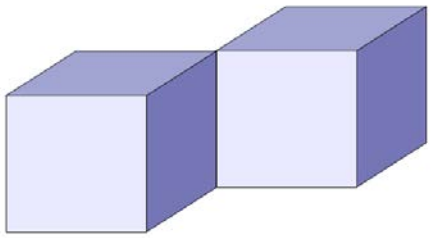


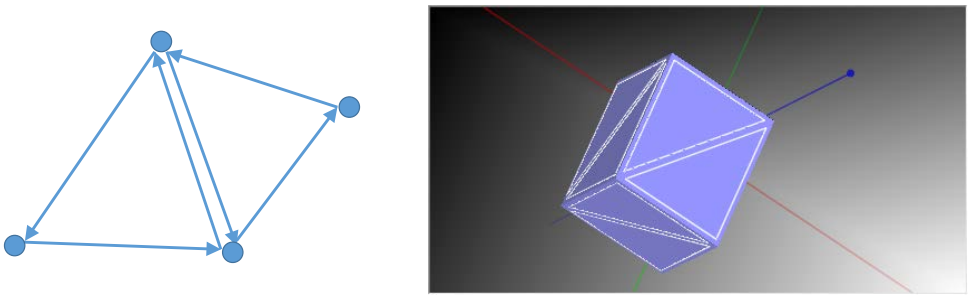
图 2 正则三维形体，但边界并非 2-流形

在实际应用中，非正则形体是很常见的。实际上在大实验进行当中，我们的系统除了能够处理 2-流形正则形体之外，也具有一定非 2 流形正则形体的处理能力。我们允许非流形的连接关系，即允许两个几何体共点、共边、共面，但不允许存在独立的点、边、面，参与布尔运算的形体必须是三角网格封闭实体。如果在运算过程中产生此类点、边、面，则会被删去。

数据结构

半边（coedge）结构是一种比较流行的数据结构，它以指针的形式保存几何形体的拓扑结构。

系统中用的半边结构文件（.OBJ 类型文件）由 3D Max 导出。文件中存储内容为顶点，法向量以及三角形（三角形由三个顶点和法向量表示）。



(a) **(b)**

图 3 半边结构表示的面片和几何实体。(a)半边结构表示的相邻三角面片 (b)半边结构表示的三角网格实体，图中白色三角形为组成实体表面的半边三角形。

读取之后的半边结构三角网格实体由三个对象组成：顶点（vertex）、半边（coedge）和三角面片（coface）。顶点表示形体中的实际点，半边可视为一条

有向边，封闭的网格中的一条边由 2 条方向相反的半边组成，一个封闭的三角面片包括逆时针连接的半边。

每个实体对象中包含一个 coface 数组，存储该实体的所有三角面片。每一个三角面片对象中包含一个 coedge 数组，逆时针存储组成该面片的半边。每一个 coedge 半边包含三个指针，分别指向它所属于的面片、coedge 的起点（from 指针）和 coedge 的终点（to 指针）。

算法过程

Step1:三角面片面面求交

对于两个形体中的每个面进行求交，得到一个封闭折线环。实体模型同一条边重合导致实体相切的情况视为分离。面面相切和部分边相切时得到的切线不成环。

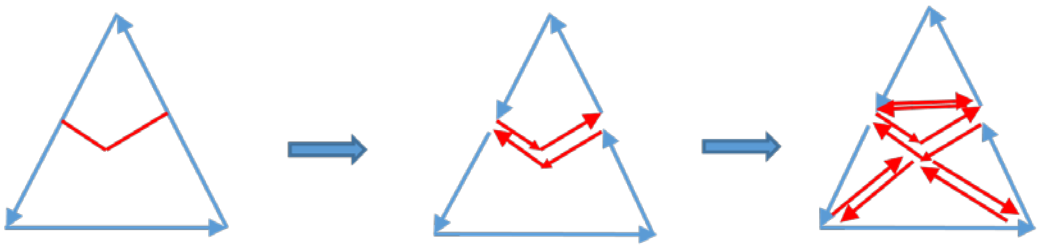
两个实体求交的过程中，用一个实体的每一个三角面片和另一个实体的每一个三角面片分别求交，循环次数为 n^2，复杂度比较高。为了降低时间复杂度，每两个面片求交之前会进行包围盒相交测试，如果两个三角面片包围盒不想交，就没有交线，不用参与求交。

如果两个三角面片包围盒相交，则进行求交处理，根据求交结果返回求交类型。如果是正常相交，则返回交线段。如果两个面重合或者共面，则相对应进行处理。

为了减少面面包围盒相交测试的次数，后来我们利用包围盒建立了 RTree，求交时可快速剪枝，减少了不必要的检测。这部分会在优化部分进行说明。

Step2:用交线切割三角面片

面面求交得到了封闭的折线环，每一条折线段均有指向其所在三角面片的指针。因此，遍历折线环中的每一条折线段，分割其所在的三角形面片。三角面片分割的过程如图 4(a)到 4(b)所示，每一条折线段形成两条反向的半边，三角形被交线分割成新的多边形，目前还没有得到多边形的拓扑。



(a) (b) (c)

图 4 用交线切割三角面片，形成新的多边性面片，并重新三角剖分

切割三角面片要处理一些特殊情况：交线在一个三角形的边上，面面完全重合，面面部分重合切割后的完全重合部分。

Step3:搜寻 Coedge 环

每一个三角面片被分割之后，尚未搜索得到其被分割之后的面片，这就需要搜索 coedge 环，得到所有的新多边形，并把它三角分割。搜寻过程中要考虑内环情况。

搜索过程：从任一点开始搜索，对与 coedge->to 邻接的所有 coedges 进行转角排序，使用非共线的转角最小的一条 coedge 作为下一个起始 coedge 递归搜寻。

结束条件：成环则产生一个新的面；所有的边被处理则结束递归。

输出：多个由有序 coedge 环构成的面。

此时如图 4(b)中的三角形，才真正被分割成为多个新的多边形。

Step4:三角剖分

对于每一个环进行 Delaunay 三角剖分，容许有内环、内环内嵌套外环等情况。

由于布尔运算中三角面片被分割后形成的新多边形，边数都比较少，可能存在少量内环的情况，分割复杂度比较小，且多边形边为有序序列。因此我们采用了[5]中所提出的一种任意多边形的 Delaunay 完全三角剖分算法。这种算法的复杂度为$O(n^3)$，但如之前所述，n 值很小，复杂度很低。该算法的优点是思路简单，简洁通用，能直接处理多内环情况，不需要额外处理。

Step5:三角面片分类

对于所有重新三角剖分产生的三角面片，为了方便布尔运算时取舍面片，需要将新生成的三角面片分为以下几类：

AinB（BinA）：属于 A(B)且被 B(A)包含的三角面片

AoutB（BoutA）：属于 A(B)且不被 B(A)包含的三角面片

AonB+（AonB-）：A 上的与 B 重合的三角面片，且两面片同(异)向

三角面片的核心问题是如何判定一个三角形在另外一个体的内部。系统所有方法为射线法，即从改三角形内任一点为起点的任一方向的射线，与另外一个体求交，求交点数为奇数时为内部，偶数时为外部。要处理一些特殊情况，如射线与面重合、射线交于多个三角形的边、顶点等。

Step6:根据布尔运算类型进行面取舍

布尔运算的最后一步是据布尔运算类型进行面取舍，将运算中多余的面删除，保留有效的面。取舍的规则如下三个公式：

$boolean(A \cup B)=AoutB \ \& \ BoutA \ \& \ AonB+$

$boolean(A \cap B)=AinB \ \& \ BinA \ \& \ AonB+$

$boole(A-B)=AoutB \ \& \ (BinA)\text{^}(-1) \ \ \& \ AonB-$

优化

Rectangle R-Tree

系统中利用包围盒建立 R-Tree，在求交、分类时，可以快速剪枝。

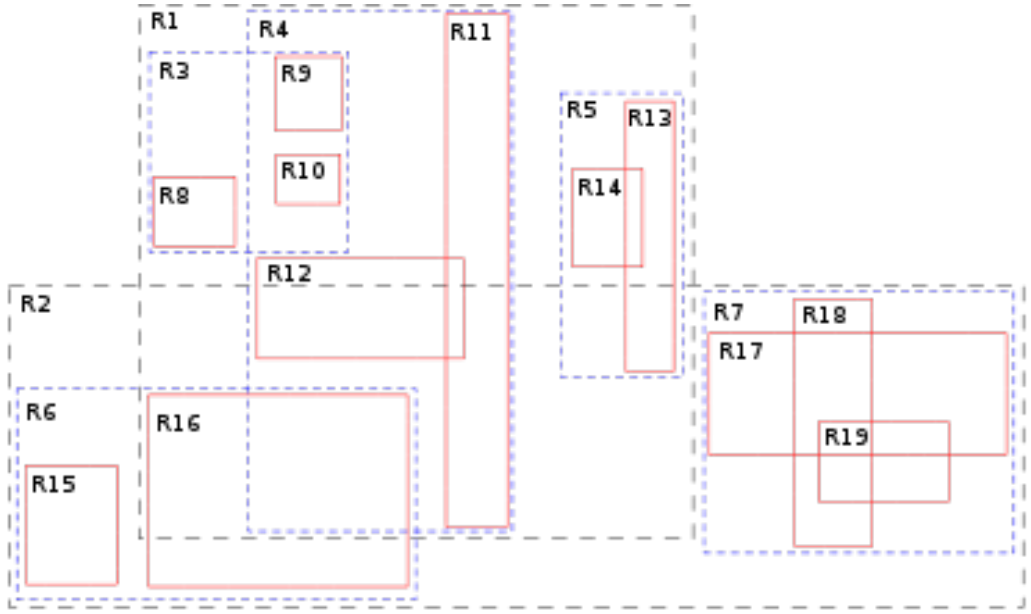
R-树是一种类似于 B 树的高度平衡树，采用最小外接矩形（包围盒）界定空间实体。对于一棵 M 阶的 R-树，其节点结构可描述如下[6]：

叶子节点：(COUNT,LEVEL,<OI_1,MBR_1>,<OI_2,MBR_2>,…,<OI_M, MBR_M>)；

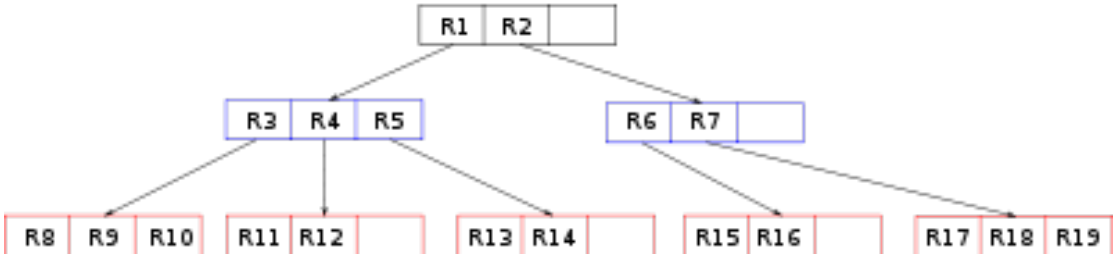
中间节点：(COUNT,LEVEL,<CP_1,MBR_1>,<CP_2,MBR_2>,…,<CP_M, MBR_M>)。

其中，<OI_i, MBR_i>称为数据项，OI_i 为空间目标的标识，MBR_i 为该目标在 k 维空间中的最小包围矩形(简称为数据矩形)；<CP_i,MBR_i>称为索引项，CP_i 为指向子树根节点的指针，MBR_i 代表其子树索引空间，为包围其子树根节点中所有目录矩形或数据矩形的最小包围矩形(简称目录矩形)；COUNT≤M 指示节点

中用的索引项或数据项个数(即该节点的孩子个数)；LEVEL≥O 指示该节点在树中的层数(O 表示叶节点)。



(a) 平面中的包围盒



(b) 建立的 R-Tree

图 5 空间中的包围盒划分以及以此建立的 R-Tree

图(a)中的红色实线框(R_8—R_{19})表示空间目标的包围盒，R_1—R_7 表示中间节点索引项对应的索引空间，其中黑色虚线框 R_1,R_2 为根节点，R_3—5_7 表示中间节点。

图(b)是对应图(a)中空间对象集合的 R-Tree。系统中为每一个三角面片均有一个三维空间包围盒，为它们的包围盒建立一个 R-Tree，每一个叶子节点含有一个指向相对应的三角面片的指针。此时对于求交过程，只需要将同一个父节点之下的三角面片进行求交即可，不在同一个父节点下的三角面片不需要互相求交。

R-Tree 对系统运行效率的提升非常明显。对于 dinosaur（80554 个面片）和环形曲面体 torus（2880 个面片）的交并补运算，采用 R-Tree 之前需要大概 100 秒能正确显示结果，采用 R-Tree 之后 1 秒之内就可以显示结果。而对于 dinosaur（80554 个面片）和 budda（60866 个面片）的交并补运算，采用 R-Tree 之前需要 20-30 分钟能显示结果，采用 R-Tree 之后仅需 1.5 秒钟左右。（R-Tree 真是个神器呀！）

容差

容差（tolerance）有很多优势，能够增强运算的准确性，避免极端数值时的判定错误，方便进行误差控制和预测等。采用了容差之后，在对数值大小、向量长度进行判定时，不再以单纯的相等或 0 等简单数值来界定，而是以判定条件在容差规定范围内为准。

在计算机编程中，容差可以避免计算机在很多场合下（浮点数运算）的误判断问题。例如比较 a 与 b（double 或 float）是否相等时，传统的比较方式为 $a == b$，对于判断可能存在问题。如果设 tolerance=10^{-8}，以$|a - b| < tolerance$为判定公式，就可以在 tolerance 范围之内将 a，b 判定为相等（比如 a=10^{-10}，b=10^{-12}；或 a=1.0，b=1.0+10^{-9}）。

使用容差时一个很重要的问题是容差大小的选取。容差选取过小和过大，均有产生误判的可能，如果由于误差选取而产生 bug，则需要重新选取误差，反复测试。在程序中要区分不同的场合和公式，根据需求设置不同的容差，例如简单数值比较时需要数值容差，向量长度或模比较时的可能就不能采用数值容差而需要单独设置长度容差，此外还有数值平方容差等等。

系统中在进行重合判定、垂直判定、平行判定等算法中，均使用了容差。提高了重合、判定时的准确性。

问题与拟解决方案

目前系统已经可以处理大部分的形体布尔运算，且不出现明显 bug，但是仍有些特殊情况的细节待处理。

射线与三角面片求交算法中有 bug，这使得极个别分割后的面片分类时出现错误，可能布尔运算求出来的几何形体中缺一块三角面片。

针对布尔运算结果的准确性测试和系统的性能测试需要完善。

有一些算法细节还需要优化，特别是针对大规模三角模型的处理，最好是使用 OpenMP、OpenCL 等加速。

总结

从搭建 SVN 服务器，开始编写系统框架，到编写和完善系统的算法流程，系统可运行之后的调试、优化，再到最后编写文档，总过经过一个月的时间，我们总共提交了 180 个版本的代码，顺利完成了计算几何的大实验。

系统支持基于半边结构的二流形正则三角网格实体布尔运算，包括三角网格实体的交、并、差运算，以及步骤演示。并可以根据选择，实现不同内容和形式的显示。为了方便演示，我们不仅实现了 MFC 界面的演示程序，还使用 Javascript 语言对核心算法部分进行了处理，实现了可以在网页中展示的 demo。系统的详细功能和使用方法，请参见《使用手册》。

目前的系统具有较高的鲁棒性，不会因为异常情况而崩溃。经过优化之后，MFC 演示程序具有较高的吞吐量和运行效率，能够快速完成由上万规模的三角面片组成的实体之间的布尔运算。Javascript 版本实现了更多交互操作上的控制，用户可以指定模型的位置、方向、包括渲染设置等；但是由于浏览器中 Javascript 本身的特性，程序运行效率较低，对大模型处理起来力不从心。

计算几何课程中讲到的面面求交、线面求交、线线求交、三角剖分、包含测试、点定位、R-Tree 等知识在系统得到了运用。通过课程的学习，让我们能够以更简单实用的方法来解决系统中的算法问题，不仅正确性得到了保障，算法复杂度也做到了很大程度上的降低。计算几何的强大能力在系统中得到了体现。

感谢邓老师在课堂上耐心细致的讲解，我们在课程中学到了很多，在实践中应用了很多，这才有了一些明悟。也非常感谢我们三位组员在实验过程中的

通力合作，系统编写过程中我们分工明确，各司其职，共同克服和解决了一些困难和问题，最终完成了大实验。

参考文献

[1] Hoffmann C M. Geometric and Solid Modeling - An Introduction. Morgan Kaufmann, 1989.

[2] Hachenberger P, Kettner L. 3D Boolean Operations on Nef Polyhedra, September,2011. http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Nef_3/Chapter_main.html.

[3] 百度百科：布尔运算. http://baike.baidu.com/view/638530.htm

[4] Requicha A G. Representations for rigid solids: Theory, methods, and systems. ACM Computing Surveys, 1980, 12(4):437–464d.

[5] 涂治红，桑农。用VC 语言实现任意多边形的Delaunay完全三角剖分算法。计算机与数字工程，第33卷，34-36。

[6] 刘润涛，安晓华，高晓爽。一种基于R-树的空间索引结构。计算机工程，2009年12月，第35卷，第23期。