

文章编号:1001-9081(2010)11-3002-03

三维网格模型的快速拓扑重建算法

侯宝明,崔红霞,刘雪娜

(渤海大学 信息科学与工程学院,辽宁 锦州 121000)

(hbm_008@sina.com)

摘要:为了提高重建三维网格模型拓扑的速度,选择半边结构作为表示实体模型拓扑关系的结构模型,设计了新的用于加快顶点合并的索引方法。在顶点合并时直接定位到欲查找的顶点位置上,无须借助 AVL 等辅助查找表。拓扑重建的时间复杂度由原来的 $O(n \log n)$ 降低至 $O(n)$ 。通过 SMF 格式文件进行的测试结果表明,在普通 PC 上重建含有 10 万个三角面片模型的拓扑结构也只需 1 s。

关键词:三维(3D)网格模型;拓扑重建;SMF 文件;半边结构;三角面片

中图分类号: TP391.72;TP301.6 **文献标志码:** A

Fast topological reconstruction algorithm for 3D mesh model

HOU Bao-ming, CUI Hong-xia, LIU Xue-na

(College of Information Science and Engineering, Bohai University, Jinzhou Liaoning 121000, China)

Abstract: In order to speed up the reconstruction of the topology of 3D mesh model, half-edge structure was selected to represent the topological relation of solid model. A new index method to quicken vertices combination was designed. During the process of vertex combination, directly locating the vertex position was searched, without the AVL lookup table, so that the time complexity of the topological reconstruction was reduced to $O(n)$ from $O(n \log n)$. The results of test by SMF format files show that the model with one hundred thousand triangular facets can be reconstructed within a second in popular PC.

Key words: 3D mesh model; topological reconstruction; SMF file; half-edge structure; triangular facet

0 引言

三维模型系统通过读取表示模型的数据文件在系统中重建模型,由于系统一般要对输入后的模型做各种处理(缺陷修复、简化和优化等),在内存中存储模型的拓扑关系十分必要,所以读取模型数据文件时一般都对模型的拓扑关系进行重建。读取模型的数据文件本身并不耗时,但是在建立拓扑关系过程中由于涉及到顶点的查找和合并,所以比较耗时,一般的模型系统重建一个较大三维模型(包含几万个三角面片)的拓扑需要十几秒,甚至更多,提高模型的拓扑重建速度一直是一个需要解决的问题。近年来对这个问题的研究一直没有停止,且主要集中于 STL 格式的实体模型,原因是 STL 格式文件应用广泛,比如几何造型、快速成型制造等,而且还可以使用 STL 文件直接生成有限元网格^[1]。为了提高拓扑重建速度,一般都借助一个辅助查找表来加快顶点合并,文献[2-6]采用虚的 AVL 树作为辅助查找表,文献[7]采用红黑树,文献[8]采用散列表。本文提出一种基于 SMF 格式文件的快速拓扑重建算法,与以往的算法相比时间复杂度降低了一个数量级,速度更快。

表示三维模型的文件格式有许多种,3D 模型格式转换软件 3D Object Converter 就能够处理 571 种格式文件。不同领域的模型系统有自己常用的格式文件,比如 CAD 系统一般采用 IGES、STL、STEP、SAT 等。当前比较流行的有 3DS、OBJ、SMF、PLY、IV、OFF 等。与 STL 相比 SMF 格式更为简单,冗余度更低。

1 SMF 文件格式简介

SMF 格式文件设计最初是为了描述几何模型,由 OBJ 格式文件演化而来。SMF 格式文件结构非常清晰、简单,由一系列的行组成,每行都是自解释的,可以是下面 3 种形式之一。

1) 空行,这些行可以完全忽略。

2) 注释行,以#开始。

3) 命令行, <op> <arg> *。第一个标记表示命令的名字,余下的是参数,参数的含义与命令名有关,它们之间以空格分割。

命令可以简单分成以下几类:

1) 关键运算符。

v <x> <y> <z>

描述一个三维 vertex 坐标,有多行描述模型时 vertex 编号从 1 开始。

f <v1> <v2> <v3>

描述一个三角面片(face),v1,v2,v3 是 vertex 的编号,三角面片的 3 个顶点被逆时针列出。

2) 模型属性运算符。

bind [c|n|r] [vertex|face] 为 vertex 或 face 绑定一个属性值。

c <r> <g> 表示颜色。

n <a> <c> 表示法向量,自动被转换成单位法向量。

r <s> <t> 表示纹理坐标。

更多的格式细节请参见文献[9]。

收稿日期:2010-05-14;修回日期:2010-07-21。 基金项目:辽宁省教育厅重点实验室基金资助项目(2008s02)。

作者简介:侯宝明(1976-),男,辽宁瓦房店人,讲师,硕士,主要研究方向:几何造型、计算几何;崔红霞(1969-),女,黑龙江伊春人,教授,博士,主要研究方向:图形图像处理;刘雪娜(1976-),女,辽宁锦州人,讲师,硕士,主要研究方向:计算几何。

2 拓扑重建算法

2.1 拓扑信息表示

表示实体模型拓扑关系可以采用很多结构模型,常见的有:半边结构(half-edge structure)^[10]、翼边结构(wing-edge structure)、放射边结构(radial-edge structure)^[11]等。本文采用半边结构模型来描述拓扑关系,模型中的主要结构包括:Solid(实体)、Face(三角面片)、Edge(边)、Half-edge(半边)、Vertex(顶点)。Face、Edge 和 Vertex 节点都采用双向链表来组织。

```
struct Solid
{
    Id solidno;          //实体编号
    Face *sfaces;        //指向三角片链表的指针
    Edge *sedges;        //指向边链表的指针
    Vertex *sverts;       //指向顶点链表的指针
    struct Solid *nexts;
    struct Solid *prevs;
    ...                  //实体的其他属性(略)
};
```

整个模型的核心结构是 Half-edge,一个 Edge 对应两个 Half-edge,它们方向相反,一个 Face 的 3 个 Half-edge 形成一个封闭的环。图 1 中,边 E 与两个半边 H_1 、 H_2 相对应,半边的引入使得一个边被共享该边的两个三角面片所共享,如果某一个边缺少一个半边,说明与该半边对应的三角面片缺失。

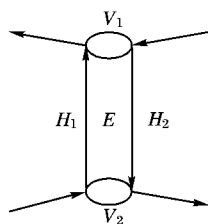


图 1 半边结构

模型建立之后可以轻松实现下述操作:通过 Solid 遍历它的所有 Face,所有的 Edge 和所有的 Vertex;已知一个 Edge 可以找到它的两个 Half-Edge;已知一个 Face,可以找到它的 3 个 Half-Edge;已知一个 Half-Edge 可以找到它的起始 Vertex 和终止 Vertex。拓扑重建就是建立起这些节点并设置好相互之间的关系,重建完成之后在内存中就存在一个完整的三维实体模型供后续操作使用。

2.2 重建算法

拓扑重建工作分成以下几个方面:

1) 建立 Solid 节点,从 SMF 文件中读取 Vertex、Face 信息建立 Face 节点,将 Face 节点加入到 sfaces 链表中。

2) 检查 Face 中的 3 个 Vertex 在 Vertex 链表中是否存在,对已存在的 Vertex 进行合并,为不存在的 Vertex 之间的 Edge 新建节点。假定 3 个 Vertex 依次是 V_1 、 V_2 、 V_3 ,如果 V_1 和 V_2 至少有一个不存在,则建立 V_1 和 V_2 之间的 Edge 并加入到链表中,同时要为不存在的 Vertex 新建节点。

3) 为当前 Face 的每一个 Edge 建立一个 Half-edge 节点,将该节点与对应的 Edge 关联起来。由于每个 Face 都要做这个工作,所以共享同一个 Edge 的两个 Face 通过 Half-edge 建立起了邻接关系。

4) 所有 Face 读进来之后模型的拓扑结构就建立起来了。

重建算法中比较耗时的操作是顶点的合并。在读取三角面片时被多个三角面片共享的同一个顶点会被多次输入,在

实体模型中该顶点只能用一个 Vertex 节点表示,所以读取文件的顶点时都需要判断该顶点在顶点链表(sverts)中是否存在,若存在则直接使用,否则新建 Vertex 节点。

判断一个顶点在顶点链表中是否存在最直接的方法就是遍历顶点链表,下面我们分析一下它的时间复杂度。一个正则实体模型的顶点、边和面之间满足 $V - E + F = 2$ (欧拉公式), V 为顶点个数, E 为边个数, F 为面片个数。由于模型表面由三角面片组成,每个面片有 3 条边,每条边被两个面片共享,所以有 $E = 3F/2$,代入上面的欧拉公式得到顶点数和三角面片数之间的关系为 $V = F/2 + 2$,对于分析复杂度而言可简化为 $V = F/2$ 。每读取一个三角面片,它的 3 个顶点都要到顶点链表中查找,所以顶点合并的复杂度为 $3 \times O(F/2)$,即 $O(3F/2)$ 。后续的拓扑关系设置都是以顶点合并为基础,读取所有三角面片需要 $O(F)$ 时间,最终时间复杂度为 $O(F \times 3F/2) \sim O(F^2)$ 。

1) 改进算法。

本文作者在文献[2]曾提出一种改进算法,主要改进之处是借助一个 AVL 树来降低顶点查找的复杂度。每读入一个三角面片,首先对它的 3 个顶点在 AVL 树中查找相同的顶点,若找不到,则建立新的顶点节点,将该节点的地址插入到 AVL 树中,调整树的节点使其保持平衡。AVL 树起辅助查找作用,树的每个节点只存放顶点链表节点的地址,并不存放顶点的实际坐标数据,所以 AVL 树是虚的存储结构。在 AVL 树中查找的效率取决于树的高度,AVL 树的高度为 $\log F$,所以借助 AVL 树查找完成的顶点合并需要 $O(\log F)$,整个拓扑重建时间复杂度为 $O(F \log F)$ 。

2) 本文算法。

本文算法以 SMF 格式文件为基础,SMF 文件模型数据的布局有这样的特点:顶点坐标数据描述在前,三角面片数据描述在后,多个顶点按自然顺序从前往后进行编号,编号从 1 开始。基于上面的特点,算法在打开 SMF 文件之后首先建立一个动态的顶点结构数组(格式见图 2),然后把文件中的顶点坐标数据依次读入到对应的顶点结构中。

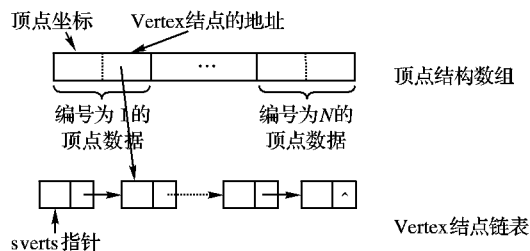


图 2 实现顶点快速合并示意图

接下来读取三角面片时每建立一个 Face 节点都要检查对应的 3 个顶点是否已经存在,即顶点合并。本文算法利用数组可以进行随机定位的特点,将读入的顶点编号作为顶点结构数组的下标直接定位到对应顶点的数组元素上,查看数组元素的 Vertex 节点地址域是否为空,不为空说明该顶点的 Vertex 节点已经存在,直接取出该数据进行后续拓扑关系设置,为空说明该顶点的 Vertex 节点尚未创建,创建新的 Vertex 节点并把节点地址回送到顶点结构数组对应数组元素的 Vertex 节点地址域中。

在进行顶点合并的过程中涉及到查找定位,由于将顶点编号作为数组下标直接定位,所以时间复杂度是 $O(1)$,整个重建的时间复杂度为 $3 \times O(F \times 1)$,即 $O(F)$ 。顶点结构数组

是一个临时存放顶点坐标数据和方便查找定位的数组,在使用完之后可以释放其所占的内存。

3 实例测试

本文在1台P4 2.80 GHz CPU,512 MB内存的PC上对改进算法^[2]和本文算法进行了实现,并测试了几个SMF文件的实例,测试结果在表1中列出。

表1 测试实例

测试例子名称	三角面片数	改进算法用时/s	本文算法用时/s
Cow	5 804	0.098	0.051
阀体	23 470	0.458	0.237
轮毂	50 940	1.017	0.511
Bonny	69 451	1.375	0.687
Horse	96 966	2.079	0.951
dragon	108 588	2.154	0.987

4 结语

从测试数据来看,由于改进了辅助查找索引方式,即从AVL树改为数组,所以拓扑重建的时间复杂度也从 $O(n \log n)$ 直接降为 $O(n)$,时间大为缩短,较大模型的效果尤其明显。

模型有了拓扑关系之后,对模型的后续处理工作变得相对容易,在缺陷修复和网格简化中利用拓扑关系可以直接找到邻接三角面片,无需遍历或借助辅助结构查找。现在可以说拓扑重建的时间复杂度不能再降低了,因为从数据文件中读取三角面片本身就需要 $O(n)$ 时间。

参考文献:

- [1] BECHET E, CUILIERE J-C, TROCHU F. Generation of a finite element MESH from stereolithography (STL) files[J]. Computer Aided Design, 2002, 34(1): 1-17.
- [2] 刘金义,侯宝明. STL格式实体的快速拓扑重建[J]. 工程图学报, 2003, 24(4): 34-39.
- [3] 张必强,邢渊,阮雪榆. 面向网格简化的STL拓扑信息快速重建算法[J]. 上海交通大学学报, 2004, 38(1): 39-42.
- [4] 戴宁,廖文和,陈春美. STL数据快速拓扑重建关键算法[J]. 计算机辅助设计与图形学学报, 2005, 11(11): 2447-245.
- [5] 张翔,廖文和,程筱胜,等. STL格式文件的拓扑重建方法研究[J]. 机械科学与技术, 2004, 24(9): 1093-1096.
- [6] 侯宝明,刘雪娜. STL实体模型的拓扑重建及其缺陷修复[J]. 计算机工程, 2005, 31(3): 213-214, 217.
- [7] DAI CHUNXIANG, JIANG YING, HU QINGXI, et al. Efficient topological reconstruction for medical model based on mesh simplification [M]. Berlin: Springer-Verlag, 2007: 526-535.
- [8] 邱元庆,周惠群,朱珊珊,等. 利用散列对STL文件进行拓扑重建和修复[J]. 机械科学与技术, 2009, 28(6): 795-802.
- [9] BURKARDT J. SMF format description [EB/OL]. [2010-05-20]. <http://people.sc.fsu.edu/~burkardt/data/smf/smf.txt>.
- [10] MANTYLA M. An introduction to solid modeling [M]. New York: W H Freeman & Co, 1988: 119-132.
- [11] WEILER K. Edge-based data structures for solid modeling in curved-surface environments [J]. IEEE Computer Graphics and Applications, 1985, 5(1): 21-40.
- [12] 严蔚敏,吴伟民. 数据结构: C语言版[M]. 北京: 清华大学出版社, 1998: 227-238.

(上接第2966页)

的反复“筛选”上。对深度为 h 的堆,筛选算法中进行关键字比较次数至多为 $2(h-1)$ 次,则在建含 n 个元素、深度为 h 的堆时,由于第 k 层上的节点数至多为 $2^{h-k}-1$,以它们为根的二叉树的深度为 $h-k+1$,则调用 $\lfloor \frac{n}{2} \rfloor$ 次HeapAdjust(堆排序的筛选算法,见文献[7]282页的算法10.10),该算法执行时进行的关键字比较次数为:

$$\sum_{i=h-1}^1 2^{i-1}(h-i) = \sum_{i=h-1}^1 2^i(h-i) = \sum_{j=1}^{h-1} 2^{h-j} \cdot j \leq 2n \sum_{j=1}^{h-1} j/2^j \leq 4n$$

又因为含有 n 个节点的完全二叉树的深度为:

$$\lfloor \lg n \rfloor + 1$$

则调整建新堆时调用HeapAdjust过程 $n-1$ 次,总共进行比较的次数不超过 $2n(\lfloor \lg n \rfloor)$,即:

$$2(\lfloor \lg(n-1) \rfloor + \lfloor \lg(n-2) \rfloor + \cdots + \lfloor \lg 2 \rfloor) < 2n(\lfloor \lg n \rfloor)$$

因此,堆排序在最坏的情况下,其算法执行的时间复杂度才为 $O(n \log n)$,显然比前几种方法的查找效率都高。

3 结语

综上所述,将数据元素期望被查找的先验概率作为建立哈希大顶堆时插入节点的顺序依据,查找过程与原哈希查找方法相比在不增加其他消耗的同时降低了冲突时执行查询的查找长度,从而使查询响应时间更短。但该方法对记录数 n 较

小的文件并不值得提倡,仅对 n 较大的文件还是很有效的。因此,对于记录数 n 较大的场合该方法有很好的应用价值。

参考文献:

- [1] KUMAR S, CROWLEY P. Segmented Hash: An efficient Hash table implementation for high performance networking subsystems [C]// Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems. New York: ACM Press, 2005: 91-103.
- [2] 王果,徐仁佐. 结合哈希过滤的一种改进多连接查询优化算法[J]. 计算机工程, 2004, 30(7): 57-59.
- [3] 张科. 多次Hash快速分词算法[J]. 计算机工程与设计, 2007, 28(7): 1716-1718.
- [4] 完美哈希函数[EB/OL]. [2010-06-27]. <http://blog.csdn.net/chixinmuzi/archive/2007/08/05/1727195.aspx>.
- [5] BOTELHO F C, KOHAYAKAWA Y, ZIVIANI N. A practical minimal perfect Hashing method [EB/OL]. [2010-06-28]. <http://homepages.dcc.ufmg.br/~nivio/papers/wea05.pdf>.
- [6] BOTELHO F C, PACH R, ZIVIANI N. Simple and space-efficient minimal perfect Hash functions [EB/OL]. [2010-06-28]. <http://homepages.dcc.ufmg.br/~nivio/papers/wads07.pdf>.
- [7] 严蔚敏,吴伟. 民数据结构: C语言版[M]. 北京: 清华大学出版社, 2006: 251-262.
- [8] 马如林,蒋华,张庆霞. 一种哈希表快速查找的改进方法[J]. 计算机工程与科学, 2008, 30(9): 66-68.
- [9] 周伟明. 多任务下的数据结构与算法[M]. 武汉: 华中科技大学出版社, 2006.