基于边收缩的快速网格简化算法

孟 军,宋 磊
MENG Jun,SONG Lei

大连理工大学 计算机科学与工程系,辽宁 大连 116023
Department of Computer Science and Engineering,Dalian University of Technology,Dalian,Liaoning 116023,China
E-mail:mengjun@dlut.edu.cn

MENG Jun,SONG Lei.Fast mesh simplification algorithm based on edge-collapse.Computer Engineering and Applications,2007,43(20):62-64.

Abstract: A fast mesh simplification algorithm based on Garland's QEM is introduced.Vertex weight is presented to express the significance of a vertex.Vertex weight can restrict the region affected by edge collapse in a small area.A priority queue containing the weight of all vertices is utilized to control the sequence of edge collapse.This algorithm is easy to be implemented and its cost of time is much lower.

Key words: mesh simplification;edge-collapse;priority queue;quadric error

摘 要:根据 Garland 的 QEM 算法提出了一种快速的网格模型简化算法。算法使用顶点权值来表示顶点的重要程度,顶点权值可以将收缩的边所影响的范围控制在较小的区域内; 顶点的权值被存储在一个优先权队列中并且利用优先权队列来控制边收缩的顺序,顶点的优先权队列所存储的元素比较少并且易于维护。该算法实现容易、执行速度快。

关键词:网格简化;边收缩;优先权队列;二次误差
文章编号:1002-8331(2007)20-0062-03 **文献标识码**:A **中图分类号**:TP391

1 引言

在基于计算机图形技术的场景绘制中,往往需要对大量的图形元素(图元),如顶点、边、多边形面等进行包括纹理、光照、裁减等复杂的运算。如果要进行这些运算的三维模型含有较少的图元显然可以提高图形处理系统的效率,所以如何在保证模型精度的情况下尽量减少三维模型的图元数量一直是一个值得研究的问题。一般情况下,采用网格这样一种简单直观的表示形式来表示一个三维模型,对三维模型的简化也就转化为对网格模型的简化。三角形网格是最为普遍的一种网格形式,因为其它的多边形网格均可以通过对多边形面片三角化而转化为三角形网格, 所以利用三角形网格来说明问题不失一般性。本文提到的网格模型如无特殊说明均指三角形网格模型。

网格模型的简化实际上就是一种数据压缩的方法,它是在试图保持外形不变或者变化很小的情况下尽可能减少详细模型的多边形数量。在实时应用中,通常需要这个过程来减少存储的顶点数量并将这些顶点发送到图形管线,以此来提高系统的实时性。对于网格模型的简化已经有很多基本的方法。包括顶点聚类、近平面结合、顶点删除、边删除、三角形删除、边收缩和三角形收缩等方法。在上述方法中,三角形收缩本质上与边收缩是一样的。收缩型的方法已经得到广泛的应用,这是因为收缩型的简化方法有利于构成过渡的多个 LOD 表示模型,便于多分辨率模型的管理,与删除型的方法相比,收缩型的方法将模型简化后不产生空洞,省去了费时的补洞操作,所以得到了非常广泛的应用。但是现存的简化算法主要将精力集中在简化的精度上而忽视了简化的效率,在一般的应用系统中,模型简化是一个离线处理的过程,在不同的视觉条件下加载不同的层次模型。当然也可以通过基于视点的模型简化来生成连续的 LOD 模型,但即便如此,在人机交互过程中要实时浏览虚拟场景时距离生成流畅的画面还有不小的差距。

基于实时应用的需求,特别是需要大量人机交互的应用环境中对算法效率的要求,本文在著名的 Garland 的二次误差简化算法 QEM 基础上提出一种计算简单、实现容易同时又尽量保证精度的简化算法,称之为快速网格模型简化算法 FMS。

在本文提出的网格模型简化算法中,首先针对模型的每一个顶点计算出它的曲率和所有邻接边的长度,进而确定该顶点的权值,然后将顶点的权值存储在一个优先权队列中,并对该队列进行从小到大的堆排序。在简化时每次从堆顶取出最小的权值,并且找到该值对应的顶点,在该顶点的邻接边集合中寻找符合条件的边,然后执行收缩操作。

2 二次误差简化算法

边收缩算法在每一次简化操作中,边是被简化掉的基本图形元素。边收缩操作如图 1 所示,在边(i,j)收缩后形成一个新的顶点 k,同时以(i,j)为公共边的两个三角形(如图中阴影所示)被删除,顶点 i 和 j 的邻接三角形成为新顶点 k 的邻接三角形。

基于边收缩的简化算法已经有一些比较通用且成熟的算法,比如 QEM 就是一种典型的基于边收缩的算法[1,2]。这个算法

作者简介:孟军,副教授,主要研究方向为数据库技术,数据挖掘,虚拟现实技术;宋磊,硕士研究生,主要研究方向为数字图像处理。

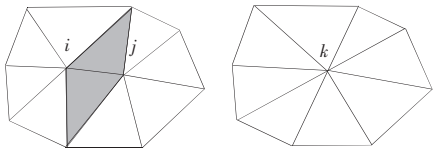

图 1　边收缩

将每个顶点都看作是一组三角形平面的交点,顶点的误差定义为该点到这组平面的平方距离的和,这种误差定义可以比较好的保证模型的整体形态。

QEM 算法首先为每一个顶点 $v=[v_x\ v_y\ v_z\ 1]^T$ 设一个 4×4 的对称矩阵 $\boldsymbol{Q}$ 称为误差矩阵,顶点 v 的误差定义为 $\Delta(v)=v^T\boldsymbol{Q}v$。假设边$(v_1,v_2)$收缩成顶点 v_3,那么 v_3 的误差矩阵就可以定义为 $Q_3=Q_1+Q_2$。为了完成边的收缩,必须计算出新顶点 v_3 的位置,一种简单的策略是在 v_1,v_2 和$(v_1+v_2)/2$ 中选择一个可以使得 $\Delta(v_3)$最小的值。为了得到更好的效果,可以计算出使得 $\Delta(v_3)$取最小值 v_3 的的位置坐标,这就要通过解方程如式(1):

$$\frac{\partial\Delta}{\partial x}=\frac{\partial\Delta}{\partial y}=\frac{\partial\Delta}{\partial z}=0 \tag{1}$$

来求得该坐标,上述方程可以变换为式(2):

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v_3=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

假设方程中的矩阵是可逆的,又得到式(3):

$$v_3=\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

如果矩阵不可逆,QEM 算法会试图沿着边(v_1,v_2)找到一个最优化的顶点,如果这样仍旧失败,那么就要在 v_1,v_2 和$(v_1+v_2)/2$ 中寻找一个合适的位置。为了得到误差矩阵 $\boldsymbol{Q}$,设 $p=[a\ b\ c\ d]^T$ 表示了一个平面:$ax+by+cz=d$,而且 $a^2+b^2+c^2=1$。因此有式(4):

$$\Delta(v)=\Delta([v_x\ v_y\ v_z\ 1]^T)=\sum_{p\in planes(v)}(p^Tv)^2 \tag{4}$$

其中 $planes(v)$表示所有以 v 为顶点的三角形集合,式(4)变形为式(5):

$$\Delta(v)=\sum_{p\in planes(v)}(v^Tp)(p^Tv)=\sum_{p\in planes(v)}v^T(pp^T)v=v^T(\sum_{p\in planes(v)}K_p)v \tag{5}$$

其中 $K_p=pp^T=\begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$,得到:

$$Q=\sum_{p\in planes(v)}K_p$$

QEM 算法的执行过程是:

(1)对所有的顶点计算 Q;

(2)选择合适的顶点对;

(3)为每一个顶点对计算最佳的收缩目标顶点;

(4)将所有的顶点对按照收缩代价由小到大的顺序放到一个堆中;

(5)从堆顶迭代地取出顶点对并完成收缩操作,同时更新相关顶点对的收缩代价并且更新堆;

(6)如果满足结束条件,结束;否则,转到(5)。

很多算法其误差度量都是采用了 Garland 的二次误差简化算法中的度量方式,但是在一些数据结构上做出过改变,在 Antônio W.Vieira 和 Luiz Velho 等人提出的快速星形算法中[3],将大多数简化算法都有的优先权队列舍弃,采用了一种随机选择顶点集的方法,每一步简化操作都从待简化模型中随机选择一个顶点集合,然后将该集合中权值最小的顶点简化掉,但是这种算法的前提是简化后的模型的顶点数要远远小于原模型的顶点数,只有这样才能保证随机顶点集合中不全是最后应该保留的顶点。因为这一限制,该算法就不适合用来对一些较为简单的模型进行简化。Chang Ha Lee 和 Amitabh Varshney 等人提出一种基于网格特征(mesh saliency)的简化算法[4],这个算法在二次误差估计的基础上引入了一个称为特征权值的参数,那些特征权值越大的顶点越能反映模型的特征,这样的顶点要尽量保留。

上面所描述的几个算法的误差度量都是基于二次误差度量的,二次误差度量已经被很多简化算法采用,尽管这种误差度量方式已经大大提高了简化效率,但是在对实时性要求很高的应用中采用这种误差度量方式来计算一个顶点的简化误差还是比较费时的。在某些实时应用中,比如刑侦现场重现、虚拟会议等,所应用的模型本身数据量比较小,往往只具有几百个或几千个最多几万个顶点,但是为了提高对大量模型渲染的效率,仍然需要对场景中的模型进行简化,这种情况下如果采用传统算法进行模型简化,相对于简化掉的图形元素来说,简化过程所耗费的时间太多了,特别是在图形硬件配置比较低的环境下,这个问题更为突出。

3　快速网格模型简化算法

本文提出的快速网格简化算法 FMS 在执行过程上与 QEM 算法类似,但是在收缩代价的定义和计算上有很大差别。为了提高计算速度,FMS 算法舍弃了顶点的对称矩阵 $\boldsymbol{Q}$,并且用一个称为顶点权值的数值来表示一个顶点的重要性,权值越大的顶点就越重要,对应于 QEM 中的收缩代价就越大。同时 FMS 中用来控制收缩顺序的堆存储的元素是顶点的权值,这样相对于 QEM 以顶点对为存储对象,其中的元素减少了约 2/3。

3.1　顶点权值

为了能在一个迭代的简化过程中选择合适的边来进行收缩,需要知道每条边的收缩代价,在基于二次误差度量的算法中,为了计算收缩代价,首先要得到每个顶点的误差,为此每一个顶点都要先计算一个误差矩阵,然后再计算顶点误差,最后计算边的收缩代价。

为了能在误差计算中降低计算量,快速简化算法 FMS 舍弃了顶点的误差和误差矩阵,而采用了顶点权值来描述一个顶点的重要程度,对于顶点权值,仅考虑了曲率和边长因素。如果一个顶点的曲率越大说明该顶点的位置越能表现模型的形体特征,如果一个顶点所邻接的边的长度越长说明该顶点在模型表面上影响的区域越广。所以在综合考虑了曲率和边长后,顶点权值可以较好地反映模型上一个顶点的重要程度。

为了得到顶点的曲率,首先要计算顶点的法线向量 $\boldsymbol{n}_v$,在由三角形数据构成的曲面上计算某点的法向量最简单的方法

是计算每个三角形的法线向量,然后将相邻三角形的法线进行平均,对于相邻三角形的共同顶点使用平均法线。要计算三角形的法向量,可以取三个顶点 v_1、v_2 和 v_3,然后按式(6)计算向量积:

$$\boldsymbol{n}=\begin{bmatrix}x_1-x_2\\y_1-y_2\\z_1-z_2\end{bmatrix}\times\begin{bmatrix}x_2-x_3\\y_2-y_3\\z_2-z_3\end{bmatrix} \tag{6}$$

$\boldsymbol{n}_v$ 可以利用式(7)计算

$$\boldsymbol{n}_v=\sum_{f\in v.faces}\boldsymbol{n}_f \tag{7}$$

其中 $v.faces$ 表示顶点 v 的所有邻接三角形集合,三角形面 f 的单位法向量 $\boldsymbol{n}_f=n/\Lambda n|$,一般情况下也会将 $\boldsymbol{n}_v$ 归一化。

得到顶点的法向量后,可以通过式(8)计算顶点曲率:

$$\boldsymbol{c}_v=\frac{\sum_{f\in v.faces}\angle(\boldsymbol{n}_v,\boldsymbol{n}_f)}{|v.faces|} \tag{8}$$

其中 $\angle(\boldsymbol{n}_v,\boldsymbol{n}_f)$ 表示向量 $\boldsymbol{n}_v$ 与 $\boldsymbol{n}_f$ 的夹角,$|v.faces|$ 表示顶点 v 的邻接三角形个数,称 $\boldsymbol{c}_v$ 为顶点的相对曲率值,顶点相对曲率值能代表顶点的几何位置的重要性:相对曲率值越大,顶点的位置就越重要。

边 (v,u) 的收缩代价 $E(v,u)$ 可以利用式(9)计算:

$$E(v,u)=l_{vu}^2\times[1+\frac{1}{2}(c_v+c_u)] \tag{9}$$

其中 l_{vu}^2 是边 (v,u) 的边长的平方。

一个顶点的权值通过式(10)计算:

$$\varepsilon_v=\min_{i\in v.ertices}E(v,i)+\alpha\lambda \tag{10}$$

其中 $v.vertices$ 是顶点 v 的邻接顶点集合,λ 是收缩后顶点 v 的权值与其初始权值之差的绝对值,α 是一个可调参数,在第一次收缩前 $\lambda=0$,引入 λ 的目的是防止在某一区域内过度的简化。ε_v 表示一个顶点的几何位置的重要性,这种定义形式可以将边收缩所影响的区域控制在收缩形成的新顶点的邻接三角形区域内。

在选择要收缩的边时,本文算法先从优先权队列中取出堆顶的权值所对应的顶点,然后在其邻接顶点集合内找到满足式(10)的边进行收缩操作。

3.2 优先权队列

优先权队列用来控制边收缩的顺序,在传统的边收缩算法中所使用的优先权队列中所存放的元素是边、三角形或者是满足一定收缩条件的顶点对,如果优先权队列中的元素越少,那么对队列的维护所花费的时间和空间代价就越小,所以在优先权队列中最好存放网格模型中数量最少的图形元素。网格模型的图形元素包括顶点、边和三角形,根据欧拉定理可以证明,一个具有 v 个顶点的三角形网格,约有 $2v$ 个三角形和 $3v$ 条边,所以本文算法采用了顶点的优先权队列。为了避免在更新队列时改变顶点在顶点序列中的索引,优先权队列中并没有直接存放顶点而是存放了顶点的权值。执行简化操作时从队列中取出最小的权值,然后找到它所对应的顶点,进而选择合适的边进行简化。

为了每次都能从队列中容易的取出最小权值需要对队列排序,和 QEM 算法一样本文也是采用了堆排序的方法,在第一次简化之前要将所有的权值都进行排序时间复杂度为 $O(n\log n)$,每次在边收缩后调整堆的时间复杂度是 $O(\log n)$。

3.3 边收缩

假设要收缩的边是 (v_1,v_2),这条边收缩后将形成新的顶点 v_3,理论上 v_3 的位置应该使得它的权值最小,但是这样无疑将增加计算的复杂性,一个折衷的方案是如果 v_1 和 v_2 中有边界顶点,那么新顶点的位置取边界顶点的位置,否则就取 $(v_1+v_2)/2$ 的位置,边界顶点在网格预处理的过程中已经做好标记,这个选择新顶点位置的策略是影响简化精度的,但是可以大幅度的提高效率。

产生新顶点 v_3 后,首先删除 v_1 和 v_2 的公共三角形并将其余的三角形都作为 v_3 的邻接三角形,然后要计算新顶点 v_3 的权值,还要重新计算 v_3 的所有邻接顶点的权值。

3.4 FMS 的执行过程

FMS 算法的执行过程与 QEM 算法基本上是一致的,执行步骤如下:

(1)计算所有顶点的权值 ε_v;

(2)将优先权队列中的顶点权值由小到大堆排序;

(3)找到堆顶的权值所对应的顶点并在该顶点的邻域内找到满足式(10)的边;

(4)进行边收缩操作;

(5)重新计算收缩形成的新顶点邻域内的权值,并对优先权队列进行调整;

(6)如果满足了简化的约束条件则停止,否则转到(3)继续。

4 实验结果

在同一硬件环境下,对 QEM 算法和本文的 FMS 算法进行了比较,实验采用 Garland 的个人网站提供的模型,如图 2所示。

图 2 三个网格模型的线框表示

通过对比表 1 中各模型的图形元素数量,可以看出这些模型符合前文中提到的根据欧拉定理所得到的关于各图形元素数量关系的结论,也可以将它们看作对这个结论的印证。本文在程序实现中参考了 Horst Birthelmer 等人提出的网格表示形式,这种表示形式有利于对网格的简化操作[5],针对 FMS 算法又增加了顶点权值的结构。

表 1 图 2 中各模型的三角形数、边数和顶点数

模型	三角形个数	边数	顶点数
bunny	69 451	104 288	35 947
bones	4 204	6 306	2 154
cow	5 804	8 706	2 904

4.1 简化精度比较

通过对上述模型的简化,首先比较一下两个算法的简化精度。图 3 和图 4 表示了分别对 bunny、bones 和 cow 模型采用两个算法进行简化的情况,简化后 bunny 模型有 9 589 个三角形,bones 模型有 2 204 个三角形,cow 模型有 2 804 个三角形。两种算法在精度上的差别很小。QEM 算法简化后的模型比 FMS 简化后的模型三角形的分布更加均匀。

通过图 3 和图 4 来比较 QEM 算法和 FMS 算法在简化精度上的差别,可以看出在模型的整体形态上差别并不大,如果仅从模型显示的角度来看,这种差别可以忽略。

4.2 简化效率的比较

FMS 算法在损失一定精度的情况下,在简化效率上有了较

(下转 116 页)