视点相关且拓扑可变的多分辨网格动态构造算法

陈 鹏 高 宇 吴玲达
(国防科技大学多媒体研究与开发中心,长沙 410073)

摘 要 随着近几年 3 维扫描和图形建模技术的快速发展,3 维模型的数据量不断增大,其在存储、显示及传输上都面临巨大的挑战,因此,必须构造模型的简化表示。通过对当前网格模型动态简化算法的分析,提出了一种网格简化算法来构造拓扑可变的网格模型累进表示,在此基础上,通过对简化后的模型数据进行再组织,为简化模型建立了一种紧凑、灵活的动态多分辨率结构,并相应地给出了基于视点的动态简化算法。理论分析和实验结果表明,新方法能够随着视点参数的变化动态生成适当细节的简化模型,简化结果好,简化后的模型不仅能够较好地保留原模型的基本几何形状,而且能够较好地保留原始模型的颜色等属性特征,具有存储量小、适用范围广和自适应性强等特点。

关键词 视点相关 拓扑简化 多分辨率 网格简化
中图法分类号:TP391 **文献标识码**:A **文章编号**:1006-8961(2009)01-0161-08

Algorithm for Dynamically Constructing View Dependent and Topology Alterable Multiresolution Mesh

CHEN Peng, GAO Yu, WU Ling-da
(*Multimedia Technology Research Center of National University of Defense and Technology, Changsha* 410073)

Abstract With the rapid development of recent 3D scanning and graphic modeling technique, the size of 3D model data increases drastically, which brings many challenges in its storage, display and transfers. Therefore a simplistic representation of 3D model data is in great demand. First current algorithms of dynamic mesh simplification were analyzed, and then an algorithm for mesh simplification was designed to construct topology alterable progressive mesh representation of 3D model. Based on this progressive mesh, a view dependent, compact and dynamic multiresolution structure was constructed by reorganizing the simplified model data. Finally a view dependent dynamic simplifying algorithm was presented. Theory analysis and experiment result showed that this novelty method can generate proper detailed simplified model based on view parameters, which achieved good simplification effect preserving both its geometry shape, and its attribute characters such as color etc. The presented algorithm can be widely used in many applications because of its small storage requirement, low cost and high adaptability.

Keywords view dependent, topological simplification, multiresolution, mesh simplification

1 引 言

随着 3 维扫描和图形建模技术的快速发展,3 维模型的数据量不断增大。如 Stanford 大学 Digital Michelangelo Projectv 采集的几尊大型雕像模型,所包含的三角形数量高达数十亿之多,这些庞大的模型数据在存储、显示、传输上都面临巨大挑战。解决办法之一就是网格模型简化,即用尽可能少的数据逼近原始模型。而在简化过程中为了保持模型表示方式的紧凑,需要在绘制过程中根据某一准则(如视点相关等)生成相关层次的简化模型。Hoppe 最

收稿日期:2007-01-29;**改回日期**:2007-06-30

第一作者简介:陈鹏(1977 ~),男。国防科技大学系统工程专业博士研究生。主要研究方向为虚拟现实技术。E-mail:pengchen@nudt.edu.cn

早提出了累进网络(PM)[1]的概念,在此基础上,将累进网格的顶点以自顶向下的方式组织成层次结构,根据视域四棱锥、表面法向和屏幕空间误差等条件,从顶点层次结构选取合适的顶点以及相应的 vsplit 或 ecol 算子,动态更新场景中的三角形网格模型[2]。类似的顶点层次树结构在近年来的多分辨率建模方法中被广泛应用[3-4]。这种树形结构的构造简单,在网格简化的同时就可以完成。但是由于将边收缩后的顶点看成是一个完全不同的顶点,因此,最后生成的树形结构中的节点数大约是原始模型顶点数的两倍,增大了模型的存储开销。Lindstrom 等人提出的高度场场景层次细节实时连续绘制算法,同样使用了这种树结构[4]。Xia 等人提出了一种实时的基于视点的三角网格模型简化算法[5]。De Floriani 等人提出了多分辨率三角剖分算法(MT)[6],在绘制时搜索效率也较高。但是三角网格的局部更新都是固定的,因此 LOD(level of details)分布变化的自由度比顶点层次树结构差。Pajarola 等人提出的 FastMesh,是一种基于视点的 LOD 生成和网格动态简化框架[7],采用半边数据结构和半边折叠操作,通过构造半边折叠层次结构,建立了一种紧凑、高效的多分辨率数据结构,和顶点层次结构相比,半边折叠层次结构减少了模型的存储开销,提高了动态简化的搜索效率。冯洁设计了一种基于 PM 的网格简化方法[8],并构建了一种新的多分辨率数据结构——点分裂记录树,通过模拟注视点区域的视觉模型,使模型表面的 LOD 以观察者的注视点为中心连续分布。文献[9]利用 Delaunay 三角剖分来限制候选顶点对的数目,他们认为所有“虚拟边”的集合是网格顶点的 Delaunay 边的一个子集,并通过以网格顶点为结点的 3 维 Voronoi 图来得到候选“虚拟边”,但是这种方法计算量很大,运行速度较慢。

上述算法在简化的过程中都不能改变模型的拓扑结构。Luebke 等人采用八叉树对空间进行划分,提出了一种视点相关的网格动态简化框架[10],能够根据需要改变模型的拓扑形状。但是这种方法基于特定的空间分割方法,因而难以控制拓扑简化的效果。El-Sana 等人通过构造基于一般顶点对收缩的视点依赖树,实现了视点相关的几何和拓扑简化[9],同时他们还研究了在视点依赖算法中产生条带三角形的办法[11]。Borodin 等人对顶点对收缩进行扩展[12],采用一般对收缩操作,能够有效地控制模型边界的拓扑简化,消除网格边界的缝隙。但是一般对收缩无法用统一的数据结构表示,难以构建视点相关的多分辨率层次结构。

综合上述各种算法的优点,本文设计了一种视点相关且拓扑结构可变的网格动态简化方法。基于基本的顶点对操作构造网格模型的累进表示,然后根据视点参数动态生成视点相关的简化网格,以达到实时的效果。

2 拓扑可变的网格简化

对于由多个不同部分组成或是带有很多孔洞的模型,保留一些不重要的拓扑细节将可能占用过多的三角形数目。如果在简化的过程中支持拓扑结构的改变将有利于进行大幅度简化;且在大幅度模型简化时,改变拓扑结构所带来的视觉误差往往比拓扑保持的简化方法还要小。为此,可基于一般顶点对收缩操作,设计一种拓扑结构可变的网格简化算法。

2.1 顶点对收缩

一般的顶点对收缩操作允许任意两个顶点进行合并而无需考虑它们是否在一条边上,因而可能合并模型中并不相连的区域,比边折叠具有更大的普遍性,边折叠操作只是顶点对收缩的一种特例。对于一个包含 n 个顶点的三角形网格,其顶点对共有 $n(n-1)/2$ 个,为了避免确定候选顶点对时的 $O(n^2)$ 搜索时间 $O(n^2)$ 为时间复杂度,采用一种自适应的距离阈值 ε 方法,其取值不是固定的,随着简化过程不断增加。为获取 ε 的初始值,假设已知三角形网格模型的包围盒为 $([x_{\min},y_{\min},z_{\min}],[x_{\max},y_{\max},z_{\max}])$,且顶点是均匀分布的,将包围盒均匀划分成边长为 ε 的正方体,则有如下关系:

$$\frac{(x_{\max}-x_{\min})(y_{\max}-y_{\min})(z_{\max}-z_{\min})}{\varepsilon^3}=n$$

$$\Rightarrow \varepsilon=\sqrt[3]{\frac{(x_{\max}-x_{\min})(y_{\max}-y_{\min})(z_{\max}-z_{\min})}{n}} \tag{1}$$

在确定某个顶点相关的候选顶点对时,只需要检查与该顶点位于同一立方体或是与其相邻的立方体内的顶点。为了防止很小的 ε 使得空间划分后立方体的数量过大造成很大的内存开销,使用一个比顶点个数大的哈希表来存储每个顶点的相关信息,表中每一项仅存储一个指针。采用链地址法处理哈

希冲突。对于插入哈希表的每一个新的顶点,如果出现哈希冲突而且冲突导致哈希表线性链表的元素个数超过一个预先给定的阈值(本文设为 10),认为初始的 ε 不合法,将当前的 ε 减半,重新构建哈希表。空间距离小于 ε 的候选顶点对称为局部顶点对,否则称为全局顶点对。如果剩余的候选顶点对收缩产生的误差均大于 ε 或者已经没有剩余的局部顶点对,将当前的 ε 翻倍,然后重新为剩余顶点建立新的哈希表,并将剩余的顶点插入到新的哈希表中。

2.2　候选顶点对的选择及排序

3 维模型通常不仅包含几何信息,还包括表面属性如颜色、纹理坐标和法向等信息。表面属性的处理对模型简化的效果有着重要影响。分别为各属性特征建立独立的二次误差矩阵[13-15]来控制属性误差,并将其与几何误差相结合,反映每个候选顶点对收缩的误差大小。其中几何误差采用面积加权的二次误差矩阵方法。对于属性(颜色、纹理坐标和法向)误差,类似于几何误差的度量,其中颜色的 RGB 分量分别对应 x,y,z 3 个坐标轴,以点到平面的距离来考虑简化操作对顶点周围区域属性值变化的影响,将二次误差度量应用到属性误差的计算当中。

由于顶点几何坐标的范围是 $(-\infty,+\infty)$,而颜色分量值的范围是[0,1],在 RGB 彩色空间中两个颜色点的欧氏距离最大为 $\sqrt{3}$;而纹理坐标是一个 2 维向量,其取值范围为[0, 1],在纹理坐标空间中两个纹理坐标的欧氏距离最大为 $\sqrt{2}$;法向量空间是一个单位球,两个法向量之间的欧氏距离最大应该为 2。为了防止过大的几何误差使其他属性误差失去作用,将顶点 $\bar{v}$ 的总误差定义为

$$E(\bar{v}) = E_g(\bar{v})\left(1+\frac{\alpha}{\sqrt{3}}E_c(\bar{v})+\frac{\beta}{\sqrt{2}}E_t(\bar{v})+\frac{\gamma}{2}E_n(\bar{v})\right) \tag{2}$$

式中,E_g、E_c、E_t 和 E_n 分别表示几何误差、颜色误差、纹理误差和法向误差,系数 α、β、γ 用于调节属性误差对总误差的影响,以达到不同的简化效果。例如,要更好地保留颜色特征,就可以适当地增大 α。特别地,如果三角形网格的顶点不包含某个属性,只需将对应系数取为 0 即可。

确定了顶点的误差以后,选择使顶点对收缩的几何误差最小的那个顶点作为新顶点的位置。对于新顶点属性的计算,采用类似于 Erikson 的方法[16],通过对三角形 3 个顶点的属性进行插值得到新顶点的属性值。

使用一个最小堆来对所有的候选顶点对进行排序。每执行一个顶点对收缩操作,就重新计算包含了新顶点的候选顶点对的误差。由于顶点对收缩操作同时改变了与新顶点相邻的三角形面片,因此那些包含了与新顶点相邻的顶点的候选顶点对也需要进行调整。但是在大多数情况下,这些顶点对的误差的变化影响很小,为避免每次重新计算误差,减少堆中元素更新调整的计算,为这些顶点对设置一个“Dirty”标记,而只有当标记为“Dirty”的顶点对成为堆中的最小元素时,才重新计算它的误差,然后再插入到最小堆中。

2.3　算法流程与累进网格表示

根据上面的讨论,拓扑可变的网格简化算法步骤描述如下:

(1)读入初始网格模型,标记出模型的几何和属性特征边,计算每个顶点的几何和属性二次误差矩阵以及相邻三角形的总面积;

(2)确定初始的距离阈值 ε,为网格顶点建立哈希表,确定候选顶点对,计算每个候选顶点对的误差和收缩后新顶点的位置,然后按照误差进行最小堆排序;

(3)如果剩余候选顶点对产生的误差大于 ε 或者当前没有剩余的局部顶点对,那么将 ε 翻倍并重新为剩余顶点建立新哈希表,再重新计算候选顶点对的误差;

(4)如果最小堆中误差最小的候选顶点对被标记为“Dirty”,那么重新计算它的误差,再将其插入到最小堆中;

(5)从最小堆中找出误差最小的候选顶点对,删除两个收缩的顶点,计算新顶点的几何和属性二次误差矩阵,删除退化的三角形和引起重合的三角形,更新哈希表,找出包含新顶点的候选顶点对,并计算其误差;

(6)找出需要更新的顶点对,将需要更新的顶点对标记为“Dirty”;

(7)如果顶点对收缩引入的误差达到一定阈值或者剩余三角形数目已达到用户的要求,则退出;否则转步骤(3)。

为得到连续的多分辨率简化模型,在简化过程中把每个顶点对收缩操作都记录下来。最终除了得到原始网格的简化模型外,还得到一个顶点对收缩操作序列,这个序列的逆序对应着简化

网格执行顶点对收缩的逆操作——顶点对分裂操作,通过顶点对收缩和分裂就可以得到原始网格和简化网格之间任意细节的简化模型,即累进网格表示。

为了能够在动态简化的过程中正确完成顶点对收缩或分裂操作,在每个顶点对收缩操作记录中存储了以下的信息:①顶点对收缩中被删除顶点的索引;②新顶点的坐标增量 $\Delta v = v_i - \bar{v}$;③顶点对收缩中删除的三角形的集合;④新顶点的属性增量(如果需要)。

3 视点相关的多分辨率结构

通过拓扑可变的网格简化算法,完成了对三角形网格模型的简化,并生成了具有连续多分辨率的累进网格表示,但还不能随着视点变化自适应地调节模型不同部分的细节。下面通过对简化模型进行再组织,将累进网格序列组织成一种视点相关的动态多分辨率结构,使得在场景绘制时,能够根据视点的变化对网格模型进行自适应调整,动态生成场景的细节模型。

3.1 动态层次结构的构造

根据简化操作之间的依赖关系,将相互关联、相互依赖的点对收缩操作按照执行的先后顺序进行排序,将线性的简化操作序列组织成一个树形的层次结构,从而能够有效地发掘不同简化操作之间的独立性并维护它们之间的相关性。如图 1 所示,对于一个顶点对收缩操作 $(v_i, v_j) \rightarrow \bar{v}$,定义它的父节点是对顶点 $\bar{v}$ 进行进一步收缩的操作,而它的两个子节点分别是收缩产生顶点 v_i(左子节点)和 v_j(右子节点)的两个操作。由于将收缩后的新顶点保存在顶点 v_i 中,所以父节点实际上就是在顶点对收缩操作序列中顺序在它之后且对 v_i 进行进一步收缩的最近的操作,而两个子节点则分别是顺序在它之前且收缩产生 v_i 和 v_j 的两个最近操作。

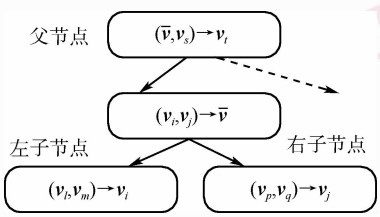


图 1 顶点对收缩操作层次中的父子关系

Fig. 1 Paternity in the level of vertex-pair shrinking

在构造的树形层次结构中,每个节点就是一个顶点对收缩操作,与 Hoppe[2] 和 El-Sana[9] 等所采用的顶点树结构相比,由于顶点树结构的叶节点实际上并未包含点对收缩或分裂所需的任何信息,因此,本文的节点数目大约只是顶点树结构的一半,从而有效地减少了搜索和存储的开销。

假设顶点对收缩操作序列为 VC,每个顶点对收缩操作记录对应的父节点指针为 father,两个子节点指针为 left 和 right,由顶点对收缩操作 $(v_i, v_j) \rightarrow v$ 序列构造层次树的算法描述为:

(1)按照逆序从右至左遍历 VC,对每个顶点对收缩操作 $VC[i]$,执行步骤(2) ~ 步骤(5);

(2)从 $i-1$ 开始按逆序遍历 VC,对每个 $VC[j]$,执行步骤(3) ~ 步骤(5);

(3)若 $VC[j].v = VC[i].v_i$ 且 LeftNodeOK = FALSE,那么 $VC[j]$.father := i,$VC[i]$.left := j,LeftNodeOK := TRUE,转步骤(5);

(4)若 $VC[j].v = VC[i].v_j$ 且 RightNodeOK = FALSE,那么 $VC[j]$.father := i,$VC[i]$.right := j,RightNodeOK := TRUE,转步骤(5);

(5)若 LeftNodeOK = TRUE 且 RightNodeOK = TRUE,则 $i := i-1$,LeftNodeOK := FALSE,RightNodeOK := FALSE,转步骤(2);否则,$j := j-1$,转步骤(3);

上述算法是一个自顶向下的构造过程,图 2 给出了构造顶点对收缩操作层次树的一个示例。实际上,整个场景的层次结构通常是一个顶点对收缩操作树林,将所有根节点的索引保存在一个线性表 Root 中。在顶点对收缩操作树林中,相关的顶点对收缩操作都处于同一棵子树中,而不同子树中的顶点对收缩操作之间不存在依赖关系,可以独立执行而互不影响,这也就为基于视点的动态简化提供了条件。

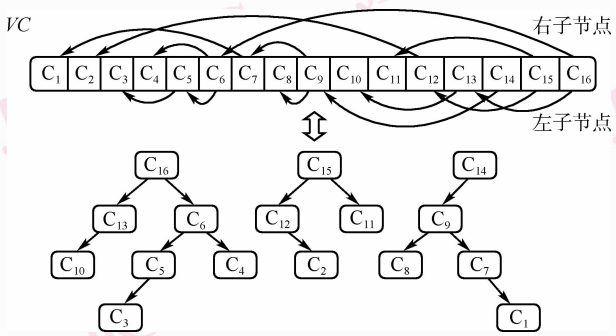


图 2 顶点对收缩操作层次树的构造过程

Fig. 2 Process of the construction for vertex-pair shrinking level tree

3.2　基于视点的网格动态简化

在顶点对收缩操作层次结构中,每个可能的简化模型都对应了其中的一个割集,以双向链表的形式记录当前割集,称为活动节点列表(以下简记为 anl),anl 中包含了所有恰好未被执行的顶点对收缩操作,也就是说,在 anl 下方(不包括 anl)的顶点对收缩操作均已被执行,而处在 anl 及其上方的节点所对应的顶点对收缩操作都未被执行。所以,如果 anl 越靠近叶节点,则执行的顶点对收缩操作越少,简化模型也越接近于原始模型;相反,如果 anl 越靠近层次的上方,则执行的顶点对收缩操作越多,得到的模型也越简单。基于视点的网格动态简化,实际上也就是根据视点的动态变化,对网格和 anl 进行动态更新的过程。

初始化时 anl 由所有的叶节点组成。在绘制时,通过执行适当的顶点对收缩或分裂操作,对当前的 anl 进行动态调整,就可以生成所需细节的简化模型。这里需要注意的是,因为得到的是一个顶点对收缩操作树林,所以有可能某棵子树的所有节点都不在 anl 中,也就是该子树所有节点的顶点对收缩操作均已执行。在这种情况下,将该子树的根节点标记为“active”,而在更新过程中首先遍历那些标记为“active”的根节点。整个基于视点的动态简化算法描述如下:

(1)遍历 Root 中所有的根节点,对于“active”根节点,如果它所对应的顶点对分裂操作应该被执行,则执行顶点对分裂操作,并将根节点插入到 anl 中;

(2)遍历活动节点列表 anl,对于其中的每一个节点,执行步骤(3)~步骤(4);

(3)如果当前节点的子节点对应的顶点对分裂操作应该被执行,则执行顶点对分裂操作,并将该子节点插入 anl 中,如果当前节点的两个子节点都已插入 anl 中,则将该节点从 anl 中删除;

(4)如果当前节点对应的顶点对收缩操作可以执行,则判断其子节点所对应的顶点对收缩操作是否都已执行,如果存在未执行的子节点,那么先执行子节点的顶点对收缩操作,然后再执行当前节点的顶点对收缩操作,对于执行的每一个顶点对收缩操作,将该节点从 anl 中删除,如果该节点存在父节点,则将其父节点插入 anl 中,否则,将该根节点标记为“active”;

在步骤(4)中可能会出现如图 3 所示的情况,假设当前节点为 C_{16},如果它所对应的顶点对收缩操作可以被执行,但是其子孙节点对应的顶点对收缩操作未执行,由于顶点对收缩操作之间存在依赖关系,因此只有递归地执行了节点 C_5、C_4 和 C_6 的顶点对收缩操作后,才可以执行当前节点的顶点对收缩操作。由于在网格简化的过程中使用了顶点映射机制,所以只需要保证父子节点之间的先后依赖关系,就可以保证最终简化网格拓扑结构的正确性。

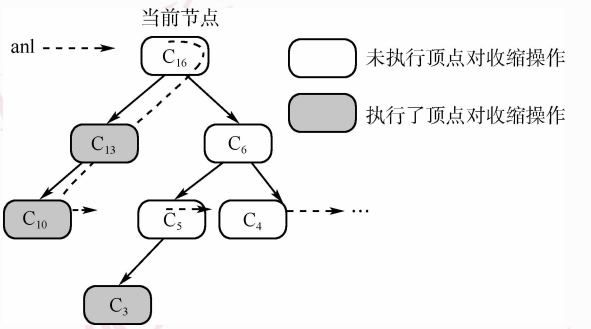


图 3　子节点对应的顶点对收缩操作未被执行

Fig. 3　The case of vertex-pair shrinking corresponding to child node not be executed

如何判断一个顶点对收缩或分裂操作是否应该被执行,不仅取决于物体的远近,还应当包括观察视域、可见性以及其他众多视觉因素。在文献[2]、[7]中,分别使用了一组类似的判断标准,包括屏幕投影误差准则、视域准则、背面准则、轮廓准则等。本文主要考虑 3 种判断标准,分别是屏幕投影误差准则、视域准则和背面准则。在预处理过程为每个节点计算一个包围球和一个法向锥,动态简化时,根据当前视点与节点包围球以及法向锥的关系,来确定每个节点的顶点对收缩或分裂操作是否需要被执行。

4　实验结果

采用前述算法,对几组模型进行了简化,并建立了对应的多分辨率结构。实验中计算机硬件配置为 PIV3.0GHz CPU、512M 内存、ATI RADEON X600 显卡,操作系统为 Windows XP SP2,编程工具为 Visual C++ 6.0 和 OpenGL。

程序运行分为 4 个部分:模型读入、网格简化、多分辨率结构的建立以及绘制,表 1 给出了实验模型的数据量、执行各部分的运行时间,表 2 给出了原始模型与多分辨率模型绘制一帧所需时间的比较,从结果可以看出,文中提出的算法明显加速了模型

的绘制过程,平均绘制时间表明了随着视点变化模型绘制的平均开销,而峰值时间表明了在简化程度高时的绘制开销。其中兔子、龙和佛像为来自斯坦福大学的 3 维扫描模型,船和机器人是由多个部件组成的复杂模型,而且包含了颜色属性,地形模型是利用 256×256 的 Lena 图像生成的地形网格。

表 1 拓扑可变的多分辨率结构建立的实验结果统计

Tab. 1 Statistics of the experiment result for topology alterable multiresolution mesh constructing

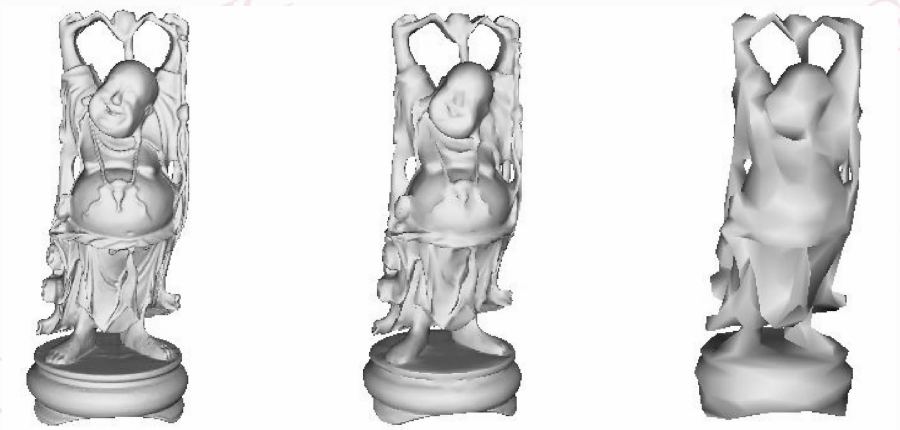
模型	原始网格		基网格		运行时间(s)		
	三角形数	顶点数	三角形数	顶点数	读入	网格简化	多分辨构造
Happy Buddha	1 087 716	543 652	1 088	531	39.727	19.268	10.879
Colony ship	229 774	129 345	4 596	2 760	8.783	2.964	1.178
Robotb9	238 656	73 783	2 386	679	9.481	3.225	1.436
Huge bunny	69 451	34 834	346	201	2.373	1.201	0.562
Terrain	130 050	65 536	1 300	699	4.136	2.163	1.229
Dragon	871 414	457 645	872	436	31.405	15. 342	8.463

表 2 拓扑可变的多分辨率结构模型绘制的实验结果统计

Tab. 2 Statistics of the experiment result for topology alterable multiresolution mesh model rendering

模型	绘制时间(s)					
	Happy Buddha	Colony ship	Robotb9	Huge bunny	Terrain	Dragon
原始模型	0.091	0.068	0.069	0.049	0.058	0.079
多分辨模型平均时间	0.041	0.029	0.031	0.022	0.025	0.032
多分辨模型峰值时间	0.014	0.019	0.018	0.011	0.017	0.014

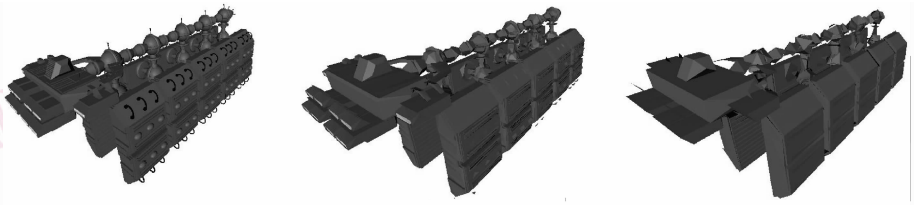
图 4 ~ 图 6 是原始模型在不同简化程度下分别得到的结果,即图中的简化模型 I 和 II。其中,图 4 是佛像模型在删除了大部分顶点和三角形面片后的简化结果,从图中可以看出,本文算法仍能较好地保留原模型基本的几何特征,当然简化厉害的时候几何特征损失的就多。图 5 和图 6 分别是由多个对象组成的船和机器人模型的实验结果,从实验结果可以看出,简化过程中合并了空间上不相连的对象,取得较好的简化结果,同时简化模型仍较好保持了原模型的颜色特征,说明算法对颜色等属性信息的处理是合理有效的。



(a) 原始模型(1087716 Tris) (b) 简化模型Ⅰ (108716 Tris) (c) 简化模型Ⅱ (1088 Tris)

图 4 佛像模型的简化结果比较

Fig. 4 Comparison of the result of Happy Buddha model simplification



(a) 原始模型(229774 Tris)　(b) 简化模型Ⅰ (11489 Tris)　(c) 简化模型Ⅱ (4596 Tris)

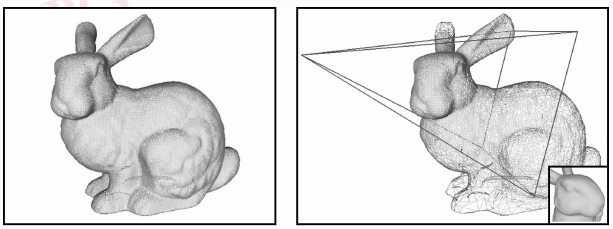
图 5　船模型的简化结果比较

Fig. 5　Comparison of the result of Colony ship model simplification

图 6　机器人模型的简化结果比较

Fig. 6　Comparison of the result of Robotb9 model simplification

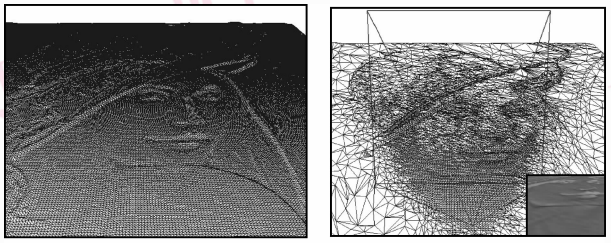
图 7 ~ 图 9 给出了几组利用构建的动态多分辨率结构进行基于视点的网格动态简化的实验结果。其中,图 7 ~ 图 9 各图中(a)都为原始网格,(b)都是在第 3 视点位置观察到的基于视点的网格简化结果,其中锥体部分显示了当前的视域大小,每个图的右下角是在当前视点下所观察到的结果。从图中可以看到,越是靠近视点的区域,模型表面的分辨率越高;而随着距离的增大,模型表面的分辨率逐渐下降;距离视点最远的部分分辨率最低。此外,位于视域之外的网格也是最低的分辨率。当视点变化时,三角形网格随之发生变化,这正是由于本文的网格顶点映射机制所实现的不同细节模型之间的快速转换,以及不同 LOD 模型连续的细节过渡。



(a) 原始网格 (69451 Tris)　(b) 简化结果 (15036 Tris)

图 7　兔子模型基于视点的动态简化效果

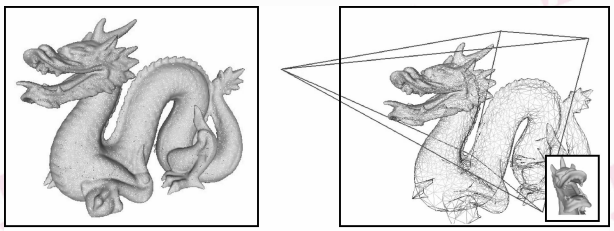
Fig. 7　View dependent dynamic simplification effect of Huge bunny model



(a) 原始网格 (130050 Tris)　(b) 简化结果 (17037 Tris)

图 8　地形模型基于视点的动态简化效果

Fig. 8　View dependent dynamic simplification effect of Terrain model



(a) 原始网格 (871414 Tris)　(b) 简化结果 (13268 Tris)

图 9　龙模型基于视点的动态简化效果

Fig. 9　View dependent dynamic simplification effect of dragon model

5　结　论

本文以一般顶点对收缩作为基本的简化操作,

设计了一种拓扑可变的网格简化算法,通过自适应地选择距离阈值来控制候选顶点对的数目,并按照几何和属性信息的统一误差来确定顶点对收缩的顺序。同时,基于顶点对收缩操作,将简化网格重新组织成一个基网格和一个顶点对收缩操作树林,树林中的父子关系就是顶点对收缩操作执行的先后依赖关系,为简化的网格模型建立了一种紧凑、灵活的动态多分辨率结构,并通过网格顶点映射机制实现不同场景细节模型之间的快速转换,相应地给出了基于视点的动态简化算法。

理论分析和实验结果表明,本文算法不仅能得到较好的简化效果,而且简化后的模型即能较好地保留基本的几何形状特征,也能较好地保留原始模型的颜色等属性特征。算法所构建的动态多分辨率结构可以实现基于视点的网格动态简化,三角形网格能够随视点运动动态更新,模型的表面细节呈连续分布。在减少存储开销的同时,扩大了适用范围,增强了适应性和灵活性。

参考文献(References)

1 Hoppe H. Progressive meshes [A]. In: Proceedings of the SIGGRAPH'96[C], New Orleans, USA, 1996:99-108.

2 Hoppe H. View-dependent refinement of progressive meshes[A]. In: Proceedings of the SIGGRAPH'97[C], Los Angeles, CA, USA, 1997:189-198.

3 Soetebier I, Birthelmer H, Sahm J, *et al*. Managing large progressive meshes[J]. Computers & Graphics, 2004, **28**(5):691-701.

4 Lindstrom P, Koller D, Ribarsky W, *et al*, Real-time, continuous level of detail rendering of height fields[A]. In: Proceedings of SIGGRAPH'96[C]. New York: ACM Press, 1996: 109-118.

5 Xia J, El-Sana J, Varshney A. Adaptive real-time level-of-detail-based rendering for polygonal models[J]. IEEE Transactions on Visualization and Computer Graphics, 1997, **3**(2):171-183.

6 De Floriani L, Magillo P, Puppo E. Efficient implementation of multi-triangulation[A]. In: Proceedings of IEEE Visualization[C], Research Triangle Park, North Carolina, USA, 1998:43-50.

7 Pajarola R, DeCoro C. Efficient implementation of real-time view-dependent multiresolution meshing [J]. IEEE Transactions on Visualization and Computer Graphics, 2004, **10**(3):353-368.

8 Feng Jie. Simplification and view-dependent LOD control for large 3D mesh models[J]. Journal of Computer-aided Design & Computer Graphics, 2006,**18**(2):186-193.[冯洁. 大型 3 维网格模型的简化及基于视点的 LOD 控制[J]. 计算机辅助设计与图形学学报, 2006,**18**(2):186-193.]

9 El-Sana J, Varshney A. Generalized view-dependent simplification [J]. Computer Graphics Forum, 1999, **18**(3): 83-94.

10 Luebke D, Erikson C, View-dependent simplification of arbitrary polygonal environments[A]. In: Proceedings of the SIGGRAPH'97 [C], Los Angeles, CA, USA,1997:199-208.

11 Isenburg M, Gumhold S. Out-of-core Compression for gigantic polygon meshes[J]. ACM Transactions on Graphics, 2003, **22**(3): 935-942.

12 Borodin P, Gumhold S, Guthe M, *et al*. High-quality simplification with generalized pair contractions [A]. In: Proceedings of International Conference on Computer Graphics & Vision [C], Moscow, 2003:147-154.

13 Garland M, Heckbert P. Surface simplification using quadric error metrics[A]. In: Proceedings of the SIGGRAPH'97[C], Los Angeles, CA, USA, 1997:209-216.

14 Garland M, Heckbert P. Simplifying Surfaces with Color and texture using quadric error metrics [A]. In: Proceedings of IEEE Visualization[C], California: CA, USA, 1998:263-270.

15 Lindstrom P. Out-of-core simplification of large polygonal models [A]. In: Proceedings of the SIGGRAPH'00[C], New Orleans, Louisiana, USA, 2000: 259-262.

16 Erikson C, Manocha D. GAPS: General and automatic polygonal simplification[A]. In: Proceedings of the SIGGRAPH'99[C], Los Angeles, USA: ACM Press, 1999: 79-88.