基于层次聚类树的点曲面视点相关绘制

徐爱国 黄 琦 孙守迁
(浙江大学计算机科学与技术学院, 杭州 310027)

摘 要 针对点曲面的视点相关绘制问题,提出了一个新的表面基层次聚类简化算法。区别于普遍采用的空间剖分基策略,该算法的显著优势在于能够运用法向锥半角误差标准有效跟踪曲面的起伏变化,并以此为聚类简化过程提供可靠的全局误差控制。离线简化阶段,连同各种预定义的聚类约束条件,算法构造了点曲面模型的连续层次多分辨率表达。实时绘制阶段,层次可见性裁剪以及优化的树遍历提高了系统的整体性能。此外,通过引入附加的轮廓增强机制,在较大的屏幕投影误差和较高的模型简化率情况下,系统仍然能够保证较好的绘制视觉质量。

关键词 模型简化 多层次细节 点基绘制 多分辨率
中图法分类号: TP391.72 **文献标识码**: A **文章编号**: 1006-8961(2007)02-0356-10

View-Dependent Point Set Rendering Based on Hierarchical Clustering Tree

XU Ai-guo, HUANG Qi, SUN Shou-qian
(*College of Computer Science and Technology, Zhejiang University, Hangzhou* 310027)

Abstract This paper proposes a novel algorithm for surface-based hierarchical clustering simplification that aims to accelerate view-dependent point set rendering. The remarkable advantage of this algorithm is that it uses a normal cone semi-angle metric to trace surface curvature variation and provides the clustering simplification process with a reliable global error control, rather than commonly used spatial partition scheme. In off-line simplification phase, combined with various predefined clustering constraint conditions the algorithm constructs a continuous multi-resolution hierarchical representation for point set model. In real-time rendering phase, hierarchical visibility culling and optimized tree traversal efficiently improve system integral performance. Moreover, an additional silhouette enhancement mechanism is introduced to ensure a well rendering vision quality in spite of a larger screen projection error and higher model reduction rates.

Keywords simplification, level of detail, point based rendering, multi-resolution

1 引 言

近年来,随着 3D 精确扫描获取技术的普及应用,大量几何模型不经过传统的曲面重建过程而直接表示为点的集合。这种点基表达(point based representation, PBR)的曲面形式受到了国内外学者广泛关注,它能够有效结合隐式曲面和参数曲面的一些特有优势,例如大幅度的几何变形和实体间的布尔操作摆脱了拓扑关系的严格约束等等。由此,相关点曲面绘制技术的研究也随之展开并取得了一系列重要成果[1,2]。采用椭圆加权平均(EWA)表面投射技术,每一个采样点在对象空间表示为一个有向椭圆盘,将其投影变换到图像空间并进行高斯重采样滤波,最终可以得到高质量的模型绘制效果。然而,以点作为绘制图元也存在其固有缺陷。相对于多边形网格,PBR 的复杂性与几何模型的形状复杂性无关[3],结果造成点图元数目庞大。此外,现

基金项目:国家重点基础研究发展规划"973"计划(2002CB312106);国家自然科学基金项目(60475025);浙江省科技计划(2004C33086)
收稿日期:2005-10-14;**改回日期**:2005-10-24

第一作者简介:徐爱国(1975~),男。浙江大学计算机科学与技术专业在读博士研究生。主要研究领域为计算机图形学、计算机动画、虚拟人技术等。

有的图形显示硬件也只针对多边形网格进行优化,点曲面绘制的大部分处理过程必须依靠软件计算。由于受到点图元数量的严重制约,CPU 往往成为绘制瓶颈而使系统无法提供满意的交互性能。

作为影响点曲面建模方法实用化的一个核心问题,目前针对绘制性能的研究工作正在不断深入。一方面围绕开发新一代图形硬件[4-6]的可编程能力实现硬件加速。另一方面,在实时交互式图形系统中引入模型简化、层次细节(levels of detail, LOD)以及多分辨率技术,根据给定的模型显示质量需求大幅度地减少点图元的数目,缩短系统处理时间来提高响应性能。本文从软件加速绘制的角度出发,给出了一个表面基聚类点曲面简化算法,进一步通过构造绘制图元的层次树实现模型的连续多分辨率表达和视点相关绘制。

本文的主要贡献:

(1) 法向锥半角简化误差测量　不同于文献[7]中以局部表面分析(local surface analysis)进行曲率估计实现自适应简化的方法,本文引入法向锥半角作为简化误差测量标准,结合表面基聚类,算法不但可以对结果模型提供可靠的全局误差控制,而且能够有效地克服现有空间剖分基层次聚类算法中存在的一些不足。

(2) 优化聚类节点的生成　算法增加了有效的约束条件来引导聚类种子点的选择,这对于保持良好的聚类形状、提高聚类分布的均匀性以及降低层次聚类树的深度都是非常重要的。另外,通过优化生成聚类节点的空间位置坐标,使相邻椭圆盘之间的冗余覆盖程度降低,进而减少了绘制阶段不必要的计算量。

(3) 轮廓增强机制　实时绘制阶段,节点法向锥参数用于检测模型的轮廓,并在相应区域分配更多的点图元来提高绘制质量。该机制也可以使系统在保证同等模型显示质量情况下,增大屏幕投影误差阈值来获取更大的模型动态简化比率。

2　前人的工作

模型简化、多分辨率表达以及视点相关绘制技术一直是计算机图形学领域的研究热点。近期,随着人们对点曲面建模方法的日益关注,与此相关的研究工作也陆续得到展开。

文献[7]中,Pauly 等人针对点采样几何模型简化问题提出了多种策略。他们实现了曲率自适应聚类简化,但由于存在局部表面分析条件和采样密度下限的约束,该方法很难进一步实现多分辨率绘制所必须的层次数据结构。他们的层次聚类方法基于二元空间剖分,处理过程不能保证得到良好的聚类区域形状。迭代简化虽然提供了最小的估计误差,但计算量大,而且二叉树的深度和树节点记录信息的内存消耗等等局限性也成为制约该方法得到实际应用的重要因素。最后,粒子模拟方法同样存在计算多分辨率表示的低效问题,另外,对于一个动态 LOD 绘制系统,在粒子模拟简化框架下如何实现两个不同细节层次之间的平滑过渡也需要作进一步探讨。

文献[8]中,Linsen 通过连续采样点删除实现点云简化。Alexa 等人[9]提出了类似的算法。这两种方法生成的简化模型是原始点集的准确采样子集,其结果在多分辨率传输方面存在较大优势。然而,对于视点相关的绘制环境,他们在层次树的组织和遍历效率方面都存在明显不足。

Moenning 等人[10]给出了一个均匀分布、特征敏感且密度可控的点云简化算法,其核心思想源于测地 Voronoi 图和快速匹配最远点采样。在文献[11]中,该算法被进一步扩展到处理非均匀分布点集和包含孔洞的几何模型。他们强调算法内在支持构造多分辨率点曲面表达和高效 LOD 视点相关绘制,但文献中没有关于这方面技术的详细论述。

最近,Wu 等人[12]以优化子采样方式简化高密度点集,并生成其在对象空间无孔洞的离散圆或椭圆盘表达。该方法实现了结果模型的最大误差限定,而且子采样点集的分布均匀。尽管由此可以得到模型一系列不同分辨率的离散简化版本,但如何进一步构造用于实时绘制的高效连续的点集 LOD 表达仍然是待解决的问题。

文献[13]、[14]中的研究工作与本文的相关度较大。Pajarola[13]提出了一个针对大规模点曲面层次表达和绘制的高效生成算法。然而,这种点八叉树层次构造策略基于自适应空间剖分,无法有效保证良好的聚类形状和避免简化过程中存在不合理聚类的问题。尤其对于非均匀点集,这些缺陷将导致较差的模型简化质量以及视点相关绘制时错误的可见性裁剪。Szymon 等人[14]给出的实时递进绘制系统构架称为 Qsplat,与文献[13]类似,他们的方法采用层次包围球,同样无法克服前面提到的纯粹空间

基划分构造带来的一些问题。

一般来说,构造给定点曲面模型的连续 LOD 表示需要综合考虑两方面因素:一是简化算法本身对结果模型质量的影响;二是数据结构支持实时绘制执行的效率。基于以上分析,本文提出了一种表面基聚类简化算法。

3 聚类简化算法

顶点聚类算法在网格简化中得到了广泛的应用,它具有内存和计算需求小、效率高的特点。普遍作法是剖分模型的包围盒为均匀的栅格 cell,将落入同一个 cell 中的所有点用一个公共代表点来表示。然而,这种空间剖分基的方法在处理点曲面时存在几个方面的问题。一是完全依赖于 3D 欧拉距离测量,会很容易将曲面两个不相干的部分聚类连接,简化质量比较差,特别是对简化比率相对较大的模型;二是相对于公共代表点的位置,同一 cell 中的聚类点通常并非该点的最近相邻点集,其严重受限于不同的栅格形状。在随后的绘制阶段,模型轮廓和高曲率区域会出现明显的简化视觉缺陷;三是非均匀的采样点分布带来栅格分辨率选取的困难。细粒度的固定尺寸剖分会造成过多的不必要的 cell,内存和计算消耗过大。自适应空间剖分,例如八叉树结构,则容易形成较大的栅格,使不相关曲面聚类连接的问题更为突出,因而也无法有效克服这一缺陷。

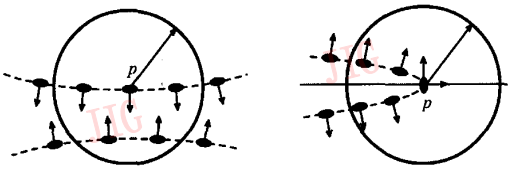
典型的视点相关绘制需要简化原始模型并通过构造层次树来提供一个连续多分辨率的模型表达,实时遍历该树,进而根据视点的不同,选择当前合适的 LOD 进行绘制。本文借助点曲面层次聚类实现这一机制,考虑到空间剖分策略本身存在的不足,采用表面基方法对模型进行简化,进一步完成相应层次数据结构的构造。

3.1 曲面表示

为算法叙述方便,先引入一些相关点曲面表示的概念。设点集 $\boldsymbol{P}=\{p_i\}$ 是对 3D 几何模型光滑二流形表面 S 的充足采样,$\boldsymbol{P}$ 的属性可以表示为一个三元组 (C,N,E),这里 $\boldsymbol{C},\boldsymbol{N}$ 和 $\boldsymbol{E}$ 分别对应采样点空间位置坐标 c_i、法向量 $\boldsymbol{n}_i$ 和椭圆盘 $\boldsymbol{e}_i(u_i,v_i,r_i,s_i)$ 的集合。椭圆盘 $\boldsymbol{e}_i$ 与 $\boldsymbol{n}_i$ 正交,其中心位于点 c_i,并由两个归一化正交切向量 $\boldsymbol{u}_i$ 和 $\boldsymbol{v}_i$ 定义主、辅轴方向,r_i 和 s_i 为椭圆盘 $\boldsymbol{e}_i$ 的主轴长和主辅轴比率,所有椭圆盘在对象空间形成一个表面 S 的无孔洞的表达。关于 $\boldsymbol{e}_i$,进一步定义通过其所在平面的 2D 局部坐标系 LCS_i,原点为 c_i。对于每一个采样点 p_i,基于 3D 欧拉距离计算 k 个最近相邻点的集合,用 $N_k(p_i)$ 表示。本文主要处理视点相关绘制并假设所有曲面采样点已包含属性信息 $\boldsymbol{n}_i$ 和 $\boldsymbol{e}_i$,对于如何由纯粹点集计算这些属性信息,有多种方法可以选择,详细内容参见文献[12]、[15]、[16]。

3.2 聚类简化过程中存在的问题

尽管可以假设原始点集有充足的采样密度能够满足局部表面分析的条件,但在整个聚类简化过程中仍然会遇到一些问题。首先,k-邻域查询球包含曲面两个不相关的部分从而形成不合理的聚类(图 1(a))。如前所述,这种情况很容易在基于空间剖分聚类简化方法中出现,并且无法以自适应策略有效避免。其次,曲面曲率过大和数据点采样密度不足导致错误的局部表面分析结果(图 1(b)),异常的切平面和法向量估计将给后续的简化操作和绘制带来不利的影响。



(a) 不合理聚类 (b) 错误的局部表面分析

图 1 不合理聚类和错误的局部表面分析

Fig. 1 Unreasonable clustering and wrong local surface analysis

通常,对于一个采样点,错误包含在其 k-邻域查询球中的聚类点之间的法方向差异较大。因此,第一个问题可以通过法向量一致性检测来解决。对于第二个问题,需要用其他的方法表征模型表面曲率而不能简单采用文献[7]类似的策略。

3.3 几何简化误差测量

定义误差测量标准是模型简化需要解决的一个基本问题。目前,多数算法采用几何基元的空间距离测量例如点到面的距离,相应的计算方式通常依赖于模型局部表面的有效估计或是确定性的几何拓扑信息。点曲面没有显式可用的拓扑信息,而且聚类简化过程本身也不能保证采样点均匀分布和局部表面分析方法有效,采用上述测量标准显然不合适。为此,在本文聚类简化算法中,引入了一种新的标准——法向锥误差测量。

该误差标准基于如下分析:平面或曲面低曲率区

域采样点的法方向具有高度的一致性,相反,高曲率区域则存在较大的差异。给定一个初始均匀光滑二流形点曲面模型,可以由聚类区域采样点之间的法方向差异估计局部表面曲率变化。确切地说,以聚类法向锥半角值 θ_n 作为误差标准,算法能够实现点曲面模型的曲率自适应简化。文献[7]中,Pauly 等人通过局部表面分析引入表面差异 σ_n 的概念,他们的想法基于曲面任何时候都具备充足采样密度的假设,但一个迭代聚类简化过程通常无法满足该条件。

定义 θ、V 分别为法向锥半角及其轴向量,设 m 为法向锥的度,即该法向锥所属的聚类包含原始采样点的数目。最初,每一个原始模型采样点设置的法向锥参数为 $\theta_i=0$ 和 $V_i=n_i$(图 2)。当代表新点 p_{cluster} 由 k 个最近邻域点 $\{p_1,p_2,\cdots,p_k\}$ 聚类生成时,按以下各式计算 p_{cluster} 的法向锥参数:

$$\begin{aligned} m_{\text{cluster}} &= k \\ V_{\text{cluster}} &= \frac{1}{m_{\text{cluster}}}\sum_{i=1}^{k} n_i \\ \theta_{\text{cluster}} &= \max\left(\operatorname{acos}\left(\frac{V_{\text{cluster}}\cdot n_i}{\|V_{\text{cluster}}\|\|n_i\|}\right)\right) \end{aligned} \tag{1}$$

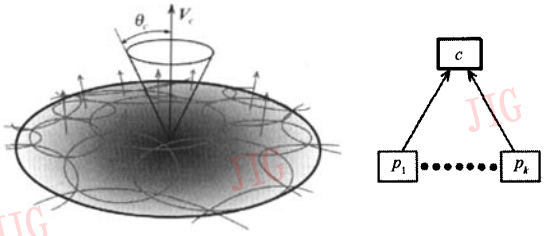


图 2　附带法向锥的新聚类生成

Fig. 2　Generation of a new cluster with normal cone

进一步,考虑采样点 p_j 被加入一个存在的聚类 cl 形成新聚类 $cl+1$,得到新聚类法向锥:

$$\begin{aligned} m_{cl+1} &= m_{cl}+1 \\ V_{cl+1} &= \frac{1}{m_{cl+1}}(m_{cl}\cdot n_{cl}+n_j) \\ \theta_{cl+1} &= \begin{cases} \theta_{cl} & \text{如果 } \operatorname{acos}\left(\dfrac{V_{cl+1}\cdot V_{cl}}{\|V_{cl+1}\|\|V_{cl}\|}\right)+ \\ & \theta_{\text{cluster}} \le \operatorname{acos}\left(\dfrac{V_{cl+1}\cdot n_j}{\|V_{cl+1}\|\|V_j\|}\right) \\ \operatorname{acos}\left(\dfrac{V_{cl+1}\cdot V_j}{\|V_{cl+1}\|\|n_j\|}\right) & \text{其他} \end{cases} \end{aligned} \tag{2}$$

即跟踪所有聚类采样点法向量与锥轴的最大夹角来确定当前的聚类误差。

最后,设 α、β 为两个现有聚类,它们形成一个新的聚类 γ,计算如下:

$$\begin{aligned} m_\gamma &= m_\alpha + m_\beta \\ \gamma &= \frac{1}{m_\gamma}(m_\alpha\cdot n_\alpha + m_\beta\cdot n_\beta) \\ \theta_\gamma &= \max(\theta'_\alpha,\theta'_\beta) \end{aligned} \tag{3}$$

其中,$\theta'_\alpha=\operatorname{acos}\left(\dfrac{V_\gamma\cdot V_\alpha}{\|V_\gamma\|\|V_\alpha\|}\right)+\theta_\alpha$

$$\theta'_\beta=\operatorname{acos}\left(\frac{V_\gamma\cdot V_\beta}{\|V_\gamma\|\|V_\beta\|}\right)+\theta_\beta$$

用户定义误差阈值 ε,由下面的不等式来决定每一步聚类操作:

$$\theta_{\text{cluster}} \le \varepsilon$$

由此可见,所有聚类法向锥的半角都不会超过给定的阈值 ε,这意味着算法能够为最终的模型简化结果提供一个全局可控的误差上限。

另外,对于 3.2 节提到的聚类问题,只要选取合适的 ε 值例如 $\pi/6$,法方向背离的采样点聚类操作就可以被有效避免。同时,利用法向锥半角跟踪表面曲率变化,不需要进行局部表面分析[7]。

3.4　形状保持

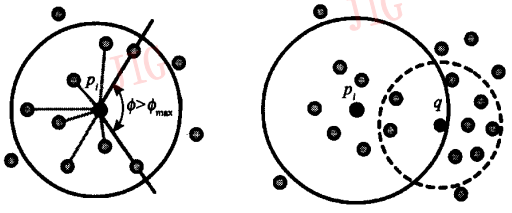
如图 2 所示,为保证在对象空间无孔洞的模型表达,每一次聚类生成新点的椭圆盘将覆盖所有该次操作被聚类点。显然,良好的聚类形状能够有效降低相邻盘之间的冗余覆盖程度,进一步减少绘制计算量,因此,聚类种子点的选取至关重要。在本文的算法实现中,定义了两个约束检测条件来优化聚类种子点的选取策略。

设 p_i 是当前处理的采样点,在其成为一个有效种子点之前,需要进行如下测试:

(1) 最近邻域形状　将 p_i 的所有 k-邻域点沿 n_i 投影到 LCS_i 坐标平面生成一个 2D 投影点集,转换该点集到以 p_i 为原点的极坐标系并按极角排序。如果所有相邻投影点之间的极夹角不大于给定的形状保持因子 $\phi_{\max}$,那么 p_i 为一个可能的聚类种子点;否则,p_i 是一个无效的种子点(图 3(a))。

(2) 最近邻域相关性　如果有一个采样点 q 在 p_i 的 k-邻域中而 p_i 本身却不在 q 的邻域中,即二者不满足 k-互相邻关系[17],那么 p_i 不是一个有效的种子点(图 3(b))。

以上两个约束检测条件使得聚类种子点的选取更为合理,这有助于改善后续采样点分布的均匀性



(a) 最近邻域形状测试 (b) 最近邻域相关性

图 3 聚类形状保持的约束性条件

Fig. 3 Constraint conditions of cluster shape preservation

和生成聚类区域的形状。

3.5 聚类节点生成

结合 3.3 节定义的简化误差测量标准,给出表面基聚类节点的生成过程如下。

试探区域生长 对于一个候选种子点 p_i,首先计算它的 k_{max}-邻域 $N_{k_{max}}(p_i)$,k_{max} 为用户定义的最大聚类尺寸。随后操作包含 5 个步骤:

(1) 生成一个只包含点 p_i 的虚拟聚类 $Cl_{virtual}$。

(2) 选择当前最近邻点 p_j,同时要求 $p_j \in N_{k_{max}}(p_j)$ 和 $p_j \notin Cl_{virtual}$,如果 p_j 满足以下两个聚类必要条件则将其并入 $Cl_{virtual}$:

① 关于 p_j,点 p_i 通过 3.4 节的最近邻域相关性测试;

② 将 p_j 并入 $Cl_{virtual}$,重新计算的法向锥半角 $\theta_{virtual}$ 小于用户定义的误差阈值 ε。

否则转向步骤 4。

(3) 如果聚类 $Cl_{virtual}$ 中点的数目达到 k_{max},转向步骤 5;否则,返回步骤 2。

(4) $Cl_{virtual}$ 中的采样点数目大于等于 k_{min}(用户定义的最小聚类尺寸),转向步骤 5;否则,删除 $Cl_{virtual}$ 并中止点 p_i 相关的聚类操作。

(5) $Cl_{virtual}$ 中的所有采样点通过 3.4 节的最近邻域形状测试,将 $Cl_{virtual}$ 转换为一个真实聚类 Cl_{real},进一步计算聚类参数;否则,删除 $Cl_{virtual}$ 并中止点 p_i 相关的聚类操作。

聚类参数计算 对于新聚类点,需要计算其法向量、位置坐标、椭圆盘参数以及聚类包围球半径。法向量 $\boldsymbol{n}_{cluster}$ 为归一化的法向锥轴 $\boldsymbol{V}_{cluster}$,对于位置坐标 $\boldsymbol{c}_{cluster}$ 和椭圆盘参数 $e_{cluster}$,过程如下:

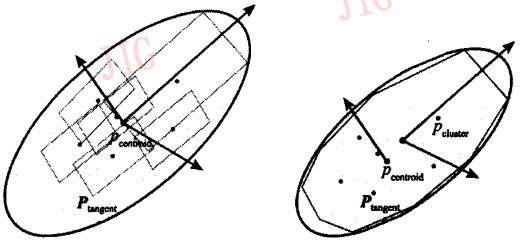
首先计算所有聚类点的平均位置

$$\bar{\boldsymbol{c}} = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{c}_i$$

这里 k 为实际的聚类尺寸。对每个被聚类点的椭圆盘 e_i,由其主副轴生成一个有向包围矩形。在切平面 $P_{tangent}(\boldsymbol{x}):(\boldsymbol{x}-\bar{\boldsymbol{c}})\cdot\boldsymbol{n}_{cluster}=0$ 上,以 $\bar{\boldsymbol{c}}$ 为原点,连同该平面上两个归一化的正交向量定义一个 2D 坐标系统,进一步计算该坐标系到 3D 世界坐标系的变换矩阵 $\boldsymbol{M}=\boldsymbol{TR}$,其中,$\boldsymbol{T}$ 和 $\boldsymbol{R}$ 分别为平移和旋转矩阵。将所有有向包围矩形的角点 $c_i^j(j=0,1,2,3)$ 沿着法方向 $\boldsymbol{n}_{cluster}$ 投影到切平面 $\boldsymbol{P}_{tangent}$,并得到局部坐标系下的一系列投影坐标。由这些投影点的集合,计算一个子空间内的 2D 凸壳多边形和它的重心 $\boldsymbol{c}_{centroid2D}$,重心点经下式变换即得到新聚类点的 3D 空间位置坐标:

$$\boldsymbol{c}_{cluster} = \boldsymbol{M}\cdot\boldsymbol{c}_{centroid2D}$$

如图 4(b)所示,由凸壳多边形的顶点和重心 $\boldsymbol{c}_{centroid2D}$,采用齐次协方差基方法[13]在 2D 子空间求得覆盖所有被聚类点的椭圆盘 $\boldsymbol{e}_{cluster}$,其主辅轴 $\boldsymbol{u}_{cluster}$ 和 $\boldsymbol{v}_{cluster}$ 经矩阵 $\boldsymbol{M}$ 变换到 3D 世界坐标系。对比直接采用所有聚类点有向包围矩形角点的计算方式(见图 4(a)),在对象空间能得到相邻聚类椭圆盘覆盖冗余更少的优化结果,这一点对于减少处理时间、加速绘制以及提高简化质量都是非常重要的。



(a) (b)

图 4 聚类椭圆盘参数的优化计算

Fig. 4 Optimal computation for cluster elliptical splat parameters

最后,由下式计算聚类包围球半径

$$R_{cluster} = \max(\|\boldsymbol{c}_{cluster} - \boldsymbol{c}_i^j\|)$$

3.6 聚类树构造

作为一个离线处理过程,视点相关树通过迭代聚类操作以自下向上的方式构造。用户先设置聚类简化参数如误差阈值 ε、形状保持因子 ϕ_{max} 和最大最小聚类尺寸等等,然后所有采样点 p_i 初始化为节点描述 $Node_i$ 并加入到循环链表 L。反复执行遍历操作直到 L 中不再包含满足聚类条件的节点或一次循环后 L 节点数目的变化率小于预定义阈值,例如 1%,下面给出的是节点遍历和链表更新的实现

细节。

设 $Node_{current}$ 为当前节点。如果 $Node_{current}$ 不能完成 3.5 节提到的聚类过程,直接将链表指针移动到下一个节点;否则,执行如下几个步骤:

(1) 生成聚类节点 $Node_{new}$,在 $Node_{new}$ 与 $Node_{current}$ 以及其他相关节点之间建立父子关系。

(2) 将 $Node_{new}$ 插入 L。

(3) 移动链表指针到下一个节点,并保证其在 L 中不是 $Node_{new}$ 的子节点。

(4) 将 $Node_{new}$ 的所有子节点移出 L。

在整个聚类过程中有新采样点的生成,算法采用动态网格数据结构[18]进行 k-最近邻域查询。聚类树构造完毕后,所有节点被分为 4 类,即

根节点——有子无父的节点

内部节点——同时具有父子的节点

叶节点——有父无子节点

游离节点——没有任何父和子的节点

部分原始采样点没有参与任何聚类操作,称为游离节点,其数目依赖于几何模型的复杂度以及用户定义的简化参数。为了保证聚类椭圆盘有较好的形状,算法没有采用文献[7]的策略将这些节点并入相邻的聚类,因此,得到的层次聚类树结构是一个森林(树的集合),原始点集包含所有叶节点和游离节点对应曲面模型的最精细版本。

4　层次点基绘制

利用层次聚类树进行视点相关绘制,必须考虑一些附加信息如相机参数等等。如文献[19]、[20]所述,实时绘制系统要求引入几个重要的视点相关 LOD 选择标准。另外,良好的数据组织结构以及高效的树遍历方式也能够有效减少处理时间。

4.1　视点相关 LOD 选择标准

可见性裁剪:裁剪对当前模型显示没有任何贡献的点可以有效提高系统的绘制性能,本文算法的视棱锥裁剪充分利用了节点聚类树层次结构的特性。给定一个测试节点 $Node_{test}$,其包围球包含所有后代节点,如果 $Node_{test}$ 的包围球与视棱锥不相交,则该节点的所有后代节点不需要更进一步的测试。另外,与视线背离的椭圆盘不需要绘制,可以通过计算节点法向量与视方向的角度关系执行背面剔除,类似于视域四棱锥裁剪,层次聚类树结构同样能够保证该操作具有较高的效率。

屏幕空间投影误差:当点曲面模型远离视点,一些节点椭圆盘的屏幕投影面积可能小于用户定义的阈值 τ,它们将被各自对应的父节点聚类代替或转换可见性状态。设 e_i 为节点 $Node_i$ 的椭圆盘,屏幕空间投影标准可公式化为

$$\cos(\gamma-\theta_i)\cdot\frac{\pi\|\boldsymbol{u}_i\|\|\boldsymbol{v}_i\|r_i^2}{|v-c_i|^2\cdot s_i}d^2<\tau \tag{4}$$

这里 d 为视点 v 到投影平面的距离,γ 是视线与法向量 $\boldsymbol{n}_i$ 的夹角[19]。

轮廓增强:几何模型的边界轮廓携带了大量的形状信息因而视觉敏感,视点相关绘制可以通过检测和增强点曲面的轮廓来提高模型显示质量。设节点的位置为 c,法向量为 $\boldsymbol{n}$,法向锥半角为 θ,视点 v 到 c 的向量为 $\boldsymbol{vc}$。如果不等式(5)成立,则判定该节点为模型轮廓的一部分,即轮廓节点。

$$\left|\arccos\left(\frac{\boldsymbol{n}\overrightarrow{vc}}{\|\boldsymbol{n}\|\|\overrightarrow{vc}\|}\right)-\pi/2\right|<\theta_{threshold} \tag{5}$$

式中,$\theta_{threshold}$ 为用户定义的轮廓检测阈值。

对于一个轮廓节点,定义增强因子 κ(取值从 0 到 1)按比例缩小对其进行测试的屏幕投影误差阈值。于是,与其他区域相比较,模型的轮廓区域可以保证有更高的点绘制密度。

4.2　层次聚类树遍历

系统通过遍历层次聚类树来动态重建 LOD 模型用于实时绘制,为有效执行这一过程,引入一个双向链表——活动链表。该链表的元素包含所有可见节点、部分不可见的游离节点和根节点,并以此反映点曲面模型当前的绘制状态。

如图 5 所示,测试活动链表节点进而实现聚类树的快速遍历,两个互逆操作 EXPAND 和 CONTRACT 负责链表的动态更新。

设 $Node_{active}$ 为活动链表中当前处理的节点,具体步骤如下:

(1) 确定 $Node_{active}$ 的类型。如果为根节点则转向步骤 3;如果为游离节点则转向步骤 4;其他节点

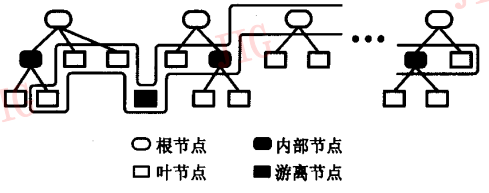


图 5　层次聚类树的遍历和绘制状态

Fig. 5　Status for the traversal and rendering of hierarchical cluster tree

类型继续；

（2）测试 $Node_{active}$ 的父节点 $Node_{parent}$。如果 $Node_{parent}$ 需要简化，执行一个 CONTRACT 操作：将 $Node_{parent}$ 插入活动链表，并移出 $Node_{parent}$ 的所有后代节点，然后转向步骤 5；否则对于叶节点转向步骤 5，对于内部节点继续；

（3）测试 $Node_{active}$ 的每一个子节点 $Node_{child}$。如果 $Node_{child}$ 需要细化，则执行一个 EXPAND 操作：将 $Node_{active}$ 的所有子节点插入到活动链表，并移出 $Node_{active}$，然后转向步骤 5；否则，对于内部节点转向步骤 5，对于根节点继续；

（4）由视点相关标准测试 $Node_{active}$ 决定其是否绘制，转换该节点的显示状态并保持活动链表不变；

（5）遍历活动链表的下一个节点。

通常情况下，实时绘制相邻帧变化细微，更新活动链表只涉及很少的一部分聚类节点，因此，上述遍历方式可以充分利用模型显示本身的时间相关性来加速绘制。

4.3 数据结构

本文算法采用的主要数据结构是 4 种类型的层次聚类树节点，具体实现如下：

```
class BaseNode{
public:
    unsigned char flag; //节点标记
    BaseNode * prev, * next; //活动链表指针
    float c[3]; //空间坐标
    float radius; //包围球半径
    float theta; //法向锥半角
    NormalIndex n; //法向量
    struct {
        NormalIndex u; //椭圆盘主轴朝向
        NormalIndex v; // 椭圆盘辅轴朝向
        float r; //椭圆盘主轴长度
        float s; //主辅轴长度比率
    } Esplat;
}
class RootNode: public BaseNode{
public:
    unsigned char nchild; //子节点的个数
    BaseNode * * pchild; //子节点指针数组
}
class InternalNode: public BaseNode{
public:
    BaseNode * parent; //父节点指针
    unsigned char nchild; //子节点的个数
    BaseNode * * pchild; //子节点指针数组
}
class LeafNode: public BaseNode{
public:
    BaseNode * parent; //父节点指针
}
class StrayNode: public BaseNode{}
```

对于节点标记变量，其头两位编码为不同的节点类型，第 3 位指示该节点当前是否需要绘制。如果系统指定了轮廓增强功能，第 4 位标示一个轮廓节点。另外，为了有效节省存储空间，算法采用索引到预计算的量化法向量表来记录节点的法向量信息[13]。

5 试验结果

为验证算法的有效性，对几个经典模型进行了处理。图 6 和表 1 显示的是本文表面基层次聚类简化算法应用于 Bunny 模型的结果，原始模型包含 104 282个采样点，时间测量于CPU 为 P4-2.2GHz，1GB 内存的微机硬件平台。可以看到不同简化比率下曲面点的分布比较均匀，而且生成的聚类树也具有较低的深度。如前所述，这些特性对于得到高质量的简化结果和视点相关绘制中加速层次聚类树的遍历都是非常重要的。

（a）原始模型 （b）简化模型 $\varepsilon=0.09$ （c）简化模型 $\varepsilon=0.17$ （d）简化模型 $\varepsilon=0.26$

图 6 不同误差阈值下 Bunny 模型的简化结果

Fig. 6 Results of simplifying Bunny model with different error tolerance

表 1 Bunny 模型简化的定量结果

Tab. 1 Quantitative results for Bunny model simplification

阈值(ε)	根节点数目	游离节点数目	树深度	时间(s)
0.09	7 309	14 884	4	3.69
0.17	3 642	6 071	5	2.80
0.26	1 514	2 837	7	2.48

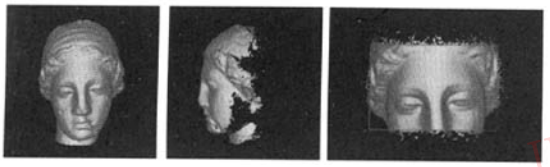
表 2 列出了另外一些模型以其对应简化层次聚类树的信息。

表 2　相关试验模型的层次聚类树信息

Tab. 2　Hierarchical cluster tree information for relative experimental models

模型	原始点集数目	阈值(ε)	节点数目	树深度	聚类大小(k_{min} ~ k_{max})
Igea	134 345	0.22	150 320	6	5 ~ 9
Buddha	543 652	0.26	604 058	8	5 ~ 11
Dragon	437 645	0.26	486 273	7	5 ~ 11

图 7 为对 Igea 模型的可见性裁剪结果,分别对应有效的背面剔除和视棱锥裁剪。

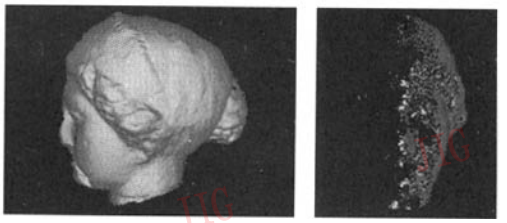


(a) 模型正面视图　(b) 保持背面剔除状态下的模型侧面视图　(c) 针对假定视棱锥的裁剪

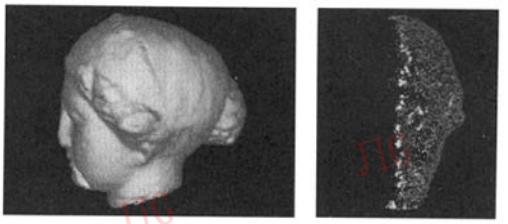
图 7　视点相关绘制下的可见性裁剪

Fig. 7　Visibility culling for view-dependent rendering

图 8 是利用轮廓增强机制提高简化模型视觉质量的对比效果。图 8(a)中模型绘制的轮廓区域产生了较为明显的视觉缺陷,设置相同的摄像机参数,尽管图 8(b)中的简化模型具有相对较大的屏幕投影误差阈值和简化比率,但其视觉质量仍优于图 8(a)。



(a) 无轮廓增强的模型绘制;16 063 个节点 $\tau = 9$pixels



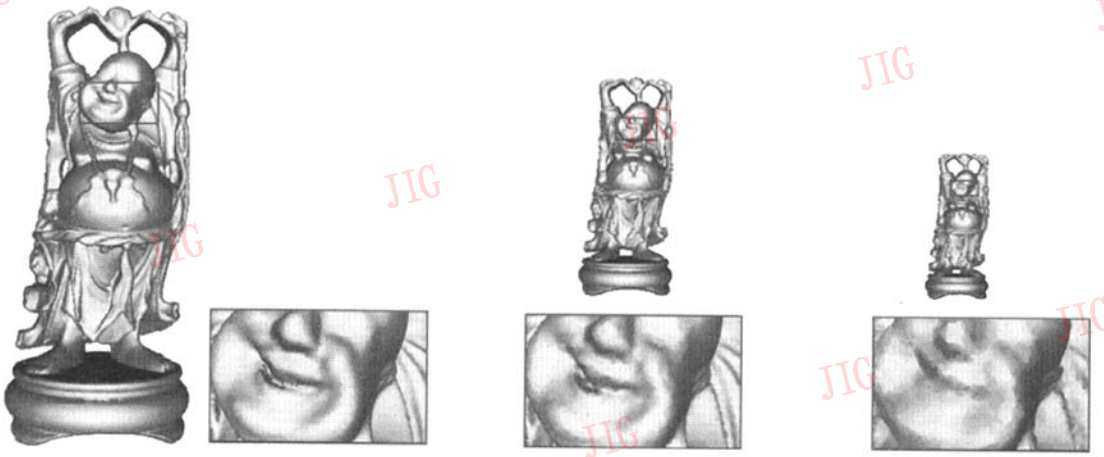
(b) 附带轮廓增强的模型绘制;12 581 个节点 $\tau = 16$pixels, $\theta_{threshold} = 0.1$, $\kappa = 0.2$

图 8　使用轮廓增强提高简化模型的绘制质量

Fig. 8　Improving render quality of simplified modely by using sihouette enhancement

因此,本文算法中引入的轮廓增强机制可以在保持模型较好视觉质量的同时有效提高简化率。

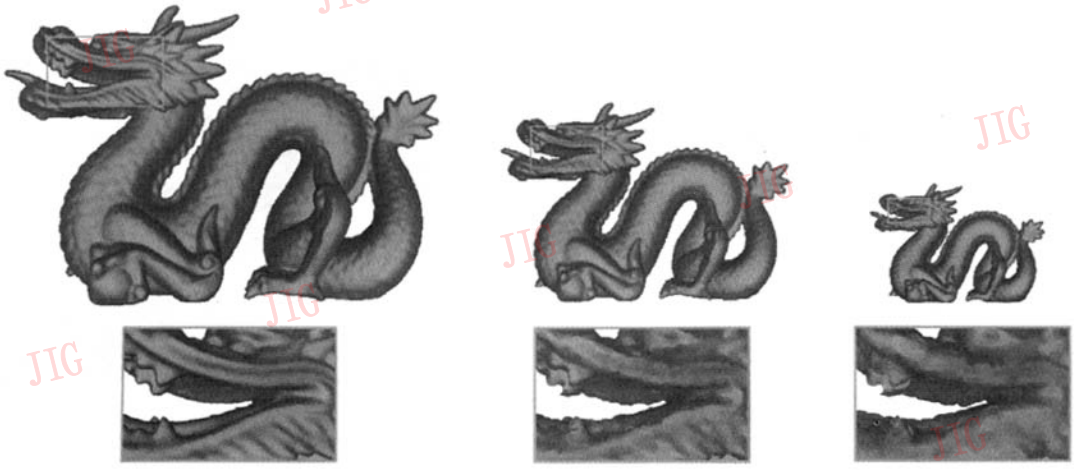
图 9 和图 10 为固定摄像机内外参数,通过改变模型与视点的距离来显示屏幕投影误差对视点相关 LOD 选择的影响。



(a) 127 578 个节点　(b) 83 074 个节点　(c) 35 625 个节点

图 9　变换相对于视点的距离来动态重建 Buddha 点曲面模型的不同 LOD 表示($\tau = 25$pixels)

Fig. 9　Switching distances form viewpoint for constructing dynamically different LODs of Buddha model with screen projection tolerance $\tau = 25$pixels



(a) 103 034 个节点 (b) 60 939 个节点 (c) 29 861 个节点

图 10 变换相对于视点的距离来动态重建 Dragon 点曲面模型的不同 LOD 表示($\tau = 25$pixels)

Fig. 10 Switching distances form viewpoint for constructing dynamically different LODs of Dragon model with screen projection tolerance $\tau = 25$pixels

6 结 论

本文算法以表面基聚类构造点曲面模型的多分辨率层次表达。其中法向锥半角作为简化测量标准,不但提供了误差上限控制,而且能够有效避免普通空间剖分基聚类过程中出现的一些问题。算法在实时绘制阶段执行的高效层次可见性裁剪和聚类树动态管理,对于加速点基绘制至关重要。此外,轮廓增强机制的引入改善了简化模型的绘制质量。

后续的研究工作可以在如下两个方面展开:(1)扩展现有聚类简化算法使其能够处理附带色彩、纹理等属性信息的点曲面,并且动态视点相关绘制中结合更多的视觉敏感信息例如光照等等。为此,需要同时在对象和屏幕空间定义更为合适的简化误差测量标准。(2)考虑包含几百万个采样点的大规模几何实体,无论是模型简化过程还是交互式视点相关绘制,系统都无法将所有的树节点载入内存,这时需要设计高效的外存简化算法。

参考文献(References)

1 Pfister H, Zwicker M, van Baar J, *et al*. Surfels: Surface elements as rendering primitives[A]. In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics[C], New Orleans, Louisiana, USA, 2000: 335 ~ 342.

2 Zwicker M, Pfister H, van Baar J, *et al*. Surface splatting[A]. In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics[C], Los Angeles, California, USA, 2001: 371 ~ 378.

3 Botsch M, Wiratanaya A, Kobbelt L. Efficient high quality rendering of point sampled geometry[A]. In: Proceedings of Eurographics Workshop on Rendering[C], Pisa, Italy, 2002: 53 ~ 64.

4 Botsch M, Kobbelt L. High-quality point-based rendering on modern GPUs[A]. In: Proceeding of the Pacific Graphics[C], Canmore, Alberta, Canada, 2003: 335 ~ 343.

5 Guennebaud G, Paulin M. Efficient screen space approach for hardware accelerated surfel rendering[A]. In: Proceedings of Vision, Modeling, and Visualization 2003[C], Munich, Germany, 2003: 485 ~ 493.

6 Zwicker M, Räsänen J, Botsch M, *et al*. Perspective accurate splatting[A]. In: Proceedings of the Graphics Interface 2004 Conference[C], London, Ontario, Cannda, 2004:247 ~ 254.

7 Pauly M, Gross M, Kobbelt L. Efficient simplification of point-sampled surfaces[A]. In: Proceedings of IEEE Visualization 2002 [C], Boston, Massachusetts, USA, 2002: 163 ~ 170.

8 Linsen L. Point Cloud Representation[R]. TR, 2001-3, Germany: Faculty for Computer Science, Universitaet Karlsruhe.

9 Alexa M, Behr J, Fleishman S, *et al*. Computing and rendering point set surfaces[J]. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(1), 3 ~ 15.

10 Moenning C, Dodgson N A. A new point cloud simplification algorithm[A]. In: Proceedings of the 3rd IASTED Conference on Visualization, Imaging and Image Processing[C], Benalmadena, Spain, 2003: 1027 ~ 1033.

11 Moenning C, Dodgson N A. Intrinsic point cloud simplification[A].

In: Proceedings of GraphiCon 2004[C], Moscow, Russia, 2004: 1147 ~ 1154.

12 Wu J, Kobbelt L. Optimized Sub-sampling of Point Sets for Surface Splatting[A]. In: Proceedings of Eurographics 2004[C], Grenoble, France, 2004: 643 ~ 652.

13 Pajarola R. Efficient level-of-details for point based rendering[A]. In: Proceedings of the IASTED Computer Graphics and Imaging[C], Honolulu, Hawaii, USA, 2003: 141 ~ 146.

14 Rusinkiewicz S, Levoy M. Qsplat: A multiresolution point rendering system for large meshes [A]. In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics[C], New Orleans, Louisiana, USA, 2000: 343 ~ 352.

15 Dey T. K, Hudson J. PMR: Point to mesh rendering, a feature-based approach[A]. In: Proceedings of IEEE Visualization 2002 [C], Boston, Massachusetts, USA, 2002: 155 ~ 162.

16 Zwicker M. Continuous reconstruction, rendering, and editing of point sampled surfaces[D]. Zurich, Switzerland: The Swiss Federal Institute of Technology Zurich, 2003.

17 Weyrich T, Pauly M, Heinzle S, *et al*. Post-processing of scanned 3D surface data[A]. In: Proceedings of Symposium on Point-Based Graphics'04[C], Zurich, Switzerland, 2004: 85 ~ 94.

18 Pauly M. Point primitives for interactive modeling and processing of 3D geometry[D]. Zurich, Switzerland: The Swiss Federal Institute of Technology Zurich, 2003.

19 Pajarola R. Fastmesh: Efficient view-dependent meshing[A]. In: Proceedings of the Pacific Graphics[C], Tokyo, Japan, 2001: 22 ~ 30.

20 Luebke D, Erikson C. View-dependent simplification of arbitrary polygonal environments [A]. In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics[C], Los Angeles, California, USA, 1997: 199 ~ 208.