计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(12):35−44 [doi: 10.15888/j.cnki.csa.007728]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于视点互信息的树叶实时简化方法①

王超凡[1,2], 王　标[1,2], 佘江峰[1,2,3]

[1](南京大学 地理与海洋科学学院 自然资源部国土卫星遥感应用重点实验室, 南京 210023)
[2](南京大学 地理与海洋科学学院 江苏省地理信息技术重点实验室, 南京 210023)
[3](江苏省软件新技术与产业化协同创新中心, 南京 210023)
通讯作者: 佘江峰, E-mail: gisjf@nju.edu.cn

摘　要: 三维树木模型在虚拟地理环境, 三维城市场景等领域中应用广泛, 但由于树木中包含丰富的几何信息, 难以对大规模的森林场景进行有效的渲染. 为此我们设计了一种基于视点互信息 (Viewpoint Mutual Information, VMI) 的树木实时简化方法. 在预处理中按照树枝间的拓扑关系将树木划分为具有父子关系的节点, 然后根据 VMI 计算每片树叶在多个视点下的平均重要度并以此对树叶进行排序, 重要程度较小的树叶在简化过程中将会被优先删除. 实时简化过程中, 我们提出了一种视点依赖的简化方法, 大大降低了需要渲染的数据量. 为了提高渲染森林场景时的性能, 我们使用了多种渲染优化措施以避免不必要的细节层次 (Level Of Detail, LOD) 切换.

关键词: 视点互信息; 虚拟地理环境; 树叶简化; 视点相关

引用格式: 王超凡,王标,佘江峰.基于视点互信息的树叶实时简化方法.计算机系统应用,2020,29(12):35–44. http://www.c-s-a.org.cn/1003-3254/7728.html

Real-Time Tree Simplification Based on Viewpoint Mutual Information

WANG Chao-Fan[1,2], WANG Biao[1,2], SHE Jiang-Feng[1,2,3]

[1](Key Laboratory for Land Satellite Remote Sensing Applications of Ministry of Natural Resources, School of Geography and Ocean Science, Nanjing University, Nanjing 210023, China)
[2](Jiangsu Provincial Key Laboratory of Geographic Information Science and Technology, School of Geography and Ocean Science, Nanjing University, Nanjing 210023, China)
[3](Collaborative Innovation Center for Novel Software Technology and Industrialization of Jiangsu Province, Nanjing 210023, China)

Abstract: The three-dimensional tree model is widely used in the fields of virtual geographic environment, three-dimensional city scenes, etc. However, due to the rich geometric details of trees, it is still a huge challenge to effectively render trees in large-scale forest scenes. In this study, a real-time simplification method for tree foliage based on Viewpoint Mutual Information (VMI) is proposed. In the preprocess, the whole tree is divided into parent-child branch and leaf nodes according to its own topological relationship. Then, the average importance of each leaf under multiple viewpoints is calculated based on the VMI value and the leaves are sorted based on it. The leaves of less importance are pruned first during the real-time simplification process. In the real-time simplification process, we propose a view-dependent simplification method which greatly reduces the number of primitives needed to be rendered. In order to improve the rendering performance in forest scenes, a variety of rendering optimization measures are taken to avoid unnecessary Level Of Detail (LOD) transition.

Key words: Viewpoint Mutual Information (VMI); virtual geographic environment; simplification for foliage; view-dependent

① 收稿时间: 2020-05-04; 修改时间: 2020-06-12; 采用时间: 2020-06-28; csa 在线出版时间: 2020-11-30

1 引言

1.1 研究背景

树木是户外场景的重要组成部分, 广泛地应用在虚拟地理环境, 电脑游戏, 城市仿真等领域中. 真实感较强的树木或植被模型能大大提高场景的真实感以及角色的沉浸式体验[1–3]. 但由于树木具有复杂的拓扑结构以及丰富的几何细节, 需要大量的几何图元进行表达, 因此对树木有效的渲染是计算机图形学领域内十分富有挑战性的课题[4,5]. 尽管计算机软硬件在近几十年间快速发展, 但仍然远远不能满足绘制复杂虚拟环境的需求[6]. 对于包含大量树木的森林场景, 其中的数据量将轻易超出可使用的内存, 同时对所有数据的渲染也将大大超出 GPU 的负荷. 为了提高场景中存在大量数据时的渲染效率, 许多学者分别提出了相应的细节层次 (Level Of Detail, LOD) 模型[7–9], 其主要思想是根据几何模型在三维场景中所处的不同位置、重要程度来适当降低非重要物体的几何细节, 从而获得更高效率的渲染效率. 对于树枝来说, 一般用多边形棱柱来表达其几何形状, 通过在横向和纵向上改变多边形棱柱的细分程度构建不同细节层次的树枝模型[9–11]; 而对于树叶模型, 一般由包含两个三角形的四边形构成[8,12,13], 其结构简单、相互独立且数量众多, 这三个特点使得对树叶进行实时简化十分富有挑战性[14].

1.2 国内外研究现状

为了对存在大量离散树叶的树木进行有效简化, 许多学者提出了多种简化算法. 2002~2009 年, 有学者通过迭代得合并两个树叶以对树木进行简化. Remolar 等提出了第一个用于树叶简化的算法 (Foliage Simplification Algorithm, FSA)[12,15]. FSA 算法选择两片树叶作为输入, 然后通过一个成本函数来控制树叶的合并, 进而生成一个可以大致保持原先两片树叶形状的新树叶. 重复执行以上操作即可对树冠进行简化. Zhang 等在 FSA 算法的基础上提出了连续树叶合并 (Progressive Leaves Union, PLU) 算法[16], 其改进了 FSA 算法中的成本函数, 选择更合理的树叶进行合并, 从而获得了更好的简化效果. 上述两种方法只对四边形形状的树叶有效, 且遍历和计算的过程相当耗时. 2006 年, Zhang 等将由三角形组成的树叶也考虑进来并引入了植物学中的花序, 叶序等概念. 他们使用“分级”的思想对树叶进行简化, 提出了基于植物器官的层次合并 (Hierarchical Union of Organs, HUO) 算法[13]. 2009 年, Deng 等利用树叶密度的概念调整树冠的局部简化率, 叶密度的使用降低了树冠中稀疏部分的简化程度, 可以达到更好的简化效果[8]. Bao 等提出了一种通过使用叶片纹理来简化叶片网格的新方法[17]. Zhang 等提出了一种基于信息论的误差测量方法来测量树叶合并前后的几何误差, 从而可以生成较好的树木 LOD 模型[18]. 尽管上述方法可以迭代得合并树叶从而大幅减少渲染数据量, 但往往难以取得较好的可视化效果, 尤其是当简化程度很高时, 树冠会出现严重的“失真”现象. 此外, 由于合并过程中的计算量较大, 这一类方法并不适用于对树叶的实时简化.

另一类值得提起的树叶简化方法是基于随机裁剪的树叶简化方法[19,20]. 随机裁剪算法假定树冠中的树叶有着相同的重要程度, 在简化过程中对树叶进行随机的裁剪. 基于此, 很多学者进行了相关的研究并提出了相应的方法. Gumbau 等提出了一种新颖的基于视点的树叶裁剪算法[14]. 得益于在预处理过程中对树木中每个单元可见性的量化, 该算法的主要优势是被遮挡的更严重的树叶单元有着更高的简化程度. Bao 等针对大规模的植物场景提出了一种优化的随机裁剪算法并引入了精确率与召回率 (Precision and Recall, PR) 作为衡量简化后视觉质量的指标[17].

在大部分情况下, 利用随机裁剪算法可以取得较好的简化效果. 但是考虑到树木中的树叶在面积、朝向以及在树冠中的位置的不同, 不同的树叶实际上有着不同的重要度. Gasch 等使用视点互信息 (Viewpoint Mutual Information, VMI) 的变化作为一个新的指标来衡量树叶简化过程中某一树叶被裁剪的视觉影响与依此判定树叶被裁剪的优先级[21]. 实验结果表明基于 VMI 的树叶简化可以更好地保持树冠的外部轮廓,同时加大对内部树叶的简化程度.

针对大规模的森林场景渲染, 基于图像的绘制方法 (Image Based on Render, IBR) 可以将远处的树木几何模型用图像代替, 从而降低渲染负担, 提升渲染效率[22,23]. 但 IBR 方法一般只适用于中距离或者远距离的树木渲染. 当视点距离观察对象较近时, 由于图像分辨率有限, IBR 方法往往难以取得较好的可视化效果. 此外, IBR 方法

往往难以对光照、阴影, 树木摇曳等动态效果进行仿真与模拟[6,24]. 另外一些学者将传统基于几何的方法与IBR 方法相结合, 可以在可视化效果与渲染效率之间达到一定的平衡[25,26]. 但树木由图像转变为几何物体时, 屏幕上往往会出现较为严重的跳变感.

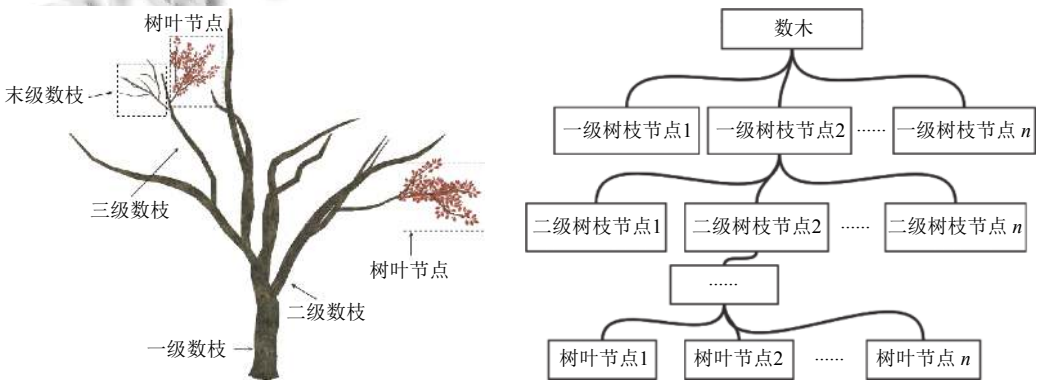
1.3 研究方法概述

本文提出了一种新颖的基于视点互信息的树木实时简化算法. 在预处理阶段首先对树木建模并根据树枝间的拓扑关系将树木划分为多个节点, 然后计算节点中每片树叶的重要度并对树叶进行排序, 简化过程中重要度较低的树叶将优先被简化. 在实时浏览过程中, 根据多个影响因子计算节点的简化率, 据此在每一帧中选取出每个节点中最重要的数据进行渲染. 通过这种方法, 在取得较高简化率的同时较好的保持了树木的几何形状与外观特征, 并有效降低了不同 LOD 切换时的跳变感. 在实时运行阶段还使用了一些优化的措施, 使得本研究在渲染效率和可视化效果之间达到较好的平衡.

2 基于 VMI 的树叶简化算法

2.1 节点划分

在预处理过程中首先根据树枝间的拓扑关系将树木划分为具有父子关系的节点. 如图 1 所示, 每个树枝都由一个节点表示, 树枝节点具有树枝层级的属性, 树干的树枝层级为 1, 后续层级的树枝依次递增 1. 末级树枝下的子节点包括了所有在该树枝上生长的树叶. 节点数据以多叉树结构组织. 这样的树状数据结构使我们可以通过父节点方便地控制子节点的简化率、渲染状态等信息, 同时可以在渲染时对 drawing call 进行合并, 从而可以大大减少 CPU 与 GPU 间的通信.



(a) 节点划分 (b) 节点的数据结构

图 1 按照拓扑关系将树木划分为具有父子关系的节点

2.2 基于 VMI 的树叶排序

VMI 的概念由 Castelló 等在 2008 年首次提出[27], 其基于信息论中互信息的概念, 衡量视点与物体面片之间的相关性. VMI 的主要优点在于以下两个方面: (1) VMI 对物体形变十分敏感, 与视觉感受有很强的相关性,可以更好的保持外部轮廓信息; (2) VMI 可以更好的简化树冠内部因遮挡而不可见的树叶. 因此, 利用 VMI 对树叶节点进行重要度排序, 可以在达到较高简化率的同时保持与原模型的视觉相似性.

假设视点集合为V, 以相同的概率均匀分布在物体四周, 则V的边缘概率为$p(v)=1/N_v$, N_v为视点个数. 物体面片的集合设为O, a_o为面片o在以视点v为球心的球面上的投影面积, $a_t=\sum_{o\in O}a_o$, 它表示在视点v下所有面片投影面积之和, 则面片o在视点v下的可见性的条件概率$p(o|v)=a_o/a_t$, 由此得出面片o在所有视点下的平均可见性为:

$$p(o)=\sum_{v\in V}p(v)*p(o|v)=\frac{1}{N_v}*\sum_{v\in V}p(o|v) \tag{1}$$

需要注意的是, 投影面积a_o指的是面片o投影面积中可见的那一部分. 如果一个面片o'完全被遮挡, 则o'的投影面积为 0. 最后, 即可得出 VMI 的定义如式 (2), 其反映了物体面片集合O在某一视点v下的整体可见性.

$$I(v,O)=\sum_{o\in O}p(o|v)*\log\frac{p(o|v)}{p(o)} \tag{2}$$

VMI 对物体外部的面片非常敏感, 相比于内部被

遮挡的树叶, 当外部的一片树叶被裁剪掉时, VMI 值会变化更大, 因此可用来衡量节点中树叶的视觉重要程度. 当一个节点因简化从O变为O'时, 由此带来的 VMI 误差为:

$$e_{\mathrm{VMI}}=\sum_{v\in V}\left|I(v,O)-I(v,O')\right| \tag{3}$$

为了实时运行阶段的树叶简化顺序更加合理, 我们在预处理中首先根据 VMI 对树叶按照视觉重要度进行排序. 在计算过程中, 每个树叶节点都会被看作一个包含多片树叶的物体O, 这些节点在预处理过程和实时简化过程中均相互独立. 在对树叶按照重要度排序的过程中, 每次选取e_{VMI}最大的树叶作为最重要的树叶. 需要注意的是, 由于一个树叶被裁剪掉之后可能会影响其它树叶的可见性, 因此每当一个树叶被裁剪掉, 需要重新进行 VMI 的计算.

2.3 实时简化率计算与渲染数据确定

为了使不必要的几何数据不被渲染, 在每一帧都会为每一个节点计算简化率R_s以确定其 LOD. 为了能够准确计算出节点的简化率, 本研究综合考虑了以下影响因子: 包围球层级 (Bounding Sphere Level, *BSL*)、树枝层级 (Branch Level, *BL*)、树叶密度 (Leaf Density, *LD*)、视距 (*dis*)、树枝方向与视线方向的夹角 (*dir*). 在实时浏览时, *BSL*、*BL* 的值不会发生变化, *LD* 等于初始叶密度乘以当前简化率, *dis*、*dir* 这两个影响因子则会随着视角、节点的不同而不同. 节点最终的简化率的计算公式如下:

$$\begin{aligned}R_s=&\alpha_1*f_1(BSL)+\alpha_2*f_2(BL,dist)+\alpha_3*f_3(LD)\\&+\alpha_4*f_4(dist)+\alpha_5*f_5(dir)\end{aligned} \tag{4}$$

其中, $f_1(BSL)$, $f_2(BL,dist)$等函数代表根据相应的影响因子计算出的简化程度; α_1, α_2等参数表示各个函数的权重且$\alpha_1+\alpha_2+\alpha_3+\alpha_4+\alpha_5=1$. 各个简化函数的定义如下:

$$f_1(BSL)=\begin{cases}0.0, & BSL=1\\0.4, & BSL=2\\0.8, & BSL=3\end{cases} \tag{5}$$

这里 *BSL* 值表示节点在树冠包围球中靠近树冠中心的程度. 由于内部的节点往往被外部的节点所遮挡, 因此相同条件下, 内部节点应比外部节点的简化率高. $BSL=1$时表示该节点暴露在树冠包围球的最外部, $f_1(BSL)$的值为 0.0; $BSL=3$时表示节点在树冠内部, 此时该节点被遮挡严重, $f_1(BSL)$应取较高值以降低不必要的渲染; 值为 2 时介于两者之间, 对应的函数值为 0.4.

浏览者在近处和远处的观察重点并不一致. 在近处浏览者更多的关注视觉上的细节, 但随着视距拉远, 树木的细节逐渐模糊, 浏览者变得对树木整体结构的变化更加敏感. 为了与浏览者的关注重点一致, 本研究中利用$f_2(BL,dist)$函数调节节点在近处和远处的简化策略,保证远距离观察时的视觉效果, 其定义如下:

$$f_2(BL,dis)=\begin{cases}0.0, & dist\le near\\k_1*BL*\log_h dis, & dis\in(near,far)\\0.5, & dist>far\end{cases} \tag{6}$$

上述公式确定了在不同的视距下, 树枝参数如何影响简化率. $near$, far分别代表视点到最精细与最粗糙 LOD 模型对应的距离. 当视距大于$near$且小于far时, 对于生长在 *BL* 值为 1, 2, 3 树枝节点上的树叶节点, 适当降低其简化率以保持树木的整体特征. 参数 h 控制 $f_2(BL,dist)$随距离变化的幅度; 参数k_1保证 $f_2(BL,dis)$的结果不大于 0.5.

$f_3(LD)$函数用来控制稀疏的树叶节点不会被过度简化. 叶密度 *LD* 等于初始叶密度乘以简化率, 其中初始叶密度为:

$$ld=\frac{num_{\mathrm{leaves}}}{len_{\mathrm{branch}}} \tag{7}$$

其中, num_{leaves}, len_{branch}分别代表初始模型中节点个数以及树枝长度, 这两个参数均可以在建模时得到. $f(LD)$的定义如下:

$$f_3(LD)=k_2*(LD-ld0) \tag{8}$$

其中, $ld0=\min(ld1,0.5*ld2)$, $ld1$, $ld2$分别表示初始树叶节点中的最低叶密度和平均叶密度; $k_2=1/ld0$. 当叶密度大于$ld0$时, $f_3(LD)$为正数, 树叶密度促进节点的简化; 当叶密度小于$ld0$时, $f_3(LD)$为正数, 树叶密度对简化率呈抑制作用.

距离参数对简化率的影响是最直接的, 也是最大的. $f_4(dist)$的定义如下:

$$f_4(dist)=\begin{cases}0.0, & dist\le near\\k_3*\log_{10}dist, & dist\in(near,far)\\1.0, & dist\ge far\end{cases} \tag{9}$$

其中, $k_3=1/\log_{10}far$. 距离与简化率的对数关系降低了距离在远处的影响.

为实现视点相关的简化, 本研究中利用dir参数调节节点在不同视角下的简化率, $f_5(dir)$定义如下:

$$f_5(dir)=\begin{cases}\cos(dir), & dir\in[0,\pi/2]\\ 0, & dir\in[\pi/2,\pi]\end{cases} \tag{10}$$

其中, dir代表视线方向dir_eye与树枝节点 (对于树叶节点, 则是其父节点) 生长方向dir_branch的夹角. 当dir处于$[0,\pi/2]$范围内时, 说明树枝生长在以视点为参考点的树冠的背面, 将被前面的树枝所遮挡, 且dir值越大, 树枝被遮挡的通常会越严重; 当dir处于$[\pi/2,\pi]$范围内时, 说明树枝生长在树冠的正面, 此时将$f(dir)$的值置为 0.

考虑到距离对简化率的影响最大, 在本研究中令$\alpha_4=0.3$, $\alpha_2=0.1$, $\alpha_1=\alpha_3=\alpha_5=0.2$. 在实时运行中的每一帧, 节点最终的简化率将由以上 5 个简化函数加权得到.

在获得节点的简化率R_s之后, 接下来要确定哪些数据应该被渲染. 由于树叶在预处理中已经按照重要性从高到低的顺序排列, 因此我们只需要简单地删除最后$(size*|R_c-R_l|)$片树叶即可实现对树叶节点的简化或者细化, 其中$size$表示节点中树叶的数量, R_c, R_l分别代表当前帧和上一帧的简化率. 节点的细化过程与简化过程类似, 只需要重新渲染对应的树叶即可实现.

3 渲染优化措施

在渲染大规模森林场景时, 一方面, 场景中存在着大量的几何模型, 这将会消耗大量的内存; 另一方面, 由于场景中节点数量众多, 过多的 LOD 模型切换使得我们很难在每一帧中为所有的节点计算出准确的 LOD 并进行有效的渲染. 为了解决上述的效率与内存问题, 本研究在几何着色器中实时放大树叶来保持树冠整体的视觉效果. 为了减少不必要的 LOD 切换, 本研究提出了一种基于距离的 LOD 切换控制措施以调节不同节点 LOD 切换的频率与幅度. 此外, 我们在预处理中生成闭合的树冠轮廓模型以代替远处树木的几何模型, 从而大大降低渲染负担.

3.1 树冠外观保持

当简化率过大时, 往往无法用剩余的树叶较好的表达出原始树冠的视觉效果, 这时可能会给浏览者带来一些视觉上不好的体验, 比如树冠逐渐由树叶的颜色变为树枝的颜色. 但由于此时节点往往距离较远或者被严重遮挡, 因此我们可以在实时运行过程中执行一些补偿措施以弥补简化率较大时的“失真”现象. 在本研究中采用 Gumbau 等在文献 [14] 中的提出的策略: 在实时浏览过程中改变树叶面积并保持树叶的总面积大致不变. 对于节点 i, 假设其初始状态下树叶个数为 n, 所有树叶的总面积为 S. 当在某一时刻简化率变为 r, 意味着将有$n\times r$片树叶被裁剪掉. 此时树叶的总面积可以近似的表示为$S*(1-r)$. 为弥补由于简化而造成的视觉损失, 我们将剩余的树叶沿树叶的中心向外扩展 s 倍以保持树叶总面积大致不变, 其中:

$$s=sqrt((1.0-R_l)/(1.0-R_c)) \tag{11}$$

此时该节点的树叶总面积为:

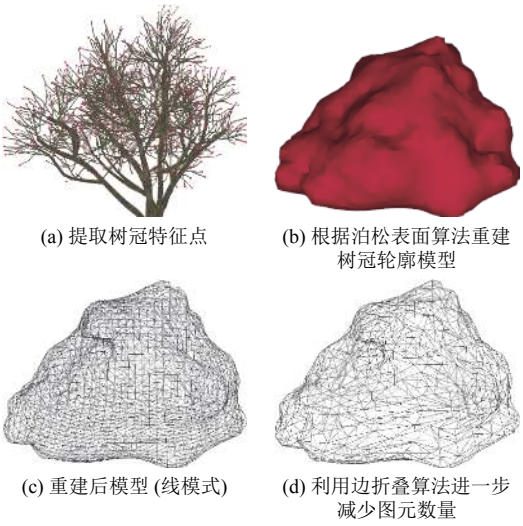
$$S_{\text{cur}}=S_{\text{last}}*s*s \tag{12}$$

其中, S_{cur}, S_{last}分别表示节点 i 在上一帧和当前帧的树叶总面积. 树叶中心的坐标等于树叶的 4 个顶点的平均值. 上述操作在 GPU 渲染管线中的几何着色器阶段执行, 几乎不会带来额外的渲染成本.

3.2 基于泊松表面重建算法的树木轮廓模型构建

当照相机距离树木较远时, 此时, 树木内部的细节已经基本不可见, 浏览者此时往往只能注意到树木的整体轮廓. 在浏览场景时, 这些树木模型占据了较多的内存, 但是对浏览者的观感作用有限. 同时, 这些树木中的节点也没有必要进行过多的 LOD 切换, 否则在电脑屏幕上将会出现一些不断闪烁的“噪点”, 这将给用户带来非常不好的体验.

为了解决上述问题, 本研究在预处理过程中为树木预先构建外部轮廓模型. 这种模型是一种连续的网格结构, 与实际上的树木模型相差较远, 但远距离观察时, 这种简化的轮廓模型与树木整体的形状十分相似, 因此可以在距离较远时代替原始树木模型进行渲染. 具体操作步骤如图 2 所示: 首先选取所有树枝的顶部端点作为特征点, 其发现方向为树冠中心指向该点的方向. 但是有些特征点会更靠近树冠内部而不是外部轮廓. 为此我们设置一个距离阈值 θ, 将所有到树冠中心的距离小于 θ 的特征点删除, 然后利用泊松表面重建算法[28,29] 生成树冠轮廓模型, 如图 2(b)、图 2(c). 生成的树冠模型可以比较真实的展现出树冠的整体轮廓, 最后利用边折叠的方式进一步的压缩顶点数量, 最终结果如图 2(d) 所示. 相比于原先树冠中 310 364 个顶点, 最终所构建的树冠轮廓模型中仅有 744 顶点, 降低了渲染负担.



(a) 提取树冠特征点 (b) 根据泊松表面算法重建树冠轮廓模型

(c) 重建后模型 (线模式) (d) 利用边折叠算法进一步减少图元数量

图 2 基于泊松表面重建算法构建树冠轮廓模型

3.3 LOD 管理系统

在对 LOD 切换的实验过程中, 我们发现对近处和远处的节点应采取不同的 LOD 管理方式. 具体来说: 近处的节点由于更靠近照相机, 在屏幕上往往会占据更多的像素, 因此其 LOD 变化不应太剧烈, 否则将产生强烈的跳变感; 而对于远处的节点, 由于距离较远且遮挡严重, LOD 变化较小时几乎不会被浏览者注意到. 因此对于这些节点, LOD 没有必要变化太频繁.

在上述的讨论中, LOD 变化的频率与幅度都与距离紧密相关. 在具体的实施过程中, 我们在每一帧中为每个节点记录其简化率的变化$\Delta rate$以及该简化率保持不变的时间$time_{\text{last}}$. 只有当$\Delta rate$与$time_{\text{last}}$满足式 (13) 时, 节点的 LOD 才会被更新.

$$\begin{cases} \Delta rate < \delta_1 * (dist - near)^{k_1} \\ \Delta time_{\text{last}} > \delta_2 * (dist - near)^{k_2} \end{cases} \tag{13}$$

式中, $\delta_1 = 1/(far - near)^{k_1}$, $\delta_2 = 1/(far - near)^{k_2}$. k_1, k_2均大于 0, 分别用来控制$\Delta rate$, $\Delta time_{\text{last}}$随距离变化的强度.

由于近处节点的距离较小, 此时的$\Delta rate$阈值也较小, 保证了近处节点不会产生视觉上的突变; 而对于远处的节点, $\Delta time_{\text{last}}$阈值较大, 这确保了远处的节点不会频繁的改变其细节层次.

4 结果与讨论

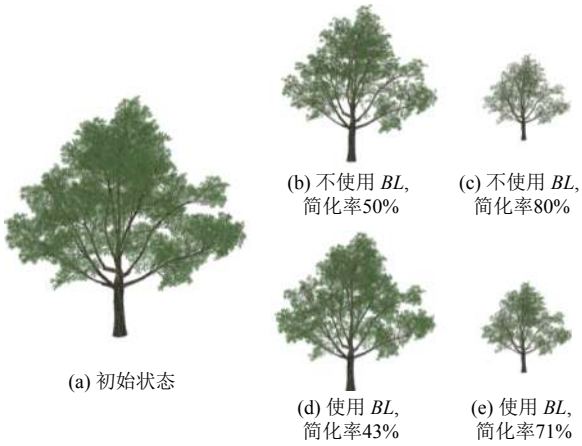
本文中三维算法是使用 OpenSceneGraph[30] API 开发的, 编程语言为 C++. 试验环境为: 2.8 GHz Intel Core i7-7700HQ CPU、16 GB RAM 以及 NVIDIA GeForce 940MX GPU. 本研究中所使用的树的实验参数列于表 1.

表 1 本研究中所使用的树木的参数

树种	顶点个数	树叶个数	三角形个数
夏栎	158 362	53 912	133 327
毛脉槭	268 441	81 677	176 236
血皮槭	321 162	109 523	215 142
椴树	634 812	292 581	334 700
桤木	1183 324	173 602	811 710

4.1 树木可视化效果

为了阐明本文中的方法的合理性, 我们进行了一系列的实验来验证本文中的因子以及优化措施是如何提高效率并改善视觉效果. 图 3(a) 表示树木的初始状态, 图 3(b), 图 3(c) 分别表示不使用 *BL* 参数时树木的简化效果; 对应的, 图 3(d), 图 3(e) 表示使用 *BL* 参数时的树木的简化效果. 图 3(b)–图 3(e) 的简化率分别为 50%, 80%, 43%, 71%, 从图 3 中可以看出, 使用 *BL* 参数虽然会轻微的增加渲染数据, 但却大大提高了树木的可视化效果.



(a) 初始状态 (b) 不使用 *BL*, 简化率50% (c) 不使用 *BL*, 简化率80%

(d) 使用 *BL*, 简化率43% (e) 使用 *BL*, 简化率71%

图 3 利用 *BL* 参数可以更好得保持树木整体形状

本研究利用 *Dir* 参数实现视点相关的实时树木简化. 如图 4 所示, 图 4(b)–图 4(d) 展示了夏栎树在简化率为 50%, 距离为 25 m 时各个角度的观察效果. 从图中可以明显看出, 背面的节点相比于正面的节点有着更高的简化率.

图 5 展示了在相同的简化程度下, 放大树叶与不放大树叶所造成的效果差异. 从左到右树木的距离与简化率分别为 5 m/0%, 30 m/90%, 100 m/98%, 10 m/98%. 显然, 放大之后的树叶可以有效地保持树冠

外观在不同 LOD 下的一致性.

在综合考虑了本文中的所有影响因子之后, 我们为表 1 中的 3 种典型树木生成相应的 LOD 模型. 如图 6 所示: 自上到下分别是夏栎树, 桤木以及椴树, 从左往右简化率分别为 0%, 50%, 98%. 结果表明我们的方法可以在大大降低渲染数据时保持较高的保真度.

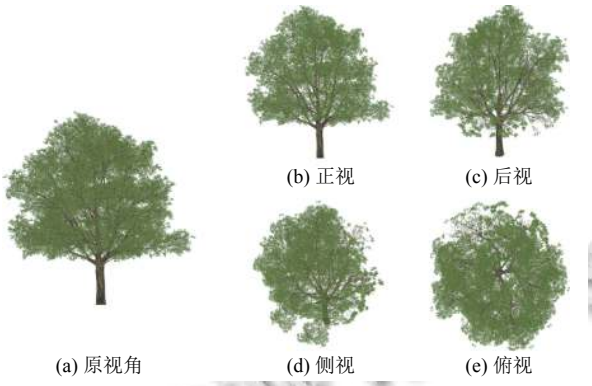


图 4 利用 *Dir* 参数提高背面节点的简化程度

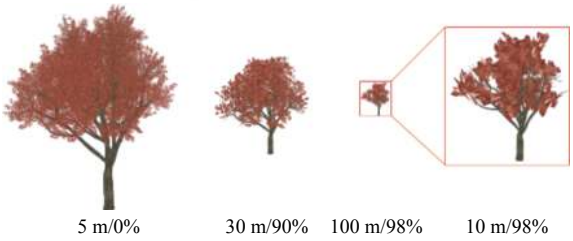


图 5 放大树叶以保持树冠外观的一致性

由于我们在预处理中为树木生成与原始模型十分相似的三维轮廓模型, 我们可以轻易地渲染出包含大量树木的森林场景. 图 7 展示了一个包含 1660 棵树的森林场景, 其中视锥体内共有 621 棵树, 对应的初始状态下共有 4251 万个三角形图元. 而在当前帧中仅有 199.75 万个三角形图元需要渲染, 整个场景的平均简化率为 4.7%, 平均帧率为 30.21 fps, 是不做任何简化处理的 15.18 倍.

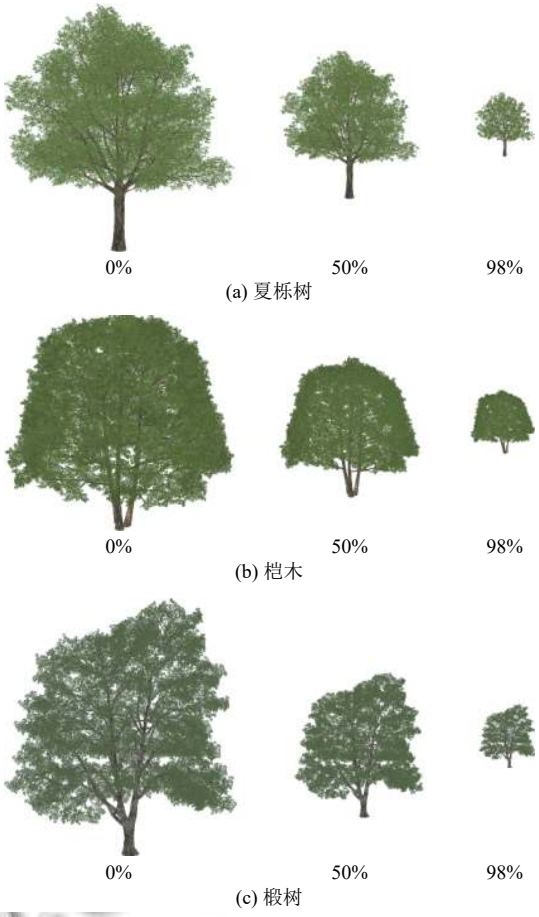


图 6 不同树木的 LOD 模型

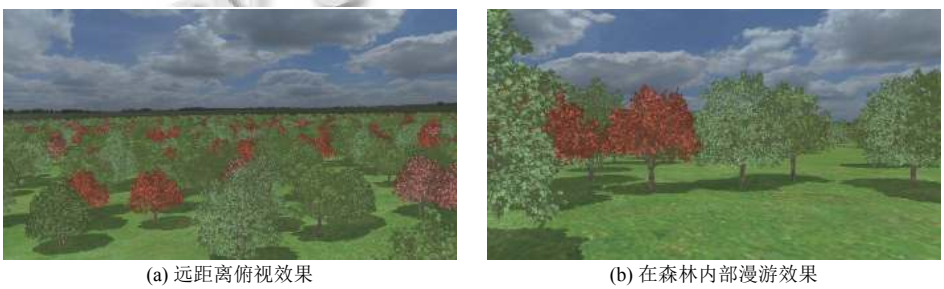


图 7 大规模森林场景的可视化效果

4.2 对比分析

在森林场景中实时浏览时, 存在两个不可避免的难题. 首先, 表达整个场景所需的几何数据将很容易超出可用内存; 其次, 浏览中大量的 LOD 切换不仅会导致屏幕闪烁, 而且还会带来很多 drawing calls, 这将大大降低渲染效率. 对于第一个问题, 本研究基于泊松表

面重建算法, 在预处理中生成了封闭的树木轮廓模型以用于远处树木的渲染. 如图 8 所示: 第一行以线模式展示了 5 种树木模型的轮廓模型, 第二行以点模式展示了对应树木的原始形态, 最后一行将两者叠加以展示其形状对比. 表 2 对比了轮廓模型和原始模型之间的顶点数量. 结合图 8 以及表 2 可以看出该方法在保持树的整体形状的同时可以大大降低渲染负担.

为解决第二个问题, 本研究提出了一种基于距离的 LOD 控制方案, 它使得近处和远处的节点具有不同的 LOD 转换条件. 这种措施减少了附近节点突然改变的可能性, 并降低了远处节点的 LOD 转换的频率. 我们的方法对于包含了成千上万个节点的大规模森林场景尤其有效. 在图 7 的场景中, 与没有 LOD 转换控制的实验相比, 帧率平均提高了 12.6 fps.

(a) 轮廓模型

(b) 原始形态

(c) 叠加及形状对比

图 8 树木轮廓模型和原始树模型的对比

表 2 树木轮廓与原始树冠模型的三角形图元数量对比

树种	初始模型的三角形图元数量	轮廓模型的三角形图元数量
夏栎	133 327	4198
毛脉槭	176 236	4946
血皮槭	215 142	5230
椴树	334 700	6713
桤木	811 710	5624

得益于在预处理中对树叶按照视觉上的重要程度进行排序, 我们的方法在裁剪树叶的顺序方面比其它树叶简化方法明显更加合理, 因此也有着更好的可视化效果. 图 9 展示了我们的方法与随机裁剪算法的对比. 从左到右, 分别为树木初始状态, 基于随即裁剪算法的效果以及我们算法的效果, 简化率均为 50%. 我们使用 PR 值定量比较这两种算法的优劣. 基于随机裁剪算法的 PR 值分别为 0.90/0.71、0.92/0.78, 而使用本文中的方法得到的 PR 值分别为 0.95/0.82、0.96/0.88. 从中可以看出, 我们的算法在取得高简化率时可以更好地保持树木的外观特征.

5 总结与展望

我们提出了一种基于 VMI 的树木实时渲染算法. 我们使用 VMI 值的变化作为为树叶简化顺序的依据. 在实时运行阶段, 综合考虑遮挡, 视距等因素, 在视觉效果和简化率上达到了较好的平衡. 为了有效处理大规模森林场景中存在的大量 LOD 变换, 我们提出了一套新的判断准则来决定节点的 LOD 应该改变. 此外, 我们在预处理中为树木生成闭合的三维轮廓模型以供视距较远时动态加载, 这大大降低了场景中的内存与

渲染负担. 实验结果表明我们的方法在渲染效率和可视化效果上都可以取得令人满意的效果.

我们计划在以下方面继续改进我们的研究. 第一, 探索更优或者自动化的参数确定方法以提供更好的简化策略; 其次, 我们将研究如何更高效的利用粗糙的 LOD 模型取代那些被遮挡的树木; 最后, 我们考虑将本研究中提出的 LOD 数据结构应用在对树木的网络传输中以降低网络传输和客户端渲染的压力.

(a) 原始树模型 (b) 随机裁剪方法 (c) 本文方法

图 9 本研究的方法与基于随机裁剪的方法的效果对比

参考文献

1 卢宇. 大规模森林场景建模与渲染关键技术的研究与实现 [硕士学位论文]. 成都: 电子科技大学, 2011.

2 刘真余, 芮小平, 董承玮. 森林三维真实感建模与可视化 LOD 技术研究. 中国科学院研究生院学报, 2011, 28(3): 322–327.

3 Favorskaya MN, Jain LC. Large scene rendering. In: Favorskaya MN, Jain LC, eds. Handbook on Advances in Remote Sensing and Geographic Information Systems: Paradigms and Applications in Forest Landscape Modeling. Cham: Springer, 2017. 281–320.

4 刘峰. 大规模森林场景的实时绘制及动态模拟 [博士学位论文]. 杭州: 浙江大学, 2011.

5 张高原. 面向树木可视化的沉浸式虚拟场景的研究与实现 [硕士学位论文]. 西安: 西北农林科技大学, 2017.

6 董天阳, 范允易, 范菁. 保持视觉感知的三维树木叶片模型分治简化方法. 计算机辅助设计与图形学学报, 2013, 25(5): 686–696. [doi: 10.3969/j.issn.1003-9775.2013.05.013]

7 邓擎琼. 基于植物模型处理的森林景观实时绘制 [博士学位论文]. 北京: 中国科学院自动化研究所, 2008.

8 Deng QQ, Zhang XP, Yang G, *et al*. Multiresolution foliage for forest rendering. Computer Animation & Virtual Worlds, 2010, 21(1): 1–23.

9 Zhang XP, Bao GB, Meng WL, *et al*. Tree branch level of detail models for forest navigation. Computer Graphics Forum, 2017, 36(8): 402–417. [doi: 10.1111/cgf.13088]

10 Max N. Cone-spheres. Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques. Dallas, TX, USA. 1990. 59–62.

11 Garland M, Shaffer E. A multiphase approach to efficient Surface simplification. Proceedings of the Conference on Visualization. Boston, MA, USA. 2002. 117–124.

12 Remolar I, Chover M, Oscar B, *et al*. Geometric simplification of foliage. Proceedings of Eurographics 2002 Short Presentations. Geneva, Switzerland. 2002.

13 Zhang XP, Blaise F, Jaeger M. Multiresolution plant models with complex organs. Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and

Its Applications. Hong Kong, China. 2006. 331–334.
14 Gumbau J, Chover M, Remolar I, *et al*. View-dependent pruning for real-time rendering of trees. Computers & Graphics, 2011, 35(2): 364–374.
15 Remolar I, Chover M, Ribelles J, *et al*. View-dependent multiresolution model for foliage. Journal of WSCG, 2003, 11(1–3): 370–378.
16 Zhang XP, Blaise F. Progressive polygon foliage simplification. Proceedings of International Symposium on Plant Growth Modeling, Simulation, Visualization and Their Applications. Beijing, China. 2003. 182–193.
17 Bao GB, Li HJ, Zhang XP. Realistic real-time rendering for large-scale forest scenes. 2011 IEEE International Symposium on Virtual Reality Innovation. Singapore, Singapore. 2011. 217–223.
18 Zhang SL. Foliage simplification based on multi-viewpoints for efficient rendering. Journal of Software, 2014, 9(7): 1655–1665.
19 Cook RL, Halstead J. Stochastic pruning. Proceedings of Eurographics Workshop on Natural Phenomena. Dublin, Ireland, 2005.
20 Cook RL, Halstead J, Planck M, *et al*. Stochastic simplification of aggregate detail. ACM Transactions on Graphics, 2007, 26(3): 79. [doi: 10.1145/1276377.1276476]
21 Gasch C, Remolar I, Chover M, *et al*. Viewpoint-driven simplification of plant and tree foliage. Entropy, 2018, 20(4): 213. [doi: 10.3390/e20040213]
22 Fuhrmann AL, Umlauf E, Mantler S. Extreme model simplification for forest rendering. Proceedings of the First Eurographics Conference on Natural Phenomena. Goslar, Germany. 2005. 57–67.
23 Decaudin P, Neyret F. Volumetric billboards. Computer Graphics Forum, 2009, 28(8): 2079–2089.
24 黄瑞荣, 张怀清, 李永亮, 等. 森林场景中太阳与阴影实时动态可视化模拟. 南京林业大学学报 (自然科学版), 2017, 41(6): 109–114.
25 Lee J, Kuo CJ. Tree model simplification with hybrid polygon/billboard approach and human-centered quality evaluation. Proceedings of IEEE International Conference on Multimedia and Expo. Suntec City, Singapore. 2010. 932–937.
26 Kohek Š, Strnad D. Interactive Large-scale procedural forest construction and visualization based on particle flow simulation. Computer Graphics Forum, 2018, 37(1): 389–402. [doi: 10.1111/cgf.13304]
27 Castelló P, Sbert M, Chover M, *et al*. Viewpoint-driven simplification using mutual information. Computers & Graphics, 2008, 32(4): 451–463.
28 Kazhdan M, Bolitho M, Hoppe H. Poisson surface reconstruction. Proceedings of the 4th Eurographics Symposium on Geometry Processing. Goslar, Germany. 2006. 61–70.
29 Kazhdan M, Hoppe H. Screened poisson surface reconstruction. ACM Transactions on Graphics, 2006, 32(3): 29.
30 OpenSceneGraph 3.0: Beginner's guide. https://www.packtpub.com/game-development/openscenegraph-30-beginners-guide. [2020-02-15].