基于局部熵和四叉树结构的地形简化算法

尤克非 明德烈 王广君 田金文 柳 健
(华中科技大学电子与信息工程系,图像信息处理与智能控制教育部重点实验室,武汉 430074)

摘 要 地形实时简化在三维地形可视化和虚拟现实的应用中是非常重要的,为此提出了一种基于地形局部熵的实时地形简化算法.该算法采用四叉树结构进行地形简化,并使用局部熵作为误差测度来提高简化结果的质量.在四叉树结构"裂缝"的消除方面,采用更加合理的数据结构来提高算法的效率.实验结果表明,该算法具有实时、高效的特点,可以满足三维地形可视化和虚拟显示应用中地形实时简化与显示的要求.

关键词 地形简化 局部熵 四叉树 LOD 视点相关
中图法分类号: TP391.9 P208 **文献标识码**: A **文章编号**: 1006-8961(2002)10-1083-06

Real-time Terrain Simplification Algorithm Based on Local Entropy and Quadtree Structure

YOU Ke-fei, MING De-lie, WANG Guang-jun, TIAN Jin-wen, LIU Jian
(Department of Electronics and Information Engineering, State Key Lab. for Image Processing & Intelligent Control, Huazhong Univ. of Science & Technology, Wuhan 430074)

Abstract Real-time simplification of terrain is very important in visualization of 3D terrain and Virtual Reality (VR). To best exploit the rendering performance, the scene complexity must be reduced as much as possible without leading to an inferior visual representation. The most common way to increase efficiency is the use of different levels of detail(LODs) for different areas of the scene. The algorithm in this paper is a real-time terrain simplification algorithm, based on local entropy. The algorithm adopts quadtree structure, which can provide real-time computation and high efficiency. The algorithm uses "Local Entropy" as error metric, improving the quality of simplified results. In an aspect of fixing "gaps" that is unavoidable in quadtree structure, the algorithm adopts more rational data structure to acquire higher efficiency. The experiment results indicate that the algorithm is practice and effective, which can meet the demand of real-time rendering in 3D terrain simplification and VR.

Keywords Terrain simplification, Local entropy, Quadtree, Level of detail, View dependent

0 引 言

随着地形可视化和虚拟现实应用需求的不断提高,具有真实自然视觉效果的地形仿真建模技术将变得越来越重要.三维地形可视化通常与虚拟环境中物体的运动直接有关[1],在建模和实时显示两个方面都要求较高.在模型方面,地形一般采用数字高程模型(DEM)来表示,其数据结构多为格网数据结构;在实时显示方面,虽然计算机运算速度、存储器容量、图形加速硬件有了很大的发展,但同时也应看到,实际应用中所需的图形数据量往往比硬件可以实时显示的数据量大一个甚至几个数量级.由于地形数据量大,模型的复杂程度往往超过当前图形工作站的实际处理能力,因此如果不对地形数据进行适当的简化,要进行地形可视化仿真是非常困难的.

目前,地形简化算法基本分为两种:一是基于四叉树结构的简化算法;二是基于三角网的简化算法.

基金项目:国防基础研究基金项目(J1600B001)
收稿日期:2001-10-08;改回日期:2001-12-17

前者速度快,但是简化效果不如后者;后者的效果应该可以达到最优,但速度却要比前者慢一个甚至几个数量级[2]. Lindstorm 较早给出了四叉树结构的实时地形简化算法[3],王宏武等提出了基于多分辨率地形模型的视点相关模型[4],郝鹏威等改进了四叉树线索化的结构[5]. 通过考察常规的四叉树简化算法,发现在简化质量和简化速度上还可以进一步提高.

1 基于四叉树地形简化算法

1.1 DEM 四叉树的结构

DEM 四叉树的结构一般由一个树根节点和若干树枝和叶子节点组成(如图 1 所示). 每个节点包括网格的 4 个顶点的位置,4 个子节点指针和一个父节点指针. 如果一个节点的 4 个子节点指针均为空,则该节点为叶子节点;如果一个节点的父节点为空,则该节点为根节点;否则,该节点为树枝节点.

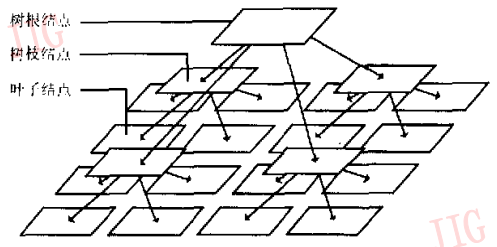


图 1 DEM 四叉树结构图

DEM 四叉树的构造过程为:首先取整个地形为根节点,然后检查这个节点,如果该节点满足某种条件,则可以认为它是叶子节点,否则将该节点四分为 4 个子节点,再使用递归方式分别检查 4 个新的节点是否满足成为叶子节点的条件. 这样不断进行下去,直到节点不可再分为止. 最后,叶子节点即为产生的简化结果.

1.2 基于局部熵测度的叶子节点判定和产生

叶子节点的判定和产生直接影响到输出结果的质量. 通常,对每个节点计算一个重要性测度,如果这个重要性测度大于某个门限,则将该节点作为叶子节点. 必须提到的是,这里的节点不是一个顶点,而是一个矩形. 计算重要性测度时,可以分别计算矩形的每一个点,也可以计算中心点,其要视重要性测度的计算方法而定. 因此,一个地形简化算法好坏的关键取决于在某节点上的重要性测度计算准则.

对于实时简化算法,一个好的重要性测度应该具有如下特征:必须计算起来简单快捷;对任意的高度场信息都应该产生好的结果,并不具有特殊性;应该尽量只使用邻近局部范围内的信息.

测度计算方法通常有局部误差估计、曲率估计和全局误差估计[6],这里提出一种新的基于局部熵的测度计算方法.

设 $z=H(x,y)$ 为地形表面二维方向上的点 (x,y) 处的高程,点 (x,y,z) 位于实际的地形表面上. 对于一个范围为 $M\times N$ 大小的地形区域,定义 H_f 为该地形区域的熵,即

$$H_f=-\sum_{i=1}^{M}\sum_{j=1}^{N}p_{ij}\lg p_{ij} \tag{1}$$

式中,$p_{i,j}$ 为相对高程分布

$$p_{ij}=\frac{H(i,j)-k}{\sum_{x=1}^{M}\sum_{y=1}^{N}[H(x,y)-k]+\varepsilon}$$

$$k=\min\{H(x,y);x\in[1,M],y\in[1,N]\}$$

ε 是一个很小的数,加 ε 是为了防止 p_{ij} 的分母为零时发生除零错误(局部高程数据完全一致平坦的情况). 如果 $M\times N$ 是地形的一个局部区域,则称 H_f 为地形的局部熵.

局部熵反映了地形高程的起伏程度. 所以根据地形的局部熵可以把地形中起伏较为剧烈的目标区域分割出来,加以特殊处理. 因为局部熵是由窗口内多点高程数据共同的贡献,对于单点噪声不敏感,所以,局部熵本身具有一定的滤波效果. 由式(1)定义的熵涉及到对数运算,计算量大. 由定义可知,$p_{ij}\ll 1$,因此可通过泰勒级数展开舍去高次项得到近似计算公式

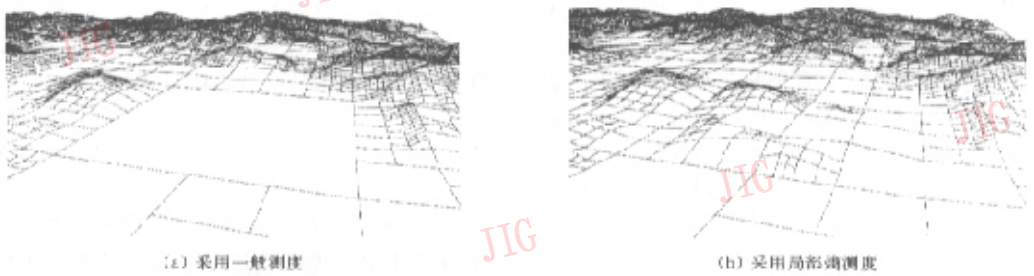
$$H_f\approx-\sum_{i=1}^{M}\sum_{j=1}^{N}p_{ij}(p_{ij}-1)=1-\sum_{(i,j)\in(M,N)}p_{ij}^2 \tag{2}$$

基于局部熵的重要性测度与局部误差测度一样,计算简单快捷,但其质量却优于局部误差. 图 2 给出了采用一般测度和采用局部熵测度的结果比较. 可以看到,用一般测度简化时,一些山丘被"削平"了. 基于局部熵的测度适用于各种类型的高度场,而且不难从计算方法看出,它的计算过程高度局部化.

1.3 DEM 四叉树的基本操作

1.3.1 节点的数据结构

节点是构成四叉树的最基本单位. 由于数据量比较大,应该使每个节点所占空间尽可能地小. 下面



(a) 采用一般测度　　(b) 采用局部熵测度

图 2

用 C++ 伪码给出节点的数据结构：

```
class CDEMCell
{
public :
    CDEMCell(…);      // 为节点构造函数
    BOOL CanBeLeaf(…);  //重要函数，上一节所述的
                        重要性测度的计算和叶
                        子节点的判定，均由此函
                        数完成
    void operator = (CDEMCell & Cell);
      //重载 = 操作符，完成节点的赋值操作
    CDEMCell * m_paSon[4];      //4 个子节点指针
    CDEMCell * m_pParent;       //父节点指针
    int16  m_nLeft, m_nTop, m_nRight, m_nBottom;
      //4 个角点下标
};
```

1.3.2　四叉树的建立

在以前的一些算法中，大多是先建立一个完整的四叉树，然后通过选择叶子节点来得到结果. 本文的算法则是根据节点的重要性测度控制生成树的深度，从而提高速度. 四叉树的建立有从粗到细的分裂法和由细到粗的合并法两种. 合并法将从每一个最基本的网格开始，最终达到树根，是一棵完整的树，算法的实现比较繁琐；而分裂法从树根开始，可以容易的控制深度，并且可以用递归方法来实现，比较简洁易懂. 本文采用的是分裂法，用分裂法建立 DEM 四叉树的基本算法很多文章有所涉及[3,4]，这里不再详述.

1.4　消除四叉树结构“裂缝”方法的改进

在基于四叉树的简化算法中，不可避免地会出现“裂缝”. 裂缝是由于相邻网格的分辨率不一致所造成的. 如图 3 所示，节点 C_0 具有较低的分辨率，而与之相邻的节点 C_1，C_2，C_3 有较高的分辨率，这就使得 C_0 和 C_1，C_2，C_3 之间形成了没有被覆盖的区域 $P_1P_2P_3P_4$，从而在地形绘制的时候将出现裂缝.

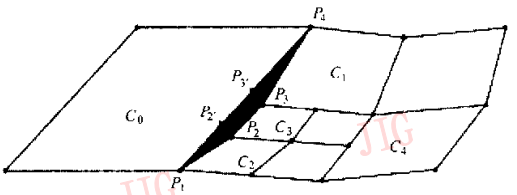


图 3　裂缝的形成

消除裂缝的方法是将 P_2 点拉到 P_2' 点处，将 P_3 点拉到 P_3' 点处. 但是必须从边长大的节点开始纠正裂缝，否则仍然会残留一些裂缝不能消除. 通常的方法是先对生成的结果，按边长进行一次排序，然后从大边开始进行纠正. 虽然排序时可以利用节点边长有 2^n 的形式，并且不需要全排序（只需要将边长一样的节点放在一起），但是排序的运算量仍然很大.

从表 1 可以看出，排序时间比生成叶子节点的时间还长，这显然是不能容忍的. 因此，要进一步提高速度，必须减少排序时间. 但是排序本身能够优化的空间有限，于是希望在生成叶子节点时能自动排序.

表 1　基本算法和改进算法部分步骤时间（和视点无关简化算法）

原始网格数	简化程度(%)	基本算法修复裂缝时间（生成叶子节点时间＋排序时间＋消除裂缝时间）(ms)	改进算法修复裂缝时间（生成叶子节点时间＋消除裂缝时间）(ms)
256×256	7.02	64(30+30+4)	34(30+4)
512×512	6.00	225(100+105+20)	125(105+20)
800×800	5.83	601(270+281+50)	330(280+50)
5 006×829	5.83	7 155(3 295+3 415+481)	3 916(3 295+3 415+481)

根据这个思路，可以在生成叶子节点时，就根据边长，将节点分别放入若干个节点链表中，并使每个链表中的节点边长相等，然后再将链表根据不同边长排序. 注意，此时链表的个数为 $O(\log_2 N)$. 于是就将所需排序的元素个数从 $O(N^2)$ 降为 $O(\log_2 N)$，这就大幅度减少了排序时间，而且生成叶子节点的时间几乎不变.

2 视点相关的简化算法

2.1 视点相关模型

虽然使用四叉树可以对地形数据进行高度简化，但是由于简化程度基本不变(见表 1“简化程度”列)，当原始地形网格数增加，数据量变大时，简化后的网格数也会随之增多，这将会影响仿真系统简化和显示的速度，有可能满足不了实时简化的要求. 在实际虚拟现实的仿真中，由于视点常常离地表很近，人们所见的只是局部视觉敏感区域的地形，而对较远的区域不感兴趣，甚至不可见，因此，可以进一步改进算法，使简化和视点相关，即在计算每个节点的重要性测度时，考虑和视点距离相关的测度，并且以适当的权值加权到总的测度中去. 但有两个问题需要解决：如何确定视觉敏感区域位置以及视觉中心和简化程度的关系，视点变化时，如何使简化模型平滑过渡.

为了确定视觉敏感中心位置，引入如下模型[3](如图 4 所示)：

点 V_0 为观察者位置，矢量 $\overrightarrow{V_0V_1}$ 为视觉方向. P 为水平面，V_2 为 V_0 在 P 面上的投影，α 为 $\overrightarrow{V_0V_1}$ 和 $\overrightarrow{V_0V_2}$ 的夹角. 定义 S 为视觉敏感中心点，即

$$S = V_2(1 - \cos\alpha) + V_1\cos\alpha \tag{3}$$

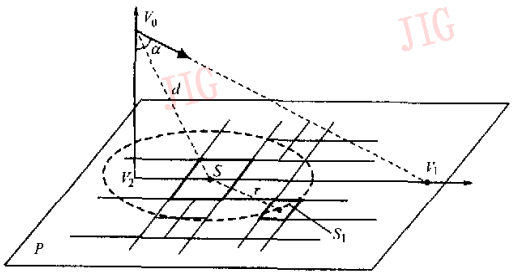


图 4 视觉敏感区域模型

视觉中心位置 S 应该具有最高的分辨率. 设 d 为 V_0 到 S 的距离，r 为任一节点 S_1 到 S 的距离. 显然，地形采取的分辨率应该与 d 和 r 成反比. 考虑到人眼的特性以及分辨率尽可能平滑过渡的要求，需给出计算视点距离时产生的误差测度

$$\epsilon = \lambda \times \log_2(d \times r^2) \tag{4}$$

其中，λ 为可调参数，可视不同的地形做一些调整.

2.2 视点相关动态简化

当需要随着视点的变化，动态实时地简化地形时，可采取以上视点模型，不断地重新生成四叉树. 由表 2 可以看出，采用视点相关的简化算法，简化程度是非常高的，而且从图 5 曲线看出，原始地形网格数越多，简化程度越高. 那么简化后的效果如何呢？图 6、图 7 给出了同一块地形采用视点无关和视点相关算法的效果图. 从图中可以看出，简化后的效果很好，视点敏感中心区域的细节基本都保留下来了. 如果贴上地表纹理，完全可以达到视觉仿真的要求.

表 2 改进的视点相关简化算法步骤时间

原始网格数	简化后网格数	简化程度(%)	生成叶子节点时间(ms)	消除裂缝时间(ms)
256×256	3 571	5.45	20	3
512×512	5 299	2.02	40	10
800×800	8 107	1.27	64	20
5 006×829	10 903	0.263	72	31

表 3 视点相关动态简化速度

单位：fps

原始网格数	不简化速度	简化程度(%)	简化后网格数	速度
512×512	1.5	2.02	5 299	20
800×800	<1	1.27	8 107	11
5 006×829	<1	0.263	10 903	8

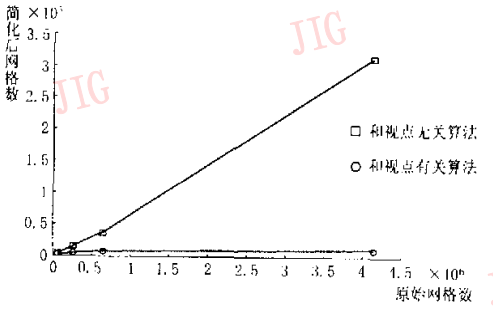
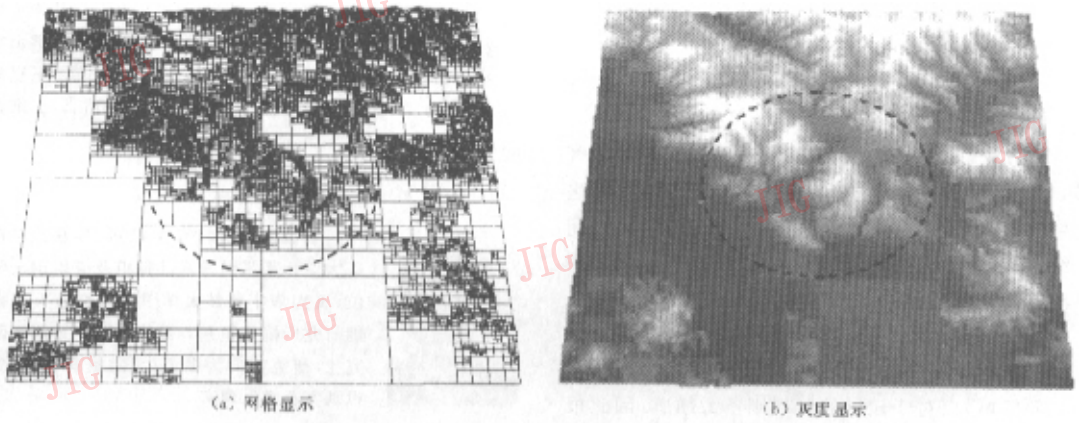


图 5 视点相关算法的复杂度分析

程序的测试环境：

硬件平台：Pentium Ⅲ 933MHz，256M 内存，nVidia GeForce2 MX400，64M 显存

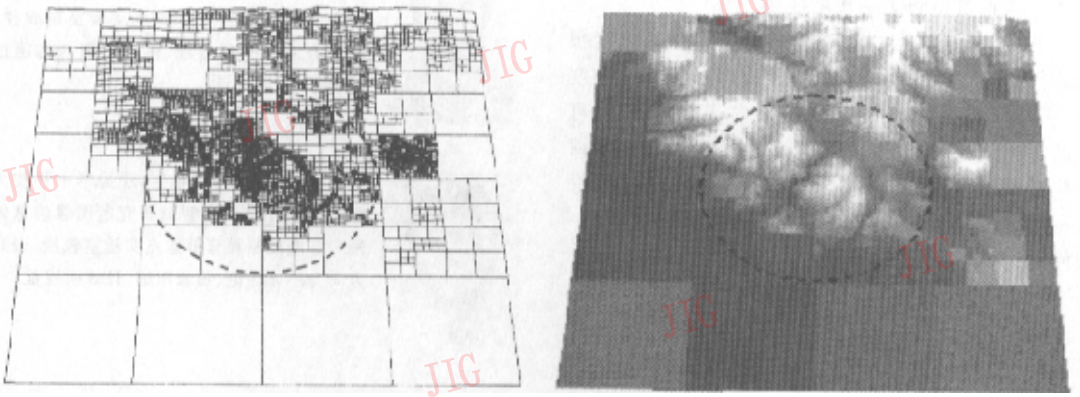
软件平台：中文 Windows 2000 Server，OpenGL 1.2.2



(a) 网格显示　　(b) 灰度显示

图 6　使用视点无关简化的结果(DEM)

(原始地形为 800×800 共 640 000 个网格,使用 37 339(5.834%)个网格,虚线圆选定的部分为视觉敏感区域)



(a) 网格显示　　(b) 灰度显示

图 7　使用视点相关简化的结果(DEM)

(原始地形为 800×800 共 640 000 个网格,使用 6 565(1.025 9%)个网格,虚线圆选定的部分为视觉敏感区域)

2.3　进一步提高简化速度

前面给出的是一些理论模型,但是由于三角和对数函数计算较慢,在实际设计仿真系统时,为了加快计算速度,可以对计算进行一些近似简化.例如用泰勒公式对式(3)和式(4)展开,取若干项,或者对于 a、d、r 等参数进行较粗的量化,预先计算一个结果表,然后在实时简化计算时进行查表,而不是计算,以提高速度.

另外,如果事先用局部熵计算整个地形每个可能节点的误差测度,并保存到一个哈希表(Hash Table)中,那么在实时简化时就不需要计算每个节点的误差测度了.这种方法可以进一步提高每一帧的计算效率,但是较大地增加了预处理时间.是否采用这种加速方法,可依设计者的实际情况而定.

3　结　论

基于局部熵的地形实时简化算法具有如下几个性质:

(1) 算法产生的结果是基于四叉树的矩形网格结构

四叉树编码的空间结构和 DEM 具有天然的一致性,查找,定位十分快捷,从而为快速实时简化提供了条件.

(2) 简化过程是快速实时的

三角网简化模型,从理论上可以达到最优(Delaunay 三角网[5]),但是速度很慢,完全满足不了实时性的要求.本算法的简化过程是快速的,可以

满足实时性的要求，而且，算法在产生结果的细节质量和所需时间上都是连续可调的.

(3) 该四叉树网格结构是连续 LOD 多分辨率模型

LOD 模型是一种多细节层次、多分辨率的模型，但用于仿真时，应选择适当的细节程度. 本算法产生的结果是一种连续 LOD，即各个细节层次之间连续，可以平滑过渡，从而为与视点相关简化模型的平滑过渡提供了条件.

(4) 算法可以产生与视点相关的结果

在地形可视化仿真时，人们往往比较关心视觉中心的区域，而对于较远的区域则不太敏感. 因此根据这一点，可以对视觉敏感区域的地形数据保留较多细节，而对于不敏感区域则进行较深层次的简化，从而使数据量得到进一步的压缩.

该方法采用四叉树的结构，地形简化的速度很快，时间和空间复杂度不高，适用于各种类型的高度场. 实验结果表明，对于实时地形简化，该算法是可行的，具有一定的应用价值. 本文讨论了基于四叉树结构的地形简化算法，但是对于如何对简化效果进行一个定量的分析，如何控制四叉树的节点数等一些问题，仍然值得进一步研究.

参 考 文 献

1 Hoppe H. Progressive meshes [A]. In: SIGGRAPH'96 Proc. [C]. New Orleans, Louisiana USA, 1996: 99～108.

2 Cignoni P, Montani C, Scopigno R. *et al*. A comparison of mesh simplification algorithms [J]. Computers & Graphics, 1998, 22 (1): 37～54.

3 Lindstrom P, Koller D, Ribarsky W *et al*. Real-Time, continuous level of detail rendering of height fields [A]. In: SIGGRAPH '96 Proc [C]., New Orleans, Louisiana USA, 1996: 109～118.

4 王宏武，董上海. 一个与视点相关的动态多分辨率地形模型[J]. 计算机辅助设计与图形学报，2000，12(8)：575～579.

5 郝鹏威，黄波，朱重光. 用于地面可视化的 DEM 四叉树改进[J]. 中国图象图形学报，1997，2(8,9)：578～584.

6 Garland M, Heckbert P S. Fast polygonal approximation of terrains and height fields [R]. Comp. Sci. Dept., Carnegie Mellon University. 1995. 9, Technical Report CMU-CS: 95～181.

尤克非 1977 年生，1999 年获华中科技大学电子与信息工程系信息工程学士学位，现为华中科技大学电子与信息工程系硕士研究生. 研究方向为三维仿真、三维数据压缩、虚拟现实.

明德烈 1974 年生，1999 年获武汉理工大学计算机科学系计算机及应用硕士学位，现为华中科技大学图像识别与人工智能研究所模式识别与智能系统专业博士研究生. 研究方向为图象处理、模式识别、虚拟现实及增强现实.

王广君 1964 年生，现为华中科技大学图像识别与人工智能研究所图像信息处理与智能控制教育部重点实验室副教授. 研究方向为图象处理、模式识别与智能控制、数字系统设计.

田金文 1961 年生，现为华中科技大学图象识别与人工智能研究所图像信息处理与智能控制教育部重点实验室教授. 研究方向为小波理论、图象压缩、计算机视觉.

柳 健 1939 年生，现为华中科技大学电子与信息工程系，华中科技大学图像识别与人工智能研究所图像信息处理与智能控制教育部重点实验室教授，博士生导师. 研究方向为遥感图象处理、计算机视觉.