ROAM算法原理及其可视化应用研究

赵慧中

(河南理工大学 测绘与国土信息工程学院,河南 焦作 454000)

摘 要:3维可视化是GIS、计算机仿真、虚拟现实等领域中的关键技术之一,而基于多层次细节的实时优化自适应网格动态地形渲染算法(ROAM)凭借其简单性和可扩展性成为解决海量高程数据地形可视化的常用方法。本文详细剖析了ROAM算法的原理及其特点,并给出了该算法的一个具体的实现过程。

关键词:地形渲染;ROAM;层次细节

中图分类号:P208 **文献标识码**:B **文章编号**:1672-5867(2010)05-0124-03

Research on Theory and Visual Application of ROAM Algorithm

ZHAO Hui-zhong

(School of Surveying and Land Information Engineering, Henan Polytechnic University, Jiaozuo 454000, China)

Abstract: Three-dimensional visualization is one of the key technologies in GIS, computer simulation, virtual reality. Because of simplicity and expansibility, the Real-time Optimally Adapting Meshes (ROAM) algorithm, based on multiple levels of detail, has become a common method to solve the problem of massive terrain elevation data visualization. This paper analyzes the ROAM algorithm theory and features, and gives a concrete implementation.

Key words: terrain rendering; ROAM; levels of detail

0 引 言

地形可视化一直是计算机图形学的研究热点,并广泛应用于地理信息系统、计算机仿真、虚拟现实等领域[1]。通常的地形算法是通过高程数据构建三角网来逼近真实的地形。但是地形是自然界复杂的景物,要想生成3维真实感的地形,就需要大量的数据。庞大的数据集对于目前的图形硬件设备而言是极大的压力。因此,为了减轻硬件的压力,以模型简化技术、多分辨率技术和细节层次技术为基础的实时地形可视化算法就成为近年来研究的主要方向。ROAM算法就是经典的研究成果,它能够根据视点的位置和模型的起伏形状而动态地计算模型的细节层次,减少每帧渲染多边形的数量,并具有较高的图像质量[2]。

对于规则格网表示地形的实时可视化,Lindstrom和Duchainean分别提出了基于四叉树的自适应四叉树算法和基于二叉树的ROAM算法[2-3],他们本质上都是一种层次树结构,根据视点到地面的距离确定不同区域需要绘制的层次,这两种方法已成为规则格网地形的重要算法。下面将着重介绍ROAM算法的思想及其实现方法。

1 ROAM算法思想

ROAM(Real-time Optimally Adapting Meshes,简称ROAM)算法即基于面层次模型算法[2],其基本思想是:在对地形进行3维显示时,根据视点的位置和视线的方向来计算视点距离地形表面的三角片元的距离,再根据目标格网的空间粗糙程度来判断是否对地形表面的三角片元进行一系列基于三角型二叉分割的分解和合并,最终生成逼近真实地形的无缝无重叠的简化连续三角化地形表面[4-7]。

等腰直角三角形是ROAM算法的基本数据单元。通过从其直角顶点到斜边递归的二叉剖分,形成具有层次结构的二元三角树。它使用这些二元三角树来保持三角坐标而不是存储一个巨大的三角形坐标数组来描绘地形结构。这里的每一个小块(Patch)都是一个简单的正二等边三角形。从三角形的顶点到其斜边的中点对其进行分解,可生成两个新的正二等边三角形。这个分解过程是递归的并且可以被子三角形重复直到达到希望的细节等级。ROAM分割成小方块的速度很快,而且可以动态更新高度图。如图1所示的是二元三角树的最初几个结构等级[8]。

收稿日期:2010-05-17

作者简介:赵慧中(1984-),女,河南周口人,地图学与地理信息系统在读硕士研究生,主要研究方向为地理信息服务。

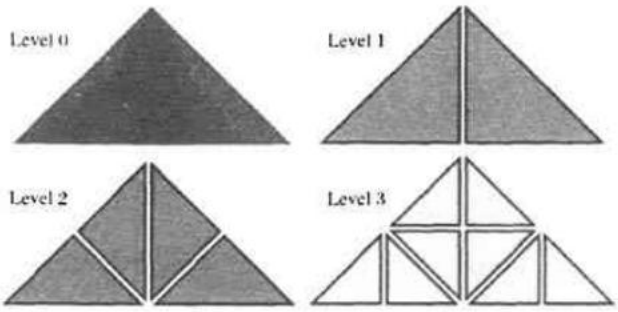


图 1　二元三角树的结构等级

Fig. 1　The structure level of Binary Triangle Trees

在算法中我们用 TriTreeNode 结构来保存二元三角树,同时还追踪 ROAM 所需要的 5 个最基本的邻接关系,即 5 个指向其他树节点的指针。其中两个指向自己的儿子节点,另外 3 个分别指向其邻居节点,如图 2 所示。

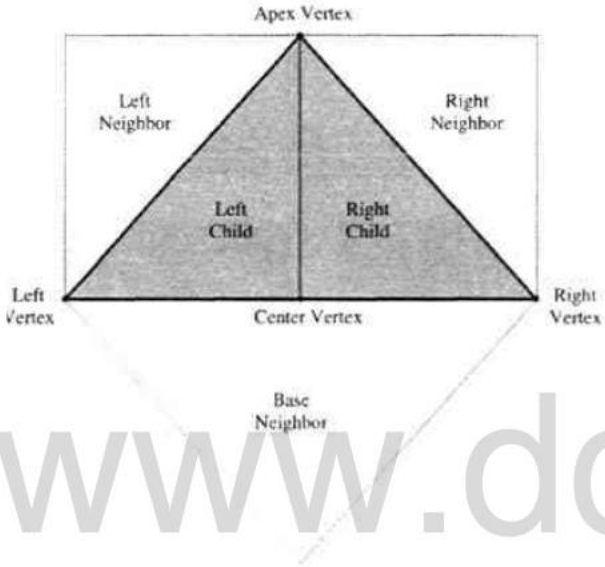


图 2　二元三角树节点的 5 个指针

Fig. 2　The five pointers of the nodes of Binary Triangle Trees

TriTreeNode 结构可定义如下:

```
struct TriTreeNode
{
    TriTreeNode * LeftChild;
//左孩子
    TriTreeNode * RightChild;
//右孩子
    TriTreeNode * BaseNeighbor;
//基邻居
    TriTreeNode * LeftNeighbor;
//左邻居
    TriTreeNode * RightNeighbor;
//右邻居
};
```

世界坐标空间中的网格体是通过对每一个二叉树的节点赋以相应的世界坐标位置 $w(v)$ 形成的。当任意两个二叉树在公共顶点或边界处互不叠置时,一个二叉树三角形就形成了一个连续的网格体。这种连续的网格体就是二叉树三角形网格体或简单三角形网格体。图 3 是显示三角形网格体中一个三角形 T 的典型的邻接关系。T_B 是其底部邻接三角形,共享底边(v_0,),T_L 是其左邻接三角形,共享左边(,v_0),T_R 是其右邻接三角形,共享右边(v_1,v_a)。

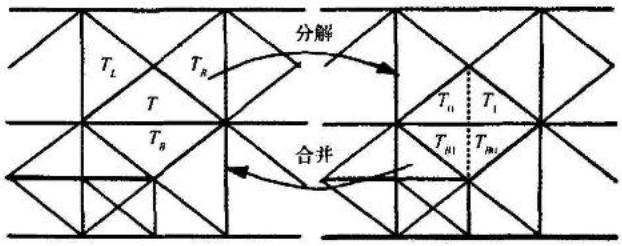


图 3　二叉树三角网的分解与合并

Fig. 3　The decomposition and consolidation of Binary Triangle Trees

二叉树三角形风格的关键是邻接三角形既可与 T 处于同一个二叉树层次,也可以是左和右邻接三角形的下一层级 $L+1$,或者底部邻接三角形的上一层级 $L-1$。所有这些可能的关系都在图 3 的三角形中描述出来。

2　裂缝的处理

几乎所有的 LOD 算法都面临裂缝问题,在 ROAM 算法中也不例外。引起裂缝的主要原因是不连续的侵害在 patch 边界造成的。

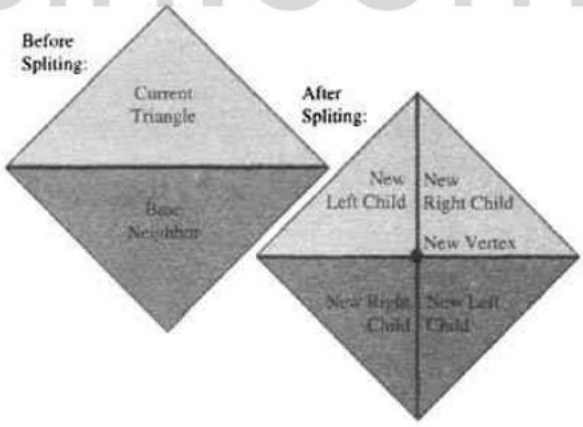


图 4　在钻石上进行分割操作

Fig. 4　The splitting operation on diamond

为了解决这个问题,我们可以运用 ROAM 网格本身关于邻节点的一个规律:一个细节节点和它的邻节点只存在两种关系:共直角边关系(如左右邻节点)和共斜边关系(如下邻节点),我们可以把这个原理应用到建立网格上,以保持可信的树与我们同步。我们来看一下如何

使用这个规则:对于一个节点,我们只在它与它的下邻节点呈相互下邻关系时才实行分割(如图 4 所示),这个关系可以把它当作一个钻石来看,这样形容是因为在钻石上分割一个节点可以很容易地镜像到其他节点,因此在网格上不会出现裂缝。

当我们需要分割一个节点时存在 3 种可能:①节点是钻石的一部分——分割它和它的下邻节点。②节点是网格的边——只分割这个节点。③节点不是钻石的一部分——强制分割下邻节点。

强制分割指的是递归的遍历整个网格直到发现钻石的节点或网格边。工作流程如下:当分割一个节点时,首先看是不是钻石的一部分,如果不是,然后在下邻节点上调用第二个分割操作建立一个钻石,然后继续最初的分割。第二个分割操作将做同样的工作,重复处理下一个节点,一旦一个节点被发现可以递归的分割,就一直分割下去,如图 5 所示。

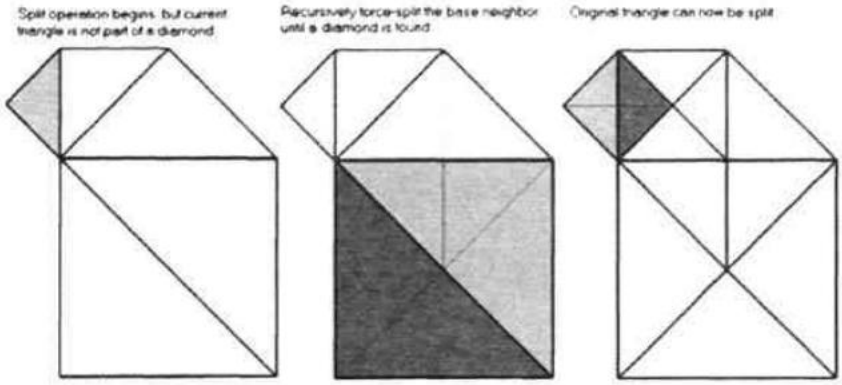


图 5 强制分割操作

Fig. 5 The force – split operation

用上面提到的方法所得到的渲染结果是导致离视点近的区域有比较高的层次细节,稍远区域的层次细节递减,这比较符合人的视觉特性,同时也避免了裂缝的产生。

3 ROAM 算法的实现

ROAM 算法包括 3 个部分,核心为 Patch 和 Landscape 两个类和一个 TriTreeNode 结构。TriTreeNode 结构在前面已经提过,下面对这两个类的功能及一些操作进行说明。

1) Patch 类是这个算法的灵魂,可以分为两个部分,一半是递归部分,另一半是基本函数部分,它包括 Init( ) 函数,需要高度图和世界坐标的偏移值,他们用来对地形进行绽放,指向高度图的指针已经经过调整,指向了这个 Patch 物体所需要数据的第一个字节。Reste( ) 函数释放所有无用的 TriTreeNode 结构,接着重新连接两个二元三角树成为一个 Patch,每一个 Patch 物体都有两个单独的二元三角树构成一个正方形,Tessellate( ) 函数简单的传递适当的高级三角形参数(每一个 Patch 物体的两个根节点)给一个递归版本的函数,函数 Render( ) 和 ComputeVariance( ) 也是这样。

2) Landscape 类管理大的地形块,协调块与块之间的沟通。它相当于地形渲染的所有细节的封装体。它主要包括 Init( ) 函数,用来初始化所有的地形块;Tessellate( ) 函数,用来分割地形块以生成近似网格;Render( ) 函数,用来渲染每个地形块,并调整 gFrameVariance 的值;Reset( ) 函数,用来在每帧渲染的开始重新设置每个地形块,并在需要的时候重新计算变差值。

讨论完 ROAM 算法的实现之后,我们就可以在 PC 机上进行实现,效果如图 6 所示。

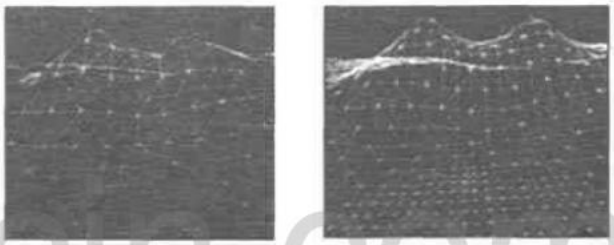


图 6 低 LOD 和高 LOD 的地形网格

Fig. 6 The terrain grid of low LOD and high LOD

4 结束语

本文详细阐述了 ROAM 算法的基本思想及网格生成过程中裂缝的处理问题,并在 PC 机上加以实现。可以看出,同一幅图中,离视点近的地形块中有比较多的三角形,因而具有较高的层次细节,层次细节随着离视点的距离的增加而递减;同时起伏程度比较大的区域比平坦的区域拥有较高的层次细节。由此可以看出,ROAM 技术能够大量减少地形可视化系统中渲染三角形的数量,并可以较好地接近真实地形。该技术可以广泛地应用于 3 维可视化、仿真领域中。

参考文献:

[1] 蔡兴泉,李凤霞,战守义. 动态地形可视化算法研究[J]. 计算机工程与应用,2005,41(1):36 – 37.

[2] Duchaineau M. ROAMing Terrain: Real – Time Optimally Adapting Meshes[C] // Proc of the Conf on Visualization [C]. 1997,81 – 88.

(下转第 129 页)

是光流的垂直部分。

3　试验结果与分析

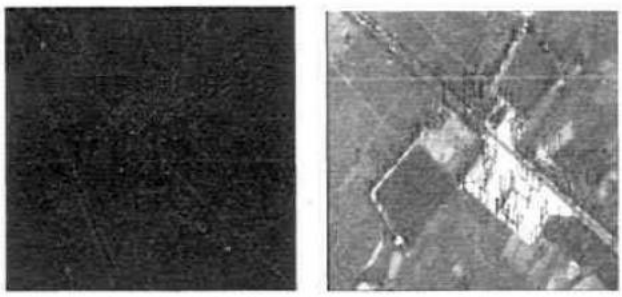
利用该函数在 VC++6.0 的开发环境下就可以实现该算法对图像的运动估计。图 3 是相邻两帧图像:

第一帧影像　　第二帧影像

图 3　无人机视频序列影像

Fig. 3　The UAV sequent video images

图 4 是用金字塔 Lucas_Kanade 算法得到的光流场图像:



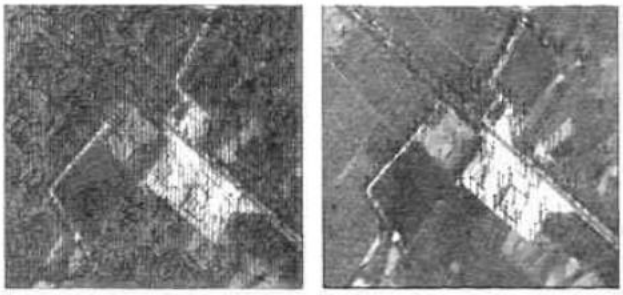
不用金字塔光流场　　6层金字塔光流场

图 4　光流场对比 1

Fig. 4　The first contrast of optical flow fields

通过对相邻两帧图像计算的光流比较分析得出,对于场景中特征比较明显(图像中比较明显的边缘、角点)的图像,在经过 6 层金字塔图像处理后,光流达到稳定效果。这表明金字塔 Lucas_Kanade 算法的光流场的方向比较接近图像的运动方向,运动估计较好。而不使用金字塔 Lucas_Kanade 算法在特征点匹配上会出现一些误差,运动的方向稍显得比较零乱,运动估计的效果不如前者。

用块匹配算法获得的光流场与金字塔 Lk 算法获得的光流场比较效果如图 5 所示:



块匹配法　　金字塔LK算法

图 5　光流场对比 2

Fig. 5　The second contrast of optical flow fields

在实验中块匹配算法计算的速度相比较 LK 算法慢得多。从上面的光流场图像来看,取得的效果也比较差,只能大致地辨别图像的运动方向。可以这样解释,如前文所说,块匹配的运动计算是在块级上进行的,所以输出结果图像的坐标是对块而言的,而不是原始影像的每个像素。

4　结束语

本文在详细分析了光流算法的基础上,采用了 OpenCV 库,在 VC++ 6.0 的环境下,实现了金字塔的 Lucas_Kanade 算法和块匹配算法,并对实验的结果进行了比较。实验表明在特征点比较明显的情况下,金字塔 Lucas_Kanade 算法取得了较好的成果。在 OpenCV 库函数下,根据自己所要实现的功能选择我们所需的库函数,能够大大地减少在计算机视觉领域中编程的时间,缩短了开发周期,提高了工作效率。

参考文献:

[1] 刘瑞祯,于仕琪. OpenCV 教程基础篇[M]. 北京:北京航空航天大学出版社,2007.

[2] 贾云得. 机器视觉[M]. 北京:科学出版社,2000.

[3] 陈胜勇,刘盛. 基于 OpenCV 的计算机视觉技术实现[M]. 北京:科学出版社,2008.

[4] Grary Bradski, Adrian Kaebler. Learning OpenCV[M]. 南京:东南大学出版社,2009.

[5] 聂伟乐,瞿建荣. 基于 OpenCV 的运动目标光流算法仿真[J]. 应用光学,2008,29(6):867-869.

[编辑:胡　雪]

(上接第 126 页)

[3] LINSTROM. Real-Time Continuous Level of Detail Rendering of Height Fields[J]. SIGGRAPH, 1996(8):109-118.

[4] Vincent L, Soile P. Watersheds in DigitM Spaces: An Efficient Algorithm Based on Immersion Simulations[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1991, 13(6):583-598.

[5] 涂超. ROAM 算法原理及其应用研究[J]. 辽宁工程技术大学学报,2003,22(2):176-179.

[6] 柯希林. 基于 RSG 虚拟地形环境的 ROAM 实时绘制[J]. 测绘科学与工程,2003,24(1):38-41.

[7] [美]Shreiner D. OpenGL 参考手册[M]. 孙守迁,王剑,林宗楷,等译. 北京:机械工业出版社,2001.

[8] 任远红,杨克俭. 视点相关的动态 ROAM 算法研究[J]. 计算机与数字工程. 2007,35(5):17-19.

[编辑:宋丽茹]