第46卷 第12期
2016年12月

中国海洋大学学报
PERIODICAL OF OCEAN UNIVERSITY OF CHINA

46(12):142~150
Dec., 2016

海底地形实时绘制技术研究和实现*

王晨明[1]，苏天赟[2]，王国宇[3]，宋庆磊[2]

(1. 青岛朗讯科技通讯设备有限公司，山东 青岛 266100；2. 国家海洋局第一海洋研究所，山东 青岛 266061；
3. 中国海洋大学信息科学与工程学院，山东 青岛 266100)

摘 要： 为了提高大规模海底地形的绘制效率，LOD(细节层次，Levels of Details)技术必不可少。在ROAM(Real-time Optimally Adapting Meshes，实时优化自适应网格)算法的基础上，通过数据加载、视域剪裁、建立评价方法等技术手段，根据视点位置实时更新可视区域，避免了多余三角面片的生成和绘制。同时，采用对不共斜边节点强制分割的方法处理裂缝问题，通过索引坐标与实际坐标转换以及无效值处理实现任意范围海底地形对ROAM算法的应用，消除了传统ROAM算法对数据网格大小的限制，保证了绘制的效果和正确性。最后，通过GPU实时计算和绘制各顶点的法线和颜色，实现了大规模海底地形的实时建模和高效绘制，满足了高精度、海量海底地形漫游浏览的需求，特别是针对起伏比较大的地形漫游浏览。

关键词： 细节层次(LOD)；二叉树；实时优化自适应网格(ROAM)；海底地形；评价因子

中图法分类号： P714 **文献标志码：** A **文章编号：** 1672-5174(2016)12-142-09

DOI： 10.16441/j.cnki.hdxb.20140248

引用格式： 王晨明，苏天赟，王国宇，等. 海底地形实时绘制技术研究和实现[J]. 中国海洋大学学报(自然科学版)，2016，46(12)：142-150.

WANG Chen-Ming，SU Tian-Yun，WANG Guo-Yu，et al. Research and implementation on real-time undersea terrain visualization technology[J]. Periodical of Ocean University of China，2016，46(12)：142-150.

随着陆地资源的急剧减少和能源价格的不断攀升，人们将视线转移到了海洋。海底地形也是如大陆地形一样复杂多变，不但有高山深谷，而且还有平原丘陵。外貌不但奇特壮观，规模更是非常巨大。目前多种格式的海洋数据以及爆发式增长的信息量，在三维场景渲染时需要占据很大内存空间。由于传统二维电子海图在表现海底地形信息时不够清晰直观，因此，在不影响显示效果的前提下，尽量简化场景模型显得尤为重要。

1976年Clark首次提出了LOD[1]的概念。在距离视点较近的区域，采用较高的细节层次，进行较为逼真的3D渲染；较远的区域，采用较低的细节层次，大致渲染。除了保证画面细节，还能节省计算量。

1991年至今，LOD技术日趋完善。1996年，P. Lindstrom第一次实现了基于视点的连续层次细节，采用了四叉树的边剖分方法表示地形[2]。在此基础上，Duchaineau等提出了ROAM算法，用三角形二叉树表示地形网格[3]。1998年，Hugues Hoppe等在他们早期提出的一种可以从任意网格增减三角形的方法的基础上，提出了基于视向的渐进网格VDPM[4]。2001年，Lindstrom等又提出了基于外存(out-of-core)的大规模地形可视化技术，使用操作系统文件映射的虚拟内存空间来存储地形数据[5]。2003年，陆艳青提出一种对视点预测的预装载策略，根据视点移动方向来预测下一帧视点的位置，绘制当前帧的同时，装载下一帧用到的地形数据[6]。2004年，芮小平等人提出了一种改进的ROAM算法，基于非等腰三角形实现ROAM算法[7]。2005年，Hoppe提出了在geometry clipmaps技术基础上基于GPU的硬件加速的静态多分辨率地形渲染算法[8]。2009年，YotamLivny等提出了另一种基于GPU的四叉树地形算法[9]。在LOD技术的发展过程中，ROAM算法的简单和可扩展性使其成为目前地形渲染中被广泛研究的算法。

ROAM是一种自底向上的基于三角形二叉剖分的算法。由于该算法在帧更新过程中只分裂和合并少量三角形，因此能较快更新地形网格，一度成为LOD地

* 基金项目：海洋公益性行业科研专项(201205001)；国家科技重大专项"大型油气田及煤层气开发"子任务(2011ZX05056-001-01)资助
Supported by The Public Science and Technology Research Funds Projects of Ocean (201205001)；sub-project of "The Important Project of Science and Technology in Developing Great Oil & Gas Field and Coal Bed Gas"(2011ZX05056-001)

收稿日期：2014-10-05；修订日期：2015-08-20

作者简介：王晨明(1990-)，女，硕士生。E-mail：wangchenming1990@163.com

形算法研究的热点。本文在 ROAM 算法的基础上，通过改进数据建模和绘制过程的关键技术，实现了大规模、高精度、任意边界范围海底地形的实时绘制。

1　算法

ROAM 算法中，整个地形由多个二叉树组成。二叉树结构的特点是使用二元三角树来保持三角坐标，每个节点都是一个正二等边三角形。从三角形的顶点到其斜边的中点进行分割，生成 2 个新的等边三角形，这两个三角形构成原节点的 2 个孩子节点。

在算法中每个节点需要包含 5 个指向树中其它节点的指针，分别指向基邻居、左邻居、右邻居、左孩子和右孩子，它们构成了合并与增加三角形操作的基础，也是 ROAM 技术得以实现的关键所在（见图 1）。

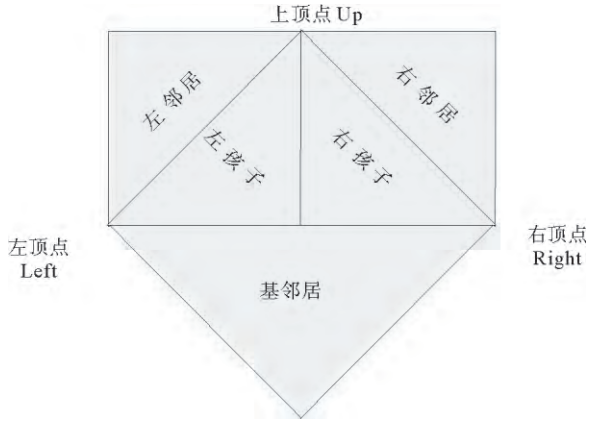


图 1　二元三角树节点的 5 个指针

Fig. 1　The five pointers of triangle bintree

每个二叉树根节点会根据一定准则分割为所需的细节层次，递归分割过程（见图 2）：

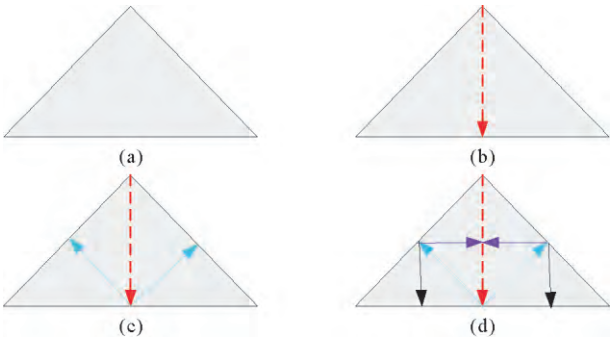


图 2　递归分割过程

Fig. 2　Recursive partitioning process

2　关键技术

采用如下步骤对海底地形进行实时建模：

(1)载入地形数据；

(2)初始化地形，将地形分为 N×N 个小方块 Patch，每个 Patch 由 2 个正三角形组成，每个正三角形为 1 个根节点；

(3)将所有 Patch 互连，建立相互间的邻接关系，斜边处于边界的节点，其基邻居设为 NULL；

(4)获得视点位置，根据视点位置进行视域剪裁，(6)(7)(8)都是只对可见区域作用；

(5)根据视点位置、地形粗糙度和地形块大小获得评价准则，根据这个准则决定细节层次；

(6)递归计算视域内每个节点的误差，若大于阈值，则分割；否则，不分割；

(7)每个节点被分割时进行裂缝处理；

(8)通过 OpenGL 的顶点着色器和片元着色器[11]绘制每个顶点的颜色、法线等，渲染所有二叉树的叶子节点；

(9)操作鼠标和键盘改变视点，通过新视点设置新的可见区域，重复(4)。

相应的海底建模流程图（见图 3）：

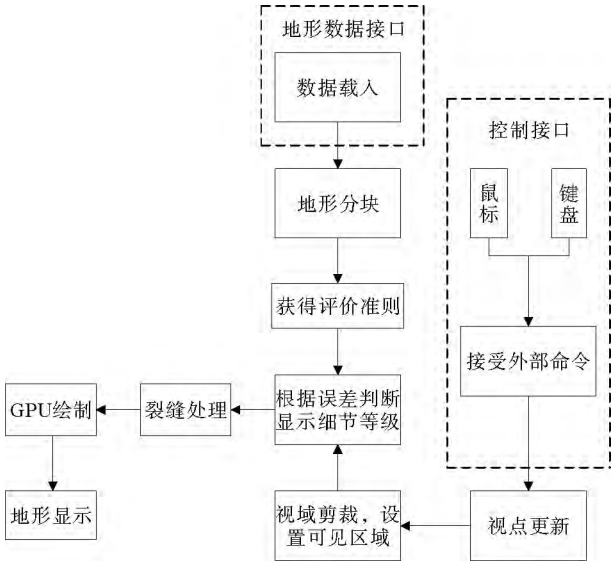


图 3　海底建模流程图

Fig. 3　Seabed modeling flow chart

2.1 数据与处理

ROAM 算法有 2 个限制条件[7]：

(1)地形为正方形；

(2)地形数据量必须为 $(2^n+1)\times(2^n+1)$。

为了便于 ROAM 算法的实施，本文的地形数据采用网格文件进行组织，文件中包含地形要分割的行数、列数、网格大小、地形的起始经纬度、无效值和所有顶点的高度值。

为了提高系统对于数据文件的兼容性，本文对文件读取部分做了改进。读取文件时，记录 Patch 的行列数、宽度、起始经纬度和无效值。在每行顶点高度取出

的同时，系统判断本行数据的个数是否为 2^n+1。若是，继续取出下一行数据，同时将数据放入指针数组；若不是，补充无效值到本行，直到达到 2^n+1 的个数，剩余行也同样处理。通过这种方法，系统可以适用于行列数为任意值的高程数据。

如图 4，本文可将有效数据量为 $x\times y$ 的地形（黄色部分）补充为 $(2^n+1)\times(2^n+1)$。使得算法不再受到传统 ROAM 算法中地形原始数据量必须为 $(2^n+1)\times(2^n+1)$ 的限制。

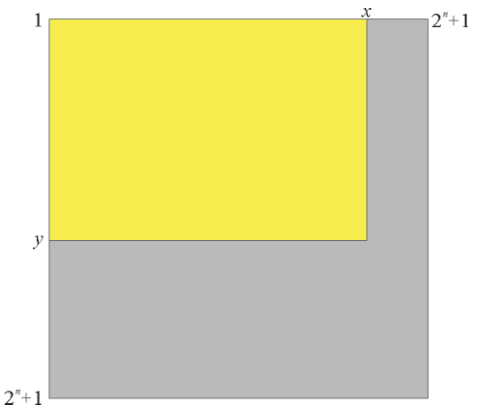


图 4 补充无效值

Fig. 4 Supplement the invalid values

整块地形渲染前，要对地形初始化。把它分成若干个 Patch，对每一个 Patch 沿对角线划分为基本左三角形和基本右三角形。

2.2 视域剪裁

为提高绘制效率，应尽量减少渲染的三角形个数，因此需要进行视域剪裁。

图 5 表示对分块后的地形进行裁剪测试，如果其中心点位于可视范围之外，将其视为不可见；否则，可见。此外，将地形中所有高度为无效值的区域设为不可见。用户可以根据需要自行调整视角大小。根据剪裁规则，采用全局变量 g_bIsVisibile 标识每个 Patch 的可见状态，并且只针对可见区域进行邻接关系建立、节点误差计算、裂缝处理、绘制等操作，以便于提高绘制效率。

判断可见区域的实现方法如下：

令 Patch 的中点坐标为 $patchCenter(x,y)$，则

$$a(x,y)=right(x,y)-eye(x,y)$$

$$b(x,y)=patchCenter(x,y)-eye(x,y)$$

$$result=a(x,y)\times b(x,y)$$

如果 $result>0$，Patch 可见；否则不可见。

2.3 空白区域处理

在传统 ROAM 方法的基础上，为了提高绘制效率，只需要绘制可见区域内的节点。漫游模式下，随着视点的不断变化，在视域两侧会出现没有绘制的空白区域。由于最小的绘制单位为 Patch，在视点移动时，视域内会存在已经被视为不可见 Patch 的一部分，该部分不绘制导致了空白区域的出现。因此，为了提高绘制效果，本文对不可见区域不分割，只绘制出最低细节层次，并且通过建立邻接关系，避免与可见区域细分网格之间出现裂缝（见图 6）。这样，既解决了不可见区域的空白问题，又不影响绘制效率。

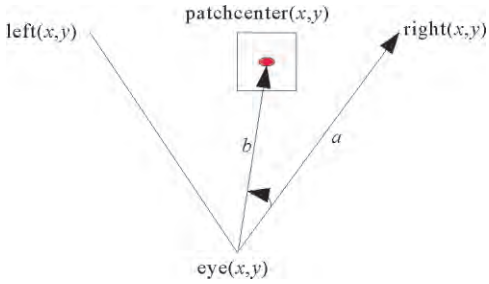


图 5 视域剪裁

Fig. 5 Viewport clipping

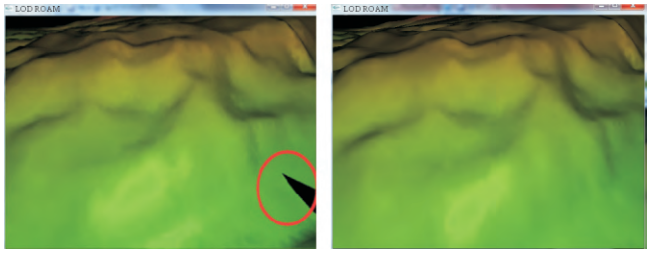


图 6 消除空白区域

Fig. 6 Eliminate the blank area

2.4 评价方法的建立

如何决定某个时刻使用哪个层次细节度是 ROAM 实现的关键。因此，需要建立一个节点评价方法，根据评价因子，设置阈值，实时更新所需的细节层次。

（1）评价因子

评价因子 1：动态视点依赖误差

视点依赖误差是物体在屏幕上的投影大小，通常用观察者到物体的距离度量。距离视点近的区域具有更高精度，精细分割；距离视点远的区域具有较低精度，简单分割或不分割（见图 7）：

节点 M 的动态视点依赖误差的计算方式表示为：

$$d_M=\sqrt{(d_{Mcenterx}-eyex)^2+(d_{Mcentery}-eyey)^2}。$$

其中：$d_{Mcenterx}$、$d_{Mcentery}$ 表示节点斜边中点坐标；eye_x、eye_y 表示视点坐标。如果 $d_M<$ 动态视点依赖误差阈值，分割；否则，不分割。

仅根据节点动态误差分割的效果如图 8，视点位于屏幕下方，动态视点依赖误差阈值分别为 656、196 m，即中心坐标与视点距离小于 656、196 m 的节点分割：

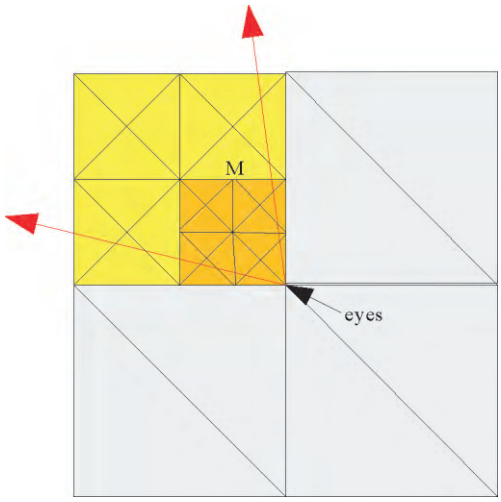


图 7　根据视点分割

Fig. 7　Splitaccording to viewpoint

a.动态视点依赖误差阈值为656m
a.The threshold ofdynamic viewpoint-dependent erroris 656meters

b.动态视点依赖误差阈值为196m
b.The threshold of dynamic viewpoint-dependent erroris 196meters

图 8　仅根据节点与视点的距离分割

Fig. 8　Split according to the viewpoint and the distance

评价因子 2：静态地形起伏误差（即物体的粗糙程度）。

由于地形的复杂程度不同，起伏大的区域需要更多三角形表示。因此要在地形平坦处采用低分辨率三角网，地形粗糙处采用高分辨率三角网。

由于分割一个节点只改变了该三角形斜边中点的高度值，二元三角树中一个节点的局部偏差可以通过该点的插值高度与其实际高度差计算[3]（见图 9）。

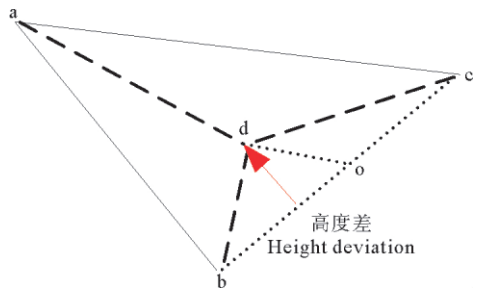


图 9　节点的局部偏差

Fig. 9　Partial deviations of the node

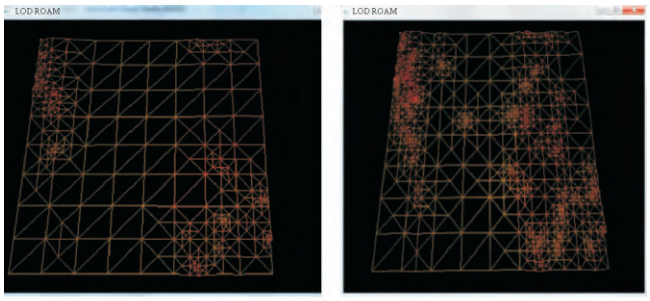
节点静态地形起伏误差的计算方式表示为：

$$h_{\Delta}=h_d-\frac{h_b-h_c}{2},$$

h_b、h_c、h_d 分别表示点 b、c、d 的高度。

如果 $h_{\Delta}>$静态地形起伏误差阈值，分割；否则，不分割。

仅根据节点静态地形起伏误差分割的效果如图 10，地形最大、最小高度差为 1 200 m，静态地形起伏误差阈值分别为 90、150 m：



a.静态地形起伏误差阈值为150m
a.The threshold of static terrainchanges error is 150meters

b.静态地形起伏误差阈值为90m
b.The threshold of static terrainchanges error is 90 meters

图 10　仅根据地形粗糙度分割

Fig. 10　Split according to the terrain roughness

评价因子 3：地形尺寸。

对于较大地形，当观察全部区域时，视点距离整个地形都很远，根据评价因子 1（动态视点依赖误差），只能观察到较低的细节层次。为了满足对大地形的层次显示，将地形的尺寸也作为一个评价因子，用于下一步综合评价。

评价因子 4：帧误差。

当整个场景的三角形数量没有达到或者超过预期的三角形数量时，通过帧误差调节三角形数量，直到达到预期三角形个数。

通过如下公式调节帧误差：

$$FrameVariance=FrameVariance+\frac{Nodes-DesiredTris}{DesiredTris},$$

式中：$FrameVariance$ 是用户最初指定的帧误差，绘制过程中会自行调整；$Nodes$ 是场景中实际的三角形数量；$DesireTris$ 表示期望的场景三角形总数。利用帧之间的相关性，自动调节帧误差，在一定程度上减少了计算量，从而提高绘制速度。

（2）联合 4 个因子，进行综合评价，以判别该节点是否进一步分割。评价公式如下：

$$f=\frac{h_{\Delta}\times size\times k}{d}-FrameVariance,$$

式中：h_{Δ} 是节点的静态地形起伏误差；d 是节点的动态依赖误差；$size$ 是地形尺寸，即地形每行、列的点数；K 是调节因子，通过改变 K 值调整地形分割程度；$FrameVariance$ 是用户刚开始指定的帧误差；通过判断

f 的正负决定节点是否分割，$f>0$ 时三角形分割，$f<=0$ 时不分割。

经测试，建议 $FrameVariance$ 的设定规则为：

$$FrameVariance=(hmax-hmin)/10,$$

其中，$hmax$、$hmin$ 分别为地形的最大和最小高度值。k 的建议取值为 2。

(3)内存控制因子

为防止分割过程中内存溢出，本文采用内存池($PoolSize$)、期望三角形数($DesiredTris$)和误差树深度($VarianceDepth$)等参数控制内存的使用。

所有二叉树节点被存放在内存池中，若节点数大于内存池容量，停止分割；

$DesiredTris$ 为每一帧设置最大能容纳的三角形个数，若个数过多，停止分割。

$VarianceDepth$ 控制二元三角树的误差计算深度，当一个 Patch 中节点数超过 $2^{VarianceDepth}$ 时，误差仍然不满足停止分割的要求，则不再计算误差，而且以节点边长 L 作为停止分割的阈值。本文测试中 $L=8$。

经测试，内存池(PoolSize)的建议取值规则为：

$$PoolSize=\frac{Num_{Patchx}\times Num_{Patchy}\times(h_{max}-h_{min})}{2},$$

Num_{Patchx} 表示地形每行分割的 Patch 数；

Num_{Patchy} 表示地形每列分割的 Patch 数；

h_{max}、h_{min} 分别表示地形的最大和最小高度值；

$DesiredTris$ 的建议取值规则为：$DesiredTris=2.5\times PoolSize$；

$VarianceDepth$ 的建议取值规则为[3]：

$$VarianceDepth=\sqrt{(Patch\_size)}+1,$$

$Patch\_size$ 表示每个 Patch 的边长。

2.5 裂缝问题

二元三角树网格关于邻节点有一个规律，即一个节点和它的邻节点只存在两种关系：共直角边关系和共斜边关系[6]。

如果相邻子块的分辨率不一致，在构建三角形网格时，会因为相邻边界的细节层次不同而出现裂缝(见图 11)。

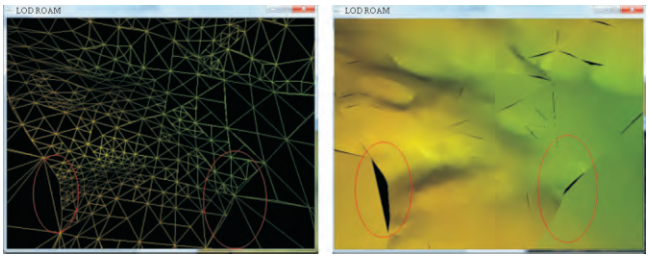


图 11 裂缝

Fig. 11 Cracks

因此，为了避免产生裂缝，分割 2 个构成菱形的节点，称之为“钻石结构(diamond)”[3]。

分割一个节点时存在 3 种可能：

(1)节点和其基邻居互为下邻关系——分割两者(见图 12)；

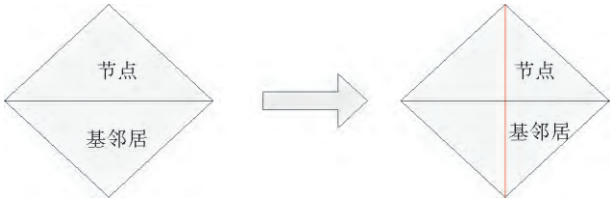


图 12 分割钻石结构

Fig. 12 Split diamond

(2)节点在地形块的边界上——只分割这个节点；

(3)节点不在边界上并与基邻居不为下邻关系——强制分割，直到出现情况(1)或(2)。

强制分割过程为图 13 中的(a)(b)(c)：

当需要剖分黄色三角形时，由于其斜边邻居处于高一层细节层次，必须对其斜边邻居强制剖分，以此类推，直到找到钻石结构，再反方向逐级剖分[3]。

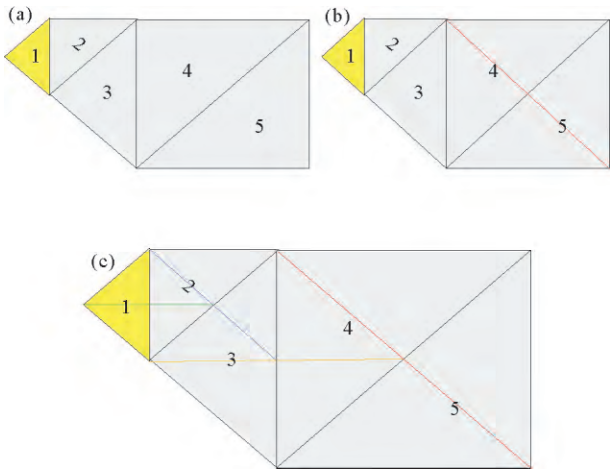


图 13 强制分割过程

Fig. 13 The process of mandatory segmentation

2.6 坐标转换

传统 ROAM 算法要求地形大小为$(2^n+1)\times(2^n+1)$[3]。初始化时，每个 Patch 边长为 1，给算法的应用带来很大局限性。

为打破这种局限性，本文设置了索引坐标 I_x、I_y，用于对 ROAM 算法的地形分块进行索引。同时，设置了 2 个转换函数，分别用来将行和列对应的索引坐标转化为实际坐标，真实再现任意大小的地形。

$$\begin{cases}T_x=I_x\times d_x+I_{minx}\\T_y=I_y\times d_y+I_{miny}\end{cases},$$

T_x、T_y 表示 x、y 的实际坐标；

I_x、I_y 表示 x、y 的索引坐标；

d_x、d_y 表示每个小块的实际长、宽；

I_{minx}、I_{miny} 表示初始绘制点坐标。

这 2 个函数只在模型转换、视点移动、顶点绘制和法线计算时用到。

2.7 绘制

为了减少运行时刻 CPU 的动态三角化负载，本文利用 GPU 进行重建后的地形绘制工作，主要包括法线和颜色的绘制，提高地形的显示效率和效果。

2.7.1 法线　海底地形凹凸不平，必须进行法线计算达到光照下的立体效果。OpenGL 中用表面表示物体，一个物体就是一组平面。光线照射在平面上会产生反射，入射线与反射线的角平分线就是法线，它垂直于平面。

鉴于使用面法线会使整个地形看起来分块比较明显，为使其整体效果更佳，需要计算每个顶点的法线。

OpenGL 中定义面法线和顶点法线（见图 14）：

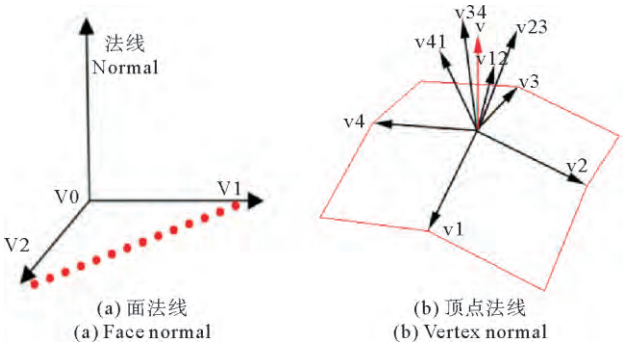


图 14　面法线、顶点法线

Fig. 14　Face normal and vertex normal

如上图所示，根据 V_0、V_1、V_2 确定的平面，可以求出这个平面的 2 条相交线，这两条线叉乘即为所求法线。

$$\begin{cases} V_a = V_1 - V_0 \\ V_b = V_2 - V_0 0 \end{cases},$$

面法线 $V_{normal} = V_a \times V_b$，

顶点法线为包含该顶点三角形面法线之和：

$$V = V_{12} + V_{23} + V_{34} + V_{41},$$

绘制前，定义一个 size×size×3 的法线数组，存放各个顶点的法线，数组的索引是各顶点的索引坐标，size 为地形尺寸，即每行、列的点数。遍历这一帧的所有节点，通过坐标转换函数获得每个节点中 3 个顶点的实际坐标，计算面法线，将面法线累加到这 3 个顶点的法线数组里，从而得到这一帧所有顶点的法线。图 15 的(a)(b)分别展示了不加法线和加法线的效果：

2.7.2 颜色　为达到更明显地形高度差异和更清晰的效果，需要为地形的每个顶点赋予不同颜色值。本文采用高级着色语言 GLSL，实现在 GPU 上对地形不同高度值快速着色。

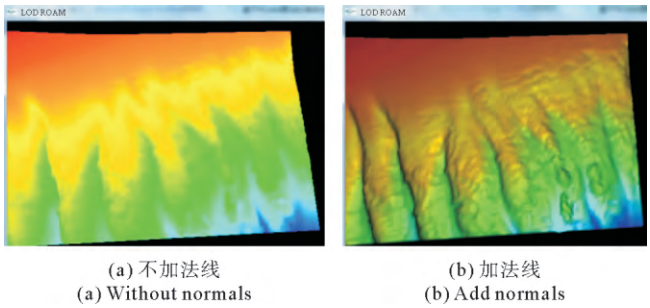


图 15　不加法线和加法线的效果图

Fig. 15　The effect picture about without normal and add normals

着色分为 N 个色段，高度最小值 H_{min} 对应颜色最暗色（蓝色），最大值 H_{max} 对应最亮色（红色），对落在色段之间的高度按比例进行颜色插值。如下式和图 16 所示，高度为 Hi 的顶点落在 M 色段上，M_{min}、M_{max} 的颜色值分别为 $color_{Mmin}$、$color_{Mmax}$，计算 Hi 的颜色值 $color_{Hi}$：

$$color_{Hi} = \frac{H_i - M_{min}}{M_{max} - M_{min}} \times (color_{Mmax} - color_{Mmin}) + color_{Mmin}$$

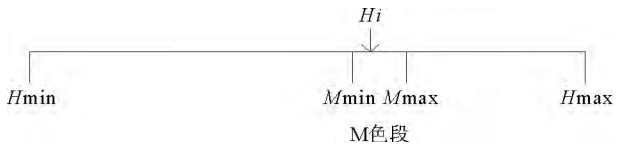


图 16　计算顶点的颜色

Fig. 16　Calculate the vertex color

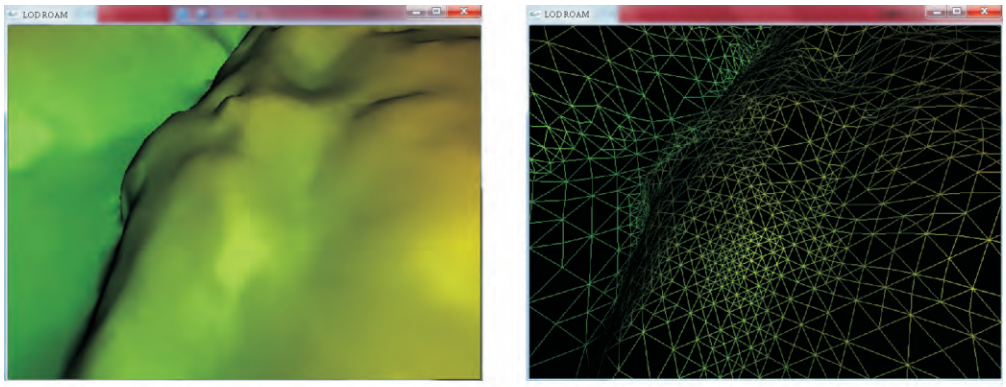
颜色渲染效果（见图 15）。

3　实验结果

为了进行应用测试，本文对研究的算法进行了开发和实现。测试平台为普通计算机，CPU 为 Intel T5670，显卡为 ATI Mobility Radeon X1350，内存为 2G。操作系统为 Windows XP，开发环境为 VisualStudio 2008 C++，底层图形接口采用 OpenGL。

本文使用的数据为南海某峡谷区域，数据分辨率为 10 m，地区大小约为 8.3 万 km^2，存放于网格文件中，网格数为 5 732×2 892，顶点数量超过 10^7。为使地形起伏更明显，绘制时将高度放大 10 倍，显示效果如图 17 所示。

表 1 是在某一地形起伏较大处测试的参数对比。网格大小 PatchSize = 32，地形高度差 $h_{max} - h_{min}$ = 1 322.375，内存池大小 $PoolSize$ = 200 000，期望的三角形总数 $DesiredTris$ = 50 000，最大误差计算深度 $VarianceDepth$ = 12 时，改变帧误差值的效果比较（见表 1）：



(a) 海底漫游效果 Effect of the Ocen Odyssey

(b) 与a对应仅用网格显示的效果 The grid display corresponding to figure a

图 17 测试结果

Fig. 17 Test result

表 1 改变误差的结果表

Table 1 The result of change error

帧误差 Frame variance	初始场景三角形总数 Initialnumber of triangles	初始帧速 Initial frame rate/frame×s^{-1}	帧误差平衡所需时间 Time required for balance of frame error/s	平衡后的帧误差 Frame variance after balance	平衡后的三角形总数 Number of triangles after balance	平衡后的帧速 Frame rate after balance/frame×s^{-1}
450	47 951	21	31	299.92	41 605	20
300	70 494	12	22	296.406	35 820	21
150	104 376	10	15	294.437	36 048	22
100	103 481	10	25	298.496	35 840	21
50	102 040	10	29	299.1	35 736	19
25	102 481	10	35	299.076	35 510	20

表 1 证实了帧误差会自我调整，直到平衡。经测试，上述所有情况平衡后的帧速均能达到 20 帧/s，三角形总数约为 36 000 个。帧误差 $FrameVariance=300$ 时，既能较好绘制地形，又能使三角形总数最快达到稳定。本文建议的评价因子（如期望三角形总数、最大误差计算深度、调节因子 K 和帧误差等）阈值是基于现有数据集测试后的结果，不一定适用于所有地形数据。因此，在使用本文提出的地形建模方法时，需要根据建议值对评价因子进行测试，以确定最佳的评价因子的阈值，来实现海底地形的快速、动态建模。

表 2 为不同地形算法的帧速对比图，内容均为实际实验数据。表二绘制帧速及绘制效果的对比表明，本文方法在绘制效果相同的情况下，绘制速率比传统的 ROAM 方法提高了大约 30%，适用于大规模、任意网格大小海底地形的动态绘制。

表 2 绘制效率对比

Table 2 Drawing efficiency contrast

方法 Method	三角形数目 Triangle Number	帧速 Frame rate /frame×s^{-1}	显示效果 Display result
不使用 LOD Without LOD	102 000	5	好
传统 ROAM Traditional ROAM	45 000～48 000	15～18	有空白区域
本文方法 Our method	35 000～41 000	19～21	好

5 结语

为了提高大规模三维地形的实时绘制效率，本文在 ROAM 算法基础上，提出了一种高效地形实时绘制算法，针对数据加载、视域剪裁、评价方法建立、裂缝处理、GPU 绘制等关键技术进行了研究和实现，并应用在

海底地形的动态绘制中。通过对无效值的处理提高了数据的兼容性，索引坐标到实际坐标的转换消除了对网格大小的限制，解决了不可见区域不参与绘制导致的空白区域问题。通过测试，本方法能够提高大数据量海底地形的绘制效率和显示效果，为海洋工程勘察和科学研究提供高效、直观的海底地形动态浏览服务。

参考文献：

[1] Clark J H. Hierarchical geometric models for visible surface algorithms[J]. Communications of the ACM, 1976, 19(10): 547-554.

[2] Lindstrom P, Koller D, Ribarsky W, et al. Real-time, continuous level of detail rendering of height fields[C]// New York: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM, 1996: 109-118.

[3] Duchaineau M, Wolinsky M, Sigeti D E, et al. ROAMing terrain: real-time optimally adapting meshes[C]// Arizons: Proceedings of the 8th Conference on Visualization'97. IEEE Computer Society Press, 1997: 81-88.

[4] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering[C]// North Carolina: Visualization '98. Proceedings. IEEE, 1998: 35-42.

[5] Lindstrom P. Out-of-core simplification of large polygonal models [C]// New York: Proceedings of the 27th Annual Conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000: 259-262.

[6] 陆艳青. 海量地形数据实时绘制的技术研究[D]. 杭州：浙江大学，2003.
Lu Y Q. Study of the Real-Time Rendering for Large-Scale Terrain Dataset[D]. A Dissertation Presented to the Graduate School of Zhejiang University in Partial Fulfillment of the Requirement for the Degree of Doctor of Philosophy, 2003.

[7] 芮小平，杨崇俊，张彦敏. 一种改进的 ROAM 算法[J]. 武汉大学学报(信息科学版)，2005，29(11)：1008-1011.
Rui X. An improved LOD algorithm based on ROAM[J]. Editorial Board of Geomatics& Information Science of Wuhan University, 2005.

[8] Asirvatham A, Hoppe H. Terrain rendering using GPU-based geometry Clipmaps[J]. GPU gems, 2005, 2(2): 27-46.

[9] Livny Y, Kogan Z, El-Sana J. Seamless patches for GPU-based terrain rendering[J]. The Visual Computer, 2009, 25(3): 197-208.

[10] Woo M, Neider J. OpenGL 编程指南（第四版）[M]. 北京：人民邮电出版社，2005.
Woo M, Neider J. Opengl Architecture Review Board[M]. Beijing: PosTs & TELECOM PRESS, 2005.

[11] Wright Jr R S, 赖特, Lipchak B, et al. OpenGL 超级宝典[M]. 北京：人民邮电出版社，2005.
Wright R S, Lipchak B. OpenGL Superbible[M]. Beijing: PosTs & TELECOM PRESS, 2005.

Research and Implementation on Real-Time Undersea Terrain Visualization Technology

WANG Chen-Ming[1], SU Tian-Yun[2], WANG Guo-Yu[3], SONG Qing-Lei[2]

(1. Alcatel-Lucent Qingdao, Qingdao 266100, China; 2. The First Institute of Oceanography, SOA, Qingdao 266061, China; 3. College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

Abstract: With the rapid development of computer graphics, the three-dimensional terrain modeling and visualization are widely researched at home and abroad. The speed of real-time rendering becomes the key factor to determine whether a three-dimensional visualization system is good. But the large amount of data has became the huge obstacles of three-dimensional visualization. Scholars at home and abroad used LOD (Level of Detail) technology widely to solve this problem. The main function of this technology is to simplify the part of flat terrain and distant objects, to reduce the number of drawing primitives, to improve the rendering efficiency under the premise of ensuring visual effects. The content of this paper is study the draw technology of large-scale three-dimensional terrain. It introduces the research status and development in the relevant field at home and abroad. LOD (Levels of Details) technology is essential on improving the efficiency of large scale terrain. The main purpose of this project is to study the seabed terrain rendering progress at home and abroad, to analyze the pros and cons of different ways, to obtain feasible solutions, to optimize programs on this basis, at last, to improve the rendering efficiency and to meet the authenticity. Based on ROAM(Real-time Optimally Adapting Meshes) algorithm, the display area can be updated in real-time according to the viewpoint position by means of data source processing, field crop and the establishment of evaluation methods, which can avoid rendering superfluous triangular facets. At the same time, the nodes, which have different hypotenuses, are compulsively split to solve the problem of crack. Through transforming from the indexical coordinate to actual coordinate and utilizing the NONE data value, then any scope of undersea terrain can be modeled and visualized based on ROAM, which can clear up the limit of mesh size and guarantee the rendering effect and correctness. By computing and rendering each vertex's normal and color with GPU, we accomplish the goal of large-scale terrain modeling and efficient rendering in real-time, which meets the needs of the high precision, massive undersea terrain roaming, especially for the terrain which ups and downs.

Key words: levels of details; binary tree; ROAM; undersea terrain; evaluation factors

责任编辑 陈呈超