非封闭三角网模型的简化与分辨率控制*)

Simplification and Resolution Control of Unclosed Triangle Mesh Models

冯 洁 查红彬

(北京大学视觉与听觉信息处理国家重点实验室 北京 100871)

摘 要 在某些应用领域中,我们得到的三维数据不可避免地存在边界和漏洞。为了对这种非封闭的模型进行有效的简化和分辨率控制,本文实现了一种基于渐进网格的分辨率控制方法。这种方法针对不同的边界条件采取了不同的简化和重建策略,从而在简化过程中较好地保存了边界部分的信息。并且在浏览模型时,可以通过选择性细化使模型的不同部分具有不同的分辨率,从而快速地显示浏览者所关心的局部细节。

关键词 三维数据,非封闭三角网模型,分辨率控制

1. 引言

一般在使用三角形网格来描述复杂的表面时,网格的数据量会非常庞大,这给三维表面的绘制、浏览和传输都带来许多困难。解决这些问题的一个方法就是根据不同的需求在同一个三角网格模型的不同区域中使用不同的分辨率,如:对于物体中靠近观察者的部分,采用较高分辨率的网格表示,而离观察者远的部分则采用较低分辨率的网格表示。

在已有的文献中可以见到许多种不同的多分辨率模型。它们的共同特点是先对网格进行局部分辨率调整,然后将调整的记录存放在某种数据结构中,最后在绘制模型时根据需要从该数据结构中提取相关的调整记录以达到分辨率控制的目的。例如,Floriani 等提出的多分辨率地形模型(Multiresolution Terrain Models)和多分辨率三角剖分(Multi-Triangulation)[1,2],又如 Hoppe 提出的渐进网格(Progressive Meshes:PM)[3,4]等就是典型的例子。其他相关的研究还有:Cohen 等的多边形-点混合绘制模型[5]、To 等的多分辨率渐进传输方法[6]及 Guskov 等提出的混合网格(Hybrid Meshes)[7]等。

但是,通常的网格简化和分辨率控制都是针对比较完整的、没有边界和漏洞的模型来进行的。而在某些应用领域中,我们得到的三维数据不可避免地存在边界和漏洞。例如在机器人视觉中,机器人所携带的三维测距仪采集的三维数据通常是不完整的,而且其数据量可能非常大。为了及时提取其中有用的信息,就必须对模型进行简化。而如何对这种大量存在边界的模型进行有效的简化和分辨率控制,从而使机器人能够实时地识别场景中的物体,就成了一个值得研究的问题。

本文中实现了一种基于渐进网格的分辨率控制方法。这种方法更多地强调了在边界条件下的处理。首先,在第 2 节中简要回顾了 PM 的基本原理,并介绍了本文中 PM 的表示方法。第 3 节、第 4 节则分别介绍了 PM 的建立和浏览过程。在第 5 节中给出了一些实验结果,最后就当前的实现方法以及未来的工作做了一些讨论。

2. PM 的表示

本节首先简要回顾 PM 的基本原理,然后介绍本文中对三角形网格和 PM 的表示方法。文中的有关数据结构部分采用 C++语言的记号表示。

2.1 PM 的基本原理

PM 的基本原理是每次从原始网格 M 中删除一条边,逐步将分辨率降低,最后得到一个简化的粗糙网格 M^0 和一系列细节信息记录。根据这一系列的细节信息记录,重新向网格中插入结点和三角形,就可以恢复出具有原始分辨率的模型。因此在实现分辨率控制的同时,PM 也可达到数据压缩的目的。

从 M 到 M^0 的简化过程是通过称为"边坍缩(edge collapse)"的操作实现的,即:从网格中删除某条边(v_l,v_r),并且将两顶点 v_l、v_r 融合形成一个新结点,与这条边相邻的两个三角形 f_l、f_r 也一并被删除,从而降低了网格的分辨率。(见图 1)

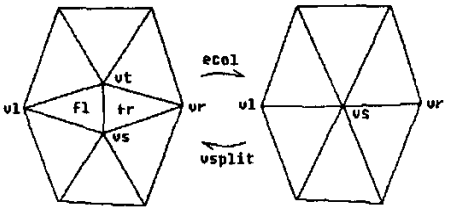

图 1 边坍缩与点分裂操作

相反,从 M^0 重建出 M 的过程是由"点分裂(vertex split)"操作完成的。这恰是边坍缩操作的逆过程,它向网格中添加了一个结点 v_t 和两个三角形 f_l 和 f_r。

上述过程可以表示为:

*)本文得到国家 973 课题(G199803606)与高校博士点基金(20010001004)的资助.

简化过程：$(M=M^n)\xrightarrow{ecol_{n-1}}M^{n-1}\cdots M^2\xrightarrow{ecol_1}M^1\xrightarrow{ecol_0}M^0$

重建过程：$M^0\xrightarrow{vsplit_0}M^1\xrightarrow{vsplit_1}M^2\cdots M^{n-1}\xrightarrow{vsplit_{n-1}}(M^n=M)$

于是，$\{M^0,\{vsplit_0,\cdots,vsplit_{n-1}\}\}$就组成了 M 的一个 PM 表示。

2.2 三角形网格的表示

三角形网格的组成要素有结点和面(三角形)。结点记录了表面上各顶点的空间坐标，而三角形则决定了整个表面的拓扑形状，即各顶点间的相邻关系。图 2 给出了三角形网格的一种表示方法。

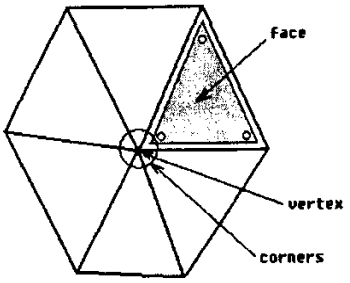

图 2 三角形网格结点的结构

对于结点，除了记录该结点的空间三维坐标以外，还记录了与该结点邻接的三角形的信息：首先，我们称一个结点 v 及其相邻三角形 f 构成的二元组 (v,f) 为一个 *corner*，于是与结点 v 相邻的三角形集合记为 $v.corners$。与结点相比，面(三角形)的结构非常简单，仅存放了该三角形三个顶点的索引号。

在定义了上述点、面结构以后，三角网格的结构就变得非常简单，只需用一个结点序列和一个面序列就可以描述出整个网格的几何形状。

此外，三角形网格通常还带有很多其他的信息，例如法向量、颜色、纹理和材质等，它们都可以作为属性信息附加在结点或三角形上。本文中暂时不考虑法向量以外的属性信息。

2.3 PM 的表示

如 2.1 节中所述，一个 PM 是由一个简化的、粗糙的初始网格 M^0 和一组点分裂记录的序列组成的，即：$PM=\{M^0,\{vsplit_0,\cdots,vsplit_{n-1}\}\}$。这里，$M^0$ 可以用前面介绍的三角形网格描述，而点分裂记录则存储在一个有序的链表中。

为了在网格重建时正确地恢复出原有的拓扑结构，必须在点分裂记录中保存如下的信息：

- 在边坍缩中被删除的边(v_s,v_t)；
- 与边(v_s,v_t)相关的两个结点 v_l 和 v_r；
- 点分裂过程中 v_s 点的坐标增量 $delta\_vs$；
- 点分裂的类型 $split\_type$。

如图 1 所示，我们总是假设在一次边坍缩中被删除的边是(v_s,v_t)，而 v_s、v_t 融合后所得的新结点仍记为 v_s。v_l、v_r 是与点分裂密切相关的两个结点，进行点分裂时，需要根据这两个结点来决定插入新三角形的位置。当(v_s,v_t)是边界时，总是假设 v_r 和 f_r 不存在。最后一项的点分裂类型非常重要，在执行边坍缩的逆过程-点分裂操作时将根据这一参数来恢复 v_s、v_t 邻域的三角形相邻关系。

点分裂的分类如图 3 所示。其中，在 0 型和 1 型这两种情况下，边(v_l,v_s')和(v_s',v_r)都处于边界上，因此恢复邻接关系的操作将非常简单；相反，在 2 型和 3 型这两种情况下，边(v_l,v_s')和(v_s',v_r)都不是边界(3 型中(v_s',v_r)不存在)，这时我们将用一种平面切割的方法来恢复邻接关系；而 4 型和 5 型是相对比较少见的情况，边(v_l,v_s')和(v_s',v_r)中有一条在边界上，而另一条不是，这时将采用局部搜索的方法来恢复邻接关系。本文中暂时不考虑这六种类型之外的情况，因为在网格简化的过程中，总是可以通过特定的判断来避免这些例外情况的出现。详细的算法在 4.1 节中再作叙述。

3. PM 的建立——网格的简化

要建立 PM，首先要对原始网格 M 进行简化，即进行一系列的边坍缩操作，以得到初始网格 M^0。在每一步边坍缩的过程中，需将被删除的三角形的信息保存下来，以便在网格重建过程中加以恢复。网格简化的基本算法如下：

1. 从等待删除的边的集合中找出下一步要删除的边(v_s,v_t)。

2. 删除边(v_s,v_t)，将记录的信息$\{v_s,v_t,v_r,delta_vs,split_type\}$加入点分裂记录序列。

3. 更新待删除边的集合。

4. 如果满足结束条件则退出；否则转 1，继续下一次边坍缩操作。

在上述算法中，如何选择简化的顺序，即如何决定下一步删除哪一条边，是压缩效果好坏的关键。选择方法的一个极端是完全随机地选取下一条要删除的边，而另一个极端是求解最优化方程以找到最优的选择[3,8]。

本文用的是一种折衷的方法，即：为每一条边计算一个权值，然后对每个三角形取其最小的边权为该三角形的权。这样在网格简化的过程中，只要每次找到权值最小的三角形，并删除其权值对应的边即可。也就是说，权值比较小的边将被优先删除。当然，如何定义边的权值又有许多可以选择的方法。一种比较简单的方法是用边长作为权值，即每次删除网格中最短的一条边。从直观上亦容易理解，删除一条较短的边对总体几何形状的影响会比较小。但是，最短边策略会使网格中的边长趋于平均，从而不利于保存表面的几何特征，因此我们对这一策略做了一些改进。

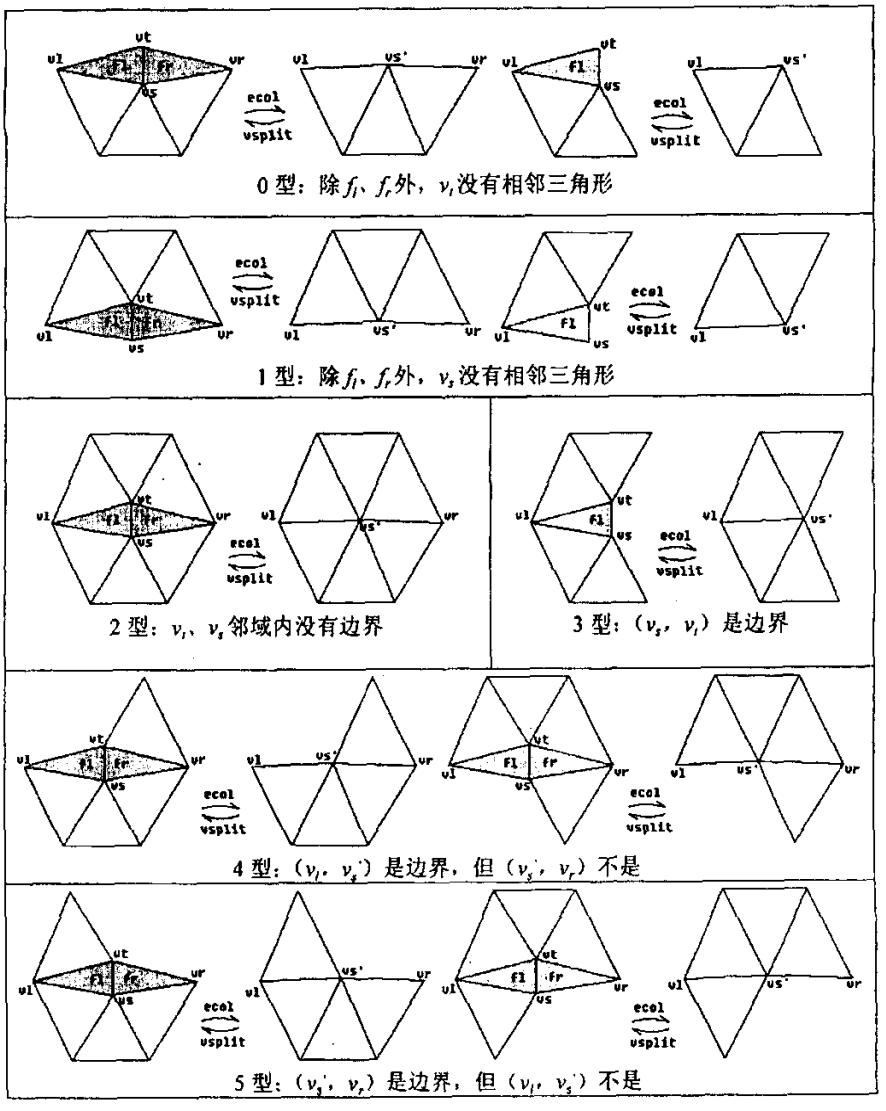

图 3　点分裂类型

首先，假设 $\mathbf{n}_i$ 是 v_i 的法向量，$\mathbf{n}_t$ 是 v_t 点的法向量，记：

$$c_{t,i}=\cos\langle \mathbf{n}_t,\mathbf{n}_i\rangle=\frac{\mathbf{n}_t\cdot\mathbf{n}_i}{|\mathbf{n}_t||\mathbf{n}_i|} \tag{1}$$

于是，边(v_t,v_t)的权值可以定义为：

$$w_{s,t}=l_{s,t}-\min_{v_i\in v_t.\,corners} c_{t,i} \tag{2}$$

这里，$l_{s,t}$表示边(v_s,v_t)的欧式长度。上式中第二项体现了 v_t 点与其相邻点之间法向量的最大夹角。这个夹角越大，说明 v_t 点附近几何形状变化越大，于是这条边的删除就更应该置后。另外，由于第二项的取值总是在[−1,1]中，为防止在边长很大的情况下第二项失去作用，对上式修改如下：

$$\begin{aligned} w_{s,t}&=l_{s,t}-\min_{v_i\in v_t.\,corners}(c_{t,i}\cdot l_{s,t})\\ &=l_{s,t}\cdot(1-\min_{v_i\in v_t.\,corners} c_{t,i}) \end{aligned} \tag{3}$$

这样，网格的几何特征就得到了较好的保存。但是这一策略有时会使网格中产生一些狭长的三角形，为了消除这些狭长三角形，再对权值做进一步修改：

$$w_{s,t}=l_{s,t}\cdot(1-\min_{v_i\in v_t.\,corners} c_{t,i}+\frac{l_{\min}}{l_{\max}}) \tag{4}$$

其中 $l_{\min}$表示三角形中最短边长度，$l_{\max}$则表示最长边长度。于是，三角形越是狭长，该项比例就越小，被删除的优先度就越高。如果给这三项加上不同的系数，就可以进一步改变各项对总权值的影响大小。

与 Hoppe 的求解最优化网格的方法相比，这种给三角形加权的方法在计算上更简单，而且只要合理地定义三角形的权值，也可以得到很好的简化效果。

另外，上述算法中每一次循环都要寻找下一条要删除的边，假如每次都进行全局搜索将非常耗时。

因此我们在算法开始前就对每个三角形计算权值并存入一个排序表中，这样在算法第 1 步中就可以直接从表中读出权值最小的三角形。当然，每做完一次边坍缩，必须更新与 v_s 相邻的三角形的权值，同时调整排序表。但这只是局部调整，时间消耗相对较小。

4. PM 的浏览

经过上述简化过程以后，低分辨率的初始网格 M^0 与点分裂记录序列一起就组成了一个 PM 结构。浏览该 PM 模型时，只要依据点分裂记录序列逐步向网格中插入或删除结点，就可以使网格的分辨率升高或降低。在本节中，首先介绍按点分裂记录序列顺序进行细化和简化的浏览过程，然后再简略介绍选择性细化浏览过程。

4.1 细化浏览——网格重建

细化浏览时，按照点分裂记录序列的顺序依次执行点分裂操作，每次向网格中插入一个新结点和两个三角形(如果是边界的情况，则只加入一个三角形)，就可以使网格的分辨率不断提高，最终可从 M^0 恢复出原始网格 $M=M^n$。点分裂操作的基本算法如下：

1. 从点分裂记录序列中读入下一条分裂记录 $\{v_s, v_l, v_r, delta\_vs, split\_type\}$。

2. 向网格中插入新结点 v_t，并根据 $delta\_vs$ 恢复 v_s 和 v_t 的坐标。

3. 根据 $split\_type$，将适当的三角形从 v_s.$corners$ 移动到 $v_t.corners$，从而恢复 v_s, v_t 周围的几何邻接关系。

4. 向网格中插入两个新的三角形：$f_l=\{v_l, v_s, v_t\}$ 和 $f_r=\{v_r, v_s, v_t\}$(v_r 不存在时不插入 f_r)。

5. 恢复点 v_s 和 v_t 的法向量。

其中第 3 步中要根据参数 $split\_type$ 来恢复 v_s、v_t 邻域内的几何邻接关系，对于不同的点分裂类型，将采取不同的恢复方法：

• 0 型：直接向网格中插入新三角形 f_l 和 f_r 即可，无需改变邻接关系。

• 1 型：只需将 $v_s'.corners$ 中的三角形全部移入 $v_t.corners$ 即可。

• 2 型：这是最常见的一种情况，本文采用一种平面切割方法来决定哪些三角形应当移入 $v_t.corners$ 中。如图 4 所示，设由向量 $\mathbf{n}_1=(v_s', v_l)$ 和 v_s' 点法向量 $\mathbf{n}_s'$ 确定的平面为 P_1，由 $\mathbf{n}_2=(v_s', v_r)$ 和 $\mathbf{n}_s'$ 确定的平面为 P_2。用这两个平面切割 $v_s'.corners$，可以得到图 5 所示的两种情形。在这两种情形下，分别用 $v_s'.corners$ 中每个三角形的中心 p 与 v_s 做比较。在图 5(a)的情况下，应移动到 $v_t.corners$ 的三角形所占的区域(灰色部分)大于应保留的部分(白色部分)。这时只要 p 与 v_s 处于 P_1、P_2 二者之一的异侧，就把该三角形移入 $v_t.corners$ 中。而在图 5(b)的情况下，应移动部分小于应保留部分，因此只有 p 与 v_s 同时处于 P_1 和 P_2 的异侧时才将它移入 $v_t.corners$ 中。

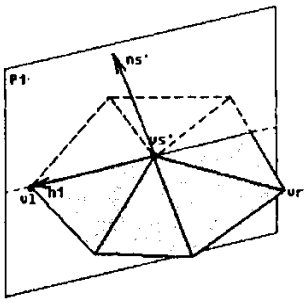

图 4　为执行 2 型点分裂所作的切割平面

• 3 型：与 2 型类似，只是由于 v_r 不存在，因此只需用一个平面 P 做切割，并将 $v_s'.corners$ 中与 v_s 处于 P 异侧的三角形移入 $v_t.corners$ 中。

• 4 型：以 v_s' 为中心，边(v_s', v_r)为半径，按逆时针方向旋转，将该边扫过的三角形移入 $v_t.corners$ 中，直到扫描到边界为止。

• 5 型：与 4 型类似，只是以 v_s' 为中心，边(v_s', v_l)为半径，按顺时针方向旋转。

在上面的方法中，0 型和 1 型的处理都非常简单。2 型和 3 型中计算切割平面时也只用到简单的内积和外积运算，计算也很简单。4 型和 5 型的计算复杂度比前面几种类型要高，但这两种情形出现的概率比较低，因此对整体的计算复杂度影响不大。

4.2 简化浏览

简化浏览时，顺序执行边坍缩操作，从网格中不断删除结点和三角形，就可以使网格的分辨率降低，最后将得到最粗糙的网格 M^0。除了不需要记录点分裂信息外，与网格简化算法基本相同。

4.3 选择性细化浏览

虽然根据点分裂记录序列的顺序依次插入或删除结点可以达到提高或降低模型分辨率的目的，但是在一些情况下并不需要使整个模型都达到最高的分辨率。例如模型的数据量非常大，而浏览者关心的只是其中某个局部时，就只需使模型的局部达到最高分辨率，而其余的部分保持较低的分辨率即可。这时就需要对初始网格进行选择性的细化。

实现选择性细化的方法是：对于点分裂信息$\{v_s, v_l, v_r, delta_vs, split_type\}$，只有满足下列条件时才对 v_s 进行分裂，否则跳过这一条记录：

1. v_s、v_l、v_r 都存在于网格中；

2. v_s 处于需要细化的区域内。

选择性细化对于基于视点的模型浏览是十分有用的，只要对上述第 2 个条件进行一些修改，就可以实现以注视点为中心的分辨率的连续变化。

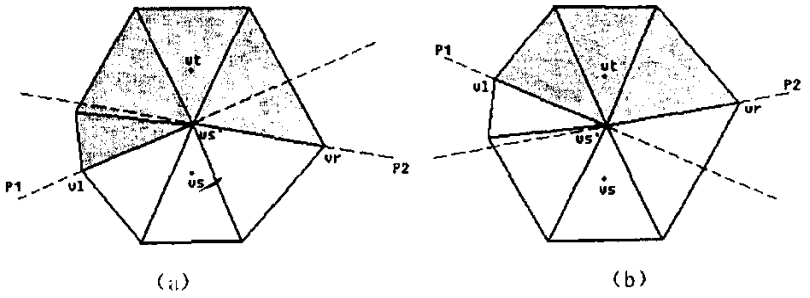

图 5　2 型点分裂过程中平面切割的两种情形

5. 实验结果

运用上文介绍的方法，我们对几组三维模型进行了网格简化，并建立了多分辨率模型。表 1 列举了这几组模型的实验数据，其中除了三叶虫化石模型外，都是非封闭的。对于颅骨和三叶虫两组数据，我们采用了式(3)的策略进行简化，而其余模型采用的是式(4)的策略。从实验结果中可以看出，使用前者

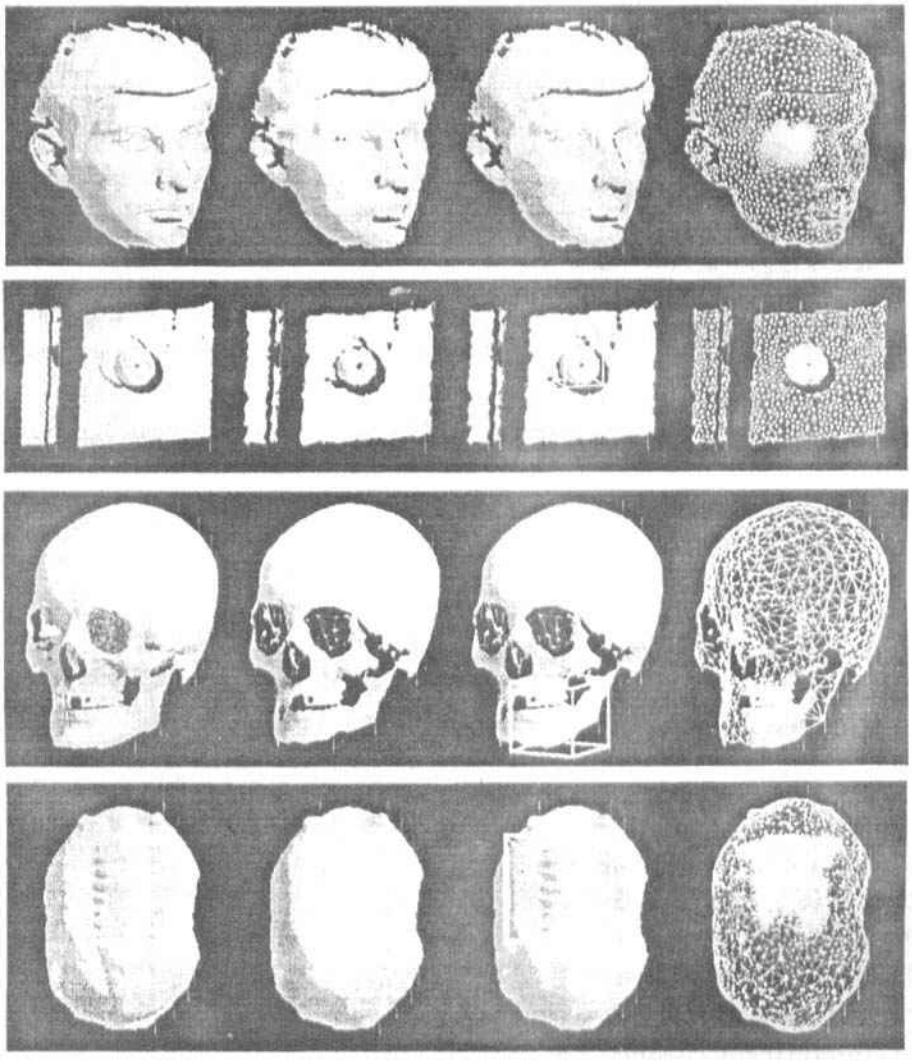

图 6　网络简化及选择性细化的实验结果。图中模型自上向下为罗马青年、门锁手柄、颅骨和三叶虫化石；自左向右顺序为：原始模型、简化后的模型、选择细化的模型及其三角网格表示(白色线框表示要求细化的区域)。

时能够较好地保存模型的几何特征点，但是在边缘部分表现不是很好；而使用后者时则较好地保持了模型的边缘，但计算时间相对较长。各模型的简化和选择性细化结果可以在图 6 和图 7 中看到。以上的执行时间是在 Pentium4 1.7GHz、256Mb 内存的 PC 机上取得的。

表 1　几组模型的实验数据

模型名称	原始网格		简化后的网格		源文件大小	PM 文件	简化时间
	结点数	三角形数	结点数	三角形数	(obj)	大小	(分钟)
罗马青年	14579	28053	1579	2602	1,933Kb	287kb	9
门锁手柄	15913	30714	913	1443	2,152Kb	310kb	12
颅骨	26101	50000	1101	1568	3,521Kb	508kb	19
三叶虫化石	28410	56815	1410	2816	3,915Kb	557kb	29
场景	43280	83749	3280	5405	6,011Kb	849kb	90

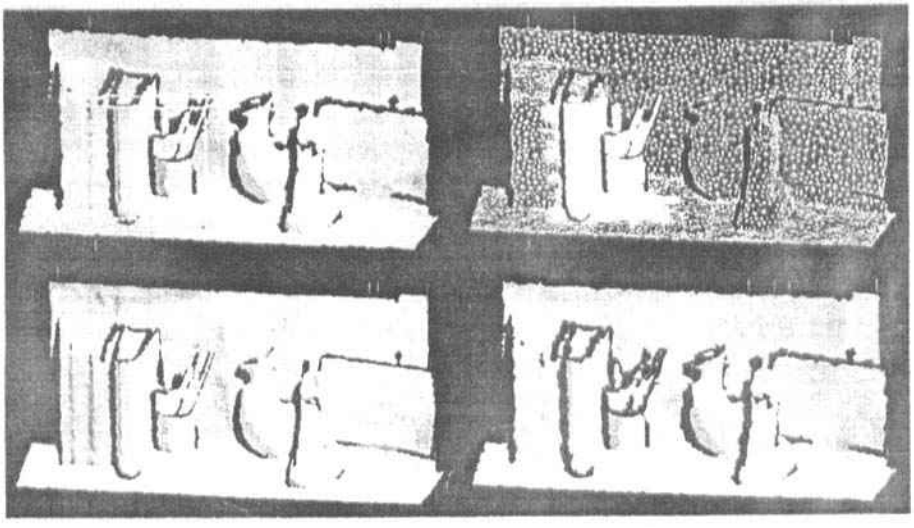

图 7　场景模型的网格简化和选择性细化实验结果。左上为原始网格，右上为简化网格，左下为选择细化的模型，右下为对应的三角网格表示(白色线框表示要求细化的区域)。

结论　本文详细介绍了一种针对非封闭三维模型的分辨率控制方法。在已有工作的基础上，以下的几个方面还值得进一步的研究：

(1) 由于我们主要针对的是存在边缘和漏洞的模型，因此必须进一步考虑在网格简化过程中如何更好地保存边缘部分的信息。在目前的实现中，进行边坍缩操作时我们把合并的新结点取为原来两个结点的平均值，即被删除的边的中点。因此在大规模简化后，可以看到边缘部分有收缩现象，这就有可能失去很多有用的特征。要解决这个问题，可以根据被删除边周围的拓扑性质，将新结点取为原来两个结点之一，这样就能更好地保持边界部分的形状。另外，还可以在计算三角形权值的时候加上一项边缘权值，从而推迟边界三角形的删除。

(2)另一个需要考虑的是计算的实时性问题。要将分辨率控制技术应用于机器人视觉等领域，就必须考虑如何提高计算效率，缩短计算时间，以便实时地进行网格简化和分辨率控制。

致谢　本文研究中所使用的三叶虫化石由北京大学地质学系地质档案博物馆提供，特此感谢。

参 考 文 献

1 De Floriani L, Magillo P, Puppo E, Bertolotto M. Variable resolution operators on a multiresolution terrain model. In: Proc. 4th ACM Workshop on Advances in Geographic Information Systems, 1996, 121～128

2 De Floriani L, Magillo P, Puppo E. Building and traversing a surface at variable resolution. In: Proc. IEEE Visualization, 1997. 103～110

3 Hoppe H. Progressive meshes. ACM SIGGRAPH Proc., 1996. 99～108

4 Hoppe H. Efficient implementation of progressive meshes. Computers & Graphics, 1998, 22(1): 27～36

5 Cohen J D, Aliaga D G, Zhang W. Hybrid simplification: combining multi-resolution polygon and point rendering. In: Proc. IEEE Visualization, 2001. 37～44

6 To D, Lau R, Green M. An adaptive multi-resolution method for progressive model transmission. Teleoperators and Virtual Environments, 2001, 10(1): 62～74

7 Guskov I, Khodakovsky A, Schroder P, Sweldens W. Hybrid Meshes: Multiresolution using regular and irregular refinement. To appear in Proc. Symposium on Computational Geometry, 2002

8 Hoppe H, et al. Mesh optimization. ACM SIGGRAPH Proc., 1993: 19～26