基于面坍陷的四面体网格简化与重建

文俊[1]，赵飞[2]

(1 重庆市电力公司沙坪坝供电局，重庆市，400030；2 厦门大学自动化系，福建厦门 361005)

摘要：如何简化和重建大规模的四面体网格数据来达到实时渲染的目的，已经引起越来越多研究人员的重视。本文提出了一种新颖的使用面塌陷的方法来实现四面体网格的简化和重建。我们使用最大三角形拉伸率作为最主要的判别标准，来判断选择的三角面是否适合进行面塌陷操作。在我们的实验中，使用我们的方法在每次进行面塌陷操作时，平均可以删除 11 个四面体，并且我们的算法在简化和重建操作中都有很高的效率。我们的方法也很好的保持四面体网格的边界。

关键词：四面体网格，简化，重建

1 引言

随着硬件技术和现代计算方法的发展，大规模和高精度的体数据越来越受到研究人员的关注。近年来，体数据在诸如科学可视化、医学图像处理、有限元分析等方面有了广泛的应用。但是，如何选择有效的方法，进行体数据有效的存储、近似和渲染等，是一个相对困难的问题。和其他体数据的表示方式比较起来，四面体网格数据结构具有更好的灵活性和更少的限制，在体数据表示、传输、渲染等方面有很多优点。

由于理论和应用的重要性，二维流型的三角网格在计算机图形学领域已经得到了广泛的研究。三角网格的分割、参数化、映射等都基本趋于成熟，更多的细节可以参考[1]。在这些研究的基础上，四面体网格的体参数化[2]、体映射[3]、纹理合成[4]也在近年来得到了广泛的发展。

本文提出了一种基于面塌陷的方法实现四面体网格的简化与重建，实验发现，使用我们的方法进行网格简化时，平均一次面坍陷操作可以删除 11 个四面体。并且我们的算法在简化和重建操作中都有很高的效率。

本文的组织结构为：第二部分回顾了四面体简化和重建的研究现状，第三部分阐述了我们使用的面坍陷的方法，第四部分为实验结果，第五部分为结论及展望。

2 相关工作

为了实现四面体网格的简化和重建，最为重要的操作是实现四面体网格的简化，因而本部分主要关注于四面体网格简化的操作。四面体网格的简化策略，主要可以分为：基于边塌陷的四面体网格简化、基于面塌陷的四面体网格简化、基于体塌陷的四面体网格简化。

2.1 基于边坍陷的四面体网格简化

Hoppe et al. [5]在三角网格上提出了基于边坍陷和顶点分裂的策略，Staadt and Gross [6]将渐进式网格[5]扩展到四面体网格上，提出了渐进式四面体（PTM）的概念。该方法也使用边坍陷和顶点分裂的操作，该文提出了“代价函数”来确定边坍陷的序列，并使用了静态测试和动态测试来保证算法的正确性。但使用该方法将 576,576 个四面体简化到 117,139 个四面体要花费将近 5 个小时的时间。

Y . Chiang and X. Lu [7]在使用边坍陷进行四面体网格简化的过程中，尽量的保证所有等值面的拓扑。该方法首先将四面体网格分割成拓扑等价的区域，然后对这些区域进行简化。该方法将 149,026 个简化到 11.3%，需要花费 44 分钟。

2.2 基于体坍陷的四面体网格简化

Chopra and Meyer[8]提出了基于体坍陷的四面体简化方法并命名为 TetFusion。该方法以最大基向量的拉伸率（RMAX）和删除后造成的标量属性误差来确定待删除四面体的优先级。使用该方法可以获得较高的简化效率，可以在两分钟内将 827,904 个四面体简化为原模型的 98%。但是

该方法不对四面体网格的表面进行处理。

2.3 基于面坍陷的四面体网格简化

Xuanming Wang et al.[9]提出了基于面坍陷的方法来进行四面体网格的简化。该方法以三角形面积的变化率来确定待删除四面体的优先序列，并且将 11,198 个四面体简化到 3,602 四面体花费了 41.35 秒。但该方法仅仅使用三角形面积的变化率来确定优先序列，没有考虑在简化过程中各种可能造成网格退化的情况。

3 基于面塌陷的渐进式四面体

3.1 综述

我们将 Hoppe et al. [5]和 Staadt and Gross [6]中提到的方法扩展为渐进式四面体中的面坍陷和顶点分裂操作。

我们可以通过一系列的面坍陷操作，将四面体网格 T = T^n 简化到基本网格 T^0：

$$T^n \xrightarrow{fcol_{n-1}} T^{n-1} \xrightarrow{fcol_{n-2}} \cdots \xrightarrow{fcol_0} T^0$$

在每次进行面坍陷 $fcol_i$ 时，该面的三个顶点 V_a V_b 和 V_c 将会被一个新建的顶点 V 所替代。

我们也可以通过 n 次的顶点分裂 vsplit 来将基本网格 T^0 重建为初始四面体网格 T^n。

$$T^0 \xrightarrow{vsplit_0} T^1 \xrightarrow{vsplit_1} \cdots \xrightarrow{vsplit_{n-1}} T^n$$

为了表示初始网格 T = T^n，我们使用如下的定义来表示渐进式四面体（progressive tetrahedral mesh PTM）：

$$(T^0, vsplit_0, vsplit_1, \ldots, vsplit_{n-2}, vsplit_{n-1})$$

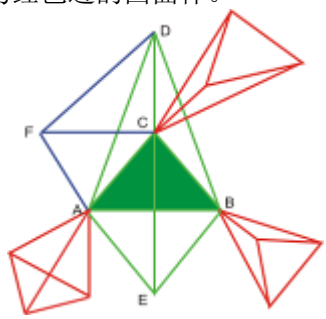
3.2 面坍陷

在阐述面坍陷操作前，我们先给出如下的定义。我们将关联于待删除面的四面体分为三类：面相关四面体（TFace）、边相关四面体（TEdge）、点相关四面体（TVertex）。

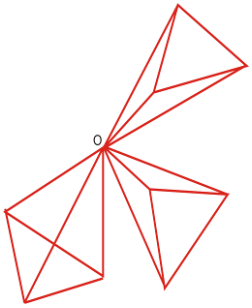
面相关四面体（TFace）：待删除面所在的四面体，如图一（a）所示的四面体 ABCD 和 ABCE。

边相关四面体（TEdge）：和待删除四面体共享并且仅共享一条边的四面体，如图一（a）所示的四面体 ACDF。

点相关四面体（TVertex）：和待删除四面体共享并且仅共享一个顶点的四面体，如图一（a）所示的标示为红色边的四面体。



（a）



（b）

图一 面坍陷操作（三角形ABC为待坍陷的面）

在面塌陷操作中，所有的面相关的四面体 TFace（图一中绿色边的四面体）和边相关四面体 TEdge（图一中蓝色边的四面体）都将被删除，所有的顶点相关的四面体 TVertex（图一中红色边的四面体）都将被拉伸到待删除面重心点的位置（如图一(b)所示）。

塌陷面的判别：我们使用最大三角面拉伸率来衡量该面是否适合进行坍陷操作。我们先做如下的定义：

三角形形状因子 ρ：

$$\rho = \frac{4(q-a)(q-b)(q-c)}{abc}$$

其中 a,b,c 为三角形三条边的长度，q 为三角形的半周长。

三角面拉伸率 θ：$\theta = (|\rho 1 - \rho 2|)$，其中 ρ_1 和 $\rho 2$ 分别为该三角形在面坍陷操作前和面塌陷操作后，该三角形的形状因子。

最大三角面拉伸率 θ_M：所有的顶点相关的四面体 TVertex 中相关三角面中最大的三角面拉伸率 θ。

任给一个三角面，如果其最大三角面拉伸率小于预先设置的阈值，那么该面通过可以进行相应的面坍陷，否则不能对该面进行面塌陷操作。

特征错误测试：为了避免在面塌陷操作中出现诸如翻转、自相交等退化的情况，我们需要在面塌陷过程中进行特征误差测试。

测试 I：几何错误测试。如前所述，任给一个三角面，只有 $\theta(f) < \theta_{MAX}$，该面才通过几何错误测试；

测试 II：翻转错误测试。为避免四面体发生翻转，需要进行此测试，可参考文献[8]。

3.3 贪心简化算法

为了提升算法的效率，我们采用贪心算法来进行简化操作。

（1）输入简化精度和最大三角面拉伸率 θ_{MAX}；

（2）标示所有面为未访问面，并标示第一个面为 F_{begin}；

（3）对于从 F_{begim} 开始的所有的面，如果没有通过测试 I 和测试 II，标示该面已经访问过，标示该面的下一个面为 F_{begim}，继续执行（3）；否则跳入（4）继续执行；

（4）进行如下操作：

（a）对通过测试的面进行面塌陷操作，并将该面从面列表中删除；

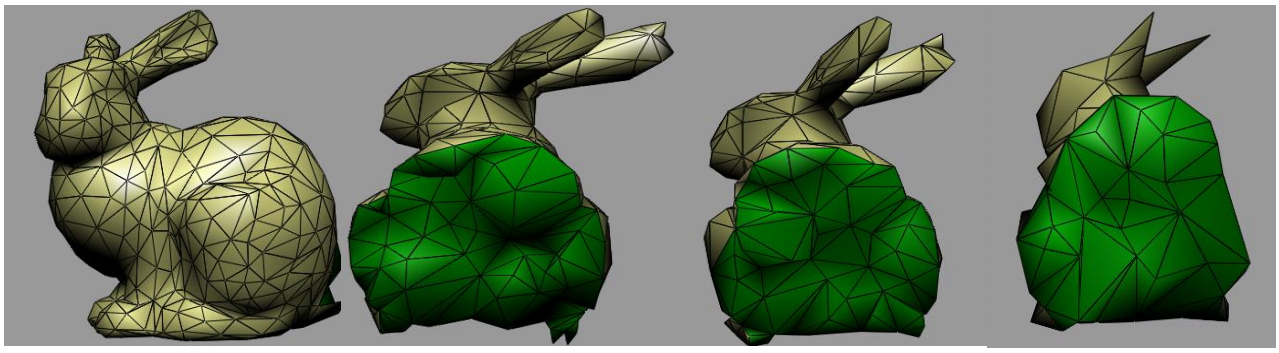
（b）标示该面的下一个面为 F_{begim}，并跳转（3）继续执行；

（5）如果达到预设简化精度，则结束；否则转（2）继续执行。

4 实验结果

本部分的实验数据在 Intel Pentium 4 2.8G 内存为 2G 的环境下测试得到。

图二为使用我们的基于面坍陷的算法，得到的简化后的分层次的结果。为了显示四面体网格内部的简化情况，我们首先给出测试模型，简化的结果以该模型的同一个切面的剖面图的形式给出。图表一给出了使用我们的算法对不同的模型进行测试的实验数据。



(a) 原始四面体模型 (b) 简化80% 后的结果 (c) 简化50% 后的结果(d) 简化20% 后的结果图三：Bunny（3,123个四面体）模型的层次表示

图表一 对三个不同的模型进行测试的实验数据

模型	拓扑	原始模型 T^n	基模型 T^0	简化率	θ_{MAX}	简化时间 (Sec.)	重建时间 (Sec.)
Bunny	Genus=0	3,123	470	15.1%	0.03	1.53	0.81
Torus	Genus=1	66,294	16,673	25.3%	0.04	599.34	312.72
Greek	Genus=4	238,479	103,026	43.2%	0.055	2448.16	1260.47

5 结束语

在本文中，我们提出了一种新颖的基于面坍陷实现渐进式四面体的方法。我们以最大三角形拉伸率作为最主要的判别标准，来判断选择的三角面是否适合进行面塌陷操作。通过实验发现，使用我们的方法进行网格简化时，平均每一次面坍陷操作可以删除 11 个四面体。该方法具有较高的简化和重建效率，较好的鲁棒性，并且能处理任意拓扑的四面体网格模型。

渐进式四面体可以构造分层次的细节模型，该方法有着广泛的应用。在未来的工作中，我们准备使用该方法进行四面体网格的分割、四面体网格的体映射等方面。

参考文献

[1] K. Hormann, B. L ́evy, and A. Sheffer. Mesh parameterization: Theory and practice. In SIGGRAPH 2007 Course Notes, volume 2, pages 1–122, San Diego, CA, August 2007. ACM Press.

[2] Martin T, Cohen E, Kirby RM. Volumetric parameterization and trivariate Bspline fitting using harmonic functions. Computer Aided Geometric Design 2009;26(6):648_64.

[3] Xin Li , Xiaohu Guo , Hongyu Wang , Ying He , Xianfeng Gu, Hong Qin, Harmonic volumetric mapping for solid modeling applications, Proceedings of the 2007 ACM symposium on Solid and physical modeling, June 04-06, 2007, Beijing, China.

[4] Cignoni, P.,Montani, C., Rocchini, C., and Scopigno, R. 1998a. A general method for preserving attribute values on simplifiedme mesh. In IEEE Visualization 98 Conference Proceedings. 59–66, 518.

[5] Hoppe, Hugues. Progressive Mesh. In Proceedings of SIGGRAPH’96 (New Orleans, Louisiana, August 1996),pages 99-108. ACM SIGGRAPH, ACM Press. August 1996.

[6] O. G. Staadt and M. H. Gross, “Progressive tetetrahedralizations,” in IEEE Visualization ’98, 1998, pp. 397–402.

[7] Y.-J. Chiang and X. Lu. Progressive Simplification of Tetetrahedral Mesh Preserving All Isosurface Topologies. Computer Graphics Forum, 22(3):493–504, 2003.

[8] Chopra, P. and Meyer, J. 2002. TetFusion: An algorithm for rapid tetetrahedral simplification. In IEEE Visualization 2003.133–140.

[9] Xuanming Wang,Guying Wu, and Enhua Wu. “A Novel Technique Based on Triangle Decimation for Tetetrahedral Simplification”, Acth Electronica Sinica, Vol. 35 No. 12,2007,pp.2343-2346.