高阶平滑表面提取算法的CUDA并行实现

袁红星[1] 吴少群[1] 郭 立[2] 朱仁祥[1]
(宁波工程学院电子与信息工程学院 宁波 315016)[1]
(中国科学技术大学电子科学与技术系 合肥 230026)[2]

摘 要 高阶平滑表面提取算法可有效抑制传统步进立方体算法存在的鳞状失真现象,但引入了较复杂的最优化运算,降低了表面提取的效率。针对该问题,提出基于图形处理器的并行加速优化方法。首先将算法分解成分界区域、窄带区域、嵌入函数边界值、嵌入函数值最优化和三角面网格提取5个计算步骤,其次根据每个处理步骤的运算特点进行任务分解。为便于图形处理器并行优化,将其中最耗时的嵌入函数值最优化计算,表示成矩阵运算形式,通过投影雅可比迭代估计最优解。实验结果表明,在GeForce GT 240M显卡上并行优化后平均加速比可达到9以上。

关键词 步进立方体,表面提取,图形处理器,高阶平滑
中图法分类号 TP391.4 **文献标识码** A

Higher-order Smooth Surface Extraction CUDA Parallel Implementation

YUAN Hong-xing[1] WU Shao-qun[1] GUO Li[2] ZHU Ren-xiang[1]
(School of Electron and Information Engineering, Ningbo University of Technology, Ningbo 315016, China)[1]
(Department of Electronic Science and Technology, University of Science and Technology of China, Hefei 230026, China)[2]

Abstract Higher-order smooth surface extraction method can overcome aliasing artifacts of marching cubes algorithm. However, it introduces extra computation burden especially with optimal embedding function calculation. To resolve the problem, a parallel implementation based on graphics processing unit was presented. The original higher-order smooth surface extraction algorithm was divided into five parts including margin region, narrow band region, embedding function margin values, optimal embedding function and triangular mesh extraction. Then they were paralleled with task assignments method. The most complexity function embedding calculation is approximated by projected Jacobin method. The experimental results show that the speedup achieves more than 9 after parallelization on GeForce GT 240M GPU.

Keywords Marching cubes, Surface extraction, Graphics processing unit, Higher-order smooth

1 引言

步进立方体算法(Marching Cubes, MC)是用于面绘制的经典等值面提取算法(Iso-surface Extraction),在医学图像三维重建、三维可视化、散乱点云曲面重建等领域具有广泛的应用。MC算法使用三角面网格逼近三维数据场中的等值面。当前显卡对三角面网格绘制都有硬件加速支持,且MC算法原理简单、易于实现,因此它得到广泛的应用。

现有针对MC算法的研究主要关注以下3个问题:一是等值面连接的二义性问题,若值为1和值为−1的顶点位于对角线的两端,则存在两种可能的连接方式[1];二是逼近精度问题,MC算法假设曲面沿体单元边界呈线性变化,采用线性插值计算等值面与体单元边的交点,并用这些交点组成逼近等值面的多边形,当要求逼近精度较高时,很难达到要求;三是存储空间问题,MC算法需要用大量三角面片来逼近等值面,增加了存储空间和传输带宽,加大了内存消耗。大部分算法仅针对其中某个问题进行改进和优化。如梁秀霞等人通过求解三线性插值函数临界点来解决二义性问题[2]。Lopes等人利用肩点来提高MC算法逼近等值面的精度[3]。薛强等人通过多分辨率分解的方式来减少MC算法生成的三角面个数[4]。MC算法的前两个问题会导致提取的曲面存在规则化的失真,称为鳞状失真。由于人眼对规则性的形状较敏感,因而严重损伤了表面提取的质量[5]。虽然可通过高阶插值抑制该问题,但仍无法完全消除鳞状失真[6]。MC算法3个问题的本质原因是采样混叠。但很少有算法从采样混叠角度来探讨这些问题。针对这一现象,Victor为降低采样混叠,提出高阶平滑表面提取算法[7],即通过添加高阶平滑性约束,体素顶点值可能会取−1和1之间的某个值,从而减少了拓扑二义性问题,同时可提高逼近精度,有效抑制了鳞状失真现象。但该方法增加了计算量,影响了表面提取的速度。为此,本文的

到稿日期:2012-08-05 返修日期:2012-11-09 本文受宁波市自然科学基金(2012A610043),浙江省自然科学基金(LY12F01001),国家自然科学基金(61071173)资助。

袁红星(1980−),男,博士,高工,CCF会员,主要研究方向为信号与信息处理、3D视频信号处理、高性能计算,E-mail:yuanhx@mail.ustc.edu.cn;吴少群(1981−),女,硕士,讲师,主要研究方向为信号与信息处理;郭 立(1946−),男,教授,博士生导师,主要研究方向为音、视频信号处理;朱仁祥(1971−),男,博士,讲师,主要研究方向为通信信号处理。

前期工作通过多核并行实现来提高其计算效率[8,9]。

近年来图形处理器(Graphics Processing Unit,GPU)大规模并行计算的推广与普及为表面提取的实时计算提供了硬件平台。英伟达(NVIDIA)公司提出的统一计算设备架构(Compute Unified Device Architecture,CUDA)[10]编程模型显著降低了基于GPU的并行化难度和工作量。CUDA编程模型对C语言进行了扩展,将CPU和GPU分别视为host和device,两者协同工作。CPU承担逻辑性强的事务和串行计算任务,而GPU负责数据密集型的并行计算任务。为进一步提高高阶平滑表面提取的计算效率,本文在前期工作基础上通过GPU对其进行并行优化。

2 高阶平滑表面提取算法分析

高阶平滑表面提取算法的思想是在三维数据场上嵌入一个高阶平滑函数,$f:G\to R$。其中G表示三维数据场。假设三维数据场中体元顶点在x,y,z方向上坐标分别用i,j,k表示。不失一般性,假设物体表面包围区域之内的顶点值为-1,包围区域之外的为1。则在式(2)的约束条件下通过求解式(1)可得到嵌入到体元顶点上的最优函数值f_{ijk}。

$$\sum_{ijk}[(f_{i+1jk}+f_{i-1jk}-2f_{ijk})^2+(f_{ij+1k}+f_{ij-1k}-2f_{ijk})^2+(f_{ijk+1}+f_{ijk-1}-2f_{ijk})^2]\to\min \tag{1}$$

服从约束:

$$v_{ijk}f_{ijk}\geqslant m_{ijk} \tag{2}$$

式中,v_{ijk}表示坐标值为(i,j,k)的顶点,其取值为1或-1,分别表示在物体表面包围区域之外和之内。m_{ijk}为一个非负值,是嵌入函数的边界约束值,其定义如式(3)所示。

$$m_{ijk}=\min_{(\alpha,\beta,\gamma)\in B}\sqrt{(i-\alpha)^2+(j-\beta)^2+(k-\gamma)^2} \tag{3}$$

式中,B表示顶点取值分界处的顶点集合(即顶点取值为-1和1之间的过渡区域)。高阶平滑表面提取算法通过求解式(1)在体元的顶点上嵌入一个高阶平滑函数f,然后利用经典的步进立方体算法从f中提取表面。为降低计算量,文献[7]仅在一个窄带区域对体元顶点进行函数值的嵌入计算。窄带区域NB的定义方法如式(4)所示,其中C为一个小的正常数,用于控制窄带区域的大小。

$$NB=\{v_{ijk}\mid dist(v_{ijk},B)\leqslant C\} \tag{4}$$

高阶平滑表面提取算法主要包括计算边界区域B;根据式(4)确定参与计算的窄带区域;计算约束条件的下限值m_{ijk};求解式(1);利用步进立法体算法提取三角面网格。其算法流程如图1所示。

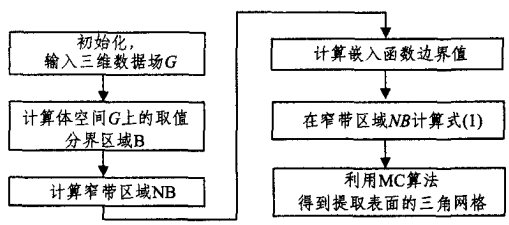

图1 高阶平滑表面提取算法流程图

3 高阶平滑表面提取的CUDA并行实现

高阶平滑表面算法的基本处理单位是三维数据场的体元顶点,具有数据级并行性,适合于用GPU进行并行加速。

3.1 计算分界区域B

如果某个体元的顶点取值和相邻顶点取值不一样,就将该顶点添加到分解区域B中。每个体元顶点的判断是完全并行的,因此用GPU并行扫描体元顶点,每个线程扫描一个顶点。

3.2 计算窄带区域NB

如式(4)所示,窄带区域NB由顶点到集合B的欧式距离决定,各顶点的计算是不相关的,因而也采用GPU并行扫描的方式进行加速处理。

3.3 计算嵌入函数边界值

嵌入函数边界值的计算是在窄带区域NB上每个体元的顶点上进行的,如式(3)所示,各顶点间的计算也是完全并行的,因此采用GPU并行扫描的方式进行优化。

3.4 嵌入函数值的最优化计算

最优化的嵌入函数值是在满足式(2)约束条件下使式(1)最小的解,这是一个典型的二次规划问题。本文的前期工作表明,这部分的计算是最耗时的。为便于并行优化,将式(1)用矩阵表示如下:

$$\min E(F)=F^{\mathrm{T}}QF \tag{5}$$

式中,$F\in R^{N\times 1}$为顶点函数值f_{ijk}构成的列向量,$f_{ijk}=F_{((k\times N_z+j)\times N_y)+i}$($N$为三维数据场上总的体元顶点数目,$N_y$和$N_z$分别表示$y,z$方向上体元顶点个数);$Q\in R^{N\times N}$表示式(1)中的有限差分关系。假设矩阵$H\in R^{N\times N}$,初始化时将其设置为$H=0^{N\times N}$。之后,将窄带区域$NB$上所有顶点$v_{ijk}$及其邻域顶点$v_{i-1jk}$、$v_{i+1jk}$、$v_{ij-1k}$、$v_{ij+1k}$、$v_{ijk-1}$和$v_{ijk+1}$在矩阵$H$上对应位置的元素值按式(6)—式(12)进行设定。

$$H(((k\times N_z+j)\times N_y)+i,((k\times N_z+j)\times N_y)+i)=-6 \tag{6}$$

$$H(((k\times N_z+j)\times N_y)+i,((k\times N_z+j)\times N_y)+i-1)=1 \tag{7}$$

$$H(((k\times N_z+j)\times N_y)+i,((k\times N_z+j)\times N_y)+i+1)=1 \tag{8}$$

$$H(((k\times N_z+j)\times N_y)+i,((k\times N_z+j-1)\times N_y)+i)=1 \tag{9}$$

$$H(((k\times N_z+j)\times N_y)+i,((k\times N_z+j+1)\times N_y)+i)=1 \tag{10}$$

$$H(((k\times N_z+j)\times N_y)+i,(((k-1)\times N_z+j)\times N_y)+i)=1 \tag{11}$$

$$H(((k\times N_z+j)\times N_y)+i,(((k+1)\times N_z+j)\times N_y)+i)=1 \tag{12}$$

则,矩阵Q的计算方法如式(13)所示:

$$Q=H^{\mathrm{T}}H \tag{13}$$

本文采用投影雅可比迭代法在式(2)的约束下求解式(5)。为求解式(5),将其对F求导得:$QF=0$。进而根据文献[11]的公式(18),可用式(14)所示的雅可比迭代得到F的解。为满足式(2)所示的约束条件,用式(15)对F的解进行范围限定。因此,通过式(14)、式(15)所示的迭代过程可求得定义在物体表面的高阶平滑函数F。图2给出了$E(F)$随迭代次数$iter$变化的曲线。由图可知,随着迭代次数的增加,$E(F)$呈递减趋势,迭代次数到达600左右时,$E(F)$已趋于平稳。因此,实验中迭代次数均设为600。

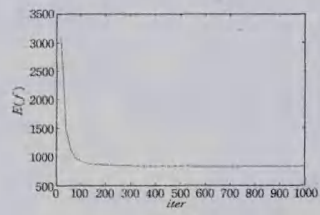

图 2　$E(f)$随迭代次数变化曲线

$$F_{iter}=-\omega\times diag(Q)^{-1}(Q-diag(Q))+(1-\omega)\times F_{iter-1} \tag{14}$$

$$F_{iter}^{ijk}=F_{iter}((k\times N_z+j)\times N_y+i)=\begin{cases}\max(F_{iter}^{ijk},m_{ijk}), & v_{ijk}=1\\ \min(F_{iter}^{ijk},-m_{ijk}), & v_{ijk}=-1\end{cases} \tag{15}$$

式(13)只需计算一次，式(14)、式(15)需进行迭代计算直至相邻两次迭代计算结果非常接近。每次迭代生成的顶点值，仅跟上一次迭代的顶点值和邻域顶点值相关，顶点值之间不存在依赖关系，因此各顶点值的生成是独立的。本文采用GPU并行扫描的方式对式(14)、式(15)进行优化，式(13)和$diag(Q)^{-1}$、$Q-diag(Q)$在迭代前由CPU预先算好传送给GPU。

3.5　三角面网格提取

三角面网格提取实际上就是在三维数据场嵌入的高阶平滑函数f上提取0等值面，这里采用了经典的步进立方体算法。其并行优化思路借鉴了CUDA提供的例程“Marching Cubes Isosurfaces”[12]。

4　实验与分析

4.1　实验环境

用CUDA C语言实现算法，算法在Visual Studio 2010中通过编译、调试。测试机器配置：操作系统是Windows Server 2008 R2；处理器是Intel酷睿双核E8400，主频3.00GHz，内存4.00GB；显卡为GeForce GT 240M 1.21 GHz GPU，16kB共享内存，436MB全局内存。

4.2　加速比测试

图3示出Bunny模型[13]使用本文方法和多核并行优化方法[4,5]在不同体空间分辨率情况下的加速比比较。从图中可以看出，CUDA并行优化的平均加速比达到9.17，而多核并行优化的平均加速比为1.87。通过GPU加速，显著提升了高阶平滑表面提取算法的计算性能。图中CUDA的加速比先是随着体元数目的增加而增大，随后又有所下降。这是因为，当GPU处理单元个数和并行计算任务数目相匹配时性能最佳，当GPU处理单元有空闲或不足以分配所有任务时，并行性能会有所下降。

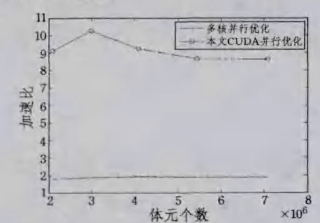

图3　Bunny测试数据并行加速比

4.3　提取表面绘制效果

图4—图7显示了Bunny模型[13]、Furukawa提供的dinosaur[14]和Seitz等人提供的dinoSparseRing、templeSparseRing[15]测试数据提取表面的绘制效果。实验中，体元个数均设为128^3，式(14)中ω均置为0.5。对原始高阶平滑算法(下称Sebvhos算法)、本文CUDA优化的Sebvhos算法和经典的MC算法提取表面的绘制效果进行了对比。从图中可以看出，MC算法提取的表面存在明显的鳞状失真，高阶平滑算法可得到细节保持的平滑曲面，有效抑制了鳞状失真现象。本文CUDA并行优化后的算法和原始算法的提取表面在视觉上完全一样。这表明本文可在不损伤提取表面质量的情况下得到显著的加速性能提升。

(a) MC算法　(b) 原始Sebvhos算法　(c) CUDA并行优化(加速比:9.11)

图4　Bunny数据提取的表面绘制效果

(a) MC算法　(b) 原始Sebvhos算法　(c) CUDA并行优化(加速比:9.14)

图5　Dinosaur数据提取的表面绘制效果

(a) MC算法　(b) 原始Sebvhos算法　(c) CUDA并行优化(加速比:9.12)

图6　dinoSparseRing数据提取的表面绘制效果

(a) MC算法　(b) 原始Sebvhos算法　(c) CUDA并行优化(加速比:9.12)

图7　tempSparseRing数据提取的表面绘制效果

结束语　提出了高阶平滑表面提取算法在GPU上的并行优化方法，通过对算法计算步骤的分解和多线程任务分配，提高了表面提取的计算效率。英伟达在GPU上并行优化的实验表明，平均加速比可达到9以上。为便于并行实现，将其中最为耗时的最优函数嵌入问题表示成矩阵计算形式，并利用投影雅可比迭代估计最优解。

参 考 文 献

[1]　僧德文，李仲学，李翠平，等. Marching Cubes算法改进研究及应用[J]. 计算机应用研究，2006，7：50-51

(下转第70页)

群调整次数，这说明 CAECS 提高了分群稳定性；(2)当权值取值适当时，CAECS 降低了分群维护开销。以上两点说明：当权值取值适当时，CAECS 要优于 MSWCA。

结束语 在自组网中，考虑运动相关性的分群算法有利于提高分群稳定性，而 MSWCA 是该类分群算法中在分群稳定性度量方面考虑得最全面的典型算法。本文针对 MSWCA 存在的“只考虑群内稳定性，而忽视群间稳定性”的问题，提出一种增强分群稳定性的分群算法(CAECS)，它以提高分群稳定性为目标，这也有利于降低分群维护开销。该算法基于移动预测思想，综合考虑群内稳定性、群间稳定性和分群优化，通过调节权值使算法适用于不同的场景。本文首先描述了 CAECS 的基本思想，然后描述了基于上述基本思想的分群算法，最后采用 NS2 仿真软件对 CAECS 和 MSWCA 的算法性能进行了仿真比较分析。仿真结果表明：当权值取值适当时，CAECS 在分群稳定性和分群维护开销等性能指标上都要优于 MSWCA。

参 考 文 献

[1] 陈嘉宁. 基于备份的移动自组织网络分簇策略研究[D]. 长沙：湖南大学，2007

[2] Basu P, Khan N, Little T D C. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks[C]// 2001 International Conference on Distributed Computing Systems Workshop, 2001. 2001:413-418

[3] Dhurandher S K, Singh G V. Stable Clustering with Efficient Routing in Wireless Ad Hoc Networks[C]// 2nd International Conference on Communication Systems Software and Middleware, 2007. 2007:1-12

[4] Tolba F D, Magoni D, Lorenz P. Connectivity, Energy and Mobility Driven Clustering Algorithm for Mobile Ad Hoc Networks [C] // IEEE Global Telecommunications Conference, 2007. 2007:2786-2790

[5] Torkestani J A, Meybodi M R. A Mobility-based Cluster Formation Algorithm for Wireless Mobile Ad-Hoc Networks[J]. Computer Science, 2011, 14(4):311-324

[6] Zhong Z D, Zhao D M. MPBC: A Mobility Prediction-based Clustering Scheme for Ad Hoc Networks[J]. IEEE Transactions on Vehicular Technology, 2011, 60(9):4549-4559

[7] Hussein A R, Yousef S, Al-Khayatt S, et al. An Efficient Weighted Distributed Clustering Algorithm for Mobile Ad Hoc Networks[C]// 2010 International Conference on Computer Engineering and Systems (ICCES), 2010. 2010:221-228

[8] Choi W, Woo M. A Distributed Weighted Clustering Algorithm for Mobile Ad Hoc Networks[C]// Advanced International Conference on Telecommunications-International Conference on Internet and Web Applications and Services, 2006. 2006:73

[9] Hwang Y C, Jeong Y S, Lee S H, et al. Advanced Efficiency and Stability Combined Weight based Distributed Clustering Algorithm in MANET[C]// Future Generation Communication and Networking. 2007:478-483

[10] Kawai Y, Sasase I. A Stable Clustering Scheme by Prediction of the Staying Time in a Cluster for Mobile Ad hoc Networks[C]// Proceedings of 14th Asia-Pacific Conference on Communications, 2008. 2008:1-5

[11] 黄卫红，李仁发，彭献武. 基于移动保持时间的无线自组网分簇算法[J]. 网络与通信，2007，23(3)：95-96，119

[12] Tao Y, Wang J, Wang Y L, et al. An Enhanced Maximum Stability Weighted Clustering Algorithm in Ad Hoc Network[C]// 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008. 2008:1-4

[13] Xu Y, Wang W Y. MEACA: Mobility and Energy Aware Clustering Algorithm for Constructing Stable MANETs[C]// Military Communications Conference, 2006. 2006:1-7

[14] 蒋毅，史浩山. 一种基于移动预测的自适应 Ad Hoc 网络分簇算法[J]. 计算机科学，2007，34(3)：28-29

(上接第 31 页)

[2] 梁秀霞，张彩明. 拓扑结构正确的三线性插值曲面的三角片逼近[J]. 计算机研究与发展，2006，43(3)：528- 535

[3] Lopes A, Brodlie K. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing[J]. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(1): 16-29

[4] 薛强，蔡文立，石教英. Marching Boxes：一个多精度等值面抽取算法[J]. 计算机辅助设计与图形学学报，1998，10(1)：7-14

[5] 朱经纬，蒙培生，王乘. 一种改进的 MC 算法[J]. 中国图象图形学报，2008，13(7)：1359-1366

[6] 吕理伟，顾耀林. 移动立法体算法的三重线性插值研究[J]. 计算机工程与应用，2005，32：41-44

[7] Lempitsky V, Boykov Y. Global optimization for shape fitting [C]// Proceedings of Computer Vision and Pattern Recognition, 2007. Minneapolis, USA: IEEE Press, 2007:1-8

[8] Li Wen, Guo Li, Yuan Hong-xing, et al. Parallel implementation and optimization of the Sebvhos algorithm[J]. Journal of Electronics, 2011, 28(3):277-283

[9] 李文，郭立，袁红星，等. 一种高阶平滑表面并行提取方法[J]. 中国科学院研究生院学报，2012，29(2)：251-256

[10] NVIDIA Corporation. CUDA programming guide 4.0 [EB/OL]. http://www.developer.nvidia.com/, 2011-06-27

[11] Cremeters D, Kolev K. Multiview stereo and silhouette consistency via convex functionals over convex domains [J]. IEEE Transactions on pattern analysis and machine intelligence, 2011, 33(6):1161-1174

[12] NVIDIA Corporation. CUDA SDK C Samples[EB/OL]. http://developer.nvidia.com/cuda-cc-sdk-code-samples, 2012-06-27

[13] Culess B, Levoy M. A volumetric method for building complex models from range images [C] // Proceedings of ACM SIGGRAPH, 1996. New Orleans, LA, USA: ACM Press, 1996:303-312

[14] Yasutaka F, Jean P. 3D Photography Dataset [EB/OL]. http://www.cs.washington.edu/homes/furukawa/research/mview/index.html, 2011-06-27

[15] Steve S, Brian C, James D, et al. The multi-view stereo evaluation[EB/OL]. 2011-06-27