第 37 卷 第 2 期 计 算 机 工 程 2011 年 1 月
Vol.37 No.2 Computer Engineering January 2011

• 图形图像处理 • 文章编号：1000—3428(2011)02—0224—03 文献标识码：A 中图分类号：TN911.73

基于 CUDA 的点云去噪算法

徐 波，唐 杰，武港山
(南京大学计算机科学与技术系，南京 210093)

摘 要：提出一种基于统一计算设备架构(CUDA)的双边滤波点云去噪算法，将点云去噪划分为多个并行度较高的步骤，利用 GPU 的并行计算能力，设计每个步骤的 CUDA 核函数。采用高斯加权的法矢计算方法，在双边去噪算法中加入面积权重缓解过光顺。实验结果表明，该算法能有效提高法矢计算的准确度，与 CPU 算法相比，计算速度提高了多个数量级。

关键词：统一计算设备架构；GPU 并行计算；点云去噪；双边滤波

CUDA-based Point Cloud Denoising Algorithm

XU Bo, TANG Jie, WU Gang-shan
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

【Abstract】This paper proposes a Compute Unified Device Architecture(CUDA)-based improved bilateral filtering point cloud denoising algorithm. The point cloud denoising algorithm is divided into several steps in a very high degree of parallelism. It separately designs CUDA kernel functions for each step, effectively emploies the GPU’s parallel computing power. It uses Gaussian-weighted method of calculating normal vector and effectively improves the accuracy of normal vector calculation. The bilateral denoising algorithm adds the weight of the area to alleviate the excessive smoothing. Experimental results show that the algorithm is stable and efficient, faster than the CPU calculation of multiple orders of magnitude.

【Key words】Compute Unified Device Architecture(CUDA); GPU parallel computing; point cloud denoising; bilateral filtering
DOI: 10.3969/j.issn.1000-3428.2011.02.078

1 概述

随着计算机技术的高速发展以及现代先进的制造技术和精密的测量技术的出现，点云数据模型在逆向工程中得到了广泛的应用。然而在获取点云数据的过程中，由于人为的扰动或仪器本身的缺陷等不确定的因素，往往使生成的点云数据含有噪声。在对含有噪声的点云数据进行后续处理之前，通常需要对其进行光顺去噪以提高后续处理的效果。大规模点云数据的去噪是一项非常耗时但并行性非常高的工作。目前可编程的 GPU 已发展成为一种高度并行化、多线程、多核的处理器，具有杰出的计算功率和极高的存储带宽且廉价。NVIDIA 公司推出的统一计算设备架构(Compute Unified Device Architecture, CUDA)编程模型和运行环境使 GPU 的并行功能更加公开化，更方便使用。

CUDA 是 NVIDIA 的 GPGPU 模型，可以使用 C 编写在显示芯片上执行的程序。在 CUDA 下，应用程序由两部分组成：一部分执行在 CPU 上，称为 Host 端；另一部分执行在显示芯片上，称为 Device 端又称为 Kernel[1]。通常在 Host 端准备数据，复制到显卡中，再由显卡执行 Kernel 程序，运行结果再复制回 Host 端。数据复制速度较慢，尽量避免频繁的数据传输。

CUDA 处理问题和 CPU 是不同的，例如 CPU 的内存存取延迟主要是通过 Cache 来消除，而 CUDA 主要通过高度并行化来隐藏。最适合 CUDA 处理的问题是可以大量并行化的问题，高度的并行化才能掩盖内存延迟，有效利用显卡上的大量并行单元。同时上千个 thread 是正常的，否则无法体现 CUDA 的优势，另外程序的分支结构尽量少。

目前点云去噪算法主要是通过滤波来实现的，滤波充分考虑了采样曲面的内在属性，根据采样曲面属性的不同，选择相应的滤波算子，从而使得采样数据经滤波处理后与原始曲面差异最小。根据滤波算子的不同，可将现有的滤波算法分为：Laplace 滤波算法，Wiener 滤波算法，双边滤波算法，移动最小二乘算法等。Laplace 算子和 Wiener 算子[2]随着迭代次数的增加会产生不同程度的收缩，导致模型变形。移动最小二乘算法[3]对于特征明显的物体，难以取得较好的效果。相比较而言，双边滤波算法可以对点云数据所表现的实体表面，根据需求很好地实现噪声剔除，且能将特征信息的损失降到最低。

本文对 BMD(Bilateral Mesh Denoising)算法[4]进行改进，将 BMD 细分成多个子问题，其中每个子问题都具有高度的并行化，为每个子问题设计 CUDA 核函数，充分利用 GPU 并行性，并改进了法矢计算算法以及光顺函数。实验结果表明，本文算法速度较 CPU 算法有大幅提高，去噪效果较好。

2 基于 CUDA 的双边滤波点云去噪算法

BMD 算法是一种扩散光顺算法，对网格模型中的一点，通过将其沿法矢方向移动一段距离 d_i 来达到光顺效果：

$$v_i^{'} = v_i - \boldsymbol{n}_i \cdot d_i \tag{1}$$

$$d_i = \frac{\sum\limits_{v_j \in N(v_i)} \boldsymbol{n}_i \cdot (v_i - v_j) W_c(\| v_i - v_j \|) W_s(\boldsymbol{n}_i \cdot (v_i - v_j))}{\sum\limits_{v_j \in N(v_i)} W_c(\| v_i - v_j \|) W_s(\boldsymbol{n}_i \cdot (v_i - v_j))} \tag{2}$$

基金项目：江苏省高技术研究计划(工业)基金资助项目(BG2007037)；江苏省自然科学基金资助项目(BK2008262)
作者简介：徐 波(1986－)，男，硕士研究生，主研方向：计算机图形学，三维建模；唐 杰，副教授；武港山，教授、博士生导师
收稿日期：2010-06-20 **E-mail：**xubonjucs@gmail.com

其中，$\boldsymbol{n}_i$ 为点 v_i 的法矢；W_c 和 W_s 分别为距离惩罚函数和特征保持函数，均为高斯函数；$N(v_i)$ 为邻居点。

BMD 算法的几何意义如图 1 所示。其中，$\|v_{i+1}-v_i\|$ 是点 v_{i+1} 到 v_i 的欧氏距离，$\boldsymbol{n}_i\cdot(v_{i+1}-v_i)$ 为向量 $(v_{i+1}-v_i)$ 在法矢上的分量，即图中虚线；t 是点 v_i 切平面。光顺时，v_i 在法向上的偏移量即是所有虚线的加权和。

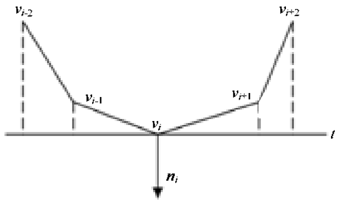


图 1　BMD 算法的几何意义

然而点云数据没有显式邻接关系，常用的方法是将点云数据沿法向和切向进行分离，相邻采样点切向距离近似为空间邻近度，法向距离近似为相邻像元的相似度。本文充分考虑 GPU 的并行性，将光顺过程细分成 4 个并行度较高的步骤：(1)空间划分；(2)K 近邻搜索；(3)法矢计算；(4)光顺，并分别编写 CUDA 核函数。

2.1　空间单元格划分

寻找 K 近邻算法有很多，本文采用适合并行运算的平行单元格法。将点云中每个点分配到某个单元格的过程是完全独立的，每个点的划分采用一个线程。Device 端采用 3 个数组存放计算数据：*buf* 存放顶点坐标信息；*grid* 存放划分后每个单元格中顶点链表头；*link* 存放链接信息。*link*[*i*]中存放的是同一 *grid* 中下个顶点的索引。

具体算法如下：

Host 端：

(1)载入顶点数据。

(2)调用 cudaMalloc 为 3 个数组分配空间，调用 cudaMemcpy 将顶点坐标复制到 buf 中。

(3) 调用核函数 *uniform_grid*<<<(*v_num*-1)/*thread*+1, *thread*>>>(*grid, buf, link, gridsize, bound*)。其中，*v_num* 为顶点数目；*thread* 是自定义的每个 block 中使用的线程数；(*v_num*-1)/*thread*+1 为总共需要的线程块数；*gridsize* 存放 3 个方向上分块数；*bound* 存放包围盒。

Device 端：uniform_grid 核函数

(1)利用下式计算准备划分的顶点索引 *idx*：

$$\begin{aligned} idx = &(blockIdx.y\times blockDim.y+threadIdx.y)\times \\ &(gridDim.x\times blockDim.x)+blockDim.x\times \\ &blockIdx.x+threadIdx.x \end{aligned} \tag{3}$$

(2)根据索引、包围盒及分块数确定该点属于的单元格。

(3)将该点索引插入到对应单元格顶点链表头，修改 *link* 和 *grid* 数组，算法结束。

2.2　K 近邻搜索

寻找每个顶点的 K 邻近点也是一个完全独立的过程，同样为每个顶点分配一个线程。Host 端负责为 GPU 准备数据，并调用 KNN 核函数。

Device 端：KNN 核函数算法

(1)利用式(3)计算顶点索引和单元格索引，设置初始搜索域为当前单元格。

(2)在搜索域中尚未被搜索过的单元格里计算距离当前点的距离，更新邻近点列表，若查找到的邻近点数小于 K，转(3)，否则转(4)。

(3)搜索域沿空间 6 个方向各扩大 1 个单元格距离，转(2)。

(4)计算当前点距离搜索域 6 个面的距离，若存在某个方向上的距离小于当前最大邻近点距离，则沿该方向扩大 1 个单元格，转(2)，否则转(5)。

(5)算法结束。

2.3　法矢计算

寻找 K 近邻是为了能估算当前点的法矢。计算法矢的方法有很多，常用的有计算最小二乘平面、计算移动最小二乘平面、构建局部三角网格。计算移动最小二乘平面涉及非线性优化的问题，计算开销很大；构建局部三角网格的方法不太适合带有噪声的点云。本文采用改进的计算最小二乘平面的方法。

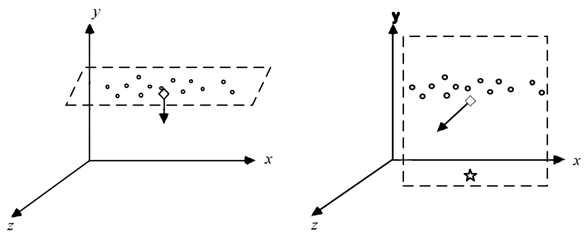
计算最小二乘平面即对点集 N 计算平面 e，使所有 N 中的点到 e 的距离和最小，可用式(4)和式(5)来描述：

$$E=\min\left(\sum_{x\in N}((x-b)\cdot\boldsymbol{n})^2\right) \tag{4}$$

$$b=\frac{\sum_{x\in N}x}{vnum} \tag{5}$$

其中，$\boldsymbol{n}$ 是最小二乘平面的法矢；b 是 N 的重心；*vnum* 是 N 中点的数目。本文采用主成分分析法，以式(4)的协方差矩阵的最小特征值对应的特征向量作为近似的法矢。

然而，由于点云模型中存在着噪声，某些情况下采用上述计算最小二乘平面的方法将会导致法矢与原始模型不符，沿法矢方向光顺将达不到平滑效果。法矢比较如图 2 所示。其中，星点是离群点；圈点是原始点；宝石点是重心。



(a)无噪声的点集　　(b)带噪声的点集

图 2　法矢比较

图 2(a)中虚线平面是对原始模型拟合的最小二乘平面，法矢方向为 y 轴，而图 2(b)中由于存在噪声点，拟合平面的法矢方向为 z 轴，与原始模型不符。噪声使法矢产生偏差，图 2(b)描述的是极端情况。产生这种偏差的原因是在拟合时噪声点和原始点具有了相同的权值。因此，本文提出高斯加权的迭代最小二乘平面拟合算法来解决该问题。

加权的最小二乘平面拟合主要思想是根据点集中不同点对整体的贡献度不同赋予不同权值。远离点集重心的点是噪声点的可能性较大，且对法矢的贡献度较小，如图 2(b)中的星点对整体的法矢贡献度最低。因此，本文考虑根据与点集质心的不同距离赋予点不同的权值，权值函数采用高斯函数，质心也是通过加权迭代计算而得，实验结果表明，法矢的计算效果提升较高。

在 CUDA 上法矢计算(gen_nomal)的核函数算法如下：

(1)利用式(3)计算该线程的顶点索引。

(2)计算此顶点与 K 个邻近点组合成的点集的质心 C_1。

(3)根据每个顶点与 C_1 距离进行高斯加权，再次计算质心 C_2，判断 C_1C_2 距离是否小于某个阈值，若小于则结束质心计算，否则将 C_2 赋给 C_1，重复(3)。

(4)利用下式计算高斯加权的协方差矩阵 $\boldsymbol{M}$：

$$\boldsymbol{M}=\sum_{x\in N}(w_x\cdot(x-C_1))^{\mathrm{T}}\cdot(w_x\cdot(x-C_1)) \tag{6}$$

(5)对 $\boldsymbol{M}$ 进行主成分分析，采用雅可比迭代计算特征值和特征向量，取最小特征值对应的特征向量作为近似法矢，算法结束。

2.4 点云光顺

BMD 算法中首先计算一个点周围的一些点对该点的影响，然后通过沿该点法矢方向移动一段距离来抵消这个影响达到光顺的效果，如式(2)所示。为了能充分保持模型的特征，本文加入了面积加权，如果一个点与周围点的面积权重小则认为其对模型特征的影响较小，并采用 K 近邻的距离均值的平方来近似面积。

Smooth 核函数算法如下：

(1)利用式(3)计算当前光顺的顶点 v_i 的索引，并计算单元格索引。

(2)确定与 v_i 的距离小于阈值 e 的单元格集合，利用式(7)计算影响因子。其中，W_a 是 v_j 与 K 邻近点的距离平均值的平方。

$$d_i=\frac{\sum_{\|v_i-v_j\|<e}\boldsymbol{n}_i\cdot(v_i-v_j)\cdot W_c\cdot W_s\cdot W_a}{\sum_{\|v_i-v_j\|<e}W_c\cdot W_s\cdot W_a} \tag{7}$$

(3)利用式(1)光顺，算法结束。

3 实验结果与分析

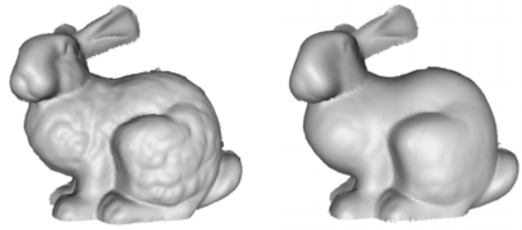
本文算法的实验环境：CPU 为 Intel Core 2 Duo E6550@ 2.33 GHz，内存为 2 GB，显卡为 GeForce9800GT。

通常光顺的效果仅是通过观察而得到的，为能定量反映某些性质，本文创建了一个球面，半径 R 为 20，球面具有 50 833 个点，在距离球心 0.8R~1.2R 的空间内添加 3 000 个随机噪声点，实验结果如表 1 所示，其中，K 为计算法矢时选取的邻近点数；R 为球面的平均半径；E 为噪声点与球面距离的期望；D 为噪声点与球面距离的方差；N 为与球面距离超过 0.05R 的噪声点数；T 为算法执行时间。

表 1 球面模型光顺结果比较

模型	K	R	E	D	N	T/ms
原始模型	-	20.00	2.04	5.46	2 231	-
BMD	32	20.46	0.23	0.28	40	1 825
BMD+面积加权	32	20.38	0.57	0.50	473	1 967
BMD+迭代法矢	32	20.46	0.18	0.05	4	1 863
BMD+迭代法矢+面积加权	32	20.43	0.24	0.12	28	2 003

比较表 1 中一般 BMD 算法与高斯迭代法矢的 BMD 算法可见，平均半径 R 相同，但 E, D, N 都明显减小，说明本文提出的高斯迭代法矢是行之有效的，能改善点云中法矢的计算，尤其对于大尺度的噪声；面积加权的 BMD 算法相比于一般 BMD 算法，除了平均半径 R 缩小，E, D, N 都明显增大，这是由于采用面积加权能更好地保持模型的细小尖锐的特征，缓和过光顺。因此，与一般 BMD 算法相比有所削弱光顺效果，但模型细节更佳，本文综合采取了高斯迭代法矢和面积加权的方法。实验效果如图 3 所示。



(a)噪声模型重建结果 (b)光顺后重建结果

图 3 点云光顺重建结果

关于法矢的计算，举一例说明实验，有如下几个点：(1, 0, 0), (0, 0.1, 0), (2, 0.15, 0), (1.5, -0.1, 0), (0.5, -0.15, 0), (2.5, 0.05, 0), (3.0, -0.1, 0)，这 7 个点在 xy 平面上，法矢为(0, 0, 1)，加入噪声点(1.5, 0,4)，等权计算法矢的结果为(0.004, 0.999, -0.002)，而采用本文算法计算结果为(0, 0.000 2, 1)，本文的方法更加符合模型法矢，错误的法矢将会使光顺失效，因此，采用加权的法矢计算方法使得光顺更有效。

将本文算法与 CPU 算法进行比较，结果如表 2 所示，可见，本文算法速度有大幅提升。

表 2 本文算法与 CPU 算法的光顺结果比较

点数	K	CPU 算法/s	本文算法/s	加速比
12 580	8	23	0.16	148
50 880	8	106	0.7	151
114 450	8	250	1.19	210
458 070	8	719	1.37	525
1 272 740	8	1 439	8.12	177

4 结束语

本文提出的基于 CUDA 的点云去噪算法改善了法矢计算，缓和了双边滤波的过光顺，并充分利用 CUDA 并行性极大地提高了计算速度，使工程中应用更方便有效。目前算法主要利用的是 CUDA 全局存储空间，下一步的工作需要充分利用 CUDA 的线程共享空间，设计各级存储，提高算法效率。

参考文献

[1] CUDA Programming Guide 2.3[Z]. (2009-08-26). http://www.nvidia.com/object/cuda_develop.html.

[2] Peng J, Strela V, Zorin D. A Simple Algorithm for Surface Denoising[C]//Proc. of the 12th Conference on Visualization. San Deigo, California, USA: IEEE Press, 2001: 107-112.

[3] Alexa M, Behr J, Cohen-Or D. Computing and Rendering Point Set Surface[J]. IEEE Trans. on Visualization and Computer Graphics, 2003, 9(1): 3-15.

[4] Fleishman S, Drori I, Cohen-Or D. Bilateral Mesh Denoising[C]//Proc. of SIGGRAPH'03. San Deigo, California, USA: [s. n.], 2003: 950-953.

编辑 金胡考

(上接第 223 页)

[3] Leacock C, Chodorow M, Miller G A. Combining Local Context and WordNet Similarity for Sense Identification[M]//Fellbaum C. WordNet: An Electronic Lexical Database. Cambridge, USA: MIT Press, 1998: 265-283.

[4] Fan Jianping, Keim D A, Gao Yuli, et al. Personalized Image Recommendation via Exploratory Search from Large-Scale Flickr Image Collections[J]. IEEE Trans. on Circuits and Systems for Video Technology, 2008, 18(8): 1-19.

[5] 卫颖奇, 彭进业. 一种基于混合模型的特征提取方法[C]//第 3 届全国嵌入式技术和信息处理联合学术会议论文集. 昆明: [出版者不详], 2009.

[6] Gao Yuli, Fan Jianping, Luo Hangzai, et al. A Novel Approach for Filtering out Junk Images from Google Search Results[C]//Proc. of Int'l Conf. on Multimedia Modeling. Kyoto, Japan: [s. n.], 2008.

编辑 金胡考