一种基于 CUDA 的三维点云快速光顺算法

唐 杰, 徐 波, 宫中樑, 武港山
(南京大学软件新技术国家重点实验室, 南京 210093)

摘 要：提出了一种基于 CUDA 的点云光顺算法。算法细分成点云空间划分，K 邻近搜索，法矢估算以及光顺等四个独立的且并行程度非常高的步骤。*设计了基于 CUDA 的点云空间平均单元格划分算法及数据结构，有效提升了点云的划分效率；设计了基于 CUDA 的空间 K 邻近搜索算法；改进了点云法矢估算方法，提出了高斯加权的法矢计算方法，有效改善了法矢估算效果；在光顺过程中加入了邻近点的面积影响因子，缓和了过光顺等不足。*最后通过实验验证了算法的有效性。

关键词：光顺; CUDA; GPU 计算; 点云
中图分类号：TP391.9 **文献标识码：**A **文章编号：**1004-731X (2012) 08-1633-05

Fast Fairing of 3D Point Clouds Using CUDA

TANG Jie, XU Bo, GONG Zhong-liang, WU Gang-shan
(National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

Abstract: A CUDA-based point cloud fairing algorithm was proposed. The algorithm is composed of four steps with great parallelism including point cloud space partitioning, K-nearest neighbors searching, the normal estimation and fairing. *A CUDA-based point cloud partition method as well as its data structure which utilizes the uniform grid was designed, which improved the efficiency of partitioning greatly. A CUDA-based algorithm for K-nearest neighbors search was designed. An improved normal estimation method was proposed which utilized Gaussian weighted method to calculating normal vector and improved the precision of normal estimation. The impact factor of the adjacent area was introduced to improve the effect of smoothing and alleviate the degree of over smoothing.* Finally, the experiments verify the effectiveness of the algorithm.
Key words: fairing; CUDA; GPU computing; point clouds

引 言

随着计算机技术的高速发展及精密的测量技术的出现，点云数据模型在三维建模及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因素使得生成的点云数据往往含有噪声。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。点云光顺的目的就是去除其中的噪声高频信号。一个好的光顺去噪算法除了能够有效地剔除点云模型中的各种噪声外还需要能够有效保持模型的固有几何特征，同时具有较低的算法时间复杂度和空间复杂度。这几条往往相互冲突，很难协调。Fleishman[1]等人提出了一种基于双边滤波的三角网格模型光顺算法，很好地解决了上述问题。然而，点云模型与三角网格模型不同，它没有显式的邻接关系，也不好确定每个顶点的法矢方向，这些都给点云的光顺带来很大的挑战。

近年来，图形处理单元在通用计算中的应用越来越广泛。随着 nVidia 公司推出统一设备架构（CUDA），研究人员可以更加方便地开发 GPU 通用高性能计算。自 CUDA 问世后，就一直成为众多领域的研究热点，其中与三维模型处理相关的研究也非常多。Zhou[2]和 Santos[3]提出了采用 CUDA 来进行模型的空间 KD-tree 划分，Kalojanov[4]提出了对模型进行空间平均单元格划分的 CUDA 方法。Garcia[5]提出了基于 CUDA 的空间 KNN 搜索算法。虽然上述这些算法极大地促进了三维模型处理的进展，但结合具体需求，依然存在很多问题需要去研究。

收稿日期：2010-07-13 **修回日期：**2011-11-07
基金项目：国家高技术研究发展计划(863) (2007AA06A402), 国家科技重大专项(2011ZX05035-004-004HZ)
作者简介：唐杰(1971-), 男, 江苏南京人, 博士, 副教授, 研究方向为三维建模, 多媒体技术; **徐波**(1986-), 男, 硕士生, 研究方向为三维建模; **宫中樑**(1984-), 男, 硕士研究生, 研究方向为 GPU 计算; **武港山**(1967-), 男, 博士, 教授, 博导, 研究方向为多媒体技术。

本文对网格模型双边滤波算法[1]进行了改进，将其推广为基于 CUDA 的点云模型光顺。算法细分成点云空间平均单元格划分，点云邻域搜索，法矢估算和光顺四个独立的并且具有非常高并行性的步骤，分别为每个步骤设计 CUDA 核函数。另外，提出了适应 GPU 的高斯迭代加权法矢估算方法，并在光顺过程中加入了邻近点的面积影响因子，有效地缓和了过光顺问题。最终实验验证了本文算法的有效性和高效性。

1 相关工作

点云滤波根据特征保持性和噪声在各个方向上的扩散方式可以分为各向同性算法和各向异性算法。根据滤波算子的不同，可以将现有的滤波算法分为如下几类：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等。

Laplace 滤波算法：Laplace 滤波算子通过扩散高频几何噪声从而达到滤波的目的[7,8]，Laplace 滤波的本质是一种曲面能量极小化的问题。随着迭代次数的增加，将不可避免地产生模型收缩现象，导致模型变形。为了解决这个问题，Vollmer[9]提出了将滤波后的顶点推向先前位置来避免模型的变形，但未能从根本上解决问题。这三个算法都是各向同性的滤波算法，对于特征点和噪声点同样处理，造成模型的一些重要特征丢失。

局部自适应 Wiener 滤波算法：Peng[10]将局部自适应滤波算法成功应用到网格和点云的去噪中，但这种方法必须知道局部的拓扑结构。Alexa[11]定义了一种用于网格滤波的 Wiener 滤波算子，该算子类似 Laplace 算子，也是通过将高频噪声扩散到相邻区域来实现去噪的，该算法也必须建立在局部拓扑结构已知或者是参数曲面的基础上。

双边滤波算法：与 Laplace 滤波算子不同，双边滤波算子通过采用相邻采样点的加权来修正当前采样的位置，从而达到光顺效果，同时剔除与当前采样点差异太大的相邻采样点，从而达到保形的目的。Fleishmann[1]、Jones[12]等人成功地将双边滤波算子应用到了网格的滤波处理之中。

移动最小二乘算法：移动最小二乘算法(Moving Least squares,简称 MLS)是一种基于迭代的局部曲面逼近技术，对于输入的点云数据，算法首先构建一个高度函数来逼近采样曲面，并通过一系列后处理对构建的曲面方程进行调整，从而得到满足局部最小二乘准则的曲面逼近方程。Alexa[13]等人基于迭代优化方法，为点集曲面建立一个移动最小二乘曲面，通过将噪声点移至所逼近的二次曲面上来达到去噪的目的，然而该方法难以保持点模型的特征。

综上，Laplace 滤波算子和 Wiener 滤波算子无论各向同性还是各向异性，随着迭代次数的增加，模型将产生不同程度的收缩，导致变形。MLS 能够取得很好的滤波效果，但是对于特征明显的物体也难以取得较好效果，且将在很大程度上削弱模型的轮廓特征。相比较而言，双边滤波算法无论对于点云还是网格均可以根据需求，很好地实现噪声剔除，且在剔除过程中，特征信息损失较低。

实际上对于大规模点云模型进行光顺处理是非常耗时的，而利用 GPU 来加速点云光顺去噪的算法却很少，Jalba[14]提出了一种采用 GPU 加速的基于曲率流的网格光顺算法，该算法采用传统 GPGPU 思想，速度提升较快。另外，Ni[15]提出了一种采用 GPU 加速的四边形网格光顺方法，这些算法处理的对象都并非点云。

2 算法

双边滤波算法[1]具有较好的去噪效果，同时能有效地保持原始模型的尖锐特征。然而，现有的算法只能处理三角网格等具有显式连接性（connectivity）信息的模型。此外，在处理非均匀分布的模型时，现有算法大多利用网格模型中的三角片的面积作为权值来调整每个顶点得贡献。但在处理点云模型时，就没有这么方便了。为此，本文对双边滤波算法进行了改进，将其推广到点云模型的光顺，同时利用 CUDA 技术实现了基于 GPU 计算的并行光顺。

2.1 双边滤波算法

双边滤波算法是一种扩散光顺算法，网格模型中的一点根据其受到邻近点的影响，沿法矢方向移动一段距离来达到光顺的效果：

$$v_i^{'} = v_i - n_i \cdot d_i \tag{1}$$

$$d_i = \frac{\sum_{v_j \in N(v_i)} n_i \cdot (v_i - v_j) \cdot W_c(\|v_i - v_j\|) \cdot W_s(n_i(v_i - v_j))}{\sum_{v_j \in N(v_i)} W_c(\|v_i - v_j\|) \cdot W_s(n_i(v_i - v_j))} \tag{2}$$

其中，$\boldsymbol{n_i}$是顶点$\boldsymbol{v_i}$的法矢，$N(v_i)$是顶点$\boldsymbol{v_i}$的邻居点，定义为：$\| v_i - q_i \| < r = 2\sigma_c$,即所有到点$\boldsymbol{v_i}$的距离小于$2\sigma_c$的点。W_c是距离惩罚函数: $W_c(x) = \exp(-x^2/2\sigma_c^2)$, W_s 是特征保持函数: $W_s(x) = \exp(-x^2/2\sigma_s^2)$。

然而点云数据模型既没有法矢信息，也没有显式的邻接关系，这给其光顺操作带来了不小的麻烦。本文充分考虑了 GPU 的并行性，将光顺过程细分成 4 个并行度非常高的步骤，分别设计核函数：

1. 点云的空间平均单元格划分，建立点云的快速索引结构
2. 点云空间 K 邻近搜索，确定每个点的邻域
3. 根据每个点的邻域，对法矢进行近似估算
4. 双边滤波光顺

2.2 平均单元格划分

平均单元格是一种常用的数据结构，用于快速获取多

维空间中指定位置的某点。一般的做法是首先将点云外接包围盒平均划分成若干个大小相同的小单元格，每个顶点按其空间位置分配到一个小单元格中。使用时，我们可以获取制定位置的所有顶点。

我们使用 CUDA 技术实现了基于 GPU 的点云平均单元格数据结构。CUDA 不支持指针，也不支持动态分配内存。而每个单元格中的顶点数是不一样的。这给基于 CUDA 的平均单元格构建带来了困难。为此，我们设计了一种适合 CUDA 的平均单元格数据结构，如图 1 所示。数据结构包括三个固定长度的数组，分别是顶点几何信息数组 d_vertex，平均单元格数组 d_grid_head 以及顶点链接数组 d_vertex_link。

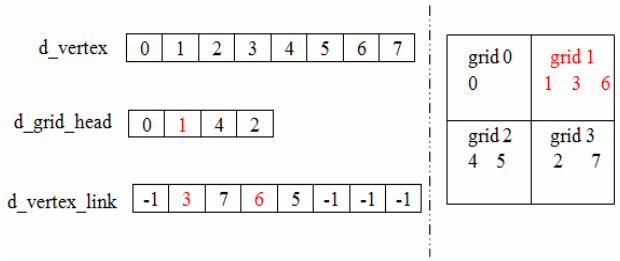

图 1　平均单元格的数据结构

划分时，Host 端载入顶点数据，为 Device 端的数组分配空间，并将顶点数据传送到 Device 端，调用核函数 *UniformGrid*，每个线程块的线程数定义为 B，线程块数为 (v_num-1)/B+1。Device 端的算法伪代码如下：

算法 1 点云空间单元格划分算法

```
__global__ void UniformGrid(){
    int vidx = blockIdx.x*blockDim.x+threadIdx.x;
    if(vidx>vnum) return;
    根据单元格的分辨率及顶点坐标计算该点属于的单元格索引 gidx。
    d_vertex_link[vidx]=atomicExch(&d_grid_head[gidx],vidx);
}
```

由于多个线程可能同时对 d_grid_head 数组中同一位置进行写操作，为了使得互斥读写，避免冲突，因此算法 1 中采用了一个原子操作 *atomicExch*()来交换两个元素。

2.3 K 邻近查询

点云模型中由于缺乏点之间的拓扑关系，因此通常通过采样点的邻域来估算采样点的属性，如法矢，曲率等。一般采样点的邻域构造有两种方式，一种是欧氏邻域[5]，欧氏邻域将所有在以采样点 S 为中心，半径为 r 的球的内部的采样点作为该采样点的邻域；另外一种方式是 K 近邻邻域[18]，K 近邻方法选取采样空间中距离采样点 S 距离最近的 K 个点作为 S 的邻域。欧氏邻域受到模型分布的影响，对于不规则的模型处理效果较差，K 近邻的自适应性就比较好，能够处理各种模型。

K 近邻算法的最简单的做法是先计算采样点与所有样本的距离，然后通过排序选择出最近的 K 个点，这是种蛮力算法，效率较低。优化的做法是先对样本空间进行划分，然后计算距离，空间划分的目的是为了削减参加距离计算的样本个数。

K 近邻算法是一个并行度非常高的算法，因此比较适合采用 GPU 来运算。Garcia[5]提出了 GPU 加速的 KNN 算法。他们的算法属于蛮力算法。对于维度很高，样本数少的数据效果较佳，但对于点云模型这种样本很多，维度较低的数据，这种算法并不能达到很好效果。本文采用空间划分的方式来实现 CUDA 下的 K 近邻搜索，上节已经详细介绍了点云的 GPU 下划分的方法，具体的 K 近邻搜索核函数算法如算法 2 所示。

算法 2 K 邻近搜索算法

Step1: 计算当前线程处理的顶点索引和单元格索引，设置搜索域为当前单元格。

Step2: 在搜索域中尚未被搜索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 Step3,否则转 Step4。

Step3: 搜索域沿六个方向上各扩大一个单元格，转 Step2。

Step4: 计算当前点距离搜索域六个面的距离，若存在某个方向上的距离小于当前最大邻近点距离，则沿该方向上扩大一个单元格，转 Step2，否则转 Step5。

Step5: 算法结束。

图 2 说明了搜索域需要延伸的情况，搜索域延伸的界限即模型的包围盒。图中采样点 ***S*** 在单元格 B 中的 K 邻近点的最大距离是 r，***S*** 距离单元格 A 的距离 $d< r$，因此需要将搜索域延伸至 A，很显然 A 中采样点 ***S***'距离 ***S*** 更近。

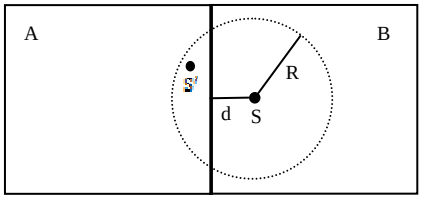

图 2　K 邻近算法搜索域延伸示例

2.4 法矢计算

BMD 算法中需要沿采样点的法矢方向调整模型，因此计算法矢是一个非常重要的过程，错误的法矢可能会导致光顺效果还不如原始模型。目前，关于点云法矢估算的研究很多，常见的是对每个采样点的 K 近邻进行三角剖分，计算出相邻三角片的法向量，再对这些法向量进行加权计

算采样点的法向，如 Guo[16]等。Hoppe[17]最早提出了利用最小二乘平面来逼近采样点邻域，把该平面的法向量作为采样点的法向。另外一种法向估计方法即移动最小二乘法[13]，该方法虽然比较精确，但计算量也较大。

Hoppe 的算法在点云分步密度比较均匀时，效果较好。但噪声使得简单的平面拟合计算的法矢产生偏差。产生这种偏差的原因是由于在拟合时噪声点与采样点具有了相同权值，距离重心近的点与距离重心远的点具有了相同权值。本文提出了高斯加权的最小二乘平面拟合算法来解决此问题（如式 3 所示）。采用高斯加权的方法需要比较大的计算量，如果采用 CPU 来执行会比较耗时，但 GPU 比较适合处理计算量大的任务，因此能够较好地进行高斯加权最小二乘平面计算。

$$\begin{aligned} M &= \sum_{v_j \in N(v_i)} (w_g(\| v_j - v_c \|)(v_j - v_c)^T)(w_g(\| v_j - v_c \|)(v_j - v_c)) \\ v_c &= \sum_{v_j \in N(v_i)} w_g(\| v_j - v_i \|) v_j \\ w_g(x) &= e^{-x^2/(2\sigma^2)} \end{aligned} \tag{3}$$

其中，N 是某顶点的邻域，$\boldsymbol{v}_c$ 是 N 的重心。

2.5 点云光顺

2.1 节中已经介绍了网格模型双边滤波算法。该算法是将网格模型中每个顶点沿其法矢方向移动一个距离。这个距离由每个顶点的邻域顶点确定。在应用过程中，我们发现双边滤波算法对分布较为均匀的点云模型效果较好。而对非均匀分布的点云则效果不是十分理想。究其原因，是因为非均匀分布的点对其它点的贡献也应该不一样。为此，我们对网格模型双边滤波算法进行了改进，将其应用到点云模型上。同时增加了面积权因子来调节每个点在计算其它顶点位移时的贡献。由于点云没有显示的面积指标，我们采用每个点到其周围最近的 5 个顶点的平均距离的平方来作为代替。具体公式如下所示：

$$d_i = \frac{\sum_{v_j \in N(v_i)} n_i \cdot (v_i - v_j) \cdot W_c(\|v_i - v_j\|) \cdot W_s(n_i(v_i - v_j)) W_a(v_j)}{\sum_{v_j \in N(v_i)} W_c(\|v_i - v_j\|) \cdot W_s(n_i(v_i - v_j)) W_a(v_j)} \tag{4}$$

其中 $W_a(\boldsymbol{v}_j)$是点 $\boldsymbol{v}_j$ 到其 5 个最近点的平均距离的平方。该计算在 CPU 上非常耗时，但在 GPU 上，即使是上百万顶点的点云，我们也可以基本上做到实时计算。

3 实验结果

本文算法已在 CUDA 2.3 下实现，并在多种不同实验环境下进行了测试。软件开发平台是 MS Visual Studio 2008，编程语言为 C。实验的 GPU 设备指标如表 1 所示。

3.1 *K* 邻近查询

图 3 是 Buddha 模型（543652 个顶点）在测试环境二下的 K 近邻搜索结果，从图中可以看出 K=8 和 K=16 的曲线走势基本一致，随着单元格分辨率变高，K 近邻搜索时间大幅降低。K=8 的最快搜索时间约为 0.4s，基本上能够满足交互需求。

表 1 GPU 测试设备性能指标

	Quadro 1700	Geforce 9800	Tesla c1060
处理器个数	4(32)	16(128)	30(240)
处理器频率	—	1688 MHz	933 GFlops
显存	512M	512M	4G
显存带宽	12.8GB/s	70.4GB/s	102GB/s

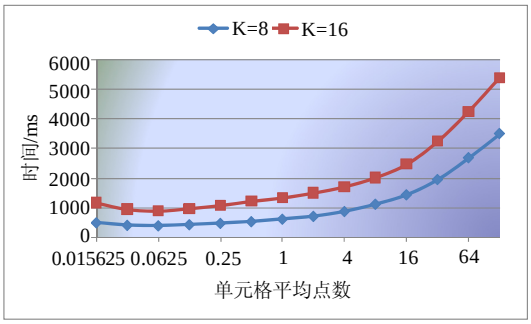

图 3 Buddha 模型的 K 近邻搜索结果

Connor[18]等人提出了一种快速的 k-邻近查询算法。在文章中，他们给出了一些实验结果。其中查询 2 百万左右 k=1 的邻近查询需要 2.1 秒左右。但当 k 增加时，他们算法的耗时也迅速增加。在处理 226 万点云查找 k=20 的邻近点时，本文算法在测试环境一、二和三上分别耗时 9.5 秒、3 秒和 1 秒。而 Connor 等人的算法在 CPU 上耗时约 500 秒。

3.2 光顺

通常点云的光顺效果仅是通过观察而得，为了能够定量反应某些性质，我们创建了一个球面，半径为 20，球面具有 50000 个点，在距离球心 0.8R~1.2R 的空间内添加了 3000 个噪声点，光顺的结果如表 2 所示。

表 2 球面模型光顺结果比较

模型	K	R	E	D	N
原始模型	/	20	1.9740	5.2418	2226
BMD	32	20.3055	0.4080	0.7763	235
面积加权 BMD	32	20.2497	0.9283	1.7609	434
高斯加权法矢+面积加权 BMD	32	20.3010	0.2271	0.2104	151
高斯加权法矢+BMD	32	20.3142	0.1815	0.1342	65

表 2 中 K 为邻近点数目，R 为球面的平均半径，E 为噪声点与球面的距离的期望，D 为噪声点与球面距离的方差，N 为与球面距离超过 0.05R 的噪声点数目。比较表中一般 BMD 算法与高斯迭代法矢的 BMD 算法，高斯迭代法矢的 BMD 算法半径略有增加，但 E，D，N 都明显减少，说明本文提出的高斯迭代法矢计算方法是行之有效的，能

够改善点云中法矢的计算，尤其对于大尺度的噪声。比较面积加权的 BMD 算法与一般 BMD 算法，除了平均半径缩小，E，D，N 都明显增大，这是由于采用面积加权能够更好地保持模型的细小尖锐特征，缓和过光顺，因此与一般的 BMD 算法相比有所削弱光顺效果，但模型细节更佳。面积加权与高斯迭代法矢并不是一对矛盾的概念。本文综合采取了高斯加权法矢和面积加权的方法。

本文对 CUDA 下的整个光顺算法最快时间与 CPU 算法进行了比较，如表 3。GPU 算法的时间包括了算法所有步骤的时间，除了模型的加载时间和光顺结果的写回时间，K 近邻搜索中的 K=8。

表 3　CPU 与 GPU 光顺速率比较

模型点数	12683	50833	114446	458074	1272739
CPU(s)	23	106	250	719	1439
测试环境二(s)	0.135	0.533	1.202	4.954	14.682
加速比	170.4	198.9	208.0	145.1	98.0
测试环境三(s)	0.027	0.099	0.215	0.883	2.617
加速比	851.9	1070.7	1162.8	814.2	550.0

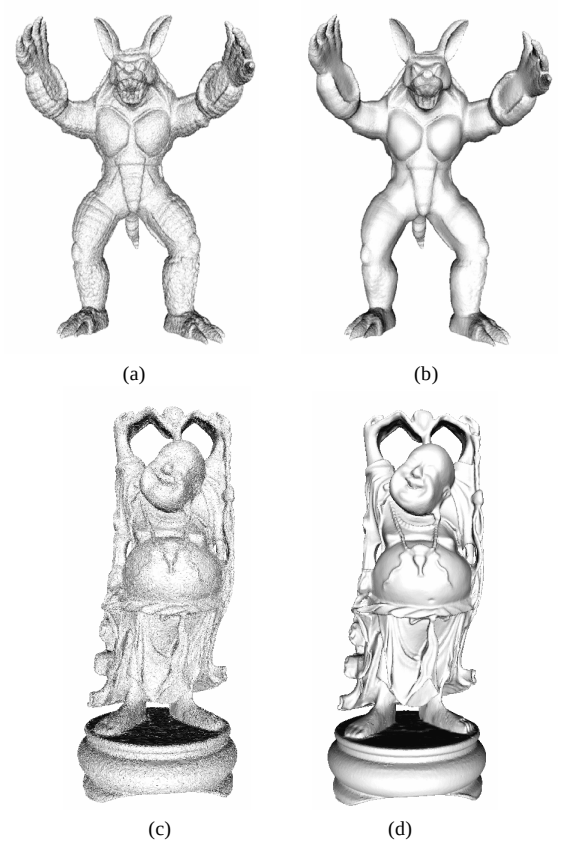

图 4　本文算法的光顺结果

图 4 所示为本文算法的一些结果，其中(a)(c)是噪声模型，(b)(d)是光顺模型。图 4b 描述了对 Armadillo 噪声模型光顺的结果，光顺方法采用高斯迭代法矢与面积加权的 BMD 算法。从图中可以看出光顺效果比较明显，能够对原始模型的表面进行平滑处理。但依然存在缺陷，一些重要的细节变得模糊，如 Armadillo 的腿部，手部以及手指上的皱纹被平滑了一部分，这是由于 BMD 算法在光顺一个点时是通过周围点对当前点的影响来进行处理的，在平滑噪声的同时，不可避免的会对一些细小特征也进行平滑。

4　结论

点云的光顺去噪是三维模型处理中非常重要且经常必须要经历的一个过程。由于点云模型的数据量越来越大，光顺这些模型比较耗时。本文提出了一种基于 CUDA 的点云光顺算法。该算法将网格模型双边滤波算法推广到点云模型光顺应用中，同时将双边滤波算法细分成多个具有非常高的并行度的步骤：点云空间单元格划分，K 近邻搜索，法矢估算和双边滤波光顺。设计了 CUDA 下的数据结构并分别为这四个步骤编写 CUDA 核函数。另外改进了法矢估算的方法，引入了高斯加权估算法矢的方法，优化了噪声信号法矢的估算。最后在光顺过程中加入了面积加权来缓和双边滤波算法的过光顺。实验表明，本文算法的运行速度比较快，光顺效果也较好。

点云模型越来越复杂，如果我们在整个点云范围内使用同样的光顺操作，容易在局部产生过光顺或欠光顺的问题。非线性各向异性光顺可以解决这个问题，这也是本文后续的工作之一。在基于 CUDA 的 GPU 计算中，如果能有效地利用 GPU 的共享内存，就可以获得更高的提速效果。然而，结合具体的应用需求，如何利用共享内存仍然是需要研究的问题。

参考文献：

[1] Fleishman S, Drori l, Cohen-Or D. Bilateral mesh denoising [C]// Proceedings of ACM SIGGRAPH 2003. USA: ACM, 2003: 950-953.

[2] Zhou K, Hou Q, Wand R, Guo B. Real-time kd-tree construction on graphics hardware [J]. ACM Transaction on Graphics (S0730-0301), 2008, 27(5): 1-11.

[3] Santos A L D, Teixeira J M X N, Farias T S M C D, *et al.* KD-Tree traversal implementations for ray tracing on massive multiprocessors: a comparative study [C]// Computer Architecture and High Performance Computing, Sao Paulo, Brazil, October, 2009. USA: IEEE, 2009: 41-48.

[4] Kalojanov J, Slusallek P. A parallel algorithm for construction of uniform grids [C]// Proceedings of the Conference on High Performance Graphics, New Orleans, Louisiana, USA, 2009. USA: ACM, 2009: 23-28.

[5] Garcia V, Debreuve E, Barlaud M. Fast k nearest neighbor search using GPU [C]// Computer Vision and Pattern Recognition Workshops, Anchorage, AK, USA, June, 2008. USA: IEEE Computer Society, 2008: 1-6.

[6] Shenshen Liang, Cheng Wang, Ying Liu, Liheng Jian. CUKNN: A parallel implementation of K-nearest neighbor on CUDA-enabled GPU [C]// Information, Computing and Telecommunication, Beijing, China, 2009. USA: IEEE Computer Society, 2009: 415-418.

（下转第 1642 页）

这种关系。

从图 3 中还可以看出在方案二和方案三中，犯规事件的测试结果都较低，这主要是因为现场环境音有些尖锐噪音容易被误检为哨音，从而使得犯规事件检测存在一定的误检，使得查准率较低。因此后期工作中对犯规事件可以考虑检测裁判员镜头，以提高犯规事件的检测效果。

6 结论

提出了一种基于 HMM 融合多模态对象的视频语义分析方法。先是基于 CHMM 实现音频分类，然后根据时间对应关系融合视频流和音频流，在视频流中相应的镜头中基于 DHMM 再次融合图像对象、运动对象和文本对象实现精彩事件和一般事件的检测。与已有的基于 HMM 实现模式分类的论文不同的是，本文对 DHMM 的模型结构、参数初始值尤其是参数约束条件进行了详细的描述。实验证明算法效果较好。在后续的工作中，将要确定更丰富、更高层次的镜头语义，如裁判员镜头等，还要细化“语义事件”，如角球、点球等，并将本文方法推广到其他体育类视频检测中。

参考文献:

[1] 金国英, 陶霖密, 徐光佑, 张翔. 基于 HHMM 的多线索融合和事件推理方法[J]. 清华大学学报(自然科学版), 2007, 47(1): 112-115.

[2] 杨颖, 林守勋, 张勇东, 唐胜. 基于动态规划融合多模态的足球视频事件分析[J]. 计算机辅助设计与图形学学报, 2008, 20(8): 1056-1063.

[3] J Y Chen, Y H Li, L D Wu, S Y Lao. Semantic event detection in soccer video by integrating multi-features using Bayesian network [C]// Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speed Proceeding, 2004, Oct.

[4] 刘宇驰, 栾悉道, 戴端辉, 吴玲达. 多模态体育视频语义分析[J]. 计算机科学, 2007, 34(1): 109-111.

[5] Y Yang, S X Lin, Y D Zhang, *et al.* Highlights extraction in soccer avideos based on goal_mouth detection [C]// IEEE Proc. ISSPA 2007. USA: IEEE, 2007: 1-4.

[6] 王扉. 体育视频的内容分析技术研究 [D]. 北京: 中国科学院计算技术研究所, 2005.

[7] 刘宇驰, 吴玲达. 基于 HMM 的足球视频语义结构分析[J]. 计算机工程与应用, 2006, 28: 174-176.

[8] 张玉珍, 魏带娣, 王建宇, 戴跃伟. 基于多媒态融合的足球视频语义分析[J]. 计算机科学, 2010, 37(7): 273-276.

[9] 张玉珍, 王建宇, 戴跃伟. 基于自适应双阈值和主色率的足球视频镜头的分割[J]. 南京理工大学学报(自然版), 2009, 4(8): 432-437. (EI 收录: 20093912337997)

[10] 张玉珍, 何新, 王建宇, 戴跃伟. 一种基于 SVM 的高效球门检测方法[J]. 南京理工大学学报(自然版), 2010, 34(1): 13-18. (EI 收录: 20101612858626).

[11] 谢锦辉. 隐 Markov 模型(HMM)及其在语音处理中的应用[M]. 第 1 版. 武汉: 华中理工大学出版社, 1995.

[12] 彭培华, 曲波, 陈荣胜. 基于支持向量机的小波域视频字幕检测与提取[J]. 华南理工大学学报(自然科学版), 2004, 32(S): 63-66.

(上接第 1637 页)

[7] G Taubin. A signal processing approach to fair surface design [C]// Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Los Angeles, California, USA, 1995. USA: ACM, 1995: 351-358.

[8] M Desbrun, M Meyer, P Schröder, *et al.* Implicit fairing of irregular meshes using diffusion and curvature flow [C]// Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Los Angeles, California, USA, 1999. USA: ACM, 1999: 317-324.

[9] Vollmer J, Mencl R, Muller H. Improved Laplacian smoothing of noisy surface meshes [J]. Computer Graphics Forum (S0167-7055), 1999: 18(3): 131-138.

[10] Peng J, Strela V, Zorin D. A simple algorithm for surface denoising [C]// Proceedings of the conference on Visualization, San Diego, California, USA, 2001. USA: IEEE Computer Society, 2001: 107-112.

[11] Alexa M. Wiener filtering of meshes [C]// Proceedings of Shape Modeling International, Calgary, Alberta, Canada, 2002. USA: IEEE Computer Society, 2002: 51-57.

[12] Jones T R, Durand F, Desbrun M. Non-iterative, feature- preserving mesh smoothing [C]// SIGGRAPH 2003. USA: ACM, 2003: 943-949.

[13] Alexa M, Behr J, Cohen-Or D, *et al.* Computing and rendering point set surfaces [J]. IEEE Transaction on Visualization and Computer Graphics (S1077-2626), 2003, 9(1): 3-15.

[14] Jalba A C, Jos B T M Roerdink. Efficient surface reconstruction from noisy data using regularized membrane potentials [J]. IEEE Transactions on Image Processing (S1057-7149), 2009, 18(5): 1119-1134.

[15] Ni T, Yeo Y, Myles A, Goel V, Peters J. GPU smoothing of quad meshes [C]// IEEE International Conference on Shape Modeling and Applications, Stony Brook, NY, USA, June, 2008. USA: IEEE Computer Society, 2008: 3-9.

[16] Guo B. Surface reconstruction: from points to splines [J]. Computer Aided Design (S0010-4485), 1997, 29(4): 269-277.

[17] Hoppe H, DeRose T, Duehamp T, *et al.* Surface reconstruction from unorganized points [J]. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, New York, NY, USA, 1992. USA: ACM, 1992: 71-78.

[18] Connor M, Kumar P. Parallel Construction of k-Nearest Neighbor Graphs for Point Clouds [C]// Eurographics Symposium on Point-Based Graphics. USA: ACM, 2008: 25-32.