DOI:10.13203/j.whugis20200001 文章编号:1671-8860(2022)07-1035-08

一种利用GPU加速的轨迹线热力图生成显示方法

朱峥嵘[1] 黄亚锋[1] 赵立营[1]

1 南京电子技术研究所,江苏 南京,210039

摘 要:为解决大数据量带来的热力图生成效率低的问题,引入基于图形处理器(graphic processing unit,GPU)的并行计算方法,并结合轨迹线模型,提出了一种利用GPU加速的轨迹线热力图生成显示方法。首先,针对轨迹点分布不均、邻域半径设置不合理等条件下产生的热力值不连续、不均等问题,采用轨迹线模型提升了热力图的效果。其次,针对大规模数据计算产生的热力图生成效率低的问题,通过GPU并行计算并配合内核函数参数调优、循环展开、像素缓冲对象显示等策略大幅提升算法计算效率。实验结果表明,所提方法较传统的基于中央处理器(central processing unit,CPU)的方法计算效率提升了5~30倍,且随着图像分辨率和轨迹数据的增加,算法加速比有逐步上升的趋势。

关键词:热力图;GPU并行计算;轨迹线;循环展开;PBO显示

中图分类号:P208 **文献标志码**:A

随着全球定位系统(global positioning system,GPS)、雷达等位置传感器技术的发展,人们能够获得越来越多的目标运动轨迹线数据。轨迹线是包含轨迹点位置和属性的动态数组,是目标运动数据最常用的表达形式[1-2]。轨迹线分析对于挖掘目标活动规律、预测目标行为[3-7]具有重要意义。轨迹线热力图分析是目标运动轨迹线分析的有效工具,常以轨迹线簇为输入来分析轨迹线的核密度等特征,并通过颜色、饱和度等对轨迹线簇特征进行可视化。热力图分析综合利用分析计算和信息可视化方法,通过人机交互方法挖掘预设结果外的新模式特征,是解决信息爆炸问题的重要手段,文献[8]将其归类为可视化分析方法。因此,热力图分析和显示成为地理信息科学、数据挖掘、战场态势等众多领域关注的热点问题。

现有热力图分析研究包括热力图效果优化和图像生成效率提升两个方向。其中图像效果优化方向的典型研究有时空条件下的热力图生成扩展[9]、网络约束条件或多属性条件下的热力图生成方法[10]、结合核密度和空间自相关理论的热力探测优化方法等[11]。在效率提升方向,利用并行计算来加速计算过程是提升效率的有效手段[12-13],如文献[14]在Spark计算框架下利用多核并行方法大幅提升路网核密度估计的效率。文献[15]利用图形处理器(graphic processing unit,GPU)实现了核密度估计的并行计算,并分析了输入点数、带宽、网格大小等不同参数条件下的算法效率。文献[16]提出利用共享内存块与循环展开方法对原始核密度算法进一步优化。文献[17]使用自适应的核密度计算方法,将自适应的带宽阈值生成过程也纳入到GPU加速算法中。

由上述分析可知,现有的基于GPU的热力图生成方法大都是在轨迹点模型基础上进行核密度值的计算,对热力图从生成到可视化整个过程的研究较少。本文以轨迹线模型为基础,利用GPU通用并行计算来研究热力图生成与显示方法及其优化策略。

1 轨迹线模型下的热力图生成

1.1 轨迹线模型的引入

计算核密度值是热力图生成的基础。设T_r为邻域半径,n为距离小于或等于T_r的要素个数,K为核函数,则空间内的任一点p的核密度D_p的计算公式为:

收稿日期:2020-03-29

第一作者:朱峥嵘,硕士,工程师,主要从事高性能计算和图像处理与显示方面的研究。1365852721@qq.com

$$D_p=\frac{1}{T_r^2}\sum_{i=1}^{n}K\left(\frac{R(p,f_i)}{T_r}\right) \tag{1}$$

式中，f_i为第i个邻域内要素；$R(p,f_i)$为像素点与第i个要素间的距离。式(1)中的要素可以为轨迹点或者轨迹线，因而轨迹热力图生成方法分为基于轨迹点模型[1-2]和基于轨迹线模型[3-7]两种方法。轨迹点和轨迹线两种要素均可用来研究目标轨迹对于场模型中每一像素的影响，这种影响常用核密度来表征。但轨迹点模型和轨迹线模型对于目标轨迹这种研究对象的最小分解尺度的要求是不同的，轨迹点模型认为轨迹点作为最小分解单元对场模型中的像素形成影响，而轨迹线模型认为轨迹线是最小分解单元，轨迹线作为一个整体对场模型中的像素形成影响。这种目标分解尺度的不同造成核密度计算方法不同。点模型中像素的核密度值是其邻域内轨迹点影响的综合，邻域计算方法为：当轨迹点与像素点之间的欧氏距离满足邻域半径限制时，则判定该轨迹点在邻域内；线模型中像素的核密度值是其邻域内轨迹线影响的综合，邻域计算法为：当轨迹线到像素点的最短距离满足邻域半径限制时，则判定该轨迹线在邻域内。

基于轨迹点模型的热力图生成方法计算过程简单，但它没有区分轨迹点所属轨迹线的情况，生成的热力图效果易受轨迹点分布情况以及预设邻域半径的影响。当邻域半径设置偏小时，相邻轨迹点的热力影响没有重合区域，轨迹点间部分像素的核密度值为零，导致基于轨迹点模型生成的热力图出现不连续分布的颗粒状甚至热力断层现象，如图1(a)所示，其中红色线为轨迹线。图1(b)为基于轨迹线模型生成的热力图，轨迹线作为一个整体对一个像素的核密度值只产生一次影响，因此效果未受上述因素影响。

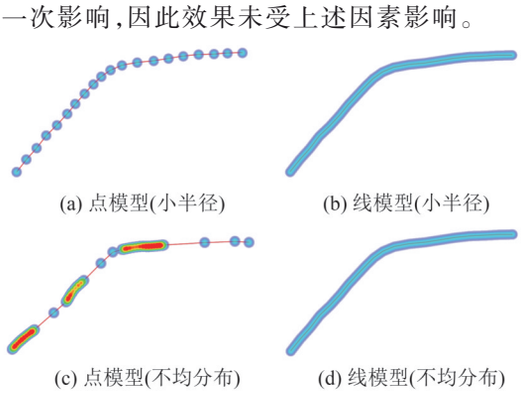

图1 两个模型的热力图效果

Fig.1 Heat Maps of Two Models

此外，在轨迹点模型中，由于每条轨迹线上可能存在多个轨迹点对同一像素产生大小不同的热力影响，导致当轨迹点疏密不均时，轨迹点模型生成的热力图在轨迹点密集处热力大、稀疏处热力小，如图1(c)所示。图1(d)为轨迹点疏密不均时轨迹线模型生成的热力图，轨迹线模型不考虑单个轨迹点的分布情况，而将每条轨迹线作为一个整体来计算线状要素的核密度，每条轨迹线对一个像素只产生一次热力影响，因此轨迹线不同位置的热力均匀分布。

综上，基于轨迹点模型的热力图生成方法有一定的局限性，需要在合适的邻域半径、理想的轨迹数据输入条件下才适用。而基于轨迹线的热力图生成方法在一定程度上摆脱了邻域半径和轨迹点分布情况的限制，采用该方法生成的热力图在轨迹线任意位置的邻域内都连续且平滑。因此，轨迹线模型是一种更通用的热力图计算模型。

1.2 热力图生成流程

基于§1.1提出的轨迹线模型的核密度定义，给出了基于中央处理器(central processing unit，CPU)串行方式的热力图生成流程(见图2)。

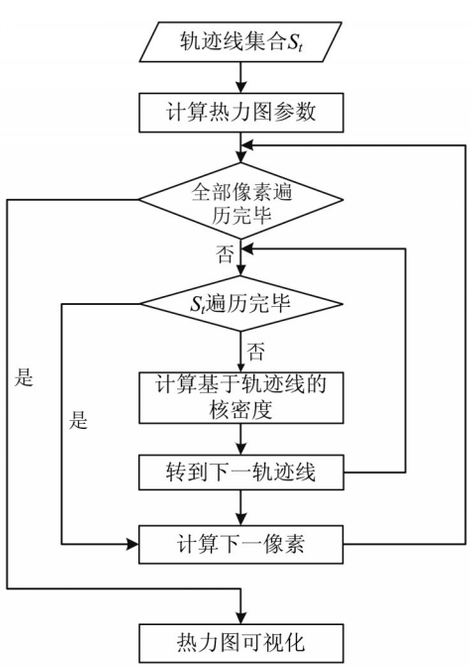

图2 基于CPU的热力图生成算法流程图

Fig.2 Flowchart of Heat Map Generation Based on CPU

对热力图中的每一像素点，执行图2中的计算过程，获得每一个像素点p的热力值。首先，以轨迹线集合S_t为输入，根据计算出的热力图参数将每个像素点转换至地理坐标系下，再计算出当前像素点到第i条轨迹线的各轨迹段的垂距$r_\perp$；然后，求出垂距集合中的最短垂距作为该像素点到第i条轨迹线的最短距离m_i，计算公式为：

$$r_j = M(R(p,p_{j-1}),R(p,p_j)) \tag{2}$$

$$r_\perp = \begin{cases} R(p,p_\perp), p_\perp \in e_j \\ r_j, p_\perp \notin e_j \end{cases} \tag{3}$$

$$m_i = M(S_{r_\perp}) \tag{4}$$

式中,r_j为像素点到第j条轨迹段的最短距离值;M为取最小值函数;p_j是轨迹线的第j个轨迹点;$R(p, p_j)$为像素点与第j个轨迹点间的距离;e_j为轨迹线中第j条轨迹段;$p_\perp$表示像素点到e_j的垂足点;$S_{r_\perp}$为像素点到各轨迹段的垂距集合。

最后,遍历每一条轨迹线,判断m_i是否满足邻域半径T_r的约束,对满足约束的轨迹线,根据式(1)计算其基于轨迹线的核密度D_p,其中:

$$R(p,f_i) = m_i \tag{5}$$

综合核密度分析结果,对输出的核密度矩阵进行可视化生成热力图,将核密度矩阵中的各个元素值进行归一化处理后的结果N_{xy}与色彩空间C_u建立下面的映射关系:

$$f(N_{xy}):N_{xy} \rightarrow C_u \tag{6}$$

式中,u为色彩空间分段;x和y为二维矩阵的索引;f为映射规则,即在每一色彩区间内根据线性变化规则计算出N_{xy}对应的各色彩分量结果。

2　基于GPU的热力图生成算法

传统的CPU串行方式生成一幅热力图需要按序计算每一个像素点的热力值,其算法的时间复杂度为$O(W \times H \times n)$,W和H是图像的宽和高,n是轨迹点总数。当图像分辨率较大或轨迹点数量较多时,传统CPU方法的计算复杂度高,热力图生成的效率低,难以适用于实时性要求较高的场景,本文提出了一种基于GPU的轨迹线热力图生成与显示算法,以获得更高的计算效率。

2.1　GPU模型

GPU模型包括硬件模型和编程模型,在硬件模型中,GPU的硬件结构分为主机端和设备端,主机端运行在中央处理器(central processing unit,CPU)上,负责资源分配和数据传输;设备端运行在GPU上,负责大规模数据的并行计算。GPU的处理核心是流多处理器(streaming multi-processor,SM),SM的核心组成元素流处理器(streaming processor,SP)是GPU最基本的处理单元。在GPU的软件编程模型中,线程是最小单元,每个线程都会执行在GPU中运行的核,完成数据的计算,多个线程组成一个线程块,多个线程块组成一个线程网格。线程束是SM调度和执行的基本单位,线程束中的所有线程以不同的数据资源执行相同的指令。

2.2　基于GPU的轨迹线热力图生成算法

基于GPU的算法设计需要协同CPU和GPU的资源共同完成计算流程,首先由运行在CPU上的主机端对轨迹数据和热力图参数数据进行读取、解析与传输,然后在GPU中完成热力图中每一像素点热力值的计算,最后将计算结果传输回主机端,对其进行可视化后生成热力图。图3是基于GPU的轨迹线热力图生成算法的流程图。

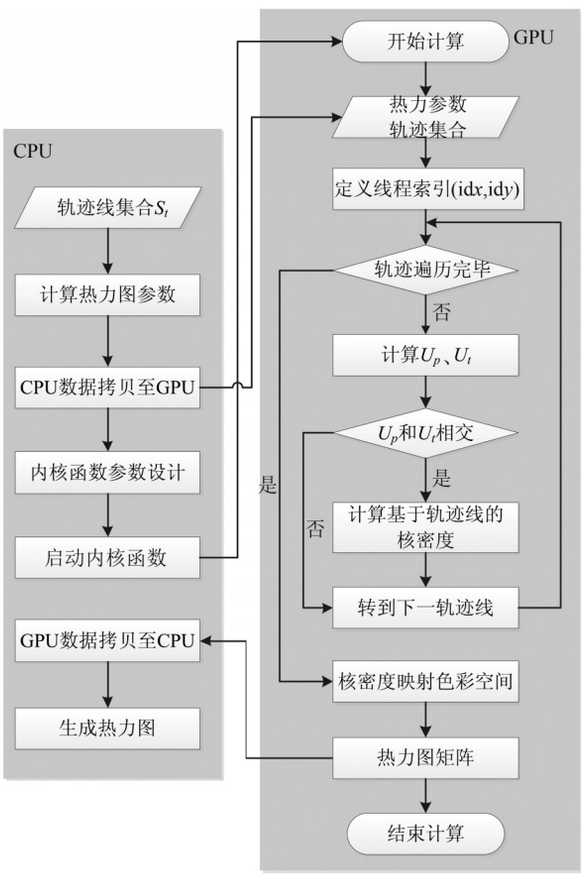

图3　基于GPU的轨迹线热力图生成算法流程图

Fig.3　Flowchart of Heat Map Generation Based on GPU

将主机端计算出的热力图参数和轨迹数据S_t从CPU内存拷贝至显存,启动核函数在GPU中进行热力值计算。

在核函数中,线程索引组合(id$x$,id$y$)与热力图的像素坐标相对应,每个线程计算一个像素的热力值。首先,根据热力图参数将像素坐标转换成相应地理坐标;然后,判断像素邻域U_p与当前轨迹线的外接矩形U_t是否相交,其中U_p为以像素点为中心、边长为邻域半径T_r的正方形。若二者不相交,则说明在T_r限制下,当前的轨迹线对该像素无热力影响,直接计算下一轨迹线;若相

交,则与CPU中方法一样,计算其基于轨迹线的核密度值。全部轨迹线计算完毕后,在内核函数中将核密度值线性映射到预设的色彩空间中。最后,执行完线程网格中的全部线程,计算出每一个像素的色彩值获得最终的热力图矩阵。内核函数执行结束后,将热力图矩阵数据传回主机内存,然后在CPU中进行可视化生成最终的热力图。

基于CPU的轨迹线热力图生成方法按顺序进行每个像素的热力值计算,而本文基于GPU的轨迹线热力图生成方法对每一个像素的热力值计算都由GPU上的一个线程完成,且核密度值与色彩空间的映射也由每一个线程分别完成,数个线程并发执行,同一时间能够完成多个像素点热力值的计算。

2.3 GPU算法优化分析

GPU资源利用率、统一计算设备架构(compute unified device architecture,CUDA)编程方式、内存设计等多方面因素均会影响到并行计算的加速效率,优化系统组织架构与计算流程是进一步提高GPU算法效率的有效手段。本节从以下几个方面来进行优化设计:

1)内核函数参数调优。GPU算法的优化设计首先应考虑提升活跃线程的占用率O_c,占用率即SM中实际活跃线程束的数量W_a与最大线程束数量W_m的比率:

$$O_c=\frac{W_a}{W_m} \tag{7}$$

当一个SM中全部的线程同时工作时,认为此SM占用率为100%,处理效率最高。内核函数参数设计是影响占用率高低的重要因素,其中内核函数的参数主要包含线程网格和线程块。

首先,线程束是SM调度和执行的最小单位,每个线程束包含32个线程,设计内核函数参数时应使每个线程块中的线程数量是32的整数倍,让每个线程块正好被分解成数个线程束的组合,避免不完整线程束中不活跃的线程占用SM资源。

然后,不同架构的英伟达(NVIDIA)显卡有不同的参数,设GPU中每个SM中最多可贮存k个线程块,每个SM最大可同时并发的线程束数量是g个,即同时可并发的线程束最多为$g/32$个。当线程块中线程数量小于g/k时,SM中线程块贮存量的限制会导致永远无法同时活跃$g/32$个线程束,无法实现100%占用率。因此,内核函数参数设计应考虑将线程块中的线程数量设置成g/k以上的整数。

最后,应使得线程块中线程的数量是SM最大并发数量g的约数,否则会导致SM中实际贮存的线程数量少于g个,导致永远无法达到100%的占用率。

2)循环展开策略。循环展开是一种指令级的优化方法,是通过牺牲程序的尺寸来加快执行速度的优化方法。在GPU中进行循环展开有两种策略,其一是展开内核函数内部的循环过程,即一次循环进行多组数据的计算,减少总的循环判断次数;其二是GPU级循环展开,即设置更少数量的线程块,而让每个线程做多个数据的操作。在本文基于GPU的轨迹线热力图生成方法中应用上述两种循环展开策略。

(1)展开内核函数内部的循环体,遍历轨迹线的全部轨迹点计算像素到轨迹线的最短垂距,为避免末尾轨迹点丢失计算,设计循环展开过程为:

$$B=(N_p-1)\%E$$
$$E=\begin{cases}0, & 0\leqslant i<B\\ E, & B\leqslant i<(N_p-1)\end{cases} \tag{8}$$

式中,N_p表示轨迹点数量;E表示展开率;B为循环展开边界,表示开头几个轨迹点采用0展开,以让最后一次展开循环恰好计算到最后一个轨迹点。

(2)对于GPU级循环展开,先设置线程块数量为原来的$1/E$,再让每个线程处理相邻的E个像素点,一个循环中计算E个像素点的核密度,减少指令消耗,且轨迹线等中间过程数据能进行重用。

3)基于像素缓冲对象(pixel buffer object,PBO)的显示优化方法。GPU中计算出的数据需要拷贝回CPU,并在CPU中进行渲染和绘制才能显示出最终的热力图,此过程中涉及的数据传输、图像显示等占据的时间消耗使得整个算法效率降低。针对此问题,本节给出一种基于PBO的热力图生成与显示优化方法。OpenGL在GPU上创建并管理的缓冲内存区域称为缓冲对象,内核函数将一段缓冲映射到GPU内存空间,使得在内核函数中修改像素值可直接映射到PBO中,不需要进行主机与设备间的内存拷贝,因此这种映射是一种处理速度很快的低开销操作。图4是该优化算法的流程图。

在图4所示的算法中:(1)初始化OpenGL,创建图像显示窗口;(2)在GPU内存中开辟缓冲

区,创建像素缓冲区对象PBO,并对PBO进行注册,之后建立PBO与GPU中内存的映射,即在PBO与CUDA内核函数中输出的图像热力矩阵之间创建映射关系;(3)启动内核函数在GPU中进行热力分析,采用图3中计算热力图矩阵的方法,将计算出的每一个像素点的热力值都写入到显存中,同时即映射到了PBO缓存中;(4)在GPU计算完毕之后,OpenGL渲染出热力图并显示到预先创建的显示窗口中;(5)解除PBO注册并删除PBO。

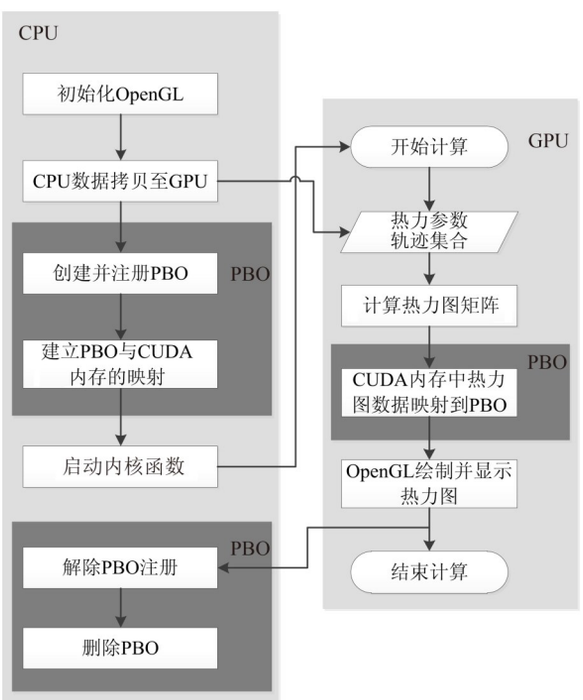

图4　PBO优化方法流程图

Fig.4　Flowchart of PBO Optimization

3　效率对比实验与讨论

为验证本文方法的有效性,设计了两个实验:CPU与GPU热力图生成效率对比实验、GPU算法优化前后效率对比实验。

3.1　CPU与GPU热力图生成效率对比

本文实验所用的显卡是NVIDIA的GeForce GT 730,计算能力是2.1,该型号显卡共4 GB显存,包含2个SM,共96个计算核心,每个SM最大可并发的线程数量是1 536个,每个SM中最多可贮存8个线程块。采用VS 2010和CUDA 7.5为开发环境,设计4 000×4 000像素、2 000×2 000像素、1 000×1 000像素这3种热力图分辨率,使用t_1~t_6共6组不同的轨迹数据分别测试CPU与GPU的热力图生成和显示流程的执行时间。其中,从t_1到t_6轨迹数量逐步增加,实验结果如表1~3所示。

表1　4 000×4 000像素分辨率下CPU和GPU时耗

Tab.1　Time Consumption of CPU and GPU at 4 000×4 000 Pixels Resolution

算法	轨迹数据					
	t_1/s	t_2/s	t_3/s	t_4/s	t_5/s	t_6/s
CPU	5.48	17.77	33.96	60.47	101.10	111.00
GPU	0.45	0.73	1.22	1.69	2.40	2.62
加速比	12.2	24.3	27.8	35.8	42.1	42.4

表2　2 000×2 000像素分辨率下CPU和GPU时耗

Tab.2　Time Consumption of CPU and GPU at 2 000×2 000 Pixels Resolution

算法	轨迹数据					
	t_1/s	t_2/s	t_3/s	t_4/s	t_5/s	t_6/s
CPU	1.53	4.65	8.84	15.18	25.49	28.30
GPU	0.28	0.40	0.65	0.67	0.82	0.91
加速比	5.5	11.6	13.6	22.7	31.1	31.1

表3　1 000×1 000像素分辨率下CPU和GPU时耗

Tab.3　Time Consumption of CPU and GPU at 1 000×1 000 Pixels Resolution

算法	轨迹数据					
	t_1/s	t_2/s	t_3/s	t_4/s	t_5/s	t_6/s
CPU	0.52	1.33	2.85	3.86	6.37	6.91
GPU	0.23	0.29	0.50	0.40	0.43	0.48
加速比	2.3	4.6	5.7	9.7	14.8	14.4

从表1~3中可以看出,对于4 000×4 000像素的热力图分辨率,输入t_6轨迹数据时,基于CPU的轨迹线热力图生成与显示算法共耗时111.00 s,效率非常低,而基于GPU的轨迹线热力图生成与显示算法只用了2.62 s,速度提升了42.4倍多,算法执行效率实现跃升。综合全部的实验结果来看,对于6组不同的轨迹数据,4 000×4 000像素、2 000×2 000像素、1 000×1 000像素这3种热力图分辨率下,GPU方法的速度平均比CPU方法分别提升了30.8、19.3、8.6倍。由此证明,基于GPU的轨迹线热力图生成与显示算法有效改善了CPU方法的效率问题。

从表1~3中还可以看出,相同轨迹数据下,当热力图分辨率增大时,GPU算法相比于CPU算法的加速比呈现逐步增大的趋势。这是因为当轨迹数据量或热力图分辨率较大时,内核函数的计算复杂度相对较高,使得GPU并行加速部分的时耗占整个热力图生成和显示总时耗的比例更高,因此加速比会逐步增大。

由于GPU硬件资源有限,同一时间能够同时

执行的线程数量有限，不可能实现分辨率数量级的加速比。进一步实现性能提升最直接的方法就是提高显卡的品质，品质越高的显卡包含的计算核心的数量也越多，意味着更高的计算性能。目前，NVIDIA的GeForce系列品质较高的显卡GeForce GTX 2080ti已经包含4 352个计算核心，是本文使用显卡的计算核心数量的45倍多，如此多数量的流处理器若应用到本文的热力图生成算法中，会获得更加惊人的速度提升。

3.2 GPU优化前后算法效率对比

§2.3描述了从内核函数参数调优、循环展开、PBO显示几个方面来进一步优化基于GPU的轨迹线热力图生成与显示算法。下文给出了每种方法优化前后的实验结果对比。

1）内核函数参数调优。根据§2.3讨论的内核函数参数调优方法和GeForce GT 730显卡的各项参数，可知每个SM最大可同时并发的线程束数量为48个，为了能达到100%占用率至少应设置每个线程块中的线程数量为192，因此本文设计4×8、13×15、12×16、16×16、16×32、25×32 6种线程块尺寸（线程块尺寸表示每个二维线程块中x维度和y维度线程的数量）来验证不同内核函数参数设计对算法性能的影响。其中4×8、13×15、25×32分别用来验证线程块尺寸不足192、线程块尺寸不是32的整数倍、线程块尺寸不是最大并发线程数的约数时的计算效率，其他为满足各项约束条件时的内核函数参数。在4 000×4 000像素分辨率条件下，使用6组不同的轨迹数据集对不同的线程块尺寸下的热力图矩阵的计算效率进行测试，实验结果如图5、图6所示。

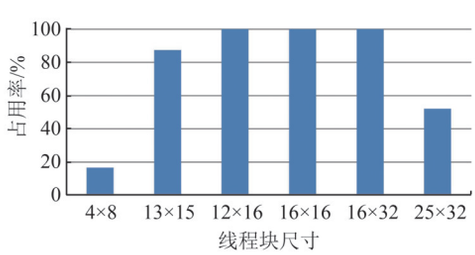

图5 内核函数参数对占用率的影响

Fig.5 Influence of Kernel Function Parameters on Occupancy

从图5、图6中可以看出，4×8、13×15、25×32等不满足约束条件的线程块尺寸会导致占用率达不到100%，相比其他几种线程块尺寸，算法时耗明显升高，尤其是当线程块尺寸小于192时，占用率的明显下降导致算法时耗显著增加。与之相比，线程块尺寸为12×16、16×16、16×32时，占用率为100%，算法时耗相对较短，效率比13×15和25×32平均提升了约12%，比4×8平均提升了约48%。

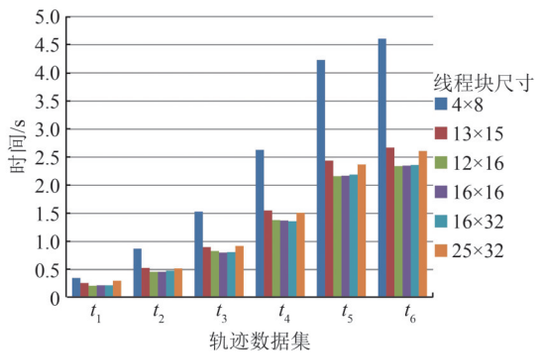

图6 内核函数参数调优效果

Fig.6 Tuning Effect of Kernel Function Parameter

2）循环展开。在4 000×4 000像素分辨率、线程块尺寸为16×16不变的条件下，采用t_1～t_6共6组轨迹数据为输入，设计0展开、2展开、4展开、8展开、16展开5种循环展开度来测试GPU中热力图矩阵的计算时间，进而观察循环展开的效果。实验结果如图7所示。

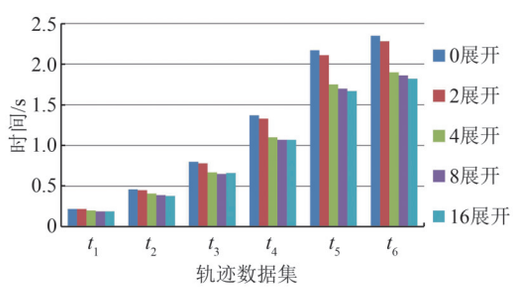

图7 循环展开效果

Fig.7 Effect of Loop Unrolling

循环展开优化程度与循环体中计算复杂程度、代码重用度等因素有关，通常情况下，循环展开度越高，速度提升越明显，完全展开时能够获得最大程度的优化效果。但是循环展开程度越高，则程序体积越大，且可能消耗的寄存器也会大幅增加，导致占用率下降，反而可能会降低计算效率。从图7中可以看出，对循环进行4展开和8展开可以获得较好的性能提升，平均比循环展开前提升约20%。再往上增大展开度对性能提升作用很小，用牺牲大量程序空间的代价换取时间的性价比降低，因此建议展开循环时视实际情况而定，一般进行4展开或8展开为优。

3）基于PBO的热力图显示优化方法。同样采用上述t_1～t_6轨迹数据，设计2 000×2 000像素和4 000×4 000像素热力图分辨率，保持线程块

尺寸为16×16不变,不对代码进行循环展开,分别测试引入PBO和不引入PBO时整个热力图生成与显示流程的时间消耗。实验结果如图8、图9所示。

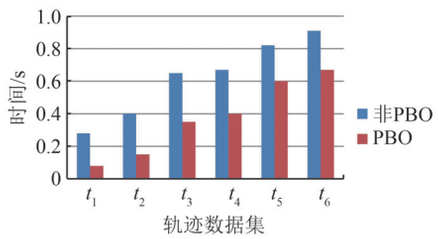

图8　2 000×2 000像素分辨率下PBO优化效果

Fig.8　Effect of PBO at 2 000×2 000 Pixels Resolution

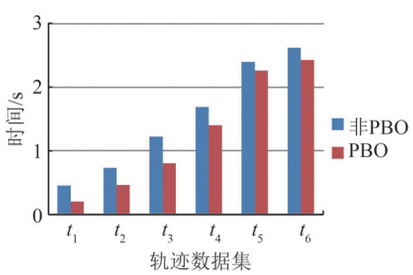

图9　4 000×4 000像素分辨率下PBO优化效果

Fig.9　Effect of PBO at 4 000×4 000 Pixels Resolution

从图8、图9中可以看出,输入6组不同的轨迹数据时,对于2 000×2 000像素和4 000×4 000像素热力图分辨率,使用基于PBO的热力图生成显示优化方法相比于不引入PBO的方法分别平均提升了约26%和38%的效率。

4　结　语

大轨迹数据量、高分辨率条件下热力图生成显示效率一直是困扰热力图推向实际工程应用的关键所在。针对该问题,本文给出了一种利用GPU加速的轨迹线热力图生成显示方法。该方法协同CPU、GPU资源,在CPU上完成轨迹数据的读取、解析与传输,在GPU上完成像素点热力值的计算和色彩映射。该方法具有如下特点:(1)采用轨迹线模型生成热力图,有效避免了因轨迹点分布不均或邻域半径设置不合理而导致的热力断层、密度不均等情况的发生;(2)利用GPU并行计算、内核函数参数调优、循环展开等多种方法优化算法效率,达到性能备增的目的;(3)生成与显示一体,GPU不仅进行核密度值的计算,而且把色彩映射也进行并行加速,并通过PBO方式将显存内容直接映射至主机内存进行显示。

未来研究方向包括:(1)GPU并行计算结合轨迹线空间索引来进一步提高热力图生成的计算效率。(2)将GPU并行计算推广到如预警探测等具有海量数据计算场景的领域。

参 考 文 献

[1] Meng Xiaofeng, Ding Zhiming, Xu Jiajie. Moving Objects Management, Models, Techniques and Applications[M]. Beijing: Tsinghua University Press, 2014

[2] Ralf H G, Markus S. Moving Objects Databases [M]. Beijing: Higher Education Press, 2009

[3] Ying J J C, Lee W C, Weng T C, et al. Semantic Trajectory Mining for Location Prediction[C]//The 19th ACM SIGSpatial International Conference on Advances in Geographic Information Systems Chicago, Illinois,USA, 2011

[4] Ribes D, Bowker G C. A Learning Trajectory for Ontology Building[J]. *Journal of the Association for Information Systems*, 2005,6(1): 1-10

[5] Dong X, Pi D C. Novel Method for Hurricane Trajectory Prediction Based on Data Mining[J]. *Natural Hazards and Earth System Sciences*, 2013, 13(12): 3211-3220

[6] Jeung H, Shen H T, Zhou X F. Convoy Queries in Spatio-Temporal Databases[C]//The 24th International Conference on Data Engineering, Cancun, Mexico, 2008

[7] Scheepens R, van de Wetering H, van Wijk J J. Contour Based Visualization of Vessel Movement Predictions[J]. *International Journal of Geographical Information Science*, 2014, 28(5): 891-909

[8] Andrienko G, Andrienko N, Wrobel S. Visual Analytics Tools for Analysis of Movement Data[J]. *ACM SIGKDD Explorations Newsletter*, 2007, 9(2): 38-46

[9] Demšar U, Virrantaus K. Space-Time Density of Trajectories: Exploring Spatio-Temporal Patterns in Movement Data[J]. *International Journal of Geographical Information Science*, 2010, 24(10): 1527-1542

[10] Dong Haoyang, Zhang Dongge, Wan Yiping, et al. Research on Construction Method of Heat Map for Battlefield Situation [J]. *Command Control and Simulation*, 2017, 39(5): 1-8 (董浩洋, 张东戈, 万贻平, 等. 战场态势热力图构建方法研究[J]. 指挥控制与仿真, 2017, 39(5): 1-8)

[11] Yu Wenhao, Ai Tinghua, Yang Min, et al. Detecting "Hot Spots" of Facility POIs Based on Kernel Density Estimation and Spatial Autocorrelation Technique

[J]. *Geomatics and Information Science of Wuhan University*, 2016, 41(2): 221-227 (禹文豪, 艾廷华, 杨敏, 等. 利用核密度与空间自相关进行城市设施兴趣点分布热点探测[J]. 武汉大学学报·信息科学版, 2016, 41(2): 221-227)

[12] NVIDIA. CUDA Compute Unified Device Architecture: Programming Guide V5.0 [R]. Santa Clara: NVIDIA Corporation, 2012

[13] Fang Liuyang, Wang Mi, Pan Jun. CPU/GPU Cooperative Fast Band Registration Method for Multispectral Imagery [J]. *Geomatics and Information Science of Wuhan University*, 2018, 43(7): 1000-1007 (方留杨, 王密, 潘俊. CPU和GPU协同的多光谱影像快速波段配准方法[J]. 武汉大学学报·信息科学版, 2018, 43(7): 1000-1007)

[14] Guo Yuda, Zhu Xinyan, Guo Wei, et al. Parallel Algorithm for Road Network Kernel Density Estimation Based on Spark Computing Framework [J]. *Geomatics and Information Science of Wuhan University*, 2020, 45(2): 289-295 (郭宇达, 朱欣焰, 禹维, 等. 基于Spark计算框架的路网核密度估计并行算法[J]. 武汉大学学报·信息科学版, 2020, 45(2): 289-295)

[15] Kamara A. Improving Kernel Density Estimation Calculation Time by Parallel Processing [EB/OL]. (2014-10-18) [2019-12-21]. http://faculty.salisbury.edu/~ealu/REU/REU_2014html

[16] Michailidis P D, Margaritis K G. Accelerating Kernel Density Estimation on the GPU Using the CUDA Framework [J]. *Applied Mathematical Sciences*, 2013, 30(7): 1447-1476

[17] Zhang G M, Zhu A, Huang Q Y. A GPU-Accelerated Adaptive Kernel Density Estimation Approach for Efficient Point Pattern Analysis on Spatial Big Data [J]. *International Journal of Geographical Information Science*, 2017, 31(10): 2068-2097

A Method of Generating and Displaying Trajectory Line Heat Map with GPU Acceleration

ZHU Zhengrong[1] *HUANG Yafeng*[1] *ZHAO Liying*[1]

1 Nanjing Research Institute of Electronics Technology, Nanjing 210039, China

Abstract: Objectives: To deal with the high computation expense in data-intensive heat map generation, parallel computing technology based on graphic processing unit(GPU) is introduced to the heat map generation filed. With combination of trajectory line model, we propose a novel method for trajectory line heat map generation and display based on GPU. **Methods**: Firstly, in order to avoid discontinuity and inhomogeneity of heat value resulted from different dense of trajectory points or unreasonable neighborhood threshold, trajectory line model is adopted to improve the effect of generated heat map. Secondly, aiming at the problem of low efficiency of heat map generation caused by large-scale data calculation, the proposed method can improve the efficiency greatly via a combination of GPU parallel computing, tuning of kernel function parameter, loop unrolling and pixel buffer object. **Results**: Experimental results show that processing speed of the proposed method is from 5 to 30 times faster than CPU(central processing unit))-based implementation. **Conclusions**: Additionally, the accelerator ratio tends to be higher as the resolution of heat map or the increase of trajectory data.

Key words: heat map; GPU parallel computing; trajectory line; loop unrolling; PBO display

First author: ZHU Zhengrong, master, engineer, specializes in high performance computing and image processing and display. E-mail: 1365852721@qq.com

引文格式: ZHU Zhengrong, HUANG Yafeng, ZHAO Liying. A Method of Generating and Displaying Trajectory Line Heat Map with GPU Acceleration[J]. Geomatics and Information Science of Wuhan University, 2022, 47(7): 1035-1042. DOI: 10.13203/j.whugis20200001 (朱峥嵘, 黄亚锋, 赵立营. 一种利用GPU加速的轨迹线热力图生成显示方法[J]. 武汉大学学报·信息科学版, 2022, 47(7): 1035-1042. DOI: 10.13203/j.whugis20200001)