文章编号：0253-374X(2013)11-1732-06 DOI:10.3969/j.issn.0253-374x.2013.11.021

面向CPU+GPU异构计算的SIFT特征匹配并行算法

肖 汉[1,2]，郭运宏[3]，周清雷[1]

(1.郑州大学 信息工程学院，河南 郑州 450001；2.郑州师范学院 信息科学与技术学院，河南 郑州 450044；
3.郑州铁路职业技术学院 电气工程系，河南 郑州 450052)

摘要：依据图形处理器(GPU)计算特点和任务划分的特点，提出主从模型的CPU+GPU异构计算的处理模式.通过分析和定义问题中的并行化数据结构，描述计算任务到统一计算设备架构(CUDA)的映射机制，把问题或算法划分成多个子任务，并对划分的子任务给出合理的调度算法.结果表明，在GeForce GTX 285上实现的尺度不变特征变换(SIFT)并行算法相比CPU上的串行算法速度提升了近30倍.

关键词：遥感影像；特征匹配；图形处理器(GPU)；统一计算设备架构(CUDA)；尺度不变特征变换(SIFT)

中图分类号：P231；TP391 **文献标志码**：A

Parallel Algorithm of CPU and GPU-oriented Heterogeneous Computation in SIFT Feature Matching

XIAO Han[1,2], *GUO Yunhong*[3], *ZHOU Qinglei*[1]

(1. School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China; 2. School of Information Science and Technology, Zhengzhou Normal University, Zhengzhou 450044, China; 3. College of Electrical Engineering, Zhengzhou Railway Vocational & Technical College, Zhengzhou 450052, China)

Abstract: According to the basis of features about graphic processing unit (GPU) computation and tasks division, the study tries to bring forward a method of Master/Slave CPU+GPU heterogeneous computation. This paper presents an analysis and definition of the parallel data structures, and a description of the mapping mechanism for computing tasks on compute unified device architecture (CUDA). A logical scheduling algorithm is proposed to divide an issue or algorithm into many sub- tasks. The result shows that the speed of SIFT parallel algorithm in the Geforce GTX 285 is about 30-time of the serial algorithm running in the CPU.

Key words: remote sensing image; feature matching; graphic processing unit (GPU); compute unified device architecture (CUDA); scale invariant feature transform (SIFT)

随着摄影测量和计算机视觉的发展，影像特征匹配越发成为遥感资源分析、三维重建和模式识别等领域研究的基本问题.传统的特征点检测算法中，基于模板匹配的特征点检测算法不易设计出大量模板来匹配纹理细腻的特征点；基于边缘检测的特征点检测算法精度不高；基于亮度变化的特征点检测算法受噪声、光照的影响很大.尺度不变特征变换(scale invariant feature transform，SIFT)算法[1]是一种基于尺度空间理论提取影像局部特征的有效算法，SIFT特征对尺度缩放、旋转、光照变化均保持不变.但是其算法复杂度高、计算时间长，在实时性要求较高的应用场合中受到了限制.

Sinha等[2]提出了基于OpenGL的SIFT特征点提取算法，提速10倍左右.Warn等[3]提出了一种运行在混合机群上的SIFT算法SOHC.袁修国等[4]提出了利用统一计算设备架构(compute unified device architecture，CUDA)，在一种变型的SIFT算法GLOH上进行并行化，但是GLOH算法训练出来的投影矩阵只对某类图像起作用，缺乏广泛的适用性.高玉鹏等[5]实现了CUDA-SIFT算法提取影像匹配特征点.Mikolajczyk等[6]提出一种GLOH的改进算法，但其匹配效率没有明显提高.田文等[7]提出一种基于CUDA的尺度不变特征变换快速算法，测试性能达到了CPU实现的30～50倍.但是从

收稿日期：2012-10-27
基金项目：国家自然科学基金(41171357)；国家"九七三"重点基础研究发展计划(2012CB719900)；中国博士后科学基金(2012M510110)；河南省重点科技攻关项目(132102310003)；河南省教育厅科学技术研究重点项目(13A520354)
第一作者：肖 汉(1970—)，男，教授，工学博士，主要研究方向为遥感影像处理、高效能计算. E-mail:xiaohan70@163.com
通讯作者：周清雷(1962—)，男，教授，博士生导师，工学博士，主要研究方向为并行算法、软件工程. E-mail:ieqlzhou@zzu.edu.cn

实验数据中无法看出 SIFT 算法的执行时间中是否包括了数据传输时间这一关键因素.

综上所述，采用以 CPU 为核心的多核机、并行集群的传统计算资源和传统的 OpenGL 图形 API 开发模式等方式来换取效率，均难以满足在保证影像匹配质量的同时人们对系统性能提升的要求. 与此同时，图形处理器(GPU)的可编程能力和并行处理能力越来越强大. 为此，本文提出一种对 SIFT 特征匹配算法进行多级并行优化的并行算法. 实验结果表明，在考虑了 CPU-GPU 间数据传输时间的情况下，SIFT 特征匹配 GPU 并行算法较 CPU 串行算法速度提高了近 30 倍，大幅缩短数据处理的时间，实时性有了很大的提高.

1　GPU 及 CUDA 架构

NVIDIA 公司推出的 CUDA 模型是一个全新的软硬件基础架构，它以更便捷的方式将 GPU 的计算能力开放出来.

1.1　Tesla 2 GPU 并行计算构架

Tesla 2 架构中的所有可编程的计算任务都由流处理器阵列(streaming processor array，SPA)执行. SPA 的结构可以被分为两层：第一层是由若干个线程处理器集群(thread processor cluster，TPC)组成的整个阵列，以构成核心计算框架；第二层是组成 TPC 的若干个流多处理器、一条纹理流水线和一个与渲染输出单元通信的端口.

GPU 执行通用并行计算处理任务时的工作流程：首先 GPU 通过 PCI-Express 总线与主板北桥芯片组通信，从 CPU 及内存中读入工作负载到输入装配单元，然后由计算分配单元将输入装配单元取出的计算线程组成的协作线程阵列分发到 SPA 中的各个 TPC 中执行.

1.2　CUDA 编程模型

CUDA 采用串-并行混杂的编程模式. 任务组织和发送是在 CPU 中完成，每当 CPU 遇到需要并行计算的任务，则将要做的运算组织成内核 kernel. 将内存中需要计算的数据从内存由北桥经过高速的 PCI-Express 总线复制到显存中，再由 GPU 执行设备端程序，在计算结束后由主机端程序将结果数据从显存复制到内存. CUDA 可以通过 kernel 创建出成千上万个线程，然后将这些线程送到 GPU 中的上百个核上去运行.

2　SIFT 匹配算子描述

2.1　建立多尺度空间

一幅二维影像，在不同尺度下的尺度空间表示可由影像与高斯核空间域卷积得到，如式(1)所示.

$$\boldsymbol{L}(x,y,\sigma)=\boldsymbol{G}(x,y,\sigma)\otimes\boldsymbol{I}(x,y) \tag{1}$$

式中：$\boldsymbol{G}(x,y,\sigma)$代表尺度可变的二维高斯函数；$\otimes$代表卷积运算；$\boldsymbol{I}(x,y)$代表输入影像；$\boldsymbol{L}$ 代表影像的尺度空间. 为了有效地在尺度空间检测到稳定的关键点，Lowe 提出了利用高斯差分函数(difference of Gaussian，DOG)对原始影像进行卷积，如式(2)[1]所示.

$$\begin{aligned}\boldsymbol{D}(x,y,\sigma)=&\\ &[\boldsymbol{G}(x,y,k\sigma)-\boldsymbol{G}(x,y,\sigma)]\otimes\boldsymbol{I}(x,y)=\\ &\boldsymbol{L}(x,y,k\sigma)-\boldsymbol{L}(x,y,\sigma)\end{aligned} \tag{2}$$

式中：k 为不同的高斯核尺度，k 的初值为 1，尺度以 k 倍递增. 图 1 为生成高斯差分尺度空间的示意图.

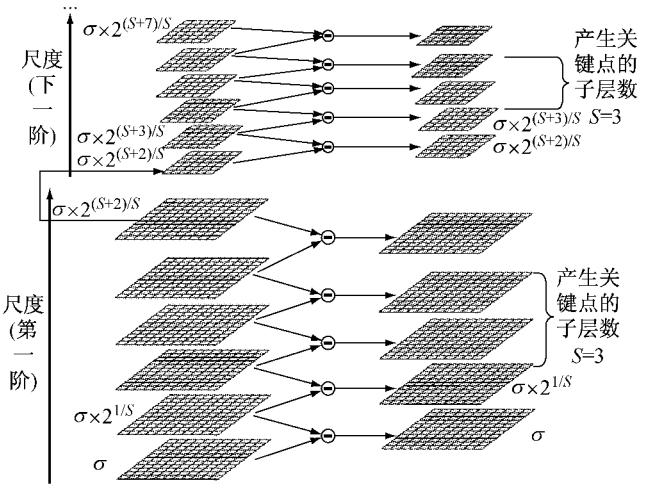


图 1　高斯差分尺度空间的生成

Fig.1　Generation of scale-space in the difference of Gaussian

2.2　尺度空间关键点检测及精确定位

在高斯差分金字塔影像中间 S 层中，将影像中的每个点和它同尺度的八邻域点以及上下相邻尺度对应的 9×2 个邻域点比较是否为极值点，如为极值点则记录其所处的位置和对应的尺度，该点即为候选特征点. 检测到的所有极值点就是 SIFT 候选特征点的集合[8].

由于 DOG 算子会产生较强的边缘响应，还需要舍弃低对比度的特征点和不稳定的边缘响应点，经过进一步的检验才能精确定位为关键点，以增强匹配的稳定性，提高抗噪声能力[9].

2.3 关键点方向参数的确定

利用关键点邻域像素的梯度方向分布特征为每个关键点指定方向参数,如式(3)和(4)所示.

$$m(x,y)=\{[L(x+1,y)-L(x-1,y)]^2+[L(x,y+1)-L(x,y-1)]^2\}^{\frac{1}{2}} \quad (3)$$

$$\theta(x,y)=\arctan\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)} \quad (4)$$

式中:$m(x,y)$和$\theta(x,y)$分别为高斯金字塔影像(x,y)处梯度的模值和方向,L所用的尺度为每个关键点各自所在的尺度.

2.4 提取特征描述符

SIFT特征匹配算法关键点由4×4共16个种子点组成,每个种子点有八个方向的向量信息.将每个种子点的八方向向量信息依次排序,对于每个关键点就能够产生128个数据,形成4×4×8共1×128维的特征向量即SIFT特征描述符,记特征描述向量为$\boldsymbol{N}=[n_1,n_2,\cdots,n_{128}]^{\mathrm{T}}$.此时的SIFT特征描述符已经去除了尺度变化、旋转、变形因素和抗噪声的影响[10].最后将特征向量长度进行归一化,可以进一步增强对光照变化的鲁棒性[11].

2.5 SIFT特征匹配

当两幅影像的SIFT特征描述符生成以后,采用关键点特征描述符的欧氏距离作为两幅影像中关键点的相似性判定度量.

3 SIFT特征匹配并行算法分析与设计

3.1 SIFT特征匹配并行计算模型

在SIFT特征匹配算法中进行的高斯金字塔影像的构建过程中,每一子层的建立是相对独立的.在尺度空间极值点检测过程中,子层与子层之间并无任何数据通信.特征匹配时在寻找左影像中某个关键点的匹配点的遍历计算中也是相互独立完成,数据之间并无依赖关系.据此,SIFT特征匹配GPU并行化处理执行模式如图2所示.

SIFT特征匹配并行计算具体过程为:① 首先利用CPU读取输入影像到主存,将不同尺度的高斯核数据由主存传入到GPU的常数存储器中.在GPU中进行分步连续滤波加速高斯尺度空间金字塔的构建,并将高斯金字塔存储在全局存储器内.然后高斯金字塔影像被回读到CPU,相邻尺度的两个高斯影像相减得到DOG金字塔多尺度空间表示.随后在CPU中检测高斯差分尺度空间中的局部极值点(最大值或最小值),并记录其所处的位置和对应

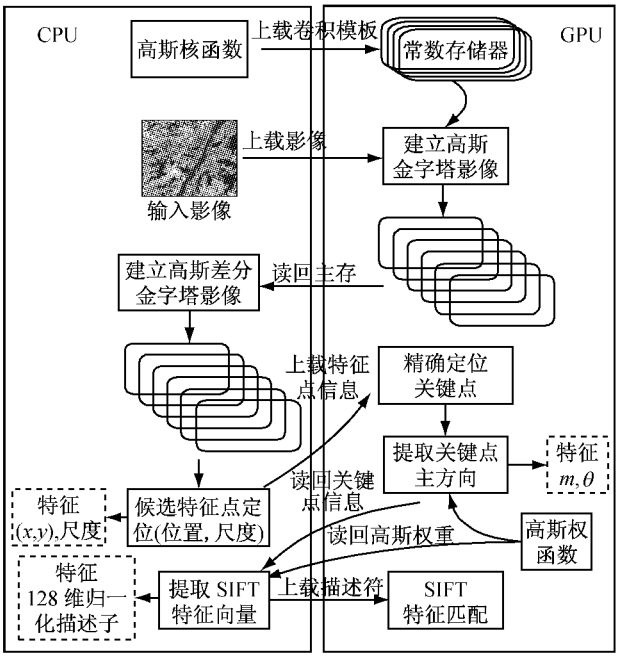


图2 SIFT特征匹配并行算法执行模式

Fig.2 Execution mode of parallel algorithm for SIFT feature matching

的尺度.这是因为在GPU上建立多尺度的DOG空间并确定局部极值点所花费的时间超过通过一个小的回读把数据传递到CPU上进行计算的时间.当把候选特征点集合信息上传至GPU中后,便可在GPU内对高斯差分金字塔所有候选特征点进行精确定位.计算关键点周围影像强度的主曲率,通过一个2×2的Hessian矩阵计算特征值比率,检测关键点主曲率是否超过设定的阈值,通过去掉多余的点后,确定关键点集合并精确标记关键点的位置、尺度.关键点位置、尺度将在GPU中恢复.② 利用在GPU中计算的关键点集合信息,启动kernel计算在关键点附近像素的梯度大小和方向.利用已经存储在全局存储器中的高斯权重函数,对关键点邻域窗口内的各像素的梯度大小进行高斯加权并累加建立方向直方图,检测直方图的峰值,确定关键点主方向.③ 计算128维的SIFT描述符.以关键点为中心的16×16影像数据块根据关键点的尺度、位置和方向构造SIFT描述符的过程,在CPU上实现则可以发挥CPU逻辑分支、判断能力强的优势,高效地完成任务.④ SIFT特征匹配,确定匹配点位.按照原始点的自然顺序将维度数据读入共享存储器,优化的重点是距离计算.首先必须保证每个维度差值的平方在同一时刻被计算,而不是使用内循环方式.其次必须保证维度数目中间结果的累加方式高效.

基于GPU的SIFT特征匹配算法通过并行性

分析,将许多计算分割在 CPU 和 GPU 之间分别计算,发挥了各自的计算优势,充分体现了 CPU+GPU 异构计算的强大能力.

3.2　基于 CUDA 的并行化数据结构

3.2.1　卷积并行

(1) 卷积层特征映射

多阶高斯金字塔影像特征映射集可描述为一个由两个二维数组构成的数据结构,即

$$L_{ij}=\{I_{ij}[M][N],G_{ij}[K][K]\} \tag{5}$$

式中:M,N 分别为影像大小的行列数;K 为高斯卷积窗口大小;$i=0,1,2,\cdots,O-1$;$j=0,1,2,\cdots,5$. L_{ij} 表示 O 阶六层高斯金字塔影像特征映射集,为一二维数组.

以层为单位,映射到 CUDA 上后,构成一个一维的数组集,如式(6)所示.

$$\begin{aligned}\_L_i=&\{\_I_{ij}[M\times N],\_G_{ij}[K\times K]\}=\\&\{\{\_I_{i0}[M\times N],\_G_{i0}[K\times K]\},\\&\{\_I_{i1}[M\times N],\_G_{i1}[K\times K]\},\\&\{\_I_{i2}[M\times N],\_G_{i2}[K\times K]\},\\&\{\_I_{i3}[M\times N],\_G_{i3}[K\times K]\},\\&\{\_I_{i4}[M\times N],\_G_{i4}[K\times K]\},\\&\{\_I_{i5}[M\times N],\_G_{i5}[K\times K]\}\}\end{aligned} \tag{6}$$

式中:$\_L_i$ 表示第 i 阶高斯金字塔特征映射集,且为一维数组. $\{\_I_{ij}[M\times N],\ \_G_{ij}[K\times K]\}$ 为第 i 阶第 j 层特征映射集,每层特征映射共享一个 kernel,同时接收来自上层特征映射结果. 所有 $\_L_i$ 特征映射集组成 L_{ij}.

在主机端上将影像数据读入内存并传输到设备内存中,根据问题的规模和处理方式来确定需要的网格的个数. 在 GPU 中进行高斯金字塔影像的计算,需要为计算配置六个处理内核,用于对要处理的问题进行划分. 每一个 kernel 配置项包括问题的分块数 GridDim 以及每个块内线程数 BlockDim. 对于一幅 row × col 的影像,高斯滤波窗口选取为 GAUSSIANDIM × GAUSSIANDIM,Grid 栅格大小和 kernel 启动的执行配置如下:

```
xblock=row-GAUSSIANDIM+1;
yblock=col-GAUSSIANDIM+1;
blocks(xblock,yblock);
threads(GAUSSIANDIM×GAUSSIANDIM);
gaussConvolution<<<blocks, threads>>>
(LeftImage_D,RightImage_D,col);
```

(2) 棋盘式卷积

卷积过程是一个顺序运算的过程,主要包括两步,分别是系数的卷积乘计算和积的累加计算. 棋盘式并行计算卷积乘原理如图 3 所示,其中 T 为线程.

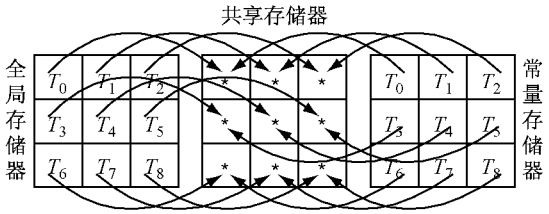


图 3　棋盘式卷积乘示意图

Fig. 3　Schematic diagram of chessboard convolution multiplication

在内核 gaussConvolution 启动后,待处理的影像数据就存放在全局存储器中,高斯核函数数据存放在常数存储器中. block 中各个线程将对应的全局存储器和常数存储器的数据取出完成卷积乘运算后存入共享存储器中. 根据单指令多线程执行模型的工作原理可知,活动 warp 是以时间片分配的:线程调度程序定期从一个 warp 切换到另一个 warp,以最大限度利用多处理器的计算资源. 在同一个 warp 中的 32 个线程是被绑定在一起执行同一指令的.

当 block 中所有线程卷积乘计算完毕后,采用并行归约求和算法计算出影像中某个像素点的卷积值. 最后将计算结果通过主机端控制从设备端转移到主机端的内存.

3.2.2　线程的任务分配及映射策略

(1) 剔除边缘点并行计算中线程的任务分配及映射

在候选特征点检测完成后,形成了一个线性数据结构集合. 因此可以分配一个线程去完成是否剔除一个边缘点的判断. 通过 CUDA 在每个 block 中开辟 512 个线程并行判断是否保留该候选特征点. 候选特征点位置与线程块中的某一个线程在整个网格中的一维编号之间的地址映射关系为

```
tid=blockDim.x * blockIdx.x+threadIdx.x;
```
(7)

(2) 欧氏距离并行计算中线程的任务分配及映射

关键点的 128 维描述子可以看成是一个线性数据结构集合,因此可以分配一个线程去完成一对相应维度的差平方计算,其线程地址映射关系为 tid=threadIdx.x. 通过 CUDA 在每个 block 中开辟 128 个线程并行计算出每一对维度的差平方值,并将结果保存到共享存储器中,每个线程还负责一对差平方值的求和,其地址映射关系为 bid=blockIdx.x.

4 实验结果与分析

4.1 实验运算平台

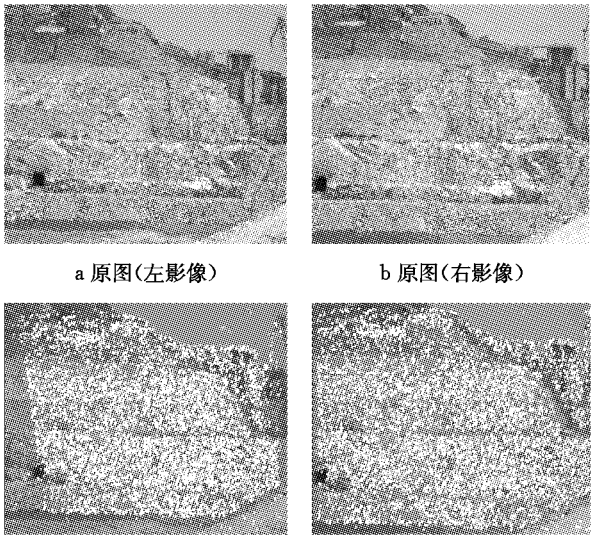
硬件平台:GPU为GeForce GTX 285,含有30颗流多处理器,240颗CUDA核,其核心频率为648 MHz,CUDA核频率为1 476 MHz,显存位宽为512 bit,显存带宽为159 GB·s^{-1}.CPU为Intel i7 Quad Core 2.67 GHz,系统内存为4.0 GB.

软件平台:操作系统为Microsoft Windows XP,Visual Studio C++ .NET 2005,CUDA Driver 190.38,CUDA Toolkit 2.3.

4.2 实验步骤与数据记录

为了进行多组数据的对比实验,首先对原始影像数据进行预处理,通过裁剪获得影像大小分别为101×128,135×187,284×375,489×561,687×765,784×832,874×962,1 082×1 174,1 200×1 435,1 521×1 689,1 782×1 847,2 045×2 184和2 350×2 570共13组实验数据.

图4a和b是影像大小为687×765的原始影像对,图4c和d为对原始影像对采用11×11滤波窗口进行CPU串行处理SIFT特征匹配和GPU并行处理SIFT特征匹配.



a 原图(左影像) b 原图(右影像)

c CPU和GPU影像匹配的左影像 d CPU和GPU影像匹配的右影像

图4 SIFT特征匹配效果

Fig.4 Effect of feature matching for SIFT

以经过预处理的13幅不同影像大小的影像对,滤波窗口为21×21进行SIFT特征匹配对比实验,分别运行SIFT特征匹配的CPU系统和GPU系统,并记录处理时间,见表1.以489×561影像大小的影像对进行不同滤波窗口大小的SIFT特征匹配对比实验,分别运行SIFT特征匹配的CPU系统和GPU系统,并记录处理时间,见表2.

表1 不同影像大小SIFT特征匹配串并行性能对比

Tab.1 Serial and parallel SIFT feature matching algorithm performance comparison of different image sizes

影像大小	串行处理时间/ms	并行处理时间/ms	加速比
101×128	560.900	22.084	25.40
135×187	1 217.200	44.935	27.09
284×375	5 876.500	209.730	28.02
489×561	15 909.400	556.242	28.60
687×765	30 954.300	1 073.684	28.83
784×832	38 657.400	1 338.553	28.88
874×962	50 016.600	1 727.022	28.96
1 082×1 174	76 250.100	2 618.794	29.12
1 200×1 435	103 609.600	3 523.316	29.41
1 521×1 689	155 734.800	5 342.918	29.15
1 782×1 847	199 875.200	6 933.413	28.83
2 045×2 184	273 734.400	9 748.376	28.08
2 350×2 570	369 437.500	21 204.042	17.42

表2 不同滤波窗口SIFT特征匹配串并行性能对比

Tab.2 Serial and parallel SIFT feature matching algorithm performance comparison of different filter windows

滤波窗口大小	串行处理时间/ms	并行处理时间/ms	加速比
5×5	1 196.250	194.352	6.16
7×7	1 857.300	263.887	7.04
9×9	2 385.780	275.318	8.67
11×11	3 303.667	300.598	10.99
13×13	4 425.000	358.651	12.34
15×15	7 460.950	410.507	18.17
19×19	13 101.550	551.527	23.76
21×21	16 065.650	634.668	25.31

4.3 性能分析

左、右影像提取的特征点数分别为52 456,54 265,匹配成功点数8 137;利用定向参数统计匹配点上下视差,中误差5.32 μm(0.546个像素).

实验结果表明:随着影像大小的提高和滤波窗口的增大,GPU的加速效果十分明显.例如:对影像大小为1 200×1 435的影像进行SIFT特征匹配GPU运算,加速比达到了近30倍.对滤波窗口为21×21的影像进行SIFT特征匹配GPU运算,加速比达到了25倍.

图5和6列出了基于GPU的SIFT特征匹配并行算法的加速比对比曲线图.从图5可以看出,当影像大小较小时,随着影像大小的增大,GPU的加速比有一个小幅上升过程并达到峰值.当影像大小超

过 1 200×1 435 之后，加速比曲线开始出现一个比较平缓的下降过程，但是当影像大小达到 2 350×2 570 时出现了较陡峭的下降趋势. 这主要是由于当影像大小较大时，由于总线带宽的限制，显存-主存间大量的数据通信耗时较多，出现了明显的通信瓶颈现象.

从图 6 可以看出，随着影像滤波窗口的增大，曲线斜率急剧变大，曲线变得十分陡峭，加速比曲线呈现出一种近线性增长的趋势，并行处理的性能提升效果比较明显，这说明对于采用大滤波窗口时系统可以获得较大的加速效果. 随着影像滤波窗口的增大，每个线程块中包含的线程数也随之增大，提高访问全局存储器和共享存储器的效率，这样越容易隐藏存储器延时.

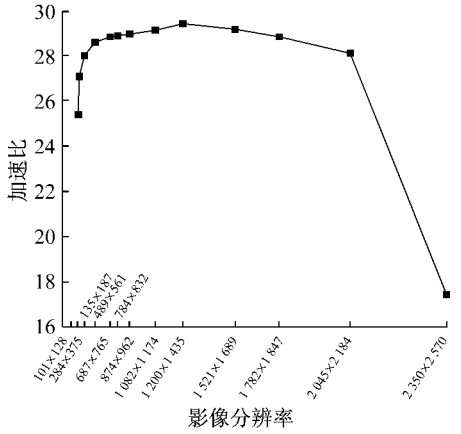


图 5　不同影像大小 GPU 加速效率趋势图

Fig.5　Trend chart of GPU acceleration efficiency for different image sizes

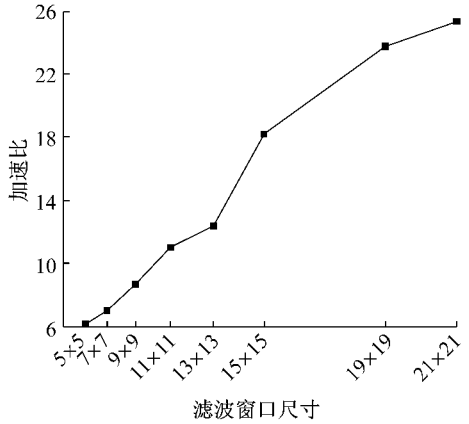


图 6　不同滤波窗口的 GPU 加速效率趋势图

Fig.6　Trend chart of GPU acceleration efficiency for different filter windows

5　结语

针对 SIFT 特征匹配算法在处理遥感影像中遇到的算法复杂度高、数据量大、计算时间长，导致影像匹配的速度较慢，应用受到了限制这一问题，本文提出了基于单指令多线程处理模式，在 GPU 上实现的 SIFT 特征匹配并行算法，并具体给出了 SIFT 匹配算法实现的步骤和性能优化的方法. 实验结果表明，与 CPU 计算相比，采用 GPU 加速能够获得最大近 30 倍的加速比. GPU 大量的并行计算单元和超强的计算能力，为遥感影像处理提供了高效、廉价的处理平台，为海量遥感数据高效处理提供新途径.

参考文献：

[1] Lowe D G. Distinctive image features from scale-invariant keypoints [J]. International Journa of Computer Vision, 2004, 60(2):91.

[2] Sinha S, Frahm J, Pollefeys M, et al. Feature tracking and matching in video using programmable graphics hardware [EB/OL]. [2007-3-25]. http://citeseerx. ist. psu. edu/viewdoc/download? doi=10.1.1.81.9737&rep=rep1&type=pdf.

[3] Warn S, Apon A, Cothren J. Accelerating SIFT on hybrid clusters [C]// Proceedings of the ACM SIGSPATIAL 2nd International Workshop on High Performance and Distributed Geographic Information Systems. [S.l.]: ACM SIGSPATIAL, 2011:2-9.

[4] 袁修国，彭国华，王琳. 基于 GPU 的变型 SIFT 算子实时图像配准 [J]. 计算机科学，2011，38(3):300.
YUAN Xiuguo, PENG Guohua, WANG Lin. GPU-based real time image registration with variant SIFT [J]. Computer Science, 2011, 38(3):300.

[5] 高玉鹏，何明一. 动态背景下基于帧间差分与模板匹配相结合的运动目标检测[J]. 电子设计工程，2012，20(5):142.
GAO Yupeng, HE Mingyi. Moving target detection combined two frame differences with template matching methods under dynamic background [J]. Electronic Design Engineering, 2012, 20(5):142.

[6] Mikolajczyk K, Schmid C. A performance evaluation of local descriptors[J]. IEEE Trans Pattern Analysis and Machine Intelligence, 2005, 27(10):1615.

[7] 田文，徐帆，王宏远，等. 基于 CUDA 的尺度不变特征变换快速算法 [J]. 计算机工程，2010，36(8):219.
TIAN Wen, XU Fan, WANG Hongyuan, et al. Fast scale invariant feature transform algorithm based on CUDA [J]. Computer Engineering, 2010, 36(8):219.

[8] Jonathan Horgan, Francis Flannery, Daniel Toal. Towards real time vision based UUV navigation using GPU technology [C]// OCEANS 2009-EUROPE. Bremen: [S.l.], 2009: 1-6.

[9] Seth Warn, Wesley Emeneker, Jackson Cothren, et al. Accelerating SIFT on parallel architectures [C]// IEEE International Conference on Cluster Computing and Workshops. [S.l.]: IEEE, 2009: 1-4.

[10] YU Qiao, WEI Wang, LIU Jianzhuang, et al. A theory of phase singularities for image representation and its applications to object tracking and image matching [J]. IEEE Transactions on Image Processing, 2009, 18(10):2153.

[11] Marko Heikkilä, MattiPietikäinen, Cordelia Schmid. Description of interest regions with local binary patterns [J]. Pattern Recognition, 2009, 42(3):425.