科学通报 2015 年 第 60 卷 第 15 期: 1408 ~ 1413
论 文 www.scichina.com csb.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

一米真空太阳望远镜 Level 1 级图像选帧的 GPU 实现

施正[①], 向永源[②], 邓辉[①], 季凯帆[①], 卫守林[①②*]

① 昆明理工大学云南省计算机技术应用重点实验室, 昆明 650500;
② 中国科学院云南天文台, 昆明 650001
* 联系人, E-mail: wsl@cnlab.net

2015-01-09 收稿, 2015-02-13 接受, 2015-04-29 网络版发表
国家自然科学基金(11263004, 11203077, 11163004, 11403009)资助

摘要 抚仙湖一米真空太阳望远镜Level 1数据是利用图像选帧位移叠加技术获得的, 目前Level 1处理的时间约20 s, 大约是图像采集时间的1.7倍. 为了达到减少其所需时间的目的, 本文采用GPU技术来实现选帧流程, 包括斑点干涉选帧和互相关图像位移叠加. 在CUDA环境下, 实现了利用GPU处理大量傅里叶变换和图像矩阵的并行四则运算等功能. 在我们的实验环境下, 对100帧图像的处理只需要0.6 s. 同时对整个流程中各个模块的运行时间也进行了详细的测量、分析和讨论.

关键词
NVST
GPU
互相关
斑点干涉
图像重建选帧

太阳高分辨成像观测是抚仙湖一米真空太阳望远镜(NVST)最重要的观测方式[1]. 凭借了优良的视宁度以及精细的图像处理技术, 从2010年正式观测以来, NVST已经获得了大量的高分辨太阳图像, 并开始产出越来越多的科学成果, 得到国际国内的日益重视.

众所周知, 地基望远镜的成像受大气湍流影响导致图像的闪烁和抖动从而限制了图像分辨率[2]. Lucky Imaging技术是目前应用比较成熟的选帧技术, 其基本原理是按照一定的像质评价标准, 挑选出像质较好的一部分图像进行配准叠加得到分辨率明显提高的图像, 在可见光波段可以很好地工作, 对未经自适应改正的可见光波段的短曝光图像进行处理, 也可得到高分辨率图像[3]. NVST高分辨观测数据采用事后斑点成像技术来提高图像的分辨率, 根据斑点重建算法的不同, 所重建出来的数据分Level 1和Level 1+两个等级, 两个级别的重建算法都是对大量的瞬时短曝光图进行大量的复杂计算得到[4]. 数据处理都比较耗时无法达到准实时观测, 比如当前色球数据, 100帧的短曝光图观测时间是12 s, 而Level 1处理时间是20 s, 处理时间是观测时间的1.7倍. 这仅仅是对图像做整数像元的处理, 若对图像做亚像元处理, 那么将会花费更长的运行时间[5,6]. 诚然, 采用性能更好的计算机或者大规模的计算机集群, 肯定是提高计算速度的一种方法, 如美国大熊湖天文台[7]采用80核的计数机集群做图像重建工作. 但一些新技术的应用也可以达到大大提高图像处理能力的效果, 如GPU(Graphic Processing Unit)技术[8]. 2013年, 中国科学院国家天文台怀柔观测基地实现了一套基于GPU技术的实时磁场深积分系统, 获得非常好的结果.

本文的研究工作是针对NVST的Level 1的数据处理, 该处理方法是现在云南天文台所使用的算法, 此算法已经在云南天文台使用多年, 在不改变原有算法的情况下将原有算法利用GPU实现. 在CUDA(Compute Unified Device Architecture)环境下实现了短曝光图像的选帧功能. 实验表明, 对100帧图像的Level 1处理速度小于1 s. 既保证了和原有算法同样的精准度同时又提高了程序的运行效率.

引用格式: 施正, 向永源, 邓辉, 等. 一米真空太阳望远镜 Level 1 级图像选帧的 GPU 实现. 科学通报, 2015, 60: 1408–1413
Shi Z, Xiang Y Y, Deng H, et al. The frame selection for New Vacuum Solar Telescope Level 1 data processingbase on GPU (in Chinese). Chin Sci Bull, 2015, 60: 1408–1413, doi: 10.1360/N972015-00032

1 GPU和CUDA技术

GPU最早是为提高计算机的图形图像显示的实时性和高效性而诞生的. 但随着它的发展, 已经成为了高密集度、高并行度、多线程, 拥有强大计算能力和极高存储器带宽的多核处理器[9]. 由于有更多的晶体管用于数据的处理而不是用于数据的缓存和数据的流量控制, 因此特别适合处理数据的并行计算. 现在大多数的GPU处理器是集成在计算机显示卡上或者一个独立的GPU卡上面[10]. 图1表示了一个GPU的层次结构, 数据首先从主机复制到GPU设备内存中, 然后由设备中的线程来完成对数据的操作, 并将最后的结果返回到主机. 在GPU设备中包含了多个网格(grid), 每个网格中包含了多个线程块(block), 线程块中又包含了多个线程(thread), 其中每个单元的最大数量由GPU卡的级别所决定.

CUDA是一种由英伟达(NVIDIA)公司推出的通用并行计算架构. 开发人员可以通过一些高级语言, 如C语言在CUDA架构下编写程序, 所编写出的程序就可以在支持CUDA的GPU处理器上运行. CUDA为一些复杂的计算提供了很多的应用程序接口, 例如本实验使用到CUBLAS(NVIDIA, 2013. CUBLAS_Library. http://docs.nvidia.com/cuda/pdf/CUBLAS_Library. pdf)和CUFFT库(NVIDIA, 2013. CUFFT_Library. http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf). CUBLAS中提供了大量对矩阵的基础性操作, 而CUFFT库中提供了一到三维各种数据类型的傅里叶变换函数.

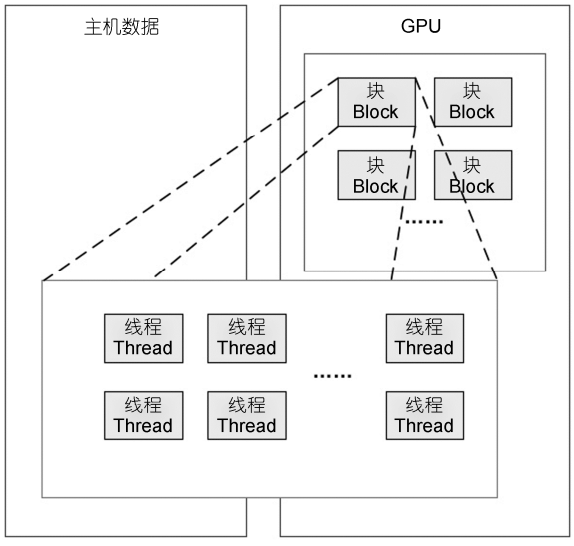

图 1 GPU 卡的层次结构
Figure 1 The hierarchy of GPU card

2 基于斑点重构技术的选帧方法

NVST的Level 1数据处理采用的是基于斑点归一化功率谱统计的选帧方法[11], 然后再对质量较好的短曝光像进行位移叠加.

其步骤主要包括下面几个方面:

(1) 对一组短曝光像(如100帧)进行序列对齐, 即将所有图像对齐到基准图像[12];

(2) 在对齐的图像序列中选取相同的子区域加窗后进行傅里叶变换, 并除以零频率的功率, 得到每帧图像的归一化的加窗功率谱;

(3) 在计算图像序列的平均归一化功率谱的基础上, 得到每个图像功率谱与平均功率谱的比值, 即相对功率谱;

(4) 根据我们关心的频率带宽计算相对功率谱在这个带宽中的积分;

(5) 对所有图像的这个带宽积分值进行排序;

(6) 根据图像质量排序, 选取一定比例的优质图像, 例如30%;

(7) 首先将质量最好的图像作为基准图像, 然后把选出的图像相对于基准图像进行对齐, 并进行位移叠加, 得到选帧叠加的结果.

在整个处理过程中, 主要包括的是两次图像位移量的测量和利用功率谱的选帧. 从计算过程上看, 二维图像的傅里叶变换是整个算法的核心.

3 GPU实现

Level 1选帧的工作流程如图2所示, 分为CPU和GPU两大流程. 首先CPU部分从硬盘中将所有图像数据读取到内存, 然后再将所有图像数据复制到GPU显存中, 在GPU中完成本次实验的全部过程, 最后将处理后的图像输出到CPU内存中.

虽然GPU可以对显存中的数据作处理也能对内存中的数据作处理, 但GPU卡是通过PCIE接口与主板相连接, 这就会涉及到主板的传输速率. 就目前的情况来说GPU显存的读写速度大大高于CPU内存, 所以在数据量大、计算程度复杂的情况下, 将所有数据存放于GPU显存中会大幅的提高程序执行效率.

第一, 我们通过使用CUDA提供的标准函数cudaMemcpy将数据从CPU内存中复制到GPU内存中.

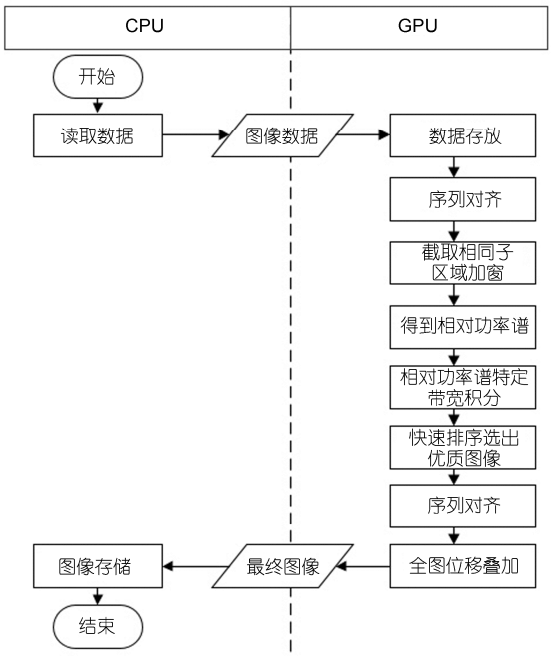

图 2 Level 1 选帧流程图

Figure 2 Flow chart of Level 1 frame selection

第二，在GPU端的处理流程中，快速傅里叶变换是使用最多的模块，也是最耗时的. 在我们的处理过程中，如果图像序列含有M张图像，希望提取最优图像数为N张，那么我们在第1次序列对齐时需要做$2M$次快速傅里叶变换，在计算功率谱时需要做M次快速傅里叶变换，第2次序列对齐时需要做$2N$次快速傅里叶变换，整个流程一共需要进行$3M+2N$次的快速傅里叶变换. 对于这些傅里叶变换，我们采用了CUDA所提供的CUFFT标准库函数. 在考虑到算法的精度需求和自身设备的内存容量及运算时间的优化，采用了二维单精度复数的快速傅里叶变换函数. 为了进一步的优化运行时间，在利用互相关算法计算位移量时，原始的算法是一个图像的傅里叶变换乘上另外一个图像的傅里叶变换的共轭，但我们直接对所需要配准的图像做逆傅里叶变换就直接得到了其傅里叶变换共轭，这使得我们的程序减少了复数共轭运算.

第三，在整个流程中还有大量的图像之间的四则运算，这部分是非常适合采用并行的方法处理的，因为都是像元之间的操作. 对于这部分，我们采用线程并行的方法来实现. GPU中的线程号($\boldsymbol{id}$)是由线程块号($\boldsymbol{a}$)，线程块中总线程数($\boldsymbol{b}$)，线程块中的线程号($\boldsymbol{c}$)3部分组成. 那么就会有$id=a\times b+c$. 很明显id的取值范围就是从0到总线程数. 这里我们将总线程数设置为图像的像素值，那么图像中的每个像元就分配了一个线程，只要每个线程执行完一次四则运算，那么对一整幅图像的运算也就完成了.

第四，在图像对齐的时候，我们采用两种方式，一种是图像偏移截取的方式，一种是图像位移的方式.

截取子区域时采用了偏移截取的方法.

(1) 将总线程数定义为子区域的像素值，那么就将子矩阵的每个点分配给了一个线程;

(2) 因为截取的子区域是原图中的一部分，所以不需要做任何复杂的运算，只需要将线程号对应到原图中的坐标即可;

(3) 根据偏移量的不同选择不同的起始位置;

(4) 对应点的赋值.

在对优质图像位移叠加时，采用图像位移的方式:

(1) 将总线程数定义为原始图像的像素值;

(2) 将偏移量作为起始坐标;

(3) 判断线程所对应的点是否在图像内;

(4) 在图像内的点赋值，图像外的点设置为0.

第五，在互相关算法中需要查找最大值和相对功率谱特定带宽积分，这两个步骤我们采用了CUDA所提供的CUBLAS库. 其中cublasIcamax函数来获取矩阵中最大值的下标，cublasSasum函数可对矩阵进行积分运算.

第六，采用快速排序方法，将带宽积分值由大到小排序. 其具有带宽最大值的图像即为质量最佳的图像，也是最后位移对齐的基准.

4 性能测试结果

对上述的选帧方法的GPU实现进行了全面的性能测试. 实验采用的计算机配置为AMD opteron processor 4180处理器，主频为2.6 GHz，内存为32 GB; GPU卡为英伟达公司的Tesla C2050，其具有448个核频率为1.15 GHZ，单精度峰值性能超过1 Tera-flop; Linux操作系统版本为Community Enterprise Operating System 6; 开发环境为NVIDIA Nsight Eclipse Edition 6.0.

测试数据为NVST于2014年11月5日获得的100帧H_α短曝光图像，图像尺寸为1024像素×1024像素. 图像序列对齐尺寸为图像中央的612像元×612像元，

功率谱计算区域为图像中央512像元×512像元. 选帧比例为30%. 测试结果如表1所示.

在实际环境下图像数据直接来源于CCD, CCD的采集数率大约为120 ms一帧, 即对100帧图像的操作在12 s内完成即可达到实时性的要求.

本实验中GPU的总运行时间为0.6 s, 其中包含了400 M数据从内存传入GPU显存和得到最终大小为4 M的Level 1图像. CCD在实时系统中直接存放入内存, 也就不存在硬盘读入的问题. 在写入文件方面以5400转硬盘为例, 速率大约为30 MB/s, 最后将Level 1图像写入硬盘大约0.13 s. 综合所有情况来看, 本实验处理时间不会超过1 s, 这样在CCD采集下一组100帧图像的时候, 就完全可以有时间将前一组数据处理完成, 并存盘, 从而达到实时处理的目的.

5 讨论

GPU的对图形操作的并行性和密集运算能力在本实验中得到了充分的体现. 100帧短曝光图像的Level 1选帧的GPU计算时间花费为0.6 s, 基本上可以达到实时处理短曝光图像的能力. 从各个处理部分的时间开销上看, 最耗时的是100帧图像的对齐, 占了整个处理过程的42%. 其原因是在这个过程中包括2次傅里叶变换. 虽然每次傅里叶变换的速度可以达到1.2 ms, 但100张图像的配准就耗时250 ms; 第二耗时的部分是数据从内存传输到GPU上. 这一直是GPU应用的一个瓶颈, 其受制于GPU卡的总线宽度. 在我们的实验环境下, 实测的传输速率为2.7 GB/s, 与PCIE ×16的总线速度基本上是一致的; 另外所有图像的归一化加窗功率谱的计算也有一定的时间开销. 这3项的时间开销占了总处理时间的78%. 但实时上, 这3个步骤在实时处理的时候是可以采用流式计算的方式进行的. 因为单帧的处理时间仅为4.7 ms, 也就是在下一帧图像的曝光时间中, 前一帧图像的这些处理步骤就可以同步完成.

在第一次调用CUBLAS库和CUFFT库进行运算时, 会有一个一次性CUDA库环境加载的时间. 在我们的测试环境下, 每个库启动的时间都在140 ms左右. 在性能测试中, 我们将这部分的时间扣除在外. 无论是在实时系统, 还是对历史数据的批处理系统中, 这个时间均是一次性的开销.

另外, 在我们的测试中也没有考虑硬盘的读出写入时间, 这个和存储设备的性能有很大的关系. 同时在实时系统中, 我们可以从CCD中就直接得到图像数据, 而不经过硬盘IO, 所以这个时间开销也不用去考虑.

在本实验中, 为节省系统开销和加快运算速度, 我们采用了单精度简化. 从实验结果来看, 用单精度选出的图像帧和双精度的是完全一样的, 同时我们也对这个简化做了进一步的分析. 由于选帧的核心是对用于代表图像质量的带宽积分值进行排序, 因此我们计算了100帧图像的平均带宽积分值和其标准差, 同时也计算了单/双精度计算之间的平均误差(双精度平均值: 8736.001; 双精度标准差: 88.126; 单精度与双精度误差: 0.012). 可以看到单/双精度之间的误差远远小于带宽积分的标准差, 因此, 使用单精度

表 1 利用 GPU 进行选帧的性能测试结果

Table 1 The time used of GPU frames-selection

项目	时间(ms)	占用时间比例(%)
传输 100 帧图像(合计 400 MB)数据到 GPU	145.7	24.4
基准图像对齐区域单次傅立叶变换时间	1.2	0.2
计算后续 99 张图像相对于基准帧的偏移量	249.4	41.7
分别计算 100 帧图像的归一化加窗功率谱	72.1	12.0
计算 100 张图像的平均功率谱	6.2	1.0
分别计算 100 张图像的相对功率谱	8.2	1.4
分别计算 100 张图像的带宽功率积分	6.0	1.0
对带宽功率积分排序, 并选择最佳质量的 30%图像	4.1	0.7
计算选出图像相对于最佳图像的位移量	77.3	12.9
对选出图像进行位移叠加, 得到 Level 1 图像	19.9	3.3
其他	8.5	1.4
GPU 运行总时间	598.6	

并不会对结果造成影响. 除非这些图像之间的质量差异非常小, 但在这种情况下, 选帧也失去了意义.

我们的实验和性能测试都证明了合理运用GPU可以大大提高图像的处理速度, 同时完全保证了数据处理结果的正确性. 本工作我们只是实现了NVST的 Level 1级数据的处理, 未来还将尝试将Level 1+级数据的处理也GPU化, 从而提高望远镜的整体数据产出率.

参考文献

1 Liu Z, Xu J, Gu B, et al. New vacuum solar telescope and observations with high resolution. Res Astron Astrophys, 2014, 14: 705–718

2 Yang Z L, Liang Y H, Hu H J, et al. Theoretical and experimental research of lucky imaging technique about extended objects (in Chinese). Laser Optoelectron Prog, 2010, 47: 51–56 [杨忠良, 梁永辉, 胡浩军, 等. 扩展目标幸运成像技术的理论和实验研究. 激光与光电子学进展, 2010, 47: 51–56]

3 Law N M, Mackay C D, Baldwin J E. Lucky imaging: High angular resolution imaging in the visible from the ground. Astron Astrophys, 2006, 446: 739–745

4 Huo Z X, Zhou J F. Methods of astronomical image reconstruction from speckle image (in Chinese). Progr Astron, 2010, 28: 72–92 [霍卓玺, 周建锋. 由斑点图重建天文图像的方法. 天文学进展, 2010, 28: 72–92]

5 Liang B, Shi Z, Lin J B, et al. Correcting false structures in solar deep-integration magnetogram observations (in Chinese). Chin Sci Bull, 2014, 59: 3603–3608 [梁波, 施正, 林佳本, 等. 太阳深积分磁场观测中异常结构的改正. 科学通报, 2014, 59: 3603–3608]

6 Shen Y B, Lin J B, Ji K F, et al. New real-time correlation solar observing system based on GPU for acquiring the deep-integration magnetogram. New Astron, 2013, 25: 32–37

7 Cao W, Gorceix N, Coulter R, et al. Scientific instrumentation for the 1.6 m new solar telescope in big bear. Astro Nachr, 2010, 331: 636–639

8 Yang X, Lin G H, Bai X Y, et al. Applications of GPU computing in solar physics (in Chinese). e-Sci Technol Appl, 2012, 3: 69–76 [杨潇, 林钢华, 白先勇, 等. GPU计算在太阳物理中的应用. 科研信息化技术与应用, 2012, 3: 69–76]

9 Owens J D, Houston M, Luebke D, et al. GPU computing. Proc IEEE, 2008, 96: 879–899

10 Lu F S, Song J Q, Yin F K, et al. Survey of CPU/GPU synergetic parallel computing (in Chinese). Comput Sci, 2011, 38: 5–9 [卢风顺, 宋君强, 银福康, 等. CPU/GPU协同并行计算研究综述. 计算机科学, 2011, 38: 5–9]

11 Liu C Y, Xiang Y Y, Jin Z Y. Image processing with frame selection for Hα solar chromosphere images (in Chinese). Astron Res Technol, 2014, 11: 140–144 [刘长玉, 向永源, 金振宇. 太阳色球图像的选帧处理. 天文研究与技术, 2014, 11: 140–144]

12 Chen J, Feng S, Deng H, et al. Comparison of correlation-based techniques for correcting and stacking solar magnetic-field images (in Chinese). Astron Res Technol, 2013, 10: 201–206 [陈洁, 冯松, 邓辉, 等. 太阳磁场观测中相关位移叠加算法的比较. 天文研究与技术, 2013, 10: 201–206]

A method of Level 1 frames-selection based on GPU for new vacuum solar telescope

SHI Zheng[1], XIANG YongYuan[2], DENG Hui[1], JI KaiFan[1] & WEI ShouLin[1,2]

[1] *Kunming University of Science and Technology Yunnan Key Laboratory of Computer Technology Application, Kunming 650500, China;*
[2] *Yunnan Observatory of Chinese Academy of Sciences, Kunming 650001, China*

The sun high resolution imaging observation is the most important way of New Vacuum Solar Telescope (NVST) located in Fuxian Lake of Yunnan. This paper mainly describes the method of the Level 1 data processing of NVST. Currently, the times used by NVST to calculate the 100 images is about 20 s, which is almost 1.7 times of 100 image acquisition time of 12 s. It can't meet the demand of real time processing. In this situation, we have decided to use GPU to process the data. Because the GPU has multi-threading parallelism and a superiority in image processing. We implemented the algorithm by GPU to short the processing time to meet the demand without changing the original algorithm. Thus, it ensures the accuracy of the original operation without affecting the subsequent operation. The experiment includes two main parts, frames-selection method and shift-and-add method. Frames-selection uses speckle interferometry (SI). It selects part of the images with better quality and then shifts-and-adds them. Not all of the image data are required. Usually we only concerned with the necessary part. So in the election process, we only calculated the frequency and the bandwidth. It will highly improve the accuracy of the selected frame. The Fourier transform spent the most time in this experiment. It runs three times in the frame selection and processes second times in the shift-and add. This part of the length of time consuming operation largely determines the efficiency of the program. The CUDA provides the standard library functions, which called CUBLAS and greatly improve the operating efficiency of the program. The library functions CUBLAS provides matrix operations. In this study, most of the matrix operations referenced standard library functions to achieve a large number of image matrix of the parallel arithmetic and other functions. The rest of them used a GPU multi-threading technology. In our experiments, 100 images calibration only takes 0.25 s, GPU runtime also only 0.6 s, the processing time of a single frame is only 4.7 ms. The time of processing image exposure spends only 12 ms. Therefore, it means that in the next frame of the image exposure time, the processing steps of the previous frame image exposure can be completed synchronously. In the current conditions, it meet the demand of real-time processing.

NVST, GPU, cross-correlation, speckle-interferometry, frames-selection

doi: 10.1360/N972015-00032