第17卷 第12期 2017年4月
1671—1815(2017)012-0223-05

科 学 技 术 与 工 程
Science Technology and Engineering

Vol. 17 No. 12 Apr. 2017
ⓒ 2017 Sci. Tech. Engrg.

遥感图像海陆分割的GPU并行加速

张 帆[1,2] 张立国[1]

(中国科学院长春光学精密机械与物理研究所[1],长春 130033; 中国科学院大学[2],北京 100049)

摘 要 海陆分割是光学遥感图像海上目标识别过程中一个重要步骤,海陆分割的结果直接影响下一步目标识别的效率和正确率;而遥感数据的大量增长,使得图像处理速度变慢,因此分割算法的执行效率变得越来越重要。利用Nvidia开发的统一计算架构CUDA(compute unified device architecture),将海陆分割流程的一系列函数移植到GPU(graphic processing unit)上进行并行处理,能够有效提高算法执行速度。经实验验证最终完成2 000×2 000大小的图片在11 ms内的海陆分割处理。该方法能够满足对于图像数据的传输在25 ms内的处理,适合用于建立地面或者航空搭载的"实时"处理平台。

关键词 海陆分割 并行加速 GPU 实时处理

中图法分类号 TP751.1; **文献标志码** A

光学遥感图像的海上目标识别是指从遥感图像中正确区分出海上区域和陆地区域,再从海上区域提取出感兴趣的目标。因此,海上目标探测的一个很重要的步骤就是进行海陆分离。在光学遥感图像上,陆地景物的细节相比于海上目标更多,因此如果不能将陆地区域和海上区域有效地分离开来,会对后面的海上目标探测造成很大的干扰。因此海陆分割的结果直接影响海上区域目标识别的结果和后面算法的执行效率。

以往对于海洋区域的探测是通过对低分辨率的SAR图像分析实现的。而近年来,搭载着高分辨率光学相机传下来的光学遥感图像的分辨率有了很大提高,比如美国的Quick Bird相机全色空间分辨率达到了0.61 m。这种方法能在光学遥感图像上分辨出更多细节,然而随之带来的是遥感数据的大量增长,使得进行图像分割速度变慢。而在图像分割加速方面,国内外已有许多研究。文献[1]和文献[2]介绍了图像分割的并行化在FPGA系统上的实现,利用硬件的方法可以对分割算法实现加速效果,但是对像遥感图像的大幅图像的处理速度仍然无法满足实时性要求;文献[3]研究了图像分割算法在CUDA上的并行实现,该方法在图像分割的处理速度上有了很大提高,但是所采用的单一的区域生长分割方法不能直接用于空间遥感等复杂场景图像的分割。在国外的研究文献里,文献[4]提出了基于实时性目的分割算法处理,但其实时性不能满足本文分析的大尺寸25 ms实时处理要求。

本文使用显卡上的图形处理单元GPU,将在CPU上串行执行的代码移植到GPU上运行,执行效率有数十倍的提高。该加速方法流程能够快速处理大尺寸遥感图像,在目标识别以前能够快速的进行图像预处理,处理速度能够满足"实时性"要求[5]。同时该方法成本相对于硬件平台较低,在制作地面处理平台或者航空搭载平台方面有一定优势。

1 海陆分割流程

1.1 算法流程概述

在众多的海陆分割算法中,主要有基于先验信息的海陆分离和基于图像特征差异的海陆分离。在基于图像特征差异的海陆分离里面主要包括了基于阈值分割的方法、基于边缘检测的方法和基于区域的方法[6]。

海面上水体在光学遥感图像中表现的灰度值偏低,陆地区域的灰度值偏高,因此如果能找到一个阈值高于海上区域的灰度并且低于陆地区域的灰度,便可将海上区域和陆地区域正确地区分开来。遥感图像海上区域和陆地区域肉眼可分,但是其直方图没有明显的波谷。针对这种类型的图像使用基于图像特征差异的Otsu阈值分割能够获得较好的分割效果,当两个灰度区域均值离的越远其类间方差越大[7]。Otsu法(最大类间方差法)需要求出一个阈值$T(k)$,它将源图像中的像素值分割成C_1和C_2两

2016年9月20日收到

第一作者简介:张 帆(1992—),男,汉族,硕士。研究方向:图像处理目标识别和并行计算加速方面的研究。E-mail: zhangfan_6284@126.com

引用格式:张 帆,张立国. 遥感图像海陆分割的GPU并行加速[J]. 科学技术与工程,2017,17(12):223—227

Zhang Fan, Zhang Liguo. Parallel acceleration of splitting land and sea in the remote sensing image with GPU[J]. Science Technology and Engineering, 2017, 17(12): 223—227

部分。设整个图像灰度值为 i 所对应的发生概率为 P_i。C_1 类的发生概率和平均灰度值为

$$P_1(k) = \sum_{i=0}^{k} P_i \tag{1}$$

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} iP_i \tag{2}$$

C_2 类的发生概率和平均灰度值为

$$P_2(k) = \sum_{i=k+1}^{L-1} P_i \tag{3}$$

$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} iP_i \tag{4}$$

整个图像的平均灰度定义为

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{5}$$

σ_B^2 类间方差的定义为

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \tag{6}$$

最大类间方差法即是通过求出一个最佳的阈值 $T(t)$ 使上面的公式求出的类间方差 σ_B^2 最大。

基于 Otsu 求阈值方法的海陆分割算法的主要思想是将输入的图片取单通道或者单波段的灰度图进行预处理，然后根据 Otsu 方法求出海陆分割阈值，再对分割后的图片进行后处理，得到海陆分割结果。图 1 为海陆分割流程图。

(1) 将输入的图片转成灰度图，并进行平滑滤波。

(2) 取灰度图计算 Otsu 分割阈值。

(3) 根于分割阈值将原图分割成二值图片，陆地为 1，海上为 0。

(4) 将 3 中的二值图片进行膨胀、滤波等处理，去除陆地上小的干扰。

(5) 再根据 4 中的处理结果，提取原图对应的二值图中黑色也即是海上区域。

图 2 是采用上面的算法流程后，在 CPU 端执行 5 步时海陆分割的结果，可以看到陆地的部分已经

图 1 海陆分割流程图

Fig. 1 Flow chart of segmentation

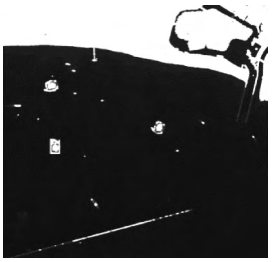

图 2 海陆分割的结果图，灰度图片

Fig. 2 Segmentation result of land and sea, gray image

被置白，只有小部分残留。

1.2 算法执行的效率分析

对上面的算法介绍进行 C 实现，并给出了程序运行的结果。表 1 是在 I5—2400 3.1 GHz 主频，系统内存 16 GB 的 PC 机上对这段算法程序流程进行测时的结果，表 1 给出了我们从步骤 1 到步骤 5 几个主要函数的用时。可以看出，在分割一幅 2 000 × 2 000像素的 8 位深度图片时，整个程序耗时达到了近 200 ms。

表 1 CPU 端算法执行耗时

Table 1 Algorithm elapsed time on CPU

参数	灰度转换/ms	膨胀/ms	阈值分割/ms	均值滤波/ms	总耗时/ms
CPU	5	40	40	90	200

处理从卫星传下来的数据过程中，我们希望计算机的处理速度能够与数据传输的速度相当，达到“0”延迟实时处理。目前发射卫星搭载的相机单个 2 000 × 2 000 的大阵列 CCD 的拍摄帧频已经达到了 40 左右甚至更高。这就意味着在处理图片时，单幅图片的处理速度要在 25 ms 以内甚至更高。而在使用 CPU 端的程序执行时，耗时达到200 ms。这个速度远没有达到要求，因此需要并行加速处理的方法来对程序进行加速。

2 分割算法并行化

在并行化处理中，让 CPU 和 GPU 尽可能同时工作，不让处理器处于等待时间[8]。本文从算法流程以及核函数实现两个方面对于原分割算法进行了改进和优化。

2.1 串行转并行算法流程

分割算法串行转并行的流程图如图 3 所示。

多个函数的并行化要用到 GPU 的内存空间，进行各个核函数之间的数据存储与转移。在 GPU 内存使用方面。我们将图像数据存储于 GPU 的全局存储器中，使各个线程都可以读取操作，而函数执行过程中经常要使用的一些变量和常量则分配到局部

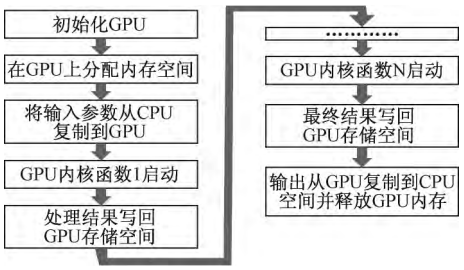

图 3　CUDA 程序函数执行流程图

Fig. 3　Flow charts of functions on CUDA

存储器或者寄存器中,这些存储器的读取访问速度更快。各个核函数在 CPU 端调用之后在 GPU 上执行计算任务,而 CPU 负责整个函数的逻辑流程[9]。

在算法流程里面,本文选取表 1 中耗时比较多的膨胀函数、均值滤波和阈值分割来进行核函数实现和优化加速。

2.2　均值滤波与膨胀并行实现

在图像处理中均值滤波采用的方法多是 8 临域的平均值法。公式为

$$g(x,y) = \frac{\sum f(x,y)}{m} \tag{7}$$

式(7)中,m 为模板中包含当前像素在内的像素总个数,$f(x,y)$ 是相应坐标位置的像素值,$g(x,y)$ 是目标像素位置的像素值。将每个像素的计算映射到每个线程中。通过并发处理上千个线程,来实现运行时间的极大压缩[10,11]。

膨胀函数是一个求临域极大值的过程。

$$dilate(x,y) = \max\left[src(x+x',y+y')\right] \tag{8}$$

同样,将每个像素点求极大值的操作赋一个线程送到流处理器中执行以实现并行。由于排序操作比较耗时,因此我们以求和操作代替求最大值。将像素点 8 临域求和,结果不为 0 则该点赋值 1。

2.3　求 Otsu 阈值并行实现

Otsu 算法的主要计算过程就是图像平均灰度值和类间方差 σ_B^2 的计算。

计算总的灰度值和 C_1、C_2 的分布概率涉及累加计算。在 CPU 中每次只能加一个数的操作。在 GPU 中可以通过归约计算并发多个线程来压缩时间,原理如图 4 所示。图 4 中 8 个数据相加的过程中,CPU 需要 7 次累加计算。在 GPU 计算中,第一次计算,并发 4 个线程执行 4 组数据的加法,第二次计算并发两个线程,第三次计算并发一个线程,只需要 3 次加法操作便完成了 8 个数据的累加操作。理论上 n 个数据的求和计算,并行执行需要 $\log_2 n$ 次加法操作即可。

在类间方差 σ_B^2 的计算中,需要不断地更改阈值 T 以求得不同的方差值,再从中寻找最大值。CPU 端的循环计算是一种非常耗时的计算过程。在 GPU 中可以将循环展开放入每个线程为每个阈值 T 分别进行计算方差,最后再比较求出最大方差[12]。比较最大值类似于加法计算归约求和的过程。在图 4 的加法操作过程中,我们只需要将每次的加法操作更换为比较操作便实现了排序的并行化。

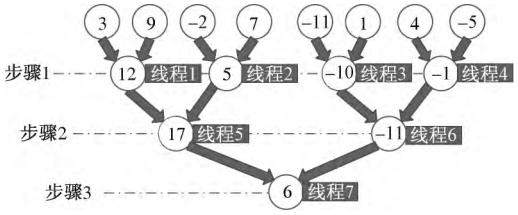

图 4　并行归约求和示意图

Fig. 4　Schematic diagram of parallel reduction

3　实验结果

3.1　实验条件

硬件条件: PC 机采用 I5—2400 3.1 GHz 主频,系统内存 16 GB,Nvidia GTX750Ti 显卡,显存 2 GB 显存位宽 128 Bit,核心频率 1 150 MHz。

软件条件: 在 Windows7 64 位系统,集成开发平台 Visual Studio 2012,GPU 并行计算开发工具 CUDA Toolkit 6.5,计算机视觉库 OpenCV 2.4.8。

3.2　分割和检测结果

GPU 实现海陆分割效果如图 5 所示,表 2 是基于 GPU 并行化处理不同尺寸的遥感图像的处理时间及加速比。实验分别采用1 024 ×1 024(表 2)、2 048 × 2 048(表 3)、4 096 ×4 096(表 4)像素大小的图片。

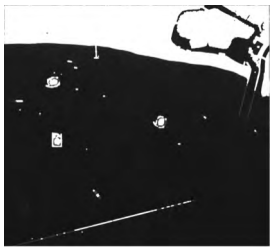

图 5　GPU 实现海陆分割效果

Fig. 5　Segmentation result sea on GPU

表 2　1 024 ×1 024 大小 8 位深度图片 CPU 和 GPU 算法执行耗时

Table 2　Elapsed time on CPU and GPU to the image of 1 024 ×1 024 pixels and 8 bits depth

参数	灰度转换/ms	膨胀/ms	阈值分割/ms	均值滤波/ms	总耗时/ms
CPU	3	8	7	25	52
GPU	0.01	3	0.02	0.06	4
加速比	375	2.6	368	403	13

表 3 2 048 ×2 048 大小 8 位深度图片 CPU 和 GPU 算法执行耗时

Table 3 Elapsed time on CPU and GPU to the image of 2 048 ×2 048 pixels and 8 bits depth

参数	灰度转换/ms	膨胀/ms	阈值分割/ms	均值滤波/ms	总耗时/ms
CPU	5	40	40	90	200
GPU	0.03	9	0.05	0.09	11
加速比	161	4	727	1 000	18

表 4 4 096 ×4 096 大小 8 位深度图片 CPU 和 GPU 算法执行耗时

Table 4 Elapsed time on CPU and GPU to the image of 4 096 ×4 096 pixels and 8 bits depth

参数	灰度转换/ms	膨胀/ms	阈值分割/ms	均值滤波/ms	总耗时/ms
CPU	2	106	114	384	652
GPU	0.09	47	0.08	0.11	48
加速比	285	2.3	1 443	3 428	13.6

表 4 是程序经过优化，生成 Release 版本的结果。从表 3 我们可以看出，2 000 ×2 000 像素的图片从最初的三通道彩色图像经过并行处理分离出海上区域，整个程序花费了 11 ms，相比于之前的 200 ms压缩了近 18 倍。在不考虑后面的目标提取和识别的算法，这个海陆分割目前的算法所达到的速度已经有了很大的提高。单从这部分来看已经能够达到我们所要求的 25 ms 的处理时间。

4 结论

从并行化之后的分割结果图可以得出以下结论。

(1) 基于 GPU 的并行化算法对于遥感图像的处理能够很好的还原基于 C 分割的一系列流程算法，得到相同的结果。

(2) 根据运行的时间结果表格可以看出本文的方法对于原算法在速度上有了很大提升，将海陆分割的算法执行从 CPU 端的 200 ms 加速到 11 ms，能够达到与 FPGA + DSP 的硬件平台相当甚至更快的执行速率。并且 GPU 在提供相同的计算能力的基础上，功耗上相比于硬件平台要小很多。此外，GPU 能够进行算法深入优化和嵌入式硬件开发，有望将算法的执行速率进一步提高。

参 考 文 献

1 封士永，康 彬. 基于神经网络的图像分割算法在 FPGA 上的实现. 电子设计工程，2015;(5): 128—133
Feng Shiyong, Kang Bin. The implementation of neural network based image segmentation algorithm on FPGA. Electronic Design Engineering, 2015;(5): 128—133

2 孙宁建. 基于 FPGA 的一种高速实时图像分割系统研究及其应用. 南京: 南京理工大学，2013
Sun Ningjian. A high-speed real-time image segmentation research and application based on FPGA. Nanjing: Nanjing University of Science and Technology, 2013

3 刘思彤，程 红，孙文邦，等. 面向海上目标的海陆分离方法研究. 电子设计工程，2014;22(15): 96—100
Liu Sitong, Cheng Hong, Sun Wenbang, *et al*. Studies of sea-land segment methods oriented to targets on the sea. Electronic Design Engineering, 2014;22(15): 96—100

4 Hua J X, Jeong M H. Real-time range image segmentation on GPU. 2014 14th International Conference on Control, Automation and Systems (ICCAS). New York: IEEE, 2014: 150—153

5 周海芳. 遥感图像并行处理算法的研究与应用. 长沙: 国防科学技术大学，2003
Zhou Haifang. Research and application of remote sensing image parallel processing algorithm. Changsha: National University of Defense Technology, 2003

6 刘桂雄，申柏华，冯云庆，等. 基于改进的 Hough 变换图像分割方法. 光学精密工程，2002;10(3): 257—260
Liu Guixiong, Shen Bohua, Feng Yunqing, *et al*. Study of image segmentation based on improved Hough transform. Optics and Precision Engineering, 2002;10(3): 257—260

7 程万胜，臧希喆，赵 杰，等. 面向 Otsu 阈值搜索的 PSO 惯性因子改进方法. 光学精密工程，2008;16(10): 1907—1912
Cheng Wansheng, Zang Xizhe, Zhao Jie, *et al*. Modified strategy to inertia weight in PSO for searching threshold of Otsu rule. Optics and Precision Engineering, 2008;16(10): 1907—1912

8 Nvidia C. Nvidia CUDA Programming Guide-v7.5.2016 [2016]. http://docs.nvidia.com/cuda/cuda-c-programming-guide.

9 Cui X H, Charles J S, Potok T. GPU enhanced parallel computing for large scale data clustering. Future Generation Computer Systems, 2013;29(7): 1736—1741

10 Feng W, Xiang H, Zhu Y. An improved graph-based image segmentation algorithm and its gpu acceleration. 2011 Workshop on Digital Media and Digital Content Management (DMDCM). New York: IEEE, 2011: 237—241

11 侯广峰. CUDA 的图像分割并行算法的设计与实现. 大连: 大连理工大学，2013
Hou Guangfeng. Image segmentation parallel algorithm design and implementation based on CUDA. Dalian: Dalian University of Technology, 2013

12 周海兵. 基于 GPU 加速的 Otsu 图像阈值分割算法实现. 大连: 大连理工大学，2009
Zhou Haibing. GPU-accelerated Otsu algorithm of image threshold segmenting. Dalian: Dalian University of Technology, 2009

Parallel Acceleration of Splitting Land and Sea in the Remote Sensing Image with GPU

ZHANG Fan[1,2], ZHANG Li-guo[1]

(Changchun Institute of Optics, Fine Mechanics and Physic, Chinese Academy of Sciences[1], Changchun 130033, P. R. China;
University of Chinese Academy of Science[2], Beijing 100049, P. R. China)

[Abstract] Segmentation algorithm of land and sea is a very important step in remote sensing image target identification. Results of dividing land and sea directly affect the efficiency and accuracy of next target recognition. With the massive growth of remote sensing data, the image processing speed is slower. So the efficiency of segmentation algorithm execution is becoming much more important. The use of CUDA (compute unified device architecture) can transplant the segmentation process from CPU to GPU, accelerating the segmentation algorithm When processing the image date of 2 000 × 2 000 pixels, experiment proved that this method can complete the segmentation of land and sea in image in 25 ms. And it can be used to build a ground or airborne "real-time" processing platform.

[Key words] segmentation of land and sea　parallel processing　GPU　real-time processing