第38卷 第18期
Vol.38 No.18

计 算 机 工 程
Computer Engineering

2012年9月
September 2012

• 图形图像处理 •

文章编号：1000—3428(2012)18—0190—04 文献标识码：A 中图分类号：TP391.41

基于CUDA的拉普拉斯边缘检测算法

孟小华[a,b]，刘坚强[a]，区业祥[a]，张庆丰[a,b]

(暨南大学 a. 计算机科学系；b. 天体测量、动力学与空间科学中法联合实验室，广州 510632)

摘 要： 拉普拉斯边缘检测算法常用于去除CCD天文图像中的宇宙射线噪声，但其串行算法计算复杂度较高。为此，分析拉普拉斯边缘检测算法的并行性，在统一计算设备架构(CUDA)并行编程环境下，提出一种基于CUDA的拉普拉斯边缘检测图形处理单元(GPU)并行算法。分割天文图像得到多幅子图，根据GPU的硬件配置设定Block和Grid的大小，将子图依次传输到显卡进行并行计算，传回主存后拼接得到完整的图像输出。实验结果表明，图像尺寸越大，该并行算法与串行算法相比具有的速度优势越大，可获得10倍以上的加速比。

关键词： 拉普拉斯边缘检测算法；图形处理单元；统一计算设备架构；并行处理；天文图像；宇宙射线

Laplacian Edge Detection Algorithm Based on CUDA

MENG Xiao-hua[a,b], LIU Jian-qiang[a], OU Ye-xiang[a], ZHANG Qing-feng[a,b]

(a. Department of Computer Science; b. Sino-France Joint Laboratory for Astrometry, Dynamics and Space Science, Jinan University, Guangzhou 510632, China)

【Abstract】 Laplacian edge detection algorithm is widely used in the removal of cosmic ray noise in the CCD astronomical images, but it has higher computation complexity of single CPU. To solve this problem, this paper proposes a parallel Laplacian edge detection algorithm with Graphic Processor Unit(GPU) based on Compute Unified Device Architecture(CUDA) by analyzing the parallelism of Laplacian edge detection. The main algorithm running on the main CPU is responsible for split the astronomical image into some subgraphs. Then it sets Block and Grid size according to the GPU hardware configuration, and transfers the subgraph to graphics card for parallel computing. Finally it retrieve the processed subgraph to main memory and joining together to get complete image output. Experimental results show that, with the image size increases, the speed advantage of the parallel algorithm is greater than the serial algorithm, and it obtains more than ten times speedup measured.

【Key words】 Laplacian edge detection algorithm; Graphic Processor Unit(GPU); Compute Unified Device Architecture(CUDA); parallel processing; astronomical image; cosmic ray

DOI: 10.3969/j.issn.1000-3428.2012.18.051

1 概述

CCD天文图像在采集时会受到宇宙射线噪声的影响，破坏图像中的重要信息或者产生星体假象。在众多宇宙射线去除算法中，文献[1]提出基于拉普拉斯边缘检测的宇宙射线去除算法。该算法的优点是不需将整个宇宙射线噪声与周围环境对比，而是仅探测出清晰的宇宙射线边缘独立出宇宙射线的形态，从而能精确地探测并有效地去除噪声点。因此，该算法被广泛使用于去除CCD中的宇宙射线[2]。但由于该算法复杂度比较高，传统的单CPU串行实现效率非常低，尤其是处理批量大型天文图像时这一缺点更为突出。

针对该问题，文献[3]利用MPI在CPU上实现了在多处理器的并行模型，效率提高了50%以上，而且可作为一般性的通用的图形图像处理算法的并行框架，但其缺点是对硬件要求非常高。基于拉普拉斯边缘检测的宇宙射线去除算法是一种易并行算法[4]，即可以把图像简单的划分为多个子图像进行并行处理，而子图像处理间不需要任何通信。基于这种特性，可以对拉普拉斯边缘检测算法进行并行处理，以提高其处理效率。2007年，NVIDIA推出了统一计算设备架构(Compute Unified Device Architecture, CUDA)，方便开发者在图形处理单元(Graphic Processor Unit, GPU)上实现并行化处理[5-6]。利用CUDA强大的并行处理和浮点运算能力，可有效提高拉普拉斯边缘检测算法效率，因此，本文基于CUDA的并行架构实现拉普拉斯边缘检测的并行算法。

2 GPU与CUDA架构

GPU是目前普遍使用的显卡的图像处理器，已成为强大的通用的并行处理单元。它具有单指令流多数据流的特

基金项目： 国家自然科学基金资助项目(10973007)；广东省部产学研结合引导基金资助项目(2011B090400490)；广州市动漫产业发展基金资助项目(2060404)

作者简介： 孟小华(1965－)，男，副教授、硕士、CCF会员，主研方向：并行分布式系统；刘坚强、区业祥，硕士研究生；张庆丰，副教授、博士

收稿日期： 2011-11-22 **修回日期：** 2012-01-31 **E-mail：** xhmeng@163.com

性，主要功能是进行浮点运算生成三角形，进行定点处理和着色处理。GPU 具有强大的并行处理、浮点运算等能力，和高效的数据传输能力。同样，GPU 有自身的缺点，如 GPU 只能通过图形应用接口来编程，不利于程序员开发；GPU 编程有很大的限制性，不够灵活；因为存在带宽(GPU 和显存相互之间一次读入的数据量)瓶颈，使 GPU 计算能力不能充分发挥。

CUDA 是显卡厂商 NVIDIA 推出的计算平台。CUDA 是一个基础架构，采用 C 语言作为编程语言，且提供大量高性能计算指令的开发能力，可在 GPU 强大计算能力的基础上建立一种效率更高的密集数据计算解决方案。CUDA 架构下一般分为 2 个部分：host 端(运行在 CPU 上)和 device 端(运行在 GPU 上)。CUDA 可以让程序员更好地对 GPU 进行编程，让 GPU 在不同领域发挥其强大的计算能力，而不仅局限在图像处理方面[5]。

3　拉普拉斯边缘检测算法

拉普拉斯边缘检测算法[1-2]首先进行拉普拉斯卷积；然后构造噪声模型和好结构模型；再通过拉普拉斯卷积后的值与两模型比得到信噪比，将其与阈值相比较，从而识别出宇宙射线；最后通过 $n\times n$ 模板的中值滤波值替代噪声点，从而消除宇宙射线。

3.1　拉普拉斯算子

因为图像边缘有较大的灰度变化[7]，所以图像的二阶偏导数会在边缘处通过零点。二维函数 $f(x,y)$ 的拉普拉斯算子是如式(1)定义的二阶导数。为更适合数字图像处理，这一方程需要表示为离散形式如式(1)所示：

$$\nabla^2 f=[f(x+1,y)+y+f(x-1),y+f(x,y+1)+f(x,y-1)]-4f(x,y) \tag{1}$$

由此得到拉普拉斯算子的一种模板形式如式(2)所示：

$$\nabla^2 f=\frac{1}{4}\begin{pmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{pmatrix} \tag{2}$$

使用式(2)对一个图像进行卷积得到拉普拉斯图像，通过检测拉普拉斯图像的零点位置，即能检测到原图像的边缘位置。

3.2　算法原理

简单地使用式(2)对 CCD 图像进行卷积，当周围出现高亮像素点时就会出现负值。在这种情况下，与宇宙射线相连接的像素由于负值而遭受到衰减。因此，在进行卷积之前，原始图像 I 需要进行子采样。子采样的结果与子采样因子无关，一般以 2 为子采样因子节省系统开销。

然后将子采样图像与拉普拉斯模板式(2)做卷积得到拉普拉斯图像，如式(3)所示：

$$L^{(2)}=\nabla^{(2)}f\times I^{(2)},\ I^{(2)}_{i,j}=I_{\mathrm{int}[(i+1)/2],\mathrm{int}[(j+1)/2]} \tag{3}$$

在宇宙射线的外部，其边界的拉普拉斯卷积结果为负；在宇宙射线的内部，其边界的拉普拉斯卷积结果为正。因此，把所有拉普拉斯图像值为负的像素都设置为 0，受宇宙射线影响的像素保留，则负值的图像信息就被移除。

由于式(3)将原图像放大了 4 倍，因此需要执行以下操作：

$$L^{+}_{i,j}=\frac{1}{4}\left(L^{(2)+}_{2i-1,2j-1}+L^{(2)+}_{2i-1,2j}+L^{(2)+}_{2i,2j-1}+L^{(2)+}_{2i,2j}\right) \tag{4}$$

此外还需要构造噪声模型，如式(5)所示。该模型仅由泊松噪声和读出的噪声组成，并不包含宇宙射线噪声。

$$N=g^{-1}\sqrt{g\left(M_5\times I\right)+\sigma^2_{rn}} \tag{5}$$

其中，g 是在单位 ADU-1 电子的增益；σ_m 是电子的读入噪声；M_5 是一个 5×5 的中值滤波。

由此可以得到噪声比率 S，如式(6)所示：

$$S=\frac{L^{+}}{f_s N} \tag{6}$$

其中，f_s 是子采样因子(本文为 2)。信噪比 S 反映图像中每个像素点含有噪声的比率，像素点噪声越大，则其比值越大。

拉普拉斯给出了一个像素与它领域像素的区别，但不包含边缘特征性质的信息。由于泊松噪声，真正的天体会在拉普拉斯图像中产生信号，而且它们在像素采样中有着非常均匀平稳的轮廓。这种“采样通量”普遍较小，只要它不超过预测的泊松波动(式(6))就不会产生虚假检测。然而，对于明亮的天体，特别是当点扩散函数没有很好的进行采样时，采样通量可能会很大。因此，有必要去除这种采样通量，移除步骤如下[1,7]：

(1)所有的结构在大于等于 5 个像素的范围内都是平滑的，因此，通过使用 5×5 的中值滤波可以去除，如式(7)所示：

$$S^{'}=S-\left(S\times M_5\right) \tag{7}$$

经过该处理，可以去除高亮星象。根据设定好的阈值 σ_{lim}，则可以判断 $S^{'}$ 值大于 σ_{lim} 的像素点为噪声。

(2)区别星象与宇宙射线。在区域 3×3 的像素范围内宇宙射线与星象有非常相似的像素间变化规律，且它们都有很高的灰度值，得到的噪声比率值都很高。但星象有对称性的特点，这是宇宙射线所没有的，因此，可以根据这个特点来将两者加以区分，可构造一个好结构图像，如式(8)所示。进而可构造第 2 个比率 $F^{'}$，如式(9)所示。其中，M_n 是以 $n\times n$ 为模板的中值滤波，会得到图像中的中频信息，星象的 F 值会比较大，而宇宙射线的 F 值比较小。

$$F=\left(M_3\times I\right)-\left[\left(M_3\ \ I\right)\ \ M_7\right] \tag{8}$$

$$F^{'}=\frac{L^{+}}{F} \tag{9}$$

通过 2 个条件 $S^{'}>\sigma_{\mathrm{lim}}$ 并且 $F^{'}>f_{\mathrm{lim}}$ 时，原图像中此位置的图像就被识别为宇宙射线。其中，f_{lim} 是拉普拉斯图像与好结构图像之间的最小比值。

(3)在原图像中对宇宙射线的像素使用 9×9 为模板的中值滤波进行处理，以达到消除的目的。

4　基于 CUDA 的拉普拉斯算法实现

能够使用 GPU 计算的程序必须具有以下特点：需要处理的数据量比较大，数据以数组或矩阵形式有序存储，

并且对这些数据要进行的处理方式基本相同，各个数据之间的依赖性或者说耦合很小，即具有易并行性。拉普拉斯算法具有易并行性，而处理的 CCD 天文图像一般都比较大，因此，非常适合在 GPU 上实现。

4.1 CUDA 程序特性

GPU 的线程以网格(Grid)的方式组织，而每个网格中又包含若干个线程块(Block)。在同一线程块中，众多线程不仅能够并行执行，而且能够通过共享存储器(Shared memory)和栅栏(barrier)通信。这样，同一网格内的不同块之间存在不需要通信的粗粒度并行，而一个块内的线程之间又形成了允许通信的细粒度并行。这些就是 CUDA 的关键特性：线程按照粗粒度的线程块和细粒度的线程 2 个层次进行组织、在细粒度并行的层次通过共享存储器和栅栏同步实现通信[5-6]。

4.2 并行算法设计

根据 CUDA 的特性[5]，拉普拉斯算法的设计必须对大图像进行一定的分割，分割成有一定像素冗余的子图像，然后子图像被分割到不同的线程块(Block)中并行执行，最后把处理后的子图像拼接起来，组成完整的处理后的图像。算法的设计过程如图 1 所示。

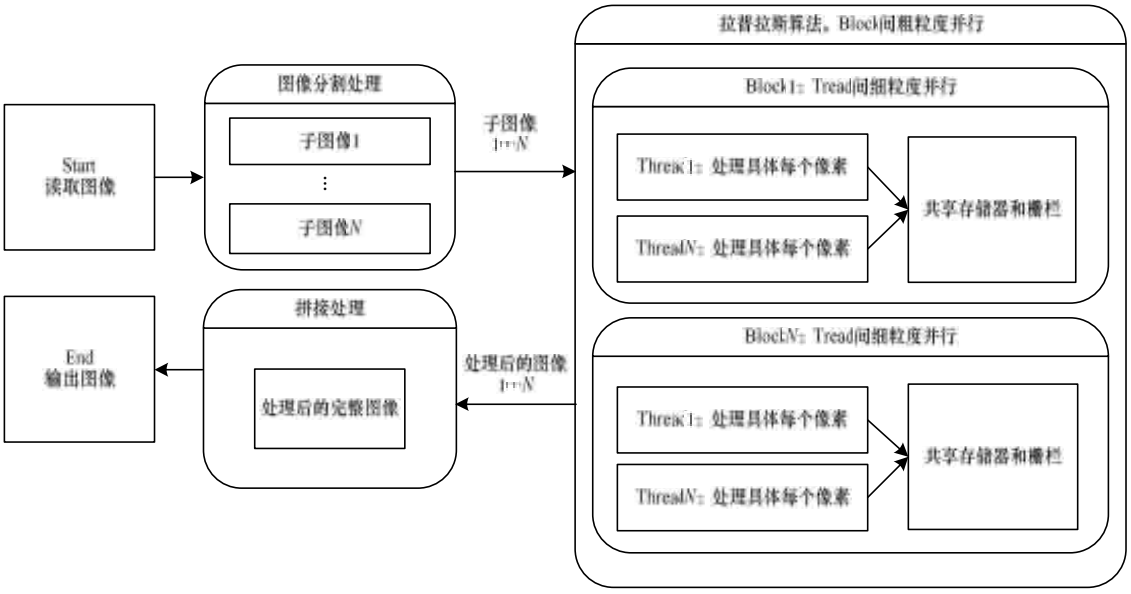

图 1 基于 CUDA 的拉普拉斯算法设计过程

具体步骤如下：

(1)从磁盘读取天文图像文件数据。天文图像的格式基本以 Fits 为主，可以通过 Cfitsio 开源程序库实现对 Fits 图像文件读写操作。

(2)图像的分割处理。这部分运行在 CPU 上。这里子图像的分割大小主要根据 Block 可容纳的最大线程数目，根据 CUDA 的编程手册，GeForce 9300M 中的 Block 可容纳的最多线程数为 512。考虑到 GPU 的线程结构和特性，包含多个流处理器(Streaming Multiprocessor, SM)，能容纳最多的 Thread，效率较高，因此，本文把 Block 的大小设计为 256，子图像的大小也为 256 个像素(16×16 像素)。对于边界位置，无法分割成 16×16 像素的子图像，以补 0 的方式处理，而处理时，根据原图像的边界进行忽略。由于处理子图像时，会关联到相邻的像素，因此子图像中需要增加冗余像素，即子图像的边界位置是相邻子图像的边界，对于不包含原图像边界的子图像，其冗余像素为 60，即真正处理的像素只有 15×15 像素。而对于包含边界的子图像，作例外处理，其冗余像素为 31(包含 2 条原图像边界)或者 46(包含 1 条原图像边界)。

(3)利用拉普拉斯算法对每个像素进行处理。其中，首先需要对应开辟显存内存，然后把分割好的子图像数据输送到显存上，在 GPU 上并行执行，最后把处理后的子图像数据传到内存，让 CPU 做后续处理。CUDA 编程主要是分配 Grid 和 Block 的大小，这里的依据主要是子图像的大小。譬如，对于 $N\times N$ 的原图像，其 Block 的大小为 16×16 像素，而 Grid 的大小为$(N/15+1)\times(N/15+1)$。由于拉普拉斯算法的复杂性，这个算法的处理无法在一次的并行处理完成，需要分开几个步骤处理，如图 2 所示。每个步骤都是并行处理(运行在 GPU 上)，而步骤间是串行处理(运行在 CPU 上)。

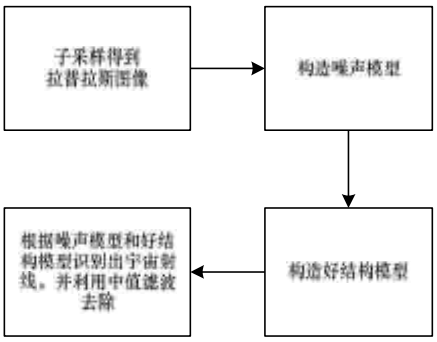

图 2 拉普拉斯算法的 CUDA 模型

(4)把处理好的子图像拼接在一起，拼接时要处理好冗余像素。最后输出处理好的图像文件。

4.3　移植图像处理算法到 GPU 上的并行框架

由于本文的设计是对原图像进行了分割处理，这样具有一定的通用性。因为根据 CUDA 的特性[5]，Block 内部的多个线程是可以互相通信地并行运行，而分割成的子图像大小刚好是一个 Block 的大小，以 Block 为单位运行。所以对于不同的图像处理算法，即使是非完全易并行的算法，利用本文设计好的图像分割和拼接，然后根据算法本身，对子图像进行处理(根据 CUDA 的语法)，即能简单地移植到 GPU 上。因此，本文设计具有一定的通用性，可以简单地把其他图像处理算法移植到 GPU 上并行执行，提高效率。

5　实验与结果分析

实验环境为台式电脑：CPU 1.83 GHz 酷睿双核 T5550，显卡为带 16 个流处理器的 NVIDIA GeForce 9300M。实验方法是先通过天文图像处理软件工具 IRAF[8] 产生一批不同尺寸的加入了宇宙射线的模拟天文图像，然后对比测试 IRAF 串行拉普拉斯处理程序和本文实现的拉普拉斯 CUDA 并行程序分别处理这些图像的速度。由于 2 种不同的实现对于图像文件的读写都是一致的，都是运行在 CPU 端，因此忽略图像文件的读写所消耗的时间。基于 GPU 并行和基于 CPU 串行 2 种不同实现所消耗的时间对比如图 3 所示，其中数据是经过 50 次的测试取平均所得。

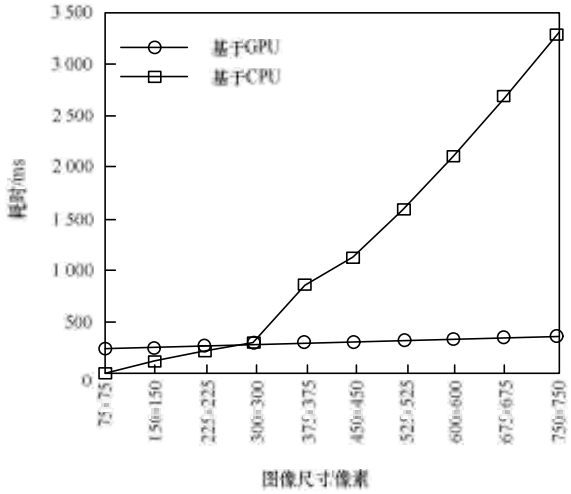

图 3　CUDA 显存与内存间数据传送消耗的时间

如图 3 所示，当像素小于 900 000 像素时(300×300 像素)，运行在 CPU 上的串行算法比运行在 GPU 的并行算法的效率要高，这是由于基于 CUDA 编程需要显存到内存的数据交换，数据交换带来了较大的时间损耗，约为 200 ms±50 ms(像素小于 750×750 的情况下)，因此，当图像较小时，其时间损耗较高，如 75×75 像素图像，数据传送的时间损耗占了约 90%，当图像像素较小时，不太适合运行在 GPU 上。而当图像像素大于 900 000 像素时，GPU 的优势非常明显，而且随着像素的增长，GPU 的加速比呈现指数上升，如图 3 所示，对于 750×750 像素图像，GPU 比 CPU 提速了 10 倍以上。

6　结束语

在大批量处理大型 CCD 天文图像时，尽管拉普拉斯边缘检测算法有良好的效果，但因其速度问题而缺乏实用性。然而正是由于拉普拉斯边缘检测算法具有易并行性，可以实现在具有强大图像处理和浮点计算能力的 GPU 上并行处理，从而大大提高了拉普拉斯边缘检测算法的效率。实验结果表明，只有 16 个流处理器的低端 GeForce 9300mG 显卡上的 GPU 并行实现比酷睿双核 CPU 的串行实现速度提高了 10 倍以上，从而有效地解决了算法速度上的缺陷。本文设计具有一定的通用性，可作为通用性框架，根据本文的设计框架，开发者可以简单地把其他图像处理算法移植到 GPU 上并行执行，提高开发效率。下一步研究工作是从 2 个方面对拉普拉斯边缘检测去宇宙射线算法本身进行改进：(1)优化拉普拉斯算子的模版，进一步提高宇宙射线识别率，同时避免误判。(2)研究如何避免算法中中值滤波处理可能带来的图像信息丢失。

参考文献

[1] van Dokkum P G. Cosmic-ray Rejection by Laplacian Edge Detection[J]. Publications of the Astronomical Society of the Pacific, 2001, 113(789): 1420-1427.

[2] 刘婷婷, 彭青玉. 消除 CCD 图像中宇宙射线的算法的比较[J]. 天文研究与技术, 2010, (2): 140-149.

[3] Mighell K J. CRBLASTER: A Parallel-processing Computational Framework for Embarrassingly-Parallel Image-analysis Algorithms[J]. Publications of the Astronomical Society of the Pacific, 2010, 122(896): 1236-1245.

[4] Mighell K J. CRBLASTER: A Fast Parallel-processing Program for Cosmic Ray Rejection[C]//Proceedings of Advanced Software and Control for Astronomy II. Marseille, France: SPIE, 2008.

[5] 孟小华, 刘坚强. AVS 标准中整数 DCT 变换的 CUDA 并行算法[J]. 微计算机应用, 2011, (11): 40-46.

[6] Bolz J, Farmer I, Grinspun E, et al. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid[J]. ACM Transactions on Graphics, 2003, 22(3): 917-924.

[7] Gonzalez R C, Woods R E. Digital Image Processing[M]. 3rd ed. Upper Saddle River, USA: Prentice-Hall, 2002.

[8] Wells L A, Bell D J. Cleaning Images of Bad Pixels and Cosmic Rays Using IRAF[EB/OL]. [2011-12-02]. http://iraf.noao.edu/docs/recommend.html.

编辑　金胡考