基于图形处理器的边缘检测算法

张 楠[1,2] 王建立[1] 王鸣浩[1,2]
(中国科学院长春光学精密机械与物理研究所 长春 130033)[1]
(中国科学院研究生院 北京 100039)[2]

摘 要 边缘检测是一种高度并行的算法,计算量较大,传统的CPU处理难以满足实时要求。针对图像边缘检测问题的计算密集性,在分析常用边缘检测算法的基础上,利用CUDA(Compute Unified Device Architecture,计算统一设备架构)软硬件体系架构,提出了图像边缘检测的GPU(Graphics Processing Unit,图形处理器)实现方案。首先介绍GPU高强度并行运算的体系结构基础,并将Roberts和Sobel这两个具有代表性的图像边缘检测算法移植到GPU,然后利用当前同等价格的CPU和GPU进行对比实验,利用多幅不同分辨率图像作为测试数据,对比CPU和GPU方案的计算效率。实验结果表明,与相同算法的CPU实现相比,其GPU实现获得了相同的处理效果,并将计算效率最高提升到了17倍以上,以此证明GPU在数字图像处理的实际应用中大有潜力。

关键词 图像处理,边缘检测,图形处理器,计算统一设备架构
中图法分类号 TP391 **文献标识码** A

Edge Detection Based on GPU

ZHANG Nan[1,2] WANG Jian-li[1] WANG Ming-hao[1,2]
(Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China)[1]
(Graduate University of Chinese Academy of Sciences, Beijing 100039, China)[2]

Abstract Edge detection is a highly parallel algorithm with great computation. It is difficult to increase the speed of the algorithm by CPU to satisfy the real time application. Aiming at the compute-intensive character of image edge detection, this paper analyzed some methods of edge detection based on CPU, using the programmer friendly CUDA framework, and proposed a method based on GPU, to realize the image edge detection. The efficient architecture of GPU was introduced firstly. Then, two representative image edge detection algorithms, Roberts and Sobel, were implemented on GPU. At last, using the same market price level CPU and GPU as hardware platform, and using various resolution images as test data, compared the computational efficiency of GPU and CPU. Numerical experiments show that the speed of the algorithm can be improved by up to more than 17 times compared with CPU-based implementations, with the same processing results. It proves that the GPU is practical for some applications of image processing.

Keywords Image processing, Edge detection, GPU, CUDA

边缘检测技术是图像处理和计算机视觉等领域最基本的技术,也是图像处理第一个基本的处理步骤。通过边缘检测可以保留物体边界形态的结构信息,极大地降低图像处理的数据量,从而简化图像的分析过程。如何快速而精确地提取图像的边缘信息,一直是国内外研究的热点[1,5]。

近年来,GPU中加入了可编程内部处理器,使GPU具备很高的运算性能。GPU凭借多核共同驱动及很高的内存读写带宽,使其不仅能进行图形处理,在通用计算方面也可以提供更多的运算资源。目前,GPU已经广泛应用到通用信号处理、物理仿真到金融数据分析以及生物科学等领域,并取得了令人瞩目的成果。在分子动力学领域,Geforce 8800GTX获得了较CPU快240倍的运算速度;在模拟脑电波、视觉和嗅觉计算领域,CPU与GPU协同工作获得了130倍速度的性能提升;在地理信息系统(GIS)处理中,有了GPU的帮助,从前需要20min才能完成的运算现在只需30s即可完成,而从前需要30s到40s完成的运算现在能够实现实时运算[3]。

GPU在计算高度数据并行任务时才能发挥作用。在这类任务中,需要处理大量的数据,而对这些数据进行的处理则基本相同。边缘检测本质上是一种局部的邻域算法,易于并行化,因此可考虑利用GPU的独特体系结构来实现边缘检测。本文以Roberts和Sobel这两个有代表性的算子为基础,充分利用GPU的并行处理优势,实现了图像边缘检测算法向GPU上的移植,大幅提升了边缘检测的效率。

本文第1节介绍了GPU通用计算的体系结构基础;第2节给出了基于GPU的边缘检测实现;第3节分析了实验结果。

到稿日期:2009-02-25 返修日期:2009-04-30 本文受863国家重点基金项目(2007AA703104)资助。
张 楠(1984—),女,硕士生,主要从事基于GPU的图像处理研究,E-mail:zhangn599@163.com;王建立(1971—),男,研究员,主要从事光电跟踪控制技术研究;王鸣浩(1981—),男,博士生,主要从事图像处理研究。

1 GPU 软硬件特性分析

近年来，以游戏加速和图形处理为初衷设计的 GPU 以超出摩尔定律的速度高速发展，并开始在非图形的高性能计算领域被大量使用，成为绝对的计算主力。图 1 为近年来 GPU 与 CPU 的浮点运算速度对比。

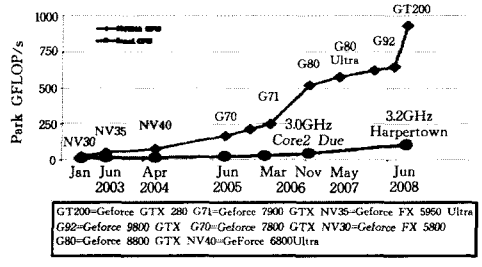


图 1 GPU 和 CPU 的 FLOPS 理论峰值[3]

1.1 GPU 体系结构

GPU 通过单指令多数据(SIMD)指令类型来支持数据并行计算。如图 2 所示，在 SIMD 结构中，单一控制部件向每条流水线分派指令，同样的指令处理部件同时执行。例如 NVDIA 8800GT 中包含有 14 组多处理器(Multiprocessor)，每组多处理器有 8 个处理器(Processor)，但每组多处理器只包含一个指令单元(Instruction Unit)。从线程角度讲，每个多处理器可并行运行 768 个活动线程，即包含 14 组多处理器的 GPU 可并行运行 10752 个活动线程。从存储器角度讲，每个多处理器有 16kB 可读写共享内存、8kB 只读常量内存 Cache、8kB 只读纹理内存 Cache 和 8192 个 322bit 寄存器[4]。

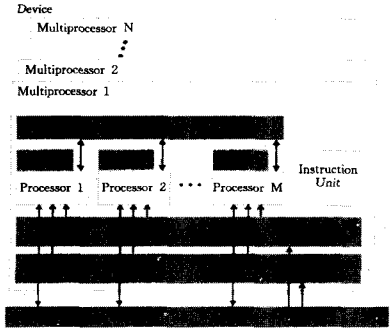


图 2 硬件模型[4]

1.2 CUDA 编程模型

一直以来，利用 GPU 进行通用计算编程受到两大因素的制约：1)必须通过图形 API 完成，GPU 的功能极大地受限于这些图形 API；2) 传统 GPU 仅能进行任意的存储器读操作，缺乏灵活的存储器操作。

CUDA 的推出弥补了这些不足：1) 提供了硬件的直接访问接口，不必依赖图形 API 对 GPU 进行访问；2)采用 C 语言作为编程语言，提供大量的高性能计算指令开发能力；3) 引入了片内共享存储器，支持随机写入。

CUDA 的基本思想是尽量开发线程级并行(Thread Level Parallel)，这些线程能够在硬件中被动态地调度和执行。CUDA 编程模型的重点是将 CPU 作为终端(Host)，而 GPU 作为协处理器(Coprocessor)或设备(Device)，从而让 GPU 来运行一些能够被高度线程化的程序。

运行在 GPU 上的程序称为核 (Kernel)，核以网格(Grid)的形式执行，不同的网格则可以执行不同的核；每个网格由若干个线程块(block)组成，每一个线程块又由最多 512 个线程(thread)组成。属于同一线程块的线程拥有相同的指令地址，能够并行执行，并且能够通过共享存储器(Shared memory)进行线程块内通信。线程块的执行没有顺序，完全并行。

如图 3 所示，在 CPU 上由循环实现的重复操作在 CUDA 中以网格(grid)的形式被分割为多个大小相等的线程块(block)。

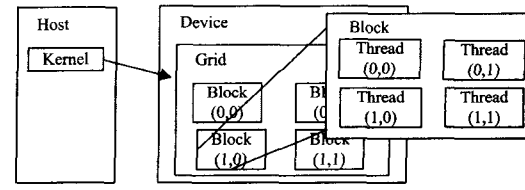


图 3 线程划分

如图 4 所示，CUDA 程序是 CPU 和 GPU 的混合代码，经过 NVIDIA C 编译器编译后，GPU 和 CPU 的代码将被分离，GPU 代码被编译成 GPU 计算的机器码，而 CPU 的 C 代码输出由标准的 C 编译器进行编译。CUDA 所提供的 FFT (Fast Fourier Transform)和 BLAS(Basic Linear Algebra Subprograms)数学函数库，可以简化程序编写。

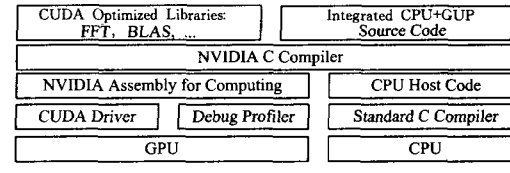


图 4 CUDA 架构

2 基于 GPU 的边缘检测算法实现

2.1 边缘检测实现原理

图像的边缘在图像中表现为局部范围灰度的突变(不连续性)。边缘检测算法是对原始图像按像素的某邻域构造边缘检测算子。大多数检测算法使用基于方向导数掩模求卷积的方法对每个像素的邻域进行检查，并对灰度变化率进行量化，也包括方向的确定。常用的边缘检测算子有 Roberts，Sobel，Prewitt，Kirsch 及 LOG 等[1,5]。

本文通过 Roberts 和 Sobel 算子实现边缘检测。Roberts 算子采用的是对角方向相邻的两个像素之差近似梯度幅值检测边缘，算子如下：

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

Sobel 算子在边缘检测中，对像素位置的影响做了加权，算子如下：

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2}$$

2.2 计算分配

由于 GPU 的数据处理能力远高于 CPU，且直接负责显示工作，在与显示相关的算法设计中应尽可能地把计算量分配到 GPU 端进行并行处理后直接绘制，CPU 只负责向显存装载数据。在与显示无关的通用算法设计中，GPU 端并行计算数据并得到计算结果后，将数据传回主存，如图 5 所示。

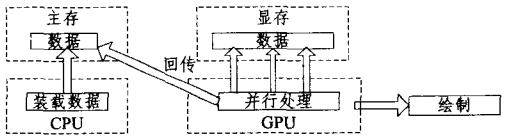


图 5　CPU 和 GPU 的计算分配

2.3　实现过程

本文实验方案的主要步骤如下：

(1)读入图像数据，按照图像的大小在 GPU 上分配设备内存 A，把数据传到设备内存 A；

(2)设定 GPU 的执行参数，即 grid 和 block 的维度；

(3)将图像数据与 GPU 的纹理内存绑定，然后建立与 GPU 内存的映射；

(4)在 OpenGL 中开辟 GPU 程序运行所需的缓冲区，完成 GPU 与 OpenGL 的连接；

(5)运行 kernel 函数对图像进行边缘检测；

(6)OpenGL 显示处理结果。

方案流程图如图 6 所示。

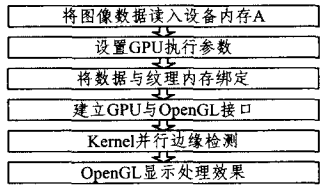


图 6　方案流程图

定义一个 kernel 函数要用到“__global__”限定符，而每次调用所并行执行的线程数量用扩展的“〈〈〈 〉〉〉”语法来指定。在程序中 kernel 被声明为：

EdgeDetect〈〈〈imageH，64〉〉〉(imageIn，imageW，imageW，imageH)　　(3)

其中，“〈〈〈〉〉〉”中参数表示分配了 imageH 个线程块，每个线程块中包含 64 个活动线程。imageIn 为要处理的图像数据起始地址指针，imageW 为图像宽度，imageH 为图像高度。

线程块的大小声明为图像高度 imageH，则每一个线程块负责一行像素的边缘检测运算；每一个线程块中包括线程数为 64 个(GPU 特性决定，线程块大小为 32 的整数倍时 GPU 运算效率相对较高)，则每一个线程负责 imageW/64 个像素点的卷积运算，这样就有 64×imageH 个独立的线程并行执行式(3)。

3　实验结果与分析

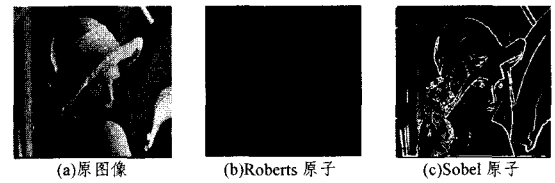
在 CPU 和 GPU 上分别运行 Roberts，Sobel 边缘检测算法，其中 CPU 实现根据算法步骤直接编程得到，并未对其进行任何优化，程序运行环境如表 1 所列。

表 1　实验平台参数

CPU(Intel Pentium Dual E2160)		GPU(GeForce 8800GT)	
主频	1.8GHz	主频	1.5GHz
L2 Cache	1024kB	处理器组数	14
内存	1.0 GB	设备内存	512MB

编译环境：Visual Studio 2005

取 LENA 图像作为实验图像，如图 7(a)所示。Roberts 算子与 Sobel 算子处理效果如图 7(b)、图 7(c)所示。



(a)原图像　(b)Roberts 原子　(c)Sobel 原子

图 7　处理效果图

使用 CUDA 自带的测定时间函数可以得到程序运行的时间，由于得到的时间只能为毫秒，因此要采用多次(1000 次或 100 次)反复处理后求平均时间的方式来得到。不同大小图像数据的 CPU 运算时间、GPU 运算时间及加速比如表 2、表 3 所列。

表 2　Roberts 算子耗时测试结果

图像尺寸	128×128	256×256	512×512
CPU(ms)	0.172	0.711	2.897
GPU(ms)	0.055	0.077	0.169
加速比	3.127	9.233	17.172

表 3　Sobel 算子耗时测试结果

图像尺寸	128×128	256×256	512×512
CPU(ms)	0.210	0.815	3.615
GPU(ms)	0.058	0.091	0.204
加速比	3.621	8.956	17.721

从表中可以看出，相对于 CPU，GPU 的处理速度有显著提升，平均加速比为 9.884 和 10.099；同时，随着数据量和运算量的增加，加速比也在增加，GPU 更能显出其在运算速度上的优势。

结束语　本文实现了基于 GPU 的边缘检测算法，并对实验结果进行了分析。结果表明，与 CPU 相比，GPU 依据其硬件结构先天的并行计算特点，在运行以边缘检测为代表的可分割为独立单元的算法中能够得到极高的效率提升。在图像数据量增大时，这种提升更加明显。在本实验平台上，GPU 运算获得了最高 17 倍和平均 10 倍的提升，显示出 GPU 在通用高密集数据运算方面具有较高的运算能力，可为目标快速检测与跟踪提供硬件支持。

参 考 文 献

[1]　章毓晋. 图像工程(上册)，图像处理(第二版)[M]. 北京：清华大学出版社，2006

[2]　Pharr M. GPU gem2[M]. Boston：Addison-Wesley，2005

[3]　Nvidia Official Site[DB/OL]. http://www.nvidia.com/

[4]　NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide(Version 1.1)[M]. 2007

[5]　段瑞玲，李庆祥，李玉和. 图像边缘检测方法研究综述[J]. 光学技术，2005，31(3)：415-419

[6]　田玲. 图像边缘检测中并行算法的应用与研究[D]. 成都：电子科技大学，2006：45-57

(上接第 238 页)

[9]　Han J C，Hu X H，Lin T Y. A New Computation Model for Rough Set Theory Based on Database Systems[C]//Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery. Prague，Czech Republic，2003：381-390

[10]　王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2003