基于图形处理器的模糊C均值聚类分割算法

刘 刚[1,2] 梁晓庚[3] 贺学剑[4]

(西北工业大学自动化学院 西安 710072)[1] (河南科技大学电子信息工程学院 洛阳 471003)[2]

(洛阳光电技术发展中心 洛阳 471009)[3] (河南科技大学林业职业学院 洛阳 471002)[4]

摘 要 针对模糊C均值聚类图像分割算法运算量大、难于实时处理的问题,提出了一种基于图形处理器的加速算法。通过分析模糊C均值聚类算法各阶段可以并行处理的运算部分,利用计算统一设备架构软硬件结构,分别将隶属度矩阵计算、聚类中心计算和像素按隶属度归类3个部分改造成适合图形处理器硬件并行运行的形式。实验结果表明,相对于CPU串行算法,基于图形处理器的加速算法效率提升明显。鉴于大多数图像处理算法均具有可并行处理的部分,利用图形处理器进行加速具有普适性。

关键词 模糊C均值聚类,图像分割,图形处理器,计算统一设备架构

中图法分类号 TP391.41 **文献标识码** A

Graphics Processing Unit Based Fuzzy C-means Clustering Segmentation

LIU Gang[1,2] LIANG Xiao-geng[3] HE Xue-jian[4]

(Department of Automatic Control, Northwestern Polytechnology University, Xi'an 710072, China)[1]

(Department of Electronics and Information, Luoyang 471003, China)[2]

(Luoyang Optoelectro Technology Development Center, Luoyang 471009, China)[3]

(Henan University of Science and Technology of Forestry Vocational College, Luoyang 471002, China)[4]

Abstract In order to accelerate the segmentation algorithm of FCM(fuzzy c-means clustering), an accelerating algorithm based on GPU(graphics processing unit) was proposed. Firstly, this method analyses the various phases of FCM algorithm which could be paralleled. Then, in order to adapt to the GPU's hardware architecture, this method transforms the computing of membership grade and clustering center and the classifying of every pixels according to the membership grade with CUDA(Compute Unified Device Architecture). Experimental results show that the efficiency of the FCM segmentation algorithm accelerated by GPU is improved obviously compared with CPU's serial algorithm. In view of the parallel features of most image processing algorithms, the acceleration based on GPU is universal.

Keywords Fuzzy C-means clustering, Image segmentation, Graphics processing unit, Compute unified device architecture

1 引言

模糊C均值聚类[1](Fuzzy C-Means clustering, FCM)是一种较为重要的模糊聚类算法,已经广泛用于图像分割领域。利用FCM算法进行分割具有一定的优越性。该方法没有将像素点硬分,而是将像素点按对类别的隶属程度进行分类。但是,图像分割运算量大,使得基于串行方式运行的FCM算法难以满足实时计算的要求,限制了其应用场合。

图像处理的本质是大规模矩阵运算,特别适合并行处理,但CPU通用计算很难利用该特性。近年来出现的图形处理器(Graphics Processing Unit, GPU)为图像处理算法的加速提供了硬件支持。与CPU不同,GPU是一个并行的向量处理器,以一个程序多次数据的模式工作,在并行数据运算上具有强大的能力。GPU除了用于3D图形处理外,其应用领域已扩展到了几何造型[2]、数值计算[3]、流体模拟[4]、三维重建[5]、场景绘制[6]、数据库操作[7]等通用计算领域。传统的GPU通用计算方法是通过OpenGL等一类现有的图形函数库来编写渲染程序,将通用计算转化为图形计算,其程序编写较为繁琐,对开发人员的专业知识要求较高。2007年,NVIDIA公司发布了全新的开发环境:计算统一设备架构(Compute Unified Device Architecture, CUDA),使得GPU打破图形语言的局限,成为真正的并行数据处理的超级计算机,极大扩充了GPU通用并行计算的应用领域。

本文以FCM算法为研究对象,充分利用GPU的并行处理优势和CUDA提供的通用API,实现了基于CPU的FCM图像分割算法向GPU的移植,大幅提升了图像分割的效率。

2 FCM分割算法原理

FCM分割算法的目标函数可表示如下:

$$J=\sum_{i=1}^{c}\sum_{j=1}^{n}u_{ij}^{m}d_{ij}^{2} \tag{1}$$

式中,c是聚类数目,n表示图像的像素个数,u_{ij}表示像素j属

到稿日期:2011-02-23 返修日期:2011-05-26

刘 刚(1974-),男,博士生,讲师,主要研究方向为图形图像处理、智能控制、软件工程,E-mail:lg19741011@163.com;梁晓庚(1960-),男,研究员,博士生导师,主要研究方向为控制与仿真、飞行器设计、图像处理;贺学剑(1975-),男,讲师,主要研究方向为计算机应用。

于类 i 的概率(即隶属度)，m 为模糊加权指数，d_{ij} 是像素 x_j 与聚类中心 x_i' 之间在灰度意义上的绝对值距离：

$$d_{ij}=||x_j-x_i'|| \tag{2}$$

隶属度 u_{ij} 的约束条件为：

$$\sum_{i=1}^{c}u_{ij}=1,\forall j=1,2,\cdots,n \tag{3}$$

FCM 聚类算法可以看作是在约束条件(3)下，通过确定各个隶属度和聚类中心，求取目标函数的极小值。采用拉格朗日乘数法，在约束条件下，令：

$$F=\sum_{i=1}^{c}\sum_{j=1}^{n}u_{ij}^{m}d_{ij}^{2}+\sum_{j=1}^{n}\lambda_j(\sum_{i=1}^{c}u_{ij}-1) \tag{4}$$

对 u_{ij} 和 x_i' 求偏导数，并令其等于零：

$$\frac{\partial F}{\partial x_i}=0,\frac{\partial F}{\partial u_{ij}}=0 \tag{5}$$

解上面的方程组可以得到使式(4)达到最小的必要条件：

$$x_i'=\frac{\sum_{j=1}^{n}u_{ij}^{m}x_j}{\sum_{j=1}^{n}u_{ij}^{m}},u_{ij}=\frac{d_{ij}^{2/1-m}}{\sum_{k=1}^{c}d_{kj}^{2/m-1}} \tag{6}$$

3 基于 CUDA 的通用计算模型

可以认为 CUDA 由两部分组成：一部分是硬件驱动程序；另一部分是在不同层次的软件层面。它以扩展的 C 语言形式，向编程者提供的一套直接面向 GPU 编程的接口。CUDA 软件结构如图 1 所示。

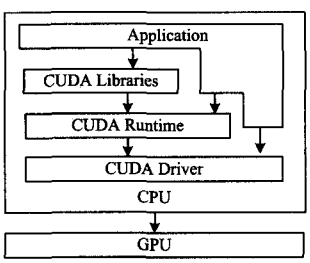

图 1 CUDA 软件结构

用 CUDA 编程实现的时候，GPU 可看作一个可以并行处理很多线程(Thread)的运算设备，可将程序的数据并行和密集运算部分分配给 GPU 设备来处理[8]。更准确地说，如果一个程序的某个函数对于不同的数据要重复执行多次，就可以把这部分函数独立出来，对应到 GPU 的许多不同的 Thread 来执行。编译器将这个函数编译为在 GPU 上执行的指令集，该目标程序称作核(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，每一个线程块最多由 512 个 thread 组成。

CUDA 在主程序初始化核心程序时，需指定线程块数、线程数等参数。各线程根据其线程 ID 来定位和执行数据。在 GPU 上基于 CUDA 的通用计算模型步骤如下：

步骤 1 将待处理的数据装入显存；

步骤 2 CPU 通过通用 API 向 GPU 装入核心程序；

步骤 3 GPU 并行运行核心程序，访问显存，对数据进行并行计算；

步骤 4 可返回步骤 2 进行多遍计算；

步骤 5 显示并导出显存的结果。

4 基于 GPU 的 FCM 算法

考虑到 GPU 和主机之间带宽有限，数据在二者之间的传输代价将非常昂贵，因此应尽可能地在 GPU 上实现整个算法，以减少数据在二者之间的传输。

CUDA 根据 GPU 内存类型的不同，对线程内存访问模式进行了划分[8]，如表 1 所列。

表 1 内存模式

内存类型	速度	存取方式	访问对象
local	慢	读写	一个线程
shared	快	读写	块内所有线程
global	慢	读写	所有线程＋主机
constant	慢	读	所有线程＋主机
texture	慢	读	所有线程＋主机

针对内存的这些特点，本文算法先将数据分解成能装入共享内存(shared)的数据块，并将数据从全局内存(global)载入共享内存，然后从共享内存载入数据，进行并行处理。最后，将处理结果从共享内存批量写回全局内存。

在 FCM 聚类算法中，由式(6)可看出，计算每个像素相对于当前各个样本中心隶属度的方法是相同的，即主要是通过计算每个样本点与中心点的距离来获得隶属度，而处理的数据不同且相互独立，这符合 GPU 并行处理的条件。得到隶属度矩阵后，新聚类中心的计算也具有针对不同数据进行相同处理的特点，同样适合 GPU 处理。当 FCM 迭代达到指定的精度，需要对像素按照隶属度进行归类时，显然每个像素所经历的处理是相同的，可利用 GPU 的并行特性来进行加速。

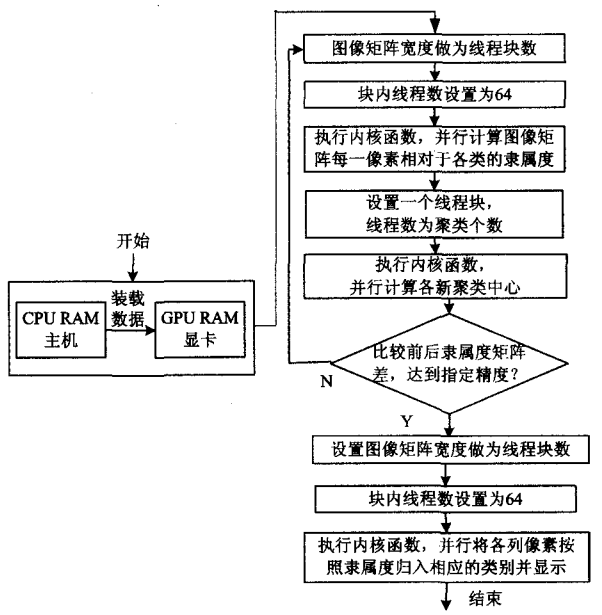

图 2 基于 GPU 加速的 FCM 算法流程图

假定 *width* 和 *height* 分别为图像宽度和高度，首先计算隶属度矩阵。每个线程块使用的线程数设置为 64，每个线程块负责一列像素，则每个线程计算 *height*/64 个像素点的隶属度，线程块数为图像宽度 *width*。当然，每个线程块也可负责一行像素点的隶属度计算。核心函数按式(6)计算，一个像素对应于各个聚类中心的隶属度，声明如下：

ComMembership(〈〈〈*width*,64〉〉〉(*data*,*data2*,*data3*)

〈〈〈〉〉〉中参数表示定义线程块数为 *width*，每个线程块 64 个线程，()中定义了核心函数参数，*data* 表示图像数据起始指针，*data2* 代表隶属度数据指针，*data3* 为聚类中心数据指针。核心装入 GPU 后，共有 *width* * 64 个线程运行该函数。

(下转第 294 页)

tional Partitioning[C]// IEEE/ACM Int. Symp. System Synthesic. New York, USA, 1995: 28-33

[10] Vahid F, Gajski D D, Gong J. A Binary-constraint Search Algorithm for Minimizing Hardware During Hardware/software Partitioning[C]// IEEE/ACM European Design Automation Conference (EDAC). CA, USA, 1994: 214-219

[11] Quan G, Hu X, Greenwood G W. Preferencedriven Hierarchical Hierarchical Hardware/software Partitioning[C]// IEEE Int. conf. on Computer Design. Austin, Texas, USA, 1999: 652-657

[12] Ernst R, Henkel J, Benner T. Hardware-Software Cosynthesis for Micro-controllers[J]. IEEE Design and Test of Computer, 1993, 10(4): 64-75

[13] Wiangtong T, Cheung P Y K, Luk W. Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware-software Codesign[J]. Design Automation for Embedded Systems, 2002, 6: 425-449

[14] Chatha K S, Vemuri R. MAGELLAN: Multiway hardware-software partitioning and scheduling for latency minimization of hierarchical control-dataflow task graphs [C] // CODES. New York, NY, USA, 2001: 42-47

[15] Wu Ji-gang, Srikanthan T. Algorithmic Aspects of Hardware/Software Partitioning: 1D Search Algorithms[J]. IEEE Transactions on computers, 2010, 59(4): 532-544

[16] Toth M S P. Knapsack Problems: Algorithms and Computer Implementations[M]. New York, NY, USA: John Wiley & Sons, 1990: 52-200

[17] Glover F, Laguna M. Tabu Search[M]. Boston: Kluwer Academic Publishers, 1997: 10-150

(上接第 286 页)

在得到隶属度矩阵的基础上，重新在 GPU 中申请一个线程块，块中线程数为聚类数目。按式(6)计算各个新聚类中心，核心函数声明如下：

ComCenter<<<1, *number*>>>(*data*, *data2*, *data3*)

number 为聚类数目。迭代计算隶属度矩阵和各新聚类中心，直到满足所要求的精度。最后，在 GPU 中重新分配线程块及线程数，编写新的内核函数。线程块执行该函数，将每列像素按照隶属度归入相应的类别，声明如下：

Classify<<<*width*, 64>>>(*data*, *data2*)

基于 GPU 加速的 FCM 算法流程如图 2 所示。

5 实验与分析

实验的硬件平台：CPU 为 Pentium4 2.4G；2G 内存；显卡为 Geforce 8800GT。软件平台：Visual Studio 2003。实验均使用红外图像，如图 3 所示。其中，海洋人和马图像大小为 256 * 256，飞机爬升和飞机尾后图像大小为 128 * 128。处理时间比较如表 2 所列。分割结果如图 4 所示。对于 FCM 算法所需要的初始聚类数目，根据图像特点并考虑灰度因素，海洋人图像分为天空、人与摩托艇、深色海水和浪花 4 类；马图像分为道路、草地、树与栅栏、马匹 4 类；飞机图像分为机身、尾焰和天空 3 类。从表 2 中可看出，基于 GPU 的 FCM 分割算法效率改进明显。对于 256 * 256 图像，平均加速比为 10.489。当图像大小为 128 * 128 时，平均加速比达到 4.084。随着数据量的增大，基于 GPU 的算法获得了更大的速度提升。从图 4 可看出，马、飞机爬升和飞机尾后得到了期望的分割结果。而对于海洋人图像，将人物、摩托艇与天空划分为一类，未达到期望效果，说明将 FCM 算法用于图像分割还需要进行改进。

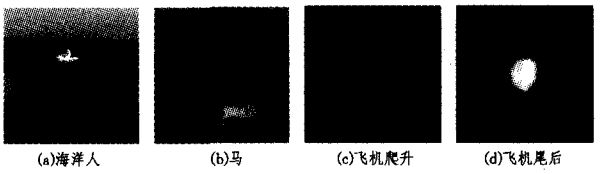

图 3 红外原图像

表 2 基于 GPU 与 CPU 的 FCM 算法执行时间比较

时间(s)	海洋人 4 类	马 4 类	飞机爬升 3 类	飞机尾后 3 类
串行算法	6.4	3.4	0.258	0.125
并行算法	0.605	0.327	0.052	0.039
加速比	10.579	10.398	4.962	3.205

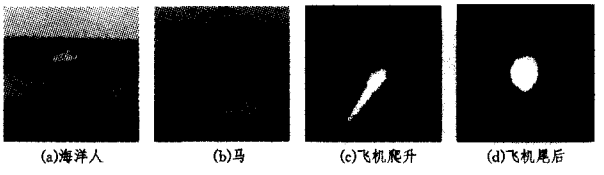

图 4 FCM 分割效果图

结束语 本文实现了基于 GPU 的 FCM 图像分割算法，并对实验结果进行了分析。结果表明，通过分析 FCM 算法各阶段不同数据相同处理的特性，利用 GPU 硬件结构先天的并行计算特点，将 FCM 算法改造成适合 GPU 运行的形式。与 CPU 串行算法相比，本文方法获得了明显效率提升。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 算法获得了平均 7 倍的提升。鉴于大多数图像处理算法均具有大量数据进行相同运算的特性，利用 GPU 进行加速具有普适性。

参 考 文 献

[1] Bezdek J. Pattern Recognition with Fuzzy Objective Function Algorithms[M]. New York: Plenum Press, 1981

[2] Guthe M, Balázs á, Klein R. GPU-based Trimming and Tessellation of NURBS and T-spline Surfaces [J]. ACM Transaction on Graphics, 2005, 24(3): 1016-1023

[3] Thompson C J, Hahn S, Oskin M. Using modern graphics architectures for general-purpose computing: A framework and analysis[A]// Proceedings of International Symposium on Microarchitecture[C]. Istanbul, 2002: 306-317

[4] 周世哲，满家巨. 基于多重网格法的实时流体模拟[J]. 计算机辅助设计与图形学学报，2007，19(7)：935-940

[5] 韩峰. 基于 GPU 的有限角度投影数据的 CT 三维重建[D]. 上海：上海交通大学，2008

[6] Govindaraju N K, Henson M, Lin M C, et al. Interactive Visibility Ordering and Transparency Computations Among Geometric Primitives in Complex Environments [C]// Proc. of ACM Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2005

[7] 杨珂. 基于图形处理器的数据管理技术研究[D]. 浙江：浙江大学，2008

[8] NVIDIA Corporation. CUDA Programming Guide 1.0 [EB/OL]. http:// www. nvidia. com, 2007