基于 CUDA 的大型实对称矩阵并行求逆算法

霍迎秋，王武星，彭楚风，方 勇

（西北农林科技大学 信息工程学院，陕西 杨凌 712100）

摘 要：针对大型实对称矩阵数值求逆算法运算量大、计算时间长的问题，分析分块迭代求逆算法的并行性，设计基于 CUDA（compute unified device architecture）的并行求逆算法。设计对比分析实验，实验结果表明，该算法能够提高大型实对称矩阵求逆的速度，当矩阵大小为 8000×8000 时，加速比高达 279 倍，很好满足了实际工程中对实时性要求高的需求，且计算精度基本保持不变。

关键词：实对称矩阵；分块迭代求逆；图形处理单元；统一计算设备架构；并行算法

中图法分类号：TP311 **文献标识号**：A **文章编号**：1000-7024（2015）08-2133-05

doi：10.16208/j.issn1000-7024.2015.08.025

CUDA-based parallel matrix inverse algorithm of large-scale real symmetric matrix

HUO Ying-qiu, WANG Wu-xing, PENG Chu-feng, FANG Yong

(College of Information Engineering, Northwest A and F University, Yangling 712100, China)

Abstract: Some problems including the huge amount of computations and time-consumption exit in the process of computing the inverse matrix of a large real symmetric matrix using block-based iterative inversion algorithm. To solve these problems, a parallel analysis of the iterative inversion algorithm was made, and then it was optimized based on CUDA. The comparison between the sequential algorithm and the parallel one was designed. The experimental result shows that the parallel algorithm is able to improve the speed of the algorithm greatly. When the size of matrix is 8000×8000, the speedup is 279 times. Hence this parallel algorithm can meet the high real-time requirement well in actual applications, while maintaining almost the same accuracy.

Key words: real symmetric matrix; block iterative inverse; GPU; CUDA; parallel algorithm

0 引 言

矩阵求逆被大量应用于卫星导航定位[1]、雷达脉冲压缩[2]、图像处理[3]及预测控制[4]等许多工程技术领域。目前主要的矩阵求逆算法如 QR 分解法[5,6]、Jacobi 数值法[7]、LU 分解法[8,9]、极小多项式求逆[10]等算法对大型实对称矩阵求逆时，运行速度慢，不能满足实际工程中对实时性要求高的需求。因此，针对大型矩阵设计并行求逆算法，在保证精度满足要求的前提下，提高算法的运行速度，在实际工程具有重要的实际应用意义。

1 分块迭代求逆算法

对大型实对称矩阵求逆，可用分块迭代法实现，其原理如下：

将一个 n 阶实对称矩阵 $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ 分块，设分块形式为

$$\boldsymbol{W}_t = \begin{bmatrix} \boldsymbol{W}_{t-1} & \boldsymbol{r}_t \\ \boldsymbol{r}_t^T & \rho \end{bmatrix} \tag{1}$$

式中：$\boldsymbol{W}_{t-1}$ ——矩阵 $\boldsymbol{A}$ 的前 $t-1$ 阶方阵，$\boldsymbol{r}_t^T$ —— A 矩阵的第 t 行的前 $t-1$ 个元素，$\boldsymbol{r}_t$ —— $\boldsymbol{r}_t^T$ 的转置，ρ ——$\boldsymbol{A}$ 矩阵第 n 行第 n 列元素。

根据分块迭代求逆算法利用 $\boldsymbol{W}_{t-1}^{-1}$ 递推 $\boldsymbol{W}_t^{-1}$，令

$$\boldsymbol{q}_t = -\boldsymbol{W}_{t-1}^{-1}\boldsymbol{r}_t \tag{2}$$

$$\beta_t = \rho_t + \boldsymbol{r}_t^T\boldsymbol{q}_t \tag{3}$$

则 n 阶实对称矩阵的逆矩阵 $\boldsymbol{W}_t^{-1}$ 可用 $\boldsymbol{W}_{t-1}^{-1}$、$\boldsymbol{q}_t$、$\beta_t$ 表示如下

收稿日期：2014-08-18；**修订日期**：2014-10-20

基金项目：国家自然科学基金项目（61271280）；国家级大学生科技创新基金项目（201410712095）

作者简介：霍迎秋（1978-），男，河北唐山人，博士，实验师，研究方向为压缩感知、并行计算；王武星（1993-），男，陕西铜川人，本科，研究方向为并行计算；彭楚风（1993-），女，河南周口人，本科，研究方向为并行计算；方勇（1979-），男，江西抚州人，教授，博士生导师，研究方向为算术码码谱及其应用。E-mail：redline2003@163.com

$$\boldsymbol{W}_t^{-1}=\begin{bmatrix}\boldsymbol{W}_{t-1}^{-1}+\dfrac{1}{\beta_t}\boldsymbol{q}_t\boldsymbol{q}_t^T & \dfrac{1}{\beta_t}\boldsymbol{q}_t\\ \dfrac{1}{\beta_t}\boldsymbol{q}_t^T & \dfrac{1}{\beta_t}\end{bmatrix} \tag{4}$$

当 $t=n$ 时，$\boldsymbol{W}_t^{-1}$ 即为矩阵 $\boldsymbol{A}$ 的逆矩阵。

基于 CPU 实现串行分块迭代求逆算法，流程如图 1 所示。

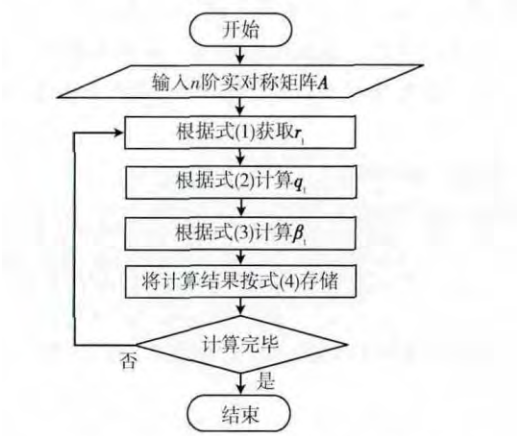

图 1 串行分块迭代求逆算法流程

2 CUDA 架构简介

CUDA 是由 NVIDIA 在 2007 年 2 月推出的一种新的 GPU 架构，使人们可以更加方便的使用 GPU 对其所进行的工作加速，其开发语言与 C、C++非常类似，且不用借助图形 API 等额外操作。CUDA 优异的性能，使得它被推出之后广泛应用于模式识别[11]、视频检测[12]、压缩感知[13]、信号处理[14]、流体力学[15]等领域。

支持 CUDA 的 GPU（graphic processing unit）与传统的 GPU 最大区别在于它拥有片内共享存储空间用以减小片外访问延迟[16]。CUDA 源程序由在主机端（CPU）上运行的控制程序和运行于设备端（GPU）上的核心计算程序（kernel）两部分组成，kernel 在一个线程格（Grid）上执行，每个线程格由大小相同的一组线程块（Block）组成，同一线程块内的所有线程可以共享存储空间用来共同协作完成计算任务，线程块之间不能共享存储空间。每个线程块在一个 SM（streaming multiprocessor）上运行。为了管理同时运行的数百个线程，SM 利用了一种称为 SIMT（single instruction multiple thread）的新架构[16,17]，各线程被 SM 映射到一个 TP（thread processor）核心上独立执行。SIMT 以 32 个线程为一组进行调度，这种线程组被称为 warp 块。

CUDA 硬件模型有多个 SM，每个 SM 又有若干个 SP（stream processor）和若干个 32-bit 的寄存器。此外，模型中还包括所有线程可访问的常量内存和纹理内存，以加速对数据的读取。CUDA 硬件模型如图 2 所示[17]。

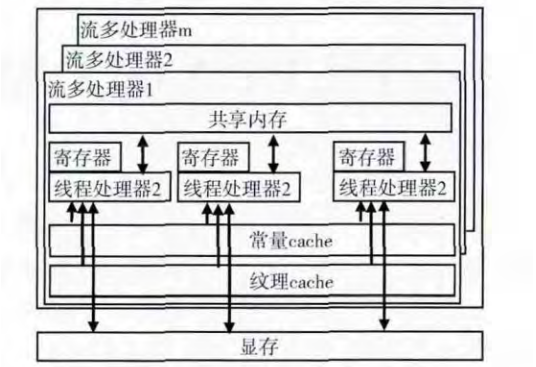

图 2 CUDA 硬件模型

3 基于 CUDA 的并行分块迭代求逆算法

3.1 分块迭代求逆算法并行性分析

分块迭代求逆算法是一个循环迭代计算的算法，各循环之间有着强烈的依赖关系，因此算法不能完全展开使之在 GPU 上运行。但经仔细分析发现，每个循环中包含大量的向量与向量相乘、矩阵与向量相乘、向量数乘和矩阵相加等运算，而迭代算法主要的计算量集中在这些运算之上。并且矩阵运算中各元素独立计算，互相之间没有依赖关系，非常适合基于 GPU 对算法进行并行优化设计。

3.2 迭代求逆算法的并行设计

针对迭代算法每个循环中，行向量与列向量相乘、列向量与行向量相乘、矩阵与向量相乘、矩阵相加和矩阵与向量相乘等部分进行并行优化设计。求逆算法整体流程不变，基于 CUDA 的并行算法流程如图 3 所示。

3.2.1 矩阵与向量相乘

矩阵与相量相乘结果为向量。结果向量的每个元素由矩阵的某行与向量对应项相乘结果之和。设计 kernel 函数：每个线程块计算矩阵的某行与向量对应项相乘，然后进行“块内归约求和”，其结果即为结果向量的一个元素值。主要设计思路如图 4 所示。

归约求和：若对数组 array 归约求和，则每个线程将 array 的两个元素相加，再将结果保存回 array。每个线程都将 array 中的两个元素合并为一个元素，此步骤完成之后，结果的数量就为原来的一半。重复执行上述操作，直至将 array 中所有元素归约为一个值。归约求和方法如图 5 所示。

3.2.2 向量相乘的并行设计

算法中涉及到的向量相乘分为行向量与列向量相乘、列向量与行向量相乘，分别对这两种向量相乘进行并行设计。

（1）行向量与列向量相乘

计算行向量与列向量相乘。首先，计算两向量对应元素乘积，然后对所有乘积求和。设计核函数，计算两向量对应项乘积时为 kernel 函数分配 32 个一维线程块，每个线

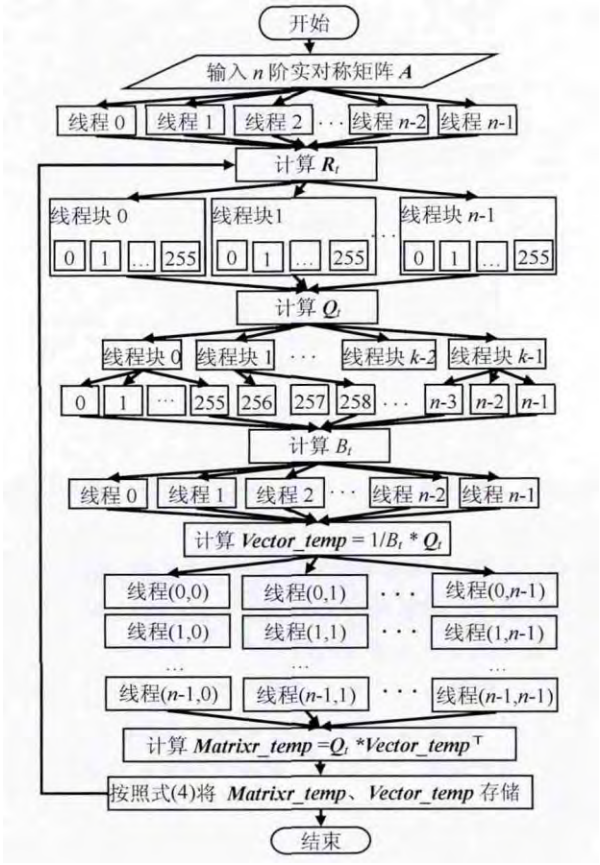

图 3　并行算法流程

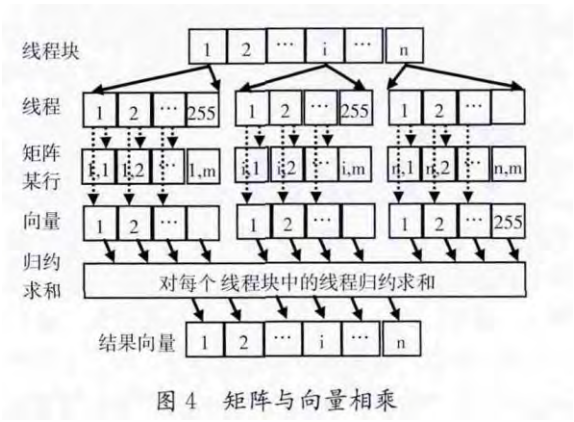

图 4　矩阵与向量相乘

程块分配 256 个线程。每个线程块计算两向量 256 个对应项元素相乘。并进行“块内归约求和”，将求和结果写入显存，同步所有线程块，在显存上进行“二次归约求和”。求和结果即为两向量相乘的结果。设计思路如图 6 所示。

“块内归约求和”与“二次归约求和”设计思路相同，只是计算操作的区域不同，“块内归约求和”是在线程块的共享内存中进行，“二次归约求和”是在显存上进行。归约方法如图 5 所示。

（2）列向量与行向量相乘

列向量与行向量相乘，其结果为一个矩阵，矩阵的每个元素为两向量对应元素的乘积。设计核函数：为 kernel

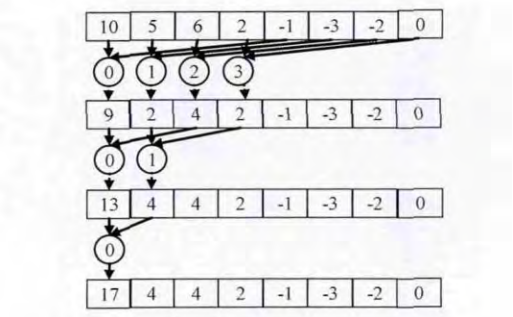

图 5　归约求和

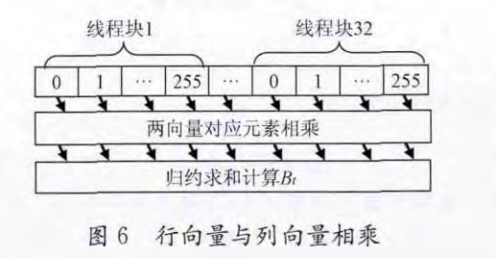

图 6　行向量与列向量相乘

函数分配二维线程块，每个维度大小为（向量长度 n + 15）/16，每个线程块分配 16×16 个线程，线程的索引即为结果矩阵对应元素的索引，使得每个线程计算两个对应项元素相乘，同时将计算结果写入显存。设计思路如图 7 所示。

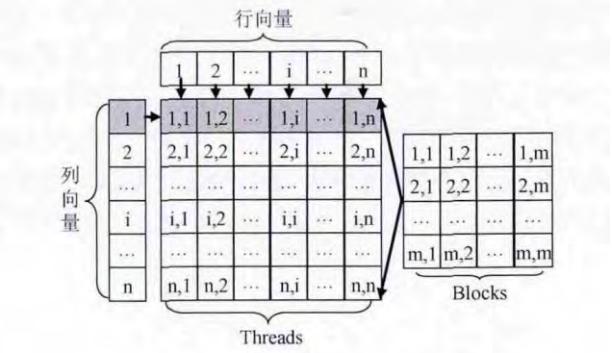

图 7　列向量与行向量相乘

3.2.3　向量的数乘

向量的数乘即为一个数字和某向量的乘积。设计核函数：分配线程个数不少于向量长度，每个线程计算数字与一个向量元素的乘积，将结果写入显存。具体设计方法类似于图 6 归约求和之前部分。

3.2.4　矩阵相加

矩阵相加即为两矩阵相同索引处两元素之和。设计核函数：分配（m，m）个线程块，每个线程块分配（16，16）个线程（m×16 不小于两相加矩阵的任一维度），则对应线程的索引即为两矩阵对应元素的索引，使得每个线程计算两矩阵对应元素之和，将求和结果写入显存。设计思路如图 8 所示。

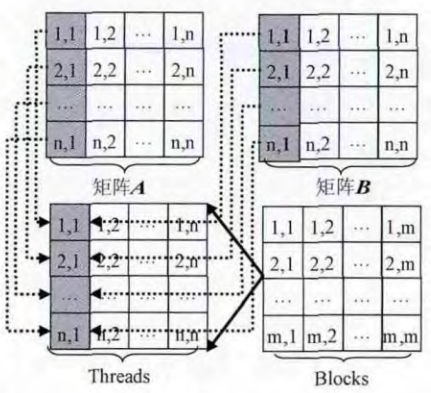

图 8 矩阵相加

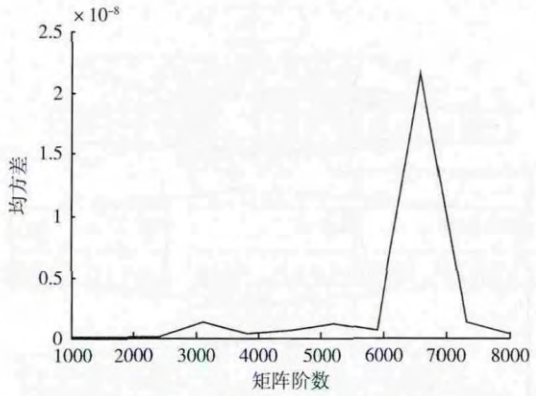

图 10 均方差

4 实验结果与分析

为对比分析并行分块迭代求逆算法的性能，本文基于 C 语言实现串行算法，基于 CUDA 架构设计并行优化算法。硬件环境：Intel(R) Core(TM) 2 Quad CPU Q8400，内存 4GB，GPU 采用 NVIDIA（英伟达）GPU 处理器 GeForce GTX 780，其主要参数：12 个 SM，每个 SM 有 192 个 SP，共享内存 48K，显存 3GB。软件环境：WIN7 旗舰版 32 位操作系统，Microsoft Visual Studio 2010。

设计一个随机生成实对称矩阵函数，生成指定阶数的实对称矩阵，分别用串行求逆算法和并行优化算法对同一个实对称矩阵进行求逆运算，利用 C 语言的 clock() 函数对串行求逆算法计时（单位：ms），利用 CUDA 事件对并行优化算法计时（单位：ms）。计算出并行优化算法相对于串行求逆算法的加速比，计算两种算法求逆结果的均方差，以此判断并行优化算法计算的相对误差。实验结果如图 9、图 10 所示。

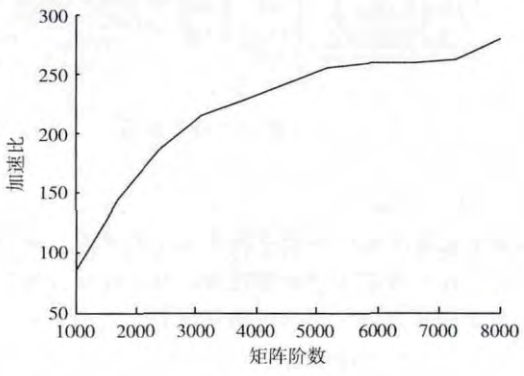

图 9 加速比

通过对图 9 的实验结果分析发现，并行优化算法的运行速度明显要比串行迭代求逆算法的运行速度快，在矩阵阶数为 8000×8000 时，串行分块迭代求逆算法需要运行 88.50 分钟，而并行优算法只需要 19.03 s，与串行求逆算法相比，其加速比可以达到 279，且随着矩阵阶数的增加，其加速比也越来越大，可以很好的满足实际工程中对实时性的要求。

通过对图 10 实验结果分析发现：本文所设计的并行优化算法所用浮点型数据采用单精度浮点型表示时，其与串行求逆算法两者计算结果的均方差在 10^{-8} 级别，均方差的大小也并未随着矩阵阶数的增加而增加。当矩阵阶数更大时，本文所设计并行求逆算法与串行求逆算法计算结果的均方差大概稳定在 10^{-8} 级别，此精度完全可以满足工程的实际需要。

综上所述，本文所设计的并行优化算法与串行求逆算法相比，虽然有些许误差，但是可以满足实际工程对精度的要求，而且大大提高了求逆算法的运行速度，随着矩阵阶数的增加，加速比也越来越大。因此，此并行优化算法具有很高的实际应用价值。

5 结束语

本文简要说明了分块迭代求逆算法的原理，给出了串行求逆算法和并行优化算法的流程图，同时对并行优化算法做了详细的介绍，并基于 C 语言实现了串行分块迭代求逆算法，基于 CUDA 架构设计了并行优化算法。对 10 个随机产生的大型实对称矩阵进行求逆运算，统计了两种算法求逆结果的均方差和并行优化算法相对于串行求逆算法的加速比。实验结果表明，并行优化算法表现出了优异的性能，特别算法是运算时间得到了极大的提高，与串行算法相比，在矩阵阶数为 8000×8000 时，其加速比高达 279，并且随着矩阵阶数的增大，加速比越来越大；而两种算法求逆结果的均方差表明，本文所设计的并行分优化算法可以满足工程实际中对精度的要求。因此，本文设计的并行优化算法在工程实际中具有重要的应用价值。

参考文献：

[1] ZENG Degui. Matrix inversion and its application in beidou double-star positioning system [J]. Information and Computer,2010 (9): 31-32 (in Chinese). [曾德贵. 矩阵求逆及其

在北斗双星定位系统上的应用［J］. 信息与电脑，2010（9）：31-32.］

［2］ SONG Yixin，YAO Zhendong. RMMSE radar pulse compression fast algorithm of matrix inversion of the FPGA implementation［J］. Journal of Chengdu Information Engineering College，2009，24（5）：435-439（in Chinese）.［宋一鑫，姚振东. RMMSE 雷达脉冲压缩快速算法中矩阵求逆的 FPGA 实现［J］. 成都信息工程学院学报，2009，24（5）：435-439.］

［3］ ZHENG Zuoyong，ZHANG Ruixia. A quick method for inversing a circulant matrix on GPU［J］. Engineering and Computer Science，2012，34（7）：84-88（in Chinese）.［郑作勇，张瑞霞. GPU 上循环的快速求逆算法［J］. 计算机工程与科学，2012，34（7）：84-88.］

［4］ SONG Jian. The implementation of the predictive control algorithm based on FPGA［D］. Dalian：Dalian Maritime University，2010（in Chinese）.［宋见. 基于 FPGA 的预测控制算法的实现［D］. 大连：大连海事大学，2010.］

［5］ NI Tao，DING Haifeng，RUAN Liting，et al. Implementation of arbitrary order complex matrix inversion based on the QR decomposition algorithm in DSP［J］. Electronic Science and Technology，2010，23（4）：99-101（in Chinese）.［倪涛，丁海峰，阮黎婷，等. 基于 QR 分解算法的任意阶复矩阵求逆的 DSP 实现［J］. 电子科技，2010，23（4）：99-101.］

［6］ HAN Liangliang，YE Ping. A Jacobi matrix inversion method for redundant robotic manipulator base on QR decomposition［J］. Software，2013，34（11）：64-66（in Chinese）.［韩亮亮，叶平. 基于 QR 分解的冗余度机械臂雅可比矩阵求逆方法［J］. 软件，2013，34（11）：64-66.］

［7］ WANG Jiexian，FENG Baoxin. Application of Jacobi numerical method in inverse problem of real symmetry matrix［J］. Geodesy and Geodynamics，2010，30（1）：74-76（in Chinese）.［王解先，冯宝新. Jacobi 数值方法在实矩阵求逆中的应用［J］. 大地测量与地球动力学，2010，30（1）：74-76.］

［8］ LI Tao，ZHANG Zhongpei. Matrix inversion by FPGA［J］. Communication Technology，2010，43（11）：147-149（in Chinese）.［李涛，张忠培. 矩阵求逆的 FPGA 实现［J］. 通信技术，2010，43（11）：147-149.］

［9］ ZHAO Liqun. The inverse and determinant computation of some sparse matrix［D］. Zhangzhou：Zhangzhou Normal College，2011（in Chinese）.［赵立群. 一些稀疏矩阵的逆和行列式的计算［D］. 漳州：漳州师范学院，2011.］

［10］ YIN Yunxing，ZHAO Qing. Minimal polynomial in the application of matrix inversion［J］. Science，Technology and Engineering，2009，9（5）：1217-1218（in Chinese）.［殷云星，赵清. 极小多项式在矩阵求逆中的应用［J］. 科学技术与工程，2009，9（5）：1217-1218.］

［11］ ZHANG Shu. Pattern recognition parallel algorithm and the GPU implementation study at a high speed［D］. Chengdu：University of Electronic Science and Technology，2009（in Chinese）.［张舒. 模式识别并行算法与 GPU 高速实现研究［D］. 成都：电子科技大学，2009.］

［12］ FU Cheng. Research on video detection algorithm for vehicles red-light running based on GPU［D］. Chengdu：Xihua University，2012（in Chinese）.［付诚. 基于 GPU 的闯红灯车辆视频检测算法研究［D］. 成都：西华大学，2012.］

［13］ Yong Fang，Liang Chen. GPU implementation of orthogonal matching pursuit for compressive sensing［C］//Proc IEEE ICPADS，2011：1044-1047.

［14］ ZHAO Fangbin. Research on passive radar signal processing platform based on FPGA and GPU［D］. Chengdu：University of Electronic Science and Technology，2013（in Chinese）.［赵芳斌. 基于 FPGA 和 GPU 的外辐射源雷达信号处理平台的研究［D］. 成都：电子科技大学，2013.］

［15］ DONG Yanxing，LI Xinliang，LI Sen，et al. Acceleration of computational fluid dynamics codes on GPU［J］. The Computer System Application，2011，20（1）：104-109（in Chinese）.［董延星，李新亮，李森，等. GPU 上计算流体力学的加速［J］. 计算机系统应用，2011，20（1）：104-109.］

［16］ GAN Xinbiao，SHEN Li，WANG Zhiying. Parallelizing full search algorithm for motion estimation using CUDA［J］. Journal of Computer Graphics，2010，22（3）：457-460（in Chinese）.［甘新标，沈立，王志英. 基于 CUDA 的并行全搜索并行估计算法［J］. 计算机图形学学报，2010，22（3）：457-460.］

［17］ NVIDIA. CUDA programming guide 2.0［M］. Santa Clara：NVIDIA Corporation，2008.