第 40 卷 第 11 期 计 算 机 工 程 2014 年 11 月
Vol. 40 No. 11 Computer Engineering November 2014

·开发研究与工程应用· 文章编号：1000-3428(2014)11-0288-04 文献标识码：A 中图分类号：TP311.1

面向高清视频监控系统的实时运动检测算法

彭 爽，蒋荣欣

（浙江大学浙江省网络多媒体技术研究重点实验室，杭州 310027）

摘 要： 针对高清视频监控系统中精确运动检测的高实时性需求，提出一种基于计算统一设备架构（CUDA）的运动检测算法。采用一种改进帧差与背景差分相结合的方法，减少背景更新干扰，提升运动检测的精确性。在 CUDA 内进行视频运动检测计算，避免传统图形处理器硬解码后视频数据在显示内存与 CPU 之间传输的问题。运用块内多线程合并访问共享内存的方式，减少 52.9% 全局内存访问量，解决 CUDA 大规模访问全局内存延迟较大的问题。实验结果表明，该算法在保证准确性的同时，针对高清视频每秒可传输 52.6 帧，能够满足实时性要求。

关键词： 视频监控；运动检测；帧差法；背景差分法；硬解码；合并访问

中文引用格式： 彭 爽，蒋荣欣. 面向高清视频监控系统的实时运动检测算法[J]. 计算机工程，2014，40(11)：288-291.

英文引用格式： Peng Shuang，Jiang Rongxin. Real-time Motion Detection Algorithm for High Definition Video Surveillance System[J]. Computer Engineering，2014，40(11)：288-291.

Real-time Motion Detection Algorithm for High Definition Video Surveillance System

PENG Shuang，JIANG Rong-xin

（Network Multimedia Technology Research Laboratory of Zhejiang Province，Zhejiang University，Hangzhou 310027，China）

【Abstract】 Aiming at high real-time demand for accurate motion detection in high-definition video surveillance system，this paper proposes an efficient motion detection algorithm based on Compute Unified Device Architecture（CUDA）. By using an improved frame difference and background subtraction method of combining，background interference update is reduced and the accuracy of motion detection is enhanced. By performing video motion detection calculations in the CUDA，the traditional Graphic Processing Unit（GPU） hard decoded video data avoid storing in display memory transferred to CPU，which is the bottleneck problem. This paper uses multiple threads within a block coalesced access shared memory，reduces the amount by 52.9% of the global memory access，solves large-scale access to global memory latency CUDA larger problem. Experimental results show that for high-definition video monitoring system，the proposed motion detection algorithm can reach 52.6 frames per second，ensuring the accuracy as well，and can meet the real-time requirements.

【Key words】 video surveillance；motion detection；frame difference method；background subtraction method；hard decode；coalesced access

DOI： 10.3969/j.issn.1000-3428.2014.11.057

1 概述

运动检测是监控领域的一个重要课题。特别是在需要安全防范的场合下，运动检测能在发生异常情况时有效帮助跟踪定位运动目标，自动报警或自动存储视频，是监控系统达到智能化的必要手段[1]。然而，随着高清视频的普及化，视频的解码及运动检测很难达到精确性与实时性兼顾的要求。1080P 分辨率的 H264 视频码流，通常码率为 8 Mb/s，帧率为 25 f/s，意味着监控系统解码的负荷量为 8 Mb/s，运动检测计算的负荷量为 320 Mb/s，对实时精确地检测运动目标构成了极大的挑战[2]。

视频监控系统运动侦测主要包含视频解码和运动检测计算。视频解码分为软解码和硬解码，高清视频的软解码和运动检测对 CPU 性能挑战大，无法达到实时性需求。传统的视频硬解码和运动检测存在以下问题：(1) 运动检测算法主要有光流法、背景差分法、帧差法[3]。这 3 种算法都存在很大的局限性。光流法计算复杂，背景差分法对光线等背景变化敏感，帧差法无法提取完整的运动目标及内部纹理；(2) 解码后的数据存放于显示内存中，而显示内存与主存之间的数据传输速度受限于总线的传输速度[4]。

基金项目： 浙江省级重点科技创新团队基金资助项目(2011R09021-02)。

作者简介： 彭 爽(1989－)，女，硕士研究生，主研方向：网络多媒体，并行编程；蒋荣欣，副研究员。

收稿日期： 2013-11-07 **修回日期：** 2013-12-02 **E-mail：** vivo_peng@126.com

针对上述问题,本文提出基于 CUDA 的硬解码及运动检测算法,具体内容包括:(1) 采用一种改进的帧差与背景差分结合的方法,完整地提取运动目标,有效避免背景误判;(2) 在 CUDA 环境下进行运动检测计算,避免了显示内存与内存传输的瓶颈问题;(3) 针对 CUDA 对全局内存访问延迟大的问题,采用块(block)内多线程共享内存,一个线程处理 4 ×4 个像素,合并访问其邻域,降低运算复杂度。

2　改进前后的算法对比

2.1　传统的运动检测算法

背景差分法通过计算当前帧和设定背景的特征差,进行背景消去,然后与预设阈值作比较。若所得到的像素数大于预设阈值,则判定被监视场景中有运动物体,从而得到运动目标[5]。其优点是原理和算法设计简单,但是对光线变化敏感,同时容易忽略与背景亮度相似的运动物体。帧差法是采用计算相邻帧差绝对值是否大于阈值来检测移动物体。其优点是算法简单,对光线变化不敏感;缺点是无法完整地提取运动目标,且如果选择帧间隔过大则判断缺乏准确,帧间隔小则计算量较大[6]。在运动分量检测后,通常画面的噪点较多,需进行降噪操作以保证检测的精确度[7]。

2.2　改进的运动检测算法

针对背景差分法对背景变化敏感,帧差法无法完整提取运动目标等问题,本文采用一种改进的基于帧差和背景差分结合的运动检测算法。将当前帧与上一参考帧相减,将所得图像与背景差分后图像进行位或运算后可准确提炼出运动目标所在区域。随后针对存在移动物体的 8 ×8 宏块单元进行开运算,去掉毛刺,去掉孤立的噪点像素[8-9]。将运动目标以外的区域更新到背景中,如图 1 所示。

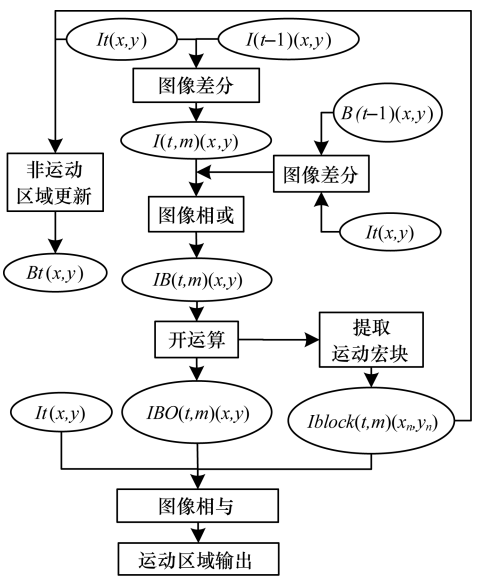


图 1　改进的运动检测算法

在图 1 中,$I_t(x,y)$是当前帧,$I_{(t-1)}(x,y)$是上一参考帧,$I_{(t-1)}(x,y)$为上一次更新过的背景图像,其表达方式为:

$$B_t(x,y) = B_{(t-1)}(x,y) \cap (T_{\text{block}}(t,m)(x_n,y_n) \cap I_t(x,y))(n=8) \tag{1}$$

将当前帧与参考帧进行绝对值减法运算:

$$I_{(t,m)} = |I_t(x,y) - I_{(t01)}(x,y)| \tag{2}$$

再将 $I_{(t,m)}(x,y)$与 $B_{(t-1)}(x,y)$进行差分运算,后得到背景差分后运动区域。将此区域与帧差所得运动区域叠加,可得到完整的运动物体。图像根据 d 阈值进行二值化,将运动区域与非运动区域区分出来:

$$IB_{(t,m)}(x,y) = \begin{cases} M & I_{(t,m)}(x,y) \geqslant d \\ \overline{M} & I_{(t,m)}(x,y) < d \end{cases} \tag{3}$$

此时运动区域尚有图像噪点。利用图像开运算提取运动目标骨干信息,去掉毛刺,减少孤立噪点的影响:

$$IBO_{(t,m)} = Opening(IB_{(t,m)}(x,y)) \tag{4}$$

同时,将运动区域按 8 ×8 宏块存储,若当前区域内有运动像素,则置为 255,反之置为 0:

$$I_{b(t,m)}(x_n,y_n) = \begin{cases} 255 & \sum_{n=8} IBO_{(t,m)}(x,y) \geqslant 1 \\ 0 & \sum_{n=8} IBO_{(t,m)}(x,y) < 1 \end{cases} \tag{5}$$

该算法相比帧差法能更完整地检测到运动目标,并能准确完整地反映运动物体的位置与移动量。同时相比背景差分法降低了光线等因素的影响,可精确地获得运动物体的形状与内部纹理。

3　视频运动检测算法设计与实现

3.1　CUDA 硬解码及运动检测算法

本文选取 NVIDA 公司提供的 CUDA 开发包进行实验。将网络码流解析后组成完整视频帧,送入 GPU 进行解码,解码出来的数据为 NV12 或者 YV12 格式。这 2 种数据格式都属于 YUV 格式,其 Y 分量都是连续的,NV12 的 U,V 分量为交叉分布,YV12 的 U,V 分量为连续分布[10]。而运动检测算法通常只需取像素点的亮度分量,即 Y 分量进行计算。获取 Y 分量后进行运动检测计算,计算完成后将 YUV 数据格式转化成显卡可渲染的 RGB 数据进行显示,同时更新背景帧。通知业务层进行运动侦测的后处理,如自动报警或自动存储等,如图 2 所示。

GPU 解码后的数据存放于纹理内存中,即显示内存[11]。而显示内存与内存间的数据传输速度为当前的难题。基于 CUDA 的运动检测计算是直接从纹理内存读取数据,有效地避免这一瓶颈。

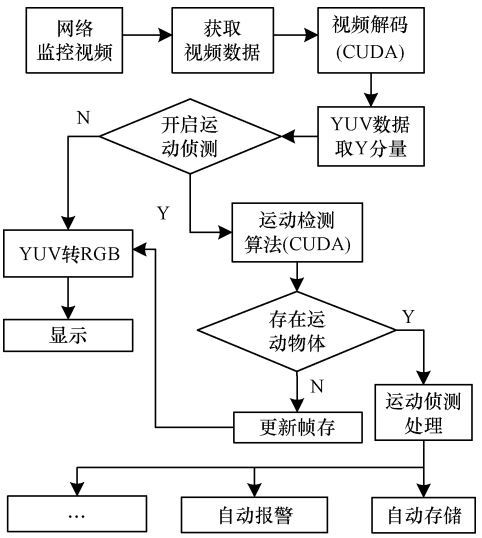


图 2 视频监控系统运动检测算法流程

在 CUDA 中将待检测的 Y 图像分量作帧差与背景差分运算,得到的 2 个运动区域进行叠加,利用开运算将背景中的毛刺去除,提取准确的运动目标。将运动区域以外的图像块更新到背景中,流程如图 3 所示。

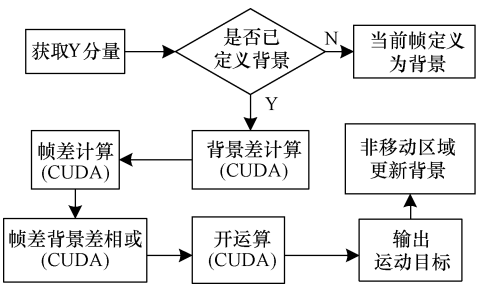


图 3 CUDA 环境内数据处理流程

3.2 CUDA 数据处理布局

CUDA 架构的内存分为寄存器、常量内存、纹理内存、共享内存、本地内存和全局内存等。共享内存读写速度相当快,一般为 1 个 ~2 个 GPU 时钟周期,而全局内存读写速度高达 400 个 GPU 时钟周期左右[12]。共享内存要求数据有局部性重用(一个 block 内共享),且每个 GPU 处理单元共享内存最大限制为 48 KB,所以不适用针对高清图像的多步处理;全局内存容量大,适合图像批处理数据的存储。为了提高访问全局存储器的效率,本文采用了一种称为合并访问的机制来加速全局存储器的访问[13]。

当运动检测计算时,大量的访问全局内存存在冗余性,效率较低。因此,本文采用一种共享内存和全局存储器合并访问的方式。同一个块内线程可访问同一共享内存,每个线程可将当前待处理数据和上下邻域数据存储至共享内存中,如图 4 所示。当所有线程读取完毕后,线程处理时可直接从共享内存中获取需要访问的邻域数据。为了能够高效地访问显存,读取和存储必须对齐,宽度为 4 Byte。如果没有正确的对齐,读写将被编译器拆分为多次操作,极大地影响效率[14]。计算每个输出点时,需要取其邻域的 8 个像素,相邻像素处理时读取到的数据有重叠(如图 5(a)所示),为了避免重复访问所以每个线程处理 4 ×4 个像素,计算此 16 个像素点时,共访问 36 个邻域像素值,为 4 字节对齐(如图 5(b)所示)。4 ×4 像素计算时,每个线程平均需要访问全局内存 68 次,而采用共享内存后,每个线程平均访问全局内存次数降低至 36 次,节省了 52.9% 的全局内存访问。

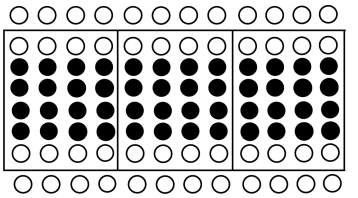


图 4 同一块内共享内存存储数据

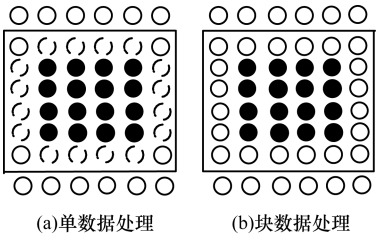


图 5 单数据处理和块数据处理方式对比

运用 CUDA 进行块内的计算时,需要通过 < < < *numBlocks*, *threadPerBlock* > > > 进行内核数(numBlocks)和块内线程数(threadsPerBlock)的配置[12]。针对上文中提到的每次处理 4 ×4 个数据,每次访问到的数据 *x* 坐标 *idx*,*y* 坐标 *idy* 分别为:

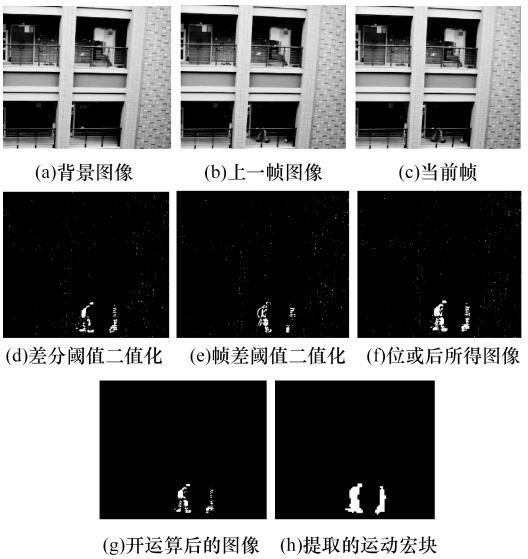
$$idx = blockIdx.x \times (blockDim.x << 2) + (threadIdx.x << 2)$$
$$idy = blockIdx.y \times (blockDim.y << 2) + (threadIdx.y << 2) \quad (6)$$

其中,*blockIdx* 为当前块的索引号;*blockDim* 表示每个块的维度;*threadIdx* 为当前块内线程索引号。

4 实验结果及分析

该算法实验环境的 GPU 为 NVIDIA Geforce 8400 GS(computer capability:1.1),CPU 为 Intel(R) Core(TM)2 Duo。网络视频分辨率分别设置为 720 ×480 像素,1 280 ×720 像素和 1 920 ×1 080 像素。实验数据如图 6 所示。其中,图 6(a)为上一次

更新过的背景图像;图 6(b)为上一帧图像;图 6(c)为当前帧;图 6(d)为图 6(a)与图 6(c)进行差分阈值二值化的图像;图 6(e)为图 6(b)与图 6(c)进行帧差阈值二值化的图像;图 6(f)为图 6(d)和图 6(e)图像位或后所得图像;图 6(g)为图 6(f)进行开运算后的图像;图 6(h)为提取的运动宏块,此运动宏块以外的区域进行背景更新。



(a)背景图像　(b)上一帧图像　(c)当前帧

(d)差分阈值二值化　(e)帧差阈值二值化　(f)位或后所得图像

(g)开运算后的图像　(h)提取的运动宏块

图 6　高清监控运动检测实验输出

实验结果表明,本文采取的改进帧差与背景差分结合的方法可以准确检测出运动目标的轮廓以及内部纹理,将图像进行开运算后去除了孤立噪点,消除了误判断成运动目标的小物体。所以本文中的基于 CUDA 的解码及运动检测算法保证了输出数据的精确度。

在相同环境下,运用相同算法对比 CPU 和 GPU 的运动检测计算,实验数据如表 1 所示,其中,总时间比为 CPU 运动检测时间与 GPU 运动检测的时间之比。720 × 480 像素分辨率下平均码率为1.5 Mb/s,I 帧大小为 50 Kb,帧率为 25 f/s。1 280 × 720 像素分辨率平均码率为 4 Mb/s,I 帧大小平均为 100 Kb,帧率为 25 f/s。1 920 × 1 080 像素分辨率平均码率为 8 Mb/s,I 帧大小平均为 150 Kb,帧率为25 f/s。表 1 中为不同分辨率的视频解码及运动检测所需时间的平均数据。

表 1　不同分辨率的视频解码及运动检测所需时间对比

分辨率/像素	CPU 解码/ms	GPU 解码/ms	CPU 运动检测/ms	GPU 运动检测/ms	总时间比
720 × 480	4. 382	0. 211	60. 143	12. 146	5. 222
1 280 × 720	5. 936	0. 266	162. 554	27. 753	6. 015
1 920 × 1 080	14. 357	0. 333	358. 849	35. 462	10. 426

表 1 数据显示,针对 720 × 480 像素,1 280 × 720 像素,1 920 × 1 080 像素分辨率的视频,基于 CUDA 的硬解码及运动检测与 CPU 软解码和运动检测相比,总时间分别缩短了 5. 2 倍、6. 0 倍、10. 4 倍。

从表 2 数据可看出,基于 CUDA 的运动检测计算运用本文中优化后的内存访问方案较优化前平均时间缩短了 1. 91 倍。表中的时间比表示优化前的检测时间与优化后的检测时间之比。针对 1 080 P 分辨率,基于 CUDA 的视频运动检测方法帧率可达 52. 6 f/s,满足了高清智能监控系统的实时性要求。

表 2　不同分辨率的 GPU 运动检测优化前后时间对比

分辨率/像素	优化前/ms	优化后/ms	时间比
720 × 480	12. 146	6. 344	1. 915
1 280 × 720	27. 753	14. 513	1. 912
1 920 × 1 080	35. 462	18. 586	1. 908

5　结束语

本文设计了一个在高清智能监控系统中的环境实时运动检测算法。该算法兼顾了高清视频运动检测的准确性与实时性。实验结果表明,本文提出的运动检测算法精度高,计算复杂度较低。在 GPU 内进行图像运动检测计算可避免传统 GPU 硬解码后显示内存中视频数据与 CPU 之间的传输瓶颈问题。在 CUDA 内部的运动检测算法采用了多线程共享内存与合并访问的方式,减少了数据访问延迟,保证了高清智能监控系统的实时性要求。

参考文献

[1] Hu Weiming, Tan Tieniu, Wang Liang, et al. A Survey on Visual Surveillance of Object Motion and Behaviors [J]. IEEE Transactions on Systems, Man, and Cybernetics, 2004, 34(3): 334-352.

[2] Scotti G, Marcenaro L, Coelho C, et al. Dual Camera Intelligent Sensor for High Definition 360 Degrees Surveillance [J]. IEEE Proceedings on Vision, Image and Signal Processing, 2005, 152(2): 250-257.

[3] 周　游,刘艳滢,王春民,等. 几种人体运动检测算法的比较分析[J]. 吉林大学学报:信息科学版,2009,27(6):652-657.

[4] Padalikar S, Diamos G. Exploring the Latency and Bandwidth Tolerance of Cuda Applications[Z]. 2009.

[5] McHugh J M, Konrad J, Saligrama V, et al. Foreground-adaptive Background Subtraction [J]. IEEE Signal Processing Letters, 2009, 16(5): 390-393.

[6] Davide A, Matteucci M, Naccari M. A Revaluation of Frame Difference in Fast and Robust Motion Detection [C]//Proc. of the 4th ACM International Workshop on Video Surveillance and Sensor Networks. New York, USA: ACM Press, 2006: 215-218.

(下转第 296 页)

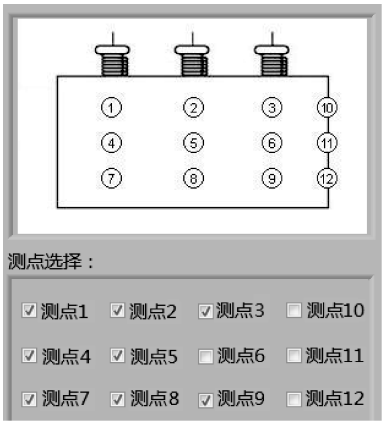


图 7 振动测点选择界面

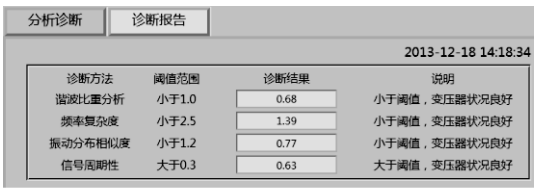


图 8 诊断分析界面

4 结束语

本文设计了一种基于振动的便携式电力变压器状态监测与故障诊断系统(PTCMS),描述了系统的硬件和软件的设计,同时介绍了本文系统的主要诊断算法。该系统可以带电采集并分析变压器稳态和瞬态运行时的振动信号,根据内置的诊断算法对变压器状态进行诊断,若诊断结果为变压器异常则发出警报。在浙北变电站现场的试用结果表明,该系统可以方便、有效地实现对变压器振动、电流、电压等信号的采集和变压器运行状态的分析诊断工作,为工作人员提供了一种监测变压器运行状态的便捷手段。今后,需要对 PTCMS 做进一步的结构优化,减小其体积和重量,同时还需对诊断算法进行深入研究,以提高系统对变压器状态诊断的准确性。

参考文献

[1] 廖怀东. 变压器油色谱分析及故障判断[J]. 高电压技术,2006,32(1):112-113.

[2] Garcia B, Burgos J C, Alonso A M. Transformer Tank Vibration Modeling as a Method of Detecting Winding Deformations——Part I: Theoretical Foundation [J]. IEEE Transactions on Power Delivery, 2006, 21(1): 157-163.

[3] Garcia B, Burgos J C, Alonso A M. Transformer Tank Vibration Modeling as a Method of Detecting Winding Deformationss——Part II: Experimental Verification [J]. IEEE Transactions on Power Delivery, 2006, 21(1):164-169.

[4] 洪凯星,潘再平,黄 海. 电力变压器绕组轴向振动的建模与分析[J]. 变压器,2010,47(12):32-37.

[5] 郑 婧,何婷婷,郭 洁,等. 基于独立成分分析和端点检测的变压器有载分接开关振动信号自适应分离[J]. 电网技术,2010,34(11):208-213.

[6] 陈祥献,王婧頔,黄 海,等. 基于 Hilbert-Huang 变换的电力变压器铁芯压紧力监测方法[J]. 振动与冲击,2010,29(9):9-12.

[7] 黄 海,潘家强,郑 婧. 基于切贝雪夫-里兹法的大型电力变压器铁芯三维自由振动分析[J]. 中国电机工程学报,2011,31(36):138-143.

[8] 郭 洁,黄 海,唐 昕,等. 500 kV 电力变压器偏磁振动分析[J]. 电网技术,2012,36(3):70-75.

[9] 李晓兰,黄 海,陈祥献,等. 基于振动法的电力变压器在线状态监测系统设计[J]. 变压器,2008,45(12):60-63.

[10] 郭 洁. 电力变压器振动及故障特征提取研究[D]. 杭州:浙江大学,2011.

[11] 郭 洁,陈祥献,黄 海. 交叉递归图在变压器铁芯压紧力变化检测中的应用[J]. 高电压技术,2010,36(11):2731-2738.

[12] 王惠文. 偏最小二乘回归方法及其应用[M]. 北京:国防工业出版社,1999.

编辑 顾逸斐

(上接第 291 页)

[7] Gonzalez R C, Wintz P. Digital Image Processing[M]. 2nd ed. New York, USA: Academic Press, 1987.

[8] Yang J, Tian M. Denoising of Coal Flotation Froth Image Using Opening and Closing Filters with Area Reconstruction and Alternating Order Filtering [M]// Deng Hepu, Miao Duoqian, Lei Jingsheng. Artificial Intelligence and Computational Intelligence. Berlin, Germany: Springer, 2011: 376-382.

[9] Rane M A. Fast Morphological Image Processing on GPU Using CUDA[D]. Pune, India: Pune University, 2013.

[10] Han Bo, Zhou Bingfeng. Efficient Video Decoding on GPUs by Point Based Rendering [C]//Proc. of Eurographics Symposium on Graphics Hardware. Vienna, Austria: ACM Press, 2006: 79-86.

[11] Wang Guohui, Wu M, Sun Yang, et al. A Massively Parallel Implementation of QC-LDPC Decoder on GPU [C]//Proc. of the 9th Symposium on Application Specific Processors. [S. l.]: IEEE Press, 2011: 82-85.

[12] NVIDIA. CUDA Programming Guide Version 2. 1[M]. Santa Clara, USA: NVIDIA Corporation, 2009.

[13] Rao Chao, Liu Shuoqi. Research of CUDA in Intelligent Visual Surveillance Algorithms [C]//Proc. of the 3rd Chinese Conference on Intelligent Visual Surveillance. [S. l.]: IEEE Press, 2011: 73-76.

[14] NVIDIA. The CUDA Compiler Driver NVCC [M]. Santa Clara, USA: NVIDIA Corporation, 2009.

编辑 顾逸斐