计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):95−100 [doi: 10.15888/j.cnki.csa.006079]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 GPU_CPU 异构并行加速的人头检测方法①

彭景维, 童基均

(浙江理工大学 信息学院, 杭州 310018)

摘　要: 多尺度协同的人头检测系统中, 梯度方向直方图应用于高清视频监控领域时常因特征提取时的海量计算而不能满足监控视频的实时性要求, 提出一种基于 GPU_CPU 异构并行加速的人头检测方法, GPU 端负责 HOG 特征提取的庞大的密集型的区块的并行计算, CPU 端负责检测过程中的其它模块的执行. 传统的并行归约算法因其在 HOG 特征提取过程中的时间复杂度不够理想, 提出改进的并行归约算法, 通过“下扫”的并行计算方式, 减少节点被计算的次数, 降低了 HOG 特征提取时的时间复杂度. 实验表明, 提出的方法检测速率优于传统的 CPU 的检测方法, 其效率提升约 10 倍.

关键词: 梯度方向直方图; CUDA; 异构; 并行归约; 人头检测; GPU

引用格式: 彭景维,童基均.基于 GPU_CPU 异构并行加速的人头检测方法.计算机系统应用,2017,26(11):95–100. http://www.c-s-a.org.cn/1003-3254/6079.html

Human Head Detection Based on GPU_CPU Heterogeneous Parallel Acceleration

PENG Jing-Wei, TONG Ji-Jun

(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: In the multi-scale collaborative human head detection system, the gradient direction histogram cannot meet the real-time requirement of video surveillance because of the massive computation in the high-definition video surveillance field. This paper proposes a method of human head detection based on GPU_CPU heterogeneous parallel acceleration. The GPU is responsible for the HOG feature extraction of large-intensive block parallel computing, and CPU is responsible for the implementation of other modules. The traditional parallel reduction algorithm is not excellent in the HOG feature extraction, and an improved parallel reduction algorithm is therefore proposed, which reduces the time complexity by the parallel computing of down-sweep to reduce calculated times of nodes, and the experimental results show that the proposed method is more efficient than the traditional one for over about 10 times.

Key words: gradient direction histogram; CUDA; isomerism; parallel reduction; head detection; GPU

计算机智能视频监控系统是近年来的热点研究方向, 监控视频中的行人人头检测技术在此领域已获得较好的发展[1-3], 然而常常受到检测速度缓慢的制约. 图形处理器 (Graphics Processing Unit, GPU) 在通用计算能力上的飞跃发展受到越来越多研究学者的关注和实际应用, Pogorelov 等[4]提出一种基于医疗系统的 GPU 的处理框架, 应用于内窥镜视频信息的实时自动分析、实时流和处理和资源消耗方面, 并已达到预期效果. Zhang 等[5]搭建起一款可编程的 GPU_CPU 架构, 为程序员提供包括数据竞争在内的多个并发线程软件漏洞的检测平台, 减少了软硬件开发过程中的繁琐任务. Li 等[6]将 CPU 的并行加速能力用于卷积神经网络 (Convolutional Neural Network, CNN) 的前馈计算, 将目标检测的速度提升约 6.97 倍. 本文将 GPU 的这一优越性能运用于多尺度协同的人头检测系统中.

在此检测系统中, 对于输入的视频帧序列, 一般通

① 基金项目: 浙江省重点研发计划 (2015C03023); 浙江理工大学“521 人才培养计划”
收稿时间: 2017-02-28; 修改时间: 2017-03-27; 采用时间: 2017-03-29

过图像尺度缩放、梯度方向直方图 (Histogram of Oriented Gradient, HOG) 特征提取、检测窗口的扫描、分类器的分类、高斯金字塔的融合处理及最终的决策标定等主要步骤实现. 此方法在标清分辨率下的监控视频领域已取得明显成效, 然而其检测速率仍难以满足高清视频监控领域检测速率的实时性要求. 经统计分析发现, 导致检测速率下降的主要原因是 HOG 特征提取阶段存在的庞大的计算量[7]. 为有效响应快速发展的高清视频监控领域的需求, 在准确检测到监控视频中的行人人头的前提下, 保证视频监测画面的流畅性, 本文对传统的 HOG 特征提取方法作出改进, 提出一种在 CUDA 平台下开发的基于 GPU_CPU 异构并行加速的人头检测方法.

1 GPU_CPU 异构并行的加速方法

1.1 基于 CPU 的 HOG 特征提取分析

传统的 HOG 特征提取方法主要依赖于 CPU 的全局控制. 在梯度的计算前, 为减少局部光照、阴影等不可控因素的影响, 对于摄像头捕获的一帧图像, 需作标准化的 Gamma 空间和颜色空间校正, 以常用的梯度算子[-1 0 1]和$[1\ 0\ -1]^T$ 分别对图像作水平方向和垂直方向的卷积运算, 并计算出与之对应的每一个像素点的梯度方向及大小. 依照 Dalal 等人[8]的算法思想, 将一幅图像先划分为若干的像素单元 (Cell), 梯度方向也可平均的划分成 9 个区间 (bin), 统计计算 9 维特征向量的梯度信息; 最后将设定的像素单元块组成一个块 (Block); 同一个单元的特征在不同的块内多次计算, 经标准化处理后, 使得相应的值体现在最后的特征向量中; 收集图像中每一块对应的直方图向量, 并组合起来, 便可形成最终的 HOG 特征[9]. 然而常常受到提取 HOG 特征时的大量计算任务的制约, 使得该方法应用于高清视频监控领域时在检测速率方面仍有不小挑战. 因此, 本文借助 CUDA 平台作出改进.

1.2 CUDA 平台简介

2007 年 4 月, NVIDIA 公司推出了用以专门解决高性能计算的并行通用平台 CUDA(Compute Unified Device Architecture)[10], 可支持 C/C++语言的编写开发. 一套完整的 CUDA 架构由 CPU 的串行和 GPU 的并行步骤组合而成, 如图 1 所示. CPU 一般应用于 CUDA 架构中编程模型的主机 (Host), GPU 因其显著的并行计算处理能力, 作为主机的协处理端 (Device). 也因为二者的不同特性分配不同的工作任务, CPU 在这里主要负责串行计算, GPU 则主要负责线程化的并行计算任务, 以此分配方式对于 CPU 在带宽的限制及计算速度上都可以较好的优化.

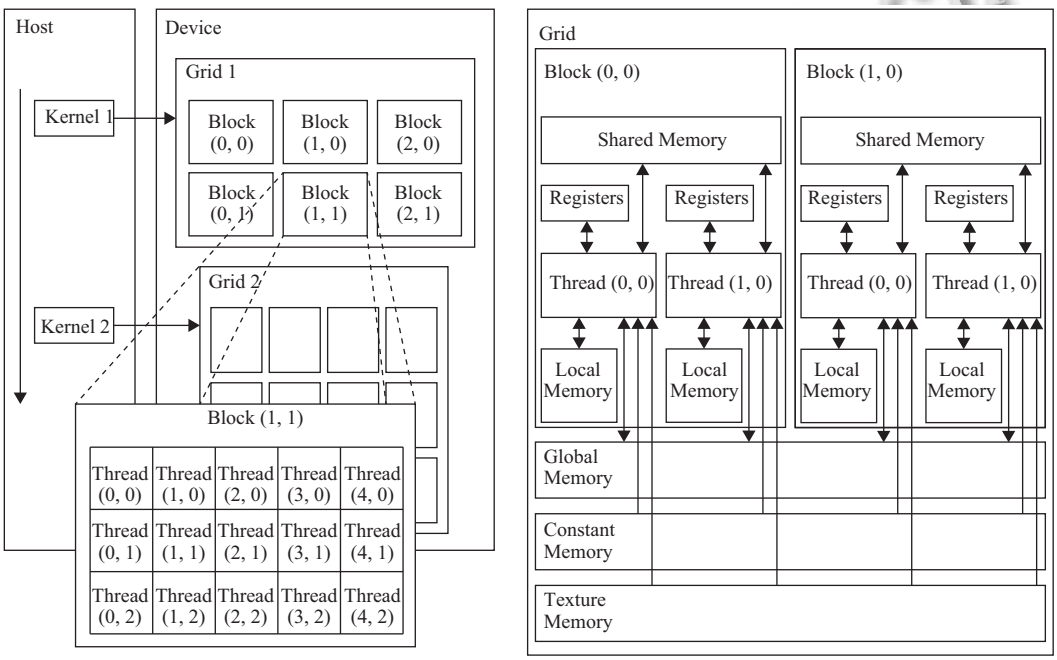

图 1 CUDA 存储器的体系结构

在并行计算过程中, GPU 通常执行整个程序中的计算密集型的模块, 并且将其封装成函数, 被称为内核函数 (Kernel). 在 CUDA 的程序架构中, 可较好的利用该函数实现并行计算, 并已成为此过程中的必选步骤[11].

内核函数 Kernel 接收程序请求的指令后, 可作出实时反应并依次执行, 构建具有逻辑性的程序执行顺序[12]. 这里, 每一核函数会开启与之对应的线程网格 (Grid), 一个线程网格由若干的线程单元块 (Block) 组成, 若干的线程 (Thread) 则可构成一个线程单元块, 同时, 线程单元块中的线程也共享相同的内存空间. 例如, 对于一个基本内存单元, 如全局内存 (Global Memory, GM)、纹理内存 (Texture Memory, TM) 及常量内存 (Constant Memory, CM), 所有的线程都可较易访问. 然而, 只有同一块的线程可以访问共享内存 (Shared Memory, SM)[13]. 每一个多处理器都有仅属于自己的寄存器 (Registers) 和片上共享内存, 每一线程也同时包含本地内存 (Local Memory, LM) 和其对应的寄存器. 全局存储器可以通过主机 (Host) 读取或写入, 然而在读写速度上比共享内存慢得多[14].

1.3 基于 GPU 的 HOG 特征提取分析

本文的区块梯度直方图特征提取的异构体系, 先由 CPU 端负责数据接收及传输, 采集摄像机捕获的视频信息, 经过 GPU 端的密集型的并行化处理, 并不断的将每帧处理数据拷贝回主机端 (Host), 再由 CPU 端控制其它相应的模块数据处理. 其中 GPU 设备端的并行化处理主要包括梯度的计算、梯度方向直方图的构建、直方图归一化处理及 HOG 描述符的提取等主要步骤, 其计算流程如图 2 所示.

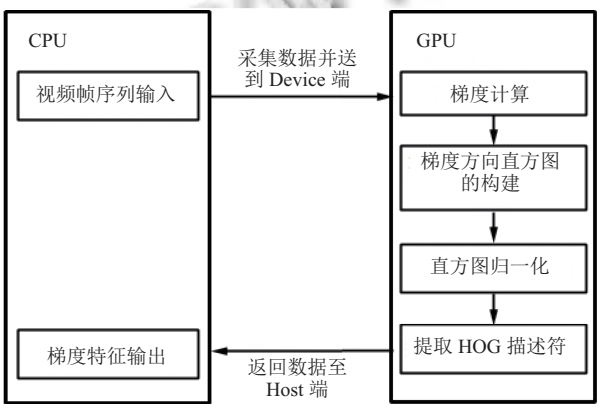

图 2 异步并行计算流程

1.3.1 梯度的并行化计算

一般摄像机的信息读取主要来源于彩色图像和灰度图像. 在梯度的并行化计算时, 可预先设定两个内核函数, 分别用于彩色图像和灰度级图像的处理. 这里预先设定一个线程块包含 512 个线程, 每个线程计算出一个像素单元, 因此, 一个线程块的运行一次性便可执行 512 个像素单元的梯度计算.

对于灰度图像, 每个线程需计算一个梯度幅值和方向, 这里可使用共享内存来存储每个像素的信息. 一幅彩色图像包含三个通道的特征计算, 每个线程则需计算三个通道的颜色的梯度幅值和方向. 最后选择最大的梯度幅值和方向写回 Host 端.

1.3.2 梯度直方图的构建

每一个 CUDA 的线程块响应一个 HOG 描述符的区块 (简写为 HOG-Block) 的计算任务. 人头检测扫描窗口中, 定义一个 HOG-Block 块包含 4 个细胞单元 (Cell), 而一个细胞单元由一个 8×8 的像素单元构成, 所以, 需使用的线程数量为 4×8=32 个. 在逻辑上将每一个线程块划分为 4 个子块, 这样, 每个线程处理 8 个像素单元. 处理过程中, 为避免全局内存的访问延迟, 这里动态申请的方式调用共享内存 (SM) 作为直方图数据的临时存储空间[15], 存储每个线程的计算结果. 在动态的调用共享内存前, 为了能够较易获取存储空间的首地址, 需首先声明外部共享数组. 本文该部分的执行过程的伪代码如下:

```
extern_shared_char firstArray[];//声明外部共享数组, 其首地址为 firstArray
_global_void myKernel()//核函数
{
  …
  BYTE * oriLpm = (BYTE *) & firstArray; //前一帧局部像素矩阵
  //下一帧局部像素矩阵, nexLpmScale 为 nexLpm首地址
  BYTE * nexLpm = (BYTE *) & nexLpm[nexLpmScale];
  …
  for i from 0 to oriLpmH //oriLpm 局部像素矩阵的高
    for j from 0 to oriLpmW//oriLpmW 局部像素矩阵的宽
      …
      拷贝上一帧中与 (i, j) 对应的点到 oriLpm 的 (i, j) 位置
      …
    end for
  end for
  for i from 0 to oriLpmW
    for j from 0 to oriLpmW
      …
      下一帧中与 (i, j) 对应点到 nexLpm 的 (i, j) 位置
      …
```

```
        end for
      end for
    }
```

1.3.3 直方图归一化

每一个 CUDA 线程块负责与之对应的一个 HOG 描述符中的 Block 块的归一化处理, 因此, CUDA 的线程块的数量分配应与 HOG 描述符中的 Block 块的请求数量一一对应. 由于定义的一个 HOG-Block 块中包含 4 个细胞单元, 且每一个细胞单元设定 9 维的特征向量, 这样, 不难计算出所需请求用于归一化的线程数量为 4×9=36, 这里为了简化后续的计算, 可将线程的数量向上增设为 64 个. 数据存储时为避免访问延迟, 这里仍可调用共享内存来访问.

在执行共享内存空间时, 每一个线程块将占用 64×4=256 个字节空间. 线程中的每个元素的归一化方式为平方求和运算, 考虑到计算量与速度的直接关联关系, 这里采用并行归约 (Parall Reduction, PR) 的计算方式作求和运算. 然而, 传统的并行归约算法[16]因其较长的时间复杂度, 在直方图的归一化过程中并不能有很好的表现[17,18], 本文专门针对 HOG 特征的归一化提出一种改进的计算方式, 如图 3 所示.

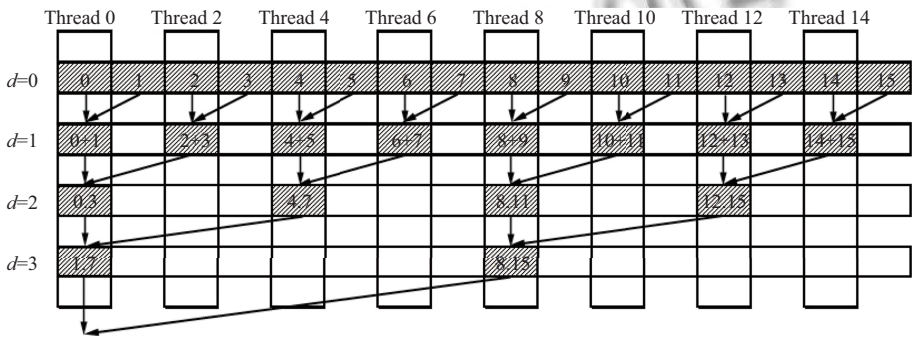

图 3 改进的并行归约计算方式

例如, 现计算一个含有 16 个计算单元的数组, 则需首先申请 16 个线程 (Thread) 单元, d 表示所需计算的倒二叉树的层级, 这里为$d = \log_2 16 = 4$层, 则第 i 层对应的倒二叉树的节点数为$2^{d-i}(0 \le i \le d)$. 线程数量与归一化的数据量一一对应, 在每一个线程中添加额外的一个步幅块, 方便后面的特征提取的计算. 经 k 次步幅长度的添加之后, 可以被 $2k$ 及前 $2k$-1 个元素整除, 此时, 这里的 $2k$ 个元素便已经包含了包括自身的求和的那部分. 与 Harris 等人[16]的算法的时间复杂度相比, 本文时间复杂度已降为 O(logn), 其中 n 代表数组的长度.

1.3.4 提取 HOG 特征描述符

人头的扫描检测过程中, 设定的扫描窗口大小为 64×64 像素, 每 8×8 的像素大小构成一个 Cell 单元, 每 2×2 个 Cell 单元构成一个块, 预先设定的一个 Block 块的大小为 16×16, 而每一个 Cell 有 9 个特征, 易计算得到, 每个块内有 4×9=36 个特征向量, 每一步幅长度设定为 8 个像素单元, 则水平和垂直方向各有 7 个扫描窗口. 因每一个 CUDA 的线程块负责一个 HOG 特征的计算, 则所需要申请的线程数量为 16×16=256 个. 计算过程中, 可再申请两个不同于前面的内核空间, 分别作用于每一帧图像行和列的 HOG 特征的提取. 这样, HOG 特征的描述符较易从每一输入的视频帧序列的高、宽、Block 块及检测窗口的步幅长度中计算得出.

在多尺度协同的人头检测系统中, 当 HOG 特征描述符的快速提取后, 可继续作滑动窗口的扫描分析, 判断当前的视频帧中是否存在行人人头的目标对象, 及时融合并标定检测结果.

2 实验及结果分析

当前集成的开发环境为 Visual Studio 2013, 并以 CUDA Toolkit 3.4.3 及 NVIDIA GPU Computing SDK 作为开发工具, 实验开发环境详细配置如表 1 所示.

实验调取了三种不同分辨率 (320×180, 640×360 和 1280×720) 下的视频监控信息, 并相应的随机截取每一组中的一段有效视频信息作为人头检测的测试视频. 对于每组测试视频, 分别以传统 CPU 模式和本文的 GPU_CPU 异构并行模式下作了测试对比. 实验直接将 windows 系统定义的 QueryPerformanceCounter[19]函数作为计时方式并有效的使用, 得到具有高精度的

计时器的值. 实验过程中将每帧所耗时长累加, 最后除以总帧数, 得到平均每帧耗时及帧率 (FPS), 统计并记录了三组实验数据, 如表 2 所示.

表 1 实验开发环境

资源配置	规格参数
GPU	NVIDIA GeForce GTX 750 Ti
GPU Memory	4GB
CPU	Intel (R) Core(TM) i5-4460 CPU @3.20GHz
CPU Memory	4GB
CUDA Toolkit	v3.4.3
OS	Window 7 Ultimate 64 bits

与传统的 PR 计算方式相比, 本文改进的并行计算的时间复杂度已降为 O(logn), HOG 特征提取时的处理时间明显减少. 从表 2 中的三组实验数据不难发现, 在运行速度上, 本文 GPU_CPU 异构模式的加速算法与传统的 CPU 模式下相比有明显的提升, 加速比约为 10 倍. 另一方面, 本文在人头目标的识别算法上与传统的 CPU 模式下保持一致, 因此没有影响到检测结果的准确性. 因此, 本文的 GPU_CPU 异构模式下的人头检测方法, 不仅可为普通的安防监控视频 (分辨率 352×288) 的分析检测提供流畅性的平台. 同时, 对于 720P 的高清视频的监控环境下, 帧率也已大于 20, 基本满足实时性要求.

表 2 三组实验数据分析

视频分辨率	CPU模式		GPU_CPU异构模式		加速比
	平均每帧耗时 (ms)	FPS	平均每帧耗时 (ms)	FPS	
320×180	92.76	10.78	10.07	99.31	9.21
640×360	218.34	4.58	21.29	46.97	10.25
1280×720	478.47	2.09	49.27	20.31	9.74

3 结论

HOG 特征在多尺度的人头检测系统中已有显著的效果, 对于不断发展的高清视频的检测速率难以提升的障碍, 本文的提出的 GPU_CPU 异构并行加速的检测策略, GPU 端负责 HOG 特征提取的庞大的密集型的区块的并行计算, CPU 端负责检测过程中的其它模块的执行. 对于传统的并行归约算法在 HOG 特征提取时的时间复杂度不够理想, 本文也提出了改进的并行归约算法, 降低了 HOG 特征提取过程中的算法的时间复杂度, 使得本文的方法在没有影响到检测准确率的情况下, 加快了检测速率. 当前, 本文方法不仅能提高普通安防监控视频的检测效率, 对于不断发展的高清视频也已基本满足实时检测的要求.

参考文献

1 Peng HR, Chang KW, Roth D. A joint framework for coreference resolution and mention head detection. Proc. of the 19th Conference on Computational Language Learning. Beijing, China. 2015. 12–21.

2 Irani R, Nasrollahi K, Moeslund TB. Improved pulse detection from head motions using DCT. Proc. of the 2014 International Conference on Computer Vision Theory and Applications (VISAPP). Lisbon, Portugal. 2014. 118–124.

3 Aziz K, Merad D, Iguernaissi R, *et al*. Head detection based on skeleton graph method for counting people in crowded environments. Journal of Electronic Imaging, 2016, 25(1): 013012. [doi: 10.1117/1.JEI.25.1.013012]

4 Pogorelov K, Riegler M, Halvorsen P, *et al*. GPU-accelerated real-time gastrointestinal diseases detection. Proc. of the 29th International Symposium on Computer-Based Medical Systems (CBMS). Dublin, Ireland. 2016. 185–190.

5 Zhang WH, Yu SQ, Wang HJ, *et al*. Hardware support for concurrent detection of multiple concurrency bugs on fused CPU-GPU architectures. IEEE Trans. on Computers, 2016, 65(10): 3083–3095. [doi: 10.1109/TC.2015.2512860]

6 Li SJ, Dou Y, Lv Q, *et al*. Optimized GPU acceleration algorithm of convolutional neural networks for target detection. Proc. of the 18th International Conference on High Performance Computing and Communications. 2016. 224–230.

7 Mizuno K, Terachi Y, Takagi K, *et al*. Architectural study of HOG feature extraction processor for real-time object detection. Proc. of the 2012 IEEE Workshop on Signal Processing Systems (SiPS). Quebec City, QC, Canada. 2012. 197–202.

8 Dalal N, Triggs B. Histograms of oriented gradients for human detection. Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Diego, CA, USA. 2005, 1. 886–893.

9 Saeed A, Al-Hamadi A, Ghoneim A. Head pose estimation on top of Haar-like face detection: A study using the Kinect sensor. Sensors, 2015, 15(9): 20945–20966. [doi: 10.3390/s150920945]

10 Saveetha V, Sophia S. Optimization of K-means clustering on graphics processing unit using compute unified device architecture. Journal of Computational and Theoretical Nanoscience, 2017, 14(1): 789–795. [doi: 10.1166/jctn.2017.6274]

11 Chu CH, Hamidouche K, Venkatesh A, *et al*. CUDA kernel based collective reduction operations on large-scale GPU

clusters. Proc. of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). Cartagena, Columbia. 2016. 726–735.

12 Wahib M, Maruyama N. Automated GPU kernel transformations in large-scale production stencil applications. Proc. of the 24th International Symposium on High-Performance Parallel and Distributed Computing. Portland, OR, USA. 2015. 259–270.

13 Khoirudin, Jiang SL. GPU application in Cuda memory. Advanced Computing An International Journal, 2015, 6(2): 1–10. [doi: 10.5121/acij]

14 Borcovas E, Daunys G. CPU and GPU (Cuda) template matching comparison. Science-Future of Lithuania/Mokslas-Lietuvos Ateitis, 2014, 6(2): 129–133. [doi: 10.3846/mla.2014.16]

15 Abid M, Jerbi K, Raulet M, *et al*. Efficient system-level hardware synthesis of dataflow programs using shared memory based FIFO. Journal of Signal Processing Systems, 2017(in Press). [doi: 10.1007/s11265-017-1226-x]

16 Harris M, Sengupta S, Owens JD. Parallel prefix sum (scan) with CUDA. Nguyen H. GPU Gems 3. 2007. 851–876.

17 Nabila M, Yousra BJ, Eric W. Optimized HOG descriptor for on road cars detection. Proc. of the 10th International Conference on Distributed Smart Camera. Paris, France. 2016. 166–171.

18 Moustafa M, Ebeid HM, Helmy A, *et al*. Parallel implementation of super-resolution based neighbor embedding using GPU. Proc. of the International Conference on Advanced Intelligent Systems and Informatics. Cham, Germany. 2016. 628–638.

19 Zhu YB, Eran H, Firestone D, *et al*. Congestion control for large-scale RDMA deployments. ACM SIGCOMM Computer Communication Review, 2015, 45(5): 523–536. [doi: 10.1145/2829988]