组网雷达估测降水系统并行化方案的设计与实现

吴石磊[1] 安 虹[1] 李小强[1] 周 伟[2] 刘 谷[1] 魏学超[1]
(中国科学技术大学计算机科学与技术学院 合肥 230027)[1]
(中国人民解放军陆军军官学院计算机教研室 合肥 230031)[2]

摘 要 国家气象局天气组网雷达定量估测降水系统不仅拥有较大的计算量,而且具有较大的数据吞吐量,同时对实时性要求较高。如果缩短其执行时间,无疑将会带来巨大的收益。鉴于这些特点,使用 VTune Amplifer XE 对串行程序进行了热点分析和并行性分析,得出程序中有较多线程级并行性,从而制定了相应的并行化方案;然后使用 Win32 多线程和 OpenMP 两种技术对该程序在 Intel 四核处理器平台上进行了并行化。程序主要由单站处理和组网处理两部分组成。由于计算资源的限制,并行后的单站处理程序只有大约 10%的性能提升,而组网处理程序则可以达到近似线性的性能提升。通过调整计算负载,并行化版本的加速比可以达到 5.5。最后,可以得出该并行化方法适用于计算密集且数据吞吐量较大的一类应用。

关键词 热点分析,并行性分析,OpenMP,Win32 多线程,天气组网雷达
中图法分类号 TP302.7 **文献标识码** A

Parallel Program Design and Implementation on Precipitation Program of Networking Weather Radar System

WU Shi-lei[1] AN Hong[1] LI Xiao-qiang[1] ZHOU Wei[2] LIU Gu[1] WEI Xue-chao[1]
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)[1]
(Computer TD&R Lab, PLA Army Officer Academy, Hefei 230031, China)[2]

Abstract Quantitative estimation precipitation system of China Meteorological Administration is compute-intensive and a real-time application. If one work can reduce its execution time, it will bring enormous benefits to the society. First, implementing hotspot analysis and concurrency analysis on serial code with VTune Amplifer XE. Then, parallelizing this program on Intel 4-core platform using both Win32 thread and OpenMP. This program has two separated part: single-station process and networking process. Because of limited computing resource, the parallel version of single-station process gains only 10% performance increase. But performance of networking process can achieve nearly linear increase. By balancing the load of the whole program, parallel version may achieve speedup of 5.5. And through analysis of methods to parallelize code, summarizing the general method of parallelizing a class of program.

Keywords Hotspot analysis, Concurrency analysis, OpenMP, Win32 thread, Networking weather radar

1 引言

当今,不论是在嵌入式设备还是大型高性能计算系统中,多核处理器都占主导地位。但是如何高效地在多核处理器上编程,以利用其越来越强大的处理能力,一直是困扰着工业界和学术界的一个重大难题。国家气象局天气组网雷达定量估测降水系统实现了组网雷达基本反射率和组网估测降水两类产品的处理和显示功能。一方面,该应用对实时性要求较高,缩短程序的执行时间迫在眉睫;另一方面,现有的串行程序已经不能有效地利用多核平台的硬件资源,提高处理器的利用率有利于降低成本。本文对组网雷达估测降水系统的并行化方案进行了设计与实现,并得出了以下认识。

(1)对于计算密集型且数据吞吐量较大的一类应用[1]的并行化方法是有规律可循的。

(2)通过调整各个计算单元的计算负载,可以大大减少整个系统的执行时间。

2 问题描述

天气组网雷达定量估测降水系统需要处理 134 部雷达的单站数据并实现多种组网产品的产生和显示。其实时性要求为在 6 分钟内完成以下 3 件事情:1)单站程序从基数据服务器上取得相应雷达站点的基数据,并将这些数据处理生成单站产品;2)组网程序从基数据服务器中取得所有雷达站点的单站产品,并对这些单站产品进行组网生成各种组网产品;3)降水产品展示子系统从组网机器上取得组网程序的计算结果并进行产品展示。

到稿日期:2011-04-13 返修日期:2011-07-05 本文受国家自然科学基金重点项目(60633040),国家自然科学基金项目(60970023),国家 973 计划项目(2005CB321601),国家 863 计划重大项目(2006AA01A102),国家 863 计划项目(2009AA01Z106),国家科技重大专项项目(2009ZX01036-001-002),教育部-英特尔信息技术专项科研基金项目(MOE-INTEL-08-07)资助。
吴石磊(1986—),男,硕士生,主要研究方向为高性能处理器体系结构,E-mail:shileiwu@mail.ustc.edu.cn。

实验平台配置：Intel Core2 Q8200 处理器，4GB 内存，系统为 Windows XP sp3，用 Visual Studio 2005 进行编译。本文中对该程序进行并行化是为了实现以下两个目标：(1)尽量地缩短程序在目标平台上的计算时间；(2)在不增加程序在目标平台上的计算时间的同时，提高程序的可扩展性，让程序适应未来的硬件升级需求。

3 热点分析与并行性分析

英特尔推出了最新的并行程序开发工具套件 Parallel Studio XE 2011[2]，该版本可以帮助软件开发者更轻松地提高串行和并行应用的性能和可靠性，以便充分利用最新的多核处理器。本文中主要用到了性能分析工具 VTune Amplifer XE[3]，它支持最新的热点分析和并行性分析。热点分析跟以前的研究有些不一样，以前 VTune 有采样和调用图，现在比较方便地把这些功能合二为一；并行性分析对我们来说比较重要，因为通过并行分析可发现整个程序的并行度，再结合热点分析的结果，最后得出热点的地方是不是有条件并行化。

3.1 热点分析

使用 VTune Amplifer XE 对单站产品的生成过程进行热点分析，得出了程序中各个模块对应的运行时间，如图 1 所示。从图 1 中可以看出，单站程序的热点位于数据压缩输出模块和质量控制模块。本文中的优化工作也正是从程序最耗时的部分入手来提高程序的性能。

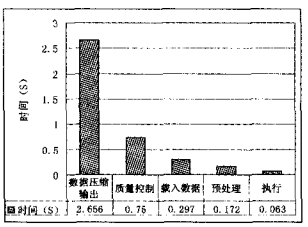

图 1 单站程序热点分析结果

3.2 并行性分析

使用 VTune Amplifer XE 对单站产品的生成过程进行并行性分析，得出了单站程序中各个模块对应的并行度，如图 2 所示。

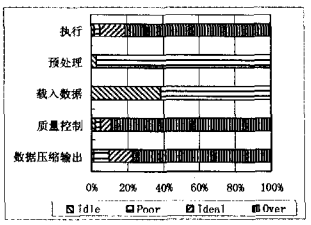

图 2 单站程序并行性分析结果

其中 Idle 表示 CPU 处于空闲状态；Poor 表示只有一个线程处于执行状态；Ideal 表示有两个线程处于执行状态；Over 表示大于等于 3 个线程处于可执行状态。从图 2 可以看出，数据压缩输出和质量控制模块不仅是单站程序的热点，同时具有较好的可并行性；载入数据模块和预处理模块不适合做并行处理；执行模块虽然并行度较高，但由于其耗时较短，并行化后对单站程序的整体性能提升不大，不属于重点考虑的方案。

组网程序没有进行热点分析，原因是组网产品的生成没有划分模块，对单一模块的热点分析只能测出总的执行时间，而不能发现最耗时的部分。不过，我们仍然可以使用 VTune Amplifer XE 对每个组网程序进行并行性分析，得出了 3 种组网程序的并行度，如图 3 所示。从图 3 中可以看出，组网产品中基本反射率 BREF、组网组合反射率 CREF、组网降水率估测 MQPR 都具有很好的并行性。

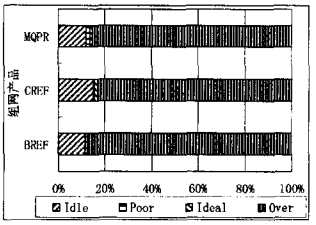

图 3 组网程序的并行性分析结果

3.3 并行化方案概述及简要结果

通过上述分析，发现程序中存在较多的线程级并行性[4]。本文中主要通过使用 Win32 多线程[6]、OpenMP[7] 以及两者结合的方法对程序进行了并行化，并优化了关键路径上比较耗时的循环体的结构，进一步提高了程序的性能。在单站程序每台机器处理 8 个雷达站点，组网程序一台机器对所有的 134 部雷达进行组网生成 BREF 产品的条件下，程序的并行版本和串行版本的运行时间对比如图 4 所示。从并行版本和串行版本的性能对比图示中可以看出，程序的时间主要花费在组网程序上，单站程序的并行版本并没有太大的性能提升，而组网程序的加速效果较好，有接近线性的性能提升。

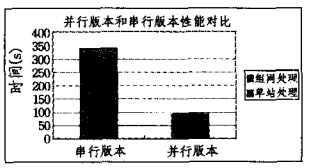

图 4 程序并行版本和串行版本的对比结果

4 实验内容及结果对比

4.1 单站产品的处理

单站产品串行版本处理流程以及并行化版本的数据处理流程如图 5 所示。

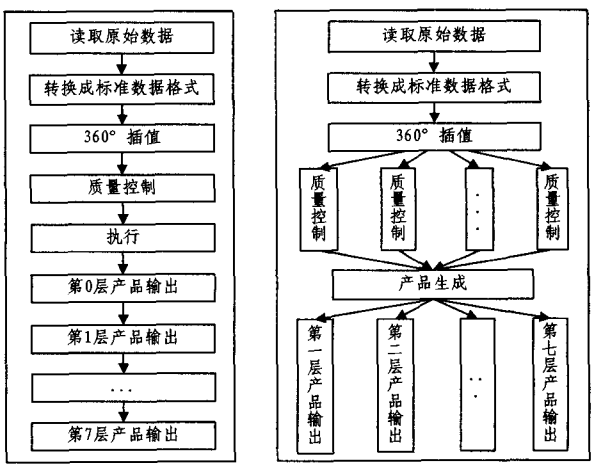

图 5 单站数据处理流程(左)和其并行化流程(右)

在图 5 中，转换成标准数据格式和 360°插值两部分对应于图 1 中的预处理，我们已经分析得出这两部分不适合并行化处理。质量控制和 bzip2 压缩输出部分占用了大部分的计

算时间，且有较好的级并行性，因此决定将这两部分进行并行化。因为质量控制计算时间本来就相对较短，并且每部雷达的每个数据包都要通过单站处理来生成单站产品，这会导致单站程序处理流程有着大量的重复执行次数，所以这里采用最底层的 Win32 系统线程来做并行化工作[8]，以期最小化并行化的开销。同样的理由，压缩输出部分也采用了 Win32 多线程的方法进行并行化。

在质量控制部分，有若干可以进行并行化操作的代码片段。在此举一例进行说明，代码片段如下：

```
for k=0 to n:
{
    ap(Output,ref[k],ref[k+1],other_args);
  //函数对 ref[k]进行修改，ref[k+1]保持不变
}
```

可以看出这两次循环之间有关于 ref[k]的读后写依赖，不能直接并行化，可以通过增加存储空间来存储 ref[k]中的值，以此来消除两次循环之间的依赖[9]。修改后代码循环之间没有依赖，易于并行，如下所示：

```
ref_copy[n]; //增加存储空间
for i=0 to n-1:
{
  ref_copy[i]=ref[i+1];//拷贝数据
}
for k=0 to n;//修改代码，可并行化
{
  ap(Output,ref[k],ref_copy[k],other_args);
}
```

在现实中，一台刀片进行 15 个雷达站点的单站处理，一台刀片包含两个四核的处理器，对应到实验平台上就是在一台机器上进行 8 个雷达站点的单站处理，并行版本和串行版本的性能对比如图 6 所示，并行版本的性能比串行版本平均要好 15%。

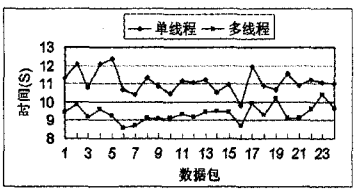

图 6　一台机器处理 8 个站点的性能对比

假设在硬件资源增加一倍的情况下，一台机器上进行 4 个雷达站点的单站处理。并行版本和串行版本的性能对比如图 7 所示，并行版本的性能比串行版本平均要好 24%，可以看出，在硬件资源更多的情况下，并行版本相对于串行版本的性能优势更加明显。

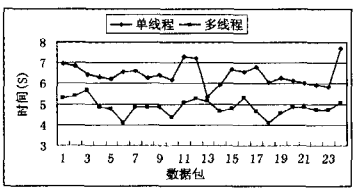

图 7　一台机器处理 4 个站点的性能对比

从图 7 中可以看出：单站程序并行版本的性能并没有很大的提升。这是因为多个进程同时运行已经可以比较好地利用处理器，这时处理器资源成为瓶颈，因此单站处理程序的并行版本相对于串行版本在更好地利用 CPU 这一方面已经没有很大优势了。不过并行版本相对于串行版本仍然具有两个优势[10]：1)更好地扩展性。当增加计算资源时，并行版本可以更好地利用这些资源，并且相比于串行版本的性能优势更明显。2)并行版本最短的执行时间要比串行版本少 55%。如果要提升实时性限制或者增加计算量，并行版本可以较好地满足要求。

4.2　组网 BREF 产品的并行化方法

BREF 产品计算过程如下：

```
for k=0 to K:
{
  Load 第 k 层计算所需的数据
  对第 k 层进行组网
  输出第 k 层的组网结果
}
```

从以上代码中可以看出这个计算过程比较容易并行化。我们提出了以下两种并行化方案。

并行方案 1

```
for k=0 to K:
{
    启动新线程载入第 k 层数据并进行组网和输出
}
```

并行方案 2

```
for k=0 to K:
{
  Load 第 k 层计算所需的数据
  启动新线程对第 k 层进行组网并输出
}
```

方案 2 相比于方案 1 的优势是开发了在“读入数据-组网计算-输出结果”这个层面上的并行性；将读入数据和组网计算并输出这两部分的时间重叠，掩盖了数据读入时间。

但是像方案 2 这样做可以利用的最大处理器核心数目有限，因为总共启动 7 个线程使用 7 层的数据来分别生成 7 层的产品，而且线程是在数据载入完毕后才启动的，这样第一个计算线程启动时，只有两个线程在运行，处理器利用率较低。还可能出现最早启动的线程已经执行完成，而后面的线程还没有启动的情况，这样最大的同时运行的线程数目甚至无法达到 7，所以如果在更多处理器核心(比如 8 或者 16 个)的机器上，这个并行化版本就会出现不能完全利用所有处理器核心计算能力的情况，扩展能力有限。而像方案 1 那样虽然可以同时利用 7 个处理器核心，但是因为所有的线程同时载入“载入数据-组网计算-输出结果”这一步骤，会造成处理器和 IO 资源的忙闲不均，失去了计算和 IO 重叠的好处，并且会造成 IO 或者处理器资源的过度繁忙，导致更长的等待时间，因而性能会比方案 2 有所下降。

因此最终的并行化方案采用了方案 2＋OpenMP 的方法，用 OpenMP 开发了组网计算部分中更细粒度的并行性[11]。

从图 8 中可以看出用 OpenMP 在多线程方案上并行化并没有减少很多的时间，这主要是因为计算资源已经成为多线程方案的瓶颈，但是 OpenMP 方案的优势是大大增加了程序的可扩展性。

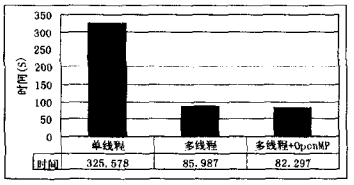

图 8　组网 BREF 产品的串行方案、多线程方案以及多线程+OpenMP 方案的性能对比

4.3　组网 CREF 的并行化方法

组网 CREF 产品的计算过程类似于 BREF 产品的计算过程，不同之处在于 CREF 只是使用一层的单站数据进行计算，因此该产品的并行化类似于用 OpenMP 并行化 BREF 产品的计算过程。

从图 9 中可以看到，用 OpenMP 并行后的程序执行时间大约是原来的 1/4，加速比接近线性，达到了较好的效果。

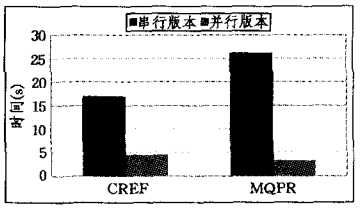

图 9　CREF 和 MQPR 产品计算时间对比

4.4　组网 MQPR 产品的并行化方法

组网产品 MQPR 的生成过程是先从 01－07 层数据中搜索最低高度的回波值，然后保存到 00 层的数组中；再对 00 层数组进行处理，就会得到当前时刻的组网降水率估计 MQPR。

本文中通过牺牲空间来换取时间的方法对 MQPR 的生成过程进行了并行化，也即每个线程负责搜索一层数据，然后把一些冗余结果也写入第 00 层，最后对第 00 层数据做统一处理[12]。从图 9 可以看出，并行版本的时间约为串行版本的 1/8，为超线性加速[13]，主要原因是原 MPQR 串行程序的 for 循环中存在一些冗余计算，我们在并行之前对循环体进行了优化，从而大大减少了程序的计算量。

4.5　通过调整刀片的计算负载优化程序

对一个程序进行并行化应该从整体上考虑，最先将最耗时的过程进行并行，然后才考虑其他不太耗时的过程。从图 1 中可以看出，天气组网雷达定量估测降水程序中占用时间比例最大的部分是组网 BREF 产品的生成和压缩输出部分，而且这一部分有很明显的并行性：7 层组网中每层组网的计算和结果输出是完全独立的，很容易将计算分散到多台刀片上进行，而且这个过程中的主要限制是计算资源，因此如果想要更快地完成整个过程，可以考虑在这部分增加计算能力[14]。而对于单站程序来说，如果数据包随机到来，可能会造成一种无法完全利用处理器资源的情况。综合这两方面的情况，得出了一个带有调整计算资源分配的并行化解决方案：减少进行单站处理的刀片数量，增加计算基本组网反射率 BREF 的刀片数量，以求尽可能地减少总的执行时间。

测试结果如图 10 所示，总共测试了 3 种方案的执行时间，左侧方案为原始方案，对 9 台刀片进行单站计算、一台刀片进行组网计算的结果；中间方案则调整为对 6 台刀片进行单站计算、4 台刀片进行组网计算的结果；右侧方案是对 9 台刀片进行单站计算，增加 3 台刀片共 4 台刀片进行组网的结果。可以看出：在不增加计算资源的前提下，通过调整用于单站计算和组网计算的刀片数量，可以降低大约 30％的总体执行时间，而如果增加用于组网 BREF 的计算资源，则可以减少大约 45％的总体执行时间。

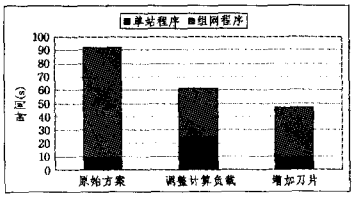

图 10　3 种不同的并行化方案时间对比

结束语　通过多种优化方法，本文在不添加新的硬件的条件下，将国家气象局天气组网雷达定量估测降水程序的执行时间由原来的 336 秒提升到 61 秒，加速比达到 5.5，取得了较好的效果。

本文中的天气组网雷达定量估测降水程序的并行化方法，代表了计算密集型且数据吞吐量较大的一类应用的并行化方案的实现。针对这一类程序的特性，均可使用本文中的方法加以并行化。

从程序整体上来考虑程序的并行化工作，综合考虑整个计算任务在所有计算资源上的分配，尽量做到负载平衡，可以取得较大的性能提升。同时，根据 Amdahl 定律[15]，要想程序并行化后性能提升尽可能大，应该先将最耗时的过程进行并行，然后才考虑其他不太耗时的过程。

未来的工作将从以下 3 个方面为出发点，继续改进程序的性能。1)并行版本的代码虽然比串行版本有较大的性能提升，但是仍有提升空间，可以考虑挖掘更深层次的并行性；2)考虑如何统筹利用现有的计算资源，以及如何分配新的计算资源，以使得整个过程的运行时间达到最小；3)在真实的平台上进行测试和改进程序。

参 考 文 献

[1] Diamond J, Robatmili B, Keckler S W, et al. High Performance Dense Linear Algebra on a Spatially Distributed Processor[C]// Ppopp'08: Proceedings of the 2008 Acm Sigplan Symposium on Principles and Practice of Parallel Programming. 2008: 63-72

[2] Intel® Parallel Studio 2011 Reference Manual

[3] Introduction to Performance Analysis on Intel® Core™ 2 Duo Processors

[4] Akhter S, Roberts J. Multi-core programming: Increasing performance through software multi-threading [S. l.]. Intel Corporation, 2004: 139-175

[5] Stratton J, Stone S, Hwu W. An Efficient Implementation of CUDA Kernels for Multi-core CPUs[C]// Languages and Compilers for Parallel Computing: 21th International Workshop. Edmonton: Springer-Verlag, 2008: 16-30

[6] High-Performance Recipes for IA-32 Platforms(Second Edition)

[7] OpenMP Guide Reference Manual(C/C++ Edition). Version 3.6. Kuck & Associates, Inc

[8] Ja Ja. An introduction to parallel algorithms[M]. Addison-Wesley Publishing Company, 1992

[9] Atallah M J, Cole R, Goodrich M T. Cascading divide-and-conquer: A technique for designing parallel algorithms[J]. SIAM J. Comput, 1989

[10] Sun X H, Rover D T. Parallelism in computation problems[J]. SIAM J. Comput. IEEE Trans. on Parallel and Distributed Systems, 1994

[11] Horowitz E, Zorat A. Divide and conquer for parallel processing [J]. IEEE Trans. On Comput, 1983

[12] Shiloach Y, Vishkin U. Finding the maximum, merging and sorting in parallel computation model[J]. J. of Algorithms, 1981

[13] Valiant L G. Parallelism in comparison problems[J]. SIAM J. Comput, 1991

[14] Malyshkin V. Parallel Computing Technologies [C]// Proc of PACT'05. Krasnoyarsk, Russia: [s. n.], 2005

[15] Hill M, Marty M. Amdahl's Law in the Multicore Era[J]. IEEE Computer, 2008, 41(7): 33-38

(上接第 255 页)

文献[19]聚类行为模式成为中间语义行为，然后标注低内聚性聚类为异常行为，但这一方法难以实现在线异常检测。文献[15]可实现在线检测，但其仍需要观察到整个行为模式后才能判决。本文提出的在线异常检测和正常识别方法能够实现实时检测，其通过延时决策解决由于缺乏充分视觉证据而引发的行为类型歧义问题。

结束语 提出了一种具有层次结构的语义主题模型——主题隐马尔科夫模型（Topic Hidden Markov Model, THMM），它从中间语义描述的层次结构角度捕获运动词语的共现信息、动作的共现信息以及行为之间的时序信息；解决了 PLSA 和 LDA 等语义主题模型没有建模“运动词袋”之间的关联关系，从而难以确定“词袋”采集时间窗口大小的问题；其不但聚类运动词汇成简单动作，而且聚类简单动作成全局行为，同时建模了行为时间上的相关性。本模型构建两个具备层次结构的语义主题空间，使语义主题表示具备更强的判别力。在取自实际监控场景的实验数据集上的性能比较说明，本模型无论是在运动词包描述还是中间语义建模方面均能取得最好性能。

参 考 文 献

[1] Wang L, Hu W M, Tan T N. Recent developments in human motion analysis[J]. Pattern Recognition, 2003, 36(3): 585-601

[2] Johnson N, Hogg D. Learning the distribution of object trajectories for event recognition[J]. Image and Vision Computing, 1995, 14(8): 609-615

[3] Brand M, Oliver N, Pentland A. Coupled hidden markov models for complex action recognition [C]// Proceedings of IEEE International Conference on Computer Vision. San Juan, Puerto Rico: IEEE, 1997: 994-999

[4] Dollar P, Rabaud V, Cottrell G. Behavior recognition via sparse spatio-temporal features [C]// Proc. 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. China, 2005: 65-72

[5] Hongeng S, Nevatia R. Multi-agent event recognition[C]// Proceedings of Eighth International Conference on Computer Vision. Vancouver, BC, Canada: IEEE, 2001: 84-91

[6] Russo R, Shah M, Lobo N. A computer vision system for monitoring production of fast food [C]// Proceedings of The 5th Asian Conference on Computer Vision. Vancouver, Melbourne, Australia, 2002

[7] Wren C, Azarbayejani A, Darrell T. Pfinder: Real-time tracking of the human body[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(7): 780-785

[8] Haritaoglu I, Harwood D, Davis L S. W4: Who when where what a real time system for detecting and tracking people[C]// Proceedings of International Conference on Face and Gesture Recognition. Nara, Japan: IEEE, 1998

[9] Johnson N, Hogg D. Learning the distribution of object trajectories for event recognition[J]. Image and Vision Computing, 1995, 14(8): 609-615

[10] Brand M, Oliver N, Pentland A. Coupled hidden markov models for complex action recognition[C]// Proceedings of IEEE International Conference on Computer Vision. San Juan, Puerto Rico: IEEE, 1997: 994-999

[11] Medioni G, Cohen I, Bremond F. Event detection and analysis from video streams[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(8): 873-889

[12] Naphade M R, Huang T S. A probabilistic framework for semantic indexing and retrieval in video [C] // Proceedings of IEEE International Conference on Multimedia and Expo. New York, NY, USA: IEEE, 2000: 475-478

[13] Zhong H, Shi J B, Visontai M. Detecting Unusual Activity in Video [C]// Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington, D. C., USA: IEEE, 2004: 819-826

[14] Hamid R, Johnson A, Batta S, et al. Detecting Unusual Activity in Video [C] // Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Diego, CA, USA: IEEE, 2005: 1031-1038

[15] Boiman O, Irani M. Detecting Irregularities in Images and in Video [C] // Proceedings of Tenth IEEE International Conference on Computer Vision. Beijing, China: IEEE, 2005: 462-469

[16] Fleischman M, Decamp P, Roy D. Mining temporal patterns of movement for video content classification [C]// Proc. 8th ACM Int. Workshop Multimedia Information Retrieval. 2006: 183-192

[17] Xing E P, Yan R, Hauptmann A G. Mining associated text and images using dual-wing harmoniums [C] // Proc. Uncertainty Artifical Intelligence. 2005

[18] Xiang T, Gong S. Beyond tracking: modelling activity and understanding behaviour[J]. International Journal of Computer Vision, 2006, 67(1): 21-51

[19] Niebles J C, Wang H C, Li F F. Unsupervised learning of human action categories using spatial-temporal words[J]. International Journal of Computer Vision, 2008, 79(3): 299-318

[20] Zhong H, Shi J, Visontai M. Detecting unusual activity in video [C]// Proceedings of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2004: 819-826