片上多核中一种共享感知的数据主动推送Cache技术

王得利，高德远
（西北工业大学计算机学院，710072，西安）

摘要：针对片上多核处理器的二级Cache访问延时持续增加以及并行程序在运行时线程间执行速率差异大的问题，提出了一种基于共享感知的数据主动推送Cache技术(SAAPC). SAAPC技术充分考虑并行程序的系统性能由速度最慢的线程所决定这一重要特性，根据并行线程间读数据共享程度高以及共享读数据访问局部性好的特征，采用基于指令的方法来预测共享读数据流，在后行线程需要共享数据之前将其主动推送至该线程的一级Cache中去，从而减少较慢线程的数据访问延时，提高执行速率，降低较慢线程与先行线程间执行速率的差异. SAAPC技术避免了预取技术所带来的额外片外带宽增加的缺点. 使用SESC模拟器对来自于SPLASH2测试程序集的5个存储敏感型并行程序进行了测试仿真，结果表明，与传统的共享Cache相比，使用SAAPC技术减少了并行线程间执行速率的差异，系统的每周期指令数平均提高了7%，最高达到13.1%.

关键词：片上多核处理器；并行程序；共享感知；主动推送；执行速率
中图分类号：TP302 **文献标志码**：A **文章编号**：0253-987X(2010)10-0018-06

A Sharing-Aware Active Pushing Cache Technology on Chip-Multiprocessor

WANG Deli, GAO Deyuan
(School of Computer Science and Technology, Northwestern Polytechnical University, Xi′an 710072, China)

Abstract: A sharing-aware active pushing Cache technology (SAAPC) is proposed to solve the problems of the increasing L2 Cache latency and the high deviation of progressive rates among the simultaneous threads in parallel applications when they are running on chip multi-processors. SAAPC fully takes the important characteristic into consideration that the whole system performances of parallel applications are constrained by the slowest thread in parallel phases. Based on the high share degree of read Cache blocks among different threads and the locality of the shared read accesses, SAPPC exploits the program counter to predict the shared data streams. The shared data are actively pushed to the L1 Caches of slower threads before it is needed so that the data access latencies for the slower threads are reduced and the progressive rates are increased. Therefore, the deviation of progressive rates is decreased. SAAPC avoids the problem caused by increasing off-chip bandwidth demand of the prefetch technique due to its inaccuracy. 5 memory intensive parallel programs from SPLASH2 benchmark suit are simulated using the simulator called SESC. Experimental results and comparisons with conventional shared Cache show that the SAAPC reduces the progressive rates deviation, and the average system instruction per cycle improvement is 7% and can be up to 13.1%.

Keywords: chip multi-processor; parallel program; sharing aware; active push; progressive rate

由于传统超标量单核处理器在继续发掘指令级并行性时遇到瓶颈及其在功耗和散热上存在问题，

收稿日期：2010-03-06. 作者简介：王得利(1981—)，男，博士生；高德远(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60773223，60736012)；国家“863计划”资助项目(2009AA01Z110).

因此体系结构研究的趋势已转向片上多核处理器(CMP)[1]. CMP 可在一个芯片上集成多个设计较为简单的核,通过挖掘线程级并行性来提升系统性能. 并行程序因本身富含的线程级并行性,必将在 CMP 时代获得更多的利用[2-3].

然而,在 CMP 中由于多个线程并发执行,存储访问频率相比单核有显著增加,因而对存储系统的带宽提出了更高的要求. 目前的 CMP 多采用大容量的 Cache 以满足多个核的同时访存请求,但是由于多个核 Cache 访问请求的竞争以及 Cache 容量的增大导致了片上 Cache 访问延时的增加,特别是二级 Cache 访问延时的增加[4],从而造成一级 Cache 缺失代价太大,系统性能下降. 以往优化二级 Cache 延时的技术在应用于并行程序时没有考虑它的一个重要特性,即并行程序在某个并行阶段执行时,该阶段的系统性能由所有线程中执行最慢的那个来决定,片面地对所有线程同等优化反而可能因较慢线程获得更少的共享资源而使整体性能下降.

本文从研究并行程序在 CMP 上运行的特点入手,通过并行测试程序在模拟器上的运行情况,得出并行程序在并行阶段多个线程间执行速率差异性大、并行程序的读数据访问共享程度高、共享读数据访问局部性好这 3 个主要特点,提出了基于指令的共享感知的数据主动推送 Cache 技术 SAAPC. 该技术可更智能地存储、传输和管理数据,从而减少后行线程二级 Cache 的有效访问延时,减少并行线程间的执行速率差异,提升并行程序在 CMP 中执行时的系统性能.

1　并行程序在 CMP 上运行时的特点

要提升并行程序在 CMP 的执行性能,必须首先了解并行程序在 CMP 上运行时的特点. 本文选用来自于科学计算基准测试程序集 SPLASH2[5] 的 5 个存储敏感型并行测试程序来了解并行程序的特点,通过这些程序在 SESC 模拟器上的仿真(模拟器的具体配置在第 4 部分给出)以及其他已有研究数据,发现并行程序有以下 3 个显著特点.

(1)并行程序在并行阶段多个线程间执行速率差异性大. 并行程序由多个交替的并行执行阶段和顺序执行阶段组成. 在并行执行阶段,有多个线程同时活动,并发地执行计算任务,而在顺序执行阶段,在同一时间只能有一个活动的线程,一般进行线程间同步操作. 并行程序之所以能够比同样任务的单线程程序性能有所提高,就是因为在并行执行阶段,在各个线程负载均衡的情况下,每个线程只处理原任务的一部分,所有并发线程齐头并进,从而使计算时间大幅降低. 但是,如果任务在线程间已得到了很好的均衡分配,各个线程却因对 CMP 共享资源争用,出现部分线程占用了较多资源(比如 Cache 空间、片外带宽),其存储请求响应相对及时,执行速度会比那些竞争资源占用率低下的线程快. 这种线程间的执行速率差异对并行程序的系统性能是有害的,因为在任何一个并行阶段,该段的执行时间是由执行最慢的线程来决定. 在优化 Cache 访问时,如果单纯加速速度最快的线程并不能获得性能提升,反而因为最慢线程的共享资源的利用率进一步下降而有损性能的提升. 在对 SPLASH2 的测试程序仿真时,发现所有 5 个测试都存在并行执行阶段线程间的执行速率差异大的情况,表 1 给出了在选定的并行阶段,最快线程与最快线程的执行速率比值,从中可以看到与期望值 1 相比,各个并行程序的比值均大于 1,且 Cholesky 的值接近 2,也就是说它最快的线程有 50%时间都浪费在等待最慢线程的完成上. 因此,在并行执行阶段并行程序多个线程间执行速率差异大,需要减少这种差异,从而提高系统性能.

表 1　测试程序中最快线程与最慢线程执行速率比值

测试程序	Cholesky	FFT	LU	Ocean	Radix
速率比值	1.97	1.17	1.34	1.12	1.33

(2)并行程序多线程之间的读数据访问共享程度高. 并行程序中由于线程之间并不完全独立,它们可能需要对同一套数据的不同部分进行相同处理,也可能是对同一部分做不同处理,同时也需要数据进行线程间通信. 同一数据被不同线程访问称为共享数据,只被一个线程访问称为私有数据. 这里只关注对共享数据的读访问,因为在程序执行时,读数据缺失后可能很快就导致整个处理器的暂停,而写数据却不一定. 本文从文献[5]中抽取出 5 个反映测试程序读访问情况的数据,如表 2 所示. 从表 2 中可以看出,所有测试程序的共享读访问次数占读访问总次数的比例均超过 60%,平均共享读比例超过了 90%. 由此可见,并行程序多线程之间的读数据访问共享程度高,对共享读数据的优化存在很大的余地.

(3)并行程序在多线程间共享读数据的访问局部性好. 这里的局部性是指共享读数据被先行线程首次访问后在一定时间范围内会再次被其他线程访

表2 测试程序中的共享读访问次数与读访问总次数

测试程序	共享读访问次数	读访问总次数	共享读访问次数占读访问总次数的比例/%
Cholesky	75.87	111.86	67.83
FFT	4.05	4.07	99.51
LU	93.20	104.00	89.62
Ocean	80.26	81.19	98.85
Radix	12.06	12.06	100.00

问. 对5个测试程序在8个线程运行时的二级Cache访问进行了足迹分析,计算了从共享数据首次被某个线程访问而引起的二级Cache块的读缺失开始到该数据块被其他线程访问而标记成共享读数据块之间发生的所有二级Cache缺失的次数,也就是该共享数据从首次二级Cache请求发起,到数据被载入二级Cache并被其他线程访问的相对时间间隔,这里称为该共享数据的共享访问间隔. 如果共享访问间隔较小,则说明在先行线程访问共享数据之后,仅经过数个或者数十个二级Cache访问缺失的时间,该共享数据便被慢线程访问. 表3给出了测试程序共享访问间隔小于64的共享读数据占所有共享读数据的比例,可以得知所有5个程序的80%左右的共享读数据的共享访问间隔都小于64,这说明共享读数据访问的良好数据的共享访问间隔都小于64,这反映共享读数据访问的良好局部性.

表3 测试程序共享访问间隔小于64的共享读数据占所有共享读数据的比例

测试程序	Cholesky	FFT	LU	Ocean	Radix
比例/%	89.11	79.23	78.34	79.96	77.49

2 SAAPC技术

2.1 SAAPC技术的基本思想

SAAPC技术的基本思想源自于并行程序的3个特点. 当并行程序在CMP上运行时,如果后行线程需要访问共享数据,尽管该数据已被先行线程取入二级共享Cache,后行线程访问共享数据延时已较直接访问片外内存有所减少,然而仍然有提高的余地. 在共享数据被先行线程取入二级Cache后,如果能够侦测出该共享数据被哪些后行线程所访问,可以提前将该数据主动推送至这些后行线程的一级Cache. 根据共享数据读访问局部性好的特点,后行线程在一段时间内会访问这些共享数据,由于共享数据已被推送至它的一级Cache,后行线程将发生一级Cache访问命中,不用再访问延时更大的二级Cache,因此后行线程共享读数据访问速度加快. 又因为共享读数据占整个读数据比例非常大,所以后行线程执行速率获得有效提高. 这种做法的另外一个好处是减少了后行线程二级Cache的访问次数,将降低所有线程的二级Cache请求冲突,从而减少二级Cache的请求排队延时,也就相应地减少了二级Cache访问的有效延时. 这就是SAAPC技术的基本思想,同预取技术相比,SAAPC技术并不会过早地发出数据读请求,而是在共享数据被先行线程取入二级Cache的同时被推送至其他线程的一级Cache中,没有增加额外的片外访存,因此不会增加对片外带宽的额外需求.

2.2 SAAPC技术的工作流程

SAAPC技术需要找到共享数据,也就是需要预测共享数据. 共享数据的预测方法可分为2大类,一类是基于地址历史的方法[6],通过地址的历史访问足迹去推测下一个共享数据的地址,但是这种方法的硬件开销较大,且只能预测出比较规范的数据结构,对于链表等结构支持的不好. 另一类是基于指令的方法[7],所依据的原理是:在一个循环体Loop中,在某次迭代中某条指令对应地址的数据是共享的,下次迭代中该条指令对应的下个地址的数据有很大的可能性还是共享的,可以找到共享数据所对应的指令. 这种方法将不再受限于数据结构形式,但是其难点在于在二级Cache中很难捕捉到某个数据到底是由哪条指令发起的. 本文提出的解决方法如下:在一级数据Cache缺失时,将指令信息与对应物理地址一起保存至一级缺失状态寄存器表MSHR中,访问二级Cache时,将一级MSHR表中所对应的指令信息与物理地址及操作类型也一并送出,这样二级Cache中也可获悉物理地址与指令之间的对应关系.

SAAPC技术的工作流程由以下4个步骤组成. ①先行线程在执行Loop中的第i次迭代,率先访问某个共享数据,此时将先后引发一级及二级Cache缺失,二级Cache缺失后内存访问请求发起,并在二级MSHR表中写入相关信息,包括其对应的指令程序计数器PC、线程号CoreID、操作类型OpType和物理地址Paddr. ②因为共享读数据访问具有局部性,所以后行线程在一定时间内会对同一个

地址再次访问,查找二级 MSHR 表,发现不同线程对相同地址的访问. 发现后,将相关信息放置在一个共享数据指令信息寄存器表(Shared Data Instruction Holding Register,SDIHR)中,包括先行线程号 PreID,先行线程指令程序计数器 PrePC、当前物理地址 LPaddr 及后行线程号 PostID. ③ 先行线程执行 Loop 中的 $i+1$ 迭代,访问下一个共享数据,再次引发二级 Cache 缺失,并在二级 MSHR 中新分配一个条目. 查找 SDIHR 表,发现 SDIHR 某条目的 PreID 值与先行线程的线程号相同、PrePc 值与新分配 MSHR 条目中的 PC 值相同且其 LPaddr 与 MSHR 条目中 Paddr 不一致时,则表明该指令在读取下一个可能的共享数据. 当该数据从片外返回时,将其送入 PreID 指示的线程的一级数据 Cache 中,同时将该数据推送至 PostID 指示核的推送缓冲 PushBuf 中,PushBuf 位于一级 Cache 内. ④PostID 指示的后行线程在执行中,遇到 load 指令,将同时访问 PushBuf 和一级数据 Cache,当 PushBuf 命中时,将其数据挪至一级数据 Cache 内,清空该缓冲条目,并将信息反馈给 SDIHR 表. SAAPC 技术的流程如图 1 所示.

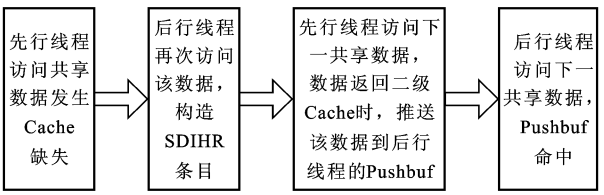

图 1　SAAPC 技术的工作流程

2.3　SAAPC 技术的 2 个重要指标

SAAPC 技术有效与否,与共享读数据检测的覆盖率与准确率密切相关. 共享读数据预测的覆盖率由所能捕捉到的共享读数据指令数目决定,而所能捕捉到的共享读数据指令数目又直接与 MSHR 条目的活动周期相关. 延长 MSHR 条目的活动周期,将可以提高共享读数据预测的覆盖率. 为此,本文修改了 MSHR 的提交释放条件,在二级 MSHR 表中增加了 Match 位,当 MSHR 某条目与另外一个线程二级 Cache 请求中的物理地址相同时,将 Match 位置位,当数据由片外内存中返回时,该 MSHR 项可以被提交并释放. 如果当数据由片外内存中返回时,Match 位尚未被置位,则该 MSHR 项暂时不会被释放,直至该 MSHR 条目中的 Match 位被置位,或者该 MSHR 条目最久未被释放且 MSHR 表无空余项. 需要注意的是,无论一个新到的二级 Cache 请求是否命中,都将访问 MSHR 表,以查找共享情况,通过延长 MSHR 条目在该表中的活动周期,将指令与数据的对应关系延长,从而保证捕捉到足够的共享数据指令. 此外,MSHR 条目的深度也决定了可以被捕捉到的共享数据指令的个数,从表 3 可以得知,80%的读共享数据的共享访问间隔都在 64 以下,因此将 MSHR 条目设置为 64 可以保证捕捉到大部分共享指令.

共享数据预测的准确率由 SDIHR 表中各条目的置信度所决定. 置信度低的共享指令所对应的推送数据被后行线程所访问的概率较低,则该共享指令 SDIHR 条目应该被置无效. 为此,在 SDIHR 条目中加入了 DisCnt 域,并在 PushBuf 中加入了 SDIHRtag 域. SDIHRtag 域指示当前的 PushBuft 条目中的数据来自于哪一个 SDIHR 条目,DisCnt 域指明当前 SDIHR 条目所推送的数据还有几个未被使用. 当新分配一个 SDIHR 条目时,该 DisCnt 值置 0. 每推送一个新数据,就将其加 1,而该条目对应的 PushBuf 中每命中一次,则根据 SDIHRtag 值,将相应 SDIHR 条目中的 DisCnt 值减 1. 当 DisCnt 值超过一个最大值时,表明该 SDIHR 项置信度很低,该项负责推送的数据有很多未被使用,则将该条目置为无效. 在本文的实验中,SDIHR 表拥有 256 个条目,采用最近最久未使用算法(LRU)替换,每个核的 PushBuf 设置为 16 个条目,也采用 LRU 算法替换,DisCnt 域的最大值为 7.

3　SAAPC 技术的实现

SAAPC 技术的 2 个核心任务是查找到共享数据所对应的指令并将共享数据推送到后行线程的一级 Cache. 任务具体实现由 2 个相应模块组成,一个是共享数据指令检测引擎(Shared Data Instruction Detecting Engine,SDIDE),另一个是数据主动推送单元(Actively Pushing Data Unit,APDU),如图 2 所示. SDIDE 是其中的核心部件,它包含 SDIHR 表,并负责维护及更新. APDU 包含一个待推送数据先入先出缓冲 FIFO,并将其中数据推送至线程的 PushBuf 中.

4　仿真实验及结果分析

4.1　仿真环境的建立

本文采用目前使用非常广泛的多核仿真器 SESC 作为仿真平台. 本文对 SESC 中的相关源文件作了一些修改. 最终的实验配置如下:采用了 8 个

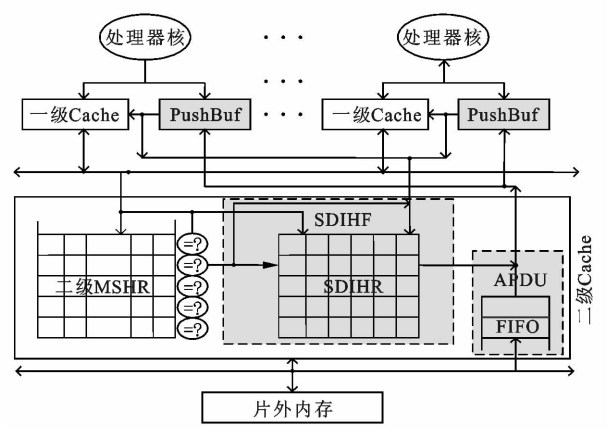

图 2 SAAPC 技术中一个具体实现的组成结构及其在处理器中的位置

频率为 3.3 GHz 的乱序执行的超标量核，该核发射、译码、提交均为 4 条指令，整数和浮点单元各 2 个，物理寄存器个数为 80，load/store 队列的条目深度为 40，采用混合型分支预测. 一级指令 Cache 为 2 路组相连，大小为 64 kB，访问延时为 0.3 ns，行大小为 32 B. 一级数据 Cache 为 4 路相连，大小为 64 kB，访问延时为 0.6 ns，行大小为 32 B. 8 个核共享大小为 4 MB 的 16 路组相连二级 Cache，访问延时为 5.1 ns，行大小为 32 B. 内存访问延时为 150 ns. 本文选取了 5 个存储密集型的测试程序. 表 4 显示了这些测试程序以及其输入数据集.

表 4 测试程序及输入数据集

测试程序	输入数据集
Cholesky	Tk28.0
FFT	20 Million
Lu	512×512 matrix，16×16 blocks
Ocean	258×258 array
Radix	2M keys

4.2 仿真结果及分析

对表 4 的 5 个测试程序的仿真结果如表 5 所示. 与未使用 SAAPC 技术的传统共享 Cache 相比，改进后各程序的最快线程与最慢线程的执行速率比值较表 1 有所减小，因此并行线程间执行速率差异也有所减小，从而有利于整体并行程序性能的提高. 由表 5 可知，与未使用 SAAPC 技术的平台相比，各程序每周期指令数(IPC)都获得了提升，平均提升为 7%，最高为 13.1%. 表 5 还给出了每个程序 PushBuf 的命中情况，可以看出，各 PushBuf 的平均命中率都较高，均超过了 70%，较高的命中率，保证了推送数据的有效利用.

表 5 各程序仿真结果

测试程序	并行阶段最快线程与最慢线程的执行速率比	IPC 提高率/%	推送缓冲平均命中率/%
Cholesky	1.69	7.7	82.1
FFT	1.06	13.1	95.1
LU	1.26	2.7	78.0
Ocean	1.08	2.3	83.7
Radix	1.17	9.2	82.9

5 总结及未来工作

本文依据并行程序在 CMP 上执行时读数据访问共享程度高以及共享读数据访问局部性好的特点，提出了一种共享感知的数据主动推送 Cache 技术 SAAPC，采用基于指令的方法来预测共享数据，当共享数据发生 Cache 缺失时，在先行线程将数据从片外内存取入二级 Cache 后，将该数据在适当的时机主动推送至其他需要该共享数据线程的一级数据 Cache 中. 实验结果表明，该技术可以减少并行程序线程间执行速率的差异，有效地提高了并行程序的性能. 但是，SAAPC 技术的共享数据检测覆盖率还受限于二级 MSHR 条目的深度，需进一步改进，同时各线程同步执行速率还有进一步优化的潜力，比如可以优化较慢线程的 Cache 空间利用率，并提高其片外带宽请求优先级，这是我们进一步的工作方向.

参考文献：

[1] KUNLE O, BASEM A, LANCER H. et al. The case for a single-chip multiprocessor [J]. ACM Sigplan Notices, 1996, 31(9):2-11.

[2] CHEN Yu, LI Wenlong, LIN Junmin, et al. Data sharing analysis of emerging parallel media mining workloads [C]// Proceedings of 2008 High Performance Computing. Berlin, Germany: Springer, 2008: 87-96.

[3] CHEN Yu, LI Wenlong, KIM C K, et al. Efficient shared Cache management through sharing-aware replacement and streaming-aware insertion policy[C]// Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing. Los Alamitos, CA, USA: IEEE Computer Society, 2009:1-11.

[4] BRADFORD M B, DAVID A W. Managing wire delay in large chip-multiprocessor Caches [C]// Proceed-

ings of the 37th annual IEEE/ACM International Symposium on Microarchitecture. Los Alamitos, CA, USA:IEEE Computer Society,2004:319-330.

[5] STEVEN C W,MORIYOSHI O. The SPLASH-2 programs: characterization and methodological considerations [C]// Proceedings of the 22nd Annual International Symposium on Computer Architecture. New York, USA: ACM, 1995:24-36.

[6] THOMAS F W, STEPHEN S, NIKOLAOS H, et al. Temporal streaming of shared memory [C]// Proceedings of the 32nd Annual International Symposium on Computer Architecture. Los Alamitos, CA, USA: IEEE Computer Society, 2005:222-233.

[7] STEFANOS K, JAMES R G. Improving CC-NUMA performance using instruction-based prediction[C]// Proceedings of the 5th International Symposium on High Performance Computer Architecture. Los Alamitos, CA, USA: IEEE Computer Society, 1999: 161-170.

(编辑　刘杨　武红江)

[优秀博士论文摘登]

图像匹配与跟踪研究

作者:吕娜　　导师:冯祖仁

单位:西安交通大学电子与信息工程学院

图像跟踪在军事目标跟踪、人机界面、驾驶员辅助系统等领域有着广泛的应用.现存的图像跟踪算法主要分为基于模板匹配的算法、基于运动信息的算法、基于滤波的算法和基于分类的算法.其中,基于模板匹配的算法最为直观,易于操作,应用最为广泛.基于模板匹配的图像跟踪算法有 3 个关键环节,分别为图像特征表示、相似度计算和目标搜索算法.本文主要针对第 2 个环节进行了深入的研究,并对第 1 和第 3 个环节进行了有益的探索.本文的主要贡献包括以下几个方面:

(1)提出了一种评价相似度指标匹配性能的图像匹配模型,并基于该匹配模型,对几种常用的相似度指标的匹配性能进行了分析和比较.图像匹配领域存在多种相似度指标,目前只能通过实验判断某种指标在实际问题中的匹配性能.为了能够从理论上对相似度指标的匹配性能进行分析比较,在分析匹配失败的实质原因的基础上,对图像特征进行分类,从而建立了一种包含位置变量的图像匹配模型.该模型可以用于评价相似度指标的匹配性能,指导我们选择或改进已有的相似度指标,以及创建新的相似度指标.基于该匹配模型,对多种常用相似度指标的匹配性能进行了理论分析和比较,给出了其各自匹配出现偏差和错误的条件,实际匹配实验验证了根据匹配模型所得的分析结果的正确性和匹配模型的有效性.

(2)在所建立的图像匹配模型的指导下,对现有的一种典型的相似度指标——巴氏指标进行了改进,提高了其图像匹配性能,并且建立了一种新的相似度指标——背景抑制指标.应用本文模型的实验证明,背景抑制指标的匹配函数关于匹配位置是线性关系,能够很好地抑制背景特征对相似度计算的影响.大量实验表明,背景抑制指标能够大大提高图像的跟踪精度,这从另一方面也验证了匹配模型的有效性.

(3)提出了一种后验概率指标,并以该指标为基础,提出了 2 种快速图像跟踪算法.经证明,在特殊参数时,背景抑制指标等价为特征匹配的后验概率.据此,建立了最大后验概率指标.该指标利用搜索区域的统计特征,能有效抑制待匹配区域特征中背景因素的影响,同时突出了目标特征的权重,与巴氏指标相比明显改进了匹配函数的峰值特性.这种指标的另一个突出优点是计算复杂度很低,容易得到全局最优解.经分析,该指标可以表示为像素相似度的线性求和.以此为基础,提出了 2 种基于后验概率指标的快速图像跟踪方法,快速平移算法和质心迭代算法.此外,还建立了一种变分辨和尺度自适应的跟踪方法.大量对比实验表明了本文跟踪算法的有效性.

(4)本文对一种重要的图像特征——角点进行了研究,提出了一种新的基于交点累积空间的角点提取算法.由于角点特征没有明确的参数定义方法,无法应用传统的 Hough 变换转换到参数空间进行计算.本文提出了一种新的 Monte Carlo 框架下的随机角点检测方法,不是在参数空间中求解,而是将角点检测转换为“交点累积空间”中寻找局部极值的问题.“交点累积空间”是本文根据角点实质上是直线交点的特征提出的一种新的概念.文中证明了算法的思想,推导了算法的具体步骤.该算法具有各向同性,对图像的旋转是鲁棒的,而且对噪声不敏感,并可以有效得避免斜边上伪角点的影响.大量实验表明,与 Harris 算法、Shen & Wang 算法、Wang & Brady 算法、SIFT 特征等算法相比较,新算法具有一定的优越性,并可成功地应用于图像跟踪.