基于 AVS+实时编码的多核并行视频编码算法

蒋骁辰 李国平* 王国中 赵海武 藤国伟
(上海大学通信与信息工程学院 上海 200072)

摘 要：面向 3D 高清的国家广电行业标准 AVS+较大地提高了编码效率，但也增加了编码复杂度，并行编码技术是解决高复杂度编码器的有效方法。该文改变了传统视频编码的反馈环，提出一种新的视频并行编码框架，有效地实现了编码的并行化；并将该框架应用于 AVS+实时编码器中，实现了一种基于 AVS+实时编码的并行算法。实验结果表明，该算法充分发挥多核处理器的运算能力，在编码性能损失极小，编码延迟可控的条件下极大地提高了视频编码速率。

关键词：视频编码；AVS+；并行编码；并行运动估计
中图分类号： TN919.8 **文献标识码**： A **文章编号**：1009-5896(2014)04-0810-07
DOI: 10.3724/SP.J.1146.2013.00845

Multi-core Parallel Video Coding Algorithm Based on AVS+Real-time Encoding

Jiang Xiao-chen Li Guo-ping Wang Guo-zhong Zhao Hai-wu Teng Guo-wei
(*Communication and Information Engineering Department of Shanghai University*, *Shanghai* 200072, *China*)

Abstract: The national TV industry standard AVS+, which faces the demand of 3D and high definition, got a high progress in coding efficiency, but it also increases complexity. Parallel coding technique is an efficient solution to deal with high-complex encoder. A new parallel video coding framework is proposed by changing the traditional video coding back feed loop to realize efficient parallel coding. The framework is applied to the AVS+ real-time encoder, and implements a parallel video coding algorithm based on AVS+ real-time coding. Experimental results demonstrate that, the algorithm makes full use of the computing ability of the multi-core processor, and improves the video coding rate significantly.

Key words: Video coding; AVS+; Parallel coding; Parallel motion estimation

1 引言

AVS 标准[1]作为我国制定的具有自主知识产权的音视频编解码标准，对我国数字化音视频产业的发展具有重要意义。AVS 与 H.264 相比大大降低了算法复杂度，然而编码效率也有不少降低。为了进一步提高 AVS 的编码效率，2012 年 3 月 18 成立了 AVS+工作推进组，在原有 AVS 标准基础上，在运动矢量预测、量化、熵编码等方面增加 4 项关键技术，形成了 AVS+视频编码标准[2]。AVS+标准于 2012 年 7 月发布成为广电行业标准，2013 年 3 月 18 日正式应用于卫星电视播放。

由于 AVS+新增了多项关键技术，虽然提高了编码效率，但其编码端的复杂度必然会大大提升，而且该标准主要应用于 3D 高清电视，这些都为编码器的实现提出极大的挑战，因此研究编码器加速的实现方法是 AVS+产业化的关键课题。

近年来，随着多核处理器浪潮的不断来袭，利用多核处理器的视频编码并行加速成为了标准制定以及编解码产品设计的主流方向。在快速发展的多核硬件系统以及并行工具的条件下[3]，需要有更好的并行编码的软件和算法来实现更加快速有效的视频编码功能。对于传统的视频编码系统，在 MPEG-2 之后引入了 slice 的概念，大大提升了编码系统的抗误码能力，而且逐渐出现了并行编码的理念，slice 的相关技术也一直应用于后续的并行编码算法。但是仍然存在对于边缘数据主要是边缘运动矢量和帧内预测的无法预测性，并且降低了并行编码效率。当前基于 H.264 的并行视频压缩是国内外学者研究并行编码的热点。Intel 研究中心的研究人员提出了基于 Intel 超线程体系结构下的 H.264 编码器的并行编码模型[4−6]。对于如何进行帧级并行化，slice 级并行化和宏块级并行化分别进行了讨论，并采用

2013-06-14 收到，2013-10-16 改回
电子信息产业发展基金(1213711)，国家自然科学基金(271212)和上海市科委重点项目(2511502502)资助课题
*通信作者：李国平 liguoping@shu.edu.cn

slice 级并行的方法进行实施，编码速度显著提高，加速比接近处理器核心的数目，这样做的后果是显著地增大码率。浙江大学和 Intel 中国研究中心的研究人员对 H.264 的离散余弦变换(Discrete Cosine Transform, DCT)和解码[7]进行了并行化研究，采用 Intel 的扩展指令集(Streaming Simd Extensions, SSE)的向量运算特性，对 H.264 计算量较大的变换和量化部分进行了指令集的优化。文献[8]中提出了将编码和运动估计并行处理，并且通过原始图像作为运动估计参考帧来充分解决依赖性问题，虽然这种并行处理提高了硬件的利用率，但是没有完全去除编码线程和运动估计之间依赖性。对于新一代视频编码标准 HEVC, JCT-VC (Joint Collaborative Team on Video Coding)也专门针对提高并行编码性能提出了有价值的技术，如 WPP (Wavefront Parallel Processing)和 Tiles [9]。WPP[10]技术简单来说只需要上一行的第 2 个最大编码单元(LCU)，编解码完毕即可以开始当前行的编解码，以此提高编解码器的并行处理能力。而 Tiles 技术[11]则是利用垂直和水平的边界将图像划分为一些行和列，划分出的矩形区域为一个 Tile，每一个 Tile 包含整数个 LCU 且 Tile 之间相互独立，以此来实现并行。上述两种技术只适用于 HEVC 标准，且都是利用空间并行化，因此并行化程度不高，且难以处理好码率分配问题。AVS+标准不支持 WPP 和 Tiles 并行化技术，只有 slice 编码可用于并行化计算，而帧内 slice 的并行编码不仅降低编码效率，而且也会产生条带效应[12]。

通过对上述传统并行编码技术的分析，我们发现传统的视频编码框架是影响并行编码的最主要因素，因此，本文改变了传统的编码框架，提出了一种新的视频并行编码框架。该框架包括并行视频预处理，并行运动估计，帧内/帧间并行编码结构，并将新的编码框架应用于基于 AVS+标准的实时编码器中，得到一种基于 AVS+实时编码的视频并行编码算法。本文实验表明，该并行编码算法有效地去除帧间参考相关性，从而充分利用了多核多处理器的运算能力，极大地提高了视频编码速度。

2　传统并行编码算法

传统的视频编码标准的语法结构层次由高到低依次为序列(sequence)，图像组(group of pictures)，帧(frame)，片(slice)，宏块(macroblock)和块(block)。一个视频序列可以包含一个或多个图像组，每个图像组总是由 I 帧开始。

2.1 帧内并行编码

一帧图像可以分为一个或多个片，每个片在编码时不能参考同个图像的其它片，这样可以限制误码的扩散。因此片间是编码独立的。由于标准中有其特有的片语法结构，因此帧内并行编码采用片级并行更易实现码流的封装，因为，在编码数据流中所有的视频序列头信息、图像头信息和片头信息都放置在输出码流的前部，所以主线程将各个任务线程的压缩数据流收集后，进行数据同步和封装更容易些[13]。

帧内片级并行编码的基本原理如图 1 所示。将 1 帧图像分成 4 个片，每个片通过 1 条工作线程进行编码，各工作线程之间并行工作，所有线程完成编码任务之后进行数据同步。虽然片级并行编码充分利用了语法结构特点实现编码并行化，且易于实现，但编码效率不高，且会产生条纹效应。

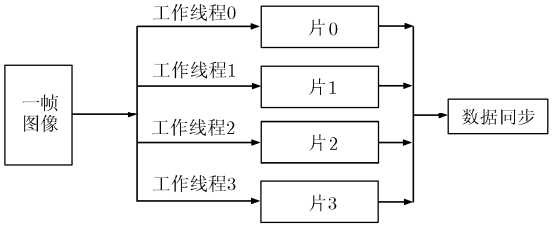


图1 帧内片级并行编码示意图

2.2 帧间并行编码

传统的帧间级并行编码模式主要利用几帧之间可共用参考帧的特点进行并行化编码，对于一个序列，在串行编码起始的 I 帧和 P 帧后，可以将使用相同参考帧的 P 帧和 B 帧同时进行编码。

如图 2 所示，假设输入视频为 IBBPBBP 的形式，第 1 个 I 帧 I(0)和 P 帧 P(3)必须串行编码，第 2 个 P 帧 P(6)和前两个 B 帧 B(1), B(2)，它们不需要相互参考，没有数据相关性，所以可以并行编码。这 3 帧的参考帧都是 I(0)和 P(3)，因此在串行编码完 I(0)和 P(3)后，我们可以分出 3 个线程来并行处理 P(6), B(1)和 B(2)。这 3 帧并行编码完成后，程序又回到主线程，准备编码 P(9), B(4)和 B(5)。这种并行方式的缺点在于没有很大程度地消除帧间相关性，从而并行度不是很高，且由于帧间较大的依赖性和参考性以及各个帧编码时间的不完全一致性，难免会在某些时候造成延时等待，削弱了系统性能。另外，其算法扩展性也不佳，它的加速比较大地受限于图像组 B 帧的数量。

2.3 具有参考帧依赖性的帧间并行编码

根据运动物体存在连续性的特点，视频图像中的宏块的搜索范围是有限的，因此无需等待当前帧全部编完，只要后面参考当前帧的待编码帧已经获得所需要的参考信息，它就可以同时进行编码[14]。

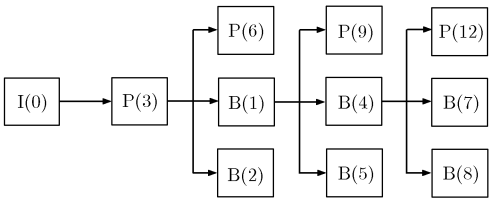


图2 帧间并行示意图

如图 3 所示，假设输入序列为 IBBPBBP，序列中的每帧都采用片级的并行编码方式进行编码，而在各帧之间，其基本编码顺序与传统帧间编码方式相同，只是每帧之间有时间上的延迟，而不是如图 2 那样各级之间是一种串行模式。在图 3 中，当编码第 1 个 I(0)时，只要 P(3)已经获得所需要的参考数据时它就能开始并行地进行编码，同理 B(1)，B(2)和 P(6)也不需要等待前面两帧编完就可以开始并行编码。但这种方式造成了参考图像之间存在依赖性，从而导致编码并行度的下降，为了提高编码速度，往往会采用加大并行帧数的方式，这样会导致比特数的分配不准。

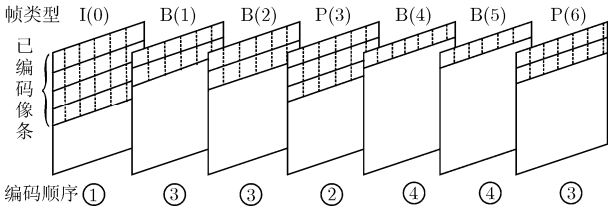


图 3 具有参考依赖性的帧间并行编码示意图

3 基于 AVS+实时编码的并行编码算法研究

3.1 传统视频编码框架的并行局限性

对于传统的视频编码框架，主要包括两条编码路径：前向路径和重构路径。原始图像 F_n 通过前向路径：预测(帧内或帧间)、整数变换、量化和熵编码获得编码码流，而重构路径：反量化、反变换、运动补偿来获得重建图像 F_n'，用于预测编码的参考帧。这样由前向路径和重构路径构成了整个编码反馈环。由于这个编码反馈环的存在，很难在这种结构下实现编码的并行性。因此我们需要改变传统的编码框架，设计一种新的并行编码结构。

3.2 改变编码反馈环的并行视频编码框架

在实际的视频编码器中一般都会存在一定的时间延迟，具体延迟时间根据实际的应用情况而定。基于这个特点，本文提出一种新的并行视频编码框架，充分利用这个延时特性，改变了传统的视频编码框架，充分让编码模块并行工作，设计了如图 4 所示的结构。

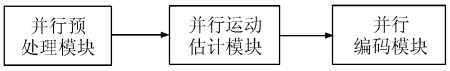


图 4 并行编码模块框图

该框架主要包括 3 个主要模块，分别为预处理模块、运动估计模块、以及编码模块。虽然 3 个模块串行地对编码序列进行相应处理，但各模块之间有一定的帧数量延迟时间，因此各个模块就可以独立并行工作：当第 1 个模块进行了 M 帧数据的处理后，第 2 个模块便开始运行；当第 2 个模块对数据进行了 N 帧处理后，第 3 个模块就开始进行编码处理。

整个并行编码系统完整的结构框图如图 5 所示。相比传统的视频编码结构，新的框架改变了反馈回路，主要添加了预处理模块和并行整像素运动估计模块，在并行编码模块中，主要用亚像素运动估计代替原来的运动估计。图中 F_n^* 为进行并行预处理后的原始图像，$\mathbf{MV}'$ 为经过整像素运动估计后的运动矢量，$\mathbf{MV}$ 为经过亚像素运动估计后的最终运动矢量。

(1)并行预处理　图 5 中的并行预处理模块中首先对原始视频图像序列进行去噪滤波，平滑滤波以及 RGB 到 YUV 颜色空间转换等预处理，由于预处理在空间和时间上都基于原始帧进行，因此不存在相互之间的依赖性，可以并行进行处理。另外在这个模块中，最重要的工作是通过相关性参数的计算，来对序列进行场景检测和帧类型的判断，通过并行滤波以及并行场景检测和帧类型判断，快速地为后续运动估计模块和并行编码模块提供所需的相关数据信息。

(2)并行运动估计　由于在实际的视频编码系统中，除去视频数据的输入输出，存储设备的初始化及读写时间以及硬件平台的准备时间，单纯在编码模块中运动估计部分往往需要花费最大量的时间。而在运动估计中，整像素运动估计在多参考帧模式下占据最大的计算量和计算时间，因此，整像素的并行化运动估计是解决问题的关键。然而并行化运动估计需要满足下面条件：(1)参考帧之间没有相关性；(2)可以同时获得运动估计参考帧和被估计帧；(3)必须已经确定好被估计帧和参考帧之间的参考关系。

如 3.1 节中所述，由于传统编码框架中重建帧编码回路的设计，很难在传统的编码结构中进行并行化运动估计。针对上述问题，如图 5 所示，本文采用将原始图像序列作为参考帧的方法，将消耗最大编码时间的运动估计部分独立分开，进行并行化

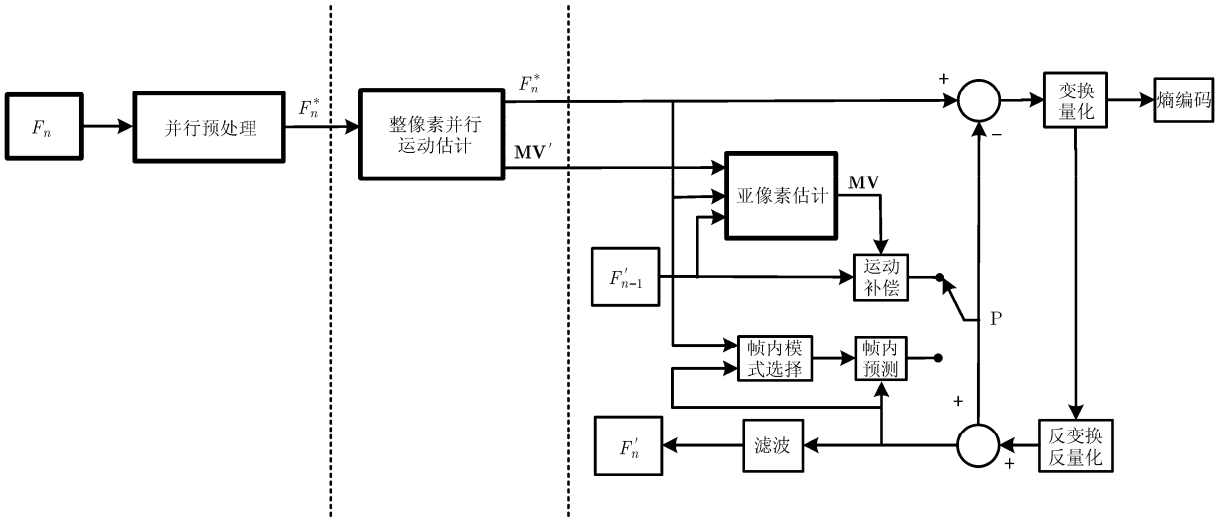


图 5 并行编码结构框图

运动估计。由于在并行预处理模块中原始帧数据也已经被缓存，且已经获得了所有图像的帧类型参数(I, P, B 帧类型)，这里只需要根据帧类型参数进行原始参考帧选择，因此满足并行化运动估计的所需条件，多帧可以同时并行进行整像素运动矢量搜索，从而大大提高了编码并行度。

将原始图像序列作为参考帧的方法首次在 MPEG-2 的开源编码器 TM5 中得到使用，旨在提高运动估计的准确度，因此这种方法能够保证运动估计的正确性。在文献[8]中提出了将编码和运动估计并行处理，也采用原始图像作为参考帧来解决运动估计对参考帧的依赖性，但是它只能保证编码反馈环内的运动估计的独立性，无法实现多帧间运动估计的并行性。

本文提出的编码框架，利用编码延迟能够缓冲帧的特点，同时充分利用预处理模块获得的帧类型参数，使运动估计模块对每帧数据独立地并行化运动估计，经过运动估计后的视频帧才能进入编码模块。这种并行运动估计极大地提高了整个编码框架的并行性，也极大地提高了并行硬件计算资源的利用率。并行运动估计参考结构如图 6 所示，假设输入序列为 IBBPBBP，箭头所指方向即为当前帧进行运动估计的可选参考帧，根据上面的分析，如果硬件计算资源足够，图 6 中的所有帧可以同时进行并行运动估计。

(3)并行帧内、帧间编码　通过之前两个模块已经获得了帧类型参数以及运动矢量，使得在编码模块中的计算量大大减小，这里采用隔行编码的方式，每个场作为一个片，各个场之间独立进行编码，并且采用 2.3 节中所示的帧间编码方式，如图 5 所示，

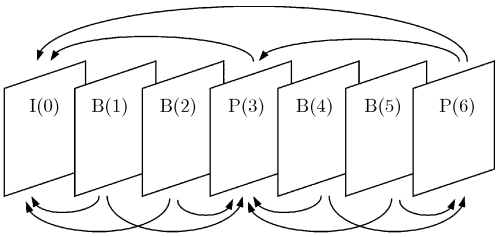


图 6 并行运动估计参考帧示意图

首先进行宏块模式选择，并将第 2 个模块中获得的整像素运动矢量 **MV′** 作为亚像素搜索的起始点，进行亚像素搜索得到最终的运动矢量 **MV**，然后经过运动补偿，对残差图像进行整数变换、量化以及熵编码来得到 AVS 码流。由于在这个模块中没有进行整像素运动估计，因此计算速度提高很多，也提高了帧间预测的效率，从而能够减小并行帧数，克服了 2.3 节中所述方法的缺点。

3.3 基于 AVS+实时编码的并行编码算法

AVS+视频编码标准不支持 WPP 和 Tiles 并行化技术，只有片编码可用于并行化计算。AVS+标准主要应用于未来高清及 3D 的网络视频和数字电视，适用于处理大分辨率下的单向视频快速编码，故而允许一定的延时。且 AVS+支持隔行编码。根据 AVS+标准的上述特点，我们将上述并行编码框架应用到 AVS+视频编码标准的实时编码器中，得到了基于 AVS+实时编码的并行视频编码算法。该算法适用于 3D 及高清的数字电视和网络电视，对于大分辨率单向视频编码有很大优势，而对于双向视频通信，若同样处理高清视频，考虑其缓存时延和信道时延，再加上编码延迟时间，难以满足实际双向视频通信的最低延迟要求。算法的具体步骤如下：

步骤 1　建立 L 个并行预处理线程。由于预处理是在原始图像上进行，并且算法复杂度不高，因此预处理线程之间没有相关性，线程数 L 一般不要超过 2，每个线程对应一帧原始图像。

步骤 2　建立 $2\times M$ 个整像素并行运动估计线程。由于经过步骤 1 后，每帧的编码帧类型已经确定，因此根据各自对应的原始图像参考帧进行并行运动估计。AVS+视频编码标准主要应用于隔行图像，在运动估计中，每帧图像分成 2 个场，每个场的运动估计独立作为一个线程工作。

步骤 3　建立 $2\times N$ 个编码线程。每个编码线程根据步骤 2 获得的整像素运动矢量进行亚像素搜索，这样有效降低了编码的复杂度。每帧间编码采用 2.3 节介绍的参考帧之间相互依赖的并行编码方式；考虑到 AVS+视频编码标准主要应用于隔行图像，因此每帧内分为 2 场，每场作为一个片，各个场之间独立进行编码，这样既提高了编码的并行度，也没有编码效率的损失和条带效应的出现。

步骤 4　并行编码线程的码率控制。由于步骤 2 进行运动估计之后，不仅获得了编码图像的运动矢量，而且也得到了编码图像的绝对差值和(SAD)，这样获得的 SAD 可以作为编码图像的复杂度参考，为编码图像的比特分配提供了依据。

步骤 5　编码延迟控制。从上面的编码步骤可以推导出编码器的编码延迟为 $L+M+N$ 帧的时间，其值越小，说明编码延迟越小。L 和 M 的选取主要取决于硬件资源条件，其值越大占用的硬件资源越多，对整个编码系统的并行度影响越大。而 N 的取值在占用硬件资源的同时也影响系统的编码并行度，一般来说，N 越大编码并行度越大，但 N 达到一定数量后由于重建帧之间的相关性导致并行度不变，但 N 越大将会给码率控制造成更大的误差。因此，L, M, N 的分配原则是，在硬件资源 CPU 核心足够的情况下 L 和 M 越大越好，而 N 的取值要综合考虑编码速度和码率控制误差，在不影响编码速度的前提下取较小值。

根据上面的算法步骤，以及编码器硬件资源的可得性，可以确定编码器延迟要求以及线程处理瓶颈，合理地分配 L, M, N。

4　实验结果与分析

为了验证所提出的并行编码框架性能，本文使用一台 Dell 服务器作为多核测试平台，CPU 为 Intel Xeon, 2×12 核，单核主频 3.06 GHz，测试序列采用广电总局广播科学院规划院的 1920×1080 高清序列：parterre, dial, basketball, volleyball, leaf, birdcage，以及一个将上述 6 个序列拼接而成的高清测试序列。编码顺序为 IPBB, GOP 是 30，搜索范围±32，测试软件为根据 AVS+视频编码标准[15]开发的 AVS+高清实时编码器，该编码器既支持传统的 AVS+视频编码方式(串行编码)，也支持本文提出的基于 AVS+的并行视频编码算法(并行编码)。实验中分配预处理线程 L 为 1，并行运动估计线程 M 为 4, N 为帧间并行帧数，其它参数名称的意义描述如表 1 所示。

表1 参数名称的意义描述

名称	意义描述
ME2	UMHexagonS 算法[16]
ME1	简化的 UMHexagonS 算法 (不包含 5×5 方形和大六边形搜索)
亚像素搜索 1	半像素搜索 2 次，1/4 像素搜索 4 次
亚像素搜索 2	半像素搜索 4 次，1/4 像素搜索 10 次
亚像素搜索 3	半像素搜索 6 次，1/4 像素搜索 12 次
帧间并行帧数	2.3 节(图 3)中参与并行编码的帧数

4.1 CPU 利用率对比

当计算机利用多核处理器进行并行运算时，CPU 利用率是体现算法效率的一个重要特性。若程序运行时能充分调动各个 CPU 核同时工作，必然会大大提高计算速率。

图 7(a)为采用串行编码对测试序列进行编码时的 CPU 利用率，仅有第 11 个线程在进行工作，说明串行编码无法充分利用 CPU 核。而图 7(b)中采用本文的算法进行并行编码，运动估计设置为 ME2，使用亚像素搜索 3，且设置帧间编码帧数为 16，可以看到每个 CPU 核都参与了计算，且利用率都较高。

4.2 加速比

加速比是衡量并行计算算法性能的一个重要指标，反映了并行优化后和优化前程序运算速度之间的关系。本文的编码算法中，加速比定义为并行算法优化后编码帧率和优化前的编码帧率的比：

$$\mathrm{SU}=\frac{\text{并行优化后的编码帧率}}{\text{并行优化前的编码帧率}} \tag{1}$$

(a)串行编码的CPU使用情况

(b)并行编码的CPU使用情况

图7 CPU利用率比较

式中 SU 为加速比，其值越大，算法的并行性越好，编码速度也越快。

表 2 所示为测试 1500 帧拼接高清序列在不同的运动估计参数 ME1 和 ME2 以及亚像素搜索 1, 2, 3 下实现的加速比，fps 表示编码帧率。可以看出，采用本文并行算法能够获得足够的平均加速比，完全满足实际需求。

表2 不同运动估计复杂度下的加速比

运动估计	亚像素搜索	并行编码(fps)	串行编码(fps)	平均加速比
ME1	亚像素搜索 1	51.4	4.2	12.24
ME2	亚像素搜索 1	43.7	3.7	11.81
ME1	亚像素搜索 2	50.4	4.2	12.00
ME2	亚像素搜索 2	42.4	3.6	11.78
ME1	亚像素搜索 3	50.5	4.2	12.02
ME2	亚像素搜索 3	41.3	3.5	11.80

4.3 帧间并行度

如上所述，并行帧间预测编码中的并行帧数是一个决定编码并行度的关键问题。若帧间并行编码帧数 N 太少，并行度就达不到预期的目标，从而影响编码速率；而并行帧数太大，会增大码率分配的误差。因此确定适当的帧间并行编码帧数是一个关键问题。

表 3 给出了在不同帧间并行帧数下进行并行编码得出的实验结果。设置亚像素搜索参数为 3。由于并行帧数越大，越有利于提高解码帧主观质量，但同时需要考虑帧率是否也能获得相应增加。由表 3 可见，帧间并行帧数为 8 时，编码速率已经完全满足实际编码器的要求，而此时的编码延迟为 13 帧 (L=1, M=4, N=8)。而帧间并行帧数越少，对编码器码率控制越有利。另外，由表 3 可见，当并行帧数从 4 开始累加后，帧率有大幅度提高，但当其达到 12 以后，帧率开始出现波动且没有继续增长，因此，我们可以在满足编码速度的前提下尽量减少帧间并行帧数，这为更好地实现码率控制提供了有利的条件。

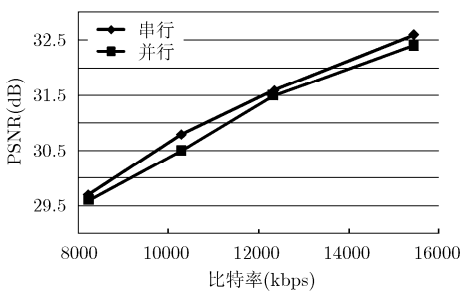
表3 帧间并行帧数对编码速率的影响

并行编码帧数	ME2(fps)	ME1(fps)
4	24.9	24.7
5	29.0	30.4
6	34.0	35.4
7	38.0	39.8
8	40.1	43.3
10	40.8	48.3
12	41.3	50.5
16	40.9	49.4

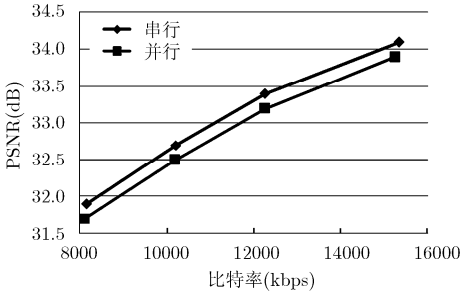
4.4 率失真比较

我们对 6 个高清序列进行测试后，绘制不同比特率下，在并行和串行条件下编码的率失真曲线图，如图 8 所示，两条不同黑白的曲线分别是串行编码和并行编码的结果。对于重建帧的质量主要通过 PSNR 在不同比特率下的率失真曲线来比较。通过计算得到 6 组序列的平均 PSNR 损失值为 0.14 dB，最小损失为 0 dB，最大损失在 0.23 dB 以内。

采用原始帧作为运动估计的参考帧虽然可以大大提高编码并行度，但对运动估计的准确性还有一定影响，因为在运动补偿的时候使用了解码回路的重建帧。对 6 个序列进行测试后，图 8 给出了 2 个 PSNR 损失最大序列的率失真曲线(我们需要考虑在最坏环境下的编码情况，其它 4 个序列效果较好)，两条曲线分别是在不同码率下串行编码和并行编码的结果。可以看出，即使在最差的条件下也仅有 0.2 dB 左右的损失，故采用原始帧作为运动估计参考图像的方式对此并行编码框架下的编码图像结果的影响极小，是切实可行的，因此在解码图像质量方面，本文提出的并行编码算法能和一般串行编码一样获



(a) volleyball序列的率失真曲线



(b) basketball序列的率失真曲线

图8 不同序列的率失真曲线

得比较好的效果，而并行编码在速率上的提升很明显，这也进一步验证了本文算法性能好且损耗低的优势。

5 结束语

本文改变了传统视频编码框架，利用编码器延迟的特点，设计了一种新的并行编码框架，将这种并行编码框架应用于 AVS+标准高清视频编码标准，提出了一种基于 AVS+实时视频编码的并行视频编码算法。实验结果表明，该算法能充分发挥多核处理器的运算能力，在编码性能损失极小，编码延迟可控(在 0.5 s 以上)的条件下，极大地提高了视频的编码速度。

参 考 文 献

[1] 张晓亮. AVS 地面数字电视产业化推广和应用[J]. 电视技术, 2012, 36(22): 3–4.
Zhang X L. The promotion and application of terrestrial digital TV industrialization[J]. *Video Engineering*, 2012, 36(22): 3–4.

[2] 董文辉, 邓向冬. AVS+视频压缩技术及应用[J]. 广播与电视技术, 2012, 12(s1): 41–43.
Dong W H and Deng X D. Techniques and applications of AVS+video compressing[J]. *Radio & TV Broadcast Engineering*, 2012, 12(s1): 41–43.

[3] Chen Da and Singh D. Fractal video compression in OpenCL: an evaluation of CPUs, GPUs, and FPGAs AS Acceleration platforms[C]. 18th Asia and South Pacific Design Automation Conference(ASP-DAC), Yokohama, 2013: 297–304.

[4] Ge S, Tian X, and Chen Y K. Efficient multithreading implementation of H.264 encoder on Intel hyper-threading architectures[C]. IEEE Proceedings of the 2003 Joint Conference of the Fourth International Conference, Singapore, 2003: 469–473.

[5] Tian X, Chen Y K, Girkar M, *et al*.. Exploring the use of hyper-threading technology for multimedia applications with Intel OpenMP compiler[C]. Proceedings of the Intel Parallel & Distributed Processing Symposium, Nice, 2003: 36–43.

[6] Chen Y K, Tian X, Ge S, *et al*.. Towards efficient multi-level threading of H.264 encoder on Intel hyper-threading architectures[C]. Proceedings of the 18th Intel Parallel and Distributed Processing Symposium, Santa Fe, 2004: 63–72.

[7] Ye Jian-hong and Liu Ji-lin. Fast parallel implementation of H.264/AVC transform exploiting SIMD instructions[C]. Intelligent Signal Processing and Communication Systems, Xi'an, 2007: 870–873.

[8] 张超. 基于 TILE64 的 H.264 多线程并行编码[D]. [硕士论文], 西安电子科技大学, 2011.
Zhang C. Multithreaded parallel programming of H.264 on TILE 64[D]. [Master dissertation], Xidian University, 2011.

[9] Alvarez-Mesa M, Chi C C, and Juurlink B. Parallel video decoding in the emerging HEVC standard[C]. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, 2012: 1545–1548.

[10] Chi Ching-chi, Alvarez-Mesa M, and Juurlink B. Parallel scalability and efficiency of HEVC parallelization approaches [J]. *IEEE Transactions on Circuit and System for Video Technology*, 2012, 22(12): 1827–1838.

[11] Fuldseth, A, Horowitz M, and Xu Shi-li. Tiles for managing computational complexity of video encoding and decoding[C]. Picture Coding Symposium (PCS), Krakow, 2012: 382–392.

[12] 许昌满, 李国平, 王国中. AVS 编码器 Slice 并行处理算法研究与实现[J]. 中国图象图形学报, 2009, 14(6): 1108–1113.
Xu C M, Li G P, and Wang G Z. Study and implementation of AVS encoding algorithm based on slice parallel[J]. *Journal of Image and Graphics*, 2009, 14(6): 1108–1113.

[13] 宁华, 梅铮, 李锦涛. 基于 slice 的 H.264 并行视频编码算法[J]. 计算机工程, 2005, 31(4): 181–182, 211.
Ning H, Mei Z, and Li J T. Slice-based parallel algorithm of H.264 video encoder[J]. *Computer Engineering*, 2005, 31(4): 181–182, 211.

[14] 于俊清, 李江, 魏海涛. 基于同构多核处理器的 H264 多粒度并行编码器[J]. 计算机学报, 2009, 32(6): 1100–1109.
Yu J Q, Li J, and Wei H T. Multi-Grain parallel H.264 encoder for homogeneous multi-core architecture[J]. *Chinese Journal of Computers*, 2009, 32(6): 1100–1109.

[15] AVS Working Group. GY/T257.1-2012, Radio and television -Advanced coding and decoding of audio and video-Part 1: Video[S]. 2012.

[16] Zhao W Q and Xu S. Research and optimization of UMHexagons algorithm based on H.264[C]. 2012 Fourth International Conference on Multimedia Information Networking and Security (MINES), Nanjing, 2012: 600–603.

蒋骁辰： 男，1988 年生，硕士，研究方向为视频编码技术和并行数据处理.

李国平： 男，1974 年生，高级工程师，研究方向为研究数字音视频编解码技术、复用技术、网络传输技术等.

王国中： 男，1962 年生，教授，研究方向为视频编解码与多媒体通信.