X265 的平均码率控制算法优化*

陈辰灿 滕国伟 王国中 赵海武 李国平
(上海大学通信与信息工程学院 上海 200072)

摘 要：码率控制算法通过动态的调整编码参数，使产生的码率能在有限的带宽内稳定传输，并尽可能提高视频的质量。目前针对 HEVC 的并行编码以及对应码率控制已成为研究热点，现有并行结构下的平均比特率控制算法受到帧间依赖性的约束，待编码帧无法及时获得与其并行帧的实际编码比特数进行码率控制，因此本文算法利用比特数预测模型通过预测并行帧的实际比特数作为码率控制的参数，及时的进行并行编码并且在此基础提出了缓冲区的优化过程。仿真结果表明，相比原有算法，减少的平均码率误差约为 4.43%，同时提高的平均 PSNR 约为 0.21 dB。

关键词：X265；并行；码率控制；ABR 算法
中图分类号：TN919.8 **文献标识码**：A **国家标准学科分类代码**：510.60

Average bit-rate algorithm optimization for rate control of X265

Chen Chencan Teng Guowei Wang Guozhong Zhao Haiwu Li Guoping
(School of Communication and Information Engineering, Shanghai University, Shanghai 200072, China)

Abstract: Rate control algorithm dynamically adjusts the encoding parameters to ensure stable transmission rate within the limited bandwidth, and maximize the quality of the video. The parallel encoding and rate control for HEVC has become advanced research focus. The average bit rate control algorithms under existing parallel structure is restrained by the inter-dependent, so the coding frame can't timely obtain the actual bits of frames which are in parallel with it. Therefore, in this paper, it uses the model which can predict frame's number of bits in parallel architecture to control bit-rate, and on this basis the optimization process is proposed of buffer. The results show that the method reduces the rate of error by 4.43%, and improves the average Peak Signal-to-Noise Ratio (PSNR) of 0.21 dB simultaneously.

Keywords: X265; parallel; rate control; ABR algorithm

1 引 言

随着人们生活水平的提高，对于高清、超高清视频的需求已经越来越普遍。因此为了满足高清、超高清视频的压缩编码，国际电信联盟(ITU-T)的视频编码专家组(video code expert group，VCEG)以及国际标准化组织(ISO)与国际电工委员会(IEC)的运动图像专家组(motion picture expert group，MPEG)共同组成的视频联合协作小组(joint collaborative team on video coding，JCT-VC)[1]在 2013 年发布了最新一代的视频编码标准(high efficiency video coding，HEVC)[2]。

HEVC 中引入了许多新的特性，例如残差编码结构、Tile、波前并行编码技术 WPP(wave front parallel processing)[3]等，使得编码效率增加了一倍，但同时复杂度成本也剧烈增加，因此实时编解码技术成为了研究热点，其中码率控制尤为重要[4-5]。在视频应用环境中，网络的异构性、传输带宽的波动会导致视频质量的不稳定[6]，因此为了使产生的码率能在有限的带宽内稳定传输，并尽可能提高视频的质量，就需要进行码率控制。

简单地将传统的码率控制方法应用到 HEVC 编码器中因难以满足并行处理框架特点而导致其精确度明显下降，因此研究并行的码率控制算法具有重要应用价值[7]。文章主要内容包括：在第 2 节主要分析现有并行算法的优缺点；第 3 节提出改进算法；第 4 节给出实验结果。

2 并行架构下码率控制算法分析

2.1 X265 的并行编码架构

X265 采用了波前并行技术和具有参考帧依赖性的帧间并行技术[8]相互配合的混合编码架构。如图 1 所示，为了提高并行处理能力，在帧内进行编码时运用了波前并行

收稿日期：2016-01
* 基金项目：上海市自然基金资助(14ZR1415200)、国家自然基金资助(61271212)；上海市科技创新行动计划(14511105602)资助项目

技术,同时帧与帧之间的并行方式又采用了具有参考帧依赖性的帧间并行技术,这两种并行技术同时使用提高了编码器的实时性。

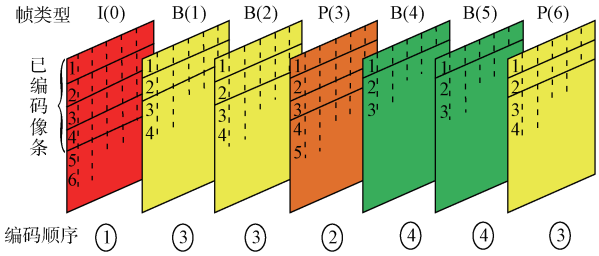

图 1 X265 混合并行编码框架

2.2 并行架构下的 ABR 码率控制算法分析

平均码率控制算法(ABR 算法[9-10])中当前帧需要获得前一帧编码产生的实际比特数和到前一帧为止的所有帧的实际编码比特数和作为码率控制的参数来确定 QP。在串行架构下,当前帧在前一帧完成编码后才开始编码,所以能够获得确定 QP 的这两个码率控制参数。而在并行架构下,当前帧开始编码时前面还有未完成编码的帧,因此当前帧无法及时获得前一帧编码产生的实际比特数和到前一帧为止的所有帧的实际编码比特数和。

为了解决上述 ABR 算法在并行架构下存在的问题,X265 将码率控制参数的一次更新更改为两次更新,并且规定当前帧只有在前一帧完成第一次码率控制参数更新后才能开始编码。在原来的串行架构下,码率控制参数在每一帧完成编码后进行更新,而在当前的并行架构下,运用参数两次更新的方法来解决参数更新不及时的问题:

1)第 1 次码率控制参数更新:考虑到帧与帧之间进行参考的依赖关系,第 1 次的参数更新发生在当前帧的一半 CTU 行完成编码后,这样即可以利用已编码 CTU 行产生的信息来进行下 1 帧的码率控制,同时可以获得下一帧第 1 个 CTU 行编码所需的参考信息。更新后的参数根据当前帧的编号与线程数的关系会有两种不同的用途,当有 m 个线程并行编码时,第 1 帧~第(m-1)帧的第一次参数更新后的参数将直接用于下一帧 QP 的确定,而编号大于(m-1)的帧的第一次参数更新后的参数将用于与其并行的第 1 帧的第 2 次码率控制参数的更新。例如第 1 帧的第一次参数更新后的参数直接用于第 2 帧 QP 的确定,而第 m 帧的第 1 次参数更新后的参数将用于第 1 帧的第 2 次参数更新,第(m+1)帧的第一次参数更新后的参数将用于第 2 帧的第 2 次参数更新,依次类推;

2)第 2 次码率控制参数更新:当前帧完成编码时进行第 2 次的更新,但是第 2 次更新则需要等待与当前帧并行处理的最后一帧完成它的第一次参数更新,例如第 1 帧的第二次参数更新需要等待第 m 帧完成它的第 1 次参数更新,第二次更新得到的码率控制参数将用于第(m+1)帧 QP 的计算。

假设 X265 并行处理 2 帧图片,如图 2 所示,当第 1 帧的一半(红色部分)CTU 行完成编码后当前帧进行第 1 次码率控制参数的更新,更新完成后第二帧才能开始编码并且使用第 1 帧更新的参数来确定 QP(如图中①所示)。而当第一帧完成编码后(橙色部分),将等待第二帧的第 1 次参数更新完成,当第 2 帧在一半(黄色部分)CTU 行完成编码后进行第一次参数更新,并且更新后的参数用于第一帧的第 2 次参数更新(如图中②所示),当第 1 帧的第二次码率控制参数更新后将用于第 3 帧 QP 的确定(如图中③所示),可见码控参数两次更新的方法牺牲了一定的并行度来提高码率控制的精确度。

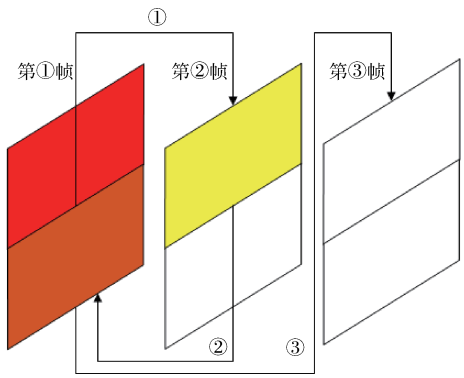

图 2 并行架构码率控制过程

经过上述分析,可知 X265 采用了波前并行和具有参考帧依赖性的帧间并行混合编码架构,同时采用了码率控制参数两次更新的方法来提高 ABR 码控算法在并行架构下的准确度,但是此方法仍然存在着两个问题:1)在参数更新时,仅仅用当前帧已编码部分的比特数来替代帧完成编码后的实际比特数来确定下一帧的 QP,此时 QP 值并不准确;2)因为无法得到到前一帧为止所有帧的实际编码比特数和,所以也无法得到前面所有帧的实际偏差,而仅仅利用已完成编码帧和未完成编码的已编码部分的实际偏差来调整 QP,因此用此 QP 编码后的实际偏差只能抵消帧的已编码部分的偏差,而不能抵消帧的尚未完成编码部分的偏差,从而导致实际码率偏离目标码率。

3 并行架构下的 ABR 算法优化

3.1 比特数预测模型

针对参数两次更新方法存在的不足,本文在此基础上提出了相应的优化方法:当前帧根据已编码的 CTU 行产生的实际编码比特数来预测当前帧剩下未编码 CTU 行的编码比特数,可以得到当前帧的估计编码比特数,利用已完成编码帧的实际比特数和未完成编码帧的估计编码比特数可以得到到当前帧为止的所有帧的编码比特数和,从而使待编码的下一帧能够类似在单线程下获得前一帧的编码比特数和前面所有帧的实际编码偏差来调整 QP,下一帧图像用此 QP 编码后的实际偏差不仅能抵消已完成编码帧的偏差,而且能抵消尚未完成编码帧的偏差,最终使平均码率偏

差趋近于 0，达到实际码率接近目标码率的目的。该方法分两步：1)是利用比特数预测模型得到当前帧的未完成编码部分的预测比特数；2)是计算当前帧的估计编码比特数和到当前帧为止的所有帧的实际编码偏差。

图 3 是码率控制算法在并行架构下改进后的参考图，图中第 1 帧，在进行第 1 次参数更新前会预测当前帧的未编码部分(格子部分)的编码比特数，然后使用帧的估计比特数进行参数更新用于第 2 帧 QP 的确定(如图中①所示)，同样当第 2 帧在一半 CTU 行完成编码后预测未编码部分(格子部分)的实际比特数来进行第一次的参数更新，而第 1 帧在第 2 帧完成第一次参数更新后进行第 2 次的参数更新(如图中②所示)用于第 3 帧 QP 的确定(如图中③所示)。图 3 与图 2 相比，增加了对未编码部分的比特数预测用于码率控制参数的更新。

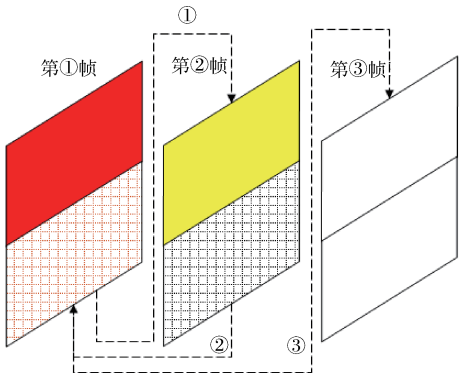

图 3　改进后的并行架构码率控制过程

第 1 步利用比特数预测模型得到当前帧未完成编码部分的预测比特数：文献[7]中根据 X264 码率控制策略提出了一种动态预测比特数的模型，我们在此模型基础上进行了修改并且提出了适合参数两次更新方法的预测模型。预测方法如式(1)所示：

$$pre\_bits = \frac{coeff\_blur \times remain\_SAD}{init\_QP} \tag{1}$$

式中：pre_bits 是当前帧未完成编码部分的预测比特数，$remain\_SAD$ 是当前帧所有未完成编码 CTU 行的 SAD 值，$init\_QP$ 是当前帧的码率控制确定的初始 QP 值。$coeff\_blur$ 是模糊复杂度系数，其计算公式如式(2)所示：

$$coeff\_blur = \frac{coeff\_sum}{count} \tag{2}$$

式中：$count$ 代表加权累加更新的次数，计算公式如式(3)所示；$coeff\_sum$ 代表了累计复杂度系数，其计算公式如式(4)所示。

$$count'' = count' \times damping + 1 \tag{3}$$

$$coeff\_sum'' = coeff\_sum' \times damping + coeff \tag{4}$$

式(3)、(4)中：$damping$ 是衰减系数，$count''$ 和 $coeff\_sum''$ 表示更新后的值，$count'$ 和 $coeff\_sum'$ 表示更新前的值。$coeff$ 的值如式(5)所示。

$$coeff = \frac{pctu\_bits \times QP'}{SAD'} \tag{5}$$

式中：SAD' 和 QP' 分别表示前一 CTU 行的 SAD 值与平均 QP 值，$pctu\_bits$ 表示前一行实际的编码比特数。通过式(3)、(4)可以看到，模糊复杂度系数的计算引入迭代关系，这样可以保证前后 CTU 行的实际编码比特数的平稳性。

第 2 步是计算当前帧的估计编码比特数和到当前帧为止所有帧的编码偏差：使用上面的预测模型可以得到当前帧未完成编码的 CTU 行的预测编码比特数 pre_bits，利用 pre_bits 可以计算当前帧的估计编码比特数 est_bits，如式(6)所示，其中 ctu_bits 是所有已编码 CTU 行产生的实际比特数：

$$est\_bits(i) = ctu\_bits(i) + pre\_bits(i) \tag{6}$$

因此在第一次参数更新时我们能够用当前帧的估计编码比特数来计算到当前帧为止的所有帧的编码比特数和，如式(7)所示。

$$total\_bits(i) = \sum_{n=1}^{i-m} act\_bits(n) + \sum_{n=i-m+1}^{i} [ctu\_bits(n) + pre\_bits(n)] \tag{7}$$

在式(7)中到当前帧为止的所有帧的编码比特数和由两部分组成，一是已经完成编码的帧产生的实际编码比特数，二是还未完成编码的帧的估计编码比特数。如果此时帧的编号 i 不大于线程数 m 时，则当前帧之前还没有帧完成编码，因此到当前帧为止的编码比特数和只有未完成编码帧的估计编码比特数部分，此时式(7)将简化为式(8)为：

$$total\_bits(i) = \sum_{n=1}^{i} [ctu\_bits(n) + pre\_bits(n)] \tag{8}$$

在参数的两次参数方法中，当 i 大于$(m-1)$时，第 i 帧的第 1 次参数更新后的参数将用于与其并行的第 1 帧(即第$(i-m+1)$帧)的第二次码率控制参数的更新，则在第$(i-m+1)$帧的第 2 次码率控制参数更新时，第$(i-m+1)$帧已经完成编码，所以到第 i 帧为止的所有帧的比特数和需要再次进行更新，将使用第$(i-m+1)$帧的实际编码比特数来替代原来的估计编码比特数，如式(9)所示。

$$total\_bits(i)'' = total\_bits(i)' + act\_bits(i-m+1) - [ctu\_bits(i-m+1) + pre\_bits(i-m+1)] \tag{9}$$

将式(9)化简后得到式(10)：

$$total\_bits(i) = \sum_{n=1}^{i-m+1} act\_bits(n) + \sum_{n=i-m}^{i} [ctu\_bits(n) + pre\_bits(n)] \tag{10}$$

式(10)代入 ABR 算法中计算 overflow 的算式中将得到式(11)。

$$overflow(i) = 1.0 + \frac{\sum_{n=1}^{i-m} act\_bits(n) + \sum_{n=i-m+1}^{i-1} [ctu\_bits(n) + pre\_bits(n)] - wanted\_bits(i-1)}{abr\_buffer(i)} \tag{11}$$

因为到当前帧为止的所有帧的编码比特数和由两部分组成,所以从式(11)中可知,前面所有帧的编码比特数偏差也由两部分组成:已完成编码帧的实际偏差和未完成编码帧的偏差;而未完成编码帧的偏差又由两部分组成,即已完成编码的 CTU 行的实际偏差和未完成编码的 CTU 行的估计偏差。因此可知计算第(m+1)帧的 QP 原先用第 1 帧与未完成编码的第 2 帧~第 m 帧已编码 CTU 行的实际偏差,现在需要加上第 2 帧~第 m 帧未编码 CTU 行的估计偏差;第(m + 2)帧的 QP 原先是用第 1 帧、第 2 帧和未完成编码的第 3 帧~第(m+1)帧已编码 CTU 行的实际偏差计算,现在需要加上第 3 帧~第(m + 1)帧的估计偏差,以此类推,因此可以得出:计算当前帧 QP 时,需要加上前面未完成编码的共(m−1)帧的估计偏差。

相对于原来的参数两次更新方法,从式(11)中可知本方法多出了当前帧前面未完成编码帧的未完成编码部分的估计偏差。经过这部分估计偏差修正的 overflow 能够使 QP 在进行两次调整后获得更加准确的 QP 值,使最终的实际码率接近目标码率。

3.2 缓冲区更新优化

开始码率控制之前当前帧需要更新缓冲区的状态,在 ABR 码率控制算法中对缓冲区的管理十分宽松,它的大小仅仅通过经验来控制,并且只与初始值、当前帧数和编码帧率有关,如式(12)所示:

$$abr\_buffer = \begin{cases} abr\_buffer\_init, & \left|\frac{i\_frame}{i\_fps\_num}\right| \leqslant 2 \\ abr\_buffer\_init \times \sqrt{\left|\frac{i\_frame}{i\_fps\_num}\right|}, & \left|\frac{i\_frame}{i\_fps\_num}\right| \geqslant 2 \end{cases} \tag{12}$$

式中:$abr_buffer(i)$为当前帧的平均缓存区大小,i_fps_num 为帧率,i_frame 为当前编码的帧数,abr_buffer_int 为初始的缓冲区大小,当平均目标比特数确定后 abr_buffer_int 值是一个常量。因此从式(12)中可以得知 abr_buffer只会随着帧数的增加而无限增加,因此缓冲区的大小没有设置上限,属于开放型的缓冲区。

针对上述缓冲区控制太过宽泛的缺点,本文提出了一种并行架构下的缓冲区更新优化方法。在 ABR 码率控制算法中,一帧数据从缓冲区传出的恒定比特数 $tran\_bits$ 是相同的:

$$tran\_bits = \frac{avr\_bitrate}{i\_fps\_num} \tag{13}$$

式中:$avr\_bitrate$ 和 i_fps_num 分别是设置的平均码率值和帧率。而根据比特数预测模型可以在第一次参数更新时获得当前帧的估计编码比特数,利用这个估计编码比特数与目标比特数可以得到每一帧的估计编码偏差 est_diff:

$$est\_diff(i) = tran\_bits(i) - est\_bits(i) \tag{14}$$

而在第二次参数更新时,当前帧已经完成编码,所以帧的实际编码比特数与目标比特数可以得到每一帧的实际编码偏差 act_diff:

$$act\_diff(i) = tran\_bits(i) - act\_bits(i) \tag{15}$$

因此当前帧在码率控制开始时能够使用前面所有帧的实际偏差与估计偏差更新缓冲区状态,如式(16)所示。

$$abr\_buffer(i) = abr\_buffer\_init + \sum_{n=1}^{i-m} act\_diff(n) + \sum_{i-m+1}^{i-1} est\_diff(n) \tag{16}$$

因此到当前帧为止的编码偏差小于 0 时,表示当前的实际编码比特数超过了目标比特数,则下一帧的平均缓冲区 abr_buffer 应该减小来限制产生过大的比特数,所以应使 $overflow$ 尽量变大,从而增大下一帧的 QP,使得下一帧编码产生的实际比特小一些。当到当前帧为止的编码偏差大于 0 时,情况与之相反。

3.3 码率控制优化算法流程

对 ABR 码率控制算法的优化主要发生在 3 个阶段中:1)码率控制开始阶段;2)第一次码率控制参数更新阶段;3)第二次码率控制参数更新阶段。优化后的码率控制算法具体步骤如下:

1)在码率控制开始前根据式(16)更新缓冲区的状态,然后按照原来的 ABR 算法的步骤使用前面更新的码率控制参数得到当前帧的初始 QP 值;

2)使用初始的 QP 进行编码,每编码一个 CTU 行之后,按照式(3)~(5)进行 count 与 coeff_sum 更新,并且在第一次码率控制参数更新后停止 count 与 coeff_sum 的更新;

3)在第一次参数更新时根据 count 与 coeff_sum 按式(2)得到模糊复杂度系数 coeff_blur,再由 coeff_blur、remain_SAD 和 QP 按式(1)计算当前帧未完成编码部分的预测比特数 pre_bits;然后利用 pre_bits 对另外 3 个重要参数进行了更新:①根据 pre_bits 按照式(6)计算当前帧的估计编码比特数;②由当前帧的估计编码比特数根据式(7)计算到当前帧为止的所有帧的编码比特数和;③由当前帧的估计编码比特数根据式(14)计算每一帧的编码估计偏差,这 3 个参数将用于下一帧码率控制 QP 的确定。

4)第二次参数更新时对两个参数进行了更新:①当前帧完成编码后使用编码产生的实际编码比特数替换它的编码估计比特数来更新当前已编码的所有帧的比特数和,例如当前帧为第 i 帧,此时更新的是到第(i+m−1)帧为止的所有帧的比特数和,并将用于第(i+m)帧的码率控制。②对当前帧实际编码比特数与目标编码比特数的偏差,如式(15)所示。

4 实验结果与分析

为了验证 ABR 码率控制优化算法性能,本文使用了 1 台 Dell 服务器作为多核测试平台,CPU 为 8 个 Inter Core

i3-2120 核，每个核的主频为 3.30 GHz，测试序列从 Class A，B，C，D，E 中各取出 1 个序列：Traffic、Cactus、BasketballDrill、BQSquare、FourPeople，编码顺序为 IPBB，测试软件为根据 HEVC 视频编码标准开发的 X265 实时编码器并且运行在 Windows 平台上。其他编码参数使用了 X265 编码器的默认设置，其中除 Traffic 序列总帧数为 150 帧，编码帧数设为 150 帧，其它序列编码帧数为 250 帧。为了考察优化算法对码率控制的效果，主要对比特数预测模型和码率控制的准确度进行了测试，在下面实验中原码率控制算法是 X265 的码率控制参数两次更新方法，优化后的方法是本文提出的码率控制方法。

4.1　比特数预测模型的准确度

实验的 5 个测试序列分别在指定的目标码率下进行编码，为了直观地分析每个序列在 ABR 码率控制下使用比特数预测模型的预测准确情况，绘制了每一帧在第一次参数更新后未编码部分产生的实际比特数和使用比特数预测模型预测的比特数对比曲线图，如图 4～图 8 所示。

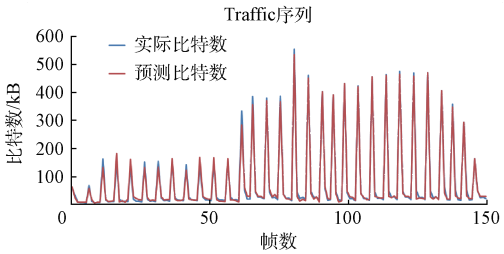

图 4　Traffic 序列实际比特数和预测比特数曲线

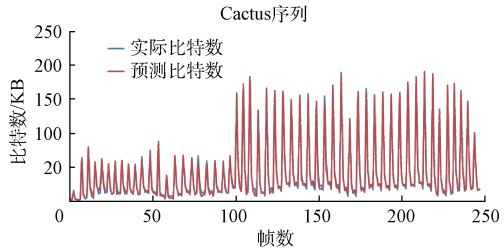

图 5　Cactus 序列实际比特数和预测比特数曲线

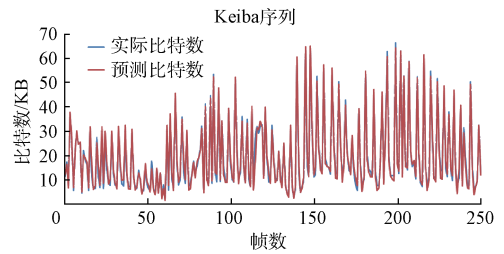

图 6　Keiba 序列实际比特数和预测比特数曲线

由上述测试序列的每一帧的未编码 CTU 行的预测比特数与实际编码比特数对比图可以看出，虽然预测比特数与实际编码比特数没有完全的重合，但是基本接近，所以在能够接受的误差范围内可以使用这个比特数预测模型预测的比特数来作为实际编码比特数进行下一帧的码率控制。

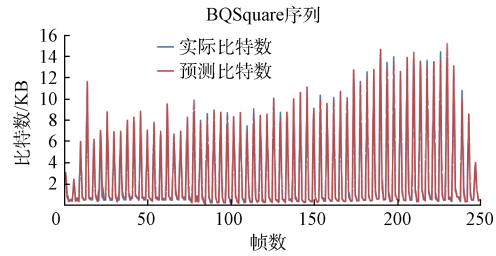

图 7　BQSquare 序列实际比特数和预测比特数曲线

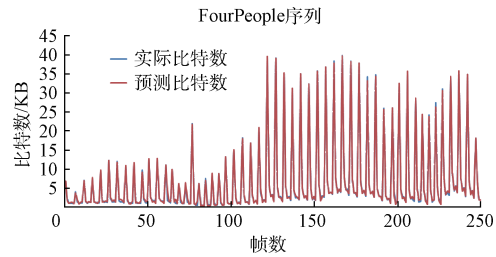

图 8　FourPeople 序列实际比特数和预测比特数曲线

4.2　码率控制准确度对比

实验选取的 5 个序列采用 HEVC Call for Proposal (CAP)(ITU-T，et al.，2010)中规定的码率作为目标码率对本文提出的 ABR 算法的优化算法进行验证。实验结果如表 1 所示，显示的是实际输出码率分别在预测优化算法作用下与原始 ABR 算法作用下的结果。

表 1　码率控制效果对比

序列	目标码率 A(Kbit/s)	原始 ABR 算法		优化算法	
		实际码率 B(Kbit/s)	码率误差/(%)	实际码率 C(Kbit/s)	码率误差/(%)
Traffic	14 000	13 212.25	5.63	1 3659.54	2.43
	8 000	7 426.95	7.16	7 834.27	2.07
	3 500	3 155.02	9.86	3 416.56	2.38
	2 500	2 221.23	11.15	2 446.94	2.12
Cactus	10 000	9 595.1	4.05	9 917.77	0.82
	7 000	6 637.44	5.18	6 968.92	0.44
	3 000	2 799.96	6.67	2 985.23	0.49
	2 000	1 861.31	6.93	1 987.46	0.63
Keiba	2 000	1 871.49	6.43	1 985.86	0.71
	1 200	1 136.63	5.28	1 191.11	0.74
	512	485.03	5.27	508.44	0.70
	384	365.15	4.91	381.24	0.72
BQ Square	1 500	1 464.76	2.35	1 476.81	1.55
	850	822.62	3.22	830.1	2.34
	384	366.06	4.67	375.14	2.31
	256	241.79	5.55	245.8	3.98
Four People	4 000	3919.28	2.02	3 970.5	0.74
	2 000	1913.58	4.32	1 990.59	0.47
	1 000	927.85	7.22	995.67	0.43
	512	473.27	7.56	509.17	0.55

表 1 的结果表明 5 个序列在原来 ABR 算法下的平均码率误差分别约为 8.45%、5.7%、5.47%、3.94%、5.28%,而采用预测比特数模型的优化码率控制算法的平均码率误差分别约为 2.25%、0.6%、0.72%、2.54%、0.55%,平均码率误差分别减小了约为 6.2%、5.1%、4.75%、1.4%、4.73%,可见优化算法减小了码率控制误差,提高了准确度。

4.3 PSNR 对比

从表 1 中可知本文优化算法提高了码率控制的精确度,但是同时需要考察本文优化算法对 PSNR 的影响,为了更直观的分析 PSNR 的情况,绘制了在原始算法和优化算法下的失真对比曲线图,如图 9~图 13 所示。

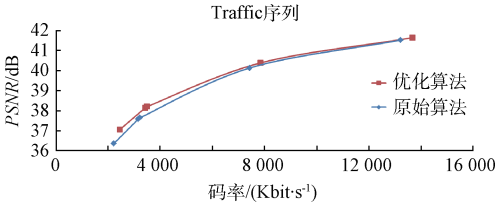

图 9 Traffic 序列原始码控算法和优化码控算法下的平均 PSNR 曲线

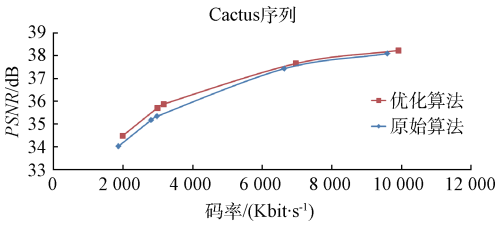

图 10 Cactus 序列原始码控算法和优化码控算法下的平均 PSNR 曲线

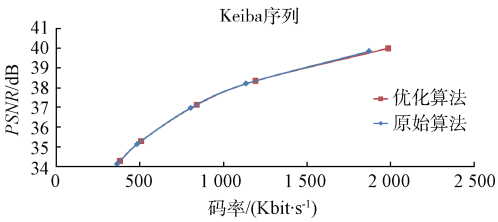

图 11 Keiba 序列原始码控算法和优化码控算法下的平均 PSNR 曲线

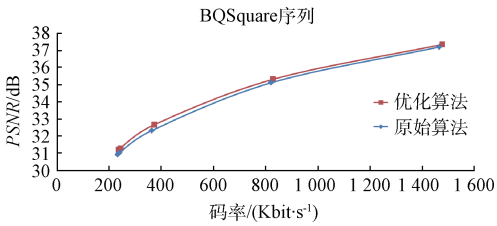

图 12 BQSquare 序列原始码控算法和优化码控算法下的平均 PSNR 曲线

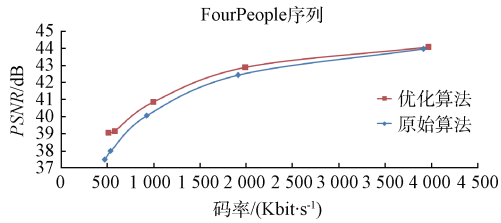

图 13 FourPeople 序列原始码控算法和优化码控算法下的平均 PSNR 曲线

由上述测试序列的平均 PSNR 对比图可以看出,除了 Keiba 序列之外其他序列在本文提出的优化算法下的平均 PSNR 相对于原始码率控制算法下的平均 PSNR 略有提高,而 Keiba 序列的平均 PSNR 也未下降太多,属于可以接受的范围。

5 结 论

为了解决 X265 并行架构下 ABR 码率控制会碰到的问题,X265 采用了码率控制两次参数更新的方法,但是两次更新参数的方法仍旧存在缺点,针对这些缺点本文提出了比特数预测模型。利用比特数预测模型可以在并行架构下得到前一帧的编码比特数、到前一帧为止的所有帧的编码比特数和缓冲区情况来优化码率控制。通过对比特数预测模型准确度、码率控制效果和 PSNR 的测试可以发现本文提出的比特数预测模型与缓冲区优化方法使得输出码率更接近目标码率,同时略微提高了编码客观质量。

参考文献

[1] POURAZAD M T, DOUTRE C, AZIMI M, et al. HEVC: the new gold standard for video compression: how does HEVC compare with H.264/AVC[J]. IEEE Consumer Electronics Society, 2012, 1(3): 36-46.

[2] HAN G J, OHM J R, HAN W J, et al. Overview of the high efficiency video coding (HEVC) standard[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1649-1668.

[3] 朱秀昌, 李欣, 陈杰. 新一代视频编码标准—HEVC[J]. 南京邮电大学学报, 2013, 33(3): 1-11.

[4] 周怡然, 陈耀武. 一种基于编码域的自适应灵活宏块次序方法[J]. 仪器仪表学报, 2011, 32(9): 2017-2131.

[5] WANG S. Rate-GOP based rate control for high efficiency video coding[J]. Selected Topics in Signal Processing, 2013, 7(6): 1101-1111.

[6] 卢鑫, 林茂六, 金雪松, 等. 新一代可伸缩视频编码标准:背景、特征、技术及其应用[J]. 电子测量与仪器学报, 2015, 29(10): 1415-1423. (下转第 78 页)