计算能力可伸缩的运动估计率失真优化

陆寄远[1],朝红阳[2],黄承慧[1],侯 ▮[1]
(1.广东金融学院计算机科学与技术系,广东广州 510521;2.中山大学软件学院,广东广州 510275)

摘 要: 不同硬件设备具有不同的计算能力,能否在任意给定计算能力约束下达到最好的编码效率,是当前视频编码研究领域的一个极具挑战性问题.同时,随着分块结构越来越灵活的编码标准不断出现(如:HEVC,H.264 等),运动估计不得不反复地应用在大小不同的各种分块上,导致其对编码总体计算复杂度的影响愈加重要.在此背景下,本文提出了一种针对运动估计的计算能力可伸缩(Complexity scalable)优化算法.我们通过对运动估计过程中预测失真度和计算复杂度的变化规律建模,发现根据各宏块的特性设置不同的预测失真度阈值可以优化地分配计算资源.而该阈值的大小则恰恰是各宏块的最小预测失真度加上一个由复杂度约束统一决定的偏移量.有鉴于此,我们进一步构造了计算能力可伸缩的优化运动估计算法,在不增加额外计算量的前提下,快速地得到各个宏块所对应的优化阈值,并完成运动估计.通过实验分析,该算法不仅具备自动适应不同计算复杂度约束的能力,而且在任意给定的复杂度约束下,都能提供优化的编码性能.

关键词: 视频编码; 运动估计; 计算能力可伸缩
中图分类号: TP391.41 **文献标识码:** A **文章编号:** 0372-2112 (2014)08-1495-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.08.006

Rate Distortion Optimization of Complexity Scalable Motion Estimation

LU Ji-yuan[1], CHAO Hong-yang[2], HUANG Cheng-hui[1], HOU Fang[1]
(1. *Department of Computer Science and Technology, Guangdong University of Finance, Guangzhou, Guangdong* 510521, *China*;
2. *School of Software, Sun Yat-sen University, Guangzhou, Guangdong* 510275, *China*)

Abstract: One of the major challenges of video coding is how to attain the best coding performance under different constraints of computational complexity with various hardware. The computational scalability of coding algorithms is particularly important on this occasion. On the other hand, more and more complicated coding structures are employed to repeatedly applied motion estimation on each block. And the algorithms of motion estimation exerts significant influence on the overall performance of video coding. Therefore, we proposed an optimized motion estimation algorithm with scalable complexity. This algorithm can automatically adapt to different video contents and optimally allocate the computing resources by setting appropriate distortion thresholds. These thresholds are the minimal predicted distortion of each macroblocks plus a unified offset. And all of these thresholds can be computed easily with negligible costs. According to our experiments, our proposed algorithm not only provides scalable complexity, but also achieves better coding performance at the same computational expenses.

Key words: video coding; motion estimation; scalable complexity

1 引言

视频编码的率失真性能是衡量视频压缩率和质量的综合指标,而计算复杂度则是评价视频编码计算消耗的重要指标.在编码器设计中,硬件的成本决定了设备的计算能力.如何在相同的硬件成本(即给定的计算能力)条件下,得到最优的视频编码效率(率失真性能),是目前视频编码硬件生产厂家的竞争要点,也是视频编码领域里的挑战性问题[1].这一挑战背后的科学问题就是计算复杂度约束下的视频编码率失真优化问题.至今为止,要整体解决这个问题还有相当大的难度,本文将针对视频编码中占计算资源较多的运动估计部分讨论该问题的一个解决方案.

目前大多数对于运动估计的研究成果都集中在构建快速算法[2].这些算法的出现大大地提高了运动估计的速度,令运动估计新技术的实用成为可能.然而,快速算法的主要目的在于加快编码速度,并没有考虑如何在计算能力相差悬殊的各种平台上优化编码性能的问题.

收稿日期:2013-07-08;修回日期:2013-09-18;责任编辑:马兰英
基金项目:国家自然科学基金(No.61173081);广东省自然科学基金(No.S2011020001215,No.S2012040007847)

直接把这些快速算法应用在不同的计算平台上,或者由于简化不足而超出了设备的计算能力,或者因为过于简化而浪费了系统的计算资源.尽管这些算法可以通过设置提前终止条件、搜索范围等控制参数调整其计算复杂度和运算时间,但是这些参数仅在给定经验值或推荐值附近有好的效果,除此以外,编码性能会急剧地下降.因此,真正实用的方法是为运动估计过程建立率失真和计算复杂度模型,并从一个整体角度优化.

由于率失真和计算复杂度的关系比较复杂,所以这方面的研究不多.Kannangara 等人[3]利用视频编码中判定 SKIP 模式的阈值调整运动估计的计算复杂度.该方法欠缺了率失真优化方面的考虑,在控制算法复杂度的同时,会造成编码性能的损失.Vanam[4] 等人使用机器训练的方法对运动估计的参数进行控制.他们提出了机器学习的方法优化复杂度约束下的率失真性能.这两种方法采用离线的方式训练参数,其准确率依赖于训练所使用的样本,难以动态适应不同视频的内容.FADTS 算法[5]在运动估计过程中采用了不同的搜索策略调整运动估计的复杂度.这些算法都寻求在不同计算约束下的优化编码方案.但要达到率失真性能的最优化,前提是对编码算法的率失真性能和计算复杂度有准确的预测.如能建立起两者的关系模型,那么就可以进一步地提高复杂度受限情况下的率失真性能.

为了提高计算能力受约束下编码算法的率失真性能,中科院和微软亚研院的 Su 等人利用虚拟计算资源缓冲(Virtual Computational Resource Buffer)为不同宏块分配计算资源,并构建了不同的运动估计和模式决策算法[6].此外, Liang 等人[7]通过建立码率-复杂度-失真度模型,提出针对视频编码优化的框架.上述的这些文章不仅给出了一个针对视频编码全过程的优化框架,而且算法的计算也十分简单.他们的主要思想是当系统的计算资源不足时,只使用最核心且效率最高的算法,而当资源充足时,逐层嵌套地使用更为复杂的计算模块.但是整体的优化依赖于每一部分的优化结果,上述研究并没有专门针对运动估计建立模型,当视频内容变化较大的时候,这些算法会因为不能自动适应,而引起部分运动估计的率失真损失.

要在计算受限的情况下,获得最好的率失真性能,同时避免复杂的额外计算,最根本的问题就是要预测各种运动估计算法率失真收益和计算代价之间的关系.本文主要的研究成果之一就是建立了一个指数模型描述不同宏块运动估计预测失真度与所消耗计算量之间的关系.该成果弥补了当前方法在准确预测运动估计过程中率失真和计算复杂度关系上的不足.除此之外,通过对此模型的优化,我们还发现只要为每个宏块准确地设置不同的预测失真度阈值,就可以十分简单地分配计算资源并令总体预测失真度达到最优.有鉴于此,为了准确设置该阈值,本文给出了一种自适应的算法.该算法非常简捷地解决了优化分配运动估计计算资源的问题.与此同时,我们另一个针对分数运动估计和分数插值协同优化的相关成果[8]已经发表在业内权威杂志 IEEE Transaction on Circuit and System for Video Technology.该成果与本文配合使用,将有效地解决视频编码中帧间预测的率失真优化问题.

2 预测失真度-计算复杂度模型

为了行文方便,我们引入了一些记号.如图 1 所示,i,j 表示当前运动估计宏块所在的位置,即第 i 行,第 j 列的宏块.D 是指编码帧中所有宏块运动估计后的总体预测失真度,该预测失真度的大小决定了编码该图像需要的码率及图像帧重建的质量.C 是指运动估计总共所需要消耗的计算复杂度,$C_{\text{constraint}}$是运动估计的复杂度约束.为了简化该问题的讨论,本文统一用搜索点数作为计算复杂度的量度.另外,本文用大写的 D 和 C 表示整帧图像的预测失真度和计算复杂度,而用小写的 $d_{i,j}$和 $c_{i,j}$表示不同宏块运动估计的预测失真度和计算复杂度.图 1 表示宏块(i,j)应用菱形模板进行运动估计的过程.圆圈中的数字表示应用某种运动估计模板(如菱形模板)的先后顺序.$c^{i,j}$表示宏块(i,j)运动估计过程中的计算复杂度,即宏块的比较次数,其度量单位是搜索点数.而该宏块在不同计算复杂度下的预测失真度用 $d^{i,j}$表示,该值会随计算复杂度 $c^{i,j}$的不同而变化.宏块(i,j)的初始预测失真度,即在起始位置 1 的 $d_{init}^{i,j}$由两部分组成:一部分表示通过运动估计能够达到的最小预测失真度(全搜索算法的预测失真度),也就是不能通过运动估计消除的那部分失真度,用 $d_{\text{non-removable}}^{i,j}$表示.另一部分是通过运动估计可以消除的那部分预测失真度,用 $d_{\text{removable}}^{i,j}$表示.对某一宏块,其 $d_{\text{removable}}^{i,j}$和 $d_{\text{non-removable}}^{i,j}$都是固定的.使用不同的运动估计算法最多只能消除某个宏块的 $d_{\text{removable}}^{i,j}$部分.

我们知道,在运动估计过程中 $d^{i,j}$总是随着 $c^{i,j}$的增加而单调下降.第(i,j)*th* 宏块的计算复杂度与预测

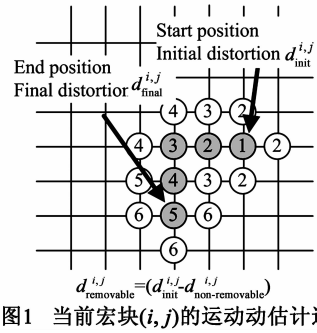


图1 当前宏块(i,j)的运动动估计过程

失真度关系可以用一个单调递减的函数表示.

$$d^{i,j} = F^{i,j}(c^{i,j}) \tag{1}$$

当运动估计花费计算资源越多的时候,该宏块的预测失真度就越低.如果不进行运动估计,宏块的预测失真度为初始运动估计预测失真度,也是预测失真度的最大值.该预测值由以下两部分组成:

$$c^{i,j}=0,\ d^{i,j} = d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}} \tag{2}$$

即包括可消除部分和不可消除部分.当运动估计可以无限使用计算资源,那么该宏块的预测失真度为:

$$c^{i,j}\to\infty,\ d^{i,j} = d^{i,j}_{\text{non-removable}} \tag{3}$$

即不可消除的部分.从上面所定义的关系可知,运动估计过程其实是一个令预测失真度由 $d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}}$ 到 $d^{i,j}_{\text{non-removable}}$ 变化的一个过程.图 2 是 $d^{i,j}$ 随 $c^{i,j}$ 变化的一个示意图.图中的粗实线是 $F^{i,j}(c^{i,j})$,即 $d^{i,j}$.直接使用一个数学模型表示 $F^{i,j}(c^{i,j})$ 是困难的,但通过大量的实验数据分析,我们发现计算复杂度越大,预测失真度的下降幅度会随之减慢.如图 2,$F^{i,j}(c^{i,j})$ 的下降速率 $dF^{i,j}/dc^{i,j}$ 会随着当前预测失真度与最小预测失真度的差值($F^{i,j}(c^{i,j}) - d^{i,j}_{\text{non-removable}}$)减少而同时趋向于 0.当 $F^{i,j}(c^{i,j})$ 与 $d^{i,j}_{\text{non-removable}}$ 相等时,$dF^{i,j}/dc^{i,j}=0$,预测失真度也不再减少.

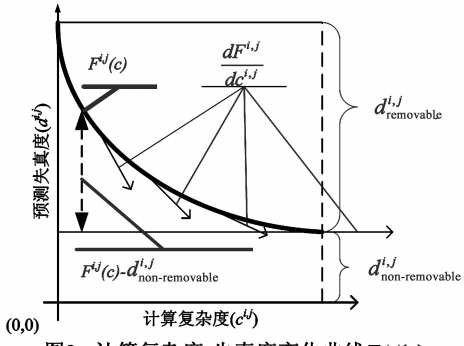


图2　计算复杂度-失真度变化曲线$F^{i,j}(c)$

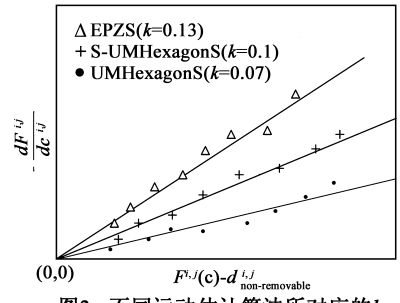


图3　不同运动估计算法所对应的k

我们对多个视频序列采用运动估计,然后标记不同时刻的预测失真度下降的速率 $dF^{i,j}/dc^{i,j}$ 和预测失真度与最小预测失真度之间的差值($F^{i,j}(c^{i,j}) - d^{i,j}_{\text{non-removable}}$).我们发现在使用相同运动估计算法的情况下两者成线性关系.图 3 是我们对 H.264 参考模型 JM[9] 中的三种运动估计算法所作的实验,其中为了令曲线存在于第一象限,我们使用 $-dF^{i,j}/dc^{i,j}$ 作为 y 轴.从图 3 可知,相同运动估计算法的 $-dF^{i,j}/dc^{i,j}$ 和($F^{i,j}(c^{i,j}) - d^{i,j}_{\text{non-removable}}$)不仅可以用一条直线拟合,而且该直线会经过原点.这是因为当 $F(c^{i,j}) - d^{i,j}_{\text{non-removable}}=0$ 时,$F^{i,j}(c^{i,j})$ 不再下降,$-dF^{i,j}/dc^{i,j}$ 也等于 0.有鉴于此,我们用下面的常微分方程描述这两者的关系:

$$\frac{dF^{i,j}}{dc^{i,j}} = -k\times(F(c^{i,j}) - d^{i,j}_{\text{non-removable}}) \tag{4}$$

其中,k 是一个参数,表示不同运动估计方法对预测失真度-复杂度曲线的影响.当 k 越大时,$-dF^{i,j}dc^{i,j}$ 增长越快,而 $F^{i,j}(c^{i,j})$ 下降越快.这说明该算法的效率越高,即用较短的时间就可搜索到最优位置.根据我们对实验数据的拟合,在联合模型 JM 中的三种运动估计算法:UMHexgonS[10]、EPZS[11] 和 S-UMHexgonS[12],其对应的 k 值分别是 0.13、0.1 和 0.07.公式(4)说明当前预测失真度与最小预测失真度相差越远时,在花费单位计算资源所能获得的预测失真度下降收益则越大,反之亦然.基于这个关系,加上公式(2),我们把求 $F^{i,j}(c^{i,j})$ 转化为常微分方程的初始值问题,如下:

$$\begin{cases}\dfrac{\mathrm{d}F^{i,j}}{\mathrm{d}c^{i,j}} = -k\times(F(c^{i,j}) - d^{i,j}_{\text{non-removable}})\\ F^{i,j}(0) = d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}}\end{cases} \tag{5}$$

其中,当计算复杂度为 0 时,初始的预测失真度是 $d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}}$.通过求解,我们可以得出如下的负指数模型:

$$d^{i,j} = F^{i,j}(c^{i,j}) = d^{i,j}_{\text{removable}}\times e^{-k\times c^{i,j}} + d^{i,j}_{\text{non-removable}} \tag{6}$$

根据这一模型,当 $c^{i,j}=0$ 的时候(没有运动估计),$d^{i,j} = d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}}$;而当 $c^{i,j}\to\infty$ 的时候(相当于全搜索,检查所有的搜索位置),$d^{i,j}\to d^{i,j}_{\text{non-removable}}$,即运动估计复杂度不断增加,实际预测失真强会最终等于不可消除的预测失真度.下面将简单地介绍该模型的物理特性.

我们知道,绝大多数快速运动估计算法的搜索原则是:越大机会成为最优点的位置,越有可能被优先搜索.在开始搜索时,预测失真度下降的速度较快,而随着搜索点数的增加(算法计算复杂度的增加),预测失真度的下降趋于平缓.这恰好与负指数曲线的斜率随着横坐标的增加而减少的特性相吻合.我们曾经用多个标准序列进行了大量的实验,其结果也证明了这一点.

我们的模型共有的三个参数,k、$d^{i,j}_{\text{removable}}$ 和 $d^{i,j}_{\text{non-removable}}$.其中,$k$ 表示所使用运动估计方法的有效程度,而其余两个参数用以适应不同的视频内容,会随着宏块的不同而变化.$d^{i,j}_{\text{removable}}$ 表示运动估计作用在该区

域上所能减少预测失真度.目前的运动估计只能消除视频中平移运动所产生的时间冗余.某个宏块的 $d^{i,j}_{\text{removable}}$值越大,表示该宏块运动估计可以减少的预测失真度越大,也就说明运动估计作用在这个宏块上越有效,应该分配更多的计算资源.而 $d^{i,j}_{\text{non-removable}}$则表示运动估计不可消除的预测失真度.这部分失真度主要由非平移运动所引起.例如视频中出现镜头的拉伸或遮蔽,这时候相应宏块的残差值会很大.就算在这些宏块上进行运动估计,也不能有效地减少预测失真度.直观上看,一幅图像中变化严重和运动剧烈的部分需要分配更多的运动估计资源,以消除更多的预测失真度.但是,实际上计算资源应该优先分配到 $d^{i,j}_{\text{removable}}$较大的宏块上,才能真正有效地降低预测失真度.因为变化严重只是 $d^{i,j}_{\text{removable}}+d^{i,j}_{\text{non-removable}}$的值比较大,并不代表可以通过运动估计降低这部分的预测失真度.对此,我们将在下一节中介绍如何根据上面的模型,最优地分配计算资源.

3 在复杂度约束下的优化

根据前一部分预测失真度和计算复杂度的指数模型,我们将在给定计算复杂度约束下,为不同宏块分配计算资源,以获得整帧最优的预测失真度.在帧层面上对模型进行优化时,我们发现通过设置预测失真度的阈值间接地为每个宏块分配计算复杂度,比直接控制不同宏块的计算配额更为简单和有效.根据下面的推导,该阈值就是在每个宏块的不可消除预测失真度上加上一个相同的固定值.只要动态根据计算约束调整这一固定值的大小,就可得到每一帧最优化的预测失真度.

有了上一节的运动估计计算复杂度和预测失真度关系模型,我们可以求出任意给定计算约束下所能达到的最优预测失真度.该问题就是在给定约束下求最小值的问题:

$$\min \sum d^{i,j},\ \text{s.t.}\ \sum c^{i,j} \leqslant C_{\text{constraint}} \tag{7}$$

其中,$d^{i,j}$和 $c^{i,j}$分别是第$(i,j)th$ 宏块的预测失真度和所分配的计算资源.整帧图像的最终预测失真度是所有宏块预测失真度之和,$\sum d^{i,j}$.而计算复杂度的消耗则是 $\sum c^{i,j}$,该消耗必须满足小于或等于给定的计算约束 $C_{\text{constraint}}$这一条件.为了简化后续的表达方式,我们把对总体计算复杂度的约束转换成对每个宏块平均计算约束:

$$A_{\text{constraint}} = \frac{1}{n} \times C_{\text{constraint}} \tag{8}$$

其中,n 是每帧图像中宏块的数目.然后,我们把上一节中得出的计算复杂度模型式(6)代入到该优化问题中,可得:

$$\min \sum d^{i,j}_{\text{removable}} \times \mathrm{e}^{-k\times c^{i,j}} + d^{i,j}_{\text{non-removable}},\quad \text{s.t.}\ \sum c^{i,j} \leqslant A_{\text{constraint}} \times n \tag{9}$$

如果系统的计算能力不受限制,即 $\sum c^{i,j} \to \infty$,那么当前帧的预测失真度(D)会达到最小值,即 $D \to \sum d^{i,j}_{\text{non-removable}}$.然而任何系统的计算能力总是有限的,高效的运动估计算法应该在给定计算能力的情况下,使 $\sum d^{i,j}$ 达到最小.因此,如何为不同宏块分配计算资源(即计算 $c^{i,j}$)在满足约束条件 $\sum c^{i,j} \leqslant A_{\text{constraint}} * n$ 的情况下,令 $\sum d^{i,j}$ 最小是我们要面对的问题.为此,我们采用拉格朗日乘子法把约束下的优化问题转成无约束下的优化问题:

$$\sum d^{i,j}_{\text{removable}} \times e^{-k\times c^{i,j}} + d^{i,j}_{\text{non-removable}} + \lambda\left(\frac{1}{n} \times \sum c^{i,j} - A_{\text{constraint}}\right) = 0 \tag{10}$$

并对该问题求解:

$$\begin{cases} c^{i,j} = -\dfrac{1}{k}\ln\left(\dfrac{\lambda}{k \times d^{i,j}_{\text{removable}}}\right) \\ \lambda = k \times \mathrm{e}^{-k\times A_{\text{constraint}}} \times \sqrt[n]{\prod\limits_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}} \end{cases} \tag{11}$$

把 λ 消除后可得:

$$c^{i,j} = A_{\text{constraint}} - \frac{1}{k}\ln\left(\frac{\sqrt[n]{\prod\limits_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}}}{d^{i,j}_{\text{removable}}}\right) \tag{12}$$

其中,k、n 和 $A_{\text{constraint}}$都是常数,分别表示运动估计模型的参数,每帧宏块的数目以及每个宏块的平均计算约束.从上面公式可知,每个宏块的 $c^{i,j}$不仅与自身的可消除预测残差相关,而且与图像中其它宏块的可消除预测残差都有联系,直接通过上面公式计算 $c^{i,j}$必须准确地预测图像中各个宏块的可消除预测残差.同时,因为模型的式(6)主要参数由失真度 $d^{i,j}_{\text{removable}}$和 $d^{i,j}_{\text{non-removable}}$构成,使用这些参数来表示 $c^{i,j}$会较为复杂,并不直观.

为此,我们使用通过控制预测失真度阈值的方法,间接地为宏块分配计算复杂度.每个宏块都设置一个阈值,当运动估计过程中的预测失真度达到这个阈值的时候,则终止对该宏块的搜索.要通过预测失真度阈值分配计算复杂度,我们必须转换模型公式(6),把计算复杂度 $c^{i,j}$变成 $d^{i,j}$的函数:

$$c^{i,j} = -\frac{1}{k} \times \ln\left(\frac{d^{i,j} - d^{i,j}_{\text{non-removable}}}{d^{i,j}_{\text{removable}}}\right) \tag{13}$$

然后,代入上面的优化模型式(9),可得:

$$\min \sum d^{i,j},\quad \text{s.t.}\ \frac{1}{n} \times \sum\left(-\frac{1}{k} \times \ln\left(\frac{d^{i,j} - d^{i,j}_{\text{non-removable}}}{d^{i,j}_{\text{removable}}}\right)\right) \leqslant A_{\text{constraint}} \tag{14}$$

再使用拉格朗日乘子法:

$$\sum d^{i,j} + \lambda\left(\frac{1}{n} \times \sum\left(-\frac{1}{k} \times \ln\left(\frac{d^{i,j} - d^{i,j}_{\text{non-removable}}}{d^{i,j}_{\text{removable}}}\right)\right)\right. \\ \left. - A_{\text{constraint}}\right) = 0 \tag{15}$$

求解如下:

$$\begin{cases} d^{i,j} = d^{i,j}_{\text{non-removable}} + \dfrac{\lambda}{k} \\ \lambda = k \times \mathrm{e}^{-k \times A_{\text{constraint}}} \times \sqrt[n]{\prod_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}} \end{cases} \tag{16}$$

其中,在一帧图像中 λ/k 可以看成一个常数.不同计算约束下的失真度阈值其实就是不可消除的预测残差 $d^{i,j}_{\text{non-removable}}$(最小预测残差)加上这一常数.这个常数的大小根据计算约束 $A_{\text{constraint}}$ 计算得出.虽然 λ 的计算比较复杂,但并不需要直接使用公式(16)计算.下一节我们将给出一种自适应的方法,找出最适合的 λ/k.

综上所述,使用公式(16)的关键在于对 $d^{i,j}_{\text{non-removable}}$ 的计算,而使用公式(12)的关键在于对 $d^{i,j}_{\text{removable}}$ 的计算.由于无论 $d^{i,j}_{\text{non-removable}}$ 还是 $d^{i,j}_{\text{removable}}$ 都无法在完成运动估计前准确知道,所以只能通过时域相邻宏块的预测失真度近似预测.该预测的准确度就决定了采用不同公式分配计算资源的优化程度.图 4 是我们对多个视频序列编码,使用当前宏块的 $d^{i,j}_{\text{non-removable}}$ 和 $d^{i,j}_{\text{removable}}$ 分别减去前一帧对应宏块的 $d^{i,j}_{\text{non-removable}(t-1)}$ 和 $d^{i,j}_{\text{removable}(t-1)}$ 的概率密度函数.从图可知 $d^{i,j}_{\text{non-removable}} - d^{i,j}_{\text{non-removable}(t-1)}$ 的概率分布远比 $d^{i,j}_{\text{removable}} - d^{i,j}_{\text{removable}(t-1)}$ 集中.这说明对 $d^{i,j}_{\text{non-removable}(t-1)}$ 实施预测,其准确性会更高,从而使用公式(16)能更优化地分配计算资源.

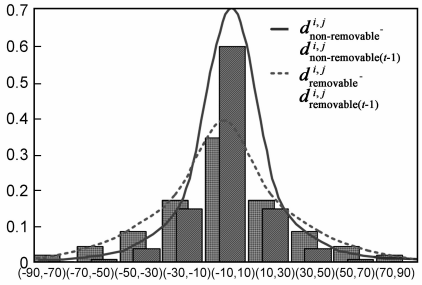


图4　$d^{i,j}_{\text{non-removable}}$-$d^{i,j}_{\text{non-removable}(t-1)}$和$d^{i,j}_{\text{removable}}$-$d^{i,j}_{\text{removable}(t-1)}$的概率分布

4　自适应优化算法

根据上一节的结论,在一个给定的计算约束下,要令整帧的预测失真度达到最小,可以根据公式(16)为不同的宏块设置预测失真度阈值,分配计算资源,而且这个阈值的设定是在每个宏块的不可消除预测失真度上加上一个统一的固定值(λ/k).直接通过公式(16)计算 λ 不仅需要预先知道整帧图像内所有宏块的可消除预测残差,而且比较复杂.我们将给出一种自适应的算法,根据给定的 $A_{\text{constraint}}$ 自动调整 λ/k 使算法满足平均计算复杂度约束.为了简化公式的表达,下面用 Δd 统一表示 λ/k.

我们的算法有两个关键步骤:(1)计算每帧所对应的 Δd.我们利用帧与帧之间的相关性动态地对 Δd 进行逼近,以使平均计算复杂度满足计算约束 $A_{\text{constraint}}$.(2)预测每个宏块的不可消除预测残差.因为准确的 $d^{i,j}_{\text{non-removable}}$ 只能在宏块进行了完全的运估计后才能得到,所以一般的做法是用前一帧相应宏块的不可消除预测残差($d^{i,j}_{\text{non-removable}(t-1)}$)作为近似.但是,由于计算复杂度约束环境下的运动估计都提前终止在某一阈值,并不保证达到最小的预测失真度,那么已经编码宏块的不可消除残差值不能直接得到.为此,我们给出一种根据前一宏块的实际计算复杂度和实际预测残差值推算出 $d^{i,j}_{\text{non-removable}}$ 的方法.下面我们将分析如何计算 Δd 和 $d^{i,j}_{\text{non-removable}}$,并给出总体的算法.

首先,为每帧计算 Δd.根据公式(16)可得:

$$\Delta d = \frac{\lambda}{k} = \mathrm{e}^{-k \times A_{\text{constraint}}} \times \sqrt[n]{\prod_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}} \tag{17}$$

其中,$\sqrt[n]{\prod_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}}$ 是所有宏块可消除预测失真度的几何平均数,对其直接计算需要逐个宏块进行预测,既复杂也难以保证准确.虽然上面已经提到各个宏块 $d^{i,j}_{\text{removable}}$ 的时域相关性比 $d^{i,j}_{\text{non-removable}}$ 要低,但帧与帧之间 $d^{i,j}_{\text{removable}}$ 的几何平均数的时域相关性还是比较高的.这是因为前后帧相同宏块的内容可以变化很大,而前后帧总体内容的变化则相对较小.因而,导致了 $d^{i,j}_{\text{removable}}$ 几何均值的变化会比单个 $d^{i,j}_{\text{removable}}$ 的变化要低.为此,我们利用前一帧实际消耗的计算复杂度 A_{t-1} 以及所对应的? d_{t-1} 得出该均值的近似:

$$\sqrt[n]{\prod_{k=0,l=0}^{n} d^{k,l}_{\text{removable}}} = \frac{\Delta d}{\mathrm{e}^{-k \times A_{\text{constraint}}}} \approx \frac{\Delta d_{t-1}}{\mathrm{e}^{-k \times A_{t-1}}} \tag{18}$$

再把该结果代入公式(17)可得:

$$\Delta d = \Delta d_{t-1} \times \mathrm{e}^{-k \times (A_{\text{constraint}} - A_{t-1})} \tag{19}$$

就可计算出 Δd.

另外,计算完 Δd 后,我们要预测每个宏块的 $d^{i,j}_{\text{non-removable}}$.相邻帧对应宏块间的 $d^{i,j}_{\text{non-removable}}$ 有着较强的相关性,可以用前一帧对应宏块的 $d^{i,j}_{\text{non-removable}(t-1)}$ 替代.但是,由于各个宏块的运动估计都在计算受限的情况下进行,其最终预测的预测失真度 $d^{i,j}_{\text{final}(t-1)}$ 并不等于 $d^{i,j}_{\text{non-removable}(t-1)}$.我们除了把上一帧对应宏块所消耗的计算复杂度和最终预测失真度代入公式(6),还需要利用当前宏块的初始预测失真度.联立这两个方程如下:

$$\begin{cases} d^{i,j}_{\text{initial}} = d^{i,j}_{\text{removable}} + d^{i,j}_{\text{non-removable}} \\ d^{i,j}_{\text{final}(t-1)} = d^{i,j}_{\text{removable}} \times \mathrm{e}^{-k \times c^{i,j}_{(t-1)}} + d^{i,j}_{\text{non-removable}} \end{cases} \tag{20}$$

其中，$d^{i,j}_{\text{initial}}$是当前宏块的初始预测失真度，而 $d^{i,j}_{\text{final}(t-1)}$和 $c^{i,j}_{(t-1)}$是前一帧对应宏块的计算复杂度和相应的最终预测失真度.第一个方程表示当前初始的预测失真度等于 $d^{i,j}_{\text{removable}}$与 $d^{i,j}_{\text{non-removable}}$之和.第二个方程是 $d^{i,j}_{\text{final}(t-1)}$和 $c^{i,j}_{(t-1)}$的关系.解该方程组可得：

$$d^{i,j}_{\text{non-removable}}=\frac{(d^{i,j}_{\text{final}(t-1)}-d^{i,j}_{\text{initial}}\times e^{-k\times c^{i,j}_{(t-1)}})}{(1-e^{-k\times c^{i,j}_{(t-1)}})} \quad (21)$$

公式(19)与式(21)之和就是每个宏块所对应的最优预测失真度阈值.

图5是最优化阈值设置的一个例子.图中共有5条复杂度失真度曲线，每条曲线所对应的最优阈值(optimal $d^{i,j}$)就是 $d^{i,j}_{\text{non-removable}}+\Delta d$.$\Delta d$由计算复杂度约束所决定.计算复杂度约束越小$\Delta d$则越高，反之亦然.两者成反比关系.从图中我们可以直观地知道，当一个宏块的 $d^{i,j}_{\text{removable}}$越大，其对应的优化阈值则越靠右，所分配的计算资源也会越多.这是因为优化的分配方案总是优先把资源使用在最有效减少失真度的宏块上.

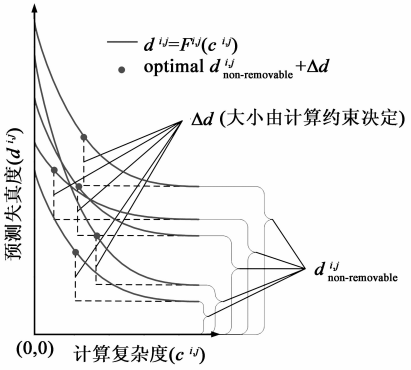


图5 不同宏块最优预测失真度阈值的设置

最后，我们给出算法总体的实现步骤如下：

step1 不考虑任何计算约束，得到首帧所有宏块的最小预测残差，即 $d^{i,j}_{\text{non-removable}}$.

step2 根据给定的 $A_{\text{constraint}}$，为后续每帧计算 $\Delta d=\Delta d_{t-1}\times e^{-k\times(A_{\text{constraint}}-A_{t-1})}$.

step3 计算这些后续帧中每个宏块的 $d^{i,j}_{\text{non-removable}}=\frac{(d^{i,j}_{\text{final}(t-1)}-d^{i,j}_{\text{initial}}\times e^{-k\times c^{i,j}_{(t-1)}})}{(1-e^{-k\times c^{i,j}_{(t-1)}})}$

step4 对每个宏块设置提前终止条件为 $d^{i,j}_{\text{non-removable}}+\Delta d$.

step5 下一帧运动估计开始时重复执行 step2.

5 实验结果

为了评估本算法的性能，我们配合H.264的联合模型(Joint Model 15.0，JM)[9]对多个样本视频序列(包括CIF，4CIF，720p)进行了测试.总体实验结果表明，本算法不仅可以准确地控制运动估计的计算复杂度，而且在不同的约束下都能提供比其它算法优秀的率失真性能.因为篇幅有限，下面只列出高质量($QP=16$)和低质量($QP=26$)两种情况下的部分结果.所有序列的编码帧率为30fps，编码结构为IPPP.采用固定量化参数和高精度率失真优化选项，运动估计算法为EPZS($k=0.1$).虽然测试在JM上进行，但为避免实现的平台对算法的影响，本文使用宏块的平均搜索点数作为运动估计复杂度的量度.另外，本文使用运动估计后的预测失真度MSE(Mean Square Error)或码率和信噪比表示运动估计的率失真性能.平均搜索点数和对应的MSE越低，说明算法效率越高.

首先，我们在表1给出本算法作用在4种不同分辨率的样本序列上的实验结果，以此表明本算法能提供准确的复杂度控制和可伸缩的率失真性能.表中的$A_{\text{constraint}}$表示本文方法所设置的平均计算复杂度约束，A_{actual}表示实际量度出来的宏块平均搜索点数.A_{actual}与$A_{\text{constraint}}$越接近，说明本文的方法对运动估计的复杂度控制越准确.从表1的数据可以知道，本算法的 A_{actual}

表1 本文的算法对复杂度控制的实验结果

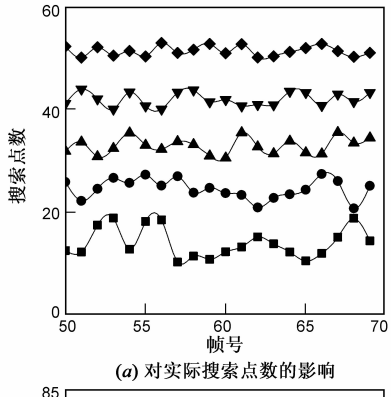
QP	$A_{\text{constraint}}$	Foreman (CIF)		Mobile (4CIF)		Parkrun (720p)		Mocal(720p)	
		MSE	A_{actual}	MSE	A_{actua}	MSE	A_{actual}	MSE	A_{actual}
	10	106	8	121	9	72	8	99	10
	20	86	21	112	20	43	21	85	20
	30	83	29	95	31	39	30	78	31
	40	80	41	88	41	37	40	68	38
	50	74	51	82	51	40	51	62	49
	10	110	9	106	9	76	9	91	11
	20	91	19	97	19	48	21	82	18
	30	79	30	98	29	40	30	76	28
	40	76	38	90	41	40	38	65	41
	50	73	48	85	51	32	48	63	50

与 $A_{constraint}$的平均差值只有 2.2.同时,运动估计的预测失真度 MSE 也随着复杂度的增加而分级递减.目前 $A_{constraint}$是一个固定值,但也可以对其动态设置,以适应多变的实际应用需求.

为了说明视频内容变化对本算法的影响,图 6(*a*)和(*b*)分别标识了 Forman 序列第 50 帧到第 70 帧 MSE 和实际搜索点数的变化情况.图 6 中的横坐标是帧号,纵坐标分别是 MSE 和实际搜索点数.在图 6(*b*)的 55 ~ 60 帧之间 MSE 有一个明显的下降,这说明了该时段视频的运动由剧烈转向平缓.但观察图 6(*a*)对应的位置,实际搜索点数并不随着运动情况的不同,而产生变化.相反,随着计算约束 $A_{constraint}$逐渐增大,本算法无论在运动剧烈或是平缓的时候,都能提供分级的 MSE.当约束 $A_{constraint}$越低时,实际搜索点数与 $A_{constraint}$的差值越大.由于本算法优先搜索运动相对剧烈的宏块,而这些宏块需要耗费更多的搜索点数,那么在低复杂度的情况下本算法对复杂度调整的粒度就会较大.

图 7 是本算法在不同计算约束下的率失真曲线.从图中可以看出,随着计算约束 $A_{constraint}$的增加,算法的率失真性能随之上升.而图 7(*b*)中的率失真曲线间隔明显比图 7(*a*)中的曲线间隔要大,这说明在低分辨率序列中增加运动估计算法复杂度所获得的率失真收益,比在高清序列中要明显.除此之外,在不同分辨率的情况下,算法在高码率的时对计算复杂度所产生的影响大于低码率的情况.综上两点说明了运动估计在低图像分辨率和高码率的情况下有更好的率失真收益.

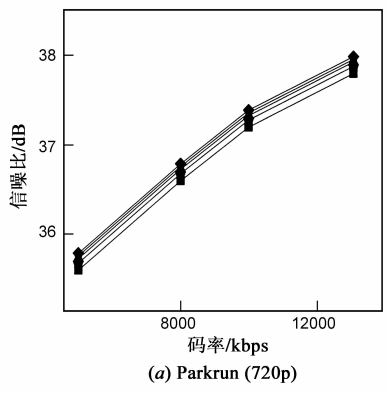
其次,图 8 给出本算法与其它算法(FADTS[5]和 SU's CAME[6])的比较结果.通过比较可以明显看出本算法在相同计算复杂度的情况下,能提供更优秀的率失真性能.图 8 是各种算法作用在 Foreman 和 Parkrun 序列上的计算复杂度 – 预测失真度曲线.其中,横坐标表示计算复杂度,纵坐标表示预测失真度(MSE).当个算法的计算复杂度增加,预测失真度都会减少,所以图中所有的曲线总是单调下降.从图 8 中可以看出,本算法的计算复杂度-预测失真度曲线下降得最快,总位于其余两条曲线的左端.曲线下降得越快,则说明在相同计算复杂度的情况下,本算法能达到比其它算法更低的预测失真度.



(*a*) 对实际搜索点数的影响

(*b*) 对MSE的影响

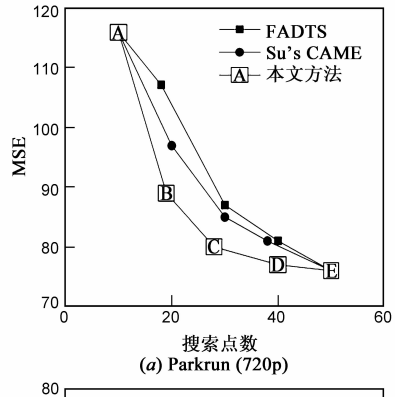
图6　运动情况的不同对本算法的影响



(*a*) Parkrun (720p)

(*b*) Forman (CIF)

图7　本文算法的率失真性能变化曲线



(*a*) Parkrun (720p)

(*b*) Foreman (CIF)

图8　不同算法的复杂度-预测失真度曲线

6　结论

本文针对视频编码中的运动估计过程,提出了一种在计算约束下的率失真优化方法.该方法为各个宏块设置不同的预测失真度阈值,优化地分配运动估计计算资源.我们首先采用一个指数模型描述运动估计过程中预测失真度与计算复杂度之间的关系,然后在帧层面上对该模型优化,从而建立了一个简洁的算法,自适应地计算不同宏块所对应的阈值.与其它复杂度可伸缩的运动估计方法相比,本文的方法总是在耗费

相同计算量的情况下,达到更优化的率失真性能.

参考文献

[1] 陈胜刚,陈书明,谷会涛,刘尧[J].一种用于并行 h264 编码器的语法元素级分组并行算术编码器体系结构的评估.电子学报,2012,40(2):400 – 405.
Chen Shenggang, Chen Shuming, Gu Huitao et al. A VLSI architecture evaluation of a syntax element level parallel arithmetic entropy coder for parallel H.264 Encoder[J]. Acta Electronica Sinica, 2012, 40(2): 400 – 405. (in Chinese)

[2] 王▮,刘贵忠,钱学明.一种高效的基于 h264/avc 压缩域信息的全局运动估计方法[J].电子学报,2011,39(3A):19 – 23.
Wang Zhe, Liu Guizhong, Qian Xueming. An efficient global motion estimation algorithm on H.264/AVC compression domain[J]. Acta Electronica Sinica, 2011, 39(3A): 19 – 23. (in Chinese)

[3] C S Kannangara, I E Richardson, A J Miller. Computational complexity management of a real-time h.264/avc encoder[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(9): 1191 – 1200.

[4] Rahul Vanam, Eve Riskin, and Richard Ladner. H.264/mpeg-4 avc encoder parameter selection algorithms for complexity distortion tradeoff [A]. Proceedings of Data Compression Conference 2009 [C]. Snowbird, UT, USA : Data Compression Conference, 2009. 372-381.

[5] Sorwar Golam, Murshed Manzur, S Dooley Laurence. A fully adaptive distance-dependent thresholding search (FADTS) algorithm for performance – management motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2007, 17(4): 429 – 440.

[6] Li Su, Yan Lu, Feng Wu, Shipeng Li, Wen Gao. Complexity-constrained h.264 video encoding[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2009, 19(4): 477 – 490.

[7] Yongfang Liang, Ishfaq Ahmad. Power and distortion optimization for pervasive video coding[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2009, 19(10): 1436 – 1447.

[8] Jiyuan Lu, Peizhao Zhang, Hongyang Chao, Paul Fisher. On combining fractional-pixel interpolation and motion estimation: A cost-effective approach[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21(6): 717 – 728.

[9] H.264 reference software version jm15.0[EB/OL]. http://iphome.hhi.de/download/jm15.0.zip, 2011-1-6.

[10] Zhibo Chen, Jianfeng Xu, Yun He, Junli Zheng. Fast integer-pel and fractional-pel motion estimation for h.264/avc[J]. Journal of Visual Communication and Image Representation, 2006, 17(2): 264 – 290.

[11] Alexis Michael Tourapis, Hye-Yeon Cheong, Pankaj Topiwala. Fast me in the jm reference software[A]. jvt-p026.doc [C]. Poznan, Poland: Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU – T SG16 Q.6), 2005. 1 – 13.

[12] Xiaoquan Yi, Jun Zhang, Nam Ling, Weijia Shang. Improved and simplified fast motion estimation for jm[A]. jvt-p021.doc [C]. Poznan, Poland : Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), 2005. 1 – 15.

作者简介

陆寄远 男,1976 年生于广东广州,2011 年获中山大学工学博士学位.现为广东金融学院系统分析师.研究方向为图像处理、视频编码.
E-mail: dtc005001@163.com

朝红阳 女,1988 年获中山大学理学博士学位,现任中山大学软件学院教授,信息科学院博士生导师.主要研究方向为图像处理,视频编码.