软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(8):2214−2226 [doi: 10.13328/j.cnki.jos.005122]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

HEVC 的高效分像素运动补偿*

陆寄远[1], 刘宇熹[1], 侯 昉[1], 黄承慧[1], 朝红阳[2]

[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 510006)

通讯作者: 陆寄远, E-mail: dtc005001@163.com

摘 要: 现有的视频编码方法单独针对不同的编码技术进行优化,虽然简化了算法的实现,但却忽略了这些技术之间的内在关系而局限了性能的进一步提升.尝试针对 HEVC 中的分像素插值和分像素运动估计,建立一种整合的优化算法,同时提升两者的计算速度.首先,通过细分不同分像素的插值方法,把每个分像素的插值代价融合在分像素运动估计的搜索中,并构建一种性价比优先的分像素运动估计方法.该运动估计方法的每一步搜索都按照最小化计算代价和最大化率失真收益的原则进行,保证以最小的计算消耗获得最优的性能.其次,为了配合该方法,针对 HEVC 的特性构建了一种分区域分像素集的分像素插值算法.该插值算法在分像素运动估计过程中仅动态地计算所用到的分像素,解决了现有插值算法无法同时降低重复计算和冗余计算的问题.实验结果表明,该整合算法可以在几乎不产生率失真性能损失的情况下,大幅度地加快分像素运动估计和插值的速度.

关键词: HEVC;分像素运动估计;分像素插值;性价比优先算法

中图法分类号: TP391

中文引用格式: 陆寄远,刘宇熹,侯昉,黄承慧,朝红阳.HEVC 的高效分像素运动补偿.软件学报,2017,28(8):2214−2226. http://www.jos.org.cn/1000-9825/5122.htm

英文引用格式: Lu JY, Liu YX, Hou F, Huang CH, Chao HY. High efficiency algorithm of fractional pixel motion compensation for HEVC. Ruan Jian Xue Bao/Journal of Software, 2017,28(8):2214−2226 (in Chinese). http://www.jos.org.cn/1000-9825/5122.htm

High Efficiency Algorithm of Fractional Pixel Motion Compensation for HEVC

LU Ji-Yuan[1], LIU Yu-Xi[1], HOU Fang[1], HUANG Cheng-Hui[1], CHAO Hong-Yang[2]

[1](Department of Computer Science and Technology, Guangdong University of Finance, Guangzhou 510521, China)

[2](School of Software, Sun Yat-Sen University, Guangzhou 510006, China)

Abstract: Separately optimizing different video coding technologies simplifies the implementation of coding algorithms, but it results in slowdown by not taking their internal relation into consideration. An integrated algorithm is proposed to jointly improve the speed of fractional pixel interpolation (FPI) and fractional pixel motion estimation (FPME) for HEVC. Different fractional pixels are distinguished by their interpolation filters to construct a FPME algorithm with the knowledge of their interpolation cost. The FPME algorithm produces a cost effective order for refinement search which checks fractional search positions (SPs) by not only maximizing the expectation for the R-D gains but also minimizing the computational cost at the same time. On the other hand, an on-demand FPI algorithm for HEVC is also proposed to save redundant interpolation as well as reduce duplicate calculation. Experimental results show that the introduced algorithm significantly improves the overall speed of FPME and FPI with negligible coding lost.

Key words: HEVC; fractional pixel motion estimation; fractional pixel interpolation; cost effective algorithm

* 基金项目: 国家自然科学基金(61173081); 广东省自然科学基金(2014A030313662)

Foundation item: National Natural Science Foundation of China (61173081); Guangdong Natural Science Foundation of China (2014A030313662)

收稿时间: 2015-01-20; 修改时间: 2015-05-29, 2016-03-18; 采用时间: 2016-07-08; jos 在线出版时间: 2016-10-11

CNKI 网络优先出版: 2016-10-12 16:26:50, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.016.html

由于视频编码技术的不断更新换代,视频数据的编码效率得到了大幅度的提升.其中,分像素运动补偿是目前一种可以明显提高编码效率和视频重建质量的技术.该技术通过消除视频信号中的频率混淆实现编码性能的提高.采用该技术需要增加以下两个编码过程:分像素插值(fractional-pixel interpolation,简称 FPI)和分像素运动估计(fractional-pixel motion estimation,简称 FPME).根据 Minoo 等人的研究,这两部分的计算在视频编码过程中占了很大的比例[1].

分像素插值的计算消耗主要来源于像素生成的计算,而且由于分像素插值技术日趋复杂,如 1/8 像素精度的运动补偿(1/8 pixel motion compensation)和自适应插值滤波(an adaptive interpolation filter,简称 AIF)等技术的引入,都将进一步导致单个分像素的计算复杂度以及需要计算分像素总量的大幅增加.另一方面,分像素运动估计的计算消耗主要来源于搜索最优分像素运动矢量的计算.随着分像素运动矢量精度的不断提高以及候选搜索区域的不断扩大,分像素运动估计越来越趋于复杂.常用的 1/4 像素精度运动补偿技术,分像素候选搜索区域为 7×7;而 1/8 像素精度运动补偿技术,其分像素运动估计的搜索区域扩充为 15×15,这相当于整像素运动估计搜索区域的大小.运动估计搜索区域的扩大,直接导致了分像素运动估计时间的增加.

现有的快速算法仅单独地针对分像素运动估计和分像素插值进行优化,但从本质上看,两者是密切相关的.分像素插值是为了生成分像素运动估计所需要的像素,为此,分像素插值计算量与分像素运动估计所检查的搜索位置数(search positions,简称 SPs)直接相关.根据我们的前期调研[1],有效的分像素运动估计和分像素插值不仅要在不影响精度的情况下减少运动估计中搜索位置的数目,而且也要尽量避免多余的分像素插值.本文将以这个目标为出发点,提出一种面向 HEVC[2]的分像素运动估计和分像素插值协同优化算法.

现有的绝大多数分像素运动估计算法都假设所有的分像素已经在运动估计前预先计算.为此,快速分像素运动估计算法往往只着重于如何减少运动估计的搜索数.然而,不同搜索位置所对应分像素的插值滤波器是不同的,单纯地减少搜索数目而不考虑其对应插值代价的差异性,会无法更进一步地优化分像素插值.为了更好地说明本文所解决的问题及难点,下面将分别针对分像素运动估计和分像素插值给出相关的文献回顾和问题难点剖析.

分像素运动估计是在分像素候选区域中找到与当前分块最匹配位置的过程.这些快速算法一般有两个步骤:首先,通过相邻运动矢量的相关性或某种分像素残差模型预测最优的分像素运动矢量(motion vector,简称 MV);然后,在该预测矢量的周围应用一个细化搜索模板进行校正.目前已出现了不少快速分像素运动估计方法,我们根据分像素运动矢量预测技术的不同,把这些方法分成以下两类.

- 第 1 类是基于相邻分像素预测运动矢量的算法.

这类算法利用相邻宏块运动矢量的相似性确定一个起始搜索位置,然后采用细化模板(如菱形模板)搜索.其中,CBFPS[3]是一种被广泛使用的快速算法,它应用中值运动矢量作为起始搜索位置.HAPBFS[4]是在 CBFPS 基础上的一个改进算法,利用中值预测矢量和上层预测矢量的分数部分来预测.上述这些方法多数从整像素运动估计方法扩展而来,所以缺少专门针对分像素运动估计的特性而优化的部分.如果能够利用分像素运动估计的特性,算法的速度仍有进一步提升的空间.

- 第 2 类是基于参数模型的快速分像素运动估计算法.

这类方法通过建立不同的分像素残差曲面模型预测最优分像素运动矢量.模型参数通过相邻整像素位置的残差值计算.传统的参数模型有线性模型、一维和二维的抛物线模型[5]以及 Hill 等人[6]和 Dikbas 等人[7]使用的多项式模型.但是,这些理论上的模型总会或多或少地与实际残差曲面存在误差,所以后续采用细化搜索校正是必要的.目前,这些细化搜索策略仅考虑了不同分像素搜索位置的率失真性能而忽视了这些搜索位置所对应的计算代价.不同分像素搜索位置的计算代价本质上由比较运算和插值运算两部分构成,有效的细化搜索算法需要在最大化率失真收益的同时最小化计算消耗.我们在国际数据压缩年会(Data Compression Conf.)发表的前期工作[1]给出了一个关于如何整合分像素插值和分像素运动估计的初步想法,但是许多处理上的细节仍待进一步厘清.本文将给出实现该想法的一种可行手段.

另一方面,除分像素运动估计外,对于分像素的插值计算也出现了不少的算法.其中,最简单的分像素插值

算法是预先计算所有分像素的全插值算法.因为所有的分像素只被计算 1 次,所以全插值算法不产生任何的重复计算.但是,大部分的分像素在实际编码过程中不会被访问到,所以全插值算法的冗余计算最为严重.除此以外,全插值算法还需要耗费大量的内存空间保存所有的分像素,占用空间的大小与分像素运动矢量的精度和参考帧的数目成正比.为了克服这些问题,Minoo 等人[8]提出了一种 Reciprocal 算法来减少分数运动估计对内存和计算的消耗.该方法并不对参考帧插值,而是对当前帧插值.它利用了运动的相对性,在当前块中找出最匹配的参考块.Reciprocal 算法能够正确找到最优分像素运动矢量的重要前提是:视频信号只存在平移运动.但在现实中,视频除了拥有平移运动以外,还存在许多其他类型的运动.这不仅会引起 Reciprocal 算法的误差,而且会带来率失真性能上的损失.与其他快速算法不同,分像素插值算法必须是无损的快速算法,其计算必须严格遵照标准中的定义,否则将引发编解码端参考帧不一致的问题(drifting),误差不断累积.

此外,开源的编码器 X264[9]采用了一种针对 H.264 视频编码标准动态计算 1/4 像素的插值方法.下面将这种方法称为 XFPI 方法.H.264 中分像素的计算顺序是先计算 1/2 像素,然后计算 1/4 分像素.XFPI 方法正是利用了这个特性,预先计算不同参考帧所有的 1/2 像素,然后在分像素运动估计过程中动态计算所用到的 1/4 像素.XFPI 虽然充分地利用了 H.264 分像素插值的特性,但仍存在不少的冗余计算和重复计算.冗余计算主要表现在大量的 1/2 像素不被使用(特别在应用快速分像素运动估计时),重复计算主要表现在多次用到的 1/4 像素每次都必须重新计算(特别在采用多分块模式运动补偿时,该情况尤为严重).

综上所述,分像素插值算法的关键在于既要减少重复计算,同时也要消除冗余计算.重复计算是指由于同一个分像素在编码过程中被多次使用而进行了多次的计算;冗余计算是指在编码过程中对那部分不需要用到像素的计算.两者是互斥的,如果要完全消除冗余计算,必然会引发大量的重复计算;反之亦然.由上述算法可以看出,全插值算法为了避免重复计算导致了大量的冗余计算和内存消耗.另一方面,如果所有的分像素都动态计算,那么视频编码标准中所定义的分像素依赖关系将引发十分严重的重复计算.XFPI 虽然作为这两者中的一个平衡点(预先计算所有的 1/2 像素,动态计算所有的 1/4 像素),但是也不能完全解决重复计算和冗余计算的问题.真正有效的分像素插值算法既要消除重复计算也要减少冗余计算.

针对上述分像素运动估计和分像素插值的这些问题,我们的算法有以下两个主要贡献.首先,本文针对分像素运动估计的细化搜索给出了一个性价比优先的搜索顺序.该顺序不仅最大化每步搜索的率失真收益,同时最小化每步的计算代价(包括运动估计代价和分像素插值代价).其次,本文给出了一种适合 HEVC 视频编码标准中分像素计算关系的动态插值方法,所有的分像素采用分区域分像素集的方式插值,在消除重复计算和减少冗余计算两者之间获取最优的平衡.

1 HEVC 和 H.264 分像素运动补偿技术比较和分析

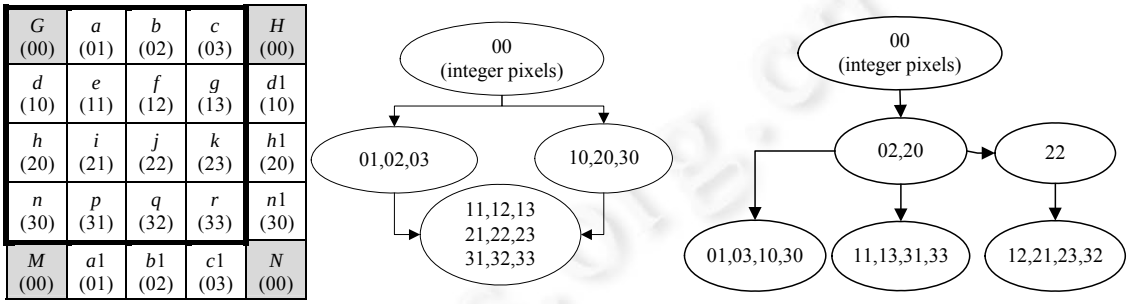
本节将通过比较 HEVC 和 H.264 两个标准中定义的分像素运动补偿方法,说明 H.264 中使用的分层式滤波器只能很好地配合传统的全分像素运动估计算法(full fractional pixel search,简称 FFPS),而 HEVC 中所定义的可分离滤波器则可灵活地配合快速分像素运动估计算法使用.

总括来说,H.264 优先计算 1/2 像素,然后再计算所有的 1/4 像素;HEVC 则优先计算与整像素同一行或同一列的分像素,然后再计算剩余的分像素.因为 H.264 定义的插值顺序与 FFPS 方法的搜索顺序吻合,H.264 编码器在使用 FFPS 方法进行搜索时可以采用动态插值算法(如 XFPI 算法)减少冗余计算.但是,绝大多数的快速分像素运动估计方法[3−5]并不遵照 H.264 所定义的插值顺序进行搜索.另一方面,HEVC 提供了更为灵活的插值方法,只要算法设计恰当,即可大幅度地减少多余的分像素插值和运动估计消耗.

图 1(a)为 HEVC 和 H.264 标准中分像素的计算示意图.大写字母所在的阴影方格表示整像素,小写字母所在的空白方格表示分像素.在所有的分像素当中,a,b,c 和 d,h,n 分别表示与整像素同一行或同一列的分像素,其他的则是不与整像素正交的分像素.HEVC 和 H.264 为了消除视频信号中的频率混淆,定义了不同的低通滤波器.HEVC 定义的是一组可分离滤波器,而 H.264 定义的是一组分层滤波器.由于分像素的数目 15 倍于原始视频信号的大小,所以计算和存储这些分像素都将耗费极大的资源.

我们把所有分像素根据各自插值滤波器和插值方法的不同划分为 15 个不同的像素集.图 1(b)所示为(*G*,*H*,*M* 和 *N*)4 个整像素包围的区域.不同像素中括号里的数字表示不同的插值像素集.例如,“*a*”属于 01 插值像素集,“*c*”属于 03 插值像素集,“*d*”属于 10 插值像素集等.其中,02,20,22 插值像素集是 3 个 1/2 插值像素集;00 表示整像素集;剩下的是 1/4 插值像素集.

图 1(b)表示 HEVC 所定义的分像素依赖关系.根据 HEVC 的定义,与整像素水平或垂直的分像素可以通过一个 8 抽头的滤波器从整像素中计算得出.换句话说,图中的 01,02,03,10,20,30 像素集可以由整像素集 00 直接算出.其余的分像素集的计算则依赖于它们.图 1(c)表示 H.264 所定义的分像素依赖关系.与 HEVC 所定义的依赖关系不同,1/2 像素集 02,20 和 22 首先被计算,1/4 像素集的计算依赖于这些 1/2 像素集.



(a) 4 个相邻整像素区域 (b) HEVC 中的依赖关系 (c) H.264 中的依赖关系

Fig.1 Interger/Fractional pixels set diagram

图 1 整/分像素集示意图

分像素运动估计在最优整像素运动矢量周围一个像素的区域内找出最匹配的分像素运动矢量.图 2 是分像素搜索区域中 FFPS 和快速分像素运动估计算法的示例.分像素搜索区域是图中虚线所包围的一个 7×7 的区域.除了中心的整像素位置外,该分像素搜索区域有 48 个候选分像素搜索位置.分像素运动估计把当前块与不同分像素搜索位置的分像素相比较,找出最匹配的位置.不同搜索位置所对应的分像素总是隶属于同一个插值像素集,即这些分像素采用相同的滤波器和插值方法计算.

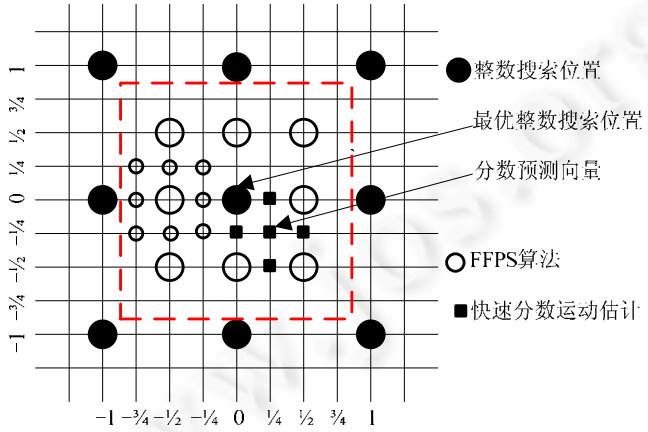


Fig.2 FFPS and fast fractional pixel motion estimation diagram

图 2 FFPS 和快速分像素运动估计示意

全分像素运动估计算法(FFPS)是最传统的分像素运动估计算法,能够提供最高的搜索精度,同时也需要耗费最多的计算资源.图 2 是 FFPS 算法(空心圆圈部分)和快速分像素运动估计算法(实心方形部分)的示例.FFPS

首先检查分像素搜索区域中所有的 8 个 1/2 分像素搜索位置,然后检查最优位置周围的 8 个 1/4 分像素搜索位置.由于 FFPS 的搜索顺序与 H.264 定义的插值顺序恰好相同,XFPI 插值算法利用了这个特性,固定计算所有的 1/2 像素集,即 02,20,22 像素集,然后在搜索过程中动态计算与每个搜索位置所对应的 1/4 像素.

但是,快速分像素运动估计并不像FFPS算法那样采用固定的搜索顺序.如图2中的实心方块所示,快速算法总是在分像素预测运动矢量的领域内采用菱形模板细化搜索.由于这些快速算法的搜索顺序并不固定而且只检查很少一部分的搜索位置,配合 H.264 标准使用时不得不产生大量的冗余计算.然而,HEVC 采用了可分离的滤波器[2],提供了更为灵活的插值方式:与整像素同列或同行的分像素可直接算出,其余的像素可以自由选择两者之一进行计算.就算按照不同的计算顺序都可以得出相同的像素值.也正是由于这种灵活的插值方式,为我们设计优化的分像素运动估计和插值算法提供了基础.

2 性价比优先的分像素运动插值和估计

本节将分 3 部分给出我们的性价比优先算法.第 1 部分给出一个动态的性价比优先细化搜索顺序.该搜索顺序不但考虑了搜索模板上不同位置成为最优分像素运动矢量的概率,而且也顾及了不同搜索位置对应分像素的插值代价.按照该顺序搜索,可以在付出最小计算代价的同时获得最好的率失真性能.第 2 部分给出一种配合这个搜索顺序的分区域分像素集插值方法.该插值方法解决了现有插值算法无法同时降低冗余插值和重复插值的问题.第 3 部分给出了进一步优化分像素运动估计策略,其目的是为了使运动估计在保证精度的情况下提前收敛,尽量减少无用的搜索和插值计算.

2.1 性价比优先搜索顺序

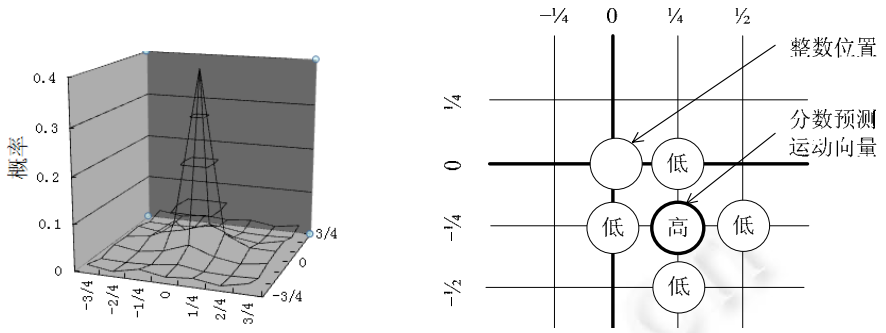
由于对分像素运动矢量的预测总会或多或少地产生误差,所以在预测阶段后进行细化搜索是必须的.这里,我们将针对不同的分像素搜索位置给出 6 个性价比优先等级.细化搜索模板中,每个搜索位置都可以映射到这 6 个等级中,并根据等级的高低顺序进行搜索.每个等级拥有不同的“性能”和“代价”.其中,“性能”指成为最优分像素运动矢量的概率,而“代价”则指所对应分像素的插值代价.现有的快速分像素运动估计算法只单纯地把整像素运动估计算法的思想扩展到分像素区域,仅按照不同搜索位置成为最优分像素运动矢量概率的大小搜索.换句话说,搜索的优先顺序仅以率失真收益的高低为标准.这样的做法忽略了分像素运动估计与整像素运动估计的一个重要区别:不同分像素搜索位置的计算代价是不同的.这种不同来源于不同分像素方法和滤波器的不同.高效的分像素运动估计算法应该优先检查率失真收益高和计算代价低的搜索位置,这就是性价比优先分像素运动估计算法的主要思路.

其中的难点主要在于:虽然分像素的插值滤波计算方法是固定的,但是在计算过程中,实际的计算消耗却是动态变化的.因为 HEVC 中使用了可分离的插值滤波器,分像素之间存在依赖关系,分像素的计算复杂程度不仅由自身的插值滤波器决定,而且还与相依赖的分像素是否被计算有关.下面我们将首先介绍在一个细化模板中不同位置所对应的“性能”高低;然后介绍不同搜索位置对应的“代价”大小;最后,通过一个例子说明如何在搜索过程中确定不同搜索位置的性价比优先等级.

要设计一种同时最大化性能和最小化代价的算法,首先需要分析不同分像素搜索位置成为最优分像素运动矢量的概率.图3(a)是我们通过比对多个不同的视频序列所得到的预测和最优运动矢量相对位移的概率分布图.从图中可以看出:最优运动矢量都集中在以预测运动矢量为中心的区域;并且多数的预测运动矢量经过细化搜索后,最终也成为了最优运动矢量.另外,如图 3(b)所示,由于细化搜索一般采用菱形模板,所以模板的中心位置拥有比周边位置更高的概率成为最优运动矢量.也就是说,搜索模板的中心比四周具有更高的率失真收益.

另一方面,每个分像素搜索位置的计算代价本质上由分像素运动估计的比较运算和相应分像素的插值运算组成.对于不同的搜索位置,比较运算的代价是相同的,但由于所对应的分像素不同,其插值的代价也不同.插值代价的大小依赖于对应分像素所采用插值滤波器的大小.图4(a)表示在一个分像素搜索区域内不同分像素位置所使用插值滤波器的大小.根据HEVC的定义,与整像素在同一行或同一列的分像素使用7/8抽头滤波器计算(图中标识为 7/8).剩余的分像素位置则在此基础上,再用一个 7/8 抽头滤波器计算(图中标识为 7/8×8/7).图 4(a)

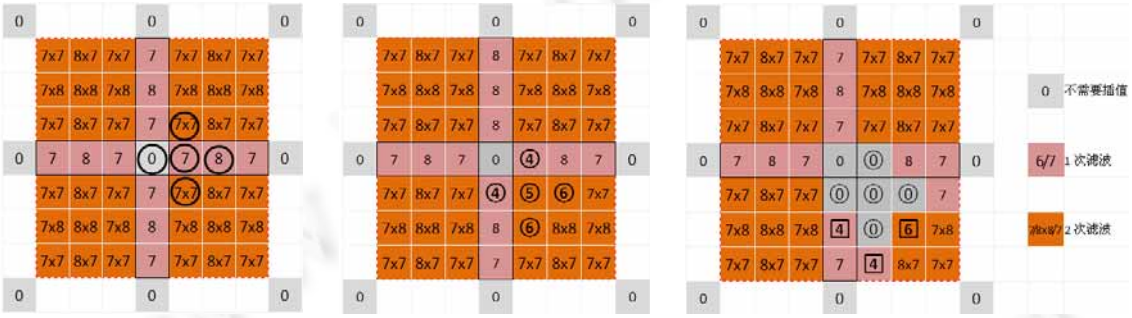
中的每个搜索位置都对应不同复杂度的插值计算:0 不需要进行插值计算,7/8 需要进行一次插值计算, 7/8×8/7 需要进行两次插值计算.



(a) 最优分像素运动矢量的概率分布 (b) 菱形模板上的概率分类

Fig.3 Probability distribution of being the best MV

图 3 最优运动矢量的概率分布



(a) 不同分像素搜索位置对应的插值滤波器 (b) 初始位置的性价比优先等级 (c) 中心移动后的优先等级

Fig.4 An example of fractional pixel search pattern

图 4 分像素搜索模板示例

特别需要注意的是:由于 HEVC 中定义的插值计算是相互关联的,所以分像素位置的插值复杂度会动态变化.例如,如果图 4(a)中菱形模板中心位置已经插值,那么与之同一列的所有分像素位置只需进行一次插值计算即可得出,其计算复杂度就相当于与整像素在同一行或同一列的分像素了.

综上所述,图 3 表示不同搜素位置所对应的率失真收益(分别为高或低),图 4 表示不同搜索位置所对应的计算代价(分别为 0,1 次滤波或 2 次滤波).表 1 结合了这两者的特性,给出了 6 种不同的性价比优先等级.高的等级表示有更高的概率成为最优分像素运动矢量,同时付出的插值代价也更小.这些等级的确定原则如下:在相同性能的情况下,插值代价低的位置优先搜索;相同插值代价的情况下,性能高的位置优先搜索;在插值代价和性能各有优劣的情况下,由于插值计算远比搜索计算复杂,所以插值代价低的位置优先于性能高的位置搜索.细化搜索模板中的每个搜索位置都会映射到这个 6 个等级中,然后按照等级高低的顺序搜索.

图 4 还用一个具体的例子说明本文所提出的性价比优先搜索顺序.图中深浅不同的格子表示不同插值复杂度的搜索位置.图 4(b)中,菱形模板的中心位置表示预测运动矢量的位置,圆圈中的序号表示优先等级.根据表 1 的定义,我们对菱形搜索模板的所有搜索位置都标注了性价比优先等级.菱形模板的搜索不是按照传统的先中心后四周的顺序,而是按照等级的高低进行,即先搜索标为 4 的位置,再搜索标为 5 的位置,最后搜索标为 6 的位置.如图 4(c)所示,当模板中心需要往下移动时,不同搜索位置的插值代价也发生了相应的变化.图中圆形(并

被标为 0)的位置表示已经被搜索过的位置,正方形表示将要搜索的位置,正方形中的序号表示优先等级.由于部分分像素位置已经在前面的搜索中被计算出来,所以正方形搜索位置的插值代价也发生了变化.其中,有两个位置都只需要一次 6 抽头的插值滤波,其优先等级都为 4;剩下的那个位置仍需要两次 6 抽头的插值滤波,其优先等级为 6.算法先搜索标为 4 的两个位置,然后再搜索标为 6 的位置.搜索过程的每一步都优先选择等级最高的位置搜索,同时,动态更新不同搜索位置的插值代价以便得到各搜索位置当前的等级,保证了搜索按照性价比优先的顺序进行.

Table 1 Priority ranking for different search positions

表 1 不同搜索位置对应的性价比优先等级

率失真性能	插值代价		
	0 次滤波	1 次滤波	2 次滤波
高	1	3	5
低	2	4	6

2.2 分区域分像素集的插值方法

前面我们把不同分像素的计算代价细分到了不同的搜索位置上,并给出了一个优化的搜索顺序.要配合这样的搜索,就必须构造相应的分像素插值算法.目前,全分像素插值算法由于要一次性预先计算所有的分像素,所以产生了大量的冗余计算.而边搜索边插值的算法由于没有考虑分像素之间计算的关联性,而导致大量的重复计算.分像素插值算法中,最大的难点是如何平衡冗余计算和重复计算的消耗.针对该难点,本节提出了一种分区域分像素集插值算法,同时给出以下 3 个方面的细节:分像素的存储和计算策略;多分块模式运动补偿下插值方法的调整;单指令多数据流的多媒体指令集(SIMD)下的优化.

首先,这里给出了分区域分像素集插值方法的存储和计算策略.为了在冗余计算和重复计算之间获得一个最优的平衡,我们按照分区域的方式存储分像素,而在一个区域中的分像素则根据实际需要计算相应的像素集.这里所说的区域是指在最优整像素运动矢量周围(上、下、左、右),一个像素的搜索区域.因为分像素运动估计只在最优整像素运动矢量一个像素范围内搜索,所以分像素运动估计所使用到的像素仅在这一范围内.例如,在编码单元大小为 64×64 模式下进行分像素运动估计,分像素插值区域就是一个(64+2)×(64+2)的区域,即,向上、下、左、右各扩展一个像素.

当分像素插值区域确定后,每个区域中的分像素按照图 1(a)的定义划分成 15 个像素集.不同的像素集按实际需要计算,并以像素集为最小单位保存,直到当前块的运动补偿结束.由于 HEVC 中的分像素插值相互关联,所以不同像素集的计算并非独立,而是存在依赖关系.为此,本文为每个区域引入 15 个标志位,标识不同的像素集像素是否已被计算,并通过对标志位判断来减少分像素的重复计算.算法将在第 2.3 节的算法 1 中给出.

其次,如果视频编码中采用了多分块模式运动补偿技术,对每种模式单独实施上述算法则会导入严重的重复计算[10,11].为了避免因模式重叠而引发的重复插值,在多模式的情况下,我们采用扩充区域的方法来解决该问题.本文对同一个 CU 中的不同模式统一采用一个矩形区域进行像素集的计算.如前所说,分像素插值区域的位置和大小依赖于最优整像素运动矢量和相应分块的大小.为了确定该矩形区域的位置和大小,不同模式下所有分块的整像素运动必须预先完成,而所有分像素运动估计统一在其后进行.插值区域就是包括所有最优整像素候选块的最小矩形区域.在该区域内的分像素,会以像素集为最小单位来计算.

最后,由于同一个像素集中的分像素使用相同的插值滤波器,所以本文的插值方法可以利用单指令多数据流技术(SIMD)并行计算.多个分像素可以在同一个 CPU 运算周期内访问和处理.单指令多数据流的技术对于加快分像素插值存在两点优势:首先,分像素以矩形块的方式存放,多个像素可被同时访问,而不是一个个像素单独地处理;此外,单指令多数据流技术通常在一条指令里就并行地计算多个数据.换而言之,如果 CPU 同时装入 16 个像素,那么 CPU 进行的加、减、乘、除等操作就会同时作用在这 16 个数据上.正是因为我们按照分区域分像素集的方式计算分像素,所以该方法非常适合采用单指令多数据流技术优化.

2.3 分像素运动估计优化策略和总体算法

由于分像素的插值必须在搜索前完成,所以减少分像素搜索位置不仅可以提升分像素运动估计的速度,而且可以减少多余的分像素插值.本节将提出 3 种减少搜索位置的策略,同时给出整体算法的流程.

第一,我们利用时域相邻静态分块的相关性以决定是否跳过当前块的分像素插值和运动估计.对于静止场景中的分块,最好的编码手段是跳过所有的分像素插值和运动估计.现有的提前终止条件[12−14]利用了相邻分块最优残差值的相关性,但实际上,相邻分块最优残差值之间的变化非常不稳定[9],特别是在视频包含剧烈运动的情况,相邻分块的最优残差值相差甚远.为此,难以为各种情况设定一个统一提前终止条件.然而,如果该分块的内容是静止的,最优残差值之间的时域相关性就会相当高.直观上看,静止视频序列中的残差值,其变化也相对平稳.

我们针对当前块是否为一个静止分块给出如下 3 个条件:当前块的最优整像素运动矢量为(0,0);当前分块的时域相邻分块的运动矢量为(0,0);两个分块的最优整像素残差值足够接近.同时具备这 3 个条件的分块就是一个静止分块,并且无须在该分块上应用分像素运动估计和插值.对于第 1 和第 2 个条件,可以通过检查相应分块的运动矢量判断.这里特别针对第 2 个条件指出:如果时域相邻分块为帧内预测分块,则当前分块不可能成为静止分块.对于第 3 个条件,判断相邻两个分块残差值是否足够接近,本文使用阈值 $Th1$ 比较两者之间的大小.由上可知,该策略的关键是 $Th1$ 的优化设置.由于不同情况下,$Th1$ 的优化值相差很大,我们给出一种自适应算法来计算 $Th1$.本文引入一个与 $Th1$ 相关的比率,即漏判数与成功数比率.

$$Ratio_{threshold}=\frac{MissingNumber}{HittingNumber}\times 100(\%) \tag{1}$$

其中,$MissingNumber$ 是指因为采用当前阈值而漏掉跳过的那部分搜索位置,而 $HittingNumber$ 则指采用当前阈值成功跳过的那部分搜索位置.因为在没有实际检测搜索位置的情况下不可能准确地知道 $HittingNumber$ 的大小,所以本文直接使用阈值所跳过的那部分搜索位置数作为 $HittingNumber$.在编码过程中,算法动态地量度与 $Th1$ 对应的 $MissingNumber$ 和 $HittingNumber$,并通过经验值 $R1$ 决定当前阈值 $Ratio_{threshold}$ 是否恰当,从而调整 $Th1$.例如,当 $Th1$ 的 $Ratio_{threshold}$ 大于 $R1$ 时,则表明当前的 $Th1$ 太大了,以至于不能在静态场景下跳过无用的搜索位置,这时候应该降低 $Th1$;反之亦然.

第二,我们还给出一种搜索模板裁剪策略减少细化搜索的计算复杂度.该裁剪策略的思想源自于分像素残差曲面是单峰曲面这一假设.该假设被现有的许多分像素运动估计算法[5,6]所使用.目前,多数的分像素运动估计算法都使用中心对称的搜索模板进行细化搜索.基于上述假设,在一个中心对称模板相反两个方向上的搜索位置不可能同时拥有比中心残差值更低的残差.为此,当某个搜索位置的残差值比细化模板中心位置残差值要小的时候,细化搜索应该跳过与该位置中心对称位置的搜索.

分像素运动矢量搜索区域的残差曲面可被认为是一个拥有最小值的单峰曲面,也就是说,分像素运动估计的局部最优位置同时也是全局的最优位置.为了分析分像素残差曲面,本文把单峰曲面简化成一维的情况讨论.图 5 是一维搜索的一个例子.单峰曲线 $f(x)$中的横坐标 x 表示位置,纵坐标表示相应位置上的残差值.其中,x_0 表示分像素区域的最优位置.

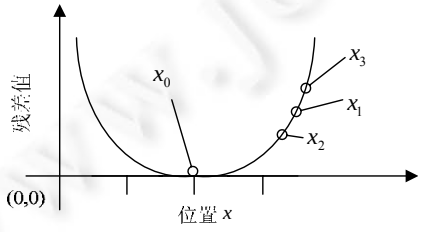


Fig.5 One dimentional residual curve

图 5 一维的残差曲线

根据单峰曲线的特性有:

$$\begin{cases} f'(x)<0,\ x<x_0 \\ f'(x)=0,\ x=x_0 \\ f'(x)>0,\ x>x_0 \end{cases} \tag{2}$$

使用菱形模板搜索,在一维的情况下就相当于搜索当前位置的左右两个搜索点.其中,x_1 是当前搜索位置,x_2 和 x_3 分别是 x_1 左右的两个相邻位置.这 3 点可以出现在曲线的任意位置上,只要保持它们之间的位置的相对性. x_2 和 x_3 的导数只可能是以下 3 种情况之一:

$$\begin{cases} f'(x_2)<0, f'(x_3)<0,\ x_2<x_0, x_3<x_0 \\ f'(x_2)<0, f'(x_3)>0,\ x_2<x_0, x_3>x_0 \\ f'(x_2)>0, f'(x_3)>0,\ x_2>x_0, x_3>x_0 \end{cases} \tag{3}$$

而不可能出现的情况是:

$$f'(x_2)>0,\ f'(x_3)<0 \tag{4}$$

这是因为 x_2 在 x_1 的左侧,而 x_3 在 x_1 的右侧.公式(4)说明,x_2 和 x_3 两个位置的残差值不可能同时比 x_1 小.由此,推广到二维的情况,则有以下结论:两个关于中心对称的搜索点的残差值不可能同时比中心点的残差值小.根据该结论,当某个分像素位置的残差值比模板中心的残差值要小时,就不需要再搜索与它相反方向的搜索位置了.

上述的这些策略不但进一步地加快了分像素运动估计,而且由于配合了动态插值方法,也大幅度减少了无效的插值计算.综合以上所提到的方法,下面给出本文算法的总体步骤.

算法 **1**. 分区域分像素集插值算法.

输入:需要计算的分像素集(00~33).

输出:需要计算像素集中的分像素.

步骤 1. 检查当前需要计算像素集的标志位.If 如果该像素集已经被计算,Then 返回.

步骤 2. If 需要计算像素集的前驱已经存在(参考图 1(b))

Then

直接通过这些前驱计算该像素集,并设置该像素集的标志位和返回.

Else

否则,递归地调用本算法计算前驱像素集.

步骤 3. 通过前驱像素集计算当前所需的像素集并设置标志位返回.

算法 **2**. 总体算法.

输入:当前 CU.

输出:CU 中各个分块的最优分像素运动矢量.

步骤 1. 分像素运动估计初始化.当前 CU 的 15 个像素集的标志位全部设为 0.

步骤 2. If 当前 CU 满足静止分块条件 Then 跳过当前分块的分像素运动估计和分像素插值.

步骤 3. 使用我们的前期工作[4]预测起始的分像素运动矢量,也就是细化搜索的起始中心位置.

步骤 4. 细化搜索:

While 当前菱形模板未搜素过 do

While 当前菱形模板剩余搜索位置>0 do

步骤 4.1. 在匹配每个搜索位置前,调用算法 1 计算相应的像素集.

步骤 4.2. 检查当前搜索位置.

步骤 4.3. IF 如果满足模板裁剪策略 Then 相反方向的搜索位置被跳过.

End While

End While

算法 1 所说的前驱像素集是指在计算当前像素集之前需要优先计算的那部分像素集.如图 1(b)所示,由于

HEVC 中非正交分像素的计算要通过与整像素同行或同列(正交)分像素算出,所以 11,12,13,21,22,23,31,32,33 像素集的前驱像素集是 01,02,03 或者 10,20,30.

3 实验结果

为了评估本算法的性能,本节给出了总体性能的比较.实验结果表明,本文所提出的整合算法能够在不损失率失真性能的情况下,大幅度地提高分像素运动估计和插值的计算速度.算法实现在 HEVC 的参考模型 HM10.0 中,每个序列仅第 1 帧使用 I 帧,其余都是 *P* 帧.量化参数分别是 22,27,32 和 37.针对静止分块和分像素搜索是否提前终止的判断,我们根据经验值把 *R*1 设置为 0.8.实验在表 2 的 18 个不同分辨率和不同运动情况的序列上完成.这些序列按照分辨率的不同被划分成 *A*~*E* 共 5 个组.

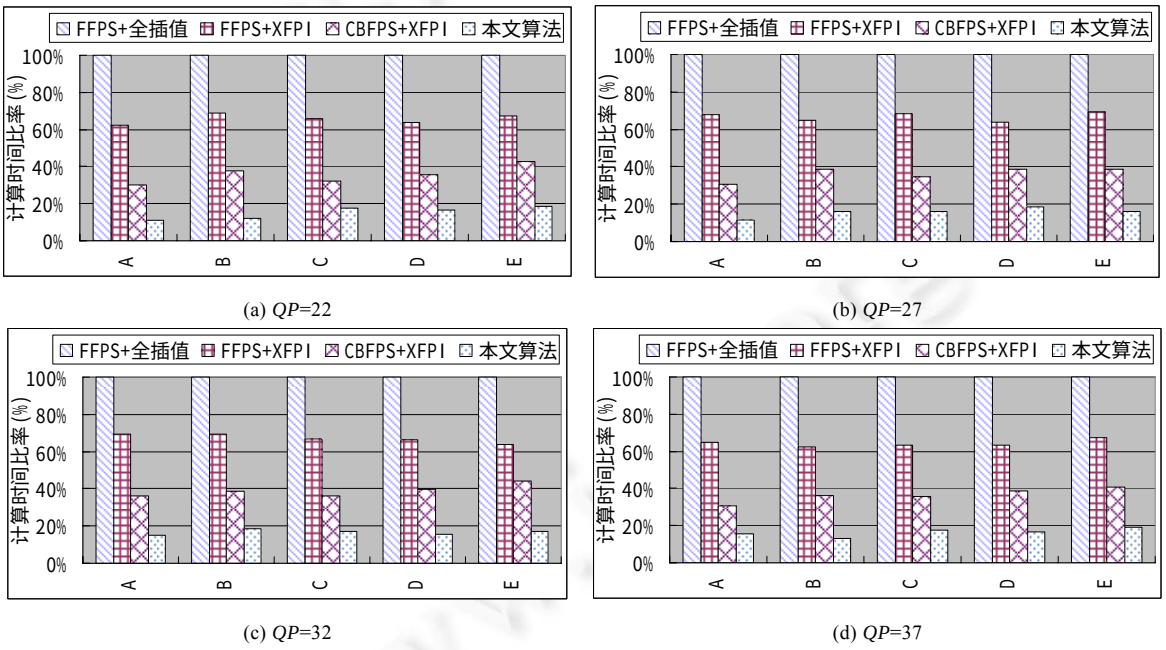
Table 2 Different groups of sequences for testing

表 2 测试序列的分组情况

序列集	格式	序列
A	2560×1600	PeopleOnStreet, Traffic
B	1920×1080	BasketballDrive, BQTerrace, Cactus, Kimono1, ParkScene
C	1280×720	vidyo1, vidyo3, vidyo4
D	832×480	BasketballDrill, BQMall, PartyScene, RaceHorses
E	416×240	BasketballPass, BlowingBubbles, BQSquare, RaceHorses

本算法的计算消耗由分像素运动估计和分像素插值两部分构成.其中,分像素运动估计的计算复杂度与搜索位置的多少相关,分像素插值的计算复杂度与插值像素的数目及其对应插值滤波器的大小相关.由于两者的计算复杂度难以统一测量,所以本文采用计算时间比率来量度各算法的计算消耗.

图 6 给出了 4 种算法——FFPS+全插值、FFPS+XFPI、CBFPS+XFPI 与本文算法的总体性能比较.



(a) *QP*=22 (b) *QP*=27

(c) *QP*=32 (d) *QP*=37

Fig.6 Computational performance of different algorithms

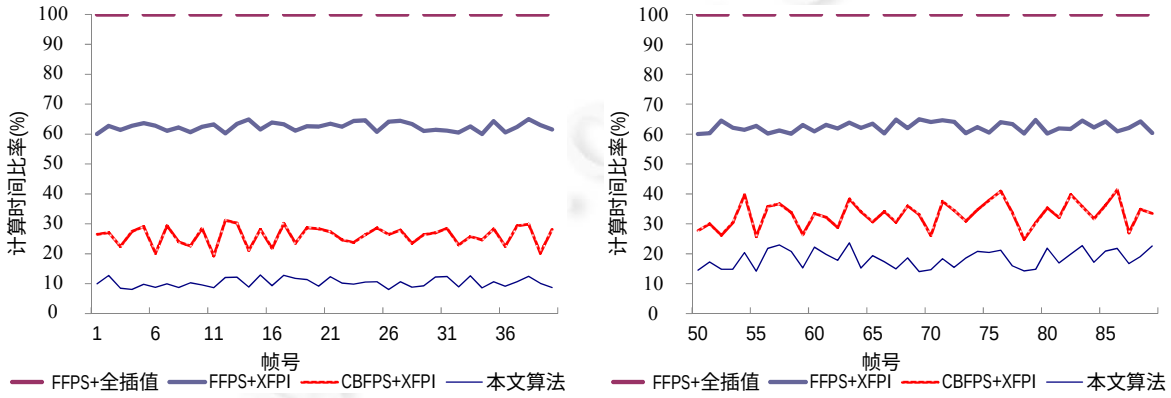
图 6 不同算法计算性能的比较

本文以 FFPS+全插值算法的计算复杂度作为基准,即 100%.与其他几种算法组合相比,本算法在速度上分别提高了 84%、71%和 52%.

从图 6 可以看出,视频分辨率越高本文算法性能越好.其主要原因在于:分辨率越高,其频率混淆程度就越低,最优分像素运动矢量也就越趋集中于整像素运动矢量.为此,分像素运动估计需要搜索的范围就越小.本文的算法可以直接计算这些分像素,而不需要计算多余的分像素,非常适合在这种情况下使用.

影响算法性能的另一个重要因素是量化参数.如图 6 所示,量化参数越大,本文算法的性能越好.这是由于当量化参数增大、视频质量下降时,分像素运动估计提前终止在次优位置将产生与最优位置相同的率失真效果.在这种情况下,我们的算法既可以提前终止分像素运动估计,也可以减少对相应分像素的插值计算.

此外,视频内容的运动剧烈程度也会影响算法的性能.图 7 是不同算法在 BasketballPass 序列上的逐帧计算时间比较.当 BasketballPass 序列开始时,场景内部的运动不多,而且镜头也是固定的,所以前 40 帧为静止场景,计算性能如图 7(a)所示;BasketballPass 序列在 40 帧之后,不但内部运动大量增加,而且镜头也同时大幅度地移动,所以我们将其定义为运动场景,计算性能如图 7(b)所示.通过对比图 7(a)和图 7(b),本文算法应用在静止场景上的性能明显优于应用在运动场景上的性能.这些额外的计算收益主要来源于本文提出的静态场景跳过策略.



(a) 静止场景 (b) 运动场景

Fig.7 Frame by frame computational performance on BasketballPass sequence

图 7 BasketballPass 序列上的逐帧计算性能

另外,表 3 给出了上述测试条件下的率失真性能比较.这里需要特别说明的是:分像素插值算法必须是无损算法,产生与 HEVC 标准定义一致的分像素,否则将引发编解码端参考帧不一致的 Drifting 问题.因此,率失真性能的差别主要来自于不同的分像素运动估计算法.为此,表 3 中 FFPS+XFPI 算法和 FFPS+全插值算法的率失真性能是相同的.

为了方便展示不同算法率失真性能的比较结果,本文在表 3 中给出了本算法、CBFPS+XFPI 算法和 FFPS+XFPI 算法相对于 FFPS+全插值算法在码率和视频重建质量上的相对性能.该相对性能的计算方式如下:

$$\begin{cases}\Delta PSNR = PSNR_x - PSNR_{FFPS} \\ \Delta Bit\ rate = (Bit\ rate_x - Bit\ rate_{FFPS}) / Bit\ rate_{FFPS}\end{cases} \tag{5}$$

其中,$PSNR_{FFPS}$ 和 $Bit\ rate_{FFPS}$ 分别是 FFPS 运动估计算法的信噪比和码率,$PSNR_x$ 和 $Bit\ rate_x$ 是其他 3 种算法的信噪比和码率.表 3 中的结果表明,本文算法具有与 CBFPS+XFPI 算法相同的率失真性能.与作为基准算法的 FFPS+全插值算法相比,本文的算法平均性能仅仅在信噪比上下降了 0.01dB 以及在码率上增加了 0.38%.

Table 3 Rate distortion performance of different algorithms

表 3 不同算法的率失真性能

量化参数(QP)	序列集	ΔPSNR (dB)			ΔBit rate (%)		
		FFPS+XFPI	CBFPS+XFPI	本文算法	FFPS+XFPI	CBFPS+XFPI	本文算法
22	*A*	0.00	0.00	0.00	0.00	0.60	0.30
	B	0.00	–0.02	–0.01	0.00	0.40	0.30
	C	0.00	–0.02	–0.01	0.00	0.20	0.20
	D	0.00	–0.01	0.00	0.00	0.00	0.70
	E	0.00	–0.02	–0.02	0.00	0.41	0.31
27	*A*	0.00	–0.02	0.00	0.00	0.37	0.42
	B	0.00	–0.01	0.00	0.00	0.12	0.33
	C	0.00	–0.02	–0.02	0.00	0.63	0.33
	D	0.00	0.00	0.00	0.00	0.40	0.70
	E	0.00	–0.01	–0.01	0.00	1.19	0.98
32	*A*	0.00	–0.01	0.00	0.00	0.78	0.23
	B	0.00	–0.02	–0.01	0.00	0.93	0.38
	C	0.00	–0.01	–0.01	0.00	0.11	0.31
	D	0.00	0.00	0.00	0.00	0.13	0.42
	E	0.00	0.00	0.00	0.00	0.36	0.56
37	*A*	0.00	–0.01	0.00	0.00	0.41	0.31
	B	0.00	0.00	–0.01	0.00	0.51	0.13
	C	0.00	0.00	0.00	0.00	0.00	0.11
	D	0.00	0.00	–0.01	0.00	0.40	0.41
	E	0.00	–0.01	0.00	0.00	0.31	0.21

4 结束语

我们通过分析分像素运动估计搜索和分像素插值之间的关系,把分像素插值的计算消耗投射到运动估计过程的每步中,并建立了一种符合性价比优先原则的搜索顺序.同时,为了配合该搜索顺序,我们设计了一种分区域分像素集的插值算法.该算法能够最大限度地同时减少分像素插值中的重复计算和冗余计算.根据实验分析,我们的算法不仅提高了分像素运动估计的计算速度,而且大幅度地避免了对冗余分像素的计算.而且,算法只占用固定的内存,也相应地节省了大量的存储访问时间.

References:

[1] Lu JY, Zhang PZ, Chao HY, Paul F. An integrated algorithm for fractional pixel interpolation and motion estimation of H.264. In: Proc. of the Data Compression Conf. (DCC). 2010. 541–541. [doi: 10.1109/DCC.2010.101]

[2] Sullivan GJ, Ohm J, Han WJ, Wiegand T. Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. on Circuits and Systems for Video Technology, 2012,22(12):1649–1668. [doi: 10.1109/TCSVT.2012.2221191]

[3] Chen ZB, Xu JF, He Y, Zheng JL. Fast integer-pel and fractional-pel motion estimation for H.264/AVC. Journal of Visual Communication and Image Representation, 2006,17(2):264–290. [doi: 10.1016/j.jvcir.2004.12.002]

[4] Chao HY, Lu JY. A high accurate predictor based fractional pixel search for H.264. In: Proc. of the 2006 IEEE Int'l Conf. on Image Processing. 2006. 2365–2368. [doi: 10.1109/ICIP.2006.312901]

[5] Chang JF, Leou JJ. A quadratic prediction based fractional-pixel motion estimation algorithm for H.264. Journal of Visual Communication and Image Representation, 2006,17(5):1074–1089. [doi: 10.1016/j.jvcir.2006.01.001]

[6] Paul H, Chiew TK, David B, Nishan C. Interpolation free subpixel accuracy motion estimation. IEEE Trans. on Circuits and Systems for Video Technology, 2006,16(12):1519–1526. [doi: 10.1109/TCSVT.2006.885722]

[7] Dikbas S, Arici T, Altunbasak Y. Fast motion estimation with interpolation-free sub-sample accuracy. IEEE Trans. on Circuits and Systems for Video Technology, 2010,20(7):1047–1051. [doi: 10.1109/TCSVT.2010.2051283]

[8] Koohyar M, Nguyen T. Reciprocal subpixel motion estimation: Video coding with limited hardware resources. IEEE Trans. on Circuits and Systems for Video Technology, 2007,17(6):707–718. [doi: 10.1109/TCSVT.2007.896647]

[9] Lu JY, Zhang PZ, Chao HY, Fisher PS. On combining fractional-pixel interpolation and motion estimation: A cost-effective approach. IEEE Trans. on Circuits and Systems for Video Technology, 2011,21(6):717−728. [doi: 10.1109/TCSVT.2011.2129830]

[10] Lu JY, Hou F, Huang CH, Liu YX, Chao HY. Multi-Mode decision under computational resource constraints for video coding. Ruan Jian Xue Bao/Journal of Software, 2014,25(11):2690−2701 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4695.htm [doi: 10.13328/j.cnki.jos.004695]

[11] Lu JY, Chao HY, Huang CH, Hou F. Rate distortion optimization of complexity scalable motion estimation. Chinese Journal of Electronics, 2014,42(8):1495−1502 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2014.08.006]

[12] Luo JC, Ishfaq A, Liang YF, Viswanathan S. Motion estimation for content adaptive video compression. IEEE Trans. on Circuits and Systems for Video Technology, 2008,18(7):900−909. [doi: 10.1109/TCSVT.2008.923423]

[13] Kim JH, Kim BG. Fast block mode decision algorithm in H.264/AVC video coding. Journal of Visual Communication and Image Representation, 2008,19(3):175−183. [doi: 10.1016/j.jvcir.2007.09.001]

[14] Xie Z, Liu Y, Liu J, Yang T. A general method for detecting all-zero blocks prior to DCT and quantization. IEEE Trans. on Circuits and Systems for Video Technology, 2007,17(2):237−241. [doi: 10.1109/TCSVT.2006.888812]

附中文参考文献:

[10] 陆寄远,侯昉,黄承慧,刘宇熹,朝红阳.计算资源受限的视频编码多模式决策.软件学报,2014,25(11):2690−2701. http://www.jos.org.cn/1000-9825/4695.htm [doi: 10.13328/j.cnki.jos.004695]

[11] 陆寄远,朝红阳,黄承慧,侯昉.计算能力可伸缩的运动估计率失真优化.电子学报,2014,42(8):1495−1502. [doi: 10.3969/j.issn.0372-2112.2014.08.006]

陆寄远(1976－),男,广东南海人,博士,教授,CCF 学生会员,主要研究领域为视频编码,图像处理.

黄承慧(1976－),男,博士,系统分析师,CCF 专业会员,主要研究领域为视频信息检索,多媒体数据挖掘.

刘宇熹(1971－),男,博士,高级工程师,主要研究领域为视频信息数据分析.

朝红阳(1957－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为视频编码,图像压缩.

侯昉(1975－),男,博士,讲师,主要研究领域为视频存储.