计算机科学
COMPUTER SCIENCE

基于统计分析的仿射运动估计快速算法

钟煜城, 黄晓峰, 牛伟宏, 崔燕

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230400081

基于统计分析的仿射运动估计快速算法

钟煜城[1]　黄晓峰[1]　牛伟宏[1]　崔　燕[2]
1 杭州电子科技大学通信工程学院　杭州 310018
2 浙江省经济信息中心　杭州 310007
(yuchengzhong327@163.com)

摘　要　为降低新一代通用视频编码标准(Versatile Video Coding,VVC)的计算复杂度,提出了一种基于统计分析的仿射运动估计(Affine Motion Estimation,AME)快速算法。从加速 AME 过程的角度出发,首先摒弃 AME 的 3 种运动矢量(Motion Vector,MV)精度中的整像素和 1/16 像素精度,保留 1/4 像素精度;其次利用迭代次数与量化参数(Quantization Parameter,QP)、slice 类型以及编码单元(Coding Unit,CU)大小的关系,得到一个迭代次数的自适应计算式来减少 AME 迭代次数;然后将细粒度搜索(Fine Granularity Search,FGS)算法中 CU 4 个角落的 4 个整像素用 2 个对角分像素进行替代;最后运用绝对变换差和(Sum of Absolute Transform Difference,SATD)代价来替代率失真(Rate Distortion Optimization,RDO)代价。实验结果表明,与 H.266/VVC 参考软件 VTM-10.0 相比,提出的算法在低延迟(Low Delay B,LDB)和随机访问(Random Access,RA)配置下分别节省了 8.34%和 8.83%的时间,与此同时性能损失仅为 0.10%和 0.12%。

关键词:通用视频编码;仿射运动估计;像素精度;细粒度搜索;绝对变换差和
中图分类号　TN919

Fast Algorithm for Affine Motion Estimation Based on Statistical Analysis

ZHONG Yucheng[1],HUANG Xiaofeng[1],NIU Weihong[1] and CUI Yan[2]
1 School of Communication Engineering,Hangzhou Dianzi University,Hangzhou 310018,China
2 Zhejiang Economic Information Center,Hangzhou 310007,China

Abstract　To reduce the computational complexity of the new generation video coding standard-versatile video coding(VVC),a fast affine motion estimation(AME) calculation method based on statistical analysis is proposed. In the proposed method,we first abandon the integer pixel and 1/16-pixel accuracy,while retaining 1/4-pixel accuracy of the three motion vector(MV) accuracies. Secondly,we build the relationship between the iterations and quantization parameters(QP),slice type,and coding unit(CU) size to obtain an adaptive formula for reducing the number of iterations in AME. Then,the four integer pixels in the four corners of CU in the fine granularity search(FGS) algorithm are replaced by two diagonal sub pixels. Finally,the sum of absolute transform difference(SATD) cost is used to replace the rate distortion optimization(RDO) cost. Experimental results show that compared with the H.266/VVC reference software VTM-10.0,the proposed algorithm saves 8.34% and 8.83% of time in low delay B (LDB) and random access(RA) configurations,while the performance loss is only 0.10% and 0.12%,respectively.

Keywords　Versatile video coding,Affine motion estimation,Pixel accuracy,Fine granularity search,Sum of absolute transform difference

1　引言

视频编码领域飞速发展,从 20 世纪 90 年代至今,视频编码标准已经基于初代的 H.261 衍生出了许多新标准。20 世纪初,为了支持更高的视频分辨率和解决上一代高级视频编码(Advanced Video Coding,AVC)[1]存在的一些问题,国际组织提出了高效视频编码[2],HEVC 与 AVC 相比节约了近 50%的比特率。如今,国际组织联合视频探索小组(Joint Video Experts Team,JVET)推出了最新的通用视频编码(Versatile Video Coding,VVC)[3],这种最新的视频编码标准相比上一代的 HEVC,压缩效率又可以提升近 50%,这对于存储 4K,8K 乃至更高分辨率的视频提供了巨大便利。

如此高的压缩率不容易得到,其带来了极高的计算复杂度。在全帧内配置(All-intra Configuration)的情况下,VVC 测试模型(VTM)的计算复杂度是 HEVC 测试模型(HM)的 18 倍[4]。

在块划分部分,HEVC 中的编码树单元(Coding Tree Unit,CTU)的大小可以设置为 64×64,而 VVC 中的 CTU 大小提高为 128×128。将允许的最大 CTU 大小从 64×64 扩展到 128×128 虽然有助于为更高空间分辨率的视频实现更好的性能,但是会导致计算成本显著增加[5]。

在帧内预测部分,VVC 将帧内预测角度从 35 个扩展到

基金项目:国家电网有限公司总部管理科技项目(5700-202325308A-1-1-ZN)
This work was supported by the Technology Project Managed by State Grid Corporation of China Headquarter(5700-202325308A-1-1-ZN).
通信作者:崔燕(cuiyan_9816@126.com)

了 67 个，因此 VVC 中的帧内预测技术除了包括与 HEVC 类似的 DC 和 Planar 模式，还有与 HEVC 相比角度更多的细粒度角度预测(Finer-granularity Angular Prediction，FGAP)、广角度预测(Wide-angle Intra Prediction，WAIP)以及基于矩阵的帧内图像预测(Matrix-based Intra-picture Prediction，MIP)等模式，它们在获得更强性能的同时复杂度也急剧增高。

在帧间预测部分，现实世界中运动的物体并非都是刚体运动，很多运动都存在旋转、缩放等非刚体运动，为了能够更加清晰有效地描述这种非刚体运动，VVC 引入了全新的仿射运动估计[6]，而且引入自适应运动矢量精度(Adaptive Motion Vector Resolution，AMVR)使得最大分像素精度达到了 1/16。

在变换与量化部分，VVC 通过多变换核选择(Multiple Transform Selection，MTS)[7]为从 4×4 到 32×32 的所有方形和非方形亮度块大小定义了 4 个额外的独立 DST type-VII 和 DCT type-VIII 整数内核的水平/垂直组合；通过子块变换(Sub-Block Transform，SBT)模式可以选择要编码的帧间预测 CU 剩余块的残差子分区并跳过剩余部分，其中编码的残差子分区可以是 CU 大小的一半或四分之一，但编码的一半或四分之一子分区可以是左、右、上或下部分，从而导致每个 CU 总共有 8 个模式[8]。

VVC 的块划分、帧内预测、帧间预测和变换量化的计算十分复杂。为了降低计算复杂度，文献[9]提出了一种基于深度学习的快速四叉树嵌套多类型树(Quad-tree nested Multi-type tree，QTMT)方法，用 CNN 来预测编码单元映射，并将结果反馈给 VVC 编码器来提前终止块分区过程。文献[10]通过使用两个具有计算特征的额外树模型来简化块划分编码过程，两个模型分别用于确定是否提前终止分区、最佳分区方向和选择更好的分区模式。文献[11]发现 MTS 中的 DCT type-II，DST type-VII 和 DCT type-VIII 变换内核分别适用于平坦、逐渐增加和逐渐减小的残差分布，因此通过算法来选择最可能的变换内核，以缩短 MTS 的 RDO 时间。文献[12]发现 VVC 中 I 帧的编码块分区存在一些冗余分区，因此设计了一种冗余跳分区(Redundant Partition Skipping，RPS)方案来跳过 VVC 中不必要的定向分区(Direction Partition Skipping，DPS)。此外，文献[13]提出了一种频率和空间 QP 自适应机制(Frequency and Spatial QP-adaptive Mechanism，FSQAM)，以帮助 CNN 滤波器实现可测量的更好的 BD 率性能。

除此之外，为了降低 AME 的计算复杂度，文献[14]分析了 SKIP 模式的特征，提前终止了父 CU 的 AME 过程，减少了当前 CU 中的参考系数量，从而跳过了冗余的 AME 过程。文献[15]使用纹理梯度对块分区进行分层，并让 AME 过程中的四/六参数模型使用特定的像素精度。文献[13-14]中没有提到相邻块，因此文献[16]通过检查上个 CU 的最佳帧间模式来预测当前 CU 的最佳模式，如果上个 CU 的最佳帧间模式是 AME，那么当前 CU 的 AME 类型与上个 CU 相同。此外，文献[17]提出了一种基于监督机器学习的 AME 早期搜索终止方法，该方法使用有监督的机器学习来预测基于 QP、帧速率、QT 深度、MTT 深度等一些特征的最佳输入模式，以准确预测是否应执行、部分执行或跳过 AME 步骤，避免不必要的计算。

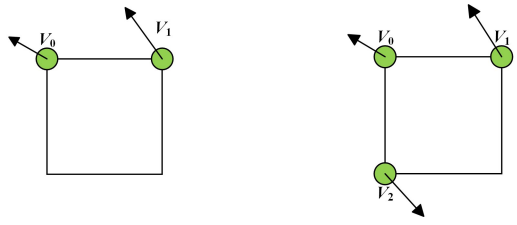
由于上述 AME 加速过程很少针对 RDO 进行优化，且 AME 在整个运动估计(Motion Estimation，ME)过程中占据大量时间，文献[14]分析了一些序列中单向预测、双向预测和仿射预测分别的时间占比，其中 AME 占用了平均 54.75%的复杂度。因此，本文主要从 4 个方面来降低 AME 中 RDO 决策的计算复杂度。首先统计了 AME 中 3 种矢量精度的占比，舍弃了两种占比较小的精度，以减少 RDO 代价的计算次数；然后分析了迭代次数与 QP、slice 类型以及 CU 大小的关系，得到了迭代次数的自适应计算式，以减少 RDO 代价的计算次数；接着将细粒度搜索算法中 CU 4 个角落的 4 个整像素用 2 个对角分像素进行替代，从而减少确定最优 MV 时所需比较各个分像素 RDO 成本的次数；最后用 SATD 决策来替代 RDO 决策，无需在变换后进行反量化、反变换等步骤，从而降低使用 RDO 决策时的计算复杂度。

2 算法原理

为了利用视频中存在的时间冗余，当前基于块的混合视频编码器均实现了帧间预测，它主要由两个步骤组成：为运动估计(Motion Estimation，ME)和运动补偿(Motion Compensation，MC)。ME 的结果是一个 MV，其用于 MC 过程来预测和重建 CU。以前标准(如 HEVC)的 ME 存在局限性，因为其只能用一个控制点(Control Point，CP)来执行，即其只能用于表示块之间的平移运动。

2.1 仿射运动模型

VVC 在帧间预测模式中新引入了 AME。AME 将用于运动估计的 CP 数量扩展到了 2 个(即四参数)和 3 个(即六参数)，如图 1 所示。通过 CP 数量的扩展，可以检测和映射其他类型的运动，如旋转和缩放。



(a)四参数仿射模型　　(b)六参数仿射模型

图 1 基于控制点的仿射运动模型

Fig. 1 Affine motion model based on control points

基于子块的仿射运动补偿(Affine Motion Compensation，AMC)即基于各控制点的控制矢量(Control Point Motion Vector，CPMV)来计算每个子块的 MV，然后通过这个 MV 进行运动补偿得到预测块。

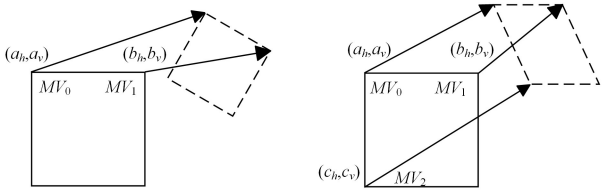
对于中心像素是(x,y)的块以及如图 2(a)所示的旋转、伸缩等运动，可通过四参数模型得到每个子块 CPMV，具体计算式如下：

$$\begin{cases} MV_h(x,y)=\dfrac{b_h-a_h}{w}x+\dfrac{b_v-a_v}{w}y+a_h \\ MV_v(x,y)=\dfrac{b_v-a_v}{w}x+\dfrac{b_h-a_h}{w}y+a_v \end{cases} \tag{1}$$

其中，a_h，a_v，b_h，b_v分别表示左上角和右上角 CPMV 的横纵坐标，w表示当前块的宽度；同样地，对于更复杂的规则形变运动，如图 2(b)所示，可通过六参数仿射模型得到每个子块的 CPMV，具体计算式如下：

$$\begin{cases} MV_h(x,y)=\dfrac{b_h-a_h}{w}x+\dfrac{c_h-a_h}{h}y+a_h \\ MV_v(x,y)=\dfrac{b_v-a_v}{w}x+\dfrac{c_v-a_v}{h}y+a_v \end{cases} \tag{2}$$

其中，a_h，a_v，b_h，b_v，c_h，c_v 分别表示左上角、右上角和左下角 3 个位置的 CPMV 的横纵坐标，w 和 h 分别表示当前块的宽和高。



(a)四参数仿射模型　　(b)六参数仿射模型

图 2　经过平移等运动的仿射运动模型

Fig. 2　Affine motion model after translational and other movements

为降低计算复杂度，AMC 会将当前块划分为 4×4 的亮度子块，再通过式(1)则可以得到四参数仿射模型下各个子块的运动矢量，如图 3 所示。

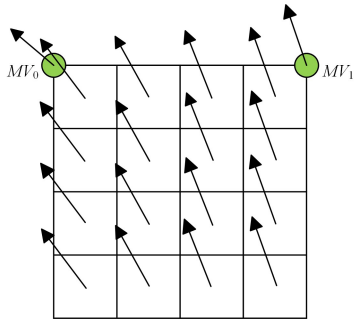


图 3　AMC 后划分的 4×4 个子块

Fig. 3　AMC division 4×4 sub blocks

2.2　仿射 AMVP 模式

2.2.1　MVP 获取

对于旋转、缩放、拉伸等非平移运动 CU 需要使用仿射 AMVP 模式来达到高效预测编码。

在 H.266/VVC 中，仿射 AMVP 模式的 CPMVP(CPMV 的预测值)候选项有以下 5 种构造方式。

1)空域相邻仿射模式 CU 继承

该构造方式要求两个相邻 CU 的参考图像是一致的以及需要进行两次扫描，扫描方向分别如图 4 所示的绿色箭头，将两次扫描路径上第一个有效 CU 设为 CPMVP 候选项。

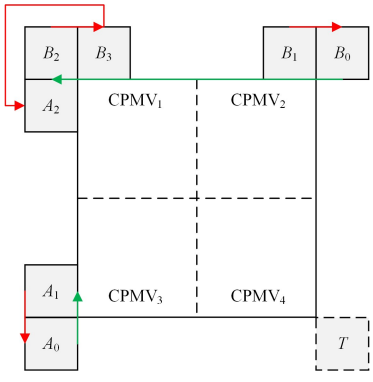


图 4　候选相邻块的位置

Fig. 4　Position of candidate adjacent blocks

假设 A 块被选定，则当前 CU 的 CPMV 可根据 A 块的仿射模型类型(四参数或六参数)并结合式(1)或式(2)得到。

2)空域相邻 CU 的平移 MV 构造

候选 CPMVP 也可以通过空域相邻 CU 的平移 MV 构造得到。其中 $CPMV_x$(x 为[1,3])沿着如图 4 所示的红色箭头路径扫描，将 3 次扫描路径上第一个有效 CU 设为 $CPMV_x$(x 为[1,3])。

3)空域相邻 CU 的平移 MV 填充

利用一个固定的 MV 生成四参数和六参数仿射模型，MV 依次选用 2)中的 $CPMV_3$，$CPMV_2$，$CPMV_1$(需有效)。

4)时域平移 MV 填充

在同位图像中得到如图 5 所示的 Br 块的同位块，如该同位块无效或不是帧间模式，则使用中心块 Ctr 的同位块。若同位块有效，则将经过时域缩放后的 MV 作为填充的固定 MV 后生成四参数和六参数仿射模型。

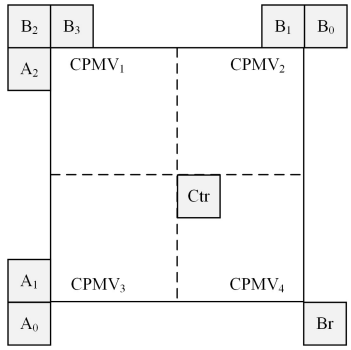


图 5　相邻 CU 的位置

Fig. 5　Position of adjacent CU

5)零值 MV 填充

若经过上述 4 种构造，AMVP list 中的候选项数量仍然小于 2，则将零值插入候选列表的末尾。

2.2.2　MV 获取

在推导出 AMVP list 中的两个候选项后，需要执行仿射 ME 来找到最优参数，即一个块的两个 MV。由于 ME 的高复杂性，快速 ME 算法[18-19]一直是研究的热点。如在 AVC 参考软件中引入了著名的增强预测区域搜索(Enhanced Predictive Zonal Search，EPZS)[20]算法，在 HEVC 参考软件中引入了基于 EPZS 的测试区域搜索(Test Zone Search，TZS)[21]算法，这两种算法对于平移运动模型的 ME 有较好的表现，但直接将它们用于其他复杂运动模型会比较困难，因此文献[22-23]提出了一种基于梯度的快速仿射 ME 算法，并被采用到 VVC 参考软件中。在每次迭代过程中，该算法能够使两个 MV 快速收敛至最优组合并且能对两个 MV 同时进行求解。该算法的具体步骤如下。

1)初始化两个 MV 为$\{MV_0^{(0)},MV_1^{(0)}\}$。

设置 3 个 MV 候选项$\{MVP_0^0,MVP_1^0\}$，$\{MVP_0^1,MVP_1^1\}$和$\{MV^T,MV^T\}$，其中MVP_0和MVP_1为 AMVP list 中的两个候选项，MV^T为在 AME 过程前进行的平移运动补偿(Translational Motion Compensation，TMC)的 ME 过程所输出的 MV。接下来，对 3 个候选项分别计算执行 AMC 后预测块的 SATD 代价，将 SATD 代价最小的值和候选项分别设为$\{MV_0^{(0)},MV_1^{(0)}\}$和$Cost^{(0)}$。最后，将迭代初的变量 i 初始化为

0，$\{MV_0^*, MV_1^*\}$初始化为$\{MV_0^{(0)}, MV_1^{(0)}\}$，$Cost^*$初始化为$Cost^{(0)}$。

2)通过$\{MV_0^{(i)}, MV_1^{(i)}\}$推导出$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$。

设$MV^{(i)}(x,y)=(MV_h^{(i)}(x,y), MV_v^{(i)}(x,y))^T$，通过式(1)可以计算出$MV^{(i)}(x,y)$，则用$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$进行AMC时，将当前块$A$的总误差记为$Err$，则有：

$$Err=\sum_{(x,y)\in A}\| Org(x,y)-Ref(x+MV_h^{(i+1)}(x,y), y+MV_v^{(i+1)}(x,y))\|^2 \tag{3}$$

其中，$Org(x,y)$表示当前像素强度，$Ref(x,y)$表示参考像素强度。设$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$和$\{MV_0^{(i)}, MV_1^{(i)}\}$的差值为$dMV$，则将$dMV$代入式(1)可得：

$$d(x,y)=MV^{(i+1)}(x,y)-MV^{(i)}(x,y)=-A(x,y)dMV \tag{4}$$

$$A(x,y)=\begin{pmatrix} 1-\frac{x}{w} & \frac{x}{w} & \frac{y}{w} & -\frac{y}{w} \\ -\frac{y}{w} & \frac{y}{w} & 1-\frac{x}{w} & \frac{x}{w} \end{pmatrix} \tag{5}$$

再根据泰勒展开，则有：

$$\begin{aligned} &Ref(x+MV_h^{(i+1)}(x,y), y+MV_v^{(i+1)}(x,y)) \\ &=Ref(x+MV_h^{(i)}(x,y)+d(x,y), y+MV_v^{(i+1)}(x,y)+d(x,y)) \\ &\approx Ref(x+MV_h^{(i)}(x,y), y+MV_v^{(i)}(x,y))+G(x,y)\,d(x,y) \end{aligned} \tag{6}$$

$$G(x,y)=\left(\frac{\partial Ref}{\partial x}, \frac{\partial Ref}{\partial y}\right)(x+MV_h^{(i)}(x,y), y+MV_v^{(i)}(x,y)) \tag{7}$$

其中，$G(x,y)$是在参考块A中像素点$(x+MV_h^{(i)}(x,y), y+MV_v^{(i)}(x,y))$的梯度值。然后，将式(4)和式(6)代入式(3)可得：

$$Err=\sum_{(x,y)\in A}\| -err(x,y)+G(x,y)A(x,y)dMV\|^2 \tag{8}$$

$$err(x,y)=Org(x,y)-Ref(x+MV_h^{(i)}(x,y), y+MV_v^{(i)}(x,y)) \tag{9}$$

可以看出，$err(x,y)$实际上表示的是第i次迭代中像素点(x,y)的预测误差。则dMV^*，即使Err最小的dMV，可通过对如下方程式求解得到：

$$\frac{\partial \sum_{(x,y)\in A}\| -err(x,y)+G(x,y)A(x,y)dMV\|^2}{\partial dMV}=0 \tag{10}$$

通过求解出的dMV^*，再由式(4)可以通过$\{MV_0^{(i)}, MV_1^{(i)}\}$推导出$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$。

3)更新$Cost^*$和$\{MV_0^*, MV_1^*\}$。

对推导出的$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$计算执行AMC后预测块的SATD代价$Cost^{(i+1)}$，若$Cost^{(i+1)}<Cost^*$，则将$Cost^*$设为$Cost^{(i+1)}$，$\{MV_0^*, MV_1^*\}$设为$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$。

4)进行ME迭代。

若$\{MV_0^{(i+1)}, MV_1^{(i+1)}\}$与$\{MV_0^{(i)}, MV_1^{(i)}\}$不相同且$i$小于最大迭代次数，则令$i=i+1$后重复步骤2)和步骤3)，反之将$\{MV_0, MV_1\}$设为$\{MV_0^*, MV_1^*\}$，并提前终止ME过程。

3 算法简介

为降低计算复杂度，本文从VVC新加入的AME帧间模式入手，主要从4个方面来降低AME的计算复杂度。首先，通过分析AME中各像素精度的占比，将AME中的像素精度只保留1/4像素精度，摒弃其余两种像素精度；其次，对迭代次数采用一个自适应计算式进行删减，以此实现在最大程度上减少迭代所需次数；然后，将细粒度搜索算法中CU的左下、左上、右上和右下角4个整像素用左下和右上角两个1/2像素进行替代；最后，采用SATD代价替代RDO代价，以此减少一些对算法影响不大但复杂度高的步骤。算法的流程图如图6所示。

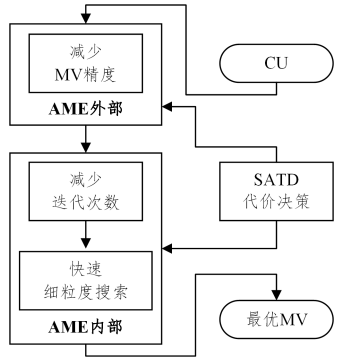


图6 AME快速算法的流程图

Fig. 6 Flowchart of AME fast algorithm

4 基于运动矢量精度的快速算法

为了更加精确地描述旋转、缩放的运动块，在VVC的AME中，存在3种运动矢量精度，即整像素、1/4和1/16像素精度。这3种精度的MV同样是基于RDO决策来选择代价最小的精度作为最优精度。对于Affine AMVP模式来说，如果在检查Affine merge/skip模式、merge/skip模式、1/4精度普通AMVP模式和1/4精度Affine AMVP四者直接的RD成本后未选择Affine inter模式，则无需再检查1/16精度和整像素精度Affine inter模式。此外，在1/4精度Affine inter模式下获得的Affine参数可以被用作1/16亮度采样和1/4精度Affine inter模式下的起始搜索点，以减少RDO代价的计算次数。

因此本文统计了几种运动矢量精度的占比情况，如图7所示。可以发现，在AME中，1/4像素精度的占比在3种像素精度中最高。对此，直接摒弃其余两种像素精度，即整像素和1/16像素精度，只保留1/4像素精度。

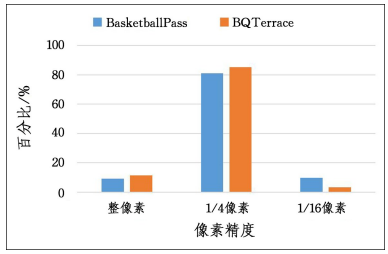


图7 AME中几种像素精度的占比情况

Fig. 7 Proportion of several pixel accuracies in AME

5 快速仿射运动搜索方案

5.1 迭代过程优化

在AME的迭代过程中，每次迭代都会更新CPMV，然后基于CPMV进行运动补偿，得到预测图像，进而计算RDO

代价。迭代完成后，具有最小代价的 CPMV 会被保留下来。

其中，VTM-10.0 中的标准迭代次数的计算式如下：

$$iterTime=\begin{cases}4\text{-}parameterAME\begin{cases}3, & Bslice\\ 5, & Pslice\end{cases}\\ 6\text{-}parameterAME\begin{cases}3, & Bslice\\ 4, & Pslice\end{cases}\end{cases} \tag{11}$$

为了最大程度地减少迭代次数，本文分析了迭代次数与量化参数（Quantization Parameter，QP）、slice 类型以及 CU 大小的关系。经过统计分析，发现迭代次数与 QP 以及 CU 的尺寸大小成正比，且与参数模型密切相关，从而得到了如下的自适应迭代计算式。

$$iterTime=\begin{cases}\max\left(\min\left(floor\left(\dfrac{M\cdot QP\cdot\alpha}{M_{max}}\right),\theta\right),1\right), & Bslice\\ \max\left(\min\left(floor\left(\dfrac{M\cdot QP\cdot\alpha}{M_{max}}\right),\eta\right),1\right), & Pslice\end{cases} \tag{12}$$

其中，M 表示 CU 尺寸大小；QP 表示量化参数；α 是调节参数，在 QP 大于等于 27 时被设为 2，小于 27 时被设为 4；θ 和 η 是两个固定值，在四参数中，两者分别为 3 和 5，在六参数中，两者分别为 3 和 4。

5.2 快速 CPMV 细粒度搜索

当基于梯度的迭代过程完成时，可以获得具有最低 RDO 成本的最优 CPMV。然而，迭代后的 CPMV 可能不是最终的最优 MV，可能需要精细调整。因此，无论是否根据 RDO 成本调整当前 CPMV，CU 周围都添加了 8 个方向，如图 8 所示。

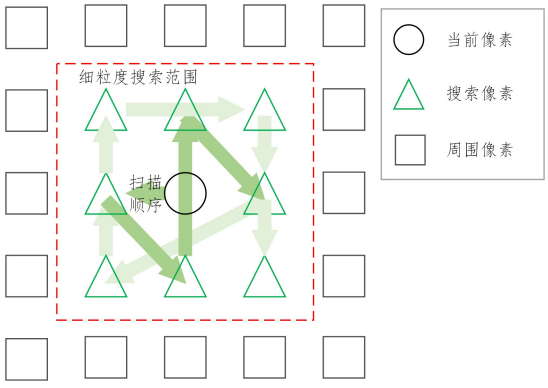


图 8 VTM-10.0 细粒度搜索方案

Fig. 8 VTM-10.0 fine granularity search scheme

在确定是否调整 CPMV 的过程中，某个方向的 RDO 成本可能会变大。为了减少不必要的计算，VTM-10.0 优化了细粒度搜索过程。首先，执行上、下、左和右方向的初始搜索，如果 4 个方向中的某个方向使 RDO 成本变小，则将继续进一步搜索，即左下、左上、右上和右下方向；否则将停止细粒度搜索过程，并将细粒度搜索之前的 CPMV 反转为最优 MV。

然而，由于最初 4 个方向的遍历，细粒度搜索的高复杂性仍然存在。因此，在 VTM-10.0 算法的基础上，提出了一种优化的细粒度搜索算法。

如图 9 所示，在当前像素的左下角和右上角插入了两个 1/2 像素，这两个分像素用于替换顶部、底部、左侧和右侧 4 个完整像素的搜索过程。如果这两个分像素的运动补偿后的 RDO 成本小于迭代完成后的 RDO 成本，则认为需要进行细粒度调整，否则直接跳过细粒度调整过程。

如图 10 所示，对 BQMall、Johnny 等多个序列进行了基于原始搜索方向和新给定的分像素搜索方向的 RDO 成本计算，并对各个搜索范围的成本的平均值进行了比较。可以看出，基于分像素的搜索在判断细粒度搜索的必要性方面具有与原始全像素搜索相同的效果，并且对是否执行进一步的细粒度搜索具有准确的判断。因此，本文使用两个分像素而不是 4 个整体像素来确定细粒度优化的必要性的方法是完全可行的，基于该方法可以简化 AME 过程。

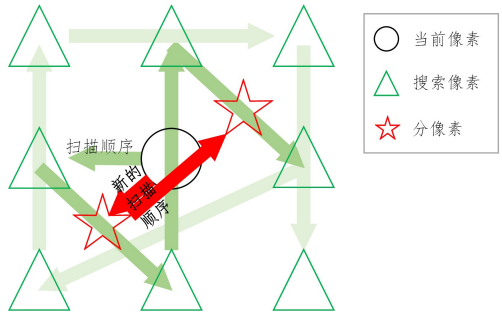
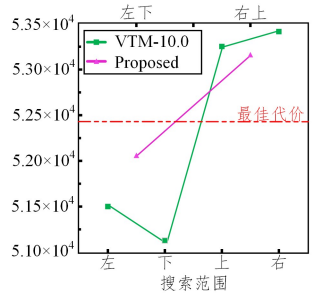
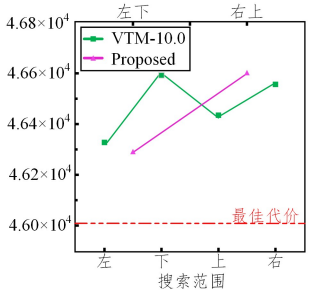


图 9 提出的快速细粒度搜索方案

Fig. 9 The proposed fast fine granularity search scheme



(a)需要细粒度搜索



(b)无需细粒度搜索

图 10 提出的快速细粒度搜索方案代价比较

Fig. 10 Cost comparison of the proposed fast fine granularity search schemes

6 基于 SATD 的快速率失真决策

在 AME 内部，每次迭代都要计算 RDO 代价，完成迭代后，具有最小代价的 CPMV 会被保留下来，用于之后的运动补偿等一系列步骤。除此之外，还存在是否进行细粒度调整的 RDO 代价比较。在 AME 外部，要将 AME 与其他帧间模式进行 RDO 代价比较，来选择最优的帧间模式。由此可见，RDO 代价的比较计算在帧间预测部分的复杂度是极高的。

本文方案主要从 AME 内部来降低复杂度。具体来说，对于每次迭代的 CPMV，没必要每次都计算 RDO 代价。

SATD 代价在一定程度上也可以用于决策，只是没有 RDO 代价精确度高。但是，计算 SATD 不需要经过反量化、反变换等步骤，可以说是极大降低了复杂度。因此，本文方案在迭代过程中采用 SATD 代价决策来选择最优的 CPMV，这可以比 RDO 决策缩短大量时间。具体来说，SATD 的计算式如下：

$$\Psi=\sum_i |d_i| \tag{13}$$

其中，Ψ 表示 SATD，i 是一个索引值，d_i 表示 i 位置的变换系数。紧接着，将码率 R 和 SATD 采用拉格朗日优化因子进行加权叠加，表达式如下：

$$\omega=\Psi+\lambda \cdot R \tag{14}$$

至此，就得到了可以用来代替迭代过程 RDO 代价的 SATD 代价，即上式中的 ω。需要说明的是，这种方式可以不经过反量化和反变换等步骤，变换结束后就可以计算。相比原始的 RDO 代价，极大地降低了计算复杂度。

7 实验结果与分析

7.1 实验结果

为评估本文提出的仿射运动估计快速算法在 RD 性能和时间复杂度上的表现，使用 VVC 参考软件 VTM-10.0[24] 进行实验。实验采用 JVET 通用测试条件(Common Test Condition，CTC)[25] 在默认配置下进行测试。QP 值的取值为 {22，27，32，37}，使用 BD-Rate[26] 作为评价编码的性能指标。由于算法复杂度的变化无法直接得到，因此本文定义时间节省 ΔT_{Enc} 计算来衡量算法时间复杂度的变化，表达式如下：

$$\Delta T_{\mathrm{Enc}}=\frac{T_{\mathrm{Anc}}-T_{\mathrm{Pro}}}{T_{\mathrm{Anc}}}\times 100\% \tag{15}$$

其中，T_{Pro} 表示本文算法的总编码时间，T_{Anc} 表示原始 VVC 算法的总编码时间。

表 1 和表 2 分别列出了本文算法在 LDB 和 RA 配置下的编码性能和时间节省，为了展示本文算法的有效性，将其与文献[14]中的算法进行了比较。

表 1 LDB 配置下的编码性能

Table 1 Encoding performance with LDB configuration

(%)

Class	Sequence	文献[14]中的方法		本文算法	
		De-Enc	BD-Rate	De-Enc	BD-Rate
B	MarketPlace	5.89	0.11	9.69	0.22
	RitualDance	4.45	0.08	10.56	0.16
	Cactus	5.82	0.17	11.34	0.14
	BasketballDrive	6.35	0.08	7.63	0.20
	BQTerrace	6.15	0.04	7.28	0.17
C	BasketballDrill	4.92	0.09	8.05	0.13
	BQMall	6.15	0.03	9.72	0.02
	PartyScene	4.34	0.15	7.26	−0.02
	RaceHorses	4.82	0.02	8.34	0.14
D	BasketballPass	5.91	0.06	7.85	0.18
	BQSquare	8.35	0.24	7.05	0.04
	BlowingBubbles	6.17	0.08	6.34	0.19
	RaceHorses	6.35	0.02	9.04	0.16
E	FourPeople	5.29	0.02	6.32	0.02
	Johnny	4.15	0.04	7.65	0.13
	KristenAndSara	4.34	0.10	8.34	0.12
F	BasketballDrillText	8.61	0.08	8.54	0.12
	ArenaOfValor	5.82	0.02	8.77	0.01
	SlideEditing	6.34	0.03	6.98	0.05
	SlideShow	7.92	−0.05	10.12	−0.11
Average		5.91	0.07	8.34	0.10

表 2 RA 配置下的编码性能

Table 2 Encoding performance with RA configuration

(%)

Class	Sequence	文献[14]中的方法		本文算法	
		De-Enc	BD-Rate	De-Enc	BD-Rate
A1	Tango2	7.34	0.04	10.15	0.04
	FoodMarket4	6.86	0.06	12.32	0.02
	Campfire	4.32	0.02	7.15	0.08
A2	CatRobot	7.28	0.08	9.85	0.04
	DaylightRoad2	7.01	0.17	8.65	0.10
	ParkRunning3	5.15	0.06	6.19	0.15
B	MarketPlace	5.34	0.12	10.82	0.34
	RitualDance	4.32	0.05	10.64	0.29
	Cactus	4.15	0.18	6.82	0.10
	BasketballDrive	5.28	0.12	7.27	0.15
	BQTerrace	7.47	0.09	9.64	0.21
C	BasketballDrill	3.43	0.10	9.91	0.14
	BQMall	5.78	0.07	10.38	0.10
	PartyScene	4.66	0.30	6.25	0.05
	RaceHorses	3.44	0.03	9.34	0.13
D	BasketballPass	3.15	0.06	6.91	0.22
	BQSquare	9.87	0.48	12.36	0.09
	BlowingBubbles	6.06	0.14	9.72	0.05
	RaceHorses	5.11	0.07	8.81	0.17
F	BasketballDrillText	7.44	0.15	9.05	0.13
	ArenaOfValor	5.15	0.09	4.64	0.08
	SlideEditing	6.32	0.01	7.17	0.06
	SlideShow	7.11	0.01	9.09	−0.05
Average		5.74	0.11	8.83	0.12

但由于文献[14]是中的算法在较早的 VTM-3.0 上进行的快速算法，因此本文将其移植在了同一平台进行，即 VTM-10.0。可以看出，在 LDB 配置下，本文算法节省了 8.34% 的时间，性能仅仅损失 0.10%；在 RA 配置下，本文算法节省了 8.83% 的编码时间，性能仅仅损失 0.12%。

7.2 结果分析

通过 7.1 节的实验结果，可以看出本文算法在时间复杂度上优于文献[14]中的算法。主要原因是文献[14]中的算法仅仅是针对加速使用多类型树(MMT)时 AME 的编码复杂度，当遇到使用 MMT 较少的序列时加速效果较差。而本文从 AME 的外部以及内部共 4 个方面来加速 AME，因此适用性更强，加速效果更明显，从实验结果可以看出本文算法相比文献[14]中的算法编码时间节省高出近 3 个百分点，从而证实了本文算法的高效性。

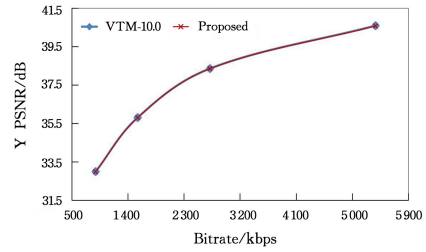
但是，从实验结果也能看出本文算法还存在一定的性能损失。主要原因如下：

1)尽管在 AME 中大部分运动矢量精度是 1/4 像素精度，但依旧存在其他两种像素精度，在这种情况下，本文算法没有达到相应的精度，造成了一定的性能损失；

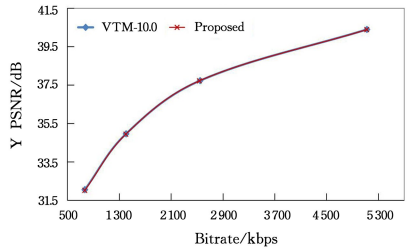
2)尽管通过迭代次数的自适应计算式在最大程度上减少了迭代次数，但是得到的可能是局部的最小代价的 CPMV，从而造成了一定的性能损失；

3)尽管用 SATD 代价来代替 RDO 代价大大降低了计算复杂度，但是 SATD 代价的精度没有 RDO 代价的高，因此存在一定的性能损失。

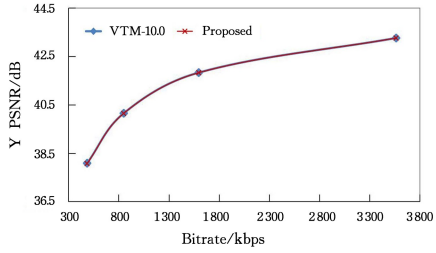
此外，图 11 给出了本文算法在 LDB 和 RA 配置下，与 VTM-10.0 相比对于 BQMall 和 Johnny 两个序列的性能损失情况，可以发现两条曲线几乎重合，即性能损失几乎可以忽略不计。因此，也印证了本文算法的合理性和有效性。



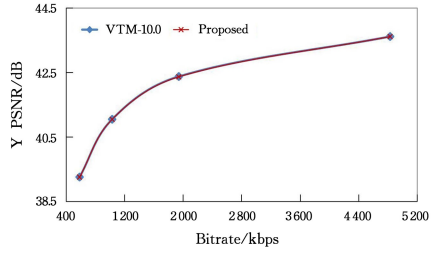
(a) LDB 配置下的 BQMall 序列



(b) RA 配置下的 BQMall 序列



(c) LDB 配置下的 Johnny 序列



(d) RA 配置下的 Johnny 序列

图 11　性能对比

Fig. 11　Performance comparison

结束语　最新的 VVC 视频编码标准以付出巨大计算复杂度的代价为前提，相比 HEVC 来说提高了 50% 的压缩效率。为降低其中的计算复杂度，本文提出了一种基于统计分析的仿射运动估计快速算法，并通过实验验证了该算法的高效性，同时对本文方法的优缺点进行了分析。实验与分析结果表明，与 H.266/VVC 参考软件 VTM-10.0 相比，该算法在性能几乎无损失的情况下，一定程度上解决了由 H.266/VVC 新引入的 AME 帧间模式所带来的计算复杂度高的问题。AME 算法复杂度的降低对整个帧间预测乃至 VVC 编码器复杂度的降低做出了一定贡献。此外，由于样本的多样性，本文提到的快速细粒度搜索方法只适用于部分样本，学者可以自行测试效果来选择是否使用该方法。

参 考 文 献

[1] WIEGAND T, SULLIVAN G J, BJONTEGAARD G, et al. Overview of the H.264/AVC video coding standard[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(7): 560-576.

[2] SULLIVAN 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] WIECKOWSKI A, MA J, SCHWARZ H, et al. Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard[C]// 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019: 4130-4134.

[4] TUN E E, ARAMVITH S, ONOYE T. Low complexity mode selection for H.266/VVC intra coding[J]. ICT Express, 2022, 8(1): 83-90.

[5] HUANG Y W, AN J, HUANG H, et al. Block partitioning structure in the VVC standard[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 31(10): 3818-3833.

[6] PARK S H, KANG J W. Fast affine motion estimation for versatile video coding (VVC) encoding[J]. IEEE Access, 2019, 7: 158075-158084.

[7] CHOWDARY T A, NALLURI P. Multiple Transform Selection in Versatile Video Coding: a Review[C]// 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, 2022, 1: 991-996.

[8] BROSS B, WANG Y K, YE Y, et al. Overview of the versatile video coding (VVC) standard and its applications[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 31(10): 3736-3764.

[9] HOANGVAN X, NGUYENQUANG S, DINHBAOM, et al. Fast QTMT for H.266/VVC intra prediction using early-terminated hierarchical CNN model[C]// 2021 International Conference on Advanced Technologies for Communications (ATC). IEEE, 2021: 195-200.

[10] WANG K, LIANG H, ZHANG S, et al. Fast CU Partition Method Based on Extra Trees for VVC Intra Coding[C]// 2022 IEEE International Conference on Visual Communications and Image Processing (VCIP). IEEE, 2022: 1-5.

[11] WANG Z, WANG J, YANG J, et al. A Fast Transform Algorithm for VVC Intra Coding[C]// 2022 11th International Conference on Communications, Circuits and Systems (ICCCAS). IEEE, 2022: 237-240.

[12] ZHANG Z, FU C H, XIE K, et al. Fast VVC Intra Coding by Skipping Redundant Coding Block Structures and Unnecessary Directional Partition[C]// 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE, 2022: 84-89.

[13] LIU C, SUN H, KATTO J, et al. QA-filter: A QP-adaptive convolutional neural network filter for video coding[J]. IEEE Transactions on Image Processing, 2022, 31: 3032-3045.

[14] PARK S H, KANG J W. Fast affine motion estimation for versatile video coding (VVC) encoding[J]. IEEE Access, 2019, 7: 158075-158084.

[15] GUAN X, SUN X. VVC Fast ME Algorithm Based on Spatial Texture Features and Time Correlation[C]// 2021 International

Conference on Digital Society and Intelligent Systems(DSInS). IEEE,2021:371-377.
[16] JUNG S,JUN D. Context-based inter mode decision method for fast affine prediction in versatile video coding[J]. Electronics, 2021,10(11):1243.
[17] DUARTE A,GONÇALVES P,AGOSTINI L,et al. Fast affine motion estimation for vvc using machine-learning-based early search termination[C]// 2022 IEEE International Symposium on Circuits and Systems(ISCAS). IEEE,2022:1-5.
[18] ZHU S,MA K K. A new diamond search algorithm for fast block-matching motion estimation[J]. IEEE Transactions on Image Processing,2000,9(2):287-290.
[19] ZHU C,LIN X,CHAU L P. Hexagon-based search pattern for fast block motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology,2002,12(5):349-355.
[20] TOURAPIS A M. Enhanced predictive zonal search for single and multiple frame motion estimation[C]// Visual Communications and Image Processing 2002. SPIE,2002,4671:1069-1079.
[21] PURNACHAND N,ALVES L N,NAVARRO A. Improvements to TZ search motion estimation algorithm for multiview video coding[C]// 2012 19th International Conference on Systems,Signals and Image Processing(IWSSIP). IEEE,2012:388-391.
[22] ZHANG K,CHEN Y W,ZHANG L,et al. An improved framework of affine motion compensation in video coding[J]. IEEE Transactions on Image Processing,2018,28(3):1456-1469.
[23] LI L,LI H,LIU D,et al. An efficient four-parameter affine motion model for video coding[J]. IEEE Transactions on Circuits and Systems for Video Technology,2017,28(8):1934-1948.
[24] CHEN J,YE Y,KIM S. Algorithm description for versatile video coding and test model 10(VTM 10) JVET-S2002[C]// 19th Meeting of the Joint Video Exploration Team(JVET). 2020: 1-67.
[25] BOYCE J,SUEHRING K,LI X,et al. JVET-J1010:JVET common test conditions and software reference configurations[C]// 10th Meeting of the Joint Video Experts Team. 2018:JVET-J1010-v1.
[26] BJONTEGAARD G. Calculation of average PSNR differences between RD-curves(VCEG-M33)[Z]. 2001.

ZHONG Yucheng, born in 2002, undergraduate. His main research interests include video processing and compression technology.

CUI Yan, born in 1988, postgraduate. Her main research interests include image video coding and VLSI design.