

高效率视频编码快速帧内预测算法

张峻, 董兰芳*, 余家奎

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)

(*通信作者电子邮箱 lfdong@ustc.edu.cn)

摘要:针对高效率视频编码(HEVC)帧内预测过程中,编码单元四叉树划分算法计算复杂度极高的问题,提出一种基于多重纹理特征的 HEVC 帧内编码单元快速划分算法,该算法能够缩小划分的深度区间。首先,使用自定义的纹理提取方法提取出编码单元中的多重纹理特征;其次,使用支持向量机(SVM)对多重纹理特征参数进行训练,得出决策函数;最后,根据决策函数,跳过前面不必要的划分和提前终止划分。实验结果表明,同原始 HM 12.0 相比,快速划分算法编码时间平均减少 43.23%,码率平均增加 0.84%,明显提高了帧内编码效率。此外,所提算法容易与其他算法进行融合,进一步降低 HEVC 的帧内计算复杂度。

关键词:高效率视频编码;帧内预测;编码单元;四叉树划分;支持向量机

中图分类号: TN919.8 **文献标志码:** A

Fast intra prediction algorithm for high efficiency video coding

ZHANG Jun, DONG Lanfang*, YU Jiakui

(College of Computer Science and Technology, University of Science and Technology of China, Hefei Anhui 230027, China)

Abstract: Concerning the problem that the computational complexity of Coding Unit (CU) quad-tree splitting algorithm for High Efficient Video Coding (HEVC) intra prediction is very high, a fast intra CU splitting algorithm was proposed, which can narrow depth range of CU splitting. Firstly, multiple texture features were achieved through new defined texture extraction algorithm. Secondly, in order to get a decision function, Support Vector Machine (SVM) was employed to train the feature parameters. At last, based on decision function, CU splitting depth range was narrowed by skipping unnecessary splitting and early terminating splitting. The experimental results show that compared with the reference model HM 12.0, the fast CU splitting algorithm provides about 43.23% time savings on average with 0.84% increasing on bit-rate which obviously improves the coding efficiency. Besides, the proposed algorithm is easy to be combined with other methods to further reduce the computational complexity for HEVC intra coding.

Key words: High Efficiency Video Coding (HEVC); intra prediction; Coding Unit (CU); quad-tree splitting; Support Vector Machine (SVM)

0 引言

2013年1月视频编码联合组(Joint Collaborative Team on Video Coding, JCT-VC)发布了高效率视频编码(High Efficiency Video Coding, HEVC)标准^[1]。在保证相同图像质量的条件下,HEVC与上一代高级视频编码标准(Advanced Video Coding, AVC)^[2]相比,节省了大约50%的码率,但编码复杂度增加了2到4倍^[3-4]。HEVC要想在市场中代替AVC,实现商业化,就必须降低计算复杂度。

HEVC视频编码标准中采用了一种新的编码单元(Coding Unit, CU)划分算法^[5-6]。该算法按照深度优先遍历方式将最大编码单元(Largest Coding Unit, LCU)进行递归划分,划分深度在0~3,LCU被划分成尺寸从64×64到8×8的不同编码子单元。相对于AVC标准中的16×16宏块(MacroBlock, MB)结构^[7],HEVC中新的CU划分方式提高了编码的效率;但也大大增加了编码的复杂度,限制了HEVC标准的普及和应用。因此在保证压缩效率和图像质量的条件下,研究快速CU划分算法来降低计算复杂度是很有必要

的^[3-4]。

帧内编码单元划分的快速算法有两种不同的优化策略。一种是基于图像的空间相关性:文献[8]提出了一种快速CU划分决策算法,算法利用空间相关性,根据相邻CU的划分深度来预测当前CU的划分深度,达到降低编码复杂度的效果;实验结果显示,与标准测试模型HM5.2相比,改进后的算法平均编码时间减少了21%。另一种是基于图像的梯度纹理特性:文献[9]提出一种基于非标准化的方向直方图(non-normalized Histogram of Oriented Gradient, n-HOG)的快速算法,该算法使用n-HOG特征线下训练出一个码本,然后通过码本预判划分深度;实验表明,该算法平均减少了33.28%的编码时间。文献[10]使用绝对误差和(Sum of Absolute Transformed Difference, SATD)来提前终止CU分割,根据CU中最小SATD与阈值的关系决定是否提前终止划分;实验表明,该算法平均编码时间减少30.5%。文献[11]使用灰度共生矩阵(Gray Level Co-occurrence Matrix, GLCM)来反映图像的纹理分布,根据灰度共生矩阵提前终止CU划分;实验表明,该算法平均节约26.08%的编码时间。文献[12]提出了

收稿日期:2015-03-24;修回日期:2015-05-19。

作者简介:张峻(1989-),男,江苏扬州人,硕士研究生,主要研究方向:视频编解码;董兰芳(1970-),女,安徽舒城人,副教授,硕士,CCF会员,主要研究方向:视频编码、视频智能分析;余家奎(1990-),男,安徽阜阳人,硕士研究生,主要研究方向:视频智能分析。

一种基于纹理复杂度的快速 CU 划分算法,使用像素的标准偏差(Standard Deviation, SD)来衡量纹理的复杂度,根据标准偏差对 CU 划分进行优化;实验数据表明,该算法平均节省 37.6% 的编码时间。

在已有算法的基础上,本文提出一种新的算法对编码单元划分进行剪枝。首先根据本文提出的纹理提取算法将图像的纹理特征分为三类,分别为强纹理、弱纹理和填充,并使用方差衡量纹理的分布特征;然后使用支持向量机(Support Vector Machine, SVM)^[13]对这些特征进行线下训练得到决策函数;最后根据决策函数对 CU 划分的深度进行预测和裁剪,从而实现 CU 划分算法的复杂度优化。

1 多重纹理提取

1.1 基于 3×3 像素块的纹理提取方法

对标准测试序列 BasketballPass 使用帧内编码后,图像中各个 LCU 的划分方式如图 1 所示。帧内预测的思想是将 CU 左边缘和上边缘的像素作为参考像素,对参考像素进行插值滤波,得到 CU 中的预测像素。因此预测像素在某个方向上是平坦均匀的。从图 1 中可见,当 LCU 内原图像素分布平坦均匀、纹理简单时,CU 划分的尺寸较大;当 LCU 内原图像素纹理十分复杂时,划分的尺寸较小。

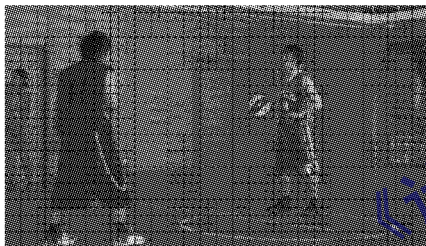


图1 帧内预测 CU 划分

根据以上观察,帧内 CU 划分后,如果划分后得到的子 CU 中的预测像素与原图像素十分相似,就近似认为该尺寸的划分是近似最优解。由于预测像素是由周围像素插值滤波得到,所以预测像素具有均匀平坦的特性,如果原图也有这个特性,则认为预测像素与原图像素相似。本文采用像素分布的均匀平坦性来评价预测像素与原图像素的相似性。为客观描述像素的均匀平坦性,本文使用像素值与八邻域均值之差的绝对值来衡量像素的均匀性^[14],计算公式如下:

$$\bar{P}_{i,j} = \frac{1}{8} \left(\sum_{m=-1, m \in \mathbb{Z}}^1 \sum_{n=-1, n \in \mathbb{Z}}^1 P_{i+m, j+n} - P_{i,j} \right) \quad (1)$$

$$D = |\bar{P}_{i,j} - P_{i,j}| \quad (2)$$

其中: $P_{i,j}$ 代表图像中第 i 行第 j 列的像素值, $\bar{P}_{i,j}$ 代表第 i 行第 j 列的像素的八邻域均值。当 D 越大时,像素 $P_{i,j}$ 与八邻域的均值差异越大,则八邻域内的像素值变化剧烈,该 3×3 的像素块内纹理可能很复杂,不适合采用插值滤波预测;反之当 D 越小时,像素 $P_{i,j}$ 与八邻域的均值差异越小,八邻域内的像素值平坦均匀,适合使用插值滤波预测。一幅图像中,如果像素位于图像的边缘,那么它是不存在八邻域的,因此不能使用式(1)。对于不存在八邻域的像素点,本文都不进行处理。

根据上述分析,只需要两个像素值 ($P_{i,j}, \bar{P}_{i,j}$) 就可以简要描述一个 3×3 像素方块内的纹理情况,当式(1)中的 $P_{i,j} \in [0, 255], P \in \mathbb{N}$ 时,根据排列组合原理一共存在 256×256 种 3×3 像素方块纹理,因此可以使用数组 $a[256][256]$ 的下标表示出所有的纹理种类。例如 $a[1][2]$, 数组的下标

为 (1,2), 表示 $P_{i,j} = 1, \bar{P}_{i,j} = 2$ 的 3×3 像素方块纹理,而数组 $a[1][2]$ 所存储的数值,表示该 3×3 像素方块纹理出现的次数。

为了反映一幅图像中 3×3 像素方块纹理和 D 的统计特性,定义了一种统计直方图,具体实现过程分为三步:

1) 将大小为 $m \times n$ 的 RGB 图像转化为灰度图像,则灰度图像中任意一点的灰度值可以表示为 $P_{i,j}, P_{i,j} \in [0, 255]$ ($i \in [0, m-1], j \in [0, n-1]$)。

2) 使用数组 $a[256][256]$ 表示所有的 3×3 像素块纹理,将数组 $a[256][256]$ 初始化为 0,由于本文不对图像边缘像素点进行处理,所以只计算图像中 $(m-2) \times (n-2)$ 个非边缘像素点,根据式(3)统计出一幅图像中不同纹理出现的次数:

$$\sum_{i=1}^{m-2} \sum_{j=1}^{n-2} (a[P_{i,j}][\bar{P}_{i,j}]++) \quad (3)$$

3) 将二维数组 $a[256][256]$ 生成一幅灰度图像。因为统计图是以灰度图像的形式表达,所以需要遍历数组,找到最大值 a_{\max} ,将 a_{\max} 按照比例系数缩放至 255,同时也将数组中所有的元素进行缩放,最终生成的统计图如图 2 所示。

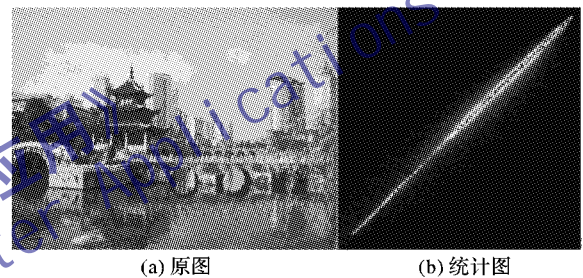


图2 图像转换为统计图

根据式(3),统计图中 x 轴等于 $P_{i,j}$, 统计图中 y 轴等于 $\bar{P}_{i,j}$, 数组 $a[x][y]$ 中存储的数值表示图中像素值为 x 、八邻域均值为 y 的 3×3 像素块的纹理在 $m \times n$ 的图像中出现的次数, D 等价于 $|x - y|$ 。由统计图可见,3×3 像素块纹理主要集中在对角线上,统计图中点的颜色越亮,表示该纹理出现的次数越多。统计图中主要有两种统计特性,一种是远离对角线的纹理统计点,这些点分布离散,呈现离散粒子状态。按照定义可知,距离主对角线远意味着 D 大,表明 3×3 像素块内各个像素点灰度突变越大,在图像中应该体现为物体的边界强纹理。根据以上分析,假设 $D > T_s$ 的像素点就认为是强纹理,其中 T_s 表示阈值。对 50 幅不同场景的图片进行实验,根据实验效果对比,取阈值 $T_s = 5$ 。当 $D > 5$ 时,将原图像中的像素点标记为白色,得到图 3。

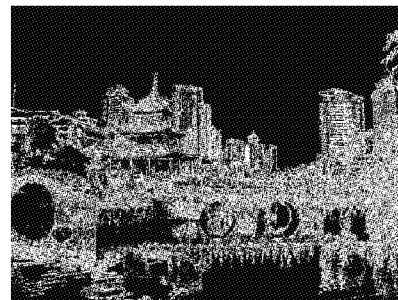


图3 强纹理提取图

另一种特性是靠近对角线的纹理统计点,这些点分布集中,呈现块状,但是中间会有黑色孔洞点。按照定义,距离主对角线近意味着 D 小,表示 3×3 像素块内各个像素点灰度突变

小,灰度值分布平坦均匀,在图像中应该体现为物体内部连续的填充纹理。但是统计图中黑色空洞部分在图像中代表什么呢?为了找出答案,需要进一步进行实验。黑色空洞产生的原因是这种 3×3 的纹理出现的次数较少,灰度值小,所以呈现黑色。那么 $a(x, y) < T_w$ 就能表示出这些出现次数较少的纹理,其中 T_w 表示阈值。为了求得 T_w 的值,对50幅不同场景的图像进行实验,当统计图中 $a(x, y)$ 与周围点值相比变化剧烈且点值较小时,视为黑色空洞点。实验过程中,当 $D \leq 5$ 且 $a(x, y)$ 小于周围点的均值三分之一时,视为黑色空洞点。对这些黑色空洞点求取均值,结果为20,所以本文取 $T_w = 20$ 。根据分析,黑色空洞点满足以下条件: $D \leq 5, a(x, y) < 20$ 。对原图像中满足以上两种特性的点进行标记,得到图4。

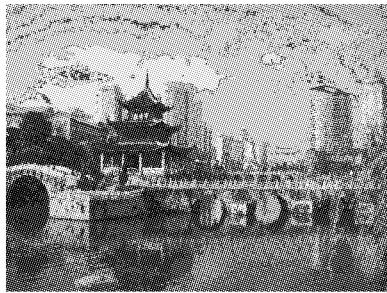


图4 原图中标记黑洞点图

从图4中可以看出,这些黑色空洞点主要集中在图片的天、水面等部分,也就是图像纹理渐变的地方,定义为弱纹理。为了说明强、弱纹理提取算法的通用性,对不同场景图像进行处理,处理结果如图5所示,从图中可见,多重纹理提取有一定的通用性。

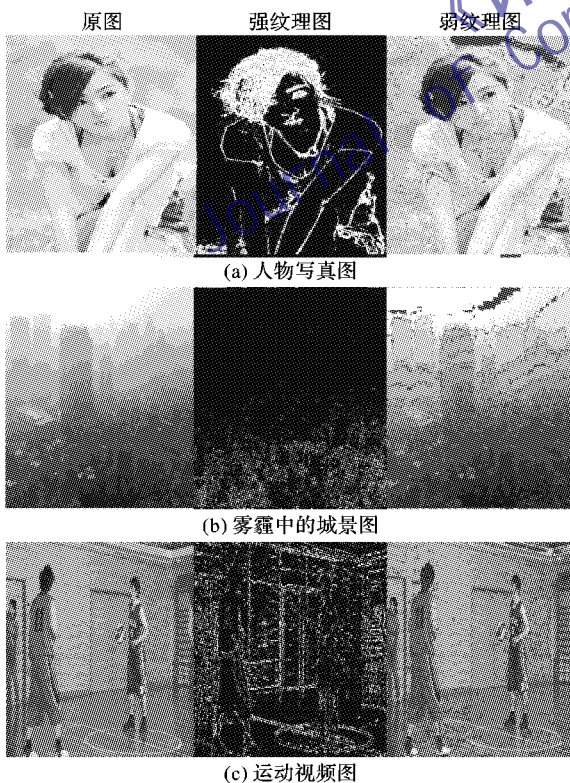


图5 不同场景的纹理提取图

1.2 训练特征参数选择

根据以上分析,将图像分为三种纹理,分别为强纹理、弱纹理和填充,这三种纹理的特征如表1所示。

表1 多级纹理提取参数表

| 纹理 | $ x - y $ | $a[x][y]$ |
|-----|-----------|-----------|
| 强纹理 | > 5 | 任意 |
| 弱纹理 | ≤ 5 | < 20 |
| 填充 | ≤ 5 | ≥ 20 |

使用表1中的阈值,提取出测试序列BasketballPass的弱纹理,如图6所示。从图中可见,CU中的弱纹理较少时,CU的最终尺寸较大。但有些尺寸较大的CU内也含有较多的弱纹理,为了去除这种干扰,引入了两个参数,分别是CU内 D 的最大值 D_{Max} 和CU中 D 的均值 D_{Mean} ,如果CU内 D_{Max} 和 D_{Mean} 值非常小,说明CU内部纹理突变小,像素值分布平坦,不需要再进行划分。



图6 弱纹理在CU划分中的分布图

但强纹理、弱纹理、填充、 D_{Max} 和 D_{Mean} 这些特征只能反映出CU内纹理的复杂度,不能反映出CU内纹理的分布情况。如果一个CU内的纹理复杂,但是这些纹理具有方向性,就不需要精细划分。如果只根据纹理的复杂度进行预判,很可能不会对CU继续划分。为了解决这一问题,使用方差来衡量CU中的纹理分布特性。方差计算公式如下:

$$V = \frac{1}{4} \sum_{i=0}^3 (X_i - \bar{X})^2 \quad (4)$$

其中: X_i 表示对当前CU进行四叉树划分后,第 i 个子CU的纹理特征参数,为了方便比较,需对纹理特征参数进行统一度量, X_i 表示的是CU内纹理特征点的个数占CU总像素个数的比例,所以 X_i 的取值区间在 $[0, 1]$; \bar{X} 表示划分后得到的4个子CU中纹理特征的均值。综上,本文一共引入了10组参数,如表2所示。

表2 纹理特征参数表

| 特征参数 | 意义 | 方差 | 意义 |
|------------|---------------|-------------|-------------------------|
| R_S | 强纹理占比例 | V_S | 4个子CU的强纹理方差 |
| R_W | 弱纹理占比例 | V_W | 4个子CU的弱纹理方差 |
| R_F | 填充占的比例 | V_F | 4个子CU的填充方差 |
| D_{Mean} | $ x - y $ 均值 | V_{MeanD} | 4个子CU的 $ x - y $ 均值的方差 |
| D_{Max} | $ x - y $ 最大值 | V_{MaxD} | 4个子CU的 $ x - y $ 最大值的方差 |

2 帧内快速算法

2.1 支持向量机线下训练

为了使用SVM训练出决策函数,需要对训练样本进行标定,以 64×64 的CU为例,经过编码后,如果CU未进行划分,标记为-1;如果CU进行了划分,则标记为1。实验样本从1080P的视频中挑选,其中划分深度 $depth = 0, 64 \times 64$ 的CU样本,一共选取5000个;划分深度 $depth = 1, 32 \times 32$ 的

CU 样本,一共选取了 10 000 个;划分深度 $depth = 2$, 16×16 的 CU 样本,一共选取了 20 000 个。评价决策函数主要使用准确率 P 和召回率 R , 计算公式如下:

$$P = \frac{\text{预测结果与样本同为正确的个数}}{\text{预测结果为正确的个数}} \quad (5)$$

$$R = \frac{\text{预测结果与样本同为正确的个数}}{\text{样本中所有正确的样本数}} \quad (6)$$

10 种纹理参数与 CU 划分的相关性如表 3 所示。在分类过程中,相关性强的参数可以增加预测的精确度,相关性弱的参数可以增加预测的召回率。

表 3 参数相关系数表

| 参数 | $depth = 0$ | $depth = 1$ | $depth = 2$ |
|-------------|-------------|-------------|-------------|
| R_S | 0.39 | 0.33 | 0.34 |
| R_W | 0.21 | 0.17 | 0.13 |
| R_F | -0.45 | -0.38 | -0.37 |
| D_{Mean} | 0.37 | 0.34 | 0.42 |
| D_{Max} | 0.45 | 0.38 | 0.28 |
| V_S | 0.25 | 0.37 | 0.33 |
| V_W | 0.13 | 0.19 | 0.12 |
| V_F | 0.24 | 0.29 | 0.31 |
| V_{MeanD} | 0.22 | 0.33 | 0.13 |
| V_{MaxD} | 0.23 | 0.25 | 0.28 |

SVM 在训练时,采用的是线性核函数,得到的决策函数如下所示:

$$(-3.032R_F + 0.000917V_F + 0.000658V_W + 0.00186V_S + 0.00358V_{MaxD} - 1.098 < 0)? -1:1 \quad (7)$$

$$(-0.750178R_W + 0.0068571V_W + 0.04106625V_{MeanD} - 1.015 < 0)? -1:1 \quad (8)$$

$$(-0.004759R_S - 0.00414R_W + 0.49857R_F - 0.00147D_{Max} - 0.0111D_{Mean} - 0.0134V_F + 0.0146V_W - 0.00986V_S - 0.00444V_{MaxD} - 4.228V_{MeanD} + 0.761 < 0)? -1:1 \quad (9)$$

$$(-0.227R_S + 1.345R_F + 0.0299D_{Max} - 0.382D_{Mean} - 0.000279V_W - 1.971V_{MeanD} + 0.816 < 0)? 1:-1 \quad (10)$$

其中函数(7)、(8)分别为深度为 0、1 时,跳过该层划分的决策函数:函数返回 -1 表示 CU 不需要进一步划分,不能跳过当前划分;1 表示 CU 需要进一步划分,可以跳过当前划分。函数(9)、(10)为深度为 1、2 时,终止下层划分的决策函数:函数返回 -1 表示 CU 不需要进一步划分,提前终止划分;1 表示 CU 需要进一步划分,不终止。

使用决策函数对测试集进行预测,预测结果的准确率和召回率如表 4 所示。

表 4 预测结果精度表

| 函数 | 深度 | 类别 | 准确率/% | 召回率/% |
|------|----|----|-------|-------|
| 跳过划分 | 0 | -1 | 15.59 | 97.35 |
| | | 1 | 99.39 | 45.17 |
| | 1 | -1 | 39.64 | 85.99 |
| | | 1 | 73.31 | 22.71 |
| 终止划分 | 1 | -1 | 71.12 | 63.20 |
| | | 1 | 79.62 | 84.89 |
| | 2 | -1 | 89.56 | 71.64 |
| | | 1 | 54.24 | 80.13 |

在训练过程中,对于跳过划分函数,为使得失真最小,尽量使 -1 的召回率达到最大。因为对于一个本来就不需要划分的 CU,如果跳过该层的方向预测和率失真代价(Rate Distortion, RD)计算直接进入下一层,那么既会增大失真,又会增加计算复杂度。对于提前终止划分函数,为了减少计算,应该尽量使 -1 的精确度和召回率达到最大。但是召回率增加的同时会导致失真的增加,本文以 1% 的率失真为上界。

在 SVM 训练过程中,训练不同划分深度的决策函数时,不同的参数对决策函数的召回率和准确率的贡献并不相同。为了选择最优的决策函数,本文采用的是对参数进行加权。训练过程中参数权值的取值范围为 $[0,1]$, 权值的步长取 0.2,遍历所有权值,取召回率或精确度最大的决策函数为最优决策函数。从训练得到的函数中可以看出,在不同深度,参数的作用并不相同,许多参数的权重为 0,这些权重为 0 的参数在判断划分时并不能起到很好的作用,因而舍去。

2.2 帧内 CU 划分优化过程

对 CU 划分的优化函数有两个:

1) 跳过 CU 划分函数。当 CU 中纹理十分复杂时,最优 CU 划分深度较大概率等于 2 或 3,因此跳过前面深度为 0 或 1 的划分,节省了帧内方向预测和率失真代价计算的耗时。具体流程如下:

a) 首先对 CU 作预处理,得到表 2 中的参数。

b) 如果划分深度等于 0,根据决策函数(7)判断:如果返回 1,则跳过帧内方向预测和率失真代价计算,划分深度 +1;如果返回 -1,不跳过。

c) 如果划分深度等于 1,根据决策函数(8)判断:如果返回 1,则跳过帧内方向预测和率失真代价计算,划分深度 +1;如果返回 -1,不跳过。

2) 终止 CU 划分函数:当划分深度达到 1 或 2 时,划分后的子 CU 尺寸变小,子 CU 内的纹理复杂度明显下降,如果子 CU 内纹理较少且分布均匀,那么就没有必要继续划分,可以提前终止。具体流程如下:

a) 首先对 CU 作预处理,得到表 2 中的参数。

b) 如果划分深度等于 1,根据决策函数(9)判断:如果返回 -1 且在划分深度为 0 时没有跳过划分,则停止划分;如果返回 1,则继续划分。

c) 如果划分深度等于 2,根据决策函数(10)判断:如果返回 -1,则停止划分;如果返回 1,则继续划分。

3 实验结果与分析

本文的快速算法基于 HM12.0^[15] 上进行测试,编码配置文件采用的是全帧内配置模式(All Intra, AI),测试使用的视频序列是 JCT-VC 提供的标准测试视频。编码性能使用三个参数来度量,分别为:1) $\Delta BR(\%)$,表示改进后算法编码比特率相比原代码增加的百分比;2) ΔP_Y (单位:dB),表示改进后编码的亮度分量相比原代码的峰值信噪比(Peak Signal-to-Noise Ratio, PSNR)损失;3) $\Delta D_T(\%)$ 表示改进算法编码时间相对于原代码减少的百分比。具体计算公式如下:

$$\Delta BR = \frac{BR_{prop} - BR_{orig}}{BR_{orig}} \times 100\% \quad (11)$$

$$\Delta P_Y = P_{prop,Y} - P_{orig,Y} \quad (12)$$

$$\Delta D_T = \frac{T_{orig} - T_{prop}}{T_{orig}} \times 100\% \quad (13)$$

其中: BR_{prop} 、 $P_{prop,Y}$ 和 T_{prop} 分别表示本文提出的快速算法编

码后的比特率、亮度分量的峰值信噪比和编码时间; BR_{orig} 、 $P_{orig,Y}$ 和 T_{orig} 分别表示原 HM12.0 编码后的比特率、亮度分量的峰值信噪比和编码时间。

文献[11]是使用灰度共生矩阵度量 CU 内的纹理分布情况,本文是根据自定义的多重纹理反映 CU 内的纹理分布,因此将本文和文献[11]中的方法对比。文献[11]数据是在 HM10.0 上得到的,由于 HM10.0 和 HM12.0 在 CU 划分算法一致,所以直接引用了原文数据。实验过程中,训练得到的决策函数与视频分辨率相关,如 $1920 \times 1080P$ 视频的决策函数用于 832×480 的视频编码时,平均加速效率 24%,率失真为 0.2%,即决策函数条件过强,需要对 832×480 分辨率重新训练决策函数。

表5 实验性能对比

| 分辨率 | 序列 | 本文方法 | | | 文献[11]方法 | | |
|-------------|-----------------|----------------|------------------------|----------------|----------------|------------------------|----------------|
| | | $\Delta BR/\%$ | $\Delta P_Y/\text{dB}$ | $\Delta DT/\%$ | $\Delta BR/\%$ | $\Delta P_Y/\text{dB}$ | $\Delta DT/\%$ |
| 2560 × 1600 | PeopleOnStreet | 1.33 | -0.03 | 44.16 | 1.07 | -0.06 | 28.75 |
| | Traffic | 0.59 | -0.04 | 44.26 | 0.90 | -0.05 | 33.51 |
| 1920 × 1080 | BasketballDrive | 1.29 | -0.02 | 54.65 | 0.42 | -0.01 | 29.12 |
| | BQTerrace | 0.07 | -0.01 | 31.26 | 0.99 | -0.06 | 29.48 |
| | Cactus | 0.73 | -0.03 | 42.32 | 0.67 | -0.03 | 28.56 |
| | Kimono | 0.84 | -0.03 | 74.45 | 1.07 | -0.04 | 30.25 |
| | ParkScene | 0.05 | -0.04 | 43.71 | 0.60 | -0.03 | 27.28 |
| 832 × 480 | BasketballDrill | 1.71 | -0.09 | 54.69 | 1.29 | -0.06 | 25.11 |
| | BQMall | 1.09 | -0.06 | 39.40 | 1.02 | -0.06 | 23.25 |
| | PartyScene | 0.35 | -0.03 | 30.33 | 1.09 | -0.08 | 24.33 |
| | RaceHorsesC | 1.12 | -0.07 | 46.82 | 1.05 | -0.08 | 23.60 |
| 416 × 240 | BasketballPass | 1.37 | -0.06 | 46.32 | 0.73 | -0.04 | 21.37 |
| | BlowingBubble | 0.47 | -0.04 | 30.30 | 0.80 | -0.05 | 20.29 |
| | BQSquare | 0.19 | -0.03 | 26.58 | 0.90 | -0.08 | 22.53 |
| | RaceHorses | 1.41 | -0.08 | 39.14 | 1.06 | -0.08 | 21.10 |
| 均值 | | 0.84 | -0.04 | 43.23 | 0.94 | -0.05 | 26.08 |

从表5可知,与 HM12.0 相比,本文提出的算法效率最高可以达到 74.45%,平均为 43.23%,比特率平均增加了 0.84%,平均 PSNR 损失为 0.04 dB;对于 1080P 视频,平均编码效率提升 49.23%,比特率平均增加了 0.59%。在几乎没有影响视频质量条件下,本文算法加速效果优于文献[11]。同时对于文献[10]中 0.43% 的率失真和 30.52% 的加速,本文算法在加速效率上有进步,但是率失真稍有增加;同时也优于文献[12]中 37.64% 的加速。

4 结语

本文基于相关性分析,揭示了多重纹理与 CU 划分尺寸之间的关系,提出一种新的 HEVC 帧内 CU 划分的快速算法。该算法利用了 CU 中的多重纹理信息,对 CU 的四叉树划分进行了剪枝。实验表明,在比特率平均增加 0.84% 的情况下,平均编码时间减少 43.23%,极大地降低了编码复杂度。此外,该算法实现简单,有利于软硬件的实现。本文在优化过程中存在一些不足,如纹理分类算法存在通用性和阈值选择的问题,训练过程中存在参数选择和个数设置的问题。下一步工作将考虑对纹理的分类进行进一步的研究,在后期的工作中将考虑参数的优化问题,进一步降低计算复杂度。

参考文献:

[1] BROSS B, HAN W J, OHM J R, *et al.* High Efficiency Video Coding (HEVC) text specification draft 8, JCTVC-J1003 [S]. Stock-

holm: Joint Collaborative Team on Video Coding, 2012: 7.

- [2] MARPE D, WIEGAND T, SULLIVAN G J. The H. 264/MPEG4 advanced video coding standard and its applications [J]. IEEE Communications Magazine, 2006, 44(8): 134-143.
- [3] OHM J, SULLIVAN G J, SCHWARZ H, *et al.* Comparison of the coding efficiency of video coding standards including High Efficiency Video Coding (HEVC) [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1669-1684.
- [4] BOSSEN F, BROSS B, SUHRING K, *et al.* HEVC complexity and implementation analysis [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1685-1696.
- [5] SULLIVAN G J, OHM J R. Overview of the High Efficiency Video Coding (HEVC) standard [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1649-1668.
- [6] LAINEMA J, BOSSEN F, HAN W J, *et al.* Intra coding of the HEVC standard [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1792-1801.
- [7] 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.
- [8] SHEN L, ZHANG Z, AN P. Fast CU size decision and mode decision algorithm for HEVC intra coding [J]. IEEE Transaction on Consumer Electronics Society, 2013, 59(1): 207-213.
- [9] XIONG J. Fast coding unit selection method for high efficiency video coding intra prediction [J]. Optical Engineering, 2013, 52(7): 071504.
- [10] XU D, LIN Q. Fast intra prediction algorithm for high efficiency video coding [J]. Journal of Computer Applications, 2014, 34(8): 2375-2379. (许东旭,林其伟.高效率视频编码帧内快速算法[J].计算机应用,2014,34(8):2375-2379.)
- [11] SHEN Y, ZHANG S, YANG C. Image texture based fast CU size selection algorithm for HEVC intra coding [C]// Proceedings of 2014 IEEE International Conference on Signal Processing, Communications and Computing. Piscataway: IEEE, 2014: 363-367.
- [12] LIU Y, LIU X, WANG P. A texture complexity based fast prediction unit size selection algorithm for HEVC intra coding [C]// Proceedings of 2014 IEEE 17th International Conference on Computational Science and Engineering. Piscataway: IEEE, 2014: 1585-1588.
- [13] CRISTIANINI N, SHAW-ETAYLOR J. An introduction to support vector machines and other kernel-based learning methods [M]. Cambridge: Cambridge University Press, 2000: 32-40.
- [14] XIA Y. Research on structural texture feature descriptor and its application [D]. Hefei: University of Science and Technology of China, 2014: 41-45. (夏瑜.基于结构的纹理特征及应用研究[D].合肥:中国科学技术大学,2014:41-51.)
- [15] CANN K M, BROSS B, HAN W J, *et al.* High Efficiency Video Coding (HEVC) test model 12 encoder description [C]// Proceedings of 12th Joint Collaborative Team on Video Coding Meeting, No. JCTVC-N1002. Stockholm: Joint Collaborative Team on Video Coding, 2013: 10-11.