doi:10.11959/j.issn.1000-436x.2016057

基于纹理平滑度的视点合成失真优化快速算法

窦环[1]，贾克斌[1]，陈锐霖[2]，萧允治[2]，吴强[1]
（1. 北京工业大学电子信息与控制工程学院，北京 100124；2. 香港理工大学电子资讯工程系，香港 九龙 999077）

摘 要：针对 3D-HEVC 中深度图编码采用的视点合成失真优化方法的高复杂度问题，提出一种基于纹理平滑度的快速算法。首先结合帧内 DC 预测特性和统计学方法分析平坦纹理图中像素规律并设定基于纹理图平坦度的跳过准则；然后在深度图编码采用视点合成失真优化方法时提前分离出纹理图平坦区域所对应的深度图区域，并终止该区域像素基于虚拟视点合成的视点合成失真计算过程。实验结果证明该算法的有效性，能在保持编码质量的同时减少大量编码时间。

关键词：3D-HEVC；深度图编码；视点合成失真优化；纹理图平坦度
中图分类号：TN919.81 **文献标识码**：A

Fast view synthesis optimization algorithm based on texture smoothness

DOU Huan[1], JIA Ke-bin[1], CHEN Rui-lin[2], XIAO Yun-zhi[2], WU Qiang[1]
(1. Department of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China;
2. Department of Electronic and Information Engineering, HongKong Polytechnic University, HongKong 999077, China)

Abstract: A fast algorithm was proposed in order to reduce the view synthesis optimization (VSO) process in depth coding for 3D-HEVC based on texture map smoothness. With the coding information derived from texture video sequences, the pixel regularity of smooth texture region was analyzed to set the skip rule using the properties of intra DC prediction and statistical methods. Then the depth regions corresponding to the flat texture map regions could be extracted and the VSO process of pixels belonging to this type of depth regions could be skipped. Experimental results show the effectiveness of the proposed algorithm.

Key words: 3D-HEVC, depth coding, view synthesis optimization, texture smoothness

1 引言

3D 视频由多台相邻摄像机从不同角度对同一场景拍摄得到，不同角度的视频间具有细微的视角差异，并通过 3D 显示技术将这些视角不同的视频分别投射给观看者双眼，从而呈现立体感[1]。相比普通的 2D 视频，3D 视频能为观众提供更为身临其境的深度视觉感受。随着近年来多媒体技术的进步，3D 视频的分辨率逐渐提高，而仅能提供 2 个视点的双目立体显示器已逐渐不再满足用户需求，能够提供更多视点数量的多视点显示器已成为目前多媒体信息产业的研究焦点。然而，视点数的增加会带来数据量的成倍增多，为了降低因视点数增多带来的高数据码率，需要采用更为有效的 3D 视频编码方案。

目前，最新的 2D 视频编码标准——高效率视频编码(HEVC, high efficiency video coding)于 2013 年被国际电信联盟 ITU-T 正式批准，该标准又被

收稿日期：2015-05-04；修回日期：2015-11-06
通信作者：贾克斌，kebinj@bjnt.edu.cn
基金项目：计算智能与智能系统北京市重点实验室基金资助项目(No.002000546615004)；北京市自然科学基金及教委重点科技基金资助项目(No.KZ201310005004)
Foundation Items: The Fund of Beijing Key Laboratory of Computational Intelligence and Intelligent System (No.002000546615004), Key Project of Beijing Municipal Education Commission(No.KZ201310005004)

称为 Recommendation H.265 或 ISO/IEC 23008-2 (MPEG-H Part 2)。鉴于其编码 2D 视频的优异性能，运动图像专家组 MPEG 和 3D 视频编码联合组 JCT-3V 开始着手 3D 视频编码的标准化工作，该标准被称为 3D-HEVC[1~3]。

3D-HEVC 标准支持多视点视频加深度(MVD, multi-view video plus depth)视频格式的编码。MVD 视频格式包括 2 个或 3 个纹理图视频和各自对应的深度图序列[4]，解码端通过利用深度图像绘制(DIBR, depth image based rendering)技术可以在原始的视点间合成新的多个虚拟视点[5]。由于深度图序列相比纹理图序列更为平滑，对深度图进行编码可以节省更多码流并降低编码复杂度。深度图序列在 3D-HEVC 中用以合成虚拟视点，因此由编码深度图造成的失真会直接导致虚拟视点的失真[6]，如果仍然使用常规的视频编码器来编码深度图视频则不能获得最佳的编码效果。由于深度图视频不会直接提供给用户观看，因此在深度图编码模式决策过程中采用的率失真优化方法需要考虑到合成的虚拟视点失真，这种技术又被称为视点合成失真优化(VSO, view synthesis optimization)。

目前，国内外众多学者都提出了有关计算合成视点失真值的方法[7~16]。这些方法根据其是否基于虚拟视点合成过程分成 2 类。第一类是基于虚拟视点合成过程的方法[7~11]，需要在 3D 视频编码端调用基于 DIBR 的虚拟视点合成过程，并根据其合成的结果计算合成视点失真值，其中最著名的方法是计算合成视点失真值变化(SVDC, synthesized view distortion change)[7]，该方法已被 3D-HEVC 视频编码测试模型 HTM 采用[17]，被用于模式选择、CU 划分以及运动矢量继承和合并中。由于引入了虚拟视点合成过程，计算 SVDC 可以获得准确的虚拟视点失真值，但也由于其基于全像素遍历的虚拟视点合成过程而带来巨大的计算复杂度。有学者提出方法对 SVDC 过程进行低复杂度优化，文献[8]提出一种快速跳过(ES, early skip)模式，通过判断编码后深度值是否发生变化来跳过深度图中不必要的虚拟视点合成过程，该方法能够节省大量编码时间并同时保证编码质量，已被 HTM 采用[17]；文献[9~11]分析了已失真深度图中像素不造成合成视点失真的条件，通过判定像素是否满足该条件来跳过不必要的编码过程。第二类是通过设定模型来计算虚拟视点失真情况的方法，不需要调用虚拟视点合成过程[12~16]，其中最有名且被 HTM 采用的方法是视点合成失真(VSD, view synthesis distortion)计算方法[12, 13]，通过分析深度图质量和合成质量间关系定义合成视点的失真值。这类方法由于不在编码端引入虚拟视点合成过程，能够大幅度降低编码时间，但是不能保证其计算得到的值能正确估计实际的合成视点失真。在 HTM 中，为了不增加计算复杂度，VSD 仅被用在帧内模式预选择和残差四叉树划分中[17]。

本文基于上述第一类中的 SVDC 方法，提出一种快速算法来加速其计算过程。首先，根据视点合成过程中深度图像素与纹理图及合成视点的纹理图像素间关系，利用位于平坦纹理区域的失真深度值不会造成合成视点失真的原理，通过结合帧内 DC 预测的特性和统计学方法分析平坦纹理图中像素规律并设定基于纹理图平坦度的像素跳过准则，然后利用该准则判定是否可以跳过深度图失真像素的虚拟视点合成过程，降低编码复杂度。

2　基于 SVDC 的 VSO 概述

常规的视频编码器采用基于拉格朗日的率失真优化 (RDO) 准则为模式选择和运动估计过程选取最优的模式类别和运动矢量。每个候选模式和候选运动矢量都需要计算率失真代价，然后选取率失真代价最小的作为最终的模式和运动矢量。率失真代价的计算如式(1)所示。

$$J = D + \lambda R \tag{1}$$

其中，J 是率失真代价，D 是失真，λ 是拉格朗日乘子，R 是编码所需比特数。失真值 D 通常由计算差值平方和(SSD, sum of squared differences)或绝对误差和(SAD, sum of absolute differences)得到。

由于对深度图进行有损编码后，已编码深度图的失真将会直接导致合成视点的失真，因此，式(1)中的失真值 D 需要同时考虑虚拟视点的失真。所以在 3D-HEVC 深度图编码方案中，失真值的计算如式(2)所示。

$$Distortion = w_1 D_{\text{depth}} + w_2 D_{\text{synth}} \tag{2}$$

其中，$Distortion$ 表示通过 VSO 过程得到的失真；w_1 和 w_2 表示 2 个权值；D_{depth} 表示深度图自身的失真，通常由计算 SSD 或 SAD 获得；D_{synth} 表示合成视点的失真，在 3D-HEVC 标准中由计算 SVDC 或 VSD 获得。

2.1 SVDC 定义

SVDC 表征了深度图失真与合成视点失真之间的关系，并引入虚拟视点合成技术来提高深度图编码的率失真优化性能。SVDC 定义了由于深度值变化造成的合成视点中的总体失真值变化情况，具体来说 SVDC 定义了 2 个合成的虚拟纹理图 S'_{T} 和 $\tilde{S}'_{\mathrm{T}}$ 之间的失真值之差 ΔD，如式(3)和图 1 所示。

$$\begin{aligned}\Delta D &= \tilde{D} - D \\ &= \sum_{(x,y)\in I}\left[\tilde{S}'_{\mathrm{T}}(x,y) - S'_{\mathrm{T,Ref}}(x,y)\right]^2 - \\ &\quad \sum_{(x,y)\in I}\left[S'_{\mathrm{T}}(x,y) - S'_{\mathrm{T,Ref}}(x,y)\right]^2 \end{aligned} \tag{3}$$

其中，(x,y) 表示像素坐标，I 表示合成视点中的所有像素集合，$S'_{\mathrm{T,Ref}}$ 表示由原始未失真纹理图和深度图合成的参考虚拟视点纹理图；S'_{T} 表示由重建后的纹理图和深度图 S_{D} 合成的纹理图，S_{D} 包含已编码块的已编码深度数据和其他未编码块的原始深度数据；$\tilde{S}'_{\mathrm{T}}$ 表示由重建后的纹理图和深度图 $\tilde{S}_{\mathrm{D}}$ 合成的纹理图，$\tilde{S}_{\mathrm{D}}$ 与 S_{D} 的区别在于 $\tilde{S}_{\mathrm{D}}$ 包含当前编码块的失真深度图信息。

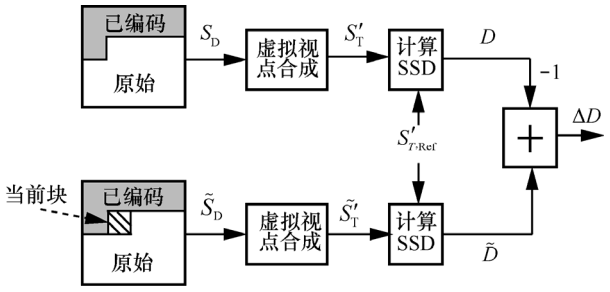

图 1 SVDC 定义

2.2 SVDC 的计算

根据上节可知，计算 SVDC 需要引入虚拟视点合成过程，但如果直接将虚拟视点合成过程放在编码端运行，会带来巨大的编码复杂度。为了避免由虚拟视点合成过程导致的编码端计算复杂度提高，HTM 引入一种方法使加入虚拟视点合成技术的 SVDC 计算过程可以简化，该方法称之为绘制器模型。绘制器模型有 3 个主要步骤，分别是初始化、局部重绘制和 SVDC 计算。

初始化过程在深度图编码前进行，原始的深度图和原始的纹理图在该阶段合成完整的虚拟视点纹理图 $S'_{\mathrm{T,Ref}}$；一旦深度图中某一块编码完毕且其最终编码的深度值已获取，局部重绘制过程开始工作以更新绘制器模型，绘制器模型将深度图中已编码块的原始数据更改为编码后数据得到新的深度图 S_{D}，并且以行为单位重新绘制因深度数据改变而影响的局部位置，得到虚拟视点 S'_{T}；在模式选择过程中，将深度图 S_{D} 中当前编码块的像素值更新为根据当前模式进行编码后得到的值来获得深度图 $\tilde{S}_{\mathrm{D}}$，并根据 $\tilde{S}_{\mathrm{D}}$ 绘制出虚拟视点 $\tilde{S}'_{\mathrm{T}}$，然后按照式(3)计算 SVDC。

3 基于纹理图平滑度的 VSO 快速算法

原始的 SVDC 计算过程需要对编码块中的每一个像素进行变换、内插、融合等操作来绘制虚拟视点，然后再利用式(3)计算 SVDC，这种全像素遍历的绘制过程会使需要进行绘制的像素数目随着 3D 视频分辨率的提高而增多，提高了编码复杂度。

实际上 SVDC 的计算过程并不需要对每个像素都进行虚拟视点绘制，如文献[8]所述的 ES 模式通过判断其深度数据是否失真来跳过一部分像素的 SVDC 计算过程以节省编码时间。除此之外，仍有虽然深度数据发生变化，但虚拟视点不产生失真的其他像素区域可以跳过。为了判定出这类像素区域来进一步提高编码效率，本文提出一种基于纹理图平滑度的 VSO 快速算法，通过结合由帧内 DC 预测和统计学方法获得的纹理图平滑区域，跳过不产生虚拟视点失真的对应深度图像素区域的 SVDC 计算过程。

3.1 算法描述

在虚拟视点合成过程中，当前纹理图中的像素利用其对应深度图中的深度值计算得到视差矢量，然后利用该视差矢量对当前纹理图中像素进行搬移以找到其在合成视点中的像素位置，当纹理图中所有像素搬移完后，虚拟视点合成完毕。对深度图进行有损编码，会导致深度图中像素值的失真，而深度图像素值失真会直接导致由其计算得到的视差矢量失真，进而导致对纹理图中像素进行搬移后在合成视点中的像素位置发生变化，使虚拟视点产生失真。但是，并非所有失真的深度图像素会导致虚拟视点失真，如果找到这类深度图像素并跳过其 SVDC 的计算过程，则能节省 VSO 的编码时间，并且保证合成视点的质量不发生变化。因此，如何找到这类深度图像素是本文提出算法的核心问题。

为了找到这类深度图像素，需要对纹理图像素的搬移过程进行分析，如图 2 所示，其中点 A 和点

B 表示在原视点中的 2 个像素位置，实线箭头表示由原始的未失真深度值计算得到的视差矢量，虚线箭头表示由编码后的失真深度值计算得到的视差矢量, 点 A'和点 B'表示点 A 和点 B 在合成视点中正确的像素位置，点 A''和点 B''表示点 A 和点 B 在合成视点中失真的像素位置。从图 2 中可以看出在原视点中 A 像素处于一个纹理平坦的背景区域，即使其深度值发生失真, 其在合成视点中的失真位置 A'' 仍然与正确位置 A'具有相似的纹理信息；而原视点中 B 像素处于一个纹理并不平坦的区域，如果其深度值发生失真，那么得到的合成视点失真像素位置 B''与 B'的纹理信息会有较大差异，必然造成合成视点失真。由此可见，如果对应的纹理图中像素处于一个相对平坦的区域，即使深度值发生失真，也不会造成合成视点中的失真，因此跳过这类深度图像素的 SVDC 过程不会造成合成视点质量的下降。一般来说，纹理平坦的像素区域通常处于视频中的背景区域或运动缓慢的区域，在视频内容中所占比例相对较大, 因此如果跳过这些像素区域的 SVDC 计算过程，可以节省较多的编码时间。

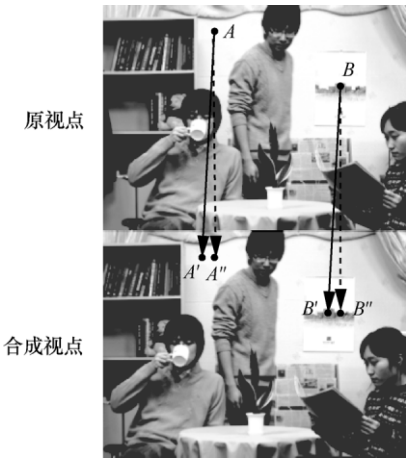

图 2　纹理图像素搬移过程

基于此，本文提出一种快速算法，结合视点合成过程中深度图像素与纹理图及合成视点的纹理图像素间关系，利用位于平坦纹理区域的失真深度值不会造成合成视点失真的原理, 在编码器使用 ES 模式之后，利用基于帧内 DC 预测特性和统计学方法的纹理图平坦度跳过符合上述条件的深度图像素区域，来进一步加速 VSO 的处理过程。首先对当前深度图编码块采用 ES 模式进行加速，若块中的某一行像素不满足 ES 模式的跳过条件，则判断其是否符合本文提出的基于纹理图平坦度的像素跳过准则规定的跳过条件，若符合则跳过该像素行中所有像素的 SVDC 计算过程，减少 VSO 的编码时间。本文提出算法的流程如图 3 所示，基于帧内 DC 预测特性和统计学方法的纹理图平坦度跳过准则的原理及设置过程如下节所述。

3.2　基于纹理图平滑度的阈值设计

本文提出算法能够减少 VSO 编码时间的前提条件，是需要找到位于对应的平坦纹理区域的深度图像素，因此本文提出算法的关键在于根据纹理图已编码信息分析纹理图的平坦度，然后根据平坦度分析结果设定深度图中像素的 VSO 跳过准则，即图 3 中虚线框中内容。

3.2.1　纹理平坦像素值规律设定

一般来说，图像的亮度值 Y 和色度值 UV 可以用来测量和判定纹理的平滑度。如果一个区域平滑，则该区域内部的所有像素的亮度值和色度值应当具有规律性。在本文中使用的判定某一行像素区域为纹理平坦区域的像素值规律如式(4)所示。

$$\forall p,q:\left|Y_p-Y_q\right|\le T,\ 0\le p,q\le n-1 \tag{4}$$

其中，Y 表示像素的亮度值，p 和 q 表示相邻像素序号，n 表示当前像素行长度，T 表示反映纹理平坦规律的阈值。因此，本文通过判断相邻 2 个像素亮度值之差是否小于等于设定的跳过阈值来判定纹理是否平坦。

3.2.2　纹理平坦判定条件

HEVC 采用多方向的帧内预测技术来消除图像的空间相关性，因此，帧内预测能够准确地表征视频图像的纹理规律。在 HEVC 中，帧内 DC 预测与 H.264/AVC 中的类似，使用当前编码单元(CU, coding unit)左边和上边 2 个方向上参考像素的均值来对当前 CU 内部所有像素进行预测, 因此帧内 DC 预测模式非常适用于图像的平坦区域编码。因此如果某一 CU 采用帧内 DC 模式作为其最优编码模式，则可以认为该 CU 纹理平坦。基于此，本文在纹理图编码完成后，记录使用帧内 DC 模式作为最优编码模式的 CU 尺寸以及该 CU 内部所有像素亮度值，然后计算该 CU 的像素亮度均值和亮度平均差值，如式(5)和式(6)所示。

$$\bar{Y}=\frac{1}{n\times n}\sum_{x}^{n}\sum_{y}^{n}Y(x,y) \tag{5}$$

$$AvgDiff=\frac{1}{n\times n}\sum_{x}^{n}\sum_{y}^{n}\left|Y(x,y)-\bar{Y}\right| \tag{6}$$

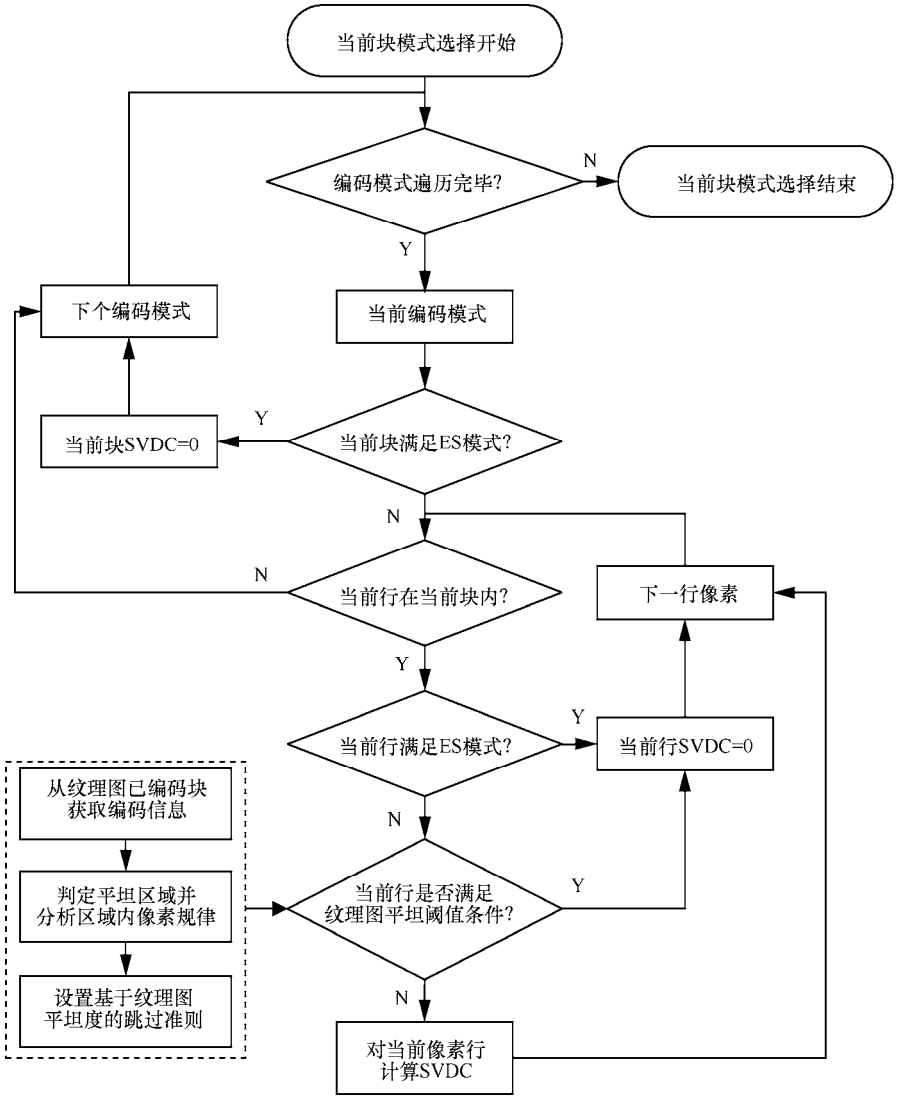

图 3 算法流程

其中，$\bar{Y}$ 表示当前 CU 内部像素平均亮度值，x 和 y 表示当前 CU 内部各像素坐标，n 表示当前 CU 宽度，*AvgDiff* 表示当前 CU 的像素亮度平均差值。

基于式(5)和式(6), 本文选择纹理分布和运动幅度较为适中的 Balloons 序列作为训练序列，训练帧数为 60 帧, 对于该序列中所有选择帧内 DC 模式作为最优模式的 CU 计算 *AvgDiff* 值, 然后统计出现次数最多的 *AvgDiff* 值和所有计算出的 *AvgDiff* 平均值，如表 1 所示。从统计结果可以看出，出现次数最多的 *AvgDiff* 值（$AvgDiff_1$）和 *AvgDiff* 平均值（$AvgDiff_2$）在 3 个视点和 4 个 *QP* 情况下的平均值分别为 2.75 和 3.42。因此，在本文提出的基于纹理平坦度的跳过准则中，跳过阈值基于表 1 的 2 个统计值结果 *AvgDiff*，选取这 2 个 *AvgDiff* 的平均值进行下取整作为最终的全局跳过阈值，如式(7)所示。

$$SkipThresholds = \lfloor \mathrm{mean}(AvgDiff_1, AvgDiff_2) \rfloor \quad (7)$$

表 1 ***AvgDiff*** **值统计**

视点序号	*QP*	出现次数最多的 *AvgDiff* 值($AvgDiff_1$)	*AvgDiff* 平均值($AvgDiff_2$)
1	25	6	5
	30	2	2
	35	2	2
	40	2	2
3	25	2	5
	30	3	4
	35	3	5
	40	3	5
5	25	2	3
	30	2	2
	35	2	2
	40	4	4
平均		2.75	3.42

为了证明本文提出的基于纹理平坦度的跳过阈值的准确度，本文对所有测试序列中采用帧内 DC 模式进行编码的各个尺寸 CU 进行统计，并对本文提出的阈值与符合条件 CU 的像素亮度均差进行比较并计算命中率，结果如表 2 所示。

表 2　　命中率统计

测试序列	视点序号	命中率/%
Newspaper	1	14.3
	3	77.2
	5	99.1
Balloons	2	60.4
	4	33.2
	6	63.6
Kendo	1	7.4
	3	12.1
	5	22.2
GTFly	1	100
	5	19.9
	9	90.9
UndoDancer	1	27.0
	5	69.2
	9	65.8
PoznanStreet	3	92.9
	4	92.1
	5	97.0

从表 2 可以看出，命中率的差距主要体现在同一序列内部不同视点间的命中率差距以及不同序列间的命中率差距。1) 同一序列内部不同视点间的命中率差距大，是因为本文统计帧内 DC 模式编码块的像素亮度均差来设置阈值。然而帧内 DC 模式在模式选择过程中所占比例并不大，所以如果在某些视点方向上选择帧内 DC 模式的 CU 数量较少并且恰好这些 CU 的像素亮度均差不等于本文设置的跳过阈值，所以造成同一序列内部某些视点的命中率较低。然而不同视点描述的是同一场景信息，如果某一视点方向命中率低，其余视点方向的命中率高，仍然可以表明选定的阈值有效。2) 不同序列间的命中率差距大，其原因是每个序列的纹理特性不同，所以每个序列帧内 DC 块的像素亮度均差结果不同。因此某些序列如 Kendo 序列其命中率不高。但是如果只选择最小的均差作为最终阈值，在编码平坦区域较多且帧内 DC 块亮度均差较大的序列时会造成跳过率不高、编码时间的节省量小，编码质量却并不能高于阈值稍高时的结果。因此在个别序列命中率并不高的情况下，阈值的选择需要具有普适性，本文选取的阈值对个别序列的命中率稍低，但对于其余多数序列命中率高，仍然表明选定的阈值有效。

因此，采用本文提出的基于纹理平坦度的跳过阈值，可以获得较好的命中率结果，同时该统计并非限定于某特定尺寸的 CU，因此可以认为本文提出的跳过阈值对各尺寸的 CU 较准确地判定纹理平滑情况。

为了证明本文提出的基于纹理平坦度的跳过阈值的加速效果，本文对所有深度图编码块内部未采用 ES 模式跳过的像素行，使用上述的跳过阈值判定出纹理平滑的像素行，并对其所占百分比进行了统计，统计结果如表 3 所示。从表 3 可以看出，利用本文提出的基于纹理图平坦度的像素跳过准则，可以判定出一半左右的纹理平坦像素行。

表 3　　纹理平坦像素行百分比统计

序列	平滑像素行百分比/%			
	QP=25	*QP*=30	*QP*=35	*QP*=40
Newspaper	38.81	47.32	54.25	62.17
Balloons	50.03	57.73	64.53	70.49
Kendo	32.73	43.00	50.25	55.47
GTFly	40.06	52.55	67.72	74.57
UndoDancer	37.51	50.70	61.86	66.79
PoznanStreet	33.67	44.19	53.57	63.86

4　实验结果

为验证本文算法效率，将 3D-HEVC 参考模型 HTM9.2[18]作为测试平台。由于 HTM 中的 ES 算法与本文提出算法协同工作，且文献[9]提出算法是改进的 ES 算法，因此依照文献[19]公布的 3DV 国际通用测试标准对 HTM 中无 ES 模式的原编码算法(ori)、文献[9]方法、ES 模式快速算法和本文提出的快速算法 (prop) 进行测试和比较。测试序列为 Balloons (1024×768)、Kendo (1024×768)、Newspaper (1024×768) 、 GTFly (1920×1088) 、 PoznanStreet (1920×1088)和 UndoDancer (1920×1088)。每个测试序列均编码 3 个视点和对应的 3 个深度图序列[19]，GOP 长度设置为 8，帧内周期设置为 24，虚拟视点

步长设置为 0.25，纹理图量化参数 *QP* 设置为 25、30、35、40，深度图量化参数 *QP* 设置为 34、39、42、45，测试序列 Balloons、Kendo、Newspaper 的测试帧数为 300 帧、帧率为 30，测试序列 GTFly、PoznanStreet、UndoDancer 的测试帧数为 250 帧、帧率为 25。所有实验在配置为 Intel(R) Xeon(R) E31230 3.2 GHz CPU, 8 GB RAM 的 PC 上独立执行。

4.1 算法复杂度分析

为评价提出算法的计算效率，本文统计了提出算法的像素行跳过率，并统计了分别采用无 ES 模式的 HTM 原编码算法 (ori)、文献[9]方法、ES 模式快速算法 (ES) 和提出的快速算法 (prop) 对各测试序列进行编码后的深度图编码时间。其中，ori 算法采用原始的逐像素虚拟视点合成过程生成虚拟视点以计算 SVDC 值的方法；文献[9]方法采用 ES 模式以及判断当前像素左右相邻像素梯度值是否相同来跳过 SVDC 计算过程；ES 模式算法通过跳过编码后深度值与编码前深度值相同的像素行来加速 SVDC 计算过程；本文提出算法在 ES 模式基础上，利用纹理平坦度设定阈值跳过平坦纹理的像素行来加速 SVDC 计算过程。各算法的深度图编码时间及时间比较结果如表 4 所示，各算法的像素行跳过率如表 5 所示。从表 4 给出的深度图编码时间比较结果可以看出，对于所有测试序列，本文提出方法的编码时间平均占 HTM 原编码时间的 57.54%，占文献[9]提出方法和 ES 模式算法编码时间的 67.97%和 85.72%。文献[9]提出的算法由于其判别条件严苛，符合跳过条件的像素行相比本文提出算法少，而又由于其判别条件中的阈值计算过程需要大量迭代过程导致编码时间增加，因此本文提

表 4 不同算法的深度图编码时间比较结果

序列	*QP*	不同方法的编码时间/s				各方法占 ori 比例/%			prop 占文献[9]比例/%	prop 占 ES 比例/%
		ori	ES	文献[9]	prop	文献[9]	ES	prop		
Balloons	25	8 769.607	5 654.49	7 201.647	5 175.079	82.12	64.48	59.01	71.86	91.52
	30	7 046.539	4 521.449	5 812.683	4 133.457	82.49	64.17	58.66	71.11	91.42
	35	6 373.172	4 157.844	5 308.588	3 729.521	83.30	65.24	58.52	70.25	89.70
	40	5 897.246	3 991.447	5 030.381	3 441.448	85.30	67.68	58.36	68.41	86.22
Kendo	25	10 428.268	6 830.559	8 648.402	5 674.497	82.93	65.50	54.41	65.61	83.08
	30	8 257.474	5 447.459	6 846.684	4 430.036	82.91	65.97	53.65	64.70	81.32
	35	7 315.150	4 875.937	6 094.086	3 881.701	83.31	66.66	53.06	63.70	79.61
	40	6 543.406	4 470.179	5 760.688	3 565.596	88.04	68.32	54.49	61.90	79.76
Newspaper	25	7 939.096	5 621.668	6 810.838	5 314.792	85.79	70.81	66.94	78.03	94.54
	30	6 658.800	4 737.368	5 739.726	4 374.851	86.20	71.14	65.70	76.22	92.35
	35	6 225.230	4 435.27	5 399.458	4 017.670	86.74	71.25	64.54	74.41	90.58
	40	5 878.834	4 233.907	5 145.484	3 716.928	87.53	72.02	63.23	72.24	87.79
GTFly	25	20 825.208	12 566.592	17 100.368	11 084.931	82.11	60.34	53.23	64.82	88.21
	30	15 750.355	9 938.751	13 619.904	8 387.293	86.47	63.10	53.25	61.58	84.39
	35	13 889.304	9 419.958	12 676.683	7 272.111	91.27	67.82	52.36	57.37	77.20
	40	12 394.614	8 524.951	11 556.152	6 321.666	93.24	68.78	51.00	54.70	74.15
PoznanStreet	25	14 286.545	9 717.326	11 584.369	8 813.954	81.09	68.02	61.69	76.08	90.70
	30	11 914.260	8 186.395	9 889.045	7 058.361	83.00	68.71	59.24	71.38	86.22
	35	11 478.971	8 087.928	9 972.184	6 571.761	86.87	70.46	57.25	65.90	81.25
	40	10 604.318	7 552.166	9 427.314	5 846.729	88.90	71.22	55.14	62.02	77.42
UndoDancer	25	17 372.154	10 704.598	13 164.029	9 940.111	75.78	61.62	57.22	75.51	92.86
	30	14 078.756	8 993.902	11 300.668	8 111.834	80.27	63.88	57.62	71.78	90.19
	35	12 285.994	8 205.154	10 390.345	7 151.651	84.57	66.78	58.21	68.83	87.16
	40	11 816.451	8 027.814	10 179.028	6 394.110	86.14	67.94	54.11	62.82	79.65
平均值						84.85	67.16	57.54	67.97	85.72

表 5　各算法的像素行跳过率

序列	*QP*	文献[9]	ES	prop
Balloons	25	64.31%	63.63%	77.70%
	30	62.28%	61.65%	79.73%
	35	58.03%	57.46%	80.47%
	40	49.55%	49.01%	80.66%
Kendo	25	62.78%	61.95%	80.97%
	30	61.10%	60.33%	93.11%
	35	56.70%	55.91%	84.27%
	40	49.97%	49.14%	84.90%
Newspaper	25	50.17%	49.63%	65.78%
	30	46.49%	46.01%	68.80%
	35	41.56%	41.00%	70.27%
	40	37.42%	36.74%	71.60%
UndoDancer	25	69.13%	67.37%	78.13%
	30	63.11%	61.36%	77.90%
	35	55.48%	53.37%	77.92%
	40	45.28%	43.45%	79.70%
GTFly	25	67.35%	66.10%	79.66%
	30	62.97%	61.30%	81.54%
	35	52.97%	50.40%	83.99%
	40	48.84%	46.08%	86.29%
PoznanStreet	25	53.13%	52.44%	70.21%
	30	48.74%	48.19%	74.48%
	35	44.59%	43.04%	78.18%
	40	40.71%	39.08%	79.80%
平均值		53.86%	52.69%	78.59%

出的方法能够比文献[9]提出的算法获得更好的编码效率，编码复杂度更低。本文提出算法是在 ES 模式的基础上，针对非 ES 模式跳过行进行进一步的跳过优化，因此相比 ES 模式算法能够降低更多的编码复杂度。从表 5 可以看出，使用本文提出算法对 6 个测试序列进行编码，其像素行跳过率的平均值为 78.59%，而对于某些纹理平坦的序列如 GTFly 序列和 Kendo 序列，由于其平坦的纹理区域相比其余序列更多，使用本文提出的算法能够检测到更多的平坦区域，因此其平均像素行跳过率能超过 80%。对所有测试序列来说，像素行跳过率随着 *QP* 值的升高而增加，这是因为编码器在计算 SVDC 时，虚拟视点合成过程使用的是重建纹理图，而随着 *QP* 值的升高编码后的重建纹理图会存在更多平滑的纹理区域，因此在高 *QP* 值情况下使用本文提出算法的像素行跳过率比低 *QP* 值情况下的高。ES 模式快速算法的平均像素行跳过率为 52.69%，文献[9]的平均像素行跳过率仅比 ES 模式提高了不到 1.2%，本文提出算法相比 ES 模式能提高 26%的平均像素行跳过率。GTFly 序列和 Kendo 序列相比其他序列所用的深度图编码时间更少，该结果也与表 5 所示的像素行跳过率结果相对应。

4.2　率失真性能分析

为验证提出算法的率失真性能，本文利用 Bjontegaard 方法[20]来比较不同算法间的绝对比特率变化量。表 6 给出了不同算法与 HTM 中的原编码算法的比较结果，其中“Base”、“Syn”和“All”分别表示原视点、合成视点以及两者平均的绝对比特率变化百分比。

表 6　Bjontegaard 方法得到的率失真性能比较结果(BDBR)

序列	文献[9]与 ori 比较/%			prop 与 ori 比较/%		
	Base	Syn	All	Base	Syn	All
Balloons	0.0	−0.2	−0.1	0.0	0.2	0.1
Kendo	−0.1	0.1	0.0	0.0	0.6	0.3
Newspaper	0.0	−0.1	0.0	0.0	1.6	1.0
GTFly	0.0	0.0	0.0	0.0	0.1	0.1
PoznanStreet	0.0	0.0	0.0	0.0	0.2	0.2
UndoDancer	0.0	0.2	0.1	0.0	0.9	0.7
平均值	0.0	0.0	0.0	0.0	0.6	0.4

从表 6 可以看出，与 HTM 的原始算法相比，本文提出算法的率失真性能虽然略低于文献[9]提出的算法，但实际上合成视点的比特率仅上升 0.6%，原视点和合成视点的平均比特率仅上升 0.4%，基本与原编码算法保持了相同的编码性能。文献[9]提出的跳过算法判别条件非常严格，能够保证低误差，因此率失真性能几乎与原算法一致，但也如 4.1 节所述的由于其跳过的像素行较少且阈值计算过程复杂导致其深度图编码时间相比本文提出算法略高。本文提出的算法利用帧内 DC 预测能检测平坦纹理的特性，跳过更多纹理平滑的像素区域，因此能够在加速编码时间的同时保证编码性能。所提算法针对 GTFly 序列能获得最好的率失真性能，这是因为 GTFly 相比其他序列含有更多平坦的背景区域，因此利用本文提出的算法可以最大程度的发挥基于纹理平坦度阈值的优势。Newspaper 序列和 UndoDancer 序列由于纹理特性更为复杂，使其率失真性能略低于其他序列。

5 结束语

本文旨在结合纹理图的平坦特性加速深度图编码中的视点合成失真优化过程，在保证视频编码质量的同时降低深度图编码复杂度。针对深度图编码中费时的视点合成失真优化部分，通过利用帧内 DC 预测的编码块纹理特性和统计学方法分析像素纹理平坦规律并设定基于纹理平坦度的跳过准则，然后分离出符合该跳过准则的深度图像素区域并跳过该区域的视点合成失真优化过程。实验结果表明提出的基于纹理平坦度的快速算法能够在保证 3D 视频及合成视点编码质量的条件下，降低深度图编码时间。

参考文献：

[1] MULLER K, SCHWARZ H, MARPE D, et al. 3D high-efficiency video coding for multi-view video and depth data[J]. IEEE Transactions on Image Processing, 2013, 22(9): 3366-3378.

[2] ISO/IEC JTC 1/SC 29/WG 11 (MPEG). High efficiency video Coding[S]. ITU-T Rec H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, 2013.

[3] SULLIVAN G J, OHM J, 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.

[4] CHEN Y, VETRO A. Next-generation 3D formats with depth map support[J]. MultiMedia IEEE, 2014, 21(2): 90-94.

[5] SMOLIC A, MULLER K, DIX K, et al. Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems[C]//IEEE International Conference on Image Processing. San Diego, CA, c2008: 2448-2451.

[6] MERKLE P, MORVAN Y, SMOLIC A, et al. The effect of depth compression on multiview rendering quality[C]//3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video. Istanbul, c2008: 245-248.

[7] TECH G, SCHWARZ H, MULLER K, et al. 3D video coding using the synthesized view distortion change[C]//Picture Coding Symposium (PCS). Krakow, c2012: 25-28.

[8] OH B, LEE J, PARK D, et al. 3D-CE8. h results on view synthesis optimization by Samsung, HHI and LG-PKU[R]. ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 1st Meeting, JCT3V-A0093, Stockholm, Sweden, 2012.

[9] MA S, WANG S, GAO W. Low complexity adaptive view synthesis optimization in HEVC based 3D video coding[J]. IEEE Transactions on Multimedia, 2014, 16(1): 266-271.

[10] MA S, WANG S, GAO W. Zero-synthesis view difference aware view synthesis optimization for HEVC based 3D video compression[J]. Visual Communications & Image Processing, IEEE, 2012, 333(6):1-6.

[11] WANG S, MA S, LIU H, et al. CE8.h: results of simplification of view synthesis optimization by detection of zero distortion change in synthesized view[R]//1st JCT3V Meeting, ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 JCT2-A0083, Stockholm, SE, 2012.

[12] OH B T, LEE J, PARK D S. Depth map coding based on synthesized view distortion function[J]. IEEE Journal of Selected Topics in Signal Processing, 2011, 5(7): 1344-1352.

[13] OH B T, OH K J. View synthesis distortion estimation for avc-and hevc-compatible 3D video coding[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2014, 24(6): 1006-1015.

[14] ZHANG Q, TIAN L, HUANG L, et al. Rendering distortion estimation model for 3D high efficiency depth coding[J]. Mathematical Problems in Engineering, 2014,(1):1-7.

[15] LI C, JIN X, DAI Q. A novel distortion model for depth coding in 3D-HEVC[C]//IEEE International Conference on Image Processing. Paris, c2014: 3228-3232.

[16] LI C, JIN X, DAI Q. An efficient distortion model configuration for depth coding in 3D-HEVC[C]//IEEE Asia Pacific Conference on Circuits and Systems. Ishigaki, c2014: 9-12.

[17] TECH G, WEGNER K, CHEN Y, et al. Test Model 8 of 3D-HEVC and MV-HEVC[R]//8th JCT3V Meeting, ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JCT3V-H1003, Valencia, ES, 2014.

[18] FRANK B, DAVID F, KARSTEN S. HEVC 3D extension software [EB/OL].https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-9.2/.

[19] MULLER K, VETRO A. Common test conditions of 3DV core experiments[R]. JCT3V meeting, JCT3VG1100, San Jose, US, 2014.

[20] BJONTEGAARDA G. Calculation of average psnr differences between rd-curves[R]. ITU-Telecommunications Standardization Secto vceg-m33, 2001.

作者简介：

窦环（1988-），女，江苏仪征人，北京工业大学博士生，主要研究方向为多视点视频编码。

贾克斌（1961-），男，河南安阳人，北京工业大学教授、博士生导师，主要研究方向为多媒体信息处理。

陈锐霖（1971-），男，中国香港人，香港理工大学教授、博士生导师，主要研究方向为信息处理、视频编码。

萧允治（1950-），男，中国香港人，香港理工大学教授、博士生导师，主要研究方向为图像和视频技术。

吴强（1972-），男，山西沁县人，北京工业大学副教授、硕士生导师，主要研究方向为多媒体信息处理。