Geometric Partitioning Mode in Versatile Video Coding: Algorithm Review and Analysis

Article *in* IEEE Transactions on Circuits and Systems for Video Technology · November 2020

DOI: 10.1109/TCSVT.2020.3040291

4 authors, including:

Han Gao
Tencent America
12 PUBLICATIONS **118** CITATIONS
SEE PROFILE

Eckehard Steinbach
Technische Universität München
484 PUBLICATIONS **8,459** CITATIONS
SEE PROFILE

Geometric Partitioning Mode in Versatile Video Coding: Algorithm Review and Analysis

Han Gao, Semih Esenlik, Elena Alshina, and Eckehard Steinbach, *Fellow, IEEE*

***Abstract*—This paper presents an overview of the geometric partitioning mode (GPM) algorithm that is a part of the most recent Versatile Video Coding (VVC) standard. The GPM algorithm aims to increase the partitioning precision of moving objects using non-rectangular and asymmetric rectangular partitions on top of the conventional rectangular block partitioning structure of VVC. Novel features of GPM contributing to the increase in coding efficiency and the reduction in encoder and decoder complexity are detailed and analyzed in this paper. Evaluated with VVC test model version 8.0 under the joint video experts team common test conditions, experimental results show that the presented GPM algorithm provides luma Bjøntegaard Delta rate reduction of $0.70\,\%$ for random access and of $1.55\,\%$ for low delay with B slices configurations, with roughly $3\,\%$ to $5\,\%$ additional encoding time and negligible decoder runtime change. Furthermore, as GPM provides more precise partitions for the boundaries of the moving objects, an improvement of visual quality is seen in GPM coded sequences.**

***Index Terms*—Geometric partitioning mode (GPM), interpicture prediction, joint video experts team (JVET), partitioning, Versatile Video Coding (VVC), video compression.**

I. Introduction

MOTION-COMPENSATED prediction (MCP) is one of the fundamental methods of removing temporal redundancies in video coding. Generally, the current picture is predicted from previously coded pictures by the use of motion compensation. The motion information used in MCP is estimated by the encoder. Only the motion information and the residual between the current picture and its predictor are transmitted in the bitstream to the decoder. To reduce the implementation difficulty and improve coding performance, block-based MCP is widely used to generate the predicted picture. In block-based MCP, the current picture is typically partitioned into non-overlapping rectangular blocks. Each block is associated with one or multiple motion vectors (MVs) that describe the displacement between the block in the current picture and the reference block(s) in previously coded picture(s). The motion estimation and MCP are performed at the block level.

A rectangular block partition is used most often for block-based MCP in video coding standards. In Advanced Video Coding (AVC) [1], [2], variable block-size tree-structured MCPs are supported. The picture is first divided into macroblocks (MBs) with 16×16 luma samples. Each MB can directly perform MCP or be further split into smaller blocks with luma sizes of 16×8, 8×16, and 8×8. If the 8×8 partition mode is selected, the sub-MB can either perform MCP as an entire block or be split into 8×4, 4×8, and 4×4 luma size blocks to perform MCP. The High Efficiency Video Coding (HEVC) [3], [4], as the successor of AVC, provides a more flexible quadtree (QT) structure for block partitions [5]. In HEVC, the picture is first divided into non-overlapping square coding tree units (CTUs) with a maximum luma sample size of 64×64. Each CTU can be recursively split into coding units (CUs) that are $2^k \times 2^k$, where $k \in \{3 \dots 6\}$. In interpicture prediction, each CU can be further split into one, two, or four prediction units (PUs) that comprise the same motion information for MCP. Eight motion partitioning modes are supported, including QT-based, symmetric and asymmetric modes, as shown in Fig. 1. Lately, Versatile Video Coding (VVC) [6], provides even more partitioning flexibility. Unlike HEVC, VVC unifies the PU and CU concepts. That is, the MCP is directly performed on the CU level. To better fit the local motion properties on a CU level, the CU is split from the CTU using a nested QT, binary-tree (BT) [7], and ternary-tree (TT) [8] structure with multiple depths. When BT and TT are used together, it is called a multi-type tree (MTT).

However, the rectangular block partition is suboptimal for MCP for natural video contents. The motion field of a picture in a video sequence is usually segmented by the boundaries of moving objects, as illustrated in Fig. 2(a) and Fig. 2(b). This is because objects typically exhibit movement relative to a static background or other moving objects, and object boundaries in natural sequences rarely adhere to rectangular block patterns. As shown in Fig. 2(c), finer block partitions are required when approximating moving object boundaries using rectangular blocks, which increases the rate overhead for signaling the partition and the prediction syntax elements for these blocks. In addition, the approximated partitioning boundary rarely follows the actual motion field boundaries. Consequently, the prediction error is higher, which increases the bitrate for residual signaling.

To increase the partition accuracy, segmentation-based partitioning approaches were integrated into the existing video coding standards structure [9]–[13]. In the segmentation-based partitioning approaches, the partitioning map is obtained by applying a segmentation algorithm on a reference block in the reference picture at the encoder. The same segmentation algorithm is applied at the decoder to avoid transformation of

Han Gao is with with Huawei Technologies, 80992 Munich, Germany, and also with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany (e-mail: han.gao@tum.de).

Semih Esenlik and Elena Alshina are with Huawei Technologies, 80992 Munich, Germany (e-mail: semih.esenlik@huawei.com; elena.alshina@huawei.com).

Eckehard Steinbach is with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany (e-mail: eckehard.steinbach@tum.de).

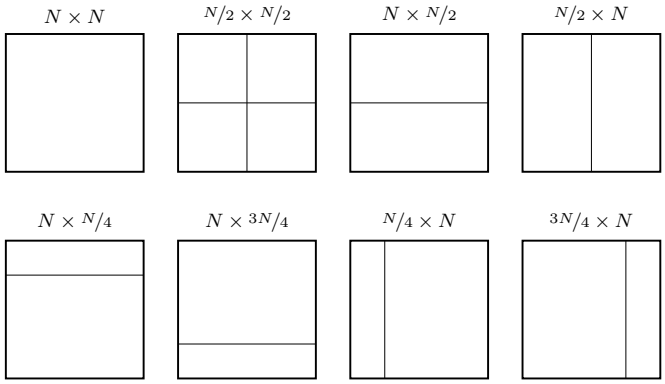
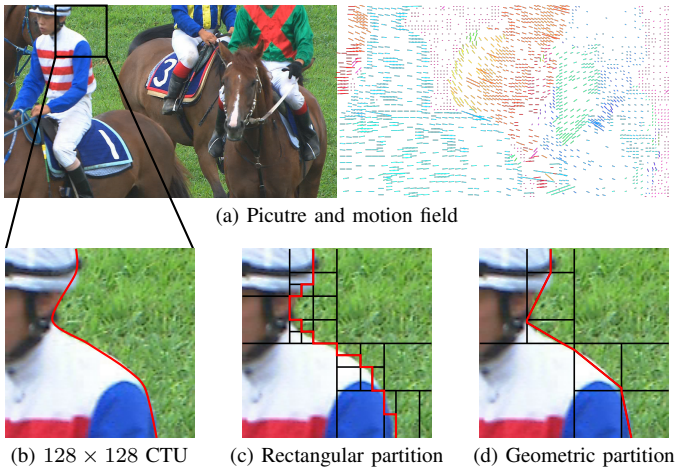


Fig. 1. Supported PU partitioning modes for MCP in HEVC; N is equal to the luma CU size; the lower row shows asymmetric partitioning modes.



(a) Picutre and motion field

(b) 128×128 CTU (c) Rectangular partition (d) Geometric partition

Fig. 2. (a) The 22nd picture of *RaceHorsesC* sequence and its motion field; (b) the second CTU of this picture and the actual motion field segmentation; (c) the rectangular block partition; (d) the geometric block partition.

the partitioning map. The current block is split into multiple partitions by the partitioning map, and each partition either individually performs MCP with its associated MV or derives the sample values from other partitions. An additional MV can be signaled to the decoder or derived at the decoder side to indicate the position of the reference block, where the segmentation algorithm is applied. Although the segmentation-based partitioning approaches approximate the actual outline of objects well, the pixel-accurate segmentation algorithms significantly increase computational complexity and lead to hardware implementation difficulty, especially at the decoder. Moreover, the segmentation-based partitioning approaches cannot efficiently process large blocks or blocks that contain multiple edges caused by a complicated texture, such as on the left side of Fig. 2(b).

Geometric partitioning schemes and their variants have been proposed in the literature and studied in core experiments (CEs) during the development of HEVC and VVC. The geometric partitioning schemes aim to increase the partition accuracy and minimize the decoder-side computational complexity and implementation difficulty. Instead of the segmentation algorithm, a set of predefined straight lines are used to geometrically split a rectangular block into two partitions, as shown in the example in Fig. 2(d). Each partition associates with its motion information and individually performs MCP. The motion information and the partition information are transmitted to the decoder. The block is directly reconstructed by the parsed motion and partition information at the decoder side. Thus, the decoder complexity increase of geometric partitioning schemes is minor.

The geometric partitioning schemes have been extensively studied in the context of AVC [11], [14]–[20], where promising bitrate reduction with relatively high encoding complexity has been reported. During the development of HEVC, a geometric partitioning scheme was proposed in [21] in response to the call for proposals (CfP). Simplified geometric partitioning schemes were also studied in several CEs [22]–[24] during HEVC development to balance the coding efficiency and encoding complexity. In [25], a unified solution was proposed for a geometric partitioning scheme that included asymmetric motion partitioning (AMP), non-rectangular motion partitioning (NRMP), and overlapped block motion compensation (OBMC). The results show a coding gain of 1.4 %, with 74 % higher encoder and 17 % higher decoder complexity as compared to the HEVC test model version 2.0 (HM 2.0) using the random access (RA) configuration. Due to the high levels of encoding complexity and the additional memory requirements at the decoder side of the proposed schemes, only AMP was adopted for the HEVC main profile [3], as illustrated in the second row of Fig. 1.

In the recent VVC standardization activity, many newly designed geometric partitioning schemes, including geometric partitioning (GEO) [26], [27], triangular prediction mode (TPM) [28], and geometric partitioning mode (GPM), described in this paper, were proposed and subsequently discussed in CEs. Both GEO and TPM were proposed as coding tools in response to the CfP for VVC [29]. GEO proposed an intercept-points-based partitioning mode with shape adaptive transformation and the combination of inter- or intrapicture predictions, whereas TPM proposed only diagonal and anti-diagonal partitioning of a rectangular block using merge-mode-based MCP. Later, TPM was adopted into the VVC specification draft 3 [30], because TPM exhibited a better trade-off of encoding complexity and coding efficiency. GPM was initially proposed in our contributions [31]–[34] in addition to VVC specification draft 5 and draft 6 as extensions and generalizations of TPM. After two rounds of CE studies [35], [36], the improved GPM algorithm [37] was adopted as a replacement for TPM in VVC specification draft 8 (draft international standard, DIS) [6] because in comparison to TPM, GPM provided better coding performance and more flexible partitions while having similar encoder and decoder complexities.

In this paper, we present the GPM algorithm that has been adopted as a coding tool in the VVC standard. Different from the aforementioned approaches, the presented GPM leads to high partitioning flexibility and coding efficiency as well as minimizes the additional encoder and decoder complexities.

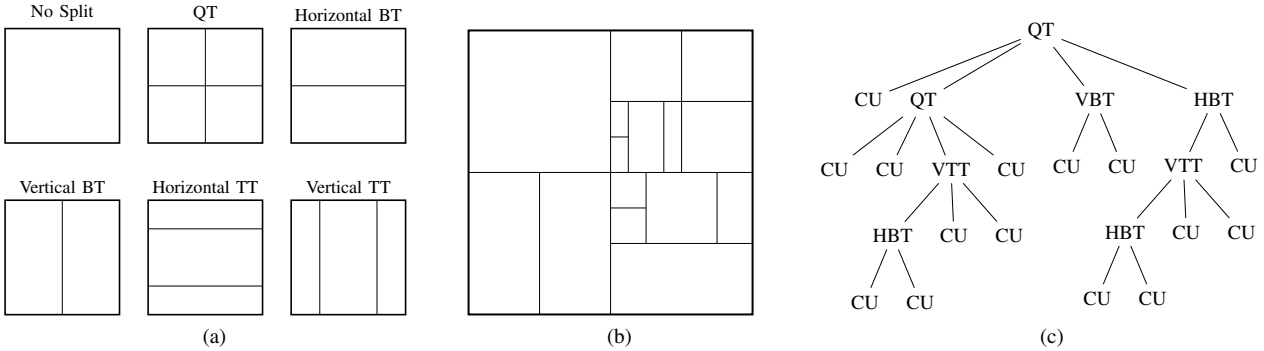


Fig. 3. (a) Allowed partitioning modes for each node in VVC; (b) partitioned CTU example; (c) corresponding partitioning tree structure of (b).

GPM splits a CU that is a leaf node of the QT plus MTT partitioning structure into two parts by a straight line. The location of the splitting boundary is derived from an angle parameter and an offset parameter that are jointly coded into the bitstream as a GPM partition index. Each resulting partition is predicted through a merge-mode-based unidirectional MCP, where the merge index of each partition is coded into the bitstream. The MCP is performed on rectangular shapes to avoid additional shape adaptive transformation. To combine the two partitions into a final predicted block, a set of blending matrices are generated on the fly based on the sample displacement to the partitioning boundary. At the decoder side, only the GPM partition index and the merge indices of both partitions are parsed from the bitstream. Because unidirectional MCP and multiplication-free blending matrices generation are used in the GPM design, the additional decoder complexity and additional memory requirements are negligible. Moreover, a new four-stage-based GPM fast encoder was designed to fit the coding architecture of VVC. GPM increased coding time only 3 % in tests with the VVC test model version 8.0 (VTM 8.0) [38] under the RA configuration of common test conditions (CTC) [39].

The rest of this paper is organized as follows. Section II briefly describes the QT plus MTT partitioning structure and the merge mode interpicture prediction in VVC, which are the fundamentals of the GPM algorithm. Section III introduces the details of the presented GPM algorithm. Our experimental results and the corresponding analysis are described and discussed in Section IV. Finally, Section V concludes this paper.

II. Block Partitioning Structure and Merge Mode Interpicture Prediction in VVC

A. QT Plus MTT Partitioning Structure

The main purpose of block partitioning in image and video coding is to group samples which can be efficiently processed and are coded together. In HEVC, pictures are first divided into CTUs, and then CTUs are recursively split into CUs using a QT structure. A CU can be further split into PUs for inter- and intrapicture prediction, or into transform units (TUs) for transform coding. To simplify the coding structure, the CU, PU, and TU concepts are unified in VVC. That is, the CTU is recursively split into CUs that are directly used as PUs and TUs for prediction and transformation. The partitioning flexibility is also extended in VVC. The QT-based CU partitioning structure is extended with an MTT partitioning structure that includes the BT- and TT-based partitions. BT symmetrically splits a node into two parts, whereas TT splits a node into three parts with a ratio of 1:2:1. Both BT and TT can be horizontally or vertically applied. Fig. 3(a) shows the possible partitioning modes for a node. BT and TT can be interleaved and applied on any node, whereas QT can only be applied on CTUs or child nodes of a QT partition. A CTU is recursively split by QT, BT, and TT, until a leaf node CU is reached. Fig. 3(b) shows a CTU partitioning example, and Fig. 3(c) illustrates the corresponding partitioning tree.

Compared with the previous video coding standards, the partitioning flexibility of VVC is significantly increased by adopting the QT plus MTT partitioning structure. However, the asymmetric motion partitioning modes of HEVC, as shown in the second row of Fig. 1, are not supported in VVC. Moreover, the QT plus MTT partitioning structure is still rectangular-block-based and thus has the aforementioned drawbacks. The presented GPM is designed to cover both non-rectangular and asymmetric partitions of an interpicture predicted CU that is a leaf node of a QT plus MTT partitioning tree.

B. Merge Mode Interpicture Prediction

Merge mode and skip mode (a merge mode without coded residuals) in HEVC [40] are efficient ways to code motion information. In interpicture prediction, the motion information, including the direction of prediction, reference picture index, and corresponding MVs are typically determined by the encoder and explicitly transmitted in the bitstream to the decoder. In merge mode, a merge candidate list, which contains the motion information from the spatial and temporal neighboring blocks of the current block, is constructed both at the encoder and decoder sides. Instead of explicitly transmitting the motion information, only the best merge candidate index and a merge mode enabling flag are coded into the bitstream. In VVC, the merge mode is inherited and enhanced by, for example, history-based motion vector prediction (HMVP) [41], merge mode with MV difference (MMVD) [42], decoder-side motion vector refinement (DMVR) [43], [44], combined inter/intrapicture prediction (CIIP) [45], and subblock-based merge mode including affine [46] and subblock-based temporal motion vector prediction (SbTMVP) [47]. These

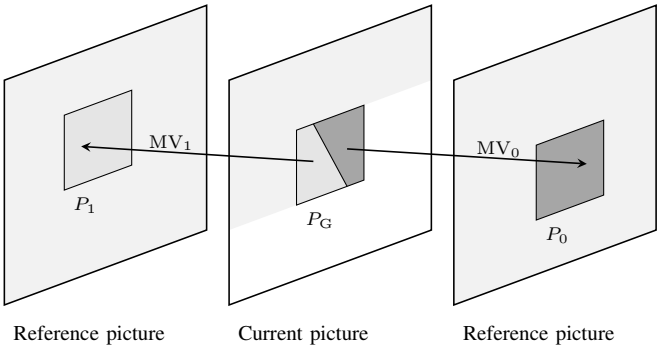


Fig. 4. The process of GPM prediction.

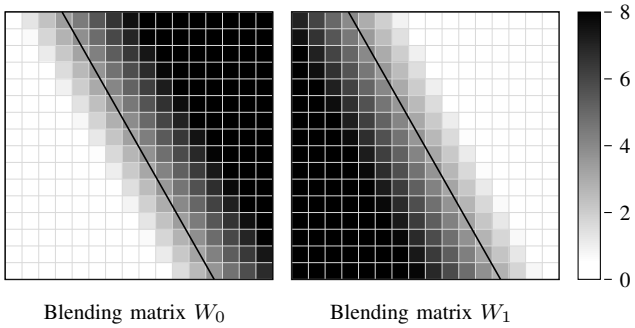


Fig. 5. Example of the GPM blending matrices.

newly adopted coding tools notably increase the precision of merge-mode-coded MVs by involving motion information of non-adjacent spatial neighboring blocks (HMVP), compensating the merge-mode-coded MVs (MMVD, DMVR), multi-hypothesizing the inter- and intrapicture prediction (CIIP) or using non-translational models on a subblock level (affine and SbTMVP).

The merge mode minimizes the transmission of the side information for motion information coding. Moreover, the encoder search for the best merge candidate is usually faster than regular motion information search because there are significantly fewer combinations searched. In the presented GPM mode, the motion information of both split parts are coded with merge mode to reduce the signaling cost of motion information and the additional encoder complexity.

III. Geometric Partitioning Mode

A. Core Algorithm of GPM in VVC

Because the intrapicture predicted CUs in VVC can use angular and non-linear prediction modes, the non-horizontal and non-vertical edges are handled well. Therefore, the presented GPM focuses on the interpicture predicted CUs. When GPM is applied to a CU, this CU is split into two parts by a straight partitioning boundary. The location of the partitioning boundary is mathematically defined by an angle parameter φ and an offset parameter ρ. These parameters are quantized and combined into a GPM partitioning index lookup table. The GPM partitioning index of the current CU is coded into the bitstream. In total, 64 partitioning modes are supported by GPM in VVC for a CU with a size of $w \times h = 2^k \times 2^l$ (in terms of luma samples) with $k, l \in \{3 \ldots 6\}$. Moreover, GPM is disabled on a CU that has an aspect ratio larger than 4:1 or smaller than 1:4, because narrow CUs rarely contain geometrically separated patterns.

The two GPM partitions contain individual motion information that are used to predict the corresponding parts in the current CU. Only a unidirectional MCP is allowed for each part of the GPM so that the required memory bandwidth for MCP in the GPM is equal to that for the regular bidirectional MCP. To simplify the motion information coding and reduce the possible combinations for the GPM, the motion information is coded with merge mode. The GPM merge candidate list is derived from the conventional merge candidate list, to ensure that only unidirectional motion information is contained.

Fig. 4 illustrates the prediction process of GPM. The right-side predicted part of the current CU with a size of $w \times h$ is predicted by MV_0 from P_0, whereas the left-side part is predicted by MV_1 from P_1. A final GPM prediction P_G is generated by performing a blending process using integer blending matrices W_0 and W_1, containing weights in the value range of 0 to 8. This can be expressed as

$$P_\mathrm{G} = (W_0 \circ P_0 + W_1 \circ P_1 + 4) \gg 3 \tag{1}$$

with

$$W_0 + W_1 = 8J_{w,h}, \tag{2}$$

where $J_{w,h}$ is a matrix of ones with a size of $w \times h$. The weights of the blending matrix depend on the displacement between the sample location and the partitioning boundary. Fig. 5 shows a set of example blending matrices. The computational complexity of blending matrices derivation is extremely low, so that these matrices can be generated on-the-fly at the decoder side.

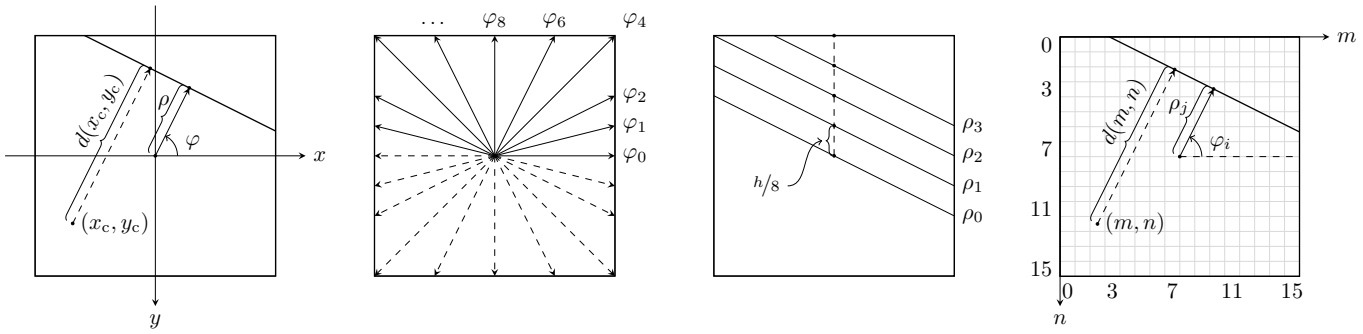
The generated GPM prediction P_G is then subtracted from the original signal to generate the residuals. The residuals are transformed, quantized, and coded into the bitstream using the regular VVC transformation, quantization, and entropy coding engines. At the decoder side, the signal is reconstructed by adding the residuals to the GPM prediction P_G. When the residuals are negligible, a skip mode is also supported by GPM. That is, the residual is dropped by the encoder, and the GPM prediction P_G is directly used by the decoder as the reconstructed signal.

B. Partitioning Boundary Definition and Quantization

1) Definition of the partitioning boundary: The partitioning boundary is defined as a straight line that is geometrically located inside the CU. The equation of this line is expressed in Hessian normal form as

$$x_\mathrm{c} \cos(\varphi) - y_\mathrm{c} \sin(\varphi) + \rho = 0, \tag{3}$$

with $(x_\mathrm{c}, y_\mathrm{c})$ defining continuous positions relating to the central position of the CU. In the example illustrated in Fig. 6(a), the angle parameter φ in (3) describes the anti-clockwise angle from the x-axis to the normal vector of the



(a) (b) (c) (d)

Fig. 6. (a) Example of a Hessian normal form-based partitioning boundary; (b) quantized angle parameter φ_i; (c) quantized offset parameter ρ_i; (d) discretization of the partitioning boundary.

TABLE I
QUANTIZED COSINE LOOKUP TABLE WITH ANGLE INDEX i

i	0	1	2	4	6	7	8	9	10	12	14	15
cosLut[*i*]	8	8	8	4	4	2	0	−2	−4	−4	−8	−8
i	16	17	18	20	22	23	24	25	26	28	30	31
cosLut[*i*]	−8	−8	−8	−4	−4	−2	0	2	4	4	8	8

TABLE II
REDUNDANT QUANTIZED OFFSETS AND CORRESPONDING REASONS

Modes	**Overlapped with**	**Numbers**
$i \geqslant 16,\ j = 0$	$i < 16,\ j = 0$	$N_\varphi/2$
$i \in \{0, 8\},\ j = 0$	BT split line	2
$i \in \{0, 8, 16, 24\},\ j = 2$	One of the TT split lines	4

partitioning boundary, whereas the offset parameter ρ in (3) is the displacement of the partitioning boundary from the origin that is defined at the center of the CU. Note that the y-axis is reversed to simplify the discretization of the partitioning boundary.

The displacement d of an arbitrary position (x_c, y_c) relative to the partitioning boundary is used to derive the blending matrices W_0 and W_1. Based on (3), the value of d is given as

$$d(x_c, y_c) = x_c \cos(\varphi) - y_c \sin(\varphi) + \rho. \tag{4}$$

As displacement is directional, the value of d can be positive or negative. The sign of d indicates which partition the position (x_c, y_c) belongs to, whereas the magnitude of d is equal to the distance of (x_c, y_c) to the partitioning boundary.

2) Quantization of angle parameter φ: To achieve a reasonable number of defined partitioning boundaries, the angle parameter φ and offset parameter ρ have to be quantized. As shown in Fig. 6(b), the angle parameter φ is quantized into 20 discrete angles φ_i, with the range of $[0, 2\pi)$ symmetrically divided. Because the diagonal or anti-diagonal partitioning boundaries are most frequently used in GPM, the quantized φ_i is therefore designed with fixed $\tan(\varphi_i)$ values that are equal to the possible aspect ratios of CU where GPM is applied. For example, if GPM is applied on a CU with an aspect ratio $w/h = 1/2$, the diagonal partitioning boundary is aligned with the line of (3) defined by $\varphi_i = \arctan(1/2)$ and $\rho = 0$. In the presented GPM algorithm, $\tan(\varphi_i)$ is limited as a value in $\{0, \pm 1/4, \pm 1/2, \pm 1, \pm 2, \infty\}$. Note that the tangent values of ± 4, which yield a near horizontal partitioning boundary, are not included, because the near horizontal partitioning of a motion field is less frequently used for natural video content.

Depending on the tangent-value-based angle parameter quantization, the $\cos(\varphi)$ in (4) is discretized into the 3-bits precision lookup table shown in Table I. The input i of $\mathrm{cosLut}[\cdot]$ is the angle index of the quantized φ_i. Note that some unused values of angle index i in Table I are skipped. Based on the trigonometric identities, the discretized lookup table of $\sin(\varphi)$ can be easily derived by

$$\mathrm{sinLut}[i] = -\mathrm{cosLut}[(i + 8)\%32]. \tag{5}$$

Thus, (4) is re-formed as

$$\begin{aligned} d(x_c, y_c) = {} & (x_c \cdot \mathrm{cosLut}[i]) \gg 3 \\ & + (y_c \cdot \mathrm{cosLut}[(i + 8)\%32]) \gg 3 + \rho. \end{aligned} \tag{6}$$

3) Quantization of offset parameter ρ: Fig. 6(c) shows an example of the offset parameter ρ quantization. The offset parameter ρ is quantized into ρ_j depending on the CU width w and height h. The offset index j is defined in $\{0 \ldots 3\}$. To avoid unequally distributed partitioning boundary offsets between different block sizes, the ρ_j is first factorized with

$$\rho_j = \rho_{x,j} \cos(\varphi_i) - \rho_{y,j} \sin(\varphi_i) \tag{7}$$

or

$$\begin{aligned} \rho_j = {} & (\rho_{x,j} \cdot \mathrm{cosLut}[i]) \gg 3 \\ & + (\rho_{y,j} \cdot \mathrm{cosLut}[(i + 8)\%32]) \gg 3. \end{aligned} \tag{8}$$

The $\rho_{x,j}$ and $\rho_{y,j}$ are then coupled with w and h using

$$\rho_{x,j} = \begin{cases} 0 & i\%16 = 8 \text{ or } (i\%16 \neq 0 \text{ and } h \geqslant w) \\ \pm j \cdot w/8 & \text{otherwise} \end{cases} \tag{9}$$

and

$$\rho_{y,j} = \begin{cases} \pm j \cdot h/8 & i\%16 = 8 \text{ or } (i\%16 \neq 0 \text{ and } h \geqslant w) \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

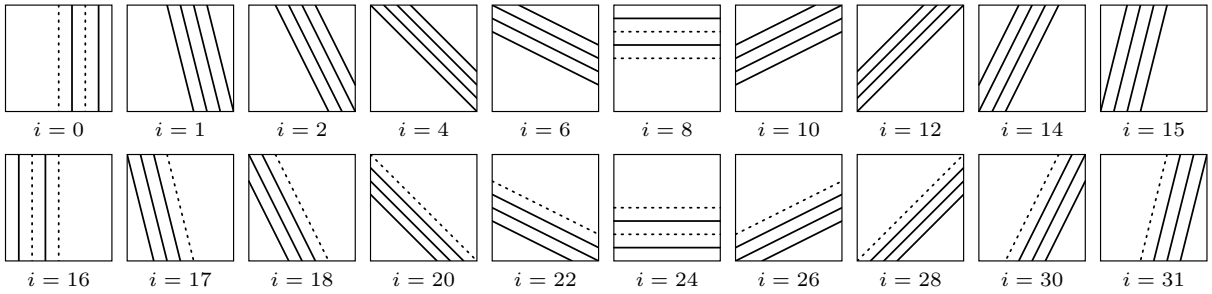


Fig. 7. Visualized examples of the 64 supported GPM partitioning modes grouped by identical angle index i; the offset indices j in each subfigure vary in the range of $\{0 \dots 3\}$; the removed redundant quantized offsets are illustrated by dotted lines.

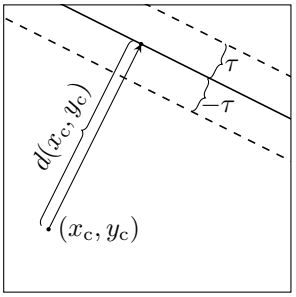


Fig. 8. Soft blending area surrounding the GPM partitioning boundary; dashed lines illustrate the width of the soft blending area.

where j is the offset index of the quantized offset ρ_j. The sign of $\rho_{x,j}$ and $\rho_{y,j}$ in (9) and (10), which indicate the offset directions, are set as positive if angle index $i < 16$, and are otherwise negative. After offset parameter quantization, (6) is re-formed as

$$d(x_c, y_c) = ((x_c + \rho_{x,j}) \cdot \mathrm{cosLut}[i]) \gg 3 + ((y_c + \rho_{y,j}) \cdot \mathrm{cosLut}[(i+8)\%32]) \gg 3. \quad (11)$$

Several redundant quantized offsets, which are listed in Table II, are removed in the presented GPM algorithm. Therefore, the total number of GPM partitioning modes is $N_{\mathrm{GPM}} = N_\varphi N_\rho - {}^{N_\varphi}\!/_2 - 2 - 4 = 64$ with $N_\varphi = 20$ and $N_\rho = 4$. The GPM partitioning modes are visualized in Fig. 7, grouped by identical angles index i. The dotted partitioning lines in Fig. 7 demonstrate the redundant modes that are not included in the presented GPM design. The 64 supported GPM partitioning modes are indexed by the syntax element *merge_gpm_partition_idx* that is coded into the bitstream using the VVC entropy coding engine.

4) Discretization of sample positions: To avoid additional interpolation in the GPM prediction process, the integer positions of P_0 and P_1 are weighted in (1). Therefore, the blending matrices W_0 and W_1 need to be derived from an integer position-based $d(m, n)$ with $m, n \in \mathbb{Z}^{\geq 0}$ instead of $d(x_c, y_c)$ with $x_c, y_c \in \mathbb{R}$. Moreover, the origin is shifted to the top-left in Fig. 6(d), because the sample locations are typically referred to the top-left sample of a CU in VVC. Thus, a fully discretized form of (11) is given by

$$d(m, n) = ((m + \rho_{x,j}) \ll 1 - w + 1) \cdot \mathrm{cosLut}[i] + ((n + \rho_{y,j}) \ll 1 - h + 1) \cdot \mathrm{cosLut}[(i+8)\%32], \quad (12)$$

which is used to generate the blending matrices W_0 and W_1 in Section III-C. As the $\mathrm{cosLut}[\cdot]$ comprise 3-bits precision and m and n are left shifted one bit, the relationship of $d(m, n)$ and $d(x_c, y_c)$ obeys

$$d(m, n) = \lfloor 16\ d(x_c, y_c) + 0.5 \rfloor. \quad (13)$$

Note that in (12), the $d(m, n)$ could be implemented without any multiplication, because values in $\mathrm{cosLut}[\cdot]$, ${}^{w}\!/_8$ in $\rho_{x,j}$, and ${}^{h}\!/_8$ in $\rho_{y,j}$ are all shift-based values (i.e., 2^k with $k \in \mathbb{Z}^{\geq 0}$). A consequence of the multiplication-free design of (12) is that the encoder and decoder complexity of GPM are extremely low.

C. Blending Matrices of GPM

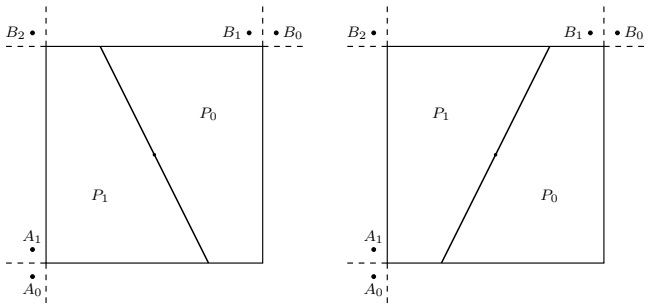
In the presented GPM algorithm, a soft blending process is applied. That is, as shown in Fig. 8, ramped weighting values in the range of $(0, 8)$ are used when the sample is located inside the soft blending area (i.e., between the dashed lines); otherwise, a full weighting value of 0 or 8 is selected. The weighting values of individual sample positions in one of the blending matrices are given by a ramp function as

$$\gamma_{x_c, y_c} = \begin{cases} 0 & d(x_c, y_c) \leqslant -\tau \\ \frac{8}{2\tau}(d(x_c, y_c) + \tau) & -\tau < d(x_c, y_c) < \tau \\ 8 & d(x_c, y_c) \geqslant \tau, \end{cases} \quad (14)$$

where τ defines the width of the blending area. In the presented GPM algorithm, τ is selected as two samples. Based on (13) with $\tau = 2$, (14) is discretized as

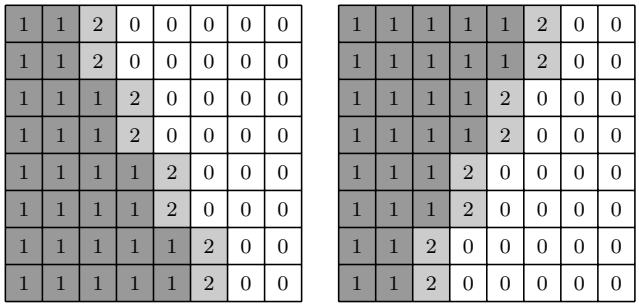
$$\gamma_{m,n} = \mathrm{Clip3}(0,\ 8,\ (d(m, n) + 32 + 4) \gg 3), \quad (15)$$

where $d(m, n)$ is calculated from (12). The weights of the other blending matrix can then be easily calculated by (2). Fig. 5 visualizes an example set of blending matrices. Note that (2) and (15) are used to generate the blending matrices of the luma component, and the chroma blending matrices are simply subsampled from the luma blending matrices based on the color format of the video sequence.



(a) $i = 2,\ j = 0$ (b) $i = 14,\ j = 0$

Fig. 9. (a) Blending matrices assignment example with $i \notin [12, 26]$; (b) blending matrices assignment example with $i \in [12, 26]$.



(a) $i = 2,\ j = 0$ (b) $i = 14,\ j = 0$

Fig. 10. (a) Motion storage map example with $i \notin [12, 26]$; (b) motion storage map example with $i \in [12, 26]$.

The blending matrices assignment is based on the following rules: If the angle index i is not in the range of $[12, 26]$, the output $\gamma_{m,n}$ of (15) is assigned to blending matrix W_0 with $m \in \{0 \dots w-1\}$ and $n \in \{0 \dots h-1\}$, whereas W_1 is calculated by (2). Otherwise, if the angle index i belongs to $[12, 26]$, $\gamma_{m,n}$ is assigned to W_1, and W_0 is derived. Fig. 9 shows examples to explain these rules. The GPM merge list, which is derived from the regular merge list (discussed in Section III-E), is constructed by prioritizing the motion information of spatial neighboring CUs in the order of B_1, A_1, B_0, A_0, and $B2$. The code word of the B_1 index is typically shortest during the entropy coding. When the motion information of prediction P_0 is predicted by B_1 (i.e., coded with the shortest code word), the signaling cost of motion information in GPM is minimized, because the merge index of P_0 is signaled prior to the merge index of P_1 (discussed in Section III-F). Thus, in the case of Fig. 9(a) with $i \notin [12, 26]$, $\gamma_{m,n}$ of (15) is assigned to W_0, whereas $\gamma_{m,n}$ is assigned to W_1 in the case of Fig. 9(b).

D. Motion Information Storage

In VVC, the motion information of a CU is stored in the cache using 4×4 samples granularity after MCP. The stored motion information is used for the MV prediction and merge list construction of the subsequently coded CU, or used to derive the boundary strength of the deblocking filter. In the regular MCP of VVC, the motion information that is actually used to predict the current CU is spanned and stored into 4×4 units. However, three types of motion information are involved in GPM. That is, P_0 and P_1 contain their own unidirectional MVs, whereas the blending area is physically predicted by motion information for both. Therefore, the motion information storage of GPM is adaptively designed based on the partitioning.

Similar to the blending matrices derivation, the motion storage type $\mathcal{T}$ for a 4×4 unit is determined by the displacement d from the center of this unit to the partitioning boundary. Here, d is calculated with (12) using the integer central position of the 4×4 unit, namely $(4m+2, 4n+2)$ with $m \in \{0 \dots w-1\}$ and $n \in \{0 \dots h-1\}$. Then, the motion storage type $\mathcal{T}$ for this 4×4 unit is given by

$$\mathcal{T}_{4m,4n} = \begin{cases} 1 - \mathcal{A} & d(4m+2, 4n+2) \leqslant -32 \\ 2 & -32 < d(4m+2, 4n+2) < 32 \\ \mathcal{A} & d(4m+2, 4n+2) \geqslant 32, \end{cases} \quad (16)$$

where $\mathcal{A}$ indicates the partition assignment that is aligned with the blending matrices alignment and is given by

$$\mathcal{A} = \begin{cases} 1 & i \in [12, 26] \\ 0 & otherwise. \end{cases} \quad (17)$$

Fig. 10 shows examples for the motion storage maps of a GPM with an angle index $i \notin [12, 16]$ and $i \in [12, 16]$. $\mathcal{T} = 0$ or $\mathcal{T} = 1$ indicates that the corresponding 4×4 units store the unidirectional motion information for predicting P_0 or P_1, whereas the units marked with $\mathcal{T} = 2$ is for sorting bidirectional motion information that is combined from the two unidirectional motions. If the motion information of P_0 and P_1 are from the same reference picture list, the combined bidirectional motion information is set to the motion information of P_1. Note that the motion storage process of GPM is usually performed in parallel with the blending process in the hardware implementation, and therefore, the $d(4m+2, 4n+2)$ is typically recalculated during the motion storage process instead of reusing the $d(m, n)$ from the blending matrices derivation.

E. GPM Merge List Derivation

The motion information of GPM are coded by merge mode. As already mentioned, only unidirectional motion information is used in each part of GPM to avoid additional memory bandwidth for MCP of GPM. However, the regular merge list possibly contains unidirectional or bidirectional merge candidates that is directly copied from neighboring CUs. Because of the bidirectional merge candidates, the regular merge list cannot be directly used as a GPM merge list.

During the development of TPM, many TPM merge list construction or derivation methods [48], [49] have been studied. Inspired by these, the GPM merge list is derived from the regular merge list by the parity of the GPM merge index. That is, for a candidate with an even value of the GPM merge index, the MV_0 from reference picture list L0 of the regular merge list with corresponding regular merge index is used as the

TABLE III
EXAMPLES OF REGULAR MERGE LIST AND GPM MERGE LIST

merge_idx	0	1	2	3	4	5
Cand.	MV_0	MV_0	–	–	MV_0	MV_0
	MV_1	MV_1	MV_1	MV_1	MV_1	MV_1

(a) Regular merge list

GPM_idx$_{0,1}$	0	1	2	3	4	5
Cand.	MV_0	MV_1	MV_1	MV_1	MV_0	MV_1

(b) GPM merge list

TABLE IV
GPM SYNTAX ELEMENTS TABLE

Syntax	**Descriptor**
merge_data() {	
...	
if(!ciip_flag){	
merge_gpm_partition_idx	ae(v)*
merge_gpm_idx0	ae(v)
if(MaxNumGpmMergeCand $>$ 2)	
merge_gpm_idx1	ae(v)
}	
...	
}	

* Context-adaptive arithmetic entropy-coded syntax element.

GPM merge candidate. If MV_0 is not available, MV_1 from the reference picture list L1 is used instead. Conversely, MV_1 is chosen as the default GPM merge candidate for an odd value of the GPM merge index, and if MV_1 is unavailable, MV_0 is used instead. Compared with other merge list construction methods studied in [48], the index parity-based method directly extracts the GPM merge candidates from the regular merge list without pruning, which minimizes the implementation complexity.

Table III shows an example of GPM merge list derivation. In the example of the regular merge list as shown in Table III(a), the merge candidates of indices 2 and 3 contain unidirectional MVs with only MV_1 available, whereas other merge candidates contain bidirectional MVs. The corresponding GPM merge list is shown in Table III(b), MV_1 is used for the candidate with index 2 because the default MV_0 is not available. The rest of the GPM merge candidates are extracted from the regular merge list based on the parity of the GPM merge index.

F. Entropy Coding

Several syntax elements of GPM are coded into the bitstream using entropy coding. In the high-level syntax (HLS), the GPM enabling flag and the maximum number of GPM merge candidates are coded in the sequence parameter set (SPS). The GPM enabling flag is coded with fixed-length code, whereas the maximum number of GPM merge candidates is signaled relative to the maximum number of regular merge candidates and coded with 0-th order exponential-Golomb code.

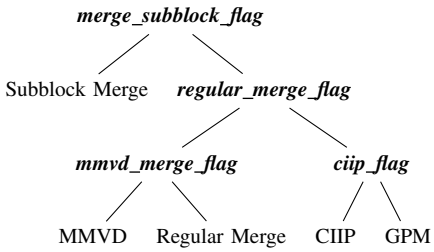


Fig. 11. Merge data syntax tree; signaled syntax elements are bold; prediction modes are on leaf nodes.

At the CU level, the GPM syntax elements are signaled inside the merge data syntax. Fig. 11 shows the syntax tree of merge data. Because the GPM mode is signaled at the bottom of the merge data syntax tree, a GPM enabling flag is not required at the CU level. When *ciip_flag* obtained is 0 and the applied conditions of GPM are fulfilled, GPM is implicitly enabled.

Table IV is the partial syntax elements table for GPM.[1] Three syntax elements, including the GPM partitioning index (*merge_gpm_partition_idx*), the merge index of P_0 (*merge_gpm_idx0*), and the merge index of P_1 (*merge_gpm_idx1*), are coded using the context-based adaptive binary arithmetic coding (CABAC) engine of VVC [6]. Because *merge_gpm_partition_idx* is roughly uniformly distributed, it is binarized with fixed-length code and coded using the bypass mode of CABAC (without context model) to simplify the coding process, whereas *merge_gpm_idx0* and *merge_gpm_idx1* are binarized with 0-th order truncated Rice code and coded using one single context model that is initialized with the same initial value of the regular merge index. Note that the used GPM merge indices $\mathrm{GPM\_idx}_{0,1}$ are derived from the signaled syntax elements by

$$\mathrm{GPM\_idx}_0 = merge\_gpm\_idx0 \tag{18}$$

and

$$\mathrm{GPM\_idx}_1 = merge\_gpm\_idx1 + (merge\_gpm\_idx1 \geqslant \mathrm{GPM\_idx}_0)\ ?\ 1\ :\ 0, \tag{19}$$

because the two merge indices are not allowed to be identical.

G. Encoder Search

Instead of a texture-based or statistic-based geometric partitioning search method such as those presented in [11], [17]–[20], a rate distortion optimization (RDO)-based partitioning encoder search is applied in the presented GPM algorithm. Compared with the texture-based search methods, the RDO-based search more easily catches the geometrically separated motion field that contains similar textures. Moreover, the RDO-based search is more versatile than statistic-based methods for different video content.

Assuming the signaled maximum number of GPM merge candidates is six, each partitioning mode has in total $6 \times 5 = 30$

[1]The complete syntax elements table and the corresponding semantics can be found in [6].

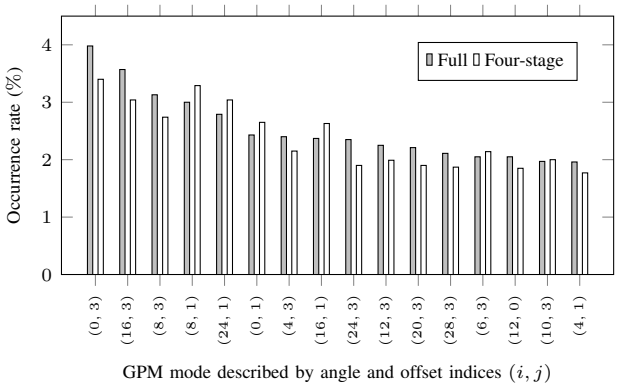


Fig. 12. 16 most commonly used GPM modes in the full search and the corresponding occurrence rate of these modes in the four-stage-base search; statistics are collected under the RA configuration with the *GPM_Tool_On* test condition.

motion information combinations because the $\mathrm{GPM\_idx}_0$ and $\mathrm{GPM\_idx}_1$ are not the same. As $N_{\mathrm{GPM}} = 64$ partitioning modes are designed in GPM, one out of $6 \times 5 \times 64 = 1920$ combined GPM candidates of partitions and motion information is to be selected by the encoder. As a comparison, up to $6 \times 5 \times 2 = 60$ combined TPM candidates are exhaustively tested with full-search-based RDO in VTM 7.0 [50]. The number of combined GPM candidates is prohibitively large for the RDO to perform the full blending and transform coding process at the encoder side. To reduce the computational complexity of the GPM encoder search, the best matching GPM candidate is determined using the following four stages:

1) Stage 1: For each unidirectional motion information candidate of the GPM merge list, full CU MCPs over the six GPM merge candidates are performed to obtain six rectangular predictors the same size as the current CU. The sum of absolute differences (SAD) in luma component between the predictors and the original signal is calculated as $\mathrm{SAD}_{\mathrm{CU},k}$ with index $k \in \{0 \ldots 5\}$. Identical entries of the GPM merge candidate list are excluded for this stage. If all motion information in the merge candidate list is identical, the entire GPM encoder search is aborted.

2) Stage 2: For each GPM partitioning mode, the parts predicted by $\mathrm{GPM\_idx}_0$ of the six rectangular predictors are masked out, using a hard blending matrix (i.e., only 0 or 8 is selected as a weighting value, depending on the sign of the displacement $d(m, n)$.). The SAD in luma of these parts are computed and denoted as $\mathrm{SAD}_{\mathrm{P0},k,l}$ with $k = 0 \ldots 5$ and $l = 0 \ldots 63$. Since the hard blending matrix (no blending area) is used, the SAD of the parts predicted by $\mathrm{GPM\_idx}_1$ is obtained by

$$\mathrm{SAD}_{\mathrm{P1},k,l} = \mathrm{SAD}_{\mathrm{CU},k} - \mathrm{SAD}_{\mathrm{P0},k,l}. \tag{20}$$

A combined rate-distortion (RD)-cost $J_{\alpha,\beta,l}$, for the partitioning mode with index l, is given by

$$J_{\alpha,\beta,l} = \mathrm{SAD}_{\mathrm{P0},\alpha,l} + \mathrm{SAD}_{\mathrm{P1},\beta,l} + \lambda(R_\alpha + R_\beta), \tag{21}$$

where α and β denote the predictor indices $\mathrm{GPM\_idx}_0$ and $\mathrm{GPM\_idx}_1$, and R_α and R_β denote the corresponding estimated motion rates. Both α and β belong to $\{0 \ldots 5\}$, but $\alpha = \beta$ cases are excluded in the encoder search. The combined GPM candidates are sorted by $J_{\alpha,\beta,l}$. Moreover, GPM candidates with $J_{\alpha,\beta,l} > \mathrm{SAD}_{\mathrm{CU},k} + \lambda R_k$ are excluded, where R_k is the estimated motion rate of the k-th GPM merge candidate.

3) Stage 3: The best 60 (or less) combined GPM candidates from stage 2 are used to conduct the soft blending process as described in Section III-C to generate P_{G} in luma component. The sum of absolute transformed differences (SATD) of luma between GPM predictor P_{G} and the original signal for each combined candidate is computed as $\mathrm{SATD}_{\mathrm{PG},l}$. The RD-cost is updated as

$$J'_{\alpha,\beta,l} = \mathrm{SATD}_{\mathrm{PG},l} + \lambda(R_\alpha + R_\beta + R_l), \tag{22}$$

where R_l represents the estimated rate of the partitioning index. The combined GPM candidates are sorted again by $J'_{\alpha,\beta,l}$. In this stage, the lowest SATD costs of previously tested coding tools, such as regular merge or affine, are used for early termination.

4) Stage 4: The corresponding chroma component of the best eight (or less) candidates from stage 3 is generated with the soft blending process. Residual transform coding (if there is a residual) and CABAC-based rate estimation are applied on these candidates to obtain the accurate rate cost R_{GPM} that includes the rate for motion, partitioning mode, and residual coding. The distortion over three components between these candidates and the original signal is measured by the sum of squared differences (SSD) as $\mathrm{SSD}_{\mathrm{PG},l}$. Finally, the GPM candidate with the lowest overall RD-cost

$$J''_{\alpha,\beta,l} = \mathrm{SSD}_{\mathrm{PG},l} + \lambda R_{\mathrm{GPM}}, \tag{23}$$

is selected as the final GPM mode.

Fig. 12 shows the 16 most commonly used GPM modes in the full search[2] and the occurrence rate of of these modes in the presented four-stage-bases search. It is observed that the occurrence rate trend of the four-stage-based search is similar to that of the full search, which indicates that the four-stage-based search can efficiently preserve the coding performance with relatively low encoder complexity. Further details of the coding performance and encoder complexity are discussed in Section IV. Note that the encoder search algorithm of GPM is not limited to the presented four-stage-based method. The statistics presented in Fig. 12 can be used as a guide of other encoder implementations.

IV. Experimental Results

A. Test Conditions

1) Software settings: The VVC reference software VTM 8.0 [38] was used as the baseline software in our experiments. Because GPM has been incorporated as a coding tool in the VVC standard since draft 8, the implementation of the presented GPM algorithm has been already included in VTM 8.0. Therefore, the *GPM_Tool_Off* and *GPM_Tool_On* tests were conducted to evaluate the coding performance and relative complexity of the GPM algorithm.

[2]The full search applies the soft blending process to all of the combined GPM candidates instead of 60 or fewer of them.

TABLE V
CODING PERFORMANCE AND RELATIVE COMPLEXITY OF THE GPM TOOL-ON AND TOOL-OFF TESTS BASED ON VTM 8.0

Sequence	*GPM_Tool_Off* (anchor: VTM 8.0)										*GPM_Tool_On* (anchor: VTM 8.0 w/o VVC tools)									
	RA (%)					**LB** (%)					**RA** (%)					**LB** (%)				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
Tango	0.65	1.36	1.18	98	100						−1.17	−2.83	−2.60	123	99					
FoodMarket	0.43	0.55	0.54	97	98						−0.77	−0.93	−0.93	124	100					
Campfire	0.21	0.15	0.63	97	99						−0.72	−0.80	−2.66	128	101					
Average A1	**0.43**	**0.69**	**0.78**	**97**	**99**						**−0.89**	**−1.52**	**−2.06**	**125**	**100**					
CatRobot	0.68	0.99	1.11	97	99						−1.66	−2.87	−2.65	128	99					
DaylightRoad	0.32	0.31	0.41	98	99						−1.26	−1.77	−1.43	127	98					
ParkRunning	0.54	0.76	0.75	96	100						−1.16	−1.39	−1.49	130	99					
Average A2	**0.51**	**0.68**	**0.76**	**97**	**100**						**−1.36**	**−2.01**	**−1.86**	**129**	**99**					
MarketPlace	0.42	0.77	1.05	97	97	0.88	1.29	0.91	95	99	−0.77	−0.83	−1.37	130	100	−1.51	−1.79	−2.04	130	97
RitualDance	0.56	0.86	1.17	97	97	1.01	1.11	1.30	95	102	−1.14	−1.88	−2.56	129	100	−1.73	−2.37	−2.92	131	99
Cactus	0.66	0.76	0.95	97	98	1.45	1.71	1.92	94	103	−1.21	−1.46	−1.75	131	102	−2.87	−3.11	−2.91	128	93
BasketballDrive	0.27	0.94	0.87	97	97	0.86	1.70	1.45	95	98	−1.01	−2.39	−1.90	130	100	−1.75	−3.45	−2.96	128	97
BQTerrace	0.30	0.28	0.31	97	98	0.59	0.64	−0.22	96	99	−1.07	−0.67	−0.56	131	101	−1.59	−0.96	−1.57	126	98
Average B	**0.44**	**0.72**	**0.87**	**97**	**97**	**0.96**	**1.29**	**1.07**	**95**	**100**	**−1.04**	**−1.45**	**−1.63**	**131**	**101**	**−1.89**	**−2.34**	**−2.48**	**129**	**97**
BasketballDrill	1.16	1.88	1.91	96	102	2.39	3.45	3.03	93	98	−2.17	−3.35	−3.48	137	104	−3.68	−5.06	−5.50	136	102
BQMall	2.13	3.43	3.46	97	100	2.53	3.54	2.82	94	101	−4.18	−5.28	−5.38	137	104	−4.76	−6.04	−5.61	137	97
PartyScene	0.71	1.07	1.23	96	99	1.14	1.34	1.73	93	100	−1.84	−1.88	−2.14	138	105	−2.87	−3.01	−3.18	142	101
RaceHorsesC	1.40	2.30	2.07	96	101	1.73	2.45	2.86	93	100	−3.04	−5.93	−5.62	138	103	−3.44	−5.29	−5.69	144	104
Average C	**1.35**	**2.17**	**2.17**	**96**	**100**	**1.95**	**2.69**	**2.61**	**93**	**100**	**−2.81**	**−4.11**	**−4.16**	**137**	**104**	**−3.69**	**−4.85**	**−5.00**	**140**	**101**
BasketballPass	0.80	2.53	1.45	96	99	1.65	3.38	3.09	93	102	−2.11	−4.29	−4.15	140	103	−3.94	−7.03	−6.24	148	109
BQSquare	0.13	0.13	1.18	97	98	0.84	1.69	0.10	95	103	−1.49	−1.10	−1.32	135	103	−3.13	−3.15	−1.42	143	106
BlowingBubbles	0.73	1.13	1.19	96	99	1.81	2.15	2.63	92	105	−2.36	−2.22	−2.50	139	101	−4.55	−4.56	−4.98	142	102
RaceHorses	1.44	2.67	2.00	95	100	1.95	3.12	3.32	92	101	−3.49	−6.50	−5.63	143	100	−4.77	−6.78	−6.24	152	106
Average D*	**0.77**	**1.62**	**1.45**	**96**	**99**	**1.56**	**2.59**	**2.28**	**93**	**103**	**−2.36**	**−3.53**	**−3.40**	**140**	**102**	**−4.10**	**−5.38**	**−4.72**	**146**	**106**
FourPeople						1.84	1.83	1.37	95	101						−4.23	−4.24	−4.74	128	97
Johnny						2.51	0.95	1.45	98	100						−6.09	−5.29	−5.32	126	104
KristenAndSara						1.71	0.62	0.97	96	101						−5.52	−4.20	−3.60	129	101
Average E						**2.02**	**1.13**	**1.26**	**96**	**101**						**−5.28**	**−4.57**	**−4.55**	**128**	**101**
BasketballDrillText	1.16	1.97	1.53	97	99	2.30	2.36	2.88	94	99	−2.22	−3.14	−3.44	105	104	−3.71	−4.05	−5.13	104	101
ArenaOfValor	0.97	1.29	0.97	98	99	1.71	2.17	2.43	95	103	−2.62	−3.19	−2.92	102	97	−3.72	−4.66	−4.34	98	95
SlideEditing	0.04	−0.03	−0.01	99	99	−0.10	0.07	−0.05	98	103	−0.59	−0.28	−0.30	109	102	0.07	−0.96	−0.84	108	100
SlideShow	0.19	0.27	0.38	98	99	−0.04	−0.32	−0.85	96	97	−0.43	−0.89	−0.92	108	102	−1.01	−2.21	−1.49	108	104
Average F*	**0.59**	**0.88**	**0.72**	**98**	**99**	**0.97**	**1.07**	**1.10**	**96**	**101**	**−1.47**	**−1.87**	**−1.89**	**106**	**101**	**−2.09**	**−2.97**	**−2.95**	**104**	**100**
Overall	**0.70**	**1.09**	**1.18**	**97**	**99**	**1.55**	**1.72**	**1.63**	**95**	**100**	**−1.54**	**−2.28**	**−2.44**	**130**	**101**	**−3.34**	**−3.73**	**−3.84**	**132**	**99**

* Not included in the overall average results based on [39].

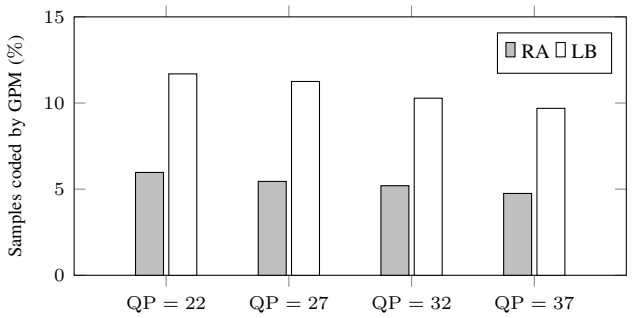


Fig. 13. Percentage of samples coded by GPM under RA and LB configurations; statistics are collected over the overlapped sequences (i.e., class B, C, D, and F) of RA and LB with the *GPM_Tool_On* test condition.

In the *GPM_Tool_Off* test, GPM was turned off in VTM 8.0, whereas the unmodified VTM 8.0 serves as an anchor. The coding loss of the *GPM_Tool_Off* test when compared with its anchor indicates the bitrate savings of the GPM algorithm under the coding architecture of VVC.

In the *GPM_Tool_On* test, the extended coding tools in VVC[3] were switched off, except for GPM, while VTM 8.0, without all the extended coding tools of VVC, was used as an anchor. The coding gain of the *GPM_Tool_On* test when compared with the corresponding anchor shows the coding performance of GPM without the interaction of other extended coding tools of VVC.

[3]The disallowed tools include CST, DQ, CCLM, MTS, ALF, AFF, SbT-MVP, AMVR, BDOF, CIIP, DMVR, MMVD, BCW, MRLP, ISP, DMVR, SBT, LMCS, SMVD, MIP, LFNST, JCCR, PROF, CCALF, and MTT. The full forms of the abbreviations of the tool names are listed in [51].

2) Encoder configurations and test sequences: The CTC defines four types of encoder configurations: all intra (AI), random access (RA), low delay with B slices (LB), and low delay with P slices (LP). Only RA and LB were tested in our experiments, because the GPM algorithm is only applied for interpicture prediction and naturally uses bidirectional prediction with the two predicted parts.

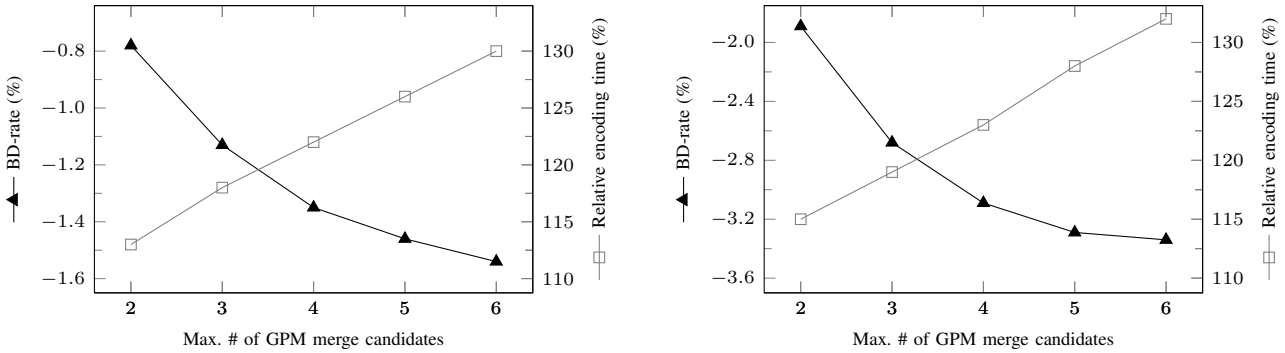
The test sequences from classes A to F with different resolutions and contents, as listed in [39], were tested. For each sequence, quantization parameter (QP) values 22, 27, 32, and 37 were used to generate the different rate points. In accordance with the CTC, class D (the lowest resolution) and class F [screen content (SCC) and mixed natural video and SCC] were not included in the overall average.

3) Coding efficiency and complexity metrics: Bjøntegaard Delta rate (BD-rate) [52], [53] savings were used for measuring the coding efficiency. The BD-rate value gives the bitrate reduction (in %) for the identical level of peak signal-to-noise ratio (PSNR). Note that negative BD-rate values indicate a bitrate saving. The complexity was measured by the relative encoding and decoding time, where the runtimes were measured on homogeneous physical machines equipped with an Intel Xeon E5-2680 v4 CPU on a Linux 64-bit platform.

B. Overall Performances

The coding performance of the GPM algorithm based on the test conditions described in IV-A are given in Table V.

For the tool-off test of the GPM, the positive BD-rate (bitrate increase) is listed, because the GPM implementation, which was turned off in the test, is part of the anchor. The tool-off test results indicate that an average luma BD-rate reduction



(a) RA configuration

(b) LB configuration

Fig. 14. Coding performance and complexity as a function of the maximum number of GPM merge candidates.

of $0.70\,\%$ for RA and $1.55\,\%$ for LB can be achieved with GPM. Up to $2.13\,\%$ RA and $2.53\,\%$ luma coding gains were achieved for the sequence *BQMall*.

The pure GPM coding performance is shown in the tool-on test results, where a negative BD-rate (bitrate reduction) was observed, because GPM was part of the test instead of the anchor. In the coding architecture without other VVC coding tools, it can be reported that overall $1.54\,\%$ and $3.34\,\%$ luma coding gains were provided by GPM for RA and LB configurations, respectively. The best performances were achieved by *BQMall* for RA with $4.18\,\%$ and *Johnny* for LB with $6.09\,\%$ coding gains.

In some sequences, for example, *BQMall*, *RaceHorsesC*, *BasketballDrill*, and *Johnny*, the coding performances were notably better than the average for both the RA and LB configurations. The particular favorable results of these sequences are mainly attributable to the fact that clearly distinctive motion field and object boundaries were contained in these sequences, where GPM was more frequently used.

Another observation is that GPM occasionally causes a tiny coding loss on the pure SCC sequences (i.e., *SlideEditing* and *SlideShow*) under the LB configuration. This is because that the pure SCC sequences usually contain motion objects with sharp edges, and the soft blending matrices can overly smooth these edges. A switchable blending process (i.e., switch between soft and hard blending matrices by content) such as that proposed in [54] can achieve better performance for the pure SCC sequences. However, this method was not adopted in the GPM design of VVC because of the implementation difficulty.

Furthermore, it is observed that the performance of GPM for LB was generally better than that for RA. Fig. 13 shows that the GPM mode GPM is more often selected in LB than RA, which yields the higher coding gain. The reason is that the RA configuration contains reference pictures from both the past and further in display order, whereas the reference pictures in LB are only from the past. Less flexible reference picture lists in LB lead to more imprecise rectangular block-based MCP, where the need for the GPM mode is greater.

C. Performance Analysis

One factor that impacts the performance of GPM is the maximum number of GPM merge candidates. Fig. 14 shows the results of BD-rate reductions and relative encoding times of GPM with different maximum numbers of GPM merge candidates based on the tool-on test condition, where VTM 8.0 without extended coding tools severed as the anchor. It was observed that the coding gain of GPM increased when a larger maximum number of GPM merge candidates was used with both the RA and LB configurations. A larger number of merge candidates yield better matched motion information for the two parts of GPM. Thus, the GPM mode is more easily selected by the encoder, which leads to larger gains. Conversely, the encoder complexity increased proportionally with the maximum number of GPM merge candidates because more combined GPM candidates were checked by the encoder. Because only the best GPM candidate is parsed at the decoder, the decoding complexity does not depend on the maximum number of GPM merge candidates. The configurable maximum number of GPM merge candidates can serve as a tool to trade off the performance and encoder complexity of GPM, and it is therefore signaled into the bitstream as described in Section III-F.

The design of the blending process also influences the performance. Table VI (first and second columns) compares the BD-rate and the relative runtime for the soft (proposed) and hard blending processes of GPM tool-on tests. The soft blending process is the blending process for generating matrices described in Section III-C, whereas the hard blending process uses a weighting value of only 0 or 8 to generate the matrices (i.e., no soft blending area). It is reported that the soft blending process provided significantly greater coding gains than the hard blending process for both the RA and LB configurations, because the soft blending matrices smooth the overly sharp partitioning boundaries that did not appear in the original signals for natural video content. Moreover, the encoder complexity of hard and soft blending processes were similar. The reason is that the soft blending process is only fully applied to a small part of the combined GPM candidates on the encoder, as described in Section III-G.

TABLE VI
COMPARING THE PRESENTED GPM (SOFT BLENDING, FOUR-STAGE-BASED ENCODER SEARCH) WITH HARD BLENDING GPM, FULL ENCODER SEARCH, AND TPM WITH TOOL-ON TEST CONDITION (ANCHOR: VTM 8.0 W/O VVC TOOLS)

Class	Presented GPM (%)					Hard Blending GPM (%)					Full Encoder Search (%)					TPM (%)				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
A1	−0.89	−1.52	−2.06	125	100	−0.23	−0.29	−0.68	122	100	−1.02	−1.60	−2.39	298	100	−0.43	−0.57	−0.91	119	100
A2	−1.36	−2.01	−1.86	129	99	−0.45	−0.76	−0.76	126	100	−1.52	−2.37	−2.20	324	99	−0.67	−0.96	−0.92	121	100
B	−1.04	−1.45	−1.63	131	101	−0.39	−0.53	−0.57	128	102	−1.20	−1.58	−1.85	349	103	−0.64	−0.72	−0.74	123	101
C	−2.81	−4.11	−4.16	137	104	−1.56	−1.99	−1.99	136	105	−2.99	−4.46	−4.65	365	110	−1.48	−1.90	−1.89	125	104
D*	−2.36	−3.53	−3.40	140	102	−1.64	−2.06	−2.09	137	103	−2.57	−3.93	−3.97	366	106	−1.41	−1.66	−1.62	125	102
F*	−1.47	−1.87	−1.89	106	101	−1.34	−1.54	−1.51	105	103	−1.61	−2.08	−2.13	170	106	−0.64	−0.74	−0.81	110	101
Avg.	**−1.54**	**−2.28**	**−2.44**	**130**	**101**	**−0.68**	**−0.91**	**−1.01**	**128**	**102**	**−1.70**	**−2.51**	**−2.77**	**337**	**103**	**−0.83**	**−1.05**	**−1.11**	**122**	**101**

(a) RA configuration

Class	Presented GPM (%)					Hard Blending GPM (%)					Full Search (%)					TPM (%)				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
B	−1.89	−2.34	−2.48	129	97	−1.04	−1.40	−1.42	128	100	−2.10	−2.51	−2.85	313	97	−1.20	−1.20	−1.21	124	99
C	−3.69	−4.85	−5.00	140	101	−2.51	−3.11	−3.23	138	105	−4.03	−5.44	−5.71	333	104	−2.17	−2.56	−2.56	126	101
D*	−4.10	−5.38	−4.72	146	106	−3.75	−3.49	−3.29	126	101	−5.54	−4.98	−4.41	387	97	−1.95	−2.06	−2.02	126	104
E	−5.28	−4.57	−4.55	128	101	−3.37	−3.94	−3.45	144	107	−4.39	−5.29	−6.41	344	112	−2.71	−3.11	−2.97	128	105
F*	−2.09	−2.97	−2.95	104	100	−1.86	−2.50	−2.58	105	104	−2.24	−3.03	−2.98	164	100	−0.92	−0.89	−0.98	112	100
Avg.	**−3.34**	**−3.73**	**−3.84**	**132**	**99**	**−2.21**	**−2.50**	**−2.49**	**131**	**102**	**−3.60**	**−4.11**	**−4.20**	**337**	**99**	**−1.71**	**−1.87**	**−1.86**	**126**	**102**

* Not included in the overall average results based on [39].

(b) LB configuration

D. Complexity Assessment

The relative runtimes for individual sequences of *GPM_Tool_Off* and *GPM_Tool_On* are listed in Table V. It is observed that roughly 3 % and 5 % additional encoding time was introduced by GPM for RA and LB under the coding architecture of VVC (*GPM_Tool_Off* test). For the *GPM_Tool_On* test, roughly 30 % and 32 % additional encoding time was reported for RA and LB, because the anchor is significantly simplified by switching off many other coding tools.

One of the main reasons for the relatively low encoding complexity is the proposed four-stage-based encoder search method for GPM, as described in Section III-G. Table VI (first and third columns) compares the complexity and performance of four-stage-based search and full search under the tool-on test condition. The relative encoding time significantly reduced from about 337 % to about 130 % by applying the four-stage based encoder search, whereas the performance of GPM only slightly changed.

The relative decoding time is reported to be roughly 100 % for all tests and configurations. One reason for this is the multiplication-free partitioning boundary and blending matrices derivation discussed in Sections III-B and III-C. Furthermore, the memory requirements and complexity of the blending MCP process of GPM are comparable with regular or weighted bidirectional predictions because only unidirectional MVs are allowed in each split part of GPM.

E. Comparison with TPM Mode

Compared with the TPM that was removed in VTM 8.0, the presented GPM provides newly designed partitioning methods, blending process, motion storage and entropy coding. To compare the performance of TPM, which was removed in VTM 8.0, with the presented GPM, we reintegrated the TPM into VTM 8.0 and tested with the tool-on test condition. Table VI (first and last columns) shows the performance and complexity of the presented GPM and TPM. It was observed that the coding gain of GPM was significantly greater than that of TPM over all tested classes, especially for LB configurations. The average BD-rate gains for GPM were 0.71 % and 1.63 % for the RA and LB configurations, when taking the results of GPM as test and the results of TPM as the anchor. This can be explained by the geometrical partitions of GPM better matching boundaries of moving objects compared to the diagonal and anti-diagonal partitions of TPM. Moreover, due to the four-stage-based encoder search and the low complexity blending process of GPM, the encoding and decoding complexity of GPM and TPM stay at the same level.

F. Visual Impression

In addition to the improved coding performance and low complexity, an additional benefit of GPM is the subjective improvements. For the sequences containing motion of rigid objects, the GPM was used predominantly for the coding of moving object boundaries, as shown by the examples in Fig. 15 (a) and (b). The additional partitions of GPM enable a precise segmentation of the video content according to the foreground and background motion. Therefore, sharp and clear edges of moving objects are visible in the coded sequences with GPM instead of the serrated and blurred moving object boundaries caused by rectangular partitions in the coded sequences without GPM. Still picture comparisons of examples with and without GPM coded sequences are shown in Fig. 15 (c), (d), (e), and (f), where GPM coded sequences had fewer coding artifacts on object boundaries for both the RA and LB configurations.

V. CONCLUSION

GPM as a new non-rectangular and asymmetric rectangular partitioning tool has been adopted in the VVC standard. In this paper, a complete system design of the GPM algorithm was presented. Several design aspects, including partitioning boundaries definition and quantization, soft blending process, motion information storage, GPM merge list construction,

(a) GPM predicted CU example of the 31st picture of *RaceHorsesC*.

(b) GPM predicted CU example of the 92nd picture of *BQMall*.

(c) VTM 8.0 without GPM

(d) VTM 8.0 (GPM included)

(e) VTM 8.0 without GPM

(f) VTM 8.0 (GPM included)

Fig. 15. Visual examples of GPM based on VTM 8.0; (a), (c), (d) coded at QP 37 in RA configuration; (b), (e), (f) coded at QP 37 in LB configuration.

entropy coding, and enhanced encoder search speed were discussed. Experimental results revealed that the GPM provided around $0.70\,\%$ and $1.55\,\%$, on average, luma BD-rate reduction in RA and LB configurations for VTM 8.0. Due to the optimized encoder search, merge-mode-based motion information coding, and low complexity blending process, the encoding runtime increased roughly $3\,\%$ to $5\,\%$, depending on the configuration, whereas the decoding runtime change was negligible. Moreover, the presented GPM algorithm more precisely partitioned the moving object boundaries, and therefore, the visual quality of the coded sequences with GPM was improved.

REFERENCES

[1] *Advanced video coding for generic audiovisual services*, ITU-T. Rec. H.264 and ISO/IEC 14496-10, version 1, 2003.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] *High efficiency video coding*, ITU-T. Rec. H.265 and ISO/IEC 23008-2:2013, version 1, 2013.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] I. Kim, J. Min, T. Lee, W. Han, and J. Park, "Block partitioning structure in the hevc standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1697–1706, Dec. 2012.

[6] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, "Versatile video coding (Draft 8)," JVET, Brussels, BE, 17th meeting, document JVET-Q2001, Jan. 2020.

[7] J. An, H. Huang, K. Zhang, Y.-W. Huang, and S. Lei, "Quadtree plus binary tree structure integration with JEM tools," JVET, San Diego, CA, USA, 2nd meeting, document JVET-B0023, Feb. 2016.

[8] X. Li *et al.*, "Multi-type-tree," JVET, Chengdu, CN, 4th meeting, document JVET-D0117, Oct. 2016.

[9] J. Chen, S. Lee, K.-H. Lee, and W.-J. Han, "Object boundary based motion partition for video coding," in *2007 Picture Coding Symp. (PCS)*, Lisbon, Portugal, 2007, pp. 1–5.

[10] J. H. Kim, A. Ortega, P. Yin, P. Pandit, and C. Gomila, "Motion compensation based on implicit block segmentation," in *2008 15th IEEE Int. Conf. Image Process.*, San Diego, CA, USA, 2008, pp. 2452–2455.

[11] Q. Wang, X. Ji, M. Sun, G. J. Sullivan, J. Li, and Q. Dai, "Complexity reduction and performance improvement for geometry partitioning in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 338–352, Feb. 2013.

[12] M. Bläser, C. Heithausen, and M. Wien, "Segmentation-based partitioning for motion compensated prediction in video coding," in *2016 Picture Coding Symp. (PCS)*, Nuremberg, Germany, 2016, pp. 1–5.

[13] Z. Wang, S. Wang, X. Zhang, S. Wang, and S. Ma, "Three-zone segmentation-based motion compensation for video compression," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 5091–5104, Oct. 2019.

[14] S. Kondo and H. Sasai, "A motion compensation technique using sliced blocks in hybrid video coding," in *2005 IEEE Int. Conf. Image Process.*, vol. 2, Genova, Italy, 2005, pp. II–305.

[15] E. M. Hung, R. L. De Queiroz, and D. Mukherjee, "On macroblock partition for motion compensation," in *2006 IEEE Int. Conf. Image Process.*, Atlanta, GA, USA, 2006, pp. 1697–1700.

[16] O. D. Escoda, P. Yin, C. Dai, and X. Li, "Geometry-adaptive block partitioning for video coding," in *2007 IEEE Int. Conf. Acou., Speech, Signal Process.*, vol. 1, Honolulu, HI, USA, 2007, pp. I–657–I–660.

[17] R. U. Ferreira, E. M. Hung, R. L. de Queiroz, and D. Mukherjee, "Efficiency improvements for a geometric-partition-based video coder," in *2009 16th IEEE Int. Conf. Image Process.*, Cairo, Egypt, 2009, pp. 1009–1012.

[18] L. Guo, P. Yin, Y. Zheng, X. Lu, Q. Xu, and J. Sole, "Simplified

geometry-adaptive block partitioning for video coding," in *2010 IEEE Int. Conf. Image Process.*, Hong Kong, 2010, pp. 965–968.

[19] P. Bordes, E. Francois, and D. Thoreau, "Fast encoding algorithms for geometry-adaptive block partitioning," in *2011 18th IEEE Int. Conf. Image Process.*, Brussels, Belgium, 2011, pp. 1205–1208.

[20] A. A. Muhit, M. R. Pickering, M. R. Frater, and J. F. Arnold, "Video coding using fast geometry-adaptive partitioning and an elastic motion model," *J. Vis. Commun. Image Representation*, vol. 23, no. 1, pp. 31–41, Jan. 2012.

[21] M. Karczewicz *et al.*, "A hybrid video coder based on extended macroblock sizes, improved interpolation, and flexible motion representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1698–1708, Dec. 2010.

[22] E. Francois, X. Zheng, and P. Chen, "CE2: Summary of core experiment 2 on flexible motion partitioning," JCT-VC, Daegu, KR, 4th meeting, document JCTVC-D229, Jan. 2011.

[23] X. Zheng, P. Bordes, P. Chen, and I.-K. Kim, "CE2: Summary of core experiment 2 on flexible motion partitioning," JCT-VC, Geneva, CH, 5th meeting, document JCTVC-E022, Mar. 2011.

[24] X. Zheng, P. Bordes, P. Chen, and I.-K. Kim, "CE2: Summary of core experiment 2 on motion partitioning and OBMC," JCT-VC, Torino, IT, 6th meeting, document JCTVC-F022, Jul. 2011.

[25] P. Bordes, P. Chen, I.-K. Kim, L. Guo, H. Yu, and X. Zheng, "CE2: Unified solution of flexible motion partitioning," JCT-VC, Geneva, CH, 5th meeting, document JCTVC-E374, Mar. 2011.

[26] M. Bläser, J. Schneider, J. Sauer, and M. Wien, "Geometry-based partitioning for predictive video coding with transform adaptation," in *2018 Picture Coding Symp. (PCS)*, San Francisco, CA, USA, 2018, pp. 134–138.

[27] M. Bläser, J. Sauer, and M. Wien, "Description of SDR and 360° video coding technology proposal by RWTH Aachen University," JVET, San Diego, CA, USA, 10th meeting, document JVET-J0023, Apr. 2018.

[28] T. Toma *et al.*, "Description of SDR video coding technology proposal by Panasonic," JVET, San Diego, CA, USA, 10th meeting, document JVET-J0020, Apr. 2018.

[29] B. Bross *et al.*, "General video coding technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1226–1240, May 2020.

[30] B. Bross, J. Chen, and S. Liu, "Versatile Video Coding (Draft 3)," JVET, Macao, CN, 12th meeting, document JVET-M1001, Oct. 2018.

[31] S. Esenlik *et al.*, "Non-CE4: Geometrical partitioning for inter blocks," JVET, Gothenburg, SE, 15th meeting, document JVET-O0489, Jul. 2019.

[32] M. Bläser *et al.*, "Low-complexity geometric inter-prediction for versatile video coding," in *2019 Picture Coding Symp. (PCS)*, Ningbo, China, 2019, pp. 1–5.

[33] H. Gao *et al.*, "Simplified GEO without multiplication and minimum blending mask storage (harmonization of JVET-P0107, JVET-P0264 and JVET-P0304)," JVET, Geneva, CH, 16th meeting, document JVET-P0884, Oct. 2019.

[34] H. Gao *et al.*, "Advanced geometric-based inter prediction for versatile video coding," in *2020 Data Compression Conf. (DCC)*, Snowbird, UT, USA, 2020, pp. 93–102.

[35] C.-C. Chen, H. Yang, and X. Xiu, "CE4: Summary report on inter prediction," JVET, Geneva, CH, 16th meeting, document JVET-P0024, Oct. 2019.

[36] C.-C. Chen, R.-L. Liao, X. Xiu, and H. Yang, "CE4: Summary report on inter prediction with geometric partitioning," JVET, Brussels, BE, 17th meeting, document JVET-Q0024, Jan. 2020.

[37] H. Gao *et al.*, "Integrated Text for GEO," JVET, Brussels, BE, 17th meeting, document JVET-Q0806, Jan. 2020.

[38] (2020) *VVC reference software VTM-8.0*. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-8.0/

[39] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, "JVET common test conditions and software reference configurations for SDR video," JVET, Geneva, CH, 14th meeting, document JVET-N1010, Jan. 2019.

[40] P. Helle *et al.*, "Block merging for quadtree-based partitioning in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1720–1731, Dec. 2012.

[41] L. Zhang *et al.*, "History-based motion vector prediction in versatile video coding," in *2019 Data Compression Conf. (DCC)*, Snowbird, UT, USA, 2019, pp. 43–52.

[42] S. Jeong, M. W. Park, and C. Kim, "CE4 ultimate motion vector expression in JVET-J0024 (Test 4.2.9)," JVET, Ljubljana, SI, 11th meeting, document JVET-K0115, Jul. 2018.

[43] H. Gao, S. Esenlik, Z. Zhao, E. Steinbach, and J. Chen, "Decoder side motion vector refinement for versatile video coding," in *2019 IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP)*, Kuala Lumpur, Malaysia, 2019, pp. 1–6.

[44] H. Gao, S. Esenlik, Z. Zhao, E. Steinbach, and J. Chen, "Low complexity decoder side motion vector refinement for VVC," in *2019 Picture Coding Symp. (PCS)*, Ningbo, China, 2019, pp. 1–5.

[45] M.-S. Chiang, C.-W. Hsu, Y.-W. Huang, and S.-M. Lei, "CE10.1: Combined and multi-hypothesis prediction," JVET, Ljubljana, SI, 11th meeting, document JVET-K0257, Jul. 2018.

[46] K. Zhang, Y.-W. Chen, L. Zhang, W.-J. Chien, and M. Karczewicz, "An improved framework of affine motion compensation in video coding," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1456–1469, Mar. 2019.

[47] W.-J. Chien, Y. Chen, J. Chen, L. Zhang, M. Karczewicz, and X. Li, "Sub-block motion derivation for merge mode in HEVC," in *SPIE Appl. of Digit. Image Process. XXXIX*, vol. 9971, San Diego, CA, USA, 2016, pp. 482 – 488.

[48] H. Yang, G. Li, and K. Zhang, "CE4: Summary report on inter prediction and motion vector coding," JVET, Geneva, CH, 14th meeting, document JVET-N0024, Jan. 2019.

[49] X. Wang, Y.-W. Chen, X. Xiu, and M. T.-C., "CE4-related: An improved method for triangle merge list construction," JVET, Geneva, CH, 14th meeting, document JVET-N0340, Jan. 2019.

[50] (2019) *VVC reference software VTM-7.0*. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-7.0/

[51] W.-J. Chen and J. Boyce, "Methodology and reporting template for coding tool testing," JVET, Brussels, BE, 17th meeting, document JVET-Q2005, Jan. 2020.

[52] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," ITU-T SG16/Q6, Austin, TX, USA, 13th meeting, document VCEG-M33, Apr. 2001.

[53] K. Andersson, F. Bossen, J.-R. Ohm, A. Segall, R. Sjöberg, J. Ström, and G. J. Sullivan, "Summary information on BD-rate experiment evaluation practices," JVET, Brussels, BE, 17th meeting, document JVET-Q2016, Jan. 2020.

[54] H. Gao *et al.*, "CE4-4: Geometric inter prediction with adaptive blending for SCC," JVET, Brussels, BE, 17th meeting, document JVET-Q0060, Jan. 2020.

Han Gao received the B.Sc. degree in electrical engineering from the Xidian University, Xi'an, China, in 2012 and the M.Sc. degree in electrical engineering and information technology the Technical University of Munich (TUM), Munich, Germany, in 2015, where he is currently pursuing the Ph.D. degree. In 2016, he joined Chair of Media Technology at TUM as well as Audiovisual Technology Laboratory at Huawei Technologies, Munich, Germany. Since 2016, he has been actively involved in the development of the VVC standard that was jointly issued by ITU-T VCEG and ISO/IEC MPEG. His research focuses on image and video processing, traditional and neural network-based video compression and coding.

Semih Esenlik received his B.S. degree on Electrical and Electronics Engineering from Bogazici University of Istanbul in 2008 and his M.S. degree from Technical University of Munich on Telecommunications Engineering in 2010. From 2010 to 2013 he worked as Research Engineer in Panasonic R&D Center of Germany, in the Multimedia Coding and Standardization Group. Between 2013 and 2016, he worked as technical product manager at Turkcell in Turkey. In 2016 he started working in Huawei R&D Center of Germany as principal research engineer and shortly after became team leader of video coding standardization team. He has actively participated in the development of HEVC, EVC and VVC codecs, is among the main contributors of decoder side motion vector refinement and geometric partitioning mode technologies in VVC. He was the primary chair of core experiment on decoder side motion vector derivation and refinement during the development of VVC.

Elena Alshina received the M.S. in physics from Lomonosov Moscow State University (MSU) in 1995. She continued her post-graduate study at the Faculty of Computational Mathematics and Cybernetics (MSU) and received Ph.D. in mathematical modeling in 1998. Since that time, she worked for Institute for Mathematical Modeling Russian Academy of Science performing research on high accuracy numerical methods and new finite-difference scheme development. Simultaneously, she worked as associate professor for Moscow Institute of Electronic Technology.

In 2006 Elena joined Multimedia Platform Laboratory of Samsung, Digital Media and Communications Research and Development Center. She is an active participant of international standardization (MPEG/VCEG/JCTVC/JVET) since 2008. She is currently the Chief Video Scientist in Huawei Munich since May, 2018. She is in charge of Machine Oriented Video, AI-based codec development, and international standardization.

Eckehard Steinbach (Fellow, IEEE) received the degree in electrical engineering from the University of Karlsruhe, Germany, the University of Essex, Great-Britain, and ESIEE, Paris, and the Ph.D. degree in engineering from the University of Erlangen-Nuremberg, Germany, in 1999. He is currently a Professor of Media Technology with the Technical University of Munich (TUM). His research interests include visual-haptic information processing and communication, telepresence and teleoperation, surface haptics, tactile Internet, and networked and interactive multimedia systems. From 1994 to 2000, he was a member of the Research Staff with the Image Communication Group, University of Erlangen-Nuremberg. From February 2000 to December 2001, he was a Post-Doctoral Fellow with the Information Systems Laboratory, Stanford University. In February 2002, he joined the Department of Electrical Engineering and Information Technology, TUM. He was elected as a Fellow of the IEEE in 2015 for his contributions to visual and haptic communications. He was a recipient of the 2011 Research Award of the Alcatel-Lucent Foundation. He serves as the Chair for the IEEE standardization activity P1918.1.1 a.k.a. Haptic Codecs for the Tactile Internet.