Affine Intra-prediction for Versatile Video Coding

Jayasingam Adhuran and Anil Fernando
University of Surrey
Guildford, UK

Gosala Kulupana and Saverio Blasi
BBC Research and Development
London, UK

***Abstract*—New algorithms are being investigated in the context of next generation video coding to achieve higher compression efficiency. In particular, intra block copy is well known to accurately predict screen content and artificially generated content where patterns and edges may repeat within the frame. On the other hand, such type of video content is often subject to geometrical transformations such as transitions, rotations, zooms etc, which may not be accurately captured by simply copying neighbouring pixels. A new intra-prediction scheme is presented in this paper whereby blocks of samples from already reconstructed areas are processed by means of an affine-type transformation. This allows more accurate prediction of blocks which improve compression efficiency in a variety of conditions. Experiments in the context of Versatile Video Coding standard show the proposed method can improve intra-coding compression efficiency by 2.01% BD-rates on average, and up to 4.81%, with negligible impact on the decoder complexity.**

***Index Terms*—VVC, intra-coding, affine, intra block copy**

I. Introduction

New video coding initiatives, including Versatile Video Coding (VVC) standard [1], are being developed with the goal of addressing the demand for more video content at better qualities and larger resolutions. More and more applications rely on efficient compression of not only natural video content, but also artificially generated content (referred to as screen content), including video games, screen sharing applications, video conferencing, as well as the inclusion of computer-generated overlays on natural video content.

To this end, VVC focuses on versatility, as it tries to address coding different types of content in different conditions, including lossy and lossless compression. This relies on the investigation of new, advanced tools specific to screen content applications, including the Palette mode [2], Block Differential PCM (BDPCM) [3] and Intra Block Copy (IBC) [4]. These tools, in combination with new versatile tools such as affine inter-prediction [5], multiple transforms [6] [7], secondary transforms [8] [9] or new quantization techniques [10], are significantly contributing to the development of next generation versatile video compression. For instance, the VVC Test Model (VTM) 7.0 [5] is capable of compressing screen content using 40.4% less bitrate than its predecessor HEVC test Model (HM) at the same objective video quality [11].

In order to further reduce the necessary bitrates to transmit screen content, new methods are still being investigated in the context of screen content coding category in VVC. Consequently, a novel tool for improving the efficiency of VVC screen content intra-prediction is proposed in this paper. Artificially generated content is often subject to geometrical transformations such as transitions, rotations, zooms etc, which may not be accurately predicted by conventional techniques. The proposed method applies affine transformations to reconstructed intra-prediction reference samples to address these cases. Two Control Point Vectors (CPVs) are used and transmitted in the bitstream to derive the transformation parameters. Specific techniques to limit the impact of signalling the required information are presented, as well as a novel control point estimation process. The proposed method is shown to consistently improve the compression efficiency of VVC, as shown in the rest of this paper.

Email: j.adhuran@surrey.ac.uk. This research was supported by CONTENT4ALL, a Horizon 2020 project funded by the European Commission (Grant Number 762021).

II. State of the art

Many tools and algorithms have been proposed in the context of improving video coding intra-prediction, both for natural content as well as for screen content coding. In this section, a small selection of tools proposed in the context of VVC standardization is presented. Many more references and papers are available in the literature.

In the context of coding natural content, Multiple Reference Line Prediction (MRLP) [12] is a new tool that allows intra-prediction to exploit reference pixels not directly adjacent to the current block. Intra Sub Partitions (ISP) [13] has also been proposed to improve intra-prediction. When using ISP, smaller partitions of samples in a block are independently predicted, transformed and reconstructed. Matrix-based Intra Prediction (MIP) [14] was proposed to perform intra-prediction using pre-trained, neural-network based matrices.

Additional tools were presented in the specific context of screen content coding. BDPCM [3] was proposed to exploit horizontal or vertical redundancies among quantized coefficients in the case of sharp edges typical of artificially generated content. Palette mode [2] was proposed to compress content by means of dictionary-based prediction methods.

IBC [4] was proposed as a tool to exploit spatial redundancies especially present in screen content. When using IBC, a motion estimation is performed within the already reconstructed area in the current frame. The corresponding block of reference samples is then used as prediction for the current block. The corresponding motion vector is then transmitted in the bitstream similar to inter-prediction. IBC is

particularly effective in areas with text, repeating geometrical patterns or periodic textures.

In this paper, an algorithm based on extending the functionalities of IBC is proposed. An affine model is used to predict rotations, zooms, or non-translational transformations that may produce better prediction blocks from the reference area in the current frame in case of screen-recorded sequences which often include transitions, zooms, etc. While similar methods have been proposed in the past [15], many new aspects are considered in this paper, including an efficient method to transmit CPVs based on Motion Vector Prediction (MVP) and Adaptive Motion Vector Resolution (AMVR), as well as a new CPV estimation algorithm specifically designed for IBC-coded affine-predicted blocks.

III. Proposed method

VVC already incorporates a simplified affine prediction model in inter predicted frames to exploit geometrical correlations between objects in the current block and reference frame [5]. In order to minimize the side information necessary to transmit the model parameters, VVC makes use of either a 6- or 4-parameters model. The 4-parameter model can represent transformations such as rotations or zooms, while the 6-parameter model can represent more complex transformations such as stretches, at the cost of more bits to transmit the additional parameters in the bitstream.

A. IBC coding with affine prediction

Due to its simplicity and smaller impact on bitrate, the 4-parameter model was used as basis for the method proposed in this paper. An affine transformation is typically represented by a zoom factor ρ, rotation angle θ, and horizontal and vertical displacements c and f, respectively. In video coding applications though, a CPV-based representation is preferred, as the motion vectors obtained can be easily fed to existing motion compensation schemes to produce predicted samples. Formally, the affine transformation between reference and current block is obtained as:

$$\begin{aligned} mv^h_{(x,y)} &= x^| - x = (\rho\cos\theta - 1)\cdot x + \rho\sin\theta\cdot y + c \\ mv^v_{(x,y)} &= y^| - y = -\rho\sin\theta\cdot x + (\rho\cos\theta - 1)\cdot y + f \end{aligned} \quad (1)$$

where $mv^h_{(x,y)}$ is the motion vector representing the displacement between the pixel in location x, y in the current block, and the corresponding reference pixel in the affine-transformed block. From the aforementioned equation, two CPVs $CPV0$ and $CPV1$ can be obtained by substituting $((x, y)$ with $(0, 0)$ and $(w, 0)$, respectively, where w is the current block width. Eq. (1) can then be formulated as a function of the CPVs:

$$\begin{aligned} mv^h_{(x,y)} = \;& CPV0^h + \frac{(CPV1^h - CPV0^h)}{w}\cdot x \\ & - \frac{(CPV1^v - CPV0^v)}{w}\cdot y \\ mv^v_{(x,y)} = \;& CPV0^v + \frac{(CPV1^v - CPV0^v)}{w}\cdot x \\ & + \frac{(CPV1^h - CPV0^h)}{w}\cdot y \end{aligned} \quad (2)$$

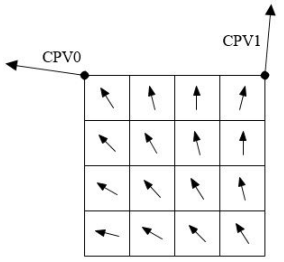

Fig. 1. CPV-based affine model representation

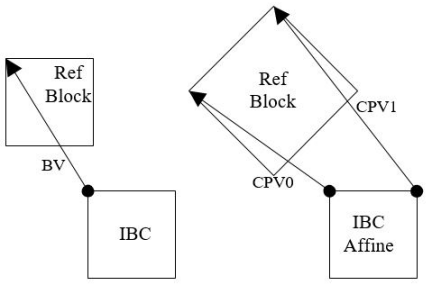

Fig. 2. Conventional IBC (left) and IBC with affine prediction (right)

where $CPV0^h$, $CPV0^v$, $CPV1^h$, $CPV1^v$ are the horizontal and vertical components of $CPV0$ and $CPV1$, respectively.

Eq. (2) can be used to determine different motion vectors to predict each sample. Unfortunately, in order to address limitations due to decoder complexity as well as buffering and throughput requirements, it is not practical to predict each pixel with a different motion vector. For that reason, the VVC affine model imposes splitting the current block into smaller sub-blocks of 4×4 samples, and predicting all samples in a sub-block with the same motion vector, derived from Eq. (2) at the centre of the sub-block, as illustrated in Fig. 1. Given these constraints, only CUs whose size is equal or larger than 16×16 samples can be affine predicted.

The 4-parameter affine model was integrated in the context of IBC intra-prediction. When using conventional IBC, a Block Vector (BV) is used to locate the top-left corner of the reference block to predict the current block, as illustrated in Fig. 2. This paper proposes to extend such mode to include affine transformations. Several aspects need to be taken into account to achieve this, including limiting the impact of signalling the affine parameters, as well as limiting the complexity of the method with efficient motion estimation.

Similar to IBC, the method needs to ensure that the reference samples pointed by the motion vectors resulting from the affine model refer to already reconstructed samples. This implies that only the top and left portion of the frame with respect to the current block can be searched. Moreover, differently than inter-predicted blocks, where sub-pixel motion estimation filters are available to produce fractional interpolated reference samples, such filters are not typically available in intra-coded blocks. Making these filters available in intra frames may be impractical, as that would drastically change decoder pipelines, as well as introduce additional decoder complexity to compute the interpolated samples. For that rea-

son, the proposed affine model restricts motion compensation to integer-valued motion vectors, meaning that each motion vector as output from Eq. (2) is rounded to the nearest integer-valued component. This requires a simple copy of reference samples for each sub-block. As an additional advantage, using only integer-valued motion vectors implies that no additional padding is necessary in the surrounding of the current block to ensure that the reference sample are already reconstructed, to account for the large interpolation filters necessary to produce fractional reference samples.

In order to be able to use the proposed method, several elements need to be available at the decoder side. More specifically:

- An IBC flag to indicate whether a block is IBC-coded.
- An affine flag, to determine whether an IBC-coded block makes use of conventional IBC or the proposed method.
- In case the block is affine-coded, two CPVs to determine the sub-block motion vectors.

Coding of the CPVs can be very bitrate-expensive. For that reason, the proposed method shares features of conventional inter-prediction coding to limit such signalling. More specifically, MVP is used to reduce the magnitude of the CPVs components. Such coding can then be performed using Golomb-style coding, whereby an element with a smaller magnitude produces a smaller number of bits for its binary representation. A specific MVP process was designed for IBC-coded affine-predicted blocks, inspired by the process used in inter blocks. In this regard, both IBC and IBC affine-coded neighbours are considered in a predefined order to generate MVPs, as shown in Fig. 3. A list of two MVP candidates is considered. When a candidate is available from the following process, it is inserted in the list, and the process is interrupted as soon as the list is full with two candidates.

In a first step, a number of neighbouring blocks are investigated to check whether they are affine-coded. Blocks A0 and A1 are investigated first, to extract maximum one MVP candidate; blocks B0, B1 and B2 are then checked to possibly extract another MVP candidate. In case one of these blocks is found to be affine-coded, its CPVs are considered. These are not directly inserted in the MPV list though. Conversely, the affine model in Eq. (2) is used to determine the motion vectors in the relative locations x and y at the top-left and top-right corners of the current block. Such new motion vectors are inserted in the list as potential MVPs for the current block.

In a second step, in case there is space available in the MVP list, a number of neighbouring blocks are investigated to check whether they are IBC coded. Given that IBC-coded blocks only have one BV available, this process is independently performed to obtain a potential MVP for CPV0 and CPV1, respectively. Specifically, blocks A, B and C as in Fig. 3 are visited in this order to possibly obtain MPV for CPV0. Blocks D and E are visited in this order to possibly obtain an MPV for CPV1. Finally, zero vectors are used to fill any space remaining in the MVP list.

Following this process, the encoding of CPV components proceeds as follows. First, a flag is signalled to indicate

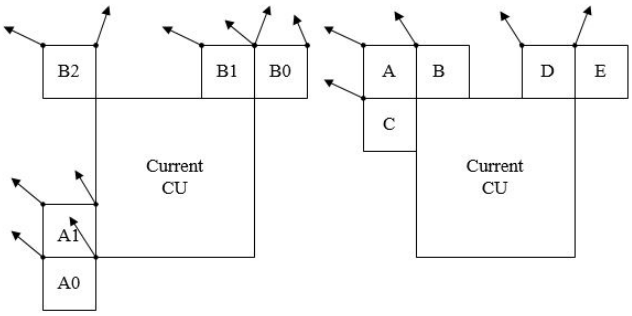

Fig. 3. MVP process in IBC affine-coded blocks. IBC affine-coded neighbours (left), and IBC coded neighbours (right) are considered.

which MVP is used between the two in the list. Then, CPV differences are computed:

$$\begin{aligned} CPVD0 &= CPV0 - MVP0 \\ CPVD1 &= CPV1 - MVP1 - CPVD0 \end{aligned} \tag{3}$$

where $MVP0$ and $MVP1$ are the MVPs for CPV0 and CPV1, respectively. The CPV differences are finally coded in the bitstream. A similar strategy to the usage of AMVR for coding of motion vector differences in inter blocks is used here. When using AMVR, a parameter is first signalled to determine the precision of motion vectors, namely full-pel, or 4-pel precision. The components decoded from the bitstream are interpreted in accordance to the signalled resolution, meaning that a component equal to 1 can correspond to either a displacement of a pixel, or four pixels, depending on the AMVR value. This can drastically reduce the number of bits needed to transmit large components. All four components from $CPVD0$ and $CPVD1$ are considered. In case they are all representable in 4-pel resolution, then the corresponding AMVR value is transmitted, and components are divided by four before encoding. The inverse process is performed at the decoder to reconstruct the CPVs.

B. CPV estimation

In parallel to the normative aspects of the method, an important aspect of the proposed approach relies on efficient derivation of the CPVs at the encoder side. Testing too many options is undesirable due to the increase in complexity. Moreover, in contrast to than conventional affine inter-prediction where large vectors may be beneficial, in the case of intra-predicted blocks prediction accuracy tends to decrease as the distance from the current block increases. This means that it is unlikely that large CPVs could produce good predictions.

For that reason, in this paper a method to efficiently reduce the number of tested CPV candidates is proposed, resulting in limited impact on the encoder complexity. A list of candidates is used where potential vectors are included following different processes. A restricted search range is allowed comprising a few neighbouring Coding Tree Units (CTUs) in addition to the already reconstructed portion of the current CTU as shown in Fig. 4. Any candidate that falls outside of the range is discarded and not tested.

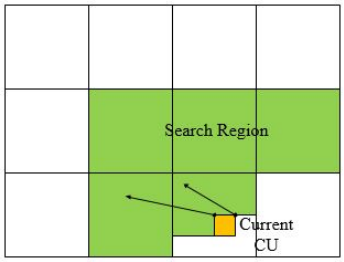

Fig. 4. Search region of the proposed CPV estimation process.

A cache is used to keep track of potential CPVs to test for a given block. The content of the cache is updated every time a block is tested. More specifically, the best N CPVs from a prediction distortion sense are inserted at the top of the cache. In case the cache is already full, vectors are pulled out starting from the oldest, ensuring the cache always contains most recently tested CPVs. Obviously, the larger the cache, the more candidates are tested, resulting in better performance but correspondingly higher encoding times. A cache of 64 vectors is considered in this paper.

In addition, neighbouring blocks are checked to see if they are affine-coded or IBC-coded. In case a neighbour block is IBC coded, the corresponding BVs are included in the candidate list to test as potential CPVs. In case a neighbouring block is affine-coded, then their control points are used to derive new potential candidates for the current block, taking into account the neighbouring block's width and height. The inherited CPVs from affine-coded neighbouring blocks are further processed to possibly result in larger CPV components, as shown in Fig. 5. The distant CUs pointed by the CPVs are checked. In case one of these is affine-coded, then the corresponding CPVs of such CUs are also considered. These are then processed so that new candidate CPVs are obtained, which point to the same areas as the CPVs of the distant CU. Assume that a neighbouring CU to the current CU is affine-coded with CPVs $CPV_{neigh}0$ and $CPV_{neigh}1$, and further assume that the distant CU pointed by $CPV_{neigh}0$ is also affine-coded. Denote as $CPV_{dist}0$ and $CPV_{dist}1$ its corresponding CPVs. Denote also as W the vector along the top edge of the current block, and as W_{dist} the vector along the top edge of the distant CU. New CPVs $CPV0$ and $CPV1$ are finally tested, obtained as:

$$\begin{aligned} CPV0 &= CPV_{neigh}0 + CPV_{dist}0 \\ CPV1 &= W + CPV_{neigh}0 - W_{dist} + CPV_{dist}1 \end{aligned} \tag{4}$$

Anytime a given pair of CPVs is tested, an additional process is applied to possibly refine the vectors based on the gradient of the reference samples, similarly to the affine motion estimation used in VVC [16]. The idea is to try to find the four optimal affine parameters that transform the current reference block so that the obtained prediction distortion is minimised. In order to do so, the directionality of the block is estimated by determining the edges within the block by means of applying a Sobel filter to the reference samples. Then, the reference samples are compensated by the estimated gradient. The parameters that minimise the distortion between compensated reference samples and original samples are considered as possible candidates.

Formally, for a given block of samples $b_{(x,y)}$ and a given pair of CPVs, the reference samples $ref_{(x,y)}$ pointed by the derived motion vectors are considered. The corresponding gradient matrix H is derived as:

$$H_{(x,y)} = \left(\frac{\partial ref_{x,y}}{\partial x}, \frac{\partial ref_{x,y}}{\partial y} \right) (x + mv^h_{(x,y)}, y + mv^v_{(x,y)}) \tag{5}$$

Denoting as $A = \{c, \rho\cos\theta, f, \rho\sin\theta\}$ the vector of affine parameters, the compensated predicted samples can then be defined as:

$$p_{(x,y)} = ref_{(x,y)} + H_{(x,y)} \cdot A \tag{6}$$

Finally, the four affine parameters that minimise the distortion between original samples $b_{(x,y)}$ and compensated predicted samples $p_{(x,y)}$ are considered. From those parameters, a new pair of CPVs is computed using Eq. 1 and added to the list of candidate CPVs.

The aforementioned CPV estimation processes ensure that a reduced set of potentially valid candidates is tested for a given block, while limiting the number of operations performed for a block. To further reduce the impact of the search on complexity, each candidate is tested only from prediction distortion perspective, using the Sum of Absolute Differences (SAD). Only the optimal pair of CPVs for a given block is fully tested in a Rate-Distortion sense. In this case, the corresponding predicted block is used to compute residual samples, that are then transformed quantized, inverse quantized and inverse transformed, to produce a reconstructed block. The reconstruction error is then considered together with the corresponding bitrate to encode the block parameters, to determine whether the block should be IBC affine-coded.

IV. Results

In order to validate its performance, the proposed method was implemented in the context of VTM 7.0 and tested under the All-Intra (AI) configuration under Common Testing Conditions (CTC) for screen content coding as in JVET [17]. Three classes of sequences, referred to as class Text and Graphic with Motion (TGM) comprising 4 sequences at 1920×1080 resolution, class F and class Mixed comprising 4 sequences at various resolutions (as in [17]) are considered in the CTCs. Coding performance was measured using Bjontegaard BD-rates [18] as well as encoding and decoding time increase, in

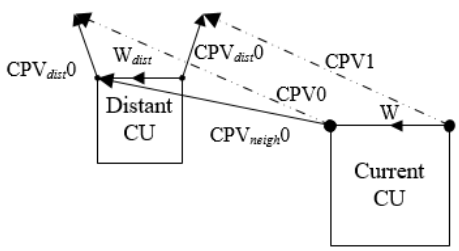

Fig. 5. Inheriting CPVs from distant CUs by vector concatenation.

TABLE I
CODING PERFORMANCE, IN PERCENTAGE (%).

Sequence	BD-rate	Enc. Time	Dec. Time
FlyingGraphics	-2.62	168	94
Desktop	-4.81	193	96
Console	-4.00	202	99
ChineseEditing	-1.03	159	102
Avg. Class TGM	**-3.11**	**180**	**98**
BasketballDrillText	-0.02	132	92
ArenaOfValor	-0.03	136	101
SlideEditing	-1.12	170	93
SlideShow	-0.43	166	93
Avg. Class F	**-0.40**	**151**	**95**
Basketball_Screen	-2.24	147	105
MissionControlClip2	-1.94	134	94
MissionControlClip3	-3.86	147	96
Avg. Mixed	**-2.68**	**143**	**98**
Avg. Overall	**-2.01**	**159**	**97**

TABLE II
DISTRIBUTION OF MODE SELECTION IN PERCENTAGE (%).

Sequence	Intra	IBC	Proposed IBC affine
FlyingGraphics	58.96	15.91	25.13
Desktop	61.23	14.03	24.74
Console	55.69	25.59	18.72
ChineseEditing	80.48	6.35	13.17
BasketballDrillText	98.32	1.54	0.13
ArenaOfValor	99.25	0.51	0.24
SlideEditing	85.74	5.34	8.92
SlideShow	98.24	0.92	0.85
Overall	**74.79**	**11.09**	**14.12**

percentage, with respect to an unmodified VTM 7.0 anchor. Full sequences were used for class TGM and F, whereas a limited number of 20 frames was used for class Mixed to reduce testing times.

The coding performance of the proposed IBC coding with affine prediction is illustrated in Table I. It is evident that the proposed method outperforms the anchor for all sequences producing, consistent gains of on average 2.01%, with a limited impact on encoding complexity (on average 159%) and no impact on the decoder complexity. Most significant gains are achieved in mixed material (class TGM and Mixed), where up to 4.81% gains are obtained (for the *Desktop* sequence). On the other hand, the method does not bring benefits in 3D-graphics content such as *Arena of Valor*, or predominantly natural content such as *BasketBallDrillText*. This is due to the method tackling geometrical transformations that are mostly found in artificially-generated content, as opposed to natural or 3D reconstructed material. On the other hand, thanks to the nested signalling of the technique, that depends on the usage of IBC, no negative impact on performance is obtained even when the method is not useful.

Furthermore, Table II displays the distribution of usage of conventional intra-prediction, conventional IBC, and proposed method, on blocks where the method is allowed. The approach is used on average more than conventional IBC, showing that the benefits of using affine transformations overcome the additional signalling costs. Usage of the method is substantial in class TGM sequences, with up to 25% of blocks using the approach in some cases.

V. CONCLUSION

A new tool for improving efficiency of screen content intra coding for next generation video coding was presented in this paper. Affine 4-parameter transformations are used in intra frames to account for geometrical transitions such as rotations, zooms and translations. Efficient CPV coding and estimation processes are illustrated to ensure the method is efficient with limited impact on the encoder complexity. The experiments show significant benefits in compression efficiency with an average bitrate reduction by 2.01%. More advanced affine models can be investigated to further improve the method.

REFERENCES

[1] B. Bross, J. Chen, and S. Liu, "Versatile video coding (VVC) draft 6," *Document JVET-O2001*, Gothenburg, Sweden, July 2019.

[2] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging hevc screen content coding extension," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 50–62, Jan 2016.

[3] M. Abdoli, F. Henry, P. Philippe, and G. Clare, "AHG11: Block DPCM for screen content coding," *Document JVET-L0078*, Macao, October 2018.

[4] X. Xu, S. Liu, T. Chuang, Y. Huang, S. Lei, K. Rapaka, C. Pang, V. Seregin, Y. Wang, and M. Karczewicz, "Intra block copy in hevc screen content coding extensions," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 4, pp. 409–419, Dec 2016.

[5] S. Kim J. Chen, Y. Ye, "Algorithm description for Versatile Video Coding and test model 3 (VTM)," *Document JVET-J1002*, Macao, October 2018.

[6] Zhao X., Chen J., Karczewicz M., Zhang L., Li X., and Chien W., "Enhanced multiple transform for video coding," in *Data Compression Conference (DCC), 2016*, 2016, pp. 73–82.

[7] J. Lainema, "Shape adaptive transform selection," *Document JVET-M0303*, Marrakech, Morocco, Jauary 2019.

[8] M. Koo, M. Salehifar, J. Lim, and S. Kim, "Reduced secondary transform," *Document JVET-N0193*, Geneva, Switzerland, March 2019.

[9] X. Zhao, J. Chen, A. Said, V. Seregin, H. Egilmez, and M. Karczewicz, "Nsst: Non-separable secondary transforms for next generation video coding," in *2016 Picture Coding Symposium (PCS)*. IEEE, 2016, pp. 1–5.

[10] H. Schwarz, N. Tung., D. Marpe, and T. Wiegand, "Transform coefficient coding and dependent quantization," *Document JVET-K0071*, Ljubljana, Slovenia, July 2018.

[11] F. Bossen, X. Li, and K. Sühring, "Test model software development," *Document JVET-Q0003*, Brussels, Belgium, January 2020.

[12] Y. Chang, H. Jhu, H. Jiang, L. Zhao, X. Zhao, X. Li, S. Liu, B. Bross, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand, "Multiple reference line coding for most probable modes in intra prediction," in *2019 Data Compression Conference (DCC)*, March 2019, pp. 559–559.

[13] S. De-Luxán-Hernández, V. George, J. Ma, N. Tung, H. Schwarz, D. Marpe, and T. Wiegand, "Intra sub-partitions coding mode," *Document JCTVC-M0102*, Marrakech, Morocco, January 2019.

[14] M. Schäfer, B. Stallenberger, J. Pfaff, P. Helle, H. Schwarz, D. Marpe, and T. Wiegand, "An affine-linear intra prediction with complexity constraints," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 1089–1093.

[15] J. Cao, Z.-R. L, J. Wang, F. Liang, Y.-F. Yu, and Y. Liu, "An intra-affine mode for screen content coding," *Document JVET-O0682*, Gothenburg, Sweden, July 2019.

[16] K. Zhang, Y. Chen, L. Zhang, W. Chien, and Marta M. Karczewicz, "An improved framework of affine motion compensation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1456–1469, 2018.

[17] J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations," *Document JVET-J1010*, Ljubljana, Slovenia, July 2018.

[18] G. Bjontegaard, "Calculation of average PSNR differences between rd-curves," *VCEG-M33*, 2001.