Improving Coding Efficiency of Massive Parallel Intra Prediction Using Alternative References

Iago Storch, *Member, IEEE*, Nuno Roma, *Senior Member, IEEE*, Daniel Palomino, *Senior Member, IEEE*, and Sergio Bampi, *Senior Member, IEEE*

***Abstract*— Exploring massive parallelism is a common strategy to mitigate the processing time of modern video encoding standards. Nonetheless, data dependencies challenge parallelism exploitation, especially during intra prediction, where the reconstructed adjacent blocks are used as references. Some works use the original frame samples as references to decouple adjacent blocks and allow parallelism. Still, the original samples are static and cannot model the nuances of different bitrates. In this context, this work seeks to improve the coding efficiency of parallel intra prediction implementations by using alternative reference samples based on low-pass filters that better represent the nuances of different bitrates for any partitioning structure. Variations in multiple aspects of the filters are considered, such as their dimension and also the precision and distribution of their coefficients. Experimental evaluations assessed the similarity of such alternative samples when compared to the regular ones, in addition to their impacts on coding efficiency and the processing overhead required to obtain such samples. The results from such experiments demonstrate that the alternative references improve coding efficiency when compared to the original samples, especially at lower bitrates. Furthermore, the additional filtering stage poses negligible timing overhead in most computing systems.**

***Index Terms*— Massive parallelism, intra prediction, block-level parallelism, video coding, low-pass filters.**

I. Introduction

VIDEO coding applications are known for being computationally burdensome. Newer codecs usually introduce more encoding tools to improve coding efficiency, which pushes the complexity issue even further [1]. Many works have been proposed to mitigate these difficulties, usually by simplifying or designing dedicated hardware architectures for some encoding stages [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. Nonetheless, even by skipping some encoding stages, the speedup provided by solutions based on sequential computing is severely limited. With the widespread availability of heterogeneous computing devices composed of multiple processing units, exploring parallelism is a prime requirement to achieve substantial encoding acceleration.

This scenario has already been explored in modern video coding standards that support encoding tools specifically designed to explore parallel processing in multicore CPUs: tiles, slices and wavefront parallel processing (WPP) [12], [13], [14].

Although these tools are easy to employ, they only explore the availability of multiple processing cores in CPUs. Many modern computing systems are heterogeneous [15], where CPUs work alongside GPUs to collaboratively complete a task. Hence, extracting the most performance out of modern heterogeneous computing systems requires distributing the workload among processing devices with different characteristics.

A large share of the workload from modern encoding standards comes from testing multiple partitioning structures and prediction modes for the same frame region seeking the best coding efficiency [16]. The simultaneous processing of multiple blocks poses significant acceleration potential. Nonetheless, video coding algorithms explore temporal and spatial redundancies in the textures to improve coding efficiency, leading to data dependencies that create complex challenges in conceiving parallel solutions. This data dependency is critical during the intra prediction, since the prediction of one block depends on the reconstructed pixels of adjacent blocks.

Two strategies were proposed to deal with this issue. The first one is based on rearranging the encoding order of blocks [17] to allow launching the encoding of a block as soon as its data dependencies are satisfied, using a non-raster encoding order. However, this solution is unfeasible for modern encoding standards due to the high flexibility supported by their partitioning structures. The second strategy is to simulate the reconstructed samples of adjacent blocks [18], [19], [20], and then conceive parallel solutions based on GPUs to accelerate the encoding. Under this strategy, the most common approach is considering the original frame samples as references during intra prediction. However, they do not represent the reconstructed samples properly, especially at low bitrates. Therefore, it is important to investigate different strategies to improve the coding efficiency of massive parallel intra prediction solutions.

Received 27 November 2024; revised 7 March 2025 and 2 April 2025; accepted 16 April 2025. Date of publication 6 May 2025; date of current version 27 October 2025. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) under Grant 001, in part by FAPERGS, in part by CNPq, and in part by Fundação para a Ciência e a Tecnologia (FCT) under Project UIDB/50021/2020. The Article Processing Charge (APC) for the publication of this research was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES (ROR identifier: 00x0ma614). For the purpose of open access, the authors have assigned a Creative Commons CC BY license to any accepted version of the article. This article was recommended by Associate Editor W. Liu. *(Corresponding author: Iago Storch.)*

Iago Storch and Daniel Palomino are with the Video Technology Research Group (ViTech), Postgraduate Program in Computing (PPGC), Federal University of Pelotas (UFPel), Pelotas 96010-610, Brazil (e-mail: icstorch@inf.ufpel.edu.br; dpalomino@inf.ufpel.edu.br).

Nuno Roma is with INESC-ID and the Instituto Superior Técnico, Universidade de Lisboa, 1649-004 Lisbon, Portugal (e-mail: nuno.roma@inesc-id.pt).

Sergio Bampi is with PPGC, Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre 91501-970, Brazil (e-mail: bampi@inf.ufrgs.br).

Digital Object Identifier 10.1109/TCSI.2025.3564455

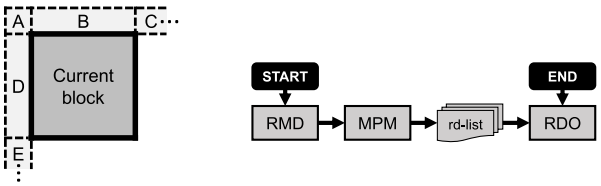
This work tackles the second strategy from a circuits and systems perspective. Instead of simply using the original frame samples as references, we propose introducing pre-processing low-pass filters designed to generate alternative samples that better represent the nuances of encodings at different bitrates. These filters are applied to the original samples to degrade high-frequency details and approximate the quantization effect.

Furthermore, the current environment of computing systems is becoming more complex. The popularization of internet of things devices and easily accessible cloud platforms boost the development of high-performance computing systems, whereas the increasing usage of edge computing requires highly efficient embedded devices. Simultaneously, computing systems are becoming increasingly heterogeneous, where CPUs and GPUs work alongside to solve problems. To account for such a heterogeneity, the proposed solution is implemented into multiple computing systems, varying from high-end CPUs and GPUs to embedded devices, to assess its feasibility according to multiple deployment scenarios.

Experimental results show that the proposed alternative references obtain substantial coding efficiency gains when compared to the popular strategy of using the original frame samples as references to expose parallelism. While using the original frame samples as references leads to an average coding efficiency of 0.870% BD-BR, the proposed alternatives reach coding efficiency results as good as 0.552% BD-BR. These gains are possible because the parameters of low-pass filters can be tailored to approximate the texture degradation at different bitrates. Besides, the timing results demonstrate that the additional filtering operation provides negligible overhead.

It should be noted that this paper is an extension of previous work [21]. The original paper presented a simple correlation and coding efficiency evaluation of 9 empirical low-pass filters. Compared to [21], the proposed paper introduces 30 novel filters not yet considered in the literature to produce better alternative references. It also expands the discussion regarding correlation and coding efficiency and provides a study on the computing overhead of the proposed method. **The novel contributions of this paper are summarized as follows:**

- A deeper discussion on the available methods for exposing block parallelism during intra prediction along with their limitations (Section III);
- 30 novel low-pass filters not yet considered to generate alternative reference samples for intra prediction. The filters are systematically defined into three families according to the following parameters: distribution (Gaussian or pseudo-Gaussian), coefficient representation precision (integer or floating-point), and implementation complexity (separable or not) (Section IV);
- A thorough evaluation of the correlation of each alternative sample with the regular references (Section VI);
- A detailed assessment of the coding efficiency results obtained by different families of filters, and between filters of the same family (Section VII);
- An evaluation of the processing time required by the additional filtering stage when executed in GPUs and multicore CPUs, along with their overhead when compared to modern software encoders (Section VIII).



(a) Reference sample positions (A-E). (b) Intra mode decision scheme used in the HM and VTM encoders.

Fig. 1. Intra reference samples and mode decision.

II. Intra-Frame Prediction Overview

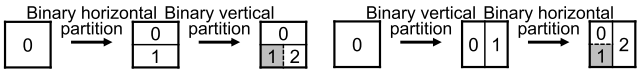
The intra-frame prediction (or intra prediction, for short) is designed to explore the spatial redundancies within a video frame representing the current block based on the samples of adjacent blocks that were previously encoded. Each encoding standard supports a different set of intra prediction concepts, like filtering the samples of adjacent blocks, creating a flat surface based on the samples of adjacent blocks, copying adjacent samples directly into the current block, and deriving the predicted samples after data-driven methods [12], [14].

Although highly specialized intra prediction concepts may represent the current block based on blocks from arbitrary positions of the frame [22], most intra prediction concepts explore the samples of directly adjacent blocks. These adjacent blocks are usually in one of the five regions labeled from **A** to **E** in Fig. 1a. Each intra mode explores only a subset of these neighbors at a time. Also, some of these neighbors may be unavailable depending on the current block partitioning.

Testing all prediction possibilities to decide the best mode for each block poses a substantial workload. Many encoders circumvent this issue by breaking the mode decision into multiple stages. The reference encoders for the HEVC [23] and VVC [24] standards, for instance, break the mode decision into three stages named *Rough Mode Decision* (RMD), *Most Probable Modes* (MPM), and *Rate-Distortion Optimization* (RDO). An overview of this scheme is depicted in Fig. 1b.

The RMD stage is responsible for a preliminary evaluation of most encoding modes. This evaluation ignores the impacts of transform and quantization, and the bitrate is estimated in a simplified context. The **N** modes that present the best performance are added to a rd-list containing the modes that are likely to be selected and should be further evaluated. The value of **N** depends on the implementation. Then, the MPM stage checks the prediction modes selected in adjacent blocks and derives a set of highly probable modes, adding them to the rd-list as well. Finally, the modes in the rd-list are evaluated by the RDO stage. The latter thoroughly evaluates the modes, going through the transform and quantization stages, and demands considerably more processing time. The mode with the best performance in RDO is selected for the current block.

Even though this strategy reduces the processing time, the overall process still poses a substantial workload. Exploring parallel processing is a promising strategy to deal with this large workload. One possibility consists of testing multiple prediction modes simultaneously for each block, such as [25]. This strategy is easily implemented due to the lack of data dependencies between prediction modes; nonetheless,



(a) Top and top-right regions of block 1 are already encoded.

(b) Only the top region of block 1 is already encoded.

Fig. 2. Example of a case where the available references for the same block (shaded in gray) differ based on the partition structure. Numbers represent the encoding order.

the speedup potential is limited since most of the encoding complexity comes from testing several partitioning possibilities [16]. Under these circumstances, encoding multiple blocks in parallel promises great acceleration potential. However, the spatial data dependencies between blocks (especially those for intra prediction discussed in Fig. 1a) create challenges that must be addressed to explore block-level parallelism without imposing substantial harm to coding efficiency.

Therefore, this work is concerned with breaking these data dependencies to allow a coding-efficient exploration of block-level parallelism during intra prediction. More precisely, a set of alternative reference samples is proposed to satisfy the data dependencies before encoding the adjacent blocks.

III. Related Works

One of the main challenges in designing efficient parallel algorithms for intra prediction lies in breaking the data dependencies between adjacent blocks to allow them to be processed concurrently. There are two major ways to deal with the block-level data dependencies of intra prediction: changing the encoding order of blocks and changing the reference samples.

A. Enabling Parallelism Using Alternative Encoding Order

Alternative encoding order consists of encoding a block as soon as its data dependencies are satisfied, allowing multiple blocks to be processed simultaneously. This technique is used by [17] to accelerate the intra prediction on the H.264 [26] and AVS [27] standards using GPUs. It resembles the WPP tool [28] for parallel processing. However, WPP works by breaking the entropy contexts among blocks and performing the complete encoding for each block in parallel.

Although it allows an efficient parallel intra prediction, this method was proposed for older standards that supported only a few block sizes in highly regular positions. The flexible partitioning structures of modern encoding standards allow the neighborhood of a block to be composed of multiple block combinations with distinct encoding orders. An example of this situation on the VVC standard is depicted in Fig. 2, where the numbers represent the encoding order of blocks and the shaded region depicts the block currently being encoded.

Note that the shaded block is in the same position in Fig. 2a and 2b. Yet, due to the different set of partitions employed in each case, a different set of references is available in each case: whereas the top and top-right references are available in Fig. 2a, only the top references are available in Fig. 2b.

Furthermore, this example does not consider the ternary partitions or the left neighborhood (regions A, D, and E), which would introduce more dependency possibilities. Clearly, the flexible partitioning provided by modern video coding standards creates complex data dependencies that cannot be modeled as in [17], rendering this strategy unfeasible.[1]

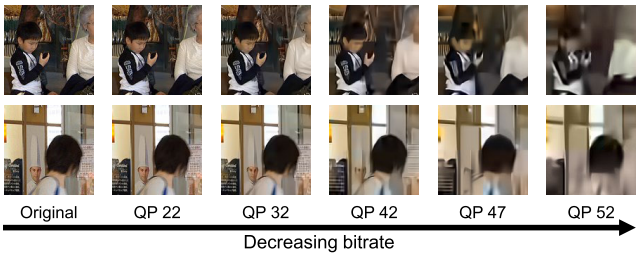


Fig. 3. Original and reconstructed blocks of *BQMall* sequence encoded at multiple bitrates in VTM encoder [24].

B. Enabling Parallelism Using Alternative Reference Samples

The strategy of alternative reference samples consists of performing the intra mode decision using reference samples that are available before the adjacent blocks are encoded. The prediction residual is signaled in the bitstream considering the regular references once they are available to avoid introducing drift. Most works dealing with block-level parallelization during intra prediction employ this strategy [18], [19], [20].

The authors of [20] designed a scheme to perform the intra prediction with all angular modes on the GPU and a GPU-based method to accelerate the block partitioning. The GPU accelerates only the RMD's mode decision while the CPU performs RDO. Combined, these schemes accelerate the overall encoding in 2 times. The solution from [18] explores CPU and GPU parallelism simultaneously to tackle the whole RMD and RDO stages of intra mode decision. Depending on the operation point, this strategy accelerates the complete encoder in as much as 19.32 times. The proposal from [19] accelerates the prediction with MIP modes during the RMD stage, where all blocks and prediction modes are evaluated concurrently using GPUs. Here, the evaluation of MIP modes during RMD stage is accelerated in about 70.7 times.

Although the parallelization details vary from one work to another [18], [19], [20], they all use the original frame samples as references to break the data dependencies between adjacent blocks and allow their parallel processing. This strategy has a key effect during the Rate-Distortion Optimization: intra prediction modes yield different results than those obtained using the reconstructed samples. As a result, the optimization process selects suboptimal solutions, leading to a degradation in coding efficiency, as demonstrated in [18], [19], and [20]. Higher quantization levels degrade the quality the most, increasing the disparity between the original and reconstructed samples. Hence, low-bitrate encodings are affected the most by the strategy of using the original samples as references.

An example of such impact is depicted in Fig. 3, which represents two distinct 128 × 128 blocks from the *BQMall* video sequence [30] encoded at different bitrates with the VVC Reference Encoder (VTM) [24]. The leftmost column represents the original blocks, whereas the columns to the right represent blocks encoded with decreasing bitrates. The

[1]The CTUs still present a regular structure that is explored by WPP [29]. The challenges refer to the recursive partitioning of CUs.

bitrate is controlled by a quantization parameter (QP), where the larger the QP, the lower the bitrate.

When comparing the original blocks with those encoded at multiple QPs, it is visible that the encoding performed at QP 22 causes almost imperceptible degradation. At QP 32, a clear degradation is already visible in the block of the upper row, especially in the face and sweater. Then, starting at QP 42, the fine details are lost and blocking artifacts are already visible. In summary, due to the large disparity between reconstructed and original samples at large QPs (low bitrates), using the latter as references during intra mode decision misleads the encoder decisions and produces considerable coding efficiency losses.

C. Final Remarks

Designing an alternative processing order presents challenges due to the flexible partitioning mechanisms supported by modern video encoding standards. Conversely, using the original frame samples as references presents limitations due to the disparity between original and reconstructed samples.

Most of the distortion in video coding pipelines comes from quantization, where the high-frequency details of the image are attenuated. This observation raises the possibility of modeling the reconstructed samples with a smoothed version of the original ones, with the smoothing intensity controlled by the target bitrate (i.e., the quantization level). One way to achieve this is the spatial low-pass filter [31] that allows scalable smoothing by setting its coefficients. Under this premise, this work explores the usage of low-pass filters to approximate the reconstructed samples and decouple adjacent blocks, allowing for a massively parallel and efficient encoding.

IV. Proposed Alternative References for Intra Prediction

This work proposes to improve massive parallel intra prediction solutions through a set of alternative reference samples that break the data dependencies between adjacent blocks, allowing the prediction of all blocks to be performed concurrently with improved coding efficiency. It builds on top of our previous work [21] and proposes a set of low-pass filters that, when applied to the original frame samples, approximates the degradation of the encoding process observed in the reconstructed samples. This filtering operation aims to produce alternative reference samples that resemble the regular references more closely than simply using the original frame samples as widely used in literature [18], [19], [20].

Fig. 4 presents an overview of how our contribution relates to the general video coding framework and related works. The color scheme is detailed in the legend at the bottom of Fig. 4. A regular sequential video coding framework is depicted in Fig. 4a. Since the intra prediction of one block depends on the reconstructed samples of adjacent blocks, only one block is processed at a time, and its samples are used as references for the following adjacent blocks. Note that the reconstructed samples of $block_i$ are used as references to predict $block_{i+1}$. Fig. 4b represents the encoding scenario of related works that use the original frame samples as references to decouple adjacent blocks during intra mode decision [18], [19], [20]. Since the original frame samples are always available, the intra prediction mode of all blocks is decided concurrently at the beginning. From then on, the blocks are reconstructed one at a time using the regular references. Finally, the proposed method is presented in Fig. 4c. Here, the proposed filtering stage is executed at the beginning and filters the complete frame concurrently. The filter output (smoothed original samples) is fed into the parallel mode decision module, which decides the intra prediction mode for all blocks concurrently with improved coding efficiency. Then, the reconstruction of blocks is performed in a serial fashion as in related works. It should be noted that the proposed solution is not a parallelization strategy for the intra prediction itself. Instead, it is a strategy to break the data dependencies and improve the coding efficiency of existing parallel intra solutions based on alternative samples.

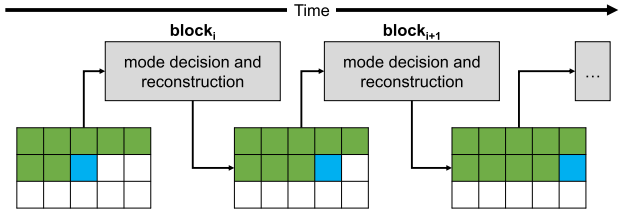
Although most of the experimental validation (Sections V, VI, VII, and VIII) is performed with the VVC standard, the proposed solution is compatible with any block-based video coding standard since the impacts of quantization on the reconstructed samples are similar irrespective of the standard. It can be coupled with various intra prediction concepts such as angular, DC, planar, and MIP — the latter a simplification of deep learning methods [32]. In summary, the proposed solution is a standalone module that can be plugged into an encoder following any video coding standard (including future ones) that explores block-level parallelism and performs intra prediction based on reconstructed adjacent samples. Low-pass filters with multiple properties are considered to account for the nuances of different bitrates and hardware platforms.

The newly proposed filters are specifically tailored according to two main properties: representation (integer or floating point) and separability (separable or not). Filters with floating point representation may be able to represent the nuances of various bitrates more properly than integer coefficients, whereas integer coefficients are more hardware-friendly for custom implementations. When it comes to separability, separable filters require fewer multiplication-accumulation operations, but they introduce an additional data dependency between applying the filter in one and another direction. The tradeoff between these properties may vary depending on the computing system in which they are applied. The following sections delve further into the details of these low-pass filters.

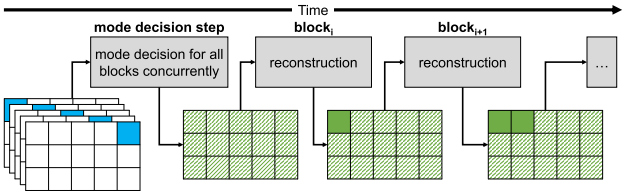
A. Empirical Low-Pass Filters

Our previous work [21] showed the potential of using alternative references produced by 3×3 and 5×5 low-pass filters. These families of filters are referred to as **Emp_3 × 3** and **Emp_5 × 5**, respectively. Furthermore, the i_{th} variation of these empirical 3×3 and 5×5 filters are labeled as **Emp_3 × 3_vi** and **Emp_5 × 5_vi**.

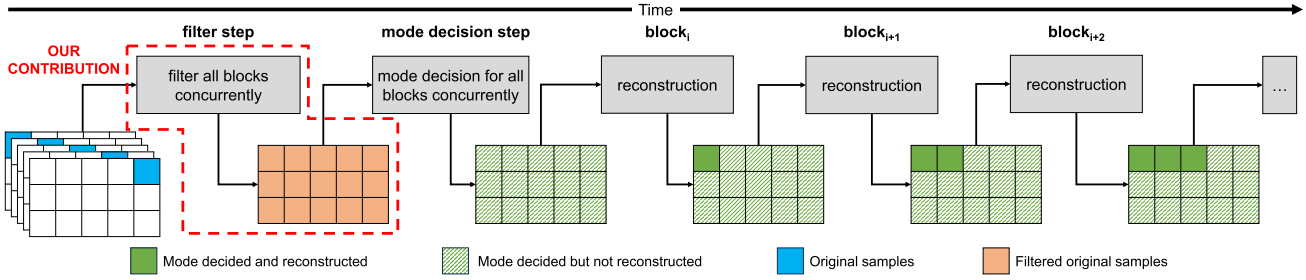
Although these filters demonstrated the coding efficiency potential of smoothing the original frame samples before using them as references, they present some limitations. First, the coefficients' distributions are designed empirically and may not reproduce the quantization noise properly. Second, using integer coefficients limits the filter's strength flexibility. Third, except for the filters 3×3_v0 and 5×5_v0, the remaining filters are not separable, which prevents the usage of some



(a) Conventional intraframe prediction performs the mode decision and reconstruction for one block at a time.



(b) Related works [18]–[20] use the original frame samples as references to perform mode decision for all blocks concurrently. Blocks are reconstructed one at a time.



(c) Proposed method applies a filtering operation to the original samples before using them as references during parallel mode decision. Then, blocks are reconstructed one at a time.

Fig. 4. Overview of the proposed solution (c) when compared to the regular encoding (a) and related methods (b).

well-known implementation optimizations discussed in the next sections. This work expands the potential of alternative references by proposing two novel sets of low-pass filters represented with floating point and integer precision.

B. Floating Point Precision Low-Pass Filters

This work proposes a set of low-pass filters based on the Gaussian distribution[2] to approximate the reconstructed samples out of the original ones. Using such filtered samples as references allows for massive parallelism exploration while providing a better coding efficiency than the method of using the original samples. The 2D Gaussian distribution is defined by (1), where x and y represent the displacement with respect to the center of the distribution and σ is the standard deviation.

$$G(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2} \tag{1}$$

In addition to providing a more appropriate low-pass filter, the 2D Gaussian function is separable into the product of two 1D functions [31]. This property allows optimized hardware implementations with reduced arithmetic operations.

Since this work deals with filters in the discrete space (i.e., samples in a picture), we subsample the Gaussian distribution at integer positions. The normalization is based on the sum of the coefficients instead of $2\pi\sigma^2$, hence, the proposed filters follow the distribution from (2) and (3). r represents the radius of the matrix: 1 for 3×3 matrices and 2 for 5×5 matrices.

$$G(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{S}, (x, y) \in [-r, +r] \tag{2}$$

$$S = \sum_{i=-r}^{r} \sum_{j=-r}^{r} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{3}$$

[2]The frequency response of an ideal low-pass filter is the rectangular function, which translates into a sinc function in the spatial domain. The main lobe of a sinc function is very similar to a Gaussian distribution.

0.04	0.12	0.04
0.12	0.38	0.12
0.04	0.12	0.04

(a) Float_3×3_065.

1	3	1
3	11	3
1	3	1

(b) Int_3×3_065.

1	3	1
3	9	3
1	3	1

(c) Pseudo_3×3_3.

Fig. 5. Examples of proposed alternative reference samples.

This work considers 3×3 Gaussian filters with standard deviations between 0.30 and 0.65, in steps of 0.05. Fig. 5a depicts the 3×3 filter generated with $\sigma = 0.65$. Due to the separable property of the Gaussian function, these filters are separable. The remaining of this paper uses **Float_3 × 3_0XX** when referring to a filter that follows a Gaussian distribution with a standard deviation of 0.XX and represented in floating point precision. Similarly, **Float_3 × 3** refers to the family of filters discussed in this section.

C. Integer Precision Low-Pass Filters

The floating-point precision provides a faithful representation of the Gaussian distribution, but it may pose a substantial overhead depending on the computing device. Two sets of filters based on integer coefficients are proposed to mitigate this concern: one is based on **rounding the original filters**, while the other is based on **approximating the Gaussian distribution while maintaining its separability**.

Rounding the original filters consists of scaling all coefficients linearly such that the smallest value equals one, and then rounding the remaining coefficients. This process is applied to all filters from Section IV-B. Fig. 5b demonstrates the integer 3×3 filter with $\sigma = 0.65$. On the one hand, it solves the issue of dealing with floating-point representation. On the other hand, the separability property is lost.[3] A scaled filter

[3]This is not the case for all rounded distributions, but most of them.

with a standard deviation equal to 0.XX is referred to as **Int_3 × 3_0XX**, while its family is labeled as **Int_3 × 3**.

Approximating the Gaussian distribution while maintaining its separability is based on creating the complete filter out of its separable components. Due to the separability property, any integer 3×3 Gaussian filter can be represented by a 1D array as $[1, N, 1]$, where $N \geq 1$. Similarly, 5×5 Gaussian filters can be represented as $[1, M, N, M, 1]$, where $N \geq M \geq 1$. Therefore, we propose a set of filters that approximate a Gaussian distribution by defining N and M explicitly. The proposed 3×3 filters employ $N \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14\}$. Conversely, 5×5 filters have the following parameters: $(M, N) \in \{(2, 3), (2, 4), (3, 8)\}$. We refer to these filters created from their separable components as pseudo-Gaussian. Fig. 5c presents one of such 3×3 pseudo-Gaussian filters with $N = 3$. These 3×3 and 5×5 filters are labeled as **Pseudo_3 × 3_N** and **Pseudo_5 × 5_N_M**, referring to the N and M values previously discussed. Similarly, **Pseudo_3 × 3** and **Pseudo_5 × 5** represents the general families.

D. Summary of Proposed Alternative Samples

- **Emp_3 × 3_vi and Emp_5 × 5_vi** correspond to filtering the original frame samples using the kernels proposed in [21], which were created empirically. The **9 possibilities** detailed in Section IV-A compose these families;
- **Float_3 × 3_0XX** corresponds to the 3×3 low-pass filters that follow a Gaussian distribution, with standard deviation equal to 0.XX, and represented with floating-point precision. This family is composed of **8 separable filters** discussed in Section IV-B;
- **Int_3 × 3_0XX** corresponds to the Gaussian 3×3 low-pass filters with standard deviation 0.XX that are linearly scaled and represented with integer precision. Due to rounding operations, these filters are not separable. This family contains **8 filters** discussed in Section IV-C;
- **Pseudo_3 × 3_N and Pseudo_5 × 5_N_M** are low-pass filters created by defining their 1D separable components explicitly. These filters are separable by definition. **Pseudo_3 × 3** contains **11 possibilities**, while **Pseudo_5×5** contains **3 possibilities**; details are available in Section IV-C.

V. Evaluation Setup

This work is concerned with the suitability of the alternative reference samples in substituting the regular ones and the timing overhead of the additional filtering stage. The suitability of the alternative references is assessed in terms of correlation with the regular references and coding efficiency obtained by using such alternative samples during intra prediction.

The correlation and coding efficiency evaluations are performed according to the VVC standard and using the VVC Reference Encoder (VTM) [24]. Furthermore, the encoder uses the *All Intra* configuration, in which only the intra prediction tools are enabled. The experiments regarding processing overhead use the VTM [24], x265 [33], and vvenc [34] encoders.

To evaluate the correlation results, a set of videos is encoded with the baseline VTM encoder at different bitrates and the reconstructed frames are traced. These represent the best encoding decisions according to the VTM encoder. Then, the original frame samples are filtered according to the proposed filters. By computing the correlation between the reconstructed frames (regular references) and the frames produced by each low-pass filter variation it is possible to estimate the accuracy of such filters in approximating the quantization degradation.

TABLE I
Hardware Used for Experimental Evaluation

System	Device (RAM)	Boost freq.	Cores
Xeon	2× CPU Intel Xeon Gold 6430 (128 GB)	2.1 GHz	2× 32
Cortex A15	ARM Cortex A15 (2 GB)	2.0 GHz	4
GTX 1080	NVIDIA GTX 1080 (8 GB)	1.733 GHz	2560
RTX 3060	NVIDIA RTX 3060 (10 GB)	1.710 GHz	8704
Mali-T628	Mali-T628 (2 GB)	600 MHz	96

Computing the coding efficiency variation requires modifying the VTM encoder to use alternative samples as references during intra prediction. Then, the same set of videos is encoded multiple times, using the alternative references produced by a different filter in each encoding. Finally, the rate-distortion performance obtained by each alternative is compared to the baseline encoder to measure their impacts on coding efficiency.

The timing overhead is assessed by comparing the processing time of the filtering operation to that of multiple software encoders: VTM [24], x265 [33], and vvenc [34]. Conversely, custom implementations of the proposed low-pass filters are conceived targeting CPU and GPU computing devices.

The VVC standard counts with Common Test Conditions (CTCS) [30], which determines the guidelines to be followed to guarantee that the results obtained by different authors are comparable. The experimental evaluation of this work considers the complete dataset recommended by the CTCs, which includes 26 videos of resolutions from 240p up to 4k. This dataset includes natural content and also synthetic content with sharp edges and fast transitions, which combined, encompass a variety of textures and motion. Furthermore, the CTCs require using QPs equal to 22, 27, 32, and 37. Nonetheless, additional encoding scenarios using QPs 42, 47, 52, and 57 are also considered to assess the suitability of the proposed alternative references in a wider range of bitrates.

The experiments are carried out on the computing systems listed in Table I, where "System" labels each computing system and "Device (RAM)" depicts the computing device along with the available RAM (or VRAM). "Cores" depict the number of processing cores in each device. All filtering experiments are replicated in all devices. Due to the large number of alternative references being evaluated, the experiments consider the first 32 frames of each video sequence.

VI. Correlation Between Regular and Alternative References

The original frame samples are smoothed using the low-pass filters detailed in Section IV. Then, the Pearson correlation coefficient is computed between the regular references and the samples in each alternative. The correlation is computed for each QP individually. The closer to 1 is the correlation of one alternative, the more similar it is to the regular references.

An overview of the range of correlation results is presented in Fig. 6. Here, the correlation obtained when applying the

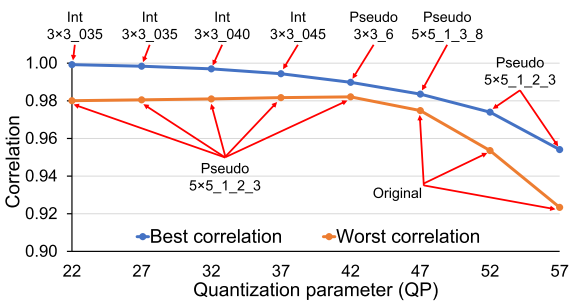


Fig. 6. Best and worst correlation result for each QP, along with the respective alternative samples.

same filter to all videos is averaged in each QP. The horizontal axis depicts the QP values from 22 to 57, whereas the vertical axis represents the average correlation in such a QP. For clarity, the blue line represents only the best correlation while the orange line represents the worst correlation. The filters leading to the best and worst results in each QP are labeled.

When the worst correlation results from Fig. 6 are inspected, it is visible that a low-pass filter based on 5×5 coefficients provides the worst correlation at QPs between 22 and 42. This happens because, in general, larger filters provide greater smoothing intensity while videos encoded at small QPs are less susceptible to degradation. When QPs between 47 and 57 are considered, however, the worst correlation is obtained when the original frame samples are used as references. This demonstrates that at larger QPs, any low-pass filter considered in this work outperforms the widely used strategy of using the original frame samples as alternatives to the regular ones.

When the best cases are inspected, different filters provide the best correlation results depending on the QP. For QPs between 22 and 42 the best alternative is based on 3×3 filters, whereas for QPs between 47 and 57, the best alternatives employ 5×5 filters. Furthermore, for filters with the same dimension, the relative weight of the central coefficient decreases with larger QPs, hence, filters with stronger smoothing intensity are more suitable in scenarios of more quantization.

Although Fig. 6 shows that distinct alternative references result in the best correlation depending on the QP, it does not provide insights into the performance of each variation individually. This comparison among alternatives is provided in Fig. 7. The horizontal axis represents the quantization parameters, whereas the vertical axis represents the considered alternative references. Furthermore, each square in the heatmap represents the average results (among videos) obtained by one alternative in one QP. Each value in Fig. 7 represents the *correlation difference with respect to the best alternative in the respective QP*. Naturally, the best alternative for each QP has a value of zero. These cases are highlighted with a black outline. The remaining alternatives on the same QP have a negative value, representing their correlation difference with respect to the correlation obtained by the best alternative in the same QP.

The results from Fig. 7 show that the novel set of alternative reference samples proposed in this work provide a better correlation with the regular references when compared to those proposed in [21] (**Emp_3 × 3** and **Emp_5 × 5**). This is

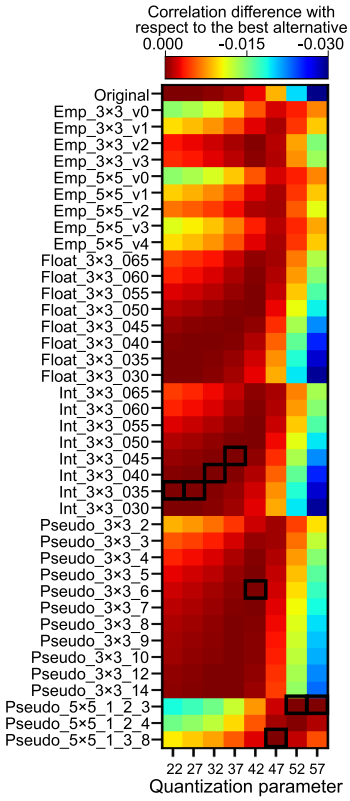


Fig. 7. Variation in the correlation of every alternative with respect to the best one in each QP. The best correlation for each QP is highlighted with a black outline. Remaining alternatives are represented as a difference in relation to the best case.

confirmed by having the best alternative within the set of novel alternatives and their darker red tone. Finally, note that the proposed **Pseudo_5 × 5** filters achieve the best correlation for QPs between 47 and 57, even though only three alternatives are considered. Further investigation of coefficient distributions following a similar strategy could provide better results.

VII. Coding Efficiency of Alternative References

The coding efficiency is obtained by replacing the regular references with alternative ones during the intra prediction. More precisely, the RMD stage of the intra mode decision from VTM (see Fig. 1b) is modified to consider the alternative reference samples. In this case, the alternative references are used to apply the angular, DC, Planar, and MIP modes. The regular references are used by RDO to generate the correct residual information for the best mode on the bitstream. Hence, the proposed method interferes by forwarding a different set of modes from the RMD into the RDO stage, but the filtered frame does not interfere with the final prediction residual. The coding efficiency of a video coding solution is characterized by its compression and visual quality. This information is assessed using rate-distortion plots and Bjøntegaard delta-bitrate (BD-BR) [35], which represents the bitrate increase required by a target encoder to provide the same objective quality as a reference encoding. Larger BD-BR values represent a worse coding efficiency. In this work, the BD-BR is computed considering the coding efficiency of the encoder with alternative references and the baseline VTM encoder.

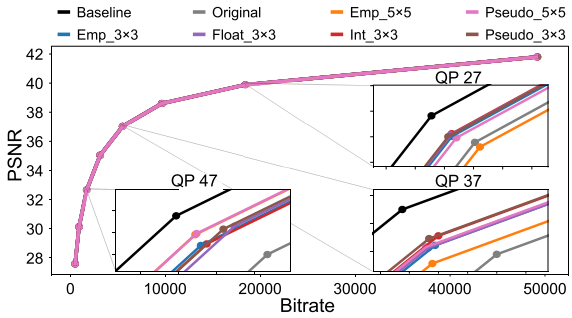


Fig. 8. Rate-distortion performance of the best alternative references within each family for the *BasketballDrive* sequence.

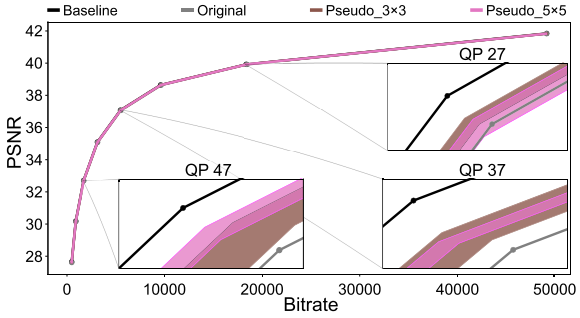


Fig. 9. Rate-distortion performance range of **Pseudo_3×3** and **Pseudo_5×5** families for the *BasketballDrive* video sequence.

A. Best Alternative References for Each Filter Family

Different alternative references can provide different coding efficiency results for each bitrate (QP). This work identifies the best combination of alternative references for each video and QP through exhaustive evaluation. More precisely, all permutations of alternative references for a specific filter family are enumerated considering the eight QPs from Section V. A family with N alternatives has N^8 combinations. Each combination represents an encoding scenario where a different reference is used for each QP. The BD-BR for each combination is computed, and the one with the best BD-BR is deemed the best for their respective QP. This process is repeated for each family to determine the combination of alternatives inside each family that leads to the best coding efficiency.

B. Coding Efficiency Evaluation Using Rate-Distortion Plots

Fig. 8 presents the rate-distortion lines for all low-pass filter families considered in this work for the sequence *BasketballDrive*. Each color corresponds to a family of alternative references, and the lines represent the rate-distortion performance obtained by the best combination of alternative samples in such a family. The black and gray lines represent the performance of the baseline encoder and the encoder using original frame samples as references. Three inserts at QPs 27, 37, and 47 provide interesting insights into their behavior.

The main conclusion from Fig. 8 is that, depending on the QP, the best coding efficiency is obtained by an alternative reference from a different family. At QP 27, for instance, the best results are obtained by alternative references from the **Int_3 × 3**, **Float_3 × 3**, **Pseudo_3 × 3**, and **Emp_3 × 3** families, whereas using the alternatives from the **Emp_5 × 5** family leads to the worst coding efficiency results. Notably, the minor degradation caused by such a small QP is efficiently modeled by the weaker low-pass filters based on 3×3 kernels. Similarly, at QP 37, the best coding efficiency results are obtained using alternative references from the **Int_3 × 3** and **Pseudo_3×3** family, but the coding efficiency of filters based on 5×5 kernels improve significantly when compared to QP 27. Furthermore, using all families of proposed alternative references outperforms using the original frame samples. Finally, at QP 47, the distortion caused by quantization is substantially larger than in the previous cases. In this situation, the best coding efficiency results are obtained using 5×5 low-pass filters. Clearly, larger filters allow stronger smoothing and reproduce the distortion at larger quantization levels more efficiently.

These results show that although the original frame samples are good candidates at larger bitrates, there are multiple alternative reference samples that provide better coding efficiency results at lower bitrates. Similar conclusions can be drawn for other video sequences. The main difference is that the best family of alternative references for each QP differs slightly.

Even though the previous analysis demonstrated the potential of alternative reference samples over the original frame samples, it considered the best-case scenario for each family of alternatives. Clearly, different variations within the same family provide distinct coding efficiency results. This phenomenon is further assessed through Fig. 9, which presents the coding efficiency variability within two representative families when encoding the *BasketballDrive* video sequence: **Pseudo_3 × 3** in brown and **Pseudo_5 × 5** in magenta. Each shaded area in Fig. 9 represents the area between the lower and upper coding efficiency bounds of each family. If plotted, the rate-distortion curve from all variations of a family would lie within the shaded area. The black and gray lines represent the coding efficiency of the baseline encoder and the encoder using the original frame samples as references, respectively. Inserts are presented at QPs 27, 37, and 47 to provide detailed insights.

The first conclusion from Fig. 9 is that the variability within the **Pseudo_5 × 5** family is smaller than that of the **Pseudo_3 × 3** family. This observation is likely due to the **Pseudo_5 × 5** family having fewer variations than **Pseudo_3 × 3** family (3 variations against 11) and not a direct consequence of their suitability. Irrespective of that, the coding efficiency range of these two families overlaps at least partially for the whole range of bitrates. Consequently, there is no obvious choice of filter family and the actual distribution of coefficients is relevant. Yet, it is visible that most of the low-pass filter possibilities (even in their worst case) provide coding efficiency results better than those of using the original frame samples. Even if there is no way of assessing the video texture to define the best low-pass filter, using any of the proposed filter possibilities is likely more efficient than using the well-known method of using the original frame samples.

C. Coding Efficiency Evaluation Using BD-BR

To shed some light on the overall results of using alternative references during intra prediction, another evaluation is conducted in which the coding efficiency is measured in terms

TABLE II
CODING EFFICIENCY RESULTS IN TERMS OF BD-BR [%], CONSIDERING THE EXTENDED QP RANGE (22-57)

Video	Orig	Static alternative irrespective of QP: Int 3×3 μ=0.60	Pseudo 3×3 N=6	Pseudo 3×3 N=9	Pseudo 3×3 N=14	Pseudo 5×5 M,N=2,3	Best alternative for each video-QP (single family): Emp 3×3	Emp 5×5	Float 3×3	Int 3×3	Pseudo 3×3	Pseudo 5×5	All Best
Campfire	0.807	0.782	0.764	0.758	0.725	1.103	0.696	1.051	0.669	0.660	0.629	0.822	0.577
CatRobot	1.250	1.016	1.008	1.123	1.142	0.989	0.931	0.931	0.983	0.974	0.928	0.895	0.788
DaylightRoad	1.436	0.992	1.052	1.151	1.217	0.958	0.835	0.898	0.891	0.905	0.895	0.844	0.705
FoodMarket	0.589	0.351	0.505	0.498	0.470	0.324	0.294	0.219	0.364	0.306	0.298	0.277	0.204
ParkRunning	0.505	0.449	0.469	0.472	0.475	0.485	0.427	0.462	0.424	0.419	0.419	0.438	0.362
Tango	1.253	0.979	1.041	1.100	1.102	0.807	0.828	0.758	0.907	0.914	0.884	0.772	0.714
Avg Class A	**0.973**	**0.762**	**0.807**	**0.850**	**0.855**	**0.778**	**0.669**	**0.720**	**0.706**	**0.696**	**0.675**	**0.675**	**0.558**
BasketballDrive	1.206	0.743	0.828	0.901	0.960	0.885	0.665	0.741	0.676	0.706	0.608	0.601	0.520
BQTerrace	1.046	0.969	0.949	0.948	0.945	1.656	0.915	1.670	0.864	0.867	0.859	1.140	0.780
Cactus	1.002	0.841	0.841	0.838	0.897	1.120	0.763	1.128	0.795	0.778	0.740	0.821	0.642
MarketPlace	0.485	0.403	0.429	0.467	0.452	0.514	0.390	0.441	0.379	0.377	0.361	0.413	0.333
RitualDance	0.787	0.744	0.747	0.764	0.725	0.907	0.637	0.870	0.658	0.631	0.633	0.760	0.564
Avg Class B	**0.905**	**0.740**	**0.759**	**0.784**	**0.796**	**1.017**	**0.674**	**0.970**	**0.674**	**0.672**	**0.640**	**0.747**	**0.568**
BQMall	0.709	0.654	0.620	0.622	0.639	1.363	0.613	1.447	0.510	0.539	0.517	0.841	0.487
BasketballDrill	1.336	1.230	1.177	1.167	1.205	2.267	1.115	2.075	0.996	1.004	0.957	1.470	0.886
PartyScene	0.714	0.768	0.689	0.669	0.637	1.797	0.712	1.810	0.556	0.565	0.569	1.081	0.543
RaceHorses	0.815	0.793	0.760	0.747	0.719	1.215	0.690	1.158	0.655	0.651	0.649	0.880	0.574
Avg Class C	**0.893**	**0.861**	**0.811**	**0.801**	**0.800**	**1.661**	**0.783**	**1.622**	**0.679**	**0.690**	**0.673**	**1.068**	**0.623**
BQSquare	0.767	1.020	0.910	0.816	0.791	2.749	0.943	2.693	0.689	0.677	0.708	1.556	0.648
BasketballPass	0.761	0.850	0.724	0.659	0.640	1.572	0.648	1.667	0.509	0.546	0.558	0.937	0.452
BlowingBubbles	0.683	0.702	0.638	0.664	0.686	1.420	0.598	1.318	0.500	0.531	0.520	0.865	0.447
RaceHorses	0.594	0.640	0.580	0.563	0.640	1.287	0.601	1.198	0.452	0.466	0.440	0.878	0.398
Avg Class D	**0.701**	**0.803**	**0.713**	**0.676**	**0.689**	**1.757**	**0.697**	**1.719**	**0.538**	**0.555**	**0.557**	**1.059**	**0.486**
FourPeople	0.865	0.739	0.800	0.772	0.819	0.980	0.691	1.029	0.700	0.691	0.666	0.757	0.589
Johnny	1.001	0.880	0.896	0.937	0.928	1.192	0.788	1.210	0.820	0.808	0.771	0.821	0.598
KristenAndSara	0.959	0.929	0.881	0.873	0.894	1.295	0.865	1.252	0.792	0.818	0.786	0.995	0.714
Avg Class E	**0.942**	**0.849**	**0.859**	**0.861**	**0.880**	**1.156**	**0.760**	**1.302**	**0.712**	**0.718**	**0.695**	**0.908**	**0.597**
ArenaOfValor	0.687	0.696	0.659	0.668	0.669	1.242	0.666	1.228	0.589	0.601	0.590	0.907	0.561
BasketballDrillText	1.173	1.140	1.037	1.095	1.135	2.114	0.990	2.061	0.912	0.927	0.871	1.322	0.776
SlideEditing	0.592	0.967	1.012	0.821	0.822	1.967	0.936	1.943	0.447	0.457	0.533	1.063	0.235
SlideShow	0.592	1.286	1.039	1.178	1.067	2.217	0.905	2.285	0.436	0.398	0.693	1.344	0.257
Avg Class F	**0.761**	**1.022**	**0.937**	**0.940**	**0.923**	**1.885**	**0.874**	**1.880**	**0.596**	**0.596**	**0.672**	**1.159**	**0.457**
Average All	**0.870**	**0.829**	**0.810**	**0.818**	**0.823**	**1.324**	**0.736**	**1.290**	**0.661**	**0.662**	**0.657**	**0.904**	**0.552**

of the BD-BR metric. This evaluation considers two encoding scenarios. In the first scenario, the same alternative reference is used for all QPs. This represents a static encoding system that applies identical filtering irrespective of QP and video. In the second scenario, the employed alternative reference can change from one QP to another and also between videos, selecting the best alternative available in each family. This represents an adaptive encoding system capable of interpreting the video content and applying distinct filtering depending on the context. We compute the coding efficiency results considering the extended QP range (22-57) for both cases.

Table II shows the coding efficiency results for all evaluated video sequences considering the static alternative references (between columns **Orig.** and **Pseudo_5 × 5 M,N=2,3**) and the best alternative references (between **Emp_3 × 3** and **Pseudo_5 × 5**). Finally, **All Best** represents the case in which each video and QP are encoded with the best available alternative, without constraining the selection to a single family.

It is possible to observe that several of the proposed alternative references outperform the conventional strategy of using the original frame samples. The left half of Table II shows average coding efficiency improvements even if the same low-pass filter is used for all videos and QPs. In general, the variations based on 3×3 filters provided coding efficiency gains for most of the videos and resolutions. The **Int_3×3_060** and **Pseudo_3 × 3_6** variations obtained the best results on average, as they reduced the average coding efficiency penalty from 0.870% BD-BR down to 0.810% and 0.818% BD-BR, respectively, when compared to using the original frame samples as references. However, the **Pseudo_5 × 5_2,3** variation was only able to improve the average coding efficiency of 4k videos. In general, more aggressive filters are more suitable for larger resolutions, whereas less aggressive ones are efficient for smaller resolutions. Finally, these results show that the coding efficiency of parallel intra prediction systems can be improved by simply adding a static filtering pre-processing stage without any further assessment of the video content.

The right half of Table II provides interesting insights on the coding efficiency results in case the best alternative reference for each video and QP could be predicted. It is visible that, for most families of filters, there is a combination of alternatives that provides substantial coding efficiency gains when compared to using the original samples as references. Remarkable results are obtained by the **Pseudo_3×3** family, which reduces the average coding efficiency penalty from 0.870% BD-BR down to 0.657% BD-BR, on average. Similarly to the previous evaluation, even though the **Pseudo_5 × 5** family counts with only three filter alternatives, its performance is similar to that of the best alternative in the 4k resolution. Nonetheless, it is not efficient in replacing the original samples when smaller resolutions are considered. Finally, when the best-case

scenario is considered (**All Best**), the average coding efficiency is improved from 0.870% BD-BR to 0.552% BD-BR.

These results demonstrate the coding efficiency potential of using the proposed alternative reference samples to enable massive parallelism exploration during intra prediction. Coding efficiency gains are observed even in a naive case where the encoder uses the same filter variation irrespective of the video content or QP. Furthermore, if the video content could be pre-processed to identify the best filtering possibility, the coding efficiency loss related to exploring block-level parallelism could almost be cut in half. Finally, it is visible that selecting the proper alternative references provides coding efficiency gain for all classes of videos, ranging from 4k (Class A) down to 240p (Class C) resolution, including natural and synthetic (Class F) content. Hence, the proposed method adapts well to a variety of video content and encoding scenarios.

VIII. Overhead of Using Alternative References

A. CPU Implementation

For CPU devices, two parallel implementations using OpenMP [36] are designed. One uses integer precision, while the other uses floating-point precision. These implementations are based on convolving the 3×3 or 5×5 kernels with the original frame samples to obtain the filtered results. The parallelization is performed row-wise, in which the sample rows are equally divided among the available threads. If N threads are employed, then thread T_i is responsible for filtering the samples in rows R_i, R_{N+i}, R_{2*N+i}, and so on.

B. GPU Implementation

Since GPU devices require the memory hierarchy to be explicitly managed to provide the best throughput, four distinct implementations are conceived using OpenCL [37], namely: Int 2D, Int 1D, Float 2D, and Float 1D. **Int** and **Float** refer to the coefficient precision, whereas **1D** and **2D** correspond to separable and not separable implementations, respectively.

The GPU implementations break the video frame into multiple patches composed of 128×32 or 128×16 samples, depending on the device, and each work-group (a set of work-items that represent threads in the OpenCL API) is responsible for filtering the samples in a distinct patch. The patch size is defined based on the local memory limitations (48 KiB for NVIDIA and 32KiB for ARM) and the requirement of storing data with floating-point precision. Furthermore, filtering samples at the patch's edges requires accessing samples outside the patch. To account for that, in addition to the samples of the patch itself, a halo containing these additional samples is also cached in local memory. The number of such halo samples depends on the radius (r) of the filter: 3×3 filters have $r = 1$, whereas 5×5 filters have $r = 2$. Conversely, d stands for the filter diameter: $d = 3$ for 3×3 filters, and $d = 5$ for 5×5 filters. Overviews of how each patch is processed in 2D and 1D implementations in NVIDIA devices are presented in Algorithms 1 and 2, respectively. These implementations employ 256 work-items in each work-group where successive work-items process successive samples in the patch. The ARM implementation is similar, but each work-group counts with 128 work-items and each patch comprehends 128×16 samples.

Algorithm 1 Procedure for 2D Filtering in GPU (NVIDIA)

```
    Input: globalOrig Output: globalFiltered
      // (128 + 2 * r) * (32 + 2 * r) global reads
1:  localOrig ← fetchSamples(globalOrig)
2:  Synchronize
      // 128 * 32 * d^2 multiplications
3:  localFiltered ← filter2d(localOrig)
4:  Synchronize
      // 128 * 32 global writes
5:  globalFiltered ← localFiltered
```

Algorithm 2 Procedure for 1D Filtering in GPU (NVIDIA)

```
    Input: globalOrig Output: globalFiltered
      // (128 + 2 * r) * (32 + 2 * r) global reads
1:  localOrig ← fetchSamples(globalOrig)
2:  Synchronize
      // 128 * (32 + 2 * r) * d multiplications
3:  localTemp ← filterHorizontal(localOrig)
4:  Synchronize
      // 128 * 32 * d multiplications. localOrig buffer is reused
5:  localOrig ← filterVertical(localTemp)
6:  Synchronize
      // 128 * 32 global writes
7:  globalFiltered ← localOrig
```

In the case of **2D filtering** (Algorithm 1), the procedure starts by fetching the original frame samples into the local memory (**line 1**). Here, the original samples of the patch, plus r additional rows/columns of samples around the patch, are obtained. Sequential workitems fetch sequential samples to coalesce memory accesses. Then, the threads synchronize to guarantee that all samples are cached before proceeding (**line 2**). In sequence, the filtering is performed considering the horizontal and vertical directions simultaneously (**line 3**). To avoid global memory accesses, the original samples and the filtering output are cached in local memory. This operation requires 3×3 or 5×5 multiplications to filter each sample in the patch. The threads synchronize again (**line 4**) and the result is offloaded into the global memory (**line 5**) with coalesced memory accesses. With the 2D filtering procedure, each 128×32 patch requires 36864 and 102400 multiplications when 3×3 and 5×5 filters are considered, respectively.

When **1D filtering** is employed (Algorithm 2), the same set of samples comprehending the patch and the additional halo is first fetched into local memory (**line 1**), and the threads synchronize as well (**line 2**). Then, the filtering is applied only in the horizontal direction (**line 3**), and the result is stored in a temporary buffer in local memory. Filtering in a single direction requires only 3 or 5 multiplications per sample. Nonetheless, the samples in the top and bottom halos must be filtered horizontally since they will be used as references during vertical filtering. In summary, $128 \times (32+2 \times r)$ samples are filtered, each one requiring d multiplications. The threads synchronize (**line 4**), and then a similar operation is performed for the vertical filtering (**line 5**), where only the samples within the patch are processed with d multiplications. The cached version of the original samples is not required anymore, so this memory region is reused to cache the final results. Finally, the

TABLE III
PROCESSING TIME PER FRAME OF EACH IMPLEMENTATION [ms]

Device	Precision / # threads	3x3 filters					5x5 filters				
		240p	480p	720p	1080p	4K	240p	480p	720p	1080p	4K
CPU 2x Xeon Gold 6430	Int / 1	3.4	13.2	30.2	69.2	279.0	10	39.8	90.0	174.6	698.6
	Float / 1	7.0	20.6	62.6	97.0	347.6	16.2	67.6	84.0	787.6	3150.4
	Int / 4	2.2	4.8	8.0	17.0	67.6	6.4	10.4	21.0	44.6	177.0
	Float / 4	3.2	8.8	11.8	25.4	85.0	6.6	11.6	23.8	198.8	795.4
	Int / 8	1.0	3.2	4.6	8.8	34.6	2.8	5.4	12.2	23.2	95.4
	Float / 8	1.6	3.6	6.0	13.6	44.2	3.0	6.6	13.0	101.2	403.2
	Int / 64	0.2	0.6	0.8	1.0	5.0	0.4	1.4	2.8	4.2	22.2
	Float / 64	0.6	1.0	1.4	2.6	9.6	0.4	1.6	2.0	16.8	66.6
	Int / 128	0.2	0.6	0.4	1.0	4.2	0.8	0.8	2.8	3.2	18.6
	Float / 128	0.2	0.6	0.8	1.4	4.6	0.2	0.8	1.8	8.8	33.4
	Int @ 30fps	1 thr. (294)	1 thr. (75)	1 thr. (33)	2 thr. (30)	9 thr. (33)	1 thr. (100)	3 thr. (53)	3 thr. (35)	4 thr. (39)	25 thr. (30)
	Float @ 30fps	1 thr. (142)	1 thr. (48)	3 thr. (52)	4 thr. (39)	11 thr. (30)	1 thr. (70)	3 thr. (52)	3 thr. (33)	4 thr. (39)	N.A.
CPU Odroid Cortex A15	Int / 1	17.2	69	175.2	400.2	1644.6	77.4	112.6	175.2	400.2	1644.6
	Float / 1	47.6	112.6	254.8	619.2	3080.6	106	229.2	254.8	619.2	3080.6
	Int / 2	8.4	34.4	85.4	195.8	798.8	101.8	151.2	85.4	195.8	798.8
	Float / 2	55.4	151.2	182.6	400.6	2231.8	120.8	222.2	182.6	400.6	2231.8
	Int / 4	4.2	27.8	43	100.4	439.6	27.8	54	57.2	100.4	439.6
	Float / 4	12.4	54	63.6	196	755.8	32.8	57.4	83.8	196	755.8
	Int @ 30fps	1 thr. (58)	3 thr. (43)	N.A.	N.A.	N.A.	4 (35)	N.A.	N.A.	N.A.	N.A.
	Float @ 30fps	3 thr. (46)	N.A.	N.A.	N.A.	N.A.	4 (30)	N.A.	N.A.	N.A.	N.A.
GPU GTX 1080	Int 2D	0.07	0.21	0.54	1.70	8.92	0.07	0.22	0.58	1.79	9.32
	Int 1D	0.07	0.21	0.55	1.73	9.04	0.07	0.22	0.57	1.74	9.11
	Float 2D	0.06	0.20	0.54	1.72	8.87	0.07	0.21	0.57	1.73	9.01
	Float 1D	0.06	0.20	0.55	1.70	8.91	0.07	0.22	0.57	1.72	9.03
GPU RTX 3080	Int 2D	0.03	0.08	0.26	1.09	5.94	0.04	0.09	0.27	1.11	6.04
	Int 1D	0.03	0.08	0.26	1.09	5.92	0.03	0.08	0.27	1.10	6.01
	Float 2D	0.03	0.08	0.26	1.06	5.92	0.03	0.09	0.26	1.09	6.03
	Float 1D	0.03	0.08	0.26	1.07	5.93	0.03	0.08	0.27	1.09	5.99
GPU Odroid Mali-T628	Int 2D	4.27	18.58	41.65	87.46	322.29	10.1	41.09	87.22	185.76	722.58
	Int 1D	2.81	13.06	27.84	61.15	235.47	5.51	23.68	50.13	108.74	415.89
	Float 2D	4.28	18.18	38.07	86.56	321.0	10.26	39.70	86.36	188.98	726.25
	Float 1D	2.79	12.66	29.46	65.74	228.15	5.56	25.25	52.50	111.85	411.19

threads synchronize (**line 6**), and the final result is offloaded into global memory (**line 7**). With the 1D filtering procedure, each 128×32 patch requires 25344 and 43520 multiplications when 3×3 and 5×5 filters are considered, respectively.

C. Processing Time Overhead

The CPU execution overhead considers only the running time of the filtering operation. For the GPU execution, the processing time is composed of the kernel execution time and the time to move the original and filtered samples between the main memory and GPU memory – i.e., the communication. Finally, the executions are repeated five times and averaged.

Table III presents the processing times for both implementations where the running time is averaged for all videos of the same resolution and presented in milliseconds. The CPU implementation based on OpenMP is evaluated on Xeon and Cortex A15 systems. Xeon counts with 128 threads using hyper-threading technology, whereas Cortex A15 counts with four threads. In both cases, the experiments used between 1 and the number of threads available in the devices. In the case of GPU implementations, since the complete frame is filtered in parallel, the number of threads increases with the resolution. 1080p videos are processed with 138,240 OpenCL work-items (i.e., GPU threads), whereas 4k videos are processed by 522,240 OpenCL work-items. The GPU processing times in Table III include the kernel execution and communication between CPU and GPU before/after the kernel execution to exchange the original/filtered samples.

From Table III, it is visible that when CPU devices are considered, filtering video frames with floating-paint precision takes more computing time than integer precision, whereas 5×5 kernels demand more processing time than their 3×3 counterparts. This relationship holds for both Xeon and ARM devices. Furthermore, when single-threaded implementations are considered, only videos of small resolution can be filtered in less than 33ms to provide real-time processing. The Xeon processor achieves real-time up to 720p resolution with 3×3 filters, whereas ARM achieves real-time only to 240p resolution with 3×3 filters. As the number of threads increases to 128, the processing time per frame of the Xeon processor for 1080p videos is reduced to the range between 1.0 and 8.8 ms, whereas for 4k frames, the processing time lies in the range between 4.2 and 33.4 ms. In general, except for the case of filtering 4k frames with 5×5 kernels, real-time processing is easily achieved with 128 threads. Finally, the lines "Int @ 30fps" and "Float @ 30fps" represent the minimum number of CPU threads required to provide a throughput of at least 30 frames per second (FPS) in each resolution and kernel size. Numbers between parenthesis represent the actual FPS. **N.A.** denotes the combinations of CPU device, resolution and kernel dimension that do not achieve real-time processing.

When the GPU implementations are considered, the GTX 1080 and RTX 3060 devices can process a 4k frame in at most 9.322 and 6.040 ms, respectively, which is enough to provide a throughput of over 100 FPS. For these devices, the filter dimensions, separability, and coefficient representation pose a small impact as the communication between GPU and host dominates the total GPU time. The embedded Mali-T628 device, on the other hand, is highly sensitive to the filter details. When 5×5 filters are applied with such a device, 1D implementations require at most 60% of the execution time from 2D ones. As this is an older device, it provides real-time in resolutions as large as 720p with 3×3 filters.

TABLE IV
GPU EXECUTION TIME WITHOUT CPU AND GPU COMMUNICATION [ms]

Device	Implementation	3x3 filters					5x5 filters				
		240p	480p	720p	1080p	4K	240p	480p	720p	1080p	4K
GTX 1080	Int 2D	0.019	0.028	0.046	0.089	0.303	0.023	0.044	0.077	0.156	0.555
	Int 1D	0.017	0.029	0.052	0.105	0.370	0.021	0.040	0.074	0.150	0.543
	Float 2D	0.015	0.023	0.038	0.075	0.245	0.020	0.034	0.060	0.119	0.417
	Float 1D	0.016	0.024	0.042	0.082	0.288	0.020	0.034	0.062	0.137	0.489
RTX 3080	Int 2D	0.013	0.016	0.021	0.034	0.112	0.017	0.022	0.033	0.056	0.199
	Int 1D	0.013	0.015	0.021	0.034	0.103	0.016	0.018	0.027	0.046	0.157
	Float 2D	0.012	0.013	0.018	0.029	0.089	0.015	0.019	0.027	0.046	0.162
	Float 1D	0.012	0.014	0.020	0.032	0.095	0.015	0.017	0.026	0.045	0.156
Mali T628	Int 2D	3.691	12.927	27.657	62.215	246.629	9.455	33.560	71.547	162.950	647.592
	Int 1D	2.326	8.206	17.613	39.563	158.410	4.940	17.597	37.907	85.600	339.687
	Float 2D	3.695	12.906	27.389	62.179	247.237	9.648	34.006	72.953	165.095	650.990
	Float 1D	2.221	7.876	16.724	38.012	151.117	4.949	17.576	37.829	85.408	338.095

TABLE V
AVERAGE PROCESSING TIME [SECONDS] PER FRAME CONSIDERING DIFFERENT ENCODERS

Encoder	240p			480p			720p			1080p			4k		
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
VTM	1.012	0.447	0.062	3.857	1.608	0.091	4.460	1.171	0.132	429.5	101.2	7.6	1648.9	231.6	24.9
x265 @ medium	0.022	0.012	0.009	0.044	0.026	0.020	0.054	0.033	0.026	0.070	0.038	0.027	0.173	0.094	0.076
x265 @ fast	0.019	0.013	0.008	0.042	0.028	0.018	0.050	0.032	0.028	0.067	0.037	0.027	0.165	0.096	0.074
vvenc @ medium	0.818	0.407	0.084	2.997	1.454	0.214	1.890	0.804	0.183	4.636	1.666	0.296	13.699	4.304	1.052
vvenc @ faster	0.066	0.035	0.046	0.240	0.116	0.064	0.142	0.080	0.053	0.380	0.162	0.087	1.213	0.534	0.341
GPU Filtering*	0.00007 (0.0068% - 0.863%)			0.00022 (0.0056% - 1.2%)			0.00057 (0.0129% - 2.208%)			0.00173 (0.0004% - 6.41%)			0.00901 (0.0005% - 12.2%)		

Using GTX 1080 to apply 5×5 low-pass filter with Float 2D implementation

Table IV presents the execution time for the GPU implementations to assess their performance while ignoring the communication between CPU and GPU. The time required to access the GPU memory during kernel execution is still considered. Comparing the results of different implementations under the same device shows that larger kernels demand larger processing times due to the number of arithmetic operations and memory accesses. Furthermore, for the NVIDIA devices, filters based on floating-point coefficients are faster than their integer-representation counterparts, whereas for Mali-T628, a similar performance is obtained with either representation. This variation is due to modern GPUs having more arithmetic units conceived for floating-point than integer computations [38]. Besides, whereas for Mali-T628 the 1D implementations are almost twice as fast as 2D implementations, the separability poses minor impacts on the execution time of NVIDIA devices. Although the 1D implementations require fewer arithmetic operations, they also demand an extra synchronization point (see Algorithms 1 and 2). When these two factors are considered on the NVIDIA GPUs with thousands of cores, the slowdown from the extra synchronization counters the acceleration of performing fewer operations. Hence, minor performance variations are observed between 1D and 2D implementations. Finally, the communication overhead on Mali-T628 represents a small share of the total time, whereas for NVIDIA devices, communication accounts for most of the time. Clearly, in an embedded computer such as the Odroid XU3, the communication overhead between GPU and main memory is less relevant when compared to conventional computing systems with discrete parts. The small overhead of communication on the Mali T-628 device influences on the variability of results in Table III. Finally, in case the prediction operation was performed directly in GPU, such as in [18], [19], and [20], the execution workload would increase and this communication overhead would be less relevant.

We also compare the total GPU processing time with that of modern encoders to demonstrate the feasibility of incorporating alternative references into existing encoders. Table V presents the processing time per frame, measured in seconds, of some well-known software encoders running in *All Intra* configuration. The Class F videos from the CTCs [30] are not considered as they have multiple resolutions. These experiments are performed on a server with two Intel Xeon Gold 6430 CPUs totaling 128 threads, 128 GB of RAM memory and supported on Ubuntu 24.04.

The x265 encoder [33] implements the High-Efficiency Video Coding (HEVC) standard [13], whereas VTM [24] and vvenc [34] implement the VVC standard. The experiments with x265 and vvenc were performed twice: first with the *medium* preset, and then with the *fast* or *faster* preset. Under the default configurations, both x265 and vvenc perform multithreaded encoding: vvenc uses 8 threads whereas x265 uses all 128 available threads. The processing time of three target bitrates (R1, R2, R3) are provided for each resolution. R1, R2, and R3 correspond to using QPs 22, 37, and 57 in the VTM and vvenc encoders. Conversely, they represent constant rate factors (CRF) of 12, 27, and 47 in the x265 encoder. Finally, **GPU Filtering** represents the processing time to filter a frame with a 5×5 filter following the Float 2D implementation on the NVIDIA GTX 1080 device. This is the most flexible implementation since the coefficients can be set to perform 3×3 or 5×5 filtering, with any coefficient precision. Besides, the numbers between parenthesis represent the share of the encoding time consumed by the filtering operation.

The results from Table V demonstrate that the additional filtering operation represents a negligible processing time when compared to the encoding time in most scenarios. When considering the fast and widely available x265 encoder in its fast preset, the filtering represents at most 12.2% of the encoding time – this happens for the 4k resolution videos. When

smaller resolutions and different encoders are considered, the filtering represents an even smaller share of the encoding time.

The time required to filter a video frame leads to two conclusions. First, if the GPU filtering was introduced as an additional operation in sequence with the encoder, it would increase the total processing time by at most 12.2%. This is a pessimistic estimate since exploring the parallel potential enabled by the alternative references reduces the encoding time. Second, since the GPU filtering operation is faster than the encoding, the GPU can filter frame i while the CPU encodes frame $i-1$, hiding the filtering overhead. In any case, the filtering stage poses a small or negligible overhead. Finally, the negligible processing time overhead imposed by the filtering operation in most scenarios demonstrates that the proposed method can be coupled with existing parallel intra mode decision solutions such as [18], [19], and [20] to provide substantial encoding acceleration with improved coding efficiency.

IX. Conclusion

This work proposed a set of alternative reference samples based on low-pass filters to improve the exploration of massive block-level parallelism during intra prediction. This is the first work to perform a thorough evaluation of the performance of alternative samples, including a discussion of their similarity with regular references, their impacts on coding efficiency, and the processing overhead to obtain such samples.

The experiments discussed in this work show that when compared to the popular strategy of using the original frame samples as references to provide parallelism, the proposed alternative references provide the same parallelism potential with improved coding efficiency. Furthermore, choosing the appropriate alternative based on the target bitrate and video resolution drastically reduces the coding efficiency penalty. Selecting the best alternative for each video and bitrate leads to a coding efficiency penalty of only 0.552% BD-BR, whereas using the original frame samples as references incurs a coding efficiency penalty of 0.870%, on average.

Multiple implementations of the proposed low-pass filters are conceived targeting CPU and GPU devices. Experimental evaluations showed that the proposed alternative references can be produced with minor timing overhead, especially when GPU and multithreaded CPU implementations are considered. Finally, when everything is considered, the alternative reference samples herein proposed provide the same block-level parallelism potential as related works with improved coding efficiency while posing negligible processing time overhead.

In future works, we plan to expand this solution in two directions. In one, we plan to design a mechanism to adaptively choose between a set of fixed filters. It can be achieved by expanding the experiments to a larger dataset and computing the most efficient filters for each resolution-QP pair (useful in computing systems that cannot process the video content to infer the best filters) or exploring machine learning methods to analyze the video content and select filters accordingly. In addition, we plan to develop a method to generate the coefficients based on the video content. Machine learning models are a great candidate to achieve the last goal as well.

References

[1] F. Bossen, K. Sühring, A. Wieckowski, and S. Liu, "VVC complexity and software implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3765–3778, Oct. 2021.

[2] K. Wang, H. Liang, S. Zhang, and F. Yang, "Fast CU partition method based on extra trees for VVC intra coding," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2022, pp. 1–5.

[3] A. Gou, H. Sun, J. Katto, T. Li, X. Zeng, and Y. Fan, "Fast intra mode decision for VVC based on histogram of oriented gradient," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 3028–3032.

[4] G. Wu, Y. Huang, C. Zhu, L. Song, and W. Zhang, "SVM based fast CU partitioning algorithm for VVC intra coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[5] F. Sagrilo, M. Loose, R. Viana, G. Sanchez, G. Corrêa, and L. Agostini, "Learning-based fast VVC affine motion estimation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.

[6] T. L. A. Bubolz, R. A. Conceição, M. Grellert, L. Agostini, B. Zatt, and G. Correa, "Quality and energy-aware HEVC transrating based on machine learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 6, pp. 2124–2136, Jun. 2019.

[7] S. Chatterjee and K. Sarawadekar, "Approximated core transform architectures for HEVC using WHT-based decomposition method," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4296–4308, Nov. 2019.

[8] I. Seidel, M. Monteiro, B. Bonotto, L. V. Agostini, and J. L. Güntzel, "Energy-efficient Hadamard-based SATD hardware architectures through calculation reuse," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 6, pp. 2102–2115, Jun. 2019.

[9] G. Sanchez, M. Saldanha, R. Fernandes, R. Cataldo, L. Agostini, and C. Marcon, "3D-HEVC bipartition modes encoder and decoder design targeting high-resolution videos," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 415–427, Feb. 2020.

[10] B. Silveira et al., "Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 12, pp. 3126–3137, Dec. 2017.

[11] M. M. Corrêa, B. H. Waskow, J. W. Goebel, D. M. Palomino, G. R. Corrêa, and L. V. Agostini, "A high-throughput hardware architecture for AV1 non-directional intra modes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1481–1494, May 2020.

[12] B. Bross et al., "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3736–3764, Oct. 2021.

[13] 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.

[14] Y. Chen et al., "An overview of coding tools in AV1: The first video codec from the alliance for open media," *APSIPA Trans. Signal Inf. Process.*, vol. 9, no. 1, p. 6, 2020.

[15] S. Singh, "Computing without processors: Heterogeneous systems allow us to target our programming to the appropriate environment," *Queue*, vol. 9, no. 6, pp. 50–63, Jun. 2011.

[16] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4130–4134.

[17] N.-M. Cheung, O. C. Au, M.-C. Kung, P. H. W. Wong, and C. H. Liu, "Highly parallel rate-distortion optimized intra-mode decision on multicore graphics processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 11, pp. 1692–1703, Nov. 2009.

[18] S. Radicke, J.-U. Hahn, Q. Wang, and C. Grecos, "A parallel HEVC intra prediction algorithm for heterogeneous CPU+GPU platforms," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 103–119, Mar. 2016.

[19] I. Storch, N. Roma, D. Palomino, and S. Bampi, "GPU acceleration of MIP intra prediction in VVC," in *Proc. 31st Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2023, pp. 600–604.

[20] J. Ma, F. Luo, S. Wang, N. Zhang, and S. Ma, "Parallel intra coding for HEVC on CPU plus GPU platform," in *Proc. Vis. Commun. Image Process. (VCIP)*, Dec. 2015, pp. 1–4.

[21] I. Storch, N. Roma, D. Palomino, and S. Bampi, "Alternative reference samples to improve coding efficiency for parallel intra prediction solutions," in *Proc. IEEE 15th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2024, pp. 1–5.

[22] W. Peng et al., "Overview of screen content video coding: Technologies, standards, and beyond," *IEEE J. Emerg Sel. Top. Circuits Syst.*, vol. 6, no. 4, pp. 393–408, Dec. 2016.
[23] *Hm Hevc Reference Software*. Accessed: Sep. 20, 2024. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/HM
[24] (2024). *VTM VVC Reference Software*. Accessed: Sep. 20, 2024. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM
[25] V. Galiano, H. Migallón, V. Herranz, P. Piñol, O. López-Granado, and M. P. Malumbres, "GPU-based HEVC intra-prediction module," *J. Supercomput.*, vol. 73, no. 1, pp. 455–468, Jan. 2017.
[26] 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.
[27] L. Yu, F. Yi, J. Dong, and C. Zhang, "Overview of AVS-video: Tools, performance and complexity," *Proc. SPIE*, vol. 5960, pp. 679–690, Jun. 2005.
[28] C. C. Chi et al., "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012.
[29] Y.-K. Wang et al., "The high-level syntax of the versatile video coding (VVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3779–3800, Oct. 2021.
[30] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, *VTM Common Test Conditions and Software Reference Configurations for SDR Video*, document Jvet-t2010, Oct. 2020.
[31] R. Gonzales and R. Woods, *Digital Image Processing*. London, U.K.: Pearson, 2018.
[32] J. Pfaff et al., "Intra prediction and mode coding in VVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3834–3847, Oct. 2021.
[33] *X265 Hevc Encoder*, Multicoreware, USA, May 2014.
[34] A. Wieckowski et al., "Vvenc: An open and optimized Vvc encoder implementation," in *Proc. IEEE Int. Conf. Multimedia & Expo Workshops (ICMEW)*, Shenzhen, China, 2021, pp. 1–2.
[35] G. Bjøntegaard, "VCEG-M33: Calculation of average PSNR differences between RD-curves," ITU, Austin, TX, USA, Tech. Rep. VCEG-M33, 2001.
[36] R. Chandra, *Parallel Programming in OpenMP*. New York, NY, USA: Academic, 2001.
[37] *The Opencl Specification, Version 3.0.14*, Khronos Group, USA, Apr. 2023.
[38] *Cuda C++ Programming Guide*, NVIDIA, USA, Oct. 2024.

Iago Storch (Member, IEEE) received the B.S. degree in computer engineering and the M.Sc. degree in computer science from the Federal University of Pelotas (UFPel) in 2018 and 2020, respectively, and the Ph.D. degree in computer science from the Federal University of Rio Grande do Sul (UFRGS), in 2024. His past experience includes six months as an Intern Researcher at the University of Coimbra, Portugal, and another six months as an Intern Researcher at the University of Lisbon, Portugal. He is currently a Post-Doctoral Researcher with the Video Technology Research Group (ViTech), UFPel, and the Secretary of the IEEE Signal Processing Society Rio Grande do Sul Chapter (SPS-RS). His research interests include GPU and CPU parallelization, video coding, immersive videos, and the encoding of light field images.

Nuno Roma (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisbon, Portugal, in 2008. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, IST, and a Senior Researcher at the Instituto de Engenharia de Sistemas e Computadores Research and Development (INESC-ID). His research interests include computer architectures, specialized and dedicated structures for digital signal processing, energy-aware computing, parallel processing, and high-performance computing systems. He contributed to more than 130 manuscripts in journals and international conferences. He served as a Guest Editor of *IEEE Design & Test*, *Journal of Real-Time Image Processing* (Springer), and *EURASIP Journal on Embedded Systems*. He has a consolidated experience in funded research project leadership and is a member of the HiPEAC network of excellence. He is a Senior Member of the IEEE Circuits and Systems Society and ACM.

Daniel Palomino (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the Federal University of Rio Grande do Sul (UFRGS) in 2013 and 2017, respectively. Since 2014, he has been a Professor at the Center for Engineering (CENG), Federal University of Pelotas (UFPel), Brazil. He is also a Lead Researcher at the Video Technology Research Group (ViTech) and a Faculty Member at the Postgraduate Program in Computer Science (PPGC), UFPel, where he advises M.S. and Ph.D. students. His international experience includes a six-month research internship at Karlsruhe Institute of Technology (KIT), Germany, and an 11-month tenure as a Visiting Professor at the University of Lisbon, Portugal. He has authored or co-authored over 90 papers in international journals and conferences, focusing on efficient video coding systems and machine learning applications for image and video processing. His research interests include power-efficient computing systems, hardware architectures, and image and video coding algorithms. He is the Chair of the IEEE Signal Processing Society Rio Grande do Sul Chapter (SPS-RS) and a member of the IEEE CAS Visual Signal Processing and Communications Technical Committee (VSPC-TC) and the Brazilian Committee on Audio, Image, Multimedia, and Hypermedia Coding. He served as a Guest Editor for IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS and currently serves as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Sergio Bampi (Senior Member, IEEE) received the B.Sc. degree in electronics and the B.Sc. degree in physics from the Federal University of Rio Grande do Sul (UFRGS), in 1979, and the M.Sc. and Ph.D. degrees in electrical engineering from Stanford University, in 1982 and 1986, respectively. He was the Technical Director of the Microelectronics Center, CEITEC, from 2005 to 2008; and the President of the FAPERGS Research Funding Foundation. He was a Visiting Research Professor at Stanford University from 1998 to 1999 and the Director of the National Supercomputer Center from 1993 to 1996. He is currently a Full Professor with the Institute of Informatics, UFRGS. His research interests include IC design and modeling, mixed-signal and RF CMOS circuit design, architectures and SoCs for image and video processing, nano-CMOS devices, ultra-low power digital CMOS design, and dedicated VLSI architectures for DSP. He has co-authored more than 500 articles in the above fields and in MOS devices, circuits, technology, and CAD. He served as the President and the VP for the scientific societies SBPC-RS and SBMICRO from 2002 to 2006.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614