Journal of Real-Time Image Processing (2023) 20:50
https://doi.org/10.1007/s11554-023-01308-5

RESEARCH

Complexity and compression efficiency analysis of *libaom* AV1 video codec

Isis Bender[1] · Alex Borges[2] · Luciano Agostini[2] · Bruno Zatt[2] · Guilherme Correa[2] · Marcelo Porto[2]

Received: 24 February 2023 / Accepted: 19 April 2023 / Published online: 5 May 2023

Abstract

The recent research effort aiming to provide a royalty-free video format resulted in AOMedia Video 1 (AV1), which was launched in 2018. AV1 was developed by the Alliance for Open Media (AOMedia), which groups several major technology companies such as Google, Netflix, Apple, Samsung, Intel, and many others. AV1 is currently one of the most prominent video formats and has introduced several complex coding tools and partitioning structures in comparison to its predecessors. A study of the computational effort required by the different AV1 coding steps and partition structures is essential for understanding its complexity distribution when implementing fast and efficient codecs compatible with this format. Thus, this paper presents two main contributions: first, a profiling analysis aiming at understanding the computational effort required by each individual coding step of AV1; and second, a computational cost and coding efficiency analysis related to the AV1 superblock partitioning process. Experimental results show that the two most complex coding steps of the *libaom* reference software implementation are the inter-frame prediction and transform, which represent 76.98% and 20.57% of the total encoding time, respectively. Also, the experiments show that disabling ternary and asymmetric quaternary partitions provide the best relationship between coding efficiency and computational cost, increasing the bitrate by only 0.25% and 0.22%, respectively. Disabling all rectangular partitions provides an average time reduction of about 35%. The analyses presented in this paper provide insightful recommendations for the development of fast and efficient AV1-compatible codecs with a methodology that can be easily replicated.

Keywords AV1 · Libaom · Complexity analysis · Time profiling

1 Introduction

Video coding is an important research area recently boosted by the increasing demand for Ultra-High Definition (UHD) digital video in applications such as digital television broadcasting, video streaming, and video conferencing. YouTube is an excellent example of a video streaming service, having over 2 billion users per month [32] accessing high-quality content. Besides, the consumption of streaming video has dramatically increased due to the Covid-19 pandemic. In March 2020, the Bitmovin company reported that the number of videos played increased by 118% compared to February 2020, and the time of watched videos was increased by 220% [4]. Another study shows that the average growth of Facebook live streaming was up to 200% in Europe and 300% in the USA [5].

Uncompressed digital videos require large amounts of bits to be stored or transmitted over the Internet. Therefore, video compression is essential. Besides that, the

✉ Isis Bender
idbender@inf.ufpel.edu.br

Alex Borges
amborges@inf.ufpel.edu.br

Luciano Agostini
agostini@inf.ufpel.edu.br

Bruno Zatt
zatt@inf.ufpel.edu.br

Guilherme Correa
gcorrea@inf.ufpel.edu.br

Marcelo Porto
porto@inf.ufpel.edu.br

[1] Video Technology Research Group (ViTech), Federal University of Pelotas (UFPel), Pelotas, Brazil

[2] Federal University of Pelotas, Pelotas, Brazil

increase in spatial image resolution, frame rate, and pixel bit depth encourages the development of new video formats and standards seeking better compression efficiency. Many video coding standards have been developed in the last decades, mostly by standardization groups such as the International Standards Organization (ISO), the International Engineering Consortium (IEC), and the Telecommunication Standardization Sector (ITU-T). These standardization groups were responsible for the development of the MPEG-1 [15], H.262/MPEG-2 [16], H.263 [17], H.264/AVC [18], and H.265/HEVC [19]. The most recent standard is the H.266/VVC [20], which was released in 2020. These video formats share a commercial characteristic: a complex licensing and royalty payment system.

Google, one of the world's leading internet companies, embarked on the WebM project [34] to develop open-source and royalty-free video codecs. The first codec released as part of the project was entitled VP8, which was later succeeded by VP9 [28]. In 2015, several high-tech companies, such as Amazon, Google, and Meta, joined the effort and founded the Alliance for Open Media (AOMedia) [3], which is responsible for the development of a new royalty-free codec to compete with HEVC, named AOMedia Video 1 (AV1) [14]. Since its release and the publication of the official format specification [29], in 2018, encoder and decoder prototypes based on the AV1 format have been developed by AOMedia partners. The main one is the AV1 Codec Library (*libaom*) [2], which was developed by AOMedia.

The *libaom* encoder development is based on the proprietary VP9 video coding scheme [25], but many other compression tools have been included in the project in the last years to increase its compression efficiency. Table 1 compares some AV1 and VP9 tools. It is possible to observe the increase in the number of tools supported by the AV1 encoder in comparison to its predecessor in several coding steps, such as intra-frame prediction, inter-frame prediction, transform, and filters. Besides, there was also a significant increase in the amount and types of block partitioning possibilities. AV1 uses a 10-way tree structure for block partitioning [7], with six more possibilities than VP9. In the AV1 encoder, a video frame can be partitioned into superblocks (SBs) of size 128×128 [7], while in VP9 the maximum block size is 64×64 [24]. AV1 adopts 56 directional modes in the intra-frame prediction, whereas VP9 uses only eight modes [14]. In the inter-frame prediction, VP9 uses five combinations of reference frames [8], while AV1 includes 16 options [14].

Section 2 discusses in detail these novel characteristics of AV1. Notice that the new tools presented in Table 1 increase the encoding mode possibilities and the number of decisions required during the encoding process, increasing the AV1 complexity and the time required to encode a video. This can be observed in [23] and [6], which show that the AV1 encoding time can be up to 27 and 21 times slower than the VP9, respectively.

The frame partitioning choice affects different coding steps, which makes the analysis of AV1 partitioning

Table 1 Comparison between AV1 and VP9 video formats

	AV1	VP9
Directional Intra modes	56 (from 36° to 212°)	8 (from 45° to 207°)
Non-directional Intra modes	DC, 3 Smooth, Paeth, 5 Recursive, CfL, Color Pallete, IntraBC	DC and TM (True Motion)
Inter references frames	Up to 7	Up to 3
Types of Inter references frames	LAST_FRAME, GOLDEN_FRAME, ALTREF_FRAME, LAST2_FRAME, LAST3_FRAME, BWDREF_FRAME, ALTREF2_FRAME	LAST_FRAME, GOLDEN_FRAME, ALTREF_FRAME
Reference frames combinations	16	5
Transform block size	4×4, 4×8, 8×4, 4×16, 16×4, 8×8, 8×16, 16×8, 8×32, 32×8, 16×16, 16×32, 32×16, 16×64, 64×16, 32×32, 32×64, 64×32, 64×64	4×4, 8×8, 16×16, 32×32
Transform modes	DCT, ADST, flipADST, IDTX	DCT, ADST, WHT
Transform combinations	16	8
Entropy	Huffman, Multi-symbol	CABAC variation
Reconstruction filters	DBF, CDEF, LRF	Loop Filtering
Range of block sizes	4×4 up to 128×128	4×4 up to 64×64
Allowed block sizes	4×4, 4×8, 8×4, 4×16, 16×4, 8×8, 8×16, 16×8, 8×32, 32×8, 16×16, 16×32, 32×16, 16×64, 64×16, 32×32, 32×64, 64×32, 64×64, 64×128, 128×64, 128×128	4×4, 4×8, 8×4, 8×8, 8×16, 16×8, 16×16, 16×32, 32×16, 32×32, 32×64, 64×32, 64×64
Type of block partitioning	One quadratic, one split, two binary, four ternary, and two quaternary divisions	One quadratic, one split, two binary divisions

structures also relevant. Notice that the use of different block sizes allows the prediction tools to achieve better compression efficiency. Although the use of more block sizes usually guarantees higher compression efficiency, it also leads to higher computational costs and time to perform the compression. In [21] an analysis of the HEVC partitioning structures is presented, which shows the high efficiency of the new partitioning structures over its predecessor, the H.264/AVC. However, a similar analysis for the AV1 encoder is not available in the literature. Thus, the impact of the different partitioning combinations allowed in AV1 still needs to be assessed.

Apart from evaluating the impact of block partitioning on video encoding with AV1, it is equally important to assess the overall computational costs of the encoder. Previous studies, such as [36] and [9], have evaluated the AV1 reference encoder, but they do not present a complete complexity analysis of the AV1 encoder. Instead, they analyzed the computational cost when enabling/disabling certain tools during the video encoding, similarly to the strategy presented in this article for the block partitioning analysis. Therefore, to the best of the authors' knowledge, this is the first article to fully address the complexity analysis of the AV1 encoder.

This work presents an analysis divided into two main contributions. First, we present a complexity analysis of the AV1 reference video encoder, evaluating the computational cost distribution across its encoding steps. Second, we conduct a complexity analysis of the AV1 block partitioning process, evaluating its computational cost and impact on coding efficiency based on 18 different variations of the partitioning structure. These two analyses provide baseline results and a benchmark for developing and evaluating future solutions aimed at reducing and controlling the complexity of the AV1 reference software.

The paper is organized as follows: Sect. 2 presents the AV1 partitioning structure and the general AV1 encoding flow. Section 3 presents the experimental setup for the two analyses presented in this work. Section 4 presents the AV1 complexity analysis, where the assessment methodology for the complexity profiling (Sect. 4.1) and superblock partitioning analysis (Sect. 4.2) are discussed. Section 5 discusses the experimental results. Finally, Sect. 6 concludes this work and presents final remarks.

2 The AV1 encoder

AV1 is based on a hybrid video coding scheme as shown in Fig. 1. The encoding process starts at the prediction steps: intra-frame and inter-frame predictions. Residual coding data obtained from the subtraction between predicted and original samples are generated and processed by the transform and quantization steps. The entropy coding step then processes the quantized data as well as all lateral signaling information, such as prediction modes and motion vectors. To allow for compatible inter-frame prediction between the encoder and decoder, a reconstruction loop is required in the encoder, which is composed of the inverse transform, inverse quantization, and filter steps, as presented in Fig. 1.

The intra-frame prediction of AV1 presents significant improvements and innovations when compared to its predecessor, supporting a total of 69 prediction modes (59 more

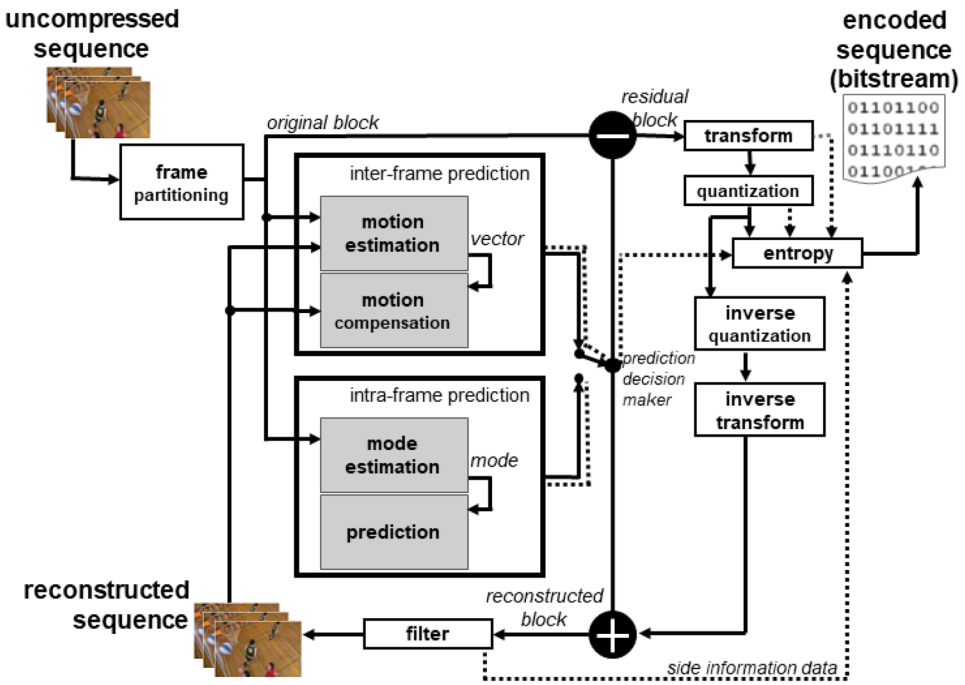

Fig. 1 Generic overview of a hybrid video encoder

than VP9). The AV1 directional intra-frame prediction is similar to the one found in VP9, but the number of prediction angles is expanded to 56. Besides, it includes the following non-directional predictors (in addition to the DC mode): three smooth types, the Paeth mode, five recursive filtering modes, the Croma from Luma (CfL) mode, the Intra Block Copy (IntraBC) mode, and the Color Palette mode [14].

The AV1 inter-frame prediction extends the number of reference frames to up to seven, i.e., four reference frames more than VP9. The available reference frames are:

- *LAST*, *LAST2*, *LAST3*: The neighboring past frames;
- *GOLDEN*: The past frame, which is referenced several times during the encoding process;
- *ALTREF*: A future reference built through filtering;
- *ALTREF2*: An intermediate filtered future reference between *GOLDEN* and *ALTREF*;
- *BWDREF*: A future reference obtained without applying a filter.

In the compound prediction, AV1 allows 11 different combinations of reference frames beyond the allowed in VP9. AV1 also introduces five advanced compound predictions, the warped motion compensation, the overlapped block motion compensation (OBMC), and the dynamic spatial and temporal motion vector referencing [14].

The AV1 transforms are also more diverse than those available in VP9. VP9 supports three transform modes, while AV1 has support for four: DCT (Discrete Cosine Transform), ADST (Asymmetrical Discrete Sine Transform), flipADST, and IDTX (Identity Transform) [18]. These transforms can be combined in 16 ways in AV1, while in VP9 this number is only 4 [24]. The AV1 also allows 19 transform block sizes, 15 more than in VP9.

In the entropy coding step, AV1 uses a symbol-to-symbol adaptive multi-symbol arithmetic coder [14], whereas VP9 employs a tree-based Boolean non-adaptive binary arithmetic encoder to encode all syntax elements. Besides, AV1 includes three in-loop filters (two filters more than in VP9), called Deblocking Filter (DBF), Constrained Directional Enhancement Filter (CDEF), and Loop Restoration Filters (LRF) [14].

The VP9 maximum superblock (SB) size is 64×64 and it allows a recursive partitioning down to 4×4 blocks in a 4-way tree structure [24]. The AV1 overview [14] and specification [29] show that the AV1 superblock partitioning structures are more complex than VP9. In AV1, the SB represents an area of up to 128×128 samples of the frame. Each SB has a quadtree structure for partitioning, with up to six depth levels, as shown in Fig. 2. Every node of this SB quadtree presents a partitioning tree, called *PC TREE* (in black), followed by a transform tree (in red). The *PC TREE* defines the block size used in the prediction step, while the transform tree defines the block size and type partitioning used in the transform step. AV1 allows block sizes from 128×128 down to 4×4 samples and it supports ten block formats, six formats beyond the supported in VP9.

Each candidate block in AV1 can be split into up to ten possibilities of partition types, as can be observed in Fig. 3.

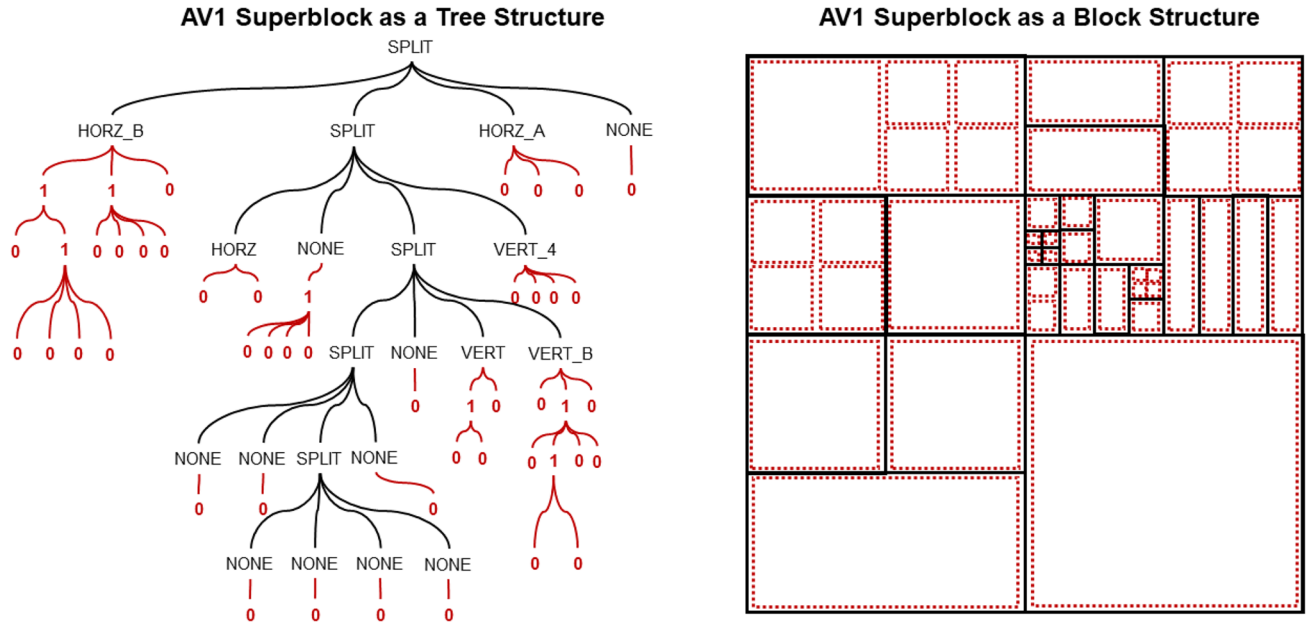

Fig. 2 AV1 superblock partitioning structure (**a**) as a tree and (**b**) as a set of blocks. Partition trees are colored in black for prediction and red for transforms

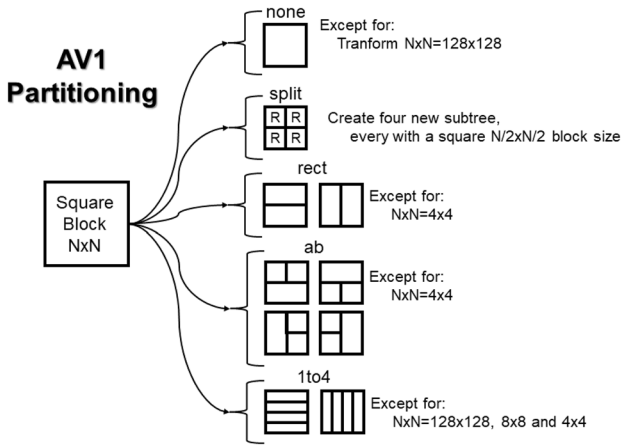

Fig. 3 AV1 partitioning structure for prediction and transform steps

The blocks can be symmetrical or asymmetrical and they can assume two different geometric shapes: squares or rectangles. AV1 allows the following partition types:

- *NONE*: the block is not split, keeping the 1:1 ratio of the current squared size;
- *HORZ* or *VERT*: the block is split into two parts with 1:2 or 2:1 ratio;
- *HORZ_A*, *HORZ_B*, *VERT_A*, and *VERT_B*: the block is split into three parts, composed of two quadratic blocks and a rectangular block;
- *VERT_4*, *HORZ_4*: the block is partitioned into four parts with 1:4 or 4:1 ratio, an innovation when compared to VP9 [6];
- *SPLIT*: this is a particular node of the partition tree, which recursively creates four new subtrees with half the size of the current tree node size. This partition type is not applied to a 4×4 node size.

The partitioning tree of the transform step presents some particularities. Transform partitioning rules applied to predicted blocks with the inter-frame prediction can be seen in Fig. 4. The transform tree can have up to two levels, and each tree branch can be partitioned. Besides, the *SPLIT* type is not applied exclusively in quadratic blocks, as occurs in the prediction step. Regardless of the prediction type applied to the block, its transform size is the same as the predicted block size up to 64×64. Larger blocks must be split to the maximum size of 64×64 to be processed in the transform step.

The complexity involved in the AV1 encoding process is significantly higher than its predecessor, the VP9. AV1 has an expressive number of tools in each of its different coding steps. Knowing the complexity of each coding step and how much it represents in the total coding time is very important since it allows researchers to understand where to focus the effort to develop fast and efficient solutions in the future.

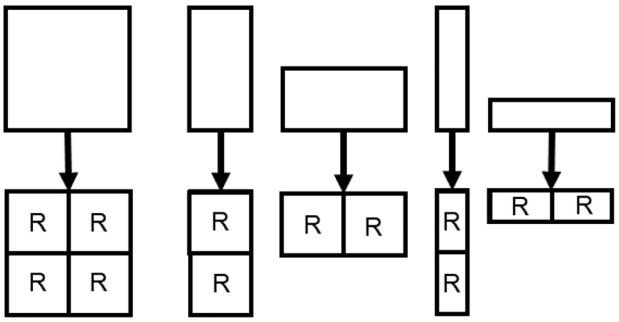

Fig. 4 Subdivisions allowed in the transform tree

This kind of evaluation could be also beneficial for the design of dedicated hardware for AV1 since it is relevant to know in advance the AV1 tools' complexity distribution and the most computationally costly ones. So, it is possible to focus project decisions on coding steps and/or tools with the most acceleration potential. Besides, it is extremely relevant to analyze the computational cost and the coding efficiency related to the AV1 block partitioning. By understanding its behavior, fast decision solutions can be proposed to avoid the evaluation of many blocks partitioning possibilities, reducing the complexity and also hardware demand (area and power/energy).

3 Experimental setup

In this paper, two different analyses of the AV1 reference software *libaom* are presented. The first one focuses on the complexity of the AV1 encoding steps and is based on a profiling assessment. The second analysis aims at observing the complexity and coding efficiency associated with the AV1 superblock partitioning process. The experimental setup and methodology are shared between these two analyses, as discussed in the next paragraphs. The encodings were performed using the 60 first frames of four HD1080 (1920×1080 pixels) and four UHD4K (4096×2160 pixels) video sequences, all of them recommended in the IETF-NETVC-Testing document [10]. The same encoder configuration used in [14] was employed in the experiments. To guarantee that the sequences differ from each other in both spatial and temporal characteristics, an analysis of the spatial activity index (SI) and the temporal activity index (TI) [35] was performed. Figure 5 shows the resulting SI (x-axis) and TI (y-axis) of 21 HD1080 video sequences (Fig. 5a) and eight UHD4K video sequences (Fig. 5b). According to this SI-TI analysis, the following HD1080 sequences were chosen: *crowd_run*, *guitar_hdr_amazon*, *Netflix_Crosswalk*, and *Netflix_TunnelFlag*. The same analysis allowed choosing the following UHD4K

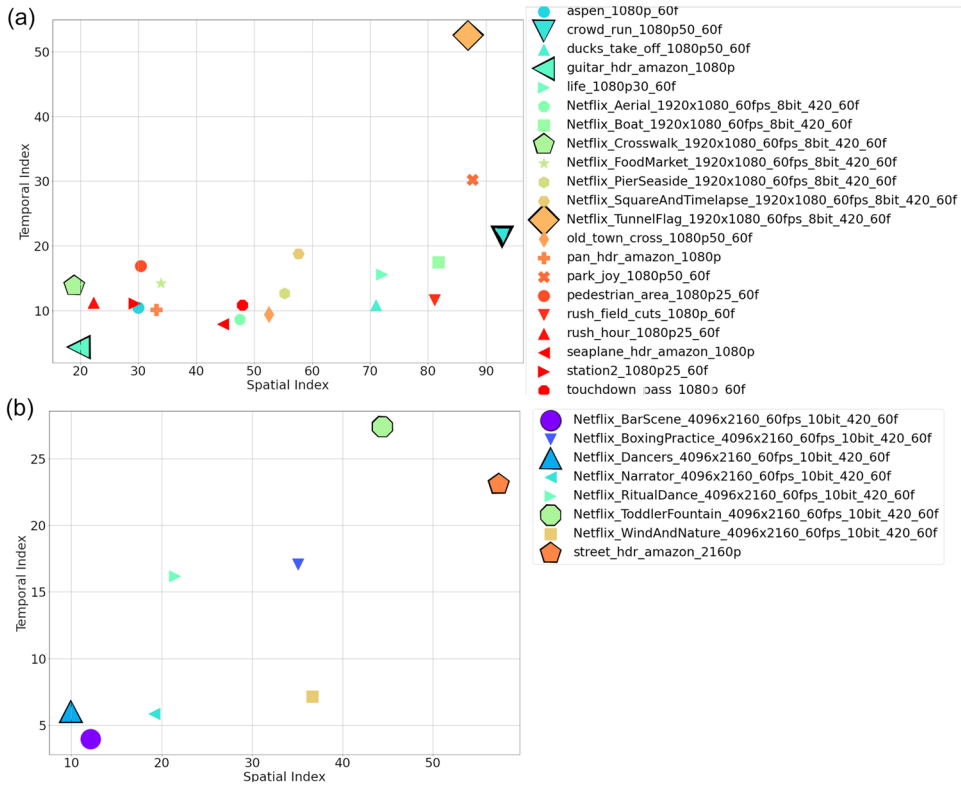

Fig. 5 Spatial Index and Temporal Index for (**a**) HD1080 videos and (**b**) UHD4K videos

video sequences: *Netflix_BarScene*, *Netflix_Dancers*, *Netflix_ToddlerFountain*, and *street_hdr_amazon*.

As recommended in the IETF-NETVC-Testing document [10] and the AV1 overview paper [6], all encodings performed with *libaom* followed the command line below, where CQ is an acronym for Constrained Quality and it assumes the values 20, 32, 43, and 55.

```
-verbose -psnr -lag-in-frames=19 -
passes=2 -frame-parallel=0 -tile-
columns=0 -cpu-used=0 -threads=1 -
kf-min-dist=1000 -kf-max-dist=1000
-end-usage=q -cq-level={CQ}
```

The experiments of the AV1 complexity profiling analysis encoding steps were run on an Intel Xeon E5645 processor with six 2.4 GHz cores and 24GB of RAM, using the *libaom* hash code `cd653f1`. For the experiments related to complexity and efficiency analysis of the AV1 superblock partitioning structure, an Intel Xeon E5-4650 v3 server with eight 2.10 GHz cores and 512GB of RAM was used, running the *libaom* hash code `a5e3f02`. Coding efficiency results were calculated in terms of Bjøntegaard-Delta (BD) rate metrics using the average Peak Signal-to-Noise Ratio (PSNR) value from luminance and chrominance (blue and red) components.

4 AV1 complexity analysis

This section presents the complexity analysis of the AV1 encoding steps. The Sects. 4.1 and 4.2 present, respectively, the assessment methodology for the complexity profiling analysis and superblock partitioning analysis of the AV1.

4.1 Complexity profiling

The GPROF [12] profiler was used to analyze the *libaom* complexity because the AV1 reference software offers native support to this profiling tool. Moreover, GPROF has been widely used for complexity analysis of other video codecs, such as HEVC ([13] and [30]) and VVC [30]. As AV1 is based on the hybrid video coding scheme described in Sect. 2, the following coding steps are considered in the analysis: intra-frame prediction, inter-frame prediction, transform, quantization, entropy coding, and filtering.

GPROF accesses the private memory of software under analysis during its execution, while this software reads and writes data in its memory space. After the software execution, GPROF creates a file description of the used

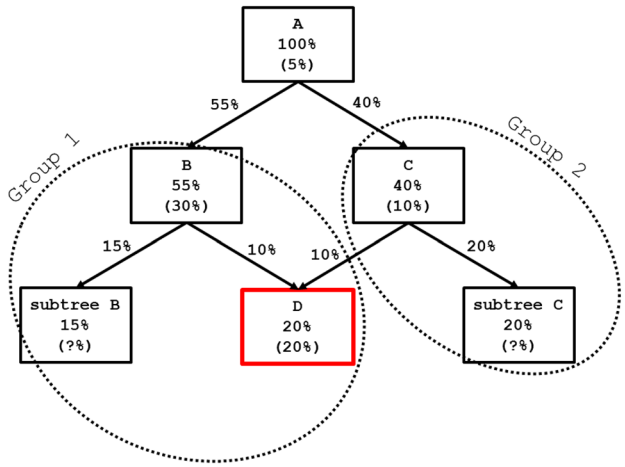

Fig. 6 Example of chart used in the GPROF analysis

functions, containing information that identifies their execution time and calls. To allow for a clear understanding of the obtained data, the *gprof2dot* tool [11] was used to transform information extracted from *libaom* into a graphical representation.

The graphical representation created by the *gprof2dot* tool [11] comprises a tree that shows the hierarchy of the functions, composed of nodes and edges, as shown in the example of Fig. 6. Each node represents a function of the analyzed software and within each node, the following information can be observed:

- The name of this function;
- The execution time (as a percentage of the overall execution time) of this function plus its subtrees (callee functions);
- The execution time of this function by itself;
- The number of times that this function was executed.

The edges in Fig. 6 indicate the internal calls to other functions and present two pieces of information: the percentual time used by the caller function to execute the callee function; and the percentual time that the caller executed the callee function. It is important to note that the child node usually presents a smaller percentage of time in comparison to its parent node, except for the cases when several parent nodes call the same child node. In these cases, the callee function may present a percentage time greater than their parents.

Due to the number of functions implemented in the *libaom* software (over a thousand), a full graphical representation is not suitable. Thus, just the functions that present relevant processing times are presented by the *gprof2dot* tool [11]. A thorough analysis of all results obtained with the GPROF was performed to identify which of the main *libaom* functions correspond to each AV1 coding step. This analysis revealed that some child nodes are shared by several parent nodes, which means that some functions are used in more than one coding step. This fact makes it difficult to obtain the exact percentage time share of these functions that correspond to each coding step.

Therefore, to obtain the approximate encoding time of each step, the main *libaom* functions of each AV1 coding step were identified, as can be observed in Table 2. The execution time percentage of an encoding step is a sum of the execution time percentage of nodes present in Table 2, subtracted from the sum of the percentage time present in the edges of these nodes.

4.2 Superblock partitioning analysis

As discussed before, a greater number of partitioning possibilities are observed in the AV1 encoder when compared to VP9, which significantly increases the number of prediction choices for every single region of the video. Defining the best block size demands a very large number of tests, but the computational cost in terms of encoding time associated with this task is unknown. Besides, the coding efficiency gains obtained from each different partitioning possibility also need to be evaluated for a better understanding of the AV1 encoding process.

Using the provided *libaom* tools to limit the AV1 partitioning possibilities, eighteen different configurations were defined for this analysis. Table 3 summarizes these variations, where the anchor configuration (00) employs the

Table 2 Key functions for each *libaom* coding step

Coding step	Key function name
Intra-Frame	av1_rd_pick_intra_mode_sb
	av1_rd_pick_intra_sbuv_mode
	rd_pick_intra_angle_sby
	av1_super_block_yrd
	av1_super_block_uvrd
	av1_encode_intra_block_plane
Inter-Frame	av1_rd_pick_inter_mode_sb
Transform	av1_foreach_transformed_block_in_plane
	search_txk_type
	av1_xform_quant
Quantization	av1_quantize_fp_facade
	av1_quantize_b_facade
	av1_highbd_quantize_fp_facade
	av1_highbd_quantize_b_facade
Filter	av1_pick_filter_restoration
	av1_cdef_search
Entropy	av1_get_entropy_contexts
	av1_get_txb_entropy_context

Table 3 List of experiments varying the *libaom* partitioning

Config.	Description
0	Baseline/Anchor
1	- -enable-rect-partitions=0
2	- -enable-ab-partitions=0
3	- -enable-1to4-partitions=0
4	- -sb-size=64
5	- -min-partition-size=64 - -max-partition-size=128
6	- -min-partition-size=32 - -max-partition-size=128
7	- -min-partition-size=16 - -max-partition-size=128
8	- -min-partition-size=8 - -max-partition-size=128
9	- -min-partition-size=32 - -max-partition-size=64
10	- -min-partition-size=16 - -max-partition-size=64
11	- -min-partition-size=8 - -max-partition-size=64
12	- -min-partition-size=4 - -max-partition-size=64
13	- -min-partition-size=16 - -max-partition-size=32
14	- -min-partition-size=8 - -max-partition-size=32
15	- -min-partition-size=4 - -max-partition-size=32
16	- -min-partition-size=8 - -max-partition-size=16
17	- -min-partition-size=4 - -max-partition-size=16
18	- -min-partition-size=4 - -max-partition-size=8

default *libaom* parameters described in Sect. 3. The proposed configurations limit the allowed partitioning types (experiments 01–03), modify the SB size from 128×128 to 64×64 (experiment 04), and control the allowed partition tree depth (experiments 05–18). The partition depth delimitation was configured by changing the maximum and minimum quadratic block size allowed, always observing the following rule: the maximum block size must be greater than the minimum block size.

As previously described in Sect. 2, two partitioning trees are used in the AV1, one for the prediction and one for the transform step. The root node of the transforms trees is the same as the prediction leaf node, except when the block size is larger than 64×64. Besides, the choice of transform kernel depends on the prediction mode observed during the encoding process [33]. In this context, it is important to note that all configurations defined in the experiments affect both prediction trees and transform trees.

5 Experimental results

This section discusses the results obtained for both analyses described in Sect. 4.

5.1 Complexity profiling results

Tables 4 and 5 present the obtained results for HD1080 and UHD 4K resolution, respectively. These tables show the relative time (RT), in percentage, and the absolute time (AT), in hours, for each AV1 video coding step, considering the four recommended CQ values [10].

The intra-frame prediction, quantization, filters, and entropy coding steps correspond jointly to 3.48% and 4.84% of the total *libaom* execution time for HD1080 and UHD4K videos, respectively. The execution time of the intra-frame prediction corresponds to only 0.76% and 1.72% for HD1080 and UHD4K videos, respectively. These values were already expected, since other video encoder analyses (e.g., [13]), had already shown a time share of 3% for intra-frame prediction when encoding HD videos. This small portion of time for the intra-frame prediction can be explained by the fact that other tools, such as the inter-frame prediction and the transform steps, require much longer execution times.

It should be noted that in other encoders the relative complexity of the intra-frame step is more expressive. In the HEVC, it corresponds to 3% of the total encoding time [13]. In VVC this step requires an encoding time 108 times larger than in HEVC [30].

The quantization step is controlled by the CQ value, which directly impacts the coding efficiency. It is a simple process that attenuates the transformed coefficients

Table 4 Profiling results for HD1080 resolution

Coding Step	CQ Level							
	20		32		43		55	
	RT (%)	AT (h)	RT (%)	AT (h)	RT (%)	AT (h)	RT (%)	AT (h)
Intra-Frame	0.38	0.13	0.52	0.15	0.83	0.16	1.31	0.18
Inter-Frame	71.59	25.20	77.45	22.46	79.5	15.42	79.38	11.19
Transform	24.96	8.79	19.80	5.74	17.28	3.35	16.11	2.27
Quantization	1.80	0.63	1.09	0.32	0.68	0.13	0.54	0.08
Filters	1.24	0.44	1.13	0.33	1.71	0.33	2.65	0.37
Entropy	0.04	0.01	0.02	0.01	0.01	0.00	0.01	0.00
Total	100	35.20	100	29.01	100	19.39	100	14.09

Table 5 Profiling results for UHD4K resolution

Coding Step	CQ Level							
	20		32		43		55	
	RT (%)	AT (h)	RT (%)	AT (h)	RT (%)	AT (h)	RT (%)	AT (h)
Intra-Frame	2.69	6.13	1.28	1.77	0.68	0.57	2.22	1.44
Inter-Frame	51.10	116.51	62.06	85.58	71.79	60.16	72.81	47.18
Transform	43.60	99.41	34.12	47.05	24.64	20.65	20.52	13.30
Quantization	1.69	3.85	1.72	2.37	1.20	1.01	0.84	0.54
Filters	0.87	1.98	0.79	1.09	1.68	1.41	3.60	2.33
Entropy	0.05	0.11	0.03	0.04	0.01	0.01	0.01	0.01
Total	100	227.99	100	137.9	100	83.81	100	64.8

according to the CQ value, so it has a low percentage of the total encoder complexity: around 1% of the encoding time for both resolutions.

The filters require 1.68% and 2.57% of the *libaom* encoding time for HD1080 and UHD4K, respectively. The encoding time demanded by this coding step enhances as the CQ level increases. This is because the higher the CQ, the greater the clipping (quantization) in the transformed coefficients, causing information loss in the residual blocks. Thus, filtering for artifact removal and smoothing becomes more important in these cases.

As the entropy coding processes the incoming data linearly with no need for testing different operation modes, this step requires only about 0.2% of the total *libaom* encoding time for both resolutions.

As expected, the inter-frame prediction presents the highest computational cost among all steps. This can be seen in the results presented in Tables 4 and 5, which show that at least 51.10% of the encoding time is dedicated to the inter-frame prediction in both resolutions. This value can reach up to 79.50% in some cases. On average, this step requires 76.98% and 64.44% of the execution time for HD1080 and UHD4K videos, respectively.

The transform step presents between 16.11 and 43.60% of the total time of the *libaom* execution. On average, it requires 19.54% and 30.72% of the execution time for HD1080 and UHD4K sequences, respectively. Notice that the transform encoding time expressively changes according to the CQ level employed. In the HD1080 resolution, the difference in the encoding time between CQ 20 and CQ 55 is 8.85 percentual points. In the UHD4K resolution, this value is even more expressive, with 23.08 percentual points.

Figure 7 presents an analysis considering the average results between all HD1080 videos for all CQ values. For the inter-frame step, three groups of functions present the highest execution time: convolve, compound, and subpixel functions, presenting an average of 23.60%, 21.31%, and 4.91% for CQ 20, respectively. These values remain approximately the same throughout all CQs, as can be seen in the orange bars in Fig. 7. The first function is related to the mathematic convolution operation ([31] and [26]), which applies a subsampled interpolation by two one-dimensional convolutions. First, a horizontal filter is used to build up a temporary array, and then this array is vertically filtered to obtain the final prediction [29]. The second function is related to the AV1 compound mode, which considers eight candidate reference frames as a combination of a forward and a backward prediction [22]. The third function is responsible for the search in fractional pixels of the AV1.

In the transform step, the inverse DCT and direct DCT are the most complex functions, representing an average of 3.08% and 1.62% of the total encoding time in CQ 20, respectively. It is possible to observe that the DCT is the most time-demanding transform mode. This is expected since the DCT type 2 (DCT-2) and the ADST are the most frequently selected primary transforms on intra blocks [37].

Table 6 presents a comparison between the computational cost of the AV1 coding steps and the corresponding steps in the HEVC [1] and VVC [27] reference encoders. The results represent the average values obtained for the four QPs in HEVC and VVC and the four CQ levels in AV1. It is worth noting that [1] and [27] consider a different set of video sequences in the analysis. Also, they combine the transform and quantization computational costs into a single step, so the results for these two steps presented in Table 6 are also combined for AV1. Additionally, the results presented for VVC in Table 6 are approximate percentage values as this information was not directly provided in [27].

The encoding steps with the highest computational cost in AV1, as observed in this study, are the inter-frame prediction and transform, which is consistent with HEVC and VVC results. However, there is a notable difference in the computational cost of the intra-frame prediction and entropy coding, which are significantly lower in AV1 compared to the other two formats. This could be attributed to the way AV1 handles the frame organization structure, where

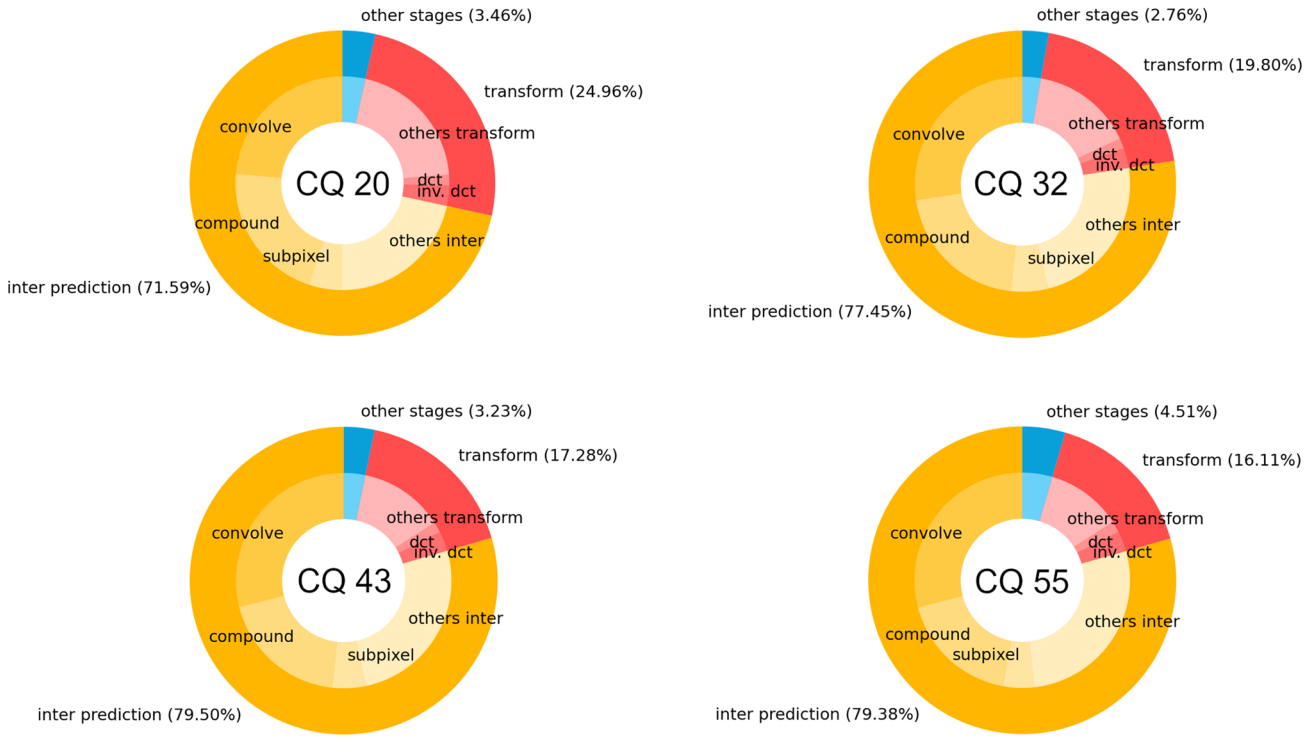

Fig. 7 Distribution of the execution time percentage of each coding step (HD1080 average)

a dynamically sized Group of Pictures (GoP) is typically used with a tendency towards 16 frames. Furthermore, the keyframe frame, which is the only one encoded entirely with intra-frame prediction, is applied only once according to the recommended settings for AV1. Therefore, the relatively low computational cost of the intra-frame prediction in AV1 is reasonable. Additionally, the entropy coding step of AV1 is relatively simpler than that of HEVC and VVC, which accounts for the lower computational cost observed in AV1.

The results obtained in this analysis show that the *libaom* complexity is distributed in a very heterogeneous way among the coding steps. Based on the presented results, it is possible to affirm that solutions aiming at reducing the complexity of the inter-frame and the transform steps tend to present expressive results. In contrast, optimizations on the intra-frame prediction, quantization, filters, and entropy coding steps lead to minimal impact on the whole complexity of the AV1 *libaom* software implementation, since they represent only 4.16% of the encoding computational cost.

Table 6 Profiling comparison among AV1, HEVC, and VVC encoders (average for four QPs and CQs)

Coding Step	HEVC (%) [1]	VVC (%) [27]	AV1 (%) ours
Intra-Frame	2.25	7.25	0.76
Inter-Frame	79.03	55.50	76.98
Transform/Quantization	14.48	19.25	20.57
Filters	0.18	6.50	1.68
Entropy	2.98	5.75	0.02

5.2 Superblock partitioning analysis results

This subsection presents the results of the complexity and coding efficiency analysis of the AV1 superblock partitioning structure.

The encoding time reduction of each configuration (TR_C) is calculated according to Eq. (1), where the configuration encoding time (T_C) is compared to the encoding time of the baseline configuration (T_{00}).

$$TR_C = 100 * \left(1 - \frac{T_C}{T_{00}}\right). \quad (1)$$

Table 7 shows the obtained results for all configurations considering HD1080 and UHD4K resolutions. The average results for each resolution can also be graphically seen in Fig. 8. Configuration 01 disables the rectangular partitioning type and partition types *ab* and *1to4*. When using this configuration, AV1 can choose only *NONE* partitions as leaf nodes for prediction. This configuration is bound to an expressive computational cost: a TR_C equal to 35.47%, with a moderate coding efficiency loss of 3.39%. While

Table 7 Complexity and coding efficiency results for the SB partitioning analysis (HD1080 and UHD4K), in percentages

Configuration	HD1080		UHD4K		Average	
	BD-Rate	TR_C	BD-Rate	TR_C	BD-Rate	TR_C
1	4.81	36.22	1.97	34.72	3.39	35.47
2	0.39	12.88	0.10	4.73	0.25	8.81
3	0.33	13.20	0.11	3.66	0.22	8.43
4	0.49	15.49	1.08	6.12	0.79	10.81
5	92.68	65.75	19.64	66.86	56.16	66.31
6	30.17	39.81	6.34	45.61	18.26	42.71
7	8.03	24.13	1.34	19.15	4.69	21.64
8	0.45	14.15	0.00	0.16	0.23	7.16
9	30.65	53.57	7.59	55.72	19.12	54.65
10	8.57	35.49	2.43	28.84	5.50	32.17
11	0.95	20.13	1.01	9.21	0.98	14.67
12	0.48	17.40	1.00	9.16	0.74	13.28
13	14.27	49.58	5.36	47.39	9.82	48.49
14	6.49	36.39	4.13	27.54	5.31	31.97
15	6.06	33.61	4.13	27.60	5.10	30.61
16	28.90	60.31	8.00	58.63	18.45	59.47
17	28.35	55.80	8.00	58.61	18.18	57.21
18	103.56	76.66	26.33	81.82	64.95	79.24

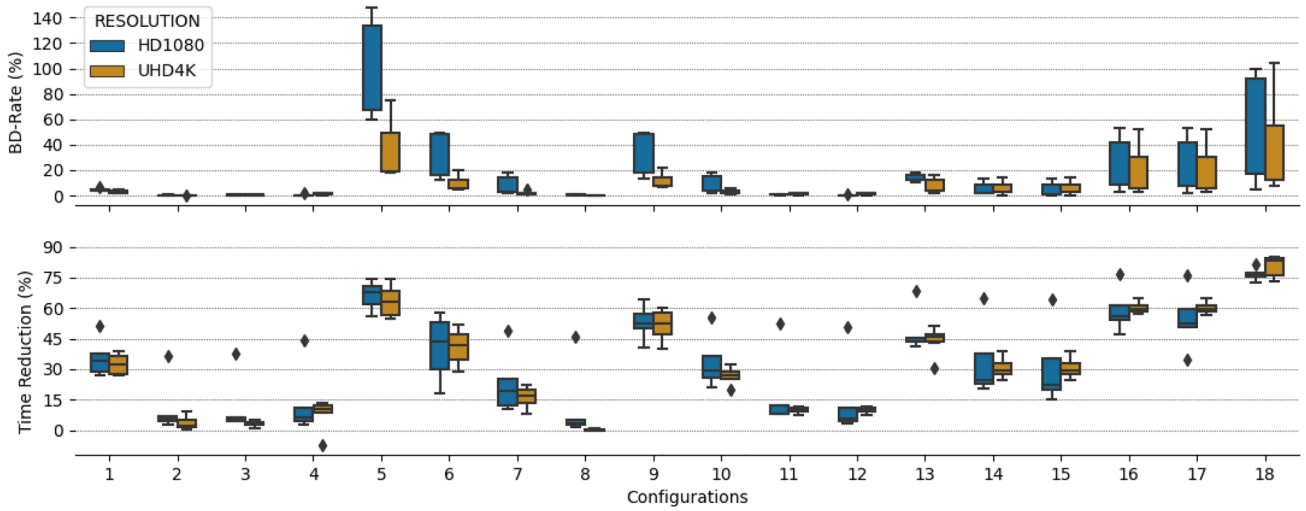

Fig. 8 Average results of the AV1 partitioning analysis for 18 different configurations (HD1080 and UHD4K)

configuration 01 disables three partition types at the same time, configurations 02 and 03 only disable the *ab* and the *1to4* partition types, respectively. These configurations present lower TR_C than configuration 01, but also present better BD-Rate, as expected. The ratio between TR_C and BD-Rate is a fair way to compare the efficiency of these three configurations. The ratio value of the configuration 01, 02, and 03 are 10.46, 35.24 and 38.32, respectively, i.e., disabling only *ab* or *1to4* partition type is more efficient than disabling the rectangular partition type.

From configuration 05–18, the *libaom* partition tree depth is limited, and *libaom* can only choose a set of depths allowed in the configuration. When *libaom* chooses only partitions between 16×16 and 8×8 or between 16×16 and 4×4 (configurations 16 and 17, respectively), it is possible to see an average BD-Rate increase of 18%. The same occurs with configuration 18, which only allows partitions between 8×8 and 4×4. In this case, a severe BD-Rate increase of 64% is noticed. Despite the TR_C values above 50% associated with these configurations, the poor coding efficiency is explained

by the fact that only small partitions are allowed, resulting in a high bitrate to encode the video due to the large amount of lateral signaling information associated to each block. It is important to highlight that the block size 4×4 is disabled in *libaom* for resolutions greater than or equal to UHD4K. In other words, *libaom* uses configuration 08 as default for the UHD4K resolution.

From Table 7, it is possible to observe that configurations 04 and 12 are quite similar, both limiting the encoder to use a minimum block size of 4×4 and a maximum block size of 64×64. However, the RD cost values calculated for configuration 04 starts from block size 64×64, while in configuration 12 this calculation starts from block size 128×128. This occurs because in configuration 04 the superblock size flag is configured to 64×64, and in configuration 12, this flag is 128×128, although this block size is not allowed to be used during the encoding processing. This difference can be seen in the results when configuration 04 shows an average time reduction of 10.81% and a BD-Rate of 0.79%, while configuration 12 leads to a time reduction of 13.28% and a BD-Rate of 0.74%.

The analysis presented in this subsection is important to support decisions for a fast block partitioning solution, aiming to skip some block sizes/formats during the AV1 encoding process. If the focus is just encoding time reduction, configuration 05 represents the best option, promoting an encoding time reduction close to 80% in both resolutions. On the other hand, if the aim is to reduce the complexity by keeping the higher encoding efficiency, configurations 03 and 02 are the best option for the HD1080 and UHD4K resolutions, because they lead to the lowest BD-Rate values, 0.33% and 0.10%, respectively. However, when considering the configurations with the best trade-off between encoding time reduction and encoding efficiency, configurations 02, 04, and 12 are the ones more indicated for the HD1080 resolution, and configurations 01, 03, and 12 are the best ones for the UHD4K resolution.

The presented results show that the partition tree processing and block size decision has a significant impact on the overall computational cost of *libaom*. Thus, it is expected that this process affects several steps of the video coding process, especially the inter-frame prediction, the intra-frame prediction, and the transform step. Therefore, it is interesting to investigate the real impact of this complex decision process in each individual coding step.

Among the eighteen different partitioning configurations analyzed in Sect. 4.2, configuration 01 was identified as the one with the highest computational cost given an acceptable coding efficiency loss (BD-Rate equal 3.39%). This way, the experiments presented in Sect. 4 were repeated with the aid of the GPROF tool, using the four video sequences with HD1080 resolution mentioned in Sect. 3 and the four recommended CQs [10]. Table 8 shows the difference between the original encoding process (configuration 00) and the encoding process that allows only quadratic blocks (configuration 01).

Table 8 Profiling results for HD1080 resolution comparing configuration 01 against original configuration

Coding Step	CQ Level				Average
	20	32	43	55	
Intra-Frame	−0.05	0.24	0.59	1.04	0.46
Inter-Frame	−2.24	−0.64	0.06	−0.36	−0.80
Transform	2.72	0.53	−0.46	−1.19	0.40
Quantization	−0.10	−0.28	−0.43	−0.23	−0.26
Filters	−0.31	0.17	0.27	0.76	0.22
Entropy	−0.01	−0.02	−0.02	−0.02	−0.02

Notice that the relative encoding time of each step differs between the two analyzed configurations. That is, the limitation of the partitioning tree impacts the encoding time of each step, as expected. Positive and negative variations can be observed between the relative values presented, which means that depending on the coding step and CQ used, the imposed limitation increases or decreases the encoding time.

As seen in Table 8, configuration 01 promotes a reduction in the average encoding time of the intra-frame and transform steps. Disabling rectangular blocks reduces the types of blocks to be processed by the encoder and the number of calculations performed in these coding steps. However, in the inter-frame step, the average encoding time is increased, particularly for the encoding under CQ 20. A hypothesis is that it becomes more challenging to find the best candidate blocks without using rectangular blocks, requiring longer searches for the best matching.

6 Conclusions

The main focus of this paper was on analyzing the computational cost associated with the AV1 reference encoder, which was chosen due to its high computational cost and complex mode decision process. The obtained results can be useful in guiding future research aimed at reducing or controlling the complexity of the AV1 encoder. The complexity analyses were performed using the GPROF profiling tool to assess the computational cost required by the different AV1 coding steps in the *libaom* reference software. The computational cost and coding efficiency associated with different AV1 superblock partitioning configurations were also analyzed by evaluating different configurations of its partitioning tree. Each analysis was performed using HD1080 and UHD4K video sequences and all recommended CQ values.

In the first analysis, the profiling results show that the coding steps that require the highest computational effort

are the inter-frame prediction and transform. Inter-frame prediction requires on average 70.71% of the total execution time of *libaom*. The most time-consuming functions at this step are the convolve and compound predictions. The transform step requires an average of 25.13% of the total execution time, and its most complex functions are the inverse DCT and inverse ADST. The obtained results show that these two coding steps are the most indicated for further optimizations and complexity reduction solutions for fast implementations of AV1. The remaining coding steps represent together less than 4.16% of total *libaom* complexity, which leads to the conclusion that they are not critical and do not require significant optimizations when developing fast AV1 codec implementations. It is worth noting that AV1 is a relatively new format that has gained increasing interest from both the industry and scientific community. Therefore, any contributions to the understanding and exploration of this format are valuable. While this study has identified similarities in the computational cost of AV1 with other established formats such as HEVC and VVC, it is essential to recognize that direct comparisons cannot be made due to the distinct characteristics of each format. Overall, the results of this study provide important insights for future research aimed at improving the efficiency of the AV1 encoder.

The second analysis shows that solutions that avoid the ternary *ab* and quaternary *1to4* partition types are the best options in the development of fast AV1 encoders without causing a significant impact on BD-Rate. Disabling *ab* partition types leads to an average BD-Rate impact of 0.25% and an average time reduction of 8.81%. Avoiding *1to4* partition types leads to an average increase of 0.22% of BD-Rate and 8.43% of time reduction. The obtained results also allowed concluding that making use of only small block sizes or only large block sizes is not a good option, since removing the encoder flexibility to deal with block size heterogeneity results in a high encoding time acceleration but lead to a prohibitive BD-Rate cost. In these cases, the BD-Rate value can reach up to 103.56% for HD1080 videos and 26.33% for UHD4K videos.

It is worth noting that AV1 is a relatively new format that has gained increasing interest from both the industry and academy. This is the first work in the literature that deals with the computational costs of all tools of the AV1 reference encoder. Therefore, contributions to the understanding of this format, such as those presented in this paper, are valuable for the definition of novel solutions for AV1 complexity control and reduction, which is important for the deployment of this format in video coding systems.

As future work, the results obtained in the complexity analysis presented in this work will be employed to develop a dynamic complexity control mechanism for the AV1 encoder, which would enable real-time adaptation of the encoding process based on the available computational resources and the desired video quality.

Acknowledgements This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*—Brasil (CAPES)—Finance Code 001, Foundation for Research Support of the State of Rio Grande do Sul (FAPERGS), and National Council for Scientific and Technological Development (CNPq).

Data availability The authors declare that the data supporting the findings of this study are available within the article.

References

1. Ahn, Y., Hwang, T., Yoo, S., Han, W.J., Sim, D.: Statistical characteristics and complexity analysis of HEVC encoder software. J. Broadcast Eng. **17**(6), 1091–1105 (2012). https://doi.org/10.5909/JBE.2012.17.6.1091
2. Alliance for open media: AV1 Codec Library (2015). https://aomedia.googlesource.com/aom/
3. AOMedia: Alliance for open media: The open and royalty-free codec for next-generation ultra high-definition media (2015). http://aomedia.org/
4. Bitmovin: Bitmovin Raises $25 Million to Drive New Innovations in the Video Streaming Industry (2021). https://bitmovin.com/press-room/bitmovin-raises-series-c-25m/
5. Böttger, T., Ibrahim, G., Vallis, B.: How the internet reacted to covid-19: A perspective from facebook's edge network. In: Proceedings of the ACM Internet Measurement Conference, IMC '20, p. 34-41. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3419394.3423621
6. Chen, Y., Mukherjee, D., Han, J., Grange, A., Xu, Y., Parker, S., Chen, C., Su, H., Joshi, U., Chiang, C.H., et al.: An Overview of Coding Tools in AV1: the First Video Codec from the Alliance for Open Media. APSIPA Trans. Signal Inform. Process. **9**, 1–15 (2020). https://doi.org/10.1017/ATSIP.2020.2
7. Chen, Y., Murherjee, D., Han, J., Grange, A., Xu, Y., Liu, Z., Parker, S., Chen, C., Su, H., Joshi, U., Chiang, C., Wang, Y., Wilkins, P., Bankoski, J., Trudeau, L., Egge, N., Valin, J., Davies, T., Midtskogen, S., Norkin, A., de Rivaz, P.: An Overview of Core Coding Tools in the AV1 Video Codec. In: 2018 Picture Coding Symposium (PCS), pp. 41–45 (2018). https://doi.org/10.1109/PCS.2018.8456249
8. Chiang, C., Han, J., Xu, Y.: A Multi-Pass Coding Mode Search Framework For AV1 Encoder Optimization. In: 2019 Data Compression Conference (DCC), pp. 458–467 (2019). https://doi.org/10.1109/DCC.2019.00054
9. Chuang, H.C., Lei, Z., Opalach, A., Norkin, A.: Analysis of AV1 coding tools. In: Andrew G. Tescher and Touradj Ebrahimi (ed.) Applications of Digital Image Processing XLV, vol. 12226, p. 122260Q. International Society for Optics and Photonics, SPIE. https://doi.org/10.1117/12.2635956
10. Daede, T., Norkin, A., Brailovskiy, I.: Video codec testing and quality measurement (2020). https://tools.ietf.org/html/draft-ietf-netvc-testing-09
11. Fonseca, J.: gprof2dot (2014). https://github.com/jrfonseca/gprof2dot
12. Free Software Foundation, Inc: gprof(1) - linux man page (2009). https://linux.die.net/man/1/gprof
13. Grellert, M., Shafique, M., Khan, M.U.K., Agostini, L., Mattos, J.C.B., Henkel, J.: An adaptive workload management scheme for HEVC encoding. In: 2013 IEEE International Conference on

Image Processing, pp. 1850–1854 (2013). https://doi.org/10.1109/ICIP.2013.6738381
14. Han, J., Li, B., Mukherjee, D., Chiang, C.H., Grange, A., Chen, C., Su, H., Parker, S., Deng, S., Joshi, U., Chen, Y., Wang, Y., Wilkins, P., Xu, Y., Bankoski, J.: A Technical Overview of AV1. Proceedings of the IEEE pp. 1–28 (2021). https://doi.org/10.1109/JPROC.2021.3058584
15. International Organization for Standardization: ISO/IEC 11172-MPEG-1 (11/1993): coding of moving pictures and associated audio for digital storage media up to about 1.5Mbit/s - part 2: video (1993). https://www.iso.org/standard/22411.html
16. International Telecommunication Union: Recommendation H.262: Generic Coding of Moving Pictures and Associated Audio Information: Video (1995). https://www.itu.int/rec/T-REC-H.262
17. International Telecommunication Union: Recommendation H.263: Video Coding for Low Bit Rate Communication (1996). https://www.itu.int/rec/T-REC-H.263/
18. International Telecommunication Union: Recommendation H.264: Advanced Video Coding for Generic Audiovisual Services (2003). https://www.itu.int/rec/T-REC-H.264/
19. International Telecommunication Union: Recommendation H.265: High Efficiency Video Coding (2013). https://www.itu.int/rec/T-REC-H.265-201911-I
20. International Telecommunication Union: Recommendation H.266: Versatile Video Coding (2020). https://www.itu.int/rec/T-REC-H.266-202008-I
21. Kim, I.K., Min, J., Lee, T., Han, W.J., Park, J.: Block Partitioning Structure in the HEVC Standard. IEEE Trans. Circ. Syst. Video Technol. **22**(12), 1697–1706 (2012). https://doi.org/10.1109/TCSVT.2012.2223011
22. Lin, W.T., Liu, Z., Mukherjee, D., Han, J., Wilkins, P., Xu, Y., Rose, K.: Efficient AV1 Video Coding Using a Multi-layer Framework. In: 2018 Data Compression Conference, pp. 365–373 (2018). https://doi.org/10.1109/DCC.2018.00045
23. Mansri, I., Doghmane, N., Kouadria, N., Harize, S., Bekhouch, A.: Comparative Evaluation of VVC, HEVC, H.264, AV1, and VP9 Encoders for Low-Delay Video Applications. In: 2020 Fourth International Conference on Multimedia Computing, Networking and Applications (MCNA), pp. 38–43 (2020). https://doi.org/10.1109/MCNA50957.2020.9264275
24. Mukherjee, D., Han, J., Bankoski, J., Bultje, R., Grange, A., Koleszar, J., Wilkins, P., Xu, Y.: A technical overview of VP9-the latest open-source video codec. SMPTE Motion Imag. J. **124**(1), 44–54 (2015). https://doi.org/10.5594/j18499
25. Nguyen, T., Marpe, D.: Future Video Coding Technologies: A Performance Evaluation of AV1, JEM, VP9, and HM. In: 2018 Picture Coding Symposium (PCS), pp. 31–35 (2018). https://doi.org/10.1109/PCS.2018.8456289
26. Oppenheim, A., Schafer, R., Stockham, T.: Nonlinear filtering of multiplied and convolved signals. IEEE Trans. Audio Electroacoustics **16**(3), 437–466 (1968). https://doi.org/10.1109/TAU.1968.1161990
27. Pakdaman, F., Adelimanesh, M.A., Gabbouj, M., Hashemi, M.R.: Complexity Analysis Of Next-Generation VVC Encoding And Decoding. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 3134–3138 (2020). https://doi.org/10.1109/ICIP40778.2020.9190983
28. Parker, S., Chen, Y., Han, J., Liu, Z., Mukherjee, D., Su, H., Wang, Y., Bankoski, J., Li, S.: On transform coding tools under development for VP10. In: A.G. Tescher (ed.) Applications of Digital Image Processing XXXIX, vol. 9971, pp. 407 – 416. International Society for Optics and Photonics, SPIE (2016). https://doi.org/10.1117/12.2239105
29. de Rivaz, P., Haughton, J.: AV1 Bitstream & Decoding Process Specification (2018). https://aomediacodec.github.io/av1-spec/
30. Siqueira, I., Correa, G., Grellert, M.: Rate-distortion and complexity comparison of hevc and vvc video encoders. In: 2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS), pp. 1–4 (2020). https://doi.org/10.1109/LASCAS45839.2020.9069036
31. Smith, S.W.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Pub. San Diego, California (1997)
32. Statista: Number of monthly logged-in YouTube viewers worldwide as of May 2019 (2021)
33. Su, H., Chen, M., Bokov, A., Mukherjee, D., Wang, Y., Chen, Y.: Machine Learning Accelerated Transform Search For AV1. In: 2019 Picture Coding Symposium (PCS), pp. 1–5 (2019). https://doi.org/10.1109/PCS48520.2019.8954514
34. WebM Project: WebM: an open web media project (2012). https://www.webmproject.org
35. Webster, A., Wolf, S.: Spatial and temporal information measures for video quality (1992). https://www.its.bldrdoc.gov/publications/details.aspx?pub=2617
36. Xu, M., Jeon, B.: User-Priority Based AV1 Coding Tool Selection. IEEE Transactions on Broadcasting **67**(3), 736–745 (2021). https://doi.org/10.1109/TBC.2021.3071013
37. Zhao, X., Liu, S.: Unified Secondary Transform for Intra Coding Beyond Av1. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 3393–3397 (2020). https://doi.org/10.1109/ICIP40778.2020.9191190