Fast VP9-to-AV1 Transcoding based on Block Partitioning Inheritance

Alex Borges, Daniel Palomino, Bruno Zatt, Marcelo Porto, Guilherme Correa
Video Technology Research Group (ViTech)
Graduate Program in Computing (PPGC), Federal University of Pelotas (UFPel), Brazil
{amborges, dpalomino, zatt, porto, gcorrea}@inf.ufpel.edu.br

***Abstract*— This paper proposes a fast VP9-to-AV1 video transcoding algorithm based on block partitioning inheritance. The proposed algorithm relies on the reuse of VP9 block partitioning during the AV1 re-encoding process. This way, the exhaustive search for the best block size option is avoided to save encoding time. The reuse of VP9 block partitioning is proposed based on a statiscal analysis that shows the relation of block parititioning sizes between VP9 and AV1. The analysis demontrates that there is a high probability of the AV1 encoding process to choose block sizes of the same size as in the VP9 encoding. Experimental results show that the proposed algorithm is able to accelerate the VP9-to-AV1 transcoding process by 28% on average at the cost of only 4% increase in the BD-Rate when compared with the complete decoding and re-encoding process.**

Keywords—AV1, VP9, transcoding, video coding

I. Introduction

In 2015, the Alliance for Open Media (AOMedia) consortium was created to develop modern royalty-free video coding formats for online applications, such as on-demand video transmission, videoconferences, and live streaming. Partially based on the VP9 [1], Daala [2] and Thor [3] codecs, AOMedia launched the AOMedia Video 1 (AV1) [4] format. Along with the specification, the *libaom* [5] reference software was released in 2018. Since then, many other fast AV1 codecs versions have been developed and released by AOMedia members, such as the Scalable Video Technology for AV1 Encoder (SVT-AV1) [6] developed by Intel and Netflix, the CISCO-AV1 [7] developed by CISCO, and the rav1e [8] developed by XIPH.

One of the main goals of AV1 is to overcome the compression efficiency achieved by VP9 and replace it as current state-of-the-art technology based on royalty-free codecs. To accomplish that, AV1 includes several new tools and features with much more efficient signal processing operations and frame partitioning structures in comparison to VP9. However, this efficiency is achieved at the cost of a considerable complexity increase in comparison to VP9. The authors in [9] and [10] show that the reference *libaom* codec requires an encoding time more than 100 times larger in comparison to VP9. Thus, time-saving strategies for AV1, especially those leading to small or no penalties in terms of compression efficiency, are currently essential to reduce this gap and enable the deployment of AV1 codecs.

VP9 owner and developer, Google, is the main company that makes use of VP9 in its video services, like the YouTube platform [11], one of the most popular free video platforms in the world. According to [12] more than 500 hours of video content around the world is published every minute on YouTube. All these videos are stored in large data centers and require a huge space in hard drives. In [9], [10], and [4] the authors demonstrate that AV1 can achieve a compression efficiency gain of 20%, 18%, and 28%, respectively, when compared to VP9 (considering the same image quality). This represents an average superiority of 22% for AV1 over VP9, an economy of 1/5 in storage resources and other costs involving video transmission services.

Video transcoding is the process that converts from one video bitstream format to the same format with different configurations (homogeneous transcoding) or to another bitstream format (heterogeneous transcoding), as presented in Fig. 1. With the advent of AV1, converting legacy content from previous formats, such as VP9, becomes an essential task for service providers that intend to benefit from its compression efficiency. However, as the computational cost required by *libaom* is too high, speeding up the encoding process is important to allow fast transcoding without decreasing compression efficiency significantly.

VP9 and AV1 share several characteristics that can be harnessed during the transcoding process. Both VP9 and AV1 follow a block-based hybrid video coding process, as they divide the frame into smaller parts called blocks for prediction, transform, and quantization. Blocks can assume different sizes and shapes, as defined by the codec specification. To achieve the best compression efficiency, the encoder needs to find the best block size to use in each region of the video. Usually, this is performed by exhaustive search over all block size possibilities, which requires a huge computational effort. In [13], a method for inheriting the best block size from H.264/AVC to H.265/HEVC during the transcoding process is proposed. In [14], the H.265/HEVC Coding Unit depth information is used to accelerate the transcoding from H.265/HEVC to AV1. Although the methods proposed in [13] and [14] present good results and are based on block partitioning inheritance, they focus on the H.264/AVC and H.265/HEVC standards, which employ a very different set of block sizes, partitioning modes and coding tree structure in comparison to VP9. Thus, they cannot be directly employed to accelerate the VP9-to-AV1 transcoding process. To the best of the authors' knowledge, there is no other work focusing on VP9-to-AV1 transcoding.

This paper proposes a fast VP9-to-AV1 transcoding process based on Block Partitioning Inheritance. The proposed solution saves time by reusing the VP9 block partitioning direction to filter out AV1 partitioning possibilities during the re-encoding process. The idea is based on a statistical analysis performed over a set of VP9 and AV1 bitstreams, which allowed identifying partition modes rarely used under certain circumstances.

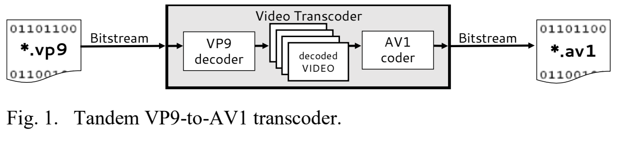

Fig. 1. Tandem VP9-to-AV1 transcoder.

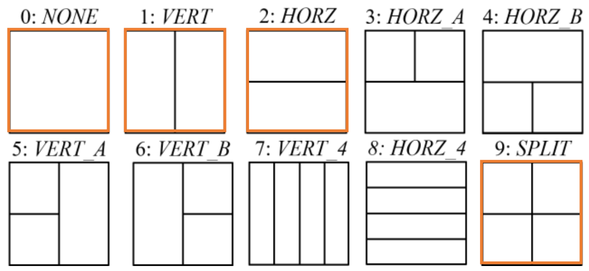

Fig. 2. Block partitioning allowed in AV1 and VP9 (highlighted).

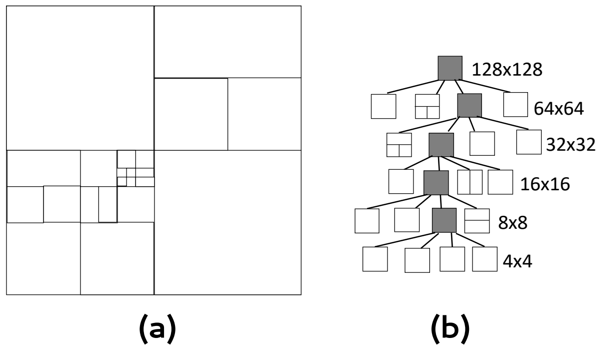

Fig. 3. An AV1 superblock partitioned into blocks: (a) block view, (b) tree view. Gray blocks represent *Split* partitioning mode.

II. VP9/AV1 Partitioning Correlation Analysis

A block can assume square or rectangular shapes organized in some configurations, called partitions type. In VP9 there are three partition types: square (named as *None*), vertical (*Vert*) and horizontal (*Horz*), as shown in Fig. 2. In AV1, blocks can assume nine partition types based on the three directions observed in VP9. Besides, both codecs also allow a *Split* partition type, which recursively subdivides the current block into four square blocks. This process follows a coding block tree structure, as shown in Fig. 3. It is likely that the same block partitioning will be used in both VP9 and AV1 codecs, since the same video region is being encoded. Considering this, we performed a correlation analysis between the partitioning chosen by each codec to use it as a basis for the proposed fast transcoding algorithm.

A. Experimental Setup

The Spatial Information and Temporal Information (SI-TI) analysis [15] was performed over all the test sequences available in [16], section "objective-2-slow", to identify those with most heterogeneous characteristics in order to enable a diverse set of videos to be used for testing. Videos sequences selected for the statistical analysis were *Blue Sky*, *BQ Highway*, *Dirt Bike*, *Guitar HDR Amazon*, *Netflix Dinner Scene*, *Netflix Food Market 2*, *Netflix Tunnel Flag*, and *Water HDR Amazon*, as available in [17]. To perform the experiments, 60 frames of each video sequence were encoded.

The reference codec software for both VP9 and AV1 was used in the experiments. For VP9, the *libvpx* [18] codec, version 1.8.2 (hash code 50d1a4), was used. For AV1, the *libaom* [5] codec, version 1.0.0 (hash code db8f27), was used. The reference software implementations were chosen because they represent the most complete versions of the encoder specifications, including all the available modes and partitioning possibilities allowed in both formats.

IETF-NETVC-T [16] is the documentation that defines the test configurations used for both video codecs. Following the document, the *High Latency CQP* configuration was used in the experiments and the Constrained Quality (CQ) parameter was set to values 20, 32, 43, and 55. All experiments are executed sequentially in the same workstation (Intel i7@2.7 GHz processor, 8 GB RAM, Ubuntu OS), in terminal mode.

The CQ parameter was set to 20 for quantization in the VP9 encoder, aiming at transcoding from the best quality available in the recommended settings. For AV1, CQ values 20, 32, 43, and 55 were used, as defined in [16]. As AV1 introduces new partitioning possibilities in both the horizontal and vertical directions, the occurrence rate of modes belonging to each direction was summed up in the analysis. Thus, modes 2, 3, 4, and 8 in Fig. 2 are considered as *Horz* modes, whereas modes 1, 5, 6, and 7 are considered as *Vert* modes.

B. Correlation Analysis

For each block in a VP9-encoded video, the same region in the AV1-encoded video was observed. For that, a label was attributed to each 4×4 area in the frame, indicating which block size and partition mode were chosen during the

TABLE I. Correlation analysis between partition types chosen by VP9 and AV1 (CQ 20).

AV1 \ VP9		64×64			32×32			16×16			8×8		
		None	***Horz***	***Vert***	***None***	***Horz***	***Vert***	***None***	***Horz***	***Vert***	***None***	***Horz***	***Vert***
128×128	***None***	39.18	25.16	10.92	4.56	1.86	1.72	1.61	0.79	1.40	1.49	0.00	0.00
	Horz	1.15	0.38	0.39	0.16	0.06	0.06	0.04	0.12	0.01	0.03	0.00	0.00
	Vert	0.07	11.62	0.00	0.03	0.02	0.01	0.19	0.01	0.03	0.01	0.00	0.00
64×64	***None***	36.93	15.36	30.23	12.54	5.02	4.92	3.48	2.36	2.97	5.58	0.00	0.00
	Horz	4.35	16.43	7.34	5.30	3.70	2.16	1.73	1.42	0.93	2.20	0.00	0.00
	Vert	6.21	5.25	17.55	5.82	1.78	3.57	1.41	0.90	1.17	2.01	0.00	0.00
32×32	***None***	7.38	13.01	18.84	31.41	19.85	21.65	15.82	12.01	10.40	11.53	0.00	0.00
	Horz	1.62	6.01	4.32	13.32	25.30	13.42	16.45	15.73	9.73	10.97	0.00	0.00
	Vert	1.65	2.75	5.48	10.62	10.66	20.63	14.32	7.96	15.67	9.97	0.00	0.00
16×16	***None***	0.83	2.17	2.65	8.52	15.77	16.33	22.78	17.48	21.09	17.19	0.00	0.00
	Horz	0.31	0.85	0.93	4.03	9.53	6.60	10.68	21.50	11.43	16.19	0.00	0.00
	Vert	0.24	0.71	0.99	2.79	4.64	7.16	8.82	8.64	18.87	13.23	0.00	0.00
8×8	***None***	0.05	0.21	0.26	0.66	1.40	1.33	2.02	9.33	4.53	6.24	0.00	0.00
	Horz	0.01	0.05	0.05	0.13	0.25	0.21	0.38	1.08	0.67	1.65	0.00	0.00
	Vert	0.01	0.05	0.06	0.09	0.14	0.21	0.28	0.46	0.86	1.18	0.00	0.00
4×4	***None***	0.00	0.00	0.01	0.02	0.04	0.04	0.07	0.19	0.22	0.52	0.00	0.00
	Horz	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Vert	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
test_av1 = None
if size_AV1 == 64x64:
   if size_VP9 == 64x64 and mode_VP9 == Horz
      and cq == 20:
         test_av1 = Horz
   if size_VP9 == 32x32 and mode_VP9 == Vert
      and cq == 43:
         test_av1 = Vert
if size_AV1 == 32x32:
   if size_VP9 == 32x32 and mode_VP9 == Horz
      and (cq == 20 or cq == 32):
         test_av1 = Horz
   if size_VP9 == 16x16 and (mode_VP9 == None
      or mode_VP9 == Horz) and cq == 20:
         test_av1 = Horz
   if size_VP9 == 16x16 and mode_VP9 == Vert
      and (cq == 20 or cq == 32):
         test_av1 = Vert
if size_AV1 == 16x16:
   if size_VP9 == 16x16 and mode_VP9 == Horz
      and (cq == 20 or cq == 32 or cq == 43):
         test_av1 = Horz
```

Fig. 4. Pseudo-algorithm to select AV1 modes for testing.

encoding process for the region. Then, for each video sequence the occurrences of each combination between block size and partitioning were accumulated and a simple percentage calculation was performed taking the VP9 block size and partitioning combination as anchor.

Table I shows the average correlation between VP9 and AV1 partitionings, considering CQ 20. VP9 partitions are presented in the columns, whereas AV1 mode directions are presented in the rows. For example, whenever a 32×32 block size and the *Horz* mode are chosen in VP9, the AV1 encoder also decided for a 32×32 block and a *Horz* mode in 25.3% of the cases. Similarly, in 19.9% of the cases, AV1 decided for a 32×32 block and the *None* mode. It is also noticeable that 4×4 blocks are very rarely chosen during the AV1 encoding process, even though they are very numerous to be tested and incur a significant computational cost.

Table I allows identifying that some partitioning candidates present a much higher probability of occurrence than others. In general, the higher probabilities are concentrated around the diagonal area of the table (highlighted), which indicates a tendency of AV1 choosing the same or similar partitions to VP9. The same occurs when other CQ values are used, with a small increase of occurrence for larger blocks in AV1. The full data obtained in the experiment are available in [17], including the analysis for all CQ values recommended.

III. Proposed Fast Transcoding Algorithm

Based on the statistical analysis presented in the previous section, the proposed algorithm follows a strategy that only considers the most likely partitions to occur for every mode observed in VP9 during decoding. Thus, the AV1 encoder performs the full encoding loop (i.e., prediction, transform, quantization, entropy coding) for only one partition type by block. For example, considering the analysis for CQ 20 (Table I), if a 64×64 block and the *Horz* mode is detected for a frame region during the VP9 decoding process, then the AV1 encoder should test only the *Horz* modes when considering a 64×64 block since this is the most probable mode class to be chosen for this block size (16.4%).

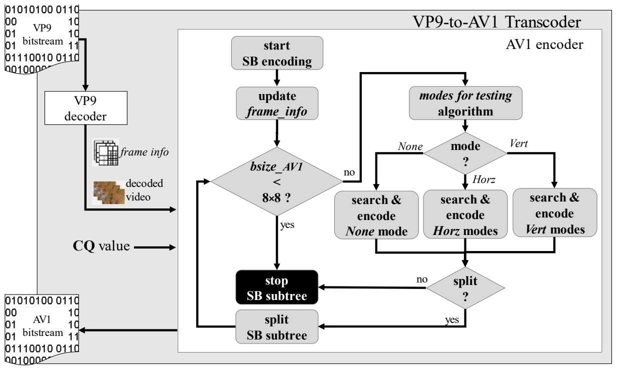

Fig. 5. Fast transcoding algorithm flow proposed.

Fig. 4 presents the pseudo-algorithm with generalized rules obtained based on the video sequences, CQ values, and modes, as explained in the analysis presented in section II. This pseudo-algorithm is based on the block size and the partitioning mode observed during the VP9 decoding process, and the CQ used. These three parameters determine the occurrence probability of each mode in AV1 re-encoding, according to the statistics gathered in the correlation analysis. In Fig. 4 *test_AV1* represents the selected test mode allowed for AV1, *size_AV1* represents the current block size and *size_VP9* and *mode_VP9* are the observed block size and partitioning mode observed during the VP9 decoding.

The proposed algorithm, summarized in the Fig. 5, is implemented in the VP9-to-AV1 transcoder and executed in three steps. First, the block sizes and partition modes observed in the VP9 bitstream decoding are exported. During the VP9 decoding, the block size, partition type, column and row positions, and frame number of the superblock is exported to a file (*frame_info*). Second, the data is adapted so that the AV1 codec can handle the values. For example, in VP9 frame positions are multiples of eight, whereas in AV1 they are multiples of four. Third, the AV1 encoder reads *frame_info* data for every superblock in the re-encoding process. After that, the pseudo-code to select modes for testing (Fig. 4) is executed before every block size candidate is encoded. If none of the allowed partition types result in the best encoding option, AV1 applies the recursive *Split* process. The *Split* process is allowed from 128×128 to 16×16 blocks only since 8×8 blocks are split in less than 0.5% of the occurrences (see 4×4 blocks in Table I).

IV. Experimental Results

The setup described in section II.A was used for the experiments presented in this section. The Bjøntegaard Delta rate (BD-rate) metric [19] is generally used to compare two video codecs. It represents the bitrate increase (positive value) or decrease (negative value) considering the same image quality. In this work, BD-rate values indicate how the fast *libaom* codec compares to the original *libaom* codec in terms of bitrate, considering the same image quality. The Peak Signal-to-Noise Ratio for Luminance (PSNR-Y) and bitrate per second are used to calculate BD-rate. Average time savings (TS) are calculated according to (1) considering the four CQ values evaluated, where $time_{fast}$ represents the run

$$TS = 100 * \left(1 - \frac{\sum_{cq}\left({time_{fast}}/{time_{original}}\right)}{4}\right) \tag{1}$$

TABLE II. COMPRESSION EFFICIENCY AND TIME SAVING RESULTS FOR THE FAST VP9-TO-AV1 TRANSCODER

Class resolution	**Video Sequence**	**BD-rate (%)**	**TS (%)**	**Ratio**
A 432x240	*BQFree*	4.7555	24.18	0.197
	BQZoom	4.4228	23.37	0.189
	Chairlift	4.0765	26.89	0.152
	Mozzoom	2.7979	26.15	0.107
B 640x360	*Rain2HDRAmazon*	2.9938	32.73	0.091
	RedKayak	1.9096	31.62	0.060
	SnowMnt	1.6583	33.61	0.049
	TacoManArrows	3.5571	18.65	0.191
C 1280x720	*Dark*	4.1424	23.02	0.180
	Johnny	6.0483	24.74	0.245
	NetflixDrinvingPOV	6.3701	31.58	0.202
	NetflixRollerCoaster	6.6917	41.73	0.160
D 1920x1080	*CrowdRun*	3.5322	34.25	0.103
	NetflixCrossWalk	4.0039	18.01	0.222
	ParkJoy	5.8649	42.67	0.137
	SeaplaneHDRAmazon	4.9808	24.53	0.203
E 1920x1080 4:4:4	*DOTA2*	3.3484	25.49	0.131
	Minecraft	4.8084	47.58	0.101
	Starcraft	2.8454	21.22	0.134
	Wikipedia	8.0104	11.26	0.711
	Average Group A	**4.0132**	**25.15**	**0.161**
	Average Group B	**2.5297**	**29.15**	**0.098**
	Average Group C	**5.8131**	**30.27**	**0.197**
	Average Group D	**4.5955**	**29.86**	**0.166**
	Average Group E	**4.7532**	**26.39**	**0.269**
	Average ALL	**4.3409**	**28.16**	**0.178**
	Standard Deviation	**1.6411**	**8.92**	**0.136**

TABLE III. BD-RATE COMPARISON BETWEEN ORIGINAL AND FAST AV1 ENCODERS OVER VP9 ENCODER AS ANCHOR

Class resolution	**Video Sequence**	**original *libaom***	**fast *libaom***	**absolute difference**
A 432x240	*BQFree*	-17.5447	-13.6765	3.8682
	BQZoom	-29.6577	-26.9454	2.7123
	Chairlift	-16.1249	-12.8527	3.2722
	Mozzoom	-20.1364	-17.8323	2.3041
B 640x360	*Rain2HDRAmazon*	-24.7490	-22,4829	2.2661
	RedKayak	-0.1282	1.7653	1.8935
	SnowMnt	-28.7661	-27.6131	1.1530
	TacoManArrows	-37.5510	-35.4298	2.1212
C 1280x720	*Dark*	-38.1851	-36.3775	1.8076
	Johnny	-35.0278	-31.1030	3.9248
	NetflixDrinvingPOV	-17.2691	-12.3370	4.9321
	NetflixRollerCoaster	-25.4314	-20.4854	4.9460
D 1920x1080	*CrowdRun*	-7.3598	-4.2105	3.1493
	NetflixCrossWalk	-19.1816	-16.0049	3.1767
	ParkJoy	-8.5279	-2.0603	6.4676
	SeaplaneHDRAmazon	-19.6745	-15.7872	3.8873
E 1920x1080 4:4:4	*DOTA2*	-17.5706	-14.9240	2.6466
	Minecraft	-17.7022	-14.0173	3.6849
	Starcraft	-17.6433	-15.4120	2.2313
	Wikipedia	-39.2217	-34.0994	5.1223
	Average Group A	**-20.8659**	**-17.8267**	**3.0392**
	Average Group B	**-22.7986**	**-20.9401**	**1.8585**
	Average Group C	**-28.9784**	**-25.0757**	**3.9026**
	Average Group D	**-13.6860**	**-9.5157**	**4.1702**
	Average Group E	**-23.0345**	**-19.6132**	**3.4213**
	Average ALL	**-21.8727**	**-18.5943**	**3.2784**
	Standard Deviation	**10.5591**	**10.8024**	**1.3405**

time of the fast *libaom* and the $time_{original}$ is the run time of the original *libaom* version.

Table II shows the experimental results for all video sequences, clustered in five classes. Besides BD-rate and TS for each video, the last column shows the ratio between BD-rate and TS. This ratio indicates the compression efficiency loss for each percent in obtained time savings. Ratios close to zero indicate a better tradeoff between compression efficiency and time savings. Detailed experiment results with data used to calculate both BD-rate and TS can be obtained in [17].

The proposed transcoding algorithm reduces on average 28.16% of the execution time, at the cost of an average BD-rate of 4.3409%. TS results present a standard deviation of 8.92%, whereas BD-rate values present a standard deviation of 1.6411%. The best-case shown in Table II corresponds to the *Snow Mmt* video, Group B. This video presents a TS of 33.61% at the cost of a BD-rate increase of 1.6583%. On the other hand, the worst case is the *Wikipedia* video, Group E, which achieves a TS of 11.26% at the cost of a BD-rate increase of 8.0104% in comparison to the original *libaom* implementation. *Wikipedia* is the only screen content video used in the experiments. As AV1 includes specific techniques to encode this type of media, the proposed algorithm which was devised for general-content sequences, is not able to take advantage of this specific case.

As previously mentioned, the current AV1 version achieves an average BD-rate decrease of 22% over VP9. This is noticeable in Table III, which shows a comparison between the original *libaom* and the proposed fast *libaom* version, taking the VP9 encoder as the baseline. For that, the BD-rate values were calculated using the same input video sequences for both VP9, original *libaom* and fast *libaom* software. When compared to VP9, the original AV1 encoder achieves an average BD-rate decrease of 21.8%, as expected. The fast

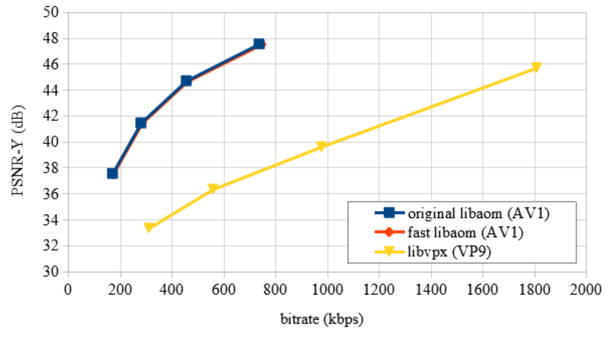

Fig. 6. Comparison between VP9, original *libaom* and proposed fast *libaom* for the *Snow Mnt* sequence.

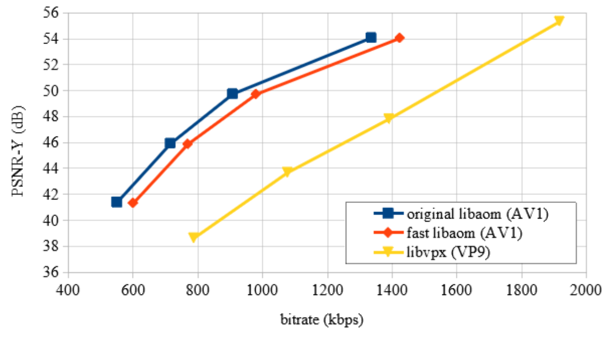

Fig. 7. Comparison between VP9, original *libaom* and proposed fast *libaom* for the *Wikipedia* sequence.

libaom version for transcoding achieves a BD-rate decrease of 18.5% in comparison to VP9, which is a slight difference given the computational cost decrease in only 3.2% of BD-Rate. These results show that even when employing the fast *libaom* version for transcoding, AV1 still excels by far in comparison to VP9 in terms of compression efficiency.

Rate-distortion efficiency results are presented in the charts of Fig. 6 and Fig. 7 for the best and worst cases presented in Table II (*Snow Mnt* and *Wikipedia*, respectively). The figures compare the compression efficiency between the video sequences encoded with VP9, the original *libaom*, and the proposed fast *libaom* encoder. Notice that in both the best and worst-case scenarios, the proposed encoder presents compression efficiency results very close to the original *libaom*. Also, it is clearly noticeable that AV1 presents a much better compression efficiency than VP9, either the original video transcoder or the proposed fast video transcoder.

V. Conclusion

Launched recently by the *libaom* consortium, AV1 will gradually become a widely adopted encoding format for online video streaming. Big companies with a large amount of video content will eventually migrate from previous formats to AV1. Thus, transcoding from VP9 to AV1 is in the interest of several companies that employ royalty-free formats. However, fast transcoding approaches are required due to the high computational cost required by the AV1 encoding process. This paper presents an algorithm that allows reusing block partitioning decisions provided by the VP9 decoding process to accelerate the VP9-to-AV1 transcoding. The strategy is based on a statistical analysis that allowed identifying the most probable partitions in AV1 when certain modes and blocks sizes are observed during VP9 decoding. By doing so, the *libaom* execution time is reduced by 28.16% on average, at the cost of a BD-rate increase of 4.3409%. Also, when compared to the VP9 compression efficiency, the fast AOM encoder still yields a BD-rate gain of 18.6%, which is close to the obtained by the original AOM (21.8%).

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul, Brasil (FAPERGS).

References

[1] D. Mukherjee et al., "A Technical Overview of VP9—The Latest Open-Source Video Codec," in SMPTE Motion Imaging Journal, vol. 124, no. 1, pp. 44-54, Jan. 2015. doi: 10.5594/j18499

[2] "Daala video compression," Xiph.org, 2013. [Online]. Available: https://xiph.org/daala/. [Accessed: 20- Jan- 2020].

[3] G. Bjøntegaard, T. Davies, A. Fuldseth and S. Midtskogen, "The Thor Video Codec," in 2016 Data Compression Conference (DCC), Snowbird, UT, pp. 476-485, 2016. doi: 10.1109/DCC.2016.74

[4] Y. Chen et al., "An Overview of Core Coding Tools in the AV1 Video Codec," in 2018 Picture Coding Symposium PCS 2018 - Proceedings, pp. 41–45, 2018. doi: 10.1109/PCS.2018.8456249

[5] "AV1 Codec Library," Alliance for Open Media, 2015. [Online]. Available: https://aomedia.googlesource.com/aom. [Accessed: 20- Jan- 2020].

[6] "Scalable Video Technology for AV1 Encoder," Intel, 2015. [Online]. Available: https://github.com/OpenVisualCloud/SVT-AV1. [Accessed: 20- Jan- 2020].

[7] T. Davies, "Cisco Leap Frogs H.264 Video Collaboration with Real-Time AV1 Codec," CISCO, 2019. [Online]. Available: https://blogs.cisco.com/collaboration/cisco-leap-frogs-h-264-video-collaboration-with-real-time-av1-codec. [Accessed: 20- Jan- 2020].

[8] "rav1e," Xiph.org 2015. [Online]. Available: https://github.com/xiph/rav1e. [Accessed: 20- Jan- 2020].

[9] D. Grois, T. Nguyen, D. Marpe, "Performance comparison of AV1, JEM, VP9, and HEVC encoders," in SPIE 10396, Applications of Digital Image Processing XL, 103960L. Proceedings, 2018. doi: 10.1117/12.2283428

[10] L. Guo, J. De Cock and A. Aaron, "Compression Performance Comparison of x264, x265, libvpx and aomenc for On-Demand Adaptive Streaming Applications," in 2018 Picture Coding Symposium (PCS), San Francisco, CA, 2018, pp. 26-30. doi: 10.1109/PCS.2018.8456302

[11] M. Srinivasan, "VP9 Encoder and Decoders for Next Generation Online Video Platforms and Services," in SMPTE 2016 Annual Technical Conference and Exhibition, Los Angeles, CA, 2016, pp. 1-14. doi: 10.5594/M001734

[12] J. Clement, "Hours of video uploaded to YouTube every minute," Statista, 2019. [Online]. Available: https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/. [Accessed: 20- Jan- 2020].

[13] A. J. Díaz-Honrubia, J. L. Martínez, P. Cuenca, J. A. Gamez and J. M. Puerta, "Adaptive Fast Quadtree Level Decision Algorithm for H.264 to HEVC Video Transcoding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 154-168, Jan. 2016. doi: 10.1109/TCSVT.2015.2473299

[14] A. Borges, B. Zatt, M. Porto and G. Correa, "Fast Hevc-to-Av1 Transcoding Based On Coding Unit Depth Inheritance," in 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 3571-3575. doi: 10.1109/ICIP.2019.8803482

[15] International Telecommunication Union, "Recommendation ITU-T P.910: Subjective Video Quality Assessment Methods for Multi-Media Applications," ITU-T, 2008.

[16] T. Daede, A. Norkin, I. Brailovskiy, "Video Codec Testing and Quality Measurement draft-ietf-netvc-testing-08," Internet Engineering Task Force, 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-netvc-testing-08. [Accessed: 20- Jan- 2020].

[17] A. Borges, "EUSIPCO 2020," Github, 2020. [Online]. Available: https://github.com/amborges/EUSIPCO_2020. [Accessed: 28/02/2020].

[18] "WebM VP8/VP9 Codec SDK," Chromium, 2019. [Online]. Available: https://chromium.googlesource.com/webm/libvpx/. [Accessed: 20- Jan- 2019].

[19] G. Bjøntegaard, "VCEG-M33: Calculation of Average PSNR Differences between RD curves," Video Coding Experts Group (VCEG) 2001.