Fast H.264/H.265 to AV1 Stream Transcoding Using a Moving Object Tracker

Ivan Liapin
Department of Television, Faculty of Radio Engineering and Telecommunications
Saint-Petersburg Electrotechnical University
Saint-Petersburg, Russia
ivanliapin@gmail.com

***Abstract*—With the AV1 codec soon coming to the global video streaming market it is important to find a way to efficiently transcode existing streams encoded with previous generation codecs, such as H.264/AVC and H.265/HEVC to AV1. Because of the massively increased computational complexity of AV1 direct re-encoding is not a viable option as of today. This paper proposes a method of fast H.264/H.265 to AV1 stream transcoding based on reusing motion data from H.264/H.265 to track objects in input compressed domain and remove the need to do most of the complex inter-prediction in AV1 encoder. It is shown that the reduction in encoder run time can be significant enough to provide transcoding speeds on par with reference encoders of the previous codec generation.**

Keywords—AV1, H.264, H.265, codecs, transcoding, object tracking

I. Introduction

Compared to video coding standards of the previous generation, such as H.264/AVC and H.265/HEVC, the upcoming AV1 video codec provides significant coding gains (aimed at 40-50% gain over VP9/HEVC), which is in line with the current market tendency of growing image resolution, framerate and bit depth. However, these compression gains come at the cost of encoder and decoder complexity, which leads to an increase in reference encoder run time by a factor of more than 25 compared to HEVC, and more than 100 compared to VP9 [1].

It is worth mentioning that at the time of writing this paper, the AV1 development is still in progress and the bitstream has not been frozen yet. This means that there is no hardware encoding/decoding support and the best way of assessing AV1 performance is using its reference encoding and decoding software. However, most of the development is already done and large-scale changes in bitstream and encoder/decoder structure are unlikely, so current reference encoder performance evaluation results will hold after bitstream freeze.

Because the royalty-free AV1 codec is supported by many software, hardware and streaming industry leaders (members of Alliance for Open Media), more efficient encoder/decoder implementations are destined to come at some point in future, followed by rapid AV1 adoption. In the meantime, the global market will have to find a way to move infrastructures that use previous-generation video codecs to support AV1. One specific task in this process is transcoding existing video streams encoded with H.264/H.265 into AV1. Because of the massively increased AV1 encoder complexity plain H.264 stream decoding followed by encoding into AV1 is not a considerable option, especially if real-time transcoding is required (for example, to store live video feeds).

The most time-consuming task in contemporary video encoding is inter-prediction. During inter-prediction the encoder searches through available reference frames to find a block that can be shifted to the position of the currently encoded block so their pixel difference and offset (motion vector) can be encoded, rather than pixels of the current block directly (motion compensation). Because modern encoders have a complex macroblock structure, allow sub-pixel motion vector precision and employ other more elaborate coding tools, encoder has to sift through a great number of possible decisions during inter-prediction to find the most optimal match, which, depending on encoder settings, can take up to around 90% of encoder run time. Any measures taken to decrease the time spent doing inter-prediction can greatly increase encoder performance.

In this paper, a method of fast H.264/H.265 to AV1 stream transcoding is proposed. A moving object tracker based on H.264 motion vector extraction is employed to detect areas that correspond to moving objects. Motion vectors from those areas are then reused in the AV1 encoder to decrease computational complexity, and areas that contain no motion are coded with no offset (skipped). Such process is best used to transcode static-camera sequences, where background and foreground objects are clearly separated. It is shown that reusing motion vectors can greatly increase transcoding performance, to the point of AV1 reference encoder run time being on par with the AVC reference encoder (JM 19.0).

II. Inter-prediction Techniques in AV1

AV1 builds on coding techniques used by mainstream video codecs, expands them and adds its own tools to increase coding efficiency [2]. To build a transcoder from H.264 to AV1 an understanding of differences in inter-prediction processes of both codecs is required. This chapter provides an overview of new inter-prediction related coding tools introduced in AV1 and their effect on reusing motion information from H.264/H.265.

A. Block Partitioning Decisions and Reference Frames

H.264 uses 16x16 blocks for both inter- and intra-prediction. For inter-prediction a 16x16 block can be split into two 16x8 or 8x16 blocks, or split into four 8x8 blocks, and each of those can be split into 4x8, 8x4 or 4x4 partitions. Thus, the

978-1-5386-4615-1/18/$31.00 ©2018 IEEE

smallest available unit for motion compensation is a 4x4 block. In H.265 a Coding Tree Block (sizes from 64x64 to 16x16) can be split into Prediction Blocks in eight different ways, but the smallest available block also a 4x4. AV1 introduces Recursive Block Partitioning, which simplifies block partitioning structure: any block (starting with a 128x128 superblock) can be split into two blocks vertically or horizontally (including 1:2,2:1 and 1:4,4:1 ratios, similar to HEVC), or split into four smaller blocks, and each of these four can be split in a similar manner down to a 4x4 block.

Because all block partition techniques used in AV1 are also included in H.264 and H.265, partitioning decisions encoded in input streams can be reused "as-is" during transcoding. This will, however, mean that re-encoded stream won't be able to benefit from larger prediction block sizes supported by AV1, which will result in some loss of compression efficiency on high resolution content.

AV1 also supports up to 7 unique reference frames for one frame. This is the same number as the maximum allowed reference unique picture count for both H.264 and H.265, so reference frame decisions can also be reused "as-is", without losing compression efficiency or visual fidelity.

B. Compound Block Modes

AV1 introduces special predictor blending scheme for inter-predicted blocks: Compound Modes. In H.264/H.265 it was possible to define the current inter-predicted block as a weighted sum of two other inter-predicted blocks, with weights either being signaled explicitly or implicitly calculated based on the distance between reference frames. It is also possible to apply weighting and offset to a single prediction. This idea is further expanded in AV1: more blending modes are allowed, such as blending using masks form a wedge-codebook. This method blends two predictions into a target block, which, as result, has a smooth spatial blend from one prediction to another. It is also possible to blend inter-coded and intra-coded blocks.

The transcoder should use a Compound Mode COMPOUND_DISTANCE (select weights based on temporal distance between frames) if the weights of bi-predicted frames are signaled implicitly, and if the weights are specified, the transcoder must make a choice between COMPOUND_WEDGE (blend spatially using an oblique wedge selected from the codebook), COMPOUND_DIFFWTD (pixel weight depends on difference between prediction pixels) and COMPOUND_AVERAGE (average both predictions) based on their rate-distortion metrics. This needs to be done because AV1 does not support bi-prediction with explicit coefficients, and thus, has no way to use them even if they are provided.

C. Global Motion and Warped Motion

The Global Motion and Warped Motion tools in AV1 aim to reduce MV redundancy arising from camera motion. The detection of Global Motion is performed using a feature-matching algorithm, motion parameters can then be packed with each frame, and each macroblock can be signaled to use this information to reconstruct its motion vectors. Warped Motion examines local inter-predicted blocks to find a common motion pattern (such as warped motion from forward/backwards camera movement) and encodes it using two 2x2 shear matrices. Macroblocks then also have the option to use this information for MV reconstruction.

Because these coding tools are essentially an alternative process of retrieving inter-predicted samples (the main process being motion estimation and compensation), their usage does not conflict with the concept of reusing motion data on a block-by-block basis. However, this means that the encoder must choose either of these methods based on RDO (Rate-Distortion Optimization) considerations, thus doubling the number of possible following decisions to consider. It is also worth mentioning that our transcoder aims primarily on encoding sequences shot with a static camera, so neither Global Motion nor Warped Motion will yield any significant compression gain.

D. Overlapped Block Motion Compensation

Block-based motion compensation approach, just like any other block-based processing method, has the drawback of potential discontinuities arising at block edges. To enhance prediction quality near block edges AV1 allows the usage of OBMC, which uses neighboring inter blocks to refine the prediction of the current block.

Because OBMC is applied after the initial inter-prediction for the current block and its neighbors has been made, it does not conflict with the method of reusing motion data.

III. Object Tracking in H.264 Compressed-domain

Our transcoder must only process motion information from pictures areas that contain moving objects. This can either be done by building a motion tracker in spatial domain (using fully decoded frames to construct motion flow and then process the acquired dense motion field) or by building the tracker in compressed H.264/H.265 domain that uses motion vectors generated during the decoding process and processes the acquired sparse motion field. While the former approach has the benefit of higher motion field quality (and, as consequence, better tracking performance), building a motion field from decoded frames is a computationally demanding task that will hinder overall transcoding performance. Using a compressed-domain tracker is computationally cheap because the motion field acquisition is done during the decoding process. However, an encoder does not generate a true motion flow of a provided image sequence, but instead picks block offsets that provide most compression gain. This results in a noisy vector field, and additional processing is required to ensure satisfactory tracking performance.

The topic of compressed-domain object tracking in video streams is well studied, and many solutions to restore motion field quality exist. For the purposes of this work, a sample compressed-domain object tracker has been built and tested.

First, the tracker acquires raw motion vectors from a H.264 stream. To simplify this procedure a set of video stream decoding/processing libraries provided by FFMpeg was used. This raw partitioned motion vector field is uniformized by assigning motion vectors to the smallest possible block size in

a macroblock, thus splitting all partitions larger than 4x4 into multiple 4x4 blocks.

At this point we have a motion vector field that suffers from encoding noise. A multi-step MV field filtering is then initiated. First, the MV field is filtered in temporal domain against MV fields belonging to the previous and following frames, according to [3]. This algorithm estimates motion vector reliability by back-projecting MVs from the following frame and forward-projecting MV field from the previous frame. It then computes two similarity metrics and estimates reliability for all MVs in the current frame. This process outputs a mask of areas that contain motion and have high motion vector reliability and will be processed further.

The next step is morphological processing to further filter areas with small MV fluctuations. The mask acquired after temporal filtration is morphologically closed with a 3x3 "cross" structure element. 4-connected areas (blobs) are then detected, and for each area an average motion direction is calculated. At this point we have a mask of areas around moving objects for the current frame.

The next step is to track the acquired areas. Tracking between frames requires a buffer for storing past (or future) area masks and MV information, so at this point we introduce a buffer which stores 4 area masks (and 4 MV fields) for frames that will follow the current frame. After this buffer is filled we can begin tracking. For each area in the current frame we look for a match in the following frame. Areas are considered matching if the average motion direction of the area in present frame points to the centroid of another area in the next frame within a small search area (32px) around it. This area matching method works well for areas within the frame, but when an area leaves the frame its centroid may shift unpredictably. To compensate for this, we modify the algorithm so that if an area is within some distance of frame border, we project bounding box from the area candidate in previous frame along its motion direction, and look whether or not the centroid of this area is within this projected bounding box. This area matching process is performed each time a new area mask is added to the buffer.

After all areas in the current frame are matched we employ a technique proposed in [4]: if an area represents a moving object, it is likely to have continuous presence during a certain frame interval. Therefore, for each area we check how many times it appears in the following four frames. If it appears less than two times, it is discarded, otherwise, its position and area is interpolated for frames where this area is missing.

This multi-step filtering process yields a mask for the motion field in the current frame that segments moving objects from background encoder noise. Motion data in areas marked by this mask can be forwarded to the AV1 encoder, and unmarked areas can be skipped.

IV. Transcoder Structure

The proposed transcoder structure is shown in Fig. 1. For the purposes of this paper this structure does not include blocks related to intra-prediction, as they remain unchanged by the method of reusing motion data. The main difference from a standard spatial-domain transcoder is that the encoder motion estimation block is now supplied with a refined motion vector field from an object tracker. This means that it can now skip a massive portion of inter-prediction process and associated RDO routines, and thus gain a significant encoding time reduction.

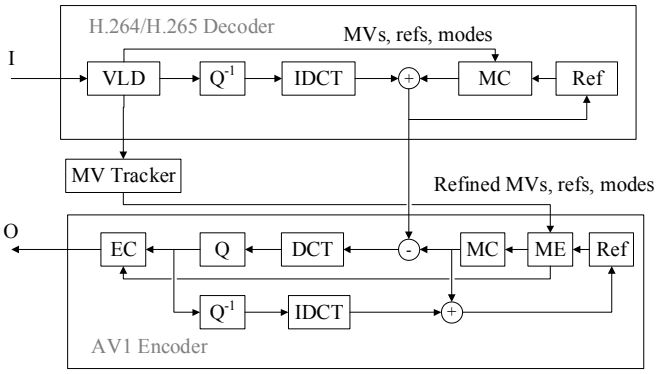

Fig. 1. Proposed transcoder structure

After the H.264/H.265 decoder has decoded motion information, it is fed to the tracker to generate a refined motion field, where only areas associated with moving objects have motion vectors. This motion vector field, along with reference frames and block partition choices is input into the inter-prediction routine of AV1. The encoder then processes this motion data, making adjustments to compound prediction modes as noted in Chapter 2. It then feeds motion data and prediction modes into AV1's entropy coder, completing inter-prediction for this sequence segment.

V. Performance Evaluation

In this chapter, performance of the proposed compressed-domain object tracker is evaluated. For the proposed transcoding scheme, reduction in AV1 encoder run time is estimated based on the assumption that the amount of calls to inter-prediction routines is minimized. All performance testing was done on an i7 4700MQ CPU running at 3.4 GHz on a Debian GNU Linux distribution.

A. Object Tracker Performance

The compressed-domain moving object tracker has been tested on 1280x720@30fps and 1920x1080@30fps video sequences encoded with H.264 (High@4.2). Fig. 2 presents a frame from a testing sequence, a grayscale image with pixel intensities proportional to MV magnitudes before motion field processing and an image containing all detected objects.

We can observe that the proposed simple compressed-domain moving object tracker successfully de-noises the initial motion vector field and is capable of separating areas around moving objects from a static background. The tracker runs at 75.8 FPS when processing a 720p H.264 video stream and at 34.1 FPS when processing a 1080p stream. Considering that the tracker runs on one thread, and that no specific optimization measures were taken, this performance can be considered satisfactory for usage during transcoding on a multi-core CPU.

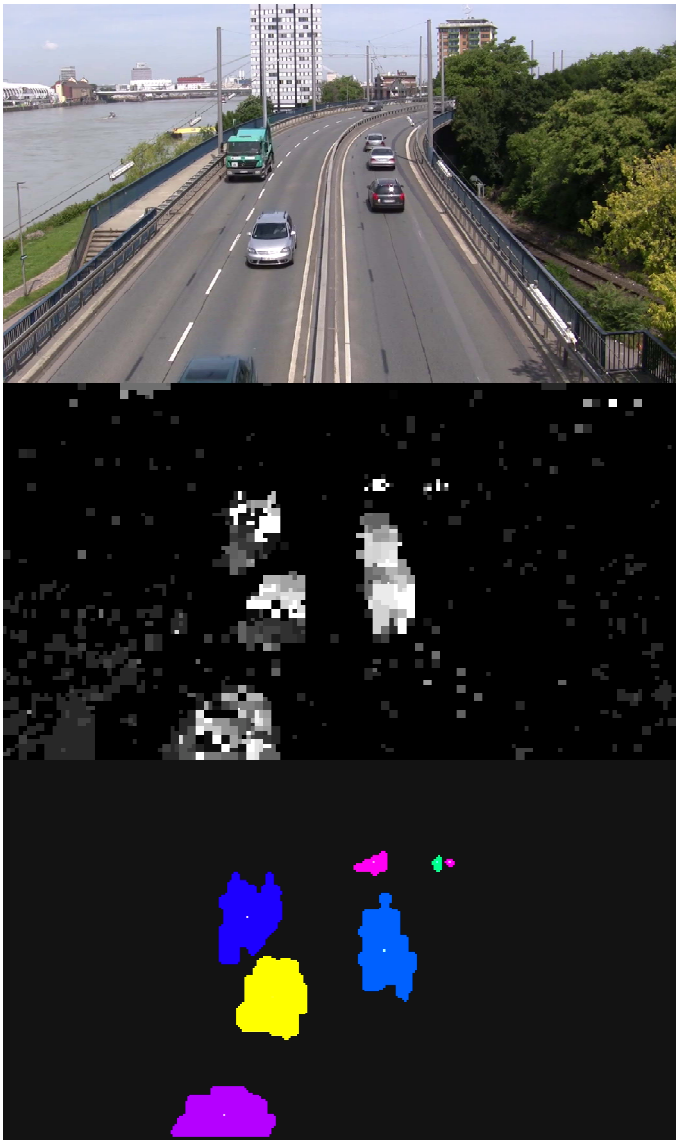

Fig. 2. A frame from a testing sequence (top), representation of motion vector magnitude for this frame (middle) and segmented objects for this frame.

B. Transcoder Performance Estimation

To estimate the reduction in encoder run time for the proposed transcoding scheme, a reference AV1 encoder was used. The encoder was profiled to find calls to routines that took the most time during inter-prediction. Calls to those routines were then bypassed to see how much encoding run time can be possibly saved in an ideal scenario where all of motion data from H.264/H.265 could be reused in AV1 and no additional inter-prediction needed to be made (or, if the image contained no moving objects, and thus all blocks had to be skipped). We can then consider that in a real scenario the encoder will have to select between different compound block modes as well as pick from numerous available MV coding methods. For a static-camera frame sequence (such as a surveillance camera feed, like the one in Figure) we can expect inter-prediction to take up to 40% of the time it would take to do inter-prediction normally. It should be noted that this figure is a very rough estimate based on amount of area taken by moving objects in used testing sequences (10-15% of total area), the fact that moving areas are likely split into smaller partitions (thus increasing total block count) and assuming that the encoder still performs some RDO for MV prediction.

Encoding times for first 30 frames of sequence *akiyo* (352x288) for AV1 reference encoder (aomenc, commit hash: *3db753071d615f0c75d54381a0f9b564a8e288d3,* settings: *--limit=10 --cpu-used=8 -p 2*) both with full inter-prediction and with bypassed inter-prediction routines, H.264 reference encoding software JM 19.0 (using default configuration file *encoder.cfg*) and H.265 reference encoding software HM 16.18 (using default configuration file *encoder_lowdelay.cfg*) are presented in Table 1.

TABLE I. ENCODER PERFORMANCE EVALUATION RESULTS

Performance Measurement	Tested Encoders			
	AV1	***AV1 (no inter pred.)***	***H.264***	***H.265***
Encoding time, s	33.97	6.98	12.35	28.78

We can observe that inter-prediction occupied around 80% (27 seconds) of total AV1 encoder run time. Assuming the estimated 40% run time increase in case of a real scenario where motion data from an H.264/H.265 stream is used during inter-prediction, we can estimate that the run time reduction from reusing motion data can result in encoder run time being around 3-4 times less than run time with a direct stream re-encode. This estimate is consistent with results obtained in [5], where the concept of reusing and refining motion data during MPEG-2 to H.264 transcoding was studied.

Past studies have also shown that reusing motion data during transcoding has little impact on objective quality metrics of the transcoded video [6]. However, in our case a greater visual quality decrease can be expected, because instead of reusing all motion data we only reuse it for areas where moving objects were found. This can backfire if a moving object has not been detected by the object tracker (for example, if it was moving too slow and was filtered out during initial temporal filtering). Actual loss in objective metrics and subjective quality is determined by an exact implementation of both object tracker and modified inter-prediction routines of AV1 encoder, as well as characteristics of the processed frame sequence.

VI. CONCLUSIONS

To facilitate rapid conversion of the current H.264/H.265-dominated video coding market to the incoming AV1 codec it is important to be able to transcode existing video streams into the new generation codec. As of today, computational complexity of AV1 prevents effective usage of direct transcoding schemes. A method of reducing computational complexity for H.264/H.265 to AV1 transcoding process based on selective re-usage of decoded motion data using a compressed-domain moving object tracker is proposed. Differences in inter-coding schemes of AV1 and H.264/H.265 and their effect on re-using motion data is discussed. Performance of the proposed moving object tracker is

evaluated, and it is concluded to be suitable for usage in the proposed transcoding scheme. Transcoding time reduction is roughly estimated, and it is found that the proposed transcoding scheme can use up to 3-4 times less time than a direct re-encode. Future work includes improvements to the motion tracker to be able to detect smaller objects with less encoder noise and building a fully functional H.264/H.265 to AV1 transcoder using the proposed scheme.

REFERENCES

[1] D. Grois, T. Nguyen and D. Marpe, "Performance Comparison of AV1, JEM, VP9, and HEVC Encoders," *Applications of Digital Image Processing XL,* 2017.

[2] "AV1 Bitstream & Decoding Process," 2018. [Online]. Available: https://aomediacodec.github.io/av1-spec/av1-spec.pdf. [Accessed 27 04 2018].

[3] S. De Bruyne, C. Poppe, S. Verstockt, P. Lambert and R. Van de Walle, "Estimating Motion Reliability to Improve Moving Object Detection," in *ICME 2009*, 2009.

[4] M. Syah Houari Sabirin and M. Kim, "Object Tracking and Indexing in H.264/AVC Bitstream Domains," InTech, 2011

[5] G. Fernandez-Escribano, H. Kalva, P. Cuenca, and L. Orozco-Barbosa, “Reducing Motion Estimation Complexity in MPEG-2 To H.264 Transcoding,” ICME 2007, pp. 440–443, 2007.

[6] Y. Wonsang, "Spatial-Domain Transcoder with Reused Motion Vectors in H.264/AVC," KAIST ICC, 2006.