EFFICIENT TRANSCODING FOR SPATIALLY MISALIGNED COMPOSITIONS FOR HEVC

Johan De Praeter, Jan De Cock, Glenn Van Wallendael,
Sebastiaan Van Leuven, Peter Lambert, and Rik Van de Walle

Ghent University - iMinds - Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium

ABSTRACT

The visualization of (ultra) high-resolution compositions created from multiple input bitstreams requires several decoders at the receiving device. Therefore, not all devices can properly display such compositions. To address this problem, the input streams are decoded, merged into a single video, and re-encoded by a transcoder in the network. However, this approach requires a computationally complex re-encoding step. To reduce this complexity, information from the input bitstreams can be reused during transcoding. In HEVC, simply reusing the original encoding information is not compression efficient if the inserted content is not aligned with the grid of coded blocks. In this paper, we applied feature selection based on a decision tree, which was used in a fast HEVC transcoding model for misaligned content. The performance varies depending on the shift and average transform size in the original sequence, resulting in complexity reductions of up to 76%.

Index Terms— HEVC, picture composition, transcoding

1. INTRODUCTION

In many industries such as video surveillance, it is necessary to present data from different encoded video sources on a single display. This means that all video sources must be decoded by the receiving device. However, depending on the device, the number of available decoders is limited, which makes it harder to decode all input streams in parallel. Moreover, if the video sequences have to be overlaid on top of each other, only parts of some sequences are needed by the receiver. Therefore, to save bandwidth and to allow devices with a very limited amount of decoders to properly display a composition of video sequences, the encoded bitstreams can be merged into one bitstream in the network. To accomplish this, the input bitstreams are transcoded by re-encoding a composition of the decoded bitstreams. Since the multiple input streams are combined into one composition, the client devices require only a single decoder.

Simply re-encoding the composition is a computationally complex operation. Moreover, to improve compression efficiency, bitstreams are encoded with High Efficiency Video Coding (HEVC), which offers a bit rate reduction of 50% for the same perceptual quality compared to its predecessor H.264/AVC [1]. However, HEVC has a higher computational complexity compared to older video compression standards, which makes it crucial to reduce the transcoding complexity.

A common approach to achieve complexity reduction is by reusing information from the input bitstreams to simplify the re-encoding step [2–4]. To exploit correlation between input and output bitstreams, machine learning techniques have often proved to be a useful tool [5–9]. Previous work on transcoding compositions mostly focused on the H.264/AVC compression standard. Solutions for video insertion included a partial re-encoding transcoder (PRET) scheme [10], guided encoding [11], and techniques based on Flexible Macroblock Ordering and Variable Length (De)coding [12]. For HEVC, previous work has determined that encoding decisions can easily be reused if the inserted content is properly aligned with the grid of Coding Tree Units (CTUs) [13]. However, if the inserted content is no longer aligned with the CTU-grid, efficiently reusing information such as the Coding Unit (CU) structure is not trivial.

In this paper, we focus on the misalignment with the CTU-grid of the inserted content. We propose a fast pixel-domain technique for re-encoding the misaligned sequence based on features selected with a decision tree. To prevent other sequences in the composition from biasing the results, we decode a single input bitstream and spatially shift the video by padding the top and left edge of the picture. The complexity reduction of the re-encoding is achieved by exploiting the correlation between the information in the original encoded bitstream and the CU structure of the shifted bitstream. The technique determines both a stopping and splitting criterion for evaluating CUs in the encoder.

The rest of the paper starts with a brief introduction to HEVC. In Section 3 we then present the analysis which leads to the implementation of our model. This model is evaluated in Section 4. Finally, we give our conclusion in Section 5.

The activities described in this paper were funded by Ghent University, iMinds, the Agency for Innovation by Science & Technology (IWT), the Fund for Scientific Research (FWO Flanders), and the European Union, and were carried out using the Stevin Supercomputer Infrastructure at Ghent University.

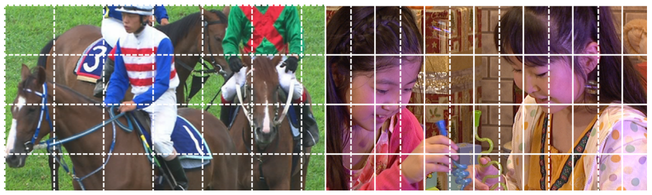


Fig. 1. The CTU-grid of the sequence on the right (full lines) is misaligned with the grid of the composition (dashed lines).

2. HIGH EFFICIENCY VIDEO CODING

HEVC is a new video compression standard offering better compression efficiency compared to its predecessors [1]. To achieve this, HEVC uses a more flexible scheme for dividing the picture into blocks [14]. The picture is first divided into CTUs of 64×64 pixels, which are recursively split into smaller CUs according to a quadtree structure, with the smallest possible block being 8×8 pixels. Each CU has an associated prediction mode (intra or inter) and is split into one of the eight Prediction Unit (PU) sizes. Depending on the mode of the CU, each PU contains either intra- or inter-prediction information. For residual coding, each CU is also recursively split into Transform Units (TUs) according to a quadtree structure, with the smallest TU size being 4×4 pixels. For each CTU, the HEVC-encoder determines the block structure with the optimal compromise between bitrate and distortion, which results in the increased complexity of HEVC compared to its predecessors.

In the rest of the paper, the depth of a block refers to the number of times a CTU has been split. Therefore, for CUs and TUs, depth 0 refers to a block of 64×64 pixels, depth 1 refers to 32×32 pixels, up to depth 4 for a 4×4 block.

3. ANALYSIS AND MODEL IMPLEMENTATION

To combine several input video sequences into a composition, two operations can be used. Video sequences can either be placed next to one another, or overlaid on top of each other. The combination of these two operations results in an arbitrary composition. However, in both cases the video sequence that was inserted into the composition might be misaligned with the CTU-grid (Fig. 1). Therefore, it is not efficient to simply reuse the CU structure of the input sequences.

To analyse the effect of misalignment on the CU structure, an input bitstream is decoded, shifted, padded at the top and left edge of the picture, and re-encoded with the same quantisation parameter as the original bitstream. This means that both the CU structure of the re-encoded bitstream as well as the encoding information of the input bitstream is available. Machine learning techniques such as decision trees can then analyse which input results in a certain CU structure [15]. To achieve this, the image from the re-encoded bitstream is divided into blocks of 64×64, 32×32 or 16×16 pixels, once for each block size. Based on the output CU structure, it is possible to determine if such a block is split. This split flag serves as the output for the decision tree algorithm J48. To determine the split flag, the decision tree uses features from the co-located block in the input bitstream.

The decision tree was trained on the first fifty frames of *BasketballDrillText*, *BlowingBubbles*, *BQSquare*, *Cactus*, *CrowdRun*, *FourPeople*, *Johnny* and *RaceHorsesC*. These sequences were encoded with the HEVC reference software (HM12) [16] using a GOP-structure of IPPP and QP-values of 22, 27, 32 and 37. They were re-encoded with a shift of either 8, 16 or 32 pixels in both x- and y-direction.

The decision tree algorithm was initially supplied with twenty-one features. The first two features are the shift of the output bitstream and the considered block size. The third feature is the intra fraction of the co-located block, which indicates the percentage of pixels that were intra-coded in the input bitstream. The other eighteen features consist of the mean, variance, maximum, minimum, and dominant CU, PU, and TU depth, as well as the variance of the motion vectors, and both variance and the mean of the variance of the residual.

Since the decision tree algorithm considers different kinds of block sizes in the output bitstream, it is possible that the co-located block contains parts of several CUs in the input bitstream. Therefore, the features based on co-located blocks are all weighted. For instance, the mean of the CU depth is calculated by dividing the original input stream into blocks of 8×8 pixels and assigning a depth to each block based on the CU structure. When examining the co-located block of a block in the output bitstream, the resulting mean is the mean of the depths of the 8×8 blocks within the co-located block.

Based on the decision tree, the most important features were selected from the initial set: the shift, block size, intra fraction, and the mean CU and TU depth. By considering only these features, the following observations yielded a model to predict the split flag for each block in the output bitstream.

1. If the shift of the video sequence is a multiple of the block size, the CU structure of the input bitstream can be copied for those block sizes with a confidence of more than 85%. This indicates that a shift of N pixels causes all CUs of $N \times N$ and smaller to remain aligned with an $N \times N$ grid.

   This rule is implemented by retaining the CU structure for block sizes that are equal to or smaller than the shift.

2. If the co-located block contains both intra- and inter-coded pixels, the CU is often split. This behaviour is likely caused due to the differences in coding mode within the co-located block making it harder to find a good prediction mode for the CU in the output bitstream. This observation has a confidence of more than 93% for a shift of 32 pixels with all block sizes and for a shift of 16 pixels with a block size of 16 pixels.

   In the implementation of the model, blocks that satisfy the above requirements are always forced to split.

3. Besides the previous two observations, the mean TU depth (mTU) in the co-located block also influences the splitting behaviour. A higher depth (a smaller transform size) leads to a higher chance to split, whereas a lower depth (bigger size) means that the block might not be split. If mTU is greater than a certain value α_b for block size b, the block should be split. On the other hand, if mTU is less than a value β_b, with $\beta_b \leq \alpha_b$, the block should not be split. This rule has a high confidence for output block sizes of 64 and 32 for all three shifts. The rule is not valid for block sizes of 16.

   This rule was implemented by forcing the encoder to split a CU if mTU is greater than α_b, to stop splitting if mTU is less than β_b, and to test both options if mTU lies between both values. Moreover, α_b and β_b were replaced as follows, with d_b the depth associated with block size b (Section 2) and $b \in \{64, 32\}$:

$$\begin{cases} \alpha_b = \gamma_b + t_{split} \\ \beta_b = \gamma_b - t_{stop} \end{cases} \qquad \text{with } \gamma_b = (d_b + 1)$$

   The value γ_b was defined such that $\gamma_b \in [\beta_b, \alpha_b]$. Thresholds t_{split} and t_{stop}, both $\in [0, 1]$, try to account for a possible variability of α_b and β_b depending on the video sequence. Consequently, the splitting behaviour depends on these two variable thresholds.

4. RESULTS

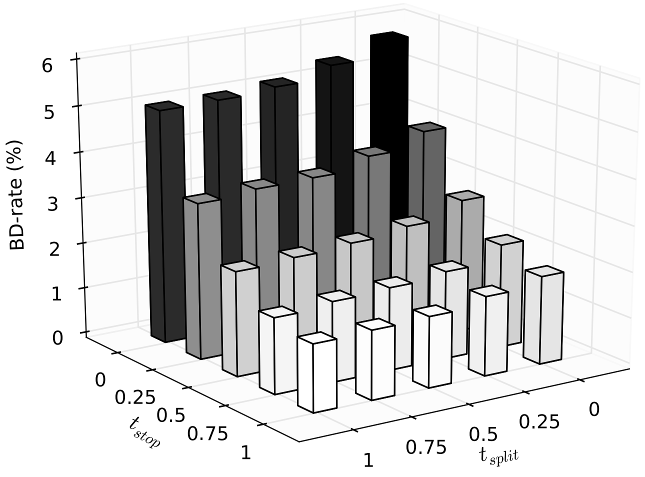
The model was evaluated on the full sequences as used in Section 3, as well as on *Kimono*, *ParkScene*, *BQTerrace*, *PartyScene*, *RaceHorses*, *BasketballPass*, *KristenAndSara* and *ChinaSpeed*. All sequences were encoded with the same parameters as in Section 3. The sequences were also re-encoded with no shift, so the original CU structure was reused. To allow measurements of complexity reduction and Rate-Distortion (RD) performance, all sequences were re-encoded with an unmodified reference encoder.

To determine the complexity reduction achieved by the model, the encoding time of the fast encoder is compared to the encoding time of the reference encoder. Since the time to decode and shift the original bitstream is negligible compared to the re-encoding step, accounting only for encoding times is a good indicator for complexity. The fast encoder is compared to the reference encoder in terms of time saving (TS):

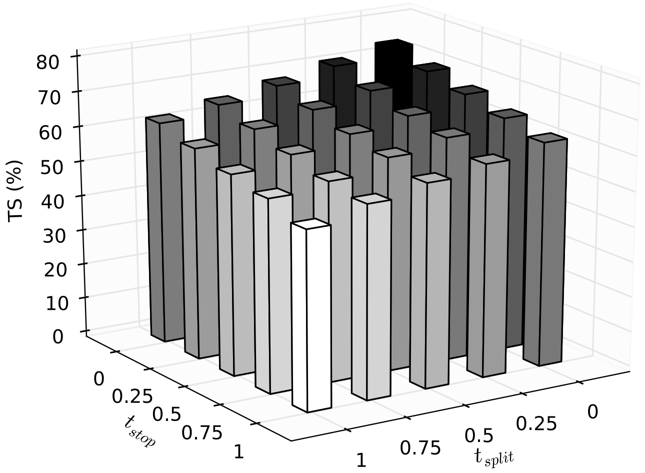
$$TS(\%) = \frac{T_{ref}(ms) - T_{fast}(ms)}{T_{ref}(ms)}$$

The difference in RD performance between sequences encoded with the reference encoder and sequences encoded with the fast encoder is measured in Bjøntegaard delta (BD) rate [17]. This shows the average bitrate difference between four encoded sequences of each set over the measured PSNR-Y range. To calculate the PSNR-Y for a sequence, the shifted bitstream is decoded, shifted back by removing the padding, and compared to the original, uncompressed video.

For the majority of test sequences, enforcing a stop too early has a worse impact on quality than incorrectly forcing a split, whereas the impact on complexity is similar. Changing the split threshold t_{split} causes the BD-rate to vary less than the stop threshold t_{stop} (Fig. 2(a)). On the other hand, the complexity reduction varies similarly for both thresholds (Fig. 2(b)). Therefore, the best compromise between BD-rate and complexity reduction is achieved for $t_{stop} \geq t_{split}$. These thresholds can be chosen depending on the quality and complexity requirements of the use case. In the rest of this paper, $t_{split} = 0$ and $t_{stop} = 0.5$ are selected as representative values.



(a)



(b)

Fig. 2. BD-rate (a) and time saving (b) for the sequence *ParkScene* with a shift of 16 pixels. Enforcing a stop (smaller t_{stop}) has a more negative impact on the BD-rate than enforcing a split (smaller t_{split}), whereas both have a similar positive effect on the time saving.

		BD-rate (%)				Time saving (%)			
Sequence	Average TU depth	No shift	Shift32	Shift16	Shift8	No shift	Shift32	Shift16	Shift8
RaceHorses	2.25	0.95	1.73	2.28	1.57	69.82	70.54	66.85	42.53
BQSquare	2.17	0.74	2.73	2.32	2.15	71.75	72.30	69.78	49.19
BlowingBubbles	2.16	0.79	1.91	2.01	1.46	70.90	71.40	68.92	45.33
CrowdRun	2.12	0.56	0.75	1.08	0.51	70.89	69.36	67.72	44.05
PartyScene	2.05	0.58	0.80	1.41	0.67	72.63	71.79	68.16	47.73
Average		**0.72**	**1.58**	**1.82**	**1.27**	**71.20**	**71.08**	**68.29**	**45.77**
RaceHorsesC	1.92	1.14	1.55	2.35	1.57	72.30	70.04	65.91	46.62
BasketballPass	1.85	0.83	1.48	2.49	2.26	71.40	71.11	67.43	44.11
BasketballDrillText	1.53	1.04	1.51	2.94	2.53	74.08	71.49	67.17	49.81
ChinaSpeed	1.51	1.76	2.30	3.10	2.29	75.07	72.11	68.03	53.57
BQTerrace	1.50	1.45	2.01	1.92	1.81	75.24	72.15	70.24	56.15
ParkScene	1.42	1.51	1.93	2.91	2.28	74.72	71.62	69.42	54.40
Cactus	1.30	1.29	2.19	3.69	2.75	74.90	71.41	69.69	54.39
Kimono	1.12	1.15	1.85	2.93	3.30	75.80	71.45	69.30	57.33
Average		**1.27**	**1.85**	**2.79**	**2.35**	**74.19**	**71.42**	**68.40**	**52.05**
FourPeople	0.67	2.02	4.15	7.16	7.43	78.54	75.46	72.86	62.06
KristenAndSara	0.57	2.86	4.56	7.59	9.01	78.92	75.05	71.94	65.59
Johnny	0.50	3.20	4.77	8.43	12.87	79.38	76.26	74.28	67.15
Average		**2.69**	**4.49**	**7.73**	**9.77**	**78.95**	**75.59**	**73.03**	**64.93**

Table 1. Results for $t_{split} = 0$ and $t_{stop} = 0.5$. The model produces better quality (a lower BD-rate) for a smaller transform size (higher TU depth) in the input bitstream. The type of shift influences the performance of the method as well. Shifts that let more blocks remain aligned with a grid of their own size produce a better result.

Table 1 shows the BD-rate and TS for all tested sequences. The sequences are ordered according to descending average TU depth. The BD-rate and TS are given for transcoding without shifting, as well as for three different shifts.

The results show that the model has a better RD performance for smaller transform sizes. For instance, the sequences which contain many small blocks (average TU depth greater than 2) have an average BD-rate of 1.58% for a shift of 32 pixels. On the other hand, the sequences with larger blocks (average TU depth less than 1) have a BD-rate of 4.49% for the same shift. However, the sequences with larger blocks have an average TS of 75.59% for a shift of 32 pixels, whereas the sequences with smaller blocks have an average TS of 71.08%. This indicates that for input bitstreams with predominantly large TU sizes, the model stops splitting early. In that case, bigger CU sizes will be selected more often. Consequently, there are less CUs for which the PU- and TU-structure must be evaluated.

The model also performs better for shifts that allow more blocks to remain aligned with a grid. For no shift and an average TU depth greater than 2, the TS is 71.20% while the BD-rate is 0.72%. This quality loss is due to requantisation slightly altering the optimal CU structure compared to the input bitstream, and because the Sample Adaptive Offset filter and some mode decisions in the HM software use information from the skipped process. For a shift of 32 and 16 pixels, the average TS decreases to 71.08% and 68.29%, while the BD-rate increases to 1.58% and 1.82%. This behaviour is likely caused by the preservation of the original CU structure for block sizes smaller than the shift (Section 3). For a shift of 8 pixels, the BD-rate decreases to 1.27%. However, the TS is reduced to 45.77%, because the splitting behaviour for block sizes of 16 pixels is not determined by the TU depth. Instead, the CUs are evaluated at both depth 2 and 3.

5. CONCLUSION

Our analysis determined that the transform size is a good indicator for the CU structure of the output bitstream. Moreover, the original CU structure is better retained when good grid alignment is maintained for more block sizes.

By evaluating the model, we found that the behaviour varies depending on the average size of the TUs in the original bitstream. Bigger block sizes in the input stream result in worse quality because the model causes the CUs in the output stream to stop splitting too early. Moreover, incorrectly forcing the encoder not to split a CU has a worse impact on the quality than erroneously forcing a split. However, both forcing a stop or a split have a similar impact on complexity reduction. On average, BD-rates below 2% with complexity reductions up to 71% are achieved for sequences with a small average TU size (average TU depth greater than 2). For a bigger average TU size (average TU depth less than 1), the BD-rate increases up to 10%, while the complexity reduction for transcoding shifts reaches up to 76%.

6. REFERENCES

[1] J. Ohm, G.J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[2] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, 2003.

[3] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct 2005.

[4] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, Jan 2005.

[5] E. Peixoto, B. Macchiavello, E.M. Hung, A. Zaghetto, T. Shanableh, and E. Izquierdo, "An H.264/AVC to HEVC video transcoder based on mode mapping," in *20th IEEE International Conference on Image Processing (ICIP)*, Sept 2013, pp. 1972–1976.

[6] L. Pham Van, J. De Cock, G. Van Wallendael, S. Van Leuven, R. Rodriguez-Sanchez, J.L. Martinez, P. Lambert, and R. Van de Walle, "Fast transrating for high efficiency video coding based on machine learning," in *20th IEEE International Conference on Image Processing (ICIP)*, Sept 2013, pp. 1573–1577.

[7] G. Fernandez-Escribano, H. Kalva, J.L. Martinez, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "An MPEG-2 to H.264 Video Transcoder in the Baseline Profile," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 5, pp. 763–768, May 2010.

[8] T. Shanableh, E. Peixoto, and E. Izquierdo, "MPEG-2 to HEVC Video Transcoding With Content-Based Modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1191–1196, July 2013.

[9] E. Peixoto, T. Shanableh, and E. Izquierdo, "H.264/AVC to HEVC Video Transcoder Based on Dynamic Thresholding and Content Modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 99–112, Jan 2014.

[10] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang, "A fast H.264-based picture-in-picture (PIP) transcoder," in *IEEE International Conference on Multimedia & Expo (ICME)*, June 2004, pp. 1691–1694.

[11] Y. Michalevsky and T. Shoham, "Fast H.264 picture in picture (PIP) transcoder with B-slices and direct mode support," in *15th Mediterranean Electrotechnical Conference (MELECON)*, April 2010, pp. 862–867.

[12] D. Grois, M. Loants, O. Hadar, R. Ohayon, and N. Amram, "Ultra-fast live video-in-video insertion for H.264/AVC," in *IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2013, pp. 635–636.

[13] J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle, "Efficient Picture-in-Picture Transcoding for High Efficiency Video Coding," in *IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, Sept 2013, pp. 514–515.

[14] G.J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[15] I. H. Witten and F. Eibe, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.

[16] F. Bossen, D. Flynn, and K. Suehring, "HEVC HM12 Reference Software," Tech. Rep. JCTVC-N1010, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Aug. 2013.

[17] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.