AN H.264/AVC TO HEVC VIDEO TRANSCODER BASED ON MODE MAPPING

E. Peixoto ⋆† B. Macchiavello ⋆ E. M. Hung ⋆ A. Zaghetto ⋆ T. Shanableh ⋆ E. Izquierdo †

⋆ Universidade de Brasília, Brazil.
† College of Engineering, American University of Sharjah, UAE.

ABSTRACT

The emerging video coding standard, HEVC, was developed to replace the current standard, H.264/AVC. However, in order to promote inter-operability with existing systems using the H.264/AVC, transcoding from H.264/AVC to the HEVC codec is highly needed. This paper presents a transcoding solution that uses machine learning techniques in order to map H.264/AVC macroblocks to HEVC coding units (CUs). Two alternatives to build the machine learning model are evaluated. The first uses a static training, where the model is built offline and used to transcode any video sequence. The other uses a dynamic training, with two well-defined stages: a training stage and a transcoding stage. In the training stage, full re-encoding is performed while the H.264/AVC and the HEVC information are gathered. This information is then used to build a model, which is used in the transcoding stage to classify the HEVC CU partitioning. Both solutions are tested with well-known video sequences and evaluated in terms of rate-distortion (RD) and complexity. The proposed method is on average 2.26 times faster than the trivial transcoder using fast motion estimation, while yielding a RD loss of only 3.6% in terms of bitrate.

Index Terms— Transcoding, HEVC, machine learning.

1. INTRODUCTION

The emerging video coding standard called High Efficiency Video Coding (HEVC), known formally as Recommendation ITU-T H.265 or ISO/IEC 23008-2 and referred here simply as HEVC, was developed by a collaboration between the ITU-T and JCT-VC groups to replace the current H.264/AVC standard [1]. The main goal of the HEVC codec is not to provide video compression with different features, such as error correction or scalability capabilities, but rather to significantly improve the rate-distortion (RD) performance compared to the H.264/AVC, in order to allow for new applications such as ultra high-definition resolutions. The HEVC outperforms the H.264/AVC by up to 50%, in terms of RD performance [2]. Therefore, the motivation for a H.264/AVC to HEVC transcoder is twofold: (i) to be ready to promote inter-operability for the legacy video encoded in H.264/AVC format; and (ii) to be able to take advantage of the superior RD performance of the HEVC, especially for systems that have extensive video libraries.

By definition, transcoding is the process that converts from one compressed bitstream (called the source or incoming bitstream) to another compressed bitstream (called the transcoded or outgoing bitstream) [3, 4, 5]. While it is always possible to use the trivial transcoder approach, consisting of fully decoding the source bitstream and completely re-encoding in the target bitstream, without using any information found in the source bitstream, this approach is not efficient from the complexity point of view. Therefore, the main goal in transcoding is to use the information found in the source bitstream to speed-up the transcoding process, while still maintaining a high quality for the decoded video.

The contributions reported in the transcoding literature may be classified as algorithms for mode mapping, for motion vector (MV) approximation and for MV refinement. The goal of mode mapping algorithms is to use information in the incoming bitstream to map the modes for the target codec, in order to avoid testing all macroblock (MB) modes for the target codec. The goal for both MV approximation and MV refinement algorithms is to reduce the cost of the motion estimation (ME) module in the target codec. The MV approximation algorithms attempt to maximise the reuse of the MVs found in the source bitstream, while the MV refinement algorithms attempt to improve the reused and approximated MVs.

The HEVC is able to use the same GOP (group of pictures) structure as the H.264/AVC and, if this is the case, it would not need MV approximation algorithms, as MV reuse would always be possible. At the same time, in the reference software implementation for the HEVC, fast ME is already used, so MV refinement algorithms would not yield the same gain as in other transcoders. However, the HEVC codec uses a very large number of modes, making mode mapping algorithms the most important part for this transcoder.

In this paper, we present a transcoding architecture that focuses on the mode mapping algorithms, particularly designed to map the 16 MBs found in the H.264/AVC to 64 × 64 and 32 × 32 coding units (CUs) found in the HEVC. The pre-
sent model mapping algorithm consists of a low complexity machine learning algorithm, based on linear discriminant functions (LDFs) [6]. Two alternatives to build the machine learning model are evaluated, with dynamic and static trainings. In the former, there are two well-defined stages: a training stage, where all modes are tested, and a transcoding stage, where a subset of the modes is tested based on the classification returned by the model. In the latter, the training stage is performed beforehand, and the same model is used to transcode any video sequence. Both alternatives are tested with well-known video sequences and evaluated in terms of rate-distortion (RD) and complexity.

2. RELATED WORK

There is an extensive literature on transcoding between many different codecs. Here, we focus on transcoders that involve the HEVC or that involve machine learning techniques.

Since the HEVC is a very new codec, there are not many transcoding solutions from H.264/AVC to the HEVC. One solution [7] is based on the power-spectrum rate-distortion optimization (PS-RDO) method [8]. In this method, the cost of a motion vector in the transcoder is estimated from the MV variation and power-spectrum of the prediction signal resulting from that MV. In this work, the PS-RDO model is used both for mode mapping, to determine the CU partitioning in the HEVC, and for MV approximation, determining the MV used for each prediction unit (PU).

In a previous work [9], we proposed a simple transcoder based on the MV variance distance (MVVD). The MVVD is computed for each CU, and it is defined as the square root of the variance for each H.264/AVC MV component within the region defined by that CU. The transcoding algorithm starts at the largest CU, and uses two thresholds, $T_{low}$ and $T_{high}$, in order to decide which modes will be tested for that CU, and if that CU will be split. If the CU is split, the algorithm repeats itself for each of the four sub-CUs. Although this algorithm presents a good performance over a range of sequences, its main weakness is that it uses fixed thresholds (computed offline), regardless of the content of the sequences or the quantization parameters used to encode it and, therefore, it achieves different results depending on the sequence being transcoded.

In order to overcome the limitations of the transcoder based on MVVD, we have also proposed two techniques [10]: dynamic thresholding and content modeling using LDFs. Both techniques are based on two stages: a training and a transcoding stage. The dynamic thresholding is similar to the MVVD transcoder, but it uses information gathered in the training stage in order to compute the thresholds that are used to transcode the rest of the sequence (in the transcoding stage). The transcoder proposed in this paper is a modification of the content modeling transcoder.

Some works use machine learning algorithms to perform the mode mapping in the context of MPEG-2 to H.264/AVC transcoding [11, 12, 13]. All of these solutions are based on a similar idea: for each MB, some features are computed using information found the MPEG-2 bitstream and stored in a dataset, along with the optimal mode used to encode said MBs found using the trivial transcoder. The dataset is constructed using a small number of frames of a few test sequences. Once the dataset is generated, a machine learning approach is used to map the features computed using the incoming bitstreams into modes to be tested in the target codec. The training is performed offline, with the goal of developing a single, generalized, mapping that can be used for transcoding any MPEG-2 video. In the first of these solutions [11], the features used include the MPEG-2 MB coding mode, the coded block pattern, and the means and variances for each $4 \times 4$ residual block, generating a total of 37 features. In the other solutions [12, 13], the list of features was expanded to include the MPEG-2 DCT coefficients, neighbouring MB information, coded block pattern, the motion vectors, the mean and variance of the $4 \times 4$ residual blocks, and the variance of the means and mean of variances for each group of means and variances, generating a total of 131 features.

3. THE PROPOSED TRANSCODER

The two transcoding solutions presented in this paper share essentially the same architecture. The only difference is in how they build the model to map H.264/AVC MBs into HEVC CUs. Therefore, the common parts of both transcoders are presented first, and the difference between them, namely the dynamic and static trainings, are detailed afterwards.

The transcoding operations are based on the HEVC CU. The decision always starts at the LCU, used here as $64 \times 64$, and continues to the sub-CUs. According to the depth of the current CU, different mapping strategies are used. Regardless
of the depth, a simple MV reuse and refinement algorithm is applied. Each of these three modules are detailed next.

The content modeling approach is used for the depths 0 and 1 (i.e., for CUs of 64 × 64 and 32 × 32 pixels). The machine learning algorithm chosen is the LDFs [6]. In order to use this, seven incoming features are computed using the H.264/AVC information. For all features, only the H.264/AVC information for the region within the current CU is used. The features, detailed elsewhere [10], are: (i) the total number of H.264/AVC partitions; (ii) the variance of the x and y MV component; (iii) the MV variance distance; (iv) the MV phase variance; (v) the number of non-zero DCT coefficients encoded; and (vi) the average energy of the DCT coefficients encoded (this is energy of the coefficients divided by the number of coefficients. If there are no non-zero DCT coefficients within the CU, then it is considered as zero).

The transcoder attempts to classify the CU in one of two classes: split and not split. If the CU is classified as split, then only the SKIP mode is tested for the current depth, and the CU is split into four sub-CUs. Then, for each sub-CU, the algorithm is applied again (or, if the next depth is the depth 2, a different mode mapping is used). If the CU is classified as not split, then all PU modes for this depth are tested, the CU is not split, and the transcoder proceeds to the next CU. If the CU contains any intra coded MB, then all PU modes are tested at this depth, and the CU is split. This algorithm is shown in Fig. 1. The main difference between the algorithm presented in this paper and the content modeling transcoder previously reported [10] is that the latter uses a threshold based on the MVVD to pre-classify the split class, while the former uses only the LDF to perform the classification.

For the depths 2 and 3, the transcoder could directly reuse the partitioning used by the H.264/AVC. However, tests have shown that keeping the exact same partitions as the incoming H.264/AVC leads to a higher loss in RD performance and yields a small gain in complexity. For this reason, a simple mapping is proposed: to test, in the HEVC, the PUs that are larger than the H.264/AVC partitioning. For instance, if the H.264/AVC MB was partitioned into two 16 × 8 partitions, then only the SKIP, 2N × 2N, 2N × N and the asymmetric partitions 2N × nD and 2N × nU are tested in the HEVC.

Regarding the MV reuse, tests have shown a high correlation between the H.264/AVC and the HEVC MVs [9]. Therefore, for any outgoing HEVC PU size, all H.264/AVC MVs within the area defined by that PU are considered for integer ME, and no further refinement is performed at this level. Then, the default HEVC sub-pixel search is applied.

3.1. Building the LDF model

The LDF model consists of the weights that optimally classify the feature vectors in the training data set, according to the classes to which these feature vectors belong. Here, the LDF weights are computed by minimising the $L^2$ norm [6], because it leads to a non-iterative solution that is suitable to use within the transcoder loop. The difference between the two transcoding solutions is the content of the training data set, as explained next. Both solutions are shown in Fig. 2.

3.1.1. Static Model

In the static model, the first 5 inter frames of a few sequences, namely PartyScene, RaceHorses and BQMall, all part of the HEVC test sequences, encoded with four different QPs (47, 32, 27, 22), are encoded and the H.264/AVC feature vectors, along with the HEVC class (i.e., split or not split), are stored. Then, the LDF weights are computed and stored within the transcoder. These weights are then used to map H.264/AVC MB into HEVC CUs when transcoding any video sequence. Therefore, when transcoding a video, just the training stage is performed. This is shown in Fig. 2(a).

3.1.2. Dynamic Model

In the dynamic model, the LDF model is computed using information of the current sequence that is being transcoded, as shown in Fig. 2(b). For a sequence of n frames, the first k frames are used for training, and the transcoding operates in the following $n - k$ frames. In the training stage, all modes in the HEVC are tested, and the H.264/AVC information is used only for training purposes. Using the information gathered at this stage, the transcoder builds the model that will be used during the transcoding stage. Note that, if the number of frames used for training is kept small, the impact on the transcoding complexity will be small as well, as the ratio $\frac{n-k}{n}$ will be close to 1. In a previous work [10], it is discussed the number of frames needed to build an efficient model for the dynamic training. It was found that, while using more frames may lead to a better model, 10 frames are enough to result in a sufficient data set.
Table 1. Transcoder Results compared to RT-EPZS (BD-Rate 0.0 and Speed-Up 1.0).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Method</th>
<th>BD-Rate %</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Average</td>
</tr>
<tr>
<td>Kimono1 1920 x 1080</td>
<td>RT-MVVD</td>
<td>7.98</td>
<td>2.63</td>
</tr>
<tr>
<td></td>
<td>24 Hz</td>
<td>2.75</td>
<td>2.91</td>
</tr>
<tr>
<td></td>
<td>PT-Static</td>
<td>3.71</td>
<td>3.92</td>
</tr>
<tr>
<td></td>
<td>PT-Dynamic</td>
<td>3.21</td>
<td>3.68</td>
</tr>
<tr>
<td>Tennis 1920 x 1080</td>
<td>RT-MVVD</td>
<td>3.04</td>
<td>2.68</td>
</tr>
<tr>
<td></td>
<td>24 Hz</td>
<td>2.49</td>
<td>2.05</td>
</tr>
<tr>
<td></td>
<td>PT-Static</td>
<td>3.21</td>
<td>3.68</td>
</tr>
<tr>
<td></td>
<td>PT-Dynamic</td>
<td>2.92</td>
<td>3.68</td>
</tr>
<tr>
<td>ParkScene 1920 x 1080</td>
<td>RT-MVVD</td>
<td>4.60</td>
<td>8.04</td>
</tr>
<tr>
<td></td>
<td>24 Hz</td>
<td>4.94</td>
<td>3.63</td>
</tr>
<tr>
<td></td>
<td>PT-Static</td>
<td>5.14</td>
<td>4.31</td>
</tr>
<tr>
<td></td>
<td>PT-Dynamic</td>
<td>4.34</td>
<td>3.63</td>
</tr>
<tr>
<td>Cactus 1920 x 1080</td>
<td>RT-MVVD</td>
<td>9.00</td>
<td>9.56</td>
</tr>
<tr>
<td></td>
<td>50 Hz</td>
<td>5.95</td>
<td>4.86</td>
</tr>
<tr>
<td></td>
<td>PT-Static</td>
<td>6.28</td>
<td>4.77</td>
</tr>
<tr>
<td></td>
<td>PT-Dynamic</td>
<td>5.52</td>
<td>4.62</td>
</tr>
<tr>
<td>BasketballDrive 1920 x 1080</td>
<td>RT-MVVD</td>
<td>4.64</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td>50 Hz</td>
<td>5.21</td>
<td>4.56</td>
</tr>
<tr>
<td></td>
<td>PT-Static</td>
<td>5.52</td>
<td>4.62</td>
</tr>
<tr>
<td></td>
<td>PT-Dynamic</td>
<td>5.21</td>
<td>4.56</td>
</tr>
</tbody>
</table>

4. EXPERIMENTAL RESULTS

In order to evaluate the proposed transcoder, four transcoding options are compared: (i) the trivial transcoder, using fast ME and fast mode decision (RT-EPZS); (ii) the reference transcoder based on the MV Variance Distance [9], with parameters $T_{low} = 1$ and $T_{high} = 100$ (RT-MVVD); (iii) the proposed transcoder with static training (PT-Static); and (iv) the proposed transcoder with dynamic training (PT-Dynamic). For the H.264/AVC, the reference software JM 14.2 [14] is used, and for the HEVC, the reference software HM 9.1 [15] is used. For all sequences, the QPs are 37, 32, 27 and 22, and the full length of the sequence is transcoded (10 seconds). Both codecs are using a low-delay coding configuration with 1 reference frame.

The complete results are shown in Table 1. It can be seen that the RD performance of RT-MVVD shows the most variance among the transcoding options tested. This is expected, since the fixed thresholds used for this option may be adequate for some sequences, but not for others. For instance, the thresholds are adequate for Kimono1 sequence (where it shows a loss of only $+2.65\%$). On the other hand, the RD performance is particularly poor for ParkScene ($+8.04\%$) and Cactus ($+9.50\%$) sequences. In terms of speed-up, this option is also the slowest among all options tested, particularly for Kimono1 sequence (only 2.09 times faster than RT-EPZS).

Using the static training, compared to RT-MVVD, it shows a lower RD loss for Cactus and ParkScene sequences, but a worse RD loss for Kimono1 sequence. For Tennis and BasketballDrive, the RD loss is comparable. It must be noted that the occurrence of intra macroblocks in the H.264/AVC bitstream is much higher for these sequences, which impacts the transcoder behaviour. However, the overall RD losses are more limited (up to a maximum of 5.27% for Cactus sequence) compared to RT-MVVD. In terms of speed-up, this option is always faster than RT-MVVD.

Using the dynamic training yields the best overall results in terms of RD performance, and it also shows more uniform results, with RD losses in the range of 2.37 to 4.86 %. It outperforms PT-Static for all sequences, and RT-MVVD for most sequences, except for Kimono1 and BasketballDrive, where RT-MVVD shows a good performance. Even in this case, however, the results for PT-Dynamic are comparable to RT-MVVD. In terms of speed-up, it is as fast as PT-Static during the transcoding stage. This can be seen in Fig. 3, where the number of transcoded frames versus elapsed time is shown for the specific case of Kimono1 sequence using QP 27. In this figure, it is also possible to see that, even though PT-Dynamic performs a training during the first 10 frames (therefore, resulting in the same speed-up as the trivial transcoder), it reaches RT-MVVD after just 35 frames, as the transcoding stage is significantly faster.

5. CONCLUSIONS

In this paper we compared two transcoding solutions from H.264/AVC to HEVC based on mode mapping using linear discriminant functions. Our experiments show that both of the proposed transcoding solutions outperform the previous transcoder based on MV variance distance. Regarding the two presented solutions we show that using the dynamic training yields more stable results over a wide range of sequences. Although the gain of the dynamic training over the static training, in terms of RD performance, may not seem so expressive, it is important to notice that the results for the latter are already good, not leaving enough room for significant gains. Also, the only advantage of the static training is that it is slightly faster, since it does not have an online training stage. However, during the transcoding stage, both solutions have similar complexity. Moreover, the reliability of the dynamic training may compensate for this small increase in complexity. Moreover, even though our tests did not show it, it is possible that using the static training may fail for some specific sequences, the same way as the fixed threshold algorithms failed for Cactus and ParkScene sequences.

Considering the classification in only two classes, the chosen features are enough to build an efficient model. However, in order to reduce the complexity, the mode mapping should consider a larger number of classes. To achieve that goal, the set of features might also need to be changed. The way that the transcoder handles intra H.264/AVC macroblocks should also be improved, as the results for Tennis sequence indicates. All of these topics are subject of future work.
6. REFERENCES


