Peixoto, E., Shanableh, T., & Izquierdo, E. (2014). H. 264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling. IEEE Transactions on Circuits and Systems for Video Technology, 24(1), 99-112.
DOI: 10.1109/TCSVT.2013.2273651

H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling

Eduardo Peixoto, *Member, IEEE,* Tamer Shanableh, *Member, IEEE,* and Ebroul Izquierdo, *Senior Member, IEEE*

***Abstract*—The new video coding standard, HEVC, was developed to succeed the current standard, H.264/AVC, as the state of the art in video compression. However, there is a lot of legacy content encoded with H.264/AVC. This paper proposes and evaluates several transcoding algorithms from the H.264/AVC to the HEVC format. In particular, a novel transcoding architecture, in which the first frames of the sequence are used to compute the parameters so that the transcoder can "learn" the mapping for that particular sequence, is proposed. Then, two types of mode mapping algorithms are proposed. In the first solution, a single H.264/AVC coding parameter is used to determine the outgoing HEVC partitions using dynamic thresholding. The second solution uses linear discriminant functions to map the incoming H.264/AVC coding parameters to the outgoing HEVC partitions. This paper contains experiments designed to study the impact of the number of frames used for training in the transcoder. Comparisons with existing transcoding solutions reveal that the proposed work results in much lower rate-distortion loss at a competitive complexity performance.**

***Index Terms*—Transcoding, HEVC, machine learning.**

I. Introduction

THE new video coding standard, so called High Efficient Video Coding (HEVC) [1], developed by the JCT-VC group to replace the current H.264/AVC standard [2]. 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 performance, compared to the current standard, H.264/AVC, in order to allow for new applications, such as beyond high-definition resolutions (so called $4K$, 3840×2160 pixels, and $8K$, 7680×4320 pixels).

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, when new applications using the HEVC emerge; and (ii) to be able to take advantage of the superior rate-distortion performance of the HEVC. The first will be useful when the first applications are launched that use the new standard, while the second could be used straight away to migrate the abundant existent video content encoded in the H.264/AVC format.

In this paper, we build on our previous work [3], in which we proposed a H.264/AVC to HEVC transcoder based on a metric called Motion Vector Variance Distance [3], and another work in which we proposed a MPEG-2 to HEVC transcoder based on content modeling [4]. Here, we explore other transcoding solutions based on a content-based modeling approach, in which the transcoder adapts the transcoding parameters based on the contents of the sequence being transcoded, and further evaluates the concept of content-based modeling on the transcoder efficiency.

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) [5], [6], [7]. Several properties may change during transcoding: the video format [8], [9], the bitrate of the video [10], [11], the frame rate [12], [13], the spatial resolution [14], [15], the coding tools used (i.e., one bitstream might use B frames, while the other might not, or scalability layers are added to the target bitstream) [16], and even the insertion of new information on the video, such as watermarking [17], hidden data [18] or a layer for error resilience [19].

In transcoding, it is always possible to use a combination of a suitable decoder and encoder in tandem, completely decoding the incoming bitstream and then completely re-encoding it in the target format. Here, this is defined as the *trivial transcoder*. While this approach usually achieves high quality of the transcoded sequence and can be used for any target conditions, it is not efficient from the point of view of complexity.

The two main categories of transcoders are: *homogeneous transcoding* (the conversion of bitstreams within the same format) and *heterogeneous transcoding* (i.e., between different formats). Homogeneous transcoding is commonly used to change the bitstream in order to adapt it to a new functionality, such as a different bitrate or spatio-temporal resolution. Heterogeneous transcoding can also provide the functionalities of homogeneous transcoding, such as reduction of bit rate and change of spatio-temporal resolution, but it is mainly defined by the change of format. The H.264/AVC to HEVC transcoder falls in the latter category.

In many solutions, heterogeneous transcoding is achieved by completely decoding the source stream and re-encoding it in the target format reusing information present in the source bitstream to speed up the transcoding. This is known as the *cascaded pixel domain approach* [5], [6].

This paper is organised as follows: Section II provides a review of the relevant literature on heterogeneous transcoding, especially on algorithms for mode mapping, which is the main focus of this paper. Section III details our previous work on the topic, which is used as benchmark to evaluate the transcoding options proposed here, while Section IV details

these transcoding options. Finally, Section V presents the experiments and Section VI concludes the paper.

II. Relevant Literature

A simple way of classifying the contributions reported in the transcoding literature is to separate them into algorithms for mode mapping, algorithms for motion vector approximation and algorithms for motion vector refinement. The goal of the mode mapping algorithms is to use information on the incoming bitstream in order to avoid testing all modes for the target format. On the other hand, the goal of the motion vector approximation algorithms is to maximize the reuse of the motion vectors in the incoming bitstream in order to avoid costly motion estimation operations in the target encoder. Finally, the goal of the motion vector refinement algorithms is to improve the reused and approximated motion vectors so that a good prediction can be achieved.

The HEVC is able to use the same reference frame structure as the H.264/AVC [20] and, if this is the case, it would not need motion vector approximation algorithms. At the same time, the impact of the motion estimation module in the HEVC complexity is much smaller than the impact for other codecs, such as the H.264/AVC, and so MV refinement algorithms would not yield the same gain as in other transcoders. However, the HEVC uses a large number of modes, making mode mapping algorithms very important for the transcoder.

In order to reuse the coding mode of a particular macroblock in the incoming bitstream, a range of algorithms have been proposed. A simple mode mapping algorithm was proposed in the context of a H.264/AVC to MPEG-2 transcoder [8]. In this algorithm, the H.264/AVC macroblock types are classified in three categories, skipped, inter and intra, for macroblocks encoded in SKIP mode, inter or intra modes, respectively. Then, in the transcoder, only the modes associated with these classes are tested. Another simple algorithm, used in the context of VC-1 to H.264/AVC transcoding, was proposed [21]. In this work, since the VC-1 codec offers a smaller number of modes than the H.264/AVC (for instance, only blocks sizes of 16×16 and 8×8 are used for motion compensation, and there is no skip mode for a macroblock), the transcoder uses both the macroblock type and the size of the transform used in VC-1, proposing some rules based on heuristics, summarized in the form of a look-up table, to decide which modes are tested in the outgoing H.264/AVC video.

The block mode statistics are used in a MPEG-4 to H.264/AVC transcoder [22]. In this work, several test sequences are transcoded using a trivial transcoder in order to gather the macroblock mode conversion statistics. This information is then used to generate a look-up table, which is used during transcoding to decide which H.264/AVC modes are tested according to the MPEG-4 mode. A similar approach was used in a H.264/AVC to MPEG-4 transcoder [23].

In other reported solutions, the idea of using the block mode statistics is expanded. Machine learning algorithms are used to map the modes in the incoming bitstream and decide how the modes in the target codec are tested, in the context of MPEG-2 to H.264/AVC transcoding [24], [25], [26]. All these solutions are built around similar ideas: first, few frames of test sequences are transcoded using a trivial transcoder. For these frames, some features are computed and stored for each macroblock, along with the optimal mode used to encode said macroblocks. Then, a machine learning approach is used to generate an algorithm to map 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 [24], the features used include the MPEG-2 macroblock coding mode, the coded block pattern, and the means and variances for each 4×4 residual block, generating a total of 37 features. In the other solutions [25], [26], the list of features was expanded to include the MPEG-2 DCT coefficients, neighbouring macroblock information, coded block pattern, the motion vectors, the mean and variance of the 4×4 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. A similar approach was presented by some of the same authors in the context of a Wyner-Ziv to H.264/AVC transcoder [27]. In this solution, three features are used to generate the mapping algorithm, being the SAD of the residual computed in the Wyner-Ziv decoding process, the length of the motion vector generated by the Wyner-Ziv decoding process, and information from the Wyner-Ziv reconstruction process, and the same offline training process is used.

A transcoding solution from H.264/AVC to HEVC has also been proposed by Zhang et. al. [28]. In this work, a method to transcode intra frames is proposed, mainly based on selective merging of the incoming H.264/AVC intra modes and mapping them to larger HEVC CUs and PUs, according to the prediction direction found in the H.264/AVC bitstream. For inter pictures, it builds on the power-spectrum based rate-distortion optimization (PS-RDO) [29]. In this method, the cost of a motion vector in the transcoder is estimated from the motion vector variation and power-spectrum of the prediction signal resulting from that motion vector. The PS-RDO model is used to determine both the CU partitioning and the motion vector used for each PU.

III. Previous work

In this work, we build on our previous work of H.264/AVC to HEVC transcoder [3]. Two transcoders were proposed: one is based on MV reuse; and the other is based on a metric called MV Variance Distance. Both are briefly discussed here, as they are used to evaluate the proposed transcoders in this paper.

All transcoding methods presented in this paper are based on mode mapping algorithms. Therefore, the main idea is to use the H.264/AVC information in order to decide the CU and PU partitioning, instead of testing every possible CU and PUs. In this section, and for the remainder of this paper, the testing of a PU is defined as the assessment of the best way to encode that particular PU (i.e., deciding the parameters - motion vectors, transforms, etc...) and producing a rate-distortion cost. Similarly, the testing of a motion vector is

defined as the evaluation of the cost of that motion vector, and comparing this cost with the motion vectors that were previously tested for that particular PU. In all cases, the default metrics used in the HM reference software are used.

A. A Transcoder based on MV Reuse

This simple transcoder was presented before [3] for the sole purpose of evaluating the effect of the motion vector reuse technique [30], [31] in a H.264/AVC to HEVC transcoder, which is a technique that is ubiquitous in transcoding. It is not by any means designed to be a very efficient transcoder, but it is useful to identify the areas where the largest gain, in terms of transcoding efficiency, can be achieved. The workflow of the algorithm is the same for each coding unit (CU) in the HEVC, and it is based on two main ideas:

1) If any part of this CU was encoded in intra mode in H.264/AVC, then all possible intra and inter modes are tested; otherwise, only the inter modes are tested.
2) For any inter partition unit (PU), all H.264/AVC motion vectors within the current PU are tested. The motion vectors are reused at integer-pixel level, without any further refinement at this level. Then, at half-pixel and quarter-pixel, the default HM search is applied (testing the eight neighbours at half-pixel level, then the eight neighbours at quarter-pixel level).

Note that this transcoder reuses the incoming motion vectors, but not the partitioning. All inter modes available in the HEVC are considered, including the Asymmetric Motion Partition, AMP [32] - for these partitions, the AMP speed-up setting, present in the $HM4.0rc1$ reference software [33], is enabled. The remaining HEVC settings are the same as the low-delay configuration for $HM4.0rc1$, including the fast mode decision flag (which is enabled). Therefore, this transcoder saves complexity only by avoiding the motion estimation (which is performed using a fast motion algorithm, based on the Enhanced Predictive Zonal Search, EPZS [34]), and by not testing all intra modes.

B. A Transcoder based on MV Variance Distance

This transcoder is based on a similarity metric, the MV Variance Distance, and, according to this metric, make the decision of how to test a particular CU. The MV Variance Distance metric produces a value $\upsilon \geq 0$ for each CU that can be tested in the HEVC. This metric is based on the variance of the H.264/AVC motion vectors, and it is computed as:

$$\upsilon = \sqrt{\left(\sigma_x^2\right)^2 + \left(\sigma_y^2\right)^2} \tag{1}$$

where σ_x^2 and σ_y^2 are the variances of each component of the H.264/AVC motion vectors within the CU. If the motion vectors do not have the same reference frame, they are scaled using the formula:

$$mv_{n \to n-\beta} = \left(\frac{\beta}{\alpha}\right) \cdot mv_{n \to n-\alpha} \tag{2}$$

where n is the current frame, $n-\alpha$ is the reference frame used by the H.264/AVC motion vector and $n-\beta$ is the target reference frame. If the scaling is necessary, then all motion vectors are scaled to the frame which is closest to the current frame. If any part of this CU was encoded using an intra mode, then the metric does not produce a value. Before the metric is computed, the motion vectors are propagated to the 4×4 blocks (i.e., the minimum size in H.264/AVC), and then the variance is calculated. This way, the motion vectors are weighted according to the area that they represent.

The idea of using this metric is that, if a large area has a low value υ, it means that all motion vectors in this area are similar, and thus it is more likely that this partition will be encoded using a larger CU in the HEVC, as it is more likely that a single motion vector will accurately predict the whole CU. On the other hand, if the same area has a high value υ, then the motion vectors within this area are very different, and thus it is less likely that this block will be encoded using a large CU in the HEVC (meaning it is more likely that it will be split). This way, it is possible to combine the information for different H.264/AVC macroblocks and make a decision for a large block in the HEVC codec.

Two thresholds are used to decide how a particular CU will be tested, namely T_{low} and T_{high}, which defines three different regions R_1 ($\upsilon \leq T_{low}$), R_2 ($T_{low} < \upsilon \leq T_{high}$) and R_3 ($\upsilon > T_{high}$).

The transcoder algorithm works independently for each CU, regardless of the CU size. The possible prediction units (PUs) that can be tested are divided in four groups: (i) SKIP; (ii) inter $2N \times 2N$; (iii) all remaining inter modes ($2N \times N$, $N \times 2N$, the AMP modes, and $N \times N$); and (iv) the intra modes ($2N \times 2N$, $N \times N$ and PCM). In addition, the transcoder can decide if the CU will be split or not (if so, the CU is split in four sub-CUs, as usual). Then, depending on the value of the MV Variance Distance υ for this particular CU, four different settings can be used:

1) if the CU is considered similar (i.e., if $\upsilon \leq T_{low}$), then only the PU groups (i) and (ii) will be tested and the CU will *not* be split;
2) if the CU is considered as dissimilar (i.e., if $\upsilon > T_{high}$), then only the PU groups (i) and (iii) will be tested, and the CU will be split.
3) if the CU is not similar nor dissimilar (i.e., if $T_{low} < \upsilon \leq T_{high}$), then the PU groups (i), (ii) and (iii) (i.e., all inter modes) will be tested and the CU will be split; and
4) if the value υ cannot be computed (i.e., if one H.264/AVC partition within the CU was encoded as intra), then all PU groups are tested and the CU is split.

The algorithm starts from the largest CU size (64×64), computing the MV Variance Distance υ for that CU. Then, according to the υ value for the CU, the transcoder tests only the PUs for the groups indicated by the aforementioned rules. If the rules state that the CU should not be split (i.e., if $\upsilon \leq T_{low}$), then the transcoder selects the best mode, according to the modes tested, and proceeds to the next CU. If the CU is split, the algorithm is applied in the same manner to each of the four sub-CUs, computing a new similarity value υ for the sub-CUs, until the final possible depth is reached.

IV. Proposed Transcoding Solutions

The transcoder presented in the previous section offers a good rate-distortion and complexity performance, and offers an interesting insight into tackling the H.264/AVC to HEVC transcoder. If the best mode to encode a given CU could be accurately predicted from information in the H.264/AVC bitstream, then a large amount of computation could be saved, with virtually no quality loss. Even if the best mode cannot be predicted, if the less likely modes are ruled out, then the transcoding could still be made faster with a small penalty in rate-distortion performance.

In the reviewed transcoder based on MV Variance Distance, however, the choice of the thresholds is still a concern. First, the thresholds are used regardless of the depth of the CU, which gives the same tolerance to decide the split for a 64×64 partition and a 16×16 partition. Second, and most important, the same thresholds are used for different sequences, regardless of the content of the sequences or the quantization parameters used to encode it. Finally, only the MV Variance distance is used in the decision loop - other information from the H.264/AVC bitstream are ignored.

In order to overcome these issues, we propose two new solutions: dynamic thresholding and content modeling using linear discriminant functions. This transcoder also uses features computed from information in the H.264/AVC bitstream (such as the MV Variance Distance) to decide how to test a given outgoing CU. However, the thresholds used in this proposed transcoder are computed adaptively for the current sequence being transcoded. Also, the features are only used to decide the modes for the 64×64 and 32×32 CUs. As for the 16×16 and 8×8 CUs, the mode used in the H.264/AVC bitstream is used in the decision process instead.

Other solutions involving mode mapping using machine learning algorithms have been proposed [24], [25], [26], [27]. However, all these solutions attempt to build a single, generalized, mapping that can be used for transcoding any bitstream. Also, most of these solutions use a large number of features, and use a machine learning algorithm whose training is complex, not being suitable for use in the proposed transcoder.

Both the dynamic thresholding and the linear discriminant functions are based on the same training stage. For a sequence of n frames, the first k frames are used for training, and the transcoding operates in the following $n - k$ frames, as shown in Fig. 1. When transcoding the training sub-sequence (i.e., the first k frames), 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 computes the thresholds or the weights for the linear discriminant functions, as explained in the following sections.

The advantage of using the training stage is that the transcoding parameters can be adapted to the content of the current sequence being transcoded. 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 addition, the transcoder efficiency can be improved, since the parameters will be more related to the content, compared to a transcoding solution where no training is performed, or then training is performed using other video sequences.

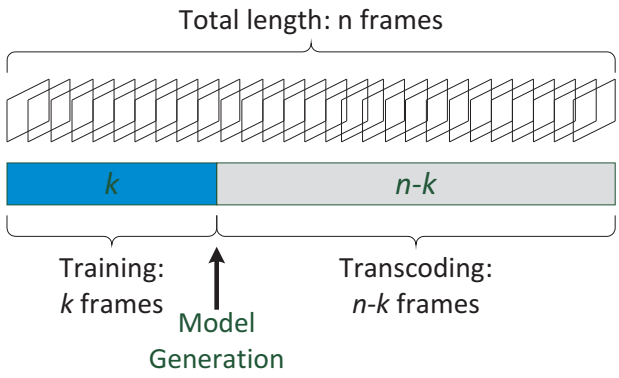

Fig. 1. Transcoding operation. When transcoding the first k frames, all possible modes in the HEVC are tested. This part is called *training* phase. Then, the transcoder builds the model (with information gathered in the training phase). Finally, it starts the *transcoding* phase, where the model is used to select which partitions will be tested.

A. Common transcoding settings

The transcoding settings explained in this sections are the same for all of the proposed transcoding options based on dynamic thresholding and content modeling using linear discriminant functions. Hence, these techniques, which are the main novelty of this paper, can be better compared to each other.

The MV Reuse and MV Refinement techniques are used in all of the proposed transcoders. For any outgoing HEVC PU size, all H.264/AVC motion vectors within the area defined by the PU are considered for integer motion estimation, and no further refinement is performed at the integer pixel level. The search is then followed by the default HEVC sub-pixel search.

Also, for all of the proposed transcoders defined in this section, the dynamic thresholding and the linear discriminant functions are applied only to the lower HEVC coding depths 0 and 1 (i.e., for CUs of size 64×64 or 32×32 - in this paper, we are always assuming a largest CU size of 64×64). For higher depths (i.e., for CUs of size 16×16 or smaller), a simple mode mapping algorithm that uses only the H.264/AVC partitioning is used. Note that the mode mapping for the higher depths is only used if the algorithm chooses to split the larger CU.

Tests have shown that keeping the exact same partitions as the incoming H.264/AVC leads to large rate-distortion losses, and the complexity reduction is small. For this reason, a different strategy has been designed. The rationale used is to test HEVC partitions that are of the same size or larger than the H.264/AVC partition. A simple look-up table is used to decide which modes will be tested in the HEVC outgoing bitstream, according to the H.264/AVC macroblock and sub-macroblock types. The complete look-up tables for the 16×16 and 8×8 CUs are listed in Tables I and II, respectively.

TABLE I
PUs TESTED FOR A 16×16 CU ACCORDING TO THE H.264/AVC MACROBLOCK TYPE.

		H.264/AVC Macroblock Type					
		PSKIP	16×16	16×8	8×16	8×8	INTRA
HEVC PUs Tested	*SKIP/MERGE*	X	X	X	X	X	X
	$2N \times 2N$		X	X	X	X	X
	$2N \times N$			X		X	X
	$N \times 2N$				X	X	X
	$2N \times nU$			X		X	X
	$2N \times nD$			X		X	X
	$nR \times 2N$				X	X	X
	$nL \times 2N$				X	X	X
	$N \times N$						
	INTRA						X
	SPLIT					X	X

TABLE II
PUs TESTED FOR A 8×8 CU ACCORDING TO THE H.264/AVC SUB-MACROBLOCK TYPE.

		H.264/AVC Sub Macroblock Type					
		BSKIP	8×8	8×4	4×8	4×4	INTRA
HEVC PUs Tested	*SKIP/MERGE*	X	X	X	X	X	X
	$2N \times 2N$		X	X	X	X	X
	$2N \times N$			X		X	X
	$N \times 2N$				X	X	X
	$2N \times nU$			X		X	X
	$2N \times nD$			X		X	X
	$nR \times 2N$				X	X	X
	$nL \times 2N$				X	X	X
	$N \times N$					X	X
	INTRA						X

B. Proposed Dynamic Thresholding

In this technique, a single feature from the incoming H.264/AVC bitstream is considered, and the mapping is performed using two thresholds, T_{low} and T_{high}. As such, the training stage is used in order to compute these thresholds. Here, four different incoming features are considered, one at a time: the MV variance distance, the MV phase variance, the number of DCT coefficients and the energy of DCT coefficients. The features are computed for each outgoing HEVC CU, and they are only computed if all incoming H.264/AVC macroblocks within that CU are encoded in *inter* mode. Also, all incoming features produce positive values equal to or higher than zero. The MV variance distance was explained in Sec. III-B, and the other features are explained next.

1) MV Phase Variance: Different from the MV variance distance, which measures the variance of the magnitude of the motion vectors, this feature is computed as the variance of the phases of all incoming motion vectors within a particular block. When multiple reference frames are used, the idea to use the phase, instead of the magnitude of the motion vectors, is necessary to overcome the limitation of scaling the motion vectors so that they point to the same reference frame. The phase is computed as:

$$phase = atan2\left(mv^k_{n \to n-\alpha}.y, mv^k_{n \to n-\alpha}.x\right) \qquad (3)$$

Note that the phase lies in the closed interval $[-\pi, \pi]$, according to Fig. 2. Also, of particular interest, the phase of the $(0, 0)$ motion vector is considered to be 0. Clearly, before computing the variance, the motion vectors are also propagated to each 4×4 block.

2) Number of DCT Coefficients: This simple incoming feature is the number of non-zero DCT coefficients encoded in the H.264/AVC bitstream for a particular block. The magnitude of these coefficients is not used, just whether or not a coefficient was transmitted. The idea of using the number of DCT coefficients is that, if there is a small number of DCT coefficients for a given block, it means that the prediction for that block was good and the residual is small and, therefore, a good prediction may be found using a larger block. On the other hand, if there is a larger number of DCT coefficients for a given block, then the prediction for that block is not good, and the block may need to be sub-partitioned to find a good prediction. Naturally, the information on the motion vectors may help on this decision, but this will be considered later in this paper.

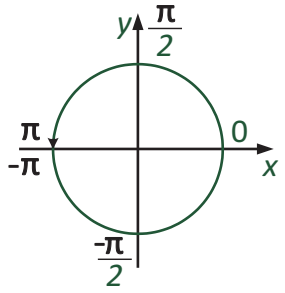

Fig. 2. Computing the motion vector phase. Additionally to the phases shown on the plot, the phase of the $(0, 0)$ motion vector is considered to be 0.

3) Energy of DCT Coefficients: This feature is computed as $E_C = \sum_i C_i^2$, where C_i denotes the DCT coefficients within a particular incoming block. Note that, since the minimum size for which the feature is computed is for a 32×32 block, the number of coefficients is the same whether the H.264/AVC 4×4 or 8×8 transform was used within the block. The idea of using the energy of DCT coefficients is the same as the number of DCT coefficients, but the energy gives a more complete information about the magnitude of the residual than just the number of coefficients.

Computing the thresholds

In order to adapt the thresholds to the content of the current sequence being transcoded, during the training stage, the transcoder computes the relevant incoming feature for each block using the H.264/AVC information, and stores these features in an array $\mathbf{F}^d = \left[f_0^d\ ,\ f_1^d\ ,\ \ldots\ ,\ f_{N-1}^d\right]$, with elements f_i^d, where d refers to the depth of the outgoing CU and i refers to the feature computed for a particular CU i, and N refers to the number of CUs. For the training frames, the computed features are not used to decide which partitions will be tested, instead, all HEVC modes are tested for these frames. After the decision for a given CU has been made, the transcoder stores the mode chosen in an array $\mathbf{C}^d = \left[c_0^d\ ,\ c_1^d\ ,\ \ldots\ ,\ c_{N-1}^d\right]$, where c_i^d refers to the class of the i-th CU. In order to simplify the large number of modes in the HEVC codec, the transcoder stores the chosen mode information in three classes: (i) if the CU was split; (ii) if the CU was encoded as SKIP or with a $2N \times 2N$ PU; and (iii) if the CU was encoded using any other mode.

After the training stage is done, the transcoder uses the two arrays, $\mathbf{F}^d$ and $\mathbf{C}^d$, to compute the thresholds T_{low}^d and T_{high}^d (where d corresponds to the depth of the outgoing HEVC CU)

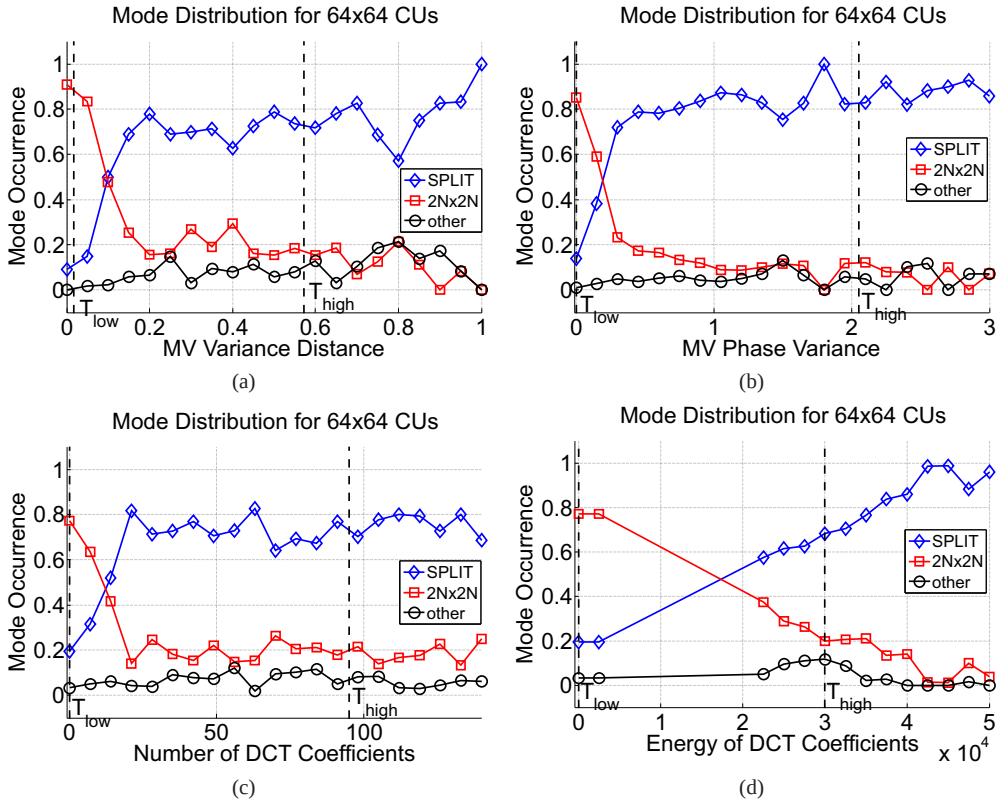

Fig. 3. Mode distribution and threshold computation for different features: (a) MV Variance Distance; (b) MV Phase Variance; (c) Number of DCT Coefficients; and (d) Energy of DCT Coefficients. The sequence being encoded is Basketball Drill, the data refers to the first 25 frames and the CU is 64×64 pixels.

using a percentile criterion. For each depth, the thresholds are computed as:

- T^d_{low} is chosen as the highest value for which 90% of the HEVC partitions with $f^d_i \leq T^d_{low}$ are encoded using either the SKIP or $2N \times 2N$ mode.
- T^d_{high} is chosen as the lowest value for which 90% of the HEVC partitions with $f^d_i > T^d_{high}$ are encoded using a higher depth (i.e., encoded in split mode).

Note that the 90% percentile is derived heuristically. This could be though as the transcoder complexity control. Fig. 3 illustrates the computation of the thresholds using the four features. In the figure, it can be seen that, for all features, the higher the value of the feature, the more likely it is that that CU is split in the HEVC. On the other hand, the lower the value of the feature, the more likely it is that the CU is encoded with a PU size of $2N \times 2N$. Regardless of the feature value, a small amount of CUs are encoded using the other modes (class (iii)).

Once the transcoding of the training sub-sequence is finished, the thresholds are computed. For the rest of the frames, the transcoder uses the computed thresholds to decide which HEVC partitions will be tested. For this transcoder, according to the incoming feature value f^d_i for the corresponding outgoing CU, the transcoder will apply the following rules, which are shown in Fig. 4.

- If $f^d_i \leq T^d_{low}$ (R_0, in the figure), then only the SKIP and the $2N \times 2N$ modes are tested, and the CU is *not* split.
- If $f^d_i > T^d_{high}$ (R_2, in the figure), then only the SKIP mode is tested for this depth, and the CU is split.
- Otherwise (i.e., if $T^d_{low} < f^d_i \leq T^d_{high}$, shown as region R_1, in the figure), then all modes are tested for this depth, and the CU is split.

Fig. 4. Threshold-based feature classification.

According to the incoming feature value, the transcoder tests the outgoing PUs according to these rules. If the rules state the CU should not be split, it chooses the best mode, among those tested, and proceeds to the next CU. Otherwise, if the CU is split, the algorithm is repeated for the four children CUs, unless the child CU is of size 16×16, in which case a simple mode mapping from the incoming H.264/AVC bitstream is used.

C. Content-Modeling Using Linear Discriminant Functions

In the previous sections the proposed transcoder uses only one feature at a time. However, an alternative approach would involve the use of many features in an attempt to better predict the outgoing HEVC partitioning. There are several methods in the literature that could be used to classify a given set of features [35], [36], [37]. Here, a simple method is used,

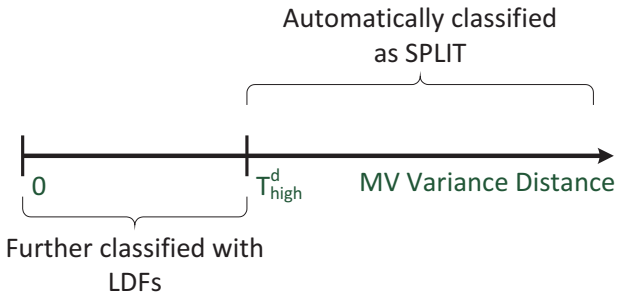

Fig. 5. Example of classification. All CUs with a MV Variance Distance higher than the threshold T^d_{high} are automatically classified as split. For the remaining CUs, linear discriminant functions (LDFs) are used to classify it between split or not split.

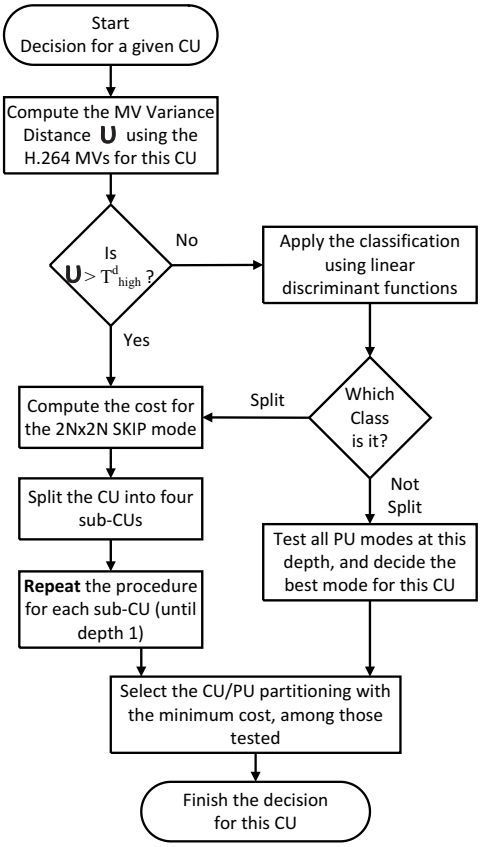

Fig. 6. Flowchart of CU mode decision using the linear discriminant function approach.

called Linear Discriminant Functions [38]. The main reason this method was chosen is that both the training and the classification themselves are very low complexity operations. Basically, a linear mapping is computed between the incoming features and the outgoing CU classes. The linear mapping is based on the well-known least-squares method using a non-iterative solution [38]. Thus, this method can be used inside the transcoder loop without hindering the transcoder complexity.

During the training stage, the transcoder computes all features for each block, storing it in an array $\mathbf{Fs}^d$ (where d refers to the depth of the CU). Also, for the training sub-sequence, all the outgoing HEVC modes are tested, and the decision for each CU is stored in an array $\mathbf{Cs}^d$. The first modification, compared to the dynamic thresholding solution, is that the transcoder stores the chosen mode information in only two classes: (i) if the CU was split; and (ii) if the CU was not split. Therefore, the transcoder is attempting to classify only whether the CU was split or not.

It was noted that one of the incoming features, the MV variance distance, has a very high correlation with one of the two aforementioned classes. A block with a high value for the MV variance distance is most likely to be split. For this reason, the incoming blocks with a MV variance distance v higher than a threshold (T^d_{high}, computed as the 90-th percentile, as explained in Sec. IV-B) are removed from the set on the assumption that they shall be split. For the rest of the incoming blocks (i.e., for which $v \leq T^d_{high}$), the classification is applied using the linear discriminant function solution. This procedure is illustrated in Fig. 5.

In order to use the linear discriminant functions, seven incoming features are considered for the 64×64 and 32×32 CU sizes:

1) The total number of H.264/AVC partitions in the incoming CU. This is the number of different inter-prediction blocks in the H.264/AVC for the region defined by the current CU.
2) The MV variance distance, as introduced in Sec. III-B.
3) The variance of the *x* component for the motion vectors within the CU.
4) The variance of the *y* component for the motion vectors within the CU.
5) The MV phase variance, as introduced in Sec IV-B.
6) The number of DCT coefficients, as introduced in Sec. IV-B.
7) The average energy of the DCT coefficients. This is energy of the DCT coefficients (as introduced in Sec. IV-B) divided by the number of DCT coefficients. If there are no non-zero DCT coefficients within the CU, then it is considered as zero.

After it finishes transcoding the training sub-sequence, and after the CUs with MV variance distance $v > T^d_{high}$ are removed from the set, the remaining elements in the set are used to compute the optimal weights for the linear discriminant functions. Denote the sequence of feature vectors in this set that belongs to class i by $\mathbf{X}_i = [\mathbf{x}_{i,1} \;\; \mathbf{x}_{i,2} \;\; \ldots \;\; \mathbf{x}_{i,N_i}]^T$, where $\mathbf{X}_i$ is a $N_i \times M$ matrix, M is the dimensionality of the feature vector and N_i is the total number of feature vectors in class i. Concatenating the $\mathbf{X}_i$ matrices for both classes (i.e., split and not split) results in $\mathbf{X} = [\mathbf{X}_0 \;\; \mathbf{X}_1]^T$, where $\mathbf{X}_0$ represents the feature vectors for class 0 (split) and $\mathbf{X}_1$ represents the feature vectors for class 1 (not split). At the same time, let $\mathbf{y}_i$ be the ideal output vector for class i, which is a column vector comprised of zeros and ones, such as: $\mathbf{y}_0 = [\mathbf{1}_{N_0} \;, \; \mathbf{0}_{N_1}]^T$ and $\mathbf{y}_1 = [\mathbf{0}_{N_0} \;, \; \mathbf{1}_{N_1}]^T$. Again, $\mathbf{y} = [\mathbf{y}_0 \;\; \mathbf{y}_1]^T$.

The training procedure consists of computing the optimum weight vector $\mathbf{w}_i^{opt}$ that minimises the distance between $\mathbf{y}_i$ and a linear combination of the training feature vectors $\mathbf{X}\mathbf{w}_i$ such that

$$\mathbf{w}_i^{opt} = \underset{\mathbf{w}_i}{\arg}\ min\ \ \|\mathbf{X}\mathbf{w}_i - \mathbf{y}_i\|_p \tag{4}$$

Eq. 4 indicates that the optimal weight vector $\mathbf{w}_i^{opt}$ could be obtained by minimising the L^p-norm of the error vector $\mathbf{e}_i = \mathbf{X}\mathbf{w}_i - \mathbf{y}_i$. Minimising this function using the L^2-norm leads to:

$$\mathbf{w}_i^{opt} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}_i \tag{5}$$

which can be simply solved. Once the optimal weights are computed, using the classifier is a simple operation. Let X_C be the feature vector that needs to be classified as one of the two classes. This can be done by evaluating the output of this feature vector against all 2 models, computing a set of *scores* s_i, such as:

$$\mathbf{s}_i = \mathbf{X}_C\mathbf{w}_i^{opt} \tag{6}$$

Then, the class of the sequence X_C is determined by choosing the class with the highest score:

$$c = \underset{i}{\arg}\ max\ \ (\mathbf{s}_i) \tag{7}$$

Then, the following algorithm is applied to decide which partitions are tested for each CU. When deciding which modes will be tested for a given CU, the transcoder first computes the MV variance distance υ. If $\upsilon > T^d_{high}$, then this partition is split into four sub-CUs and the algorithm repeats itself for the CUs at the next depth (only the SKIP mode is tested at the current depth, before splitting). Otherwise, the transcoder computes the remaining features and applies the classification again for the four sub-CUs. If the outcome is split (i.e., if the score for the split class is higher than the score for the not split class), then this partition is split and the algorithm repeats itself for the CUs at the next depth (again, only the SKIP mode is tested at this depth). Otherwise, if the outcome is not to split, then all modes at this depth are tested and the partition is *not* split. Finally, the transcoder decides for the best mode among those tested, and proceeds to the next CU. This algorithm is shown in Fig. 6.

This algorithm is applied for the 64×64 and 32×32 CUs. For the 16×16 and 8×8 CUs, the H.264/AVC macroblock and sub-macroblock types are used, as explained in Sec. IV-A. Also, if there is an intra block within the CU, then the algorithm is not used and all partitions are tested for that CU, and the CU is split.

V. Experimental Results

In this section, we present two sets of experiments in order to evaluate the proposed methods. The first set is designed to evaluate the training stage, while the second set evaluates the proposed transcoder in more practical scenarios.

A. Evaluating the Training Stage

In the following experiments, first the original sequence is encoded using H.264/AVC at High Profile, and using an *IPP* configuration with 1 reference frame. The reference software JM 14.2 [39] is used.

Afterwards, the H.264/AVC bitstream is transcoded to the HEVC using one of the transcoding options. For HEVC, the reference software for the $HM4.0rc1$ encoder is used with the default settings [33], with a 64×64 largest CU (LCU) size and LCUs are encoded in raster scan order, using the same reference frame structure as the H.264/AVC (called the low-delay reference frame structure in HEVC, with one reference frame). For all transcoding options, fast motion estimation and fast mode decision are used for HEVC , as defined in the $HM4.0rc1$ reference software.

The QPs used to encode the H.264/AVC bitstream are $\{37, 32, 27, 22\}$, and the same QPs are used for HEVC. Four sequences are used to evaluate the proposed transcoders: Basketball Drill 832×480 50 Hz, BQMall 832×480 60 Hz, Party Scene 832×480 50 Hz and Race Horses 832×480 30 Hz. All of these sequences are part of the HEVC testing dataset.

Five options of the proposed transcoder are evaluated (i.e., four of the dynamic thresholding solution and one using the linear discriminant functions solution). Three other options are used as benchmarks. The full list of tested transcoders, as referred to in this section, are as follows:

1) RT-EPZS: this corresponds to the trivial transcoder, decoding the entire sequence and re-encoding using the HEVC standard fast motion estimation and mode decision algorithms. This option is used as anchor both for BD-rate calculation and speed-up results.
2) RT-MVR: the transcoder based on MV Reuse: this corresponds to the transcoder shown in Sec. III-B [3].
3) RT-MVVD: The transcoder based on the MV Variance Distance as seen in Sec. III-B, with parameters $T_{low} = 1$, $T_{high} = 100$ [3].
4) PTDT-MVVD: The dynamic thresholding transcoder (Sec. IV-B) using MV Variance Distance.
5) PTDT-MVPV: The dynamic thresholding transcoder (Sec. IV-B) using MV Phase Variance.
6) PTDT-NDCT: The dynamic thresholding transcoder (Sec. IV-B) using the number of DCT coefficients.
7) PTDT-EDCT: The dynamic thresholding transcoder (Sec. IV-B) using the energy of DCT coefficients.
8) PTCM-LDF: The content modeling transcoder (Sec. IV-C) using Linear Discriminant Functions.

The experiments are designed to provide answers to the following questions: (i) how many frames are needed during the training process to build an efficient model; and (ii) for how long will the training model remain valid (i.e., for how many frames can the model be successfully applied). Thus, three different training lengths were tested: 10, 25 and 50 frames, and, in order to investigate the impact of using the same parameters for a longer period, three different lengths of the sequences were used: 2.5, 5 and 10 seconds. The complete results are shown in Tables III, IV and V, both in terms of BD-

TABLE III
TRANSCODER RESULTS USING 10 FRAMES FOR TRAINING, USING RT-EPZS AS ANCHOR. BD-RATE FIGURES ARE SHOWN IN PERCENTAGE.

		Basketball Drill		BQ Mall		PartyScene		RaceHorses	
	Transcoder	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up
Length = 2.5 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.29	1.09	2.27	1.08	1.39	1.10	1.94	1.10
	RT-MVVD	5.34	2.15	7.80	1.95	15.1	2.19	3.99	1.65
	PTDT-MVVD	3.61	1.77	4.36	1.85	2.69	1.79	2.77	1.50
	PTDT-MVPV	7.11	1.73	6.61	1.72	3.21	1.72	2.79	1.45
	PTDT-NDCT	5.13	1.89	6.85	2.03	2.92	1.88	4.08	1.67
	PTDT-EDCT	5.57	2.11	7.91	2.28	3.19	2.08	4.54	1.76
	PTCM-LDF	4.93	2.00	5.98	2.12	3.23	1.99	5.42	1.71
Length = 5 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.24	1.09	2.15	1.07	1.35	1.10	1.95	1.10
	RT-MVVD	5.80	2.14	7.10	2.01	14.2	2.16	3.70	1.76
	PTDT-MVVD	4.04	1.82	4.00	1.94	2.97	1.86	2.79	1.58
	PTDT-MVPV	7.78	1.78	5.88	1.79	3.53	1.79	2.80	1.52
	PTDT-NDCT	5.31	1.97	6.55	2.18	3.36	1.96	3.84	1.79
	PTDT-EDCT	5.77	2.20	7.69	2.48	3.63	2.18	4.31	1.89
	PTCM-LDF	5.43	2.08	5.58	2.26	3.59	2.09	4.94	1.76
Length = 10 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.36	1.09	1.89	1.07	1.09	1.10	2.05	1.10
	RT-MVVD	6.15	2.13	7.17	1.96	16.2	2.33	4.37	1.69
	PTDT-MVVD	4.43	1.86	3.92	1.96	2.68	2.00	3.35	1.63
	PTDT-MVPV	8.26	1.81	5.33	1.80	4.15	1.96	3.34	1.56
	PTDT-NDCT	5.66	2.00	6.42	2.22	3.32	2.14	4.71	1.83
	PTDT-EDCT	6.09	2.25	7.58	2.58	3.58	2.40	5.33	1.94
	PTCM-LDF	5.84	2.12	5.93	2.34	3.34	2.27	6.59	1.87

TABLE IV
TRANSCODER RESULTS USING 25 FRAMES FOR TRAINING, USING RT-EPZS AS ANCHOR. BD-RATE FIGURES ARE SHOWN IN PERCENTAGE.

		Basketball Drill		BQ Mall		PartyScene		RaceHorses	
	Transcoder	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up
Length = 2.5 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.29	1.09	2.27	1.09	1.39	1.09	1.94	1.08
	RT-MVVD	5.34	2.16	7.80	1.96	15.1	2.19	3.99	1.66
	PTDT-MVVD	2.96	1.57	4.02	1.63	2.09	1.55	2.05	1.31
	PTDT-MVPV	3.38	1.52	4.48	1.55	2.14	1.52	2.04	1.29
	PTDT-NDCT	4.42	1.67	6.35	1.82	2.36	1.64	2.95	1.42
	PTDT-EDCT	4.68	1.79	7.00	1.98	2.63	1.78	3.36	1.47
	PTCM-LDF	4.42	1.74	5.63	1.87	3.07	1.72	4.71	1.44
Length = 5 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.24	1.09	2.15	1.07	1.35	1.11	1.95	1.10
	RT-MVVD	5.80	2.18	7.10	2.05	14.2	2.21	3.70	1.79
	PTDT-MVVD	3.64	1.72	3.82	1.79	2.55	1.73	2.45	1.48
	PTDT-MVPV	4.17	1.65	4.12	1.69	2.65	1.68	2.42	1.44
	PTDT-NDCT	5.06	1.84	6.28	2.08	2.96	1.84	3.29	1.64
	PTDT-EDCT	5.28	2.01	7.09	2.31	3.29	2.04	3.78	1.73
	PTCM-LDF	5.14	1.95	5.69	2.13	3.62	1.96	5.00	1.68
Length = 10 s	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.36	1.09	1.89	1.07	1.09	1.10	2.05	1.10
	RT-MVVD	6.15	2.18	7.17	2.01	16.2	2.38	4.37	1.72
	PTDT-MVVD	4.05	1.80	3.73	1.84	2.35	1.87	3.19	1.56
	PTDT-MVPV	4.52	1.72	4.09	1.75	2.75	1.84	3.08	1.50
	PTDT-NDCT	5.54	1.94	6.33	2.19	2.96	2.04	4.40	1.74
	PTDT-EDCT	5.77	2.13	7.26	2.48	3.39	2.31	5.04	1.85
	PTCM-LDF	5.65	2.18	6.41	2.27	3.45	2.19	7.78	1.78

rate [40] and transcoding speed-up. For all cases, RT-EPZS is used as anchor for both BD-bitrate and speed-up figures.

Note that the speed-up figures for RT-MVR and RT-MVVD for this experiment are lower than those shown in the original paper [3]. The reason is that in this experiment only one reference frame was used, instead of four.

It can be seen that reference transcoder based on the MV Reuse (RT-MVR) shows a loss of up to 2.29% in terms of BD-rate (for Basketball Drill sequence encoding 2.5 seconds, shown in Tables III, IV and V), and 1.9% on average, but the speed-up is only of 1.09, on average. These speed-up figures implies that relying on the MV reuse alone is not sufficient for an efficient transcoder, as the gains in terms of complexity are rather low. On the other hand, the good RD performance implies that the H.264/AVC motion vectors are highly correlated to the HEVC motion vectors. The RT-MVVD shows good speed-up figures (up to 2.38, for PartyScene sequence encoding 10 seconds of the sequence, shown in Table IV, and 2.04 on average), but the RD loss is significantly higher, especially for the PartyScene sequence (up to 16.2%, shown in Table IV) and BQMall (up to 7.80%, shown in Table V) sequences.

Using the dynamic thresholding approach, the RD performance loss is significantly lower, regardless of the features used. In the worst case, the BD-rate loss is 8.26%, when using the MV phase variance as feature (PTDT-MVPV) (for Basketball Drill sequence, using 10 frames for training and transcoding 10 s, seen in Table III). Among the features, the MV variance distance (PTDT-MVVD) presented the lowest BD-rate loss for the majority of the cases, with a BD-rate loss

TABLE V
TRANSCODER RESULTS USING 50 FRAMES FOR TRAINING, USING RT-EPZS AS ANCHOR. BD-RATE FIGURES ARE SHOWN IN PERCENTAGE.

		Basketball Drill		BQ Mall		PartyScene		RaceHorses	
	Transcoder	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up	BD-rate	Speed-up
Length = 2.5 *s*	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.29	1.09	2.27	1.09	1.39	1.10	1.94	1.10
	RT-MVVD	5.34	2.15	7.80	1.96	15.1	2.19	3.99	1.66
	PTDT-MVVD	2.24	1.40	2.93	1.45	1.47	1.36	1.09	1.14
	PTDT-MVPV	2.36	1.36	3.10	1.40	1.43	1.33	1.03	1.13
	PTDT-NDCT	3.14	1.45	4.41	1.51	1.66	1.41	1.42	1.17
	PTDT-EDCT	3.40	1.52	5.31	1.68	1.80	1.46	1.61	1.19
	PTCM-LDF	3.24	1.49	5.01	1.61	2.52	1.47	2.59	1.19
Length = 5 *s*	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.24	1.09	2.15	1.07	1.35	1.11	1.95	1.10
	RT-MVVD	5.80	2.13	7.10	2.01	14.2	2.17	3.70	1.76
	PTDT-MVVD	3.24	1.60	3.12	1.65	2.18	1.56	1.97	1.35
	PTDT-MVPV	3.52	1.54	3.18	1.58	2.19	1.66	1.91	1.32
	PTDT-NDCT	4.44	1.69	4.90	1.79	2.59	1.74	2.46	1.44
	PTDT-EDCT	4.70	1.79	6.09	2.05	2.78	1.77	2.91	1.50
	PTCM-LDF	4.63	1.76	5.89	2.01	3.62	2.17	4.16	1.48
Length = 10 *s*	RT-EPZS	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
	RT-MVR	2.36	1.09	1.89	1.07	1.09	1.10	2.05	1.10
	RT-MVVD	6.15	2.13	7.17	1.97	16.1	2.33	4.37	1.48
	PTDT-MVVD	3.90	1.72	1.89	1.76	2.04	1.75	2.86	1.43
	PTDT-MVPV	4.25	1.65	3.35	1.67	2.11	1.68	2.80	1.61
	PTDT-NDCT	5.16	1.85	5.40	1.95	2.68	1.90	3.81	1.70
	PTDT-EDCT	5.45	1.99	6.75	2.32	3.00	2.03	4.45	1.66
	PTCM-LDF	5.51	1.94	7.35	2.14	3.69	2.06	7.59	1.68

ranging from 1.09% (for RaceHorses sequence, with 50 frames for training and encoding 2.5 *s*, seen in Table V) to 4.43% (for Basketball Drill sequence, with 10 frames for training and encoding 10 *s*, seen in Table III). For most of the tests, the MV phase variance (PTDT-MVPV) also presents a very low loss, close to PTDT-MVVD, with the notable exception of Basketball Drill and BQMall sequences using 10 frames for training (seen in Table III), where the loss is among the highest. However, the speed-up figures for both PTDT-MVVD and PTDT-MVPV are the lowest for the dynamic thresholding approach. The fastest option for the majority of the cases is using the energy of the DCT coefficients (PTDT-EDCT), at the expense of a worse RD loss. Using the Linear Discriminant Functions (PTCM-LDF), the speed-up figures are closer to PTDT-EDCT, while the rate distortion performance loss is lower.

Analysing the tables, it can be seen that, as expected, the higher the number of frames used for training and the shorter the sequence being transcoded, the better the rate distortion performance (although there are a few exceptions, notably when using the Linear Discriminant Functions (PTCM-LDF) encoding 5 and 10 seconds). However, apart from the MV Phase Variance feature (PTDT-MVPV), this gain in performance is rather low. Using 25 frames for training results in -0.9%, -0.57% and -0.40% average BD-rate gains when encoding 2.5, 5 and 10 seconds, respectively, compared to using only 10 frames for training, on average among all features and sequences. Using 50 frames for training results on -1.15%, -0.59% and -0.38% average BD-rate gains when encoding 2.5, 5 and 10 seconds, respectively, compared to using 25 frames for training. This happens for two reasons: generally, a longer training yields a better model, and thus the loss in the transcoding phase is lower; and because the best rate-distortion performance is obtained in the training part, when full encoding is being performed. Note that the largest difference occurs when the ratio of the number of frames used for training compared to the length of the sequence being encoded is the highest (50 frames used for training and encoding only 2.5 seconds).

TABLE VI
AVERAGE RESULTS FOR PTCM-LDF, USING RT-EPZS AS ANCHOR.

		BD-Rate			Speed-Up		
		Number of Frames used for Training					
		10	25	50	10	25	50
Length	2.5*s*	4.89	4.45	3.34	1.95	1.69	1.44
	5*s*	4.88	4.86	4.57	2.04	1.93	1.85
	10*s*	5.42	5.82	6.03	2.15	2.10	1.95

On the other hand, it can also be seen that the speed-up gain when encoding a larger sequence is rather small. Encoding 5 seconds results in a speed-up gain of 1.05, 1.13 and 1.20, when using 10, 25 and 50 frames for training, respectively, on average among all features and sequences, compared to encoding only 2.5 seconds. Encoding 10 seconds results in a speed-up gain of 1.04, 1.07 and 1.09, when using 10, 25 and 50 frames for training, respectively, on average among all features and sequences, compared to encoding 5 seconds. Notice that the largest difference occurs when using 50 frames for training and encoding shorter sequences (2.5 and 5 seconds), and even in this case the gain is only 20%.

Table VI compares the number of frames used for training and the length of the sequence being encoded for the specific case of the Linear Discriminant Functions (PTCM-LDF). Although there are a few outliers, the behaviour discussed in the previous paragraphs can be observed: the longer the training sequence and the shorter the sequence, the better the rate distortion performance (the lowest average loss, 3.34%, occurs for training with 50 frames and encoding 2.5 seconds of the sequence). At the same time, the shorter the training sequence and the longer the sequence, the fastest the transcoder (the fastest option, at 2.15, occurs for training with 10 sequences and encoding 10 seconds). However, for both cases, this difference is rather small.

Among the incoming H.264/AVC features that have been tested, the best compromise between complexity and rate-distortion performance is obtained when a combination of all features, combined by a linear classifier (in this case, using linear discriminant functions, PTCM-LDF), is used. The experiments were carried out to find out the number of frames needed for training and also if the model built is robust enough to be used for a long period. From the results of these experiments, it was observed that the gain in rate-distortion performance of using more frames to generate the model is rather low. At the same time, while the rate-distortion performance losses of applying this model for a long period are small, the gain in complexity is also rather low. This leads to the conclusion that a better transcoding option would involve using less frames for training and encoding a shorter sequence, repeating the training more often, in order to keep track of the properties of the sequence being transcoded. This way, the speed-up could be kept at roughly 2 (for a low-delay configuration using only one reference frame - larger speed-up figures are expected if more reference frames are used), while the benefits of a small rate-distortion loss would be retained. In addition, the transcoder would be robust if the sequence properties are changed. The exact parameters could be changed according to the application, or it could be done adaptively, using algorithms to detect scene changes [41], [42] to decide when to build a new model, or developing an algorithm specifically to decide when a new model should be built.

B. Experiments in more practical scenarios

In this set of experiments, the proposed LDF method is evaluated in a more practical scenario. Again, the original sequence is encoded using H.264/AVC at High Profile, however, three reference frame structures are used: (i) an *IPP* structure with 1 reference frame, denoted as $IPP1$; (ii) an *IPP* structure with 4 reference frames, denoted as $IPP4$; and (iii) an *IBBP* structure with 1 reference frame, denoted as $IBBP$. The reference software JM 14.2 [39] is used.

Afterwards, the H.264/AVC bitstream is transcoded to the HEVC using three transcoding options: (i) the trivial transcoder, RT-EZPS; (ii) the transcoder based on the MV Variance Distance as seen in Sec. III-B, with parameters $T_{low} = 1$, $T_{high} = 100$ [3], RT-MVVD; and (iii) the proposed method with the linear discriminant functions, using the first 10 frames for training, PT-LDF. For these experiments, the reference software for the $HM9.1rc1$ encoder is used with the default settings [43].

The QPs used to encode the H.264/AVC bitstream are $\{37, 32, 27, 22\}$, and the same QPs are used for HEVC. Two sequences are used: Kimono1 1920×1080 24 Hz and ParkScene 1920×1080 24 Hz. Both sequences are part of the HEVC testing dataset. The results are presented in Table VII and Fig. 7.

The first thing that can be seen from these results is that RT-MVVD does perform really well in some cases, such as for Kimono1 using $IBBP$ structure, where it offers a loss of only 1.57% with a speed-up factor of 1.94. However, the range of the RD loss quite large (between 1.57% and 9.83% of average bitrate, being as high as 14.4% when only high bitrates are considered, as seen in Table VII). This is expected, since this method uses fixed thresholds that may not be suitable for every sequence.

TABLE VII
TRANSCODER RESULTS IN PRACTICAL SCENARIOS.

	Sequence	Method	Speed-Up	BD-Rate		
				Average	Low	High
IBBP	Kimono1	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	1.94	1.57	1.14	1.97
		PT-LDF	2.53	2.95	2.40	3.30
	ParkScene	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	2.27	4.31	2.67	5.60
		PT-LDF	2.91	4.22	3.14	4.49
IPP4	Kimono1	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	2.16	2.96	1.95	4.09
		PT-LDF	2.69	3.75	3.25	3.92
	ParkScene	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	2.91	9.83	5.07	14.4
		PT-LDF	3.06	4.42	4.83	3.74
IPP1	Kimono1	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	2.04	2.79	2.02	3.55
		PT-LDF	2.43	3.59	3.36	3.43
	ParkScene	RT-EPZS	1.00	0.00	0.00	0.00
		RT-MVVD	2.75	8.09	4.42	11.5
		PT-LDF	2.72	4.26	5.02	3.49

On the other hand, the proposed method, PT-LDF, is much more reliable, with RD loss in the range 2.95% to 4.42%. Also, even performing the training for the first 10 frames, it is also faster than RT-MVVD for all cases tested except one (ParkScene with $IPP1$ structure, where both methods yield the same speed-up factor. In addition, the PT-LDF offers the same level of performance for both low and high bitrates.

Finally, the results show that the method performs well for all reference frame structures tested, and that it can be scaled for HD content.

VI. Conclusions and Future Work

In this paper, transcoding solutions based on dynamic thresholding and content modeling are proposed, in which parts of the sequence are used for training. During the training stage, full re-encoding is applied, and the transcoder then uses this information, along with the information from the H.264/AVC bitstream, to generate a content-specific model to map the H.264/AVC partitions into HEVC CUs. Afterwards, the transcoder applies this model to decide which partitions are tested for the rest of the sequence. Experiments have shown that the performance of this transcoder is much better than similar transcoding options based on fixed thresholds, yielding a much lower rate-distortion loss at a competitive complexity performance. In particular, tests have shown that the proposed approach using linear discriminant functions yields good results even in difficult enviroments, such as using multiple reference frames or B-frames.

Among the transcoding options proposed, the dynamic thresholding using the MV variance distance as the metric resulted in the lowest RD loss, while the dynamic thresholding using the energy of the DCT coefficients as metric was the fastest, for most of the cases. Using linear discriminant functions to combine several features presented the best compromise, with a speed-up close to the dynamic thresholding

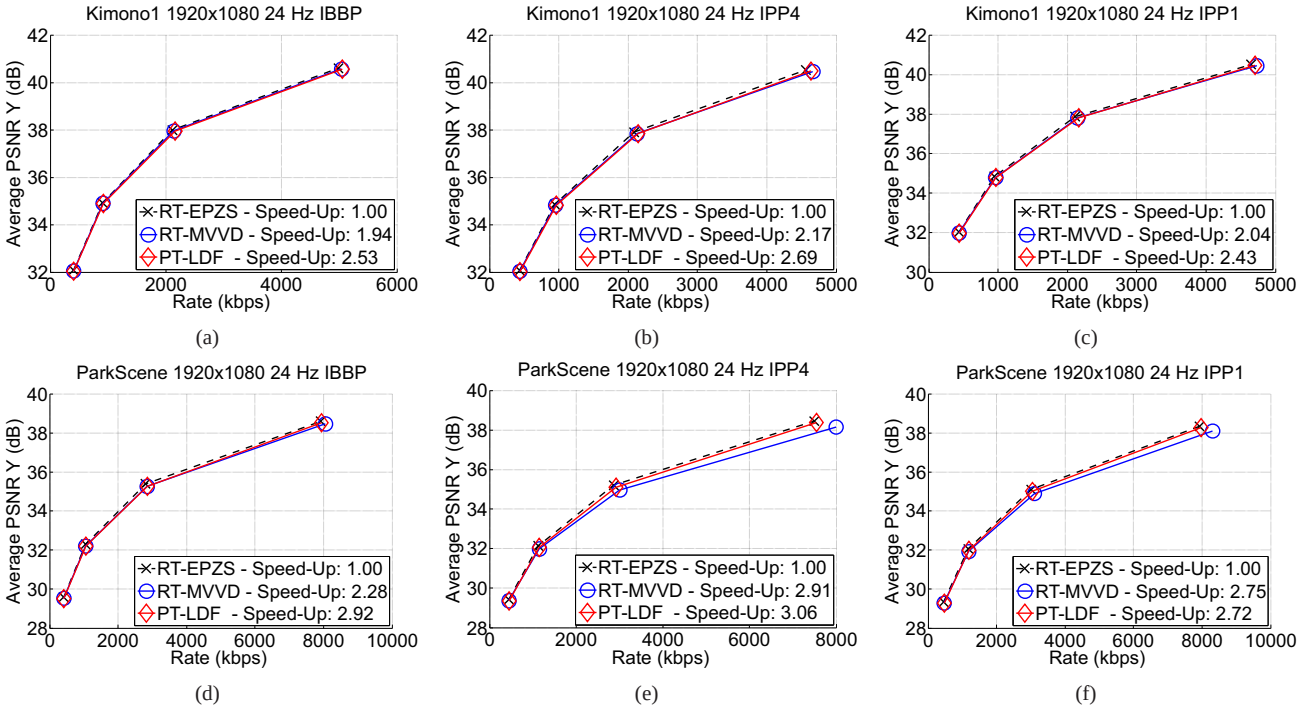

Fig. 7. RD Results for Kimono1 sequence using: (a) $IBBP$; (b) $IPP4$ and (c) $IPP1$ reference frame structures, and for ParkScene sequence using: (d) $IBBP$; (e) $IPP4$ and (f) $IPP1$ reference frame structures.

using the energy of the DCT coefficients but at a lower RD loss, compared to the trivial transcoder.

The work presented in this paper also opens up several interesting directions for future work. For instance, other machine learning techniques could be used, instead of the simple linear discriminant functions, and different set of features could be explored to improve the transcoder performance. Also, new options to decide when to re-build the transcoding model (i.e., when to perform the training stage again) could be explored, either based on scene change detection algorithms or a kind of "internal control" to detect when the model is no longer optimal and trigger a new training.

REFERENCES

[1] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, "High efficiency video coding (hevc) text specification draft 9, document jctvc-k1003," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Tech. Rep., 2012.

[2] ITU-T, "ITU-T Recommendation H.264, Advanced video coding for generic audiovisual services," ITU-T, Tech. Rep., May 2003.

[3] E. Peixoto and E. Izquierdo, "A complexity-scalable transcoder from H.264/AVC to the new HEVC codec," in *IEEE International Conference on Image Processing (ICIP 2012)*, September 2012, pp. 737–740.

[4] 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. PP, no. 99, pp. 1–1, 2013.

[5] 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, October 2005.

[6] 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, March 2003.

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

[8] P. Kunzelmann and H. Kalva, "Reduced complexity H.264 to MPEG-2 transcoder," in *International Conference on Consumer Electronics (ICCE 2007)*, January 2007, pp. 1–2.

[9] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, pp. 101–110, June 2000.

[10] P. Assuncao and M. Ghanbari, "Transcoding of MPEG-2 video in the frequency domain," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1997)*, vol. 4, April 1997, pp. 2633 –2636 vol.4.

[11] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 191–199, April 1996.

[12] T.-K. Lee, C.-H. Fu, Y.-L. Chan, and W.-C. Siu, "A new motion vector composition algorithm for fast-forward video playback in H.264," in *IEEE International Symposium on Circuits and Systems (ISCAS 2010)*, June 2010, pp. 3649–3652.

[13] I.-H. Shin, Y.-L. Lee, and H. W. Park, "Motion estimation for frame-rate reduction in H.264 transcoding," in *IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, May 2004, pp. 63–67.

[14] H. Shu and L.-P. Chau, "The realization of arbitrary downsizing video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 540–546, April 2006.

[15] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, November 2002.

[16] J. Cock, S. Notebaert, P. Lambert, and R. V. de Walle, "Architectures for fast transcoding of H.264/AVC to quality-scalable SVC streams," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1209–1223, November 2009.

[17] D. Xu and P. Nasiopoulos, "Logo insertion transcoding for H.264/AVC compressed video," in *IEEE International Conference on Image Processing (ICIP 2009)*, November 2009, pp. 3693–3696.

[18] T. Shanableh, "Matrix encoding for data hiding using multilayer video coding and transcoding solutions," *Signal Processing: Image Communication*, vol. 27, no. 9, pp. 1025–1034, October 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596512001129

[19] T. Shanableh, T. M., and F. Ishtiaq, "Error resiliency transcoding and decoding solutions using distributed video coding techniques," *Signal Processing: Image Communication*, vol. 23, no. 8, pp. 610–623, 2008.

[20] G. J. Sullivan, J. Ohm, H. Woo-Jin, 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.

[21] M. Pantoja, H. Kalva, and J.-B. Lee, "P-frame transcoding in VC-1 to H.264 transcoders," in *IEEE International Conference on Image Processing (ICIP 2007)*, vol. 5, October 2007, pp. V –297 –V –300.

[22] Y.-K. Lee, S.-S. Lee, and Y.-L. Lee, "MPEG-4 to H.264 transcoding using macroblock statistics," in *IEEE International Conference on Multimedia and Expo (ICME 2006)*, July 2006, pp. 57–60.

[23] J.-H. Hur and Y.-L. Lee, "H.264 to MPEG-4 transcoding using block type information," in *TENCON 2005 IEEE Region 10*, November 2005, pp. 1–6.

[24] G. Fernandez-Escribano, P. Cuenca, L. O. Barbosa, and H. Kalva, "Very low complexity MPEG-2 to H.264 transcoding using machine learning," in *ACM International Conference on Multimedia (ACM Multimedia 2006)*. ACM, 2006, pp. 931–940.

[25] G. Fernandez-Escribano, H. Kalva, J. 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.

[26] C. Holder, T. Pin, and H. Kalva, "Improved machine learning techniques for low complexity MPEG-2 to H.264 transcoding using optimized codecs," in *International Conference on Consumer Electronics (ICCE 2009)*, January 2009, pp. 1–2.

[27] J. Martinez, G. Fernandez-Escribano, H. Kalva, W. Fernando, and P. Cuenca, "Wyner-Ziv to H.264 video transcoder for low cost video encoding," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1453–1461, August 2009.

[28] D. Zhang, B. Li, J. Xu, and H. Li, "Fast transcoding from H.264 AVC to high efficiency video coding," in *IEEE International Conference on Multimedia and Expo (ICME 2012)*, July 2012, pp. 651–656.

[29] H. Shen, X. Sun, and F. Wu, "Fast H.264/MPEG-4 AVC transcoding using power-spectrum based rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 6, pp. 746–755, June 2008.

[30] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 88–98, February 1998.

[31] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion estimation for high performance transcoding," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 649–658, August 1998.

[32] I.-K. Kim, W.-J. Han, J. H. Park, and X. Z. Zheng, "CE2: Test results of asymmetric motion partition (AMP)," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Tech. Rep. JCTVC-F379, July 2011.

[33] K. McCann, B. Bross, S.-I. Sekiguchi, and W.-J. Han, "HM4: High efficiency video coding (HEVC) test model 4 encoder description," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Tech. Rep. Doc. JCTVC-F802, July 2011.

[34] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Visual Communications and Image Processing (VCIP 2002)*. SPIE, 2002, pp. 1069–1079.

[35] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[36] C. M. Bishop, *Pattern Recognition and Machine Learning*, 2nd ed. Springer, 2007.

[37] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.

[38] T. Shanableh and K. Assaleh, "Feature modeling using polynomial classifiers and stepwise regression," *Neurocomputing*, vol. 73, no. 10-12, pp. 1752–1759, June 2010.

[39] ITU-T, "Joint model (JM), H.264/AVC reference software," Artech House Inc., Tech. Rep. JM 14.2 KTA 1.0, 2008.

[40] G. Bjontegaard, "Improvements of the BD-PSNR model," ITU-T Telecommunications Standardization Sector, Video Coding Experts Group (VCEG), Tech. Rep. VCEG-AI11, July 2008.

[41] K.-D. Seo, S. J. Park, and S.-H. Jung, "Wipe scene-change detector based on visual rhythm spectrum," in *International Conference on Consumer Electronics (ICCE 2009)*, January 2009, pp. 1–2.

[42] D. Lelescu and D. Schonfeld, "Statistical sequential analysis for real-time video scene change detection on compressed multimedia bitstream," *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 106–117, March 2003.

[43] K. McCann, B. Bross, W.-J. Han, I. K. Kim, K. Sugimoto, and G. J. Sullivan, "High efficiency video coding (HEVC) test model 9 (HM 9) Encoder Description," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Tech. Rep. Doc. JCTVC-F802, July 2012.