This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

Authors' accepted manuscript. Article published in *IEEE Transactions on Circuits and Systems for Video Technology* 28(12), Dec. 2018

Efficient H.264-to-HEVC Transcoding Based on Motion Propagation and Post-Order Traversal of Coding Tree Units

Jean-François Franche, *Member, IEEE,* and Stéphane Coulombe, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a fast H.264-to-HEVC transcoder composed of a motion propagation algorithm and a fast mode decision framework. The motion propagation algorithm creates a motion vector candidate list at the coding tree unit (CTU) level and, thereafter, selects the best candidate at the prediction unit level. This method eliminates computational redundancy by pre-computing the prediction error of each candidate at the CTU level and reusing the information for various partition sizes. The fast mode decision framework is based on a post-order traversal of the CTU, and includes several mode reduction techniques. In particular, the framework permits the early termination of the rate distortion cost computation, a highly complex task, when a mode is unpromising. Moreover, a novel method exploits the data created by the motion propagation algorithm to determine whether a coding unit (CU) must be split. This allows the pruning of unpromising sub-partitions. Compared to a cascaded pixel-domain transcoding approach, the experimental results show that the proposed solution using one reference frame is on average 8.5 times faster, for an average BD-Rate of 2.63%. For a configuration with 4 reference frames, the average speed-up is 11.77x and the average BD-Rate is 3.82%.**

***Index Terms*—Fast mode decision, H.264, HEVC, Motion estimation, Video transcoding**

I. Introduction

HIGH Efficiency Video Coding (HEVC) is the most recent video coding standard jointly developed by ITU-T and the ISO/IEC Moving Picture Experts Group (MPEG) [1]. Like prior video coding standards, HEVC adopts the hybrid video coding scheme. Existing coding tools have been improved, at the cost of greater computational complexity, and new tools have been introduced, notably for better parallel processing support. Coding efficiency is significantly improved, in particular by replacing the macroblock (MB) with a more flexible structure, called the coding tree unit (CTU). Compared to its predecessor, H.264 [2], HEVC halves the bit rate for similar video quality.

To benefit from this increased coding efficiency, and to allow interoperability between H.264 and HEVC systems, several H.264 video sequences must be transcoded to HEVC. We denote two important use cases: offline transcoding and on-the-fly transcoding. The first one consists in transcoding and storing the resulting stream for a future usage, while the latter consists in transcoding and delivering the resulting stream in real time to a device. In this paper, we focus on this second scenario on H.264 sequences using an IPPP structure (a structure commonly used by mobile devices).

This work was funded by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 428942-11). Emails: jean-francois.franche.1@ens.etsmtl.ca, stephane.coulombe@etsmtl.ca

The simplest transcoding solution is to fully decode the input sequence and re-encode the pixels data in the output format. This approach, called cascaded pixel-domain transcoding (CPDT), achieves high coding efficiency and does not add any constraint on the encoding parameters. However, its computational complexity is very high.

Since video compression is based on spatial and temporal predictions, modern transcoders reuse decoded information from the input stream to accelerate tasks related to prediction. These transcoders adapt the prediction information from one format to another, taking onto account the prediction model differences between these two formats. In practice, this adaptation must achieve a good compromise between coding efficiency and computational complexity reduction.

Hence, several works propose different mode reduction techniques since the computational complexity greatly depends on mode evaluation [3]–[14]. Some of them perform a mapping between the H.264 decoded information and the HEVC modes to evaluate. For example, Fang et al. propose heuristics based on H.264 modes, motion vectors (MVs) variance and encoded residual to determine a list of candidate HEVC modes [3]. Jiang et al. adapt the CTU depth search range according to H.264 encoded bits, and eliminate unpromising partitioning structures by an MV clustering method [4]. Because only H.264 information is considered, this kind of approach achieves a limited speed-up or greatly affects the coding efficiency. To improve performance, other approaches exploit HEVC information in addition to H.264 information. For example, Peixoto et al. build a statistical model of HEVC rate distortion (RD) cost to establish mode-specific thresholds [5]. When the RD cost for a candidate mode is below its corresponding threshold, the rate distortion optimization (RDO) process ends. Similarly, Shen et al. propose numerous conditions based on H.264 information, depth levels statistics and HEVC RD cost to terminate the CTU splitting process [10]. However, these methods process the coarsest regions firstly and are then not very effective in complex regions that require a fine partitioning structure.

Motion estimation is another important issue extensively covered in the video transcoding literature. In video coding, motion search algorithms usually define a large search window to find the best MV since the motion is unknown. In contrast, several works on video transcoding refine the motion from the input sequence by using a smaller window. This motion

Copyright © 2017 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

refinement is usually performed in three steps. The first step determines an initial integer MV, usually an H.264 MV located in the corresponding H.264 region. The second step refines this MV with a small search range. Finally, the third step refines the best integer MV at half and quarter pixel precision. For the second step, some approaches use a constant search range. For example, Shen et al. propose a heuristic, based on H.264 MV and HEVC predictors, to select the starting point and refine it with a search range of 4 pixels [10]. Other authors compute the search range dynamically. For example, Zong et al. use advanced motion vector prediction (AMVP) as a starting point and define the search range as the greatest difference between this point and all H.264 MVs covering the processed region [3]. These different approaches reduce the computational complexity for the integer pixel search stage. However, they do not reduce the complexity of the last step, a complex process that requires the application of interpolation filters to calculate the pixel values located at fractional positions.

In our previous work [15], we proposed an H.264-to-HEVC transcoder combining a fast motion propagation algorithm and a fast mode decision framework. The motion propagation algorithm creates an MV candidate list at the CTU level. Thereafter, it selects the best candidate for each inter partition evaluated during the mode decision process. Compared to other methods, this algorithm requires no motion refinement and avoids computational redundancies by pre-computing prediction errors during the CTU initialization. The fast mode decision framework is based on a post-order traversal of the CTU, and includes several mode reduction techniques. In particular, the framework permits the early termination of the rate distortion cost computation, a highly complex task, when a mode is unpromising. The finest partitioning structure of the CTU is determined by using a structural split decision method. This method reuses the extracted H.264 modes to determine whether a coding unit (CU) must be split to evaluate sub-CUs. It preserves the coding efficiency, but overestimates the maximal quadtree depth, notably for a high quantization parameter (QP) transcoding. This overestimation limits the speed-up since more modes are evaluated.

In this paper, we extend our previous work to improve the transcoding performance in terms of speed-up and coding efficiency. The motion propagation algorithm is improved by supporting multiple reference frames, by considering the chroma components in the motion cost equation and by using a summed area table (integral image) to store the pre-computed prediction errors. To improve the fast mode decision process, a method to compute the lower bound of motion cost given a partitioning structure is proposed. This method computes the bounds by reusing information created by the motion propagation algorithm during the CTU initialization. These lower bounds are notably used by a novel CTU split decision algorithm. The paper also presents a detailed analysis of our motion propagation algorithm, as well as a more detailed description of the various components and methods of the proposed transcoder.

An overview of the proposed transcoder is given in Fig. 1. The transcoding system receives extracted information (modes, MVs and residuals) from the H.264 decoder. This system also receives information (MVs, RD cost, best mode) from the HEVC encoder, as illustrated by the feedback loop. The transcoding system is composed of the motion propagation algorithm, represented by the yellow boxes, and the fast mode decision process, illustrated by the blue boxes. These processes will be discussed in detail in this paper.

TABLE I: H.264 and HEVC tools comparison

Tool	H.264	HEVC
Partition Size	16×16 (MB)	64×64[1] (CTU)
Inter partitioning	16×16 to 4×4	64×64 to 8×4 and 4×8
Motion prediction	Median predictor	Advanced motion vector predictors (2 candidates)
Motion copy	Skip (1 candidate)	Skip/merge (5 candidates)
Motion precision	Half and quarter pel	Quarter pel
Intra Partitioning	16×16, 4×4	From 64×64 to 4×4
Intra prediction	Up to 9 predictors	Up to 35 predictors

[1] CTU size is configurable from 8×8 to 64×64 samples. For our experiments, the CTU size is set to 64×64.

The rest of this paper is organized as follows: Section II presents a technical background of the H.264 and HEVC standards, and of the HEVC HM Reference Software version 12.1. Section III presents and analyzes our motion propagation algorithm. Section IV describes the improved fast mode decision framework and present the novel split decision method based on motion propagation information. Section V shows the experimental results of the proposed transcoder. Section VI summarizes the work and draws conclusions.

II. Technical Background

This section provides the technical background and notations essential to understanding the rest of this paper. Section II-A compares H.264 and HEVC coding tools, with a focus on prediction tools. Section II-B and section II-C respectively describe the mode decision process and the motion search algorithms of HEVC HM Reference Software 12.1 [16], two important modules modified in the implementation of our approach.

A. HEVC and H.264 Coding Tools Comparison

In H.264 [2], the basic processing unit is the macroblock (MB), and represents a block of 16×16 samples. Each MB has a prediction mode (intra, inter or skip). An intra MB supports 2 partition modes: 16×16 and 4×4. An inter MB must be partitioned into 16×16, 16×8, 8×16 or 8×8 blocks. An 8×8 block can be sub-partitioned into 8×4, 4×8 or 4×4 blocks. Each inter block has its own MV. The skip MB is a special case of an inter MB encoded with the predicted MV and without residual data.

In HEVC [1], the maximum block size is extended to 64×64 samples, notably to reduce the prediction cost of simple regions [17]. The basic processing unit is the coding tree unit (CTU), and is represented by a quadtree structure. In this tree, each node is associated with a coding unit (CU) denoted $C_{i,j}$, representing the jth CU at depth level i. The quadtree maximum depth is 4 and the CU minimum size is 8×8. When a CU $C_{i,j}$ is split, its children correspond to sub-CUs $C_{i+1,4j+k}$ with $k = 0 \ldots 3$. Fig. 2 shows an example of

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

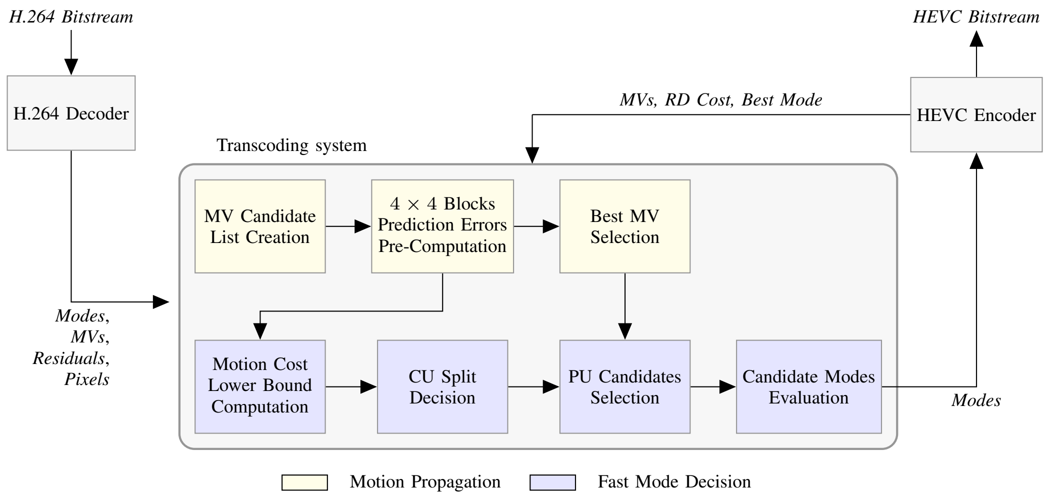
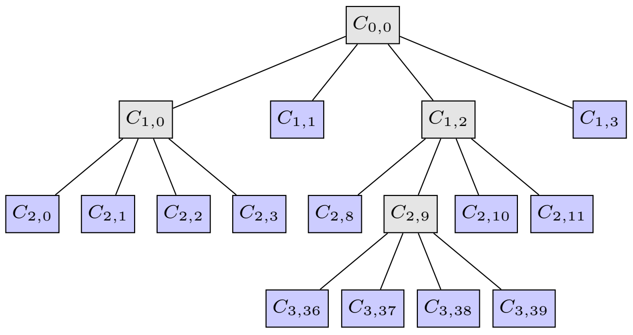
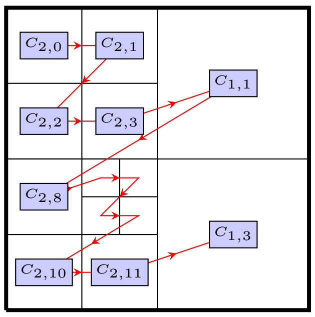


Fig. 1: Proposed transcoder architecture.



(a) Quadtree representation



(b) Geometrical representation

Fig. 2: Example of CTU partitioning. The red path shows the z-scan coding order and dependencies between CUs.

a CTU quadtree and its corresponding partitioning structure. The red path in (b) shows the z-scan coding order of CUs. This coding order determines dependencies between CUs. Hence, a CU can only be processed when sibling nodes located to its left and their descendants have been processed, as shown in Fig. 2(a).

When a CU is a leaf node, it is associated with a prediction unit (PU). The PU contains the prediction model. A PU may be of type intra, inter or skip/merge. An inter PU supports 4 symmetric partition modes (2N×2N, N×2N, 2N×N, N×N) and 4 asymmetric motion partition (AMP) modes (2N×nD 2N×nU, nL×2N, nR×2N).

For motion compensation, H.264 and HEVC support quarter pel precision, but HEVC uses more efficient interpolation filters [18]. H.264 predicts the motion by a median predictor. HEVC improves the motion prediction by supporting two modes: the AMVP mode and the merge mode. The AMVP mode supports up to two motion predictors, and commonly uses a motion search algorithm to find the best MV. The merge mode copies motion information from a neighboring (spatial or temporal) block. This mode supports up to 5 candidates and activates a skip flag when no residual is encoded. In general, the merge process propagates motion information in uniform regions, while the AMVP process discovers a new MV in a complex region. Table I summarizes the main differences between H.264 and HEVC.

B. Mode Decision in the HM Encoder

To determine the best mode, the HM mode decision process visits the CTU tree in a pre-order traversal, as shown in Fig. 3(a). When a CU is visited, the skip/merge, inter, and then intra modes, are evaluated by the RD cost function defined as:

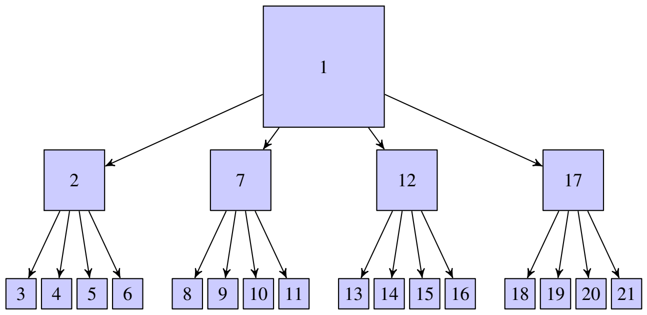
$$J_{\text{RD}} = (\text{SSE}_{\text{luma}} + w_{\text{chroma}}\,\text{SSE}_{\text{chroma}}) + \lambda_{\text{mode}} \cdot R_{\text{mode}} \quad (1)$$

where SSE_{luma} and $\text{SSE}_{\text{chroma}}$ are the sum of squared errors between the original input image block and the reconstructed block for luma and chroma, respectively; w_{chroma} is a weighting factor dependent on the encoding configuration, as specified in [16]; R_{mode} is the number of bits to encode the current mode and λ_{mode} is the Lagrange multiplier.

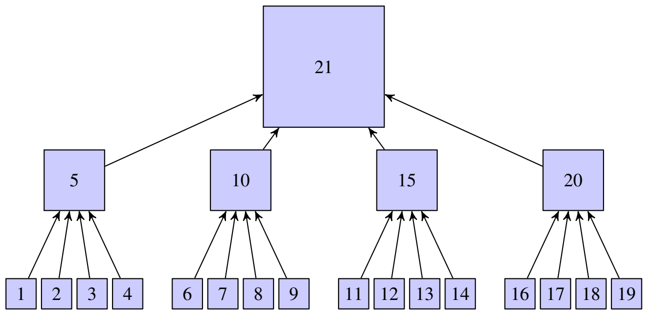
When all the descendants of the current CU have been visited, its J_{RD} is compared with the combined J_{RD} values of these 4 sub-CUs. The CU is split when the sum of the J_{RD} values of the sub-CUs is smaller than that of the current CU. This process is repeated recursively on each sub-CU when the CU is split. At the end of the process, the best mode and CTU partitioning are obtained.

To reduce the number of evaluated modes, the HM early terminates the mode decision process when the current best mode contains no residual data. This method is particularly effective in uniform regions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology



(a) Pre-order traversal



(b) Post-order traversal

Fig. 3: Comparison between a) HM CTU traversal and b) proposed CTU traversal. Indices in nodes show the order in which CUs are processed by the traversal method.

C. Motion Search in the HM Encoder

To find the best MV for an AMVP PU partition, the HM encoder applies a four-stage motion estimation algorithm as follows:

- **1:** Select the best starting point between the two AMVP predictors and $V_{(0,0)}$.
- **2:** Perform a motion search with integer pel precision.
- **3:** Refine the best MV at half and quarter pel precision.
- **4:** Update the selected AMVP predictor as needed.

The third stage successively performs a refinement at half and quarter pixel precision. Each refinement evaluates the 8 neighbors of the best MV. This is a complex stage because all evaluated candidates must be interpolated with an 8-tap filter.

To select the best MV, the HM defines two motion cost functions, one for the integer pixel precision and one for sub-pel precision, defined respectively as :

$$J_{\text{SAD}} = \text{SAD}(\mathbf{D}) + \lambda_{pred} \times R_{\text{motion}} \quad (2)$$

$$J_{\text{SATD}} = \text{SATD}(\mathbf{D}) + \lambda_{pred} \times R_{\text{motion}} \quad (3)$$

where R_{motion} is the number of bits to encode motion information, $\mathbf{D}$ is the difference between the predicted block and the original block, and λ_{pred} is the Lagrange multiplier of the cost function, defined as:

$$\lambda_{pred} = \sqrt{\lambda_{mode}} \quad (4)$$

For integer pixel precision, the prediction error is measured by the sum of absolute differences (SAD), defined as:

$$\text{SAD}(\mathbf{D}) = \sum_{i=1}^{N} \sum_{j=1}^{M} |(d_{ij})| \quad (5)$$

where N and M are respectively the height and the width of the block. For fractional pixel precision, the prediction error is measured by the sum of absolute transformed differences (SATD). However, to limit the transform complexity, the current block is divided into partitions of 8×8 pixels or, for smaller blocks (4×8 and 8×4), into partitions of 4×4 pixels. More formally, each block is divided into partitions $\mathbf{D}_{i,j}$ of $S \times S$ pixels. Then, the SATD is applied to each partition using:

$$\text{SATD}(\mathbf{D}_{i,j}) = \text{SAD}(\text{T}(\mathbf{D}_{i,j})) \quad (6)$$

where $\mathbf{D}_{i,j}$ is the difference between the predicted partition and the original partition, and $\text{T}(\mathbf{D}_{i,j})$ its Hadamard transform. Finally, the SATDs of each PU partition are summed up as:

$$\text{SATD}(\mathbf{D}) = \sum_{i=1}^{N/S} \sum_{j=1}^{M/S} \text{SAD}(\text{T}(\mathbf{D}_{i,j})). \quad (7)$$

The last stage selects the AMVP predictor that minimizes the motion cost defined by Eq. (3). The HM applies this four-stage motion search algorithm to each AMVP PU partitions.

III. Proposed Motion Propagation Algorithm

Several approaches proposed in the video transcoding literature assume the motion information extracted from the H.264 sequence must be refined during the transcoding to preserve the coding efficiency, compared to a CPDT approach. In fact, motion refinement is required to exploit more accurate motion estimation tools, such as MVs with a greater fractional pixel precision and smaller block sizes. For example, an H.263-to-H.264 transcoder must refine the input MVs from a half-pel precision to a quarter-pel precision to reduce the prediction error [19] and hence achieve a better coding efficiency.

For an H.264-to-HEVC transcoder, motion refinement seems less necessary since both coding formats support quarter-pel precision MVs, and the smallest block size increased in HEVC from 4×4 to 8×4 and 4×8 pixels. However, other factors may motivate the use of motion refinement. For example, Zong et al. observed that motion refinement improves the coding efficiency when the H.264 MV and the AMVP selected by the HEVC encoder are different [8].

In this section, we present a motion propagation algorithm as an alternative solution to motion refinement. The main idea behind this algorithm is that H.264 contains motion information sufficiently accurate to be reused in HEVC without further refinement, but this motion information is propagated differently in HEVC than in H.264. Hence, although the best MV (including the reference frame index) for the currently processed region in HEVC is generally located in the corresponding region in H.264, a better MV might be found in the neighborhood of this region or in a list of previously encoded HEVC MVs.

The proposed algorithm creates an MV candidate list during the CTU initialization, as will be explained in section III-A. This list is composed of H.264 and HEVC MVs that have a high probability of being propagated in the CTU, as analyzed in section III-C. Thereafter, during the mode decision process, the algorithm selects, for each AMVP PU, the best MV

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

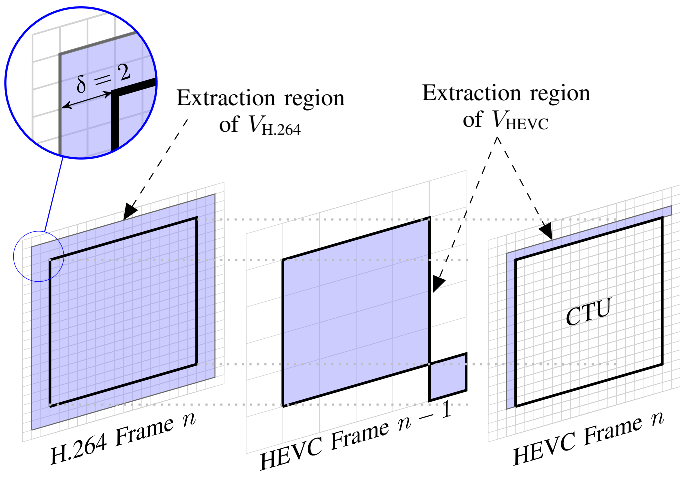

δ = 2
Extraction region of $V_{\text{H.264}}$
Extraction region of V_{HEVC}
CTU
H.264 Frame n
HEVC Frame $n-1$
HEVC Frame n

Fig. 4: The motion vector candidate list is composed of H.264 and HEVC MVs extracted from the blue regions. The cells in the grids represent the stored motion data. In the H.264 frames and the current HEVC frame, the motion is stored on a 4×4 block basis. For HEVC reference frames, the motion data is sub-sampled on a 16×16 block basis. The parameter δ defines the neighborhood size around the collocated CTU, in 4×4 blocks, of the extracted H.264 motion data.

candidate. Since all PUs evaluate the same MV candidate list, the prediction error of each candidate is pre-computed during the CTU initialization on a 4×4 block basis to eliminate computational redundancy, as explained in section III-B.

A. Motion Vector Candidate List Generation

The MV candidate list generated by the proposed method is composed of H.264 and HEVC MVs (including the reference frame indexes) extracted from the regions shown in Fig. 4. Since videos usually contain several regions without motion, the null MV, denoted $V_{(0,0)}$, is also added to the list for all reference frames. All duplicate MVs are removed from the list, and the number of candidates is denoted K.

We denote $V_{\text{H.264}}$ the list of all H.264 MVs extracted from the regions collocated with the CTU region or its neighborhood. The parameter δ defines the neighborhood size and manages the trade-off between the coding efficiency and the computational complexity. This parameter unit is a 4×4 block because H.264 stores the motion data on a 4×4 block basis. When δ is set to 0, the H.264 MVs are only extracted from the region collocated in the CTU region.

We denote V_{HEVC} the list of HEVC MVs extracted from regions already processed in the CTU neighborhood (and stored on 4×4 block basis) or collocated in the HEVC reference frame (and stored on a 16×16 block basis since the HEVC standard mandates the sub-sampling of temporal MVs). These MVs are required as they can be merge candidates or AMVP predictors for the current CTU.

B. Motion Estimation Computational Redundancy Elimination

The complex partitioning structure of a CTU can cause major computational redundancy for motion estimation. As the motion data of HEVC can be represented on a 4×4 block basis, up to 24 overlapping AMVP modes (3 symmetric modes by depth for depths 0 to 3; 4 asymmetric modes by depth for depths 0 to 2) can cover the same 4×4 block. This number increases to 28 if the merge modes are also considered. Hence, the prediction error (SATD) and interpolated pixels (in case of fractional pixel) for a given MV can be computed up to 28 times for the same 4×4 block. The exact amount of redundancy depends on various factors, such as the motion activity in the CTU, the AMVP modes evaluated by the mode decision process, and the motion estimation approach employed (motion search, motion refinement or motion propagation).

When a motion search or a motion refinement algorithm is employed, the computational redundancy is difficult to eliminate because the evaluated MVs are generally determined at the PU level, and can vary from one PU to another in the same CTU. In the literature, some parallel motion estimation approaches have been proposed to eliminate this redundancy [20], [21]. However, these approaches reduce the coding efficiency because they don't consider the real AMVP predictors, and are not appropriate for a sequential execution.

In contrast, since the MV candidates are fixed for the whole CTU, our motion propagation approach easily removes this redundancy at the CTU level by pre-computing the prediction errors (SATD) on a 4×4 block basis for each MV candidate, as described in section III-B1. Thereafter, during the mode decision process, the prediction errors (SATD) of each 4×4 block covering a partition are summed up to get the total prediction error (SATD) for a PU candidate, as explained in section III-B2.

1) Prediction Errors Pre-Computation: For all MV candidate, the prediction errors are pre-computed in two steps. Since the MV has a quarter-pel precision, the first step interpolates the predicted CTU region, usually a 64×64 region, if necessary. The second step computes the prediction error for each 4×4 block covering the CTU region. For a 4×4 block located at position ($4x$,$4y$) relative to the CTU upper left corner, the prediction error function is defined as:

$$\begin{aligned}\mathbf{e}_{4\times4}(x,y,k) = &\,\text{SATD}(B^{\text{luma}}_{x,y,k}) + \\ &\text{SATD}(B^{\text{Cb}}_{x,y,k}) + \text{SATD}(B^{\text{Cr}}_{x,y,k})\end{aligned} \tag{8}$$

where $B^{\text{luma}}_{x,y,k}$ contains the differences between the predicted and the current 4×4 luma blocks at position $(4x,4y)$ for the kth MV candidate, with $x,y \in \mathbb{N}$. $B^{\text{Cb}}_{x,y,k}$ and $B^{\text{Cr}}_{x,y,k}$ represent the same differences for the corresponding 2×2 chroma blocks. Contrary to the original HM and our previous transcoder [15], the chroma blocks are considered in this equation (in addition to the luma block) to improve the prediction accuracy.

2) Partition's Prediction Error Computation: At the PU level, the prediction errors of the 4×4 blocks covering the PU partition region are summed up to get the total prediction error. Hence, for a given PU of $4M\times4N$ samples located at position ($4x$, $4y$) relative to the CTU upper left corner, the prediction error of the kth MV candidate is computed as:

$$\mathbf{e}_{M\times N}(x,y,k) = \sum_{i=x}^{(x+M-1)} \sum_{j=y}^{(y+N-1)} \mathbf{e}_{4\times4}(i,j,k) \tag{9}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

3) Prediction Errors Using a Summed Area Table: The complexity of prediction errors summation at the PU level can be reduced by using a summed area table (integral image) [22]. Hence, Eq. (8) and (9) can be advantageously replaced by the following equations:

$$\begin{aligned}\mathbf{I}_{4\times4}(x,y,k) =&\mathbf{e}_{4\times4}(x,y,k)+\\ &\mathbf{I}_{4\times4}(x-1,y,k)+\\ &\mathbf{I}_{4\times4}(x,y-1,k)-\\ &\mathbf{I}_{4\times4}(x-1,y-1,k)\end{aligned} \tag{10}$$

$$\begin{aligned}\mathbf{e}_{M\times N}(x,y,k) =&\mathbf{I}_{4\times4}(x+M,y+N,k)+\\ &\mathbf{I}_{4\times4}(x,y,k)-\\ &\mathbf{I}_{4\times4}(x+M,y,k)-\\ &\mathbf{I}_{4\times4}(x,y+N,k)\end{aligned} \tag{11}$$

4) Best Motion Vector Selection: The cost function for an AMVP partition is obtained by adding the motion cost to Eq. (11) as follows:

$$\begin{aligned}\mathrm{J}_{M\times N}(x,y,k,l) =&\mathbf{e}_{M\times N}(x,y,k)+\\ &\lambda_{pred}\times R_{\text{motion}}(k,l)\end{aligned} \tag{12}$$

where k is the index of the selected MV and l the index of the selected AMVP predictor. Finally, the proposed motion propagation algorithm selects, for each partition, the k and l combination that minimizes Eq. (12), computed as follows:

$$(k^*,l^*) = \underset{k=1\ldots K,l=1..L}{\arg\min}\left(\mathrm{J}_{M\times N}(x,y,k,l)\right) \tag{13}$$

C. Motion propagation analysis

The design of the proposed motion propagation algorithm is based on the following hypotheses:

- *Hypothesis 1*: The H.264 MVs are precise enough to be reused in HEVC without motion refinement.
- *Hypothesis 2*: Although the best H.264 MV for an HEVC region is often located in the corresponding region in the H.264 frame, in many cases, a better H.264 MV can be found in the close neighborhood of this H.264 region.

In order to validate these hypotheses and measure the performance of the proposed algorithm, we conducted several experiments on an H.264-to-HEVC transcoder. In section V, we describe the complete methodology and show that the motion propagation algorithm has a negligible impact on the coding efficiency compared to the HM Test Zone Search (TZS) motion estimation algorithm used in a CPDT transcoder. Consequently, these results validate Hypothesis 1. However, these results are insufficient to validate the second hypothesis, as they do not provide any information on the motion propagation process.

To resolve this problem, in this subsection, we analyze the relationship between the H.264 MVs and the HEVC MVs obtained with our motion propagation approach. Given an encoded HEVC block and its optimal MV, we want to determine the relative location of the nearest H.264 block having the same (or a similar) MV. Hence, if the nearest H.264 block is always collocated in the HEVC block, Hypothesis 2 is invalidated, and the parameter δ (defined in section III-A) must be set to 0. This would lead to a transcoder in which the MV candidate list only needs to include H.264 MVs collocated in the HEVC block. Conversely, if the nearest H.264 block is often located outside the collocated region, Hypothesis 2 is validated, and the parameter δ must be determined.

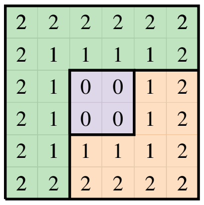


Fig. 5: Representation of various block locations and their Chebyshev distance. Values inside the blocks represent the Chebyshev (chessboard) distance between a PU and a 4×4 block. Each 4×4 block belongs either to the PU (PU) region, the above or left (AL) region or the bottom or right (BR) region.

To determine the impact of δ on the encoded HEVC MVs, we conducted an experiment with the parameter δ set to 16 (i.e., a size of 64 pixels). This experiment computes the distance D_{block} in 4×4 blocks, between each HEVC block and its nearest H.264 4×4 block having a similar MV. As illustrated in Fig. 5, the distance D_{block} is measured by the Chebyshev distance. We compute the Chebyshev distance between two blocks located at positions $\mathbf{b}_1$ and $\mathbf{b}_2$ as follows:

$$D_{\text{block}}(\mathbf{b}_1,\mathbf{b}_2) = \max(|b_{1_x}-b_{2_x}|,|b_{1_y}-b_{2_y}|). \tag{14}$$

The distance between two blocks' MVs, $\mathbf{m}_1$ and $\mathbf{m}_2$, is also measured using the Chebyshev distance:

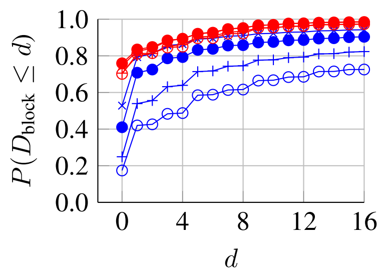
$$D_{\text{V}}(\mathbf{m}_1,\mathbf{m}_2) = \max(|m_{1_x}-m_{2_x}|,|m_{1_y}-m_{2_y}|). \tag{15}$$

When this distance is lower or equal to a tolerance parameter t, the MVs are considered similar. In summary, for each HEVC block, we compute the smallest distance D_{block} among the H.264 blocks having MVs such that $D_{\text{V}}\leq t$ with respect to the HEVC MV.

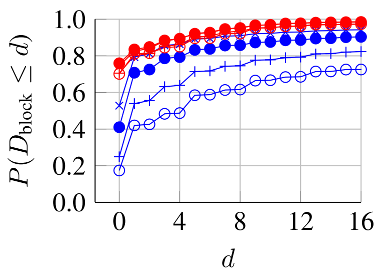
The CDFs of the distance between an HEVC block and the nearest H.264 block having a similar MV are shown in Fig. 6 for three sequences and three t values (0/4, 1/4, 3/4 pixels). The data shows two different trends for the AMVP and merge modes. For the AMVP mode, the HEVC MV is often collocated in the H.264 region and the CDF rapidly converges to 1. For the merge mode, the HEVC MV is less frequently collocated in the H.264 frame, in particular for high QP, and the CDF does not always reach 1. This phenomenon is explained by the fact the merge and AMVP modes play complementary roles in HEVC. The merge mode is typically associated to a region with a uniform motion. It propagates an existing HEVC MV in its neighborhood and the H.264 MVs have no direct influence on this mode. Inversely, the AMVP mode is usually used to represent a discontinuity in motion. This discontinuity is generally present in the collocated H.264 region or its neighborhood. Finally, increasing the QP favors the merge modes (as shown in Fig. 7(a)) and coarser partitions. Consequently, the HEVC MVs located in the neighborhood gain influences over the H.264 MVs when the QP increases,

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

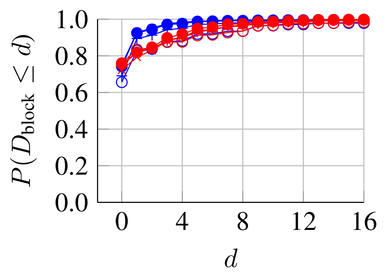
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology



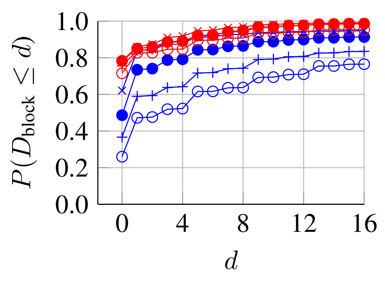
(a) RaceHorses, $t = 0$ pixel



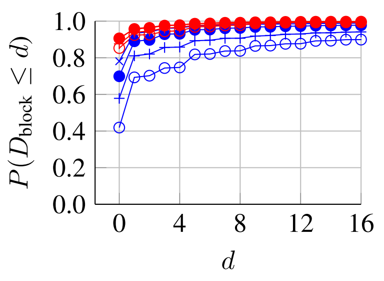
(b) BQSquare, $t = 0$ pixel



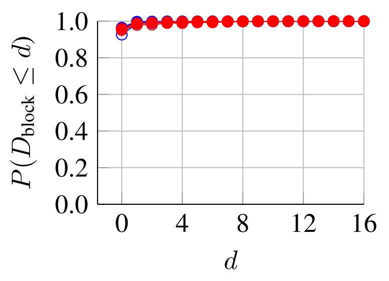
(c) Keiba, $t = 0$ pixel



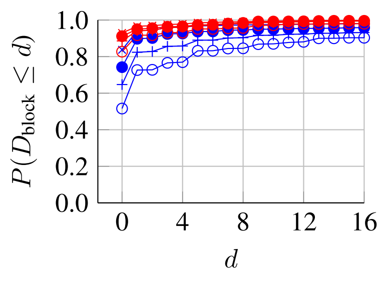
(d) RaceHorses, $t = 1/4$ pixel



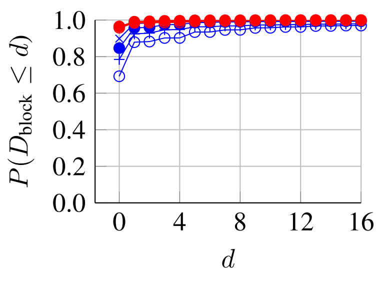
(e) BQSquare, $t = 1/4$ pixel



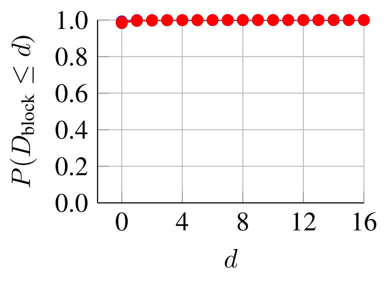
(f) Keiba, $t = 1/4$ pixel



(g) RaceHorses, $t = 3/4$ pixel



(h) BQSquare, $t = 3/4$ pixel



(i) Keiba, $t = 3/4$ pixel

▭ PU ▭ AL ▭ BR

Fig. 6: CDF of the distance D_{block}, measured in $4{\times}4$ blocks, between an HEVC block and the nearest H.264 block having a similar MV for AMVP and merge modes using RaceHorses, BQSquare and Keiba video sequences at various QPs and tolerance levels. Two MVs are considered similar when their Chebyshev distance is less or equal to a tolerance of t pixels.

and the HEVC MV associated with a merge mode is more rarely collocated in the H.264 frame.

These results validate Hypothesis 2 for both modes. Indeed, the encoded HEVC MV is not always present in the collocated H.264 region because $P(D_{\text{block}} = 0) < 1$, but is present in a close neighborhood since $P(D_{\text{block}} = 0) < P(D_{\text{block}} \leq d)$ with small values of d. However, it is important to note that the merge mode has no direct influence on the choice of δ since the merge mode uses MVs already present in V_{HEVC}. Although the increase of δ makes it possible to find a better H.264 MV for a PU, we found experimentally that the best trade-off between coding efficiency and computational complexity is achieved when the δ is set to 1 (i.e., 4 pixels).

From Fig. 6, we can also observe that when the tolerance is set to 3/4 pixels, the CDF of the AMVP modes are close to 1 for $d = 0$. That means the increase of δ when $t = 0$ generally has the effect of finding a MV similar to an H.264 MV collocated in the PU region followed by a refinement. Hence, the data suggests that the motion propagation algorithm (without refinement) over a neighborhood can achieve a similar MV as a fractional-pixel motion refinement algorithm using collocated MVs. We will show this is the case in section V.

Since the motion is predicted from previously encoded blocks located above or to the left of the current block, the next experiment studies the relationship between the encoded HEVC MV and the H.264 location where this MV is found. We define four locations: the AL, PU, BR and absent locations. As shown in Fig. 5, the AL location is an H.264 region located above or to the left of the PU region; the PU location is the H.264 region collocated in the PU region; and the BR location is an H.264 region located below or to the right of the PU region, but absent from the AL location. These locations are analyzed in this order to determine where HEVC MV is found for the first time. If the HEVC MV is not found, the MV is associated with the absent location.

From Fig. 7(b) and Fig. 7(c), we once again observe two different trends for the merge and AMVP modes that corroborate the idea that these modes have two distinct roles. Hence,

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

the PU location is largely more frequent for the AMVP mode, as compared to the merge mode. This location represents the emergence of a new MV in the H.264 region (i.e., an MV not present in the AL location). The merge mode is inappropriate to represent this discontinuity in motion. Inversely, the location AL is more frequent in the merge mode, since this mode propagates an HEVC MV located above or to the left of the PU. The merge mode also includes a greater occurrence of absent MVs from the considered H.264 MVs (i.e., only present in V_{HEVC}). An MV absent from the H.264 MVs may correspond to an H.264 MV spatially located outside the H.264 extraction region, but propagated via an HEVC MV located in the neighborhood. Alternatively, it may correspond to an HEVC MV extracted from an HEVC reference frame.

In summary, the proposed motion propagation algorithm is based on two hypotheses. Hypothesis 2 was validated in this section, and Hypothesis 1 will be validated in section V. We have shown that the relationship between the H.264 and HEVC MVs is different for the AMVP and the merge modes. Our experiments have shown that the best trade-off between coding efficiency and computational complexity is achieved when the δ is set to 1.

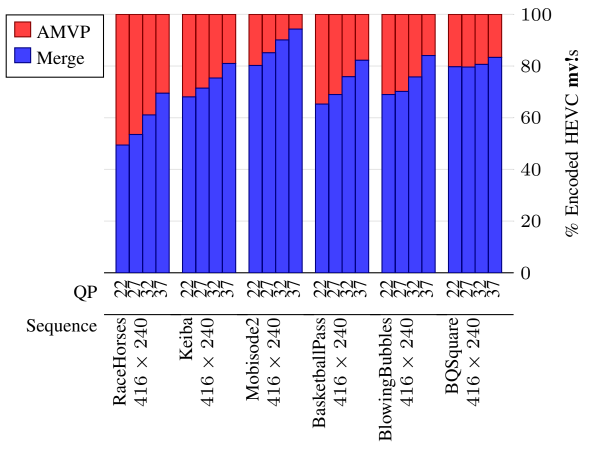
IV. Fast mode decision framework based on post-order traversal of the CTU structure

As mentioned earlier, many researchers have proposed approaches to reduce the complexity of the mode decision process in an H.264-to-HEVC transcoding context. These approaches reuse information extracted from H.264 to create a subset of modes to be evaluated by the HEVC mode decision process. In addition to these mode reduction techniques, few approaches early terminate the mode decision process when the current best HEVC mode is satisfactory; for instance, when it has an RD cost under a threshold.

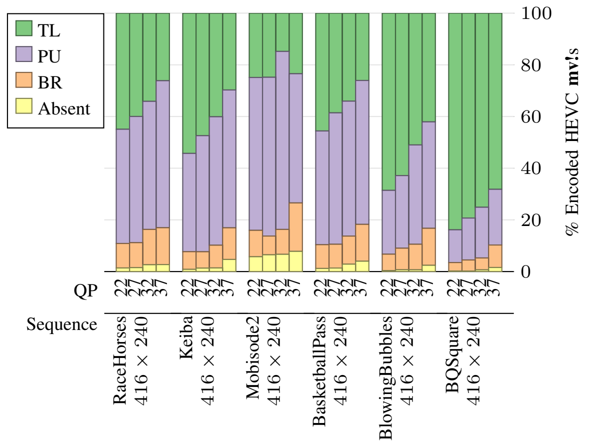
All of these approaches are based on a pre-order traversal of the CTU structure, as shown in Fig. 3(a). To reduce the number of evaluated CUs, these approaches must successively:

1) Determine whether the current CU must be processed, usually by exploiting extracted information from H.264, and process it if needed.
2) Determine whether the sub-CUs must be processed, ideally by exploiting extracted information from H.264 and the HEVC information of the current CU, when this latter has been processed.

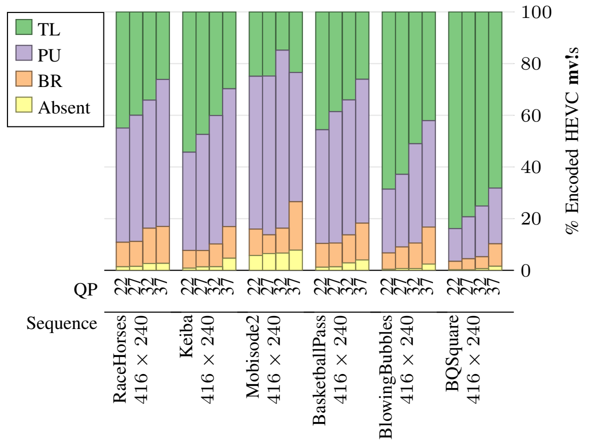
These are two complex decisions because many combinations of sub-CUs and PUs are possible, and any of them may improve the coding efficiency of the current CU. For example, CU #1 in Fig. 3(a) competes with several combinations of sub-CUs and PUs. It is therefore difficult to determine if the transcoder must process this CU, its sub-CUs, or both of them. In addition to this problem, the pre-order traversal allows a comparaison between a CU (the CU #2) and its sub-CUs (#3, #4, #5 and #6) only when all their modes have been processed. The current CU (CU #2) can't be compared to a sub-CUs (CU #3), since the two CUs have different sizes and are therefore incompatible. Hence, information on the current CU is not helpful to take a decision on a single sub-CU in a pre-order traversal.



(a) Distribution of merge and AMVP modes



(b) Distribution of H.264 locations for AMVP modes



(c) Distribution of H.264 locations for merge modes

Fig. 7: Motion propagation characteristics of encoded HEVC MVs: (a) shows HEVC MVs are often propagated since the merge mode is more frequent than the AMVP mode, (b) shows the encoded HEVC MV is frequently collocated with the H.264 region for an AMVP mode, (c) shows this MV is frequently located above or left of the H.264 region for the merge mode.

To address these issues, we opted for a post-order traversal of the CTU structure as shown in Fig. 3(b). This traversal reverses the previous problem and, hence, the transcoder must successively:

1) Determine whether the sub-CUs of the current CU must be processed, by exploiting extracted information from H.264 and information derived by the motion propagation algorithm, and process recursively the sub-CUs if needed.
2) Determine whether the current CU must be processed, by exploiting information extracted from H.264 and computed during the processing of sub-CUs.

For a sub-CUs combination (CUs #1, #2, #3 and #4 on Fig. 3(b)), the transcoder must decide whether the parent CU (CU #5) must be processed and, if that is the case, which PU modes must be processed. This problem is simpler than the problem related to the pre-order traversal for two reasons. First, the current best mode is in competition with only one mode a time (a PU mode of the parent CU). Second, the best sub-CUs combination covers the same region as the parent CU. Hence, information of these sub-CUs can be combined and used to take a decision on the parent CU. For example, in the proposed approach each promising mode computes a low-complexity cost, denoted by J_{PM}. The J_{PM} of the best sub-CUs combination are summed and compared with the J_{PM} cost of the parent CU to early terminate its processing if necessary.

The main algorithm of the proposed approach is shown in Fig. 8, where $C_{i,j}$ represents the jth CU at depth level i as shown in Fig. 2. The first part (lines 3-10) of this algorithm processes all the descendants of the current CU recursively according to a post-order traversal. The CU-Split has the role to determine if the sub-CUs must be visited by exploiting available information. The second part (lines 11-18) processes the PU modes of the current CU returned by the Candidate-Modes function (line 12). A PU mode is evaluated in two steps. The first step (line 13) computes the low-complexity cost J_{PM} by calling the PU-PM-Cost function. If the candidate mode is promising (condition on line 14 is satisfied), the second step computes the RD cost, a high-complexity cost function denoted by J_{RD}, by calling the PU-RD-Cost function. The condition on line 14 is very effective at reducing computations in the proposed post-order traversal because the J_{PM} of the best sub-CUs combination can be compared with the J_{PM} of a PU belonging to the parent CU.

A. Split Decision

The first step (line 4) of the CU-Process function (Fig. 8) consists in determining if the current CU must initially be split. The split decision is taken by the CU-Split function shown in Fig. 9. This function comprises two alternative split decision methods : a structural split decision method (lines 6-7), already proposed in our previous work, and a novel motion-based split decision method (lines 9-15).

The structural approach creates a direct mapping between the H.264 partitioning structure and the split decision for a $16{\times}16$ CU based on the assumption that HEVC partitioning is rarely finer than H.264 partitioning. Hence, the H.264-Depth

```
1  procedure CU-Process($C_{i,j}$)
2      $J_{\text{RD}}[C_{i,j}] \leftarrow J_{\text{PM}}[C_{i,j}] \leftarrow \infty$
3      ▷ Recursively process the sub-CUs
4      if CU-Split($C_{i,j}$)
5          $J_{\text{RD}}[C_{i,j}] \leftarrow \lambda_{\text{mode}} \cdot R_{\text{split}}$
6          $J_{\text{PM}}[C_{i,j}] \leftarrow \lambda_{\text{pred}} \cdot R_{\text{split}}$
7          for $k \leftarrow 0$ to $3$
8              CU-Process($C_{i+1,4j+k}$)
9              $J_{\text{RD}}[C_{i,j}] \leftarrow J_{\text{RD}}[C_{i,j}] + J_{\text{RD}}[C_{i+1,4j+k}]$
10             $J_{\text{PM}}[C_{i,j}] \leftarrow J_{\text{PM}}[C_{i,j}] + J_{\text{PM}}[C_{i+1,4j+k}]$
11     ▷ Process inter PUs of CU
12     for each $m$ in Candidate-Modes($C_{i,j}$)
13         $J_{\text{PM}}[m] \leftarrow$ PU-PM-Cost($C_{i,j}$,$m$)
14         if ($J_{\text{PM}}[m] < (J_{\text{PM}}[C_{i,j}] + T)$
15             $J_{\text{RD}}[m] \leftarrow$ PU-RD-Cost($C_{i,j}$,$m$)
16             if $J_{\text{RD}}[m] < J_{\text{RD}}[C_{i,j}]$
17                 $J_{\text{RD}}[C_{i,j}] \leftarrow J_{\text{RD}}[m]$
18                 $J_{\text{PM}}[C_{i,j}] \leftarrow J_{\text{PM}}[m]$
```

Fig. 8: The CU-Process procedure is the main algorithm of the proposed fast mode decision framework. The CTU traversal is performed by passing the parameter $C_{0,0}$ to this procedure.

```
1  function CU-Split($C_{i,j}$)
2      $size \leftarrow$ CU-Size($C_{i,j}$)
3      if $size = 8$                    ▷ If maximal depth reached
4          return FALSE
5      if $size = 16$
6          ▷ Structural split decision
7          return (H.264-Depth($C_{i,j}$) $= 1$)
8      else
9          ▷ Motion-based split decision
10         if H.264-HaveIntra($C_{i,j}$)
11             return TRUE
12         else
13             $\widehat{J}_{\text{NonSplit}} \leftarrow$ CU-Min-Cost($C_{i,j}$,$i$,$i$,$i$)
14             $\widehat{J}_{\text{Split}} \leftarrow$ CU-Min-Cost($C_{i,j}$,$i+1$,3,$i$)
15             return ($\widehat{J}_{\text{Split}} < \widehat{J}_{\text{NonSplit}}$)
```

Fig. 9: The boolean CU-Split function takes the decision whether or not to split the CU $C_{i,j}$ to evaluate sub-CUs.

function returns 1 when the smallest H.264 partition is equal to or less than $8{\times}8$ samples. Since HEVC supports larger partitions than H.264, this method is not applicable for a CU greater than $16{\times}16$ pixels.

To resolve this limitation, our novel motion-based split decision method exploits data created by the motion propagation algorithm to determine if a CU must be initially split or not. This method is applicable on $64{\times}64$ and $32{\times}32$ CUs, except when the collocated H.264 region has at least one intra MB. In this case, the CU is automatically split since motion information is not accurate enough to predict the partitioning structure and a H.264 intra region usually represents a complex region that requires a fine partitioning structure in HEVC.

1) Motion-Based Split Decision: The motion-based split decision method reuses the motion vector candidate list and the pre-computed prediction errors generated by our motion prop-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

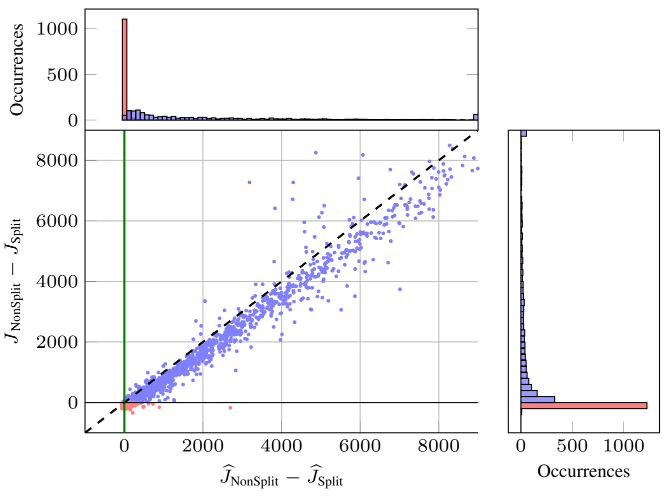


Fig. 10: Illustration of the linear relationship between the differences of lower bounds of motion cost, $\widehat{J}_{\text{NonSplit}} - \widehat{J}_{\text{Split}}$, and the differences of real motion costs, $J_{\text{NonSplit}} - J_{\text{Split}}$. To get the lower motion cost, the CU must be split for blue observations, and non-split for red observations. The observations are obtained from CUs of 32×32 samples.

```
1  function CU-MIN-COST($C_{i,j}, l, u, c$)
2      $J_{\text{min}} \leftarrow \infty$
3      if $c \geq l$
4          ▷ Process inter PUs of CU $C_{i,j}$
5          for each $mode$ in INTER-MODES($C_{i,j}$)
6              $J \leftarrow \lambda_{\text{pred}} \cdot mode.R_{\text{min}}$
7              for each $partition$ in $mode$
8                  $(x, y) \leftarrow partition.position$
9                  $(M, N) \leftarrow partition.size$
10                 $J \leftarrow J + \min(\mathbf{e}_{M\times N}(x, y, k)|k = 1..K)$
11             if $J < J_{\text{min}}$
12                 $J_{\text{min}} \leftarrow J$
13     if $c < u$
14         ▷ Recursively process the sub-CUs
15         $c \leftarrow c + 1$
16         $J \leftarrow \lambda_{\text{pred}} \cdot mode.R_{\text{Split}}$
17         for $k \leftarrow 0$ to 3
18             $J \leftarrow J +$ CU-MIN-COST($C_{i+1,4j+k}, l, u, c$)
19         if $J < J_{\text{min}}$
20             $J_{\text{min}} \leftarrow J$
21     return $J_{\text{min}}$
```

Fig. 11: The CU-MIN-COST function recursively computes a lower bound for J_{PM}, denoted J_{min}, given a CU $C_{i,j}$, a lower depth l, an upper depth u and the current depth c.

agation, during the CTU initialization, to determine whether a CU split has the potential to reduce the motion cost. Since the motion predictors are partially unknown when the CU CU-SPLIT function (Fig. 9) is called, the proposed method approximates the real motion cost by computing two lower bounds. One bound, denoted $\widehat{J}_{\text{NonSplit}}$, is computed for the non-split case (line 13), and the other, denoted $\widehat{J}_{\text{Split}}$, is computed for the split option (line 14).

As shown in Fig. 10, the difference between $\widehat{J}_{\text{NonSplit}}$ and $\widehat{J}_{\text{Split}}$ is highly correlated with the difference between J_{NonSplit} and J_{Split}, the real motion costs. Moreover, the plots show the difference between J_{NonSplit} and J_{Split} is generally lower than the difference between $\widehat{J}_{\text{NonSplit}}$ and $\widehat{J}_{\text{Split}}$, i.e., the data is generally located under the dashed line. Based on this latter observation, the motion-based split decision method decides to split a CU only when $\widehat{J}_{\text{Split}}$ is lower than $\widehat{J}_{\text{NonSplit}}$ (line 15). This condition allows to preserve the coding efficiency and reduces the number of evaluated modes compared to the structural method presented in our previous work.

2) Lower Bound Calculation: The lower bounds of motion cost are computed by the recursive function CU-MIN-COST presented in Fig. 11. The input parameters are: the CU $C_{i,j}$, the lower and upper allowed depths, denoted l and u, respectively, and the current depth level c. Hence, line 15 of the CU-SPLIT function (Fig. 9) computes the lower bound of $C_{i,j}$ for the non-split case (lower bound for the current depth level i). While, line 16 computes the lower bound for the split case (lower bound for depth levels from $(i+1)$ to u). In our experiments, we set u to 3. This depth level corresponds to a CU of 8×8 samples, the smallest CU size allowed by the HEVC standard.

The first part (lines 4-12) of the CU-MIN-COST function (Fig. 11) computes the lower bound for each inter modes (2N×2N, N×2N and 2N×N) in the current CU and selects the lowest one as the J_{min}. The second part (lines 14-20) recursively computes the lower bounds for sub-CUs and updates the J_{min} as needed. The lower bound calculation is performed by reusing the prediction errors pre-computed by the motion propagation algorithm during the CTU initialization. Since the neighboring CUs are partially or totally unknown, the lower bound calculation assumes the motion predictors are unknown, and the motion cost is then not considered. However, a penalty cost is added based on the partitioning structure.

This penalty cost corresponds to an approximation of the minimal number of bits required to encode the prediction information. Hence, the lower bound of a PU (lines 6-12) is computed by summing the minimization of Eq. (12) (line 10) for each partition and adding the penalty bits, denoted $mode.R_{min}$, multiplied by the Lagrange multiplier λ_{pred} (line 6). The lower bound for the sub-CUs is computed similarly by summing the lower bounds of the 4 sub-CUs and adding a penalty cost for the split.

We have experimentally determined the penalties with the objective of preserving the coding efficiency. For a PU with a single partition, 3 bits of penalty are used, while 8 bits are used for a PU with two partitions. A penalty of 1 bit, denoted $mode.R_{\text{Split}}$, is added when a CU is split, in order to evaluate sub-CUs. Finally, the complexity of the CU-MIN-COST function is low since the computation of J_{min} reuses the MV candidate list and the pre-computed prediction errors created by the motion propagation algorithm during the CTU initialization.

B. Candidate Modes Selection

The CANDIDATES-MODES function called by the CU-PROCESS returns a list of PU modes to be evaluated by

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

the current CU. The modes are returned in the following order: inter modes, from finer to coarser partitions; merge/skip modes, sorted in ascending order based on their J_{PM} cost; and intra modes, from coarser to finer partitions. All inter modes are disabled when the collocated H.264 region only has intra modes. Moreover, when the H.264 region has no residual, finer partitioning than H.264 is disabled in HEVC since the partitioning structure is already efficient in H.264. All these rules are applied when the fast mode decision (FMD) option is enabled (see section V).

When the ultra fast mode decision (UFMD) method is activated, more modes are removed by applying the following rules:

- Intra modes are disabled when the collocated H.264 region has no intra MB, as proposed by other authors [10], [23].
- The AMP modes are disabled since they are time-consuming and have little influence on the coding efficiency, as observed by other authors [9], [10], [12].

The use of the UFMD method is only recommended when an extra speed-up, in addition to the one obtained by the other proposed approaches is desired, since it comes with an additional RD cost penalty.

C. *Early Termination of the RD Cost Calculation*

The motion propagation algorithm presented in section III selects, at a low computational cost, the MVs for a PU mode. However, the processing of such mode is still a complex task in the HM because the full RD cost calculation requires the performance of highly complex operations, such as recursive transforms and entropy coding [24]. For an intra CU, an important part of the complexity is also due to the RD cost computation. However, the rough mode decision (RMD) method implemented in the HM reduces this complexity by selecting a subset of prediction candidate modes to be evaluated by the full RD cost calculation. The selection of the subset modes is based on a low complexity cost defined as:

$$J_{\text{PM}} = (\text{SATD}_{\text{luma}}) + \lambda_{\text{pred}} \cdot R_{\text{pred}} \tag{16}$$

where SATD is the sum of absolute Hadamard transformed coefficients of the prediction error, λ_{pred} is equal to $\sqrt{\lambda_{\text{mode}}}$ and R_{pred} is the number of bits needed to encode prediction information.

This method reduces the complexity of the RD cost computation for an intra mode. However, this reduction is limited since information on other modes is ignored. Moreover, the J_{PM} cost is unexploited by inter modes. To resolve these issues, the proposed mode decision process divides the mode processing into two parts: the PU-PM-COST (line 13 in Fig. 8) and the PU-RD-COST (line 15 in in Fig. 8). The first part computes the low complexity cost, J_{PM}, and the second, the high-complexity RD cost, J_{RD}, only for a subset of candidates.

For an inter mode, the J_{PM} corresponding to the cost returned by the motion propagation algorithm and for an intra mode, is the best J_{PM} computed by the RMD method. The PU-RD-COST function is only computed when the following

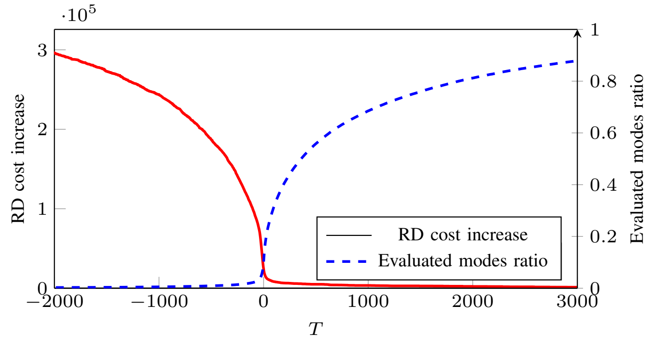


Fig. 12: Impact of threshold T on the total RD cost increase and the ratio of evaluated modes for the Racehorses sequences encoded with a QP set to 22.

criterion is true (otherwise, the mode evaluation is early terminated):

$$J_{\text{PM}}^{\text{Current}} < (J_{\text{PM}}^{\text{Best}} + T), \text{where } T \geq 0 \tag{17}$$

where $J_{\text{PM}}^{\text{Current}}$ is the low complexity cost of the currently processed mode and $J_{\text{PM}}^{\text{Best}}$, is the low complexity cost of the best mode so far. The threshold T controls the trade-off between the coding efficiency and computational complexity of the transcoder. For our experiments, it is empirically set to $3 \times \lambda_{\text{pred}}$. Fig. 12 shows how the threshold T influences: 1) the ratio between the number of modes for which the J_{RD} cost is evaluated and the total number of modes 2) the increase of the total RD cost error when a mode is not tested but has a better J_{RD} cost. We can see that the best compromise is when $T \approx 0$.

The early termination criterion can only be applied when the compared modes have the same size. In the pre-order traversal, the usage of this criterion is limited since it is impossible to compare a CU with a sub-CU. However, the proposed post-order traversal allows this criterion to be applied on the combination of 4 sub-CUs (the J_{PM} obtained by the recursive process of Fig. 8) and the current PU candidate (the J_{PM} computed on line 13 of Fig. 8).

Finally, the mode evaluation is also early terminated when an inter PU comprises two partitions having the same MV. This condition assumes that a coarser PU, with only one partition, will encode the same MV at a lower cost during the mode decision process.

V. EXPERIMENTAL RESULTS

In our experiments, the input stream is generated by the H.264 JM Reference Software 18.2 using the Baseline profile and the fast full search motion estimation algorithm. The proposed transcoding approach is compared to a CPDT approach. Both transcoders are based on the HEVC HM Reference Software 12.1, and use a *low delay P setting*. The CPDT transcoder employs the TZS search, a fast search motion estimation algorithm. We evaluate two reference frame structures: an non-hierarchical IPPP structure with one reference frame, denoted IPPP1; and an non-hierarchical IPPP structure with 4 (short-term) reference frames, denoted IPPP4. The H.264 sequences are generated to have the same number of reference frames as the one used by the transcoding simulations.

We ran the simulations on the recommended classes B to D sequences from the common test conditions [25]. The test sequences were fully encoded and transcoded with a QP set to 22, 27, 32 and 37. The impact on the coding efficiency was measured by using the Bjøntegaard Delta-Rate (BD-Rate) [26]. The computational complexity reduction was measured as the speed-up between the HEVC encoding time of the proposed transcoder and the CPDT transcoder. The H.264 decoding time was not considered, but it would have had little impact on the results.

Table II summarizes the performances of the proposed approach relative to the CDPT transcoder. Three configurations of the proposed transcoder are compared: i) the proposed transcoder with only the motion propagation enabled, denoted MP; ii) the complete transcoder with the FMD method enabled, denoted MP+FMD; and iii) the complete transcoder with the UFMD method enabled, denoted MP+UFMD.

A. Comparison of the Transcoding Configurations

As shown in Table II, the proposed motion propagation algorithm (motion propagation (MP) only) reduces the encoding time and tends to preserve or improve the coding efficiency, especially when one reference frame is used. The impact on the BD-Rate demonstrates that the proposed motion propagation algorithm is precise enough to replace a motion refinement algorithm in the proposed transcoding context. Indeed, in many cases, the motion propagation algorithm outperforms the HM, executing a full re-encoding, as shown by the negative BD-Rates. These slight coding efficiency improvements are caused by different factors, such as the chroma components considered in the motion cost function and the higher number of quarter-pel positions evaluated by the proposed algorithm. Results also show that the speed-up increases significantly when a higher number of reference frames is used at the cost of a slight increase of BD-Rate.

When the MP+FMD configuration is used, the speed-up increases dramatically and the coding efficiency is still reasonable. The MP+UFMD increases more the speed-ups, but at at the cost of a greater impact on the coding efficiency. Usage of this latter configuration is only recommended when the execution time is crucial.

B. Comparison With Related Transcoding Works

The proposed transcoder with the novel motion-based split decision method outperforms related transcoding works in terms of speed-up, and has a low impact on the coding efficiency. For the IPPP1 reference frame structure, the MP+FMD simulations achieves an average speed-up of 8.5x and average BD-Rate of 2.63%. In contrast, the fast transcoder based on content modeling and early termination (CM+ET) proposed by [5] Peixoto et al. achieve an average speed-up of 3.83x and an average BD-Rate of 7.56% [5] for similar test conditions. Table III presents a comparison between our MP+FMD approach, our previous approach [15] and the CM-ET approach [5] for four sequences. Results show our MP+FMD approach increases the speed-up with a low impact on the coding efficiency compared to our previous work. Moreover, we achieve better results than the CM-ET for all compared sequences.

For an IPPP4 reference frame structure, the MP+FMD simulations achieve an average speed-up of 11.77x and average BD-Rate of 3.82%. In comparison, the fast transcoder based on region feature analysis (RFA) proposed by Jiang et al. achieves an average speed-up of 2.0x for an average BD-Rate of 1.73% [4] for similar test conditions. Several sequences are compared in Table IV.

Shen et al. proposed a parallel transcoder with SIMD acceleration that achieves a speed-up of up to 70x [10]. However, the speed-up decreases between 2 and 10x when the SIMD acceleration and parallel processing are disabled, and the coding efficiency is largely affected (the authors didn't measure the average BD-Rate).

TABLE II: Performances of the proposed transcoder compared to a CPDT transcoder. Three transcoding configurations and two refrence frame structures are evaluated

		Reference frame structure			
		IPPP1		IPPP4	
Sequence	Method	Speed-Up	BD-Rate	Speed-Up	BD-Rate
BasketballDrive	MP	1.48	-0.23	2.71	-0.38
1920×1080	MP+FMD	6.42	4.36	11.63	2.99
(Class B)	MP+UFMD	6.90	7.29	12.28	8.07
BQTerrace	MP	1.45	-0.96	2.11	-0.44
1920×1080	MP+FMD	9.62	1.44	12.65	6.20
(Class B)	MP+UFMD	11.15	1.93	15.54	6.68
Cactus	MP	1.40	0.54	2.23	0.85
1920×1080	MP+FMD	8.26	4.64	12.25	5.62
(Class B)	MP+UFMD	9.36	7.85	14.32	8.24
Kimono	MP	1.39	0.96	2.44	1.26
1920×1080	MP+FMD	6.79	5.86	11.08	5.93
(Class B)	MP+UFMD	7.76	6.27	12.97	5.93
ParkScene	MP	1.44	0.70	2.18	1.74
1920×1080	MP+FMD	8.90	2.56	11.97	3.75
(Class B)	MP+UFMD	10.46	3.87	14.91	4.95
BasketballDrill	MP	1.39	-0.66	2.30	0.08
832×480	MP+FMD	8.36	3.34	13.66	5.26
(Class C)	MP+UFMD	10.21	6.33	15.86	8.25
BQMall	MP	1.38	-0.34	2.14	0.16
832×480	MP+FMD	9.11	2.89	13.40	4.17
(Class C)	MP+UFMD	11.27	5.55	16.18	6.39
PartyScene	MP	1.34	-0.15	1.95	0.44
832×480	MP+FMD	8.83	2.00	12.06	3.11
(Class C)	MP+UFMD	10.91	3.27	14.27	4.41
RaceHorses	MP	1.38	-0.99	2.34	-0.46
832×480	MP+FMD	7.41	1.00	12.57	1.72
(Class C)	MP+UFMD	9.12	2.94	14.52	3.53
BasketballPass	MP	1.35	0.09	2.00	0.08
416×240	MP+FMD	8.44	2.33	10.49	2.77
(Class D)	MP+UFMD	9.38	4.96	12.22	5.12
BQSquare	MP	1.35	0.17	1.77	1.01
416×240	MP+FMD	10.80	0.78	11.10	2.39
(Class D)	MP+UFMD	13.03	1.60	14.22	3.69
BlowingBubbles	MP	1.35	0.38	1.83	1.48
416×240	MP+FMD	9.11	2.17	10.01	3.94
(Class D)	MP+UFMD	10.49	3.89	12.09	5.45
RaceHorses	MP	1.33	-0.87	2.02	-0.25
416×240	MP+FMD	8.48	0.87	10.17	1.75
(Class D)	MP+UFMD	9.25	3.18	12.14	3.84
	MP	1.39	-0.10	2.16	0.45
Average	MP+FMD	8.50	2.63	11.77	3.82
	MP+UFMD	9.92	4.53	13.80	5.91

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2754491, IEEE Transactions on Circuits and Systems for Video Technology

TABLE III: Comparisons of the proposed transcoder, our previous work [15], and the CM+ET approach [5], for a *low delay P* setting with one reference on 416×240 (Class B) sequences

	Proposed MP+FMD		Previous work [15]		CM-ET [5]	
Sequence	Speed-Up	BD-Rate	Speed-up	BD-Rate	Speed-Up	BD-Rate
Kimono1	6.79	5.86	6.20	5.73	3.88	4.54
ParkScene	8.90	0.88	7.27	1.57	4.55	6.39
Cactus	8.26	4.64	6.66	3.50	4.51	8.95
BasketballDrive	6.42	4.36	5.80	4.44	3.70	6.57
Average	**7.59**	**3.96**	**6.49**	**3.81**	**4.16**	**6.61**

TABLE IV: Comparisons of the proposed transcoder and the RFA approach [4], for a *low delay P* setting with 4 reference frames

		Proposed MP+FMD		RFA [4]	
Resolution	Sequence	Speed-Up	BD-Rate	Speed-up	BD-Rate
1920×1080 (Class B)	Kimono1	11.08	5.93	2.00	0.01
	ParkScene	11.97	3.75	1.91	0.58
	Cactus	12.25	5.62	1.90	0.05
	BasketballDrive	11.63	2.99	1.99	0.03
832×480 (Class B)	RaceHorses	12.57	1.72	2.01	0.65
	BQMall	13.40	4.17	1.93	0.91
	PartyScene	12.06	3.11	1.82	3.45
	BasketballDrill	13.66	5.26	1.98	0.03
416×240 (Class B)	RaceHorses	10.17	1.75	2.02	2.79
	BQSquare	11.10	2.39	1.78	4.55
	BlowingBubbles	10.01	3.94	1.71	4.38
	BasketballPass	10.47	2.77	1.99	1.87
Average		**11.70**	**3.62**	**1.92**	**1.61**

As several related works, the proposed approach doesn't support video transcoding scnenarios with B-frames. An example of an approach supporting B-frames have been proposed by Peixoto et al [27]. The authors claim a speed-up between 1.97 and 2.91, and a BD-Rate between 1.57% and 9.83%.

C. Comparison With Fast Mode Decision Approaches

We compared the results of our MP+FMD approach with two fast encoding approaches employed in a transcoding context: the fast HM (FHM) and an efficient mode decision schema (EMDS) proposed by Vanne et al. [28]. The FHM is the HM 12.1 configured to use the adaptative motion search range (ASR) and three fast mode decision algorithms already present in the HM: the early CU (ECU) [29], the early skip detection (ESD) [30] and the coded block flag fast mode (CFM) [31]. The EMDS is a fast mode decision approach that uses some heuristics to determines which symmetric motion partitions (SMPs) and AMPs must be evaluated. The authors present several schemas offering different trade-offs between speed-up and coding efficiency. For our simulation, we implemtend in the HM 12.1 the scheme identified as S_{28} by the authors. This schema favors speed-up over coding efficiency. Finaly, we have combined the FHM and EMDS in an approach called FHM+EMDS.

Results for IPPP1 and IPPP4 reference frame structures are presented respectively in Table V and Table VI. Compared to the FHM-EMDS, the MP+FMD is about 3 to 5x faster and acheives better BD-Rate for several sequences. Compared to the FHM and EMDS, our method is about 4 to 8x faster with a reasonnable BD-Rate. So, the performance of our approach are especially interresting for a real-time transcoding context, where the encoding speed is more important than the BD-Rate.

VI. Conclusion

In this paper, we proposed a fast H.264-to-HEVC transcoding approach combining a motion propagation algorithm and a fast mode decision framework. The motion propagation algorithm requires no motion refinement, eliminates motion estimation computational redundancy, and has a very low complexity at the PU level. The mode decision framework is based on a post-order traversal of the CTU. It includes some candidate mode reduction techniques and a criterion to early terminate the RD cost computation. Finally, the structural split decision developed in our previous work was advantageously replaced by a motion-based split decision method.

Our experiments show that the proposed transcoder is much faster than related works, and achieves a coding efficiency similar to a CPDT transcoder. Moreover, our novel split decision method outperforms our previous method.

In our future research, we intend to adapt the proposed transcoder to other coding applications, such as multi-rate HEVC video coding. Furthermore, we intend to improve the transcoding performance of intra modes by replacing the RMD method with a faster method, such as the method proposed by Jamali et al. [32]. Finally, we plan to improve the proposed approach by supporting B-frames.

References

[1] ISO/IEC JTC 1/SC 29/WG 11, "High efficiency video coding," 2013.

[2] ITU-T SG16 Q.6 and ISO/IEC JTC 1/SC 29/WG 11, "ITU-T recommendation H.264: Advanced video coding for generic audiovisual services," 2003.

[3] J.T. Fang, Z.Y. Chen, T.L. Liao, and P.C. Chang, "A fast PU mode decision algorithm for H.264/AVC to HEVC transcoding," *Computer Science & Information Technology*, pp. 215–225, 2014.

[4] W. Jiang, Y. Chen, and X. Tian, "Fast transcoding from H.264 to HEVC based on region feature analysis," *Multimedia Tools and Applications*, pp. 1–22, 2013.

[5] E. Peixoto, B. Macchiavello, E.M. Hung, and R.L. de Queiroz, "A fast HEVC transcoder based on content modeling and early termination," *21st IEEE International Conference on Image Processing (ICIP 2014)*, pp. 1–5, 2014.

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

[7] W. Jiang and Y.W. Chen, "Low-complexity transcoding from H.264 to HEVC based on motion vector clustering," *Electronics Letters*, vol. 49, no. 19, pp. 1224–1226, 2013.

[8] C. Zong-Yi, Tseng C.T., and C. Pao-Chi, "Fast inter prediction for H.264 to HEVC transcoding," in *3rd International Conference on Multimedia Technology (ICMT-13)*. Atlantis Press, 2013, pp. 1301–1308.

[9] R. Luo, R. Xie, and L. Zhang, "Fast AVS to HEVC transcoding based on ROI detection using visual characteristics," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 1–6.

[10] T. Shen, Y. Lu, Z. Wen, L. Zou, Y. Chen, and J. Wen, "Ultra fast H.264/AVC to HEVC transcoder," in *Data Compression Conference (DCC), 2013*. IEEE, 2013, pp. 241–250.

[11] Z.-Y. Chen, J.-T. Fang, T.-L. Liao, and P.-C. Chang, "Efficient PU mode decision and motion estimation for H.264/AVC to HEVC transcoder," *Signal & Image Processing: An International Journal (SIPIJ)*, vol. 5, no. 2, pp. 81–93, Apr 2014.

[12] P. Xing, Y. Tian, X. Zhang, Y. Wang, and T. Huang, "A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos," in *Visual Communications and Image Processing (VCIP), 2013*. IEEE, 2013, pp. 1–6.

1051-8215 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

TABLE V: Comparisons of the proposed transcoder and FHM, EMDS and FHM+EMDS for a *low delay P* setting with 1 reference frame

		Proposed MP+FMD		FHM		EMDS [28]		FHM+EMDS	
Resolution	Sequence	Speed-Up	BD-Rate	Speed-up	BD-Rate	Speed-up	BD-Rate	Speed-up	BD-Rate
1920×1080 (Class B)	Kimono1	6.79	5.86	1.54	0.61	1.53	1.41	2.04	2.77
	ParkScene	8.90	2.56	2.08	3.71	1.51	2.36	2.77	6.53
	Cactus	8.26	4.64	1.89	3.41	1.58	2.25	2.32	6.11
	BasketballDrive	6.42	4.36	1.74	2.76	1.54	2.29	2.30	4.46
832×480 (Class C)	RaceHorses	7.41	1.01	1.29	1.39	1.40	2.42	1.62	3.89
	BQMall	9.11	2.89	1.74	2.15	1.48	2.98	2.24	5.34
	PartyScene	8.83	2.02	1.24	1.97	1.36	2.54	1.62	4.57
	BasketballDrill	8.36	3.34	1.69	2.68	1.51	2.58	2.17	5.43
Average		**7.12**	**2.96**	**1.47**	**2.08**	**1.32**	**2.09**	**1.90**	**4.34**

TABLE VI: Comparisons of the proposed transcoder and FHM, EMDS and FHM+EMDS for a *low delay P* setting with 4 reference frames

		Proposed MP+FMD		FHM		EMDS [28]		FHM+EMDS	
Resolution	Sequence	Speed-Up	BD-Rate	Speed-up	BD-Rate	Speed-up	BD-Rate	Speed-up	BD-Rate
1920×1080 (Class B)	Kimono1	11.08	5.93	1.53	0.61	1.75	1.40	2.39	2.77
	ParkScene	11.97	3.75	1.53	1.49	1.75	2.07	2.39	3.12
	Cactus	12.25	5.62	1.77	4.03	1.65	2.39	2.55	6.91
	BasketballDrive	11.63	2.99	1.77	2.22	1.82	1.81	2.80	4.26
832×480 (Class C)	RaceHorses	12.57	1.72	1.31	1.30	1.59	2.33	1.91	3.68
	BQMall	13.40	4.17	1.69	2.18	1.59	3.23	2.33	5.28
	PartyScene	12.06	3.11	1.34	2.19	1.48	2.65	1.82	4.89
	BasketballDrill	13.66	5.26	1.63	2.62	1.63	1.84	2.35	5.11
Average		**12.33**	**4.06**	**1.61**	**2.05**	**1.65**	**2.34**	**2.34**	**4.61**

[13] H. Yuan, C. Guo, J. Liu, X. Wang, and S. Kwong, "Motion-homogeneous based fast transcoding method from h.264/avc to hevc," *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, 2017.

[14] A. J. Díaz-Honrubia, J. L. Martínez, P. Cuenca, J. A. Gamez, and J. M. Puerta, "Adaptive fast quadtree level decision algorithm for H.264 to HEVC video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 154–168, Jan 2016.

[15] J-F. Franche and S. Coulombe, "Fast H.264 to HEVC transcoder based on post-order traversal of quadtree structure," *22nd IEEE International Conference on Image Processing (ICIP 2015)*, pp. 1–5, 2015.

[16] K. McCann, K. Sugimoto, B. Bross, W-J. Han, and G.J Sullivan, "High efficiency video coding (HEVC) test model 12 (HM 12) encoder description," Tech. Rep. October, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Vienna, Austria, 2013.

[17] G.J. Sullivan, J. Ohm, W.J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[18] H. Lv, R. Wang, X. Xie, H. Jia, and W. Gao, "A comparison of fractional-pel interpolation filters in HEVC and H.264/AVC," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*. IEEE, 2012, pp. 1–6.

[19] J. Bialkowski, M. Barkowsky, and A. Kaup, "Overview of low-complexity video transcoding from H.263 to H.264," in *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006, pp. 49–52.

[20] W-N. Chen and H-M. Hang, "H.264/AVC motion estimation implmentation on compute unified device architecture (CUDA)," in *Multimedia and Expo, 2008 IEEE International Conference on*. IEEE, 2008, pp. 697–700.

[21] Y. Gao and J. Zhou, "Motion vector extrapolation for parallel motion estimation on GPU," *Multimedia tools and applications*, vol. 68, no. 3, pp. 701–715, 2014.

[22] Franklin C Crow, "Summed-area tables for texture mapping," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 207–212, 1984.

[23] E. Peixoto and E. Izquierdo, "A complexity-scalable transcoder from H.264/AVC to the new HEVC codec," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*. IEEE, 2012, pp. 737–740.

[24] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1685–1696, 2012.

[25] F. Bossen, "Common test conditions and software reference configurations Output," *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG1*, vol. JCTVC-L110, no. 12th Meeting: Geneva, pp. 1–10, 2013.

[26] G. Bjøntegaard, "Document VCEG-M33: Calculation of average PSNR differences between RD-curves," in *ITU-T VCEG Meeting, Austin, Texas, USA, Tech. Rep*, 2001, pp. 2–4.

[27] Eduardo Peixoto, Tamer Shanableh, and Ebroul 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, 2014.

[28] Jarno Vanne, Marko Viitanen, and Timo D Hämäläinen, "Efficient mode decision schemes for hevc inter prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1579–1593, 2014.

[29] K Choi, S.H. Park, and E.S Jang, "Coding tree pruning based CU early termination," *JCTVC-F092, Torino, Italy*, 2011.

[30] J Yang, J Kim, K Won, H Lee, and B Jeon, "Early SKIP detection for HEVC," *JCTVC-G543, Geneva, Switzerland*, 2011.

[31] R.H Gweon and Y.L Lee, "Early termination of cu encoding to reduce hevc complexity," *JCTVC-F045, Torino, Italy*, 2011.

[32] M. Jamali, S. Coulombe, and F. Caron, "Fast HEVC intra mode decision based on edge detection and SATD costs classification," in *Data Compression Conference (DCC), 2015*, April 2015, pp. 43–52.

Jean-François Franche received the B.Eng. degree in IT engineering, the M.Sc. degree, and the Ph.D. degree from École de technologie supérieure, Montreal, QC, Canada, in 2008, 2011 and 2016, respectively. He is a Software Developer with Vantrix Corporation, Montreal, operating on H.264/AVC, H.265/HEVC and video cameras.

Stéphane Coulombe (S'90-M'98-SM'01) received a B.Eng. degree in Electrical Engineering from École Polytechnique de Montréal, Canada, in 1991, and a Ph.D. degree in Telecommunications (image processing) from INRS-Telecommunications, Montreal, in 1996. He is currently a Professor in the Software and IT Engineering Department, École de technologie supérieure (ÉTS is a constituent of the Université du Québec network). From 1997 to 1999, he was with Nortel Wireless Network Group in Montreal, and from 1999 to 2004, he worked with the Nokia Research Center, Dallas, TX, as Senior Research Engineer and as a Program Manager in the Audiovisual Systems Laboratory. He joined ÉTS in 2004, where he currently carries out research and development on video processing and systems, compression, and transcoding. Since 2009, he has held the Vantrix Industrial Research Chair in Video Optimization.