Transcoding from Hybrid Nonscalable to Wavelet-Based Scalable Video Codecs

Eduardo Peixoto, *Student Member, IEEE,* Toni Zgaljic, and Ebroul Izquierdo, *Senior Member, IEEE*

***Abstract*—Scalable video coding (SVC) enables low complexity adaptation of compressed video, providing an efficient solution for content delivery through heterogeneous networks and to diverse displays. However, legacy video and most commercially available content capturing devices use conventional nonscalable coding, e.g., H.264/AVC. This paper proposes an efficient transcoder from H.264/AVC to a wavelet-based SVC. It aims at exploiting the advantages offered by fine granularity SVC technology when dealing with conventional coders and legacy video. The proposed transcoder was developed to cope with important functionalities of H.264/AVC, such as flexible reference frame (RF) selection. It is able to work with different coding configurations of H.264/AVC, including *IPP* or *IBBP* with multiple RFs. Moreover, many of the techniques presented in this paper are generic in the sense that they can be used for transcoding with many popular wavelet-based and hybrid-based video coding architectures. To reduce the transcoder's complexity, motion information and residual data extracted from a compressed H.264/AVC stream are exploited. Experimental results show a very good performance of the proposed transcoder in terms of decoded video quality and system complexity.**

***Index Terms*—Scalable video coding (SVC), transcoding.**

I. Introduction

SCALABLE video coding (SVC) allows real-time content adaptation since the extraction of a lower resolution, frame-rate, and/or quality video can be conducted by a simple parsing of the compressed bitstream. However, video stored in the server is often encoded using conventional codecs, such as nonscalable H.264/AVC [1]. In this case, an efficient low-complexity video adaptation cannot be achieved. To address this issue, a nonscalable bitstream can be converted into a scalable stream by transcoding. Here, transcoding would be required only once and, afterward, the scalable stream can be adapted many times.

The most straightforward way to transcode between formats is to cascade the required decoder and encoder, known as the cascaded pixel-domain transcoder [2], [3]. It results in high-quality compressed video, but at an elevated computational cost, which renders this approach unfeasible for many important applications. To reduce complexity, one could use the motion information [modes, partitions, motion vectors (MVs), and reference frames (RFs)] gathered from the source stream to speedup the transcoding process, avoiding a costly full motion estimation (ME). Here, we define the trivial transcoder when no intermediate processing is performed.

Transcoding between hybrid-based video coding structures has been extensively studied before [2], [3], even targeting hybrid-based scalable codecs [4]. However, hybrid-based to wavelet-based scalable video transcoding has not been fully investigated in the literature. Although a hybrid-based approach was chosen for standardization of SVC within MPEG [5], concurrently, a significant amount of research has also been carried out on wavelet-based scalable video coding (W-SVC). Several W-SVC systems proposed recently have shown a very good performance in different types of application scenarios [6]–[12], while still being able to deliver some attractive features not supported by the standard, such as fine grain scalability (FGS) and superior performance at specific coding and decoding rates.

The codec used for the research reported in this paper, denoted simply as W-SVC in the sequel, is an extended and improved version of the encoder reported by Sprljan *et al.* [6]. It supports quality (with FGS), spatial (resolution) and temporal scalability, and any combination of them. Its main features include hierarchical variable size block matching ME supporting MV scalability [13], flexible selection of filters for both spatial and temporal wavelet transforms at each level of spatiotemporal (ST) decomposition [14], [15], user-defined flexible decomposition path [14], [15], support for conventional frame-based coding and object-based coding, motion adaptive spatial filtering [16], bitplane coding based on embedded zero block coding (EZBC) with binary arithmetic coding, and low-complexity post compression rate-distortion optimization (RDO) for bitstream allocation [17].

In this paper, we build on our previous work [18], [19], improving the way motion information coming from the H.264/AVC bitstream is exploited when transcoding to W-SVC. From here on, H.264/AVC refers to the nonscalable part of the H.264/AVC recommendation [20] (thus without [5, Annexure G]). Since H.264/AVC supports RFs selection in a very flexible way, not all MVs can be directly reused in the targeted W-SVC. Thus, missing MVs are obtained by low-complexity approximation and refinement. An additional key issue is that the motion information from H.264/AVC is optimized for a

Manuscript received December 5, 2010; revised April 9, 2011 and June 20, 2011; accepted August 13, 2011. Date of publication September 15, 2011; date of current version April 2, 2012. This work was supported in part by the European Commission, under Contracts FP7-247688 3DLife and FP7-248474 SARACEN. This paper was recommended by Associate Editor P. Yin.

The authors are with the School of Electronic Engineering and Computer Science, Queen Mary, University of London, London E1 4NS, U.K. (e-mail: eduardopeixoto@ieee.org; toni.zgaljic@elec.qmul.ac.uk; ebroul.izquierdo@elec.qmul.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2168175

single rate-distortion point, and therefore it has to be modified to support the addressed scalability functionality. The proven high performance of the proposed transcoder is obtained by efficient approximation of extracted MVs. Here, a flexible MV composition algorithm is introduced that is able to cope with different coding configurations in the H.264/AVC stream, such as *IPP*, *IBBP*, and *hierarchical* configurations. Furthermore, the transcoder efficiently exploits the similarity of H.264/AVC MVs and the information about the presence of discrete cosine transform (DCT) coefficients in a H.264/AVC block to greatly reduce the transcoder complexity at negligible influence on decoded video quality.

The proposed transcoder is generic in the sense that it can be used with most of the well-known wavelet-based coding architectures [8], [9], [12], particularly with any architecture that uses motion-compensated temporal filtering (MCTF) [7], [21] at the same ST resolution as the source codec. Furthermore, the transcoder framework, presented in Section III, can be used with any cascaded pixel domain transcoder, regardless if they are hybrid or wavelet codecs. Specially, the MV approximation algorithms presented here could be easily adapted to any transcoder where RF mismatch is an issue [22]–[27], particularly the MV composition presented in Section III-C4. For instance, the coding structure between the source and the target codecs may be different either due to format restrictions (some codecs or profiles cannot handle *B*-frames, or impose RF restrictions [25], [27]), or because some feature is desired, such as temporal scalability [26] (this applies even in transcoding between H.264/AVC and its scalable extension). Therefore, the transcoder is rather based on an open and easily adaptable model since the key strategies proposed for the adaptation of motion information are independent from the W-SVC coder and its underpinning algorithms.

The remaining of this paper is structured as follows. Section II presents the W-SVC codec used. Section III presents the proposed transcoder, while Section IV presents the experiments and results made to evaluate the transcoder, and Section V concludes this paper.

II. W-SVC Codec

The W-SVC codec [6], [14], [16] used in this paper is a scalable codec that applies a discrete wavelet transform, MCTF [7], [21] and embedded coding [17] to achieve FGS. The architecture of this codec is different than that of H.264/AVC and other hybrid codecs. Even techniques that are present in both codecs, like ME and entropy coding, are applied differently. This section explains the basic structure of the utilized W-SVC codec and details its ME module, since it will play a fundamental role in the transcoding. On the other hand, the H.264/AVC codec is an industry standard [20] and it has been extensively reported in the literature [1], [28], and therefore it is not detailed here.

A. W-SVC Codec Architecture

A high-level block diagram of the coding architecture utilized in the W-SVC codec is shown in Fig. 1. First, the input video is subjected to MCTF [7], [21], which aims to reduce the correlation between consecutive frames and provide a basis for temporal scalability, avoiding drift problems. Temporally, decorrelated frames are subjected to spatial decomposition, which reduces the correlation in the spatial direction and provides a basis for spatial scalability. This particular coding architecture is called the *t*+2-D architecture [14] since spatial decomposition is preceded by temporal decomposition (TD). Different decomposition orders, such as 2-D+*t* and 2-D+*t*+2-D architectures [14], can also be used in the codec. Here, the focus will be on the *t*+2-D architecture, since it is the simplest of all architectures and provides the best performance in case of temporal and quality scalability [29]. In the W-SVC codec, MCTF is implemented via lifting scheme [30]. The purpose of the ME module is to estimate the motion between frames so that the temporal filtering can be performed in the direction of MVs. Transforms used for spatial and TD at an arbitrary decomposition level are defined in the encoding parameters. Many different transforms are supported, such as the 9/7, 5/7, 5/3, and 1/3 among others [31]. In this paper, we have used the 5/3 filter bank for TD and the 9/7 filter bank for spatial decomposition.

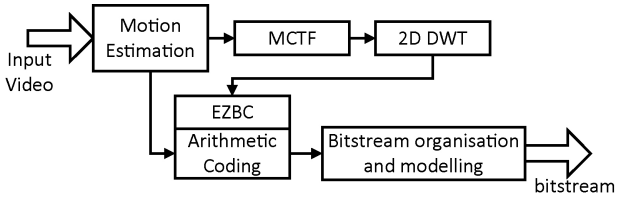

Fig. 1. Framework of the W-SVC codec when using *t*+2-D configuration.

The texture coding is performed by EZBC [32], which encodes wavelet coefficients in an embedded manner in order to remove the redundancies remaining after the wavelet decomposition and provide the basis for quality scalability. Finally, the resulting data are mapped into the scalable stream in the bitstream organization module, creating a layered representation of the compressed data [33], which can then be extracted at the desired ST resolution and quality.

B. ME in W-SVC

In the W-SVC codec, the macroblock (MB) size itself is an encoder parameter, which should be a power of two. Another encoder parameter is the maximum number of partitioning levels allowed (i.e., the depth of the quadtree). The MB is partitioned in a recursive way: if the MB size is $n \times n$ pixels, and the parameters allow for k partitioning levels, then at the first level the MB will have four blocks of $\frac{n}{2} \times \frac{n}{2}$ pixels. Each of these blocks could be further partitioned (provided $k > 1$). The minimum block size will be $\frac{n}{2^k} \times \frac{n}{2^k}$. This flexible scheme is particularly useful if spatial scalability is used.

The MV precision is also an encoder parameter. While the most commonly used precision is quarter-pixel, other precisions can be used, such as 1/2 and 1/8. Although 1/8 precision may result in worse compression performance than 1/4, due to the overhead in MV information that needs to be transmitted [29], it is useful in the codec when spatial scalability is applied.

TABLE I
PROFILE OF THE CHOSEN PARTITIONS FOR THE H.264/AVC AND W-SVC CODECS

	Soccer		*Crew*		*City*		*Harbor*	
Partition Size	H.264 (%)	W-SVC 3 TD	H.264 (%)	W-SVC 3 TD	H.264 (%)	W-SVC 5 TD	H.264 (%)	W-SVC 6 TD
INTRA	5.28	12.66%	21.16	12.66%	0.45	3.33%	1.09	1.67%
16 × 16	39.33	76.12%	23.86	77.78%	46.50	88.98%	29.03	90.71%
16 × 8 or 8 × 16	15.43	N/A	22.31	N/A	12.92	N/A	16.58	N/A
8 × 8	19.53	10.12%	16.42	9.00%	21.76	6.86%	27.37	6.96%
8 × 4 or 4 × 8	17.18	N/A	14.39	N/A	15.18	N/A	22.72	N/A
4 × 4	3.25	1.08%	1.89	0.54%	3.18	0.83%	3.20	0.66%

The number of TDs used in W-SVC is shown for each sequence, and the H.264/AVC sequences were encoded with *IPP* configuration and QP = 20.

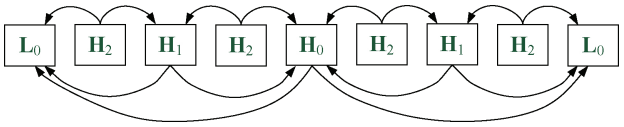

Fig. 2. Hierarchical structure for selecting the RF with three levels of TD.

C. Selection of RFs

Due to the dyadic TD utilized in the W-SVC, there are rigid limitations on the RFs that the block matching procedure in ME can use. Each motion block may use up to two RFs, one past and one future frame. This is depicted in Fig. 2, which shows a three-level dyadic TD of nine frames of the input video, using MCTF with bidirectional prediction. The markings $\mathbf{L_x}$ and $\mathbf{H_x}$ in the figure represent low-pass and high-pass temporal subbands, respectively, at the xth decomposition scale, and the arrows point from the frames being predicted to the RFs in the process of MCTF. The RF structure is fixed at each decomposition scale—frames at higher scales are never used as reference at lower scales. This strict condition straightforwardly enables temporal scalability.

Depending on wavelet filters chosen for TD, each block can choose between four modes: intra, forward, backward, and bidirectional prediction. It is important to note that RFs used in ME are the original frames (or frames resulting from a previous TD), and not the locally decoded frames, as it is common with hybrid codecs.

D. RDO in ME

The optimization in the W-SVC codec is different than that of H.264/AVC. In both codecs, RDO consists of three main elements: 1) selecting an appropriate partitioning of a MB into smaller blocks; 2) selecting modes for each partition; and 3) selecting a MV for each partition (or two MVs if the prediction is bidirectional). However, in the W-SVC, the stream can be decoded at multiple ST resolutions and qualities, and the same set of partitioning, modes and MVs, is used for all decoding points (unless scalable motion [13] is used, but this case is not considered in this paper). Therefore, the optimization has to be tailored to a wide range of bitrates and ST resolutions. On the other hand, as others ITU-T and ISO/IEC standards, only the reference decoder is defined in the H.264/AVC standard itself [20]. The encoder implementation is free—the only constraint is that it has to generate a conforming bitstream. Thus, the optimization is dependent on the encoder that is used. However, encoder implementations generally try to optimize the rate-distortion for a fixed bitrate and ST resolution. Also, in addition to the optimization of partitioning and MVs, quantization parameters (QPs), the selection of transforms and other parameters can be considered in the optimization.

Although it is outside the scope of this paper, we briefly discuss the RDO procedure in H.264/AVC and W-SVC here. In both the W-SVC and H.264/AVC (considering the JM implementation), the cost of a MV is computed as $J = D+\lambda \cdot R$, where D is the distortion (measured as the sum of absolute differences, SAD, or as the sum of squared differences, or other measure) and R is the rate (or estimated rate) of transmitting this motion information. The difference in the optimization in the two codecs comes from the choice of the Lagrange multiplier λ, which controls the tradeoff between rate and distortion. In many H.264/AVC implementations, λ is a function of the QP [34], and so the tradeoff between rate and distortion is optimized for each bitrate. Thus, in the H.264/AVC codec, different QPs lead to different sets of MVs, since the optimization changes with the QP. For the W-SVC codec, on the other hand, the same set of MVs has to be used for all bitrates, in case of quality scalability, since the result of a W-SVC encoding is a single, scalable bitstream, that can be extracted at different qualities. Thus, the choice of λ has to consider all bitrates at a particular ST resolution. Due to the scalability, sometimes a small decrease in quality at high bitrates has to be traded for a large increase at low bitrates. The net effect of this optimization is that larger blocks are preferred over smaller ones, since the former usually yields better overall results. This is different than what usually happens in a H.264/AVC codec. To illustrate this difference, Table I shows the percentage of different block sizes chosen for four different sequences at CIF (352 × 288) resolution, both for H.264/AVC (using *IPP* configuration and a QP=20) and W-SVC codecs. It can be seen from the table that, indeed, larger block sizes are chosen more often in the W-SVC codec than in the H.264/AVC codec.

Fig. 3 further illustrates the effect of this difference in optimization. The sequence *Soccer* at CIF resolution was encoded with H.264/AVC with a three-level hierarchical configuration, with no intra MBs in inter-frames. This configurations allows for full reuse of the MVs in the W-SVC codec, since the RFs for all MVs match. The figure shows the performance of the

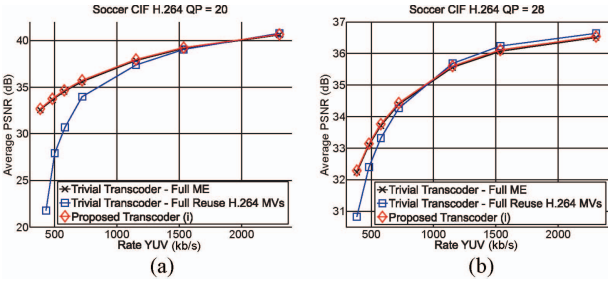

Fig. 3. Transcoder results for *Soccer* CIF sequence encoded with a three-level hierarchical configuration, with no intra MBs in inter-frames. The original H.264/AVC sequence uses (a) QP = 20 and (b) QP = 28. The full reuse of H.264/AVC MVs yields to a large performance loss in the W-SVC codec.

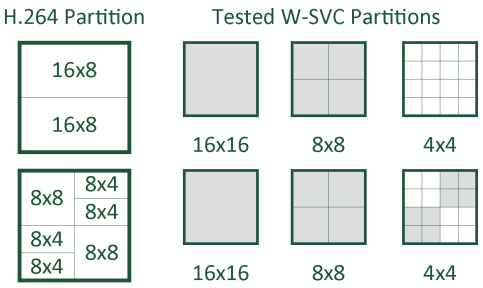

Fig. 4. Example of partitioning decision. The shaded partitions are the ones that will be tested.

trivial transcoder with full ME, a transcoder with full reuse of the MVs, and the proposed transcoder, which will be explained in Section III. The transcoder with full reuse means that the W-SVC uses the same set of MVs as the H.264/AVC stream. It can be seen from Fig. 3(a) and (b) that these sets of MVs work well in W-SVC for high bitrates, with the set of MVs for QP=28 even outperforming the W-SVC full ME by 0.1 dB. For low bitrates, however, the performance is very low, with more than 10 dB loss for the set of MVs for QP=20 and 1.5 dB loss for the one computed with QP=28. This happens because the W-SVC codec uses most of the bitrate for the motion information, and only a small part is left for the transform coefficients. As the bitrate increases, this balance is not so important, and the performance also increases, to the point where, at high bitrates, it can benefit from a more precise motion information. This effect has to be taken into account when developing the transcoder framework.

III. Transcoder

Since the two codecs are fundamentally different, the transcoder architecture chosen for this paper is the cascade pixel-domain approach [2], [3]. It consists of decoding the source H.264/AVC sequence, performing an intermediate processing on the decoded motion information, and re-encoding the sequence using the W-SVC codec and processed motion information. In this way, complexity is largely reduced, since full ME does not need to be performed, which is the most time-consuming task in a video encoder.

The two main issues that need to be addressed by the transcoder are the RF mismatch and the optimization issue. As it is common in heterogeneous transcoding, the transcoder tries to reuse the incoming H.264/AVC MVs as much as possible [35]. However, the different RF structure in the two codecs restricts direct reusing of H.264/AVC motion information in the W-SVC codec. Only those MVs that point to the same RF in W-SVC and H.264/AVC can be directly reused; those whose RF do not match have to be approximated and refined. The RF mismatch is a common problem in transcoding, specially from H.264/AVC to other codecs, like MPEG-2, H.263, and VC-1, and also in frame-rate reduction transcoding. The RF mismatch occurs when the MV for a given partition in the source codec does not point to the RF required by the target codec. A common solution to this problem is to use algorithms for MV composition [22] or MV scaling [36]. These algorithms attempt to follow the path of motion, composing a new MV that points to the required RF. However, MV scaling is not very accurate when the frames are further apart, and most algorithms for MV composition are not suited to tackle the complex motion structures that can be found in H.264/AVC streams. Furthermore, due to the optimization issue discussed in Section II-D, even the MVs that can be reused may not be optimal for W-SVC and, therefore, should also be refined.

To cope with these issues, an efficient framework for transcoding was developed, which reuses the information about the H.264/AVC partitioning and MVs, taking into account the optimization issue. Also, a novel and efficient MV composition algorithm, which is able to cope with complex motion structures, and new strategies to reduce the transcoder complexity were developed.

A. Handling the H.264/AVC Partition

The transcoder decisions are made on a MB level, taking into account the information found in the H.264/AVC MB. In order to maximize the reuse of H.264/AVC MVs, the MB size in the transcoder is fixed and it is set as 16×16, and the following partition sizes can be tested: 16×16, 8×8, and 4×4 pixels. The testing of a partition is defined as the approximation (or reuse) and refinement of MVs for each direction (forward and backward) and the mode selection (forward, backward, or bidirectional). If more than one partition size is tested, the one that yields the lowest cost, in rate-distortion sense, is chosen (as seen in Section II-D).

The H.264/AVC MB partitioning is taken into consideration when deciding which partition sizes will be tested in the transcoder. This is a common strategy in transcoding [25]. However, as said previously, the H.264/AVC partitioning may not be optimal for the W-SVC codec. In our transcoder, if the MB was encoded in inter mode in H.264/AVC, then only partitions that are equal or bigger than those in the H.264/AVC MB will be tested. This is applied regardless whether MVs can be directly reused or not. However, since the W-SVC codec uses a strict quadtree partitioning, this strategy cannot be applied straightforwardly. As an example, if the H.264/AVC MB was partitioned into two 16×8 blocks, then both 16×16 and 8×8 partitions will be tested in W-SVC. This procedure is depicted in Fig. 4. If the H.264/AVC MB was encoded in intra mode, then the partitions are tested as follows: first 16×16 and 8×8 partitions will be tested and, if the latter has a lower cost, the 4×4 partitions will be tested.

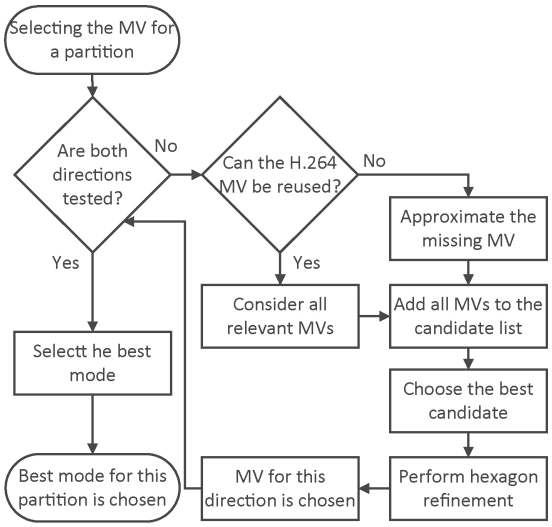

Fig. 5. Selecting the MV and the mode for a given partition.

B. Framework for Approximation and Refinement

To efficiently use the motion information found in the H.264/AVC stream a framework for approximation and refinement of MVs was developed. As discussed in Section III-A, the H.264/AVC partitioning is used to decide which partitions will be tested in the W-SVC codec. This framework specifies how these partitions are tested. The flowchart of the algorithm is shown in Fig. 5.

The transcoder keeps a MV candidate list for each direction, which starts empty. All H.264/AVC MVs that can be reused are added to the appropriate candidate list. Otherwise, if none MVs can be reused for a particular direction, several strategies are used to populate the corresponding candidate list(s), which will be explained in the next section. The transcoder then evaluates the cost of all MVs in each candidate list. Afterward, a refinement step is applied for each direction, starting at the best candidate selected for that direction. In all cases, the refinement considered is a hexagon search [37]. All candidate MVs are in quarter-pixel scale. When the refinement is performed, the best candidate is rounded and the refinement is applied in integer-pixel scale, with two further refinements for half-pixel and quarter-pixel which consider only the eight immediate neighbors. Finally, the cost of bidirectional prediction is evaluated, using the best MV for each direction. No further refinement is made to test for the bidirectional prediction. The transcoder then decides the mode for this partition (forward, backward, or bidirectional).

C. MV Approximation Techniques

The main reason why the transcoder needs to approximate MVs is the RF mismatch. To cope with this problem, the transcoder uses MV composition and scaling. In addition to these two main techniques, the transcoder can also use spatial and inversion approximation techniques.

For all techniques, we are using a similar notation as found in [26]. Let B_n^k represent the block at the position k in frame n. The MV for a particular block B_n^k, which points to RF $n-\alpha$, is similarly denoted as $mv_{n\to n-\alpha}^k$.

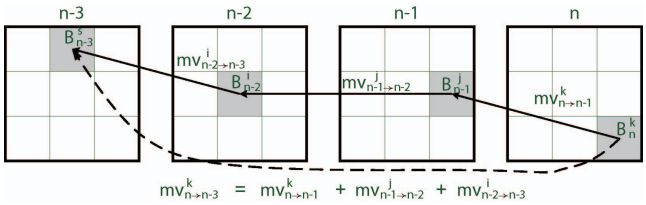

Fig. 6. Example of MV composition.

1) *Spatial Approximation:* Here, two candidate MVs are generated. The first is the median MV, computed in the same way as the H.264/AVC standard [20]. The second MV is formed as the weighted average of MVs belonging to blocks above and on the left of the currently observed block. The equation used to generate the second MV is

$$mv_{n\to n-\alpha}^k = \frac{\sum_{i\in\Gamma} w_i \cdot mv_{n\to n-\alpha}^i}{\sum_{i\in\Gamma} w_i} \tag{1}$$

where Γ denotes the set of partitions directly above and to the left of the current partition, and w_i is the weight, which is equal to the number of pixels in the current partition B_n^k that are neighboring the partition B_n^i. The advantage of these methods is that they use MVs already refined by the W-SVC codec, and thus it does not need a H.264/AVC MV, being able to generate a candidate even if the current H.264/AVC MB was encoded in intra mode.

2) *MV Inversion:* This method is used to create backward MV candidates from already found forward MV candidates. It inverts all MVs from the forward candidate list, adding them to the backward candidate list. This method can be useful for instance in *IPP* coding configuration when only forward MVs are available. The equation for inversion is

$$mv_{n\to n+\alpha}^k = (-1)\cdot mv_{n\to n-\alpha}^k. \tag{2}$$

3) *MV Scaling:* This is not a particularly accurate technique, but its advantage is its low computational complexity and the possibility to always produce a candidate, given a starting MV. The scale factor is directly proportional to the distance between the H.264/AVC and the W-SVC RFs and the current frame [36]. Let n be the current frame, $n-\alpha$ be the RF used by the H.264/AVC MV, and $n-\beta$ be the RF used by the W-SVC MV, then, the equation for scaling is

$$mv_{n\to n-\beta}^k = \left(\frac{\beta}{\alpha}\right)\cdot mv_{n\to n-\alpha}^k. \tag{3}$$

4) *MV Composition:* An example of MV composition is depicted in Fig. 6. In this example, the target is to get the MV $mv_{n\to n-3}^k$ for the block B_n^k. However, the H.264/AVC MV for this block is $mv_{n\to n-1}^k$. Starting from the given block in frame n, the MV composition checks where its MV points to in frame $n-1$. Then, it gets a new H.264/AVC MV from the partition corresponding to that position in frame $n-1$ ($mv_{n-1\to n-2}^j$ in the figure), and then checks where this MV points to in frame

$n-2$. The process iterates until the desired RF is reached. The final MV is then composed as

$$mv^k_{n\to n-3} = mv^k_{n\to n-1} + mv^j_{n-1\to n-2} + mv^i_{n-2\to n-3}. \quad (4)$$

A few problems may arise in MV composition.

a) A H.264/AVC MV generally does not point to a position in the grid. Thus, the partition pointed to by the MV may overlap with other partitions. A popular algorithm that deals with this issue is forward dominant vector selection (FDVS) [22]. The FDVS will select the position on the grid that has the largest overlapping area with the position estimated in an intermediate step of MV composition. An alternative to FDVS is to use telescopic vector composition (TVC) [23], which will always look for MVs in the same position as the starting partition, therefore avoiding the grid-matching problem.

b) Even if the new position matches the grid, there may be more than one MV for the matching partition. As an example, the original starting partition in frame n may have been 16×16, while the partition in frame $n-1$ may have been coded as two 16×8 partitions. A common solution to this problem is to use a weighted average on these MVs [24]. The equation is

$$mv^k_{n\to n-\alpha} = \frac{\sum_{i\in K} w_i \cdot mv^i_{n\to n-\alpha}}{\sum_{i\in K} w_i} \quad (5)$$

where i is the index of a subpartition within the partition B^k_n, K represents the set of subpartitions within the current partition B^k_n, and w_i represents a weight, which is the area occupied by the partition with index i.

c) A MB in the composition process may have been coded in intra mode and, therefore, may have no MV, or its MVs may use invalid RFs (out of the range from current frame to the target RF, or in an invalid direction).

Although the issues described in the last point above may occur in a H.264/AVC stream, most MV composition methods are not suited to tackle these issues [22]–[27], [38]–[40], and are applied to situations where the H.264/AVC stream was encoded in *IPP* configuration with one RF. To deal with more complex motion structures, we developed a new MV composition method that is similar in spirit to FDVS [22] and TVC [23], extended to work with different coding configurations, multiple RFs, and variable block sizes occurring in H.264/AVC motion information. It combines both FDVS and TVC to overcome the issues listed before, and it is explained in the following sections.

MV Composition Based on FDVS Supporting Multiple RFs and Variable Block Sizes (MRVB-FDVS): The aim here is to compose a MV from the current frame n to the W-SVC RF $n-\beta$. The example here is constructed based on the test of a 16×16 partition for B^k_n. The algorithm works in the same way for other partition sizes.

The algorithm is based on an ordered MV list, which starts empty (list = $\{\cdot\}$). At each step of composition, the list is refreshed. In the first step, all H.264/AVC MVs within the W-SVC partition $P_0 = B^k_n$ are considered. These MVs are grouped according to their RFs; each group contains MVs that point to the same RF. The algorithm then calculates a weighted average of MVs within each group, with weights that are proportional to the size of their corresponding partition area. The averaged MVs whose RF is between the frames n and $n-\beta$ are added into the list, ordered such that the elements toward the end of the list have RFs that are closer to $n-\beta$. This first step is depicted in Fig. 7. In the figure, $k0$ and $k1$ denote the two 16×8 partitions within B^k_n. In the example, the partition B^k_n has two MVs, which point to two different RFs, $n-1$ and $n-2$. They are classified in two groups, and there is no need to compute an average because each group has only one MV, and thus $mv^k_{n\to n-1} = mv^{k0}_{n\to n-1}$ and $mv^k_{n\to n-2} = mv^{k1}_{n\to n-2}$. The two MVs are added to the ordered list, which is now

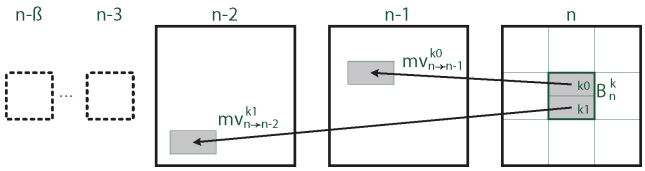

Fig. 7. Step 1 of MV composition using MRVB-FDVS.

$$\text{list} = \left\{ mv^k_{n\to n-1}, mv^k_{n\to n-2} \right\}. \quad (6)$$

In the next step, the algorithm pops the last MV on the list (i.e., the one closer to the target RF, $mv^k_{n\to n-2}$, in the example) and adds it into the current composition group CG. The CG is the group of MVs that are effectively used to compose the MV from frame n to frame $n-\beta$. It also starts empty and, at this point in the example, it is

$$\text{CG} = \left\{ mv^k_{n\to n-2} \right\}. \quad (7)$$

The new position of the partition in the frame $n-2$ (in this example) is calculated as $P_i = P_0 + \sum_{i\in \text{CG}} \text{MV}_i$. In the example, we have $P_1 = P_0 + mv^k_{n\to n-2}$. To continue the composition, the position P_1 is aligned to the grid (of the same size as the original partition, which, in the example, is 16×16). The aligned position, therefore, corresponds to the partition that has the highest number of overlapping pixels with the 16×16 area at the nonaligned position P_1. This is depicted in Fig. 8, where P_1 is aligned to the position of the block B^j_{n-2}. In the example, this block is also partitioned in two 16×8 blocks (the partitions B^{j0}_{n-2} and B^{j1}_{n-2}), and both of its MVs point to frame $n-3$. In this case, in order to have only one MV for this step, the MVs are averaged

$$mv^j_{n-2\to n-3} = \frac{mv^{j0}_{n-2\to n-3} + mv^{j1}_{n-2\to n-3}}{2}. \quad (8)$$

This MV is then added to the ordered list, which is now

$$\text{list} = \left\{ mv^k_{n\to n-1}, mv^j_{n-2\to n-3} \right. . \quad (9)$$

In the next step, the element at the back of the list, $mv^j_{n-2\to n-3}$, is popped and the algorithm repeats again until the target RF, $n-\beta$, is reached. The composed MV is

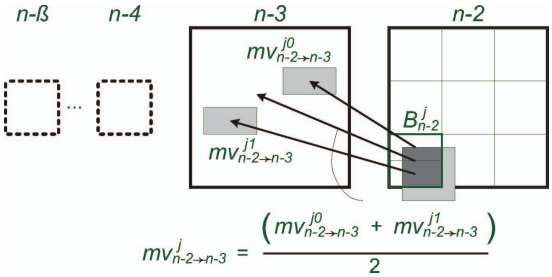

Fig. 8. Step 2 of MV composition using MRVB-FDVS.

$mv^k_{n\to n-\beta} = \sum_{i\in CG} MV_i$. If, at some intermediate step, the MVs cannot be used (either because they point outside the desired range or the block has been coded in intra mode), the algorithm will remove all MVs from CG that were added in any subsequent step to the one in which the currently last element of the ordered list has been added. Then it will pop another MV from the ordered list, using it in the same way as before. The algorithm continues in such a manner until the target RF $n-\beta$ has been reached or until the list is empty. Once the target RF has been reached, the algorithm stops, and the remaining elements in the lists are cleared.

MV Composition Based on TVC Supporting Multiple RFs and Variable Block Sizes (MRVB-TVC): This method is very similar to the method described in the previous section. The only difference is that, at each composition step, the new position is estimated as $P_i = P_0$. In other words, the position of the block is considered to be the same throughout the frames. The final MV is composed in the same way. Note that, even though the position is kept the same, the final composed MV does not have to be zero. It only means that the MVs for the composition are collected from partitions at the same position as the originating partition P_0.

Composing MVs in IPP coding configuration with One RF: For the simple case of *IPP* coding configuration with one RF, only the forward MV is available from H.264/AVC, but the W-SVC codec still needs two MVs to form a bidirectional prediction. Let frame n be the current frame, frame $n-\beta$ be the target RF for the forward MV, frame $n+\beta$ be the target RF for the backward MV, and B^k_n be the current partition, for which it is desired to compose the MV. Then, we have the following.

a) For the forward MV, MRVB-FDVS is directly applied for MV composition, starting at block B^k_n and generating the approximated MV $mv^k_{n\to n-\beta}$.
b) For the backward MV, the approach is changed slightly. Since there is no backward MV, the transcoder applies MRVB-TVC starting at block $B^k_{n+\beta}$, generating a MV $mv^k_{n+\beta\to n}$. Then, it uses inversion on this MV to get the approximated MV $mv^k_{n\to n+\beta}$. TVC is used instead of FDVS because the latter cannot guarantee that the composition will arrive at the same block, which is not the case with the former.

Composing MVs in IPP Coding Configuration with Multiple RFs: In the case of multiple RFs, the composition method is performed in two phases. The second phase is executed only if the composition is unsuccessful in the first phase. The first phase is similar to the composition for *IPP* coding configuration with one RF, with the main difference that the algorithm may record composed MVs for frames beyond the target RF.

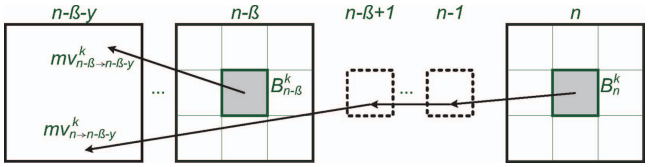

Fig. 9. Example of MV composition in two phases. The goal is to compose the MV $mv^k_{n\to n-\beta}$ for block B^k_n. However, for some reason, the composition can only compose the MV $mv^k_{n\to n-\beta-\gamma}$ in the first phase (using MRVB-FDVS). The second phase attempts to compose the MV $mv^k_{n-\beta\to n-\beta-\gamma}$ for block $B^k_{n-\beta}$, using MRVB-TVC. The final composed MV is then $mv^k_{n\to n-\beta} = mv^k_{n\to n-\beta-\gamma} - mv^k_{n-\beta\to n-\beta-\gamma}$.

First, the algorithm tries to compose a MV starting at the block B^k_n to the desired RF $n-\beta$, using MRVB-FDVS. If it cannot compose such MV, it allows the last iteration to pop a MV with a RF beyond the target RF, generating the MV $mv^k_{n\to n-\beta-\gamma}$. Then, it will try to compose a MV starting at $B^k_{n-\beta}$ to the RF $n-\beta-\gamma$, using MRVB-TVC. Finally, the final composed MV is $mv^k_{n\to n-\beta} = mv^k_{n\to n-\beta-\gamma} - mv^k_{n-\beta\to n-\beta-\gamma}$. This is depicted in Fig. 9.

The same strategy is applied to compose the backward MV. For this case, since there is no backward MV in the block B^k_n to start the composition, the algorithm applies MRVB-TVC starting at block $B^k_{n+\beta}$. If the desired RF n cannot be reached, it allows the last iteration to pop a MV with a RF beyond the frame n, generating the MV $mv^k_{n+\beta\to n-\gamma}$. Then, it will try to compose a MV starting at B^k_n to the RF $n-\gamma$, also using MRVB-TVC. Finally, the final composed MV is

$$mv^k_{n\to n+\beta} = (-1)\cdot \; mv^k_{n+\beta\to n-\gamma} - mv^k_{n\to n-\gamma} \quad . \qquad (10)$$

Composing MVs for Other Coding Configurations: In the more general case, when *B*-frames are present in the stream, the composition for a given block B^k_n to a target RF $n-\beta$, for the forward MV, and $n+\beta$, for the backward MV, works as follows.

a) If the block B^k_n has a forward MV, then MRVB-FDVS is used starting at this block. If a direct MV $mv^k_{n\to n-\beta}$ cannot be composed in the first phase, the algorithm tries to compose the MV using the second phase.
b) If the block B^k_n does not have a forward MV, then the algorithm checks if the block $B^k_{n-\beta}$ has a backward MV. If this is the case, then MRVB-TVC is used to compose the MV $mv^k_{n-\beta\to n}$, which will then be inverted to get the desired MV. Similarly, a second phase may be performed if the composition is not successful in the first phase.
c) If the block B^k_n has a backward MV, MRVB-FDVS is performed starting at this block, also allowing two phases.
d) If the block B^k_n does not have a backward MV, then the algorithm checks if the block $B^k_{n+\beta}$ has a forward MV. It will then use MRVB-TVC to compose the MV $mv^k_{n+\beta\to n}$, which will be inverted to get the desired MV. Similarly, a second phase may be performed if the first phase is unsuccessful.

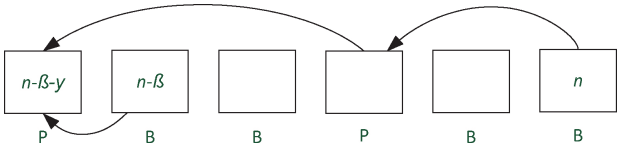

Fig. 10. Example of MV composition for *IBBP* configuration. In this figure, the arrows point to the RF used by each frame. The goal is to compose the MV $mv^k_{n\to n-\beta}$. Here, the first phase of composition will be unsuccessful for every block in frame n, being able only to generate a MV $mv^k_{n\to n-\beta-\gamma}$. Thus, the second phase is applied to adjust the composition.

Note that the second phase may be particularly useful in the case of *IBBP* configuration, as depicted in Fig. 10. However, even with the second phase, the MV composition method proposed here cannot always succeed in generating a candidate. A number of reasons may cause this, like a high number of intra MBs, or very different RFs. For this reason, this method is used in conjunction with other MV approximation methods. This is further discussed in Section IV-C.

D. Reduced Complexity Module

The reduced complexity (RC) module comprises three optional strategies that can be added to the proposed transcoder: using the H.264/AVC coded block pattern (CBP), the similarity of MVs, and a criterion to decide if the MV for a given partition will be refined or not.

1) *Using the H.264/AVC CBP:* In the H.264/AVC codec, a decoded block is given as

$$B_{DEC} = P + R + D_F(P + R) \tag{11}$$

where B_{DEC} is the decoded block, P is the prediction for this block, R is the residual, and $D_F(\cdot)$ is the effect of the deblocking filter [41], which is applied to $P + R$. Thus, if no coefficient is transmitted, then the residual is zero, and the decoded block is given as

$$B_{DEC} = P + D_F(P). \tag{12}$$

Since the W-SVC codec uses MCTF, ME is performed using the original frames, which, in the transcoder are the decoded H.264/AVC frames. To further reduce the complexity, the transcoder avoids refining the MVs when the residual for the corresponding block is zero. When this happens, H.264/AVC MVs are directly assigned to that partition, without further refinement. Other partitions may also be tested even when this happens, in which case the usual approach of approximation and refinement is used.

In the transcoder implementation, the syntax parameter CBP is used to check if the residual is zero. However, the CBP only tells the presence or absence of AC DCT coefficients in each 8×8 block inside the MB [5]. Thus, if the CBP for a given block is zero, it does not necessarily mean that the residual is zero, since it can still have a DC coefficient. Furthermore, the CBP does not have the information for each 4×4 block separately. However, tests comparing using the CBP or using the actual number of nonzero DCT coefficients to drive the transcoder yield very similar results. Therefore, the transcoder uses the CBP, since it is more readily available.

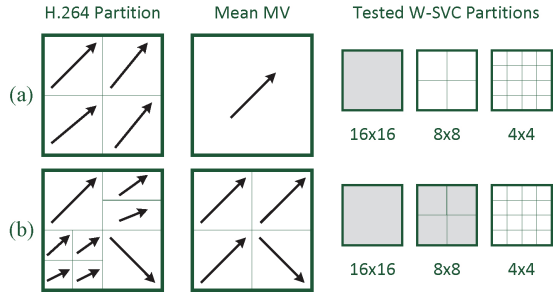

Fig. 11. Example of application of MV similarity. (a) Since all MVs are similar to the mean MV, only the 16×16 partition will be tested (the shaded partitions in the figure). (b) MVs are similar for each 4×4 partition inside the 8×8 partition, but not for the 8×8 partition. Therefore, both 16×16 and 8×8 partitions will be tested.

2) *MV Similarity:* Since the W-SVC codec favors larger partitions instead of smaller ones [18], the transcoder may avoid the testing of smaller partitions when the H.264/AVC MVs corresponding to these partitions are similar. First, a mean MV, $\overline{mv^k_{n\to n-\alpha}}$, is computed for each 8×8 partition, using the four 4×4 blocks within each 8×8 partition. Then, each of the four MVs is compared to the mean MV of the encompassing 8×8 partition, using a similarity measure that considers each component separately. Let $mv^k_{n\to n-\alpha}.x$ and $mv^k_{n\to n-\alpha}.y$ be the horizontal and vertical components of MV $mv^k_{n\to n-\alpha}$, respectively. Then, a single MV is considered similar to the mean MV if both these conditions are satisfied

$$\begin{aligned} \left|\overline{mv^k_{n\to n-\alpha}}.x - mv^k_{n\to n-\alpha}.x\right| &< T \\ \left|\overline{mv^k_{n\to n-\alpha}}.y - mv^k_{n\to n-\alpha}.y\right| &< T \end{aligned} \tag{13}$$

where T is a threshold, which is set as 0.5, in integer-pixel scale. Note that, inside each 8×8 partition, all MVs share the same RF. If all available H.264/AVC MVs inside an 8×8 block are considered similar, then 4×4 partitions will not be tested for this 8×8 block, regardless of the H.264/AVC partitioning. The similarity measure is applied to the four 8×8 blocks within a MB in a similar manner: if they are considered similar, 8×8 partitions will not be tested for this MB. In this case, however, the RF is not necessarily the same for all partitions. If they are not the same for all MVs, then they are considered as not similar. This process applies whether or not the H.264/AVC MVs can be directly reused. An example is depicted in Fig. 11. However, when the distance between the current frame and the RF becomes too large, the correlation between the H.264/AVC and the W-SVC motion information drops. Thus, this strategy is only used if the current frame and the target RF are distant from each other up to a threshold, which is set to four frames.

3) *Deciding on the Refinement:* After all MV candidates are considered, the transcoder uses a SAD criterion to decide if this MV will be refined or not. This criterion is similar to that used in the enhanced predictive zonal search (EPZS) algorithm [42], and is also commonly found in other fast

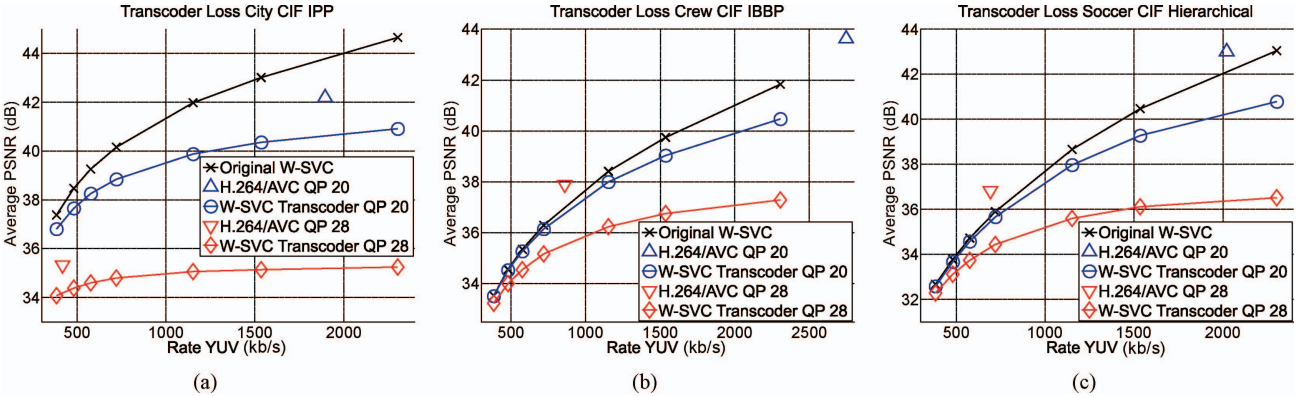

Fig. 12. Results of the W-SVC codec and the trivial transcoder for (a) *City* CIF *IPP*, (b) *Crew* CIF *IBBP*, and (c) *Soccer* CIF *hierarchical*.

TABLE II
NUMBER OF TEMPORAL DECOMPOSITIONS PERFORMED IN THE W-SVC CODEC FOR EACH SEQUENCE AND RESOLUTION

Sequence	CIF	4CIF
City	5	5
Soccer	3	4
Crew	3	4
Harbor	6	6

search algorithms. Here, a static threshold is used as follows:

$$T_{\mathrm{SAD}} = 2 \cdot N \cdot N \cdot \left(\sqrt{2}\right)^{l} \tag{14}$$

where N denotes the current partition size and l denotes the temporal decomposition level of the current frame (starting at $l = 0$). The factor $\left(\sqrt{2}\right)^{l}$ accounts for the change in the dynamic range of the frame after l temporal decompositions. If the SAD for the best candidate is lower than this threshold, no refinement is performed (not even subpixel refinement), and the MV for that direction is chosen as the best candidate.

IV. EXPERIMENTAL RESULTS

The H.264/AVC codec used in the experiments is the reference software JM 14.2. In all cases, the peak-to-signal noise ratio (PSNR) shown is the average among all frames of the luma component, while the rate also includes the two chrominance components. The PSNR is always computed using the original sequence as the reference, and the W-SVC codec uses MCTF with bidirectional prediction. Three different structures are used in H.264/AVC: 1) *IPP* with one RF; 2) *IBBP* with five RFs for *P*-frames and one RF, each side, for *B*-frames; and 3) *hierarchical*, with three levels of hierarchy, and one RF each side for *B*-frames and one RF for *P*-frames, similar to what is shown in Fig. 2. They are referred to *IPP*, *IBBP*, and *hierarchical*, respectively, in the remainder of this section. Four different sequences at CIF resolution with 30 frames per second (f/s) and 4CIF resolution (704 × 576 pixels) with 60 f/s are used in the experiments. The number of frames used is 300, 298, and 297 for CIF resolution, and 600, 597, and 593 for 4CIF resolution, for *IPP*, *IBBP*, and *hierarchical* structures, respectively. In order to account for different W-SVC codec configurations, a different number of temporal decompositions was used for each sequence. This is shown in Table II.

In this section, the proposed transcoder is defined as the transcoder using the techniques in Sections III-A through III-C, while the proposed transcoder with the RC module also uses techniques found in Section III-D. They are referred to as proposed transcoder (PT) and PT-RC, respectively, in the remainder of this section.

A. Reference Transcoders

The proposed transcoder is compared to three reference trivial transcoders: 1) using full ME; 2) using hexagon search [37]; and 3) using EPZS [42]. They are referred as RT-FS, RT-HS, and RT-EPZS, respectively, in the remainder of this section. The full ME uses a search window of 32, while the others use a search window of 60, in integer-pixel scale. The search window always enlarges for higher levels of temporal decomposition. Both the full ME and the hexagon search start at the predicted MV, given by (1). The EPZS algorithm implemented here is the same as that implemented in the H.264/AVC reference code, JM 14.2. The only modification of the algorithm is that the dynamic range change due to temporal filtering is considered (similar to Section III-D3). Note that this implementation is more complex than the one in the original paper [42], using more MV predictor techniques to ensure that the best PSNR is achieved, such as the median, spatial, spatial memory, temporal, and window-based predictors. On the other hand, the hexagon search used here tests only a simple hexagon pattern, and thus it is more likely to converge to local minima. Also, as discussed in Section II-D, the λ used in the W-SVC codec is usually higher than that used in a hybrid codec. The side effect of this λ choice on hexagon search is that the search will be more confined to the starting point, and thus more inclined to converge to local minima, making this search fast, but with an impact on performance. Furthermore, in order to reduce complexity, all three reference transcoders test the partitions in the following pattern: first, the 16×16 and 8×8 partitions are tested, then, only if the latter has a lower cost, the 4×4 partitions are tested. Testing all partitions leads to a marginal gain in terms of quality, but it increases complexity by 115%, on average.

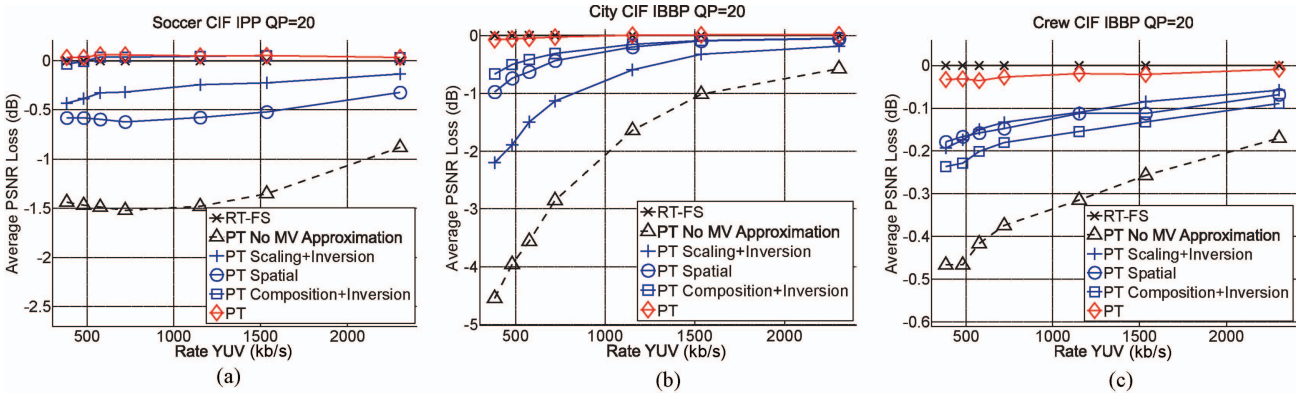

Fig. 13. Evaluation of MV approximation on the transcoder for (a) *Soccer* with *IPP* structure, (b) *City* with *IBBP* structure, and (c) *Crew* with *IBBP* structure. All sequences are CIF and the H.264/AVC QP used for this test was 20.

B. Transcoder Loss

The first point analyzed here is the loss caused by transcoding. Fig. 12 shows the rate-distortion point of the decoded H.264/AVC sequence using two different QPs (20 and 28), and the performance of the W-SVC codec operating on the original sequence and on the two H.264/AVC streams. It can be seen that a certain loss of quality is present in transcoding, specially at medium and higher bitrates. This is expected, since the transcoder operates on a quantized sequence, and it is therefore limited to the quality of this sequence.

C. Evaluation of MV Approximation Techniques

The second point analyzed here are the MV approximation techniques used. To evaluate these techniques, the following transcoder configurations are compared: 1) no MV approximation, i.e., reusing H.264/AVC MVs and partitions where possible, and when the MVs cannot be reused, a refinement starting at the (0, 0) MV is applied; 2) MV scaling and inversion only are used for MV approximation; 3) spatial approximation only is used; 4) MV composition and inversion only are used; and 5) all approximation techniques presented in Section III-C are used (i.e., the proposed transcoder). The results are shown in Fig. 13, relative to RT-FS. When the MV approximation methods fails to generate any candidate, the refinement starts at the (0, 0) MV.

It can be noted that using MV approximation techniques is crucial for the transcoder, yielding gains of up to 4 dB, when compared to not using MV approximation at all. For both *Soccer* and *City* sequences, shown in Fig. 13(a) and (b), the MV composition method contributes to the PSNR gain more than the other MV approximation methods, even though it does not succeed in generating a MV for every block. In order to evaluate the rate of success in generating a candidate by the MV composition method, the transcoder was applied to the first 100 frames of the four sequences used so far at the CIF resolution. For the *IPP* configuration, on average, the MV composition method was able to generate a candidate in 86.15% and 90.59% of the cases where it was required to do so, for the forward and backward directions, respectively. For *IBBP* configuration, a candidate was generated 75.16% and 74.90% of the times, for the forward and backward direction, respectively. Note that the standard FDVS [22] and TVC [23] methods cannot produce any MV for the backward direction, and while the rate of success for the forward direction is similar in the case of *IPP* configuration (86.15% and 86.40%, for FDVS and TVC, respectively—note that, for this particular case, FDVS and MRVB-FVDS behave in the same way), the rate of success for the *IBBP* configuration is almost insignificant (5.42% and 5.43%, for FDVS and TVC).

For *Crew* sequence, shown in Fig. 13(c), the MV scaling and MV spatial techniques yield better performance than MV composition. This can be explained by the fact that the latter can fail to produce a candidate, especially in the case when the H.264/AVC sequence has many intra MBs. On the other hand, MV scaling is always able to produce a candidate, given a starting MV, and MV spatial does not use H.264/AVC MVs at all. For the specific case shown in Fig. 13(c), 21.36% of the MBs in inter frames are encoded in intra mode, and the rate of success for MV composition is 52.15%, for the first 100 frames. In all cases, the use of all techniques together yields a higher performance, which clearly shows the rationale for integrating all of them into the proposed transcoder.

D. Proposed Transcoder

Selected results for PT and PT-RC are shown in Fig. 14. The complete results for coding structures *IPP*, *IBBP*, and *hierarchical* are presented in Tables III–V, respectively, where the average PSNR loss compared to RT-FS, between all bitrates, is shown. For CIF resolution, the following bitrates were used: {384, 480, 576, 720, 1152, 1536, 2304} kb/s, while for 4CIF resolution the bitrates used were {1280, 1536, 1792, 2048, 2304, 2688, 3072} kb/s.

It can be seen that the PT has a performance very close to RT-FS, and it consistently outperforms RT-HS and RT-EPZS, for all sequences, resolutions, and coding configurations tested. The sole exception is for *Soccer* CIF *IBBP*, where RT-EPZS performs marginally better, by 0.01 dB. For some sequences and resolutions, the PT even outperforms RT-FS by a small margin (up to 0.16 dB). This is because the latter uses a smaller search window (32, instead of 60), and it does not test all partitions (as seen in Section IV-A). The search window for full ME is smaller due to the complexity, which increases with

TABLE III
AVERAGE PSNR LOSS FOR *IPP* STRUCTURE COMPARING TO RT-FS

Resolution	QP	Sequence	RT-HS	RT-EPZS	PT	PT-RC
CIF	20	*City*	−0.69	−0.19	0.01	−0.03
		Soccer	−0.85	−0.07	0.04	0.02
		Crew	−0.26	−0.07	−0.02	−0.06
		Harbor	−0.24	−0.06	0.01	−0.01
	28	*City*	−0.27	−0.07	−0.01	−0.03
		Soccer	−0.44	−0.04	0.02	−0.01
		Crew	−0.19	−0.04	0.00	−0.04
		Harbor	−0.14	−0.05	0.01	−0.01
4CIF	26	*City*	−0.53	−0.23	0.02	−0.03
		Soccer	−0.93	−0.05	0.10	0.03
		Crew	−0.32	−0.13	−0.01	−0.18
		Harbor	−0.21	−0.07	0.01	−0.02
	34	*City*	−0.22	−0.11	−0.02	−0.06
		Soccer	−0.37	−0.02	0.07	−0.01
		Crew	−0.18	−0.08	0.01	−0.11
		Harbor	−0.12	−0.03	0.00	−0.04

TABLE IV
AVERAGE PSNR LOSS FOR *IBBP* STRUCTURE COMPARING TO RT-FS

Resolution	QP	Sequence	RT-HS	RT-EPZS	PT	PT-RC
CIF	20	*City*	−0.67	−0.19	−0.02	−0.06
		Soccer	−0.80	−0.07	−0.08	−0.09
		Crew	−0.27	−0.06	−0.02	−0.05
		Harbor	−0.22	−0.06	0.01	0.00
	28	*City*	−0.26	−0.07	−0.03	−0.06
		Soccer	−0.41	−0.04	−0.02	−0.05
		Crew	−0.16	−0.04	−0.01	−0.04
		Harbor	−0.15	−0.04	0.00	0.00
4CIF	26	*City*	−0.55	−0.24	0.00	−0.02
		Soccer	−0.90	−0.05	0.09	0.10
		Crew	−0.31	−0.14	0.00	−0.07
		Harbor	−0.21	−0.07	0.01	0.00
	34	*City*	−0.22	−0.12	−0.03	−0.06
		Soccer	−0.39	−0.03	0.04	0.02
		Crew	−0.18	−0.08	−0.01	−0.04
		Harbor	−0.10	−0.03	0.00	−0.01

the window size. Since both the transcoder and other fast ME only test a small number of MVs, a larger search window has low impact on the complexity. Moreover, PT also benefits from the partitioning found in the H.264/AVC stream.

Using the RC module (PT-RC) has a low impact on the transcoder quality (−0.01 dB and −0.05 dB for CIF and 4CIF resolutions, respectively, on average). There are a few cases where this loss is higher, notably in the case of *Crew* sequence. In very few cases (10% of the cases tested), the RT-EPZS shows a slightly better performance than PT-RC. Such an example is shown in Fig. 14(f), where the average performance of the PT-RC and RT-EPZS is similar. However, even in this case, at medium and high bitrates, PT-RC still yields a better PSNR performance.

E. Complexity Analysis

Since the transcoder is mainly based on reducing the ME complexity, the analysis in the remainder of this section is focused on the ME module. However, although it is out of the scope of this paper, we briefly discuss the complexity of ME compared to the other modules in the W-SVC codec here. In our experiments, ME accounted for 97% of the execution time of the RT-FS, making it the most complex

TABLE V
AVERAGE PSNR LOSS FOR *Hierarchical* STRUCTURE COMPARING TO RT-FS

Resolution	QP	Sequence	RT-HS	RT-EPZS	PT	PT-RC
CIF	20	*City*	−0.69	−0.22	0.03	−0.01
		Soccer	−0.82	−0.07	0.01	−0.01
		Crew	−0.26	−0.06	0.02	−0.02
		Harbor	−0.23	−0.06	−0.01	−0.01
	28	*City*	−0.33	−0.11	0.01	0.00
		Soccer	−0.43	−0.04	0.01	0.02
		Crew	−0.17	−0.05	0.04	0.01
		Harbor	−0.16	−0.04	0.01	0.02
4CIF	26	*City*	−0.63	−0.27	−0.08	−0.15
		Soccer	−0.91	−0.03	0.16	0.09
		Crew	−0.31	−0.13	0.02	−0.29
		Harbor	−0.20	−0.06	−0.05	−0.13
	34	*City*	−0.19	−0.07	0.04	0.03
		Soccer	−0.40	−0.02	0.09	0.11
		Crew	−0.19	−0.09	0.02	−0.08
		Harbor	−0.14	−0.03	−0.01	0.06

TABLE VI
COMPLEXITY FOR *IPP* STRUCTURE

			RT-HS		RT-EPZS		PT		PT-RC	
Resolution	QP	Sequence	SAD	Time	SAD	Time	SAD	Time	SAD	Time
CIF	20	*City*	0.65	19.3	0.89	32.6	0.41	14.9	0.21	8.0
		Soccer	0.87	20.5	1.18	34.2	0.66	17.5	0.40	10.7
		Crew	0.82	21.6	1.17	36.7	0.73	20.6	0.57	15.5
		Harbor	0.45	18.5	0.64	32.8	0.40	20.3	0.27	13.0
	28	*City*	0.67	19.6	0.88	31.6	0.37	11.9	0.18	6.6
		Soccer	0.91	20.8	1.15	31.8	0.58	13.9	0.35	8.8
		Crew	0.85	21.8	1.17	35.4	0.67	17.3	0.50	12.8
		Harbor	0.45	18.7	0.64	32.7	0.36	17.8	0.25	11.9
4CIF	26	*City*	0.57	159.4	0.75	253.5	0.32	98.8	0.16	54.2
		Soccer	0.59	169.2	0.74	255.7	0.37	112.6	0.20	65.3
		Crew	0.55	177.5	0.77	286.7	0.41	131.9	0.29	93.3
		Harbor	0.47	157.8	0.59	250.8	0.34	129.2	0.24	88.1
	34	*City*	0.59	162.3	0.72	233.8	0.28	76.8	0.10	34.5
		Soccer	0.59	172.5	0.68	234.6	0.33	97.3	0.15	50.2
		Crew	0.56	179.0	0.74	267.4	0.37	115.2	0.24	73.5
		Harbor	0.48	158.2	0.58	235.2	0.27	93.8	0.17	61.5

SAD is a percentage of RT-FS, and time refers to ME for all frames, in seconds.

module in transcoding. It is worth mentioning, however, that the W-SVC codec used in this paper was not optimized in the software, especially the interpolation and entropy coding modules. Therefore it is expected that the complexity of ME relative to the other modules may be even higher.

The complexity was measured both in terms of the average number of SAD operations and running time. Recall from Section II-D that the cost of each MV is computed as $J = D + \lambda \cdot R$. The complexity of estimating R is fairly low, since R is calculated for the whole block by simple prediction from the neighboring motion information. However, calculating D, which is measured as SAD, involves many more operations and it is the most complex operation in ME, even for fast algorithms. The SAD of a block B can be expressed as $\mathrm{SAD}_B = \sum_{i=0}^{M-1} |p_i - \hat{p}_i|$, where M is the number of considered pixels in the block, p_i is the pixel with the index i in the block B, and $\hat{p}_i$ is the pixel with the index i in the reference block. Following the previous equation, we define a SAD operation as calculating a difference between

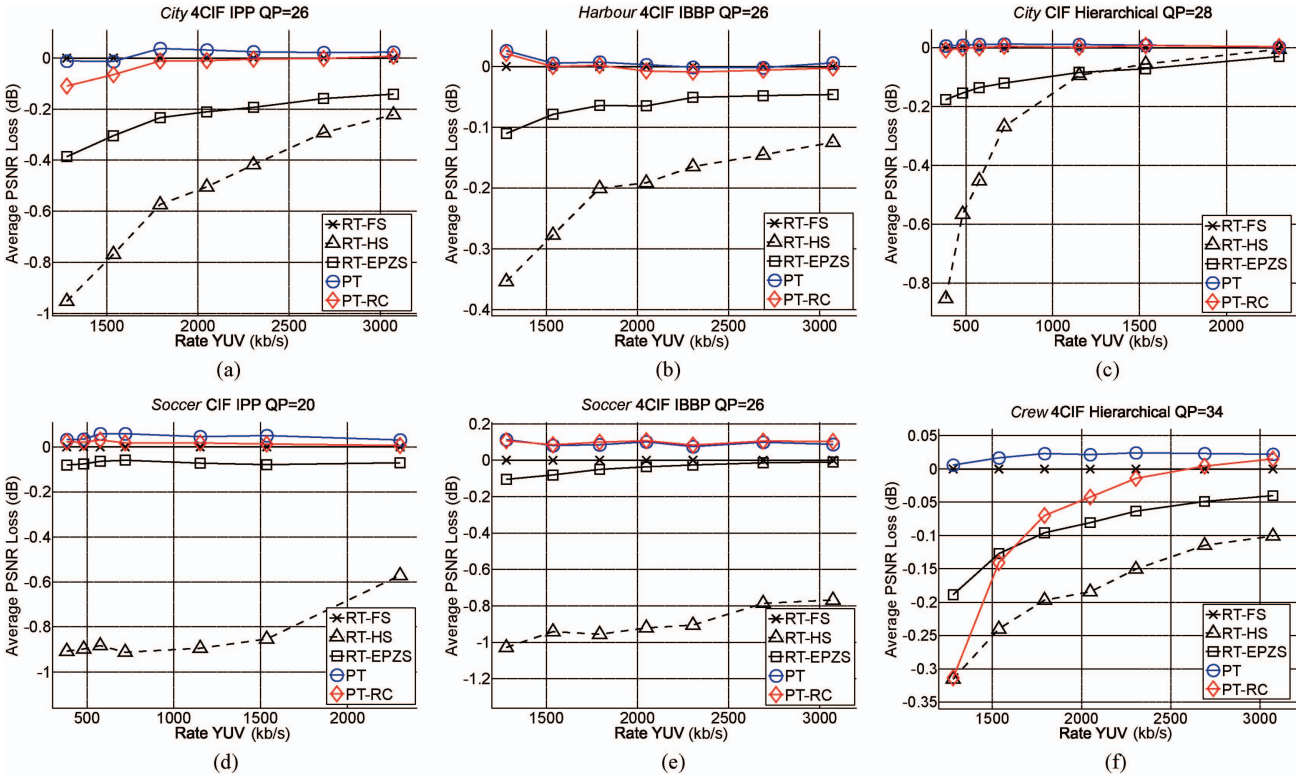

Fig. 14. Transcoder results for (a) *City* 4CIF *IPP* QP = 26, (b) *Harbor* 4CIF *IBBP* QP = 26, (c) *City* CIF *hierarchical* QP = 28, (d) *Soccer* CIF *IPP* QP = 20, (e) *Soccer* 4CIF *IBBP* QP = 26, and (f) *Crew* 4CIF *hierarchical* QP = 34.

two pixels, calculating the absolute value of that difference and adding the absolute difference to the sum of previously calculated absolute differences. Thus, calculating the SAD for the block B consists of M SAD operations. Using the number of SAD operations as an indication of complexity has the advantage of being an objective measure, independent of the software optimizations and the hardware used for experiments. However, it does not account for the complexity of the MV approximation techniques, or other modules of the transcoder. To measure the running time, a PC with an Intel Core 2 Duo CPU P8700 running at 2.53 GHz, with 4 GB of RAM and Windows 7 64-bit was used. Since the transcoder is mainly based on MI reuse, only the time spent on ME is considered here. Each sequence was encoded five times, and the average time spent on ME for all frames was measured. Execution times and the number of SAD operations are shown in Tables VI–VIII, for the *IPP*, *IBBP*, and *hierarchical* structures, respectively. The SAD operations are shown relative to RT-FS, but the execution times are shown only for RT-HS, RT-EPZS, PT, and PT-RC. In a test for *IPP* configuration, the PT was from 130 to 300 times faster than RT-FS, for CIF resolution, and from 250 to 400 times faster, for 4CIF resolution.

From the tables it can be seen that both the SAD operations and the execution time show that the complexity of the PT is dependent on the H.264/AVC structure and QP, as well as the resolution of the sequence. It can be seen that the PT is significantly faster than RT-EPZS for all sequences, resolutions, and coding structures. Comparing to RT-HS, it is faster for all cases except one. For the *IPP*, *IBBP*, and *hierarchical* structures, respectively, the PT is 1.42, 1.57, and 1.71 times faster than RT-HS, and 2.24, 2.50, and 2.67 times faster than RT-EPZS. In the above-mentioned single case, PT is 9% slower than RT-HS. In the best case, it is 2.22 times faster. Comparing to RT-EPZS, the PT is 1.62 and 3.21 times faster, in the worst and in the best case, respectively.

Using the RC module significantly speeds up the transcoder, by 71%, on average. In the worst case, the PT-RC is 1.34 and 2.35 times faster, and in the best case, it is 7.32 and 10.60 times faster than RT-HS and RT-EPZS, respectively. For the

TABLE VII
COMPLEXITY FOR *IBBP* STRUCTURE

			RT-HS		RT-EPZS		PT		PT-RC	
Resolution	QP	Sequence	SAD	Time	SAD	Time	SAD	Time	SAD	Time
CIF	20	*City*	0.65	19.4	0.89	32.4	0.39	13.2	0.22	8.2
		Soccer	0.88	20.6	1.18	33.7	0.64	16.3	0.44	11.3
		Crew	0.82	21.7	1.17	36.8	0.68	18.6	0.56	15.2
		Harbor	0.45	18.6	0.64	32.5	0.35	16.7	0.30	13.8
	28	*City*	0.69	19.6	0.89	31.6	0.35	11.0	0.19	6.6
		Soccer	0.91	20.8	1.15	31.9	0.58	13.6	0.38	9.3
		Crew	0.86	21.7	1.17	35.4	0.62	16.1	0.49	12.6
		Harbor	0.45	18.3	0.64	31.6	0.32	14.7	0.27	12.1
4CIF	26	*City*	0.58	161.2	0.76	251.8	0.30	88.4	0.16	52.2
		Soccer	0.59	170.6	0.75	254.4	0.35	104.3	0.21	64.7
		Crew	0.56	178.7	0.77	285.9	0.38	121.9	0.28	90.1
		Harbor	0.47	159.3	0.59	246.9	0.31	108.0	0.25	89.1
	34	*City*	0.61	162.8	0.75	237.4	0.29	77.7	0.12	39.5
		Soccer	0.59	172.7	0.70	238.8	0.33	95.0	0.18	54.6
		Crew	0.57	178.9	0.75	269.5	0.36	108.6	0.24	72.9
		Harbor	0.49	155.3	0.57	223.4	0.26	85.1	0.19	63.7

SAD is a percentage of RT-FS, and time refers to ME for all frames, in seconds.

TABLE VIII
COMPLEXITY FOR *Hierarchical* STRUCTURE

			RT-HS		RT-EPZS		PT		PT-RC	
Resolution	QP	Sequence	SAD	Time	SAD	Time	SAD	Time	SAD	Time
CIF	20	*City*	0.64	19.3	0.87	32.5	0.36	12.0	0.21	7.5
		Soccer	0.88	20.5	1.16	33.2	0.58	14.1	0.37	9.5
		Crew	0.83	21.6	1.16	36.5	0.62	16.4	0.48	13.1
		Harbor	0.44	18.5	0.64	32.3	0.31	13.5	0.27	11.7
	28	*City*	0.66	19.8	0.85	32.2	0.31	10.1	0.13	5.2
		Soccer	0.91	20.8	1.13	31.8	0.52	11.9	0.27	6.9
		Crew	0.85	21.7	1.16	35.2	0.56	14.2	0.36	9.6
		Harbor	0.44	18.4	0.62	31.7	0.28	11.9	0.21	9.4
4CIF	26	*City*	0.58	159.4	0.75	247.9	0.29	81.2	0.12	41.5
		Soccer	0.59	169.2	0.74	250.4	0.33	86.3	0.15	50.2
		Crew	0.55	177.5	0.76	281.2	0.36	114.6	0.21	71.8
		Harbor	0.47	157.8	0.58	242.2	0.27	91.0	0.20	71.3
	34	*City*	0.60	162.4	0.73	235.3	0.28	73.2	0.06	22.2
		Soccer	0.59	171.7	0.69	235.1	0.32	90.6	0.09	32.1
		Crew	0.57	178.1	0.74	264.4	0.34	103.3	0.13	45.0
		Harbor	0.48	154.8	0.55	222.2	0.25	78.7	0.10	40.1

SAD is a percentage of RT-FS, and time refers to ME for all frames, in seconds.

TABLE IX
SUMMARY OF THE TRANSCODER RESULTS

Resolution	Option		*IPP*	*IBBP*	*Hierarchical*	Average
CIF	RT-HS	PSNR	−0.39	−0.37	−0.39	−0.38
		Speedup	1	1	1	1
	RT-EPZS	PSNR	−0.07	−0.07	−0.08	−0.08
		Speedup	0.6	0.6	0.6	0.6
	PT	PSNR	0.01	−0.02	0.01	0.00
		Speedup	1.2	1.4	1.6	1.4
	PT-RC	PSNR	−0.02	−0.04	0.00	−0.02
		Speedup	2.0	1.9	2.4	2.1
4CIF	RT-HS	PSNR	−0.36	−0.36	−0.37	−0.36
		Speedup	1	1	1	1
	RT-EPZS	PSNR	−0.09	−0.09	−0.09	−0.09
		Speedup	0.7	0.7	0.7	0.7
	PT	PSNR	0.02	0.01	0.03	0.02
		Speedup	1.6	1.8	1.9	1.7
	PT-RC	PSNR	−0.05	−0.01	−0.05	−0.04
		Speedup	2.8	3.0	4.0	3.3

PSNR results are relative to RT-FS, and the speedup is relative to RT-HS.

specific case shown in Fig. 14(f), the PT-RC is 3.96 times faster than RT-HS, 5.87 times faster than RT-EPZS, and 2.29 times faster than PT. It can also be seen from the tables that the number of SAD operations is a good measure of the transcoder complexity, with the advantage of being an objective measure, and therefore easily reproducible. It is important to note that the mismatch in complexity obtained by measuring the number of SAD operations and by measuring the ME running time can be reduced by further optimization of software implementation of proposed MV approximation techniques.

The difference in the quality of the H.264/AVC sequence has a small impact on the performance of the PT, relative to RT-FS operating on the same sequence, in terms of PSNR. In terms of complexity, the transcoder is usually faster when operating on the lower quality sequence due to the fact that less partitions are tested, since the incidence of larger blocks is higher when the QP is higher. When the RC module is used, the complexity is further reduced, since less DCT coefficients are present in the stream, which triggers the technique shown in Section III-D1. The average transcoder results, for all resolutions and coding configurations, and both in terms of quality and complexity, are shown in Table IX.

V. CONCLUSION

An efficient transcoder from H.264/AVC to a wavelet-based SVC codec was presented. Its key features were flexible motion approximation techniques able to cope with multiple coding configurations in H.264/AVC, efficient optimization of partition sizes used in motion compensation, and a RC configuration based on H.264/AVC MVs' similarity, CBP information, and adaptive decision on MV refinement.

Thorough evaluation showed that the proposed transcoder offers a performance close to RT-FS, but keeps the complexity much lower than RT-FS and RT-EPZS. On average, the proposed transcoder was 1.57 and 2.68 times faster than RT-HS and RT-EPZS, respectively. With the RC module, it was 2.47 and 4.23 times faster, without a significant impact on quality. Comparing to RT-FS, the PT showed a gain of 0.01 dB, and PT-RC showed a loss of 0.03 dB, on average. In the worst case, the PT showed a loss of 0.12 dB and, with the RC module, a loss of 0.31 dB, comparing to RT-FS.

REFERENCES

[1] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

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

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

[4] 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 Trans. Multimedia*, vol. 11, no. 7, pp. 1209–1223, Nov. 2009.

[5] ITU-T and ISO/IEC, *Advanced Video Coding for Generic Audiovisual Services*, document 14496-10, ITU-T and ISO/IEC Rec. H.264, Nov. 2007.

[6] N. Sprljan, M. Mrak, T. Zgaljic, and E. Izquierdo, *Software Proposal for Wavelet Video Coding Exploration Group*, document M12941, ISO/IEC JTC1/SC29/WG11/MPEG2005, 75th MPEG Meeting, Jan. 2006.

[7] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. V. der Schaar, J. Cornelis, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Process.: Image Commun.*, vol. 19, no. 7, pp. 653–673, Aug. 2004.

[8] N. Adami, M. Brescianini, M. Dalai, R. Leonardi, and A. Signoroni, "A fully scalable video coder with inter-scale wavelet prediction and morphological coding," *Proc. SPIE Vis. Commun. Image Process.*, vol. 5960, no. 1, p. 59601M, Jul. 2005.

[9] S. Issa and O. Khalifa, "Performance analysis of Dirac video codec with H.264/AVC," in *Proc. ICCCE*, May 2010, pp. 1–6.

[10] A. Naman and D. Taubman, "Rate-distortion optimized delivery of JPEG2000 compressed video with hierarchical motion side information," in *Proc. 15th IEEE ICIP*, Oct. 2008, pp. 2312–2315.

[11] A. Naman and D. Taubman, "Rate-distortion optimized JPEG2000-based scalable interactive video (JSIV) with motion and quantization bin side-information," in *Proc. IEEE ICIP*, Nov. 2009, pp. 3081–3084.

[12] A. Naman and D. Taubman, "Predictor selection using quantization intervals in JPEG2000-based scalable interactive video (JSIV)," in *Proc. 17th IEEE ICIP*, Sep. 2010, pp. 2897–2900.

[13] M. Mrak, N. Sprljan, and E. Izquierdo, "Motion estimation in temporal subbands for quality scalable motion coding," *Electron. Lett.*, vol. 41, no. 19, pp. 1050–1051, Sep. 2005.

[14] N. Sprljan, "A flexible scalable video coding framework with adaptive spatio-temporal decompositions," Ph.D. dissertation, Queen Mary, Univ. London, London, U.K., Aug. 2006.

[15] M. Mrak, T. Zgaljic, and E. Izquierdo, "Influence of downsampling filter characteristics on compression performance in wavelet-based scalable video coding," *IET Image Process.*, vol. 2, no. 3, pp. 116–129, Jun. 2008.
[16] N. Sprljan, M. Mrak, and E. Izquierdo, "Motion driven adaptive transform based on wavelet transform for enhanced video coding," in *Proc. 2nd Int. Conf. Mobile Multimedia Commun.*, Jul. 2006, pp. 1–8.
[17] T. Zgaljic, N. Sprljan, and E. Izquierdo, "Bit-stream allocation methods for scalable video coding supporting wireless communications," *Signal Process.: Image Commun.*, vol. 22, no. 3, pp. 298–316, 2007.
[18] E. Peixoto, T. Zgaljic, and E. Izquierdo, "Transcoding from H.264/AVC to a wavelet-based scalable video codec," in *Proc. 17th IEEE ICIP*, Sep. 2010, pp. 2845–2848.
[19] E. Peixoto, T. Zgaljic, and E. Izquierdo, "H.264/AVC to wavelet-based scalable video transcoding supporting multiple coding configurations," in *Proc. PCS*, Dec. 2010, pp. 562–565.
[20] ITU-T, *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264, May 2003.
[21] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Trans. Image Process.*, vol. 8, no. 2, pp. 155–167, Feb. 1999.
[22] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 30–40, Mar. 1999.
[23] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Trans. Multimedia*, vol. 2, no. 2, pp. 101–110, Jun. 2000.
[24] I.-H. Shin, Y.-L. Lee, and H. W. Park, "Motion estimation for frame-rate reduction in H.264 transcoding," in *Proc. 2nd IEEE Workshop Softw. Technol. Future Embedded Ubiquitous Syst.*, May 2004, pp. 63–67.
[25] J.-H. Hur and Y.-L. Lee, "H.264 to MPEG-4 transcoding using block type information," in *Proc. TENCON IEEE Region 10*, Nov. 2005, pp. 1–6.
[26] 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 *Proc. IEEE ISCAS*, Jun. 2010, pp. 3649–3652.
[27] P. Kunzelmann and H. Kalva, "Reduced complexity H.264 to MPEG-2 transcoder," in *Proc. ICCE: Dig. Tech. Papers*, Jan. 2007, pp. 1–2.
[28] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," *Proc. SPIE Conf. Applicat. Digital Image Process. XXVII*, vol. 5558, pp. 454–474, Aug. 2004.
[29] M. Mrak, "Motion scalability for video coding with flexible spatiotemporal decompositions," Ph.D. dissertation, Queen Mary, Univ. London, London, U.K. Jan. 2007.
[30] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.
[31] T. Nguyen and G. Strang, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 2009.
[32] S. T. Hsiang, "Highly scalable subband/wavelet image and video coding," Ph.D. dissertation, Dept. Electric., Comput., Syst. Eng., Rensselaer Polytech. Inst., Troy, NY, Jan. 2002.
[33] T. Zgaljic, "Entropy coding schemes for scalable video coding supporting flexible bit-stream organisation and adaptation," Ph.D. dissertation, Queen Mary, Univ. London, London, U.K., Jun. 2008.
[34] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. New York: Wiley, 2010.
[35] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *IEEE Trans. Consumer Electron.*, vol. 44, no. 1, pp. 88–98, Feb. 1998.
[36] J. Xin, A. Vetro, S. Sekiguchi, and K. Sugimoto, "Motion and mode mapping for MPEG-2 to H.264/AVC transcoding," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Jul. 2006, pp. 313–316.
[37] C. Zhu, X. Lin, L.-P. Chau, K.-P. Lim, H.-A. Ang, and C.-Y. Ong, "A novel hexagon-based search algorithm for fast block motion estimation," in *Proc. IEEE ICASSP*, vol. 3. Jun. 2001, pp. 1593–1596.
[38] J. Xin, J. Li, A. Vetro, H. Sun, and S.-I. Sekiguchi, "Motion mapping for MPEG-2 to H.264/AVC transcoding," in *Proc. IEEE ISCAS*, May 2007, pp. 1991–1994.
[39] Y. Shin, N. Son, N. D. Toan, and G. Lee, "Low-complexity heterogeneous video transcoding by motion vector clustering," in *Proc. ICISA*, Apr. 2010, pp. 1–6.
[40] S. Sharma and K. R. Rao, "Transcoding of H.264 bitstream to MPEG-2 bitstream," in *Proc. APCC*, Oct. 2007, pp. 391–396.
[41] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 614–619, Jul. 2003.
[42] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," *Proc. SPIE Vis. Commun. Image Process.*, vol. 4671, pp. 1069–1079, Jan. 2002.

Eduardo Peixoto (S'09) was born in Brasilia, Brazil. He received the Engineering and M.S. degrees from Universidade de Brasilia, Brasilia, in 2005 and 2008, respectively. He is currently pursuing the Ph.D. degree from Queen Mary, University of London, London, U.K.

In 2008, he was a Research Associate with the DSP Research Group, Universidade de Brasilia. His current research interests include video transcoding, distributed video coding, and scalable video coding.

Toni Zgaljic received the Dipl.Ing. degree in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, and the Ph.D. degree from Queen Mary, University of London, London, U.K., in 2003 and 2008, respectively.

He is currently a Research Assistant with the Multimedia and Vision Group, Queen Mary. His current research interests include scalable video coding and transmission, universal multimedia access, surveillance centric coding, and video transcoding. He has published more than 20 technical papers in these areas, including chapters in books. He was a contributor to the European Project aceMedia and the Surveillance Centric Coding Project—a joint project between United Technology Research Centre and Queen Mary.

Ebroul Izquierdo (M'95–SM'03) received the M.S. degree and the Ph.D. (rerum naturalium) degree from the Humboldt University, Berlin, Germany.

He is the Chair of the Multimedia and Computer Vision and the Head of the Multimedia and Vision Group, School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, U.K. He has been a Senior Researcher with the Heinrich-Hertz Institute for Communication Technology, Berlin, and the Department of Electronic Systems Engineering, University of Essex, Essex, U.K. He holds several patents in the area of multimedia signal processing and has published over 450 technical papers including chapters in books.

Prof. Izquierdo is a Chartered Engineer, a Fellow of the Institution of Engineering and Technology (IET), a member of the British Machine Vision Association, Past Chairman of the IET Professional Network on Information Engineering, a member of the Visual Signal Processing and Communication Technical Committee of the IEEE Circuits and Systems Society, and a member of the Multimedia Signal Processing Technical Committee. He has also been an Associate and Guest Editor of several relevant journals in the field including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the *EURASIP Journal on Image and Video processing*, the Elsevier journal *Signal Processing: Image Communication*, the *EURASIP Journal on Applied Signal Processing*, the IEEE PROCEEDINGS ON VISION, IMAGE, AND SIGNAL PROCESSING, the *Journal of Multimedia Tools and Applications*, and the *Journal of Multimedia*.