A Wyner–Ziv Video Transcoder

Eduardo Peixoto, *Student Member, IEEE,* Ricardo L. de Queiroz, *Senior Member, IEEE,* and Debargha Mukherjee, *Senior Member, IEEE*

***Abstract*—Wyner–Ziv (WZ) coding of video utilizes simple encoders and highly complex decoders. A transcoder from a WZ codec to a traditional codec can potentially increase the range of applications for WZ codecs. We present a transcoder scheme from the most popular WZ codec architecture to a differential pulse code modulation/discrete cosine transform codec. As a proof of concept, we implemented this transcoder using a simple pixel-domain WZ codec and the standard H.263+. The transcoder design aims at reducing complexity as a large amount of computation is saved by reusing the motion estimation, calculated at the side information generation process, and the *I*-frame streams. New approaches are used to generate side information and to map motion vectors for the transcoder. Results are presented to demonstrate the transcoder performance.**

***Index Terms*—Distributed video coding, H.263+, transcoder, Wyner–Ziv coding.**

I. Introduction

VIDEO CODECS traditionally follow the differential pulse code modulation (DPCM)/discrete cosine transform (DCT) model [1]–[4]. At the encoder, extensive computation is made to decide the best modes to predict and to encode each macroblock. The decoder, however, has to reverse the operations and to reconstruct the video sequence. Typically, the encoder is many times more complex than the decoder, mainly due to the motion estimation process [5]–[7]. Hence, traditional codecs are suitable to applications where the video content is encoded once and decoded multiple times, e.g., home digital versatile discs and Blu-ray discs, or decoded by many devices, e.g., broadcasting.

Distributed video coding (DVC) has recently received a lot of attention. While traditional video codecs have a high-complexity encoder and a low-complexity decoder, DVC allows for a shift of complexity from the encoder to the decoder. This shift of complexity makes DVC suitable for encoding video where computational resources are scarce. However, a DVC decoder is not indicated for a constrained resources environment. Usual applications for a DVC codec are in video coding for mobile devices or surveillance cameras, where decoding can be made on a more resourceful machine or offline.

DVC rests on the Slepian–Wolf and Wyner–Ziv (WZ) theorems [8], [9], for lossless and lossy coding of correlated sources, respectively. These results are used in DVC to explore the video statistics at the decoder. Theoretically, a DVC codec can achieve almost the same performance as a traditional codec. In practice, however, traditional codecs outperform DVC codecs. A good review on DVC [10] and some solid frameworks for DVC codecs were proposed before: a pixel-domain WZ codec [11], a transform-domain WZ codec [12], the PRISM framework [13], the Discover codec [14], and other layered frameworks [15]–[17]. In order to enable a better performance, WZ codecs usually have a feedback channel between the encoder and the decoder. An analysis of the feedback channel for pixel-domain WZ codecs can be found elsewhere [18]. Recently, some alternatives for the feedback channel were also studied [19], [20].

An important step for a DVC codec resides in its side information (SI) generation. The purpose of the SI generation is to create, at the decoder, a prediction of the WZ frame sent by the encoder [21], [22]. A good SI generation algorithm might lead to better performance both in quality and rate.

While DVC fits well some low-encoding-complexity applications, it just shifts the complexity to the decoder, which is assumed without constraints. Some DVC schemes encode the WZ frames in layers [15], [17], so that, in a low-resource machine, only the base layer may be decoded. Hence, decoding complexity is largely reduced, at a certain performance penalty.

In order to address the problem where low-complexity is required at both ends, one might use a transcoder from a WZ scheme to a traditional one. A transcoder is also needed when content encoded with a WZ codec has to be transmitted in a network that works with another standard. Furthermore, since WZ codecs usually have a lower performance than traditional codecs, a transcoder could be used for storage purposes. Hence, a transcoder might expand the applications horizon for a WZ codec.

Fig. 1 illustrates the network process when the transcoder is used to provide low-complexity mobile communications. The transmitter encodes the video sequence with a WZ encoder, transmitting the data to a server. The server acts as a transcoder, changing the video sequence to another format, which requires a less complex decoder. The transcoded data are sent to the receiver, the addressee of the video content.

Manuscript received July 16, 2008; revised December 24, 2008. First version published September 1, 2009; current version published February 5, 2010. This paper was recommended by Associate Editor H. Sun. This work was supported by Hewlett-Packard Brazil, and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grant 47.3696/2007-0.

E. Peixoto was with Department of Electrical Engineering, Universidade de Brasilia, and is now with Queen Mary, University of London, London E1 4NS, U.K. (e-mail: eduardopeixoto@ieee.org).

R. L. de Queiroz is with the Department of Electrical Engineering, Universidade de Brasilia, Brasilia, DF 70910-900, Brazil (e-mail: queiroz@ieee.org).

D. Mukherjee is with Hewlett Packard Laboratories, Palo Alto, CA 94304-1126 USA (e-mail: debargha.mukherjee@hp.com).

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.2009.2031374

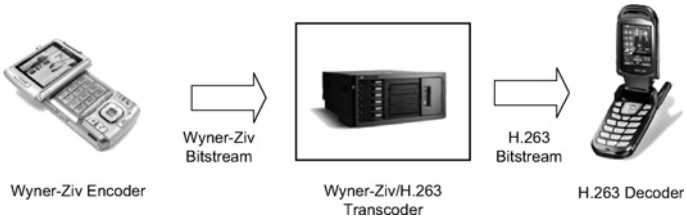

Fig. 1. Mobile video communications architecture.

As a single server might handle several communications, the complexity of the transcoder is an issue and has to be considered in its design.

Even though the need for a transcoder has been mentioned before [10] and a simple implementation has also been presented [23], we propose a more efficient and universal transcoder from a WZ codec to a traditional codec. The transcoder presented could be used in any pixel-domain or transform-domain WZ codecs that follows the Stanford architecture [11], [12], with or without a feedback channel. Also, since the transcoder is mainly concerned with motion estimation, it could also be used to transcode the sequence to a variety of widely used standards, such as moving picture experts group (MPEG)-2 [1], H.263 [3] or H.264/AVC [4]. As a proof of concept, the transcoder is implemented using a simple residual pixel-domain WZ codec and H.263+ as the traditional codec.

WZ codecs with a feedback channel and interactive decoding may increase the overall system delay to impractical levels. Although the rate of the feedback channel is usually small [18], for more practical implementations, WZ codecs without a feedback channel [19], [20] are more suitable for online communications. If one considers that real-time video would allow a few milliseconds to complete the encoding/decoding of one frame, the chain of transmission delays in the channel decoding loop would make it impractical, even without taking into consideration processing delay. However, as a proof of concept, the WZ codec used here is good enough for our purposes and the transcoder could work with a WZ codec without feedback channel as well. Work is under way to remove the feedback channel in the WZ transcoder.

This paper is presented as follows. The codecs used and the SI generation scheme are presented in Section II. In Section III, the transcoder is detailed. Section IV presents the experimental results made to evaluate the transcoder performance, while Section V presents the conclusion of this work.

II. The Codecs Used

The source WZ codec chosen to implement the transcoder is a pixel-domain Wyner–Ziv codec [24], for its simplicity and popularity. Many other WZ implementations are very similar to this one, such as the transform-domain WZ codec, among others [12]. The implementation uses a low-density-parity-check accumulated (LDPCA) code [25] as the channel coder. However, turbo codes or others channel coders could also be used.

The destination codec chosen is H.263+ [26] because it is an efficient international telecommunication union-telecommunication standard, targeted at low-bit rates. However, as mentioned, the transcoder is mainly concerned in reusing motion information and, thus, it could be adapted to other standards, such as MPEG-2 [1] and H.264/AVC [4].

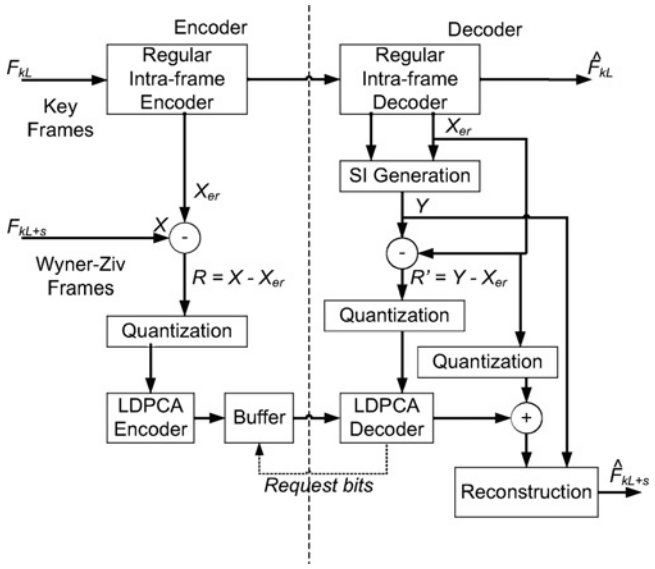

Fig. 2. Wyner–Ziv codec architecture.

The transcoder is designed to generate a fully decodable H.263+ bitstream, so that the transcoding method is transparent to the decoder. Thus, many transcoder options are tied to the options available in the H.263+ standard.

A. Pixel-Domain Wyner–Ziv Codec

The codec architecture is shown in Fig. 2. At the encoder, some frames, known as key frames, are encoded with a regular intraframe encoder (in this case, H.263+ Intra) using quantization parameter QP. These frames are denoted as F_{kL} in Fig. 2, where L is the length of the group of pictures (GOP) [5] and k is an integer. The other frames, called WZ frames, are denoted as F_{kL+s} in Fig. 2, where $0 < s < L$. Any number of WZ frames could be encoded between two key frames. However, the GOP length of a WZ codec is usually 2, 4, and 8, for reasons that will become clear in Section II-F. The more WZ frames are encoded between each key frame, the lower the encoder complexity. However, the correlation of the side information decreases as the GOP length increases, thus reducing the quality of the encoded sequence. For the sake of simplicity, without loss of generality, we assume the GOP length of the WZ codec to be 2 throughout the paper, unless stated otherwise.

The WZ frames are encoded as follows: the difference between the current frame, $X = F_{kL+s}$, and the encoder reference frame, $X_{er} = \hat{F}_{kL}$, is fed to the WZ encoder. X_{er} is taken as the previous reconstructed key frame, which is available to both the encoder and decoder, in order to avoid drifting errors. The pixels of the residual frame $R = X - X_{er}$ are quantized using a uniform scalar quantizer with quantization parameter $QPWZ$. The result is then reordered and encoded by a LDPCA encoder. The code length used is 1584 bits. In order to optimize the performance, different values of $QPWZ$ are used for each QP.

The decoder generates SI for the current frame, denoted as Y. Y can be viewed as a prediction of the current transmitted frame which is available only at the decoder. It can be as simple as to use $Y = X_{er}$ (the previous reconstructed frame), or to use the mean of the previous and the next reconstructed frames. However, there are many techniques in the literature [21], [22] that outperform these simple approaches. In our scheme, we propose a motion-based technique for SI generation that is explained in Section II-B. The residual side information $R' = Y - X_{er}$ is quantized, and the result is fed to the LDPCA decoder. Ideally, the output of the LDPCA decoder is equal to the input of the LDPCA encoder, since the decoder will iterate until the error probability reaches a predefined threshold, which is usually small. The output of the LDPCA decoder is then added to $Q(X_{er})$ in order to find the quantized indexes X_Q. The last step of the decoding process is the pixel reconstruction $X_r = E(X|X_Q = x_Q, Y = y)$. In order to evaluate this expression, the probability density function of the random variable $X|Y$ is needed. Thus, a new random variable Z is defined as $Z = X - Y$. This variable can be well modeled by a Laplacian distribution with zero mean and variance λ. Thus, by definition, $X = Y+Z$. Assuming that Y is independent of Z, the probability density function of $X|Y$ would also follow a Laplacian distribution with mean Y and variance λ. The WZ decoder needs to know λ, since Y is available. Several tests were made with popular sequences to estimate λ, and these values are stored at the decoder as a table. More details of the reconstruction process can be found elsewhere [27].

B. Side Information Generation

The SI generation process plays an important role in any DVC codec. For the transcoder, the SI generation is particularly important. As motion estimation is performed to generate the SI estimation for the current frame, we reuse the motion estimation information to generate the motion vectors for the transcoded sequence. In our WZ codec, there is no motion estimation at the encoder.

There are many works on SI generation methods [21], [22]. We will focus on two of them. Both (and the proposed method) use motion estimation over neighboring key frames to generate a frame in between them.

In order to explain the motion estimation process between the two key frames, let x be the sequence of decoded key frames and $x(\mathbf{p}, k)$ be a pixel in it, where $\mathbf{p} = [p_i, p_j]$ is the spatial location and k is the temporal location (i.e., the frame number). As the key frames are encoded without use of the WZ frames, we can change the decoding order so that the two adjacent key frames are decoded first. Each frame can be divided into macroblocks of 16×16 pixels. Thus, $x(16\mathbf{m} + \mathbf{n}, k)$ represents each macroblock within the k-th frame, where $\mathbf{m} = [m_i, m_j]$ is the macroblock index and $\mathbf{n} = [n_i, n_j]$ is the pixel index within the block.

The motion vectors $\mathbf{v_m}$ are chosen as the displacement vector that minimizes

$$\mathrm{SAD} = \sum_{\mathbf{n}} |\mathrm{x}(16\mathbf{m} + \mathbf{n}, \mathrm{k} + 1) - \mathrm{x}(16\mathbf{m} + \mathbf{n} - \mathbf{v}_{\mathrm{m}}, \mathrm{k} - 1)| \tag{1}$$

where SAD stands for sum of absolute differences for the $\mathbf{m}$-th block. Note that $x(\mathbf{p}, k + 1)$ is the next key frame and $x(\mathbf{p}, k - 1)$ is the previous key frame, and we want to generate the SI frame $y(\mathbf{p}, k)$ in between them. $y(\mathbf{p}, k)$ could be made equal to $x(\mathbf{p}, k - 1)$, i.e., the SI frame would be the previous key frame. However, as it does not use other available information, it is not the best SI frame.

C. Side Estimatior B (SE-B)

The method called Side Estimatior B (SE-B) [21] models the motion vector for the current WZ frame using the motion vectors calculated between the two adjacent key frames. Using $x(\mathbf{p}, k)$ as source and $x(\mathbf{p}, k-1)$ as reference, we calculate the motion vectors set MV_F. Then, for each block in the current frame $y(\mathbf{p}, k)$, it uses half of the motion vector of the co-located block as reference. So, the SI frame is defined by

$$y(16\mathbf{m} + \mathbf{n}, k) = x\left(16\mathbf{m} + \mathbf{n} - \left\lfloor \frac{\mathbf{v}_m}{2} \right\rfloor, k - 1\right) \tag{2}$$

where $\lfloor\ \rfloor$ is the rounding operator and $\mathbf{v_m}$ is the motion vector between frames $k - 1$ and $k + 1$ for the $\mathbf{m}$-th block. SE-B can also perform backwards motion estimation to improve the quality of the SI frame [21].

D. Frame Interpolation with Spatial Motion Smoothing

A method was proposed [22] in which both key frames are low-pass filtered prior to motion estimation in order to improve motion vector reliability. Then, forward and bidirectional motion estimation are used. An additional step of spatial smoothing can also be used [22]. Finally, bidirectional motion compensation is carried to generate the SI frame.

In the motion estimation process used in that method [22], both the reference block and the motion vector are changed. When modeling the motion between two key frames to interpolate the SI frame, overlapped and uncovered areas might appear. This is because the motion vectors do not necessarily intercept the interpolated frame at the center of each nonoverlapped block of this frame. So, among the motion vectors that intercept each block, this method chooses the one which is closest to the center of each block. At this point, the SI frame is defined by

$$y(16\mathbf{m} + \mathbf{n}, k) = x(16\mathbf{m} + \mathbf{n} - \mathbf{v}_i, k - 1) \tag{3}$$

where $\mathbf{v}_i$ is the motion vector closest to the center of the $\mathbf{m}-th$ block of the SI frame. This method changes the motion vector in order to fill the SI frame without overlapping or uncovered areas. Also, this method uses spatial smoothing and motion vector refinement techniques to further enhance the quality of the SI frame.

E. Proposed Method

The proposed method models the motion between two key frames $x(\mathbf{p}, k - 1)$ and $x(\mathbf{p}, k + 1)$ as linear. Thus, the motion between $x(\mathbf{p}, k - 1)$ and the current frame $x(\mathbf{p}, k)$ would be half of the motion between $x(\mathbf{p}, k - 1)$ and $x(\mathbf{p}, k + 1)$. For a given macroblock in $x(\mathbf{p}, k+1)$, it searches the reference frame $x(\mathbf{p}, k - 1)$ to find the best match, named the reference block. This reference block is kept and translated by $MV_F/2$. This

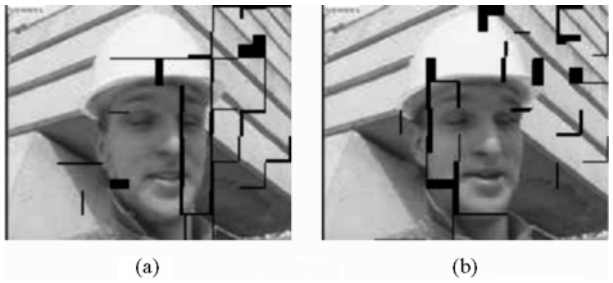

Fig. 3. Generating the SI frame using both (a) forward $MV_F/2$ and (b) backward $MV_B/2$ estimation for the second frame of *Foreman* sequence.

Fig. 4. Combining the frames in Fig. 3.

approach leads to overlapping and blank areas. The SI frame is defined as

$$y\left(16\mathbf{m}+\mathbf{n}-\left\lfloor\frac{\mathbf{v}_m}{2}\right\rfloor,k\right)=x\left(16\mathbf{m}+\mathbf{n}-\mathbf{v}_m,k-1\right). \quad (4)$$

Hence, there are three cases for a given pixel:

1) it is uniquely defined by a single motion vector;
2) it is defined by more than one motion vector (an overlapping occurred);
3) it is not defined by any motion vector (it is left blank).

In order to assign a motion vector or filling process for every pixel, when more than one option for a pixel exists, a simple averaging might suffice. The last case is more challenging. If no motion vector points to a pixel, then it is not easy to guess its value. Using the co-located pixel in the previous frame is not very efficient due to the motion. In Fig. 3(a), it is shown the estimated second frame of the *Foreman* sequence using $MV_F/2$. In this case, the key frames are in quarter common intermediate format (QCIF) resolution and were coded with H.263+ Intra with $QP = 4$. The overlapping areas were averaged and, as expected, there are some blank areas. However, in Fig. 3(b), it is shown the estimated frame using $MV_B/2$, which is half the motion vector calculated using $x(\mathbf{p}, k+1)$ as reference and $x(\mathbf{p}, k-1)$ as source. Again, there are some blank areas, but most of them are in different places, as compared to Fig. 3(a).

Combining the forward with the backward estimations will result in a frame with less blank areas, which is depicted in Fig. 4. Such a combination is simply an average between both estimations. In regions with blank areas in only one

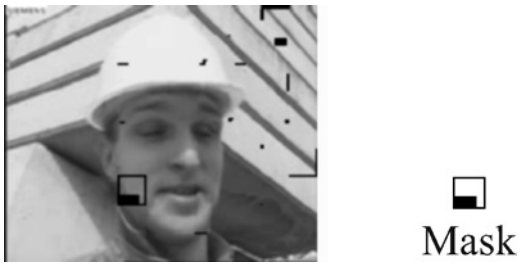

Fig. 5. Performing motion estimation using the current SI frame.

Fig. 6. Final SI frame. PSNR = 33.38 dB.

estimation, the pixels are taken from the other one. If, after joint compensation, the SI frame is still left with some blank areas, the frame is divided into blocks of 16×16 pixels, in case the frames are in QCIF resolution, or into blocks of 32×32, in case the frames are in common intermediate format resolution. Then, if there is a blank area in a macroblock, bidirectional motion estimation is performed for this macroblock, as illustrated in Fig. 5. The blank area is not considered when calculating the SAD. Once the new reference block is found, it is used to replace the macroblock with the gaps. However, only the blank area is replaced (filled). The remaining macroblock pixels are left untouched. The frame in Fig. 5 after processing is shown in Fig. 6.

In summary, the reference block found using the motion estimation process is kept and translated to the SI frame by a motion vector that is half the original motion vector. In SE-B, the reference block is changed while the motion vector is kept. In the method in Section II-D, motion vectors are changed to point to the middle of the current block in the SI frame in order to prevent the uncovered and overlapping areas.

F. Generating SI for Larger GOPs

In the previous sections, it was discussed how to generate the SI frame for a GOP length of 2, i.e., only one WZ frame between two key frames. The more WZ frames are encoded between two key frames, the higher the complexity reduction of the codec. However, as the SI frames become less reliable, the codec performance is also reduced.

Besides the basic GOP length of 2, two others GOP lengths are commonly used in the WZ literature: 4 and 8. The reason for this becomes clear when the technique to generate the SI frames for these GOPs is explained. For these larger GOPs,

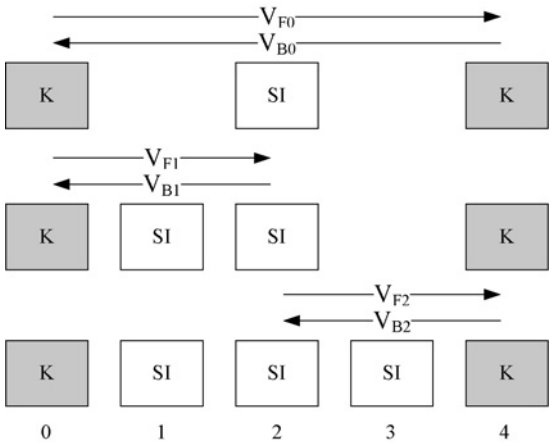

Fig. 7. Generating SI for GOP length of 4.

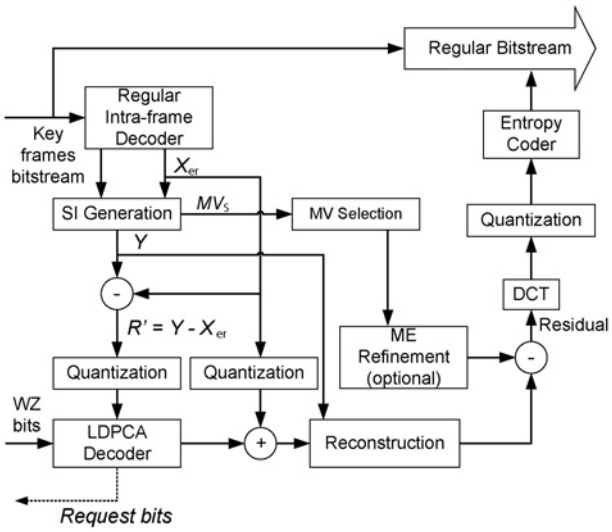

Fig. 8. Wyner–Ziv transcoder architecture.

the SI frames are generated in an iterative way that can be seen in Fig. 7 for the GOP length of 4.

For the GOP length of 4, first, the motion vectors between the two key frames (F_{4K} and $F_{4(K+1)}$) are calculated and one SI frame is generated in the middle temporal position (i.e., $F_{4(K+2)}$). Then, a new SI frame is generated using the previous key frame (F_{4K}) and this SI frame ($F_{4(K+2)}$). Naturally, the SI frame generated is in position $F_{4(K+1)}$. Finally, the remaining SI frame ($F_{4(K+3)}$) is generated using the first SI frame ($F_{4(K+2)}$) and the next key frame ($F_{4(K+1)}$). For the GOP length of 8, the same process is applied, with more steps.

III. The Transcoder

The WZ–H.263+ transcoder diagram is depicted in Fig. 8. Instead of re-encoding the sequence, the transcoder uses information that was calculated in the WZ decoding process to speed up the transcoding. The main ideas are to copy the bitstreams of the intraframes that will remain *I*-frames in the transcoded sequence, and to reuse the motion estimation performed in the SI generation process instead of performing a new motion estimation [28].

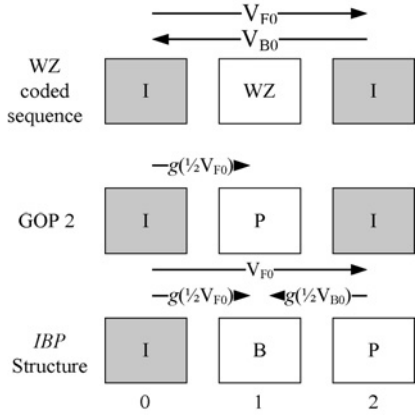

Fig. 9. Options of the GOP for the transcoded sequence starting with a WZ GOP of two frames. The motion vectors are mapped using a function $g(\cdot)$, defined in (6). The others are calculated prior to the SI generation process.

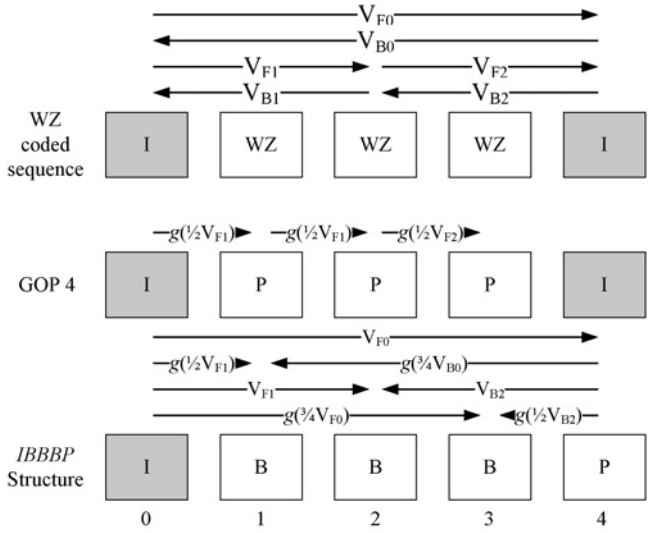

Fig. 10. Options of the GOP for the transcoded sequence starting with a WZ GOP of four frames. The motion vectors are mapped using a function $g(\cdot)$, defined in (6). The others are calculated prior to the SI generation process.

There are two main degrees of freedom for the transcoder: the GOP structure and the motion vector mapping.

A. Defining the Transcoder GOP

The GOP size of the transcoded sequence does not need to be of the same size as the original WZ sequence. Instead, it can be changed to a wide range of GOPs. Any GOP size which is an integer multiple of the original GOP size is possible. It is also possible to use *B*-frames in the transcoded sequence.

Some of these options are depicted in Fig. 9. In the top row, we illustrate a WZ coded sequence. The key frames were encoded as H.263+ Intra frames by the WZ encoder, and their bitstreams are to be passed through virtually untouched. The motion vectors are mapped using a function $g(\cdot)$ which will be explained in Section III-B.

If the transcoded sequence has a GOP length of 2, it has the form of the second row of Fig. 9. The *I*-frames have the same structure of its pairs in the WZ coded sequence and are just passed through. The last row in Fig. 9 refers to the use of

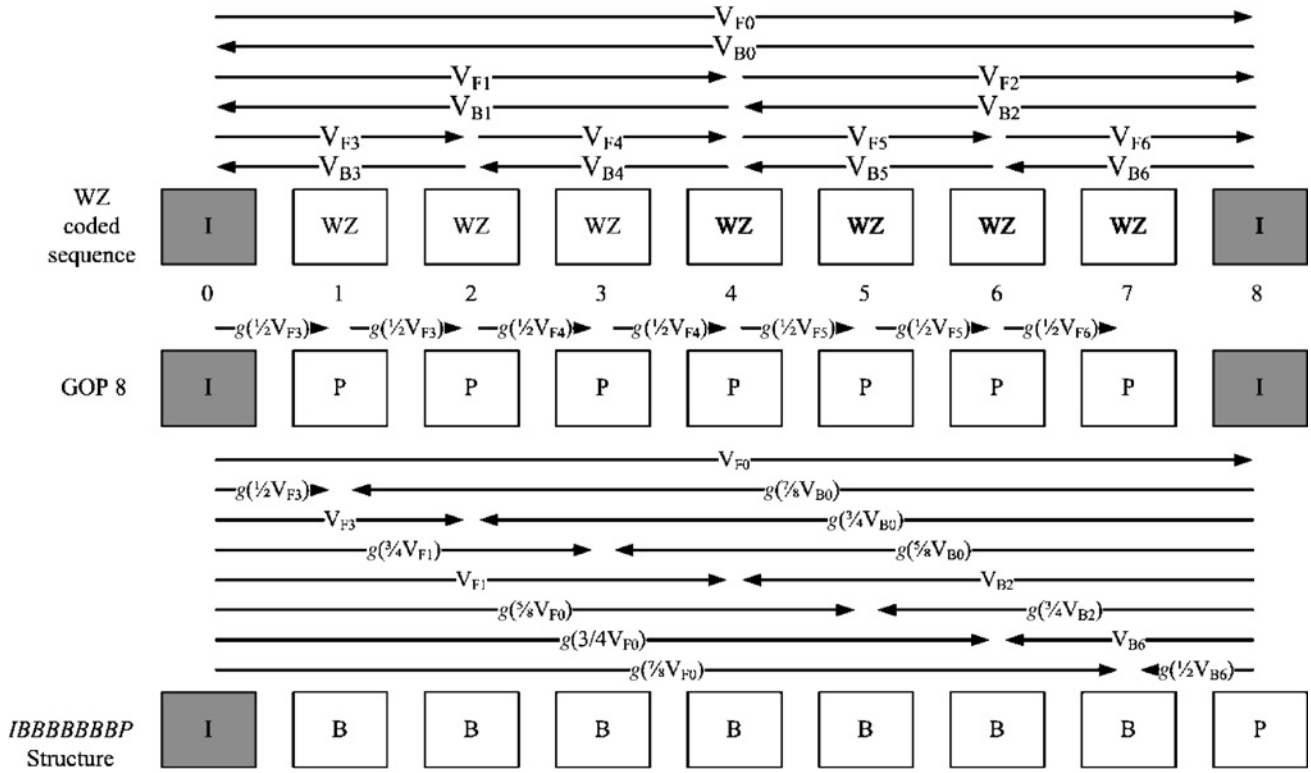

Fig. 11. Options of the GOP for the transcoded sequence starting with a WZ GOP of eight frames. The motion vectors are mapped using a function $g(\cdot)$, defined in (6). The others are calculated prior to the SI generation process.

B-frames. As the SI generation process calculates forward and backward motion vectors, it is easy to relate them to *B*-frames. For a GOP length of 2 in the WZ coded sequence, the motion vectors can be used to generate a stream of *IBPBP...*-frames. For such a structure, the motion vectors for the *P*-frames are readily available from the SI generation process. The *B*-frames, however, require a more sophisticated approach, which will be described in Section III-B.

Following the same idea, Figs. 10 and 11 show how to use the motion vectors for the transcoder if the WZ coded sequence has a GOP lengths of 4 and 8, respectively. The second row shows the motion vectors used if the transcoder uses the same GOP length as the WZ coded sequence and the third row shows the motion vectors when *B*-frames are used. In these cases, the transcoder always use the motion vectors calculated for the same frames when they are available. When they are not, the transcoder maps the nearer motion vectors using a function $g(\cdot)$ which will be explained in Section III-B.

B. Mapping Motion Vectors

H.263+ does motion estimation by minimizing (1) and uses the motion vectors to generate the compensated frame, which is used as a prediction to encode the current frame.

There is motion information, generated by the WZ decoding process. For example, using a GOP length of 2 for the WZ sequence, we have the best set of motion vectors calculated using the previous key frame ($2k$) as reference and the next key frame ($2k+2$) as source. So, an idea would be to use this set of motion vectors and apply it to the WZ frame ($2k+1$) as the source with the same previous key frame ($2k$) as reference. Recall that the SI frame y is constructed using (4).

We want to create a grid for the SI frame and to try to decide, for each macroblock, the best motion vector among those available within it. Note that the spatial position $16\mathbf{m}+\mathbf{n}-\lfloor \mathbf{v_m}/2 \rfloor$ does not necessarily fit the macroblock grid for the SI frame y. So, overlaps may occur and the same pixel can be pointed by two or more motion vectors. Also, different regions of the same macroblock might be filled by different motion vectors, making it difficult to decide which motion vector to use at the transcoder.

Stating the problem: for a given macroblock, there are pixels with zero, one or more motion vectors. Yet, we have to decide for a single motion vector $\hat{\mathbf{v}}$ for the whole macroblock.

We propose to perform a weighted mean for the motion vectors within a macroblock, that is

$$\mathbf{v}_{\text{mean}} = \frac{\sum_n \omega_n \cdot \mathbf{v}_n}{\sum_n \omega_n}. \tag{5}$$

The weight ω_n reflects the category of the pixel: if it is not defined, uniquely defined, or multiply defined. They are used to count the number of motion vectors considered in a given macroblock—as some pixels are not defined while others are multiply defined, it is important to count the number of considered motion vectors so that the average is calculated accordingly. In our experiments, we only counted uniquely defined pixels, i.e., $\omega_n = 1$ for this case, or zero, otherwise.

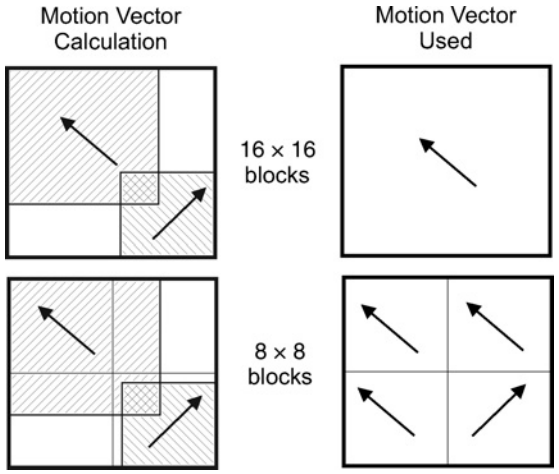

Fig. 12. Calculating the motion vectors for the transcoder. When there are overlapping areas, the case of 8 × 8 blocks allows for extra flexibility when choosing the output motion vector.

However, it is possible that $\mathbf{v}_{\text{mean}}$ does not represent any group of motion vectors. So, we are looking for a convex solution, in the form of

$$\hat{\mathbf{v}} = g(\mathbf{v}_l) = \{\mathbf{v}_l | l = min(\|\mathbf{v}_{\text{mean}} - \mathbf{v}_l\|)\}. \quad (6)$$

In other words, $\hat{\mathbf{v}}$ is the motion vector in the group that is closest to $\mathbf{v}_{\text{mean}}$.

Note that the function $g(\cdot)$ operates on the motion vector set scaled to the represent the motion vector for a given frame. For example, when used in a GOP length of 2, the motion vectors $\mathbf{v}_l$ were calculated using the frames in positions $2k$ and $2k + 2$. Thus, what is used for the WZ frame in position $2k + 1$ is the motion vector $g(1/2 * \mathbf{v}_l)$. This can be seen in Fig. 9.

An advantage of this process is that it assigns motion vectors based on pixels, not blocks. So, we can group the motion vectors in different ways. For example, we can group the motion vectors in 8 × 8 blocks even though they were originally calculated in 16 × 16 blocks. The 4 motion vectors for 8 × 8 blocks inside a 16 × 16 block are not necessarily equal to the single motion vector for the 16 × 16 block, since overlaps may have occurred in this area. Hence, we can have an 8 × 8 motion vector set with a search performed within 16 × 16 blocks. This is depicted in Fig. 12.

If the WZ frame is transcoded to a *B*-frame, both the forward and backward motion vectors are chosen using (6). In all cases, the transcoder uses the best motion vector between $\hat{\mathbf{v}}$ and the null motion vector [i.e., the (0, 0) one].

For some macroblocks, there is still another motion vector to consider. Back to the SI generation in Section II-E, in order to fill the blank areas, a new motion estimation is carried for the macroblocks in which they appear. Then, for these macroblocks, we did carry motion vector search. This motion vector might be a better choice than the motion vector given by (6), provided this information has already been calculated in the SI generation process. Thus, for those macroblocks, said motion vector is used instead of $\hat{\mathbf{v}}$.

C. Refining the Motion Vectors

A good way to improve the motion vectors inherited from the WZ decoding process is to use them as seeds for a fast motion estimation method, such as a diamond or hexagonal search. Instead of always starting at motion vector (0, 0), the refinement would begin at the motion vector calculated in Section III-B. This seed is compared (in terms of SAD) with the null motion vector, and the search starts at the best of these two. This approach has proven to lead to the testing of less candidates.

IV. Experimental Results

A. Rate-Distortion Performance

We evaluated the proposed transcoder using the WZ codec detailed in Section II-A and the standard H.263+. For all tests, the peak signal-to-noise ratio (PSNR) is shown as an average of the PSNR of the luminance component only, while the rate also considers the chrominance components. For the sequences with 30 frames/s, the first 299 frames of each sequence were used for the GOP length of 2, while the GOP length of 4 uses the first 297 frames were used. For the sequences with 15 frames/s, the GOP lengths of 2 and 4 use the first 149 frames, while the GOP length of 8 uses the first 145 frames. The test sequences at 15 frames/s were obtained by discarding every even frame of the original sequences at 30 frames/s.

A more detailed evaluation of the SI generation method used here can be found elsewhere [29]. The proposed method performs better than SE-B, but it is outperformed by frame interpolation with spatial smoothing and motion vector refinement [22]. However, the proposed method is slightly more complex than SE-B and significantly less complex than the frame interpolation technique. Nevertheless, spatial motion smooothing and motion vector refinement can also be incorporated into the present framework, potentially increasing its performance. As complexity is a key issue for the transcoder, we chose to use the motion model approach in this paper.

Althought it is not the focus of this paper, Fig. 13 shows the results for the WZ codec, both for QCIF and CIF resolutions. The WZ codec is compared against H.263+ Intra, H.263+ *IPIP* with the same GOP length of the WZ codec performing full motion estimation and H.263+ *IPIP* with twice this GOP length with zero motion vectors. Note that the WZ codec implementation is not the state of the art WZ codec, but a good proof of concept for the transcoder.

As it can be seen in Fig. 13, the pixel-domain WZ codec shows good performance for sequences with low motion, such as *Salesman*. For higher motion sequences the performance is inferior, but it is still better than H.263+ Intra. This is a common result for WZ codecs since the performance is heavily dependent on the SI, which becomes less reliable for sequences with fast motion.

Tests were made to evaluate the transcoder effectiveness, comparing three options: 1) the trivial transcoder (i.e., the tandem connection of a decoder with a full encoder) with full motion estimation; 2) the proposed transcoder; and 3) the proposed transcoder with motion vectors refinement. For each

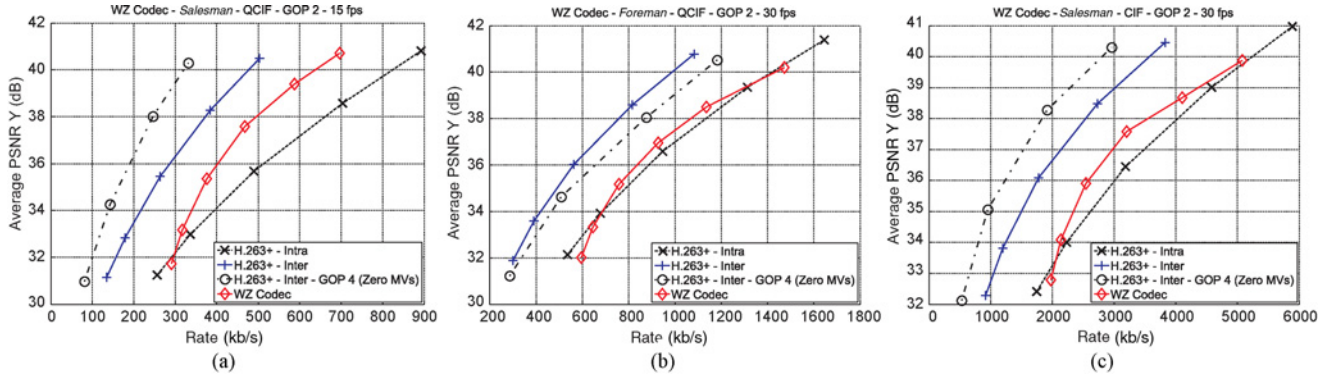

Fig. 13. WZ codec performance for different resolutions. (a) *Salesman* QCIF 15 frames/s. (b) *Foreman* QCIF 30 frames/s. (c) *Salesman* CIF 30 frames/s.

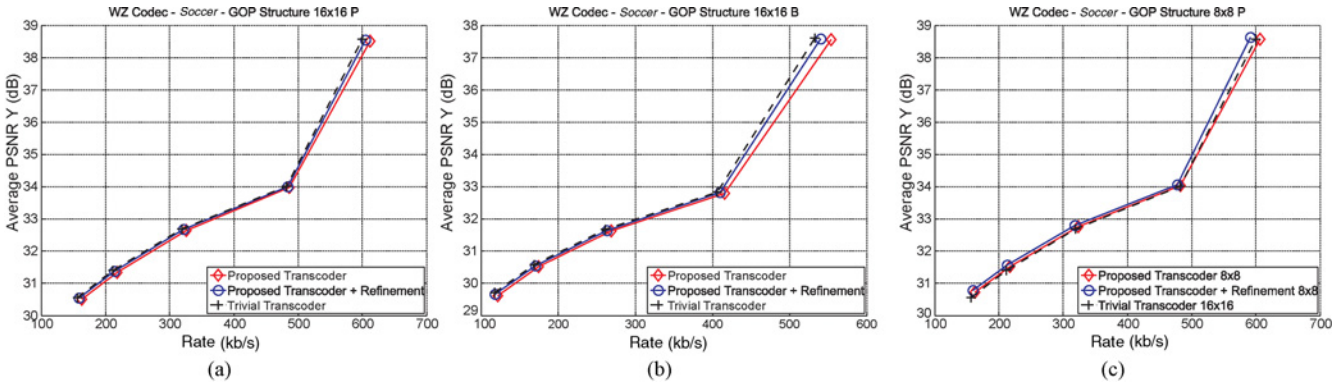

Fig. 14. Results of the transcoder for the *Soccer* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 2. (a) GOP: *IPIP*...; (b) GOP: *IBPBP*...; and (c) GOP: *IPIP*.. using 8×8 motion vectors.

case, the motion vectors for the H.263+ stream are chosen as follows.

1) The integer precision motion vectors are chosen with a full search with search range $[-16, 15]$ and half-pixel refinement on the eight neighbors of the chosen motion vector.
2) The transcoder tests two motion vectors (the null motion vector and $\hat{\mathbf{v}}$) and performs the half-pixel refinement on the eight neighbors of the best of these two.
3) The transcoder performs a diamond refinement starting at the null motion vector or $\hat{\mathbf{v}}$ (whichever has a lower SAD), and then performs the same half-pixel refinement.

The results for the transcoder are shown in Figs. 14–19. As it can be seen, the proposed transcoder closely matches the performance of the trivial transcoder for all sequences and GOP structures tested, and also for QCIF and CIF resolutions. For shorter GOP lengths and sequences with low motion, the optional diamond refinement leads to a very small gain. However, for higher GOP lengths and sequences with high motion, this refinement yields larger gains, allowing for the proposed transcoder virtually the same performance as the trivial transcoder.

The proposed transcoder is also tested using the optional 8×8 motion vectors in an *IPIP* GOP structure. Since the H.263+ standard does not allow 8×8 motion vectors for *B*-frames, this option was not used in the *IBPBP*... GOP structure. The results are shown in Figs. 14(c)–19(c). In this case, the proposed transcoder has a performance very close to the trivial transcoder. For some sequences, the latter is even outperformed by the former. As no search is made in 8×8 blocks, the complexity of the transcoder does not change significantly when using this mode. Therefore, when the 8×8 mode is available, it should be used instead of the 16×16 one.

It can be seen that the proposed transcoder performance is virtually the same to that of the trivial transcoder with full search motion estimation for all the sequences and GOP structures tested.

B. Complexity Analysis

In a simplified point of view, the trivial transcoder has to perform the following tasks: 1) key frame decoding; 2) SI generation; 3) channel decoder; 4) reconstruction; 5) motion estimation; and 6) regular encoding (which encompasses transform and entropy coding, among other things). Of these tasks, three of them are significantly more complex than the others: SI generation, channel decoder, and motion estimation. Let C_{SI}, C_{CD}, and C_{ME} be the complexity of these tasks, respectively. Considering the transcoding from a WZ codec with GOP length of 2 to H.263+ with GOP length of 2, we have the following complexity for the trivial transcoder (denoted as C_{TT} and measured in numbers of operations per frame):

$$C_{TT} \approx \frac{1}{2}C_{SI} + \frac{1}{2}C_{CD} + \frac{1}{2}C_{ME}. \tag{7}$$

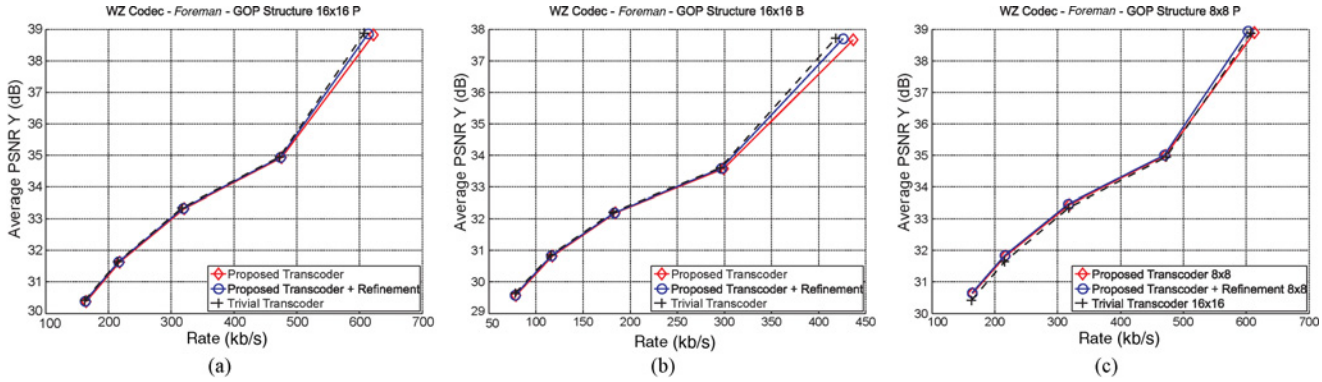

Fig. 15. Results of the transcoder for the *Soccer* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 2. (a) GOP: *IPIP...*; (b) GOP: *IBPBP...*; and (c) GOP: *IPIP...* using 8×8 motion vectors.

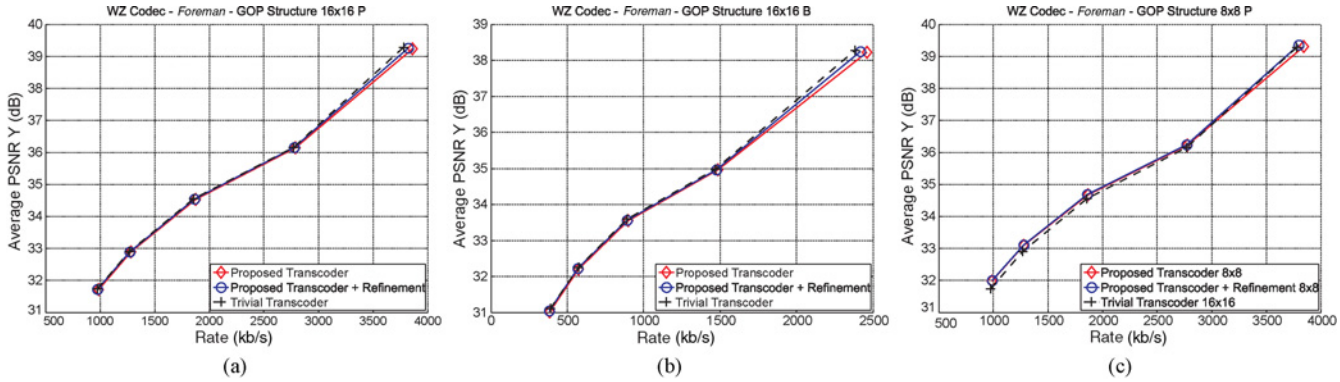

Fig. 16. Results of the transcoder for the *Foreman* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 2. (a) GOP: *IPIP...*; (b) GOP: *IBPBP...*; and (c) GOP: *IPIP...* using 8×8 motion vectors.

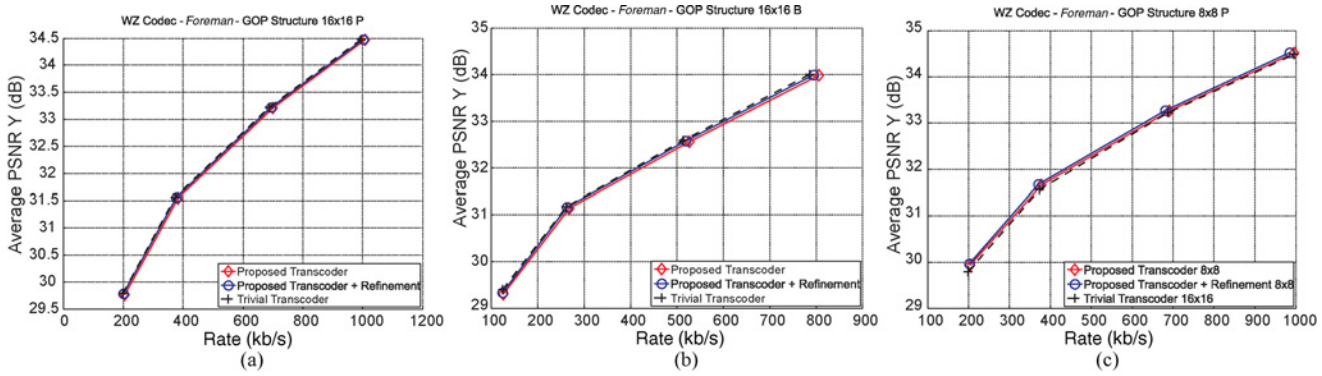

Fig. 17. Results of the transcoder for the *Foreman* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 2. (a) GOP: *IPIP...*; (b) GOP: *IBPBP...*; and (c) GOP: *IPIP...* using 8×8 motion vectors.

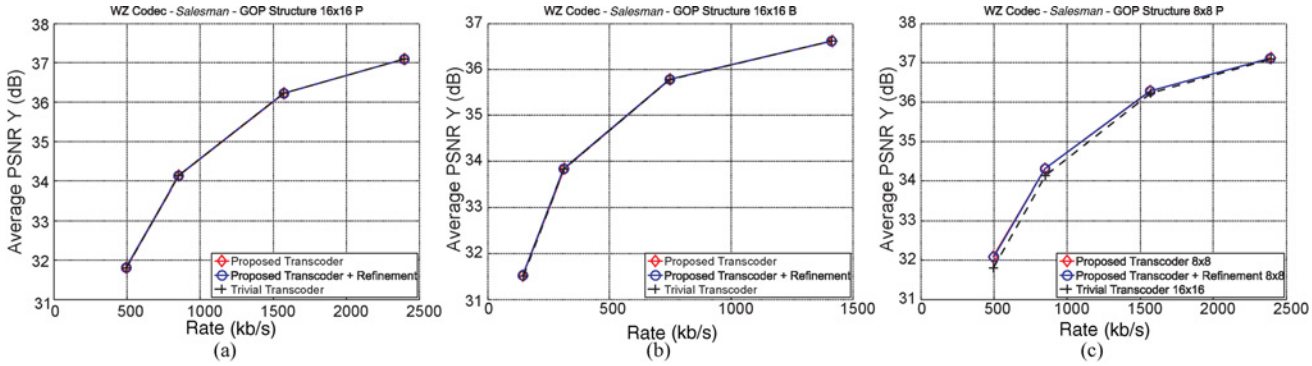

Fig. 18. Results of the transcoder for the *Salesman* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 8. (a) GOP: *IPPPPPPPI...*; (b) GOP: *IBBBBBBBP...*; and (c) GOP: *IPPPPPPPI...* using 8×8 motion vectors.

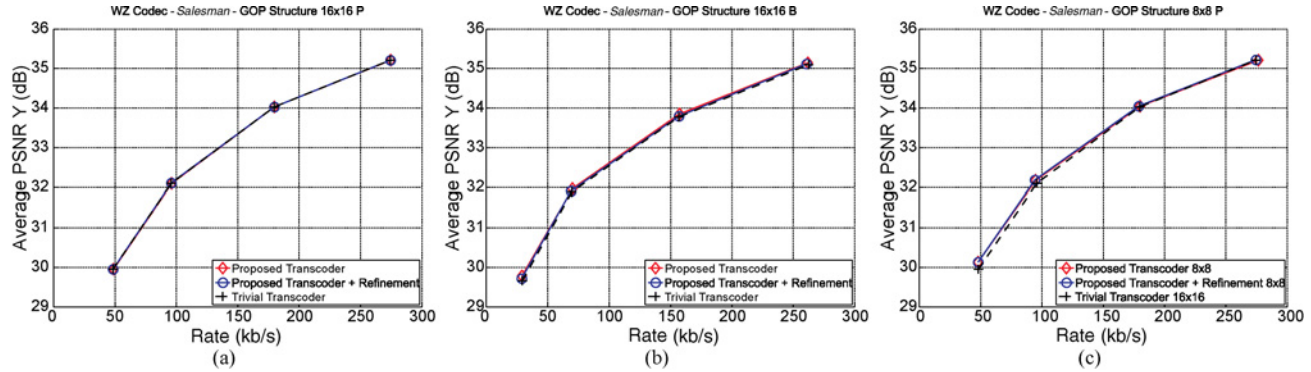

Fig. 19. Results of the transcoder for the *Salesman* sequence at QCIF resolution and 15 frames/s using different GOP structures. The original WZ coded sequence has a GOP length of 8. (a) GOP: *IPPPPPPPI...*; (b) GOP: *IBBBBBBBP...*; and (c) GOP: *IPPPPPPPI...* using 8×8 motion vectors.

Assuming that $C_{SI} \approx 2 \cdot C_{ME}$ (since the main operations involved in the SI generation are two motion estimations, one for the previous key frame and the other for the next key frame), we have

$$C_{TT} \approx \frac{1}{2} C_{CD} + \frac{3}{2} C_{ME}. \tag{8}$$

The proposed transcoder does not have a motion estimation step. Instead, it has a motion vector mapping step and, optionally, a refinement step. Both of these tasks are significantly less complex than SI generation and channel decoder. Thus, the proposed transcoder, for the above-mentioned conditions, would have

$$C_{PT} \approx \frac{1}{2} C_{CD} + 1 C_{ME} \tag{9}$$

where C_{PT} is the complexity of the proposed transcoder, which saves $1/2C_{ME}$ per frame. However, if we consider the transcoding from GOP 2 to a *IBPBP...* structure, we would have, for the trivial transcoder (considering that the complexity cost of a *B*-frame is twice the complexity cost of a *P*-frame)

$$C_{TT} \approx \frac{1}{2} C_{CD} + \frac{5}{2} C_{ME} \tag{10}$$

and for the proposed transcoder

$$C_{PT} \approx \frac{1}{2} C_{CD} + 1 C_{ME} \tag{11}$$

resulting in savings of the order of $3/2C_{ME}$ per frame. This is because the proposed transcoder has almost constant complexity regardless of the GOP structure.

V. Conclusion

We proposed a transcoder from streams generated by WZ codecs to those generated by traditional DPCM/DCT video codecs. As a proof of concept, the transcoder was implemented and tested using a simple pixel-domain WZ codec and H.263+ as the regular codec. However, the same transcoder architecture can be used by other popular WZ codecs, such as the Discover codec, and can also be modified to work with other standards, such as MPEG-2 and H.264, since the proposed transcoder is mainly based on reusing motion information.

The performance of the proposed transcoder is virtually the same as the performance of decoding and re-encoding the whole sequence, at a much lower computational cost. For a GOP structure of *IPIP*, the 8×8 transcoding mode even outperforms the regular 16×16 full search mode, without additional complexity cost. The transcoder is also adjustable for different complexity requirements, being able to reduce the bit rate of the transcoded sequence by changing the transcoded frame structure.

In the implementation here presented, many transcoding options are tied to the options of the H.263+ standard, such as the use of 8×8 motion vectors only for *P* frames. As future

work, it is planned to move the WZ codec and the transcoder toward the H.264 standard and also to remove the feedback channel. Furthermore, we plan to make block sizes adaptive in the SI generation method.

REFERENCES

[1] *Generic Coding of Moving Pictures and Associated Audio Information*, document ISO/IEC 13818, ISO/IEC JTC 1/SC 29, 1995.

[2] *Coding of Audio-Visual Objects–Part 2: Visual*, document ISO/IEC 14496-2, ISO/IEC JTC 1/SC 29, 2001.

[3] *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263, 1996.

[4] *Advanced Video Coding for Generic Audiovisual Services*, document ISO/IEC 14496-10 and ITU-T Recommendation H.264, Advanced Video Coding, 2003.

[5] K. R. Rao and J. J. Hwang, "Hybrid coding and motion compensation," in *Techniques and Standards for Image, Video and Audio Coding*, Upper Saddle River, NJ: Prentice-Hall, 1996, pp. 85–96.

[6] J. D. Gibson, T. Berger, T. Lookabaugh, R. Baker, and D. Lindbergh, "MPEG compression," in *Digital Compression for Multimedia: Principles and Standards*, San Francisco, CA: Morgan Kaufmann, 2006, pp. 376–384.

[7] I. E. G. Richardson, "Video coding concepts," in *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*, Chichester, West Sussex, U.K.: Wiley, 2003, pp. 30–41.

[8] J. D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[9] A. D. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, pp. 1–10, Jan. 1976.

[10] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," in *Proc. IEEE Special Issue Adv. Video Coding Deliv.*, 2005, pp. 1–12.

[11] A. Aaron, R. Zhang, and B. Girod, "Wyner–Ziv coding of motion video," in *Proc. 36th Asilomar Conf. Signals, Syst. Comput.*, vol. 1. Pacific Grove, CA, Nov. 2002, pp. 240–244.

[12] A. Aaron, S. Rane, E. Setton, and B. Girod, "Transform-domain Wyner–Ziv codec for video," in *Proc. Soc. Photo-Opt. Instrum. Eng. Visual Commun. Image Process.*, vol. 5308. San Jose, CA, Jan. 2004, pp. 520–528.

[13] R. Puri, A. Majumdar, and K. Ramchandran, "PRISM: A video coding paradigm with motion estimation at the decoder," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2436–2448, Oct. 2007.

[14] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The Discover codec: Architecture, techniques and evaluation," *Picture Coding Sympos.*, vol. 17, no. 9, pp. 1103–1120, Nov. 2007.

[15] Q. Xu and Z. Xiong, "Layered Wyner–Ziv video coding," in *Proc. Soc. Photo-Opt. Instrum. Eng. Visual Commun. Image Process.*, vol. 5308. San Jose, CA, Jan. 2004, pp. 83–91.

[16] H. Wang, N.-M. Cheung, and A. Ortega, "A framework for adaptive scalable video coding using Wyner–Ziv techniques," *Eur. Assoc. Signal Image Process. J. Appl. Signal Process.*, vol. 2006, no. 5, pp. 1–18, Jan. 2006.

[17] D. Mukherjee, B. Macchiavello, and R. L. de Queiroz, "A simple reversed-complexity Wyner–Ziv video coding mode based on a spatial reduction framework," in *Proc. Soc. Photo-Opt. Instrum. Eng. Visual Commun. Image Process.*, vol. 6508. San Jose, CA, Jan. 2007, pp. 1–12.

[18] C. Brites, J. Ascenso, and F. Pereira, "Feedback channel in pixel domain Wyner–Ziv video coding: Myths and realities," in *Proc. 14th Eur. Signal Process. Conf.*, Florence, Italy, Sep. 2006.

[19] M. Morbée, J. Prades-Nebot, A. Pizurica, and W. Philips, "Rate allocation algorithm for pixel-domain distributed video coding without feedback channel," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 1. Honolulu, HI, Apr. 2007, pp. 521–524.

[20] T. Sheng, G. Hua, H. Guo, J. Zhou, and C. W. Chen, "Rate allocation for transform domain Wyner Ziv video coding without feedback," in *Proc. 16th Assoc. Comput. Mach. Int. Conf. Multimedia*, 2008, pp. 701–704.

[21] Z. Li and E. J. Delp, "Wyner–Ziv side estimator: Conventional motion search methods revisited," *Proc. Int. Conf. Image Process.*, vol. 1, no. 1, pp. 825–828, Sep. 2005.

[22] J. Ascenso, C. Brite, and F. Pereira, "Improved frame interpolation with spatial motion smoothing for pixel domain distributed video coding," in *Proc. 5th Eur. Assoc. Signal Image Process. Conf. Speech Image Process. Multimedia Commun. Services*, Smolenice, Slovakia, Jun.–Jul. 2005, pp. 1–6.

[23] E. Peixoto, R. L. Queiroz, and D. Mukherjee, "Mobile video communications using a Wyner–Ziv transcoder," in *Proc. Int. Soc. Opt. Eng. Visual Commun. Image Process.*, vol. 6822. San Jose, Jan. 2008.

[24] A. Aaron, D. Varodayan, and B. Girod, "Wyner–Ziv residual coding of video," in *Proc. Int. Picture Coding Symp.*, Beijing, China, Apr. 2006.

[25] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive distributed source coding using low-density parity-check codes," in *Proc. Conf. Rec. 39th Asilomar Conf. Signals Syst. Comput.*, Oct.–Nov. 2005, pp. 1203–1207.

[26] G. Coté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 849–866, Nov. 1998.

[27] D. Mukherjee, "Optimal parameter choice for Wyner–Ziv coding of Laplacian sources with decoder side-information," HP Labs, Palo Alto, CA, Tech. Rep. HPL-2007-34, 2007.

[28] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," *IEEE Trans. Consum. Electron.*, vol. 44, no. 1, pp. 88–98, Feb. 1998.

[29] B. Machiavello, F. Brandi, E. Peixoto, R. L. de Queiroz, and D. Mukherjee, "Side-information generation for temporally and spatially scalable Wyner-Ziv coders," *Eur. Assoc. Signal Proc. J. Image Video Process., Special Issue on Distributed Video Coding*, vol. 2009, article ID 171257, p. 11, 2009.

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

In 2008, he was a Research Associate with the Digital Signal Processing Research Group at the Universidade de Brasilia, where he developed this work. His research interests include video transcoding, distributed video coding, and scalable video coding.

Ricardo L. de Queiroz (S'86–M'94–SM'99) received the B.Eng. degree in electrical engineering from the Universidade de Brasilia, Brasilia, Brazil, the M.S. degree in electrical engineering from the Universidade Estadual de Campinas, Campinas, Brazil, and the Ph.D. degree in electrical engineering from the University of Texas, Arlington, in 1987, 1990, and 1994, respectively.

From 1990 to 1991, he was a Research Associate with the Digital Signal Processing Research Group at the Universidade de Brasilia. He joined Xerox Corporation, Rochester, NY, in 1994, where he was a member of the Research Staff until 2002. From 2000 to 2001, he was with the Rochester Institute of Technology, Rochester, NY, as an Adjunct Faculty Member. He is currently with the Department of Electrical Engineering at the Universidade de Brasilia. He is the author of over 100 articles in journals and conferences, and has contributed chapters to books as well. He holds over 40 issued patents. His research interests include image and video compression, multirate signal processing, and color imaging.

Dr. de Queiroz is a Member of the IEEE Signal Processing Society Image, Video, and Multidimensional Signal Processing Technical Committee. He is currently an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and in the past he has served as an Editor for IEEE SIGNAL PROCESSING LETTERS. He has been actively involved with the Rochester chapter of the IEEE Signal Processing Society, where he served as Chair and organized the Western New York Image Processing Workshop since its inception until 2001. He is the General Chair of the International Symposium on Circuits and Systems, 2011, and of the International Workshop on Multimedia Signal Processing, 2009. He was part of the Organizing Committee of the International Conference on Image Processing, 2002, and he is also a Member of both the Brazilian Telecommunications Society and the Brazilian Society of Television Engineers.

Debargha Mukherjee (M'99–SM'09) was born in Kolkata, India. He received the B.Tech. degree in electronic and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1993, and the M.S. and Ph.D. degrees in electrical communication engineering from the University of California, Santa Barbara, in 1995 and 1999, respectively.

Since 1999, he has been with Hewlett Packard Laboratories, Palo Alto, CA, where he is currently a Senior Research Scientist in the Multimedia Communications and Networking Laboratory. From 2002 to 2004, he was involved with MPEG-21 standardization activities, and several of his proposals were adopted in "MPEG-21 part 7 on digital item adaptation" in *Encyclopedia of Multimedia* (Berlin, Germany: Springer-Verlag). He has co-authored more than 60 papers in refereed conferences and journals. His research interests include standard and distributed source compression, multimedia signal processing, information theory, and data security.

Dr. Mukherjee received the IEEE Student Paper Award at the IEEE International Conference on Image Processing at Chicago, IL, in 1998.