MOTION AND TEXTURE RATE-ALLOCATION FOR PREDICTION-BASED SCALABLE MOTION-VECTOR CODING

Joeri Barbarien[1], Adrian Munteanu, Fabio Verdicchio, Yiannis Andreopoulos, Jan Cornelis and Peter Schelkens

Vrije Universiteit Brussel (VUB) – Interdisciplinary Institute for Broadband Technology (IBBT)
Department of Electronics and Information Processing (ETRO)
Pleinlaan 2, B-1050 Brussels, Belgium

ABSTRACT

Modern video coding applications require data transmission over variable-bandwidth wired and wireless network channels to a variety of terminals, possibly having different screen resolutions and available computing power. Scalable video coding technology is needed to optimally support these applications. Recently proposed wavelet-based video codecs employing spatial-domain motion-compensated temporal filtering (SDMCTF) provide quality, resolution and frame-rate scalability while delivering compression performance comparable to that of H.264, the state-of-the-art in single-layer video coding. These codecs require quality-scalable coding of the motion vectors to support a large range of bit-rates with optimal compression efficiency. In this paper, the practical use of prediction-based scalable motion-vector coding in the context of scalable SDMCTF-based video coding is investigated. Extensive experimental results demonstrate that, irrespective of the employed motion model, our prediction-based scalable motion-vector codec (MVC) systematically outperforms state-of-the-art wavelet-based solutions for both lossy and lossless compression. A new rate-distortion optimized rate-allocation strategy is proposed, capable of optimally distributing the available bit-budget between the different frames and between the texture and motion information, making the integration of the scalable MVC into a scalable video codec possible. This rate-allocation scheme systematically outperforms heuristic approaches previously employed in the literature. Experiments confirm that by using a scalable MVC, lower bit-rates can be attained without sacrificing motion-estimation efficiency and that the overall coding performance at low rates is significantly improved by a better distribution of the available rate between texture and motion information. The only downside of scalable motion vector coding is a slight performance loss incurred at high bit-rates.

---

[1] Corresponding author: Joeri Barbarien, Vrije Universiteit Brussel (VUB) – Interdisciplinary Institute for Broadband Technology (IBBT), Department of Electronics and Information Processing, Pleinlaan 2, B-1050 Brussels, Belgium. E-mail: jbarbari@etro.vub.ac.be Tel.: ++32 2 629 29 80, Fax: ++32 2 629 2883.

I. INTRODUCTION

The increasing demand for multimedia over networks and the heterogeneous profile of today's networks and playback devices (from low-resolution portable devices to HDTV platforms) impose the need for scalable video coding. Scalable video codecs based on existing standards using a hybrid coding architecture (e.g. MPEG-4 Fine Grain Scalability-FGS framework) were proposed in the past [1, 2]. These solutions however fail to provide competitive rate-distortion performance compared to single-layer coding schemes (i.e. MPEG-4, H.264), preventing their global acceptance by industry. Recently proposed open-loop wavelet-based video coding architectures using spatial domain motion compensated temporal filtering (SDMCTF) [3-10] form an alternative solution to the demand for scalable video coding. These codecs employ motion-compensated temporal filtering (MCTF), which is equivalent to performing a wavelet transform along the trajectory of the motion, followed by spatial decomposition of the resulting temporally-filtered frames using a classical two-dimensional discrete wavelet transform (DWT). The coefficients obtained using this spatio-temporal decomposition are encoded using an embedded coder, ensuring support for quality, resolution and temporal scalability. The most advanced among these codecs yield compression performance on par with that of the state-of-the-art non-scalable H.264-codec [11]. Their promising performance results, combined with their ability to support a broad range of scalability, motivate the extensive research into this type of video coding schemes.

MCTF-based video codecs typically employ a *lossless, non-scalable* motion-vector coding technique. This implies that the minimum bit-rate that can be attained by the video codec is bound by the rate needed to losslessly code the motion information. As a consequence, the motion estimation (ME) process must often be forced to generate less complex (but less accurate) motion fields in order to limit the motion-vector coding cost when low bit-rates need to be supported. However, sacrificing motion-field accuracy in this way significantly affects the overall performance of the scalable video codec at all rates. This problem can be solved by employing a quality-scalable motion vector codec (MVC) [12-16]. An additional benefit of using a quality-scalable MVC is the capability to optimally distribute the available bit-rate between motion and texture data. In comparison to a video codec using a non-scalable MVC, this yields a systematically better rate-distortion performance [13-15], especially at low bit-rates.

Quality-scalable motion-vector coding techniques based on performing an integer wavelet transform of the motion vector components followed by embedded coding of the resulting wavelet coefficients were proposed in [12, 13]. While these techniques are quality scalable, they have several disadvantages. First of all, their lossless compression performance is significantly lower than that of traditional prediction-based MVCs [12].

Secondly, it is not straightforward to adapt this approach to motion-vector fields generated by more complex motion estimation algorithms, such as block-based motion estimation using variable block-sizes. The wavelet transform would have to be performed on an irregular sampling grid to be able to efficiently compress this kind of data. Alternatively, a regular sampling grid constructed based on the smallest block size can be utilized, but this results into very poor compression performance, as we will illustrate in section IV.1.

Other quality-scalable motion-vector coding techniques were later published in [14-16]. These algorithms were all designed to encode motion information generated by multi-hypothesis block-based motion estimation using variable block-sizes [6, 17]. Although these MVCs were developed independently from each other, they exploit similar ideas to support quality-scalability of the motion information. Basically, they all generate a bit-stream consisting of a base layer and an enhancement layer to achieve quality scalability.

The MVC of [15] generates the base layer by replacing the motion vectors belonging to blocks with sizes smaller than 16x16 pixels with the motion vector(s) of their encompassing block of 16x16 pixels and coding the resulting motion field using a prediction-based MVC. The difference between the motion field encoded in the base layer and the original motion field is thereafter coded as an enhancement layer. The scalable MVC supports only coarse-grain quality scalability (the enhancement layer can either be transmitted or discarded). In the MVC of [16], the base layer is generated by rounding the original motion vectors, having 1/4-pel or 1/8-pel accuracy, to 1/2-pel accuracy, and coding the rounded motion vectors using a prediction-based MVC. The rounding errors are coded using the CABAC-coder [11], producing the enhancement layer.

In contrast to these approaches, the MVC we introduced in [14] constructs the base layer by quantizing the original motion vectors and coding the resulting quantized motion-vectors using a prediction-based MVC. The quantization errors are thereafter coded using an embedded bit-plane coding technique, producing a fine-grain quality-scalable enhancement layer. Experimental results show that this MVC outperforms wavelet-based scalable MVCs for both lossless and lossy compression, as we will illustrate in section IV.1.

It is important to note that, in order to make the integration of a quality-scalable MVC into a scalable video codec possible, the problem of appropriately distributing the rate between motion and texture data for any given target bit-rate needs to be addressed. In [16] this problem is not tackled, whereas heuristic rate allocation techniques are employed in [15] and in our previous publications on quality-scalable motion vector coding [14, 18, 19].

The focus of this paper is to demonstrate the practical use of the scalable MVC we introduced in [14] in the context of scalable MCTF-based video coding. Additionally, a new rate-allocation strategy, capable of

optimally distributing the available bit-budget between the different frames and between the texture and motion information is proposed.

The paper is structured as follows: our prediction-based scalable MVC is described in section II. Section III discusses the rate allocation approach. The conducted experiments and their results are presented in section IV. Finally, the conclusions of the paper are formulated in section V.

II. QUALITY AND RESOLUTION SCALABLE PREDICTION-BASED MOTION VECTOR CODING

II.1 Structure of the motion information

SDMCTF-based scalable video codecs have to employ advanced motion models to be able to reach the performance level of H.264. The majority of existing SDMCTF-based codecs that perform on par with H.264 [6, 7, 9] use multi-hypothesis block-based motion estimation with variable block-sizes and multiple reference-frames [17]. The considered prediction-based scalable motion-vector codec [14] is designed to code motion information produced by this type of motion-estimation algorithms. For each macro-block, the motion information that needs to be encoded consists of:

1. Splitting information: If and how the macro-block is split into sub-blocks.
2. For each separately predicted block:
    a. The hypothesis information, indicating the way the block is predicted (i.e. intra prediction, by a single block in one of the reference frames, or by the average of several blocks each lying in one of the reference frames).
    b. Depending on the hypothesis, zero (intra), one or more motion vectors.
    c. For each motion vector, the associated reference frame index.

In this paper, it is assumed that the motion-vector components are integer values lying in the range $\left[-pel \cdot searchrange, pel \cdot searchrange - 1\right]$ where pel indicates the employed motion-estimation accuracy ($pel = x$ corresponds to $1/x$ –pel accuracy) and $searchrange$ denotes the motion-estimation search range.

The next subsections will describe how this motion information is encoded by our prediction-based scalable MVC.

II.2 General setup of the prediction-based scalable motion-vector codec

The architecture of the considered prediction-based motion vector codec [14] is shown in Figure 1.

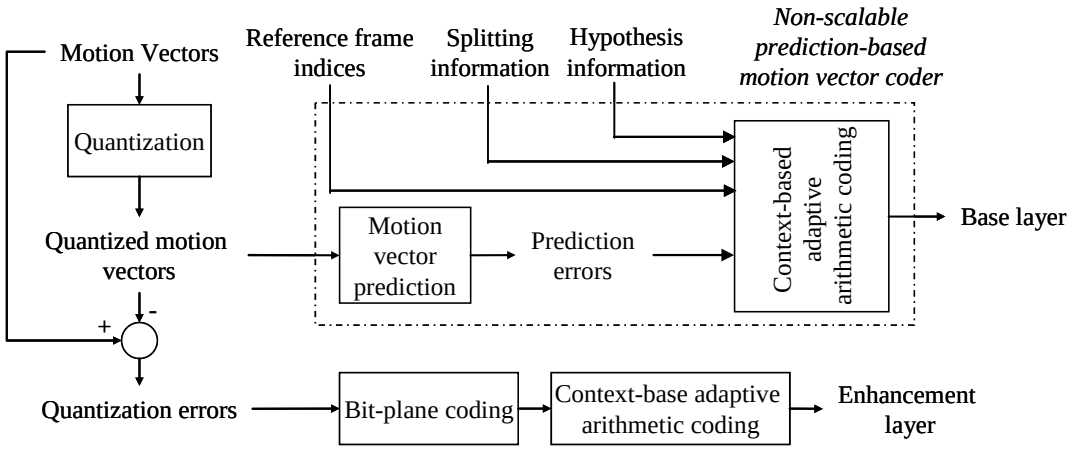


Figure 1: General architecture of the prediction-based scalable MVC [14].

The motion information to be coded consists of two parts, the actual motion vectors and the side information, including the reference frame indices, the hypothesis information and the macro-block splitting information. The motion vectors are first quantized by discarding the information on the lowest bit-plane(s). For a given quantization step size Q of the form $Q = 2^k$, the quantized motion vector $\left(QMV_x, QMV_y\right)$ and the quantization error $\left(QE_x, QE_y\right)$ corresponding to a motion vector $\left(MV_x, MV_y\right)$ are calculated as follows:

$$\begin{aligned} QMV_i &= \operatorname{sgn}\left(MV_i\right) \cdot \left\lfloor \frac{\left|MV_i\right|}{2^k} \right\rfloor \\ QE_i &= MV_i - \operatorname{sgn}(QMV_i) \cdot \left(2^k \cdot \left|QMV_i\right|\right) \\ i &= \{x, y\} \end{aligned} \qquad (1)$$

In equation (1), $\operatorname{sgn}(\ )$ is the sign operator and $\lfloor x \rfloor$ is the integer part of x.

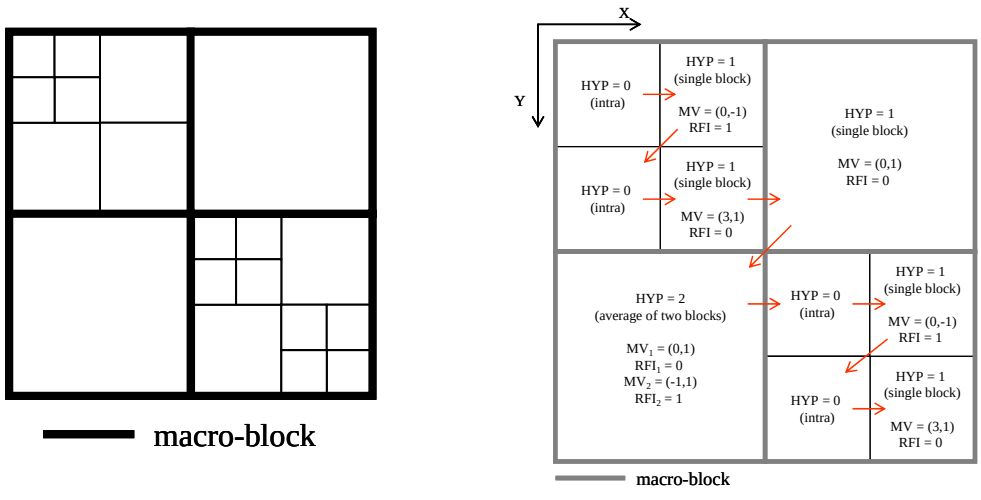
The quantized motion-vectors are thereafter coded together with the side information using a non-scalable, prediction-based motion-vector coding technique, producing the *base layer* of the bit-stream (see Figure 1). This base layer must always be decoded losslessly to avoid drift [14]. While any non-scalable prediction-based motion vector codec can be used to encode the quantized motion vectors and side information, we employ our own codec which is described in subsections II.3.1-II.3.4. The quantization errors are coded using an embedded bit-plane coding technique. The resulting compressed data forms the *quality scalable enhancement layer* of the final bit-stream. The embedded bit-plane codec used to produce the enhancement layer is described in subsection II.3.5.

By appropriately truncating the enhancement layer, the total motion vector bit-rate can be chosen to lie anywhere between the bit-rate needed to losslessly code the base layer and the bit-rate needed for a complete, lossless reconstruction of the motion information. At the decoder side, the base layer is then combined with the selected part of the enhancement layer to reconstruct the motion vectors employed in the inverse MCTF. The base layer bit-rate can be controlled in the encoder by changing the motion-vector

quantization step-size Q. Opting for a smaller base-layer size increases the range of supported bit-rates but in most cases decreases the overall compression performance of the MVC [14, 18].

Not only quality scalability but also resolution scalability and temporal scalability of the motion information can be supported by this coding architecture. Temporal scalability is automatically supported since the motion information associated to each H-frame is coded independently of the motion information associated to other H-frames. To support resolution scalability, the quantization step-size Q must be selected so that the number of bit-planes in the enhancement layer is larger than or equal to the number of lower resolutions that need to be supported by the video codec. When decoding to a lower resolution, the decoder scales down the original motion field to match this resolution, causing the lowest bit-planes of the motion vector components to become irrelevant. If the quantization step-size is chosen as described earlier, the potentially irrelevant bit-planes are part of the enhancement layer and can therefore be discarded before transmitting the motion information to the decoder. Note that, although our prediction-based MVC can support resolution scalability, we did not experiment with this feature since the focus of our paper is mainly on quality-scalability of the motion information. It is possible that other techniques to support resolution scalability of the motion vector field, such as the one employed in the Microsoft SVC codec [9, 10] may yield better coding performance.

II.3 Detailed description of the prediction-based scalable MVC



(a) **(b)**

Figure 2: (a) Hierarchical decomposition of macro-blocks into sub-blocks by block-based ME using variable block-sizes. (b) Scanning order illustrated by arrows and coordinate system used in the coding process.

When using block-based motion estimation with variable block-sizes, each macro-block can be recursively split into smaller sub-blocks that are predicted separately (Figure 2 (a)). While more elaborate splitting modes are used by H.264 [11], for a simplified description, we consider in the following that each block (macro-block or sub-block) can only be split into four equally sized sub-blocks, as depicted in Figure 2(a). This simplification is consistent with our instantiation of multi-hypothesis block-based motion estimation, as

described in [20]. This motion model is employed in the SDMCTF codec of [6], which is used in our experiments.

During the motion vector coding process, the same scanning order is employed in the encoding of the side information (splitting information, hypothesis (HYP) information, and reference-frame indices (RFIs)), and in the encoding of the quantized motion-vectors and quantization errors. In each of these coding steps, the macro-blocks are visited in raster order and, in case a macro-block is split, its sub-blocks are scanned in depth-first quadtree scanning order. An example of this scanning order is shown in Figure 2(b).

In the following sub-sections, the side-information and motion-vector coding algorithms are described in detail.

II.3.1 Splitting information

Context-based adaptive arithmetic coding is used to encode the splitting information for each visited macro-block. The following notations are introduced:

- N_{mb} and M_{mb}: The number of columns and rows of macro-blocks in the predicted frame.
- $MB(n,m)$: The macro-block located at the m-th row and n-th column in the predicted frame, with $0 \le n < N_{mb}$ and $0 \le m < M_{mb}$.
- MB_c: The currently visited macro-block $MB_c = MB(n_c, m_c)$, with $0 \le n_c < N_{mb}$ and $0 \le m_c < M_{mb}$.

The splitting state of a macro-block is defined as a binary value that indicates whether the macro-block is split into sub-blocks or not. The splitting state operator $split(\ )$ is defined as:

$$split(MB) = \begin{cases} 0 & if\ the\ macro-block\ MB\ is\ not\ split \\ 1 & otherwise \end{cases}. \qquad (2)$$

The coding process for the currently visited macro-block MB_c starts by coding its splitting state $split(MB_c)$. Adaptive arithmetic coding is performed using one of three different probability models. The probability model is selected based on $split(MB(n_c - 1, m_c))$ and $split(MB(n_c, m_c - 1))$, i.e. the splitting states of the macro-blocks above and to the left of $MB_c = MB(n_c, m_c)$. If $split(MB_c) = 1$, the exact splitting configuration of the macro-block must also be coded. This is done using depth-first quadtree coding [21] combined with binary arithmetic coding.

II.3.2 Hypothesis information

The hypothesis information describes the way each block (macro-block or sub-block, in case the macro-block is split) is predicted. For each separately predicted block, the hypothesis number HYP must be encoded. A block can be predicted as intra ($HYP = 0$), by a block of the same dimensions in one of the reference frames ($HYP = 1$) or by the average of $k, k > 1$ equally-sized blocks, each located in one of the

reference frames ($HYP = k$). The hypothesis numbers are coded using context-based adaptive arithmetic coding. The following symbols are defined:

- B_c : The currently visited block. If MB_c is not split, $MB_c = B_c$, otherwise B_c is one of the sub-blocks of MB_c. The size of B_c is $S_c \times S_c$ pixels. The top-left pixel of block B_c has coordinates (x_c, y_c).
- HYP_c : The hypothesis used to predict B_c. This corresponds to the number of different motion vectors generated in the ME process to predict B_c.
- $HYP(x, y)$: The hypothesis used for the prediction of the block containing the pixel at position (x, y). The coordinate system depicted in Figure 2(b) is used.

The following operator is defined:

$$\mathcal{H}(HYP) = \begin{cases} HYP & if\ HYP \le 2 \\ 3 & if\ HYP > 2 \end{cases}. \tag{3}$$

HYP_c is encoded by first calculating and encoding $\mathcal{H}(HYP_c)$. $\mathcal{H}(HYP_c)$ is coded by employing adaptive arithmetic coding using one of five different probability models. The selection of the appropriate probability model is based upon $\mathcal{H}(HYP(x_c - 1, y_c))$ and $\mathcal{H}(HYP(x_c, y_c - 1))$, where $HYP(x_c - 1, y_c)$ and $HYP(x_c, y_c - 1)$ respectively correspond to the hypothesis numbers of the blocks to the left and above the currently visited block B_c. Notice that this manner of referencing motion information in neighboring blocks is also used in the H.264 standard [11]. If $HYP_c > 2$, an additional offset value must also be coded. This offset value is defined as:

$$\mathcal{E}(HYP_c) = HYP_c - 3, \tag{4}$$

and is encoded using a single probability model in the adaptive arithmetic entropy coder.

At the decoder end, HYP_c is reconstructed from $\mathcal{H}(HYP_c)$ and $\mathcal{E}(HYP_c)$ as:

$$HYP_c = \begin{cases} \mathcal{H}(HYP_c) & if\ \mathcal{H}(HYP_c) \le 2 \\ \mathcal{H}(HYP_c) + \mathcal{E}(HYP_c) & if\ \mathcal{H}(HYP_c) = 3 \end{cases}. \tag{5}$$

II.3.3 Reference frame indices

Consider a block B (which can be a macro-block or a sub-block) that is predicted by one or more equally-sized blocks each lying in one of the possible reference frames. For each of the blocks used in the prediction of B, a motion vector and a reference-frame index (RFI) must be encoded. The reference-frame index points to the reference frame the block is located in, while the motion vector indicates the position of this block in the reference frame, relative to the position of the predicted block in the predicted frame. Let us denote by f_p the frame number of the currently predicted frame, assuming that the frames in the video sequence are numbered linearly starting from 0. The reference frame index RFI of a frame with frame number f_r is then calculated as:

$$RFI = \begin{cases} 2\cdot\left(f_r - f_p - 1\right)+1 & if\ \ f_r > f_p \\ 2\cdot\left(f_p - f_r - 1\right) & if\ \ f_r < f_p \end{cases}. \qquad (6)$$

From (6), it is clear that frames lying temporally closer to f_p are assigned smaller reference frame indices. The RFIs are coded using median-based prediction combined with context-based adaptive arithmetic coding. In the coding process, all RFIs belonging to the same block are processed first before proceeding to the next block. The notations introduced in previous subsections are reused; some new symbols are also defined:

- RFI_i^c : The reference-frame index of the i-th block involved in the prediction of the currently visited block B_c, with $1 \le i \le HYP_c$, for $HYP_c > 0$.
- $RFI_i\left(x,y\right)$: The reference-frame index of the i-th block used to predict the smallest block containing the pixel $\left(x,y\right)$, $1 \le i \le HYP\left(x,y\right)$, for $HYP\left(x,y\right) > 0$.

Each reference-frame index RFI_i^c, $1 \le i \le HYP_c$ belonging to B_c is first predicted based upon the RFIs of neighboring blocks. Its prediction RFI_i^p is calculated using median-based prediction as follows:

$$RFI_i^p = median\left(RFI_i\left(x_c - 1, y_c\right), RFI_i\left(x_c, y_c - 1\right), RFI_i\left(x_c + S_c, y_c - 1\right)\right) \qquad (7)$$

If one of the three RFIs involved in the prediction is unknown or does not exist, it is replaced by 0 in the calculation. In case two of these RFIs are unknown, RFI_i^p is set equal to the remaining neighboring RFI. When none of the neighboring RFIs is available, the predicted RFI is set to zero. After the prediction step, the obtained prediction errors are coded using adaptive arithmetic coding. The interval of possible prediction error values is divided into a number of sub-intervals S_i:

$$S_i = \begin{cases} \{0\}\ if\ i = 0 \\ \left[-\left(2^i - 1\right), -\left(2^{i-1}\right)\right] \cup \left[2^{i-1}, 2^i - 1\right]\ if\ i > 0 \end{cases} \qquad (8)$$

For each prediction error, an index representing the sub-interval S_i it belongs to is coded first using a single probability-model in the adaptive arithmetic-coder. Thereafter, the offset within the interval is coded using a separate probability model for every sub-interval.

II.3.4 Quantized motion vectors

The quantized motion-vectors are encoded by performing motion-vector prediction followed by lossless coding of the resulting motion-vector prediction errors. The prediction is performed by taking the median of a set of motion vectors U_p, containing the motion vectors associated with the blocks located to the left, top and top-left of the currently-visited block B_c and any previously-predicted motion vectors belonging to B_c. Context-based adaptive arithmetic coding is used to encode the prediction errors. The horizontal and vertical components of the prediction errors are coded separately. The interval of possible component values is split into a number of sub-intervals S_i, as given by (8). For each component value, a symbol representing the sub-

interval S_i it belongs to is coded first. Thereafter, the offset of the prediction error component within the sub-interval is coded. For a detailed description of the coding process for the quantized motion vectors, the reader is referred to [14, 18, 19].

II.3.5 Embedded coding of the motion-vector quantization errors

The horizontal and vertical components of the quantization errors are coded in a bit-plane by bit-plane fashion. Each bit-plane is coded in two passes, a significance pass followed by a refinement pass. A threshold $T = 2^i$ is associated with each bit-plane i. A quantization error component c_e is said to be significant for a threshold T if $|c_e| \geq T$. All bit-planes are coded sequentially, starting with the most significant bit-plane. In the significance pass, the significance of all previously non-significant components is encoded. Multiple quantization-error vectors associated to the same block are visited sequentially before proceeding to the next block. When a component becomes significant for the first time and its corresponding quantized motion vector component is 0, its sign is also coded. In the refinement pass, already significant components are refined by coding the binary value corresponding to the current bit-plane.

The significance, refinement and sign information is compressed using context-based adaptive arithmetic coding. To facilitate the discussion of this coding process, the following notations are introduced:

- $\overrightarrow{\mathbf{v}_i^c}$: the i-th motion vector involved in the prediction of B_c, with $1 \leq i \leq HYP_c$, for $HYP_c > 0$.
- $\overrightarrow{\mathbf{v}_i}(x, y)$: the i-th motion vector generated in the ME process to predict the smallest block containing the pixel (x, y), with $1 \leq i \leq HYP(x, y)$.
- $QE\left(\vec{\mathbf{v}}\right)$: the quantization error produced by quantizing the motion vector $\vec{\mathbf{v}}$.

The significance information for the components of $QE\left(\overrightarrow{\mathbf{v}_1^c}\right) = QE\left(\overrightarrow{\mathbf{v}_1}(x_c, y_c)\right)$ is coded using three probability-models per component. The choice of the model is based upon the significance of the corresponding components of $QE\left(\overrightarrow{\mathbf{v}_1}(x_c - 1, y_c)\right)$ and $QE\left(\overrightarrow{\mathbf{v}_1}(x_c, y_c - 1)\right)$, i.e. the quantization errors associated to the blocks to the left and above the currently visited block. A first model is used if both components are significant or if one of the components is significant and the state of the other is unavailable (if $x_c = 0$ or $y_c = 0$ or if one of the blocks referred to is coded in intra-mode). The second model is used if both components are not significant or if one of the components is not significant and the state of the other is unavailable. The third model is used in all other cases. Finally, two models for each component are used to code the significance of the components of the remaining $QE\left(\overrightarrow{\mathbf{v}_i^c}\right)$ for all $i, 1 < i \leq HYP_c$. A first model is used if the corresponding component of $QE\left(\overrightarrow{\mathbf{v}_1^c}\right)$ is significant, the second if it is not.

III. BASE-LAYER RATE-CONTROL AND GLOBAL RATE-ALLOCATION

Integration of a scalable motion-vector coding technique into an MCTF-based video codec requires a rate allocation strategy, capable of distributing the available bit-rate between the different frames in the video sequence and between motion and texture information. In our previous publications related to quality-scalable motion vector coding [14, 18, 19], a heuristic rate allocation technique was employed. This technique is described in detail in subsection III.2. In subsection III.3, a new R-D optimized rate-allocation scheme based on Lagrangian rate-distortion optimization is proposed.

The bit-rate needed to compress the base-layer motion information must always be lower than the lowest target bit-rate supported by the scalable video codec. A control mechanism for the bit-rate of the compressed base-layer motion information is therefore needed. The technique to accomplish this is described next.

III.1 Base-layer rate-control

The size of the base-layer part of the compressed motion information can be influenced by adjusting the quantization step Q of the motion vectors (see subsection II.2). Increasing Q lowers the base-layer size and vice-versa. The employed rate–control mechanism enforces a constant bit-rate, meaning that for each frame, the bit-rate of the base-layer motion information is kept below the given target bit-rate $b_{\max}$. The algorithm to control the base-layer bit-rate for a given frame can be described as follows:

1. Set $Q = 1$ (no enhancement layer)
2. Compress motion vectors of the current frame using quantization step size Q.
3. if compressed base-layer bit-rate $> b_{\max}$,
   {
   $Q = 2 \cdot Q$
   Goto 2
   }
   else
   compressed base-layer bit-rate is met.

Experimental results (see section IV.3) show that, in most cases, the overall compression performance of the motion vector codec decreases when Q is increased. The proposed algorithm takes this into account by ensuring that the minimal value of Q that satisfies the bit-rate constraint is selected. It may seem prohibitive to repeatedly compress the motion information from a complexity viewpoint. However, the number of iterations is typically small and the computational complexity of the proposed scalable MVC is several orders of magnitude smaller than that of the motion-estimation step, making its impact on the global complexity of the video codec negligible.

III.2 Heuristic technique for global rate-allocation

Previous publications discussing quality-scalable motion-vector coding employ heuristic techniques to allocate the available bit-rate between texture and motion information [14, 15]. In this section, our heuristic rate-allocation approach used in [14, 18, 19] is presented in detail. To simplify the discussion, it is assumed that only quality scalability must be supported by the video codec. To support frame-rate scalability, the encoder only sends the motion information associated with the H-frames in the temporal subbands that are effectively transmitted. Resolution scalability is supported by discarding the lowest j bit-planes of the motion-vector enhancement layer when the decoder requests a lower resolution $W/2^j \times H/2^j$, with W and H the original width and height of the frames.

The rate allocation mechanism proposed in this subsection allocates the available bit-rate heuristically. For a given target bit-rate, the motion vectors are first decoded at 6 different rates, evenly spread out between the base-layer rate and the rate needed for the lossless reconstruction of the vectors (see Figure 3). The remaining bit-rate is allocated to the texture information. The combination of motion and texture rates delivering the best quality (PSNR) is retained.

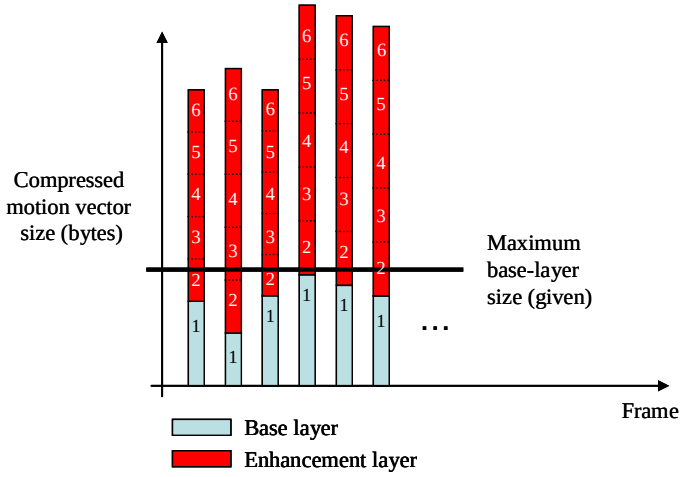


Figure 3: The motion information is decoded at 6 different rates.

The bit-rate available for coding the texture information is also allocated in a heuristic way. Notice that a similar way of allocating the texture bit-rate was used in [6, 7]. In SDMCTF-based scalable video codecs the texture information is typically coded using embedded bit-plane coding techniques such as EBCOT [22], EZBC [7] or QT-L [21, 23]. These algorithms code each bit-plane in a number of coding passes. The rate-allocation algorithm assumes that the temporal transform was made approximately unitary during encoding [24]. To accomplish this, the wavelet coefficients in each temporally- and spatially-filtered frame are multiplied by a scaling factor sf depending on the temporal subband the frame belongs to. Each temporal subband is the result of a sequence of high-pass and low-pass filtering operations along the trajectory of the motion. The scaling factors are calculated as $sf = \left(\sqrt{2}\right)^{lf}$, with lf equal to the number of low-pass filtering

operations involved in the generation of the corresponding temporal subbands. For example, when four temporal levels of decomposition are applied, the coefficients in the low-pass frame of the fourth level are multiplied by $sf = \left(\sqrt{2}\right)^4 = 4$.

To facilitate the description of the rate-allocation scheme, the following symbols are defined:

- $f_{b,p}^{c}(k)$: The sub-stream (fragment of the final bit-stream) generated by the p-th coding pass of the b-th bit-plane for the c-th color channel of frame k. Its size in bytes is denoted as $s\left(f_{b,p}^{c}(k)\right)$. It must be noted that $f_{b,p}^{c}(k)$ is empty if no information is generated in pass p or if the most significant bit-plane that must be encoded for component c of frame k is lower than b.
- s_t: The total number of bytes available for coding the texture information of the entire video sequence.
- N_p: The number of coding passes used to code every bit-plane.
- N_f: The number of frames in the video sequence.
- N_c: The number of color components per frame. For YUV4:2:0 color sequences, $N_c = 3$, with $c = 0$ corresponding to the Y component, $c = 1$ to the U component and $c = 2$ to the V component.
- $bp_{\max}$: The most significant bit-plane to be coded in the entire temporally- and spatially- transformed sequence.

The rate-allocation procedure is given in the following:

```
k = 0 ;  b = bp_max  ; p = 0 ;  c = 0
totalsize = 0
while ( totalsize < s_t )
{
        Add  f_{b,p}^{c}(k)  to the extracted bit-stream
        totalsize = totalsize + s( f_{b,p}^{c}(k) )
        c = c + 1
        if( c > N_c − 1 )
        {
                c = 0
                k = k + 1
                if( k > N_f − 1 )
                {
                        k = 0
                        p = p + 1
                        if( p > N_p − 1 )
                        {
                                p = 0
                                b = b − 1
                        }
                }
        }
}
```

To summarize, the bit-rate available to code the texture information is allocated (1) by sorting the $f_{b,p}^{c}(k)$ sub-streams according to their importance as given by their associated bit-plane, coding pass and color component, and (2) by concatenating them until the target bit-rate is met.

III.3 Rate allocation using Lagrangian rate-distortion optimization

For a simplified presentation, it is again assumed that only quality scalability must be supported by the video codec. The rate-allocation mechanism proposed in this subsection allocates the available bit-rate using Lagrangian rate-distortion optimization. The bit-stream generated by coding a video sequence consists of several contributions: the compressed texture information for the color components of the L-frames and H-frames and the compressed motion information associated with each H-frame. Each of these contributions is coded in an embedded way. The rate-allocation scheme must then decide where to cut the bit-stream fragments associated to each of the contributions in order to obtain the best quality for a given bit-budget. To facilitate the description, it is assumed that the number of temporal levels for each group of pictures (GOP) is chosen so that each temporally-transformed GOP contains only one L-frame. The following notations are introduced:

- N_g : the number of GOPs in the sequence.
- N_h^i : the number of H-frames in GOP i, $0 \le i < N_g$.
- N_c : the number of color components in each frame.
- L_c^i : The c-th color component of the L-frame in the i-th GOP, $0 \le i < N_g$, $0 \le c < N_c$.
- $H_c^{i,j}$: The c-th color component of the j-th H-frame in the i-th GOP, with $0 \le i < N_g$, $0 \le j < N_h^i$ and $0 \le c < N_c$.
- $M_B^{i,j}$: The base-layer motion information associated to the j-th H-frame in the i-th GOP, with $0 \le i < N_g$ and $0 \le j < N_h^i$.
- $M_E^{i,j}$: The enhancement layer motion information associated to the j-th H-frame in the i-th GOP, with $0 \le i < N_g$ and $0 \le j < N_h^i$.
- $R_B^{i,j}$: The rate in bytes needed to losslessly code the base-layer motion information $M_B^{i,j}$.
- R_T : The target rate in bytes.

The embedded bit-stream fractions corresponding to each contribution $X \in \left\{L_c^i, H_c^{i,j}, M_E^{i,j}\right\}$, $0 \le i < N_g, 0 \le j < N_h^i, 0 \le c < N_c$ can be cut off in a number of points defined at encoding time. For each of these truncation points, the rate is measured and the corresponding distortion of the contribution is measured or estimated. These points are ordered according to increasing rate and stored for use in the rate allocation

procedure. The following operators are defined for every $X \in \left\{L_c^i, H_c^{i,j}, M_E^{i,j}\right\}$, $0 \le i < N_g, 0 \le j < N_h^i, 0 \le c < N_c$:

- $N_p\left(X\right)$: The number of truncation points defined for the bit-stream representing X.
- $D^n\left(X\right)$: The distortion of X associated to truncation point $n, 0 \le n < N_p\left(X\right)$.
- $R^n\left(X\right)$: The rate in bytes associated to truncation point $n, 0 \le n < N_p\left(X\right)$

The distortions are considered to be additive. Hence, the global distortion function for the entire video sequence is given by:

$$D_T = \sum_{i=0}^{N_g-1}\left(\sum_{c=0}^{N_c-1}\left(\mu_c \cdot D^{n_{i,c}}\left(L_c^i\right) + \sum_{j=0}^{N_h^i-1}\eta_c^j \cdot D^{m_{i,j,c}}\left(H_c^{i,j}\right)\right) + \sum_{j=0}^{N_h^i-1}\xi^j \cdot D^{l_{i,j}}\left(M_E^{i,j}\right)\right). \qquad (9)$$

The weight factors μ_c, η_c^j, ξ^j are introduced to fine-tune the distortion function in order to better approximate the behavior of the actual distortion after decoding. The appropriate selection of these weight factors is discussed later in this section.

In the rate-allocation process, the truncation points $n_{i,c}, m_{i,j,c}, l_{i,j}$ for the bit-streams of the different contributions are determined as to minimize the total distortion D_T given the rate constraint:

$$\sum_{i=0}^{N_g-1}\left(\sum_{c=0}^{N_c-1}\left(R^{n_{i,c}}\left(L_c^i\right) + \sum_{j=0}^{N_h^i-1}R^{m_{i,j,c}}\left(H_c^{i,j}\right)\right) + \sum_{j=0}^{N_h^i-1}\left(R^{l_{i,j}}\left(M_E^{i,j}\right) + R_B^{i,j}\right)\right) \le R_T \qquad (10)$$

which is equivalent to:

$$R_T^{'} \triangleq \sum_{i=0}^{N_g-1}\left(\sum_{c=0}^{N_c-1}\left(R^{n_{i,c}}\left(L_c^i\right) + \sum_{j=0}^{N_h^i-1}R^{m_{i,j,c}}\left(H_c^{i,j}\right)\right) + \sum_{j=0}^{N_h^i-1}\left(R^{l_{i,j}}\left(M_E^{i,j}\right)\right)\right) \le R_{\max} = R_T - \sum_{i=0}^{N_g-1}\sum_{j=0}^{N_h^i-1}R_B^{i,j} \quad .(11)$$

This problem is solved by using Lagrangian rate-distortion optimization, which implies minimizing the functional $J = D_T + \lambda\left(R_T^{'} - R_{\max}\right)$ for some $\lambda > 0$. Define:

$$\lambda_{i,c}\left(z\right) \triangleq \min_{z'<z}\frac{D^{z'}\left(L_c^i\right) - D^{z}\left(L_c^i\right)}{R^{z}\left(L_c^i\right) - R^{z'}\left(L_c^i\right)}, \lambda'_{i,j,c}\left(z\right) \triangleq \min_{z'<z}\frac{D^{z'}\left(H_c^{i,j}\right) - D^{z}\left(H_c^{i,j}\right)}{R^{z}\left(H_c^{i,j}\right) - R^{z'}\left(H_c^{i,j}\right)}, \lambda''_{i,j}\left(z\right) \triangleq \min_{z'<z}\frac{D^{z'}\left(M_E^{i,j}\right) - D^{z}\left(M_E^{i,j}\right)}{R^{z}\left(M_E^{i,j}\right) - R^{z'}\left(M_E^{i,j}\right)}$$

for $z > 0$, and $\lambda_{i,c}\left(0\right) \triangleq \infty, \lambda'_{i,j,c}\left(0\right) \triangleq \infty, \lambda''_{i,j}\left(0\right) \triangleq \infty$. From the sets of rate-distortion points generated by the embedded coding of the texture and motion information, only those points that lie on the convex-hull of the distortion-rate characteristic for each frame are eligible, and thus retained. Denote by $\mathcal{H}_{i,c}, \mathcal{H}'_{i,j,c}, \mathcal{H}''_{i,j}$ the sets of feasible truncation points for the contributions $L_c^i, H_c^{i,j}, M_E^{i,j}$ respectively.

The necessary and sufficient conditions for a truncation point z to belong to one of these sets are respectively [22]:

$$z \in \mathcal{H}_{i,c} \Leftrightarrow \lambda_{i,c}(z) > 0 \ \ and \ \ \lambda_{i,c}(z) > \max_{t>z} \frac{D^z\left(L_c^i\right) - D^t\left(L_c^i\right)}{R^t\left(L_c^i\right) - R^z\left(L_c^i\right)},$$

$$z \in \mathcal{H}'_{i,j,c} \Leftrightarrow \lambda'_{i,j,c}(z) > 0 \ \ and \ \ \lambda'_{i,j,c}(z) > \max_{t>z} \frac{D^z\left(H_c^{i,j}\right) - D^t\left(H_c^{i,j}\right)}{R^t\left(H_c^{i,j}\right) - R^z\left(H_c^{i,j}\right)}, \qquad (12)$$

$$z \in \mathcal{H}''_{i,j} \Leftrightarrow \lambda''_{i,j}(z) > 0 \ \ and \ \ \lambda''_{i,j}(z) > \max_{t>z} \frac{D^z\left(M_E^{i,j}\right) - D^t\left(M_E^{i,j}\right)}{R^t\left(M_E^{i,j}\right) - R^z\left(M_E^{i,j}\right)}.$$

Minimizing the functional J for a given λ corresponds to finding the optimum truncation points $n_{i,c}^{opt}, m_{i,j,c}^{opt}, l_{i,j}^{opt}$ as:

$$n_{i,c}^{opt} = \max\left\{h \in \mathcal{H}_{i,c} \mid \lambda_{i,c}(h) > \frac{\lambda}{\mu_c}\right\},$$

$$m_{i,j,c}^{opt} = \max\left\{h \in \mathcal{H}'_{i,j,c} \mid \lambda'_{i,j,c}(h) > \frac{\lambda}{\eta_c^j}\right\}, \qquad (13)$$

$$l_{i,j}^{opt} = \max\left\{h \in \mathcal{H}''_{i,j} \mid \lambda''_{i,j}(h) > \frac{\lambda}{\xi^j}\right\}.$$

The global optimization problem then reduces to finding the optimum value λ^{opt} as:

$$\lambda^{opt} = \min\left\{\lambda \mid \sum_{i=0}^{N_g-1}\left(\sum_{c=0}^{N_c-1}\left(R^{n_{i,c}^{opt}}\left(L_c^i\right) + \sum_{j=0}^{N_h^i-1} R^{m_{i,j,c}^{opt}}\left(H_c^{i,j}\right)\right) + \sum_{j=0}^{N_h^i-1}\left(R^{l_{i,j}^{opt}}\left(M_E^{i,j}\right)\right)\right) \le R_{\max}\right\} \qquad (14)$$

Since R'_T is monotonically decreasing function of λ, the search for λ^{opt} can be performed by using the well-known bi-section method [22].

An important aspect of the proposed algorithm is the estimation of the employed distortions $D^{n_{i,c}}\left(L_c^i\right)$, $D^{m_{i,j,c}}\left(H_c^{i,j}\right)$ and $D^{l_{i,j}}\left(M_E^{i,j}\right)$. The SDMCTF-based video codec used in our experiments employs the QT-L embedded coding algorithm of [21] to encode the texture information of the L-frames and H-frames. This algorithm codes the wavelet coefficients in a bit-plane by bit-plane fashion. Each bit-plane is coded using three coding passes, the non-significance pass, the significance pass and the refinement pass. These passes respectively resemble the significance propagation, normalization and magnitude-refinement coding passes of EBCOT [22]. A threshold $T = 2^i$ is associated with each bit-plane i. A wavelet coefficient c_w is said to be significant for a threshold T if $|c_w| \ge T$. In the non-significance and significance passes, the significance of all previously non-significant components is encoded [21]. When a component becomes significant for the first time its sign is also coded. The refinement pass encodes the current bit-plane of the wavelet coefficients that are already significant.

The distortions $D^{n_{i,c}}\left(L_c^i\right)$ and $D^{m_{i,j,c}}\left(H_c^{i,j}\right)$ are obtained by estimating the average distortion-reduction after each decoding pass. Specifically, to obtain the average distortion-reduction for the non-significance or significance decoding passes of bit-plane k, the number of coefficients that become significant is multiplied

with the estimated average distortion-reduction $\Delta D_S(k)$ obtained when a single coefficient becomes significant. $\Delta D_S(k)$ can be calculated as [21, 22]:

$$\Delta D_S(k) = \frac{27}{12} 2^{2k} \tag{15}$$

The distortion reduction for the refinement pass of bit-plane k is calculated by multiplying the number of refined coefficients with the estimated average distortion-reduction $\Delta D_R(k)$ obtained by refining a single coefficient. $\Delta D_R(k)$ is given by [21, 22]:

$$\Delta D_R(k) = 2^{2(k-1)} \tag{16}$$

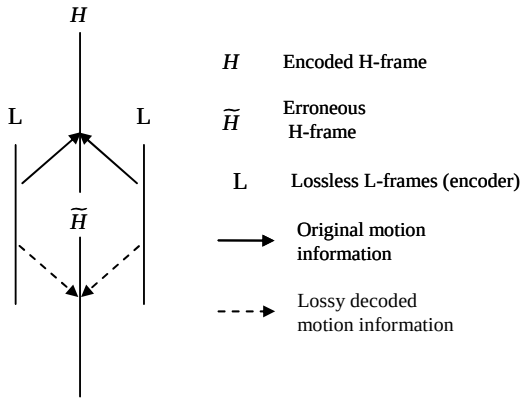


Figure 4: Calculation of the distortion incurred by the truncation of the motion information.

Finally, the distortions $D^{l_{i,j}}\left(M_E^{i,j}\right)$ associated to the motion information are not estimated but measured. At encoding time, each H-frame H is produced by motion-compensated temporal filtering using the motion information generated in the motion-estimation process and one or more lossless L-frames. For each truncation point l of the compressed motion-information associated to H, an erroneous H-frame $\widetilde{H}^l$ is generated using the lossyly-decoded motion information. The distortion D^l associated to the given truncation point l is then measured as the MSE between H and $\widetilde{H}^l$. Figure 4 illustrates this process for the simple case of unconstrained motion-compensated temporal filtering (UMCTF) [6].

The weight factors μ_c, η_c^j, ξ^j in the distortion function D_T account for the difference in behavior between the unweighted distortion function and the real distortion observed after decoding. This difference occurs because of two reasons. First of all, the procedure for calculating the distortions $D^{l_{i,j}}\left(M_E^{i,j}\right)$ does not take into account that errors introduced in a single frame by the truncation of motion information will also propagate to other frames in the GOP through the MCTF process. This means the distortion caused by motion-information truncation is underestimated. Moreover, the higher the temporal decomposition-level of the H-frame in which the error is introduced, the more frames in the GOP will be affected by the propagation of the error. As a consequence, the higher the temporal level of the H-frame, the more the distortion contribution associated to the truncation of its motion information is underestimated. The weight factors ξ^j

can be used to compensate for these error propagation effects. Larger weights should be used for the distortions associated to H-frames of higher temporal levels.

Secondly, the distortion contributions for the texture information $D^{n_{i,c}}\left(L_c^i\right)$ and $D^{m_{i,j,c}}\left(H_c^{i,j}\right)$ are estimated in the wavelet domain. In a spatio-temporally transformed GOP, distortions introduced in frames of higher temporal decomposition-levels have a larger impact on the total distortion of the decoded GOP. Two different solutions can be used to take this into account. The first solution is to appropriately choose μ_c, η_c^j so that the distortion contributions for frames of higher temporal levels get larger weights assigned to them. The second solution is to make the temporal transform approximately unitary prior to coding and distortion estimation. This can be done by multiplying the wavelet coefficients with a scaling factor depending on the frame's temporal decomposition level, as proposed in [24]. The procedure was detailed in section III.2. When this solution is chosen, the weight factors μ_c, η_c^j can be set to 1, assuming all color components are attributed equal importance. In our experiments, the latter approach is adopted. This means that only the weight factors ξ^j must be experimentally determined. With this respect, the SDMCTF-codec of [6], equipped with the prediction-based scalable MVC and the R-D-optimized rate allocation technique was used to code a limited set of test sequences using different values of ξ^j. The following values of ξ^j yielded the best compression performance: for the highest temporal level, ξ^j is set to 2.5, for level 3 to 2, for level 2 to 1.66, and for level 1 to 1.43.

IV. EXPERIMENTAL RESULTS

In a first set of experiments, our prediction-based scalable MVC is compared to a wavelet-based quality-scalable motion-vector coding technique. The experimental setup and results are presented in subsection IV.1. In the second set of experiments, the impact on the compression performance of using a prediction-based scalable MVC instead of a classical, non-scalable MVC is demonstrated for two different SDMCTF-based video codecs. The details of these experiments are given in subsection IV.2. In the last set of experiments, the base-layer rate-control mechanism is validated, the performance of the proposed R-D-optimized rate-allocation scheme is assessed, and the rate distribution obtained using this rate-allocation scheme is examined in detail. The experimental setup and the results are discussed in subsection IV.3.

IV.1 Comparison of prediction-based versus wavelet-based scalable MVC

Our prediction-based scalable MVC is compared against a wavelet-based quality-scalable MVC similar to the one proposed in [13]. This MVC separately codes the components of the motion vectors by performing a 5/3 integer wavelet transform on the motion-vector components followed by quality-scalable coding of the

resulting transform coefficients. The difference with respect to [13] is that the encoding is performed using the QT-L codec of [21, 23], whereas the JPEG2000 codec [22] is used in [13]. Since the performance of QT-L is on par with that of JPEG2000 [21], both scalable MVC systems are expected to yield similar coding performance.

Name	Resolution	No. of frames	Framerate (Hz)
Football	CIF	260	30
Canoa	CIF	220	30
Bus	CIF	150	30
Container	CIF	300	30

Table 1: Test sequences used in the experiments.

In the first experiment, the wavelet-based and prediction-based scalable MVCs are applied to motion information generated by multi-hypothesis block-based motion estimation using 2 reference frames, 2 block sizes and quarter-pel accuracy and their lossless compression performance is compared. The motion-estimation algorithm is employed in a 4-level 5/3 MCTF-decomposition without update step (UMCTF). In its original form, the wavelet-based scalable MVC cannot handle the irregularly sampled motion-field generated by block-based motion estimation with variable block-sizes. The most straightforward solution to this problem is to transform the original motion field into a new motion field with a regular sampling-grid corresponding to the smallest block-size (see Figure 5) and to apply wavelet-based motion-vector coding to this new field. This technique is used in our experiment. To encode the side information (reference indices, hypothesis information), the wavelet-based MVC employs the same coding mechanisms as the prediction-based scalable MVC.

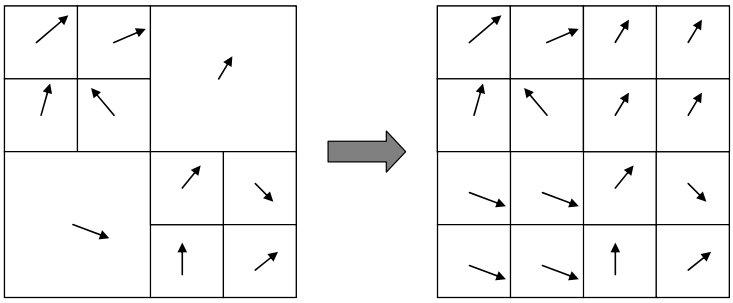


Figure 5: Resampling of the motion field prior to wavelet-based motion vector coding.

In the prediction-based scalable MVC, the base-layer size was kept below 427 bytes/frame (corresponding to 96 kbps) by the base-layer rate-control algorithm proposed in subsection III.1. The four sequences described in Table 1 were used in the experiment. In Table 2, we report the average number of bytes needed to losslessly code the motion information using the prediction-based and wavelet-based MVCs. For reference purposes, the average uncompressed size of the motion information is also reported. The results show that the prediction-based scalable MVC significantly outperforms the wavelet-based scalable MVC. Moreover, the wavelet-based MVC does not achieve any compression for 3 of the 4 sequences. These results clearly

indicate that adapting the wavelet-based MVC to motion information generated by multi-hypothesis block-based motion estimation using multiple reference-frames and variable block-sizes is not a straightforward problem indeed.

In the following set of experiments, the difficulties of adapting the wavelet-based MVC to more complex motion information are avoided by employing a simpler motion model. Specifically, we evaluate the lossy and lossless compression performance of the MVCs for motion information generated by a multi-hypothesis block-based motion estimation algorithm, using two reference frames, no intra mode, no macro-block splitting and quarter-pel accuracy. The motion estimation is employed in a 4-level 5/3 MCTF-decomposition without update step (UMCTF).

Sequence	Prediction-based scalable MVC	Wavelet-based scalable MVC	Uncompressed
Football	817	3455	938
Canoa	1022	3656	1262
Bus	668	2562	1058
Container	43	180	200

Table 2: Lossless compression performance of the scalable MVCs when coding motion information generated by multi-hypothesis block-based motion estimation using 2 reference frames, 2 block sizes and quarter-pel accuracy (the average number of bytes per frame is reported).

The experiments were again conducted using the four sequences of Table 1. The lossless compression performance is compared first. In Table 3, the average number of bytes per frame needed to code the motion information is given. In the proposed MVC, the base-layer size was kept below 200 bytes/frame.

Sequence	Prediction-based scalable MVC	Wavelet-based scalable MVC
Football	682	722
Canoa	670	709
Bus	578	616
Container	161	211

Table 3: Lossless compression performance of the prediction-based and wavelet-based scalable MVCs when coding motion information generated by multi-hypothesis block-based motion estimation using two reference frames, no intra mode, no macro-block splitting and quarter-pel accuracy (the average number of bytes needed to code the motion information of an H-frame is reported).

Next, the lossy coding performance of both motion-vector coding schemes is compared. For each predicted frame, the MSE of the motion vectors is calculated for different motion vector bit-rates. Based on these MSE figures, a global PSNR value is calculated for the entire sequence. The MSE of the motion vectors associated to predicted frame k is calculated as follows:

$$MSE_k = \frac{1}{2N_k}\sum_{i=0}^{N_k-1}\left(\left(x_i - \tilde{x}_i\right)^2 + \left(y_i - \tilde{y}_i\right)^2\right), \qquad (17)$$

where N_k represents the number of motion vectors associated to predicted frame k, x_i, y_i are the original motion vector components, with $0 \le i < N_k$, and $\tilde{x}_i$, $\tilde{y}_i$ are the reconstructed motion-vector components. The global motion-vector PSNR for the entire sequence is then calculated as:

$$PSNR_{MV} = 10 \cdot \log_{10} \left(\frac{(MAX - MIN)^2}{\frac{1}{N} \sum_{k=0}^{N-1} MSE_k} \right) \quad (18)$$

In equation (18), MAX and MIN respectively denote the maximum and the minimum of the range of possible motion-vector component values, and N denotes the number of predicted frames in the sequence. The obtained results are shown in Figure 6. The base-layer size in the proposed MVC is again kept below 200 bytes/frame for all sequences. Remark that, since the motion information is highly correlated for the "Container" sequence, lossless compression with less than 200 bytes/frame is attained for some of the frames in the sequence.

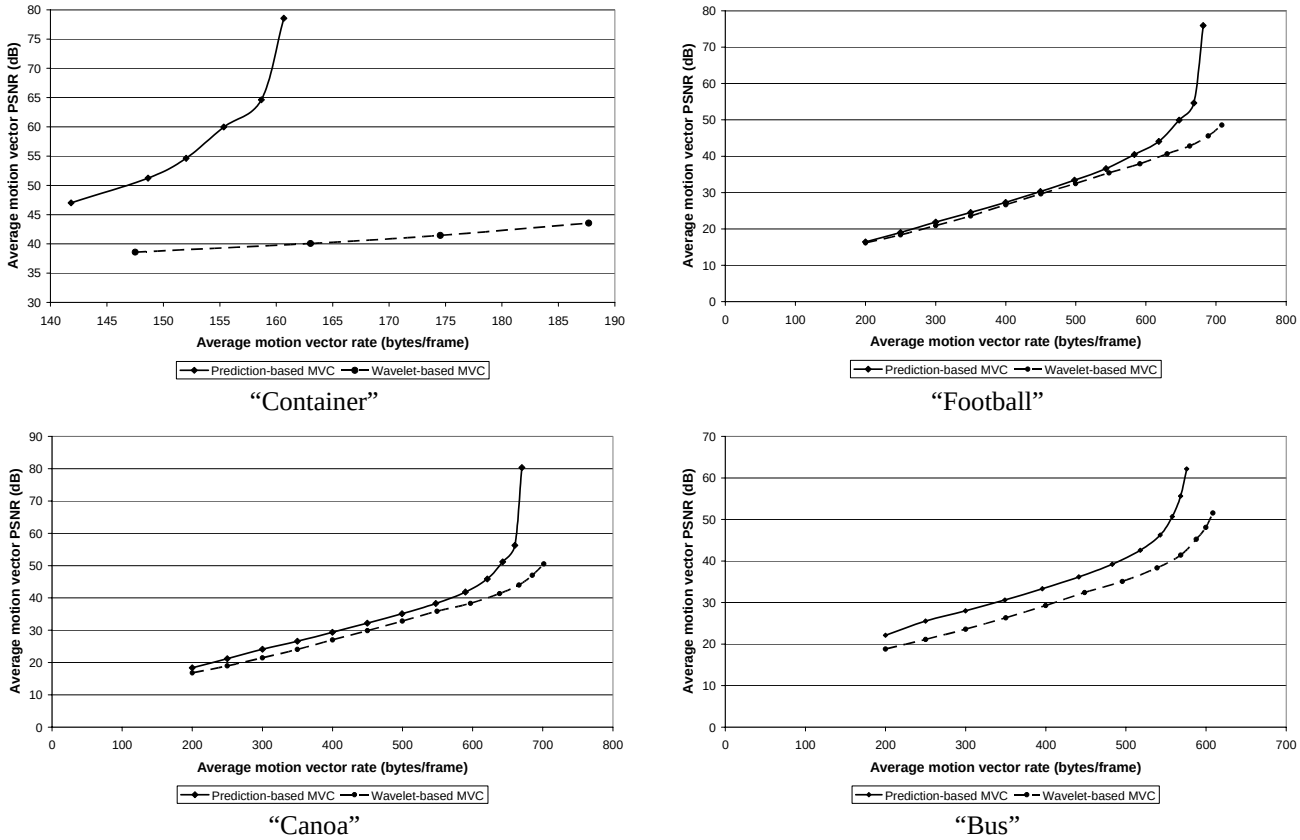


Figure 6: Comparison between the lossy compression performance of the prediction-based and wavelet-based scalable MVCs when coding motion information generated by multi-hypothesis block-based motion estimation using two reference frames, no intra mode, no macro-block splitting and quarter-pel accuracy.

From these results we can clearly draw the conclusion that, when coding motion information generated by multi-hypothesis block-based motion estimation, the prediction-based scalable MVC systematically outperforms wavelet-based scalable motion vector coding for both lossless and lossy compression.

Notice that the first experiments served to compare the coding performance of the scalable MVCs for motion information generated by block-based motion estimation. However, the wavelet-based scalable MVC of [13] was originally designed to code motion information generated by mesh-based motion estimation. Hence, in

order to ensure a fair comparison, we must also evaluate the coding efficiency of both scalable MVCs for this type of data. The mesh-based motion estimation algorithm used in this experiment employs a backward-mapped warping approach [25], bilinear interpolation and a 4-step iterative search of the node positions to obtain the optimal deformation of the mesh. This motion estimation algorithm is employed in a 4-level Haar MCTF-decomposition without update step (UMCTF). The test sequences described in Table 1 are used in this experiment. In the prediction-based scalable MVC the base-layer is restricted to 150 bytes/frame. Table 4 shows the lossless compression results, while the lossy compression results are summarized in Figure 7.

These results show that, when coding motion information generated by mesh-based motion estimation, the prediction-based scalable MVC again outperforms the wavelet-based scalable MVC for both lossless and lossy compression.

Sequence	Prediction-based scalable MVC	Wavelet-based scalable MVC
Football	472	495
Canoa	447	470
Bus	391	401
Container	139	173

Table 4: Lossless compression performance of the prediction-based and wavelet-based scalable MVCs when coding motion information generated by mesh-based motion estimation (the average number of bytes per frame is reported).

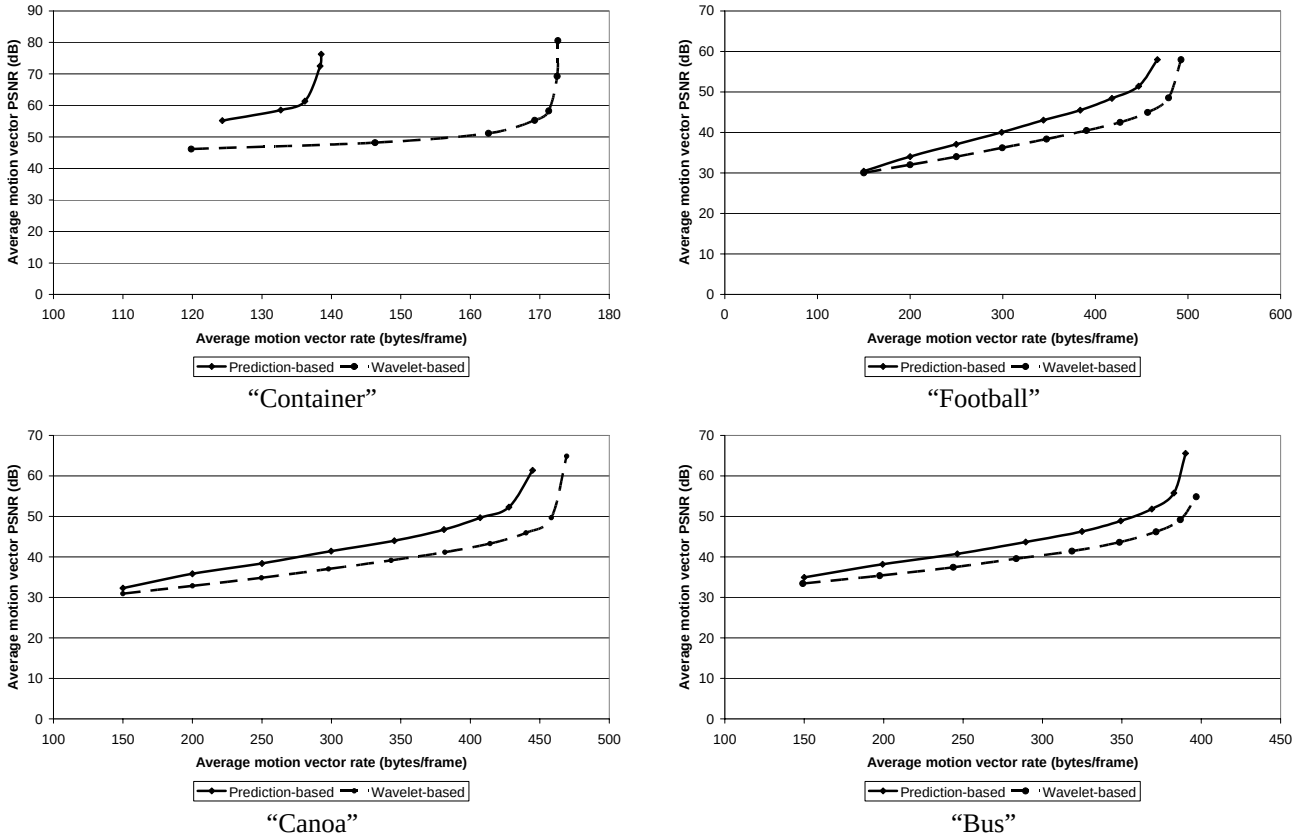


Figure 7: Comparison between the lossy compression performance of the prediction-based and wavelet-based scalable MVCs when coding motion information generated by mesh-based motion estimation

We conclude that, irrespective of the employed motion model, the prediction-based scalable MVC systematically outperforms the wavelet-based technique for both lossless and lossy compression. This might

be explained by the very nature of the motion vector field. The wavelet bases overlap the high density of singularities present in the motion vector field, creating a large number of high amplitude coefficients that are expensive to code. On the other hand, in contrast to the wavelet transform, which is linear, median-based prediction is a non-linear operation, and therefore, less affected by the presence of singularities. This results into a better compression performance, as observed in the experiments. Additionally, similar to the wavelet-based approach, the predictive-based MVC supports arbitrarily fine granularity of the motion vector rate, while providing better lossy coding performance at all rates higher than the base-layer rate.

IV.2 Benefits of scalable motion vector coding

In this set of experiments, a prediction-based MVC is incorporated into two different SDMCTF-based video codecs and the overall compression performance of the video codecs is compared when using scalable and non-scalable motion vector coding. In the first experiment, our video codec described in [6] is employed. It uses a 5/3 MCTF-decomposition without update step and multi-hypothesis motion estimation with two reference-frames, two block-sizes and quarter-pel accuracy [6]. The employed non-scalable motion vector codec is identical to the prediction-based motion vector codec used to encode the base-layer information in our scalable MVC (see section II.3). The base-layer size in the scalable MVC is kept below 96 kbps. When using the scalable MVC, the distribution of the rate between texture data and motion information is performed using the heuristic technique described in subsection III.2. When the non-scalable motion vector coding technique is employed, the bit-rate needed to losslessly code the motion information is subtracted from the target bit-rate and the remaining rate is allocated to the texture information, as described in subsection III.2. This means that the bit-rate available to code the texture information is allocated in the same way, regardless of the motion-vector codec used (scalable or not). Four levels of temporal and spatial decomposition are performed during encoding. The results of the experiment are shown in Table 5-Table 8 for the sequences described in Table 1. We report the average PSNR (in dB) for each color component and the average PSNR per frame calculated as [26]:

$$PSNR_{avg} = \left(4 \cdot PSNR_Y + PSNR_U + PSNR_V\right)/6 \qquad (19)$$

Visual results are shown in Figure 8 and Figure 9.

Football		Target bit-rate (kbps)					
		128	**192**	**256**	**384**	**512**	**1024**
Scalable MVC	$PSNR_Y$ (dB)	21.86	25.07	25.75	27.44	28.23	30.83
	$PSNR_U$ (dB)	27.70	29.07	30.47	31.99	32.94	35.43
	$PSNR_V$ (dB)	32.64	33.28	34.39	35.35	35.97	37.92
	$PSNR_{avg}$ (dB)	**24.63**	**27.10**	**27.98**	**29.52**	**30.30**	**32.78**
Non-scalable MVC	$PSNR_Y$ (dB)	-	22.99	25.99	27.46	28.24	30.84
	$PSNR_U$ (dB)	-	27.15	29.86	32.02	32.96	35.43
	$PSNR_V$ (dB)	-	32.46	33.83	35.37	35.98	37.93
	$PSNR_{avg}$ (dB)	-	**25.26**	**27.94**	**29.54**	**30.32**	**32.79**

Table 5: Comparison between scalable and non-scalable MVC employed in the codec of [6]: "Football" sequence. The symbol "–" signifies that the corresponding rate could not be met.

Scalable MVC

Non-scalable MVC

(a)

Scalable MVC

Non-scalable MVC

(b)

Figure 8: Comparison between scalable and non-scalable MVC employed in the codec of [6]: Visual comparison for the "Football"-sequence decoded at 192 kbps. (a) Frame 0. (b) Frame 8.

Canoa		Target bit-rate (kbps)					
		128	**192**	**256**	**384**	**512**	**1024**
Scalable MVC	$PSNR_Y$ (dB)	20.26	22.02	23.52	25.02	26.05	28.55
	$PSNR_U$ (dB)	30.93	32.46	33.13	34.25	34.86	36.42
	$PSNR_V$ (dB)	28.62	30.69	31.59	32.89	33.64	35.85
	$PSNR_{avg}$ (dB)	**23.43**	**25.21**	**26.46**	**27.87**	**28.78**	**31.08**
Non-scalable MVC	$PSNR_Y$ (dB)	-	-	23.16	25.13	26.10	28.58
	$PSNR_U$ (dB)	-	-	31.88	34.09	34.91	36.44
	$PSNR_V$ (dB)	-	-	30.32	32.75	33.69	35.87
	$PSNR_{avg}$ (dB)	-	-	**25.81**	**27.89**	**28.83**	**31.11**

Table 6: Comparison between scalable and non-scalable MVC employed in the codec of [6]: "Canoa" sequence. The symbol "–" signifies that the corresponding rate could not be met.

(a)

Scalable MVC Non-scalable MVC

(b)

Scalable MVC Non-scalable MVC

Figure 9: Comparison between scalable and non-scalable MVC employed in the codec of [6]: Visual comparison for the "Canoa"-sequence decoded at 256 kbps. (a) Frame 0. (b) Frame 2.

Bus		**Target bit-rate (kbps)**					
		128	**192**	**256**	**384**	**512**	**1024**
Scalable MVC	$PSNR_Y$ (dB)	19.76	23.25	24.89	27.15	28.23	31.20
	$PSNR_U$ (dB)	33.03	34.05	35.16	36.91	37.77	39.58
	$PSNR_V$ (dB)	34.23	35.64	36.32	37.97	39.16	41.23
	$PSNR_{avg}$ (dB)	**24.38**	**27.11**	**28.51**	**30.58**	**31.64**	**34.27**
Non-scalable MVC	$PSNR_Y$ (dB)	-	23.53	25.14	27.24	28.34	31.26
	$PSNR_U$ (dB)	-	34.05	35.42	37.02	37.84	39.60
	$PSNR_V$ (dB)	-	35.61	36.44	38.16	39.25	41.24
	$PSNR_{avg}$ (dB)	**-**	**27.30**	**28.74**	**30.69**	**31.74**	**34.31**

Table 7: Comparison between scalable and non-scalable MVC employed in the codec of [6]: "Bus" sequence. The symbol "–" signifies that the corresponding rate could not be met.

Container		**Target bit-rate (kbps)**					
		128	**192**	**256**	**384**	**512**	**1024**
Scalable MVC	$PSNR_Y$ (dB)	31.05	33.04	33.96	35.59	36.59	39.25
	$PSNR_U$ (dB)	38.58	40.65	41.65	43.12	44.08	46.76
	$PSNR_V$ (dB)	38.17	40.47	41.49	43.16	44.15	46.79
	$PSNR_{avg}$ (dB)	**33.49**	**35.55**	**36.49**	**38.11**	**39.10**	**41.76**
Non-scalable MVC	$PSNR_Y$ (dB)	31.06	33.05	33.96	35.59	36.60	39.25
	$PSNR_U$ (dB)	38.59	40.66	41.65	43.12	44.08	46.76
	$PSNR_V$ (dB)	38.18	40.48	41.49	43.16	44.15	46.80
	$PSNR_{avg}$ (dB)	**33.50**	**35.55**	**36.50**	**38.11**	**39.10**	**41.76**

Table 8: Comparison between scalable and non-scalable MVC employed in the codec of [6]: "Container" sequence. The symbol "–" signifies that the corresponding rate could not be met.

In a second experiment, the state-of-the-art scalable video codec developed by Microsoft Research Asia (MSRA) [9, 10] is equipped with a prediction-based scalable MVC. In this MVC, the base-layer motion

information is encoded using the prediction-based motion vector codec originally employed by the MSRA SVC codec. The enhancement layer is coded as described in subsection II.3.5. We again compare the performance of the video codec when using the scalable MVC and when using the original MVC of [9, 10] configured to generate a single-layer (i.e. non-scalable) motion representation. For the scalable MVC, the base-layer size is kept below 72 kbps. The distribution of the rate between texture data and motion information is performed using the heuristic technique described in subsection III.2. Four levels of temporal and spatial decomposition are used during encoding. Table 9-Table 10 summarize the results for the "Canoa" and "Bus" sequences described in Table 1. Visual results are shown in Figure 10.

Canoa		**Target bit-rate (kbps)**					
		96	**128**	**192**	**256**	**384**	**512**
Scalable MVC	$PSNR_Y$ (dB)	21.30	22.78	24.25	25.16	26.62	27.67
	$PSNR_U$ (dB)	28.69	32.45	34.02	34.62	35.09	35.85
	$PSNR_V$ (dB)	28.98	30.93	32.49	33.53	34.30	35.16
	$PSNR_{avg}$ (dB)	**23.81**	**25.75**	**27.25**	**28.13**	**29.31**	**30.28**
Non-scalable MVC	$PSNR_Y$ (dB)	-	14.67	24.20	25.39	26.86	27.84
	$PSNR_U$ (dB)	-	24.86	33.77	34.49	35.17	36.01
	$PSNR_V$ (dB)	-	25.66	31.94	33.50	34.49	35.26
	$PSNR_{avg}$ (dB)	**-**	**18.20**	**27.08**	**28.26**	**29.52**	**30.44**

Table 9: Comparison between scalable and non-scalable MVC (MSRA SVC codec): "Canoa" sequence. The symbol "–" signifies that the corresponding rate could not be met.

Bus		**Target bit-rate (kbps)**					
		96	**128**	**192**	**256**	**384**	**512**
Scalable MVC	$PSNR_Y$ (dB)	21.27	23.54	25.93	27.33	29.12	30.43
	$PSNR_U$ (dB)	32.04	35.21	36.05	37.04	38.20	38.90
	$PSNR_V$ (dB)	34.86	35.33	37.24	37.95	39.66	40.31
	$PSNR_{avg}$ (dB)	**25.33**	**27.45**	**29.50**	**30.72**	**32.39**	**33.49**
Non-scalable MVC	$PSNR_Y$ (dB)	20.99	23.91	26.21	27.50	29.24	30.51
	$PSNR_U$ (dB)	31.64	34.81	36.30	37.06	38.30	39.02
	$PSNR_V$ (dB)	33.90	35.33	37.38	38.03	39.66	40.50
	$PSNR_{avg}$ (dB)	**24.91**	**27.63**	**29.75**	**30.85**	**32.48**	**33.60**

Table 10: Comparison between scalable and non-scalable MVC (MSRA SVC codec): "Bus" sequence. The symbol "–" signifies that the corresponding rate could not be met.

Scalable MVC Non-scalable MVC

Figure 10: Comparison between scalable and non-scalable MVC (MSRA SVC codec) for the "Canoa" sequence decoded at 192 kbps, frame 0.

The benefits of integrating the designed quality-scalable MVC into an SDMCTF-based video codec are clearly demonstrated by these results; lower bit-rates can be attained without sacrificing motion estimation efficiency and the overall coding performance at low rates is improved by a better distribution of the available rate between texture and motion information. The only drawback of using scalable motion vector coding is the minor performance penalty at higher bit-rates (in the order of 0.05 dB for the video codec of [6] and 0.1-0.2 dB for the MSRA SVC codec of [9, 10]).

IV.3 Base-layer rate control and global rate allocation

The proposed mechanism to control the base-layer rate (see section III.1) finds the minimum quantization step size needed to meet the target base-layer bit-rate. The underlying assumption is that the performance of the prediction-based scalable MVC diminishes if the quantization step-size is increased. In the first experiment of this section, this assumption is verified. Motion information generated by a multi-hypothesis block-based motion estimation algorithm, using two reference frames, no intra mode, no macro-block splitting and quarter-pel accuracy, integrated into a 4-level 5/3 MCTF-decomposition without update step (UMCTF) is compressed with the prediction-based scalable MVC using different quantization step-sizes. The sequences described in Table 1 are used in the experiment. The results are shown in Table 11. We report the average number of bytes needed per H-frame to losslessly code the motion information. Q denotes the employed quantization step-size.

Sequence	Q=1 (no quantization)	Q=2	Q=4	Q=8	Q=16
Football	652	662	673	682	689
Canoa	642	650	656	664	677
Bus	536	546	549	567	594
Container	161	162	158	154	150

Table 11: Impact of the quantization step-size on the lossless compression performance of the prediction-based scalable MVC (The average number of bytes per frame needed to losslessly code the motion information is reported).

For three of the four test sequences, the compression performance of the prediction-based scalable MVC indeed diminishes as the quantization step-size is increased. Only for the "Container" sequence, a slight increase in compression performance is observed when Q is increased. However, since the obtained motion information bit-rate for the "Container" sequence is very small, the performance deficit caused by inappropriately selecting a low quantization step size is practically negligible. We can therefore conclude that the underlying assumption made in the proposed base-layer rate-control algorithm is appropriate.

In the next experiment, the proposed R-D-optimized rate allocation strategy is compared against our own heuristic rate-allocation technique, which was employed in [14, 18, 19]. The SDMCTF-based video codec of [6], using multi-hypothesis motion estimation with two reference-frames, two block-sizes and quarter-pel accuracy is equipped with the proposed scalable MVC and with both rate-allocation schemes. The codec uses an approximately unitary temporal transform (see subsection III.2). Four levels of temporal and spatial decomposition are performed. In both cases, the base layer bit-rate is kept below 96 kbps. The sequences presented in Table 1 are used in these experiments. The PSNR figures obtained with the two rate-allocation techniques are given in Table 12-Table 15.

Football		**Target bit-rate (kbps)**					
		128	**192**	**256**	**384**	**512**	**1024**
R-D optimized	$PSNR_Y$ (dB)	22.80	24.95	25.74	27.11	27.89	30.47
	$PSNR_U$ (dB)	29.50	31.39	32.61	34.31	34.96	37.62
	$PSNR_V$ (dB)	33.34	34.96	35.85	36.98	37.55	39.49
	$PSNR_{avg}$ (dB)	**25.67**	**27.69**	**28.57**	**29.95**	**30.68**	**33.17**
Heuristic	$PSNR_Y$ (dB)	21.86	25.07	25.75	27.44	28.23	30.83
	$PSNR_U$ (dB)	27.70	29.07	30.47	31.99	32.94	35.43
	$PSNR_V$ (dB)	32.64	33.28	34.39	35.35	35.97	37.92
	$PSNR_{avg}$ (dB)	**24.63**	**27.10**	**27.98**	**29.52**	**30.30**	**32.78**

Table 12: Comparison between the proposed rate allocation schemes: results for the "Football" sequence.

Canoa		**Target bit-rate (kbps)**					
		128	**192**	**256**	**384**	**512**	**1024**
R-D optimized	$PSNR_Y$ (dB)	20.98	22.87	23.75	24.93	25.76	28.29
	$PSNR_U$ (dB)	32.81	34.10	34.50	35.73	36.25	37.96
	$PSNR_V$ (dB)	31.08	32.60	33.19	34.93	35.60	37.83
	$PSNR_{avg}$ (dB)	**24.64**	**26.37**	**27.11**	**28.40**	**29.15**	**31.49**
Heuristic	$PSNR_Y$ (dB)	20.26	22.02	23.52	25.02	26.05	28.55
	$PSNR_U$ (dB)	30.93	32.46	33.13	34.25	34.86	36.42
	$PSNR_V$ (dB)	28.62	30.69	31.59	32.89	33.64	35.85
	$PSNR_{avg}$ (dB)	**23.43**	**25.21**	**26.46**	**27.87**	**28.78**	**31.08**

Table 13: Comparison between the proposed rate allocation schemes: results for the "Canoa" sequence.

Bus		Target bit-rate (kbps)					
		128	**192**	**256**	**384**	**512**	**1024**
R-D optimized	$PSNR_Y$ (dB)	20.84	23.58	24.97	27.18	28.30	31.11
	$PSNR_U$ (dB)	34.42	36.40	37.64	38.61	39.55	41.11
	$PSNR_V$ (dB)	35.72	37.28	39.13	40.10	41.04	42.77
	$PSNR_{avg}$ (dB)	**25.58**	**28.00**	**29.44**	**31.24**	**32.30**	**34.72**
Heuristic	$PSNR_Y$ (dB)	19.76	23.25	24.89	27.15	28.23	31.20
	$PSNR_U$ (dB)	33.03	34.05	35.16	36.91	37.77	39.58
	$PSNR_V$ (dB)	34.23	35.64	36.32	37.97	39.16	41.23
	$PSNR_{avg}$ (dB)	**24.38**	**27.11**	**28.51**	**30.58**	**31.64**	**34.27**

Table 14: Comparison between the proposed rate allocation schemes: results for the "Bus" sequence.

Container		Target bit-rate (kbps)					
		128	**192**	**256**	**384**	**512**	**1024**
R-D optimized	$PSNR_Y$ (dB)	31.06	32.79	33.87	35.31	36.40	38.96
	$PSNR_U$ (dB)	41.24	43.07	43.86	45.93	46.89	48.35
	$PSNR_V$ (dB)	41.08	43.06	43.93	45.85	46.87	48.78
	$PSNR_{avg}$ (dB)	**34.42**	**36.22**	**37.21**	**38.84**	**39.89**	**42.16**
Heuristic	$PSNR_Y$ (dB)	31.05	33.04	33.96	35.59	36.59	39.25
	$PSNR_U$ (dB)	38.58	40.65	41.65	43.12	44.08	46.76
	$PSNR_V$ (dB)	38.17	40.47	41.49	43.16	44.15	46.79
	$PSNR_{avg}$ (dB)	**33.49**	**35.55**	**36.49**	**38.11**	**39.10**	**41.76**

Table 15: Comparison between the proposed rate allocation schemes: results for the "Container" sequence.

The results of Table 12-Table 15 clearly show that in the average-PSNR sense the rate-distortion optimized approach systematically outperforms the heuristic rate-allocation strategy. A part of the gain in performance can be attributed to an improved allocation of the texture bit-rate. This can be noticed for instance from the results reported in Table 15 for the "Container" sequence. For this sequence, the motion vectors are always coded in a lossless fashion by both rate-allocation schemes because the motion vector bit-rate is negligible (around 9 kbps). By consequence, any difference in performance (up to 0.8 dB) is solely caused by a better allocation of the texture bit-rate. The other part of the performance gain occurs especially at low bit-rates and is caused by an optimized distribution of the rate between motion and texture information.

To provide a better insight into the R-D-optimized rate-allocation algorithm, the rate distribution obtained in the previous experiment for each target bit-rate was recorded. The results for the "Football" and "Canoa" sequences, encoded at 128, 384 and 1024 kbps are shown in Figure 12-Figure 17. The rates allocated to the texture and motion information are reported for each frame of the sequence. Additionally, the rate needed to losslessly code the motion information is shown as a reference. For each GOP, the frames are ordered according to their temporal level, from the A-frame of the highest temporal level to the last H-frame of temporal level 1. The ordering is illustrated in Figure 11 for the first GOP of the "Football" sequence,

encoded at 128 kbps. In this figure, A denotes the A-frame of the highest temporal level, while $H\ Lx-y$ denotes the y-th H-frame of temporal level x.

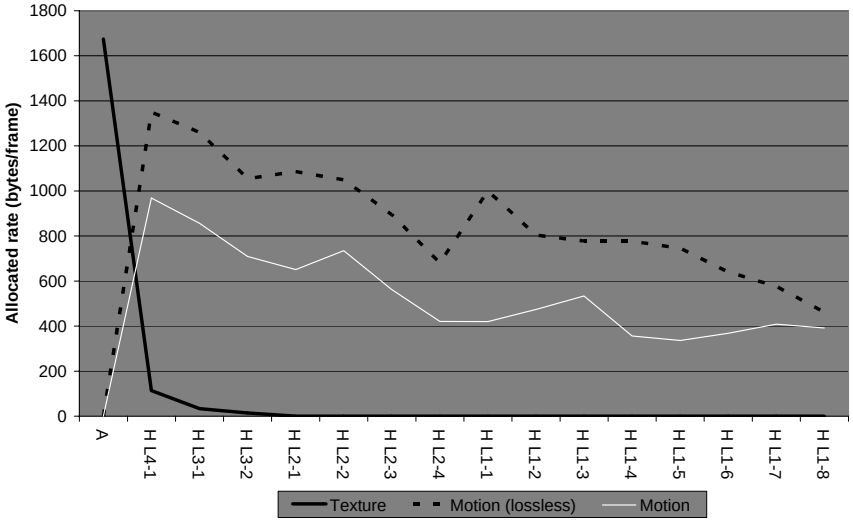


Figure 11: Rate distribution obtained for the first GOP of the "Football" sequence, encoded at 128 kbps. Notice that the frames are ordered according to descending temporal level, starting with the A-frame of the highest temporal level.

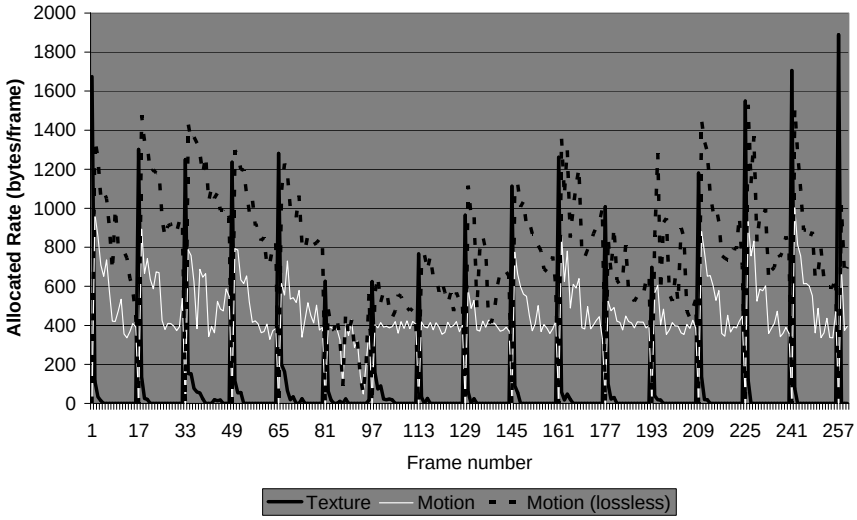


Figure 12: Rate distribution obtained for the "Football" sequence at a target bit-rate of 128 kbps.

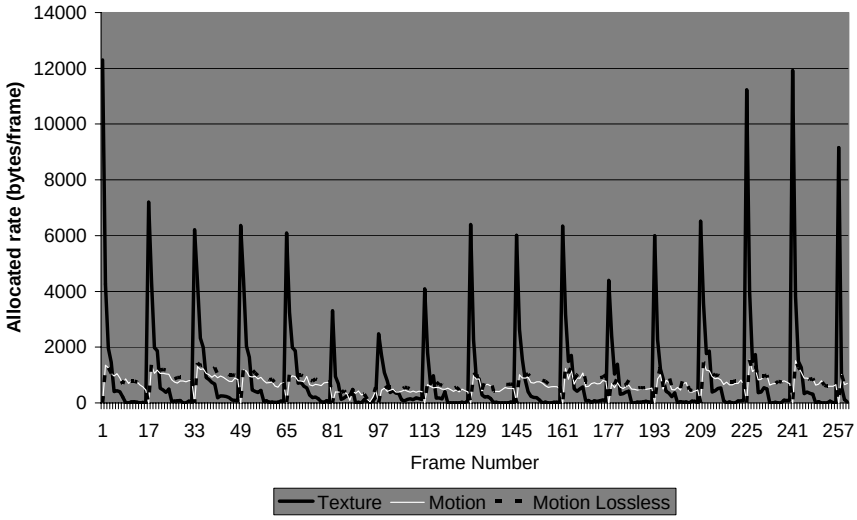


Figure 13: Rate distribution obtained for the "Football" sequence at a target bit-rate of 384 kbps.

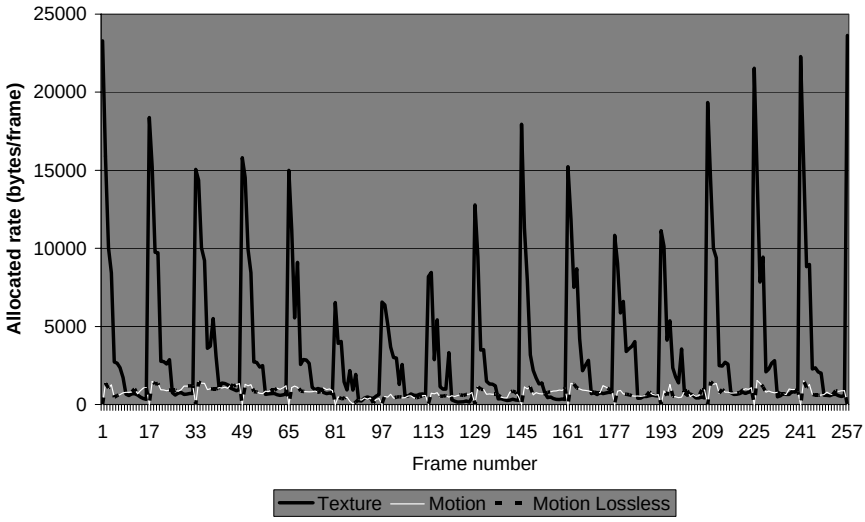


Figure 14: Rate distribution obtained for the "Football" sequence at a target bit-rate of 1024 kbps.

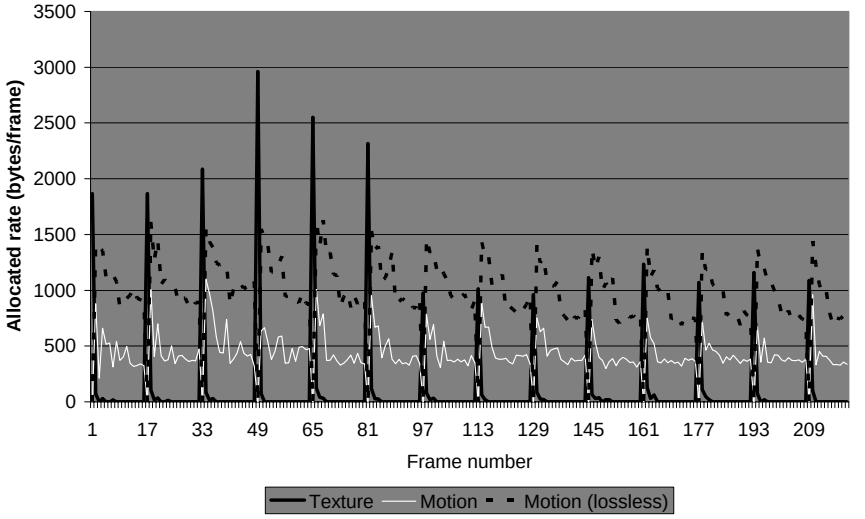


Figure 15: Rate distribution obtained for the "Canoa" sequence at a target bit-rate of 128 kbps.

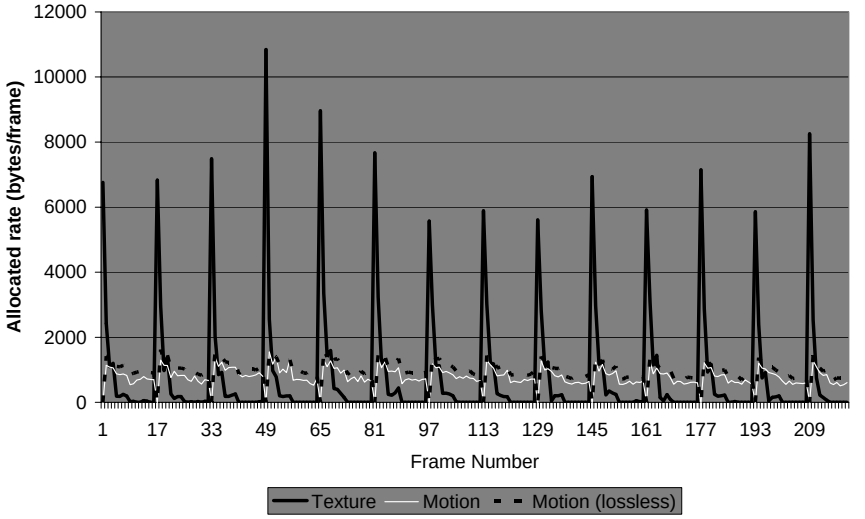


Figure 16: Rate distribution obtained for the "Canoa" sequence at a target bit-rate of 384 kbps.

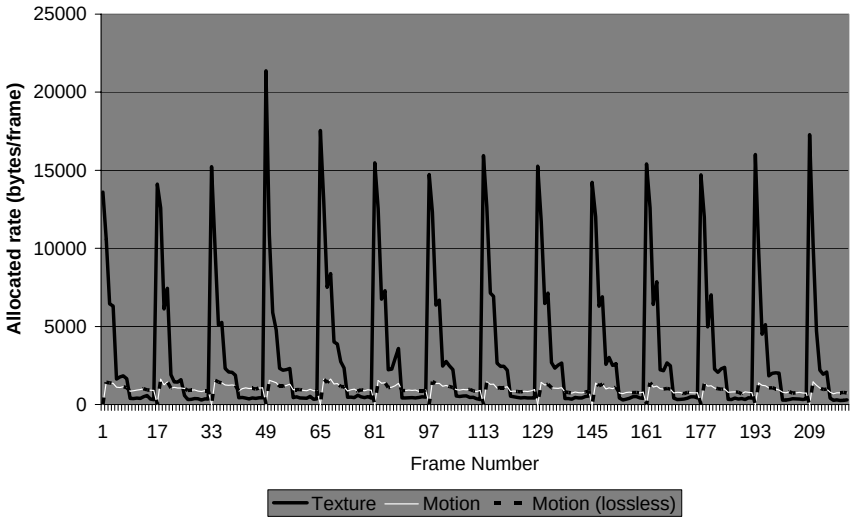


Figure 17: Rate distribution obtained for the "Canoa" sequence at a target bit-rate of 1024 kbps.

Based on these graphs, several observations can be made. Within a GOP, the rate spent to code a frame is directly proportional to the associated temporal level, with the largest part of the rate being spent on the A- and H-frames of the highest temporal level. This behavior is desirable since the impact on the overall distortion of quantization errors introduced at a certain temporal level t decreases with t. Additionally, the motion information bit-rate at 128 kbps is significantly lower than the lossless motion information bit-rate. This is consistent with the results obtained in section IV.2, showing that, for low target-bit-rates, a lower distortion can obtained by sacrificing motion information bit-rate in favor of texture bit-rate. As the target bit-rate is increased, the motion vector rate gradually increases and at high bit-rates, the motion information is practically coded losslessly. This is again in correspondence with the results in section IV.2, showing that, at higher target bit-rates, employing a non-scalable MVC (and thus lossless coding of the motion information) gives the best rate-distortion performance.

V. CONCLUSIONS

The scalable motion-vector coding technique we introduced in [14] combines the compression efficiency of classical prediction-based motion-vector codecs with support for scalability. The technique is designed to code motion information generated by multi-hypothesis block-based motion estimation with variable block-sizes and multiple reference-frames, and can be employed in any scalable video codec employing this motion model. The algorithm encodes the motion information by first quantizing the motion vectors. The quantized motion vectors and the side information (hypothesis information. splitting information and reference frame indices) are coded using non-scalable prediction-based motion vector coding. The resulting compressed data forms the base-layer of the final bit-stream. This base-layer must always be coded losslessly. The quantization errors are coded using an embedded bit-plane coding technique, producing the quality-scalable

enhancement layer. The coding architecture is completely different from that of current state-of-the-art scalable motion vector codecs [13], performing an integer wavelet-transform of the motion vector components, followed by scalable coding of the produced wavelet coefficients. Extensive experimental results show that the proposed scalable MVC systematically outperforms wavelet-based scalable MVCs.

To be able to effectively use the prediction-based scalable MVC, a novel rate allocation scheme is presented in this paper. The technique is based on Lagrangian rate-distortion optimization. The proposed rate allocation scheme is compared to the heuristic rate allocation scheme employed in [14, 18, 19]. Experiments show that the proposed rate allocation strategy yields the best output quality for any given target bit-rate.

The prediction-based scalable MVC was incorporated into the SDMCTF-based video codec of [6] and into the Microsoft SVC codec of [9, 10]. Experiments comparing the performance of these codecs using both scalable and non-scalable motion vector coding confirm that, by using a scalable MVC, lower bit-rates can be attained without sacrificing motion estimation efficiency and that the overall coding performance at low rates is significantly improved by a better distribution of the available rate between texture and motion information. The only drawback of scalable motion vector coding is a slight performance loss observed at higher bit-rates.

VI. ACKNOWLEDGEMENTS

This work was supported by the Flemish Institute for the Promotion of Innovation by Science and Technology (PhD bursary J. Barbarien), DWTC (IAP Phase V - Mobile Multimedia) and the Fund for Scientific Research – Flanders (FWO) (project G.0053.03.N and post-doctoral fellowships A. Munteanu and P. Schelkens).

VII. REFERENCES

[1] M. van der Schaar and H. Radha, "Adaptive motion-compensation fine-granular-scalability (AMC-FGS) for wireless video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 360-371, 2002.

[2] Y. He, R. Yan, F. Wu, and S. Li, "H.26L-based fine granularity scalable video coding," ISO/IEC JTC1/SC29/WG11, Pattaya, Thailand, M7788, 2001.

[3] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 559-571, 1994.

[4] D. Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 572-588, 1994.

[5] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 155-167, 1999.

[6] I. Andreopoulos, J. Barbarien, F. Verdicchio, A. Munteanu, M. van der Schaar, J. Cornelis, and P. Schelkens, "Response to call for evidence on scalable video coding," ISO/IEC JTC1/SC29/WG11 (MPEG), Trondheim, Norway, M9911, 2003.

[7] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with lifting implementations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 10, pp. 1183-1194, 2004.

[8] G. Pau, C. Tillier, B. Pesquet-Popescu, and H. Heijmans, "Motion compensation and scalability in lifting-based video coding," *Signal Processing: Image Communication, Special issue on subband/wavelet interframe video coding*, vol. 19, no. 7, pp. 577-600, 2004.
[9] J. Xu, R. Xiong, B. Feng, G. Sullivan, M.-C. Lee, F. Wu, and S. Li, "3D Sub-band Video Coding using Barbell lifting," ISO/IEC JTC1/SC29/WG11 MPEG, Munich, M10569/S05, 2004.
[10] L. Luo, F. Wu, S. Li, Z. Xiong, and Z. Zhuang, "Advanced motion threading for 3D wavelet video coding," *Signal Processing: Image Communication, Special issue on subband/wavelet interframe video coding*, vol. 19, no. 7, pp. 601-616, 2004.
[11] T. Wiegand and G. Sullivan, "Draft ITU-T recommendation and final draft international standard of joint video specification," ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6.
[12] J. Barbarien, I. Andreopoulos, A. Munteanu, P. Schelkens, and J. Cornelis, "Coding of motion vectors produced by wavelet-domain motion estimation," ISO/IEC JTC1/SC29/WG11 (MPEG), Awaji island, Japan, M9249, 2002.
[13] D. Taubman and A. Secker, "Highly scalable video compression with scalable motion coding," Proceedings of Int. Conf. Image Processing (ICIP 2003), vol. 3, pp. 273-276, Barcelona, Spain, 2003.
[14] J. Barbarien, A. Munteanu, F. Verdicchio, I. Andreopoulos, J. Cornelis, and P. Schelkens, "Scalable motion vector coding," *IEE Electronics Letters*, vol. 40, no. 15, pp. 932-934, 2004.
[15] S. S. Tsai, H.-M. Hang, and T. Chiang, "Motion information scalability for MC-EZBC: Response to call for evidence on scalable video coding," ISO/IEC JTC1/SC 29/WG 11 (MPEG), Trondheim, Norway, M9756, 2003.
[16] Y. Wu, A. Golwelkar, and J. W. Woods, "MC-EZBC video proposal from Rensselaer Polytechnic Institute," ISO/IEC JTC1/SC29/WG11 (MPEG), Munich, Germany, M10569/S15, 2004.
[17] M. Flierl and T. Wiegand, "Rate-constrained multihypothesis prediction for motion-compensated video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 957-969, 2002.
[18] J. Barbarien, A. Munteanu, F. Verdicchio, I. Andreopoulos, J. Cornelis, and P. Schelkens, "Scalable motion vector coding," Proceedings of IEEE International Conference on Image Processing, pp. 1321-1324, Singapore, 2004.
[19] J. Barbarien, A. Munteanu, F. Verdicchio, I. Andreopoulos, P. Schelkens, and J. Cornelis, "Scalable motion vector coding," Proceedings of SPIE International Symposium on Optical Science and Technology (SPIE's 49th Annual Meeting), vol. 5558, pp. 395-409, Denver, Colorado, U.S.A., 2004.
[20] I. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelis, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Processing: Image Communication*, vol. 19, no. 7, pp. 653-673, 2004.
[21] A. Munteanu,"Wavelet image coding and multiscale edge detection," PhD thesis, Department of Electronics and Information Processing (ETRO), Vrije Universiteit Brussel, 2003
[22] D. Taubman and M. W. Marcellin, "JPEG2000 - Image Compression: Fundamentals, Standards and Practice." Hingham, MA, Kluwer Academic Publishers, 2001.
[23] P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro i Nieto, and J. Cornelis, "Wavelet Coding of Volumetric Medical Datasets," *IEEE Transactions on Medical Imaging*, vol. 22, no. 3, pp. 441-458, 2003.
[24] A. Said and W. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Transactions on Image Processing*, vol. 5, pp. 1303-1310, 1996.
[25] G. Wolberg, "Digital Image Warping," IEEE Computer Society Press, 1992.
[26] A. Golwelkar, I. Bajic, and J. W. Woods, "Response to Call for Evidence on Scalable Video Coding," ISO/IEC JTC1/SC29/WG11 (MPEG), Trondheim, Norway, M9723, 2003.