

# SCALABLE CODING OF VARIABLE SIZE BLOCKS MOTION VECTORS

*Davide Maestroni, Augusto Sarti, Marco Tagliasacchi, Stefano Tubaro,*

Dipartimento di Elettronica e Informazione - Politecnico di Milano  
Piazza Leonardo da Vinci 32, 20133 Milano - Italy

## ABSTRACT

In this paper we discuss an algorithm that is able to provide a scalable (multiresolution) representation of the motion field information. It has been recently demonstrated that, for an open loop wavelet video coder, it is possible to use at the decoder side a scaled version of the original motion information and the residual coefficients computed with the full resolution/quality motion field without incurring into drift. We propose a fully scalable wavelet based video coder that performs motion estimation-compensation in the wavelet domain. In particular the coding scheme is specifically designed for variable size block matching algorithms. In this scenario motion vectors are distributed across an irregular lattice according to a quadtree structure. The developed system allows a scalable representation of the motion field and a flexible allocation of the bit budget between motion and residual data. The simulations that we have carried out show, at low bit-rates, a significant gain of the proposed approach with respect to the case in which the motion information is coded lossless.

## 1. INTRODUCTION

Today's video streaming applications demand for coders to provide a bitstream that can be flexibly adapted to the characteristics of the network and the receiving device. In other words such coders have to fulfill the scalability requirements in such a way that encoding is performed only once, while decoding takes place at different resolutions, frame rates and bitrates. Wavelet based video coders proved to be able to naturally support scalability owing to their open loop structure, therefore avoiding the problem of drift that affects in-loop conventional DPCM coders. Traditionally, only the wavelet coefficients can be actually scaled, while motion information still occupies a fixed amount of the bit budget decided at encoding time and unaware of the decoding bitrate. This fact has two major drawbacks. First, the video sequence cannot be encoded at a target bitrate lower than the one necessary to encode the motion vectors in a lossless way. Moreover no optimal trade-off between motion and residuals bit budgets can be computed. Recently it has been

demonstrated [1] that in the case of open-loop wavelet based video coders it is possible to use a quantized version of the motion field during decoding together with the residual coefficients computed with the lossless version of the motion. A scalable representation of the motion is achieved by coding the motion field as a two-component image using a JPEG2000 like scheme. This is possible as long as the motion vectors lie on a regular lattice, as it is the case for fixed size block matching or deformable meshes using equally spaced control points. In this paper we introduce a novel algorithm able to build a scalable representation of the motion vectors which is specifically designed for variable size block matching. We tested our implementation using, as input, the motion field provided by an algorithm similar to HVSBM (Hierarchical Variable Size Block Matching) [2] that performs the estimate operating in the wavelet domain. Nonetheless any motion estimation algorithm producing a quadtree like data structure can be used. In Section 2 we briefly review the main concepts related to motion vector scalability. Section 3 contains a detailed description of our algorithm while experimental results and conclusions are reported in Section 4 and 5.

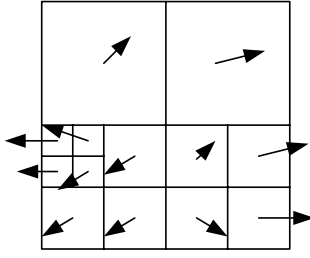
## 2. MOTION VECTORS SCALABILITY

In this section we briefly summarize the results contained in [1], in order to emphasize the theoretical justification for motion vector scalability. Let us consider a 2D signal  $y[\mathbf{n}] = W(x)[\mathbf{n}]$ , which is the warped version of a signal  $x[\mathbf{n}]$ . If a quantized version of motion is available through the operator  $W'$ , we are able to obtain  $y'[\mathbf{n}] = W'(x)[\mathbf{n}]$ . For the sake of simplicity we assume a constant displacement error  $\delta$ , so that  $y'[\mathbf{n}] = y[\mathbf{n}-\delta]$ . The total square error turns out to be related to the spectral characteristics of the image:

$$D_y = \sum |y[\mathbf{n}] - y'[\mathbf{n}]|^2 \approx \|\delta\|^2 \frac{\psi_{1,x} + \psi_{2,x}}{2}$$

Where  $\psi_{1,x}$  ( $\psi_{2,x}$ ) is the horizontal (vertical) sensitivity factor that assumes higher values when the energy is concentrated at high frequencies:

$$\psi_{1,x} = \frac{1}{(2\pi)^2} \iint S_x(\omega) \omega_1^2 d\omega$$



**Figure 1 – Quadtree structure of the motion field estimated by HVSBM**

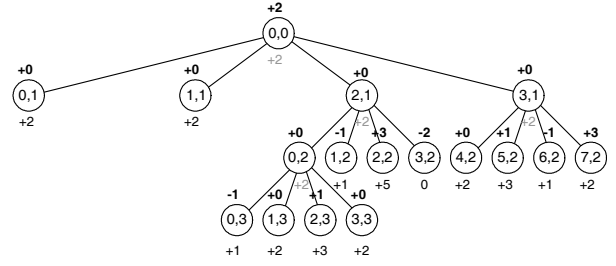
This means that for small  $\delta$  there exists a linear model relating the error due to motion vector quantization and the error of prediction residuals. This has been generalized to the case of non-constant motion error yielding:

$$D_y \approx k \frac{\psi_{1,x} + \psi_{2,x}}{2} D_w$$

Where  $D_w$  is the mean squared error in the motion vector parameters. The constant parameters  $k$  depends on the motion estimation approach adopted and turns out to be 1 for block matching and  $2/3$  for deformable triangular meshes. In [1] motion estimation-compensation is obtained using the latter approach and control points are distributed on a regular grid. The resulting motion vector field is then spatially transformed using a 5/3 DWT and temporal transformed using Haar in order to reduce the redundancies existing in the motion field representation. The resulting wavelet coefficients are coded with a JPEG2000 scheme providing a layered scalable representation. While the encoder adopts the lossless representation of motion in order to compute the spatio-temporal analysis, the decoder receives a quantized version of the motion field. Both brute force and model-based rate allocation strategies are described in the same paper. This scheme cannot be adopted as it is in the case of variable size block matching motion estimation. In fact motion vectors are organized in an irregular lattice as shown in Figure 1 and a conventional 2D DWT cannot be performed to reduce spatial redundancy. The next section details the proposed algorithm, which is specifically conceived, to deal with the quadtree structure computed by HVSBM.

### 3. MULTIREOLUTION REPRESENTATION OF MOTION VECTORS

The HVSBM algorithm produces a quadtree data structure in output, as illustrated in Figure 1. Block sizes range from  $N_{max} \times N_{max}$  to  $N_{min} \times N_{min}$  and they tend to be smaller in regions characterized by complex motion. Neighboring blocks usually manifest a high degree of similarity, therefore a coding algorithm able to reduce spatial redundancy is needed. In the standard implementation of HVSBM a simple nearest neighbor predictor is used for this purpose. Although it achieves a good lossless



**Figure 2 - Tree-like representation of the motion vector field generated by HVSBM. The bold numbers are the values that are actually encoded. Only one component of the motion vectors is shown**

compression ratio, it does not provide a scalable representation. The proposed algorithm aims at achieving the same performances when operates lossless allowing at the same time a scalable representation of the motion information. In order to tackle spatial redundancy a multiresolution pyramid is built in a bottom-up fashion. At the beginning of the algorithm only the leaf nodes are assigned with a value, representing the two components of the motion vector. Although HVSBM provides motion vector estimates also for intermediate nodes, we decided not to consider them. Conversely, independently for each component, we compute the value of the node as a simple average of its four offspring, since that strategy turned out to be the most effective choice as far as coding efficiency is concerned. Then we code each offspring as the difference between each value and its parent. We iterate these steps further up the motion vector tree. At the top level, HVSBM has as many nodes as the largest block size. Nonetheless we keep on grouping blocks until only one root node is formed. The latter will contain an average of the motion vectors over the whole image. Depending on the size of the image and  $N_{min}$  the root node might have fewer than four offspring. Figure 2 illustrates a toy example that clarifies how our multiresolution representation is built starting from the HVSBM tree. The motion vectors are the numbers just below each leaf node. The averages computed on intermediate nodes are shown in grey, while the values to be encoded are written in bold typeface. The same figure also shows the labeling convention we use: each node is identified by a pair  $(i,d)$  where  $d$  represents the depth in the tree while  $i$  is the index number starting from zero of the nodes at a given depth.

Since the motion field usually exhibits a certain amount of spatial redundancy the leaf nodes are likely to have a smaller absolute values. In other words walking down from the root to the leaves we can expect the same sort of decay that is specific of wavelet coefficients across subbands following parent-children relationships. This fact suggested us that the same ideas underpinning

wavelet based image coders could be exploited here. Specifically, if an intermediate node turns out to be insignificant with respect with a given threshold, then it is likely that its descendants are also insignificant. That is why the proposed algorithm inherits some of the basic concepts of SPIHT [3] (Set Partitioning In Hierarchical Trees). Before detailing the steps of the algorithm, it is important to point out that in the quadtree representation that we have built so far the nodes should be weighted accordingly to their depth in the tree. Let us consider only one node and its four offspring. If we wish to achieve a lossy representation of the motion field, these nodes will be quantized. If we make an error in the parent node, that will badly affect its offspring, while the same error will have fewer consequences if one of the children is involved. If we use the mean squared error as a distortion measure, it turns out that the parent node needs to be multiplied by a factor of 2, in such a way that errors are weighted equally and the same quantization step sizes can be used regardless of the node depth.

The proposed algorithm encodes the nodes of the quadtree from top to bottom starting from the most significant bitplane. As in SPIHT, the algorithm is divided into a sorting pass that identifies which nodes are significant with respect to a given threshold, and a refinement pass that refines the nodes already found significant in the previous steps. There are four lists that are maintained both at the encoder and the decoder, which allow to keep track of each node status. The LIV contains those nodes that have not been found significant yet. The LIS represents those nodes whose descendants are insignificant. On the other hand, LSV and LSS contain either nodes found significant or whose descendant are significant. A node can be moved from LIV to LIS and from LIS to LSS but not vice versa. Only the nodes in the LSV are refined during the refinement pass. The following conventions are used throughout the rest of this paper:

- $P(i, d)$ : coordinates of parent node of node  $i$  at depth  $d$
- $O(i, d)$ : set of coordinates of all offspring of node  $i$  at depth  $d$
- $D(i, d)$ : set of coordinates of all descendants of node  $i$  at depth  $d$
- $H(0, 0)$ : coordinate of the quad-tree root node

$$S_n(i, d) = \begin{cases} 1, & |c_{i,d}| \geq 2^n, \quad 0 \leq n \leq msb \\ 0, & otherwise \end{cases}$$

The textbox below shows the details of the proposed algorithm. Note that  $\hat{d}$  keeps track of the depth of the current node. This way instead of scaling by a factor of 2 all the intermediate nodes with respect to their offspring, the significance test is carried out at bitplane  $n + \hat{d}$ , i.e.  $S_{n+\hat{d}}(i_d, \hat{d})$ .

As for SPIHT, encoding and decoding uses the same algorithm, where the word output is substituted by input at the decoder side. The symbols emitted by the encoder are

- 1) Initialization:**
  - output  $msb = n = \lfloor \log_2(\max_{(i,d)} \{|c_{i,d}|\}) \rfloor$ ;
  - output  $max\_depth = \max(d)$ ;
  - set the LSS and the LSV as empty lists
  - add  $H(0, 0)$  to the LIV and to the LIS;
- 2) Sorting Pass:**
  - 2.1) set  $\hat{d} = 0$ , set  $i_d = 0$  ( $d = 0, 1, \dots, max\_depth$ );
  - 2.2) if  $0 \leq n + \hat{d} \leq msb$  do:
    - 2.2.1) if entry  $(i_d, \hat{d})$  is in the LIV do:
      - output  $S_{n+\hat{d}}(i_d, \hat{d})$ ;
      - if  $S_{n+\hat{d}}(i_d, \hat{d}) = 1$  then move  $(i_d, \hat{d})$  to LSV and output the sign of  $c_{i_d, \hat{d}}$ ;
    - 2.3) if entry  $(i_d, \hat{d})$  is in the LIS do:
      - 2.3.1) if  $n + \hat{d} < msb$  do:
        - $SD = 0$ ;
        - for  $h = \hat{d} + 1$  to  $max\_depth$  do:
          - for each  $(j, h) \in D(i_d, \hat{d})$  do:
            - if  $S_{n+h}(j, h) = 1$  then  $S_D = 1$ ;
        - output  $S_D$ ;
        - if  $S_D = 1$  then move  $(i_d, \hat{d})$  to LSS, add each  $(k, l) \in O(i_d, \hat{d})$  to the LIV and to the LIS, increment  $\hat{d}$  by 1, and go to Step 2.2);
    - 2.4) if entry  $(i_d, \hat{d})$  is in the LSS then increment  $\hat{d}$  by 1 and go to Step 2.2);
- 3) Refinement Pass:**
  - 3.1) if  $0 \leq n + \hat{d} \leq msb$  do:
    - 3.1.1) if entry  $(i_d, \hat{d})$  is in the LSV and was not included during the last sorting pass, then output the  $n$ th most significant bit of  $|c_{i_d, \hat{d}}|$ ;
  - 3.2) if  $\hat{d} \geq 1$  do:
    - 3.2.1) increment  $i_d$  by 1;
    - 3.2.2) if  $(i_d, \hat{d}) \in O(P(i_d - 1, \hat{d}))$  then go to Step 2.2); otherwise decrement  $\hat{d}$  by 1 and go to Step 3);
- 4) Quantization-Step Update:** decrement  $n$  by 1 and go to Step 2).

arithmetic coded. Currently a context independent coder is used. We argue that a context adaptive arithmetic coding would yield better results.

#### 4. EXPERIMENTAL RESULTS

We carried out several experiments in order to assess the performances of the proposed algorithm. All the tests are performed using our implementation of the IB-MCTF (In-

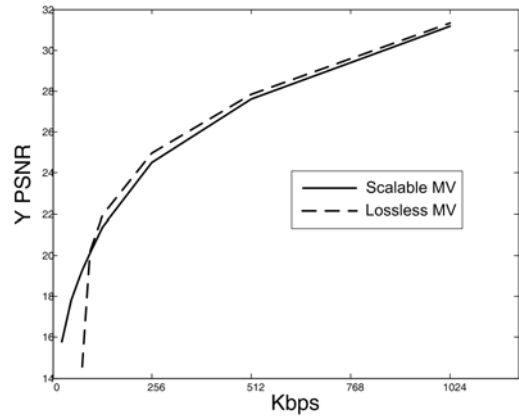
**Table 1 - Lossless motion information bitrates for some CIF test sequences at quarter pixel accuracy**

	Predictive Coding	Scalable Coding
<b>Mobile</b>	45 Kbps	53 Kbps
<b>Foreman</b>	57 Kbps	76 Kbps
<b>Football</b>	102 Kbps	142 Kbps
<b>Bus</b>	69 Kbps	96 Kbps

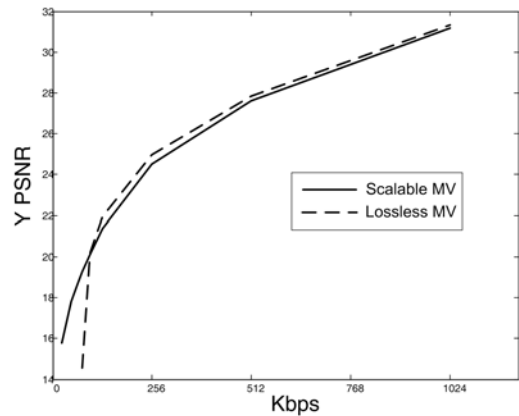
band Motion Compensated Temporal Filtering) coding scheme [4]. First, we compared the proposed algorithm in the lossless configuration with respect to the predictive coder available in the standard implementation of HVSBM. Our tests demonstrate that in this scenario the predictive coder achieves a better compression ratio as shown in Table 1. All the sequences have CIF resolution and motion is estimated with quarter pixel accuracy. The largest block size is 32x32 while the smallest blocks are 4x4. The nice properties of our algorithm emerge when a lossy representation is needed to achieve scalability of motion information. In the current implementation the best trade-off between motion and residuals is simply computed by brute force. A look-up table is built by the encoder and added as header information in such a way that the decoder knows for each spatio-temporal resolution and target bitrate what is the optimal amount of bit budget that has to be spent for motion. Figure 3 shows the average Y PSNR for the *Bus* sequence at CIF resolution and 30fps. The dashed lines is obtained using the predictive coding of motion vectors, whereas the solid line represents our algorithm. It can be noticed that scalability of motion allows to reach the low bitrate range that is unreachable by the predictive coder. Moreover at the cost of no more than 0.1-0.5dB at high bitrates, a gain of up to +4dB is registered at low bitrates. The same conclusions can be drawn from the *Foreman* sequence (see Figure 4).

## 6. CONCLUSIONS

We propose an algorithm that performs a scalable compression of the motion field in the case of variable size block matching algorithm such as HVSBM. Experimental results demonstrate that a significant gain can be achieved at low bitrates when scalability of motion information is in place. Future research activities will focus on the way to compute a spatially adaptive trade-off between motion and residuals, following the ideas presented in [1].



**Figure 3- Bus, CIF@30fps. Performance of the IB-MCTF coder with different types of motion field**



**Figure 4 - Foreman, CIF@30fps. Performance of the IB-MCTF coder with different types of motion field**

## 6. REFERENCES

- [1] A. Secker and D. Taubman, "Lifting-based Invertible Motion Adaptive Transform (LIMAT) Framework for Highly Scalable Video Compression". Submitted to IEEE Trans. on Image Processing, May 2003
- [2] S. Choi, J.W. Woods, "Motion-Compensated 3-D Subband Coding of Video". In IEEE Trans. on Image Processing, vol. 8, No. 2, February 1999
- [3] A. Said, W.A. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees". In IEEE Trans. on Circuits and Systems for Video Technology, vol. 6, NO. 3, June 1996.
- [4] J.C. Ye, M. van der Schaar, "Fully Scalable Overcomplete Wavelet Video Coding using Adaptive Motion Compensated Temporal Filtering". In Proceedings of VCIP2003, July 2003, Lugano, Switzerland