AN ENTROPY CODING SCHEME FOR MULTI-COMPONENT SCALABLE MOTION INFORMATION

Toni Zgaljic, Marta Mrak, Nikola Sprljan and Ebroul Izquierdo

Multimedia and Vision Lab, Queen Mary, University of London
Mile End Road, E1 4NS, London, United Kingdom
{toni.zgaljic, marta.mrak, nikola.sprljan, ebroul.izquierdo}@elec.qmul.ac.uk

ABSTRACT

Fully scalable video bit-stream requires layered structure of most of its components. For that reason few methods targeting scalability on motion information have been proposed over the last decade. However, layered representation requires new entropy coding strategies able to efficiently handling of redundancies between different layers. In this paper three methods for entropy coding of layered motion information are proposed. The influence of these schemes on the reconstructed video quality has been also studied.

1. INTRODUCTION

In applications where adaptability of multimedia content is required, scalable coding has emerged as an inevitable tool. For the adaptation of video content fine-granular scalability is achieved using embedded coding of texture information. To enable this, different approaches, originally developed for still image coding, have been adopted. However, on low bit-rates, when just a fraction of the full texture information remains in the adapted bit-stream, it is desirable to have a scalable structure of remaining data - motion information. When a specific low bit-rate is targeted, transmission of additional texture information can be achieved by discarding a suitable amount of motion information. If a well balanced ratio between texture and motion information is achieved, the resulted adapted video sequences will show better quality.

Although standard motion information coding techniques provide unbeatable compression ratios, they do not support scalable motion representations. In this paper we focus on two main scalable representations of motion information, namely motion with layered structure and layered motion vector value scalability. For these two representations we propose an entropy coding scheme that enables layered coding. The proposed scheme distinguishes implementations of binary and m-ary arithmetic coder with context modeller. In such way data containing binary symbols can be encoded through faster binary arithmetic encoder. Similar to the approach introduced in [1] and in order to exploit the correlation of the data being encoded, prediction is used and prediction error is coded. Since some inter-symbol redundancies still remain after prediction, context modelling for encoding of binary symbols is employed to achieve better compression. This is done by following the schema reported in [2].

The remainder of this paper is organised as follows: In section 2 the organisation and types of scalable motion information are presented. Section 3 describes the entropy coding technique. Selected results are reported in section 4 and the paper closes outlining the conclusions of this work in section 5.

2. SCALABLE MOTION INFORMATION

Motion information consisting of macroblock partitioning (motion structure), modes of macroblock and final nodes as well as motion vector values is considered. In the same applications, each motion block also contains the information on reference frame index, which is included in the motion block mode information. Without lost of generality we use macroblock partitioning into square blocks. The partitioning of a macroblock can be represented by a tree structure as shown in Figure 1 for example using two tree levels. The motion tree root corresponds to the macroblock while the external (leaf) nodes correspond to each final motion block used for the temporal decomposition. For the external nodes of a motion tree block the mode (intra and subclasses of inter modes) is known. If a block is an inter block, the corresponding external tree node carries the information on associated motion vectors. Motion vector values belong to a finite set whose number of elements depends on the motion vector range and their precision, e.g. 1/4.

The types of scalability imposed on motion information can be classified into two basic groups: scalability of motion structure and scalability of motion vector values. It is also possible to combine these two resulting in more flexible

This work was supported by the European Commission under contract FP6-001765 aceMedia.

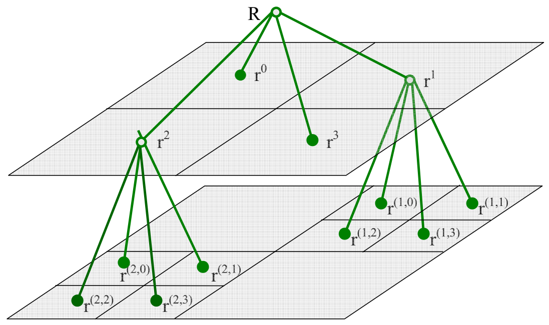


Figure 1 Macroblock partitioning described by a tree structure.

representations.

Although the motion compensation is performed using so called full structure, inverse operation at the decoder can be performed using the subsets of the original motion trees. Here, the subset (T_m) of one tree shares the same root node as the original tree (T). The decoding from the subsets is possible as long as the external nodes of the subsets carry the information on the block mode and motion vectors. The layers of the motion structure are formed by the coarser trees and the differences between structure layers. The 0-th layer consists of the coarsest subset of the motion tree (T_0) for each macroblock. If the targeted number of motion structure layers is M, i.e., $T = T_{M-1}$, then each m-th layer consists of the differential structure between two consecutive trees, i.e. of ($T_m \setminus T_{m-1}$) and a properly chosen connections between the T_{m-1} and the differential structure.

As the temporal frames are obtained using full motion structure, for the reconstruction, using coarser structure motion vectors in internal full motion trees nodes have to be carefully selected. The method recently proposed in [3] enables optimal selection of motion vectors for the internal nodes of the full motion trees for unidirectional UMCTF. This method is also used in this work.

The scalability of compressed motion information can be also achieved using scalable coding of motion vector values. The main idea in such approaches is to use bit-plane coding of the values or the combination of base layer (coded using variable length code, VLC) and bit-plane coding. A scheme named Precision Limited Coding (PLC) that enables scalable representation of motion vector values has been proposed in [4].

In this paper we combine these approaches to enable higher motion scalability. Motion vectors corresponding to each structure layer are encoded using PLC. The scheme for joint encoding is presented in Figure 2. PL represents the part of motion vector value (MV) that consists of the lowest bit-planes (R_1,..., R_N) of MV. Each R_i is encoded in a separate layer. The base layer consists of the differential structure between structure layer m and structure layer (m - 1). The entropy coding module is described in the following section.

3. CODING OF SCALABLE MOTION INFORMATION

For each layer of motion structure joint encoding with PLC is proposed (see Figure 2). The compressed base layer of m-*th* structure layer consist of motion structure related parameters and corresponding precision limited components of the motion vectors. The enhancement sub-layers consist of lower bit-planes of motion vector values. PLC values are differentially encoded by prediction from already encoded PLC values in the entropy coding module. Modelling and arithmetic coding scheme used for all components is based on [5]. Depending on alphabet size of a symbol to be encoded, either m-ary or binary arithmetic encoding engine is used. The two encoding engines are distinguished because binary arithmetic encoder requires less multiplication and division operations comparing to m-ary

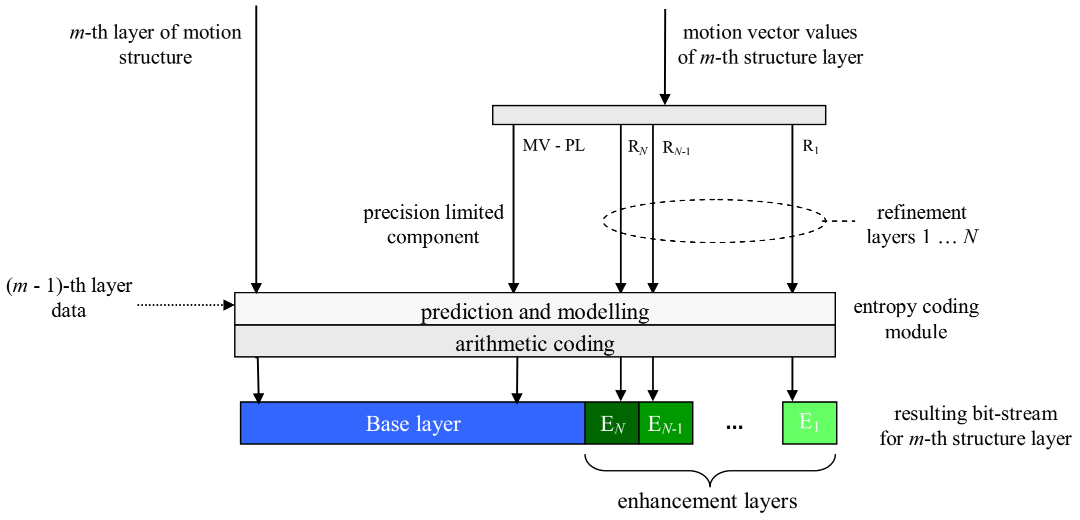
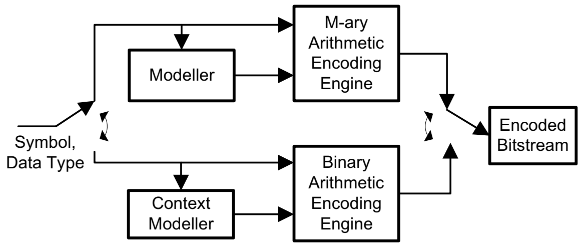


Figure 2 Modified PLC with joint encoding of motion structure layer.



M-ary Arithmetic Encoding Engine
Modeller
Encoded Bitstream
Symbol, Data Type
Binary Arithmetic Encoding Engine
Context Modeller

Figure 3 Encoding block diagram.

arithmetic encoder. Also, the modelling stage for the binary arithmetic encoder is less complex, therefore it is more effective. However, m-ary codec is used for non-binarised precision limited values of motion vector prediction differences as it provides high compression efficiency for those still highly correlated values.

Arithmetic encoding engine takes the probability estimate of the symbol to be encoded from the modeller which is updated with each occurring symbol. For the binary encoding case exists an additional phase of context modelling in which probability model is selected depending on the previously encoded symbols.

The symbol on the encoder input can belong to one of the data types shown in Table 1. For each data type number of contexts employed and size of the alphabet is shown.

Data type	Number of contexts	Alphabet size
mv_diff_row	-	m-ary
mv_diff_col	-	m-ary
mv_map	3	binary
mv_ref_diff_row	2	binary
mv_ref_diff_col	2	binary

Table 1 Data Types and coding settings.

The *mv_map* data type represents motion tree structure, i.e. partitioning of blocks into smaller ones. Therefore it contains information if the current block of the current structure layer is divided into the smaller blocks or not. The alphabet size for this type is two and for coding modeller it uses three contexts determined by the following rules:

$$context\_no = \begin{cases} 0, \text{ if } actc(left) + actc(above) = 0 \\ 1, \text{ if } actc(left) + actc(above) = 2 \\ 2, \text{ otherwise} \end{cases},$$

where *actc(left)* = 1 if the block on the left of the current block is further divided in the current layer, 0 if not, and *actc*(*above*) = 1 if the block above of the current block is further divided in the current layer, 0 if not. For modelling within each context histogram of previously occurred symbols is used.

The *mv_diff_row* and *mv_diff_col* data types represent the difference between predicted and actual precision limited values of motion vector row and column component respectively. The alphabet size for these data types depends on motion vector range, current level of temporal decomposition, motion vector precision and number of refinement bits. For the current layer of the motion structure, neighbouring motion vectors in the same motion structure layer and motion vectors from previous layers are used for prediction. The predicted value of the current block motion vector from the neighbouring blocks is calculated as the sum of weighted motion vector values from the block above and left of the current block. Weighting factors w_i for the neighbour block B_i are obtained using

$$w_i = \frac{N_p(B_i, B_x)}{N_{px}(B_x) + N_{py}(B_x)}, \qquad (1)$$

where $N_p(B_i, B_x)$ is the number of neighbouring pixels between the current block B_x and the neighbour block B_i. $N_{px}(B_x)$ and $N_{py}(B_x)$ are the current block horizontal and vertical dimensions in pixels respectively. Two scenarios are shown in Figure 4. For prediction of motion vector corresponding to block B_X; motion vectors corresponding to the grey shade blocks are used. For these cases $w_1 = w_2 = w_5 = 0.5$ and $w_3 = w_4 = 0.25$.

For the motion structure higher layers, motion vector value of the parent node is also used to predict motion vectors of its corresponding descendant nodes. Child nodes of a specific parent node are all nodes in the sub tree for which the parent node is the sub tree root node. This type of prediction is used along with previously described prediction from neighbouring blocks. Differential data is sent to arithmetic encoder. For modelling, the histogram of previously occurred symbols is used.

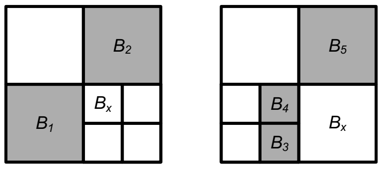


Figure 4 Examples of blocks used in prediction.

The binary data types *mv_ref_diff_row* and *mv_ref_diff_col* are used to signal if a row or column component of a motion vector refinement bit has been correctly predicted or not. Prediction is performed from the same blocks as for *mv_diff_row* and *mv_diff_col* data types. The value of the bit predicted from neighbouring blocks is calculated as a nearest integer rounded sum of weighted bit values in the neighbouring blocks. Weighting factors are defined as in (1). Also, the values of the ancestor nodes from previous layer of the motion structure are used for prediction in the higher structure levels. For the *mv_ref_diff_row* the modeller uses two contexts determined by following rules:

$$context\_no = \begin{cases} 0, \text{if } mv\_diff\_row = 0 \\ 1, \text{otherwise} \end{cases}.$$

As probability estimator within each context histogram of previously occurred symbols is used. For *mv_ref_diff_col* data type, context is determined in the same way but considering *mv_diff_col* value instead of *mv_diff_row*.

4. EXPERIMENTAL RESULTS

The multi-component scalable motion and proposed entropy coding scheme have been integrated in [6]. The tests were performed on two sequences: "Basket" (4CIF, 30fps) and "Mobile" (CIF, 30 fps). Each encoded sequence contained 3 refinement layers and 4 motion structure layers. Layers of the motion structure were selected in such way that layer 0, layer 1 and layer 2 contained 10%, 20% and 30% nodes of the full motion structure nodes respectively. Motion block size of 8x8 to 64x64 and 1/8 pixel precision were used.

Two tests have been performed. First test measures the compression efficiency on each refinement layer and motion map data with and without employed contexts proposed in the previous section. Results are shown in Table 2. Compression ratio (*CR*) is defined as

$$CR = \left(1 - \frac{out\_bits}{in\_bits}\right) \cdot 100\% , \qquad (2)$$

where *out_bits* and *in_bits* are the numbers of compressed bits and uncompressed bits respectively for each data shown in Table 2. It can be seen that the higher compression is obtained with context modelling.

Data being compressed		**Contexts employed [yes/no]**	***CR* [%]**	
			“Mobile”	**"Basket"**
Bitplane refinement layer (*n*)	**1**	no	1.65	1.97
		yes	1.81	3.04
	2	no	15.07	8.72
		yes	16.76	13.21
	3	no	35.03	17.40
		yes	39.92	27.47
Motion map		no	24.11	17.53
		yes	24.63	25.72

Table 2 Compression efficiency of the binary data types.

In the second test we have examined the influence of the removal of the refinement layers on the decoding quality. Results are shown in Table 3. The sequences were decoded with full texture in order to avoid the influence of rate-distortion method applied in the real scenarios and to show the influence of lossy received motion information on the decoding quality. However, the Mean Square Error (MSE) caused by loss in motion information can be almost linearly added to the MSE caused by loss of texture data.

In Table 3, *k* represents the number of layers that have been removed from motion data. Those layers can be refinement layers or motion structure layers. Results for removal of refinement layers of motion vector values ("bitplanes") are displayed for full motion structure. On the other hand, the results for removal of motion structure layers ("map"), all bitplanes of motion vector values have been decoded. Finally we have removed refinement layers from motion vectors when only coarsest layer of motion structure remains to obtain combined scalability. It can be seen that that removing the motion structure layers gives finer scalability than removing of refinement bits.

Sequence	**Type of scalability**	***PSNR$_Y$* after *k* removed layers [dB]**		
		***k* = 1**	***k* = 2**	***k* = 3**
"Mobile"	**bitplane**	38.76	30.04	23.25
	map	43.45	41.60	39.18
	combined	35.64	29.21	22.96
"Basket"	**bitplane**	40.30	32.02	25.32
	map	42.68	38.17	31.83
	combined	31.24	28.92	24.47

Table 3 PSNR after removing layers.

5. CONCLUSIONS

In this paper a scheme for multi-component scalable motion information coding has been presented. The scheme uses efficient combinations of inter and intra layer predictions as the base for modelling of probability states used by arithmetic coding. Additional coding gains have been achieved using the contexts in the probability estimation stage. Influence of the different motion data layers removal on the PSNR has been also studied. It has been shown that removing a large number of nodes in the motion structure performs better than removing large number of refinement bits. However, it is also shown that finer granularity of motion information can be achieved by applying both strategies.

REFERENCES

[1] Barbarien, J., Munteanu, A., Verdicchio, F., Andreopoulos, Y. Cornelis J., and Schelkens, P., "Scalable motion vector coding," *Proc. Int'l Conference on Image Processing (ICIP)*, pp. 1321-1324, Singapore, 2004.

[2] Marpe, D., Schwarz, H., and Wiegand, T., "Context-Based Adaptive Binary Arithmetic Coding in the H.264 / AVC Video Compression Standard", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 620-636, July 2003.

[3] Mrak, M., Sprljan, N., and Izquierdo, E., "Motion Estimation in Temporal Subbands for Quality Scalable Motion Coding", *Electronics Letters*, Vol. 41, Iss. 19, pp. 1050-1051, 15 September 2005.

[4] Mrak, M., Abhayaratne, G.C.K., and Izquierdo, E., "On the Influence of Motion Vector Precision Limiting in Scalable Video Coding," *Proc. 7th International Conference on Signal Processing*, ICSP 2004, Vol. 2, pp. 1143-1146, August 2004.

[5] Witten, I.H., Neal, R. and Cleary, J.G., “Arithmetic coding for data compression”, *Comm. ACM*, vol. 30, pp. 520-540, June 1987.

[6] Sprljan, N., Mrak, M., Abhayaratne, G.C.K., and Izquierdo, E., "A Scalable Coding Framework For Efficient Video Adaptation", *Proc. 6th Int'l Workshop on Image Analysis for Multimedia Interactive Services* (WIAMIS 2005), Montreux, Switzerland, April 2005.