Scalable Motion Vector Coding Based on CABAC for MC-EZBC

Yongjun Wu, *Member, IEEE*, and John W. Woods, *Fellow, IEEE*

***Abstract*—In the scalable video coder MC-EZBC, the motion-vector (MV) bitstream was not scalable in bitrate or resolution. In this short paper, we enhance MC-EZBC with a new scalable MV coder based on context adaptive binary arithmetic coding. Alphabet general partition of MV symbols is proposed to achieve accuracy or quality scalability of MVs. A selective layered structure is used to reduce the number of MVs transmitted when appropriate, mainly needed for resolution scalability. With these two additions, we have a layered temporal, SNR, and resolution scalability for the MV bitstream. Experimentally, we find that this gives significant visual and objective improvement for low bitrates and/or resolutions with only very slight PSNR loss and unnoticeable visual loss at high bitrates.**

***Index Terms*—Alphabet general partition (AGP), layered structure coding., motion compensated temporal filtering, scalable motion vector coding, scalable video coding.**

I. Introduction

STARTING from [1] and [2], highly scalable video coding based on a motion-compensated 3-D subband/wavelet transform (SWT) offers high coding efficiency with three types scalability: temporal, spatial, and bitrate/SNR, as was required in the *Call for proposals on scalable video coding technology* [15], such as MC-EZBC [3], LIMAT [4], in-band motion-compensated scalable video coder [5], and 3D-ESCOT [6]. However, typically, the motion vector data in a scalable SWT coder are optimized for high bitrate and full resolution and are coded losslessly without bitrate or resolution scalability [3]–[6]. Although temporal scalability of motion information comes easily by grouping the SWT data and motion information in different temporal levels separately [3], the motion information bitstream does not have resolution or bitrate/SNR scalability. This degrades the video performance at low bitrates and/or resolutions.

Recently, several techniques [7]–[11] were proposed to achieve motion vector scalability in SWT video coding. Layered structure for motion vector coding is proposed in [7] and [11]. However, the work in [7] codes some redundant motion information, and the results in [11] may not be suitable for median motion vector (MV) prediction and context adaptive binary arithmetic coding (CABAC). In [8], a quality-scalable MV coding technique is presented. In principle, this technique is the same as our alphabet general partition (AGP) of MVs first proposed in [10] (all of our MPEG input documents are available[1]) and presented below. The only difference is that in [8] a context modeling is also used for the coding of quantization errors, but this is not needed, as we will show. In [9], a wavelet transform-based quality scalable MV coding technique is proposed. However, as shown in [8], the quality scalable vector coding in [9] is not as efficient as prediction-based quality scalable coding due to the limited local correlation among the MVs. In [12], a multiple-layer structure was proposed for motion information representation. All motion layers are used for analysis by the encoder, while only motion layers are used for synthesis by the decoder.

In this short paper, we propose a selective layered structure and alphabet general partition (AGP) [14] coding of the MV symbols for MC-EZBC [3], [10], as shown in Fig. 1. In principle, our AGP in [10] is the same as the quality-scalable MV coder in [8] (which was independently proposed at about the same time). The selective layered structure coding of MVs achieves the same functionality as the redundant MV coding technique in [7], while avoiding possible large distortion from MV outliers and is achieved without redundant coding of MVs. Section II presents some background on MV prediction and an overview and needed modifications to CABAC. Section III contains the AGP of MV symbols. Section IV presents our selective layered MV structure. Then, the two methods are combined, and experimental results are presented in Section V. Conclusions are given in Section VI.

II. Background

Here, we first introduce MV prediction in MC-EZBC and then overview CABAC of MVs.

We have four block types in MC-EZBC: NORMAL, PBLOCK, BACKWARD, and directional IBLOCK [16], where PBLOCK and NORMAL blocks have MVs between the current frame and the next frame, BACKWARD blocks have MVs between the current frame and the previous frame, and IBLOCK does not have any MV. For the MV in a given block, if there is at least one MV among the three direct neighbor blocks with the same type (either NORMAL/PBLOCK or BACKWARD) as that in this current block, we predict it from its spatial neighbors. If there is no MV with the same type in these three neighbors, we predict the current block from the previous MV with the same type in quad-tree scan order. We consistently use this combined spatial and scan-order prediction in the remainder of this paper. The prediction residual is subsequently entropy coded.

In our MV coding application, each prediction residual is simply multiplied by 8 to convert it to an integer symbol, since

Manuscript received February 25, 2006; revised September 14, 2006 and November 30, 2006. This work was supported in part by the National Science Foundation under Grant CCF-0430837. This paper was recommended by Associate Editor X. Li.

Y. Wu is with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: yongjun.wu@gmail.com).

J. W. Woods is with Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA.

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

Digital Object Identifier 10.1109/TCSVT.2007.894043

[1][Online]. Available: http://www.cipr.rpi.edu/people/faculty/woods/html

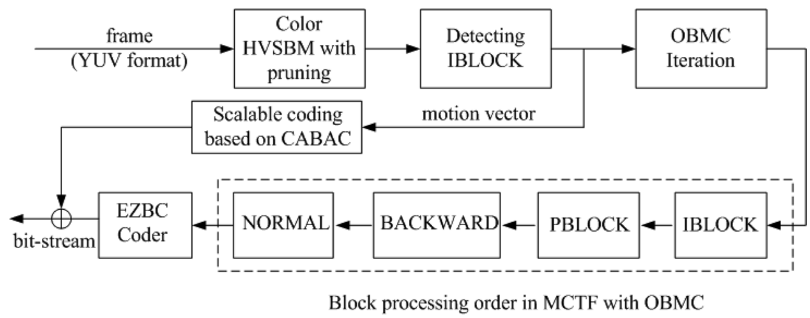

Fig. 1. Overall functional block diagram for the scalable video coder MC-EZBC.

the maximum pixel accuracy is 1/8. The context model involves a context template with up to two neighboring symbols in the scanned *past* of the current symbol. One is from the top neighbor, and the other is from the left neighbor, assuming a progressive raster. The context model is separate for x and y component residuals. Moreover, we code absolute values of residuals separately from their signs, with these signs coded in equal-probability mode [13] and different context models are used for different bins.

III. AGP of Motion Vectors

We first present an efficiency analysis for AGP in entropy coding of MVs [14] using statistical information from MC-EZBC. Then, we specify the implementation of AGP.

A. Efficiency Analysis for AGP of MVs

MV subpixel bits are quite random [17] due to both sensor and quantization noise, especially the quarter and eighth pixel bits. Hence, we expect that the low-order bits of the MVs can be coded directly without variable-length coding. The method to accomplish this is AGP.

For an i.i.d. source with M symbols, let the symbol i occur with probability p_i, then the first-order entropy for the source is $\mathcal{H} = \sum_{i=1}^{M} p_i \log_2(1/p_i)$. For any positive integer N, we can partition the source symbols into nonempty sets $\mathcal{S}_n$, where $n = 1, 2, \ldots, N$, and denote the number of elements in $\mathcal{S}_n$ by M_n [14]. If we code the set number and the index in the set separately, the overall coding rate is

$$\mathcal{H}_1 = \sum_{n=1}^{N} P_n \log_2 \frac{1}{P_n} + \sum_{n=1}^{N} \sum_{i \in \mathcal{S}_n} p_i \log_2 \left(\frac{P_n}{p_i} \right) \quad (1)$$

where $P_n \triangleq \sum_{i \in \mathcal{S}_n} p_i$ is the total probability for the set $\mathcal{S}_n$. The first sum in (1) is the rate to code the set, and the second item is for the rate to conditionally code the symbol index in a set. As is known [14], the overall rate in (1) is equal to the first-order source entropy. Hence, there is no efficiency loss when we separate the coding into two steps. We can simplify the coding of the symbol index within a set by coding all indexes in the set $\mathcal{S}_n$ with a fixed-length $\log_2 M_n$-word binary code. The simplified method has a total rate

TABLE I
STATISTICS OF THE ONE-EIGHTH-PIXEL OR LSB OF MVS AT VARIOUS TEMPORAL LEVELS

L	$N/P(s_0)$	$N/P(s_1)$
1	4752/0.49	4924/0.51
2	3193/0.50	3163/0.50
3	2196/0.52	2032/0.48
4	1499/0.52	1387/0.48
All	11640/0.50	11506/0.50

$$\mathcal{H}_2 = \sum_{n=1}^{N} P_n \log_2 \left(\frac{1}{P_n} \right) + \sum_{n=1}^{N} \sum_{i \in \mathcal{S}_n} p_i \log_2 M_n. \quad (2)$$

The difference between $\mathcal{H}_2$ and $\mathcal{H}_1$ is as follows:

$$\Delta\mathcal{H} = \sum_{n=1}^{N} \sum_{i \in \mathcal{S}_n} p_i \log_2 \left(\frac{M_n p_i}{P_n} \right). \quad (3)$$

The relative loss $\Delta\mathcal{H}/\mathcal{H}$ will be very small if the partition satisfies $p_i \approx P_n/M_n, \forall i \in \mathcal{S}_n$, i.e., the distribution inside the set $\mathcal{S}_n$ is approximately uniform [14].

Test results of this hypothesis for the MV data are given in Tables I and II, obtained from the first two GOPs of the popular test clip *Bus*. Here N/P(s_i) indicates "number/probability" of the subsymbol $i/8$, and L means temporal level. Table I shows that the LSB subsymbols or one-eighth pixel bits are totally random at all temporal levels. However, if we partition motion vectors with three LSB subsymbols, the distribution is highly skewed from a uniform distribution until temporal level 4, as shown in Table II. Summarizing, we find that there should be little efficiency loss for AGP of one-eighth pixel accurate MVs with up to two LSB subsymbols.

B. AGP of MV Symbols in MC-EZBC

MCTF can benefit from 1/8 pixel accuracy [3]. Hence, we decompose the estimated MV m into three subsymbols

$$m = m_1 + m_2 + m_3. \quad (4)$$

TABLE II
STATISTICS OF THE THREE LSBs OF MVs AT VARIOUS TEMPORAL LEVELS

L	$N/P(s_0)$	$N/P(s_1)$	$N/P(s_2)$	$N/P(s_3)$	$N/P(s_4)$	$N/P(s_5)$	$N/P(s_6)$	$N/P(s_7)$
1	2232/0.23	3516/0.36	1825/0.19	828/0.09	422/0.04	310/0.03	273/0.03	270/0.03
2	915/0.14	1771/0.28	1590/0.25	892/0.14	467/0.07	296/0.05	221/0.03	204/0.03
3	463/0.11	642/0.15	809/0.19	843/0.20	658/0.16	358/0.08	266/0.06	189/0.04
4	348/0.12	335/0.12	389/0.13	379/0.13	398/0.14	396/0.14	364/0.13	277/0.10
All	3958/0.17	6264/0.27	4613/0.20	2942/0.13	1945/0.08	1360/0.06	1124/0.05	940/0.04

where m_1 is the major symbol with 1/2 pixel accuracy, and m_2 and m_3 are the subsymbol for quarter and eighth pixel accuracy, respectively. We predict the major symbol with the scheme in Section II and code the prediction residual with our MV CABAC, but code the subsymbols with fixed-length codewords. Then, we have four parts for our MV bitstream: major symbol m_1, subsymbols m_2, m_3, and an additional sign bit for subsymbols whose original symbol m is in the range $[-0.375, +0.375]$ which comes together with the first significant bitplane in the subsymbol. As an alternative, we can partition the MVs into more subsymbols; however, the resulting increase in scalability comes with some loss in efficiency.

At high bitrates and full resolution, we transmit all of the MV parts. However, at low bitrates, we can discard some subsymbols and sign parts for transmission and thus allow more SWT data (sometimes called *texture*) for the same total bitrate. At the decoder, we then reconstruct only lossy MVs, but the resulting average total performance is improved. Also, at low resolution, since in MC-EZBC the MVs will be scaled down anyway, we do not need as much MV accuracy.

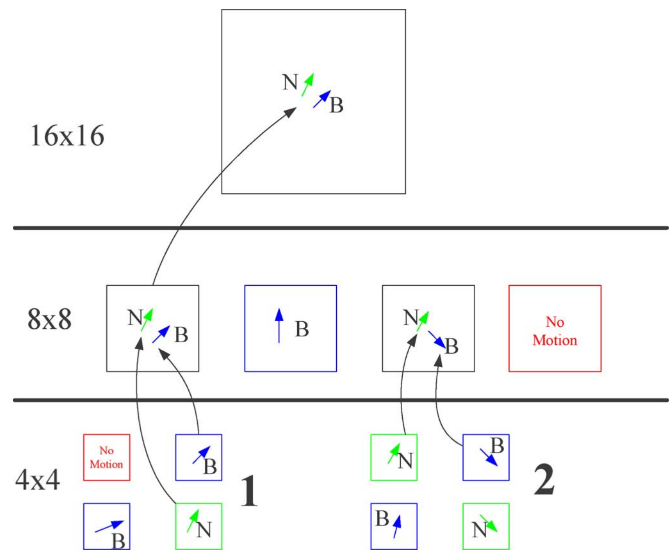

Fig. 2. Example of MV subsampling in our selective layered structure. The 16 $\times$ 16 block is split into Set 1—four 4 $\times$ 4 blocks and Set 2—four 4 $\times$ 4 and two 8 $\times$ 8 blocks. The red blocks are IBLOCKs (no MVs), blue arrows (labeled with "N") are NORMAL MVs, and the green arrows (labeled with "B") are BACKWARD MVs.

IV. SELECTIVE LAYERED STRUCTURE FOR MV CODING

We first present a subsampling scheme for MVs coming from hierarchical motion estimation, with selective merging rules to reduce unnecessarily large distortions in complex motion areas. The subsampling scheme with selective merging in Section IV-A forms the base layer and enhancement layer for MVs that we call a layered structure for MV. In Section IV-B, we present the prediction and coding schemes for these layers.

A. MV Subsampling Scheme With Selective Merging

For each spatial level downwards, the block size is halved, e.g., 16 $\times$ 16 blocks become 8 $\times$ 8 blocks. Obviously, after one or two spatial levels down, we do not need the same number of MV blocks anymore. Moreover, the MVs themselves are also scaled down by two after each spatial level downwards. If two adjacent MVs have a difference less than $\mathcal{T}_s$ pixels at full resolution, then at half resolution their difference will be less than $\mathcal{T}_s/2$.

Assume that four grouped blocks (children) in the quadtree structure shown in Fig. 2 have similar MVs. Then, we can replace them by one representative after one or two spatial levels down. For simplicity, the first MV in the quadtree scan order is chosen as the *representative* for the four MVs in the child blocks. Since the NORMAL MVs are between the current and next frame, while a BACKWARD MV is between the current and previous frame, we need to reserve two representatives for the four children, one for NORMAL and the other for BACKWARD MVs.

Fig. 2 shows an example for the representative selection, i.e., MV subsampling from 4 $\times$ 4 down to 16 $\times$ 16 blocks. During motion estimation, the shown 16 $\times$ 16 block was split into Set 1 and Set 2. So, first, we need to select the NORMAL and BACKWARD representatives from the four 4 $\times$ 4 child blocks and store them for the parent 8 $\times$ 8 block, as shown in *Set 1* and *Set 2* in the figure, where the red blocks are IBLOCKs, the blue blocks are either NORMALs and PBLOCKs (in which NORMAL MVs are located), and the green blocks are BACKWARD blocks. Select the first NORMAL and BACKWARD MVs again from the four 8 $\times$ 8 children and store them in the 16 $\times$ 16 parent block. This finishes the subsampling scheme from 4 $\times$ 4 to 16 $\times$ 16. Repeating the subsampling scheme once again for 16 $\times$ 16 blocks, we can subsample the MVs from 16 $\times$ 16 to 32 $\times$ 32.

The assumption that four grouped blocks have similar MVs may be invalid sometimes, causing significant distortion in the reconstructed frames when the MVs in the four grouped blocks are replaced by their representatives. Hence, we control this subsampling scheme with some selective merging rules as follows.

1) If MVs in an $S \times S$ block have differences $\Delta_x \triangleq |mv_{xi} - mv_{xs}| \leq \mathcal{T}_s$ and $\Delta_y \triangleq |mv_{yi} - mv_{ys}| \leq \mathcal{T}_s$, we replace all the MVs in that $S \times S$ block by up to two representatives, where $mv_{x/yi}$ are the x/y components of the MVs in the

$S \times S$ block, and $mv_{x/ys}$ is the x/y component of the subsampled MV.

2) Otherwise, if the difference among the MVs in each $(S/2) \times (S/2)$ block, $\Delta_x \leq \beta \mathcal{T}_s$ and $\Delta_y \leq \beta \mathcal{T}_s$, we replace all of the MVs in the $(S/2) \times (S/2)$ block by up to two representatives.

3) If neither 1) nor 2) can be satisfied, we code all of the MVs without subsampling.

We thus obtain the base and enhancement layers for MVs. At the decoder, if we only receive the base layer, we will replace all of the NORMAL MVs in an $S \times S$ or $(S/2) \times (S/2)$ block by the NORMAL representative. For example, as shown in Fig. 2, we will replace all five BACKWARD (blue) MVs in the 16×16 macroblock by the BACKWARD (blue) representative stored in the top 16×16 block, and all three NORMAL (green) MVs by the NORMAL (green) one stored in the 16×16 block.

Generally, rule 1) works well in the slow moving background (green blocks). In those areas, MVs have small differences and are similar to one another, but rule 2) works better inside moving objects (yellow blocks). Other blocks may be on object boundaries or other irregular motion areas, or their block sizes are already very large. Generally, we select $S = 32$, $\beta = 1.5$, and $\mathcal{T}_s = 3 \sim 6$ pixels.

B. MV Prediction in Selective Layered Structure

In the selective layered structure, for the MVs in the base layer, if there is a NORMAL representative in a block, we regard the entire macro-block MV as the NORMAL representative MV, although actually its four child blocks may have different NORMAL MVs, as shown in Fig. 2. Then, we can predict the NORMAL MVs in the base layer as usual or use it to predict other neighbor MVs as in the nonlayered structure. We predict BACKWARD MVs in the base layer in the same way.

For MVs in the enhancement layer, since we already have their MV representatives from the base layer, we predict the current MV from the three direct neighbors in the enhanced MV field. This will improve efficiency because the enhanced MV field is the final MV field, and that makes prediction most efficient. Since we choose the first NORMAL and BACKWARD MVs in the quadtree scan order as the representatives, when we code the enhancement layer, we can skip coding the first NORMAL and BACKWARD MVs in the four child blocks. Thus, in our scheme, the total number of coded MVs is the same as in nonlayered MV coding. Moreover, although the first NORMAL and BACKWARD MVs are not coded in the enhancement layers, they are predicted again from the enhanced MV field, and their prediction residuals will be used as context models for coding the following MVs via CABAC.

V. Combined MV Structure and Results

Sections III and IV presented two new methods for MV coding. Although there is some small overhead for each method, they can make the MV bitstream quite scalable with respect to both SNR and resolution. AGP of MV symbols is mainly for bitrate/SNR scalability, while the selective layered structure is mainly for resolution scalability. The combination of these two methods comes naturally. Then, we have several layers for the MV bitstream, with each layer divided into several parts, as seen in Fig. 3. Also an empirical bit allocation model can be applied in [20, Secs. 5.4.3, 5.5.4, and 5.6].

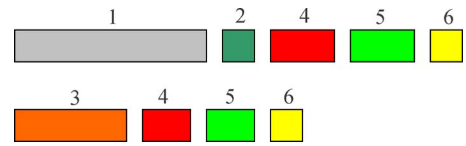

Fig. 3. Combined scalable MV structure: gray (1), sea green (2), and orange (3) parts are for major symbols in base layer, indication for grouping of blocks and enhancement layer for major symbol, respectively. Red (4), green (5), and yellow (6) parts are for quarter pixel, eighth pixel, and additional sign bits, respectively.

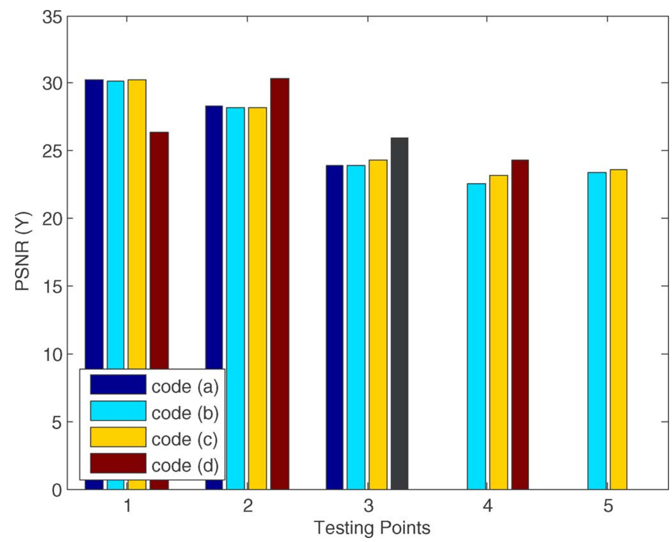

Fig. 4. PSNR (dB) comparison for full *Bus* clip. Coder (a) is without scalable MV coding, coder (b) results were presented to *MPEG Call* [10], coder (c) uses both AGP and selective layered structure, and coder (d) does not employ scalable MV coding and uses a big lambda ($\lambda = 48$ instead of general setting $\lambda = 24$). Note that, for coder (d), the chroma PSNRs are comparable to those of (a)–(c) at testing point 1. Testing points, 1: 512 Kbps_30fps_CIF, 2: 256 Kbps_15 fps_CIF, 3: 128 Kbps_15 fps_CIF, 4: 64 Kbps_15 fps_QCIF, 5: 48 Kbps_7.5 fps_QCIF.

In Figs. 4 and 5, the reference frames for low-resolution and/or low-frame-rate videos are decoder-side reference frames and are the same for all the coders. Coders (a)–(d) are defined in the figure captions, with all of the testing points specified in [15]. We can note that the combination of AGP and the selective layered structure is preferred. Scalable MV coding with AGP and the selective layered structure significantly improves performance at low bitrates and/or low resolution with only a very slight loss at high bitrates. As a reference, in Fig. 4, we also list the results with a large λ, i.e., $\lambda = 48$ instead of the general setting $\lambda = 24$. We found that although a big λ in rate-distortion optimization can reduce the bits spent on MV and thus more bits for residual data at low/medium rates, it may hurt at high bitrate, similar to the observation in [12]. All of the coded/decoded results, along with their references, are available for download[2] in the common .yuv format.

[2][Online]. Available: http://www.cipr.rpi.edu/ftp_pub/personal/wuy2/smv_paper/

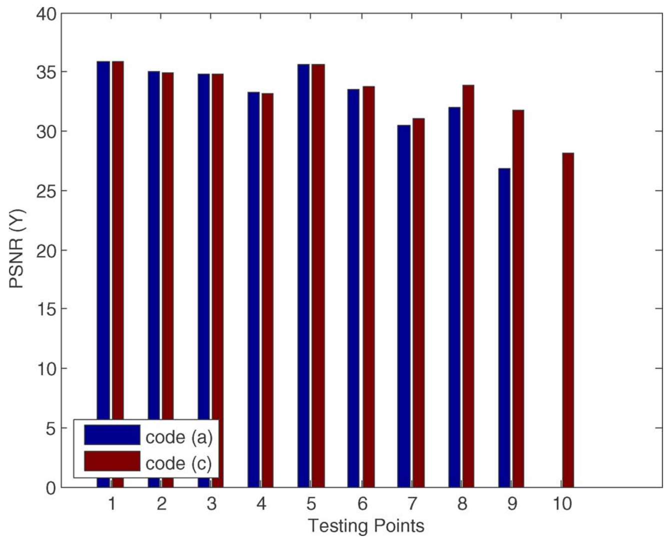

Fig. 5. PSNR (dB) comparison for full *City* clip. Coder (a) is without scalable motion vector coding and coder (c) uses both AGP and selective layered structure. Testing points, 1: 2048 Kbps_60 fps_4CIF, 2: 1536 Kbps_60 fps_4 CIF, 3: 1024 Kbps_30 fps_4 CIF, 4: 738 Kbps_30 fps_4CIF, 5: 512 Kbps_30 fps_CIF, 6: 384 Kbps_30 fps_CIF, 7: 256 Kbps_30 fps_CIF, 8: 128 Kbps_15 fps_QCIF, 9: 96 Kbps_15 fps_QCIF, 10: 64 Kbps_15 fps_QCIF.

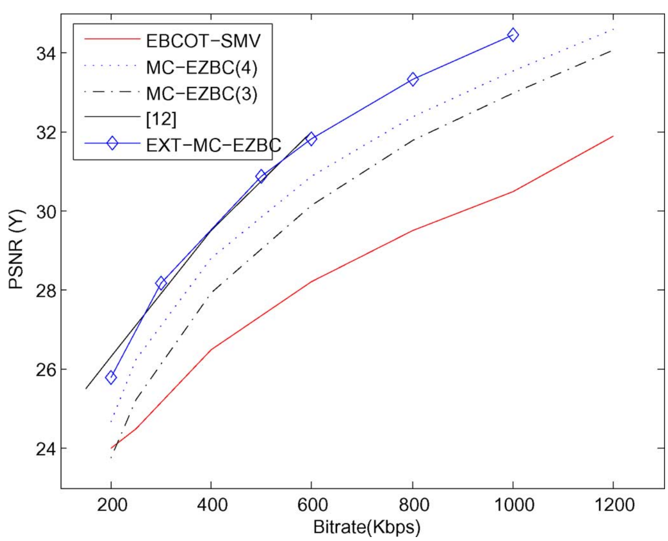

Fig. 6. PSNR (Y) comparison of full *Bus* at full resolution, full frame rate, and various bitrates for MC-EZBC with three temporal levels' decomposition *MC-EZBC(3)*, MC-EZBC with four temporal levels' decomposition *MC-EZBC(4)*, extended Aachen MC-EZBC *EXT-MC-EZBC*, the scalable video coder *EBCOT-SMV* from [9], and the results from [12].

Fig. 6 shows MC-EZBC(3) and MC-EZBC(4) with three and four levels' temporal analysis, respectively, and EBCOT-SMV with three levels' temporal analysis [9]. Although the platforms for scalable MV coding techniques are very different, and the PSNR difference may be due to other components inside the video coders, we list the full PSNR comparisons for the Y component as a reference on Fig. 6. The PSNR points for EBCOT-SMV are deduced from [9, Fig. 7]. We may notice that, at high to medium bitrates, MC-EZBC has significantly better performance than EBCOT-SMV. At very low bitrates, EBCOT-SMV with 12 motion quality layers down to 5 Kbps has better performance than MC-EZBC with our maximum six motion quality layers. In Fig. 6, the results from [12, Fig. 6] are also listed for reference purposes. Again, the components in the whole scalable video coding system are very different between MC-EZBC and the system from MSRA [12]. The PSNR comparison in Fig. 6 only serves as a reference. As a more fair comparison with the results from [12], we also listed the results from our extended version of the Aachen MC-EZBC named *EXT-MC-EZBC* [21] with directional I-block, OBMC, and scalable motion vector coding as specified in [20, ch. 6]. The performance of *EXT-MC-EZBC* is similar to that from [12], confirming a 1-dB gain due to the adaptive MCTF and advanced motion estimator.

VI. Conclusion

We presented two methods for scalable MV coding, i.e., AGP of motion vectors and selective layered structure coding, with entropy coding based on MV CABAC. These two new methods enable SNR and resolution scalability of the MV bitstream in our MCTF-based scalable video coder. With the temporal MV scalability that already existed in MC-EZBC, we now have temporal, SNR, and resolution scalability of the MV bitstream. Experimental results show the effectiveness of this approach. Our method also compares favorably with a previous quality scalable MV coding technique [9].

References

[1] J. R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Process.*, vol. 3, no. 9, pp. 559–571, Sep. 1994.
[2] S. J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Trans. Image Process.*, vol. 3, no. 2, pp. 155–167, Feb. 1999.
[3] P. Chen and J. W. Woods, "Bi-directional MC-EZBC with lifting implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1183–1194, Oct. 2004.
[4] A. Secker and D. S. Taubman, "Lifting-based invertible motion adaptive transform (LIMAT)," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1530–1542, Dec. 2003.
[5] V. Bottreau, M. Benetiere, and B. Pesquet-Popescu, "A fully scalable 3-D subband video codec," in *Proc. ICIP*, Oct. 2001, pp. 1017–1020.
[6] J. Xu, Z. Xiong, S. Li, and Y. Zhang, "Three-dimensional embedded subband coding with optimized truncation (3-D ESCOT)," *Appl. Computat. Harmonic Anal.*, vol. 10, pp. 290–315, 2001.
[7] S. S. Tsai and H. Hang, "Motion information scalability for MC-EZBC," *Signal Process.: Image Commun.*, vol. 19, pp. 675–684, 2004, Elsevier.
[8] J. Barbarien, A. Munteanu, F. Verdicchio, Y. Andreopoulos, J. Cornelis, and P. Schelkens, "Scalable motion vector coding," *Electron. Lett.*, vol. 40, no. 1, pp. 932–934, Jul. 2004.
[9] A. Secker and D. Taubman, "Highly scalable video compression with scalable motion coding," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1029–1041, Aug. 2004.
[10] Y. Wu, A. Golwelkar, and J. W. Woods, MC-EZBC Video Proposal from Rensselaer Polytech. Inst. JTC1/SC29/WG11/M10596/S15, ISO MPEG, Mar. 2004. Munich, Germany.
[11] Z. Hu, M. van der Schaar, and B. Pesquet-Popescu, "Scalable Motion Vector Coding for MC-EZBC," in *Proc. EUSIPCO*, Vienna, Austria, Sep. 2004, pp. 657–660.
[12] R. Xiong, J. Xu, F. Wu, S. Li, and Y. Zhang, "Layered motion estimation and coding for fully scalable 3-D wavelet video coding," in *Proc. ICIP*, Singapore, Oct. 2004, pp. 2271–2274.
[13] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.
[14] A. Said and W. A. Pearlman, "Low complexity waveform coding via alphabet and sample-set partitioning," in *Proc. VCIP, SPIE*, San Jose, CA, Feb. 1997, pp. 25–37.

[15] *Call for Proposals on Scalable Video Coding Technology*, JTC1/SC29/WG11/N6193, ISO MPEG, Dec. 2003, Waikoloa, HI.
[16] Y. Wu and J. W. Woods, "Directional spatial I-BLOCK for the MC-EZBC video coder," in *Proc. ICASSP*, May 2004, vol. 3, pp. 129–132.
[17] T. Wedi and H. G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 577–586, Jul. 2003.
[18] Y. Wu and J. W. Woods, Recent Improvements in MC-EZBC Video Coder JTC1/SC29/WG11/M10396, ISO MPEG, Dec. 2003, HI.
[19] ——, "Scalable motion vector coding for MC-EZBC," in *Proc. SPIE Electron. Imaging*, San Jose, CA, Jan. 2005, pp. 335–346.
[20] Y. Wu, "Fully scalable subband/wavelet video coding system," Ph.D. dissertation, ECSE Dept. Rensselaer Polytech. Inst., , Troy, NY, Aug. 2005.
[21] T. Rusert, K. Hanke, and M. Wien, "Optimization for locally adaptive MCTF based on 5/3 filtering," presented at the Picture Coding Symp., San Francisco, CA, 2004.