Enhanced MC-EZBC Scalable Video Coder

Yongjun Wu, Member, IEEE, Konstantin Hanke, Thomas Rusert, and John W. Woods, Fellow, IEEE

Abstract—In this paper, we provide some recent extensions to the scalable subband/wavelet video coder MC-EZBC. The enhanced MC-EZBC employs an adaptive motion-compensated temporal filter (MCTF) framework. Directional I-BLOCKS and overlapped block motion compensation (OBMC) further improve MCTF efficiency. A scalable motion vector coder based on CABAC is shown to improve the overall performance at low bitrates/resolution. Frequency rolloff is incorporated to reduce spatial aliasing at low resolution without PSNR loss at full resolution. Experimental results show that the new features significantly improve the performance of MC-EZBC. We provide several comparisons to other recent coders.

Index Terms—Directional I-BLOCK, frequency rolloff, motion-compensated temporal filter (MCTF), overlapped block motion compensation (OBMC), scalable motion-vector coding, scalable video coding, spatial high-pass transition filtering.

I. INTRODUCTION

CALABLE video coding encodes the video only once at the highest frame rate, bitrate, and spatial resolution but enables transmission and/or decoding from partial streams at arbitrary lower bitrates, frame rates, and spatial resolutions. This is a simple and sufficient solution for video streaming over current heterogeneous networks. During bitstream transmission, scalability enables both multicast and unicast streaming applications with minimal additional processing and low-complexity decoding [1]. Recently, a number of techniques have been proposed to achieve fully scalable video coding. Many of them are based on the application of the subband/wavelet transform (SWT) in both spatial and temporal directions. Starting from [2] and [3], highly scalable video coding based on a motion-compensated 3-D SWT offers high coding efficiency with three types of scalability: temporal, spatial, and bitrate or SNR, such as MC-EZBC [4], LIMAT [5], in-band motion-compensated scalable video coders [6], [7], and 3D-ESCOT [8].

In this paper, we report on the combined work on MC-EZBC from the research group at RWTH Aachen University [9], [10] and the CIPR lab at Rensselaer Polytechnic University [12], which significantly updates MC-EZBC and improves its performance as compared to its earlier versions [19]. In Section II, an adaptive MCTF framework with adaptive switching between LeGall and Tabatabai (LGT) 5/3 and Haar 2/2 SWT is briefly introduced. In Section III, we present enhanced features for this adaptive MCTF framework, including directional I-BLOCKS [13], overlapped-block motion compensation (OBMC) [14], a scalable motion-vector coder [17] based on CABAC, and frequency rolloff [18]. In Section IV, we compare the enhanced MC-EZBC with its earlier versions, with JSVM v. 1.0,1 and with two other SWT coders.

II. ADAPTIVE MCTF FRAMEWORK

Here, we briefly review the main components in the adaptive MCTF framework based on LGT 5/3 and Haar 2/2 SWT, including block modes, frame modes, and motion-estimation/motion decision [9], [10].

A. Block Modes in Motion Estimation

We use block-based motion compensation with a quadtree partitioning structure and associated motion-vector scan order according to [3]. Table I shows the details about different block modes as explained in the following.

- In the prediction step, LGT 5/3 filters are used for block modes 1, 4, 7, and 8, while Haar 2/2 filters are used for block modes 2, 3, 5, and 6.
- In the update step, the motion vector in a block with modes 1, 2, 3, 7, or 8 may be inverted. In blocks with modes 4, 5, or 6, the update step is skipped.
- The number of motion vectors is equal to the number of reference frames for modes 1, 2, 3, 4, 5, and 6. In mode 7, the forward motion vector has the same magnitude as the backward one but opposite sign. In mode 8, the motion vector is perfectly predicted according to median prediction from neighbor motion vectors.

B. Frame Modes in Adaptive MCTF

Each frame $B_k$ in Fig. 1 is assigned to one out of four frame modes in adaptive MCTF, as shown in Table II. The chosen frame mode results in the minimum total cost for the whole frame, with a specific Lagrangian factor $\lambda$, which is typically chosen to optimize performance at medium to high bitrates, full

1JSVM Software from jvtuser@garcon.ient.rwth-aachen with permission. Also, the document JVT-N022.zip on the website is for this software. [Online]. Available: ftp://standards.polycom.com/2005_01_HongKong/
frame rate, and full resolution. Scene change information is reflected in the choice of the frame mode, which propagates to the lower temporal levels. The frame mode is initialized to bidirectional at the beginning of the MCTF process. In some frame modes, not all block modes are allowed.

C. Mode Decision in Motion Estimation

Motion estimation block-size varies from 64 × 64 down to 4 × 4. An original or low temporal frame is assigned an initial frame mode, according to the scene-change propagation. Motion estimation is performed between frame \(B_t\) and two neighbor frames \(A_{t-1}\) and \(A_{t+1}\), as shown in Fig. 1, based on luminance data. Only the block modes as shown in Table II are tested. A cost \(C_i(X)\) is generated for block \(X\) in frame \(B_t\), which includes a sum of absolute differences (SAD) between that block and the reference block \(X_{ref}\), plus a weighted rate term \(\lambda R(X, M V(X))\), where \(\lambda\) is a Lagrange multiplier factor that may vary with temporal level, and \(R(X, M V(X))\) is a bitrate term representing the expected number of bits to be spent on coding the motion vectors \(M V_t(X)\) and/or \(M V_r(X)\), where \(M V_t(X)\) and \(M V_r(X)\) are “left” and “right” motion vectors. The block mode and motion vector with minimum cost \(C_i(X)\) is chosen. To improve the update step and avoid introducing artifacts into low temporal frames, the results from motion estimation are classified to determine whether the MCTF update step is omitted, according to the variance of the residual blocks. For details, please see [9], [10], and [12, Sec. 6.1.3].

D. Adaptive MCTF Framework

Fig. 1 shows the prediction and update steps for the analysis stage of the adaptive MCTF. Assuming one motion-vector field \(M V_t\) between frames \(A_{t-1}\) and \(B_t\) and another \(M V_r\) between frames \(A_{t+1}\) and \(B_t\), the prediction step of the high temporal frame \(H_t\) is then given by

\[
H_t = \alpha_1 M[A_{t-1}, M V_t] + \alpha_2 B_t + \alpha_3 M[A_{t+1}, M V_r]
\]

while the update step is given by

\[
L_{t-1} = \beta_1 M^{-1}[H_{t-2}, M V_{t-2}] + \beta_2 A_{t-1} + \beta_3 M^{-1}[H_{t}, M V_{t+1}]
\]

where the warping operator \(M\) represents motion compensation and \(M^{-1}\) its inverse. The filter coefficients \(\alpha_i\) and \(\beta_i\) are given in Table III. For the generation of an inverse motion field, motion vectors associated with blocks with mode 4–6 are ignored in the update step. In the case of pixels with multiple connections, the first motion vector in the motion-vector scan order is used. During inverse motion compensation, unconnected regions can occur in the low temporal frame. MCTF, including the analysis and synthesis process, is implemented via a sliding window approach [9].

III. NEW FEATURES FOR MC-EZBC

Here, we present the new features for MC-EZBC in the adaptive MCTF framework.

A. Frequency Rolloff

We incorporated spatial frequency rolloff into the adaptive MCTF framework. The idea of frequency rolloff is to make extracted low resolution images/videos coming from SWT have similar visual quality as those from MPEG4 low-pass filtering via bit-plane shift in extractor and decoder-side weighting. With this idea, we can obtain smoother low-resolution images/videos with great reduction in aliasing but no performance loss at full resolution. Since frequency rolloff is mainly implemented in the EZBC coder, the idea fits into the adaptive framework with little difference from that in the Rensselaer bidirectional MC-EZBC [18], [19]. Without frequency rolloff, the produced low-resolution videos may have unnecessary sharp edges or aliasing, and the PSNR with respect to its decoder-side reference frames would drop about 2–3 dB.

### TABLE I

<table>
<thead>
<tr>
<th>Mode</th>
<th>Block mode</th>
<th>Reference frame</th>
<th>Update step</th>
<th>Motion vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bi-connected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>left-connected</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>right-connected</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>bi-predicted</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>left-predicted</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>right-predicted</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>parallel</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>8</td>
<td>spatial-direct</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Fig. 1. MCTF framework with adaptive switching based on 5/3 and Haar filters [9]. \(H_t\) is temporally aligned with \(B_t\), and \(L_t\) is temporally aligned with \(A_t\). MC and IMC are motion warping and inverse operators, respectively. The numbers along the branches are weighting factors.

Authorized licensed use limited to: Rensselaer Polytechnic Institute. Downloaded on February 11, 2010 at 15:05 from IEEE Xplore. Restrictions apply.
B. Directional IBLOCK

We know that, in some regions of image frames, such as occluded areas or areas with complex motion, that motion compensation is not efficient for prediction [13]. In this case, we can employ spatial prediction for the current block from already coded pixels in neighboring blocks. Directional IBLOCK does this prediction in a spatially adaptive way. It can improve coding efficiency at low to medium bitrates without performance loss at high bitrates. Without directional IBLOCK, at medium to low bitrates, the PSNR will drop around 0.5 dB.

C. OBMC Extended

Since SHTF [11] is a similar tool to OBMC, when using OBMC, we disable SHTF. The same set of weighting matrices in [20] is used and the shrinking scheme is adopted. Moreover, we also incorporated a directional IBLOCK into the OBMC framework similarly to [20]. The prediction step in MCTF with either SHTF or OBMC can be expressed as

\[ H_t = \alpha_2 B_t + \sum_{k \in S} h(k)(\alpha_1 M[A_{t-1}, MV_{t,k}] + \alpha_3 M[A_{t+1}, MV_{t,k}]). \]

(2)

In SHTF, \( h(k) \) is the set of filtering coefficients, and \( S \) is the support area for SHTF, typically 7x7 for motion compensation of one pixel in \( B_t \). For OBMC, the \( h(k) \) are the predefined window coefficients, and \( S \) is the set of effective direct neighbors. Typically, no more than three neighbors are used. With the shrinking scheme in OBMC, we can avoid oversmoothing at object boundaries. Experimentally, SHTF and OBMC achieve similar performance improvement. SHTF and OBMC do not only reduce blocky artifacts and provide smoother video, but achieve about 1-dB gain consistently for all bitrates.

D. Scalable Motion-Vector Coding

The scalable motion-vector coding uses both alphabet general partition (AGP) and layered structure coding to scale both motion-vector accuracy and their total number [17]. As shown in [17], the efficiency of our scalable motion vector coding is better than [15] and similar to [16], AGP works in the same way. Selective layered-structure coding for motion vectors works less well than in the Haar MC-EZBC [17], although it still achieves number scalability for motion vector coding. There are two reasons: first motion estimation in the adaptive MCTF [9] is done in an R-D optimization framework, so the motion vector prediction and coding is quite compact. The loss of coding efficiency is more significant within the adaptive MCTF framework than in Haar MC-EZBC. Second, there are more advanced block modes in the motion estimator of adaptive MCTF. Sometimes we may destroy the initial prediction structure and lose much of the coding efficiency for those motion vectors with layered structure coding. Switching off scalable motion-vector coding leads to a much worse performance, especially at low bitrates, where we lose up to 2-3 dB coding performance.

E. Block-Mode Coding Versus Temporal Level

We found that a separate Huffman code should be used for coding the block modes at temporal levels 3 and higher. As the temporal level increases, the areas with simple motion modes decrease, and the areas with complex motion increase. Thus, we can favor the simple motion modes in temporal levels 0, 1, and 2 but favor complex motion modes in temporal level 3 or higher. Please refer to [12, Sec. 6.25] for detailed experiments and statistics. This adaptive block-mode coding achieves about 0.1-0.2 dB gain at low bitrates.

IV. NEW RESULTS WITH ADAPTIVE MCTF FRAMEWORK AND ENHANCED FEATURES

Here, the PSNRs are calculated with respect to their own decoder-side reference frames for SWT-based coders and MPEG4 low-pass filtered frames for JSVM. Refer to [12, Sec. 6.3] for a full set of results and refer to [21] for visual comparisons.

First, we compare the enhanced MC-EZBC to the existing bidirectional MC-ECBC from Rensselaer [19] and RWTH MC-EZBC with original adaptive MCTF framework [9], [10] from Aachen. We list only the results for Bus and City. Coder 1 is Rensselaer MC-EZBC, which uses Haar filters for MCTF with bidirectional motion estimation. Coder 2 is RWTH MC-EZBC, which has the MCTF framework based on adaptive switching between Haar 2/2 and LGT 5/3 filters. Coder 3 is the Aachen/Rensselaer enhanced MC-EZBC with all the advanced tools. We set the \( \lambda \) in the motion estimator exactly the same for coders 2 and 3, which results in almost the same motion fields for these two. We set the \( \lambda \) in coder 1 to make the resulting motion bits similar to those in coder 3, not coder 2, since coder 1 also employs scalable motion vector coding. In our experiments, the difference in the amount of motion information produced by coders 1 and 3 never exceeded 5%. From Figs. 2 and 3, we can see that coders 2 and 3 have nearly the same performance at high bitrates. The slight PSNR loss comes

<table>
<thead>
<tr>
<th>Frame mode</th>
<th>Scene change</th>
<th>Permitted block modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-direction</td>
<td>Left</td>
<td>Right</td>
</tr>
<tr>
<td></td>
<td>1 2 3 4 5 6 7</td>
<td>8</td>
</tr>
<tr>
<td>Uni-left</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Uni-right</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Intra</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

**TABLE II**

FRAME MODES AND THEIR CORRESPONDING PERMITTED BLOCK MODES [9]

**TABLE III**

**COEFFICIENTS \( \alpha_i \) AND \( \beta_i \) DEPENDING ON THE MODE OF THE CURRENT BLOCK [9], \( \rho = \sqrt{(23/32)}, \phi = \sqrt{(3/2)}, \tau = \sqrt{(48/25)} \)

<table>
<thead>
<tr>
<th>Frame mode</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
<th>( \alpha_3 )</th>
<th>( \beta_1 )</th>
<th>( \beta_2 )</th>
<th>( \beta_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidirectional</td>
<td>-0.5 ( \rho )</td>
<td>-0.5 ( \rho )</td>
<td>0.25 ( \tau )</td>
<td>( \phi )</td>
<td>0.25 ( \tau )</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>-0.5 ( \rho )</td>
<td>0.25 ( \tau )</td>
<td>( \phi )</td>
<td>0.25 ( \tau )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Right</td>
<td>-0.5 ( \rho )</td>
<td>-0.5 ( \rho )</td>
<td>-0.5 ( \rho )</td>
<td>0.25 ( \tau )</td>
<td>( \phi )</td>
<td>0.5 ( \tau )</td>
</tr>
<tr>
<td>Unconnected</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
from the slight overhead of frequency rolloff and the small loss of coding efficiency from scalable motion vector coding. However, the enhanced MC-EZBC (coder 3) outperforms coder 2 significantly at low bitrates and/or low resolutions. Sometimes coder 3 is worse than coder 1 at low bitrate and/or low resolution, due to its heavier load of motion information. Note that the amount of motion information generated by coders 2 and 3 can be adjusted by choosing higher values of \( \lambda \) during mode decision, however this would lead to decreased performance at higher bitrates. With the adaptive MCTF framework coder 2 has better performance than coder 1 at high bitrates. However, at low bitrates and/or low resolution, usually coder 2 is worse, due to its heavier load of motion information.

Next, we present another set of comparisons between JSVM version 1.0 of the emerging scalable extension of H.264/AVC [22] and our enhanced MC-EZBC with all the advanced tools. Both JSVM and enhanced MC-EZBC provide a fully embedded representation in terms of temporal scalability. Regarding spatial scalability, JSVM employs a layered approach, and each spatial layer is optimized in order to meet specific application needs, while MC-EZBC is fully embedded in terms of spatial scalability. Also, JSVM defines a quality base layer where the FGS/MGS information is put on top. In contrast, MC-EZBC does not have any target testing points. For both coders, the motion information has to be optimized in order to achieve best coding performance for a given test scenario. In order to have a fair comparison, we have selected parameters optimized for the same set of operating points. Figs. 4 and 5 show performance comparisons for \( Y \) PSNR between JSVM 1.0 and enhanced MC-EZBC. We can see that enhanced MC-EZBC has comparable or even better performance than JSVM at high bitrates, however, it does not work as well as JSVM at low bitrates. But as the requirement of scalability increases, especially the case of greater resolution scalability in the SD sequences, the advantage of enhanced MC-EZBC becomes more significant. Some of the \( Y \) PSNR gains from enhanced MC-EZBC at lower resolutions and/or frame rates may come from the somewhat smoother MCTF reference frames. Thus, PSNRs in the testing points 1 thru 4 in Figs. 4 and 5 are not strictly comparable. All we can say is that, if the various reference frames are found equally acceptable, then these are the PSNRs that result. Any scalable video coder must produce it’s own reference frames for the lower frame-rate and/or spatial resolution outputs.

As a sample comparison among different SWT-based scalable video coders, Fig. 6 shows MC-EZBC(3 level) and MC-EZBC(4 level) with 3 and 4 levels’ temporal analysis of codec 1, respectively, and EBCOT-SMV with 3 levels of temporal analysis [15]. The PSNR points for EBCOT-SMV are deduced from Fig. 7 in [15]. 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 6 motion quality layers. In Fig. 6, the results from Fig. 6 in [16] are also listed for reference purposes, here labelled MSRA-Codec. Finally we list the results from our enhanced MC-EZBC, labeled as Enh MC-EZBC. The performance of Enh MC-EZBC is similar to that of the MSRA-Codec.

V. CONCLUSION AND FUTURE WORK

In this paper, we briefly presented an enhanced MC-EZBC video coder, which works under an adaptive MCTF framework along with recently developed advanced tools. Enhanced MC-EZBC retains its excellent performance at high bitrates,
and achieves significant improvement at low bitrates and low resolutions. We provided several sets of experimental comparisons, first among the various versions of MC-EZBC from the CIPR at Rensselaer and RWTH at Aachen. A second set of comparisons is given between JSVM version 1.0 and enhanced MC-EZBC. At high bitrates and full spatial resolution, enhanced MC-EZBC is comparable to JSVM version 1.0, but, at low bitrates and/or resolution, JSVM is better. Additionally, a sample PSNR comparison among different SWT-based scalable video codecs in the literature is presented. We should mention that, during the development of this paper, some more advanced tools were proposed for MC-EZBC, such as the decoder-side deblocking technique and operating point adaptation [23] from RWTH. These developments could offer further improvements.

REFERENCES


