A Low Complexity Motion Compensated Frame Interpolation Method

Jiefu Zhai[1*], Keman Yu[2], Jiang Li[2], and Shipeng Li[2]
[1]Institute of Computing, Chinese Academy of Sciences, Beijing, China, jeff@ict.ac.cn
[2]Internet Media Group, Microsoft Research Asia, Beijing, China, {kmyu; jiangli; spli}@microsoft.com

***Abstract*—In low bit-rate video communication, temporal sub-sampling is usually used due to limited available bandwidth. Motion compensated frame interpolation (MCFI) techniques are often employed in the decoder to restore the original frame rate and enhance the temporal quality. In this paper, we propose a low-complexity and high efficiency MCFI method. It first examines the motion vectors embedded in the bit-stream, then carries out overlapped block bi-directional motion estimation on those blocks whose embedded motion vectors are regarded as not accurate enough. Finally, it utilizes motion vector post-processing and overlapped block motion compensation to generate interpolated frames and further reduce blocking artifacts. Experimental results show that the proposed algorithm outperforms other methods in both PSNR and visual performance, while its complexity is also lower than other methods.**

I. INTRODUCTION

Low bit-rate video compression techniques play an important role in multi-user videoconferencing applications. When people deliver real-time video contents over the Internet, they often use temporal sub-sampling as well as frame size reduction in order to decrease the bandwidth requirement. To restore the original frame rate and improve the temporal quality, people usually reconstruct the skipped frames using interpolation in the decoder. How to accurately reconstruct the skipped frames using interpolation without introducing significant computational overhead is a key challenge.

Some simple interpolation methods such as frame repetition and frame averaging are often used because of their simplicity and low complexity. However, these simple methods might introduce annoying jerky or blurry artifacts to which human eyes are very sensitive. In recent years, there has been a special interest in motion compensated frame interpolation (MCFI), which takes motion into account to improve interpolation performance. Although MCFI techniques were formerly used to convert frame rate between PAL, NTSC and HDTV, many MCFI methods have been proposed for video streaming and conferencing applications [1]-[4].

General MCFI methods are based on the assumption that motion in video is smooth, continuous, and translational. This assumption is true for most video sequences, especially for sequences with relatively small motions. The general approach is to do motion estimation on the previous frame and the current frame, and then generate the interpolated frame by averaging the pixels in the previous frame and the current frame pointed by the half of obtained motion vectors. Consequently, most research efforts aim to improve the accuracy of estimated motion vectors. Because residual information of the skipped frames is unavailable, the more accurate the motion vectors are, the better the result that can be achieved.

Block-based motion estimation and pixel-wise motion estimation are the two main categories of motion estimation methods. In general, pixel-wise motion estimation can attain accurate motion fields, but needs a substantial amount of computation. Thus, it is often used in off-line MCFI rather than real-time processing. In contrast, block matching algorithms (BMA) can be efficiently implemented and provide good performance. Most MCFI methods are based on BMA. A comparison between pixel-wise and block based motion estimation for MCFI is discussed in [9].

It is well known in video coding that using a smaller block size might reduce the energy of residual images and consequently improve the coding efficiency. Although small block sizes, such as 8×8, 4×4 have been widely used in the state-of-art video coding standards such as H.264 [5], to our knowledge, by far most existing MCFI methods use a block size of 16×16 in motion estimation. The reason is that using small block sizes in general BMA methods usually results in motion vectors with minimized residual energy rather than true motion vectors. In our study, we found that using an appropriate motion estimation method together with a small block size works better than existing methods.

For real-time video applications, reducing the complexity of a computing process is significant. If the motion vectors embedded in a bit-stream can be utilized, the computations of motion estimation can be significantly reduced. However, some embedded motion vectors might bring serious artifacts if they are used directly. A motion vector post-processing method is proposed in [7] to smooth the motion field and improve the subjective perception, but some artifacts still remain.

In this paper, we propose a low-complexity and high efficiency MCFI method. We first examine the motion vectors embedded in the bit-stream, and then carry out overlapped block bi-directional motion estimation on those blocks whose embedded motion vector is regarded as not accurate enough. Finally, we utilize motion vector post-processing and overlapped block motion compensation to generate interpolated frames and further reduce blocking artifacts.

* The work presented in this paper was carried out in Microsoft Research Asia.

The rest of the paper is organized as follows. In Section II, we describe the proposed method in details. Experiment results are given in Section III. We conclude our work in Section IV.

II. The Proposed Method

The proposed method comprises several steps, as shown in Fig. 1. First, the motion vectors embedded in a bit-stream are classified into two groups: one group contains those motion vectors that are considered to represent true motion, thus could be directly used in interpolation; the other group contains "bad" motion vectors. Then, overlapped block bi-directional motion estimation (OBBME) is carried out on the second group. After that, a motion vector post-processing technique is used to smooth the motion field. Finally, overlapped block motion compensation (OBMC) is employed to generate the interpolated frame.

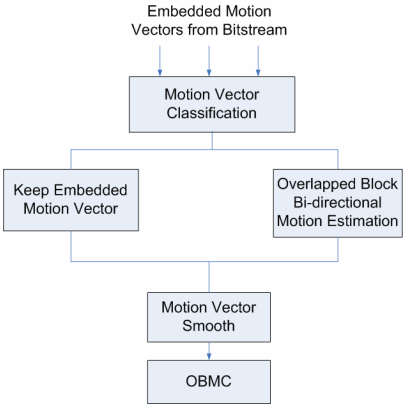

Figure 1. A block diagram of the proposed MCFI algorithm.

A. Motion Vector Classification

Generally, for most sequences, the majority of motion vectors embedded in the bit-stream are close to the true motion. However, some other embedded motion vectors may result in serious blocky artifacts if they are directly used in frame interpolation, because motion vectors embedded in the bit-stream are obtained for the purpose of minimizing rate-distortion cost.

In the video coding literature, the sum of absolute differences (SAD) is used to detect zero motion; while in the error concealment literature, people use the boundary absolute difference (BAD) to measure the accuracy of motion compensation [8]. Unfortunately, neither SAD nor BAD is efficient in picking out bad motion vectors. SAD often fails to point out true motion when local variation is small, while BAD often fails when local variation is large. In experiments, we found that combination of these two measurements can provide a better performance.

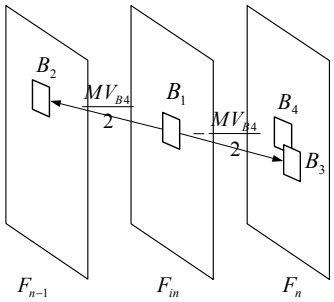

Figure 2. The judgement of the correctness of the motion vector.

As shown in Fig. 2, there are three frames in the discussion. They are the previous frame, the interpolated frame and the current frame. For an 8×8 block in the interpolated frame, e.g. block B_1, we firstly get the motion vector of the block at the same location in current frame (Block B_4 in Fig. 2). Then following the trajectory pointed by $MV_{B4}/2$, we can obtain two blocks, B_2 and B_3. If $MV_{B4}/2$ is close to the true motion of block B_1, the SAD between B_2 and B_3 is expected to be small, and likewise the BAD between B_2 and pixels surrounding B_3 is also small.

Assume that T_1, T_2, T_3 and T_4 are four thresholds and $T_3 > T_1$, $T_4 > T_2$. We determine whether a motion vector is good or bad as follows.

Calculate SAD and BAD
If SAD < T_1 or BAD < T_2
    MV is considered to be good
Else if SAD < T_3 and BAD < T_4
    MV is considered to be good
Else
    MV is considered to be bad

The percentages of blocks that need motion estimation are listed in Table I. We can see that only a small fraction of the blocks need motion estimation. The computational complexity is greatly reduced by motion vector classification.

TABLE I. Percentage of Blocks That Need Motion Estimation

Sequence	Percentage
Foreman	24.2%
Coastguard	18.5%
Carphone	27.1%
Mother	3.0%
Salesman	2.7%
Mobile	63.0%
Clair	2.0%

In the above discussion, we use embedded motion vectors at the 8×8 level. In some video standards such as H.264, block sizes vary from 16×16 to 4×4. A straightforward way to obtain a motion vector for each 8×8 block is "splitting and merging". For block sizes larger than 8×8 (such as 16×16, 8×16 and 16×8), each constituent 8×8 block may have the same motion vector as that of the original block. For any 8×8 block which has been split into several sub-blocks, we regard the motion vector of the 8×8 block as the average of the motion vectors of all its sub-blocks. In this way, the proposed MCFI scheme can be applied to a bit-stream that is coded by almost any video coding standard.

B. Overlapped Block Bi-directional Motion Estimation

As we have described before, embedded motion vectors do not always represent the true motion. For bad motion vectors we should perform true motion estimation for the corresponding blocks.

An 8×8 block size is used in our scheme instead of 16×16, which is widely used in other MCFI schemes. Using a smaller block size leads to a denser motion field and consequently brings two advantages: i) The neighboring motion vectors are more highly correlated thus the motion vector prediction will be more effective. ii) Better motion estimation especially at the boundary of moving object is obtained if the motion vector is accurate. Both advantages will be discussed more specifically below.

First, we examine the motion vectors of temporal and spatial neighboring blocks and select the one with minimal distortion as the

initial search point as done in [1] and [2]. Then, we perform motion search from the initial point, as illustrated in Fig. 3. Although a small block size leads to a denser motion field, it is also likely to obtain inaccurate motion vectors if common BMA methods are used. The reason is that when the block size becomes small, motion search tends to fall into a local minimum point.

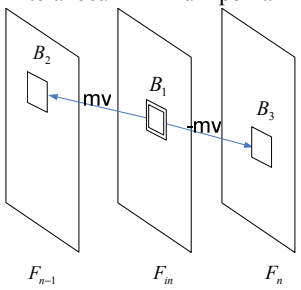

Figure 3. Overlapped block bi-directional motion estimation.

To overcome this problem, we use overlapped block motion estimation together with bi-directional motion estimation. For each 8×8 block in the interpolated frame, we firstly enlarge the block size to 12×12, and then use the enlarged block to do bi-directional motion estimation and give the computed motion vector to the 8×8 block. A 12×12 block size is used because it provides a good tradeoff between accuracy and complexity. If the enlarged block size exceeds 12×12, the computational complexity rises significantly without bringing obvious improvement to the accuracy of motion vectors.

As shown in Fig. 3, the aim of bi-directional motion estimation is to find the mv=(mv_x, mv_y) that minimizes:

$$D = \sum_{i=0}^{11}\sum_{j=0}^{11}\left(F_{n-1}(x_0 + mv_x + i, y_0 + mv_y + j) - F_n(x_0 - mv_x + i, y_0 - mv_y + j)\right) \quad (1)$$

where (x_0, y_0) is the coordinate of the top left point of the enlarged block in the interpolated frame. Since the motion vector is searched from the interpolated frame to the previous frame and the current frame, it does not introduce holes or overlapped areas, as described in [4]. Similar to general BMA, we can use fast motion estimation algorithms to speed up the search process.

C. *Motion Vector Post-processing and OBMC*

It is observed that there are still a few bad motion vectors which will bring annoying artifacts and degrade video quality significantly. Most artifacts originate from discontinuities in the motion field. So it is desirable to find a method to identify such motion vectors which break the continuity of the motion field. Fig. 4 shows an example of a single bad motion vector.

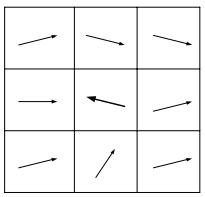

Figure 4. Motion vector discontinuity.

We calculate the variation of each motion vector and its neighboring motion vectors. If the variation exceeds a certain threshold, the motion vector is regarded as a single bad motion vector and then vector median filtering is applied. Vector median filtering [10] finds the one motion vector among the eight neighboring motion vectors that minimize:

$$\sum_{i=1}^{8}\left(\left|mv_x - mv_{xi}\right| + \left|mv_y - mv_{yi}\right|\right) \quad (2)$$

where (mv_{xi}, mv_{yi}) represents neighboring motion vectors.

After motion vector post-processing, OBMC [4] is finally applied to generate the interpolated frame, as illustrated in Fig. 5. Solid lines stand for original blocks and dashed lines stand for enlarged blocks of size 12×12 which overlap with neighboring blocks.

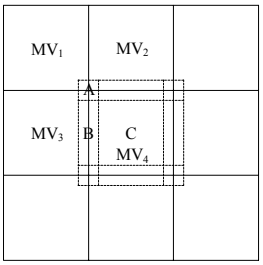

Figure 5. Overlapped block motion compensation

Suppose the top left four neighboring blocks have separate motion vectors (mv_{1x}, mv_{1y}), (mv_{2x}, mv_{2y}), (mv_{3x}, mv_{3y}) and (mv_{4x}, mv_{4y}). For a pixel $F_{in}(x, y)$ in region A that overlaps four blocks,

$$F_{in}(x,y) = (\sum_{i=1}^{4}(F_{n-1}(x + mv_{ix}, y + mv_{iy}) + F_n(x - mv_{ix}, y - mv_{iy}))/8 \quad (3)$$

For region B which overlaps two blocks

$$F_{in}(x,y) = (\sum_{i=3}^{4}(F_{n-1}(x + mv_{ix}, y + mv_{iy}) + F_n(x - mv_{ix}, y - mv_{iy}))/4 \quad (4)$$

For region C

$$F_{in}(x,y) = (F_{n-1}(x + mv_x, y + mv_y) + F_n(x - mv_x, y - mv_y))/2 \quad (5)$$

III. EXPERIMENTAL RESULTS

In this section, we present some experimental results to demonstrate the performance of the proposed algorithm. We implemented the methods described in [1], [2], and [3] and compare the PSNR results and objective perception with our method. In these experiments, raw sequences in QCIF format are encoded by the state-of-art H.264 reference software JM 8.0 [6] with baseline profile. The quantization parameter (QP) value is set to 25 and rate control is disabled. Odd frames are skipped by the encoder and they are interpolated by different MCFI methods respectively in the decoder. PSNR results are given in Table II. It can be seen that the proposed method outperforms other methods by 0.2~1.1 dB. For sequences that possess moderate motion, the proposed method obviously outperforms other methods. This is owed to using a small block size and OBBME. For sequences that possess small motion, the proposed method still slightly works better than other methods.

For subjective evaluation, four frames interpolated by different methods and their local details are shown in Fig. 6. Results show that our method provides better visual performance. In image (a), (b) and (c), obvious artifacts can be seen near the man's right eyebrow. However, image (d) that was generated by the proposed method does not suffer from such artifacts. This can be seen more clearly in image (e), (f), (g), and (h) that represent local details.

TABLE II. PSNR COMPARISON OF DIFFERENT METHODS

Sequence	Method[1]	Method[2]	Method[3]	Our method
Foreman	33.95	34.09	33.92	34.66
Coastguard	33.94	34.05	34.13	34.08
Carphone	33.35	33.42	33.20	33.63
Mother	37.75	37.79	37.87	38.01
Salesman	36.96	37.05	37.04	37.18
Mobile	29.97	30.46	30.24	31.38
Clair	41.38	41.42	41.45	41.65

The proposed method possesses relatively low complexity. As described before, the proposed method consists of three steps. The motion vector classification step performs only one SAD and BAD calculation for each block, which requires very little computational cost. In the OBBME step, although the block size in motion search is enlarged to 12×12, because only a fraction of blocks need motion estimation after motion vector classification, the total computational complexity is still lower than general BMA methods. The following motion vector post-processing and OBMC are also low-complexity calculations. We ran different methods on an Intel P3 600MHz machine and compare their execution time in Table III. The implementations are not optimized by SIMD instructions. Results show that our method is obviously faster than other methods.

TABLE III. PROCESSING TIME (IN SECONDS) COMPARISON

Sequence	Method[1]	Method[2]	Method[3]	Our method
Foreman	45.5	22.7	25.1	16.4
Coastguard	48.4	24.4	26.1	18.0
Carphone	44.0	22.4	24.4	17.1
Mother	38.1	20.0	21.2	8.0
Salesman	45.0	20.3	24.2	11.3
Mobile	43.6	23.4	24.3	23.2
Clair	37.6	20.3	22.1	8.8

IV. CONCLUSIONS

In this paper, we propose a low-complexity motion compensated frame interpolation method. It is composed of three steps. First, it examines the accuracy of the motion vectors embedded in the bit-stream. Second, it carries out overlapped block bi-directional motion estimation on those blocks whose embedded motion vector is regarded as not accurate enough. Finally, it utilizes motion vector post-processing and overlapped block motion compensation to generate interpolated frames and further reduce blocking artifacts. Experimental results show that the proposed algorithm not only outperforms other methods in both PSNR and visual performance, but also possesses lower complexity compared to other methods.

REFERENCES

[1] T. Chen, "Adaptive temporal interpolation using bidirectional motion estimation and compensation", IEEE International Conference of Image Processing 2002, pp.313-316.

[2] K. Hilman, H.-W. Park, and Y.-M. Kim, "Using motion compensated frame-rate conversion for the correction of 3:2 pulldown artifacts in video sequences," IEEE Trans. on CSVT, Vol. 10, No. 6, pp. 869-877, Sep. 2000.

[3] Al-Mualla, M.E.,"Motion field interpolation for frame rate conversion", IEEE International Symposium on Circuits and Systems 2003, pp. II 652 -655.

[4] B.-T. Choi, S.-H. Lee, and S.-J. Ko, "New frame rate up-conversion using bi-directional motion estimation," IEEE Trans. on Consumer Electronics, Aug. 2000, Vol. 46, No. 3, pp. 603-609.

[5] MPEG4-AVC/H.264 JVT document archives: ftp://ftp.imtc-files.org/jvtexperts

[6] JVT reference software version 8.0: http://bs.hhi.de/~suehring/tml/download/old_jm/

[7] G. Dane and T. Nguyen, "Motion vector processing for frame rate up conversion", IEEE International Conference on Acoustics, Speech, and Signal Processing, May, 2004, Montreal.

[8] W.M. Lam, A.R. Reibman, B. Liu, "Recovery of lost or erroneously received motion vectors", IEEE International Conference on Acoustics, Speech, and Signal Processing, April 1993, Minneapolis, pp. 417-420.

[9] C.W. Tang, O.C. Au, "Comparison between block-based and pixel-based temporal interpolation for video coding", IEEE Int. Sym. on Circuits & Systems, Vol. 4, pp.122-125, May 1998.

[10] J. Astola, P. Haavisto, Y. Neuvo, "Vector median Filters", Proceedings of the IEEE, April 1990, Vol. 78, Issue 4, pp. 678-689

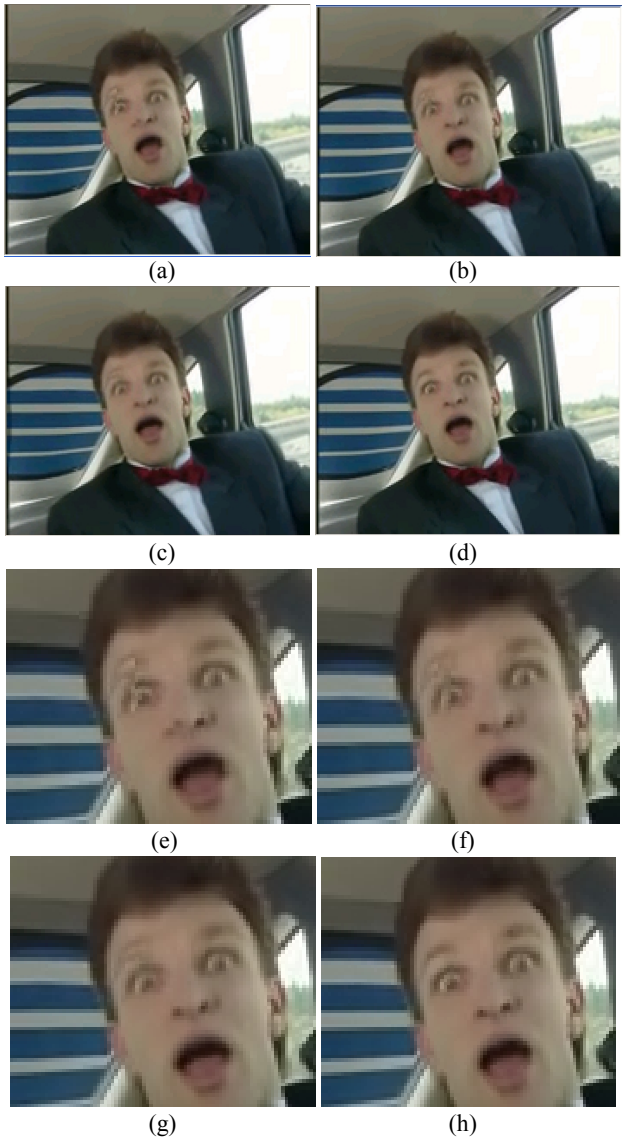

Figure 6. Visual comparison between different methods. Image (a), (b), (c) and (d) illustrate the interpolated results of frame 57 of the Carphone sequence using methods in [1], [2], [3] and ours respectively. Image (e), (f), (g), and (h) show the local details of (a), (b), (c), and (d) respectively.