EFFICIENT MEMORY MANAGEMENT CONTROL FOR H.264

Hyukjune Chung and Antonio Ortega

Integrated Media Systems Center,
Signal and Image Processing Institute
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, California, 90089-2564, U.S.A.

ABSTRACT

Multi-frame motion compensated prediction [1] can achieve high prediction gain with additional frame buffer memory requirement. For mobile-telephony applications which utilize mobile processor and relatively small memory space, reducing the memory requirement is very important. In this paper, we propose an efficient memory management control technique which reduces memory requirement for storing reference frames at the decoder side. Our proposed technique is a frame-based memory management scheme which discards the reference frames that are least important in terms of prediction gain performance. The optimal solution for this problem requires checking all the rate-distortion performance degradation for all the combination of discarded reference frames, and therefore we propose a greedy search that checks a subset of reference frame combinations.

1. INTRODUCTION

Multi-frame motion-compensated prediction has been introduced in the literature [1] as an approach to extend the motion search range by using multiple decoded frames as reference frames, instead of using just one decoded frame as in conventional motion compensation. Due to the potential performance gains, it is used in the most recent video compression standard JVT H.264 [2]. However, due to this multi-frame usage for motion compensation, the memory requirement to store reference frames at the decoder side increases significantly. High capacity, fast access memories are now available at increasingly lower cost. Reference frames, however, have to be fetched in expensive processor cache memory due to large number of accesses to the reference frames. For mobile telephony application, it is difficult to have enough cache memory for multi-frame motion-compensation. Therefore, reducing the memory requirement for the motion compensation is very important specifically for mobile applications.

For memory management control, which aims to reduce the memory requirement for motion compensation, there are block-based and frame-based techniques. Block-based memory management schemes form new references by using only relevant blocks for motion compensation. Frame-based memory management schemes discard least significant reference frames in order to form the smaller reference space.

In [3], a block-based efficient memory management control for the multi-frame motion-compensated prediction was proposed. In this paper, the author proposed a scheme that selects relevant blocks for motion compensation, and forms a memory space by using these relevant blocks. This block-based technique selects the relevant blocks based on a similarity measure by the same search algorithm at the encoder and decoder side. Therefore, this technique does not require signaling to the decoder side to select the chosen blocks for motion compensation, however, due to this relevant block search at the decoder side, this block-based scheme requires a special decoder to be used.

A block based memory management scheme with signaling is possible with no search at the decoder side. However, we need to signal the utilization of every blocks for each frame, and therefore, this signaling requires substantial overhead. However, signaling for the frame-based memory management scheme requires much lower overhead. For example, for H.264 video compression standard, utilization of each frame can be signaled by the memory management control operation (MMCO) [2].

In this paper, we propose a novel frame-based memory management scheme. Our proposed technique results in minimal performance degradation and requires no search at the decoder side. Therefore, our technique can be used with any standard decoder. Our goal is to discard the least important reference frames from the frame buffer, i.e., those resulting in the smallest prediction gain penalty. An optimal solution would require checking all the possible combinations of discarded reference frames to exactly determine the prediction gain penalties, and thus lead to significant search

0-7803-8554-3/04/$20.00

complexity and long encoding delays. For the proposed algorithm, we employ a greedy search which requires low additional complexity and short encoding delay.

The simplest memory management control scheme is the sliding-window technique which drops the oldest reference frames. Usually this technique works well because temporal correlation of video signals tends to decrease for older reference frames. However, there are a significant number of cases in which the oldest reference frames are important, e.g., sequences with repetitive motions and/or uncovered backgrounds by object motions. For these cases, dropping the oldest reference frames may degrade the prediction performance significantly.

This paper is organized as follows. In section 2, we describe the proposed algorithm. In section 3, we show experimental results. In section 4, we describe conclusion and future work.

2. EFFICIENT MEMORY MANAGEMENT CONTROL FOR MULTI-FRAME MOTION-COMPENSATED PREDICTION

Our goal in the proposed algorithm is to discard the least important reference frames from the frame buffer to form a set of smaller number of reference frames with the smallest prediction gain degradation. To find the optimal solution we would need to check all the combinations of discarded reference frames. Because for each combination the encoding result of a frame may be different, to find the optimal solution we need to encode a frame multiple times for each combination. Therefore, finding the optimal solution requires high computational complexity. Also, the optimal solution requires very long encoding delay because we need to check all the frames until the next intra refreshment. Due to the high computational complexity and the long encoding delay of the optimal solution, we propose to use a greedy search as a suboptimal solution. To describe our proposed algorithm, let us define some notations first:

- X_n : reconstructed frame with index n.
- Y_m : original frame with index m.
- ${B_v}^m(M) = \{X_{m-M}, \ldots, X_{m-1}\}$: virtual frame buffer of length M for the motion-estimated prediction of Y_m. Virtual frame buffer contains all possible reference frames for a given frame.
- ${B_r}^m(L)$: real frame buffer of length L for the motion estimated prediction of Y_m. Real frame buffer contains the reduced number of reference frames which is decided by a memory management control.
- $S_r(m) = \{X_{m-1}\} \cup ({B_r}^{m-1}(L) \cap {B_v}^m(M))$: the set of reference frames available for the motion estimated prediction for Y_m by our proposed technique.
- $D_i(X_n, Y_m)$: the distortion between i_{th} block of Y_m and the best match in X_n.
- $S_a(m)$: all the frames which can use at least one frame in $S_r(m)$ as a reference.

The proposed algorithm is composed of three stages, namely, data generation stage, frame-selection stage, and encoding stage. Our proposed algorithm discards a reference frame from $S_r(m)$, whenever $|S_r(m)| > L$, and does not refer to the reference frame for encoding future frames after it is discarded. Therefore, in our proposed algorithm, when we encode Y_m, we need to discard at most one reference frame from $S_r(m)$ because $|S_r(m)| \leq L + 1$. In our proposed algorithm, rather than basing our frame selection on all future frames, as required to obtain an optimal solution, we propose to discard the frame from $S_r(m)$ that is least important in terms of achieving efficient motion compensation of the frames in $S_a(m)$. In this way, we can achieve smaller encoding delay by reducing the amount of the look-ahead encoding.

2.1. Data generation stage

At the data generation stage, we perform the motion estimation for each frame $Y_k \in S_a(m)$. Note for some of the $Y_k \in S_a(m)$, the motion estimation has already been performed when making a decision on the previous frames, and the corresponding motion vectors and distortion values have been stored. Because the motion estimation process is the most complex processing for video encoding, we use these stored motion vectors and distortion values, and perform the motion estimation only for Y_k for which the motion estimation has not been performed. This may result in some coding inefficiency. However, in this way, we can perform the motion estimation just once for each frame Y_k.

If $|S_r(m)| \leq L$, at the data generation stage, we just perform normal encoding of Y_m without look-ahead encoding and frame-selection. Otherwise, we perform look-ahead encoding as follows. To perform the motion estimation for $Y_k \in S_a(m)$, we need to use encoded and reconstructed version of Y_j, $m \leq j < k$ as reference frames. However, we do not know the corresponding ${B_r}^j(L)$ of Y_j yet. Therefore, for an optimal solution, we need to generate multiple X_j corresponding to all the combinations of ${B_r}^i(L), m \leq i \leq j$ choices. However, checking all the combinations requires multiple encodings of one frame, and therefore, requires huge computational complexity. In our proposed algorithm, instead, we use all the available reconstructed frames in $B_v^k(M)$ as references to generate reconstructed frames X_k which will be used as reference frames for look-ahead encoding of the future frames $Y_l \in S_a(m), l > k$. X_k generated in this way will have the minimal effect on the motion estimation for the look-ahead encoding because

the reconstructed frame quality is about the same. Also, in this way, we can avoid long encoding delay and high computational complexity of the optimal solution.

After motion estimation, for each block, i, of frame $Y_k \in S_a(m)$, we store the motion vectors $\mathbf{mv}_i(X_j, Y_k)$ and the corresponding distortion values $D_i(X_j, Y_k)$ for each reference frame used, X_j, $j \in \{k-M, \ldots, k-1\}$. As the distortion D_i, in this paper, we use $SATD + \lambda_{MOTION} \cdot COST$, where *SATD* is the sum of absolute transformed differences, *COST* is the bit cost to encode a motion vector and a reference frame index, and λ_{MOTION} is the Lagrangian parameter used to incorporate rate constraint. We choose this metric as the distortion measure because it is used to find the best reference frame in H.264 reference software [4].

2.2. Frame-selection stage

In the second stage, using D_i obtained from the data generation stage, we compute $C(X_j, Y_k)$, the total distortion increase for each frame $Y_k \in S_a(m)$ when $X_j \in S_r(m)$ is discarded.

$$C(X_j, Y_k) = \sum_{i \in A} (D_i(X_{j_i}, Y_k) - D_i(X_j, Y_k)) \quad (1)$$

where A is the set of block indices for which $D_i(X_j, Y_k)$ is the minimum $\forall j$. In (1), X_{j_i} is the second best reference frame for i_{th} block of Y_k. By using $C(X_j, Y_k)$, we compute the total distortion increase, $\Gamma(X_j, m)$, of all $Y_k \in S_a(m)$ by discarding X_j from $S_r(m)$. $\Gamma(X_j, m)$ is computed as follows.

$$\Gamma(X_j, m) = \sum_{Y_k \in S_a(m)} C(X_j, Y_k) \quad (2)$$

Then, we discard X_{min} which minimizes $\Gamma(\cdot)$ from $S_r(m)$ to form ${B_r}^m(L)$ as follows.

$$X_{min} = arg \min_{X_j \in S_r(m)} \Gamma(X_j, m) \quad (3)$$

After discarding X_{min} from $S_r(m)$, we delete the stored motion vectors and the distortions corresponding to X_{min}, because stored motion vectors and distortions will be used for frame-selection of the future frames.

2.3. Encoding stage

By using ${B_r}^m(L)$ decided from the frame-selection stage, we encode current frame Y_m. We use the motion vectors collected at the data-gathering stage for the encoding. Therefore, we do not need to perform the motion estimation again for this stage. For the encoding, for each i_{th} block of Y_m, we choose the best motion vector $\mathbf{mv}_i^*$ as follows.

$$X_j^* = arg \min_{X_j \in {B_r}^m(L)} D_i(X_j, Y_m) \quad (4)$$
$$\mathbf{mv}_i^* = \mathbf{mv}_i(X_j^*, Y_m)$$

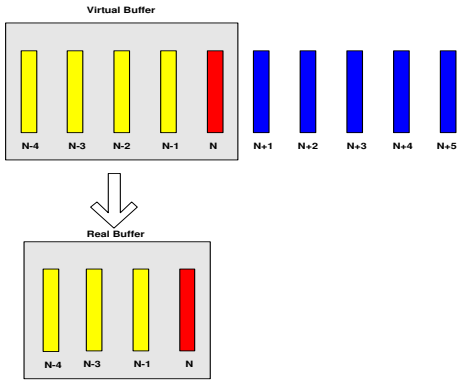

Fig. 1. Discarding a reference frame in the real frame buffer to form a set of references for the virtual frame buffer.

Using these chosen motion vectors, we encode Y_m, then add the X_m to the frame buffer to be used for encoding the future frames.

2.4. Example

Let us describe our proposed algorithm by a simple example in Figure 1. Assume that the length of the real frame buffer $L = 4$, and the length of the virtual frame buffer $M = 5$. We will encode frame Y_{N+1} by the proposed technique. Now we want to discard a reference frame from $S_r(N+1) = \{X_N, X_{N-1}, X_{N-2}, X_{N-3}, X_{N-4}\}$ as shown in Figure 1. First, at the data gathering stage, we perform the motion estimation for $S_a(N+1) = \{Y_{N+1}, \ldots, Y_{N+5}\}$, and store all the motion vectors $\mathbf{mv}_j(X_{N-i}, Y_{N+1})$ and the distortions $D_j(X_{N-i}, Y_{N+1})$. Then, at the frame-selection stage, we compute $\Gamma(X_{N-i}, N+1)$ as follows.

$$\Gamma(X_{N-i}, N+1) = \sum_{m=N+1}^{N-i+5} C(X_{N-i}, Y_m), \quad (5)$$
$$i \in \{0, 1, 2, 3, 4\}.$$

By comparing all $\Gamma(X_{N-i}, N+1)$, $\forall i \in \{0, 1, 2, 3, 4\}$, we discard the reference frame which provides the smallest $\Gamma(\cdot)$. Let us assume that $\Gamma(X_{N-2}, N+1)$ is smallest, then we discard X_{N-2} from $S_r(N+1)$. The real frame buffer is ${B_r}^{N+1}(4) = \{X_N, X_{N-1}, X_{N-3}, X_{N-4}\}$. Then we delete $D_j(X_{N-2}, Y_{N+1})$ and $\mathbf{mv}_j(X_{N-2}, Y_{N+1})$ from stored motion vectors and distortions.

3. EXPERIMENTS

In this section, we show the experimental results of our proposed memory management control technique. For the experiments, we implement the proposed algorithm into a H.264 reference software, JM8.1a [4] with the baseline profile. For the experiments, we use CIF sequences of length

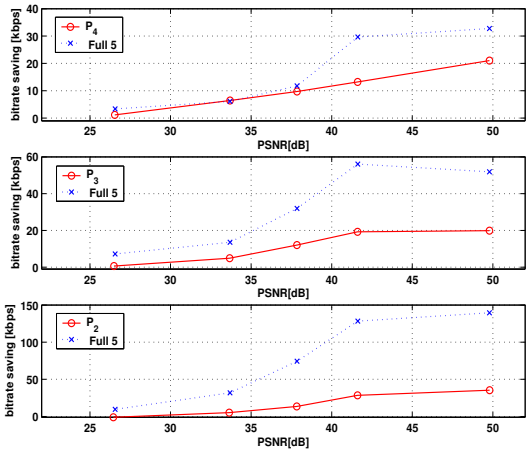

Fig. 2. Relative rate-distortion performance of the proposed technique with respect to the sliding-window technique, Top: SW4, middle: SW3, and bottom: SW2. **Full 5** uses virtual and real buffers of size 5 frames. The result is for 100 frames of CIF *Stefan* sequence.

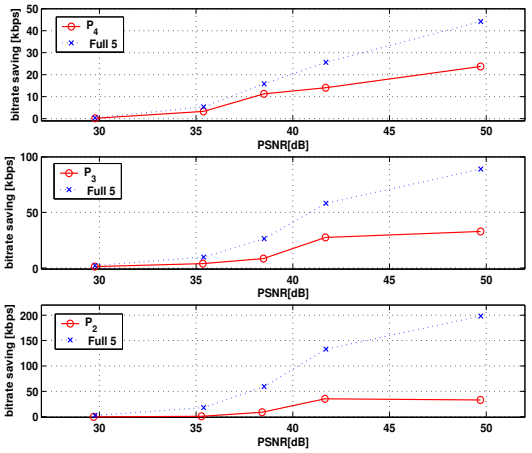

Fig. 3. Relative rate-distortion performance of the proposed technique with respect to the sliding-window technique, Top: SW4, middle: SW3, and bottom: SW2. **Full 5** uses virtual and real buffers of size 5 frames. The result is for 100 frames of CIF *Foreman* sequence.

100. Among 100 frames for each sequence, the first frame is intra-coded, and remaining frames are inter-coded. For motion estimation, we use the fast motion estimation implemented in the reference software and quarter-pel refinement, set the search parameter to 32, and use 16×16 macroblock partitioning. For the experiments, we use the virtual frame buffer whose size is 5, and vary the size of the real frame buffer. The proposed memory management scheme requires signaling the utilization of the frame buffer at the decoder side. For this, we use the memory management control operation (MMCO) of H.264[2].

The proposed algorithm is compared with the sliding-window memory management scheme which discards the oldest frames. In Figure 2 and 3, we show the relative rate-distortion (RD) performance of the proposed algorithm with respect to that of the sliding-window technique. In these figures, **SW4**, **SW3**, and **SW2** represents the sliding-window technique which discards the oldest 1, 2, and 3 reference frames among 5 reference frames, respectively. Also, P_4, P_3, and P_2 represents the proposed technique which utilizes 4, 3, and 2 frames, respectively. As one can see from these results, our proposed algorithm usually achieves better RD performance compared to the sliding-window technique. However, the relative gain of our proposed technique decreases in the lower bit-rate range. This is because as bit-rate decreases, more matches tend to be found from more recent reference frames.

4. CONCLUSION AND FUTURE WORK

In this paper, we propose an efficient memory management control scheme which can reduce the memory requirements for multi-reference motion-compensated prediction at the decoder side. The proposed technique employs a greedy search to discard the least important reference frames, and shows better prediction gain performance than the sliding-window memory management scheme which discards the oldest reference frames.The proposed technique is implemented with relatively low additional complexity at the encoder side, and requires no search at the decoder side. However, the proposed technique requires look-ahead encoding which introduces relatively small amount of encoding delay, and requires the signaling to let the decoder know when to discard reference frames.

For a video sequence where many matches come from the older reference frames, our proposed technique shows substantial gain performance enhancement.

5. REFERENCES

[1] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 70–84, Feb. 1999.

[2] T. Wiegand (ed.), "Joint working draft, version 2 (wd-2)," *Joint Video Team(JVT) of ISO/IECMPEG and ITU-T VCEG, JVT-B118r2*, Mar. 2002.

[3] R. Kutka, "Content-adaptive long-term prediction with reduced memory," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sep. 2003, vol. 3.

[4] *JVT Reference Software, version JM8.1a*, http://bs.hhi.de/ suehring/ tml/ download/.