Unidirectional Motion Compensated Temporal Interpolation

Chi-Wah Tang* and Oscar C. Au**

Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong.
Email: eetommy@ee.ust.hk*, eeau@ee.ust.hk**

Abstract

Common techniques such as frame repetition or linear interpolation for reconstructing skipped frames in temporally subsampled video sequence tend to introduce undesirable artifacts. A previously proposed technique, motion compensated temporal interpolation (MCTI) can interpolate video frames in the time domain with good image quality at the cost of high computational requirement and substantial memory requirement. In this paper, we propose an algorithm called Unidirectional Motion Compensated Temporal Interpolation (UMCTI) which has lower computational and memory requirement but achieving considerable better Peak-signal-to-noise ratio (PSNR). Several variants by incorporating stationary block detection and/or overlapping motion compensation are also studied.

I. INTRODUCTION

In video conferencing applications, the frame rate is usually low to achieve very low bit rate. The low frame rate tends to result in jerky motion. Temporal interpolation is a post-processing technique to synthesize one or more frames between two reconstructed frames at the decoder to produce smoother motion as the frame rate is increased during playback.

In a previous paper[1], we proposed a scheme called Motion Compensated Temporal Interpolation (MCTI) to perform block based motion compensated temporal interpolation which can yield synthesized frames with reasonably good visual quality. However, the computation requirement was very high. A fast algorithm called Fast Motion Compensated Temporal Interpolation (FMCTI) was also proposed to reduce the computation by two order of magnitude but the visual quality was compromised.

In this paper, we proposed a modified scheme called Unidirectional Motion Compensated Temporal Interpolation (UMCTI) which is effectively a simplification of MCTI with some modification. Essentially unidirectional motion estimation is used instead of bidirectional motion estimation. The interpolation procedure is also modified to be

similar to the PB frame operation in ITU-T H.263[4]. The proposed scheme is found to out-perform MCTI significantly in visual quality (in terms of PSNR) while at the same time achieving significantly lower computational complexity and memory requirement.

We also propose a fast version of UMCTI called Fast Unidirectional Motion Compensated Temporal Interpolation (FUMCTI) by performing stationary block detection.

Furthermore, we introduce overlapping [3-4], which is the same technique used in H.263, to UMCTI and FUMCTI to further improve the visual quality of the interpolated video sequence. We called these algorithms UMCTI with Overlapping (UMCTIO) and FUMCTI with Overlapping (FUMCTIO).

II. TEMPORAL INTERPOLATION ALGORITHMS

2.1 Review of MCTI

For any k , the goal of temporal interpolation is to generate a frame to be inserted between the $(k-1)^{\text{th}}$ and k^{th} received frames so that the effective frame rate can be increased and the object motions would appear to be smoother. In MCTI, the $(k-1)^{\text{th}}$ frame, the k^{th} frame and the inserted frame are divided into blocks of size $N \times N$. Forward and backward block-based motion estimation are performed to find the motion vector. Then the appropriate motion vectors are added to the candidate motion vector lists of the blocks in the inserted frame. For each block, the candidate motion vector with the largest overlapping area is chosen and the inserted frame is then generated by motion compensation.

Since we need to do both forward and backward block-based exhaustive search, the computational requirement is very high. Also, in order to store the candidate motion vector, the memory requirement is quite large.

In this paper, we propose an algorithm similar to MCTI but with fewer computational and memory requirement.

2.2 Unidirectional MCTI (UMCTI)

In the proposed algorithm, similar to MCTI, for any k , our goal is to generate a frame to be inserted between the $(k-1)^{th}$ and k^{th} received frames so that object motions would be smoother.

We divide the k^{th} frame and the inserted frame into blocks of size $N \times N$. Instead of performing bidirectional motion estimation, we only perform the forward motion estimation. For any block B_1 located at (x, y) in the k^{th} frame, we define a search area of size $(2W+1) \times (2W+1)$ in the $(k-1)^{th}$ frame and perform an exhaustive motion search using mean absolute difference (MAD) as the distortion measure. Let the best match be the block B_2 located at $(x+dx, y+dy)$ in the $(k-1)^{th}$ frame. Assuming linear translational motion, the block B_3 in the inserted frame corresponding to B_1 and B_2 is located at $(x+dx/2, y+dy/2)$. This implies a motion vector of $(dx/2, dy/2)$ for the block B_3 . This motion vector will be used to construct block B_4 at (x, y) in the inserted frame. Typically, block B_4 overlaps with B_3 . Let the portion of B_4 that overlaps with B_2 be B_4^o (shaded region in fig.1). Let the remaining portion of B_4 be B_4^{no} . Let f_{ins} , f_k and f_{k-1} be the inserted, k^{th} and $(k-1)^{th}$ frames. B_4^o is constructed by taking the average between the corresponding portion of B_1 and B_2 while B_4^{no} is constructed by unidirectional prediction.

$$f_{ins}(i, j) = [f_{k-1}(i+dx/2, j+dy/2) + f_k(i-dx/2, j-dy/2)]/2, \quad \text{for } (i, j) \text{ in } B_4^o.$$

$$f_{ins}(i, j) = f_{k-1}(i+dx/2, j+dy/2), \quad \text{for } (i, j) \text{ in } B_4^{no}$$

We repeat this for all the blocks of the frame.

Figure 1 Motion Estimation and Interpolation

Since only forward motion estimation is performed, we have saved half of the computation for motion estimation which occupied most of the computation. Also, computation is saved by eliminating the candidate lists and the search for the largest overlapping area to choose the best

motion vector. No memory is required to save the candidate list and hence the memory requirement is reduced.

2.3 Fast UMCTI (FUMCTI)

For applications like video-conferencing, most of the blocks of a frame are stationary. If we can identify those stationary blocks and avoid performing the motion estimation, much computation can be saved. In order to do that, we modify the motion estimation portion of UMCTI. Rather than performing exhaustive full motion search, we first calculate the MAD for zero motion. If the MAD is lower than a pre-defined threshold, we will consider the block as stationary and give it a zero motion vector. If the MAD is higher than the threshold, the regular exhaustive search of UMCTI is carried out.

2.4 UMCTI with Overlapping (UMCTIO) and FUMCTI with Overlapping (FUMCTIO)

One problem of UMCTI and FUMCTI is that the interpolated image is rather blocky. To reduce the blocking artifacts, we apply the overlapping algorithm which is similar to [3]. For each block, we interpolate using not only the corresponding motion vector, but also the motion vector of the neighboring blocks. We take the weighted average of the blocks generated by different motion vectors to reduce the blockiness. The corresponding methods are called UMCTI with Overlapping (UMCTIO) and FUMCTI with Overlapping (FUMCTIO).

III. SIMULATION RESULTS AND DISCUSSION

The "Miss America" and the "Salesman" sequences of size 352×288 are used in our simulation. For the sake of simplicity, the k^{th} generated frame is interpolated from the original $(k-1)^{th}$ and $(k+1)^{th}$ frames. In our simulation, we use block size(N) of 16 and search window size(W) of 15.

Figure 2 and 3 show the PSNR of the interpolated frames using MCTIO, UMCTI, FUMCTI, UMCTIO and FUMCTIO for the sequences "Miss America" and "Salesman", respectively. MCTI with overlapping (MCTIO) is used instead of MCTI in the simulation because MCTIO was reported in [3] to have even higher PSNR than MCTI and FMCTI. In Fig. 2, all the proposed algorithms are found to achieve higher PSNR than MCTIO most of the time. In particular, the PSNR of UMCTI and UMCTIO are consistently and significantly higher than that of MCTIO. In Fig. 3, the proposed algorithms are found to achieve higher PSNR than MCTIO most of the time, though in two brief periods (frames 8-18 and 52-57), MCTIO has comparable or even higher PSNR. The hands of the salesman were undergoing rapid nonlinear motion in these two periods resulting in relatively poor interpolated image

quality. It is observed that algorithms without overlapping do not perform as well in these two periods. This suggests that overlapping is useful in improving image quality in regions with rapid or non-linear motion. Apart from these two brief periods, all the proposed algorithms perform better than MCTIO.

Fig.2 PSNR plot of the "Miss America" sequence

Fig.3 PSNR plot of the "Salesman" sequence

The average PSNR and the computational requirement of different sequences using different approaches are summarized in Table 1.

Table 1: Summary of simulation results

	MissA	Salesman	Computation
UMCTI	38.63dB	35.18dB	1.95×10^8
FUMCTI	38.10dB	35.11dB	6.05×10^6
UMCTIO	38.88dB	35.42dB	1.95×10^8
FUMCTIO	38.17dB	35.31dB	6.05×10^6
MCTIO	37.84dB	35.13dB	3.90×10^8

The computational requirement in Table 1 reports only the amount of addition operations. An additional 2×10^5 shift operations are needed for overlapping in UMCTIO, FUMCTIO and MCTIO.

Since MCTIO has higher PSNR than MCTI and FMCTI, it can be concluded from Table 1 that the proposed algorithms using unidirectional motion estimation, namely, UMCTI, FUMCTI, UMCTIO, perform better than their counterparts MCTI, FMCTI and MCTIO using bidirectional motion estimation. The proposed algorithms can achieve higher PSNR at lower computational cost.

Some of the most challenging frames (lower PSNR) are shown in Figures 4 and 5. The visual quality of the proposed algorithms is found to be quite acceptable. However, there are still blocking artifacts at areas with fast motion, such as the eyes of Miss America and the moving hands of the salesman.

In the simulation, more than 97% and 94% of blocks of "MissA" and "Salesman", respectively, are declared as stationary blocks in fast algorithms and thus the computationally expensive motion estimation was avoided. In spite of the reduced computation, the visual quality of the fast algorithms is still very similar to the corresponding algorithms with full search. This suggests that the fast algorithm is very effective and worthwhile of doing.

Although additional computation is needed, overlapping is effective in reducing blocking artifacts, especially in regions with rapid, nonlinear motion.

IV. CONCLUSIONS

In this paper, we propose some modified temporal interpolation algorithms which use unidirectional motion estimation to simplify the computational requirement of a previously proposed algorithm MCTI. Simulation results suggest that UMCTI, FUMCTI, UMCTIO and FUMCTIO give considerably better PSNR than MCTI with lower computational and memory requirement.

V. ACKNOWLEDGEMENT

This work is supported in part by an RGC grant and by an HKTIIT grant.

VI. REFERENCE

- [1] C. K. Wong and O.C. Au, "Fast motion compensated temporal interpolation for video", *Proc. SPIE: Visual Communication and Image Processing*, Vol.4, pp.1108-1118, May 95.
- [2] C. K. Wong and O. C. Au, "Modified Motion Compensated Temporal Interpolation for Very-Low-Bit-Rate Video", *Proc. of IEEE ICASSP*, Vol. 4, pp.2327-2330, May 96.
- [3] C. K. Wong, O. C. Au and C. W. Tang, "Motion compensated temporal Interpolation with overlapping", *Proc. of IEEE ISCAS*, Vol-2, pp.608-611, May 96.
- [4] "Video Coding for Low Bitrate Communication", *Draft ITU-T Recommendation H.263*, Dec. 95.

Original frame 65 "Miss America" sequence

UMCTI - frame 65 "Miss America" sequence

UMCTIO - frame 65 "Miss America" sequence

FUMCTIO - frame 65 "Miss America" sequence

Figure 4

Original frame 56 "Salesman" sequence

UMCTI - frame 56 "Salesman" sequence

UMCTIO - frame 56 "Salesman" sequence

FUMCTIO - frame 56 "Salesman" sequence

Figure 5