Fast Motion Compensated Temporal Interpolation for Video

Chi-Kong Wong* and Oscar C. Au**

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

Abstract

Recently, MPEG-4 is being formed to study very-low-bit-rate (VLBR) video coding for applications in videotelephony. In this paper, we propose a possible postprocessing technique for VLBR coding. In videophone applications, temporal subsampling is a simple technique which can be combined with other compression schemes to achieve very large compression ratio, so as to satisfy the VLBR requirement. As a result, object motions tend to be jerky and disturbing to the human eyes. To smooth out object motions, we propose a postprocessing technique, motion compensated temporal interpolation (MCTI), to increase the instantaneous decoder frame rate. In MCTI, block-based exhaustive motion search is used to establish temporal association between two reconstructed frames. Both forward and backward searches are used to account for uncovered and newly covered areas properly. With MCTI, we show that one or more frames can be interpolated with acceptable visual quality. After showing the feasibility of MCTI, we propose a fast algorithm FMCTI with reduced computation requirement and negligible performance degradation.

Keywords: Temporal Interpolation, Temporal Subsampling, Motion Compensation, Pixel Decimation, Search Area Subsampling.

1. INTRODUCTION

In video conferencing applications, the bit rate of sending video sequences must be kept low due to the limited channel bandwidth. This is especially stringent in videotelephony applications, of which the target bit rate is considerably less than 64 kbit/s. For this reason, the Motion Picture Expert Group is forming the MPEG-4 standard committee to study very-low-bit-rate (VLBR) video compression techniques for bit rates as low as 10 or 20 kbit/s. Many approaches are being explored by researchers worldwide including model-based coding, fractal-based coding, segmentation-based coding, transform-based coding and postprocessing techniques. In this paper, we propose a postprocessing technique for VLBR coding which may be useful for MPEG-4.

To satisfy the VLBR requirement of the telephone channels, temporal subsampling is a simple technique which may be combined with other compression schemes such as CCITT H.261 to achieve very large compression ratio. In other cases, temporal subsampling can be a natural event. During a video conferencing session, large or frequent object motions usually result in larger-than-average bit rate which the channel may not be able to handle. This can cause a large backlog in the transmission buffer forcing the encoder to lower the instantaneous frame rate in order to clear the backlog. This is effectively instantaneous temporal subsampling. In all these cases, the skipped frames need to be reconstructed at the receiver. It is well known that simple frame reconstruction techniques such as frame repetition or linear interpolation introduce disturbing artifacts¹. Frame repetition generates jerky object motions because object movements are simply not considered and thus not accounted for. Linear interpolation by temporal filtering exhibits blurring in the moving areas because object motions are not considered and pixel values of different object regions are used in the interpolation resulting in the blurring of object region boundaries. Object motions must be compensated in order to remove these artifacts. In this paper, we propose to use motion compensated temporal interpolation (MCTI) to reconstruct the skipped frames with considerably fewer artifacts.

In MCTI, we compensate for the object motions by tracking the objects between adjacent received frames at the decoder. Knowing the trajectory of each object, we can place the object at the appropriate location in the interpolated frames. In MCTI,

use block-based motion estimation to establish blockwise association between each pair of adjacent received frames. The criterion for block matching is mean absolute difference (MAD). Both forward and backward motion estimation would be performed to account for the uncovered regions as well as the newly-covered regions. These regions can be found in only one but not both of the received frames. The collection of all the motion vectors defines a motion field which is used to set up a database for each interpolated frame. The database would then be used to construct the interpolated frames. The feasibility of the MCTI is verified by simulation using two videoconferencing test sequences.

One disadvantage of MCTI is the huge computation requirement in performing the forward and backward exhaustive motion estimation, making it impractical. With feasibility of MCTI verified, we propose fast motion compensated temporal interpolation (FMCTI) which is a modified MCTI with much reduced computation. The exhaustive motion search in MCTI is replaced by selective motion estimation. A simple scheme is used to determine if each block is stationary or not. Computation is reduced by performing forward motion estimation on the nonstationary blocks only and by performing backward motion estimation only on the blocks with poor forward motion estimation. Computation of the unidirectional motion estimation is further reduced by pixel decimation and search area subsampling. We show by simulation that FMCTI has similar visual quality as MCTI.

2. ALGORITHMS

Here we assume that the object motions are translational and are slow enough so that the motions are approximately linear over time among temporally adjacent frames. We also assume that there is no camera zooming which is usually the case in video-conferencing applications. In addition, we assume that the $N \times N$ blocks used are small compared with the object sizes. With these assumptions, we can use block motion estimation to establish object association between adjacent received frames. With the object trajectories tracked, the objects can be placed at the appropriate location in the interpolated frames

For simplicity, the criterion for block motion estimation is chosen to be mean absolute difference (MAD). For the present k^{th} frame, we denote the intensity value of the pixel with coordinates (i, j) by $f_k(i, j)$. We refer to a block of $N \times N$ pixels by the coordinate (x, y) of its upper left corner. The MAD between the block at (x, y) of the present received frame and the block at $(x + m, y + n)$ of the previous received frame can then be calculated as

$$MAD_{(x,y)}(m,n) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f_k(x+i, y+j) - f_{k-1}(x+m+i, y+n+j)|$$

The best matching block is defined as the block with minimum $MAD_{(x,y)}(m,n)$ among all locations (m,n) within the search area.

2.1 Motion-Compensated Temporal Interpolation (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 $(k-1)^{th}$ frame, the k^{th} frame and the inserted frame into blocks of size $N \times N$ such that a grid of blocks is formed in each frame. Firstly, we perform forward motion estimation. For a 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 with MAD being the distortion measure. If the best match is the block B_2 located at $(x + dx, y + dy)$ in the $(k-1)^{th}$

frame, then B_2 should also appear at $\left(x + \frac{dx}{2}, y + \frac{dy}{2}\right)$ in the inserted frame due to the assumed linear translational motion. However, the $N \times N$ blocks B_3 at $\left(x + \frac{dx}{2}, y + \frac{dy}{2}\right)$ in the inserted frame usually would not fit into the block grids exactly. Instead it would usually cover four $N \times N$ blocks C_1, C_2, C_3 and C_4 in the block grid. To solve this problem, we set up a list of motion vector candidates for each $N \times N$ block in the inserted frame. For the case discussed here, the forward motion vector $\left(\frac{dx}{2}, \frac{dy}{2}\right)$ is added to the candidate lists of all C_1, C_2, C_3 and C_4 together with the area of overlap between B_2 and corresponding blocks C_1, C_2, C_3 and C_4 . For any block in the inserted frame, the motion vector candidates with larger overlapping areas should be more reliable than those with smaller overlapping areas. The idea is shown in figure 1.

Next, we perform backward motion estimation. For each block B_1 located at (x, y) in the $(k-1)^{th}$ frame, a search area is defined in the k^{th} frame. The same exhaustive search is performed. If the block of the best match, B_2 , is at $(x + dx, y + dy)$ of the k^{th} frame, the backward motion vector $\left(\frac{dx}{2}, \frac{dy}{2}\right)$ is added to the candidate list of the blocks covered by the $N \times N$ block B_3 located at $\left(x + \frac{dx}{2}, y + \frac{dy}{2}\right)$ in the inserted frame.

Figure 1. Frame interpolation by forward motion estimation.

After the forward and backward motion estimation, each block C in the inserted frame will have a list of candidate motion vectors. To choose the best motion vector for block C located at (i, j) in the inserted frame, we pick the candidate with the maximum associated overlapping area. This is the motion vector whose associated block B_3 is the "closest" to the block C . If the best candidate (v_x, v_y) is a forward motion vector, block C is estimated by averaging the block located at $(i + v_x, j + v_y)$ in the $(k-1)^{th}$ frame and at $(i - v_x, j - v_y)$ in the k^{th} frame. We have assumed the residue R , an L-shaped lighter-colour block in figure 1, of the block B_3 associated with the block C_1 share the same motion information of the block B_3 , since the pixels inside the residue R are very close to the pixels inside the block B_3 . However, for the case in which the block at $(i + v_x, j + v_y)$ in the $(k-1)^{th}$ frame (or at $(i - v_x, j - v_y)$ in the k^{th} frame) goes beyond the image boundary, the block located at $(i - v_x, j - v_y)$ in the k^{th} frame (or at $(i + v_x, j + v_y)$ in the $(k-1)^{th}$ frame) is copied to block C . Similarly, if (v_x, v_y) is a backward motion vector, block C is estimated by averaging the blocks located at $(i + v_x, j + v_y)$ in the k^{th} frame and at $(i - v_x, j - v_y)$ in the $(k-1)^{th}$ frame. And if the block at $(i + v_x, j + v_y)$ in the k^{th} frame (or at $(i - v_x, j - v_y)$ in the $(k-1)^{th}$ frame) goes beyond the image boundary, the block located at $(i - v_x, j - v_y)$ in the k^{th} frame (or at $(i + v_x, j + v_y)$ in the $(k-1)^{th}$ frame) is copied to block C .

frame) is not completely within the image boundary, the block at $(i - v_x j - v_y)$ in the $(k-1)^{th}$ frame (or at $(i + v_x j + v_y)$ in the k^{th} frame) would be copied to block C . Once we have a collection of motion field (v_x, v_y) of the inserted frame, the frame can be interpolated.

2.2 Interpolation of M Successive Frames by MCTI

With the same argument stated above, we can insert M frames between the $(k-1)^{th}$ and k^{th} frames, each block of each of the M inserted frames will have a candidate list. Let the inserted frames be numbered, such that the 1^{st} one and M^{th} one are next to the $(k-1)^{th}$ and k^{th} frames respectively. The forward and backward motion estimation between the $(k-1)^{th}$ frame and the k^{th} frame are performed as described in section 2.1. For forward motion estimation, if the block B_1 located at (x, y) in the k^{th} frame is best matched to the block B_2 located at $(x + dx, y + dy)$ in the $(k-1)^{th}$ frame, the block B_2 is mapped to the block B_3 located at $\left(x + \frac{i}{M+1}dx, y + \frac{i}{M+1}dy\right)$ in the i^{th} inserted frame. The forward motion vector $\left(\frac{M-i+1}{M+1}dx, \frac{M-i+1}{M+1}dy\right)$ is added to the candidate lists of the four blocks covered by the block B_3 in the i^{th} inserted frame. Similarly, for the backward motion estimation, if the block B_1 located at (x, y) in the $(k-1)^{th}$ frame is best matched to the block B_2 located at $(x + dx, y + dy)$ in the k^{th} frame, the block B_2 is mapped to the block B_3 located at $\left(x + \frac{M-i+1}{M+1}dx, y + \frac{M-i+1}{M+1}dy\right)$ in the i^{th} inserted frame. The backward motion vector $\left(\frac{i}{M+1}dx, \frac{i}{M+1}dy\right)$ is added to the candidate lists of the four blocks covered by the block B_3 in the i^{th} inserted frame. The procedure to find the best candidate for each block inserted frame is the same as section 2.1.

2.3 Fast Motion Compensated Temporal Interpolation (FMCTI)

The motion search used in the algorithm described in section 2.1 is exhaustive full search which has a huge computation requirement. For interpolating a frame of CIF format using MCTI, the number of additions needed is around 4.4×10^8 . Therefore it is not suitable for real time applications. In this section, three modules are added into the algorithm which reduce the computation requirement of MCTI by more than two order of magnitude with negligible performance degradation. They are:

1. Classification of stationary and non-stationary blocks,
2. Pixel Decimation, and
3. Search Area Sampling.

2.3.1 Classification of Stationary and Non-stationary Blocks

Before we perform forward motion estimation, we check for stationary blocks. For any block B_1 located at (x, y) in the k^{th} frame, we compute the MAD between B_1 and the corresponding block B_2 located at (x, y) in the $(k-1)^{th}$ frame. If the MAD is less than a threshold T_1 , the block is declared as a stationary block. By classifying blocks as stationary blocks, we avoid performing the computation intensive motion estimation on all the stationary blocks. If the MAD is greater than or equal to T_1 , we perform forward motion estimation. After we found the best match block B_3 located at $(x + dx, y + dy)$ in the $(k-1)^{th}$ frame, we check the MAD between the blocks B_1 and B_3 . If the MAD is larger than or equal to certain threshold T_2 , it may be a poor match suggesting that B_1 may be covered in the $(k-1)^{th}$ frame. We then perform backward motion estimation. If the MAD is small than T_2 , no backward motion estimation is performed.

2.3.2 Pixel Decimation

When matching a block in the present frame to a block in the previous frame, the MAD is usually evaluated using every pixel of the block. Since block matching is based on the assumption that all pixels in a block move by the same amount, a good estimate of the motion could be obtained, in principle, by using only a fraction of the pixels in a block and the accuracy of the motion estimation should be similar. The technique of the subsampling of the pixel blocks as a means of reducing computational complexity of the block-matching algorithms has been previously reported by Liu². Figure 2 shows a pattern of a 16×16 block of pixels used in evaluating the MAD. Only the white pixels in each block are used resulting in a computation reduction factor of 2. A 4-to-1 subsampling ratio is also studied, but it results in blocky interpolated frame and thus is not used.

Figure 2. Pattern of pixel in pixel decimation.

2.3.3 Search Area Subsampling

For the sake of reducing the amount of computation needed for motion search, search area subsampling is a way to save the computation³. Here we perform motion estimation with 4-to-1 search area subsampling in a $(2W+1) \times (2W+1)$ search area, we only compare the shaded locations using the pattern in figure 3. To improve the search, we perform a local neighbourhood search around the "best" location. Using this 4-to-1 search area subsampling, we can have a reduction in computation by a factor of 4.

Figure 3. Pattern of 4-to-1 search area subsampling.

3. SIMULATION RESULTS

The proposed algorithms are simulated in the luminous component of the "Miss America" and "Salesman" sequences which are in CIF (288×352) format. The parameters W and N are both 16. In most frames, both MCTI and FMCTI can reconstruct the skipped frames with high image quality and accuracy for $M = 1$ and 2. In both sequences, the maximum blockwise MAD, evaluated using all the pixels in the $N \times N$ blocks, between any two consecutive frames is around 30. So we choose the thresholds T_1 and T_2 to be 14 and 9 respectively. In our experimentation, if T_1 is considerably larger than 14 (e.g. $T_1 = 17$), we observe many blocking artifacts and block mismatch in the interpolated frames for both sequence. The reason for choosing a T_2 less than T_1 is to reduce the possibility of block mismatch after forward motion estimation.

Assuming the range of the pixel value is from 0 to 255, PSNR is given by the following equation,

$$PSNR = -\log_{10} \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |f(x, y) - \tilde{f}(x, y)|^2}{M \times N \times 255^2}$$

where $f(x, y)$ and $\tilde{f}(x, y)$ are the pixel intensity of the original frame and the estimated frame of size $M \times N$ respectively.

In figure 4, the PSNR of both "Salesman" and "Miss America" sequences are plotted against the frame number. Here each ($M = 1$) frame is interpolated from its previous and subsequent frames. For "Salesman" sequence, the average PSNR of MCTI is 27.03 and good visual quality in general is observed. However, there are two exceptionally low PSNR regions near the 10th and 60th frame. These are the frames with very fast and large motions in both the left and right hands. In fact, the left hand is moving fast that it is blur even in the original image. This suggests that MCTI is not very effective in large motion or blurry regions. Both the 10th and 60th frames are shown in figure 7. Although the PSNR is low in both frames, the visual quality is good enough for VLBR. The average PSNR of FMCTI here is 26.96 which is very close to that of MCTI. This suggests that FMCTI is indeed a good and fast approximation of MCTI. However, the PSNR of FMCTI are considerably lower than that of MCTI around the 10th and 60th frame. This suggests that FMCTI is even less effective than MCTI in large motion regions and blurry regions. However, from figure 7, we find that the visual quality of FMCTI is still acceptable for VLBR application. For "Miss America" sequence, the average PSNR of MCTI and of FMCTI are 37.18 and 37.45 respectively. The PSNR of FMCTI is larger than that of MCTI before the 45th frame of "Miss America" sequence. Possibly because the motion is small before the 45th frame and most of the pixels are very similar to the adjacent frames, many blocks are classified as stationary in FMCTI and thus the pixels in the corresponding blocks of the interpolated frame are also very close to the original pixels at the same location. However, in MCTI, these stationary blocks are shifted by motion vectors. So the pixels in the interpolated frame may deviate from the original pixels at the same location. After the 45th frames, there are more large motions and MCTI appears to be slightly better than FMCTI in figure 4. These suggest that MCTI is slightly more effectively in large motion regions, but FMCTI is more effective in stationary regions.

In figure 5, we only plot the FMCTI for $M = 2$. We just interpolate the 2nd and 3rd frames from 1st and 4th frames; 5th and 6th frames from 4th and 7th frames of the original sequence. Both "Salesman" and "Miss America" sequences are simulated and an acceptable visual quality is observed. We observe the minima around the 10th and 60th frames are lower than the case of ($M = 1$) is expected. With a larger M , more frames are interpolated between two frames which are further apart and thus less correlated. Lower accuracy and PSNR are expected for cases with larger M . In figure 6, we also plot the FMCTI for $M = 3$. Similar to the case of $M = 2$, we can obtain 2nd, 3rd, 4th, 6th, 7th and 8th interpolated frames etc. Both "Salesman" and "Miss America" sequences are simulated but an further visual degradation are observed. The minima around the 10th and 60th frame are lower than that the case of ($M = 2$) The visual quality is marginally acceptable. These reveal that FMCTI can interpolate

one, two or even three skipped frames with acceptable image quality. Beyond that, it is likely to generate unacceptable image quality even for VLBR applications.

Another interesting phenomena can be observed in the 6th, 7th and 8th frames of the "Miss America" sequence for both MCTI and FMCTI as shown in figure 8. Here three ($M = 3$) frames are inserted between the 5th and 9th frames. The inserted 6th, 7th and 8th frames shown in figure 8 are very similar to the corresponding original frames as shown except the eyes. The eyes are wide open in the 5th frame, half open in the 6th frame, fully closed in the 7th and 8th frames, and half open in the 9th frame. When the original sequence is subsampled by a factor 4, only the 5th and 9th frames are sent and thus the eye closing is missed. It is thus impossible for any interpolation scheme to use only the 5th and 9th frames to reconstruct the eye closing sequence, including the proposed MCTI and FMCTI. However, with the proposed MCTI and FMCTI, although the interpolated eye segment is inaccurate with low PSNR, the reconstructed eye sequence are reasonable with good visual quality.

As far as computation concerned, for MCTI, since the motion estimation is based on the exhaustive full search, there are 792 blocks needed for unidirectional motion estimation (counting all forward and backward searches). The number of additions required for interpolating a frame is around 4.4×10^8 for both sequences. In FMCTI using the thresholds T_1 and T_2 , unidirectional motion estimations is performed only 21 and 18 times per frame for the "Salesman" and "Miss America" sequences respectively on the average. We also have an additional computation reduction by a factor 8 using pixel decimation and search area subsampling. The computation requirements of FMCTI required for interpolating a frame, on the average, are shown in Table 1

Table 1: Computation Requirements of FMCTI

	Salesman	Miss America
Ave. no of unidirectional search (MCTI)	792	792
Ave. no. of additions needed for interpolating a frame (MCTI)	4.4×10^8	4.4×10^8
Ave. no. of unidirectional search (FMCTI)	21	18
Ave. no. of additions needed for interpolating a frame (FMCTI)	1.5×10^6	1.3×10^6

Thus, FMCTI reduces the computation requirement of MCTI by more than two orders of magnitude with negligible performance degradation.

4. CONCLUSION

In this paper, we present a postprocessing technique call motion compensated temporal interpolation (MCTI), to be used in conjunction with source temporal subsampling, to interpolate skipped frames at receiver as a post-processing means from a subsampled video sequence. We also propose fast version of MCTI, the FMCTI, which can save the computation requirement by two order of magnitude with negligible performance degradation. The major problem associate with temporal frame interpolation is the moving objects particularly the uncovered areas and the newly covered areas around moving objects. Using both forward and backward motion searches to compensate for object motions, the problem can apparently be solved by MCTI and FMCTI. For a video sequence in CIF format, using a 16×16 block and 33×33 search area, we can interpolate a frame with around 1.5×10^6 additions using FMCTI.

5. REFERENCES

1. H. G. Musmann, P. Pirsch, and H. J. Graller, "Advances in picture coding," Proc. of the IEEE, vol. 73, no. 4, pp. 523-548, Apr. 1985.
2. B. Liu and A. Zaccarin, "New algorithms for the estimation of block motion vectors," IEEE Trans. on Circuits and Systems for Video Technology, vol. 3, no. 2, pp. 148-157, Apr. 1993.
3. Y. H. Fok and O. C. Au, "A fast block matching algorithm in feature domain," Proc. of IEEE Workshop on Visual Signal Processing and Communications, Melbourne, pp. 199-202, 21-22 Sept. 1993.

Figure 4. PSNR of the interpolated frames of "Salesman" (left) and "Miss America" (right) sequence ($M=1$).

Figure 5. PSNR of the interpolated frames of "Salesman" (left) and "Miss America" (right) sequence ($M=2$).

Figure 6. PSNR of the interpolated frames of "Salesman" (left) and "Miss America" (right) sequence ($M=3$).

Original

MCTI
Estimated

FMCTI
Estimated

10th

60th

Figure 7. Comparison of the image quality between MCTI and FMCTI in "Salesman" sequence. (M=1)

Figure 8. "Salesman" Sequence (M=3)