LOST MOTION VECTOR RECOVERY ALGORITHM

Choong Soo Park, Jongchul Ye, and Sang Uk Lee
Signal Processing Lab., Dept. of Cont. & Inst. Eng.
Seoul National University, Seoul 151-742, KOREA
Fax : 822-885-6620 email:pcs@asrispmp1.snu.ac.kr.

ABSTRACT

In motion compensated video coding, if motion vectors are lost or received with errors, not only will the current frame be corrupted, but also the errors will propagate to succeeding frames. The effects of errors can be further magnified by the fact that motion vectors are usually coded differentially. In this paper, we propose a technique using motion vector smoothing in the encoder and both boundary matching and motion vector consistency check in the decoder to compensate for lost or errorneously received motion vectors. The proposed technique produces noticeably better results than those reported previously. And the proposed motion vector smoothing algorithm causes virtually no degradation in the decoded image quality when motion vectors have no errors.

1. INTRODUCTION

In most video coding standard, the motion compensated coding technique is widely used to exploit the temporal redundancy. But in motion compensated video coding, if the motion vectors are lost or received with errors, then not only will the current frame be corrupted, but also the errors will propagate to the succeeding frames. Furthermore, since the motion vectors are usually coded differentially, the effects of errors can be magnified, implying that even single lost or erroneously received motion vector can damage the decoded images severely. Thus a recovery technique to compensate for the lost motion vectors is of importance in the decoder. To recover lost or erroneously received motion vectors, the intra frame median or a past motion vector can be used [1]. A technique based on the boundary matching also has been proposed [2], and the algorithm in [2] yields noticeably better results than that in [1]. Notice that both [1] and [2] attempt to recover lost or erroneously received motion vectors only in the decoder. However, if the motion vectors are preprocessed properly in the encoder, we shall show that we can recover the corrupted motion vectors more easily and accurately. In this paper, in an attempt to recover the corrupted motion vectors, we propose a technique based on both boundary matching and motion vector consistency check. In addition, to perform motion vector recovery more efficiently, we propose a motion vector smoothing technique.

The main purpose of the proposed motion vector smoothing algorithm is to make the motion vector for current block equal to one of the motion vectors of its surrounding blocks. Thus, the lost or erroneously received motion vector can be correctly recovered from the motion vectors of surrounding 8 blocks. However, if there is no error, then one might expect some performance degradation in the reconstructed image by the motion vector smoothing, because the displaced frame difference inevitably increases to some extent [3]. However, experimental results show that the proposed motion vector smoothing yields very little performance degradation in the reconstructed image. In [2], to choose the proper motion vector among candidate vectors, the boundary matching algorithm has been suggested. The boundary matching algorithm is based on the boundary smoothing constraint and the motion vector is chosen so as to minimise the inter-sample differences between neighboring blocks. But this technique seems inadequate to our applications, since the boundary smoothing constraint does not work well if the pixel values change abruptly, as in the area near edges. Thus we propose an improved boundary matching algorithm to cope with slanting edges and abrupt gray level changes in the image. In addition, we employ the number of motion vector occurences in the 3×3 window of the motion vector domain, as well as inter-sample differences between neighboring blocks, to select a proper motion vector. By adding the motion vector consistency criterion to the improved boundary matching algorithm, the performance of motion vector recovery is noticeably improved compared with [2].

In section 2, for the sake of completeness, we describe briefly coding scheme considered in this paper and the boundary matching algorithm [2]. In section 3, we propose a motion vector recovery algorithm, and section 4 presents several simulation results and comparisons with [2]. Finally, section 5 provides the conclusion.

2. BOUNDARY MATCHING ALGORITHM

The coding technique considered in this paper to demonstrate the performance of the proposed motion vector recovery algorithm is similar to the MPEG2 standard [4]. However, the main difference is that every frame is either I (intra) or P (predictive) frame in our approach, implying that there are no B (interpolative) frames. The I frames occur at every 16th frames and all the other frames are P frames. Notice that in MPEG2, the motion vectors are coded differentially. Thus, if a motion vector is lost or received erroneously, then the error propagates to the succeeding blocks within a slice. In our approach, it is assumed that only the motion vectors are corrupted by errors and their positions are also known. It is noted that the unit 'block' in this paper corresponds to 'Macro Block' in MPEG2, which consists of an array of 16×16 pixels.

Now, let us revisit an existing algorithm in [2]. The boundary matching algorithm [2] selects the motion vector $\hat{d}$ among a set of candidate vectors which minimizes the total variation between the boundaries of the current image block and those of the adjacent ones. The total variation C is defined as

$$\begin{aligned} C_A &= \sum_{x=x_0}^{N-1+x_0} (\hat{f_R}(x, y_0) - f_R(x, y_0 - 1))^2 \\ C_L &= \sum_{y=y_0}^{N-1+y_0} (\hat{f_R}(x_0, y) - f_R(x_0 - 1, y))^2 \\ C_B &= \sum_{x=x_0}^{N-1+x_0} (\hat{f_R}(x, y_0 + N - 1) - f_R(x, y_0 + N))^2 \\ C &= C_A + C_L + C_B \end{aligned} \quad (1)$$

where $f_R(x, y)$ is the pixel of the reconstructed frame and $\hat{f_R}(x, y)$ is the pixel of estimated image block using an estimated motion vector. The size of an image block is $N \times N$ and the upper left coordinate of the block is (x_0, y_0).

A full search algorithm, which uses the set of all possible motion vectors as candidate vectors, may be too complicated to be implemented in the decoder. Thus, in [2], the candidate vectors are chosen as the motion vectors of the same block in the previous frame, the available neighboring blocks, the average of the available neighboring blocks, and the zero motion vector. However, it is noted that a true motion vector may not exist in the candidate vectors, resulting in some degradation in the reconstructed image.

3. PROPOSED MOTION VECTOR RECOVERY ALGORITHM

The proposed motion vector recovery algorithm consists of the motion vector smoothing in the encoder and the proper motion vector selection among candidate vectors in the decoder. As described previously, in [2], among candidate motion vectors considered, there may not exist a true motion vector. To resolve this problem, we use a motion vector smoothing technique in the encoder. Generally, the motion vector smoothing increases the correlation between motion vectors [3]. Therefore, we can improve the differential encoding effiency of the motion vector. On the other hand, the DFD (displaced frame difference) variance increases slightly due to the fact that some of correct motion vectors are replaced with incorrect motion vectors.

Now, let us describe the proposed motion vector smoothing algorithm in detail. Consider 3×3 window shown in Fig. 1, where B_c denotes the current block. If any motion vectors of neighboring blocks $B_1, B_2, \cdots, B_8$ are not the same as the motion vector of current block B_c, then among the motion vectors of neighboring blocks, the one which minimizes the DFD of current block B_c is selected. Then, the motion vector of current block B_c is substituted with

B1	B2	B3
B4	Bc	B5
B6	B7	B8

Figure 1. 3×3 window of blocks.

the selected one. Through this motion vector smoothing, at least one of neighboring blocks $B_1, B_2, \cdots, B_8$ contains the same motion vector as that of current block B_c. Thus, if motion vectors are lost or erroneously received, using the motion vectors of neighboring blocks as candidate vectors, the motion vector of the current block can be correctly recovered. In order to choose a proper motion vector among candidate vectors, the boundary matching algorithm [2] can be used. The boundary matching algorithm in [2] relies on the fact that the image pixels in original image are highly correlated. But, unfortunately, it is found the the probability of finding a correct motion vector is not high with the boundary matching algorithm. Thus, in this paper, we propose a motion vector selection technique using both the improved boundary matching algorithm and the motion vector consistency between neighboring blocks.

Improved Boundary Matching Algorithm

It can be observed that neighboring pixel values change abruptly when there exist edges or the spatial sampling ratio is not high enough to cope with the rapid gray level change in the origianl image. As described previously, the boundary matching algorithm [2] does not work properly when pixel values change abruptly.

200	200	190	5	7	8	9	6	7	170	165	166	180	190	175	180
200	190	5	6	7	8	9	9	170	165	166	167	180	190	170	175

Figure 2. Pixel value near a slanting edge

For example, when there are slanting edges as shown in Fig. 2, the boundary matching algorithm in [2] yields very large variation. To overcome this ambiguity, we take into account the differences between diagonally connected pixel values :

$$\begin{aligned} Diff_1(x, y_0) &= \hat{f_R}(x, y_0) - f_R(x - 1, y_0 - 1), \\ Diff_2(x, y_0) &= \hat{f_R}(x, y_0) - f_R(x, y_0 - 1), \\ Diff_3(x, y_0) &= \hat{f_R}(x, y_0) - f_R(x + 1, y_0 - 1). \end{aligned} \quad (2)$$

In addition, in order to cope with rapid gray-level changes in the image, we also consider differences, given by

$$\begin{aligned} Diff_4(x, y_0) &= \hat{f_R}(x, y_0) - \bar{f_R}(x - 1, y_0) \\ Diff_5(x, y_0) &= \hat{f_R}(x, y_0) - \bar{f_R}(x + 1, y_0) \end{aligned} \quad (3)$$

where $\bar{f_R}(x - 1, y_0)$ and $\bar{f_R}(x + 1, y_0)$ are sub-pixel values illustrated in Fig. 3 and defined as

$$\begin{aligned} \bar{f_R}(x - 1, y_0) &= \frac{1}{4} \times (f_R(x - 1, y_0 - 1) + \\ &\quad f_R(x, y_0 - 1) + \hat{f_R}(x - 1, y_0) + \hat{f_R}(x, y_0)), \\ \bar{f_R}(x + 1, y_0) &= \frac{1}{4} \times (f_R(x + 1, y_0 - 1) + \\ &\quad f_R(x, y_0 - 1) + \hat{f_R}(x + 1, y_0) + \hat{f_R}(x, y_0)). \end{aligned} \quad (4)$$

Then, a variation between the boundary of current image block and that of above block C'_A is defined as

$$\begin{aligned} C'_A = \sum_{x=x_0}^{N-1+x_0} & Min\{|Diff_1(x, y_0)|, |Diff_2(x, y_0)|, \\ & |Diff_3(x, y_0)|, |Diff_4(x, y_0)|, |Diff_5(x, y_0)|\}. \end{aligned} \quad (5)$$

compared with [2], (5) reflects the effects of slanting edges and rapid gray-level change in the image. The variations between the image block and the one to its left and the one below it, denoted by C'_B and C'_L, can be similarly obtained. The total variation C' is defined as

$$C' = C'_A + C'_L + C'_B. \quad (6)$$

Motion Vector Consistency Criterion

As described previoulsy, through motion vector smoothing, the correlation between neighboring motion vectors becomes very high. This criterion exploits such property. After we compute total variance C' for all candidate vectors, the motion vectors yielding smaller total variation C' than a pre-determined threshold are selected. Here, the threshold is empirically chosen to be 1.25 times the smallest C' of current block. Then, among the selected motion vectors, the one with maximum number of occurrences in the 3×3 window is finally chosen. If the numbers of occurrences are same for some selected motion vectors, the one which yields the smallest C' is chosen.

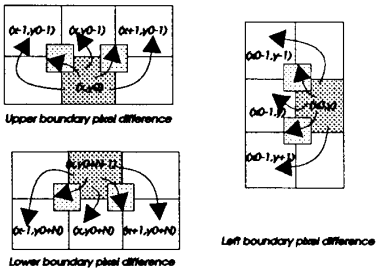

Figure 3. Inter-sample differences for each direction

4. EXPERIMENTAL RESULTS

First, we shall examine the effects of the proposed motion vector smoothing algorithm on the reconstructed images, assuming there are no error. Based on the coding scheme described previouly, the reconstructed images of 'FLOWER GARDEN' and 'FOOTBALL' sequences with and without motion vector smoothing are compared. The PSNR's are presented in Fig. 4 and Fig. 5, respectively. As can be seen, the results are almost comparable. This observation is due to the fact that through motion vector smoothing, the encoding effiency of the motion vector is improved, while the DFD variance slightly increases, and their effects in the coded image quality cancel each other.

The proposed motion vector recovery algorithm and [2] are compared in Table 1. In Table 1, both the case of perfect reconstruction and reconstruction within the estimation accuracy of 1 pixel error are provided. Table 1 clearly demonstrates that the inter-block smoothing constraint, which minimizes the inter-sample differences between adjacent blocks [2], seems an inappropriate criterion to select the correct motion vector. On the other hand, the proposed motion vector recovery scheme, which utilizes not only the improved boundary matching algorithm but also motion vector consistency check between neighboring blocks, provides much improved performance. In Fig. 4 and

Table 1. Motion Vector Recovery Performance.

Image	Algorithm	Recovery accuracy	
		0 pixel	1 pixel
Flower garden	Proposed	72.981(%)	88.953(%)
	[3]	43.545(%)	62.324(%)
Foot-ball	Proposed	80.118(%)	89.040(%)
	[3]	41.350(%)	63.201(%)

Fig. 5, the PSNR's of the decoded sequences of 'FLOWER GARDEN' and 'FOOTBALL' are also provided. In the experiment, the loss probability of the motion vectors is set to be 2 vectors/frame, and it is assuemed that error does not occur in contiguous slices. Since the motion vectors are coded differentially, if one motion vector is lost, then the succeeding motion vectors within a slice are all considered to be lost. Thus, the motion vectors of neighboring blocks within a slice, such as motion vectors of block B_4 and B_5 of Fig. 1, are not considered in the proposed motion vector smoothing and recovery procedure. As is observed, the PSNR's of the decoded sequences using the proposed algorithm is 1-2 dB higher than those of [2]. From Fig. 6 to Fig. 9, we illustrate the reconstructed images with the boundary matching algorithm [2] and the proposed algorithm. For given probability of loss, practically, about 70 motion vectors are lost per each frame, and the effects of errors in the motion vectors are visible, even with the proposed motion vector recovery algorithm. But, compared with [2], the proposed algorithm provides noticeably better image quality. In conclusion, if motion vectors are lost or erroneously received, compare to existing algorithms [1, 2], the proposed algorithm recovers the motion vector more accurately, and provides noticeably better image quality.

5. CONCLUSIONS

In this paper, we proposed the lost motion vector recovery algorithm using motion vector smoothing in the encoder and both improved boundary matching algorithm and motion vector consistency check in the decoer. The proposed motion vector smoothing algorithm provides virtually the same image quality when there are no erros. When motion vectors are lost or erroneously received, compare to the previous algorithm [2], the proposed algorithm recovers motion vector more accurately and provides better image quality.

REFERENCES

[1] P. Haskell and D. Messerschmitt, "Resynchronization of motion compensated video affected by ATM cell loss," *Proc. ICASSP*, vol 3, pp. 545-548, Mar. 1992.

[2] W. M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or errorneously received motion vectors," *Proc. ICASSP*, vol. 5, pp. 417-420, Mar. 1993.

[3] W. Y. Choi and R. H. Park, "Motion vector coding with conditional transmission," *Signal Processing*, vol. 18, pp. 259-267, 1989.

[4] CCITT SG XV, Coded representation of picture and audio information, ISO-IEC/JTC1/WG11, MPEG92/N0245, July, 1992.

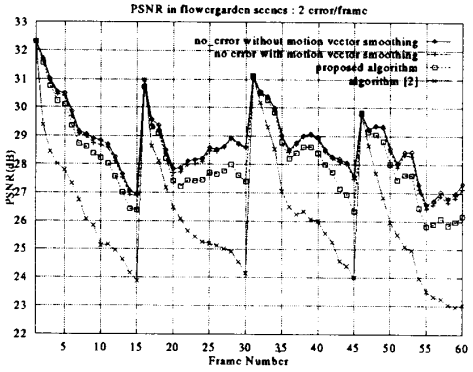

Figure 4. PSNR's for decoded 'FLOWER GARDEN'.

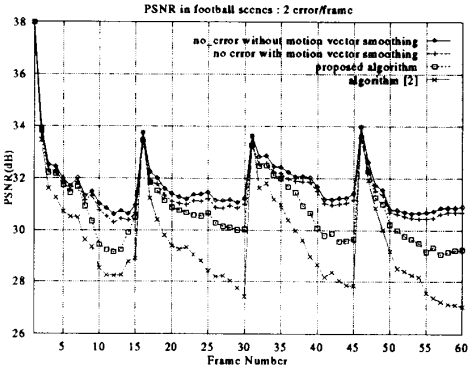

Figure 5. PSNR's for decoded 'FOOTBALL'.

Figure 6. Error recovery of 'FLOWER GARDEN' with [2].

Figure 7. Error recovery 'FLOWER GARDEN' with proposed.

Figure 8. Error recovery of 'FOOTBALL' with [2].

Figure 9. Error recovery of 'FOOTBALL' with proposed.