INTERPOLATION OF SKIPPED IMAGE FRAMES FOR IMAGE SEQUENCE CODING

Ehud Weiner and David Malah

Technion - Israel Institute of Technology
Department of Electrical Engineering
Haifa 32000, Israel

ABSTRACT

Image-sequence coding for Video-conferencing and Video-phone requires high compression ratios to meet the required low transmission rates. Frame skipping at the encoder and temporal interpolation at the decoder are commonly used techniques to achieve this goal. This work presents several algorithms for temporal interpolation which permit frame skipping with less degradation in the quality of the interpolated frames. In the developed algorithms there is a trade-off between the quality of the interpolated frames and the amount of additional side-information used. A smoothing algorithm of the motion-vector field, based on a selective median filter is also introduced. This smoothing is intended to prevent blocking effects created by sub-optimal block matching or stray motion. The performance of the proposed algorithms are compared in image-sequence coding simulations.

I. INTRODUCTION

Some applications of image-sequence coding are in Video-conferencing and Video-phone. These applications require image transmission at low bit rates such that even B or 2B ISDN channels could be used, meaning rates of 64Kbit/s or 128Kbit/s, respectively.

The basic algorithm for image-sequence coding used in this work is described in section II. It is based on predicting frame $n+1$ from the already reconstructed frame n, using motion compensation, and transform coding of blocks that need to be replenished.

One of the means for reducing the transmission rate is to skip frames, while encoding the image-sequence, and filling-in the skipped frames at the decoder using *temporal interpolation*. This work deals with improving the quality of interpolated frames, while keeping the additional side-information needed at low values.

Many articles have been written about temporal interpolation. Some use *frame repetition*, e.g. [1], - resulting in jerky motion in the reconstructed sequence, while others, e.g. [2] require transmission of a relatively large amount of additional side-information, such as motion-vectors related to the skipped frames. At the above low bit-rates, the transmission of additional motion-vectors could require a considerable part of the given rate, leaving an insufficient number of bits for coding the error image. Another approach is *linear temporal interpolation* of the skipped frames [3], using motion-vectors relating to the coded frames only. This approach, while generally better than frame repetition, typically results in blocking effects. In section III of this work we propose several algorithms which avoid jerky motion and reduce blocking effects considerably, while requiring a small amount or even no additional side-information.

In section IV of this paper we present an algorithm for smoothing the motion-vector field. The smoothing reduces the effects of stray motion and suboptimal matching, and helps to improve the quality of the interpolated frame. Section V presents simulation results and section VI concludes the paper.

II. CODING SYSTEM

The basic coding system used in this study is a hybrid/DCT coder based on CCITT recommendation H.261 [4]. Assuming that we have previously reconstructed frame R_n, we reconstruct frame $n+1$ using data and parameters computed from R_n and from the original frame X_{n+1}. The luminance frames are divided into macroblocks of 16×16 pels, where each macroblock is further divided into four subblocks of 8×8 pels - if it is coded. First, motion-vectors between frames X_{n+1} and R_n are found. For this purpose the Block Matching Algorithm (BMA) is used for the displacement estimation of each macroblock, applying a *three-step algorithm* [5] to reduce the complexity of the BMA in simulations. This algorithm searches for the minimum Mean Absolute Difference (MAD) value for a macroblock in X_{n+1} with respect to a macroblock in an appropriate search area A in R_n. The minimum MAD value for block (k,l) is:

$$MAD^{k,l}_{mc} = \min_{v_k, v_l \in A} \sum_{i,j \in B_{k,l}} | X_{n+1}(i,j) - R_n(i+v_k, j+v_l) | \quad (1)$$

where the values of v_k and v_l minimizing the MAD are components of the motion-vector (displacement) for the macroblock $B_{k,l}$. MAD_{mc} denotes the MAD value with motion compensation. With the completion of the motion estimation stage we have the motion-vector for each macroblock and the MAD values before and after motion compensation (MAD and MAD_{mc}, respectively). Frame $n+1$ could, in principal, be reconstructed by copying each macroblock from frame n shifted by the appropriate motion-vector:

$$R_{n+1}(i,j) = R_n(i+v_k, j+v_l) \quad ; i,j \in B_{k,l} \quad (2)$$

However, the reconstruction using motion-compensation is not good enough for some blocks in the frame, because the motion-vectors do not always represent the real motion between two frames, and because there may be other changes between the two frames. To improve the quality of the reconstructed frame, conditional block replenishment is used (i.e. some blocks are coded), as follows: For a given replenishment threshold T_r, every macroblock with $MAD_{mc} > T_r$ is coded either by *inter-frame* coding, or *intra-frame* coding [4,6]. Each subblock to be coded is transformed using the two-dimensional Discrete Cosine Transform (DCT), and the transform coefficients are quantized.

The motion-vectors, block types (such as inter, intra, coded), and the quantized DCT coefficients are entropy coded using variable length codes (VLC) and transmitted to the decoder.

In the decoder the coefficients are decoded, an Inverse DCT (IDCT) is performed on the coded blocks, and frame $n+1$ is reconstructed as described above.

Since frame $n+1$ is reconstructed in the decoder, based on the already reconstructed frame n, all the computations in the encoder must also be made on the same frame as in the decoder. Therefore, the above reconstruction process is performed in the encoder as well. After the reconstruction, a two dimensional smoothing filter may be applied on some of the macroblocks [4], both in the encoder and the decoder.

III. TEMPORAL INTERPOLATION

In an algorithm combining frame skipping and interpolation we code each k-th frame in the encoder, using the above coding algorithm, while skipping $k-1$ frames between every two coded frames. The missing frames are reconstructed in the decoder by temporal interpolation. We developed several algorithms for temporal interpolation, in which the effects created by the algorithms described in section I are alleviated. The proposed algorithms depend on the number of frames skipped between any two coded frames, and are as follows:

A. Interpolation of one skipped frame

If one in every two frames is skipped, let frame n be the previous coded frame, frame $n+2$ the next coded frame, and frame $n+1$ the temporally interpolated one. We propose the following three algorithms which offer a trade-off between quality and side-information.

Algorithm I

In this algorithm each block in the interpolated frame is reconstructed as a linear combination of matched blocks in the previous and next coded frames. I.e. for $i,j \in B_{k,l}$:

$$R_{n+1}(i,j) = W_1 R_n(i+v'_k, j+v'_l) + W_2 R_{n+2}(i-v'_k, j-v'_l) \quad (3)$$

where v'_k, v'_l are the motion vectors components used in the interpolation and obtained by taking half the value of v_k, v_l respectively, truncated to integer values. Note that for blocks copied from the next coded frame (R_{n+2}) the direction of the motion is reversed, i.e., a negative sign is used.

The optimal values of W_1 and W_2 are found by minimizing *the sum of squared errors* (SSE) for each block of the error image E:

$$E(i,j) = X_{n+1}(i,j) - R_{n+1}(i,j) \quad (4)$$

From (3) and (4), the SSE for block (k,l) is given by:

$$SSE \triangleq \sum_{i,j \in B_{k,l}} E^2(i,j) = W_1^2 a + W_2^2 b + f + 2W_1 W_2 c - 2W_1 d - 2W_2 e \quad (5)$$

where:

$$a \triangleq \sum_{i,j \in B_{k,l}} R_n^2(i+v'_k, j+v'_l)$$

$$b \triangleq \sum_{i,j \in B_{k,l}} R_{n+2}^2(i-v'_k, j-v'_l)$$

$$c \triangleq \sum_{i,j \in B_{k,l}} R_n(i+v'_k, j+v'_l) R_{n+2}(i-v'_k, j-v'_l)$$

$$d \triangleq \sum_{i,j \in B_{k,l}} R_n(i+v'_k, j+v'_l) X_{n+1}(i,j)$$

$$e \triangleq \sum_{i,j \in B_{k,l}} R_{n+2}(i-v'_k, j-v'_l) X_{n+1}(i,j)$$

$$f \triangleq \sum_{i,j \in B_{k,l}} X_{n+1}^2(i,j) \quad (6)$$

Differentiating eqn. (5) with respect to W_1 and W_2 and equating to zero the minimum value of SSE is obtained for the following values of W_1, W_2:

$$W_1^* = \frac{bd - ce}{ab - c^2} \quad (7a)$$

$$W_2^* = \frac{ae - cd}{ab - c^2} \quad (7b)$$

If the weights are constrained by the condition that $W_1+W_2=1$, eqn. (3) is replaced by:

$$R_{n+1}(i,j) = W_1 R_n(i+v'_k, j+v'_l) + (1-W_1) R_{n+2}(i-v'_k, j-v'_l) \quad (8)$$

Substituting $W_2 = 1-W_1$ in (5), the corresponding SSE is:

$$SSE = W_1^2 a + (1-W_1)^2 b + f + 2W_1(1-W_1)c - 2W_1 d - 2(1-W_1)e \quad (9)$$

Differentiating eqn. (9) with respect to W_1, and equating to zero, we get the minimum value of the SSE for

$$W^*_{1_c} = \frac{b + d - c - e}{a + b - 2c} \quad (10)$$

The calculation in (10) (or in (7) if the weights are unconstrained) is done for each macroblock of the interpolated frame. The values of W_{1_c} (or W_1, W_2 - if the weights are unconstrained) are quantized, encoded, and transmitted as additional side-information for the interpolated frame.

Algorithms II & III

In the other two algorithms the blocks of the interpolated frame $n+1$ are copied from one of the three following sources:

1. The previous coded frame (R_n), using motion-compensation based on the motion-vector field computed between frames X_{n+2} and R_n:

$$R_{n+1}(i,j) = R_n(i+v'_k, j+v'_l) \quad ; i,j \in B_{k,l} \quad (11)$$

2. The next coded frame (R_{n+2}), using motion compensation which is again based on the motion-vector field computed between frames X_{n+2} and R_n:

$$R_{n+1}(i,j) = R_{n+2}(i-v'_k, j-v'_l) \quad ; i,j \in B_{k,l} \quad (12)$$

3. The average of the previous coded frame (R_n) and the next coded frame (R_{n+2}), using motion compensation with respect to both frames. This is similar to the ISO-MPEG approach for reconstructing a "bidirectional" frame [6], but without computing motion-vectors for the interpolated frame. Thus, for $i,j \in B_{k,l}$,

$$R_{n+1}(i,j) = \frac{1}{2}\left[R_n(i+v'_k, j+v'_l) + R_{n+2}(i-v'_k, j-v'_l)\right] \quad (13)$$

Algorithm II

In this algorithm the decision from where to copy each block is based on the MAD value between the block in the original frame ($n+1$) and the appropriate block in one of the three possible sources described above. The source giving the minimum MAD value is used. In this algorithm we need a small amount of side-information to indicate to the decoder from which of the three sources each block is to be copied. The bits which indicate this information are part of the classification data transmitted for each macroblock of the coded frame.

Algorithm III

Here, the decision from where to copy each block is made considering the *change* in the motion-vector field from the previous frame (motion in X_n with respect to frame R_{n-2}, denoted by $\underline{V}_n$) to the next frame (motion in X_{n+2} with respect to frame R_n, denoted by $\underline{V}_{n+2}$). This algorithm takes into account the fact that the motion is not necessarily linear, meaning that the velocity is not constant between frames X_{n+2} and X_n. The change in the the motion-vector field represents the *acceleration* in the motion of each block. For block (k,l) an estimate of the acceleration vector is given by:

$$\underline{A}(k,l) = \underline{V}_{n+2}(k,l) - \underline{V}_n(k,l) \quad (14)$$

If $\underline{A}(k,l)$ has positive values in *both* coordinates, the block from the previous coded frame is copied. If $\underline{A}(k,l)$ has negative

values in both coordinates, the block is copied from the next coded frame. In all other cases the blocks are copied from the average of the two motion-compensated frames. Since $\underline{A}(k,l)$ can be computed also at the decoder, no additional side-information is needed at the decoder to make the decision from where to copy each macroblock.

B. Interpolation of two skipped frames

If two in every three frames are skipped, let frame n be the previous coded frame, frame $n+3$ the next coded frame, and frames $n+1$ and $n+2$ are the ones to be interpolated. Algorithms I and II described above, with some minor necessary modifications, are more suited here than Algorithm III. The needed modifications are:

1. In both Algorithms I & II, the motion-vector components used in the interpolation are $\frac{1}{3}v_k, \frac{1}{3}v_l$ or $\frac{2}{3}v_k, \frac{2}{3}v_l$ - in proportion to the time interval between the input frames (X_n, X_{n+3}) and the interpolated frame. The calculation of the weight W_{1_ϵ} (or W_1, W_2 if the weights are unconstrained) in Algorithm I is done for each interpolated frame independently.
2. In addition, in algorithm II, the average frame previously computed by (13) is now different for each interpolated frame. The average of the coded frames (with motion compensation) is now a weighted average, with the weights being proportional to the time interval between the particular interpolated frame and the coded frame.

C. Interpolation of three skipped frames

When three in every four frames are skipped, let frame n be the previous coded frame, frame $n+4$ the next coded frame, and frames $n+1$, $n+2$, $n+3$ are the ones to be interpolated. We first reconstruct the middle missing frame (frame $n+2$), using two additional motion-vectors for each block (k,l):

1. f_k, f_l: the components of the motion-vectors computed between the original frame, X_{n+2}, and the previous coded frame R_n.
2. g_k, g_l: the components of the motion-vectors computed between the original frame, X_{n+2}, and frame R_{n+4}.

Each block in the interpolated frame $n+2$ is then copied from one of the three following sources:

1. The previous coded frame using motion compensation with motion-vector components f_k, f_l:

$$R_{n+2}(i,j) = R_n(i+f_k, j+f_l) \quad ; i,j \in B_{k,l} \tag{15}$$

2. The next coded frame using motion compensation with motion-vector components g_k, g_l:

$$R_{n+2}(i,j) = R_{n+4}(i+g_k, j+g_l) \quad ; i,j \in B_{k,l} \tag{16}$$

3. The average of the previous coded frame (n) and the next coded frame ($n+4$), both with motion compensation. This possibility is like the approach used for a "bidirectional" frame by ISO-MPEG [6]. I.e., for $i,j \in B_{k,l}$,

$$R_{n+2}(i,j) = \frac{1}{2}\left[R_n(i+f_k, j+f_l) + R_{n+4}(i+g_k, j+g_l)\right] \tag{17}$$

As before, the decision from where to copy each block is based on the MAD value between the original block and each of the three possible sources.

The additional motion-vectors with components f_k, f_l, g_k, g_l, are needed for improving the quality of the interpolation of frame $n+2$, because this frame is used in the sequel for the reconstruction of the interpolated frames $n+1$ and $n+3$. This requires, of course, more side-information for these additional motion-vectors. However, since more frames are skipped here, this additional side-information can be tolerated. Frames $n+1$ and $n+3$ are interpolated using any one of the three algorithms presented above for the case of one skipped frame, regarding the interpolated frame $n+2$ as a reconstructed frame. Frame $n+2$ is reconstructed in the encoder as well, since it is needed for the interpolation of frames $n+1$ and $n+3$.

IV. MOTION-FIELD SMOOTHING

The motion estimation algorithm produces a field of motion-vectors which represent the displacement of blocks between an original frame X_{n+k} and the previous coded frame R_n. These motion-vectors do not always represent the real motion between the two frames because of noise, local minimum obtained by the Block Matching Algorithm (as with the three-step approach), limited size of the search area in the motion estimation algorithm, rotational motion, etc. In background areas, or other areas without motion, the motion estimation algorithm could still produce non-zero motion-vectors ("stray motion") due to noise and slight jitter in the sampling points (particularly in textured areas). Non-smoothed motion-vectors may therefore result in incorrect displacement values of some blocks in the coded frame and therefore also in the interpolated frames. Reconstruction of frames with these motion-vectors could also result in the need for more replenished blocks, which under low bit-rate constraints may result in blockiness in the reconstructed frame. The smoothing of the motion-vector field helps to alleviate this problem. In particular, since the interpolated frames are obtained from the coded frames with motion compensation, these effects are even more pronounced in such frames, and smoothing of the motion-vector field is of great importance.

We applied therefore a smoothing algorithm to smooth the motion-vector field. This algorithm is a *selective median filter* which operates on the motion-vectors of groups of 3×3 macroblocks, with each component computed separately. A selection procedure is used during the filtering operation: Of the 9 motion-vectors, only those for which the improvement achieved by the motion-compensation exceeds a threshold (i.e. the ratio $MAD/MAD_{mc} > T_{imp}$) are taken into account in the median operation for each component. Furthermore, after the median value is found, the MAD value for the center macroblock in the 3×3 group of blocks, using the median motion-vector, denoted here as MAD_{mf} (Median Filtered MAD) is computed. If the ratio MAD_{mf}/MAD_{mc} is larger than a threshold T_{rej}, we reject this new motion-vector and keep the original one.

V. SIMULATION RESULTS

The proposed algorithms were examined by computer simulations. The performance of the interpolation algorithms is shown here by Peak SNR (PSNR), which is an objective measure of quality, although subjective quality was also considered. The simulations were done using the first 100 frames of the sequence "Miss America". The replenishment thresholds T_r and $T_{r'}$ (section II) were set between 1.5 and 2 gray levels per pel, depending on the number of skipped frames. The transmission rate was limited to 100Kbit/s for the luminance images, which corresponds to 128Kbit/s for a full color sequence. Simulations were made for Algorithms I & II as well as for *frame repetition* with decimation ratios of 2:1, 3:1, 4:1. Algorithm III was examined only for one skipped frame (decimation ratio of 2:1), as this algorithm is not suited for other decimation ratios. The simulation of the image-sequence coder with no interpolation was also performed - for comparison purposes.

In Figure 1, the motion-vectors are shown before (Fig. 1a) and after (Fig. 1b) smoothing. The smoothing in this example, as in the simulations, is done with T_{imp}=1.2 and T_{rej}=1.7 (see section IV). For lower values of T_{imp} the undesired motion-vectors effect the smoothed motion-vector field; whereas for higher values, the smoothing is not effective in the area with actual motion. Lower values of T_{rej} cause the rejection of too many filtered vectors, while higher values allow intolerable degradation in some of the blocks in the reconstructed frame. The "stray motion" seen in the background area of Fig. 1a is smoothed by the algorithm as shown in Fig. 1b where there is no motion in this area.

In Table 1 the average PSNR values obtained for the interpolated frames, the coded frames, and the whole sequence are given. The amount of additional side-information required by each algorithm is also shown. The average PSNR values do not always represent the perceived quality of an image sequence by a human observer, since there are artifacts which are hardly reflected in the PSNR, but could be annoying to the viewer. Based on informal viewing, the quality of the reconstructed sequence using any of the proposed algorithms was judged to be very good and the effects mentioned in section I were hardly observed.

It is obvious from Table 1 that for interpolation with the proposed algorithms, the average PSNR of the coded frames, as well as for the whole sequence, is better than the average PSNR obtained without interpolation. For all decimation ratios used, all the proposed algorithms perform better than simple frame repetition. For a decomation rate of 2:1, Algorithm II has the best overall performance, because Algorithm I requires a much larger amount of side-information than Algorithm II, which leaves less bits for the coded frames. The interpolated frames produced by Algorithm III, which uses no additional side-information, has lower average PSNR compared with the other two algorithms, but still their subjective quality is quite good. If a decimation ratio of 3:1 is used, Algorithm I is the best, whereas if the decimation ratio is 4:1, again Algorithm II is slightly better.

VI. SUMMARY AND CONCLUSIONS

In this paper three interpolation algorithms are presented, followed by results of simulations made to examine them. The main conclusions from the simulation results are that all three algorithms have close performance for the rate of 100Kbit/s (for luminance frames) considered here, although there is an advantage for Algorithm I with a decimation ratio of 3:1. For another transmission rate, using a different decimation ratio or/and interpolation algorithm may be more advantageous. Another consideration which should be taken into account is the complexity of the proposed algorithms: Algorithm I is the most complex one, while Algorithm III is the simplest. In order to reduce the transmission rate further, there is a need for additional means, such as spatial decimation and vector quantization.

References

[1] S.Y. Othman, S.G. Wilson, "Image Sequence Coding at 64KBPS Using Vector Quantization and Block Matching", *ICASSP-89* pp. 1913-1916, 1989.

[2] C. Cafforio, F. Rocca, S. Tubaro, "Motion Compensated Image Interpolation", *IEEE Trans. Commun.* vol. 38 pp. 215-222, Feb. 1990.

[3] C.C. Evci, H. Waldburger, "Low Bit Rate Videophone Codec for ISDN Applications", *Image Communications*, vol. 1, pp. 225-238, 1989.

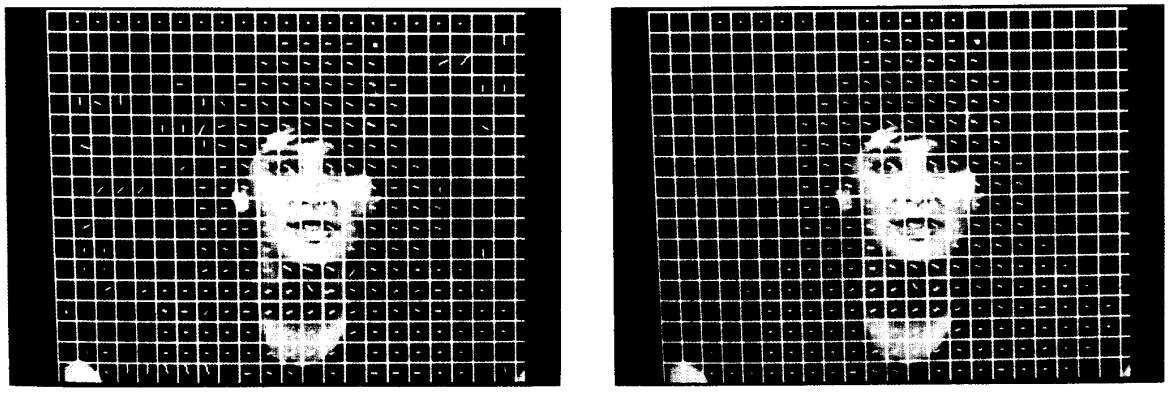
[4] "CCITT Draft revision of recommendation H.261 - Video Codec for Audiovisual Services at k × 64 Kbit/s", *Image Communications*, vol. 2, pp. 221-239, Aug. 1990.

[5] J.R. Jain, A.K. Jain, "Displacement Measurement and its Applications in Interframe Image Coding", *IEEE Trans. Commun.* vol. 29 pp. 1799-1808, Dec. 1981.

[6] A. Nagata, I. Inoue, A. Tanaka, N. Takeguchi, "Moving Picture Coding System for Digital Storage Media Using Hybrid Coding", *Image Communications*, vol. 2, pp. 109-116, Aug. 1990.

TABLE 1: Average PSNR values and additional side-information for the different algorithms:

Decimation ratio	Alg.	PSNR			Additional side-information Kbit/s
		Coded frames	Interp. frames	Total	
1:1	-	37.69	-	37.69	-
2:1	rep.	38.76	35.24	37.00	-
	I	38.59	37.63	38.11	7.92
	II	38.76	37.64	38.20	1.67
	III	38.76	36.66	37.71	-
3:1	rep.	39.04	35.89	36.94	-
	I	38.69	38.17	38.34	17.42
	II	38.86	37.73	38.10	5.34
4:1	rep.	39.47	34.97	36.10	-
	I	38.94	37.85	38.13	28.52
	II	39.31	38.00	38.33	17.16



(a) (b)

FIGURE 1: Motion-vector field (a) before smoothing (b) after smoothing