QUALITY ENHANCEMENT IN STANDARD COMPLIANT FRAME RATE UP CONVERSION BY MOTION SMOOTHING

Gökçe Dane and Truong Nguyen

Electrical and Computer Engineering Department, University of California San Diego
email: gdane@ucsd.edu, nguyent@ece.ucsd.edu

ABSTRACT

In this paper, we address the problem of standard compliant frame rate up conversion (SC-FRUC) at the decoder by using received motion vectors analysis and processing for low bit rate applications. In the proposed SC-FRUC scheme, the skipped frames are generated at the decoder by using received motion vectors only. We introduce a smooth motion vector interpolation method to enhance visual quality in FRUC. We also discuss the visual artifacts that degrade the quality of the interpolated frames in FRUC applications. Experimental results show that the proposed motion processing algorithm improves the visual quality of frames both spatially and temporally.

1. INTRODUCTION

In order to meet low bandwidth requirements, video applications such as video telephony or video streaming reduce the bit rate by encoding the video at a lower frame rate using frame skipping. However, low frame rate video produces artifacts in the form of motion jerkiness. Temporal frame interpolation methods also known as frame rate up conversion, or picture rate conversion is necessary at the decoder to interpolate the missing or skipped frames.

Frame interpolation is performed in two ways: (i) interpolation along temporal axis without taking object motion into account and (ii) interpolation along the motion trajectory. The first group of methods include Frame repetition (FR) and Frame averaging (FA). FR is interpolation by using a zero order hold filter and FA is the interpolation with haar filter (when two reference frames are used). FR produces irregular and jerky motion whereas FA results in blurring of the object in motion.

In order to achieve successful interpolation along motion trajectory, motion vectors should reflect the true object motion. However, *codec* motion vectors do not reflect the *natural* motion of the video. Better motion vectors can be obtained by using a more reliable motion estimation algorithm at the encoder or at the decoder. Some of the methods that uses motion information for frame interpolation is forward or backward motion compensated repetition and motion compensated averaging, which are known as linear methods. Besides linear filtering, non-linear combinations of motion compensated frames can also be used in picture interpolation. [3] presents various non-linear techniques such as static, dynamic or cascaded median filtering or soft switching, which aim to decrease the visibility of artifacts that are introduced due to erroneous motion vectors. In [3] the emphasis is given to interpolation methods rather than correcting or processing motion vectors.

In this paper, we study motion processing and correction problem for standard compliant (SC) motion compensated frame rate conversion (MC-FRUC). In standard compliant FRUC applications, the motion information for the interpolated frame comes from the motion information from the frame other than the one which we want to interpolate (see Fig.1). These motion vectors are not always reliable enough to use directly for interpolation of the skipped frame. As one interpolates the skipped frames at the decoder, local artifacts are introduced due to incorrect motion estimation. Furthermore, for intra-coded blocks there is no motion information available, so extra estimation or processing is necessary at the decoder.

In the rest of the paper, we first review the motion compensated frame rate conversion in section II. We also discuss some of the interpolation artifacts that occurs in frame interpolation, and then explain the problem of standard compliant MC-FRUC. In section III, we present the proposed motion vector processing algorithm, and explain how to perform motion vector assignment to make it applicable for FRUC. Section IV presents the experimental results on various uncompressed and compressed sequences with different simulation parameters. We also suggest simple techniques for further improvement to the current solution. Finally last section concludes our work.

2. MOTION COMPENSATED FRAME RATE UP CONVERSION (MC-FRUC)

Assume that the temporal variation in the image between time 0 and time t is caused by a translational motion with constant velocity $v = (v_x, v_y)$, then the image I at time t can be written as $I(x,y,t) = I(x+v_x, y+v_y, 0)$. Now assume that we have received two images P_1 at time t_1 and P_3 at time t_3. Doubling the frame rate requires interpolating the image S at time t_2 where t_2=$t_1 + (t_3 - t_1)/2$ as shown in Fig. 1. In order to predict S, motion vectors that are estimated between P_1 and P_3 are used. During encoding, frame P_3 is partitioned into blocks and a motion vector v_{ij} is assigned for each block a_{ij}, where i and j represents the row and column index respectively. The constant velocity assumption suggests that the motion vector at time t_2 is half of the motion vector v_{ij} in amplitude and it has the same direction as v_{ij} (for 1:2 upconversion). Therefore each block in the missing frame S can be interpolated as

$$a_{ij}(x,y,t_2) = f_1 \cdot a_{ij}(x+v_{x,ij}/2, y+v_{y,ij}/2, t_1) + f_2 \cdot a_{ij}(x-v_{x,ij}/2, y-v_{y,ij}/2, t_3) \quad (1)$$

where $\mathbf{f}$=[f_1 f_2] is the temporal interpolation filter. By performing the interpolation for all the blocks in the frame, we can obtain the frame S. Typically f_1 and f_2 is chosen as $1/2$ for standard motion compensated interpolation [2]. However, using equal weight temporal interpolation filter (i.e.,

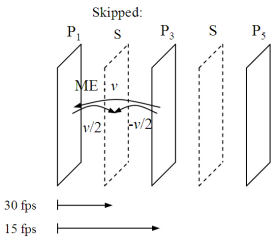
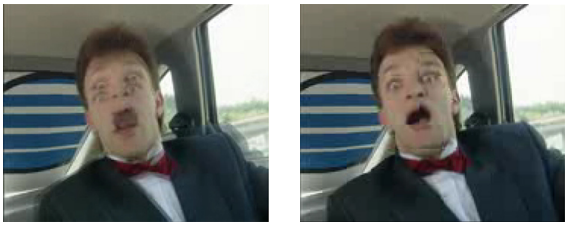


Figure 1: MC-FRUC



(i) Blurring artifact (ii) Blocking artifact

Figure 2: Visual artifacts in motion compensated frame rate up conversion

direct averaging) is not optimal for FRUC applications. In [1], we have shown that the interpolation filters can be found as a function of motion vector errors and they should be adjusted based on motion vector reliability measures. We have also found in [1] that combining multi blocks does not always decrease the prediction error. If either of the forward or backward motion compensated block has greater motion vector error variance; then combining two blocks will increase the prediction error. However, the visual quality of combining two blocks with different motion vector error variance which falls in a determined range is similar. Only beyond certain level, the artifacts are more noticeable. Next section presents some of the artifacts that can occur in FRUC.

2.1 Interpolation Artifacts in FRUC

The interpolated frames can be degraded by various artifacts such as blurriness and blockiness that occur spatially or by artifacts such as jerkiness and flickering that occur temporally. These artifacts can be briefly explained as follows.

Blur Artifact occurs when forward and backward motion compensated blocks that are used in the frame interpolation do not match properly. The mismatch is mainly caused by unreliable motion vectors and the limited search range of the motion estimation algorithms which can not capture fast moving objects. The artifacts become more noticeable around the boundaries of objects as shown in Fig. 2 (i).

Blockiness Artifact in FRUC (as shown in Fig. 2 (ii)) is somewhat different than the blockiness in standard video codecs. In standard video coding, blockiness is due to the quantization of DCT coefficients in adjacent blocks of a frame. However, in FRUC applications it is mainly due to non-smooth motion vectors between spatially adjacent blocks and the lack of prediction error for these frames.

Motion Jerkiness Artifact occurs when the motion is not smooth temporally. Smooth motion cannot be generated if the frame rate is below a certain threshold. This artifact can be seen on medium speed video, and it is more noticeable especially on camera pans.

Flickering Artifact is caused by the fluctuation of spatial image quality between adjacent frames. The quality can fluctuate due to various reasons. For example, changing the quantization steps in every alternating frame causes flickering. Or, if the interpolated frames are more blurry or blocky compared to consecutive or reference frames, the video sequence exhibits flickering. In short, any type of filtering or post processing which do not blend well within the frame will cause flickering artifacts.

The detection of these artifacts is more difficult than correcting them. The distortion measure such as the peak signal-to-noise ratio (PSNR) do not reflect the perceptual quality of the image, therefore it will not be adequate for artifact detection. For example, a slight shift may cause a low PSNR value, but it will not cause visual distortion for the still frame. A slight shift may also cause a jerkiness artifact, but still it is hard to say that the change in PSNR values of consecutive frames completely corresponds to flickering artifact. Because of these shortcomings, to make MC-FRUC an acceptable application for users, these artifacts should be reduced regardless of the PSNR value of the interpolated frame.

2.2 Standard Compliant FRUC

As reviewed in the previous section, the major problem in standard compliant MC-FRUC is the unreliable motion vectors that cause various artifacts in the interpolated frames. In order to obtain better motion vectors; methods such as motion estimation algorithm at the decoder, motion estimation at the encoder, motion processing at the decoder have been investigated. A new motion estimation algorithm at the decoder is not appropriate for low-power applications. Furthermore, since the frame is already skipped at the encoder before transmission, the motion estimation between the reference and current frames at the decoder does not provide the true motion vectors for the frame that we want to interpolate. In order to obtain better motion vectors, natural or true motion estimation is performed at the encoder, and these motion vectors are transmitted [4]. However in that case, for the decoder to be successful without any motion vector correction, the encoder should send very accurate motion vectors.

In order to be standard compliant with low-complexity constraint, received motion vectors should be used. In this paper, we propose a standard compliant MC-FRUC algorithm with decoder motion processing as shown in Fig. 3. The proposed SC-FRUC scheme uses a novel motion vector processing technique to generate the skipped frames at the decoder. Next section presents the proposed motion smoothing algorithm.

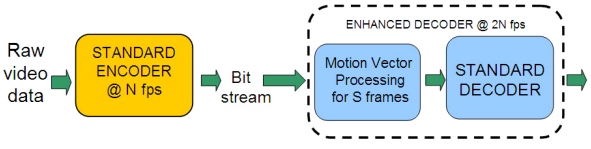


Figure 3: Standard compliant FRUC with motion vector processing

3. SMOOTH MOTION VECTOR PROCESSING AND ASSIGNMENT

Using the co-located motion vectors which are estimated between P_1 and P_3 directly for interpolation of S frame does not provide the best image quality. When the motion vectors received for P_3 (see Fig. 1) are halved for 1:2 up conversion, the motion vector field for S frame can be formed. However, some locations in motion vector field for S frame will be left uncovered, covered once, or covered twice as illustrated by the white areas, lighter gray areas and darker gray areas respectively as illustrated in Fig. 4 (i). The proposed algorithm not only assigns motion vectors for the uncovered regions, but also corrects the motion vectors of already assigned areas and replaces them with smoother ones.

We write the general smoothness measure by $(v) = {}_i w_i \ {}_i(v)$, where $i = 1...k$ represents various directions such as north, east, diagonal, center and w_i represents the weights of smoothness in each direction. Let ${}_{ver}$ denote the vertical smoothness, then ${}_{ver}$ can be written as follows.

$$ {}_{ver} = \sum_{c=0}^{C-1}\sum_{r=0}^{R-1} (v_{i+r,j+c} - v_{i+r+1,j+c})^2 + \sum_{c=0}^{C-1} (b_{i-1,j+c} - v_{i,j+c})^2 + (v_{i+R-1,j+c} - b_{i+R,j+c})^2. \quad (2) $$

In (2), $b_{i,j}$ represents the received motion vector information that is available in the neighborhood of the motion vector that we want to interpolate and $v_{i,j}$'s are the unknown motion vectors. R and C represents the number of motion vectors in one column and one row of a block. $R \times C$ gives the total number of motion vector found for a block. If the received motion vector corresponds to a block size of $P \times Q$, then the size of sub-blocks that we assign motion vector can be found by P/R and Q/C. Referring to Fig. 4 (ii), motion vectors v_1, v_2, v_3 and v_4 are unknown motion vectors, which are denoted as $v_{i,j}$ in (2), whereas the known motion vectors (i.e., b_i's) are v_N, v_S, v_E, v_W, and v_D. The parameters that correspond to Fig. 4 are R=C=2, $b_{i-1,j+c}$=v_N for c=0 and 1, $v_{i,j}$=v_1, $v_{i,j+1}$=v_2 and so on. For covered and multiple-covered areas as shown in Fig. 4 (i), the known motion vectors $b_{i,j}$'s can be found by methods like averaging or maximum overlap. Combining the smoothness measures in all directions in similar fashion to (2) yields

$$ (v) = \sum_{i=1}^{k} w_i \|A_i v - b_i\|^2. \quad (3) $$

The matrix A_i depends on the directions of the smoothness. The vector b_i contain the received motion vector information in the corresponding direction that we impose smoothness. Each component of (v) in (3) can be expanded as

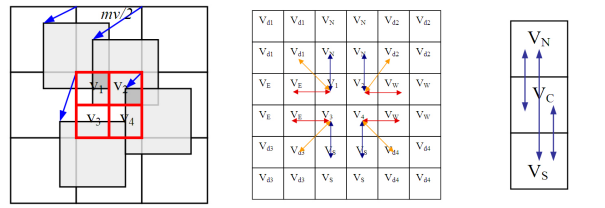
$$ \|A_i v - b_i\|^2 = v^T A_i^T A_i v - 2 b_i^T v + b_i^T b_i. \quad (4) $$

After writing the weighting matrices (A_i's) in the other directions similarly and substituting them in (3), the overall smoothness function becomes

$$ (v) = v^T A v - 2 b^T v + c \quad (5) $$

where A, b and c are

$$ A = \sum_{i=1}^{k} A_i^T A_i, \quad b = \sum_{i=1}^{k} A_i^T b_i, \text{and } c = \sum_{i=1}^{k} b_i^T b_i. \quad (6) $$



(i)Motion vector assignment (ii) Smoothing directions (iii) Vector pairs

Figure 4: Motion vector smoothing and assignment

The optimal solution is obtained by minimizing (v) as follows

$$ \frac{\partial (v)}{\partial v} = 2(Av_{opt} - b) = 0 \quad \Rightarrow \quad v_{opt} = A^{-1} b. \quad (7) $$

The weights can be found by using the relative orientation of the vectors along each of these axes as shown in Fig. 4 (i). The normalized dot product is computed between each of the three pairs of vectors along each direction. The correlation between v_C and each of its 8-neighbors given by the normalized dot product is directly used as a weight to minimize the corresponding term in the cost function. Fig. 4 (iii) illustrates calculation of weights for the vertical direction. For example weight in vertical direction i.e., w_v is obtained by using the dot product pairs between v_N, v_C and v_S. Let a_1, a_2, a_3 be the normalized dot products for a particular direction, (for example vertical direction in Fig. 4 (iii), then the weight $w_V = w^2$ in that direction is found by $w = (a_1 + a_2 + a_3)/3$ if $(a_1 + a_2 + a_3) > 0$ or 0 elsewhere. If the sum of the normalized dot product is negative the vectors along this axis are completely uncorrelated and should not be smoothed. As a special case, when one of the motion vectors $\mathbf{v}$ is zero, then the weight assigned to that vector is $w = \max[(1 - \frac{\|\mathbf{v}\|}{th}), 0]$, with threshold th. Threshold is chosen as $1/3^{rd}$ of the motion vector search range in the simulations.

4. EXPERIMENTAL RESULTS

The performance of the proposed motion smoothing algorithm for frame rate conversion is tested on several QCIF size video sequences such as Salesman, Carphone, Highway, Stefan, Table Tennis and so on. Full search motion estimation with 16x16 pixels block size and ± 15 pixels search range is used in simulations. Every other frame is skipped during the encoding process and they are interpolated using motion-compensated interpolation by using various motion processing methods. Although the input motion vectors are found for 16×16 block sizes, the output motion vectors of the motion processing can be assigned to 4×4 blocks or to units as small as pixels with the proposed method. However, in the case where motion vectors are assigned to each pixel, the motion vector values will differ from their neighbors in the 3^{th} or 4^{th} fractional digit. Motion compensation with more than quarter pixel accuracy will consume high power and will not affect the visual quality much. We restrict ourselves to low complexity algorithms and at most half pixel motion compensation therefore, we have used 4x4 as the smallest block size. In order to obtain block size 4x4 from 16x16 motion vector field, the same smoothing procedure is repeated on the obtained 8x8 block size from first smoothing pass. The proposed algorithm is compared to motion

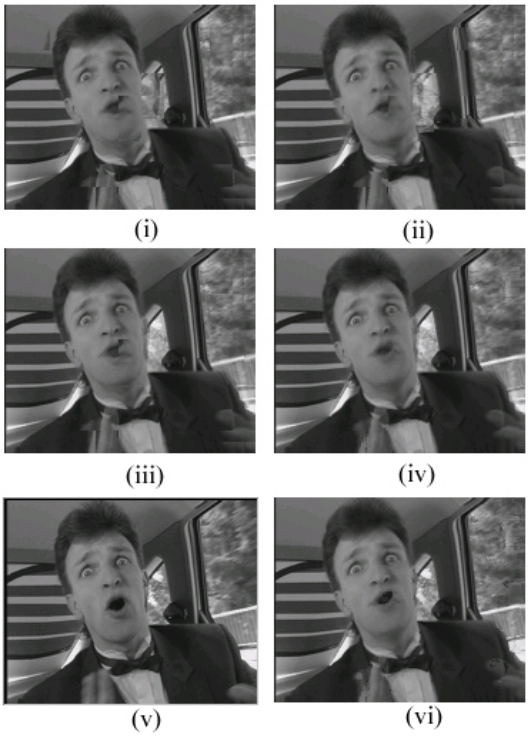


Figure 5: Visual comparison of motion processing methods: (i) No processing-21.5491dB, (ii) Mean-21.7894dB, (iii) Median-22.7894dB, (iv) M6-22.21dB, (v) Original image, (vi) Blur reduction with adaptive weights-21.62dB

compensated interpolation (MCI) cascaded with motion processing methods such as no processing (M1), median (M2), smoothing (M3), weighted smoothing (M4), double smoothing 1 (M5) i.e., cascade of M4 and M3, double smoothing 2 (M6) i.e, cascade of M3 and M3. The PSNR results are averaged over 100 frames and they are given in Table I. The first column represents the PSNR result of MCI with no processing, and the other columns represent the PSNR improvement over first column. Double smoothing with cascaded weighted smoothing and smoothing provides the best PSNR results. Although the PSNR enhancement obtained by median processing is close to the proposed algorithm, the visual enhancement in the proposed algorithm is superior compared to other methods. Fig. 5 demonstrates the visual comparison of the 99^{th} frame in the Carphone sequence. Using motion vectors without any processing for interpolating the missing motion vectors results in the worst quality. Mean and median methods perform better compared to no processing, however there is still some artifacts in the mouth area. The proposed algorithm provides better visual quality by providing smooth transition in the boundaries. It also yields higher PSNR value. The only sequence which all the algorithms gave lower PSNR results compared to no-processing is Table Tennis sequence, where there is more rapid and global motion. However, the visual quality of the frames are still comparable.

The proposed algorithm eliminates almost all the blocking artifacts due to wrong motion vectors, and most of the blurring artifacts. When the motion is fast, or objects start entering or leaving the scene, blur artifact is introduced. However, this is a problem of all MC-interpolation algorithms. One approach to determine blur is to calculate the difference between forward and backward motion compensated blocks. If the difference is above a threshold the filter weights of the forward or backward motion compensated frames can be adjusted. This blur reduction method by adapting filter weights is based on our previous work in [1]. Fig. 5 (vi) shows the resulting image, where the interpolation filter weight is adjusted based on adaptive weighting. Even though the PSNR is not as high as M6, adjusting interpolation filter taps for blur reduction provides good visual results.

Table 1: Average PSNRs (dB) of interpolated frames

	M1	M2	M3	M4	M5	M6
Mother-daughter	38.98	0.34	0.41	0.41	0.47	0.43
Silent	32.14	0.49	0.57	0.55	0.56	0.57
Salesman	39.21	0.12	0.16	0.16	0.16	0.17
Stefan	23.12	0.12	0.08	0.17	0.08	0.04
Carphone	30.65	0.16	0.25	0.25	0.33	0.31
Highway	32.69	-0.1	0.03	0.03	0.07	0.06
Mobile	29.49	0.85	0.86	0.83	0.85	0.86
Table	27.16	-0.09	-0.22	-0.11	-0.19	-0.25

5. CONCLUSIONS

In order to make SC-FRUC an acceptable application for the end-users, we need to simultaneously reduce blocking, blurriness, flickering and motion jerkiness artifacts. By constraining ourselves to low-power, decoder-only applications, we have focused on motion vector processing at the decoder rather than performing a new motion estimation algorithm. By using the described smooth proposing techniques, it is possible to eliminate some of the artifacts such as blocking and blurring. However, just by processing motion vectors and by using averaging interpolation filter blur artifact may still remain in the fast moving areas of the video, or on locations where an object does not appear in either of the reference or future frame. In that case adaptive weighting of the motion compensated frames eliminates some of the blurring problems. The presented motion smoothing algorithm can also be used in other applications such as scalable video coding or transcoding.

REFERENCES

[1] G. Dane, and T. Q. Nguyen, "Analysis of motion vector errors for frame rate up conversion," in *Proc. of Asilomar Conference in Signals, Systems and Computers*, Nov. 2003.

[2] C. Cafforio, F. Rocca, and S. Tubaro, "Motion-compensated image interpolation," *IEEE Trans. on Communications*, vol.38, no.2, pp. 215-221, 1990.

[3] G. de Haan, *Video Processing for Multimedia Systems*, University Press Eindhoven, 2000.

[4] F. J. de Brujin, W. H. A. Bruls, D. Brazerovic, and G. de Haan,"Efficient video coding integrating MPEG-2 and picture-rate conversion," *IEEE Trans. on Consumer Electronics*, vol. 48, no. 3, pp. 688-693, Aug. 2002.

[5] G. Dane and T. Nguyen, "Smooth motion vector resampling for standard compatible video post-processing," *Proc. of Asilomar Conference on Signals, Systems and Computers*, Nov. 2004.