28th Picture Coding Symposium, PCS2010, December 8-10, 2010, Nagoya, Japan

A NOVEL INLOOP FILTER FOR VIDEO-COMPRESSION BASED ON TEMPORAL PIXEL TRAJECTORIES

Marko Esche, Andreas Krutz, Alexander Glantz, Thomas Sikora

Communication Systems Group, Technische Universität Berlin
Sekr. EN1, Einsteinufer 17, D-10587 Berlin, GERMANY
{esche, krutz, glantz, sikora}@nue.tu-berlin.de

ABSTRACT

The objective of this work is to investigate the performance of a new inloop filter for video compression, which uses temporal rather than spatial information to improve the quality of reference frames used for prediction. The new filter has been integrated into the H.264/AVC baseline encoder and tested on a wide range of sequences. Experimental results show that the filter achieves a bit rate reduction of up to 12% and more than 4% on average without increasing the complexity of either encoder or decoder significantly.

Index Terms— video-compression, temporal inloop filter, pixel trajectories, deblocking

1. INTRODUCTION

In block-based motion compensated video codecs such as the H.264/AVC there exist two main blocks that introduce noise at the encoder. One of these is the motion compensation itself, the other is the quantization of block-based integer discrete cosine transform (DCT) coefficients. Inloop filters such as the deblocking filter by List et al. described in [1] have been shown to improve both the subjective and the objective quality of the decoded video. Although block artifacts are introduced by transferring blocks of pixels from reference frames into the current one, most inloop filters only make use of spatial information from the current frame itself. In this paper a novel inloop filter based on temporal pixel trajectories is proposed that compensates this deficiency. A pixel trajectory can be defined as 2D-locations through which a certain image point moves from frame to frame. It will be shown that the temporal trajectory filter (TTF) performs well when integrated into the H.264/AVC baseline profile, providing significant bit rate savings for a wide range of sequences. The concept of motion compensated temporal filtering was first introduced in [2]. The notion of a motion trajectory and its applications to video coding were first described in [3], which was extended to a temporal filtering approach in [4]. Approaches with a similar aim have been presented in [5] and [6], although neither fully exploits the potential of identifying each pixel with a motion of its own. Section 2 gives mathematically sound definitions of the pixel trajectory and associated quantities as well as providing some theoretical background. Section 3 introduces criteria to determine optimum filter characteristics and shows how to reliably reconstruct the pixel trajectory from H.264/AVC motion vectors. In section 4 details are given concerning the implementation and the observed PSNR gains and bit rate savings for a number of commonly used test sequences. Section 5 summarizes the paper and discusses future work.

2. DEFINITIONS AND THEORETICAL BACKGROUND

Our challenge is to improve the RD performance of the H.264/AVC codec. For this purpose we reduce the remaining coding noise in the reconstructed frames at encoder and decoder. Since these reconstructed frames are successively used for prediction of future frames, every degree of noise reduction will increase the performance of the codec. Our strategy involves identification and tracking of identical image content over the past coded and noisy frames. For each pixel in frame F_i an individual motion trajectory is constructed that identifies N pixel amplitudes in the past N frames. We assume that image content in the pixels identified along the motion trajectory does not change and that any deviation of these pixel amplitudes is caused by additive independent (white) coding noise. Statistical signal theory implies that averaging the N pixel amplitudes along the trajectory potentially reduces the noise variance by a factor of N, resulting in a coding gain. Throughout the remainder of the paper $Y_i(x, y)$, $U_i(x, y)$ and $V_i(x, y)$ shall denote the luminance and chrominance components of the i-th frame of a video sequence at pixel position $(x, y)^T$ respectively. Since in H.264/AVC motion vectors are assigned not to individual pixels but rather to blocks of various sizes, all pixels within such a block are assumed to have identical motion, resulting in a field of motion vectors with components $dx_i(x, y)$ and $dy_i(x, y)$. These give the motion vector needed to retrieve the pixel $(x, y)^T$ in frame F_i from

978-1-4244-7133-1/10/$26.00 ©2010 IEEE

its respective reference frame. In this paper we consider the H.264 baseline profile with an IPPP-structure, thus there exists always only one reference frame, which is the previously encoded frame F_{i-1}. Starting with frame F_i currently to be encoded the trajectory is retrieved for a pixel $(x_0, y_0)^T$ inside that frame. The first luminance and chrominance components along the trajectory Y_0^t, U_0^t and V_0^t are identical to the current image values $Y_i(x_0, y_0)$, $U_i(x_0, y_0)$ and $V_i(x_0, y_0)$. The location occupied by the pixel $(x_0, y_0)^T$ in the previous frame F_{i-1} can be calculated by adding the motion vector associated with $(x_0, y_0)^T$ to that location as given in

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_i(x_0, y_0) \\ dy_i(x_0, y_0) \end{pmatrix}. \quad (1)$$

Since $(x_1, y_1)^T$ generally points not to an integer pixel position but rather has quarterpel resolution, a rounding operation is necessary to retrieve the motion vector from the next motion vector field. The resulting pixel location in frame F_{i-2} is subsequently given by

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} dx_{i-1}(\lfloor x_1 \rfloor, \lfloor y_1 \rfloor) \\ dy_{i-1}(\lfloor x_1 \rfloor, \lfloor y_1 \rfloor) \end{pmatrix}. \quad (2)$$

The associated luma component is $Y_2^t = Y_{i-2}(x_2, y_2)$. It is thus possible to formulate a general recursive formula for determining the j-th position of a pixel from frame F_i along the trajectory for any given frame F_j as notated in equation 3.

$$\begin{pmatrix} x_j \\ y_j \end{pmatrix} = \begin{pmatrix} x_{j-1} \\ y_{j-1} \end{pmatrix} + \begin{pmatrix} dx_{i-j}(\lfloor x_{j-1} \rfloor, \lfloor y_{j-1} \rfloor) \\ dy_{i-j}(\lfloor x_{j-1} \rfloor, \lfloor y_{j-1} \rfloor) \end{pmatrix} \quad (3)$$

Equivalently the trajectory's j-th luminance sample is $Y_j^t = Y_{i-j}(x_j, y_j)$, the same holds for both U- and V-channels, too.

3. ADAPTIVE FILTER DESIGN AND CALCULATION OF PIXEL TRAJECTORIES

The averaging along the pixel trajectory as described above can be integrated into the H.264 encoder depicted in figure 1. Firstly, all predicted frames that have not yet been deblocked are stored in a seperate queue. Additionally, all motion vectors that have been used to predict these frames are also stored. In order to minimize memory usage only eight past frames and motion vector fields are kept. The new inloop filter is inserted in the encoder in front of the List filter, which filters according to remaining block edges and does not reverse TTF-induced improvements. For every pixel inside a frame to be deblocked a trajectory is calculated as described in section 2. The luma components, that have to be obtained with quarterpel accuracy according to equation 3, are calculated using the standard H.264 interpolation filter. For the chroma components an eighthpel interpolation is used equivalently. This results in the formation of a list of luminance values $Y_0^t, ..., Y_7^t$ and two lists of chrominance values for every pixel in the current frame. Before applying an averaging operation along the trajectory the validity of the predicted trajectory has to be verified. Basically, there are two indicators for a badly predicted trajectory.

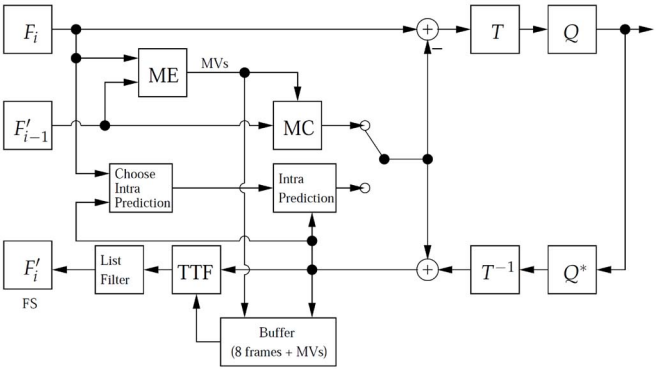

Fig. 1. H.264/AVC encoder with integrated TTF filter in front of the inloop deblocking filter. F_i and F'_i are the original frame and the respective locally reconstructed frame.

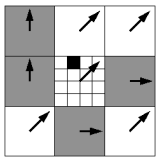

Fig. 2. The BV metric gives the number of 4×4 blocks surrounding the trajectories current location, whose motion differs from the block through which the trajectory passes. The location of the trajectory is marked in black, all blocks with a different motion are grayed out.

3.1. Absolute error along the trajectory

A sudden change of the luminance value $\Delta Y_j = |Y_{j+1}^t - Y_j^t|$ can indicate that a retrieved motion vector no longer correctly describes a pixel's trajectory. This is for instance the case when foreground objects are partly inside a background block thus interrupting the trajectory. In order to avoid the inclusion of wrong luminance values the trajectory is only continued, if $\Delta Y_j \leq T_Y$ for a given threshold T_Y. The derivation of an optimal T_Y per frame will be described in subsection 3.4.

3.2. Inconsistent motion vectors

Another indicator for a badly predicted trajectory are strong variations in the motion vector field. During the retrieval of the motion vector for a specified pixel as given in equation 3, the retrieved motion vector is compared against the vectors for the eight neighboring 4×4 blocks as illustrated by figure 2. The block vote metric $BV_j(x_j, y_j)$ denotes the number of 4×4 blocks surrounding location $(x_j, y_j)^T$ whose motion vectors differ from the one in which $(x_j, y_j)^T$ lies. In figure 2 the $BV_j(x_j, y_j)$ value is 4. Based on this metric a second threshold T_{BV} can be used to control the length of the predicted trajectory based on the confidence in the motion vector

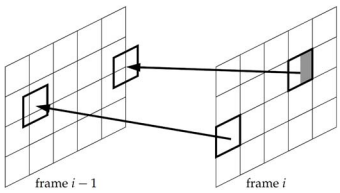

Fig. 3. In cases where a block in frame i is predicted from a block that lies partly outside of frame $i-1$ a number of pixels have no valid trajectory. All pixels whose trajectories are interrupted in frame i are marked in gray.

accuracy as given in

$$BV_j(x_j, y_j) \leq 8 - T_{BV}. \tag{4}$$

The BV metric effectively checks for every pixel along the trajectory if a number of surrounding 4×4 blocks has identical motion to the current trajectory's motion. As an example $T_{BV} = 0$ would not restrict the motion at all, while $T_{BV} = 8$ only continues a trajectory if all surrounding blocks have identical motion. In this context 4×4 blocks lying outside the frame are not considered at all, so that blocks at the edge of the frame are only checked against a smaller number of reference blocks. *Intra*-coded blocks are assumed to have zero motion.

3.3. Other interruptions of the trajectory

There are two further cases in which a trajectory is interrupted: The first being *intra*-coded blocks. Should a trajectory reach such a block, which has no associated motion vector, it is terminated. The second case occurs when the predicted block retrieved from the previous frame lies partly outside the frame as shown in figure 3. In this case no motion information is available for a number of pixels whose trajectory is also interrupted.

3.4. Filter optimization

Both thresholds introduced above are now optimized for each frame. For all possible combinations of $1 \leq T_Y \leq 8$ and $0 \leq T_{BV} \leq 4$ the trajectories for each pixel in the frame to be deblocked are calculated and the luma and chroma components of every pixel are then replaced by the average along its respective trajectory of length $K \leq 8$ as given by

$$\begin{aligned} Y_{opt} &= \tfrac{1}{K} \textstyle\sum_{j=0}^{K-1} Y_j^t \\ U_{opt} &= \tfrac{1}{K} \textstyle\sum_{j=0}^{K-1} U_j^t \\ V_{opt} &= \tfrac{1}{K} \textstyle\sum_{j=0}^{K-1} V_j^t. \end{aligned} \tag{5}$$

The current implementation only applies the filter to the luma component, leaving both chroma components in their original state. Each combination can now be assigned a mean squared error (MSE) compared to the original frame to be encoded.

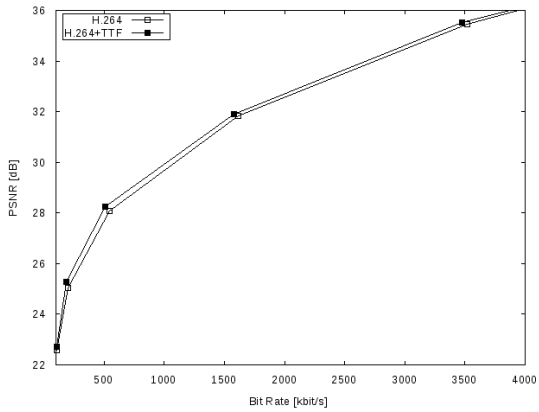

Fig. 4. PSNR vs. bit rate for the *BQSquare* sequence, 416×240, $60Hz$, 600 frames (QP from 25 to 45)

The optimum thresholds yielding the minimum MSE are then chosen and encoded in the bitstream requiring five additional bits per frame. Instead of computing all 32 different versions of the filtered frame it is easily possible to calculate all MSEs simultaneously only having to process every pixel in the current frame once. The first advantage of this approach is that no additional side information is required on the macroblock level. Secondly, every pixel is filtered with an individual filter length which helps to break up blocking artifacts.

4. EXPERIMENTAL EVALUATION

The proposed filter has been implemented in *C* and integrated into the *JM 11 (KTA 2.2)* software using the H.264/AVC baseline profile. The resulting bit rates and PSNR values are compared against the same software without the additional inloop filter, which behaves exactly as H.264/AVC. The implementation has been tested on a large variety of sequences including five sequences in TV resolution and 15 MPEG test sequences. Examplary RD-curves for the *BQSquare* sequence are shown in figure 4. From the underlying data it becomes instantly

Sequence	BD-rate	Δ_{PSNR}	Resolution	frames
Allstars	-2.49%	$+0.09\,\mathrm{dB}$	704x576	250
Basketball	-2.76%	$+0.11\,\mathrm{dB}$	1024x576	300
BBC-Pan-13	-7.62%	$+0.42\,\mathrm{dB}$	720x576	110
Desert	-9.13%	$+0.44\,\mathrm{dB}$	720x400	240
Entertainment	-5.86%	$+0.25\,\mathrm{dB}$	720x576	250
Stanford	-3.29%	$+0.10\,\mathrm{dB}$	720x480	304

Table 1. BD-rate and average PSNR gain for all TV sequences (QP from 20 to 40)

apparent, that the filter is less effective for high QPs. This is due to the fact that at very low quality reference blocks themselves are of such poor quality that averaging along the trajectory cannot significantly improve the current frame. In the final implementation the filter was therefore switched off au-

Fig. 5. A segment from frame 160 of the *BQTerrace* sequence, *left:* H.264/AVC at 3.596MBit/s (31.14 dB), *right:* H.264/AVC+TTF at 3.203MBit/s (31.32 dB). Especially in the fence more detail is present in the TTF filtered version.

tomatically for QPs higher than 45. For all sequences average PSNR gain and bit rate reduction were measured using the Bjøntegaard metric [7]. The average values are given in tables 1 and 2. These values show, that the filter performs well for sequences of resolutions ranging from 416×240 (*BQSquare*) to 2560×1600 (*Traffic*). Additionally, the actual frame rate also seems to have no significant impact on the resulting quality, as significant bit rate reduction is present both for sequences with $24Hz$ (*ParkScene*) and with $60Hz$ frame rates (*BQTerrace*). Since the filter makes a pixel-wise decision

Sequence	BD-rate	Δ_{PSNR}	Resolution	Frames
BQMall	−3.15%	+0.15 dB	832x480	600
BQSquare	−7.73%	+0.29 dB	416x240	600
BQTerrace	−12.46%	+0.31 dB	1920x1080	600
Cactus	−0.12%	0.00 dB	1920x1080	500
Kimono1	−2.81%	+0.12 dB	1920x1080	240
ParkJoy	−2.67%	+0.09 dB	2560x1600	250
ParkScene	−7.15%	+0.29 dB	1920x1080	240
PartyScene	−4.73%	+0.19 dB	832x480	500
PeopleOnStreet	−0.28%	+0.01 dB	2560x1600	150
RaceHorses	−1.73%	+0.07 dB	832x480	300
Traffic	−4.11%	+0.16 dB	2560x1600	300
vidyo1	−1.25%	+0.04 dB	1280x720	600
vidyo3	−3.98%	+0.16 dB	1280x720	600
vidyo4	−2.59%	+0.11 dB	1280x720	600

Table 2. BD-rate and average PSNR gain for all MPEG sequences (QP from 20 to 40)

concerning the filter length, the visual quality of the decoded bit stream is also improved as is illustrated by the segments from the *BQTerrace* sequence shown in figure 5. The performance gain illustrated here, however, still does not explore the full potential of the proposed filter. The MSE minimization is, for instance, not necessarily the best criterion for determining the transmitted thresholds. Ideally, an RD-optimization scheme should be adopted to derive both T_Y and T_{BV} for each frame. This is especially important since there is no linear dependency between the transmitted error residual and the amount of bits needed to encode it. Equivalently, maximizing the MSE of any reference picture does not result in a minimization of the total bit rate for the entire video sequence. Nevertheless, the proposed algorithm has been shown to improve the compression efficiency of the H.264/AVC baseline profile for all tested sequences fully justifying the introduced overhead of five bits per frame. Exemplary decoded videos and their counterparts produced by H.264/AVC with the associated RD curves can be found on the accompanying website `www.nue.tu-berlin.de/research/ttf`.

5. SUMMARY AND FUTURE WORK

The main objective of this work was to investigate a new in-loop filtering approach that uses temporal information in the form of pixel trajectories to improve the quality of reference pictures used in H.264/AVC. The algorithm based on individual temporal pixel trajectories as described in this paper provides bit rate savings for all tested sequences, reducing the bit rate by more than 4% on average. This is achieved without significantly increasing computational complexity or memory usage of the encoder. Moreover, even better results could be achieved by adopting an RD-optimization scheme. Future work will therefore focus on including RD-optimization at the encoder. In addition, the algorithm will be extended to work with hierarchical B- and P-frames also.

6. REFERENCES

[1] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2003.

[2] E. Dubois and S. Sabri, "Noise reduction in image sequences using motion-compensated temporal filtering," *IEEE Transactions on Communications*, vol. 32, no. 7, pp. 826– 831, July 1984.

[3] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, September 1994.

[4] X. Wang, M. Karczewicz, J. Ridge, and Y. Bao, "Simplified update step of motion compensated temporal filtering for video coding," in *Proceedings of the 24th Picture Coding Symposium*, 2006, pp. 826–831.

[5] D. T. Vo and T. Q. Nguyen, "Optimal motion compensated spatio-temporal filter for quality enhancement of H.264/AVC compressed video sequences," in *Proceedings of the 16th International Conference on Image Processing (ICIP)*, November 2009, pp. 3173–3176.

[6] A. Glantz, A. Krutz, M. Haller, and T. Sikora, "Video coding using global motion temporal filtering," in *ICIP*, November 2009, pp. 1053–1056.

[7] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *ITU-T SG16/Q.6 VCEG document VCEG-M33*, Mar 2001.