TEMPORAL TRAJECTORY FILTERING FOR BI-DIRECTIONAL PREDICTED FRAMES

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

In this work the application of a temporal in-loop filtering approach for B-frames in video compression based on the Temporal Trajectory Filter (TTF) is investigated. The TTF constructs temporal pixel trajectories for individual image points in the P-frames of a video sequence, which can be utilized to improve the quality of the reconstructed frames used for prediction. It is shown, how this concept can be adapted to B-frames despite the fact that these already use temporal motion information to a great extent through the flexible choice of reference frames and prediction modes. The proposed filter has been integrated into the H.264/AVC encoder using the extended profile with hierarchical B-frames and was tested on a wide range of sequences. The filter produces bit rate reductions of up to -4% with an average of -1.6% over all tested sequences while also improving the subjective quality of the decoded video.

Index Terms— Video compression, temporal in-loop filter, pixel trajectories, B-frames

1. INTRODUCTION

Noise reduction in reconstructed frames of a video sequence has recently received a lot of attention, especially in the context of the Wiener-based adaptive loop filter (ALF), which was first described in [1]. In the state-of-the-art video codec H.264/AVC the deblocking filter, as proposed by List et. al in [2], serves the dual purpose of improving the subjective quality of the decoded frames and reducing the bit rate of the encoded video by improving the accuracy of the motion-compensation carried out on the reconstructed frames. However, both the ALF and the Deblocking Filter use spatial information only, i.e. they both lack temporal consistency, which especially for the Deblocking Filter can introduce new flickering artefacts. An attempt to add a temporal component to the in-loop filter was described in [3], where a global motion model was used for this purpose. The TTF introduced in [4], however, tries to overcome this particular problem through the formation of individual temporal trajectories for each pixel, which are used to reduce noise in the decoded frame. This new adaptive filter has been shown to produce bit rate savings of 5% on average over a large variety of sequences for the H.264/AVC baseline profile. The concept of temporal pixel trajectories was also used in [5]. In this paper it is shown, how the concept of pixel trajectories can be extended, so that it becomes applicable to the more general case of hierarchical B-frames, too, and how the resulting filter design needs to be adapted. The remainder of this paper is structured as follows: Section 2 briefly revisits the concept of temporal pixel trajectories and shows, how these can be used for the denoising of bi-directional predicted frames. A method for calculating suitable trajectories based on the motion vectors transmitted in the H.264/AVC extended profile bit stream is described in Section 3. Section 4 provides details on an implementation of the TTF and gives observed bit rate savings and PSNR gains. The paper is completed by a short summary and an outline of future work.

2. IN-LOOP FILTERING USING PIXEL TRAJECTORIES

In the H.264/AVC baseline profile with an IPPP coding structure every pixel at location $(x_i, y_i)^T$ in reconstructed frame number i is associated with a motion vector $(dx_i, dy_i)^T$. The corresponding image position in the previous frame $i-1$ is given by

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

where $dx_i(x, y)$ and $dy_i(x, y)$ are the x- and y-component of the motion vector field of frame i. Through concatenation of these motion vectors, a trajectory can be formed, whose j-th pixel along the trajectory is subsequently given by

$$\begin{pmatrix} x_j \\ y_j \end{pmatrix} = \begin{pmatrix} x_{j-1} \\ y_{j-1} \end{pmatrix} + \begin{pmatrix} dx_{i-j}(\lfloor x_{j-1} \rfloor, \lfloor y_{j-1} \rfloor) \\ dy_{i-j}(\lfloor x_{j-1} \rfloor, \lfloor y_{j-1} \rfloor) \end{pmatrix}. \quad (2)$$

Having calculated the corresponding image positions over the last n frames for a given pixel $(x_0, y_0)^T$ in the current frame, an associated luminance component for each of these positions can be obtained. This results in the formation of a list of previous luminance components $Y_0^t, ..., Y_{n-1}^t$ for every pixel in the current frame i, where Y_0^t is identical to the luminance component at location $(x_0, y_0)^T$. We assume that image content in the pixels along the trajectory does not change and that

any deviation of the pixel amplitudes is caused by additive independent white noise. Denoising can, therefore, be achieved by computing the average of all Y-components along the trajectory

$$Y_{opt} = \frac{1}{N} \sum_{j=0}^{N-1} Y_j^t. \tag{3}$$

The reconstructed luma sample Y_0 in frame i is then replaced by its filtered version Y_{opt}.

When considering the H.264/AVC extended profile with hierarchical B-frames, at most two motion vectors per sample at location (x_0, y_0) in frame i are available: one for reference list 0 ($(dx_i^0, dy_i^0)^T$) and one for reference list 1 ($(dx_i^1, dy_i^1)^T$) respectively. The reference picture order numbers for both lists shall be denoted by $\text{ref}_i^0(x, y)$ and $\text{ref}_i^1(x, y)$. Starting from the current frame i the two reference locations in frames $\text{ref}_i^0(x, y)$ and $\text{ref}_i^1(x, y)$ are then given by

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_i^0(x_0, y_0) \\ dy_i^0(x_0, y_0) \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_i^1(x_0, y_0) \\ dy_i^1(x_0, y_0) \end{pmatrix}. \tag{4}$$

Both of these locations can potentially contribute to the list of noisy luma samples, yielding components Y_1^t and Y_2^t, where Y_0^t is the luma sample that is to be filtered in the current frame i. Since both trajectory locations in equation 4 are also potentially predicted from two reference frames each, an additional four luma samples can be added to the list by concatenating their respective motion vectors. For example the additional trajectory locations derived from pixel $(x_1, y_1)^T$ are given by

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} dx_{r0}^0(x_1, y_1) \\ dy_{r0}^0(x_1, y_1) \end{pmatrix}$$

$$\begin{pmatrix} x_4 \\ y_4 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} dx_{r1}^1(x_1, y_1) \\ dy_{r1}^1(x_1, y_1) \end{pmatrix}, \tag{5}$$

$$\text{with } r0 = \text{ref}_i^0(x_1, y_1), r1 = \text{ref}_i^1(x_1, y_1).$$

The resulting trajectory, which no longer follows a one-dimensional path, but rather branches into several individual trajectories at every new frame, is illustrated by Figure 1. As in the case of the I-frame on the left, a trajectory can now contain not just one but several pixels within a frame. Since only one of these can represent the true location of the tracked image point, methods are needed to stop the formation of the trajectory. This problem will be adressed in Section 3. One property that makes B-frames particularly suitable for trajectory filtering, is the following: In the case of an IPPP coding structure, a trajectory would be interrupted whenever a certain image point is not visible in the previous frame. In the case of B-frames an image point may even be tracked if it is not visible for a number of frames, since both past and future frames are part of the trajectory and several different

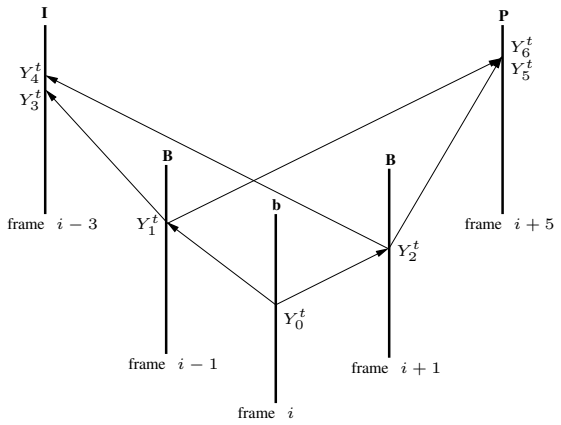

Fig. 1. Starting with a b-frame the trajectory for each pixel is computed through the concatenation of motion vectors yielding here a total of 7 luma samples along the trajectory.

trajectory paths may lead to the same location as illustrated in Figure 1. Nevertheless, the concatenation of motion vectors may not always result in true trajectories. Therefore, criteria are needed to determine, when a trajectory needs to be interrupted.

3. CONTROLLED TRAJECTORY FORMATION FOR B-FRAMES

In order to enable both encoder and decoder to perform in-loop trajectory filtering, a certain number of previous frames together with their associated motion vectors need to be stored. Since a hierarchical B-frame structure is considered, eight past B-frames with their motion vectors for list 0 and 1 as well as the last four I- and/or P-frames are stored. This ensures that even those B-frames, that have been directly predicted from a P-frame, can have a trajectory of a length greater then 1. To achieve good filtering results three thresholds are used to control the trajectories formation, the first of which has already been introduced and discussed in [4].

3.1. Absolute error along the trajectory

One indicator for a badly predicted trajectory can be a sudden change in the luminance component along the trajectory. Using the difference value $\Delta Y_j = |Y_{j+1}^t - Y_j^t|$ between two consecutive luma samples, a threshold

$$\Delta Y_j \leq T_Y \tag{6}$$

can be used to differentiate between true and false trajectories. Should the luminance difference be greater than a predefined threshold $0 \leq T_Y \leq 7$, then the trajectory for the current sample is interrupted and no further luma samples are added to the averaging process.

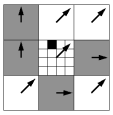

Fig. 2. The BV metric gives the number of 4×4 blocks surrounding the trajectories current location, whose motion differs from the block through which the trajectory passes. The location of the trajectory is marked in black, all blocks with a different motion are grayed out.

3.2. Spatial motion consistency

A second threshold, that was also introduced in [4], tests the spatial consistency of a motion vector that belongs to the trajectory. As depicted in Figure 2 the block vote metric $BV_j(x_j, y_j)$ calculates for any 4×4-block in frame j the number of neighboring blocks with identical motion. This metric can be interpreted as a measure of confidence for the accuracy of the current motion vector. The trajectory is consequently only continued, if

$$BV_j(x_j, y_j) \leq 8 - T_{BV}. \tag{7}$$

In the context of the H.264/AVC baseline profile with an IPPP coding structure this threshold yielded good results, when the TTF was applied before the standard Deblocking Filter. However, the metric favours the filtering of pixels within foreground objects or in the background, while pixels at object boundaries will usually violate the threshold in equation 9. In the implementation which will be described in Section 4, the TTF is instead applied after the deblocking filter to allow greater flexibility. To enable comparison between motion vectors that may even point to different reference frames, all motion vectors $(dx_i^0, dy_i^0)^T$ are scaled according to the reference frame they point to:

$$\begin{aligned}(dx_i^0(x,y))' &= \frac{dx_i^0(x,y)}{\text{ref}_i^0(x,y) - i} \\ (dy_i^0(x,y))' &= \frac{dy_i^0(x,y)}{\text{ref}_i^0(x,y) - i}\end{aligned} \tag{8}$$

Since blocking artefacts often manifest themselves at object boundaries, the trajectory is now only continued along a motion vector from list 0 when

$$BV_i^0(x,y) \geq T_{\text{BV}}, \tag{9}$$

where $BV_i^0(x,y)$ is the number of scaled neighboring motion vectors for the 4×4-block surrounding location x, y, whose x- or y-components differ from the current motion vector by a variance of more than $var_{\text{BV}} = 0.1$ or $var_{\text{BV}} = 0.5$. It is signaled in the bitstream, which of these two possible variances is used. The functionality of the spatial motion consistency is illustrated by Figure 3, which displays a section of the motion vector field for list 0. After the scaling of the motion vectors and with a predetermined exemplary threshold $T_{\text{BV}} = 3$ ($var_{\text{BV}} = 0.1$), the current pixel is identified as being part of an object boundary and the filtering along the current trajectory is continued.

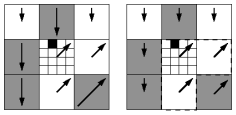

Fig. 3. *left:* The original motion vector field for list 0. All macroblocks marked in gray refer to a reference frame that has a greater temporal distance to the current frame than those for all other macroblocks. *right:* After the scaling of the motion vectors, the *BV* threshold identifies the four macroblocks in the lower left corner as belonging to the same object, while the remaining macroblocks are part of the background.

3.3. Temporal motion consistency

Since the block vote metric is now used to identify regions where the filtering needs to be applied, a different method is needed to identify motion vectors that correctly describe the true motion of an image point. To achieve this, the temporal consistency of the motion vectors along the trajectory is considered. Again, the motion vectors are scaled using equation 8. For any pixel point along the trajectory, the filtering is only continued, if the euclidean difference between the motion vector $(dx_i, dy_i)^T$ for list 0 pointing to the current trajectory location and the motion vectors $(dx_{r0}^0, dy_{r0}^0)^T$, $(dx_{r0}^1, dy_{r0}^1)^T$ pointing to the next possible locations satisfy the following threshold:

$$\sqrt{(dx_{r0}^0 - dx_i)^2 + (dy_{r0}^0 - dy_i)^2} < T_{\text{TC}}. \tag{10}$$

The same threshold $0 \leq T_{\text{TC}} \leq 7$ is also applied to all motion vectors for list 1.

3.4. Derivation of the filter parameters

At the encoder, all possible combinations of the three thresholds and all combinations with the two different motion vector variances are tested iteratively. Each of these combinations results in a different set of trajectories which also produces a different filtered picture. Effectively, all of these can be calculated simultaneously reducing the computational complexity to a single-pass process. The combination yielding the minimum MSE compared to the original frame is signaled in the bitstream using 10 additional bits per frame (3 bits per threshold and 1 bit for the motion vector variance).

4. EXPERIMENTAL EVALUATION

The TTF has been implemented in *C* and integrated into the H.264/AVC reference software *JM 16*. The resulting encoder with both Deblocking Filter and TTF is shown in Figure 4. The encoder settings are given in Table 1. Tests

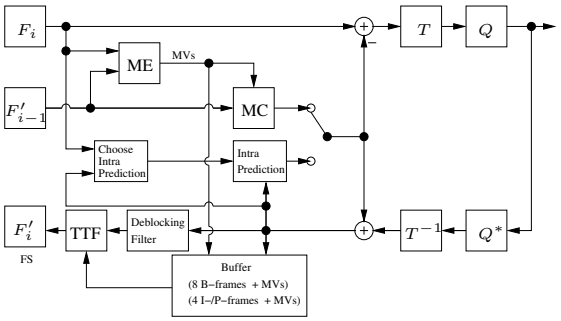

Fig. 4. The proposed filter is integrated into the local decoder loop at the encoder after the deblocking filter.

Prediction Structure	hierarchical B-frames
GOP size	8
RD Optimization	enabled
QPPSlice	QPISlice + 1
QPBSlice	QPPSlice + 2

Table 1. Settings used for the H.264/AVC Extended Profile.

have been carried out for QP 22 to 37 over a large variety of MPEG test sequences. For each of these, BD-rate and BD-gain have been calculated using the metric described in [6]. A list of all sequences and their respective measures can be found in Table 2. For sequences with large foreground

Sequence	BD-rate	Δ_{PSNR}	Resolution	Frames
BlowingBubbles	-0.99%	0.04 dB	416x240	500
BQSquare	-2.26%	0.08 dB	416x240	600
RaceHorses	-0.21%	0.01 dB	416x240	300
BQMall	-0.34%	0.02 dB	832x480	600
PartyScene	-1.28%	0.05 dB	832x480	600
RaceHorses	-0.29%	0.01 dB	832x480	300
Vidyo1	-2.59%	0.09 dB	1280x720	500
Vidyo3	-2.07%	0.08 dB	1280x720	500
Vidyo4	-1.63%	0.06 dB	1280x720	500
BasketballDrive	-0.38%	0.01 dB	1920x1080	500
BQTerrace	-3.89%	0.08 dB	1920x1080	600
Kimono1	-1.15%	0.04 dB	1920x1080	240
ParkScene	-2.39%	0.09 dB	1920x1080	240

Table 2. BD-rate and average PSNR gain for a variety of MPEG test sequences (QP from 22 to 37).

objects (both *RaceHorses* sequences) the bit rate reduction is around 0.3%. Nevertheless, the encoder performs better than H.264/AVC for all sequences. Most importantly, good results have been achieved for all HD-sequences. This is partly due to the fact that the introduced overhead has little impact at the data rates required for the transmission of HD content. Moreover, the motion vector accuracy for these sequences is generally better, since they were captured with frame rates of 50 or $60Hz$. Figure 5 shows the average trajectory lengths for the BQSquare sequence. IT also illustrates one advantage of the TTF compared to simple B-frames: For 40% of the pixels a trajectory with more than 2 luma samples is

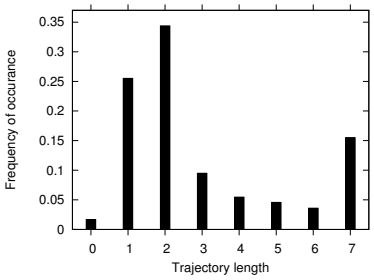

Fig. 5. Average distribution of the trajectory lengths for the BQSquare sequence (QP 27).

constructed, while a general B-frame averages two samples at most. Exemplary frames for all sequences, the decoded videos together with the associated RD-curves can be found on the accompanying website
`www.nue.tu-berlin.de/research/ttf-bframes`.

5. SUMMARY AND FUTURE WORK

The proposed filter has been shown to produce good gains especially for high resolution sequences resulting in an overall bit rate reduction of -1.6%. Due to the testing of various threshold combinations the complexity of the encoder is increased. Future work will therefore focus on deriving these thresholds directly from the video content. In addition, interactions between the deblocking filter and the TTF as well as a combination with the ALF will be studied.

6. REFERENCES

[1] S. Wittmann and T. Wedi, "Transmission of post-filter hints for video coding schemes," in *PCS*, September 2007, pp. 81–84.

[2] 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.

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

[4] M. Esche, A. Krutz, A. Glantz, and T. Sikora, "A novel in-loop filter for video-compression based on temporal pixel trajectories," in *PCS*, December 2010, pp. 514–517.

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

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