ADPATIVE DENSE VECTOR FIELD INTERPOLATION FOR TEMPORAL FILTERING

Marko Esche, Michael Tok, Thomas Sikora

Technische Universität Berlin Communication Systems Group,
Sekr. E-N 1, Einsteinufer 17
D-10587 Berlin, Germany
{esche,tok,sikora}@nue.tu-berlin.de

ABSTRACT

Inloop filters are a well known tool to improve the compression performance of hybrid video codecs. These filters generally work in the spatial domain. If temporal filters are used instead, their performance strongly depends on the available motion data. In this paper a new method to retrieve accurate motion information per pixel is introduced and evaluated within the HEVC test model HM 8.0. The interpolated motion vector field is significnatly closer to the real object motion than the block-based motion data. An average bit rate reduction of 0.4% is observed with a maximum reduction of 2.0%.

Index Terms— HEVC, Inloop Filter, Motion Estimation, Temporal Filtering

1. INTRODUCTION

During the standardization process of HEVC [1] several inloop filters were examined and investigated to suppress blocking artefacts and other noise introduced at low bit rates. Among them are the H.264/AVC deblocking filter [2], several versions of the Wiener-based Adpative Loop Filter (ALF) originating from [3], and the Sample Adaptive Offset Filter [4]. The well-known deblocking filter [2] only examines prediction block edges and performs horizontal and vertical smoothing operations to improve the quality of the reconstructed signal. The filter strength is chosen with respect to the prediction modes and reference frames of neighboring blocks. The ALF applies a two-dimensional filter kernel (either 5×5, 7×7 or 9×9 taps) to selected coding units of a slice. Where coding units roughly correspond to the prediction block structure from H.264/AVC. The ALF kernel is optimized based on the cross-correlation function between the initially reconstructed signal and the original frame. The SAO filter adds specific offset patterns to selected pixels based on a prior classification of the pixels into certain predefined categories which determine the filter parameters. All of these filters work on one individual frame at a time without taking into account the temporal nature of a video sequence. Since the visual difference between consecutive frames of a sequence is generally very small, temporal alignment of the image content and subsequent temporal filtering offer another option. The spatio-temporal filter described in [5] has been reported to produce an average PSNR gain of $0.56\,\mathrm{dB}$ for the *Foreman* test sequence at a given bit rate. This filter performs alignment on block-level only and thus requires accurate motion information per block. Techniques using global motion models [6] can produce even better results but are mainly only applicable to the image background. In order to perform filtering on the foreground as well, a filter was proposed in [7]. The Quadtree-based Temporal Trajectory Filter (QTTF) selectively concatenates motion vectors (MV) transmitted in the bit stream of the encoded video sequence to construct individual motion trajectories over several frames for each pixel. The filter is controlled by three thresholds which are signaled along with a quadtree structure to apply different parameter settings to different image regions. The filter performs better when more accurate motion information is available, as longer trajectories help to improve the filter performance. However, accurate motion information is mainly only available at very low QP values, where less artefacts occur. In this paper a method is described to derive accurate motion information at pixel level from block-based motion vectors for temporal filtering. Previous work in the area of dense motion interpolation [8] always required pixel-wise motion estimation on the raw video data which is of course not available at the decoder. The remainder of the paper is structured as follows: Section 2 briefly revisits the QTTF and describes its basic functionality. The dense motion field interpolation algorithm used here is detailed in Section 3. In Section 4 the algorithm is firstly evaluated based on a manually annotated sequence with groundtruth motion. Secondly, the compression performance of the improved filter is tested within the HEVC test model HM 8.0. Section 5 summarizes the paper.

2. TEMPORAL TRAJECTORY FILTERING

In order to construct accurate motion trajectories per image point over several frames, accurate motion information is re-

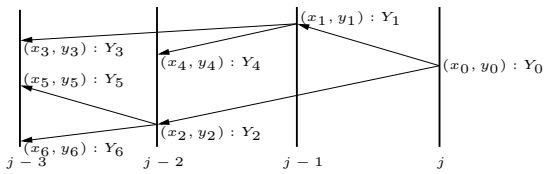

Fig. 1. The trajectory starting at an arbitrary pixel $(x_0, y_0)^T$ in frame j is split into two subtrajectories at every new B-frame. The number of samples that can potentially be used for filtering is thus increased exponentially.

quired both at encoder and decoder. A fast and relatively reliable way to obtain such information is to use the motion vectors transmitted in the bit stream. Even though these block-based motion vectors may only have been chosen due to rate-distortion optimization, they are generally good motion approximations for large portions of the associated square block of pixels. In this paper the QTTF is evaluated within the context of the HEVC main profile which uses an IBBB coding strucutre. In this setting each block has up to two motion vectors with different reference frames as shown in Figure 1. As a consequence the trajectory is split at every B-predicted pixel and can follow two paths. This enables the filter to track pixels even if they are hidden in some intermediate frames. In addition, much more image samples are hit by the trajectory and can thus be used for filtering. Let $(dx_{i,0}, dy_{i,0})^T$ and $(dx_{i,1}, dy_{i,1})^T$ be the motion vector fields for reference lists 0 and 1 for an arbitrary pixel $(x_0, y_0)^T$. Originally, these fields are derived by using the block motion vector per reference list for every quarter-pel position within the block. Then the next pixel locations along the trajectory are given by

$$\begin{aligned} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} &= \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_{i,0}(\lfloor x_0 \rfloor, \lfloor y_0 \rfloor) \\ dy_{i,0}(\lfloor x_0 \rfloor, \lfloor y_0 \rfloor) \end{pmatrix}, \\ \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} &= \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_{i,1}(\lfloor x_0 \rfloor, \lfloor y_0 \rfloor) \\ dy_{i,1}(\lfloor x_0 \rfloor, \lfloor y_0 \rfloor) \end{pmatrix}. \end{aligned} \tag{1}$$

As has been outlined above, not all calcualted trajectory locations do necessarily coincide with the true motion of a pixel. This is illustrated by pixel locations 3, 5, and 6 in Figure 1, which lie at different positions in the same frame. In order to distinguish between correct and false motion information the QTTF uses three criteria or thresholds.

2.1. Color Error Along the Trajectory

For every pixel along the trajectory the differences between the color components of the current pixel (Y_i, U_i, V_i) and the components of the previous pixel of the trajectory (Y_{pre}, U_{pre}, V_{pre}) are calculated yielding the absolute differences $\Delta Y_i = |Y_i - Y_{\text{pre}}|$, $\Delta U_i = |U_i - U_{\text{pre}}|$, $\Delta V_i = |V_i - V_{\text{pre}}|$. For pixels $(x_1, y_1)^T$ and $(x_2, y_2)^T$ the predecessor is for example pixel $(x_0, y_0)^T$. If all three differences are smaller than a threshold T, then the new luma component is included in the trajectory:

$$\Delta Y_i < T, \Delta U_i < T, V_i < T, T = \begin{cases} 2T_Y, \text{QP} < 30 \\ 4T_Y, \text{QP} \geq 30 \end{cases}, \tag{2}$$

with $0 \leq T_Y \leq 7$. Otherwise, the trajectory is continued nonetheless, but the current luminance value is ignored. This ensures, that relatively long trajectories are constructed, even if intermediate luma samples are significantly distorted. The differentiation between low and high QPs is motivated by the fact, that stronger artifacts will appear at higher QPs.

2.2. Temporal Motion Consistency

Escpecially at high frame rates pixel motion should change only slightly from frame to frame. Assuming linear motion, consecutive motion vectors should ideally be identical. In order to measure the similarity, every motion vector $(dx_i, dy_i)^T$ is scaled according to the temporal distance that it spans resulting in a motion vector $(dx'_i, dy'_i)^T$. For every new motion vector $(dx_{r0}, dy_{r0})^T$ from reference list 0 its Euclidean distance with its predecessor is measured.

$$dist = \sqrt{(dx_{r0} - dx_i)^2 + (dy_{r0} - dy_i)^2}. \tag{3}$$

The trajectory is only continued following the motion vector $(dx_{r0}, dy_{r0})^T$ from reference list 0, if $dist \leq T_{\text{TC}}$ for a given threshold T_{TC} between 0 and 7 in quarter-pel units.

2.3. Spatial Motion Consistency

Originally the spatial similarity within the transmitted motion vector field was also examined for low bit rates [7]. All vectors differing from their neighbors on 4×4 block level by a value greater than a threshold T_{SC} were also discarded. As of HM 8.0 the additional threshold does not yield any additional gain. Apparently, the motion information is now even more accurate. In the current implementation only the temporal motion consistency criterion and the color criterion are used.

2.4. Threshold Calculation and Filtering

Once a trajectory has correctly been identified, the noisy luminance component $\hat{Y}_j$ of the frame to be denoised is replaced by the weighted average of the noisy samples $\hat{Y}_i$ along the trajectory

$$Y_j^*(x_0, y_0) = \frac{1}{N} \sum_{i=0}^{N-1} \beta_i \cdot \hat{Y}_i(x_i, y_i), \tag{4}$$

where N is the number of samples referenced by the trajectory and β_i are the associated weights per frame as described in [7]. In a first implementation published in [9] the filter applied the same set of thresholds to an entire frame at a time.

At the encoder all threshold combinations are tested and the one yielding the minimum mean square error is signaled to the decoder. To extend the flexibility of the filter an optimal quadtree partitioning algorithm was included in [7]. In addition, an optimal weighting scheme of the samples was described. In [10] a scheme for reducing the quadtree overhead with appropriate CABAC context models was presented, reducing the side-information by 42% on average.

3. MOTION FIELD INTERPOLATION

In order to obtain more accurate motion information, an interpolation algorithm is now applied to the transmitted motion vector field. Firstly, all motion vectors are normalized according to their temporal distance, i.e. the number of frames that they span. It is assumed, that each motion vector accurately describes the center pixel of its Prediction Unit (PU), which corresponds to a motion compensated block in H.264/AVC. The closer an arbitrary pixel is to such a block center, the closer its actual motion will be to the block motion. Since pixels between blocks may have been falsely combined into a single block, they are interpreted as being influenced by their three closest neighboring block centers. To determine which pixel-based motion vectors are influenced by which blocks, a Delaunay triangulation is performed on the HEVC PU grid. An exemplary original block prediction structure and the triangulated mesh are shown in Figure 2. When calculating

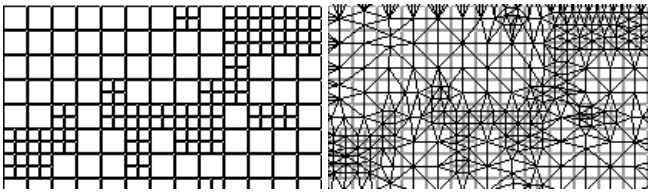

Fig. 2. *left:* Block prediction structure of the upper left part of frame 7 of the *BQSquare* sequence. *right:* Resulting triangle mesh after Delaunay triangulation.

the interpolated motion vector for any pixel $D = (x_D, y_D)^T$ within a frame, the following steps have to be carried out.

1. Determine the Triangle ABC within which D lies.
2. Calculate the areas spanned by the triangles ABD, ACD, and BCD. These are referred to as F_C, F_B, and F_A.
3. Calculate the interpolated MV $\vec{d}$ by weighting the MVs at the vertices of the triangle with their assosicated areas F_C, F_B, and F_A and rescale $\vec{d}$ according to the reference frame of its original PU.

These areas are identical to the barycentric coordinates of D. The variables used in the above algorithm are visualized in Figure 3. The choice of the interpolation function now depends on the underlying motion type. Linear weighting of the associated vertex vectors results in

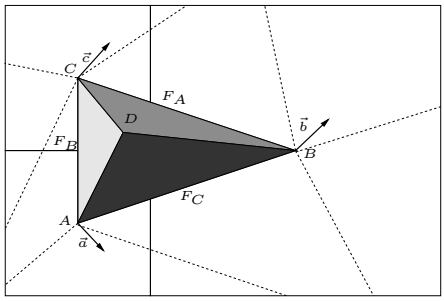

Fig. 3. The MV at location D within the Triangle ABC is computed as a weighted mean of $\vec{a}$, $\vec{b}$, and $\vec{c}$. The subtriangles between D and the three vertices A, B, C with the areas F_A, F_B, and F_C are chosen as weights.

$$\vec{d}_{\text{linear}} = \frac{\vec{a} \cdot F_A + \vec{b} \cdot F_B + \vec{c} \cdot F_C}{F_A + F_B + F_C}. \tag{5}$$

An example of a motion vector field produced by Equation 5 may be found in Figure 4. Here three blocks of equal size were chosen, that originated from an area with rotational motion. In the block-based interpretation of the motion data, each pixel within one of the blocks has the motion vector associated with the center pixel of the block ($\vec{a}$, $\vec{b}$, $\vec{c}$). The linearly interpolated motion as shown in Figure 4 clearly approximates the rotation quite well. If the linear interpolation

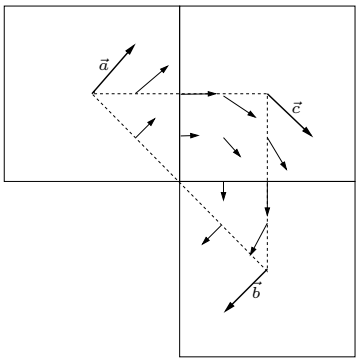

Fig. 4. Linearly interpolated MV field between three blocks of equal size. The original motion type (rotation) is successfully reconstructed.

is applied to vectors at object boundaries, the quality of the interpolated MVs is, however, worse than the quality of the original motion data (see Figure 5 (black line)). If linear interpolation is used, the pixels close to the object edge appear to move slower than they should. In this case better results are achieved, if a cubic interpolation function is used

$$\vec{d}_{\text{cubic}} = \frac{\vec{a} \cdot F_A^3 + \vec{b} \cdot F_B^3 + \vec{c} \cdot F_C^3}{F_A^3 + F_B^3 + F_C^3}. \tag{6}$$

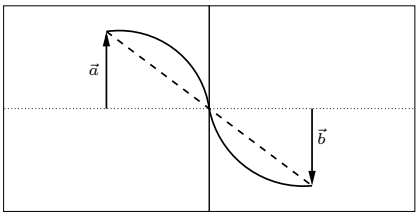

Fig. 5. Two blocks moving in opposite directions. With linear interpolation (dashed curve) intermediate motion vectors differ significantly from the true motion. With cubic interpolation (black curve) a much sharper motion edge is preserved.

Sequence	Resolution	HM 8.0+QTTF		HM 8.0+QTTFi	
		Δ_{PSNR} in dB	BD-rate in %	Δ_{PSNR} in dB	BD-rate in %
BasketballPass	416x240	0.00	−0.13	0.01	−0.22
BlowingBubbles	416x240	0.00	−0.06	0.01	−0.33
BQSquare	416x240	0.06	−1.74	0.07	−2.02
RaceHorses	416x240	0.00	−0.03	0.02	−0.44
BasketballDrill	832x480	0.00	0.01	0.00	−0.03
BQMall	832x480	0.00	0.04	0.00	0.01
PartyScene	832x480	0.00	−0.08	0.01	−0.20
RaceHorses	832x480	0.00	−0.09	0.00	−0.08
Vidyo3	1280x720	0.01	−0.35	0.01	−0.20
Vidyo4	1280x720	0.01	−0.14	0.00	−0.14
ParkScene	1920x1080	0.00	0.05	0.00	−0.02
Waterfall	704x480	0.01	−0.49	0.03	−0.93
average		**0.01**	**−0.17**	**0.01**	**−0.38**

Table 1. BD-rates and BD-PSNR for all tested sequences, both for the original QTTF interpolation and for the QTTF with interpolated motion vector fields (QTTFi).

The resulting motion distribution along the object edge may be found in Figure 5 (dashed curve). One method to determine which version of $\vec{d}$ is to be used, is to analyze the differences between the vertex MVs $\vec{a}$, $\vec{b}$, $\vec{c}$. In the current implementation cubic interpolation is utilized whenever two vertex vectors out of $\vec{a}$, $\vec{b}$, and $\vec{c}$ only differ by at most one quarter pel. In these cases it is assumed that at least two vectors describe the motion of the same object. Otherwise linear interpolation is used. Should all three vectors be similar, both interpolation methods produce the same result. Exemplary interpolated motion vectors may be found in Figure 6, where the MV distribution shows much less blockiness in the interpolated case.

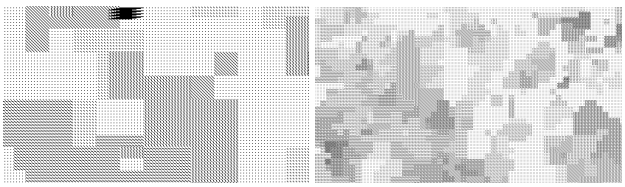

Fig. 6. *left:* Block-based motion vector field for reference list 0 of the upper left part of frame 7 of the BQSquare sequence. *right:* Interpolated motion vector field of frame 7.

4. EXPERIMENTAL EVALUATION

The proposed method has been implemented in C++ and is utilized as a preprocessing step for QTTF. It is to be noted that the interpolated MVs are only used for temporal filtering. The transmitted block-based motion field is not changed. In order to evaluate the quality of the interpolated MV field the MPEG sequence *BlowingBubbles* was manually segmented using the Human Assisted Motion Annotation Tool [11]. For the tested QP range (QP 22 to 37) the interpolated MV field improves the average endpoint error per pixel by 0.7 pel. The modified QTTF with interpolation (QTTFi) has been integrated into the HEVC test model HM 8.0 and tested on the sequences shown in Table 1. For the given dataset using the HEVC main profile QTTF alone produced an average bit rate reduction of 0.17%. With the added MV interpolation the average BD-rate [12] is increased to 0.38%. Improvements of 0.5% are observed for those sequences where QTTF already works well. In addition, sequences with large foreground objects, like the small version of *RaceHorses*, also show good improvements. However, the new method appears to work only well for low-resolution video. For the HD-sequences even slight losses occur. The current implementation of the QTTF increases the encoder runtime by 140% on average, while the decoder complexity is only increased by 30%. The additional interpolation does not significantly change these values since very efficient and fast algorithms exist both for the computation of barycentric coordinates and for Delaunay Triangulation. Future work will focus on configuring the interpolation parameters in a manner that make the method applicable to higher resolutions as well. Preliminary results indicate, that switching between linear interpolation and raw block-based motion data may further improve the results and that the threshold for identifying similarly moving blocks should also be changed depending on the QP. All decoded sequences, together with the associated RD-curves and the groundtruth motion data for *BlowingBubbles* may be found on the accompanying website `www.nue.tu-berlin.de/research/qttfi`.

5. SUMMARY

In this paper a novel method to interpolate dense MV fields from block-based motion data has been presented. The interpolated vectors are closer to the actual motion of the image points in video sequences. When the new motion data is utilized within the QTTF inloop filter, its performance is increased to an average bitrate reduction of 0.38% on the test dataset. Future work will include the adaptation of the filter to high-resolution sequences and the study of interference between QTTF and other tools within the final draft of HEVC.

6. REFERENCES

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (hevc) text specification draft 9," *ITU-T SG16 WP3, ISO/IEC JTC1/SC29/WG11 doc. JCTVC-K1003-v4*, Oct 2012.

[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)*, vol. 13, no. 7, pp. 614–619, Jul 2003.

[3] S. Wittmann and T. Wedi, "Transmission of post-filter hints for video coding schemes," *Proceedings of the 25th Picture Coding Symposium (PCS)*, pp. 81–84, Sep 2007.

[4] C.-M. Fu, C.-Y. Chen, Y.-W. Huang, and S. Lei, "Sample adaptive offset for hevc," *IEEE 13th Intl. Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–5, 2011.

[5] D. T. Vo and T. Q. Nguyen, "Optimal motion compensated spatio-temporal filter for quality enhancement of H.264/AVC compressed video sequences," *Proceedings of the 26th ICIP*, pp. 3173–3176, Nov 2009.

[6] A. Glantz, A. Krutz, M. Haller, and T. Sikora, "Video coding using global motion temporal filtering," *Proceedings of the 16th International Conference on Image Processing (ICIP)*, pp. 1053–1056, Nov 2009.

[7] M. Esche, A. Glantz, A. Krutz, M. Tok, and T. Sikora, "Quadtree-based temporal trajectory filtering," *Proceedings of the 26th International Conference on Image Processing (ICIP)*, 2012.

[8] B. Glocker, H. Heibel, N. Navab, P. Kohli, and C. Rother, "Triangleflow: Optical flow with triangulation-based higher-order likelihoods," in *11th European Conference on Computer Vision (ECCV)*, Crete, Greece, September 2010.

[9] M. Esche, A. Glantz, A. Krutz, and T. Sikora, "Adaptive temporal trajectory filtering for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 659–670, May 2012.

[10] M. Esche, M. Tok, A. Glantz, A. Krutz, and T. Sikora, "Efficient quadtree compression for temporal trajectory filtering," in *Data Compression Conference*, Snowbird, Utah, Mar. 2013 to appear.

[11] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss, "Human-assisted motion annotation," *Prodeedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2008.

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