MESH-BASED DECODER-SIDE MOTION ESTIMATION

Marco Munderloh, Sven Klomp, Jörn Ostermann, Fellow, IEEE

Institut für Informationsverarbeitung (TNT)
Leibniz Universität Hannover, Appelstr. 9a, 30167 Hannover, Germany
{munderloh, klomp, ostermann}@tnt.uni-hannover.de

ABSTRACT

Current video coding standards like H.264|AVC perform a block-based motion estimation and compensation at the encoder to exploit temporal dependencies between consecutive frames. Current research proved that motion compensation can also be done profitably at the decoder. In this so-called decoder-side motion estimation (DSME), frames are interpolated at the decoder and inserted into the reference buffer as additional information for prediction. To gather the motion information for compensation, the current approach is based on a block matching algorithm estimating one motion vector for each block of the frame to be interpolated. Therefore, only translational movement is compensated. We evaluate the applicability of a mesh-based motion compensation to DSME which models affine motion in each patch of the mesh and is able to compensate camera zooming or panning, object rotation, or object deformation.

I. INTRODUCTION

In current video coding standards, a block-based motion estimation (also called block matching algorithm, BMA) is used to exploit temporal correlation between frames and to compensate the motion prior to prediction error coding. The BMA are well known, of low complexity, and behave very well in the case of non-affine motion which can be assumed for small motion and/or small block sizes. Since all motion vectors have to be transmitted to the decoder, there is a practical lower limit for the block size and an upper limit for the number of motion parameters for each block. Therefore only translational motion is compensated, using block sizes of 4×4 and larger. For DSME frames which are computed at the decoder, no data needs to be transmitted so the limits do not apply here. Thus, dense motion fields using more than two parameters per block may be used to describe the motion between frames. In [1], which is taken as a reference, one 2D motion vector with quarter-pel resolution is estimated for each 16×16 block. The result is refined afterward using smaller block sizes of 8×8 and 4×4 as described in [2].

In case of non-translational motion, the mesh-based motion compensation approach showed up very promising [3], [4]. Instead of fixed blocks, a mesh of triangles, quadrangles, or quadrilaterals which varies in shape over time is used to compensate the motion. This allows the modeling of affine or projective transformation such as camera zooming and panning, or object rotation in x, y and z direction as well as local deformation of object shapes. This paper applies the advantages of a mesh-based motion estimation to the decoder-side temporal frame prediction and proposes a block-based / mesh-based hybrid coding system for H.264|AVC DSME.

Our mesh-based motion estimation algorithm is described in Section II. A comparison between block-based and mesh-based motion compensation is provided in Section III. Results for adding a mesh-based motion compensation to H.264|AVC DSME are plotted in Section IV.

II. ACTIVE-MESH-BASED MOTION ESTIMATION

Active-Mesh-based motion estimation algorithms automatically select features in a reference frame s_k and estimate the corresponding feature position in a second frame s_{k+1} to get motion vectors. These feature points are then used as nodes of a mesh. Each node n has a source position $\mathbf{p}_{n,k}$ in the reference frame s_k and a target position $\mathbf{p}_{n,k+1}$ in the second frame s_{k+1}. The motion of every other pixel of the image is interpolated using surrounding mesh nodes. In case of a triangular mesh, three nodes are used for the motion vector interpolation providing six parameters for an affine transform. We use a triangular mesh because of the smaller patch sizes allowing more accurate motion compensation compared to quadrilateral meshes. Given the relatively small frame-to-frame movements in a video sequence, an affine motion model should be sufficient.

In our approach, features are selected and tracked according to an extended Kanade-Lucas-Tomasi feature tracker [5]. A modified Harris corner detector is used for automatic feature selection based on the minimum Eigenvalue of a 2D image gradient matrix considering a minimum feature distance. Features are tracked by minimizing the difference between the feature window in frame s_k and the search window in frame s_{k+1} using a Newton-Raphson method.

The outliers are removed from the motion vector field by a clustering approach based on the motion field smoothness: directly neighboring motion vectors are successively clustered into one object if their spatial distance and their displacement difference are smaller than an adaptive threshold.

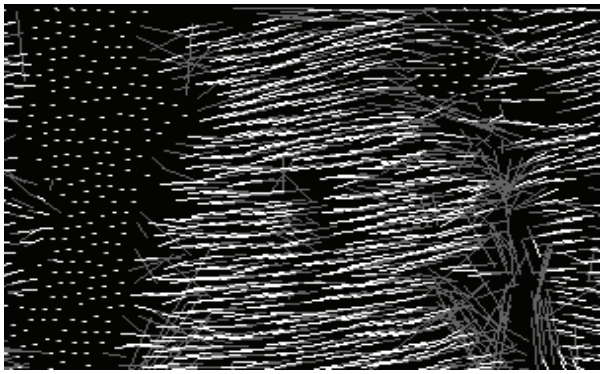

Fig. 1. Foward motion vector field for the upper left corner of frame 6 of the RaceHorses sequence. Inliers in white, outliers in gray.

Fig. 2. Triangle mesh for the upper left corner of frame 6 of the RaceHorses sequence.

If no remaining motion vector is fulfilling this requirement, a new object is created. Objects with less then 5 motion vectors are considered outliers. Results of the feature tracking and the outlier detection are shown for the RaceHorses sequence in Fig. 1.

Tracking is done in forward and backward direction. The resulting motion vector fields are merged into one field by inverting the motion direction for the backward track. During the merging process, motion vectors spatially closer than a given threshold are dropped and interpolated into one new motion vector instead. To get displacement information at the frame border, artificial features are added to the motion vector field. They are placed regularly on the borderline in frame s_k. Their displacement in frame s_{k+1} is predicted using the mesh nodes closest to each feature.

To create the dense motion field, the inlier feature point cloud is first triangulated using a Delaunay triangulation. The resulting triangle mesh is shown in Fig. 2. For each patch, the six transform parameters represented by $\mathbf{A_t}$ and $\mathbf{b_t}$ are calculated using the three nodes n_t spanning the triangle t (Eq. 2 and 3). With Eq. 1, for each pel $(x_{t,k}, y_{t,k})$

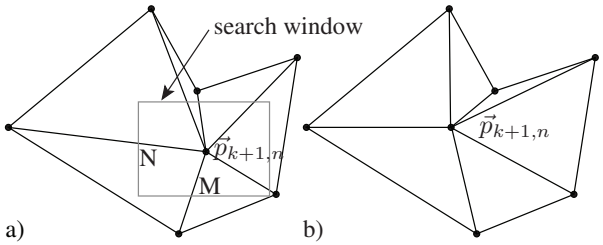

Fig. 3. Refinement of node n a) position $\mathbf{p}_{n,k+1}$ before refinement b) position $\mathbf{p}_{n,k+1}$ with minimal matching error after one refinement iteration.

inside the triangle t the associated displacement coordinate $(x_{t,k+1}, y_{t,k+1})$ is determined.

$$\begin{pmatrix} x_{t,k+1} \\ y_{t,k+1} \end{pmatrix} = \mathbf{A_t} * \begin{pmatrix} x_{t,k} \\ y_{t,k} \end{pmatrix} + \mathbf{b_t} \tag{1}$$

with $\mathbf{A_t} = \begin{bmatrix} A_t & B_t \\ C_t & D_t \end{bmatrix}$ and $\mathbf{b_t} = \begin{pmatrix} E_t \\ F_t \end{pmatrix}$ as transform parameters.

$$x_{n_t,k+1} = A_t x_{n_t,k} + B_t y_{n_t,k} + E_t \tag{2}$$

$$y_{n_t,k+1} = C_t x_{n_t,k} + D_t y_{n_t,k} + F_t \tag{3}$$

where $x_{n_t,k}, y_{n_t,k}$ specify the coordinates of the pixel inside the triangle t in frame s_k and $x_{n_t,k+1}, y_{n_t,k+1}$ in frame s_{k+1}, respectively.

Because the coordinate accuracy is not quantized but dependent on the calculation accuracy only, an appropriate interpolation filter must be used to calculate the pixel values. We use a two-stage filter. The half-pel positions are gained using the six tab Wiener filter presented in [6]. The more accurate sub-pel positions are calculated with a bilinear filter.

After creating the mesh, a refinement of the motion vectors is performed by minimizing the overall matching error of the triangles surrounding each mesh node. This is done consecutively for each node n by moving the motion vectors displacement $\mathbf{p}_{n,k+1}$ in the second frame s_{k+1} inside a search window of size $M \times N$ in sub-pel resolution while keeping the surrounding motion vectors fixed as shown in Fig. 3. For each position, the sum of the matching errors for all concatenated triangles is calculated using Eq. 1 and the position with the smallest overall matching error is selected as new motion vector displacement. The refinement process for all nodes is iteratively repeated until the matching error decrease drops below a threshold. This might also be done in closed-form as presented in [7].

The computational complexity of the presented motion estimation algorithm is comparable to that of a conventional BMA, considering the same number of motion vectors and omitting the refinement process which is not yet optimized.

III. FRAME PREDICTION RESULTS

The block matching motion estimation algorithm currently used in the DSME proposal [1] is taken as a reference.

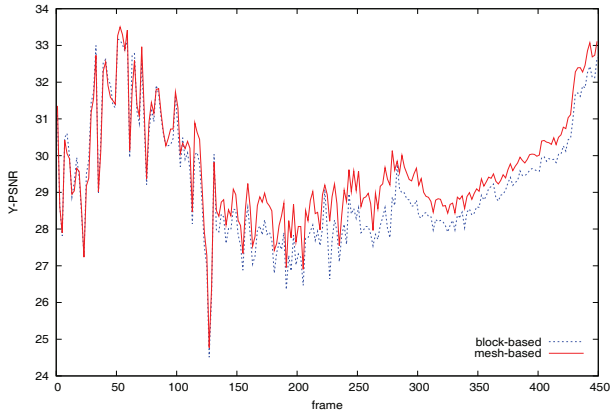

Fig. 4. Frame prediction performance for the Concrete sequence.

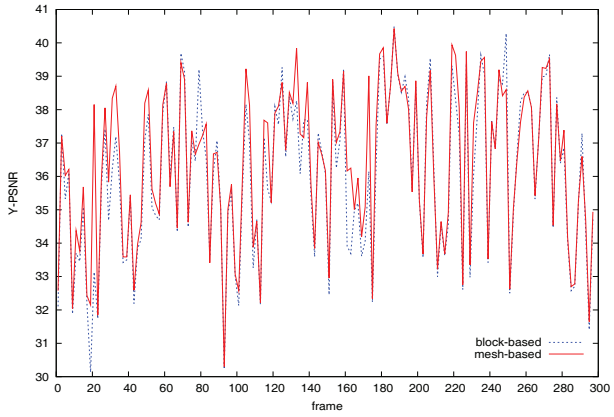

Fig. 5. Frame prediction performance for the City sequence.

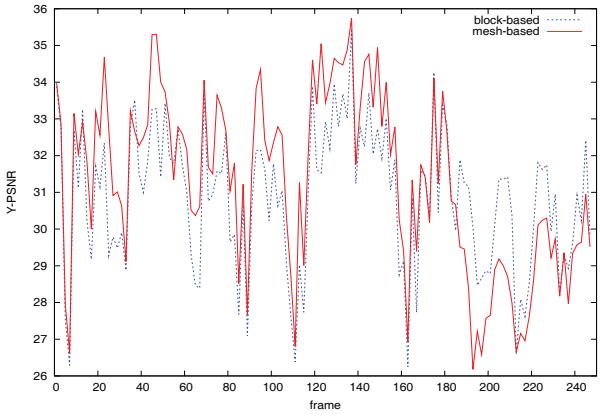

Fig. 6. Frame prediction performance for the Flowergarden sequence.

Fig. 7. Interpolation at object borders. Flowergarden sequence, mesh-based on the left, block-based on the right.

Herby, a frame s_k between two reference frames s_{k-1} and s_{k+1} is interpolated at the decoder to reduce the data rate for motion vectors and/or the prediction error in H.264|AVC B-slices. As within the current DSME implementation, linear motion is assumed so that

$$\mathbf{p}_{n,k} = \mathbf{p}_{n,k-1} + \frac{\mathbf{p}_{n,k+1} - \mathbf{p}_{n,k-1}}{2} \tag{4}$$

For minimizing quantization errors and camera noise, the pixel values of frame s_k are the average of the corresponding pixel values in the two source frames s_{k-1} and s_{k+1} by applying the inverse vector of Eq. 4 to s_{k+1}.

For comparison of the motion estimation performance, we compared the Y-PSNR between the original frame and the predicted frames s_k created using the previous and the following frame of the original CIF-sequences as source frames s_{k-1} and s_{k+1}. As expected, in case of affine motion the mesh-based motion estimation outperforms the block-based approach as for the zooming part in the second half of the Concrete sequence in Fig. 4. Even without affine motion we get similar or even better results as for the City or the Flowergarden sequence with its largely left or right scrolling content in Fig. 5 and Fig. 6. However, in case of largely moving foreground objects, the mesh-based approach with its continues mesh has more problems at object borders as the block-based one as shown in Fig. 7 for the tree in the Flowergarden sequence. This results in a inferior prediction performance from frame 185 towards the end in Fig. 6. Given this, a hybrid coding system for the DSME frame prediction in H.264|AVC is proposed able to select on a frame-to-frame basis between a block-based and a mesh-based frame prediction considering the rate-distortion.

IV. CODING RESULTS

To evaluate the coding performance, the mesh-based and the block-based decoder-side frame prediction have been implemented in the H.264|AVC reference software JM 16.1 according to [1]. Additionally, the curves of the unmodified reference coder are given as well as the curves for the combined block-based/mesh-based coder. The used test sequences are in CIF-resolution with a frame rate of 30 Hz.

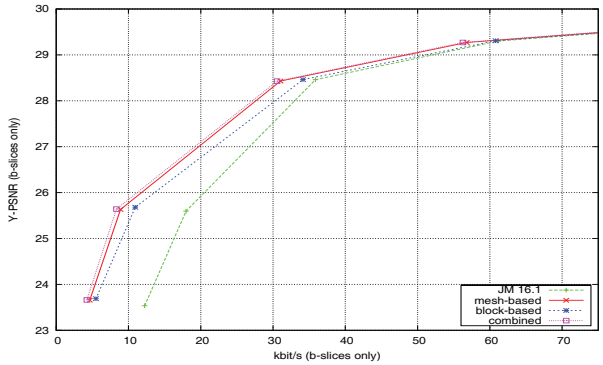

Fig. 8. RD performance for the Concrete sequence.

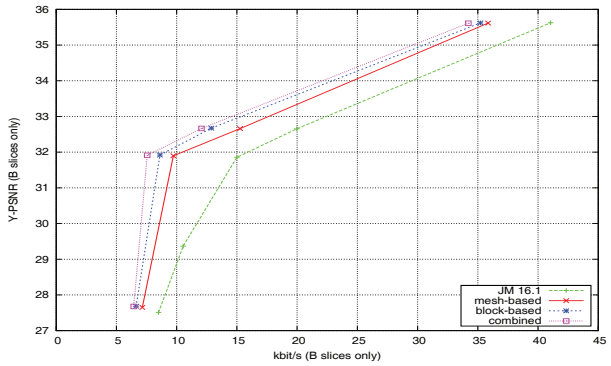

Fig. 9. RD performance for the City sequence.

Each coder used a GOP structure of I-B-P-B-P and the same coding settings. As without using hierarchical B-pictures, the influence of the frame prediction is limited to B-slices only, just these values are plotted.

Best coding results for the proposed method are reached in case of affine motion as in the Concrete sequence with its long zooming part given in Fig. 8. The mesh-based interpolation outperforms the block-based method by up to 1 dB and the reference coder by up to 2.2 dB for low bit rates. However, for higher bit rates, the coding gain is decreasing. For the City sequence in Fig. 9 the combined coder gains up to 2 dB for low bit rates compared to the block-based coder and up to 4.5 dB compared to JM. A gain of 1 dB is reached even for higher bit rates. Including the I- and P-slices, the gains for the combined coder are 0.7 dB over JM and 0.25 dB over the block-based approach. For the Foreman sequence in Fig. 10, a gain of up to 1.3 dB is reached and stays at a level of 0.6 dB for higher bit rates. For the flowergarden sequence in Fig. 6, one should note that, without transmitting any data, a Y-PSNR of around 36 dB is reached assuming fine quantization in the reference frames.

V. CONCLUSIONS

In this paper we present a mesh-based temporal frame prediction for decoder-side B-slice motion estimation and

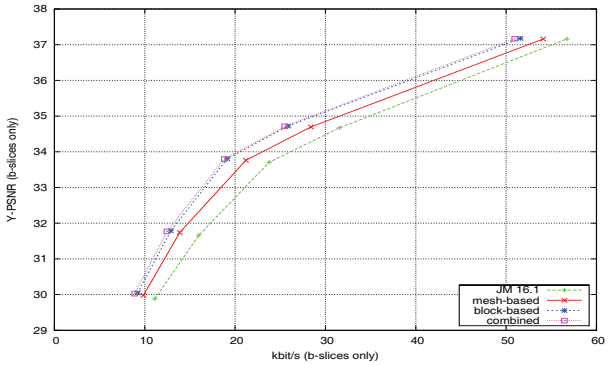

Fig. 10. RD performance for the Foreman sequence.

compare it to the current block-based approach as well as to the JM reference coder. In case of affine motion, the mesh-based interpolation outperforms the block-based approach by up to 5 dB for single frames. After coding, gains of up to 4.5 dB Y-PSNR in comparison to JM and 2 dB compared to the block-based motion estimation are achieved for b-slices using the proposed combined block-based/mesh-based coder.

VI. REFERENCES

[1] S. Klomp, M. Munderloh, Y. Vatis, and J. Ostermann, "Decoder-side block motion estimation for h.264 / mpeg-4 avc based video coding," in *Proc. of the Int. Symp. on Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 1641–1644.

[2] J. Ascenso, C. Brites, and F. Pereira, "Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding," in *5th EURASIP*, Slovak Republic, July 2005.

[3] Y. Wang and O. Lee, "Active mesh - a feature seeking and tracking image sequence representation scheme," *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 610–624, Sept. 1994.

[4] S. Tran Minh, J. Benois-Pineau, K. Fazekas, and A. Gschwindt, "Mesh-based error-scalable video object codec for variable bandwidth multimedia communications," in *Proc. of Int. Conf. on Image Processing*, vol. 1, Sept. 2002, pp. 745–748.

[5] J. Shi and C. Tomasi, "Good features to track," in *Proc. of Comp. Society Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994, pp. 593–600.

[6] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression*. West Sussex, England: John Wiley & Sons Ltd., 2003, ch. 6.5.1.4.

[7] Y. Altunbasak, A. Murat Tekalp, and G. Bozdagi, "Two-dimensional object-based coding using a content-based mesh and affine motion parameterization," in *Proc. of Int. Conf. on Image Processing*, vol. 2, Washington, DC, Oct. 1995, pp. 394–397.