5

Motion Estimation at the Decoder

Sven Klomp and Jörn Ostermann
Leibniz Universität Hannover
Germany

1. Introduction

All existing video coding standards, such as MPEG-1,2,4 or ITU-T H.26x, perform motion estimation at the encoder in order to exploit temporal dependencies within the video sequence. The estimated motion vectors are transmitted and used by the decoder to assemble a prediction of the current frame. Since only the prediction error and the motion information are transmitted, instead of intra coding the pixel values, compression is achieved. Due to block-based motion estimation, accurate compensation at object borders can only be achieved with small block sizes. Large blocks may contain several objects which move in different directions. Thus, accurate motion estimation and compensation is not possible, as shown by Klomp et al. (2010a) using prediction error variance, and small block sizes are favourable. However, the smaller the block, the more motion vectors have to be transmitted, resulting in a contradiction to bit rate reduction.

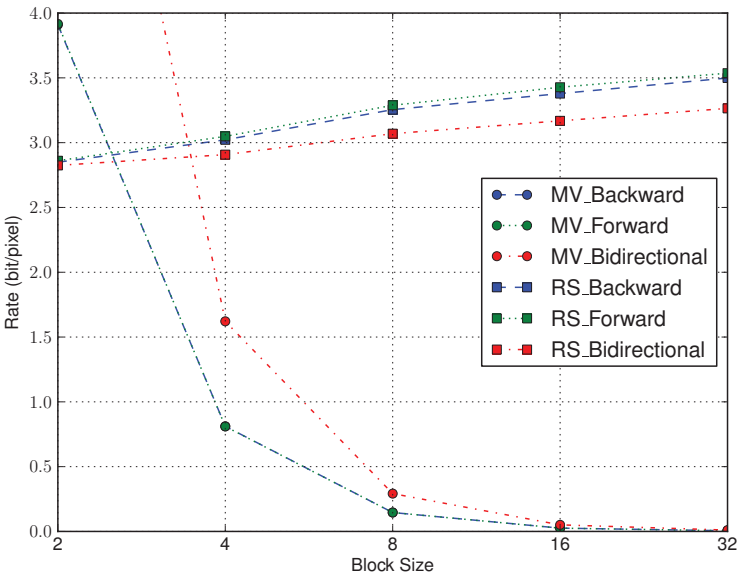


Fig. 1. Data rates of residual (RS) and motion vectors (MV) for different motion compensation techniques (Kimono sequence).

These characteristics can be observed in Figure 1, where the rates for the residual and the motion vectors are plotted for different block sizes and three prediction techniques. Backward motion compensation generates a prediction by only using the frame temporal before the current frame to be coded, whereas forward motion compensation only uses the following frame. Bidirectional motion compensation averages the motion compensated blocks from both frames to form the prediction, which is typical for B frames. As expected, the rate for the residual decreases with smaller block sizes. Bidirectional motion compensation performs best, but also needs more bits to code the motion vectors. The motion vector rates for forward and backward motion estimation are almost identical. For large block sizes, the rate for the motion vectors is negligible, but increases significantly towards small blocks. It can be observed that the block size has a significant impact on the compression performance and is, therefore, adaptively chosen for each macroblock and limited to 4×4 pixels in the ITU-T and ISO/IEC (2003) standard H.264 / AVC.

Increasing computational power allows to implement more sophisticated algorithms, even at the decoder. Recent studies have shown that motion estimation algorithms at the decoder can significantly improve compression efficiency. The estimated motion is already known at the decoder and the transmission of the motion vectors can be omitted.

A first attempt to estimate the motion at the decoder was proposed by Suzuki et al. (2006), where additional macroblock and sub-macroblock modes were added to H.264 / AVC. Those modes predict a block by performing template matching at the decoder, instead of using transmitted motion vectors. This approach was further improved by calculating a weighted average of multiple candidates (Suzuki et al., 2007).

In contrast, Kamp et al. (2008) proposed to adapt the existing P modes instead of introducing additional modes. The encoder decides to either code the motion vector explicitly or to use the derived vector by a rate-distortion optimised mode decision. This so-called decoder-side motion vector derivation (DMVD) estimates the motion vector by matching a template of already reconstructed pixels in the reference frames at different positions. In case DMVD is selected, only the decision has to be signalled and no motion vector is transmitted. The system was later extended to support DMVD also within B macroblock modes (Kamp & Wien, 2010).

Decoder-side motion estimation (DSME), proposed by Klomp et al. (2009), is another approach to reduce the rate for the motion vectors. Temporal interpolation is used to generate a prediction of the current frame at the decoder. This DSME frame is stored and can be used as additional reference for the conventional coding tools. The motion estimated by the conventional tools should be zero, as the DSME frame is already motion compensated and thus, the motion vectors are very efficient to code.

The rest of this chapter is organised as follows: Section 2 recapitulates inter frame coding used in H.264 / AVC. Detailed descriptions of the DMVD and DSME approaches are given in Section 3 and 4, respectively. Experimental results are presented in Section 5. The chapter comes to a close with conclusions in Section 6.

2. Conventional inter frame coding

This section gives a rough overview of conventional inter frame coding used in the ITU-T and ISO/IEC (2003) standard H.264 / AVC, to get a better understanding of the design changes introduced by DMVD and DSME, as described in Section 3 and 4, respectively.

In H.264 / AVC, one frame is divided into so-called macroblocks with the size of 16×16 pixels. In contrast to intra coding, where the current macroblock is predicted by only using already decoded data from within the current frame, inter coding uses reference frames for

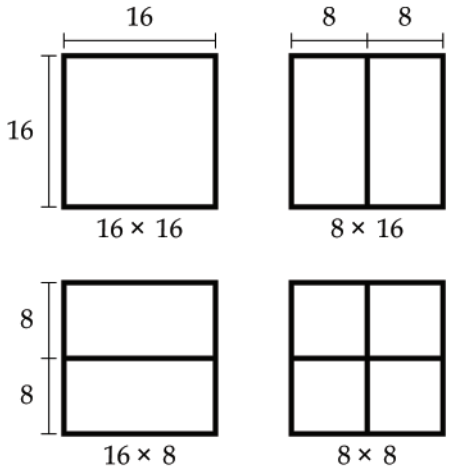


Fig. 2. Macroblock partitions used for inter coding.

motion compensated prediction. As mentioned before, the bit rate to code one block depends on its size. Therefore, a macroblock can be partitioned into smaller blocks in inter coding, to get the best trade-off between residual rate and the rate needed for coding the motion vectors, as depicted in Figure 2. Additionally, a 8×8 macroblock can be further divided into sub-partitions of sizes 8×4, 4×8 and 4×4.

At the encoder, the motion vector for each macroblock (sub-)partition is estimated and a rate-distortion optimised decision, as proposed by Sullivan & Wiegand (1998), is made on what partition structure should be used to minimise the rate for the residual and the motion vectors. Since the partitions are small compared to the size of moving objects within the sequence, it is most likely that neighbouring partitions have similar motion vectors. Thus, H.264 / AVC predicts the motion vector of the current block with the motion vectors from up to three neighbouring blocks, as depicted in Figure 3.

The prediction algorithm is described in more detail by Richardson (2003). This prediction is subtracted from the actual motion vector and only the difference is coded into the bitstream. The details on coding the residual is not of interest for DMVD nor DSME and is omitted in this description.

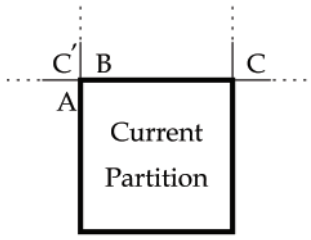


Fig. 3. Motion vectors of block A, B and C are used for motion vector prediction. For DMVD predictive search, A and C (C' if C is unavailable) are used for motion derivation.

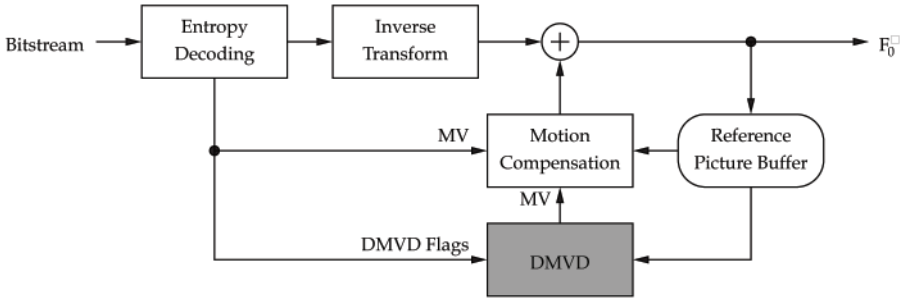


Fig. 4. Simplified block diagram of a conventional hybrid decoder with DMVD modifications, as proposed by Kamp et al. (2008), highlighted in gray.

3. Decoder-side motion vector derivation

The main idea of DMVD is to reduce the rate for the motion information by deriving the motion vectors at the decoder. In case the derivation is successful, no motion information has to be coded into the bitstream and compression is achieved. However, the derived motion vector is not always correct and might impair the compression efficiency due to large prediction error. Therefore, DMVD should be selected adaptively for each macroblock by performing a rate-distortion optimised decision at the encoder, similar to the Lagrangian optimisation described by Sullivan & Wiegand (1998). To transmit this decision to the decoder, Kamp et al. (2008) added additional flags within the H.264 / AVC macroblock layer syntax to signal either the use of the motion vector derived with DMVD, or the use of an explicitly coded motion vector: One flag for the 16×16 macroblock type is used to signal whether the motion vector is coded into the bitstream or whether DMVD is used. Two flags are sent for the two partitions of the 16×8 and 8×16 macroblock types. Furthermore, the sub-types of the 8×8 partitions are coded with one flag for each sub-block. These flags are interpreted in the entropy decoder as shown in Figure 4.

If DMVD flags are set, the motion vector is derived at the decoder and used for motion compensation. Otherwise, an explicitly transmitted motion vector is decoded and used for compensation. Kamp, Ballé & Wien (2009) have shown that the performance can be further improved by using multi-hypothesis prediction. Instead of deriving one motion vector and using the corresponding block as prediction, the two best motion vectors are derived to improve the prediction accuracy. The corresponding blocks of these motion vectors are averaged to form the improved prediction. The following section describes the motion vector derivation process in more detail.

3.1 Template-based motion derivation

For the template-based motion derivation, it is assumed that the objects of the sequence are larger than one block partition. Thus, pixels neighbouring the current block belong to the same object. If the motion of these pixels is known, it can be used for the motion compensation of the current block.

Current video coding standards have in common that the macroblocks within a frame are coded line by line. Therefore, the decoded pixel values above and left of the current macroblock are, in most cases, available. These values can be used to estimate the motion of the current block. Kamp et al. (2008) proposed to use an L-shaped template with a thickness of four pixels, which is matched in the reference frames. However, a full search within all

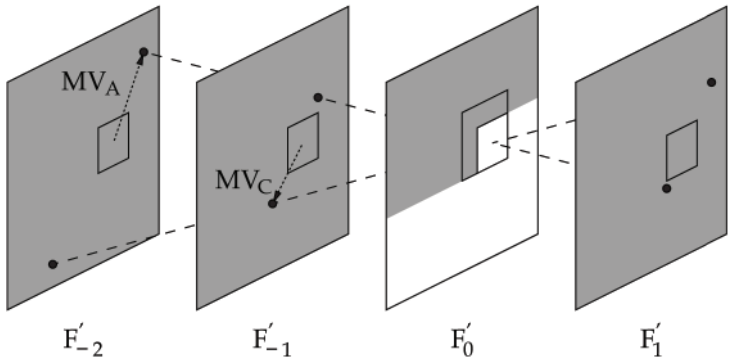


Fig. 5. L-shaped template used for motion search of the current block. Already decoded frame regions are shown in grey. The candidates (black circles) for template matching are derived from motion vectors of two neighbouring blocks (MV_A, MV_C) using linear scaling.

reference frames would induce a high amount of additional complexity at the decoder, and is, therefore, not feasible. For this reason, the template is matched for a smaller search range. It is assumed that the H.264 / AVC motion vector predictor (MVP) is already a good starting point and, thus, is used as the centre of the search range for each reference frame. MVP uses the motion vectors of up to three already decoded partitions next to the current block (Figure 3, block A, B and C) to calculate a prediction of the current motion vector.

The search range around the predicted motion vector is set to four pixels with sub-pel refinement, which introduces high complexity to the decoder. The position with the smallest sum of absolute differences (SAD) is chosen as the derived motion vector.

The complexity of the motion search can be significantly reduced by using a candidate-based approach: A set of candidate motion vectors are selected and the SAD is only calculated at these positions, in order to find the best motion vector. The number of candidates should be small to achieve noticeable complexity reduction.

Kamp, Bross & Wien (2009) proposed to use the motion vectors of neighbouring blocks as candidates for motion vector derivation. More precisely, the motion vectors from block A and C, as depicted in Figure 3, are used. In the case that block C is not available, C' is used. However, the neighbouring blocks may use different reference frames and thus, the motion vectors should be scaled according to the temporal distance of the reference frame. In Figure 5, it is assumed that three reference frames are available and that block A and C use frame F'_{-2} and F'_{-1} as reference, respectively. The two vectors are scaled, resulting in six candidates for the template matching.

Kamp, Bross & Wien (2009) observed that a sub-pel refinement results in improved predictions, although the candidates can already have fractional-pel accuracy. The refinement is performed for each candidate by performing a full-search with quarter-pel accuracy and a search range of only one pixel.

Out of these candidates, the two vectors with the smallest SAD are used to calculate the prediction as described before. Thereafter, the remaining prediction error is coded conventionally using the H.264 / AVC tools.

4. Decoder-side motion estimation

In contrast to decoder-side motion vector derivation, where the aim is to eliminate the need of transmitting motion information to the decoder, decoder-side motion estimation tries to achieve compression by providing an additional reference frame that can be used for frame prediction. Klomp et al. (2009) implemented DSME on top of a H.264 / AVC coder. A simplified block diagram of a conventional hybrid encoder with highlighted DSME changes is depicted in Figure 6.

The reference picture buffer, containing already coded pictures, feeds the previous frame F'_{-1} and the future frame F'_1 that are temporally the closest to the current frame F_0, to the DSME block. The block interpolates between the two frames to create a prediction $\hat{F}_0$ of the current frame. Any temporal interpolation algorithm can be adopted into this architecture, as shown by Klomp et al. (2009) and Munderloh et al. (2010) for block-based and mesh-based motion estimation, respectively.

Since no information regarding the motion between F_0 and the two reference frames F'_{-1} and F'_1 is available at the decoder, linear motion is assumed. Thus, the motion vector between F'_{-1} and F_0 is equal to the motion vector between F_0 and F'_1, and can be expressed by the halved motion between F'_{-1} and F'_1. This assumption can be treated as valid because of the high frame rates of modern video content. After each block of the current frame is interpolated, it is inserted into the reference picture buffer. The tools of the conventional encoder are now able to use this frame for prediction and coding in addition to the other reference frames stored in the reference picture buffer.

Since no distinction is made between the inserted DSME frame and the other reference frames, the encoder transmits the motion vector difference, as described by Richardson (2003), also for a block using the DSME frame $\hat{F}_0$. However, the motion vector estimated by the encoder should be zero, as $\hat{F}_0$ is already motion compensated, and thus, the motion vector difference is very efficient to code. This approach has the advantage that the DSME frame can also be used in cases where the assumption of linear motion is wrong, as shown in Figure 7:

The DSME algorithm estimates the motion between the two reference frames and inserts the interpolated block in the centre of $\hat{F}_0$ because of the assumption of linear motion. However, the

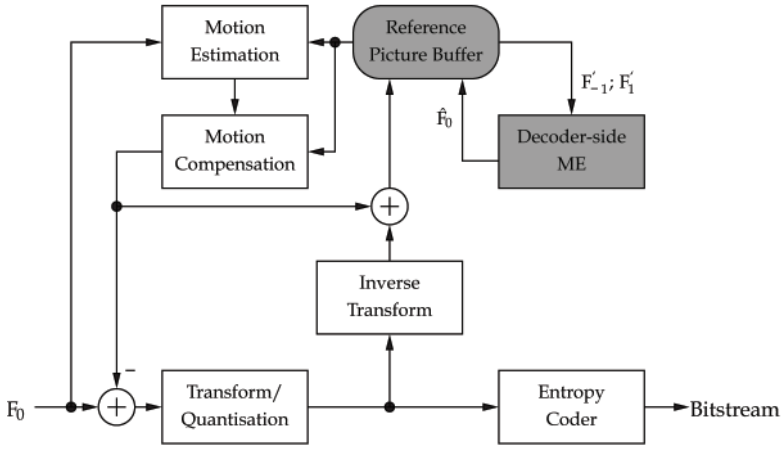


Fig. 6. Simplified block diagram of a conventional hybrid encoder with DSME modifications, as proposed by Klomp et al. (2009), highlighted in grey.

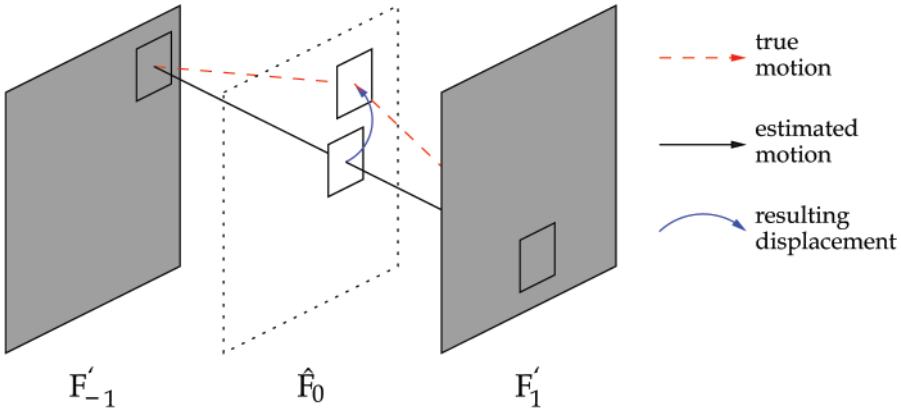


Fig. 7. Displaced interpolation due to nonlinear motion.

true motion drawn with dashed arrows is not linear and the correct position of the block is in the upper right corner. Nevertheless, the encoder can still use that block by applying motion compensation at the DSME frame and transmitting the resulting displacement as depicted in Figure 7.

The most important part of the DSME block is the motion estimation algorithm, since the performance of the proposed system is mainly affected by the accuracy of the motion vectors. Accurate motion vectors result in a good interpolation of the current frame, which will then be selected more often as a reference by the encoder tools. Conventional block matching algorithms, as used in e.g. H.264 / AVC, are not applicable to decoder-side motion estimation, since they only minimise some cost function like the sum of the absolute/squared differences (SAD/SSD) and do not search for true motion. Thus, an adapted motion estimation algorithm is needed, which is described in detail in Section 4.1.

Since the used algorithm interpolates between two frames, the approach is only applicable to B frames, in which a future frame is available at the decoder. However, the design is very flexible and extrapolation from previous frames can easily be implemented to allow DSME also for P frames.

4.1 Motion vector search

As explained in the previous section, the motion estimation algorithm of the DSME block shown in Figure 6 has a significant impact on the coding performance. In this approach, the motion is estimated by minimising the sum of the squared differences (SSD) between the two reference frames F'_{-1} and F'_1. The 6-tap Wiener filter proposed by Werner (1996), which is similar to filter standardised in H.264 / AVC, is used to interpolate sub-pixel values needed to estimate motion vectors with half-pel accuracy.

However, a false local minimum might be found if the block size is small and the search range large, as shown in Figure 8. Thus, the motion compensated interpolation fails, resulting in a distorted frame $\hat{F}_0$ (Figure 9).

To prevent inaccurate motion vectors, the hierarchical motion estimation scheme proposed by Klomp et al. (2010b) is used (Figure 10). The algorithm starts by using a block size of 64×64 pixels and a search range of 128 pixel. For the following iterations, the search range is decreased for each hierarchy level, since the coarse motion was already estimated in previous

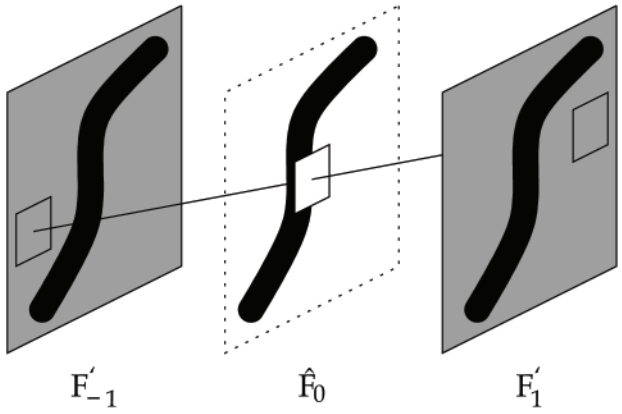


Fig. 8. Bidirectional motion compensation can fail due to large search range and homogeneous background.

levels. The two reference frames F'_{-1} and F'_1, which are used to estimate the motion, are low-pass filtered in the first iteration to prevent local minima due to the large search range. For all other iterations, the unfiltered reference frames are used, since the search ranges are smaller as previously mentioned.
At each hierarchy level, the motion vectors between the previous and the next frame (F'_{-1}, F'_1) are estimated using a conventional block matching algorithm by minimising the SSD. For smaller blocks, the matching window of the motion estimation is slightly larger than the block size during motion compensation to be more robust against image noise. This parameter is set individually for each level.
The search area used for the motion estimation in the current hierarchy level depends on the current search range and the motion vectors of the previous hierarchy level, as depicted in Figure 11: The nine neighbouring motion vectors of the previous level are applied to the current block and define a set of starting points. The search area used for motion estimation is

Fig. 9. Detail of the interpolated frame $\hat{F}_0$ of the PeopleOnStreet sequence. Failed motion compensation is highlighted in red.

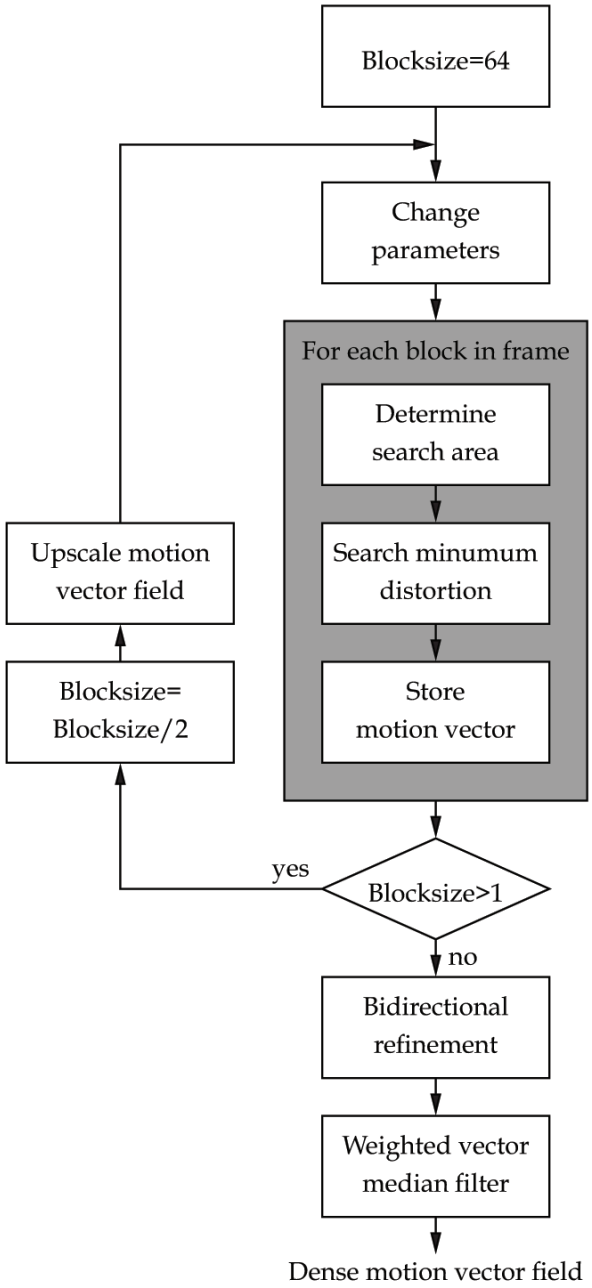
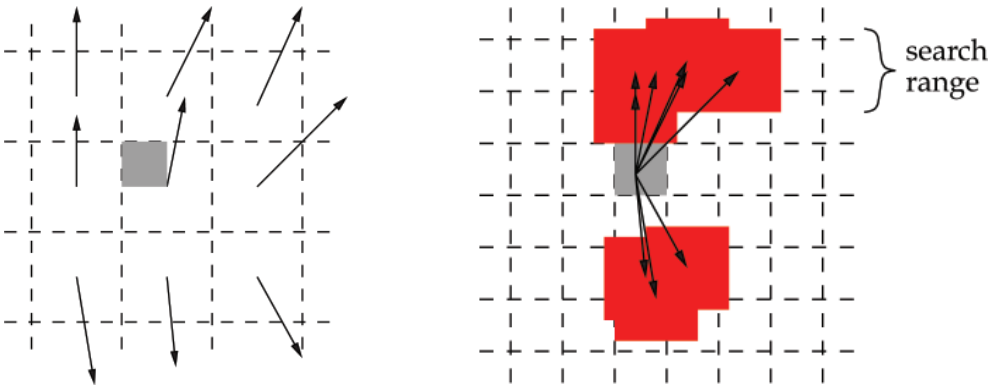


Fig. 10. Block diagram of the hierarchical motion estimation algorithm.



(a) Previous level H_{n-1} (b) Current level H_n

Fig. 11. Search area (red) derived from previous hierarchy level.

calculated by applying the search range to each starting point. Thus, the current block is able to follow the motion of every neighbouring block while still using small search ranges, which significantly reduces the amount of SSD computations compared to the search algorithm used by Klomp et al. (2009).
If the last iteration of the forward motion search is finished, the motion vectors are aligned to the block grid of the current frame. A bi-directional refinement is performed, in which the best sub-pel position is calculated for each motion vector. Thereafter, the vector field is smoothed using a vector median filter weighted by the mean of absolute differences (MAD) of the displaced blocks, as proposed by Alparone et al. (1996), in order to eliminate outliers.
After the motion vector field is estimated, the intermediate frame is interpolated in two steps. First, motion vectors is halved resulting in to motion vector pointing from the current frame to the corresponding blocks in the previous and next frame as depicted in Figure 7. The same 6-tap Wiener filter as used for motion estimation is selected to calculate pixel values at sub-pel positions. Second, the pixel values from the two reference frames are averaged and produce the interpolated frame. Using the pixel values from both frames reduces noise caused by the quantisation of the reference frames and also the camera noise. If only one reference frame is motion compensated and used as interpolated frame would give worse quality and, thus, impair the coding performance.
Thereafter, the interpolated frame is fed into the reference picture buffer an can be used by all H.264 / AVC coding tools as already explained in Section 4. The following section evaluates the coding performance of this approach and gives a comparison with DMVD.

5. Experimental results

To evaluate the performance, seven test sequences are coded using three approaches: Decoder-side motion vector derivation (DMVD), decoder-side motion estimation (DSME) and the underlying ITU-T / ISO/IEC standard H.264 / AVC as reference. These sequences are also used by ITU-T and ISO/IEC (2010) to evaluate tools for a new video coding standard. To allow random access every second, the intra frame period is individually set for each test sequence according to the frame rate. Every eighth frames is coded as P frame. The remaining frames are coded as hierarchical B frames resulting in the following GOP structure: I-b-B-b-B-b-B-b-P.

Sequence	DMVD	DSME
Kimono, 1080p, 24Hz	-9.8 %	-4.9 %
BasketballDrive, 1080p, 50Hz	-5.7 %	-5.7 %
BQTerrace, 1080p, 60Hz	-8.8 %	-4.7 %
PeopleOnStreet, 2560x1600, 30Hz	-9.0 %	-13.0 %
Cactus, 1080p, 50Hz	-5.6 %	-8.2 %
ParkScene, 1080p, 24Hz	-8.9 %	-8.4 %
Traffic, 2560x1600, 30Hz	-7.6 %	-9.7 %

Table 1. BD rate gains for DMVD, DSME compared to H.264 / AVC reference for several high-definition sequences.

The operational rate-distortion curves for four distinctive sequences are plotted in Figure 12 to 15. To obtain an objective measurement of the average rate gains, Bjøntegaard (2001) proposed a method where a third order polynomial is fit into four data points of the rate distortion curve. The so-called Bjøntegaard Delta (BD) rate gains for all seven test sequences are provided in Table 1.
The Kimono sequence used to evaluate the compression performance consist of a global camera pan with a moving person in front of trees. As shown in Figure 12, DMVD has a slightly better performance compared to DSME for the lowest rate point. However, the gain of DSME decreases for higher rates while the gain of DMVD is almost constant for all rate points. Therefore, the BD rate gain of 9.8% for DMVD is twice as big as the gain for DSME, as shown in Table 1.
The BasketballDrive sequence was captured during a Basketball match. It contains some motion blur due to the fast movements of the players. Coding this sequence also results in compression improvements for both approaches, although fast motion and occlusion due to non-rigid objects occur. DSME outperforms DMVD for lower bit rates as shown in Figure 13. Again, the compression performance of DSME decreases for higher bit rates and DMVD gives better results for those rates. Nevertheless, the average rate gains for DMVD and DSME are both 5.7%.
The gain of the DSME approach for BQTerrace (Figure 14) is with 4.7% the lowest for all sequences. The motion estimation fails at the water surface and flames due to the non-rigid motion and missing texture. DMVD can handle these characteristics slightly better.
Sequences with very high resolution can gain even more by using DMVD and DSME, as shown in Figure 15. The PeopleOnStreet sequence is a 4k $\times$ 2k sequence captured with a static camera and contains people crossing a street. The sequence was cropped to 2560×1600 pixels to limit the computational resources for the experiments. DMVD achieves 9% bit rate reduction for this sequence. The average gain is even higher for DSME: 13% bit rate reduction is achieved, since the motion can be estimated very accurately due to the high amount of detail in this high resolution sequence.
The rate-distortion plots for the different sequences have in common that the gain of DSME decreases towards higher rates. The improved quality of the key frames F'_{-1} and F'_1 have almost no influence on the DSME frame estimation accuracy. Thus, the conventional coding tools select more often the other high quality reference frames instead of the interpolated DSME frame. This characteristic can also be observed in Figure 16.
The amount of the macroblocks, which use the DSME frame for prediction, are plotted for the different rate points. The DSME usage is highest for rate point one, which corresponds to the lowest bit rate. For larger rate points, and thus, increasing bit rate and quality, the amount

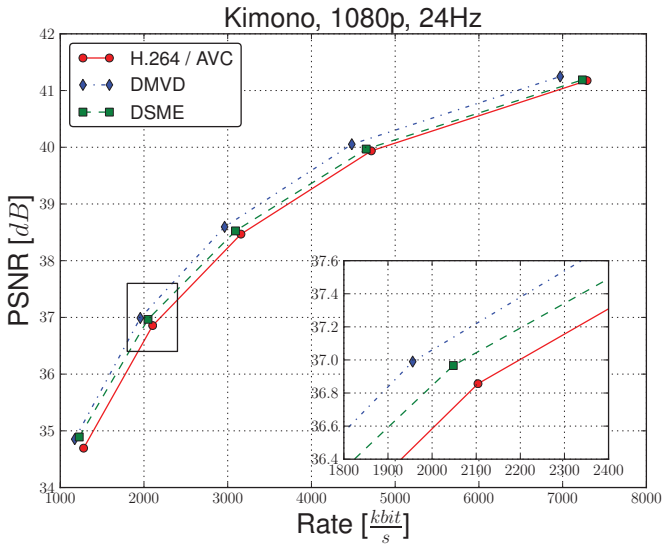


Fig. 12. Rate-Distortion performance of DMVD and DSME for the Kimono sequence.

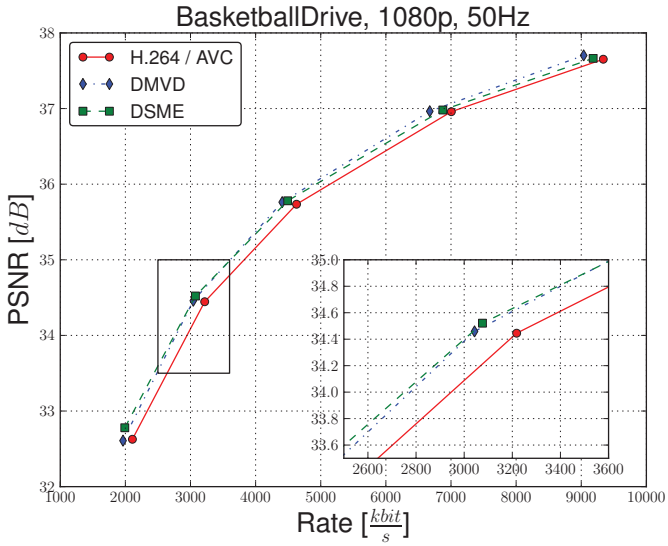


Fig. 13. Rate-Distortion performance of DMVD and DSME for the BasketballDrive sequence.

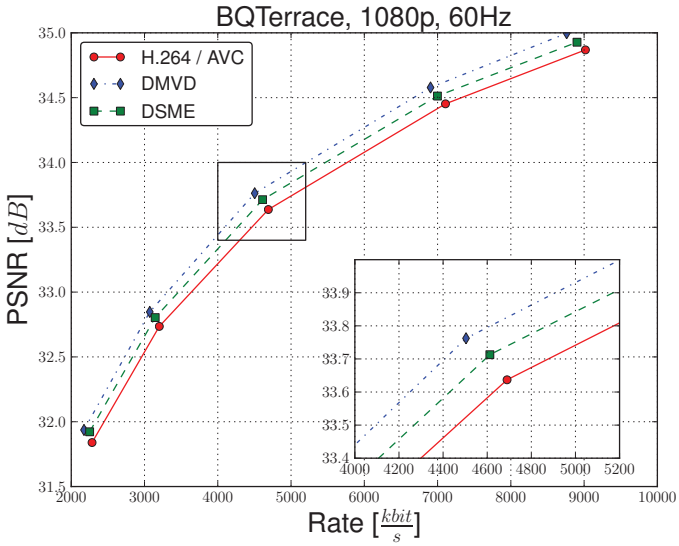


Fig. 14. Rate-Distortion performance of DMVD and DSME for the BQTerrace sequence.

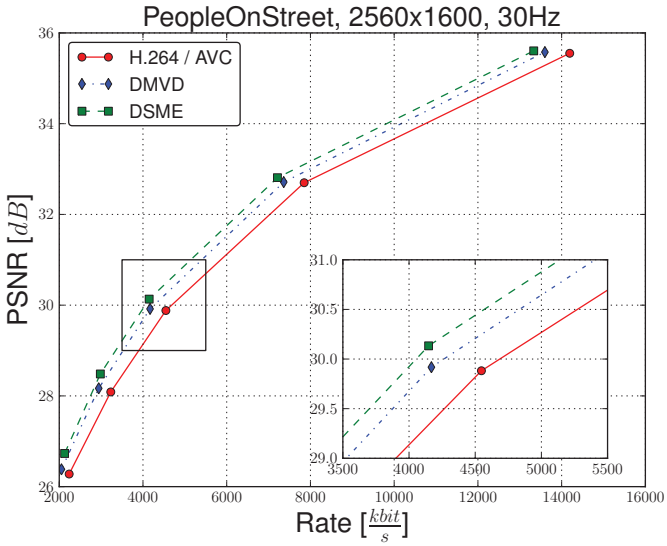


Fig. 15. Rate-Distortion performance of DMVD and DSME for the PeopleOnStreet sequence.

declines. Therefore, the performance of DSME and the H.264 / AVC reference converges. Furthermore, the reduced rate for the motion vectors has a smaller impact at high rate points. In contrast, DMVD achieves high gains for all rate points. Since DMVD uses already decoded pixel values for template matching, it can benefit from the improved quality of the current frame.
Figure 16 is also a good indicator of the overall DSME performance: For the well performing PeopleOnStreet sequence, almost 50% of the macroblocks and sub-macroblocks use the DSME frame for prediction. In contrast, the DSME usage while coding the BQTerrace sequence is around 30%.
The performance gains of the three remaining sequences of the test set have similar characteristics as shown in Table 1. DSME outperforms DMVD for the Cactus and Traffic sequences. Only for ParkScene is the gain of DMVD slightly better than DSME.

6. Conclusions

Two very promising approaches, which improve the compression efficiency of current video coding standards by estimating motion at the decoder, were described. Decoder-side motion vector (DMVD) derivation calculates a prediction of the current block at the decoder by using already decoded data. Thus, no motion vectors have to be transmitted and compression is achieved. Decoder-side motion estimation (DSME) is a frame based approach. The motion is estimated for the whole frame at once and used for motion compensated interpolation of the current frame to be coded. Thereafter, this interpolated frame can then be used as an additional reference for the conventional coding tools. Compression is achieved, since the interpolated frame is already an accurate prediction of the current frame.
DMVD achieves an average bit rate reduction of 7.9%. The average gain of DSME is with 7.8%

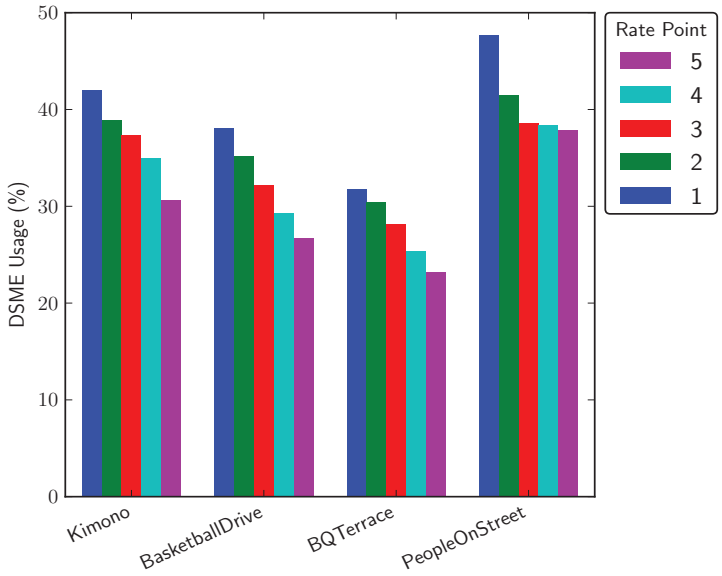


Fig. 16. Usage of the DSME frame as reference for H.264 / AVC inter prediction for different rate points.

almost the same. Interestingly, DSME outperforms DMVD for some sequences, although both approaches try to utilise similar sequence characteristics to achieve compression.
The improved compression efficiency comes along with increased computational complexity. Especially DMVD introduces high complexity due to the hierarchical motion estimation approach. However, hold-type displays, like liquid crystal displays (LCD) and plasma displays, perform motion estimation and temporal interpolation in real-time for frame rate up conversion (Choi et al., 2000). Using such algorithms, the computational complexity of DSME can be reduced.
In 2010, ITU-T Study Group 16 (VCEG) and ISO/IEC JTC 1/SC 29/WG 11 (MPEG) created the Joint Collaborative Team on Video Coding (JCT-VC) to develop a new generation video coding standard that will further reduce the data rate needed for high quality video coding, as compared to the current state-of-the-art H.264 / AVC standard. This new coding standardisation initiative is being referred to as High Efficiency Video Coding (HEVC). Due to the promising rate gains achieved with additional motion estimation algorithms at the decoder, JCT-VC initiated a tool experiment in this research field (Wien & Chiu, 2010). The goal of this tool experiment is to evaluate if techniques based on motion compensation at the decoder should be included into the HEVC standard.

7. References

Alparone, L., Barni, M., Bartolini, F. & Cappellini, V. (1996). Adaptive weighted vector-median filters for motion fields smoothing, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Georgia, USA.

Bjøntegaard, G. (2001). Calculation of average PSNR differences between RD curves, *ITU-T SG16/Q6 Output Document VCEG-M33*, Austin, Texas.

Choi, B.-T., Lee, S.-H. & Ko, S.-J. (2000). New frame rate up-conversion using bi-directional motion estimation, *IEEE Transactions on Consumer Electronics* 46(3): 603 –609.

ITU-T and ISO/IEC (2003). Draft ITU-T recommendation and final draft international standard of joint video specification, *ITU-T Rec.H.264 and ISO/IEC14496-10AVC* .

ITU-T and ISO/IEC (2010). Joint call for proposals on video compression technology, *ISO/IEC JTC1/SC29/WG11 MPEG Output Document N11113*, Kyoto.

Kamp, S., Ballé, J. & Wien, M. (2009). Multihypothesis prediction using decoder side motion vector derivation in inter frame video coding, *Proceedings of SPIE Visual Communications and Image Processing*, SPIE, Bellingham, San José, CA, USA.

Kamp, S., Bross, B. & Wien, M. (2009). Fast decoder side motion vector derivation for inter frame video coding, *Proceedings of International Picture Coding Symposium*, IEEE, Piscataway, Chicago, IL, USA.

Kamp, S., Evertz, M. & Wien, M. (2008). Decoder side motion vector derivation for inter frame video coding, *Proceedings of the IEEE International Conference on Image Processing*, IEEE, Piscataway, San Diego, CA, USA, pp. 1120–1123.

Kamp, S. & Wien, M. (2010). Decoder-side motion vector derivation for hybrid video inter coding, *Proceedings of the IEEE International Conference on Multimedia and Expo*, Singapore.

Klomp, S., Munderloh, M. & Ostermann, J. (2010a). Block size dependent error model for motion compensation, *Proceedings of the IEEE International Conference on Image Processing*, Hong Kong, pp. 969–972.

Klomp, S., Munderloh, M. & Ostermann, J. (2010b). Decoder–side hierarchical motion estimation for dense vector elds, *Proceedings of the Picture Coding Symposium*, Nagoya,

Japan. Japan, pp. 362-366. Accepted for publication.

Klomp, S., Munderloh, M., Vatis, Y. & Ostermann, J. (2009). Decoder-side block motion estimation for H.264 / MPEG-4 AVC based video coding, *Proceedings of the IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, pp. 1641–1644.

Munderloh, M., Klomp, S. & Ostermann, J. (2010). Mesh-based decoder-side motion estimation, *Proceedings of the IEEE International Conference on Image Processing*, Hong Kong, pp. 2049–2052.

Richardson, I. E. G. (2003). *H.264 and MPEG-4 Video Compression*, John Wiley & Sons Ltd., West Sussex, England, chapter 6.4.5.3.

Sullivan, G. J. & Wiegand, T. (1998). Rate-distortion optimization for video compression, *IEEE Signal Processing Magazine* 15(11): 74–90.

Suzuki, Y., Boon, C. S. & Kato, S. (2006). Block-based reduced resolution inter frame coding with template matching prediction, *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, USA, pp. 1701 – 1704.

Suzuki, Y., Boon, C. S. & Tan, T. K. (2007). Inter frame coding with template matching averaging, *Proceedings of the IEEE International Conference on Image Processing*, San Antonio, TX, USA, pp. III 409 – 412.

Werner, O. (1996). Drift analysis and drift reduction for multiresolution hybrid video coding, *Signal Processing: Image Communication* 8(5): 387–409.

Wien, M. & Chiu, Y.-J. (2010). Tool experiment 1: Decoder-side motion vector derivation, *JCT-VC Output Document JCTVC-A301*, Dresden, Germany.