International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 956372, 10 pages
doi:10.5402/2011/956372

Research Article

Decoder-Side Motion Estimation Assuming Temporally or Spatially Constant Motion

Sven Klomp, Marco Munderloh, and Jörn Ostermann

Institut für Informationsverarbeitung, Leibniz Universität Hannover, Appelstraße 9A, 30167 Hannover, Germany

Correspondence should be addressed to Sven Klomp, klomp@tnt.uni-hannover.de

Received 7 March 2011; Accepted 17 April 2011

Academic Editors: G. Camps-Valls, F. Palmieri, and L. Zhang

In current video coding standards, the encoder exploits temporal redundancies within the video sequence by performing block-based motion compensated prediction. However, the motion estimation is only performed at the encoder, and the motion vectors have to be coded explicitly into the bit stream. Recent research has shown that the compression efficiency can be improved by also estimating the motion at the decoder. This paper gives a detailed description of a decoder-side motion estimation architecture which assumes temporal constant motion and compares the proposed motion compensation algorithm with an alternative interpolation method. The overall rate reduction for this approach is almost 8% compared to H.264/MPEG-4 Part 10 (AVC). Furthermore, an extensive comparison with the assumption of spatial constant motion, as used in decoder-side motion vector derivation, is given. A new combined approach of both algorithms is proposed that leads to 13% bit rate reduction on average.

1. Introduction

All existing video coding standards, such as MPEG-2 Part 2 or H.264/MPEG-4 Part 10 (AVC), are essentially based on similar structures: the encoder estimates motion between the current frame to be coded and already encoded reference frames to exploit temporal dependencies within the video sequence. The resulting motion vectors are used to calculate a prediction of the current frame by displacing the content of the reference frames. Since only the resulting prediction error is transmitted, compression is achieved. Due to block-based motion estimation, accurate compensation at object borders can only be provided by small block sizes. However, the decoder is not able to estimate the motion vectors, and the encoder has to transmit the motion vectors in addition to the prediction error. The smaller the block size is, the more motion vectors have to be transmitted, resulting in a trade-off in bit rate reduction. It can be observed that the block size has a significant impact on the compression performance and is, therefore, limited to 4×4 pixels in AVC.

A significant amount of the total bit rate of an encoded sequence is needed to transmit the motion information, as shown in Figure 1. The percentages needed to transmit the different information, that is, mode signalling, motion vectors, coded block patterns, transform coefficients for luma and chroma, and other information, is plotted. Reasonable quality (30 dB to 40 dB) is achieved for quantisation parameters between 24 (higher quality) and 44 (lower quality). Within this range, the average amount of data used for motion vector representation is almost 20%.

Recent research shows that the data rate can be reduced by partially estimating the motion at the decoder, since no motion vectors have to be transmitted for those regions. The Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC and ITU-T considered this new research field and conducted tool experiments on this topic [1].

One approach is decoder-side motion vector derivation (DMVD, [2]), in which the motion of a block is also estimated at the decoder, instead of explicitly coding the motion vector into the bit stream. An L-shaped template of already reconstructed pixels neighbouring the current block is matched in the reference frames using the sum of absolute differences (SADs). Thus, it is assumed that the motion is constant in the spatial neighbourhood of the current block. However, the assumption is not always valid, in particular for large block sizes and object borders. Therefore, the encoder decides to either code the motion vector explicitly or use

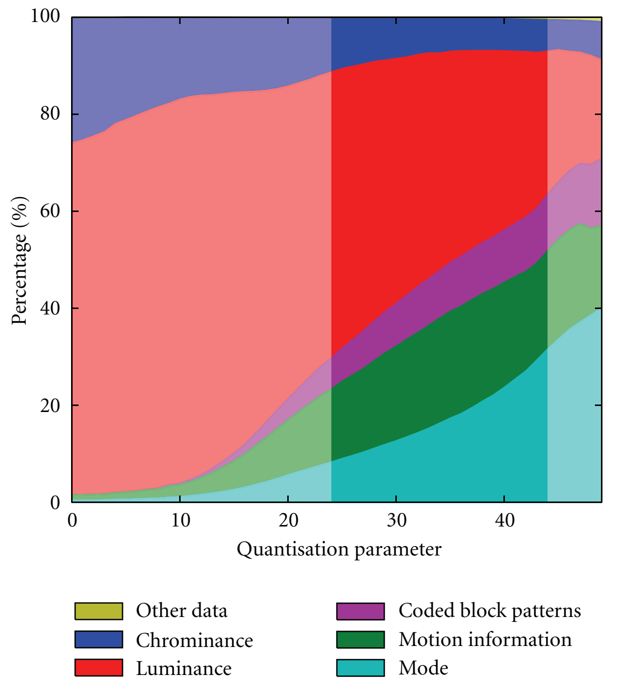


Figure 1: Rate distribution for different quantisation parameters of an AVC high profile coded bit stream with hierarchical B frames (I-b-B-b-B-b-B-b-P) for the Kimono sequence (1080p, 24 Hz). The highlighted area represents the quality range of interest (between 30 dB and 40 dB).

the derived vector by a rate-distortion optimised mode decision. In case DMVD is selected, no motion vector has to be coded, and only the mode has to be signalled.

Another approach, discussed in more detail in this paper, is decoder-side motion estimation (DSME, [3]). Hereby, a prediction of the current frame is generated by performing bidirectional interpolation using previously coded frames. Since no information of the current frame is available at the decoder, temporally constant motion is assumed. The whole frame is predicted at once, and, thus, no restrictions (e.g., to the block size) apply and any arbitrary motion estimation algorithm, like block-based [4] or mesh-based [5], may be used to create a dense motion field. Therefore, problematic areas, like object borders, can be efficiently handled. The resulting interpolated frame is then fed back into the reference picture buffer and can be used for prediction by the conventional hybrid coder. Since the interpolation is crucial for the performance of this approach, a detailed description with additional information on the motion estimation algorithm proposed in [6] is given in this paper. Furthermore, a performance comparison with the spatio-temporal autoregressive (STAR) frame rate upconversion proposed in [7] is made.

Although both approaches, DMVD and DSME, try to reduce the rate used for transmitting the motion vectors, different assumptions on the motion are made as mentioned before. DMVD assumes constant motion of spatial neighbouring pixel, whereas DSME expects constant motion over time. A combination of both approaches can lead to additional rate gains. Therefore, the different hypotheses of the motion characteristics are highlighted and evaluated in experiments, and a new combined architecture is proposed to benefit from spatially and temporally constant motion.

This paper is organised as follows. The DSME architecture is described in Section 2 with additional details on the motion estimation algorithm. In Section 3, the modifications and restrictions for the new combined approach of DSME and DMVD are discussed. The comparison of the used motion compensated interpolation with a state-of-the-art frame interpolation algorithm is shown in Section 4. Furthermore, an extensive comparison of the new combined approach with the AVC reference as well as DSME and DMVD is given in that section. The paper comes to a close with conclusions in Section 5.

2. Decoder-Side Motion Estimation Using Modified Reference List

This section explains the architecture used to allow motion estimation at the decoder, which is based on [3]. Additionally, it is shown how nonlinear motion can be compensated, although linear motion is assumed in the decoder-side motion estimation algorithm.

DSME is implemented on top of a conventional hybrid video coder, as shown in Figure 2. 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 module. In this implementation, the module contains a block-based motion estimation algorithm to interpolate the current frame F_0 from the previous and next frames. The motion vectors between F_0 and the two reference frames can be estimated by assuming linear motion, that is, 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. Thus, the approach performs best for temporally constant motion.

The frame $\hat{F}_0$ is inserted into the reference picture buffer, after the whole frame is interpolated. The tools of the conventional encoder are now able to use this frame as well as the other reference frames for prediction and coding.

In addition to this approach with the modified reference list, a pure DSME mode was proposed in [3]. For low bit rates, the interpolated DSME frame $\hat{F}_0$ may have sufficient quality and, thus, no additional residual has to be transmitted. However, the rates where pure DSME is selected are below reasonable subjective quality for the new test sequences, due to the change from CIF to HD resolution. Thus, the pure DSME mode is omitted in this implementation.

Since the proposed 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 be implemented to allow DSME also for P frames.

2.1. Incorporation of Nonlinear Motion. The DSME approach cannot compensate the motion of accelerating or decelerating objects as shown in Figure 3. The DSME algorithm estimates the true motion between the two reference frames

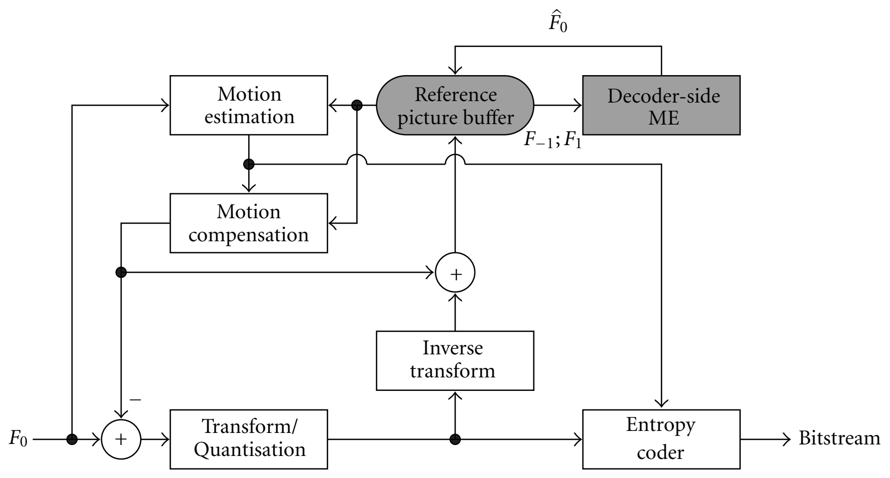


FIGURE 2: Hybrid video encoder with DSME modifications highlighted in grey.

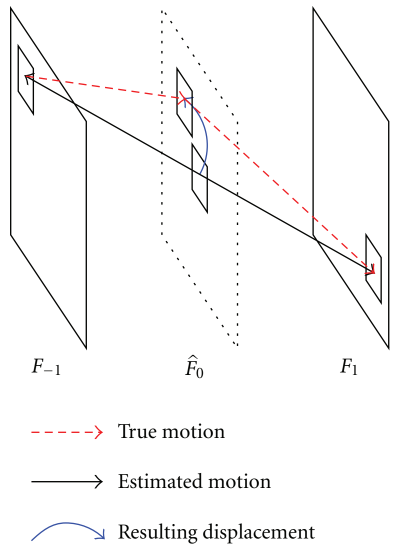


FIGURE 3: Displaced interpolation due to nonlinear motion.

and inserts the interpolated block in the centre of $\hat{F}_0$ because of the assumption of constant motion. However, the true motion drawn with dashed arrows is accelerated, and the correct position of the block is in the upper left corner. Thus, DSME is not able to create a valid prediction of the current frame although the motion estimation is successful.

To handle nonlinear motion, no distinction is made between the inserted DSME frame and the other reference frames. Therefore, the encoder also transmits the motion vector difference [8] for a block using the DSME frame $\hat{F}_0$. In the case of linear motion, the motion vector estimated by the encoder is zero, as $\hat{F}_0$ is already motion compensated. 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 temporally constant motion is wrong. The encoder can still use the wrong com-pensated block by applying motion compensation on the DSME frame and transmitting the resulting displacement, as depicted in Figure 3.

2.2. Hierarchical Motion Estimation with Vector Latching. The most crucial part of the DSME module shown in Figure 2 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 more often selected as a reference by the encoder tools.

Conventional block matching algorithms as used in, for example, AVC minimise the residual between the original frame and the motion compensated reference. From a coding point of view, it makes no difference if the estimated motion vector represents the true motion or only points to similar texture of another object. As long as the residual is small, compression is achieved. In contrast, the motion compensation in DSME is prone to wrong motion due to the assumption of linear motion. A false local minimum found during matching of a small block (Figure 4(a)) would result in a distorted frame $\hat{F}_0$, as shown in Figure 4(b). Thus, conventional motion estimation algorithms, as used in current video coding standards, are not applicable to decoder-side motion estimation.

To prevent those wrong motion vectors, a hierarchical motion estimation scheme, as depicted in Figure 5, is used. The algorithm starts with a block size of 64×64 pixel 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 levels. To accelerate the motion estimation for the large 64×64 blocks, only every second pixel values are used to calculate the matching cost. Therefore, the two reference frames F_{-1} and F_1, which are used to estimate the motion, have to be low-pass filtered in the first iteration. For all other iterations, the unfiltered reference frames are used.

At each hierarchy level, the motion vectors between the previous and the next frames (F_{-1}, F_1) are estimated using

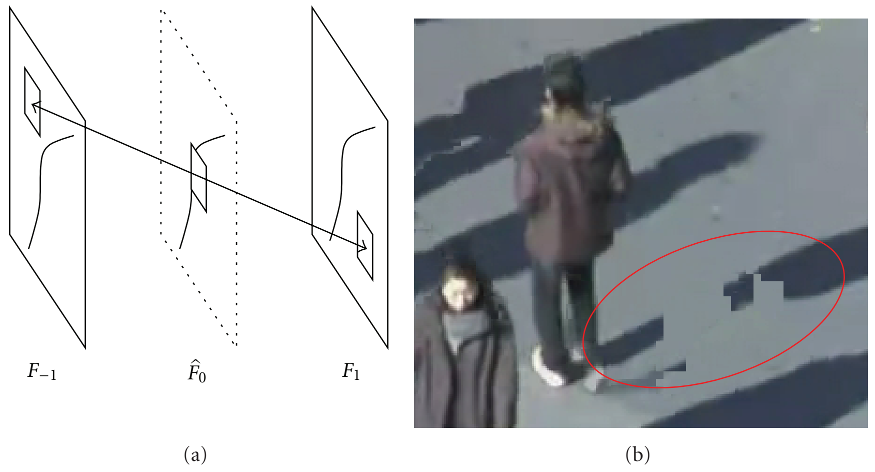


Figure 4: Bidirectional motion compensation (a) can fail as shown for a detail of the interpolated frame $\hat{F}_0$ of the PeopleOnStreet sequence (b).

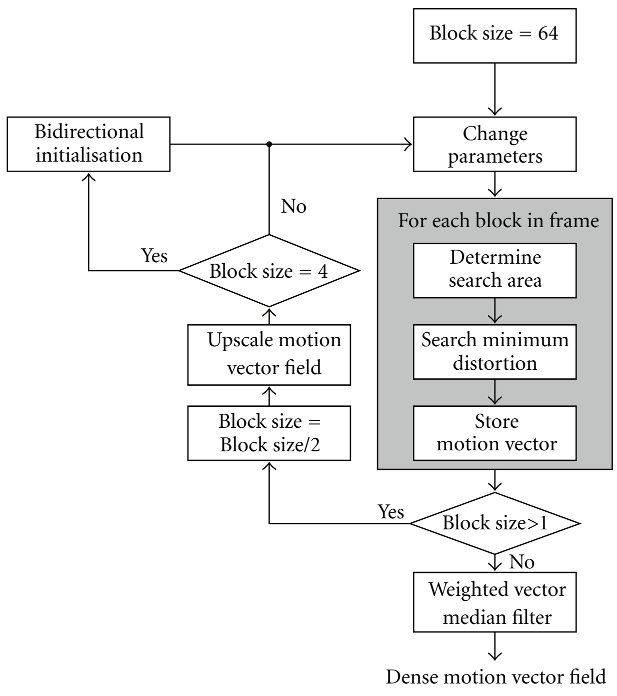

Block size = 64
Change parameters
Bidirectional initialisation
No
Yes
Block size = 4
For each block in frame
Determine search area
Search minimum distortion
Store motion vector
Upscale motion vector field
Block size = Block size/2
Yes
Block size>1
No
Weighted vector median filter
Dense motion vector field

Figure 5: Diagram of hierarchical motion estimation.

a conventional block matching algorithm, which minimises the mean of the absolute differences (MADs) between the two reference frames F_{-1} and F_1. The evaluation in Section 4.1 shows that the small gain achieved with mean of the squared differences (MSDs) as optimisation criterion is not worth the additional complexity. For blocks smaller than 16×16, the matching window of the motion estimation is 50% larger than the block size during motion compensation, for example, 12×12 matching window for a 8×8 block, to be more robust against image noise.

In [6], the search area used for the motion estimation was adapted in each hierarchy level, as depicted in Figure 6. The search area in the current hierarchy level depends on the current search range and the motion vectors of the previous hierarchy level. 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 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. This significantly reduces the amount of MAD computations compared to the search algorithm used in [3].

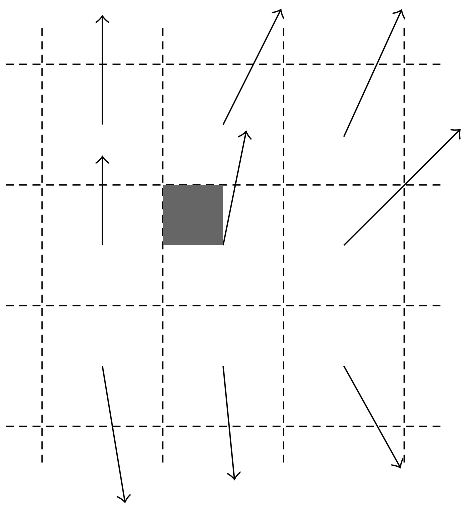
The last iteration of the forward motion search is done with half-pel accuracy using the AVC 6-tap Wiener filter [9] to interpolate subpixel values. Thereafter, bidirectional initialisation is performed to align the motion vectors to the block grid of the current frame according to [4]. This is done by selecting the motion vector that intersects the current block nearest to centre. For simplification, Figure 7 shows the selection for the one-dimensional case.

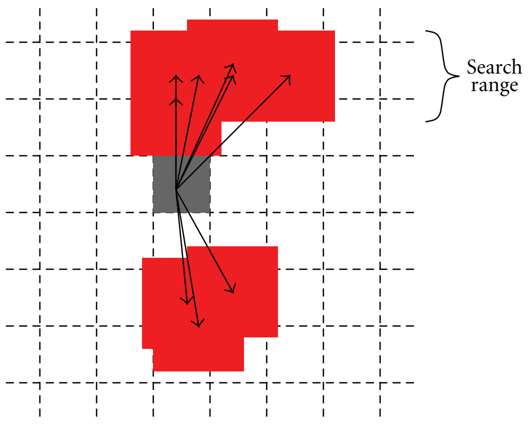
To reduce noise in the motion vector field caused by small block sizes, the motion search is switched to a candidate-based approach for the bidirectional motion search. The motion vector for the current block is hereby set to one of the surrounding motion vector candidates from the previous level without further refinement. This forces small blocks to decide which motion object they belong to and is achieved by setting the search range for those hierarchical levels to zero: the blocks "latch" to one of the motion vector candidates of the previous level. Using this technique, the resulting motion vector field can adapt to object borders more accurately.

The vector field is smoothed in the last step of the motion estimation, using a vector median filter weighted by the mean of absolute differences (MADs) of the displaced blocks [10] to eliminate outliers. After the motion vector field is estimated, the intermediate frame is predicted by averaging the displaced pixels from both reference frames.

3. Combination with Decoder-Side Motion Derivation

As described before, decoder-side motion estimation works on a frame level and has no interaction with the coding tools


(a) Previous level H_{n-1}



(b) Current level H_n

Figure 6: Search area (red) derived from previous hierarchy level.

of a conventional coder. Due to this encapsulated design, it is possible to use DSME concurrent with decoder-side motion vector derivation [2].

The main idea of DMVD is to reduce the rate for the motion information by deriving the motion vectors at the decoder by assuming spatially constant motion. In case the derivation is successful, no motion information has to be coded into the bit stream, and compression is achieved. However, the derived motion vector is not always correct and might impair the compression efficiency due to a large prediction error. Therefore, DMVD is selected adaptively for each macroblock by performing a rate-distortion optimised decision at the encoder, similar to the Lagrangian optimisation described in [11]. To transmit this decision to the decoder, additional flags within the AVC macroblock layer syntax were added in [12] to signal either the use of the motion vector derived with DMVD or the use of an explicitly coded motion vector.

In Figure 8, the decoder with both tools, DSME and DMVD, is shown. The encoder architecture presented in Figure 2 has to be modified accordingly. The first step during

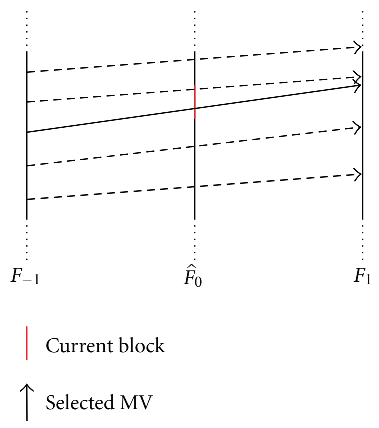


Figure 7: Alignment of the motion vectors to the block grid of the current frame to avoid uncovered and overlapped areas during motion compensation.

decoding one frame is to interpolate the DSME frame $\hat{F}_0$ as described in Section 2.2 and store it in the reference picture buffer. Thereafter, the decoding of the bit stream starts. If DMVD is signalled for the current macroblock, 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. In [13] it was shown that the performance can be further improved by using multihypothesis 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 prediction.

One limitation occurs due to the combination of DMVD and DSME. Kamp et al. [14] proposed to use the motion vectors of neighbouring blocks as candidates for motion vector derivation. 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 9, it is assumed that three reference frames are available and that the neighbouring blocks use frame F_{-2} and F_{-1} as reference. The two vectors are scaled, resulting in six candidates for the template matching.

This candidate-based predictive search is used to reduce the computational complexity of DMVD. However, the additional reference frame is already a motion compensated prediction of the current frame. Thus, the temporal distance is zero, and every candidate vector would be scaled down to zero. Therefore, the candidate-based predictive search is not used. Instead, the motion is derived by performing a full search within a search range centred to the predicted motion vector, as used in [12].

It is not to be expected that the gains of both approaches add up, since both try to reduce the rate for motion vectors. However, the different techniques used can give additional gains. On one hand, DMVD uses already decoded data from the current frame to estimate the motion. This is not possible in DSME, where the motion for the whole frame is estimated at once. On the other hand, DSME has no restrictions on the block size and can estimate motion at

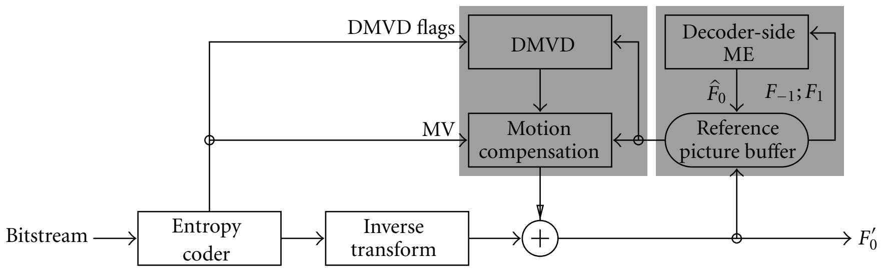


Figure 8: Simplified block diagram of a video decoder with DSME and DMVD modifications.

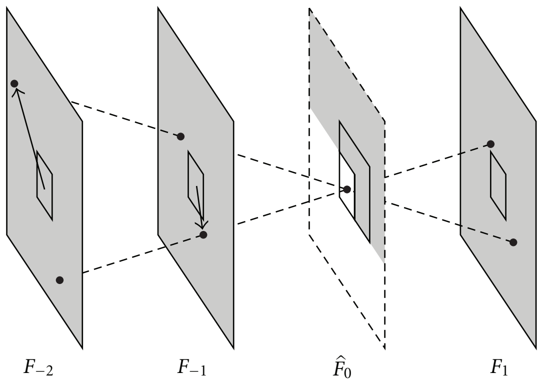


Figure 9: L-shaped template used for motion search of the current block. Already decoded frame regions are shown in grey. The candidates (black circles) for template matching are derived from motion vectors of two neighbouring blocks using linear scaling.

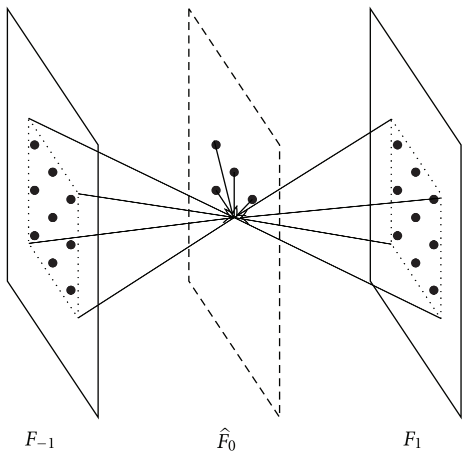


Figure 10: STAR frame interpolation: A pixel in the interpolated frame is calculated as the weighted average of spatial and temporal neighbouring pixels.

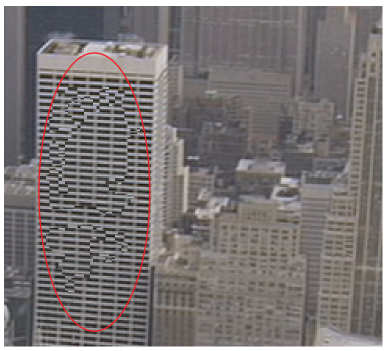


Figure 11: Motion estimation failed due to the repetitive window pattern.

object borders very well. Furthermore, the two approaches are based on different hypotheses and can complement one another: DMVD assumes spatially constant motion whereas DSME assumes temporally constant motion.

4. Experimental Results

This section evaluates the performance of the proposed methods. The interpolation accuracy of the hierarchical motion estimation approach is compared with frame interpolation methods from the literature in Section 4.1. Thereafter, the overall coding gain of the proposed architecture is evaluated and compared with DMVD in Section 4.2.

4.1. Interpolation Accuracy. As mentioned before, the motion estimation and interpolation methods are vital parts of the DSME approach. The performance directly depends on the quality of the interpolated frame. Therefore, the performance of the proposed algorithm is evaluated and compared with a state-of-the-art interpolation method.

The spatio-temporal autoregressive model (STAR) for frame rate upconversion proposed in [7] is selected for comparison since the approach has proved good PSNR gains compared to other methods, like adaptive overlapped block motion compensation (AOBMC, [15]) and 3D recursive search (3DRS, [16]). The idea of STAR interpolation is that each pixel in the interpolated frame is a linear combination of a set of pixels from the previous and next frames and also already interpolated pixels of the current frame as shown in Figure 10. The weights are derived with an iterative self-feedback training algorithm.

To compare the performance of STAR and the proposed hierarchical motion compensated interpolation, 13 test sequences with different characteristics, frame rates, and resolutions are used. Every second frame is interpolated using

Figure 12: First frames of the test sequences BasketballDrive, BQTerrace, Cactus, Kimono, ParkScene, PeopleOnStreet, and Traffic (from left to right).

Table 1: PSNR quality for the STAR interpolation method [7] and the proposed hierarchical method for several test sequences. Significant changes larger than 0.25 dB are highlighted. Furthermore, results for hierarchical motion compensation using mean of the squared differences (MSDs) are shown for comparison.

Resolution	Sequence	STAR	Hier. MC	Hier. MC (MSD)
	Mobile	36.19 dB	36.07 dB	36.06 dB
QCIF	Foreman	**39.67 dB**	37.61 dB	37.63 dB
	City	34.83 dB	**35.15 dB**	35.27 dB
	Flower	**33.45 dB**	30.51 dB	32.08 dB
CIF	Mobile	29.49 dB	**31.07 dB**	30.90 dB
	Tempete	30.86 dB	30.81 dB	30.81 dB
	Bus	27.27 dB	**29.05 dB**	29.08 dB
	City	**30.13 dB**	27.71 dB	27.62 dB
4CIF	Flower	28.70 dB	**33.56 dB**	33.61 dB
	Mobile	26.75 dB	**31.89 dB**	31.92 dB
	Spincalendar	29.51 dB	**31.99 dB**	32.11 dB
720p	Sheriff	38.08 dB	38.14 dB	38.14 dB
	City	**31.66 dB**	30.16 dB	30.20 dB

the algorithms, and the interpolation is compared with the skipped frame of the original sequence. The results are shown in Table 1.

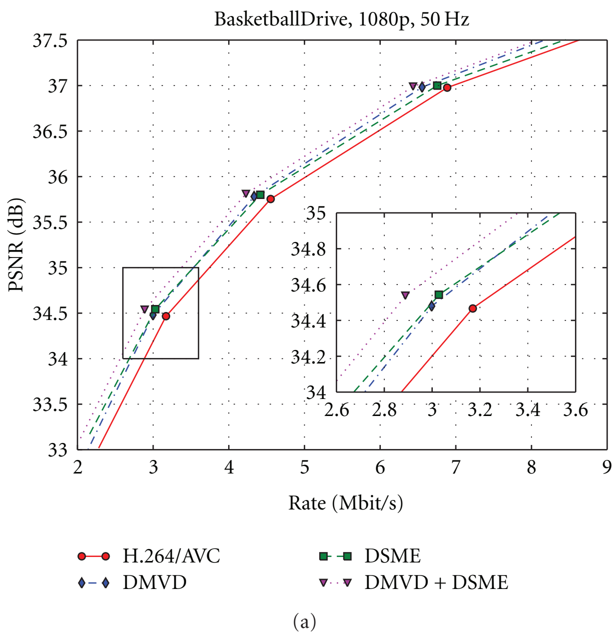
For the low resolution test sequences, no algorithm clearly outperforms the other, but hierarchical motion compensation is preferable for sequences with higher resolutions. For the 4CIF and 720p sequences, only the interpolation of the City sequence results in worse quality compared to the star interpolation. The skyscrapers with the repetitive windows makes it very difficult to find the correct correspondences. Thus, the motion compensation fails, as shown in Figure 11, and induces high errors. However, these errors are confined to a small area. The flexible DSME architecture allows to use other frames as reference for those impaired blocks. Thus, such regions will not degrade the coding performance too much.

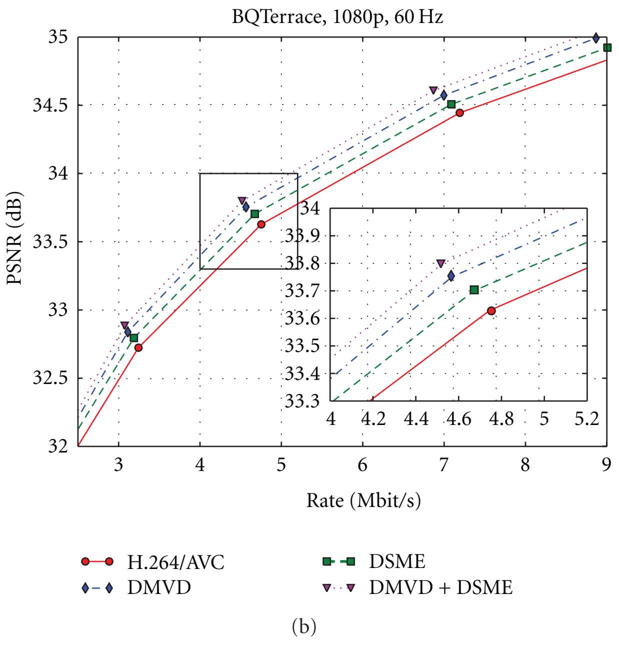
The third column in Table 1 shows the interpolation quality if MSD is used for motion estimation instead of MAD. Since the calculation of the objective quality measure

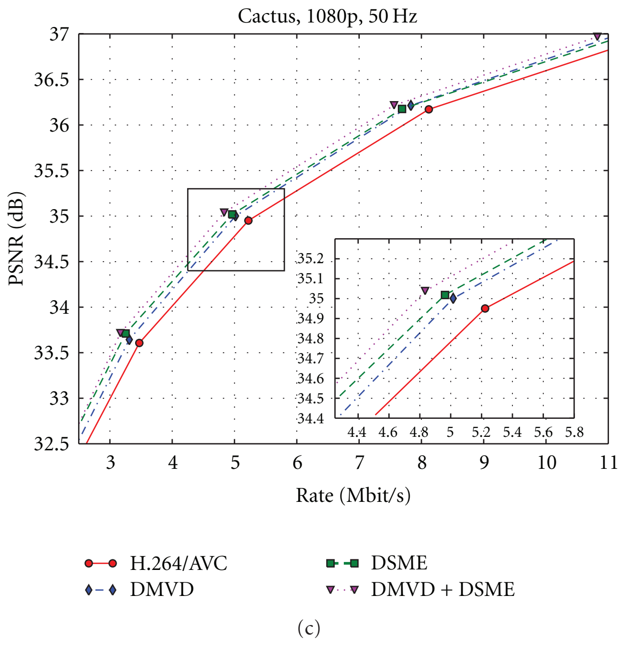
$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSD}} \tag{1}$$

also uses the squared differences, MSD is the more appropriate cost function for motion estimation. However, an average gain of 0.05 dB compared to MAD does not justify the additional complexity caused by squaring the differences. Thus, MAD is used in the following experiments despite the slight performance degradation.

4.2. Coding Efficiency. The proposed DSME approach was implemented on top of the AVC reference software JM [17]. The group of pictures (GOP) structure was set according







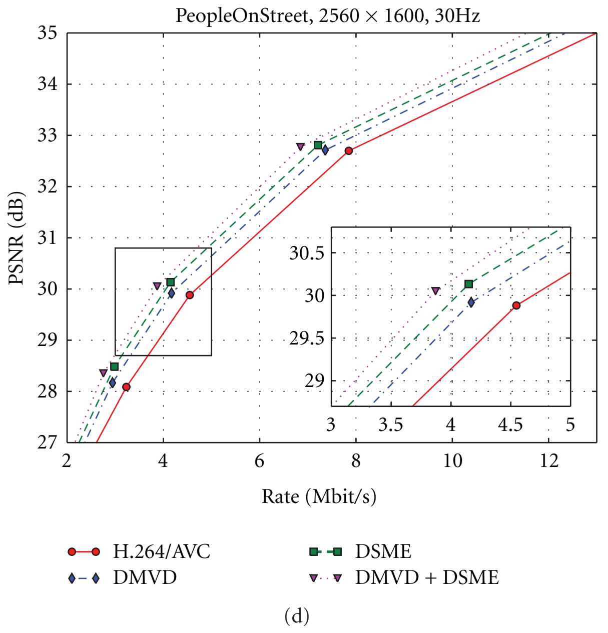


Figure 13: Rate-distortion performance of the reference AVC implementation, DMVD, DSME, and the combined approach (DMVD + DSME) for several high-definition sequences.

Table 2: BD rate gains for DMVD, DSME, and the combination of DMVD and DSME compared to AVC reference for several high-definition sequences.

Resolution	Sequence	DMVD	DSME	DMVD + DSME
1080p	BasketballDrive	−5.7%	−5.8%	−10.2%
	BQTerrace	−8.8%	−4.8%	−11.5%
	Cactus	−5.6%	−8.2%	−10.8%
	Kimono	−9.8%	−5.0%	−12.5%
	ParkScene	−8.9%	−8.5%	−13.7%
2560 × 1600	PeopleOnStreet	−9.0%	−13.0%	−17.3%
	Traffic	−7.6%	−10.1%	−13.5%

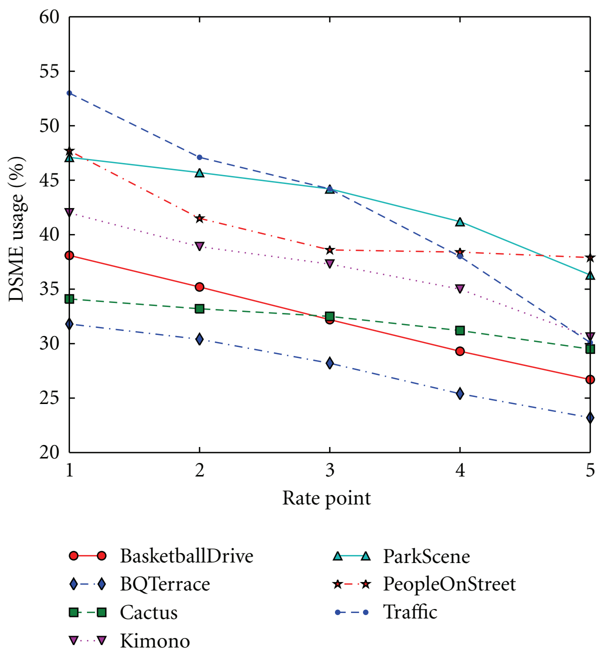


Figure 14: Usage of the interpolated DSME frame as reference for prediction in AVC. The percentage decreases for higher rate points (higher quality).

to the call for proposals [18] issued jointly by ITU-T SG16 Q.6 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG), that is, the sequences were coded with AVC high profile using an I frame every second for random access and hierarchical B frames (I-b-B-b-B-b-B-b-P). The quantisation parameters were adapted to each sequence to get similar rates.

The performance is evaluated for all test sequences with HD resolution and above that are also used by the JCT-VC group [18]. Since those sequences are quite new and not well known outside JCT-VC yet, the first frames of those sequences are shown in Figure 12. Furthermore, a short description of the major features is given in the following: All sequences were captured progressively, since interlaced video is not considered anymore for the new standard. The BasketballDrive sequence contains fast moving players causing motion blur. Interesting features of BQTerrace are the river with changing texture and the shimmering air due to the heat of some flames. Cactus contains an artificial scene of rotating and swinging objects. The sequence is very sharp with good illumination due to the experimental setup. A close-up view of a walking woman from a moving camera is shown in the Kimono sequence. The second half of the sequence shows a broader view of the scene. ParkScene contains a slow pan with fast moving cyclists. PeopleOnStreet and Traffic are both cropped to 2560 × 1600 from the original sizes 3840 × 2160 and 4096 × 2048, respectively. PeopleOnStreet shows a large pedestrian crossing with a lot of people walking in different directions. The shadows caused by the low sun is a challenge for coding algorithms. Traffic is a rather simple sequence of a busy highway. PeopleOnStreet and Traffic were both captured with a static camera.

In Figure 13, the operational rate-distortion curves for the reference AVC, DMVD, DSME, and the combination of the latter two approaches (DMVD + DSME) are plotted for four selected test sequences. To obtain an objective measurement of the average rate gains, the Bjøntegaard Delta (BD, [19]) is provided in Table 2 for all test sequences.

As shown in Figure 13(a), DSME performs well for the BasketballDrive sequence although fast motion with significant occlusions and motion blur occur in this sequence. A rate gain of 5.8% is achieved for DSME. The gain for DMVD is with 5.7% almost similar. The gains of DMVD and DSME almost add up to 10.2% bit rate reduction for the combined approach as shown in Table 2.

The water and flames in BQTerrace make the motion estimation very hard due to the non-rigid motion and missing texture. However, DSME can still save 4.8% data rate, as shown in Figure 13(b). The rate reduction for the DMVD approach almost doubles, since the approach benefits from the sequentially decoded data used for motion prediction. The combination of DMVD and DSME achieves additional performance gain even for this sequence, resulting in rate reduction of 11.5% on average.

Cactus, with the constant motion and high details in the textures of the objects, is well suited for the DSME approach. Figure 13(c) shows that DSME outperforms DMVD for almost all rate points. DMVD performs slightly better only for high rates. Thus, 8.5% bit rate reduction is achieved, whereas DMVD reduces the rate by only 5.6%. The combined approach results in bit rate reduction of 10.8%.

The proposed DSME method also works well for sequences with very high resolution, as shown for the PeopleOnStreet sequence in Figure 13(d). It clearly outperforms DMVD and achieves an average reduction of the bit rate by 13% since the vector latching proposed in [6] also estimates a very accurate motion vector field at object boundaries. However, DMVD achieves 9% rate reduction, which is also very good compared to the other sequences. The rate reduction of the combined approach is with 17.3% the largest gain for all sequences in the test set.

A common trend of all rate-distortion curves is the decreasing gain of DSME towards higher rates. The improved quality of the key frames F_{-1} and F_1 has only marginal influence on the frame estimation accuracy, and thus, the other reference frames are selected instead. Figure 14 shows how often the interpolated DSME frame is used by the conventional encoder at different rate points. The higher the rate is, the fewer the DSME frame is used as a reference.

5. Conclusions

This paper explains in detail the framework for decoder-side motion estimation while using a modified reference list. The advantage of this architecture is that any motion estimation algorithm like block- or mesh-based or even optical-flow can be used. Furthermore, a hierarchical motion estimation algorithm is described that fits the needs of a decoder-side motion estimation system.

Experimental results show that DSME reduces the bit rate by almost 8% on average, with respect to the AVC reference. A comparison with decoder-side motion vector derivation reveals that the average rate reduction for the seven test sequences is almost equal for both approaches. Interestingly, DSME and DMVD achieve the highest and smallest rate reduction for different sequences although both algorithms try to reduce the bit rate by performing some kind of motion estimation at the decoder. This behaviour is caused by the different hypotheses of the two algorithms. DSME assumes temporally constant motion whereas DMVD assumes constant motion in spatial neighbourhood.

Therefore, a combined approach, where DSME and DMVD are implemented in one coder, is proposed. Although both algorithms achieve similar compression, they can benefit from each other due to the different assumptions made in each algorithm. The combination leads to an even higher compression efficiency of average 13% bit rate reduction.

The improved compression efficiency comes along with increased computational complexity. Especially DSME introduces high complexity at the decoder due to the hierarchical motion estimation approach. However, hold-type displays, like liquid crystal displays (LCDs) and plasma displays, perform motion estimation and temporal interpolation in real time for frame rate upconversion [20]. Using such displays can significantly reduce the additional computational complexity introduced by DSME.

References

[1] M. Wien and Y.-J. Chiu, "Tool experiment 1: decoder-side motion vector derivation," in *Proceedings of the JCT-VC Output Document JCTVC-A301*, Dresden, Germany, April 2010.

[2] S. Kamp and M. Wien, "Decoder-side motion vector derivation for hybrid video inter coding," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 1277–1280, Singapore, July 2010.

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

[4] J. Ascenso, C. Brites, and F. Pereira, "Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding," in *Proceedings of the 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services,*, Smolenice, Slovak Republic, July 2005.

[5] M. Munderloh, S. Klomp, and J. Ostermann, "Mesh-based decoder-side motion estimation," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 2049–2052, Hong Kong, China, September 2010.

[6] S. Klomp, M. Munderloh, and J. Ostermann, "Decoder-side hierarchical motion estimation for dense vector fields," in *Proceedings of the Picture Coding Symposium*, pp. 362–366, Nagoya, Japan, December 2010.

[7] Y. Zhang, D. Zhao, X. Ji, R. Wang, and W. Gao, "A spatio-temporal auto regressive model for frame rate upconversion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 9, pp. 1289–1301, 2009.

[8] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression*, chapter 6.4.5.3, John Wiley & Sons, West Sussex, UK, 2003.

[9] O. Werner, "Drift analysis and drift reduction for multiresolution hybrid video coding," *Signal Processing: Image Communication*, vol. 8, no. 5, pp. 387–409, 1996.

[10] L. Alparone, M. Barni, F. Bartolini, and V. Cappellini, "Adaptively weighted vector-median filters for motion-fields smoothing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '96)*, vol. 4, pp. 2267–2270, Georgia, Ga, USA, May 1996.

[11] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for: Video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[12] S. Kamp, M. Evertz, and M. Wien, "Decoder side motion vector derivation for inter frame video coding," in *Proceedings of the International Conference on Image Processing (ICIP '08)*, pp. 1120–1123, San Diego, Calif, USA, October 2008.

[13] S. Kamp, J. Ballé, and M. Wien, "Multihypothesis prediction using decoder side motion vector derivation in inter frame video coding," in *Visual Communications and Image Processing 2009*, vol. 7257 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2009.

[14] S. Kamp, B. Bross, and M. Wien, "Fast decoder side motion vector derivation for inter frame video coding," in *Proceedings of the Picture Coding Symposium (PCS '09)*, Chicago, Ill, USA, May 2009.

[15] B. -D. Choi, J. -W. Han, C. -S. Kim, and S. -J. Ko, "Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 4, pp. 407–415, 2007.

[16] G. de Haan, P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 5, pp. 368–379, 1993.

[17] ISO/ITU-T Joint Video Team Joint Model (JM) H.264/MPEG-4 AVC software, http://iphome.hhi.de/suehring/tml.

[18] ISO/IEC JTC1/SC29/WG11 MPEG, "Joint call for proposals on video compression technology," in *ISO/IEC JTC1/SC29/WG11 MPEG Output Document W11113*, Kyoto, Japan, January 2010.

[19] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," in *ITU-T SG16/Q6 Output Document VCEG-M33*, Austin, Tex, USA, April 2001.

[20] B. T. Choi, S. H. Lee, and S. J. Ko, "New frame rate upconversion using bi-directional motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 603–609, 2000.