Decoder-Side Motion Vector Derivation for Block-Based Video Coding

Steffen Kamp, *Member, IEEE,* and Mathias Wien, *Member, IEEE*

***Abstract*—A decoder-side motion vector derivation algorithm for hybrid video coding is proposed. The algorithm is based on template matching and aims to reduce motion parameter bit-rate by re-estimating the applicable motion parameters at the decoder side. An average bit-rate savings of about 6%–8% is observed compared to the reference H.264/AVC. Decoder-side motion vector derivation was included in multiple proposals for the new High Efficiency Video Coding standard. This paper details and analyzes the algorithm and discusses its relation to other coding tools.**

***Index Terms*—H.264/AVC, High Efficiency Video Coding (HEVC), joint collaborative team on video coding (JCT-VC), key technical areas (KTA), motion estimation and compensation, moving picture experts group (MPEG), video coding experts group (VCEG).**

I. Introduction

UP TO AND INCLUDING the current state-of-the-art H.264/AVC video coding standard [1], [2], the primary method for temporal prediction in video coding is block-based motion compensation. In the predominant form of block-based motion compensation, a frame is segmented into nonoverlapping rectangular blocks of possibly different sizes. Each block is associated with one or more motion vectors comprising a horizontal and a vertical spatial displacement relative to the location of the block in the current frame, and information for identifying a previously coded and stored frame as reference frame. The prediction operation then basically consists of copying the referenced block to the block in the current frame. If more than one motion vector is associated with a block, multiple referenced regions are combined to a single prediction signal called multiple hypothesis prediction, which may provide a better adaptation to the original signal, resulting in less residual information and possibly a more efficient compression [3], [4].

The motion compensation parameters are typically determined by the video encoder and stored explicitly in the video bitstream. Most importantly, in contrast to the decoder, the encoder has access to the original uncompressed signal and is therefore able to quantify the distortion associated with the specific combinations of coding parameters. This enables the encoder to optimize the parameters, such as to minimize the overall distortion given a desired target bit-rate. This process of rate-distortion optimization, which typically is a computation intensive task, includes the determination of motion parameters using a motion estimation algorithm [5], [6]. Due to the ever increasing computational resources in modern devices, even battery-powered mobile devices are able to encode captured video in realtime.

Serving the larger and higher resolution displays available today exceeds the current communication network capacities at reasonable cost. Consequently, there is a continuing interest in more advanced video compression algorithms as confirmed by the recent start of the standardization effort of the Joint Collaborative Team on Video Coding (JCT-VC) for High Efficiency Video Coding (HEVC) [7], aiming at the same visual quality as H.264/AVC at approximately half the bitrate.

In H.264/AVC, a picture is generally composed of a set of macroblocks (MBs), with each MB comprising a 16×16 block of luma samples along with their corresponding chroma samples. The MBs within a picture are grouped into one or more slices that can be decoded independently of each other.

Inter predicted MBs may use partitionings with prediction block sizes ranging from 16×16 down to 4×4 luma samples. The prediction signal is obtained from the decoded samples of reference pictures. The reference pictures are organized in two lists. In P slices, only one list is used and each partition is associated with a single displacement motion vector (MV) and corresponding reference picture from this list. In B slices, partitions may use unidirectional prediction from either list, or bidirectional prediction with two MVs, one from each list.

For each MV in an MB partition, the index into the reference picture list is coded to the bitstream. The MVs have quarter luma pixel accuracy and are coded differentially to a motion vector predictor (MVP) which is derived from MVs in previously coded adjacent blocks. Fig. 1 illustrates the neighboring blocks A, B, C, and D that are utilized in H.264/AVC for the case when all MBs are coded as 16×16 modes, the general MVP derivation rules are found in [1].

An analysis of the bit rate distribution between different types of coded syntax elements reveals that a significant portion of the rate is spent for motion information, as shown exemplarily in Fig. 2 for individual picture types, and split up into data types for H.264/AVC coding using the GOP-8 configuration detailed later on in Section III-A.

Manuscript received April 15, 2012; revised July 11, 2012; accepted August 21, 2012. Date of publication October 2, 2012; date of current version January 8, 2013. This paper was recommended by Associate Editor F. Wu.

S. Kamp is with Panasonic Research and Development Center Germany GmbH, Langen 63225, Germany (e-mail: steffen.kamp@eu.panasonic.com).

M. Wien is with the Institute of Communications Engineering, Rheinisch-Westfaelische Technische Hochschule Aachen University, Aachen 52056, Germany (e-mail: wien@ient.rwth-aachen.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2012.2221528

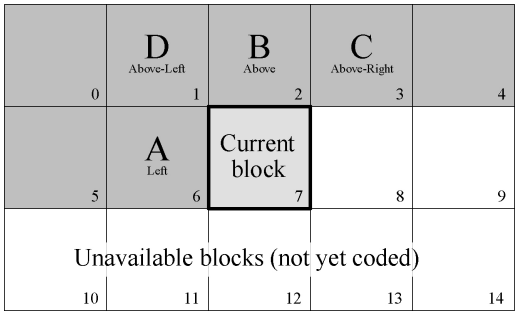


Fig. 1. For predictive coding of motion vectors, the motion vector predictor is derived from MVs of previously coded blocks (dark gray blocks). The numbers in the corners signify the blocks' coding order.

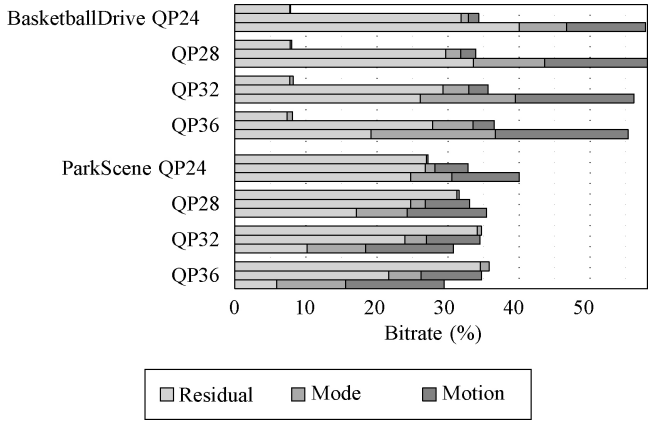


Fig. 2. Bit distribution by picture type and data type for GOP-8 coding (first 49 frames). For each sequence and QP, the group of three bars represents the bitrate fraction used for coding I (upper bar), P (middle bar), and B pictures (lower bar). For each group, the sum of the three bars is 100%.

In this paper, a decoder-side motion vector derivation (DMVD) algorithm based on template matching (TM) is proposed which aims at the reduction of motion parameter bitrate by harnessing computation capacities at the decoder. The scheme has been proposed as a tool in the call for proposals (CfP) for HEVC [8]. The presented DMVD scheme is extensively discussed in [9].

The introduction of TM-based approaches to video coding research is part of the recent development of exploring algorithms which exploit the increasing computational capabilities at the decoder side, using calculations on reconstructed signal samples to model complex signal redundancies for prediction (e.g., [10]–[12]). Notably, a scheme called template matching prediction (TMP) has been proposed in the literature [13]–[15] and builds the basis of the study presented here. TMP performs a distortion-based search between a template—reconstructed samples adjacent to a prediction target—and reconstructed samples in the current or past decoded frames in order to obtain a prediction signal without requiring the transmission of MVs. In [13], an algorithm similar to inpainting methods [16] has been proposed where the prediction for an 8×8 target block is obtained using an iterative filling approach. This originally spatiotemporal prediction approach has been developed into an algorithm that exclusively uses temporal prediction [14]. In order to further improve the prediction performance, multihypothesis prediction (MHP) was introduced to TMP in [15]. Instead of using only the best template correspondence for prediction, the weighted sum of a set of correspondences was used.

While TMP has an apparent similarity to distributed video coding [17] in that both employ a motion estimation algorithm at the decoder, a key difference is that TMP performs the identical motion estimation at the encoder in order to code a locally adaptive selection between decoder-side motion estimation and regular motion coding.

The aforementioned TMP algorithms have all been studied in the context of IPPP coding using a single reference picture and sequences at CIF or QCIF resolution. The TMP-based DMVD scheme proposed in this paper comprises a fully developed scheme including a fast search method. The presented scheme is suitable for higher spatial resolutions up to and beyond 1080p and state-of-the-art hierarchical B picture prediction structures.

The presented DMVD scheme has been developed in the context of the exploration of coding tools extending the compression performance of H.264/AVC within VCEG, with a first proposal presented to VCEG and MPEG in 2007 [18], [19] and further improvements in [20]–[22]. It was proposed as a tool in the CfP for HEVC [7] in an implementation into the VCEG KTA software, see Section III-E. The presented study inspired multiple other participants to the CfP who adopted DMVD or variations of the concept into their proposals [23]–[27]. The technology was further evaluated in a series of tool and core experiments in the development of HEVC. In the end, the tool has not been adopted into the draft specification of HEVC [28], as too much of its benefit has been resolved by other coding tools. This paper details the achievable gains relative to H.264/AVC, which served as the anchor for the whole activity. It further discusses its performance impact and highlights the relation to other coding tools that have been introduced in the context of the CfP. Transmission without losses is assumed for the simulations. As other sophisticated coding tools, DMVD is sensitive to losses. The loss effects and the handling of DMVD in such transmission scenarios are out of scope for this paper.

This paper is organized as follows. In Section II, the proposed DMVD algorithm and several design aspects are discussed. A performance analysis is presented in Section III. Conclusions are provided in Section IV.

II. Decoder Side Motion Vector Derivation

A. Template Matching

In image processing, template matching (TM) generally refers to an algorithm that tries to detect a known object within an image or scene [29]. In the context of this paper, the term will be used in a more restrictive meaning, which is common to its application in source coding systems and is further detailed in this section.

As illustrated in Fig. 3, we denote the target as the region of an image for which a prediction signal is to be obtained using a TM operation. The target is a rectangular block or

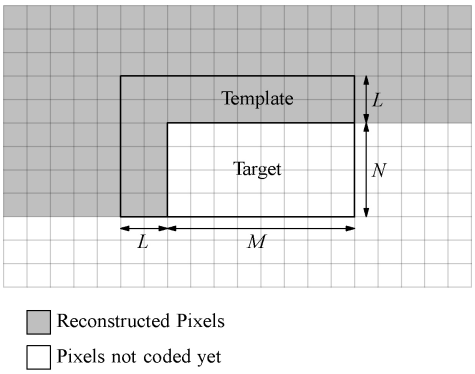


Fig. 3. Target region of size 8×4 pixels and target region of size 2. The template region is located such that it only covers samples that have already been reconstructed at the decoder.

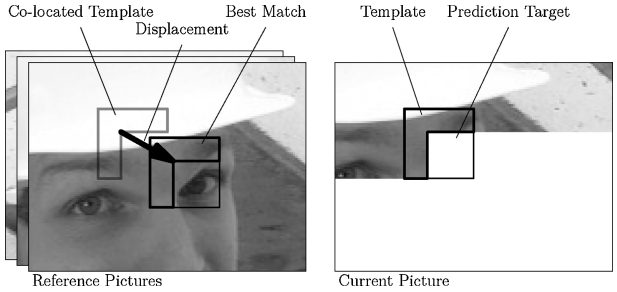


Fig. 4. Template matching operation.

partition of samples which is given by its location and size (measured in pixel units) in a picture of a sequence. The template specifies a set of already reconstructed sample values that are used in the process of obtaining the aforementioned prediction. The selection of the template samples is restricted by the MB coding order. The closest reconstructed samples to the target block form an L-shaped template region. We denote the template size as the number of pixels that the L-shaped template extends to the top and left of the target block. Fig. 3 gives an example of a target of a template of size $L = 2$.

The TM operation consists of calculating cost measures between the template region in the current picture and identically shaped displaced sample regions in the reference pictures (see Fig. 4). As it is commonly used in block-matching motion estimation, we are utilizing the sum of absolute differences (SAD) of the samples in the current and reference template regions as cost measure. Preliminary experiments showed almost no compression efficiency differences to using the computationally slightly more complex sum of squared errors. The displacement that yields the minimum template cost is considered the best match. If the motion in the respective area of the video sequence is homogeneous, this displacement can provide a good estimate for the motion of the adjacent target area. However, this is not guaranteed as TM actually estimates the displacement of the template and not the target [30]. Therefore, TM cannot fully replace regular motion vector coding. Instead, a local decision between TM prediction and regular motion coding is needed.

Conceptually, the TM search can be performed using an arbitrary block motion estimation algorithm. However, for DMVD the selected algorithm needs to be fully specified in order to produce identical results across different decoder implementations. For the initial experiments, a full search motion estimation algorithm has been used. The full search analyzes integer displacements on the original sample grid within predefined search range limits. An obvious drawback is the high computational complexity, which directly depends on the used search range. The maximum search range required to capture displacements corresponding to actual object motion is clearly sequence dependent, but a good initial search center may significantly reduce the required range to capture most displacements. In this context, the DMVD search can benefit from the regular coded motion vectors as obtained by the encoder side estimation. Until noted otherwise, the DMVD experiments in the following sections have been conducted using a full search algorithm with a search range of ± 16 pixels horizontally and vertically. It has been verified that larger search ranges can only provide negligible improvements of the coding performance while significantly increasing the required computation time. A predictive search algorithm with much lower computational complexity is presented in Section II-F.

B. Extending H.264/AVC With DMVD

The proposed DMVD scheme is discussed relative to H.264/AVC. The existing inter prediction modes are extended, such that additional flags are coded for each prediction partition that may potentially use DMVD. The flag signals whether the partition is predicted using regularly coded motion vectors or using a TM algorithm. While the approach of using per-partition flags is not generally superior to using separate additional block coding modes, it facilitates the combination of different prediction methods within one MB, e.g., in 16×8 and 8×16 MBs.

The block diagram of a hybrid video decoder with DMVD is shown in Fig. 5. If entropy decoding obtains a DMVD flag signaling to use DMVD prediction for a partition, TM is used for finding the minimum template cost among a set of spatiotemporal displacement candidates in the available reference pictures, thus directly yielding a displacement vector and an index into the ordered reference picture list. This motion information can consequently be omitted from the bitstream.

For inter coded MBs in B pictures of H.264/AVC, the usage of the prediction lists is signaled individually for each MB partition. However, subpartitions in 8×8 blocks all use the same prediction list or lists and reference pictures. With bidirectional prediction, the prediction signal is obtained by a superposition of two prediction hypotheses, one from each of the two available reference picture lists. For unidirectional prediction, the signaling includes the selection of the reference picture list from which the prediction signal is obtained. The presented design of DMVD follows this approach by restricting the template search to the corresponding reference picture list or lists. The application of DMVD in B picture MBs is proposed in [31].

The derived motion information is fed to the inter prediction module, where motion compensation (MC) is performed just as if the motion information was explicitly coded in the

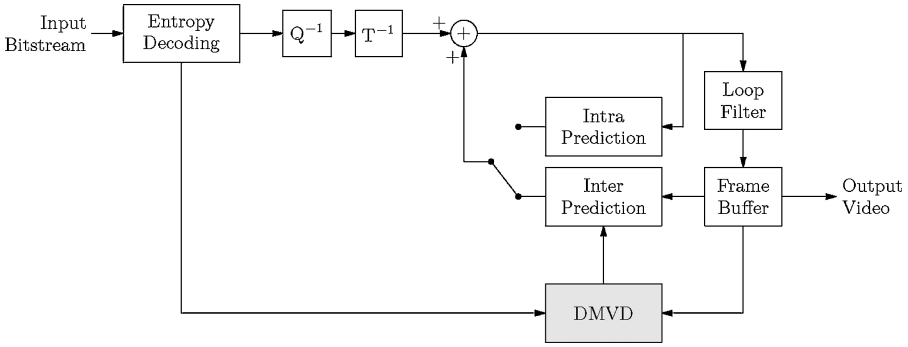


Fig. 5. Block diagram of a hybrid video decoding loop with a DMVD module. While not shown here for clarity, the intra and inter prediction modules are also fed with side information from the entropy decoding module.

bitstream. The derived information also replaces the otherwise coded data for further coding operations such as MVP calculation and obtaining candidates for the predictive search in Section II-F. If multiple MVs are derived for a partition by using DMVD, only the one with lowest template cost is used for these purposes. However, the derived information is not used for context-based adaptive binary arithmetic coding (CABAC) context determination, as this might interfere with bitstream parsing in the presence of transmission errors. If a reference picture has been lost (and possibly substituted by a concealment algorithm), the TM search might result in different motion parameters than on the encoder side. A different CABAC context selection may be the consequence, rendering the remaining parsing almost impossible until bitstream resynchronization and CABAC reinitialization occurs.

The following sections discuss specific system design aspects of DMVD as an extension to H.264/AVC. We are using the Bjøntegaard delta (BD) bitrate measure [32] for coding performance evaluation, as further detailed in Section III.

C. Target and Template

The template and target sizes are the most obvious parameters of the TM algorithm. Choosing a large template size has the drawback that the samples in the template region tend to be farther away from the target block. Therefore, the template is likely to cover image regions whose displacement is not sufficiently correlated to the target region. On the other hand, a larger template may have the advantage of yielding a more reliable displacement estimate in the presence of noise. We would therefore assume that the optimal template size needs to be large enough to be robust against noise, while not exceeding a certain size for ensuring a good correlation between template and target motion. This assumption is confirmed by our experiments with various template sizes at a fixed target size of 16×16 samples as shown in Table I(a). Generally, the optimal choice of template size is sequence dependent with best average results for a template size of 4, which is therefore used for the further experiments in this paper.

The choice of the target size is different from direct matching of the target as in regular motion estimation and coding. With target matching, the displacement is estimated by minimizing the prediction error of the target region; therefore, the prediction quality is bound to improve when decreasing the block size—at the cost of an increased number of motion vectors that need to be coded. As DMVD is signaled at the partition level, the bitrate required for DMVD signaling does not increase when obtaining motion information at finer granularity by using smaller target sizes than the partition size. However, with TM it is not guaranteed that the best match of the template corresponds to the optimal displacement for the target, especially as the block size is decreased and the template match becomes less reliable. Small target sizes, such as 1×1 or 2×2 samples have been successfully used in texture synthesis [33] and image and video coding [10], [13], [34], [35] for exploiting the self-similarity in periodic or structured textures. While small target sizes are suitable for capturing statistical properties of texture in intra coding, we can expect

TABLE I
VARIOUS TEMPLATE AND TARGET SIZES IN 16×16 PARTITIONS

Sequence	BD rate (%) IPPP coding, 49 frames, 1 reference frame			
	Template size			
	2	4	8	16
BasketballDrive	−4.34	**−4.47**	**−4.47**	−4.15
BQTerrace	**−2.13**	−1.99	−1.85	−1.75
Cactus	−1.76	−1.75	**−1.78**	−1.61
Kimono	−4.37	−4.78	**−4.94**	−4.90
ParkScene	−2.59	−2.69	**−2.74**	−2.64
BasketballDrill	−1.42	**−1.46**	−1.31	−1.06
BQMall	−0.80	**−0.84**	−0.53	−0.56
PartyScene	−0.29	−0.33	−0.31	**−0.37**
RaceHorses	**−0.86**	−0.82	−0.74	−0.58
Average	−2.06	**−2.12**	−2.08	−1.96

(a) Various template sizes with fixed target size set to 16×16.

Sequence	BD rate (%) IPPP coding, 49 frames, 1 reference frame		
	Target size		
	4×4	8×8	16×16
BasketballDrive	−2.34	−3.57	**−4.47**
BQTerrace	−0.55	−1.06	**−1.99**
Cactus	−1.28	−1.59	**−1.75**
Kimono	−2.31	−3.83	**−4.78**
ParkScene	−1.92	−2.41	**−2.69**
BasketballDrill	−1.06	**−1.47**	−1.46
BQMall	−0.53	−0.70	**−0.84**
PartyScene	−0.16	−0.23	**−0.33**
RaceHorses	−0.51	−0.69	**−0.82**
Average	−1.18	−1.73	**−2.12**

(b) Various target sizes with fixed template size set to 4.

that larger targets are preferable for inter prediction, given that objects of larger extent can typically be predicted from the reference pictures. It was found from our experiments that using target sizes identical to the partition size yields the best coding performance on average. Therefore, this setup is used in further experiments in this paper. Table I(b) gives an example for DMVD in 16×16 partitions, where a target size of 16×16, which is equal to the partition size, results in higher coding efficiency than smaller targets.

D. Subpixel Accuracy

The DMVD template search consists of an initial search (either a full search motion estimation as described above or using a fast estimation algorithm, e.g., as discussed in Section II-F) followed by an optional subpixel refinement. The refinement is initialized with the displacement of the best match of the initial search and proceeds according to the three-step search [36] by evaluating the template cost of the surrounding eight-connected half-pixel displacements and finally testing the eight-connected quarter-pixel displacements around the best half-pixel match.

When using full search as initial search, subpixel refinement accounts for almost 40% of the coding efficiency gains by DMVD [9]. With the predictive search described below, the effect of subpixel refinement is significantly lower as discussed in Section II-F.

E. Multihypothesis Prediction

In conventional block motion compensation (BMC), the application of multi-hypothesis prediction (MHP) can significantly improve the quality of the prediction signal, with most of the prediction gain obtained when averaging two jointly estimated prediction hypotheses [3].

During the TM process of DMVD, the template cost is evaluated for a set of displacement candidates. So instead of only retaining the displacement with lowest cost, a list of the N best candidates for a given target can be maintained. These candidates or hypotheses are then averaged in order to form the final prediction signal. In our experiments, the hypotheses are selected solely based on their respective template costs, as this allows us to obtain the hypotheses using a single sweep through the TM search space.

An important aspect of MHP is the selection of appropriate prediction weights for individual hypotheses. A simple method is to use equal weights for the prediction hypotheses, such that the sum of weights equals unity. Then, the weight $w_{n,1}$ for hypothesis n (with $n = 0, 1, \ldots, N-1$) is given by

$$\begin{aligned} \phi_1(n) &= 1 \\ w_{n,1} &= \phi_1(n) \cdot \left(\sum_{i=0}^{N-1} \phi_1(i) \right)^{-1} = \frac{1}{N}. \end{aligned} \tag{1}$$

In [37] and [38], the optimal MHP weights in the sense of minimizing the energy of the residual signal are derived. Under the assumption that the cross-correlation between the prediction error of the prediction hypotheses is zero, the optimal weights are inversely proportional to the energy of the residual signal of the respective prediction hypothesis [9]. While this assumption does not hold generally, it provides an acceptable simplification for the case where prediction hypotheses originate from different reference pictures. However, as the actual prediction residual is not available at the decoder, we use the square of the template mean of absolute differences (MAD) as an estimate of the residual energy. As the SAD is already used as template cost criterion, the calculation of the square MAD requires little additional computational complexity. The weights $w_{n,\lfloor \mathrm{MAD} \rfloor^{-2}}$ are given by

$$\begin{aligned} \phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}(n) &= \frac{1}{\lfloor D_{\mathrm{MAD}}(n) \rfloor^2 + 1} \\ w_{n,\lfloor \mathrm{MAD} \rfloor^{-2}} &= \phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}(n) \cdot \left(\sum_{i=0}^{N-1} \phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}(i) \right)^{-1}. \end{aligned} \tag{2}$$

Here, $D_{\mathrm{MAD}}(n)$ denotes the MAD between the current frame template region and the template region of the nth best hypothesis. As the template MAD may become zero, one is added to the denominator, avoiding division by zero and therefore bounding the weight. The addend can also be interpreted as an uncertainty added to the variance estimate due to quantization and because the error is measured on the template regions instead of the target regions, which are not available to the decoder. The floor operation $\lfloor \cdot \rfloor$ is due to the use of integer calculations.

The bitrate reductions for various DMVD hypothesis counts and weighting methods are given in Table II.[1] For single hypothesis prediction, the weighting method obviously has no impact on the coding performance with an average bitrate reduction of 2.2%. Increasing the hypothesis count to two and four increases the average gains to about 3.5% and 4.5%, respectively. Again, there is almost no difference between the weighting methods. At eight hypotheses, the behavior for different sequences is diverse. For example, while *BasketballDrill, Cactus*, and *Kimono* benefit from the additional hypotheses, others have slight to moderate losses. For *BasketballDrill* and *BQMALL*, there is only a marginal difference between prediction with four and eight hypotheses.

When losses are observed for eight compared to four hypotheses, the impact of the weighting method becomes apparent. In most cases, $\phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}$ is better than ϕ_1, but there is still room for improvement as for some sequences a hard thresholding of four hypotheses is superior to using more hypotheses.

The experiment shows the impact of a careful preselection or derivation of the number of hypotheses and the averaging weights for MHP with DMVD. It is of course straightforward to obtain sequence dependent hypothesis counts and weighting methods at the encoder. Coding of such parameters as side-information would only incur a negligible bitrate overhead as they are not likely to change much within a scene with constant characteristics and could therefore be signaled at coarse intervals. However, the encoder complexity could increase drastically if it had to determine the best among a large set of possible configurations.

[1]Note that the reported bitrate differences in Section II have been measured for the first 49 frames of the sequences. Rate-distortion results for the full sequences are provided in Section III.

TABLE II
BITRATE DIFFERENCES FOR VARIOUS HYPOTHESIS COUNTS AND WEIGHTING METHODS IN DMVD WITH MULTIHYPOTHESIS PREDICTION

Sequence	BD rate (%), IPPP coding, first 49 frames							
	MHP with relative weights ϕ_1 for hypothesis count				MHP with relative weights $\phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}$ for hypothesis count			
	1	2	4	8	1	2	4	8
BasketballDrive	−4.05	−4.80	−6.18	**−7.13**	−4.05	−4.85	−6.23	−7.08
BQTerrace	−3.17	−8.48	−11.46	−8.26	−3.17	−8.37	**−11.73**	−10.17
Cactus	−1.49	−2.50	−3.30	−3.75	−1.49	−2.44	−3.23	**−3.81**
Kimono	−4.91	−5.65	−6.88	**−7.84**	−4.91	−5.75	−6.77	**−7.84**
ParkScene	−2.98	−3.07	−3.70	−3.32	−2.98	−3.15	**−3.74**	−3.61
BasketballDrill	−1.21	−2.12	**−2.29**	−2.21	−1.21	−1.84	−2.05	−2.09
BQMall	−1.01	−1.56	−2.18	−1.97	−1.01	−1.70	−2.05	**−2.22**
PartyScene	−0.52	−1.36	−1.42	−0.69	−0.52	−1.34	**−1.50**	−0.99
RaceHorses	−0.83	−2.26	−2.78	−2.23	−0.83	−2.25	**−2.91**	−2.70
Average	−2.24	−3.53	−4.47	−4.16	−2.24	−3.52	−4.47	**−4.50**

F. Predictive Search Algorithm

A candidate-based predictive search algorithm for DMVD is proposed [39] as a replacement for the computationally complex full search. The principle of the candidate-based search is based on the recursive block-matching algorithm described in [40]. For each DMVD target, a set of unique displacement vectors is composed for which the TM costs are evaluated. Ideally, the number of vectors in the set should be relatively small in order to significantly reduce the complexity of DMVD. Several candidate types have been evaluated for their suitability:

1) *Spatial Neighbors:* With the assumption of a high spatial motion correlation in typical video sequences, the MVs of blocks adjacent to the currently estimated DMVD target are used as candidates. Specifically, we chose the MV from the partitions that are used for calculating the MVP (Fig. 1).

2) *Zero Vector:* While object motion is very common in video material, there are also frequent cases of static background or objects. Accordingly, a zero motion vector candidate could be expected to have a high probability of occurrence on such material. However, our analysis showed that the zero motion candidate actually tends to slightly reduce the coding efficiency. In areas without motion, the spatial neighbors typically include the zero vector, so it has not been used as a candidate in our further experiments.

3) *Motion Vector Predictor:* Another possible candidate is the H.264/AVC MVP, which is derived independently for the horizontal and vertical vector components using median operations. Due to the component-by-component operation, there is no guarantee that the values of both components originate from the same neighboring vector, and the resulting candidate may therefore not correspond to an MV actually found in the spatial neighborhood. Consequently, the MVP candidate did not contribute significantly to an increased coding efficiency and has not been used in our further experiments.

4) *Colocated Temporal Neighbor:* As motion fields typically exhibit a high temporal correlation, it is straightforward to include MV candidates taken from previously coded pictures. The colocated MV, found at the same spatial position as the current target in the first reference picture of the reference list corresponding to the required prediction direction is used as temporal candidate.

5) *B Picture Candidates:* Additional candidates are introduced for B picture coding, obtained as the MVs from the opposite prediction list of neighboring partitions. This method is applicable for both spatially and temporally neighboring blocks. However, in order to apply these candidates to the current block, a scaling according to (3) is used in order to adjust the MV to the reference pictures available to the current block.

In the predictive search, a candidate set may only contain unique candidates, i.e., either the spatial displacement or the associated reference picture must be different for any pair of candidates from the set. Thus, a template is never tested twice during the search, and it is assured that prediction hypotheses are unique in DMVD with MHP. If for any target the number of candidates is less than the required number of hypotheses, the averaging weights of the hypotheses are adjusted as if the missing hypotheses were identical to the hypothesis with the lowest template cost.

For partitions in B pictures, the predictive search is carried out independently on both reference lists if the partition uses bidirectional prediction, or on the respective reference list for unidirectional prediction.

The candidates described so far are connected to a specific reference picture. In order to adjust the candidates to other possible reference pictures, each candidate is linearly scaled with respect to the temporal distances between the frames in question, assuming uniform, nonaccelerating motion [39]. This follows a similar rationale as the temporal direct mode of H.264/AVC. Given an original candidate vector $\boldsymbol{v} = (v_x, v_y, v_t)$, we obtain the scaled candidate $\boldsymbol{v}'$ for a desired temporal distance v_t' similarly to the temporal direct mode vector derivation as

$$v_x' := \mathrm{round}\left[v_x \frac{v_t'}{v_t}\right] \quad v_y' := \mathrm{round}\left[v_y \frac{v_t'}{v_t}\right]. \tag{3}$$

Here, the round[·] operation rounds the resulting vector components to quarter pixel accuracy, which is the finest subpixel accuracy of MVs in H.264/AVC.

With the full search DMVD algorithm, the initial search is performed on a full pixel grid. A subsequent subpixel refinement of the best full-pixel candidates results in a sig-

TABLE III
COMPARISON OF AVERAGE BITRATE REDUCTIONS BETWEEN FULL SEARCH AND PREDICTIVE SEARCH DMVD IN IPPP CODING

Sequence	BD rate (%) IPPP coding, first 49 frames	
	Full search DMVD Search range ±16	Predictive search DMVD
BasketballDrive	**−7.08**	−5.98
BQTerrace	−10.17	**−12.00**
Cactus	**−3.81**	−2.70
Kimono	−7.84	**−8.32**
ParkScene	−3.61	**−3.89**
BasketballDrill	**−2.09**	−1.81
BQMall	**−2.22**	−1.80
PartyScene	−0.99	**−1.63**
RaceHorses	−2.70	**−3.05**
Average	−4.50	**−4.58**

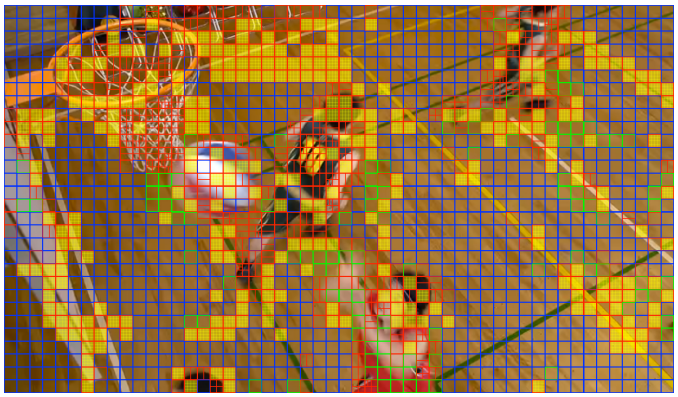


Fig. 6. DMVD utilization at QP 32 for the tenth frame of *BasketballDrill*. Shaded blocks are using DMVD prediction.

nificant improvement of the compression efficiency as described in Section II-D. In the predictive search, the initial candidates are obtained from previously coded blocks and are therefore already at subpixel accuracy. Consequently, it can be expected that the effect of a subsequent subpixel refinement is not as important as for the full search algorithm. Regarding the computational complexity, omitting the subpixel refinement is very attractive, as the refinement incurs a high relative complexity increase if the number of initial candidates is low. The impact of subpixel refinement is discussed in Section III.

With an encoder side block-by-block mode decision, predictive MV search algorithms may actually have a better rate-distortion (RD) performance than full search algorithms due to the typically more homogeneous MV field of the predictive search and the associated lower bitrate for MV coding [41]. While the homogeneity of the MV field does not have a direct impact on the bitrate required for DMVD coding, there may still be a benefit. Due to the loose correlation between template cost and actual prediction quality, full search DMVD may yield an MV that, although it corresponds to the best template cost within the search range, gives a suboptimal prediction signal. The predictive search on the other hand starts with a set of likely motion candidates—based on the assumption of temporally or spatially consistent motion—and uses the template cost as criterion for selecting MVs from this set. Additionally, due to using candidates from neighboring blocks and candidate scaling, the predictive search may cover a much larger (but sparse) search range while being operated at a lower computational complexity than the full search.

For a comparison of full search DMVD versus predictive search DMVD, we restrict the discussion to IPPP coding, results for GOP-8 coding can be found in [9]. For a full search, a search range of ±16 pixels, MHP with eight hypotheses, and relative weights $\phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}$ has been used. For a predictive search, the following DMVD configuration provided the best results according to our experiments: a candidate set consisting of three spatial candidates (taken from partitions A, C, and D in Fig. 1) plus the colocated temporal candidate. The candidates are scaled according to (3), with 1/4-pixel refinement being performed on the scaled candidates. MHP is done using four hypotheses and relative weights $\phi_{\lfloor \mathrm{MAD} \rfloor^{-2}}$.

The results in Table III show a slight advantage for the predictive search compared to full search (4.58% versus 4.50%). Consequently, the predictive search is not only an attractive alternative to full search in terms of computational complexity, but also tends to provide higher coding efficiency.

The proposed DMVD has similarities with the methods of motion representation in HEVC [28]. For motion vector merging [42], a list of up to five candidate motion vectors is built, including spatial and (scaled) collocated temporal neighbors. The applicable motion vector is signaled by a candidate list index in the bitstream. Advanced motion vector prediction (developed from motion vector competition) [43], uses a smaller list of candidate motion vectors with the motion vector difference and the applicable predictor being signaled in the bitstream. The predictive DMVD search algorithm employs a similar set of candidates but omits the signaling.

G. Entropy Coding

In the context of this paper, all simulations have been performed using CABAC entropy coding.[2] All DMVD-related syntax elements are binary flags which therefore do not require a specific CABAC binarization process. As an initial choice that was used in all simulations in this paper, DMVD flags are coded using a single context model that is initialized to equal binary symbol probability. A further investigation of conditional probabilities showed that the probability of choosing DMVD increases with the number of neighboring blocks already using DMVD (see Fig. 6 for an example of DMVD utilization in a picture). However, a five context design based on this observation provided only negligible additional compression efficiency [9]. Consequently, the simpler single context design suggests a better tradeoff between coding efficiency and implementation complexity.

III. PERFORMANCE ANALYSIS

A. Coding Conditions

The DMVD algorithm described in Section II has been implemented into the key technical areas (KTA) 2.6r1 software

[2]A further developed version of CABAC has been adopted as the entropy coding method in the committee draft of HEVC [28].

[44]. The KTA software was developed by the video coding experts group (VCEG) based on the joint model (JM) 11.0 reference software for H.264/AVC [45] as a test bed for video coding tools extending the capabilities and compression performance of H.264/AVC. In order to support DMVD, several additions are required to the H.264/AVC bitstream syntax which have been described in [8]. This section describes the simulation conditions used for evaluating the DMVD algorithm.

The simulation conditions chosen here follow the common conditions for coding efficiency experiments as recommended by VCEG [46]. BD rate savings [32] are used for reporting performance differences, which give an average bitrate reduction (in %) for identical peak signal-to-noise ratio (PSNR) between two coding configurations. The BD rate is computed from measurements of the bitrate and average PSNR at multiple RD points. In this paper, we are always comparing DMVD enabled versus DMVD disabled with an otherwise identical configuration, and by obtaining four RD points for each configuration using the quantization parameter (QP) settings detailed below. That is, a negative BD-rate value indicates that enabling DMVD improves the coding performance.

For coding experiments, the IPPP and the GOP-8 prediction structures are used. For IPPP coding, the first picture of a video sequence is coded as I picture and all subsequent pictures are coded as P pictures, allowing a low encoder-decoder delay. In GOP-8 coding (also referred to as IbBbBbBbP coding), the first picture of a video sequence is coded as I picture, every eighth picture is coded as P picture and hierarchical B pictures [47] are used in between I and P (or P and P). At regular intervals of about 1 s, I pictures are coded instead of P pictures in order to meet the random access requirements specified in [7]. Compared to IPPP, GOP-8 has a higher encoder–decoder delay but typically achieves a higher compression efficiency. For both prediction structures a maximum of four reference pictures per coded picture was used.

In the following, deviations from the VCEG common conditions are listed and detailed.

1) *Inter Polation Filter:* The original H.264/AVC subpixel inter polation filter was replaced by an improved filter that uses the same filter coefficients but performs integer rounding as late as possible in the computation. While this increases the complexity, it was found that rounding effects in the original H.264/AVC design can play a detrimental role on coding performance due to rounding error accumulation over time [48], [49]. It was further noted that there is an interrelation with hypothesis averaging in MHP as used for B pictures and multihypothesis DMVD [50]. While weighted prediction methods can be used as a countermeasure to rounding error accumulation [51] without requiring changes to the original design of H.264/AVC, a high-precision inter polation filter based on the H.264/AVC inter polation filter was added to the KTA software 1.9 which—although not compliant with the H.264/AVC specification—also alleviates such rounding effects.

2) *Search Range:* The search range for regular encoder side motion estimation has been set to ±128 pixels horizontally and vertically in order to capture large object displacements or camera movements in the test video sequences.

3) *Quantizer Settings:* In order to speed up the encoder, rate-distortion optimized quantization (RDOQ) [52] and delta QP consideration were turned off. While RDOQ and delta QP coding generally improve the coding efficiency, the implementation in KTA requires a high computational complexity at the encoder. We have also verified that the interrelation with DMVD is rather minimal as these techniques are not directly related to motion coding.

Compared to [46], a different set of QP values was chosen that better match the anticipated high bitrate operating points for high definition material in broadcast television or Blu-ray Disc encodings, currently ranging at around 6–12 Mbit/s and 20–40 Mbit/s, respectively. The chosen QP values are QPI = 24, 28, 32, 36 for intra pictures, with QPP = QPI + 1 for P pictures and QPB_0 = QPI + 2, QPB_1 = QPI + 3, and QPB_2 = QPI+4 for B pictures at different hierarchical prediction levels.

TABLE IV
AVERAGE BITRATE REDUCTIONS FOR THE FULL LENGTH SEQUENCES

Sequence	BD rate (%), IPPP coding	
	DMVD, no subpixel refinement	DMVD, with subpixel refinement
BasketballDrive	−7.40	**−7.68**
BQTerrace	−13.19	**−14.38**
Cactus	**−7.10**	−5.59
Kimono	−9.41	**−9.86**
ParkScene	−4.14	**−4.97**
BasketballDrill	**−3.96**	−3.32
BQMall	−4.69	**−4.83**
PartyScene	**−3.54**	−3.53
RaceHorses	−4.95	**−5.93**
Average	−6.49	**−6.67**

(a) IPPP configuration.

Sequence	BD rate (%), GOP-8 coding	
	DMVD, no subpixel refinement	DMVD, with subpixel refinement
BasketballDrive	−8.60	**−10.38**
BQTerrace	−8.90	**−8.98**
Cactus	**−7.54**	−7.48
Kimono	−10.71	**−12.38**
ParkScene	−7.12	**−8.13**
BasketballDrill	−5.10	**−5.36**
BQMall	**−7.05**	−7.03
PartyScene	−3.42	**−3.48**
RaceHorses	−4.09	**−5.43**
Average	−6.95	**−7.63**

(b) GOP-8 configuration.

B. Rate-Distortion Results

Average bitrate differences for DMVD on the full length sequences are given in Table IV, exemplary RD plots for *BasketballDrive* and *ParkScene* are shown in Fig. 7.

1) *IPPP Coding:* An average bitrate reduction of 6.67% with subpixel refinement and of 6.49% without subpixel refinement is observed. Bitrate reductions are mostly consistent across the whole bitrate range (see Fig. 7).

2) *GOP-8 Coding:* An average bitrate reduction of 7.63% is obtained for DMVD with subpixel refinement. Without

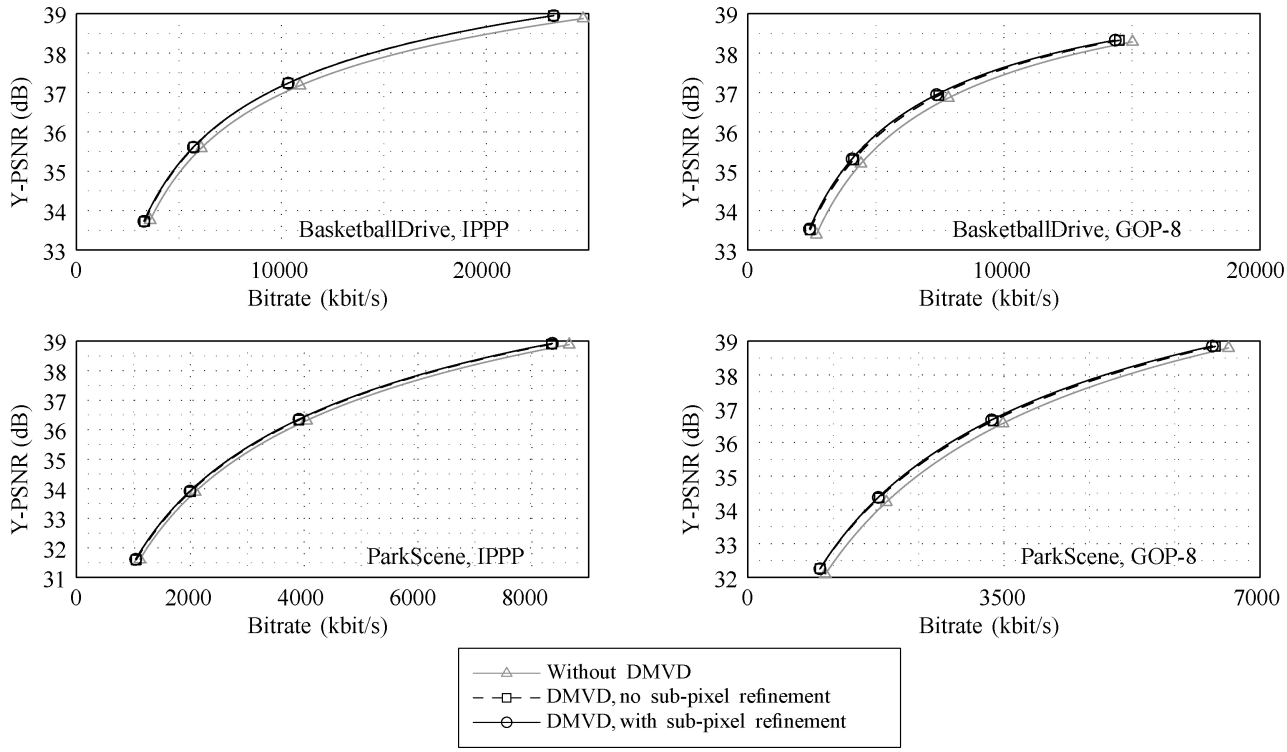


Fig. 7. Rate-distortion curves for *BasketballDrive* and *ParkScene* simulations in (a) IPPP and (b) GOP-8 configurations.

TABLE V
PERCENTAGE OF SAMPLES IN INTER CODED FRAMES WHERE DMVD WITH SUBPIXEL REFINEMENT IS USED FOR PREDICTION

Sequence	IPPP configuration at QP				GOP-8 configuration at QP			
	24	28	32	36	24	28	32	36
BasketballDrive	38%	31%	29%	26%	25%	23%	25%	28%
BQTerrace	60%	46%	46%	44%	27%	16%	15%	17%
Cactus	23%	16%	13%	11%	21%	17%	16%	14%
Kimono	36%	33%	31%	27%	30%	26%	24%	22%
ParkScene	32%	26%	21%	18%	22%	21%	22%	26%
BasketballDrill	27%	23%	18%	14%	28%	23%	19%	16%
BQMall	34%	32%	30%	25%	23%	21%	19%	19%
PartyScene	47%	41%	38%	37%	24%	23%	20%	17%
RaceHorses	38%	33%	30%	24%	28%	28%	27%	27%

subpixel refinement an average reduction of 6.95% is still observed. It is further observed from Fig. 7 that coding efficiency improvements are typically higher toward lower qualities and not nearly as constant over the whole bitrate range as for IPPP coding.

The DMVD utilization, i.e., the percentage of samples in inter coded frames where DMVD is selected for prediction by the encoder-side RD optimized mode decision, is summarized in Table V. While Table V only shows results for DMVD with subpixel refinement, percentages for DMVD without refinement have been found to be similar. The average DMVD utilization over all sequences and QPs for IPPP coding is 30%, for GOP-8 coding the average is 22%.

In both examined coding structures, enabling DMVD results in significant bitrate reductions on the test set, ranging from 3.42% (*PartyScene*, GOP-8, DMVD without subpixel refinement) to 14.38% (*BQTerrace*, IPPP, DMVD with subpixel refinement). On average, reductions of about 7% were observed and coding efficiency gains were typically higher for 1080p than for WVGA sequences. All simulations shared the same settings for DMVD target block sizes and template size. MHP settings and predictive search configuration were different for IPPP and GOP-8 but were not otherwise varied among the sequences. Only a minor difference between DMVD with and without subpixel refinement is observed, suggesting that the variant without subpixel refinement is preferable given its lower computational complexity as discussed in Section III-D.

C. Visual Impression

In both IPPP and GOP-8 configurations DMVD tends to give a smoother impression of the scene, mainly in areas with global motion or in the proximity of objects moving relative to a moving background. This effect is, e.g., visible on the walls of *BasketballDrive* in front of which the basketball players move. At QPs 28 and higher of the sequences without DMVD and especially in GOP-8 configuration, parts of the walls tend to freeze for short periods of time while the camera pans across the scene. This effect was found to be significantly less pronounced in the DMVD sequences. Similar characteristics are found in the background of the first part of *Kimono* and in the river section of *BQTerrace*.

In the IPPP configuration, DMVD also has some slight advantages in textured areas, such as the roof in the second part of *Kimono* or the background trees in *ParkScene*. In the sequences without DMVD, blocks in those areas are only occasionally updated with residual information resulting in a sudden, localized switching. Such regions often have a slightly smoother appearance in the DMVD sequences.

D. Complexity Assessment

This section provides a complexity estimation of DMVD using a lower and an upper complexity bound. While we focus on the normative application of the algorithm at the decoder, the presented bounds can also be used for evaluating the complexity increase at the encoder, where DMVD is applied during mode decision. However, at the encoder the complexity increase depends on encoder design aspects such as fast motion estimation and mode decision algorithms and is therefore more difficult to evaluate. Here, the upper complexity bound is obtained using actual run time measurements of our decoder implementation, which is focused on extensibility for research purposes and is not very well optimized for run time. For example, motion compensated samples that are used for TM undergo a redundant buffering step for simpler reuse of the existing MC software module, intermediate TM results are stored in linked lists requiring a memory allocation for each insertion, etc. We would therefore expect shorter run times for an optimized software implementation. In order to set a lower bound for the complexity, we measured the number of samples that were fetched from reference frames for the purpose of performing the TM calculations in the decoder. We assume that obtaining samples for TM is similar to MC in terms of reference memory access operations and subpixel inter polation calculations. The number of reference frame sample accesses required for TM is then set in relation to the number of sample accesses required for regular BMC to estimate the complexity increase due to DMVD. This lower bound disregards storage of DMVD MV hypotheses or decision logic in the search algorithm, correspondingly the actual complexity is expected to be higher. Obviously, these bounds can only give a rough estimate of the complexity that would be expected in a hardware implementation. In particular, the memory bandwidth increase due to accessing reference memory for TM purposes is a critical issue for decoder implementation. However, a more detailed complexity evaluation is out of the scope of this paper.

For derivation of the lower complexity bound, it is assumed for simplification that all samples of a inter coded picture are coded using MC (i.e., either using DMVD or regular BMC, without any intra coded blocks). Then, we can define the number of luma samples in a picture as a complexity index representing the complexity for predicting the luma component of a picture. That is, given a picture width of P_x and height of P_y samples, the complexity index of MC is defined as

$$C_{\mathrm{MC}} = P_x \cdot P_y. \tag{4}$$

For the experiments conducted here, TM was performed on the luma signal components only. The number of luma samples in a template for a DMVD predicted block of mode $m \in \{16\times16, 16\times8, 8\times16, 8\times8\}$ with target width M_m, target height N_m and template size L_m (see Fig. 3) is given by

$$T_m = L_m \cdot (L_m + N_m + M_m). \tag{5}$$

By measuring for each DMVD mode m the number of TM cost calculations c_m performed as part of the TM search during decoding of a picture, we can obtain the number of luma samples accessed from reference pictures in a manner similar to regular MC. This is defined as the complexity index of TM

$$C_{\mathrm{TM}} = \sum_m c_m \cdot T_m. \tag{6}$$

Then, the fraction $C_{\mathrm{TM}}/C_{\mathrm{MC}}$ gives an approximation of the time and memory bandwidth requirements of TM relative to regular MC. In combination with an estimate of the fraction τ_{MC} of decoding time required for MC in a typical H.264/AVC decoder, an estimate of the added complexity when using DMVD as an extension to H.264/AVC is given by

$$\delta_{\mathrm{TM}} = \tau_{\mathrm{MC}} \frac{C_{\mathrm{TM}}}{C_{\mathrm{MC}}}. \tag{7}$$

Exemplarily, $\tau_{\mathrm{MC}} = \frac{1}{4}$ has been chosen in this paper, based on decoder complexity studies in the literature [53]–[55].

Given that δ_{TM} only includes the complexity of TM operations and neglects algorithm logic and the actual prediction operation[3] it provides a lower bound for the complexity increase caused by DMVD.

As outlined previously, we obtain an upper bound of the additional complexity as the empirically found time ratio between the run time t_{DMVD} required for decoding a DMVD bitstream and the run time t_{anchor} for decoding the corresponding anchor bitstream for the same sequence and QP (i.e., obtained with the same encoder settings but DMVD disabled)

$$\delta_{\mathrm{time}} = \frac{t_{\mathrm{DMVD}}}{t_{\mathrm{anchor}}} - 1. \tag{8}$$

For time measurements, each bitstream was decoded three times, recording the total CPU time spent in user mode of each decoding process and computing the median of the three values. All bitstreams were decoded on the same machine with only one decoding process running at a single time. Measured decoding times were consistent among the three runs with a maximum absolute deviation from the median value of 0.2% for IPPP coding and 0.1% for GOP-8 coding.

In addition to the described complexity bounds, the fraction f_{area} of samples predicted using DMVD in inter coded slices is also reported in the results. However, f_{area} is not considered to be a suitable measure of the complexity increase, as the number of samples in the template does not linearly scale with the number of samples in the target. That is, given the same f_{area} the number of computations required for DMVD is dependent on the distribution of DMVD among the partition sizes.

Table VI exemplarily shows the complexity estimates and time ratio measurements for IPPP and GOP-8 coding of *BasketballDrive* and *ParkScene* (results for the other sequences are similar and can be found in [9]). Typically, δ_{time} is larger than $\delta_{\mathrm{TM},\tau=\frac{1}{4}}$ by a factor of 2–4 for both coding structures. This discrepancy is due to the simplifications used for deriving the lower bound and the time measurements using an unoptimized decoder implementation as upper bound. We would expect the true complexity increase in an optimized implementation

[3]In the case of single hypothesis DMVD, the prediction is performed just as a regular MC operation and therefore does not impose additional complexity compared to non-DMVD decoding. However, with MHP DMVD, multiple prediction operations and hypothesis averaging calculations have to be performed which are not considered in (7).

TABLE VI
COMPLEXITY MEASUREMENTS AND ESTIMATES

Sequence	QP	f_{area}	$\frac{C_{\text{TM}}}{C_{\text{MC}}}$	$\delta_{\text{TM},\tau=\frac{1}{4}}$	δ_{time}
		Without subpixel refinement			
BasketballDrive	24	0.39	3.51	0.88	3.29
	28	0.32	2.61	0.65	2.65
	32	0.28	2.08	0.52	2.30
	36	0.24	1.60	0.40	1.88
ParkScene	24	0.36	3.37	0.84	2.84
	28	0.30	2.54	0.63	2.40
	32	0.23	1.77	0.44	1.83
	36	0.19	1.34	0.34	1.52
		With subpixel refinement			
BasketballDrive	24	0.38	65.51	16.38	34.56
	28	0.32	50.07	12.52	29.23
	32	0.30	42.86	10.71	26.84
	36	0.27	35.98	8.99	24.07
ParkScene	24	0.32	53.19	13.30	26.56
	28	0.27	39.99	10.00	21.91
	32	0.21	29.32	7.33	17.38
	36	0.18	24.16	6.04	15.78

(a) IPPP configuration.

Sequence	QP	f_{area}	$\frac{C_{\text{TM}}}{C_{\text{MC}}}$	$\delta_{\text{TM},\tau=\frac{1}{4}}$	δ_{time}
		Without subpixel refinement			
BasketballDrive	24	0.21	1.28	0.32	0.95
	28	0.21	1.03	0.26	0.86
	32	0.21	0.84	0.21	0.81
	36	0.24	0.75	0.19	0.82
ParkScene	24	0.21	1.30	0.33	0.89
	28	0.20	1.00	0.25	0.78
	32	0.22	0.83	0.21	0.71
	36	0.25	0.72	0.18	0.72
		With subpixel refinement			
BasketballDrive	24	0.25	11.88	2.97	5.50
	28	0.23	9.52	2.38	4.70
	32	0.26	8.46	2.11	4.43
	36	0.29	7.92	1.98	4.48
ParkScene	24	0.23	11.08	2.77	4.53
	28	0.21	9.23	2.31	4.03
	32	0.23	7.94	1.99	3.77
	36	0.26	7.09	1.77	3.75

(b) GOP-8 configuration.

to lie in between both values. It is further noted that the complexity increase of DMVD over the anchor simulations varies considerably among the sequences and QPs due to the DMVD utilization as indicated by f_{area}.

Comparing complexity estimates and time measurements between configurations with and without subpixel refinement reveals a factor of about 10–20 (IPPP), respectively 5–10 (GOP-8), in favor of the predictive search without subpixel refinement. Considering the relatively minor coding performance difference between predictive search with and without subpixel refinement (see Table IV) this implies a clear preference for the variant without refinement. However, even with this configuration, from Table VI a decoder complexity increase between 18%–95% is found for GOP-8 and between 34% and 329% for IPPP. While schemes with further reduced complexity have been proposed to JCT-VC as outlined in Section III-F, DMVD generally has a high complexity due to its calculations depending on decoded samples. During the development of HEVC, it was therefore concluded that such a complexity increase is not sufficiently justified by the compression efficiency gains, and DMVD was consequently not adopted to HEVC.

E. Performance for Extended KTA Tool Base

As shown in the previous chapters, DMVD can yield considerable bitrate reductions when applied to an H.264/AVC based software implementation. This section evaluates the performance of DMVD in combination with existing KTA tools identified as yielding significant coding efficiency improvements. Many of the KTA tools, such as larger prediction and transform sizes, improved inter polation, or more efficient motion vector coding have been adopted in a more or less modified form to HEVC. Some of these tools are concerned with motion coding or temporal prediction and may therefore compete with DMVD to some degree. The combination of DMVD with other KTA tools was submitted by the authors as response to the CfP on video compression technology and is described in detail in [8].

Presumably, the most important additional tool is the extension of the MB size in inter coded pictures from 16×16 to 32×32 luma pixels [56]. In addition to the larger block size, spatial transforms of sizes 16×16, 16×8, and 8×16 are available. The coding of MVs and reference indices at larger partition sizes may directly impact DMVD performance. DMVD has been implemented for 32×32 partitions in P and B pictures using four 16×16 targets. A target size of 32×32 and the extension to 32×16 and 16×32 partitions were not implemented due to a lack of time in the preparation of the CfP response.

Another tool with strong interrelation to DMVD is competition-based MV coding [43], [57] which is a predecessor of the advanced motion vector prediction in HEVC. It attempts to reduce the bitrate used for MV coding by allowing the encoder to select the MVP for a partition among a list of spatiotemporal neighbors by coding the index into the list.

Further KTA tools that were used are a high precision inter polation filter in combination with the enhanced adaptive inter polation filter [58], and the quad-tree-based adaptive loop filter [59]. Configuration details are found in [8].

In order to analyze the effect of DMVD in the proposal, simulations with the same software, configurations, and sequences as in [8] but with DMVD switched off have been conducted. Based on the configuration with additional KTA tools, DMVD alone yields 2.69% average bitrate reduction for the random access scenario and 3.46% in low-delay coding. More detailed results can be found in [9].

As expected, the coding efficiency gains of DMVD are reduced when combined with other coding tools that are also targeting motion coding, such as extended MB sizes and competition-based MV coding. Therefore, the possible coding efficiency gains of DMVD in HEVC are diminished in comparison to H.264/AVC.

F. Performance of DMVD in TMuC and HM

As mentioned in Section I, DMVD and variants thereof have been proposed and analyzed in the CfP and in tool and core experiments in JCT-VC. Several companies implemented DMVD variants in the test model under consideration

(TMuC) as well as the HEVC test model (HM), see [60] for an overview. These approaches targeted at simplified and complexity reduced decoder side motion vector derivation. Here, an (incomplete) set of documents submitted early after the CfP are cited. These proposals were further developed in the course of the tool and core experiments.

A simplified fast version of DMVD and a merge-like DMVD-based direct mode motion vector derivation was proposed in [61]. A parallel-friendly variant was proposed in [62]. DMVD for skip and direct blocks in B pictures was presented in [63]. In [64], bidirectional motion prediction from a unidirectional motion vector with decoder side refinement derivation was proposed.

In terms of compression performance, the proposed schemes revealed consistent results with the numbers reported in III-E above. The reported bitrate reductions are a bit lower than with the version presented in this paper. This may be due to the improved HEVC prediction and coding, the simplifications introduced to DMVD, as well as implementation issues in the HEVC reference software.

IV. Conclusion

In this paper, a complete system design for DMVD in the context of hybrid video coding was presented. Central design aspects, such as target and template sizes, motion search considerations, reference frame utilization, or entropy coding, were analyzed and interpreted.

An important aspect for the improved coding efficiency is the proposed MHP scheme, which obtains multiple prediction hypotheses without an increase of the bit cost as usually required for coding of additional motion parameters in traditional multihypothesis approaches.

A fast search algorithm for DMVD was proposed, which addressed concerns regarding the computational complexity of the decoder-side TM. By evaluating only a small set of motion candidates, the number of TM operations was significantly reduced in comparison to a full search approach. It was found that the fast search actually yielded a slightly higher coding efficiency than the full search on average. The presented fast search algorithm revealed commonalities of DMVD with motion representation mechanisms in HEVC. With DMVD, the applicable motion vector is derived at the decoder side, while for motion vector merging and motion vector prediction in HEVC, the applicable motion vector was explicitly signaled in the coded bitstream.

The compression performance of the proposed DMVD system was analyzed. On average, bitrate reductions of 6.7% for a low-delay scenario and 7.6% for a random access scenario are observed relative to the comparison points. For individual sequences and especially at higher spatial resolutions, gains of 10% or more are obtained. The complexity analysis shows that compression efficiency improvements of DMVD can be enabled at a moderate increase of computational resources.

By combining DMVD with other recently proposed advanced coding techniques, a DMVD coding technology proposal was prepared and submitted in response to the CfP on video compression technology for the new HEVC standardization effort, led by the JCT-VC. While it was observed that the coding efficiency contribution by DMVD was reduced due to the combination with other tools and the time constraints for optimization of the overall system, the subjective visual test performed as part of the CfP submission evaluation has attested to the capability of DMVD.

DMVD, like other recent tools proposed for video coding, relies on increased computing capabilities on the decoder side. The decoder re-estimates information that was previously derived at the encoder. An appropriate signaling method is needed for systematic control of the parameter re-estimation. In scenarios with sufficient reliability of the transmitted information, such methods for parameter estimation at the decoder side may be a way of further improving the compression efficiency in future video coding schemes.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments that helped improve this manuscript.

References

[1] ITU-T, *Advanced Video Coding for Generic Audiovisual Services*, ITU-T, Rec. H.264, version 1, 2003, version 5, 2010.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[3] M. Flierl, T. Wiegand, and B. Girod, "Rate-constrained multihypothesis prediction for motion-compensated video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 11, pp. 957–969, Nov. 2002.

[4] B. Girod, "Efficiency analysis of multihypothesis motion-compensated prediction for video coding," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 173–183, Feb. 2000.

[5] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[6] C. Stiller and J. Konrad, "Estimating motion in image sequences," *IEEE Signal Process. Mag.*, vol. 16, no. 4, pp. 70–91, Jul. 1999.

[7] *Joint Call for Proposals on Video Compression Technology*, document VCEG-AM91/MPEG-N11113, ITU-T SG16/Q6 VCEG and ISO/IEC JTC1/SC29/WG11 MPEG, Kyoto, Japan, Jan. 2010.

[8] S. Kamp and M. Wien, *Description of Video Coding Technology Proposal by RWTH Aachen University*, document JCTVC-A112, JCT-VC, Dresden, Germany, Apr. 2010.

[9] S. Kamp, *Decoder-Side Motion Vector Derivation for Hybrid Video Coding*, (Aachen Series on Multimedia and Communications Engineering Series). Aachen, Germany: Shaker-Verlag, Dec. 2011, no. 9.

[10] J. Ballé and M. Wien, "Extended texture prediction for H.264/AVC intra coding," in *Proc. IEEE Int. Conf. Image Process.*, vol. 6. Sep.–Oct. 2007, pp. 93–96.

[11] A. Stojanovic, M. Wien, and J.-R. Ohm, "Dynamic texture synthesis for H.264/AVC inter coding," in *Proc. IEEE Int. Conf. Image Process.*, Dec. 2008, pp. 1608–1611.

[12] 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. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 1641–1644.

[13] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C. S. Boon, "Inter frame coding with template matching spatio-temporal prediction," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1. Oct. 2004, pp. 465–468.

[14] Y. Suzuki, C. S. Boon, and S. Kato, "Block-based reduced resolution inter frame coding with template matching prediction," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 1701–1704.

[15] Y. Suzuki, C. S. Boon, and T. K. Tan, "Inter frame coding with template matching averaging," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2007, pp. III-409–III-412.

[16] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[17] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 71–83, Jan. 2005.

[18] S. Kamp, M. Evertz, and M. Wien, *Decoder Side Motion Vector Derivation*, document VCEG-AG16, ITU-T SG16/Q6 VCEG, Shenzhen, China, Oct. 2007.

[19] S. Kamp, M. Evertz, and M. Wien, *Decoder Side Motion Vector Derivation,* document M14917, ISO/IEC JTC1/SC29/WG11 MPEG, Shenzhen, China, Oct. 2007.

[20] S. Kamp, M. Evertz, and M. Wien, *Decoder Side Motion Vector Derivation With Multiple Reference Pictures*, document VCEG-AH15, ITU-T SG16/Q6 VCEG, Antalya, Turkey, Jan. 2008.

[21] S. Kamp, B. Bross, and M. Wien, *Fast Decoder Side Motion Vector Derivation*, document VCEG-AJ18, ITU-T SG16/Q6 VCEG, San Diego, CA, Oct. 2008.

[22] S. Kamp, J. Ballé, and M. Wien, *Response to Call for Evidence in HVC: Hybrid Video Coding With ETP and DMVD*, document M16661, ISO/IEC JTC1/SC29/WG11 MPEG, London, U.K., Jul. 2009.

[23] Y.-J. Chiu, L. Xu, W. Zhang, and H. Jiang, "*Description of Video Coding Technology Proposal: Self Derivation of Motion Estimation and Adaptive (Wiener) Loop Filtering*, document JCTVC-A106, Joint Collaborative Team on Video Coding (JCT-VC), Dresden, Germany, Apr. 2010.

[24] S. Sakazume, M. Ueda, S. Fukushima, H. Namamura, K. Arakage, and T. Kumakura, *Description of Video Coding Technology Proposal by JVC*, document JCTVC-A108, Joint Collaborative Team on Video Coding (JCT-VC), Dresden, Germany, Apr. 2010.

[25] Y.-W. Huang, C.-M. Fu, Y.-P. Tsai, J.-L. Lin, Y. Chang, J.-H. Guo, C.-Y. Chen, S. Lei, X. Guo, Y. Gao, K. Zhang, and J. An, *A Technical Description of MediaTek's Proposal to the JCT-VC CfP*, document JCTVC-A109, Joint Collaborative Team on Video Coding (JCT-VC), Dresden, Germany, Apr. 2010.

[26] H. Yang, J. Fu, S. Lin, J. Song, D. Wang, M. Yang, J. Zhou, H. Yu, C. Lai, Y. Lin, L. Liu, J. Zheng, and X. Zheng, *Description of Video Coding Technology Proposal by Huawei Technologies and Hisilicon Technologies*, document JCTVC-A111, Joint Collaborative Team on Video Coding (JCT-VC), Dresden, Germany, Apr. 2010.

[27] Y.-W. Chen, T.-W. Wang, C.-H. Chan, C.-L. Lee, C.-H. Wu, Y.-C. Tseng, W.-H. Peng, C.-J. Tsai, and H.-M. Hang, *Description of Video Coding Technology Proposal by NCTU*, document JCTVC-A123, Joint Collaborative Team on Video Coding (JCT-VC), Dresden, Germany, Apr. 2010.

[28] B. Bross, W.-J. Han, J.-R. Ohm, G. Sullivan, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 6*, document JCTVC-H1003, Joint Collaborative Team on Video Coding (JCT-VC), San Jose, CA, Feb. 2012.

[29] A. K. Jain, *Fundamentals of Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 1989.

[30] T.-W. Wang, Y.-W. Chen, and W.-H. Peng, "Analysis of template matching prediction and its application to parametric overlapped block motion compensation," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1563–1566.

[31] S. Kamp and M. Wien, "Decoder-side motion vector derivation for hybrid video inter coding," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Jul. 2010, pp. 1277–1280.

[32] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, ITU-T SG16/Q6 VCEG, Austin, TX, Apr. 2001.

[33] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. 27th Annu. Conf. Comput. Graphics Interactive Techniques*, 2000, pp. 479–488.

[34] *Information Technology—Coded Representation of Picture and Audio Information—Progressive Bi-Level Image Compression*, ITU-T Rec. T.82, Mar. 1993.

[35] *Information Technology—Lossy/Lossless Coding of Bi-Level Images*, ITU-T Rec. T.88, Feb. 2000.

[36] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, Dec. 1981, pp. G5.3.1–G5.3.5.

[37] C. Auyeung, J. J. Kosmach, M. T. Orchard, and T. Kalafatis, "Overlapped block motion compensation," *Proc. SPIE*, vol. 1818, no. 1, pp. 561–572, Nov. 1992.

[38] G. J. Sullivan, "Multi-hypothesis motion compensation for low bit-rate video coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 5. Apr. 1993, pp. 437–440.

[39] S. Kamp, B. Bross, and M. Wien, "Fast decoder side motion vector derivation for inter frame video coding," in *Proc. Int. Picture Coding Symp.*, May 2009.

[40] 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 Trans. Circuits Syst. Video Technol.*, vol. 3, no. 5, pp. 368–379, Oct. 1993.

[41] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proc. SPIE Conf. Visual Commun. Image Process.*, vol. 4671. Jan. 2002, pp. 1069–1079.

[42] M. Winken, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, D. Marpe, S. Oudin, M. Preiß, H. Schwarz, M. Siekmann, K. Sühring, and T. Wiegand, *Description of Video Coding Technology Proposal by Fraunhofer HHI*, document JCTVC-A116, Joint Collaborative Team on Video Coding (JCT-VC), 1st meeting, Dresden, Germany, Apr. 2010.

[43] G. Laroche, J. Jung, and B. Pesquet-Popescu, "RD optimized coding for motion vector predictor selection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1681–1691, Dec. 2008.

[44] *Key Technical Area (KTA) Software Version 2.6r1 of ITU-T/VCEG* [Online]. Available: http://iphome.hhi.de/suehring/tml/

[45] Joint Model (JM). *H.264/14496-10 AVC Reference Software of ITU-T/ISO* [Online]. Available: http://iphome.hhi.de/suehring/tml/

[46] T. K. Tan, G. J. Sullivan, and T. Wedi, *Recommended Simulation Common Conditions for Coding Efficiency Experiments, revision 4*, document VCEG-AJ10r1, ITU-T SG16/Q6 VCEG, 36th Meeting, San Diego, CA, Jul. 2009.

[47] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Jul. 2006, pp. 1929–1932.

[48] S. Kamp and M. Wien, *Error Accumulation in Motion Compensation in P and B Slices*, document JVT-AA039, Joint Video Team, Geneva, Apr. 2008.

[49] M. Karczewicz, Y. Ye, and P. Chen, *Switched Interpolation Filter With Offset*, document COM-16 C.463, ITU-T SG16/Q6 VCEG, Geneva, Apr. 2008.

[50] S. Kamp and M. Wien, *Multi-Hypothesis Prediction With Decoder Side Motion Vector Derivation*, document JVT-AA040, Joint Video Team, Geneva, Apr. 2008.

[51] A. Leontarsis and A. M. Tourapis, "Weighted prediction methods for improved motion compensation," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 1029–1032.

[52] M. Karczewicz, Y. Ye, and I. Chong, *Rate Distortion Optimized Quantization,* document VCEG-AH21, ITU-T SG16/Q6 VCEG, Antalya, Turkey, Jan. 2008.

[53] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[54] X. Zhou, E. Q. Li, and Y.-K. Chen, "Implementation of H.264 decoder on general-purpose processors with media instructions," in *Proc. SPIE Conf. Image Video Commun. Process.*, vol. 5022. Jan. 2003, pp. 224–235.

[55] Y.-K. Chen, E. Q. Li, X. Zhou, and S. Ge, "Implementation of H.264 encoder and decoder on personal computers," *J. Visual Commun. Image Representation*, vol. 17, no. 2, pp. 509–532, Apr. 2006.

[56] P. Chen, Y. Ye, and M. Karczewicz, *Video Coding Using Extended Block Sizes*, document VCEG-AJ23, ITU-T SG16/Q6 VCEG, San Diego, CA, Oct. 2008.

[57] J. Jung and G. Laroche, *Competition-Based Scheme for Motion Vector Selection and Coding*, document VCEG-AC06, ITU-T SG16/Q6 VCEG, Klagenfurt, Austria, Jul. 2006.

[58] Y. Ye and M. Karczewicz, *Enhanced Adaptive Interpolation Filter*, document COM-16 C.464, ITU-T SG16/Q6 VCEG, Geneva, Apr. 2008.

[59] T. Chujoh, N. Wada, and G. Yasuda, *Quadtree-Based Adaptive Loop Filter*, document COM-16 C.181, ITU-T SG16/Q6 VCEG, Geneva, Jan. 2009.

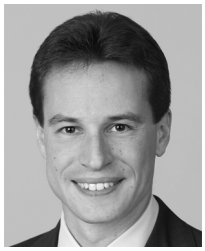
[60] Y.-J. Chiu, H. Yu, Y.-W. Huang, S.-i. Sekiguchi, and W.-H. Peng, *CE1: Summary Report of Core Experiment on Decoder-Side Motion Vector Derivation (DMVD)*, document JCTVC-E021, Joint Collaborative Team on Video Coding (JCT-VC), 5th Meeting, Geneva, Mar. 2011.

[61] S. Lin, M. Yang, J. Zhou, J. Song, D. Wang, H. Yang, J. Fu, H. Yu, Y. Wang, L. Zhang, S. Ma, and W. Gao, *TE1: Huawei Report on DMVD Improvements*, document JCTVC-B037, Joint Collaborative Team on Video Coding (JCT-VC), 2nd Meeting, Geneva, Jul. 2010.

[62] Y.-J. Chiu, L. Xu, W. Zhang, and H. Jiang, *TE1: Fast Techniques to Improve Self Derivation of Motion Estimation*, document JCTVC-B047, Joint Collaborative Team on Video Coding (JCT-VC), 2nd Meeting, Geneva, Jul. 2010.

[63] Y. Itani, S.-i. Sekiguchi, and T. Murakami, *TE1 Report: Implicit Direct Vector Derivation*, document JCTVC-C124, Joint Collaborative Team on Video Coding (JCT-VC), 3rd Meeting, Guangzhou, China, Oct. 2010.

[64] M. Ueda, *TE1.a: Implementation Report of Refinement Motion Compensation Using DMVD on TMuC*, document JCTVC-C138, Joint Collaborative Team on Video Coding (JCT-VC), 3rd Meeting, Guangzhou, China, Oct. 2010.



Steffen Kamp (S'05–M'10) received the Diploma and Dr.-Ing. degrees from Rheinisch-Westfaelische Technische Hochschule Aachen University, Aachen, Germany, in 2004 and 2011, respectively.

He is currently a Research Engineer with the Video Signal Processing Group, Panasonic Research and Development Center Germany GmbH, Langen, Germany. He has been an active contributor to ITU-T VCEG, ISO/IEC MPEG, and the Joint Video Team and Joint Collaborative Team on Video Coding (JCT-VC) of VCEG and MPEG. His current research interests include image and video processing and compression.

Mathias Wien (S'98–M'03) received the Diploma and Dr.-Ing. degrees from Rheinisch-Westfaelische Technische Hochschule Aachen (RWTH Aachen) University, Aachen, Germany, in 1997 and 2004, respectively.

From 1997 to 2004, he was a Researcher with the Institute of Communications Engineering, RWTH Aachen University. He is currently a Senior Research Scientist and the Head of Administration of the Institute of Communications Engineering, RWTH Aachen University. He has been an active contributor to H.264/AVC and has served as a Co-Editor of the SVC amendment to H.264/AVC. He has been an active contributor to the ITU-T VCEG, ISO/IEC MPEG, the Joint Video Team, and the Joint Collaborative Team on Video Coding (JCT-VC) of VCEG and ISO/IEC MPEG. He has co-chaired and coordinated several AdHoc groups and core experiments and has been a coordinator of the JCT-VC Tool Experiment on DMVD. His current research interests include image and video processing, space-frequency adaptive and scalable video compression, and robust video transmission.