Improved Video Compression Technology and the Emerging High Efficiency Video Coding Standard

Detlev Marpe*, Heiko Schwarz*, Thomas Wiegand*†, Sebastian Boße*,
Benjamin Bross*, Philipp Helle*, Tobias Hinz*, Heiner Kirchhoffer*, Haricharan Lakshman*,
Tung Nguyen*, Simon Oudin*, Mischa Siekmann*, Karsten Sühring*, Martin Winken*
*Image Processing Department, Fraunhofer HHI, Einsteinufer 37, 10587 Berlin, Germany
†Image Communication Chair, Berlin Institute of Technology, Einsteinufer 17, 10587 Berlin, Germany

***Abstract*—Today, H.264/AVC is the state-of-the-art video coding standard. Especially after the 2004 development of its High Profile (HP), it has become one of the primary formats in high definition video content delivery. Recently, a joint Call for Proposals (CfP) on video compression technology has been issued by ISO/IEC MPEG and ITU-T VCEG, targeting at the next generation of video compression standards with substantially higher compression capability than H.264/AVC. As a response to this CfP, Fraunhofer HHI proposed a newly developed video coding scheme which achieves bit rate savings of around 30% when compared to H.264/AVC HP. This paper describes the proposed video coding scheme and discusses its innovative features.**

I. Introdcution

In recent years, digital video has become the dominant form of media content in many consumer applications. Video compression is one of the enabling technologies for storage and delivery of digital video and in particular for high definition (HD) video, the current state-of-the-art video coding standard H.264/AVC [1] has become the most commonly used compression technology in many video applications. However, the core coding technology design of H.264/AVC was already finalized with the 2004 development of the High Profile (HP). Since then, exploratory work within both standardization organizations in the field of video coding, the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG), has shown potential for improving coding efficiency relative to H.264/AVC HP, in particular when considering the application to HD video and beyond. Motivated by these investigations, MPEG and VCEG agreed to form a Joint Collaborative Team on Video Coding (JCT-VC) and issued a Joint Call for Proposals (CfP) [2] for video compression technology.

As a response to the CfP, Fraunhofer HHI has developed a new video coding scheme [3] which achieves averaged bit rate savings of around 30% compared to H.264/AVC HP. According to the subjective performance evaluations, which were conducted in order to evaluate all 27 CfP responses, the proposed scheme was ranked among the best-performing proposals [4]. Design elements from those best-performing CfP responses, including the Fraunhofer HHI proposal, were incorporated into the first draft of the so-called Test Model under Consideration (TMuC) [5]. The corresponding new standardization project is also referred to as *High Efficiency Video Coding* (HEVC).

II. Overview of the proposed scheme

The proposed video coding scheme is based on the successful hybrid coding approach of using spatial and temporal, i.e., motion-compensated prediction, followed by transform coding of the residual. Its main innovative features can be summarized as follows:

- **Wide-range variable block-size prediction**: The size of prediction blocks can be adaptively chosen by using a quadtree-based partitioning. Maximum ($N_{\max}$) and minimum ($N_{\min}$) admissible block edge length can be specified as a side information.
- **Quadtree-based transform coding**: The block size used for discrete cosine transform (DCT)-based residual coding is adapted to the characteristics of the residual signal by using a nested, so-called residual quadtree (RQT) for partitioning of a given prediction block [6].
- **Merging of prediction blocks**: Neighboring blocks can be merged into one region, such that motion information has to be transmitted only once for a whole region [7].
- **Fractional-sample MOMS interpolation**: Interpolation of fractional-sample positions for motion-compensated prediction is based on a fixed-point implementation of the Maximal-Order-Minimum-Support (MOMS) algorithm using an IIR/FIR filter [8].
- **Adaptive in-loop filter**: In addition to the deblocking filter, a separable Wiener filter is applied within the coding loop. The filter is adaptively applied to selected regions indicated by the use of quadtree-based partitioning [9].
- **PIPE coding**: The novel probability interval partitioning entropy (PIPE) coding scheme provides the coding efficiency and probability modeling capability of context-based adaptive binary arithmetic coding (CABAC) [10] at the complexity level of variable-length coding [11].

III. Quadtree-based block partitioning

The basic processing unit of the described video coding scheme is a square block called *coding tree block* (CTB). Associated with each CTB is a nested quadtree structure that indicates the subdivision of the CTB for the purpose of prediction and residual coding. The outer quadtree structure divides the CTB into prediction blocks, while the inner (nested) quadtree structure specifies the further division of

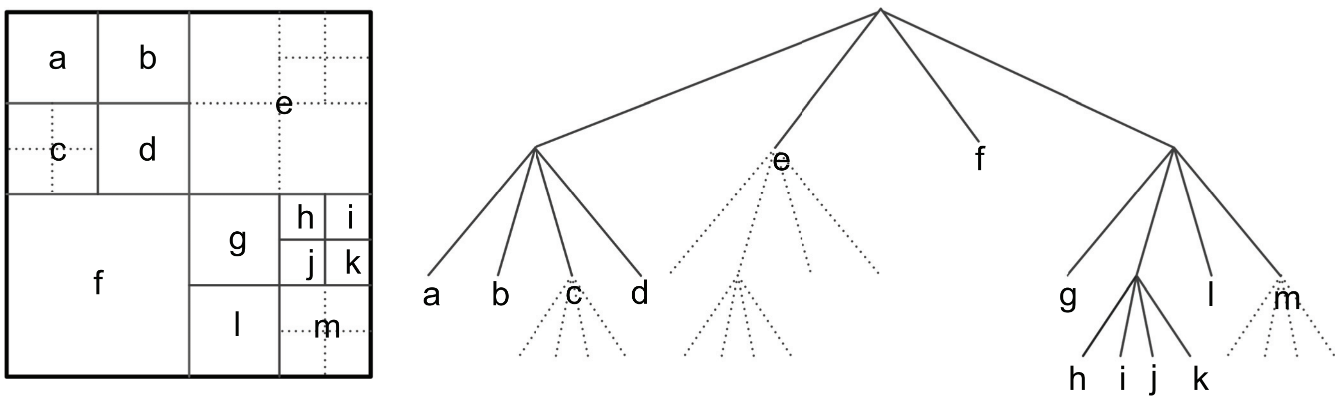

Fig. 1. Example of a nested quadtree structure (right part) for dividing a given coding tree block (left part; in black) into prediction blocks (solid gray lines) and transform blocks (dashed gray lines) of variable size. The order of parsing the prediction blocks follows their labeling in alphabetical order.

the prediction blocks into transform blocks. By dividing each picture into variable sized blocks, it is possible to adapt to the specific characteristics of the input video signal to be coded. Note that it is possible to specify different partitionings for the individual signal components (Y, Cb, Cr), or for a group of signal components (e.g., one partitioning for luma and a different one for chroma). But for the bitstreams provided as response to the CfP, one single partitioning for all signal components has been used.

Figure 1 (right) shows an example of a nested quadtree structure. The prediction quadtree is shown in solid lines, the nested residual quadtrees for transform coding are shown in dashed lines. On the left-hand side, the corresponding CTB and its subdivision into prediction and transform blocks is shown. Here, the prediction quadtree (solid lines) has four levels, with the root at level 0 corresponding to the full CTB size (maximum prediction block size), and with level 3 corresponding to a prediction block size having an edge length of one eighth of the CTB edge length. Generally, the edge length of prediction blocks at level i is always $2^{-i} \cdot N_{\max}$, where $N_{\max}$ is the edge length of the square block of luma samples associated with the CTB. Note that $N_{\max}$ is always a power of two. The CTB block size as well as the maximum and minimum prediction and transform block sizes are specified as side information in the bitstream. This allows to flexibly adapt these parameters depending on resolution, content, etc. of the input video sequence. In the encoding process, the CTBs are processed in raster scan order, and the prediction and transform blocks within each CTB are processed in depth-first order. This has the benefit, that the top and left neighboring blocks are always encoded before the current block, such that data already transmitted in these blocks can be used to facilitate encoding of the current block.

For each prediction block, either intra (spatial) or inter (temporal) prediction is used. In either case, the prediction residual, i.e., the difference between the original input signal and the prediction signal, is transform coded using variable-block size DCT. Note that, according to the residual quadtree (RQT), the prediction blocks can be further subdivided into smaller transform blocks, such that the block sizes for prediction and for DCT transform coding do not have to be the same. This is shown in Figure 1 for the prediction blocks c, e, and m. Transform block sizes in the range of 4×4 to 64×64 for the luma component and correspondingly scaled block sizes for both chroma components are supported. Note, however, that within these limits the maximum admissible RQT depth is variable and can be either signaled on a sequence parameter level, or can be constrained further by the use of appropriate profile or level limits. The transform kernel for each supported transform block size is given by a separable integer approximation of the 2D DCT-II (type-II Discrete Cosine Transform) of the corresponding block size.

The main idea for supporting variable block-size transforms is to adapt the transform to the varying space-frequency characteristics of the residual signal. DCT basis functions of larger spatial support (i.e., larger block size) provide a better frequency resolution than those having small spatial support, whereas the latter have a better spatial resolution than the former.

IV. Motion representation

Motion-compensated predicted (MCP) blocks are associated with one or two sets of motion parameters, where each set of motion parameters consists of a reference picture index and a motion vector. The use of one or two sets corresponds to P and B macroblock modes, respectively, as known from H.264/AVC. Motion vectors are represented in fractional sample units, where the accuracy of the motion vectors is signaled for each picture. In the coding experiments, the motion vector accuracy was set to units of quarter luma samples as in H.264/AVC.

For generating the prediction signal at subsample positions, we use an interpolation method based on families of so-called *maximal-order-minimal-support* (MOMS) basis functions. This requires, however, an additional prefiltering step on the reference picture before the actual interpolation. For our application case of fractional-sample interpolation, we have considered two members of the family of so-called O-MOMS

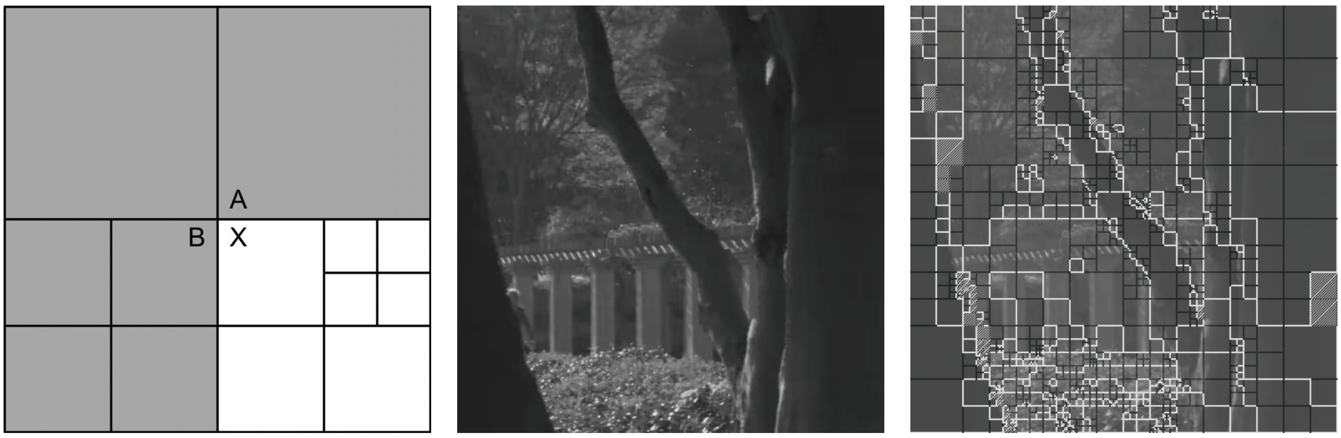

Fig. 2. Left: Schematic representation of a current block X, its set of merging candidates {A, B}, and its causal neighborhood (gray shaded). Middle: Cropped part (512×512) of a frame of the 1080p test sequence "Park Scene". Right: Illustration of MCP blocks (black lines), merged regions of MCP blocks (white lines), and intra-coded blocks (striped pattern) for the same cropped "Park Scene" content. With a chosen CTB size of 64×64, an area covering 64 CTBs is shown. Note that "Park Scene" contains a scene captured using a laterally from left to right moving camera and thus, the trees in the foreground, as shown in the middle image, appear to be moving to the left while the background park scene between the trees appears to be (slightly) moving to the right.

(optimal MOMS) with interpolation kernels of degree $L = 3$ (cubic) and $L = 5$ (quintic).

The aforementioned prefiltering step can be efficiently realized by separably applying a discrete 1D infinite impulse response (IIR) filter along rows and columns of the reconstructed picture. In the case of cubic or quintic O-MOMS, this IIR filter can be factorized into one or two sets of first-order causal and anti-causal recursive filters, respectively. Subsequent to this prefiltering stage, the actual interpolation is performed as a separable application of a 1D FIR filter with 4 taps in the cubic and 6 taps in the quintic case. Thus, in terms of required multiplication operations per fractional sample, an implementation of the cubic O-MOMS scheme including both prefiltering and interpolation is equivalent to a conventional 8-tap FIR interpolation scheme, at least when neglecting any symmetry of the filters involved. All filtering operations can be implemented using 16-bit fixed-point arithmetic without significant loss in coding efficiency. When compared to a 16-bit high-precision implementation of the H.264/AVC 6-tap interpolation filter as, e.g., being considered in [12], average bit rate savings of around 4% with maximum gains of up to 15% for individual test sequences have been achieved.

More details on the subsample interpolation scheme can be found in [8].

V. Merging of prediction blocks

The quadtree-based partitioning scheme for dividing the CTBs into prediction blocks, as described in the section III, is extended by a so-called *block merging process* for inter predicted blocks. This allows to merge neighboring prediction blocks into one contiguous region, such that the prediction parameters (i.e. motion vectors and reference indices) have to be transmitted only once for the whole region. This merging process can be viewed as a generalization of the "direct" prediction mode in H.264/AVC, since no prediction parameters are transmitted, but instead they are inferred from neighboring blocks. Figure 2 (left) illustrates the block merging process. Here, the current block is labeled as "X", and the causal neighborhood of the current block, i.e. those blocks that are encoded and transmitted before the current block, is marked gray shaded. Possible candidates for merging are the blocks labeled as "A" and "B" (i.e., the top and left neighboring blocks). Since merging is only possible for inter coded blocks, at least one of the blocks "A" or "B" has to be encoded using inter prediction, such that merging is available for the current block. In this case, it is signaled by the so-called *merge flag* whether merging is to be used. If this flag indicates merging and both "A" and "B" are inter encoded, a second flag indicates with which of the two possible blocks the current block is to be merged. Figure 2 (right) shows the resulting block structure for a sample picture after applying the quadtree-based partitioning and the block merging process. For more details on the block merging concept, the reader is referred to [7].

VI. Adaptive In-Loop Filter

Our proposed video coding scheme utilizes two types of cascaded in-loop filters: a *deblocking filter* and a subsequently applied *quadtree-based, separable 2D Wiener filter*. While the former is intended to deal with blocking artifacts in the reconstructed pictures, the latter mainly aims at reducing additional quantization noise in the output of the deblocking filter. Both types of filter are highly adaptive, and they are both applied within the coding loop with the output of the final filtering stage being stored in the reference picture buffer.

The deblocking filter is a generalization of the H.264/AVC deblocking filter for larger block sizes. Derivation of filter strengths as well as signaling of filter parameters is done as in H.264/AVC.

Following the deblocking filter, a quadtree-based separable 2D Wiener filter is applied. First, the vertical filter coefficients are calculated and then, after applying the vertical Wiener filter, the horizontal filter coefficients are derived, based on

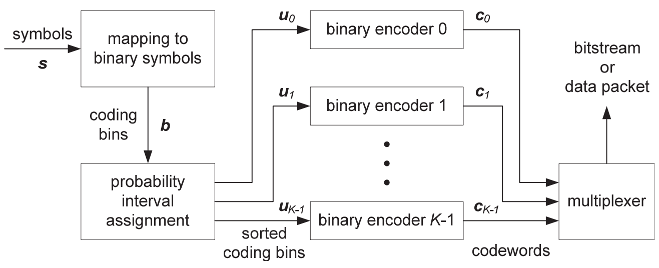

Fig. 3. Overview of the PIPE coding structure.

the output of the vertically filtered signal. The lengths of the vertical and horizontal filter are chosen from the set {3,5,7,9,11} by minimizing the Lagrangian rate-distortion (R-D) cost functional $D + \lambda R$ and taking into account the rate for transmission of the filter coefficients. Given an initial set of estimated filter coefficients, a quadtree-based block partitioning is derived by using a simple tree-pruning strategy based on the Lagrangian R-D cost functional. Then, given the R-D optimal partitioning, the filter coefficients are re-estimated by adapting them to the blocks which have been marked for filtering. The steps of filter redesign and re-partitioning can be iterated with the jointly R-D optimized result being finally applied and signaled to the decoder. Note that within our proposed filtering approach, the option of estimating and signaling two independent quadtree-based partitionings for vertical and horizontal filtering is supported [9].

VII. Entropy coding

For entropy coding, a variation of CABAC [10] is employed. Binarization and context modeling are basically the same as in CABAC of H.264/AVC, except from a few modifications in the context modeling part for large-size transform blocks [13]. However, the actual coding of binary decisions, so-called *bins*, is based on the novel concept of *probability interval partitioning entropy* (PIPE) coding that has been introduced in order to support parallelized implementations of entropy encoding and decoding as well as for decreasing the computational cost of entropy decoding [11], [14].

In Figure 3, the basic PIPE coding concept is illustrated. When a syntax element or symbol does not already represent a binary syntax element, it is first binarized, i.e., it is mapped onto a sequence of bins. For each bin a context is selected. A context represents a (binary) probability model for a class of bins; it is characterized by the probability and the value of the less probable bin (LPB). As in H.264/AVC, the LPB probability is represented by one out of 64 states. At the beginning of the encoding/decoding of a slice, the probability models are initialized using fixed values (as in CABAC of H.264/AVC). Then, after encoding/decoding a bin with a particular model, the probability and LPB value of the model is updated. The probability model update, i.e., the probability estimation process is the same as in CABAC of H.264/AVC. The association of a bin with a context model is also similar as in CABAC of H.264/AVC. It depends on the syntax element, the bin number, and, for some bins, the values of neighboring syntax elements.

With the binarization and association of bins with context models being basically the same as in CABAC of H.264/AVC, the main difference is given by the step of transforming bin into bits and vice versa. Instead of directly applying a binary arithmetic coder to the bins as in CABAC of H.264/AVC, the estimated LPB probabilities are quantized, i.e., they are mapped onto a small number K of LPB probability intervals. For each of these K probability intervals a separate *bin encoder/decoder* (bin codec) is operated. In our implementation, we use $K = 12$ probability intervals and thus 12 separate bin codecs. Each bin codec operates at a fixed LPB probability, which can be considered as the representative probability for an LPB interval. The selection of a bin codec is implemented via a look-up table that associates each of the 64 state indices for the LPB probability with a unique bin codec. Hence, the 64 states that are used for estimating the LPB probability of a context model are mapped onto 12 probability intervals, for each of which a separate bin codec is operated. For bin encoders and decoders, two alternatives have been implemented as follows.

PIPE Coding Using Arithmetic Codes: In a first PIPE coding version, the K bin encoders and decoders represent binary arithmetic encoding and decoding engines, respectively, which are similar to the M coder used in CABAC [10]. The corresponding K arithmetic codewords are written to different partitions of the bitstream with the corresponding partitioning information being transmitted in the slice header. An obvious advantage of this approach is that this way, binary arithmetic decoding can be parallelized. For instance, when operating all of the K arithmetic decoding engines in parallel, the corresponding sequences of bins can be written into K separate bin buffers. The remaining entropy decoding process can then simply read the bins from the corresponding bin buffers without the need to wait until a bin is arithmetically decoded before proceeding with the next bin.

PIPE Coding Using V2V Codes: A second version of PIPE coding uses prefix codes. For that, a variable number of bins is mapped onto variable-length codewords (also denoted as variable-to-variable (V2V) codes) and vice versa. In general, it is possible to get closer to the entropy limit when the size of the V2V code table is increased. In order to minimize the redundancy of the overall design, the probability interval partitioning and the V2V codes can be jointly optimized. The partial bitstreams associated to each bin codec can be written to different partitions of a bitstream or they can be interleaved into a single bitstream. For more details, please refer to [11].

Both PIPE coding versions have similar coding efficiency. For the generation of our CfP submitted bitstreams, the first version, i.e., the arithmetic coding based version of PIPE was used. Note, however, that lossless transcoding between the bitstreams of both PIPE versions was possible without exceeding the target bit rates given in the CfP [3].

TABLE I
BIT RATE SAVINGS RELATIVE TO THE H.264/AVC HP ANCHORS.

Class	Sequence	BD Rate CS 1 [%]	BD Rate CS 2 [%]
A (2560x1600)	Traffic	-27.92	n/a
	People on street	-17.92	n/a
	Average:	**-22.92**	**n/a**
B1 (1920x1080)	Kimono	-38.30	-38.11
	ParkScene	-24.28	-20.85
	Average:	**-31.29**	**-29.84**
B2 (1920x1080)	Cactus	-29.22	-23.43
	BasketballDrive	-35.97	-34.42
	BQTerrace	-41.92	-31.45
	Average:	**-35.70**	**-29.77**
C (832x480)	BasketballDrill	-31.88	-14.74
	BQMall	-29.71	-31.45
	PartyScene	-28.09	-16.55
	RaceHorses	-29.67	-22.39
	Average:	**-29.84**	**-21.28**
D (416x240)	BasketballPass	-21.97	-14.38
	BQSquare	-43.96	-13.68
	BlowingBubbles	-23.60	-7.44
	RaceHorses	-19.64	-14.23
	Average:	**-27.29**	**-12.43**
E (1280x720)	Vidyo 1	n/a	-28.49
	Vidyo 3	n/a	-22.58
	Vidyo 4	n/a	-28.65
	Average:	**n/a**	**-26.57**
	Total Average:	**-29.60**	**-22.68**

VIII. Coding Conditions and Results

In the CfP [2], two sets of application-specific coding conditions were specified: so-called constraint set 1 (CS 1) as a random access configuration (e.g., for broadcast applications) and constraint set 2 (CS 2) as a low-delay configuration (e.g., reflecting real-time applications). For CS 1, a hierarchical B picture coding structure with 4 temporal layers was used together with an appropriate chosen intra picture period. For CS 2, we used a hierarchical P picture coding structure with only one single intra picture at the beginning of the sequence and no picture reordering between the decoding process and the output. For both constraint sets, we configured the encoder to use a maximum prediction block edge length of $N_{\mathrm{max}} = 64$ luma samples and a maximum prediction quadtree depth of 4, corresponding to $N_{\mathrm{min}} = 4$.

Table I shows the average bit rate savings of the bitstreams as submitted to the CfP with the average for each of the test sequences obtained as mean of the Bjøntegaard delta (BD) bit rate values [15]. Overall, significant objective gains in terms of average 29.6% BD rate savings relative to the H.264/AVC HP anchor have been achieved for the random access scenario, and 22.68% BD rate savings for the low delay scenario.

As a rough measure of computational complexity, we obtained a factor of around 4 in encoding time and roughly a factor of 3–4 in decoding time relative to the H.264/AVC JM software, when averaging over all submitted bitstreams.

IX. Conclusion

We have presented our proposal for the High Efficiency Video Coding (HEVC) standardization project. It follows the same basic design principles as already established by the state-of-the-art video coding standard H.264/AVC, with generalizations for more flexibility in terms of prediction and transform block partitioning. New features of our proposal include the merging of prediction blocks, the MOMS based interpolation scheme, the adaptive in-loop filtering, and the Probability Interval Partitioning Entropy (PIPE) coding. The objective and subjective test results show significantly improved performance compared to H.264/AVC HP, with about 30% bit rate saving on average.

References

[1] *Advanced Video Coding*, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 13, Mar. 2011.

[2] *Joint Call for Proposals on Video Compression Technology*, ITU-T Q6/16 document VCEG-AM91 and ISO/IEC JTC1/SC29/WG11 document N11113, Jan. 2010.

[3] M. Winken *et al.*, "Description of Video Coding Technology Proposal by Fraunhofer HHI," JCT-VC, document JCTVC-A116, Apr. 2010.

[4] V. Baroncini, J.-R. Ohm, and G. J. Sullivan, "Report of Subjective Test Results of Responses to the Joint Call for Proposals on Video Coding Technology for High Efficiency Video Coding (HEVC)," JCT-VC, document JCTVC-A204, Apr. 2010.

[5] *Test Model under Consideration*, JCT-VC document JCTVC-A205, Apr. 2010.

[6] M. Winken, P. Helle, D. Marpe, H. Schwarz, and T. Wiegand, "Transform coding in the HEVC test model," in *Proc. IEEE International Conference on Image Processing (ICIP)*, Sep. 2011, to be published.

[7] S. Oudin, P. Helle, J. Stegemann, C. Bartnik, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand, "Block merging for quadtree-based video coding," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2011.

[8] H. Lakshman, B. Bross, H. Schwarz, and T. Wiegand, "Fractional-sample motion compensation using generalized interpolation," in *Proc. Picture Coding Symposium (PCS)*, Dec. 2010.

[9] M. Siekmann, S. Boße, H. Schwarz, and T. Wiegand, "Separable Wiener filter based adaptive in-loop filter for video coding," in *Proc. Picture Coding Symposium (PCS)*, Dec. 2010.

[10] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620 – 636, 2003.

[11] ——, "Entropy coding in video compression using probability interval partitioning," in *Proc. Picture Coding Symposium (PCS)*, Dec. 2010.

[12] M. Karczewicz, Y. Ye, and P. Chen, "Switched Interpolation Filter with Offset," ITU-T Q6/16, document VCEG-AI35, 2008.

[13] T. Nguyen, H. Schwarz, H. Kirchhoffer, D. Marpe, and T. Wiegand, "Improved context modeling for coding quantized transform coefficients in video compression," in *Proc. Picture Coding Symposium (PCS)*, Dec. 2010.

[14] D. Marpe, H. Schwarz, and T. Wiegand, "Novel Entropy Coding Concept," JCT-VC, document JCTVC-A032, Apr. 2010.

[15] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," ITU-T Q6/16, document VCEG-M33, Apr. 2001.