Title: Flexible distribution of complexity by hybrid predictive-distributed video coding

Authors: Jürgen Slowack, Jozef Škorupa, Stefaan Mys, Peter Lambert, Christos Grecos, and Rik Van de Walle

In: Signal Processing: Image Communication, Volume 25 (2), pp. 94-110, 2010.

Flexible Distribution of Complexity by Hybrid Predictive-Distributed Video Coding

Jürgen Slowack[a], Jozef Škorupa[a], Stefaan Mys[a], Peter Lambert[a], Christos Grecos[b], Rik Van de Walle[a]

[a] *Ghent University – IBBT, Department of Electronics and Information Systems – Multimedia Lab, Gaston Crommenlaan 8 bus 201, B-9000 Ghent, Belgium*
[b] *School of Computing, University of the West of Scotland, Paisley, PA1 2BE, Scotland*

Abstract

There is currently limited flexibility for distributing complexity in a video coding system. While rate-distortion-complexity (RDC) optimization techniques have been proposed for conventional predictive video coding with encoder-side motion estimation, they fail to offer true flexible distribution of complexity between encoder and decoder since the encoder is assumed to have always more computational resources available than the decoder. On the other hand, distributed video coding solutions with decoder-side motion estimation have been proposed, but hardly any RDC optimized systems have been developed.

To offer more flexibility for video applications involving multi-tasking or battery-constrained devices, in this paper, we propose a codec combining predictive video coding concepts and techniques from distributed video coding and show the flexibility of this method in distributing complexity. We propose several modes to code frames, and provide complexity analysis illustrating encoder and decoder computational complexity for each mode. Rate distortion results for each mode indicate that the coding efficiency is similar. We describe a method to choose which mode to use for coding each inter frame, taking into account encoder and decoder complexity constraints, and illustrate how complexity is distributed more flexibly.

Key words: Distributed Video Coding, hybrid video coding, Wyner-Ziv coding.

1. Introduction

In video coding systems, motion estimation is an essential but computationally complex task to exploit correlation between frames and achieve compression. In conventional, predictive coding schemes such as H.264/AVC [1], each macroblock is coded many times using several coding modes (e.g., inter, intra and direct modes), and the mode with the lowest cost in the rate-distortion sense is chosen. Many of these modes require computationally intensive calculations through motion estimation and local reconstruction of blocks, and as a result, the encoder is significantly more complex than the decoder. This asymmetry is well suited for applications where video is coded once and decoded many times or for streaming scenarios where the encoding device has many computational resources available.

Recently, Distributed Video Coding (DVC) systems have been proposed, where motion estimation is performed by the decoder instead of the encoder. For each so-called Wyner-Ziv (WZ) frame, the decoder generates an estimation using already decoded frames. This estimation – referred to as side information – is merely an approximation of the original frame available at the encoder. Hence, the encoder sends error correcting information allowing the decoder to correct the side information. This paradigm enables interesting applications featuring low-complexity encoders and complex decoders, such as networked camcorders, wireless video cameras, and visual sensor networks [2].

One of the challenges of video streaming scenarios is coping with the bandwidth requirements of the networks and the heterogeneity of the devices. Devices with different characteristics are performing video compression and streaming, ranging from high-end servers to PDA's and mobile phones. In addition, since almost every system is a multi-tasking system, available computational complexity is often non static. Furthermore, some devices are battery-constrained, and if computational complexity is decreased, techniques such as dynamic voltage scaling (DVS) can be used to extend battery lifetime [3].

As such, besides rate and distortion, available computational complexity is considered an important parameter in a video coding system. This prompted methods for rate-distortion-complexity (RDC) optimization, which have been developed primarily for predictive video coding [3, 4, 5, 6]. In predictive video coding, motion estimation is a very computationally complex task, due to the high number of coding modes and the high computational

complexity of many of these modes. Complexity is typically reduced by applying a suboptimal low-complexity motion search algorithm, or by selecting only a subset of coding modes that need to be tested while maximizing the probability that the best coding mode is in this subset. However, only complexity reduction is considered, and there is no way for the decoder to take over some of the workload[1].

For DVC systems, only limited complexity analysis has been provided so far, for example, for the DISCOVER codec [7].

We can summarize by stating that current solutions for RDC optimization do not allow motion estimation to be shifted between encoder and decoder, but only allow complexity to be decreased. Dynamic aspects such as multi-tasking, variable power-supply, and session mobility need systems that adapt better to the changing conditions, by distributing complexity between encoder and decoder according to the amount of resources available at both devices. At one time instance, the encoder device could have more resources available than the decoder device but at a later point in time this could be the other way around.

In this paper we present a codec that combines techniques from predictive video coding and DVC, to realize flexible distribution of complexity by sharing the computationally complex task of motion estimation between encoder and decoder. We firstly provide a brief overview of the state-of-the-art in DVC (Sect. 2), and we describe interesting systems in the context of this paper (Sect. 3). Next, the general operation of the proposed codec is described (Sect. 4.1). This codec features several modes for coding inter frames, using either the predictive mode (Sect. 4.3), the DVC mode (Sect. 4.4) or one of the hybrid modes (Sect. 4.5). Two variants for implementing the hybrid modes are identified. A first approach is to apply a spatial partitioning technique and predict some of the macroblocks in a frame at the encoder while estimating the remaining macroblocks at the decoder (Sect. 4.5.1). In the second approach, the motion search algorithm is split in two parts, i.e., the encoder calculates coarse motion vectors which are further refined by the decoder (Sect. 4.5.2).

The main novelty of this paper is twofold. Firstly, we extend the codec and the different modes proposed in our earlier work [8, 9], and compare them

[1]Remark: due to the use of a decoder loop at the encoder, the encoder remains more complex than the decoder at any time.

(Sect. 5). Secondly, we use a theoretical model for describing the encoder and decoder complexity in each of the modes (Sect. 4.2), which is validated by practical measurements (Sect. 6). This analysis enables us to define an optimization technique for meeting complexity constraints at both encoder and decoder (Sect. 7). Final remarks (Sect. 8) and conclusions (Sect. 9) end the paper.

2. State-of-the-art in Distributed Video Coding

Based on the theoretical work of Slepian and Wolf [10], and Wyner and Ziv [11], practical DVC systems have been developed, firstly by Puri and Ramchandran, and later on by Aaron and Girod.

Puri and Ramchandran proposed a system called PRISM: Power-efficient, Robust, hIghcompression, Syndrome-based Multimedia coding [12]. In PRISM, each frame consists of macroblocks which are either not coded (skip), intra coded, or Wyner-Ziv (WZ) coded. Intra coding is performed using traditional techniques used in for example, MPEG-x and H.26x. For each macroblock X that is WZ coded, the decoder generates a prediction Y using already decoded macroblocks. This prediction or side information can be regarded as the original macroblock corrupted by noise: $Y = X + N$. This dependency can be modeled as a noisy virtual channel which has X as input, Y as output, and noise N. To correct errors induced by this virtual channel, channel codes are used. More specifically, in PRISM, syndrome codes are used.

The architecture developed at Stanford by Aaron, Girod, and others, uses a frame-based approach where the frame sequence is split up into key frames and WZ frames. Key frames are intra coded using conventional techniques such as H.264/AVC, while WZ frames are predicted at the decoder side using techniques such as frame interpolation [13], or motion compensated interpolation/extrapolation [14]. In contrast to PRISM, a turbo codec is used in combination with a feedback channel, i.e., the decoder-side prediction (or side information) is corrected using a turbo decoding procedure where parity bits are requested from the encoder until the result is assumed to be reliable.

Many researchers proposed new techniques for DVC, in most cases using the Stanford architecture. Some important contributions have been made in the context of the DISCOVER project, such as improvements to the generation of the side information by using bidirectional motion refinement and spatial smoothing, and adaptive GOP size control [15]. Other extensions

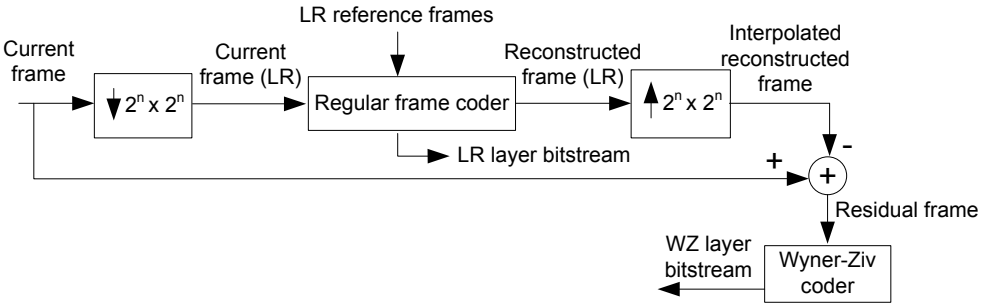

Figure 1: Spatially scalable encoding architecture proposed by Mukherjee [38].

or improvements have been proposed for side information generation, such as exploiting both temporal and spatial correlation [16, 17], and using multiple side information streams [18]. Techniques for online modeling of the correlation noise between the original and the side information have been developed [19, 20], taking into account the quantization noise in the reference frames [21]. Rate-distortion analysis has been provided for motion extrapolation [22], motion compensated interpolation [23] and hash-based side information generation [24]. The feedback channel has been studied [25] and practical request stopping criteria have been formulated [26][27] as well as how to eliminate the feedback channel [28]. Alternative channel codes have been studied such as LDPC codes [29, 30] and overlapped quasi-arithmetic codes [31].

DVC techniques have been used in other scenarios, such as traditional video coding with forward error correction using WZ coding [32, 33, 34] and multi-view coding [35, 36, 37].

3. Related work

An important system in the context of this paper has been proposed by Mukherjee [38]. A regular coder is used to code inter frames that are used by the encoder as reference frames for motion estimation. The remaining inter frames are referred to as non-reference Wyner-Ziv (NRWZ) frames, and they are coded following a hybrid approach. At the encoder (Fig. 1), these frames are subsampled with a factor $2^n \times 2^n$, where n can be chosen based on a complexity reduction target. The resulting low-resolution (LR) frames are coded using a regular coder (e.g. H.263+) which uses LR reference frames for motion estimation, and the resulting LR bitstream is sent to the decoder.

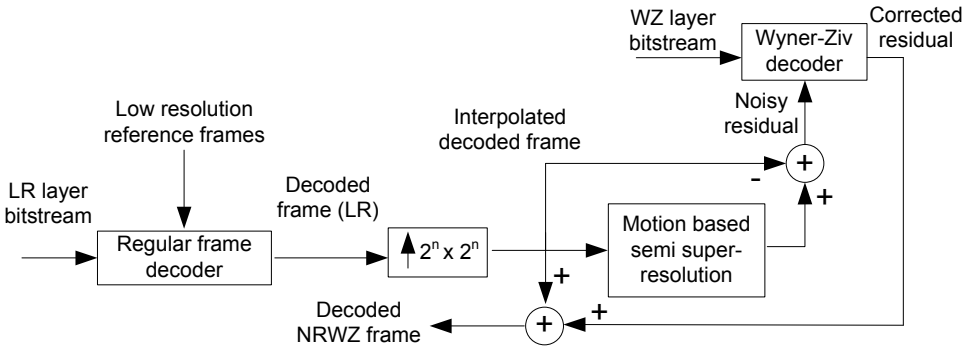

Figure 2: Spatially scalable decoding architecture proposed by Mukherjee [38].

Next, the reconstructed (decoded) LR frame is interpolated to full resolution, and the residual with the original frame is WZ coded.

At the decoder (Fig. 2), the LR frame is decoded by the regular frame decoder. Subsequently, the frame is interpolated using the same interpolation filter as the encoder, and the result is refined in a procedure called *motion based semi super-resolution*. Next, the noisy residual between the refined frame and the non-refined frame is corrected by the WZ decoder, using the WZ layer bitstream. Finally, the decoded NRWZ frame is obtained by adding the interpolated decoded frame to the corrected residual. It is unclear from this work how to choose the subsampling factor n as well as how to adapt the motion based semi super-resolution to changes in n. Intuitively, we would expect more decoder-side calculations as n becomes larger.

4. Description of the proposed video codec

An alternative system has been proposed by the authors of this paper. This system features several modes for coding frames, and each mode shares the complex task of motion estimation differently between encoder and decoder. We proposed several modes for coding frames: the predictive mode with motion estimation performed by the encoder, the DVC mode with motion estimation performed by the decoder, and the hybrid modes where motion estimation is shared. Two variants for the hybrid modes have been developed, using a spatial partitioning technique on the one hand [8], and a partitioning of the motion search algorithm on the other hand [9]. One contribution in this paper is that we extend our previous work to residual coding and subpixel refinement.

The widely-used Stanford architecture has been taken as a basis for developing our system (Fig. 3, Fig. 4). As such, the frame sequence is partitioned into intra frames I and inter frames W. Motion estimation is performed for the inter frames, at the encoder and/or decoder, depending on the mode. Only the part involved in motion estimation operates differently for each mode (denoted "mode-dependent" in Fig. 3 and Fig. 4), therefore, we split our discussion into two parts.

Firstly, we discuss the modules that are functionally independent from the coding mode used such as the WZ codec, the intra codec and the buffering system; and we describe the interaction between these modules (Sect. 4.1). The reason why the WZ codec is used even for the parts for which motion estimation is performed at the encoder side, is because of the inherent error robustness compared to conventional H.264-like solutions [39], which is a major advantage in video streaming scenarios, for example.

Secondly, the motion estimation part is described in detail for each mode (Sect. 4.3 to Sect. 4.5), and complexity is analyzed using a theoretical model for complexity (Sect. 4.2).

4.1. General codec operation

At the encoder (Fig. 3), intra frames I are coded using H.264/AVC intra coding. Decoded intra frames I' are available anyway after intra coding due to mode decision and rate-distortion optimization, hence, I' frames are stored in the *decoded I frame buffer*.

For each inter frame W, a prediction Z is generated (that will also be generated at the decoder side). The residual R between W and Z is calculated and Z is stored in the *prediction frame buffer*.

R is WZ coded as follows. Firstly, R is transformed using a 4-by-4 DCT and coefficients at the same position k are grouped into coefficient bands R_k. For example, all DC coefficients will form band R_0. Next, each band is quantized using a uniform deadzone quantizer with 2^{M_k} levels, and zero bin width 1.5 times the width of the other bins. For each band, bits at identical positions are grouped into bitplanes BP_i^k. For example, all most significant bits of all DC coefficients will form bitplane BP_0^0. Finally, each bitplane is turbo coded, and the resulting parity bits are stored in a buffer. These bits will be punctured and sent to the decoder upon request.

At the decoder (Fig. 4), intra frames are decoded into I' and stored in the *decoded I frame buffer*. For each inter frame W, side information Y is generated as well as the mutual prediction Z. The residual Y^R between Y

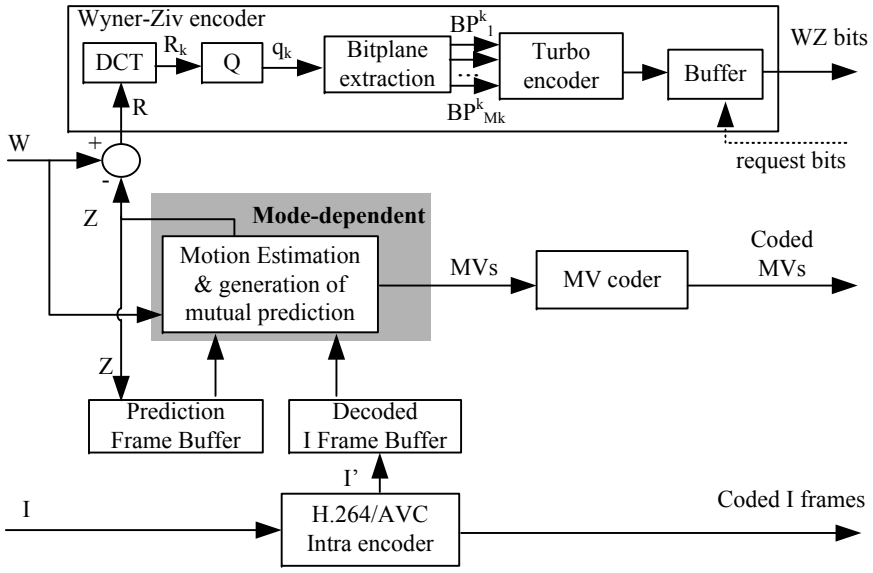

Figure 3: Encoder architecture consisting of a WZ encoder, an intra encoder, a mode-dependent part and a mechanism for buffering frames.

and Z is transformed and used by the turbo decoder. The turbo decoder requests as many bits as needed until Y^R is corrected. When all bitplanes are decoded by the turbo decoder, they are multiplexed and the unquantized coefficients are reconstructed using centroid reconstruction, as in [40]. The result is inverse transformed into R', and Z is added to obtain the decoded frame W'. For future reference, W' is stored in the *decoded W frame buffer*.

The turbo decoder needs information about the reliability of the side information. This reliability – more specifically, the correlation between the original coefficient and the side information coefficient – is modeled online using the method described by Brites and Pereira [19] (at coefficient-frame level). This method estimates the correlation using the difference between the reference blocks obtained after side information generation.

4.2. Modeling motion estimation complexity

The execution speed of a program depends on parameters such as the number of operations to be executed, the speed of these operations (additions, multiplications, conditional expressions, etc.), the number of memory requests and the delay associated with these requests (which depends on the memory architecture, cache behavior, etc.). Modeling complexity theoreti-

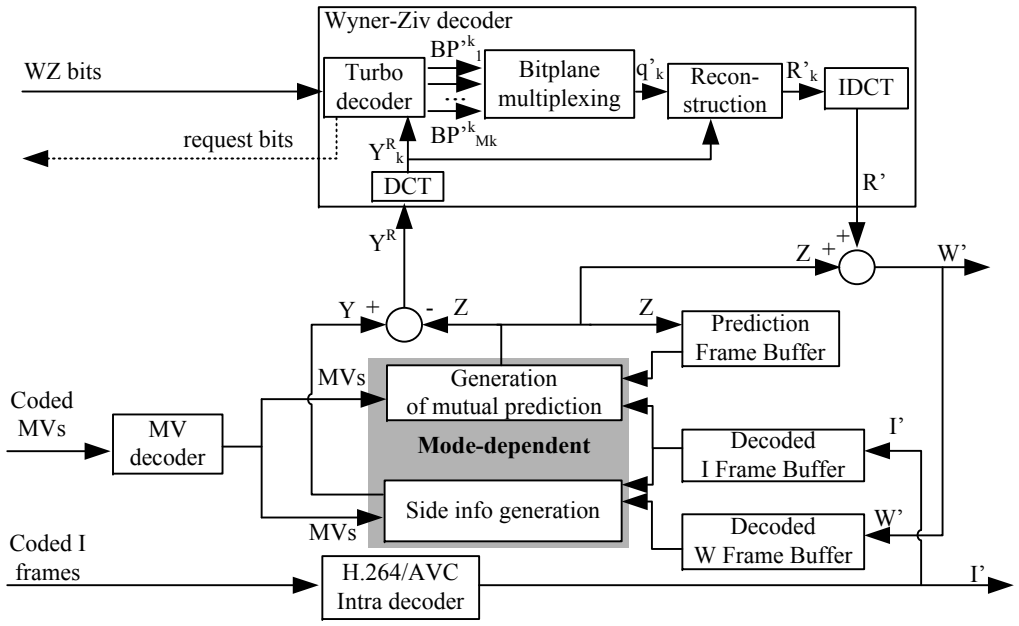

Figure 4: Decoder architecture consisting of a WZ decoder, an intra decoder, a mode-dependent part and a mechanism for buffering frames.

cally by taking all these parameters into account is difficult and hardware dependent. However, the performance of multimedia applications is mainly determined by the transfer of data from and to memory, as illustrated by e.g. Brockmeyer et al., who estimated that a software implementation of an MPEG-4 video encoder (VM 7.0) typically requires about $5 \cdot 10^9$ memory transfers per second to encode the simple profile level L2 [41].

Hence, as a complexity measure we propose to model the data transfers performed during motion estimation in our system, by pixel read and pixel write operations. The latter are defined by generalizing each step in the motion estimation process as an operation that is performed on pixel[2] data (Fig. 5). To perform the operation (e.g. spatial interpolation, Lagrangian cost calculation) a number of pixels need to be read, typically a macroblock. These pixels are read from the original frame, from a past or future reference frame, from a temporally stored frame such as the current version of the

[2]In the context of pixel read and write operations, the term "pixel" will be used to indicate one particular luma or chroma value.

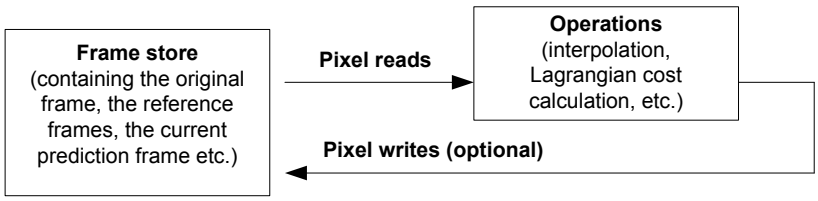

Figure 5: Modeling complexity by pixel read and write operations.

side information, etc. If the end result of the operation is pixel data, pixel write operations are performed. As a complexity measure, in the following sections we will count the total number of such pixel read/write operations performed at the encoder and the total number of pixel read/write operations (or briefly referred to as "operations") at the decoder, for each of the modes. The results are listed in Tab. 1 as reference. The accuracy of this model is evaluated in Sect. 6.

Frames are assumed to be YUV, and unless stated otherwise, motion estimation is performed only on the luma component while motion compensation is performed on both luma and chroma. 4:2:0 subsampling is used, so that frames have a (spatial) luma resolution of H(orizontal) by V(ertical) pixels whereas the chroma components are each $H/2$ by $V/2$.

4.3. Motion estimation in the predictive video coding mode

In the predictive video coding mode, motion estimation is performed solely at the encoder. Each inter frame W is partitioned into macroblocks of size M (e.g. $M = 64$ for 8-by-8 macroblocks), and motion vectors are calculated for each of the HV/M (luma) blocks using bidirectional motion estimation. More precisely, the *prediction frame buffer* and *decoded I frame buffer* are consulted and the closest past frame P and future frame F are retrieved. For each macroblock MB_i in W, the best match of full pixel precision in P is determined using a search window of size S, where the size of the search window indicates the number of macroblocks in the search space. Hence, for each of the HV/M blocks S times $2M$ pixels are compared, resulting in a total of $2SHV$ pixel read/write operations per inter frame, for this step (Tab. 1).

The best match is found by minimizing a lagrangian cost function:

$$Cost_i(\vec{v}) = D_i(\vec{v}) + \lambda R_i(\vec{v}) \tag{1}$$

where the distortion metric $D_i(\vec{v})$ is the Sum of Squared Errors (SSE) between MB_i and the macroblock in P defined by the motion vector $\vec{v}$. λ is a Lagrange multiplier that has been determined offline using several sequences, and which is set to 30, 65, 110, or 180 for quantization matrices Q_0 to Q_3 (which are defined further on in this paper). $R_i(\vec{v})$ represents the rate to code $\vec{v}$. Motion vectors are coded by first predicting them from their neighbors as in H.264/AVC and coding the residual between $\vec{v}$ and its prediction using signed exponential Golomb coding, resulting in $R_i(\vec{v})$ bits.

After motion estimation between W and P, the averages between the best block in P and each of the S candidate blocks in F are calculated. Next, each block in W is compared against each of the averages using the same cost function, and the block in F corresponding to the best match is selected. This requires an additional SHV pixels to be read from P, so that a total of $3SHV$ operations are required.

The result from the motion estimation process is that we have now two motion vectors for each macroblock in W, i.e., one referring to a block in P and one to a block in F. Since the motion vectors calculated at the encoder will also be available at the decoder, they are used to generate the mutual prediction Z through bidirectional motion compensated interpolation, performed on all color components, resulting into $9/2HV$ pixel read/write operations.

At the decoder, Z is constructed using the received motion vectors and the reference frames P and F (retrieved from the *prediction frame buffer* and/or *decoded I frame buffer*). Y is taken equal to Z so that the residual Y^R between Y and Z that is used by the turbo decoder contains only zeros.

4.4. Motion estimation in the DVC mode

In the DVC mode, no motion estimation is performed by the encoder. The mutual prediction Z equals the closest frame, past or future, that can be retrieved from the prediction frame buffer and decoded I frame buffer.

At the decoder, side information is generated for each frame W using the closest past frame P and closest future frame F, retrieved from the *decoded W frame buffer* and *decoded I frame buffer* only, since decoded frames have better quality than mutual prediction frames.

The side information Y is generated based on the work of Artigas et al. [15], implemented in the DISCOVER codec. Firstly, for better capturing the true motion field, the luma component of P and F is low-pass (LP)

Table 1: Number of pixel read/write operations executed at encoder and decoder, per W frame. Absolute numbers (expressed in $\cdot 10^6$ operations) for: $H = 352$ and $V = 288$, $S = 1089$, $M = 64$, $L = 9$, $S_R^{DVC} = 16$, $S_R^{SPAT} = 25$, $S_R^{SUB_1} = 25$, and $S_R^{SUB_2} = 9$.

	Encoder-side			**Decoder-side**		
Pred. mode	ME (W, P)	$2SHV$	220.8	Constr. Z	$9HV/2$	0.5
	ME (W, F)	$3SHV$	331.2			
	Constr. Z	$9HV/2$	0.5			
	Total		552.4	Total		0.5
DVC mode				LP filtering	$2HV(L+1)$	2.0
				ME (F, P)	$2SHV$	220.8
				Wiener interp.	$115HV$	11.7
				Refin. (16x16)	$512HVS_R^{DVC}/M$	7.3
				Refin. (8x8)	$128HVS_R^{DVC}/M$	1.8
				Spat. smooth.	$16HV$	1.6
				Bid. interp.	$9HV/2$	0.5
	Total		–	Total		252.8
Hybrid spatial mode	ME (S_1)	$5SHV/2$	276.0	LP filtering	$2HV(L+1)$	2.0
	Constr. Z	$9HV/2$	0.5	Wiener interp.	$115HV$	11.7
				Refin. (16x16)	$256HVS_R^{SPAT}/M$	10.1
				Refin. (8x8)	$64HVS_R^{SPAT}/M$	2.5
				Bid. interp.	$9HV/2$	0.5
				Constr. Z	$9HV/2$	0.5
	Total		276.5	Total		27.3
Hybrid subs. mode	Subs.	$15HV/4$	0.4	LP filtering	$2HV(L+1)$	2.0
	ME	$5SHV/16$	34.5	Wiener interp.	$115HV$	11.7
	Constr. Z	$9HV/2$	0.5	Refin. (16x16)	$512HVS_R^{SUB_1}/M$	20.3
				Refin. (8x8)	$128HVS_R^{SUB_2}/M$	1.8
				Bid. interp.	$9HV/2$	0.5
				Constr. Z	$9HV/2$	0.5
	Total		35.3	Total		36.7

filtered by replacing each pixel by the average of a group of $L = 3$-by-3 pixels having this pixel as a center. This requires a total of $2HV(L+1)$ operations[3].

Next, unidirectional block-based motion estimation is performed between the filtered versions of P and F. The candidate motion vectors are scaled with the distance $\Delta_{P,F}$ between P and F, with $\Delta_{P,F} = 1$ if the frames are adjacent to each other. This is performed for compensating for possible larger motion as the distance between the reference frames increases. Matching is performed using the following cost function (CF):

$$CF(v_x, v_y) = (1 + 0.05\sqrt{v_x^2 + v_y^2}) \cdot \text{MAD}(v_x, v_y), \tag{2}$$

where (v_x, v_y) indicates the motion vector from F to P, and MAD is the Mean Absolute Difference between the corresponding blocks in F and P, defined by (v_x, v_y).

After obtaining the motion vectors from F to P, for each macroblock in Y the motion vector intersecting the block closest to the block center is chosen and treated as a bidirectional motion vector (Fig. 6).

Next, the LP-filtered versions of P and F are upsampled to half pixel precision using a 6-tap Wiener interpolation filter (only the luma), as well as the full quality reference frames P and F (all color components, for motion compensation). For each luma component, HV values are copied to a frame at higher resolution, requiring $2HV$ operations. The remaining $3HV$ values are calculated using 6-tap Wiener interpolation, where each value is constructed by reading 6 luma values and writing one, resulting in a total of $23HV$ pixel read/write operations for the luma component. Likewise, we need $11.5HV$ operations for both chroma components together. Hence, for two luma only frames and two YUV frames, we need $115HV$ operations.

Using the half pixel LP-filtered reference frames, the motion vector of each block in Y is refined in two passes: first using reference blocks of size 16x16 and next using reference blocks of size 8x8. At all times, we assume that the motion vector is linear between P and F, and that it goes through the block center. The refinement window for a certain block is defined by the motion vectors of the neighboring blocks. As such, we define the refinement process as finding the vector that minimizes the MAD between past and

[3]From here on, we will only provide complexity analysis for steps that are not similar to techniques analyzed earlier in this work. We refer to Tab. 1 for a complete overview.

future blocks, with the additional constraint that the backward motion vector (v_x, v_y) should satisfy:

$$\min(v_x^B, v_x^D) \leq v_x \leq \max(v_x^B, v_x^D), \tag{3}$$

$$\min(v_y^A, v_y^C) \leq v_y \leq \max(v_y^A, v_y^C), \tag{4}$$

where A is the top neighbor, B the left neighbor, C the bottom neighbor, and D the right neighbor.

Due to the fact that the refinement window size S_R^{DVC} is calculated using the neighboring motion vectors, it is not constant. S_R^{DVC} will be small (on average) if the motion vector field is smooth. On the other hand, S_R^{DVC} will be large if there are a lot of discontinuities in the motion vector field. We obtained a value for S_R^{DVC} to use in our complexity analysis experimentally, using the setup described in the results section (Sect. 5). By averaging over all sequences and rate points, a value of $S_R^{DVC} = 16$ has been obtained. Hence, each of the HV/M motion vectors is refined by reading S_R^{DVC} candidate pairs of 16-by-16 blocks in the first pass and reading S_R^{DVC} candidate pairs of 8-by-8 blocks in the second pass. This results in a total of $S_R^{DVC} \cdot 512HV/M$ for the first pass and a total of $S_R^{DVC} \cdot 128HV/M$ operations for the second pass.

After motion refinement, the motion vectors are spatially smoothed by weighted vector median filtering of the motion vector for MB_i and the motion vectors of neighboring macroblocks applied to MB_i. For each of the HV/M macroblocks, applying the (approximately) eight neighboring motion vectors for calculating the weights used in median filtering results in eight times two blocks of size M to be read (one from the past reference frame and one from the future reference frame), which is a total of $16HV$ operations per frame for this step.

Finally, the calculated motion vectors are used for bidirectional motion compensation to obtain the side information frame Y.

4.5. Motion estimation in the hybrid video coding modes

In the hybrid modes, motion estimation is shared between encoder and decoder. Two variants for sharing motion estimation can be identified, based on spatial partitioning on the one hand and splitting of the motion estimation algorithm on the other hand.

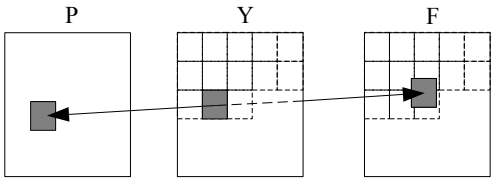

Figure 6: For each macroblock in Y, the closest intersecting motion vector is chosen and treated as a bidirectional motion vector.

4.5.1. Spatial partitioning

Motion estimation can be shared between encoder and decoder by partitioning the macroblocks in an inter frame W into two subsets S_1 and S_2, for which motion estimation will be performed at the encoder or decoder, respectively. At the encoder, we can directly apply the techniques developed in the predictive mode to the elements in S_1, resulting in an encoder complexity equal to the complexity in the predictive mode multiplied by the ratio of the number of elements in S_1 to the total number of blocks. In the DVC mode, however, the first step in motion estimation is unidirectional motion estimation between the reference frames P and F. This step is performed to enable generating an initial motion vector estimate for the blocks in W, after which this estimate is further refined on a subpixel level and finally spatially smoothed. In other words, in this first step, calculations are performed for all macroblocks in W at once, which does not directly allow us to leave out S_1, for which calculations have already been performed at the encoder side.

Hence, we eliminate the unidirectional motion search between the (LP-filtered) reference frames by choosing a checkerboard partitioning strategy for defining S_1 and S_2, so that initial motion vector estimates for each element in S_2 can be generated from its (maximum) four immediate neighbors in S_1. Consider for example a non-border block H (Fig. 7). We approximate the motion between the reference frames as being linear, as in the DVC mode, and we obtain an initial motion vector estimate for H by treating the backward and forward motion vector of each neighbor in S_1 (e.g. A in Fig. 7) separately. As such, the bidirectional vector $((v_x^F, v_y^F), (v_x^B, v_y^B))$ is split into two bidirectional vectors describing linear motion: $((v_x^F, v_y^F), \frac{\Delta_{P,Y}}{\Delta_{Y,F}} \cdot (-v_x^F, -v_y^F))$ and $((v_x^B, v_y^B), \frac{\Delta_{Y,F}}{\Delta_{P,Y}} \cdot (-v_x^B, -v_y^B))$. This is done for all neighbors A, B, C, and D, resulting into eight vectors. Each of these motion vectors is applied to H, and the one minimizing the Sum of Squared Errors (SSE) is

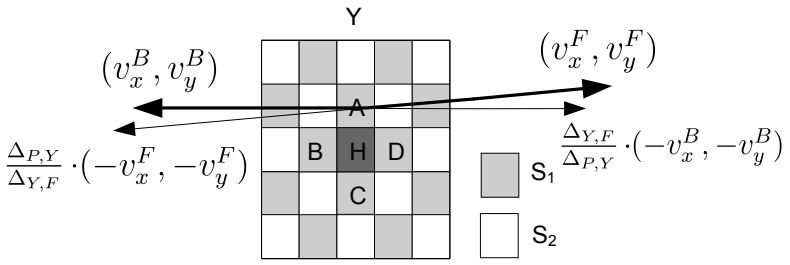

Figure 7: The hybrid spatial mode using a checkerboard pattern.

chosen.

This initial estimate is then used as a starting point for half-pixel motion refinement, as in the DVC mode (but refinement is obviously only applied to S_2). In this case a refinement window of fixed size 5-by-5 ($S_R^{SPAT} = 25$) showed better results.

Subsequent to half-pixel refinement, bidirectional motion compensation is performed to construct the side information frame Y. No spatial smoothing is performed, since information from neighboring blocks is already taken into account during initialization of the motion vector.

To construct the mutual prediction frame Z, each block in S_2 is assigned the motion vector of its left neighbor or if it does not exist, the vector of its right neighbor. This technique adds very little complexity but it enables constructing a mutual prediction frame Z through bidirectional motion compensated interpolation.

4.5.2. Splitting the motion estimation algorithm

A second way to combine predictive video coding techniques and DVC is to split up the motion search algorithm, i.e., restrict the encoder search space for each macroblock instead of restricting the number of macroblocks for which encoder-side motion estimation needs to be performed. In other words, the encoder calculates coarse motion vectors which are further refined by the decoder. Due to the generality of this definition, hybrid modes can be constructed in several ways. In this paper we use a subsampling approach.

Each frame W and its reference frames are subsampled by averaging four pixel values at full resolution for calculating one pixel value at low resolution. Next, bidirectional motion search is performed on blocks of size M, as in the predictive mode, but using a down-scaled search window of size $S/4$. Since the number of blocks of size M is reduced with a factor four, as well

as the number of candidate vectors to consider, the encoder computational complexity is reduced drastically. Subsequently, the low resolution motion vectors are coded and sent to the decoder. To create the mutual prediction frame Z, the motion vectors are upscaled and used for bidirectional motion compensation.

At the decoder, P and F are retrieved from the buffers with decoded frames and LP filtered. The decoded motion vectors are upscaled and used as a starting point for motion refinement. As in the DVC mode, half pixel motion refinement is performed in two passes, operating on blocks of size 16-by-16 in the first pass and 8-by-8 in the second pass. We set the half pixel refinement window to a fixed size of 5-by-5 ($S_R^{SUB_1} = 25$) for the first pass and 3-by-3 ($S_R^{SUB_2} = 9$) for the second pass. As such, the vector is never refined more than three half pixels (which is less than two pixels, i.e., the accuracy of encoder-side motion estimation). No spatial smoothing step is performed afterward, but bidirectional motion compensation follows directly.

Instead of using a subsample approach, other techniques can be used. For example, a heuristic motion search algorithm such as a Three Step Search (TSS) can be split up in executing one or two steps at the encoder side while executing the remaining steps at the decoder.

5. Rate-distortion performance

The rate-distortion performance of our system is compared to a number of different configurations. Firstly, the coding efficiency of the different modes is analyzed (Sect. 5.1). Next, we compare the DVC mode of our system to the state-of-the-art found in the literature, i.e. the DISCOVER codec (Sect. 5.2). Subsequently, the predictive mode is compared to H.264/AVC (Sect. 5.3), and finally the gain of the extended hybrid modes compared to our previous work is analyzed (Sect. 5.4).

For all these results, tests have been conducted on three video sequences: the Mother and Daughter sequence, the Foreman sequence, and the Table Tennis sequence, containing very little, moderate, and relatively high motion respectively. A GOP of length four is used, and for each sequence the maximum number of GOP's is coded (i.e. 297 frames: 75 GOP's plus one closing frame) at a frame rate of 30 Hz. Inter frames are hierarchically coded, meaning that the sequence $I_1W_1W_2W_3I_2$ is coded and decoded in the following order: $I_1I_2W_2W_1W_3$. Four different quantization patterns are used Q_0 to Q_3, quantizing each coefficient from 6 to 3 bits respectively. Quantization

of intra and inter frames is chosen in such a way so that the quality of the decoded frames is constant. Unless stated otherwise, rate distortion plots indicate the PSNR of the luma component as a function of the total rate of all color components.

5.1. Rate-distortion performance of the different modes

The rate-distortion results for the different modes are depicted in Fig. 9. We observe that the coding efficiency of the different modes is comparable in most cases, especially for the Table Tennis sequence and the high rate and low rate regions of Mother and Daughter and Foreman, respectively. Having similar performance is an advantage, because this means that we can choose the coding mode for a particular frame independently from the rate-distortion performance of each mode.

The differences in performance, particularly at low rates and high rates, can be explained as follows. Remark that there is typically a switch between the DVC mode and the predictive mode, in the sense that the DVC mode outperforms the predictive mode at low rates while this is the other way around at high rates. This difference is due to the coding of motion vectors in predictive mode. In predictive mode, motion vectors are used at the decoder side to generate the prediction Y, which is of better quality than the one obtained in the DVC mode. However, sending these motion vectors from encoder to decoder introduces a rate penalty that is not present in the DVC mode. At high rates, the rate of the motion vectors is negligible compared to the number of WZ bits spent. At low rates, however, sending the motion vectors to the decoder is an important penalty, especially for sequences that can be well predicted at the decoder-side (such as Mother and Daughter). This explains why the DVC mode outperforms the predictive mode for Mother and Daughter significantly for low rates.

We can draw the same conclusions for the hybrid modes, which lie more or less in between the predictive mode and the DVC mode.

5.2. Rate-distortion performance compared to DISCOVER

The executable of the DISCOVER codec is available online [42], and the codec architecture has been described in detail [15]. As for our system, we perform experiments with the DISCOVER codec using a fixed GOP of size four, and we choose the intra quantization parameters so that the quality of the intra decoded frames and WZ decoded frames are the same. Results are generated using WZ quantization patterns 1, 3, 6, 7, and 8. The results in

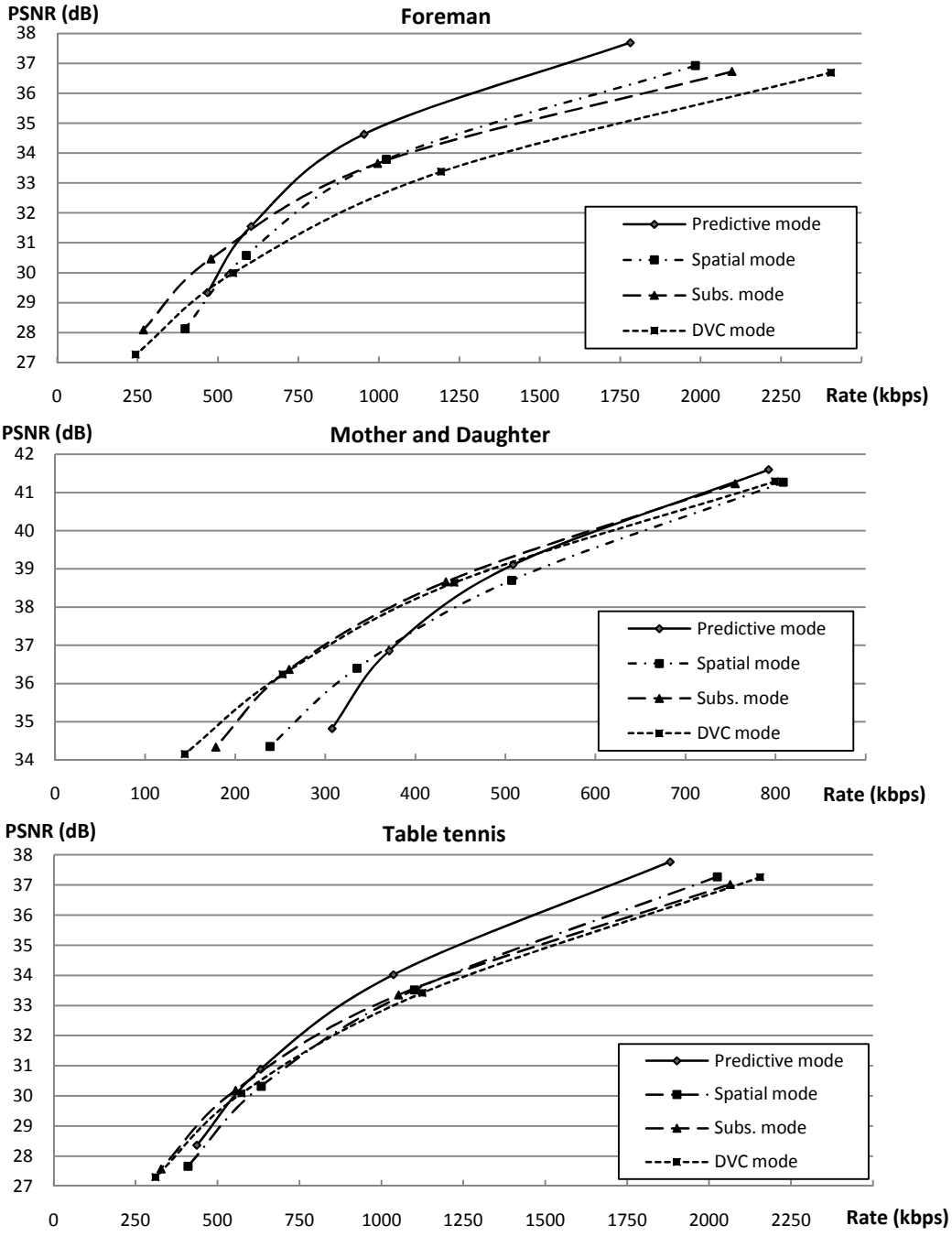

Figure 8: Rate-distortion plots for the different modes of the proposed codec.

this section are limited to the luma component only, since the DISCOVER codec does not take into account the chroma.

We expect similar performance between the DVC mode of our system and the DISCOVER codec, because motion estimation and virtual noise estimation is similar in both systems. An important difference in our system is the use of residual coding, which is expected to increase the coding efficiency [30], especially for sequences with low motion content (and a static camera) such as the Mother and Daughter sequence. This explains why the DVC mode of our codec outperforms the DISCOVER codec significantly for Mother and Daughter, while the performance for Foreman and Table are rather similar (Fig. 9). Another difference is that the DISCOVER codec uses LDPC codes which are reported to perform slightly better than turbo codes [42].

5.3. Rate-distortion performance compared to H.264/AVC

The H.264/AVC reference software (JM 13.2) is used to create two reference RD curves for each sequence: one with an IBBB GOP structure, and one with only intra-coded frames (IIII). The extended profile is used, RDO enabled, one slice per picture, CAVLC entropy coding, and the sequences were coded using a fixed QP (the same for I and B frames).

The results (Fig. 10) indicate that there is a significant gap between H.264/AVC inter coding and the predictive mode of our system. This is due to the fact that the H.264/AVC codec uses advanced techniques such as adaptive block sizes, skip modes, etc.

5.4. Rate-distortion performance compared to our previous work

The hybrid modes developed in our previous work have been extended in this paper by residual coding and subpixel motion refinement. Some other techniques are included, such as online correlation noise modeling between the original and the side information. As such, significant improvements can be observed (Fig. 11).

6. Validation of complexity analysis

The complexity for each of the modes has been calculated using the number of pixel read/write operations (Tab. 1). How these results should be mapped to the number of CPU cycles or milliseconds spent for coding each inter frame W depends on hardware details such as calculation speed, cache behavior etc. However, the scaling factor used to convert the number of pixel

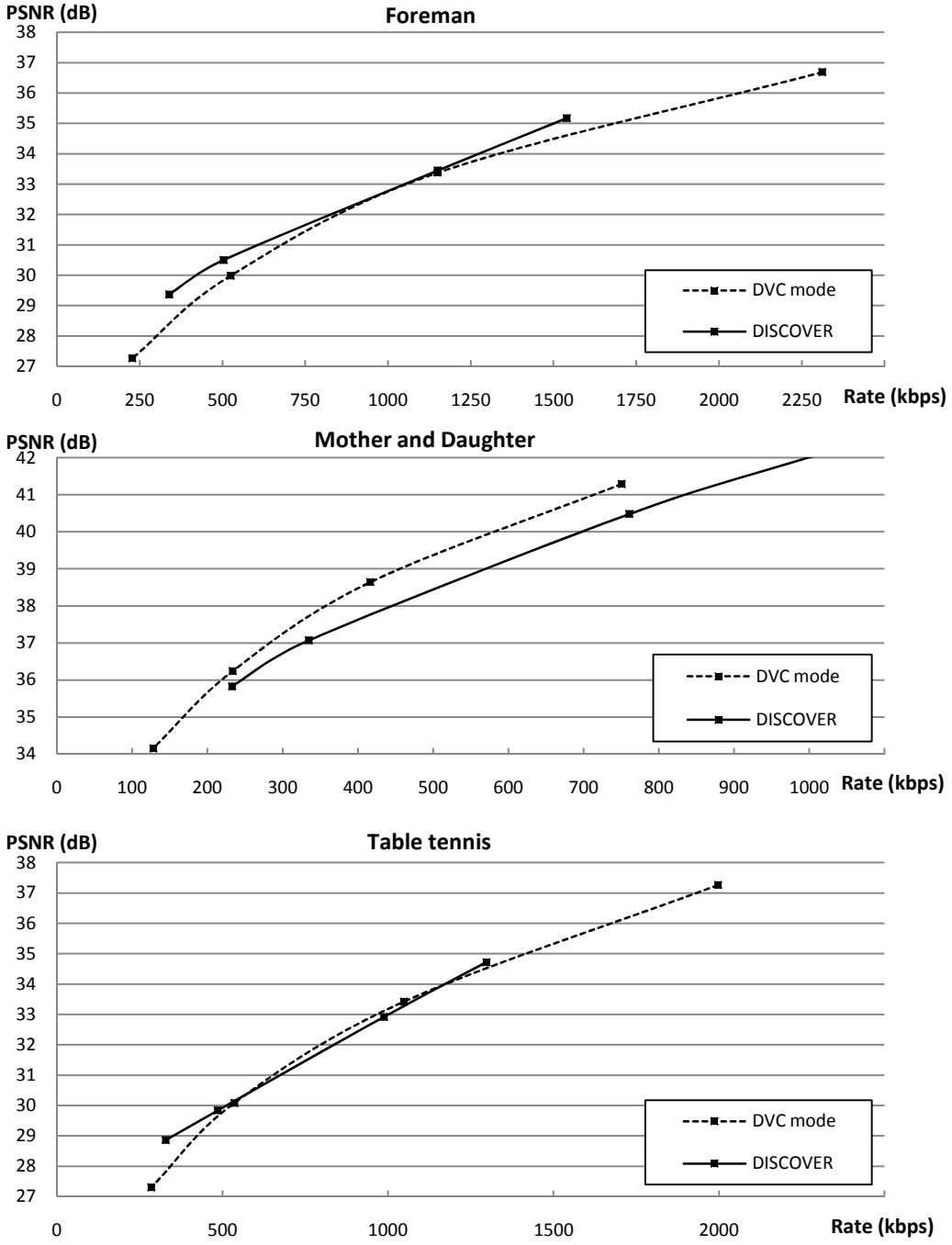

Figure 9: Rate-distortion plots comparing the DVC mode of our codec to the DISCOVER system.

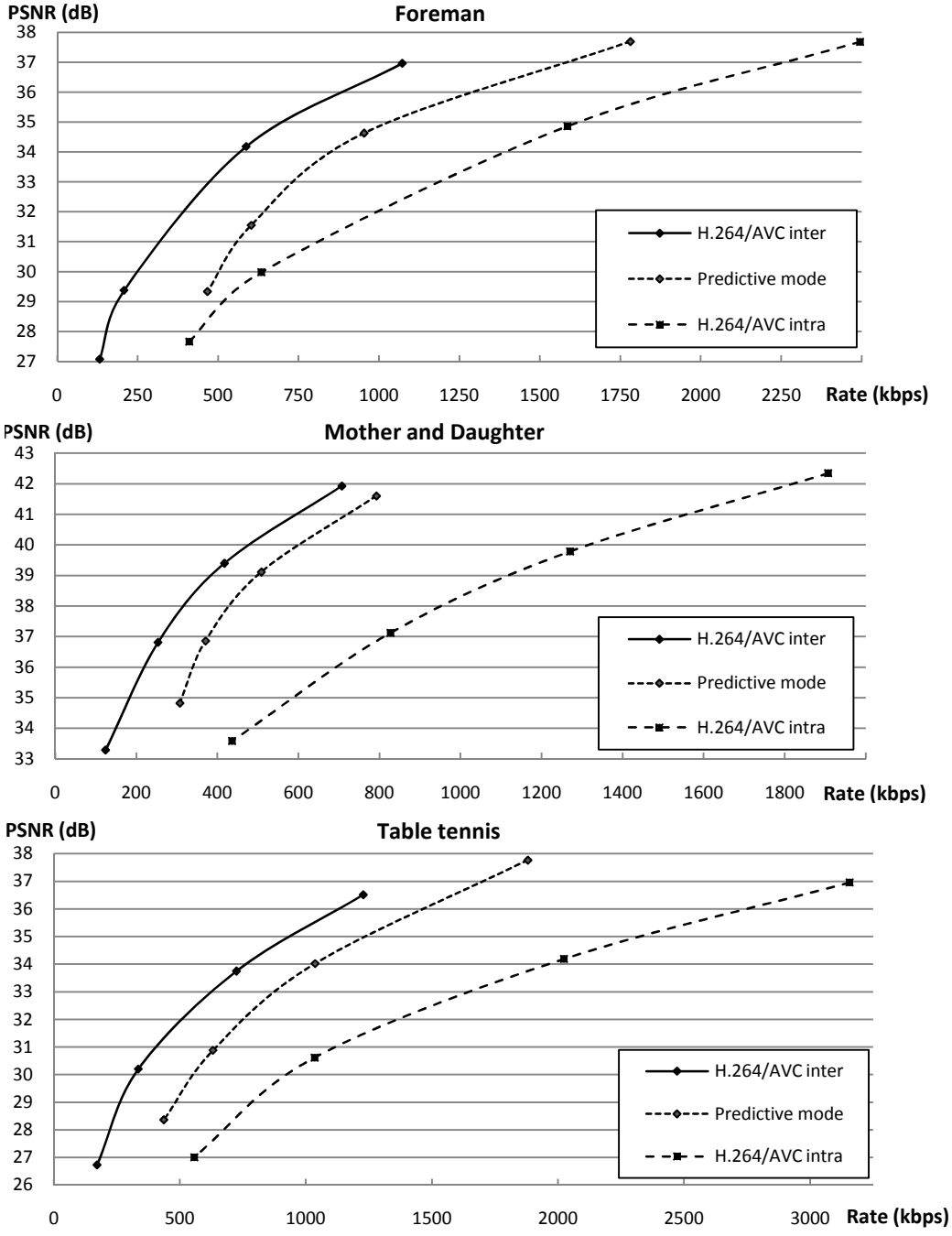

Figure 10: Rate-distortion results for the predictive mode of our codec, compared to H.264/AVC.

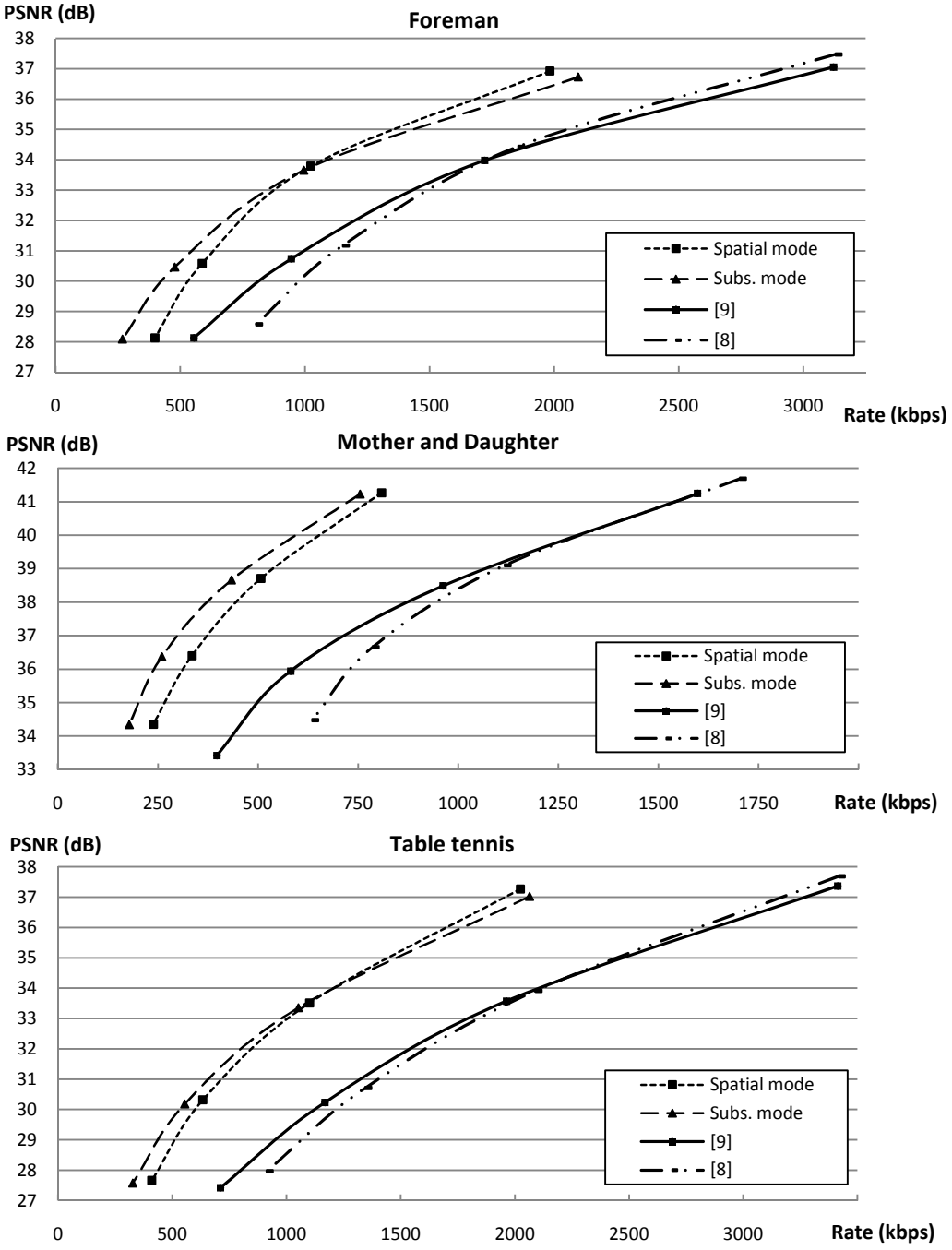

Figure 11: Rate-distortion results for the hybrid modes, compared to our earlier work.

Table 2: Comparing calculated and measured complexity of the motion estimation process shows that our model is fairly accurate.

	Calculated		**Measured**	
	Encoder	Decoder	Encoder	Decoder
Predictive mode	>99.5%	<0.5%	99%	1%
DVC mode	<0.5%	46%	<0.5%	48%
Hybrid spat. mode	50%	5%	50%	8%
Hybrid subs. mode	6%	7%	7%	8%

Table 3: Specifications of the test system used.

CPU	QC CLOVERTOWN XEON X5355 2.66Ghz 8M 1333FSB 120W
Memory	4 × DDR II 4048MB (DDR2-667 ECC FB-DIMM FMHS LP)
Operating sytem	Microsoft Windows Server R2, Standard X64 edition Service Pack 2

read/write operations to the specific metric desired should be more or less the same for each mode. Hence, theoretical and practical metrics should show the same relationships between encoder and decoder complexity. In other words, if theoretical analysis indicates that the encoder in the predictive mode is twice as complex as the encoder in the spatial mode, then this should be observed in practice also. Therefore, we normalize theoretical complexity and measured complexity to the total complexity in the predictive mode (encoder and decoder) (Tab. 2). Practical measurements have been performed on our test system (Tab. 3), measuring the execution time of the motion estimation process, and averaging values for all sequences and rate points. The results show that our model for measuring complexity using the number of pixel read/writes is reasonably accurate. Some inaccuracies can be observed for the modes that have the lowest complexity (e.g. spatial mode, decoder-side). This is due to not modeling the complexity of operations, function calls, etc.

7. Video coding with controllable complexity

The previous sections illustrated how several modes can be constructed with different distributions of complexity between encoder and decoder, with comparable rate-distortion performance of the modes. In this section we will develop a strategy for choosing which coding mode to use for coding each inter frame, in order to meet encoder and decoder complexity constraints (Sect. 7.1). An example is provided to illustrate the different configurations that can be achieved by combining modes (Sect. 7.2).

7.1. Controlling complexity using encoder and decoder complexity constraints

We assume that techniques are available to estimate or calculate the complexity available at encoder and decoder, and that these values are accurate for coding the following K inter frames W. Denote the available complexity per frame at the encoder (decoder) as C_E (C_D), respectively. We will define a method to calculate the optimal linear combination of modes that meets this constraint. The method will be optimal in the sense that the total complexity of the system is minimized.

To code one frame using mode m_i, a complexity budget of M_i^E is needed at the encoder, and a budget of M_i^D is needed at the decoder. From the K inter frames, α_i frames will be coded using mode i. Hence, this optimization problem can be formulated as follows:

Minimize:

$$\sum_i \alpha_i (M_i^E + M_i^D)$$

subject to:

$$\sum_i \alpha_i = K,$$

$$\sum_i \alpha_i M_i^E \leq K \cdot C_E,$$

$$\sum_i \alpha_i M_i^D \leq K \cdot C_D,$$

$$0 \leq \alpha_i \leq K \ , \forall i. \tag{5}$$

Remark also that one could favor encoder or decoder complexity decrease, by introducing a weighing factor δ in the cost function:

$$\sum_i \alpha_i (M_i^E + \delta M_i^D). \tag{6}$$

This problem can be solved, for example, exhaustively or by using integer linear programming techniques (ILPs). In the following section, we will use a graphical exhaustive method to find a solution.

7.2. Example

We will illustrate how to choose coding modes by means of an example for a particular set of parameters (K, C_E, C_D, and δ). While this is only one out of many configurations, similar reasoning can be used for other parameters.

As an example, we will decide which coding modes to use for coding the following $K = 3$ inter frames. We use the results from our complexity analysis (Tab. 1), and express available complexity in terms of the number of pixel read/write operations that can be performed for coding these K frames. Assume for example that $C_E = 325 \cdot 10^6$, $C_D = 225 \cdot 10^6$, and $\delta = 1$.

Each out of K inter frames can be coded using one out of N modes, resulting into $(K + N - 1, K)$ possible ways to code the GOP. In this case ($K = 3$, $N = 4$) the binomial resolves to 20. Each of these 20 solutions can be described by the tuple $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ indicating how the three inter frames are coded: using α_0 times the predictive mode, α_1 times the spatial mode, α_2 times the subsample mode, and α_3 times the DVC mode. Given the complexity of each mode (Tab. 1), each tuple has an associated average encoding complexity per frame $\frac{\sum_i \alpha_i M_i^E}{K}$ and an average decoding complexity of $\frac{\sum_i \alpha_i M_i^D}{K}$. We can use these two values as coordinates for representing the 20 solutions in a plane (Fig. 12). It is easily verified that solutions featuring only two modes i and j lie on a straight line connecting the solutions where i is used exclusively and where j is used exclusively.

From these 20 points, some are suboptimal in the sense that there exists always a better way to code the GOP, i.e., with lower or at most equal encoder and decoder complexity. These points are indicated in gray. The other points (indicated in black) are so-called pareto-optimal. These 12 pareto-optimal points provide the range for distributing complexity between encoder and decoder. We can see that $(1, 0, 0, 2)$ and $(2, 0, 0, 1)$ are not pareto-optimal, which indicates that using the hybrid modes enables more efficient distribution of complexity than combining only the DVC mode and the predictive mode. In addition, since all modes are present in the optimal set, this figure shows that no mode is redundant.

Given the encoder and decoder constraints, illustrated by the gray rectangle, from the pareto-optimal points the point minimizing the cost function

is chosen. This means that in this case all three frames should be coded using only the hybrid subsample mode $(0, 0, 3, 0)$. Other solutions satisfying the complexity constraints are suboptimal in this context, since they do not minimize the cost function.

From this example several advantages for using our system can be identified. Firstly, our system provides a solution in case neither encoder nor decoder have enough resources to perform all motion estimation (in our example, neither the predictive mode $(3, 0, 0, 0)$ nor the DVC mode $(0, 0, 0, 3)$ satisfy C_E and C_D). Secondly, due to the fact that there is no dependency between the modes, any frame can be coded using any mode at any time. As a consequence, it is possible to adapt rapidly (with a maximum delay of K frames) to varying complexity constraints that can be imposed by devices with variable power supply, or by systems featuring multi-tasking. In case encoder complexity constraints are drastically reduced, motion estimation is shifted to the decoder side yielding a different solution for coding the GOP (Fig. 13). On the other hand, if decoder complexity constraints are reduced (Fig. 14), motion estimation is shifted to the encoder side.

In this example we assumed a very short time during which encoder and decoder complexity constraints remain constant. In practice, however, complexity constraints are likely to be constant over a larger period of time. As K becomes larger, distribution of complexity can be performed more subtle, since the pareto-optimal set will contain more solutions. This is illustrated by Fig. 15, where $K = 10$ results into 40 pareto-optimal solutions.

8. Final remarks

The complexity analysis has been limited to the part involved in motion estimation because only this part is functionally different for each mode. It should be noted however that also the complexity of the turbo decoding process varies between the modes. This is due to the fact that the number of decoding iterations that need to be executed by the turbo decoder, depends on the quality of Y_R compared to the original R available at the encoder. When shifting the motion estimation from the encoder to the decoder, the quality of Y_R will typically decrease due to the absence of the original at the decoder. As a result, less bitplanes will be skipped by the turbo decoder and more decoding iterations will be needed per bitplane, as illustrated in Table 4. To avoid evaluating a possibly non optimal rate request strategy,

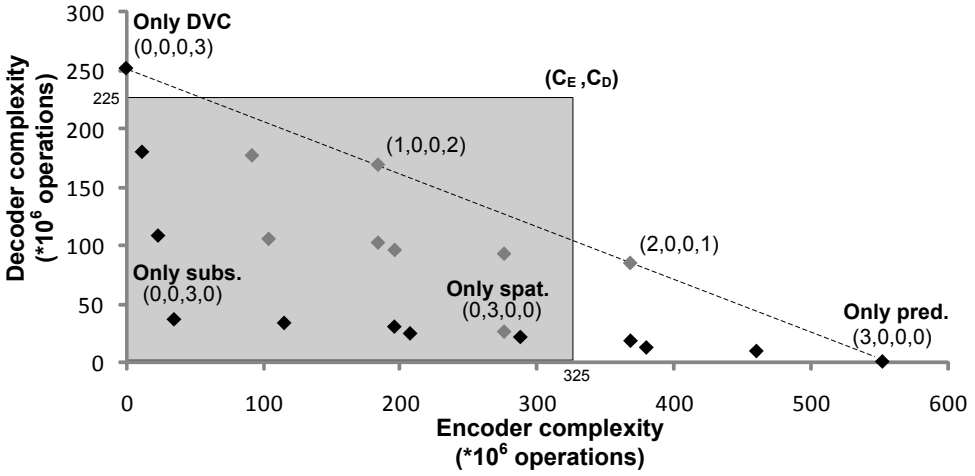

Figure 12: Encoder and decoder complexity constraints define a set of possible combinations for coding the GOP (indicated by the gray rectangle). From this set the point minimizing the cost function can be selected. In this case, coding all three inter frames using the subsample mode is optimal.

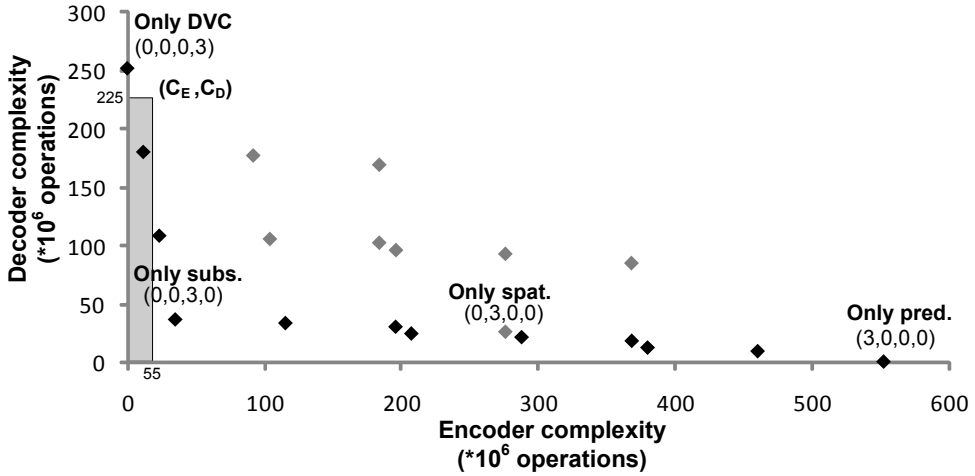

Figure 13: If encoder complexity constraints are reduced, motion estimation complexity is shifted to the decoder side, yielding $(0, 0, 1, 2)$ as the optimal solution. As such, for two out of three frames the decoder will perform all motion estimation.

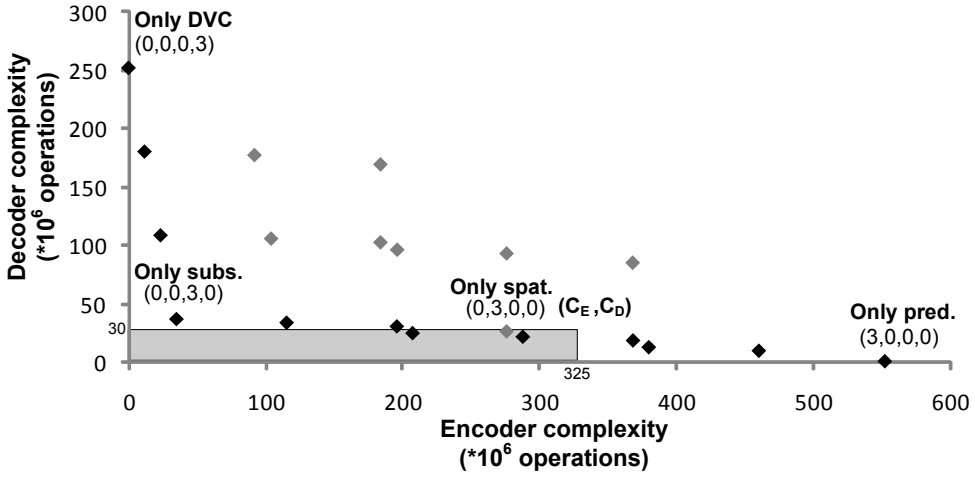

Figure 14: If decoder complexity constraints are reduced, motion estimation complexity is shifted to the encoder side, yielding $(1, 0, 2, 0)$ as the optimal solution.

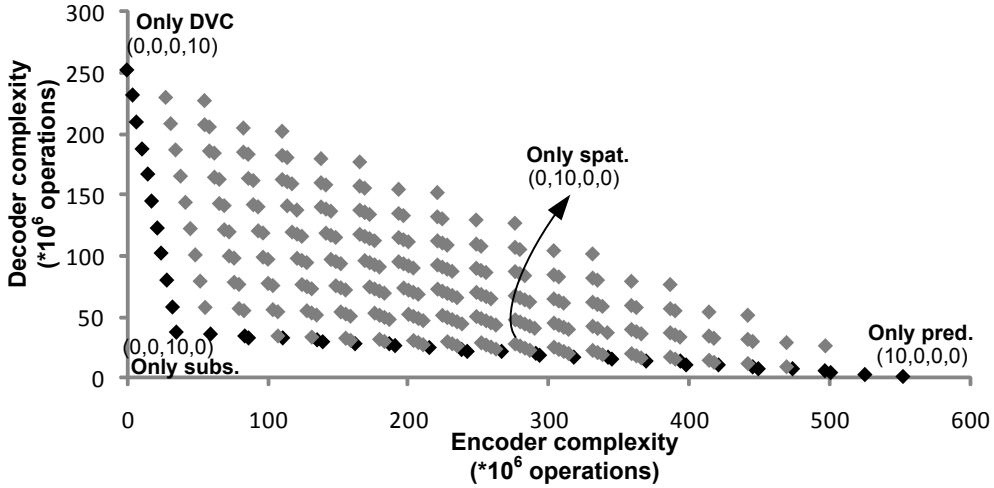

Figure 15: Complexity can be distributed more subtle if K is large. For example, $K = 10$ results in 40 points being part of the pareto-optimal set, providing a wide range of coding options between maximum encoder complexity and maximum decoder complexity.

Table 4: Average number of turbo decoding iterations per bitplane, for the Foreman sequence under the same test conditions as described in the results section.

	Pred. mode	Hybrid subs.	Hybrid spat.	DVC mode
Q_5	0.2	0.8	1.3	1.7
Q_4	1.6	2.9	3.1	4.2
Q_3	4.0	6.1	5.8	7.6
Q_2	8.0	9.9	9.7	11.5

only the final decoding pass is considered (i.e. when the correct number of WZ bits is received from the encoder).

Hence, when going from the predictive mode to the DVC mode over the hybrid modes, the complexity of the decoder is not only increased by motion estimation, but also by turbo decoding. Taking this into account is part of future work.

9. Conclusions and future work

In this paper we presented a video codec where motion estimation is shared flexibly between encoder and decoder. Several modes for coding inter frames have been defined with different distributions of complexity between encoder and decoder, and complexity analysis allowed to define a scheme for choosing which coding modes to use for coding a group of frames. As such, the video codec is able to adapt to varying complexity constraints imposed by the encoder as well as by the decoder.

Apart from extending the complexity analysis, future work includes improving the coding efficiency of the different modes. Distributed video coding systems are still not able to achieve a coding performance comparable to state-of-the-art video coding with encoder-side motion estimation, such as H.264/AVC. New insights in the domain of distributed video coding can be used to improve the hybrid modes as well, by clever partitioning of the motion estimation performed at the encoder and the motion estimation performed at the decoder.

Acknowledgements

The research activities that are described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and

Technology in Flanders (IWT-Flanders), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

[1] ITU-T and ISO/IEC, *Text of committee draft of Joint Video Specification*, 2002. ITU-T Recommendation H.264 – ISO/IEC 14496-10 AVC (MPEG-4 Part 10).

[2] F. Pereira, L. Torres, C. Guillemot, T. Ebrahimi, R. Leonardi, and S. Klomp, "Distributed Video Coding: Selecting the most promising application scenarios," in *Signal Processing : Image Communication*, pp. 339–352, 2008.

[3] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 645–658, May 2005.

[4] E. Kaminsky, D. Grois, and O. Hadar, "Dynamic computational complexity and bit allocation for optimizing H.264/AVC video compression," *Journal of Visual Communication and Image Representation*, pp. 56–74, January 2008.

[5] H. Ates, B. Kanberoglu, and Y. Altunbasak, "Rate-distortion and complexity joint optimization for fast motion estimation in H.264 video coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 37–40, October 2006.

[6] J. Stottrup-Andersen, S. Forchhammer, and S. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *Proc. IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 111–114, October 2004.

[7] C. Brites, J. Ascenso, J. Q. Pedro, and F. Pereira, "Evaluating a feedback channel based transform domain wyner-ziv video codec," *Signal Processing: Image Communication*, pp. 269–297, 2008.

[8] S. Mys, J. Slowack, J. Škorupa, P. Lambert, and R. Van de Walle, "Dynamic complexity coding: Combining predictive and Distributed

Video Coding," in *Proc. Picture Coding Symposium (PCS)*, November 2007.

[9] J. Škorupa, S. Mys, J. Slowack, P. Lambert, and R. Van de Walle, "Heuristic dynamic complexity coding," in *Proc. SPIE*, April 2008.

[10] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[11] A. D. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, pp. 1–10, 1976.

[12] R. Puri and K. Ramchandran, "PRISM: A new robust video coding architecture based on distributed compression principles," in *Proc. Allerton Conference on Communication, Control and Computing*, October 2002.

[13] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. IEEE Data Compression Conference (DCC)*, pp. 252–261, April 2002.

[14] A. Aaron, S. Rane, E. Setton, and B. Girod, "Transform-domain Wyner-Ziv codec for video," in *Proc. SPIE Visual Communications and Image Processing*, January 2004.

[15] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The DISCOVER codec: Architecture, techniques and evaluation," in *Picture Coding Symposium*, 2007.

[16] M. Tagliasacchi, A. Trapanese, S. Tubaro, J. Ascenso, C. Brites, and F. Pereira, "Exploiting spatial redundancy in pixel domain Wyner-Ziv video coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, October 2006.

[17] A. Adikari, W. Fernando, H. Arachchi, and W. Weerakkody, "Wyner-Ziv coding with temporal and spatial correlations for motion video," in *Canadian Conference on Electrical and Computer Engineering*, pp. 1188–1191, May 2006.

[18] A. Adikari, W. Fernando, H. Arachchi, and W. Weerakkody, "Multiple side information streams for Distributed Video Coding," *IEEE Electronics Letters*, vol. 42, pp. 1447–1449, December 2006.

[19] C. Brites, and F. Pereira, "Correlation noise modeling for efficient pixel and transform domain Wyner-Ziv video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, pp. 1177–1190, September 2008.

[20] J. Škorupa, J. Slowack, S. Mys, P. Lambert, and R. Van de Walle, "Accurate Correlation Modeling for Transform-Domain Wyner-Ziv Video Coding," in *Proc. Pacific-Rim Conference on Multimedia (PCM)*, pp. 1-10, December 2008.

[21] J. Slowack, S. Mys, J. Škorupa, P. Lambert, C. Grecos, and R. Van de Walle, "Accounting for quantization noise in online correlation noise estimation for Distributed Video Coding," in *Proc. Picture Coding Symposium (PCS)*, May 2009.

[22] Z. Li, L. Liu, and E. J. Delp, "Rate distortion analysis of motion side estimation in Wyner-Ziv video coding," *IEEE Transactions on Image Processing*, vol. 16, pp. 98–113, January 2007.

[23] M. Tagliasacchi, L. Frigerio, and S. Tubaro, "Rate-distortion analysis of motion-compensated interpolation at the decoder in Distributed Video Coding," *IEEE Signal Processing Letters*, vol. 14, pp. 625–628, September 2007.

[24] M. Tagliasacchi and S. Tubaro, "Hash-based motion modeling in Wyner-Ziv video coding," in *Submitted to IEEE International Conference on Acoustics Speech and Signal Processing*, April 2007.

[25] J. Pedro, C. Brites, J. Ascenso, and F. Pereira, "Studying the feedback channel in transform domain Wyner-Ziv video coding," in *6th Conference on Telecommunications - ConfTele*, May 2007.

[26] M. Tagliasacchi, J. Pedro, F. Pereira, and S. Tubaro, "An efficient request stopping method at the turbo decoder in Distributed Video Coding," in *EURASIP European Signal Processing Conference*, September 2007.

[27] Jozef Škorupa and Jürgen Slowack and Stefaan Mys and Peter Lambert and Christos Grecos and Rik Van de Walle, "Stopping criterions for turbo coding in a Wyner-Ziv video codec," in *Proc. Picture Coding Symposium (PCS)*, May 2009.

[28] M. Morbée, J. Prades-Nebot, A. Pizurica, and W. Philips, "Feedback channel suppression in pixel-domain Distributed Video Coding," in *Annual Workshop on Circuits, Systems and Signal Processing (ProRISC)*, pp. 154–157, November 2006.

[29] L. Liu and E. J. Delp, "Wyner-Ziv video coding using LDPC codes," in *Proc. IEEE Nordic Signal Processing Symposium (NORSIG 2006)*, June 2006.

[30] A. Aaron, D. Varodayan, and B. Girod, "Wyner-Ziv residual coding of video," in *Proc. Picture Coding Symposium (PCS)*, April 2006.

[31] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres, "Overlapped quasi-arithmetic codes for Distributed Video Coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, September 2007.

[32] R. Bernardini, M. Fumagalli, M. Naccari, R. Rinaldo, M. Tagliasacchi, S. Tubaro, and P. Zontone, "Error concealment using a dvc approach for video streaming applications," in *EURASIP European Signal Processing Conference*, September 2007.

[33] P. Baccichet, S. Rane, and B. Girod, "Systematic lossy error protection based on H.264/AVC redundant slices and flexible macroblock ordering," *Journal of Zheijang University, Science A*, pp. 727–736, may 2006.

[34] S. Rane and B. Girod, "Analysis of error-resilient video transmission based on systematic source-channel coding," in *Proc. Picture Coding Symposium (PCS)*, December 2004. Invited Paper.

[35] M. Flierl and B. Girod, "Coding of multi-view image sequences with video sensors," in *IEEE International Conference on Image Processing (ICIP)*, October 2006.

[36] F. Yang, Q. Dai, and G. Ding, "Multi-view images coding based on multiterminal source coding," in *Proc. IEEE International Conference*

on Acoustics, Speech and Signal Processing (ICASSP), pp. I–1037–I–1040, April 2007.

[37] I. Tosic and P. Frossard, "Wyner-Ziv coding of multi-view omnidirectional images with overcomplete decompositions," in *IEEE International Conference on Image Processing (ICIP)*, September 2007.

[38] D. Mukherjee, "A robust reversed-complexity Wyner-Ziv video codec introducing sign-modulated codes," Tech. Rep. HPL-2006-80, HP Laboratories Palo Alto, May 2006.

[39] C. Tonoli, P. Migliorati, and R. Leonardi, "Error resilience in current distributed video coding architectures," *EURASIP Journal on Image and Video Processing*,vol. 2009, article ID 946585, 2009.

[40] D. Kubasov, J. Nayak, and C. Guillemot, "Optimal reconstruction in Wyner-Ziv video coding with multiple side information," in *IEEE MultiMedia Signal Processing Workshop*, October 2007.

[41] E. Brockmeyer, L. Nachtergaele, F. V. M. Catthoor, J. Bormans, and H. J. De Man, "Low Power Memory Storage and Transfer Organization for the MPEG-4 Full Pel Motion Estimation on a Multimedia Processor," *IEEE Transactions on Multimedia*, vol. 1, No. 2, June 1999.

[42] www.discoverdvc.org