Signal Processing: *Image Communication* 23 (2008) 369–378

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Wyner–Ziv coding of video with unsupervised motion vector learning

David Varodayan *, David Chen, Markus Flierl, Bernd Girod

Max Planck Center for Visual Computing and Communication, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Received 26 March 2008
Accepted 4 April 2008

Keywords:
Wyner–Ziv video coding
Expectation maximization

ABSTRACT

Distributed source coding theory has long promised a new method of encoding video that is much lower in complexity than conventional methods. In the distributed framework, the decoder is tasked with exploiting the redundancy of the video signal. Among the difficulties in realizing a practical codec has been the problem of motion estimation at the decoder. In this paper, we propose a technique for unsupervised learning of forward motion vectors during the decoding of a frame with reference to its previous reconstructed frame. The technique, described for both pixel-domain and transform-domain coding, is an instance of the expectation maximization algorithm. The performance of our transform-domain motion learning video codec improves as GOP size grows. It is better than using motion-compensated temporal interpolation by 0.5 dB when GOP size is 2, and by even more when GOP size is larger. It performs within about 0.25 dB of a codec that knows the motion vectors through an oracle, but is hundreds of orders of magnitude less complex than a corresponding brute-force decoder motion search approach would be.

1. Introduction

Wyner–Ziv coding of video offers an alternative to predictive video coding methods for applications requiring low-complexity encoding [7,13]. In this paper, we limit the encoding complexity to a level at which video frames can be encoded separately, not jointly. Separately encoded frames can nevertheless be decoded jointly, since there is no complexity constraint at the decoder. Specifically, each Wyner–Ziv encoded frame can be decoded with reference to side information derived from one or more already reconstructed frames. The Slepian–Wolf [14] and Wyner–Ziv [20] theoretical results indicate that the penalty in performance for separate (or distributed) encoding should be small, but as a current survey paper notes, "despite recent advances, distributed video coding rate-distortion (RD) performance is not yet at the level of predictive coding" [8].

One reason is that separate encoding of frames precludes motion estimation at the encoder. The decoder, instead, estimates the motion in order to construct appropriately motion-compensated side information. Three basic approaches for estimating motion at the decoder have been suggested.

1. Repeated decoding using side information compensated by every possible motion field was proposed in [13]. This method can be employed in conjunction with blockwise Slepian–Wolf coding/decoding. But it is astronomical in computation for framewise coding/decoding because of the vast number of motion field configurations.
2. Transmission of a sufficient fraction of frames as keyframes (which are decoded without reference to side information) enables motion-compensated temporal interpolation (MCTI) at the decoder [2,12].

* Corresponding author. Tel.: +1 650 724 3647.
E-mail address: varodayan@stanford.edu (D. Varodayan).

0923-5965/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2008.04.009

This penalizes coding efficiency significantly because keyframes incur a high-encoding rate.

3. Supplementary transmission of robust hash information about each block of an encoded frame lets the decoder perform a rudimentary motion search [1]. Since the encoder transmits the hashes at a constant rate, it wastes bits when the motion is small. On the other hand, if there is too much change between frames, the fixed-rate hash may be insufficient for reliable motion search.

Each of these three methods suffers from either excessive computational burden or the need for a significant supplementary bitstream that is decoded without reference to side information. In this paper, we propose an alternative technique for the decoder to use only the Wyner–Ziv bitstream of a frame to efficiently learn the forward motion vectors with respect to the previous reconstructed frame. We have already applied this method to the closely related problem of distributed compression of stereo images, in which one encoded image is decoded with reference to side information derived from disparity-compensated versions of the other image. Our algorithm decodes the image while learning the disparity field between the pair of images [4,5,17–19].

In Section 2, we develop the technique for pixel-domain Wyner–Ziv video decoding that learns forward motion vectors unsupervised. We describe the algorithm formally within the framework of expectation maximization (EM) [6] in Section 3. Section 4 extends the technique to the transform domain to exploit the spatial redundancy within frames. Section 5 reports simulation results for transform-domain Wyner–Ziv video coding based on this technique for unsupervised motion vector learning at the decoder.

2. Pixel-domain technique for forward motion vector learning at the decoder

2.1. Overview

Consider the luminance components of two consecutive video frames, quantized to bit depth d, where $d>1$. Let X be the quantized luminance frame to be encoded, and Y the previous quantized luminance frame available at the decoder. Denote by M the forward motion vector field that relates X to Y. The challenge is to encode X efficiently in the absence of Y so that it can be reliably decoded in the presence of Y.

Fig. 1 depicts three compression systems that can be applied to this problem. The encoder of all three systems is identical. It computes the syndrome S of X with respect to a low-density parity-check (LDPC) code, as in [11]. Our implementation uses a rate-adaptive LDPC accumulate code [15,16] to facilitate rate control using a feedback channel. This permits the encoder to send additional syndrome bits, at the request of the decoder if the reconstruction of X is inconsistent with S. The three systems differ in decoding.

The baseline system in Fig. 1(a) performs decoding of X with respect to the colocated pixels of Y; in other words, with zero motion. Initially, the LDPC decoder determines the soft estimate θ (statistical estimate of X) with soft side information ψ by applying a probability model to the colocated pixels of Y. It then refines these estimates using S via an iterative belief propagation algorithm. In regions where motion exists between X and Y, this scheme performs poorly.

For comparison, Fig. 1(b) shows an impractical decoder endowed with a motion oracle. The oracle informs the probability model which pixels of Y should be used for the side information ψ during LDPC decoding.

Finally, Fig. 1(c) depicts a practical decoder that learns the forward motion field M via EM. In place of the motion oracle, a motion estimator maintains an a posteriori probability distribution on M, by comparing Y and the soft estimate θ of X from the LDPC decoder. Every iteration of LDPC decoding sends the motion estimator a current soft estimate θ in order to refine the distribution on M. In return, the probability model updates the side information ψ for the LDPC decoder by blending information from the pixels of Y according to the refined distribution on M. A formal EM treatment of this algorithm is provided in Section 3.

2.2. Joint bitplane LDPC decoding

For the motion estimator in Fig. 1(c) to update the probability distribution on the motion field M accurately, it should compare Y to the soft estimate θ over all d bitplanes at once. The consequence for the LDPC decoder is that it too must decode all d bitplanes of X at once, unlike the LDPC decoder of [11].

We propose joint bitplane LDPC decoding to model and exploit the redundancy across the d bits that represent a pixel, using the belief propagation decoding graph shown in Fig. 2. Like the LDPC decoder of [11], it decodes X at bit nodes subject to constraints set at syndrome nodes, by propagating log likelihood ratios[1] of the bit beliefs along the edges of the graph. The difference is that the side information ψ no longer supplies log likelihood ratios to the bit nodes directly, but instead feeds new nodes, called symbol nodes. As depicted in Fig. 2, each symbol node (one per pixel) collects ψ_{pixel} together with log likelihood ratios $\log(\alpha_g/(1-\alpha_g))$ from each bit node $g\in\{1,\ldots,d\}$ associated with that pixel. After local computation, the symbol node sends log likelihood ratios $\log(\beta_g/(1-\beta_g))$ to each of those bit nodes.

In the case of Fig. 2, $d=3$ so the side information distribution is eight-valued. We use a Gray mapping for reasons described in [5]:

$$\psi_{\text{pixel}}=(p_{000},p_{001},p_{011},p_{010},p_{110},p_{111},p_{101},p_{100}).$$

The log likelihood ratio sent to the bit nodes are computed via the sum–product algorithm [10]. For example, for bit node 2,

$$\log\frac{\beta_2}{1-\beta_2}=\log\frac{\sum_{a=0,1}\sum_{c=0,1}p_{a1c}\alpha_1^a(1-\alpha_1)^{1-a}\alpha_3^c(1-\alpha_3)^{1-c}}{\sum_{a=0,1}\sum_{c=0,1}p_{a0c}\alpha_1^a(1-\alpha_1)^{1-a}\alpha_3^c(1-\alpha_3)^{1-c}}.$$

[1] The log likelihood ratio is the logarithm of the ratio of the likelihood of a certain bit being 1 to the likelihood of it being 0.

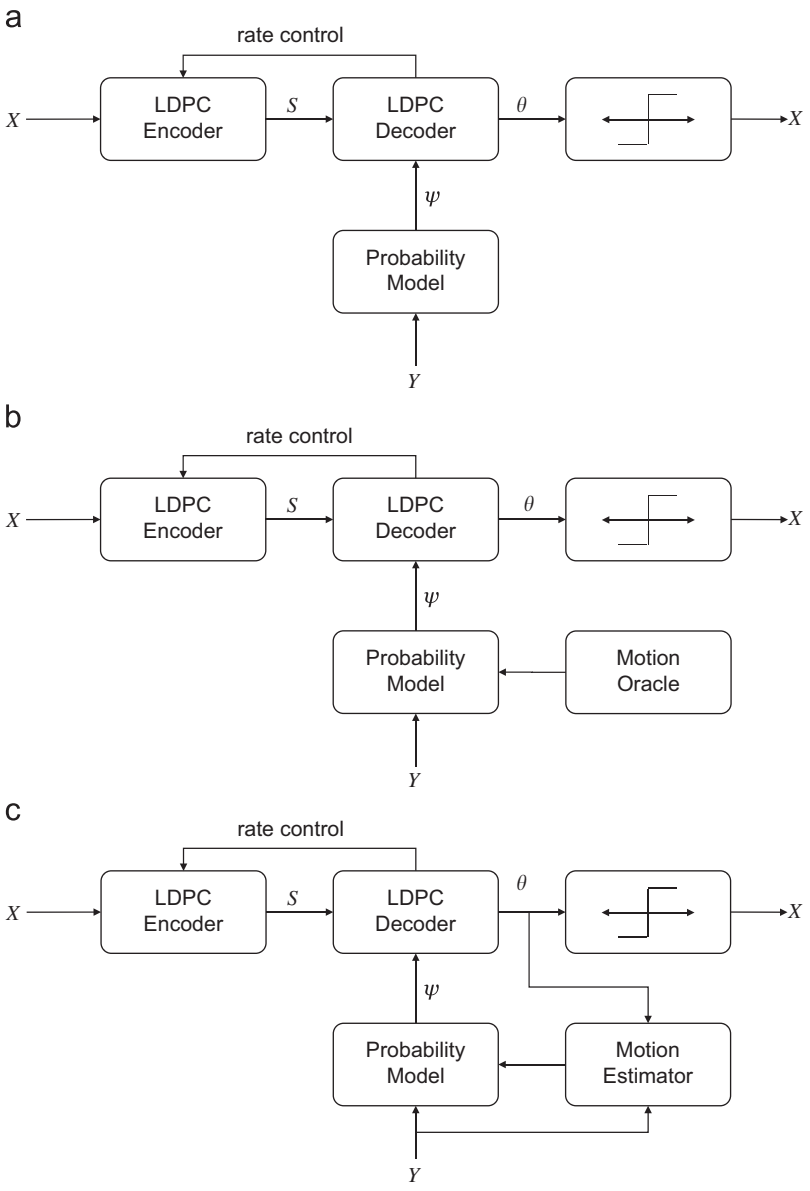

Fig. 1. Coding X with respect to Y with (a) zero motion, (b) a motion oracle and (c) unsupervised forward motion vector learning.

This formula begins by multiplying the elements p_{abc} of ψ_{pixel} by elements of the binary distributions $(\alpha_1, 1-\alpha_1)$ and $(\alpha_3, 1-\alpha_3)$ according to the values of the first bit a and third bit c, respectively, of their indices. These products are summed in two groups according to the value of the second bit b, before the log likelihood ratio is taken. In calculating $\log(\beta_2/(1-\beta_2))$, we avoid using $(\alpha_2, 1-\alpha_2)$ to prevent recycling information to bit node 2. Calculations of $\log(\beta_1/(1-\beta_1))$ and $\log(\beta_3/(1-\beta_3))$ follow analogously by shuffling the roles of the first, second and third bits of the index.

The bit nodes forward the log likelihood ratios $\log(\beta_g/(1-\beta_g))$ to the syndrome nodes, which apply LDPC syndrome decoding rules [11] to reply to the bit nodes. These in turn update the log likelihood ratios $\log(\alpha_g/(1-\alpha_g))$, sent to the symbol nodes.

Finally, the symbol nodes compute the output soft estimate θ_{pixel} by multiplying the elements of ψ_{pixel} by elements of all three updated distributions $(\alpha_1, 1-\alpha_1)$, $(\alpha_2, 1-\alpha_2)$ and $(\alpha_3, 1-\alpha_3)$ according to the values of the first, second and third bits, respectively. After inverse Gray mapping and normalization,

$$\theta_{\text{pixel}} = (q_{000}, q_{001}, q_{011}, q_{010}, q_{110}, q_{111}, q_{101}, q_{100}),$$

where

$$q_{abc} \propto p_{abc}\alpha_1^a(1-\alpha_1)^{1-a}\alpha_2^b(1-\alpha_2)^{1-b}\alpha_3^c(1-\alpha_3)^{1-c}.$$

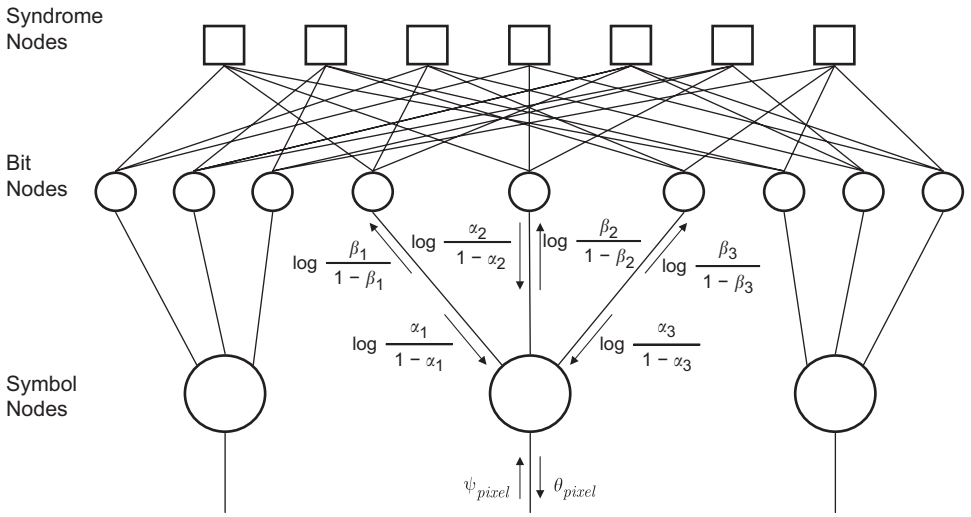

Fig. 2. Belief propagation decoding graph of the joint bitplane LDPC decoder.

3. EM algorithm

3.1. Model

Let X and Y be consecutive luminance frames of video, with X related to Y through a forward motion field M. The residual of X with respect to motion-compensated Y is treated as independent Laplacian noise Z. We model the decoder's a posteriori probability distribution of source X based on parameter θ as

$$P_{\text{app}}\{X\} \equiv P\{X;\theta\} = \prod_{i,j}\theta(i,j,X(i,j)),$$

where $\theta(i,j,w) = P_{\text{app}}\{X(i,j)=w\}$ defines a soft estimate of $X(i,j)$ over luminance values $w \in \{0,\ldots,2^d-1\}$.

3.2. Problem

The decoder aims to calculate the a posteriori probability distribution of the motion M,

$$P_{\text{app}}\{M\} \equiv P\{M|Y,S;\theta\} \propto P\{M\}P\{Y,S|M;\theta\},$$

with the second step by Bayes' Law. The form of this expression suggests an iterative EM solution. The E-step updates the motion field distribution with reference to the source model parameters, while the M-step updates the source model parameters with reference to the motion field distribution. Note that $P\{M|Y,S;\theta\}$ is the probability of observing motion M given that it relates X (as parameterized by θ) to Y, and also given S. We elaborate on $P\{Y,S|M;\theta\}$ below.

3.3. E-step algorithm

The E-step updates the estimated distribution on M and before renormalization is written as

$$P^{(t)}_{\text{app}}\{M\} := P^{(t-1)}_{\text{app}}\{M\}P\{Y,S|M;\theta^{(t-1)}\}.$$

But this operation is expensive due to the large number of possible values of M. We simplify in two ways. First, we ignore the syndrome S since it is exploited in the M-step (LDPC decoding). Second, we permit the estimation of the motion field M with block-by-block motion vectors $M_{u,v}$. For a specified blocksize k, every k-by-k block of $\theta^{(t-1)}$ is compared to the colocated block of Y as well as all those in a fixed motion search range around it. For a block $\theta^{(t-1)}_{u,v}$ with top left pixel located at (u,v), the distribution on the shift $M_{u,v}$ is updated as below and normalized:

$$P^{(t)}_{\text{app}}\{M_{u,v}\} := P^{(t-1)}_{\text{app}}\{M_{u,v}\}P\{Y_{(u,v)+M_{u,v}}|M_{u,v};\theta^{(t-1)}_{u,v}\},$$

where $Y_{(u,v)+M_{u,v}}$ is the k-by-k block of Y with top left pixel at $((u,v)+M_{u,v})$. Note that $P\{Y_{(u,v)+M_{u,v}}|M_{u,v};\theta^{(t-1)}_{u,v}\}$ is the probability of observing $Y_{(u,v)+M_{u,v}}$ given that it was generated through vector $M_{u,v}$ from $X_{u,v}$ as parameterized by $\theta^{(t-1)}_{u,v}$. This procedure, shown in the left of Fig. 3, occurs in the motion estimator.

3.4. M-step algorithm

The M-step updates the soft estimate θ by maximizing the likelihood of Y and syndrome S.

$$\theta^{(t)} := \arg\max_{\Theta} P\{Y,S;\Theta\} = \arg\max_{\Theta}\sum_{m} P^{(t)}_{\text{app}}\{M=m\}P\{Y,S|M=m;\Theta\},$$

where the summation is over all configurations m of the motion field. True maximization is intractable, so we approximate by generating soft side information $\psi^{(t)}$,

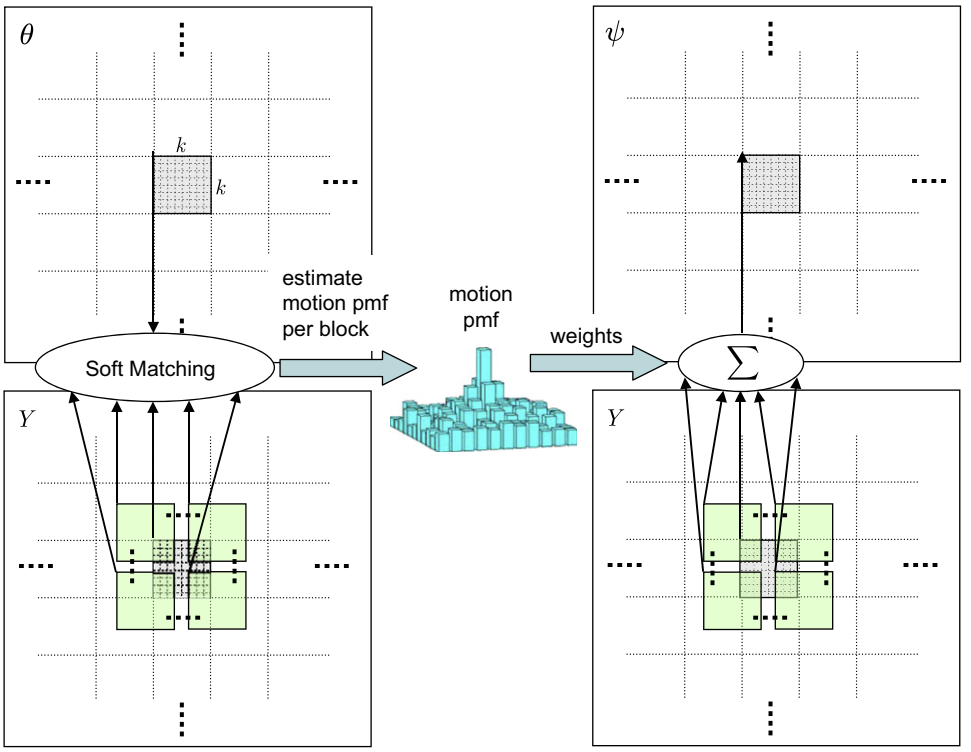

Fig. 3. E-step motion estimator (left) and probability model (right).

followed by an iteration of joint bitplane LDPC decoding to yield $\theta^{(t)}$.

The blockwise a posteriori distribution of the motion $P^{(t)}_{\text{app}}\{M_{u,v}\}$ weights the estimates from each of the blocks $Y_{(u,v)+M_{u,v}}$, which are then summed into soft side information $\psi^{(t)}_{u,v}$, as shown in the probability model step in the right of Fig. 3. More generally, the probability that the blended side information has value w at pixel (i,j) is

$$\begin{aligned}\psi^{(t)}(i,j,w) &= \sum_m P^{(t)}_{\text{app}}\{M=m\}P\{X(i,j)=w|M=m,Y\}\\ &= \sum_m P^{(t)}_{\text{app}}\{M=m\}p_Z(w-Y_m(i,j)),\end{aligned}$$

where $p_Z(z)$ is the probability mass function of the independent additive noise Z, and Y_m is the previous reconstructed frame compensated through motion configuration m.

The joint bitplane LDPC iteration, shown in Fig. 2, begins at the symbols node. Here, the soft side information $\psi^{(t)}$ is combined with log likelihood ratios of $\alpha^{(t-1)}_g$ from the previous iteration to produce log likelihood ratios of $\beta^{(t)}_g$, according to the computation in Section 2.2. The symbol nodes send these ratios to the bit nodes, which forward them to the syndrome nodes. Following the LDPC syndrome decoding rules in [11], the syndrome nodes reply to the bit nodes, which in turn reply to the symbol nodes with the current iteration of log likelihood ratios of $\alpha^{(t)}_g$. Finally, the symbol nodes produce the next soft estimate of the source X, which before normalization over $w \in \{0,\ldots,2^d-1\}$ is computed as

$$\theta^{(t)}(i,j,w) := \psi^{(t)}(i,j,w)\prod_{g=1}^{d}(\alpha^{(t)}_g)^{\mathbf{1}_{[w_g=1]}}(1-\alpha^{(t)}_g)^{\mathbf{1}_{[w_g=0]}},$$

where w_g denotes the gth bit in the Gray mapping of luminance value w and $\mathbf{1}_{[\cdot]}$ denotes the indicator function.

3.5. Termination

Iterating between the E-step and the M-step in this way learns the forward motion vectors at the granularity of k-by-k blocks. The decoding algorithm terminates successfully when the hard estimates $\hat{X}(i,j) = \arg\max_w \theta(i,j,w)$ yield a syndrome equal to S.

4. Transform-domain technique for forward motion vector learning at the decoder

The pixel-domain technique for motion vector learning, in Section 2, exploits temporal redundancy, but not spatial redundancy. Extending the technique into the transform domain enables greater compression. We now change notation: X and Y are unquantized frames, and $\hat{X}$ and $\hat{Y}$ their reconstructions after quantization in the transform domain. The goal is to encode the frame X, in order to reconstruct $\hat{X}$ with side information derived from the previous reconstructed frame $\hat{Y}$.

Fig. 4 shows the transform-domain system. The lossless part (demarcated by a dotted line) is identical to

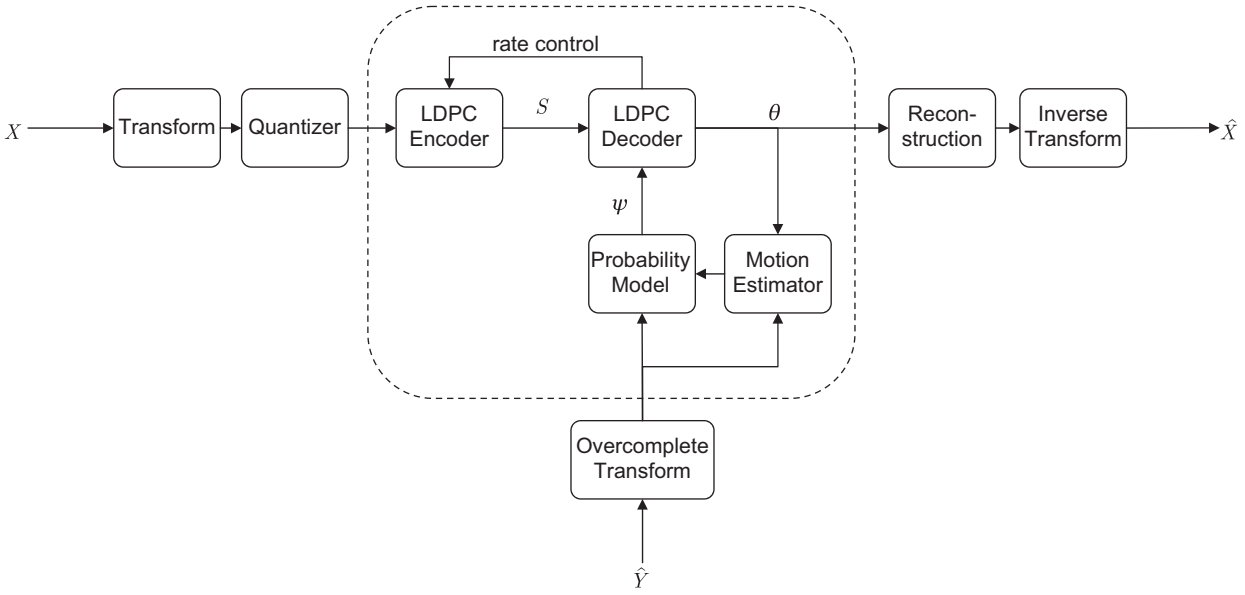

Fig. 4. Transform-domain technique with the lossless system demarcated by the dotted line.

the pixel-domain system of Fig. 1(c), except that it operates on quantized coefficient indices. At the encoder, the frame X is transformed by a k-by-k discrete cosine transform (DCT), where k matches the blocksize of the motion estimator block. The transform coefficients are then quantized into indices before entering the lossless system. At the decoder, the pixels of $\hat{Y}$ are not used directly because the lossless system works in the transform domain. Instead, $\hat{Y}$ is subjected to an overcomplete k-by-k DCT, which computes the transform of all k-by-k blocks at all integer pixel shifts. In this way, the transform coefficients of every motion candidate of $\hat{Y}$ are available in the motion estimator for comparison with the soft estimate θ of the coefficient indices of X, and in the probability model for blending of the side information ψ. Finally, the coefficient indices of X, recovered from the lossless system, are reconstructed to the centers of their quantization bins and inverse transformed into the reconstructed frame $\hat{X}$. We use central reconstruction to avoid introducing blockwise variations in $\hat{X}$, so that its overcomplete transform can be used directly for decoding the next frame. It may be worthwhile to create two reconstructions: a central reconstruction for decoding and a more sophisticated reconstruction for viewing.

An important reason for operating the lossless system entirely in the transform domain is that the soft estimate θ of the coefficient indices of X cannot easily be inverse transformed into a soft estimate in the pixel domain.

5. Simulation results

We build a Wyner–Ziv video codec using the transform-domain technique for forward motion vector learning at the decoder described in Section 4. The codec divides a video sequence into separate groups of pictures (GOPs) with constant GOP size. The first frame of a GOP is coded as a keyframe, decoded without reference to side information. The subsequent frames of a GOP are coded according to Fig. 4 using the previous reconstructed frame as decoder reference. Our experiments use 96 frames of two video sequences, *Foreman* and *Carphone*, at QCIF resolution and at 15 Hz frame rate.

Our codec uses blocksize $k = 8$ for the DCT, the motion estimator and the probability model. For the latter two blocks, the motion search range is ± 10 pixels horizontally and vertically. The EM algorithm at the decoder is initialized with a good value for the variance of the Laplacian noise Z, and experimentally chosen distributions for motion vectors $M_{u,v}$:

$$P^{(0)}_{\text{app}}\{M_{u,v}\} := \begin{cases} \left(\frac{3}{4}\right)^2 & \text{if } M_{u,v} = (0,0), \\ \frac{3}{4}\cdot\frac{1}{80} & \text{if } M_{u,v} = (0,*),(*,0), \\ \left(\frac{1}{80}\right)^2 & \text{otherwise.} \end{cases}$$

After 50 iterations of EM, if the reconstructed $\hat{X}$ still does not satisfy the syndrome condition, the decoder requests additional syndrome bits from the encoder via a feedback channel. This rate control is facilitated by using a regular degree 3 LDPC accumulate code of length 50 688 bits [15] as a platform for joint bitplane LDPC decoding with bit depth $d = 8$. At these settings, exactly 6336 transform coefficients can be Wyner–Ziv coded at a time. Hence, we divide each QCIF-sized Wyner–Ziv frame into four quadrants and code each quadrant separately using the corresponding quadrant of the previous reconstructed frame as decoder reference. Source code for a C++ implementation of this codec is available for download [3].

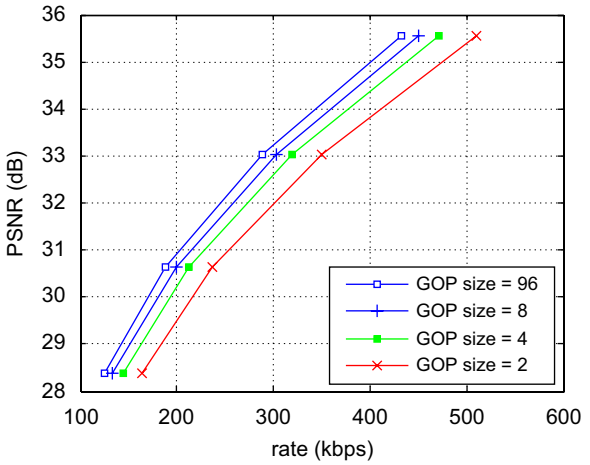

Fig. 5. RD curves for different GOP sizes for *Foreman*.

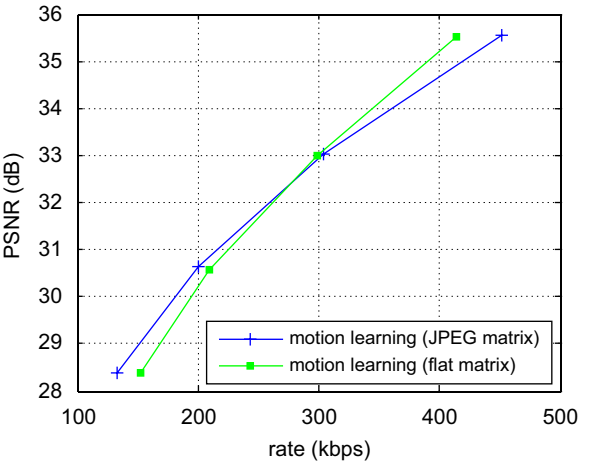

Fig. 7. RD curves for different quantization for *Foreman*.

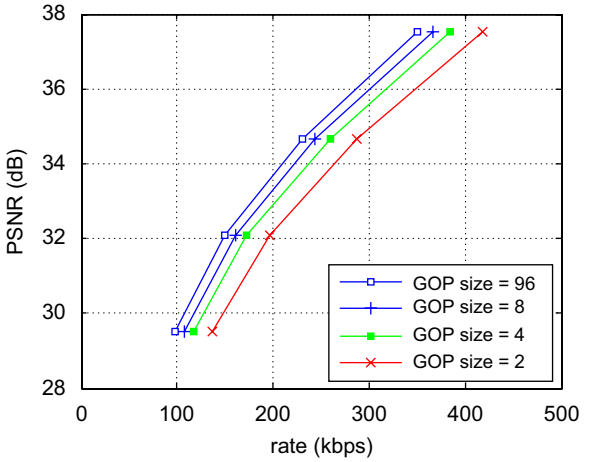

Fig. 6. RD curves for different GOP sizes for *Carphone*.

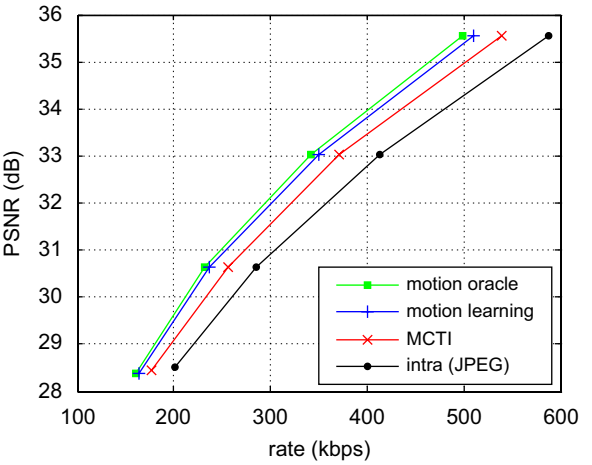

Fig. 8. RD curves for motion learning, MCTI and motion oracle codecs for *Foreman* with GOP size of 2.

5.1. Investigation of motion learning codec parameters

We study the influence of GOP size and quantization strategy on the RD performance of the motion learning codec. Figs. 5 and 6 compare the RD curves for motion learning with varying GOP size, for *Foreman* and *Carphone*, respectively. Here, we constrain the quantization matrix to be a scaled version of the one in Annex K of the JPEG standard [9] with scaling factors $Q = 0.5$, 1, 2 and 4. These results show that RD performance improves as GOP size increases, as would be desired in a video codec. Note that Wyner–Ziv codecs that do motion estimation via MCTI from keyframes generally show the opposite trend [12].

Fig. 7 compares quantization with scaled versions of the JPEG Annex K matrix versus quantization with flat matrices, for *Foreman* with GOP size of 8. That the flat matrices are often inferior in RD performance runs counter to experience with predictive coding. This suggests that overweighting the lower frequency coefficient indices in the motion estimator (as in the JPEG quantizer) can benefit motion learning overall.

In the subsequent results, we fix the GOP size to different values, but vary the quantization matrix as the JPEG Annex K matrix scaled by $Q = 0.5, 1, 2$ and 4.

5.2. GOP size 2 motion learning, MCTI and motion oracle codecs

If the GOP size is 2, keyframes alternate with Wyner–Ziv frames. We compare Wyner–Ziv video codecs that decode the Wyner–Ziv frames in three different ways: with forward motion vector learning, with MCTI between keyframes, and with a motion oracle. We employ the bidirectional MCTI method in [2] with horizontal and vertical search range of ± 20 pixels. The motion oracle knows the forward motion vectors up to horizontal and vertical search range of ± 10 pixels, that minimize the

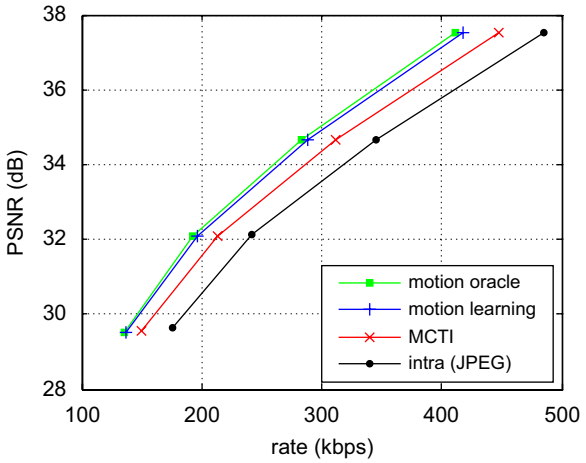

Fig. 9. RD curves for motion learning, MCTI and motion oracle codecs for *Carphone* with GOP size of 2.

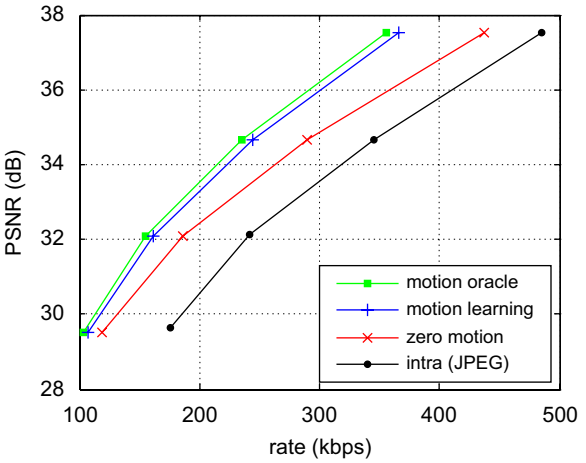

Fig. 11. RD curves for motion learning, zero motion and motion oracle codecs for *Carphone* with GOP size of 8.

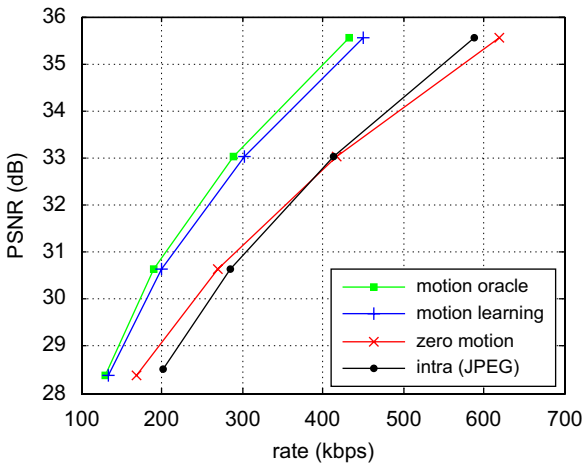

Fig. 10. RD curves for motion learning, zero motion and motion oracle codecs for *Foreman* with GOP size of 8.

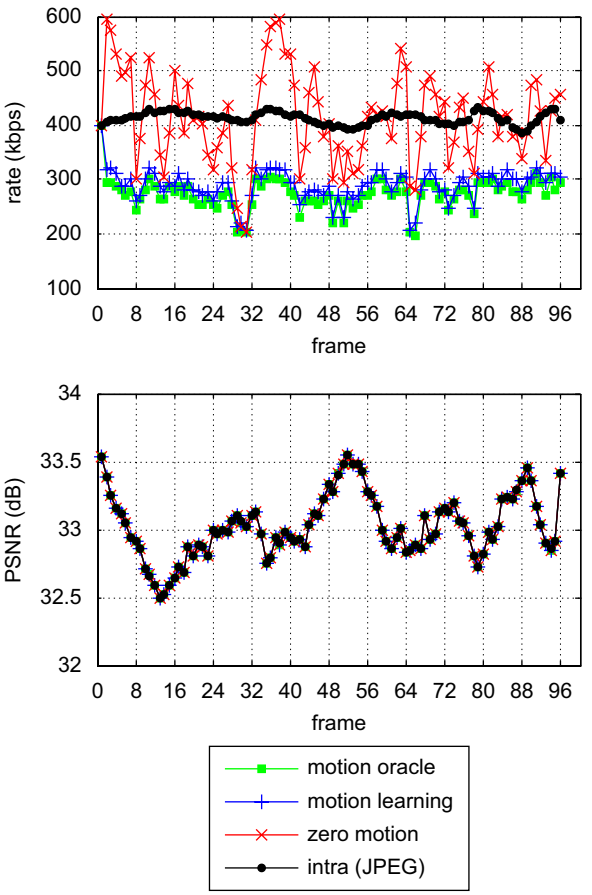

Fig. 12. Rate and PSNR traces for motion learning, zero motion and motion oracle codecs for *Foreman* with $Q = 1$.

blockwise mean square error between X and $\hat{Y}$. Figs. 8 and 9 show the RD curves for the three Wyner–Ziv codecs as well as the one for motion JPEG intra-coding, for *Foreman* and *Carphone*, respectively. At GOP size of 2, motion learning provides about 0.5 dB gain over MCTI, even though it only learns motion vectors in the forward direction, and suffers less than 0.2 dB loss versus motion oracle decoding.

5.3. GOP size 8 motion learning, zero motion and motion oracle codecs

AS GOP size increases, the performance of the motion learning codec improves (as shown in Section 5.1), but the performance of MCTI codecs degrades significantly [12]. So, for GOP size of 8, we replace the MCTI codec with a zero motion codec. Figs. 10 and 11 show the RD curves for motion learning, zero motion and motion oracle Wyner–Ziv codecs and motion JPEG intra coding, for *Foreman* and *Carphone*, respectively. Observe that motion learning can

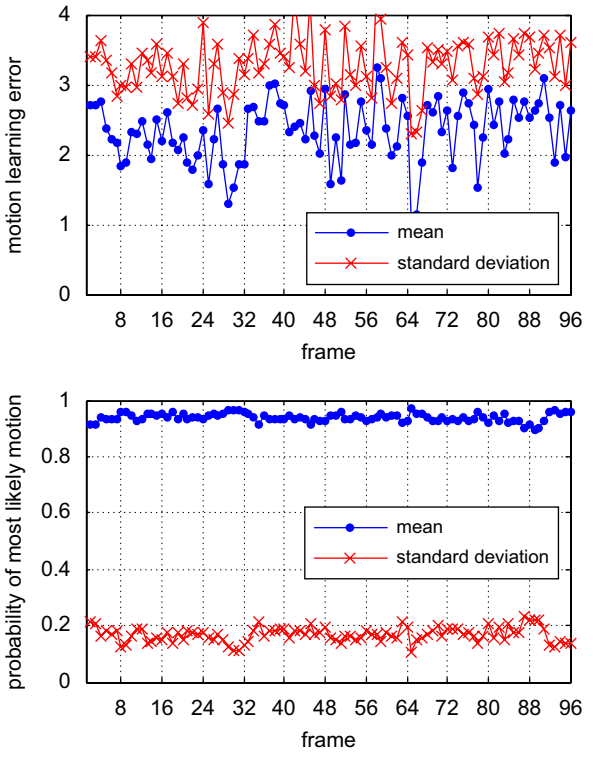

Fig. 13. Statistics of learned motion vectors for *Foreman* with $Q = 1$ and GOP size of 48.

offer up to 2 dB gain over zero motion decoding, while only suffering about 0.25 dB loss compared to motion oracle decoding, for *Foreman*.

5.4. GOP size 96 traces for motion learning, zero motion and motion oracle codecs

Fig. 12 plots traces of the rate and PSNR for the motion learning, zero motion, motion oracle and motion JPEG systems, for *Foreman* with GOP size of 96 and quantization scaling factor $Q = 1$. These traces reflect the fact that all four systems produce identical reconstructed video, but at different rates. In particular, the zero motion codec suffers rate fluctuation in accordance with the amount of motion in each frame. Motion learning reduces the bitrate significantly and makes it less dependent on the motion.

Fig. 13 traces statistics of the learned motion vectors. The motion learning error refers to the Euclidean distances between the learned motion vectors and those known by the motion oracle, and the probability of learned motion vector reflects the confidence in the learned values. Thus, part of the small rate penalty incurred by motion learning with respect to the motion oracle codec is due to minor inaccuracies in learning of the motion vectors.

5.5. Decoding complexity

We now discuss the complexity of Wyner–Ziv decoding, as a function of the number of candidates per motion vector. Consider the cost per EM iteration of decoding a quadrant of a QCIF resolution frame, as in our implementation.

The motion oracle decoder simply knows the best configuration of motion vectors, and need not perform motion search. With respect to number of motion candidates per motion vector, it has fixed cost, F_{oracle}. The cost of the motion learning decoder has both a fixed term and a marginal term (per motion candidate),

$$F_{\text{learning}} + M_{\text{learning}} \times (\#\text{ motion candidates}).$$

Average duration per EM iteration, for the motion oracle decoder and motion learning decoders with different numbers of motion candidates, is tabulated below:

Wyner–Ziv decoder (# motion candidates)	Duration per EM iteration (s)
Motion oracle	0.26
Motion learning (11^2)	0.72
Motion learning (21^2)	1.99

According to these measurements, the fixed costs are roughly consistent:

$$F_{\text{oracle}} = 0.26\,\text{s},$$
$$F_{\text{learning}} = 0.24\,\text{s},$$
$$M_{\text{learning}} = 0.004\,\text{s}.$$

By way of comparison, consider a practical brute-force Wyner–Ziv decoder based on the motion oracle decoder. Instead of knowing the best configuration of motion vectors, it runs decoding with all possible combinations. There are 99 motion vectors per QCIF quadrant when blocksize $k = 8$. With 21^2 candidates per motion vector, the number of configurations is $(21^2)^{99}$. This scaling factor makes the brute-force approach hundreds of orders of magnitude more complex than the motion learning Wyner–Ziv decoder.

6. Conclusions

In this paper, we have introduced a new technique for performing motion estimation at the Wyner–Ziv video decoder. The method applies an EM algorithm for unsupervised learning of motion vectors, and we have described it in both the pixel and transform domains. Our transform-domain Wyner–Ziv video codec demonstrates the favorable property that increasing the GOP size improves overall RD performance. This contrasts with Wyner–Ziv codecs that use MCTI for motion estimation [12]. The motion learning codec is superior to a comparable MCTI codec by 0.5 dB at GOP size of 2, and by several dB at larger GOP sizes. It remains within 0.25 dB of the performance of an impractical motion oracle Wyner–Ziv codec, while its decoding complexity is hundreds of orders of magnitude less than a similar brute-force approach.

References

[1] A. Aaron, S. Rane, B. Girod, Wyner–Ziv video coding with hash-based motion compensation at the receiver, in: Proceedings of the IEEE International Conference on Image Processing, Singapore, 2004.

[2] A. Aaron, R. Zhang, B. Girod, Wyner–Ziv coding of motion video, in: Proceedings of the Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, 2002.

[3] D. Chen, D. Varodayan, Unsupervised learning of motion for distributed video coding, ⟨www.stanford.edu/~dmchen/⟩, 2008.

[4] D. Chen, D. Varodayan, M. Flierl, B. Girod, Distributed stereo image coding with improved disparity and noise estimation, in: Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing, Las Vegas, NV, 2008.

[5] D. Chen, D. Varodayan, M. Flierl, B. Girod, Wyner–Ziv coding of multiview images with unsupervised learning of disparity and Gray code, in: Proceedings of the IEEE International Conference on Image Processing, San Diego, CA, 2008.

[6] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc. Ser. B 39 (1) (1977) 1–38.

[7] B. Girod, A. Aaron, S. Rane, D. Rebollo-Monedero, Distributed video coding, Proc. IEEE 93 (1) (2005) 71–83.

[8] C. Guillemot, F. Pereira, L. Torres, T. Ebrahimi, R. Leonardi, J. Ostermann, Distributed monoview and multiview video coding, IEEE Signal Proc. Mag. 24 (5) (2007) 67–76.

[9] ITU-T, I. JTC1, Digital compression and coding of continuous-tone still images, ISO/IEC 10918-1—ITU-T Recommendation T.81 (JPEG).

[10] F.R. Kschischang, B.J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inf. Theory 47 (2) (2001) 498–519.

[11] A. Liveris, Z. Xiong, C. Georghiades, Compression of binary sources with side information at the decoder using LDPC codes, IEEE Commun. Lett. 6 (10) (2002) 440–442.

[12] F. Pereira, J. Ascenso, C. Brites, Studying the GOP size impact on the performance of a feedback channel-based Wyner–Ziv video codec, in: Proceedings of the IEEE Pacific Rim Symposium on Image and Video Technology, Santiago, Chile, 2007.

[13] R. Puri, K. Ramchandran, PRISM: a new robust video coding architecture based on distributed compression principles, in: Proceedings of the Allerton Conference on Communications, Control and Computing, Allerton, IL, 2002.

[14] D. Slepian, J.K. Wolf, Noiseless coding of correlated information sources, IEEE Trans. Inf. Theory 19 (4) (1973) 471–480.

[15] D. Varodayan, A. Aaron, B. Girod, Rate-adaptive distributed source coding using low-density parity-check codes, in: Proceedings of the Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2005.

[16] D. Varodayan, A. Aaron, B. Girod, Rate-adaptive codes for distributed source coding, EURASIP Signal Process. J. 86 (11) (2006) 3123–3130.

[17] D. Varodayan, Y.-C. Lin, A. Mavlankar, M. Flierl, B. Girod, Wyner–Ziv coding of stereo images with unsupervised learning of disparity, in: Proceedings of the Picture Coding Symposium, Lisbon, Portugal, 2007.

[18] D. Varodayan, A. Mavlankar, M. Flierl, B. Girod, Distributed coding of random dot stereograms with unsupervised learning of disparity, in: Proceedings of the IEEE International Workshop on Multimedia Signal Processing, Victoria, BC, Canada, 2006.

[19] D. Varodayan, A. Mavlankar, M. Flierl, B. Girod, Distributed grayscale stereo image coding with unsupervised learning of disparity, in: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, 2007.

[20] A.D. Wyner, J. Ziv, The rate-distortion function for source coding with side information at the decoder, IEEE Trans. Inf. Theory 22 (1) (1976) 1–10.