The role of the virtual channel in Distributed Source Coding of Video

R.P. Westerlaken, Rene Klein Gunnewiek, R.(Inald) L. Lagendijk

Information and Communication Theory Group,
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, P.O. Box 5031, 2600 GA Delft,
The Netherlands

Philips Research
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

Keywords: Distributed Source Coding, Video Compression, Channel coding

Abstract

In distributed video coding, the complexity of the video encoder is reduced at the cost of a more complex video decoder. Using the principles of Slepian and Wolf, video compression is then carried out using channel coding principles, under the assumption that the video decoder can temporally predict side-information that is correlated with the source video frames. In recent work on distributed video coding the application of turbo codes has been studied. Turbo codes perform well in typical (tele-)communications settings. However, in distributed video coding the dependency channel between source and side-information is inherently non-stationary, for instance due to occluded regions in the video frames. In this paper, we study the modeling of the virtual dependency channel, as well as the consequences of incorrect model assumptions on the turbo decoding process. We observe a strong dependency of the performance of the distributed video decoder on the model of the dependency channel.

1 Introduction

In the past few years, it has become clear that the theories of Wyner-Ziv and Slepian-Wolf have practical potential [1, 2] for carrying out video compression in a distributed way [3, 4]. Distributed video compression allows for shifting a large share of the computational complexity of an encoder to the video decoder while (at least theoretically) maintaining the same rate-distortion performance. As a consequence, the video encoding complexity paradigm can be reversed: the encoder can be made computationally less complex than the decoder. An alternative perspective is that the computational complexity trade-off becomes a design parameter in the implementation of an encoder-decoder pair.

At the basis of distributed video compression is the observation that temporal information the encoder and decoder have available, is essentially identical. From this temporal information, correlated side-information \( Y(i, j, m) \) for the current video frame \( X(i, j, m) \) can be generated with information rate \( H(Y) \). In usual compression terminology, the process of generating side-information is called motion-compensated temporal prediction. In existing compression systems and standards, the encoder in fact needs to generate (motion estimation) and exploit (motion-compensated frame difference) this side-information as so to reduce the bit rate over the channel to \( R_X \geq H(X|Y) = H(X, Y) - H(Y) \). In distributed (Wyner-Ziv) compression the encoder does not actually compute this correlated information, hence reducing the complexity of the encoder since no motion estimation and compensation needs to be carried out. The decoder, however, now needs to generate the side-information by motion estimation and motion-compensated prediction without the assistance of the video encoder, making the video decoder substantially more complex.

Instead of quantizing and entropy encoding a motion-compensated (transformed) frame difference,
the video encoder generates compressed data at rate $R_X \geq H(X|Y)$ in a quite different fashion. The side-information (motion-compensated prediction) $Y(i, j, m)$ available at the decoder is viewed as a by-channel-errors corrupted version of the video frame $X(i, j, m)$ being compressed at the encoder. The dependency between the source video frame $X(i, j, m)$ and the side information $Y(i, j, m)$ computed at the decoder is modeled as a virtual dependency channel. Since $X(i, j, m)$ and $Y(i, j, m)$ are not identical, the virtual dependency channel can be modeled by transition probabilities $P(X|Y)$. For error-free transmission, the data (virtually) transmitted over the dependency channel should be protected by error correcting codes. Indeed, in all proposals for distributed video coders the information bits sent over the channel at rate $R_X \geq H(X|Y)$ are viewed as (the parity bits of) error-correcting codes. For that reason, distributed source coders rely heavily on efficient channel codes. Pradhan [4] and Puri [5] have used Trellis Syndrome Codes to construct their PRISM video coder, while Girod, Aaron et al. [3] have studied the use of turbo codes.

It is important to realize that distributed video compression is conceptually different from the theories of Wyner and Ziv. In the work of Wyner and Ziv, the side information is assumed to be available at the decoder without answering the question how this information got there in the first place. Source and side information are assumed to be dependent via known statistical models of the stationary virtual dependency channel, e.g. the transition probabilities $P(X|Y)$. In distributed video compression, the side information is actually estimated at the decoder. This yields advantages and disadvantages. On the positive side, there are no limitations to the way in which the side information is created. In fact, the process of motion estimation/compensation of a video frame at the decoder can be based on more complex and possibly locally more accurate motion models than at the encoder since no motion information needs to be transmitted from encoder to decoder, in this way avoiding compression overhead. On the negative side, the decoder has to make a prediction of the current video frame without actually knowing this frame, which inherently reduces locally the effectiveness of motion estimation compared to motion estimation at the encoder. As a consequence, the dependencies between the source and side information will be heavily spatially varying, creating a non-stationary virtual dependency channel between $X(i, j, m)$ and $Y(i, j, m)$.

In this paper we study the consequences of the spatially varying success of motion estimation on the processes of channel encoding and decoding in distributed video compression [6]. The channel encoder we use is a powerful turbo encoder. The turbo decoder, however, relies heavily on model information about the (non-stationary) dependency channel. The perspective that we take is that on the one hand the behavior of the virtual dependency channel is substantially more complicated than a simple BSC or AWGN channel model often assumed in communications systems that apply channel codes. On the other hand, since the generation of the side information depends heavily on motion estimation and compensation at the decoder, we are also able to provide the decoder with information about the (local) reliability of side information, in this way assisting the channel decoding process.

In Section 2 we first motivate the problem studied in this paper in more detail. In Section 3 we describe the distributed source coding system we use, based on the work of Aaron et al. [7]. We describe how we can exploit information about the reliability of the motion estimator by introducing reliability classes. In this paper we use two classes, namely occluded regions and non-occluded regions. In Sections 4 and 5 we study the performance of the system under different experimental conditions, first using synthetically generated data, then using real video sequences. In particular we focus the discussion on the modeling of the virtual dependency channel, the resulting (estimated) Slepian-Wolf bounds, and the performance of the turbo decoder. Section 6 gives conclusions on the results presented.

## 2 Problem Formulation

In video compression, the side information $Y(i, j, m)$ at the decoder is computed as the prediction of the video frame $X(i, j, m)$ the encoder is compressing. The side information is estimated using previously decoded video frames $X_{\text{coded}}(i, j, m)$, or:

$$
Y(i, j, m) = g(X_{\text{coded}}(i, j, m - 1), \ldots, X_{\text{coded}}(i, j, m - N)) = X(i, j, m) + N(i, j, m). \tag{1}
$$

Here, $g(\cdot)$ is a function describing the way in which the motion-compensated prediction is created using $N$ previous video frames. In distributed video coding, no physics-inspired analytical models can be obtained for the virtual dependency channel described by $N(i, j, m)$, which is in sharp contrast to telecommunication applications of channel coding which operate over true physical communication channels.

In the earliest work on distributed video compression and in the SEASON framework [8], the deviation of the side information $Y(i, j, m)$ from the
actual video frame $X(i, j, m)$ is modeled as an additive stationary white noise signal $N(i, j, m)$. Prakash et al. [8] state that the residual frame $N(i, j, m)$ will truly appear as white noise if the motion estimation is perfect. Although more sophisticated motion estimation algorithms can be used, we believe that the above model for the dependency channel between $X(i, j, m)$ and $Y(i, j, m)$ is fundamentally flawed because of events like occlusion.

An occluded area is defined as a region which was not visible in the previous video frame(s). Such regions occur by definition when video contains moving objects, which it always does. In occluded regions, motion cannot be estimated properly, and motion-compensated prediction obviously fails. This creates a noise contribution $N(i, j, m)$ that has statistical properties substantially different from the regions in which motion estimation and compensation can be carried out ideally. First, the noise is very location specific; occlusion noise always occurs at the edge of moving objects or the edge of a video frame in case of camera panning. Second, occlusion noise is hard to characterize. The dependencies $P(X|Y)$ in the occluded regions are impossible to estimate since we know that the revealed area is unseen. In fact, in these regions one might claim $P(X|Y) = P(X)$. Since the noise in the occluded and non-occluded areas are both modeled by $N(i, j, m)$, we have to conclude that the noise process in Eq. (1) is inherently non-stationary.

A more proper model for describing the dependency channel between $X(i, j, m)$ and $Y(i, j, m)$ is based on the observation that the noise $N(i, j, m)$ is a spatial mixture of two or more different noise processes. In this paper we restrict ourselves to a mixture of two processes. As an example, consider the original video frame $X(i, j, m)$ and estimated side information $Y(i, j, m)$ of a picture-in-picture sequence in Figures 1(a) and 1(b), respectively. We will use this data also in our experiments in Section 5. The sequence contains a picture-in-picture which moves to the left, while the background rotates in opposite direction. To make the distinction between pixels in the two different regions of interests, in Figure 1(c), blocks of 16x16 pixels are categorized to belong to either the non-occluded region (accurate motion estimation) or the occluded region (inaccurate or poor motion estimation result).

In the region $O$, the noise process $N_o(i, j, m)$ models the effect of occlusions. This process will have a large variance and follows for instance an uniform PDF model. In this region, motion estimation completely fails and the side information is (nearly) uncorrelated with the original data. In the non-occluded regions $O^c$ the noise process $N_{o^c}(i, j, m)$ is assumed to well-behaved, and follow for instance a Gaussian or Laplacian PDF model with a small variance. In case we assume source and side information to be totally independent in the occluded regions, the model for the virtual side channel becomes:

$$Y(i, j, m) = \begin{cases} X(i, j, m) + N_{o^c}(i, j, m), & (i, j) \in O^c; \\ N_o(i, j, m), & (i, j) \in O. \end{cases} \quad (2)$$

In the example we show, there is still some correlation between $X(i, j, m)$ and $Y(i, j, m)$ in occluded regions thanks to the spatial interpolation processes the motion compensator carries out in these regions ($\rho = 0.42$). Hence, an alternative to Eq. (2) would then be:

$$Y(i, j, m) = \begin{cases} X(i, j, m) + N_{o^c}(i, j, m), & (i, j) \in O^c; \\ X(i, j, m) + N_o(i, j, m), & (i, j) \in O. \end{cases} \quad (3)$$

where $N_o(i, j, m)$ follow for instance a Laplacian distribution with a large variance. In Figures 1(d) and 1(e), the histograms of the noise process $Y(i, j, m) - X(i, j, m)$ are shown for the regions $O$ and $O^c$. Two Laplacian PDFs are fitted, with $\sigma^2 = 1$ for region $O^c$ and $\sigma^2 = 510$ for region $O$.

From a channel coding perspective, the lack of
A second perspective is that the decoder knows which samples \((i, j)\) of video frame \(m\) are subject to the occlusion noise \(N_o(i, j, m)\). After all, the decoder itself carries out the motion estimation and compensation, and therefore it can approximately segment the video frame into regions \(O^c\) and \(O\). In fact, motion estimators exist that indicated the reliability of the estimated motion vectors. In a binary decision case, the samples \((i, j)\) with reliability larger than a threshold are classified as region \(O^c\), while the samples \((i, j)\) with reliability smaller than the threshold value are assigned to region \(O\). The channel decoding process – especially if the channel decoder is a soft decoder – can thus be assisted by the motion estimator to pinpoint those samples which are likely to be wrong due to the noise \(N_o(i, j, m)\).

In the following sections we will experimentally investigate the performance of a turbo code-based distributed source coding system. In particular we will consider performance differences when the single model Eqs. (1) is adopted for the virtual dependency channel, versus the case where the mixture models Eq. (2) and (3) are used. We remark that it is far from obvious that the standard turbo encoding procedures that are successfully used in telecommunications applications, are (still) suitable for distributed video compression under the the information that exists because of the rather unpredictable occlusion noise \(N_o(i, j, m)\), can be regarded in two ways. In the first place, in spite of the errors, \(X(i, j, m)\) and \(Y(i, j, m)\) are jointly typical in relation to the estimated probabilities \(P(X|Y)\) if the number of errors is relatively small, or in other words, if the size of region \(O\) is small. Further, spatial interleaving prior to channel coding eliminates the location specificity of the noise \(N_o(i, j, m)\). Hence, if the channel code is strong enough, it will still be able to correctly recover the source data from the side information even if highly unpredictable occlusion noise exists. This perspective is essentially the same as saying that the virtual dependency channel suffers from burst errors, but the occurrence of burst errors is not explicitly described by the channel model.

The Slepian-Wolf encoder is implemented as two identical 16-state convolutional constituent codes with rate 4/5 and with parity polynomials \((23, 35, 31, 37, 27)\), see Figure 3. The complete encoder uses two interleavers, i.e., both convolutional coders are preceded by an interleaver. Interleaving is done on symbol level (quantized PCM video frame samples) rather than on bit level. The reason both encoders are preceded by an interleaver is to break up the region \(O\) in which the occlusion noise occurs. Removing one of the two interleavers substantially degrades the overall performance.

Only the non-systematic bits of the two convolutional coders are transmitted to the decoder, since the systematic part is estimated by the decoder based on the side information \(Y(i, j, m)\). The coding rate 4/5 then translates into a maximum channel rate of 0.5 bit/bit, or – because of the 16-level PCM quantization of \(X(i, j, m)\) – a maximum of 2 bit/sample transmission rate. Clearly the 2 bit/sample is unrealistically high for any practical purpose, mainly because spatially correlation in the video frames is ignored in the above compression system. The objective here is, however, not to get a system to
perform close to the bit rate and quality of existing video compression standards, but to study the error correction capabilities in case the side information is non-stationary. To obtain a specific bit rate $R_X$, the output of the convolutional coders is (randomly) punctured.

![Recursive Convolutional Encoder with parity polynomials (23, 35, 31, 37, 27) in octal form.](image)

Figure 3: Recursive Convolutional Encoder with parity polynomials (23, 35, 31, 37, 27) in octal form.

3.2 Video/Turbo Decoder

At the decoder, side information is generated using an advanced motion estimation and compensation procedure. A prediction of the current video frame is made, based on the temporal information in the past. The decoder consists of two Soft-Input Soft-Output (SISO) maximum likelihood decoders for the symbols/pixels $Q$. In our setup the decoders are serially concatenated and both are preceded by the corresponding interleaver. Extrinsic information is passed between the constituent SISO decoders and the number of bit errors is decreased after every iteration.

The constituent SISO decoders use a maximum a posteriori (MAP) algorithm [9], and provide for each decoded encoder input symbol\footnote{Note that the symbols $Q$ are in fact interleaved quantized pixel values.} $Q_k$ the probabilities that this symbol take one of the possible symbol values ($Q_k \in \{0, \ldots, 15\}$), given the systematic side-information $Y$, serialized into a vector, and the received non-systematic parity sequence $e^{1,2}$.

These type of maximum likelihood estimators, like the MAP-algorithm, assumes that the underlying characteristics of the dependency channel and the a priori probabilities of the input symbols $Q$ are given. The SISO decoding performance depends heavily on this information, and providing incorrect model information may drastically degrade the performance.

First, if the chosen channel model $P(Q|Y_k)$ is an insufficient representation of the actual behavior of the channel, the decoding performance degradation is significant up to the point where the SISO decoding fails to converge. Second, if the initial conditions that are the a priori probabilities of $Q$ – do not match the real statistics of $Q$, an incorrect initialization of the iterative decoding is used, which will also result in a significant loss in decoding performance. Both types of model information should be carefully chosen, which, however, is extremely hard in the practice of distributed video compression because of the non-stationarity of the data and (hence) the limited amount of data on which probability mass functions can be estimated.

4 Experiment Based on Synthetically Generated Data

In this section we study the effect of the size of occluded areas $O$ on the decoding performance for the different noise models described in Section 2, as well as different ways for estimating the required dependency channel model. The decoding performance is analyzed by calculating the bit error rate for different (turbo) encoding rates. The entropy region we are interested in is the region where the rate is higher than the actual conditional entropy $H(Q|Y)$ that has to be transmitted to the decoder. For rates $R_X > H(Q|Y)$ reliable results can be expected in the sense that the BER becomes sufficiently close to zero.

The conditional entropy $H(Q|Y)$ is given by

$$H(Q|Y) = \sum_{Q} \sum_{Y} P(q, y) \log_2 \frac{P(q|y)}{P(q)P(y)}. \quad (4)$$

In the equation we indicate that $H(Q|Y)$ depends in part on the PMF $P(Q,Y)$ or histogram of the available realization of the data, and in part on the model for the virtual dependency channel $P(Y|Q)$ that the SISO decoder is based upon. If the PMF of the realization of $Q$ and $Y$ satisfy the model used in the decoder, the conditional entropy that needs to be sent to the decoder approximates the actual conditional entropy measured between $Q$ and $Y$. If an inaccurate channel model is chosen, the difference between the actual conditional entropy between $Q$ and $Y$ and the entropy that has to be sent to the decoder, computed by averaging $-\log_2 P(Q|Y)$ over all pairs($(Q(i,j,m),Y(i,j,m))$ to get (4), can become consequential.

Following the example in Section 2, we generate a synthetic picture-in-picture frame. As before, two regions are distinguished in the side information, non-occluded and occluded regions each having a different noise model or, equivalently, a different dependency channel. The size of the occluded area is $1/16$ or $1/32$ of the total frame of size $L \approx 40000$. The non-occluded regions are corrupted with additive Laplacian noise with $\sigma^2 = 1$, while the occluded
area is either modeled as noise uniformly distributed between 0 and 255, or is corrupted with additive Laplacian noise with $\sigma^2 = 510$.

The experiments are set up for SISO decoders with different assumptions on the channel model. The PMF $P(Q|Y)$ of the channel model can either be assumed known since we have synthetically generated the noise in regions $O$ and $O^c$, or the PMF can be measured from the real data. Clearly, the latter can only be done in controlled experiments, and never in reality. Although we know that the actual data is non-stationary, we consider the following channel models to characterized the noise $N(i,j,m)$:

- **a Laplacian model.** The channel is assumed to be completely stationary and is modeled by a Laplacian PMF $P(Q - Y)$ with $\sigma^2 = 1$ disregarding occluded regions.
- **a combined model.** The channel is assumed to be completely stationary and the PMFs of non-occluded and occluded regions are combined into a single channel model.
- **a mixed model.** The channel is assumed to be non-stationary. A different PMF is used for occluded and non-occluded regions.

Table 1 shows the conditional entropy $H(Q|Y)$, i.e., Eq. (4), that needs to be (minimally) transmitted to the decoder for a number of possible combinations. Vertically the above-mentioned three models are listed on which the decoder will operate when SISO decoding the turbo codes. Horizontally, we indicate in which way the conditional PMF $P(Q|Y)$ and joint PMF $P(Q,Y)$ are obtained, namely:

- from the channel model used for synthetically generating the data (second column),
- from the statistics of the observed realization of the data at the decoder (third column),
- $P(Q|Y)$ from the channel model used for synthetically generating the data, and $P(Q,Y)$ from the observed realization of the data (fourth column).

When applying turbo decoders in real systems, the last column reflects the situation we have to deal with. For each combination, Table 1 gives the assumed (PMF) model for the dependency channel, and the resulting conditional entropy. We see that in the most realistic practical situation (fourth column), a wrongly selected dependency model dramatically increases the entropy. This will be noticeable in the turbo decoding process as slow or no convergence of the iterations.

Figures 4(a) and 4(b) and show the BER in $\hat{Q}$ at the output of the turbo decoder for a number of rates for the case the size of the uniformly distributed region (according to Eq. (2)) is $1/32$ and $1/16$ of the total frame. At the turbo decoder, the channel is modeled using a synthetic model (Laplacian model, combined model, mixed model) and by measuring the PMF of the real data (combined model, mixed model). This experiment is also done on occluded areas described according to Eq. (3) using the Laplacian model for $N_o(i,j,m)$. The results are shown in Figures 4(c) and 4(d). The results shows that the choice of the channel model is crucial for the performance of turbo decoding. A simple model, like the Laplacian model, degrades the performance dramatically and will lead to an unacceptably high transmission rate $R_X$. For larger bit rates the BER slowly converges to zero. At some point the total number of bit errors becomes close to zero, which results in a somewhat jumpy behaviour on a log scale. The slow convergence is mainly due to the limited amount of data ($L = 40000$) and number of iterations (24). The solid lines represent the expected behaviour for $L \to \infty$ and large number of iterations.

## 5 Experiment Based on Real Data

In this section, we carry out an experiment using the picture-in-picture video sequence introduced in Section 2. A frame of this sequence and its prediction were shown in Figures 1(a) and 1(b), respectively. An example of the binary mask to separate the occluded areas from the non-occluded areas was shown in Figure 1(c). We estimated the PMF $P(Q|Y)$ for the total sequence and also for the occluded and non-occluded part of the sequence. For the soft input soft output, SISO, decoder we consider two cases, namely:

1. A SISO decoder that builds a channel/noise model based on all data (combined model), i.e. ignores the fact that the side-information can be modeled more accurate as a mixture of two stationary processes,
2. A SISO decoder that builds the channel/noise model according to the mask functions of the input (mixed model), i.e. separates side information into two PMF classes (occluded/non-occluded region).

Figure 4(e) shows the performance of the system for the two SISO decoders in terms of their bit error rate (BER) for a number of bit rates. The figure also shows the calculated conditional entropy $H(Q|Y)$, according to column three in Table 1.

The same experiment is repeated, but now the channel is modeled as a single noise channel based
on a combination of 2 Laplacian distributions \( \sigma^2 = 1 \) and \( \sigma^2 = 510 \) for the non-occluded and occluded regions of the subframe, or it is modeled as two separate Laplacian noise channels with \( \sigma^2 = 1 \) and \( \sigma^2 = 510 \) for the non-occluded and occluded regions. The results are shown in Figure 4(f), together with the conditional entropy \( H(Q|Y) \) according to column four in table 1.

These results, together with the results from Section 4, shows that the performance of the SISO decoder, and therefore of the overall system, can be improved greatly by classifying the decoder-generated side-information into two (or more) reliability classes. It also shows that the channel model should be an accurate representation of the real behaviour of the channel, otherwise the decoding performance will heavily degrade.

6 Conclusion

This paper shows that the decoder process is substantially improved by modeling the virtual dependency channel between input and side-information as two separate channels. This success heavily depends on the choice of the channel models. A wrongly chosen model leads to convergence of the decoder only at an unacceptable high transmission rate. The results shows that the performance degradation is mainly due to incorrect channel modeling of the occluded areas. Practical distributed video coders should therefore concentrate on dealing with non-stationarity by using mixture models.

References


Figure 4: Results encoding/decoding subframe with synthetic generated data with Laplacian non-occluded noise ($\sigma^2 = 1$) and uniformly occluded noise for 1/32 part (a) and 1/16 part (b) occluded area, or Laplacian occluded noise ($\sigma^2 = 510$) for 1/32 part (c) and 1/16 part (d) occluded area; and with real data by informing the decoder with an combined or mixture model based on real data (e) and on model assumption (f).