EXTENDED DYNAMIC TEXTURE PREDICTION FOR H.264/AVC INTER CODING

Aleksandar Stojanovic and Philipp Kosse

Institut für Nachrichtentechnik, RWTH Aachen University, Germany
{stojanovic,kosse}@ient.rwth-aachen.de

ABSTRACT

This paper proposes an extension to an existing H.264/AVC encoder that improves the coding performance for sequences containing dynamic textures. Dynamic textures are video sequences with moving texture showing some stationarity properties over time, e.g. water surfaces, whirlwind, clouds, or crowds. By learning the temporal statistics of such content, corresponding areas in future frames can be synthesized. The encoder decides whether to use the synthesized content. The extension to this existing framework is twofold. First, we introduce another synthesis algorithm providing different content. Second, we present an additional macroblock mode which allows a very efficient signaling to the decoder to use the synthesized content. Simulation results show that a bitrate reduction of up to 30% over H.264/AVC at equal PSNR is achieved.

Index Terms— Dynamic textures, H.264/AVC video coding, Inter frame prediction, System identification

1. INTRODUCTION

The main principle of video coding is to exploit the temporal and spatial correlation of natural image sequences for bitrate reduction. Recently, some algorithms have been presented that require *intelligent* decoders. This refers to the fact that the encoder does some extra computation to try to "guess" some of the parameters that until now were signalled to the decoder. In case the guessed information is useful, the encoder instructs the decoder to try to "guess" the missing parameters using exactly the same algorithm. This simple instruction may be much more efficient in terms of bitrate than writing the entire information explicitly into the bitstream. In [1] and [2], techniques are presented where motion vectors are estimated at the decoder side by using template matching.

Another approach to achieve bitrate reduction is to classify sequences into detail-relevant and detail-irrelevant textures as given in [3], or, for still images, to synthesize parts of images using texture synthesis, yielding lower PSNR but a similar subjective visual impression at lower bitrates, as is done in [4].

Our approach is to use synthesis for prediction. This means that in the same way as motion vectors are estimated at the decoder side, we let the decoder estimate an entire reference frame, as described in [5]. This turned out to be sensible for Dynamic Textures, which are video sequences with moving texture showing some stationarity properties over time, e.g. water surfaces, whirlwind, clouds, crowds or even head-and-shoulder sequences.

The rest of the paper is organized as follows. The H.264/AVC inter prediction and the DTP algorithm are presented in Section 2. In Section 3 an alternate dynamic texture synthesis algorithm is introduced, and in Section 4 the new DT skip mode is described. Simulation results are summarized in Section 5 and we give a brief outlook in Section 6.

2. BACKGROUND

2.1. Inter Prediction in H.264/AVC

In video coding data compression relies substantially on using the redundancy between neighboring pictures. For this purpose there are P-slices in H.264/AVC which, among other slice types, allow for the use of temporal redundancy. Namely P pictures are subdivided into macroblocks (MB) of size 16×16 pixels using a regular grid. For each macroblock there are different possibilities, so-called modes, to reference content from previously decoded frames. Roughly speaking, different modes differ in the way MBs are subdivided into smaller partitions. To each marcroblock (or partition) a motion vector and a reference frame is associated, and there is the possibility to code a quantized prediction error signal. Special attention should be given to the P-skip mode, which is very efficient in terms of rate, as all the parameters are computed automatically at the decoder without any further information and no prediction error signal is coded into the bitstream. For a more detailed description of the inter coding of P-slices in H.264/AVC we refer to [6] and a comprehensive description of H.264/AVC can be found in [7].

2.2. Dynamic Texture Prediction for H.264/AVC Inter Coding

In [8] we presented a technique based on the dynamic texture model to improve the coding performance of the inter coding scheme in H.264/AVC for sequences with content considered as dynamic textures. The dynamic texture model, introduced in [9], was initially designed for dynamic texture synthesis, i.e. the generation of endless sequences from short sample sequences. We modified the synthesis algorithm so that on one hand it can deal with the impairments the original algorithm suffers when used in a video coding system. On the other hand one single prediction picture is generated instead of a random video sequence.

The principle of the Dynamic Texture Prediction (DTP) algorithm is to use frames stored in the reference picture buffer for learning and then predict the next picture to be encoded. The main modules of an H.264/AVC encoder are shown in Figure 1. Yellow boxes show the modules generating syntax elements that undergo entropy coding and are written into the bitstream. Clearly, the DTP algorithm is automatic and does not generate any additional information that has to be transmitted, as at the decoder side the identical DTP algorithm can be applied to the reference pictures. A rate-distortion optimization strategy to find an optimal position for the predicted frame in the reference frame buffer is described in [5].

2.3. The Dynamic Texture Prediction Algorithm

A very useful model was introduced in [9] for the purpose of dynamic texture synthesis, i.e. the generation of potentially endless sequences from a sample sequence of the texture. For a given sequence

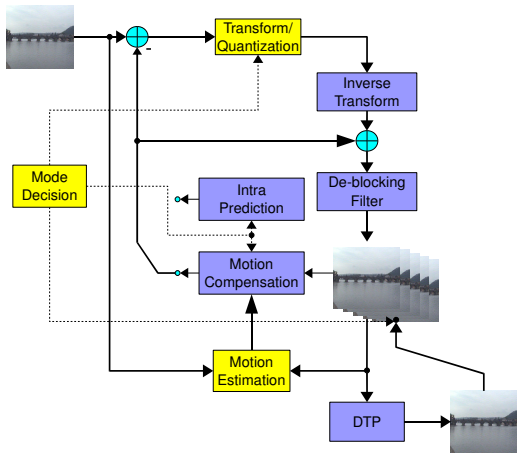

Fig. 1. The principle of the DTP prediction scheme within an H.264/AVC encoder.

$\{y(t)\}_{t=1\ldots\tau}$, $\{y(t)\} \in \mathbb{R}^n$, let $y(t) = I(t) + w(t)$ be a separation of the sequence into noise $w(t) \in \mathbb{R}^n$ and a denoised version of the sequence. Further, let $y_{\mathrm{m}} \in \mathbb{R}^n$ be the temporal mean of the sequence with $y_{\mathrm{d}}(t) = y(t) - y_{\mathrm{m}}$. The sequence of frames $y(t)$ can then be modeled as an ARMA-process (autoregressive moving average):

$$\begin{cases} x(t) = Ax(t-1) + Bv(t) \\ y(t) = Cx(t) + y_{\mathrm{m}} + w(t), \end{cases} \tag{1}$$

with initial condition for the state vector $x(0) = x_0$, an unknown stationary distribution $q(\cdot)$ with $v(t) \overset{i.i.d.}{\sim} q(\cdot)$, and given noise $w(t) \overset{i.i.d.}{\sim} p_w(\cdot)$. Let us further set $y_{\mathrm{d}}(t) = Cx(t)$. Then $x(t)$ is a low-dimensional representation of $y_{\mathrm{d}}(t)$ and hence of $y(t)$. The relation between $x(t)$ and $y(t)$ is given in the second line of equation 1. In the low dimensional space an AR model can be derived for $x(t)$, which is given in the first line of equation 1. Once all the model parameters are known, new frames are generated, which consists in finding new samples $x(t+n)$, by driving the model with random noise $v(t+n)$.

A similar model is used in the dynamic texture prediction algorithm (DTP), but the purpose is to learn the temporal behavior of a sequence from a very short learning sequence and then *predict* one single future frame. Algorithm 1 shows how the algorithm is implemented. We want to state the equations leading to the DTP algorithm and give the motivation behind them.

In the first step the frames in the reference picture buffer are written into a reference buffer matrix Y. It should be noted that the pixels from an entire frame are written consecutively into one single column of the matrix by writing all the columns of one frame into a column and appending the two chroma components. As a result, each column in the sequence matrix is one frame from the reference buffer. Clearly, the dimension of the matrix will be $n \times \tau$, where n is (1.5 times) the number of pixels in a frame and τ is the number of reference frames in the reference picture buffer. The key idea now is to treat each column of the matrix as a separate vector and find a low-dimensional equivalent for each vector.

Let us first state this in matrix form. We can apply a singular value decomposition on Y resulting in:

$$Y = U \cdot S \cdot V^{\mathrm{T}}. \tag{2}$$

By considering $C = U$ as an observation matrix, and $X = S \cdot V^{\mathrm{T}}$ as a state vector matrix, we see that we obtained a matrix X of dimension $\tau \times \tau$, so that each frame is represented by a low-dimensional vector of dimension τ, i.e. the corresponding column from X. Note here that the transform from Y into X, i.e. the dimension reduction step, is not given explicitly, but implicitly using a SVD. The inverse transform however, is given explicitly by the simple multiplication with the matrix C. This is important since we want to predict a future vector X_{pred} and then find its equivalent in the high dimensional space, i.e. the predicted frame. The necessity of this transform will be proved in the next paragraph.

The next step is to treat the vectors from X as state vectors from a linear dynamical system and derive the state transition matrix A, which is in that case:

$$A = arg\min_A \|X^n_{n-\tau+1} - A \cdot X^{n-1}_{n-\tau}\|, \tag{3}$$

where $X^n_{n-\tau+1}$ is the matrix containing the state vectors corresponding to frame number n to $n-\tau+1$ and by analogy $X^{n-1}_{n-\tau}$ the matrix containing the vectors corresponding to $n-1$ to $n-\tau$. The low dimension of the vectors is crucial here, as it guarantees that only one single solution is found for A. Once A is known, we can predict the next vector by multiplying the last vector with A, and find the corresponding predicted frame Y_{pred}. In fact:

$$X_{pred} = AX_\tau, \tag{4}$$

and finally

$$Y_{pred} = CX_{pred}. \tag{5}$$

For more information on the DTP algorithm, we refer to [5] and [8].

Algorithm 1 DTP($Y^n_{n-\tau}$)

Input: The decoded picture buffer matrix $Y^n_{n-\tau}$.
Output: The value of Y_{n+1}.

if $n >= \tau$ **then**
  $U, S, V^T \leftarrow SVD(Y^n_{n-\tau})$
  $C \leftarrow U$
  $X^n_{n-tau} \leftarrow S \cdot V^T$
  $A \leftarrow X^{n-1}_{n-\tau} \cdot \mathbf{pinv}(X^n_{n-(\tau-1)})$
  $X_{n+1} \leftarrow A \cdot X_n$
  $Y_{n+1} \leftarrow C \cdot X_{n+1}$
end if
return Y_{n+1}

3. NEW ALTERNATE DYNAMIC TEXTURE SYNTHESIS ALGORITHM

In addition to the existing DTP algorithm, we introduce an alternate dynamic texture prediction algorithm (DTS). The difference is that it is closer to Doretto's algorithm, which is described in detail in [9]. The main differences between DTP and Doretto's algorithm are stated below:

1. The temporal mean of the sequence is computed and the dynamical model is derived using the difference only.
2. A large number of frames is used for the analysis (e.g. 30) compared to conventional DTP.

3. The order of the model is lower than the rank of the sequence matrix, i.e. the smallest singular values are omitted.

In fact, initially it seemed that the original algorithm was not suitable for the prediction in a video coding system, since a general requirement in video coding is to use only a very limited number of reference frames (e.g. 4 or 5), especially when considering very high resolutions. For that reason we designed a similar algorithm that worked with only few reference frames.

For low resolutions, e.g. QCIF, a separate decoded picture buffer only for the purpose of the synthesis can be used, as the memory requirements are met by any modern PC. The block-based algorithm, which was presented in [5], performs subdivision of the image in smaller parts beforehand and does the prediction on these. In this way we can handle high resolutions.

The precise description of DTS is given in Algorithm 2. It is important to note that $t >> \tau$, i.e. the number of learning frames for DTS is higher. Furthermore $\mathbf{left_k}(A)$ is the operation of taking the left k columns of the matrix A and similarly $\mathbf{upper_k}(A)$ is the operation of taking the upper k rows of the matrix A.

Algorithm 2 DTS(Y^n_{n-t})

Input: The decoded picture buffer matrix Y^n_{n-t}.
Output: The value of Y_{n+1}.

if $n >= t$ **then**
  $Y_{mean} \leftarrow \mathbf{mean}(Y^n_{n-t})$
  $Y_{diff} \leftarrow (Y^n_{n-t}) - Y_{mean}$
  $U, S, V^T \leftarrow SVD(Y_{diff})$
  $C \leftarrow \mathbf{left_k}(U)$
  $X^n_{n-tau} \leftarrow \mathbf{upper_k}(S \cdot V^T)$
  $A \leftarrow X^{n-1}_{n-t} \cdot \mathbf{pinv}(X^n_{n-(t-1)})$
  $X_{n+1} \leftarrow A \cdot X_n$
  $Y_{n+1} \leftarrow C \cdot X_{n+1} + Y_{mean}$
end if
return Y_{n+1}

A drawback of DTS is that, in contrast to DTP, we did not succeed to provide an real-time implementation for the decoder for the sequences under consideration.

4. NEW DT SKIP MODE

The idea of introducing a new skip mode arises from the fact that with the old implementation of DTP from [5], the synthesized content can only be referenced using a mode where the reference frame is written explicitly into the bitstream. However, a more efficient way to code a frame is the skip mode, which tells the decoder to derive the motion vectors by computing the median from neighboring macroblocks, and use the most recent reference frame. In practice, this will mean that the only way to use the synthesis within the skip mode, is to replace the most recent frame in the reference picture buffer by the predicted picture. This is not viable, as it this would strongly deteriorate the compression efficiency.

To circumvent this problem, we introduced a new mode called DT skip. In contrast to the conventional skip mode which signals to the decoder to automatically derive the motion vector and reference frame, and simply copy the content from there, with DT skip, the decoder doesn't need to derive a motion vector but simply copies the content from the same location in the DTP/DTS predicted picture. In terms of signaling, we decided to use an additional flag in the conventional skip mode that signals whether to use the conventional skip mode or DT skip. The DT skip is a further mode that is considered in the RD-decision at macroblock level, just like any other H.264/AVC mode.

5. EXPERIMENTAL RESULTS

The compression performance of the extended encoding system was compared to the performance of the conventional H.264/AVC reference encoder (JM16.1). Various test sequences, all at QCIF resolution, some of which were cropped from higher resolutions were used for the experiments. The presented algorithm is integrated into the JM16.1 reference software [10]. The synthesized frames are computed using the algorithms described in sections 3 and 4. For the experimental evaluation, testing conditions based upon [11] are used. Entire sequences were encoded and other details of the setup are summarized in Table 1.

PARAMETER	VALUE
GOP STRUCTURE	IPPP
QP I	22, 27, 32, 37
QP P	23, 28, 33, 38
SEARCH RANGE	32 PIXELS
FREXT PROFILE	HIGH
RDO	ON
ENTROPY CODING MODE	CABAC
FRAME RATE	30 FRAMES/S
NUMBER REFERENCE FRAMES	5
FULL-PEL DISTORTION METRIC	SAD
HALF-PEL DISTORTION METRIC	HADAMARD SAD
QUARTER-PEL DISTORTION METRIC	HADAMARD SAD

Table 1. JM16.1 Encoder setup.

To be able to use the predicted frames from DTP and DTS simultaneously, a configuration was implemented where in the new skip mode content was copied from the DTP picture, and the DTS picture was used as an additional reference picture. In Figure 3 the compression efficiency of this configuration is compared to the variant where only a predicted DTP picture is used (see [5]) and to the conventional JM16.1 encoder.

An increase in terms of compression efficiency for the combined method can be observed for high rates, especially for the bridge sequence, where the bitrate saving amounts to 30%, in contrast to the old algorithm from [5], where only 15% were achieved. Here DTP provides a very good prediction of the future frame, and the new skip mode is a very efficient way for the encoder to signal to the decoder to use it.

For lower rates, we observed that it is more sensible to turn off the additional skip mode. In fact the amount of detail remaining in the sequence is not sufficient for a useful prediction. Moreover, the conventional skip mode is selected very frequently and considering the new skip mode costs extra rate, turning it off for the low rates provides better results. For this reason we disabled the new skip mode off for $QP = 38$ in the simulation results.

Although the algorithms presented here are inspired from the texture synthesis algorithm from [9], we want to dispel the misconception that our algorithm yields a lower PSNR at equal subjective visual quality, as can be clearly seen in the RD curves. From our point of view, the subjective quality with our encoder is not different from the quality with H.264/AVC at equal PSNR.

Sequence	Source	Predicted Reference Frame		Combined	
		Δ PSNR [dB]	Δ Rate [%]	ΔPSNR [dB]	ΔRate[%]
BRIDGE-CLOSE	ORIG.	0.058	-2.192	0.097	-3.647
CONTAINER	ORIG.	0.385	-7.922	0.438	-8.866
PREAKNESS	ORIG.	0.094	-2.232	0.131	-3.044
RUSHHOUR	CROP.	0.084	-2.590	0.109	-3.003
SEAN	ORIG.	0.095	-1.729	0.102	-1.849
AVERAGE		0.1432	-3.333	0.1754	-4.082

Table 2. BD-PSNR [12] results for JM16.1 compared to JM16.1 with adaptive DTP and compared to the new combined algorithm for selected sequences where the computation of BD-PSNR was possible.

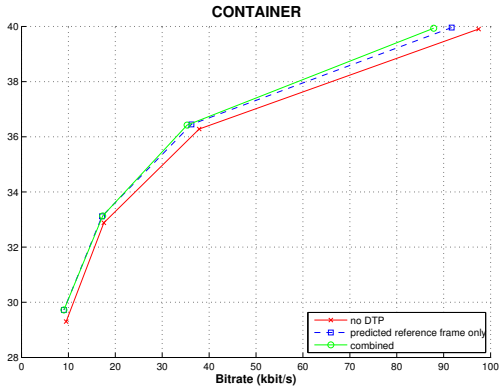

Fig. 2. RD curve for Container qcif

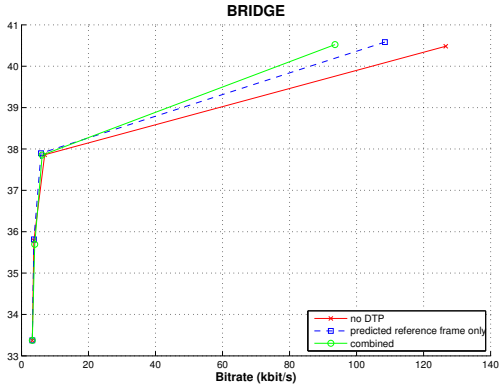

Fig. 3. For the Bridge-far sequence BD-PSNR [12] computation was not possible, but the rate savings for $QP = 23$ amounts to 30% for the combined algorithm.

6. OUTLOOK

Some preliminary results not mentioned in the last section show that most of the additional bitrate savings are rather due to the DT skip mode than to DTS. This means that we could use DTP as an additional reference frame and for DT skip at the same time and still yield similar results. This is important if we want to limit ourselves to frames in the reference picture buffer for the prediction only.

To further validate the results found so far, simulations with higher resolutions have to be done. For this purpose, the block-based algorithm from [5] dividing frames in different parts before the prediction has to be implemented for DTS as well.

Another task is to adapt an already existing motion compensation algorithm to JM16, so that sequences with camera motion can be handled as well. This will enlarge the set of sequences for which the algorithm is providing significant bitrate savings, since many video test sequences contain dynamic texture with camera motion.

7. REFERENCES

[1] Y. Suzuki, C. S. Boon, and T. K. Tan, "Inter frame coding with template matching averaging," in *Proceedings of the IEEE International Conference on Image Processing*, September 2007, vol. III, pp. 409–412.

[2] S. Kamp, M. Evertz, and M. Wien, "Decoder Side Motion Vector Derivation for Inter Frame Video Coding," in *Proceedings of the IEEE International Conference on Image Processing*, October 2008.

[3] P. Ndjiki-Nya, T. Hinz, A. Smolic, and T. Wiegand, "A generic and automatic content-based approach for improved H.264/MPEG4-AVC video coding," in *Proceedings of the IEEE International Conference on Image Processing*, September 2005, vol. II, pp. 874–7.

[4] D. R. Bull S. Ierodiaconou, J. Byrne, D. Redmill, and P. Hill, "Unsupervised Image Compression using Graphcut Texture Synthesis," in *Proceedings of the IEEE International Conference on Image Processing*, November 2009.

[5] A. Stojanovic, M. Wien, and T.K. Tan, "Synthesis-in-the-Loop for Video Texture Coding," in *Proceedings of the IEEE International Conference on Image Processing*, November 2009.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overwiev of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 291–294, July 2003.

[7] "ITU-T Recommendation H.264 - Corrigendum 1: Advanced video coding for generic audovisual services.," Jan. 2009.

[8] A. Stojanovic, M. Wien, and J.-R. Ohm, "Dynamic Texture Synthesis for H.264/AVC Inter Coding," in *Proceedings of the IEEE International Conference on Image Processing*, October 2008, pp. 1608–1612.

[9] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic Textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[10] JM16.1: http://iphome.hhi.de/suehring/tml.

[11] T.K. Tan, G. Sullivan, and T. Wedi, "Recommended simulation common conditions for coding efficiency experiments," in *ITU-T / Study Group 16 / Question 6 - VCEG*. Document VCEG-AH10r3, January 2008.

[12] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," in *ITU-T / Study Group 16 / Question 6 - VCEG*. Document VCEG-M33, April 2001.