SYNTHESIS-IN-THE-LOOP FOR VIDEO TEXTURE CODING

Aleksandar Stojanovic, Mathias Wien

RWTH Aachen University
Aachen, Germany

Thiow Keng Tan

NTT DoCoMo, Inc.
Tokyo, Japan

ABSTRACT

In this paper, we present an algorithm using dynamic texture synthesis for closed-loop video coding. Video textures, or so-called dynamic textures are video sequences with moving texture showing some stationarity properties over time, like water surfaces, whirlwind, clouds, crowds, or even parts of head-and-shoulder scenes. By learning the temporal statistics of such content, we can in principle synthesize the corresponding areas in future frames of the video. In this paper we show that this synthesized image content can also be used for prediction in a closed-loop hybrid video coding system, where the encoder decides about usage of such synthesized content and possible transmission of a residual error signal. This is done in an adaptive and rate-distortion optimized way, such that higher compression performance can be achieved for both high and low bitrates. We show that local adaptation of the algorithm can lead to better compression performance and reduce the computation complexity considerably. If PSNR is used as the quality criterion, savings of up to 15 % in bitrate have been observed experimentally.

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

1. INTRODUCTION

Over the past years, significant advances in the field of compression and synthesis of video textures, i.e. dynamic textures have been made. In [1], an algorithm for dynamic texture synthesis has been described, which is easy to implement and fast in computation. Many variations on this algorithm have been proposed, including dynamic texture modeling from Fourier descriptors [2], or a decomposition using a higher order SVD [3].

A possible application of Doretto's algorithm [1] in the field of video compression was already mentioned by Doretto et al. themselves. They reckoned that for long sequences, the total number of components of the model that are necessary for the re-creation of an approximation of the sequence is less than the total number of pixels in the sequence. Still the number of bits used to represent the different model components was not taken into account, which may be higher than the usual 8 bits for one pixel. Furthermore this compression concept can only be applied to sequences containing dynamic texture exclusively.

Also, texture synthesis algorithms were often used in the field of video and image coding, in order to achieve a better compression ratio. In particular, this was achieved by allowing intra prediction by template matching, like in [4] and [5], or motion estimation using template matching, see [6]. An approach classifying sequences into detail-relevant and detail-irrelevant textures is given in [7].

In this paper we propose an extension to the algorithm presented in [8]. We use an adapted dynamic texture extrapolation algorithm, in order to synthesize dynamic texture that can be used in a conventional coding system for prediction. This dynamic texture extrapolation algorithm is based on the representational dynamic texture model developed by Doretto et al. [1], where tools from system identification are used to learn video texture models.

The rest of this paper is structured as follows. In Section 2, we describe the basic algorithm for dynamic texture prediction. In Section 3, a description is given of how the synthesis was integrated into H.264/AVC in an adaptive way in order to maximize the bitrate savings and PSNR gains on the whole range of sensible rates. Experimental results are reported in Section 4.

2. DYNAMIC TEXTURE PREDICTION

2.1. General dynamic texture model

For a given sequence $\{y(t)\}_{t=1\dots\tau}$, $\{y(t)\} \in \mathbb{R}^m$, let $y(t) = I(t) + w(t)$ be a separation of the sequence into noise $w(t) \in \mathbb{R}^m$ and a denoised version of the sequence. Further, let $y_\mathrm{m} \in \mathbb{R}^m$ 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 $y_\mathrm{d} = Cx(t)$. This representation of a dynamic texture results from the reckoning that individual frames in a sequence consisting of dynamic texture are realizations of the output of a dynamical system driven by an independent and identically distributed (i.i.d.) process.

Using the nomenclature from system theory, $A \in \mathbb{R}^{n\times n}$ is the state transition matrix, $C \in \mathbb{R}^{m\times n}$ the observation matrix, and $x(t)$ the state vector. The system has the particularity that the observations are very high-dimensional, whereas the order of the model, n, is low-dimensional. In [1] this model was used for the purpose of extrapolation of a dynamic texture sequence, i.e. the synthesis of a similar sequence with equal statistics.

In the following, we will present the model described above written in matrix form. For the sequence $y(t)$, we use the convention that each column in the sequence matrix Y is one frame of the sequence. The frames are converted by writing the pixels of the frames consecutively into one vector. For YUV sequences the two chroma components are appended, as described in [1] and [3]. For parts of sequences, the notation $Y_k^l = [y(k), \dots, y(l)] \in \mathbb{R}^{m\times(l-k+1)}$ is used for the part of the sequence containing the frames from k to l

and m is the number of pixels per frame. Eq. 1 becomes

$$\begin{cases} X_1^\tau = AX_0^{\tau-1} + BV_0^{\tau-1} \\ Y_0^{\tau-1} = CX_0^{\tau-1} + Y_\mathrm{m} + W_0^{\tau-1}. \end{cases} \tag{2}$$

where $Y_\mathrm{m} \in \mathbb{R}^{m\times\tau}$ consists of τ equal columns y_m. In applications, the singular value decomposition is used to determine the observation matrix C by computing

$$Y_\mathrm{d} = USD^\mathrm{T}. \tag{3}$$

and setting $C = U$ and $X = SD^\mathrm{T}$. In this case we have $\tau = n$, i.e. the number of singular values is equal to the number of frames in the sequence. If singular values are omitted as in Doretto's approach, we have

$$Y = \tilde{C}\tilde{X} + Y_\mathrm{m} + W. \tag{4}$$

In fact Doretto et al. assume the order of the model n to be smaller than the number of training frames τ, so only n singular values are kept. Further a least squares approximation $\widehat{A}(\tau)$ can be found with:

$$\widehat{A}(\tau) = arg\min_A \|X_1^\tau - AX_0^{\tau-1}\|. \tag{5}$$

These model parameters are then used to synthesize new frames by driving the model with a randomly generated noise process V.

2.2. Model adaptation for prediction

As shown in [1], the model presented above can be used for synthesis, which implicitly means that it captures the statistics of the texture to a certain extent. To use this model for the purpose of prediction in video coding, some modifications have to be made to meet the constraints in a real video coder.

1. **Aiming at pixel fidelity.** Apart from viewing tests, the performance of a video encoder is measured by computing distortion between the original and reconstructed sequence. Hence pixel fidelity is the optimal result. This means that even noise, which is handled separately in Doretto's approach, should be reproduced in the best possible way in terms of pixel fidelity. For our approach this means that we consider the sequence as free of noise, i.e. $W_0^{\tau-1} = 0$ in eq. 2.
2. **Extrapolation from very short learning sequence.** As the number of reference frames in the decoded picture buffer is restricted, only few frames can be used for prediction. As a consequence we will be forced to keep all the singular values after the decomposition. Using Doretto's classical approach with singular value omission would lead into serious deterioration of the prediction image.
3. **Deterministic extrapolation.** Unlike Doretto's algorithm, the synthesis must be performed in a deterministic way, such that the extrapolated image is identical at encoder and decoder. In the model above, the implication is that the term $BV_0^{\tau-1}$ in eq. 2 can be omitted, i.e. no innovation is added.

All these factors taken together, we end up with a state transition matrix A with poles on the unit circle. This causes any extrapolated sequence to be the bare repetition of the original sequence. In the particular case of prediction, it turns out that the extrapolated frame is the first frame from the learning sequence.

To handle this problem, we performed the decomposition on the sequence without the preliminary subtraction of the the temporal mean. In other terms the SVD is done on Y instead of doing it on Y_d.

As an example, we will now show how to obtain the extrapolated frame from the five first frames in a sequence using our algorithm. In particular, we compute:

$$Y_0^4 = CX_0^4 \tag{6}$$

using a SVD. Then

$$\widehat{A} = X_1^4 \left(X_0^3\right)^+ \tag{7}$$

where $\left(X_0^3\right)^+$ is the pseudo-inverse of X_0^3. Finally

$$\begin{cases} X_5^5 = \widehat{A}X_4^4 \\ Y_5^5 = CX_5^5. \end{cases} \tag{8}$$

In this case the extrapolated frame would be the vector Y_5^5.

3. DYNAMIC TEXTURE PREDICTION IN H.264/AVC

3.1. Basic algorithm

In this section we discuss how the prediction is used in a state-of-the-art video coder. The goal is to integrate the extrapolation algorithm into H.264/AVC [9], such that the coding efficiency of H.264/AVC is improved for sequences containing video texture.

The principle is to use frames that are stored in the decoded picture buffer for extrapolation like described in the previous section. In a basic implementation, see [8], the extrapolated frame will replace the oldest frame in the reference picture buffer as shown in figure 1. This is done on the encoder and decoder side, so that we keep the principle that the encoder reconstruction and the decoded sequence are identical. Furthermore the encoder decides on whether the extrapolated frame is referenced, using a conventional rate-distortion optimized motion compensation along with mode decision. That's why we call the overall algorithm synthesis-in-the-loop.

3.2. Adaptive positioning in decoded picture buffer

Experiments have shown that in the basic implementation, see [8], the referencing of the predicted picture is costly because of it is in the last position of the reference picture buffer. The effect is that no PSNR gains or rate savings can be achieved for low bitrates as can be seen from figure 3. The reason why we chose this technique is that it has the advantage of not leading to losses in case of conventional content, as the fifth picture has hardly any influence on the encoding process.

In order to keep the advantage of no losses for sequences with conventional content, and provide gains for low rates simultaneously, we propose to perform multi-pass encoding. The idea is to encode the same picture multiple times, the position of the reference picture being different each time, as can be seen in figure 2.

The best location for the predicted picture is then chosen by a picture-based rate-distortion decision. The used cost function is:

$$c \quad = \quad d + \lambda r, \tag{9}$$

where $\lambda = 0.68 * 2^{(QP-12)/3}$, d is the sum of the distortions of the luma and the two chroma components, and r is the number of bits to encode the picture. The variant with the lowest cost is then finally written into the stream, and the position of the predicted frame is

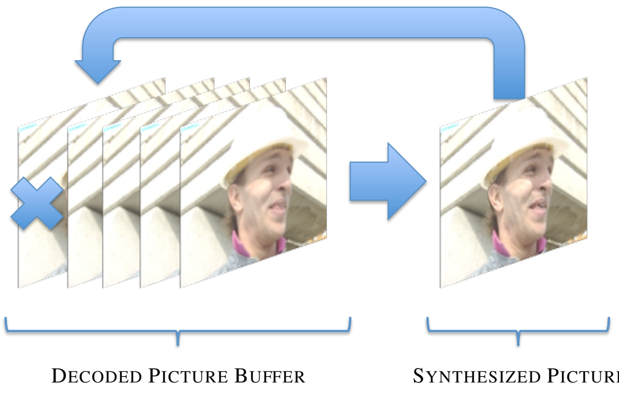

Fig. 1. The synthesized (predicted) picture replaces the oldest picture in the reference picture buffer.

written into the picture parameter set. At the decoder the position is then known and after generation of the predicted picture it is set at the same position in the decoded picture buffer. Results for this method are presented and discussed in Section 4.

3.3. Block-based prediction

In order to make our algorithm applicable to higher resolutions and lower the complexity simultaneously, we propose not make the prediction in a holistic way, but rather divide the picture into different blocks. The whole image is divided into blocks of equal size by dividing it through the same number of parts in the horizontal as in the vertical direction. This makes it possible to implement the prediction algorithm in a parallel fashion, e.g. by using OpenMP, and get a theoretical speed-up of a factor equal to the number of blocks, as the prediction for each block is independent.

Experimental tests show that this subdivision has no significant impact on the performance but for some special sequences where the performance is slightly better.

4. RESULTS

For the experimental investigations, we used the system described in Section 3 with various test sequences, all at QCIF and CIF format, some of which were cropped from higher resolutions. The presented algorithm is integrated into the JM12.4 reference software [10]. The synthesized reference frame is extracted from the five previously decoded frames.

For comparison, testing conditions based upon [11] are used. In particular, we use QP values 22, 27, 32, and 37 for the I frame, and QP values of 23, 28, 33, and 38 for the P frames. The search region is 32 pixels, rate-distortion optimization is on, and the entropy coding mode is CABAC. The prediction structure is IPPP and as five frames are used for the extrapolation, five reference frames are used.

In the basic version of the algorithm described in [8], rate savings were observed for the sequences Bridge and Container for high rates. The new algorithm is outperforming the old version at low and high rates as can be seen from figures 3 and 4.

We have performed multi-pass encoding for a large set of QCIF and CIF sequences in order to determine the performance of the prediction algorithm for any type of content that can be regarded as

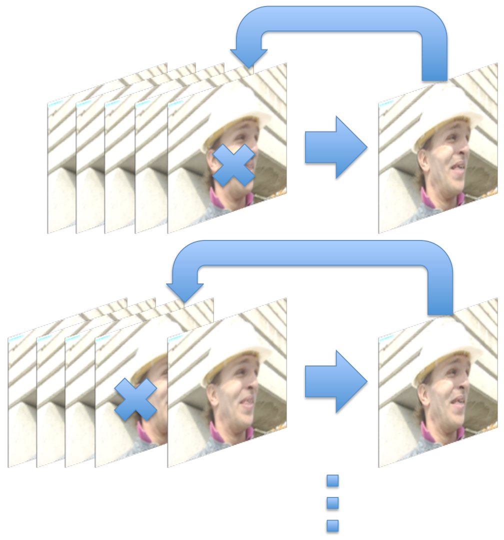

Fig. 2. Adaptive positioning of the synthesized picture in the decoded picture buffer.

Sequence	Source	ΔPSNR[dB]	ΔRate[%]
BRIDGE-CLOSE	ORIG.	0.062	-2.264
CLAIRE	ORIG.	0.105	-1.895
CONTAINER	ORIG.	0.377	-7.987
MISS-AMERICA	ORIG.	0.140	-3.236
MOTHER	ORIG.	0.073	-1.535
PREAKNESS	ORIG.	0.091	-2.188
SEAN	ORIG.	0.099	-1.800
RUSHHOUR	CROP.	0.139	-3.557
SHUTTLESTART	CROP.	0.142	-3.344
Average		0.136	-3.090

Table 1. Bjøntegaard AVSNR results for JM12.4 compared to JM12.4 with adaptive DTP for selected QCIF sequences, where the computation of AVSNR was possible.

dynamic texture or similar to dynamic texture, including head and shoulder sequences.

Table 1 shows AVSNR results for some QCIF sequences. For the Bridge-Far sequence Bjøntegaard AVSNR is not appropriate , so we provided the rate-distortion plot in figure 4.

The Preakness sequence is a sequence with complex motion of the crowd on a very small scale and therefore regarded as dynamic texture. For higher rates, where these details are transmitted with appropriate accuracy, significant bitrate savings were achieved which is reflected by the AVSNR gains in .

Surprisingly, sequences showing mainly faces of talking people, turned out to show some noticeable improvements. Talking faces, due to the complex motion, are only regarded as dynamic texture in a wide sense, but still some bitrate savings can be observed with Sean, Foreman, Claire, and Miss-America, see tables 1 and 2.

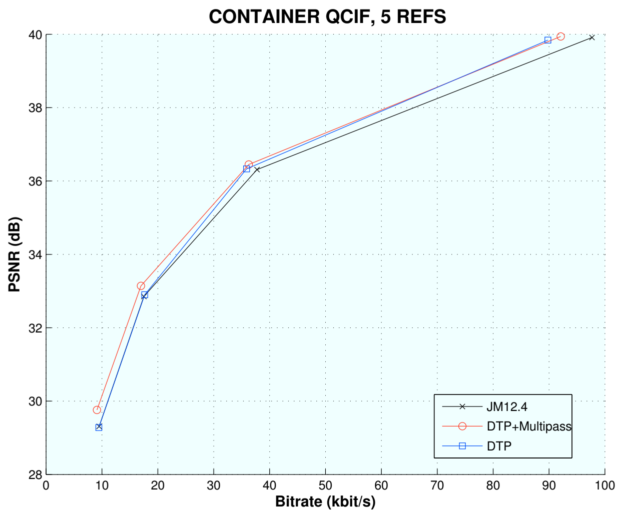

Fig. 3. Typical dynamic texture sequence containing large water surface. Here we see that gains with lower 0.45 dB at Qp = 38.

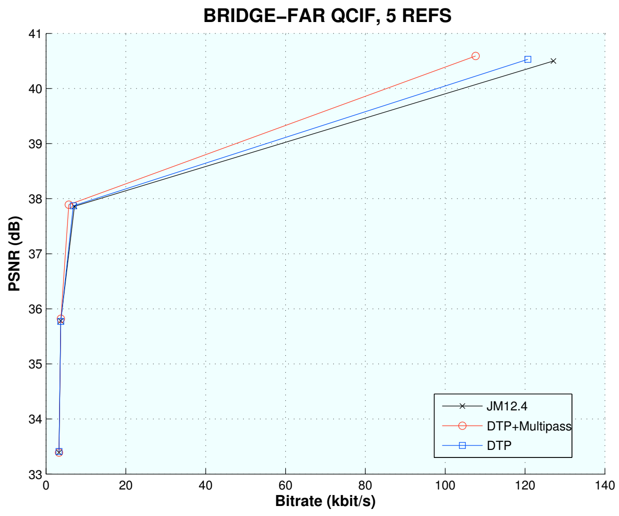

Fig. 4. 15% bitrate saving for multi-pass compared to 6% with conventional DTP for Qp = 23.

Sequence	**Source**	**ΔPSNR [dB]**	**ΔRate[%]**
CONTAINER	ORIG.	0.161	-4.334
SEA	ORIG.	0.093	-1.957
RUSHHOUR 1	CROP.	0.074	-3.174
RUSHHOUR 2	CROP.	0.061	-2.130
SHUTTLESTART	CROP.	0.137	-3.618
Average		0.105	-3.043

Table 2. Bjøntegaard AVSNR results for JM12.4 and compared to JM12.4 with adaptive DTP for selected CIF sequences, where the computation of AVSNR was possible.

5. CONCLUSION AND FUTURE WORK

In this paper we have shown that we are able to predict the future shape of textures to some extent. The predicted picture was saved in the reference picture buffer and hence used in the standard motion compensation mechanism of H.264/AVC. Hereby we generated a synthesis-in-the-loop algorithm improving the coding performance of H.264/AVC.

The development of a new metric that can detect visually equivalent content is a big and currently unsolved challenge. In the case of video texture, the temporal continuity is a special aspect that creates the visual impression of the texture and should not be lost. Therefore this task is particularly challenging in this context. Such a metric constitutes a preliminary for our long-term goal to modify the rate-distortion decision in a fashion that allows the usage of the synthesized frame not only when it is the best match in terms of MSE and thereby outperform our current algorithm.

6. REFERENCES

[1] 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.

[2] B. Abraham, O. Camps, and M. Sznaier, “Dynamic texture with Fourier descriptors,” in *Texture 2005: Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, September 2005, pp. 53–58.

[3] R. Costantini, L. Sbaiz, and S. Süsstrunk, “Higher Order SVD Analysis for Dynamic Texture Synthesis,” *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 42–52, 2008.

[4] T. K. Tan, C.H. Boon, and Y. Suzuki, “Intra Prediction by Template Matching,” in *Proceedings of the IEEE International Conference on Image Processing*, October 2006, pp. 1693–1696.

[5] J. Ballé and M. Wien, “Extended Texture Prediction for H.264/AVC Intra Coding,” in *Proceedings of the IEEE International Conference on Image Processing*, September 2007, vol. VI, pp. 82–92.

[6] 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.

[7] 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.

[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] 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.

[10] JM12.4: http://iphome.hhi.de/suehring/tml.

[11] T. K. Tan et al., “Recommended simulation common conditions for coding efficiency experiments,” in *ITU-T / Study Group 16 / Question 6 - VCEG*. Document VCEG-AA10, October 2005.