# IMPROVED TEMPORAL TEXTURE COMPRESSION USING CAMERA MOTION ESTIMATION

*Aleksandar Stojanovic, Christian Müller and Moritz Knorr*

Institut für Nachrichtentechnik
RWTH Aachen University

## ABSTRACT

In this paper, we introduce a new algorithm for continuous estimation of camera motion and zoom. We show that this method can be used to warp previously encoded frames into the perspective of the frame that is currently encoded. In particular, for so-called video or dynamic texture, that is, sequences exhibiting some kind of temporal periodicity or stationarity, a frame with a given temporal distance can be warped and added into the reference picture buffer to complement conventional reference frames. The rationale is that with these sequences, frames, or parts of frames, tend to reappear in a similar form after some time. We show that by combining a specifically tailored method for frame warping with exploitation of dynamic texture properties, bitrate savings surpassing the state-of-the-art can be achieved with a recent version of the HM software.

***Index Terms***— Video coding, Video texture, Camera motion

## 1. INTRODUCTION

Video compression has witnessed a lot of progress in the past decades and can now be considered as a relatively mature research topic. Nevertheless, recently, a lot of new techniques to further enhance compression efficiency have been presented, some of which have been integrated into HM, the reference software of the HEVC project [1]. An old idea that has never really had a break-through in standardization is to allow motion models going beyond 2D translation, like in [2] where an affine model is used, or a recent method presented at a JCTVC standardization meeting [3], that warps frames from the decoded picture buffer to match parts of the frame currently being encoded. The former method achieves most of its compression improvements from a new interpolation method and the advantage of using more complex motion is rather marginal, while the latter only works for sequences with simple motion patterns, and therefore both were not considered for adoption into the standard. The method we propose in this paper aims specifically at more complex sequences, in particular so-called dynamic or video textures [4, 5]. Previous work [6, 7, 8, 9] has shown that for textures changing over time like water, smoke, leaves in the wind or head-and-shoulder sequences, that is, dynamic textures, special techniques for in-the-loop prediction can improve the compression performance. In this work, we reduce this model to saying that these sequences are somehow cyclic, and focus on the task of camera motion compensation. This paper is structured as follows: in Section 2 we describe the general idea of our work, Section 3 explains the camera motion algorithm and Section 4 describes the encoding scheme for the homographies. The paper concludes with Section 5 which provides experimental results.

This work was supported by the National Research Fund, Luxembourg, under Grant 795405.

## 2. OVERALL SYSTEM

### 2.1. Contributions of this work

In this work we introduce a method that allows the estimation of camera pan and zoom between two successive I-frames by deciding automatically when to chose a new *key* reference frame from where to re-initialize the estimation. These key frames need a good estimation w.r.t. the previous key frame as any error occurring in this frame is carried over until the end. Therefore, a special technique to refine the estimation is introduced. Unlike in previous work, we use Levenberg-Marquardt to minimize a functional depending on pixel values directly, which, to the best of our knowledge, has not been reported previously. This set of new tools provides camera motion and zoom estimation with a precision that renders usage in *closed-loop* video coding possible, especially for sequences where this task is particularly challenging, i.e. dynamic texture. We integrated these tools into the HM software and were able to show that when exploiting the partially cyclical nature of these sequences, the compression ratio can be improved beyond the current state-of-the-art.

### 2.2. Enhanced video encoder

At its core, the method presented in this paper is rather simple. We continuously estimate camera motion and zoom in

a sequence. This information is encoded into the bitstream and will be available at the decoder. Hence we can use this information at the encoder for compression purposes. In practice, we use a long-term reference picture buffer and, with the *estimated* camera motion and zoom information, we warp a frame with a fixed time displacement into the perspective of the frame that is currently encoded. This frame is used as a substitute to the oldest reference frame in the decoded picture buffer and is therefore available for motion compensated coding in the same way as conventional reference frames. The same procedure has to be performed at the decoder, obviously using camera motion parameters from the bitstream.

The rationale for using a long–term buffer is that previous work [8] has shown that for so-called dynamic or video textures, frames or parts of frames tend to reappear in a similar form. In its simplest form, this model would assume a sequence to be cyclic, which explains our procedure. In this work we focus on compensating effects from camera motion and zoom, which are particularly difficult to handle for this type of sequence.

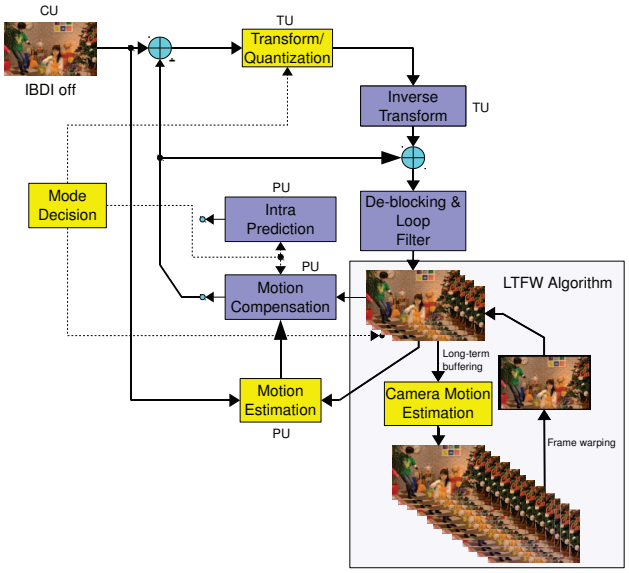


**Fig. 1**. Overall concept of the proposed method.

## 3. CAMERA MOTION AND ZOOM COMPENSATION

### 3.1. Basic principle

We introduced the basic principle of the method for camera motion and zoom compensation in [9]. The technique proved to be necessary for dynamic texture synthesis as the original model from [4] does not take into account effects from camera motion. Here we assumed that the view warping can be compensated by applying a homography to each frame in the sequence. As this assumption turned out to be valid for the case of the considered sequences containing dynamic textures, the remaining problem was to determine the homography corresponding to each frame with maximum accuracy. In a first step, point correspondences between the frames were determined and then used for homography computation. In particular, it turned out that three factors are decisive for an accurate homography estimation, that is, (a) a robust method like Least Median of Squares has to be used for homography computation, (b) points on moving objects have to be excluded and only static background points should be used and finally, (c) point correspondences should be spatially well distributed in order to attain an equally accurate registration over the entire frame.

### 3.2. Continuous estimation

At the encoder, original frames are available so we apply a technique we call continuous estimation to obtain a set of homographies that fully describe relative camera perspectives between the frames. In fact, we begin with computing a homography between frame 0 and the following frames in the sequences as can be seen in Figure 2 (a). For a relatively short period, camera motion, zoom and/or moving foreground object displacement is relatively small, such that the method described in [9] was applicable. However, when entire sequences of 300 frames or more with substantial camera motion and high foreground activity are considered, the static background portion of the frame may become so small that registration becomes impossible. For this reason, our algorithm selects new reference frames, which we termed *key reference frames*, from where warping is re-initialized in an automatic way. Re-initialization with a new key reference is triggered every time one of the following three criteria falls below a predefined threshold:

- **Frame overlap:** The percentage of the frame area visible in both views.
- **Size of Background w.r.t. foreground:** Percentage of static background compared to moving foreground.
- **Number of point correspondences:** Number of correspondences remaining in the set.

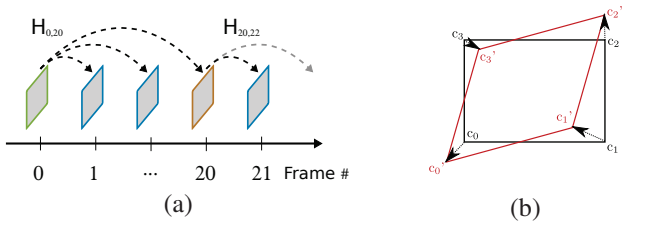


**Fig. 2**. (a) Example with key reference frames at frame 0 and 20. (b) Action of homography on corner points.

### 3.3. Subsequent refinement for key reference frames

Computation of camera motion beyond the limits of key references can be done by simple matrix multiplication of homographies. It goes without saying that any error in these key frames will be carried over to successive views. Hence a good estimation for the latter is imperative. Our method separates static background from foreground and therefore, background in the reference and warped image should be identical in the ideal case. In practice however, slight imprecision in the point correspondence locations will lead to a squared error $S$ that we define by:

$$S(\mathbf{H}) = \left|\mathbf{I_B} - \mathbf{I_B^W}(\mathbf{H})\right|, \tag{1}$$

where $H$ is the homography, $\mathbf{I_B}$ are the pixel intensities of background portions of the reference frame and $\mathbf{I_B^W}$ the intensities of the warped frame. The idea is to slightly change each entry of the homography and see how it affects $S$. The difficulty is that every entry has a different influence on the warped image, and to identify in what range each component can be varied, a preliminary experiment is required, that is, we need to determine the sensitivity of the homography with respect to noise in the point correspondence locations used for its computation. Figure 3 shows histograms for each of the parameters of the normalized homography (in the sense that the last entry is always equal to 1) for the case where we added Gaussian noise to point correspondence locations and computed multiple instances of similar homographies.

Now, to find iteratively a homography with minimal $S$, the Levenberg algorithm is used. While it has been reported in, e.g. [10], that this method can be used to minimize different geometric distances, we successfully used it to minimize pixel value differences. This is mainly due to the good initialization, as we only use the method for refinement. We should mention that the two perspective parameters were too sensitive, so the algorithm was only executed with the 6 upper entries of the matrix. The algorithm consists in finding

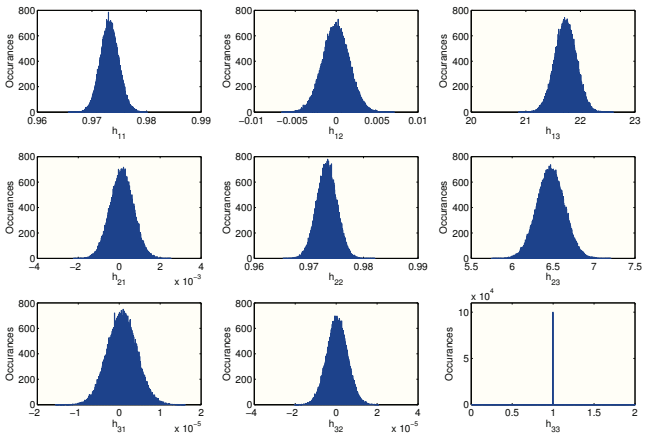


**Fig. 3**. Histograms of elements of homography matrix.

iteratively a $\delta$ that can be added to the elements of the homography. The equation to solve is given by:

$$(J^T J + \lambda I)\delta = -J^T S, \tag{2}$$

where the Jacobian can be computed using:

$$J_i = \frac{S(\mathbf{H} + \Delta h_i\, E_i) - S(\mathbf{H} - \Delta h_i\, E_i)}{2\Delta h_i}. \tag{3}$$

The step size $\Delta h_i = c * \sigma_i$ is different for each entry $h_i$ of the homography $\mathbf{H}$ and derived from the estimated standard deviation of the entries from the above mentioned Monte-Carlo simulation, while $c$ is a constant positive weighting factor.

To limit the computational complexity in this step we use a pyramid based approach, i.e. starting from a down-sampled and low pass filtered version of the images we refine the estimate at each level. The procedure is only performed for key reference frames, so that the overall computational complexity remains limited and our algorithm remains fast compared with the encoding time of the HM software.

## 4. HOMOGRAPHY CODING

Predicting frames in a scenario with camera motion renders the coding of homography parameters necessary. A scheme often used in this context is to code the motion vectors of corner points of the image instead of the homography parameters. A homography is described by a $3 \times 3$ matrix, however it has only 8 degrees of freedom, hence a unique solution can be found by solving a system of 8 equations. The latter can be obtained by filling each of the 4 point correspondences $\mathbf{c_i}$, $\mathbf{c_i'}$ into the equation:

$$\mathbf{c_i'} = \mathbf{H}\mathbf{c_i}. \tag{4}$$

In other words, a homography is fully described by its action on corner points as shown in the illustrative example of Figure 2 (b). From a coding point of view it is equivalent to transmit either the homography matrix or the corner points $\mathbf{c_i'}$, or, more precisely, it is sufficient to transmit the displacement $\mathbf{x_i} = \mathbf{c_i'} - \mathbf{c_i}$. As the former are floating point values and the latter displacement vectors that can be coded in a similar fashion as motion vectors from prediction units, we opt for the latter.

At the encoder, the four displacement vectors $\mathbf{x_i}$ are coded using a predictive coding scheme as shown in Figure 4. The rationale is that the displacement between consecutive frames varies only slightly. In practice, camera motion is usually smooth and zoom is constant over a certain time period, facts that make the usage of the proposed encoder a reasonable choice. For each component $x$ and $y$ of $\mathbf{x} = (x\ y)^T$ and each point $\mathbf{x_i}$, an individual encoder is used. A precision of .1 pixel is used in the practical implementation. The decoder is simply the counterpart to figure 4.

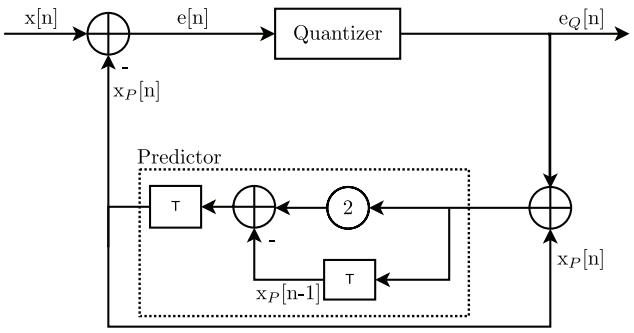


**Fig. 4**. Encoder for transmission of corner displacement encoding.

## 5. RESULTS AND CONCLUSION

For experimental results we used the HM2.2 software with and without our additional tool. The encoder settings are listed in Table 1. The temporal displacement between the current and warped frame was fixed at 30 frames. Rate savings are listed in Table 2. In particular, we achieve bitrate savings for the Sheriff and ThemePark sequences which mainly consist of water surface. Videos of water are considered typical instances of dynamic texture as similar patterns reappear after some time. In PartyScene, chaotically moving and partially periodic elements may be regarded as dynamic texture in the broader sense. The gains for the Waterfall sequence can be explained in part by the dynamic texture in the center and in part by the precise warping of rigid texture (wood) in the scene. A discussion of the latter effect is beyond the scope of this paper. As no gains for this type of sequences were reported in [3], where many homographies in the short-term were allowed, and only a very basic method for homography computation was used, we can conclude that this method complements work done so far on camera motion and provides more gains, however requiring a large frame buffer for the temporal distance. In future, we want to test the method with a broader set of sequences and other encoder configurations, but the latter will require some adaptations in the code.

| PARAMETER | VALUE |
|---|---|
| GOP STRUCTURE | IBBB |
| QP I | 22, 27, 32, 37 |
| QP B | 23, 28, 33, 38 |
| FRAME RATE | 30 FRAMES/S |
| NUMBER REFERENCE FRAMES | 4 |
| SEARCH RANGE | 64 PIXELS |
| MAXIMUM CODING UNIT WIDTH | 64 PIXELS |
| MAXIMUM CODING UNIT HEIGHT | 64 PIXELS |
| MAXIMUM CODING PARTITION DEPTH | 4 |
| MAXIMUM TU TRANSFORM SIZE | $2^5$ |
| MINIMUM TU TRANSFORM SIZE | $2^2$ |
| INTER TU MAXIMUM DEPTH | 2 |
| INTRA TU MAXIMUM DEPTH | 1 |
| ENTROPY CODING MODE | LCEC |
| HIERARCHICAL B CODING | OFF |
| DEBLOCKING LOOP FILTER | ON |
| MERGE MODE | ON |
| ADAPTIVE LOOP FILTER | OFF |

**Table 1**. HM2.2 Encoder setup.

| **Sequence** | **Res.** | **ΔPSNR** | **ΔRate** |
|---|---|---|---|
| WATERFALL | CIF | 1.59 dB | -33.1 % |
| THEMEPARK | 480X360 | 0.11 dB | -3.0 % |
| PARTYSCENE | WQVGA | 0.18 dB | -3.8 % |
| SHERIFF | 360P | 0.22 dB | -7.4 % |
| Average | | 0.525 dB | -11.8 % |

**Table 2**. Bjontegaard Delta results for HM2.2. Sheriff and PartyScene were down-sampled with factor two. ThemePark was cropped and camera motion added.

## 6. REFERENCES


[1] http://hevc.kw.bbc.co.uk/svn/jctvc-a124/.

[2] H. Lakshman, H. Schwarz, and T. Wiegand, "Adaptive motion model selection using a cubic spline based estimation framework," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 805 –808.

[3] http://wftp3.itu.int/av-arch/jctvc-site/2010_10_C_Guangzhou/JCTVC-C033.doc.

[4] 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.

[5] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa, "Video textures," in *SIGGRAPH '00*, New York, NY, USA, 2000, pp. 489–498.

[6] 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.

[7] 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.

[8] Aleksandar Stojanovic and Philipp Kosse, "Extended dynamic texture prediction for H.264/AVC inter coding," in *Proceedings of the IEEE International Conference on Image Processing*, Hong Kong, People's Republic of China, 2010.

[9] J. Ballé, A. Stojanovic, and J.-R. Ohm, "Models for static and dynamic texture synthesis in image and video compression," *Selected Topics in Signal Processing, IEEE Journal of*, vol. PP, no. 99, pp. 1, 2011.

[10] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2003.