EFFICIENT REPRESENTATION AND CODING OF PREDICTION RESIDUALS AND PARAMETERS IN FRAME-BASED ANIMATED MESH COMPRESSION

Detlev Marpe, Heiner Kirchhoffer, Karsten Müller, and Thomas Wiegand

Image Processing Department
Fraunhofer Institute for Telecommunications – Heinrich Hertz Institute
Einsteinufer 37, 10587 Berlin, Germany
[marpe|kirchhof|kmueller|wiegand]@hhi.fraunhofer.de

ABSTRACT

For compression of 3-D dynamic meshes, the novel framework of so-called frame-based animated mesh compression (FAMC) has been introduced recently. In this context, we propose an efficient scheme for representation and statistical coding which is conceptually based on our previous work on context-based adaptive binary arithmetic coding (CABAC). After reviewing the basic principles of both CABAC and FAMC, we present suitable modifications and adaptations of both concepts in order to build an integrated solution with a high degree of coding efficiency. To this end, particular focus of our study has been put on the design of appropriate binarization and context modeling schemes. In our experiments, we obtained average bit-rate savings of more than 30% for a typical test set of dynamic meshes, when comparing the final design of our CABAC enriched FAMC scheme to the original version of FAMC using a conventional *N*-ary arithmetic coder. Our integrated approach has been adopted recently as part of the MPEG-4 Animated Framework eXtension (AFX).

Index Terms— entropy coding, 3-D dynamic mesh compression, CABAC, FAMC, MPEG-4 AFX

1. INTRODUCTION

Multimedia applications aiming at enhancing consumer's sensation towards 3-D, interactivity and virtual reality are increasingly attracting interest, both from a scientific and commercial point of view. Examples for such applications are given by 3-D television, immersive videoconferencing or interactive gaming. In the context of such applications, one important problem to be solved is how to efficiently represent and encode time-varying 3-D content for the purpose of transmission or storage [1].

In the following, we consider an application scenario where dynamic 3-D scenes are recorded by multiple cameras. From this recording, typically a scene representation with *3-D video objects* (3DVOs) is reconstructed, using synthetic geometry and real video texture sequences obtained from each of the given cameras. Synthetic geometry of such 3DVOs is often approximated by planar 3-D meshes for every time instance. The 3-D meshes are further transformed into time-consistent animated mesh sequences. These mesh sequences consist of an initial *intra* or *I mesh* as well as a number of *predictive* or *P meshes*. The I mesh contains the initial 3-D vertex positions as well as the connectivity to define the mesh faces. Each P mesh only contains the new vertex positions and uses the connectivity from the I mesh. For the I mesh, typically a *static mesh coding* approach is applied, whereas a so-called *dynamic mesh coding* scheme is more appropriate for the collection of P meshes (with non-changing connectivity) in order to exploit temporal redundancies in geometric data or attributes like color and normal vectors [1].

Recently, a novel technique for efficient lossy compression of dynamic 3-D mesh sequences, so-called *frame-based animated mesh compression* (FAMC), has been introduced [2][3]. It combines a skinning-based motion compensation strategy with a temporal DCT-based compression scheme. Statistical coding of the individual information parts in FAMC is performed by using an *N*-ary or *multialphabet* arithmetic coder with an *a priori* unknown maximum alphabet size *N*. Multialphabet arithmetic coding, however, is known to be costly, both in terms of computational and modeling costs, in particular in cases where the actual number of different symbols to encode may be considerably smaller than *N*.

Context-based adaptive binary arithmetic coding (CABAC), on the other hand, has proven to be an efficient technique of statistical coding in the area of video coding [4][5]. It handles multiple sources with different alphabet sizes and different statistical properties by application of a three-step process consisting in binarization, context modeling, and binary arithmetic coding. By using a computationally efficient, multiplication-free binary arithmetic coding engine and by tuning the binarization and context modeling schemes to the individual characteristics of the given subsources, a high degree of coding efficiency can be achieved with rather moderate computational costs [4].

In this paper, we describe a novel approach of integrating CABAC into FAMC. The next section contains an overview of the FAMC framework. Sec. 3 provides a brief review of CABAC and Sec. 4 describes a first step of combining CABAC with FAMC. In Sec. 5, we present some further refinements to our initial combined solution. An enhanced encoder control mechanism for FAMC is given in Sec. 6. Sec. 7 finally, contains our experimental results.

2. OVERVIEW OF FRAME-BASED ANIMATED MESH COMPRESSION

Let $M = (M_i)_{i \in \{0,\ldots,F\}}$ denote a sequence of 3-D meshes with an initial I mesh M_0 and F subsequent P meshes M_i ($1 \le i \le F$). Here and in the following, we assume that the first I mesh has already been coded in a lossless way by using a suitable static mesh coder. Frame-based animated mesh compression basically consists of four major building blocks that are illustrated in Fig. 1 along with the initial static mesh coding.

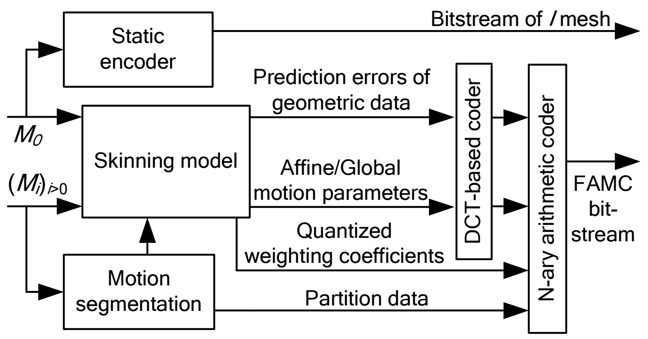

Fig. 1: Illustration of the FAMC encoding strategy.

First, a *motion-based segmentation* is performed by partitioning the mesh vertices into K clusters in such a way that their associated motion can be (sufficiently well) described by a 3-D affine motion model. Once the partitioning is fixed, a *skinning model* is used in the second step for deriving a continuous motion field over the whole mesh (relative to the initial I mesh) by linearly combining the affine motion of adjacent clusters with appropriate weighting coefficients. By using this weighted affine transform, a temporal prediction of the geometrical data (or other attributes) is performed for each vertex of each P mesh M_i. As a result, prediction residuals $\mathbf{e}_i^v = \left(e_i^{v,x}, e_i^{v,y}, e_i^{v,z}, 0\right)^T$ of a coordinate vector $\boldsymbol{\chi}_i^v = \left(x_i^v, y_i^v, z_i^v, 1\right)^T$ at time instant i and for a given vertex with index v are obtained by $\mathbf{e}_i^v = \boldsymbol{\chi}_i^v - \hat{\boldsymbol{\chi}}_i^v$ with $\hat{\boldsymbol{\chi}}_i^v$ denoting the temporally predicted vertex coordinate vector.

In the third step of FAMC, the prediction residual data as well as the affine and global motion parameters are processed by a *DCT-based coding* stage. This includes a 1-D DCT transform applied independently to the three coordinates and motion parameters along the temporal axis. As subsequent processing stages, a uniform quantization is applied, followed by a so-called delta prediction step (which will be discussed later in more detail). In the fourth and final stage of FAMC, all information parts like transform coefficient residuals, predictor side information or partition data are entropy-coded by using a conventional *N-ary arithmetic coder*.

Note that since the DCT-based coding stage is operating along the whole P mesh sequence, a considerable delay may be introduced. However, in order to limit the delay, FAMC permits the subdivision of a given mesh sequence into suitable temporal *segments* of meshes which can be encoded and decoded independently. For more details about FAMC, please refer to [2][3].

3. BRIEF REVIEW OF THE CABAC FRAMEWORK

The CABAC design, as part of the H.264/AVC video coding standard [5], is based on three components: *binarization*, *context modeling*, and *binary arithmetic coding*. Binarization enables efficient binary arithmetic coding by mapping non-binary symbols to sequences of bits which are referred to as *bin strings*. The bins of a bin string can each either be processed in an arithmetic coding mode or in a so-called *bypass* mode. The latter is a simplified coding mode that is chosen for selected bins such as sign information or lesser-significance bins with an assumed fixed probability estimate of 0.5, in order to speed up the overall decoding (and encoding) processes. The regular arithmetic coding mode provides the largest compression benefit, where a bin may be context-modeled and subsequently be arithmetic encoded. As a design decision, in most cases only the most probable bin of a bin string associated with a given symbol is supplied with external context modeling, which is based on previously decoded (encoded) bins. The compression performance of the arithmetic-coded bins is optimized by adaptive estimation of the corresponding (context-conditional) probability distributions. The probability estimation and the actual binary arithmetic coding are conducted using a multiplication-free method that enables efficient implementations in hardware and software. A detailed description of CABAC can be found in [4].

4. A FIRST STEP FOR INTEGRATING CABAC INTO FRAME-BASED ANIMATED MESH COMPRESSION

The skinning-based representation in FAMC consists of four different information parts. Two of them are further processed by the DCT-based coding stage prior to arithmetic coding, as shown in Fig. 1. In the following, we briefly outline how partition data and weighting coefficients are handled in our CABAC-based approach. After that we describe a first step of integrating CABAC into the DCT-based coding stage of FAMC. The status of this first CABAC-based approach roughly corresponds to that in [6].

4.1. Encoding of partition data

The partitioning information as a result of the motion segmentation in FAMC is stored in a vector $\mathbf{p}$ indexed by the vertex indices v, such that for each vertex the corresponding cluster index is given. The CABAC-based approach for encoding $\mathbf{p}$ consists in first transforming $\mathbf{p}$ in a sequence of pairs (*symbol, run*) with *symbol* denoting the cluster index and *run* denoting the number of consecutive occurrences of *symbol* in $\mathbf{p}$ with $\mathbf{p}$ being evaluated in the order of increasing vertex indices. These run-length pairs are then binarized using a fixed-length representation for *symbol* and a concatenated unary / exponential (Exp) Golomb code of order 0 (UEG0) for *run* [4][5]. Encoding of the resulting bin strings is then performed using a single probability model for all bins of each bin string.

4.2. Encoding of weighting coefficients

Weighting coefficients ω_k^v in FAMC are given for each vertex v to indicate the influence of neighboring clusters with index k in the temporal prediction process. The proposed coding method for those coefficients proceeds as follows. First, a binary flag $\eta(v)$ is signaled for each vertex v in such a way that $\eta(v)$ is set to 0 if $\omega_k^v = \delta_{k,k(v)}$ holds, with $k(v)$ being the cluster index to which the vertex v belongs. Then, for each cluster, the number of its neighboring clusters is UEG0 binarized and coded along with the corresponding cluster indices, expressed in a fixed-length binary representation. Finally, the uniformly quantized weights for all vertices with $\eta(v) = 1$ are binarized into bit planes, and the bit planes are encoded from the most significant bit (MSB) to the least significant bit (LSB).

4.3. DCT-based coding

DCT-based coding in FAMC consists of a 1-D DCT applied along the temporal direction, a uniform quantization, and a subsequent sample-wise prediction in the frequency domain, also referred to as *delta prediction*. Suppose the quantized transform coefficients are organized in an array of V rows, each row with F entries of transform coefficients associated with one specific vertex v ($1 \le v \le V$).

The delta predictor is acting on this array in a row-by-row fashion, starting from the top row corresponding to $v = 1$. The predictor is parameterized by a pair of non-negative integers $(\Delta, f-1)$, where $\Delta = v - w$ denotes the offset between the current v^{th} row of quantized transform coefficients to be predicted and the w^{th} row (with $w < v$) of corresponding reference coefficients and where f denotes the number of consecutive samples (beginning from the DC coefficient) that are included in the sample-wise prediction process.

First, the predictor side information $(\Delta, f-1)$ is UEG0 binarized and coded for each row. Then, a binary-valued, so-called *significance map* is coded indicating the location of non-zero (residual) coefficients in each row. After that the magnitudes of non-zero (residual) transform coefficients are binarized into bit planes, and the bit planes are encoded from MSB to LSB. Only one single context model is used for all bins. But due to the order of processing of bins and the backward-adaptation property of probability estimation in CABAC, this is virtually identical to using a separate context model for each bit plane. Finally, the sign information of non-zero coefficients is encoded by using the bypass coding mode of CABAC.

5. IMPROVED REPRESENTATION, BINARIZATION, AND CONTEXT MODELING FOR DCT-BASED CODING

In this section, we discuss some deficiencies of the previously presented coding approach for the DCT-based coding stage in FAMC. Consequently, a more elaborate solution has been developed, which better fits to the typically observed statistical properties of transformed prediction residuals related to geometric data.

5.1. Representation and coding of delta predictor

The main shortcoming of the predictor design, as presented in Sec. 4.3, is given by the fact that it includes an unnecessarily high degree of freedom in the choice of the parameter f. Usually, the number of frames F, which is equal to the dimension of the 1-D DCT, is fairly high (compared to common block sizes in image or video coding). This implies that the wavelengths of basis functions with consecutive frequency indices can be assumed to be rather close to each other and therefore, it is a reasonable assumption to assert that the energy distribution of coefficients related to consecutive frequency indices is also quite similar to each other. This, in turn, is our motivation for introducing a quantization of the f parameter, meaning that we propose to reduce the precision for representation and coding of the f parameter.

More specifically, we allow only integer multiples of 2^b for the choice of $f - 1$, where $b \geq 0$ is a suitably chosen and fixed non-negative integer (for a given encoding/decoding process). Consequently, for coding of f, the b LSBs of the binary representation of $f - 1$ are discarded. In addition to this modification, we propose to adjust the whole coding process for the predictor side information, as shown in Fig. 2 and as described in the following.

Prior to coding of f, a so-called skip flag is coded using a separate context model. It signals whether the current values of p and $f - 1$ are both identical to the previously coded parameter values (of the row above), denoted as `p_prev` and `f_prev - 1`, respectively. If the skip flag is equal to 0, first the value of p is encoded using a concatenated unary/Exp-Golomb binarization. For encoding of $f - 1$, a predictor `(f_pred - 1) >> b` (with ">>" denoting the binary right shift operator) is subtracted from the value of `(f - 1) >> b`, and the magnitude of the resulting difference `dimRes` is binarized and encoded by using the same Exp-

```
if((p == p_prev) && ((p == 0) || (f-1 == f_prev-1)))
  write_skip_flag( 1 )
else
{
  write_skip_flag( 0 )
  unary_exp_golomb( p )
  if( p > 0 )
  {
    dimRes := ( (f-1) >> b ) - ( (f_pred-1) >> b )
    unary_exp_golomb( dimRes )
    if( dimRes != 0 )
    {
      dimPos := f_pred + ( dimRes << b )
      dimNeg := f_pred - ( dimRes << b )
      if( dimPos <= f_max && dimNeg > 0 )
        encodeSign( dimRes )
    }
  }
}
```

Fig. 2: Pseudocode for encoding of predictor side information.

Golomb binarization as for the parameter p. Note that the predictor `f_pred - 1` is the previously *coded* value $f - 1$ (not necessarily of the row above). The sign of `dimRes` is encoded only if both possible values for the sign lead to admissible values of $f - 1$. In other words, if `dimPos` is smaller or equal to the largest possible value `f_max` and `dimNeg` is larger than 0, both values are valid and it has to be signaled which of both cases has been selected at the encoder.

5.2. Coding and context modeling of significance map

For a more efficient encoding of the significance map, we propose a scheme similar to the encoding of the significance map of transform coefficient levels in H.264/AVC. First, a coded_block_flag is signaled which indicates whether all coefficients of a row are zero. If not all coefficients are vanishing, for each transform coefficient, a significant_coeff_flag is encoded which indicates whether the coefficient is non-zero. After each significant_coeff_flag equal to 1, a last_significant_coeff_flag is encoded which signals whether all of the remaining transform coefficient levels of the row are equal to zero. More details about this so-called *sig/last coding* can be found in [4][5].

Note that for each column of the array of $V \times F$ transform coefficients (corresponding to a specific frequency index), one separate context model is used for both coding of significant_coeff_flag and last_significant_coeff_flag. This corresponds to the grouping of flags belonging to the same frequency index into one single statistical model.

5.3. Binarization and coding of absolute values of transform coefficient levels

We propose to encode non-zero (residual) transform coefficient levels by using a concatenated unary/k-th order Exp-Golomb (UEGk) binarization [4][5] instead of using a bit-plane approach. Our empirical analysis turned out that the usage of a fixed length of 16 for the unary prefix part together with the usage of one or two context models for all prefix bins does not lead to a noticeable bit-rate increase compared to the optimal, individually tuned combination of parameters. For the order k of the Exp-Golomb code, however, we found that it is beneficial to select its optimal value in the range of 1 – 8. We also observed that the optimal choice for k is very likely to be equal to 1/2 ×(number of bit planes − 2). Since we allow the usage of either one context model for all bins of the prefix or two context models (one for the first bin and another one for all other bins) for the unary prefix and k in the range of 1 to 8, these values have to be derived by the encoder and signaled in the bit stream. This leads to a signaling overhead of 4 bits.

6. NON-NORMATIVE ENHANCED PREDICTION SELECTION RULE

For the choice of (nearly) optimal values of p and $f - 1$ as used in the delta predictor of the DCT-based coding stage of FAMC, a suitable selection process has to be designed. In the original reference encoder of FAMC, the sum of absolute differences between the row to predict and its predictor candidate (which is defined by p and $f - 1$) was minimized. Obviously, this approach does not properly take into account the real cost in bits for signaling p and $f - 1$. It also neglects the amount of bits necessary for encoding the corresponding transform coefficients.

In order to enhance the selection process for values p and $f - 1$, it is necessary to minimize the bit-rate portion that results from coding p and $f - 1$ plus the bit-rate portion that results from encoding the resulting residual errors. In principle, however, it is hardly possible (due to the lack of computational resources) to find the optimal combination of *all* possible values p and $f - 1$ in such a rate-minimizing sense. Therefore, we restrict the predictor selection process to certain pairs $(p, f - 1)$ only, given the so far encoded values of p and $f - 1$. This does not necessarily result in the optimal choice for *all* possible values of p and $f - 1$. However, as will be shown next, this enhanced selection rule does provide an improved performance relative to the initial FAMC encoder in [3].

7. EXPERIMENTAL RESULTS

Three configurations have been experimentally evaluated relative to the initial FAMC approach using a conventional *N*-ary arithmetic coder [2][3]:

Coder 1: FAMC using the first approach as described in Sec. 4.

Coder 2: FAMC using the improved DCT-based coding approach as described in Sec. 5 (for prediction residuals of geometric data only).

Coder 3: FAMC using the enhanced prediction selection rule as described in Sec. 6, in addition to the configuration of *Coder 2*.

The original FAMC coder and each of the three coder configurations have been tested by using a representative set of mesh sequences that has been adopted as a benchmark test data set during the MPEG standardization of FAMC. Encoding of each of these mesh sequences has been performed with quantization bit-depth values 4, 6, 8, 10, 12, and 14 bits for the prediction residuals and with identical settings for the remaining parameters. Since the decoded mesh sequences for a certain quantization bit depth are *exactly* identical for each of the four configurations (*i.e.*, they produce identical distortions), it is reasonable to only compare the bit rates of the four resulting bit streams. For each mesh sequence and each quantization bit depth, the percentage of bit-rate savings relative to the initial FAMC version using the N-ary arithmetic coder was calculated. Table 1 shows these figures of percentage for each mesh sequence and for each configuration, averaged over all quantization bit depths. In addition, Table 1 shows the overall average bit-rate savings. For the most advanced configuration of *Coder* 3, an average bit-rate reduction of 31% relative to the original FAMC approach has been achieved. This configuration has also been adopted as part of the FAMC-related specification in MPEG-4 AFX [7] both for its normative and non-normative aspects.

Mesh sequence	***Coder* 1**	***Coder* 2**	***Coder* 3**
Troll	23.0%	27.7%	30.6%
Raptor	38.7%	44.2%	46.6%
Camel_collapse	17.6%	22.7%	27.0%
Camel_gallop	31.8%	39.4%	41.6%
Horse_collapse	2.2%	4.4%	9.3%
Horse_gallop	17.4%	25.1%	27.5%
Chicken	17.3%	21.6%	22.1%
Cow	12.2%	13.8%	14.3%
Dance	29.2%	33.9%	35.1%
Dolphin	27.1%	32.2%	34.8%
Humanoid	43.0%	50.2%	52.3%
Snake	36.2%	40.5%	41.0%
Eagle	12.6%	18.8%	20.9%
Average	**23.7%**	**28.8%**	**31.0%**

Table 1: Average bit-rate savings relative to the original FAMC coder using a conventional *N*-ary arithmetic coder.

8. CONCLUSIONS

We have presented an approach for integrating CABAC into the FAMC framework. The context modeling stage of the DCT-based coding stage in FAMC has been adapted to the statistical properties of the prediction residuals. In combination with an enhanced prediction selection rule, average bit-rate savings of more than 30% relative to the initial FAMC approach have been achieved.

REFERENCES

[1] K. Müller, P. Merkle, T. Wiegand: "Compressing Time-Varying Visual Content," *IEEE Signal Processing Magazine,* Vol. 24, No. 6, pp. 58-65, Nov. 2007.

[2] K. Mamou, T. Zaharia, F. Prêteux, "A skinning prediction scheme for dynamic 3D mesh compression," in *Proc. SPIE Conf. Mathematics of Data/Image Pattern Recognigion, Compression, and Encryption with Applications IX*, vol. 6315, pp. 631502, Aug. 2006.

[3] K. Mamou, T. Zaharia, B. Ivanova, M. Preda, F. Prêteux, B. Meaujean, J. Gaillard, O. Marre, "Results of core experiment CE1 on mesh animation compression: skinning-based dynamic mesh compression," ISO/IEC JTC 1/SC 29/WG 11/M14197, Marrakech, Morocco, Jan. 2007.

[4] D. Marpe, H. Schwarz, T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," *IEEE Transactions on Circ. and Sys. for Video Techn.*, Vol. 13, No. 7, July 2003.

[5] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services", ITU-T Recommendation H.264 & ISO/IEC 14496-10 (MPEG-4 AVC), Version 1, May, 2003; Version 2, January 2004; Version 3 (with FRExt), Sept. 2004; Version 4, July 2005.

[6] K. Mamou, D. Marpe, K. Mueller, T. Zaharia, F. Prêteux, "Frame-based Animated Mesh Compression: integration of the CABAC arithmetic encoder," ISO/IEC JTC 1/SC 29/WG 11/M14493, San Jose, USA, April 2007.

[7] Study text of ISO/IEC 14496-16:2006/PDAM2, Frame-Based Animated Mesh Compression, ISO/IEC JTC 1/SC 29/WG 11/W9535, Shenzhen, China, October 2007.