A CONTEXT MODELING ALGORITHM AND ITS APPLICATION IN VIDEO COMPRESSION

Marta Mrak, Detlev Marpe,** and Thomas Wiegand ***

* University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3 / XII, HR-10000 Zagreb, Croatia, marta.mrak@fer.hr
** Fraunhofer Institute for Communications – Heinrich Hertz Institute, Image Processing Department
Einsteinufer 37, D-10587 Berlin, Germany, [marpe,wiegand]@hhi.fhg.de

ABSTRACT

A new algorithm for context modeling of binary sources with application to video compression is presented. Our proposed method is based on a tree rearrangement and tree selection process for an optimized modeling of binary context trees. We demonstrate its use for adaptive context-based coding of selected syntax elements in a video coder. For that purpose we apply our proposed technique to the H.264/AVC standard and evaluate its performance for different sources and different quantization parameters. Experimental results show that by using our proposed algorithm coding gains similar or superior to those obtained with the H.264/AVC CABAC algorithm can be achieved.

1. INTRODUCTION

Context modeling in video compression has become a significant topic of research, being considered a desirable feature for highly efficient video coding. The basic idea behind context modeling in data compression is that the probability of a currently encoded symbol may depend on a number of prior encoded symbols or parts of them, called context parameters. Thus, the challenge is to design an algorithm for selection of an appropriate subset of symbols from the observed data for the estimation of conditional probabilities such that the code length of the data sequence will be minimized. An additional constraint is given by the size of the data sequence that has to be processed, which is typically bounded by the size of a slice of macroblocks of a picture.

H.264/AVC uses *Context-Based Adaptive Binary Arithmetic Coding* (CABAC) that is based on context modeling of at most two neighboring symbols [1]. In the *Growing, Reordering and Selection by Pruning* (GRASP) algorithm presented in this paper, more context parameters are used and the advantages of a context tree model are exploited. Complexity is neglected and a high degree of adaptability is sought. Targeted data on which the proposed GRASP algorithm is tested are binary-valued syntax elements or individual bins of non-binary syntax elements obtained by mapping their values onto prefix-free codes.

Powerful context modeling algorithms based on tree models have not been yet considered for video compression. Inspired by fundamental algorithmic studies, such as in [2] and [3], applications in the area of still image compression were examined in [4] and [5]. The underlying algorithms have shown to be asymptotically optimal in the sense that they achieve minimum adaptive code length for a certain class of finite memory sources when the length of source data sequences approaches infinity.

On short data sequences, however, which are typically observed in video coding applications, great care must be taken to control not only the model order, i.e., the context size, but also the ordering of the context parameters within the context trees in a suitable manner. The problem of ordering has not been given much attention in previous publications [2], [3]. Practical approaches like the CABAC method use some *a priori* gathered knowledge about the typical properties of the underlying source for a proper (fixed) selection and ordering of context parameters. However, in cases where this prior knowledge is not available or a higher degree of agreement between input source and context model is desired, the proposed GRASP method can be applied.

-Our proposed scheme can be used in two different ways. If the best compression performance is targeted regardless of the computational complexity that is required in the encoder, GRASP can be applied in a two-pass coding approach. In this case, prior to the actual encoding pass the relevant statistics of the data sequence, which may be given by a slice, picture or group of pictures (GOP), are gathered in a first pass. Based on the gathered statistics, the GRASP scheme is applied and the resulting context tree models, which have to be signaled as side information, are utilized in the subsequent adaptive entropy coding process. Alternatively, GRASP can also be used as an off-line tool for an optimized design of fixed context models in an adaptive coder. In this application scenario, prior knowledge of the statistics of the targeted source data domain may be incorporated in form of a representative training set. Applying GRASP to this training set yields a set of optimized context models for usage in a one-pass coding scheme.

2. CONTEXT MODELING IN CABAC

Context modeling in CABAC involves pre-defined sets T of past symbols, so-called *context templates*. For each symbol x to encode, in a first step the conditional probability $p(x|T)$ is estimated by switching between different probability models according to the already coded neighboring symbols in T. Then, the estimated probability distribution $p(x|T)$ is used to drive an adaptive binary arithmetic coding engine for the actual encoding of the symbol x. After encoding, the probability model is updated with the value of the encoded symbol x. Thus, $p(x|T)$ is estimated on the fly by tracking the actual source statistics.

However, estimating $p(x|T)$ using past sample statistics may cause the problem of context dilution, if there are no appropriate limits on the symbol alphabet size or on the size of the context templates. In CABAC, this problem is avoided by using very simple context templates consisting of at most two neighbors (i.e. $T = (y_0, y_1)$ in Fig. 1 (left)) and by restricting the symbol alphabet to a binary alphabet. For non-binary valued symbols, CABAC provides appropriately defined binarization schemes,

0-7803-7750-8/03/$17.00 ©2003 IEEE

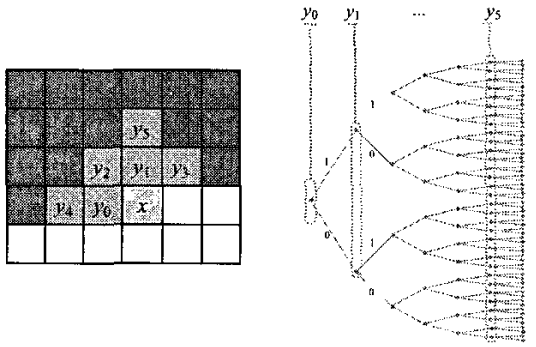

Figure 1: Relationship between a template of context parameters (left) and a full balanced tree for binary data (right).

i.e., the mapping of non-binary valued symbols to prefix-free binary codes. This property of H.264/AVC also simplifies the application of tree-based techniques for context modeling as proposed in the following section.

3. PROPOSED GRASP ALGORITHM

In this section, we describe a method for exploiting given inter-symbol redundancy to a greater extent than it is usually achievable with an ad-hoc design of context models like e.g. that used in CABAC. Our proposed method is based on context trees. The basic motivation behind this approach is given by the observation that the finite memory model for a binary alphabet can be represented as a binary tree structure [2].

For the application of tree-based context modeling on two-dimensional video data, a one-dimensional order needs to be imposed on the neighboring elements. For that, we employ a modeling template, as it is illustrated in Fig. 1 (left), and assign a full balanced tree to that template as shown in Fig. 1 (right). Let y_j^d denote the assignment of a context parameter y_j with $j \in \{0,\ldots,N-1\}$ to a node of the tree at depth d. Let the set $\{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}\}$ denote the sequence of assignments of context parameters to the sequence of nodes over which the node at depth d is reached in the tree. In CABAC, this sequence is fixed to $\{y_0^0, y_1^1\}$, where the canonical order of context parameters in Fig. 1 (right) is used.

A Markov model of order d corresponds to the leaves of a perfectly balanced tree of depth d, as shown in Fig. 1 (right) for $d = 6$. The number of context states in a Markov model grows exponentially with its order. In practice, however, a plain Markov model often contains redundant context states that yield nearly identical conditional probabilities. By using the tree structure, it is possible to remove the nodes associated to these context states, and hence reducing the full balanced tree to some suitable subtree. The GRASP algorithm, which is a two-step approach, is processed for each pre-selected bin of a syntax element separately. In the first step, we grow a full balanced tree of a pre-defined depth as described in Section 3.1. In the second step, we prune the full balanced tree as described in Section 3.2.

3.1. Tree growing by reordering

Tree growing starts in Step 0 with gathering the population of each basic context, which correspond to the leaves of a regular full balanced tree. Any tree reordered or with regular order of context parameters can be reconstructed from these basic contexts. For N context parameters there are 2^N basic contexts. Please note that the assignment of context parameters to basic contexts is manually chosen. The populated initial full balanced tree is then grown using the following Steps 1 to 3:

Step 0: Based on an initial order for the context parameters of a given context template $T = (y_0,\ldots,y_{N-1})$, construct the basic contexts. Populate the corresponding nodes by gathering the given source statistics such that each basic context contains the occurrence counts of 0's and 1's.

Step 1: Start constructing a new, reordered tree by associating the empty subsequence with the root node at depth 0.

Step 2: Determine the assignment of a context parameter y_j to the current node at depth $d < N$. Let the subsequence $\{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}\}$ be the sequence of assignments to reach the current node at depth d from the root node. The assignment $y_{j_d}^d$ has to be determined via

$$j_d = \underset{j \in \{0,\ldots,N-1\} \setminus \{j_0,\ldots,j_{d-1}\}}{\arg\min} L_j$$

with L_j denoting the *adaptive code length* to encode symbols x from nodes at level $d+1$ given as

$$L_j = \sum_{i=0}^{1} \ell(c_0^{z_{i,j}}, c_1^{z_{i,j}}, \kappa_0^{z_{i,j}}, \kappa_1^{z_{i,j}}), \text{ where}$$

$$\ell(c_0, c_1, \kappa_0, \kappa_1) = -\log_2\left(\frac{(c_0-1)!\,(c_1-1)!}{(c_0+c_1-1)!}\,\frac{(\kappa_0+\kappa_1-1)!}{(\kappa_0-1)!\,(\kappa_1-1)!}\right),$$

$z_{i,j} = \{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}, y_j^d = i\}$ is the subsequence of assignments to reach the node at depth $d+1$, and $c_0^{z_{i,j}}, c_1^{z_{i,j}}$ are the corresponding occurrence counts of 0's and 1's at that node and $\kappa_0^{z_{i,j}}, \kappa_1^{z_{i,j}}$ are the corresponding initial counts of 0's and 1's at that node, respectively. This adaptive code length is given for the following probability estimator: $p(x=i) = c_i / (c_0+c_1)$. The occurrence counts c_0 and c_1 are initialised with κ_0 and κ_1, respectively.

Step 3: Repeat Step 2 recursively until the maximum depth N is reached.

This so-called *tree growing by reordering* process of our proposed algorithm recursively builds a tree from root to leaves using information from the populated basic contexts. It starts with the root node and tests for each node, which of the context parameters is the best for splitting this node ignoring those context parameters that have been chosen on a path from the root to the actual node. Since an adaptive code is used, the criterion, on which the context parameters are compared, is based on the sum of adaptive code lengths for the child nodes. At each node, the context parameter that minimizes the sum of adaptive code lengths related to the corresponding conditional probabilities is chosen for splitting. The goal of this stage is to concentrate those context parameters near the root node that disperse the statistics most and simultaneously to shift less important context parameters towards the deepest tree levels. In this way, the subsequent pruning process can operate more efficiently.

3.2. Tree selection by pruning

Since the structure of the chosen tree model has to be transmitted and the gain in information per context parameter in one branch of a context tree cannot always compensate the increased side information when the number of nodes grows, an appropriate method for selecting the best subset of nodes is used. Thus, in order to reduce the dimensionality of the context tree, a *tree selection by pruning* is carried out to choose the best performing subtree as follows:

Step 0: To each node s of the full balanced tree, as constructed in the tree growing by reordering stage, assign the code length $\ell_s = \ell(c_0^s, c_1^s, \kappa_0^s, \kappa_1^s)$, where $c_0^s, c_1^s, \kappa_0^s, \kappa_1^s$ are the occurrence and initial counts at that node. Set the cost functional J of each terminal node to its code length: $J(s) = \ell_s$. Note that evaluation of nodes starts with the terminal nodes of the reordered full balanced tree, i.e., at depth $N-1$.

Step 1: Compare the code length of the current node s with the sum of the cost functional J evaluated at the two child nodes s_{ch0} and s_{ch1}. If the sum of the costs of the children is greater than or equal to the adaptive code length at the parent node, prune the branch below the parent node; otherwise, assign the sum of the children's cost plus a model cost term mc to the parent node, i.e.,

$$J(s) = \begin{cases} \ell_s + mc(d), & \text{if } \ell_s \le J(s_{ch0}) + J(s_{ch1}) \\ J(s_{ch0}) + J(s_{ch1}) + mc(d), & \text{otherwise} \end{cases}, \quad (1)$$

where the model cost mc depends on the actual depth d at node s and the depth N of the full tree such that $mc(d) = \lceil \log_2(N-d+1) \rceil$. mc represents the number of bits that are needed to signal the context parameter indices together with a terminal node flag.

Step 2: Repeat Step 1 recursively until the root is reached.

The tree selection process starts from the leaves of the full grown and reordered tree, which was obtained as a result of the first stage of the GRASP algorithm. Each node s of that tree is visited in a bottom-up strategy by evaluating a recursively defined cost functional J. The functional J rates both the average adaptive code length and the model description cost mc.

As a result of the selection process, in the final tree all branches are pruned that result in a higher cost than that of their corresponding root nodes. As an example, Fig. 2 shows the tree obtained by applying the proposed GRASP algorithm to the significant coefficient flag of transform coefficients in H.264/AVC. Context parameters are related to neighboring blocks (indices 0–5) and to the current block (indices 6–9).

3.3. GRASP for group of pictures

As described in the previous sections application of the basic GRASP algorithm results in the design of locally optimal trees for a single coding unit. Usually, frames of the same type (P or B) within one GOP have similar statistical properties, which may result in similar context tree models and further in bit-rate reduction, since the models do not have to be transmitted for each frame separately. Thus, we extend the GRASP algorithm to design a single optimal tree for a given syntax element (or parts of it) of each slice type (I, P or B) within a GOP, which we refer to as a *GOP-based optimal tree*. In addition, for achieving a faster adaptation during encoding, we propose to feed the parameters of a GOP-based optimal tree structure, i.e., the probability models of each node s chosen for coding with appropriate initial counts κ_0^s and κ_1^s, which are represented as an initial probability state.

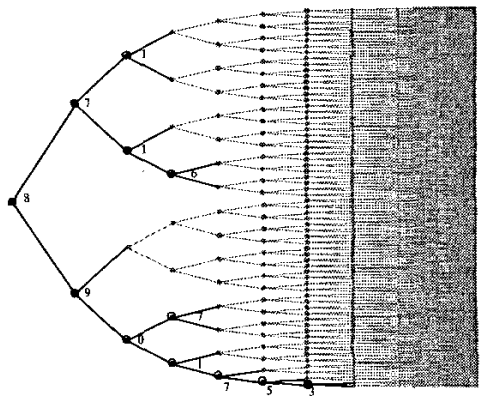

Figure 2: Example of a pruned tree over a reordered tree. Each internal final tree node is labeled by its associated context parameter index.

For that, the basic GRASP algorithm is slightly changed: First, the population of each basic context prior to the tree growing process is gathered by using all frames of a given type within a GOP. Then the tree is grown by choosing the best context parameter for splitting each node together with selecting the best initial state for that node. For the transmission of the corresponding initial count, a fixed length indicator of 6 bits is used to indicate the choice of the best fit to our pre-defined state table of 64 model probability states. To take into account the additional cost of $psc = 6$ bits for transmitting the state information, a modified cost function $\hat{J}$ as a substitute of J in (1) is defined as

$$\hat{J}(s) = \begin{cases} \ell_s + mc(d) + psc, & \text{if } \ell_s + psc \le \hat{J}(s_{c0}) + \hat{J}(s_{c1}) \\ \hat{J}(s_{c0}) + \hat{J}(s_{c1}) + mc(d), & \text{otherwise} \end{cases}.$$

4. EXPERIMENTAL RESULTS

We have applied the GRASP algorithm to various syntax elements in H.264/AVC. For that purpose, we have integrated our method into the current H.264/AVC test model and tested it for the 8 syntax elements macroblock skip flag, macroblock type, reference frame index, chroma intra prediction mode, coded block pattern for chroma information, significant coefficient flag, last significant coefficient flag, and coded block flag [6].

In our simulations, we have compared GRASP to CABAC using a set of progressive HD TV sequences (1280 × 720 pixels). Figure 3 shows the corresponding results obtained by using an IDR picture every 16 frames and two discardable bipredictive (B) frames between each pair of predictive (P) coded frames such that a GOP size of 16 frames has been obtained for all sequences. The total bit-rate savings for each of the tested context-based coding methods are plotted against the quantization parameter (QP) in the graphs of Fig. 3, where as an anchor an adaptive binary arithmetic coding method based on zero-order models for coding of the selected syntax elements was used.

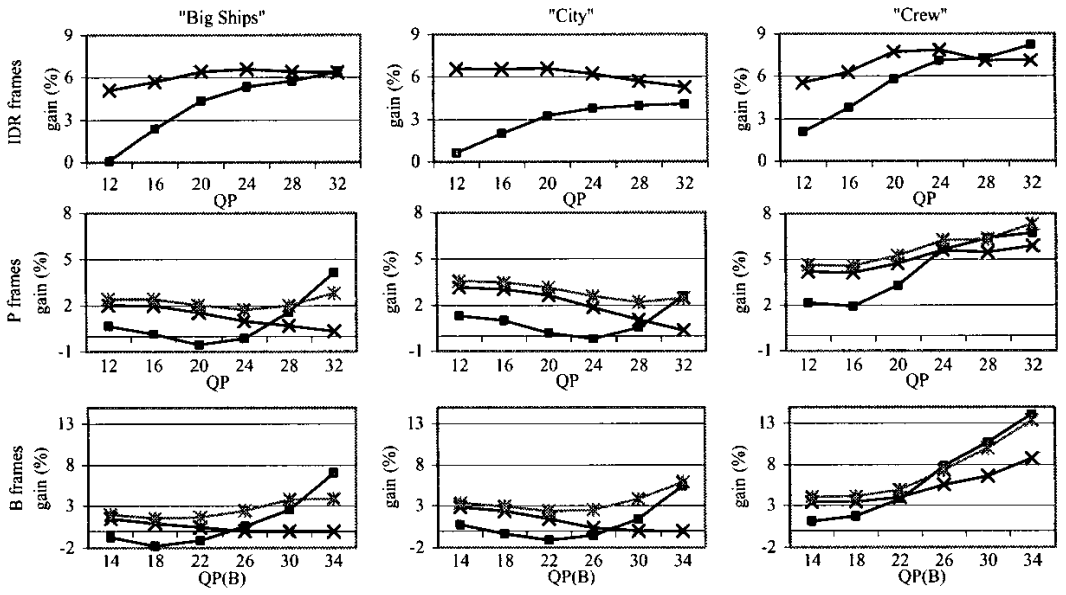

Figure 3: Comparison of GRASP for individual frames (-✕-), GRASP for group of pictures (-✳-) and CABAC (-■-) for HD TV sequences.

We have tested two versions of GRASP: the basic version, where GRASP was applied to each frame of a GOP separately and without the usage of initial states, and a second version, where a GOP-based optimal tree was selected for the frame types of P and B by applying GRASP as proposed in Section 3.3.[1] As can be seen from the graphs in the upper row of Fig. 3, for intra (I) coded frames a total bit-rate saving up to 5% relative to the original CABAC method is achieved by using GRASP. Here the gains of GRASP relative to CABAC are more significant in the range of lower QP values corresponding to higher bit rates. Compared to the anchor, a relatively constant overall gain of about 5–8% was obtained by the context-modeling method of GRASP for I frames only.

In the graphs of the middle and lower row of Fig. 3, the averaged total bit-rate savings for CABAC and the two versions of GRASP relative to the anchor are shown for coding of P frames and B frames, respectively. For these types of frames, we usually observe lower gains for all three tested context-modeling schemes, especially at lower QP values. In general, context modeling at lower rates seems to reach a point of diminishing returns, where not enough data for gathering reliable statistics of each context node is available. For the "Crew" sequence, however, relatively high gains were obtained at lower bit-rates, which are mainly attributed to the modeling of the macroblock skip flag. Overall, total bit-rate savings of up to 3% were achieved for GRASP relative to CABAC, where a consistently superior behavior can be observed for the GOP-based version of GRASP.

5. CONCLUSIONS

A new algorithm for optimization of binary context trees with application in adaptive context-based coding has been presented. The key elements of this method are a tree growing by reordering followed by tree pruning. Simulation results show that a performance similar or better than that of the original H.264/AVC CABAC algorithm can be achieved. The proposed scheme can be either used in a two-pass coding approach, where in the first pass the relevant statistics are gathered, or it can be used as an off-line tool for optimization of fixed context models in an adaptive coder.

[1] Since only one intra frame was coded in a GOP (as an IDR picture), no GOP-based optimal tree selection was performed for that frame type.

ACKNOWLEDGMENT

Marta Mrak wishes to acknowledge hospitality provided by Fraunhofer-Institute HHI, Berlin, Germany and financial support provided by Deutscher Akademischer Austauschdienst (DAAD), Germany for this research.

REFERENCES

[1] Marpe, D., Schwarz, H., Wiegand, T., "Context-Based Adaptive Binary Arithmetic Coding in H.264/AVC Video Compression Standard", *IEEE Trans. on Circ and Systems for Video Technology*, to be published.

[2] Rissanen, J., "Universal Coding, Information, Prediction, and Estimation", *IEEE Trans. on Info. Theory*, vol. 30, pp. 629-636, July 1984.

[3] Willems, F.J.W., Shtarkov, Y.M., Tjalkens, T.J., "The Context Tree Weighting Method: Basic Properties", *IEEE Trans. on Info. Theory*, Vol. 41, pp. 653-664, May 1995.

[4] Weinberger, M.J., Rissanen, J., Arps, R.B.; "Application of Universal Context Modeling to Lossless Compression of Gray-Scale Images", *IEEE Trans. on Image Processing*, Vol. 5, No. 4, pp. 575-586, April 1996.

[5] Ekstrand, N., "Lossless Compression of Grayscale Images via Context Tree Weighting", *Proc. IEEE Data Compression Conference*, pp. 132-139, April 1996.

[6] Wiegand, T. (Ed.), "Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14496-10 AVC," Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Document JVT-G050, March 2003.