COMPARISON OF CONTEXT-BASED ADAPTIVE BINARY ARITHMETIC CODERS IN VIDEO COMPRESSION

Marta Mrak,* Detlev Marpe,** and Sonja Grgic*

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

Abstract: *A comparison of two context-modeling methods, CABAC and GRASP, is presented with regard to applications in video compression. Both context-based entropy coding methods are tested for various probability estimation techniques, for different sources, and for different quantization parameters by using the test model of the H.264/AVC video coding standard. Our experimental results show that GRASP provides a significant gain over CABAC, especially for intra frames coded at medium to high bit-rates.*

Keywords: Video compression, entropy coding, probability estimation, context modeling.

1. INTRODUCTION

In video coding, where highly efficient compression is sought and complexity is not the primary concern, more sophisticated entropy coding methods can be applied in order to attain higher compression ratios. Such methods are usually based on context modeling and suitable probability estimation techniques, which both together enable the construction of context dependant probability models. Given these properly chosen probability models, the usage of arithmetic coding allows the construction of code words arbitrarily close to the minimal code length.

Context-based entropy coding was first introduced in the area of hybrid block-based video coding in *Context-Based Adaptive Binary Arithmetic Coding* (CABAC) of the emerging H.264/AVC video compression standard [1]. CABAC relies on some proper (fixed) selection and ordering of *context parameters*, i.e., an a priori chosen subset of prior encoded symbols or parts of them for the estimation of conditional probabilities. The more complex algorithm of *Growing by Reordering and Selection by Pruning* (GRASP), introduced in [2], is based on binary context tree models and offers an adaptive selection of context parameters by using a typically larger set of prior encoded data for conditioning than that exploited in CABAC.

In our comparative study, GRASP has been applied to pre-selected binary-valued syntax elements or individual bins of non-binary syntax elements in H.264/AVC [3]. Various probability estimation methods in combination with CABAC and GRASP have been explored. The performance of each combination has been evaluated within H.264/AVC by using a set of high-definition television (HD TV) test sequences.

2. PROBABILITY ESTIMATION

Arithmetic coding as the optimal entropy coding method, adds approximately $-\log_2 p(x(i))$ bits to the final code word after encoding a symbol $x(i)$ with probability $p(x(i))$. The final output code length L in bits for the data sequence of length n is ideally given by

$$L = -\log_2 \prod_{i=0}^{n-1} p\big(x(i)\big).$$

In order to minimise the code length, the arithmetic coder must operate in conjunction with a probability estimation machine that estimates the probability of each possible event at each time instance i. The probability estimation process should be adaptive in order to allow an adaptation to the real underlying source statistics.

The basic adaptive probability estimator of a binary source (with coding events $k \in \{0,1\}$) is given by

$$p\big(x(i)=k\big) = \frac{c_k(i)}{c_0(i)+c_1(i)}. \tag{1}$$

The occurrence counts of 0's and 1's ($c_0(i)$, $c_1(i)$) in (1) will be set to some initial integer values (κ_0, κ_1) at the beginning of the encoding process for each coding unit.

After encoding of each symbol the occurrence counts are updated by

$$c_k(i) = \begin{cases} c_k(i-1)+1, & \text{if } x(i-1)=k \\ c_k(i-1), & \text{otherwise} \end{cases}. \tag{2}$$

For data sequence, which contains c_0 and c_1 occurrence counts of 0's and 1's, respectively, and starting from initial values κ_0 and κ_1, the final so-called *adaptive code length* ℓ is defined by

$$\ell\big(c_0, c_1, \kappa_0, \kappa_1\big) = -\log_2 \frac{(c_0-1)!\,(c_1-1)!}{(c_0+c_1-1)!} \cdot \frac{(\kappa_0+\kappa_1-1)!}{(\kappa_0-1)!\,(\kappa_1-1)!}. \tag{3}$$

Here, ℓ does not depend on possible fluctuation of the source statistics.

The scaled-count iteration method [4], however, enables the adaptation to potential variations of the underlying source statistics by using a scaling of occurrence counts after the updating of counts in (2):

$$c_k(i) = \begin{cases} c_k(i), & \text{if } C(i)_{\min} \le C^*_{\min} \\ \beta\big(c_k(i)+\alpha\big)-\alpha, & \text{otherwise} \end{cases}, \tag{4}$$

where $C(i)_{\min} = \min\{c_0(i), c_1(i)\}$ and $\beta = \big(C(i)_{\min}+\alpha-1\big)/\big(C(i)_{\min}+\alpha\big)$. In our experiments, α is set to 1 and $C^*_{\min}$ denotes an appropriately chosen threshold (see Section 4). Note

that by using the scaled-count iteration, the probability estimator of (1) is driven by the scaled occurrence counts of (4).

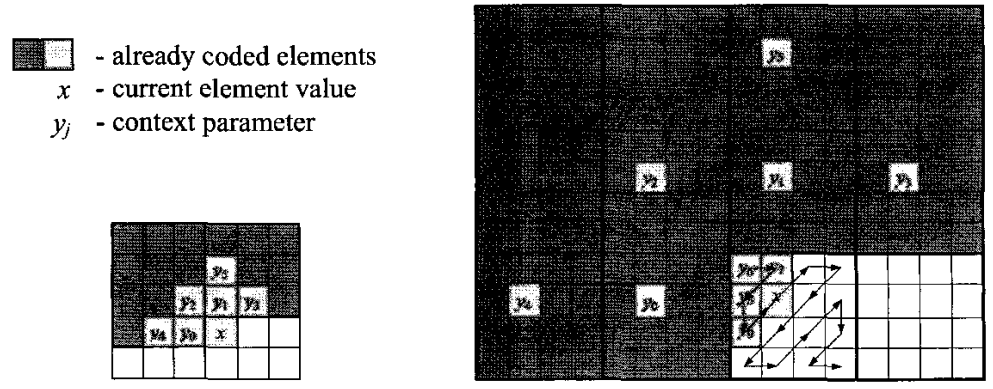
Both probability estimators require multiplications (or even divisions), which are not suitable for high-speed performance. Note that in contrast to that, CABAC relies on a low-complexity method for probability estimation by using a finite state machine with a table-based transition process between 64 different probability states [1].

3. CONTEXT MODELING

Context modeling methods provide models for probability estimation, in which the coding of each symbol is conditioned on the context in which that symbol occurs. Thus, prior encoded symbols or parts of them are used to exploit inter-symbol redundancy. Figure 2 shows two examples of *context templates T*, which are used in the application of GRASP to video compression. As shown in Figure 2-a), the corresponding context parameters can be chosen from neighboring symbols in the spatial domain, or both from the spatial and frequency domain, as shown in Figure 2-b) for the coding of the AC transform coefficient at the fourth scanning position (*AC4*). For the example of the *AC4* transform coefficient, the related context template consists of coefficients at the same frequency position (indices 0-5) in neighboring blocks as well as already coded coefficients from the current block (indices 6-9).

For each symbol x to encode, the conditional probability $p(x|T)$ is estimated by switching between different probability models according to the already coded neighboring symbols in the template T. The estimated probability distribution $p(x|T)$ is used to drive an 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, which in case of the probability estimators defined in Section 2 consists of an update of the occurrence counts according to (2) or (4).

Estimating $p(x|T)$ by using past sample statistics, however, 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. Especially for short data sequences, which are typically observed in video coding applications, great care must be taken to control the model order, i.e., the size of the template as well as the ordering of the context parameters within the template.



a) Template for macroblock and sub-block syntax elements

b) Template for *AC4* transform coefficient

Figure 2. Templates of context parameters.

3.1. Context Modeling in CABAC

Context modeling in CABAC involves pre-defined context templates T of at most two neighbors (i.e. $T = (y_0, y_1)$ in Figure 2-a)). For example, the relation

$$ctx\_idx(\text{coded\_block\_flag}(x)) = \text{coded\_block\_flag}(y_0) + 2 \times \text{coded\_block\_flag}(y_1)$$

specifies the index *ctx_idx* for addressing the probability model for coding of the syntax element *coded_block_flag* for a given block x, depending on the value of coded_block_flag at the neighboring blocks y_0 and y_1. In H.264/AVC, coded_block_flag is used to indicate for a given transform block whether there are significant (non-zero) coefficients inside the regarded block of transform coefficients.

For non-binary valued symbols, CABAC provides appropriately defined binarization schemes, i.e., mappings of non-binary valued symbols to prefix-free binary codes. This property of H.264/AVC also simplifies the application of more complex techniques for context modeling, as given in the next section.

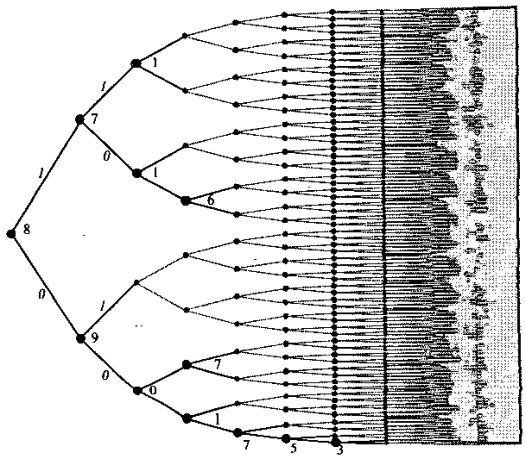


Figure 3. An example of a context tree.

3.2. GRASP

The finite memory model for a binary alphabet can be represented as a binary tree structure. In general, the nodes in a context tree are labeled by the context parameters indices and the branches are labeled by the symbols in the alphabet, as depicted in Figure 3. For a given context tree and a given observation of past symbols, the corresponding node is reached by taking the path starting at the root node and by following the nodes according to the given context parameter values.

The GRASP algorithm enables the design of efficient context trees for a given data sequence. 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, 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.

In the following, we describe a more general form of the GRASP algorithm, which offers the possibility 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 whole *group of pictures* (GOP), which we refer to as a *GOP-based optimal tree*. For achieving a faster adaptation during encoding, an appropriately chosen initial probability state is assigned to each terminal node of the GOP-based optimal tree. For the transmission of the suitable state information, a fixed length indicator of *psc* bits is used to indicate the choice of the best fit to a pre-defined state table. Each probability state in this table is related to a specific pair of initial occurrence counts κ_0 and κ_1.

GRASP is a two-step approach. In the first step, a full balanced tree of a pre-defined depth is grown, as described in section 3.2.1. In the second step, the resulting full balanced tree is pruned, as described in section 3.2.2.

Tree growing starts with gathering the population of each basic context corresponding to the leaves of a regular full balanced tree. For F frames of same type in a GOP, the population of F sets of basic contexts has to be gathered - one set of basic contexts per frame.

3.2.1. Tree growing by reordering

Let y_j^d denote the assignment of a context parameter y_j with $j \in \{0,\ldots,N-1\}$ to a node at depth d, where N is a number of context parameters. Let the set $\left\{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}\right\}$ denote the sequence of assignments of context parameter to the sequence of nodes over which the node at depth d is reached in the given context tree.

Tree growing starts from the root node by using the following Steps 1 to 3:

Step 1: Start constructing a 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 $\left\{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}\right\}$ 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

$$L_j = -\sum_{i=0}^{1} \sum_{f=1}^{F} \ell\left(c_0^{f,z_{i,j}}, c_1^{f,z_{i,j}}, \kappa_0^{z_{i,j}}, \kappa_1^{z_{i,j}}\right), \tag{5}$$

where ℓ is defined by (5). $z_{i,j} = \left\{y_{j_0}^0, y_{j_1}^1, \ldots, y_{j_{d-1}}^{d-1}, y_j^d = i\right\}$ is the subsequence of assignments to reach the node at depth $d+1$, and $c_0^{f,z_{i,j}}$, $c_1^{f,z_{i,j}}$ are the corresponding occurrence counts of 0's and 1's at that node for frame f. $\kappa_0^{z_{i,j}}$, $\kappa_1^{z_{i,j}}$ are the corresponding initial counts of 0's and 1's at that node, respectively.

The context parameter $y_{j_d}^d$ together with a properly chosen initial probability state minimises the adaptive code length of the corresponding child nodes. $y_{j_d}^d$ is assigned to the current node at depth d and the initial state indices are assigned to its child nodes.

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

3.2.2. 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 cost functional J given by:

$$J(s) = -\sum_{f=1}^{F} \ell\left(c_0^{f,s}, c_1^{f,s}, \kappa_0^s, \kappa_1^s\right) + psc\,.$$

Step 1: Set the cost functional J' of each leaf node to its J value: $J'(s) = J(s)$. The node evaluating starts at depth N - 1.

Step 2: Comparing child nodes s_{ch1} and s_{ch1} to the current node s_p, find $J'(s_p)$ as:

$$J'(s_p) = \begin{cases} J(s_p) + mc(d) + psc, & \text{if } J(s_p) + psc \le J'(s_{ch0}) + J'(s_{ch1}) \\ J'(s_{ch0}) + J'(s_{ch1}) + mc(d), & \text{otherwise} \end{cases}$$

where the model cost $mc(d) = \lceil \log_2(D - d + 1) \rceil$ represents the amount of bits that is needed to signal the context parameter indices together with a terminal node flag. If $J(s_p) + psc \le J'(s_{ch0}) + J'(s_{ch1})$ prune the branch below the parent node.

Step 3: Repeat Step 2 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 the recursively defined cost functional J'. The functional J' rates both the average adaptive code length and the model description cost. 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.

4. EXPERIMENTAL RESULTS

In this section, experimental results are presented for various combinations of probability estimation methods and context modeling algorithms:

- CABAC context modeling and CABAC table-based probability estimation (original CABAC as specified in [3]);
- CABAC context modeling with basic adaptive probability estimation;
- GRASP with basic adaptive probability estimation;
- GRASP with scaled-count iteration method for probability estimation.

CABAC was designed by a joint optimisation of context modeling and probability estimation for a wide variety of sequence resolutions [1]. However, in addition to the original CABAC of H.264/AVC, a version of CABAC using the basic adaptive probability estimation method is examined here to enable a comparison of GRASP and CABAC context modeling without the influence of a chosen probability estimation method. Since the GRASP context modeling process with the usage of basic adaptive probability estimation does not into account possible fluctuations in the underlying source statistics, the scaled-count iteration method for probability estimation is additionally tested in conjunction with GRASP.

These four context-based coders have been evaluated in the environment of the current H.264/AVC test model. We have tested it for 8 selected syntax elements of H.264/AVC:

- Syntax elements related to macroblock and sub-macroblock levels: macroblock skip flag, macroblock type, reference frame index, chroma intra prediction mode, and coded block pattern for chrominance information;
- Syntax elements related to transform coefficients: significant coefficient flag, last significant coefficient flag, and coded block flag.

For each of the given transform block types, coded block flag is signaled first to indicate whether there are significant coefficients inside the regarded block of transform coefficients. If the coded block flag is zero, no further information needs to be transmitted for the block. If the coded block flag indicates that a block has significant coefficients, a significance map is encoded. For each coefficient in scanning order, a significant coefficient flag is transmitted. If the significant coefficient flag is one, that is, if a non-zero coefficient exists at this scanning position, a further last significant coefficient flag is sent. This flag indicates if the current significant coefficient is the last one inside the block or if further significant coefficients follow. For more details, the reader is referred to [3].

In our simulations a set of progressive HD TV sequences (1280 × 720 pixels) is used. Using an IDR picture every 16 frames and two discardable B frames between each pair of P coded frames, a GOP size of 16 frames were obtained for all sequences.

Because there is only one intra (I) coded frame in a GOP, GRASP uses $F(\mathrm{I}) = 1$ for I frames. Moreover, for I frames all initial occurrence counts are set to 1 ($\kappa_0 = \kappa_1 = 1$) and $psc(\mathrm{I})$ is set to 0. For predictive (P) coded frames and bi-predictive (B) coded frames F depends on the chosen GOP structure, which in our experiments was chosen such that $F(\mathrm{P}) = 5$ and $F(\mathrm{B}) = 10$. A pre-defined state table of 64 states is used for P and B frames ($psc(\mathrm{P}) = psc(\mathrm{B}) = 6$).

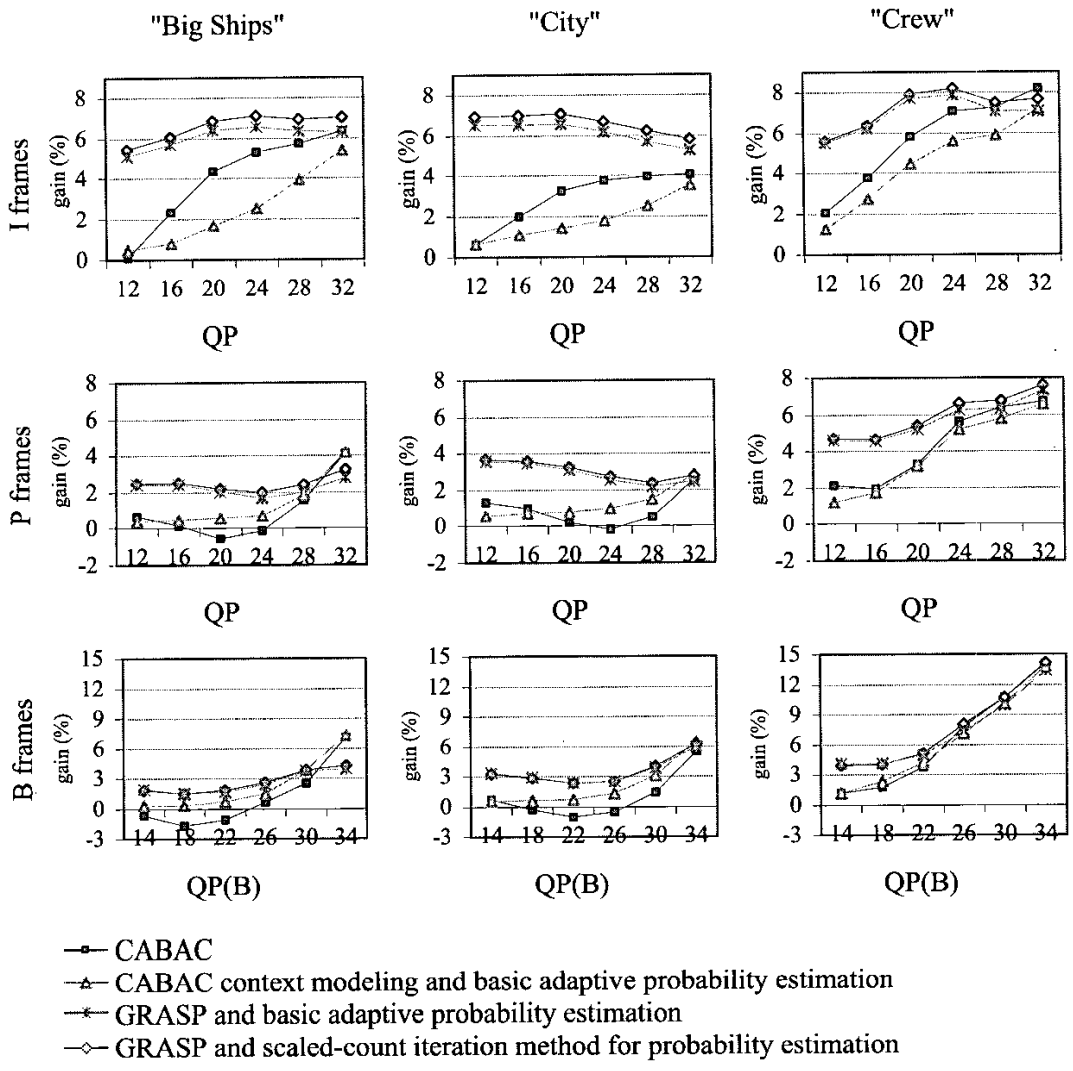


Figure 4. Comparison of context-based coders for selected elements of H.264/AVC relative to coding using zero-order context models.

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 Figure 4, where as an anchor a binary arithmetic coding using basic adaptive probability estimation based on zero-order models for coding of the selected syntax elements was chosen.

As can be seen from the graphs in Figure 4, for I frames CABAC context modeling works better if its own probability estimation method is used. For the mid-range quantization parameters the original CABAC method can achieve total bit-rate savings of up to 3% compared to CABAC using basic adaptive probability estimation. From the graphs in Figure 4 containing results for P and B frames, it can be seen that the choice of the probability estimation method does not make any significant difference in conjunction with CABAC context modeling for these frame types.

Compared to both coding methods using CABAC context modeling, GRASP in conjunction with the basic adaptive probability estimation performs significantly better. For I frames, total bit-rate savings of up to 5% relative to the original CABAC method were achieved. 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 GRASP context-modeling for I frames only.

For P frames and B frames, we usually observed lower gains of all tested context-based coding 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.

Scaled-count iteration method is tested on context models obtained by using GRASP. The parameter $C^*_{\min}$ was set to 30, 50 and 20 for I, P and B frames respectively. However, only a maximum gain of 0.5% is achieved by using the scaled-count iterated GRASP method in comparison to GRASP using the basic adaptive probability estimation method. Overall, both tested versions of GRASP have provided the most distinguished gains relative to the anchor.

5. CONCLUSION

Methods for context modeling of binary sources with application in video compression are presented. H.264/AVC CABAC uses pre-defined fixed context models for each syntax element. The GRASP method is based on a tree growing by reordering and a tree selection by pruning process for an optimized, data-dependant modeling of binary context trees. We have demonstrated its use together with appropriately chosen probability estimation techniques. For that purpose we have applied our proposed technique to the H.264/AVC standard that includes the original CABAC entropy coding method.

Our results show that context modeling in CABAC for I frame coding performs best using its own table-based probability estimation technique. CABAC context modeling itself has been compared to the more complex context modeling method of GRASP in such a way that both methods have been supplied with the same basic adaptive probability estimation method. Even in that case GRASP has shown a superior gain relative to CABAC, at least at medium to high bit-rates.

Using the scaled-count iteration method in conjunction with GRASP, possible non-stationary characteristics of the underlying statistics in each node of the context tree, obtained by GRASP, have been exploited. However, only minor additional gains have been observed in that case. Overall, total bit-rate savings of up to 5% were achieved for GRASP relative to CABAC. From that observation we conclude that a highly performing context modeling method should take great care on the ordering of context parameters within a context tree [2] and that it should be able to employ a maximum of different context parameters for building an optimal context tree.

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 Sys. for Video Technology*, to be published.

[2] Mrak, M., Marpe, D., Wiegand, T., "Application of Binary Context Trees in Video Compression", *Proc. Picture Coding Symposium* (PCS), Saint Malo, France, April 2003.

[3] Wiegand, T., Sullivan, G., "Draft Text of Final Draft International Standard (FDIS) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", JVT-G050, March 2003.

[4] Duttweiler, D.L., Chamzas, C., "Probability Estimation in Arithmetic and Adaptive-Huffman Entropy Coders", *IEEE Trans. on Image Processing*, 4(3):237-246, March 1995.