H.264 / MPEG-4 Part 10 White Paper

Context-Based Adaptive Arithmetic Coding (CABAC)

1. Introduction

The Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG are finalising a new standard for the coding (compression) of natural video images. The new standard [1] will be known as H.264 and also MPEG-4 Part 10, "Advanced Video Coding". The standard specifies two types of entropy coding: Context-based Adaptive Binary Arithmetic Coding (CABAC) and Variable-Length Coding (VLC). This document provides a short introduction to CABAC. Familiarity with the concept of Arithmetic Coding is assumed (see chapter 8 of [2] for an introduction to Arithmetic Coding).

Please note that the H.264 draft standard is not yet finalised and so readers are encouraged to refer to the latest version of the standard.

2. Context-based adaptive binary arithmetic coding (CABAC)

When entropy_coding_mode is set to 1, an arithmetic coding system is used to encode and decode H.264 syntax elements. The arithmetic coding scheme selected for H.264, Context-based Adaptive Binary Arithmetic Coding or CABAC [3], achieves good compression performance through (a) selecting probability models for each syntax element according to the element's context, (b) adapting probability estimates based on local statistics and (c) using arithmetic coding.

Coding a data symbol involves the following stages.

1. Binarization: CABAC uses Binary Arithmetic Coding which means that only binary decisions (1 or 0) are encoded. A non-binary-valued symbol (e.g. a transform coefficient or motion vector) is "binarized" or converted into a binary code prior to arithmetic coding. This process is similar to the process of converting a data symbol into a variable length code but the binary code is further encoded (by the arithmetic coder) prior to transmission.

Stages 2, 3 and 4 are repeated for each bit (or "bin") of the binarized symbol.

2. Context model selection: A "context model" is a probability model for one or more bins of the binarized symbol. This model may be chosen from a selection of available models depending on the statistics of recently-coded data symbols. The context model stores the probability of each bin being "1" or "0".

3. Arithmetic encoding: An arithmetic coder encodes each bin according to the selected probability model. Note that there are just two sub-ranges for each bin (corresponding to "0" and "1").

4. Probability update: The selected context model is updated based on the actual coded value (e.g. if the bin value was "1", the frequency count of "1"s is increased).

3. The coding process

We will illustrate the coding process for one example, MVD_x (motion vector difference in the x-direction).

1. Binarize the value MVD_x . Binarization is carried out according to the following table for $|MVD_x|<9$ (larger values of MVD_x are binarized using an Exp-Golomb codeword).

$\|MVD_x\|$	Binarization
0	0

1	10
2	110
3	1110
4	11110
5	111110
6	1111110
7	11111110
8	111111110

(Note that each of these binarized codewords are uniquely decodeable).

The first bit of the binarized codeword is bin 1; the second bit is bin 2; and so on.

2. Choose a context model for each bin. One of 3 models is selected for bin 1, based on previous coded MVD values. The L1 norm of two previously-coded values, e_k, is calculated:

$e_k = |MVD_A| + |MVD_B|$ where A and B are the blocks immediately to the left and above the current block (respectively).

e_k	Context model for bin 1
0 ? e_k < 3	Model 0
3 ? e_k < 33	Model 1
33 ? e_k	Model 2

If e_k is small, then there is a high probability that the current MVD will have a small magnitude; conversely, if e_k is large then it is more likely that the current MVD will have a large magnitude. We select a probability table (context model) accordingly.

The remaining bins are coded using one of 4 further context models:

Bin	Context model
1	0, 1 or 2 depending on e_k
2	3
3	4
4	5
5	6
6 and higher	6

3. Encode each bin. The selected context model supplies two probability estimates: the probability that the bin contains "1" and the probability that the bin contains "0". These estimates determine the two sub-ranges that the arithmetic coder uses to encode the bin.

4. Update the context models. For example, if context model 2 was selected for bin 1 and the value of bin 1 was "0", the frequency count of "0"s is incremented. This means that the next time this model is selected, the probability of an "0" will be slightly higher. When the total number of occurrences of a model exceeds a threshold value, the frequency counts for "0" and "1" will be scaled down, which in effect gives higher priority to recent observations.

4. The context models

Context models and binarization schemes for each syntax element are defined in the standard. There are a total of 267 separate context models, 0 to 266 (as of September 2002) for the various syntax elements. Some models have different uses depending on the slice type: for example, skipped macroblocks are not permitted in an I-slice and so context models 0-2 are used to code bins of mb_skip or mb_type depending on whether the current slice is Intra coded.

At the beginning of each coded slice, the context models are initialised depending on the initial value of the Quantization Parameter QP (since this has a significant effect on the probability of occurrence of the various data symbols).

5. The arithmetic coding engine

The arithmetic decoder is described in some detail in the Standard. It has three distinct properties:
1. Probability estimation is performed by a transition process between 64 separate probability states for "Least Probable Symbol" (LPS, the least probable of the two binary decisions "0" or "1").
2. The range R representing the current state of the arithmetic coder is quantized to a small range of pre-set values before calculating the new range at each step, making it possible to calculate the new range using a look-up table (i.e. multiplication-free).
3. A simplified encoding and decoding process is defined for data symbols with a near-uniform probability distribution.

The definition of the decoding process is designed to facilitate low-complexity implementations of arithmetic encoding and decoding. Overall, CABAC provides improved coding efficiency compared with VLC at the expense of greater computational complexity.

6. References

1 ITU-T Rec. H.264 / ISO/IEC 11496-10, "Advanced Video Coding", Final Committee Draft, Document JVT-E022, September 2002
2 I. Richardson, "Video CODEC Design", John Wiley & Sons, 2002.
3 D. Marpe, G Blättermann and T Wiegand, "Adaptive Codes for H.26L", ITU-T SG16/6 document VCEG-L13, Eibsee, Germany, January 2001