Proceedings of the 5th WSEAS Int. Conf. on MULTIMEDIA, INTERNET AND VIDEO TECHNOLOGIES, Corfu, Greece, August 17-19, 2005 (pp120-125)

Speeding up Dirac's Entropy Coder

HENDRIK EECKHAUT BENJAMIN SCHRAUWEN MARK CHRISTIAENS
JAN VAN CAMPENHOUT
Parallel Information Systems (PARIS)
Electronics and Information Systems (ELIS)
Ghent University (UGent)
St.-Pietersnieuwstraat 41, 9000 Gent, Belgium

Abstract: The Dirac codec is a new prototype video coding algorithm from BBC R&D based on wavelet technology. Compression-wise the algorithm is broadly competitive with the state-of-the-art video codecs but computational-wise the execution time is currently poor. One of the large bottlenecks is the arithmetic coder. Hence we took this part under investigation and replaced it with a faster variant, the M-coder, that is also used in the H.264/AVC codec. The resulting coder is substantially faster. We reached a convincing speedup. Thanks to an intelligent tuning of initialisation and adaption parameters the impact of the compression performance is very limited.

Key-Words: Dirac video codec, arithmetic coding, M-coder

1 Introduction

In January 2003, BBC R&D produced a prototype video coding algorithm, "Dirac" [1], based on wavelet technology. The algorithm originally targeted high definition video but has been further developed to optimise it for Internet streaming resolutions and seems broadly competitive with state-of-the-art video codecs. The main philosophy behind the Dirac codec is 'keep it simple' which is an ambitious aim since video codecs, particularly those with state-of-the-art performance, tend to be fearsomely complex.

The Dirac codec has been developed as a research tool, not a product, as a basis for further developments. An experimental version of the code, written in C++, was released under an Open Source licence agreement on 11th March 2004. For our experiments we used the source code of version 0.5.1 of the Dirac codec; the most recent version at the time of writing.

Dirac's weakest point is currently its execution time. Real time decoding is limited to smaller resolutions, even for the latest processors. In this paper we focus on the entropy coding part of the algorithm which is responsible for a large portion of the decoding time. Certainly for higher bit rates the arithmetic coder is an obstinate bottleneck. Hardware acceleration with FPGAs could solve this [2], but is for the time being not a general applicable solution. We solve the problem by equipping Dirac with a less computationally intense coder, a modified version of the arithmetic coder also used in H.264/AVC [3], which resulted in a much shorter execution ime.

This paper is organized as follows: In Section 2 we introduce the Dirac codec and describe its main parts. In Section 3 we describe the arithmetic coder of the Dirac codec. We pinpoint its main problem, the arithmetic coder is to computationally expensive, and we propose another arithmetic coder and report on the benefits and disadvantages. Section 4 presents the compression and timing results of the new coder versus the original version. Finally Section 5 summarizes our findings and concludes this work.

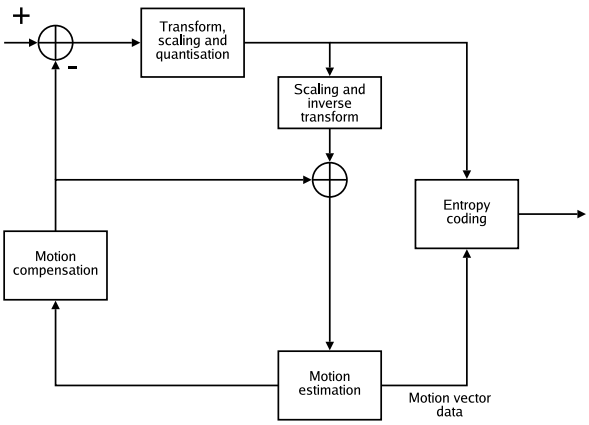

Figure 1: High-level overview of the Dirac encoder [1].

2 The Dirac Algorithm

Overall, the Dirac codec is a wavelet-based motion-compensated codec. The coder has the architecture shown in Figure 1, whilst the decoder performs the inverse operations. There are four main elements or modules to the coder:

Transform, scaling and quantisation: Transform and scaling involves taking frame data and applying a wavelet transform and scaling the coefficients to perform quantisation. For quantisation a rate distortion optimisation algorithm is used to strip information from the frame data that results in as little visual distortion as possible.

Entropy Coding (EC): EC is illustrated in Figure 2 and consists of three stages: binarization, context modeling and arithmetic coding (AC). The purpose of the first stage is to provide a bit stream with easily analysable statistics that can be encoded using AC, which can adapt to those statistics, reflecting any local statistical features. The context modelling in Dirac is based on the principle that whether a coefficient is small (or zero, in particular) or not is well-predicted by its neighbours and its parents. AC performs lossless compression and is further elaborated in Section 3. The same infrastructure is applied to quantised transform coefficients and to motion vector (MV) data.

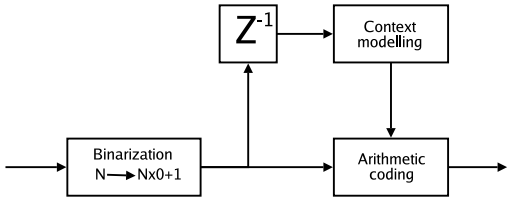

Figure 2: Entropy coding block diagram [1].

Motion estimation (ME): ME exploits temporal redundancy in video streams by looking for similarities between adjacent frames. Dirac implements hierarchical motion estimation and defines three types of frame. Intra (I) frames are coded without reference to other frames in the sequence. Level 1 (L1) frames and Level 2 (L2) frames are both inter frames, that is they are coded with reference to other previously coded frames. The difference between L1 and L2 frames is that L1 frames are also used as temporal references for other frames, whereas L2 frames are not.

Dirac uses Overlapped Block-based Motion Compensation (OBMC) to avoid block-edge artifacts which would be expensive to code using wavelets. Dirac also provides Sub-pixel Motion Compensation with motion vectors to 1/8th pixel accuracy.

Motion compensation (MC): MC is the inverse operation of ME. It involves using the motion vectors to predict the current frame in such a way as to minimise the cost of encoding the residual data (the difference between the actual and the motion estimated frame). A trade off is made between accuracy and motion vector bit rate.

3 Arithmetic Coding

Arithmetic coding is an entropy coding technique which assigns fractional length code words to the input symbols provided that a good estimate of the input distribution function (IDF) of each successive input symbol is available. An arithmetic encoder reaches its optimal compression ratio when the estimated IDF equals the actual IDF. In fact, with perfect estimates, it is possible to reach the Shannon entropy limit of coding efficiency (with an infinite precision arithmetic coder).

3.1 Dirac's Arithmetic Coder

A traditional technique to estimate the IDF of each input symbol is to use a frequency table. Each time an input symbol takes on a certain value, the frequency count corresponding to that value is increased. The probability that the next symbol will take on a certain value is then estimated by its relative occurrence in the frequency table. The frequency table is renormalised at regular intervals to prevent overflow and to allow the tracking of non-stationary IDFs. This technique is used in the Dirac codec.

To improve the IDF estimation, context modelling is used: the symbol stream is split in several sub streams with similar statistical properties which improves estimation performance. For example in Dirac, motion vectors and the various wavelet subbands types are coded separately. In the Dirac codec all context IDF estimates are initialized with equal probability for 1 and 0.

Dirac's AC implementation [1, 2] is computationally very expensive since, in order to obtain an estimate for the probability of a value, a division operation is to be performed. Also the AC itself uses a multiplication to construct the fractional code words. The decoding speed of the Dirac codec is currently the most critical point to be considered, we will thus focus on solving the computational issue without compromising the compression performance to much.

3.2 A Faster Arithmetic Coder

Because Dirac is currently too slow, we have replaced the Dirac AC with an AC which introduces several approximations (which will degrade the compression performance) but which has a much better computational performance.

The arithmetic codec used by H.264/AVC [3] (also called M-coder) was inspired by the Q-coder [4, 5, 6] and MQ-coder [7] but is faster and achieves better compression than state-of-the-art MQ-coders [8] (used for example in JPEG2000 [7]). It avoids the overhead incurred by frequency counting by estimating the IDF using a finite state machine (FSM). The state of the FSM is updated each time a new input symbol arrives. With this state an estimate for the IDF of the next input symbol is associated. The M-coder is multiplication-free due to the use of precalculated tables and quantisation.

We used an in house developed, rigorous technique which allows the M-coder to be tuned[a] to optimally operate on a input stream with predefined characteristics. The technique is based on a Markov model of the IDF estimating FSM. It takes into account the expected input stream and optimizes the estimated IDFs such that an optimal compression is attained. We are thus able to plug in the M-coder in Dirac and tune it so that it optimally operates on the characteristic symbol stream of Dirac. The optimization is done by generating a trace file of all processed symbols for a large benchmark movie and then generating a non-stationary statistical model for the various contexts. Using these models the parameters of the M-coder are tuned.

We also improved the context modelling: we introduced many more context models and we initialised the estimated IDFs for each of these context models so that as little as possible warm-up time is needed. As stated in [9], the gain of correctly initialising the AC is modest but since the cost is almost zero – the AC has to be initialised anyway – it is a welcome improvement.

We introduced a total of 482 context models which are almost the same models as those from original Dirac algorithm: 50 models for motion vector data and 16 models for subband data. But now the subband models are also dependent on the color channel

[a] Only the estimated IDF table is changed. This table associates an estimated IDF to each of the states of the FSM. The FSM itself is left unchanged.

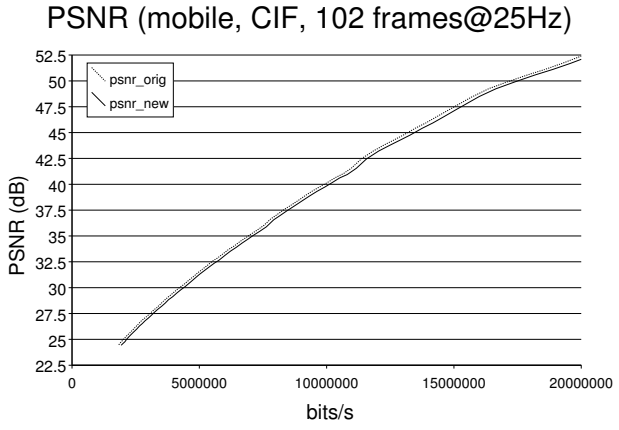

Figure 3: *PSNR vs. bitrate for 102 CIF-frames of the "mobile" sequence.*

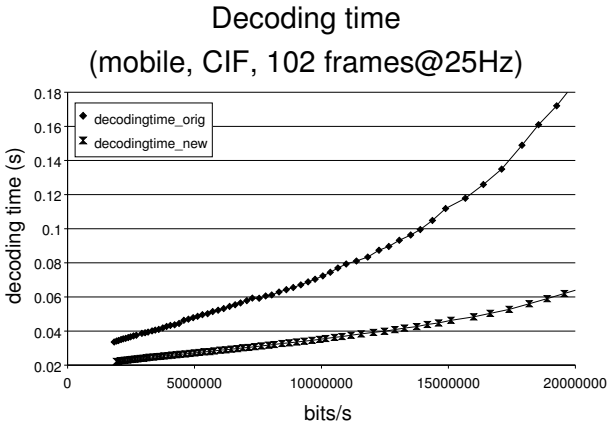

Figure 4: *Decoding time per frame vs. bitrate for 102 CIF-frames of the "mobile" sequence.*

(Y, U, V), the frame type (I, L1, L2) and the wavelet band type (DC, LF, others). Note that only 16 of them are used simultaneously; these are the 16 contexts used in this color channel, frame and wavelet band type. A context model thus has become one of the 16 old Dirac models plus an estimated IDF initialisation which is dependent on color, frame and band type.

4 Results

In Figure 3 we plotted the average PSNR for decoding 102 frames of the "foreman" sequence (CIF@25Hz) at different bit rates. The PSNR of each frame is calculated as follows. If $Y_{i,j}$, $U_{i,j}$ and $V_{i,j}$ and $Y'_{i,j}$, $U'_{i,j}$ and $V'_{i,j}$ are resp. the luminance and the two chrominance channels of the original and reconstructed frame of $h \times w$ pixels, then the PSNR is defined as follows:

$$10 \log_{10} \frac{255^2 \frac{3}{2} hw}{\sum (Y - Y')^2 + \sum (U - U')^2 + \sum (V - V')^2} \quad (1)$$

The PSNR of the original and the new algorithm is almost the same. The original AC is slightly better but the difference is never larger than 0.5 dB. We also measured the decoding time per frame for the same bit rates as Figure 3 in Figure 4. The measurements were done on a AMD Athlon 64 3500+ processor. As can be seen in Figure 4, the original algorithm never reached real-time decoding ($< 0.04s/frame$). The M-coder is real-time for bit rates up to 10 million bits per second ($\sim$ 43 dB PSNR). Notice the exponential behaviour of the original implementation versus the linear behaviour of the M-coder. This is easily explained by the fact that the original algorithm needs a multiplication and a division for each symbol. The new version only uses lookups, additions and shift operations.

We repeated the same experiment for 49 frames of another sequence with a larger resolution (720×576 pixels): the "snowboard" sequence, that is available on the Dirac website. The difference in PSNR is now smaller. The difference in decoding time on the contrary is still much better for the new arithmetic coder. It is more than two times faster for most bit rates.

Finally we also measured the coding performance and decoding time for a high definition sequence ("snowboard") of 1280 by 720 pixels. The two PSNR-curves of Figure 7 are now nearly indistinguishable. The new decoder is again at least two times faster for most bit rates, which confirms the previous conclusions.

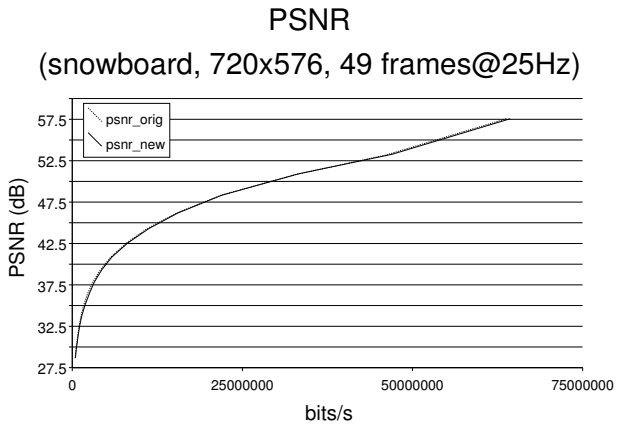

Figure 5: *PSNR vs. bitrate for 47 (*720×576*)-frames of the "snowboard" sequence.*

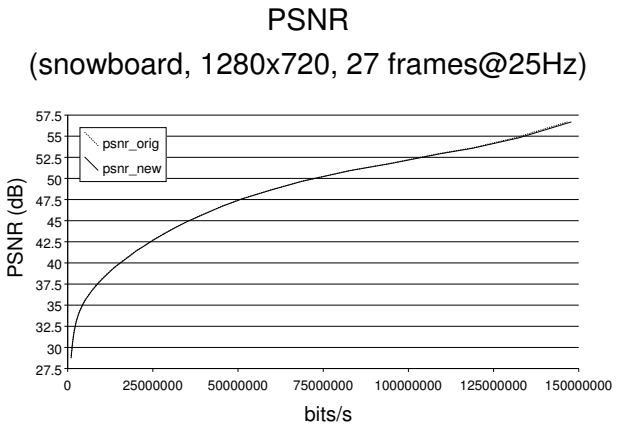

Figure 7: *PSNR vs. bitrate for 27 (*1280×720*)-frames of the "snowboard" sequence.*

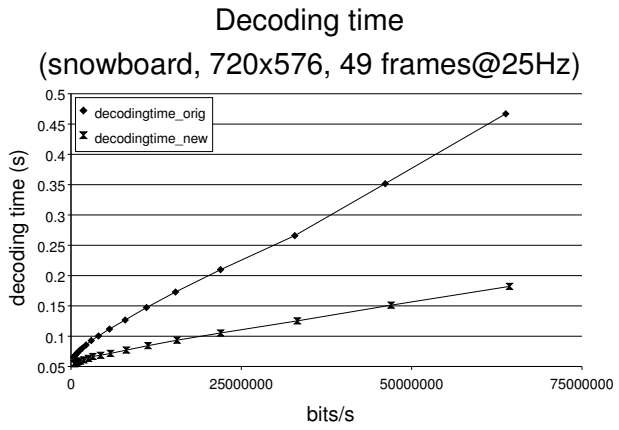

Figure 6: *Decoding time per frame vs. bitrate for 47 (*720×576*)-frames of the "snowboard" sequence.*

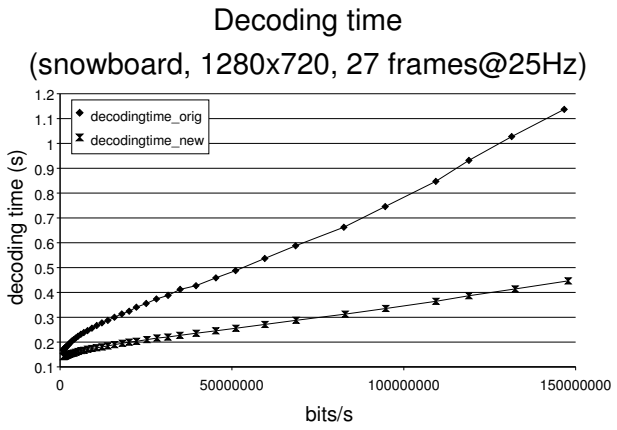

Figure 8: *Decoding time per frame vs. bitrate for 27 (*1280×720*)-frames of the "snowboard" sequence.*

5 Conclusions

Replacing the original arithmetic coder algorithm of the Dirac algorithm with an accurate configured M-coder resulted in a much faster video codec. The speedup is all the clearer as more symbols are decoded; for very high bit rates the new AD is three times faster. Because we initialised the different context models and perfected the different lookup-tables, we were able to almost retain the original compression performance even though significant approximations were introduced in the AC.

Acknowledgement This research is supported by I.W.T. grant 020174, F.W.O. grant G.0021.03, by GOA project 12.51B.02 of Ghent University.

References

[1] Davies T. The Dirac Algorithm. http://dirac.sourceforge.net/documentation/algorithm/, 2005.

[2] Bleackley P.J. Hardware for Arithmetic Coding. *Tech. rep.*, BBC R&D, 2005. URL `http://www.bbc.co.uk/rd/pubs/whp/whp112.shtml`.

[3] Marpe D., Schwarz H., and Wiegand T. Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13(7), July 2003, pp. 620–636.

[4] Pennebaker W.B., Mitchell J.L., Langdon G.G.J., and Arps R.B. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, vol. 32(6), November 1988, pp. 717–726.

[5] Pennebaker W.B. and Mitchell J.L. Probability Estimation for the Q-Coder. *IBM Journal of Research and Development*, vol. 32(6), November 1988, pp. 737–752.

[6] Mitchell J.L. and Pennebaker W.B. Optimal hardware and software arithmetic coding procedures for the Q-Coder. *IBM Journal of Research and Development*, vol. 32(6), November 1988, pp. 727–736.

[7] Taubman D.S. and Marcellin M.W. *JPEG2000, Image Compression Fundamentals, Standards and Practice*. No. ISBN 0-7923-7519-X. Kluwer Academic Publishers, Kluwer Academic Publishers Group Distribution Centre Post Office Box 322 3300 AH Dordrecht The Netherlands, 2002.

[8] Marpe D., Schwarz H., Blättermann G., Heising G., and Wiegand T. Context-Based Adaptive Binary Arithmetic Coding in JVT/H.26L. *Proc. IEEE International Conference on Image Processing (ICIP'02)*, vol. 2, September 2002, pp. 513–516.

[9] Langdon G. and Rissanen J. Compression of black-white images with arithmetic coding. *IEEE Trans. Communications*, vol. 89(6), 1981, pp. 858–867.