PERFORMANCE ANALYSIS OF MPEG-4 DECODER AND ENCODER

Fabio Cavalli[1], Rita Cucchiara[2], Massimo Piccardi[3], Andrea Prati[2]

[1] Dipartimento di Ingegneria, Università di Ferrara, Ferrara, Italy, email: fcavalli@libero.it
[2] Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia, Modena, Italy, email: {rita.cucchiara, andrea.prati}@unimo.it
[3] Department of Computer Systems, University of Technology, Sydney, Australia, email: massimo@it.uts.edu.au

Abstract: *In this paper, a performance analysis of MPEG-4 encoder and decoder programs on standard personal computer is presented. The paper first describes the MPEG-4 computational load and discusses related works, then outlines the performance analysis. Experimental results show that while the decoder program can be easily executed in real time, the encoder requires execution times in the order of seconds per frame which call for substantial optimisation to satisfy the real-time constraints.*

Key words: *Multimedia, MPEG-4, performance analysis.*

1. INTRODUCTION

Multimedia video streaming applications such as video conferencing and movie playing call for severe data compression, due the huge size of the video data and the limited available communication bandwidth. For instance, a common format like the ITU-R BT.601-4 adopted in the popular DV-NTSC camcorders uses a 720 x 480 pixels frame, 12 bpp, 30 fps with a bitrate of about 124 Mbit/s, hence still unfeasible for streaming unless substantial compression is used. The recent ISO/IEC MPEG-4 standard has introduced unprecedented levels of video compression, by adopting more efficient coding techniques than its predecessors MPEG-1 and MPEG-2 and the ITU-T H.261 and H.263 standards, together with the new concept of object-based coding. Object-based coding exploits the notion of visual entities, called Visual Objects, which can be encoded, decoded, handled separately and combined to reconstruct the scene, thus allowing for interactive management of the visual process. In addition, object-based coding allows for increased amounts of data compression, since coding is restricted to the objects of interest only.

The high level of compression typical of the MPEG-4 standard is unfortunately achieved at the cost of high computational loads for the encoding and decoding processes. These high computational loads still prevent general real-time implementations on standard computing platforms. In this paper, we present an analysis of the computational load for the decoder and encoder programs on standard personal computer, showing that real-time issues are still challenging, particularly for encoder programs. The rest of the paper is organized as follows: section 2 describes the related works. Section 3 describes the MPEG-4 encoder and decoder programs and the data sequences used for testing. Section 4 presents the experimental results and the conclusion summarises the results achieved, pointing out some potential performance improvements.

2. RELATED WORKS

Even in its draft stages, the MPEG-4 standard [1,2] has collected widespread attention in the literature for its relevance to multimedia applications. In particular, many works have addressed computational load analyses, since the heavy computational load immediately appeared as one of the major problems to solve for MPEG-4 practical application. In [3], a computational analysis of the encoder and decoder programs is proposed based on instruction-level profiling (MPEG-4 VM8.0, MoMuSys C implementation). The analysis shows that even for small-size frames (QCIF, 176 x 144 pixel) at 30 fps the decoder requires 734 MIPS and the encoder 4.5 GIPS. Instruction-level profiling is used also in [4], reporting a 27 GOPS (!) load for the encoder on the MPEG-4 Core profile, Level 2 (CIF, 352 x 288, 30 fps). In [5], a computational analysis shows that not only the computational load is very high, but it varies drastically along the frame sequence. This behavior is due to the object-based nature of coding, with the number of macroblocks per frame (MB/f) to be processed varying correspondingly with the object shapes. The irregular behavior of the computational load makes it consequently even harder to allocate adequate computational resources.
Therefore, several proposals have been made in the literature aimed at meeting the computational requirements in real time. In [3], the authors propose an architecture based on multiple specialised processors. A similar architecture is proposed also in [6], conceived for system-on-a chip implementation. In [7], algorithm optimisations are compared for the most critical task of the encoder, i.e. motion estimation; the full-search approach of the reference implementation is compared against other algorithms with different visual quality and computational efficiency. A relevant decrease in computational complexity of about one order of magnitude is reported, made it possible by more efficient algorithms [8] at the cost of only a very limited loss in visual quality. In [4], different architectures are compared on the full-search motion estimation showing that a systolic array with 64 processors can achieve a speed-up of 200 times with respect to a common RISC instruction-set architecture, while an MMX instruction-set architecture can still achieve a significant speed-up of about one order of magnitude, thanks to exploitation of the intrinsic data parallelism. In [9], an architecture able to achieve real time is proposed, yet requiring substantial dedicated hardware and limited to the encoding of binary alpha-planes. In [10], a real-time software implementation is proposed, but only for the MPEG-4 simplest profile (the Simple profile which does not support object-based coding). This literature overview shows that the problem of a general real-time implementation of MPEG-4 on standard platforms has yet to be solved.

3. MPEG-4 ENCODER AND DECODER PROGRAMS

The MPEG-4 decoder and encoder programs consist of a series of tasks (*tools*, in the MPEG-4 terminology). The encoder main tasks are:

- Motion estimation and motion compensation (ME/MC), used for inter-coded frames only;
- Computation of Discrete Cosine Transform coefficients and inverse DCT (DCT/IDCT);
- Coefficient quantization and their inverse quantization (Q/IQ);
- Variable-length coefficient coding (VLC);
- Shape encoding, needed only for object-based coding.

Inverse operations are needed as well as the direct ones since the encoder must compute the *prevision error* (the differential image for inter-coded frames) on the same image that will be

available to the decoder (i.e. the original image encoded and then decoded). The decoder main tasks are instead:

- Variable-length coefficient decoding (VLD);
- Coefficient inverse quantization (IQ) ;
- Computation of inverse DCT (IDCT);
- Motion compensation (MC), used for inter-frame coded frames only;
- Shape decoding, needed only for object-based coding.

The computational load is very different for the various tasks and is subject to change significantly with the video format and content, particularly for object-based coding. In order to enable performance evaluation over a wide variety of cases, a set of different test sequences is provided in the MPEG-4 Verification Model, including natural and hybrid natural/synthetic sequences, different levels of detail and motion, and object-based and frame-based coding [2]. Frame-based coding is still supported by MPEG-4 for all those cases where segmentation of the video sequence into visual objects is not easily feasible or convenient.

4. PERFORMANCE ANALYSIS

Performance of the encoder and decoder programs have been measured on the reference release from Microsoft, compiled with Microsoft Visual C++ 6.0 avoiding the use of any specific optimisation in order to measure reference values. The test sequences chosen are *stefan* and *brea*, sub-sampled in CIF format at 10 fps; the first sequence was coded as whole frames (frame-based coding, resulting in 396 MB/f), while the second was object-coded (230 MB/f on average).

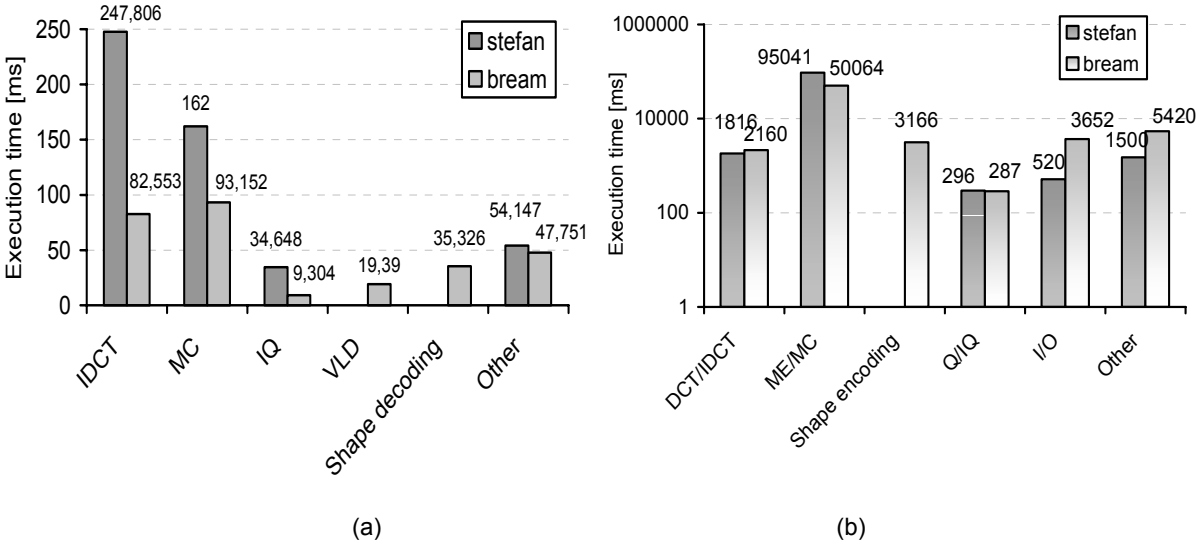

Fig. 1. (a) Decoder and (b) encoder execution times per second of video length.

Fig. 1 reports the execution times of the major decoder and encoder tasks per second of video length on a laptop computer based on a 750-MHz Intel Mobile Pentium III. For the decoder, Fig. 1.b shows that

- the overall execution time largely satisfies the real time constraints (490 ms stefan, 280 ms brea) and is approximately proportional to the number of MB/f;

- instead, the execution time of the different tasks is not strictly proportional to the number of MB/f and vary significantly with the sequence, making it difficult to perform encoding with uniform processing resources;
- the IDCT and MC are the most expensive tasks, thus calling for the highest optimisation.

The disk I/O time (not shown) needed for storing the decoded frames is not negligible with respect to the processing time. However, it can be assumed null in common applications such as for instance video conferencing and movie playing where no frame storage is required.

For the encoder, Fig. 1.b shows that

- the overall execution time (99 s stefan, 64 s brea) is enormously higher than real time; yet, again it varies significantly with the sequence, making it difficult to exploit uniform processing resources;
- the ME/MC is largely the most expensive task for both sequences, consuming most than 90% of the execution time on average; therefore, it calls for the most severe optimisations;
- the disk I/O time is largely negligible with respect to the processing time.

5. CONCLUSION

In this paper we have presented a performance analysis of MPEG-4 decoder and encoder programs on standard personal computer for frame-based and object-based sequences. Experimental results show that the decoder is a relatively light task, in the order of tens of milliseconds per frame on average, which can be executed in real time. Videos with larger formats and higher frame rates can as well be allowed real-time decoding by exploiting compiler optimisations and high-end processors. Instead, the encoder proves a very computationally heavy task, in the order of ten seconds per frame, and its real-time implementation is still challenging. Potential performance improvements can be achieved by exploiting optimisations at various levels, including lower computational complexity algorithms, instruction-level parallelism, SIMD parallelism and dedicated architectures.

REFERENCES

[1] S. Ballista, F. Casalino, C. Lande, MPEG-4: a multimedia standard for the third millennium. 1, *IEEE Multimedia*, 6(4), 1999, 74-83

[2] MPEG-4 Video Verification Model 18.0, ISO/IEC JTC1/SC29/WG11 N3908, 2001

[3] J. Kneip, S. Bauer, J. Vollmer, B. Schmale, P. Kuhn, M. Reissmann, The MPEG-4 video coding standard - a VLSI point of view, *1998 IEEE Workshop on Signal Processing Systems*, 1998, 43 –52

[4] H.C. Chang, L.G. Chen, M.Y. Hsu, Y.C. Chang, Performance analysis and architecture evaluation of MPEG-4 video codec system, *IEEE Int. Symp. on Circuits and Systems, 2000*, vol. II, 449-452

[5] P. Kuhn, W. Stechele, Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation, *Visual Comm. and Image Process.*, SPIE 3309, 1998, 498-509

[6] M. Berekovic, H.J. Stolberg, P. Pirsch, Implementing the MPEG-4 AS profile for streaming video on a SOC multimedia processor, *3rd Workshop on Media and Streaming Processors*, 2001

[7] P. Kuhn, G. Diebel, S. Hermann, A. Keil, H. Mooshofer, A. Kaup, R. Mayer, and W. Stechele, Complexity and PSNR-Comparison of Several Fast Motion Estimation Algorithms for MPEG-4, *Appl. of Digital Image Proc. XXI*, SPIE 3460, 1998, 486-499
[8] F. Kossentini and Y. Lee, Computation-Constrained Fast MPEG-2 Video Coding, *IEEE Signal Processing Letters*, vol. 4, n. 8, 1997, 224-226
[9] H.C. Chang, Y.C. Wang, M.Y. Hsu, L.G. Chen, Efficient algorithms and architectures for MPEG-4 object-based video coding, *2000 IEEE Workshop on Signal Processing Systems*, 2000, 13 –22
[10] T. Moriyoshi, H. Shinohara, T. Miyazaki, I. Kuroda, Real-time software video codec with a fast adaptive motion vector search, *1999 IEEE Workshop on Signal Processing Systems*, 1999, 44 –53