BENCHMARK THE SOFTWARE BASED MPEG-4 VIDEO CODEC[1]

Weiguo Zheng, Ishfaq Ahmad, Ming Lei Liou

wgz@cs.ust.hk, iahmad@cs.ust.hk, eeliou@ee.ust.hk

Multimedia Technology Research Center
HKUST, Clear Water Bay, Hong Kong

Abstract

Software-based implementations of H.263 and MPEG-2 video standards are well documented, recently reporting faster than or close to real-time performance. Since the complexity of MPEG-4 is higher than its predecessor standards, real time video encoding and decoding can exhaust computational resource without achieving real-time speed. In this paper, we report a software-based real-time MPEG-4 video codec (encoder and decoder) on a single-processor PC, with no frame-skip, or profile simplifying tricks, or quality loss compromise. The proposed codec is an embodiment of a number of novel algorithms. Specifically, we have designed a fast binary shape coding algorithm, a fast motion estimation algorithm, and a technique for detection of all-zero quanitzed blocks. To enhance the computation speed, we harness Intel's SIMD (Single Instruction Stream, Multiple Data Stream) instructions to implement these algorithms. On the 800 MHz Intel Pentium III, our decoder can play real-time CIF video with less than 20% system resource consumption; and our encoder realizes up to 70 frames per second for CIF resolution video, with the similar picture quality as the reference software.

1. Introduction

Due to improved coding efficiency and functionality of MPEG-4, a number of new applications using digital video, such as video conferencing, Internet video games or digital TV, are emerging on common PC as well as hand-held devices. For such application, efficient software encoding and decoding is highly desirable.

Compared to its predecessor standards, MPEG-4 has a number of additional coding modules that exacerbate its coding complexity. MPEG-4 supports arbitrary shape object based encoding described by alpha map. Multilevel alpha maps are frequently used to blend different layers of image sequences for the final film. Coding of texture for arbitrarily shaped regions is required for achieving an efficient texture representation for arbitrarily shaped objects. In addition, MPEG-4 provides multifunctional coding tools and algorithms that provide tools to support a number of content based as well as other functionalities. MPEG-4 video group has developed Video Verification Models (VMs), which has evolved by means of core experiments. The VM is a common platform with a precise definition of encoding and decoding algorithms that can be presented as tools addressing specific functionalities. New algorithms/tools are added to the VM and old algorithms/tools are replaced in the VM by successful core experiments [1].

The introduction of arbitrary shape object encoding and higher compression efficiency incur significant additional computational complexity and requirements. The speed of latest VM is still very slow and is far from practical applications. Based on our testing, the encoding speed for CIF format video (rectangular frame) is usually less than 2 frames per second on Pentium III 800 MHz processor and Windows 2000 (CIF size rectangular frame simple head and shoulder sequences).

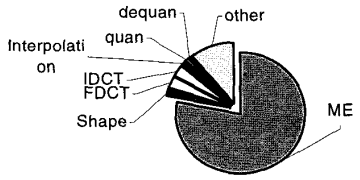

Fig.1: Complexity distribution of MPEG-4 encoder

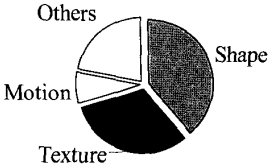

Fig.2: Complexity distribution of MPEG-4 decoder

The design of a fully standard-compliant MPEG-4 codec that can achieve real-time speed entails optimizations at all levels, including designing new algorithms for intra and inter VOP coding, software implementation with efficient data structures, and enhancing computation speed by all possible methods such as taking advantage of the machine architecture. Using profiling tools, the complexity of various modules of MPEG-4 video encoder and decoder with CIF size sequences are evaluated and depicted in Fig.1 and Fig.2. For the encoder, motion estimation occupies the largest portion of the complexity pie, while FDCT

[1] This work was supported by Research Grants Council of Hong Kong under contract # CRC98/01.EG05 and HKUST6228/99E.

and IDCT hold the second position, followed by quantization and dequantization (QUAN and DEQUAN) and shape encoding. For the decoder, IDCT, dequantization, BAB decoding and motion compensation consume most of computational resource.

In this paper, we report a software-based real-time MPEG-4 video codec on a single-processor PC, with no frame-skip, or profile simplifying tricks, or quality loss compromise. The proposed codec is an embodiment of a number of novel algorithms, such as fast binary shape coding algorithm, a fast motion estimation algorithm, and a technique for detection of all-zero quanitzed blocks, etc. To enhance the computation speed, we harness Intel's instructions to implement these algorithms.

The remainder of this paper is In Section 2 we describe the optimization of MPEG-4 decoder. Section 3 explains the optimising techniques for MPEG-4 encoder. Section 4 includes the benchmark and comparison results. Section 5 concludes the paper with some summarizing remarks.

2. Optimization for MPEG-4 Decoder

Some papers have reported how to optimize MPEG-4 decoder, but most of the reported work is for a limited range of optimizations, such as IDCT [2], [3], and without the utilization of MMX technology.

In order to identify the modules with major computationally complexity, we have profiled the reference software MPEG-4 decoder. These experiments are done with streams in the core profile@level 2. The size of GOV is 15, and there exist 4 PVOPs between IVOPs, 2 BVOPs between reference VOPs. By the order of complexity, we list the modules with major computational intensity in the following: shape decoding, texture decoding, motion compensation, VLC codeword decoding, disk access or display, etc.

2.1 Improvements in Shape Decoding

The computation of context information for pixels in the binary alpha block (BAB) is the most complex part of shape decoding. Contexts are obtained using the neighbouring pixel information. There are two types of contexts: INTRA and INTER. In the case of INTRA context computation, 10 neighbouring pixels are loaded to form a template. In the case of INTER context, 9 pixels from current and reference VOPs are loaded.

To compute a 10-bit INTRA context, we need to make 10 memory accesses for neighbouring pixels, and test if they are opaque or transparent. If a pixel is opaque, it is placed in current context location using the shift operation; otherwise, a transparent pixel is ignored.

Since the template is moved from left to right and from top to bottom, we can use three 3 64-bit registers (SIMD registers) to store their values, and keep moving in and moving out to form a new template. And use SIMD comparison and shift instructions to calculate the context. The advantage of this method is reduced memory access and improved efficiency of SIMD instruction's parallel processing.

2.2. Optimizing the Texture Decoding

The texture decoding procedure consists of decoding macroblock header, decoding DCT coefficients from VLC codeword, dequantization, and inverse DCT (IDCT), etc. According to profiling analysis, the IDCT and dequantization have the highest complexity for texture decoding.

In the dequantization processing, the VLC decoded DCT elements is multiplied by the product of quantization scale factor and the quantization matrix. To improve the speed the multiplication results can be pre-computed and stored. The quantization scale factor is used as an index to retrieve the pre-multiplication value. Thus the number of multiplication steps is reduced for each non-zero coefficient. The IDCT processing is improved by SIMD instructions and AAN algorithm.

2.3. Motion Compensation

Motion compensation includes three major functions: pixel upsampling, data copy, and pixel compensation. Pixel upsampling is necessary for half-pixel motion compensation. Pixel values are typically stored as 8-bit unsigned characters. In order to apply SIMD instruction, the upsampling has to be implemented with 8-bit quantities, and 8 upsampled pixels can be obtained in one process. In upsampling, after division by 2 (or 4), two (or four) 8-bit quantities are accumulated into a new 8-bit quantity. In order to minimize accuracy loss, a compensation value can be added to accumulated value.

Data copy also benefits from 64-bit long SIMD register. Using SIMD data move instruction, 8 pixels can be moved at the same time. Similarly, pixel compensation can take benefit from SIMD processing. The problem arises when the data types are unmatched during the addition of the delta data to the reference VOP. Pixel values in reference VOP are stored as 8-bit quantities but the delta data from IDCT are 16-bit quantities. Before the addition, the 8-bit reference pixels are unpacked into 16-bit quantities, and added with 16-bit delta data. Then, we pack the 16-bit quantities back to 8-bit with saturation. The motion compensation can be executed in a similar fashion.

3. Optimization for MPEG-4 Encoder

Compared to the decoder, the optimisation of MPEG-4 encoder is much more difficult due to several additional operations such as motion estimation, FDCT, QUANT, and mode decision, etc. However, implementation of the encoder has more room for improvement since the

standard only defines the syntax and methods for decoder and is more flexible for the encoder.

According to profiling results, the most computational intensive module is motion estimation, which is followed by FDCT/IDCT, QUAN/DEQUAN and shape encoding. The optimization for shape encoding is similar with what have been done for decoder. This section focuses on fast motion estimation and prediction of all-zero block.

3.1. Fast Motion Estimation

Motion estimation (ME) is the most important part of the MPEG-4 encoder, since it could significantly affect the output quality of the encoded sequence. This is also the most complex part with an overwhelming computational complexity compared with the other parts of the encoding process. In our testing, with ±16 search window, motion estimation with full search can swallow more than 90% of processing resource. A myriad of algorithms exist to improve the speed and performance of motion estimation, such as the three-step search, new three-step search, 2-D logarithmic search, conjugate directional search and hierarchical search [4], [5], [6], etc. The MPEG-4 Part-7 has adopted MVFAST (motion vector field adaptive fast search technique) as the core technology for fast motion estimation [7]. MVFAST can achieve substantial speedup comparing with full search. However, more efficient motion estimation is possible. We propose a new algorithm, called Adaptive Motion Search with Elastic Diamond (AMSED), which is considerably faster than MVFAST but yields the same picture quality. The main features of AMSED are:

- Adaptive threshold for stationary block;
- Definition of motion vector candidate list (MVCL);
- Detection of MB's motion difference (MD);
- Motion search with elastic diamond search pattern;
- Adaptive threshold for half-way-stop;
- Keeping the checking point history;
- Adoption motion inertia in temporal domain;
- Interpolated motion vector in motion candidate list for BVOP.

A stationary MB is the one with its motion vector at (0, 0). In MVFAST [7], the threshold is 512. In AMSED, this threshold is set adaptively according to motion of its neighbours. This allows faster and more accurate detection of stationary block.

The motion difference (MD) is determined from a list of candidate vectors. The motion vector candidate list (MVCL) includes motion vectors from adjacent MBs in spatial and temporal domain. Based on the observation of smoothness of motion field, MD is measured by calculating the maximum difference between vectors in MVCL. If the MBs in MVCL belong to the same moving object, MD is usually low, and the motion vector can be refined within a small range around the average vector. If MD is larger than L (L = 2 in this paper), AMSED uses the motion inertia property by testing MVs within the search range in the reference VOP; the closest predictor (MB position + MV) is selected as the MV candidate from the temporal domain.

AMSED uses two search patterns: Large Diamond Search Pattern (LDSP) and Small Diamond Search Pattern (SDSP). LDSP is used to find raw motion vector quickly, and SDSP is used to refine motion vector prediction. In the elastic mode, LDSP and SDSP may switch between each other. If SDSP is executed for a certain number of times, the search pattern changes to LDSP. If the center has the minimum SAD in the current round of LDSP, the search pattern changes to SDSP with the same search center, and so on.

In order to eliminate duplicated checkpoints, AMSED keeps track of the checking points that have been accessed. An adaptive half-way-stop threshold is applied to terminate the motion search once a good enough motion vector is obtained.

3.2. Prediction of All-Zero Blocks

In MC/DCT framework, forward DCT helps in removing the spatial redundancy by concentrating most relevant information to the lower coefficients in the frequency domain. Quantization is basically a process for reducing the precision of the DCT coefficients. The quantization process involves division of integer DCT coefficient value by integer quantization scales, which is chosen to minimize the perceived distortion in the reconstructed picture using the principles based on the human visual system. In practice, most quantized DCT coefficients become zero, with many blocks having all of their coefficients become zero. We refer to such a block as *all-zero block*. If such blocks can be detected prior to DCT and quantization, the associated dequantization and IDCT can be skipped as well. This can result in significant saving in the computational cost.

As opposed to the techniques proposed in [8] and [9], we combine the properties of DCT transformation and quantization of H.263 and MPEG-2 mode, respectively. The sum of absolute values (SAV) is defined for an 8×8 block that is going to be transformed and quantized.

$$SAV = \sum_{i=0}^{7} \sum_{j=0}^{7} |p(i, j)| \qquad (1)$$

We use the following criteria to predict all-zero blocks in inter-coded blocks:

- For H.263 quantization for non-intra block, if SAV < 20Q, this block is determined to be all zero block;
- For MPEG quantization for non-intra block, if SAV < 16Q, this block is determined to be all zero block.

Based on our experiments, we observe more than half of the blocks in PVOP and BVOP are skipped as all zero blocks.

4. Performance Comparison

We have tested our decoder and encoder on various processors. The results reported here are based on an Intel Pentium III 800 MHz processor running Windows 2000. The decoding speeds of reference decoder and our optimised decoder are compared in Table 1. For the encoder, the first VOP is IVOP, and the rest of the VOPs are predicted. There are two BVOPs between reference VOPs. The quantization scales are fixed for different type of VOPs; they are 10 for IVOP, 12 for PVOP and 16 for BVOP. We compare both the compression ratios and encoding frame rates (no rate control is used in these experiments). The experiment results are shown in Table 2.

5. Conclusions

In addition to the techniques mentioned above, we have also improved 4-MV motion estimation, half-pixel motion estimation, motion compensation, best-bounding box finding, upsampling, fast DCT & IDCT, and SIMD implementation, etc. Based on the proposed techniques in this paper, the optimized decoder can achieve real-time speed with less than 20% computational resource consumption; and our software based MPEG-4 encoder can encode CIF resolution video at faster than real-time speed without loss of quality and compression efficiency.

References

[1] ISO/IEC JTC1/SC29/WG11 N3093, "MPEG-4 Video Verification Model Version 15.0."

[2] Franco Casalino, Gianluca Di Cagno, and Ronco Luca, "MPEG-4 Video Decoder Optimization," *Proceeding of IEEE International Conference on Multimedia Computing and Systems,* Vol.1, 1999, pp. 363-442.

[3] Lap-Pui Chau, Nam Ling, G. Hovden, H. Lan, Hon-Cheong Ng, and Keng-Pang Lim, "An MPEG-4 Real-time Video Decoder Software," *Proceeding of ICIP'99,* Vol.1, 1999, pp. 249-253.

[4] R.Li, B.Zeng, and M.L.Liou, "A New Three-Step Search Algorithm for Fast Motion Estimation," *IEEE Transactions on Circuits & Systems for Video Technology*, Vol.4, Aug. 1994, pp. 438-442.

[5] F.Dufaux and F.Moscheni, "Motion Estimation Techniques for Digital TV: A Review and a New Contribution," *Proceeding of IEEE*, vol. 83, June 1995, pp. 858-876.

[6] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, and Ashraf Ali Kassim, "A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits & Systems for Video Technology*, Vol.8, No.4, August, 1998, pp. 369-377.

[7] ISO/IEC JTC1/SC29/WG11 N3324, "Optimization Model Version 1.0."

[8] I_Ming Pao, and Ming-Ting Sun, "Modeling DCT Coefficients for Fast Video Encoding," *IEEE Transactions on Circuits & Systems for Video Technology*, Vol.9, No.4, June 1999, pp. 608-616.

[9] Anurag Bist, Wei Wu and Albert Hsueh, "Intelligent Pre-Quantization in Motion compensated Video Coding," *ITU-T Q15-D-35*, Tampere, Finland, April 1998.

Table 1: Comparison of the reference MPEG-4 VM video decoder and our optimized decoder for core profile @ level 2 (2 BVOPs between IVOPs or PVOPs)

Sequence Name	Sequence Type	# of Frames	Speed (fps)	
			VM	Optimized
Akiyo	Rectangular	300	47	208
Foreman	Rectangular	300	39	151
News2	Arbitrary	300	25	173
Stefan	Rectangular	300	37	148

Table 2: Comparison of the reference MPEG-4 VM video encoder and our optimized encoder for core profile @ level 2 (2 BVOPs between IVOPs or PVOPs)

Sequence Name	Sequence Type	# of Frames	Comp. Ratio		PSNR (Y) (dB)		Speed (fps)	
			VM	Optimized	VM	Optimized	VM	Optimized
Akiyo	Rectangular	300	730	796	36.08	35.94	1.93	71.52
Foreman	Rectangular	300	111	114	32.24	32.18	0.55	31.66
News2	Arbitrary	300	537	557	31.10	30.97	2.07	40.76
Stefan	Rectangular	300	43	38	29.57	29.49	0.47	29.96