

MPEG-4 Video decoder optimization

Franco Casalino¹, Gianluca Di Cagno¹, Ronco Luca¹

¹ CSELT Centro Studi e Laboratori Telecomunicazioni S.p.A

10148 Torino Italy Via Reiss Romoli, 274 TEL: +39 11 2285100

E-Mail: Franco.Casalino@cse.lt Gianluca.DiCagno@cse.lt Luca.Ronco@cse.lt

Abstract

After a decade from its origin MPEG, with its current MPEG-4 project, is now facing the challenge of providing a future-proof multimedia toolkit which aims at incorporating new and emerging technologies while ensuring backward compatibility with its previous and successful audio-visual standards. This paper provides an overview of the standard focusing mainly on the video decoding aspects that, by their nature, represent one of the most peculiar features of the future specifications which are scheduled to become an International Standard by the beginning of year 1999. The paper first briefly introduces the MPEG standards focusing on the MPEG-4 Video part of the specification giving an extensive presentation of a coding/decoding video process. The main part of this paper is focused on the optimization performed to realize a real time software video decoder based on the MMX™ architecture compliant to the VM 8 MPEG-4 Video standard. A comparison with the reference video decoder is also presented.

1. MPEG Overview

The Motion Picture Expert Group (MPEG) has successfully introduced two standards for coding of audiovisual information, known by the acronyms MPEG-1 [1] and MPEG-2 [2]. The first addresses the storage of audiovisual information on CD-ROM, whereas the second handles the generic coding of digital TV and HDTV signals. The most important goal of both the MPEG-1 and MPEG-2 standards was to make the storage and transmission of digital audio visual material more efficient, through the use of compression techniques. To achieve this, both deal with frame-based video and audio. Interaction with the content is limited to the video frame level, with its associated audio.

The new MPEG-4 standard goes beyond these goals by specifying a description of digital audiovisual (AV) scenes in the form of 'AV objects' that have certain relations in space and time. Starting from this structure, MPEG-4 will:

- offer a new kind of interactivity (with each AV object, and at the levels of coding, decoding, or objects

composition), integrate objects of different natures (e.g. natural video, graphics, text, ...).

- Moreover, MPEG-4 will allow 'universal access' to multimedia information, by taking into account the specific details of a wide variety of networks.

MPEG-4, started in July 1993, has reached Committee Draft level in November 1997 and will reach International Standard level in January 1999. As for the earlier standards, the structure of the MPEG-4 IS will include the following main parts: 14496-1 System [4], 14496-2 Visual [5], 14496-3 Audio [6], 14496-4 Conformance Testing, 14496-5 Technical Report.

2. MPEG-4 standard

The selected key features of the MPEG-4 standard are the following:

MPEG-4 Visual [5] provides a natural video coding algorithm that is capable of operating from 5 kbit/s upwards to transparent quality. One important feature of the Visual part of MPEG-4 is the ability to code not just a rectangular array of pixels, but also "object" with arbitrary shape, such as a walking person or a moving car in a scene. The object can also be of synthetic origin. MPEG-4 Visual is currently supporting the definition of a synthetic human face through 68 feature point of a face. The freedom of individually coded objects, however, entails the need for a standard way to position objects in a scene (this applies to both video and audio objects). Thus a creator of the scene is the MPEG-4 equivalent of a movie director who guides set-up of a scene, positions the actors and directs the dialog.

MPEG-4 Audio [6] covers the complete bitrate range from 2 to 64 kbit/s per channel. Acceptable speech quality is obtained already at 2kbit/s and transparent quality of monophonic music sampled at 48 kHz at 16 bits per sample is obtained at 64 kbit/s. Appealing quality is obtained at 16 kbit/s for stereo music. 24 kbit/s is the bitrate used by Digital Radio Mondiale in lieu of conventional AM radio. Scalable audio is supported for Internet applications with uncertain access bandwidth.

The “ object composition” technology is provided by *MPEG-4 Systems* [4]. It provides precise synchronization of audio and video objects. MPEG-4 Systems supports both push and pull delivery of content. MPEG-4 also has support for Objects Content Identification (OCI) so that searches in databases of MPEG-4 objects becomes possible. To accommodate the needs of the content industry each MPEG-4 audio, visual and audio-visual object can be identified by a registration number similar to the well established International Standard Recording Code (ISRC) now used to identify pieces of music in Compact Disc Audio.

The abstraction of the application audio layer from the transport technology (an interactive network, a broadcast channel or a local disk) is the function of *MPEG-4 DMIF* [7] (Delivery Multimedia Integration Framework). It enables application developers to develop applications with the assurance that their investment will not be made obsolete by new delivery technologies.

3. MPEG-4 Video

3.1. MPEG-4 Video Encoder and decoder Structure

In MPEG-4 [5], the concept of Video Object (VO) and Video Object Plane (VOP) have been introduced. A video object plane represent instances in time of a given video object. Both VOs and VOPs correspond to entities in the bitstream that a user can access and manipulate. The VOP can have arbitrary shape.

Figure 1 shows the block diagram of the MPEG-4 video encoder/decoder. The most important feature of this encoder is the intrinsic representation based on VO when defining a visual scene. In fact, a user, or an intelligent algorithm may choose to encode the different VOs composing a source data with different parameters, different coding methods, or may even choose not to code some of them at all.

In most applications, each VO represents a semantically meaningful object in the scene. In order to maintain a certain compatibility with available video materials, each uncompressed VO is represented as a set of Y, U, and V components, plus information about its shape, stored frame after frame at predefined temporal intervals. It is important to note that although in MPEG-4 video test sequences the VOs were either known by construction of the sequences (hybrid sequences based on blue screen composition or synthetic sequences) or were defined by semi-automatic segmentation, this is done only due to practical considerations.

The same approach remains valid for any objects (synthetic or natural) and any dimensionality (2D or 3D), and even for any representation model used per VO.

Another important feature of the Video standard is that no temporal frame rate is explicitly defined by this approach. This means that the encoder and decoder can therefore function in different frame rates which do not even need to be constant throughout the video sequence (or the same for the various video objects).

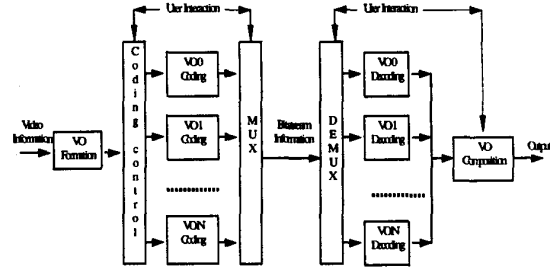


Figure 1: General structure of the encoder/decoder in the MPEG-4 Video VM.

Interactivity between the user and the encoder or the decoder can be conducted in different ways. The user may decide to interact at the encoding level, either in coding control to, for instance, distribute the available bitrate between different video objects, or to influence the multiplexing to change parameters such as the composition script at the encoder. In cases where no back channel is available, or when the compressed bitstream already exists, the user may interact with the decoder by either acting on the compositor to change the position of a video object or its display depth order. The user can also influence the decoding at the receiving terminal, by requesting the processing of a portion of the bitstream only, such as the shape.

The structure of the decoder is similar to that of the encoder, except in reverse, apart from the composition block at the end. The exact method of composition (blending) of different video objects depends on the application and the method of multiplexing used at the systems level.

3.2. MPEG-4 Video functionalities

MPEG-4 video supports eight key functionalities [3] than can be divided into three non-orthogonal classes, based on the requirements that they support:

Content-based interactivity: This class includes four functionalities focused on requirements for applications involving some form of interactivity between the user and the data, namely: content-based multimedia data access tools, content based manipulation and bitstream editing, hybrid natural and synthetic data coding, and improved temporal random access. Applications benefiting from these functionalities include: data retrieval from on-line

libraries, interactive home shopping, and movie production and editing.

Compression: This class is composed of two functionalities: improved coding efficiency and coding of multiple concurrent data streams. These essentially aim at applications requiring efficient storage or transmission of audiovisual information and their effective synchronisation. These functionalities will enhance some existing applications such as information browsing over Internet and virtual reality.

Universal access: The remaining two functionalities are: robustness in error-prone environments and content-based scalability. These functionalities allow MPEG-4 encoded data to be accessible over a wide range of media, and with various qualities in terms of temporal and spatial resolutions for specific objects, which could be decoded by a range of decoders with different complexities. Applications benefiting from these functionalities are: wireless communications, database browsing with access at different content levels, scales, resolutions, and qualities.

3.3. VOP encoder and decoder structure

Figure 2 presents the structure of the VOP encoder. In a given session, the same encoding scheme is applied when coding all the VOPs. The encoder is composed of two main parts: shape coding and the traditional motion and texture coding applied to the same VOP. Texture coding as well as motion estimation and compensation parts of the encoder are similar, in principle, to those used in other state-of-the-art standards, where care has been taken in order to extend their respective tools to objects of arbitrary shape. The truly novel component in the coding scheme is that of shape coding. Although the coding algorithm of the video standard is designed to handle arbitrarily shaped objects, all current tools are based on the concept of a macroblock. This allows a certain compatibility with other standards and a more straightforward insertion of tools developed for such environments. In order to take advantage of both macroblock and other arbitrarily shaped object coding tools, the multiplexing of the bitstream generated from the coding tools can be made in separate or combined motion-shape-texture modes.

In the combined motion-shape-texture mode, the bitstreams of all of these features are combined together on a macroblock basis and put in the final bitstream macroblock after macroblock, for each VOP.

In separate motion-shape-texture mode, the bitstreams generated for motion, shape and texture of the entire VOP are multiplexed in the final bitstream, each occupying a contiguous portion of the bitstream. Moreover, in applications where the shape information is not required

(for example a classical rectangular frame based video coding), shape coding can be disabled. For very low bitrate applications, where blocking effect may occur, a deblocking filter may be used in the coding loop.

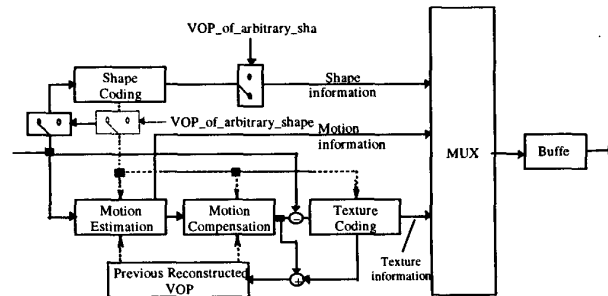


Figure 2: Video Object Plane based encoder structure in MPEG-4 Video VM.

Figure 3 presents the structure of the VOP decoder. In a given session, the same decoding scheme is applied when coding all the VOPs. The decoder is mainly composed of two parts: the shape decoder and the traditional motion & texture decoder. The reconstructed VOP is obtained by the correct combination of the shape, texture and motion information.

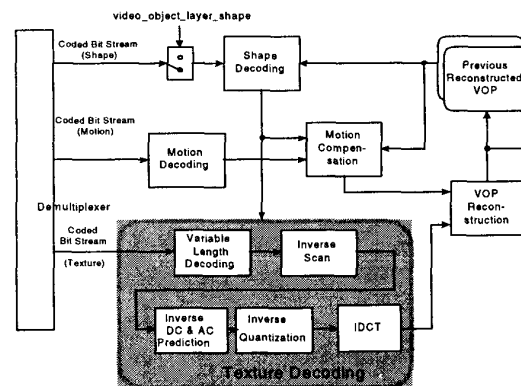


Figure 3: Video Object Plane based decoder structure in MPEG-4 Video VM.

4. MPEG-4 Video Decoder Optimization

At the beginning of this work of optimization, the reference software currently available was considered. The standard was in the Committee Draft status. At that time two different decoders were available as reference software: Microsoft and MoMuSys. The objective of the reference software is to validate the concepts introduced by the standard and to offer tools (e.g. encoders) to test

the goodness of one's implementation. Therefore performance is not required as it's not a primary target.

For this reason a study about the realization of a MPEG-4 optimized video decoder is under development. One generic optimization has been performed within the ACTS Emphasis project. Emphasis is a generically (i.e. platform independent) C optimized software and our work started taking the outcome of the Emphasis project as a starting point, being focused over the development of a dependent machine version of this software based on the MMX™ [10] architecture.

A preliminary study about the performance of the two reference software and the generic optimized one has been performed. Decoders have been time-profiled to compare their performance. Tests have been performed only over the core decoding process, leaving out rendering and presentation. Result of these tests have been achieved on a PC Pentium II MMX™ dual processor, running at 266 MHz, equipped with 64 Mbytes of memory, with Windows NT. Table 1 gives the performance measured in frames/s, and covers sequences of CIF, QCIF both shaped and non-shaped images. The measurement is made on sequences of 200 images, for every sequence the test are done 50 times and averaged to minimize the non deterministic effects of processor cache, process scheduler, and operative system management operations.

| Seq. | Img | VO shape | Option | Mic. | Emp. | Mom. |
|---------|------|----------|--------|------|------|------|
| Hall1 | QCIF | Rect. | None | 85 | 152 | 33 |
| Stephan | CIF | Rect. | OBMC | 22 | 39 | 8 |
| Stephan | CIF | Binary | None | 31 | 55 | 12 |
| Stephan | CIF | Binary | OBMC | 28 | 50 | 13 |
| Aki | QCIF | Binary | None | 173 | 323 | 67 |

Table 1: Preliminary study showing the performance (fps) of the three decoders before optimization.

5. MPEG-4 Decoder: Optimization techniques and performance results

The video decoder has been time-profiled and the results were used to drive the optimization of the modules. Optimization involved use of techniques to the exploit the parallelism of the MMX™ architecture.

5.1. MMX™ optimization techniques

Multimedia applications must process large set of data and Intel's MMX™ technology brings more power to these types of applications. In fact MMX™ technology adds new data types and a set of basic general set of integer instructions that can process data in parallel.

The highlights of the technology are:

- Single Instruction, Multiple Data (SIMD) technique

- 57 new instructions
- Eight 64-bit wide MMX™ registers
- Four new data types

The principal data type of the MMX™ instruction set is the packet, fixed point integer, where multiple integer word are grouped into a single 64-bit quantity. These 64-bit quantities can be moved into the 64-bit MMX™ registers and processed with a single instruction, providing parallelism, that greatly increases performance.

Arithmetic and logical instructions are designed to support the different packed integer data types. In graphic and multimedia applications recurring operations on small data types are executed very often.

In Mpeg-4 video decoding, graphic pixel data are generally represented in 8-bit integers, or bytes. Now, with MMX™ technology, eight of these pixels are packed together in a 64-bit quantity and moved into an MMX™ register.

When an MMX™ instructions executes, it takes all eight of the pixel values at once from the MMX™ register, performs the arithmetic or logical operation on all eight elements in parallel, and writes the result into an MMX™ register.

MMX™ optimizations of motion compensation (calculus of prediction and recombination) are based over these capabilities. In fact motion vectors used in motion compensation can address to half pixel position, like presented in Figure 4.

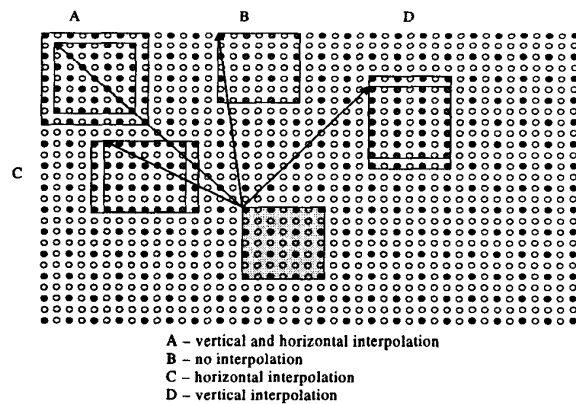


Figure 4: Examples of block matching.

In this case bilinear interpolation (block matching) is required to set pixels value of 8x8 or 16x16 predicted block. Powerful functions, with MMX™ instructions, perform vertical, horizontal or horizontal and vertical filtering to obtain eight pixels value at once.

In the same way, during the recombination process, functions can add a line of 8x8 prediction to a

correspondent line of 8x8 delta IDCT block (the prediction error), to obtain eight reconstructed pixels at once. In these operations, addition of prediction and prediction error is performed with 16-bits per pixel precision and then the value of pixels must be saturated to 8 bits. The use of MMX™ instructions allows an automatic and low cost saturation of values of pixels to a eight bit range.

The same idea can be applied to motion compensation of B-VOPs to generate prediction block, averaging, in bi-directional mode, the forward and backward prediction blocks as showed in Figure 5.

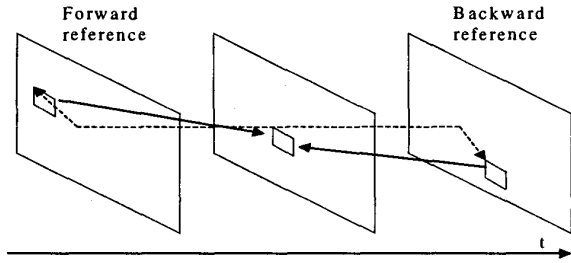


Figure 5 : Bi-directional mode prediction

Other increase of performance can be obtained during the execution of the Inverse Discrete Cosine Transform (iDCT) calculus. In fact, like in MPEG-1 and MPEG-2, 8x8 iDCT is used to perform texture decoding. Moreover, IDCT is successfully used in gray scale shape decoding. For these reasons important enhancement of performance can be achieved by the implementation of a new iDCT algorithm that is based on MMX™ parallel elaboration. The experience gained in MPEG decoding has determined the choice of Chen's algorithm because of its homogeneity as starting base to the new MMX™ algorithm. As known, in MPEG texture decoding a bi-dimensional IDCT is required.

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v)F(u, v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

Chen's algorithm allows bi-directional iDCT with a little number of operations (16 multiplications and 26 additions). Since in MMX™ cost of multiplication is cheap, the new fast algorithm is based directly onto the vectorial calculus of mono-dimensional iDCT. So the first equation can be decomposed in two parts:

In this way the bi-dimensional iDCT is modified in a mono-dimensional iDCT performed on every column, followed by another mono-dimensional iDCT valued on rows of partial results.

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 C(u) \cos \frac{(2x+1)u\pi}{16} \sum_{v=0}^7 C(v)F(u, v) \cos \frac{(2y+1)v\pi}{16}$$

The developed algorithm is organized in two steps: first the iDCT is performed on the columns of the input matrix getting two rows of the intermediate matrix for every iteration. At the second step every iteration produces two rows by applying the iDCT on rows of partial results.

Figure 6 explains this process. Using immediately the two symmetrical rows gained in every iterative cycle at the end of every step, it's possible to compute the final matrix elements without storing intermediate resulting. This allows better performance because of the slowness of the memory access. Moreover the use of $T \cdot C^T$ instead of $C \cdot T^T$ in the second step allow direct use symmetrical rows of T matrix without transposition of T matrix but using well known transposed C matrix that can be pre-calculated.

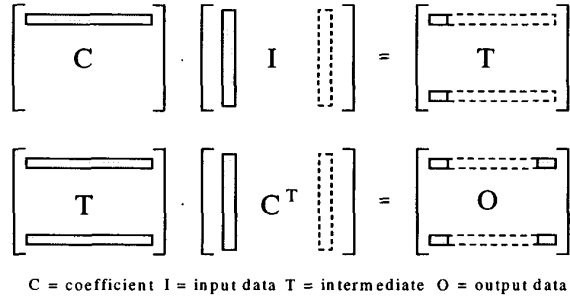


Figure 6: iDCT matrix product.

Particular relations between operands of vectorial product, allows reducing to a half the number of products and sums in every cycle. Moreover the capability to perform four operations in parallel reduces the iDCT computation of a 8x8 block to only four operations.

This new MMX™ algorithm takes a lot of care in precision of intermediate calculi so that the output results verify the limits specified in IEEE Std 1180-1990, IEEE Standard Specification for the Implementations of 8x8 Inverse Discrete Cosine Transform.

For this reason intermediate calculi are performed with 16-bits per pixel precision and then the value of pixels is saturated to 8 bits. Like in recombination process, the use of MMX™ instructions allows an automatic and low cost saturation of values of pixels to a eight bit range.

5.2. Decoder test and performance results

We paid a lot of attention in application the of Pentium MMX pairing rules.

In fact, pipelines U and V in Pentium processors are not typically maintained at their maximum capacity because of data dependency register contention, or other restriction imposed by the processor. These restrictions

are known as the Pentium pairing rules that are the internal processor guidelines that must be followed in order to execute two instructions in the same clock: one instruction executes in the U pipe, and the next one executes in the V pipe. In addition to general rules that apply to Pentium and Pentium Pro processor, there are new pairing rules that only applies to MMX instructions.

Moreover there are also rules that apply to MMX instructions scheduling. These scheduling rules are the guidelines that indicate the numbers of clocks it takes to execute certain instructions or when you can perform certain operations. Basically these rules indicate when the data is ready after certain operations [10].

A lot of benefit can be obtained from scheduling and pairing instructions, especially if the data is close to the processor (basically, in register or L1 or L2 cache).

MMXTM optimized functions have been grouped and time profiled independently from the rest of project, to compare their performance between themselves and with these before optimization. Execution times were obtained by calculi involving the processor's time-stamp, extracted by utilizing the assembler instruction RDTSC (Read Time-Stamp Counter), and the processor frequency. This method permits more precision and granularity that standard C timing functions.

Tests have been performed with different data types (i.e. range and memory alignment) and repeated more times in sequence to minimize the non-deterministic effects already mentioned.

| Seq. | Img | VO | Option | noMMX | MMX |
|---------|------|--------|--------|-----------|-----------|
| Hall1 | QCIF | Rect. | None | 152 (fps) | 190 (fps) |
| Stephan | CIF | Rect. | OBMC | 39 (fps) | 50 (fps) |
| Stephan | CIF | Binary | None | 55 (fps) | 72 (fps) |
| Stephan | CIF | Binary | OBMC | 50 (fps) | 62 (fps) |
| Aki | QCIF | Binary | None | 323 (fps) | 405 (fps) |

Table 2: performance of the Emphasis decoder before and after MMXTM optimization.

Tests were performed on this MMXTM optimized decoder and Table 2 gives the speed in frames/s, and covered sequences with CIF, QCIF and shaped images. In the same way as done at the beginning of this work, measurement was made on sequences of 200 images and for every sequence, results were averaged on 50 times. Analyzing Table 2, we can note a general increase of performance between 25% to 30%; this improvement depends from several factors such as video object format, presence of shape, use of particular types of motion-compensation, and more.

6. Conclusions

In this work we presented the MPEG-4 Video new features and we described a optimized video decoder implementation based on the MMXTM architecture.

The good performance obtained by this software video decoder has permitted to realize a software video decoder module that has been integrated in the framework of the System Reference Software Architecture [8,9]. The result was shown in Atlantic City at the 45th MPEG meeting, where world's first successful real-time display of MPEG-4 content via satellite was demonstrated.

7. Acknowledgement

We would like to thank all the people involved in the ACTS European project MoMuSys and Emphasis mentioned above since our work started from their results.

12. References

- [1] MPEG-1 (ISO/IEC 11172), "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s", 1993.
- [2] MPEG-2 (ISO/IEC 13818), "Generic Coding of Moving Pictures and Associated Audio", 1995.
- [3] VRML (ISO/IEC DIS 14772-1), "Virtual Reality Modeling Language", April 1997.
- [4] MPEG-4 Systems FCD document (14496-1), WG11, doc N2201, Tokyo Mar 1998.
- [5] MPEG-4 Video FCD document (14496-2), WG11, doc N2202, Tokyo Mar 1998.
- [6] MPEG-4 Audio FCD document (14496-3), WG11, doc N2203, Tokyo Mar 1998
- [7] MPEG-4 DMIF FCD document (14496-6), WG11, doc N2206, Tokyo Mar 1998
- [8] ISO/IEC JTC1/SC29/WG11/M3111, APIs for Systems VM Implementation 1, March 98
- [9] ISO/IEC JTC1/SC29/WG11/M3301, IM-1 2D platform ver.2.7, March 98
- [10] DirectX, RDX,RSX and MMXTM Technology, Rohan Coelho and Maher Hawash, Addison-Wesley Developers Press