INSA Rennes - Département EII

Parcours recherche 2017-2018

Rapport bibliographique

Low level optimizations of the *Future Video Coding* inverse transforms

Étudiant:
Vincent GIRAUD

Encadrants:
Pierre-Loup Cabarat
Wassim Hamidouche

Présenté le 18/12/2017

December 18, 2017

Contents

Abstract

This report presents studies on potential optimizations that might help increase the Future Video Coding (FVC) standard's efficiency on the decoding side. It focuses particularly on the inverse transform of residual data. With H.265/MPEG-4 HEVC, their processing became much more dynamic as the transform and the blocks' size can vary according to context. More specifically, this paper provides investigations results on which low level techniques for x86 SIMD architectures can be applied to accelerate residues decoding in FVC.

The Future Video Coding standard is currently under development by the Joint Video Experts Team (JVET), a committee composed of people from both the Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG). It is expected to be released by 2020. The goal is to reduce the bit rate by 50 % for an equal subjective video quality compared to H.265/MPEG-4 HEVC; while anticipating future multimedia needs.

Index terms— Discrete Cosine Transform (DCT), Future Video Coding (FVC), inverse transform design, prediction residues, Single Instruction on Multiple Data (SIMD)

1 Introduction

1.1 Digital video coding historic development

During the 2000s, digital video supports displaced the analog ones. Moreover, common screens resolution is getting bigger and bigger. The development of the video coding standards has been needed in order to keep video files' sizes manageable, while combining a correct restitution with affordable coding and decoding times.

Two organizations play an important role in this : the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), as they work on defining standards for video coding. The first one released H.261[1] and H.263[2] while the second did MPEG-1[3] and MPEG-4 Visual[4]. They also collaborated on several standards : H.262/MPEG-2 Video[5], H.264/MPEG-4 Advanced Video Coding (AVC)[6] and H.265/MPEG-4 High Efficiency Video Coding (HEVC)[7].

In the last few years, both the ITU-T VCEG and the ISO/IEC MPEG started to look forward to the next video coding standardization project, which code name is Future Video Coding (FVC). It would introduce new features and internal improvements, while keeping the foundations of the MPEG-4 standards. The intended goal is to reduce by half the bit rate at the same subjective quality, compared to H.265/MPEG-4 HEVC[8].

1.2 Structure of this report

This paper is organized as follows. Section 2 summarizes the state of the art on the subject of residues decoding in recent standards. Section 3 investigates low level methods we could develop in the FVC to push residues efficiency even beyond. Section 4 contains experiments done in order to analyze the impact of such methods. Finally, section 5 presents the conclusion.

2 State of the art

2.1 Usual video coding techniques

In order to reduce bitrate requirements while keeping image fidelity, one can use entropy encoding. This technique allows data lossless compression by exploiting the source's statistics. This algorithm does not interpret information and does not exploit the video nature of the data.

Another technique is to use differential coding, where data is defined according to other data in space or time. This principle was first used in the 1950s with Differential Pulse-Code Modulation (DPCM)[9], and first applied in video coding in the 1980s with H.120[10].

Current video compression standards such as H.265/MPEG-4 HEVC use both : differential coding with interpicture and intrapicture prediction, along with Context-Adaptive Binary Arithmetic Coding (CABAC) on the resulting data[11]. However, the prediction techniques are often not sufficient to reproduce the original video faithfully.

2.2 Residual samples management

To compensate for the loss induced by the prediction process, residues are also memorized. They are the difference between original frames and the output of the differential prediction. However, the amount of residual data is often significant; which can result in large video files sizes. The residual samples will have to be processed with multiple methods to conciliate proper restitution with small amounts of data.

During encoding, frames are divided in smaller blocks. This leads to more opportunities for parallel processing, and better correlation of data within a block. Also, calculations are thus done with matrices and vectors whose sizes are more fit with most processors' internal data bus' sizes. Transform blocks' sizes are static in H.264/MPEG-4 AVC and before[12]; whereas in H.265/MPEG-4 HEVC, each channel of a frame is divided in Coding Tree Blocks (CTB), then in Coding Blocks (CB), then possibly in Transform Blocks (TB). In the end, this means residues can work on 4×4; 8×8; 16×16; or 32×32 samples blocks[11].

2.2.1 The Discrete Cosine Transform (DCT)

To compress blocks of residual data, one can transform them into the frequency space. Switching to this domain helps decorrelate the values; especially if the prediction failed on a non-complex area of the video, such as a monochrome background. In this instance, the transform's output will likely only have values in the low frequencies, to represent the continuous component of the signal. This process is usually done with tools like the Discrete Fourier Transform (DFT) :

$$X_k = \sum_{n=0}^{N-1} x_n . e^{\frac{-i2\pi kn}{N}} = \sum_{n=0}^{N-1} x_n . [\cos(\frac{2\pi kn}{N}) - i . \sin(\frac{2\pi kn}{N})]$$

Where n is the index in the spatial domain, $k = 0, ..., N-1$ the index in the frequency space, and N the total number of samples. However, this transform produces complex numbers. This is partly why it is not used in practice in video coding.

All video coding standards between H.261 and H.265/MPEG-4 HEVC rely essentially on the Discrete Cosine Transform (DCT) for residues processing[12]. This transform results in real numbers and is defined as :

$$X_k = \sum_{n=0}^{N-1} x_n . \cos(\frac{\pi}{N} . [n + \frac{1}{2}] . k)$$

This transform provides great energy compaction, which means the output should produce coefficients more concentrated in the low-frequency components[13]. This is crucial in the compression domain, as this allows to store less values.

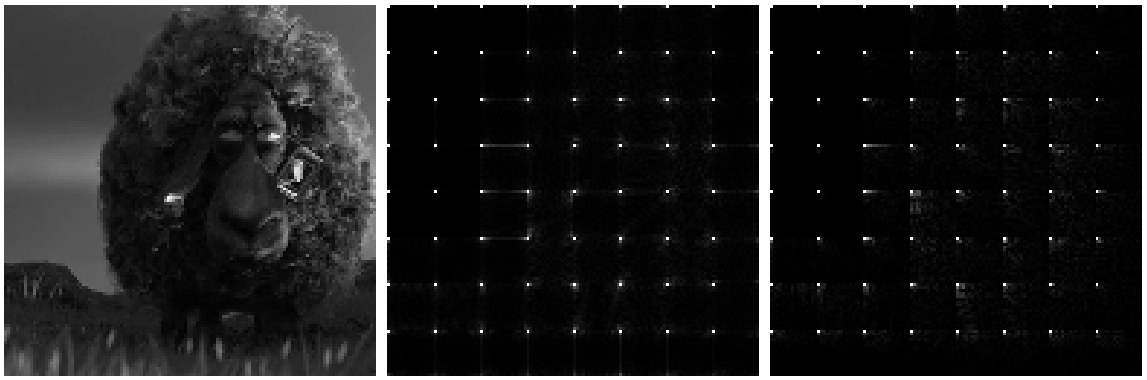

Figure 1: Comparison between the original picture (left), its output after DFT (center), and DCT as in H.265/MPEG-4 HEVC (right). Both transforms were made on a 16×16 blocks decomposition, the picture being 128×128 pixels wide.

In figure 1's outputs, light pixels represent the energy presence after the respective transformation. Unlike the DFT, the DCT concentrates it around the upper-left corners, where the low frequencies are. Also, the DCT provides more negligible values, as its overall preview is darker than the DFT's one. In both case, it becomes clear that more coefficients are needed in broader frequencies for complex areas like the sheep, in comparison to simpler ones like the sky.

In video coding and decoding, two-dimensional transforms and inverse transforms are used. Residual data is not stored in one bit string, but instead in a matrix respecting the errors' position in their block. The formulas mentioned earlier are still valid in this case, but they have to be applied on each dimension.

2.2.2 Quantization

Once the DCT has been applied, we obtain a data matrix whose values are defined on a large range of numbers (2^{16} signed possibilities in H.265/MPEG-4 HEVC[11]). There are often reasons for wanting them to be fixed on a smaller set of values. To achieve this, we can apply quantization on the data. This consists of dividing and rounding it with a chosen step value, which will represent the minimum gap between two numbers after the quantization. This process generates quantization error, which is the difference between the quantified values and the original ones. It is thus a lossy mechanism. However, doing this can provides useful benefits.

It might become possible to use less bits to store values, which has direct positive consequences on the bitrate. This can only happen if the step size is at least doubled. Indeed,

doubling it means only half of the potential values are no longer required, and when it is the case, the most significant bit becomes useless. This can be shown by studying the difference between the number of bits required to express n and $\frac{n}{2}$ values :

$$\frac{ln(n)}{ln(2)} - \frac{ln(\frac{n}{2})}{ln(2)} = \frac{ln(n) - ln(\frac{n}{2})}{ln(2)} = \frac{ln(n) - ln(n) + ln(2)}{ln(2)} = 1$$

Also, samples close but not equal to zero can be approximated and considered null. For this to happen, they have to be below half of the new step value. Finally, as the rounding happens in the frequency space, quantization error will be spread on the whole block in the spacial domain, instead of keeping a one-off nature.

Figure 2: Visualization of the different quantization parameters (QP) within a frame. Lower QP are represented greener. The lower a QP is, the lower the step is, which will result in better reproduction when decoding.

There are cases where the step value does not matter significantly. Indeed, on simple areas, there can be little to no residual data. If so, these blocks' QP will not really influence bitrate, so they can have low step values without any problem. In figure 2 we can see that most of the sky benefit from low QP despite not being a complex area; this is thanks to the success of the prediction techniques.

3 Methodology

3.1 Algorithmic considerations

3.1.1 The DCT's properties

The transforms and inverse transforms can be applied with a simple matrix multiplication, as directed by the mentioned formulas. But since we are doing them in two dimensions, two multiplications are required. One where the data is multiplied on its left by the transform matrix (for columns), and the other one where it is multiplied on its right by the transpose of the transform matrix (for rows)[14].

Inverse transform matrices are defined by the video coding standards (unlike the forward ones, which are implementation-defined)[12], thus they can be stored in the decoder's source code. The inverse DCT matrix (and thus, the forward one, as they are each other's transpose) has useful properties for implementation efficiency. The two halves of even columns are symmetric, or anti-symmetric for odd columns. Every coefficients of an $2^n \times 2^n$ inverse DCT matrix are contained in the $2^{n+1} \times 2^{n+1}$ one[14]. Theses aspects offer big opportunities for code reuse. It also means that storing half of the 32×32 inverse DCT matrix is enough to perform all DCT-based operations on H.265/MPEG-4 HEVC, regardless of the blocks' size[13].

Finally, the forward DCT's capacity to compact energy combined with quantization allows to save many operations during decoding, by skipping multiplications of whole sections of matrices filled with zeros.

3.1.2 Processing optimizations

Regardless of the executed algorithms, there are precautions to take in order to improve efficiency on modern x86 processors. Theses have branch predictors to optimize sequential pipelined executions of jump instructions. If they fail to predict the right outcome, it will result in a time penalty. It is thus considered favorable to avoid using branching; for this, we will use loop unrolling. This consists of duplicating the content of a loop to program re-executions, instead of counting on the jump to do it. If one iteration of a loop does twice the desired operation, half of the jumps disappear. However, while this technique can induce large improvements, it can also slow the execution if the content of a loop it too large : the instruction cache may not be able to keep up[15].

3.2 Exploitation of Single Instruction on Multiple Data (SIMD) architectures

Usually, a register contains one value, and any operation outputs an element of data from at least one register. Single Instruction on Multiple Data (SIMD) architectures allow to store multiple values in a single register, and offer instructions that can process them all at once, simultaneously.

The main benefit is to do in one instruction what would otherwise require the same number of instructions as there are values in the SIMD register. SIMD instructions sets have been integrated in mainstream processors since the late 1990s, with MultiMedia eXtensions (MMX), 3DNow!, Streaming SIMD Extensions (SSE); but also with Advanced Vector Extensions (AVX) in the late 2000s.

The multiplication of two matrices is a typical application of the SIMD technology, as it requires a repetitive process done on data with a fixed format. Thus, such a domain can be exploited to apply the inverse DCT in a video decoding context.

4 Experiment

4.1 Principle

To experiment the effects of SIMD and algorithmic optimizations on residual data decoding, we will test multiple ways of applying an inverse DCT (as in H.265/MPEG-4 HEVC). An 8×8 block of residues will be transformed multiple times with different optimizations turned on or off simultaneously or not to better identify the influence from each type of technique.

The consequences of loop unrolling will be measured. Two versions are tested : one where each iteration of the main loop processes a whole line of the resulting matrix, and another where it processes only one coefficient per iteration. Thus, the first one is more unrolled than the second, since here we only need 8 iterations for the whole output matrix, compared with 64.

Optimization with SIMD will be tested. One version processes the inverse DCT with SSE instructions, the other only uses regular ones.

All tests are done on a system powered by an Intel Core i5-2410M processor working at 2,90 Ghz. The application is not multithreaded and runs on a single core. Compilation has been done with the GNU Compiler Collection with parameters set to avoid unwanted optimizations; and the resulting binaries were inspected with Objdump to check for unwanted behaviors.

4.2 Results

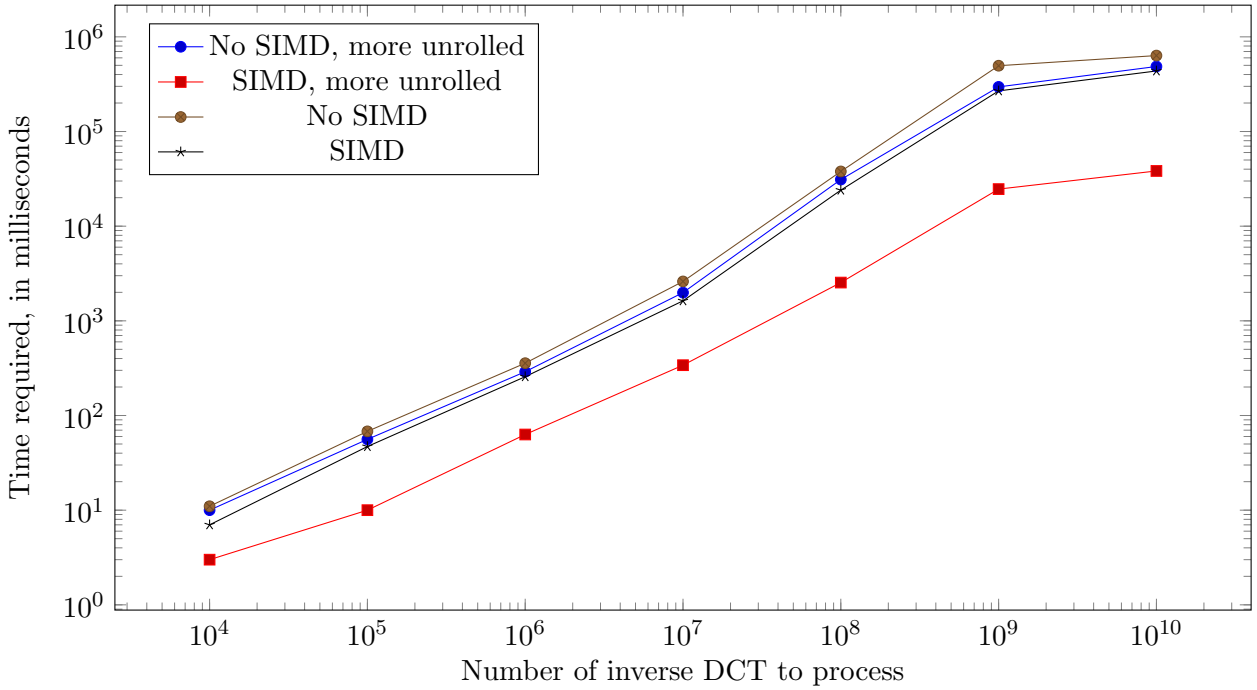

Figure 3: Times required for multiple 8×8 inverse DCT processings, depending on different optimizations. Both axis are logarithmic.

Figure 3 presents the results of the experience. The algorithm that got neither SIMD nor more loop unrolling is the one that required the most time to process inverse DCT. Those who got only one optimization out of two are slightly less time consuming. They tend to show that here, using SIMD instructions is more powerful than loop unrolling. However, when these two are combined, they provide great time savings, as the red curve shows : up to 20 times faster on long decoding sessions.

5 Conclusion

Investigations show that low level optimizations are valuable tools to use for more efficient video decoding. In order to manage the 50 % bit rate reduction targeted by the FVC standard, video decoders will have to implement such techniques. Indeed, transforms and inverse transforms tend to become more complex in order to use less data for the same subjective video quality. To face such a context, the proposed solution is to rely on already-acquired knowledge (from, for example, Institut d'Électronique et de Télécommunications de Rennes (IETR)'s openHEVC) and add to it with low level considerations.

Glossary

AVC Advanced Video Coding. 2, 3

AVX Advanced Vector Extensions. 7

CABAC Context-Adaptive Binary Arithmetic Coding. 3

CB Coding Block. 3

CTB Coding Tree Block. 3

DCT Discrete Cosine Transform. 1, 3, 4, 6–8

DFT Discrete Fourier Transform. 3, 4

DPCM Differential Pulse-Code Modulation. 2

FVC Future Video Coding. 1, 2, 8

HEVC High Efficiency Video Coding. 1–4, 6, 7

IETR Institut d'Électronique et de Télécommunications de Rennes. 8

INSA Institut National des Sciences Appliquées. 1

JVET Joint Video Experts Team. 1

MMX MultiMedia eXtensions. 7

MPEG Moving Picture Experts Group. 1–4, 6, 7

QP Quantization Parameter. 5

SIMD Single Instruction on Multiple Data. 1, 7, 8

SSE Streaming SIMD Extensions. 7

TB Transform Block. 3

VCEG Video Coding Experts Group. 1, 2

References

[1] ITU-T. H.261 : Video codec for audiovisual services at px64 kbit/s. Technical report, ITU-T, Mar 1993.

[2] ITU-T. H.263 : Video coding for low bit rate communication. Technical report, ITU-T, Nov 1995.

[3] ISO/IEC 11172-2 (MPEG-1). Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s—part 2: Video. Technical report, ISO/IEC JTC 1, 1993.

[4] ISO/IEC 14496-2 (MPEG-4 Visual version 1). Coding of audio-visual objects—part 2: Visual. Technical report, ISO/IEC JTC 1, Apr 1999.

[5] ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG 2 Video). Generic coding of moving pictures and associated audio information— part 2: Video. Technical report, ITU-T and ISO/IEC JTC 1, May 2003.

[6] ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC). Advanced video coding for generic audio-visual services. Technical report, ITU-T and ISO/IEC JTC 1, Nov 1994.

[7] Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, and Thomas Wiegand. High efficiency video coding (hevc) text specification. Technical report, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Oct 2012.

[8] ITU-T Study Group 16. Requirements for future video coding (h.fvc). Technical report, ITU-T, Jan 2017. SG16-R1.

[9] C.C. Cutler. Differential quantization of communication signals, July 29 1952. US Patent 2,605,361.

[10] M. Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding*. IET telecommunications series. Institution of Engineering and Technology, 2011.

[11] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[12] F. Loras and J. Fournier. H.264/mpeg-4 avc, un nouveau standard de compression vidéo. Technical report, CORESA and France Télecom R&D, 2003.

[13] Chia-Wei Chang, Hao-Fan Hsu, Chih-Peng Fan, Chung-Bin Wu, and Robert Chen-Hao Chang. A fast algorithm-based cost-effective and hardware-efficient unified architecture design of 4×4, 8×8, 16×16, and 32×32 inverse core transforms for hevc. *Journal of Signal Processing Systems*, 82(1):69–89, Jan 2016.

[14] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, and M. Sadafale. Core transform design in the high efficiency video coding (hevc) standard. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):1029–1041, Dec 2013.

[15] Ulrich Drepper. What every programmer should know about memory. Technical report, Red Hat, Inc., Nov 2007.