TOWARDS REAL-TIME VIDEO WATERMARKING FOR SYSTEM-ON-CHIP

G. Petitjean[*], *JL. Dugelay*[**], *S. Gabriele*[*] , *C. Rey*[**], *J. Nicolai*[*]

[*]STMicroelectronics, Advanced System Technology, Rousset Lab , http://www.st.com
[**] Institut Eurécom, Sophia Antipolis, Dept. of Multimedia Communications, http://www.eurecom.fr/~image

ABSTRACT

In the past few years, much research on watermarking has been focused on improving robustness to attacks. The starting point of the present study was an R & D non optimised algorithm for still images but offering a good trade-off in terms of capacity, visibility and robustness, and working in a full blind manner. We focused on adapting it to video and on optimising it to the world of embedded terminals such as digital still cameras, digital television, wireless terminals, where the computing power and storage resources of a Pentium are not available. This work is the result of a close collaboration between the Eurecom research institute and STMicroelectronics.

1. INTRODUCTION

The ease of modifying and perfectly copying digital data such as audio, video, photos, and the advent of high speed internet access and peer to peer networking are making increasingly difficult the challenge of protecting copyrights and guarantying integrity of digitised multimedia content.

Watermarking is viewed by many as the last barrier when encryption is broken or not feasible, or after the document is decrypted for consuming. Watermarking can be used in several ways: to prevent a compliant device from playing or copying illegally, to trace illegal copies to the indelicate person who posted them on the net, to display graphically the part of a photograph which was electronically modified, to carry invisible information such as place or date of purchase, document identifier, number of authorized viewings or copies, etc. [1]

In the last few years, research has been focused on creating watermarks resistant to various types of attacks ([2] gives a good overview), but with little emphasis on the platform running the watermark algorithm. The processor usually assumed is a powerful Pentium PC, or Unix workstation.

We think that many consumer equipments will need in the near future to run watermarking algorithms: some MP3 music players already do, DVD players and recorders will probably be next [7]. Digital TVs, digital still cameras, set top boxes, wireless appliances, in fact any device capable of displaying or capturing multimedia content may follow.

Efficient watermarking solutions being generally computationally intensive, it is not enough for a watermark to exhibit good robustness and visibility performance. It needs to be compatible with the host application platform, usually a 16 or 32-bit microprocessor, DSP or system-on-chip (SoC), equipped with only some hundreds of kilobytes of memory.

We first give a short overview of our still picture watermark algorithm that is able to defeats many (non destructive) attacks including random geometric distortions (e.g. Stirmark 3) [3]. The initial development was done on workstation, without any platform constraint.

We then describe how we have adapted it to a 32 bit DSP embedded processor, and to a 32 bits VLIW multi-issue processor core, both processors suitable for wireless, embedded applications. The adaptation was made at algorithmic level, at C code level, and finally adaptations were made for the specific targets. Results are shown in the form of execution times versus platform capabilities.

A third paragraph describes the extension of our watermark to digital video, its performances, and the hardware accelerator needed to satisfy real time video constraints. The accelerator could be attached to the main processor, in an hypothetical System-on-Chip MPEG encoder-decoder with watermark embedder and extractor.

Finally, we will conclude with future perspectives.

2. THE ALGORITHM

Our technique, first developed for still picture, is inspired from fractal image coding theory [4], in particular the notion of self-similarity [8]. The main idea is to use some invariance properties of fractal coding, such as invariance by affine (geometric and photometric) transformations, to ensure watermark robustness.

2.1 Embedding

The watermark embedding process can be described as the following three steps: formatting and encryption of the watermark, cover generation, embedding the watermark into the cover.

2.1.1 Formatting and encryption of the watermark

The message bits to be hidden are redundantly distributed: by oversampling and duplication of the message to obtain a watermark of the size of the image. This redundancy is necessary for a good robustness. Finally, the watermark is globally encrypted using a XOR with a pseudo-random noise generated from a secret key, yielding the encrypted watermark W. The XOR operation allows, on one hand, to secure the hidden message, and on the other hand, to remove repetitive patterns reducing in this way the psycho-visual impact of the watermark embedding.

2.1.2 Cover generation

First, a "fractal approximation" I_{approx} is computed from the original image I_{original} (see section 3). The cover I_{cover} corresponds to the error image, that is the signed difference between the original image and its fractal approximation.

$$I_{\text{cover}} = I_{\text{original}} - I_{\text{approx}}$$

2.1.3 Embedding the watermark into the cover

The last step of the embedding process consists in modulating the cover I_{cover} with W. The modulation consists of zeroing some of the cover pixels depending on their sign and the corresponding watermark bit to hide. Only pixels whose absolute value is below a fixed threshold are zeroed, allowing to set visual quality. A threshold of 12 allows to make the mark invisible. Finally, the modulated cover $\hat{I}_{\text{cover}}$ is added to the fractal approximation I_{approx} to produce the watermarked image $I_{\text{watermarked}}$.

$$I_{\text{watermarked}} = I_{\text{approx}} + \hat{I}_{\text{cover}}$$

2.2 Extraction

The watermark extraction algorithm is similar to the embedding algorithm (i.e. dual operations). Its complexity is very close too.

First a fractal approximation is calculated from the watermarked image, which generates a cover close to the original one. Finally the cover is decoded according to the modulation rules (e.g. a positive pixel is supposed to carry a one valued bit, and a negative pixel a zero valued bit). The crucial point is that most geometric transformations on the watermarked image are also transferred to the cover: the mark is not lost but the noise has to be correctly positioned with respect to the cover before applying XOR. Therefore, some additional bits called 'resynchronisation bits' are added to the useful message bits in order to allow a self and blind resynchronisation of samples via two procedures: one for global geometric distortions (rotation and rescaling) based on FFT properties of periodic signals, one for local geometric distortions based on block-matching. Then the watermark can be decrypted and the message rebuilt.

3. OPTIMIZATION FOR EMBEDDED PROCESSOR

3.1 Fractal approximation

The initial research prototype was very performing but not optimised: it took about 40s to watermark a 512×512 pixels image on a PIII 733MHz.

We then conducted an optimisation in two steps: first at algorithmic level, then at C code level, with some fine tuning for porting to a DSP or a VLIW processor.

We have focused on the fractal approximation part of the algorithm, since it takes about 80% of the total computing time both in embedding and extraction. Our fractal approximation is very inspired by fractal coding [4]. The image is parsed into non overlapping "range" blocks (here 8×8 pixels) which are matched with larger "domain" blocks (here 32×32 pixels, subsampled to 8×8) and with all the

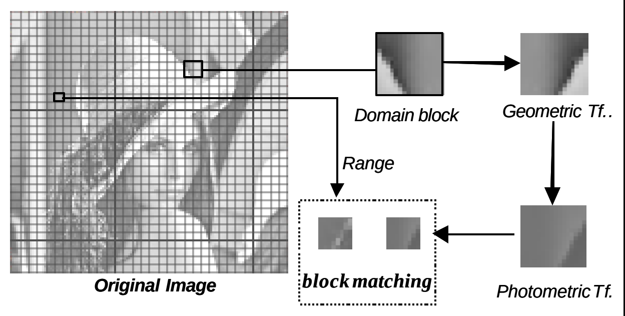

Fig. 1 : IFS computation in fractal coding. s and o are the photometric parameters computed in formulas (1) and (2)

isometric domains (we apply the classical 8 isometries: identity, 4 reflections, 3 rotations). And to allow the domains to be closer from the range, their average brightness and contrast are modified to match those of the range, according to the exact formu las,

$$s_{opt} = (n\Sigma.d_i \,.\, r_i \;-\; \Sigma\, d_i.\Sigma\, r_i) \;/\; (n\Sigma d_i^2 - (\Sigma d_i)^2) \qquad (1)$$

$$o_{opt} = (\Sigma.d_i \;-\; s_{opt}.\Sigma\, r_i) \;/\; n \qquad (2)$$

where s_{opt} and o_{opt} are the optimum contrast and brightness modifications, d the rescaled and transformed domain and r the range. Then the domain d for which $s_{\text{opt}}.d + o_{\text{opt}}$ is the closest from range r is searched. The metric used to find the best domain is the quadratic norm. The Euclidian error Q is computed according to:

$$Q=\Sigma(s.d_i + o - r_i)^2 \qquad (3)$$

which can also be computed as:

$$Q=\frac{1}{n}*\left[\sum r_i^{\,2}+s*\left(s*\sum d_i^{\,2}-2*\sum d_i*r_i+2*o*\sum d_i\right)+o*\left(n*o-2*\sum r_i\right)\right] \qquad (4)$$

The main differences with fractal coding are:

-the search window is limited to a smaller neighbourhood of the current range (a 34×34 block centred on the range) to allow good robustness especially to cropping

-after having found the best domain with the best transformation, we replace the range with this block to obtain the approximation, instead of keeping the IFS (Iterated Functions System) code. Figure 1 summarizes the general process of fractal coding.

3.2 Algorithmic optimisations

	PSNR	Visual assessment	Cover stability	Adopted
Original algorithm	38.69dB	-	41%	-
No isometric domains	37.98	Slightly visible	38%	No
Same isometry for all domain pool	38.69	Not visible	41%	Yes
Approached computation for s and o + SAD	37.27	Slightly visible	32%	No

Figure 2: evaluations of some possible algorithm variations that may reduce computing resources, for 518×744 "brandyrose" image.

Many papers deal with fractal coding complexity reduction: one can find a good overview in [5]. Inspired by those works, several variations have been implemented and evaluated: metric SAD (Sum of Absolute Differences) vs. Euclidian norm, number of isometric domains used, size of the blocks, and computing method for brightness and contrast adjustments (exact or approached formulas). To assess each of these possible variations we needed a precise methodology. As for any watermarking technology, there is a compromise between visibility, robustness and bit capacity. We have fixed the capacity to 64 bits and evaluated the other two parameters. Visibility assessment was done by comparing the PSNR original/watermarked image between original and modified algorithm and also by visual evaluation (since PSNR does not handle efficiently human visual system properties). Robustness assessment was done by comparing in the original and modified algorithms the "cover stability", i.e. the number of pixels that have the same sign in the modulated cover during embedding and in the extracted cover during extraction. The higher this number is, the more robust the algorithm is, since more correct bits will be extracted. This comparison was also done after image manipulations (noise addition and blurring).
Surprisingly the Euclidian norm gives globally better results than SAD: SAD is faster but does not allow the use of the formula (4) that avoids computing photometric changes sD+o for all the blocks, which is a very intensive task. Then SAD only decreases the computing time if it's used with approximated formulas for s and o. But those formulas damaged the cover stability.
About isometric domains, we found that computing the best isometry on the first domain and keep it for the remaining domains was a good solution. It reduces the number of matching to do from 72 (9 domains times 8 isometries) to 16, without decreasing robustness and visibility (surely because the 9 domains are very close). Figure 2 shows some of our results. Those modifications have decreased computing time from 40s to 10s to watermark a 512×512 image (PIII 733MHz).

3.3 Code optimisation and porting

We then focused on the implementation. We have done classical optimisations: decrease memory accesses, avoid counter operations in the loops (e.g. by index arrays), change float variables into integers. This last transformation was difficult for s and o computation since they must be very precise to insure good robustness. The resulting code took 0.5s to watermark the same image instead of 10s. The global gain was 80:1. At the same time the data memory space needed was divided by 3.5 to reach 2MB.
Our first porting was on a DSP from STMicroelectronics. Only little work was needed to optimise it to the target thanks to the previous generic optimisation. We coded a few inner loops in assembler. We then ported the watermarking code to an STMicrolectronics VLIW processor. It contains 4 fixed point ALUs which can process 4 instructions in parallel, thanks to its powerful compiler. Figure 3 shows the results for embedding (extraction complexity is very close).

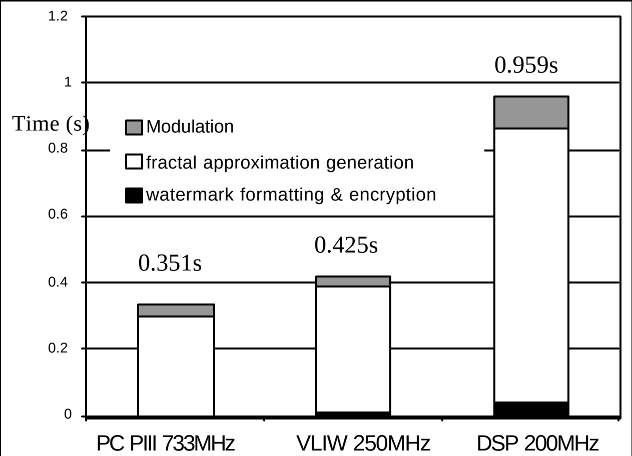

Fig.3: Complexity comparison PC / VLIW / DSP

4. EXTENSION TO REAL TIME VIDEO

As our algorithm was designed for still picture we first adapted it to video. We think that a high degree of robustness will be needed for video, as much as for still image, since video specificities together with increasing computing power, allowing easy video processing, will soon allow new specific attacks. We worked at algorithm level to merge the watermark with an MPEG2 codec [6] in order to mark video sequences during MPEG2 coding and extract watermarks during MPEG2 decoding. To reach the higher

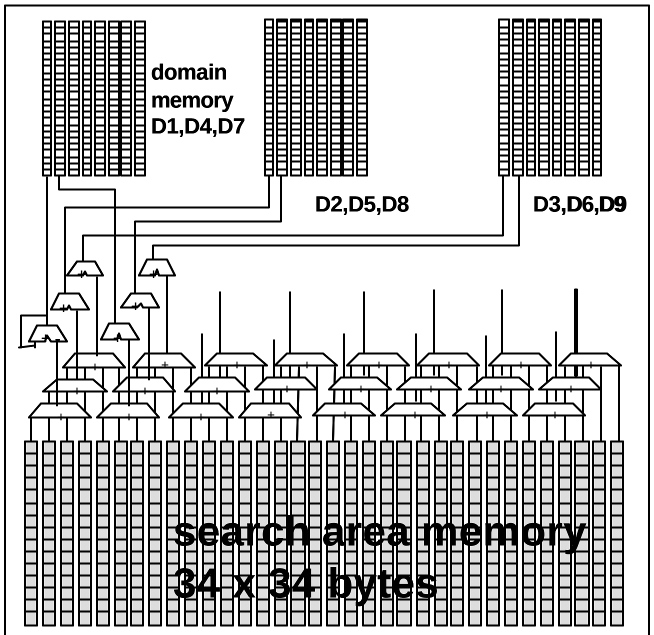

Fig. 4 : hardware architecture for domain extraction and subsampling, based on parallel accumulators. The 9 domains are extracted and subsampled in 64 clock cycles.

robustness, we first chose to mark all the frames, with the same message, and to extract watermarks using frame accumulation: each bit of the message is averaged on N frames to increase robustness. Currently we are testing other embedding strategies to improve visual quality/robustness trade-off.

In parallel we worked on the hardware part since an important issue to solve was the processing power necessary to watermark video in real-time, which cannot be done in software only. Figure 3 shows that fractal coding takes the biggest share of the CPU time, in fact in the order of 0.3 to 1 second depending on the platform. A real time video implementation then requires at least a tenfold speed improvement (in the worst case where we mark all the frames). Hence we defined a hardware accelerator, dedicated to the fractal coding part of the algorithm, which could be integrated into a system-on-chip as a coprocessor to the CPU core. This accelerator receives from the processor the 8×8 pixel range and associated 34×34 search area (luminance data only) and returns to the processor the best domain.

Such an accelerator allows running repetitive tasks in parallel. For example Figure 4 shows the domain extraction and subsampling part of the coprocessor. It creates the 9 domains and simultaneously subsamples them in 64 clock cycles (1.28 μs @ 50MHz), using massive parallelism, and specific memories to hold the search area and resulting domains. Then it computes s_{opt}, o_{opt} and the error Q in a parallel (3 domains at a time) and pipelined way. Our simulations show that the complete process takes less than 6μs on an FPGA running on a 50 MHz clock, compared to 78μs on a Pentium III 733MHz and 118μs on a VLIW 250 MHz. The fractal coding of a 512×512 frame would then require 4096 × 6μs = 25 ms, which is very close to the performance required to achieve video rate. An implementation as coprocessor inside a system-on-chip would allow a clock frequency of at least 100 MHz, bringing this time down to 12.5 ms maximum (to compare to 0.41s, software only, on VLIW), making it possible to mark video and extract a mark in real time with a single System On Chip.

5. CONCLUSION

We showed the feasibility of real-time video watermarking with a top performing, then computationally intensive, algorithm. We started from a Stirmark resistant, still picture algorithm at research level, optimised both algorithm and C code and ported it to DSP and VLIW processors. Then we designed a hardware accelerator architecture to reach real time video requirements. In the future we will focus on better adapting our algorithm to video properties to improve both robustness and visual quality.

REFERENCES

[1] S. Katzenbeisser, F. A.P. Petitcolas, *Information Hiding – Techniques for Steganography and Digital Watermarking*, Artech House, Boston-London, 2000.

[2] N. Nikolaidis, I. Pitas, *Digital image watermarking:an overview*, ICMCS 99, vol I, pp 1-6, 1999.

[3] J.-L. Dugelay & F. Petitcolas, *Image Watermarking: Possible counterattacks against Random Geometric distortions*, Conference SPIE, California, Jan. 2000

[4] Fisher Y., Editor, *Fractal image encoding and analysis*, NATO ASI Series, Series F: Computer and Systems Sciences, vol. 159, Springer-Verlag, 1998.

[5] D. Saupe , R. Hamzaoui, *Complexity Reduction Methods For Fractal Image Compression*, in Proc. IMA Conf. On Image Processing; mathematical methods and applications (1994). Oxford University Press, 1996

[6] F. Rovati, D. Pau, L. Pezzoni, E. Piccinelli, J. Bard, *An Innovative, High Quality and Search Window Independant Motion Estimation algorithm and Architecture for MPEG2 Encoding*, IEEE transactions on Consumer electronics, vol. 46 n.3, August 2000

[7] M. Maes & al, *Digital Watermarking for DVD Video Copy Protection*, IEEE Signal Processing Magazine 1053-5888, September 2000

[8] J.-L. Dugelay, S. Roche, C. Rey, Pending Patents PCT/FR99/00485(EURECOM 09-PCT), March 1999, EP 99480075.3(EURECOM 11/12 EP), July 1999, EUP 99480075.3(EURECOM 14 EP), May 2001