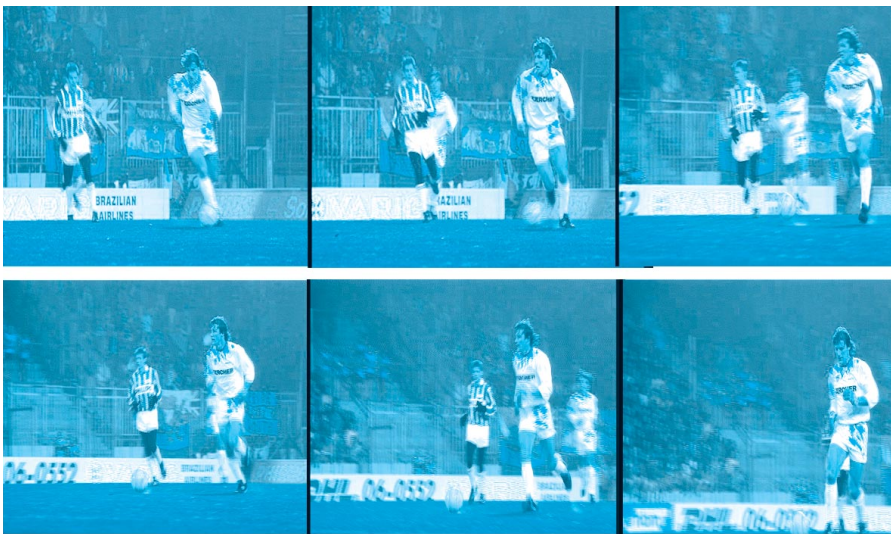
ST Journal of System Research
vol 4, number 1 March 2003

Security

From Still Picture to Real - Time Video Watermarking



Guillaume Petitjean, STMicroelectronics
guillaume.petitjean@st.com
Jean Nicolaï, STMicroelectronics
Sophie Gabriele, STMicroelectronics
Emiliano Piccinelli, STMicroelectronics
Jean-Luc Dugelay, Eurecom institute
Christian Rey, Eurecom institute

In the past few years, much research on watermarking has focused on improving robustness to attacks on still pictures. The starting point of the present study was an R & D algorithm for still images, which, though not optimized, offered 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 for the world of embedded terminals such as digital still cameras, digital television, wireless terminals, where the computing power and storage resources of a PC 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 the challenge of protecting copyrights and guarantying integrity of digitized multimedia content increasingly difficult.

Watermarking is viewed by many as the last barrier when encryption is broken or not feasible, or after the document is decrypted for consuming: it is the only known method for protecting clear multimedia content. Watermarking can be used in several ways: to prevent a compliant device from playing or copying illegally, to trace illegal copies to the 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 mainly focused on creating still picture 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 to run watermarking algorithms in the near future: 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.
Because efficient watermarking solutions are 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 which was one of the first ever able to defeat many (non destructive[1]) attacks including random geometric distortions (e.g. Stirmark 3) [3]. The initial development was done on a workstation, without any platform constraint.
We then describe how we have adapted it to a 32bit DSP embedded processor, and to a 32-bit VLIW multi-issue processor core, both processors suitable to 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. We tested different embedding strategies to reach the best trade-off visibility/robustness, in collaboration with AST Agrate Lab.
Then we describe 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 codec with watermark embedder and extractor.
Finally, we conclude with future perspectives.

2. EURECOM'S STILL PICTURE ALGORITHM

Our technique, first developed for still picture [8], is inspired from fractal image coding theory [4], in particular the notion of self-similarity. 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 the 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.

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

2.1.3 Embedding the watermark into the cover

The last step of the embedding process consists of modulating the cover I_{cover} with W. The modulation consists of zeroing some of the cover pixels[2]. More precisely if the current pixel is positive and the corresponding watermark bit is zero or if the pixel is negative and the bit is one we zero the pixel. Only pixels whose absolute value is below a fixed threshold are zeroed, allowing to set visual quality. Typically a threshold of 12 allows to make the mark invisible (PSNR is at least 37 dB). Finally, the modulated cover $\hat{I}_{cover}$ is added to the fractal approximation I_{approx} to produce the watermarked image $I_{watermarked}$.

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

1 Watermarking usually focuses on attacks that maintain the value of the multimedia document.

2 For simplification, we call "pixels" the samples of the cover but their values are between –255 and 255.

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, thanks to the modulation done in embedding). 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 'resynchronization 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. OPTIMISATION FOR EMBEDDED PROCESSOR

3.1 Fractal approximation

The initial research prototype performed well but was not optimised: it took about 40s to watermark a 512x512 pixels image on a PentiumIII™ 733MHz.

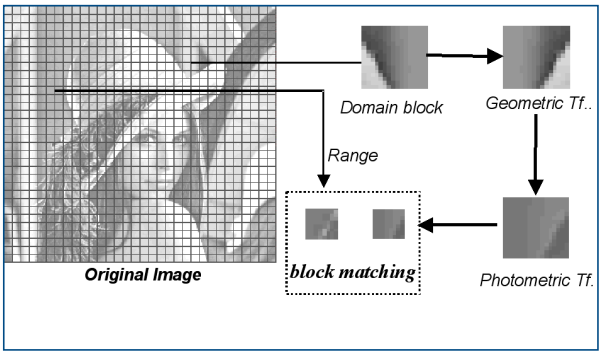


Fig 1: IFS computation in fractal coding. s and o are the photometric parameters computed in formulas (1) and (2)

We then conducted an optimisation in two steps: first at the algorithmic level, then at the 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 inspired by fractal coding [4]. The image is parsed into non overlapping "range" blocks (here 8x8 pixels) which are matched with larger "domain" blocks (here 32x32 pixels, subsampled to 8x8) and with all the isometric domains (we apply the classical 8 isometries: identity, 4 reflections, 3 rotations). To allow the domains to be closer to the range, their average brightness and contrast are modified to match those of the range, according to the exact formulas,

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

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

where s_{opt} and o_{opt} are the optimum contrast and brightness modifications, d the rescaled and transformed domain, r the range and n the number of pixels in a range (typically 64). Then the domain d for which sopt .d + oopt 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:

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

which can also be computed as:

(4)

	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 518x744 "brandyrose" image.

The main differences with fractal coding are that:

- the search window is limited to a smaller neighbourhood of the current range (a 34x34 block centred on the range) to allow good robustness especially with respect to cropping
- after having found the best domain with the best transformation, we replace the range with this block to obtain an approximation, instead of keeping the IFS (Iterated Functions System) code.

Figure 1 summarizes the general process of fractal coding.

3.2 Algorithmic optimisations

Many papers deal with fractal coding complexity reduction: one can find a good overview in [5]. Inspired by those works, we have implemented and evaluated several variations: 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 among 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 vs. watermarked image between original and modified algorithm and also by visual evaluation (since PSNR does not efficiently handle 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 is used with approximated formulas for s and o. However those formulas damaged the cover stability.

About isometric domains, we have found that computing the best isometry on the first domain and keeping 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). Table 1 shows some of our results. Those modifications have decreased computing time from 40s to 10s to watermark a 512x512 image (PIII 733MHz).

3.3 Code optimisation and porting

We then focused on the implementation. We have performed classical optimisations: decrease of memory accesses, avoidance of counter operations in the loops (e.g. by index arrays), change of floating variables into fixed point. This last transformation was difficult for s and o computation since they must be very precise to insure good robustness. The resulting code takes 0.5s to watermark the same image instead of 10s. The global gain is 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. This parallelisation is done thanks to its compiler. Figure 3 shows the results for embedding (extraction complexity is very close). The VLIW is the most performing chip at equal frequency as it could be expected since it has 4 ALUs working in parallel.

4. ADAPTATION TO 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 for 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

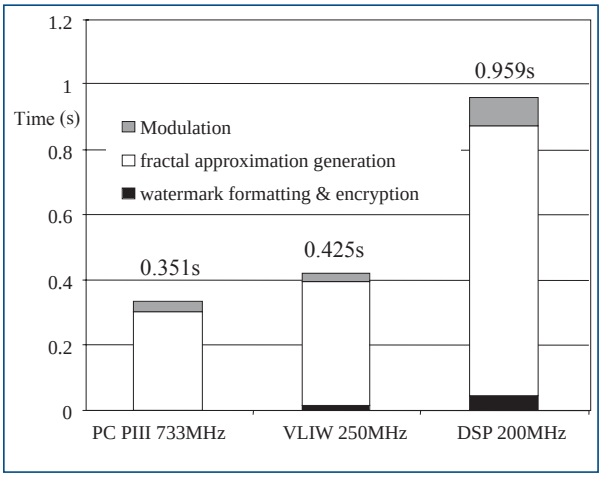


Fig. 3: Complexity comparison PC/ VLIM/DSP

during MPEG2 decoding. To reach the higher robustness, we first marked all the frames, with the same message and the same key, and extracted watermarks using frame accumulation: each bit of the message is averaged on N frames to increase robustness.

4.1 Visual quality

Unluckily this solution led to low quality sequences. The visual distortion was a kind of grid superimposed on the video. Marking a different message in each frame did not change it. We then decided to test other embedding strategies in order to reach better quality. To assess each solution we operated from human observation as no mathematical tool (like PSNR) can handle video properties efficiently enough. The observations were done on a DVS machine (Digital Video System) that allows watching digital videos on a high quality monitor, with interlaced display and especially to watch several sequences simultaneously. Our test protocol consisted of watching two sequences at a time to compare several embedding strategies: marking all frames, marking only I and P frames, marking only I frames and also two embedding positions: before or after the preprocessing that occur in an efficient MPEG2 encoding (that aims for example to reduce noise and then to improve the compression). The results can be found in Appendix. Original sequences are four 150-frames long clips having different and complementary properties (movie, football match, music clip… interlaced or progressive…), with or without noise.

4.2 Robustness

We extracted without any error the mark from all the watermarked sequences generated for visual tests (i.e. without manipulation) using a frame accumulation length of 25 (it corresponds to one second of PAL video, which seems an acceptable length to read the mark, even if the CPAC[3] requirement for future DVD watermarking standard is much less strict: a maximum detection period of 15s or 375 frames). As the best solution in terms of visual quality is to mark I and P frames before the MPEG preprocessing (see Appendix) we did deeper robustness tests with this method. On still pictures the algorithm is robust to StirMark 3.1 which is a state of the art still picture watermarking benchmark. StirMark includes random geometric distortion, JPEG compression, row or column suppression, etc… It is also robust to rescaling but, being on-going research, this last functionality has not been included in the integration. As, for video, embedding and extraction are done independently in each frame our system is very likely to resist a video version of StirMark and/or a rescaling even if it has not been tested yet. Then we mostly tested video algorithm robustness towards some video specific manipulations or attacks:

- **MPEG2 compression:** the mark was extracted without any error after a compression down to 1.5 Mbits/s. Such a bit rate is considered sufficient since the value of the video will be dramatically decreased at lower rates.

- **Over-compression:** an important manipulation/attack that is likely to occur on a MPEG2 sequence is a second compression at a lower bit-rate. Two possibilities can happen: the GOPs of both encodings can match or not. If they do, this manipulation is very close to a single encoding and

3 Copy Protection Advisory Counsel: body in charge of choosing the watermarking system for DVD movies.

the mark is found down to 1.5 or 2 Mb/s. If they don't, the previous I or P frames can become B ones and then, being much more compressed, the mark is correctly extracted only down to 2.5 Mb/s with a frame accumulation length of 100.

- **Cropping:** cropping is likely to occur during the normal use of a video (e.g. when a movie is displayed on a TV monitor). We recoded a previously 4 Mbits/s compressed sequence, cropping it with a center window whose size was 480x288 (2/3 of original size). The mark was correctly extracted (25 frames accumulation only).

5 REAL-TIME VIDEO

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 architecture, 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 8x8 pixel range and associated 34x34 search area (luminance data only) and returns to the processor the best domain. It allows running repetitive tasks in parallel.

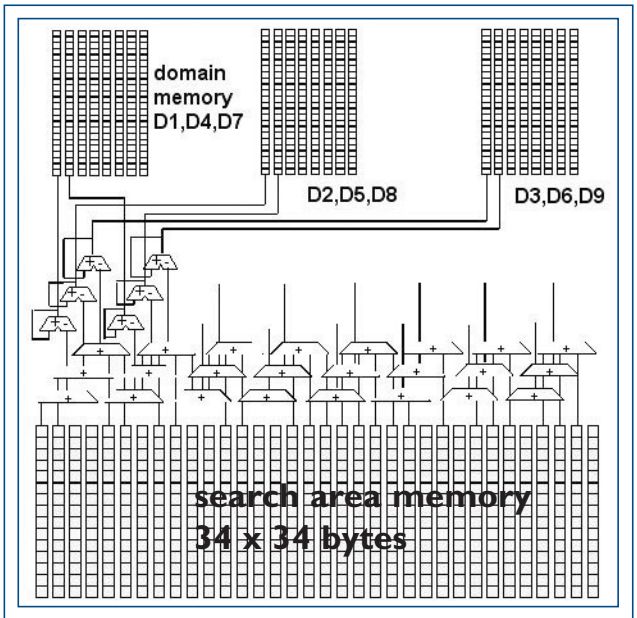


Fig. 4: hardware architecture for domain extraction and subsampling; the 9 domains are extracted form the search area and subsampled in 64 clock cyles

The implementation is still on going but we already implemented in RTL the domain extraction and subsampling part (Figure 4). It creates the 9 domains and simultaneously subsamples them in 64 clock cycles (1.28 ms@50MHz), using massive parallelism, and specific memories to hold the search area and resulting domains.

Then the coprocessor computes s_{opt}, o_{opt} and the error Q in a parallel (3 domains at a time) and pipelined way (to be implemented. Our simulations show that the complete process will take less than 6ms on an FPGA running on a 50 MHz clock, compared to 78ms on a Pentium III 733MHz and 118ms on a VLIW 250 MHz. The fractal coding of a 512x512 frame would then require 4096 x 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.

6. 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 and optimised both algorithm and C code. Then we ported it to DSP and VLIW processors. To adapt it to video we merged at software level the watermarking code with a state-of-the-art MPEG2 codec. In order to find the best trade-off visual quality/robustness we assessed different embedding solutions on a professional video environment. Then we designed a hardware accelerator architecture to reach real time video requirements.

7. 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 1, 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.

* *"Before" and "after" are the embedding positions regarding preprocessings. "I and P" and "I" specify what frames are watermarked.*

** *MC-NR stands for Motion Compensated noise reduction, a video specific noise removal algorithm. CIF stands for CIF reduction: the frames are subsam pled to CIF size in order to save bandwidth.*

° *It can be explained by the fact that in that configuration, the mark is "over sampled" when the sequence is displayed.*

*** *Simple profile: there are only I or P frames during encoding. Then the same visual effect happens than in main profile with watermarking of all the frames.*

	Reference sequence	Evaluated method	Preprocessings enabled**	Results
First clip	No WM	I and P / before	MC-NR	Very slightly visible (flickering), only on not moving areas (grass). No more visible if distance to the screen is >2m. Not annoying
	No WM	I / before	MC-NR	No visible difference
	I / before	I and P / before	MC-NR	"I" is very slightly better (flickering in "I and P" only in not moving areas). Not annoying
	I and P / before	I and P / after	MC-NR	No visible difference
Second clip	Before / I and P	After / I and P	MC-NR	Idem
Second clip with added noise	No WM	Before / I and P	MC-NR	Idem
	I and P / before	I and P / after	MC-NR	Idem
Third clip	Before / I	Before / I and P	MC-NR	No visible difference at normal speed. "I" is slightly better when slowing x 5
	I and P / before	I and P / after	MC-NR	No visible difference
	No WM	Before / I and P	MC-NR + CIF	Very slightly visible when slowing x 2
	No WM	After / I and P	MC-NR + CIF	Very visible. Annoying°
	Before / I and P	After / I and P	MC-NR + CIF	"before" is better. Distortions in "after" are annoying

Table 1: Visual comparison on DVS. Bit rate is 4 Mbits/s (1)

	Reference sequence	Evaluated method	Preprocessings enabled**	Results
Third clip with added noise	No WM	After / I and P	MC-NR + CIF	"After" is annoying but less than for un-noisy sequence
	No WM	Before / I and P	MC-NR + CIF	No visible difference
	Before / I and P	After / I and P	MC-NR + CIF	"Before" is better though less difference than for un-noisy sequence
Fourth clip	No WM	Before / I and P	MC-NR	Very slightly visible only on a not moving area. Not annoying
	Before / I and P	After / I and P	MC-NR	No visible difference

Table 2: Visual comparison on DVS. (2)

[5] D. Saupe , R. Hamzaoui, *Complexity Reduction Methods For Fractal Image Compression,* in Proc. IMA Conf. On I mage 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 Independent 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

8. ACKNOWLEDGEMENTS

We'd like to thank AST Agrate Lab staff and especially Emiliano Piccinelli and Danilo Pau for their helpful support on video, MPEG2 coding and DVS machine.

9. ABOUT THE AUTHOR(S)

Guillaume.petitjean@st.com
R&D engineer Advanced System Technology
STMicroelectronics
Sophie.Gabriele@st.com - R&D engineer
Advanced System Technology - STMicroelectronics

Jean.nicolai@st.com - Program Manager
Advanced System Technology - STMicroelectronics

Emiliano.piccinelli@st.com - Advanced System Technology
STMicroelectronics

Jean-Luc.Dugelay@eurecom.fr - Professor - Eurecom institute

Christian.Rey@eurecom.fr
PhD. student - Eurecom institute