2002

Spread spectrum-based video watermarking algorithms for copyright protection

Serdean, Cristian Vasile

http://hdl.handle.net/10026.1/563

http://dx.doi.org/10.24382/3920
University of Plymouth

Chapter

3

“And if, to be sure, sometimes you need to conceal a fact with words, do it in such a way that it does not become known ...”

Nicollo Machiavelli (1469-1527)

Spatial Domain Watermarking

The chapter presents in detail the structure and the algorithm of a spread spectrum video watermarking system. This chapter establishes the foundation and the skeleton of the spread spectrum watermarking system. The main structure and many of the components described here are used for building the DCT scheme from Chapter 5. Starting with the basic uniform marking, the system is gradually improved by embedding the watermark in a more efficient way (e.g. HVS dependent) and by using a sliding correlator for watermark recovery. The effects of pre-filtering and various block sizes on the performance of a spatial domain watermarking scheme are also investigated.

3.1 Watermark Embedding

3.1.1 Embedding the Watermark: the Spread-Spectrum Approach

As already stated, the basic idea of spread spectrum is to trade signal-to-noise ratio (SNR) for bandwidth. In other words, the signal energy is spread over a wide frequency range at low SNR so that it is difficult to detect, intercept or jam. Though the total signal power may be large, the SNR in any band is small. Moreover the signal energy resides in all frequency bands.

For watermarking this translates to a visually imperceptible watermark, spread in the entire video sequence. Since the watermark resides in all frequency bands, it is likely that even when attacked at least some parts of the spectrum still remain intact, given that usually the attacks are band limited (for example compression, which removes the high frequency components of the spectrum). The spreading signal, which is in fact a pseudo-random or pseudo-noise sequence, who will be called "PN sequence" from now on, it is unknown to a potential attacker, being generated function of a secret key.

The DSSS (Direct Sequence Spread Spectrum) is the simplest form of a spread spectrum technique. Basically the signal is modulated by a function that alternates pseudo-randomly between $+\alpha$ and $-\alpha$, at multiples of a time constant called the *chip rate*. This pseudo-random carrier contains components of all frequencies, which is why it spreads the modulated signal's energy over a large frequency band. For watermarking, the chip rate can be considered as the spacing between the pixels.

Let's assume that $u_k, u_k \in \{-1,1\}$ are the *input data bits* or the *payload* which is to be hidden into the video sequence, converted from the $\{0,1\}$ values to $\{-1,1\}$ values. This signal is spread by a large chip rate factor c_r to give the spread sequence b_i

$$b_i = u_k, \quad k \cdot c_r \le i \le (k+1) \cdot c_r \tag{3.1}$$

Each data bit now spans c_r binary bits. The spread sequence b_i is amplified with an amplitude adjustment factor α and then modulated with the binary PN sequence $p_i,\ p_i \in \{-1,1\}$ giving the watermark signal

$$w_i = \alpha \cdot p_i \cdot b_i \tag{3.2}$$

Finally the watermark is added to the luminance component of the video sequence v_i, giving the watermarked signal

$$v_i^M = v_i + w_i = v_i + \alpha \cdot p_i \cdot b_i \tag{3.3}$$

Some authors are calling this "amplitude modulation".

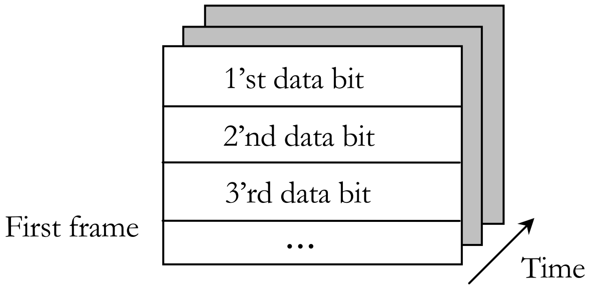


Figure 3-1 Hartung's method of spreading the payload

This algorithm is used in [Hartung et al, 1996 and 1998]. From the embedding side, the algorithm has two major weaknesses. Firstly the algorithm uses uniform marking (a constant amplification factor for the entire video sequence) and therefore is not adaptive in the sense of incorporating at least some HVS aspects. This leads to reduced robustness especially in the case of compression. Secondly, the way described in equation (3.1) for spreading the payload is not the most secure and robust way to do it, since all the data bits are grouped together in a region of the video and embedded one group after another. A possible case is illustrated in

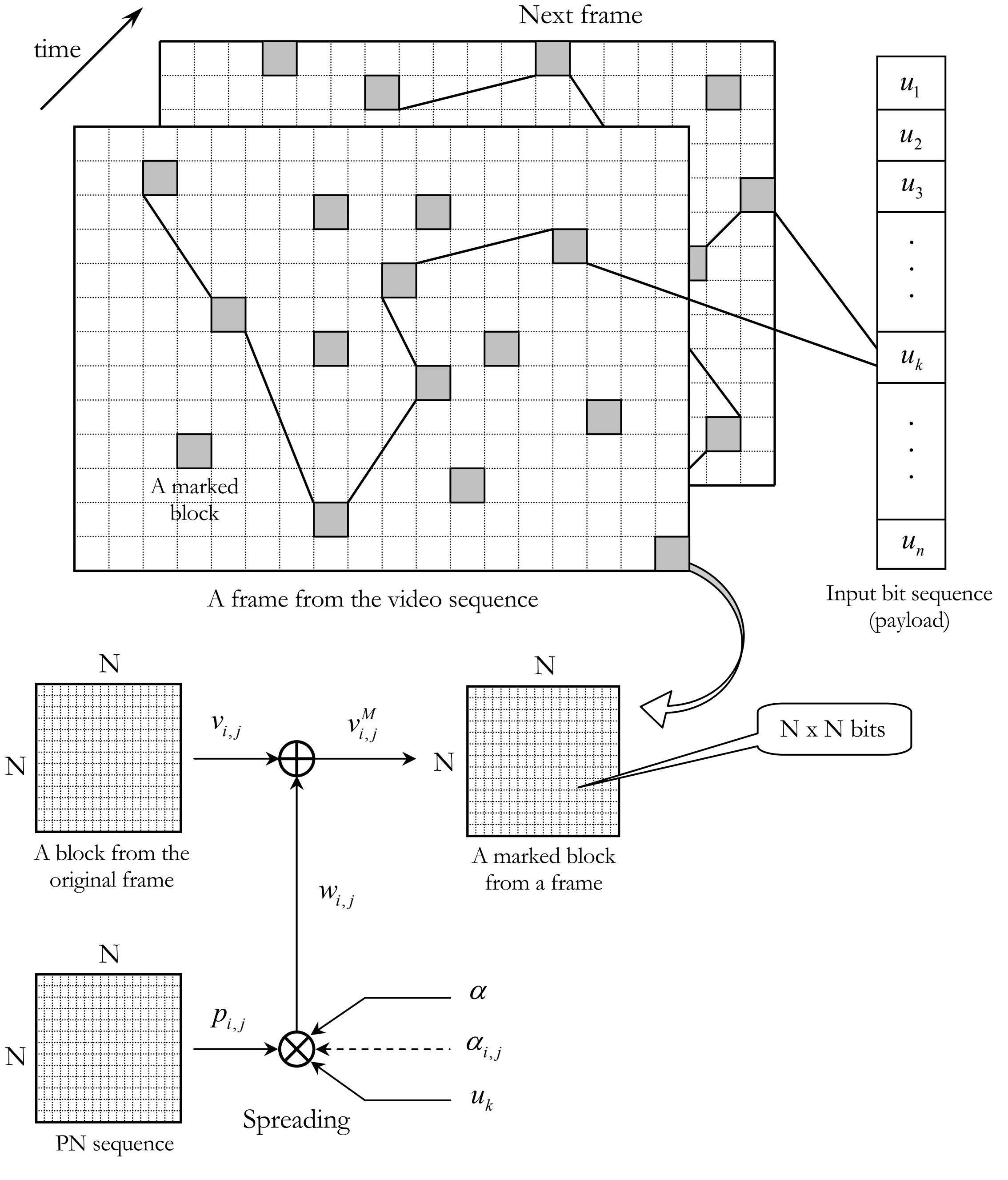


Figure 3-2 Secure, block-based video watermarking – the embedding process

Figure 3-1. In this case if an attacker removes the first frame, it is impossible to recover the first 3 bits (and possibly the next one) even by using a sliding correlator, since because of their spatial localisation they are lost. A much better way is to spread the sequence in the entire video.

The system presented in **Figure 3-2** was designed having these considerations in mind. From reasons of security and robustness to geometric attacks like line cuts or column cuts, the scheme is block based rather than full frame. Each NxN block corresponds to one data bit from the input sequence. Overall, due to the spreading the algorithm assigns a number of NxN blocks for each input data bit. To ensure a better system security, the scheme employs a scrambling mechanism which distributes all these blocks corresponding to one data bit in the entire video sequence, according to a given key. Therefore the system can have two different keys: one key is used for generating the PN sequence, and the other one for the block scrambling mechanism. The block scrambling is illustrated in **Figure 3-2** for two consecutive frames. The blocks corresponding to input bit u_k are pseudo-randomly distributed within the frame and within the video sequence (spatial and temporal spreading). For better security, the locations of the blocks are different for each frame.

The pseudo-random sequence

The binary PN sequence has a uniform distribution being generated by a uniform random number generator. The random number generator used is based on the linear congruential method [Knuth, 1981], [Schneier, 1996] and [Press et al, 1992]. This method generates a sequence of random numbers X_n according to the following formula

$$X_{n+1} = \left(aX_n + c\right) \operatorname{mod} m, \quad n \geq 0 \tag{3.4}$$

where the integer m represents the modulus ($m > 0$), a is the multiplier ($0 \leq a < m$), c is the increment ($0 \leq c < m$) and X_0 is the initial or the starting value, usually called the "seed" of the generator ($0 \leq X_0 < m$). The congruential sequence is periodic with a period not greater than m. From this reason one would like to choose m as a rather large number. If the multiplier a is properly chosen [Knuth, 1981] this leads to a period of maximal length (e.g. length m). In this case any initial "seed" choice of X_0 is as good as any other: the sequence just takes off from that point. In fact this seed constitutes the secret key of the PN generator and is a 32 bit integer.

The special case $c = 0$ leads to a faster algorithm but reduces the length of the period of the sequence. Nevertheless it is still possible to make the period sufficiently long. This case is known as the multiplicative congruential method or mixed congruential method. The generator chosen for this work is an improved multiplicative congruential pseudo-random generator [Press et al, 1992]. The algorithm uses the L'Ecuyer method with Bays-Durham shuffle and added safeguards and has a period greater than 2×10^{18}. [Press et al, 1992] calls this the "perfect" random number generator offering to pay 1000USD to the first reader who can prove the contrary. In terms of security, this generator is considered as being relatively insecure.

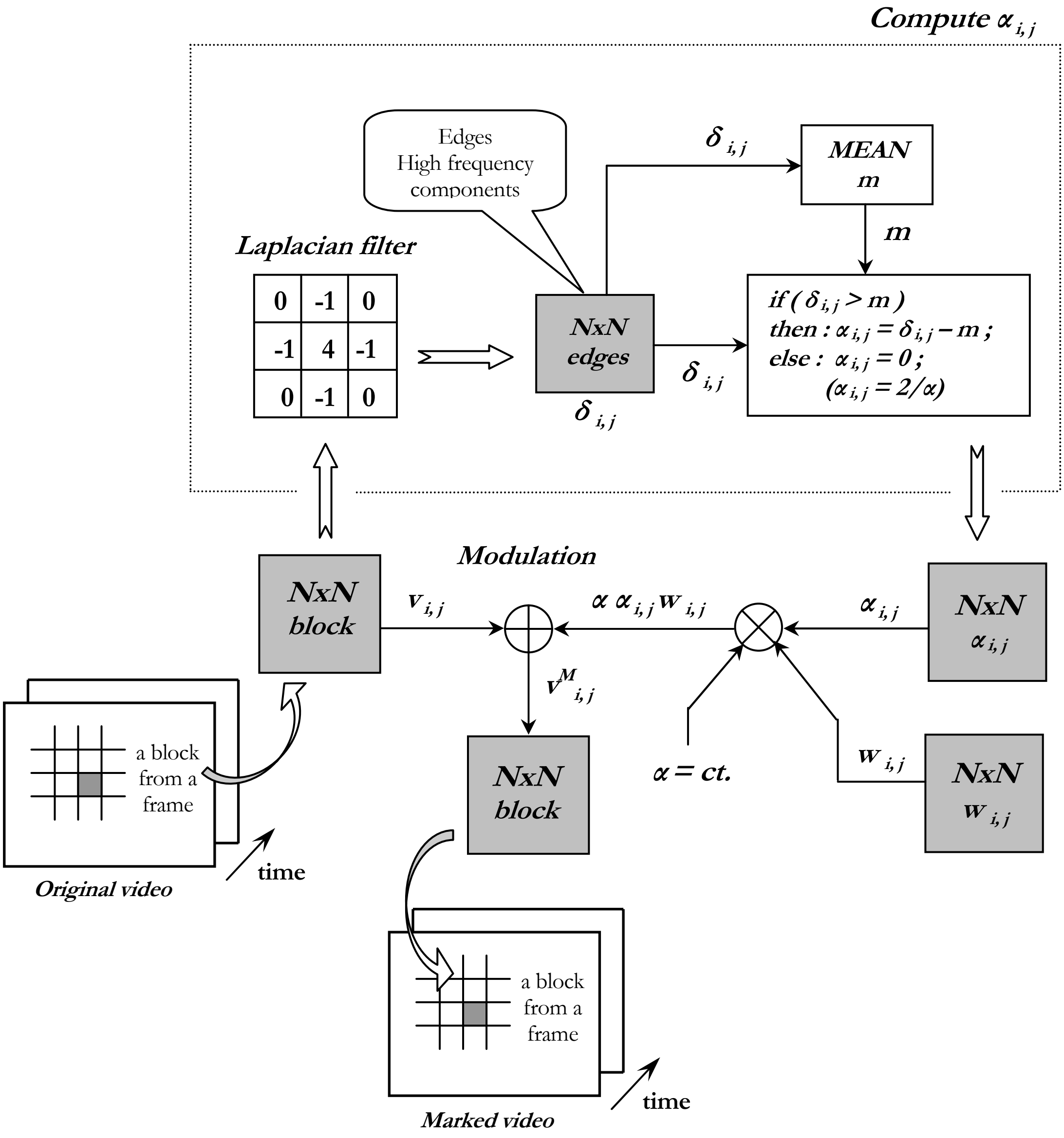


Figure 3-3 Block based "Edge marking" method

3.1.2 Adaptive Watermarking

The embedding algorithm presented in the previous section does not take into account the particularities of the video sequence. Each pixel is marked with the same constant strength, represented by the factor α in equation (3.3).

It is well known from the human vision theory that the eye is more sensitive to changes in the uniform areas of a picture compared with the same changes done in a high activity area which contains various edges and textures. On the other hand dark areas are more susceptible to visibility artefacts than light areas. This makes possible to insert a stronger watermark in the textures and edges while still maintaining the low visibility. Different image processing algorithms can be applied in order to derive the activity measure of an image. The best algorithms are obviously those who are capable of giving an activity measure estimate for each individual pixel rather than for an object, area or block. To account for this aspect, the factor $\alpha_{i,j}$ was introduced in **Figure 3-2** (with dotted line) leading to an activity based embedding

$$v_{i,j}^{M} = v_{i,j} + w_{i,j} = v_{i,j} + \alpha \cdot \alpha_{i,j} \cdot p_{i,j} \cdot b_{i,j} \tag{3.5}$$

where α now acts like a global visibility adjusting factor.

For computing the activity measure $\alpha_{i,j}$ two relatively simple algorithms are proposed: the "Edge marking" method and the "Gradient marking" method. In terms of performance and visibility, the edge marking gives better results compared with the gradient marking. This could be explain by the fact that the algorithm is in fact marking more the high frequency

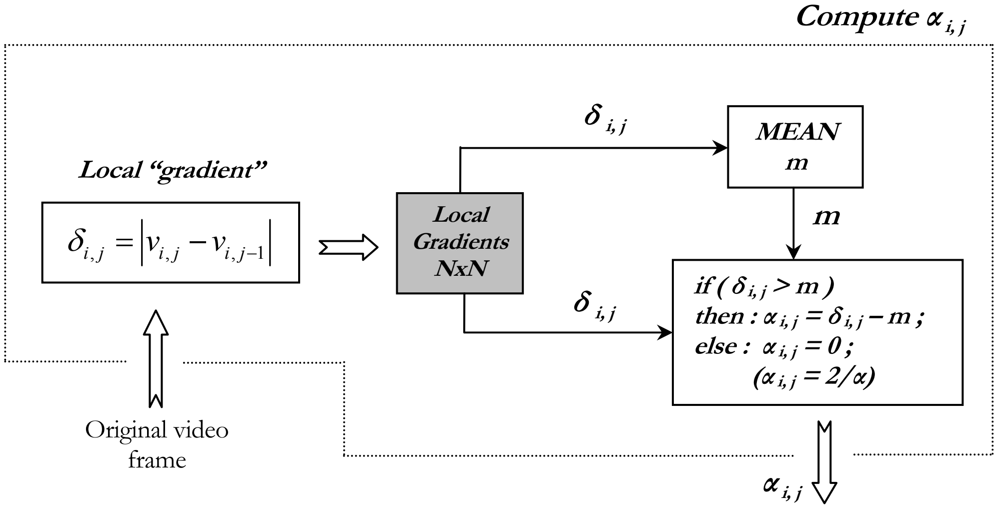
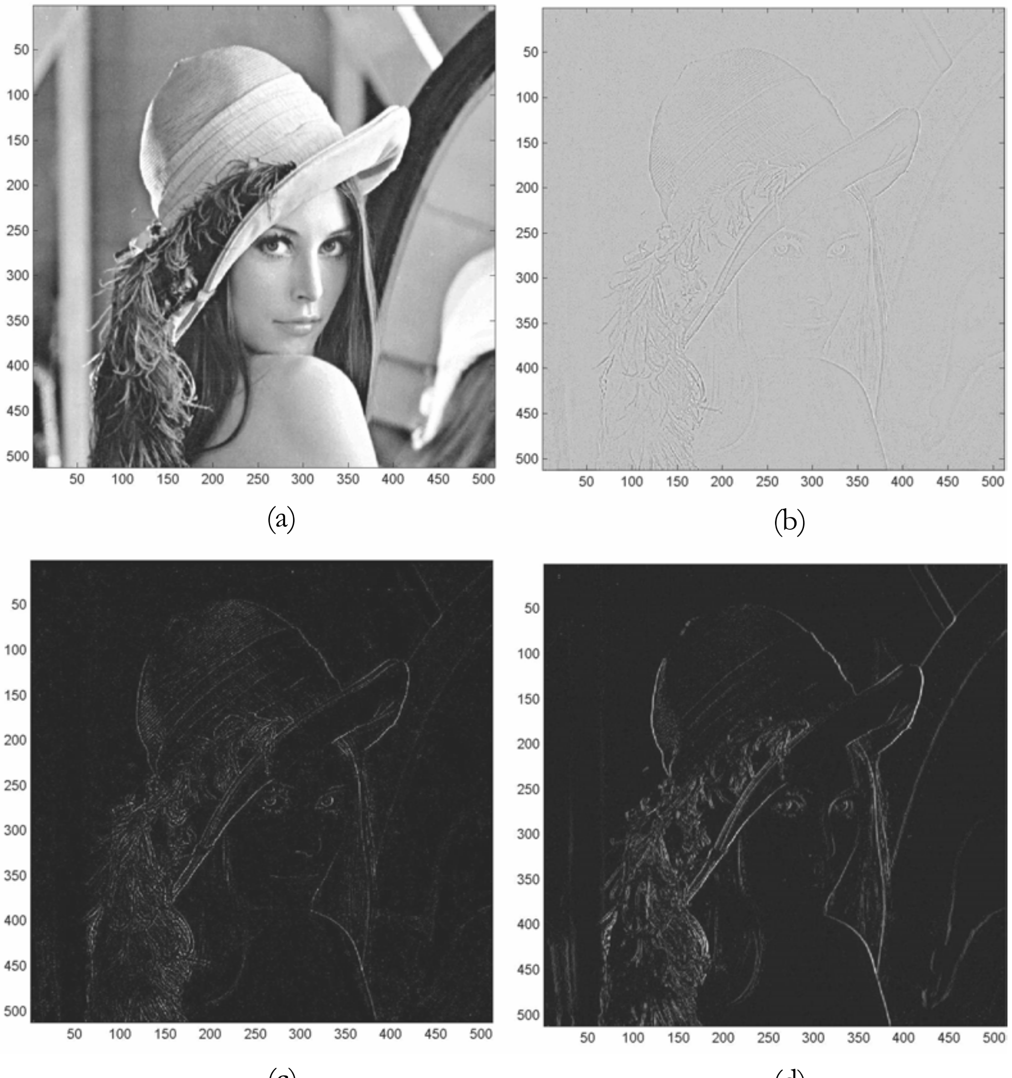

Compute $\alpha_{i,j}$
Local "gradient"
$\delta_{i,j} = |v_{i,j} - v_{i,j-1}|$
Local Gradients NxN
$\delta_{i,j}$
MEAN m
m
$\delta_{i,j}$
if ($\delta_{i,j} > m$)
then : $\alpha_{i,j} = \delta_{i,j} - m$;
else : $\alpha_{i,j} = 0$;
($\alpha_{i,j} = 2/\alpha$)
Original video frame
$\alpha_{i,j}$

Figure 3-4 Block based "Gradient marking" method

components of the video, due to the type of filter chosen as an activity measure.

Figure 3-3 shows the block scheme of the marking algorithm. The activity measure of the video is given by a high pass filter, in fact a Laplacian filter which acts as a non-directional edge detector. The values returned by the filter are first translated to positive integer values in the range 0…255 (not shown in the figure) and then processed following the algorithm described in the figure. The purpose of this operation is to derive an activity measure which assigns a higher $\alpha_{i,j}$ value to the highly textured areas and to the edges, while keeping $\alpha_{i,j}$ low for uniform areas which are very sensitive to noise. As **Figure 3-3** illustrates, this is done by computing the mean of the filtered values which is used both as a threshold and as limiting factor when a given value exceeds it. This limits the amplitude of the watermark inserted in the edges in order to keep the visibility of the mark at low levels. In the low detail regions and



(a) (b)

(c) (d)

Figure 3-5 Activity marking: **(a)** Original image, **(b)** the result after Laplacian filtering, **(c)** the factor $\alpha_{i,j}$ for "Edge marking" and **(d)** the factor $\alpha_{i,j}$ for "Gradient marking"

uniform regions, the algorithm uses a constant value usually set around a value of 2, which proves to be low enough for most pictures. The global factor α can be used to boost or lower the visibility of the watermark depending on the particular picture. The experiments show that the edge marking performs quite well in terms of visibility; for the same visibility of the watermark the edge marking scheme performs better than a uniform marking scheme.

A similar algorithm is described in **Figure 3-4**. This time instead of convolving the picture with a spatial filter, the activity measure is obtained simply by subtracting two additional pixels. The algorithm used to process the local gradients is identical with the one used in the first case. This method is simpler and quicker, but tends to amplify too much the strong edges and to reduce too much the weak edges. This leads to increased visibility of the watermark around edges and overall gives lower performance. For both methods the maximum value of the factor α is around 1/3.

Figure 3-5 shows a graphical representation of the factor $\alpha_{i,j}$ for the well known image "Lena", using both methods. It is easy to see, when one compares **(c)** with **(d)** that the gradient marking method accentuates too much the strong edges within the image in expense of the moderate edges and textures. This leads to visible artefacts around the significant edges within the image, and as a result the overall amplitude of the watermark has to be reduced, leading to a decrease in robustness.

3.2 Watermark Recovery

3.2.1 Retrieving of the Watermark

For a blind system, watermark retrieval follows basic spread-spectrum receiver theory and uses a matched filter (correlation detector) to extract the watermark bits from the received

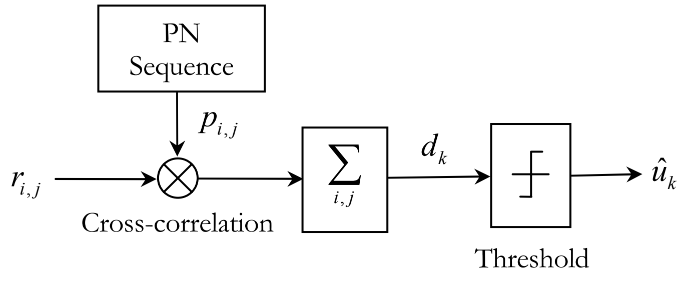


Figure 3-6 Watermark detection

video sequence. Essentially a matched filter is designed to maximise the SNR at a point in time. It does not preserve the input signal wave shape. This is not the objective of a matched filter. The objective is to distort the input signal wave shape and filter the noise so that at the sampling time the output signal level will be as large as possible with respect to the output noise level [Couch, 1987]. A particularly popular form of a matched filter is the correlation detector, often used for band pass signals. One example could be the detection of BPSK (Binary Phase Shift Keying) signals.

General schematic of the watermark detection process is illustrated in **Figure 3-6**.

Each data bit is extracted by cross-correlation of received sequence $r_{i,j}$ with the same PN sequence $p_{i,j}$ that was used for embedding, over a window of c_r bits. Assuming that the watermarked video was not attacked in any way, and therefore $r_{i,j} = v_{i,j}^M$, the detection process can be described as

$$\begin{aligned} d_k &= \sum_{(i,j)=kc_r}^{(k+1)c_r-1} p_{i,j} r_{i,j} \\ &= \sum_{(i,j)=kc_r}^{(k+1)c_r-1} p_{i,j} v_{i,j}^M \\ &= \sum_{(i,j)=kc_r}^{(k+1)c_r-1} p_{i,j} v_{i,j} + \sum_{(i,j)=kc_r}^{(k+1)c_r-1} \alpha b_{i,j} p_{i,j}^2 \end{aligned} \tag{3.6}$$

Assuming that the PN sequence $p_{i,j}$ and the original video sequence $v_{i,j}$ are uncorrelated over a large window c_r the first term from the equation (3.6) is close to zero and the equation (3.6) can be rewritten as

$$d_k \approx \sum_{(i,j)=kc_r}^{(k+1)c_r-1} \alpha b_{i,j} p_{i,j}^2 \tag{3.7}$$

As $p_{i,j}^2 = 1$ for a binary PN sequence and because over the window c_r the spread bits $b_{i,j}$ simply take on the value of the data bit u_k, the equation (3.7) becomes

$$d_k \approx \alpha c_r u_k \tag{3.8}$$

and u_k can be detected as

$$\hat{u}_k = sign(d_k) \tag{3.9}$$

by using a simple threshold.

However, in practice the first sum from equation (3.6) is not quite zero. Hartung suggests using a correction factor Δ which accounts for the different number of 1's and -1's in the PN sequence

$$\Delta = -\left(\sum_{(i,j)=kc_r}^{(k+1)c_r-1} p_{i,j} \right) mean\left(r_{i,j}\right) \tag{3.10}$$

As experimental results show, this correction leads only to marginal improvements (section 3.3).

The technique described above works very well only when the correct synchronisation between the PN sequence $p_{i,j}$ and the marked video sequence $r_{i,j}$ is maintained. Once an attack affects the synchronisation (the simplest way to "achieve" that is to cut a line/column or to slightly shift the video frame for example) the scheme is incapable of retrieving the watermark. This is a typical weakness of any spread-spectrum technique and obviously constitutes a major handicap for any robust watermarking scheme. In order to overcome this major problem one has to use a "sliding correlator" which effectively searches for the right peak or in other words re-establishes the correct synchronisation between the received sequence and the PN sequence.

The noise at the receiver, represented by the original video has much higher amplitude compared with the watermark itself and this is heavily affecting the cross correlation process. Since this component is not necessary for detection, it's a good idea to remove it. In fact, in communication systems is customary to use a pre-filter prior to cross-correlation as this improves the performance of the receiver. Following this reasoning, a pre-filter is employed prior to the cross-correlation process. Practically the received signal $r_{i,j}$ is high pass filtered in order to separate and remove the major components of the video signal itself. The filter used is a simple 3x3 spatial filter, with a Laplacian kernel. The Laplacian filters are characterised by the fact that the sum of their kernel coefficients is always zero which is equivalent to say that this filter rejects the DC component of the input signal. Therefore the filter will cut the low frequencies and the DC component of the video sequence, breaking the strong dependence between the cross-correlation process and these components. As the results will show (section 3.3.1), the filtering increases the performance of the system substantially.

3.2.2 Sliding Window Cross-Correlators

The importance of a sliding window correlator was emphasised in the previous section. Indeed one has to counteract the effects of the de-synchronisation attacks in order to have a robust system. At this time, only the case of a two dimensional (2-D) cross-correlator is considered. An improved version capable to tackle the 3-D case will be presented in Chapter 5.

The main problem of a sliding-window correlator is its complexity. Indeed even for the 2-D case the searching space grows very quickly with the "size" of the window. In other words, the sliding window has to move in a 2-D searching area defined by two parameters: a horizontal offset and a vertical offset specifying how much the window will slide left and right

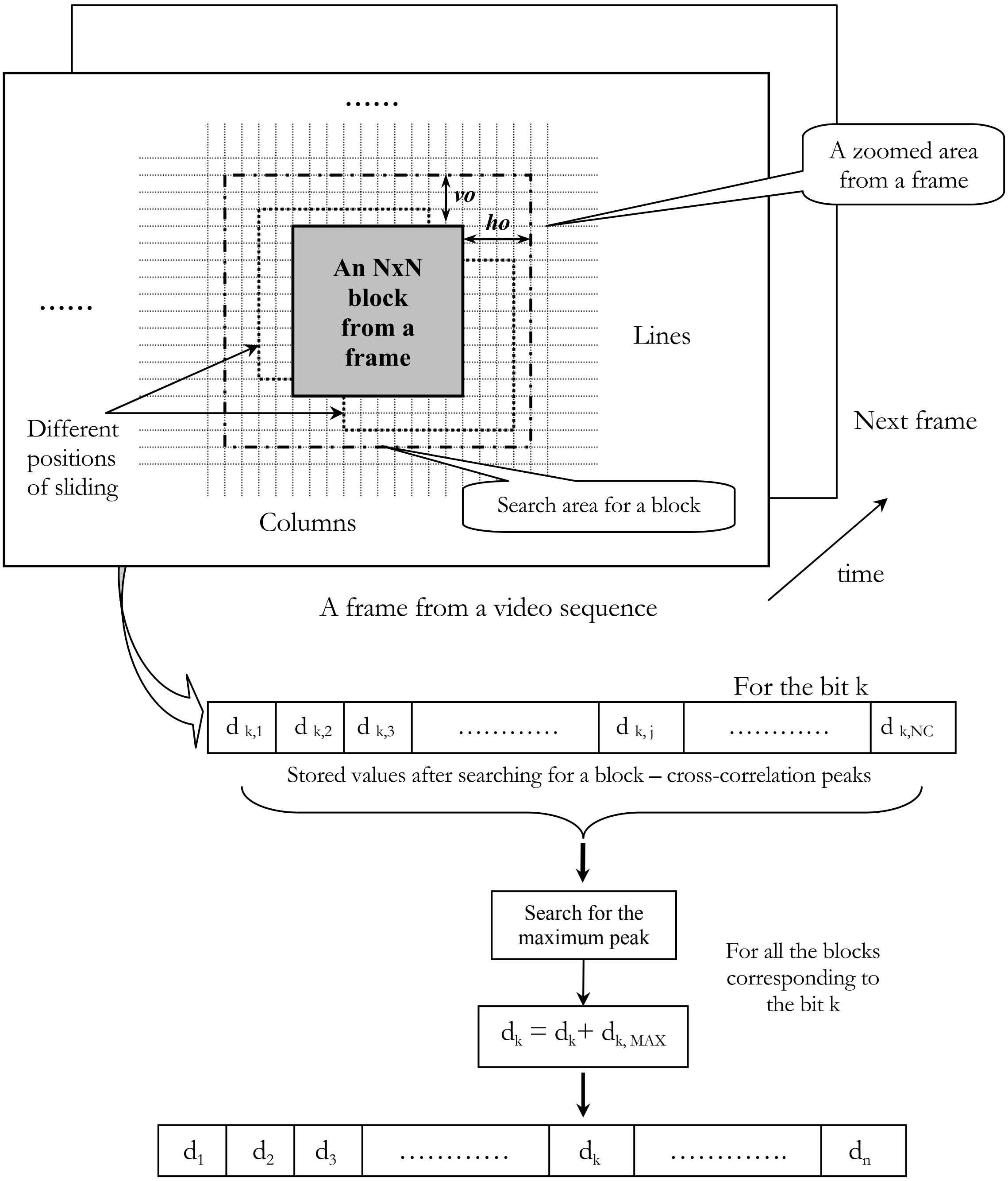

......
A zoomed area from a frame
vo
ho
An NxN block from a frame
......
Lines
Next frame
Different positions of sliding
Search area for a block
Columns
time
A frame from a video sequence
For the bit k
d k,1
d k,2
d k,3
....
d k, j
....
d k,NC
Stored values after searching for a block – cross-correlation peaks
Search for the maximum peak
For all the blocks corresponding to the bit k
dk = dk+ dk, MAX
d1
d2
d3
....
dk
....
dn
Cross-correlation values corresponding to the input bits

Figure 3-7 2-D sliding correlator

and respectively up and down of a reference point given by the position where the original block is expected to be, for both horizontal and respectively vertical locations.

Therefore the number of the cross-correlations which have to be performed can be defined as

$$NC = (2 \cdot ho + 1)(2 \cdot vo + 1) \tag{3.11}$$

where ho and vo represent the horizontal and respectively the vertical offsets, as described above. The schematic of the entire process is presented in **Figure 3-7**.

The algorithm performs a number NC of cross-correlations for each block within the current frame. All these NC intermediate values (cross-correlation peaks) are temporary stored in a buffer. When the buffer is completely filled, the maximum value is chosen as the peak corresponding to the correct position (the best matching). This search is performed for all blocks within the video sequence. For each block parsed like this, the algorithm knows the number of the bit embedded at that location and after finding the maximum value within the buffer this value is added to the sum of all previous values corresponding to that bit. When the algorithm finishes, the second buffer contains all the peaks corresponding to the input data bits. By analysing their signs it is possible to obtain (an estimate of) the watermark (input) data bits.

3.3 The Performance of the Scheme

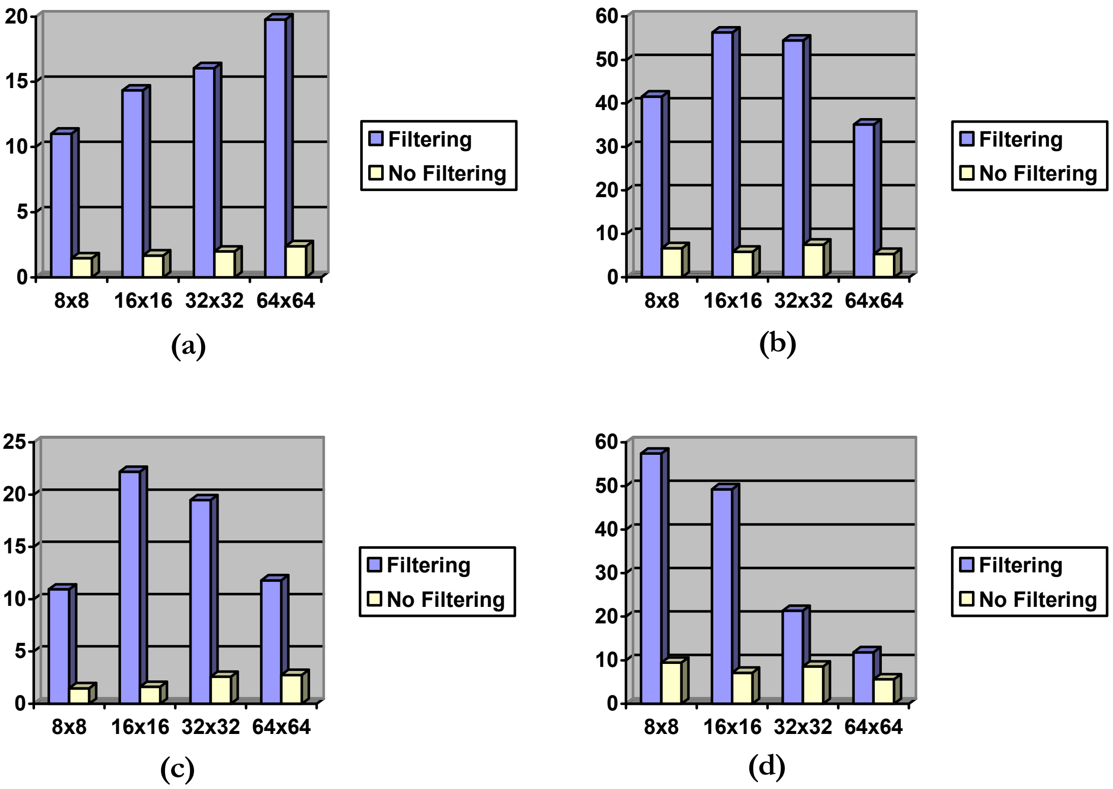
As is generally agreed, the spatial domain watermarking is not the most suitable (robust) domain for embedding a high capacity watermark, being rather weak compared with frequency domain approaches. Therefore this section will not present the results in terms of capacity or in terms of robustness to attacks since these results are nowhere near to those obtained by the frequency domain schemes. Instead, the main purpose of this section is to show some basic and yet very important aspects relative to spread spectrum embedding and recovery of the mark. The conclusions drawn here are general enough to constitute the basis of the most advanced frequency domain schemes and to be applied to these schemes in order to increase their performance.

Although due to the robustness and capacity issues of the spatial domain, such a scheme cannot be used as a highly robust, high capacity watermarking system, when the capacity is not a major requirement, these schemes can successfully be used as a very low capacity but highly robust system. One application of this case is to embed a reference watermark (1 bit watermark). This case is described in detail in Chapter 7 and its performance is analysed for a wide range of attacks.

This section will analyse several aspects like the effect of high pass filtering prior to cross-correlation, the effect of various block dimensions for both embedding and retrieving, and finally, the effect of 2-D sliding on the performance of the system. These results are presented and compared for both uniform marking and edge marking schemes. Most of these findings will constitute the basis for the more advanced DCT and DWT systems presented in Chapter 5 and respectively Chapter 6.

As **Figure 3-8** clearly indicates, pre-filtering drastically improves the performance of the system. Assuming that no sliding is performed – and this is the case illustrated in **(b)** and **(d)** – filtering improves the SNR up to 10 times for uniform marking and up to 6 times for edge marking. When sliding is performed – as shown in **Figure 3-8 (a)** and **(c)** – than the gain is up to 8 times for uniform marking and up 14 times for edge marking.

All the results presented in this section are for 25 frames (1 second) of "basketball" video sequence, for a length of the watermark of 64 bits. The amplitude factor α was set to 2 for uniform marking and to 0.2 for edge marking, which leads to a low visibility watermark. The vertical axis in **Figure 3-8**, **Figure 3-9**, **Figure 3-10** and **Figure 3-11** represents the SNR



(a)

(b)

(c)

(d)

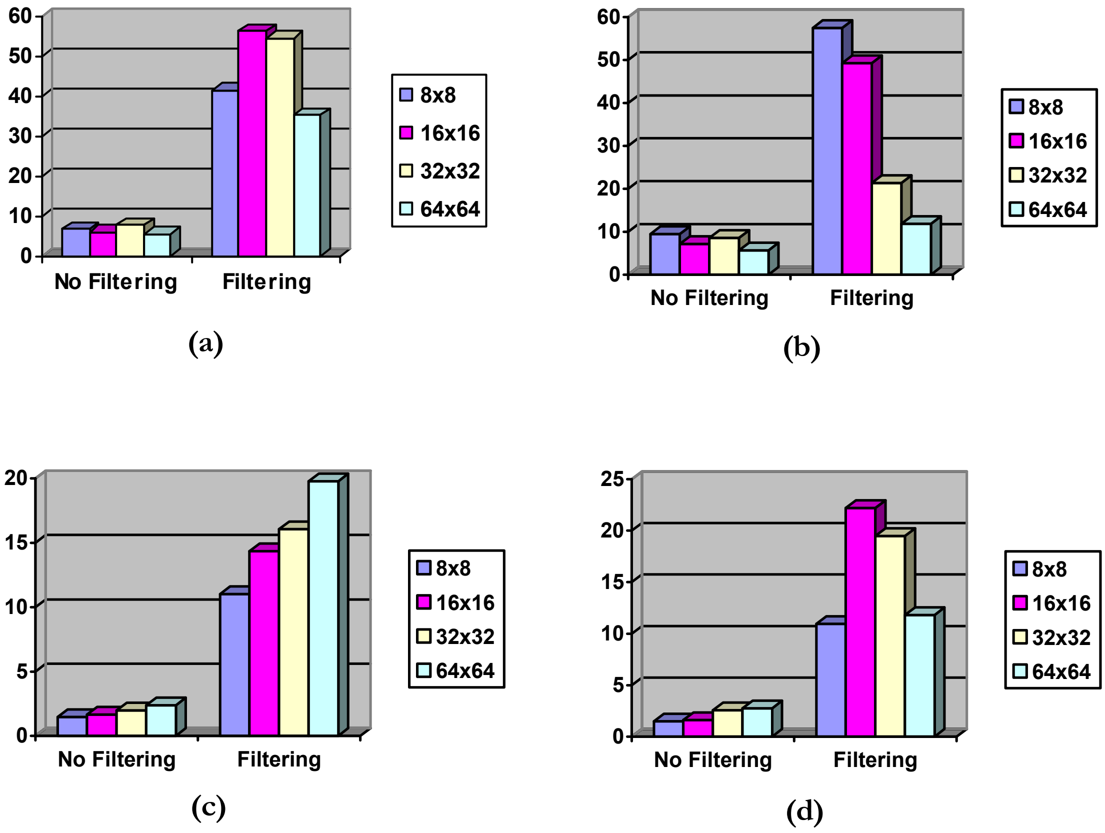
Figure 3-8 The effect of filtering for: **(a)** Uniform marking, 2x2 sliding, **(b)** Uniform marking, no sliding, **(c)** Edge marking, 2x2 sliding and **(d)** Edge marking, no sliding.

ratio of the cross-correlation peaks.

One advantage of using either 8x8 or 16x16 blocks is that 8 and 16 divides exactly the dimension of the frame which for ITU-R 601 video signals is 720x576. This is very convenient for implementation. Unfortunately this is not the case for 32x32 or 64x64. So rather than complicating the implementation the program will mark the nearest smaller area of the frame which in this case is 704x576. Therefore 16 lines are "lost", e.g. not marked. This reduces slightly the chip rate from 162000 to 158400. Anyway the difference in performance because of those 16 lines is not big enough to justify the effort of implementation. This is suggested by the results presented in **Figure 3-9 (a)**. It can be seen that the difference between the 16x16 and 32x32 cases is quite small.

Block size	8x8	16x16	32x32	64x64
Number of blocks per frame	6480	1620	396	99

Table 3-1 The number of blocks per frame for several block sizes.



(a) (b) (c) (d)

Figure 3-9 The effect of block size for: **(a)** Uniform marking, no sliding, **(b)** Edge marking, no sliding, **(c)** Uniform marking, 2x2 sliding and **(d)** Edge marking, 2x2 sliding.

Ignoring the non-filtering cases, it can be seen that the edge marking scheme performs increasingly worse as the block size increases - **Figure 3-9 (b)**. This is due to the visual model used which for higher block sizes is less and less adaptive. This suggests using an 8x8 block dimension for computing the visual model, even if the actual marking and recovery is done on a different block size. This approach was successfully followed for the DCT-based watermarking system presented in Chapter 5.

Another important aspect of choosing the block size can be observed in **Table 3-1**. The number of blocks per frame drops significantly (by a rate of 4 in this case) with the block size. This fact leads to only 99 blocks per frame for a block size of 64x64. This is fine for a relatively low number of data bits (<99), but taking into account that an entire block corresponds to only one data bit, if the length of the watermark is higher than 99 then there is not enough room in one frame to embed all the data bits. This complicates even more the embedding, could reduce the efficiency of the visual model, reduces the security of the algorithm and has serious robustness implications. For example in the case of line cuts, or column cuts, and for cropping, smaller block sizes lead to better performance.

The main advantage of choosing a higher block size is performance under sliding. As a

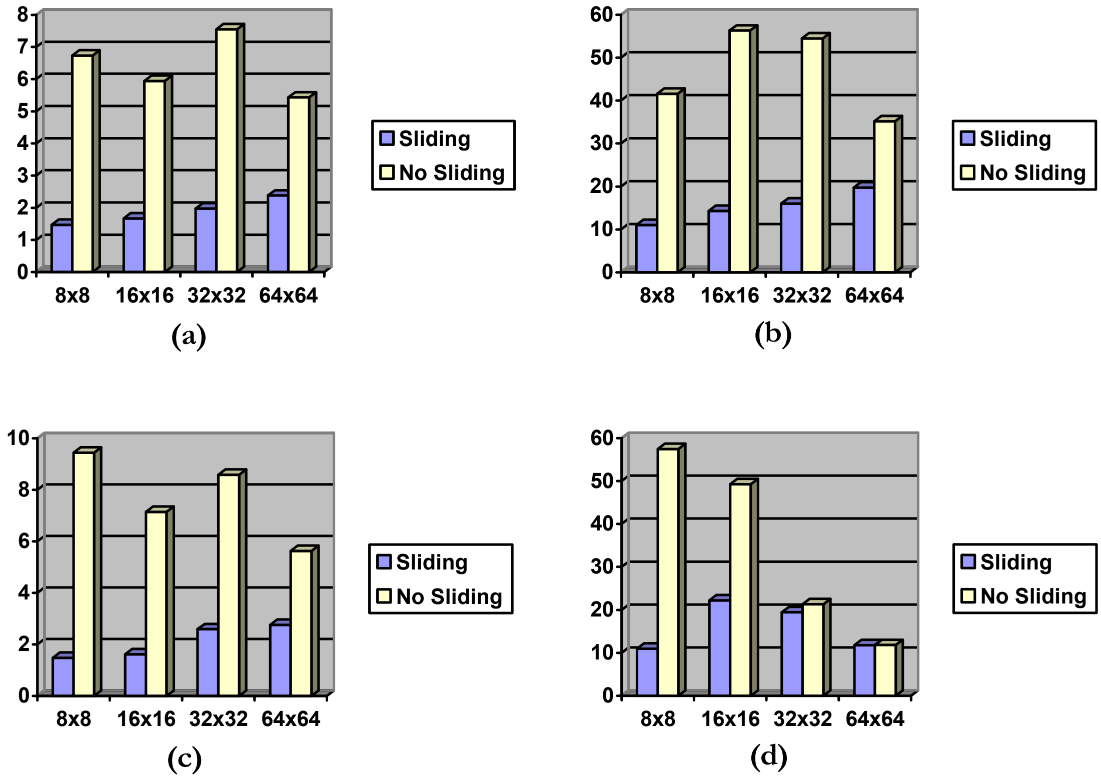


Figure 3-10 The effect of sliding for: **(a)** Uniform marking, no filtering, **(b)** Uniform marking, with filtering, **(c)** Edge marking, no filtering and **(d)** Edge marking, with filtering.

rule, the higher the block size the better the performance when a sliding correlator is used. This can be seen in **Figure 3-9 (c)**. For example just by using a block size of 16x16 instead of 8x8 the SNR increases with 27%. For a block size of 32x32 and 64x64 this percentage increases to 45% and respectively 79%. To resolve these conflicting aspects, one has to choose a compromise solution. The experiments show that a block size of 16x16 is a reasonable choice.

The effects of the sliding compared with the no-sliding situation are illustrated in **Figure 3-10**. The marked sequence wasn't attacked in any way, so these results suggest that the sliding correlator has an inherent loss even when the marked video was not attacked. Chapter 5 will show this aspect in more detail, together with the results of the sliding correlator for line/column cuts. This loss could be quite high as **Figure 3-10** shows. When no filtering is performed the loss can be up to 6 times for uniform marking and up to 5 times for edge marking. With filtering the loss is reduced to up to 4 times for uniform marking and remains approximately the same (5 times) for edge marking. **Figure 3-10** shows once more that the higher the block size is, the better is the performance which can be expected. Moreover, the difference between the sliding and non-sliding cases decreases as the block size increases.

Finally, the effect of average compensation is illustrated in **Figure 3-11**. As stated at the

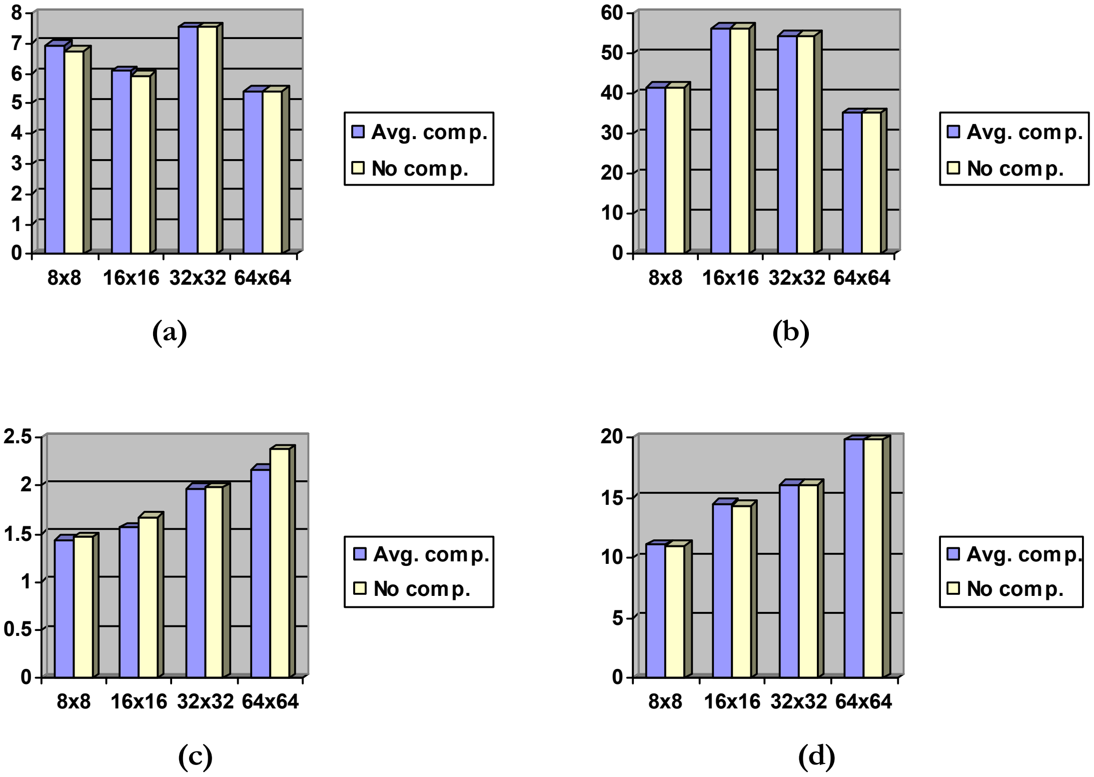


Figure 3-11 The effect of average compensation for uniform marking: **(a)** no sliding, no filtering, **(b)** no sliding, with filtering, **(c)** 2x2 sliding, no filtering and **(d)** 2x2 sliding, with filtering.

end of section 3.2.1, the average compensation doesn't really make a difference; in some instances the performance is even a little worse, although generally speaking average compensation improves very slightly the SNR. This improvement is so small that is not justifying its implementation in a practical scheme.

3.4 Conclusions

The results presented in the previous section show that a block based approach is preferred for a robust multi-bit watermarking system from several reasons. Robustness to attacks is perhaps the most important reason. Another advantage is that the HVS models tend to work better for smaller blocks. Finally, the security of the system is improved and overall the system is more flexible.

The HVS models improve the watermark invisibility and the robustness of the scheme. This is happening because the HVS model embeds more energy into the video sequence and therefore the SNR of the cross-correlation peaks will be higher.

The recovery of the watermark is greatly improved by pre-filtering the video sequence with a Laplacian kernel prior to cross-correlation. A simple 3x3 spatial filter improves the SNR of the cross-correlation peaks up to 10 times. The simulations show that the correction factor Δ suggested by Hartung is not effective, leading only to marginal improvements and therefore its presence is not justified.

In order to resynchronise a geometrically attacked video signal, a 2-D sliding correlator is needed. The downside of employing a sliding window correlator is its complexity; therefore sliding is a time consuming operation, difficult to implement in a real time system. Moreover, the sliding correlator has an inherent loss even in the case of un-attacked video; by using a sliding correlator the SNR can drop up to 6 times.

The dimension (size) of the block is a very important factor. The HVS model performs better for smaller blocks (8x8), while the recovery of the watermark works better for higher block sizes. The higher the block size, the better the performance of the system under sliding but higher block sizes are worse in terms of system's robustness, security and complexity and therefore a compromise has to be reached. As a compromise, a block size of 16x16 gives acceptable results. In conclusion it is better to use a mixed approach (Chapter 5), where marking is performed on smaller blocks (8x8) while the recovery is performed on higher block sizes (e.g. 16x16).