Multiple Blind Watermarking in the DRM-Context

Multiple Blind Watermarking in the DRM-Context

Diplomarbeit

zur Erlangung des Diplomgrades
an der Naturwissenschaftlichen Fakultät
der Universität Salzburg

eingereicht von

Christian Koidl, Bakk. techn.

Salzburg, September 24, 2009

Acknowledgement

Many people have helped me throughout my master thesis in one or the other way.

I want to thank my supervisor Andreas Uhl who guided me through my master thesis and supported me much more than a supervisor has to.

I also want to thank Peter Meerwald for his advice on watermarking topics and Martin Rieger who helped me out when my math knowledge came to an end.

Very special thanks go to my girlfriend Petra who encouraged me to keep up in the hard times of creating this thesis and always listend to me when I needed to talk, even if she only understood half of the things I said.

Thank you!

Abstract

Watermarks can be used in digital rights management as an answer to the recent developments in this sector. Digital media can be duplicated without loss of quality. Cheap mass storage devices and the ongoing advances in internet connection bandwith make the reproduction and distribution of multimedia data like images, music or video an ease. With the help of watermarks ownership information can be inseparably embedded into multimedia data to resolve copyright issues. Some applications like distribution chain tracking require the embedding of multiple watermarks.

This work is dealing with the benchmarking and improvement of the multiple watermarking abilities of watermarking algorithms. The first chapter starts with a short introduction into digital watermarking and especially multiple watermarking and some topics of wavelet analysis. In the second and third chapter all watermarking algorithms selected for our experiments are described in detail. Implementation details and their suitability for multiple watermarking and also possibilities for improvements are discussed and experimental results on the general performance and multiple watermarking are presented. The fourth chapter deals with the improvement techniques we have implemented. We show the idea behind the improvement techniques study the results when applied on the different watermarking algorithms. A new assessment criterion is introduced in chapter five, and results for selected algorithms with this new criterion are presented. In the last chapter a summary of our work on multiple watermarking improvements and the different assessment criterions is given.

Contents

List of Figures

Chapter 1

Introduction

1.1 Watermarking

Watermarking is the insertion of information into a cover media.

Watermarking can be used for many different purposes with partly opposed and maybe even conflicting properties. This makes a more precise definition of watermarking hardly possible. For many applications and also for our topic another definition could be:

Watermarking is the inseparable, imperceptible insertion of information about the cover media into the cover media.

Watermarking is a subfield of information hiding. Watermarking has its roots in the paper industry, where small logos or numbers have been inserted into paper. In digital watermarking, the cover media is most often images, videos, audio but can also be any other digital media like text, polygons or source code.

While watermarking on paper is a already old practice, which dates back to 1282 in Italy, digital watermarking has become a topic of research around 1995. With the upcome of digital recording devices, cheap high capacity storage and the Internet as a distribution medium, watermarking has been even more promoted. Classic cryptography can protect the data during the transmission to a trusted party. But for the transmission to an untrusted receiver classic cryptography fails to protect the data since the receiver can decrypt the data and then make copies of the unencrypted data. This is where watermarking comes into play. The watermark is inseparably embedded into the data and also stays there while consumption, i.e. viewing or listening. This means, unlike with cryptography, no key or other possibility to remove the watermark has to be provided to the customer.

Traditional watermarking was used to identify the producer of the paper or the production date. Digital watermarks can be used for many different purposes (see also [11] chap 2.1 or [25]):

Copyright protection a watermark is inserted, either visible or invisible, to identify the copyright holder or owner.

Fingerprinting here each distributed copy of the data is watermarked with a custom watermark identifying the receiver of the copy.

Broadcast monitoring a watermark is embedded into a video (e.g. an advertising clip) which contains identification information for the clip. So later on a watermark detection system can check whether the spot has been broadcasted or not.

Tamper detection for tamper detection a fragile watermark, i.e. a watermark which gets destroyed if the host media undergoes tampering, is inserted an so the integrity of the host can be checked.

Annotation here the embedded watermark contains meta information about the media.

Copy control the idea is, that each player has a watermark detector and the media contains standardized watermarks which encode messages like *DO NOT COPY* and every player should obey this messages. Also pay-per-view models would be implementable with this method.

Depending on the application, different properties are required. Watermarking algorithms can be characterized by their properties. These properties are sometimes conflicting with each other and there is often no general "best" value because this also depends on the application. The main properties are:

Perceptibility describes the perceptible difference between the watermarked and unwatermarked data. For most applications the watermark should disturb to cover data as little as possible. For visual watermarking, as the name already suggests, it is explicitly desired to induce some amount of distortions (e.g. [23]).

Robustness refers to the ability to withstand common signal processing operations, i.e. the watermark is still detectable in spite of the cover medium modifications. For images common signal processing operations are for instance compression, blurring, printing and scanning,

format conversion or geometrical transformations like translation, cropping or scaling. For applications like copyright protection the watermark should be as robust as possible. On the other hand for tamper detection the watermark may not be robust to indicate manipulations, such watermarks are called fragile watermarks.

Security denotes the property to withstand intentional attacks. These are attacks which sole aim is to remove the watermark, where removing does not necessarily mean the inversion of the watermarking process but to damage the watermark that much that it is not detectable any more while preserving as much image quality as possible. Another possibility to remove the watermark is to bring the detector out of synchronization, e.g. through small geometrical transforms, so that the watermark is still in the image but the detector can not "find" it. An intentional attack can also be the unauthorized embedding of a watermark. Especially with tamper detection this is a concern.

Blind or non blind detection if a watermark detector needs the original image to detect the watermark it is called a non-blind detector and a detector which does not need access to the original image is called blind. There is also a category called semi-blind detectors. They do not need the original image, but some other information e.g. a secret key. Usually non-blind detectors are more robust but impractical for some applications.

Capacity is the amount of data which can be embedded into the cover media. For this property generally the more is the better. However some applications do not need a high capacity, so it is the better choice to sacrifice some capacity in favor of other properties.

Embedding domain Watermarking algorithms can also be characterized by the domain in which they embed. The first watermarking algorithms embedded directly in the signal domain. For instance in the LSB of the data points. This technique has very low complexity and is easy to implement and we have a direct localization of the watermark, allowing special treatment for regions of interrest, but they are not very robust for a given embedding strength. But the watermark can also be embedded after transformation of the cover media into some other domain. This can for instance be the Fourier- or the DCT-domain like used in the JPEG compression standard or the wavelet domain like used in JPEG2000. The advantage in using a transformation domain is that the watermark is usually more robust by spreading it over the

whole asset, but on the other hand sacrificing some or all localization abilities. Another advantage is that, especially in case of the wavelet transform, the HVS can be exploited to achieve better invisibility or better robustness through perceptual masking techniques. If the image is given in the signal domain, additional computational effort is necessary to perform the transformation and inverse transformation. Or if the signal should be converted to some other format (e.g. JPEG), watermarking can be applied during or immediately after the conversion in that domain. Or the image is already available in the transform domain then, of course the watermarking algorithm in the signal domain has the additional effort of the transformations and the transform domain algorithms come cheaper.

An in depth introduction and further reading can be found in [11] and [1].

1.2 Multiple watermarking

1.2.1 Why multiple watermarking?

Why would someone want to embed multiple watermarks? There are different possible scenarios. As we have seen above, watermarks can be used for different applications. So the first scenario is to embed multiple watermarks for different purposes, e.g. one for copyright protection, one for integrity verification and an annotation watermark. Another scenario is to embed multiple watermarks for a single purpose. This can for instance be one watermark for the copyright holder and one for the consumer or in the case where there are multiple owners. A third scenario is distribution chain tracking. Like with fingerprinting a watermark identifying the receiver of the media is embedded. If the media gets sold/transmitted multiple times, a watermark is embedded for every receiver and maybe also for the original owner of the media. (see also [24])

1.2.2 How to embed multiple watermarks

Sheppard et al. describe several possibilities to embed multiple watermarks in [33].

Composite watermarking Multiple watermarks are not embedded separately but combined to form a single (composite) watermark which gets embedded. This approach has the need for a trusted party which

does the composition and embedding of the single watermarks and all watermarks have to be present at once.

Segmented watermarking The cover media gets partitioned into several blocks and every single watermark is embedded into one block. Disadvantage is, that the maximum number of watermarks has to be known in advance. Furthermore the location of the embedding blocks have to be open to the embedders and each embedder has to know which blocks are already occupied or as an alternative like with the composite watermarking approach a trusted embedder which records all occupied blocks has to do the embedding.

Re-watermarking is a very straight forward approach. The watermarks are simply embedded one after the other. The problem which arises is that the watermarks interfere with each other earlier embedded watermarks possibly get erased by later embedded ones. The advantage is that no central party for embedding is necessary and the embedder parties do not need to know each other and also the number of watermarks to embed does not need to be known in advance. If multiple watermarks for multiple purposes are embedded the watermarks have different robustness. Copyright protection has to be very robust and on the other side integrity verification watermarks have to be fragile. Therefore the most robust watermark has to be embedded first and the watermarks with less robustness later. Otherwise the copyright protection watermark would erase the integrity verification watermark.

1.3 Scope of our work

Our work is dealing with the benchmarking and improvement of different watermarking algorithms with respect to their suitability for multiple watermarking. All algorithms in the test are embedding in the wavelet domain. We limit our test on the benchmarking and improvement of the re-watermarking approach. Our assumed scenario is the distribution chain tracking, so we only use copyright protection watermarks, i.e. robust watermarks. For reasons we give in section 1.4 we concentrate on blind watermarking algorithms, i.e. without refering to the original image.

We use grey level images as the cover medium.

1.4 Previous work

Hartmut Wernisch studied multiple watermarking in his master thesis [36]. His studies included blind and non-blind algorithms. But there are several problems when using non-blind algorithms. First, if we want to use exactly the image before embedding as the original image in the detection stage, all intermediate images have to be stored. This means if we want to extract the fourth embedded watermark we need the image with the first three watermarks embedded for the extraction. Another possibility Wernisch examined is the usage of the original image, i.e. the image without watermarks, but it turned out that this approach does not work very well. Independently of that in some applications there is no way of implementing non-blind watermarking, e.g. if a DVD-Recorder has to read a watermark there is no possibility to support the detector with the original data. Therefore we concentrate on blind algorithms only in our work.

Wernisch applies two improvement techniques to selected algorithms in his work (parametrized filters and random wavelet packet structures). We extend the improvement studies with two other techniques (random carrier modulation and random coefficient selection).

1.5 Wavelet transform

Because all algorithms we use are working in the wavelet domain we want to give a short introduction into wavelet transform. This introduction is far from complete, in fact we only cover basic topics of the discrete wavelet transform (DWT). For a (more) complete introduction on wavelet analysis and informations on special topics of wavelets see [13, 19, 38, 37, 34, 9].

The wavelet transform has been developed to overcome certain limitations of the Fourier transform. These are first, the Fourier transformation assumes the signal to be periodic (which is not the case for most signal processing applications, especially in image processing) and therefore is inappropriate for signals which are localized in space. The second limitation is that the Fourier transform does not preserve any information of the localization of the signal in space.

Unlike the Fourier transform the wavelet transform does not use sine and cosine functions, but so called wavelets which are localized in space.

The wavelet transform is also used in the JPEG2000 compression standard [8]. This codec has better image quality, especially with low bit rates than the established JPEG codec which uses the discrete cosine transform (DCT). The benefit of the wavelet transform over the Fourier transform or

also the DCT is first the complexity. The fast wavelet transform (FWT) has a complexity of $O(n)$ where as the fast fourier transform (FFT) and also the (fast) DCT have a complexity of $O(n\ log(n))$. Furthermore, as we will later see in this section, the wavelet transform is a multi resolution transform where the different resolutions split the signal into frequency octaves, very similar as the human eye does and the DWT also provides information about edges and textures which can also be used in human visual system (HVS) considerations.

1.5.1 Discrete wavelet transform

In the discrete wavelet transform the wavelet functions are implemented by quadrature mirror filter (QMF) banks which provide perfect reconstruction of the signal. One filterbank consists of a low-pass filter (usually denoted as h) and a high-pass filter (usually denoted as g). The signal is convolved with the filters and thereby is split into a low frequency and a high frequency part. Both parts are decimated by 2, i.e. every other value is discarded. The low frequency part can be seen as an approximation of the signal (a) and the high frequency part can be seen as the residual details of the image (d).

$$a(n) = \sum_m h(m) \cdot x(2n - m)$$

$$d(n) = \sum_m g(m) \cdot x(2n - m)$$

where x is the signal, n is half the number of values of x and m is the number of values in h or g.

To reconstruct the signal the same operations are carried out in inverse direction. First the approximation and the detail component is upsampled by the factor 2, this is done by inserting a 0 after every coefficient. Then the upsampled signals are convolved with the reconstruction filterpair g' and h' and the reconstructed signal is obtained by the addition of these two signals. The basic structure is visualized in figure 1.1.

The above procedure is one decomposition step and the according reconstruction. This procedure is recursively applied on the approximation component. This leads to several detail components called detail subbands and one approximation component called approximation subband or approximation image when dealing with image processing. Because of the resulting structure this is also called the pyramidal wavelet transform (in opposition to the wavelet packet transform, see section 1.5.2). The procedure for three decomposition and reconstruction steps is shown in figure 1.2. In figure 1.3

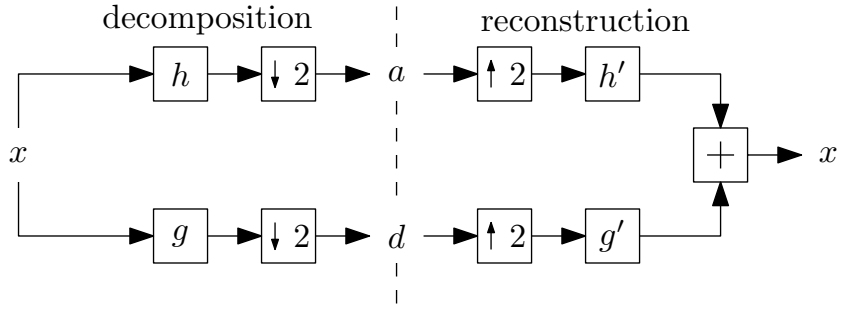


Figure 1.1: one wavelet decomposition and reconstruction step

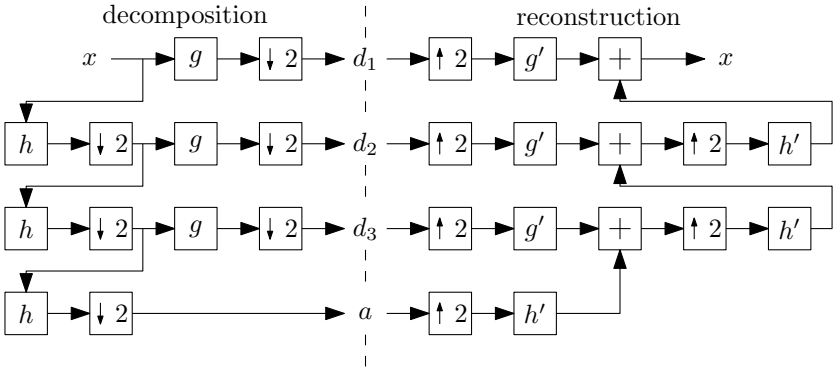


Figure 1.2: the wavelet decomposition and reconstruction steps of a three level wavelet transformation

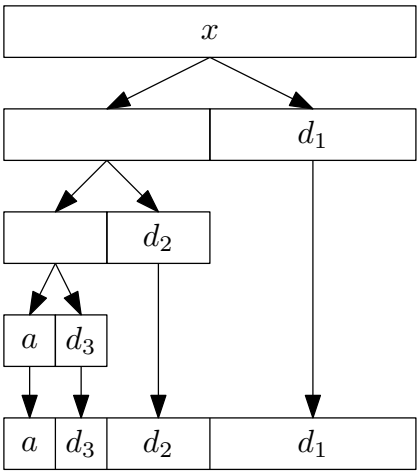


Figure 1.3: the resulting wavelet decomposition structure for one dimensional data

the resulting wavelet decomposition structure is sketched. We can observe the property, that the source signal x and the sum of the resulting subbands a, d_3, d_2, d_1 have the same length.

The extension to the two dimensional case is pretty simple. First the lines are filtered and afterwards the columns or vice versa. The resulting structure is shown in figure 1.4. L denotes the low pass filtering and H the high pass filtering. Accordingly LL is the approximation subband which is low pass filtered in both directions, horizontal and vertical. HL is the subband containing the horizontal details (therefore this subband is also denoted as h) which are obtained by horizontal high pass filtering and vertical low pass filtering. The HL subband are the vertical details (also denoted as v). The coefficients in this subband are obtained by low pass filtering the rows and high pass filtering the columns. The last subband HH are the diagonal details (d), which are high pass filtered in both directions. For a multi level two dimensional wavelet decomposition the LL subband is recursively decomposed.

The resulting structure of a three level two dimensional wavelet transformation is shown in figure 1.5. The index of the detail subbands is the level of the decomposition. The higher detail level, the coarser the image elements. A decomposed image is shown in figure 1.6. The subbands are scaled to fit the grey level range. Otherwise the detail subband would be mostly black and the approximation image would be plain white. This is because in natural images most energy is contained in the low frequency parts and

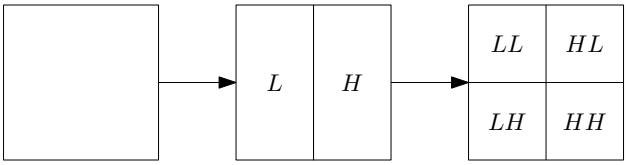


Figure 1.4: the two filtering steps of a two dimensional wavelet transform

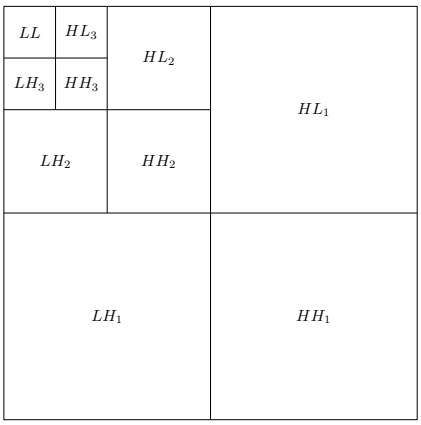


Figure 1.5: the resulting structure of a two dimensional, three level wavelet decomposition

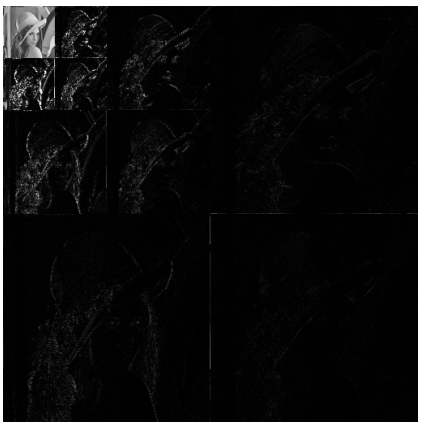


Figure 1.6: the wavelet tree of a 3 level decomposition of the Lena image

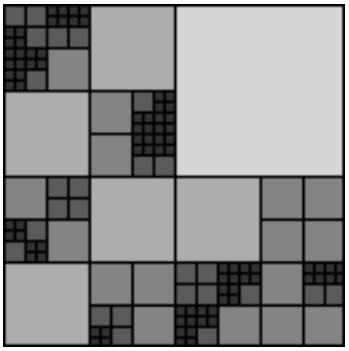


Figure 1.7: example wavelet packet decomposition structure

therefore the energy packed in the higher decomposition levels, especially in the approximation image.

1.5.2 Wavelet packet transform

In contrast to the usual (pyramidal) wavelet transform where only the LL subband is further decomposed, all subbands can be further decomposed in the wavelet packet transform. We use the wavelet packet transform for our improvement technique described in section 4.1.4. For our improvement we only use isotropic wavelet packets, this is every subband which gets decomposed is decomposed like described in the previous section. In opposite in the anisotropic wavelet packet transformation there is also the possibility to decompose only the lines and not the columns of one subband or vice versa.

An example (isotropic) wavelet packet tree can be seen in figure 1.7. The tree has a maximum decomposition decomposition level of 5 and the structure is randomly generated.

As described in section 4.1.4 we only use key dependent random generated wavelet packet structures. In most other applications the structure is not randomly generated. Also the complete decomposition of all subbands up to a specific level is not practical. One often used decomposition strategy is the best basis algorithm of Coifman et al. [10]. The reader is redirected to this resource. We do not go into details on different decomposition strategies since we do not use them in our work.

1.5.3 Redundant wavelet transform

Another form of the wavelet transform we use in our work (see section 2.3) is the redundant discrete wavelet transform (RDWT), also called stationary

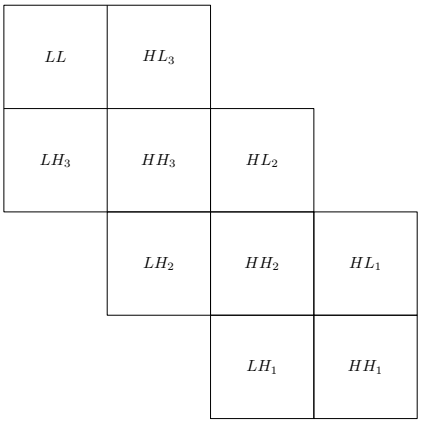


Figure 1.8: the resulting structure of a three level stationary wavelet decomposition

wavelet transform (SWT) or undecimated wavelet transform. It is basically the same as the DWT but with the difference that there is no subsampling step. This includes that each subband has the same size as the original image. The resulting structure can be visualized like in figure 1.8.

Therefore the multiresolution analysis is not done by the subsampling of the signal but through upsampling of the filters which are used in each step. This algorithm is called “algorithme à trous”, because of the “holes” in the resulting filters.

Chapter 2

Algorithms

In this chapter selected watermarking algorithms are presented. All algorithms embed the watermark in the wavelet domain and the watermark can be detected without refering to the original image (blind algorithms). For each algorithm a summary of the embedding and extraction process is given. Furthermore initial considerations about the suitability for the use with multiple watermarking are made and the applicability of several improvements for multiple watermarking are investigated.

2.1 Algorithm by Mauro Barni

In "Improved wavelet based watermarking through Pixel-wise masking" [2] Barni et al. present an algorithm that embeds a watermark in the wavelet domain and takes special care about HVS modeling. The embedding strength is controlled by a mask in order to keep the watermark invisible. Mask generation is based on the work of Lewis and Knowles [18]. The authors state that the proposed system is resistant to JPEG and wavelet-based compression, median filtering, Gaussian noise addition, cropping plus zero padding, morphing and multiple marking.

2.1.1 embedding

In the embedding stage the image is first decomposed by a four level DWT. However, the watermark is only embedded in the detail-subbands of the first decomposition level. The other levels are needed for mask generation. The watermark is then embedded as follows:

$$\tilde{I}_0^\theta(i,j) = I_0^\theta(i,j) + \alpha w^\theta(i,j) x^\theta(i,j)$$

where $\tilde{I}_0^\theta(i,j)$ is the modified coefficient, $I_0^\theta(i,j)$ is the unmarked coefficient, $\theta \in \{0,1,2,3\}$ denotes the orientation of the subband i.e. the hl, hh, lh and ll subband, α is a global parameter accounting for watermark strength (Barni et. al use a α of 1.5 for their experiments), $w^\theta(i,j)$ is a weighting function described later on and $x^\theta(i,j) \in \{+1,-1\}$ is the watermark consisting of a pseudorandom binary sequence arranged in 2-D to scan the subbands to be marked. The arrangement can be done by

$$x^\theta(i,j) = m_{(\theta MN+iN+j)}$$

with m being the pseudorandom sequence of the watermark, $2M \times 2N$ as the image size and $\theta \in \{0,1,2\}$.

The weighting function is calculated by

$$w^\theta(i,j) = q_0^\theta(i,j)/2$$

where

$$q_l^\theta = \Theta(l,\theta)\ \Lambda(l,i,j)\ \Xi(l,i,j)^{0.2}$$

Each of the three terms models a HVS consideration. The first term $\Theta(l,\theta)$ takes account of the fact that the eye is less sensitive to noise in high resolution bands and noise in the diagonal band.

$$\Theta(l,\theta) = \left\{ \begin{array}{ll} \sqrt{2} & \text{if } \theta = 1 \\ 1 & \text{otherwise} \end{array} \right\} \cdot \left\{ \begin{array}{ll} 1.00 & \text{if } l = 0 \\ 0.32 & \text{if } l = 1 \\ 0.16 & \text{if } l = 2 \\ 0.10 & \text{if } l = 3 \end{array} \right\}$$

The second $\Lambda(l,i,j)$ is used to model that the eye is less sensitive to noise in areas where the brightness is very low or very high. This is expressed with

$$\Lambda(l,i,j) = 1 + L'(l,i,j)$$

with

$$L'(l,i,j) = \left\{ \begin{array}{ll} 1 - L(l,i,j) & \text{if } L(l,i,j) < 0.5 \\ L(l,i,j) & \text{otherwise} \end{array} \right\}$$

and

$$L(l,i,j) = \frac{1}{256} I_3^3 \left(1 + \left\lfloor \frac{i}{2^{3-l}} \right\rfloor, 1 + \left\lfloor \frac{j}{2^{3-l}} \right\rfloor \right)$$

The last term $\Xi(l,i,j)$ gives a measure of the texture activity in the neighborhood of the pixel considering that the eye is more noise-sensitive near edges but less in highly textured areas. This is computed with

$$\Xi(l,i,j) = \sum_{k=0}^{3-l} \frac{1}{16^k} \sum_{\theta=0}^{2} \sum_{x=0}^{1} \sum_{y=0}^{1} \left[I_{k+l}^\theta \left(y + \frac{i}{2^k}, x + \frac{j}{2^k} \right) \right]^2 \cdot$$

$$\cdot Var\left\{I_3^3\left(1+y+\frac{i}{2^{3-l}},1+x+\frac{j}{2^{3-l}}\right)\right\}\begin{array}{l}x=0,1\\y=0,1\end{array}$$

where the first part is the local mean square value of all detail subbands at position (i,j) calculating the proximity to edges and the second value is the variance of the approximation image at the position corresponding to (i,j) giving the measure for the local texture activity.

2.1.2 extraction

The watermark extraction process is blind and done via the computation of the correlation between the marked coefficients and the watermark to be extracted.

$$\rho = \frac{1}{3MN}\sum_{\theta=0}^{2}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\tilde{I}_0^{\theta}(i,j)x^{\theta}(i,j)$$

The watermark is said to be present if the correlation is above a specific threshold T_ρ. T_ρ is chosen in a way to grant a given probability of false positive detection. The threshold can be calculated with the help of the Neyman-Pearson criterion

$$P_f \leq \frac{1}{2}erfc\left(\frac{T_\rho}{\sqrt{2\sigma_{\rho B}^2}}\right)$$

with $erfc$ being the complementary error function defined as:

$$erfc(z) = 1 - erf(z) = \frac{2}{\sqrt{\pi}}\int_z^{\infty} e^{-t^2}dt.$$

For a false detection probability of less than 10^{-8}, T_ρ would be

$$T_\rho = 3.97\sqrt{2\sigma_{\rho B}^2}$$

where $\sigma_{\rho B}^2$ can be estimated with

$$\sigma_{\rho B}^2 = \frac{1}{(3MN)^2}\sum_{\theta=0}^{2}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(\tilde{I}_0^{\theta}(i,j))^2$$

2.1.3 multiple watermarking

In the basic algorithm as proposed in the paper filter parametrization can be applied but will only have limited impact because only the first decomposition level is used for embedding. Wavelet packets can not be used because the mask generation can not be done and moreover makes no difference when only the first decomposition-level is marked. Usage of random coefficients is also a possibility of improvement.

As an straight forward approach to make the watermarking algorithm more suitable to the multiple watermarking enhancements, more decomposition levels can be used for watermarking. This will most likely improve the performance of the filter parametrization- and the random coefficient-approach directly. The usage of wavelet packets instead of the pyramidal decomposition requires the adaption of the visual mask. Therefore in the first term Θ the amplification due to the orientation of the subband has to be changed in a way that it not only amplifies every hh-subband but every subband which represents diagonal image components. The second term Λ can be left unchanged. In the last term Ξ the variance-factor of the approximation subband can be kept while the local mean square-factor has to be modified, to reflect the proximity to edges in the image. This could for instance be done by first taking the pyramidal decomposition, calculating the factor and then further decompose the pyramidal tree into wavelet packets. Because embedding in more subbands will increase distortions, the watermark strength parameter α has to be adapted to the new embedding scheme.

Another approach to make the algorithm suitable for the use with wavelet packets is to only mark the three detail subbands of the first decomposition level of the pyramidal decomposition but decompose these subband further into wavelet packets. The same changes to the visual mask as in the last approach have to be made.

2.2 Broken Arrows

"Broken Arrows" [17] is the watermarking algorithm used in the BOWS2-Contest. Influenced by different fields [20, 26, 21, 3] the authors make up a proportional-embedding, zero-bit algorithm, that means no message is embedded and one can only say the image is marked or not. Embedding takes place in the wavelet domain, although two more subspaces are used for watermark creation. The major part of the algorithm is done in these two subspaces.

2.2.1 embedding

First a 3 level wavelet decomposition of the image is taken. All subbands except the approximation image are used for the watermarking process.

But first the watermark is created, therefore the signal is transformed in the, what the authors call, secret- or correlation-subspace. The transformation is done by first generating N_v secret carrier signals($S_{C,j}$) of size N_s. The authors choose a value of 256 for N_v and N_s is given by

$$N_s = W_i H_i (1 - 1/64)$$

where W_i and H_i is the width and height of the image and $1 - 1/64$ is the factor for subtracting the approximation-subband. For a 512x512 image N_s would be 258048. The secret carriers have the form

$$S_{C,j} \in \{-1/\sqrt{N_s}, 1/\sqrt{N_s}\}^{N_s}$$

These carriers are produced by a PRNG seeded with the secret key K. The authors assume that the carriers are (pseudo-)orthogonal. This could be ensured by a Gram-Schmidt orthogonalization but this step was skipped because it would be too time consuming for the contest. The signal is now projected on these carriers via

$$v_X(j) = S_{C,j}^T s_X$$

with $v_X(j)$ as a coefficient in the secret subspace and s_X as the signal in the wavelet subspace.

Next the signal is transformed in the MCB-plane. The name MCB comes from the authors of [22], which is the first paper suggesting that host, watermark and secret should belong to one plane. The basis of this plane is given by (v_1, v_2) such that

$$v_1 = v_C^*$$

and

$$v_2 = \frac{v_X - (v_X^T v_1) v_1}{||v_X - (v_X^T v_1) v_1||}$$

with v_C^* as a vector in the secret subspace. Later on we will see, that this vector is the axis of a cone, this cone is the detection area.

To improve the security through adding diversity, v_C^* is taken out of a set of possible vectors (cones). The authors choose a number of 30 cones and because the subspace is already secret they take the first 30 elements of the canonical basis of the subspace. Out of these vectors the "nearest" to the host signal is taken:

$$v_C^* = sign(v_C^T v_X) v_C$$

with

$$v_C = arg \max_{k \in [1,N_c]} |v_X^T v_{C,k}|$$

where N_c is the number of cones i.e. 30, $v_{C,k}$ is the k^{th} vector out of the set of possible vectors and v_X is the host signal. In this subspace the host is represented as

$$c_X = \begin{pmatrix} c_X(1) \\ c_X(2) \end{pmatrix}$$

with

$$c_X(1) = v_X^T \, v_1$$

and

$$c_X(2) = v_X^T \, v_2$$

This subspace offers a clear representation of the original signal, the watermarked signal and the detection boundary.

As said before, the original signal is given by c_X and the cone-shaped detection area is given by its axis $v_1 = v_C^*$ and its angle θ. θ is calculated to provide a given false-positive ratio. The authors choose a false-positive probability of at most $3 \cdot 10^{-6}$. The equation to calculate the false-positive probability is [21]:

$$P_{fa} \leq N_c \frac{I_{N_v-2}(\theta)}{I_{N_v-2}(\pi/2)}$$

where $I_{N_v-2}(\theta)$ is the solid angle associated to θ in the N_v-dimensional space. The authors calculated a θ of about 1,2154.

Embedding is done by moving the original signal as deep as possible inside the cone. Therefore we have to distinguish two cases:
$c_X(2) \leq \rho cos\theta$: this case means the host signal is so close to the cone, that the marked signal is put on the axis of the cone and as far away as possible along the axis. In this case embedding is done by

$$c_Y = \begin{pmatrix} c_X(1) + \sqrt{\rho^2 - c_X(2)^2} \\ 0 \end{pmatrix}$$

The other case is that $c_X(2) > \rho \, cos\theta$ and the host is moved orthogonally to the edge of the cone:

$$c_Y = c_X + \rho \begin{pmatrix} sin(\theta) \\ -cos(\theta) \end{pmatrix}$$

ρ denotes the radius of the embedding circle, that means the the distortion through the embedding process may not exceed ρ. With a chosen PSNR the

radius can be calculated with:

$$\rho = \frac{\overline{|S_X|}}{\sqrt{\overline{S_X^2}}} 255 \sqrt{W_i H_i} 10^{-PSNR/20}$$

The authors chose a target PSNR of 43dB for their implementation for the contest. Finally the watermark signal in the MCB-plane is

$$c_W = c_Y - c_X.$$

The watermark signal has now to be projected back in the secret subspace and subsequently in the wavelet space where embedding takes place. This is done via

$$s_W = S_C(c_W(1)v_1 + c_W(2)v_2) = S_C(v_1, v_2)c_W$$

Now, having the watermark signal and the host signal in the wavelet space, the host is watermarked by:

$$S_Y = S_X + S_{W_p}$$

where

$$S_{W_p}(i) = |S_X(i)|s_w(i)$$

2.2.2 extraction

To decide whether or not the image is watermarked, the image has to be transformed into the MCB-plane (see 2.2.1). Basically a vector c is considered to be marked if

$$\frac{|c(1)|}{||c||} > cos(\theta)$$

with θ as the angle of the cone.

The authors propose some improvements for the extraction to better withstand some attacks. First they randomize the detection boundary in the range $[cos((\theta_{max} + \theta_{min})/2), cos(\theta_{max})]$ according to the idea of [7]. θ_{max} and θ_{min} are the angles according to a false-positive probability of $3 \cdot 10^{-6}$ and $3 \cdot 10^{-7}$. This should interfere with sensitivity attacks, because it is harder to obtain the "sensitive vector" when there is no clear detection border.

Secondly they implement an improvement the authors call "snake traps", because the improvement is a counter measure to a form of oracle attack which is called "snake" [12]. For this counter measure the detection- border is altered in the following way:
If $|c_Y(1) - \Delta\lfloor\frac{c_Y(1)}{\Delta}\rfloor| < r$ then detection is positive if

$$c_Y(1) > ||c_Y||cos(\theta_{min})$$

else, the watermark is detected if

$$c_Y(1) > ||c_Y||cos(\theta)$$

This makes some "notches" in the detection border, where the width of these notches is given by r and the periodicity is given by Δ. θ is the angle obtained from the first improvement. The authors choose $r = 4.5$ and $\Delta = 30$.

The last improvement concerning the algorithm itself is the "camouflage of the cone" where the tip of the cone is truncated, that means if $||c_Y|| < \lambda$ the detection is alway negative. Since only specific kinds of detectors are completely resistant to valuemetric scaling (e.g. manipulating the brightness of the image), the truncation is done to leave less clues on the detector which can be exploited like in [12]. The authors use a λ of 10.

Improvements concerning the BOWS-Contest but not directly the algorithm are not mentioned here.

2.2.3 multiple watermarking

Since the algorithm is a zero-bit algorithm it is not possible to embed different messages, but the algorithm can be applied multiple times. This means it can not be used for identification (fingerprinting) but for verification purposes. The algorithm spreads the watermark in an unpredictable (in wavelet- and spatial-domain) way across the image. One the one hand this makes different watermarks overwrite itself at least partially, on the other hand the embedding is key dependent and different embedders will embed into different coefficients.

Theoretically, parametrized wavelets can be used to improve multiple watermarking abilities, but it has to be tested if this leads to more robustness against watermark interference than only using different keys for the creation of the secret-subspace. Also wavelet packets could be used, resulting in a different wavelet-space-vector. But again it has to be tested if the approach gives an additional advantage. Using randomized coefficients can be combined with the standard technique by only passing a subset of the wavelet coefficients to the secret space projection and adaption of the secret-space-carriers to the new size. The authors describe a problem, that due to proportional embedding and the distortion limit, the algorithm does not spread the watermark all over the image when there are uniform areas in it, but force it on a few large coefficients. This problem could be intensified by this approach.

2.3 Algorithm by Jian-Guo Cao

"An image-adaptive watermark based on a redundant wavelet transform" [5] describes an watermarking algorithm that exploits the special ability of the redundant discrete wavelet transform (RDWT) in extracting significant features of an image. The authors assume, that salient features have rather big coefficients across all scales and make use of the fact, that the RDWT has the same sampling rate over all decomposition levels. So they build a "significance mask" by multiplying coefficients at adjacent decomposition levels. The strength of the watermark is then adjusted according to this mask.

2.3.1 embedding

Embedding of the watermark is basically done as:

$$f'_{\{V,H,D\}}(x,y) = f_{\{V,H,D\}}(x,y) + \alpha v_{\{V,H,D\}}(x,y) w_{\{V,H,D\}}(x,y)$$

where $f'_{\{V,H,D\}}(x,y)$ is the watermarked coefficient in the vertical, horizontal or diagonal subband at position (x,y). $f_{\{V,H,D\}}(x,y)$ is the according unmarked coefficient, α is a global strength parameter of the watermark, $v_{\{V,H,D\}}(x,y)$ is the watermark value to embed and $w_{\{V,H,D\}}$ is the significance mask for the different detail subbands. The authors embed the watermark in the detailsubbands of the first decomposition level and use a white gaussian noise sequence as the watermark. α is chosen to achieve a given PSNR.

Although the authors embed the watermark only in the highest level, they use the first two decomposition levels to calculate the significance mask. Mask generation is done by

$$w_{i\{V,H,D\}}(x,y) = \prod_{j=1}^{i} f^*_{j\{V,H,D\}}(x,y)$$

with $f^*_{j\{V,H,D\}}(x,y)$ denoting the normalized coefficients at scale j given as

$$f^*_{j\{V,H,D\}}(x,y) = \frac{f_{j\{V,H,D\}}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f_{j\{V,H,D\}}(x,y)^2}$$

The scaling of the watermark values by this mask puts more watermark energy in the visually significant coefficients in order to improve the robustness of the watermark.

2.3.2 extraction

To detect the watermark, the RDWT is taken and a correlation value for each watermarked detail subband is calculated

$$\rho_s = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f'_s(x,y) v_s(x,y)$$

with $f'_s(x,y)$ being the wavelet coefficient of the received image and $v_s(x,y)$ is the watermark-value to detect. The final detection value is the maximum of the per-subband-detection values.

$$\rho = max_{\{s=V,H,D\}}(\rho_s)$$

By taking the maximum detection value of each direction the algorithm should get more robust against attacks that affect only a specific direction.

2.3.3 multiple watermarking

To improve multiple watermarking capabilities parametrized wavelets can be applied and also choosing random coefficients could result in better results. To be able to use wavelet-packet decomposition an appropriate way of generating the visual mask has to be found.

2.4 Algorithm by Tung-Shou Chen

In the paper of Chen et al. "A simple and efficient watermarking technique based on JPEG2000 Codec" [6] an algorithm that can be applied in the JPEG2000 coding process is proposed. Watermarking takes place after quantization and before the entropy coding stage. The authors state, that the algorithm has low computational cost although has a high capacity and low visual distortions of the cover image.

2.4.1 embedding

The algorithm uses a black/white image as watermark (the authors use an image of size 32x32). First the watermark is scattered through a torus automorphism [35]. This is done by applying

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ k & k+1 \end{pmatrix}^n \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} mod\ N$$

where $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ are the coordinates of the original watermark bit, $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ are the coordinates of the scattered watermark bit, k is the torus automorphism parameter, N is the width of the watermark image and n is the private scatter key. This step makes the watermark visually undetectable. The scattered watermark is first embedded the LL subband then in LH,HL,HH of the smallest detail, then in the detail subbands of the next stage and so on, depending on the size of the watermark. To embed a bit of the watermark one coefficient from these subbands is taken. This coefficient consists of a sign-bit, an integer part and a decimal part. For embedding the integer part of the coefficient is used and

$$M_b = \lfloor I_n \times \alpha_b \rfloor$$

is calculated. M_b denotes the bit in which the watermark will be embedded, I_n is the number of bits of the integer part in the coefficient and $\alpha_b \in [0, 1]$ is the weighting factor for the embedding strength. The M_b bit will be replaced by the bit to be embedded. The authors describe a method called distortion reduction. This method is applied after embedding the watermark bit and is done by altering the other bits in the coefficient in a way, that the difference between the original coefficient and the marked coefficient is kept low. In their work the authors use 3 (random) alteration possibilities and choose the one with the least distortion.

2.4.2 extraction

To extract the watermark, the same order of coefficients as in the embedding stage has to be chosen. The single bits are extracted using the same formula as in the embedding stage. The watermark-image is restored by applying the torus automorphism with the private restore key as n obtained from the scattering process in the embedding stage. The private restore key is calculated by subtracting the private scatter key from the recurrence time of the torus automorphism. The recurrence time is a value $<= N$ and depends on k and N.

Besides visual decision if the watermark is present or not, the authors use the normalized correlation (NC) as a measure for the presence of the watermark.

$$NC = \frac{\sum_i \sum_j w(i,j) w'(i,j)}{\sum_i \sum_j [w(i,j)]^2}$$

where $w(i, j)$ is the original watermark coefficient and $w'(i, j)$ is the extracted one. The authors do not define a threshold for the decision if the watermark

is present or not but make the decision by comparing the correlation from different watermarks and watch for a single high peak.

2.4.3 multiple watermarking

Since the algorithm relies on single bits in the codestream and the watermark is directly written into these bits it is most likely that a subsequently embedded watermark will erase the preceding, especially when distortion reduction is applied. Embedding into random coefficient is a very straightforward approach to reduce watermark interference. Parametrized wavelets could also work well, but wavelet packets are a less promising approach, because the algorithm always starts with embedding in the approximation image which is the same for every wavelet packet decomposition. A very easy modification of the algorithm is to start embedding in the detail subbands instead of the approximation image. This makes the adoption of wavelet packets promising.

2.5 Algorithm by Oriol Guitart Pla

Pla et al. propose a blind watermarking scheme in "A wavelet watermarking algorithm based on a tree structure" [29], which is based on the work of Dugad [15]. Their scheme embeds a watermark value not only in a single coefficient but rather in a tree of coefficients depending on the significance. Additionally a synchronization template is embedded to be able to estimate and compensate geometric attacks.

2.5.1 embedding

First the image is wavelet transformed. The authors perform a three level DWT. The smallest detail subbands are used to embed the watermark. All coefficients grater than a threshold T_1 are used for embedding. The authors use a $T_1 = 40$. A coefficient is represented as

$$s_k = (b_k, x_k, y_k, v_k)$$

where b_k denotes the subband, x_k and y_k are the coordinates in the subband and v_k is the value of the coefficient. Then for each coefficient a PRNG is seeded with

$$seed = f(K_E, b_k, x_k, y_k)$$

Because the PRNG is reseeded with every coefficient and the seed is only dependent on the embedding Key K_E and the position of the coefficient it is

possible for the detector to resynchronize at each coefficient. This gives the algorithm more robustness against attacks. The coefficient s_k is marked by

$$v'_k = v_k + \alpha_k |v_k| p_{k,0}$$

where v'_k is the marked coefficient, v_k is the unmarked coefficient, α_k is a scaling factor for visual masking described later on and $p_{k,0}$ is the first random number obtained from the PRNG. After marking a coefficient all direct children of this coefficient are also marked

$$v'_{k,i} = v_{k,i} + \alpha_k |v_{k,i}| p_{k,i}$$

with $v_{k,i}$ being the unmarked coefficient of the ith child of s_k, $v'_{k,i}$ is the marked version of the aforementioned coefficient and $p_{k,i}$ is another value from the PRNG. If one of these children is greater than $T_1/2$ the children of this coefficient are also marked with the same rule as above. The threshold is halved with every step and watermarking is continued until none of the children are above the threshold or the biggest subband is reached.
When all significant coefficients and their significant children are marked, the inverse wavelet transformation is applied. As the last step in the embedding procedure the authors describe embedding a synchronization template in the DFT domain.

The template embedding scheme is based on the work of Pereira et al. [27] and Caldelli et al. [4]. The template consists of 2 sets of 4 points arranged in a line (collinear), intersecting the origin. The cross-ratio is calculated and passed to the detector. The detector uses the cross-ratio to distinguish between the template peaks and peaks from the original image. We are not going closer into the detail because the authors concede, that the template scheme does not work well, because sometimes, through the discretization of the peak positions, it is not possible to find the 4 strictly collinear points again.

The authors use a visual model, that controls the embedding strength via the scaling factor α_k. The visual model increases the embedding strength in high textured or edge regions of the image. To determine α_k an activity image A has to be created. The activity image has the same dimensions as the smallest subband and is obtained by

$$a(x,y) = \begin{cases} 1 & \text{if } |v_2(x,y)| > T_1 \text{ or } |v_3(x,y)| > T_1 \text{ or } |v_4(x,y)| > T_1 \\ 0 & \text{otherwise} \end{cases}$$

where $a(x,y)$ is a pixel of the activity image at position (x,y) and $v_n(x,y)$ is a coefficient of one of the smallest detail subbands n at position (x,y). To

calculate the scaling factor α_k a $N \times N$ window W (the authors suggest N to be 3 or 5) is slid over the activity image and

$$Z_k = \sum_{a(x,y) \in W} a(x, y)$$

is calculated. α_k is then obtained by

$$\alpha_k = \begin{cases} \gamma\alpha & \text{if } Z_k = 1 \text{ or } Z_k \geq N^2 - 1 \\ \alpha & \text{otherwise} \end{cases}$$

where α is a global scaling factor and $\gamma > 1$ is the amplification factor when an edge or high textured area is detected. The authors use $\alpha = 0.2$ and $\gamma = 2$.

2.5.2 extraction

As the first step of the extraction process the authors describe the detection of the synchronization template and the reversal of a possible geometric transformation. Again we will not go into details because of the aforementioned weaknesses of the approach.

Like in the embedding stage the wavelet transformation of the received image is applied. Afterwards, all significant coefficient of the smallest detail subbands are determined using a threshold $T_2 \geq T_1$. The authors use a $T_2 = 50$. The watermark values according to the determined coefficients and their significant children are generated using the PRNG with the detection key K_D. Now the correlation between the selected coefficients of the received image and the watermark values of the last step can be calculated

$$Z = \frac{1}{M} \sum_{i=1}^{M} \hat{V}_i y_i$$

where $\hat{V}_i$ is the ith significant coefficient, y_i is the corresponding watermark value and M is the number of coefficients. The correlation value is then compared to a threshold S

$$S = \frac{\alpha}{2M} \sum_{i=1}^{M} |\hat{V}_i|$$

The watermark is said to be present if $Z \geq S$.

2.5.3 multiple watermarking

The only possibility to improve the algorithm without constraints is to use parametrized wavelets. To enable the usage of wavelet packets an adapted parent-child relationship has to be introduced, for example like in [30], to build up the tree. Random wavelet coefficient selection can be applied in principle. A problem can arise from the rather small capacity of the algorithm. Only the large ($> T_1$) coefficients of the smallest detail subbands are used as root elements to embed the watermark values. Sharing these coefficients will amplify this shortage. As a workaround the threshold T_1 can be lowered or the watermark could be embedded in multiple passes, where the first pass embeds the watermark like proposed by the authors and every following pass embeds in the subbands at the next finer scale where not all coefficients came into consideration by comparison with T_1. In this pass all coefficients in the selected subbands are compared with the threshold for the according scale and if they are above, they are marked with the same procedure as in the first pass.

2.6 Algorithm by Gin-Der Wu

In "Image watermarking using structure based wavelet tree quantization" [39] Wu et al. propose a blind quantization based watermarking algorithm. They group coefficients in trees and quantize these trees into a specific structure that reflects an embedded 1 or an embedded 0. The authors claim that the algorithm is robust against multiple watermarking and confirm this assertion with experiments.

2.6.1 embedding

The authors employ a three level wavelet transformation and use the six smallest detail subbands. The coefficients in these subbands are grouped into trees where a tree is formed from one coefficient of the smallest subband and his four direct descendants. These trees are combined to super trees where 2x2 coefficients from the smallest subband and the associated child-coefficients are grouped together. The 2x2 coefficients from the smallest subband form a subblock of the supertree and the four descendants from each of these coefficients form the other subblocks. These supertrees are denoted by $\zeta_{n,m}(i)$ where $1 \leq n \leq 3072$ (for a 512x512 image) is the index of the super tree, $1 \leq m \leq 5$ is the index of the subblock and $1 \leq i \leq 4$ is the index of the coefficient in the subblock. A supertree is quantized by quantizing all subblocks in the supertree.

To quantize a single subblock the average of the upper two coefficients $\zeta_{n,m}(1)$, $\zeta_{n,m}(2)$ denoted as $\sigma_{n,m}(up)$ and the lower two coefficients $\zeta_{n,m}(3)$, $\zeta_{n,m}(4)$ denoted as $\sigma_{n,m}(lo)$ is calculated and the difference between the two is

$$dif = |\sigma_{n,m}(up) - \sigma_{n,m}(lo)|$$

Furthermore a quantization step Δ is chosen. The goal is to quantize the coefficients so that

$$\begin{array}{ll} \sigma'_{n,m}(up) > \sigma'_{n,m}(lo) & \text{if } W_n = -1 \\ \sigma'_{n,m}(up) < \sigma'_{n,m}(lo) & \text{if } W_n = 1 \end{array}$$

where W_n is the watermark value to embed. Now the authors distingushed some cases. First assumption is that $W_n = -1$.
First case is that $\sigma_{n,m}(up) = \sigma_{n,m}(lo)$ then

$$\begin{array}{ll} \zeta'_{n,m}(i) = \zeta_{n,m}(i) + \Delta/2 & \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) = \zeta_{n,m}(i) - \Delta/2 & \text{if } i \in [3,4] \end{array}$$

If $\sigma_{n,m}(up) > \sigma_{n,m}(lo)$ and $dif \geq \Delta$

$$\zeta'_{n,m}(i) = \zeta_{n,m}(i) \qquad \text{if } i \in [1,2,3,4]$$

If $\sigma_{n,m}(up) > \sigma_{n,m}(lo)$ and $dif < \Delta$

$$\begin{array}{ll} \zeta'_{n,m}(i) = \zeta_{n,m}(i) + (\Delta - dif)/2 & \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) = \zeta_{n,m}(i) - (\Delta - dif)/2 & \text{if } i \in [3,4] \end{array}$$

In case that $\sigma_{n,m}(up) < \sigma_{n,m}(lo)$ and $dif \geq \Delta$

$$\begin{array}{ll} \zeta'_{n,m}(i) = \zeta_{n,m}(i) + dif & \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) = \zeta_{n,m}(i) - dif & \text{if } i \in [3,4] \end{array}$$

and if $\sigma_{n,m}(up) < \sigma_{n,m}(lo)$ and $dif < \Delta$

$$\begin{array}{ll} \zeta'_{n,m}(i) = \zeta_{n,m}(i) + (\Delta + dif)/2 & \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) = \zeta_{n,m}(i) - (\Delta + dif)/2 & \text{if } i \in [3,4] \end{array}$$

Now in case that a 1 is embedded $W_n = 1$
and $\sigma_{n,m}(up) = \sigma_{n,m}(lo)$ then

$$\begin{array}{ll} \zeta'_{n,m}(i) = \zeta_{n,m}(i) - \Delta/2 & \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) = \zeta_{n,m}(i) + \Delta/2 & \text{if } i \in [3,4] \end{array}$$

Next case is $\sigma_{n,m}(up) > \sigma_{n,m}(lo)$ and $dif \geq \Delta$

$$\begin{aligned} \zeta'_{n,m}(i) &= \zeta_{n,m}(i) - dif \qquad \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) &= \zeta_{n,m}(i) + dif \qquad \text{if } i \in [3,4] \end{aligned}$$

If $\sigma_{n,m}(up) > \sigma_{n,m}(lo)$ and $dif < \Delta$

$$\begin{aligned} \zeta'_{n,m}(i) &= \zeta_{n,m}(i) - (\Delta + dif)/2 \qquad \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) &= \zeta_{n,m}(i) + (\Delta + dif)/2 \qquad \text{if } i \in [3,4] \end{aligned}$$

if $\sigma_{n,m}(up) < \sigma_{n,m}(lo)$ and $dif \geq \Delta$

$$\zeta'_{n,m}(i) = \zeta_{n,m}(i) \qquad \text{if } i \in [1,2,3,4]$$

Last case is that $\sigma_{n,m}(up) < \sigma_{n,m}(lo)$ and $dif < \Delta$ then

$$\begin{aligned} \zeta'_{n,m}(i) &= \zeta_{n,m}(i) - (\Delta - dif)/2 \qquad \text{if } i \in [1,2] \\ \zeta'_{n,m}(i) &= \zeta_{n,m}(i) + (\Delta - dif)/2 \qquad \text{if } i \in [3,4] \end{aligned}$$

2.6.2 extraction

To detect the watermark in a single subblock the ratio between the upper average value and the lower average value is considered. The detector differentiates 3 states

$$D[\zeta'_{n,m}] = \begin{cases} -1 & \text{if } \sigma'_{n,m}(up) > \sigma'_{n,m}(lo) \\ 1 & \text{if } \sigma'_{n,m}(up) < \sigma'_{n,m}(lo) \\ 0 & \text{if } \sigma'_{n,m}(up) = \sigma'_{n,m}(lo) \end{cases}$$

The watermark from a supertree is extracted by

$$S_n = \sum_{m=1}^{5} D[\zeta'_{n,m}]$$

where m is the number of the subblock in the supertree. The extracted value is obtained through

$$W'_n = \begin{cases} -1 & \text{if } S_n \geq \gamma \\ 1 & \text{if } S_n < \gamma \end{cases}$$

where γ is a threshold and $\gamma = 0$. Finally to decide whether the watermark is detected or not the normalized correlation is calculated

$$\Phi(W, W') = \frac{\sum_{n=1}^{L} W_n W'_n}{\sqrt{\sum_{n=1}^{L} W_n^2 \sum_{n=1}^{L} W'^2_n}}$$

and compared to a threshold Φ_T. If $\Phi > \Phi_T$ the watermark is said to be present, otherwise not.

2.6.3 multiple watermarking

Like the last tree-based algorithm, there are only few coefficients in the smallest detail subbands. This makes the approach of the random wavelet coefficient selection less adequate because of the drawback that sharing the coefficient increases this shortage. Also wavelet packet decomposition can not be applied because when strictly following the proposed method only the coefficients of the smallest detail subbands and their direct children are marked. Wavelet parametrization is a possibility to enhance the algorithm and can be directly applied. A possible modification is to not embed the watermark in the smallest subbands and the direct descendants but in bigger ones, which will increase the capacity of the algorithm. Additionally several decomposition levels can be used for embedding, this will increase the capacity even more. This can ease the situation with random wavelet coefficient selection and enable the usage of wavelet packets. For the usage of wavelet packets a parent-child relationship has to be defined. Like for the last algorithm [30] can be used.

Chapter 3

Implementation and general performance

For the experiments with the Broken Arrows algorithm we used the implementation of Teddy Furon and Patrick Bas which was used in the BOWS2-Contest. All other algorithms and the experiment testbed was developed in Python using the "numpy" package for array manipulations and the "pywt" package for wavelet transforms and reconstructions. The attacks were carried out either in the (Python) source code or with the help of ImageMagick.

Because not all details are covered in the according papers of the algorithms this chapter covers the assumptions that had to be made concerning the implementation and the parameters. Furthermore the test setups are described including procedure and chosen parameters.

3.1 Tests

The general performance of the algorithms is evaluated by the following criteria:

- **Capacity**: Amount of information that can be embedded.
- **Visibility**: Amount of visual distortion the embedding process causes to the cover image.
- **Robustness**: Resistance of the watermark against modification of the host image.

These criteria will in most cases conflict with each other. So almost always making the watermark less visible will lower the robustness at the same time or embedding more information makes the watermark more visible.

For capacity analysis the maximum capacity and the interrelationship to visibility are considered.

All algorithms in the test have one or more parameters which control the embedding strength. The visibility analysis tests how good the distortion of the host image can be controlled with these parameters. Therefore, the watermark is embedded with different embedding strengths and the PSNR is evaluated.

Robustness studies are carried out by embedding the watermark with standardized strength, then different attacks on the image are executed and finally the watermark is extracted and the detection value is calculated. The target of an attacker is to remove the watermark i.e. make it undetectable while preserving as much image quality as possible. In the experiments the embedding parameters are chosen to obtain a distortion of about 42dB PSNR. The following attacks are executed:

- **Compression**:
  Compression is not necessarily an attack on the watermark but watermarked images can undergo compression in everyday use. So it is crucial for a watermark not to be erased when compression is applied. For compression studies the two most common lossy image compression techniques are used.
  - **jpeg compression**: the watermarked image is jpeg compressed with decreasing quality and the detection values are evaluated.
  - **jpeg2000 compression**: the watermarked image is jpeg2000 compressed with decreasing rate and the detection values are evaluated.
- **Image processing**: Image processing is another group of operations that can be applied to an image without the intention to remove the watermark. For the tests three image processing operations are chosen that are very common.
  - **Median filtering**: blurs the image. This can be especially damaging to watermarks since many algorithms use HVS considerations to select the areas where to embed the watermark. Because the human eye is less sensitive to distortions in textured regions of the image the watermark is often embedded into these regions. For the experiments the image is median filtered with increasing radius of the filter mask.
  - **Noise adding**: this is done by adding gaussian noise with μ=0 and increasing sigma to every pixel of the image.

  - **Gamma correction**: This operation was chosen to be part of the experiments because gamma correction often improves the image quality in a certain range. So watermark algorithms should be able to cope with a large range of gamma values.

- **Geometric transforms**:
  Most watermarking algorithms are very sensitive when it comes to geometrical transforms. Although the PSNR values are rapidly decreasing when applying geometrical transforms the visual distortion is rather low. This kind of attack brings the watermark quickly out of synchronization and makes it undetectable.

  - **translation**: the translation attack is implemented by a rolling of the image to the right. This means the rightmost x pixel columns are cut and inserted at the very left of the image to keep the dimensions of the image.
  - **rotation**: for this attack the image is rotated by an given angle and afterward scaled to the dimensions of the original image.

Lastly the multiple watermarking capabilities of the algorithms are tested. Therefor a number of watermarks are embedded and afterwards the watermarks are extracted again and the detection values are evaluated. For the sake of comparability to the previous experiments the same parameters as in the other experiments are used in the watermarking process. In most cases this will lead to a bad image quality when many watermarks are embedded. Experiments with adjusted parameters for multiple watermarking are carried out in the next chapter (chap. 4).

3.2 Barni

Some changes had to be made to the algorithm. For the calcultion of Λ the approximation image was scaled to fit in the range of 0-255 and the division was changed from 256 to 255. Most probably this was intended by the authors. Without the scaling, the divisor 256 seems a little motiveless to me. Another small change is the insertion of floor operators for the calculation of the indices in the Ξ term:

$$\Xi(l,i,j) = \sum_{k=0}^{3-l} \frac{1}{16^k} \sum_{\theta=0}^{2} \sum_{x=0}^{1} \sum_{y=0}^{1} \left[I_{k+l}^{\theta} \left(\lfloor y + \frac{i}{2^k} \rfloor, \lfloor x + \frac{j}{2^k} \rfloor \right) \right]^2 \cdot$$

$$\cdot Var \left\{ I_3^3 \left(\lfloor 1 + y + \frac{i}{2^{3-l}} \rfloor, \lfloor 1 + x + \frac{j}{2^{3-l}} \rfloor \right) \right\} \begin{array}{l} x = 0,1 \\ y = 0,1 \end{array}$$

Self-evident there are only integer indices. Furthermore the authors use an α of 1.5 in their experiments, but this is far too high in our implementation. So for most experiments an *alpha* of 0.08 was used. After these changes the difference image of the watermarked and the original image still looks a little bit different (Fig. 3.2) to the one in the paper of Barni et al. [2]. But one can see, that the basic principle of the visual mask generation is the same, so the implementation is sufficiently close the the proposed algorithm.

3.2.1 Capacity

The algorithm uses the three biggest detail subbands for embedding, hence for a 512x512 image there are 196608 marked coefficients.

3.2.2 Visibility

Visibility can be controlled by a weighting factor (α). In Figure 3.1 we can see that the detection value is rising much faster than the detection threshold. For the detectability a high α would be the best. But as said before the suggested value of 1.5 for α is too high and would result in a PSNR of approximately 16dB. A value of 0.08 gives a PSNR of around 42dB.

Figure 3.2 shows the watermarked image and the difference to the original where the effect of the visual mask can be seen. The watermark is embedded around edges and in textured areas of the image.

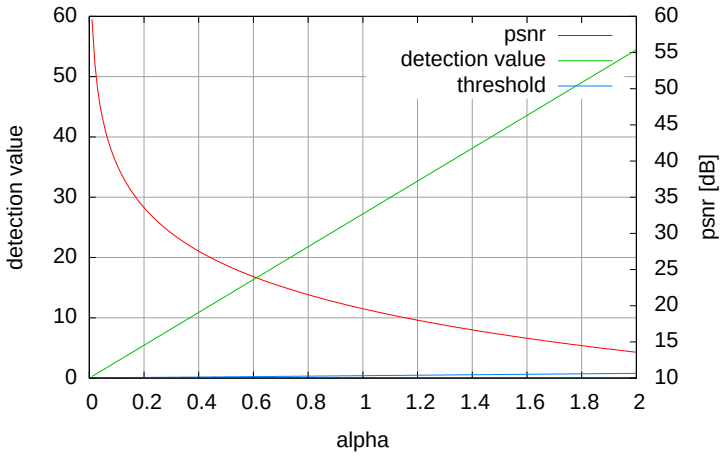
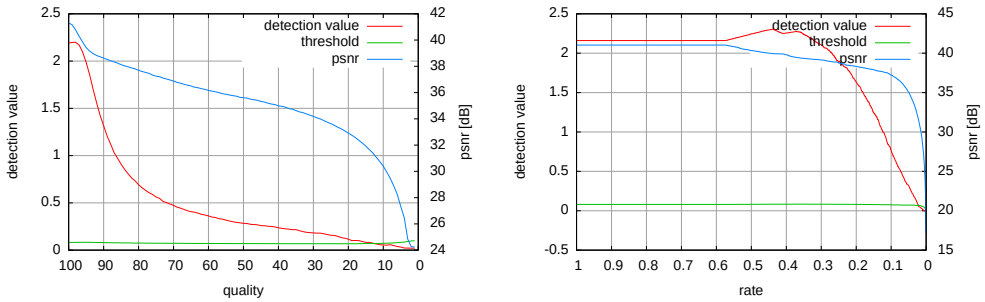


Figure 3.1: Barni: visibility

Figure 3.2: watermarked image and difference image after embedding, difference image is amplified 14.17 times; watermarked image: Lena, alpha: 0.08



(a) jpeg compression

(b) jpeg2000 compression

Figure 3.3: Compression attacks on the algorithm of Barni et al.; watermarked image: Lena, alpha=0.08

3.2.3 Robustness

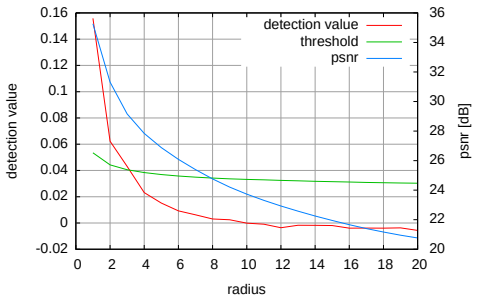
Robustness experiment results can be seen in Figure 3.3. Under jpeg compression the watermark can be detected ultimately with quality=13%, this is a PSNR of 31.4dB and jpeg2000 compression can be resisted to a rate of 0.023 (PSNR=31.8dB).

Figure 3.4 shows the results of the image processing attacks. Under median filtering the watermark can be detected until the filtering radius is above 3 (PSNR=29.2dB). The resistance against noise adding is very high. The watermark is still detectable with a sigma of 143 (PSNR=8.6dB). Curiously the threshold is rising with increasing sigma, but this can be explained by the fact, that the adding of noise produces many rather high coefficients in the detail subbands which are normally relatively sparse and this causes the threshold to go up. The effect of noise adding on the decomposition tree can be seen in Figure 3.5. Gamma correction removes the watermark if gamma is below 0.022 (PSNR=5.7dB) or above 622 (PSNR=5.3dB).

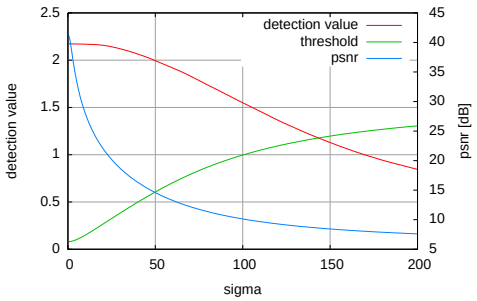
The algorithm can not withstand any of the tested geometric transforms. Already a roll of 1px or rotation of 1° removes watermark.

3.2.4 Multiple watermarking

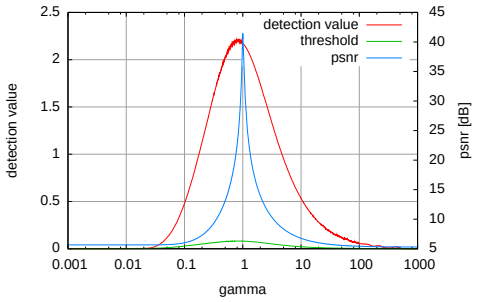
For the first test on multiple watermarking, the algorithm was carried out 30 times with different watermarks. As we can see in Figure 3.6 all 30 watermarks can be detected whereas the detection value of the first watermarks is a little lower because of the re-watermarking, but even the detection value of the first watermark is clearly above the threshold.



(a) median filtering

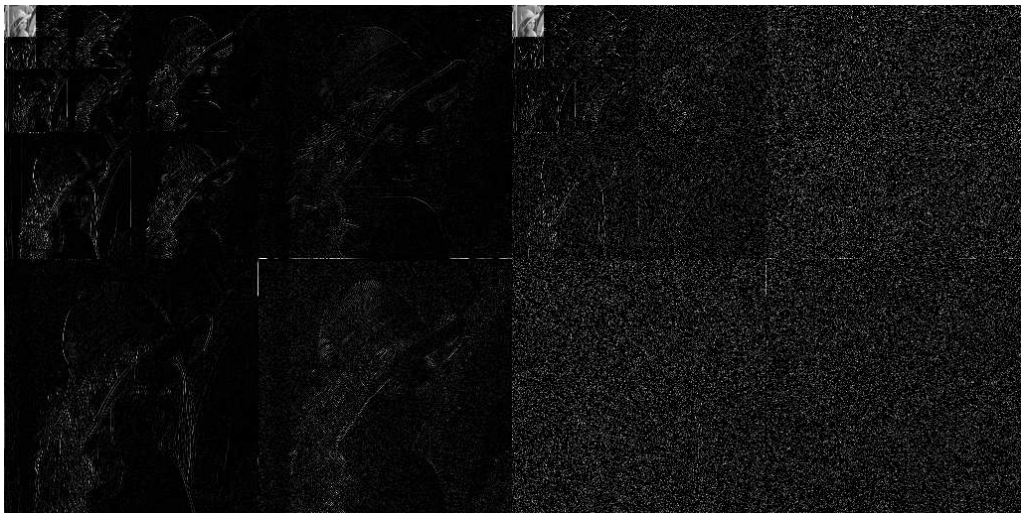


(b) noise adding



(c) gamma correction

Figure 3.4: Image processing attacks on the algorithm of Barni et al.; watermarked image: Lena, alpha=0.08



(a) wavelet tree

(b) wavelet tree after noise adding with sigma=30

Figure 3.5: wavelet decomposition trees of the plain image (3.5a) and the image after noise adding (3.5b)

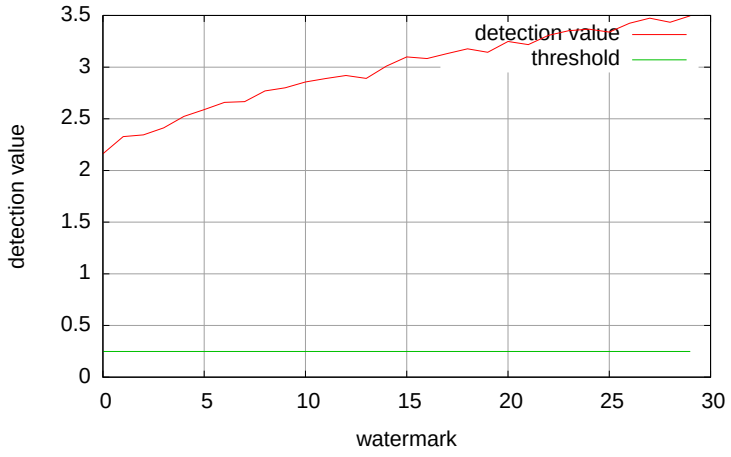


Figure 3.6: Barni: multiple watermarking

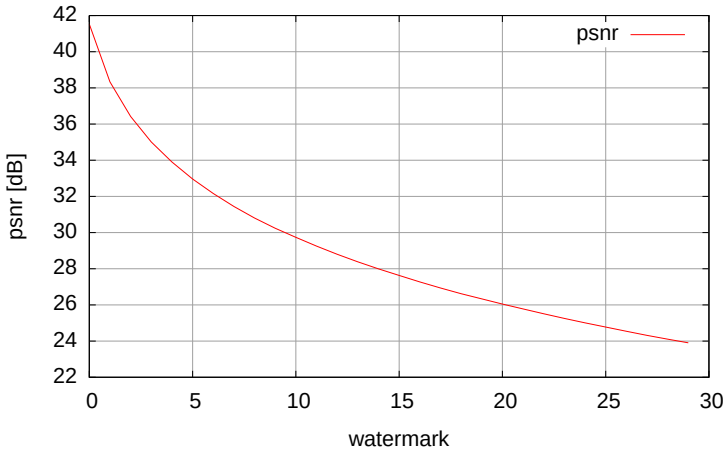


Figure 3.7: Barni: PSNR with increasing number of embedded watermarks

When using the same parameters as in the previous experiments ($\alpha = 0.08$) the PSNR is rapidly dropping as shown in Figure 3.7.

Like with the last algorithm we test the distinguishability between embedded and not embedded watermarks. Up to 50 watermarks are embedded and in every step all embedded watermarks and 50 not embedded watermarks are extracted. The result in Figure 3.8 show that for any number of embedded watermarks the worst correct detection value is clearly above the detection threshold and even more is not decreasing when more watermarks are embedded. The highest detection value of the not embedded watermarks is increasing with every embedded watermark, but the also increasing threshold compensates that.

An explanation attempt for the not decreasing worst detection value is that the visual mask "avoids" the previous embedded watermarks, but experiments, where the same visual mask is used for every embedding step, showed that the worst detection value also stays constant whereas the best detection value does not rise any more with the number of embedded watermarks. The explanation must be, that embedding the same watermark bit (either -1 or 1) twice into one coefficient exactly outweighs the embedding of a different watermark bit in case of a single visual mask and when using separate visual masks for each embedding turn (this is the normal procedure because an embedder does not necessarily know its previous embedders and therefore does not have the visual mask of previous embedding steps) previous watermarks alter the image in a way that the visual mask of subsequent turns makes the watermark be embedded stronger while the positive and negative impacts on

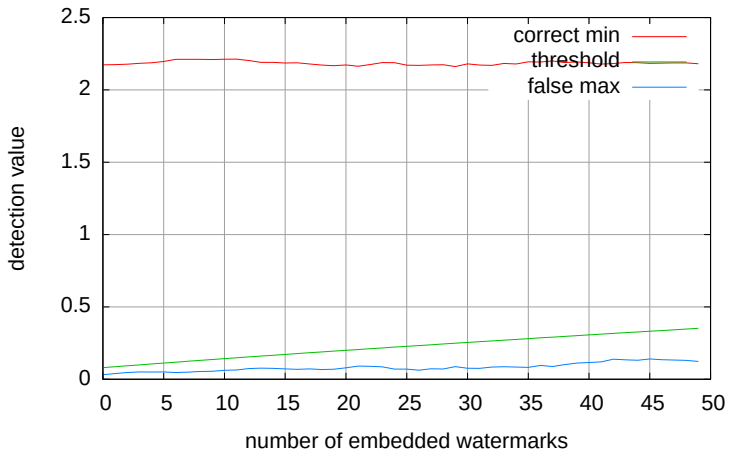


Figure 3.8: Barni: detection values of embedded watermarks compared to false watermarks

the other watermarks still outweigh each other.

3.3 Broken Arrows

I modified the algorithm to process some command line arguments for the embedding and detection parameters to enable batch processing. The experiment parameters were controlled by the Python testbed and passed to Broken Arrows via operating-system calls. Results are a returned to the testbed via files.

Since the detection threshold is a random value in this algorithm there is no definite value for it. In the experiments three different threshold are recorded. This is tmin, tmax and threshold. If the detection value is above tmin the watermark is detected anyway. If the detection value is below tmax the watermark is said to be not detected. Threshold is always between tmin and tmax and is randomly chosen. The detection value has to be above this threshold and in addition not in a snake trap (see 2.2.1). The snake traps are not illustrated in the charts.

3.3.1 Capacity

Since Broken Arrows is a zero-bit algorithm it does not carry any information except the fact that the watermark is present or not. But as we will see later on, some information can be "encoded" in the keys used for the secret projection.

3.3.2 Visibility

The algorithm offers the possibility to give a target PSNR value. Figure 3.9 shows that the achieved PSNR differs less than 1dB from the target PSNR value in the range from 17 to 65dB. We can also see that when the watermark is embedded with a PSNR of 40dB or less the detection reaches the maximum detection value. On the other end if the watermark is embedded with a PSNR of approximately 59dB or above the watermark is too weak to be detected. Figure 3.10 shows a watermarked image and the difference image after embedding with a target PSNR of 42dB.

3.3.3 Robustness

The algorithm shows excellent performance with respect to the resistance to compression attacks. Figure 3.11a shows that jpeg compression can not destroy the watermark even with a jpeg-quality of only 1%. Although the detection value is decreasing with lowering quality it always stays above the threshold. Also the resistance against jpeg2000 compression is very good. In

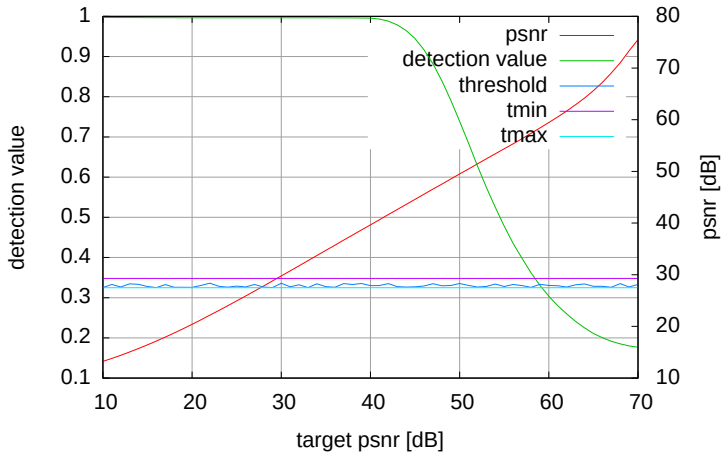


Figure 3.9: Broken Arrows: visibility

Figure 3.10: watermarked image and difference image after embedding, difference image is amplified 9.44 times; watermarked image: Lena, target PSNR: 42dB

figure 3.11b we can see that the watermark can be detected until a rate of 0.004 where the PSNR is 24.7dB.

In figures 3.11c, 3.11d and 3.11e the results for the next group of attacks can be seen. That is median filtering (3.11c), where the watermark can be detected up to a radius of 4, that results in a PSNR of 27.8dB. Noise adding (3.11d) can be resisted till sigma is above 61, PSNR=13.1dB. In case of gamma correction (3.11e) the watermark is detectable when gamma is between 0.032 (PSNR=5.7dB) and 589 (PSNR=5.3dB).

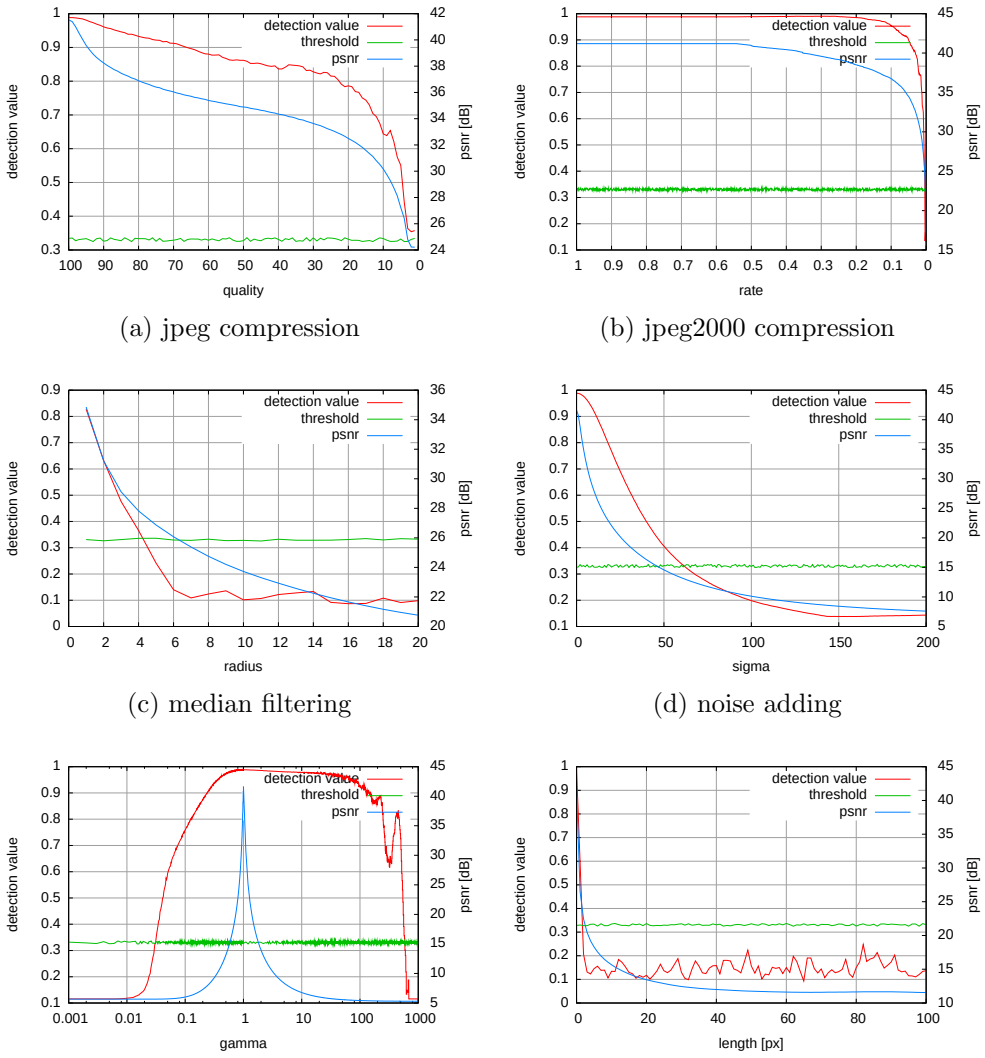
The authors state in their paper [17] that they did not care about resistance agains geometric attacks because they always result in a low PSNR and therefor are not applicable in the BOWS2 contest. Figure 3.11f show the results of the translation attack and we can see that the watermark can only be detected when translation is not more than 1px. Rotation immediately removes the watermark already with an angle of 1°.

3.3.4 Multiple watermarking

To assess the suitability for multiple watermark embedding the algorithm was applied multiple times with different Key K. This key is the 128 bit long seed for the PRNG that produces the carriers for the projection in the secret subspace. Figure 3.12 shows the results where 30 watermarks were embedded and afterwards the detection value for each of the 30 watermarks was calculated. As we can see the detection values slightly decrease with every rewatermarking but all 30 watermarks are clearly above the threshold.

If watermark embedding is done with 42dB as the target PSNR for each watermark. The PSNR value to the original, unwatermarked image is rapidly decreasing as can be seen in figure 3.13.

Another eventuality could be, that the multiple embedding makes the algorithm more prone to false positive detection. For the next test an increasing number of watermarks has been embedded and afterwards extracted again. Additionally 50 false (not embedded) watermarks are extracted and the worst detection value of one of the correct watermarks is compared to the best detection value of the false watermarks. The results can be seen in figure 3.14. The worst detection value of the correct watermarks is continuously decreasing with rising number of embedded watermarks and at the same time the best detection value of the false watermarks is slightly increasing. The test was carried out with up to 50 watermarks and even with 50 embedded watermarks there is no problem in distinguishing between correct and false watermarks. The worst detection value of a correct watermark is well above the threshold while the best false watermark has only a negligibly



(a) jpeg compression

(b) jpeg2000 compression

(c) median filtering

(d) noise adding

(e) gamma correction

(f) translation

Figure 3.11: Attacks on the Broken Arrows algorithm; watermarked image: Lena, target PSNR: 42dB

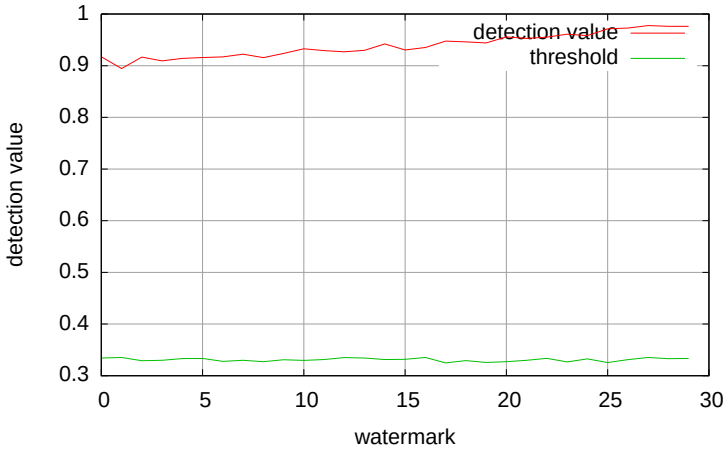


Figure 3.12: Broken Arrows: multiple watermarking

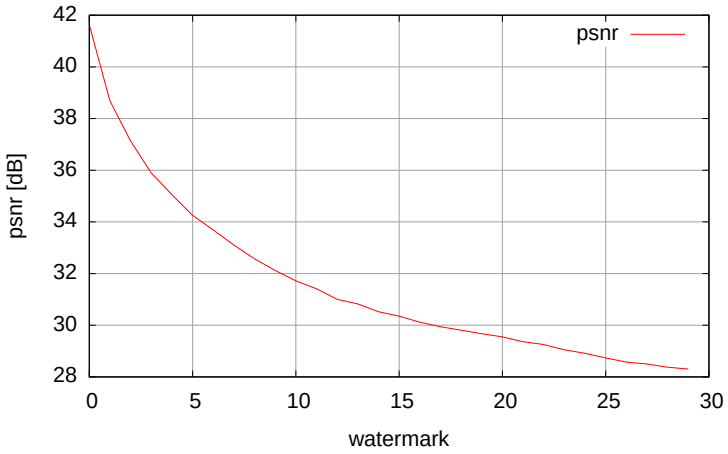


Figure 3.13: Broken Arrows: PSNR with increasing number of embedded watermarks

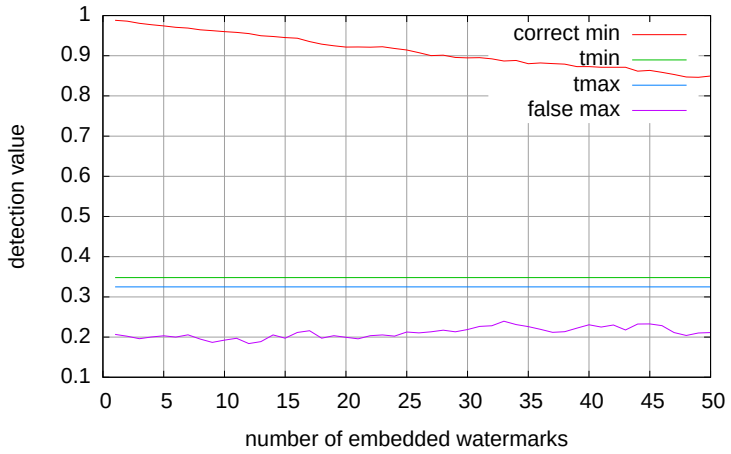


Figure 3.14: Broken Arrows: detection values of embedded watermarks compared to false watermarks

higher detection value than with only one embedded watermark.

3.4 Cao

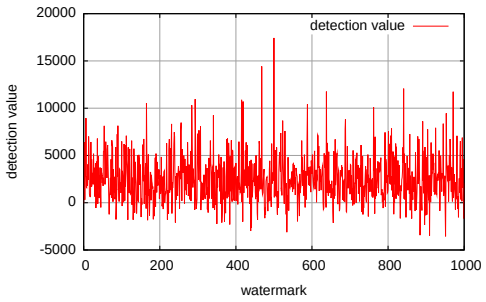
For the algorithm of Cao et al. we use our own implementation of the RDWT, since the one in the pywt-package is not complete.

In our implementation we could not reproduce the proposed results. The detector response has a completely different characteristic. The problem is the correlation mask and especially the normalization of the RDWT coefficients. The parameter α is very hard to guess and is highly dependent on the image size. For our tests with the 512x512 Lena image and an embedding strength of 42dB α is 2400000000. Further more, the watermark is embedded in only very few coefficients. We tested some different approaches to calculate a correlation mask. The detector responses are printed in figure 3.15. Watermark number 500 is the correct one, all others are randomly chosen false watermarks. Cao et al. use the peak-response gain (prg) as quality measure. They define this value as:

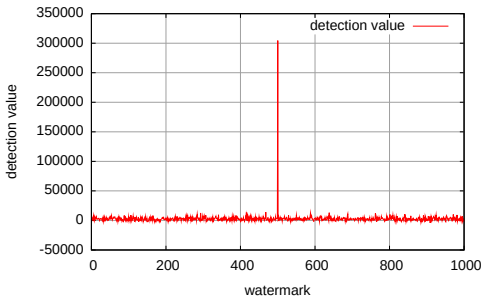
$$G = \frac{N \cdot max(\rho(n))^2}{\sum_{n=1}^{N} \rho(n)^2}$$

where $\rho(n)$ is the detector response for trial n and N is the number of trials. Cao et al. do not give a threshold but use this value as a measure how difficult it is to find a threshold that safely separates the correct from the false watermarks. The measure can also be seen in the figure. Cao has a prg of 996.4 in his results. Our result with the original mask, this is the mask as proposed by Cao, shows an prg of only 24. Even worse is the situation when we look at the second highest, or in other words the highest false, detection value. There is almost no difference between the first and the second. When we modify the prg to be the relation between the first and the second highest detection value, we will call this the peak-second gain (psg), we have only 1.2 (fig. 3.15a). A psg of 1 means, the highest false watermark response is as high as the detector response for the correct watermark, just for clarification. We can say that this mask is totally unsuitable. Our other approaches are:

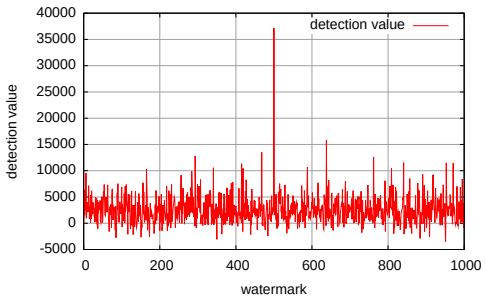
- no visual mask (fig. 3.15b): all elements in the visual mask are 1. So the watermark is constantly embedded into the whole image.
- no normalization (fig. 3.15c): since we suspect the normalization to be the problem we test the visual mask as proposed but without the normalization.
- alternative normalization (fig. 3.15d): we normalize the range from the smallest coefficient to the biggest coefficient to be [0,1].



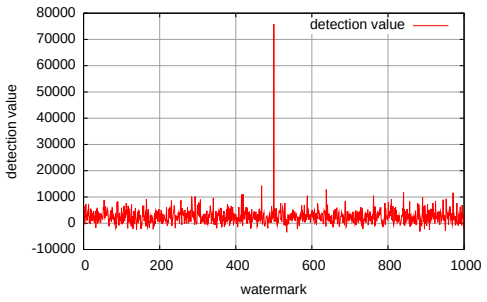
(a) original mask, prg=24



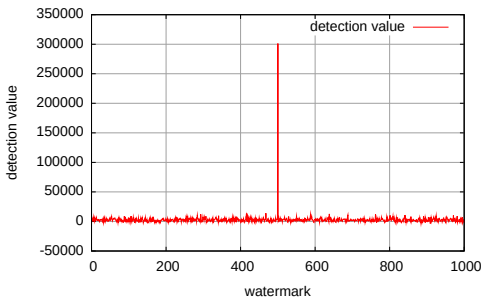
(b) no mask, prg=881



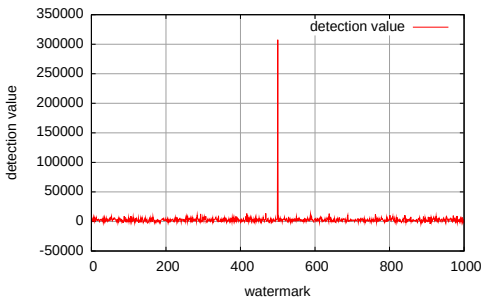
(c) no normalization, prg=99



(d) alternative normalization, prg=319



(e) 10% pla mask, prg=878



(f) 30% pla mask, prg=882

Figure 3.15: Detector response comparison of the Cao algorithm with different masks

- 10% Pla mask (fig. 3.15e): Because the constant visual mask has the best results so far, we will now test to incorporate the idea of Pla et al. (sec. 2.5). We basically embed the watermark with a constant embedding strength into the whole image, but amplify the embedding strength with a certain factor in significant regions. To decide where the significant regions are we use the approach Cao uses in his algorithm. This is, we multiply all coefficients according to a specific orientation and the same spatial position. We amplify the embedding strength by the factor 2 in the 10% with the highest product.

- 30% Pla mask (fig. 3.15f): This is the same mask as the previous, except that we amplify the highest 30%.

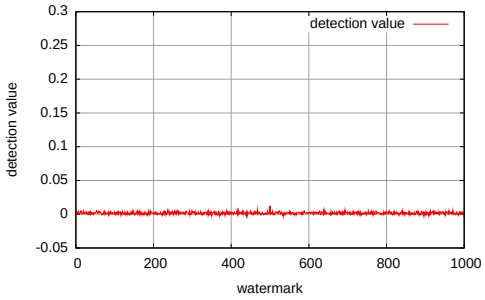
We can see, that in terms of the prg the version without mask or the two versions with the Pla-mask reach the best results by far. The mask with alternative normalization is clearly behind the first three algorithms but still has good values. The mask without normalization has a rather low prg with 99, but still is acceptable. The original version of the mask is definitely the worst and not usable.

One can observe that the detection values have huge differences in their value. We tested if the normalized correlation can also be used for watermark detection and not only that the detector responses are better comparable, also the prg is higher for all versions except for the version without coefficient normalization. Results are printed in figure 3.16. All plots use the same range on the y axis to better reflect the prg.

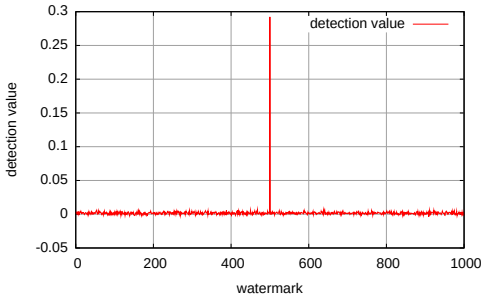
In our experiments we will focus on the algorithm with the 30% Pla-mask because it has the best results in terms of prg and also keeps the idea of the visual mask of the original algorithm in some way. We also will use the normalized correlation instead of the not normalized one because of the results above.

3.4.1 Capacity

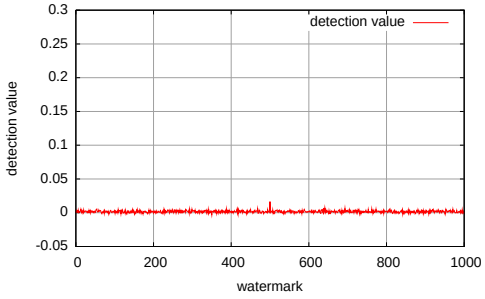
The algorithm embeds the watermark in the three finest detail subbands. Since the RDWT is used, every subband is in the size of the original image. So the number of coefficients used for watermarking is three times the number of pixels in the image. For the 512x512 image in our experiments this are 786432 coefficients.



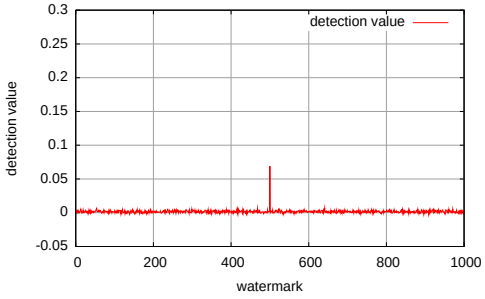
(a) original mask, prg=31



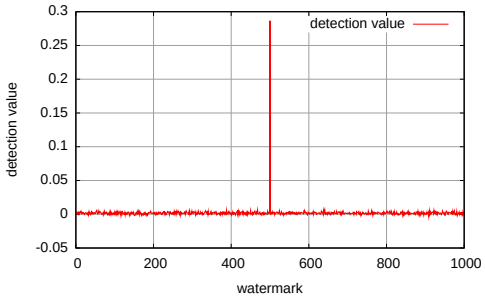
(b) no mask, prg=947



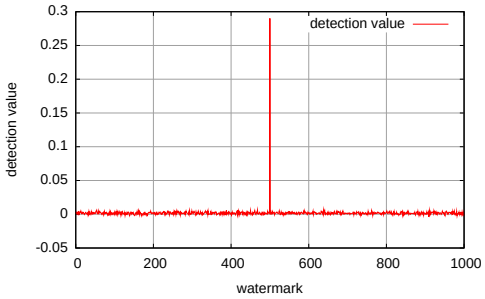
(c) no normalization, prg=54



(d) alternative normalization, prg=500



(e) 10% pla mask, prg=945



(f) 30% pla mask, prg=947

Figure 3.16: Detector response comparison of the Cao algorithm with different masks and detection value normalization

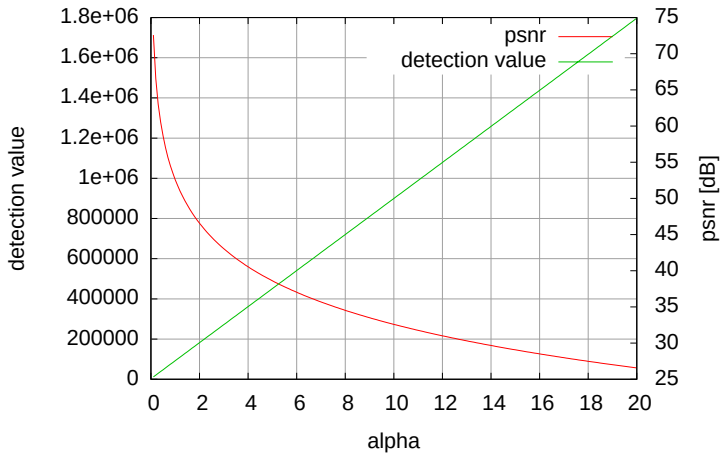


Figure 3.17: Cao: visibility

3.4.2 Visibility

The embedding strength can be controlled with a strength factor α. Figure 3.17 shows the relation between α and the embedding PSNR.

Figure 3.18 shows a watermarked image and the according difference image. The 30% Pla-mask was used for embedding and the embedding strength was chosen to get a PSNR of 42dB. The image shows the intended distribution of the watermark. The watermark is constantly embedded into the whole image and stronger around the edges.

3.4.3 Robustness

In the experiments printed in figure 3.16 no false watermark was above 0.008. The authors give no threshold, so we will use 0.008 as the detection threshold for the robustness experiments.

The results on the robustness experiments are shown in figure 3.19. With JPEG resistance to a quality of 15% (PSNR=31.9dB) and JPEG200 resistance to a rate of 0.025 (PSNR=32.2dB) the robustness against compression is in the mid-field of the tested algorithms. Under median filtering the watermark can be detected up to a radius of 4 (PSNR=27.9dB). The additive white gaussian noise robustness is the best of all the algorithms in the test, the watermark is lastly detectable with a sigma of 187 (PSNR=7.8dB). When gamma correction is applied the watermark can be detected when gamma is between 0.021 (PSNR=5.7dB) and 198 (PSNR=5.3dB). Like most other algorithms this one can not resist geometric attacks, so only a shift of 1px

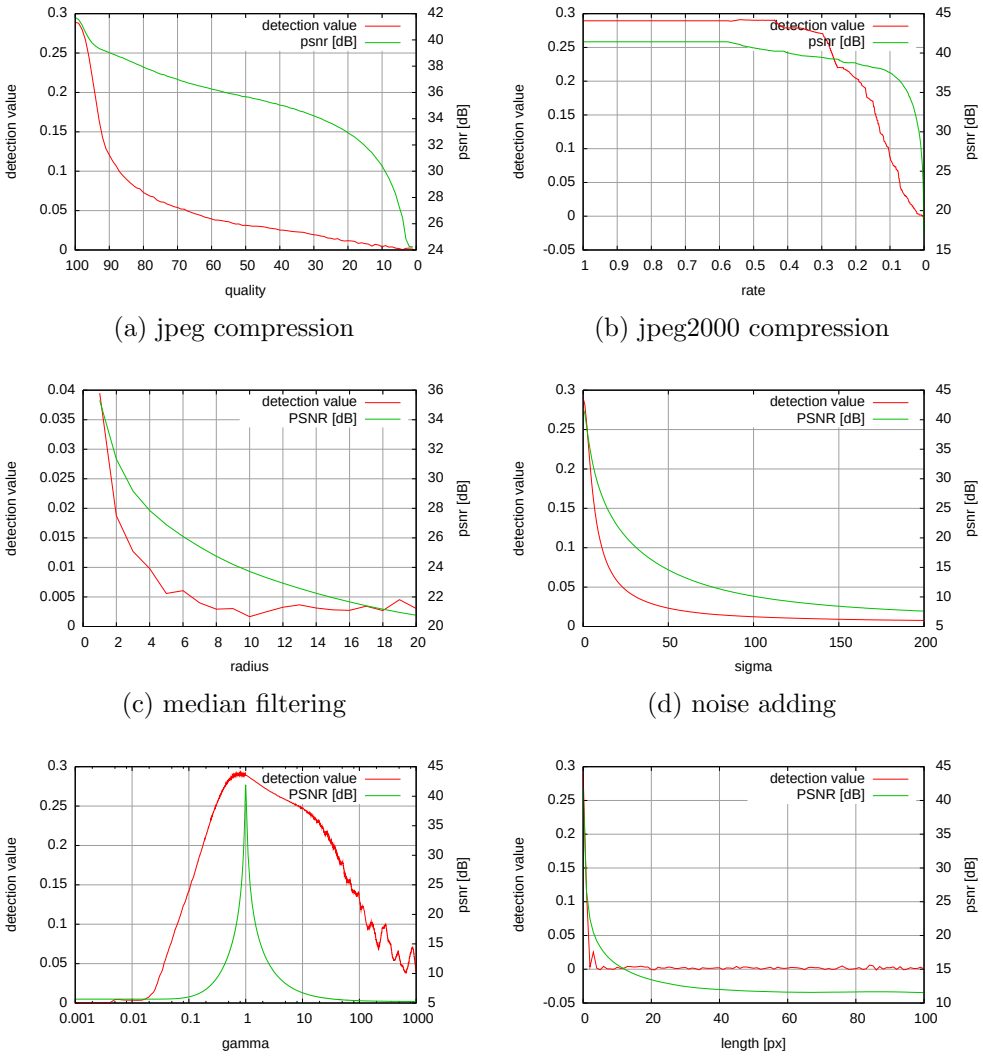
Figure 3.18: watermarked image and difference image after embedding, difference image is amplified 17 times; watermarked image: Lena, alpha: 3.4; 30% Pla-mask

Mask	jpeg	j2000	median	awgn	gamma	roll	rotation
cao mask	77	0.047	0	7	0.033 - 22	0	0
alt norm	27	0.022	3	33	0.028 - >100	1	0
no mask	18	0.027	4	>200	0.02 - >100	1	0
pla 10%	17	0.025	4	194	0.02 - >100	1	0
pla 30%	15	0.025	4	187	0.021 - >100	1	0

Table 3.1: robustness comparison of the different masks

can be coped with and rotation immediately removes the watermark.

We also tested the algorithm with the other masks to see if the different masks influence the robustness of the algorithm. A summary of the results is printed in table 3.1. The table shows where the watermark could lastly be detected. Our first thought was, that using no mask will lead to poor robustness, since the watermark is not embedded into the significant parts of the image. But as the table shows using no mask is only slightly inferior to the results of the pla mask in compression and even outperforms all other masks in median filtering- and noise adding-resistance. Not only in prg terms but also in terms of robustness the mask proposed by Cao et al. is the worst. The results of the mask with alternative normalization are in between the Cao mask and the other three. The fact that the algorithm used without a



(a) jpeg compression

(b) jpeg2000 compression

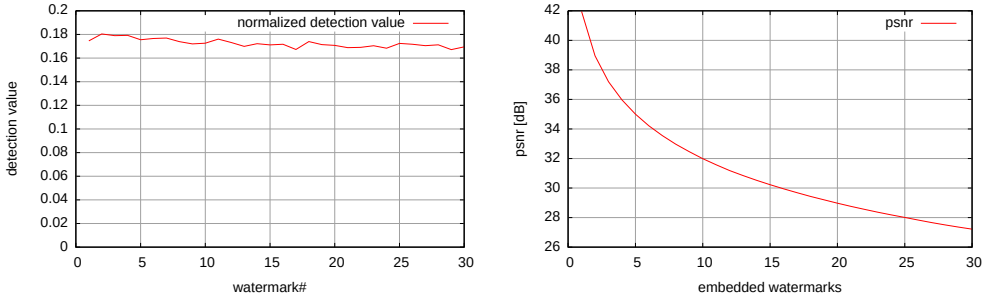
(c) median filtering

(d) noise adding

(e) gamma correction

(f) translation

Figure 3.19: Attacks on the Cao algorithm; watermarked image: Lena, alpha=3.4, mask=Pla 30%



(a) detection value (b) PSNR

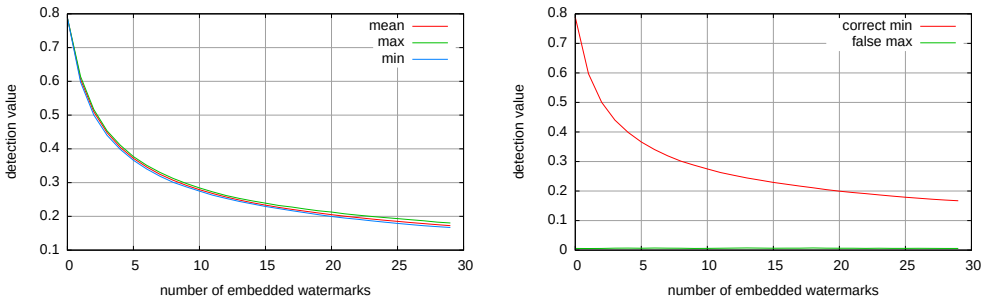
Figure 3.20: Cao: detection value and PSNR when embedding multiple watermarks

mask is only slightly worse in some robustness tests and even outperforms all others in the remaining robustness tests calls the usefulness of a mask for this algorithm in question. Another idea behind the mask proposed by Cao et al. could be, that it visually masks the watermark, meaning that with the same PSNR the watermark is less visible to the human eye when using such a mask, but this is not described by the authors. Since the evaluation of visual distortion, with respect to the HVS, is a topic of current research and no general accepted measure exists and first experiments by us showed no noticeable difference in the image quality, we will not go into detail on this subject. To keep the mask idea of the author of the algorithm we will carry on using a mask for further experiments.

3.4.4 Multiple embedding

In figure 3.20a we see that when embedding 30 watermarks, all of them can be extracted. We can further see a curious effect where later embedded watermarks have worse detection values than previous ones. This is not the case when we use the version without mask. This leads to the conclusion, that the decreasing detection values is a effect caused by the visual mask. Like with all other additive algorithms in the test the PSNR is quickly decreasing with every other embedded watermark (see fig. 3.20b).

In further experiments we could show that the decrease from the first to the last watermark is only a local trend. Figure 3.21a shows a development that has much more impact on the detection performance. This is the decrease of the average detection value when more watermarks are embedded. The worst detection value of the correct watermarks and the highest



(a) correct watermarks (b) correct versus false watermarks

Figure 3.21: Cao: development of the detection values with rising number of embedding runs

detection value of the false watermarks is plotted in figure 3.21b. The figure shows that the detection values of the false watermarks do not depend on the number of embedded watermarks and the correct and false watermarks can safely be distinguished.

3.5 Chen

Additionally to the paper [6] has to be said that the choice of the watermark is crucial for the algorithm since the recurrence time of the torus automorphism (see [35]) strongly depends on the size of the watermark and the parameter k. The recurrence time is the number n where the original matrix (the original watermark) is restored if the torus automorphism is applied n times. For some combinations there is no recurrence time at all.

The description of the distortion reduction is very loose in the paper. In the implementation a distortion limit can be passed to the distortion reduction algorithm. If the distortion introduced by watermark embedding exceeds the distortion limit, distortion reduction is applied. The algorithm sets the distortion exactly to the desired value if possible or if not to the least possible distortion. The latter is always less or equal as half of the distortion induced by watermark embedding except for the case that distortion reduction would change the number of significant bits. In this case the coefficient is set to the value giving the least distortion without changing the number of significant bits. This can result in the distortion being up to the distortion induced by watermark embedding.

The bit selection was changed to

$$M_b = \lfloor (I_n - 1) \times \alpha_b \rfloor$$

so that α_b can be in $[0, 1]$. The original selection embeds the watermark in a bit higher than the msb if α_b is close to 1. Maybe an even better solution would be

$$M_b = max(0, \lfloor (I_n - 2) \times \alpha_b \rfloor)$$

because when $\alpha_b \in [0, 1]$ then the embedding will at most embed the watermark in the bit one position lower than the msb and hence not change the number of significant bits. But since the second is a subset of possible embedding positions of the first version, the first one is used in the experiments.

3.5.1 Capacity

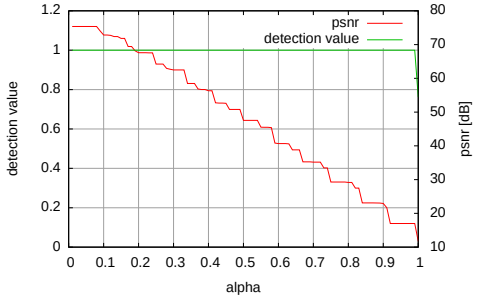
The authors propose to use an $N \times N$ image as the watermark. This restricts the watermark size to be 2^N with $N \in \mathbb{N}$. But it is not necessary to have a watermark of this shape. The watermark could also be a vector of any size. In any case the maximum capacity is the number of coefficients in the wavelet domain or in other words the number of pixel of the image.

Figure 3.22: watermarked image and difference image after embedding, difference image is amplified 28.33 times; watermarked image: Lena, alpha: 0.6, distortion limit: 100, watermark size:1024

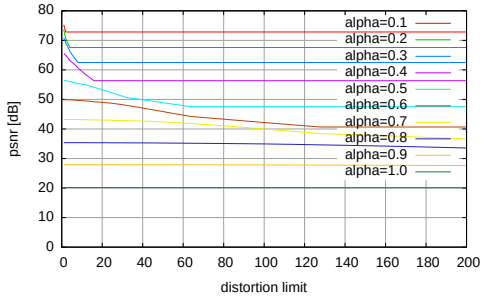
3.5.2 Visibility

The visibility depends on the strength factor α_b, on the distortion limit and on the size of the watermark. Figure 3.22 shows the image Lena with a 32×32 watermark image embedded and the difference image. The watermark is spread all over the image, because a 5 level wavelet transformation has been applied hence the watermark size exceeds the size of the approximation image and therefore the whole approximation image was used for embedding.

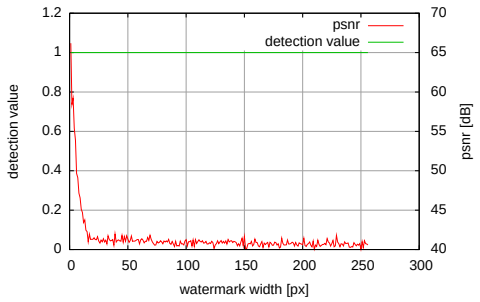
Figure 3.23 shows the effect of the different parameters on the visibility. In Figure 3.23a the impact of α can be seen. The stepwise decrease in the PSNR is because the distortion is only increased when embedding into the next higher bitplane and since the number of significant bits (number of bitplanes) is not very high, the PSNR jumps from one value to the next. When embedding with $\alpha = 1$ then the number of significant bits changes when a 0 is embedded. In this case the watermark can not be detected correctly and therefore the detection value is dropping at $\alpha = 1$. Figure 3.23b shows the effect of the distortion limit. We can see that, if the distortion limit is sufficiently small, the PSNR value increases. With rising alpha the maximum possible increase rises and also the range of distortion limit values that have an effect. This comes from the fact, that for small α-values the introduced distortion is low, keeping it inside of a small distortion range



(a) alpha (watermark length: 1024, no distortion reduction)



(b) distortion limit (watermark length: 1024)



(c) watermark length (alpha: 0.6, no distortion reduction)

Figure 3.23: effect of the different parameters on the visibility

(distortion limit) and thus distortion reduction is not applied when the distortion limit is high. The biggest effect can be obtained with an alpha of 0.6 closely followed by 0.5 and 0.4. Higher values of α can not be compensated by the distortion reduction because the distortion reduction algorithm can not always bring the coefficients in the desired distortion range and this is more often the case with a large α. In the case of $\alpha = 0.6$, this is the value the authors use in their experiment and also will be used in the robustness tests, the introduced distortion never exceeds 128 and the PSNR can be controlled in a range of 40.7 to 50dB. In Figure 3.23c the effect of the watermark length can be seen, where the PSNR drops from initially 66dB to 41dB and stays at this level. This is because when all coefficients in the approximation subband are marked the PSNR does not decrease any more. In our case this is at width 16 because we make a 5-level wavelet transform of a 512×512 image.

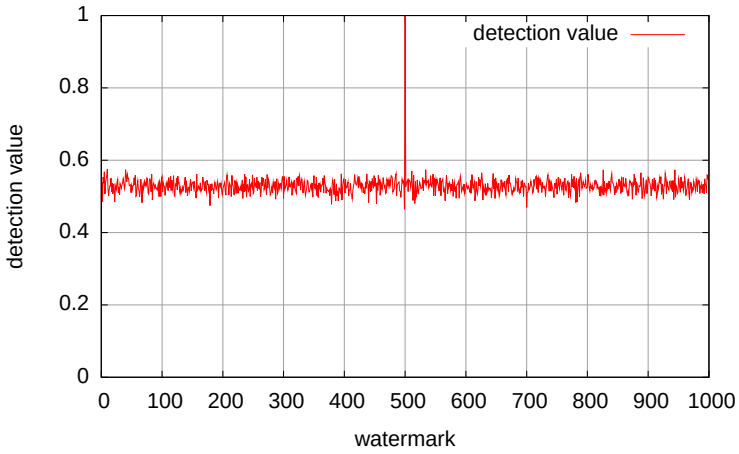


Figure 3.24: Chen: detection values of embedded watermarks compared to false watermarks

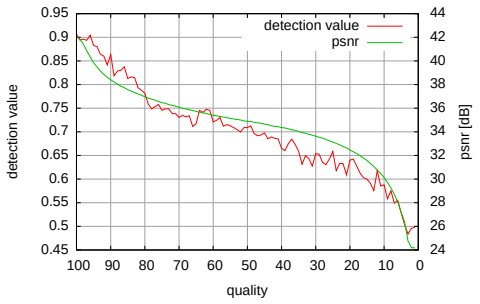
3.5.3 Robustness

The authors give no threshold for watermark detection. In Figure 3.24 we can see that the detection values of all false watermarks do not exceed 0.6. Therefore the watermark is said to be present if the detection value is above 0.6 in the following experiments.

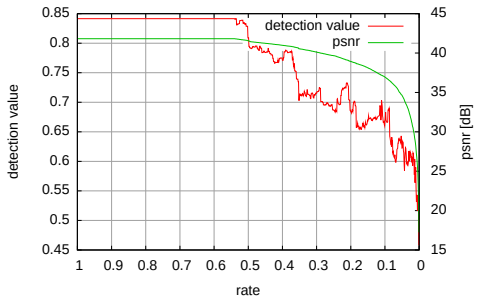
In case of jpeg compression the watermark can be detected as long as the quality does not go below 16% (PSNR=31.8dB). JPEG2000 compression removes the watermark when rate is smaller than 0.07 (PSNR=35.8dB). Median filtering can be resisted up to a radius of 4 (PSNR=27.7). Noise adding already removes the watermark when sigma is more than 19 (PSNR=22.5). The algorithm is very sensitive when gamma correction is applied. A gamma of less than 0.938 (PSNR=32.9dB) or more than 1.1 (PSNR=30.6dB) makes the watermark undetectable. Also the tested geometric transforms immediately remove the watermark.

3.5.4 Multiple embedding

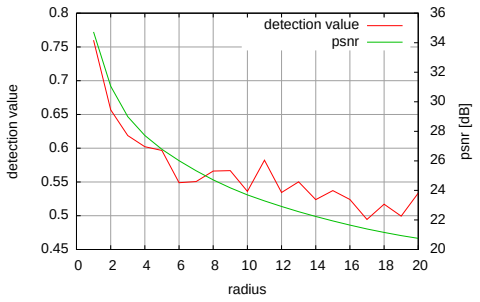
The basic algorithm is not suited for multiple embedding, as can be seen in Figure 3.26a. Only the last embedded watermark can be extracted, all others are completely deleted. This is the expected result because in every consequent embedding run exactly the same embedding positions are selected and the old watermark is overwritten. In Figure 3.26b we can see that the PSNR stays constant independent of the number of embedded watermarks.



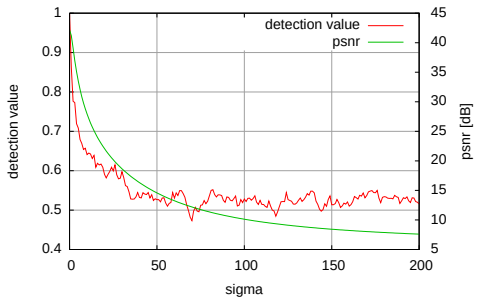
(a) jpeg compression



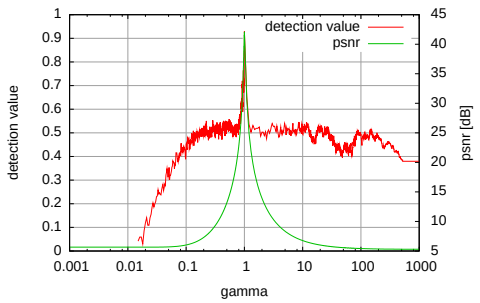
(b) jpeg2000 compression



(c) median filtering



(d) noise adding



(e) gamma correction

Figure 3.25: Attacks on the Chen algorithm; watermarked image: Lena, alpha=0.6, distortion limit: 100, watermark length:1024

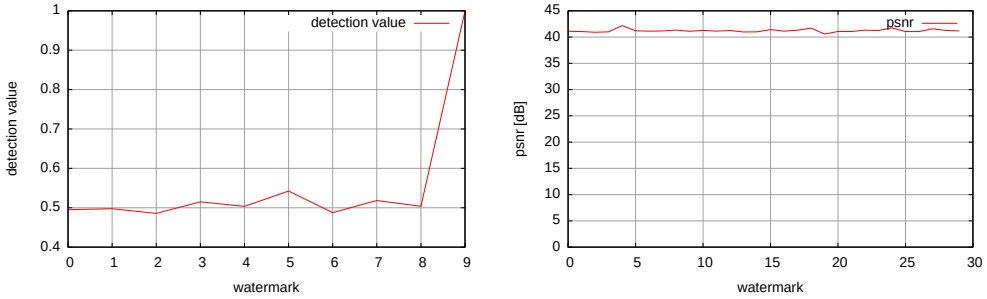


Figure 3.26: Chen: detection value and PSNR when embedding multiple watermarks

This result is also clear for the same reason. All embedding steps select the same embedding positions and so no new distortion is introduced with consequent embedding runs.

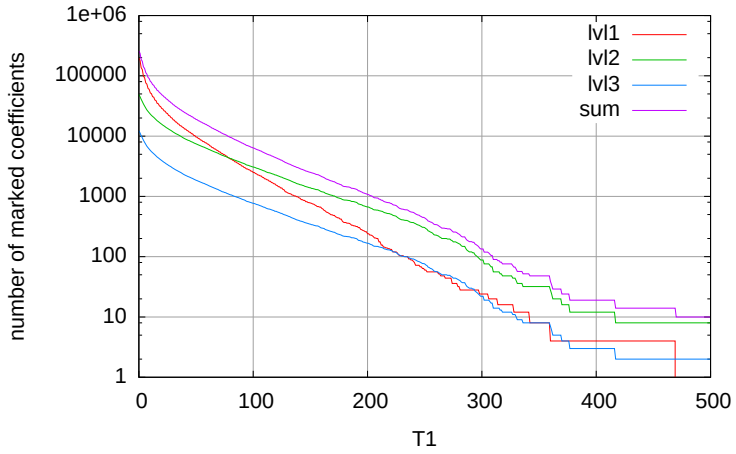


Figure 3.27: Pla: relation between threshold T1 and capacity

3.6 Pla

The template embedding scheme was not implemented because of the reasons mentioned in section 2.5. The authors give no closer explanation of the function for calculating the PRNG seed. The only requirement is that it should only depend on the embedding key and the subband and the position of the coefficient. In the implementation the seed is calculated by adding the md5 values of all terms and again computing the md5 value of the sum.

3.6.1 Capacity

The capacity of the algorithm depends on the threshold T1. Figure 3.27 shows the relation between the capacity and T1. Because the children of a coefficient of the third level (smallest subband) are used for embedding at all times, the number of marked coefficients in level 2 is always 4 times the number of marked coefficients in level 3. The number of marked coefficients in the first level is faster decreasing than the one in the third and consequently the second level because the wavelet transform compacts the energy in the lower levels.

3.6.2 Visibility

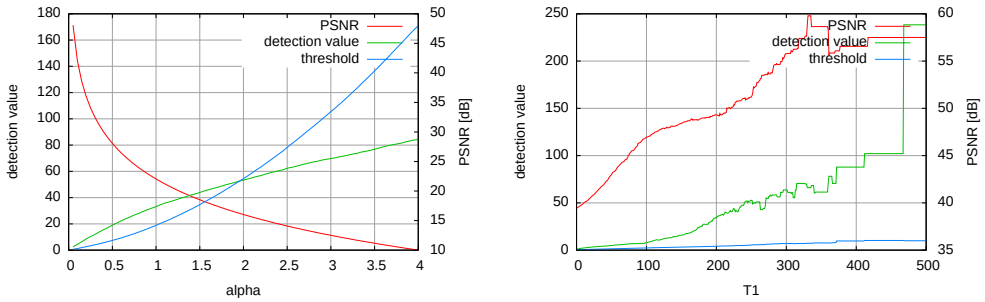
Figure 3.28 shows a watermarked image and the difference image. Embedding was done with α=0.1, γ=2 and T1=40.

Figure 3.28: Pla: watermarked image and difference image after embedding, difference image is amplified 7.29 times; watermarked image: Lena, alpha: 0.1, T1=40, gamma=2

Visibility experiments were carried out by varying the global scaling factor α and the threshold T1. The results can be seen in Figure 3.29. Figure 3.29a shows the expected result for the relation between alpha and the PSNR. An unexpected result is that higher alpha does not always result in better detectability. Because the threshold is faster increasing than the detection value, the watermark is only detectable with alpha lower than 2. The best detectability in the meaning of the biggest gap between the detection value and the threshold is at alpha=0.9, this would result in a embedding PSNR of 23dB. Figure 3.29b shows the effects of T1 on the detection value and the PSNR. Bigger T1 values give better PSNR because less coefficients are marked and also better detection values are reached because of the proportional embedding where bigger coefficients are marked stronger. For detection T2 is chosen as T1+20%.

3.6.3 Robustness

The results of the robustness tests can be seen in Figure 3.30. The algorithm has good resistance against jpeg compression, down to 3% quality (PSNR=24.8dB) can not erase the watermark. Also jpeg2000 compression can be resisted to a rate if 0.004 (PSNR=24.6dB). Under median filtering the watermark can be detected with a radius of 4. Results on noise adding are



(a) alpha (T1=40, gamma=2)

(b) T1 (alpha=0.1, gamma=2)

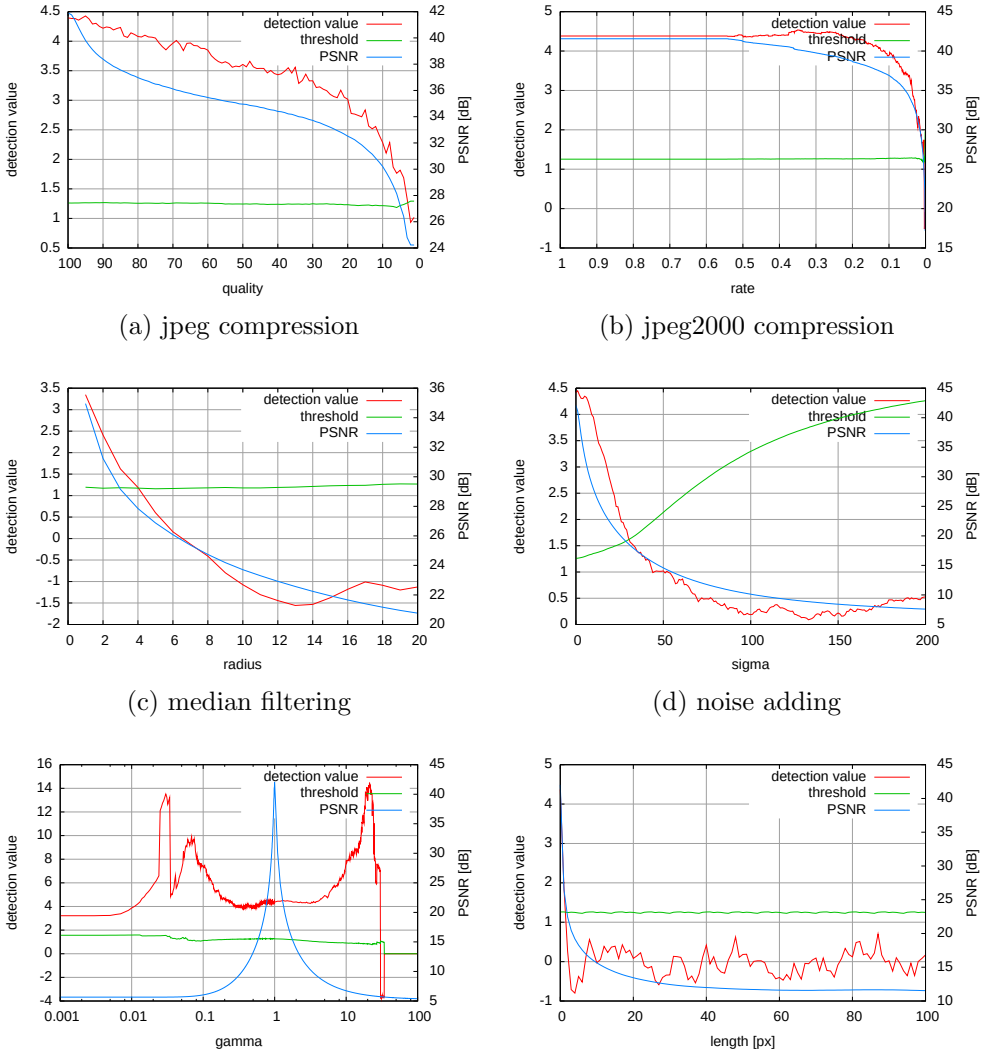
Figure 3.29: Pla: effects of the different parameters on the visibility

not so good. The watermark is removed when adding noise with sigma=30. When applying gamma correction the detection value does not drop below the threshold for gamma<1 and the watermark is last detectable with gamma=30.2 on the gamma>1 side. But gamma correction also increases the false positive value so that the first false positive detection occurs at gamma=0.089 and gamma=12.2. Like all other algorithms in the test, the algorithm of Pla et al. performs very bad when applying geometric attacks. The watermark is only detectable when a translation of maximal 1px is applied and rotation immediately removes the watermark.

3.6.4 Multiple Watermarking

In the first tests on multiple watermarking (see Figure 3.31) all 30 embedded watermarks could be extracted. Furthermore we can see that with the standard parameters of the algorithm the PSNR quickly decreases with rising number of embedded watermarks (see 3.31b). Figure 3.31a shows that the detector response shows an anomaly. Later embedded watermarks have a lower detection value than older ones. Additional experiments showed that this results from a combination of two facts. First the visual mask (the activity image) has a similar effect as the one of Barni et al. (see section 3.2.4). The second fact is that the embedding process can lower a coefficient below the threshold T_1. This decreases the number of coefficient used for embedding in later embedding steps. So ever other embedding run has less coefficients to mark.

Further experiments revealed other problems concerning the multiple detection capabilities. Although the minimum detection value decreases with increasing number of watermarks all 50 watermarks could be detected. But



(a) jpeg compression

(b) jpeg2000 compression

(c) median filtering

(d) noise adding

(e) gamma correction

(f) translation

Figure 3.30: Attacks on the Pla algorithm; watermarked image: Lena, alpha=0.1, gamma=2, T1=40, T2=50

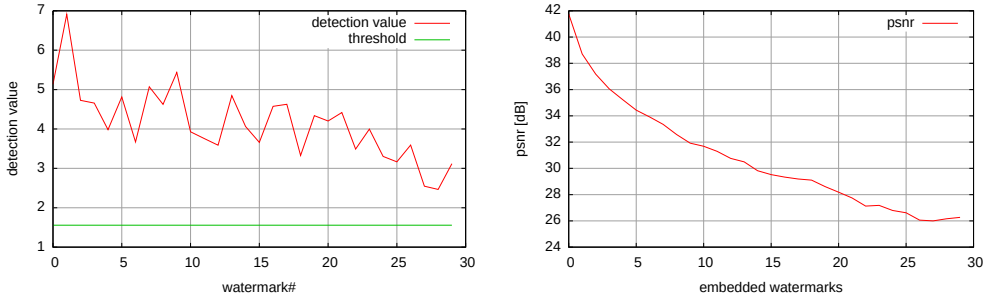


Figure 3.31: Pla: detection value and PSNR when embedding multiple watermarks

also the standard deviation of the false detection values rises while the mean stays at approximately 0 and thus the biggest false detection value also increases. So, before one of the watermarks can not be detected anymore a false positive detection occurs when embedding 16 watermarks (see Figure 3.32). Maybe the false positive and false negative border can be deferred if the calculation of the detection threshold would be modified. In Figure 3.32 the gap between the minimum positive detection value and the maximum false detection value closes when more than 40 watermarks are embedded.

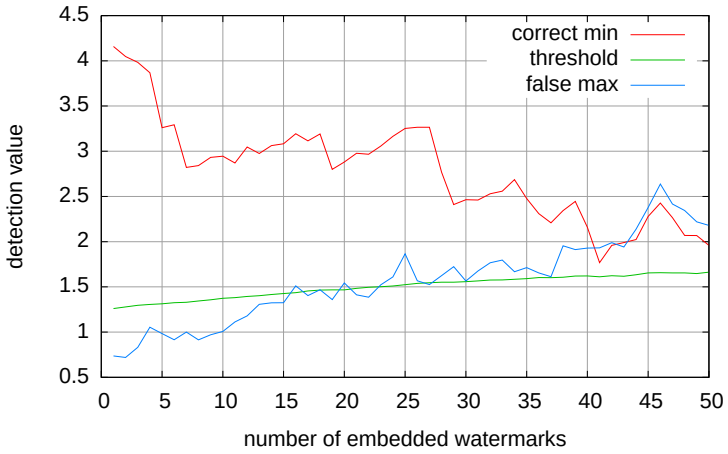


Figure 3.32: Pla: detection values of embedded watermarks compared to false detection values

3.7 Wu

For the implementation some modifications to the proposed algorithm had to be made. First of all the two cases in eq. (7) of [39] were flipped to

$$W'_n = \begin{cases} 1 & \text{if } S_n \geq \gamma \\ -1 & \text{if } S_n < \gamma \end{cases}$$

Next, in our opinion the only reasonable value for γ in the above equation is 0. In the paper γ is in fact set to 0, but because the threshold γ can not have any other value than 0 the equation can be changed to

$$W'_n = \begin{cases} 1 & \text{if } S_n \geq 0 \\ -1 & \text{if } S_n < 0 \end{cases}$$

Although the authors state that they use a 512 bit long watermark for their experiments, the difference image in [39] shows modifications to the whole image. Maybe the watermark is repeatedly embedded until all supertrees are marked but this is not described in the paper and therefore is not in our implementation (see section 3.7.2 and fig. 3.34).

Wu et al. give no order in which the subbands have to be marked. In our implementation first the complete horizontal detail subband is marked, then the vertical and last the diagonal subband.

In all embedding cases the difference between the upper and the lower mean is Δ after embedding. If the difference between the upper and the lower mean is greater than Δ and the two mean values have the wrong orientation the difference stays the same as before the embedding but with different orientation. This can induce severe distortions if the difference between the two mean values is large. We have modified these two cases, so that they also embed to a difference of Δ. This is done by making the same operations as in the cases where the difference is less than Δ. The simple substitution of the operations brings up a new problem. The modification is also considering this emergeing case where one coefficient is very small and the other coefficient is relatively big (both either in the upper or in the lower half) and the coefficients would be decreased by embedding (e.g.: block=[[6,0],[1,1]], watermark=-1, Δ=1). This would cause the small coefficient to go below 0 and so the embedding will not have the desired effect (the absolute value of the small coefficient will be increased instead of decreased). This will make the embedding fail when the small coefficient is less than $(dif - \Delta)/2$ or when the small coefficient is less than $(dif + \Delta)/2$ the embedding will be successful but not with the desired difference of Δ between the two means. This modification brings about 6dB gain in the PSNR.

Figure 3.33: Wu: watermarked image and difference image after embedding, difference image is amplified 5.93 times; watermarked image: Lena, delta: 17, watermark length: 512

3.7.1 Capacity

If the algorithm is implemented as proposed by the authors the capacity depends only on the size of the image. We assume a 512×512 image. A 3-level wavelet decomposition is applied, thus the smallest subbands are 64×64 resulting in 4096 coefficients. All detail subbands at the smallest level can be used as root elements for embedding, giving 12288 coefficients. To embed a bit of watermark information 4 adjacent coefficients from one subband are needed. So the maximum capacity is 3072 bits for a 512×512 image. The watermark is also embedded in the children of these coefficients, but this does not increase the capacity but the robustness.

3.7.2 Visibility

An example of a watermarked image and the associated difference image can be seen in 3.33. In the difference image we can see that only the upper half of the image is watermarked. This is because a 512 bit long watermark is embedded in a 512x512 image and so only one half of one (root-)subband is marked (see fig. 3.34)

The visibility depends on the watermark length and can be controlled with the quantization step size Δ. Figure 3.35b shows that the PSNR is

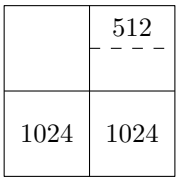
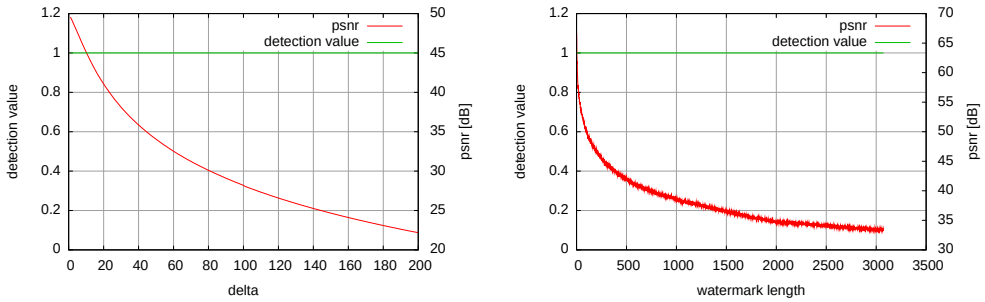


Figure 3.34: Wu: sketch of the area used for embedding. The 4 shown subbands are on the coarsest level



(a) delta (watermark length: 512) (b) watermark length (delta: 17)

Figure 3.35: Wu: effect of the different parameters on the visibility

quickly decreasing with an increasing number of embedded watermark bits. The figure was created using Δ=17 but also with Δ=1 the PSNR is around 38dB when all 3072 trees are marked (this is already with the fixed version as described in the beginning of the section).

3.7.3 Robustness

To decide whether the watermark is present or not Wu et al. compare the normalized correlation to a threshold Φ_T but they give neither a value for the threshold nor a way to calculate it. So we compared the detection value of the embedded watermark to the ones of not embedded watermark. This can be seen in Figure 3.36. No false detection value is above 0.2. For the robustness experiments the watermark is said to be present if the detection value is above 0.2.

For the robustness experiments a 512 bit watermark was embedded and Δ was set to 17 to give an embedding PSNR of approximately 42dB. The results are illustrated in Figure 3.37. Compared to the other algorithms the resistance against jpeg compression is not so good, the watermark can lastly

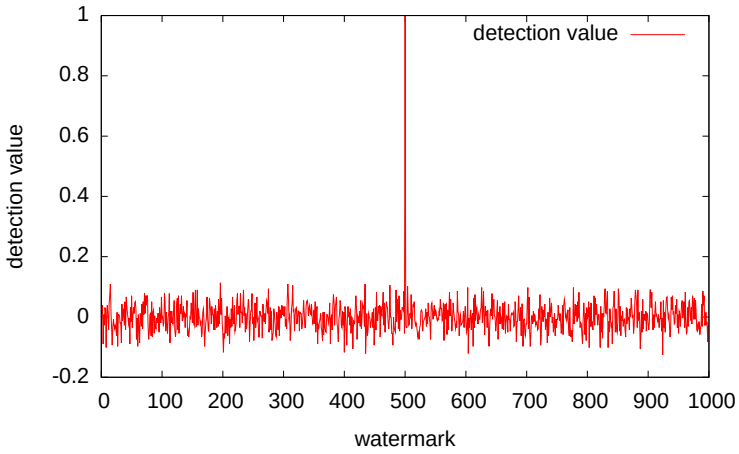
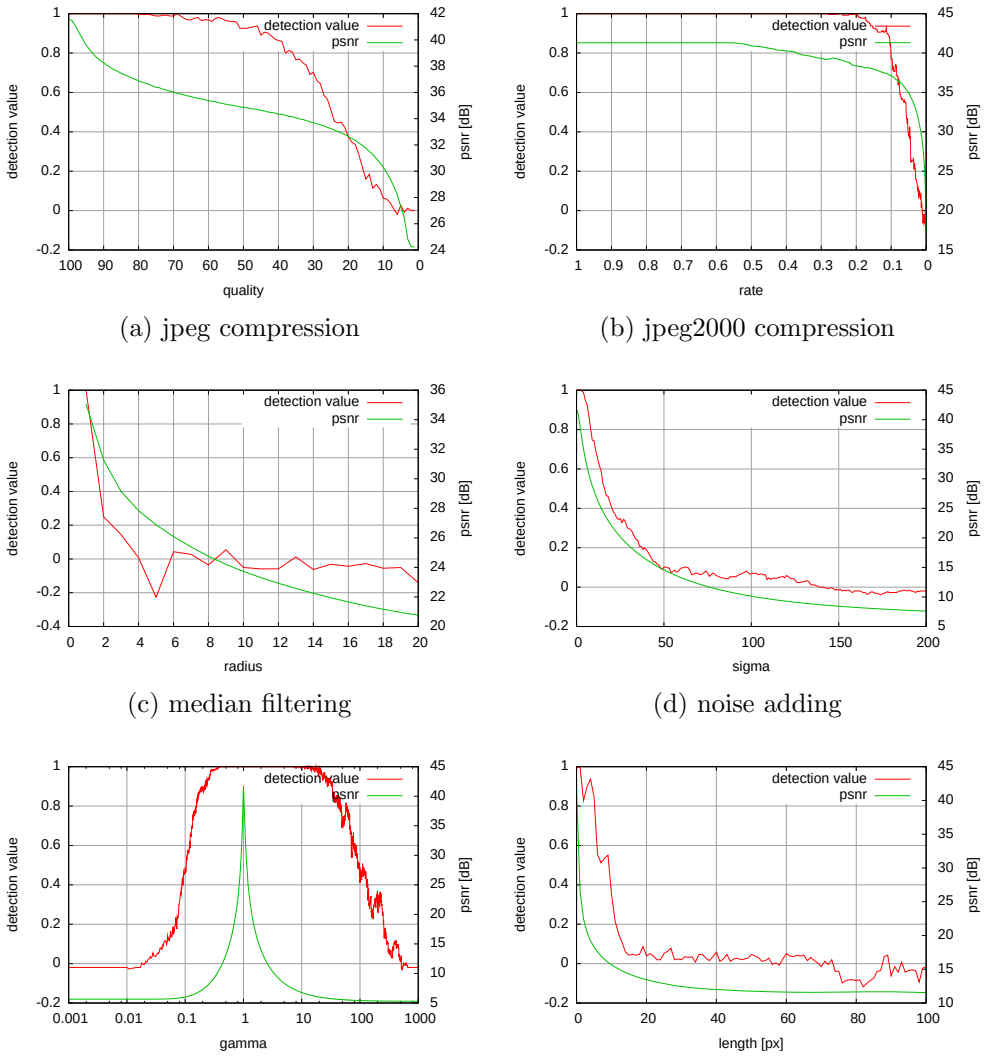


Figure 3.36: Wu: detection values of embedded watermarks compared to false watermarks

be detected with quality=16% (PSNR=32dB). Under jpeg2000 compression the watermark is detectable until rate is less than 0.031 (PSNR=33dB). Also the robustness against image processing operations is rather poor. Median filtering with radius more than 2 (PSNR=31.3dB) removes the watermark and noise can only be added with a sigma of no more than 37 (PSNR=16.9dB). The results on gamma correction is in the same range as the other algorithms with gamma between 0.073 (PSNR=5.8dB) and 255 (PSNR=5.3dB) the watermark can be detected. The algorithm of Wu et al. is the only one in the test which shows significant resistance against geometric attacks. In spite of 11px translation (PSNR=15.3dB) the watermark can still be detected. Rotation again makes the watermark undetectable even with an angle of only 1°.

3.7.4 Multiple Watermarking

With the algorithm as proposed by the authors it is not possible to embed multiple watermarks. In Figure 3.38a it can be seen that only the last embedded watermark can be extracted. In Figure 3.38b the PSNR drops from 41.8dB at the first watermark to 38.2dB at the 10th watermark and stays at this level for the rest of the embedded watermarks. This gradient is because in the first embedding runs not all coefficients are quantized (the ones which have the correct orientation and are greater than delta are left unchanged). In subsequent embedding runs more and more coefficients are quantized and



(a) jpeg compression

(b) jpeg2000 compression

(c) median filtering

(d) noise adding

(e) gamma correction

(f) translation

Figure 3.37: Attacks on the Wu algorithm; watermarked image: Lena, delta=17, watermark length=512

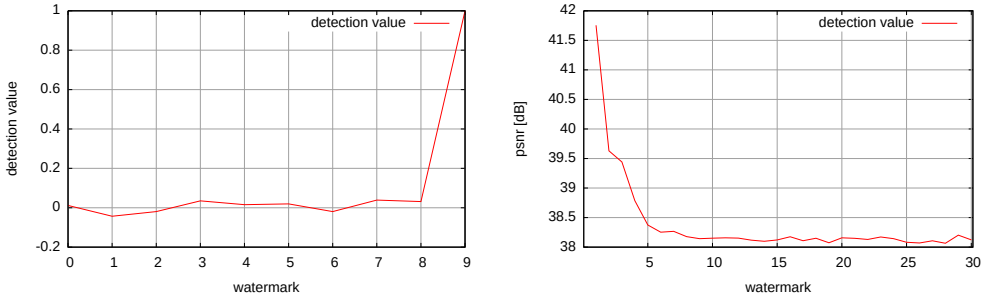


Figure 3.38: Wu: detection value and PSNR when embedding multiple watermarks

the already quantized coefficients stay quantized and just switch orientation if the other bit-value is embedded.

Chapter 4

Multiple watermarking improvements

In this chapter we want to investigate if the results on multiple watermarking of the last chapter can be improved. The basic problem is that there are collisions between the different embedded watermarks. Subsequently embedded watermarks overwrite previous ones partially or even worse in case of the algorithms of Chen and Wu all previous watermarks are completely erased. Apparently the reason is that when an algorithm is applied multiple times the same embedding location will be selected by the algorithm and so if the selected coefficients are altered the already embedded watermarks are also altered and this usually lowers the detection value.

We have implemented 4 improvement techniques to mitigate this drawback. These are the two methods that Wernisch [36] uses in his work:

- parametrized filters
- wavelet packet trees

and two additional ones:

- random coefficient selection
- random carriers

In the following section we will describe how the improvement techniques work and what the benefit for the watermarking algorithms should be. Afterwards the experimental results with the improvement approaches integrated into the watermarking algorithms of the last chapter are presented.

4.1 Improvement techniques

All techniques have in common that each embedder has a unique key which is used to control the embedding process. This key is used as the seed for a PRNG and can be generated by mapping a meaningful text e.g. the name of the embedder or some other (maybe secret) information through a deterministic function. This key is needed in the embedding and in the extraction stage. This renders all algorithms to be semi-blind. If the only purpose of the improvement techniques is to protect the watermarks from overwriteing each other the keys can also be made public. With public available keys the possibility that watermarks can be extracted by anybody is preserved. On the other hand if it is not wanted to have public detectable watermarks the improvement techniques have a second effect. This is watermark hiding and the resulting enhanced security against hostile attacks (see [14]).

4.1.1 parametrized filters

Instead of using predefined wavelet filter banks like the Daubechies-6 filters, which are used in the unimproved version, filters depending on a parameter (the embedder key) are generated. In our experiment the parametrization of Schneid and Pittner [32] is used. To generate these filters Schneid and Pittner present a parametrization for the dilation equation

$$\phi(t) = \sum_{k=0}^{N} c_k \phi(2t - k).$$

This parametrization for the coefficient vector $c^N = (c_0^N, ..., c_N^N)^T$ is done via

$$c^1 = (1, 1)^T$$

and

$$\begin{aligned} c_k^{N+2} = & \frac{1}{2}(c_{k-2}^N + c_k^N) + \\ & + \frac{1}{2}(c_{k-2}^N - c_k^N) cos(\alpha_{(N+1)/2}) + \\ & + \frac{1}{2} c_{N-k+2}^N - c_{N-k}^N)(-1)^k sin(\alpha_{(N+1)/2}). \end{aligned}$$

for $0 \leq k \leq N+2$ and where $c_k = 0$ for $k < 0$ and $k > N$. $\alpha_1, ..., \alpha_{(N-1)/2} \in \mathbb{R}$ are the parameters. From the above equation follows, that the length of the filters is controlled by the number of alphas.

The idea behind using parametrized filters is, that every embedder generates its own embedding space and so the different watermarks do not interfere with each other or at least have less interferences.

For our experiments we use 2 alphas giving a N of 5 and producing 6-tap filters. We have chosen to use 2 parameters to have the same filter length as in the unimproved version. For each embedding run the parameters are randomly ganerated in the range $-\pi \leq \alpha < \pi$ and stored for later extraction of the watermark. We use the generated filters for decomposition and reconstruction in every decomposition level and both directions (horizontal and vertical). The rest of the algorithms stays completely unchanged. Embedding and extraction is done like in the unimproved version.

4.1.2 random coefficient selection

The basic structure of a watermark embedding process in our experiments is:

1. watermark generation
2. wavelet transformation
3. coefficient selection (e.g. all coefficients of the 3 largest detail subbands)
4. embedding via $\hat{I}(x, y) = I(x, y) + \alpha w(x, y)$
5. wavelet reconstruction

We omit algorithm specific techniques like visual masking or sophisticated exploiting of coefficient dependencies in the embedding position selection process for the explanation of the random coefficient selection. Random coefficient selection should be compatible with all these techniques and if special adaptions are required, like for quantization embedding, they are described in the corresponding section later on.

The essential part in the embedding process is, that two vectors are generated. The watermark vector w (step 1) and the vector with the coefficients selected for embedding I (step 3). Random coefficient selection is inserted after the coefficient selection step and before the embedding (i.e. between step 3 and 4). Thereby a set of coefficients is randomly selected and form the new embedding vector $\dot{I}$. This is sketched in figure 4.1. The embedding step is then carried out with these coefficients.

A clear drawback of this method is, that there are less coefficients available for embedding in a single embedding run. In case of spread spectrum embedding this can lead to a worse detection value or in case of quantization

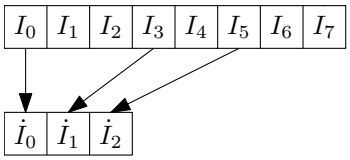


Figure 4.1: elements of the plain image vector I are selected and form a new vector $\dot{I}$

embedding in a reduced payload. The advantage is obviously that there are less interferences when the watermark is embedded in different coefficients. This approach is the adaption of segmented watermarking to our scenario, where there is no central watermarking institution nor are the embedding positions of the previous watermarks known. The difference is, that this approach does no perfect segmentation in the meaning of no overlapping, but therefore every embedder can do the random coefficient selection by himself and needs no additional information.

In our implementation we have chosen to use 10% of I for embedding.

4.1.3 random carriers

For this approach the watermark is modulated with a random carrier before embedding. Thus the basic embedding

$$\hat{I}(x,y) = I(x,y) + \alpha\ w(x,y)$$

is changed to

$$\hat{I}(x,y) = I(x,y) + \alpha\ c(x,y)\ w(x,y).$$

The carrier is chosen to modify the magnitude of the watermark a little bit around the original value. In order to keep the distortion low and not to override visual masking considerations of the algorithm the amount of modulation has to be limited. An appropriate carrier has to be chosen to achieve these results. Possible carriers c_n could be:

$$c_n = x_n'^{(-1^{y_n})}$$

where

$$y_n = \text{ samples of } Y \sim B(1, 0.5)$$

and

$$x_n' = \text{ samples of } X \sim U(1,2) \tag{4.1}$$

or

$$\begin{aligned} x'_n &= |x_n| + 1 \\ x_n &= \text{ samples of } X \sim N(0,1) \end{aligned} \tag{4.2}$$

B denotes a binomial distribution, U a uniform distribution and N denotes a normal distribution. The carrier created by (4.1) modifies the amplitude of the watermark to be in the range of $w/2 \leq w' \leq 2w$ and all possible values of w' have the same probability. The carrier created by (4.2) modifies the watermark to have normally distributed shape around the original value and where the probability that the modulated value w' is in $[w, xw]$ is the same as that the value is in $[w/x, w]$ for all $x \in \mathbb{R}$.

The idea is to lessen the mutual erasure of the watermarks by using different embedding strengths, induced by the random carriers, for different embedding runs. This improvement method can only be reasonably applied to spread spectrum algorithms.

For our experiments we produce the carriers with (4.2). The embedder key is used to seed the PSNR from which the samples of the distributions are taken.

4.1.4 random wavelet packet trees

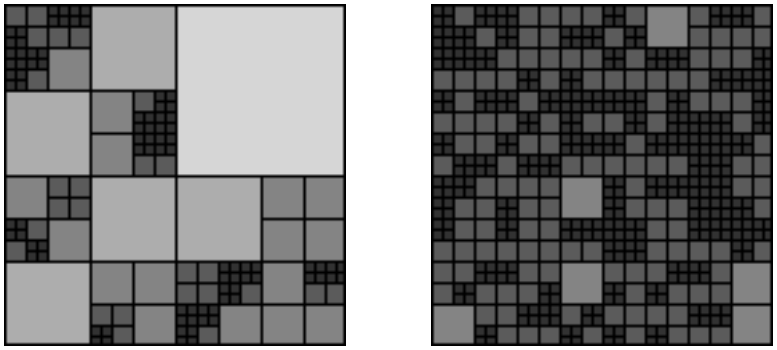
Like Wernisch did in his work [36] we also want to test the possibility for improvement by the usage of different wavelet packet tree structures. The differences between the wavelet packet decomposition and the usual (pyramidal) wavelet decomposition are described in section 1.5.2. Depending on the embedder-key, the image is decomposed into different tree structures. This can for instance be done by applying a fixed probability for each decomposition decision. Although all possible trees can be generated with this method it is more likely to generate lean trees. As Engel suggests in [16], the probability of decomposition has to be different for each decomposition level. The decomposition probability of a node in level l is calculated with

$$p(l) = 1 - \frac{1}{Q_{g-l}}$$

where g is the maximum decomposition level and Q_j is the number of possible decompositions. The number of possible decompositions can be calculated with

$$Q_j = Q_{j-1}^4 + 1$$

as proposed by Xu and Do [40]. The difference between the two embedding strategies can be seen in figure 4.2. The figure shows two typical decomposition trees. But we also like to note that, when no other restrictions are



(a) constant probability (b) level adaptive probability

Figure 4.2: illustration of the two different wavelet packet tree decomposition strategies

enforced, the probability that the image is not decomposed at all is 0.5 when using the constant decomposition probability strategy. This makes the usage of some restrictions on minimal decomposition depth absolutely necessary with this decomposition strategy.

By creating a embedder-key dependent wavelet packet structure each embedder gets his own embedding space. This should improve the multiple watermarking performance. Unfortunately almost all algorithms have to be, sometimes even radical, rewritten because often the algorithms make use of some aspects of the pyramidal structure of the discrete wavelet transform. The necessary adaptions are described in the appropriate sections in detail.

In our experiments we use the decomposition strategy where all trees have the same probability to be created.

4.2 Experimental results

In this section implementation details and results for the multiple watermarking improvements are presented.

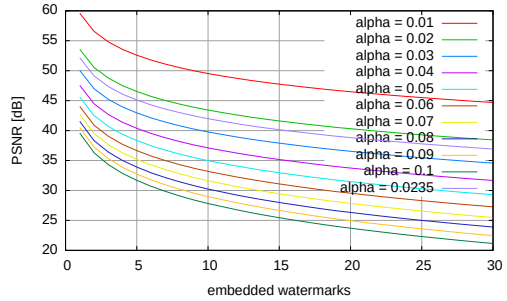
In the last chapter the experiments on multiple watermarking were carried out with the same embedding strength as in the single watermark experiments, which gives a PSNR of approximately 42dB after the first embedded watermark. The embedding strength was chosen to give results which are comparable to the other experiments in the chapter. But for practical use the images are pretty valueless after 10 embedded watermarks. Hence we will change the embedding strength to embed with a PSNR of 42dB after 10

embedded watermarks. The appropriate embedding strength is found iteratively by embedding 10 watermarks, checking the PSNR and adjusting the embedding strength.

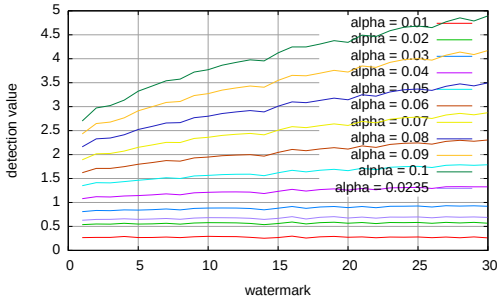
The experimental runs will include the detection performance of the unimproved algorithm and the detection values for each applicable improvement. The embedding strength will be chosen for each improvement method separately and is determined experimentally, by embedding multiple watermarks and the embedding strength is successively adapted.

In the main experiment multiple watermarks are embedded and the detection value of the individual watermarks is calculated. The experiment is repeated with different embedding strengths, especially with the one found in the last step, where the PSNR is 42dB after the tenth watermark. If the usage of a threshold is proposed in the paper of an algorithm, this threshold is also calculated and collected together with the detection values. The desirable result will be, that the detection values of all watermarks will increase or that the detection values of the first embedded watermarks will stay high enough for correct extraction.

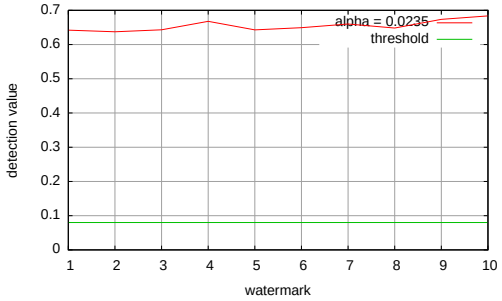
Comparison of the different improvement techniques will be done by comparing the detection values of 10 watermarks, embedded with the embedding strength found in the PSNR experiment.



(a) PSNR



(b) detection value



(c) 10 watermarks

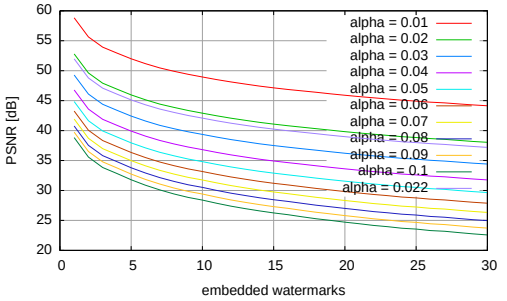
Figure 4.3: PSNR and detection values when embedding multiple watermarks with the algorithm of Barni et al. with different embedding strengths. watermarked image: Lena

4.2.1 Barni

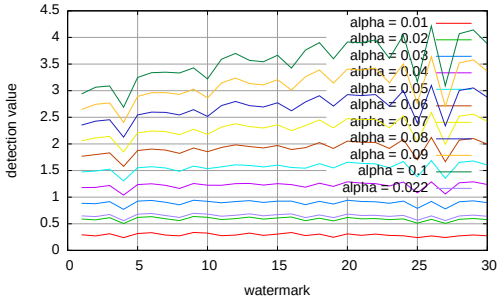
The algorithm of Barni et al. had very good multiple watermarking results in the experiments with the unimproved version in the last chapter. The prior task in this chapter will be to test if this results hold, even if the embedding strength is lowered. Furthermore, all four improvement techniques will be tested whether they can further improve the results.

no improvement

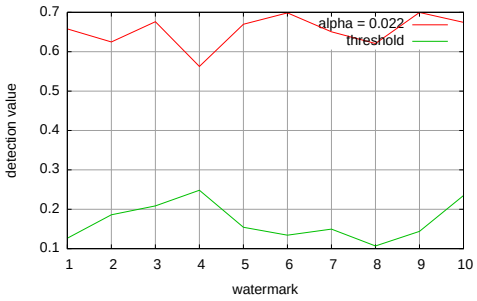
From Figure 4.3a we can see that the desired embedding strength is at $\alpha = 0.0235$, which gives 41.99dB after 10 embedded watermarks. Figure 4.3b shows us that the effect of increasing detection values for later embedded watermarks (see section 3.2.4) is increasing with increasing embedding strength. But because the embedding strength we use is on the lower end of the shown results the detection values should almost be constant for all



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.4: PSNR and detection values when embedding multiple watermarks with the algorithm of Barni et al. and parametrized filters improvement with different embedding strengths. watermarked image: Lena

embedded watermarks. The result for $\alpha = 0.0235$ is plotted in Figure 4.3c. When embedding a single watermark with an embedding strength of 42dB the detection value is 2.2, The average detection value of the 10 embedded watermarks is clearly below this value, but with an average value of 0.65 still above the threshold of 0.08.

parametrized filters

For the first improvement almost no changes have to be made to the algorithm. The only modification is, that instad of the Daubechis-6 filter, the parametrized filter is used for wavelet transformation and reconstruction.

From Figure 4.4a we see that α has to be 0.022 to get the desired embedding strength (40.08dB). The differences to the unimproved version, which can be observed when we compare figure 4.3c and figure 4.4c, is first the lower increase of detection values for later embedded watermarks and sec-

ond the detection values are not as constant as in the unimproved version. The first fact is because of the different embedding spaces which are created by the use of the different wavelets. The watermarks have less interferences and so the difference between the watermarks embedded in different runs is lower. This is the anticipated effect of the improvement technique, the impact of the interferrecnce is mitigated. But because of the very special property of this algorithm in combination with multiple watermarking, that detection values of the first watermarks are not decreasing but the ones of the later ones are increasing (see section 3.2.4) with rising number of embedded watermarks, the effect of the parametrized wavelets is not that the older watermarks are better detectable but that the detection value of later embedded ones is worse. So, although the interferences of the watermarks are reduced, the detection performance is not improved. The second fact lets us assume that the parametrization produces "good" and "poor" wavelets which lead to higher or lower detection values and higher or lower thresholds in general. All effects scale with the embedding strength and because our target embedding strength is very low, the effects are softened and do not make a big difference (see figure 4.4c).

random coefficient selection

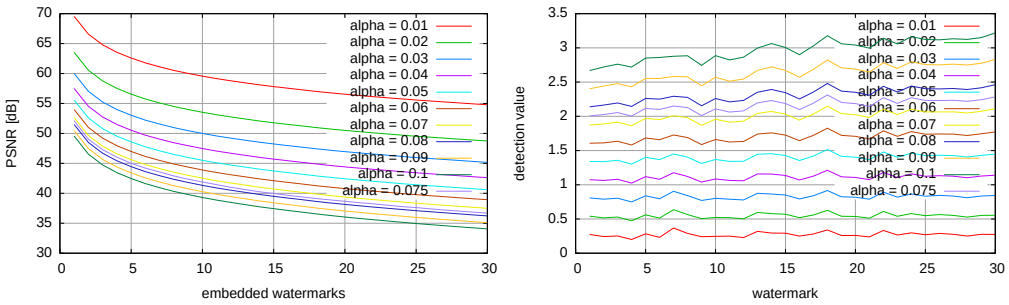
For the random coefficient selection improvement technique the watermark is embedded in randomly chosen 10% of the detail subbands of the first decomposition level. The watermark vector is shortened to cover the new embedding space. PSNR and detection value results can be seen in Figure 4.5. The PSNR after 10 embedded watermarks when embedding with $\alpha = 0.075$ is 41.9dB. Since only a fraction of the coefficients of the unimproved algorithm are marked the strength factor can be significantly higher and therefore the detection value is also higher as figure 4.5b shows. The average detection value for the target embedding strength over 10 watermarks is 2.08. At the same time the threshold is also increased to 0.27.

random carriers

First a carrier signal in the same length as the watermark is created with the technique described in the last section (4.1.3). Then the watermark is modulated with

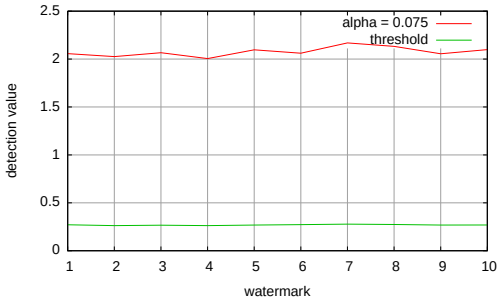
$$\overline{w}_n = w_n \, c_n.$$

The embedding process is carried out like in the unimproved version but with the modulated watermark $\overline{w}$. The strength factor to get the targeted 42dB is 0.0165. In the detection stage the vector of watermarked coefficients is



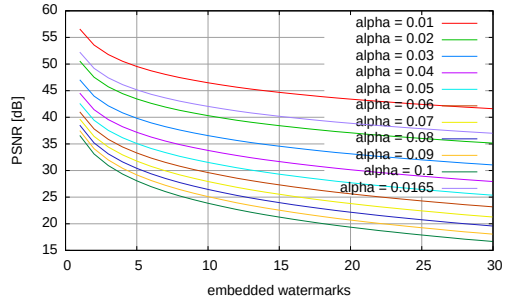
(a) PSNR

(b) detection value

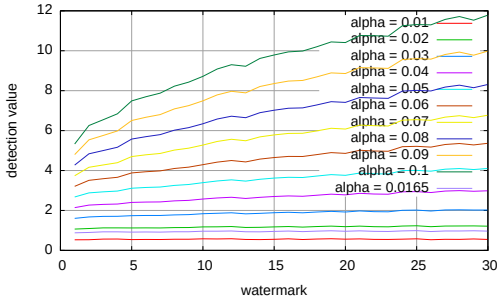


(c) 10 watermarks

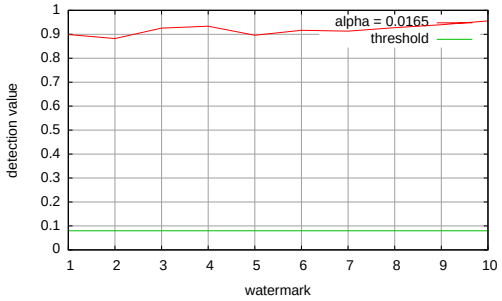
Figure 4.5: PSNR and detection values when embedding multiple watermarks with the algorithm of Barni et al. and random coefficient selection improvement with different embedding strengths. watermarked image: Lena



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.6: PSNR and detection values when embedding multiple watermarks with the algorithm of Barni et al. and random carrier improvement with different embedding strengths. watermarked image: Lena

extracted and the correlation is calculated with the modulated watermark $\overline{w}$. Figure 4.6 shows that the shape of the curves look very similar to the unimproved ones. The difference to the unimproved version becomes apparent in figure 4.6c, where we can see that the detection value with an average of 0.92 is higher whereas the threshold stays at 0.08.

random wavelet packet trees

We tried to make least possible modifications to the algorithm as proposed in the paper to include the improvement techniques. The incorporation of random wavelet packet trees comes not as cheap as the other improvements.

We first decided to keep the embedding position as in the algorithm as proposed, this are the three finest detail subbands and further decompose these three subbands into wavelet packet trees. The wavelet packet trees have a maximum decomposition depth of 4. The ll-subband is also decom-

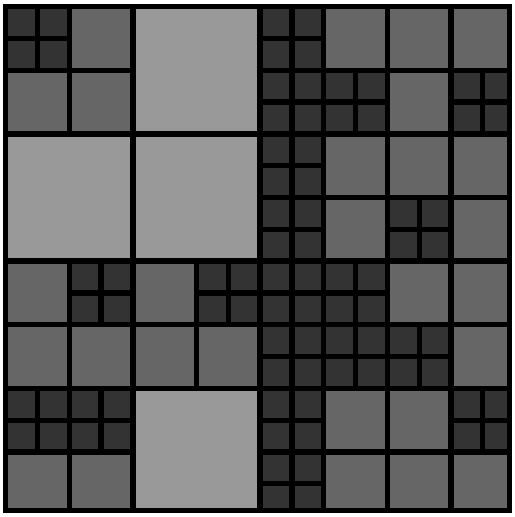
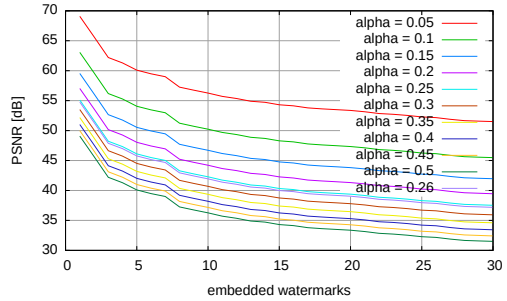


Figure 4.7: sample decomposition tree used for the random wavelet packets approach with the algorithm of Barni et al.

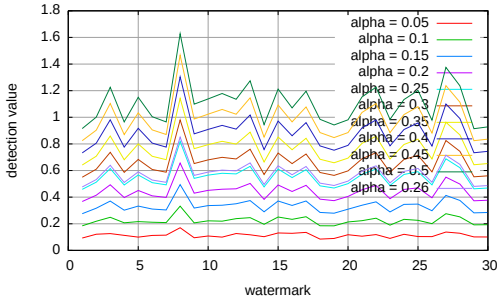
posed into 4 levels with the pyramidal decomposition strategy. This is needed for the visual mask generation. A typical decomposition tree is illustrated in figure 4.7. The watermark has the same length as in the unimproved version but a new mapping function has to be applied. We assign the watermark elements to the coefficients by walking through the tree in the order ll, hl, lh, hh and assign an appropriate number of watermark elements to the coefficients in the leafs of the tree.

Because the calculation of the visual mask is the key feature of the algorithm (see section 2.1.1) we do not want to change the way it is created. The best way to adapt the visual mask to wavelet packet trees we found is to create the visual mask of the pyramidal tree and assign the appropriate elements of the visual mask to the coefficients of the wavelet packet decomposition. For the assignment of the mask elements the decomposition depth of the coefficient and the base orientation is used. The base orientation (horizontal, vertical or diagonal) is the orientation of the decomposition at the first level. We use the base orientation because the local orientation does not necessarily reflect the orientation of image elements any more when using wavelet packet decomposition but the idea of the authors is based on the orientation of image elements.

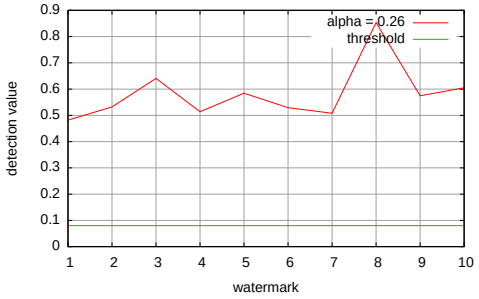
The results are shown in Figure 4.8. α is set to 0.26 which gives an embedding PSNR of 41.94dB after 10 watermarks. Like with the parametrized filters there seem to be “good” and “poor” decomposition structures result-



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.8: PSNR and detection values when embedding multiple watermarks with the algorithm of Barni et al. and random wavelet packets improvement with different embedding strengths. watermarked image: Lena

ing in unstable detection values. With an average of 0.58 the detection value is lower than with all other versions of the algorithm.

comparison

First of all, the algorithm could prove its multiple watermarking qualities even with low embedding strength. When the different improvement techniques are compared by the detection values they achieve, random coefficient selection clearly outperforms all other approaches and brings a clear enhancement in watermark detectability. The approach with random carriers also brings a little enhancement but the other two techniques make no difference compared to the original algorithm. Random wavelet packets even lower the average detection value a bit. Results are illustrated in Figure 4.9. Taking also the thresholds into consideration we see in Figure 4.10 that the parametrized filters and the random coefficient selection approach have

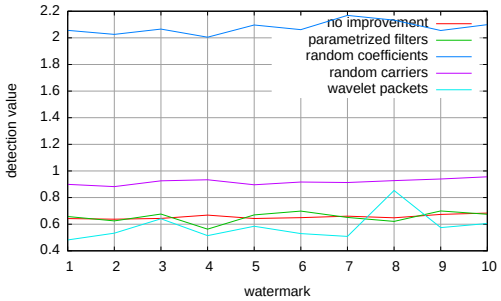


Figure 4.9: comparison of the detection values of the different improvement techniques on the algorithm of Barni et al.

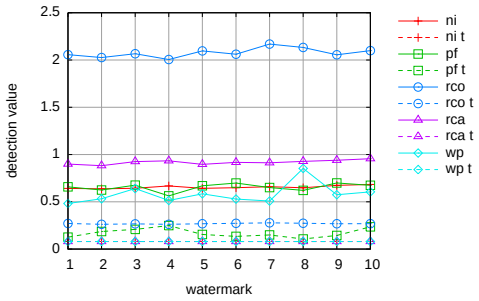
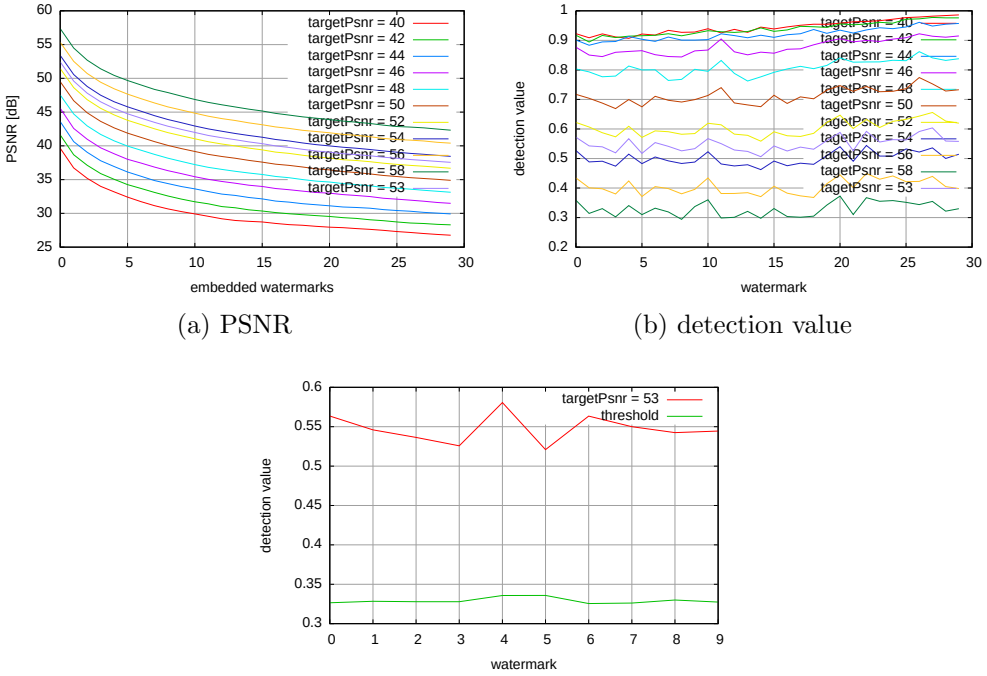


Figure 4.10: comparison of the detection values and thresholds of the different improvement techniques on the algorithm of Barni et al.

higher thresholds than in the unimproved version. Especially in the case of random coefficient selection the question arises if the higher threshold outweighs the higher detection value. This question will be discussed in the next chapter (chap. 5). For better readability we have abbreviated the improvement techniques, no improvement with “ni”, parametrized filters with “pf”, random coefficients with “rco”, random carriers with “rca” and wavelet packets with “wp”. An appended “t” denotes the according threshold.



(a) PSNR

(b) detection value

(c) 10 watermarks

Figure 4.11: PSNR and detection values when embedding multiple watermarks with the Broken Arrows algorithm with different embedding strengths. watermarked image: Lena

4.2.2 Broken Arrows

Like the algorithm of Barni et al. of the last section, Broken Arrows has excellent multiple watermarking performance without improvements. It is to test if this performance can be kept up when we target a high image quality after several embedding steps. Because the algorithm is a zero-bit algorithm we use the secret projection key to embed multiple watermarks, as already mentioned in section 3.3.4. The random carrier improvement is not applicable to this algorithm since there is no watermark vector to modulate.

no improvement

In order to reach 42dB after 10 embedded watermarks the target PSNR has to be 53 (gives 41.99dB). Figure 4.11b shows that all watermarks can be extracted as long as the target PSNR stays above 57dB. This result is similar to the results for single watermarking (see figure 3.9). With the chosen

embedding strength almost no decrease from the last to the first watermark can be observed. As we can see in Figure 4.11c the average detection value is definitely below 1 but still above the threshold so all watermarks are extractable.

parametrized filters

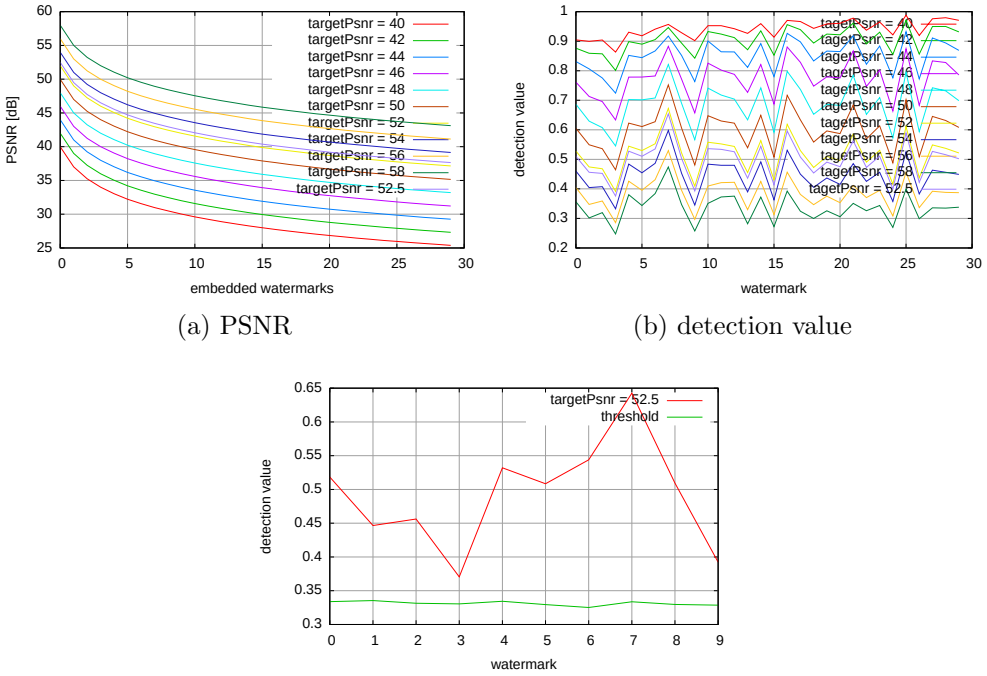
As said in section 3.3 we use the C implementation of Furon and Bas for this algorithm. In order to insert the parametrized filters improvement technique we implemented an import/export facility in C and Python. The wavelet transformation is done in the Python part, like for all other algorithms and is then imported into the Broken Arrows implementation and used as the vector S_X. Then the watermarking process is carried out and the result vector S_Y is returned to the Python program where the wavelet reconstruction is performed. The detection process is the same as the embedding process except that there is no need to return a wavelet coefficient vector.

From Figure 4.12a we see that we reach a PSNR of 42.06dB after 10 watermarks when we choose 52.5 as the target PSNR for each embedding step. The detection values results show fluctuations like with the last algorithm and the parametrized filters approach which confirms the assumption that there are well- and ill-suited wavelets. The fluctuations increase with decreasing embedding strength. From this it follows that a stronger embedded watermark can compensate the ill-suited wavelet. But because our chosen embedding strength is on the lower end of possible embedding strengths the compensation has not much impact. Though, as we can see in Fig. 4.12c, all watermarks can be detected. In fig. 4.12b we can see the expected tendency where older watermarks have a lower detection value than newer ones. In fig. 4.12c the tendency is hard to observe due to the high fluctuations.

random coefficient selection

For this improvement techniques we use the same import/export facility as for the parametrized filters. The wavelet transformation is done in Python and a subset of the coefficients (10%) is randomly chosen and passed to the Broken Arrows algorithm. These coefficients are then watermarked, passed back and inserted at the corresponding positions in the wavelet subbands.

We choose a target PSNR of 52.4 and reach a PSNR of 41.97dB after the 10th watermark. The results in fig. 4.13b also show fluctuation like with the parametrized filters approach, but with smaller amplitude. In the experiment with 10 embedded watermarks (fig. 4.13c) all watermarks can be extracted.

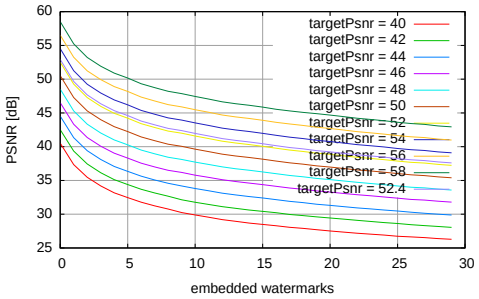


(a) PSNR

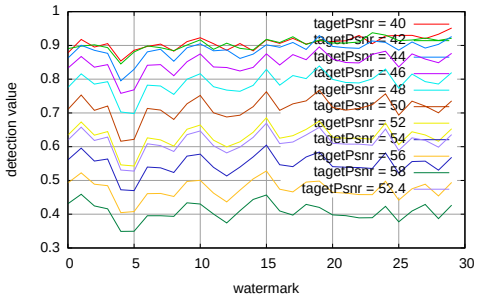
(b) detection value

(c) 10 watermarks

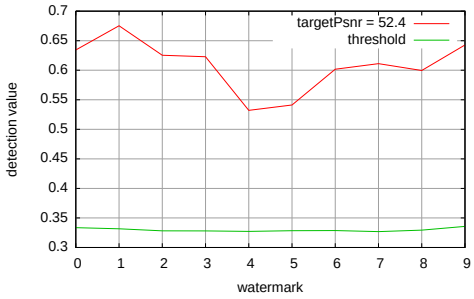
Figure 4.12: PSNR and detection values when embedding multiple watermarks with the Broken Arrows algorithm and parametrized filters improvement with different embedding strengths. watermarked image: Lena



(a) PSNR

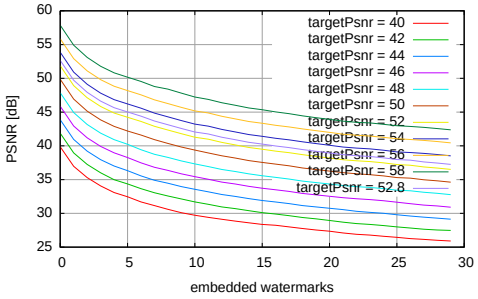


(b) detection value

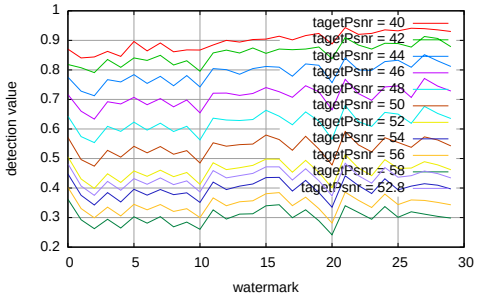


(c) 10 watermarks

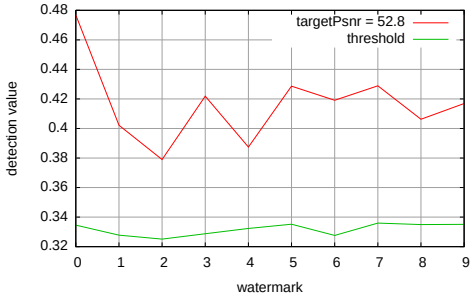
Figure 4.13: PSNR and detection values when embedding multiple watermarks with the Broken Arrows algorithm and random coefficient selection improvement with different embedding strengths. watermarked image: Lena



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.14: PSNR and detection values when embedding multiple watermarks with the Broken Arrows algorithm and random wavelet packets improvement with different embedding strengths. watermarked image: Lena

random wavelet packet trees

The wavelet packet tree generation, transformation and reconstruction is done in Python, the coefficients are passed to the Broken Arrows algorithm where they are marked.

Target PSNR=52.8 gives 42.06dB after 10 embedded watermarks. The detection values are more stable than in the other improvement techniques, but with random wavelet packets the detection value in general is the lowest of all. Also with this technique all 10 watermarks can be extracted (fig. 4.14c).

comparison

As we can see in figure 4.15 only random coefficient selection can improve the original algorithm. Although the detection values are more instable than the ones from the unimproved algorithm the detection value is higher in general.

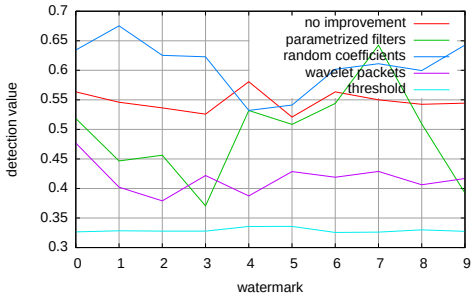


Figure 4.15: comparison of the detection values of the different improvement techniques on the Broken Arrows algorithm

The explanation for the inability of the other improvement techniques must be on the one hand the very good multiple watermarking performance of the original algorithm is hard to improve and on the other hand the secret projection already disperses the watermarks in the embedding space which can not be further improved by our techniques.

4.2.3 Cao

The results of the experiments on multiple watermarking in the last chapter showed good performance of the algorithm. All watermarks in the test could be extracted. We want to remind, that the decrease of the average detection value with rising number of embedded watermarks had a bigger impact than the local developement, this is the detection value decrease from the first to the last watermark. Therefore we have to pay attention to the average height of the detection values and whether this value can be increased by the improvement approaches.

We do not apply the random wavelet packet decompositions technique because the wavelet packet decomposition creates a big number of subbands and in combination with the RDWT where each subband has the size of the original image the computational costs and memory requirements make this approach completely unpractical.

no improvement

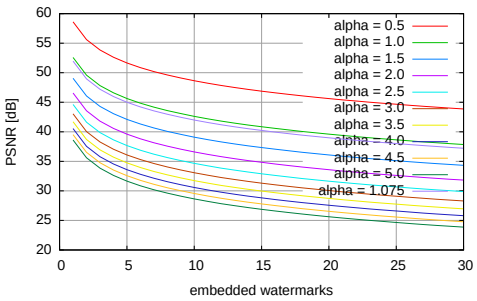
We use α=1.075 as the strength factor and obtain a embedding strength of 41.99dB.

The decrease from the first to the last embedded watermark is stronger with higher embedding strength, but because we use a very low embedding strength this effect almost disappears (see fig. 4.16b). To evaluate the multiple watermarking performance we still stick to the same procedure like for the other algorithms, allthough this algorithm heavily depends on the number of embedded watermarks. If an algorithm can detect the last 5 watermarks, it makes almost no difference for the detectability if 5 or 10 watermarks are embedded. Unlike the other algorithms this one can fail to detect any watermark if 10 are embedded but can be able to detect all watermarks if only 5 are embedded. This dependency can be observed when figure 4.16c and the according line in figure 4.16b are compared. Although the strength factor is the same the average detection value when 30 watermarks are embedded is 0.16 and the one when 10 watermarks are embedded is 0.25, while the detection values from the first to the last watermark are almost constant.

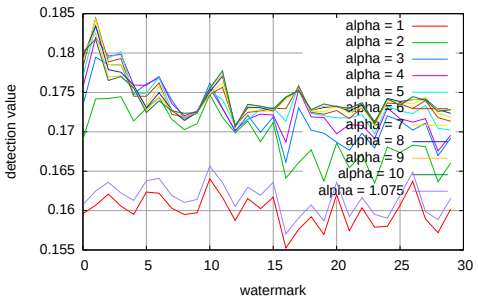
parametrized filters

Also the RDWT can use the parametrized filters without any changes, since the only difference is that instead of the subsampling of the data the filters are upsampled.

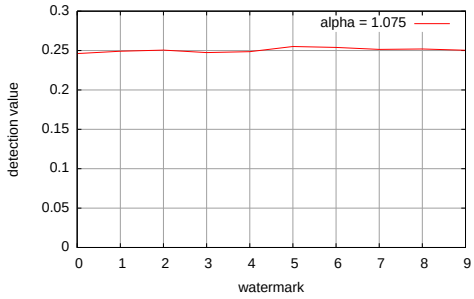
Figure 4.17a shows that we can obtain an embedding strength of 41.99dB when α is chosen to be 1.075.



(a) PSNR

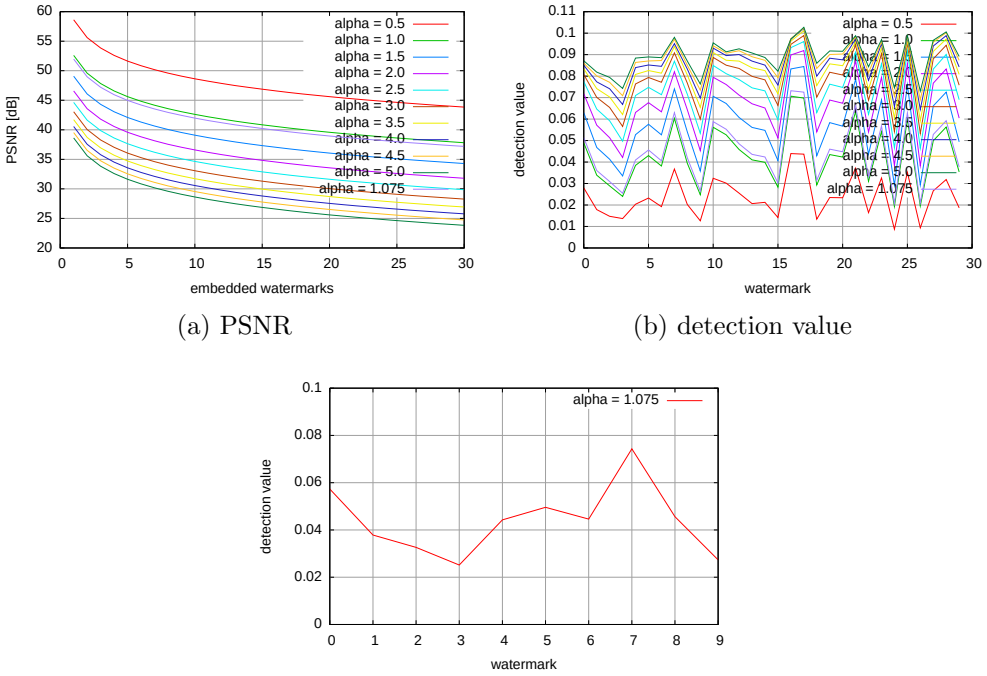


(b) detection value



(c) 10 watermarks

Figure 4.16: PSNR and detection values when embedding multiple watermarks with the algorithm of Cao et al. with different embedding strengths. watermarked image: Lena



(a) PSNR

(b) detection value

(c) 10 watermarks

Figure 4.17: PSNR and detection values when embedding multiple watermarks with the algorithm of Cao et al. and parametrized filters improvement with different embedding strengths. watermarked image: Lena

In contrast to the unimproved version the detection values are more dependent on the embedding strength. Also the detection values are more varying. We already could see this effect in the experiments with parametrized filters and other watermarking algorithms, hence the reason for the variations must be the good or bad suitability of the different paramtrized wavelets. The strong dependency of the detection value on the number of embedded watermarks is lowered with this modification (compare figure 4.17b and figure 4.17c) but the average detection value is clearly below the one of the original version of the algorithm. Though, all detection values are above the threshold of 0.008 and so all watermarks can be detected.

random coefficient selection

For the implementation of the random coefficient selection approach we randomly select 10% of the coefficients of one subband and embed the watermark in these xy-positions in all of the three subbands to mark.

Because of the smaller number of marked coefficients the α can be higher and for a α of 3.4 we get an embedding PSNR of 41.97dB.

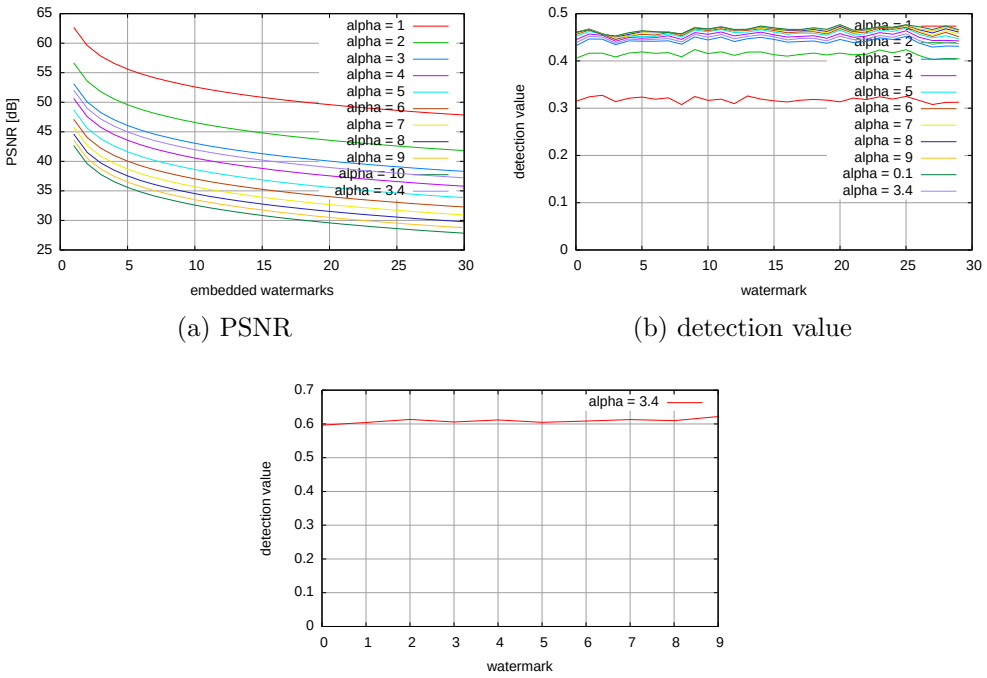
Like in the unimproved version the detection value does not increase significantly when α is 3 or above. So the detection value can not be much higher than 0.45 with the 30 embedded watermarks. The average detection value is clearly higher as in the unimproved version and the detection values are also more stable for all watermarks. We further can observe that the tendency of the detection values is like the expected one and not inverted like in the original version of the algorithm, where later embedded watermarks have a higher value than previous one (see fig. 4.18b and 4.18c).

random carriers

To integrate the random carrier modulation improvement approach, the watermark vector modulated with the generated carrier before embedding according to 4.1.3. Everything else can stay unchanged.

We embed with α=0.75, this results in an embedding PSNR of 42.09dB (see fig. 4.19a).

The detection values in fig 4.19b look very similar in terms of tendency and variability, but the average value is lower than with the unmodified algorithm. This is also the case in the experiment with 10 embedded watermarks, which is illustrated in fig 4.19c. Using the threshold of section 3.4.3 all watermarks are detectable despite of the reduced detector performance.

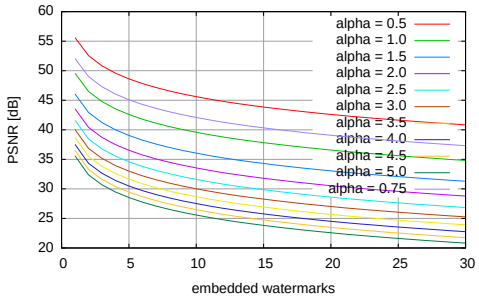


(a) PSNR

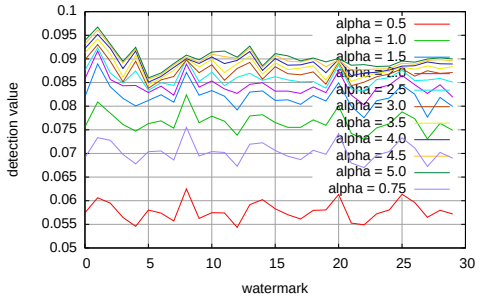
(b) detection value

(c) 10 watermarks

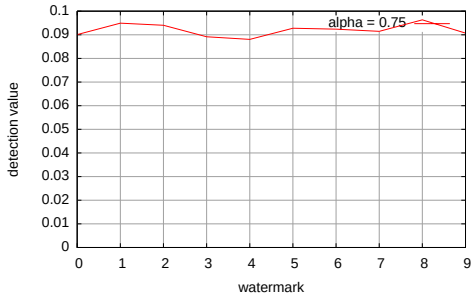
Figure 4.18: PSNR and detection values when embedding multiple watermarks with the algorithm of Cao et al. and random coefficient selection improvement with different embedding strengths. watermarked image: Lena



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.19: PSNR and detection values when embedding multiple watermarks with the algorithm of Cao et al. and random carriers improvement with different embedding strengths. watermarked image: Lena

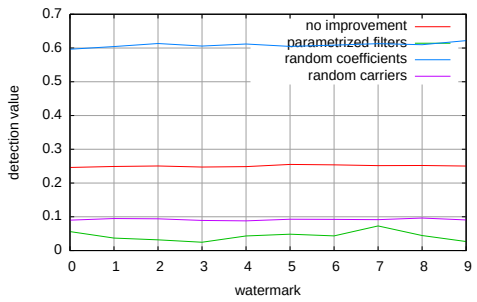
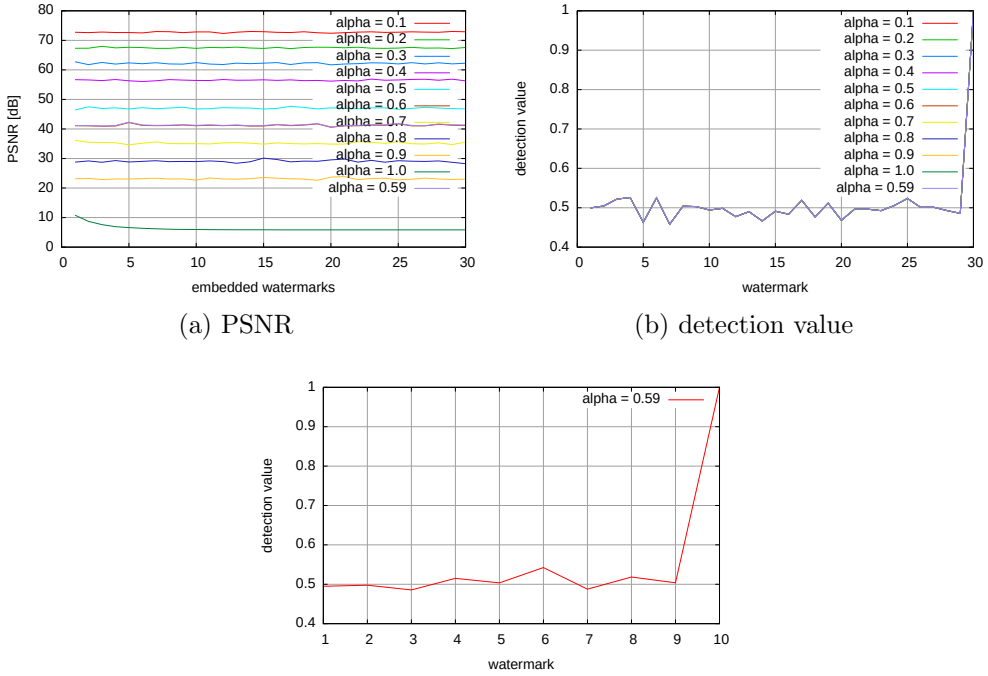


Figure 4.20: comparison of the detection values of the different improvement techniques on the algorithm of Cao et al.

comparison

Out of the three improvement approaches in th test, parametrized filters and random carrier modulation are clearly worse then the unmodified version. But random coefficient selection achieves a big advantage, the average detection value is more than doubled. However, no algorithm for threshold calculation is given by Cao et al. and we use an experimentally obtained threshold, which was determined by looking at the detector response of a single embedded watermark.



(a) PSNR

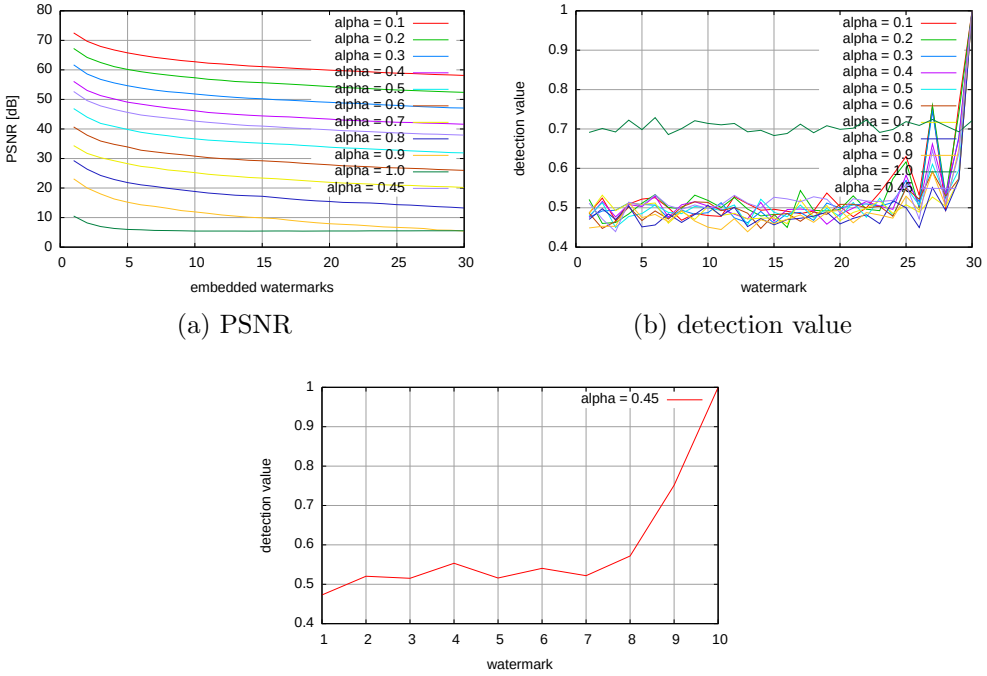
(b) detection value

(c) 10 watermarks

Figure 4.21: PSNR and detection values when embedding multiple watermarks with the algorithm of Chen et al. with different embedding strengths. watermarked image: Lena

4.2.4 Chen

The algorithm of Chen et al. is absolutely unsuited for multiple watermarking when used as proposed. Only the last embedded watermark can be extracted, all other watermarks are erased (see section 3.5.4). In this section we will test if the algorithm can be enabled to detect multiple watermarks. Because this is a quantization based algorithm the random carriers improvement can not be applied. Because of the stepwise change in the embedding PSNR it is not possible to reach an embedding PSNR of 42dB exactly without distortion reduction. However, we will not use distortion reduction in our experiments because this changes many bits in the coefficient and thus can reduce the multiple embedding performance.



(a) PSNR

(b) detection value

(c) 10 watermarks

Figure 4.22: PSNR and detection values when embedding multiple watermarks with the algorithm of Chen et al. and parametrized filters improvement with different embedding strengths. watermarked image: Lena

no improvement

For the sake of completeness we carry out the experiments and confirm the results of the last chapter. Only the last watermark can be detected independent of the embedding strength. The embedding PSNR stays constant for any number of embedded watermarks (for the explanation see section 3.5.4).

parametrized filters

The parametrized filters improvement implementation is straight forward just using the generated filters for transformation and reconstruction and the rest of the algorithm stays the same.

In contrast to the version without modifications the number of embedded watermarks now influences the image quality. This is because the different embedding runs do not exactly embed into the same positions due to the different transformation spaces. With $\alpha = 0.45$ (PSNR after 10 embedded

watermarks 42.67dB) the embedding PSNR is closest to 42dB. In Figure 4.22b the $\alpha = 1.0$ line can be ignored, besides the very low PSNR, the embedding process is errornous for some coefficients (see section 3.5.2) with this strength factor. We use the threshold 0.6 of section 3.5.3. In figure 4.22b the detection value again drops very quickly to below the threshold of 0.6. Further more the detection value is independent of the embedding strength in this experiment. Figure 4.22c shows that the last two watermarks can be detected. This is a small improvement compared to the unmodified version.

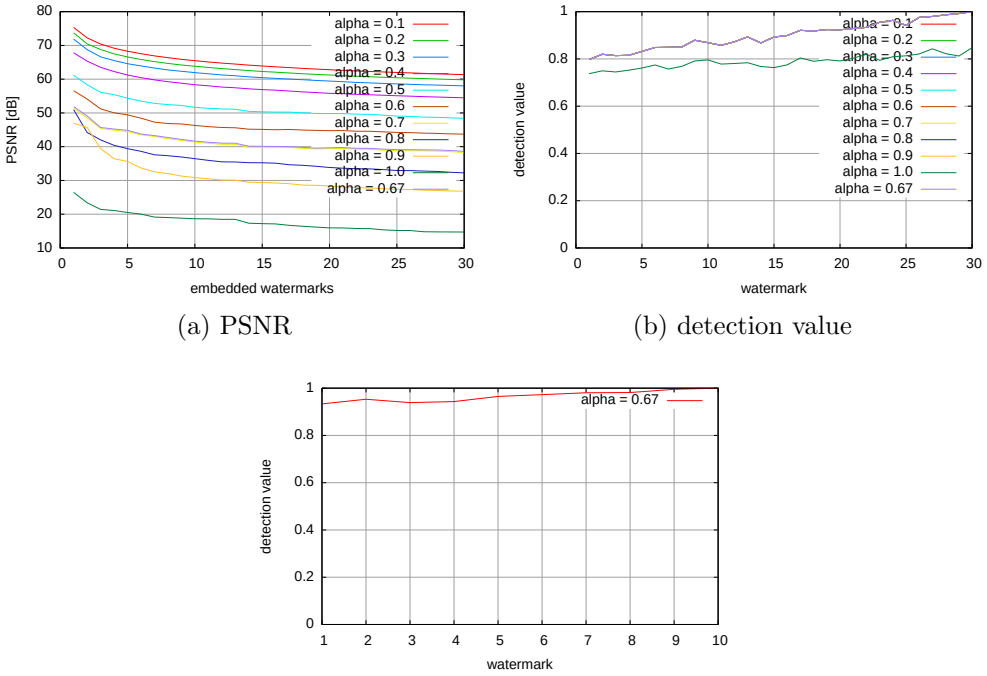
random coefficient selection

For the implementation of random coefficient selection a set of coefficients to choose from has to be defined. In the original algorithm the authors start embedding in the approximation image and then carry on embedding from the coarsest detail subbands to the finest ones, depending on the length of the watermark. To keep the intention to embed in the coarse parts of the image all subbands except the finest detail subbands are defined as the set of coefficients for random selection. Then as many coefficients as needed to embed the whole watermark are selected.

We use $\alpha = 0.67$ to reach an embedding PSNR of 41.65dB. But as we can see in fig. 4.23b the detection value is not depending on the embedding strength. This result is clear because an older value gets overwritten when the same coefficients are selected and the opposite bit is embedded. This is independent of α. The only embedding strength which has different detection values is 1, this special case is explained in section 3.5.2. Not only all watermarks in fig. 4.23c can be detected, even all 30 watermarks of fig. 4.23b. Older watermarks have slightly smaller detection values than newer ones as expected because of the collisions when choosing a coefficient position twice. Since only few coefficients (1024 as proposed) have to be marked out of 65536 there are only very few collisions and this is the reason for the good results with this improvement technique.

random wavelet packet trees

To implement this improvement technique we first decompose the image into wavelet packets with a maximum decomposition depth of 7. Then we choose the embedding subbands according to the original algorithm. We start by embedding into the approximation image and then in the detail subbands on the same level. If one of these subbands is decomposed the children of this subband are marked in the order ll, hl, lh, hh If there are watermark elements left, the detail subbands of the next level are used for embedding

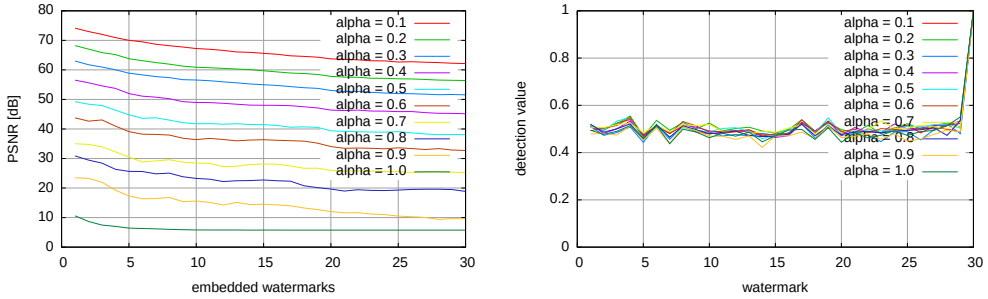


(a) PSNR

(b) detection value

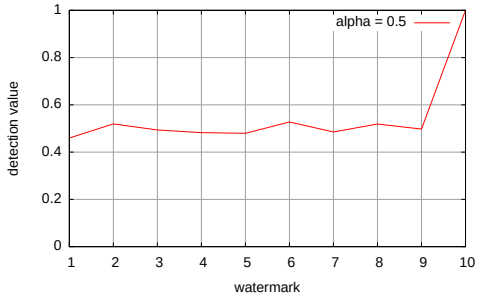
(c) 10 watermarks

Figure 4.23: PSNR and detection values when embedding multiple watermarks with the algorithm of Chen et al. and random coefficient selection improvement with different embedding strengths. watermarked image: Lena



(a) PSNR

(b) detection value



(c) 10 watermarks

Figure 4.24: PSNR and detection values when embedding multiple watermarks with the algorithm of Chen et al. and random wavelet packets improvement with different embedding strengths. watermarked image: Lena

and so on.

For this improvement approach α is chosen to be 0.5, this gives an embedding PSNR of 41.75dB. The detection results are very similar to the ones of the unimproved version. Only the last watermark can be detected. The only difference is, that the detection values of different embedding strengths are not perfectly the same any more. The reason for the poor results can be seen in fig. 4.25. The figure shows a level 7 sample decomposition and the dark grey area is the embedding position. Because of the low number of marked coefficients compared to the total number of coefficients of the image there are not many possibilities to decompose them and so the watermark is embedded in related coefficients. This leads to heavy interferences and the cancelation of the earlier watermarks. Of course it would be possible to randomly select the subbands to embed, but then the improvement would not come from the different wavelet packet structures but from the random embedding positions. When the improvement technique relies on the random

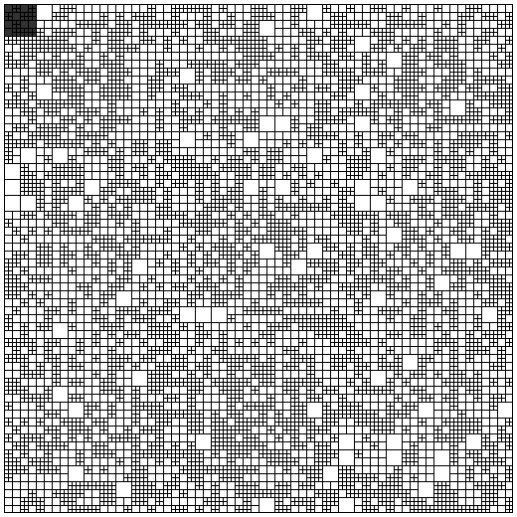


Figure 4.25: embedding positions for the random wavelet packets approach with the algorithm of Chen et al.

embedding position the random coefficient selection technique is the more consequent one and even better technique because the marked coefficients are spread over a bigger part of the coefficients and not be packed together like it is the case when a whole subband is marked. If a collision occurs, a bigger part of the watermark is damaged in the case of wavelet packet embedding.

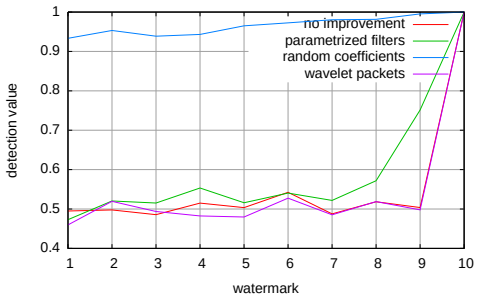
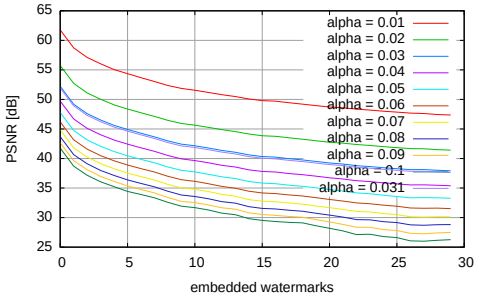


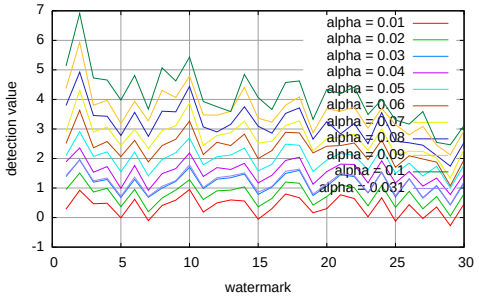
Figure 4.26: comparison of the detection values of the different improvement techniques on the algorithm of Chen et al.

comparison

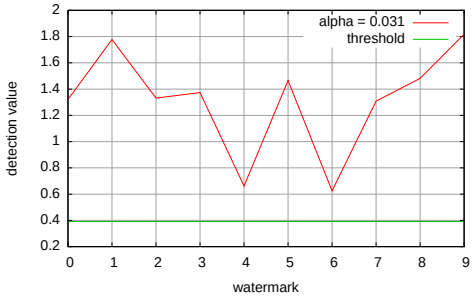
Although there was a great improvement potential due to the bad performance of the original algorithm, wavelet packets completely miss to improve the multiple watermarking capability. The parametrized filters approach achieves a little improvement. The detection value for all of the watermarks is higher compared to the original version but finally only the last two watermarks are detectable with respect to the threshold of 0.6. A big improvement could be achieved with the random coefficient selection technique. All watermarks can be extracted with this modification.



(a) PSNR



(b) detection value



(c) 10 watermarks

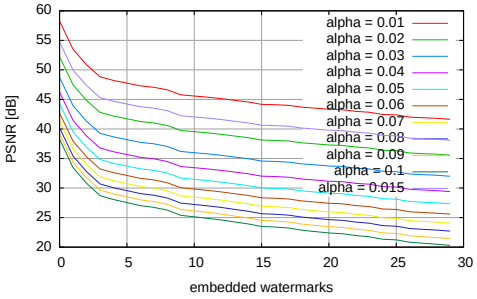
Figure 4.27: PSNR and detection values when embedding multiple watermarks with the algorithm of Pla et al. with different embedding strengths. watermarked image: Lena

4.2.5 Pla

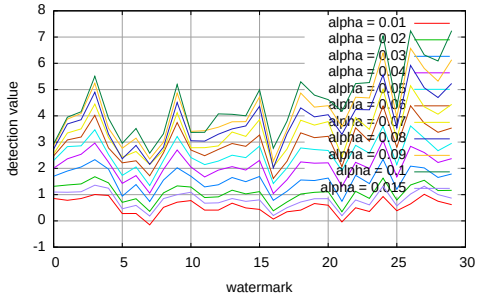
The unmodified algorithm of Pla et al. was able to detect multiple watermarks as presented in section 3.6.4. We are going to investigate if the results can be improved with any of the four improvement approaches.

no improvement

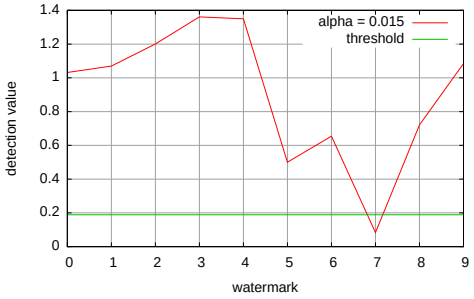
The desired embedding PSNR of 42dB can be attained with $\alpha = 0.031$. All watermarks are detectable as long as α is 0.3 or greater. Our chosen embedding strength is slightly above so all watermarks can be extracted. The detection values are rather strong varying, the higher α is, the stronger this effect occurs. Te results can be seen in figure 4.27.



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.28: PSNR and detection values when embedding multiple watermarks with the algorithm of Pla et al. and parametrized filters improvement with different embedding strengths. watermarked image: Lena

parametrized filters

The implementation of the parametrized filters improvement is as simple as with the other algorithms. Only the utilized wavelet has to be replaced with the parametrized one and the remaining algorithm stays unchanged.

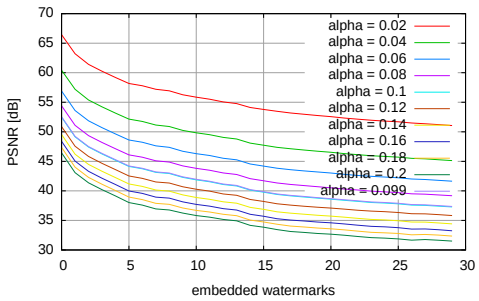
From fig. 4.28a we deduce that α has to be 0.015 for the desired embedding strength. Using this low embedding strength, not all watermarks can be extracted. This is illustrated in 4.28c. As already figured out in the experiments with the other algorithms the outliers because of the "bad" wavelets can again be seen. In figure 4.27 we see the detector response anomaly of decreasing detection values for later embedded watermarks, which was already mentioned in section 3.6.4. With the parametrized filters improvement this anomaly disapeared and is replaced by the anticipated development. This is because by the different wavelets the embedding coefficient decimation is avoided. Also the generation of the activity image does not perfectly include

the older watermarks (one of the reasons for the inverse trend of the detection value development) with this modification.

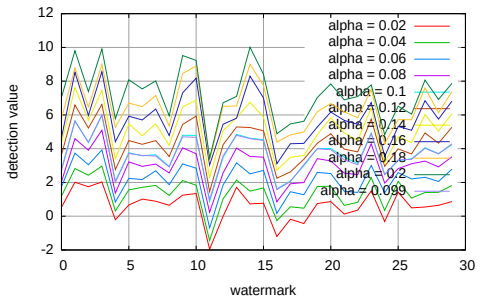
random coefficient selection

With random coefficient selection we want to embed the watermark into randomly chosen 10% of the coefficients. For this approach a few changes had to be made. The coefficients to mark can not be selected by comparing to a threshold, like in the original algorithm. The number of selected coefficients is different in the embedding and the extraction stage. First this is because two different thresholds are used for embedding (T_1) and extraction (T_2) and second the embedding process can change the number of coefficients above the threshold. This is a problem because a different number of embedding coefficients gives completely different results when selecting 10% of them by random. Also the selection of a defined number of the highest coefficients (e.g. the 10000 highest coefficients) and then from these coefficients choose the 10% random coefficients does not work, because through the embedding process the order of the coefficients changes and this again brings different results when randomly selecting coefficients from them. So first 10% of all coefficients are randomly chosen and the x highest coefficients are used for embedding. Because the authors of this algorithm suggest an embedding threshold (T_1) of 40 and this results in 2275 coefficients (for the Lena image when a 3-level decomposition with the Daubechies-6 wavelet is applied), which equal approximately 18.5%, we choose the 18.5% highest coefficients for embedding. The same procedure is applied in the extraction and in the threshold calculation stage.

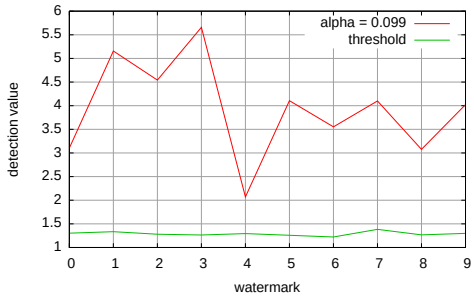
Due to the lower number of marked coefficients we can choose a higher α. $\alpha = 0.099$ gives us a embedding strength of 41.96dB. The drawback with this high embedding strength is the strong variability of the detection values. Furthermore, these variations can be amplified by the random selection of coefficients. When using this improvement approach with the modifications explained in the beginning of this section a constant number of coefficients in the smallest detail subbands is selected. In our case where we use a 512x512 image and select 10% of all coefficients in these three subbands and then the 18.5% biggest coefficients from those we get 227 coefficient to mark. The detection value now depends on the absolute value of these coefficients. Nevertheless all 10 watermarks can be extracted as illustrated in fig. 4.29c. A trend in the detection values is hard to observe due to the high variability.



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.29: PSNR and detection values when embedding multiple watermarks with the algorithm of Pla et al. and random coefficient selection improvement with different embedding strengths. watermarked image: Lena

random carriers

Since the watermark is created during the embedding process and the number of watermark elements can not be determined in advance, the carrier elements are also created on the fly. The PRNG from which the watermark elements are drawn is reseeded with every coefficient to mark (see section 2.5.1). This enables the algorithm to resynchronize with every root coefficient. To preserve this capability the PRNG for the carrier also has to be reseeded in the same way. We use the same seed creation function but with the carrier key K_C instead of the embedding key K_E.

$$seed = f(K_C, b_k, x_k, y_k)$$

b_k is the subband of the actual coefficient and x_k and y_k are the coordinates in the subband. The embedding formula is changed to

$$v'_k = v_k + \alpha_k |v_k| c_{k,i} p_{k,i}$$

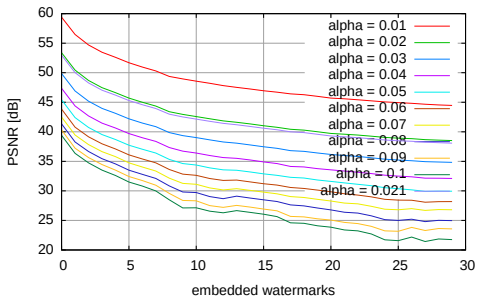
where $c_{k,i}$ are the carrier elements and $p_{k,i}$ are the watermark elements. The detection value is calculated with

$$Z = \frac{1}{M} \sum_{i=1}^{M} \hat{V}_i c_i y_i$$

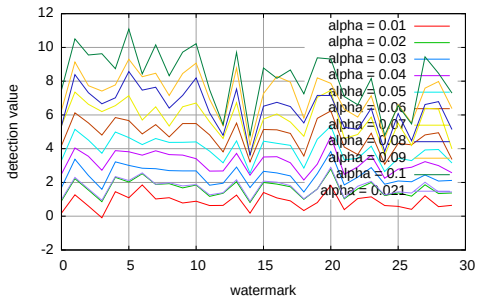
With $\alpha = 0.021$ the PSNR after the first 10 watermarks is 42.15dB. The trend of the detection values looks very similar to the unmodified algorithm. But the average detection value is significantly higher, whereas the threshold as proposed by Pla et al. is even lower. The detection value is still unsteady but the watermarks can be detected more clearly as with the original algorithm.

random wavelet packet trees

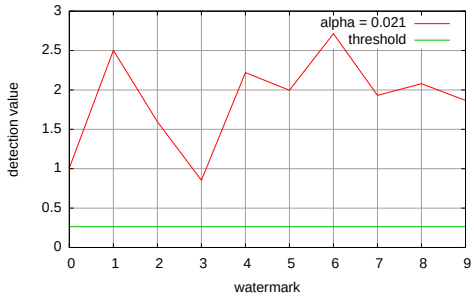
The integration of wavelet packet trees into this algorithm is a little bit tricky. Since the algorithm embeds into trees of coefficients a parent-child relationship has to be created. While this is an easy task for pyramidal decompositions, where every coefficient has 4 descendants in the next finer subband of the same orientation, it is more difficult for wavelet packets. Pla employs ideas of the embedded zerotree wavelet compression technique where the offspring of a coefficient is said to be significant if the coefficient itself is significant. The crucial attributes of the related coefficients are that they have the same orientation and belong to the same spatial location and the



(a) PSNR



(b) detection value



(c) 10 watermarks

Figure 4.30: PSNR and detection values when embedding multiple watermarks with the algorithm of Pla et al. and random carriers improvement with different embedding strengths. watermarked image: Lena

children are on a finer level as the parent. So we need to build up a tree structure on the wavelet packets which makes the same attributes apparent.

To construct the relationship tree we make use of the results of Rajpoot et al. [30, 31]. As a prerequisite for this approach the approximation image has to be on the coarsest scale. Because we have three orientations (h, v, d) we will have three trees with the corresponsding subband on the coarsest level as the root. To have a single tree we set the approximation image as the root and the roots of the h,v and d tree as its children. Then for the construction of the trees for each orientation Rajpoot gives the following rules:

1. Rule A: If a node P at a coarser scale is followed by a node C at the immediately next finer scale (like in a wavelet transform), the node P is declared as a parent of C (illustration: fig. 4.31a).

2. Rule B: If a node P is followed by four nodes C_1, C_2, C_3 and C_4 (at the same scale), P is declared to be the parent of all these four nodes (illustration: fig. 4.31b).

3. Rule C: If four nodes P_1, P_2, P_3 and P_4 at a coarser scale are followed by four nodes C_1, C_2, C_3 and C_4 at the immediately next finer scale, then node P_i is declared to be the parent of C_i (for i=1,2,3,4) (illustration: fig. 4.31c).

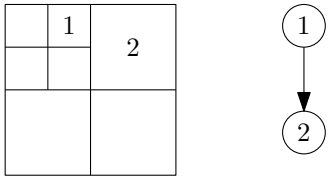
4. Rule D: If a node P is at a finer scale than four of its children, say C_1, C_2, C_3 and C_4, then P is disregarded as being the parent of all these nodes and all this bunch is moved up in the tree (illustration: fig. 4.31d).

While the first three rules are pretty straight forward, the fourth rule is a solution for the so called parenting conflict. Another, maybe simpler, solution would be to just merge the conflicting children until the conflict disappears. But we will not use this approach since it decreases the number of possible wavelet packets and therefore reduces the keyspace for the embedding keys.

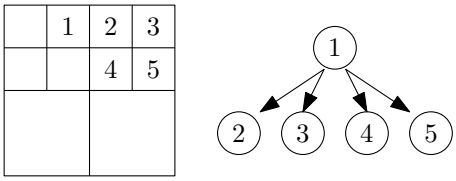
For our implementation of this approach we discard Rule C, since it is a special case of Rule A. Furthermore the four given rules do not cover the case where the offspring of a subband is two or more scales finer than the subband. The procedure for creating the tree structure used for embedding (we will call it hvd-tree) is as follows:

1. Begin with the approximation image and set the three associated detail subbands as the hvd-children.

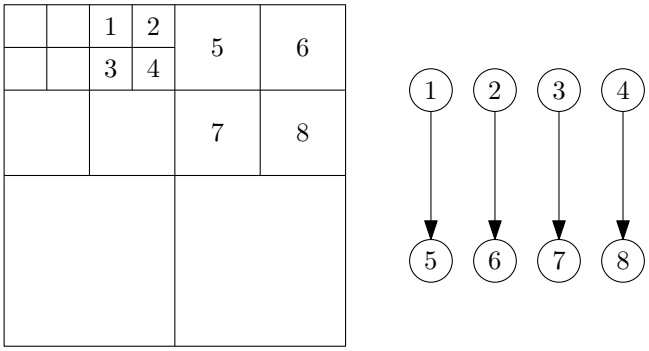
2. Build the hvd-base tree of the h, v and d orientation. This is the tree as for the pyramidal decomposition. See figure 4.32.



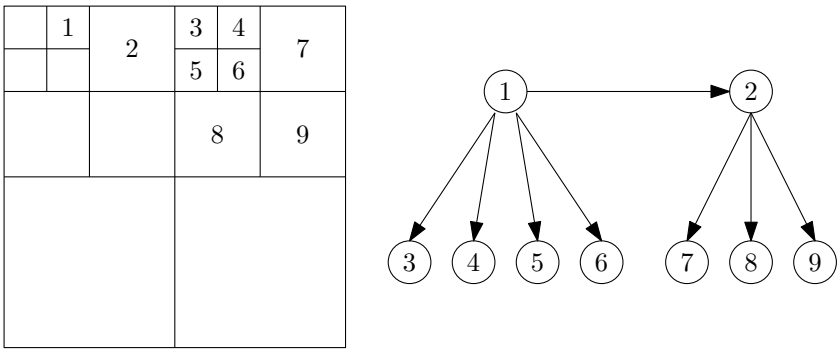
(a) Rule A



(b) Rule B



(c) Rule C



(d) Rule D

Figure 4.31: Rules for creating a hvd tree

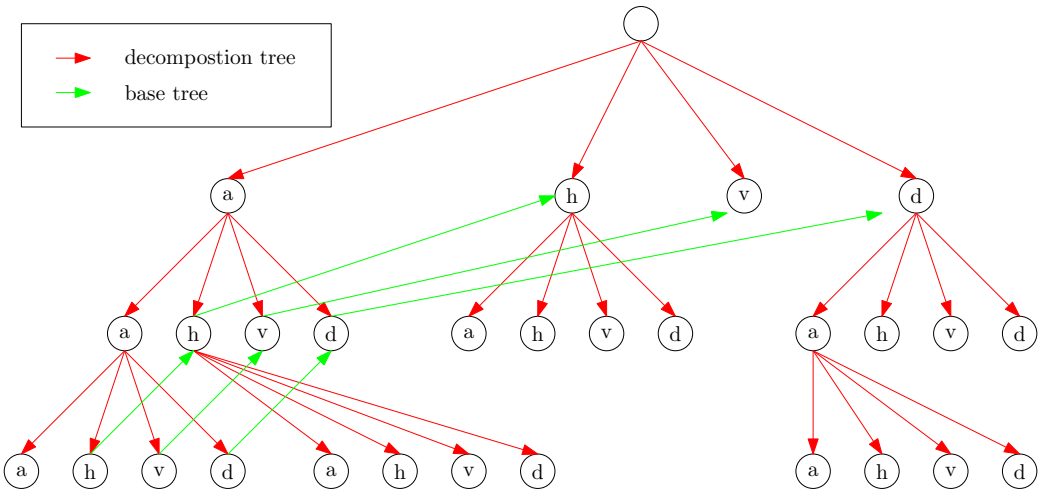


Figure 4.32: tree structure of the decomposition tree and the hvd base tree

3. For each of the three children of the approximation image (the roots for a orientation) check if the node is a decomposition-leaf.
   (a) If yes, no change in the tree structure is necessary and the hvd-child of this node is checked (there can only be one child in this stage of the tree creation).
   (b) If no, then the node is deleted from the hvd-parents child list and the decomposition-children of the node are appended to the child list of the hvd-parent. Afterwards we will check for the case that Rajpoot et al. [30, 31] do not cover in his approach. We will call this rule E. This is done by checking if the child of the actual node (this is the node which is not in the hvd-tree anymore) is also decomposed.
      i. If yes, rule C comes into effect (we use 4 times rule A, Rajpoot would use rule C) and the decomposition-child x of the hvd-child of the current node is appended to the child list of decomposition child x of the current node, where x is $\in$\{a, h, v, d\}. This case is illustrated in figure 4.33 where node 2 is the actual node to check. The dashed green line is the unlinking because node 2 is not a leaf and the children of node 2 are linked to the node 1 (parent of node 2). The outgoing links from the children of node 2 are the rule C links and rule C comes into effect, because node 3 is also not a leaf. After the relinking, rule E has to be checked for the hvd-child of the

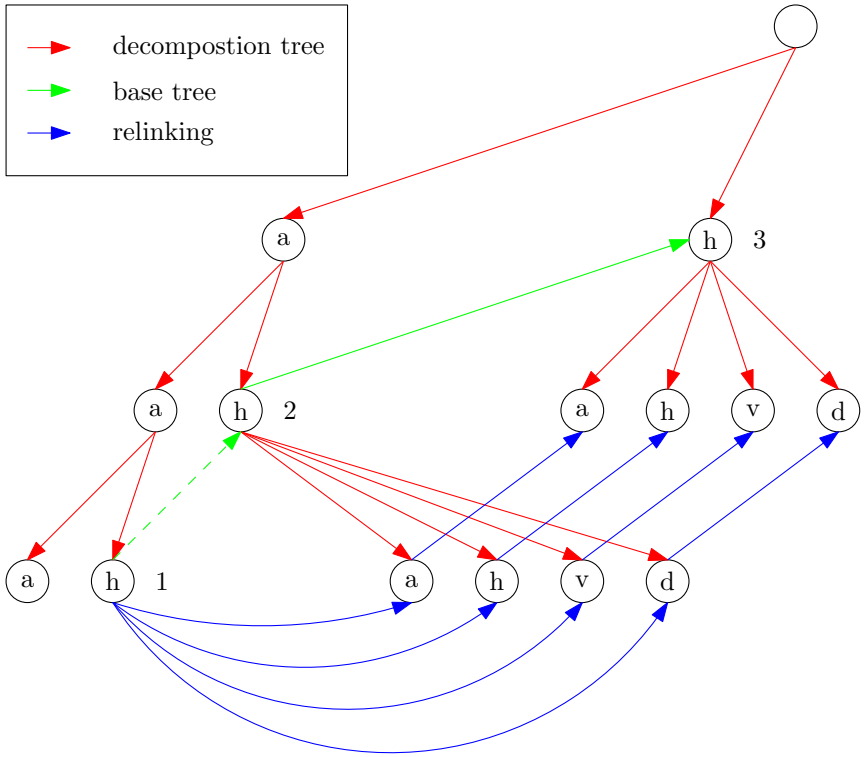


Figure 4.33: An example where rule C comes into effect during hvd tree creation

current node. In the illustration this would be node 3, which has to be checked.

ii. If no, rule E comes into effect. Then the hvd-child of the current node has to appended to the hvd-children list of the parent of the current node.

After the recursive relinking we carry out the leaf check for all the elements in the hvd-child list of the hvd-parent of the current node (this would be all children of node 1 in fig. 4.33). Actually only the relinked children have to be checked, but because there are rarely more children than the relinked ones we take into account that we maybe do some redundant work. The alternative is that we need to journalize all relinked children, what would be a bigger effort than the first alternative.

4. After all the leaf checking is done, we still have to look out for rule D occurrences. We recursively run through all nodes and check if the decomposition level of the hvd-child node is bigger than the level of

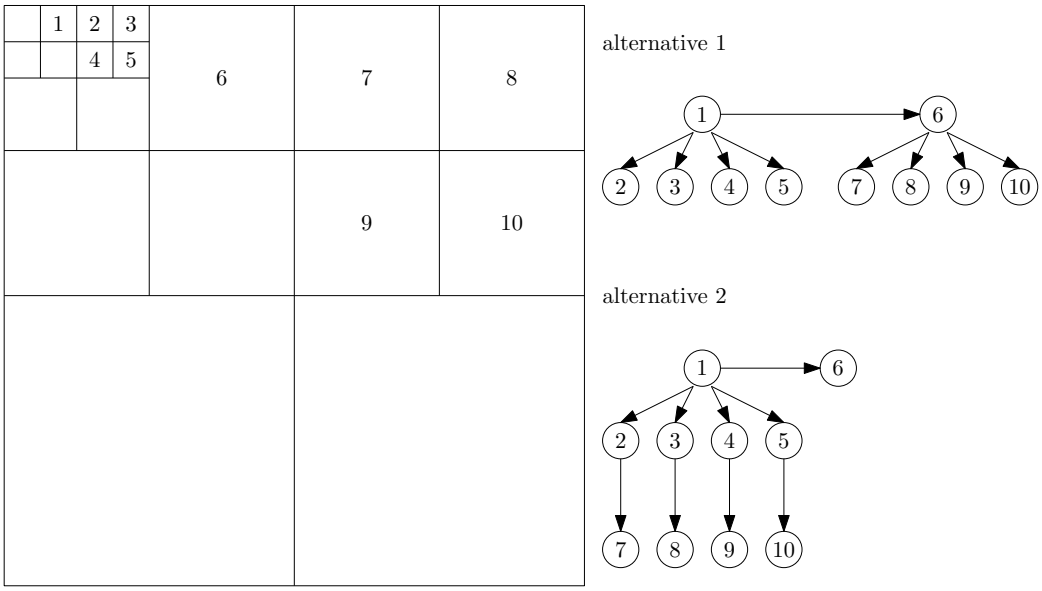
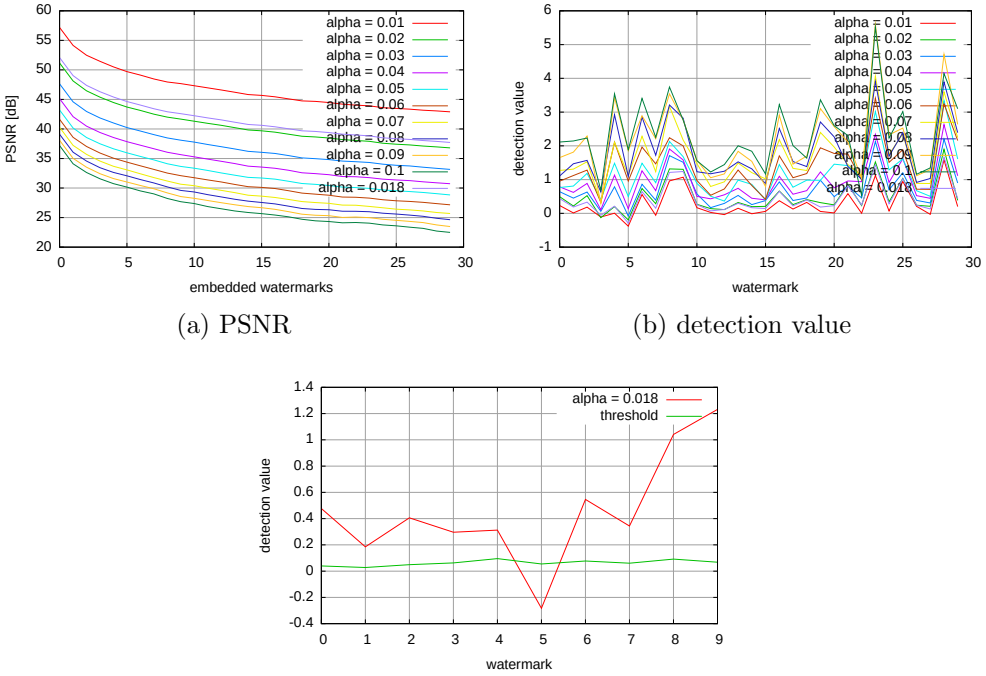


Figure 4.34: The two possible alternatives for rule E

the current node. If such an occurrence is found we climb up the tree until we find a node with equal decomposition level like the children and relink the children to this node.

There is also an alternative for rule E, in our case when the rule comes into effect, the child node and all descendants are moved up in the tree. But it could be, that a descendant is again compatible with the disregarded parent node and could again be linked to this node. We take the first approach, because in the second case the hvd tree has to be ripped up on two points. Figure 4.34 shows the two alternatives.

For the integration of the wavelet packets into the algorithm of Pla et al. we do a random wavelet packet decomposition with at most 4 level decomposition depth and we ensure, that the approximation image is on the coarsest scale. Then we create the hvd-tree with the method described above and subsequently we calculate the activity image from the wavelet packet tree. This is not very different from the original algorithm, since the activity image has to be in the same size as the approximation image is and uses the three detail subbands associated with the approximation image for calculation. Because the approximation image always has to be on the coarsest scale, the three detail subbands are on the same scale and the activity image can be calculated with the procedure as proposed by Pla et al. We embed the watermark into the 3 coarsest detail subbands and recursively



(a) PSNR

(b) detection value

(c) 10 watermarks

Figure 4.35: PSNR and detection values when embedding multiple watermarks with the algorithm of Pla et al. and random wavelet packets improvement with different embedding strengths. watermarked image: Lena

into the children of a coefficient if the coefficient is above a threshold, like in the original algorithm.

We have to embed with α=0.018 to gain an embedding PSNR of 42.24dB. Below α=0.05 not all watermarks can be detected. Additionally to the detection value fluctuations caused by the algorithm comes the unsteadyness of the detection values from the quality of the wavelet packets. Generally the detection values are lower with this approach compared to the unmodified algorithm. In our experiment with 10 embedded watermarks (fig. 4.35c) the detection values are very close to the threshold and even not all watermarks can be extracted.

comparison

In figure 4.36 we have abbreviated the improvement techniques for better readability, no improvement with “ni”, parametrized filters with “pf”, ran-

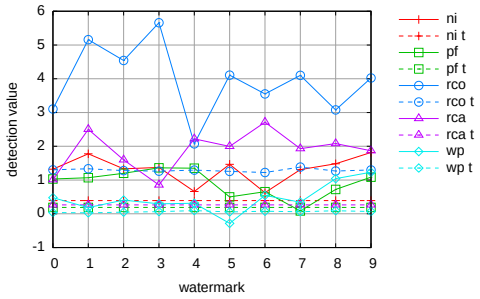


Figure 4.36: comparison of the detection values of the different improvement techniques on the algorithm of Pla et al.

dom coefficients with “rco”, random carriers with “rca” and wavelet packets with “wp”. An appended “t” denotes the according threshold. Viewing only the detection values random coefficient selection is by far the best solution. Also the random carrier modulation brings a little improvement. The other two improvement approaches achieve a worse result than the original algorithm. But again, very similar to the improvements of the algorithm of Barni et al., considering also the threshold lowers the benefit of the random coefficient selection technique because the threshold significantly rises. For all other improvements the threshold decreases compared to the unmodified algorithm. This increases the benefit of random carrier modulation. But parametrized filters and random wabelet packets still fail to improve the multiple watermarking qualities of the algorithm. We will have a closer look on the impact of the changing thresholds in the next chapter (chap. 5).

4.2.6 Wu

Like the other quantization based algorithm (by Chen et al.) we use in our work, this algorithm is not capable of detecting multiple watermarks at all. This gives much space for improvement and we will try to enable multiple watermarking with our improvements. We will implement the parametrized filters and the random coefficient selection improvement. Random carrier modulation can not be applied here, because the algorithm is quantization based. We will also not use the wavelet packets, because the algorithm needs the child subbands to be at least as big as the parents (just like the hvd trees, see section 4.2.5). If we adhere strictly to the proposed algorithm, the child subbands needs to be exactly double the size of the parent subband. Anyway, this reduces the number of possible wavelet packet decompositions to a very small number, thus the approach is not applicable.

no improvement

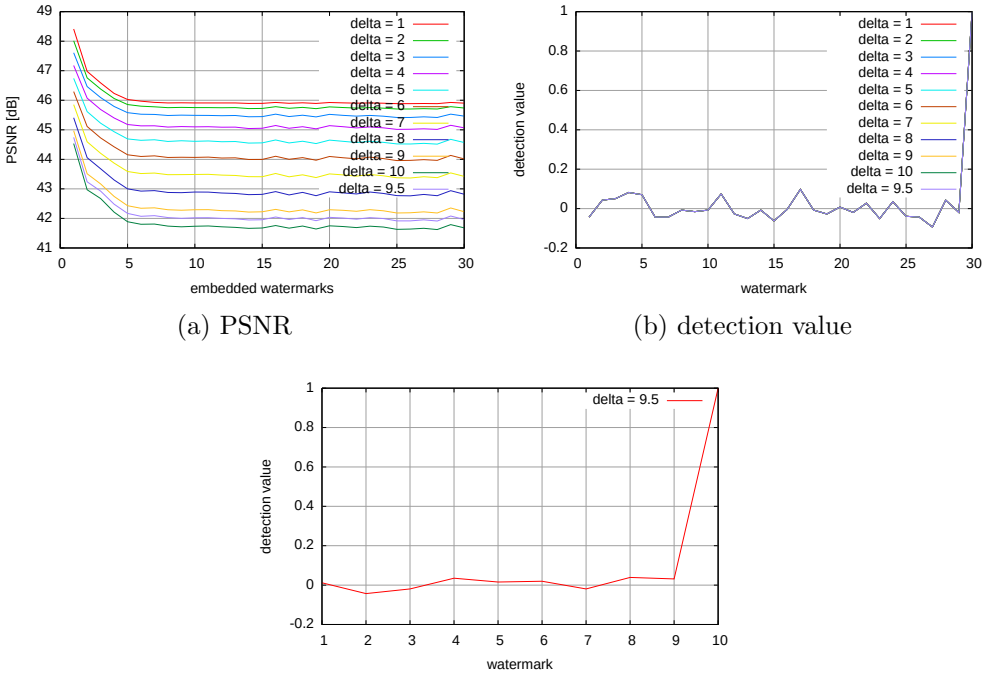
We already know, that multiple watermarking does not work with this algorithm. We carry out the experiments without improvements anyway to have data for the comparison with the improved versions.

As already mentioned in section 3.7.4 and can again be seen in figure 4.37a the PSNR drops rather quickly in the first few embedding runs and then, starting from approximately the fifth run, stays at a certain level. We choose Δ to be 9.5 and get an embedding PSNR of 40.02dB after the tenth watermark. The detection value in fig. 4.37b and 4.37c is 1 for the last embedded watermark but drops to approximately 0 for all other watermarks. For the same reasons as with the algorithm of Chen et al., this is independent of the embedding strength.

parametrized filters

For this improvement no changes have to be made to the algorithm, only the generated filter has to be passed to the wavelet transform.

Figure 4.38a shows some differences to the PSNR characteristics of the unimproved algorithm. First the PSNR does not stop to decrease when more watermarks are embedded. This is, because further watermarks do not exactly overwrite previous ones and so induce additional image distortion. Second, even with Δ=0.01 it is not possible to keep the image distortion below 42dB when embedding 10 watermarks, the attained distortion is 34.85dB in this case. In figure 4.38b we see that the average detection value is definitely higher, but very similar to the improvement in combination with the other algorithms some detection value glitches make the improvement not reliable.

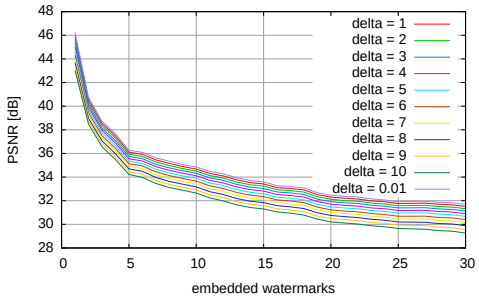


(a) PSNR

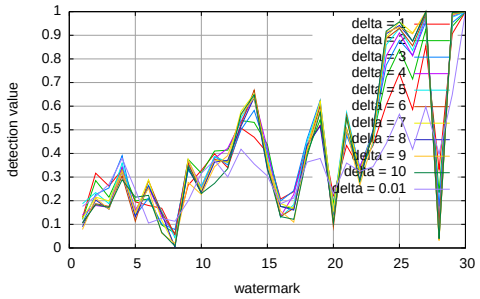
(b) detection value

(c) 10 watermarks

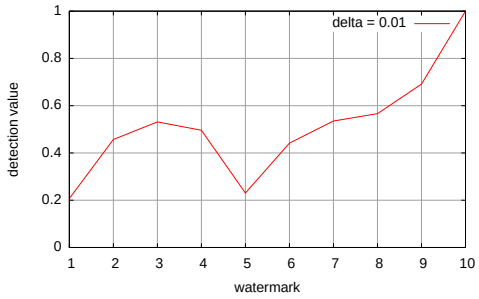
Figure 4.37: PSNR and detection values when embedding multiple watermarks with the algorithm of Wu et al. with different embedding strengths. watermarked image: Lena



(a) PSNR

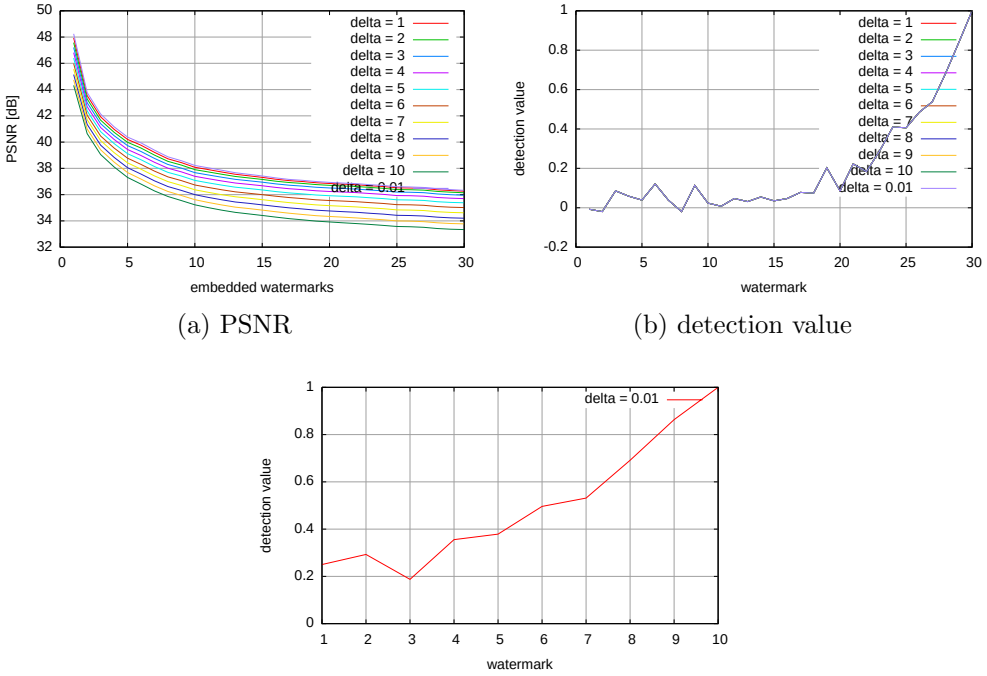


(b) detection value



(c) 10 watermarks

Figure 4.38: PSNR and detection values when embedding multiple watermarks with the algorithm of Wu et al. and parametrized filters improvement with different embedding strengths. watermarked image: Lena



(a) PSNR

(b) detection value

(c) 10 watermarks

Figure 4.39: PSNR and detection values when embedding multiple watermarks with the algorithm of Wu et al. and random coefficient selection improvement with different embedding strengths. watermarked image: Lena

Another difference to the original algorithm is that the embedding strength has a little more impact on the detection values. In section 3.7.3 we set the threshold to 0.2. With this threshold all watermarks can be extracted in figure 4.38c, although two watermarks are very close to the detection border and we have to keep in mind, that the image is noticeably damaged because of the inability of the algorithm to embed the watermark softer.

random coefficient selection

For this improvement technique we use the following procedure. First we create the supertrees like in the unmodified version. Then we randomly select as many supertrees as needed to embed the watermark. In our experiments with a 512x512 image and a 3-level wavelet decomposition we have 3072 supertrees and use 512 of them to embed the watermark.

Like with the parametrized filters improvement the image quality is worse

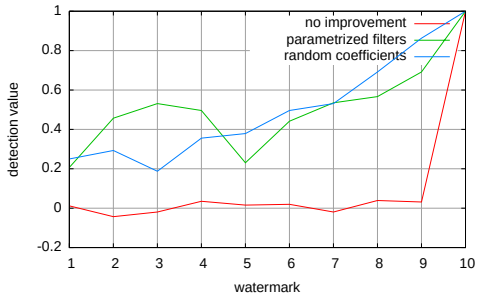


Figure 4.40: comparison of the detection values of the different improvement techniques on the algorithm of Wu et al.

then 42dB after 10 embedded watermarks (see fig. 4.39a). With Δ=0.01 we achieve an embedding PSNR of 38.22dB after the tenth watermark. The PSNR is steadily decreasing for the 30 embedding runs, but not as much as with the parametrized filters. Because the parametrized filters can produce a large number of different filters and therefore a large number of embedding spaces, the watermark induces more distortion than with the random coefficient selection. With the latter only 3072 supertrees are used for embedding. The detection values in figure 4.39b show a clear improvement to the original algorithm. Random coefficient selection also gives a more predictable result than parametrized filters. With this approach the embedding strength has no impact on the detection values. This is because when a collision occurs by selecting the same supertree twice the previous watermark gets fully overwritten and this is independent of the embedding strength. In the experiment with 10 embedded watermarks (fig. 4.39c) the last seven watermarks can detected and the watermark 3 is slightly below the threshold of 0.2.

comparison

The direct comparison of the original algorithm with the two improvement techniques brings out that both approaches can significantly improve the multiple watermarking capabilities. While the random coefficient selection algorithm produces the more steady results it suffers from the small number of supertrees to select from. The problem with the parametrized filters are the “poor” wavelets, these wavelets are not “different” enough and so some watermarks overwrite their predecessors, this is for example the watermark 29 which overwrites the watermark 28 in figure 4.38b. If we assume that the different embedders do not know each other and so do not know the embedding parameters there is no way to check if the embedding of a watermark

overwrites a predecessor. The improvement technique has good potential, but also a severe drawback with the "poor" wavelets.

We want to repeat that the practical applicability of multiple watermarking with this algorithm is hardly given, because of the inability to maintain a adequate image quality for a reasonable number of watermarks.

Chapter 5

New assessment criterion

As we have seen in the last chapter (e.g. in section 4.2.1 or section 4.2.5), there were some ambiguities concerning the benefit of the different improvement approaches in face of the varying threshold. In this chapter we use an assessment criterion which takes account for this fact. In addition the new criterion provides the possibility to define a threshold for every watermarking algorithm independently of their internal structure.

5.1 How to evaluate the performance of a watermark detector

In the previous chapters and also in most literature detection performance of a watermarking algorithm (most times after some attack) is assessed by its detector response. Unfortunately there is no common method for calculating the detector response. In most watermarking systems the correlation coefficient or the normalized correlation or some form of linear correlation is used, see [21]. The value of the linear correlation is strongly depending on the image size and the image content and therefore the other two methods are to prefer. The other two measures are in the range of -1 and 1.

But also compareing one of these two measures from different watermarking algorithms or different improvements is not unproblematic, since a higher detection value does not automatically mean that the watermark is "better" detected. On the one hand if this is done with a single watermark and/or a single image it could give misleading results because it is possible that a specific watermark and/or image is well-suited or ill-suited for a watermarking algorithm. Also the comparison of the mean of some detection runs with different watermarks/images is not absolutely suitable.

Lets have a look on what the aim of a watermark detector is. Basically the

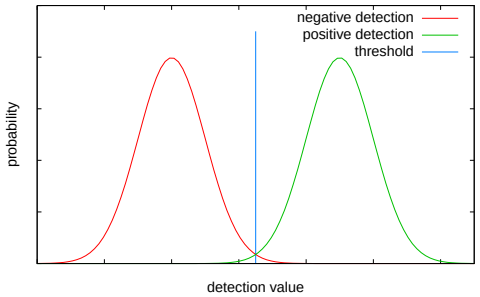


Figure 5.1: Distribution of detection values for negative detections (h0) and positive detections (h1)

detector should make no or least possible false detections. False detections can be one of two cases:

- false positive: Here the watermark is detected by the detector whereas no watermark or a watermark different from the embedded one is present.
- false negative: In this case the detector can not detect the watermark allthough the watermark is embedded in the image.

In other words the detector should safely distinguish between the embedded watermark and other watermarks or no embedded watermark. To do this, a decision threshold has to be defined. The threshold has to be chosen to give the least false detection rate. Piva et al.[28] state, that by invoking the central limit theorem the detection values are normally distributed for the case where the watermark is embedded in the image and the case where the watermark is absent or a false watermark is embedded. Piva et al. denote the fact that the watermark is embedded in the image as hypothesis 1 and an absent watermark as hypothesis 0. Therefore the distribution of the positive detection values is called h1 and the distribution of the negative detection values is called h0. Figure 5.1 shows the distribution of the detection values and the threshold giving the smallest error rate.

Under this point of view it should be clear, that by only comparing the mean of some detection runs it can not be said which detector is better since the variance of the detection values is disregarded and the threshold has not been taken into account. Figure 5.2 shows an example where one positive-detection distribution (d2) has a higher mean value than an other positive-detection distribution (d1) but a worse error rate due to the higher variance of the distribution. This shows that it should not be the main target

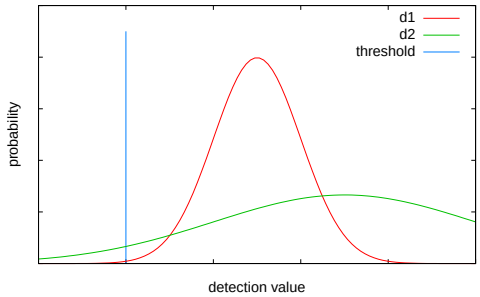


Figure 5.2: Example distributions where one distribution has a higher mean value but a worse error rate

to have the detection values very high above the threshold but to have little detection values below the threshold. For sake of simplicity figure 5.2 shows one threshold for both distributions. In practice a separate threshold has to be calculated for each watermarking algorithm.

This leads to the conclusion, that to accurately assess a watermark detector one should not examine the detection value but the error rate.

Piva further states that it is the better choice to set the threshold closer to the negative detections instead of the middle of the two cases, because attacks do not affect the distribution of the detection values for negative detections but the mean of positive-detection values will most likely be lowered while the variance is increased. This does not give the best error rates under normal circumstances but more robustness under attacks. In [28] the threshold is chosen to give a fixed false positive rate and at the same time maximize the robustness under attacks.

To compare two watermarking algorithms a threshold has to be chosen e.g. similar to [28] and the overall error probability has to be calculated or if the false positive rate is fixed the false negative rate can be used to compare the watermarking algorithms.

5.2 experimental results

In this section we repeat the experiments of the last chapter with some selected algorithms with the new assessment criterion. We have chosen not to test the algorithms of Furon and Bas and the algorithm of Chen et al. because we could achieve no or only little improvement and in the case of the Broken Arrows algorithm the calculation of the threshold is independent of the image content and the size of the image or the length of the watermark.

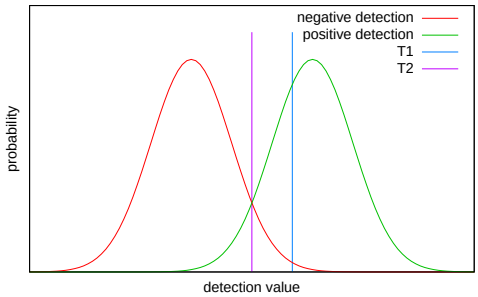


Figure 5.3: Illustration of the two thresholds used in the experiments

The algorithm of Cao et al. is also not in the test because of the huge computational effort. So we are going to test the algorithms of Barni et. al, Pla et al. and Wu et al.

5.2.1 experiment setup

The detection value for all experiments is the mormalized correlation between the embedded and the extracted watermark. This is

$$nc = \frac{\sum_i a_i \, b_i}{\sum_i a_i^2 \, \sum_i b_i^2}.$$

The threshold is calculated to provide a given false positive rate (10^{-8}). If the overall error rate

$$err = \frac{fpr + fnr}{2}$$

is higher than 10^{-8} (this is the case when the false negative rate is higher than the false positive rate), then the threshold is set to have equal fpr (false positive rate) and fnr (false negative rate). This is usually also the threshold with the least possible error rate. Figure 5.3 is an illustration of this case. T1 is the threshold where the false positive rate is 10^{-8}, but the positive detection values are too low and the threshold is far inside the h1 curve giving a very poor false negative rate. If that happens, the threshold is shifted to the point indicated by T2, this is the point with equal fpr and fnr.

The comparison of the different improvement techniques (fig. 5.5b, 5.7b and 5.9b) is done by averaging the results over over 4 different images (Lena, Baboon, Peppers, Boat) and 1000 runs each. In each run, different seeds

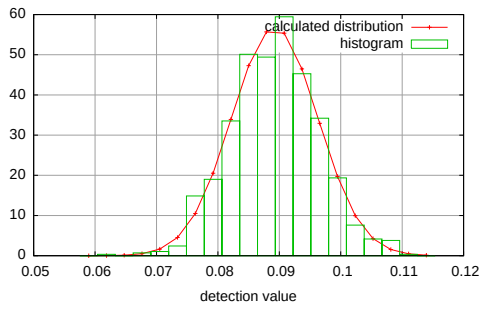
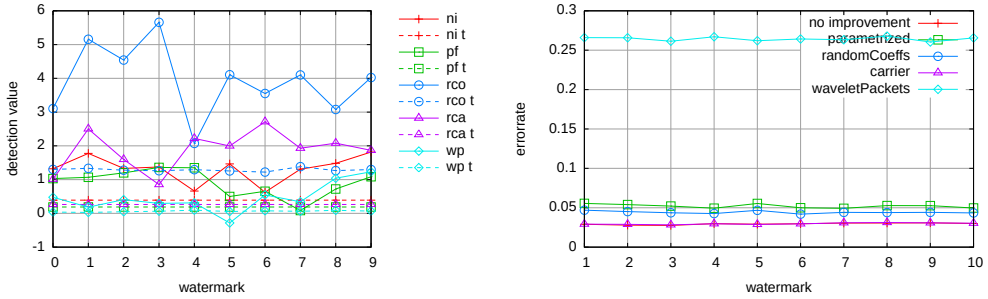


Figure 5.4: Comparison of the histogram of the data values and the calculated distribution

for the watermark and for the improvement technique (e.g. seed for creating parametrized wavelets) are used. The results for the specific improvement approaches (fig. 5.6b-5.6e, 5.8b-5.8e, 5.10b-5.10c and 5.11) consist also of 1000 runs and a single image (Lena). We assume that the detection values are normally distributed. Visual investigation of the detection values (see figure 5.4) and the chi-square test confirms this assumption. Figure 5.4 shows the h1 distribution of a single watermark embedded with the algorithm of Pla et al. (figure 5.6a). The histogram boxes are normalized to have an area of 1 like the pdf of the distribution. For the chi-square test we use 20 cells so we get 17 degrees of freedom because we use 2 parameters to estimate the distribution. For a significance level of 0.01 we get a bound of 33.41. The example from above (fig. 5.4) has a chi-squared value of 27.1 which is below the bound of 33.41, so we can say the values are normally distributed. We calculate the mean and the standard deviation of the detection values and use these values as the parameters for the distribution. With the found distribution we can now calculate the threshold and the error rate. As described above, we first calculate the threshold with a given false positive rate. Therefor we calculate the point where the cumulative distribution function (cdf) of h0 is $1 - fpr$. If the false negative rate exceeds the false positive rate at this point, then we calculate a new threshold. The new threshold is found by calculating the point where $fpr = fnr$. Where the false positive rate is calculated by $fpr = 1 - cdf(h0)$ and the false negative rate is calculated by $fnr = cdf(h1)$.



(a) Pla detection values

(b) Pla error rates

Figure 5.5: Pla results

5.2.2 Pla

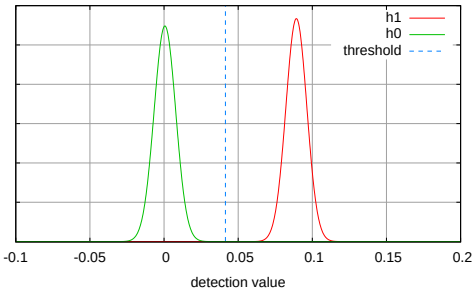
Figure 5.5 shows us the comparison between the results when only the detection value is considered and the new experiments where the improvement approaches are assessed with the error rate. The abbrevations in figure 5.5a are, “ni” for no improvement, “pf” for parametrized filters, “rco” for random coefficients, “rca” for random carriers and “wp” for wavelet packets. An appended “t” denotes the according threshold. In figure 5.5a it seems that random coefficient selection is the best improvement technique followed by the random carrier modulation and the other two, parametrized filters and random wavelet packets, can not improve the algorithm. Furthermore the detection values are very instable and no clear trend can be observed. In figure 5.5b the results look very different. First the detection values are almost constant for all watermarks and second the ranking of the algorithms is also different. Here no improvement approach can achieve better error rates. Random carriers have the same error rate as the unimproved version, the error rate of random coefficient selection and parametrized filters is a little bit higher and wavelet packets have a much worse error rate. This confirms the assumption that only looking at the detection value is not sufficient to rate a watermarking algorithm. Altogether the error rate is always pretty high when embedding 10 watermarks and heading for a PSNR of 42dB after the 10th watermark.

We will now have a closer look on the case without improvement and the case with random coefficient selection. This is the one with the best results in the earlier experiments, but worse results when looking at the error rate. The first figure (fig 5.6a) shows the distribution of detection values when only one watermark is embedded. The threshold is set to give a false positive rate

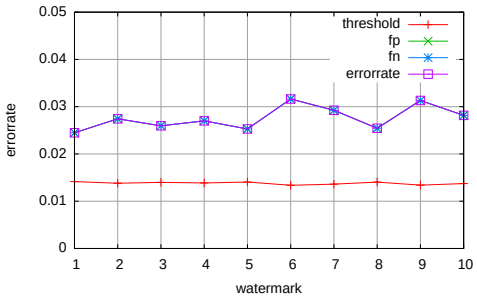
of 10^{-8} ($t = 0.041$) and the false negative rate is 0. To be exact the false negative rate never reaches 0, but the computation with the precision of python gives us 0.

In figures 5.6b and 5.6c we see the results when 10 watermarks are embedded with the unimproved algorithm. The mean positive detection value is very low, so that we can not reach an overall error rate of 10^{-8}, so we choose to set the threshold on the intersection between the h0 and the h1.10 (positive detection values for the 10th watermark) line. This gives a threshold of approximately 0.014. The error rate is then approximately 0.028. Because we choose to set the threshold to the point where fpr and fnr is equal and thus the error rate is also equal to the fpr and fnr, only one line in figure 5.6b is visible besides the threshold. In figures 5.6d and 5.6e, the mean positive detection value is higher than with the unimproved algorithm, but also the standard deviation for the false detection values and the positive detection values. This makes the threshold rise and the error rate also increases. We got a threshold of approximately 0.04 and an error rate of 0.042.

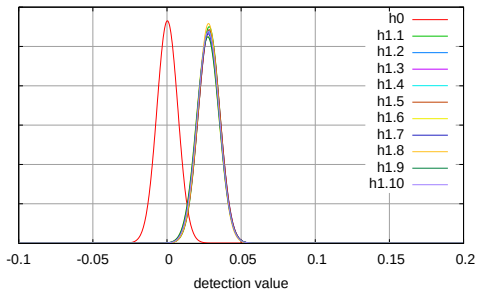
Both experiments, the one without improvement and the one with random coefficient selection show that the error rate of the single watermarks is almost independent of the order they are embedded. The first watermark and the last watermark have almost the same error rate.



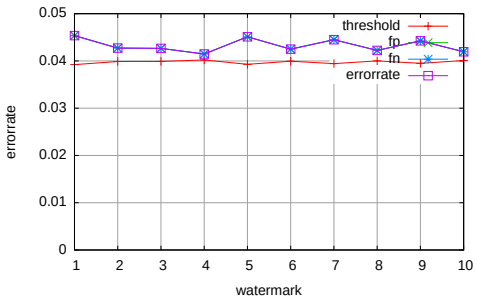
(a) Pla single watermark



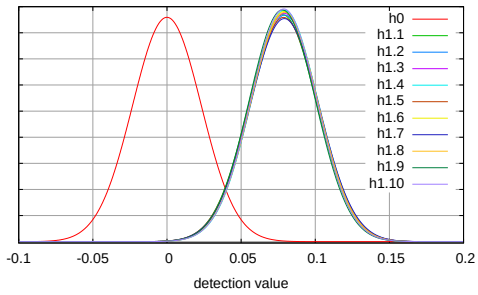
(b) Pla no improvement err



(c) Pla no improvement h0h1

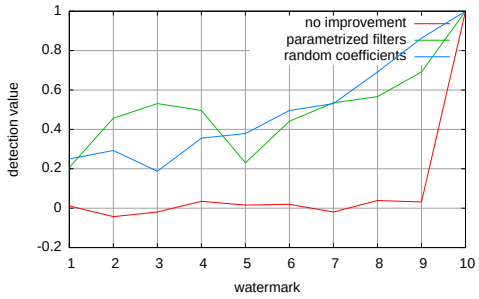


(d) Pla random coefficients err

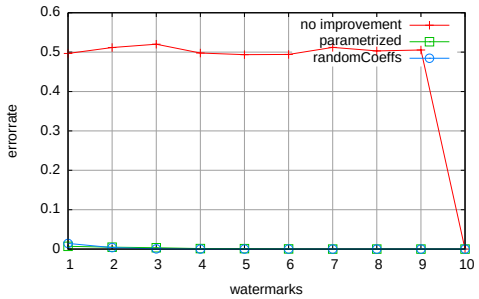


(e) Pla random coefficients h0h1

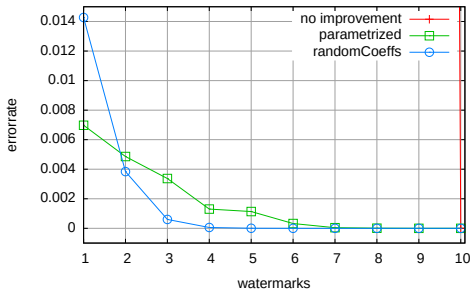
Figure 5.6: Pla results



(a) Wu detection values



(b) Wu error rates



(c) Wu error rates zoom

Figure 5.7: Wu results

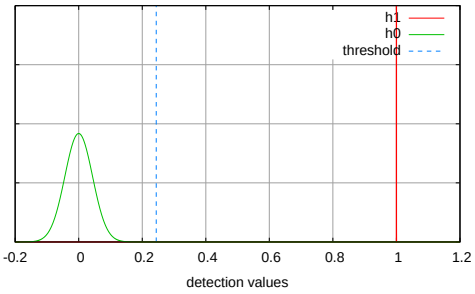
5.2.3 Wu

For the algorithm of Wu et al., the results of the experiments where only the detection values are regarded (figure 5.7a) showed that both implemented improvement techniques have much better values than the unimproved version. This can be confirmed with the error rate experiments in figure 5.7b and 5.7c. The detection value of the unimproved version is 0 for all but the last watermark and the error rate is 0.5 in these cases. The detection values of the two improved versions of the algorithm are higher for all watermarks, this is reflected in a lower error rate in figures 5.7b and 5.7c.

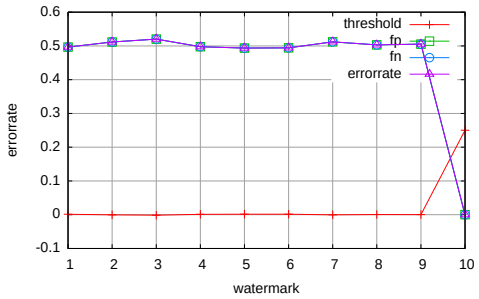
We go into details for the unimproved version and the random coefficient selection technique, like for the algorithm of Pla et al. Figure 5.8a shows the distribution of the detection values for a single embedded watermark. Like expected for quantization based algorithms, the detection values of the false watermarks is around zero and the detection value for the correct watermark is exactly 1. The threshold is set to provide a false positive rate of 10^{-8} and the false negative rate is 0.

The results of the unimproved algorithm in figure 5.8b and 5.8c also do

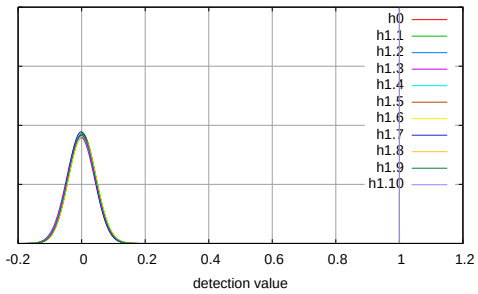
not give a surprise. The last embedded watermark has a detection value of 1 and an error rate of 0. All other positive detection values have the same distribution as the false detection values and an error rate of 0.5. More interesting are the results in figure 5.8d and 5.8e. In figure 5.8e we can see that the latest embedded watermark has the highest mean detection value and this value is constantly decreasing with every further watermark. Figure 5.8d shows, that for the last 5 watermarks the threshold is set to give a false positive rate of 10^{-8} and then the error rate can not be kept up, so the threshold is lowered. The error rate is rather low for all watermarks.



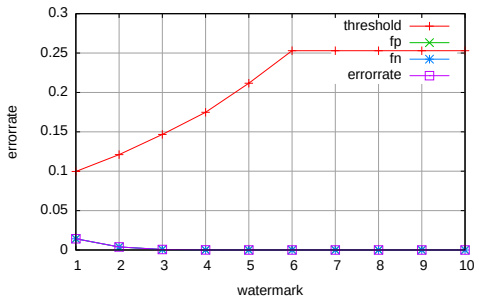
(a) Wu single watermark



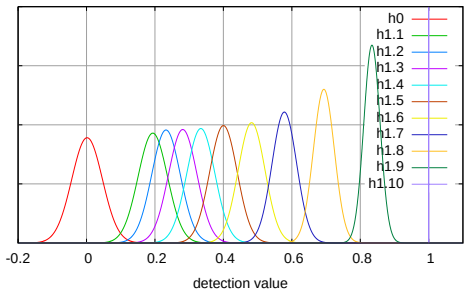
(b) Wu no improvement err



(c) Wu no improvement h0h1

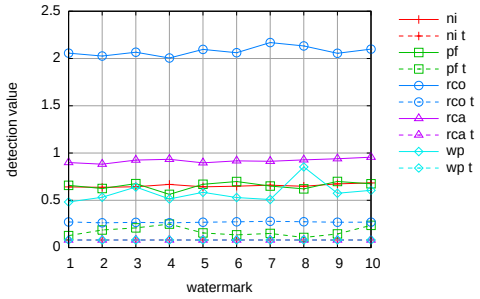


(d) Wu random coefficients err

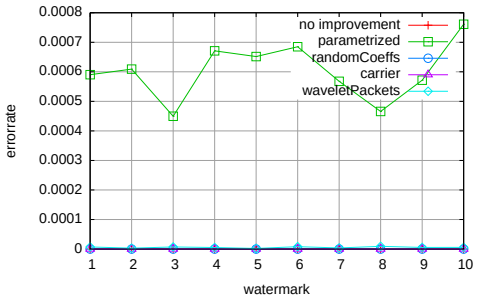


(e) Wu random coefficients h0h1

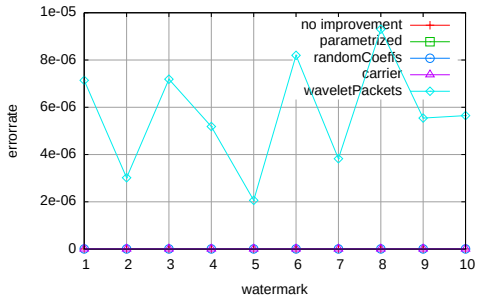
Figure 5.8: Wu results



(a) Barni detection values



(b) Barni error rates



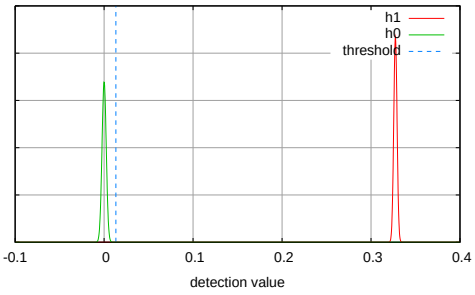
(c) Barni error rates zoom

Figure 5.9: Barni results

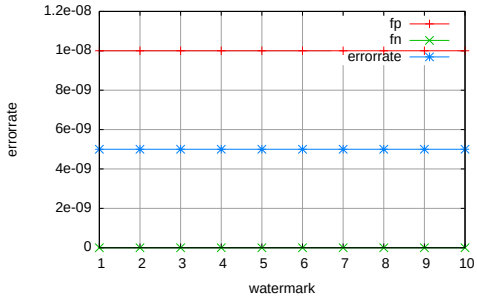
5.2.4 Barni

In figure 5.9 the comparison between the results of the last chapter and the results of the experiments on the error rate is shown. Figure 5.9a shows the random coefficient selection approach to be the best, followed by the random carrier modulation approach which could also achieve a little improvement. For better readability we have abbreviated the improvement techniques in this figure, no improvement with “ni”, parametrized filters with “pf”, random coefficients with “rco”, random carriers with “rca” and wavelet packets with “wp”. An appended “t” denotes the according threshold. On the other hand, the results in figures 5.9b and 5.9c show, that no approach did improve the detection value, because already the unimproved version had the best possible error rate for all 10 watermarks. Like in the results of the last chapter, random wavelet packets and parametrized filters are the worst approaches. The experiments on the error rate even show, that the parametrized filters are much worse than the unimproved algorithm and in figure 5.9c we can see that random wavelet packets bring a little degradation also.

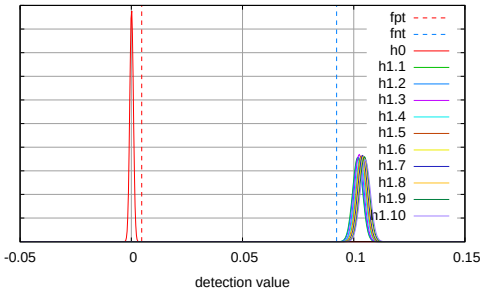
To have the possibility to evaluate different improvement approaches



(a) Barni single watermark

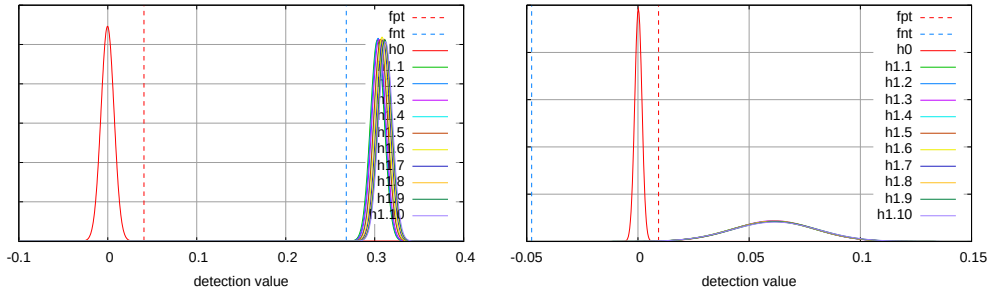


(b) Barni no improvement err



(c) Barni no improvement h0h1

Figure 5.10: Barni results 1



(a) Barni random coefficients h0h1

(b) Barni parametrized filters h0h1

(c) Barni carrier h0h1

(d) Barni wavelet packets h0h1

Figure 5.11: Barni results 2

improvement	distance
random coefficients	0.227
no improvement	0.088
random carriers	0.085
wavelet packets	0.006
parametrized filters	-0.057

Table 5.1: threshold distances for the different improvement approaches

which achieve 0 false negative rate with a given false positive rate we calculate a second threshold with a given false negative rate (10^{-8}) and calculate the distance between the two thresholds. Both thresholds are shown in figures 5.10c-5.11d. The false positive threshold (fpt) is the one which provides the given false positive rate and the false negative threshold (fnt) provides the given false negative rate. Table 5.1 show the distances of the thresholds. Detection values and threshold are calculated for the Lena image, like figures 5.10c-5.11d. Here only the parametrized filters approach has a negative distance, which means that the error rate is grater than 10^{-8}. The threshold distance of the wavelet packets approach is positive, unlike in figures 5.9b and 5.9c where the desired error rate can not be achieved. For figures 5.9b and 5.9c all 4 images are taken into consideration. We can see that only the random coefficient selection technique technique achieves better results than the unimproved algorithm. The random carrier modulation is slightly inferior to the original algorithm, random wavelet packets and parametrized filters are much worse.

5.3 Conclusion 1

Our first concerns about the suitability of the error rate as quality measure disapeared. The effect, that because of the very low standard deviation the false negative rate is always 0 when the false positive rate is fixed only exists when a single watermark is embedded. In case of multiple embedding the differences between the watermarks is clearly observable (see figure 5.8e) and also the error rate is not jumping from 0 to 0.5 (minimum to maximum) but has a slope from 0 to some higher value (see figure 5.7c). For the case of 0 false negative rate with fixed false positive rate (e.g. figure 5.10b) we use the difference between the false negative threshold and the false positive threshold as a quality measure. This makes the error rate to a practical quality measure.

Error rates achieve in fact different results as the experiments only observing the detection values and in our opinion also the "more correct" ones. Additionally it is now possible to define a threshold, either for a fixed false positive rate or for the optimal error rate also for the algorithms which gave no threshold calculation method in their paper. Due to the usage of a single value for all algorithms it is possible to compare different algorithms. This was problematic when only the correlation was used because different correlation measures (i.e. linear correlation, normalized correlation, correlation coefficient) were used and also for the same reason as with the different improvement techniques (detection value variance). Because the detection value does not always reflect the quality of the detection (this is clearly shown in section 5.2.2 and figure 5.5 the error rate is to prefer over the detection value. Furthermore, providing the possibility of setting a threshold in different algorithms with the same properties (e.g. same false positive rate) and in thereby making different algorithms or in our case different improvement techniques comparable makes the error rate the even better measure.

Disadvantage of this benchmarking approach is the high computational cost. If we take the Pla algorithm as an example it takes about 42.5 hours to find the correct embedding strength and about 103 hours to find the h0 and h1 distribution parameters for the different improvement techniques on a Xeon 2.33GHz. The implementation of the algorithm of Pla et al. is the second fastest of all tested algorithms. The algorithm of Cao has the slowest implementation and is approximately 30 times slower than the algorithm of Pla. This would be a total computation time of approximately 4365 hours (half a year).

5.4 Conclusion 2

In this chapter we tested 3 algorithms with the error rate as the assessment criterion. It showed that, unlike in the last chapter, the algorithm of Pla et al. can not be improved by our techniques. The algorithm of Wu et al., as the only quantization based algorithm in the test, could greatly be improved. In its base version the algorithm is not capable of multiple watermarking. With the parametrized filters improvement the error rate is much better, but only the last watermark can be extracted with the desired overall error rate of 10^{-8}. With the other improvement approach, random coefficient selection, the last 5 watermarks are extractable with the an error rate of less than 10^{-8}. For the algorithm of Barni et al. we had to employ a further quality measure which can also be deduced from the distribution of the false and the positive detection values. This is the distance of two threshold where one provides a

given false positive rate and the other a given false negative rate. This was necessary because already the unimproved algorithm had excellent multiple watermarking qualities. The outcome was, that the algorithm can only be improved by the random coefficient selection approach.

Chapter 6

Summary

Target of this work was to evaluate the possibility to embed multiple watermarks in a single image with the use of blind, robust watermaking algorithms. The assumed scenario was that the watermarks are successively embedded by different embedders (e.g. for distribution chain tracking) which do not know each other, i.e. can not exchange information which each other.

We have first chosen some algorithms and tested their basic performance in terms of robustness and multiple watermarking. For the choice of the algorithms we took care, that they are fairly different to see the impact of the improvement approaches on many different watermarking techniques. The watermarking techniques included additive and quantization based algorithms, embedding in trees of coefficients, embedding in the RDWT domain and different visual masking techniques. For the robustness tests we employed jpeg and jpeg200 compression, geometrical transforms and different image manipulation operations.

The main part was to test the effectiveness of different improvement approaches. The basic problem with multiple watermark is that the watermarks interfere with each other, this leads to mutual erasure. The improvement approaches attempt to mitigate the interferences. We have implemented 4 different improvements: parametrized filters decomposition, random coefficient selection, random carrier modulation and random wavelet packet structure decomposition. We carried out the tests by embedding a number of watermarks and then checking the detection values of the different watermarks. The result was that random coefficient selection seems to be the best choice for improvement. Wavelet packets are not an adequate way of improvement for any algorithm. This is either because in spite of the different wavelet packet structures the watermark is embedded in the same locations because of the embedding technique or the wavelet packets are not different enough. Random carrier modulation and parametrized filters bring small improve-

ments with some algorithms. We could notice, that for the algorithms which gave a threshold calculation algorithm, the threshold is not constant over the different improvement approaches. So we developed a different assessment criterion which takes this fact into account.

The problem when evaluating the detection value or the mean of some detection values is that these values can vary more or less. So we estimated the distribution of the detection value for false detections and the detection values for correct detections to be normally distributed. From these normal distributions we can calculate a threshold for a given false positive and a given false negative rate. For the evaluation of the detection performance we use the overall error rate or in case of very low error rates we use the distance of the two thresholds as a measure. Applying this assessment criterion we reevaluated selected algorithms. The outcome was, that in fact the evaluation with only the detection values is not very well suited. The result of the distribution experiments was fairly different to the results with the detection value of some algorithms. E.g. random carrier modulation turned out to be ineffective for all algorithms and also the benefit of the random coefficient selection was lowered for the algorithm of Pla et al. Because the main interest in watermarking algorithms is the correctness of the algorithm, i.e. a small error rate and not a high detection value the assessment with the error rate is the better choice. Additionally the consideration of the distribution of the detection values allows us to define a threshold for every algorithm. For this threshold a false positive or false negative rate can be freely chosen and thus allows the comparison of different algorithms or in our case different improvement approaches.

Finally it can be said that some algorithms can be used for multiple embedding without improvement. But none of the quantization based watermarking algorithms are suited for multiple watermarking. Random coefficient selection turned out to be the best improvement approach of all in our test. Almost all algorithms could be improved, sometimes even significantly. Parametrized filters suffer from the ill-suited wavelets and the problem of "similar" wavelets. Random wavelet packet structures are sometimes difficult or even impossible to integrate in the algorithm and can not achieve any improvement in most cases. Also random carrier modulation can not always be applied and can not provide the desired results. Especially quantization based algorithms, which are usually completely unsuitable for multiple watermarking can be improved to a considerable extent by some approaches.

6.1 Future work

Possible future work can include the research for an algorithm to generate wavelet packet decomposition structures which have a certain distance to each other even if the other structures are said to be unknown.

Because the parametrized filters improvement technique has a similar problem regarding the similarity of the generated wavelets a topic of future research can be the developement of a distance metric regarding the multiple watermarking results for the wavelets and based on that metric a selection rule for the parameters.

A drawback of the error rate assessment criterion is the huge computational cost. Especially when different images are used and because of the many runs which are needed to get the parameters for embedding and for the distributions of the detection values. Investigations on the number of runs needed to get the correct parameters can mitigate this problem.

Appendix A

Software tools

This chapter lists the software tools I used during my master thesis.

- Python: most of the software I developed for my master thesis is written in python - http://www.python.org/
  - NumPy (http://numpy.scipy.org/): package for array manipulation
  - SciPy (http://www.scipy.org/): numerical calculations (and lots of other stuff I did not use)
  - PyWavelets (http://wavelets.scipy.org/): a python package for wavelet transform
  - Psyco (http://psyco.sourceforge.net/): specializing compiler
  - Phyton Imaging Library (PIL) (http://www.pythonware.com/products/pil/): reads, writes and modifies images
  - Epydoc (http://epydoc.sourceforge.net/): Automatic API documentation generator for Python
- Eclipse (http://www.eclipse.org/): Integrated developement environment
  - Pydev (http://pydev.sourceforge.net/): Eclipse plugin for Python
- Image Magick (http://www.imagemagick.org/): command line image manipulation tool
- Gnuplot (http://www.gnuplot.info/): command line data plotting utility

- Ipe (http://tclab.kaist.ac.kr/ipe/): vector graphics editor with the feature, that text can be entered as TeX code
- TeX Live (http://tug.org/texlive/): a TeX distribution
- Kile (http://kile.sourceforge.net/): TeX editor for KDE
- Gentoo Linux (http://www.gentoo.org/): Linux distribution
- and many more open source software tools

Appendix B

Pywmtk

Pywmtk (PYthon WaterMarking ToolKit) is a collection of watermarking algorithms and useful tools around watermarking which I developed during my master thesis. I used the toolkit as a Python package and so the whole functionality is only available through python. Although I added a command line interface from which the implemented watermarking algorithms can be invoked. The prerequisites (section B.1) applies to both, usage instructions for the Python interface can be found in section B.2 and the instructions for the command line interface are in section B.3.

B.1 Prerequisites and limitations

Pywmtk needs the following packages to be installed. The versions in brackets are the ones I used and which definitely work, but others may work too.

- Python (2.5.4) - http://www.python.org/
- NumPy (1.2.1) - http://numpy.scipy.org/
- SciPy (0.7.0) - http://www.scipy.org/
- PyWavelets (0.1.7) - http://wavelets.scipy.org/ - the currently latest official release (0.1.6) should also work
- Phyton Imaging Library (1.1.6) - http://www.pythonware.com/products/pil/

You also may want to install

- Psyco - http://psyco.sourceforge.net/

which is not mandatory but speeds up the execution of the scripts.

All algorithms read and write images through the PIL library, so for a list of supported image formats, see the documentation of your PIL installation. You can use portable grey map images (.pgm) for instance. All algorithms are implemented using only quadratic, grey level images where the side length is a power of 2.

The toolkit is developed and tested under Linux, I do not know if it works under other operating systems or what is necessary to make it run under other operating systems.

B.2 Usage - Python

To use the pywmtk package you have to put the pywmtk folder from the src directory somewhere in your PYTHONPATH or you put the script you want to develope into the src directory. You do not need the scripts which are currently in the src directory, they are only for the command line interface. To use a watermarking algorithm you need to import the according module. E.g.:

```
from pywmtk.watermarkingAlgorithms import pla
```

and then call some function from it, e.g.:

```
markedArray = pla.embed(<plainArray>, <watermark>)
```

The pywmtk package is divided into several subpackages:

- experiments: the experiments I carried out for my master thesis. The modules herein will probably be of no use for you, but you can inspect the code to see how the other modules are interfaced.

- test: If you are looking for examples, this is the place to go. watermarkTest.py contains complete embedding and detection examples for all watermarking algorithms, wavetransTest.py contains examples for the python implemented wavelet transformation.

- tools: useful stuff for watermarking and wavelet transformation

- watermarkingAlgorithms: contains all implemented watermarking algorithms

- wavelet: package for wavelet transformation, including wavelet packet transformation, stationary wavelet transformation and parametrized filter generation. Be aware, that the dwt and swt are implemented in pure Python and therefore are rather slow. For the dwt you should better use the wavelet transformation from the PyWavelets package.

Every watermarking algorithm contains an embed and a detect function, additionally there can be a generateWatermark and/or a generateThreshold function. Since many watermarking algorithms need the watermark in a specific length or shape and the coefficients to be specific numerical values the generateWatermark function takes a seed which is the information you can embed. The generateWatermark function can be seen as a mapping function from your chosen watermark (seed) to the watermark which is finally embedded. The embed and detect functions need at least the source image as a numpy array and a watermark (e.g. from the generateWatermark function) as parameters. If the threshold calculation is described in the according paper of the algorithm, this is implemented in the generateThreshold function. The modules can contain some more functions which are used during the embed or detect procedure or which are improved or alternative versions of the algorithm or provide specific tools for this algorithm (e.g. statistical analysis or visualization of parts of the algorithm).

The Broken Arrows algorithm is a modified version of the original algorithm of Teddy Furon and Patrick Bas. The Pywmtk interfaces the algorithm through os calls and therefore relies on the relative path to the executables. So the BrokenArrows folder has to be in the same directory as the src directory or the path has to adapted in the brokenArrows.py module.

For more information consult the API documentation which can be found in the doc directory.

B.3 Usage - Command line

To run a watermarking algorithm the prerequisites (see section B.1) have to be satisfied. Then call the appropriate script in the src directory, e.g.:

```
python pla_embed.py 123 lena.pgm marked.lena.pgm
```

All scripts belonging to one algorithm start with the algorithm name (e.g. pla). If the watermarking algorithm needs the watermark in a specific length or shape and the coefficients to be specific numerical values a script called <algorithm name>_generateWm is available. The generateWm script takes a seed which is the information you can embed. The generateWm script

can be seen as a mapping function from your chosen watermark (seed) to the watermark which is finally embedded. The generated watermark can be used to be embedded with the <algorithm name>_embed script. Watermark detection is done with the <algorithm name>_detect script. If the threshold calculation is described in the according paper of the algorithm, this is implemented in the <algorithm name>_threshold function.

All scripts have a usage description and information on the arguments and options which can be accessed with the -h option.

The command line interface does not provide the full functionality of the algorithms and their modifications I implemented, but is capable of embedding and extracting with the most common parameters.

Additional scripts are:

- psycoTest.py: tells you if Psyco is installed and working
- diff_image: creates the difference image between two images

B.4 Legal Notice

A modified version of the Broken Arrows algorithm is distributed with the Pywmtk. This part of the software is governed by the CeCILL license, which can be obtained from http://www.cecill.info. A copy of the license is also contained in the BrokenArrows folder (Licence_CeCILL_V2-en.txt). The license commits me to say: you should read it, especially articles 8 and 9. So read it.

The pywmtk package is distributed under the following license:

This material is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY. No author or distributor accepts responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all.

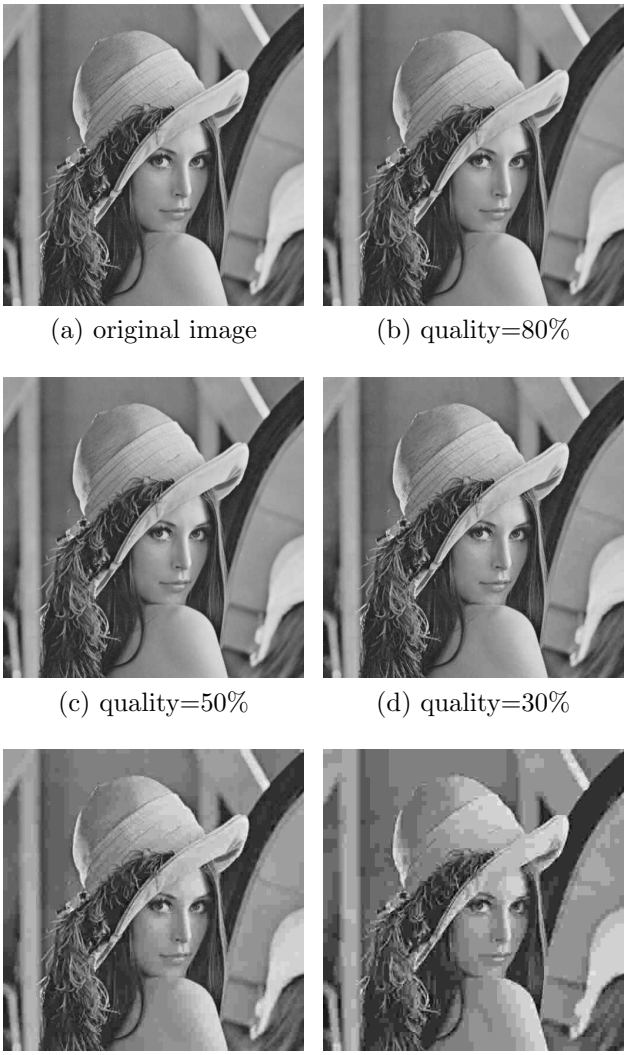
The material is prepared strictly for research use only, commercial use is prohibited. Do not distribute the material without written permission. If you publish any work based on this code, please cite the original paper.

Appendix C

attacked images

This chapter shows some attacked Images to get an impression of which distortions are added to the images.

C.1 jpeg compression



(a) original image (b) quality=80%

(c) quality=50% (d) quality=30%

(e) quality=10% (f) quality=5%

Figure C.1: jpeg compressed Lena with different qualities

C.2 jpeg2000 compression

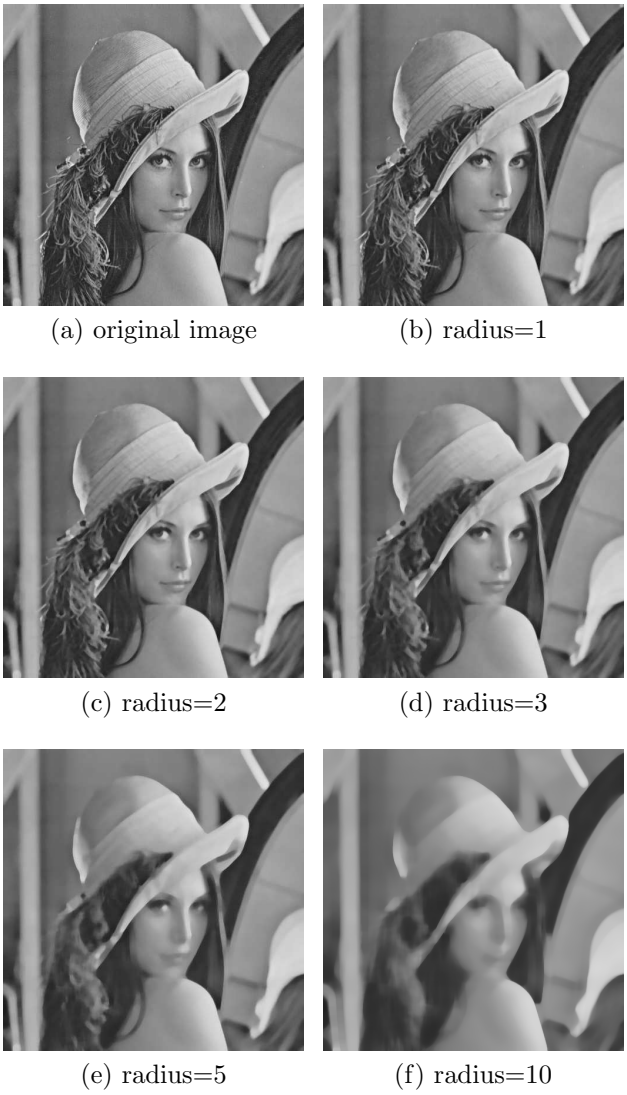
(a) original image (b) rate=1

(c) rate=0.5 (d) rate=0.2

(e) rate=0.1 (f) rate=0.01

Figure C.2: jpeg2000 compressed Lena with different rates

C.3 median filtering



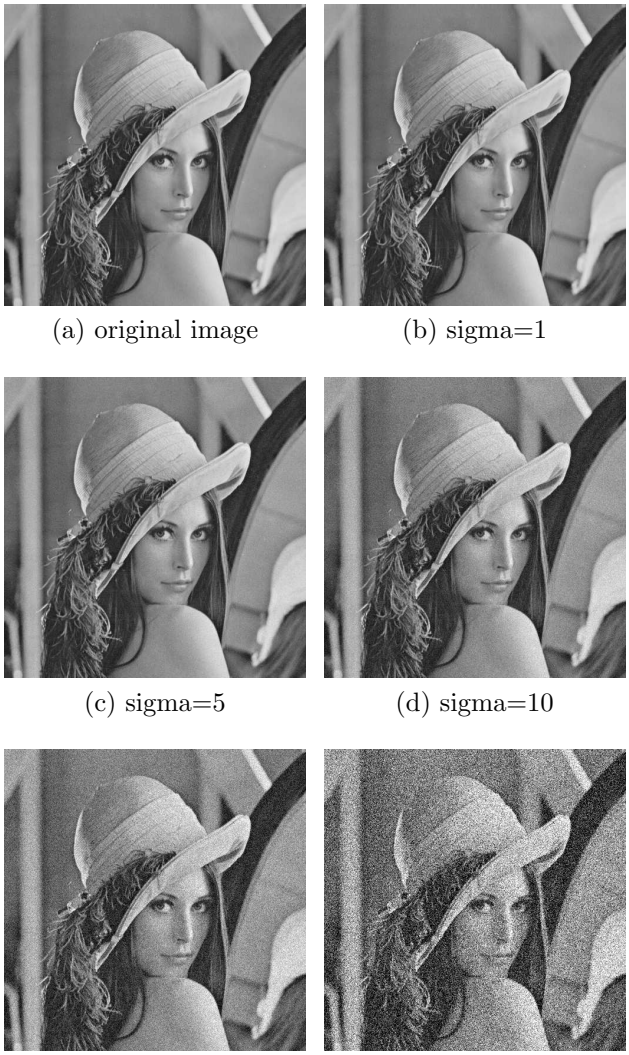
(a) original image (b) radius=1

(c) radius=2 (d) radius=3

(e) radius=5 (f) radius=10

Figure C.3: median filtered Lena with different radii

C.4 moise adding



(a) original image (b) sigma=1

(c) sigma=5 (d) sigma=10

(e) sigma=20 (f) sigma=50

Figure C.4: Lena with added noise with different sigma

C.5 gamma correction

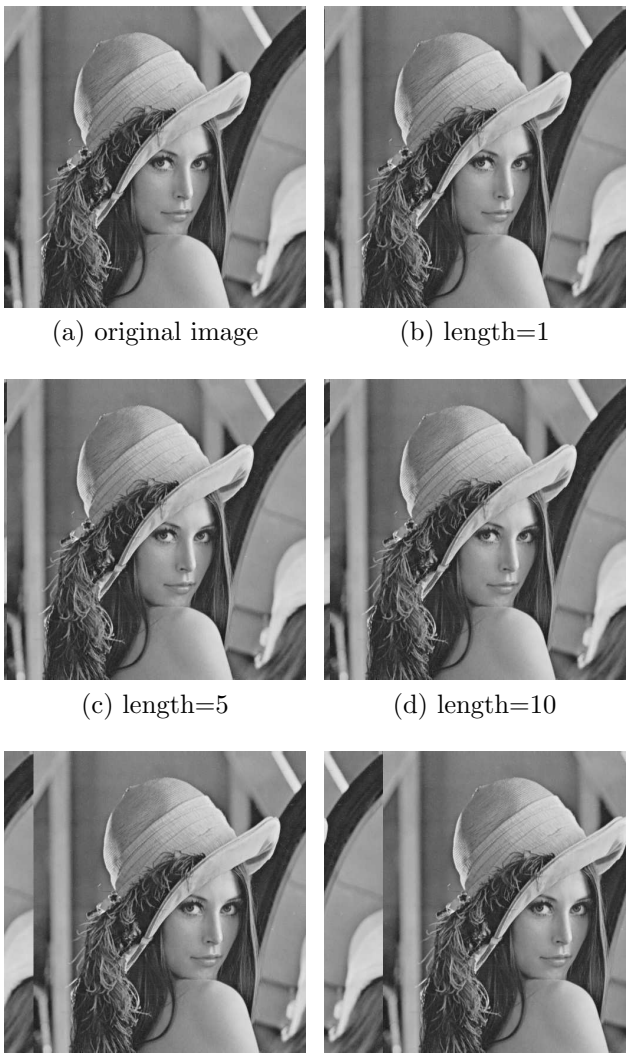
(a) original image (b) gamma=01

(c) gamma=05 (d) gamma=2

(e) gamma=10 (f) gamma=20

Figure C.5: gamma corrected Lena with different gamma

C.6 translation



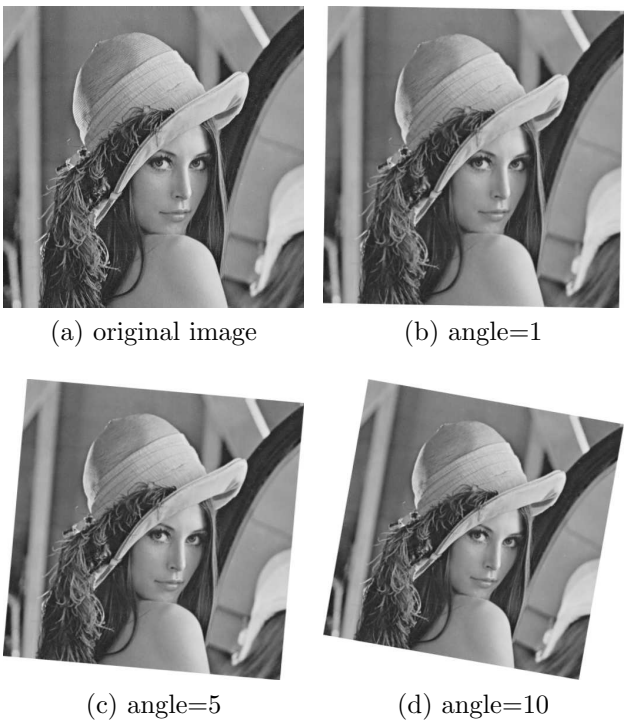
(a) original image (b) length=1

(c) length=5 (d) length=10

(e) length=50 (f) length=100

Figure C.6: translated Lena with different length

C.7 rotation



(a) original image (b) angle=1

(c) angle=5 (d) angle=10

Figure C.7: rotated Lena with different angles

Bibliography

[1] M. Barni and F. Bartolini. *Watermarking Systems Engineering*. Marcel Dekker, 2004.

[2] Mauro Barni, Franco Bartolini, and Alessandro Piva. Improved wavelet-based watermarking through pixel-wise masking. *IEEE Transactions on Image Processing*, 10(5):783–791, May 2001.

[3] Franco Bartolini, Mauro Barni, Vito Cappellini, and Alessandro Piva. Mask building for perceptually hiding frequency embedded watermarks. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, volume 1, pages 450–454, Chicago, IL, USA, October 1998.

[4] Roberto Caldelli, Mauro Barni, Franco Bartolini, and Alessandro Piva. Geometric-invariant robust watermarking through constellation matching in the frequency domain. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'00)*, Vancouver, Canada, September 2000.

[5] J.-G. Cao, J. Fowler, and N. Younan. An image-adaptive watermark based on a redundant wavelet transform. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'01)*, volume 2, pages 277–280, Thessaloniki, Greece, October 2001.

[6] Tung-Shou Chen, Jeanne Chen, and Jian-Guo Chen. A simple and efficient watermark technique based on JPEG2000 codec. *ACM Multimedia Systems Journal*, 10(1):16–26, June 2004.

[7] M. El Choubassi and P. Moulin. On the fundamental tradeoff between watermark detection performance and robustness against sensitivity analysis attacks. In Edward J. Delp and Ping W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072 of *Proceedings of SPIE*, pages 575–586. SPIE, January 2006.

[8] Charilaos Christopoulos, Athanassios N. Skodras, and Touradj Ebrahimi. The JPEG2000 still image coding system: an overwiew. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, November 2000.

[9] Mac A. Cody. The fast wavelet transform — beyond fourier transforms. *Dr. Dobb's Journal of Software Tools*, 15(4), April 1992.

[10] R. R. Coifman, Y. Meyer, S. Quake, and Mladen Victor Wickerhauser. Entropy based algorithms for best basis selections. *IEEE Transactions on Information Theory*, 38(2):713–718, March 1992.

[11] Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2007.

[12] S. Craver and J. Yu. Reverse-engineering a detector with false-alarms. In Edward J. Delp and Ping W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505 of *Proceedings of SPIE*, San Jose, CA, USA, January 2007. SPIE.

[13] Ingrid Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM Press, Philadelphia, PA, USA, 1992.

[14] Werner Dietl. Improving the security of wavelet-based watermarking systems. Master's thesis, University of Salzburg, Department of Scientific Computing, December 2002.

[15] Rakesh Dugad, Krishna Ratakonda, and Narendra Ahuja. A new wavelet-based scheme for watermarking images. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, volume 2, pages 419–423, Chicago, IL, USA, October 1998.

[16] Dominik Engel. *Media Encryption for Still Visual Data – An Analysis of Selected Techniques for Natural Images and Fingerprint Data in the Spatial and Wavelet Domain*. PhD thesis, Department of Computer Sciences, University of Salzburg, Austria, June 2008.

[17] Teddy Furon and Patrick Bas. Broken arrows. *EURASIP Journal on Information Security*, August 2008. ID 597040.

[18] A.S. Lewis and G. Knowles. Image compression using the 2-D wavelet transform. *IEEE Trans. on Image Process.*, 1(2):244–250, April 1992.

[19] S. Mallat. *A wavelet tour of signal processing*. Academic Press, 1997.

[20] Neri Merhav and Erez Sabbag. Optimal watermark embedding and detection strategies under limited detection resources. *IEEE Transactions on Information Theory*, 54(1):255–274, January 2008.

[21] Matthew L. Miller and Jeffrey A. Bloom. Computing the probability of false watermark detection. In Andreas Pfitzmann, editor, *Proceedings of the 3rd Information Hiding Workshop '99*, volume 1768, pages 146–158, Dresden, Germany, October 1999. Springer.

[22] Matthew L. Miller, Jeffrey A. Bloom, and Ingemar J. Cox. Informed embedding: exploiting image and detector information during watermark insertion. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'00)*, Vancouver, Canada, October 2000.

[23] Frederick C. Mintzer, L. E. Boyle, A. N. Cazes, B. S. Christian, S. C. Cox, F. P. Giordano, Henry M. Gladney, Jong Chan Lee, M. L. Kelmanson, A. C. Lirani, Karen A. Magerlein, A. M. B. Pavani, and Fabio Schiattarella. Toward on-line, worldwide access to vatican library materials. *IBM Journal of Research & Development*, 40(2), 1995.

[24] Frederick C. Mintzer and Gordon W. Braudaway. If one watermark is good, are more better? In *Proceedings of the 1999 International Conference on Acoustics, Speech and Signal Processing (ICASSP'99)*, volume 4, pages 2067–2070, Phoenix, AZ, USA, March 1999.

[25] Frederick C. Mintzer, Gordon W. Braudaway, and Minerva M. Yeung. Effective and ineffective digital watermarks. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'97)*, volume 3, page 9, Santa Barbara, California, USA, October 1997.

[26] Stephane Pateux and Gaetan Le Guelvouit. Practical watermarking scheme based on wide spread spectrum and game theory. *Signal Processing: Image Communication*, 18(4):283–296, April 2003.

[27] Shelby Pereira and Thierry Pun. Robust template matching for affine resistant image watermarks. *IEEE Transactions on image processing*, 9(6):1123–1129, June 2000.

[28] Alessandro Piva, Mauro Barni, Franco Bartolini, and Vito Cappellini. Threshold selection for correlation-based watermark detection. In *Proceedings of COST International Workshop on Intelligent Communications*, pages 67–72, L'Aquila, Italy, June 1998.

[29] Oriol Guitart Pla, Eugene T. Lin, and Edward J. Delp. A wavelet watermarking algorithm based on a tree structure. In Edward J. Delp and Ping W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306 of *Proceedings of SPIE*, pages 571–580, San Jose, CA, USA, January 2004. SPIE.

[30] N. M. Rajpoot, Francois G. Meyer, R. G. Wilson, and R. R. Coifman. On zerotree quantization for embedded wavelet packet image coding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, Kobe, Japan, October 1999.

[31] Nasir Rajpoot, Francois Meyer, Roland Wilson, and Ronald Coifman. Progressive wavelet packet image coding using compatible zerotree quantization. Technical report, Department of Computer Science, Yale University, USA, September 1999.

[32] J. Schneid and S. Pittner. On the parametrization of the coefficients of dilation equations for compactly supported wavelets. *Computing*, 51:165–173, May 1993.

[33] Nicholas Paul Sheppard, Reihaneh Safavi-Naini, and Philip Ogunbona. On multiple watermarking. In *Proceedings of the 9th ACM Multimedia 2001 Conference*, pages 3–6, Ottawa, Ontario, Canada, September 2001.

[34] W. Sweldens and P. Schröder. Building your own wavelets at home. ACM SIGGRAPH course notes, 1996.

[35] George Voyatzis and Ioannis Pitas. Chaotic mixing of digital images and applications to watermarking. In *European Conference on Multimedia Applications, Services and Techniques, ECMAST '96*, volume 2, pages 687–695, Louvain-la-Neuve, Belgium, May 1996.

[36] Hartmut Wernisch. Multiple watermarking for digital rights management. Master's thesis, University of Salzburg, Department of Scientific Computing, June 2006.

[37] M.V. Wickerhauser. INRIA lectures on wavelet packet algorithms. Lecture notes, INRIA, 1991.

[38] M.V. Wickerhauser. *Adapted wavelet analysis from theory to software.* A.K. Peters, Wellesley, Mass., 1994.

[39] Gin-Der Wu and Pang-Hsuan Huang. Image watermarking using structure based wavelet tree quantization. In *Proceedings of the 6th*

IEEE/ACIS International Conference on Computer and Information Science, 2007. ICIS 2007, pages 315–319. IEEE, July 2007.

[40] Dan Xu and Minh N. Do. Anisotropic 2-D wavelet packets and rectangular tiling: theory and algorithms. In Michael A. Unser, Akram Aldroubi, and Andrew F. Laine, editors, *Proceedings of SPIE Conference on Wavelet Applications in Signal and Image Processing X*, volume 5207 of *SPIE Proceedings*, pages 619–630, San Diego, CA, USA, August 2003. SPIE.