Robust Watermarking of digital Color Images using Blue Channel Amplitude Modulation

Roger Meier

University of Rhode Island
Department of Electrical and Computer Engineering

ELE585 Final Project
Spring 1999

1. Introduction

The emergence of digital media and of digital networks has made duplication of original digital artwork easier. In order to protect these creations, new methods for signing and copyrighting visual data are needed. Watermarking techniques, also referred to as signature, sign images by introducing changes that are imperceptible to the human eye but easily recoverable by a computer program. The locations in the image, where the signature is embedded, are determined by a secret key. This prevents pirates from easily removing the signature. Furthermore, it should be possible to retrieve the signature from an altered image. Alternations, also referred to as attacks, could be blurring, softening, sharpening, cropping, resizing, rotating, compressing, contrast balancing, noise contamination, filtering, etc.
The watermarking method on which this project is based, was introduced by Martin Kutter in [1]. In this technique signature bits are multiply embedded by modifying pixel values in the blue channel. These modifications are either additive or subtractive, depending on the value of the bit, and proportional to the luminance. This method has shown to be resistant to both classical and geometrical attacks. Furthermore, the signature can be extracted without the original image.
The goal of this project was it, to implement the method described in [1] with Matlab®, to confront the technique with all sorts of image attacks, and to investigate the robustness of this method to these attacks.
This report is stuctured as follows: Chapter 2 and 3 will explain the embedding and retrieval of a digital signature. Additionally, the implementation of the introduced algorithms in Matlab®, will be clarified. Chapter 4 will introduce different attacks, and the robustness of this watermarking technique to each attack will be shown.

2. Watermark embedding

The two main requirements for a digital watermark are invisibility to the human eye and robustness to alterations. Invisibility is achieved by embedding the signature in the blue channel, to which the human eye is least sensitive. Moreover, changes in regions of high frequency and high luminance are less visible. Therefore, the signature can be embedded much stronger in these regions. The blue channel amplitude modulation method favors these regions by modifying the blue channel by a fraction of the luminance. Robustness to attacks is achieved by embedding the signature multiple times at many different locations in the image.

2.1 Single Bit Embedding

2.1.1 Modulating the Blue Channel

Let b be a single bit to be embedded in the RGB image I, and $p=(x,y)$ a pseudo-random position within I. The luminance L is a linear combination of the three color channels R, G, and B, and is defined as $L=0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$ (ITU-R 601 Standard). The bit s is embedded in the blue channel B at position p by a fraction of the luminance L:

$$B_{xy} \leftarrow B_{xy} + q(2s-1)L_{xy} \tag{1}$$

where q is a constant determining the signature strength. The value q is selected such as to offer best trade-off between robustness and invisibility. In tests the value $q=0.1$ was found to be a good value.

2.1.2 Determining the Signature Embedding Positions

The positions where the signature bit will be embedded are depending on a secret key, which is used as a seed for the pseudo-random number generator, and a density parameter d, which gives the probability of any single pixel being used for embedding. Hence, this value lies between 0 and 1, where 0 means that no information is embedded, and 1 that information is embedded in every pixel. After initializing the pseudo-random number generator with the seed, a pseudo-random number x is generated for each pixel of the image. If x is smaller than d, then the pixel will be used for information embedding, otherwise the pixel will be left intact.
This can be implemented in Matlab® very easily, as shown below:

```
% Feeding random number generator with seed
% -----------------------------------------
rand('state',seed);

% Calculating embedding positions
% -------------------------------
imheight=size(I,1);
imwidth=size(I,2);
emb = (rand(imheigth,imwidth) < d);
[y,x] = find(emb);
p = [y x];
n = length(x);
```

The matrix *emb* is a matrix of the same size as the image *I*, and its elements are either 1 or 0. If it is 1, then the pixel will be used for embedding, if it is 0, then the pixel will be left unchanged. Figure 1 shows the matrix *emb* for a density parameter of $d=0.55$. The 2-column matrix p contains all the embedding positions as XY pairs. These positions are in an order that starts on the top left corner of the image and ends on the bottom right corner. n is the number of embedding positions.

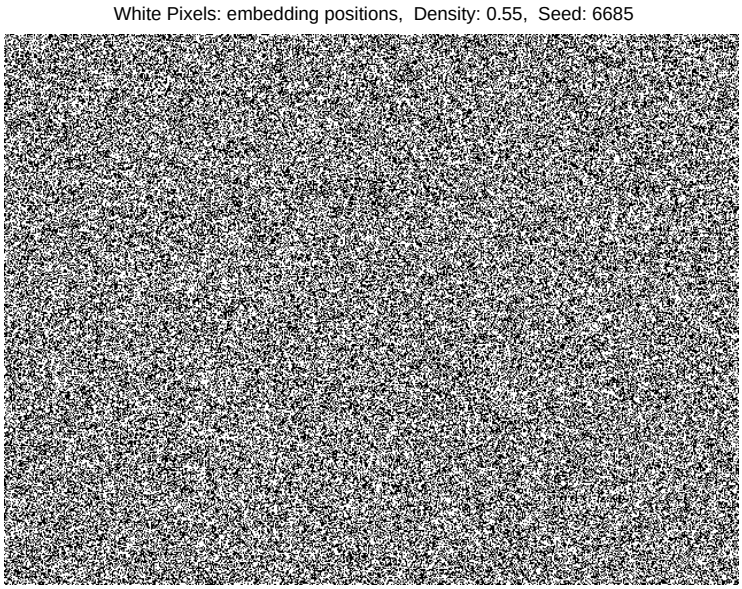


Figure 1: Calculated embedding positions for a 640 by 480 pixels image

2.2 Multibit Signature Embedding

The extension to an m-bit signature $S=\{s_0,\ldots,s_{m-1}\}$ is straightforward. Let $p_1 \ldots p_n$ be the n embedding positions selected by the routine introduced above. For each of these positions a signature bit is randomly selected and embedded. Again, before randomly selecting these positions, the pseudo-random number generator has to be fed with the same seed.
The implementation in Matlab® is as follows:

```
% Feeding random number generator with seed
% ------------------------------------------
rand('state',seed);

% 'Shuffeling' the positions of the embedding bits
% ------------------------------------------------
order=randperm(n);
p=p(order,:);

% Generating the watermark bitstring of length n
% by lining up copies of the signature bitstring S={s(1),…,s(m)}
% --------------------------------------------------------------
watermark=zeros(n,1);
for k=1:floor(n/m)
   watermark(m*(k-1)+1:m*k)=S';
end
```

Instead of randomly selecting signature bits for each position, the positions stored in the n by 2 matrix p are 'shuffled' using the Matlab® command `randperm` which also depends on the seed that is given to the pseudo-random number generator. Then a column vector of length n is created by lining up identical copies of the signature vector S. The n bits in S are then embedded at the n scattered positions in p, using the embedding method described in 2.1.1.
Figure 2 shows the 640 by 480 pixels image `mir.tif`, which is one of the images used for watermarking.

Figure 2: `mir.tif`, the original image used for watermarking

The image consists of both dark and bright areas, and regions of both low and high frequency, thus a good image for demonstrating the watermarking technique discussed in this report.
Figure 3 shows the difference between the original blue channel of `mir.tif` and the watermarked blue channel using an embedding density of d=0.55. It can be seen clearly, that the differences are bigger in the brighter regions than in the darker ones. The differences are either positive or negative, reflecting the signs of the signature bits embedded at these locations (0: negative, 1: positive). The signature strength q used in this case is q=0.1, which can be verified by looking at the color bar on the right side of the image. The maximum magnitude of the difference is 25.5, which is exactly the highest possible pixel value (i.e. 255) multiplied by q. After embedding the watermark the new blue channel consists of values that are below 0 and above 255. To avoid overflows when creating the new RGB image these values have to be limited to 0 and 255, respectively.

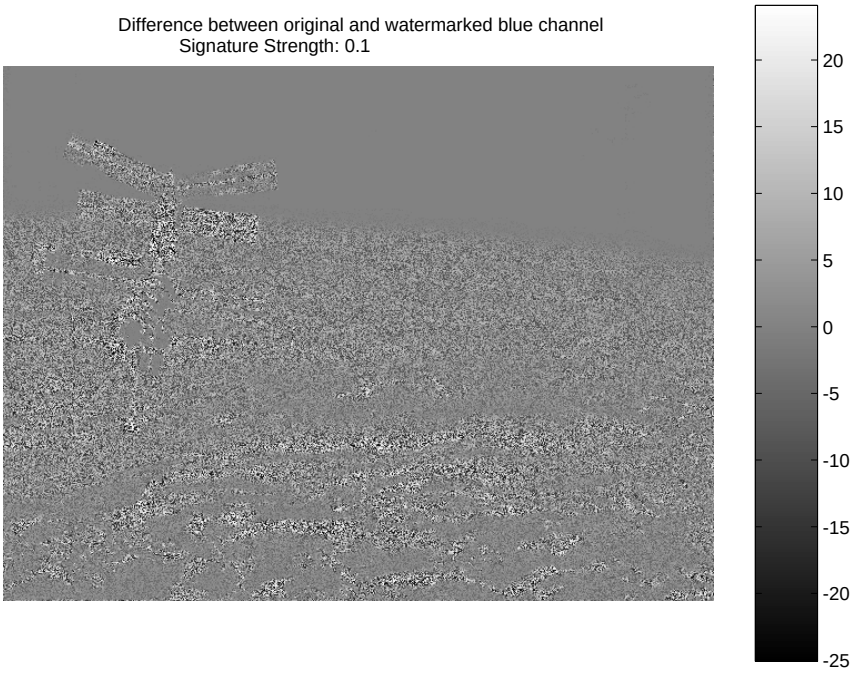


Figure 3: Difference between watermarked and original blue channel

To improve signature retrieval the signature is extended by 2 bits. These 2 bits are always set to 0 and 1, respectively. This allows defining an adaptive threshold which improves signature retrieval, and it defines a geometrical reference which is used to counter geometrical attacks, such as rotation, cropping, and translation.

2.3 Robustness versus Invisibility

Figure 4: q=0.1

Figure 5: q=0.3

The signature strength value q has to be chosen such as to offer best trade-off between robustness and invisibility. `mir.tif`, watermarked with strength q=0.1 is shown in Figure 4. This value was found to be a very good choice, the embedded signature is not visible at all, and it is very robust to all kinds of attacks, which will be shown in chapter 4. Figure 5 shows the same image but watermarked with strength q=0.3. This watermark is extremely robust, but it is also slightly visible. The image appears to be noise contaminated.

3. Watermark Retrieval

3.1 Using original Image

One way of retrieving the embedded watermark is to extract the blue channels from the original and the watermarked image, and subtract the original from the watermarked blue channel. The result is a matrix containing positive and negative values which correspond to the signs of the embedded signature bits. The only problem is, that in most cases the original image won't be available. Therefore, a way of retrieving the signature without needing the original image is needed.

3.2 Predicting the Original Pixel Values

In order to recover an embedded bit without knowing the original value of the pixel containing the information, a prediction of the original value is needed. The prediction is based on a linear combination of the pixel values in a neighborhood around p. According to [1] empirical results have shown, that taking a cross-shaped neighborhood gives best performance. The prediction is thus computed as

$$\hat{B}_{xy} = \frac{1}{4c}\left(\sum_{k=-c}^{c} B_{x+k,y} + \sum_{l=-c}^{c} B_{x,y+l} - 2B_{xy} \right) \tag{2}$$

where c is the size of the cross-shaped neighborhood.
The implementation in Matlab® is different, but more elegant:

```
% Creating cross-shaped neighborhood
% ----------------------------------
cross=zeros(2*c+1,2*c+1);
cross(:,c+1)=1;
cross(c+1,:)=1;
cross(c+1,c+1)=0;

% Estimating original blue values
% -------------------------------
B_hat=1/(4*c) * filter2(cross,B,'same');
```

Instead of building numerous loops to calculate the sums for all the single pixel locations, a cross-shaped 2 dimensional filter is generated. Then the watermarked blue channel is convolved with the filter using the Matlab® function `filter2`. Doing so allows the fastest possible estimation of the original pixel values. The size c=3 for the cross-shaped neighborhood was found to be a good choice.
To retrieve an embedded bit the difference δ between the prediction and the actual value of the pixel is taken:

$$\delta = B_{xy} - \hat{B}_{xy} \tag{3}$$

The sign of the difference δ determines the value of the embedded bit.
The embedding and retrieval functions are not symmetric, i.e. the retrieval function is not the inverse of the embedding function. Even though correct retrieval is very likely, it is not guaranteed. But since each of the signature bits is embedded several times, the probability of incorrect retrieval is reduced to a minimum. To successfully retrieve a bit s^i of the signature S, the difference between the prediction and the actual value of the pixel is computed for every position p_k where this particular bit is embedded:

$$\delta_k = B_{p_k} - \hat{B}_{p_k} \tag{4}$$

These differences are then averaged:

$$\bar{\delta}^i = \frac{1}{K}\sum_k \delta_k \tag{5}$$

The sign of the average difference determines the value of the embedded bit. This procedure is done for all the bits s^i of the signature S.
Of course, before a signature can be retrieved, all the n the embedding positions need to be known. These positions can be obtained the same way they are obtained when watermarking the image, as described in 2.1.1. All that needs to be known about the watermarked image is the seed for the pseudo-random number generator, the signature density parameter d and the signature length m. After watermarking an image these 3 numbers could be encoded into a retrieval code. This way only one number would have to be memorized.

3.3 Adaptive Threshold

The sign of each average difference is a very good decision function. However, after an attack, this might not be so anymore. Therefore the decision function needs to be adapted. Since it is known, that the first 2 bits of the signature S always have values 0 and 1, respectively, this information can be used to compute an adaptive decision threshold. This threshold is defined as the average between δ^0 and δ^1:

$$\bar{s}^i = \begin{cases} 1 & \text{if } \bar{\delta}^i > \dfrac{\bar{\delta}^0 + \bar{\delta}^1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Figure 6 shows the retrieved average differences of a 30 bit signature. For this example, the parameters have been set as follows: Embedding density d=0.55, signature strength q=0.1, size of cross-shaped neighborhood c=3.

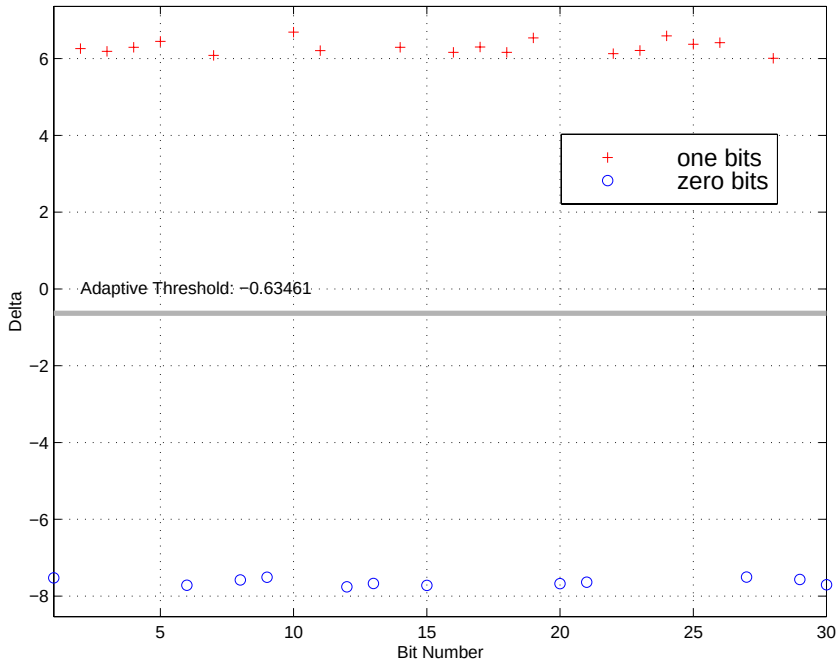


Figure 6: The average differences of a 30 bit signature

The one bits can clearly be distinguished from the zero bits, and the difference between the positive and the negative deltas is significant. The effect of an attack on these deltas will be shown in chapter 4. It will be shown, that using an adaptive threshold is very important for a successful signature retrieval.

4. Robustness to Attacks

This section is dedicated to the investigation of the robustness of the blue amplitude modulation watermarking technique introduced in the last two sections to all sorts of image tampering.
To illustrate the robustness, a test image was watermarked with a 30 bit signature several times using a wide range of different signature strengths. Then all of these watermarked images were attacked with different intesities, and for all images a signature retrieval was executed. By comparing the retrieved signature to the original in each case, the robustness in dependence to signature strength and attack intensity could be investigated. This procedure was followed for all the attacks discussed in this section. The result of such a procedure is a 2 dimensional array showing the number of correctly retrieved signature bits as a function of signature strength and attack intensity.
The test image used for this investigation was the file `mir.tif` which is depicted in Figure 2. The image was watermarked with the string `'test'`, which is a 30 bit signature (4*7 bit ASCII plus 2 reference bits).
Also, for each sort of attack, an example was created to show the effect of the attack on the image and on the retrieved average differences δ^i. The watermarked image used for these examples is depicted in Figure 7, and the retrieved average differences are shown in Figure 8. The image was also watermarked with the 30 bit signature `'test'`, using the signature strength q=0.1, and the embedding density d=0.55. The size of the cross-shaped neighborhood used for the retrieval of the signature was chosen c=3. The retrieved average differences given for each example directly reflect the quality of the signature embedded in the tampered image

Figure 7: Watermarked image, q=0.1

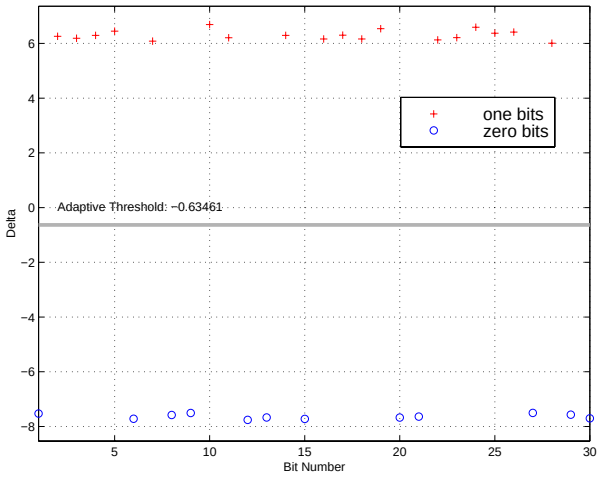


Figure 8: Retrieved average differences

4.1 Geometrical Attacks

In order to recover from geometrical attacks, it is required for the recovering algorithm to be able to determine what sort of attack (rotation, translation, cropping) has been applied to produce the tampered image. To estimate this transform, a reference is needed. The 2 first bits of the signature fulfill this requirement, since these bits are always set to 0 and 1, respectively. Therefore, a known pattern is hidden within the image, and by looking for this pattern the transform can be found.
Let F be the transform applied to the watermarked image I to obtain the tampered image J. In order to retrieve the signature, the inverse F^{-1} of F is needed. By applying F^{-1} to the tampered image J the original watermarked image I is recovered, and the signature can be extracted.
Let G be an estimation of F^{-1}, and I_G the image obtained by applying G to J. If G is equal to F^{-1}, then the 2 first bits of the signature can be clearly retrieved from I_G. Also, the confidence of the retrieval is very high, i.e. the difference between δ^0 and δ^1 is maximum. If G is slightly different from F^{-1} then the signature can still be retrieved but the difference between δ^0 and δ^1 is smaller than before. This difference gets smaller as the divergence between G and F^{-1} grows.
Unfortunately, the difference in function of the divergence between G and F^{-1} is not a smooth function. Thus, conventional optimization methods are not suitable. Instead, full search methods have to be used. If the nature of the transform F is given, then the search can be sped up a lot. For instance if it is know, that the transform is a pure rotation or translation, then the search space is greatly reduced. Furthermore, by looking at the tampered image, good assumptions about the transform can be made. For example a rotation angle or a translation distance can be guessed with high precision, and hence, the search space can be reduced even more.

4.1.1 Rotation

Figure 9: The watermarked image, rotated by an unknown rotation angle

In the case that the transform applied to the watermarked image is a pure rotation, then by looking at the rotated image, the rotation angle can be guessed. Figure 9 shows the watermarked image, that was rotated by an unknown angle. But by looking at the image, it can be said, that the rotation angle must be between 50° and 70°. Therefore the search method can be restricted to this particular interval. Once a maximum difference between δ^0 and δ^1 is found (see left graph in Figure 10), the search can be narrowed down even more, and smaller increment steps can be used for searching (see right graph in Figure 10). The rotation angle was found to be 59.75°

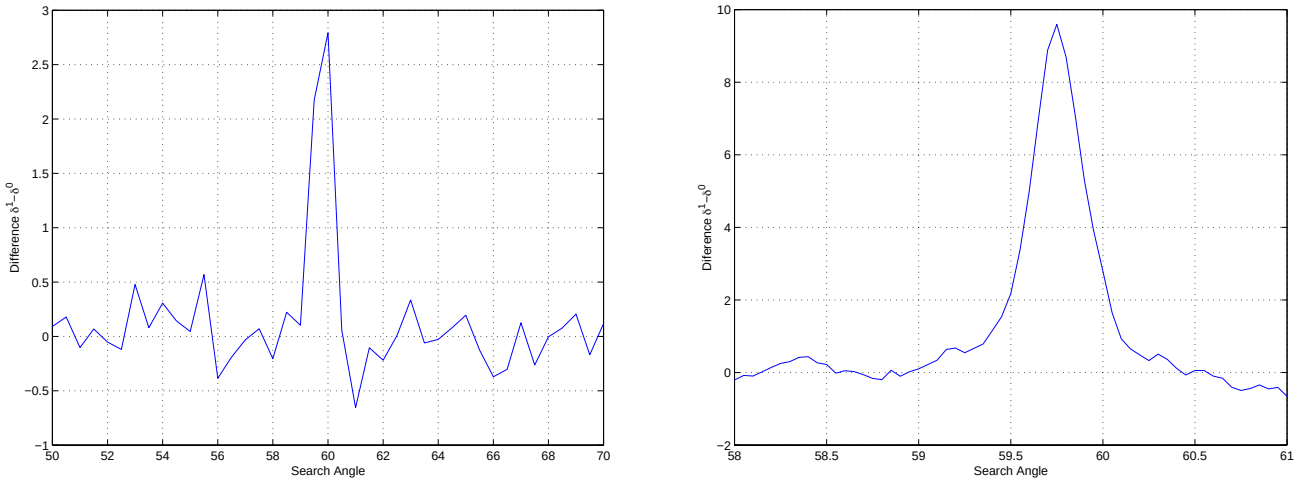


Figure 10: Result of the search method, found rotation angle: 59.75°

Figure 11: The image, rotated back by 59.75°

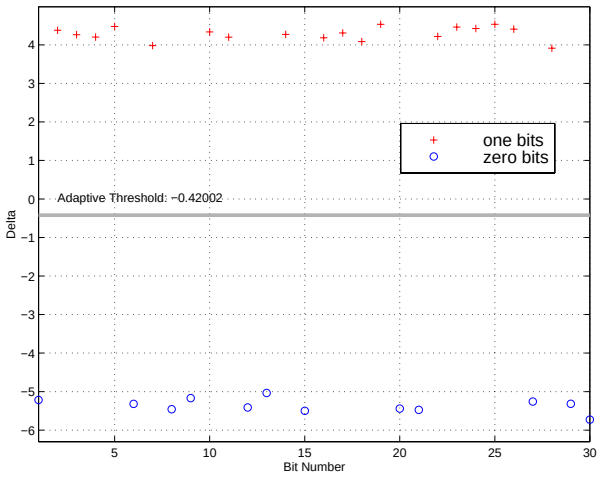


Figure 12: The retrieved average differences

The Figures 11 and 12 show the image and the retrieved signature deltas after rotating the tampered image back by the angle that was found. The signature can be retrieved correctly if the angle used for back rotation is within +/-0.5° of the actual angle

4.1.2 Cropping

To retrieve the embedded watermark from a cropped image, the original location of the cropped part within the original image has to be found. The only information about the original image necessary is the size, i.e. width and height. Then, an empty image (all pixel values = 0) of the same size is created. The cropped image is then moved around within the empty image until the location is found for which the difference δ^0 and δ^1 is maximum. Once the original location is found, the signature can be retrieved

Figure 13 shows the robustness of the watermark to cropping in dependence to signature strength and size of the cropped partial image. It can be seen clearly, that the watermark is very robust, even for very small sizes of the cropped image. For an embedding strength of q=0.1 the signature can still correctly be retrieved from a cropped image which size is only 6% of the original image.

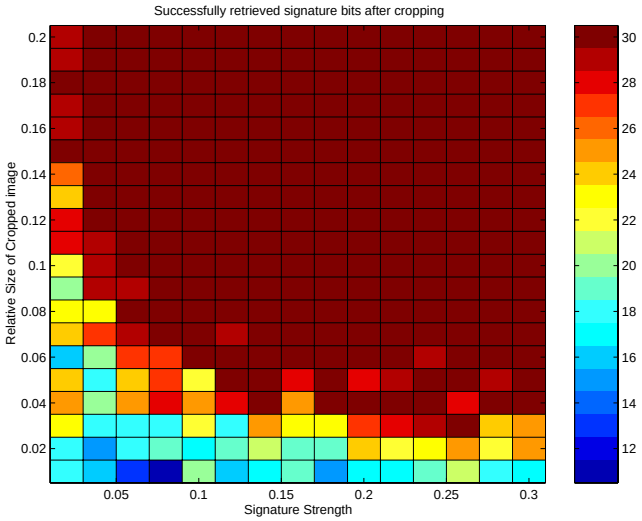


Figure 13: Robustness to cropping

Example:
A 80 by 80 pixel portion, depicted in Figure 14, is cropped from the watermarked image. Before the watermark can be retrieved, the original location of this portion within the original image has to be found. After guessing the location, a 2 dimensional search algorithm is used to find the actual position. Once the position is found the watermark can be retrieved

Figure 14: The cropped portion

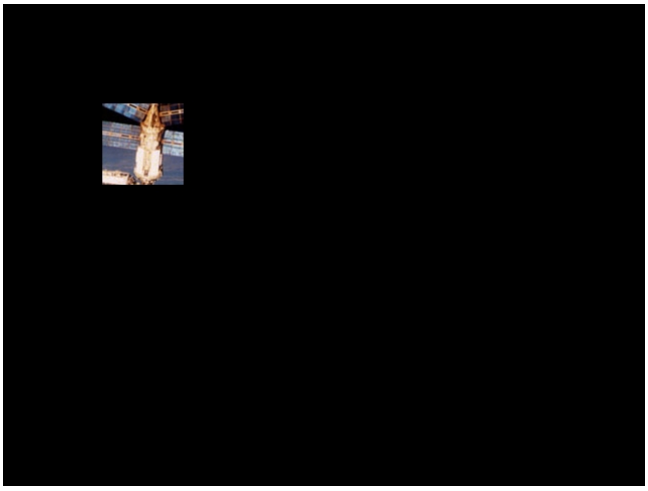


Figure 15: The cropped image, put back in place

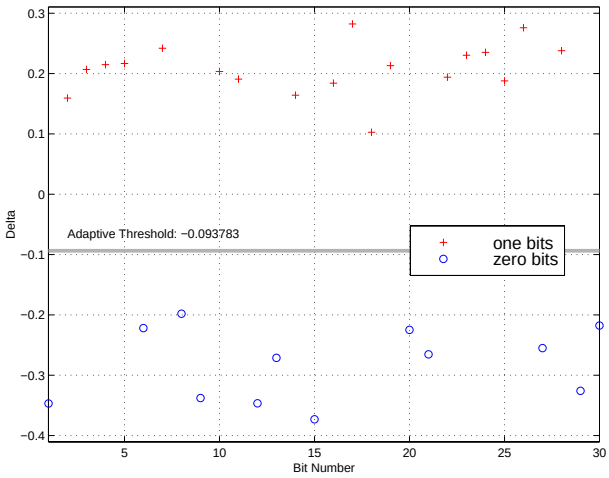


Figure 16: The retrieved average differences

4.1.3 Resizing

In the case, that a watermarked image was resized, the factor by which the original image has been scaled to obtain the tampered image, needs to be known. The tampered image then is scaled back by the inverse of the scaling factor. The inverse scale factor can also be found by dividing the size of the tampered image by the size of the original image, given that the dimensions of the original are known. After the inverse transform has been applied the signature can be retrieved from the image.

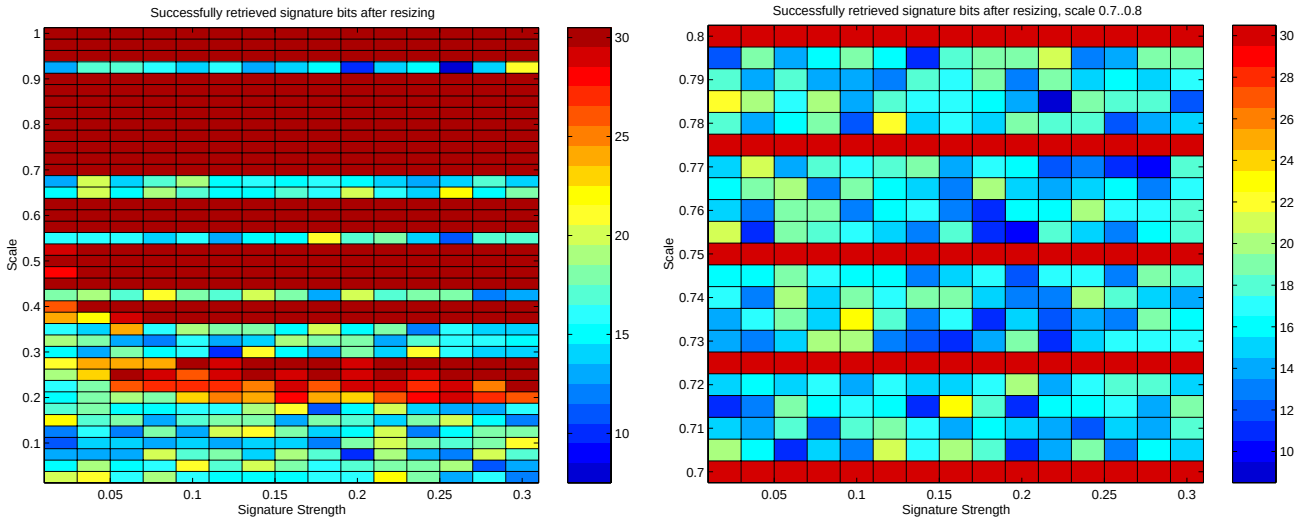


Figure 17: Robustness to image resizing

Figure 17 shows the robustness of the watermark to image resizing. The step size for the scale factor was 0.025 in the left graph and 0.005 in the right graph. It can be seen that the watermark is only really robust to resizing for scale factors that are an integer multiple of 0.025. The reason for this is probably cohering with the interpolation method used for this experiment (nearest neighbor), but this assumption could not be confirmed.

Example:
The watermarked image is scaled by the factor 0.4. Then the scaled image is resized by the factor 2.5. The result is an image of the size of the original image which is depicted in Figure 18. This image is then used to retrieve the signature. Figure 19 shows the quality of the successful retrieved signature.

Figure 18: The image, scaled back to original size

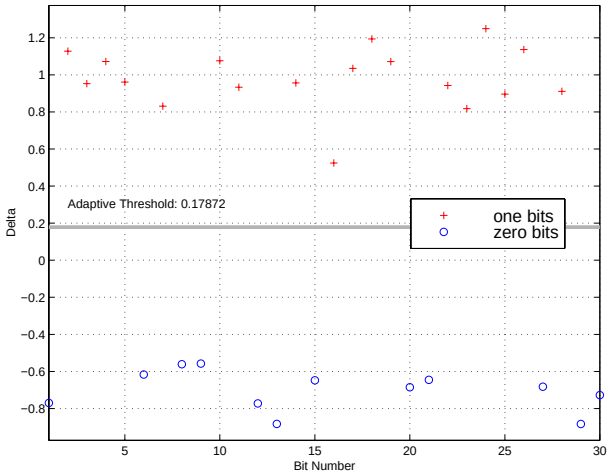


Figure 19: The retrieved average differences

4.2 Filter Attacks

In this section, the watermark is exposed to both linear and nonlinear filters.
The linear filters are: blurring, sharpening. Both filters are realized in the pixel domain, i.e. the watermarked image is convolved with a 2 dimensional filter mask.
The nonlinear filters are all order filters. These are: minimum filter, median filter, maximum filter.
During these experiments the watermark showed robustness to all the discussed filters. It is very robust to median filtering, and absolutely robust to sharpening.

4.2.1 Blurring

The blurring filters used are all averaging filters with sizes n=2..7, realized in Matlab® as `blurfilt=1/n^2*ones(n,n)`. The results of the tests are shown in Figure 20.

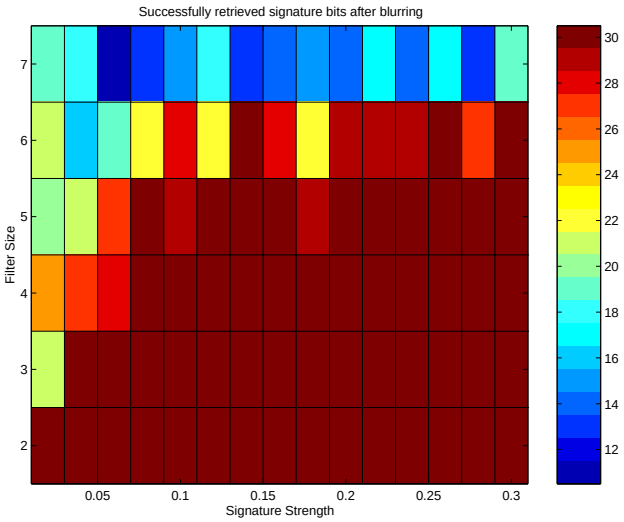


Figure 20: Robustness to blurring

Example:
The watermarked image is blurred by a 4x4 averaging filter. The blurred image is shown in Figure 21, and the quality of the watermark embedded in the blurred image is depicted in Figure 22.

Figure 21: The blurred watermarked image

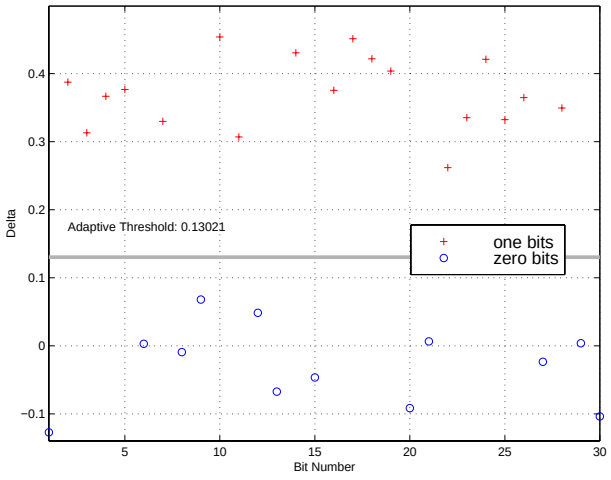


Figure 22: The retrieved average differences

4.2.2 Minimum Filter

The minimum filters used for this experiment are all order filters with sizes n=2..6. The results of the tests are shown in Figure 23.

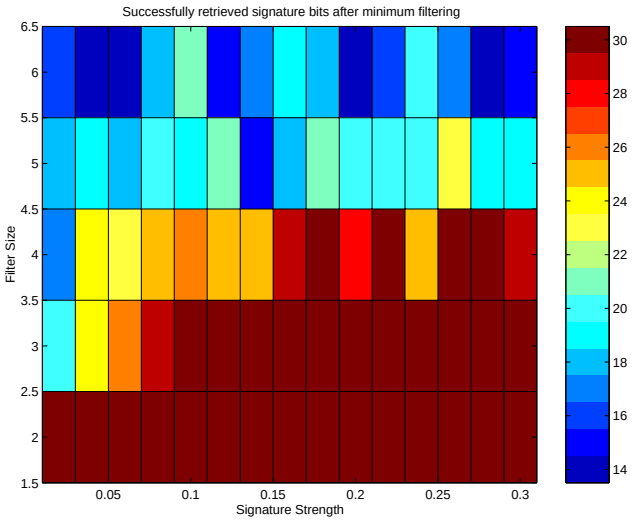


Figure 23: Robustness to minimum filtering

Example:
The watermarked image is filtered by a 3x3 minimum filter. The filtered image is shown Figure 24, and the quality of the watermark embedded in the filtered image is depicted in Figure 25.

Figure 24: The filtered watermarked image

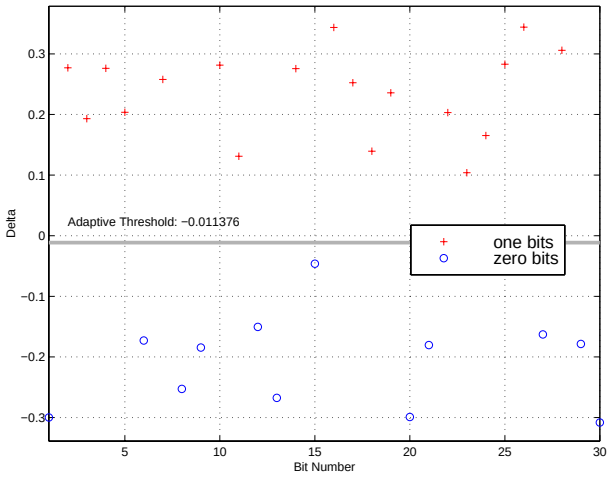


Figure 25: The retrieved average differences

4.2.3 Median Filter

The median filters used for this experiment are all order filters with sizes n=2..8. The results of the tests are shown in Figure 26.

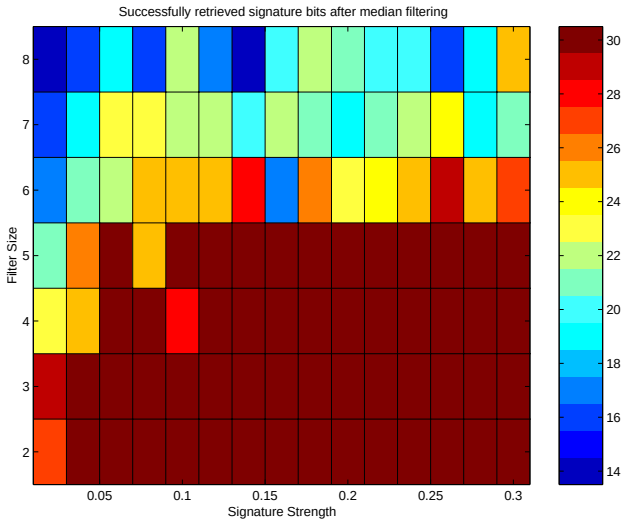


Figure 26: Robustness to median filtering

Example:
The watermarked image is filtered by a 5x5 median filter. The filtered image is shown Figure 27, and the quality of the watermark embedded in the filtered image is depicted in Figure 28.

Figure 27: The filtered watermarked image

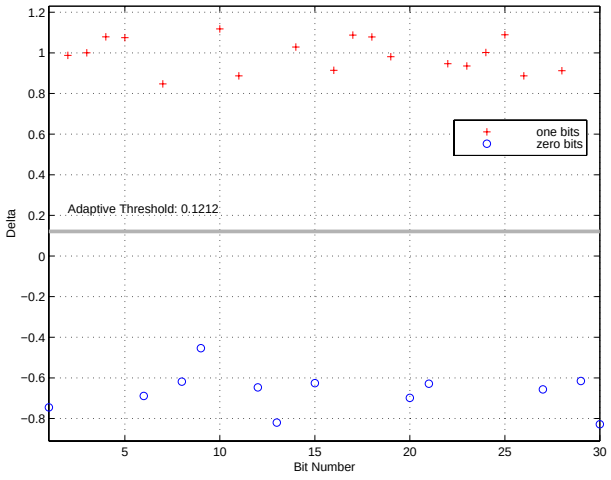


Figure 28: The retrieved average differences

4.2.4 Maximum Filter

The maximum filters used for this experiment are all order filters with sizes n=2..6. The results of the tests are shown in Figure 29.

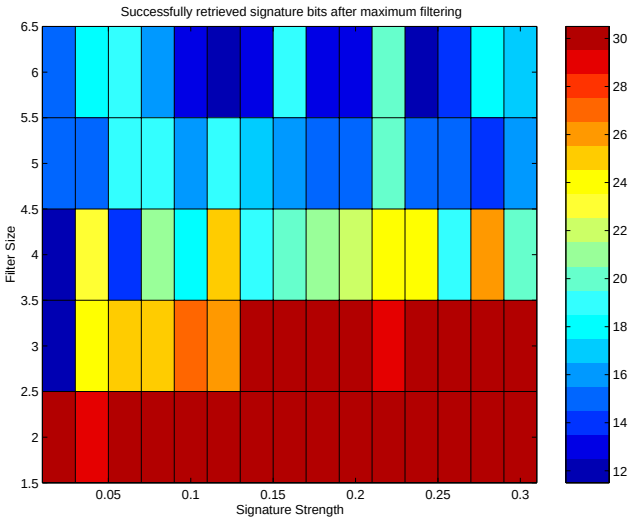


Figure 29: Robustness to maximum filtering

Example:
The watermarked image is filtered by a 2x2 maximum filter. The filtered image is shown Figure 30, and the quality of the watermark embedded in the filtered image is depicted in Figure 31.

Figure 30: The filtered watermarked image

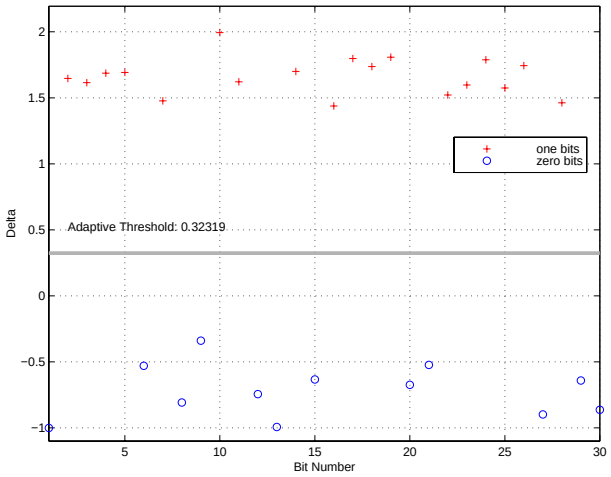


Figure 31: The retrieved average differences

4.2.5 Sharpening

The sharpening filter used for this experiment is the weighted sum of the original image and a high-pass filtered version of it. The HPF part is weighted with α and the original part is weighted with (1-α). α corresponds to the sharpening strength and lies between 0 and 1. After adding both parts the pixel values below 0 and over 255 are limited to 0 and 255, respectively.
The watermark is very robust to sharpening for all values of α. As a matter of fact, the quality of the signature is even enhanced, i.e. the retrieved average differences are bigger than the ones retrieved from the original watermarked image. That means, that even for a very low signature strength *q* the watermark would still be very robust.

Example:
The watermarked image is sharpened using the strength α=0.5. The sharpened image is depicted in Figure 32, and the quality of the signature embedded in the sharpened image can be seen in Figure 33.

Figure 32: The sharpened watermarked image

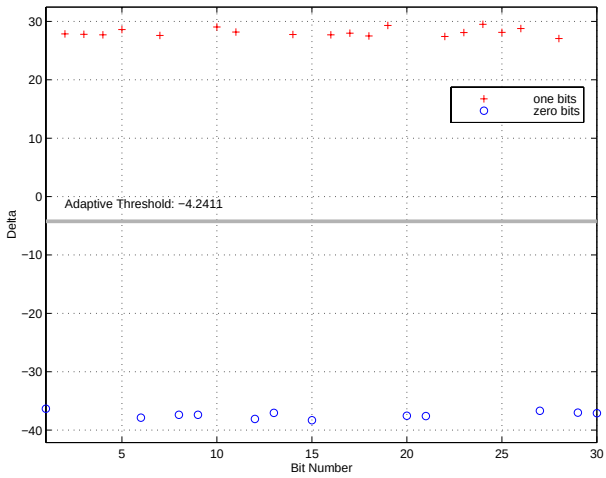


Figure 33: The retrieved average differences

4.3 Noise Attacks

In this series of experiment the watermark was exposed to 3 different kinds of noise contamination. That is: gaussian noise, uniform noise, and salt & pepper noise. The watermark showed extreme robustness to all these kinds of noise contamination. The image can very noisy, so noisy, that it is hard to recognize the image, and the signature can still be correctly received.
In all three cases, the noise contamination was realized using the Matlab® command `imnoise`.

4.3.1 Gaussian Noise

In this experiment the watermarked image was contaminated with zero-mean gaussian noise using various different variances between 0 and 5. The results of the tests are shown in Figure 34.

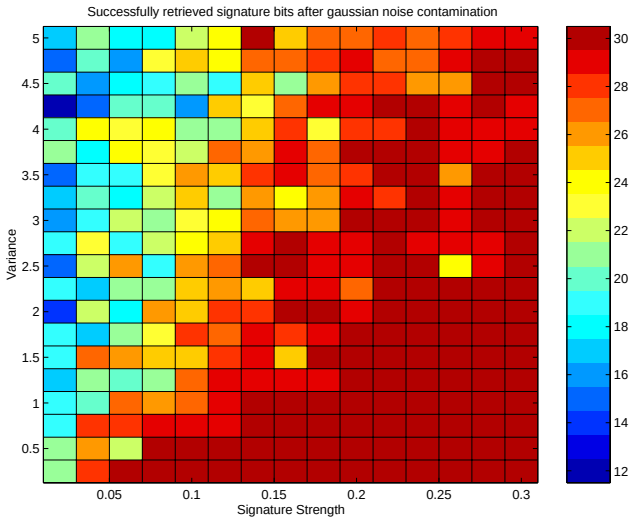


Figure 34: Robustness to gaussian noise contamination

Example:
The watermarked image is contaminated with zero-mean gaussian noise with a variance of 0.5. The noisy picture is shown in Figure 35, and the signature quality is depicted in Figure 36

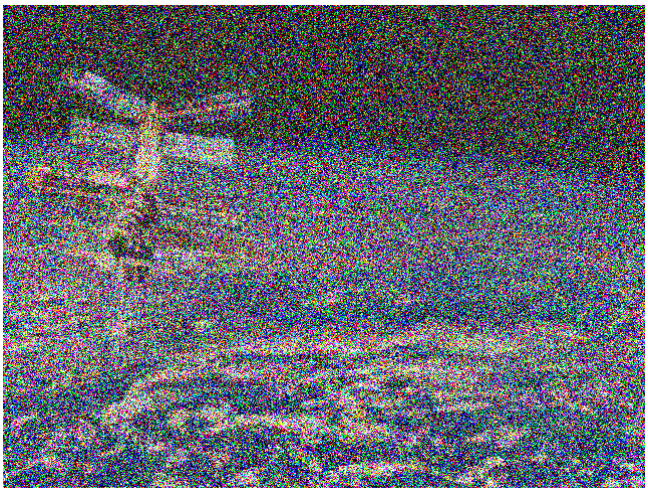


Figure 35: The noisy watermarked image

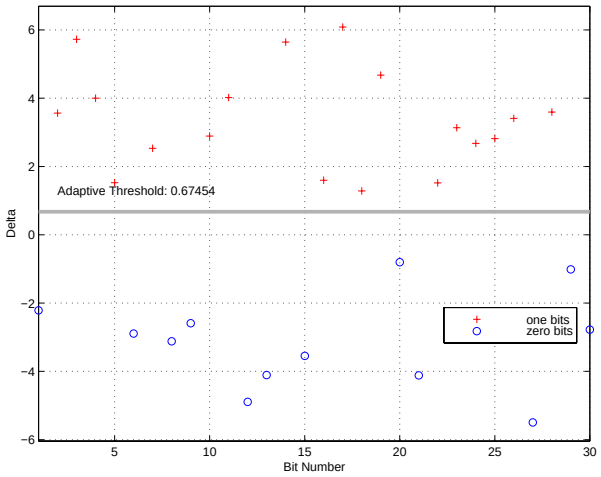


Figure 36: The retrieved average differences

4.3.2 Uniform Noise

In this experiment the watermarked image was contaminated with uniform noise using various different variances between 0 and 7. The results of the tests are shown in Figure 37.

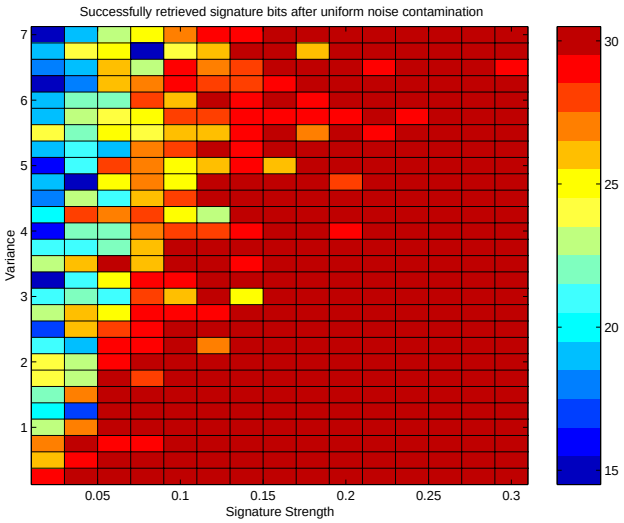


Figure 37: Robustness to uniform noise contamination

Example:
The watermarked image is contaminated with uniform noise with a variance of 2.5. The noisy picture is shown in Figure 38, and the signature quality is depicted in Figure 39

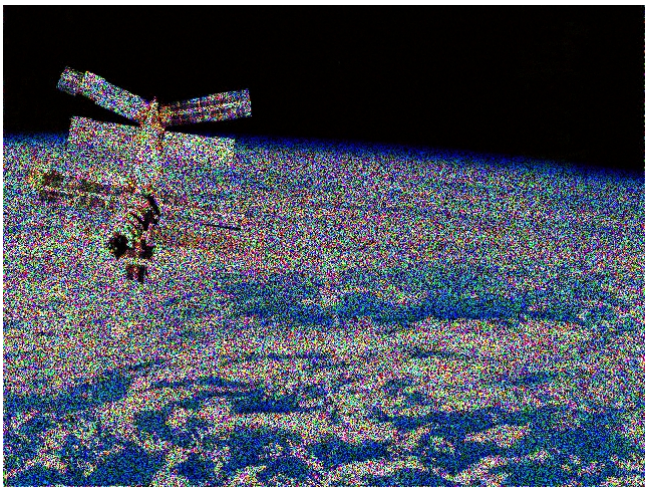


Figure 38: The noisy watermarked image

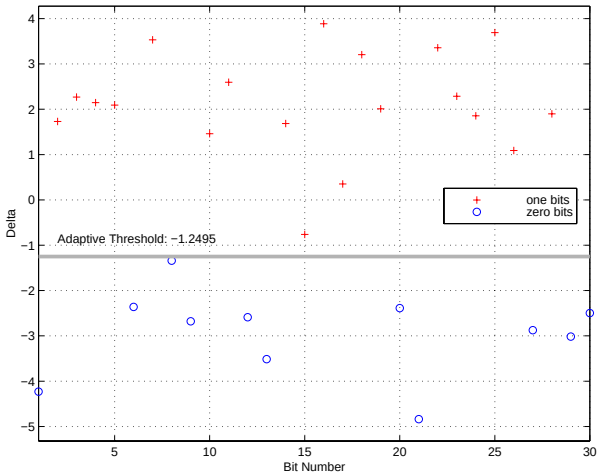


Figure 39: The retrieved average differences

4.3.3 Salt & Pepper Noise

In this experiment the watermarked image was contaminated with salt & pepper noise using various different noise densities between 0 and 1. The results of the tests are shown in Figure 40.

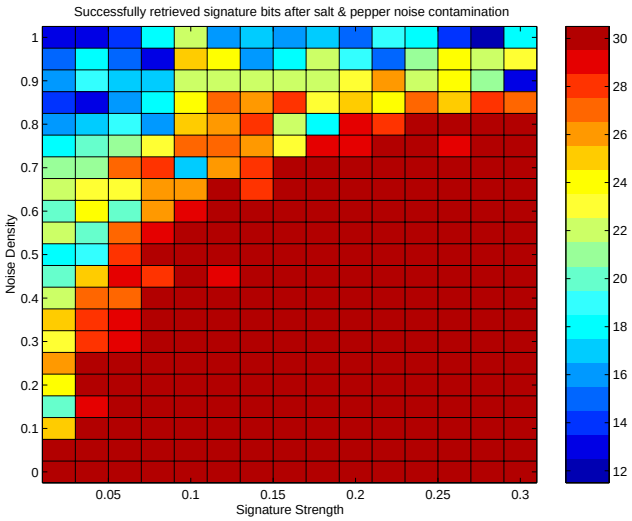


Figure 40: Robustness to salt & pepper noise contamination

Example:
The watermarked image is contaminated with salt & pepper noise with a noise density of 0.55. The noisy picture is shown in Figure 41, and the signature quality is depicted in Figure 42

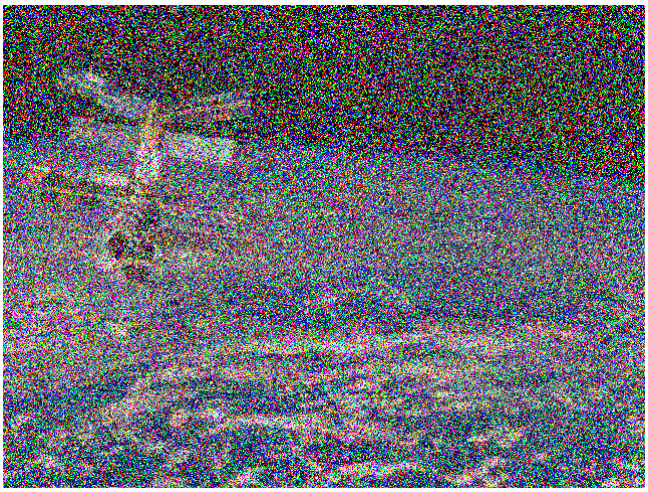


Figure 41: The noisy watermarked image

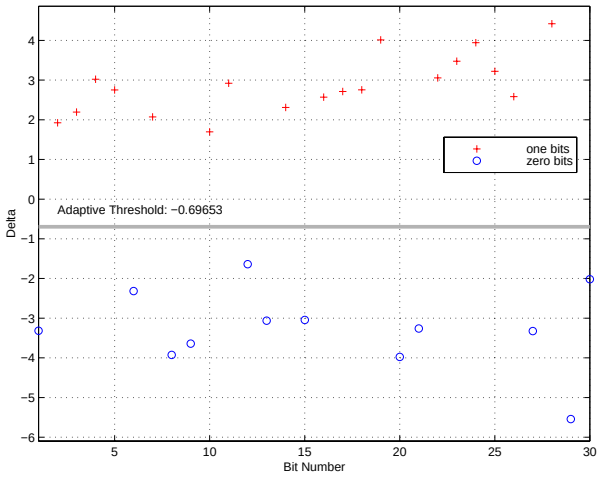


Figure 42: The retrieved average differences

4.4 Contrast Enhancement

4.4.1 Histogram Equalization

The histogram equalization was realized using the Matlab® command `histeq`, which is a function that does not require any parameters.
The watermark is very robust to histogram equalization As a matter of fact, the quality of the signature is even enhanced, i.e. the retrieved average differences are bigger than the ones retrieved from the original watermarked image. That means, that even for a very low signature strength q the watermark would still be very robust.

Example:
The watermarked image is contrast enhanced using histogram equalization. The enhanced image is depicted in Figure 43, and the quality of the signature embedded in the enhanced image can be seen in Figure 44.

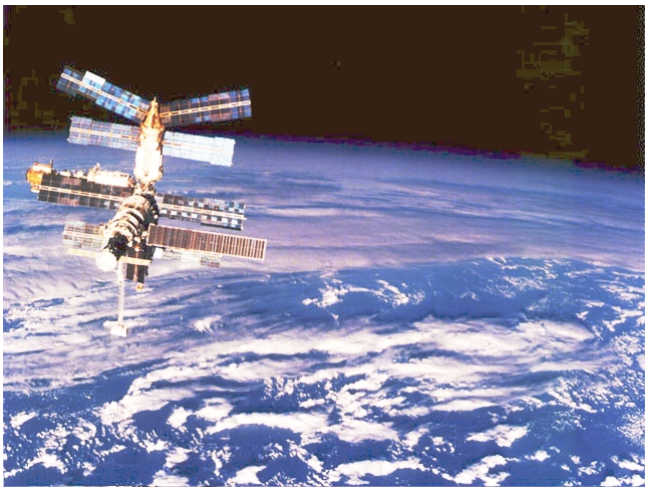


Figure 43: The enhanced watermarked image

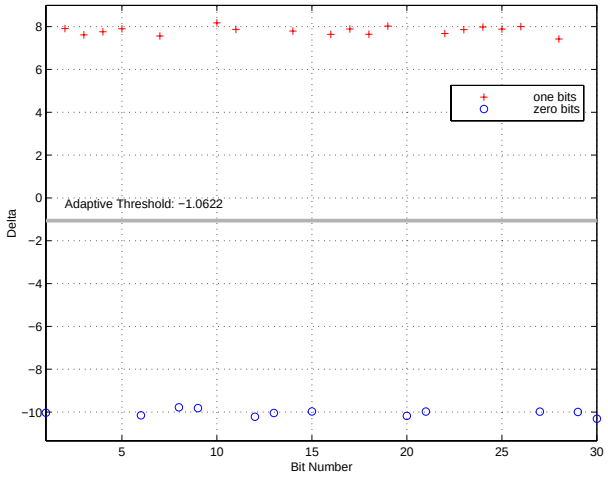


Figure 44: The retrieved average differences

4.5 Image Compression

One of the strongest requirements of a watermarking technique is robustness to image compression such as JPEG, DCT, and wavelet compression. Since implementing such compression algorithms would have exceeded the effort for this project, only robustness to JPEG compression was investigated, because this compression technique is already offered by Matlab®.

4.5.1 JPEG Compression

JPEG compression is one of the options offered by the Matlab® function `imwrite`. This command allows it to specify a value for the image quality from 0 to 100. Default value for JPEG compression is 75. Figure 45 shows the results of the tests.

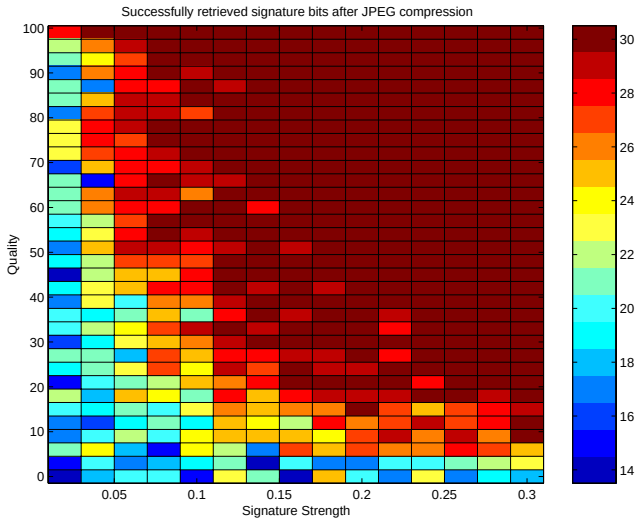


Figure 45: Robustness to JPEG compression

Example:
The watermarked image is JPEG compressed using the default quality factor 75. The compressed image is depicted in Figure 46, and the quality of the watermark embedded in the compressed image is shown in Figure 47.

Figure 46: The compressed watermarked image

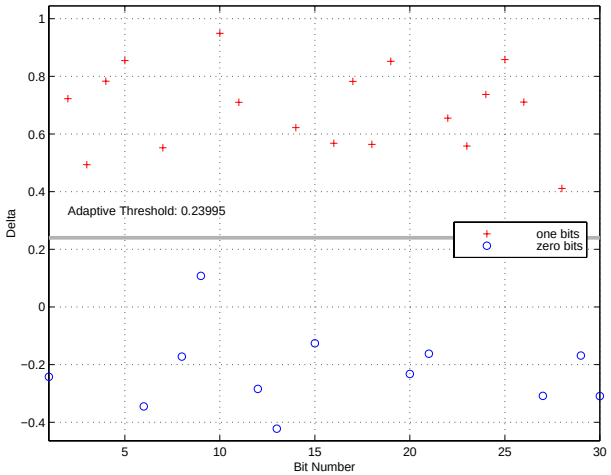


Figure 47: The retrieved average differences

5. Conclusion

It can be said, that blue channel amplitude modulation method is a very good technique for image watermarking. The signature is invisible to the human eye for an embedding strength of q=0.1, which has been found to be a very good choice after carrying through the experiments discussed in chapter 4. It has also been demonstrated that for this embedding strength the signature is immune to a variety of attacks, including filtering, contrast balancing, compression, and geometrical transforms. The resistance to the latter kind of attack without the need for the original image is the major advantage of this method.

The introduced technique could be improved in several ways. First, all three color channels could be used for signature embedding. The embedding strength in each channel would be proportional to the sensitivity of the human eye to it. Then, robustness could be further improved by using error correcting codes for the signature.

6. References

[1] M. Kutter, F. Jordan, and F. Bossen. Digital signature of color images using amplitude modulation. Signal Processing Laboratory, EPFL Switzerland, 1997

Appendix

Appendix A – The Software accompanying the Report

The files on the CD-ROM

The CD-ROM accompanying this report contains folders for each of the labs in ELE585 and a folder called `project`. This is the folder containing all the project files.

Contents of the Folder `'project'`:

Files:

- `start.m`	Main Menu

Folders:

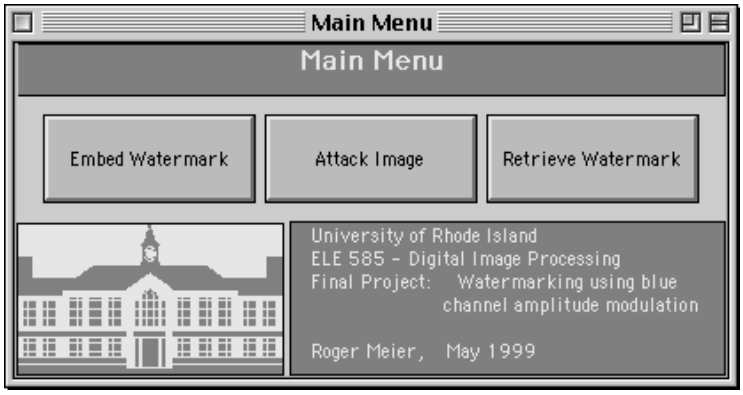
- `experiments`	Experiments for chapter 4
- `pics`	Image files
- `wmtoolbox`	Watermarking Toolbox
- `doc`	Documentation
- `gui`	Watermarking tools with graphical User interface

Getting Started

Start Matlab® and make the directory containing the m-file `start.m` your current working directory and execute the m-file `start.m`.
After the execution of this file, the current path and the path of the watermarking toolbox have been added to the Matlab® path.

The Main Menu

After the execution of `start.m` the main menu opens:



The tools are all operated through the graphical user interface, and should not need much explanation.

The Watermarking Toolbox

The command `help wmtoolbox` displays a list of all the functions of the watermarking toolbox. If `help` is used with a toolbox function, a brief explanation of the function is displayed.