Hogeschool Antwerpen
IW: Elektronica-ICT
Bachelor thesis 'Versatile Digital Watermarking Techniques Implemented In Matlab'

Master students

Heylen Kevin (heylen.kevin@student.ha.be)
Meesters Tom (meesters.tom@student.ha.be)
Verstrepen Luc (verstrepen.luc@student.ha.be)

Promoter
Tim Dams (t.dams@ha.be, tdams@etro.vub.ac.be)

With the help of
Christiaensen Tom
Vloeberghs Wouter
Van Ginneken Gregory

E-Wat User Guide

How do the techniques work and how do you use the GUI's for embedding, attacking and detecting?

Contents

1 Introduction 5
2 Overview 6
3 Main GUI 7
4 Watermarking GUI 11
5 LSB with key 12
5.1 LSB with key: technique 12
5.1.1 Embedding 12
5.1.2 Detecting 12
5.2 Embedding GUI 13
5.2.1 Step 1: Adjust the watermark 14
5.2.2 Step 2: Choose bitplane(s) 15
5.3 Detecting GUI 16
6 LSB without key 17
6.1 LSB without key: technique 17
6.1.1 Embedding 17
6.1.2 Detecting 17
6.2 Embedding GUI 18
6.2.1 Step 1: Adjust the watermark 19
6.2.2 Step 2: Embed the watermark 20
6.3 Detecting GUI 21
7 Patchwork 22
7.1 Patchwork technique 22
7.1.1 Embedding 22
7.1.2 Detecting 22
7.2 Embedding GUI 23
7.2.1 Adjust the watermark 24
7.2.2 Step 2: Embed the watermark 25
7.3 Detecting GUI 26
8 Correlation 27
8.1 Correlation technique 27
8.1.1 Embedding 27
8.1.2 Detecting 27
8.2 Embedding GUI 28

8.2.1 Step 1: Adjust the blocksize of the bit patterns 29
8.2.2 Step 2: Embed the watermark 30
8.3 Detecting GUI 31
9 DCT 32
9.1 DCT technique 32
9.1.1 Embedding 32
9.1.2 Detecting 32
9.2 Embedding GUI 33
9.2.1 Step 1: Add the message 34
9.2.2 Step 2: Embed the watermark 35
9.3 Detecting GUI 36
10 DWT 37
10.1DWT technique 37
10.1.1 Embedding 37
10.1.2 Detecting 37
10.2 Embedding GUI 38
10.2.1 Step 1: Adjust the DWT decomposition 39
10.2.2 Step 2: Add the message 40
10.2.3 Step 3: Embed the watermark 41
10.3 Detecting GUI 42
11 DS-CDMA 43
11.1 DS-CDMA technique 43
11.1.1 Embedding 43
11.1.2 Detecting 43
11.2 Embedding GUI 44
11.2.1 Embed the watermark 45
11.3 Detecting GUI 46
12 QIM 47
12.1 QIM technique 47
12.1.1 Embedding 47
12.1.2 Detecting 47
12.2 Embedding GUI 48
12.2.1: Step 1: Show the embedding process 49
12.2.2 Step 2: Embed the watermark 50

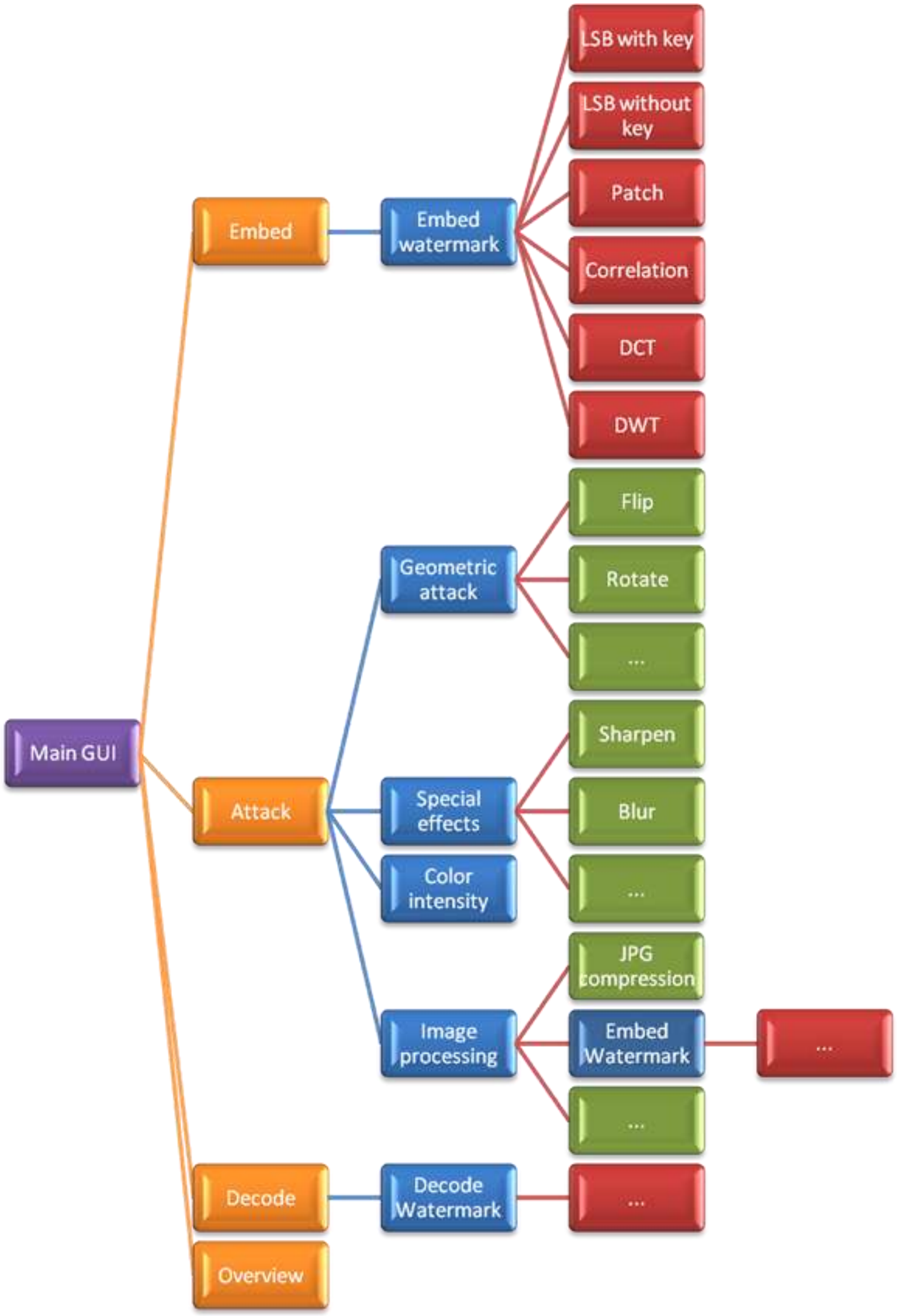
12.3 Detecting GUI 51
13 Attack GUI 52
13.1 Compression attacks 53
13.2 Enhancements 54
13.3 Color manipulations 55
13.4 Noise attacks 56
13.5 Geometric distortions 57
13.6 Data compositions 58
13.7 Multiple watermarks 59
13.8 Other attacks 60
14 Overview GUI 61
15 Zoom functions 62

1 Introduction

This is the guide to the watermarking GUI that was developed during the bachelor thesis ‘Versatile Digital Watermarking Techniques Implemented in Matlab’ by Kevin Heylen, Tom Meesters and Luc Verstrepen, students IW: Electronics-ICT at the University College of Antwerp. This guide gives an overview of the structure of all GUI’s and how the different settings can be configured. First the main GUI will be explained, then the different watermarking techniques, followed by the attacking GUI and finally the detecting GUI.

2 Overview

The following figure illustrates the structure of the GUI.



3 Main GUI

After an image is loaded, e.g. peppers.gif, this image is shown beneath the title Image original (Figure 1). Each file type or format of images can be read, but be aware that your chosen image will be rescaled to a compatible format. The maximum possible deviation is 16 rows and/or columns.

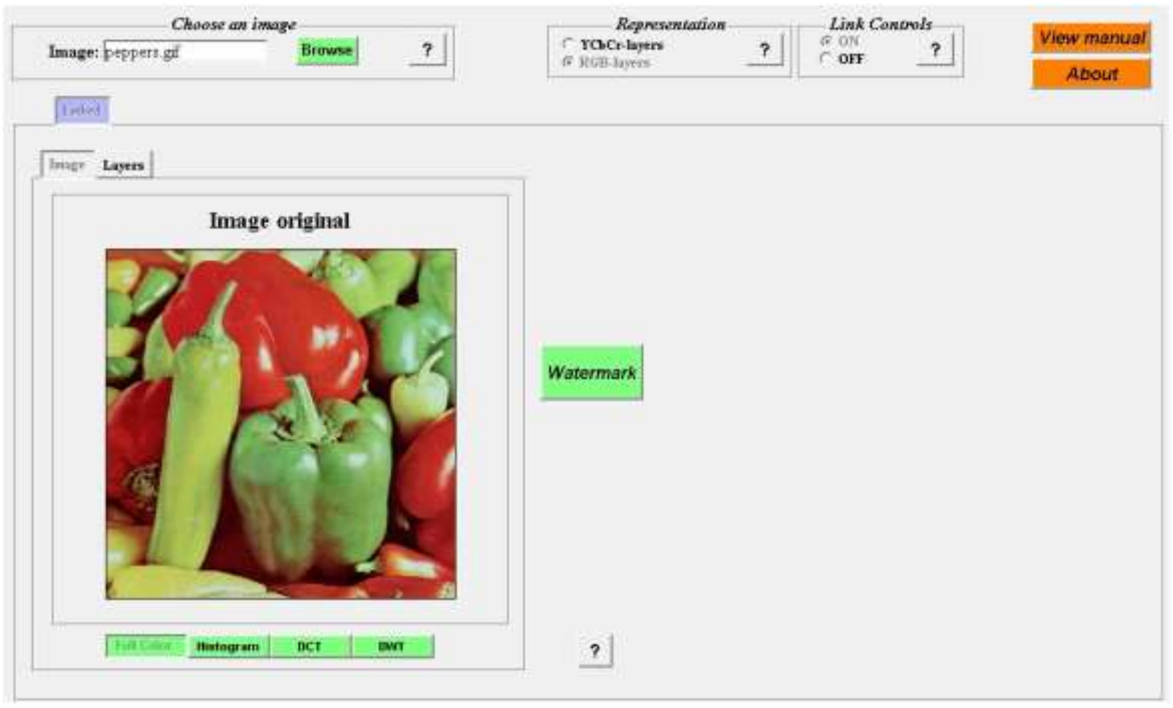


Figure 1: Main GUI

On top of the image the user has the choice between Image and Layers. In figure 2 the different layers R,G and B of the image peppers.gif are shown.

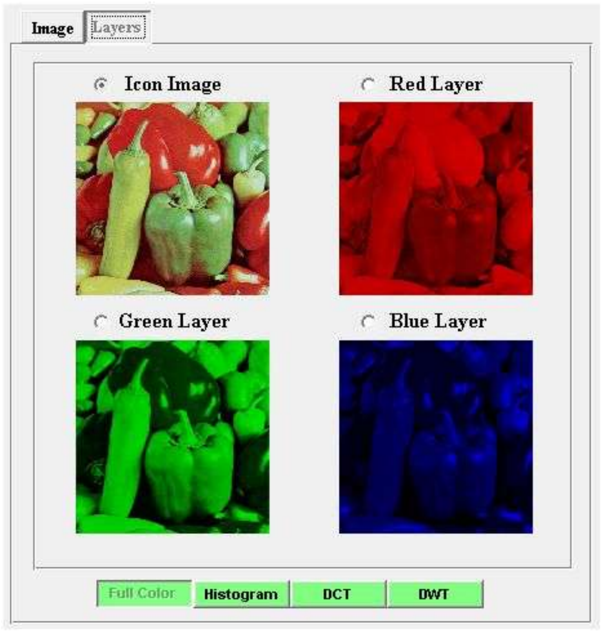


Figure 2: RGB-layers

The histogram of the image and each layer is shown after clicking the button Histogram beneath the image. In figure 3 the histograms of the image and the layers R,G and B are shown.

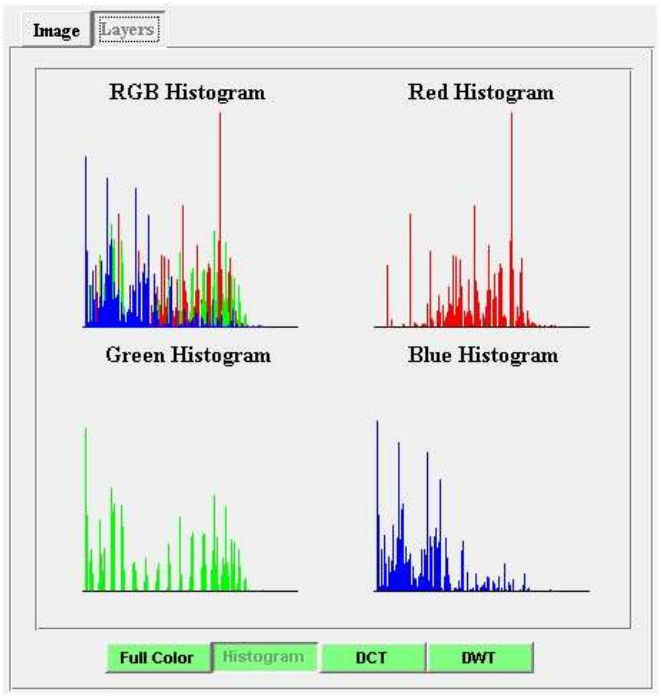


Figure 3: RGB-histograms

The DCT decomposition is shown for each layer of the image after clicking the button DCT beneath the image. In figure 4 the different DCT coefficients of the layers R,G and B are shown.

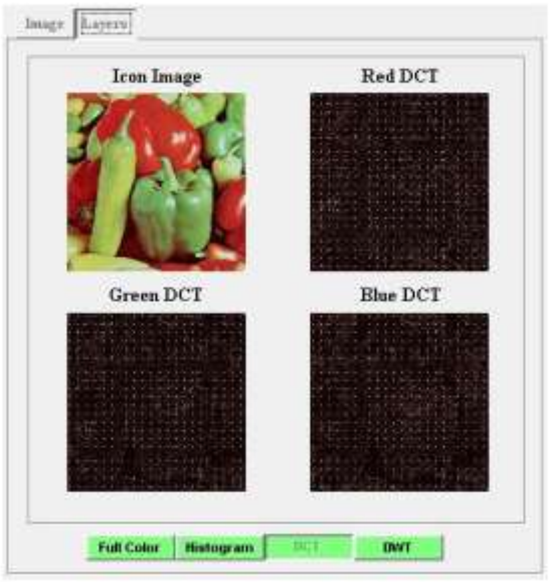


Figure 4: RGB - DCT

The DWT decomposition of each layer of the image is shown after clicking the button DWT beneath the image. The DWT decomposition consists of 5 levels with level 5 on the bottom right. In figure 5 the DWT decompositions of the layers R,G and B are shown.

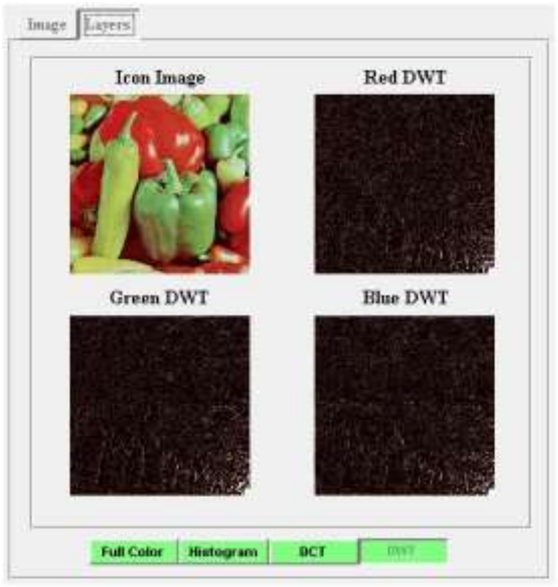


Figure 5: RGB - DWT

The same actions can be performed on the YcbCr-layers instead of the RGB-layers when the user selects the upper radiobutton in figure 6.

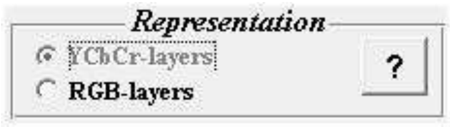
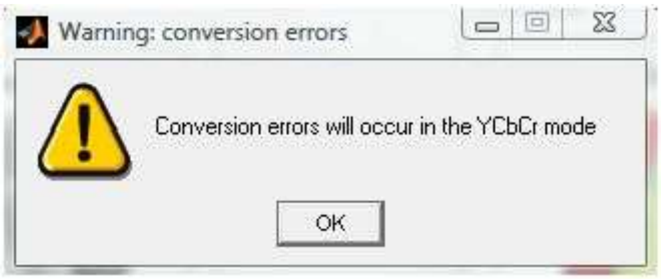


Figure 6: Image Representation

A warning message (figure 7) will appear after selecting the YCbCr representation. In this representation conversion errors will occur due to the conversion between the RGB mode and the YCbCr mode.


Warning: conversion errors

Conversion errors will occur in the YCbCr mode

Figure 7: Warning message

Further it is possible to work with linked controls. We can enable or disable this feature in figure 8.

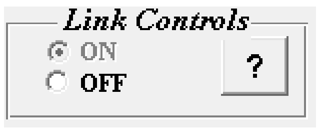


Figure 8: Link Controls

The layer which will be watermarked depends on the selected layer. In Table 1 you can find which layer will be passed for watermarking for each selected layer.

Selected layer	Passed layer
YCbCr	Y
RGB	Y
Y	Y
Cb	Cb
Cr	Cr
Red	Red
Green	Green
Blue	Blue

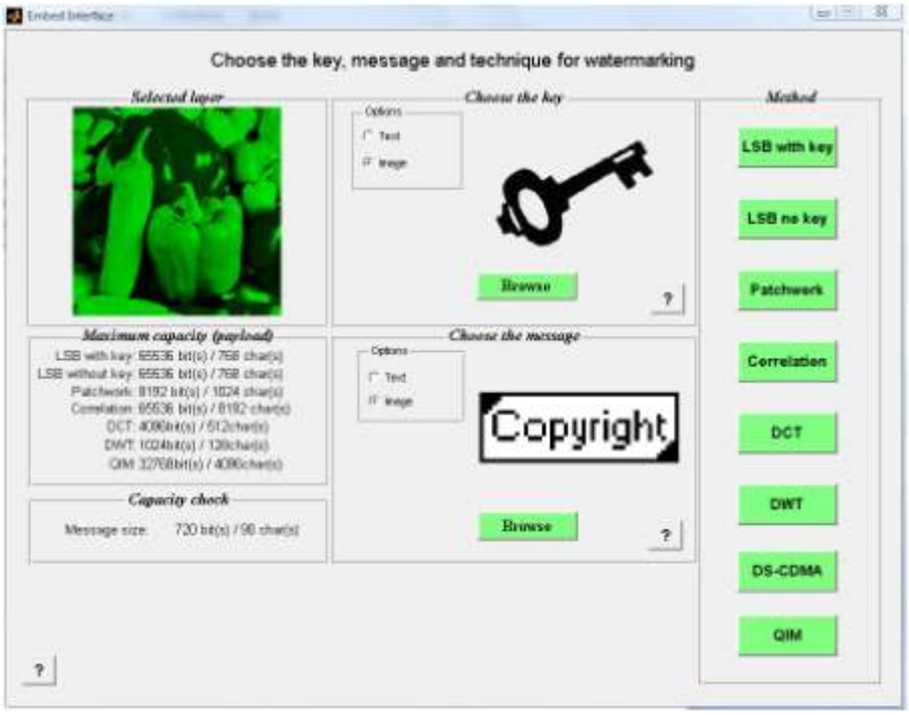
Table 1: Passed layer

To start watermarking, click on the button Watermark (figure 9).

Figure 9: Watermark

4 Watermarking GUI

In this GUI (figure 10) the user can specify the key, message and watermarking technique. On the left the passed layer of the selected image is shown. In the center of the image the user can choose the key and message. To determine which techniques support the chosen message, a capacity check is calculated on the bottom left. Finally the user can proceed with the preferred watermarking technique by clicking the corresponding button on the right.



Maximum capacity (payload)

LSB with key: 65536 bit(s) / 768 char(s)
LSB without key: 65536 bit(s) / 768 char(s)
Patchwork: 8192 bit(s) / 1024 char(s)
Correlation: 65536 bit(s) / 8192 char(s)
DCT: 4096bit(s) / 512char(s)
DWT: 1024bit(s) / 128char(s)
QIM: 32768bit(s) / 4096char(s)

Capacity check

Message size: 720 bit(s) / 90 char(s)

Figure 10: Watermarking GUI

5 LSB with key

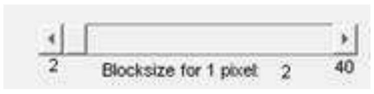
5.1 LSB with key: technique

In Least Significant Bit Watermarking the watermark is placed in the least significant bits of the pixel values. In a grayscale image each pixel has a decimal value between 0 and 255, and a binary value between 0000 0000 and 1111 1111. We obtain the first bitplane by grouping the first bits of each pixel in a matrix. By repeating this process for all bits, 8 matrices are constructed which represent the 8 bitplanes. The first bitplane consists of the most significant bits and shows the main features and lines of the original image. The eighth bitplane consists of the least significant bits and looks like a noise pattern. It is possible to place a watermark in each bitplane. However, the watermark will be clearly visible in the first bitplane. Therefore we place the watermark by default in the least significant bitplane, bitplane 8. Because this bitplane only contains the smallest details of the image, the watermark will be almost inperceptible.

5.1.1 Embedding

In LSB watermarking with key the watermark is determined by the message and key. This message can be a text or image. When the user chooses a text file or types a text as message, each character of the text is converted to an unique pattern. The key is used as seed for the Pseudo Random Number Generator (PRNG), which generates the patterns. To construct the watermark, the characters of the message are replaced by their correspondering pattern. These character patterns have a default length of 16x21 pixels, but can be dynamically enlarged depending on the length of the message. The maximum length of the patterns is based on the size of the cover work. If the message is too long for the cover work, only a part of the message will be used for the watermark. The watermark is formed by the composition of all the character patterns. Finally the chosen bitplane is replaced by the watermark.

If the message is an image, the image is first converted to a binary image. Then two patterns are generated based on the key: one pattern to represent the bit 1 and one pattern to represent the bit 0. The size of the patterns can be changed by the slider ‘Blocksize’.



To construct the watermark, the bits of the binary message image are replace bit-by-bit by their corresponding pattern. The watermark is formed by the composition of all the bit patterns. Finally the chosen bitplane is replaced by the watermark.

5.1.2 Detecting

At the detector the character patterns and the bit patterns for 0 and 1 are reconstructed. The detector then calculates the correlation between the matrices from the watermarked image and the character matrices and bit patterns. The highest correlation indicates if the recovered message will be a text or an image. By calculating and comparing the correlations for all character matrices or bit patterns, the recovered message can be constructed.

5.2 Embedding GUI

In figure 11 the overview of the embedding GUI is shown. The selected layer of the original image is shown on the left top. At startup this image is automatically decomposed in bitplanes. Furthermore the watermark is calculated with a default blocksize of 2.

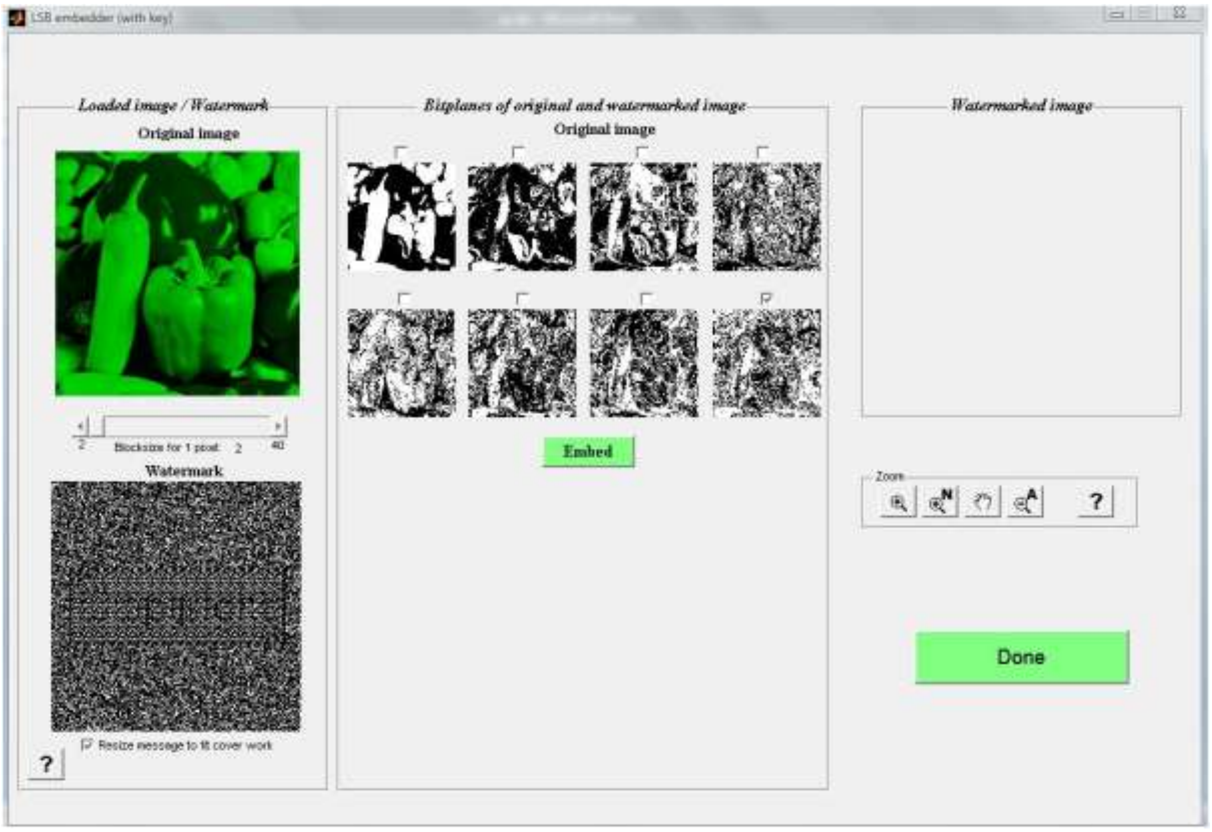


Figure 11: LSB with key embedding

5.2.1 Step 1: Adjust the watermark

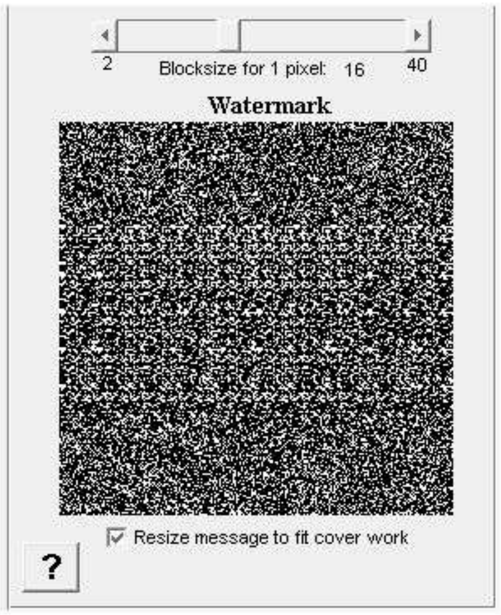


Figure 12: Adjust blocksize LSB with key

The watermark is constructed by replacing the characters or bits of the message by their corresponding patterns. When the message is an image, the size of the bit patterns can be adjusted by the slider ‘Blocksize’. By default the blocksize of the patterns equals 2. This means that each pixel of the message will be represented by a 2x2 block (4 pixels). In figure 12 the blocksize of the patterns is set at 16. By increasing the blocksize, the message will be larger. If the blocksize is too high and the message image does not fit anymore in the cover work, the checkbox beneath the watermark can be checked. Then the message will be rescaled to fit in the cover work. This implicates that the selected image will be adapted, resulting in loss of information. When the checkbox is checked out, the message will be cut when the blocksize for the message is too high for the cover work. Only the upper left part of the message will then be embedded. To prevent white space at the edges of the image, first a random noise pattern is generated with the same dimensions as the cover work. The character or bit patterns of the message are pasted on top of this noise pattern for embedding, resulting in the watermarked image.

5.2.2 Step 2: Choose bitplane(s)

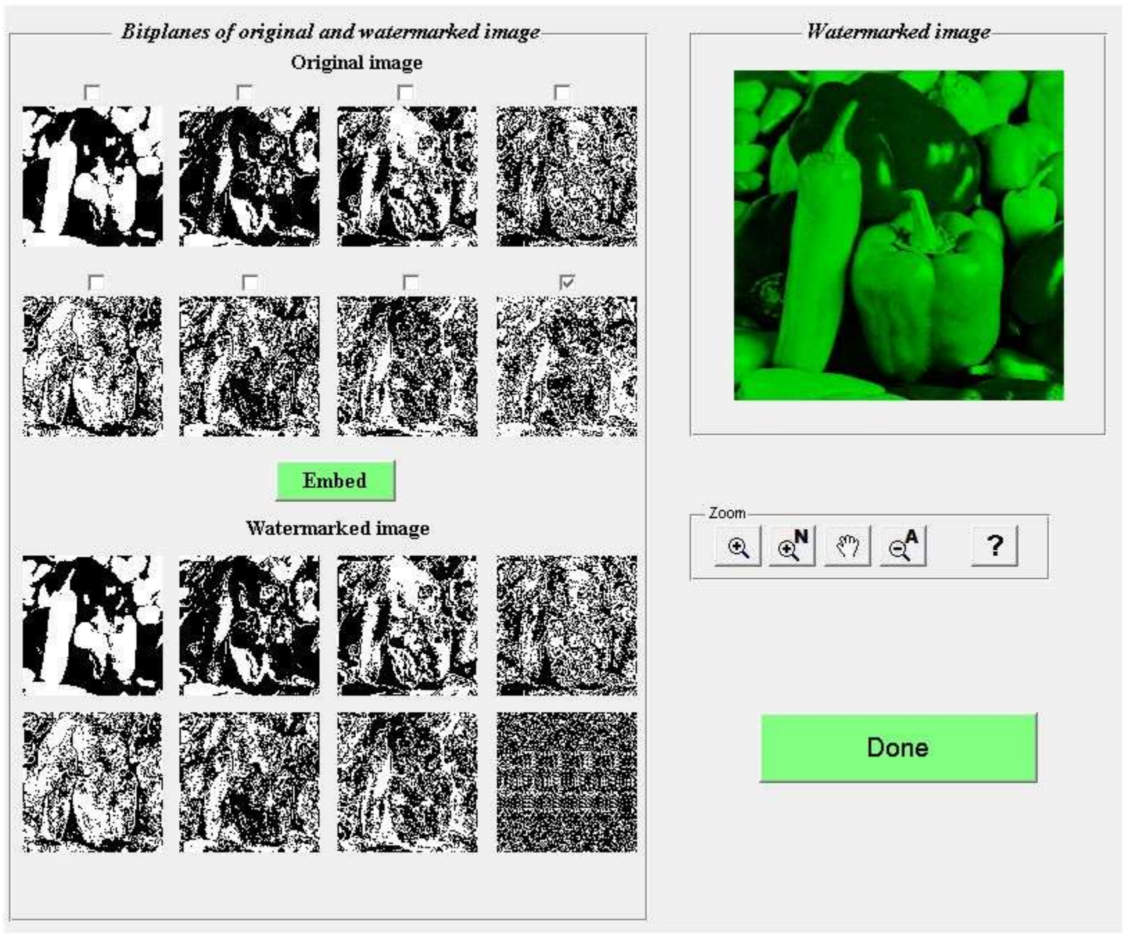


Figure 13: LSB with key bitplanes

The user can choose which bitplane(s) to watermark. In figure 13 the watermark is placed only in bitplane 8 (Least Significant Bitplane). After clicking the button Embed the original values of this bitplane are replaced by the watermark. The watermarked image is shown on the right. With the zoom functions of the zoom panel the user can zoom in and out on each bitplane or image. After clicking on the button Done the GUI is closed and the watermarked image can be further processed.

5.3 Detecting GUI

At startup of the detector GUI the bitplanes of the watermarked image are calculated and shown in figure 14.

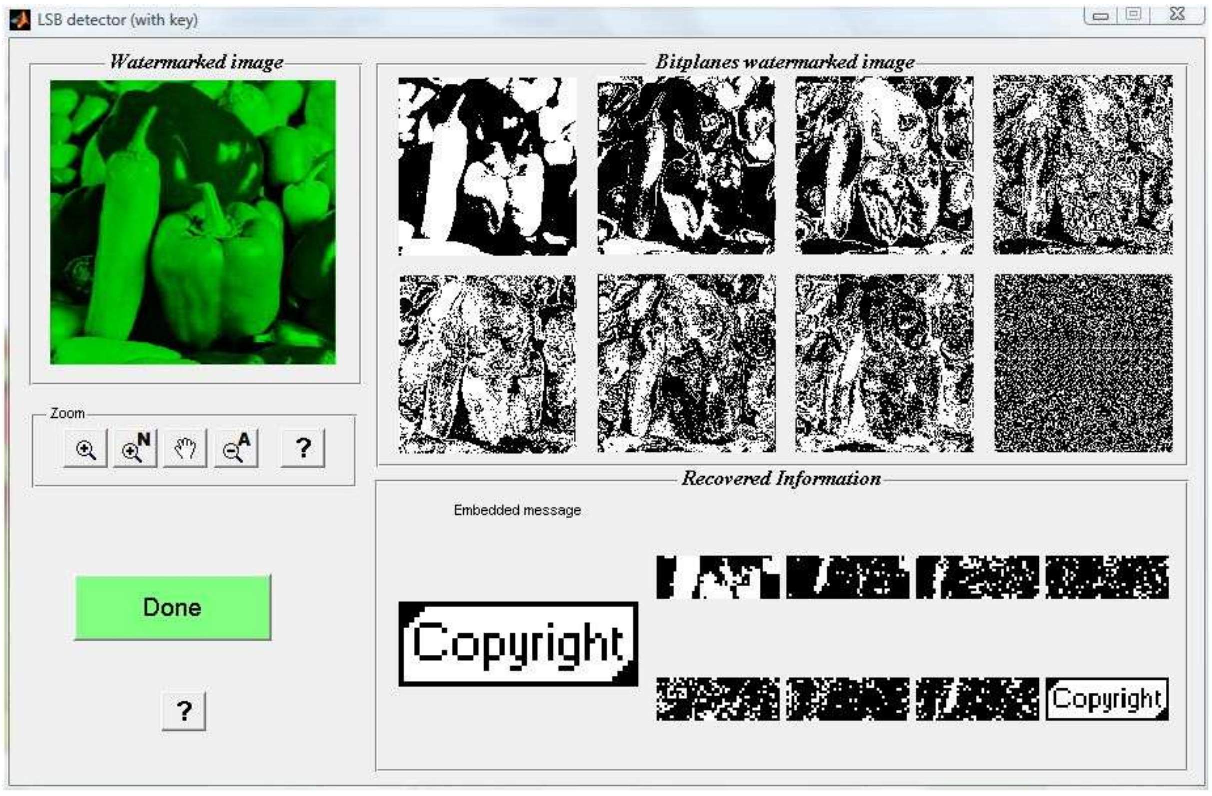


Figure 14: LSB with key decoding

Then the character patterns and bit patterns are reconstructed. The detector will now calculate the correlations between the bitplanes and the character patterns/bit patterns, depending on the type of message. The user can now compare the embedded message with the recovered message in the bitplane(s). By clicking on the button Done, the user proceeds to the Main GUI.

6 LSB without key

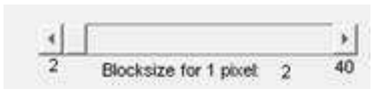
6.1 LSB without key: technique

In Least Significant Bit Watermarking the watermark is placed in the least significant bits of the pixel values. In a grayscale image each pixel has a decimal value between 0 and 255, and a binary value between 0000 0000 and 1111 1111. We obtain the first bitplane by grouping the first bits of each pixel in a matrix. By repeating this process for all bits, 8 matrices are constructed which represent the 8 bitplanes. The first bitplane consists of the most significant bits and shows the main features and lines of the original image. The eighth bitplane consists of the least significant bits and looks like a noise pattern. It is possible to place a watermark in each bitplane. However, the watermark will be clearly visible in the first bitplane. Therefore we place the watermark by default in the least significant bitplane, bitplane 8. Because this bitplane only contains the smallest details of the image, the watermark will be almost inperceptible.

6.1.1 Embedding

In LSB watermarking without key the watermark is determined only by the message. This message can be a text or image. When the user chooses a text file or types a text as message, each character of the text is converted to an unique pattern, which visually represents the character. To construct the watermark, the characters of the message are replaced by their correspondering visual pattern. These character patterns have a default length of 16x21 pixels, but can be dynamically enlarged depending on the length of the message. The maximum length of the patterns is based on the size of the cover work. If the message is too long for the cover work, only a part of the message will be used for the watermark. The watermark is formed by the composition of all the character patterns. Finally the chosen bitplane is replaced by the watermark.

If the message is an image, the image is first converted to a binary image. Then two patterns are generated: a full white pattern to represent the bit 1 and a full black pattern to represent the bit 0. The size of the patterns can be changed by the slider ‘Blocksize’.



To construct the watermark, the bits of the binary message image are replace bit-by-bit by their corresponding pattern. The watermark is formed by the composition of all the bit patterns. Finally the chosen bitplane is replaced by the watermark.

6.1.2 Detecting

At the detector the visual character patterns and the bit patterns for 0 and 1 are reconstructed. The detector then calculates the correlation between the matrices from the watermarked image and the character matrices and bit patterns. The highest correlation indicates if the recovered message will be a text or an image. By calculating and comparing the correlations for all character matrices or bit patterns, the recovered message can be constructed.

6.2 Embedding GUI

In figure 15 the overview of the embedding GUI is shown. The selected layer of the original image is shown on the left top. At startup this image is automatically decomposed in bitplanes. Furthermore the watermark is calculated with a default blocksize of 2.

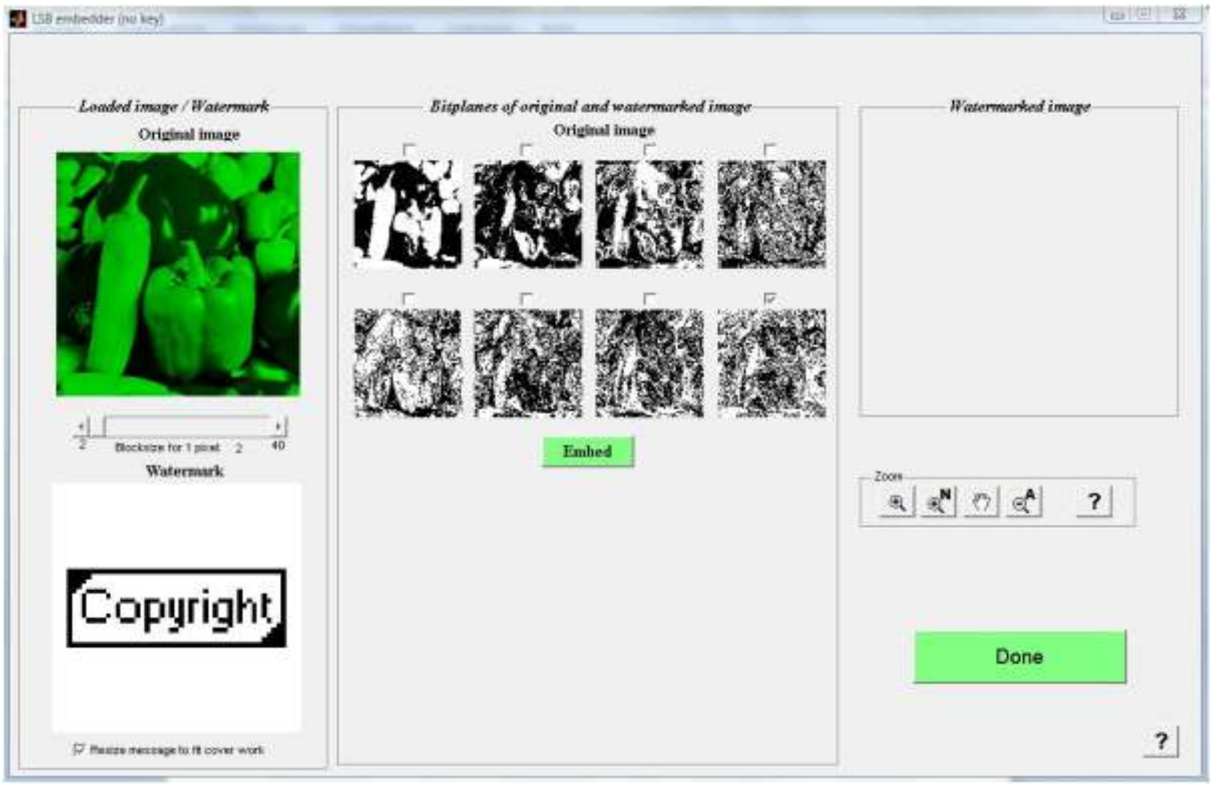
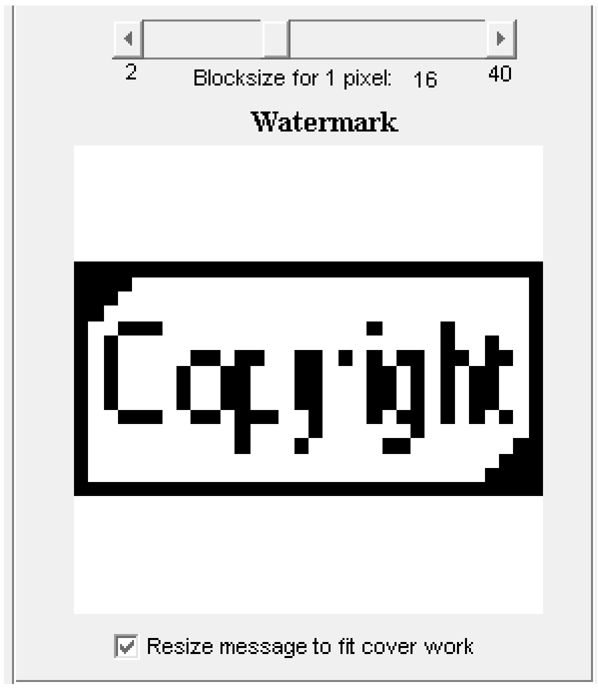


Figure 15: LSB without key

6.2.1 Step 1: Adjust the watermark



☑ Resize message to fit cover work

Figure 16: Adjust blocksize LSB without key

The watermark is constructed by replacing the characters or bits of the message by their corresponding patterns. When the message is an image, the size of the bit patterns can be adjusted by the slider ‘Blocksize’. By default the blocksize of the patterns equals 2. This means that each pixel of the message will be represented by a 2x2 block (4 pixels). In figure 16 the blocksize of the patterns is set at 16. By increasing the blocksize, the message will be larger. If the blocksize is too high and the message image does not fit anymore in the cover work, the checkbox beneath the watermark can be checked. Then the message will be rescaled to fit in the cover work. This implicates that the selected image will be adapted, resulting in loss of information. When the checkbox is checked out, the message will be cut when the blocksize for the message is too high for the cover work. Only the upper left part of the message will then be embedded. The character or bit patterns of the message are pasted on top of a white background for embedding, resulting in the watermarked image.

6.2.2 Step 2: Embed the watermark

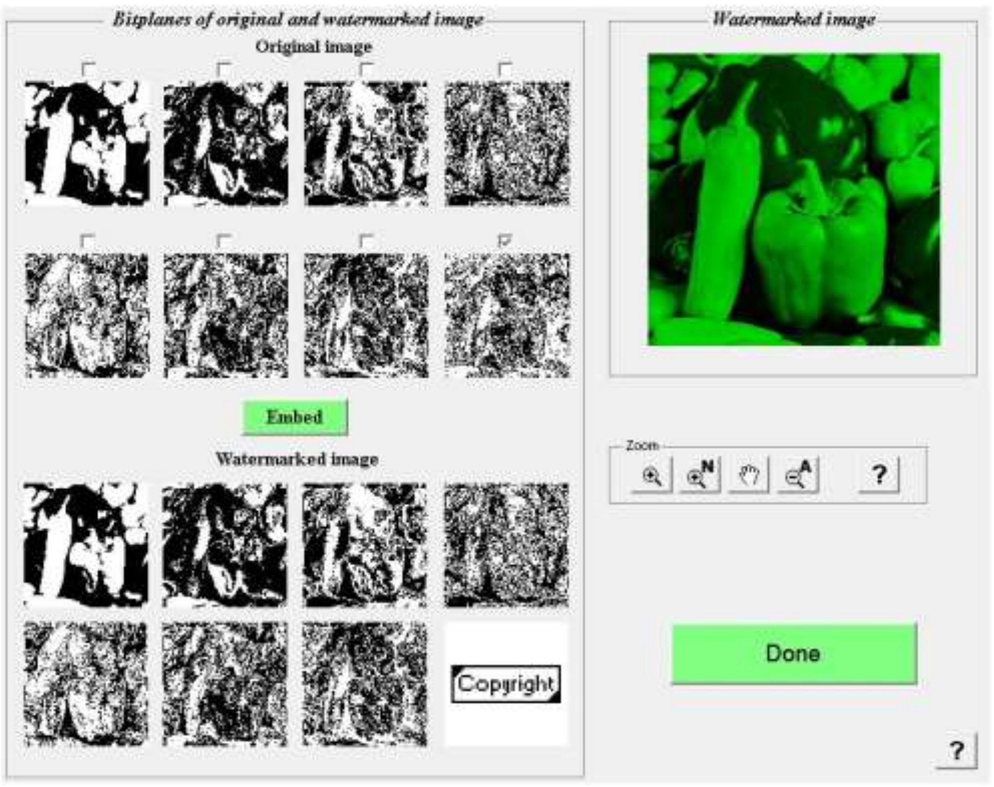


Figure 17: Overview LSB without key

The user can choose which bitplane(s) to watermark. In figure 17 the watermark is placed only in bitplane 8 (Least Significant Bitplane). After clicking the button Embed the original values of this bitplane are replaced by the watermark. The watermarked image is shown on the right. With the zoom functions of the zoom panel the user can zoom in and out on each bitplane or image. After clicking on the button Done the GUI is closed and the watermarked image can be further processed.

6.3 Detecting GUI

At startup of the detector GUI the bitplanes of the watermarked image are calculated and shown in figure 14.

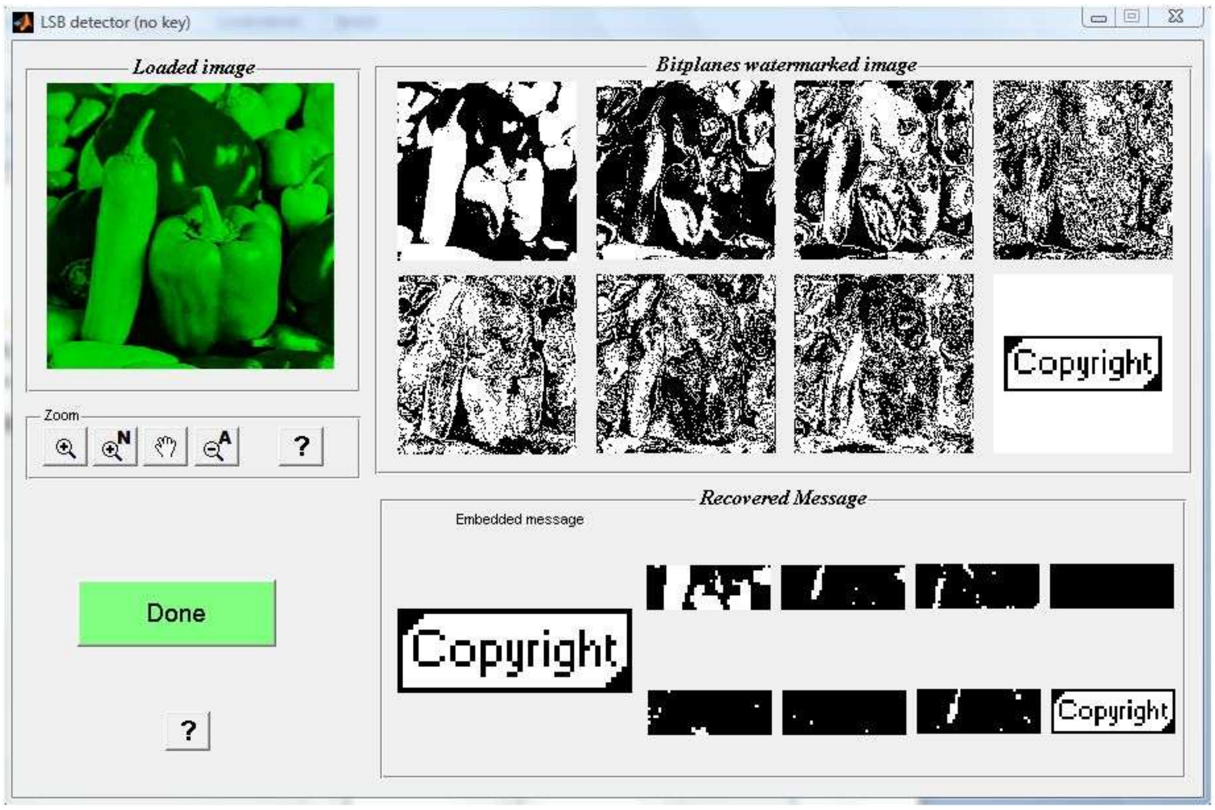


Figure 18: LSB without key decoding

Then the character patterns and bit patterns are reconstructed. The detector will now calculate the correlations between the bitplanes and the character patterns/bit patterns, depending on the type of message. The user can now compare the embedded message with the recovered message in the bitplane(s). By clicking on the button Done, the user proceeds to the Main GUI.

7 Patchwork

7.1 Patchwork technique

The patchwork technique is a simple algorithm which allows the embedding of one message bit (watermark present or not) by modifying random patches of the original image. The algorithm can be expanded to embed multiple message bits.

7.1.1 Embedding

The embedding process for one bit starts by pseudorandomly (seed is based on the key) selecting a number (n) of pixel blocks in the original image. Then we form pairs of pixel blocks by linking the pixel blocks just above the pseudorandomly selected pixel blocks. We represent this pair of pixel blocks by (a_i, b_i). In each pair of pixel blocks 1 is subtracted of the pixel values of the top block (a) and 1 is added to the pixel values of the bottom block (b).

To embed multiple bits, each pair of pixel blocks will contain a message bit, 0 or 1. We can embed a message bit of 0 by adding 1 to the pixel values of the top block and subtracting 1 of the pixel values of the bottom block. A message bit of 1 is embedded by subtracting 1 of the pixel values of the top block and adding 1 to the pixel values of the bottom block. The embedding strength for this embedding process is 1. To achieve more robustness, choose a higher embedding strength.

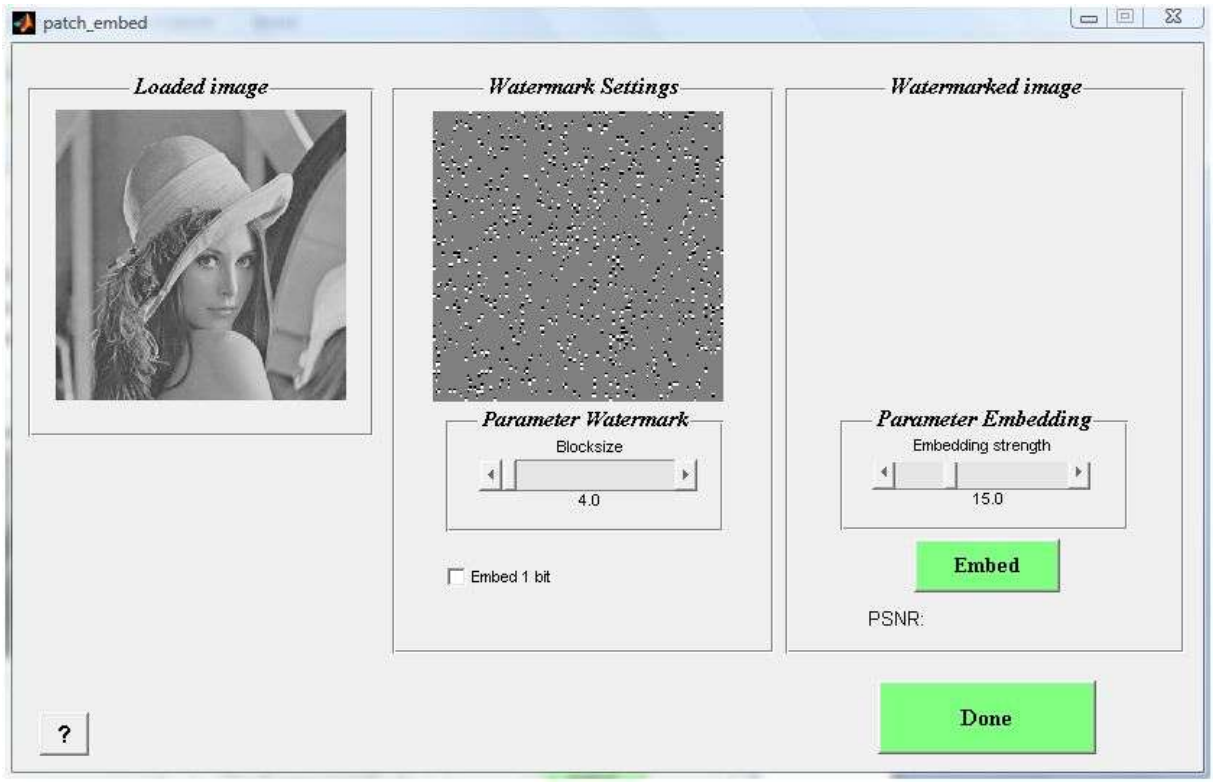
7.1.2 Detecting

The detecting process for one bit starts by selecting the same n pixel blocks in the watermarked image. Again we form pairs of pixel blocks by linking the pixel blocks just above the pseudorandomly selected pixel blocks. The detector sums (b_i - a_i) for i=1…n. At an embedding strength of 1, the difference for each value of i is on average 2. When the total sum approximately equals 2*n, the watermark is detected. If the total sum equals approximately 0, the watermark is not detected. To determine the watermark detection certainty, a threshold is used.

To detect multiple bits in the watermarked bits, the same pairs of pixel blocks will be used. If the top block has a higher average value than the bottom block, a message bit of 0 is detected. If the top block has a lower average than the bottom block, a message bit of 1 is detected. These message bits form the recovered message.

7.2 Embedding GUI

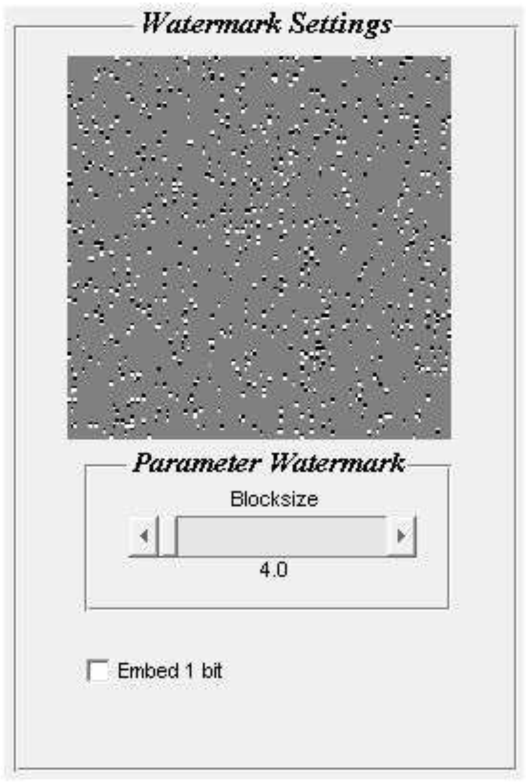
In figure 19 the overview of the embedder GUI is shown. At startup the selected layer of the original image is shown on the left top. Further the watermark is calculated with a default blocksize value.



Blocksize: 4.0
Embedding strength: 15.0
PSNR:

Figure 19: Patch GUI

7.2.1 Adjust the watermark



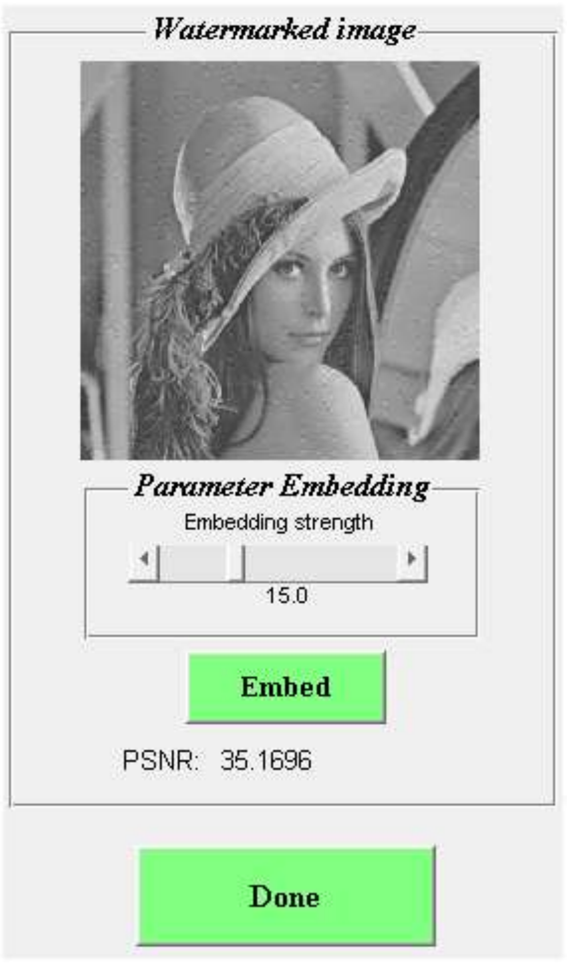
Blocksize: 4.0

☐ Embed 1 bit

Figure 20: Patch settings

In figure 20 the watermark is created for the default blocksize of 4. The gray pixels contain no embedding information. The user can choose a higher blocksize. At a higher blocksize, more pixels of the watermark will contain embedding information. By default multiple bits are embedded in the watermark. The user can embed only one bit by checking the checkbox ‘Embed 1 bit’ beneath the watermark. If one bit is embedded, all top blocks of the pairs of blocks will be black and all bottom blocks white. This allows the detector to detect if the watermark is present or not.

7.2.2 Step 2: Embed the watermark



PSNR: 35.1696

Figure 21: Patch Embed

In figure 21 the watermark is embedded with a default embedding strength of 15. The embedding strength can be lowered to increase fidelity or augmented to increase robustness. Finally the PSNR value indicates the difference between the cover work and the watermarked image. To proceed to the main GUI, click the button Done.

7.3 Detecting GUI

In figure 22 the detecting patchwork GUI for multiple bits is shown. The watermarked image is shown on the left. The watermark detector will detect the message bits by comparing the averages of the pairs of pixel blocks. The values of the differences between top and bottom blocks are shown in the middle image. The recovered message can be compared to the original message on the right.

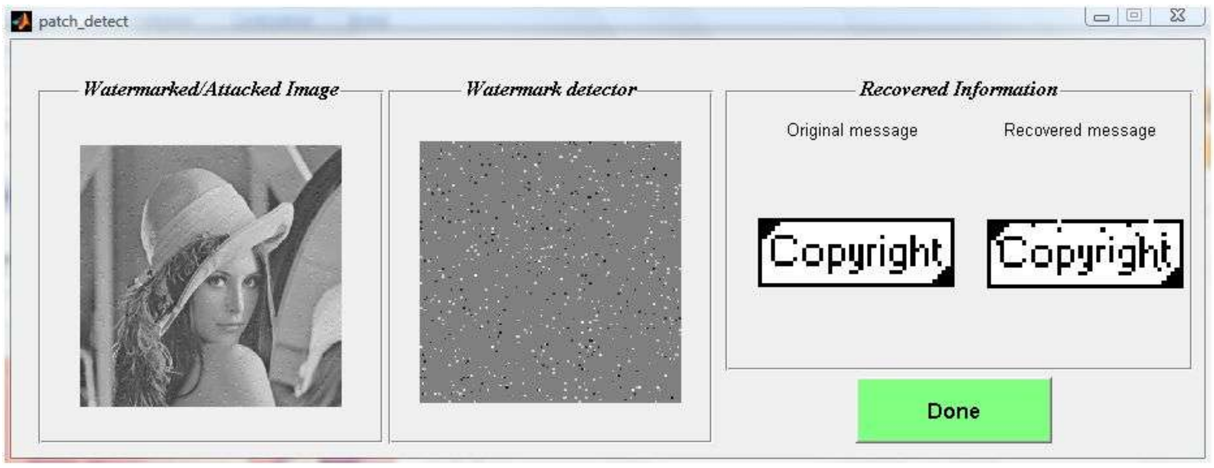
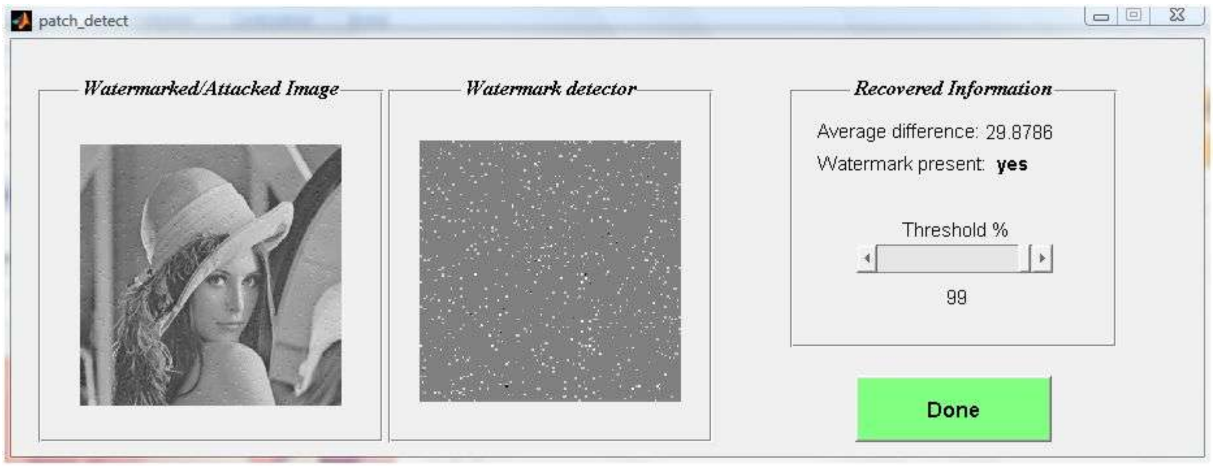


Figure 22: Patch detect – multiple bits

In figure 23 the detecting patchwork GUI for one bit is shown. The watermarked image is shown on the left. The watermark detector will detect the presence of the watermark by comparing the averages of the pairs of pixel blocks. The values of the differences between top and bottom blocks are shown in the middle image. The average difference is shown in the textbox Recovered Information on the right. By comparing the average difference to the specified threshold, the certainty of the presence of the watermark can be calculated. In this case, the user is 99% certain that the watermark is present.



Average difference: 29.8786
Watermark present: **yes**
Threshold %: 99

Figure 23: Patch detect – one bit

8 Correlation

8.1 Correlation technique

The correlation technique is a basic watermarking technique which uses the correlation between random noise patterns and the watermarked image to detect the message. The capacity of the correlation technique depends on the size of the cover work and message.

8.1.1 Embedding

The message is a text or (small) image and consists of a number of bits. First the message has to be converted to bits (0 and 1). The pseudo random number generator (PRNG) will now generate a random noise pattern for the bit 0. This pattern consists of the values 0 and 1. By choosing the noise pattern for the bit 1 as the inverse pattern of the pattern for bit 0, maximum correlation is achieved. Otherwise a random noise pattern for bit 1 can be generated by the PRNG. The blocksize defines the size of the bit patterns. The key is the seed for the pseudo random number generator and determines the composition of the bit patterns.

The message is embedded by adding the bit patterns (pattern for bit 0 or pattern for bit 1) to the cover work for each bit of the message. The robustness can be increased by multiplying the pattern with the embedding strength k.

8.1.2 Detecting

To start detecting, the patterns for the bit 0 and bit 1 are reconstructed. Then the correlation is calculated between the blocks of the watermarked image and the bit patterns. Depending on the highest correlation for these two patterns, the recovered message bit is 0 or 1. Finally the recovered message can be compared with the original message.

8.2 Embedding GUI

In figure 24 the overview of the embedder GUI is shown. At startup the selected layer of the chosen image is loaded and shown on the left. Furthermore the maximum blocksize for this message is calculated and the bit patterns are generated for this maximum blocksize.

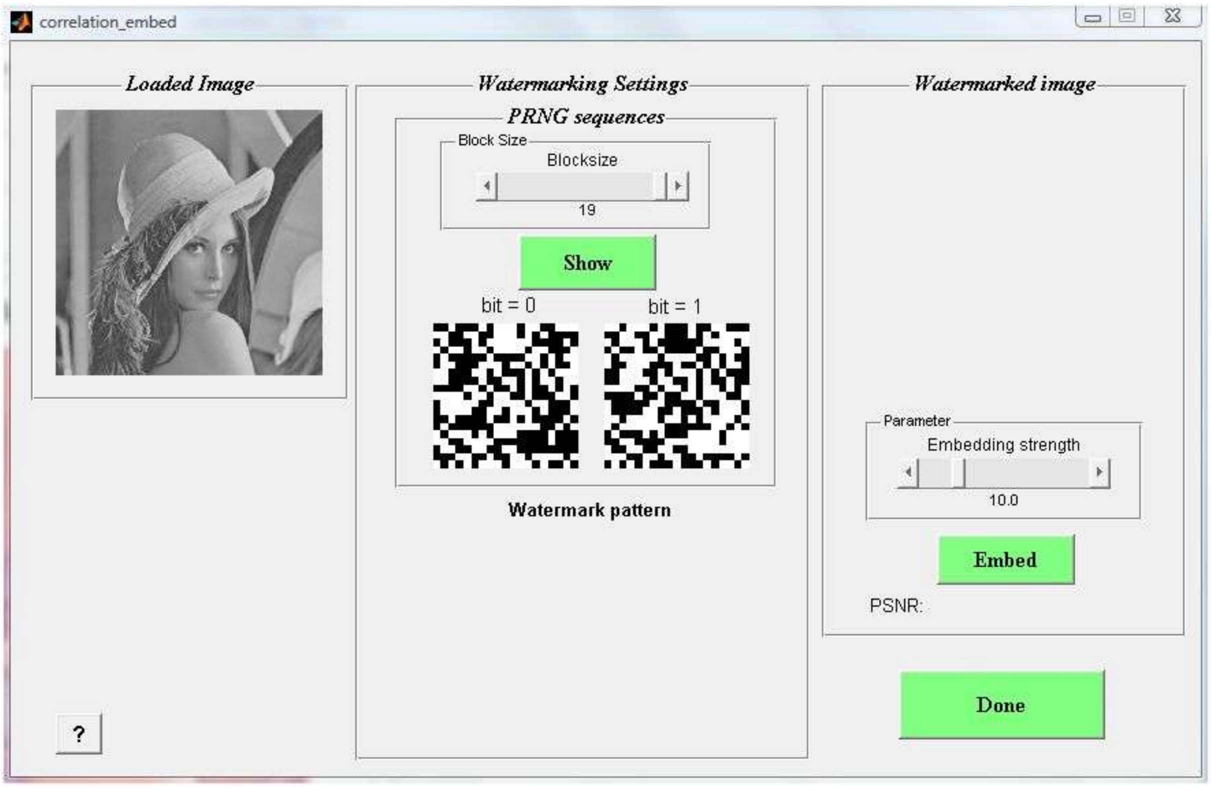


Figure 24: Overview Correlation embedding

8.2.1 Step 1: Adjust the blocksize of the bit patterns

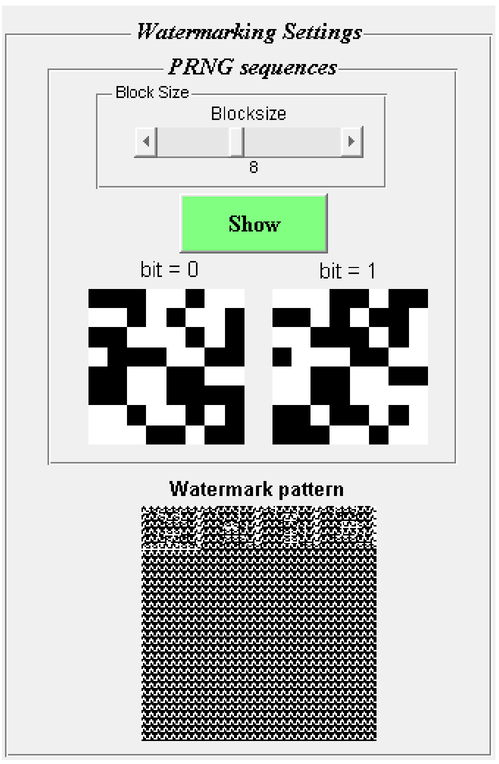


Figure 25: Bit patterns

In figure 25 the bit patterns are shown for a blocksize of 8. Thus a block has the size of 8x8 pixels. You can clearly see that the bit pattern for the bit 1 is the inverse pattern of that for bit 0. By adjusting the blocksize and clicking the button Show, the bit patterns are recalculated and the watermark pattern is reconstructed.

8.2.2 Step 2: Embed the watermark

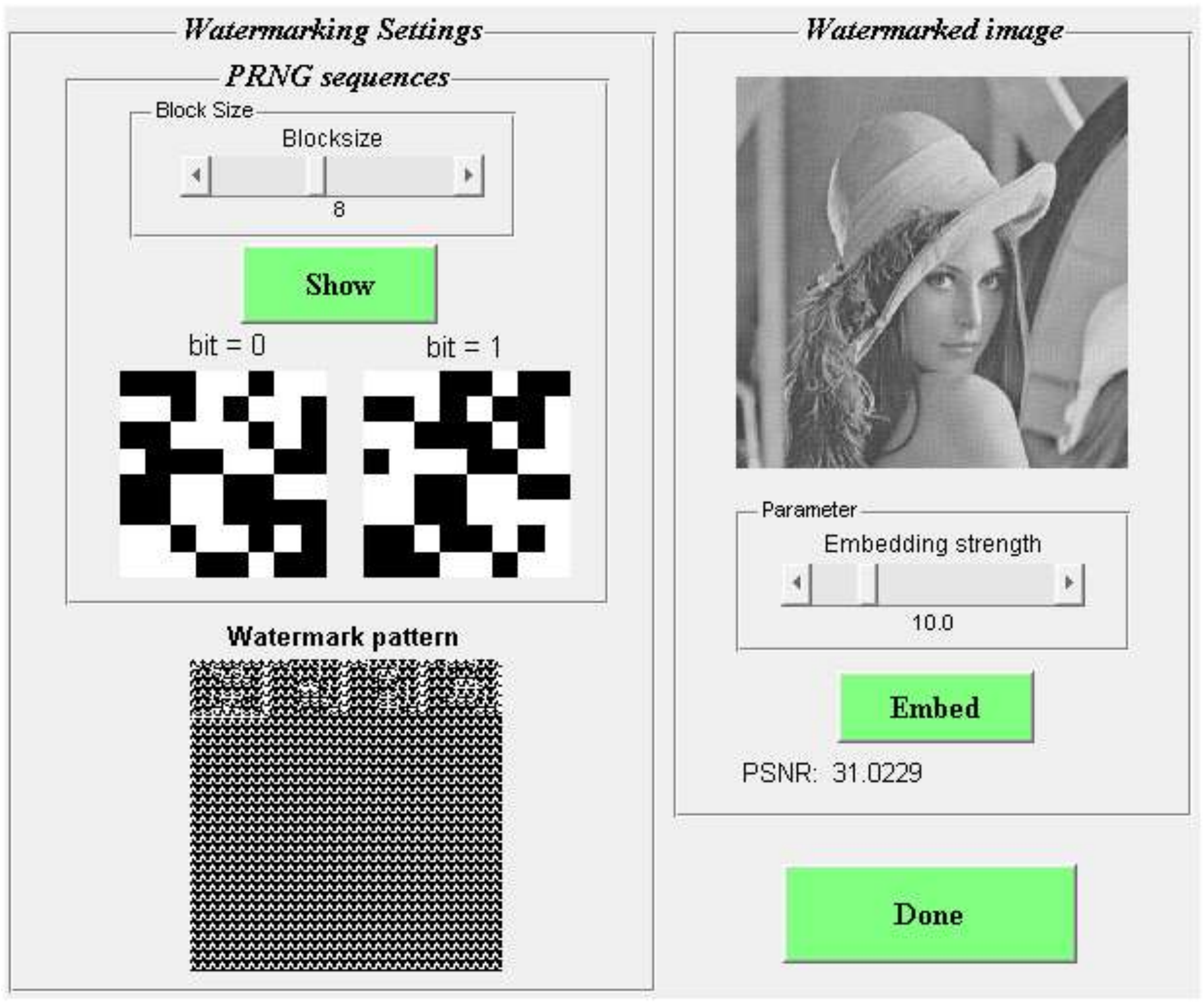


Figure 26: Correlation embedding

In figure 26 the watermark is constructed for a blocksize of 8. By clicking on the button Embed the watermark is added to the cover work with an embedding strength of 10. The embedding strength can be decreased to improve fidelity. By increasing the embedding strength, the robustness of the watermark increases. The PSNR value indicates the difference between the cover work and the watermarked image. To proceed to the attack section, click on the button Done.

8.3 Detecting GUI

In figure 27 the overview of the detector GUI is shown.

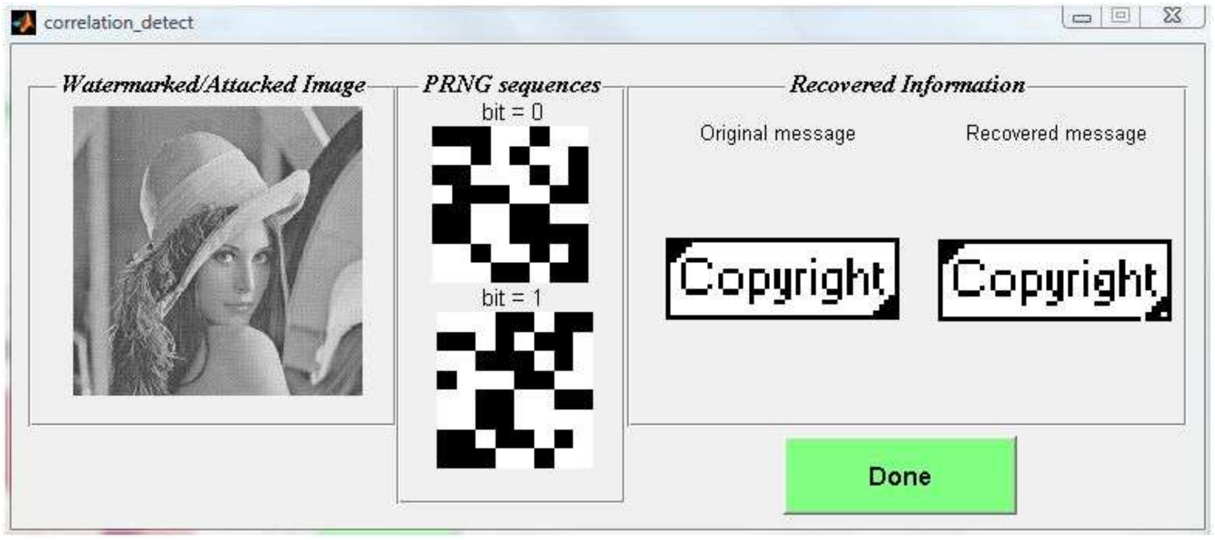


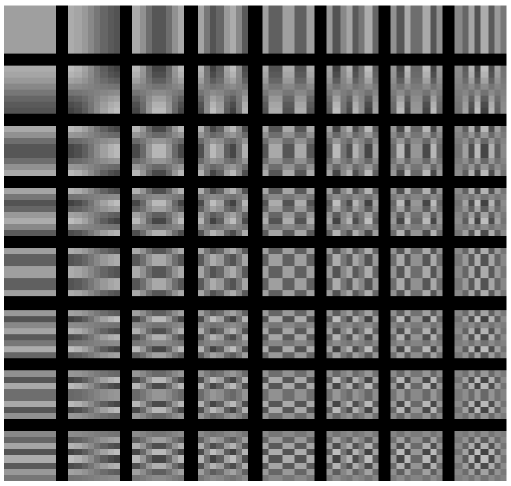
Figure 27: Correlation detection

The bit patterns are recalculated and shown using the same key as the embedder. The detector calculates the correlation between the blocks of the watermarked image and the bit patterns to recover the message bits. These message bits are composed to form the recovered message. The original and recovered message can easily be compared. In this case, three pixels have a different value. If the original message was a text, the detector will automatically compose the recovered message bits to a text. By clicking on the button Done, the detector GUI is closed.

9 DCT

9.1 DCT technique

The DCT technique will transform an image to a sum of weighted base functions. These base functions are sinus waves with different frequencies. For 8x8 matrices, the base functions are shown in this image:



The frequencies of the sinus waves increase from left to right, and from top to bottom. The base function on the upper left has a constant value and is called the DC base function. The coefficient that corresponds to this function is called the DC coefficient. By transforming an image with the DCT transformation the visual significant information is mostly represented by the DC coefficient and the lower frequency coefficients. For this reason DCT is used for image compression e.g. JPEG compression.

9.1.1 Embedding

The cover work is transformed to DCT coefficients. The user can choose in which coefficients the watermark will be embedded. The low, middle, high or a combination of these frequency coefficients can be chosen to embed the watermark. The DC coefficient cannot be watermarked, because it contains the visually most significant information. By embedding the watermark in the high frequency coefficients, the watermarked image will less change then with embedding in the low or middle frequency coefficients. In the practical implementation, the bit patterns for 0 and 1 are multiplied with a mask that selects the specified frequency coefficients. These masked bit patterns are used to transform the message to a watermark. The inverse DCT transformation results in the watermarked image.

9.1.2 Detecting

First the DCT transformation is applied to the watermarked image. Then the correlations are calculated between the DCT decomposition blocks and the masked bit patterns. These correlations are compared and result in the recovered message bits, which form the recovered message.

9.2 Embedding GUI

In figure 28 the overview of the embedder GUI is shown. At startup the selected layer of the chosen image is automatically loaded and shown. Further the DCT transformation is applied and shown beneath the loaded image. The first DCT block is enlarged at Watermark Settings. The embedding process will be visualized using this block.

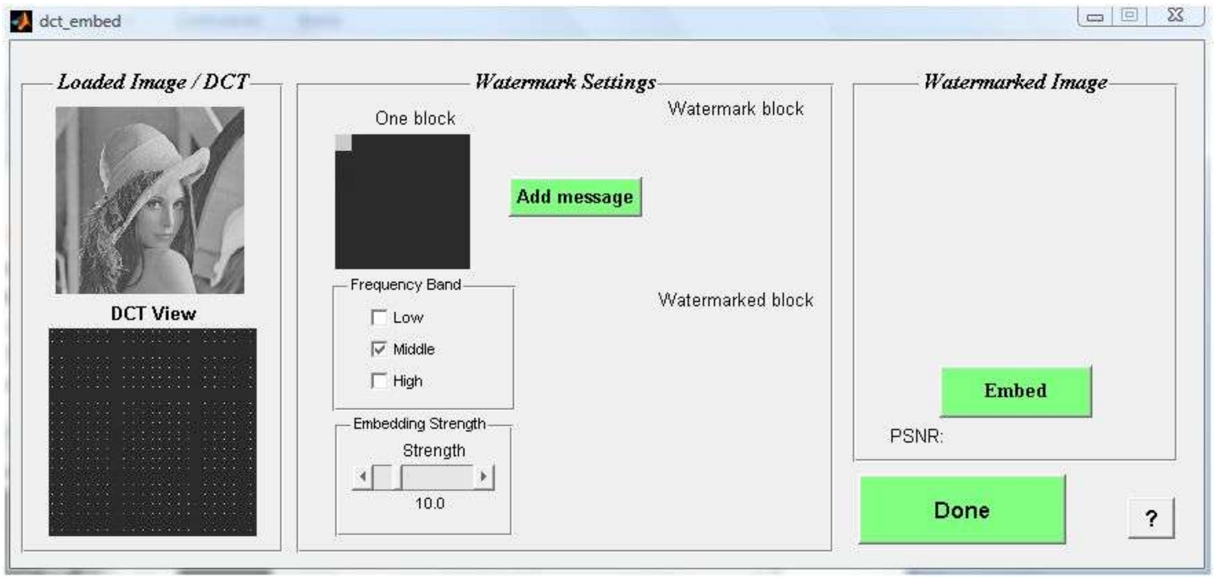


Figure 28: DCT embedding GUI

9.2.1 Step 1: Add the message

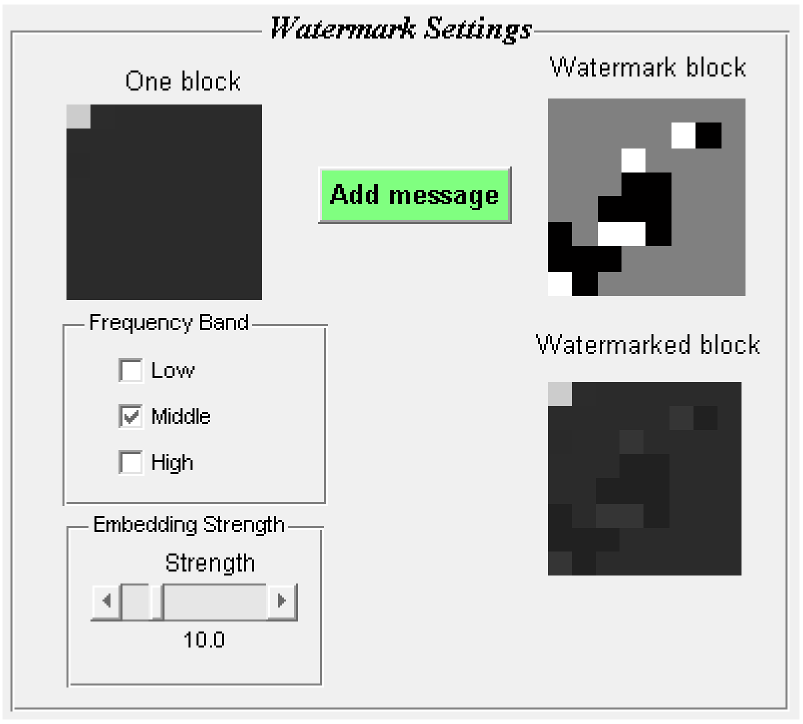
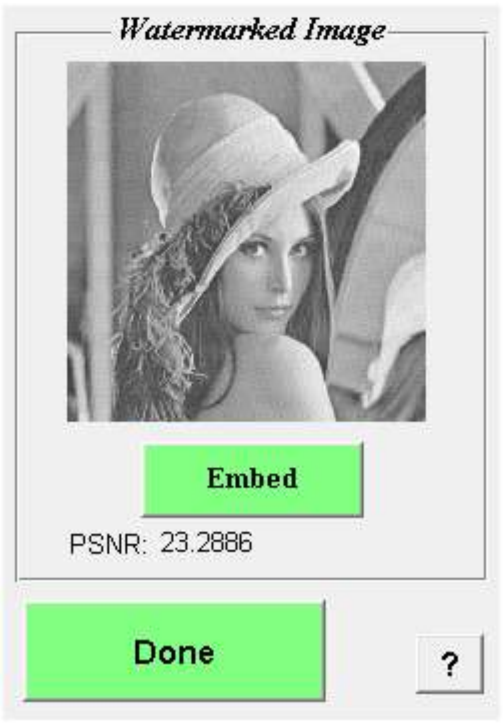


Figure 29: DCT watermark settings

In figure 29 the watermark block is added to the first block on the left, which results in the watermarked block on the right. This process is repeated for all blocks of the original image. The user can use the checkboxes to determine which frequency coefficients are used for embedding the watermark. The embedding strength can also be adjusted by the user. The embedding process is similar to the correlation technique.

9.2.2 Step 2: Embed the watermark



PSNR: 23.2886

Figure 30: DCT watermarked image

By clicking the Embed button, the watermark is embedded and inverse DCT transformation is performed. This results in the watermarked image. The PSNR value indicates the difference between the cover work and the watermarked image.

9.3 Detecting GUI

In figure 31 the overview of the DCT detecting GUI is shown.

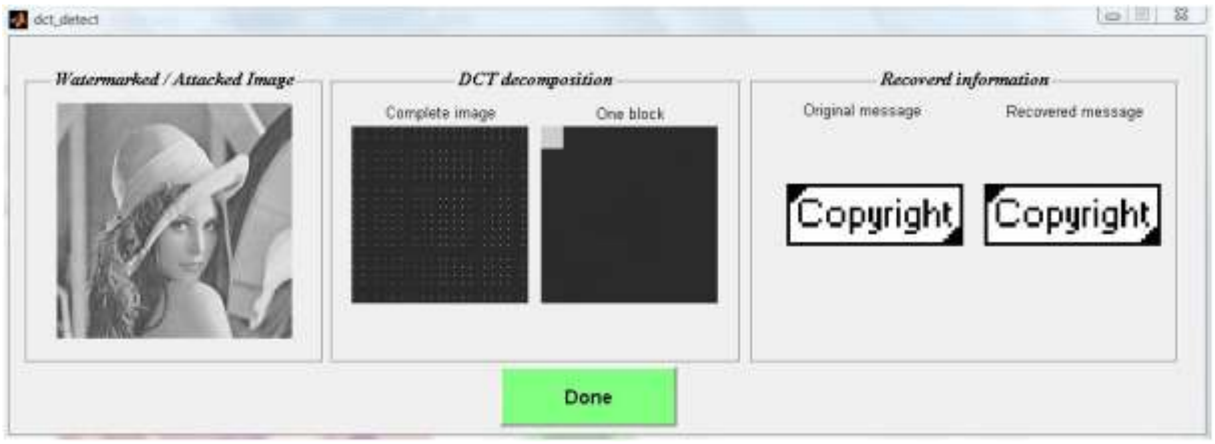


Figure 31: DCT detecting GUI

The watermarked image is shown on the left. In the middle of the figure the DCT decomposition is shown for the complete image and one watermarked block. At startup the correlations between the watermarked blocks and the masked bit patterns are calculated. These correlations result in the recovered message bits, which form the recovered message. The original and recovered message can be compared on the right.

If the message is a text, the recovered message text will be shown in a textbox next to the original text. By clicking on the button Done, the detecting GUI will be closed.

10 DWT

10.1DWT technique

The DWT technique transforms the original image to the wavelet domain before embedding the watermark. The wavelet decomposition of the original image is constructed by applying two filters and rescaling the result. The type of wavelet determines the choice of these two filters, one low pass filter and one high pass filter. The result of the wavelet transformation is an image with the same dimensions as the original image. This image consists of four matrices of the same size, one approximation matrix and three details matrices. These details matrices contain the horizontal, vertical and diagonal details. The approximation matrix contains the significant information of the original image. By applying the wavelet transformation to the approximation matrix, the second level of DWT is constructed. In this GUI, the wavelet decomposition will consist of 5 levels.

10.1.1 Embedding

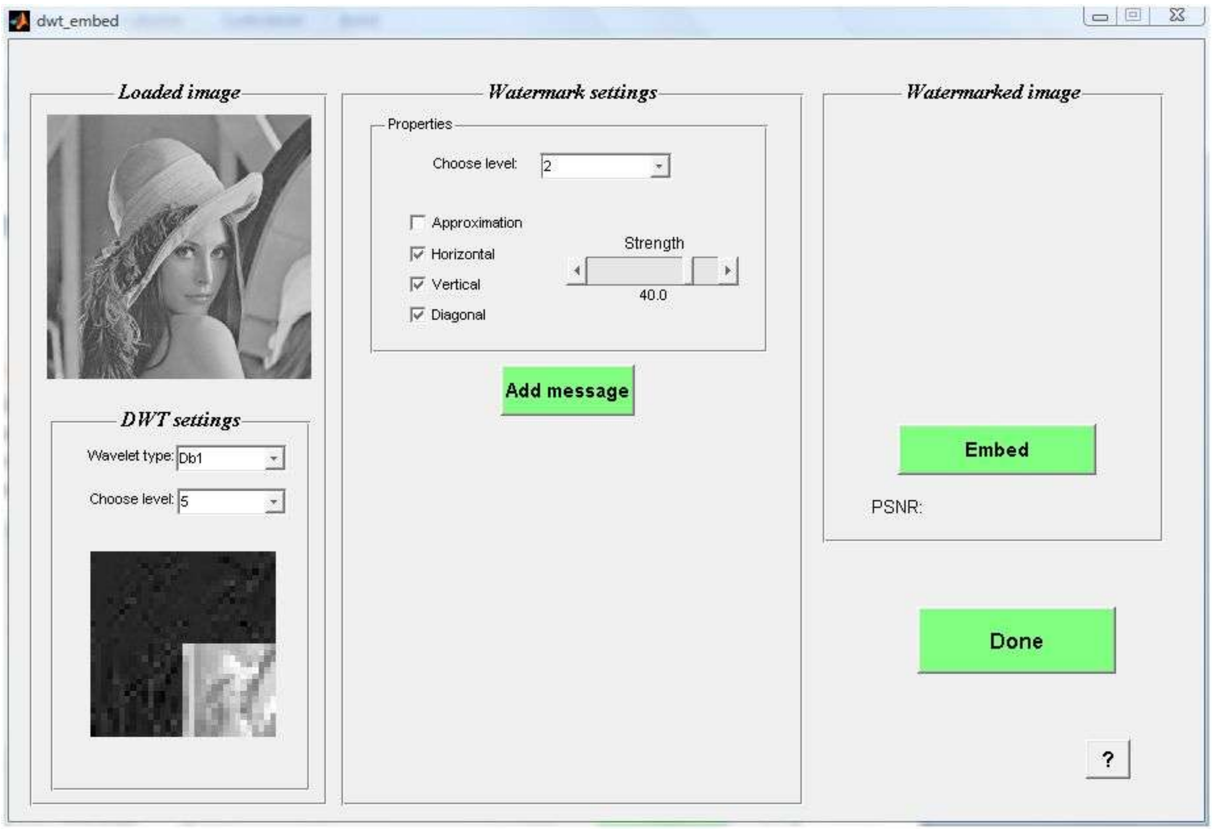
The message is first converted to bits (0 and 1). Then two bit patterns for 0 and 1 are constructed by the pseudo random number generator (PRNG). Depending on the message bits, the bit patterns are added to the original image. The watermark is a composition of these bit patterns. The watermark is by default added to all three details matrices. Eventually the watermark can be added to the approximation matrix. Finally the inverse DWT transformation is applied to obtain the watermarked image.

10.1.2 Detecting

At the detector, the DWT transformation is applied to the watermarked image. A watermarked matrix is selected and the correlations between the blocks of this matrix and the bit patterns are calculated. The recovered message bit is a 0 or 1, depending on these correlations. The recovered message bits are composed to form the recovered message.

10.2 Embedding GUI

In figure 32 the overview of the embedder GUI is shown. At startup the selected layer of the chosen image is loaded and shown. The level 5 DWT decomposition of this image is calculated and shown beneath the loaded image.



Choose level: 2

☐ Approximation
☑ Horizontal
☑ Vertical
☑ Diagonal

Strength
40.0

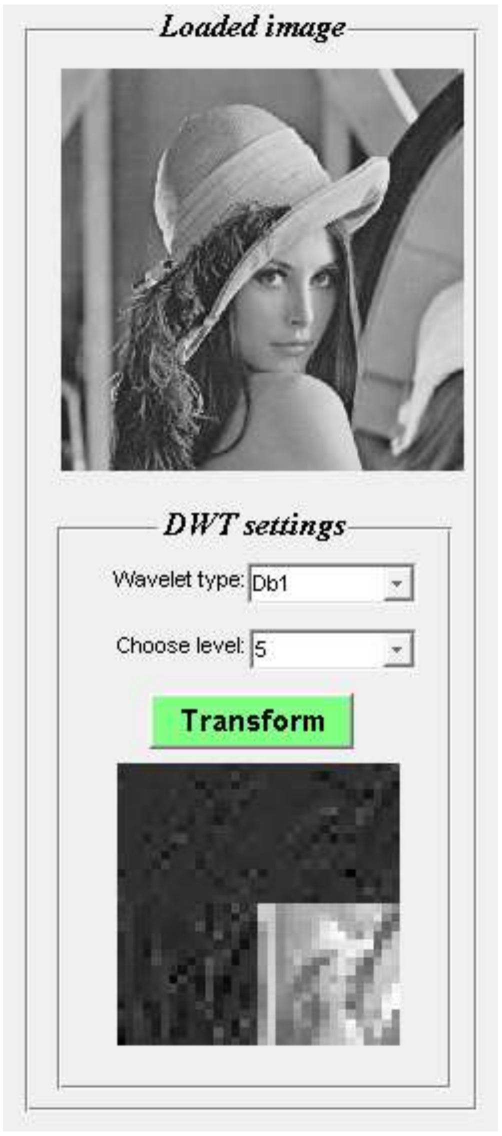
PSNR:

Wavelet type: Db1

Choose level: 5

Figure 32: DWT embedding GUI

10.2.1 Step 1: Adjust the DWT decomposition



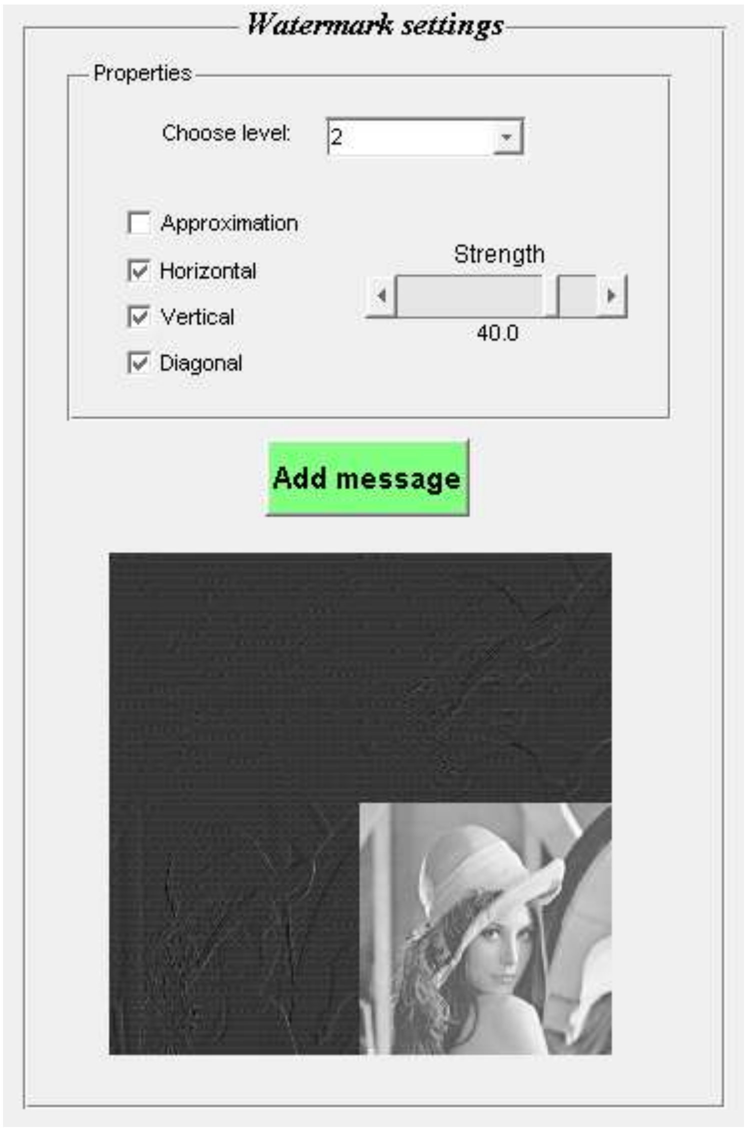
Wavelet type: Db1

Choose level: 5

Figure 33: DWT decomposition

In figure 33 the DWT decomposition at level 5 of the cover work is shown. This decomposition can be (re)calculated by clicking the button Transform. The user can specify which wavelet type will be used for the decomposition by choosing between Db1, Bior1.1 and Rbio1.1. Further the wavelet level can be selected. Overview shows the complete decomposition. The diagonal details matrix is shown on the upper left, the horizontal details matrix on the upper right. Then the vertical details matrix is shown on the bottom left and the approximation matrix on the bottom right.

10.2.2 Step 2: Add the message


Watermark settings

Properties

Choose level: 2

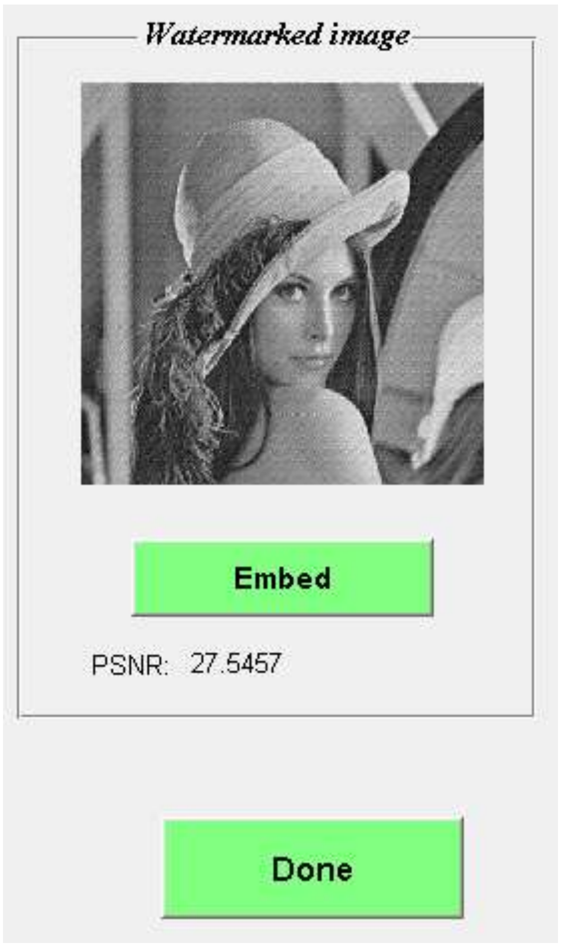
- [ ] Approximation
- [✓] Horizontal
- [✓] Vertical
- [✓] Diagonal

Strength: 40.0

Figure 34: DWT embedding settings

In figure 34 the watermark is added to the three details matrices of the second level of the DWT decomposition. These settings can be adjusted by the user in the properties box. It is also possible to embed in the third level of the DWT decomposition. The watermark can also be added to the approximation matrix. The embedding strength is by default set to 40. The embedding process is similar to the correlation technique.

10.2.3 Step 3: Embed the watermark



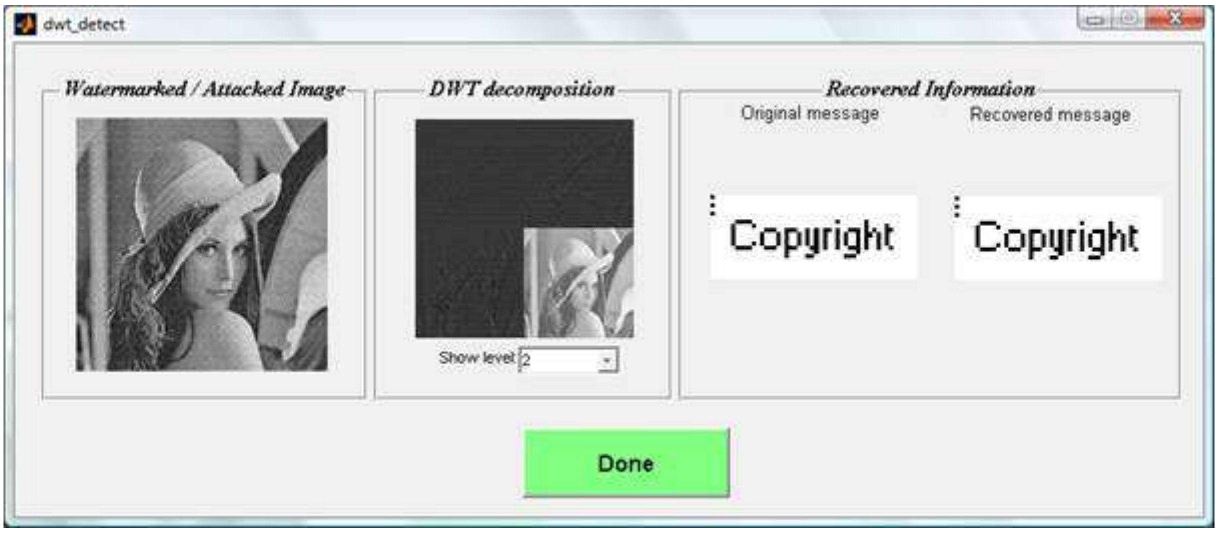
PSNR: 27.5457

Figure 35: DWT watermarked image

By clicking the button Embed, the watermark is added to the original image and the inverse DWT transformation is applied. This results in the watermarked image. The PSNR value indicates the difference between the cover work and the watermarked image.

10.3 Detecting GUI

In figure 36 the overview of the detector GUI is shown. At startup the watermarked image is automatically loaded. Furthermore the DWT decomposition at the second level is shown, which shows the watermarked details matrices. Each level of the decomposition can be selected beneath the image. Then the correlations between the bit patterns and the blocks of a watermarked matrix are calculated. These correlations result in the recovered message bits. The original and recovered message can be compared on the right. The quality of detection depends on the selected level and matrix of the decomposition.



Original message: Copyright

Recovered message: Copyright

Figure 36: DWT detecting GUI

If the message is a text, the recovered message text will be shown in a textbox next to the original text. By clicking on the button Done, the detecting GUI will be closed.

11 DS-CDMA

11.1 DS-CDMA technique

The Direct Sequence – Code Division Multiple Access (DS-CDMA) technique is a correlation-based technique. In the regular correlation technique, each bit is embedded in a block of the cover work. The DS-CDMA technique embeds each bit in the entire cover work. This causes interference at detection but cropping will usually not destroy the embedded message.

11.1.1 Embedding

The message is first converted to bits (0 and 1). Then a random noise pattern (consisting of -1 and 1) is created for each message bit. This random noise pattern has the same size as the cover work. If the message bit is 0, the random noise pattern for that bit is added to the watermark pattern. If the message bit is 1, the random noise pattern for that bit is subtracted from the watermark pattern. Finally the watermark pattern is added to the cover work to form the watermarked image.

11.1.2 Detecting

At the detector, the correlation for each message bit must be determined. First the correlation is calculated for each pixel of the watermarked image. Then the average of these correlations is calculated. If the average is positive, the message bit is 0. If the average is negative, the message bit is 1. These resulting message bits form the recovered message.

11.2 Embedding GUI

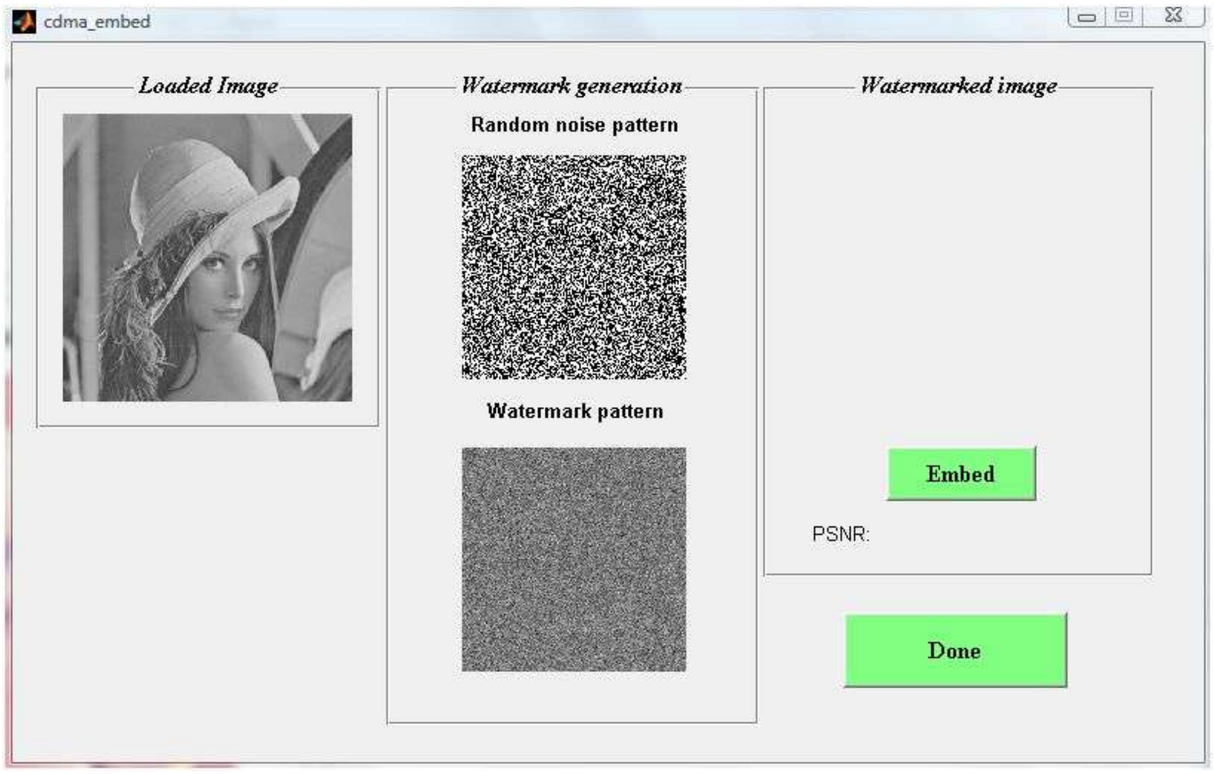


Figure 37: DS-CDMA embedding GUI

In figure 37, the embedding GUI of the DS-CDMA technique is shown. At startup, the selected layer of the chosen image is shown on the left. The random noise pattern for one message bit is shown in the middle. Beneath this random noise pattern, the watermark pattern for all message bits is shown.

11.2.1 Embed the watermark

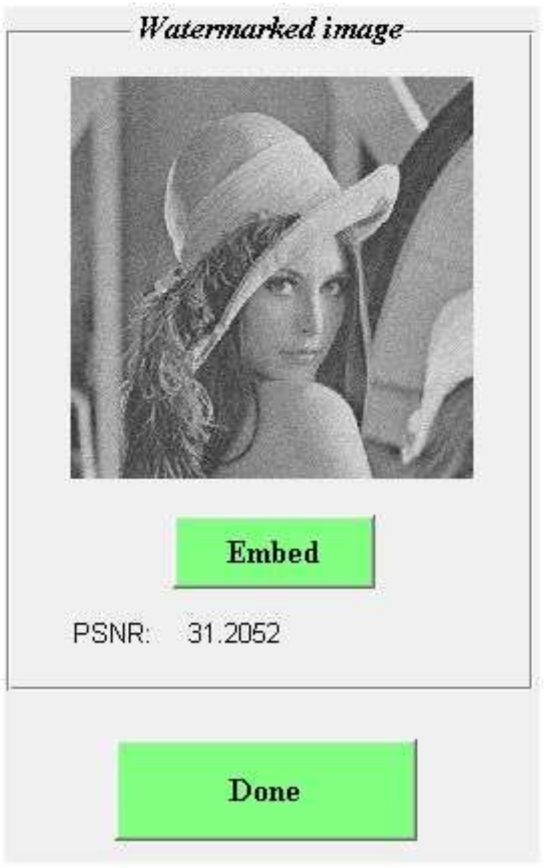


Figure 38: Watermark embedded

The watermark pattern is embedded in the cover work by clicking the button Embed (figure 38). The PSNR value indicates the difference between the cover work and the watermarked image. By clicking the button Done, the program returns to the main GUI, where the watermarked image can be attacked or the message can be detected.

11.3 Detecting GUI

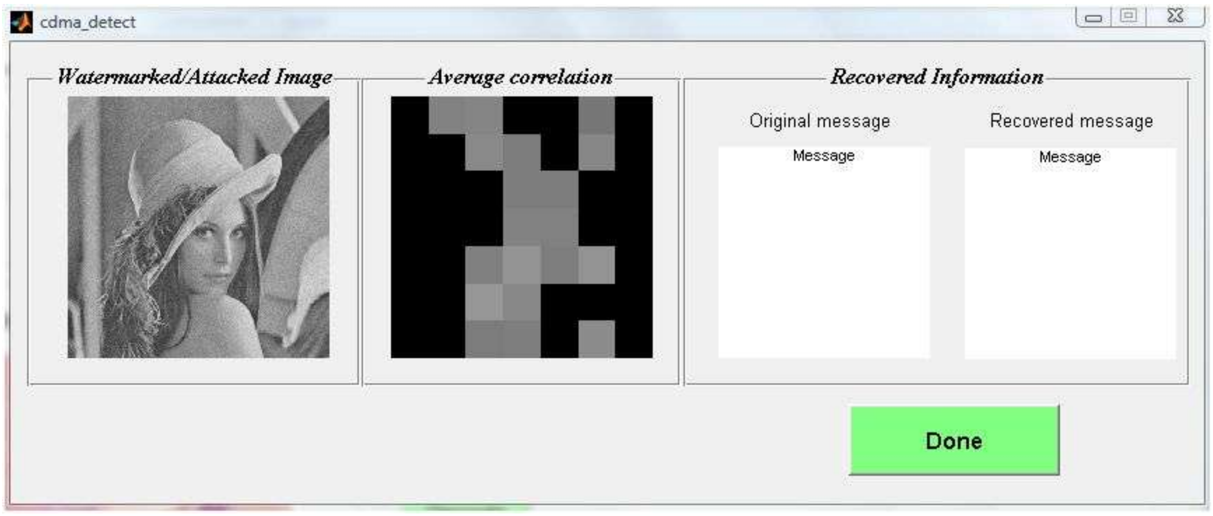


Figure 39: DS-CDMA detecting GUI

In figure 39, the DS-CDMA detecting GUI is shown. At startup, the watermarked/attacked image is shown on the left. Then the average correlation for each message bit is calculated and shown in the middle of the GUI. A black square indicates a negative value, which means that the message bit is a 1. A grey or white square indicates a positive average value, which means that the message bit is a 0. Each row in the average correlation matrix forms a character. By composing the bits to characters, the message can be recovered and compared on the right. By clicking the button Done, the program returns to the main GUI, where additional parameters can be shown by clicking the button Overview.

12 QIM

12.1 QIM technique

The Quantization Index Modulation technique is based on quantization instead of correlation. Instead of adding a watermark pattern to the cover work, the watermark data is embedded by quantization and dither modulation of the cover work. This technique provides higher fidelity than correlation-based techniques.

12.1.1 Embedding

First the message is converted to bits (0 and 1). Then two dither vectors are created by the Pseudo Random Number Generator with seed based on the provided key. The length of these vectors determines the blocksize of the algorithm. One dither vector will be used to embed the bit 0, the other dither vector will be used to embed the bit 1. The two dither vectors differ exactly delta/2, with delta as quantization index. To embed the message bit 0, the dither vector for bit 0 is first subtracted from the pixel values of the selected block. Then quantization is performed with quantization index equal to delta. The quantized value is rounded to the nearest value and then the pixel value is restored by multiplying with delta. Finally the dither vector is again added to obtain the watermarked pixel values. The same procedure is repeated for message bits equal to 1, but the dither vector for bit 1 is used instead. The resulting image is the watermarked image.

12.1.2 Detecting

At the detector, the dither vectors for the bits 0 and 1 are reconstructed. Then the pixel values of the blocks of the watermarked image are again quantized and dither modulated, but only with the dither vector for bit 0. If the difference between the modulated pixels and the pixels of the watermarked image is lower than delta/4, a message bit equal to 0 is detected. If the difference is greater than delta/4, a message bit of 1 is detected. By composing the resulting message bits, the recovered message is obtained.

12.2 Embedding GUI

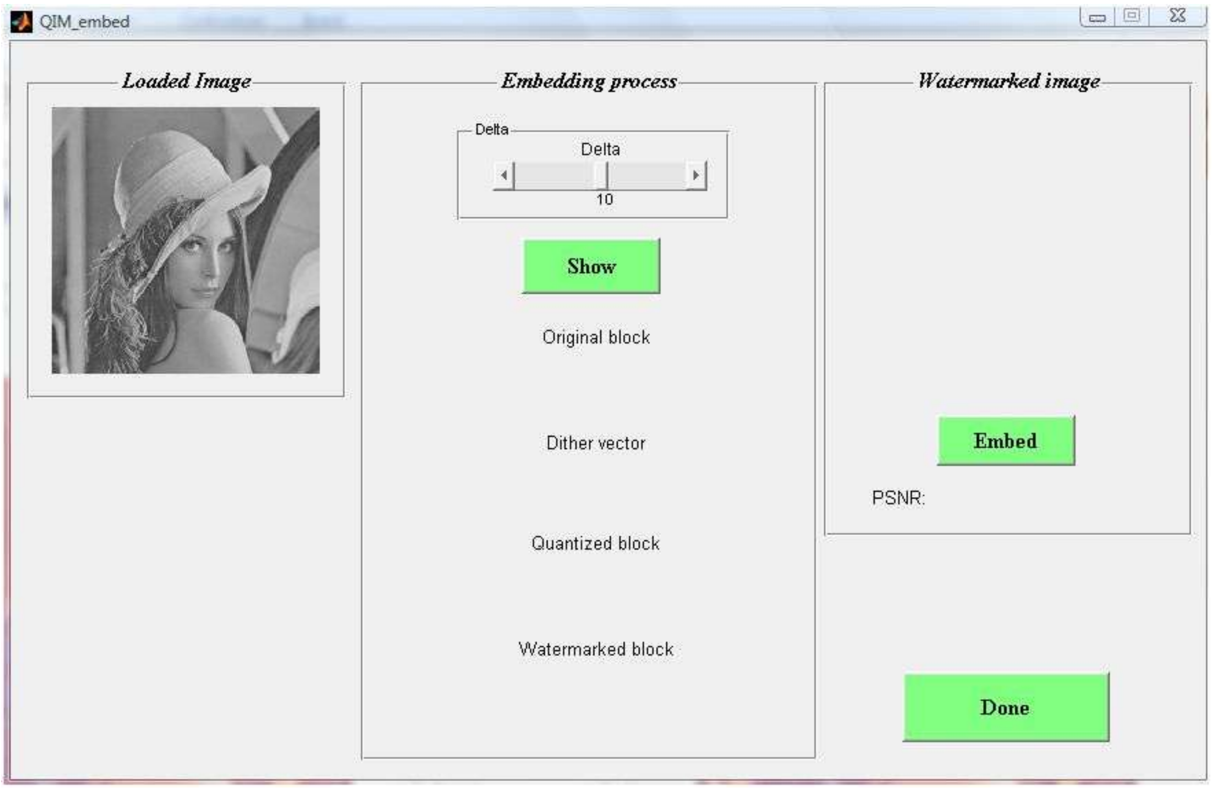


Figure 40: QIM embedding GUI

In figure 40 the QIM embedding GUI is shown. At startup, the selected layer of the chosen image is shown on the left. The embedding process will be visualized in the middle, and the watermarked image will be shown on the right.

12.2.1: Step 1: Show the embedding process

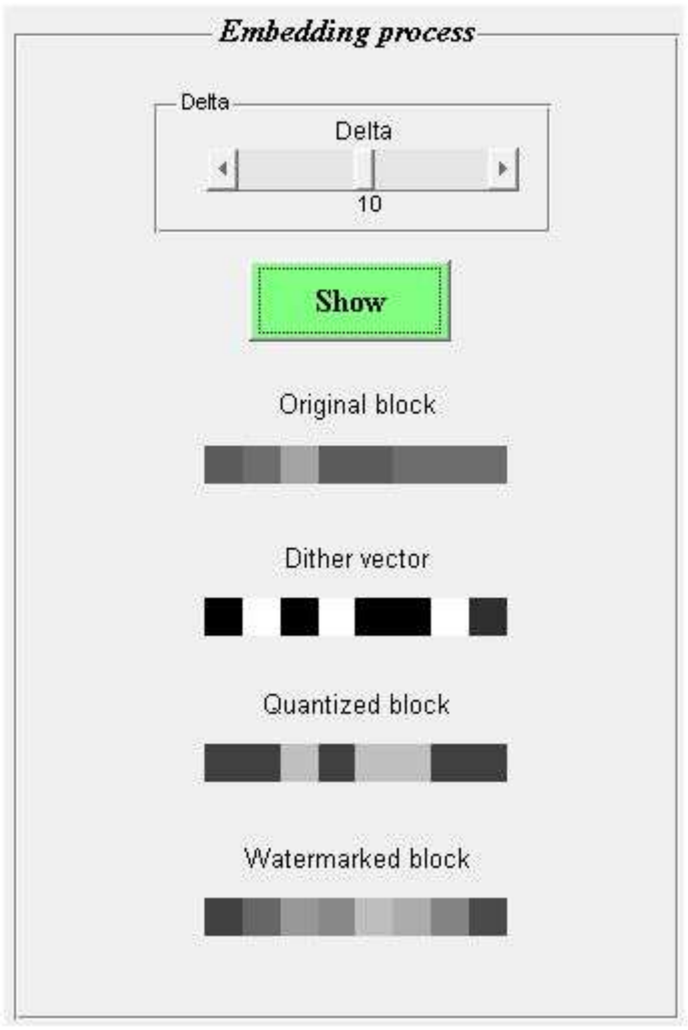
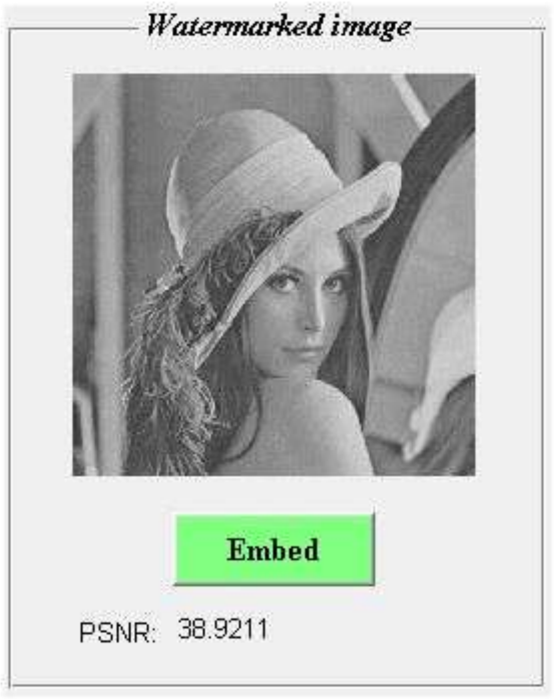


Figure 41: QIM embedding process

In figure 41 the embedding process is shown after clicking the button Show. The user can set the quantization index, delta, by adjusting the slider on top. The original block is shown beneath the button Show. The dither vector used for this block is visible beneath the original block. Then the quantized block is calculated by subtracting the dither vector and performing the quantization. The watermarked block is obtained after adding the dither vector to the quantized block.

12.2.2 Step 2: Embed the watermark



PSNR: 38.9211

Figure 42: QIM watermarked image

In figure 42 the watermarked image is shown after clicking the button Embed. The PSNR value indicates the difference between the cover work and the watermarked image.

12.3 Detecting GUI

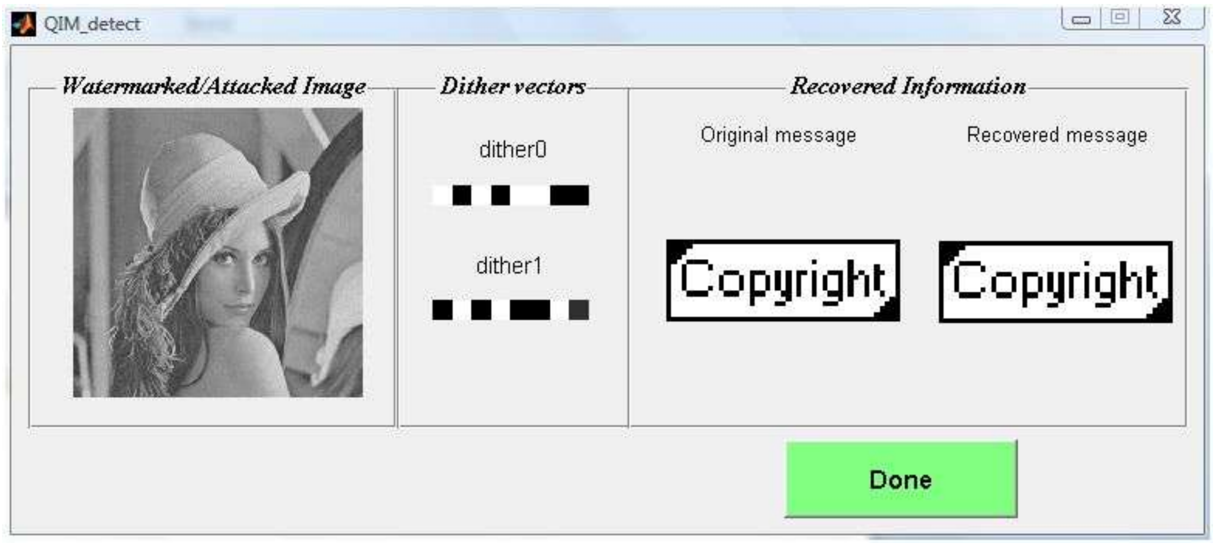


Figure 43: QIM detecting GUI

In figure 43, the QIM detecting GUI is shown. At startup, the watermarked/attacked image is loaded and shown on the left. Then the dither vectors are reconstructed and shown in the middle. The dither vectors are used to detect the message bits, which form the recovered message. The recovered message can be compared to the original message on the right. By clicking the button Done, the program returns to the main GUI.

13 Attack GUI

In figure 44 the overview of the attack GUI is shown. At startup the watermarked image is loaded and shown on the left.

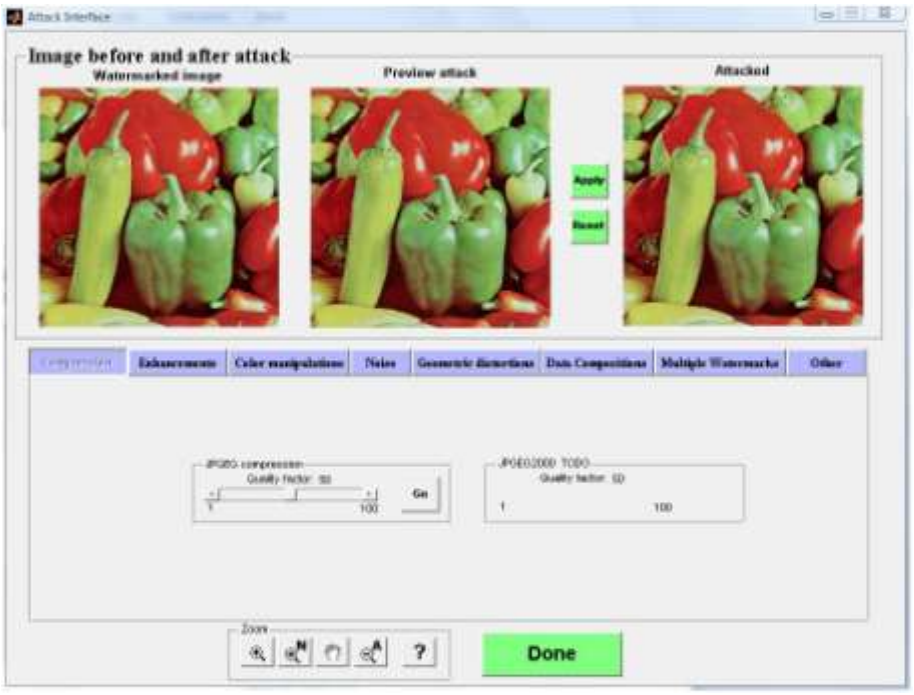


Figure 44: Attack GUI

After choosing the settings for an attack, the attack is applied by clicking on the button Go. Then a preview of the attacked image is shown in the middle of the GUI. After clicking the button Apply, the attack is performed on the watermarked image, resulting in the attacked image on the right. Several attacks can be applied after each other, but the attack is only performed when the Apply button is clicked. The user can change the settings and view the result on the preview image, before performing the attack on the watermarked image. After clicking the button Done, the attack GUI is closed and the attacked image is shown in the main GUI.

Now we will show the different possible attacks, from left to right.

13.1 Compression attacks

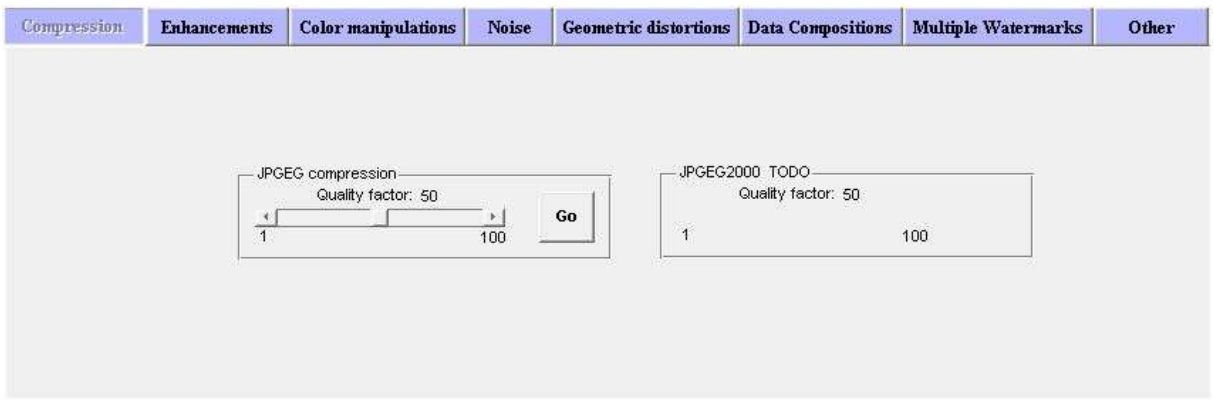


Figure 45: Compression attacks

The first type of attacks are compression attacks (figure 45). These attacks lower the quality of the watermarked image due to quantization. The quality factor of the compression is determined by the degree of quantization. A quality factor of 100 returns the original image. In figure 46 (lossy) compression is applied with a quality factor of 50.

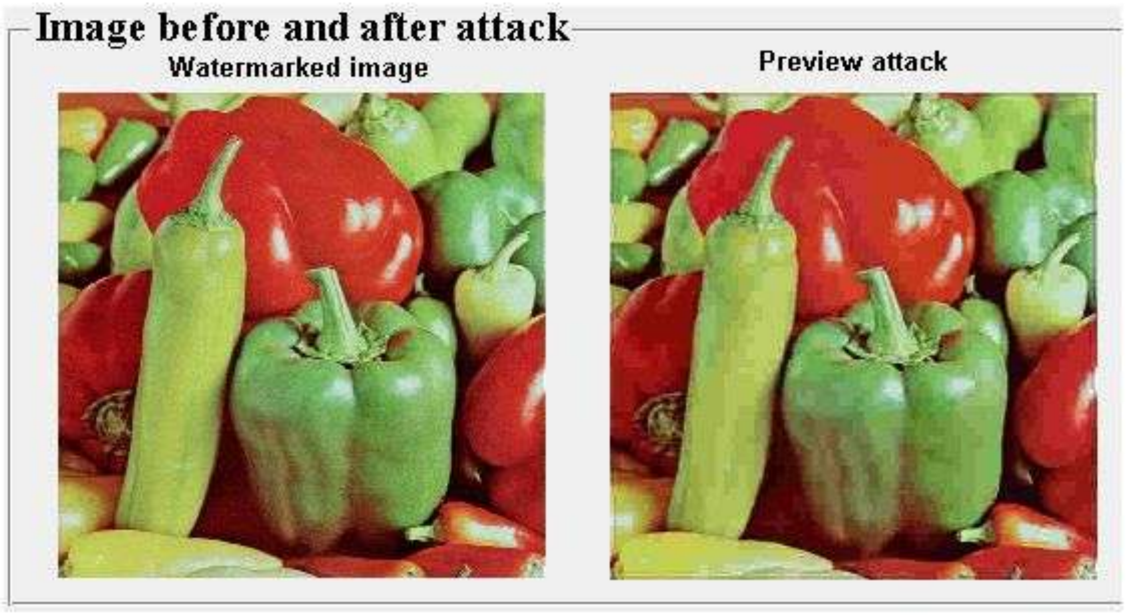


Figure 46: Lossy compression

13.2 Enhancements

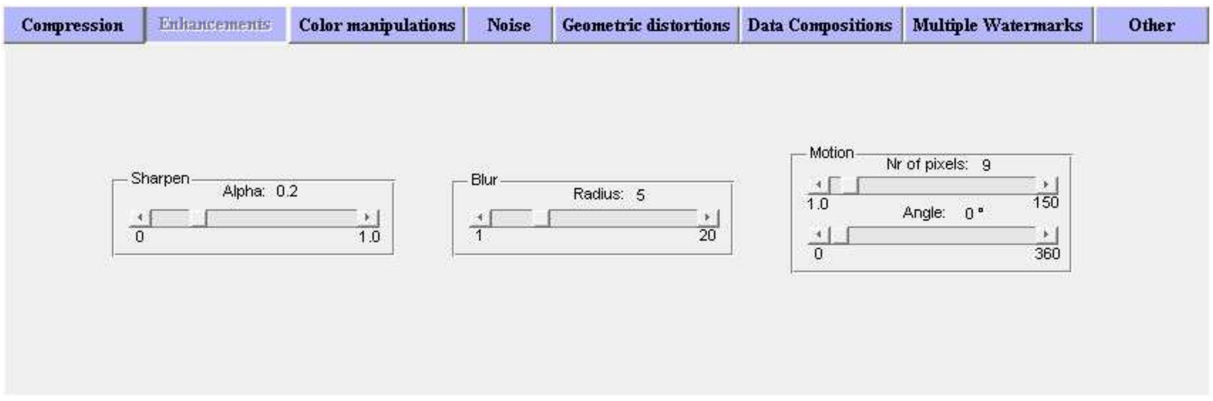


Figure 47: Enhancement attacks

The second type of attacks are enhancement attacks. These attacks are performed by the user to improve the quality of the image. The sharpening attack applies the unsharp filter, resulting in a more sharp image. The blurring attack will lead to a more vague image at increasing radius. Finally, the motion attack creates a moving image (figure 48).

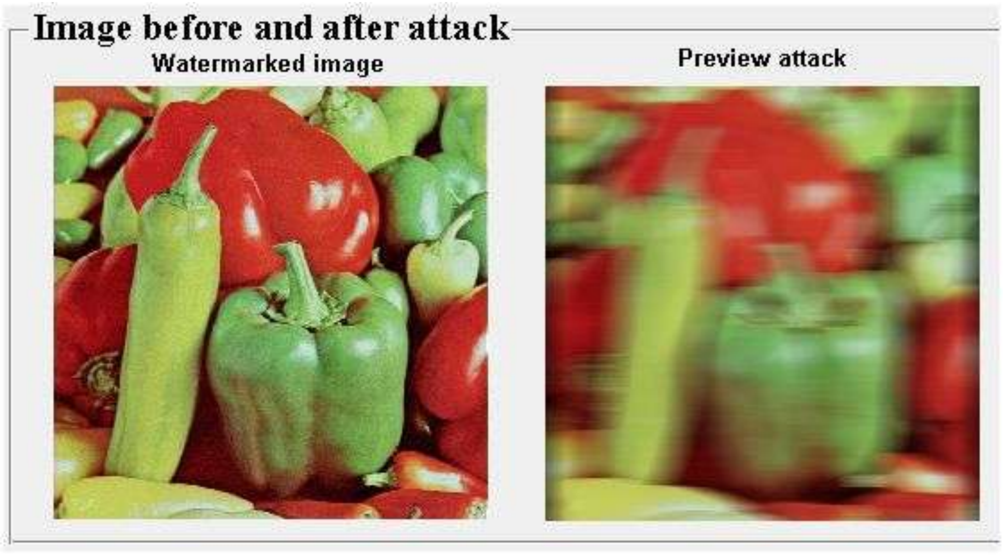
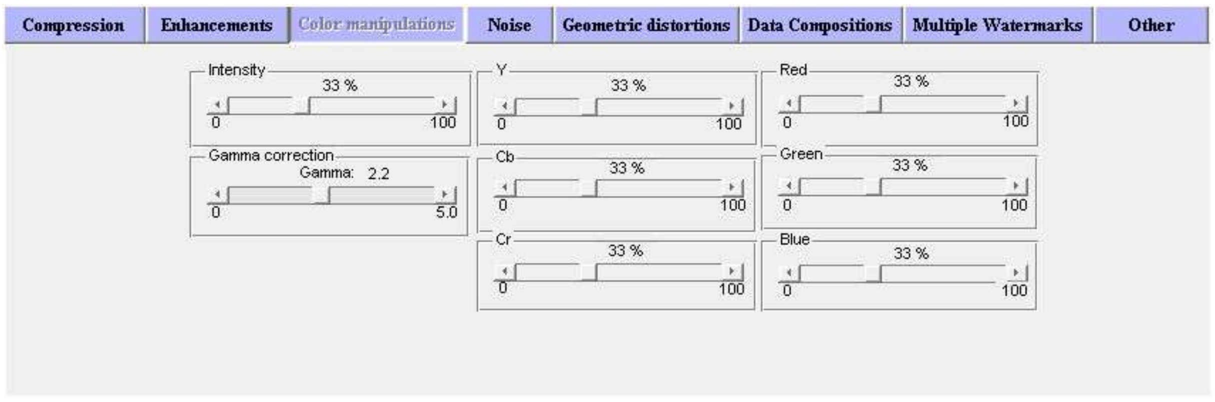


Figure 48: Motion attack

13.3 Color manipulations



Intensity: 33 % (0–100)
Gamma correction: Gamma: 2.2 (0–5.0)
Y: 33 % (0–100)
Cb: 33 % (0–100)
Cr: 33 % (0–100)
Red: 33 % (0–100)
Green: 33 % (0–100)
Blue: 33 % (0–100)

Figure 49: Color manipulations

The watermarked image can be attacked by means of color manipulations. The general intensity of the image can be changed with the slider on the left. To change the intensity of the layers, the sliders of the layers Y,Cb,Cr in the middle and/or the sliders of the layers R,G, B on the right can be adjusted. The default intensity is set to 33%. An increase in intensity makes the image or layer brighter, while a decrease in intensity darkens the image or layer. In figure 50 the intensity of the green layer has been increased to 50%.

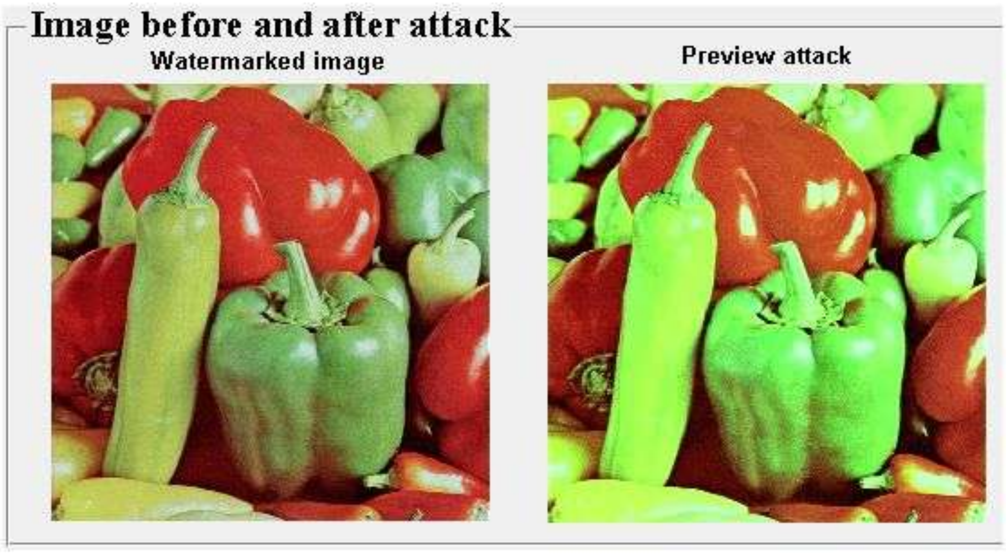


Figure 50: Intensity adjustment

The gamma correction attack also changes the intensity of the image. A gamma value of 1 maintains the current intensity, a higher value increases the intensity, while a value lower than 1 decreases the intensity.

13.4 Noise attacks

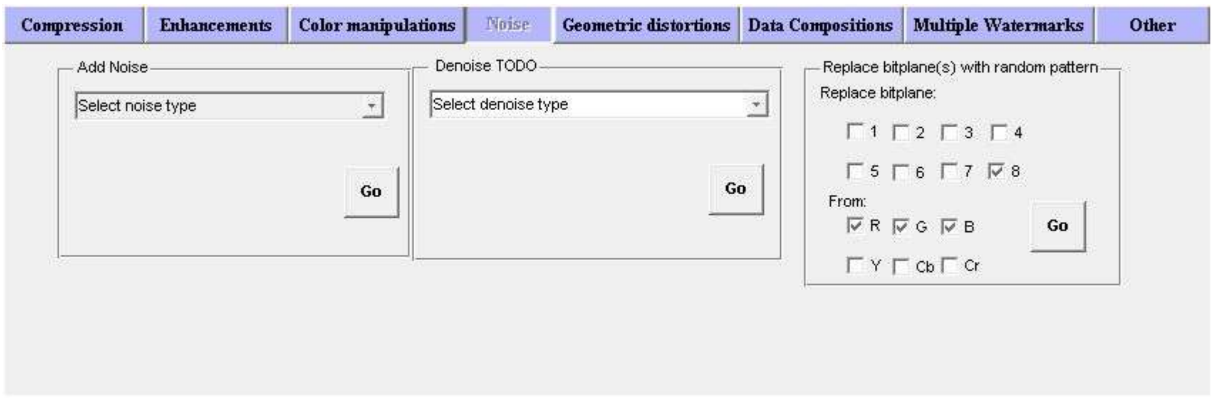


Figure 51: Noise attacks

The noise attacks add a random noise pattern to the watermarked image. Different types of noise can be chosen: Gaussian White Noise, Zero-mean Gausssian White Noise, Poisson Noise, Salt & Pepper Noise and Speckle Noise. The Gaussian White Noise attack uses two parameters: mean and variance. Mean is the intensity and varies between 0 and 1, and the variance determines the range of the added noise. Both Gaussian Noise types and Speckle Noise can be adjusted by sliding the variance. The Salt & Pepper Noise uses noise density as parameter. This noise density varies between 0 and 1 and determines the number of adapted pixels. In figure 52 the Gaussian White Noise attack is performed with mean equal to 0 and variance equal to 0.01.

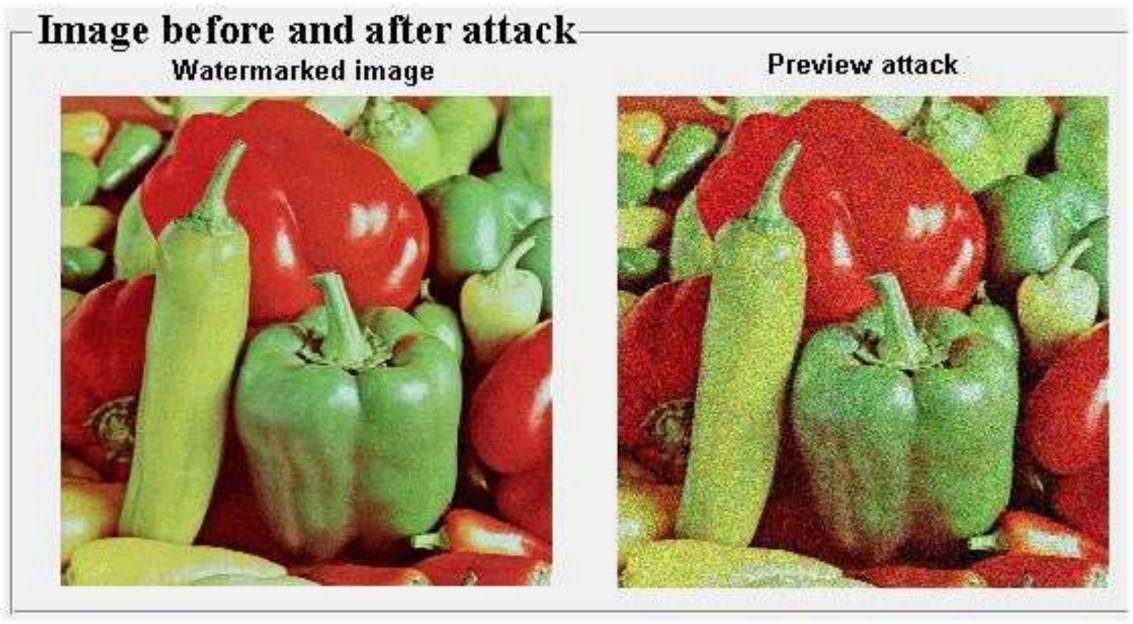


Figure 52: Gaussian White Noise attack

13.5 Geometric distortions

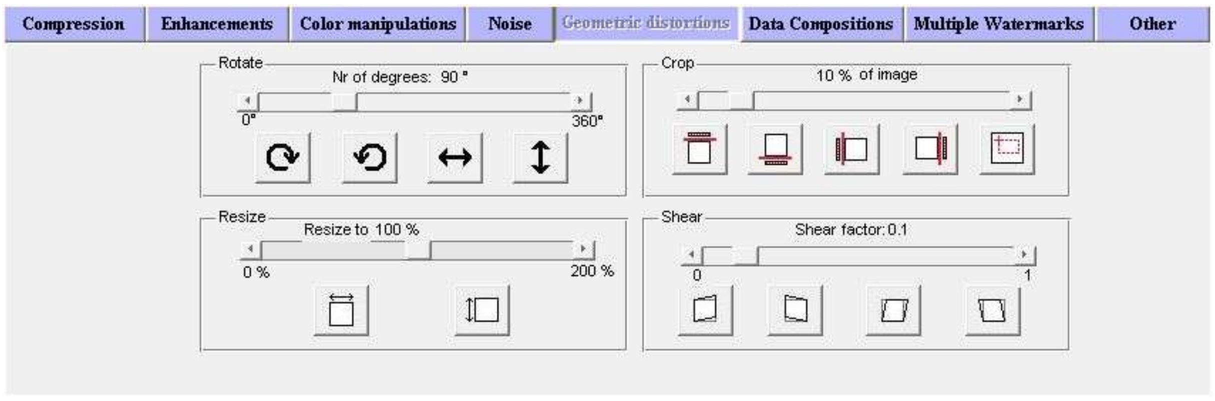


Figure 53: Geometric distortions

There are four geometric distortions available: rotation, cropping, resizing and shearing. The rotation attack rotates the image by a maximum angle of 360°. It is also possible to flip the image in horizontal or vertical direction. The cropping attack crops the image by deleting a part of the image. The slider and buttons allow to choose the size and location of the deleted part. The resizing attack rescales the image, both in horizontal and vertical direction. Finally the shear attack transforms the image as in figure 54. It is also possible to combine these geometric attacks.

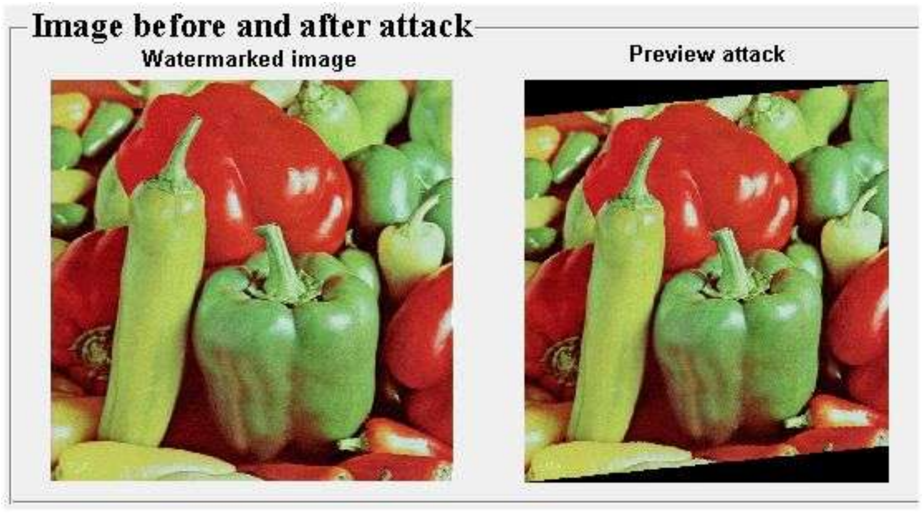
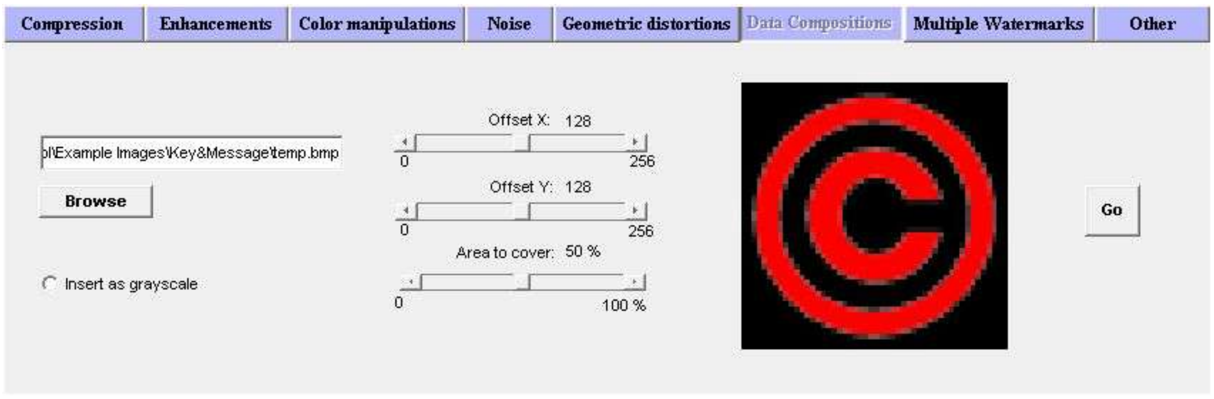


Figure 54: Shear attack

13.6 Data compositions



ol\Example Images\Key&Message\temp.bmp

Insert as grayscale

Offset X: 128 (0–256)

Offset Y: 128 (0–256)

Area to cover: 50 % (0–100 %)

Figure 55: Data compositions

The data composition attack consists of pasting an image on the watermarked image. By choosing an image with the button Browse, the settings in figure 48 appear. The X and Y offset of the image set the location where the image will be pasted. The total covered area by the pasted image can be set by the bottom slider. After clicking the button Go this image is pasted in the middle of the watermarked image (figure 56).

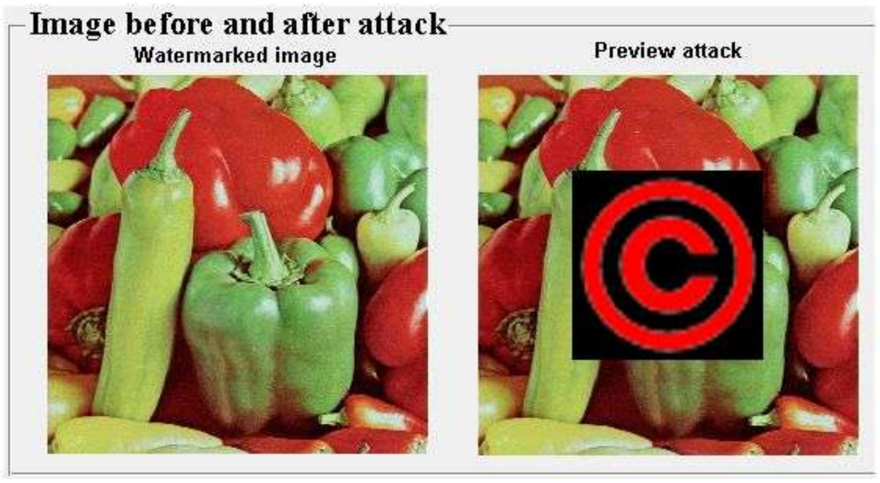


Figure 56: Pasted copyright image

13.7 Multiple watermarks

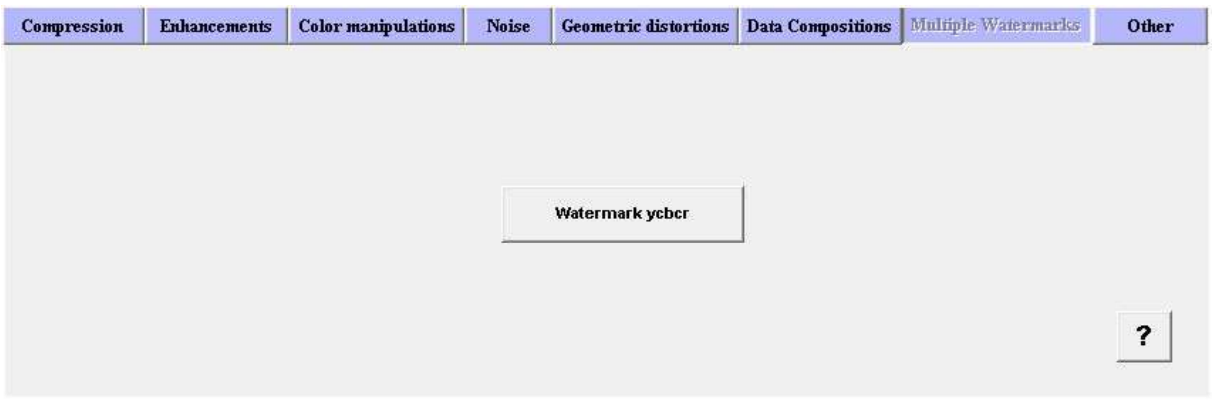
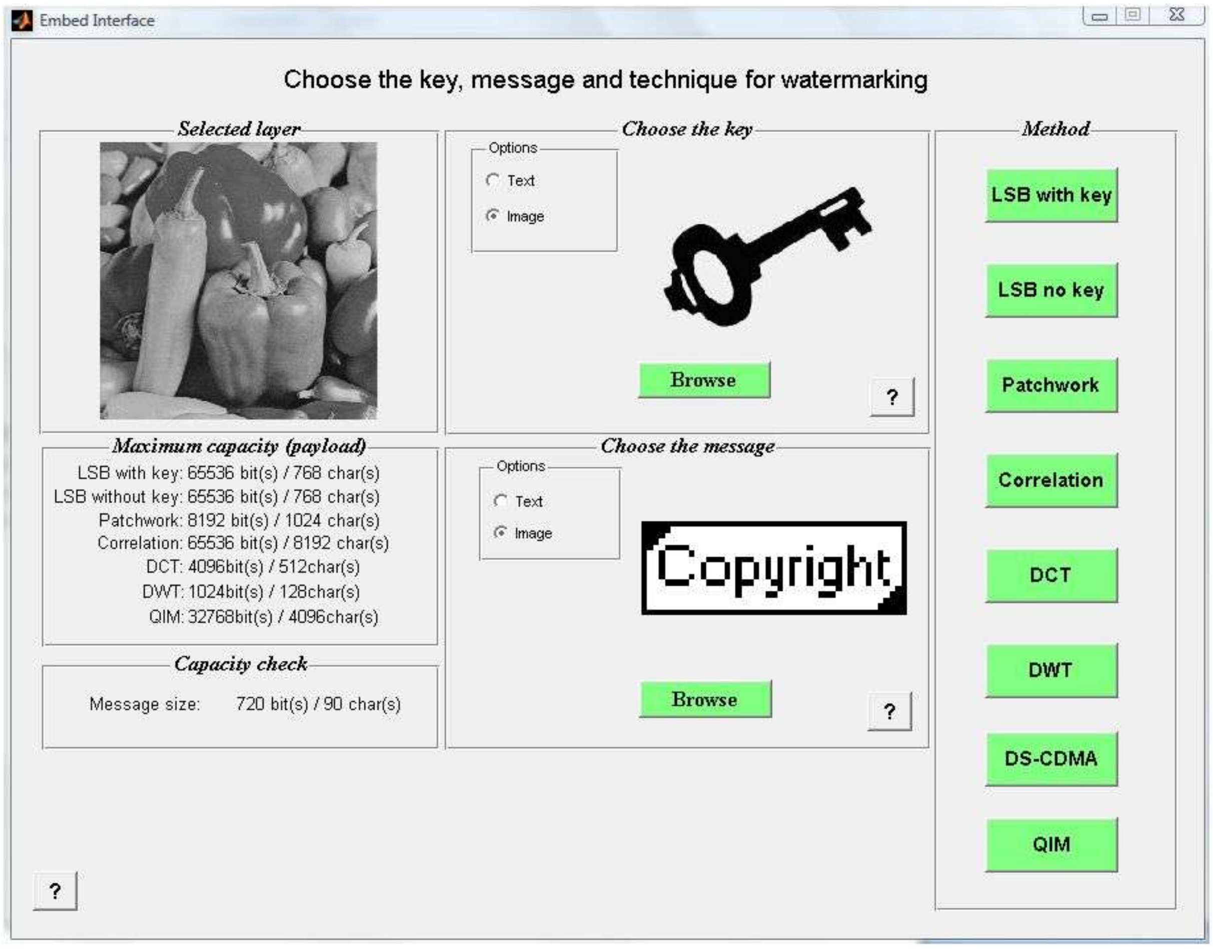


Figure 57: Multiple watermarks

This attack will add an additional watermark to the watermarked image. By clicking on the button Watermark ycbcr, the watermarking GUI will be opened and the user can select key, message and watermarking technique (figure 58).



Maximum capacity (payload)
LSB with key: 65536 bit(s) / 768 char(s)
LSB without key: 65536 bit(s) / 768 char(s)
Patchwork: 8192 bit(s) / 1024 char(s)
Correlation: 65536 bit(s) / 8192 char(s)
DCT: 4096bit(s) / 512char(s)
DWT: 1024bit(s) / 128char(s)
QIM: 32768bit(s) / 4096char(s)

Capacity check
Message size: 720 bit(s) / 90 char(s)

Figure 58: Add an additional watermark

13.8 Other attacks

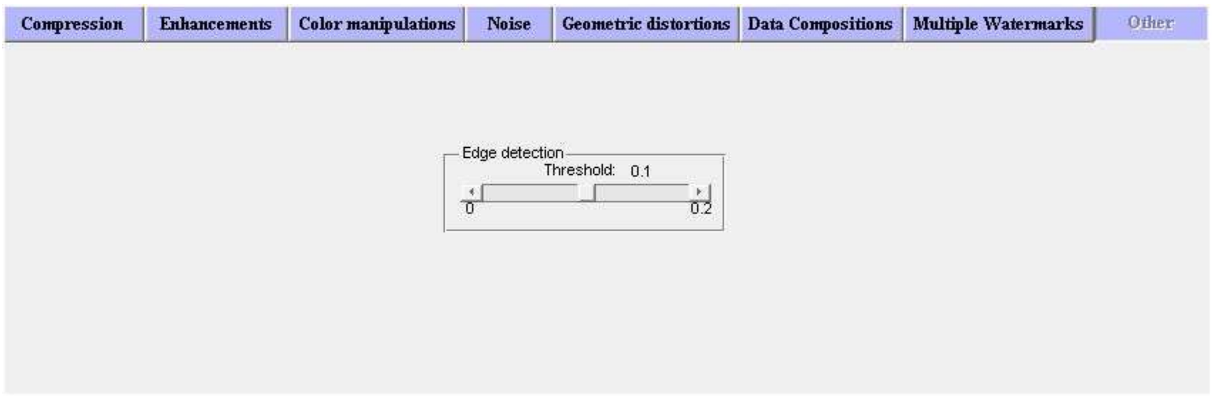


Figure 59: Other attacks

Finally the user can choose to perform edge detection on the watermarked image. The threshold can be adjusted from 0 to 0.2. Almost all edges will be detected at a threshold of 0, while almost no edges will be detected at a threshold of 0.2. In figure 60 edge detection is applied with a threshold of 0.05.

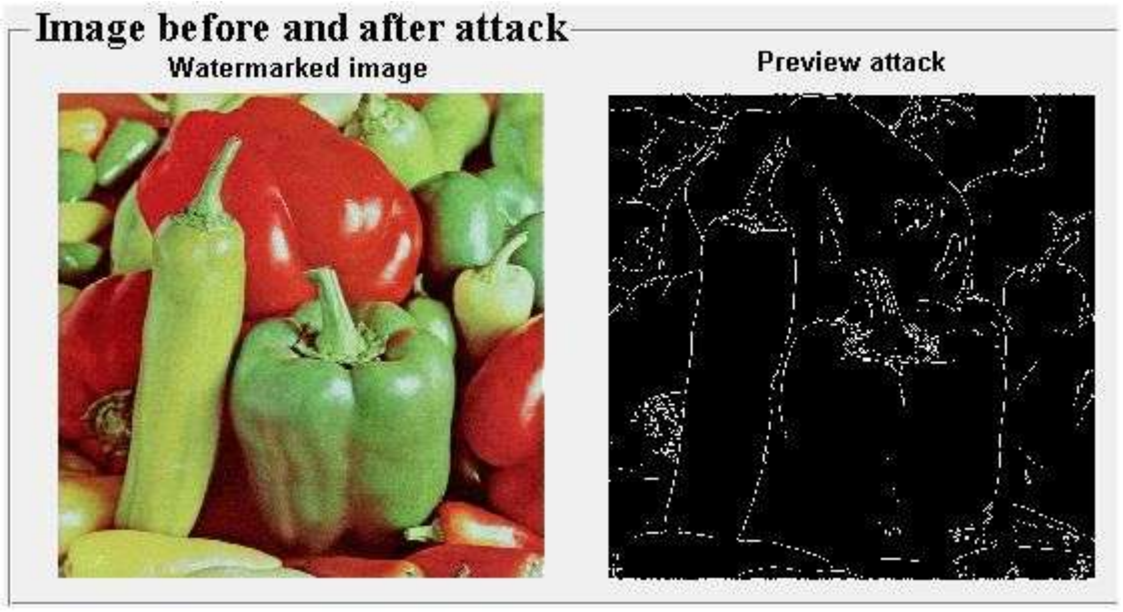
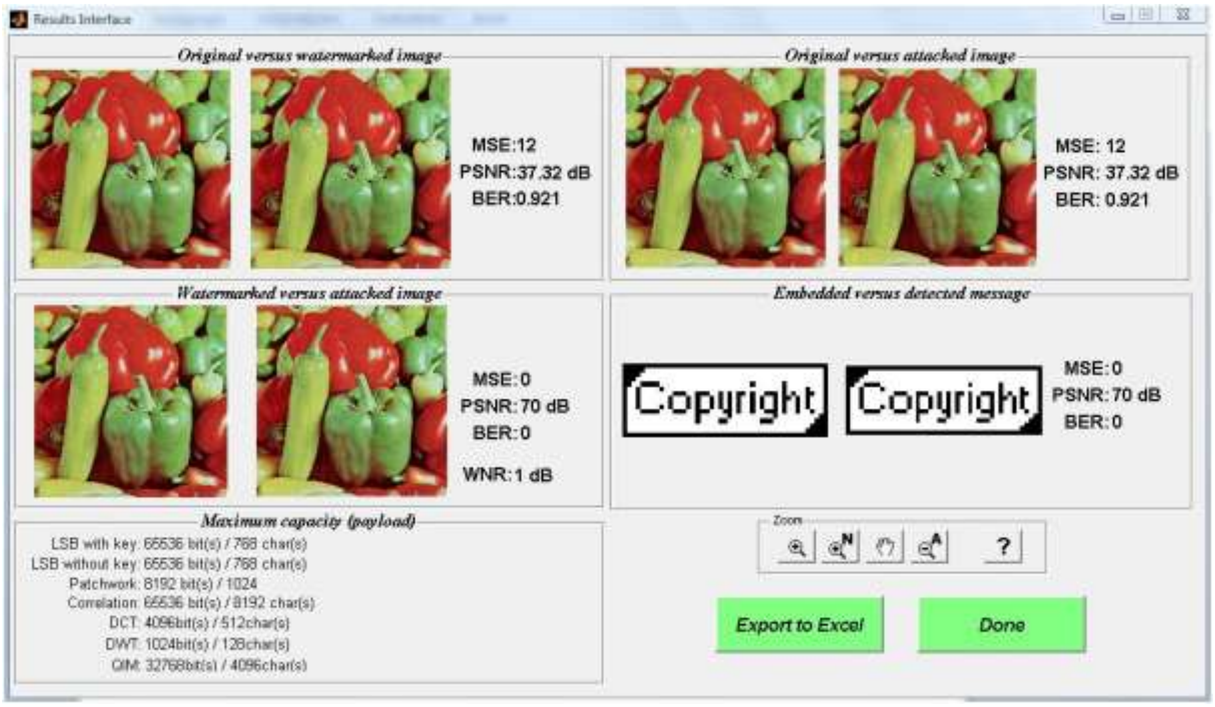


Figure 60: Example of edge detection

14 Overview GUI

In figure 61 the overview GUI is shown. This GUI can be opened by clicking on the Overview button in the main GUI.


Results Interface
Original versus watermarked image
Original versus attacked image
Watermarked versus attacked image
Embedded versus detected message
Copyright
Copyright
Zoom
?
Export to Excel
Done

```
MSE:12
PSNR:37.32 dB
BER:0.921

MSE: 12
PSNR: 37.32 dB
BER: 0.921

MSE:0
PSNR:70 dB
BER:0
WNR:1 dB

MSE:0
PSNR:70 dB
BER:0
```

Maximum capacity (payload)

```
      LSB with key: 65536 bit(s) / 768 char(s)
   LSB without key: 65536 bit(s) / 768 char(s)
         Patchwork: 8192 bit(s) / 1024
       Correlation: 65536 bit(s) / 8192 char(s)
               DCT: 4096bit(s) / 512char(s)
               DWT: 1024bit(s) / 128char(s)
               QIM: 32768bit(s) / 4096char(s)
```

Figure 61: Overview GUI

The overview GUI shows the parameters MSE, PSNR, BER, WNR and capacity between different steps in the watermarking process. MSE, Mean Square Error, indicates the average difference in pixel values between two images. PSNR, Peak Signal to Noise Ratio, calculates the difference between two images and compares it with the maximum pixel value (255 or 1). BER, Bit Error Rate, calculates the number of pixels that have a different value in two images. Finally WNR, Watermarked to Noise Ration, is the PSNR ratio before and after attacking the watermarked image. The capacity determines how much information (bits) can be embedded in the cover work. By clicking on the button Excel, these values are exported to Excel.

15 Zoom functions

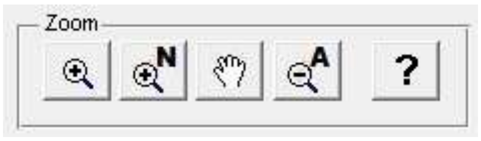


Figure 62: Zoom interface

The zoom panel can be found in most of the GUI's of the program. The user can use the different zoom functions to zoom in on images shown in the axes of the GUI's.

With the zoom in button the user can zoom in on all images or axes in the GUI's. First click on this button, then on an image to perform the zoom. After clicking one axes, all axes in the GUI will zoom to the same region. The scroll mouse button can also be used to zoom. By scrolling up you zoom in, scrolling down zooms out. It is also possible to zoom out by clicking on the image and pressing the CTRL button. Double clicking on an image restores an image to the original zoom.

This 'zoom in new figure' button opens the active image or axes (last shown or clicked image or axes) in a new figure window. In this window basic Matlab functions are provided.

After clicking the pan button, it is possible to explore the zoomed image.

By clicking the zoom out button, all images or axes in the figure will be zoomed out. With this button the original zoom of each image in the GUI will be restored.