SGN-1650/1656 Signal Processing Laboratory

Digital watermarking of images

In this exercise, we will study the digital watermarking of images. The exercise is categorized in different sections where various basic watermarking techniques are introduced. A necessary prerequisite for this exercise is some basic course in digital image processing.

After finishing the exercise, return by email your well-commented MATLAB functions and written report. In your report, you should describe what you have done, give answers to the predefined questions (**1-8**), and give some comments about the exercise. Answers to the questions can (and should) be deduced without any references. The written report should resemble more of an essay than just a discrete list of answers to the questions. Remember also to include your name(s), student number(s) and email address(es) to the report. It is recommended that the report is returned in pdf format.

NOTE: All required functions (**tasks 1-5**) should be implemented using MATLAB as separate m-files. Input parameters for the functions are specified separately for each of the tasks, but in general, the required input images must be matrices, not names of the image files. Each of the functions must have one output parameter: the result of the current task.

1 Introduction to watermarking

Fundamentally, watermarking can be described as a method for embedding information into another signal. In case of digital images, the embedded information can be either visible or hidden from the user. In this exercise, we will concentrate on hidden watermarks. Typical usage scenarios for watermarking are e.g. copyright protection and data authentication.

Question 1. Why is the watermark embedded into image instead of just inserting it into the file header?

Question 2. What is the drawback of inserting additional information into images?

1.1 Steganography

Steganography includes different methods for hiding the existence of additional information in an innocuous signal. The distinction between steganography and watermarking is not always clear. Basically, in case of watermarking the additional information is used to protect the original image (e.g. in case of copyright management), whereas in the steganography the image is used to protect the additional information (e.g. secret message). By definition, the visible watermarks are not included in steganography. Next, we will go through a basic example of information hiding, where a 'secret' RGB image is hidden in the least significant bits of a cover RGB image.

Task 1. Make a function that replaces the N least significant bit (LSB) planes of the cover image by the N most significant bit (MSB) planes of the hidden image

(`help bitand`). The function should take as input a cover image, a hidden image and a parameter N. The name of the function should be `hide_image`.

Question 3. Using the function made in Task 1, how many bytes of the image *cup.tif* you can hide into the image *baboon.tif*? How about vice versa?

2 Watermarking techniques

During the past ten years, digital watermarking has attracted the attention of numerous researchers. As a result, hundreds of studies have been published concerning the different methods for watermarking.

The information embedded as a watermark can be almost anything. It can be a bit string representing copyright message, serial number, plain text, etc. However, sometimes it can be more useful to embed a visual watermark (e.g. corporate logo) instead of a bit string as a watermark.

Question 4. What is the benefit of using visual watermark e.g. instead of hiding bits representing some ASCII characters?

Task 2. Design your own 64-bit visual watermark W. Create the watermark of your choice as a 8×8 matrix consisting of zeros and ones as illustrated below. Make a function called `generate_wm` that returns your watermark as an output. We will use your watermark in the following algorithms.

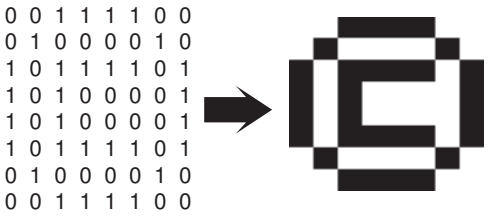

Figure 1: Example of 8×8 visual watermark

2.1 Least-significant bit modification

One of the first techniques for watermarking is the least-significant bit modification. It is based on the substitution of LSB plane of the cover image with the given watermark. The approach is basically the same as the one demonstrated in the Section 1.1. However, here we will use grayscale images.

Task 3. Make a function that replaces the least-significant bit plane of a grayscale cover image and replaces it with your binary watermark. Since your watermark is smaller than the cover image, you need tile several copies of you watermark together in order to achieve the correct size. You can assume that the dimensions of the cover image are divisible with the dimensions of the watermark. The function should take as input a cover image and a watermark. Thereafter, make a function that extracts your watermark from a watermarked image. The function should take as input a watermarked image and give the extracted watermark as output. The functions should be called `embed_lsb` and `extract_lsb`. You can test your algorithm with *lena.tif*.

2.2 Removal of watermarks

It is desirable that the watermark can not be removed from the cover image. However, several intentional and unintentional operations with the watermarked image may provide possibility for disabling the watermark. Commonly, these operations (especially the intentional ones) are referred as attacks against watermarks. In this exercise, we can not go through all possible threats against watermarking techniques and since we will concentrate on couple of very basic ones. Two common attacks we will use are JPEG compression and noise addition.

Question 5. If you consider the watermarking technique introduced in Section 2.1, what operations could destroy the embedded watermark (in addition to the above-mentioned ones)? Give at least three additional operations.

Question 6. Embed a LSB watermark into image *lena.tif* using the function implemented in Section 2.1. Add Gaussian noise into watermarked image, and try to extract the watermark. Do the same with the lossy JPEG compression. Comment the results in your report and include some example images.

2.3 Spread spectrum method

Another approach for watermarking utilizes the direct sequence code division multiple access (DS-CDMA) spread spectrum communications. First, the watermark is transformed into a bit string $b_1 b_2 ... b_{64}$. In our case, you will need to transform your two-dimensional watermark into a vector of length 64. For each bit b_i, a pseudorandom matrix R_i of integers $\{-1, 1\}$ is generated (`help rand`). *Before* the generation, MATLAB's random number generator must be initialized using a predefined seed: `rand('seed', seed)`. This initialization enables the creation of exactly the same random matrices later on in the extraction phase. As a result, we will have 64 different matrices consisting pseudorandomly of ones and minus ones, each of the same size as the original image.
Next, each of the R_i matrices will depend on the bit values b_i of the original watermark in the following way: A matrix $+R_i$ is used if b_i represents a 0, and a matrix $-R_i$ is used if b_i represent a 1. In other words, multiply the necessary R_i matrices by -1.

Thereafter, the sum of all random patterns R_i defines the watermark W:

$$W = \sum_{i=1}^{64} \pm R_i,$$

where the sign of each pseudorandom matrix is dependent on the bit value as defined in the previous paragraph. Finally, the watermarked image I_W is generated by adding the watermark into cover image I

$$I_W = I + kW,$$

where the scalar k is a gain factor which can be used to control the watermark strength.

The embedded watermark can be extracted one bit at a time by calculating the correlation between normalized watermarked image and the pseudorandom pattern R_i. In this exercise we will calculate the correlation C_i between the normalized image I'_W and the pseudorandom pattern R_i as

$$C_i = \sum((R_i - \overline{R}_i) \cdot I'_W),$$

where $\overline{R_i}$ is average of all values in R_i . The operator $\cdot$ is used for element-wise matrix multiplication, and the operator $\sum$ to calculate a sum of matrix elements. The normalized watermarked image is defined as

$$I'_W = (I_W - \overline{I_W}),$$

where $\overline{I_W}$ is the average of all values in I_W. If the resulting correlation C_i is positive the value 0 is assigned to the corresponding bit, and otherwise the value 1 is used.

Task 4. Make function `embed_cdma` for embedding a watermark according to the spread spectrum method. The function should take four inputs: a cover image, a watermark, parameter k (the gain factor), and seed for the random number generator. Thereafter, make function `extract_cdma` for extracting the watermark. The function should take as input a watermarked image and a seed for random number generator. In order to get the same random numbers in embedding and extraction functions, you will need to give the same seed as an input parameter for both functions.

Question 7. Embed your watermark with CDMA method into *lena.tif* using value 1 for the gain factor k. Thereafter, add Gaussian noise with zero mean into CDMA watermarked image and increase the noise variance gradually. Make sure that the variable type of the image (`double` or `uint8`) is correct before adding the noise to avoid problems. What is the maximum noise variance for extracting an undamaged watermark? Thereafter, compress the original watermarked image with lossy JPEG. Raise the compression rate (reduce the quality) gradually. How high compression rate you can use so that you can still extract undamaged watermark? Include some example images in your report. Try also other values for k and comment your results.

2.4 Image authentication

An interesting application for digital watermarking is image authentication. With the help of advanced image editing softwares, digital images can be manipulated maliciously. Thus, it is essential to be able to detect unauthorized image manipulations similarly as illustrated in Figure 2. In image authentication, a specific watermark is inserted into image, so that all attempts to manipulate the content of the image will alter the watermark also.

Figure 2: An example of image manipulation and detection. (A) Original image has been manipulated in (B). In (C) the algorithm has detected and highlighted the manipulated regions of the image.

We will use embedded check-sums for the watermarking. The algorithm will process the image in 8×8 blocks and calculate a specific check-sum watermark from the 7 most significant bit planes of each block. Thereafter, the check-sum is embedded in the least significant bit plane of the corresponding block. Note that the exact details of the implementation can vary to some extent, because both the embedding and extraction phases use the same algorithm. Just make sure when implementing the algorithm below that only the 7 most significant bit planes are used in the embedding, and only the least significant bits are used in storing the check-sum and extraction of the watermark.

The algorithm for embedding the watermark is following

1. *Select a large integer N.* For this exercise, select an integer with four digits.
2. *Process image in* 8×8 *blocks*
3. *For each block:*
   - 3.1. *Define a pseudorandom walk through all 64 pixels in the current block. Denote the pixels as* $p_1, p_2, ..., p_{64}$. (`help randperm`)
   - 3.2. *Generate 64 pseudorandom integers* $a_1, a_2, ..., a_{64}$ *with similar magnitude with* N. That is, generate the integers so that the mean of all values is approximately N ($\mu_a \approx N$).
   - 3.3. *Calculate the check-sum* S *given as*

$$S = \sum_{j=1}^{64} (a_j \cdot g(p_j)) \text{mod} N$$

     where $g(p_j)$ *is the intensity obtained from the seven most significant bit planes of pixel* p_j
   - 3.4. *Embed the binary form of* S *into the LSB plane of the corresponding image block.*

For extraction, the process is repeated with minor differences. The check-sum of seven MSB planes for each block is calculated and compared to the one recovered from the LSB plane. If the check-sums are not equal, the corresponding block has been manipulated.

Task 5. Make function `embed_auth` for embedding the image authentication information into image. The function should take as input a image and a seed for random number generator. Thereafter, make function `extract_auth` for extracting authentication information from image. The function should take as input a possibly manipulated image and a seed for the random number generator. The extraction function should return an image where each manipulated block has been highlighted (increase the pixel value with some constant). Test your algorithm with $lena.tif$ and include some example images in your report.

Question 8. What information should an external viewer of the image possess, in order to fool the current image authentication system?

References

[1] G. Langelaar, I. Setyawan, and R. Lagendijk. Watermarking digital image and video data. *IEEE Signal Processing Magazine*, 17:20–46, 2000.

[2] C. Rey and JL. Dugelay. A survey of watermarking algorithms for image authentication. *EURASIP Journal on Applied Signal Processing*, 6:613–621, 2002.