

# Robust Labeling Methods for Copy Protection of Images

Gerrit C. Langelaar, Jan C.A. van der Lubbe, Reginald L. Lagendijk  
Department of Electrical Engineering, Information Theory Group  
Delft University of Technology  
P.O. Box 5031, 2600 GA Delft, The Netherlands  
E-mail: {*gerhard, vdlubbe, inal*}@it.et.tudelft.nl

## Abstract

In the European project SMASH a mass multimedia storage device for home usage is being developed. The success of such a storage system depends not only on technical advances, but also on the existence of an adequate copy protection method. Copy protection for visual data requires fast and robust labeling techniques. In this paper, two new labeling techniques are proposed. The first method extends an existing spatial labeling technique. This technique divides the image into blocks and searches an optimal label-embedding level for each block instead of using a fixed embedding-level for the complete image. The embedding-level for each block is dependent on a lower quality JPEG compressed version of the labeled block. The second method removes high frequency DCT-coefficients in some areas to embed a label. A JPEG quality factor and the local image structure determine how many coefficients are discarded during the labeling process. Using both methods a perceptually invisible label of a few hundred bits was embedded in a set of true color images. The label added by the spatial method is very robust against JPEG compression. However, this method is not suitable for real-time applications. Although the second DCT-based method is slightly less resistant to JPEG compression, it is more resistant to line-shifting and cropping than the first one and is suitable for real-time labeling.

**Keywords:** Watermarking, Copy Protection

## 1. Introduction

The objective of SMASH-project, sponsored by the European Union under the ACTS-project program, is to develop a mass multimedia storage device for home usage. The development rate and success of such a digital mass storage system depends not only on technical advances, but also on the existence of an adequate copy protection method. Service providers will not offer services in digital form without a copy protection method which limits duplication of multimedia data.

A copy protection system called SCMS [1] (Serial Copy Management System) exists for digital audio recorders, like the DAT, DCC and minidisk recorders. This system adds a subcode to the data during the recording process. This subcode indicates that the data is already copied or not. If the consumer wishes to make a copy of a digital audio source, the recorder checks the subcode (if present) and refuses to record the data if the subcode indicates that the data was already copied before. Using this system, a consumer can make digital copies of any digital source, but copies can not be duplicated further using storage devices equipped with SCMS.

Since the subcode is separately stored from the data, it can easily be removed or altered by a simple computer program without affecting the quality of the data. For a multimedia recorder that has an interface to a personal computer, it is therefore better to encode this subcode directly into the multimedia data itself, by labeling the multimedia data. In this case, the objective is that the protection can not be removed without affecting the quality of the multimedia data and that the protection also remains intact across different data file formats. The current work in SMASH concentrates among other things on developing new robust real-time labeling methods for digital compressed video and images that can be used to realize a copy protection system.

Several labeling methods for labeling images can be found in literature. Most methods add the label in the spatial [2..5] or the Discrete Cosine Transform (DCT) [6..8] domain. To extract a label some of these methods only use the labeled image, where others use the labeled image together with the original one [8]. For copy protection we can only use the first kind of methods. The requirements for a label are discussed in section 2. In section 3 the image test set is described, which is used for the experiments. In section 4 and 5 two new labeling techniques are proposed. The first method extends the spatial labeling method of Pitas and Kaskalis [3]. This method adds high frequency noise to the image to embed a label. The second technique removes high frequency DCT-coefficients in some areas of the image to embed a label. Experiments are performed to see if the methods meet the requirements. Finally, the results are discussed in section 6.

## 2. Requirements

For a copy protection system as described in the introduction we need a labeling method, that meets several requirements. A labeling method is needed that:

- allows the extraction of the label without using the original image,
- can be performed directly on compressed data to add and extract a label in real-time,
- is perceptually invisible,
- is resistant to JPEG/MPEG compression up to a certain level, which can be determined beforehand,
- is resistant to simple processing techniques, which do not seriously reduce the quality of the image, like filtering, line-shifting and cropping,
- can store a few hundred bits in one image.

The first two requirements are inevitable. A recorder does not have the original data to its disposal to extract the label and re-encoding the data to add the label is too time consuming and expensive. The other requirements are more or less related with each other and ranked in order of importance. First, the label must be perceptually invisible, because a copy protection system that spoils the image quality is not acceptable. After that, a trade-off should be made between robustness and label size. Among the last requirements, the robustness against JPEG compression is the most important one, because almost all image data is offered to the consumers in this format. If the data is offered in another format, the consumers will convert it to this format to save storage space. Robustness against other processing techniques is less important, because it requires more user interaction.

A label size of one bit is enough for a SCMS-like copy protection system, but if a larger label size is available, extra features can be added, for instance the serial number of the recorder can be stored in the data to trace back illegal copies. The extra bit capacity can also be used to apply error correcting codes to the label. This improves the robustness of the label.

## 3. Image Test Set

For the experiments a set of nine true color images [9] is used. One image contains hardly any textured areas and sharp edges ("Fog in hills"), other images are also rather smooth ("Lena", "Grand Canyon"). The image "Bike" contains many sharp edges and much detail, the image "Tree" has a large textured area of leaves and branches, which covers about 70% of the image, however the background contains no texture at all. The images "Bridge", "Red square" and "House" contain all kind of areas: smooth skies, objects with sharp edges, texture etc. The image "House" also contains many lines (bricks etc.). The image "Diver" was already compressed using the JPEG algorithm.

This rather big test set with a wide variety of contents is chosen, because some labeling methods, which perform well on the "Lena" image, can not be applied to other images like "Fog in hills", since they cause a lot of perceptually visible artefacts.

## 4. Spatial Labeling Method

In this section a new block based method is discussed, which adds a label in the spatial domain. Two methods can be found in literature, which embed a label of one bit in this domain. Bender *et al* [2] describe a statistical labeling method called "Patchwork". Using this method,  $n$  pairs of image points  $(a_i, b_i)$  are randomly chosen. The brightness of  $a_i$  is increased by one and the brightness of the corresponding  $b_i$  is decreased by one. For a labeled image, the expected value of the sum of the differences of the  $n$  pairs of points is then  $2n$ . The authors show that after JPEG compression, with the quality factor set to 75, the label can still be decoded with a probability of recovery of 85%.

Pitas and Kaskalis [3] describe a similar method. Using this method the picture is split in two subsets of equal size (for example by using a random generator) and the brightness of the pixels of one subset is altered by adding a positive integer constant  $k$ . This constant  $k$  is calculated using the sample variances of the two subsets. To check the label, the difference between the means of the two subsets of pixels is calculated. The expected value is  $k$  if a label was added, otherwise zero. This method is only resistant to JPEG compression ratios up to 4:1 (quality factor of more than 90).

Smith and Comiskey use a Direct-Sequence Spread Spectrum method [4], which divides the image first into blocks to store more label-bits in an image. Each block contains one of the label-bits. A modulation function, a constant integral valued gain factor  $G$  multiplied by a pseudo-random block of bits, either +1 or -1, is added to each image block. A positive gainfactor  $G$  is used to embed label-bit "1" in an image block, otherwise a negative gainfactor  $G$  is used. The label is recovered by demodulating with the modulation function. According to the authors the label is resistant to "mild JPEGing". By JPEG compressing the pseudo-random carrier before labeling the image, the label becomes more resistant to JPEG compression. In the next subsection a new iterative method is proposed, which is more robust against JPEG compression. In subsection 4.2 the robustness of this method is experimentally tested.

#### 4.1 Iterative Spatial Labeling Method

The new labeling method is based on the method of Pitas and Kaskalis [3], which adds a positive integer constant  $k$  to the brightness of 50% of the pixels in an image. This constant  $k$  is called the label embedding-level. By dividing the image into blocks and searching an optimal  $k$  for each block instead of using a fixed  $k$  for the complete image, a larger and more robust label can be embedded in an image. The embedding level  $k$  for each block is dependent on a lower quality JPEG compressed version of the labeled block. Different variants of this method have been tested, but the method as described below combines high resistance to JPEG compression with a low visibility of the label.

##### Labeling procedure:

A label consists of a few hundred bits. Each label-bit is embedded in a block  $B$  of luminance values. The width and height of this block  $B$  are multiples of 8. Each luminance block  $B$  has a distinct location in the image, the blocks are non-overlapping and tile the image without gaps. The algorithm becomes more robust against JPEG (JFIF) compression, if a higher threshold  $T$ ,  $k_0$ ,  $k_{max}$  and a lower quality factor  $Q$  is chosen. However, the label will become visible in this case.

1. First the RGB color image is converted to the YUV domain. Only the Y (luminance) values are labeled, because the U and V (chrominance) values are more affected by the JPEG compression algorithm than the Y values.
2. A square block  $B$  is pseudo-randomly selected from the Y-image to embed one label-bit.
3. A fixed binary pseudo-random pattern  $P$  of the same size as the block is generated, consisting of the integers “0” and “1”. The embedding-level  $k$  is set to  $k_0$ .
4. If label-bit “0” must be embedded,  $R = B - k * P$   
If label-bit “1” must be embedded,  $R = B + k * P$
5. The mean  $I_0$  is calculated of those luminance values in block  $R$ , where the random pattern  $P$  is 0. The mean  $I_1$  is calculated of those luminance values in block  $R$ , where the random pattern  $P$  is 1. After that, the high quality difference  $D_{high} = I_1 - I_0$  is calculated.
6. The values of block  $R$  are temporarily stored in  $Tmp$ . The quality of block  $Tmp$  is reduced by taking the 8x8 DCT transform, quantizing the coefficients with a certain quality factor  $Q$  followed by an inverse DCT transform. The difference  $D_{low} = I_1 - I_0$  is calculated for this low quality block  $Tmp$  in the same way as it is done in step 5.
7. If label-bit “0” must be embedded and one of the two differences ( $D_{high}$  or  $D_{low}$ ) exceeds the value zero, the embedding level  $k$  is increased by 1 and the procedures (4..7) are repeated iteratively until both differences are below zero or  $k$  exceeds  $k_{max}$ .  
If label-bit “1” must be embedded and one of the two differences ( $D_{high}$  or  $D_{low}$ ) is smaller than a certain threshold  $T$ , the embedding level  $k$  is increased by 1 and the procedures (4..7) are repeated iteratively until both differences exceed  $T$  or  $k$  exceeds  $k_{max}$ .
8. Block  $B$  is replaced by block  $R$ .
9. The procedures (2..8) are applied to all pseudo-randomly selected blocks until all bits of the label are embedded.
10. Finally the YUV values are converted back to the RGB domain.

##### Label extracting procedure:

1. First the RGB color image is converted to the YUV domain.
2. A block  $B$  is pseudo-randomly selected from the image to read out one bit.
3. The fixed pseudo-random pattern  $P$  of the same size as the block is generated, consisting of the integers “0” and “1”.
4. The mean  $I_0$  is calculated of those luminance values in block  $B$ , where the random pattern  $P$  is 0. The mean  $I_1$  is calculated of those luminance values in block  $B$ , where the random pattern  $P$  is 1. The difference  $D = I_1 - I_0$ .
5. If this difference  $D$  exceeds the value zero the bit embedded in the block is one, otherwise zero.

6. The procedures (2..5) are applied to all pseudo-randomly selected blocks until all bits of the label are extracted.

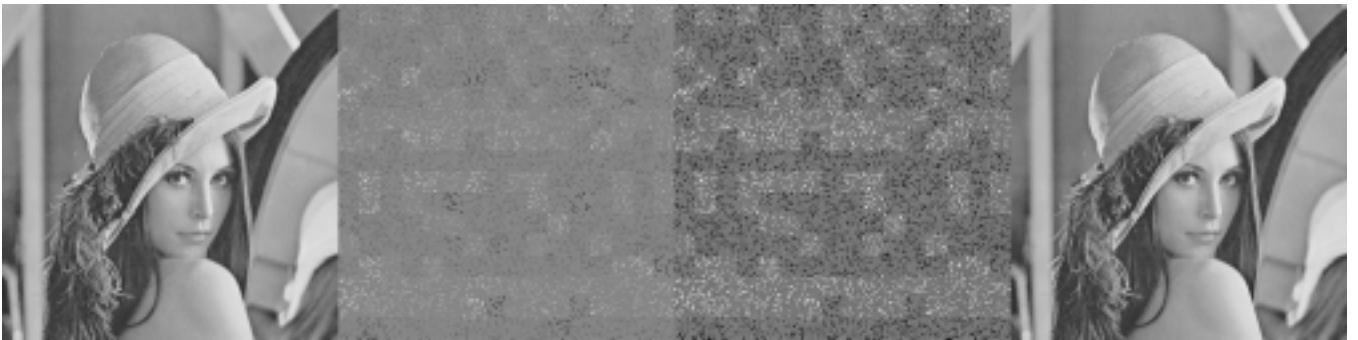
## 4.2 Experimental Results

Using the method described in the previous subsection the image test set is labeled. It appeared that, if the ratio between the numbers of ones and zeros in the random pattern  $\mathbf{P}$  is forced to be 1:5 the labeling is significantly less visible to the human eye, but marginally weaker. If 1:1 patterns are used, clearly visible chess-board patterns can be discovered in smooth areas (e.g. the sky) of the image. Therefore the ratio between the numbers of ones and zeros in the random pattern is forced to be 1:5 for all experiments. To avoid repeating patterns, for each label-bit another pseudo-random pattern is chosen.

Each label-bit is embedded in a block of 32 x 32 pixels. If a smaller blocksize (e.g. 16x16) is chosen, more label-bits can be embedded in an image, but the label is less resistant to JPEG compression in this case. Obviously, the resistance to JPEG compression increases, if a larger blocksize is used (e.g. 64x64). However, chess-board patterns appear in the smooth areas of the images. In this case, the blocksize is too large to adapt the embedding-levels  $k$  to the content of the blocks. For example, if a high value  $k$  is needed for a large smooth block, because of a small textured area in the corner of this block, it is better to split the block into 4 smaller blocks. Now, 3 lower and 1 higher embedding-levels can be chosen instead of one fixed embedding-level for the complete block, which is actually too high for 75% of the block. So, the smaller the blocksize, the better the algorithm adapts its embedding-levels  $k$  to the content of the image and the bigger the blocks, the more resistant the label is to JPEG compression. A blocksize of 32x32 is a good trade-off between the visibility, size and robustness of the label in our test set.

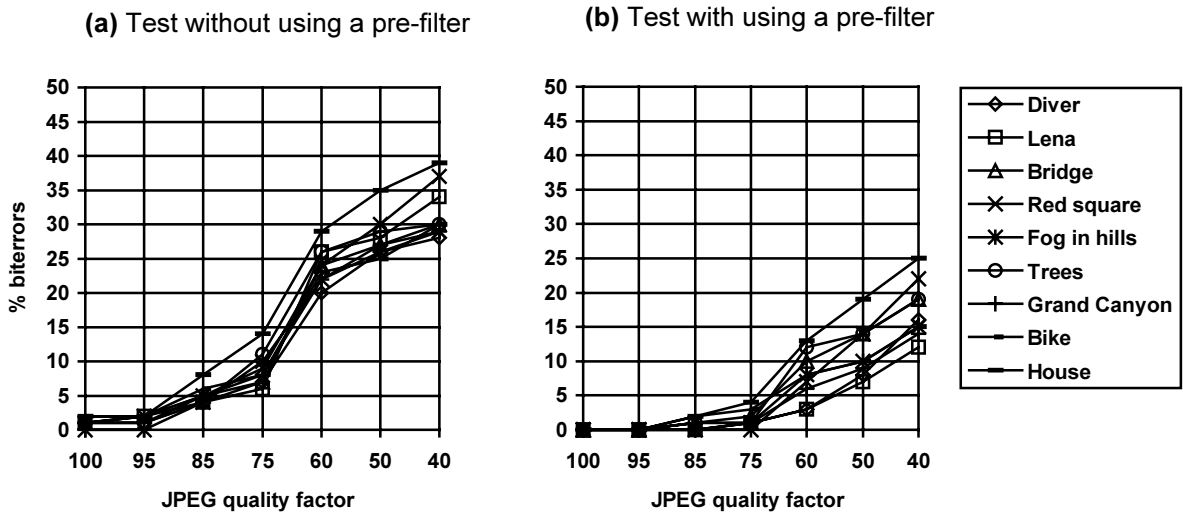
The threshold  $T$  is set to 1 and a quality factor  $Q$  of 75 is used. The label size is dependent on the size of the image. In the image “Diver” (302 x 323) 90 bits are embedded, in “Lena” (512 x 512) 256 bits and in all other images (768x 512) 384 bits. The initial embedding-level  $k_0$  is set to 4 and the maximum embedding-level  $k_{max}$  is 10. If  $k_{max}$  is not limited the label is more robust, but a few (4..10) white or black patterns are visible in the labeled image. By limiting the maximum embedding level to 10, the label is perceptually invisible in all nine images.

To get an idea of how the label looks like, the original unlabeled “Lena” image (Figure 1a) is subtracted from the labeled image (1d) and this difference signal is multiplied by 10 (1b) or by 30 (1c). In the images, which are shown in Figure 1b and 1c, a white dot stands for +128, a black dot for -128 and the gray background for 0.



**Figure 1.** (a) Unlabeled (b) Difference x 10 (c) Difference x 30 (d) Labeled image

In Figure 2 the percentages bit errors are represented, after compressing the images using the JPEG algorithm, with the quality factor set to different values (100..40). If the images are JPEG compressed using a quality factor  $Q=75$ , the compression rates range from 12:1 to 29:1. If the quality factor  $Q=40$  is used, the compression rates are in the range of 20:1 to 55:1.



**Figure 2.** % bit errors after JPEG compression using spatial labeling method, (a) without and (b) with pre-filtering.

Applying a simple  $3 \times 3$  edge-enhance filter to the Y-image before extracting the label considerably improves the results (Figure 2b), because it leaves the low frequency components, which contain redundant information (concerning the label) unchanged and it amplifies the high frequency components in which the label is embedded.

In Figure 2b we see that the percentage errors is below 5% for JPEG quality factors between 100 and 75, after this breakpoint  $Q=75$  the percentage errors increases rapidly. This is quite obvious, because the labeling algorithm estimates the embedding-level of the label also from an image that is first degraded using the JPEG compression algorithm with this quality factor. The maximum percentage of bit errors in the label is only 25% after compressing the image using a quality factor of 40 (compression rates up to 55:1).

To see how the algorithm adapts the embedding-levels, two experiments are performed. First, all images are labeled using a fixed embedding-level  $k=k_\theta$ . The label added with this fixed  $k$  turns out to be perceptually invisible, but if the label is extracted after JPEG compression ( $Q=75$ ), the error percentages vary between 10-20%. After that, the images are labeled with a fixed embedding-level  $k=k_{max}$ . The label added with  $k_{max}$  turns out to be perceptually visible, especially in smooth areas. If the images are labeled using the adaptable  $k$  the error percentage is below 5%. In this case, approximately 20% of the blocks are labeled with an embedding-level  $k$ , which is higher than  $k_\theta$ , these are mainly textured blocks.

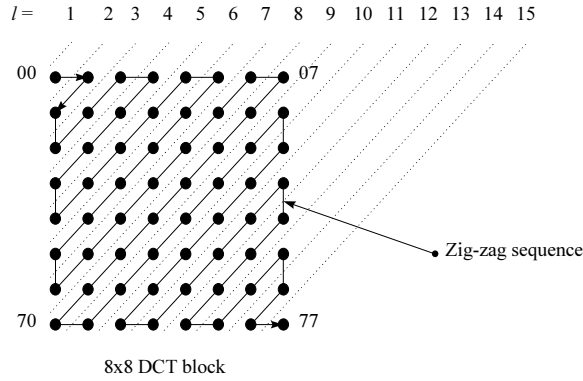
In this subsection the robustness of the label was only tested against JPEG compression. Other attacks to remove the label are also possible, like filtering, adding noise, line shifting, cropping etc. The label is immune to light smoothing and adding pseudo-random noise. However, if the image is shifted or cropped, the label is totally lost.

## 5. DCT-based Labeling Method

In the previous section, we discussed a method that adds high frequency noise in the spatial domain. To label the data with this method, the data has to be uncompressed before labeling and recompressed afterwards. The labeling procedure itself is also quite computational demanding, because of its iterative nature and the DCT / IDCT operations. For real-time applications, the compressed format should be anticipated.

Most visual data is transferred and stored in compressed JPEG (JFIF) or MPEG format to save storage space. The main compression steps for still images [10] and I-frames of moving pictures are listed below:

- The RGB image or frame is converted to the YUV domain.
- The U and V components are subsampled in one or two directions.
- Each component is divided into  $8 \times 8$  blocks and each  $8 \times 8$  block is DCT transformed.
- The 64 coefficients in the DCT blocks are uniformly quantized.
- All coefficients are ordered into the “zig-zag” sequence, shown in Figure 3, and run-level encoded.
- Finally, variable-length codes are assigned to these run-level codes.



**Figure 3.** DCT-coefficients below line  $l$  are set to zero to label the block.

Since the DCT-coefficients can easily be obtained out of the compressed data, it is attractive to add the label directly in the DCT-domain. In this case the DCT / IDCT operations are avoided and only partial decompression and recompression is required. Zhao and Koch [6] propose a method to embed a bitstream in the DCT-domain. This method pseudo-randomly selects 8x8 DCT-blocks to embed a label-bit. Out of each DCT-block, three (low frequency) coefficients are chosen. These coefficients are re-quantized using a coarser quality factor  $Q$  and the standard quantization matrix of the JPEG software. Finally, they are adapted in such a way that they have a certain order in magnitude. For example if a bit '1' must be embedded in a block, the third coefficient must be smaller than the other two. In an earlier proposal by the same authors [7], two instead of three coefficients were used.

The result of the methods [6,7] is that the size of the labeled compressed data can be bigger or smaller than the size of the unlabeled data. We propose therefore a new method which can avoid the partial recompression step and influences the size of the labeled data in a more predictable way. This method embeds the label by simply discarding the last high frequency DCT-coefficients of some selected compressed DCT-blocks. This means that the “zig-zag” ordering, run-level encoding and variable-length coding steps can be omitted for the labeling process and that the size of the labeled compressed data is always smaller than the size of the original compressed data.

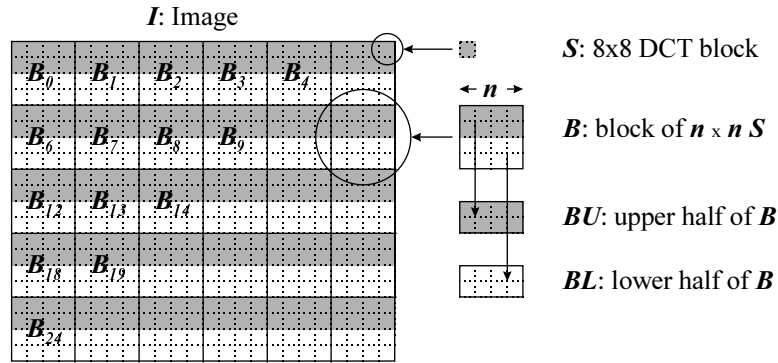
The JPEG algorithm uses a relative small amount of bits to encode these high frequency DCT-coefficients, because the human eye is less sensitive to variations in high frequencies. So, to survive JPEG compression and to achieve the same robustness as in [6,7], where 3 low frequency coefficients in one 8x8 DCT-block are used, more high frequency coefficients and more 8x8 DCT-blocks must be used to embed a single label-bit. For example all coefficients below line  $l$  in Figure 3 must be removed in 4 to 64 8x8 DCT-blocks to embed a label-bit. Although, more coefficients need to be changed to embed a label, the label can still be perceptually invisible, because the human eye is less sensitive to these high frequency coefficients.

An extra advantage of this approach is that removing high frequency DCT-coefficients can be seen as locally applying a low-pass filter to an 8x8 or larger image block. Since, for detecting label-bits, the presence or absence of high frequency components in an area can be measured, the labeling method should exhibit some degree of resistance to line-shifting, cropping etc.

In subsections 5.1 and 5.2 a basic implementation of the method is tested to check the feasibility of the labeling technique. In subsection 5.3 this method is optimized and automated. In subsection 5.4, experiments are performed using the optimized method.

### 5.1 Basic Implementation of the DCT Labeling Method Based on Removing DCT Coefficients

The algorithm described in this subsection removes all DCT-coefficients below a certain line  $l$  (see Figure 3) of a number of 8x8 DCT-blocks to embed a label-bit “1”. If a label-bit “0” must be embedded this number of 8x8 DCT-blocks is not affected. The bitstring, which is embedded using this method consists of the sequence:  $L_0H_0L_1H_1L_2H_2L_3H_3...L_\tau H_\tau$ , where  $L$  stands for a label-bit and  $H$  stands for a headerbit. The headerstring  $H_0H_1H_2H_3...H_\tau$ , a fixed pseudo-random pattern, is known by the device which adds the label and also by the device which extracts the label. To extract the labelstring  $L_0L_1L_2L_3...L_\tau$ , the headerstring is extracted first. All parameters needed for the extraction of the labelstring are chosen in such a way that a minimal number of errors is made in the headerstring.



**Figure 4.** Block definitions in an image.

### Labeling procedure:

For the labeling procedure a line number  $l$  (see Figure 3) must be selected manually. The smaller  $l$  is chosen the more visible the label is. If a higher line number is used, the label is less robust.

1. A line number  $l$  (see Figure 3) is manually selected. The label-bit counter  $i$  is set to 0.
2. A block  $B_i$  containing  $n \times n$  8x8 Y-DCT-blocks is selected from the image  $I$  to embed  $L_i H_i$  (see Figure 4).
3. All 8x8 DCT-blocks in  $BU_i$  are selected to embed  $L_i$ .  $BU$  is the upper half of block  $B$ . If  $L_i$  is 1, all coefficients below line  $l$  in these DCT-blocks are set to zero.
4. All 8x8 DCT-blocks in  $BL_i$  are selected to embed  $H_i$ .  $BL$  is the lower half of block  $B$ . If  $H_i$  is 1, all coefficients below line  $l$  in these DCT-blocks are set to zero.
5. The label-bit counter  $i$  is increased by 1. The procedures (2..5) are repeated until all label-bits  $L_i$  are embedded.

### Label extracting procedure:

Two two-dimensional arrays  $EL_{li}$  and  $EH_{li}$  are required for the extraction procedure. A threshold  $T$  is used to distinguish labeled blocks from not-affected blocks. By using a variable threshold instead of a fixed threshold, the label becomes more resistant to line-shifting adding noise and other distortions.

1. The label-bit counter  $i$  is set to 0.
2. A block  $B_i$  containing  $n \times n$  8x8 Y-DCT-blocks is selected from the image  $I$  to extract  $L_i H_i$  (see Figure 4).
3. All 8x8 DCT blocks in  $BU_i$  are selected to extract  $L_i$ .  $BU$  is the upper half of block  $B$ . The line number  $l$  is set to 4 and the squared sum  $EL_{li}$  of all DCT-coefficients below line  $l$  (see Figure 3) in all these selected 8x8 DCT-blocks is calculated. This procedure (3) is repeated for  $l = 5, 6, \dots, 12$ .
4. All 8x8 DCT-blocks in  $BL_i$  are selected to extract  $H_i$ .  $BL$  is the lower half of block  $B$ . The line number  $l$  is set to 4 and the squared sum  $EH_{li}$  of all DCT-coefficients below line  $l$  in all these selected 8x8 DCT-blocks is calculated. This procedure (4) is repeated for  $l = 5, 6, \dots, 12$ .
5. The label-bit counter  $i$  is increased by 1. The procedures (2..5) are repeated until all label-bits  $L_i$  and headerbits  $H_i$  are extracted.
6. A threshold  $T$  (between 0 and 16000) and a line number  $l$  (between 4 and 12) are searched for which the number of bit errors in the headerstring  $H_{0..z}$  is minimal.  $H_i = 0$  if  $EH_{li} > T$ , otherwise  $H_i = 1$  ( $i = 0..z$ ).
7. The label is extracted using the  $T$  and  $l$  calculated in step 6,  $L_i = 0$  if  $EL_{li} > T$ , otherwise  $L_i = 1$  ( $i = 0..z$ ).

## 5.2 Experimental Results using the Basic Implementation of the DCT-Labeling Method

The experiments described here, are not performed on compressed images, but on raw RGB data, which is first converted to the YUV domain. From these Y-values the 8x8 DCT-coefficients are calculated. So, the image test set can be seen as data which is JPEG-compressed using a quality factor  $Q=100$ .

The image test set is labeled using the method described in the previous subsection. Each label-bit is embedded in a block of  $32 \times 32$  pixels ( $n = 4$ ). The label size is dependent on the size of the image (see subsection 4.2). In Figure 5 the percentages bit errors are represented, after compressing the images using the JPEG algorithm, with the quality factor set to different values (100..40). The manually selected line numbers  $l$  are listed between brackets in the legend (lower line numbers  $l$  for smooth images, higher line numbers  $l$  for textured images).

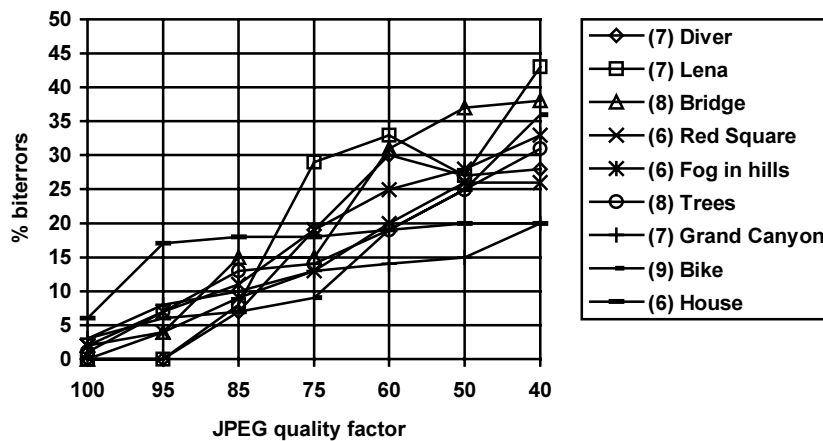


Figure 5. % bit errors after JPEG compression using non-optimized DCT-method.

The labels in “Red Square”, “Fog in hills” and “House” are slightly visible, because of the relative low parameter  $l$ . In some areas too much detail is lost or mosquito noise appears around sharp edges. However, if a higher line number  $l$  is used, the label is not sufficiently resistant to JPEG compression. If smaller blocks (e.g.  $n=2$ ) are used, the label is less visible, because small smooth (labeled) areas and more textured (unlabeled) areas alternate more often. This is less disturbing than big smooth areas. But if small blocks are used, the label is less resistant to JPEG-compression.

As can be seen in Figure 6, the label is slightly resistant to line-shifting or cropping. However, the label is not resistant to JPEG-compression using a lower  $Q$  followed by shifting. If the labeled ( $l = 7$ ,  $n = 4$ ) “Lena” image is for instance compressed with a quality factor  $Q=85$  (compression rate 16:1), shifted one pixel along the x-axis and finally compressed again with a quality factor  $Q=85$ , 43% bit errors are made in the extracted label.

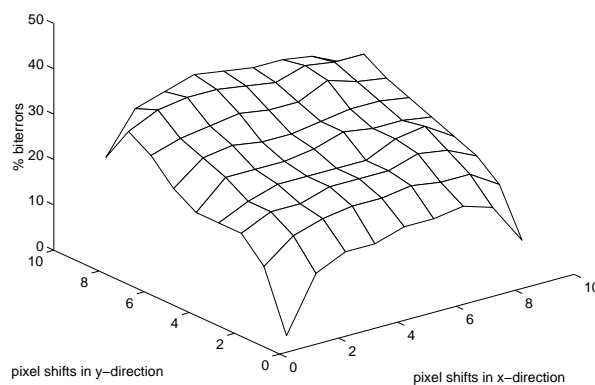


Figure 6. % bit errors after shifting the “Lena” image ( $l = 7$ ,  $n = 4$ ).

### 5.3 Optimized DCT Labeling Method Based on Removing DCT Coefficients

From the experiments performed in the previous subsection we can conclude that it is possible to embed a label by removing high frequencies DCT-coefficients. However, the method described in subsection 5.1 has three drawbacks. In the first place, we need a parameter, which determines beforehand up to which level the label is resistant to JPEG compression (e.g. the JPEG quality factor  $Q$  can be used for this purpose). Further, the label embedding-level  $l$  must be determined automatically instead of manually. Finally, an optimal embedding-level  $l$  dependent on the image content must be determined for each label-bit and image block, since a fixed label embedding-level for the complete image causes artefacts in some cases.

In order to find back the embedding-level for each label-bit a slightly different approach is used than in subsection 5.2 without using a headerstring. To embed label-bit “1” a block  $B$  containing  $n \times n$   $8 \times 8$  DCT-blocks is selected from the image (see Figure 4). All DCT-coefficients below an automatically selected line  $l$  (see Figure 3) in the  $8 \times 8$  DCT-blocks of  $BU$  are discarded, where  $BU$  is the upper half of block  $B$ . To embed label-bit “0” the same procedure is applied to  $BL$ , where  $BL$  is the lower half of block  $B$ . The label can be extracted by comparing the squared sum of all DCT-coefficients below line  $l$  in all  $8 \times 8$  DCT-blocks of  $BU$  and  $BL$ .

For smooth areas a low embedding-level  $l$  is required to find DCT-coefficients which are non-zero. For more textured areas a higher  $l$  is needed, otherwise too many coefficients are set to zero and the label will become visible. Although, using an adaptive  $l$  is a better approach than using a fixed  $l$ , the following problem can occur:

In a block with for instance almost no texture and one sharp edge, a quite low  $l$  will be chosen and the sharp edge will be distorted. Taking a larger block  $B$  to embed a single label-bit will increase the probability that enough high frequency DCT-coefficients can be found and that a higher embedding-level  $l$  is chosen. However, using larger blocks will decrease the label size. In a block with much texture, a quite high  $l$  will be chosen and only a few high frequency coefficients will be set to zero, which are very sensitive to JPEG compression (quantizing).

This problem can easily be solved by pseudo-randomly shuffling all  $8 \times 8$  DCT-blocks of image  $I$  before labeling (Figure 7b). If  $SSD$  is the squared sum of all DCT-coefficients below a certain line  $l$  in the  $8 \times 8$  DCT-blocks of  $BU$  or  $BL$  and the image is not shuffled,  $SSD$  ranges from 0 for smooth blocks  $BU$  or  $BL$  to a certain maximum for textured blocks. If all  $8 \times 8$  DCT-blocks of the image are first shuffled, smooth  $8 \times 8$  blocks and textured  $8 \times 8$  blocks will alternate in block  $BU$  or  $BL$ .  $SSD$  will now have a smaller range, the expected values vary less around the average. So, the chance that a very low or a very high embedding-level  $l$  will be used is smaller if the blocks are shuffled first. An extra advantage of shuffling is that each label-bit is scattered over the image, which makes it more difficult for a hacker to locate the label-bits.

#### Labeling procedure:

Two parameters  $E$  and  $Q$  determine the robustness of the label.  $E$  determines the amount of coefficients, which are discarded, and  $Q$  determines up to which level the label must be resistant to JPEG compression.

1. All  $8 \times 8$  DCT-Y-blocks of image  $I$  are pseudo-randomly shuffled (see figure 7b). The label-bit counter  $i$  is set to 0.
2. Block  $B_i$  containing  $n \times n$   $8 \times 8$  Y-DCT-blocks is selected from the image  $I$  to embed label-bit  $L_i$ . This block is split into blocks  $BU$  and  $BL$  (see Figure 4).
3. To estimate the line number  $l$  for this block, the values of blocks  $BU$  and  $BL$  are temporarily stored in  $TempBU$  and  $TempBL$ . The quality of  $TempBU$  and  $TempBL$  is reduced by re-quantizing the DCT-coefficients of the  $8 \times 8$  DCT-blocks with a certain quality factor  $Q$ . The line number  $l$  is set to 3.
4. The squared sum  $EU_l$  of all DCT-coefficients below line  $l$  (see Figure 3) in all  $8 \times 8$  DCT-blocks in  $TempBU$  is calculated.  $EL_l$  is calculated in the same way for block  $TempBL$ . This procedure (4) is repeated for  $l = 4, 5, \dots, 15$ .
5.  $l_e$  is set to the highest line number  $l$  (3..15) for which  $(EU_l > E)$  AND  $(EL_l > E)$
6. If  $L_i$  is 1, all coefficients below line  $l$  in all  $8 \times 8$  DCT-blocks of block  $BU$  are set to zero.  
If  $L_i$  is 0, all coefficients below line  $l$  in all  $8 \times 8$  DCT-blocks of block  $BL$  are set to zero.
7. The label-bit counter  $i$  is increased by 1. The procedures (2..7) are repeated until all label-bits  $L_i$  are embedded.
8. All  $8 \times 8$  DCT-Y-blocks are shuffled back to their original locations.

### Label extracting procedure:

For the extraction procedure a fixed threshold  $D$  and a quality factor  $Q$  are required. The threshold  $D$  influences the line number  $l$ , which is used to extract each label-bit. A relative small value must be chosen, for instance the value 10. For the quality factor  $Q$  the same value must be used as in the labeling procedure.

1. All 8x8 DCT-Y-blocks of image  $I$  are pseudo-randomly shuffled (see figure 7b). The label-bit counter  $i$  is set to 0.
2. Block  $B_i$  containing  $n \times n$  8x8 Y-DCT-blocks is selected from the image  $I$  to extract label-bit  $L_i$ . This block is split into blocks  $BU$  and  $BL$  (see Figure 4).
3. The quality of  $BU$  and  $BL$  is reduced by re-quantizing the DCT-coefficients of the 8x8 DCT-blocks with a certain quality factor  $Q$ . Line number  $l$  is set to 3.
4. The squared sum  $EU_l$  of all DCT-coefficients below line  $l$  in all DCT-blocks in block  $BU$  is calculated. The squared sum  $EL_l$  is calculated in the same way for block  $BL$ . This procedure (4) is repeated for  $l = 4, 5, \dots, 15$ .
5.  $l_u$  is set to the lowest line number  $l$  (3..15) for which  $(EU_l < D)$ .  
 $l_l$  is set to the lowest line number  $l$  (3..15) for which  $(EL_l < D)$ .
6. If  $(l_u < l_l)$  the label-bit  $L_i$  embedded in the block  $B_i$  is one.  
If  $(l_u = l_l)$  AND  $(EU_{(l_u)} < EL_{(l_u)})$ ,  $L_i$  is also one.  
Otherwise  $L_i$  is zero.
7. The label-bit counter  $i$  is increased by 1. The procedures (2..7) are applied to all blocks  $B_i$  until all bits of the label are extracted.

### 5.4 Experimental Results using the Optimized DCT-Labeling Method

The experiments described in this subsection, are also not performed on compressed images, but on raw RGB data, which is first converted to the YUV domain. From these Y-values the 8x8 DCT-coefficients are calculated. So, the image test set can be seen as data which is JPEG-compressed using a quality factor  $Q=100$ .

The image test set is labeled using the optimized method described in the previous subsection. Each label-bit is embedded in a block of 32 x 32 pixels ( $n = 4$ ), the threshold  $E$  is set to 40, the threshold  $D$  is set to 10 and a quality factor  $Q$  of 75 is used. Using these settings the label is perceptually invisible in all nine images.

To get an idea of how the label looks like, the original unlabeled “Lena” image (Figure 7a) is subtracted from the labeled image (7d) and this difference signal is multiplied by 30 (7c). In the image, which is shown in Figure 7c, a white dot stands for +128, a black dot for -128 and the gray background for 0.

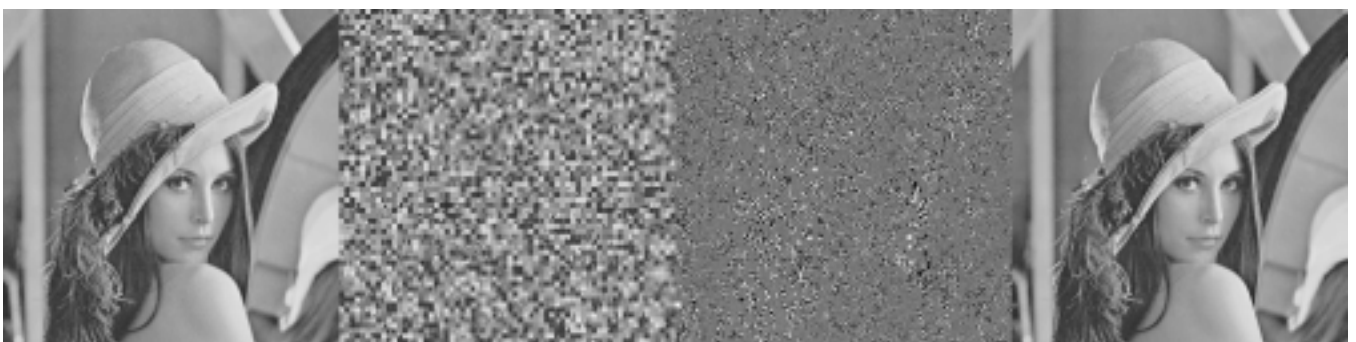
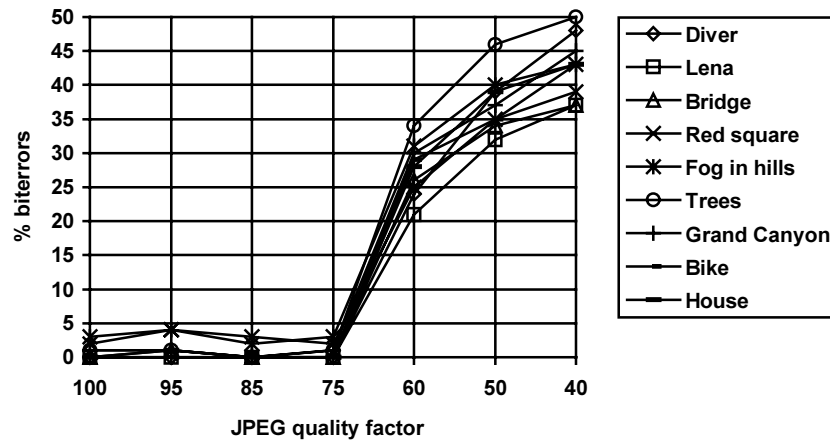


Figure 7. (a) Unlabeled (b) Unlabeled shuffled image (c) Difference x 30 (d) Labeled image

In Figure 8 the percentages bit errors are represented, after compressing the images using the JPEG algorithm, with the quality factor set to different values (100..40). In this figure we see that the percentage errors is below 5% for JPEG quality factors between 100 and 75, after this breakpoint  $Q=75$  the percentage errors increases rapidly. This is quite obvious because we embedded the label using parameter  $Q=75$ .



**Figure 8.** % bit errors after JPEG compression using optimized DCT-method.

To investigate the resistance to line-shifting and cropping the following experiment was carried out. The images are labeled using the settings described above. The images are JPEG compressed using a quality factor  $Q=85$ . After that, the images are shifted over  $(x,y)$  pixels. Finally, the images are compressed again using the JPEG algorithm ( $Q=85$ ). The results of this experiment are listed in Table 1. From these results it appears that the labeling method is slightly resistant to line-shifting or cropping.

**Table 1.** % bit errors after compressing, decompressing, shifting and again compressing the images.

Image	shift $(x,y)=(0,0)$	shift $(x,y)=(1,0)$	shift $(x,y)=(0,1)$	shift $(x,y)=(1,1)$
Diver	0	3	6	10
Lena	0	14	17	25
Bridge	0	13	23	32
Red Square	3	16	17	27
Fog in hills	2	17	15	26
Trees	0	4	5	14
Grand Canyon	0	13	12	28
Bike	0	11	13	20
House	0	18	23	33

## 6. Discussion

In this paper, we proposed two new labeling methods for copy protection of images. The first method adds the label in the spatial domain, the second method in the DCT-domain. Both methods automatically derive the parameters for adding and extracting the label from the image. The labels are resistant to JPEG/MPEG compression up to a level, which can be determined beforehand.

Using both methods a perceptually invisible label of a few hundred bits was embedded in a set of true color images. The label added by the spatial method is very robust against JPEG compression. However, this method is not suitable for real-time applications. Although the DCT-based method is slightly less resistant to JPEG compression, it is very suitable for real-time labeling.

The DCT-based method can be applied directly to compressed JPEG or MPEG data. No re-encoding is required and the size of the labeled compressed data is always smaller than the size of the original compressed data. A label added using this method also exhibits some degree of resistance to line-shifting or cropping. If the label size is reduced by using an error correcting code (e.g. repetition code), the label can be made more resistant to line-shifting and cropping.

The spatial labeling method and the DCT-labeling method hardly influence each other. Therefore, both methods can be applied to an image to double the label size. If an image is first labeled using the DCT-labeling method and after that, labeled again using the spatial labeling method, the labels are not visible. In this case, it is still possible to extract both labels with acceptable bit error percentages, even after JPEG compression (Q=75).

## References

1. Digital Audio Interface, International Standard IEC 958
2. W. Bender, D. Gruhl, N. Morimoto : "Techniques for Data Hiding", Proceedings of the SPIE, 2420:40, San Jose CA, USA, February 1995
3. I. Pitas, T. Kaskalis : "Signature Casting on Digital Images", Proceedings IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, June, Greece, 1995
4. J.R. Smith, B.O. Comiskey, "Modulation and Information Hiding in Images", Preproceedings of Information Hiding, an Isaac Newton Institute Workshop, University of Cambridge, UK, May 1996
5. G.C. Langelaar, J.C.A. van der Lube, J. Biemond : "Copy Protection for Multimedia Data based on Labeling Techniques", 17th Symposium on Information Theory in the Benelux, Enschede, The Netherlands, May 1996
6. J. Zhao, E. Koch : "Embedding Robust Labels into Images for Copyright Protection", Proceedings of the International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Vienna, Austria, August 1995
7. E. Koch, J. Zhao : "Towards Robust and Hidden Image Copyright Labeling", Proceedings IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, June, 1995
8. I.J. Cox, J. Kilian, T. Leighton, T. Shamoon : "Secure Spread Spectrum Watermarking for Multimedia", NEC Research Institute, Technical Report 95-10
9. CD-ROM: "World of Photo", US Dreams Inc, USD014, 1995
10. W.B. Pennebaker, J.L. Mitchell : "The JPEG Still Image Data Compression Standard", Van Nostrand Reinhold, New York, 1993