In: Proc. of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Vienna, August 1995

EMBEDDING ROBUST LABELS INTO IMAGES FOR COPYRIGHT PROTECTION

Jian Zhao & Eckhard Koch
Fraunhofer Institute for Computer Graphics
Wilhelminenstr. 7, 64283 Darmstadt, Germany
Email: {zhao, ekoch}@igd.fhg.de

Abstract

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for identifying image copyright holder and original distributor in digital networked environment. The embedded label is undetectable, unremovable and unalterable. Furthermore it can survive processing which does not seriously reduce the quality of the image, such as lossy image compression, low pass filtering and image format conversions.

1 Introduction

The wide use of digitally formatted audio, video and printed information in network environment has been slowed down by the lack of adequate protection on them. Developers and publishers hesitate to distribute their sensitive or valuable materials because of the easiness of illicit copying and dissemination [3],[6],[7].

Compared to ordinary paper form information, digitized multimedia information (image, text, audio, video) provides many advantages, such as easy and inexpensive duplication and re-use, less expensive and more flexible transmission either electronically (e.g. through the Internet) or physically (e.g. as CD-ROM). Furthermore, transferring such information electronically through network is faster and needs less efforts than physical paper copying, distribution and update. However, these advantages also significantly increase the problems associated with enforcing *copyright* on the electronic information.

Basically, in order to protect distributed electronic multimedia information, we need two types of protections. First, the multimedia data must contain a label or code, which identifies it uniquely as property of the copyright holder. Second, the multimedia data should be marked in a manner which allows its distribution to be tracked. This does not limit the number of copies allowed (vs. copy protection), but provides a mean to check the original distributor. In order to prevent any copyright forgery, misuse and violation, the copyright label must be unremovable and unalterable, and furthermore survive processing which does not seriously reduce the quality of the data. This requires that first the label must be secretly stored in a multimedia data, i.e. the locations for embedding this label are secret, second the label must be robust even if the labeled multimedia data has been processed incidentally or intentionally.

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for copyright protection in open networked environment. The label embedded in the image can be assigned or generated in a way that it is able to identify the copyright holder and the original purchaser (distributor).

Steganographic method is a technique embedding additional information into a data by modifying the original data without affecting the quality of the data. Many steganographic methods have been proposed to aim at storing additional information to identify or label formatted electronic documents [1], images, video [8], and audio data. However, they are far away from the requirements in protecting multimedia information in a networked environment, because although some of them provide secret locations for label embedding, none of them is able to prevent attacks on the embedded information by simple image processing, i.e. they do not adequately address the possibilities of using data compression, low pass filtering and/or simply changing the file format to remove an embedded code.

The discussion begins with a general framework for copyright label embedding. Then two specific methods are developed: one is based on the JPEG compression model for embedding labels in gray-scaled and color images, and the other is based on the black/white rate for binary images. Finally, these methods are tested experimentally and the future work is discussed.

2 Robust Label Embedding Framework

The system developed along the methods presented in this paper is called 'SysCoP' (System for Copyright Protection). It consists of a set of methods to embed robust labels into different types of images. Currently, the system supports gray-scaled, color, and binary images. These methods share an algorithm framework for both label writing and reading described below.

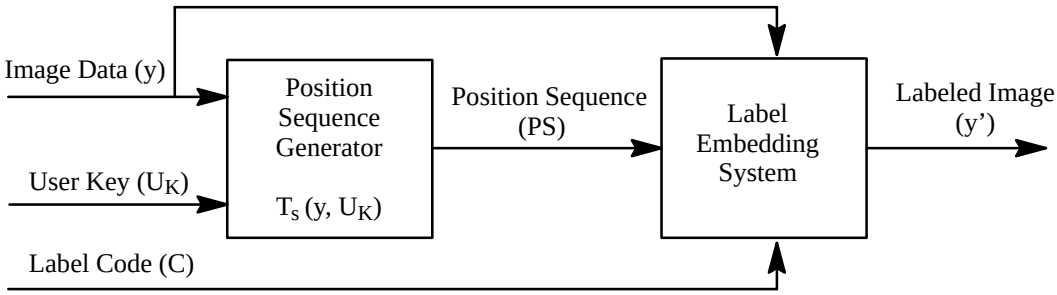

Figure 1. Write label

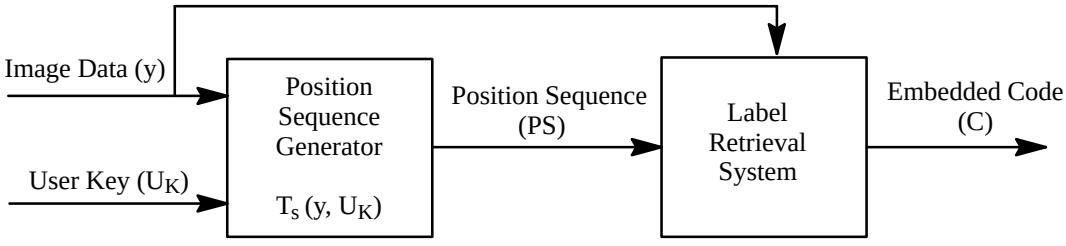

Figure 2. Read label

The framework, as shown in Figure 1 for label writing and Figure 2 for label reading, is composed of two steps. The first step generates a pseudo random position sequence for selecting blocks where the code is embedded. This step is denoted as a function $T_s(y, U_K)$ where y is the image data to be labeled, and U_K is the user-supplied secret key. The second step simply embeds or retrieves the code into or from the blocks specified in the position sequence. The methods for embedding or reading code depend on types of images, and will be described individually in the next section.

The function $T_s(y, U_K)$ firstly extracts some features from the image data and then use them together with the user secret key as the seeds for position sequence generation [4]. Ideally, the features of the image data used here must meet the following requirements:

- they must be robust against simple image processing that does not affect the visual quality of the image, and
- they must be image-dependent, i.e. the image can be identified, uniquely in an ideal case, by these features extracted from the image data.

However, to achieve the contradictory requirements above, in fact, is a very hard work. Much research has been done in related fields for other purposes, e.g. content-based image retrieval, image segment and pattern recognition, which can be found in many literature (e.g. [9]). Currently, we only use the width and height of an image for the position generation in the function $T_s(y, U_K)$.

Before describing the framework, we introduce some terminology. A *block* of an image consists of 8x8 pixels. In this framework, it is either a contiguous or a distributed block. A *contiguous block* is a 8x8 square in an image component. A *distributed block* is a collection of 8x8 pixels each of which is selected randomly from the whole image space. The purpose of distributed block is to prevent or discourage attackers from detecting embedding locations by comparing different labeled images. A block is 'invalid' for code embedding if too big modifications to the block data are needed in order to embed a bit into this block. The criteria of validation of the block depends on the specific label-embedding methods to be described in the next section.

Let *C* be the embedded code, and represented as a binary bit stream $\{c_0, c_1, ..., c_n\}$. Let i be the index of current bit in this stream. Let $\mathfrak{B}$ be the block set in which each block has been randomly selected. Initialize i to 0 and $\mathfrak{B}$ to {}. The framework for writing and reading robust labels is described below in Algorithm 1(a)-(b).

Algorithm 1(a): Framework (write).

(1) If $i \geq n$, return.

(2) Randomly select a block *b*, using the position sequence generation function $T_s(U_K, y)$ in Figure 1.

(3) If *b* exists already in $\mathfrak{B}$, go to (2), otherwise add *b* to $\mathfrak{B}$.

(4) Call ***check_write(b, c_i)*** to check wether *b* is a valid block: if this function returns False (i.e. the block *b* is an invalid block), go to (2).

(5) Call ***write(b,c_i)*** to embed a bit c_i to the block *b*.

(6) Increment i, go to (1).

Algorithm 1(b): Framework (read).

(1) If i $\geq$ n, return.

(2) Randomly select a distributed or a contiguous 8x8 block *b*, using the position sequence generation function $T_s(U_K, y)$ in Figure 2.

(3) If *b* exists already in $\mathfrak{B}$, then go to (2), otherwise add *b* to $\mathfrak{B}$.

(4) Call ***check_read(b,c$_i$)*** to check wether *b* is a valid block: if this function returns False (i.e. the block *b* is an invalid block), go to (2).

(5) Call ***read(b)*** to retrieve a bit from the block *b*.

(6) Increment i, and go to (1).

3 Robust Label Embedding Methods

3.1 JPEG-Based Label Embedding for Gray-Scaled and Color Images

In this subsection, we first introduce briefly the JPEG compression model, then describe the principle of the embedding methods based on the JPEG compression model. Finally, the algorithms for embedding labels into gray-scaled and color images are developed.

Suppose the source image composes three components: one luminance (Y) and two chrominance (I and Q). That is, each pixel in the image can be represented with a triple of 8-bit values (Y,I,Q). Each component is broken up into contiguous blocks. The JPEG compression consists of six steps: normalization, DCT transformation, quantization, zigzag scan, run-length encoding and Huffman coding steps. Since our method is applied after the quantization step, we only describe briefly the first three steps of the JPEG model. The detailed description of the JPEG model is available elsewhere [11].

The normalization step brings all image values into a range, e.g. between –128 and 127 for a 24-bit image. The DCT step applies the discrete cosine transform (DCT) to each 8x8 block, producing a new 8x8 block [10]. If we call the new block Y(k,l), with k,l $\in$ 0..7, the equation of the DCT is:

$$Y(k, l) = \frac{1}{4} \sum_i \sum_j C(i, k)C(j, l)y[i, j]$$

where

$$C(i, k) = A(k)\frac{\cos(2i + 1)k\pi}{16} \qquad A(k) = \frac{1}{\sqrt{2}} \text{ for } k = 0, \; A(k) = 1 \text{ for } k \neq 0 \qquad (1)$$

Each element of the new block is further quantized:

$$Y_Q[k, l] = \text{Round}(\frac{Y[k, l]}{q[k, l]}) \qquad (2)$$

Equation (2) represents the entire lossy modelling process of the JPEG compression. The choice of the quantization table (q[k,l]) determines both the amount of compression and the quality of the decompressed image. The JPEG standard includes recommended luminance and chrominance quantization tables resulting from human factors studies. To obtain different compression quality, we typically use a *quality factor* to scale the values of these default quantization tables.

In the JPEG decompression process, each element of $Y_Q(k,l)$ is multiplied by q(k,l) to recover an approximation of Y(k,l). Finally, the image block y(i,j) can be recovered by performing an inverse 2-D DCT (IDCT):

$$y(i,j) = \frac{1}{4}\sum_k \sum_l C(i,k)C(j,l)Y[k,l] \qquad (3)$$

The basic principle of the JPEG-based embedding method is that quantized elements have a moderate variance level in the middle frequency coefficient ranges, where scattered changes in the image data should not be noticeably visible. The specific frequencies being used to embed the code will be 'hopped' in this range to increase the robustness of the signal and making it more difficult to find [5],[2]. A label bit is embedded through holding specific relationship among three quantized elements of a block. The relationships among them compose 8 patterns (combinations) which are divided into three groups: two of them are used to represent '1' or '0' for embedded codes (valid patterns), and the other represents *invalid patterns*. If too big modifications are needed to hold a desired valid pattern representing a bit, this block is invalid. In this case, the relationships among the three elements of the selected location set are modified to any of the invalid patterns to 'tell' the label-retrieval process that this block is invalid. The criterion for invalid blocks is specified by a parameter MD, i.e. the maximum difference between any two elements of a selected location set in order to reach the desired valid pattern.

Set No.	(k_1,l_1)	(k_2,l_2)	(k_3,l_3)
1	2(0,2)	9(1,1)	10(1,2)
2	9(1,1)	2(0,2)	10(1,2)
3	3(0,3)	10(1,2)	11(1,3)
4	10(1,2)	3(0,3)	11(1,3)
5	9(1,1)	2(0,2)	10(1,2)
6	2(0,2)	9(1,1)	10(1,2)
7	9(1,1)	16(2,0)	2(0,2)
8	16(2,0)	9(1,1)	2(0,2)
9	2(0,2)	9(1,1)	16(2,0)
10	9(1,1)	2(0,2)	16(2,0)
11	10(1,2)	17(2,1)	3(0,3)
12	17(2,1)	10(1,2)	3(0,3)
13	10(1,2)	3(0,3)	17(2,1)
14	3(0,3)	10(1,2)	17(2,1)
15	9(1,1)	16(2,0)	17(2,1)
16	16(2,0)	9(1,1)	17(2,1)
17	10(1,2)	17(2,1)	18(2,2)
18	17(2,1)	10(1,2)	18(2,2)

Table 1. Possible location sets

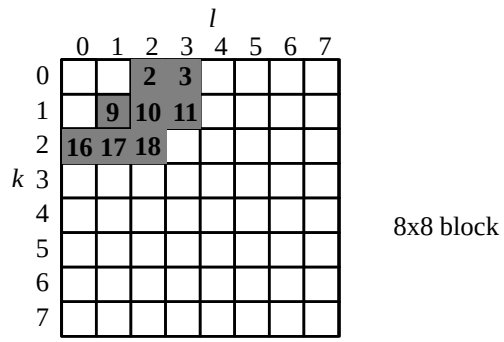

Figure 3. Possible locations for embedding code in a block

(k_1,l_1)	(k_2,l_2)	(k_3,l_3)	
H	M	L	patterns for '1'
M	H	L	
H	H	L	
M	L	H	patterns for '0'
L	M	H	
L	L	H	
H	L	M	invalid patterns
L	H	M	
M	M	M	

Table 2. '1', '0' and invalid patterns (H: High, M: Middle, L: Low)

Our statistic results of the possible locations holding the specific frequencies are illustrated in Figure 3 as shadowed areas within a 8x8 block. In Table 1, we give our statistic results of the best location sets

combined from these possible elements. The algorithms to write and read a label into and from an color or gray-scaled image are described in Algorithm 2(a)-(d).

Two parameters are provided for adjusting the robustness vs. modification visibility in an embedding process. The first one is the distance (D) between selected quantized frequency coefficients for representing an embedded bit. The default value of this distance is 1. The greater distance produces stronger robustness, but also may cause more serious modification visibilities. The second parameter is the quantization factor (Q) used to quantize the values selected for embedding code. The greater quantization factor results in less modifications to image data but weaker robustness against lossy JPEG compression. The default value of this quantization factor is 75%.

Algorithm 2(a): check_write(b,c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) When $c_i=1$, if $MIN(|Y_Q(k_1,l_1)|, |Y_Q(k_2,l_2|) + MD < |Y_Q(k_3,l_3)|$ where $|Y_Q(k_u,l_u)|$ is the absolute value of $Y_Q(k_u,l_u)$ with $u \in 1..3$, MIN is an operation that returns the minimum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize and inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(4) When $c_i=0$, if $MAX(|Y_Q(k_1,l_1)|, |Y_Q(k_2,l_2|) > |Y_Q(k_3,l_3)| + MD$ where MAX is an operation that returns the maximum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize, inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(5) For other cases, return True.

Algorithm 2(b): check_read(b, c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) If $|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2|$ and $|Y_Q(k_3,l_3)|$ form any of the invalid patterns as illustrated in Table 2, return False, otherwise return True.

Algorithm 2(c): write(b, c_i)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$,$Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) When c_i=1, modify the $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) > Y_Q(k_3,l_3) + D$, and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$

(2) When c_i=0, modify the $Q(k_1,l_1)$, $Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$

(3) $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ are de-quantized, inversely transformed (IDCT), and written back to the block *b*.

Algorithm 2(d): read(b)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$,$Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) If $Y_Q(k_1,l_1) > Y_Q(k_2,l_2) + D$ and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$, return 1.

(2) If $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$, returns 0.

(3) In other cases, the embedded bit in this block *b* has been damaged.

3.2 Black/White Rate-Based Label Embedding for Binary Images

The value of each pixel in a binary image is either '1' or '0'. This determines that, in general, there is no 'noise' space which can be used for embedding additional information. To do it, we must find appropriate image areas where modifications for embedding labels do not affect seriously the quality of the original image. Obviously, these areas are varied with individual images or at least with types of images.

The proposed method for binary images is based on the ratio of '1' and '0' in a selected block. Suppose '1' represent black bit and '0' represent white bit in the source binary image. Let $P_1(b)$ be the rate (percentage) of blacks in the selected block *b*:

$$P_1(b) = \frac{N_1(b)}{64} \quad \text{where } N_1(b) \text{ is the number of } '1' \text{ in the block } b.$$

Since the sum of percentages of blacks and whites in a block is 100%, the rate (percentage) of whites in the block *b* is $P_0(b) = 100 - P_1(b)$. A bit is embedded into a block *b* in the following way: a '1' is embedded into the block *b* if $P_1(b)$ is greater than a given threshold, and a '0' is embedded into the block *b* if $P_1(b)$ is less than another given threshold. A sequence of contiguous or distributed blocks is modified by switching whites to blacks or vice versa until such thresholds are reached.

We have classified two categories of binary images on which the generic method described above can be applied. These binary images are identified by distribution feature of blacks and whites. The first type of binary images is dithered image in which the black and white are well interlaced. The second type of binary images is black/white sharply contrasted images in which there exist clear boundaries between black and white areas.

Two modification strategies are adopted for these two types of binary images, respectively. For dithered binary images, modifications are well-distributed throughout the whole block: the bit that has most neighbors with the same value (either black or white) is reversed. For sharply contrasted binary images, modifications are carried out at the boundary of black and white pixels: the bit that has most neighbors with the opposite value is reversed. At the borders of the contiguous block, the neighbor bits in the neighbor blocks are also taken into account in both approaches. Two examples of both modification strategies are illustrated in Figure 4 and 5, respectively.

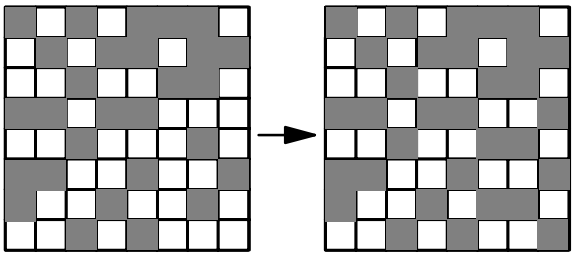

Figure 4. Well-distributed modifications

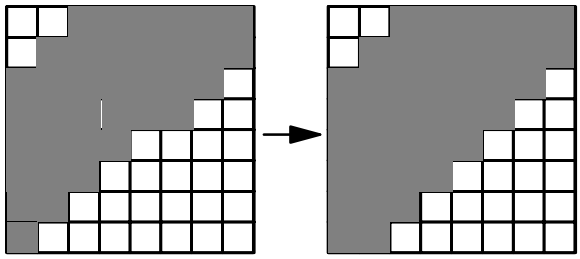

Figure 5. Modifications at black and white boundary

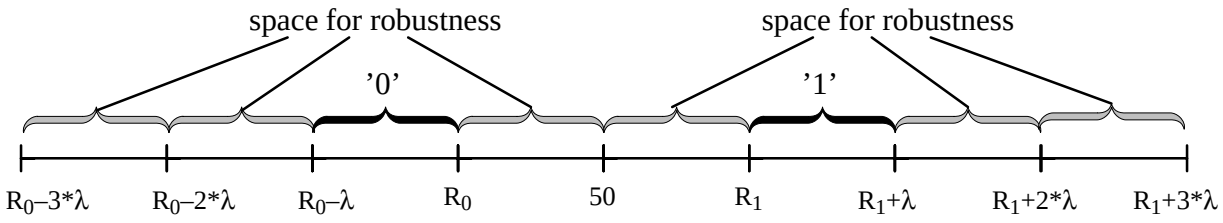

Figure 6. Achieve robustness in the black/white rate-based embedding method

Let R_1 be the threshold rate for '1'. Thus, the threshold rate for '0' is $R_0 = (100\%–R_1)$. Let λ be the robustness degree against image processing of labeled images. It represents the number of bits that can be altered after image processing without damage of embedded bits. For example, when λ is 5%, alternation (i.e. reversion from '1' to '0' or vice versa) of less than 4 bits in a block does not damage the embedded code. Our experiments have shown that the following values of them are the reasonable choices both in robustness of embedding code and the modification visibility:

$R_1 = 55$, $R_0 = 45$, and $\lambda = 5$.

The algorithms to write and read a label into and from a binary image are described in Algorithm 3(a)-(d).

Algorithm 3(a): check_write(b, c_i)

(1) If $P_1(b) > R_1 + 3*\lambda$ or $P_1(b) < R_0 - 3*\lambda$, return False.

(2) When c_i=1, if $P_1(b) < R_0$, modify the block *b* such that
$P_1(b) < R_0 - 3*\lambda$,
and then return False.

(3) When c_i=0, if $P_1(b) > R_1$, modify the block *b* such that
$P_1(b) > R_1 + 3*\lambda$,
and then return False.

(4) For other cases, return True.

Algorithm 3(b): check_read(b, c_i)

(1) If $P_1(b) > R_1 + 2*\lambda$ or $P_1(b) < R_0 - 2*\lambda$, return False.

(2) For other cases, return True.

Algorithm 3(c): write(b, c_i)

Assume that a valid block *b* has been pseudo-randomly selected. A bit c_i is embedded into *b* by switching blacks to whites or vise versa using the different modification strategies described above in order to reach a specified threshold rate.

(1) When c_i=1, modify the block *b* such that:
$P_1(b) \geq R_1$ and $P_1(b) \leq R_1 + \lambda$.

(2) When c_i=0, modify the block *b* such that:
$P_1(b) \leq R_0$ and $P_1(b) \geq R_0 - \lambda$.

(3) write the block *b* back to the image.

Algorithm 3(d): read(b)

Assume that a valid block *b* has been pseudo-randomly selected.

(1) If $P_1(b) > 50$, return 1.

(2) If $P_1(b) < 50$, return 0.

(3) For other cases, the embedded bit in the block *b* has been damaged.

4 Conclusion

The 'SysCoP' has been implemented on UNIX platform, and provides a graphical interface, a set of UNIX commands and an API (Application Programming Interface). It currently supports JPEG, PPM, GIF, and TIFF image formats. Experiments have been carried out to demonstrate the robustness of our methods against image processing. For the gray-scaled and color images using the JPEG-based embedding method, three images were labeled, and then processed by JPEG compression, format conversions and color reduction. In general, the results are quite satisfactory and meet the basic requirements for embedding codes as copyright labels. Due to the space limitation of the paper, concrete results are omitted. For the binary images, a dithered TIFF binary image and a sharply contrasted TIFF binary image were used in our tests. They are labeled first with R_1= 55% and the robustness degree (λ) 5%. The labeled TIFF images are then smoothed and converted to PBM. The embedded codes were not damaged in both labeled images after smoothing and conversions.

Our methods are still weak against physical damages (e.g. cut a pixel line, grab an area, etc.). Currently, we address this problem by allowing the user to specify 'valuable or sensitive' areas of an

image into which labels are (repeatedly) embedded. Thus, cutting a part which is not in these areas does not damage embedded labels.

The methods described in this paper for embedding robust copyright labels for images have been extended to support MPEG-1. Two additional attacks in embedding copyright labels into MPEG-1 videos have been identified: removal of frames and re-compression with different patterns. To be resistant against them, the copyright label is repeatedly embedded into each frame. Thus we ensure that the label can be retrieved from each I-frame regardless of re-compression with different patterns. Furthermore, we are developing new labeling methods for other digital media, i.e. structured electronic documents (e.g. PostScript, SGML documents) and audio data. In addition, a WWW (World Wide Web) image copyright labeling server incorporating the methods described in this paper is being developed.

Acknowledge We are grateful to Scott Burgett from GMI, USA, who initiated and completed the JPEG-based embedding method of 'SysCoP' during his visiting stay at the Fraunhofer-IGD in Darmstadt. We also want to thank Martin Claviez and Joachim Krumb for helping us in implementing the 'SysCoP' system.

References

[1] J. BRASSIL, S. LOW, N. MAXEMCHUK, L. O'GORMAN. Electronic Marking and Identification Techniques to Discourage Document Copying. AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[2] S. BURGETT, E. KOCH, J. ZHAO. A Novel Method for Copyright Labeling Digitized Image Data. Technical Report of Fraunhofer Institute for Computer Graphics, Darmstadt, August, 1994. (Also submitted to IEEE Trans. on Communication, September, 1994).

[3] A.K. CHOUDHURY, N.F. MAXEMCHUK, S. PAUL, H.G. SCHULZRINNE. Copyright Protection for Electronic Publishing over Computer Networks. AT&T Bell Laboratories, June 1994.

[4] W. DIFFIE and M. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644-654, 1976.

[5] R. C. DIXON. *Spread Spectrum Systems*. 2nd ed., Wiley, New York, NY, 1984.

[6] B. KAHIN. The strategic environment for protecting multimedia. *IMA Intellectual Property Project Proceedings*, vol. 1, no.1, 1994. pp.1-8.

[7] E. KOCH, J. RINDFREY, J. ZHAO. Copyright Protection for Multimedia Data. *Proceedings of the International Conference on Digital Media and Electronic Publishing* (6-8 December 1994, Leeds, UK).

[8] K. MANTUSI and K. TANAKA. Video-Steganography: How to secretly embed a signature in a picture. *IMA Intellectual Property Project Proceedings*, vol. 1, no. 1, 1994.

[9] W. NIBLACK, R. BARBER, W. EQUITZ, M. FLICKNER, E. GLASMAN, D. PETKOVIC, P. YANKER, C. FALOUTOS, G. TAUBIN. The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. *SPIE* vol. 1908, 1993, pp.173-187.

[10] K.R. RAO and P. YIP. *Discrete Cosine Transform: Algorithms Advantages, Applications*. Academic Press. 1990.

[11] G.K. WALLACE. The JPEG still picture compression standard. *Communications of the ACM*, vol. 34, no. 4, April 1991. pp.30-40.