INFORMATICA, 2003, Vol. 14, No. 4, 431–444

A Watermarking Scheme Based on Principal Component Analysis Technique

Chin-Chen CHANG, Chi-Shiang CHAN
Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan 621, R.O.C.
e-mail: {ccc, cch}@cs.ccu.edu.tw

Received: January 2003

Abstract. In this paper, we shall propose a new method for the copyright protection of digital images. To embed the watermark, our new method partitions the original image into blocks and uses the **PCA** function to project these blocks onto a linear subspace. There is a watermark table, which is computed from projection points, kept in our new method. When extracting a watermark, our method projects the blocks of the modified image by using the **PCA** function. Both the newly projected points and the watermark table are used to reconstruct the watermark. In our experiments, we have tested our scheme to see how it works on original images modified by JPEG lossy compression, blurring, cropping, rotating and sharpening, and the experimental results show that our method is very robust and indeed workable.

Key words: digital watermark, principal component analysis.

1. Introduction

On the Internet, people can send, transmit, and receive a wide variety of digital data at any time, including texts, images, and videos. The major characteristic of digital data is that they can be modified or forged with ease. In other words, without the lawful owner's permission, anyone can still modify the original image when it is not technically properly protected-which is unfortunately too often the case. Of all the digital data readily available over the Internet, quite a lot may be of great commercial value and protected under the copyright law. However, just because they are valuable data, illegal users can often benefit handsomely from modifying them. In order to protect digital image data online from illegal modifications, various digital watermarking techniques have been developed. These watermarking techniques have some characteristics:

(1) Invisibility: An ideal digital image watermarking technique should make the watermarked image look like the original image.
(2) Security: An ideal digital image watermarking technique should protect watermark from removing by attacker even although the process of watermark extracting has been known.
(3) Blindness: An ideal digital image watermarking technique should be able to recover the watermark image without the appearance of the corresponding original

image. This way, there will be no need for any additional storage space to keep the original image.

(4) Robustness: An ideal digital watermarking technique should be capable of recovering watermark images when the watermarked images have been modified slightly. Possible modifications include basic image processing operations like lossy image compression, blurring, and sharpening.

Now, we talk about digital watermarking techniques. Digital watermarking techniques mainly include two processes: watermark information computing and watermark recovering. In the watermark information computing process, the characteristics of the original image are extracted and the key data is produced from characteristics and the digital watermark. Then, there are two ways to put the key data in place and to retrieve it. First, we can embed it back into the original image, e.g., (Bender *et al.*, 1996; Celik *et al.*, 2002; Chang *et al.*, 2000) and (Hwang *et al.*, 2000). Once a modified digital image is to be tested, we can extract both the characteristics and the key data from the modified digital image. After extracting the key data and characteristics, the watermark image can be recovered. Instead of embedding the key data directly into the original image, the second way is to register the key data with an authentication center, e.g., (Chang and Tsai, 2000; Voyatzis and Pitas, 1999) and (Wolfgang and Delp, 1996). Those who own the copyright of the images can get their key data from the authentication center. The watermark image can be recovered from the key data and the characteristics of the modified images. This way, it is obvious that the size of the key data cannot be too large, for if the size of the key data is too large, it is even more practical to register the original images with the authentication center directly.

In 2000, Chang and Tsai proposed a watermarking technique that sorts codewords by using the **PCA** function to get a "sorted" codebook. After getting the "sorted" codebook, it randomly chooses some blocks and searches for their corresponding indices through the "sorted" codebook. Once those indices are obtained, the method can produce the key data from the indices and the watermark image. However, Chang and Tsai's method has two problems. First, the "sorted" codebook is an essential part of the method, which degrades the efficiency. Second, Chang and Tsai's method needs to search through the "sorted" codebook to find out the most similar codeword. It is very time consuming.

In order to overcome these two drawbacks, in this paper, we shall propose a new method that uses no codebook. In our method, we get indices directly according to the positions of the projection points by using the **PCA** function. Our experimental results show that our method can recover the watermark image completely without losing any data after JPEG lossy compression, blurring, and sharpening. Even after rotating and cropping, our method can still get very high bit accuracy rates.

The rest of this paper is organized as follows. In Section 2, we will briefly review Chang and Tsai's method. After that, we shall present our new method in detail in Section 3 and then present our experimental results in Section 4. Finally, the conclusion will be in Section 5.

2. Review of Chang and Tsai' s Method

In this section, we will review Chang and Tsai's method. We will first go over the principal component analysis (**PCA**) technique (Lee *et al.*, 1976) briefly. After that, we will show how Chang and Tsai have developed their watermarking technique from the **PCA** technique.

2.1. *Principal Component Analysis (PCA)*

In this section, we will go over principal component analysis (**PCA**) (Lee *et al.*, 1976). The main purpose of **PCA** is to reduce the dimensions of multi-dimensional samples; after dimension reduction, we can use those samples to do pattern recognition. First, **PCA** helps find a special vector, and then all the multi-dimensional sample points can be projected onto a linear subspace in accordance with this special vector. This linear subspace can keep the most of original characteristics of the sample points. The more different sample points there are, the longer the distances between the projection points. After sorting the projection points of the sample points, we can quickly locate them.

The major challenge here is to find this special vector that keeps the original characteristics of the sample points. In 1976, Lee *et al.* first introduced a multiple key sorting and searching method based on **PCA**. In their method, the special vector is computed by using the **PCA** technique this way: given m sample points $V_1, V_2, \ldots, V_m$, where each sample point has n-dimensions. The job here is to find a special vector $D = (d_1, d_1, \ldots, d_n)$, where $\sum_{i=1}^{n} d_i^2$=1, such that the projection points, which are projected onto the subspace in accordance with D, of the sample points can keep their largest difference among each other. The procedure of applying **PCA** to the derivation of D is as follows:

Algorithm [**PCA**]:
Input: Sample points $V_1, V_2, \ldots, V_m$ with n-dimensions.
Output: First principal component direction $D = (d_1, d_1, \ldots, d_n)$, where $\sum_{i=1}^{n} d_i^2 = 1$.

Step 1: Normalize the given sample points. Each normalized sample point is represented as $V_i' = (v_{i1}', v_{i2}', \ldots, v_{in}')$.

Step 2: Compute the covariance matrix C according to all the normalized sample points.

Step 3: Find the eigenvalues of C. We assume that those eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$, and their corresponding eigenvectors are $D_1, D_2, \ldots, D_n$.

Step 4: $D_1 = (d_1, d_2, \ldots, d_n)$ is called the first principal component direction. This is the vector that we want.

The first principal component direction D_1 can keep the most characteristics of the sample points. We can project a given normalized sample point $V_i' = (v_{i1}', v_{i2}', \ldots, v_{in}')$ onto a linear subspace by computing $v_{i1}' \times d_1 + v_{i2}' \times d_2 + \cdots + v_{in}' \times d_n$. If V_i' and V_j' are two different normalized sample points with a large difference of value, then the distance between the projection points of V_i' and V_j' is large.

2.2. *Watermarking Techniques with Codebooks*

In 2000, Chang and Tsai proposed a watermarking technique. The method has two essential components: the **PCA** function and a codebook. In their watermark information computing procedure, the method first takes the codewords as sample points and uses the **PCA** function to derive the first principal component direction from those codewords. Once the first principal component direction is obtained, the projection points of the codewords can be calculated. Chang and Tsai then sort the codewords according to their projection points and get a "sorted" codebook. Then, they use a secret key as the seed of a pseudo random number generator (**PRNG**) to generate a sequence of integer pairs (x_i, y_i) in order to draw out the scope of the block, called OB_i, where (x_i, y_i) is at the upper left corner of this block. For each watermark pixel WP_i, the method picks its corresponding OB_i and searches through the "sorted" codebook to find the most similar codeword C_i (whose index is Idx_i). They check watermark pixel WP_i: if WP_i is a white pixel, they record Idx_i into wt_i (ith element in the watermark table), which contains the so-called key data. Otherwise, if WP_i is a black pixel, they record $(Idx_i + \lfloor n/2 \rfloor) \bmod n$ into wt_i, where n is the total number of pixels in the watermark image. When every WP_i has gone through the procedure, they can get the complete watermark table.

In the watermark image recovering procedure, the method uses a secret key as the seed of **PRNG** to generate a sequence of integer pairs (x_i, y_i). It then uses the same procedure to draw out the scope of the block OB'_i and searches the "sorted" codebook to find the most similar codeword, whose index is Idx'_i. When wt_i is retrieved from the watermark table (***WT***), if the value $|wt_i - Idx'_i|$ is smaller than the preset threshold value, then WP_i is a white pixel; otherwise, WP_i is a black pixel. When every Idx'_i and wt_i have gone through this procedure, they can recover the watermark image.

The main idea of their method is that the **PCA** function can be used to find the first principal component direction and sort the codewords for further use. The first principal component direction can keep the most characteristics of the sample points. This means that if the codewords are modified slightly, they can still recover their corresponding watermark pixels according to the indices of the modified codewords. However, according to our observation, Chang and Tsai's method has two drawbacks. First, they need this huge "sorted" codebook in both the watermark information computing procedure and the watermark image recovering procedure. Second, they need to search through the "sorted" codebook to find the most similar codeword. It is very time consuming. In order to overcome these two drawbacks, we shall propose a new method that needs no codebooks as follows.

3. The Proposed Method

In this section, we shall present our method. To begin with, we shall first illustrate how to compute a watermark table. Then, we shall move on to the recovery of the watermark image.

3.1. *Watermark Table Computing*

In this subsection, we shall show how we can compute the watermark table. If we want to compute a watermark table for an image, we have to take the following steps. First of all, we partition the original image into non-overlapping blocks. Each block consists of $k \times k$ pixels, where k is a pre-determined value. We record all the blocks from the top left corner to the bottom right in the raster order. After the recording, we get blocks $B_1, B_2, \ldots, B_m$, where m is the total number of blocks in the original image. All these numbered blocks are taken as individual blocks with $k \times k$ dimensions each. This means that each B_i consists of $k \times k$ variables and can be represented as $B_i = (b_{i1}, b_{i2}, \ldots, b_{i,k\times k})$. In the next step, we use the **PCA** procedure to derive the first principal component direction from all the numbered blocks. The first principal component direction, D_1, also consists of $k \times k$ variables $(d_1, d_2, \ldots, d_{k\times k})$. The first principal component direction generation process is illustrated in Section 2.1. After getting the first principal component direction, we project B_i onto the linear subspace by using D_1. In other words, we can calculate the projection points of B_1 by using $(b_{i1} \times d_1 + b_{i2} \times d_2 + \cdots + b_{in} \times d_n)$. Following the same procedure, we can get the projection points of all the numbered blocks. The projection point of the first block B_1 is called P_1, the projection point of the second block B_2 is named P_2, and so on and so forth. From all the projection points, we pick out the one with the maximum value $P_{\max}$ and the one with the minimum value $P_{\min}$, respectively. That is to say, all the projection points are in the range between $P_{\max}$ and $P_{\min}$. Then, we go a step further and define a new variable *Interval_Number*. By the value of *Interval_Number*, we can divide the range between $P_{\max}$ and $P_{\min}$ into equal parts. For a given block B_i, we can get its projection point P_i and easily figure out which interval it belongs to by using the formula below:

$$I_i = \lfloor (P_i - P_{\min}) \times Interval\_Number/(P_{\max} - P_{\min}) \rfloor. \tag{1}$$

Our watermark image is a binary image; i.e., a watermark image consists of only black pixels and white pixels. We record all the pixels from the top left corner to the bottom right in the raster order. After the recording, we get $WP_1, WP_2, \ldots$, and WP_n, where n is the total number of pixels in the watermark image.

Since we have obtained the necessary data, now we can use them to produce the watermark table. First, we select a secret key *S* as the seed of **PRNG**, or “pseudo random number generator”, whose function is to generate random integers. After giving the seed *S* to **PRNG**, we use the **PRNG** function to generate random integers in the range from 0 to $m - 1$, where m is the number of blocks in the original image. We assume that those n random numbers are $R_1, R_2, \ldots$, and R_n, where n is the number of pixels in watermark image. Here, we can get a sequence of blocks $B_{R_1}, B_{R_2}, \ldots$, and B_{R_n}. The watermark table records the data that comes from the indices of the sequence of blocks. To be more precise, we first calculate I_1 of B_{R_1} according to formula (1), and then we continue to check the pixel WP_1. If WP_1 is a white pixel, we record I_1 into first element wt_1 of the watermark table ***WT***); if WP_1 is a black pixel, the value that we record depends on the formula below:

$$wt_i = \left(I_i + \lfloor Interval\_Number/2 \rfloor\right) \bmod (Interval\_Number). \tag{2}$$

According to formula (2), we record $(I_1 + \lfloor Interval\_Number/2 \rfloor) \bmod (Interval\_Number)$ into the first element wt_1 of the watermark table *WT* if WP_1 is a black pixel. Then, we execute the same procedure on the second block B_{R_2} and all the rest of the blocks to fill out the watermark table **WT**. Now, we have a watermark table ***WT*** with n indices. For a given original image, the owner registers the watermark image, the watermark table (***WT***), and the first principal component direction D with the authentication center. The owner does not reveal his/her secret key until a copyright dispute occurs. When the whole procedure is finished, the original image is under the protection of the watermarking technique. Our algorithm for computing the watermark table is as follows:

Algorithm [**Computing watermark table**]:
Input: Secret key S, original image O, and watermark image W.
Output: Watermark table ***WT*** and first principal component direction D.

Step 1: Partition the original image into non-overlapping blocks $B_1, B_2, \ldots$, and B_m, where m is the total number of blocks in the original image.

Step 2: Use the **PCA** function to derive the first principal component direction D from all the blocks in *Step* 1 and project all the blocks onto the linear subspace by using D.

Step 3: Find the maximum projection value $P_{\max}$ and minimum projection value $P_{\min}$ and divide the range between $P_{\max}$ and $P_{\min}$ into equal parts.

Step 4: Use the secret key S as the seed of the **PRNG** function to generate a sequence of random integers $R_1, R_2, \ldots, R_n$ and pick out blocks according to the sequence of random integers. They are $B_{R_1}, B_{R_2}, \ldots$, and B_{R_n}, where n is the total number of pixels in the watermark image.

Step 5: Use D to compute the projection points P_i of the sequence of blocks $B_{R_1}, B_{R_2}, \ldots$, and B_{R_n}. Use P_i to calculate I_i according to formula (1).

Step 6: Check each watermark pixel WP_i. If WP_i is a white pixel, we record I_i into wt_i of the watermark table (***WT***). If WP_i is a black pixel, we record the value derived from formula (2) into wt_i.

Step 7: Repeat *Step* 5 and *Step* 6 until all WP_i are processed.

Step 8: Output the watermark table ***WT*** and the first principal component direction D.

3.2. *Watermark Image Recovering*

In this subsection, let's see how we can recover the watermark image from the modified image. When there is a dispute over the copyright of a modified image, those who disagree with each other can go ask the authentication center to verify the ownership. The authentication center can partition the modified image into non-overlapping blocks and project those blocks onto the linear subspace derived from the first principal component direction D. The authentication center can get the maximum and minimum projection points $P'_{\max}$ and $P'_{\min}$, respectively. One who claims to be the legal owner must show his/her secret key S' to the authentication center. The authentication center then uses S' as the seed of the **PRNG** function to generate a sequence of integers. The authentication center then picks out blocks from the partitioned modified image according to the sequence of integers. We assume that they are $B'_1, B'_2, \ldots, B'_n$, where n is the total number of pixels in the watermark image. The projection points of the blocks $B'_1, B'_2, \ldots, B'_n$ are $P'_1, P'_2, \ldots, P'_n$. The authentication center computes the index I'_i of the projection point P'_i according to formula (1). Then, the authentication center uses I'_i and wt_i in the watermark table (***WT***) to recover the watermark image. According to the formula below, the authentication center can decide what kind of pixel the watermark pixel WP'_i is.

$$WP'_i = \begin{cases} WhitePixel, & \text{if } |I'_i - wt_i| <= (Interval\_Number)/4, \\ BlackPixel, & \text{otherwise.} \end{cases} \tag{3}$$

When all the watermark pixels WP'_i have gone through the same procedure, the watermark image can be recovered. Now the authentication center can judge the ownership of the modified image by the recovered watermark image. The algorithm for recovering the watermark image is as follows:

Algorithm [**Recovering watermark image**]:

Input: Secret key S', modified image O', watermark table ***WT***, and first principal component direction D.

Output: Watermark image.

Step 1: Partition the modified image into non-overlapping blocks.

Step 2: Project those blocks in *Step* 1 according to the first principal component direction D and find $P'_{\max}$ and $P'_{\min}$.

Step 3: Use S' as the seed of the **PRNG** function to generate a sequence of integers. Pick out blocks $B'_1, B'_2, \ldots, B'_n$ according to the sequence of integers and calculate the corresponding projection points $P'_1, P'_2, \ldots, P'_n$.

Step 4: Compute index I'_i of projection point P'_i according to formula (1).

Step 5: Use I'_i and wt_i from the watermark table (***WT***) to recover the watermark pixels WP'_i by formula (3).

Step 6: Repeat *Step* 4 and *Step* 5 until all the watermark pixels WP'_i of the watermark image are reconstructed.

Step 7: Output the watermark image.

The reason why we do not choose to derive the first principal component direction from the modified image directly is that it is too time consuming. Although we could save storage by recording only the watermark table and recovering the watermark image by re-deriving the first principal component direction from the modified image directly, it would take way too much time to do the re-derivation. Another reason is that the first principal component direction, which is calculated from the original image, may not be identical to that of the modified image.

4. Experimental Results and Discussions

In this section, we shall describe our experiments and the results of those experiments. We took three 512×512 gray-level images – "Lenna", "Plane" , and "Barb" – as the original images in our experiments. The characteristics of these three gray-level images are different. The "Barb" image is the most complex, and the "Plane" image is the least complex of the three gray-level images. This way, we can observe the influence the complexity of the original gray-level images has on our watermarking technique. On the other hand, we chose five different ways to modify images. The first one was JPEG lossy compression with the compression rate 14:1. The second one was blurring done by a 5×5 neighborhood median filter. The third was rotating, where the original gray-level images were rotated one degree in the clockwise direction. The fourth was cropping, where the original gray-level images were cropped to 1/4. The last one was sharpening. Our parameter values and experimental results are as follows.

The watermark images in our experiments were binary images with 64×64 pixels each, and the original images were gray-level images with 512×512 pixels each. The original images were partitioned into non-overlapping blocks of three different sizes in three different sets of experiments to see how the block size influences the result. The sizes were 4×4, 8×8, and 16×16 pixels. Then we used the **PCA** procedure to derive the first principal component direction. After projecting all the image blocks onto the linear subspace by using the first principal component direction, we divided the range between the maximum projection value and the minimum projection value into equal parts, and the numbers of parts, namely *Interval_Number*, were set to be 8, 16, 32, 64, 128 and 256. Here, if we set the *Interval_Number* as 8, it means we only need three bits to record each element in the watermark table; however, if the value of *Interval_Number* is set to be 256, then we must use eight bits to record them, and the storage space occupied will be the same as when codebooks are used. According to the projection values, we can easily figure out which interval a given block belongs to by using formula (1). Now, we must select a secret key S as the seed of **PRNG**. We then picked out blocks from the original images according to the sequences of the integers, which were generated by the **PRNG** function, and found their corresponding indices. The rest of the work was the same as what we described in Section 3.1. When recovering the watermark, the authentication center must get the secret key from the owner to execute the process of authentication. That means we must use secret key S as the seed of **PRNG** and pick out blocks from the modified image according to the sequence of the integers. The whole procedure for recovering the watermark image was clearly described in Section 3.2. The results of our experiments are shown from Fig. 1 to Fig. 6.

Note that the experiments from Fig. 1 to Fig. 3 and experiments from Fig. 4 to Fig. 6 are a little different. In the former experiment, we used the original procedure of our proposed method; in other words, we recorded the watermark table ***WT*** and the first principal component direction D as the key data. In the latter, we recorded some additional data, the maximum value $P_{\max}$ and minimum value $P_{\min}$ of the projection points, in order to spare the time meant for projecting all the blocks of the modified images to find their

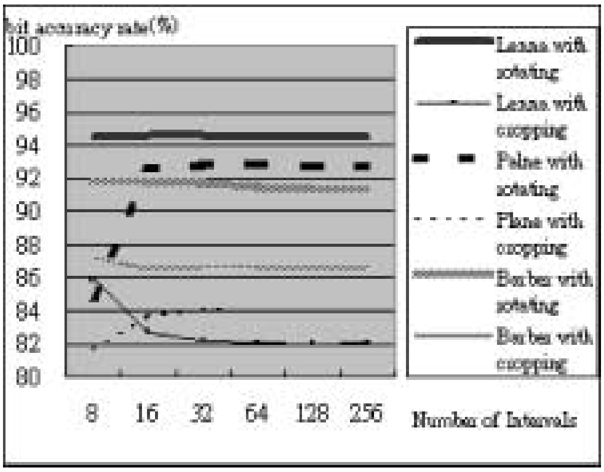

Fig. 1. The size of blocks is 4×4 pixels (without recording $P_{\min}$ and $P_{\max}$).

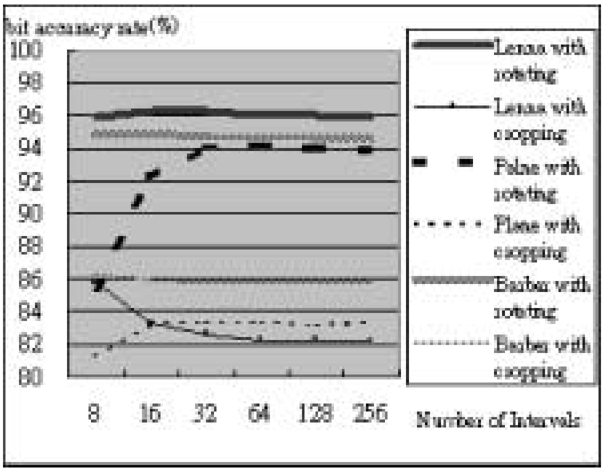

Fig. 2. The size of blocks is 8×8 pixels (without recording $P_{\min}$ and $P_{\max}$).

$P'_{\max}$ and $P'_{\min}$. The results show that the extra storage space spent on $P_{\max}$ and $P_{\min}$ pays off. In figures, we show nothing about the image processing operations of JPEG lossy compression, blurring, and sharpening. The reason is that our method can recover every detail of the original watermark image under these three kinds of image operations. That is, our method gets a 100% bit accuracy rate under JPEG lossy compression, blurring, and sharpening.

According to the results, our bit accuracy rates do not seem to improve much when the number of intervals is greater than 16, which means we do not have to go beyond that number in real-life applications. Besides, we can also save more storage space when we spend only 4 bits recording each element of the watermark table. This is quit impressive because Chang and Tsai's method records each element of the watermark table with 8

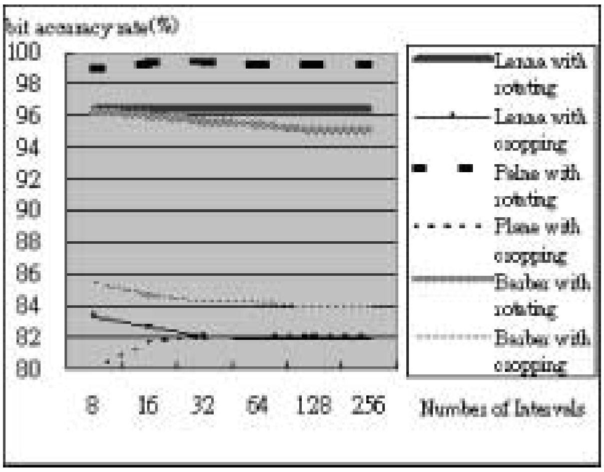

Fig. 3. The size of blocks is 16×16 pixels (without recording $P_{\min}$ and $P_{\max}$).

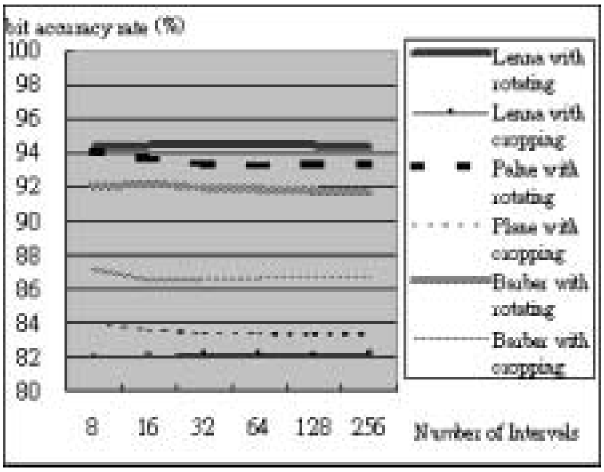

Fig. 4. The size of blocks is 4×4 pixels (with recording $P_{\min}$ and $P_{\max}$).

bits. Another fact shown in figures is that the results, with or without recording the maximum and minimum projection values, do not seem to differ much. This means that if one is most concerned with storage, the choice will be our method without recording the maximum and minimum projection values of the projection points.

In tables, we show the bit accuracy rates of the watermark image in Chang and Tsai's method and our proposed method. As we have discussed earlier, in our method, the number of intervals is set to be 16, and the block size is 16×16 pixels. From Table 1 to Table 3, the test images were Lena, Plane, and Barber. It is obvious that the experimental results of our method are better than those of Chang and Tsai's method in most cases. In summary, our method is practicable and has a superior performance on the bit accuracy rate to Chang and Tsai's method.

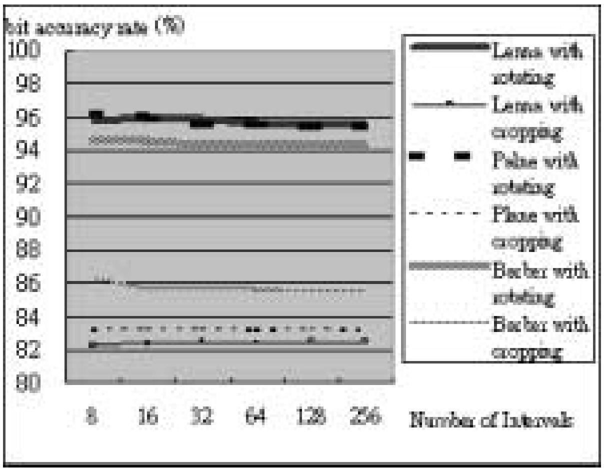

Fig. 5. The size of blocks is 8×8 pixels (with recording $P_{\min}$ and $P_{\max}$).

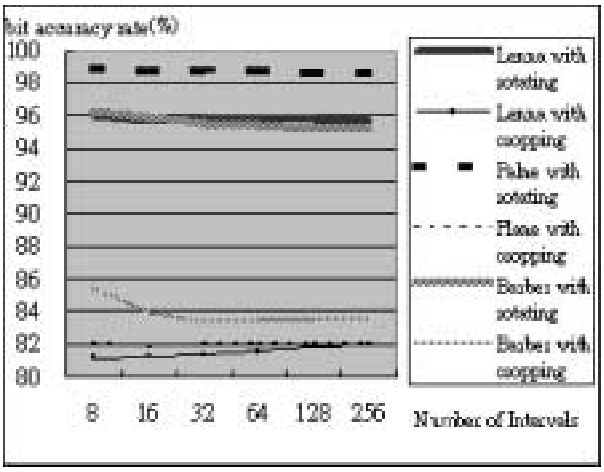

Fig. 6. The size of blocks is 16×16 pixels (with recording $P_{\min}$ and $P_{\max}$).

Table 1

The bit accuracy rates of the watermark image embedding in "Lena" image

Bit Accuracy Rates	JPEG compression	Blurring	Rotating	Cropping	Sharpening
Chang and Tsai's method	99.95%	99.95%	93.65%	86.40%	99.78%
Our proposed method	100%	100%	96.48%	82.67%	100%

Table 2
The bit accuracy rates of the watermark image embedding in “Plane” image

Bit Accuracy Rates	JPEG compression	Blurring	Rotating	Cropping	Sharpening
Chang and Tsai’s method	100%	99.90%	95.53%	95.60%	99.41%
Our proposed method	100%	100%	99.34%	81.76%	100%

Table 3
The bit accuracy rates of the watermark image embedding in “Barber” image

Bit Accuracy Rates	JPEG compression	Blurring	Rotating	Cropping	Sharpening
Chang and Tsai’s method	99.91%	99.65%	92.55%	93.04%	98.24%
Our proposed method	100%	100%	96.07%	84.23%	100%

5. Conclusions

In 2000, Chang and Tsai offered a watermarking technique that produces the watermark table of codebooks and registers the watermark table with an authentication center. Although creative, Chang and Tsai’s method has two drawbacks. First, a “sorted” codebook is needed, which means larger storage space needed. Second, Chang and Tsai’s method must search the “sorted” codebook to find out the most similar codeword, which means more processing time required. In order to overcome these two drawbacks, in this paper, we have proposed our new method that depends on no codebook. In our method, we use the **PCA** function to project the blocks onto a linear subspace. Then, we get indices directly from the positions of the projection points. Both the modified blocks and the projection points are used to reconstruct the watermark. After such modifications as JPEG lossy compression, blurring, sharpening, rotating, and cropping, our method can still recover the watermark images successfully. As our experimental results show, our proposed method is indeed a practicable and efficient watermarking technique.

References

Bender, W., D. Gruhl, N. Morimoto and A. Lu (1996). Technique for data hiding. *IBM System Journal*, **35**(3), 313–336.

Celik, M.U., G. Sharma, E. Saber and A.M. Tekalp (2002). Hierarchical watermarking for secure image authentication with localization. *IEEE Transactions on Image Processing*, **11**(6), 585–595.

Chang, C.C., T.S. Chen and P.F. Chung (2000). A technique of watermarking for digital images using (t, n)-threshold scheme. *Informatica*, **24**(1), 51–55.

Chang, C.C., and C.S. Tsai (2000). A technique for computing watermarks from digital images. *Informatica*, **24**(3), 391–396.

Hwang, M.S., C.C. Chang and K.F. Hwang (2000). Digital watermarking of images using neural networks. *Journal of Electronic Imaging*, **9**(4), 548–555.

Lee, R.C.T., Y.H. Chin and S.C. Chang (1976). Application of principal component analysis to multikey searching. *IEEE Transactions on Software Engineering*, **2**(3), 185–193.

Voyatzis, G., and I. Pitas (1999). Protecting digital-image copyrights: a framework. *IEEE Computer Graphics and Applications*, **1**, 18–24.

Wolfgang, R.B., and E.J. Delp (1996). A watermark for digital image. *Proceedings of the 1996 International Conference on Image Processing*, **3**, 219–222 (in Lausanne).

C.-C. Chang received his BS degree in applied mathematics in 1977 and his MS degree in computer and decision sciences in 1979, both from the National Tsing Hua University, Hsinchu, Taiwan. He received his Ph.D in computer engineering in 1982 from the National Chiao Tung University, Hsinchu, Taiwan. During the academic years of 1980–1983, he was on the faculty of the Department of Computer Engineering at the National Chiao Tung University. From 1983–1989, he was on the faculty of the Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. Since August 1989, he has worked as a professor in the Institute of Computer Science and Information Engineering at National Chung Cheng University, Chiayi, Taiwan. Dr. Chang is a fellow of IEEE, a fellow of IEE and a member of the Chinese Language Computer Society, the Chinese Institute of Engineers of the Republic of China, and the Phi Tau Phi Society of the Republic of China. His research interests include computer cryptography, data engineering, and image compression.

C.-S. Chan received his BS degree in computer science in 1999 from the National Cheng Chi University, Taipei, Taiwan and the MS degree in computer science and Information Engineering in 2001 from the National Chung Cheng University, ChiaYi, Taiwan. He is currently a PhD student at Computer Science and Information Engineering at the National Chung Cheng University, Chiayi, Taiwan. His research fields are image hiding and image compression.

Vandens ženklų schema, pagrįsta pagrindinių komponenčių metodu

Chin-Chen CHANG, Chi-Shiang CHAN

Šiame straipsnyje pateikiamas naujas vandens ženklų formavimo skaitmeniniuose vaizduose algoritmas. Įterpiant vandens ženklą, pradinis vaizdas yra skaidomas į blokus, kurie yra projektuojami į tiesinį poerdvį pagrindinių komponenčių metodo pagalba. Vandens ženklų lentelė yra užduodama projekcijomis į šį poerdvį, kurios kartu su lentele yra vėliau panaudojamos atstatant vandens ženklą iš nagrinėjamo vaizdo, sugretinant skaitmeninį vaizdą su vandens ženklais ir atitinkama lentele, pagal pagrindinių komponenčių metodą. Pasiūlytas algoritmas buvo testuojamas JPEG formato vaizdams, ir pasirodė esąs stabilus vaizdų apdorojimo procedūroms (vaizdo karpymui, posūkiui, kontrasto keitimui, dėmių išlyginimui.