A Shift-Resisting Public Watermark System for Protecting Image Processing Software

Ding-Yun Chen¹, Ming Ouhyoung¹, Ja-Ling Wu¹²

¹Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
²CyberLink Inc., Taiwan

{dynamic, ming, wjl}@cmlab.csie.ntu.edu.tw

Abstract

Digital watermark is a technique used to protect the ownership of images and their associated software. A useful watermark scheme should be visually imperceptible and must be robust to resist various attacks: such as JPEG compression, pixel-shifting, cropping, blurring, etc. In this paper, we propose a public watermarking system to protect image processing software and images. We embed a binary image as watermark in DCT domain, and the watermarked image is imperceptible by human visual system and robust to image shifting or cropping. We use feature-based approach to synchronize the watermark positions during embedding and extracting. For better image quality, the more complex in a block, the more bits can be embedded. Our approach doesn't need the original image for watermark extraction, which is a useful feature in protecting software applications such as real-time generated panorama images.

1. Introduction

All watermarking methods share the same generic building blocks: a watermark embedding system and a watermark extraction system [2]. Fig. 1 shows the generic watermark embedding process for protecting images. The input to the scheme is the watermark (W), the original image (I) and an optional secret key (K). The watermark can be a number, text, or an image. The secret key may be used to enforce security. The output to the scheme is the watermarked image (I'). Fig. 2 shows the generic watermark extracting process. The input to the scheme is the test image (I'), the secret key, and, depending on the method, the original image and/or the original watermark. The output is either the extracted watermark (W') or some kind of confidence measure indicating how likely it is for a given watermark at the input to be present in the test image.

There are three types of watermarking systems, and their difference is in the combination of input and output [3]:
1. Private watermarking (non-blind watermarking) system requires at least the original image (I'×K×W ? {0,1}).
2. Semi-private watermarking (semi-blind watermarking) system does not use the original image for detection but answers the question (I'×K×W ? {0,1}).
3. Public watermarking (blind watermarking) system remains the most challenging problem since it requires neither the original image (I) nor the embedded watermark (W) (I'×K ? W').

Depending on the watermarking application and purpose, different requirements arise resulting in various design issues. The purpose of our watermarking system is to protect the ownership of image processing software, which was initially intended to be put on the Internet for personal use, but not for commercial use. Traditional software uses CD-keys to be protected, but was forced to have two versions, one for limited-function trial use, the other complete-function one for legal use. Our blind watermark approach encourages full-feature software to be put on the Internet for trial use, but with a copyright for personal only. Commercial use of the software is protected by the inherent blind watermark, thus copyright protection can be guaranteed. When we design our watermarking system, the following issues have to be taken into account:

- **Content of watermark**: Watermarks, which are semantic meaningful patterns (such as binary images) that can be directly recognized by naked eyes, are important for Oriental culture. There are more and more papers [1,12,13,14] that choose semantic meaningful patterns instead of a serial number. Thus every one can identify the owner of image by seeing the extracted pattern. So, we choose a binary image as our watermark content.

- **Type of watermarking system**: Watermarking methods using the original data set in the extracting process usually feature increased robustness not only towards noise-like distortions, but also distortions in the data geometry since it allows the detection and inversion of geometrical distortions [2]. In many applications, access to the original data is not possible. The purpose of our watermarking system is to protect the ownership of image processing software and images. For an image processing software, such as VideoVR [4], which is a panorama construction tool, it is impractical to use the original image for watermark extracting, because the panoramic images

---

Watermark (W)

Original image (I)

Optional secret key (K)

Watermark embedding system

Watermarked image (I')

Fig. 1 Generic watermark embedding process

Watermark (W)

Test image (I')

Optional secret key (K)

Watermark extracting system

Extracted watermark (W')

Fig. 2 Generic watermark extracting process
are generated from the user’s side, and no original image is stored. To protect an image processing software, the software can embed the watermark when a user saves image in a file. When some image is illegally used, the ownership of the software can be verified since his software produces a watermark. Therefore, we design a public watermarking system, which can extract watermark from only the test image (I’ W' or {0,1}).

- **Choice of workspace**: The hiding operation can be performed in the spatial domain, such as the research of Patchwork algorithm [6,11] or other methods [14]. Nevertheless, this operation is most often carried out in a transform domain, such as Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) [1,9,10,12,18], Mellin-Fourier Transform [7,8], Wavelet Transform [15,16], etc. Most of still images are compressed as JPEG to communicate through Internet currently. Thus, operate in the DCT domain are often more robust to JPEG compression. Additionally, embedding operator in the compression domain will minimize the computation time. Therefore, we choose 8x8 DCT as workspace.

- **Imperceptibility**: One of the most important requirements is the perceptual transparency of the watermarked image (I). Artifacts introduced through a watermarking process are not only annoying, but also will reduce the commercial value of the watermarked image. In general, the more information to be embedded, the greater the reduction of image quality. There is a trade-off between the amount of embedded information and image quality. Therefore, to obtain visually better image quality and to embed more watermark bits, the more complex in a block, the more watermark bits should be embedded. Though the value of peak signal-to-noise ratio (PSNR) decreases more, it is still difficult for human visual perception to detect artifacts in a complicated block. On the other hand, adding watermark to a smooth block will easily expose artifacts.

- **Robustness**: Alignment problem during watermark extraction is usually inherent in watermarking system. Some image processing will destroy embedded watermark or let it miss the place of original embedded watermark during watermark extraction. It is almost impossible that a watermarking system can resist any kind of distortion or attack. Thus, robustness of a system must take practical requirements into consideration. Users often crop desired region of image. Some methods [1,14] can attack cropped image, but only if the image size is not changed. When image size is changed, those methods will lose alignment and can’t find the place of embedded watermark. One method [12], which embeds watermark in DCT domain, is robust against translation and area cropping, but image size can’t change and can’t translate in x-axis and y-axis simultaneously. Some methods use Mellin-Fourier transform as workspace to resist affine geometric transformation [7,8]. This is because the property of Mellin-Fourier transforms: DFT magnitude is a circular transformation [7,8]. This is because the property of Mellin-Fourier transforms: DFT magnitude is a circular transformation invariant, and log-polar mapping convert scaling and rotation to translation. For an image processing software, images are generated from the user’s side, and extraction process doesn’t know what kind of image to embed the watermark. The method therefore cannot resist cropping of unknown position and unknown original image size. Our approach can resist the above condition to protect an image processing software, because we use feature-based approach to synchronize the watermark embedding position to extracted positions. That is, each block is embedded information of different watermark positions, according to the feature of each block. Therefore, our approach can resist image cropping and keep the remaining part of an image, and so the size of watermarked image has been changed.

- **Secret Key**: An unauthorized user without secret key cannot detect if an image has embedded watermark. Secret key is not a requirement in our approach now, so we don’t use a secret key. If a secret key is need, we can simply use a “Linear Feedback Shift Register” [17] to generate a random sequence for permuting the watermark in embedded process and reversing permutation in extracted process [1].

In short, we propose a public watermarking system to protect image processing software. This paper is organized as follows. The embedding and extracting phase is described in Chapter 2 and 3. In Chapter 4, the experimental results are shown. The conclusion and future work is stated in Chapter 5.

### 2. Embedding Phase

Embedding phase is done by retrieving watermark bits and changing the AC values in the DCT domain of the original image. Our public watermarking system only refers target image to extract watermark (I’ W' or {0,1}). Fig. 3 shows the embedding phase. There is a binary image of 64x64 in Fig. 3, which is a typical watermark we want to embed. When we take an image, first we calculate complex of each 8x8 block. According to each block complex, we can decide amount of hidden bits in the block, this part describe in section 2.1. To avoid watermark being destroyed by shifting and cropping the target image, we predetermine watermark positions by detecting block characteristics in every 8x8 block. If some blocks can be embedded with more watermark bits, more watermark bits should be put in different positions to cover all watermark bits, this part describe in section 2.2. Once we determine how much bits can be embedded in each block and where the embedded position are, we transform image intensity to

#### Original image

- Intensity image
- FDCT
- Embedding bits
- IDCT
- Intensity watermark image

#### Watermarked image

- Amount of hidden watermark bits
- Watermark position
- Original watermark
- Two marker bits

---

**Fig. 3 Watermark embedding phase**
DCT domain and change the AC values to embed the information, as described in section 2.3. Moreover, we embed two marker bits for finding out the DCT block position during extracting phase, and will be described in section 2.4.

2.1 Amount of hidden watermark bits

First, we determine the amount of hidden watermark bits in each 8x8 DCT block by the block complex in intensity domain. If some watermark bits are embedded in a “smooth” image block, the artifacts will obvious. On the other hand, if an image block is “complicated”, the artifacts effect will be hard to see after embedding watermark bits. There are many methods to determine block complexity. We select the mean of gradient magnitude (see equation 3) of each block to measure block complex. The more complex a block is, the more watermark bits can be embedded. Fig. 4 shows how to determine the amount of hidden watermark bits for embedding.

The gradient of x and y direction is $E(i, j)$:

$$E(i, j) = \nabla I = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = \left( S_x, S_y \right)$$  \hspace{1cm} (1)

The gradient magnitude is $|E(i, j)|$:

$$|E(i, j)| = \sqrt{S_x^2 + S_y^2}$$  \hspace{1cm} (2)

In short, we use Sobel edge detector to obtain the gradient magnitude. Mean of gradient magnitude in the $(a,b)$ block is $M_E(a,b)$:

$$M_E(a,b) = \frac{1}{64} \sum_{i=1}^{8} \sum_{j=1}^{8} |E(8a+i,8b+j)|$$  \hspace{1cm} (3)

Where $M_E(a,b)$ is truncated from 0 to 199. If $M_E(a,b)$ is large, the 8x8 block should have many edges, and we consider the block to be complex. Thus we can embed more bits in a complex block that will be not obvious to human vision. Amount of hidden watermark bits of a block is determined by:

$$\text{AmountBits} = \left\lfloor \frac{M_E(a,b)}{40} \right\rfloor \times U$$  \hspace{1cm} (4)

where $U$ is a basic unit of embedded watermark bits in a block, and its value will be calculated in section 2.2. That is, embed $U$ bits if the value of $M_E(a,b)$ is from 40 to 79, embed $2U$ bits if the value is from 80 to 119, and the rest can be done similarly. Fig. 5 shows the 512x512 Lena image as an example. Fig. 5 (a) shows mean of gradient magnitude of each block; (b) shows distribution of (a). There are 740 blocks that embed $U$ bits, 381 blocks embed that $2U$ bits, 263 blocks that embed $3U$ bits, and 129 blocks that embed $4U$ bits.

2.2 Choosing embedded watermark position in each DCT block

After calculating the amount of hidden watermark bits, we choose embedded watermark position for those bits according to characteristics of each 8x8 block. We utilize two characteristics to determine watermark position: intensity and gradient of lower resolution image. Fig. 6 shows the diagram of our idea. We get the lower resolution image (1/64) of the original image and calculate its gradient. The main idea is that the lower resolution image of original image after JPEG compression or other image processing has not much difference from the lower resolution image of the original image. The small difference is in both intensity and gradient. Therefore, we use intensity and gradient to aid the search for the place to embed the watermark.

Therefore, we have two kinds of independent characteristics in an 8x8 block. Fig. 7(a) shows the determination of watermark position. In essence, the
intensity is used for Y-axis, and gradient is used for the X-axis index for a watermark, as shown in Fig. 7(b). If there are more watermark bits to embed, the watermark position must be distributed.

Lower resolution of an image is:

\[
M_i(a,b) = \frac{\sum_{i,j} I(8a+i,8b+j)}{64},
\]

where \(I(8a+i,8b+j)\) is intensity of original image, \(M_f(a,b)\) is mean of intensity in each 8x8 block. \(M_f(a,b)\) is one of the characteristics of a block, which value is from 0 to 255. The other characteristics is gradient magnitude of \(M_f(a,b)\). The definition of gradient magnitude \(|E_{gf}(a,b)|\) is same as equation (2), which value is truncated from 0 to 511.

We get the watermark position in \((a,b)\) block:

\[
\begin{align*}
P_x &= \left[ \frac{M_x(a,b)}{L_x} \right] \\
P_y &= \left[ \frac{E_{gy}(a,b)}{L_y} \right]
\end{align*}
\]

where \((P_x,P_y)\) is the watermark position which is embedded in the \((a,b)\) block, \(L_x\) is level of \(M_f(a,b)\) and \(L_y\) is level of \(|E_{gf}(a,b)|\). For example, suppose \(M_f(a,b)\) is from 0 to 255, \(|E_{gf}(a,b)|\) is from 0 to 511, and \(L_x\) is 4 and \(L_y\) is 8, then the range of \(P_x\) and \(P_y\) is from 0 to 32. That is, we can distinguish watermark position for each block from (0,0) to (32,32). Let the size of watermark image, which we want to embed, be \((w_x, w_y)\), and the range of watermark position, which we can distinguish, be \(\max(M_f) \times \max(|E_{gf}|)\). When we get a watermark position to embed, we must embed \(U\) bits in a block, the basic unit is:

\[
U = \frac{w_x \times w_y}{\max(M_f) \times \max(|E_{gf}|)}
\]

In our system, for example, we want to embed a 64x64 binary image, but we can only distinguish 32x32 different positions, since we use 4 bits as a basic unit. Of course, we can reduce the value of \(L_x\) and \(L_y\) to increase the region of position, but this will reduce the robustness. There are some test statistics in Table 1, which use 512x512 Lena as a test image. Table 1 shows the error of \((P_x,P_y)\) between original image and that after embedding watermark and some image processing. Note that the variable of error is very small ranging from 0.1 to 0.5, and the mean of error ranging from 0.1 to 2.1. Thus, it is reasonable to let \(L_x=4\) and \(L_y=8\) to quantize. The effect of brightness change is different from that of other image processing, and the mean of error is 29.88, so we can find the extracted watermark to shift in y-axis.

Note that maybe there are some positions of watermark bits not included. In section 2.1, we mention a rule to embed more bits in more complex block. Thus, if we can embed more watermark bits, the watermark position must be distributed to other places to cover more parts of watermark image. In block \([a,b]\), the order of watermark position is according to \((a,b), (a+\frac{1}{8}w_x,b+\frac{1}{8}w_y), (a+\frac{1}{2}w_x,b+\frac{7}{8}w_y)\), and \((a+\frac{5}{8}w_x,b+\frac{3}{8}w_y)\) to distribute. To distribute position, note that sum of any two can not equal sum of other two of them add. Fig. 8(a)(c) shows the embedded watermark position before distributing watermark bits, and Fig. 8(b)(d) shows the embedded watermark position after distributing watermark bits.

Table 1 Error of the two characteristics using for watermark position

<table>
<thead>
<tr>
<th>Error before and after embedding watermark and image processing</th>
<th>JPEG Blurred</th>
<th>Sharpen ed</th>
<th>Brightness increased by 30</th>
<th>Gaussian Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of intensity (Y-axis)</td>
<td>0.22</td>
<td>0.16</td>
<td>0.29</td>
<td>29.88</td>
</tr>
<tr>
<td>Variance of error</td>
<td>0.17</td>
<td>0.14</td>
<td>0.28</td>
<td>1.41</td>
</tr>
<tr>
<td>Mean of error (X-axis)</td>
<td>1.10</td>
<td>0.87</td>
<td>1.41</td>
<td>0.73</td>
</tr>
<tr>
<td>Variance of error</td>
<td>1.22</td>
<td>1.25</td>
<td>2.96</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Fig. 8 The grayscale points are watermark bits which are not embedded, where \((a,b)\) is for the Lena image Fig. 19(a), (c),(d) is for a panoramic image Fig. 16a), (b),(d) are cases where more bits are embedded, and so less bits are expressed in gray color.

2.3 Modification of AC value in each DCT block

In order to invisibly embed the watermark that can survive lossy data compression, a reasonable trade-off is to embed the watermark into the middle-frequency range of the image [1]. Fig. 9 shows the middle-frequency coefficients, which we embed watermark bits in shadow area by changing the AC value. The suffix number shows the order of embedding watermark bits in a block. The bits of slant line area are reserved for embedding “marker” bits, which are described in section 3.3.

![Fig. 9 Shows the zigzag ordering of DCT coefficients, and the middle frequency coefficients are shown in the shadow area.](image-url)
image processing. Thus, we compare AC values with DC values in the same DCT block during extracting process. In embedding process, we change AC values to be greater or smaller than DC values by depending on whether we want to embed 0 or 1. For example, we define that if [AC]>[DC], the embedded watermark bit is 1, and if [AC]=[DC], the embedded watermark bit is 0:

\[
w_b = \begin{cases} 
1, & \text{if } |AC|>|DC| \\
0, & \text{otherwise}
\end{cases}
\]  

(8)

where \(w_b\) denotes watermark bits that we want to embed. That is, if we want to embed 1, change AC value to satisfy [AC]>[DC], and if we want to embed 0, change AC value to satisfy [AC]=[DC].

\[
|AC|>|DC|
\]  

satisfy equation 8. Points 1, 2 must change value to satisfy equation 8. Points 3, 4, 7, 8 don’t change value because they don’t change value; otherwise we use the following equation to change AC value to satisfy equation 8:

\[
AC = \begin{cases} 
s \times AC_{\text{Dist}}, & \text{if } \left(|AC| > |DC| \right) \text{ and } \left(|AC| < |DC| \right) \\
AC_{\text{Dist}}, & \text{otherwise}
\end{cases}
\]  

(11)

Where

\[
s = \begin{cases} 
1, & \text{if } |AC| \geq 0 \\
-1, & \text{if } |AC| < 0
\end{cases}
\]

\[AC_{\text{Dist}} = \begin{cases} 
|AC|_{\text{Dist}}, & \text{if } \left(|AC| > |DC| \right) \\
0, & \text{otherwise}
\end{cases}
\]

(12)

3.1 To be a public watermarking system \(W' \rightarrow W'\), we must add some information to find out the original image during extracting phase, as described in section 3.1. To be a public watermarking system \(W' \rightarrow W'\), we embed two “marker” bits in some complicated blocks. Although the two bits are overhead in some blocks, thus it can’t improve the quality of extracted watermark; the two bits are useful during extracting phase. First, our system becomes a “public” watermarking system. There are many advantages of being a public watermarking; such as we can
embed different kinds of watermark using the same extracting phase. Second, before we extract all watermark bits, we can "prune" unwanted images that do not embed our watermark, by comparing the two bits. Thus, we can save much time in searching a protected image from an image database.

We embed two marker bits in DCT domain as show in the slant area of Fig. 9. We define the two marker bits as bit 1 is 1 and bit 2 is 0. The definition of modifying the marker bits value is same as equation 8. For better image quality, the same as in section 2.1, we choose more complicated blocks to embed the two bits. Once $M_E(a, b)$ is greater than 80 in a block, we embed the two marker bits. That is, we modify equation 4 as follow:

$$\text{AmountBits} = \left\lfloor \frac{M_E(a, b)}{40} \right\rfloor \times U + B$$  \hspace{1cm} (12)

Where

$$B = \begin{cases} 2 & \text{if}(M_E(a, b) > 80) \\
0 & \text{otherwise} \end{cases}$$

3. Extracting Phase

Extraction phase is done by retrieving bits from DCT domain and putting them in correct watermark positions. Fig. 12 shows the extracting phase. When we want to extract watermark from a test image, we must find out the original 8x8 DCT block. Once the extracted blocks are the same as the embedding phase, we have a chance to extract the correct watermark. We get the original DCT block by extracting the two marker bits, as described in section 2.4. Moreover, we can roughly decide whether the test image is a protected image by comparing the two marker bits.

Once we get the original DCT block, we can extract watermark bits correctly by comparing AC value with DC value. Similar to the embedding phase, we calculate each image block complexity to determine how much bits to be embedded and where the embedded position are (similar to section 2.1 and 2.2). Therefore, we can extract watermark bits and put it in correct position to obtain a watermark.

![Fig. 12 Watermark extracting phase](image-12.png)

3.1 Find out the original 8x8 DCT block

We refer to the two marker bits to find out the original 8x8 DCT block. In short, we “try” all kinds of shift to extract the two marker bits, and compare between their normalized cross-correlation (NC). We can find out the original block according to the highest value in normalized cross-correlation. Fig. 13 shows a diagram of how to find the original 8x8 DCT block.

![Fig. 13 Get Original 8x8 DCT block using two marker bits](image-13.png)

Because we embed bits to a 8x8 DCT block, there is only one correct choice out of 64 shifting combinations. Generally speaking, we can extract two marker bits from all blocks among 64 kinds of shift, and then find out the highest NC value. But this will spend much time. Therefore, we randomly choose some blocks across the image. Those chosen blocks must be complex enough according to equation 12, and then extract two marker bits from 64 kinds of shift. Each kind of shift has a NC value comparing extracted marker bits with embedded marker value. Of course, the highest NC is the most possible place to extract the correct watermark. The definition of normalized cross-correlation (NC) is the ratio of correct extracted bits to all extracted bits:

$$NC = \frac{\sum w^w}{\sum w^t}$$  \hspace{1cm} (13)

If some other NC values are very close to the highest NC value, it means that there is perhaps some error. In this situation, we choose more blocks to extract, in order to obtain more correct shift position, as shown in the place of "If ambiguous" in Fig. 13. We define the case being ambiguous as: $SecMax + 0.1 > Max$, where Max is maximum NC value, and SecMax is secondary maximum NC value. If ambiguous cases happen, note that the loop will repeat at most 2 times. Otherwise, it will spend much time during extracting image. In our implementation, we randomly choose 10 blocks and repeat at most 2 times.

Fig. 14 shows examples of normalized cross-correlation (NC) value of each kind of shift. (a) an image doesn’t shift or shift (8m,8n) pixel. (b) image shift (5,6) pixel or (5+8m,6+8n) pixel, this is because (b) image need to shift (3,2) pixel, and (5,6)+(3,2)=(8,8). Where m and n are positive integer.
The NC value has another use. When the maximum NC is small, it means that the image may not embed our watermark. Therefore, we define a threshold to be 0.7, that is, when the maximum NC is lesser than 0.7, we don’t need to extract all watermark bits.

3.2 Extract watermark bits

To extract bits in DCT block, we first calculate amount of hidden bits to be embedded. This part is same as embedded phase described in section 2.1. Then take the opposite AC value according to Fig. 9. The same as definition of embedding phase, we compare $|AC|$ with $|DC|$ to obtain 0 or 1:

$$w_b = \begin{cases} 
1, & \text{if } |AC| > |DC| \\
0, & \text{if } |AC| \leq |DC| 
\end{cases} \quad (14)$$

Put $w_b$ in correct watermark position according characteristic of its block, which is same as embedded phase described in section 2.2. After dealing with all DCT blocks, we get all embedded watermark bits and form a binary image. We can directly recognize the ownership of image by looking the watermark.

For a public watermarking system, it can’t refer original watermark. In order to measure the correct ratio of extracted watermark, we define $NC_{AW}$ and $NC_{EW}$ to compare extracted with original watermark using in experiment result. The $NC_{AW}$ value compares extracted watermark bits with original watermark bits in the extracted watermark position. That is, $NC_{AW}$ value measures percentage of correct within extracted watermark bits. The $NC_{EW}$ value compare extracted watermark bits with all original watermark bits. That is, the $NC_{EW}$ measures quality of extracted watermark.

4. Experiment result

Our approach has already combined with our own VideoVR system, which is a panorama construction tool from live video of surrounding environment. Other commercial products can also use our system to protect their copyright. Why not just use a serial number (CD-key) to protect a software? Since this is an Internet society, and many software were put on web sites for non-commercial use, both for trial use and for advertisement. However, there is a need to avoid the software being used commercially without proper licensing. Therefore, a public watermarking system is proposed.

Fig. 15 shows some 64x64 binary images that we use as watermarks. Fig. 16, Fig. 17 and Fig. 18 are three examples for panoramic image and cropped part of them. In all three examples, (a) of them are images which have embedded watermark and compressed to JPEG, and (b) of them are cropped some part from (a), as shown a block line rectangle in (a). The PSNR of Fig. 16 is only 30.061, this is because the image has many complicated block. Thus, we can embed more watermark bits, and it’s still hard to detect artifacts, as compared to Fig. 16 (f)(b) and (g)(b). On the other hand, the more bits are embedded, the more watermark quality is extracted, as shown in Fig. 16 (d), the $NC_{EW}$ value is 0.993, the highest among examples.

Fig. 19 (a) shows 512x512 original Lena image. Fig. 19 (b) shows Lena image, which are embedded watermark and compress to JPEG. Fig. 19 (c)-(h) show some image processing from (b), including shifting, cropping, blurring, sharpening, Gaussian noise adding and brightness. Note that the image processing and JPEG compression operate by other software, which is one popular commercial image processing software. Fig. 20 (b)-(h) shows the watermark extracted from Fig. 19 (b)-(h). Those $NC_{EW}$ value is also high, except Fig. 20 (h). This is because we use image intensity as one of our characteristics to decide watermark position. When brightness arises, the watermark position will arise, too. Thus, we will see the watermark shift in y-axis, even so, we can recognize the meaning of extracted watermark.

We also test some images of different resolution, as shown in Fig. 21, and different watermarks, as shown in Fig. 15. We can still recognize every extracted watermark by naked eyes.

Our approach can also resist mosaic attack [19]. The attack chops a watermarked image into smaller images, which are stuck back together when the browser renders the page, as shown in Fig 22.
Fig. 16 (a) is a panoramic image of a central courtyard; the image size is 2729x217, which is embedded a watermark using Fig. 15 (a), and then compressed to JPEG. The PSNR of (a) is 30.061 after embedding watermark. (b) is cropped from (1651,10) to (2149,209) of (a) as shown a block line rectangle in (a); the image size is 500x200. (c) is the extracted watermark from (a) before JPEG compression. (d) is the extracted watermark from (a). The (Normalized Correlation) NC\text{EW} of (e) is 0.997, and NC\text{AW} is 0.993. (e) is the extracted watermark from (b). The NC\text{EW} of (f) is 0.971, and NC\text{AW} is 0.845. (f),(h) show some part of (b). (g),(i) show the same part of original image, which doesn’t embed watermark.

Fig. 17 (a) is a panoramic image of an outdoor scene; the image size is 2620x287, which is embedded a watermark using Fig. 15 (a), and then compressed to JPEG. The PSNR of (a) is 36.088 after embedding watermark. (b) is cropped from (9,11) to (708,260) of (a) as shown a block line rectangle in (a); the image size is 700x250. (c) is the extracted watermark from (a) before JPEG compression. (d) is the extracted watermark from (a). The NC\text{EW} of (e) is 0.980, and NC\text{AW} is 0.961. (e) is the extracted watermark from (b). The NC\text{EW} of (f) is 0.918, and NC\text{AW} is 0.779. (f),(h) show some part of (b). (g),(i) show the same part of original image, which doesn’t embed watermark.
Fig. 18 (a) is a panoramic image of an outdoor scene; the image size is 2596x296, which is embedded a watermark using Fig. 15 (a), and then compressed to JPEG. The PSNR of (a) is 37.953 after embedding watermark. (b) is cropped from (1731,14) to (2430,263) of (a) as shown a block line rectangle in (a); the image size is 700x250. (c) is the extracted watermark from (a) before JPEG compression. (d) is the extracted watermark from (a). The NC\text{EW} of (e) is 0.964, and NC\text{AW} is 0.875. (e) is the extracted watermark from (b). The NC\text{AW} of (f) is 0.936, and NC\text{AW} is 0.577. (f),(h) show some part of (b). (g),(i) show the same part of original image, which doesn’t embed watermark.

Fig. 19 (a) is original Lena image; whose size is 512x512. (b) is 512x512 Lena image, which has embedded watermark, and compress to JPEG. The PSNR is 40.390 after embedding watermark. (c) is shifted (52,17) from (b), and compressed to JPEG again. The gray line denotes the image boundary. (d) is cropped from (71,45) to (438,402) of (b), and compress to JPEG again; the image size is 368x358. (e) generated by processing blur from (a), and compressed to JPEG again. (f) generated by image sharpening from (b), and compressed to JPEG again. (g) generated by adding Gaussian noise to (b), and compressed to JPEG again. (h) Brightness value increased by 30 from (b), and compress to JPEG again. Where the extract watermarks from (b) to (h) are shown from (b) to (h) in Fig. 20. Note that the image processing and JPEG compression are operated by one popular commercial image processing software: PhotoShop.
domain. Execution time of extracting watermark includes finding out original 8x8 DCT block and extracting watermark.

Table 2 Execution time of embedding phase using a Pentium III 533 MHz PC

<table>
<thead>
<tr>
<th>Test image</th>
<th>Image size</th>
<th>PSNR</th>
<th>NC EW</th>
<th>NC AW</th>
<th>Extract image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>512x512</td>
<td>42.214</td>
<td>0.925</td>
<td>0.726</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>256x256</td>
<td>38.620</td>
<td>0.927</td>
<td>0.563</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>512x512</td>
<td>30.595</td>
<td>0.988</td>
<td>0.860</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>256x256</td>
<td>32.709</td>
<td>0.972</td>
<td>0.860</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>512x512</td>
<td>38.873</td>
<td>0.970</td>
<td>0.853</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>256x256</td>
<td>35.004</td>
<td>0.946</td>
<td>0.586</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 21 test some images of different resolution and different watermarks

For software protection, we want to know whether an image has been embedded watermark or not before visually inspecting the extracted watermark pattern. Therefore, we define 0.8 as the threshold of NC EW in our watermarking system. When the NC EW of test image is larger then 0.8, the system extract watermark to recognize the copyright. This is useful to search the protected image from a large image database.

Table 3 Execution time of extracting phase using a Pentium III 533 MHz PC

<table>
<thead>
<tr>
<th>Execution time of extracting watermark</th>
<th>Fig. 16 (a)</th>
<th>Fig. 17 (a)</th>
<th>Fig. 18 (a)</th>
<th>Fig. 16 (b)</th>
<th>Fig. 17 (b)</th>
<th>Fig. 18 (b)</th>
<th>Fig. 19 (b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>2729x217</td>
<td>2620x287</td>
<td>2596x296</td>
<td>700x200</td>
<td>700x250</td>
<td>512x512</td>
<td>128x128</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>1.32</td>
<td>1.68</td>
<td>1.61</td>
<td>0.26</td>
<td>0.41</td>
<td>0.42</td>
<td>0.61</td>
</tr>
<tr>
<td>Execution time of embedding watermark</td>
<td>Fig. 16 (c)</td>
<td>Fig. 17 (c)</td>
<td>Fig. 18 (c)</td>
<td>Fig. 16 (b)</td>
<td>Fig. 17 (b)</td>
<td>Fig. 18 (b)</td>
<td>Fig. 19 (b)</td>
</tr>
<tr>
<td>Lena</td>
<td>512x512</td>
<td>512x512</td>
<td>512x512</td>
<td>700x200</td>
<td>700x250</td>
<td>512x512</td>
<td>512x512</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.60</td>
<td>0.33</td>
<td>0.72</td>
<td>0.71</td>
<td>0.72</td>
<td>0.61</td>
<td>0.20</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, a public watermarking system is proposed, which can resist image processing attacks, such as JPEG compression, shift, crop, blur, etc. The embedded and extracted watermark is a semantic meaningful pattern. Our system doesn’t need the original image for watermark extraction, which is a useful feature in protecting an image processing software. To resist shift and crop, we propose a feature-based approach to synchronize the watermark positions during embedding and extracting. To obtain better image quality, the higher the complexity in a block, the more watermark bits can be embedded.
Acknowledgement
The author would thank Chun-Hsiang Huang for the support of this work. This project is partially funded by the National Science Council (NSC, Taiwan) under the grant number NSC88-2622-E-002-002.

Reference

Ding-Yun Chen received the B.S. degree in information and computer engineering from Chung Yuan Christian University, Chung-Li, Taiwan, in 1997, and the M.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1999. He is currently a Ph.D. student in the department of computer science and information engineering, National Taiwan University. He was also the adjunct lecturer of the department of computer science and information engineering, National Chi Nan University, Puli, Taiwan, from Aug. 1999 to July 2000. His current research interests include computer graphics, virtual reality, digital image processing, and digital watermarking techniques.

Ming Ouhyoung received the BS and MS degree in electrical engineering from the National Taiwan University, Taipei, in 1981 and 1985, respectively. He received the Ph.D degree in computer science from the university of North Carolina at Chapel Hill in 1990. He was a member of the technical staff at AT&T Bell Laboratories, Middle-town, during 1990 and 1991. Since August 1991, he has been an associate professor in the department of Computer Science and Information Engineering, National Taiwan University. Then since August 1995, he became a professor. Currently he is the Director of Center for Excellence for Research in Computer Systems, College of Engineering. He has published over 70 technical papers on computer graphics, virtual reality, and multimedia systems. He is a member of ACM and IEEE.

Ja-Ling Wu received the B.S. degree in electronic engineering from TamKang University, Tamshoei, Taiwan, R.O.C. in 1979, and the M.S. and Ph.D degree in electrical engineering from Tatung Institute of Technology, Taipei, Taiwan, in 1981 and 1986, respectively. From 1986 to 1987, he was an Associate professor of the Electrical Engineering Department at Tatung Institute of Technology, Taipei, Taiwan. Since 1987, he has been with the Department of Computer Science and Information Engineering, National Taiwan University, where he is presently a Professor. He was also the Head of the Department of Information Engineering, National Chi Nan University, Puli, Taiwan, from Aug. 1996 to July 1998. Prof. Wu was the recipient of the 1989 Outstanding Youth Medal of the Republic of China, the Outstanding Research Award sponsored by the National Science Council, from 1987 to 1992, and the Excellent Research Award from NSC, 1999. Prof. Wu has published more than 200 technique and conference papers. His research interests include algorithm design for DSP, data compression, digital watermarking techniques and multimedia systems.