Images with Self-Correcting Capabilities

[a,b]Jiri Fridrich and [a]Miroslav Goljan

[a]*Center for Intelligent Systems, SUNY Binghamton, Binghamton, NY 13902-6000*
[b]*Mission Research Corporation, 1720 Randolph Rd SE, Albuquerque, NM 87501*
fridrich, bg22976@binghamton.edu

Abstract

In this paper, we introduce two techniques for self-embedding an image in itself as a means for protecting the image content. After self-embedding, it is possible to recover portions of the image that have been cropped out, replaced, damaged, or otherwise tampered without accessing the original image. The first method is based on transforming small 8×8 blocks using a DCT, quantizing the coefficients, and carefully encoding them in the least significant bits of other, distant squares. This method provides very high quality of reconstruction but it is very fragile. The quality of the reconstructed image areas is roughly equivalent to a 50% quality JPEG compressed original. The second method uses a principle similar to differential encoding to embed a circular shift of the original image with decreased color depth into the original image. The quality of the reconstructed image gradually degrades with increasing amount of noise in the tampered image. The first technique can also be used as a fragile watermark for image authentication, while the second technique can be classified as a semi-robust watermark.

1. Introduction

In the past, several techniques [1–11] and concepts based on data hiding or steganography have been designed as a means for tamper detection in digital images and for image authentication – fragile watermarks, semi-fragile watermarks, and self-embedding. The visual redundancy of typical images enables us to insert imperceptible additional information and make the images capable of authenticating themselves without accessing the originals. The goal is to prevent the possibility of creating a forgery that goes undetected. For example, a secure digital camera equipped with a watermarking chip may authenticate every image it takes before storing it on the flash card. The embedded information could be uniquely tied to the camera's serial number thus creating a link between the images and the hardware that took them. Such smart images may play an important role in detecting digital forgeries or establishing the origin of digital images.

1.1 Fragile watermarks

If the inserted watermark is fragile so that any manipulation of pixels will disturb its integrity, one can easily detect the tampered areas by checking for presence of this fragile watermark. One of the first techniques used for detection of image tampering was based on inserting check-sums of gray levels determined from the seven most significant bits into the least significant bits (LSB) of pseudo-randomly selected pixels [1]. This technique provides very high probability of tamper detection, and it can be implemented in such a manner that creating forgeries from one or multiple authenticated images is highly unlikely without a secret key. Yeung and Wong [2,3] use key dependent binary valued functions to encode a binary logo in the pixels of the digital image. The authentication step consists of checking the integrity of the binary logo using the same key dependent binary functions. This authentication fragile watermark is embedded not only in the LSBs of the image but somewhat deeper (± 5 gay scales). For a secure implementation, either the logo or the binary functions must be image dependent.

1.2 Semi-fragile watermarks

Another class of authentication watermarks is formed by semi-robust watermarks. Such watermarks are marginally robust and are less sensitive to pixel modifications. Thus, it is possible to use them for quantifying the degree of tamper and distinguish simple LSB shuffling from malicious changes, such as feature adding and removal. Van Schyndel et al. [4] modify the LSB of pixels by adding extended m-sequences to rows of pixels. For an $N \times N$ image, a sequence of length N is randomly shifted and added to the image rows. The phase of the sequence carries the watermark information.

A simple cross-correlation is used to test for the presence of the watermark. Wolfgang and Delp [5] extended van Schyndel's work and improved the localization properties and robustness. They use bipolar m-sequences of –1's and 1's arranged into 8×8 blocks and add them to corresponding image blocks. The watermark presence is evaluated using classical correlation. Zhu et al. [6] propose two techniques based on spatial and frequency masking. Their watermark is guaranteed to be perceptually invisible, yet it can detect errors up to one half of the maximal allowable change in each pixel or frequency bin depending on whether frequency or spatial masking is used. The image is divided into blocks and in each block a secret random signature (a pseudo-random sequence uniformly distributed in [0,1]) is multiplied by the masking values of that block. The resulting signal depends on the image block and is added to the original block quantized using the same masking values. Errors smaller than one half of the maximal allowable change are readily detected by this scheme. The error estimates are fairly accurate for small distortions. Fridrich [7,8] describes a technique in which an image is divided into medium-size blocks and a robust spread-spectrum watermark is inserted into each block. If watermarks are present in all blocks with high probability, one can be fairly confident that the image has not been tampered with in any significant manner (such as adding or removing features comparable in size to the block). If the watermark correlation is lower uniformly over all image blocks, one can deduce that some image processing operation was most likely applied. If one or more blocks show very low evidence for watermark presence while other blocks exhibit values well above the threshold, one can estimate the probability of tampering and with a high probability decide whether or not the image has been tampered with. Other semi-robust watermarks for detection of tamper in digital imagery based on fragile watermarks have been introduced in [9,10].

1.3 Self-embedding

The idea of self-embedding the image into itself enables not only detection of areas that have been tampered or damaged, but also recovering the missing information. The self-embedded information can be in a fragile or in a semi-fragile form. Thus, self-embedding is a means both for protecting the image content and for authentication. In this paper, we present two different self-embedding methods. In Section 2, we describe a fragile block-based watermark obtained by compressing small image blocks using a DCT and encoding them in the LSBs of other distant blocks. In Section 3, we introduce a semi-fragile watermark obtained by encoding a circularly shifted image with decreased color depth in itself using a method similar to differential encoding. In Section 4, we discuss the properties of both techniques and outline some future directions.

2. Self-embedding (Method 1)

The method starts with dividing the image into 8×8 blocks and transforming each block using a DCT. A specified number of the lowest frequency DCT coefficients are quantized using a quantization matrix corresponding to a 50% quality JPEG. The coefficients are ordered in a zig-zag manner and their values are encoded using a fixed number of bits. The number of coefficients and their encoding are carefully chosen so that the resulting bit-string for each block is exactly 64 bits long. Information about block *B* (e.g., the 64-bit string) is inserted into the LSB of the block $B + \bar{p}$, where $\bar{p}$ is a vector of length approximately 1/3 of the image size with a randomly chosen direction. If two LSBs are used for self-embedding, more quantized coefficients can be encoded using 128 bits rather than just 64. In this case, the recovered self-embedded image is perceptually indistinguishable from a 50% quality JPEG compressed original. This enables us to recover even very small features comparable to the block size. To prevent a pirate from masking a forged piece of an image, the bit-string can be encrypted. The following three steps are carried out for each block *B*:

Step 1 (Preparing the image for embedding).
Gray levels of all blocks are transformed into the interval [–127, 128] and the LSBs of all pixels are set to zero.

Step 2 (Generating the code).
Each 8×8 block *B* is transformed into the frequency domain using DCT. The first 11 coefficients (in zig-zag order) are quantized with the following quantization table *Q* that corresponds to 50% quality JPEG:

$$Q=\begin{bmatrix}16 & 11 & 10 & 16 & 24 & 40 & 51 & 61\\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55\\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56\\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62\\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77\\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92\\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101\\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99\end{bmatrix}.$$

The quantized values are further binary encoded. The bit lengths of their codes (including the signs) are shown in matrix *L*

$$L=\begin{bmatrix} 7 & 7 & 7 & 5 & 4 & 3 & 2 & 1 \\ 7 & 6 & 5 & 5 & 4 & 2 & 1 & 0 \\ 6 & 5 & 5 & 4 & 3 & 1 & 0 & 0 \\ 5 & 5 & 4 & 3 & 1 & 0 & 0 & 0 \\ 4 & 4 & 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Encoding based on L will guarantee that the first 11 coefficients from each block will be encoded using exactly 64 bits. In the rare event when the *i*-th DCT coefficient has absolute value is larger than 2^{L_i-1}, only this maximum available value will be encoded.

Step 3a (Encrypting and embedding using 64 bits).
The binary sequence obtained in Step 2 (e.g., the 64-bit string) is encrypted and inserted into the LSB of the block $B+\vec{p}$, where $\vec{p}$ is a vector of length approximately 1/3 of the image size with a randomly chosen direction. Periodic boundary conditions (torus topology) are used to get the block $B+\vec{p}$ always inside the image. After self-embedding, the marked image is modified very little. In fact, on average 50% of pixel values will not be changed, and 50% of them will be modified by one gray level. The quality of the reconstructed image is worse than for an image that has been JPEG-compressed at 50% quality. This may not be sufficient for capturing details smaller than the block size.

Step 3a (Encrypting and embedding using 128 bits).
There is an obvious tradeoff between the quality of reconstruction and the extent of modifications due to self-embedding. By using two least significant bits for self-embedding rather than just one LSB, the image quality of the reconstruction will be dramatically improved while the changes to the original image will still be very minor. The first 3 coefficients are encoded using the same number of bits as before. The next 18 bits carry information about coefficients No. 4–21. A zero means that the corresponding coefficient is 0, while ones indicate non-zero coefficients. Following these 18 bits, we encode the values of all nonzero coefficients. Coefficients of higher frequencies are encoded with correspondingly fewer bits. If the length of the code is still short enough, up to two next nonzero coefficients between the 22nd and 36th coefficient are also coded (again, their positions first and then their values). The average code length is about 100 bits (1.55 bits per pixel). The code is shorter for blocks from areas of approximately uniform brightness. If the total length of the code is less than 128, zero padding is applied. All 128 bits are utilized for detection of tampered blocks.

The original test image is shown in Figure 1. Figure 2 shows the original image with its content embedded in itself using 2 LSBs. One can easily recover the original license plate (Figure 4) from an image in which the plate has been replaced with a different one (Figure 3).

The security of this self-embedding method including attacks and countermeasures are discussed in [11].

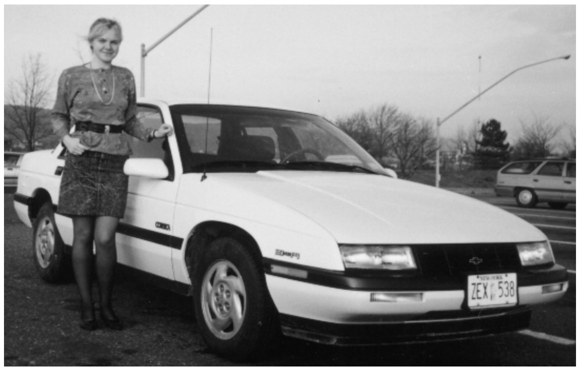

Figure 1. Original image

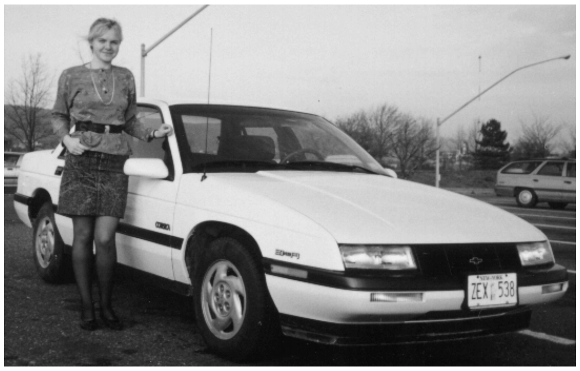

Figure 2. Self-embedded image

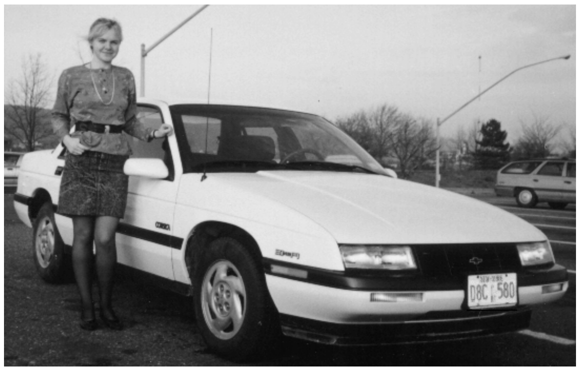

Figure 3. Tampered license plate

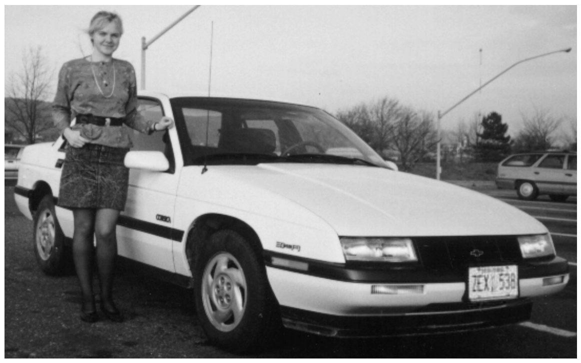

Figure 1. Reconstructed image

3. Self-embedding (Method 2)

The main advantage of Method 1 is the high visual quality of the reconstructed image. However, this has been achieved at the expense of extreme fragility. The embedded information is highly fragile and a simple randomization of the least significant bit will completely erase the embedded information. Even very high quality JPEG compression so commonly used for storing imagery will disturb the embedded data beyond practical use. This severely limits the use of this method. Method 2 is a step towards achieving a robust self-embedding technique. Even though its robustness is not sufficient for JPEG compression quality lower than 90%, it can successfully survive simple least significant bit randomization and adding small amount of noise. The quality of the reconstructed image gradually decreases with the amount of noise added to the image.

The method is a variation of simple differential encoding. First, the color depth of the original image is decreased to 16. Then, the gray levels of the recolored image are transformed to the interval [–8,8]. It is this low color depth image that will be embedded in the original image. The embedding process starts in the upper left corner and proceeds by rows from left to right, top down. We denote the gray values of the original $M\times N$ image as g_{ij}, $0 \le g_{ij} \le 255$, and for the truncated image $-8 \le t_{ij} \le 8$. Similar to Method 1, we perform a cyclic shift on t_{ij} by an integer vector $s = (s_1, s_2)$ to obtain a shifted version st_{ij}

$$st_{ij} = t_{(i-s_1) \bmod M\ (j-s_2) \bmod N}\,.$$

The gray levels of the self-embedded image are denoted g'_{ij}. As the first step, we set $g'_{11}=g_{11}$. Having adjusted the gray level g_{ij} to g'_{ij} we modify the value g_{ij+1} to g'_{ij+1} by enforcing

$$g'_{ij+1} - g'_{ij} = st_{ij}.$$

If we understand the last equation as mod 16, it is clear that we will never have to modify g_{ij+1} by more than ±8, and the average change to the gray level levels will be 4. If g'_{ij+1} spills over 255 or below 0, we subtract 16 or add 16 respectively, to enforce g'_{ij+1} to be in the range [0, 255]. In those rare cases, the change to the original gray level may be larger than 8 not more than 16.

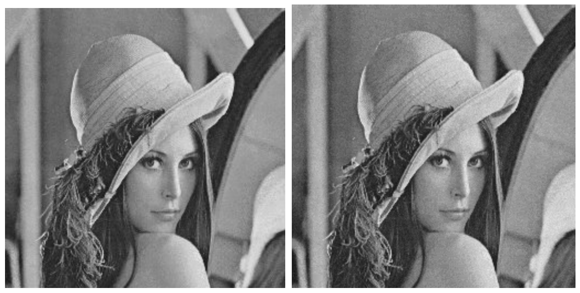

Figure 7. The original and self-embedded images

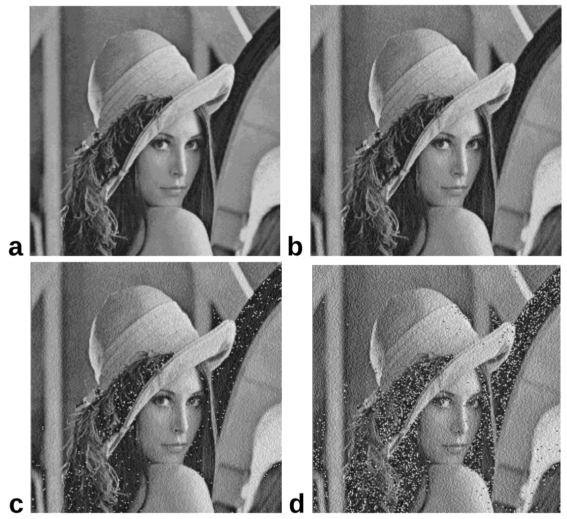

Figure 8. From upper left corner right and down: The recovered images after no attack, after randomizing the LSB, after adding random noise in the range [–1, 1] and [–2, 2]

To reconstruct the color truncated approximation to the original image at pixel (i,j), we calculate the difference $g'_{kl+1} - g'_{kl}$, where $k = (i+s_1) \bmod M$ and $l = (j+s_2) \bmod N$. Figure 7 shows the test image "Lenna" and the same image after it has been self-embedded using Method 2. The RMS difference between the original and the self-embedded image is 4.6 gray scales. Without any distortion, the embedded color truncated image can be extracted without any loss of information (see Figure 8a). If the self-embedded image has been tampered by randomizing the LSB, the reconstructed image becomes somewhat noisy, but retains its content (see Figure 8b). Reconstruction after adding uniformly distributed random noise in the range [–1, 1] and [–2, 2] results in images in Figures 8c and 8d. Clearly, the quality of the reconstructed image rapidly decreases with the amount of added noise. JPEG compression with quality factor 85% erases the embedded information. Method 2 should be viewed as a step towards practical, robust self-embedding methods that can be used with high quality lossy compression. Its robustness is still not sufficient for practical applications, but it gives us a hope that a practical self-embedding method may be within the reach.

4. Conclusions

In this paper, we introduce two methods for self-embedding in which an image is embedded in an imperceptible manner in itself. This gives images the ability to authenticate themselves or repair themselves after intentional or non-intentional tamper, such as feature removal, adding, or replacement, without having to access the original image. Such smart images will play an important role in detecting digital forgeries and recovering damaged or tampered details in images.

In the first technique, the image is divided into 8×8 blocks that are DCT transformed, quantized, and carefully encoded into the LSBs of other, distant 8×8 blocks. If two least significant bits are used for encoding, the quality of the reconstructed image is indistinguishable from a 50% quality JPEG compressed image. The technique can be easily extended to color images.

The second method is a variation of simple differential encoding. First, the color depth of the original image is decreased to 16. Then, the gray levels of the recolored image are transformed to the interval [–8,8]. This low color depth image is embedded in the original image using a principle similar to differential encoding.

The first method is very fragile and simple randomization of the LSBs will completely erase the embedded information. However, the visual quality of the reconstructed image is very high. At the same time, the original image is modified only very slightly. The first method can also be used as a fragile authentication watermark. In the second method, the original image is modified more (on average by 4 gray scales) and the visual quality of the reconstructed image is lower (lower color depth). However, the embedded information can survive adding small amount of noise. The second technique should be viewed as a first step towards self-embedding techniques that are more robust and can survive high quality JPEG compression typically used in digital cameras. This subject will be the focus of our future research.

5. Acknowlegements

The work on this paper was partially supported by Air Force Research Laboratory, Air Force Material Command, USAF, under contracts F30602-98-C-0049 and F30602-98-C-0176. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government.

6. References

[1] S. Walton, “Information Authentication for a Slippery New Age”, *Dr. Dobbs Journal*, vol. 20, no. 4, pp. 18–26, Apr 1995.

[2] M. Yeung, and F. Mintzer, "An Invisible Watermarking Technique for Image Verification", *Proc. ICIP'97*, Santa Barbara, California, 1997.

[3] P. Wong, "A Watermark for Image Integrity and Ownership Verification", *Proc. IS&T PIC*, Portland, Oregon, 1998.

[4] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, “A Digital Watermark”, *Proc. of the IEEE Int. Conf. on Image Processing*, vol. 2, pp. 86–90, Austin, Texas, Nov 1994.

[5] R. B. Wolfgang and E. J. Delp, “A Watermark for Digital Images”, *Proc. IEEE Int. Conf. on Image Processing*, vol. 3, pp. 219–222, 1996.

[6] B. Zhu, M. D. Swanson, and A. Tewfik, “Transparent Robust Authentication and Distortion Measurement Technique for Images”, preprint, 1997.

[7] J. Fridrich, “Image Watermarking for Tamper Detection”, *Proc. ICIP ’98*, Chicago, Oct 1998.

[8] J. Fridrich, “Methods for Detecting Changes in Digital images”, ISPACS, Melbourne, November 4th–6th, 1998.

[9] D. Kundur and D. Hatzinakos, "Towards a Telltale Watermarking Technique for Tamper Proofing", *Proc. ICIP*, Chicago, Illinois, Oct 4–7, 1998, vol 2.

[10] R. B. Wolfgang and E. J. Delp, "Fragile Watermarking Using the VW2D Watermark", *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, Jan 25–27, 1999, pp. 204–213.

[11] J. Fridrich and M. Goljan, "Protection of Digital images Using Self-Embedding", *Symposium on Content Security and Data Hiding in Digital Media*, New Jersey Institute of Technology, May 14, 1999.