2nd 2013 IEEE Conference on Control, Systems and Industrial Informatics (ICCSII)
Bandung, Indonesia, June 23-26, 2013

Content-Dependent Image Watermarking using Local Gray Level Co-occurrence Matrix Texture Features

Iwan Setyawan[1], Ivanna K. Timotius[2]
Dept. of Electronic Engineering
Satya Wacana Christian University
Salatiga, Indonesia
iwan.setyawan@ieee.org[1], ivanna.timotius@ieee.org[2]

Abstract—**In this paper, we present a watermarking scheme in which the watermark pattern depends on the content of the image into which it is embedded. This scheme is aimed at preventing operations in which an attacker copies a legitimate watermark and embedded into another image. The content dependency is achieved by modifying the watermark payload with the host image hash. The image hash is constructed using local Gray Level Co-occurrence Matrix (GLCM) texture features. The watermarking scheme used in this paper is the Direct-Sequence Code Division Multiple Access scheme. Our experiments show that the proposed content-dependent watermarking scheme can resist watermark copying, giving watermark BER of 49.25% when a watermark pattern copied onto another image is detected using a correct key.**

Keywords—image authentication; digital image hashing; content-dependent image watermarking; Gray Level Co-occurrence Matrix;

I. INTRODUCTION

The use of digital media to distribute images have been expanding widely since it offers a lot of advantages, including the ease with which the images can be duplicated or edited without significant loss of quality. However, these advantages also create possible abuses of digital media, for example illegal copying of images and distribution of fake (edited) images. Digital watermarking is developed to combat such abuses [1]. By embedding a watermark into an image, one can assert his/her ownership of an image or authenticate an image.

However, simply embedding a digital watermark in an image is not enough to combat the following attack scenario, known as the copy attack [2]. Let I_w be a watermarked image. An attacker does not aim to remove a watermark from I_w (or render the watermark unreadable) but instead tries to estimate the embedded watermark, W. Then the attacker embeds W into another image, X, resulting in a watermarked image X_w. The attacker does not need to know the exact watermarking scheme used or the original secret key used. When X_w is passed through a watermark detector and the correct secret key is used, the watermark W will be detected. The attacker can thus claim that image X_w also belongs to the owner of I_w since the owner's watermark is present. Such an attack can lead to fraud or slander.

The attack described above works in most watermarking schemes since the watermark content does not depend on the content of the watermarked image. Therefore, even if I_w and X_w have completely different contents, the watermark can still be correctly detected. In order to combat this attack, we have to make the embedded watermark W content-dependent. Some content-dependent watermarking systems have been proposed in the literature. For example, the author in [3] proposes the use of the relationship between two DCT AC coefficients within an image block as the image hash. The authors in [4] propose a method of constructing an N-bit image hash from projection of the input image on N smoothed random patterns, each generated using a secret key.

In this paper, we present a content-dependent watermarking system in which content dependency is achieved by performing an XOR operation between the watermark payload and the host image hash. The hash is generated based on the local Gray-Level Co-occurrence Matrix (GLCM) texture features. The rest of the paper is organized as follows. In Section II, we present a brief overview of the proposed system. In Section III, we discuss how we generate image hash from GLCM texture features. In Section IV, we discuss the watermarking scheme used in this paper. In Section V we present our experiment setup and results. Finally, in Section VI we present our conclusions.

II. OVERVIEW OF THE PROPOSED SYSTEM

In this Section, we present an overview of the proposed system. The watermark embedding process is as follows. First a hash is generated from the host image. An XOR operation is then performed between this hash and the watermark payload. This modified payload, along with a secret key, are then used to generate a watermark pattern. This pattern is then added to the host image to generate a watermarked image. This process is presented in Figure 1.

The watermark detection process is presented in Figure 2. A hash is generated from a (possibly) watermarked image. This image is also processed using a correlation detector, which will decode the embedded watermark based on the pattern generated using the same key as used in the embedding process. An XOR operation must then be performed between the output of the correlation detector and the image hash, in order to recover the original watermark payload.

978-1-4673-5817-0/13/$31.00 ©2013 IEEE

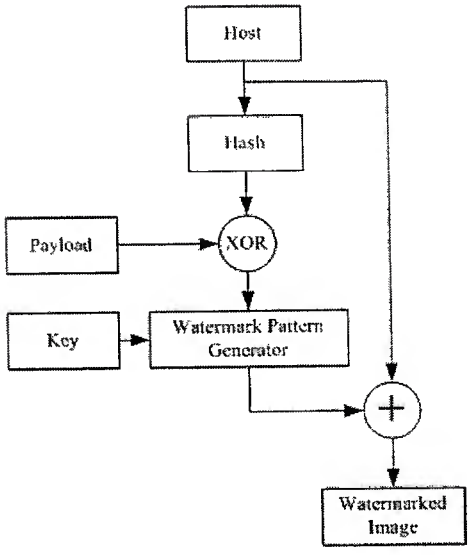

Figure 1. Block diagram of the watermark embedder

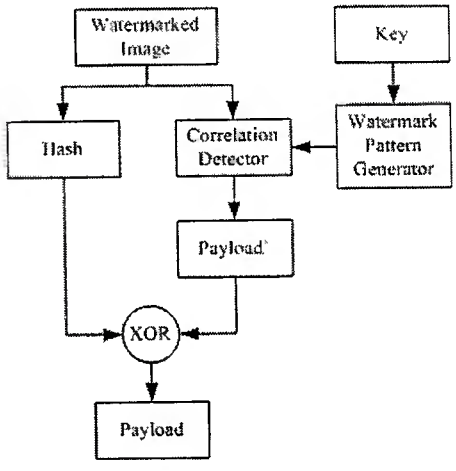

Figure 2. Block diagram of the watermark detector

III. Generating Image Hash using Gray Level Co-occurrence Matrix (GLCM) Texture Features

A. *Gray Level Co-occurrence Matrix (GLCM) Texture Features*

Gray Level Co-occurrence Matrix (GLCM) texture features are statistical based features calculated from the gray level co-occurrence matrix. This matrix is a tabulation of how often different combinations of the pixel intensity value occur in an image [5]. GLCM have been used in image watermarking applications [6][7]. However, in these works the aim of using GLCM is to improve the transparency (visual quality) and robustness of the watermark against signal processing attacks (filtering, noise, compression, etc.) and geometric attacks. The authors do not utilize GLCM to obtain content dependency of the watermark to combat copy attack.

The GLCM is built based on the relation between a reference pixel and its neighbor pixel. The orientation along these two pixels of interest is denoted by η and the distance between two pixels is denoted by d. If the parameters $\eta = 0°$ and $d = 1$ is used (as we use in this paper), the neighbor pixel is the pixel exactly on the right of a reference pixel. The four statistical values of a normalize co-occurrence matrix $P_{\eta,d}$ used as the texture feature are stated as follows [8]:

$$\text{Contrast} = \sum_i \sum_j |i-j|^{\nu} P_{\eta,d}^{\lambda}(i,j) \qquad \nu = 2, \lambda = 1 \tag{1}$$

$$\text{Energy} = \sum_i \sum_j P_{\eta,d}^2(i,j) \tag{2}$$

$$\text{Entropy} = -\sum_i \sum_j P_{\eta,d}(i,j) \log_2 P_{\eta,d}(i,j) \tag{3}$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{P_{\eta,d}(i,j)}{1+|i-j|} \tag{4}$$

B. *Generating the Image Hash*

The image hash string from an image I, H_I, is generated using the following steps:

- Normalize the image I, such that it has a mean pixel value of 128.
- Divide I into N non-overlapping blocks.
- Compute the GLCM texture features of each block. Let f_{kl}, be the k^{th} texture feature of the l^{th} image block where $k = 1, 2, 3, 4$ and $l = 1, 2, \ldots, N$.
- Generate the image hash bits for each block according to (5). In this equation, h_{kl} is the k^{th} hash bit of the l^{th} image block. An exception to this rule is that when $l = N$, then the comparison is performed between f_{kN} and f_{k1} instead.

$$h_{kl} = \begin{cases} 1, & f_{kl} > f_{k(l+1)} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

- Concatenate the hash bits from all blocks into a vector containing $4N$ bits.

IV. Direct-Sequence Code Division Multiple Access Watermarking Scheme

The watermark-embedding and detection scheme in the proposed content-dependent watermarking system uses the Direct-Sequence Code Division Multiple Access (DS-CDMA) watermarking scheme. This system is widely used in [9][10] due to the robustness of the scheme against common signal processing attacks such as filtering. It should be noted that other watermarking schemes can also be used. The DS-CDMA

scheme embeds the watermark in the Discrete Cosine Transform (DCT) domain of the host image, although the embedding process is performed in the spatial domain. The watermark is embedded in the whole region of the host image sized $m \times n$ pixels. The 2-dimensional watermark is created using the following steps [9]:

- Generate L 1-dimensional binary pseudo-random sequences, $\mathbf{s}_i$, $i = 1, 2, \ldots, L$, using a secret key, K. The variable L is the number of bits in the watermark payload. Each sequence has zero mean and contains the numbers -1 and 1. The length of each sequence is $(m \times n)/L$.
- Generate a 1-dimensional Direct Sequence spread spectrum watermark pattern $\mathbf{W}_1$ by modulating the watermark payload using the sequences created in the previous step, according to (6). In (6), p_i is the i^{th} bit of the watermark payload.

$$\mathbf{W}_1 = \sum_{i=1}^{L} (2p_i - 1)\mathbf{s}_i \tag{6}$$

- Convert $\mathbf{W}_1$ into a 2-dimensional watermark, $\mathbf{W}_2$, in using a zigzag scan order similar to that used in JPEG.
- Generate the spatial watermark pattern, w, by performing inverse DCT operation to $\mathbf{W}_2$.

The spatial watermark pattern is then embedded additively to the host image, i.e.,

$$I_w = I + \alpha w \tag{7}$$

In (7), I_w represents the watermarked image, I represents the host image and α represents the embedding strength of the watermark which is determined experimentally.

The detection process of the watermark is as follows. Given a (possibly) watermarked image, $\hat{I}_w$, the watermark is then decoded using the following steps [9]:

- Using the same secret key, K, regenerate the pseudo-random sequences $\mathbf{s}_i$, $i = 1, 2, \ldots, L$.
- Apply a forward DCT transform on the input image, $\hat{I}_w$.
- Convert the DCT matrix into a 1-dimensional vector, $\mathbf{c}_w$, using inverse zigzag scanning.
- Decode the embedded watermark, one bit at a time, by computing the correlation between $\mathbf{s}_i$ and $\mathbf{c}_w$, i.e.,

$$\hat{p}_i = \begin{cases} 1, & \text{corr}(\mathbf{c}_w, \mathbf{s}_i) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

In (8), corr($\mathbf{c}_w$, $\mathbf{s}_i$) represents the correlation between $\mathbf{c}_w$ and $\mathbf{s}_i$.

V. Experiment and Results

This section details our experiment setup and results. Our experiment is performed on a data set containing 30 8-bit grayscale images with a resolution of 600 × 600 pixels. Examples of the images from the data set are presented in Figure 3. This section is divided into two main parts. The first part discusses the experiments we performed to evaluate the performance of the image hash. The second part discusses the evaluation of the complete system.

(a) (b)

Figure 3. Examples of images in the data set

A. *Performance evaluation of the image hash*

The desired properties of the image hash are good discriminating power (i.e., the hash of two images with different contents should also be different) and robustness under operations that do not significantly alter the image contents. The discriminating power of the hash is measured by calculating the minimum, maximum and mean similarity values of the hash between different images in the test set. The robustness of the hash is measured by calculating the minimum, maximum and mean similarity values of the hash of an image after it undergoes a luminance adjustment by ± 50 graylevels and a JPEG compression with Q = 80.

The similarity value between two image hash strings H_I and H_J is calculated according to the following formula:

$$\zeta_H = 1 - (n_e / n_b) \tag{9}$$

In (9), n_b is the total number of the hash bits of each hash strings (in this paper we use n_b = 400) and n_e is the number of bits in H_I that do not match the corresponding bits in H_J. Two perfectly matched hashes will give ζ_H = 1. The average value of two randomly generated hashes will give ζ_H = 0.5.

When we compare the similarity values of the image hash between the images in the data set, we achieved a minimum similarity value of 0.38, mean similarity value of 0.51 and a maximum similarity value of 0.63. This result shows that the similarity values of the image hash between different images are low. The results of the experiments to test the robustness of the image hash are presented in Table I.

We can see from this table that the image hash generally shows good robustness, in particular against JPEG compression which does not significantly alter image contents. Some

mages, in particular those that are very dark or very bright, do ıot perform well in this test when we perform luminance ıdjustments, due to luminance value clipping.

TABLE I. IMAGE HASH ROBUSTNESS UNDER VARIOUS TEST CONDITIONS

Test Condition	Average Hash Similarity Value		
	Min	*Mean*	*Max*
Luminance adjustment (-50)	0.70	0.90	0.97
Luminance adjustment (+50)	0.63	0.86	0.99
JPEG compression (Q = 80)	0.81	0.91	0.98

3. *Performance evaluation of the complete system*

In this paper, the watermark payload embedded to each mage in the data set is 400 bits long. To evaluate the ıerformance of the whole system, we measure the average 'SNR of the watermarked images and the average bit error rate f the detected watermark. The watermark bit error rate, β, is alculated using the following formula:

$$\beta = \frac{w_e}{w_n} \times 100\% \quad (10)$$

ı (10), w_n is the total number of watermark bits and w_e is the umber of incorrectly detected watermark bits. The watermark ; detected under the following conditions:

- The watermarked images are not attacked and the correct secret key is used.
- The watermarked images are not attacked but an incorrect secret key is used.
- The watermarked images are compressed using JPEG compression with quality factor 80.
- The watermark is copied from one image (in this case we use the image shown in Figure 3(a)) and then embedded into different images (i.e., the rest of the images in the data set).

To copy the watermark from one image and embed it to ıother image, we use the following formula:

$$W' = I_w - I \quad (11)$$

(11), I is an unwatermarked image, I_w is the watermarked ːrsion of I and W' is the extracted watermark. It should be ɔted that (11) does not represent a true implementation of the ɔpy attack as proposed in [2]. Furthermore, this attack is ılikely to happen in real world scenario. Instead, this attack ın be seen as a worst-case scenario in which an attacker can timate W' from the watermarked image almost perfectly.

Our experiments show that the average PSNR of the watermarked images is 46.48 dB, which means that the watermark is visually invisible. The average BER of the detected watermark under various test conditions are presented in Table II. From this table, we can see that the watermarking system shows good robustness. If the watermark is copied from one image to another image, the average BER approaches 50%. This result is almost equivalent to the case of a randomly generated watermark payload.

TABLE II. WATERMARK BER UNDER VARIOUS TEST CONDITIONS

Test Condition	Average BER (%)
No attack	8.30
JPEG compression (Q = 80)	12.26
Watermark copying	49.25

VI. CONCLUSIONS

In this paper we have presented a content-dependent watermarking scheme using GLCM texture features. Our experiments show that the hash generated from GLCM features gives good discriminating power and robustness. The content-dependency introduced to the watermarking scheme is also shown to give good resistance against watermark copying. In the future, we aim to improve the robustness of the system against a wider range of attacks.

REFERENCES

[1] G.C. Langelaar, I. Setyawan, and R.L. Lagendijk, "Watermarking digital image and video data: A state-of-the-art overview," IEEE Signal Processing Magazine, vol. 17, no. 5, pp. 20 – 46, 2000.

[2] M. Kutter, S. Voloshynovskiy, and A. Herrigel, "The watermark copy attack," Proc. SPIE, Security and Watermarking of Multimedia Contents II, Vol. 3971, pp. 371 – 379, 2000.

[3] C.-S. Lu, "Towards robust image watermarking: combining content-dependent key, moment normalization and side-informed embedding," Signal Processing: Image Communication, no. 20, pp. 129 – 150, 2005.

[4] J. Fridrich, and M. Goljean, "Robust Hash Functions for Digital Watermarking," Proc. ITCC 2000, pp. 173 – 178, 2000.

[5] R. M. Haralick, K. Shanmugan, and I. Dinstein, "Texture features for image classification," IEEE Trans. Systems, Man, and Cybernetics, vol. SMC-3, no. 6, pp. 610 – 621, Nov. 1973.

[6] S. Kamble, S. Agarwal, V.K. Shrivastava and V. Maheshkar, "DCT based texture watermarking using GLCM," Proc. IEEE 2nd International Advance Computing Conference (IACC), vol. 19, pp. 185 – 189, 2010

[7] L. McLauchlan and M. Mehrübeoğlu, "GLCM and neural network based watermark identification," Proc. SPIE Mathematics of Data/Image Pattern Recognition, Compression and Encryption with Applications XI, vol. 7075, 70750A, 2008

[8] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed., Prentice Hall: New Jersey, 2010.

[9] P. Dong, J. G. Brankov, N. P. Galatsanos, Y. Yang, and F. Davoine, "Digital Watermarking Robust to Geometric Distortions," IEEE Trans. Image Processing, vol. 14, no. 12, pp. 2140 – 2150, 2005.

[10] P. Tzouveli, K. Ntalianis, and S. Kollias, "Confronting the synchronization problem of semantic region under geometric attacks," Proc. ICME 2006, pp. 1345 – 1348, 2006.