Postprint: Image Watermarking Algorithm Combining Texture Complexity and JND Model

Authors: Li Shuzhi, Long Xiangyu, Deng Xiaohong, Zhou Yongxin

Date: 2018-05-20T00:00:00+00:00

Abstract

To address the issue of limited capacity in existing watermarking algorithms based on the Gray Level Co-occurrence Matrix (GLCM), this paper proposes an image watermarking algorithm that integrates texture complexity with a Just Noticeable Difference (JND) model in the DCT domain. The original image is first partitioned into sub-blocks. The texture complexity of each sub-block is computed using four texture features derived from its GLCM, and the sub-blocks are subsequently sorted to determine the watermark embedding locations. The DCT transform is then applied to the original image pixel matrix, and JND values for each block are calculated using a novel partitioning approach. The watermark embedding scheme within each sub-block is determined based on the JND values and a new embedding rule. The algorithm effectively accounts for both the texture characteristics of image blocks and the sensitivity of the Human Visual System (HVS), thereby enhancing the quality of the watermarked image and increasing the watermark embedding capacity. Experimental results demonstrate that the proposed method achieves an average Peak Signal-to-Noise Ratio (PSNR) approximately 4.27% higher than existing methods when embedding watermarks of equivalent capacity. Furthermore, when embedding watermarks with twice the capacity limit of the original method, the average PSNR of the image remains at 53.4498 dB.

Full Text

Preamble

Image Watermarking Algorithm Combining Texture Complexity and JND Model

Li Shuzhi^a, Long Xiangyu^a, Deng Xiaohong^b, Zhou Yongxin^a

(^a School of Information Engineering, ^b School of Applied Science, Jiangxi University of Science & Technology, Ganzhou, Jiangxi 341000, China)

Abstract: To address the limited capacity of existing watermarking algorithms based on gray-level co-occurrence matrix (GLCM), this paper proposes an image watermarking algorithm that integrates texture complexity with a DCT-domain just noticeable difference (JND) model. The original image is first divided into sub-blocks, and the texture complexity of each sub-block is calculated using four texture features derived from its GLCM. The sub-blocks are then sorted according to their texture complexity to determine the embedding locations, after which the original image' s pixel matrix undergoes DCT transformation. Combined with a novel partitioning scheme, the JND value of each block is computed, and the watermark embedding method within each sub-block is determined based on the JND value and new embedding rules. The algorithm effectively considers both the texture characteristics of image blocks and human visual sensitivity, enhancing the quality of watermarked images and increasing watermark embedding capacity. Experimental results demonstrate that, when embedding watermarks of the same capacity, the proposed method achieves an average peak signal-to-noise ratio (PSNR) approximately 4.27% higher than existing methods. Even when embedding a watermark twice the capacity limit of the original method, the average PSNR of the resulting image remains at 53.4498 dB.

Keywords: just noticeable difference (JND); gray-level co-occurrence matrix (GLCM); texture complexity; digital watermarking

0 Introduction

Texture feature analysis was initially applied in remote sensing image interpretation. Since texture features represent quantified image characteristics derived from computational analysis, they provide effective macroscopic and microscopic information about images, making texture analysis a crucial research approach in image processing and digital watermarking. Common image texture analysis methods include statistical analysis, structural analysis, model-based analysis, and spectral analysis. Among these, the gray-level co-occurrence matrix (GLCM) proposed by Haralick represents one of the most effective statistical analysis methods.

Zhang et al. [4] introduced a DCT-based binary sequence grouping watermarking algorithm that embeds multiple bits of watermark information using a single DCT coefficient, reducing the number of coefficients requiring modification during embedding and thereby increasing capacity while enabling random selection of embedding locations and robustness against typical attacks. Shen et al. [5] proposed a passive splicing detection method called TF-GLCM, which applies discrete cosine transform to different blocks before computing GLCM to fully capture spatial relationships between texture information and image pixels, thereby reducing feature vector dimensionality and computational complexity.

Li et al. [6] utilized four GLCM-derived feature parameters to construct a math-

ematical model for calculating image complexity to guide watermark embedding. While this approach improved the PSNR of watermarked images to some extent, its embedding rules limited watermark capacity, and concentrating watermarks in the 128$×$128 most complex texture blocks affected image quality.

Meanwhile, JND models based on DWT or DCT domains have found widespread application in digital image watermarking. Li et al. [8] proposed a comprehensive DCT and DWT watermarking model that considers human visual masking characteristics. Just noticeable difference (JND) describes the maximum image distortion imperceptible to the human eye, accounting for the fact that the visual system can tolerate certain modifications without detection. In image processing, JND can be used to assess human sensitivity to distortion in different image regions. Various JND models have been applied to image processing in recent years. Hsu et al. [9] combined back-propagation neural networks with JND models, where watermark coefficient adjustment was influenced by both JND and BPNN prediction, achieving robustness against various image processing attacks with excellent imperceptibility for effective blind watermarking. Zheng et al. [10] improved transform-domain JND accuracy by considering texture component filtering when calculating contrast masking factors to address JND underestimation. Wu et al. [11] further considered human sensitivity differences toward regular and irregular regions when calculating texture masking, proposing a JND model based on luminance adaptation and structural similarity. Hu et al. [12] enhanced watermark imperceptibility by using JND to regulate parameter variation amplitude in a DCT-based blind watermarking scheme that embeds binary information, significantly improving robustness against common attacks. Ye [13] proposed a DWT-domain blind digital watermarking algorithm based on human visual system (HVS) characteristics, utilizing JND to improve watermark imperceptibility.

In summary, while GLCM-based texture analysis and transform-domain JND models each offer broad application prospects in watermarking, no existing method combines GLCM with DCT-domain JND models to compensate for their individual limitations. This paper proposes a novel watermarking approach that integrates both techniques, optimizing the JND model to expand embedding capacity and improve watermarked image quality.

---

1 Texture Complexity and JND Measurement

1.1 GLCM Complexity Calculation

The gray-level co-occurrence matrix represents a matrix function of pixel distance and angle in an image. In GLCM, the distance between image pixels is defined as d = (dx, dy) with orientation {0°, 45°, 90°, 135°}, corresponding to d {(0, d), (d, d), (d, 0), (-d, d)}. Let g(i, j, d,) denote the GLCM. Four primary feature parameters are extracted from the GLCM:

a) Energy J. Energy reflects the uniformity of image distribution and texture coarseness:

$$J = \sum_{i=1}^{k} \sum_{j=1}^{k} g(i, j, d, \theta)^2$$

b) Entropy H. Entropy describes the information content of the image:

$$H = -\sum_{i=1}^{k} \sum_{j=1}^{k} g(i, j, d, \theta) \log g(i, j, d, \theta)$$

c) Contrast D. Contrast describes image clarity, i.e., texture sharpness:

$$D = \sum_{i=1}^{k} \sum_{j=1}^{k} (i - j)^2 g(i, j, d, \theta)$$

d) Correlation COV. Correlation describes the degree of local texture variation:

$$COV = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} ij \cdot g(i, j, d, \theta) - \mu_1 \mu_2}{\sigma_1 \sigma_2}$$

By assigning weights to the four feature parameters based on mean square error, the texture complexity can be calculated as:

$$f = \omega_1 \times J + \omega_2 \times H + \omega_3 \times D + \omega_4 \times COV$$

where J_1, J_2, J_3, J_4 represent the energy in four GLCM orientations, and $_1$, $_2$, $_3$, $_4$ are weights assigned to the same feature parameter in different directions based on mean square error. According to equation (9), a $512\$\times 512 standard grayscale image Lena was divided into 32\times\$32$ blocks to compute the complexity of each block, with experimental results shown in Table 1 .

After obtaining the complexity ranking of all blocks, this information is combined with an improved DCT-domain JND model to determine watermark embedding locations and sequence.

1.2 DCT-Domain JND Calculation Combined with Texture Characteristics

The DCT-domain JND model [14] is described as the product of three components: spatial contrast sensitivity function, luminance adaptation factor, and contrast masking factor:

$$JND(i, j) = J_{CSF}(i, j) \times A_{lum}(i, j) \times F_{contrast}(i, j)$$

where J_{CSF}(i, j) is the spatial contrast sensitivity function:

$$J_{CSF}(i,j) = a \times \exp\left(-\frac{b \cdot \omega_{ij}}{c + s \cdot \omega_{ij}}\right) \times \cos\left(\frac{\pi}{4} + \varphi_{ij}\right)$$

Here, s represents the set effect (s = 0.25 in this paper), r is an empirical value (r = 0.6), *ω{ij} is the DCT normalized coefficient, and* ϕ{ij} represents the orientation angle of the corresponding DCT coefficient.

A_{lum} is the luminance adaptation factor, as the human eye exhibits different sensitivity to regions with varying brightness. The relationship between A_{lum} and the average brightness of an image region is defined as:

$$A_{lum} = \begin{cases} 1 + \frac{60-\bar{I}}{150}, & \bar{I} \leq 60 \\ 1, & 60 < \bar{I} < 170 \\ 1 + \frac{\bar{I}-170}{425}, & \bar{I} \geq 170 \end{cases}$$

F_{contrast}(i,j) is the contrast masking factor. Considering the self-masking effect of subband coefficients, the final relationship is:

$$F_{contrast}(i,j) = \begin{cases} 0.36 \times \psi \times \min\left(4, \frac{J_{CSF}(i,j)}{A_{lum}}\right), & \text{if } \frac{J_{CSF}(i,j)}{A_{lum}} \leq 16 \\ \psi \times \left(\frac{J_{CSF}(i,j)}{A_{lum}}\right)^{0.25}, & \text{otherwise} \end{cases}$$

The value of differs across image texture regions: for smooth and edge regions where human perception is more sensitive, = 1; for texture regions where sensitivity to low-frequency coefficients is relatively lower, = 2.25 for low frequencies and = 1.25 for high frequencies.

For an image, when GLCM elements are concentrated, the J value is larger, indicating relatively uniform and regular texture patterns; conversely, it is smaller. When GLCM values are uniformly distributed (i.e., the image is nearly random or highly noisy), entropy is lower; otherwise, it is higher. Smooth regions represent relatively uniform and regular texture patterns with less information content than non-smooth regions. Therefore, this paper distinguishes smooth and non-smooth regions using entropy and energy values.

First, compute the smoothness of the entire image:

$$E = \partial_1 \times \partial_2 \times \sum_{i=1}^{B} (J_i + H_i)$$

where $_1$, $_2$ are weights corresponding to energy and entropy in the complexity calculation, B represents the number of image blocks (e.g., B = 4096 for a

512$\times$512 $image divided into 8\times8 sub-blocks), and J_i is the average energy of block i. Thus, the smoothness of an 8\times$ sub-block is:

$$e_i = \partial_1 \times \partial_2 \times (J_i + H_i)$$

The classification rule is:

$$\text{True (smooth region)} \quad \text{if } e_i > E$$

$$\text{False (non-smooth region)} \quad \text{otherwise}$$

Next, classify non-smooth regions into texture and edge regions based on contrast and correlation. First, compute the texture degree of the entire image:

$$\delta = \partial_3 \times \partial_4 \times \sum_{i=1}^{B} (D_i + COV_i)$$

where $_3$, $_4$ are weights for contrast and correlation in the complexity calculation. Then compute the texture degree of each 8$\times$$8 sub-block:

$$\delta_i = \partial_3 \times \partial_4 \times (D_i + COV_i)$$

The classification rule is:

$$\text{True (texture region)} \quad \text{if } \delta_i > \delta$$

$$\text{False (edge region)} \quad \text{otherwise}$$

The algorithm for the GLCM-based improved DCT-domain JND model is presented as Algorithm 1.

Algorithm 1: GLCM-Based Improved DCT-Domain JND Algorithm - **Input:** I (original image) - **Output:** M (JND value matrix of the image) 1. read(I); // Read original image 2. compute J_{CSF}(i,j); // Calculate using formulas from Section 1.2 3. compute A_{lum}; // Calculate using formulas from Section 1.2 4. num = 512$\times$512/8$\times$8; *//Divide image into num 8\times$8 blocks* 5. for each block B in { } // Process each sub-block 6. compute J, H, D, COV; // Region type determination 7. if e_i > E; // Smooth region 8. = 1; 9. else if *$\delta i > \delta$; // Edge region 10. = 1; 11. else if (h,l) where h,l are row/column positions in sub-block 12. if (h^2 + l^2) < 22; // Texture region low frequency 13. = 2.25; 14. else; // Texture region high frequency 15. = 1.25; 16. end if 17. end if 18. end for 19. dct2(I); // Perform DCT transform on original pixel matrix 20. compute F*{contrast}(i,j); // Calculate using formulas from Section 1.2 21. M = J_{CSF}(i,j) $\times$ A_{lum} $\times$ F_{contrast}(i,j); 22. Return M

2 Image Watermarking Algorithm Combining Texture Complexity and JND Model

2.1 Watermark Embedding Process

The watermark image pixels are first converted from decimal to binary format (e.g., 0 → 00000000, 255 → 11111111). For a watermark image of size $M_1 \times N_1$, the resulting binary code length is $M_1 \times N_1 \times 8$, stored in a key array Key[3] = [8, M_1, N_1]. The detailed embedding process proceeds as follows:

a) If the original image size is $M_2 \times N_2$, divide it into $32\$\times 32 blocks, yieldingn = (M\_\{2\}\$ \times N_2)/(32 \times 32)$ blocks. For each block, compute J, H, D, and COV using equations (1)-(4), then determine its texture complexity f_1, f_2, ···, f using equation (9).

b) Sort the computed texture complexities in descending order, prioritizing watermark embedding in blocks with higher complexity.

c) Divide each $32\$\times 32 blockintosixteen 8\times 8 sub-blocks, thencomputetheJNDvalueforeachsub-blockthroughthefollowingprocess: CalculateJ, H, D, andCOVforeach 8\times 8 sub-blocktoclassifytheimageintosmooth, edge, andtextureregions. PerformDCTtransformationontheorig$ DCT block using equation (10).

d) Embed watermark binary codes in descending order of image texture complexity according to the following rule:

$$\mathrm{DCT}(i,j) = \begin{cases} \mathrm{DCT}(i,j) + \mathrm{JND}(i,j), & \text{if current watermark bit} = 1 \\ \mathrm{DCT}(i,j) - \mathrm{JND}(i,j), & \text{if current watermark bit} = 0 \\ \text{No embedding}, & \text{if } \mathrm{JND}(i,j) = 0 \end{cases}$$

According to this rule, only 7 positions in each 8$×$8 block are used for embedding, allowing $28 \times 16 = 448$ bits per $32\$\times 32 block. A 32\times 32 watermarkimagecontains 32\times 32\times\$8$ binary codes, requiring 19 such blocks for embedding. Therefore, watermark binary codes are sequentially embedded in designated positions within the top 19 most complex blocks.

e) Perform IDCT transformation on the watermarked binary-coded image matrix and convert data types to obtain the final watermarked image. The watermark embedding algorithm is presented as Algorithm 2.

Algorithm 2: Watermark Embedding Algorithm Combining Texture Complexity and JND Model - **Input:** 1) I (original image); 2) SY (watermark image) - **Output:** F (watermarked image pixel matrix) 1. T_1 = GLCM(I); // Obtain 4-direction GLCM from original matrix 2. compute f; // Calculate texture complexity ranking per Section 1.1 3. compute JND(i,j); // Calculate JND values using Algorithm 1 4. SY_1[n] = dec2bin(SY); // Convert watermark pixels to binary sequentially, store in 1D array SY_1[n] 5. for num = 1, 2, ···, n // Embed n watermark bits sequentially 6. // Embed in order of texture complexity 7. Cut into temp[8][8]; // Split

each block into 8$×8$sub - blockssequentially8.forj = 1, 2, ..., 89.forj = 1, 2, ..., 8//Processeachpositioninsub - block10.ifJND(i, j) = 0, j + +; //Skipembedding11.elseifSY_{1}$[num] = 1; 12. DCT(i,j) = DCT(i,j) + JND(i,j); 13. else 14. DCT(i,j) = DCT(i,j) - JND(i,j); 15. end if 16. end for // Next column 17. end for // Next row 18. end for // Next sub-block 19. end for // All watermark bits embedded 20. F = idct2(DCT); // Perform inverse DCT on watermarked DCT matrix 21. Return F

2.2 Watermark Extraction Process

a) Obtain the watermark image size and required embedding positions ($8 \times M_1 \times N_1$) from key Key[3]. Compute the original image' s texture complexity and JND values to determine each watermark bit' s embedding location based on the required number of positions, 32$×32blockcomplexityranking, andallowableembeddingpositionswithin8×$8 sub-blocks (where JND $\neq 0$).

b) After locating the positions, extract the embedded information using the following rule:

$$\text{Extracted bit} = \begin{cases} 1, & \text{if } \mathrm{DCT}_1(i,j) - \mathrm{DCT}_2(i,j) = \mathrm{JND}(i,j) \\ 0, & \text{if } \mathrm{DCT}_1(i,j) - \mathrm{DCT}_2(i,j) = -\mathrm{JND}(i,j) \end{cases}$$

where DCT_1(i,j) and DCT_2(i,j) represent DCT coefficients before and after watermarking, respectively.

c) After extracting the watermark's binary code, convert it to decimal format and arrange sequentially in an M×N matrix to reconstruct the watermark image. The watermark extraction process is illustrated in Figure 1 [Figure 1: see original paper].

---

3 Experimental Results and Analysis

Experimental Platform: The experiments were conducted on Windows 7 Ultimate with an Intel Core i5-7200U processor (2.6 GHz) and 4 GB RAM. Simulation software included MATLAB and Code::Blocks. Four standard 512$×$512 grayscale images (Lena, Barbara, Lake, Boat) in .bmp format were used for testing.

3.1 Comparison of Watermarked Images

Figure 4 Figure 4: see original paper shows the original image, while (b) and (c) display images embedded with 8,192-bit and 32,768-bit watermarks, respectively. Visually, the image quality shows almost no degradation after embedding watermarks at both capacities, demonstrating good watermark imperceptibility.

Detailed performance parameters for the four test images are presented in Table 2 .

Table 2: PSNR Comparison Between Proposed Algorithm and Reference [6]

Carrier Image	Watermark Capacity (bits)	PSNR (dB)	Improvement over [6]
Barbara	8,192	64.1370	+4.43%
Barbara	16,384	60.9840	+4.11%
Average	32,768	53.4498	-

As shown in Table 2, when embedding an 8,192-bit watermark, the proposed method achieves an average PSNR of 64.1370 dB, representing a 4.43% improvement over reference [6]. At 16,384 bits, the average PSNR is 60.9840 dB (4.11% improvement). Even at 32,768 bits (double the capacity limit of the original method), the average PSNR remains 53.4498 dB, significantly reducing image distortion.

To more intuitively demonstrate the superiority of the proposed method that jointly exploits texture characteristics and DCT-domain JND values, line charts comparing our approach with reference [6]' s texture-only method are provided for all four test images across various watermark capacities in Figures 3 [Figure 3: see original paper] through 6 [Figure 6: see original paper].

3.2 Comparison of Extracted Watermarks

Figure 7 [Figure 7: see original paper] shows watermarks extracted by both reference [6] and the proposed method (8,192-bit capacity). The watermark extracted by our algorithm is significantly clearer than that from reference [6]. Reference [6] achieved a watermark similarity of 91.79% with a PSNR of 37.7097 dB, while our method attained 97.55% similarity with a PSNR of 49.2851 dB, demonstrating that the improved algorithm more accurately extracts watermark information at small capacities.

The embedding rules in reference [6] limit the watermark capacity to an upper bound of 16,384 bits. In contrast, our method can embed 32,768 bits while maintaining a watermark similarity of 87.21% and a PSNR of 15.7470 dB, as shown in Figure 8 [Figure 8: see original paper]. Although the PSNR is lower at this high capacity due to reduced accuracy in less complex blocks, the watermark content remains visually identifiable.

4 Conclusion

This paper effectively combines image texture characteristics with DCT-domain JND values to maximize pixel modifications within the JND threshold while utilizing texture complexity. This approach ensures watermarked image quality and extraction accuracy while significantly increasing embedding capacity. Future research will focus on optimizing JND threshold calculation formulas and developing superior integration models of texture characteristics and JND to improve extraction accuracy at high capacities.

References

[1] Liu Fengzhu, Zhang Jingxiong, Lin Zongjian, et al. Measurement methods for gray and texture information in multispectral remote sensing images [J]. Geomatics and Information Science of Wuhan University, 2016, 41(3): 415-420.

[2] Huang Xiang, Yang Wunian. Remote sensing image classification combining gray and texture features based on dynamic windows [J]. Journal of Geomatics Science and Technology, 2015, 32(3): 277-281.

[3] Haralick R M, Shanmugam K, Dinstein I. Textural features for image classification [J]. IEEE Transactions on Systems, Man, and Cybernetics, 1973, SMC-3(6): 610-621.

[4] Zhang Hao, Yang Yongjian, Han Hongying. A DCT-based sequence grouping watermarking algorithm [J]. Application Research of Computers, 2014, 31(11): 3473-3476.

[5] Shen X, Shi Z, Chen H. Splicing image forgery detection using textural features based on the grey level co-occurrence matrices [J]. IET Image Processing, 2017, 11(1): 44-53.

[6] Li Shuzhi, Hu Qin, Deng Xiaohong. Reversible image watermarking using GLCM texture feature block selection [J]. Journal of Optoelectronics · Laser, 2017, 28(4): 411-418.

[7] Nguyen P B, Beghdadi A, Luong M. Perceptual watermarking using a new just-noticeable-difference model [J]. Signal Processing: Image Communication, 2013, 28(10): 1506-1525.

[8] Li Zhi, Chen Xiaowei. Gray image watermarking algorithm combining wavelet and cosine transforms [J]. Journal of Image and Graphics, 2006, 11(6): 834-839.

[9] Hsu L Y, Hu H T. Blind image watermarking via exploitation of inter-block prediction and visibility threshold in DCT domain [J]. Journal of Visual Communication & Image Representation, 2015, 32(C): 130-143.

[10] Zheng Mingkui, Su Kaixiong, Wang Weixing, et al. Transform-domain JND model based on texture decomposition and image coding method [J]. Journal on Communications, 2014, 35(6): 185-191.

[11] Wu J J, Qi F, Shi G M. Self-similarity based structural regularity for just noticeable difference estimation [J]. Journal of Visual Communication and Image Representation, 2012, 23(6): 845-852.

[12] Hu H T, Chang J R, Hsu L Y. Robust blind image watermarking by modulating the mean of partly sign-altered DCT coefficients guided by human visual perception [J]. AEU-International Journal of Electronics and Communications, 2016, 70(10): 1374-1381.

[13] Ye Chuang. Research on digital watermarking algorithm based on discrete wavelet transform [D]. Hangzhou: Zhejiang University.

[14] Zhang X H, Lin W S, Xue P. Improved estimation for just noticeable visual distortion [J]. Signal Processing, 2005, 84(4): 795-808.

Note: Figure translations are in progress. See original paper for figures.