Spread-spectrum Watermark by Synthesizing Texture

Wenyu Liu, Fan Zhang, and Chunxiao Liu

Huazhong University of Science and Technology, Wuhan, 430074, P. R. China
{liuwy, zhangfan}@hust.edu.cn
liucx@smail.hust.edu.cn

Abstract. Image watermarking is a mapping from watermark message to a set of image counterparts, where every version conveys the same meaning with the original image. Since textures that present single perceptual meaning have certain diversity, an intuitive idea of watermarking is to replace the texture region of an image with a similar-looking synthetic texture containing the watermark. We propose a spread-spectrum watermarking scheme by integrating existent work on texture extraction, segmentation and synthesis, and obtain suggestive results, including (1) the synthetic watermarks can resist adaptive Wiener filtering attack due to its power spectrum similar with the original image; (2) if using the spread-spectrum carrier which is designed elaborately according to the subspace spanned by the textures, hiding capacity can be improved by 20%, while effective hiding capacity under Wiener filtering attack can be doubled; (3) the proposed watermarking scheme also enlighten a sophisticate strategy for watermark attack.

1 Introduction

Watermarking, also called steganography, is to discreetly embed information into media signals without eliciting noticeable distortion. For the applications which do not demand strict media fidelity, such as covert communication or rights management for entertainment media, *unnoticeablity* of distortion is reasonable to be ruled as no changing or confusing the perceptual meaning of the media. In this way, image watermarking is a mapping from watermark message to a set of image counterparts, where every counterpart conveys the same meaning with the original image.

In his tutorial literature on data-hiding codes [1], Moulin suggested an idea of connecting image modeling and image watermarking: *“A sophisticated embedder or attacker could replace a textured portion of an image (say a grass field) with a similar-looking synthetic texture, introducing negligible perceptual degradation.”*

To realize the idea, this paper designs a watermarking-by-synthesizing scheme, which integrates existent work of texture extracting, segmenting and synthesizing.

Recent work of Balakrishnan [8] propose a image representation method called Hybrid ICA-Mixture of PPCA Algorithm (HIMPA), which use an independent component analysis (ICA) model for edge representation followed by a mixture of probabilistic principal components analyzers (MPPCA) for surface representation. We use a variant of HIMPA for texture extraction and segmentation.

A number of texture synthetic algorithms [3] create texture by clever resampling from the original texture. Although the results are visually stunning, they do not provide an explicit model for texture and their cooperation with watermarking seem unsystematic, e.g., the method in [4], known as texture block coding, which inserts a textured patch into an area of the image with the same appearance. Founded on certain texture model, some texture synthetic algorithms [5] ~ [7] adopt synthesis-by-analysis methodology to create texture by matching statistical feature with the original texture. We model the spread-spectrum watermarking as an optimization problem with a linear object and nonlinear distortion constraints, and use the algorithm in [7] to pursue the watermarked image.

The proposed scheme is introduced in section 2. Experimental results and analysis are presented in section 3. The paper closes with concluding remarks in section 4.

2 Proposed Scheme

Our scheme is shown in Figure 1. Without loss of generality, we embed one bit of message m whose value is either –1 or +1. Generated according to secret key k, $\boldsymbol{p}$ is a pseudorandom sequence with zero mean and unit variance. Modulated by m, the sequence $\boldsymbol{p}$ yields the watermark pattern $m\boldsymbol{p}$. Pseudorandom sequence $\boldsymbol{p}$ plays the role of the spread-spectrum *carrier* for watermark m.

The host image $\boldsymbol{x}$ can be divided into texture part and nontexture part, that is

$$\boldsymbol{x} = \boldsymbol{x}_{\mathrm{T}} + \boldsymbol{x}_{\mathrm{N}} \ . \tag{1}$$

The core of the watermarking scheme is to synthesize a texture part $\boldsymbol{x}_{\mathrm{T}}'$, which has the maximal inner product with $m\boldsymbol{p}$ and meanwhile shares the same texture features with $\boldsymbol{x}_{\mathrm{T}}$, as the description of (2)

$$\boldsymbol{x}_{\mathrm{T}}' = \arg\max < \boldsymbol{x}_{\mathrm{T}}', m\boldsymbol{p} > \tag{2}$$

$$\text{s.t. } \phi_k(\boldsymbol{x}_{\mathrm{T}}') = \phi_k(\boldsymbol{x}_{\mathrm{T}}), \ k = 1 \ldots N_C \tag{3}$$

where $<\boldsymbol{A}, \boldsymbol{B}>$ denotes correlation, i.e., the inner product of sequence $\boldsymbol{A}$ and sequence $\boldsymbol{B}$. Φ_k is a set of feature functions of texture, which establish texture equivalence between $\boldsymbol{x}_{\mathrm{T}}'$ and $\boldsymbol{x}_{\mathrm{T}}$. Given $m\boldsymbol{p}$ and $\boldsymbol{x}_{\mathrm{T}}$, the problem above is an optimal program with linear object (2) and nonlinear constraints (3). After finding the optimal $\boldsymbol{x}_{\mathrm{T}}'$, the watermarked image is generated as

$$\boldsymbol{y} = \boldsymbol{x}_{\mathrm{T}}' + \boldsymbol{x}_{\mathrm{N}} \tag{4}$$

The watermarked image is possibly corrupted by an attacker's noise. We only consider the additive noise, as formula (5)

$$\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{n} \ . \tag{5}$$

The receiver knows the secret key k and can recover the $\boldsymbol{p}$, and then the detection is performed. Firstly, the (normalized) correlation is calculated as

$$r = \frac{<\boldsymbol{z},\boldsymbol{p}>}{\sigma_z \sigma_p} = \frac{<\boldsymbol{x}'_{\mathrm{T}},\boldsymbol{p}> + <\boldsymbol{x}_{\mathrm{N}},\boldsymbol{p}> + <\boldsymbol{n},\boldsymbol{p}>}{\sigma_z} , \qquad (6)$$

where σ_z^2 and σ_p^2 is respectively the variance of $\boldsymbol{z}$ and $\boldsymbol{p}$. The two latter terms of numerator in (6) can be neglected due to independency between $\boldsymbol{x}_{\mathrm{N}}$ and $\boldsymbol{p}$ and independency between $\boldsymbol{n}$ and $\boldsymbol{p}$, so the one-bit watermark is estimated by the sign of r in formula (7),

$$\hat{m} = \mathrm{sign}(r) \ . \qquad (7)$$

Absolute value of r determines the error probability of detection and thereby the effective rate of watermark. In this paper, we use *averaged normalized correlation* $|r|/L$ to measure the hiding capacity of watermarking scheme, where L is the amount of pixels.

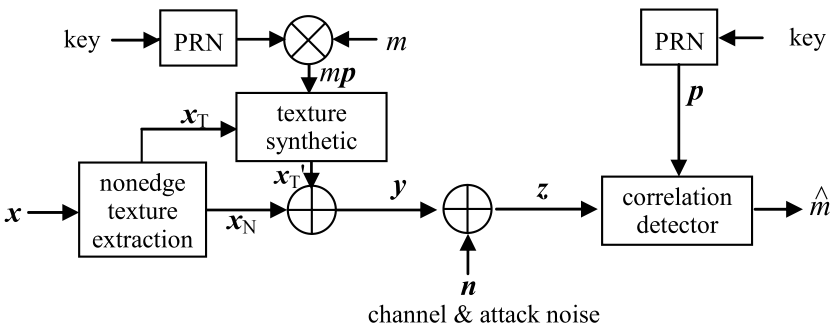


Fig. 1. The proposed watermarking scheme

How should one go about extracting the texture from natural image and synthesizing a watermarked texture? We explain the methods in following sections.

2.1 Texture extraction

For texture extraction, we propose a three-factor image model. Three factors are assumed to contribute to the appearance of nature image: (*a*) objects' intrinsic luminance and shading effects, (*b*) structural edges, and (*c*) texture. Different from HIMPA model [9], our model isolates low frequency band of image $\boldsymbol{x}_{\mathrm{L}}$ as the factor (*a*), and uses HIMPA model to divide the residual high frequency band $\boldsymbol{x}_{\mathrm{H}}$ into edge region and surface region. The reason for isolating factor (*a*) is that texture clustering will not be affected by local average luminance, and thereby will depend more on local contrast.

According to HIMPA [9], we use independent component analysis (ICA) to extract texture from the high frequency band of image. An image patch $\boldsymbol{x}_{\mathrm{H}}$ is represented as a linear superposition of columns of the mixing matrix $\boldsymbol{A}$ and a residue noise which can be neglected.

$$\boldsymbol{x}_{\mathrm{H}} = \boldsymbol{A}\boldsymbol{s} \ . \tag{8}$$

The columns of $\boldsymbol{A}$ are learned from nature images by making the components of the vector $\boldsymbol{s}$ statistically independent. In the learning algorithm, $\boldsymbol{A}$ is constrained to be an orthogonal transformation. So the vector $\boldsymbol{s}$ is calculated by

$$\boldsymbol{s} = \boldsymbol{A}^T \boldsymbol{x}_{\mathrm{H}} \ . \tag{9}$$

The elements of $\boldsymbol{s}$ with large magnitudes signify the presence of structure in an image block, corresponding to edges, oriented curves, etc. The elements of $\boldsymbol{s}$ with smaller magnitudes represent the summation of a number of weak edges to yield texture. For each image block, we divide $\boldsymbol{A}$ into two groups, $\boldsymbol{A}_{\mathrm{E}}$ and $\boldsymbol{A}_{\mathrm{T}}$, keeping all the absolute values of $\boldsymbol{s}$ corresponding to columns of $\boldsymbol{A}_{\mathrm{E}}$ are larger than $\boldsymbol{A}_{\mathrm{T}}$. The image block is also decomposed into two subcomponents.

$$\boldsymbol{x}_{\mathrm{H}} = \boldsymbol{x}_{\mathrm{E}} + \boldsymbol{x}_{\mathrm{T}} \ . \tag{10}$$

where

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}_{\mathrm{E}} (\boldsymbol{A}_{\mathrm{E}}^T \boldsymbol{A}_{\mathrm{E}})^{-1} \boldsymbol{A}_{\mathrm{E}}^T \boldsymbol{x}_{\mathrm{H}} \ . \tag{11}$$

The ratio between the column amounts of $\boldsymbol{A}_{\mathrm{T}}$ and $\boldsymbol{A}$ is determined by ICA threshold. For example, if ICA threshold is 0.2, $\boldsymbol{A}_{\mathrm{E}}$ has 128 columns while $\boldsymbol{A}_{\mathrm{T}}$ has 32 columns. Because of ratio control, texture is the residual after removing the relatively dominant edge instead of the absolutely sharp edge in HIMPA [9], and much intenser texture will be extracted from the rough region than the smooth region. Fig. 2 illustrates the process of texture extraction.

2.2 Texture segmentation

Suggested by [9], MPPCA is able to model complex textures in a real image scene due to texture clustering. The textures can be assumed to sample from a limited number of clusters. Each cluster is assumed homogenous, where a texture block from cluster k is generated using the linear model

$$\boldsymbol{x}_k = \boldsymbol{W}_k \boldsymbol{s}_k + \boldsymbol{\mu}_k + \boldsymbol{\varepsilon}_k, \quad k = 1, \ldots, K \ . \tag{12}$$

where $\boldsymbol{x}_k \in \mathbf{R}^a$ is a column vector elongated from the host image block and has a dimension of a, $\boldsymbol{s}_k \in \boldsymbol{R}^q$ is the lower dimensional subspace assumed to be Gaussian distributed with zero mean and identity covariance. Note that, in this section, image block always refers to the texture block. The dimension of $\boldsymbol{s}_k$ is q and $a > q$. $\boldsymbol{W}_k$ is a mixing matrix of a by q. $\boldsymbol{\mu}_k$ is the cluster observation mean, $\boldsymbol{\varepsilon}_k$ is Gaussian white isotropic noise with zero mean, i. e. $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(0, \sigma^2\mathbf{I})$.

$\boldsymbol{W}_k$, $\boldsymbol{s}_k$, $\boldsymbol{\mu}_k$, and $\boldsymbol{\varepsilon}_k$ are hidden variables that can be estimated from the data of observed texture. Meanwhile, posterior probability of membership $R(\boldsymbol{x} \mid k)$ can also be calculated, which measures how likely a block $\boldsymbol{x}$ belongs to cluster k and conforms to the condition $\sum_k R(\boldsymbol{x} \mid k) = 1$. Right middle of Figure 2 shows the posterior probability of three clusters using a trichrome image.

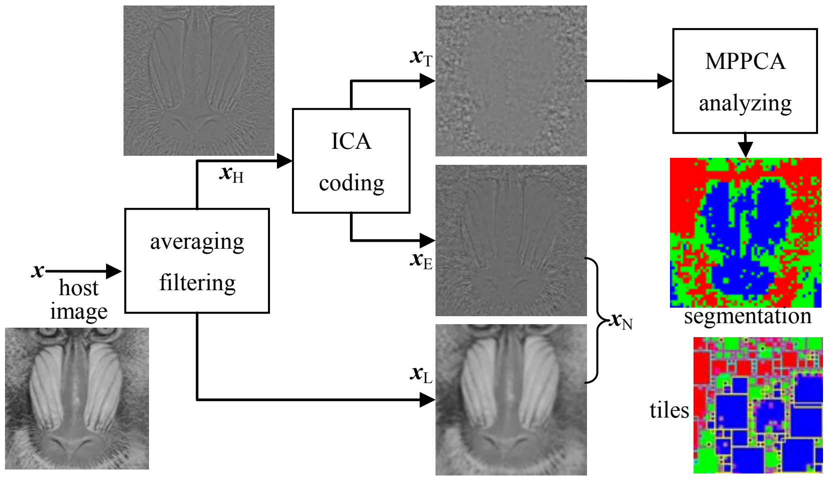


Fig. 2. Three-factor model for nonedge extraction and features learned by MPPCA

2.3 Texture Synthesis

Given a set of feature functions $\{\Phi_k\}$, and their corresponding values $\Phi_k(\boldsymbol{x}_\mathrm{T})$, the problem of synthesizing watermarked texture becomes one of looking for image $\boldsymbol{x}_\mathrm{T}'$, which has the maximal projection across watermark, from the associated texture ensemble $S_\Phi = \{\boldsymbol{x} : \Phi_k(\boldsymbol{x}) = \Phi_k(\boldsymbol{x}_\mathrm{T}),\ \forall k\}$. The complexity of the feature functions $\{\Phi_k\}$ in a realistic model of a texture makes it difficult to find the solution directly. We consider an iterative solution, in which the texture to be synthesized is initialized as the carrier $\boldsymbol{mp}$ (a random noise) and the constraints are imposed on the texture sequentially, rather than simultaneously. According to [7], we repeatedly utilize

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \sum_k \lambda_k \nabla \Phi_k(\boldsymbol{x}_t) \qquad \textbf{(13)}$$

to make the texture's features approach $\Phi_k(\boldsymbol{x}_\mathrm{T})$, where ∇ is gradient operator, and λ_k

is a scalar solved to satisfy or come to closest to satisfying $\Phi_k(\boldsymbol{x}_{t+1}) = \Phi_k(\boldsymbol{x}_T)$. And we also repeatedly use

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \varepsilon \boldsymbol{mp} \tag{14}$$

to drive $\boldsymbol{x}_{t+1}$ has a larger projection across $\boldsymbol{mp}$. ε goes to 0 as the iteration goes to the end. A rigorous theory for the algorithm convergence has yet to be found, nevertheless, the algorithm has not failed to get close to convergence for many examples tested.

We use parts of the feature functions Φ_k proposed in [7], which first decomposes the image by steerable pyramid transform [7,15] and then defines the features on the statistics of the transform coefficients and the statistics of pixel values.

The statistical operators which we selected from literature [7] include

- S_1 coarse marginal statistics. Variance, minimum and maximum of image pixels.
- S_2 fine marginal statistics. Skewness and kurtosis of image pixels and low-pass bands at each scale, and variance of the high-pass band.
- S_3 raw coefficient correlation. Auto-correlation of the low-pass bands. These characterize the salient spatial frequencies and the regularity of the structures.
- S_4 coefficient magnitude statistics. Auto-correlation of magnitude of each subband, cross-correlation of each subband magnitudes with those of other orientations at the same scale, and cross-correlation of subband magnitudes with all orientations at a coarser scale. These represent edges, bars, corners, etc.

Textures having been segmented in Section 2.2 always have irregular shape, so we use a few of rectangle tiles with size of $2^m \times 2^m$ (m = 3, 4, 5, 6) to make up the texture shape, followed by synthesizing every regular tile instead of the whole irregular texture. A greedy algorithm is used to fill irregular shape with the tiles as large as possible, so that the amount of tiles is close to minimal. (Right bottom of Figure 2 shows the tiles). During synthesis of the larger tile, we use more constraint parameters and iterate synthesis by more steps as shown in Table 1.

Table 1. Feature parameters and algorithm parameter for different tile size.

tile size 2^m	transform parameters		statistical operators				iteration steps
	scale	orientation	S_1	S_2	S_3	S_4	
64	3	4	√	√	√	√	50
32	2	4	√	√	√	×	25
16	1	4	√	√	×	×	10
8	1	4	√	×	×	×	1

3 Experimental results and analysis

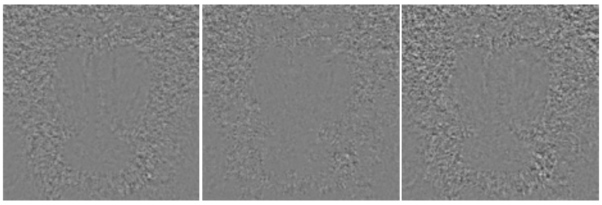
ICA mixing matrix is estimated using samples from a training set of 13 natural images downloaded with the FastICA package [12]. Note that ICA mixing matrix is independent with host images and need not be held by watermark detector.

At the watermark encoder, after 8×8 averaging filtering, texture is extracted by ICA coding, and then, partitioned into 8×8 blocks, vectorized into 64×1 vectors, and fed to MPPCA for clustering. Texture is classified into three clusters with 4 principal components in each cluster, where each cluster is divided into regular tiles and synthesized tile by tile. Finally, watermarked image is generated by summarizing the synthetic texture, the edge part and the low-pass band. The program code was developed based on NETLAB Matlab package [11] and Texture Analysis/Synthesis Matlab package [8].

At the watermark decoder, only correlation detection is enough to extract the watermark message.

(a) (b) (c) (d)

Fig. 3. Original images (top) and watermarked images (bottom). (a) *Baboon*. $|r|$=0.254, SNR=17.4dB. (b) *Bridge* $|r|$=0.150, SNR=18.2dB. (c) *Man* $|r|$=0.103, SNR=21.9dB. (d) *Lena* $|r|$=0.074, SNR=27.1dB.



(a) (b) (c)

Fig. 4. Synthetic watermark. (a) original texture $\boldsymbol{x}_T$. (b) synthetic texture $\boldsymbol{x}_T'$. (c) watermark signal $\boldsymbol{x}_T' - \boldsymbol{x}_T$. White is positive signal and black is negative signal.

4.1 Watermarked image

We test our scheme on typical images in Figure 3. Our scheme has followed advantages. Firstly, it does not incite watermark intensity at sharp edges due to the three-factor image coding before watermarking. Secondly, it prefers to embed watermark into rough region rather than smooth region because texture segmentation and texture synthesis keep texture features. The characters are clearly exhibited on the watermark signal in Figure 4. However, because our scheme exploits texture region, its advantages disappear in images with sparse texture, e.g. image *Lena*.

4.2 Carrier pruning based on prior information

Spread-spectrum watermark detection utilizes the projection of watermarked image on the carrier. Larger is the projection, higher hiding capacity can be achieved. Since watermarked images always locate in the subspace spanned by the similar-looking image counterparts, if the designed carrier approaches the said subspace, it has more possibility to generate a large projection on the watermarked image. Here, we give an example of carrier pruning, by which hiding capacity is proved to be improved in the experiments of section 4.3 (to see Figure 5).

The proposed method of carrier pruning exploits the homogenous feature of texture, that is, different blocks within a texture look similar. In another word, block permutation alert texture appearance little. So we generate the carrier by means of 8×8 block permutations within each cluster of x_T, and the permutation is controlled by a key. The pruned carrier still keeps uncorrelated with original image due to their averaged normalized correlation of about 10^{-4}, while the texture can be synthesized fairly *similar* with the pruned carrier (modulated by watermark).

Given the x_T in a 512×512 image, the amount of possible carrier is about $(512^2/(3\times8^2))!^3 \approx 10^{11068}$, where texture block permutation is controlled within each one of three clusters. Although a monotone image or an awful permutation algorithm may reduce the uncertainty of carrier, the cryptographic attack of brute-force searching the carrier is still a burdensome work.

Carrier pruning here implies the watermarking scheme is not blind one, because the detector need x_T during detection. However, one may design a more elaborate scheme, where only concise prior information and special side information are used to prune the carrier.

4.3 Robustness against Wiener filtering attack

Sophisticate attackers can make estimation-based attack if they can obtain some prior knowledge of host image or watermark's statistics [13]. Image denoising provides a natural way to develop estimation-based attacks, where watermark is treated as a noise. Given the power spectrum of host image and watermark, one of the most malevolent attacks is the denoising by adaptive Wiener filter. So the most robust watermark should have a power spectrum directly proportional to the power spectrum of the host image [13]. We compare the power spectrum of watermark under different schemes in Figure 5. We have found that spread-spectrum watermark under spatial JND (just-noticeable-distortion) mask has a white power spectrum (Figure 5d) and watermark under wavelet JND mask concentrate its power in the middle frequency (Figure 5e) in the former literature [14]. However, synthetic watermark (Figure 5b and 5c) by our scheme has more similar power spectrum with the original image (Figure 5a), especially the watermark modulating the pruned carrier (Figure 5c).

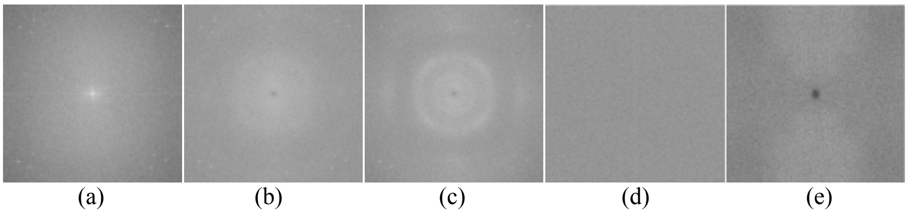


Fig. 5. Power spectrum by Fourier analysis. (a) host image *Baboon*. (b) synthetic watermark from Gaussian carrier. (c) synthetic watermark from pruned carrier. (d) spatial watermark modulating binomial carrier. (e) wavelet watermark modulating Gaussian carrier. Signal is normalized to 0 mean and unit variance before Fourier analysis. The grey level denotes logarithmic amplitude of Fourier components. Zero-frequency component is shifted to the center.

We compare our scheme with spatial JND scheme in robustness against Wiener filtering attack. The comparison is made in different situations of using common carrier and pruned carrier, and using 3×3 Wiener filter and 5×5 one. Robustness can be measured by the survival rate of correlation $|r|$. From the results in Figure 6, we can draw following conclusions. 1. Survival rate of correlation in our scheme is larger than spatial JND scheme by 30%~40%, so our scheme is more robust. 2. Robustness can be improved by using pruned carrier for either our scheme or spatial JND scheme. 3. Hiding capacity of our scheme can be improved by 10%~25% under no attack, and by 100%~200% under Wiener filtering attack if pruned carrier is used. However, pruned carrier is no helpful for spatial JND scheme under no attack.

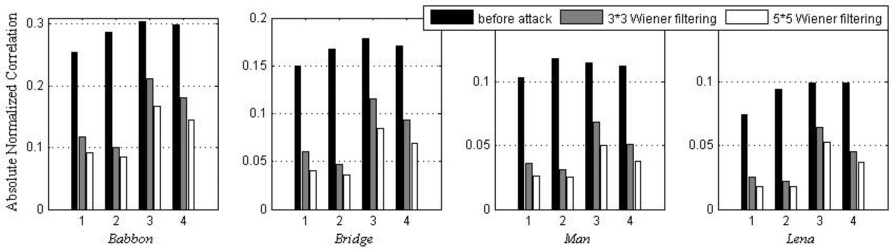


Fig. 6. Wiener attack' effect on hiding capacity under different scheme. 1 synthetic scheme using Gaussian carrier, 2 spatial JND scheme using binomial carrier, 3 synthetic scheme using pruned carrier, 4 spatial JND scheme using pruned carrier.

4 Conclusion

We have formulated the spread-spectrum watermarking as an optimization problem with a linear object and nonlinear distortion constraints, and presented a watermarking-by-synthesizing solution. Several properties of the scheme are noteworthy. First, pursuing the watermarked signal by optimization algorithm can obtain the suboptimal, if not optimal, solution with highest hiding capacity. Second, texture segmentation facilitates estimation of the *narrow* subspace where homogenous textures focalized, and is helpful to prune the carrier so as to improve hiding capacity. Last, synthetic watermarked image has a similar power spectrum with original image and thus robustness against adaptive filtering attack.

Moreover, the scheme enlightens a method of watermark attack. Under attacking-by-synthesizing, the SNR of attacked image can be fairly low, so that many traditional watermarks are hard to survive.

Property of the scheme also follows common heuristics: intensity change of low frequency band or edges in an image is easy to be noticed therefore is not suitable to utilized by watermarking scheme. Therefore, we believe it is necessary of discrimination on low frequency band, structural edges and texture for image watermarking. Watermark in structures has yet to be designed conforming to distortion measure of shape and skeleton.

Acknowledgement. This work is supported by NSFC (NO: 60572063), Doctoral Fund of Ministry of Education of China (NO: 20040487009), and the Cultivation Fund of the Key Scientific and Technical Innovation Project， Ministry of Education of China (NO: 705038).

References

1. P. Moulin and R. Koetter, "Data-hiding codes," *Proc. of the IEEE*, vol. 93, pp. 2083-2126, Dec. 2005.
2. I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Proc.*, vol. 6, pp.1673-1687, Dec. 1997.
3. L. Y. Wei, M. Levoy, "Fast texture synthesis using tree-structured vector quantization," In *Proc. ACM SIGGRAPH*, pp. 479-488, 2000.
4. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3/4, pp. 313-336, 1996.
5. Heeger, D. and Bergen, J. "Pyramid-based texture analysis/synthesis," In *Proc. ACM SIGGRAPH*, pp. 229-238, 1995.
6. S. C. Zhu, Y. N. Wu, and D. B. Mumford, "Filter, Random fields, and Maximum Entropy (FRAME) – Towards a Unified Theory for Texture Modeling," *Int'l Journal of Computer Vision*, vol. 27, pp. 107-126, 1998.
7. J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients." *Int'l Journal of Computer Vision*, vol.40, pp. 49-71, 2000.
8. J. Portilla and E. Simoncelli, "Texture Synthesis," http://www.cns.nyu.edu/~lcv/texture/, Apr. 2001.
9. N. Balakrishnan, K. Hariharakrishnan, and D. Schonfeld, "A new image representation algorithm inspired by image submodality models, redundancy reduction, and learning in biological vision," *IEEE Trans. Pattern Analysis & Machine Intelligence*, vol. 27, pp. 1367-1378, Sep. 2005.
10. M. Tipping and C. Bishop, "Mixtures of Probabilistic Principal Component Analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443-482, 1999.
11. I. Nabney and C. Bishop, "Netlab Neural Network Software," http://www.ncrg.aston.ac.uk/netlab, 2003.
12. A. Hyrarinen, "Fast ICA Matlab package," http://www.cis.hut.fi/projects/ica /fastlab, Apr. 2003.
13. S. Voloshynovskiy, S. Pereira, T. Pun, et al, "Attacks on Digital Watermarks: Classification, Estimation-Based Attacks, and Benchmarks." *IEEE Communications Magazine*, vol. 39, pp. 118-126, Aug. 2001.
14. F. Zhang, W. Y. Liu, and C. X. Liu, "High capacity watermarking in nonedge texture under statistical distortion constraint," In *Asian Conf on Computer Vision*, 2007 (to be published).
15. E. P. Simoncelli and W. T. Freeman. "The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation," In *IEEE Int'l Conf on Image Processing*, pp. 444-447, Oct. 1995.