Data-Embeddable Texture Synthesis

Hirofumi Otori and Shigeru Kuriyama

Toyohashi University of Technology, 1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi, 441-8580, Japan
{otori,kuriyama}@val.ics.tut.ac.jp

Abstract. Data hiding techniques onto images provide tools for protecting copyright or sending secret messages, and they are currently utilized as a simple input device of a cell phone by detecting a data embedded in an image with an equipped digital camera. This paper presents a method of synthesizing texture images for embedding arbitrary data by utilizing the smart techniques of generating repetitive texture patterns through feature learning of a sample image. We extended the techniques so that a synthesized image can effectively conceal the embedded pattern, and the pattern can be robustly detected from a photographed image. We demonstrate the feasibility of our techniques using texture samples including an image scanned from real material.

1 Introduction

The advanced image processing capability of handy cell phones enables a novel data input tool with images. A QR-code[1] [1] was developed as an efficient and robust 2D bar code, as shown in Fig. 1(a), and it has become a popular device in Japanese culture for inputting a URL with the photographing capability equipped on a cell phone. With this device, we can avoid the troublesome operations with the small numerical key-pad of cell phones. However, the meaningless binary pattern of the QR-code damages the aesthetic quality of the printed images. Some extension methods have been developed to overcome such a defect, for example, by coloring [2] (Fig. 1(b)) or modifying to small icons (Fig. 1(c)); these methods, however, essentially provide the aesthetic style of a 2D bar code. Recently, some data-embedding methods onto natural images have been proposed as a replacement for the QR-code. Some techniques intentionally modulate the specific color component or frequency channel of the image according to embedded data. Most methods incorporate either type of numerical techniques developed as watermarking or steganography [3]. The former technique is utilized for protecting copyright, which requires robustness against intentional attack through alteration or destruction of images, and the latter is designed to increase the amount of hidden data (or payload) while sacrificing robustness.

The data-embedding techniques on printed media, like the QR-code, have a property similar to those of steganography, sacrificing robustness against attacks

[1] QR code is a trademark of DENSO WAVE Inc.

A. Butz et al. (Eds.): SG 2007, LNCS 4569, pp. 146–157, 2007.

(a) QR code[1] (b) Color bar code (c) Design QR[2]

Fig. 1. Examples of two-dimensional bar codes

on copyright, but they require another type of robustness, that is, tolerance for the noise caused by analog transmissions through printing and photographing processes. Such robustness is often obtained by increasing the amount of modulations by sacrificing the quality of original images. Another solution introduces redundancy, for example, the use of the Reed-Solomon code [4], but this strategy is useless when most of the data is undetectable.

The watermarking and steganography mainly treat natural images, and there are few methods developed for artificial images such as cartoons or computer-generated images. For example, data embeddings with run length code [5] or block pattern [6] have been proposed for binary images, but their data detection strategies neglect robustness against noisy data. The steganography on a text document slightly modifies the dots of each letter image [7], but it assumes the high-resolution scanning process of a business copy machine and is hard to extend the capability to arbitrary types of images.

This paper focuses on the texture images for embedding data because texture patterns are widely utilized artificial images. More importantly, texture images can be automatically generated by computing the feature of the iterative patterns of real objects such as woods or cloth, and thus we can embed data by affixing a seal of a synthesized image on a real object in an inconspicuous manner, by which the aesthetic quality of the object's appearance is guaranteed.

Our approach utilizes texture synthesis technique for embedding data so as to be robust against the noise caused by analog transmission. Instead of changing the color component of images, we directly paint the data by converting its numerical value into a dotted colored pattern, and then automatically coat a texture image onto the painted pattern from a sample image (or exemplar) so as to conceal its existence with a natural texture pattern. The recently proposed texture synthesis algorithm [8] and its extensions [9,10] were utilized for synthesizing seamless and continuous images with the constraints of local patterns corresponding to the embedded data.

We first introduce an encoding method with painted patterns in Section 2 as a basic mechanism to embed and detect data. Section 3 explains about smart texture synthesis from an initial painting pattern which is suited to conceal a data-embedded pattern, and we demonstrate the feasibility of our method by showing examples of texture synthesis and data detection by photographing the printed images in Section 4. We finally give conclusions in Section 5.

[2] Design QR is a trademark of IT Design Inc. http://d-qr.net/

2 Encoding with Painted Local Binary Pattern

We utilize a feature vector for texture images, called local binary pattern (or LBP) [11]. It was first introduced as a complementary measure for local image contrast, and the definition of pixel neighborhood for computing features is extended to arbitrary circular neighborhoods. The code is computed by comparing the value of a centered pixel against those of neighboring pixels.

We first divide a texture image captured by a digital camera into pre-defined number of square blocks. The LBP is then given for each block by the difference of a pixel value, denoted by $g(\boldsymbol{p})$ at the $\boldsymbol{p}$ coordinates, between a pixel located at the center of the block and the P circular positions at an equal distance (see Fig. 2),

$$LBP_{P.R}\left(\boldsymbol{p}_c\right) = \sum_{n=0}^{P-1} s\left(g\left(\boldsymbol{p}_n\right) - g\left(\boldsymbol{p}_c\right)\right) 2^n, s\left(x\right) = \begin{cases} 1 & : x \geq 0 \\ 0 & : x < 0 \end{cases} \tag{1}$$

where $\boldsymbol{p}_c = (x_c, y_c)$ denotes the center position, and the n-th circular position is given by $\boldsymbol{p}_n = \boldsymbol{p}_c + R\left(\cos\left(2\pi n/P\right), -\sin\left(2\pi n/P\right)\right)$ with the distance R from the center. Each $LBP_{P.R}(\boldsymbol{p}_c)$ therefore represents P bits information whose n-th bit has the value of $s(g(\boldsymbol{p}_n) - g(\boldsymbol{p}_c))$. Notice that we simply determine the pixel values by discretizing the coordinate $\boldsymbol{p}_n$ to integers, instead of interpolating the values of neighboring pixels.

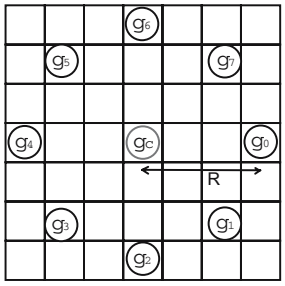


Fig. 2. Pixel sampling positions for computing LBP code (P=8)

The existing technique for analyzing texture images defines the feature by considering rotation invariance pattern and uniformity measure [11]; our method, however, neglects these constraints and allows arbitrary patterns of $LBP_{P,R}(\boldsymbol{p})$. The existing method uses grayscale level or each RGB component as a pixel value, but it can be replaced by an arbitrary color component, if it can be uniquely converted from RGB components. We usually select the component that is insensitive to human vision system; for example, the Cb component on Y-Cb-Cr color space, as the pixel value $g(\boldsymbol{p})$. However, the component must be selected so as to have a high contrast for ensuring the robustness against noisy transmission. The LBP code is then computed by extracting the insensitive

component of the color at each pixel and by using equation (1). The extracted LBP codes from all blocks represent all information embedded in the texture image. Therefore, arbitrary data are divided and embedded by painting the pattern of LBP onto each block in their sequence. A texture image is then coated so as to conceal this painted pattern by referencing an exemplar.

3 Texture Synthesis with LBP

3.1 Initial Painting with Embedded Data

The initial paint pattern of a synthesized texture is made from an exemplar. We first extract the insensitive color component as each pixel value $g(\boldsymbol{p})$ and compute the median, denoted by g_m, from all pixels. Every pixel is then divided into two classes: the upper class for $g(\boldsymbol{p}) \geq g_m + T$ and the lower class for $g_m - T \geq g(\boldsymbol{p})$, where the pixel values residing in the middle range; $g_m + T > g(\boldsymbol{p}) > g_m - T$, are neglected in painting LBP. The scalar value T isolates the pixel values in upper and lower classes from the median value g_m, and an increase of T enhances robustness against the noisy variation caused in printing and photographing. However, the larger T narrows down the range of usable colors, and we experimentally found that $T = 30$ ensures good balance for 8-bit pixel values. After categorizing the constituent colors into two classes, the central pixel is always painted by the color whose component is equivalent to the median g_m, and the surrounding pixels are painted by randomly selecting the color whose component belongs to the upper and lower classes for the embedded binary data of 1 and 0, respectively.

Fig. 3(b) represents an example of the initial paint pattern made from the exemplar in Fig. 3(a). We intentionally paint the same value at the nearby pixels of the center and surrounding pixels for increasing robustness against the positional error in sampling. Because each LBP code can represent P bits, embedding data of n bits requires $\lceil n/P \rceil$ image blocks regularly arranged as shown in Fig. 3(b), where $\lceil\ \rceil$ denotes a ceil function.

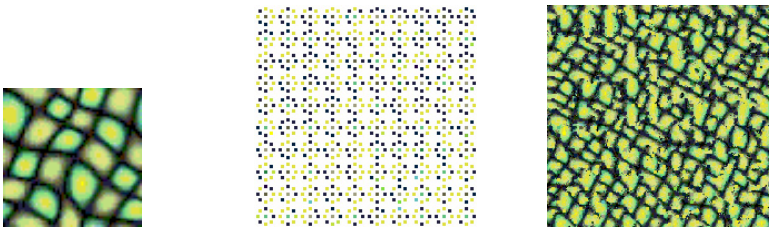
3.2 Texture Coating with Exemplar

Next, we randomly select the pixel whose value is null and paint it by computing the differences in the insensitive component of the 8 neighboring pixels to the corresponding pixels in the exemplar as,

$$S(\boldsymbol{p}, \boldsymbol{q}) = \sum_{\boldsymbol{r} \in \nu} D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) \tag{2}$$

$$D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = \begin{cases} 0 & \text{if } g(\boldsymbol{p} + \boldsymbol{r}) \text{ is null} \\ \left(g(\boldsymbol{p} + \boldsymbol{r}) - \tilde{g}(\boldsymbol{q} + \boldsymbol{r})\right)^2 & \text{else} \end{cases}$$

where $\boldsymbol{p}$ denotes the 2D position of the randomly selected pixel, ν is the set of offset vectors for indicating 8 neighbors; $\nu := \{(s,t) | s,t \in (-1,0,1), s \neq t\}$



(a) exemplar (sample image)

(b) initial painting for 100-byte data

(c) synthesized image with basic algorithm

Fig. 3. Example of texture synthesis

and $\tilde{g}(\boldsymbol{q}+\boldsymbol{r})$ is the pixel value of the exemplar. Notice that the large $S(\boldsymbol{p},\boldsymbol{q})$ corresponds to the dissimilar pattern of neighboring pixels between synthesize texture and exemplar. We then set the pixel value $g(\boldsymbol{p})$ by those of the most similar pixel of the exemplar that has minimum $S(\boldsymbol{p},\boldsymbol{q})$ as

$$\hat{\boldsymbol{q}} = \arg\min_{\boldsymbol{q}} S(\boldsymbol{p},\boldsymbol{q}) \ , \quad g(\boldsymbol{p}) = \tilde{\boldsymbol{g}}(\hat{\boldsymbol{q}}) \tag{3}$$

and the pixel at $\boldsymbol{p}$ is painted by the color whose component corresponds to $g(\boldsymbol{p})$. The equations (2) and (3) are interpreted as template matching where a 3 by 3 pixels' region at $\boldsymbol{p}$ is regarded as a template and the best matching region is searched within the exemplar. The random selection of $\boldsymbol{p}$ is repeated until all pixels are painted. Fig. 3(c) shows the texture generated by this coating algorithm from the exemplar of Fig. 3(a) and the initial painting of Fig. 3(b).

3.3 Acceleration with Coherence Map

The abovementioned coating algorithm uniquely determines each pixel value by computing the pattern dissimilarity for all pixels in the exemplar; this exhaustive search for minimum $S(\boldsymbol{p},\boldsymbol{q})$, however, requires a large amount of computation that proportionally increases for the product of the pixel sizes of the synthesized and exemplar images. We therefore introduce a method [9] that considers the coherency of the exemplar by recording the history of the similarity search onto every pixel.

A coherence map is defined as a set of two-dimensional indices, denoted by $\boldsymbol{m}(\boldsymbol{p})$, which is assigned to every pixel of a synthesized image for recording the position of the corresponding exemplar's pixel (see Fig. 4(b)). Then the searching space of $\boldsymbol{q}$ in the equation (3) is narrowed down as

$$\boldsymbol{q} \in \boldsymbol{m}(\boldsymbol{p}+\boldsymbol{r}) - \boldsymbol{r} \quad for \ \ \boldsymbol{r} \in \nu \tag{4}$$

where the index is updated by the position of the most similar pixel as $\boldsymbol{m}(\boldsymbol{p}) = \hat{\boldsymbol{q}}$. The equation (4) is interpreted as the template matching among the exemplar's

pixels whose nearby pixel is used for painting the nearby pixel of $\boldsymbol{p}$ in the same adjacency. Notice that we conduct the exhaustive search of $\boldsymbol{q}$ when the index $\boldsymbol{m}(\boldsymbol{p}+\boldsymbol{r})$ is null. We experimentally confirmed that this coherence-based similarity search can speed up the synthesis by 10 times, compared to the basic coating algorithm.

3.4 Quality Improvement with Similarity Map and Re-coating

The coherence-based similarity search is efficient, but it decreases the quality as a trade-off with drastically narrowing down the searching space of $\boldsymbol{q}$. We therefore efficiently expand the searching space by considering the similarity inside the exemplar. In particular, we pre-compute the dissimilarity of each square region of N by N pixels inside the exemplar, denoted by $B_N(\boldsymbol{q}, \tilde{\boldsymbol{q}})$, as

$$B_N\left(\boldsymbol{q}, \tilde{\boldsymbol{q}}\right) = \sum_{\boldsymbol{s}\in\mu} \left(\tilde{g}\left(\boldsymbol{q}+\boldsymbol{s}\right) - \tilde{g}\left(\tilde{\boldsymbol{q}}+\boldsymbol{s}\right)\right)^2 \tag{5}$$

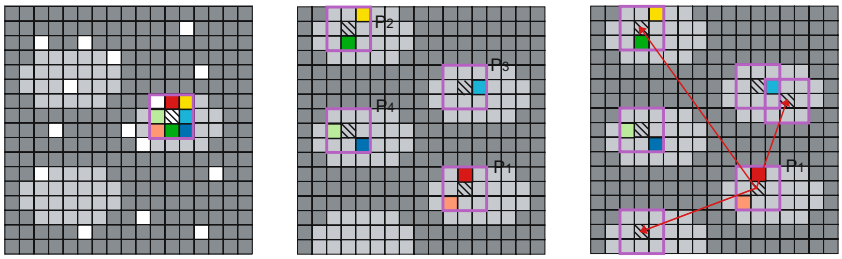
where μ denotes the offset of each pixel in the region from a center position; $\mu := \{(s,t)|s,t \in (-h,..,-1,0,1,...,h)\}$ $h = (N-1)/2$, and the values of $\boldsymbol{q}$ and $\tilde{\boldsymbol{q}}$ are constrained so that $\boldsymbol{q}+\boldsymbol{s}$ and $\tilde{\boldsymbol{q}}+\boldsymbol{s}$ fall in the pixel range of the exemplar. The size of the square region N for pattern matching is adaptively tuned in a range of 5 to 21 depending on the feature of texture, and we use $N = 15$ as a default.

We next construct a map [10] for indexing a pattern similarity among the pixels in the exemplar by setting the $\tilde{\boldsymbol{q}}$ of the i-th-smallest $B_N(\boldsymbol{q}, \tilde{\boldsymbol{q}})$ to $\boldsymbol{u}_i(\boldsymbol{q})$, up to the k positions. The most similar pixel in the exemplar is then searched among the positions indexed by the similarity map for the search space given by coherence map in equation (4) as (see Fig. 4(c))

$$\boldsymbol{q} \in \boldsymbol{u}_i\left(\boldsymbol{m}(\boldsymbol{p}+\boldsymbol{r}) - \boldsymbol{r}\right) \quad for\ \ \boldsymbol{r} \in \nu\ \ and\ \ i = 0, 1, \ldots, k \tag{6}$$

where $\boldsymbol{u}_0(\boldsymbol{x})$ represents the identical map; $\boldsymbol{u}_0(\boldsymbol{x}) := \boldsymbol{x}$. With the similarity map, the search space of $\boldsymbol{q}$ is expanded from the similarity record of coherence map to the corresponding k similar pattern in the exemplar. We experimentally confirmed that this expanded search with the similarity map can still speed up the synthesis by 4 times, compared to the basic coating algorithm.

Furthermore, we improve the quality of the synthesized images by repeating the above coating algorithm. After all pixels are painted, we re-coat them all except for the pixels painted for embedding data. The first coating phase determines the most similar pattern only with the painted pixels; in other words, the computation of similarity in equation (2) is inaccurate in the early stage of coating because the effect of many pixels is missed by the rule of $D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = 0$ if $g(\boldsymbol{p}+\boldsymbol{r})$ is null . In the second coating phase, all pixel values and indices of coherence map $\boldsymbol{m}(\boldsymbol{p}+\boldsymbol{r})$ have been tentatively determined, and thus the analysis of texture pattern becomes more accurate. We experimentally confirmed that the re-coating in more than the second trial cannot particularly increase the quality of the image, and we therefore execute the re-coating only once.



(a) synthesized image (b) coherence map (c) similarity map

Fig. 4. Schematic representation of map generations

4 Experimental Results

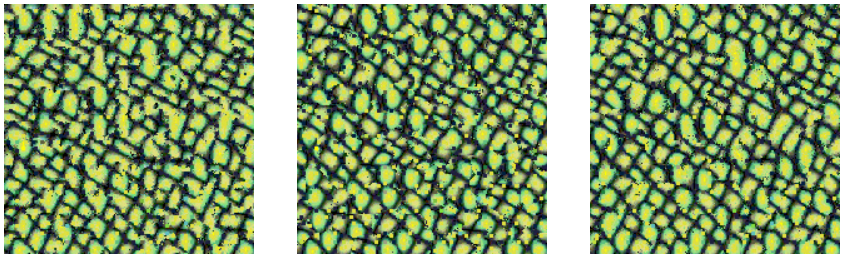
4.1 Texture Synthesis

The images in Fig. 5 present the synthesized texture from the exemplar and initial painting in Fig. 3 (a) and (b). Figure (a) is synthesized by using the basic algorithm in equation (1) (the same image as in Fig. 3(c)), figure (b) is synthesized with the coherence map, and figure (c) is synthesized by additionally using the similarity map. The texture image in figure (b) shows the increase of conspicuous spots due to the narrow search space with the coherence map, and the image in figure (c) demonstrates the decrease of the spots due to the expanded search space with the similarity map, by which image quality is recovered.

Fig. 6 demonstrates the effectiveness of our random and multiple coating. Figure (a) is synthesized by selecting the unpainted pixels in scan-line order, and figure (b) randomly selects unpainted pixels, which definitely decreases the undesirable spots of LBP. Figure (c) presents the image after re-coating the image of figure (b) in scan-line order at one time, by which tiny noisy spots are removed. These samples are generated using the Cb component of Y-Cb-Cr color space as a pixel value, and the similarity map is constructed with the matching region of $N = 15$ size.

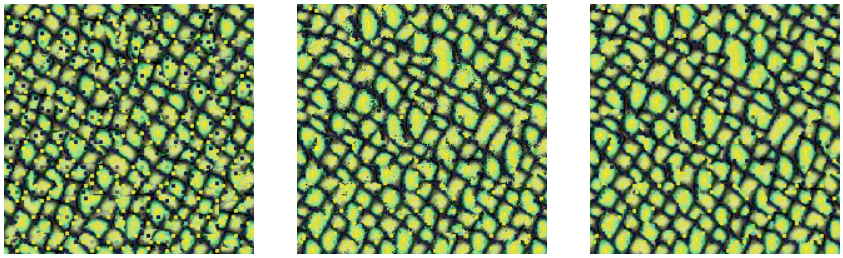
Fig. 7 shows various kinds of texture synthesis, which arranges exemplars (right), textures synthesized through ordinary procedures without embedding data (middle), and data-embedded textures with our method (left). The payload is set to 25 bytes, and the size of the matching region for the similarity map is tuned to $N =$ (a) 7, (b) 15, (c) 11. The color component of (a) Cb, (b) Y, and (c) Cr, is selected as a pixel value, respectively, and the textures without embedding data are synthesized by coating pixels in scan-line order.

Fig. 8 demonstrates the examples synthesized from the exemplar in figure (b) that is generated by scanning the surface of real wooden material in figure (a). Figures (c), (d), and (e) show initial paint patterns for the embedded data of 100, 25, and 16, bytes, respectively, and figures (f), (g), and (h) show the synthesized images using the coherence and similarity maps and re-coating for these paint patterns. These examples clearly show the trade-off between the payload and



(a) basic algorithm (b) with coherence (c) with similarity

Fig. 5. Effect of coherence and similarity maps on synthesized textures



(a) scan-lines (b) random (c) re-coating

Fig. 6. Effect of pixel coating strategy

quality. The color space of Y-Cb-Cr of this wooden image has a very low contrast for Cb and Cr components, and we therefore adopted the component Y as a pixel value.

4.2 Accuracy in Data Detection

We have examined the robustness of our data-detecting mechanism. The 100 bytes data-embedded texture images of 200×200 pixels are printed on super-fine A4 paper in a 2x2 inch square region with an EPSON PX-G5100 color ink-jet printer, and they were photographed with a CANON PowerShotG5 digital camera that is fixed on the tripod in such a way as to be parallel to the paper at a distance of 30 cm, where a fluorescent lamp was used for lighting.

Table 1 shows the result of data detection for the images in Fig. 6(c) and Fig. 8 (f), (g), (h). Error bits were computed by averaging the number of error bits for 10 trials. The embedded data cannot be perfectly detected, but the ratios of error bits are small enough to recover the information with some error correcting techniques such as the Reed-Solomon code.

Next, we have implemented our data-detecting mechanism on a cell phone of FOMA D903i with Doja API, and printed the 25 bytes data-embedded texture image in Fig. 6(c) of 100 by 100 pixels in a 1 by 1 inch square region with the

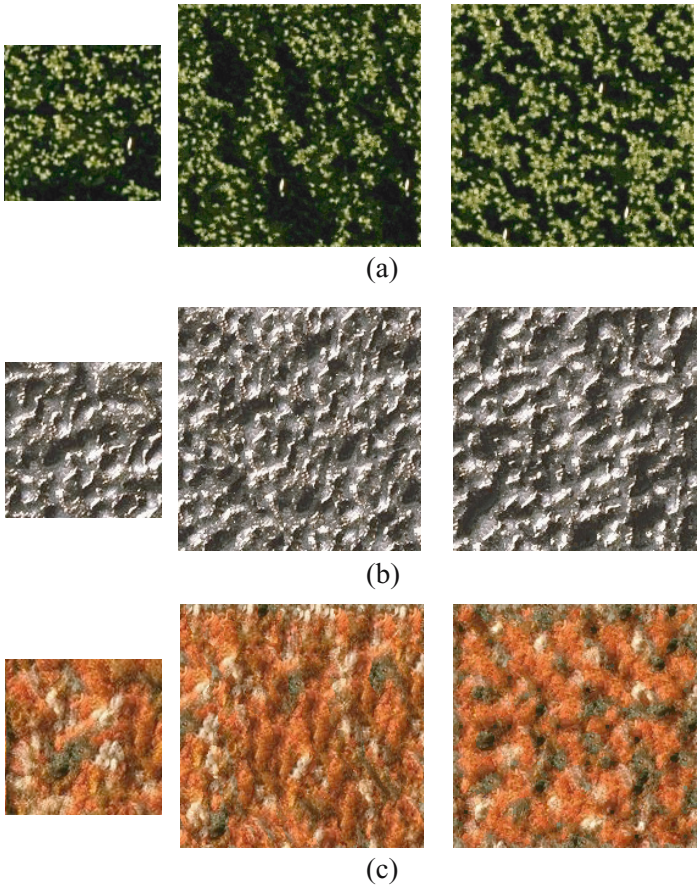


Fig. 7. Examples of exemplar, ordinary synthesis, and data-embedded synthesis

Table 1. Error rate for various size of embedded data

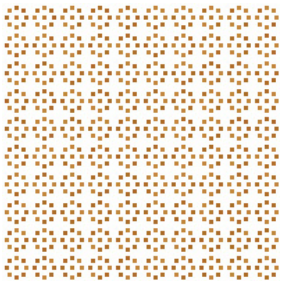
Example	Payload(byte)	Error bits	% of missing data
Fig.6(c)	100	1.2	0.15
Fig.8(f)	100	10.0	1.25
Fig.8(g)	25	0.0	0.00
Fig.8(h)	16	1.0	0.01

same printer. The printed image was captured by the phone's camera with a macro mode of 176 by 144 pixels supported by hand at a distance of 5 to 8 cm. Through 10 trials, we had average error bits of 4.8 where the maximum and minimum of the error bits was 14 and 0, respectively.

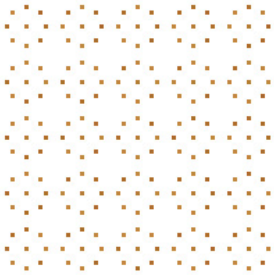
Table 2 compares the payload of our method against the existing market products. We can only show a rough comparison due to the lack of detailed

(a) real wooden material

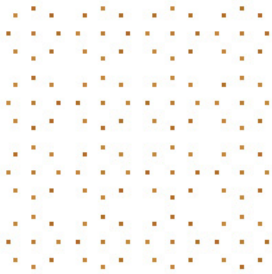
(b) exemplar scanned from (a)



(c) payloads of 100



(d) payloads of 25



(e) payloads of 16

(f)

(g)

(h)

Fig. 8. Examples of synthesized wooden textures. Figure (b) is an image scanned from the material in (a) used as an exemplar. Figures (c), (d), and (e) are initial paintings for given payloads of 100, 25, and 16 bytes, respectively. Textures in figures (f), (g), and (h) are generated from the initial painting in figures (c), (d), and (e), respectively.

specification, and these payloads cannot be fairly estimated because of the difference in measurement conditions in terms of paper-size, resolution of camera, lighting, and so on. The displayed values of payload therefore lack the accuracy in evaluation. However, we can safely conclude that the embeddable data

Table 2. Comparison of payload with market products

Product	Payload(byte)	Methodology
Our method	$25 \sim 100$	LBP code
QR-Code[1]	3000	2D bar code
Pasha-warp[12]	3	Frequency modulation
FP-code[13]	5	Color modulation

in our method is larger than those in natural image modulations, but smaller than those in the 2D bar code. This shows that our approach based on texture pattern synthesis has the advantage in the payload over natural image modulations.

5 Conclusion and Discussion

We have proposed a novel texture image synthesis for embedding data with little aesthetic defect. Our technical contribution is the introduction of random coating and re-coating to improve the quality of the texture image synthesized from the initial painting of LBP. We have also demonstrated that the efficient computation of pattern similarity using coherence and similarity maps is applicable to our texture synthesis. Our method could be the replacement of QR code in that it can provide visually meaningful images, and also might surpass existing techniques of natural image modulations in payload.

Our algorithm focuses on the textures that are iteratively generated by learning a pattern of an exemplar, and thus is infeasible for a procedurally and randomly generated pattern, for example, those generated using Perlin's noise functions [14]. However, the LBP is still available to extract features of such a class of textures. We successfully embedded the data onto the shape of a histogram of the LBPs that are computed for every pixel inside a divided image block. However, the payload of this method turned out to be far smaller than those of the method proposed in this paper.

Our current implementation requires border lines on a texture image for extracting the square region of a data-embedded area, and this limitation should be removed by developing a technique of automatically determining the square region. The texture pattern is often used as a background, and we should develop a method for coating texture with letter images. The payload of our method should be more accurately estimated in future study.

Acknowledgment

The authors would like to thank the members of Wood One Co., Ltd. for donating the sample of wooden material in Fig. 8(a).

References

1. DENSO Wave. International Standard ISO/IEC18004 (2000)
2. Tack.-don, H., Cheol.-ho, H., Nam.-kyu, L., Eun.-dong, H.: Machine readable code image and method of encoding and decoding the same. United States Patent No. US 7,020,327 B2
3. Provos, N., Honeyman, P.: Hide and Seek: An Introduction to Steganography. IEEE Security & Privacy 1(3), 32–44 (2003)
4. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. SIAM J, pp. 300-304 (1960)
5. Hwang, K.-F., Chang, C.-C.: A Run-Length Mechanism for Hiding Data into Binary Images. In: Proceedings of Pacific Rim Workshop on Digital Steganography 2002, pp. 71-74 (2002)
6. Tseng, Y. C., Pan, H. K.: Secure and Invisible Data Hiding in 2-color Images. In: Proceedings of INFOCOM 2001, pp. 887–896 (2001)
7. Fujii, Y., Nakano, K., Echizen, K., Yosiura, Y.: Digital Watermarking Scheme for Binary Image (in Japanese) Japan Patent 2004-289783
8. Wei, L.-Y., Levoy, M.: Fast Texture Synthesis using Tree-structured Vector Quantization. In: Proceedings of SIGGRAPH 2000, pp. 479–488 (2000)
9. Ashikhmin, M.: Synthesizing natural textures. Symposium on Interactive 3D Graphics, pp. 217–226 (2001)
10. Tong, X., Zhang, J., Lui, L., Wang, X., Guo, B., Shum, H.: Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces. In: ACM SIGGRAPH 2002, pp. 665–672 (2002)
11. Mäenpää, T., Pietikäinen, M.: Texture analysis with local binary patterns. Handbook of Pattern Recognition and Computer Vision 3rd ed. World Scientific, pp. 197–216 (2005)
12. Nakamura, T., Ogawa, H., Tomioka, A., Takashima, Y.: Improved Digital Watermark Robustness againt Translation and/or Cropping of an Image Area. IEICE Trans. Fundamentals E83-A(1), 68–76 (2000)
13. Noda, T., Moroo, J., Chiba, H.: Print-type Steganography Technology (in Japanese). Magazine FUJITSU 2006-5 57(3), 320–324 (2006)
14. Perlin, K.: An Image Synthesizer. In: Proceedings of SIGGRAPH '85 85, 287–296 (1985)