Texture Synthesis for Mobile Data Communications

Hirofumi Otori and Shigeru Kuriyama

***Abstract*—A digital camera mounted on a mobile phone is utilized as a data input device to obtain embedded data by analyzing the pattern of an image code such as a 2D bar code. This article proposes a new type of image coding method using texture image synthesis. Regularly arranged dotted-pattern is first painted with colors picked out from a texture sample, for having features corresponding to embedded data. Our texture synthesis technique then camouflages the dotted-pattern using the same texture sample while preserving the quality comparable to that of existing synthesis techniques. The textured code provides the conventional bar code with an aesthetic appeal and is used for tagging data onto real texture objects, which can form a basis for ubiquitous mobile data communications. This technical approach has the potential to explore new application fields of example-based, computer-generated texture images.**
Keywords:
Texture synthesis, image coding, local binary pattern, data hiding, code detection.

I. Introduction

A high-quality digital camera is equipped on most recent mobile phones, and it is often utilized as a handy image code detector. Two-dimensional bar codes are frequently utilized for coding URL that guides to relevant information services on the web. The black and white matrix-formed patterns of the 2D bar code are efficient for robustly coding data, but their meaningless images often diminish the attractiveness of printed matter such as magazines and wrapping papers. We have found this can be overcome by introducing recent CG technologies to the data-coding mechanism.

In this article, we focus on texture images of iterative patterns as a new type of image code. We directly paint colored dotted patterns to be robustly detected from a photographed image. Then, we synthesize texture images to conceal the patterns while preserving their colors. Our novel approach can embed larger information than existing technologies, and the quality of the synthesized images is assured owing to a smart algorithm of texture image generation.

Using texture images as a replacement for bar codes has two benefits. Iterative texture is sometimes used as a background image of a printed matter, and information can be embedded on some blank places unoccupied by foreground images. Another promising application is the use of a camouflage code on real material. By synthesizing a texture with a sample image scanned from a real material such as wood or cloth, we can embed data by affixing a seal of the printed texture on the material in an inconspicuous manner. This camouflage code can be used to record product information of the material while avoiding unsightly coded images. We demonstrate the feasibility of this technique through actual implementation on a mobile phone, and show that our robust and efficient data detection mechanism can read coded data within a reasonable response time and error ratio.

Toyohashi University of Technology, 1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi, 441-8580, Japan TEL:+81-532-44-6737, FAX:+81-532-44-6757 otori@val.ics.tut.ac.jp, kuriyama@ics.tut.ac.jp

Figure 1 shows various kinds of texture synthesis, which displays sample seed images, textures synthesized with an existing pixel-based method [1], and 25-byte data-coded textures with our method. Although the resulting images are slightly different in appearance, their quality is nearly the same.

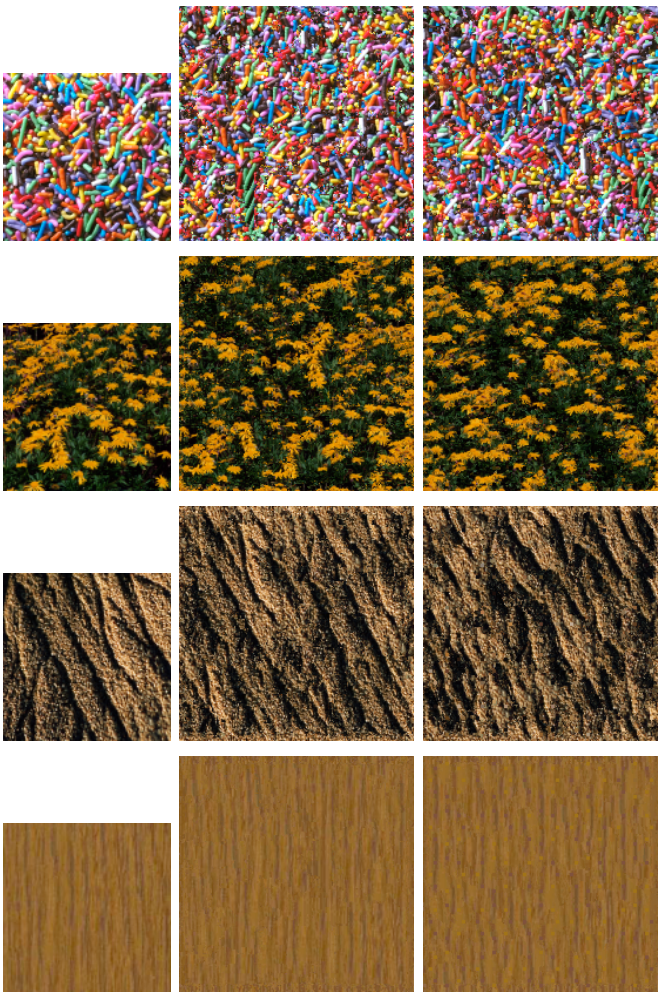


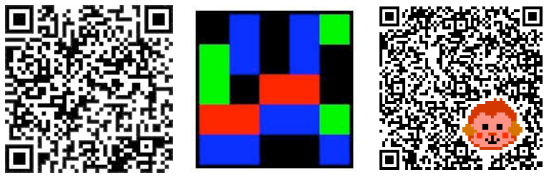
Fig. 1. Examples of (a) sample image, (b) ordinary synthesis, and (c) data-coded synthesis.

II. Related works

A. Data coding and hiding

A QR-code[1] was developed as an efficient and robust 2D bar code, as shown in Figure 2(a), and it has become a popular

device to read a URL for navigating web pages and services, without the troublesome operations of a small numerical keypad. Several methods have been developed to introduce some eye-pleasing aesthetic pattern, for example, a Color-code[2] in Figure 2(b) adds colors, and a Design-QR[3] partially modifies the patterns of the QR code to small icons, as shown in Figure 2(c). These approaches, however, embody the style of the matrix form and impose rigid constraints on their visual design.



(a) (b) (c)

Fig. 2. Examples of two-dimensional bar codes. (a) QR-code, (b) Color-code, (c) Design-QR.

Recently, natural images are also utilized as a replacement for the QR code. Some techniques intentionally modulate the specific color or frequency component of the image according to embedded data. Most methods incorporate either of two numerical techniques developed, watermarking or steganography [2]. Watermarking is utilized to protect copyright, which requires robustness against intentional attack through alteration or destruction of images, and steganography is designed to increase the amount of hidden data while sacrificing robustness. These traditional approaches have only a restricted amount of embedded data (or payload) for assuring robustness against noise caused by actual lights, lenses, and imaging devices. Also, increasing such robustness requires a large modulation in the color or frequency component of an original image, which often seriously compromises the quality of the resulting image.

B. Texture synthesis

Texture image synthesis using the example-based approach may be roughly classified into two types; pixel-based and patch-based. The former technique determines the color of every pixel via a pattern matching process using a square sample image as a template, and the latter adaptively arranges the sample image, called a patch, while ensuring seamless and continuous connection of the patches (see [3] for the detailed explanation of these techniques).

We here focus on the pixel-based methodologies [1], [4], [5], [6] that determine the colors of every pixel by referencing a sample image. These methods usually paint a texture image from scratch. Our technique, however, is to start painting from an initial dotted pattern generated for coding data. This approach is similar to texture generation from initially painted images [5] or flow fields [6] by which the resulting images are transformed. Such techniques try to preserve the visual features of the initial image while allowing a slight change in color with soft-constraints. Our method, on the other hand, preserves the color of the initial image with hard-constraints while making the visual features inconspicuous. Our approach essentially uses a greedy method for effectively conceal the initially given pattern, and more sophisticated optimizations [6] have the potential to improve the quality of the resulting image by sacrificing computational simplicity and efficiency.

[1]QR-code is a trademark of DENSO WAVE Inc. [http://www.qrcode.com/].
[2]Color-code is a trademark of ColorZip Japan [http://www.colorzip.com/].
[3]Design-QR is a trademark of IT Design Inc. [http://d-qr.net/].

III. Data Coding with LBP

Our data-coding strategy first converts bit-patterns of information into colored dotted patterns. We sparsely arrange colored dots and convert binary values into the difference of a specific color component, which is called *a coded component*, whereas existing bar codes use densely-arranged black and white dots. Although our texture synthesis allows any type of regularly-arranged pattern, we experimentally found that the existing feature vector for texture images, called local binary pattern (or LBP) [7], is best suited to synthesize high-quality images with our method for many types of textures.

This code is computed by comparing the value of a centered pixel against those of circularly-arranged nearby pixels. The coded data are divided by P bits and LBP is computed from the difference of a coded component between a pixel located at the center $\boldsymbol{p}_c$ of the block and the P circular positions regularly arranged as $\boldsymbol{p}_n = \boldsymbol{p}_c + R\left(\cos\left(2\pi n/P\right), -\sin\left(2\pi n/P\right)\right),\ n = 0, \ldots, P-1$ with the distance R from the center (see Figure 3):

$$LBP_{P,R}\left(\boldsymbol{p}_c\right) = \sum_{n=0}^{P-1} s\left(g\left(\boldsymbol{p}_n\right) - g\left(\boldsymbol{p}_c\right)\right) 2^n, \qquad (1)$$

$$s\left(x\right) = \begin{cases} 1 & : x \geq 0 \\ 0 & : x < 0 \end{cases}$$

where $g(\boldsymbol{p})$ denotes the coded component at $\boldsymbol{p}$, and $s\left(g(\boldsymbol{p}_n) - g(\boldsymbol{p}_c)\right)$ corresponds to the value of the n-th bit. Notice that the coordinates of the circular positions $\boldsymbol{p}_n$ are actually rounded off to integers for fitting them to discrete pixel positions.

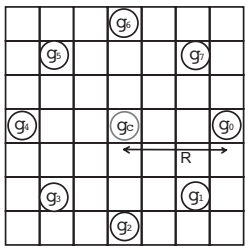


Fig. 3. Pixel sampling positions for computing LBP code (P=8).

We divide a region of synthesized texture image into a predefined number of square blocks, and arbitrary data are divided and coded by painting the pattern of LBP onto each block in a scan-line sequence (i.e. from left to right, then top to bottom). A texture image is then coated on so as to conceal this painted LBP code. Because each LBP can represent P bits, embedding data of n bits requires regularly-arranged $\lceil n/P \rceil$

image blocks, where $\lceil\ \rceil$ denotes a ceil function. Whole data are therefore recovered by concatenating the P bits for each block in the same scan-line sequence. We usually set $P = 8$ so that each block embeds one byte of data (or character).

The original LBP uses grayscale level or each RGB component as $g(\boldsymbol{p})$, but it can be replaced by an arbitrary color component that can be uniquely converted from RGB. We usually select a color component insensitive to the human vision system as the coded component; for example, the Cb component on Y-Cb-Cr color space. However, the component must have wide distribution for ensuring robustness against color distortions. For this reason, we are forced to select a visually sensitive component if all insensitive components have narrow distributions.

IV. Painting of LBP code

A. Classification of colors

Every pixel color of a synthesized texture is determined by adaptively selecting one from those included in a sample image called an exemplar, and the image of LBP code is camouflaged by selecting constituent colors from the subset of the same group. Let $\tilde{\boldsymbol{c}}(\boldsymbol{p})$ be the color of the pixel located at $\boldsymbol{p}$ in an exemplar, and let $\tilde{g}(\boldsymbol{p})$ be the corresponding coded component. We first extract the insensitive color component as $\tilde{g}(\boldsymbol{p})$ from all pixels and compute the median g_m. Every pixel color $\tilde{\boldsymbol{c}}(\boldsymbol{p})$ is then divided into three classes: the median class for $\tilde{g}(\boldsymbol{p}) \equiv g_m$, the upper class for $\tilde{g}(\boldsymbol{p}) \geq g_m + T$, and the lower class for $\tilde{g}(\boldsymbol{p}) \leq \tilde{g}_m - T$, where the coded components residing in the middle range; $g_m - T < \tilde{g}(\boldsymbol{p}) < g_m + T$, are neglected in painting LBP. The scalar value T isolates the coded components 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, too much T greatly narrows down the range of usable colors, and $T = 30$ ensures good balance for 8-bit coded components.

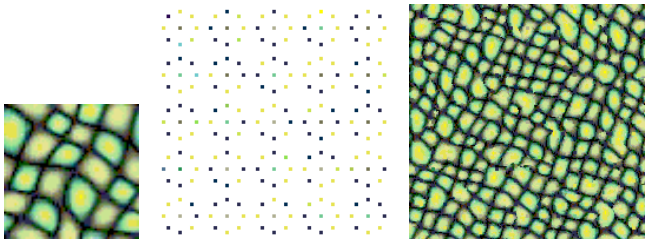
After classifying the constituent colors of the exemplar into three classes, the central pixel of a synthesized texture is always painted by the pixel color whose coded component belongs to the median class as $\boldsymbol{c}(\boldsymbol{p}_c) = \tilde{\boldsymbol{c}}(\boldsymbol{p}_m)$ where $\tilde{g}(\boldsymbol{p}_m) \equiv g_m$, and the surrounding pixels are painted by randomly selecting the pixel color whose coded component belongs to the upper and lower classes for the embedded binary data of 1 and 0, respectively.

We here assume that the process of texture generation is based on a stationary random field; the color distribution for each block is independent of its location, and this property is our ground for introducing a global threshold. Using the centered dots at every block to compute g_m seems to be redundant, but the local comparison of the coded components inside the block can enhance robustness in data detection. Color intensity of the captured image often varies depending on the location due to the change of illumination, and each locally estimated color at the center serves as the calibrated value of g_m for each block.

B. Screening of LBP colors

For increasing robustness against the positional error caused by image distortions, each dot for embedding data requires an area greater than the size of a single pixel. We therefore paint the same color on the nearby pixels (in our case, 8-link neighbors). Figure 4(b) represents examples of the LBP code for 5×5 blocks made from the exemplar in Figure 4(a).

The classified colors of an exemplar, however, become conspicuous on a synthesized image if their frequency band is higher than that of the LBP dots whose area contains 3 by 3 pixels. For this reason, high-frequency color components of the exemplar are removed from the upper or lower class. In practice, we remove the pixel colors if their outputs through a differential filter (we currently use a Sobel filter) exceed a threshold, where the threshold should be carefully determined so as to ensure a sufficient number of colors for each class. The effect of this screening is demonstrated in Figure 8.



(a) (b) (c)

Fig. 4. Examples of data-coded texture synthesis. (a) Exemplar (sample image), (b) LBP code for 25-byte data, (c) Synthesized texture.

V. Texture Synthesis with LBP

A. Texture coating with exemplar

The initially painted LBP code is concealed by adaptively determining the colors of unpainted pixels by referencing the features of the exemplar. For each unpainted pixel, we compute the differences in color $\boldsymbol{c}(\boldsymbol{p})$ of the nearby pixels with the corresponding pixels in the exemplar as

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

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

where $\boldsymbol{p}$ denotes the 2D position of the unpainted pixel, and ν is the set of offset vectors for indicating $N = (2w+1)^2 - 1$ neighbors centered at $\boldsymbol{p}$; $\nu := \{(s,t) | s,t \in \{-w, \ldots, -1, 0, 1, \ldots, w\}, (s,t) \neq (0,0)\}$. Notice that the $S(\boldsymbol{p}, \boldsymbol{q})$ decreases the more similar the synthesized texture and exemplar are in color distribution pattern for the nearby pixels of the $\boldsymbol{p}$ and $\boldsymbol{q}$ locations. We then substitute $\boldsymbol{c}(\boldsymbol{p})$ for the color at $\boldsymbol{q}$ of the exemplar, where $S(\boldsymbol{p}, \boldsymbol{q})$ takes the minimum value of:

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

The equations (2) and (3) are interpreted as template matching where a square context window of N pixels centering $\boldsymbol{p}$

is regarded as a template, and the best matching region is searched within the exemplar. Larger N can generate images more similar to the exemplar, but LBP code becomes more conspicuous as a side effect. We experimentally found that $w = 7(N = 224)$ ensures good balance of these factors.

The above procedure is repeated until all pixels are painted. Figure 4(c) shows the texture generated by using this coating algorithm from the exemplar of Figure 4(a) and the LBP code in Figure 4(b).

B. Effects of pixel-painting sequence

In pixel-based texture synthesis, the sequence of painting pixels plays an important role in the quality of the synthesized images, and existing methods usually select unpainted pixels in scan-line sequence for referencing the color coherency of previously painted pixels as much as possible. In our method, however, the sequence of scan-line often fails to conceal the LBP code. The colors near the LBP code reveal discontinuity because the top pixels used for pattern matching, marked A in Figure 5(a), have been coated in former iterations without the effect of the LBP code marked X, and even nearer pixels marked B have little effect from the LBP because of the smaller area inside the context window (notice that the Figure 5(a) schematically represents the window of $w = 2$ for simplicity, but we actually use $w = 7$). Consequently, the color inconsistency near the boundary of the LBP is unavoidable, which reveals our peculiar problem for using the LBP code as hard-constraints in generating textures.

We here introduce randomness to pixel-painting sequences (Figure 5(b)) for blurring the borders of a LBP dot. This random sequence can make the number of pattern-matched pixels uneven; the pixels near the LBP codes may be heavily influenced by the LBP when they are occasionally visited in an early stage of the coating process. A similar policy has been introduced in [8] for generating textures over surfaces of arbitrary topology.

This approach, however, seriously degrades the quality of the resulting image when the exemplar has structured patterns. The existing methods using sophisticated techniques, such as iterative optimizations or scalable pattern-matching [3], [6], have advantageous properties in accurately reproducing structured patterns in an exemplar. This reproducing capability, however, conflicts with the hard LBP constraints, so we must find a point of compromise considering this trade-off.

Our solution is to introduce a Hilbert curve sequence (Figure 5(c)), by which the sequence can be uniformly distributed while preserving the space coherency in painting order. This approach can be regarded as a reasonable compromise because the space coherency ensures the capability in reproducing structural patterns in the simplest way, similarly in the scan-line sequence, and its randomness of pixel-traversing directions can simultaneously bring about the effect of blurring to some extent.

Figure 6 shows the effects of pixel-painting sequences. The random sequence can conceal the LBP code better than a scan-line sequence, as shown in Figures 6(b) and 6(c), because it can blur the discontinuous borders of the LBP. However, it loses the distinctness of the pattern silhouette inherent in the exemplar as a side effect because of the lack of the reproducing capability. On the other hand, the Hilbert curve sequence has both the advantageous properties of blurring conspicuous spots and preserving silhouettes because it can irregularly paint pixels in series. This property is especially effective for the regular iterative pattern, as shown in Figures 6(h), 6(i), and 6(j).

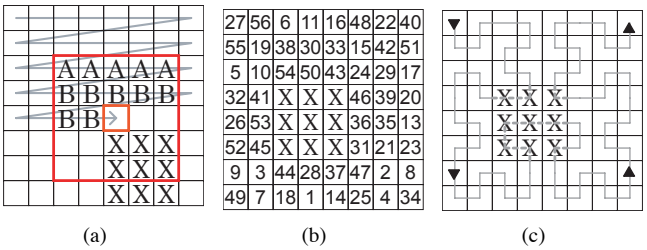


Fig. 5. Schematic representation of pixel-painting sequences: (a) scan-line, (b) random, and (c) Hilbert curve. Pixels painted for LBP code are marked X, and arrows and numbers represent the sequence of painting pixels.

C. Quality improvement with re-coating

The coating process determines the most similar pattern only with the painted pixels due to the rule of the dissimilarity computation in equation (2); $D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = 0$ if $\boldsymbol{c}(\boldsymbol{p}+\boldsymbol{r})$ is null. However, this causes inaccurate pattern analysis in the early stage of coating when we introduce randomness in pixel-painting sequences. This property is useful in blurring borders of the LBP, but damages the image quality at the regions distant from the LBP as a side effect. To eliminate this defect, we re-coat all pixels except for those on the LBP codes. In this second coating phase, all pixel colors have been tentatively determined, and thus the texture pattern analysis becomes more accurate. As a result, the quality of the synthesized images can be improved. We experimentally confirmed that the re-coating in more than the second trial cannot particularly increase the quality of the image, and therefore we do the re-coating only once. Notice that the effect of the pixel-painting sequence is negligible in the re-coating phase, and we therefore use a scan-line sequence.

Figures 6(e), 6(f), 6(k), 6(l) demonstrate the improvement of image quality through the re-coating process. As these examples show, the effects of re-coating enhance the advantageous property of their painting sequences; the random sequence makes the spots of LBP code more inconspicuous and the Hilbert curve sequence recovers the silhouette more clearly and correctly after the re-coating process.

D. Effects of LBP and randomness

The texture image is synthesized so as to have the features of an exemplar, but the features in frequency components are largely affected by the density of the LBP code. Figure 7 demonstrates the effect of the amount of the coded data where Figures 7(b) and 7(c) denote the LBP for 25- and 100-byte data, respectively, from the same exemplar in Figure 7(a). Figures 7(d) and 7(e) show the synthesized textures from the

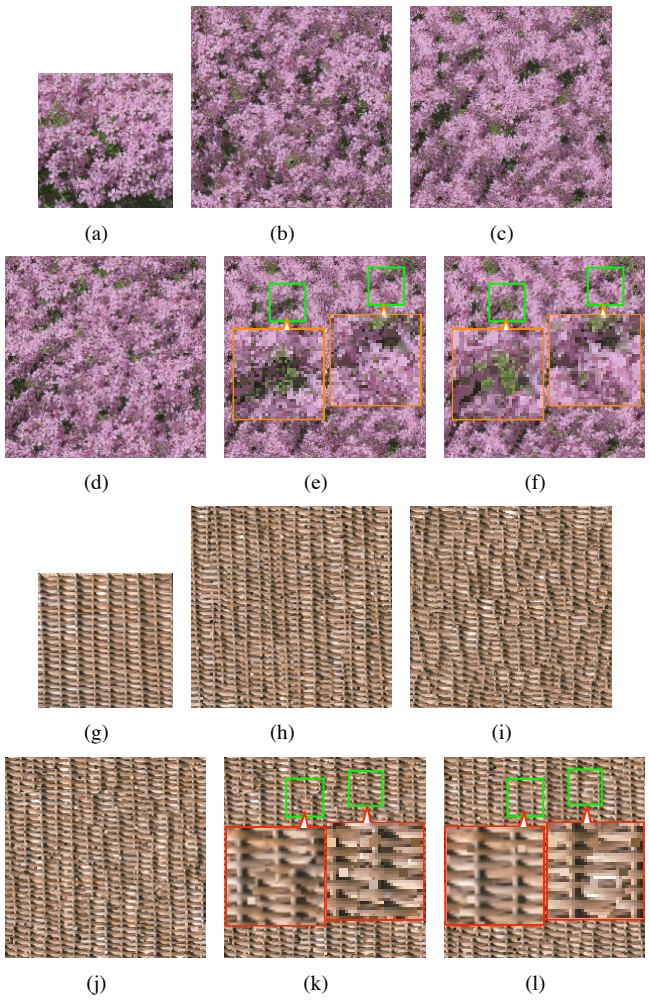


Fig. 6. Effects of pixel-painting sequences and re-coating for the exemplars of (a) random and (g) regular patterns. Texture images in (b) and (h) are synthesized using scan-line sequence, and (c),(i) and (d),(j) are synthesized using random and Hilbert curve sequences, respectively. Magnified images (e),(k) show intermediate defective patterns before re-coating and (f),(l) show their improved images after re-coating, where (e),(f) use a random sequence and (k),(l) use a Hilbert curve sequence.

LBP in Figures 7(b) and 7(c), respectively, which reveals the change in the frequency pattern. Figure 8 shows the effects of screening LBP colors, where the conspicuous spots in Figure 8(b) are successfully removed in Figure 8(c) by excluding tiny dark colors in the exemplar with our color screening.

The randomness (or larger entropy) residing in an exemplar plays an important role. Figure 9 (b) shows the capability of the existing method [1] based on the scan-line sequence in reproducing the structural feature of an exemplar of little randomness in Figure 9 (a). Our modified sequence with the Hilbert curve partially destroys the structure as a side effect of incorporating randomness in the pixel-visiting sequence, as shown in Figure 9 (c), and its data-embedding synthesis in Figure 9 (d) clearly shows the failure in concealing LBP codes. Notice that all syntheses used a larger context window ($w = 15$) for enhancing reproductive capability of the structure. These examples show that our encoding scheme is only suitable to random texture images, and suggest the difficulty in camouflaging LBP codes with general images.

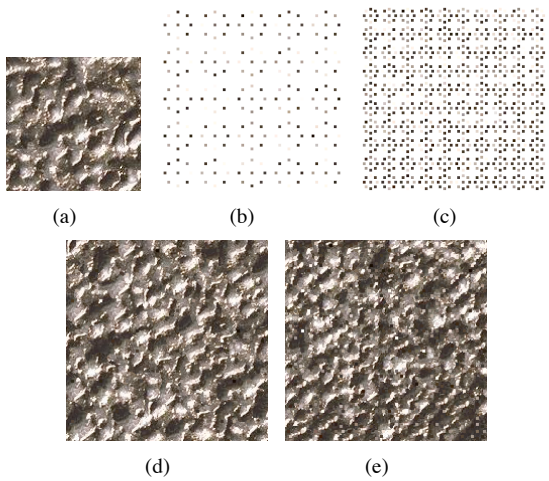


Fig. 7. Texture synthesis for various density of LBP code from the exemplar in (a). The 25 and 100 bytes data are embedded in (b) and (c), and the corresponding textures are shown in (d) and (e), respectively.

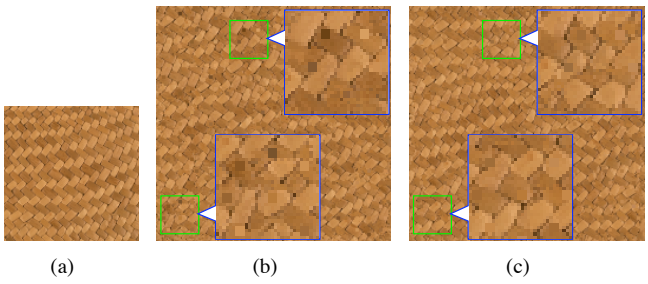


Fig. 8. The effect of screening LBP colors. The texture image in (b) is synthesized from the exemplar in (a) by randomly selecting the colors among the corresponding class, and (c) is synthesized using our color screening mechanism by which conspicuous spots can be removed.

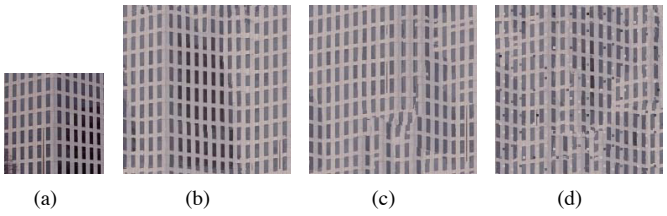


Fig. 9. The limitation of our method. The structural feature of an exemplar in (a) is reproduced in (b) synthesized with the scan-line sequence and is partially destroyed in (c) with the Hilbert curve sequence due to its randomness in pixel-visiting directions. Figure (d) reveals the difficulty in camouflaging the LBP code for the exemplar of small entropy.

VI. Experimental Results

A. Implementation on Mobile Phone

We have investigated the robustness of our data-detecting mechanism by implementing it on a mobile phone of FOMA D904i with Doja API. The data-coded texture images of 200 by 200 pixels were printed in a 2-by-2 inch square region with an EPSON PX-G5100 color ink-jet printer on super-fine A4 paper. The printed image was then captured by the

phone's camera with a macro mode of 480 by 640 pixels supported by hand at a distance of 5 to 8 cm, where a fluorescent lamp was used for lighting. Through 10 trials for each 4 examples, we had average error bits of 0.29 where the maximum and minimum of the error bits was 6 and 0, respectively. Figure 10 shows snapshots of the process for detecting data, and an example of this process is demonstrated on a movie file which can be downloaded from our web page (http://www.val.ics.tut.ac.jp/otori/forCGAdemo.wmv).

(a) (b)

Fig. 10. URL detection with a mobile phone on the market. (a) Capturing image , (b) Detecting URL.

As a data detection pre-process, we have implemented an image calibration mechanism [9] that automatically detects four corners of a square region and corrects distortions through projective transformations with the four corners. We then extract coded components of all pixels and compute their upper and lower classes for dividing up each coded component of the LBP dots, where the positions of the dots are computed from the relative position of each block subdivided by predefined numbers.

Table I shows the detailed result of data detection for the images in Figures 4(c), 6(f), 7(d), and 7(e). Error rate represents the ratio of imperfect code detections and the average and maximum error bits were computed by the number of wrong bits for the 10 trials. The embedded data cannot be completely detected, especially in Figure 4(c), but the ratios of error bits are small enough to recover the information with some error correcting codes such as Reed-Solomon [10]. It is noteworthy that the amount of embedded data (payload) has little effect on the accuracy of detection.

TABLE I
ERROR RATE FOR VARIOUS SIZES OF EMBEDDED DATA.

Example	Payload (byte)	Error rate	# of error bits	
			Average	Maximum
Fig. 4(c)	25	0.5	1.7	6
Fig. 6(f)	25	0.1	0.1	1
Fig. 7(d)	25	0.2	0.2	1
Fig. 7(e)	100	0.1	0.1	1

Table II compares the payload of our method against the existing market products. We can only show a rough comparison due to the lack of detailed specification of the products, and these payloads cannot be accurately estimated because of the difference in measurement conditions in terms of paper-size, resolution of camera, lighting, and so on. Although the evaluation of payload lacks the accuracy, our method can embed a larger amount of data than existing techniques based on natural image modulations. On the other hand, the payload in our method is smaller than those in the 2D bar code, and current capacity is not suited to embed long messages.

TABLE II
COMPARISON OF PAYLOAD WITH MARKET PRODUCTS.

Product	Payload (byte)	Encoding scheme
Our method	$\sim$ 100	LBP code
QR-Code	$\sim$ 3000	2D bar code
Pasha-warp [11]	3	Frequency modulation
FP-code [12]	5	Color modulation

VII. DISCUSSIONS

We have proposed a novel texture synthesis for coding data with little aesthetic defect. An encoding scheme with a dotted pattern, like the matrix-formed pattern of QR-code, is suited to embed larger data than the existing scheme based on color or frequency modulation used in traditional watermarking or steganography, and can guarantee robustness in code detection. We have experimentally found that our LBP is also robust against the analog channel of a printer and digital camera.

We have improved the quality of data-embedded textures by introducing adaptive LBP color selection, new pixel-painting sequences, and re-coating mechanism. However, those heuristic approaches could be generalized by the optimization that can guarantee both similarities in local and global features.

The payload in our method surpasses those in existing techniques of natural image modulations, and its amount is sufficient to embed most kinds of URLs for sending anchor information of Web services. Although the payload of our method should be more accurately estimated, it has the potential to replace existing bar codes to aesthetically improve the appearance.

The conspicuousness of LBP dots can be reduced by relaxing the restriction on the constancy in color. This relaxation should ensure the robustness in code detection, and might require integrating optimizations for both painting processes in LBP and texture. Currently, some marks or lines must be drawn on a texture image for extracting the data-embedded square region, but in some cases, their appearance also should be visually concealed. It is also a challenging work to extend our target from 2D flat images to deformed images texture-mapped on a virtual 3D object.

ACKNOWLEDGMENT

This work was supported in part by Global COE Program: Frontiers of Intelligent Sensing, MEXT, Japan.

REFERENCES

[1] Wei, L.-Y. and Levoy, M.: Fast Texture Synthesis using Tree-structured Vector Quantization. Proceedings of SIGGRAPH 2000 (2000) 479-488
[2] Provos, N. and Honeyman, P.: Hide and Seek: An Introduction to Steganography. IEEE Security & Privacy, Vol.1. No.3. (2003) 32-44
[3] Dong, F. and Ye, X.: Multiscaled Texture Synthesis Using Multi-sized Pixel Neighborhoods, IEEE Computer Graphics and Applications, Vol.27, No.3 (2007) 41-47

[4] Efros, A. A. and Leung, T. K.: Texture Synthesis by Non-parametric Sampling, IEEE International Conference on Computer Vision, (1999) 1033-1038
[5] Ashikhmin, M.: Synthesizing natural textures. Symposium on Interactive 3D Graphics (2001) 217-226
[6] Kwatra, V. and Essa, I. and Bobick A. and Kwatra, N.: Texture optimization for example-based synthesis, ACM Transactions on Graphics, Vol.24, No.3 (2005) 795-802
[7] Mäenpää, T. and Pietikäinen, M.: Texture analysis with local binary patterns. Handbook of Pattern Recognition and Computer Vision 3rd ed. World Scientific (2005) 197-216
[8] Wei, L.-Y. and Levoy, M.: Texture Synthesis Over Arbitrary Manifold Surfaces. Proceedings of ACM SIGGRAPH 2001 (2001) 355-360
[9] Katayama, A. and Nakamura, T. and Yamamuro, M. and Sonehara, N.: New high-speed frame detection method: Side Trace Algorithm (STA) for i-appli on cellular phones to detect watermarks , ACM International Conference Proceeding Series Vol.83. (2004) 109-116
[10] Reed, I.S. and Solomon, G.: Polynomial codes over certain finite fields. SIAM J (1960) 300-304
[11] Nakamura, T. and Ogawa, H. and Tomioka, A. and Takashima, Y.: Improved Digital Watermark Robustness against Translation and/or Cropping of an Image Area, IEICE Trans. Fundamentals, Vol.E83-A. No.1. (2000) 68-76
[12] Noda, T. and Moroo, J. and Chiba, H.: Print-type Steganography Technology (in Japanese). Magazine FUJITSU 2006-5 Vol.57. No.3.(2006) 320-324

BIOGRAPHY

Hirofumi Otori is a PhD student in the department of information and computer sciences at the Toyohashi University of Technology, and a research assistant of Global COE of MEXT. His research interests include data coding techniques with computer-generated images and applications with mobile graphics in general. Contact him at otori@val.ics.tut.ac.jp.

Shigeru Kuriyama is a professor at the Toyohashi University of Technology, and an invited leader of the visualization team in Digital Human Research Center at Advanced Institute of Science and Technology. His research interests include mobile graphics technology and humanoid animations. Kuriyama received a Doctor of Engineering from the Osaka University. Contact him at kuriyama@ics.tut.ac.jp.