Paper

Mini-texture Tiling

YOSHINORI NAGAI[1] and TED MADDESS[2]

(Received 17 January 2007, revised 30 January 2007)

Synopsis: Any texture composed of dots or pixels can be reorganized by tiling with small texture patches, or mini-textures. Here we investigate mini-texture properties for constructing any type of texture, and mini-texture relationships of deterministic textures generated by recursive application of rules through gliders such as the so-called Box, Cross, Zigzag, Oblong, El, Wye, and Corners gliders, to reveal that the mini-texture tiling recovers isotrigon textures by appropriate selection of tiling. We also consider the concept of a mini-texture basis to implement rule-dynamics mini-texture tiling for reconstructing the original texture.

Key words: mini-texture, tiling, isotrigon, texture, Box glider, rule-dynamics.

1. Introduction

We consider a mini-texture tiling method to reconstruct textures. This study comes from our study of isotrigon textures [1, 2] in which the deterministic textures generated by the glider product rules [3–5] are compared with random and considering the number of possible mini-textures that organize the textures. The glider method yields textures that have a very low number of mini-textures compared to random textures [1]. Thus we speculate that a function-like relationship between mini-textures determines the entire texture generated by glider rules.

The mini-texture relationships yield a tiling method for each glider-generated texture. The mini-textures can be enumerated using state values. The number of possible mini-texture variants is L^q, where L is the number of brightness levels and q the number of pixels comprising the mini-texture. Thus, the complete mini-texture space increases its number of elements exponentially. We therefore also investigate a procedure to reduce the number of mini-textures in the mini-texture set, using properties including symmetry operations and brightness level swapping. These materials are considered in section 2 for 3×3 mini-textures.

Mini-texture tiling to synthesize textures is considered in section 3. In particular we investigate a mini-texture tiling method for so-called isotrigon textures generated by the glider rules such as Box, Cross, Zigzag, Oblong, El, Wye, and Corner [1]. The mini-texture tiling for isotrigon textures shows another classification of glider generated textures, namely, three groups are found that are organized: {Box, Oblong, El, Wye, Corner}, {Zigzag}, and {Cross, Wolfram}. Mini-texture tiling requires the mini-texture numbers generated by the deterministic procedure and considers neighboring mini-textures. To determine the target mini-texture number we don't need to know all the pixel states of the mini-textures. Only

[1] Center for Information Science, Kokushikan University, and [2]ARC Centre of Excellence in Vision Science and Centre for Visual Sciences, RSBS, Australian National University.

some of the pixel states are required for each mini-texture in order to determine the target mini-texture. Thus it is expected that uncertainty exists in the decision of the target mini-texture. This is true when one uses all elements of the mini-texture space and also reduced subsets from the entire mini-texture set. However the textures are determined uniquely as long as the mini-texture set provided is drawn from the mini-texture set created by a given glider.

Since the textures generated by the Box glider can be created by the outer product of two vectors, a simple procedure to determine the target mini-texture is obtained without usage of the mini-texture set. It is known from the modified procedure of Box type mini-texture tiling that the Box-glider type of textures are essentially generated by the vector product of two vectors. Thus we found somewhat modified Box textures with this method. Those are shown in section 3.

The procedure of mini-texture tiling is equivalent to determining how to change which mini-texture will be located at a specified region within a texture. The procedure outlined here implies rule-dynamics [6, 7] for organizing textures. Taking our studies on rule-dynamics [6, 7] account into, we prepare a mini-texture basis and set mini-textures of the basis using texture field information. Thus the input mini-textures are changed locally. This means that here we use the Box-glider rule, and there the Cross-glider rule, and so on. We therefore regard this procedure as glider rule dynamics. Thus we say the procedure is a rule-dynamics for the texture construction. The important material is the mini-texture basis. A basis for mini-textures is generated by a specified method, namely, the rule. Thus the concept of rules should be extended from the original rule-dynamics considered on the cellular automata [8]. These points are discussed in section 4.

2. Properties of 3×3 mini-textures

It is obvious that any texture can be reconstructed by tiling component mini-textures. The textures considered are organized by so-called pixels, and each pixel can take one of several brightness levels. In the present study, we consider two level textures called binary textures and three level textures called ternary textures. For simplicity, a texture forms a flat plane by arranging small sized pixels packed tightly on a two-dimensional space.

We describe textures in a mathematical way. Let T denote a texture consisting of μ by ν pixels. We use the letter M to represent a mini-texture in a set of mini-textures m_1, m_2, $\cdots$, m_q, namely $M \in \{m_1, m_2, \cdots, m_q\} \equiv \Omega_M$. *Note that* q is $L^{\alpha\times\beta}$, where $\alpha\times\beta$ denotes side-lengths of each mini-texture, hence the mini-texture size, and L signifies the number of brightness levels. The texture is reconstructed with $P\times Q$ sheets of mini-textures, that is,

$$T=\begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1Q} \\ M_{21} & M_{22} & \cdots & M_{2Q} \\ \vdots & \vdots & \ddots & \vdots \\ M_{P1} & M_{P2} & \cdots & M_{PQ} \end{pmatrix} = \oplus_{\langle IJ\rangle=(1,1)}^{(P,Q)} M_{ij},$$

where M_{ij} is a mini-texture element of the mini-texture set Ω_M.

Here we investigate textures divided into mini-textures consisting of 3 by 3 pixels. These mini-textures have nine pixels so that the total number of possible mini-texture variants is L^9. The binary variants of 3×3 mini-textures are 256 and the ternary variants of those 19683 as shown in table I. For ease of consideration, fewer mini-texture variants are better. We therefore consider a reduction of mini-texture variants by a method where symmetric mini-textures are regarded as the same, two mini-textures are considered to be are the same when a transform of a mini-texture M yields another mini-texture M^* ($M^*=f(M)$). We include reduction methods such as rotational variants of pixels states (state rotational variants), black-white reversal (B–W reverse), symmetry with respect to up and down (U–D symmetry, the symmetry with respect to left and right (L–R symmetry), spatially π rotational equivalents (π space rotational symmetry variants), and spatially $\pi/2$ rotational equivalents ($\pi/2$ space rotational symmetry variants). We briefly explain below how to obtain the reduced number of mini-texture variants using symmetry properties, or other kinds of transformations,

State rotation:

The state rotation is simplest. In the binary case, it is identical to B–W reversing. The ternary state rotation means B→G→W so that the reduced number is one to third of entire mini-texture number.

Black–white reverse:

In binary mini-texture variants, the variant count reduces to which pixels are white by changing the chosen pixels until 4 of nine pixels in the 3×3 mini-textures, that is,

$$\#_{binary}={}_9C_0+{}_9C_1+{}_9C_2+{}_9C_3+{}_9C_4=256.$$

The ternary case becomes complicated, but it can be counted. Firstly we choose the gray pixels from 0 to 9 and then set the remained pixels black or white. Thus, the amount count of the number of mini-texture variants becomes follows:

1. No gray pixel: ${}_9C_0\times\#_{binary}=256$.
2. One gray pixel: ${}_9C_1\times({}_8C_0+{}_8C_1+{}_8C_2+{}_8C_3+\frac{1}{2}\times{}_8C_4)=1188$
3. Two gray pixels: ${}_9C_2\times({}_7C_0+{}_7C_1+{}_7C_2+{}_7C_3)=2304$
4. Three gray pixels: ${}_9C_3\times({}_6C_0+{}_6C_1+{}_6C_2+\frac{1}{2}\times{}_6C_3)=2688$
5. Four gray pixels: ${}_9C_4\times({}_5C_0+{}_5C_1+{}_5C_2)=2016$
6. Five gray pixels: ${}_9C_5\times({}_4C_0+{}_4C_1+\frac{1}{2}\times{}_4C_2)=1008$
7. Six gray pixels: ${}_9C_6\times({}_3C_0+{}_3C_1)=336$
8. Seven gray pixels: ${}_9C_7\times({}_2C_0+\frac{1}{2}\times{}_2C_1)=72$
9. Eight gray pixels: ${}_9C_8\times{}_1C_0=9$
10. Nine gray pixels: ${}_9C_9\times0=0$

Table I Mini-texture variants for symmetric properties and state transformations

	Binary texture	Ternary texture
Total number of 3×3 mini-textures	$2^9=512$	$3^9=19683$
State rotational variants	Same as the B-W reversing (256)	$3^8=6561$
Black-white (B-W) reverse variants	$2^8=256$	See text 9621
Up-down (U-D) symmetric variants and Left-right (L-R) symmetric variants	$2^3\times2^3=64$	$3^3\times3^3=729$
π space rotational symmetry variants	$2\times2^2\times2^3-8=56$	$2\times3^2\times3^3-27=459$
$\frac{\pi}{2}$ space rotational symmetry variants	$2\times2\times2=8$	$3\times3\times3=27$

The number of ternary B-W reverse variants is given by summing up above numbers.

Up-down symmetry or left-right symmetry:

The number of up-down symmetry variants is the same number as that of left-right symmetry variants. So we explain how to count the number of variants for the left-right symmetry case. In left-right symmetric mini-textures, the left side and right side columns are the same as each other. Thus the independent columns are two so that the number of variants is $L^3\times L^3$.

π Space rotational symmetry:

There are two cases of π rotation, namely, column rotation and raw rotation. Since the counting procedures are the same for column and raw rotations, we interpret the column case. The center column is invariant for π rotation of the column so that top and bottom pixels take the same state. Therefore two independent pixels exist in the center column. The left and right columns have the opposite order of pixel states. Those columns are therefore the same as each other in the sense of up and down opposition. The variants given by π rotation include the $\pi/2$ rotational variants in the both cases of column and raw rotations so that the number of these variants should be removed. Thus the number of variants in the present case is $2\times L^2\times L^3-\#_{(\pi/2)rotational}$.

$\pi/2$ Space rotational symmetry:

$\pi/2$ rotational variants requires that four nearest neighbor pixels to the center pixel are the same and four second-nearest neighbor pixels to the center pixel are the same. The center can take arbitrary pixel states. Thus the obtained number of variants for this symmetry is $L\times L\times L$.

Those variants described above are tabulated in Table I.

3. Mini-texture tiling for isotrigon textures

In the present section, we consider a mini-texture method of reconstructing isotrigon textures by their glider rules. The gliders considered here are follows [1, 3]:

$$\begin{array}{llll} \begin{matrix} a & b & \circ \\ c & f & \circ \\ \circ & \circ & \circ \end{matrix} & \begin{matrix} \circ & a & \circ \\ b & \circ & c \\ \circ & f & \circ \end{matrix} & \begin{matrix} \circ & a & b \\ c & f & \circ \\ \circ & \circ & \circ \end{matrix} & \begin{matrix} a & \circ & b \\ c & \circ & f \\ \circ & \circ & \circ \end{matrix} \\ \textit{Box} & \textit{Cross} & \textit{Zigzag} & \textit{Oblong} \\ \begin{matrix} a & b & c \\ \circ & \circ & f \\ \circ & \circ & \circ \end{matrix} & \begin{matrix} \circ & a & \circ \\ b & c & \circ \\ \circ & \circ & f \end{matrix} & \begin{matrix} a & b & c \\ \circ & f & \circ \\ \circ & \circ & \circ \end{matrix} & \begin{matrix} a & \circ & b \\ \circ & \circ & \circ \\ c & \circ & f \end{matrix} \\ \textit{El} & \textit{Wye} & \textit{Wolfram} & \textit{Corner} \end{array}$$

The glider rule means that the state of pixel denoted by f is determined by using the three input pixels marked with a, b, and c, namely, the state of pixel f is the value of $f(a, b, c)$ that is called a discrete function or rule. The discrete function f for organizing isotrigon textures can have the following forms [2, 3]:

Binary textures: $f(a, b, c) = \pm abc$, *for binary* $\{-1, 1\}$

$$f(a, b, c) = a + b + c + I(\mathrm{mod}2), \quad \textit{for binary } \{0, 1\}$$

Ternary texture: $f(a, b, c) = \lambda_a a + \lambda_b b + \lambda_c c + I(\mathrm{mod}3)$, *for ternary* $\{-1, 0, 1\}$

where I takes $\{0, 1\}$ for binary case and $\{-1, 0, 1\}$ for ternary case, and the coefficients λ_a, λ_b, λ_c take the values -1 or 1.

The textures organized using one of above rules can be divided into 3×3 mini-textures. The important matter is to select the correct mini-textures when we synthesize textures by tiling the mini-textures. The mini-textures can be numbering by an integer assignment that is

$$\begin{aligned} z_{ij} = {} & L^0 \cdot S_{11}(M_{ij}) + L^1 \cdot S_{12}(M_{ij}) + L^2 \cdot S_{13}(M_{ij}) + L^3 \cdot S_{21}(M_{ij}) + L^4 \cdot S_{22}(M_{ij}) + L^5 \cdot S_{23}(M_{ij}) \\ & + L^6 \cdot S_{31}(M_{ij}) + L^7 \cdot S_{23}(M_{ij}) + L^8 \cdot S_{33}(M_{ij}) \end{aligned}$$

for 3×3 mini-textures M_{ij} located at the site (i, j). Note that z_{ij} denotes the mini-texture number in the entire mini-texture space of L brightness levels, and $S_{\alpha\beta}(M_{ij})$ means the state of a pixel at (α, β) in the mini-texture M_{ij}. Isotrigon textures use some pixels of neighboring mini-textures. The neighboring mini-textures contribute to the determination of target mini-textures as illustrated in Fig. 1 for gliders Box, Cross, Zigzag, Oblong, El, Wye, Wolfram, and Corner. Those gliders are classified into three groups from the viewpoint of the number of contributed mini-textures for determining the mini-texture number of the target texture. The group {Box, Oblong, El, Wye, Corner} requires three neighboring mini-textures located at top-left corner, top, and left. The second group {Zigzag} in Fig. 1 requires four neighboring mini-textures located at top, left, top-right corner and right, and the third group {Cross, Wolfram} needs five surrounding mini-textures located at left, top-left corner, top, top-right corner, and right. As seen from Fig. 1 some pixels in each required mini-texture are used in the determination of mini-texture number of the entire mini-texture space. For example, the mini-texture number of the target in the Zigzag glider texture is determined as follows,

$$\begin{aligned} z_{ij} = {} & L^0 \cdot f(S_{31}(M_{ij-1}), S_{32}(M_{ij-1}), S_{13}(M_{i-1j})) \\ & + L^1 \cdot f(S_{32}(M_{ij-1}), S_{33}(M_{ij-1}), S_{11}(M_{ij}) = f(S_{31}(M_{ij-1}), S_{32}(M_{ij-1}), S_{13}(M_{i-1j}))) \end{aligned}$$

$$+L^2\cdot f(S_{33}(M_{ij-1}),\ S_{13}(M_{i+1\,j-1}),\ S_{12}(M_{ij})=f(S_{32}(M_{ij-1}),\ S_{33}(M_{ij-1}),\ S_{11}(M_{ij})))$$
$$+L^3\cdot f(S_{11}(M_{ij})=f(S_{31}(M_{ij-1}),\ S_{32}(M_{ij-1}),\ S_{13}(M_{i-1\,j})),\ f(S_{12}(M_{ij}),\ S_{23}(M_{i-1\,j}))$$
$$+L^4\cdot f(S_{12}(M_{ij}),\ S_{13}(M_{ij}),\ S_{21}(M_{ij})+L^5\cdot f(S_{13}(M_{ij}),\ S_{11}(M_{i+1\,j}),\ S_{22}(M_{ij}))$$
$$+L^6\cdot f(S_{21}(M_{ij}),\ S_{22}(M_{ij}),\ S_{33}(M_{i-1\,j})+L^7\cdot f(S_{23}(M_{ij}),\ S_{21}(M_{i+1\,j}),\ S_{33}(M_{ij}))$$
$$+L^8\cdot f(S_{23}(M_{ij}),\ S_{21}(M_{i+1\,j}),\ S_{32}(M_{ij}))$$

where $S_{\alpha\beta}(M_{ij})$ is determined by neighboring mini-texture pixels following chain method,

$$S_{11}(M_{ij})=f(S_{31}(M_{ij-1}),\ S_{32}(M_{ij-1}),\ S_{13}(M_{i-1\,j}))$$
$$S_{12}(M_{ij})=f(S_{32}(M_{ij-1}),\ S_{33}(M_{ij-1}),\ S_{11}(M_{ij}))$$
$$S_{13}(M_{ij})=f(S_{33}(M_{ij-1}),\ S_{31}(M_{i+1\,j-1}),\ S_{12}(M_{ij}))$$
$$S_{21}(M_{ij})=f(S_{11}(M_{ij}),\ S_{12}(M_{ij}),\ S_{23}(M_{i-1\,j}))$$
$$S_{22}(M_{ij})=f(S_{12}(M_{ij}),\ S_{13}(M_{ij}),\ S_{12}(M_{ij}))$$
$$S_{23}(M_{ij})=f(S_{13}(M_{ij}),\ S_{11}(M_{i+1\,j}),\ S_{22}(M_{ij}))$$
$$S_{31}(M_{ij})=f(S_{21}(M_{ij}),\ S_{22}(M_{ij}),\ S_{33}(M_{i-1\,j}))$$
$$S_{32}(M_{ij})=f(S_{22}(M_{ij}),\ S_{23}(M_{ij}),\ S_{31}(M_{ij}))$$
$$S_{33}(M_{ij})=f(S_{23}(M_{ij}),\ S_{21}(M_{i+1\,j}),\ S_{32}(M_{ij})).$$

Thus we know that a simple description of the number of target mini-texture is better such as,

$$z_{ij}\equiv z(M_{ij})$$
$$=G(S_{31}(M_{ij-1}),\ S_{32}(M_{ij-1}),\ S_{33}(M_{ij-1}),\ S_{13}(M_{i-1\,j}),\ S_{23}(M_{i-1\,j}),\ S_{33}(M_{i-1\,j}),$$

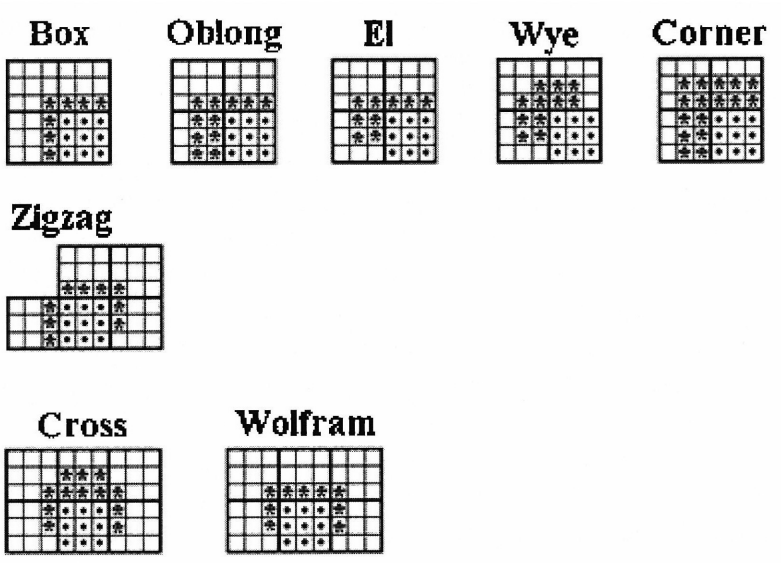

Fig. 1 Required mini-textures neighboring for glider rules.
There are three kinds of mini-texture tiling in the investigated glider, Box, Cross, Zigzag, Oblong, El, Wye, Wolfram, and Corners. Major class {Box, Oblong, El, Wye, Corner} requires three neighboring mini-textures, {Zigzag} requires four, and {Cross, Wolfram} needs five neighboring mini-textures.

$$S_{31}(M_{i+1\,j-1}),\ S_{11}(M_{i+1\,j}),\ S_{21}(M_{i+1\,j}))$$
$$=\Phi(M_{ij-1},\ M_{i-1\,j},\ M_{i+1\,j-1},\ M_{i+1\,j})$$

The method described above can reconstruct the original glider texture by tiling the specified mini-textures.

The Box glider has a simple method of vector products instead of the above method of determining the mini-texture numbers. Hereinafter of this section, we restrict our discussion to the Box glider for describing the mini-texture tiling method. For this purpose, we denote the mini-textures with raw vectors or column vectors in the following method,

$$M_{ij}=[\vec{v}_1(M_{ij})\ \vec{v}_2(M_{ij})\ \vec{v}_3(M_{ij})]=\begin{bmatrix}\vec{u}_1(M_{ij})\\ \vec{u}_2(M_{ij})\\ \vec{u}_3(M_{ij})\end{bmatrix}.$$

Then the target mini-texture in even Box texture is obtained by

$$M_{ij}(Box^{even})=S_{33}(M_{i-1\,j-1})\cdot[\vec{v}_1(M_{i-1\,j})\otimes\vec{u}_3(M_{ij-1})].$$

The above equation can recover the even Box texture as shown in Fig. 2. We also investigat-

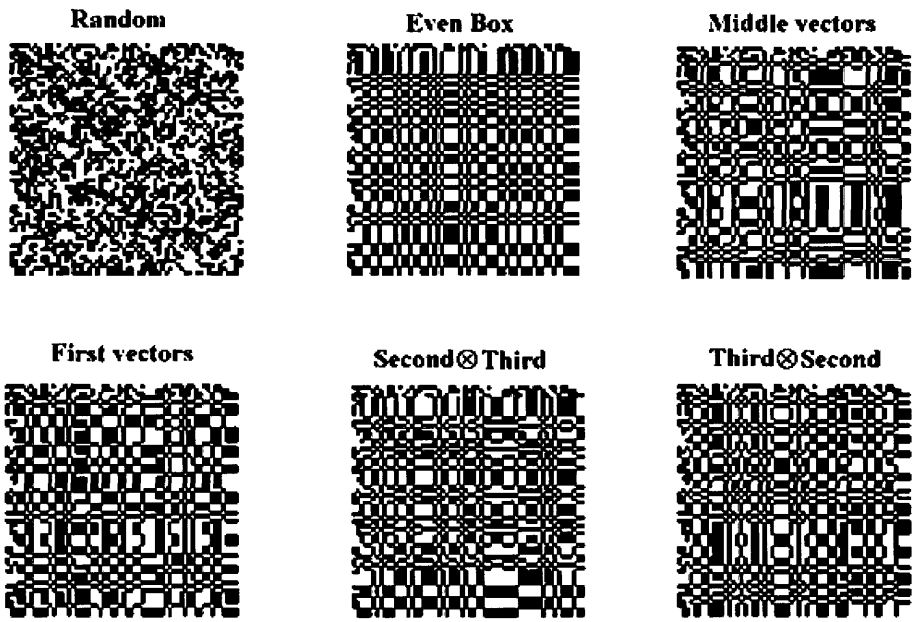

Fig. 2 Box glider mini-texture tiling and its modifications
The Box glider texture can recovered from the outer product of the third raw vector of the top neighboring mini-texture and third column vector of the left neighbored mini-texture. The vector product has the essential meaning of generating Box glider textures. The vector products using the other raw and column vectors generate the textures similar to Box glider textures. The vector product rule is applied to the random texture shown in the top-left texture. The Box glider texture can be recovered using vector product and sign of the ninth pixel of the mini-texture located at the top-left.

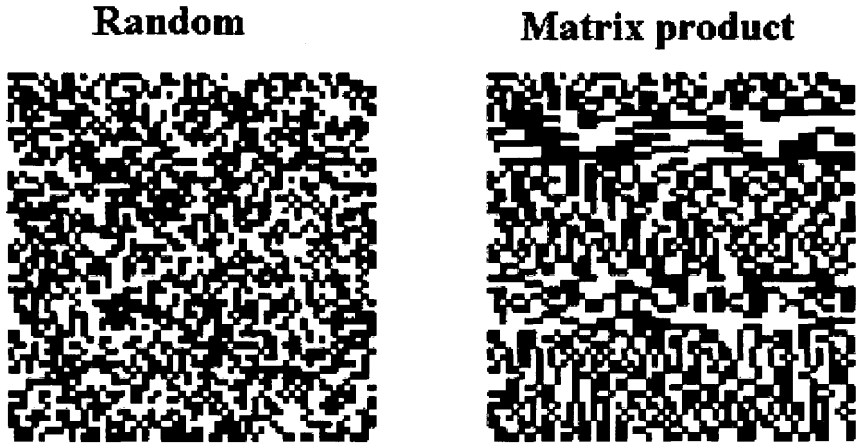

Fig. 3 Mini-texture tiling using the matrix product procedure
Matrix product is applied to determine the pixel states of target mini-textures. Three neighboring mini-textures just like the Box glider are used for the matrix product. The product procedure is (top-left mini-texture)⊗((top mini-texture)⊗(left mini-texture)).

ed the textures with mini-texture tiling using the other multiplied vectors, i.e., second vectors, third vectors, (second vector of v, third vector of u), and (third vector of v, second vector of u). The results are also seen in Fig. 2. Following those ideas, the vector product method using the two vectors from two different mini-textures, yields Box type textures that are somewhat disordered.

Moreover we investigate matrix products of mini-textures. The procedure is described with the following equation,

$$M_{ij}=M_{i-1\,j-1}\cdot(M_{ij-1}\cdot M_{i-1\,j}).$$

A tiled texture using the matrix product above is shown Fig. 3. The resulting texture becomes more ordered than the random texture that is applied for the mini-texture matrix product.

4. Mini-texture basis and rule-dynamical organizing of textures

In this section we consider rule-dynamics to understand the textures. Rule-dynamics is studied in the field of cellular automata [6]. Rule-dynamics can assist in understanding the complicated structure or materials since we found the real rule-dynamical systems in the subjects of neural networks [7], computers [8], and genetic systems [9]. If we apply the rule-dynamics of cellular automata to mini-texture tiling, it is better to consider mini-texture determination using three neighboring mini-textures, namely, the mini-texture tiling by the following rule,

$$M_{ij}=f(M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j}) \text{ or } f(M_{i-1,j-1}, M_{i,j-1}, M_{i+1,j-1}).$$

The function for mini-texture tiling should maintain causality. The functional form shown

above satisfies causality. The function applied to mini-texture tiling generates a set of mini-textures. The mini-texture tiling rules giving different mini-texture sets can organize the basis of mini-texture sets. A better choice of basis is that the selected basis can cover the almost all textures by the mini-texture tiling. The minimum reduced basis is best for organizing rule-dynamics for mini-texture tiling. We however don't know a better basis of the mini-texture tiling functions now. That theme is the subject of future study.

Several kinds of mini-texture sets are required when we establish a rule-dynamics on the mini-texture tiling. It is immediately known from Fig. 1 that the gilder group {Box, Oblong, El, Wye, Corner} provides an appropriate mini-texture tiling rule to organize the rule-dynamics for mini-texture tiling. Following the previous studies of rule-dynamics on cellular automata, a possible rule-dynamic for mini-texture tiling is

$$M_{ij} = \sum_{glider} \varepsilon_{glider}(M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j}) \Phi_{glider}(M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j}),$$

where $\varepsilon_{glider}(M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j})$ is the switching function turning on or off the mini-texture tiling function $\Phi_{glider}(M_{i-1,j-1}, M_{i,j-1}, M_{i-1,j})$ for the gliders.

Following the work of Nara et al. [10, 11], rule-dynamics can generate any pattern sequence when the corresponding rule sequence is applied for any initial patterns. This fact implies that the recognition of textures corresponds to finding a rule-sequence and regenerating the corresponding rule sequence. From this view point, only the generating method of the rule sequence is stored in memory to generate the textures. There is no requirement for storing the texture in the memory. This is an important property of rule-dynamics so that rule-dynamics may be useful to understand the brain functions.

References

[1] Maddess, T., Nagai, Y., Discriminating isotrigon textures, *Vision Res.* **41** (2001) 3837–3860.

[2] Maddess, T., Nagai, Y., Victor, J. D., Taylor, R. R. L., Multi-level isotrigon textures, *J. Opt. Soc. Am.* **A 24** (2007) 278–293.

[3] Maddess, T., Nagai, Y., James, A. C., Ankiewicz, A., Binary and ternary textures containing higher-order spatial correlations, *Vision Res.* **44** (2004) 1093–1113.

[4] Julesz, B., Gilbert, E. N., Victor, J. D., Visual discrimination of textures with identical third-order statistics, *Biological Cybernetics* **31** (1978) 137–140.

[5] Gilbert, E. N., Random colorings of a lattice on squares in the plane, *SIAM journal of Algebraic Discrete Methods* **1** (1980) 152–159.

[6] Aizawa, Y., Nagai, Y., Dynamics on pattern and rule –rule dynamics, *Bussei Kenkyu* **48** (1987) 316–320 (private communication in Japanese).

[7] Nagai, Y., Aizawa, Y., Rule–dynamical generalization of McCulloch–Pitts neuron networks, *BioSystems* **58** (2000) 177–185.

[8] Nagai, Y., Yamaguchi, H., Ichimura, A., Aizawa, Y., Ruledynamical nature of computer processes and communications, *Mem. Kokushikan U Cent. Infor. Sci.* **21** (2000) 62–77.

[9] Nagai, Y., Kito, M., Aizawa, Y., Rule dynamical property of genetic system, *Mem. Kokushikan U Cent. Infor. Sci.* **22** (2001) 51–64.

[10] Tamaru, T., Kuroiwa, J., Nara, S., Errorless reproduction of given pattern dynamics by means of cellular automata, *Phys. Rev.* **68E** (2003) pp.036707–1, 036707–8.

[11] Wada, M., Kuroiwa, J., Nara, S., Completely reproducible description of digital sound data with cellular automata, *Phys. Lett.* **306A** (2002) 110–115.