A Recurrent Fuzzy Coupled Cellular Neural Network System With Automatic Structure and Template Learning

Chun-Lung Chang, Kan-Wei Fan, I-Fang Chung, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—The cellular neural network (CNN) is a powerful technique to mimic the local function of biological neural circuits, especially the human visual pathway system, for real-time image and video processing. Recently, many studies show that an integrated CNN system can solve more complex high-level intelligent problems. In this brief, we extend our previously proposed multi-CNN integrated system, called recurrent fuzzy CNN (RFCNN) which considers uncoupled CNNs only, to automatically learn the proper network structure and parameters simultaneously of coupled CNNs, which is called recurrent fuzzy coupled CNN (RFCCNN). The proposed RFCCNN provides a solution to the current dilemma on the decision of templates and/or fuzzy rules in the existing integrated (fuzzy) CNN systems. For comparison, the capability of the proposed RFCCNN is demonstrated on the same defect inspection problems. Simulation results show that the proposed RFCCNN outperforms the RFCNN.**

***Index Terms*—Cellular neural network (CNN) template design, defect inspection, fuzzy clustering, fuzzy neural network.**

I. Introduction

THE two-dimensional (2-D) inputs and outputs of the cellular neural network (CNN) [1], [2] make it very suitable for image processing. Besides some basic image processing tasks, the CNN has been used to mimic the local function of biological neural circuits, especially the human visual pathway system [3]. A current biological study [4] shows that mammalian visual systems process the world through a set of separate parallel channels. Each subchannel can be regarded as a unique CNN. The output of these subchannels is then combined to form the new channel responses. As a result, it is widely accepted that using a set of CNNs in parallel can achieve higher level information processing and reasoning functions either from biologics or application points of views. Such an integrated CNN system can solve more complex intelligent problems.

For designing an integrated CNN system, in addition to the determination of a set of templates, another kernel problem is the way of integration. To solve this problem, the fuzzy inference system (FIS) can play an important role to integrate a set of CNNs into a system. To make a set of CNNs in parallel achieve higher level information processing, several integrated CNN systems are proposed [4]–[7]. They have two common characteristics. First, they all used many CNNs in parallel to solve a complex problem. Second, they all used FIS to make a decision. The common drawbacks of these approaches are that they all need to assign the corresponding templates of CNNs in advance (i.e., templates cannot be learned) and they all need to take the fuzzy rules manually by domain experts. Although, according to Nossek's survey [8], the template coefficients of a CNN can be found by design [8], [9] or by learning [8], [10], these techniques cannot be applied to the design or learning of an integrated CNN system directly.

To cope with these drawbacks, we proposed a novel framework for automatically constructing a multiple-CNN integrated neural system in the form of a recurrent fuzzy neural network (FNN) [11], [12]. This system, called recurrent fuzzy CNN (RFCNN) [13], can automatically learn its proper network structure and parameters simultaneously. The structure learning includes the fuzzy division of the problem domain and the creation of fuzzy rules and CNNs. The parameter learning includes the tuning of fuzzy membership functions and CNN templates. In the RFCNN, each learned fuzzy rule corresponds to a CNN. Hence, each CNN takes care of a fuzzily separated problem region, and the functions of all CNNs are integrated through the fuzzy inference mechanism. However, since the RFCNN only considers the learning of uncoupled CNN templates, its ability to solve more complex high-level machine vision problems is quite limited. In this brief, we extend our previously proposed RFCNN to automatically learn the proper network structure and parameters simultaneously of coupled CNNs. Due to the inclusion of coupled CNNs, the proposed recurrent fuzzy coupled CNN (RFCCNN) has shown more powerful abilities in detecting the fine detailed defects of color filters, compared with the previous RFCNN, in the simulations.

This brief is organized as follows. Section II describes the structure and functions of the proposed RFCCNN. Section III briefly describes the online learning algorithm for the RFCCNN. Section IV gives simulation results and discussions. Finally, conclusions are summarized in Section V.

II. Structure of the RFCCNN

Here, the structure of the proposed RFCCNN shown in Fig. 1 is introduced. For clarity, we consider a CNN, with time constant $= 1$, time step $= 1$, and neighborhood within a

Manuscript received July 5, 2005; revised November 10, 2005. This paper was recommended by Associate Editor H. Leung.

C.-L. Chang is with the Mechanical and System Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan 310, R.O.C.

C.-T. Lin and K.-W. Fan are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw).

I.-F. Chung is with the Institute of Bioinformatics, National Yang-Ming University, Taipei 112, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2006.876388

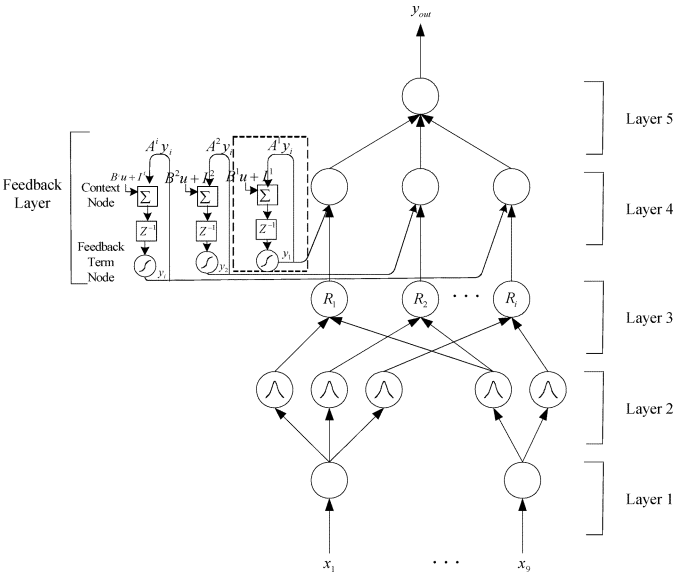


Fig. 1. Structure of the proposed RFCCNN.

radius $= 1$, which is characterized by the following templates:

$$A^i = \begin{bmatrix} a^i_{-1,-1} & a^i_{-1,0} & a^i_{-1,1} \\ a^i_{0,-1} & a^i_{0,0} & a^i_{0,1} \\ a^i_{1,-1} & a^i_{1,0} & a^i_{1,1} \end{bmatrix}$$
$$B^i = \begin{bmatrix} b^i_{-1,-1} & b^i_{-1,0} & b^i_{-1,1} \\ b^i_{0,-1} & b^i_{0,0} & b^i_{0,1} \\ b^i_{1,-1} & b^i_{1,0} & b^i_{1,1} \end{bmatrix}$$
$$I^i = z^i \tag{1}$$

where A^i, B^i, and Z^i are the feedback template, control template, and bias of the ith CNN, respectively. By defining a CNN as above, the six-layered RFCCNN network will realize a fuzzy model of the following form:

$$\text{Rule } i: \text{IF } x_1 \text{ is } M^i_1 \text{ and } \ldots x_j \text{ is } M^i_j \ldots \text{ and } x_9 \text{ is } M^i_9$$
$$\text{THEN } y_i(t+1) \text{ is } f'(A^i y_i(t) + B^i u(t) + z^i(t)) \tag{2}$$

or

$$\text{Rule } i: \text{IF } x_1 \text{ is } M^i_1 \text{ and } \ldots x_j \text{ is } M^i_j \ldots \text{ and } x_9 \text{ is } M^i_9$$
$$\text{THEN } y_i(t+1) \text{ is}$$
$$f'(a^i_{-1,-1}y_{i,1}(t) + a^i_{-1,0}y_{i,2}(t) + \cdots + a^i_{1,1}y_{i,9}(t) + b^i_{-1,-1}x_1(t) + b^i_{-1,0}x_2(t) + \cdots + b^i_{1,1}x_9(t) + z^i(t)) \tag{3}$$

where the current input vector is $u = \mathbf{x}_t = [x_1, \ \ldots, \ x_9]^T$, $A^i y_i(t)$ is $a^i_{-1,-1}y_{i,1}(t) + a^i_{-1,0}y_{i,2}(t) + \cdots + a^i_{1,1}y_{i,9}(t)$, $B^i u(t)$ is $b^i_{-1,-1}x_1(t) + b^i_{-1,0}x_2(t) + \cdots + b^i_{1,1}x_9(t)$, f' is a bipolar sigmoid function, M^i_j is a fuzzy set, and $a^i_{k,l}$, $b^i_{k,l}$, and z^i are consequent parameters representing feedback template, control template, and bias of the ith CNN, respectively. The RFCCNN is a six-layered network structure with one feedback layer and can automatically learn its proper network structure (the creation of fuzzy rules and CNNs) and parameters (the tuning of fuzzy membership functions and CNN templates) simultaneously. In this brief, as defined in (1)–(3), we extend the RFCNN to RFCCNN including the structure and parameter learning of coupled CNN cells. The six-layered network structure of the RFCCNN is mostly the same as that of the RFCNN, except for the feedback layer. In the feedback layer of the RFCCNN, the feedbacks are from coupled CNNs, i.e., $A^i y_i(t)$ is $a^i_{-1,-1}y_{i,1}(t) + a^i_{-1,0}y_{i,2}(t) + \cdots + a^i_{1,1}y_{i,9}(t)$, but, in the feedback layer of the original RFCNN, the feedbacks are from uncoupled CNNs only, i.e., $A^i y_i(t)$ is $a^i_{0,0}y_i(t)$. The details of the function of each node of the RFCNN are described in [13].

III. Learning Algorithms for the RFCCNN

Two types of learning, structure and parameter learning, are used concurrently for the RFCCNN. In the RFCCNN, the structure learning includes the fuzzy division of the problem domain (precondition structure identification) and the creation of fuzzy rules and CNNs (consequent structure identification). The precondition structure identification corresponds to the input-space partitioning. As to the consequent structure identification, the main task is to decide when to generate a new consequent term (or a new CNN) for the output variable. In our system, we proposed an online independent component analysis (ICA) mixture model [13] to realize the precondition and consequent structure identification part of the RFCCNN.

For the parameter learning, the parameters of each CNN template in the consequent parts are adjusted by the ordered derivative algorithm to minimize a given cost function. The parameters in the precondition part are adjusted by the online ICA mixture model algorithm. The RFCCNN can be used for normal operation at any time during the learning process without repeated training on the input–output patterns when online operation is required. There are no rules (i.e., no nodes in the network except the input–output nodes) in this network initially. They are created dynamically as learning proceeds upon receiving online incoming training data by performing the following learning processes simultaneously. The details of these learning processes are described in the remainder of this section.

A. Structure Learning Algorithm of RFCCNN

Efficient partition of input–output data will result in faster convergence and better performance for the RFCCNN. The most direct way is to partition the input space into grid types, with each grid representing a fuzzy if–then rule. This is called grid-based partitioning. The major problem of this kind of partitioning is that the number of fuzzy rules (and thus the number of CNNs) increases exponentially if the number of input variables or that of the partitions increases. A flexible partition method, such as the clustering-based approach which clusters the input training vectors in the input space, will reduce the rule and CNN numbers. In this brief, our proposed clustering method based on a new online ICA mixture model is used to provide a better partition of the input–output space for the proposed RFCCNN. The background and algorithm of the proposed online ICA mixture model for clustering can be found in [13] and [14].

B. Parameter Learning Algorithm of RFCCNN

After the network structure is adjusted according to the current training pattern, the network then enters the parameter identification phase to adjust the parameters of the network optimally based on the same training pattern. Notice that the following parameter learning is performed on the whole network after structure learning, regardless of whether the nodes (links)

are newly added or already existed. Since the RFCCNN is a dynamic system with feedback connections, the back propagation learning algorithm cannot be applied to it directly. Also, due to the online learning property of the RFCCNN, the offline learning algorithms for the recurrent neural networks, like back propagation through time and time-dependent recurrent back propagation [12], cannot be applied here. Instead, the ordered derivative [15], which is a partial derivative whose constant and varying terms are defined using an ordered set of equations, is used to derive our learning algorithm. The ordered set of equations, described in Section II in each layer, is summarized in (5)–(10). Our goal is to minimize the error function

$$E(t+1)=\frac{1}{2}[y_{\text{out}}(t+1)-y^d_{\text{out}}(t+1)]^2=\frac{1}{2}\varepsilon(t+1)^2 \quad (4)$$

where $y^d_{\text{out}}(t+1)$ is the desired output, $y_{\text{out}}(t+1)$ is the current output, and $\varepsilon(t+1)$ is $(y_{\text{out}}(t+1)-y^d_{\text{out}}(t+1))$. For each training data set, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output $y_{\text{out}}(t+1)$. In the following, dependency on time will be omitted unless emphasis on temporal relationships is required.

Summarizing the node functions defined in Section II, the function performed by the network is

$$y_{\text{out}}(t+1)=\sum_i u_i^{(5)} \quad (5)$$

$$u_i^{(5)}=o^{(4)}=o^{(6)}\cdot h^i \quad (6)$$

where

$$h^i=f[u_i^{(3)}]=\prod_i u_i^{(3)} \quad (7)$$

$$o^{(6)}=\frac{2}{1+e^{-2x^i}}-1 \quad (8)$$

$$x^i(t+1)=A^i_j y_i(t)+B^i_j u(t)+z^i(t) \quad (9)$$

and (1) is redefined as the following equation for clarity:

$$A^i_j=[a^i_1,a^i_2,a^i_3;a^i_4,a^i_5,a^i_6;a^i_7,a^i_8,a^i_9]$$
$$B^i_j=[b^i_1,b^i_2,b^i_3;b^i_4,b^i_5,b^i_6;b^i_7,b^i_8,b^i_9]. \quad (10)$$

With the above formula and the error function defined in (4), we can derive the update rules for the free parameters in the RFCCNN as follows.

The update rule of a^i_j (the parameter of feedback template of the ith CNN) is

$$a^i_j(t+1)=a^i_j(t)-\eta\frac{\partial^+E(t+1)}{\partial a^i_j} \quad (11)$$

$$\frac{\partial^+E(t+1)}{\partial a^i_j}=\frac{\partial E(t+1)}{\partial a^i_j}+\sum_k\frac{\partial E(t+1)}{\partial y_{\text{out},k}(t+1)}\frac{\partial^+ y_{\text{out},k}(t+1)}{\partial a^i_j}$$
$$=\frac{\partial E(t+1)}{\partial y_{\text{out}}(t+1)}\frac{\partial^+ y_{\text{out}}(t+1)}{\partial a^i_j} \quad (12)$$

where

$$\frac{\partial E(t+1)}{\partial y_{\text{out}}(t+1)}=\varepsilon(t+1) \quad (13)$$

$$\frac{\partial^+ y_{\text{out}}(t+1)}{\partial a^i_j}=\sum_k\frac{\partial y_{\text{out}}(t+1)}{\partial o^{(4)}_k}\frac{\partial^+ o^{(4)}_k}{\partial a^i_j}=\frac{\partial y_{\text{out}}(t+1)}{\partial o^{(4)}_i}\frac{\partial^+ o^{(4)}_i}{\partial a^i_j} \quad (14)$$

where

$$\frac{\partial y_{\text{out}}(t+1)}{\partial o^{(4)}_i}=\frac{\partial}{\partial o^{(4)}_i}\sum_k o^{(4)}_k(t+1)=1 \quad (15)$$

$$\frac{\partial^+ o^{(4)}_i}{\partial a^i_j}=\frac{\partial}{\partial a^i_j}[o^{(6)}_i(A^i_j y_i(t)+B^i_j u(t)+z^i(t))]h^i$$
$$=h^i(1+o^{(6)}_i)(1-o^{(6)}_i)\left[y_i(t)+a^i_j\frac{\partial y_i(t)}{\partial a^i_j}\right]. \quad (16)$$

Hence, the parameter a^i is updated by

$$a^i_j(t+1)=a^i_j(t)-\eta\varepsilon(t+1)$$
$$\times\left\{h^i(1+o^{(6)}_i)(1-o^{(6)}_i)\left[y_i(t)+a^i_j\frac{\partial y_i(t)}{\partial a^i_j}\right]\right\}. \quad (17)$$

Similarly, the parameter b^i_j (the parameters of control template of the ith CNN) is updated by

$$b^i_j(t+1)=b^i_j(t)-\eta\varepsilon(t+1)[h^i(1+o^{(6)}_i)(1-o^{(6)}_i)x_j(t)] \quad (18)$$

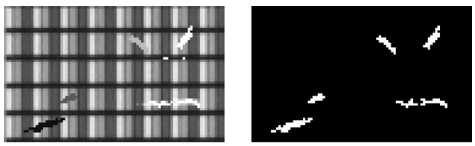
and the parameter z_i (the bias of the ith CNN) is updated by

$$z^i(t+1)=z^i(t)-\eta\varepsilon(t+1)[h^i(1+o^{(6)}_i)(1-o^{(6)}_i)]. \quad (19)$$

As shown in (14)–(16), the update rules are in recursive form. The value $(\partial^+ y/\partial a^i_j)$ is equal to zero initially. For the remaining free parameters in the RFCCNN, they are obtained during the structure-learning phase by the online ICA mixture model algorithm proposed in [13]. Notice that, according to the real-time recurrent learning (RTRL) scheme, we can also obtain the same parameter learning rules for the RFCCNN.

IV. Simulation Results and Discussions

The capability of the proposed RFCCNN is demonstrated on the real-world defect inspection problems. Here we are interested in the defect inspection of color filter, which is one of components in TFT-LCD array process and gives each pixel of LCD its own color. The difficulties in the defect inspection of color filter are its complex texture and demand for high-speed processing. For the reasons of high-speed processing, and that different kinds of defects in the color filter need different CNN templates and some complex defects cannot be detected by a single CNN, the proposed RFCCNN is a good alternative to detect defects in color filter images. To train the RFCCNN, we use a 3×3 window to get the system inputs and set the whole image as the inputs of the RFCCNN. The 3×3 window covers the central pixel and its eight connected neighbors. The training image and corresponding desired output are shown in Fig. 2(a) and (b). As mentioned in Section III, there are no rules (and no CNNs) in the RFCCNN initially. They are created dynamically as learning proceeds upon receiving online incoming training data by performing the learning processes. When the learning processes are done, six clusters (six fuzzy rules and CNN templates) were obtained. For the example of color filter, it takes approximately 90 s to learn the structure (interconnection set) and the parameters with a Pentium IV 2.0-G PC. However, the training can be done offline, so it is not a problem for the online processing of CNN. For simulation in this brief, it causes approximately 9 s. After the proposed RFCCNN is implemented by analog circuits in the



(a) (b)

Fig. 2. Training images. (a) Input image. (b) Desired output.

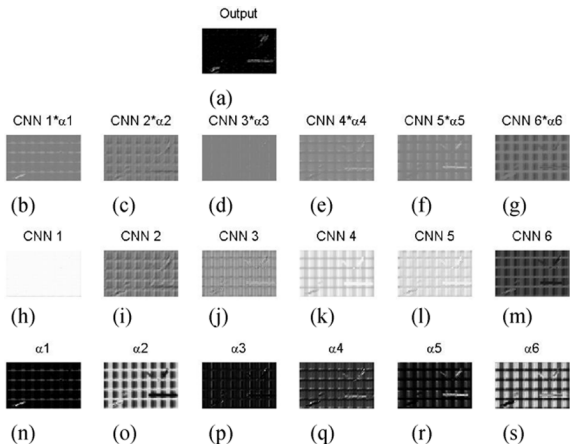


Fig. 3. The outputs of layers 3 and 4 and the feedback layer for the training image. (a) The output of the RFCCNN. (b)–(g) The outputs of the six layer-4 nodes, respectively. (h)–(m) The outputs of the six CNNs in the feedback layer, respectively. (n)–(s) The outputs of the six layer-3 nodes, respectively (firing strength of each rule).

future, we believe that the processing times will be much faster than those of the simulation.

Fig. 3 shows the output of RFCCNN and the outputs of layers 3 and 4 and the feedback layer for the training image. Fig. 3(a) shows the output of the RFCCNN. Fig. 3(b)–(g) shows the outputs of the six layer-4 nodes, respectively, i.e., the outputs of the six CNNs in the feedback layer multiplied by the outputs of the six layer-3 nodes (i.e., firing strength of each rule), respectively. Fig. 3(h)–(m) shows the outputs of the six CNNs in the feedback layer, respectively. Fig. 3(n)–(s) shows the outputs of the six layer-3 nodes, respectively (firing strength of each rule). The sum of the outputs of the six layer-4 nodes [i.e., Fig. 3(b)–(g)] forms the RFCCNN final output [Fig. 3(a)]. From Fig. 3(b)–(g), we can see that CNNs 4 and 5 take care of the defect texture in the right side of the training image, CNNs 1 and 6 mainly take care of the defect textures in the left side of the training image, and the other CNNs balance the output of the RFCCNN. The template of each learned CNN is given as

$$A^1 = \begin{bmatrix} -0.34 & -0.24 & 0 \\ -0.20 & 0 & 0.39 \\ 0.60 & 0.43 & 0.19 \end{bmatrix}$$

$$B^1 = \begin{bmatrix} 0.67 & 0.15 & -0.03 \\ -0.31 & -0.54 & 0.09 \\ 0.10 & -0.08 & 0.28 \end{bmatrix}, \; z^1 = 0.90$$

$$A^2 = \begin{bmatrix} -0.30 & -0.23 & 0 \\ -0.42 & -0.39 & -0.14 \\ 0 & -0.12 & 0 \end{bmatrix}$$

$$B^2 = \begin{bmatrix} -0.50 & -0.22 & -0.01 \\ -0.08 & 0.11 & 0.19 \\ -0.06 & 0.25 & 0.07 \end{bmatrix}, \; z^2 = 0.10$$

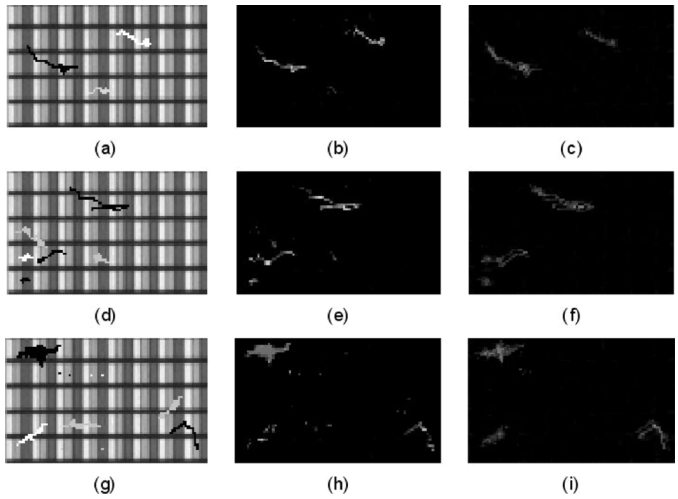


Fig. 4. Simulation (testing) results of the learned RFCCNN. (a), (d), (g) Input testing images. (b), (e), (h) Corresponding detection results of RFCCNN. (c), (f), (i) Corresponding detection results of an uncoupled RFCNN.

$$A^3 = \begin{bmatrix} -0.25 & -0.20 & 0.1 \\ -0.46 & -0.30 & -0.06 \\ -0.05 & -0.08 & -0.09 \end{bmatrix}$$

$$B^3 = \begin{bmatrix} -0.03 & 0.13 & -0.18 \\ -0.39 & 0.49 & -0.18 \\ 0.26 & 0.22 & 0.51 \end{bmatrix}, \; z^3 = -0.05$$

$$A^4 = \begin{bmatrix} -0.11 & 0.01 & 0.28 \\ -0.25 & -0.26 & -0.02 \\ -0.06 & -0.06 & 0.11 \end{bmatrix}$$

$$B^4 = \begin{bmatrix} 0.11 & 0.40 & 0.25 \\ -0.08 & 0.61 & 0.39 \\ 0. & 0.01 & 0.07 \end{bmatrix}, \; z^4 = 0.33.$$

$$A^5 = \begin{bmatrix} -0.30 & -0.34 & -0.03 \\ -0.27 & -0.16 & -0.01 \\ 0.16 & -0.10 & 0.18 \end{bmatrix}$$

$$B^5 = \begin{bmatrix} 0.64 & -0.10 & 0.27 \\ 0.12 & 0.90 & 0.09 \\ 0.08 & 0.54 & -0.15 \end{bmatrix}, \; z^5 = -0.14.$$

$$A^6 = \begin{bmatrix} -0.14 & -0.09 & 0.06 \\ -0.37 & -0.27 & -0.14 \\ 0.03 & 0 & 0.12 \end{bmatrix}$$

$$B^6 = \begin{bmatrix} 0.35 & -0.17 & -0.02 \\ -0.57 & -0.28 & -0.03 \\ -0.64 & -0.12 & -0.20 \end{bmatrix}, \; z^6 = -0.16. \quad (20)$$

Based on the learned structure and parameters of the RFCCNN, we test several images and three of those images as shown in Fig. 4. Fig. 4(a), (d), and (g) shows input testing images. Fig. 4(b), (e), and (h) displays the corresponding detection results of the RFCCNN. Fig. 4(c), (f), and (i) shows the corresponding detection results of the uncoupled RFCNN. From Fig. 4(b), (e), and (h), we can see that the learned structure and CNN templates of the RFCCNN are well suited to detect the defects of color filer images. It has also been confirmed that detection results are still good if the images are shifted or rotated. One of simulation results is shown in Fig. 5.

From Fig. 4, we can see some differences between the RFCCNN (coupled RFCNN) and the uncoupled RFCNN for

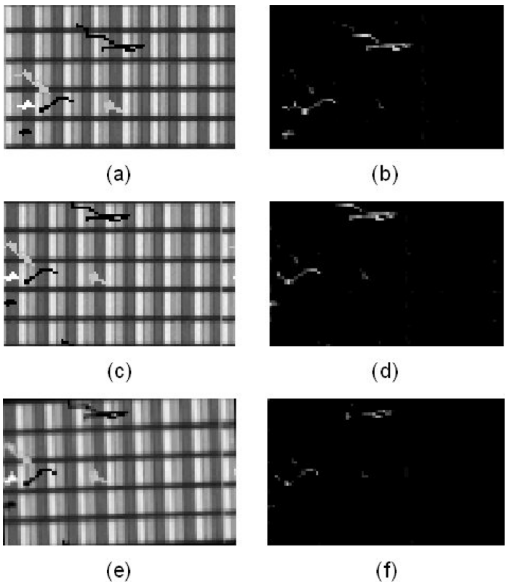


Fig. 5. Simulation results of shifted and rotated images. (a), (c), (e) Input testing images of original, shifted, and rotated ones, respectively. (b), (d), (e) Corresponding detection results of the RFCCNN.

TABLE I
COMPARISON OF DETECTION RATE

	Detection rate of the RFCCNN	Detection rate of the RFCNN
All defects	56%	43%
Large defects	76%	52%
Black defects	72%	51%

defect inspection of a color filter. First, as shown in Fig. 4(g)–(i), the results of the coupled RFCNN is better than those of uncoupled RFCNN for detecting the large defects on the top-left of test image shown in Fig. 4(g). Second, as shown in all test images and corresponding detection results, the results of the coupled RFCNN are better than those for the uncoupled RFCNN for detecting the black defects. The results of the coupled RFCNN are better for detecting the large defects in that the coupled RFCNN, like the coupled CNN, has fully output feedback and will take care of further neighboring pixels. Some quantity comparisons are shown in Table I. Here detection rate is defined as the ratio of detected defect pixel number to real defect pixel number. As we can see from Table I, the detection rates of the RFCCNN are better than those of the RFCNN, regardless of all defects, large defects, or black defects.

The main idea of the proposed RFCCNN is an integrated system of FIS and CNNs, which can construct fuzzy rules and CNN templates automatically. A scheme to implement the RFCCNN is described as follows. The circuit design technique of a multilayer CNN (MLCNN) [16] can be applied to implement the multilayer structure of the proposed RFCCNN model in Fig. 1. A CNN-based Gaussian function circuit as designed in [17] can realize the Gaussian membership function required in layer 2. The fuzzy logic operations in layers 3 and 4 can be realized by analog CNN circuits as studied in [7] and [18]. Two other layers correspond to the CNN's input and output, respectively. Therefore, it is very promising and feasible to implement the RFCCNN using high-speed analog circuits.

V. CONCLUSION

In this brief, we extended our previous work, called the RFCNN, from considering uncoupled CNNs to coupled CNNs, called the RFCCNN, for automatically constructing a multi-CNN integrated neural system. This CNN-based fuzzy neural network can automatically learn its proper network structure and parameters simultaneously. In order to verify the capability of the RFCCNN, a defect inspection problem has been demonstrated and compared with the RFCNN. The simulation results show that the performance of the proposed RFCCNN is better than that of the RFCNN. In addition, a scheme for hardware realization of the RFCCNN has been proposed. We believe that such an integrated CNN system, the RFCCNN, has potential to solve more complex intelligent problems, and our future work is to apply it to more complex real-world problems.

REFERENCES

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.
[2] ——, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1273–1290, Oct. 1988.
[3] J. Hámori and T. Roska, Receptive field atlas of the retinotopic visual pathway and some other sensory organs using dynamic cellular network models, Analogical and Neural Comput. Lab., Budapest, Hungary, MTA-SZTAKI, DNS-8-2000.
[4] L. Szatmári, D. Bálya, G. Tímár, C. Rekeczky, and T. Roska, "Multichannel spatio-temporal topographic processing for visual search and navigation," in *Proc. SPIE Microtechnologies New Millenium*, Gran Canaria, Spain, May 2003, vol. 5119-38 , pp. 297–306.
[5] F. Colodro and A. Torralba, "Cellular neuro-fuzzy networks (CNFNs) a new class of cellular networks," in *Proc. 5th IEEE Int. Conf. Fuzzy Syst.*, 1996, vol. 1, pp. 8–11.
[6] C. Rekeczky, T. Roska, and A. Ushida, "CNN-based difference-controlled adaptive nonlinear image filters," *Int. J. Circuit Theory Appl.*, vol. 26, pp. 375–423, Jul.-Aug. 1998.
[7] C. Rekeczky, Á. Tahy, Z. Végh, and T. Roska, "CNN based spatio-temporal nonlinear filtering and endocardial boundary detection in echocardiography," *Int. J. Circuit Theory Appl.*, vol. 27, pp. 171–207, Jan.-Feb. 1999.
[8] J. A. Nossek, "Design and learning with cellular neural networks," *Int. J. Circuit Theory Appl.*, no. 24, pp. 15–24, 1996.
[9] A. Zarandy, "The art of CNN template design," *Int. J. Circuit Theory Appl.*, no. 27, pp. 5–23, 1999.
[10] T. Kozek, T. Roska, and L. O. Chua, "Genetic algorithm for CNN template learning," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 40, no. 6, pp. 392–402, 1993.
[11] C. T. Lin, *Neural Fuzzy Control Systems with Structure and Parameter Learning*. New York: World Scientific, 1994.
[12] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[13] C. T. Lin, C. L. Chang, and W. C. Cheng, "A recurrent fuzzy cellular neural network system with automatic structure and template learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 5, pp. 1024–1035, May 2004.
[14] C. T. Lin, W. C. Cheng, and S. F. Liang, "An on-line ICA-mixture-model-based self-constructing neural fuzzy network," *IEEE Trans. Circuits Syst. Part I: Reg. Papers*, vol. 52, no. 1, pp. 207–221, Jan. 2005.
[15] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavior sciences," Ph.D. dissertation, DEPT. OF ???, Harvard Univ., Cambridge, MA, 1974.
[16] G. Liu and S. Oe, "Texture image segmentation method based on multilayer CNN," in *Proc. 12th IEEE Int. Conf. Tools with Artif. Intell. (ICTAI'00)*, 2000, pp. 147–150.
[17] S. A. Chen, J. F. Chung, S. F. Liang, and C. T. Lin, "Hybrid-order texture boundary detection based on CNN circuit models," in *Proc. IEEE Int. Workshop BioMedical Circuits Syst. (BioCAS'04)*, Singapore, 2004.
[18] M. Balsi and F. Voci, "Implementation of fuzzy rule based image processing on the CNN universal machine," in *Proc. Eur. Conf. Circuits Theory Des. (ECCTD)*, 1999, pp. 1167–1170.