Optical Engineering

SPIEDigitalLibrary.org/oe

Precise image alignment using cooperative neural-fuzzy networks with association rule mining–based evolutionary learning algorithm

Chi-Yao Hsu
Yi-Chang Cheng
Sheng-Fuu Lin

Precise image alignment using cooperative neural-fuzzy networks with association rule mining–based evolutionary learning algorithm

Chi-Yao Hsu
Yi-Chang Cheng
Sheng-Fuu Lin
National Chiao-Tung University
Department of Electrical Engineering
1001 Ta Hsueh Road
Hsinchu, Taiwan 300
E-mail: sflin@mail.nctu.edu.tw

Abstract. Precise image alignment is considered a critical issue in industrial visual inspection, since it performs an accurate pose to the object in inspected images. Recently, image alignment based on neural networks has become very popular due to its performance at speed. However, such a method has difficulty when applied to the alignment of images on a large range of affine transformation. To address this, a cooperative neural-fuzzy network (CNFN) with association rule mining-based evolutionary learning algorithm (ARMELA) is proposed. Unlike traditional neural network–based approaches, the proposed CNFN utilizes a coarse-to-fine alignment procedure to adapt image alignment to a larger range of affine transformation. The proposed ARMELA combines the self-adaptive method and association rules selection method to self-adjust the structure and parameters of the neural-fuzzy network. Furthermore, L2 regularization is adopted to control ARMELA such that the convergence speed increases. Experimental results show that the performance of the proposed scheme is superior to the traditional neural network methods in terms of accuracy and robustness. [DOI: 10.1117/1.OE.51.2.027006]

Subject terms: cooperative neural-fuzzy network; association rule mining; self-adaptive method; L2 regularization.

Paper 110878 received Jul. 25, 2011; revised manuscript received Dec. 17, 2011; accepted for publication Dec. 27, 2011; published online Mar. 12, 2012.

1 Introduction

Among industrial applications, such as object orientation, automatic visual inspection, assembly automation, and robotic machine vision, image alignment is the most widely used. For instance, in an industrial inspection system, objects appearing in images are not always aligned with the desired position or orientation. Therefore, an accurate geometric transformation of image alignment is desirable.

The problem of precise image alignment has been well studied in several fields. In Ref. 1, Liu et al. point out that image alignment techniques are broadly classified as feature-based[2,3] and area-based matching approaches.[4–6] In feature-based methods, the features of the sensed image and reference image represented by salient control points are usually detected first, and then the correspondence between these two images can be found by matching the descriptions of the features. Although features-based methods have been widely applied in many image registration tasks, they have difficulties to detect the respective features in less detailed images and are heavily dependent on control points. In area-based methods, such as correlation-like methods, they are popular for real-time applications because of simplicity and hardware implementation.[7,8] The main constraints of these methods are high computational complexity and are not suitable for complicated geometric deformation.

In Ref. 9, Amintoosi et al. pointed out that area-based methods produce better results than results with low signal-to-noise ratio (SNR) from feature-based methods. Moreover, Zitova and Flusser indicated[10] that area-based methods are preferably applied to less detailed images and that the captured images generally have less detail in an industrial inspection system. Therefore, the area-based methods are recommended in this paper.

In area-based fields, the most commonly used geometric transformation for image alignment is affine transformation, which consists of scaling, rotation, and translation. Next, global features of sensed images fed to a neural network are a widespread technique for estimation of affine transformation. In other words, neural networks are helpful in designing image alignment systems.

In recent years, neural network–based image alignment utilizing global features has been a relatively new research subject.[11–15] In Ref. 11, Elhanany et al. presented a feedforward neural network (FNN) to align images through 144 discrete cosine transform (DCT) coefficients as the feature vectors. Although such an approach has successfully aligned several deformed and noisy images, it still needs a larger dimension of feature vector to represent an image sufficiently in the unorthogonality of DCT-based space. To improve such approach, Wu and Xie[12] utilized low-order Zernike moments to replace DCT to estimate affine parameters. In their experiments, the alignment results were not satisfied. Recently, Xu and Guo[13] have adopted an isometric mapping (ISOMAP) method to reduce the dimension of the feature vector. For FNN improvement, Xu and Guo used a Bayesian regularization method to generalize the FNN.[14] They have shown in some comparative experiments that FNN with regularization indeed performs better than that without regularization. In addition to FNN-based methods, Sarnel et al.[15] used

0091-3286/2012/$25.00 © 2012 SPIE

a radial basis function neural network (RBFNN) to align images. According to their results, the training time of a RBFNN has been reduced, and the alignment accuracy and robustness against noise are better than those of FNN-based methods.

However, a major drawback of the existing neural network–based methods is that they have difficulty when applied to align images on a large range of affine transformation (i.e., large range of affine parameters). The reason is that a large range of affine parameters would lead to a large amount of training data such that the mapping surface becomes more complex. To solve such a phenomenon, a large-sized neural network is required, but this network is often difficult to train. Thus, applying a one-stage neural network to estimate a large range of affine parameters accurately is almost impossible.

In this paper, a cooperative neural-fuzzy network (CNFN) is proposed to overcome the problem produced by the one-stage neural network. The notion of this approach is to divide the large-sized network into several small cooperative networks, aiming to gradually reduce the image alignment error and finally obtain the desired accuracy. The cooperative network presented in this study indicates that each network manages a certain range of affine parameters and that the networks cooperate to adapt image alignment to a larger range of affine transformation. Such a phenomenon can be considered a coarse-to-fine alignment of the sensed image and the reference image. Moreover, based on the concept of the cooperative networks, this study proposes a self-organized training data–creating method that generates an appropriate training set for each network. The benefits of this method are that it not only provides a self-organized training set but also prevents the yielding redundant training data. Finally, this paper develops an association rule mining-based evolutionary learning algorithm (ARMELA) that combines self-adaptive method (SAM) and association rules selection method (ARSM) to tune the structure and parameters of the network automatically. Moreover, L2 regularization is utilized to control ARMELA such that the convergence speed increases. Therefore, these operations can make the structure and parameters of neural-fuzzy networks become more robust.

The rest of this paper is organized as follows. In Sec. 2, the proposed image alignment system is introduced. Section 3 describes ARMELA. The experimental results are presented in Sec. 4. The conclusion is presented in the last section.

2 The Proposed Image Alignment System

This section describes the proposed image alignment system that contains off-line and on-line procedures for training and executing the CNFN, respectively. Figure 1 illustrates the two procedures of the proposed approach, which will be explained in detail to show how the image alignment system works.

2.1 *Off-Line Procedure*

The objective of the off-line procedure is to train the CNFN. The four main parts of the procedure are creating synthesized training images, generating the Gabor-weighted gradient orientation histogram (WGOH) descriptor, yielding self-organized training data, and training the CNFN. These parts are described as follows.

2.1.1 *Creating synthesized training images*

The synthesized training images can be generated by applying various combination of translation, rotation, and scaling transformations within a predefined range. The transformation model is affine transformation, which is described by the following matrix equations:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = s \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 - x_c \\ y_1 - y_c \end{pmatrix} + \begin{pmatrix} x_c + \Delta x \\ y_c + \Delta y \end{pmatrix}, \tag{1}$$

where (x_1, y_1) is the original image coordinate, (x_2, y_2) is the transformed image coordinate, s is a scaling factor, $(\Delta x, \Delta y)$ is a translation vector, θ is a rotation angle, and (x_c, y_c) is the center of rotation.

2.1.2 *Generating Gabor-WGOH descriptor*

The main idea of a WGOH descriptor was inspired by a scale invariant feature transform (SIFT) descriptor.[16] It was introduced by Bradley et al.[17] to show its high speed. This descriptor calculates the orientation histograms within a region and uses the magnitude of the gradient at each pixel and the two-dimensional Gaussian function to weight the histogram.[18] However, using the pixel difference to compute the gradient is sensitive to noise. To avoid such sensitivity, Moreno et al.combined a Gabor filter with the

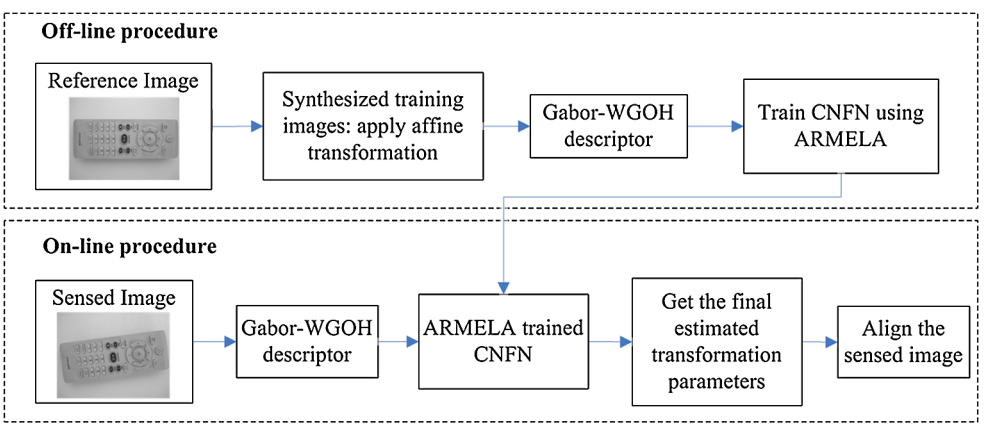


Fig. 1 Block diagram of the proposed image alignment system.

WGOH descriptor to suppress noise.[19] Thus, we adopt the Gabor-WGOH descriptor in representing an image.

To create the Gabor-WGOH descriptor, each image is split into 4×4 subimages. On each pixel of the subimage, the gradient magnitude and orientation are computed using the Gabor filter. Next, the 8-bin orientation histograms, which are weighted by the gradient magnitude and the Gaussian function, are calculated within each subimage. The 8-bin histograms of 16 subimages are then concatenated into a 128-element feature vector. To lower the dimension of the feature vector, we further employ the principal component analysis method to reduce the 128-elemet feature vector into a 33-element one. Therefore, each image can be represented by a 33-elemet feature vector.

2.1.3 *Yielding self-organized training data*

After describing the Gabor-WGOH descriptor, we propose a self-organized training data–creating method to provide an appropriate training data set for training neural networks. The major advantages of the proposed training data–creating method are that it can prevent the generation of the redundant data and supply a self-organized training data set for training a neural network efficiently. The steps for yielding the self-organized training data are as follows:

Step 1: First, generate a small training data set $\{S_{\text{train}}\}$. Then, utilize this data set to train a neural network.

Step 2: Input a fixed number of testing data sets $\{S_{\text{test}}\}$ into the neural network to create the alignment

$$\begin{aligned} &\text{alignment error}\{E_{\text{test}}\}.\text{Check each error}\{E_{\text{test}}(i)\}: \\ &\quad \text{If } E_{\text{test}}(i) > \text{PdError, then } \{S_{\text{test}}(i)\} \overset{\text{insert}}{\rightarrow} \{S_{\text{train}}\} \\ &\quad \text{and ErAcc} = \text{ErAcc} + 1. \\ &\quad \text{for } i = 1, 2, \ldots, N_{\text{test}}, \end{aligned} \tag{2}$$

where PdError is the predefined error, ErAcc is the accumulator of large error counts, and N_{test} is the number of the test data set.

Step 3: If $\text{ErAcc} < t_{\text{er}}$, then accumulate the Loop Num = Loop Num + 1. Otherwise, set Loop Num = 0. The symbol t_{er} indicates the threshold of the error accumulator, and LoopNum means the accumulating number of loop.

Step 4: If $\text{Loop Num} > \text{loop threshold} t_{\text{loop}}$ terminate the training and output the training set $\{S_{\text{train}}\}$. Otherwise, go to step 2 to run recursive training.

In Step 2, the insert testing data is the data that the neural network does not perform well. Therefore, inserting such data can enhance the learning ability of the neural network and prevent the selection of the redundant training data. Moreover, from Step 4, $\text{Loop Num} > t_{\text{loop}}$ means that the amount of training data set has converged. At this time, it also indicates that the training data set is self-organized. Thus, we can utilize the self-organized training data–creating method to provide the training data for training CNFNs.

2.1.4 *Training the CNFN*

The notion of the CNFN is to combine several networks to all cooperate in adapting to a large range of affine transformation. The aim of this operation is to improve the traditional one-stage neural network, which can cause a large amount of training data; such a network is difficult to train. The cooperative networks can be considered a coarse-to-fine alignment of the captured image and the reference image.

Figure 2 presents the process of the CNFN. Based on this figure, each stage deals with a certain range of affine parameters, and all the stages cooperate to obtain a large range of affine parameters. As an input image with an unknown pose, the CNFN gradually reduces the pose difference between the input and the reference images. Thus, the final pose with respect to the reference image can be written as the following equation:

$$P_{\text{final}} = P_1 + P_2 + \ldots + P_N, \tag{3}$$

where P_1, P_2, and P_N indicates the estimated pose from first, second, and Nth stages of the neural network.

To perform training CNFN with providing the training data, this study proposes an ARMELA to accomplish it. In CNFN, once the range of affine parameters of each stage has been determined, each network can be trained independently. Thus, the learning process of each stage of CNFN is identical. Regarding this fact, only a one-stage ARMELA

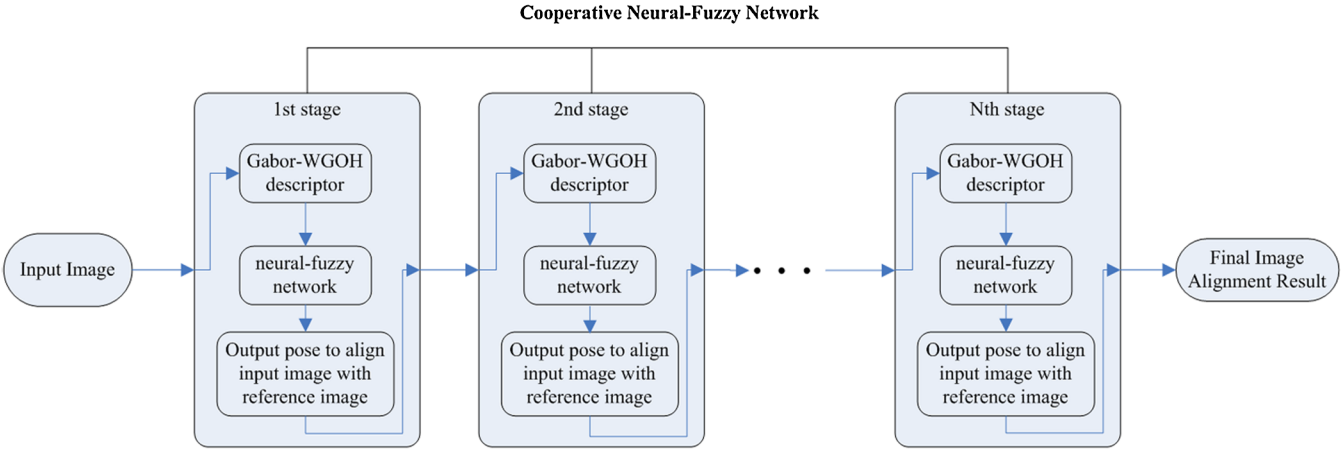


Fig. 2 Process of cooperative neural-fuzzy networks.

is discussed. The details of ARMELA are introduced in Sec. 3.

2.2 *On-line Procedure*

In the on-line phase, the sensed image is sent to the Gabor-WGOH descriptor to extract a feature vector and is then fed into ARMELA-trained CNFN to estimate the transformation parameters, which include the scaling factor s, rotation angle θ, and translation $(\Delta x, \Delta y)$, to be incorporated into aligning images. Specifically, the proposed CNFN performs N-stages of the neural-fuzzy network (Fig. 2) to align the sensed image with the reference image gradually. Thus, the image alignment error will be reduced by stage. Finally, the best aligning pose with the reference image will be obtained.

3 Association Rule Mining-based Evolutionary Learning Algorithm

In this work, ARMELA is based on a Takagi-Sugeon-Kang (TSK)-type neural-fuzzy network (TNFN)[20] employing a linear combination of the crisp inputs as the consequent part of a fuzzy rule. The structure of the TNFN is shown in Fig. 3. In the TNFN, the firing strength of a fuzzy rule is calculated by performing the "AND" operation on the truth values of each variable to its corresponding fuzzy sets by:

$$u_{ij}^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \tag{4}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(3)}$ are the outputs of the first and third layers, and m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i, respectively.

The output node of the fuzzy system integrates all of the actions recommended by the third and forth layers and acts as a defuzzifier with:

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)} \left(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i\right)}{\sum_{j=1}^{M} u_j^{(3)}}, \tag{5}$$

where $u^{(5)}$ is the output of the fifth layer, w_{ij} is the weighting value with ith dimension and jth rule node, and M is the number of a fuzzy rule.

After determining the structure of the neural-fuzzy network, we discuss further the learning process of ARMELA. Instead of multi-groups cooperation-based symbiotic evolution (MGCSE)[21] encoding the whole fuzzy rule into a chromosome, the proposed ARMELA encodes an antecedent part of a fuzzy rule into a chromosome. The consequent part of a fuzzy rule used in ARMELA is then estimated using L2 regularization. Such an operation not only reduces the number of parameters that must be trained but also increases the convergence speed. The following describes the details of the L2 regularization:

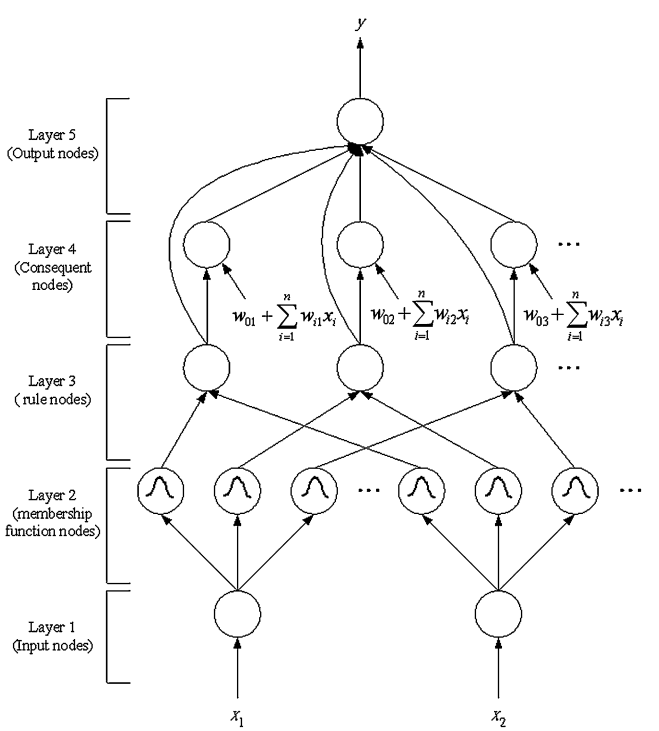


Fig. 3 TNFN structure.

3.1 *Regularization*

Assume a TSK-type neural fuzzy model composed of m fuzzy rules in the following form:

$$\begin{aligned} &R_j: \text{IF } x_1 \text{ is } A_1^j \ldots \text{and } x_n \text{ is } A_n^j, \quad \text{THEN} \\ &\quad y_j = w_o^j + w_1^j x_1 + \ldots + w_n^j x_n, \end{aligned} \tag{6}$$

where $j = 1, \ldots, m$ and A_i^j is the linguistic part with respect to input i and Rule j. From Eq. (6), the output can be written as follows:

$$y = \frac{\sum_{j=1}^{m} u_j y_j}{\sum_{j=1}^{m} u_j} = \hat{u}_1 y_1 + \hat{u}_2 y_2 + \ldots + \hat{u}_m y_m, \tag{7}$$

where u_j is the firing strength of Rule j, and $\hat{u}_j = u_j/(u_1 + \ldots + u_m)$. Thus, expressing the equation above into the following form is possible:

$$\begin{aligned} y = {} & \hat{u}_1(w_0^1 + w_1^1 x_1 + \ldots + w_n^1 x_n) + \ldots \\ & + \hat{u}_m(w_0^m + w_1^m x_1 + \ldots + w_n^m x_n) = aW, \end{aligned} \tag{8}$$

where $W = [W_1^T \ \cdots \ W_m^T]^T$, $W_j = [w_0^j \ \cdots \ w_n^j]^T, j = 1, \ldots m$ and

$$a = \begin{bmatrix} \hat{u}_1\hat{u}_1 x_1 & \cdots & \hat{u}_1 x_n \\ \hat{u}_2\hat{u}_2 x_1 & \cdots & \hat{u}_2 x_n \\ & \vdots & \\ \hat{u}_m\hat{u}_m x_1 & \cdots & \hat{u}_m x_n \end{bmatrix}^T.$$

As y and a are known values, the only unknown value is the consequent part W. Suppose a given set of training inputs and desired outputs is $\{x(t), y_d(t)\}_{t=1}^{M}$ Equation (8) can be rewritten as:

$$AW = Y_d, \tag{9}$$

where $A = [a(1)\, a(2) \ \cdots \ a(M)]^T$.

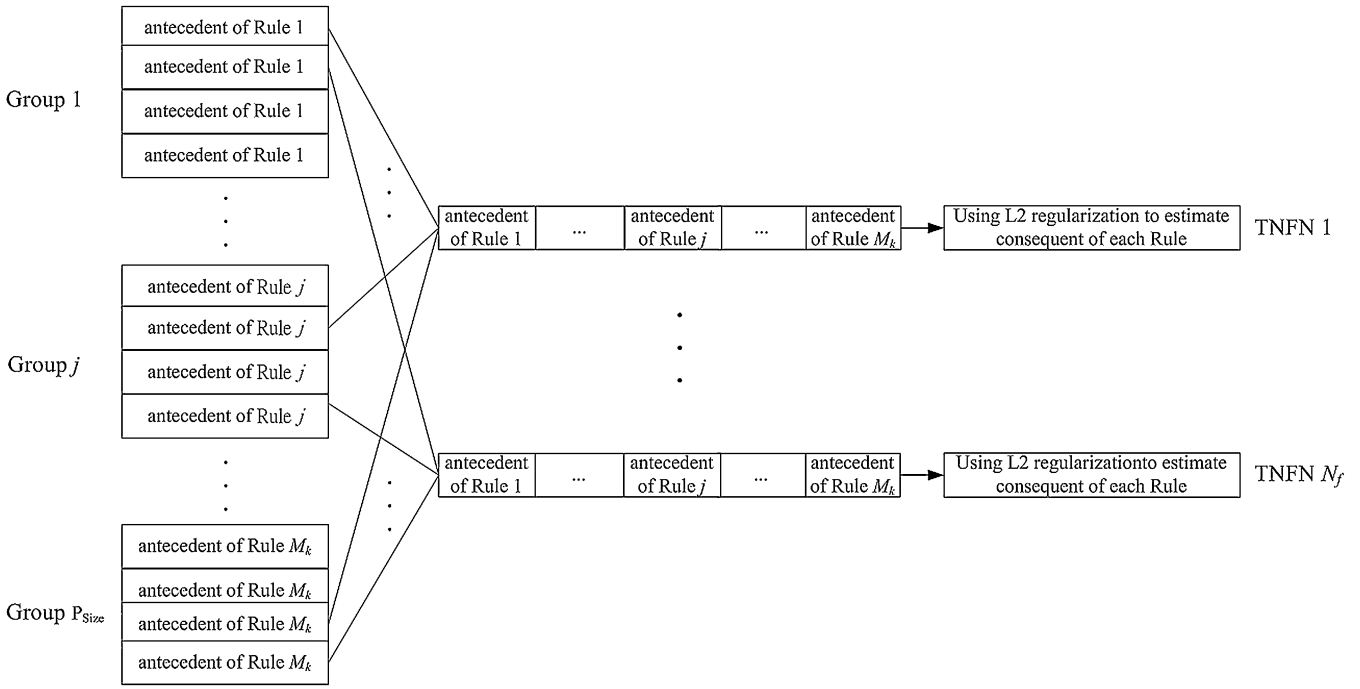


Fig. 4 Chromosome structure to construct the TNFN in ARMELA.

In general, in most cases of the proposed alignment system, the number of training sets (i.e., M) is always much greater than the dimensions of each training picture. Regarding this fact, Eq. (9) is always an overdetermined system. Thus, a least-square method can be utilized to obtain an approximate solution. However, to obtain the smooth estimation, the regularization is adopted. Thus, this method is called the L2 regularization. By using L2 regularization, the approximation solution is obtained as follows:

$$\hat{W} = (A^T A + \lambda I)^{-1} A^T Y_d, \tag{10}$$

where λ is a regularization parameter that adjusts the smoothness. Therefore, by obtaining Eq. (10), we complete the estimation of the consequent part of the fuzzy rules.

The chromosome structure to construct the TNFNs in ARMELA is discussed and shown in Fig. 4. In this figure, each antecedent part of a fuzzy rule represents a chromosome selected from a group, P_{size} denotes that there are P_{size} groups in a population, and M_k indicates that there are M_k rules used in the TNFN construction.

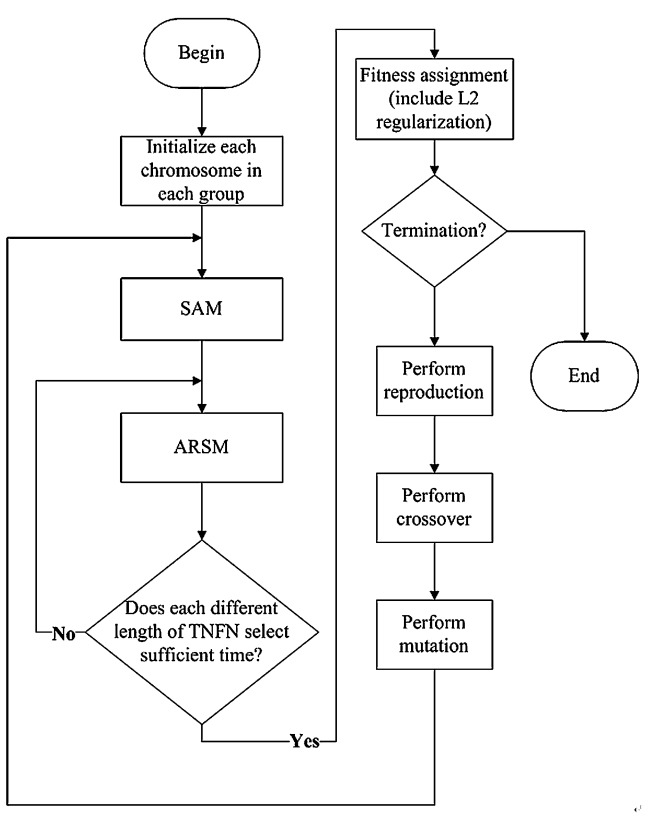


Fig. 5 Learning process of ARMELA.

3.2 *ARMELA Procedure*

The evolutionary process of ARMELA in each group involves seven major operators: initialization, SAM, ARSM, fitness assignment, reproduction strategy, crossover strategy, and mutation strategy. Figure 5 presents the learning process.

The detailed learning processes of ARMELA are described as follows:

Table 1 Transactions in the ARSM.

Transaction index	Groups	Performance Index
1	1,4,8	g
2	2,4,7,10	b
. . .	. . .	. . .
Transaction Num	1,3,4,6,8,9	g

(a) (b)

Fig. 6 Example of visual inspection images: (a) reference image, (b) transformed image with a scale of 0.9, a rotation of 10 deg, a vertical translation of 5, and a horizontal translation of 10.

3.2.1 *Initialization*

Before ARMELA learning is applied, the initial groups of individuals should be generated. The initial groups of ARMELA are generated randomly within a fixed range. The following formulations show how to generate the initial chromosomes in each group:

$$\begin{aligned} &\text{Deviation}: \text{chr}_{g,c}[p] = \text{random}[\sigma_{\min}, \sigma_{\max}], \\ &\quad \text{where } p = 2, 4, \ldots, 2n; \quad g = 1, 2, \ldots, P_{\text{size}}; \\ &\quad c = 1, 2, \ldots, N_C, \end{aligned} \tag{11}$$

Table 2 Target alignment range.

Affine parameter	The range of affine parameter
Scale	[0.7 to 1.3]
Rotation (deg)	[−100 to 100]
Vertical translation (pixels)	[−100 to 100]
Horizontal translation (pixels)	[−100 to 100]

$$\begin{aligned} &\text{Mean}: \text{chr}_{g,r}[p] = \text{random}[m_{\min}, m_{\max}], \\ &\quad \text{where } p = 1, 3, \ldots, 2n-1, \end{aligned} \tag{12}$$

where $\text{Chr}_{g,c}$ represents cth chromosome in the gth group, N_C is the total number of chromosomes in each group, p represents the pth gene in a $\text{Chr}_{g,c}$ and $[\sigma_{\min}, \sigma_{\max}]$ $[m_{\min}, m_{\max}]$ represent the predefined range to generate the chromosomes.

3.2.2 *Self-adaptive method*

To select fuzzy rules automatically, the proposed ARMELA adopts our previous research (i.e., the SAM[22]) to determine the suitability of the TNFN models with different fuzzy rules. The SAM encodes the probability vector V_{M_k}, which stands for the suitability of a TNFN with M_k rules. In addition, in SAM, the minimum and maximum numbers of rules must be predefined to limit the number of fuzzy rules to a certain bound, that is, $[M_{\min}, M_{\max}]$. The processing steps of SAM are described as follows:

Step 1: Update the probability vectors V_{M_k} according to the following equations:

$$\begin{cases} V_{M_k} = V_{M_k} + (\text{Uptvalue}_{M_k} * \lambda), & \text{if Avg} \le \text{fit}_{M_k} \\ V_{M_k} = V_{M_k} - (\text{Uptvalue}_{M_k} * \lambda), & \text{otherwise} \end{cases} \tag{13}$$

Table 3 Affine parameters range of three-stage CNFNs.

Affine parameter	The coarse range of affine parameter	The medium range of affine parameter	The fine range of affine parameter
Scale	[0.7 to 1.3]	[0.85 to 1.15]	[0.9 to 1.1]
Rotation (degrees)	[−100 to 100]	[−50 to 50]	[−5 to 5]
Vertical translation (pixels)	[−100 to 100]	[−30 to 30]	[−5 to 5]
Horizontal translation (pixels)	[−100 to 100]	[−30 to 30]	[−5 to 5]

$$\text{Avg} = \sum_{M_k=M_{\min}}^{M_{\max}} \text{fit}_{M_k}/(M_{\max} - M_{\min} + 1), \tag{14}$$

$$\text{Uptvalue}_{M_k} = \text{fit}_{M_k}/\sum_{M_k=M_{\min}}^{M_{\max}} \text{fit}_{M_k}; \tag{15}$$

$$\text{if Fitness}_{M_k} \geq (\text{BestFitness}_{M_k} - \text{Thread Fitness value}) \tag{16}$$

$$\text{then fit}_{M_k} = \text{fit}_{M_k} + \text{Fitness}_{M_k},$$

where V_{M_k} is the probability vector, λ is a predefined threshold value, Avg is the average fitness value in the whole population, BestFitness_{M_k} is the best fitness value of TNFN with M_k rules, and fit_{M_k} is the sum of the fitness values of the TNFN with M_k rules.

Step 2: Determine the selection times of TNFN with different rules according to the probability vectors as follows:

$$Rp_{M_k} = (\text{SelectionTimes}) * (V_{M_k}/\text{TotalVelocy}), \quad \text{for } M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}, \tag{17}$$

$$\text{TotalVelocy} = \sum_{M_k=M_{\min}}^{M_{\max}} V_{M_k}, \tag{18}$$

where selection_Times is the total selection times in each generation, and Rp_{M_k} is the selection times of TNFN with M_k rules in one generation.

Step 3: Accumulator calculation: If the current best combination of chromosomes does not improve, then the accumulator can be computed as follows:

$$\text{if BestFitness}_g = \text{BestFitness}, \quad \text{then Accumulator} = \text{Accumulator} + 1, \tag{19}$$

where BestFitness_g is the best fitness value of the best combination of chromosomes in the gth generation, and Best_Fitness is the best fitness value of the best combination of chromosomes in the current generations.

3.2.3 *The association rule selection method*

Following the selection times determined by SAM, the selection steps, including the selection of chromosomes and groups, are performed. In chromosome selection, chromosomes are randomly selected from groups. In the group selection, this paper proposes the use of ARSM to determine the suitable groups for chromosome selection. To prevent the selected groups from falling into the local optimal solution, ARSM uses a transaction built action and an association rule mining action to select the well-performing groups. The details of ARSM are described in the following actions:

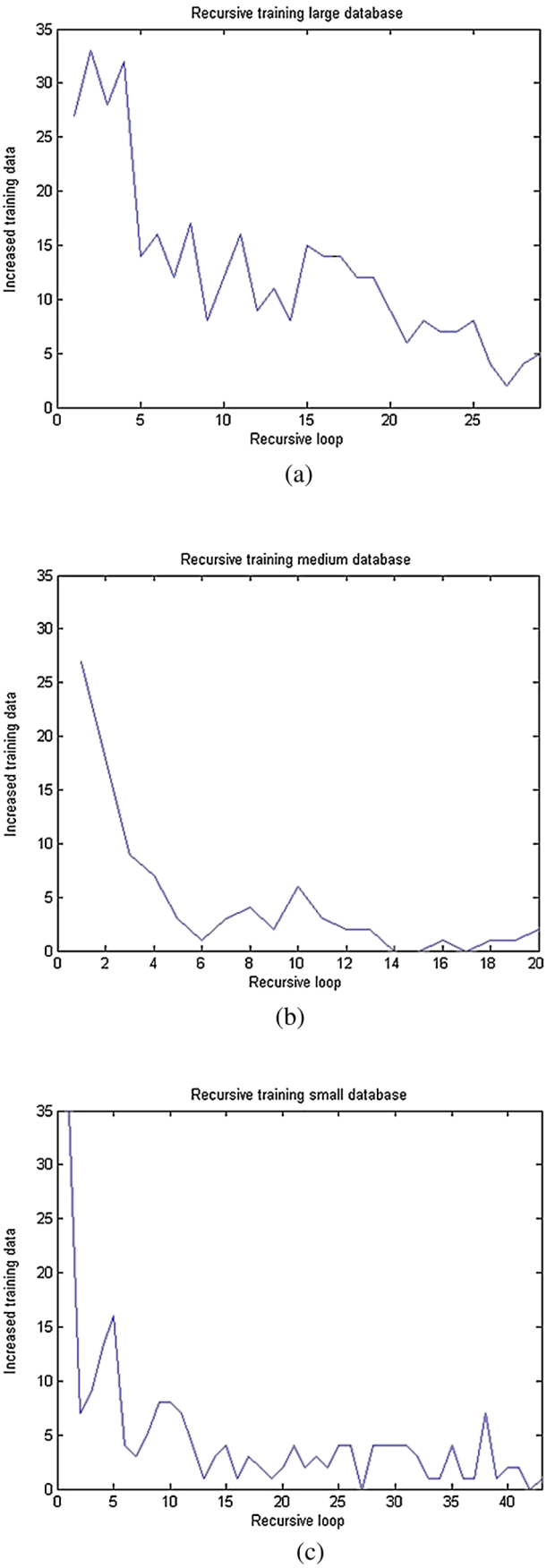


Fig. 7 Recursive training curve of performing the self-organized training data–yielding method: (a) coarse range, (b) medium range, and (c) fine range.

Step 1: Transaction built action.
The aims of this action are twofold: to accumulate the transaction set and to select groups. Regarding the accumulation of transaction set, the transactions are built using the following equations:

$$\begin{aligned} &\text{if Fitness}_{M_k} \geq (\text{Best Fitness}_{M_k} \\ &\qquad - \text{Thread Fitness value}) \\ &\text{Transaction}_j[i] = \text{TFC Rule Set}_{M_k}[i] \\ &\text{then Performance Index} = \text{g}, \end{aligned} \tag{20}$$

$$\begin{aligned} &\text{if Fitness}_{M_k} < (\text{Best Fitness}_{M_k} \\ &\qquad - \text{Thread Fitness value}) \\ &\text{Transaction}_j[i] = \text{TFC Rule Set}_{M_k}[i] \\ &\text{then Performance Index} = b, \end{aligned} \tag{21}$$

where $i = 1, 2, \ldots, M_k$, $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, $j = 1, 2, \ldots,$ TransactionNum, the $\text{Fitness}s_{M_k}$ is the fitness value of TNFN with M_k rules, ThreadFitnessvalue is a predefined value, TransactionNum is the total number of transactions, $\text{Transaction}_j[i]$ is the ith item in the jth transaction, TFC Rule Set$M_{M_k}[i]$ is the i is the ith group in the M_k groups used for chromosomes selection, and Performance Index $= g$ and Performance Index $= b$ represent the good and bad performance, respectively. Hence, transactions have the form shown in Table 1. As shown in Table 1, the first transaction indicates that the three-rule TNFN formed by the first, fourth, and eighth groups have "good" performance. In contrast, the second transaction indicates that the four-rule TNFN formed by the second, fourth, seventh, and the tenth groups have "bad" performance.
Regarding the group selection, ARSM selects groups using the following equation:

$$\begin{aligned} &\text{if Accumulator} \leq \text{Normal times} \\ &\quad \text{Group Index} = \text{Random}[1, P_{\text{size}}], \end{aligned} \tag{22}$$

where $i = 1, 2, \ldots M_k$, $M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}$, Accumulator is used to determine which action should be adopted, GroupIndex$[i]$ is the selected ith group of the M_k groups, and P_{Size} indicates that there are P_{Size} groups in a population in ARMELA. If the best fitness value does not improve for a sufficient number of generations (Normal Times), then ARSM selects groups according to the association rule mining action.

Step 2: Association rule mining action.
In the transaction-built action, suitable groups are randomly selected from populations. In the association rule mining action, suitable groups are selected according to the association rules. To consider the association rules further, they can be found using three steps: (1) obtain the frequently occurring groups from FP-growth, (2) generate association rules, and (3) select suitable groups. The details of these three steps are presented as follows.

i. Finding frequently occurring groups:
In this step, only good groups, whose performance index is "g" in Table 1, are performed by finding the frequently occurring groups; bad groups are left out. Thus, frequently occurring groups can be found according to the predefined Minimum_Support, which stands for the minimum fraction of transactions containing the item set. After Minimum_Support is defined, this paper adopts the FP-growth algorithm[23] to perform frequent pattern mining. In FP-growth, frequently occurring groups can be found by exploring the FP-tree.[23] After exploring the frequently occurring groups in the FP-tree, FP-growth data mining is completed by the concatenation of the suffix group[23] with the generated frequently occurring groups. Thus, in this paper, frequent groups denote the frequently occurring groups found by FP-growth algorithm.

ii. Generating association rules.
To produce the association rules with good performance, the frequent groups must combine with the groups with bad performance shown in Table 1 to count the confidence degree, which can be computed by the following formula:

$$\begin{aligned} &\text{confidence}(\text{frequent groups} \Rightarrow \text{good}) = P(\text{good}|\text{frequent groups}) \\ &\qquad = \frac{\text{supp}(\text{frequent groups} \cup \text{good})}{\text{supp}(\text{frequent groups} \cup \text{good}) + \text{supp}(\text{frequent groups} \cup \text{bad})}, \end{aligned} \tag{23}$$

where $P(\text{good}|\text{frequent groups})$ is the conditional probability, frequent groups $\cup$ good or bad is the union of frequent groups and good or bad performance, and supp(frequent groups $\cup$ good or bad) is the counts of frequent groups with good or bad performance occurring in transactions. Then, the rule is valid if

$$\begin{aligned} &\text{confidence}(\text{frequent groups} \Rightarrow \text{good}) \\ &\quad \geq \text{minconf}, \end{aligned} \tag{24}$$

where minconf is the minimal confidence given by a user or an expert. Hence, we can infer that if a rule satisfies Eq. (24), then the frequent groups can be considered as the suitable groups. For example, if the confidence of $\{2,5,8\} \Rightarrow < \{g\}$ is larger than the minimum confidence, we produce this association rule, which indicates that the combination of the second, fifth, and eighth groups have "good" performance. After doing so, the frequent groups are conducted to produce association rules and generate the Associated Good Pool, which contains all frequent groups that satisfy Eq. (24).

iii. Selecting suitable groups.
After the association rules are constructed, ARSM selects groups according to the association rules. The group indexes are selected from the associated good groups according to the following equations:

$$\begin{aligned} &\text{if Normal Times} < \text{Accumulator} \\ &\qquad \leq \text{Explore Times} \\ &\text{then GroupIndex}[i] = w, \\ &\text{where } w = Good\,Item\,Set[q] \\ &\qquad = \text{Random}[\text{Associated Good Pool}], \end{aligned} \tag{25}$$

where $q = 1, 2, \ldots,$Associated Goodpool Num, $i = 1, 2, \ldots M_k$, $M_k = M_{\min}$, $M_{\min+1}$, $\ldots, M_{\max}$, Explore Times is a predefined value that judge to perform the association rule mining action, Associated Good Pool is the sets of good item set obtained from the association rules, Associated Good Pool Num is the total number of sets in Associated Good Pool, and GoodItemSet$[i]$ presents a good item set randomly selected from Associated Good Pool. In the Eq. (25), if M_k is greater than the size of GoodItem Set, the remaining groups are selected using Eq. (22).

Step 3: If the best fitness value does not improve for a sufficient number of generations (Explore Times), ARSM selects groups based on the transaction built action and sets Accumulator $= 0$.

Step 4: After the M_k groups are selected, M_k chromosomes are selected from M_k groups as follows:

$$\text{Chromosome Index}[i] = q, \tag{26}$$

where $q = \text{Random}[1, N_c]$, $i = 1, 2, \ldots, k$, N_c is the total number of chromosomes in each group, and ChromosomeIndex$[i]$ is the index of a chromosome that is selected from the ith group.

3.2.4 *Fitness assignment*

To assign the fitness value of an individual, the following detailed steps in the fitness value assignment are performed:

Table 4 Initial parameters of ARMELA training.

Parameters	Value of coarse range	Value of medium range	Value of fine range
Psize	60	40	40
Nc	20	20	20
Selection_Times	50	50	50
NormalTimes	10	10	5
ExploreTimes	15	15	8
Crossover Rate	0.6	0.6	0.6
Mutation Rate	0.2	0.2	0.2
[Mmin, Mmax]	[38, 45]	[18, 25]	[18, 25]
[mmin, mmax]	[−9.5, 9.5]	[−8.5, 8.5]	−14.5, 14.5]
[σmin, σmax]	[14, 16]	[13, 15]	[40, 43]
[wmin, wmax]	L2 regularization determined	L2 regularization determined	L2 regularization determined
Minimum_Support	Transaction Num/2.5	Transaction Num/2.8	Transaction Num/3
Minimum_Confidence	60%	60%	60%
L2 regularization parameter (λ)	0.003	0.003	0.003

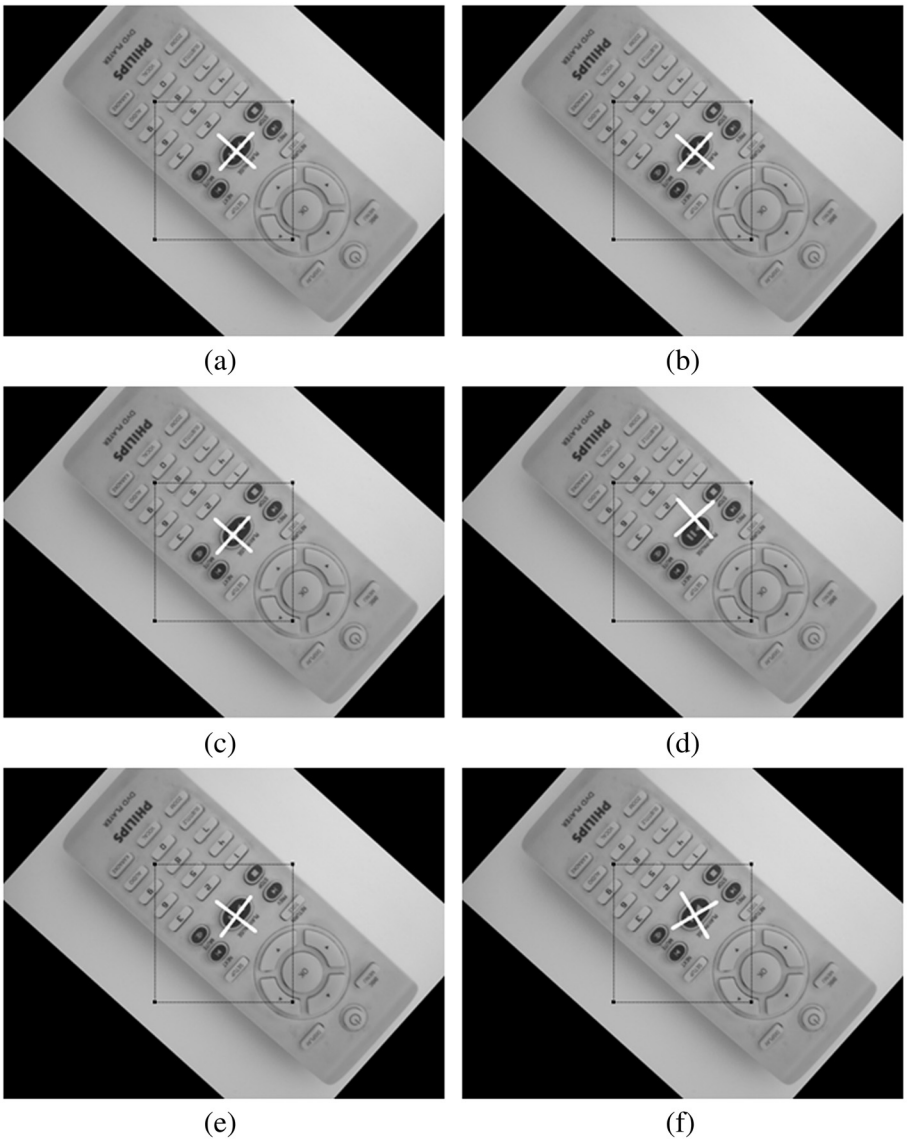


Fig. 8 Alignment results of different systems: (a) ground truth, (b) proposed system, (c) DCT, (d) FFT, (e) KICA, and (f) ISOMAP.

Table 5 Alignment errors in different image alignment systems.

	Errors							
	ErrScale		ErrAngle (deg)		ErrDx (pixels)		ErrDy (pixels)	
Method	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Proposed	0.0070	0.01134	0.0353	0.2601	0.2829	0.2164	0.3095	0.5673
DCT[15]	0.0302	0.0350	6.8495	8.8052	6.7206	10.0008	6.3597	10.6839
FFT[29]	0.0229	0.0348	7.9348	8.8924	9.7631	10.2108	9.0485	9.4451
KICA[14]	0.0333	0.0370	9.8534	14.1339	6.6953	10.9533	6.0219	9.5207
ISOMAP[13]	0.0670	0.0557	14.3922	21.0862	8.4077	14.4331	7.3752	9.7249

Step 1: Choose M_k antecedent part of fuzzy rules using L2 regularization to construct a TNFN Rp_{M_k} times from M_k groups with size N_C. The M_k groups are obtained from the ARSM.

Step 2: Evaluate every TNFN that is generated from Step1 to obtain a fitness value. In this paper, the fitness value is designed according to the following formulation:

$$\text{Fitness Value} = 1/(1 + E(y, \bar{y})), \tag{27}$$

$$\text{where } E(y, \bar{y}) = \sum_{i=1}^{N} (y_i - \bar{y}_i)^2, \tag{28}$$

where y_i and $\bar{y}_i$ represents the desired and predicted values of the ith output, respectively, $E(y, \bar{y})$ is an error function and N represents the number of the training data in each generation.

Step 3: Divide the fitness value by M_k and accumulate the divided fitness value to the selected antecedent part of fuzzy rules with their fitness value records.

Step 4: Divide the accumulated fitness value of each chromosome from M_k groups by the number of times that it has been selected.

3.2.5 *Reproduction strategy*

This study utilizes our previous research, namely, elite-based reproduction strategy (ERS),[21] to perform reproduction. In ERS, every chromosome in the best combination of M_k groups must be kept by performing the reproduction step. In the remaining chromosomes in each group, this study uses the roulette-wheel selection method[24] for the reproduction process. The well-performing chromosomes in the top half of each group[25] proceed to the next generation. The other half is created by executing crossover and mutation operations on the chromosomes in the top half of the parent individuals.

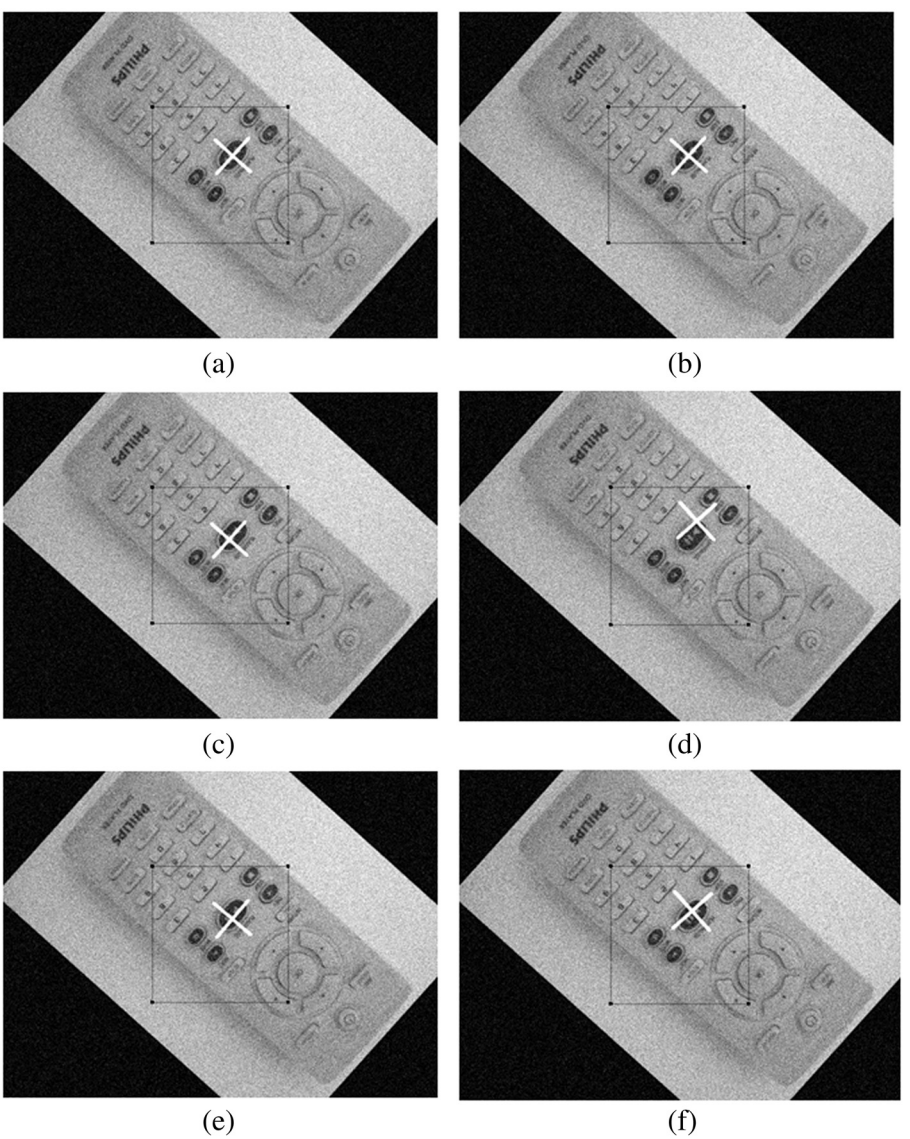


Fig. 9 Alignment results of different systems under a 10-dB SNR condition: (a) ground truth, (b) the proposed system, (c) DCT, (d) FFT, (e) KICA, and (f) ISOMAP.

3.2.6 *Crossover strategy*

The main method to attain the inheritance of parents is the crossover operator, the operation of which occurs for a selected pair with a crossover rate. In this paper, a two-point crossover strategy[24] is adopted. The benefits of the two-point crossover are the ability to introduce a higher degree of randomness into the selection of genetic material[26] and the ability to yield better performance than one-point crossover.[27]

3.2.7 *Mutation strategy*

Mutation can randomly alter the allele of a gene. In this paper, uniform mutation[24] is adopted. The mutated gene is drawn randomly from the domain of the corresponding variable. The major advantage of uniform mutation is its ability to provide new and highly diverse information for a population.[28]

- Termination condition.
  The aforementioned seven operators are performed repeatedly and are stopped when the number of generations reaches a predefined value or when the fitness value is greater than the fitness limit.

4 Experimental Results

In this section, the visual inspection images with a size of 640×480 pixels are utilized to verify the utility of the proposed alignment method. Figure 6 demonstrates such images whose left and right sides are the reference and transformed images, respectively. In this figure, the dashed window of the image represents the template window with a size of 200×200 pixels, and the cross sign denotes the reference location of the template. Furthermore, Table 2 defines the target alignment range for aligning the visual inspection images. All image alignment systems mentioned in this section are implemented to reach the target alignment range.

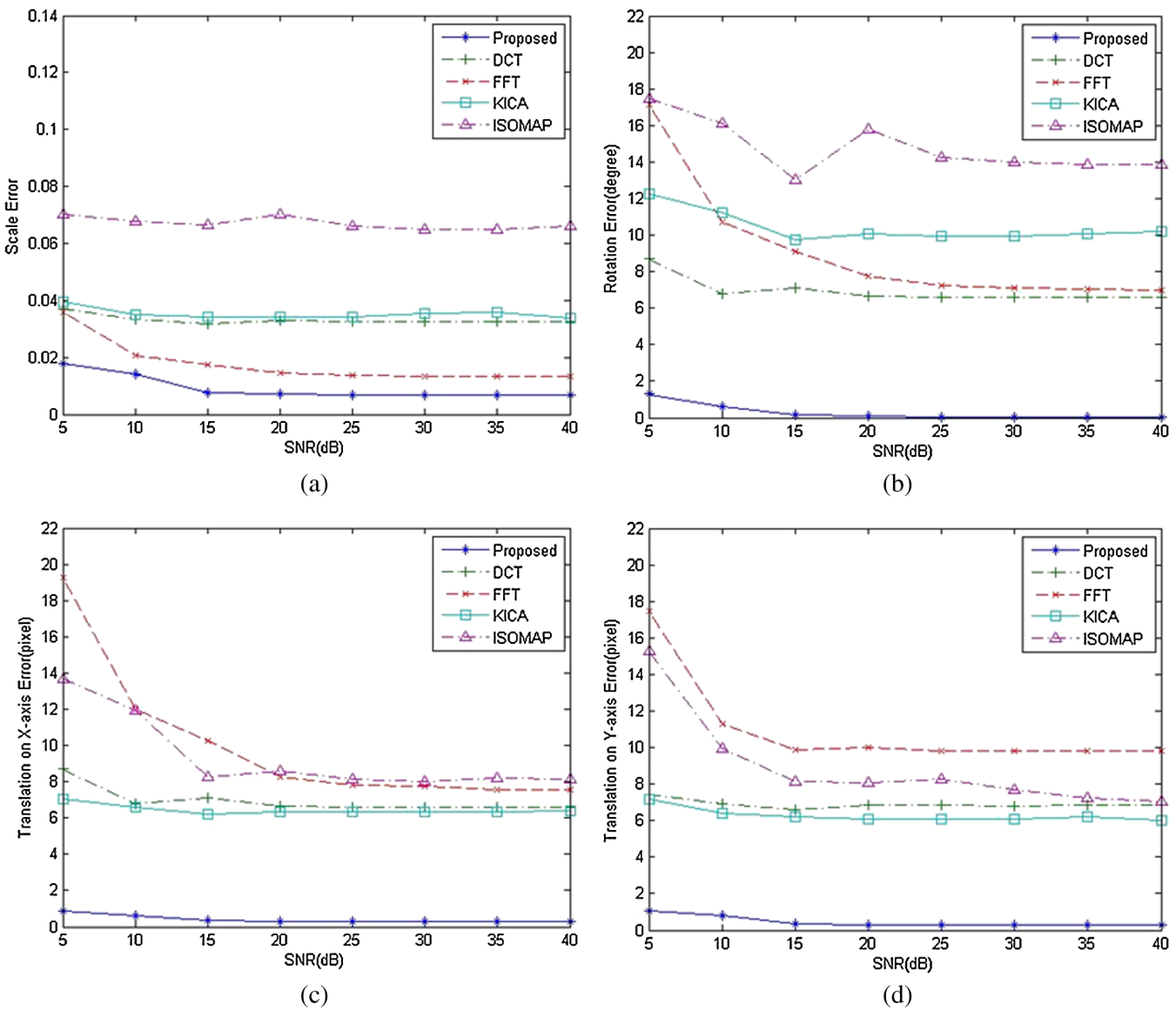


Fig. 10 Comparison of the average affine transformation errors using the proposed method, DCT, FFT, KICA, and ISOMAP under various SNRs: (a) error with respect to scale, (b) rotation, (c) translation on X-axis, and (d) translation on Y-axis.

All experiments are performed by using an Intel Core i7 860 chip with a 2.8 GHz CPU, a 3G memory, and the Matlab 7.5 simulation software.

4.1 *Cooperative Neural-Fuzzy Network with the ARMELA Training*

To achieve the target alignment range defined in Table 2, we choose three ranges of affine parameters described in Table 3 to accomplish the three-stage CNFNs. In this table, each range contains a single neural-fuzzy network, and these ranges cooperate to adapt to a coarse image alignment level. For the supply suitable training data for networks, this paper uses the self-organized training data-yielding method to generate 1165, 137, and 219 training data for coarse, medium, and fine alignment ranges, respectively. The map of recursive loop versus increased training data for each range defined in Table 3 is shown in Fig. 7. Based on this figure, the number of the increased training data decreases gradually and then self-organizes.

Prior to performing the training, the initial parameters of ARMELA are given in Table 4. Based on the training feature vectors and initial parameters, we perform the coarse, medium, and fine ARMELA training individually. These three-stage training stops when the fitness is greater than the predefined value. Therefore, once the training process has been performed, our image alignment system can be concluded to reach the target range defined in Table 2.

4.2 *Comparison with Existing Neural Network-based Image Alignment Systems*

To compare the proposed system with other existing neural network-based systems,[13–15,29] this paper carefully implements these systems according to the descriptions in their original paper. In this experiment, two typical comparisons including the alignment accuracy and robustness are discussed in the following parts.

4.2.1 *Alignment accuracy*

In the training phase, as utilizing the same number of training images (i.e., $1165 + 137 + 219 = 1521$) as in the proposed CNFN on traditional neural network–based methods[13–15,29] can yield large alignment error, we randomly generate

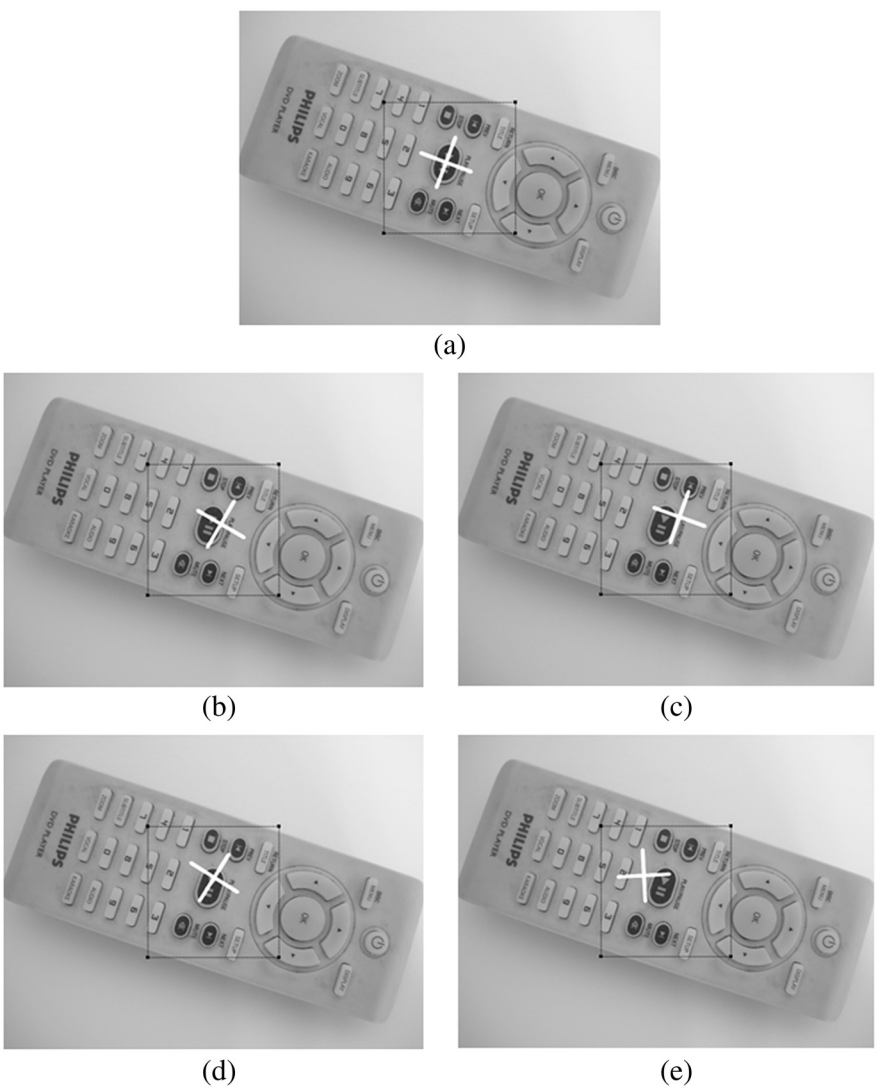


Fig. 11 Results of image alignment on real-images. (a) The proposed system, (b) DCT, (c) FFT, (d) KICA, and (e) ISOMAP.

another 4400 training images from the target alignment range described in Table 2 for training traditional methods. In the testing phase, we examine the alignment accuracy of the proposed and other systems by using the same 600 testing images randomly generated from the target alignment range.

Figure 8 presents an example of a synthesized testing image on five different systems. The cross sign in Fig. 8 denotes the estimated results. In this figure, the proposed system can estimate more accurate position and orientation of the cross sign than other systems.

To proceed to analyze the alignment accuracy, Table 5 describes the average and standard deviation error of five image alignment systems for 15 runs using different testing images. From this table, the proposed system exhibited the lowest alignment error than other systems. The result indicates that the proposed CNFN not only gets much higher alignment accuracy but also using fewer training data to reach better performance than other one-stage neural network methods.

4.2.2 *Alignment Robustness*

In this subsection, we verify further the robustness of the proposed image alignment system by adding different levels of random Gaussian noise. To achieve the aim of testing the robustness, 600 testing images are randomly generated with the addition of various strengths of Gaussian noise to examine different image alignment systems.

Figure 9 illustrates an image alignment example under a 10-dB SNR condition. From this figure, the proposed system depicts more accurate cross sign location than other methods.

Figures 10(a)–10(d) present the results of the absolute errors of the affine parameters under eight levels of SNR. As shown in these figures, the proposed system demonstrates much lower affine parameters error than other systems. This result indicates that the adopted Gabor-WGOH descriptor is not disturbed by a high noise level, and so is the proposed ARMELA-trained CNFN.

4.3 *Real-Image Alignment Testing*

In addition to the synthesized images, real-image testing cases are used to verify the alignment performance of the proposed system. Figures 11(a)–11(e) depict the experimental results of aligning the same real image utilizing the proposed system, discrete cosine transform (DCT), FFT, kernel independent component analysis (KICA), and ISOMAP, respectively. The proposed system demonstrates a more precise position and rotation of the cross sign than other systems. Thus, applying the proposed image alignment system to real-image alignment cases is feasible.

5 Conclusion

In this paper, the use of a CNFN with an ARMELA to perform image alignment tasks is proposed. The proposed CNFN offers a larger range of affine transformation and higher alignment accuracy in comparison with other traditional one-stage neural network–based approaches. Moreover, the self-organized training data-creating method can supply proper training data and prevent the selection of redundant ones for each stage of CNFN such that the total amount of training data decreases. The proposed ARMELA is a useful learning method for training CNFN such that the trained network can estimate affine parameters accurately. This evidence can be found in the experimental results of both synthesized and real-images cases. The results show that the proposed alignment system can reach a high accuracy and noise robustness level. In summary, this finding is helpful in developing accurate and robust image alignment systems.

Although the proposed model can demonstrate high performance, it still has some limitations. Specifically, as the application problem becomes more complicated, the number of cooperative neural-fuzzy networks would increase. Such condition leads the proposed model to suffer from the difficulty of choosing the suitable number of cooperative networks. If the unsuitable number of networks is chosen, the overall system will yield large estimated errors. Therefore, future works should identify a well-defined method to determine the number of cooperative neural-fuzzy networks automatically. Moreover, the noise consideration is only the Gaussian case and it is not sufficient in every case. Thus, in future studies, more noise cases should be considered to demonstrate the robustness of the proposed algorithm.

Acknowledgments

The authors thank the reviewers for their constructive comments and suggestions.

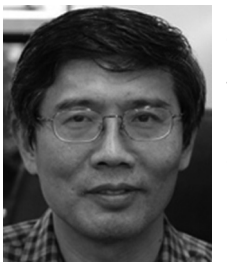
References

1. X. J. Liu, J. Yang, and H. B. Shen, "Automatic image registration by local descriptors in remote sensing," *Opt. Eng.* **47**(8), 087206 (2008).
2. X. M. Peng, W. Chen, and Q. Ma, "Feature-based nonrigid image registration using Hausdorff distance matching measure," *Opt. Eng.* **46**(5), 057201 (2007).
3. D. Skea et al., "A control point matching algorithm," *Pattern Recogn.* **26** (2), 269–276 (1993).
4. S. Manickam, S. D. Roth, and T. Bushman, "Intelligent and optimal normalized correlation for high-speed pattern matching," *NEPCON. WEST 2000*, February 27–March 2 (2000).
5. R. J. Althof, M. G. J. Wind, and J. T. Dobbins, "A rapid and automatic image registration algorithm with subpixel accuracy," *IEEE Trans. Med. Imag.* **16**(3), 308–316 (1997).
6. L. M. G. Fonseca and B. S. Manjunath, "Registration techniques for multisensor remotely sensed imagery," *Photogrammetric Eng. Remote Sens.* **62**(9), 1049–1056 (1996).
7. S. Kaneko, I. Murase, and S. Igarashi, "Robust image registration by increment sign correlation," *Pattern Recogn.* **35**(10), 2223–2234 (2002).
8. G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(10), 1858–1865 (2008).
9. M. Amintoosi, M. Fathy, and N. Mozayani, "Precise image registration with structural similarity error measurement applied to superresolution," *EURASIP Journal on Advances in Signal Processing 2009*, Article ID 305479, 7 pages (2009).
10. B. Zitova and J. Flusser, "Image registration methods: A survey," *Image Vis. Comput.* **21**(11), 977–1000 (2003).
11. I. Elhanany et al., "Robust image registration based on feedforward neural networks," in *Proceedings of IEEE International Conference on System, Man and Cybernetic*, Vol. **2**, pp. 1507–1511, IEEE, New York, NY, USA (2000).
12. J. Wu and J. Xie, "Zernike moment-based image registration scheme utilizing feedforward neural networks," in *Proceedings of the 5th World Congress on Intelligent Control and Automation*, Vol. **5**, pp. 4046–4048, IEEE, New York, NY, USA (2004).
13. A. B. Xu and P. Guo, "Isomap and neural networks based image registration scheme," *Lect. Notes Comput. Sci.* **3972**, 486–491 (2006).
14. A. B. Xu and P. Guo, "Image registration with regularized neural network," *Lect. Notes Comput. Sci.* **4233**, 286–293 (2006).
15. H. Sarnel, Y. Senol, and D. Sagirlibas, "Accurate and robust image registration based on radial basis neural networks," in *IEEE International Symposium on Computer and Information Sciences*, pp. 1–5, SPRINGER, New York, NY, USA (2008).
16. D. Lowe, "Distinctive image features from scale-invariant keypoints, " *Int. J. Comput. Vis.* **60**(2), 91–110 (2004).
17. D. M. Bradley et al., "Real-time image-based topological localization in large outdoor environments, " in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3670–3677, IEEE, New York, NY, USA (2005).

18. M. Hofmeister, M. Liebsch, and A. Zell, "Visual self-localization for small mobile robots with weighted gradient orientation histograms," in *40th International Symposium on Robotics (ISR)*, pp. 87–91, AER-ATP, Spain (2009).
19. P. Moreno, A. Bernardion, and J. S. Victor, "Improving the SIFT descriptor with smooth derivative filters," *Pattern Recogn. Lett.* **30** (1), 18–26 (2009).
20. T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern.* **15**(1), 116–132 (1985).
21. Y. C. Hsu, S. F. Lin, and Y. C. Cheng, "Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design," *Expert Syst. Appl.* **37**(7), 5320–5330 (2010).
22. S. F. Lin and Y. C. Cheng, "Two-strategy reinforcement evolutionary algorithm using data-mining based crossover strategy with TSK-type fuzzy controllers," *Int. J. Innovat. Comput. Control* **6**(9), 3683–3885 (2010).
23. J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM-SIGMOD*, pp. 1–12, ASSOC COMPUTING MACHINERY, New York, NY, USA (2000).
24. O. Cordon et al., *Genetic fuzzy Systems Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, Advances in Fuzzy Systems-Applications and Theory*, Vol. **19**, World Scientific Publishing, NJ (2001).
25. C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst. Man Cybern. Part B: Cybernetics*, **30**(2), 290–302 (2000).
26. E. Cox, *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufmann Publisher (2005).
27. G. Lin and X. Yao, "Analysing crossover operators by search step size," in *IEEE International Conference on Evolutionary Computation*, pp. 107–110, IEEE, New York, NY, USA (1997).
28. I. Dempsey, "Constant generation for the financial domain using grammatical evolution," in *Genetic and Evolutionary Computation Conference workshop program*, ACM Press, New York, NY, USA, pp. 350–353 (2005).
29. A. B. Abche et al., "Image registration based on neural network and Fourier transform," in *Proceedings of the 28th IEEE EMBS annual international conference*, pp. 803–4806, IEEE, New York, NY, USA (2006).

Chi-Yao Hsu received his BS in Department of Electrical Engineering from National Taiwan Ocean University, Taiwan, in 2001 and his MS in Department of Electrical Engineering from National Central University, Taiwan, in 2003. He is currently pursuing his PhD in the Department of Electrical Engineering, National Chiao-Tung University, Taiwan. His research interests lie in the areas of neural networks, fuzzy systems, evolutionary algorithms, pattern recognition, and computer vision.

Yi-Chang Cheng received his BS in engineering science from the Cheng Kung University, Taiwan, R.O.C., in 2005. He is currently pursuing his PhD at the Department of electrical and control engineering from the National Chiao Tung University, Taiwan, R.O.C. His research interests include neural networks, fuzzy systems, evolutional algorithms, and genetic algorithms.



Sheng-Fuu Lin received his BS and MS degrees in mathematics from National Normal University in 1976 and 1979, respectively, his MS in computer science from the University of Maryland in 1985, and his PhD in electrical engineering from the University of Illinois, Champaign, in 1988. Since 1988, he has been on the faculty of the Department of Electrical and Control Engineering at National Chiao Tung University, Hsinchu, Taiwan, where he is currently a professor. His research interests include fuzzy systems, genetic algorithms, neural networks, automatic target recognition, scheduling, image processing, and image recognition.