崑山科技大學學報第十期 第30~36頁 (民國104年9月)
Journal of Kun Shan University, No.10, pp.30~36 (September, 2015)

AOI Position and Speed Control Using Neural Networks Algorithm

*Chun-Jung Chen, *Tien-Chi Chen, *Min-Fang Wu, **Chia-Yu Lee, *Tsung-Heng Ma

*Department of Electrical Engineering, **Department of Aeronautics and Astronautics
*Kun Shan University, **National Cheng Kung University

Tainan, Taiwan, R.O.C
E-mail: ivanchen@mail.ksu.edu.tw

ABSTRACT--This paper presents a two layer recurrent neural network applied in glass position and speed control transmitted by linear servomotor in Automated Optical Inspection (AOI) system platform. The recurrent neural network consists of an identifier and a controller, the identifier is used to catch a feedback signal from the position sensor and the controller is processed in microprocessor in order to supply an adaptive PWM signal. The AOI glass is transmitted and controlled by linear servomotor. The PWM was processed by 30F40XX series microprocessor. The performance of the proposed method was proved better than without neural network algorithm. The theoretic formulations of the proposed neural networks were derived. The stability of the proposed method was also analyzed and demonstrated.

Keywords -- AOI Position Control, Speed control, Recurrent Neural networks, RNN.

1. Introduction

AOI system is used to inspect the blemishes of the glass such as Anti-Reflection (AR) glass, Indium Tin Oxide (ITO) glass, Indium Zinc Oxide (IZO) glass, etc. The defects of the inspected glass include finger touches, pinholes, scratches, and particles [1]. The transmission platform controlled by linear servomotor must be precisely controlled while glass on proper position and normal speed in AOI system platform. The glass has to be inspected consecutively and precisely controlled on the proper position. The most popular methods in AOI system are programmable logic control (PLC) method and microprocessor controller [2]. Because of the inspection efficiency concern the production cost, inspection speed, inspect accuracy and production quantity, AOI system has to find more precise and higher inspecting speedway to correct inspect in production line. Therefore, in recent years, the microchip processor controller with higher solution cameras or scanners is more favorably used in AOI system. Moreover, for more precise consideration in inspection, the PWM signals to drive servomotor transmit the platform produced by microchip processor is more popular than PLC controller, because of the neural algorithm provides robust and tolerant characteristics in position and speed control [3].

The advantages of the neural network algorithm applying in AOI system exhibit high reliability, good fault tolerance, and better performance on system control. It also can avoid the system complexity and effectively transmitted the glass with correct speed when glasses are inspecting. The inspection procedure can be simply explained as Fig. 1. The glass will be scanned twice among the inspecting procedure. X-Y motion table moves and capture the glass picture. The blemishes will be definitely found out [4, 5].

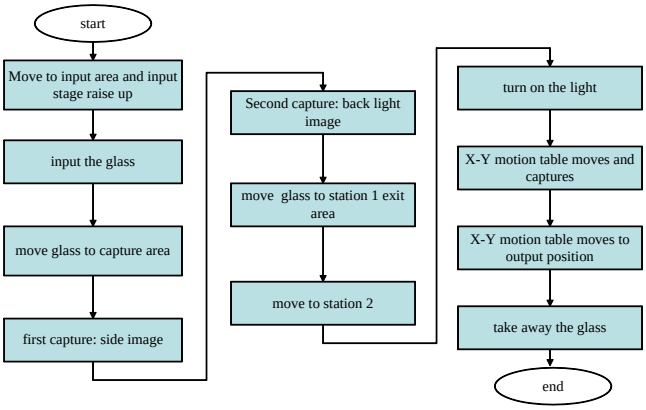


Fig. 1 the flow chart of glass inspection procedures in AOI system

For glass inspecting procedure, the glass must be very precisely transmitted by linear servomotor with adaptive speed and to proper position. Otherwise, the glass must be smoothly moved on the platform and taken the picture by line scanner. The position sensor can help the microchip to supply an adaptive PWM signal to linear servomotor. The neural network algorithm will be executed in the microchip program in order to produce an adaptive PWM control signal. Fig. 2 shows the speed control for glass inspection in AOI system platform.

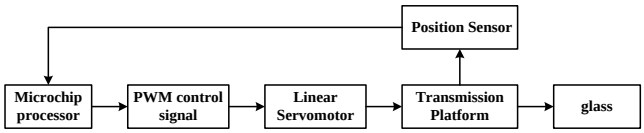


Fig. 2 the block diagram of position control for glass

The main equipments of AOI system platform are shown in Fig. 3. The platform of AOI system contains machine structure, glass, CCD scanner (charged-coupled device), position sensor, forward lighting source, back lighting source, linear servomotor, imagine DAQ card, I/O card, motion card and computer with human-interface program. The program is written by Borland C language. The microchip processor is embedded in motion card, neural network algorithm is written in microchip 30F30XX series microprocessor.

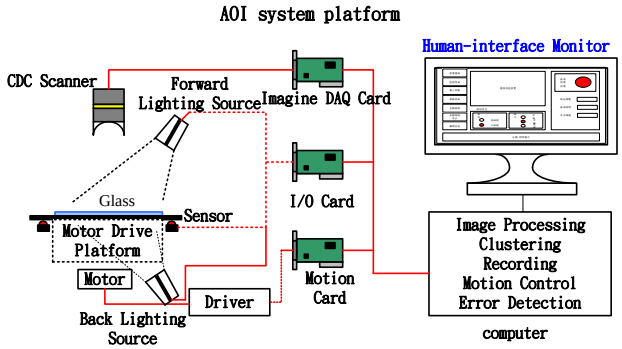


Fig. 3 The main equipments diagram of AOI system platform

The speed control based on neural network proposed in this paper is a new structure of the RNN. The proposed RNN contains two layers, first layer is called input-hidden layer and the second layer is called output layer. It differs from the conventional recurrent neural (CRNN) network, since the CRNN contains at least three layers [6], such as the input layer, hidden layer and output layer. In recent year, the effort on neural networks are searching the novel structure, lesser time consuming in calculation, number of weights parameter, and desired performance in wider operating range [7, 8]. The RNN proposed in this paper shows higher speed convergence in weight parameters, and it also improves the efficiency of learning speed, and obtains a better performance with the neural networks.

2. The Proposed AOI Position/Speed Control

The application for the proposed neural network algorithm in this paper is the AOI position and speed control by linear servomotor. The control architecture consists of a Recurrent Neural Network Identifier (RNNI) and a Recurrent Neural Network Controller (RNNC). The configuration of recurrent neural network for AOI position and speed control are shown in Fig. 4. The destination of the AOI position and speed control are used to minimize the inspection position of glass index for a linear servomotor. Simulation of RNN algorithm was compared with the RNN algorithm and without RNN algorithm for AOI Position Control [9-11].

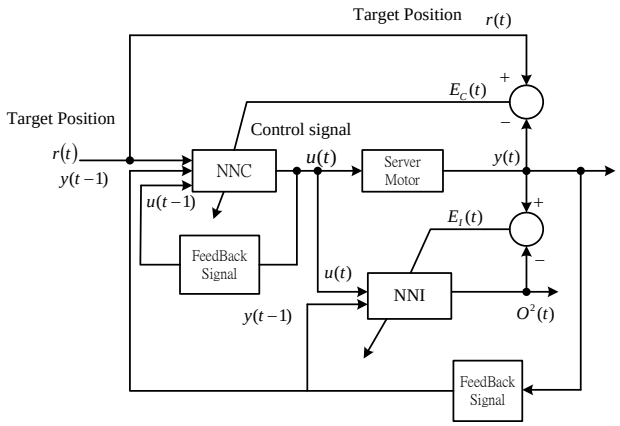


Figure 4. The configuration of RNN control system

2.1 The Recurrent Neural Network Identifier

The network structure of the proposed RNNI is shown in Fig. 5. The input layer and output layer have m_I =4 neurons and one neurons, respectively. The input and output of the *j*th neuron of the input layer are denoted by $I_j^1(t)$ and $O_j^1(t)$, $j=1,...,m_I$. The superscripts 1 and 2 represent the input layer (layer 1) and output layer (layer 2), respectively. The subscripts *j* represents the *j*th neuron. The input signals of the RNNI are $I_1(t),...,I_{m_I}(t)$, which consists of $u(t)$, $y(t-1)$, …, $y(t-3)$. Each neurons of the input layer are connected with each other using recurrent weights $W_{ji}^R(t)$, $i=1,...,m_I$ and $j=1,...,m_I$. The neurons of the input layer and output layer are connected with weights $W_j^O(t)$, $j=1,...,m_I$.

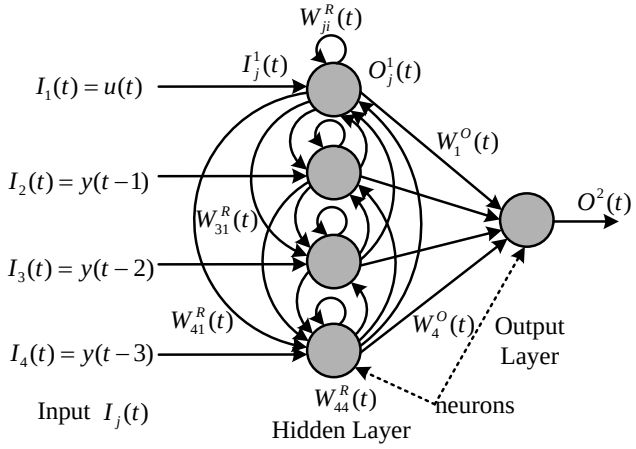


Fig. 5 The network structure of the RNNI

The mathematical operation of the RNNI can be expressed as follows.

(a) Input layer:

The input and output of the *j*th neuron of the input layer can be expressed as:

$$I_j^1(t)=I_j(t)+\sum_{i=1}^{m_I}W_{ji}^R(t)\cdot O_j^1(t-1),\ j=1,2,...,m_I \quad (1)$$

and

$$O_j^1(t)=f(I_j^1(t))=\frac{e^{I_j^1(t)}-e^{-I_j^1(t)}}{e^{I_j^1(t)}+e^{-I_j^1(t)}},\quad j=1,.....,m_I \quad (2)$$

where f(*) is a hyperbola function

(b) Output layer:

The input and output of the *k*th neuron of the output layer are equal and can be expressed as:

$$O^2(t)=I^2(t)=\sum_{j=1}^{m_I}W_j^O(t)\cdot O_j^1(t) \quad (3)$$

Due to the properties of guaranteed convergence, and minimizing the performance function, an error function for the RNNI is defined as

$$E_I(t)=\frac{1}{2}(y(t)-O^2(t))^2 \quad (4)$$

where $y(t)$ is the position platform output.

The weights $W_j^O(t)$ and $W_{ji}^R(t)$ can be adjusted using the steepest descent algorithm, which is expressed as

$$W_j^O(t+1)=W_j^O(t)+\Delta W_j^O(t)=W_j^O(t)-\eta_I^O\frac{\partial E_I(t)}{\partial W_j^O(t)} \quad (5)$$

$$W_{ji}^R(t+1)=W_{ji}^R(t)+\Delta W_{ji}^R(t)=W_{ji}^R(t)-\eta_I^R\frac{\partial E_I(t)}{\partial W_{ji}^R(t)} \quad (6)$$

where η_I^O and η_I^R and are the learning rates of the RNNI.

The gradient of error $E_I(t)$ with respect to the weights $W_j^O(t)$ and $W_{ji}^R(t)$ are represented as

$$\frac{\partial E_I(t)}{\partial W_j^O(t)}=\frac{\partial E_I(t)}{\partial O^2(t)}\cdot\frac{\partial O^2(t)}{\partial W_j^O(t)} \quad (7)$$

$$\frac{\partial E_I(t)}{\partial W_{ji}^R(t)}=\frac{\partial E_I(t)}{\partial O^2(t)}\cdot\frac{\partial O^2(t)}{\partial O_j^1(t)}\cdot\frac{\partial O_j^1(t)}{\partial I_j^1(t)}\cdot\frac{\partial I_j^1(t)}{\partial W_{ji}^R(t)} \quad (8)$$

Substituting (4) and (3) into (7) yields

$$\frac{\partial E_I(t)}{\partial W_j^O(t)}=-\left(y(t)-O^2(t)\right)\cdot O_j^1(t) \quad (9)$$

Substituting (4), (3), (2) and (1) into (8) yields

$$\frac{\partial E_I(t)}{\partial W_{ji}^R(t)}=-\left[\left(y(t)-O^2(t)\right)\cdot W_j^O(t)\cdot\left(1-\left(O_j^1(t)\right)^2\right)\cdot O_j^1(t-1)\right] \quad (10)$$

According to the above formula, the RNNI training algorithm is explained as follows.

Step 1: For t=1, arbitrarily initialize all weights $W_{ji}^R(t)$ and $W_j^O(t)$, $i=1,...,m_I$, $j=1,...,m_I$. Set the learning rates η_I^O and η_I^R.

Step 2: Forward calculate $I_j^1(t)$, $O_j^1(t)$ and $O^2(t)$ using Eqs. (1), (2) and (3) in sequence.

Step 3: Backward calculate $\frac{\partial E_I(t)}{\partial W_j^O(t)}$ and $\frac{\partial E_I(t)}{\partial W_{ji}^R(t)}$ using Eqs. (9) and (10).

Step 4: Calculate $W_j^O(t)$ and $W_{ji}^R(t+1)$ using Eqs. (5) and (6). Let t=t+1, then return step 2.

2.2 The Recurrent Neural Network Controller

The network structure of the proposed RNNC is shown in Fig. 3. The RNNC consists of a two-layered network structure, input layer and output layer. The input layer and output layer have m_C neurons and one neuron, respectively, which are setted as m_C = 4. The input and output of the *j*th neuron of the input layer are denoted by $S_j^1(t)$ and $T_j^1(t)$, $j=1,...,m_C$. The output layer has one neuron, which input and output are denoted as $S^2(t)$ and $u(t)$. The superscripts and subscripts are the same as those of the RNNI. The input signals of the RNNC are $S_1(t),...,S_{m_C}(t)$, which consists of $u(t-1)$, $y(t-1)$, …, $y(t-3)$. Each neurons of the input layer are connected with each other using recurrent weights $V_{ji}^R(t)$, $i=1,...,m_C$ and $j=1,...,m_C$. The neurons of the input layer and output layer are connected with weights $V_j^O(t)$, $j=1,...,m_C$.

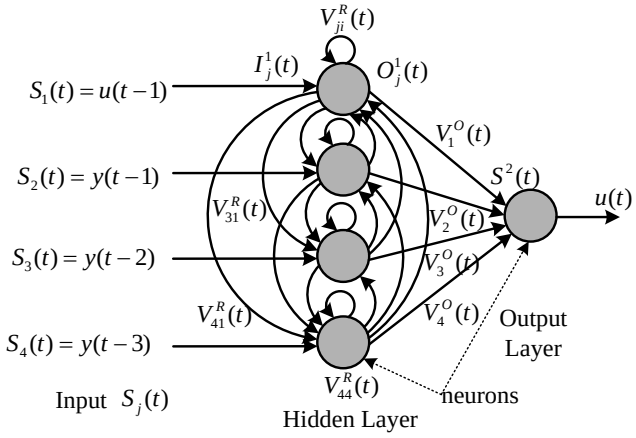


Fig. 6 The network structure of the RNNC

The mathematical operation of the RNNC can be expressed as follows.

(a) Input layer:

The input and output of the *j*th neuron of the input layer can be expressed as:

$$S_j^1(t)=S_j(t)+\sum_{i=1}^{m_C}V_{ji}^R(t)T_i^1(t-1),\quad j=1,.....,m_C \tag{11}$$

and

$$T_j^1(t)=f(S_j^1(t))=\frac{e^{S_j^1(t)}-e^{-S_j^1(t)}}{e^{S_j^1(t)}+e^{-S_j^1(t)}},\quad j=1,.....,m_C \tag{12}$$

(b) Output layer:

The output layer has one neuron, which input and output are equal and can be expressed as:

$$u(t)=S^2(t)=\sum_{j=1}^{m_C}V_j^O(t)T_j^1(t), \tag{13}$$

Due to the properties of guaranteed convergence, and minimizing the performance function, an error function for the RNNC is defined as

$$E_C(t)=\frac{1}{2}(r(t)-y(t))^2 \tag{14}$$

where $r(t)$ and $y(t)$ are the reference commands and plant outputs.

The weights $V_j^O(t)$ and $V_{ji}^R(t)$ can be adjusted using the steepest descent algorithm, which is expressed as

$$V_j^O(t+1)=V_j^O(t)+\Delta V_j^O(t)=V_j^O(t)-\eta_C^O\frac{\partial E_C(t)}{\partial V_j^O(t)} \tag{15}$$

$$V_{ji}^R(t+1)=V_{ji}^R(t)+\Delta V_{ji}^R(t)=V_{ji}^R(t)-\eta_C^R\frac{\partial E_C(t)}{\partial V_{ji}^R(t)} \tag{16}$$

where η_C^O and η_C^R and are the learning rates of the RNNC.

The gradient of error $E_C(t)$ with respect to the weights $V_j^O(t)$ and $V_{ji}^R(t)$ are represented as

$$\frac{\partial E_C(t)}{\partial V_j^O(t)}=\frac{\partial E_C(t)}{\partial y(t)}\cdot\frac{\partial y(t)}{\partial u(t)}\cdot\frac{\partial u(t)}{\partial V_j^O(t)} \tag{17}$$

$$\frac{\partial E_C(t)}{\partial V_{ji}^R(t)}=\frac{\partial E_C(t)}{\partial y(t)}\cdot\frac{\partial y(t)}{\partial u(t)}\cdot\frac{\partial u(t)}{\partial V_{ji}^R(t)} \tag{18}$$

Since the plant is normally an unknown system, the unknown values can be estimated using the RNNI. When the RNNI is learned, the dynamic behavior of the RNNI is close to the unknown plant, i.e., $y(t)\cong O^2(t)$. Therefore, using Eqs. (3), (2) and (1), $\frac{\partial y(t)}{\partial u(t)}$ becomes

$$\begin{aligned}\frac{\partial y(t)}{\partial u(t)}&\cong\frac{\partial O^2(t)}{\partial u(t)}=\frac{\partial O^2(t)}{\partial O_1^1(t)}\cdot\frac{\partial O_1^1(t)}{\partial I_1^1(t)}\cdot\frac{\partial I_1^1(t)}{\partial u(t)}\\&=W_1^O(t)\cdot\left(1-\left(O_1^1(t)\right)^2\right)\end{aligned} \tag{19}$$

Using Eqs. (13), (12) and (11), $\partial u(t)/\partial V_{ji}^R(t)$ becomes

$$\frac{\partial u(t)}{\partial V_{ji}^R(t)}=\frac{\partial u(t)}{\partial T_j^1(t)}\cdot\frac{\partial T_j^1(t)}{\partial S_j^1(t)}\cdot\frac{\partial S_j^1(t)}{\partial V_{ji}^R(t)}=V_j^O(t)\left(1-\left(T_j^1(t)\right)^2\right)T_i^1(t-1) \tag{20}$$

Substituting Eqs. (14), (19) and (13) into (17) yields

$$\frac{\partial E_C(t)}{\partial V_j^O(t)}=\left(r(t)-y(t)\cdot W_1^O(t)\cdot\left(1-\left(O_1^1(t)\right)^2\right)\right)\cdot T_j^1(t) \tag{21}$$

Substituting Eqs. (14), (19) and (20) into (18) yields

$$\begin{aligned}\frac{\partial E_C(t)}{\partial V_{ji}^R(t)}&=\left(r(t)-y(t)\cdot W_1^O(t)\cdot\left(1-\left(O_1^1(t)\right)^2\right)\right)\\&\cdot V_j^O(t)\cdot\left(1-\left(T_j^1(t)\right)^2\right)\cdot T_i^1(t-1)\end{aligned} \tag{22}$$

According to the above formula, the RNNC training algorithm is explained as follows.

Step 1: For t=1, arbitrarily initialize all weights $V_{ji}^R(t)$ and $V_j^O(t)$, $i=1,...,m_C$ and $j=1,...,m_C$. Set the learning rates η_C^O and η_C^R.

Step 2: Forward calculate $S_j^1(t)$, $T_j^1(t)$ and $u(t)$ using Eqs. (11), (12) and (13) in sequence.

Step 3: Backward calculate $\frac{\partial E_C(t)}{\partial V_j^O(t)}$ and $\frac{\partial E_C(t)}{\partial V_{ji}^R(t)}$ using Eqs. (21) and (22).

Step 4: Calculate $V_j^O(t+1)$ and $V_{ji}^R(t+1)$ using Eqs. (15) and (16) . Let t=t+1, then return step 2.

3. Stability Analysis

The system will be stable if $\Delta e<0$. Then $e(t)$ will equal to zero as time t approaches ∞ (infinity). So we should first deduce the characteristic for the differentiation of the error function of the RNN.

We first define the error function as

$$E(t)=\frac{1}{2}\left(y(t)-O^2(t)\right)^2$$
$$=\frac{1}{2}e^2(t)$$

By using the Lyapunov function,

Let $V(t)=\frac{1}{2}e^2(t)$

$$\dot{V}(t)=e(t)\cdot\dot{e}(t)$$

Let

$$\Delta V(t)=V(t+1)-V(t)$$
$$=\frac{1}{2}(e^2(t+1)-e^2(t))$$
$$=\frac{1}{2}(e(t+1)-e(t))(e(t+1)+e(t))$$
$$=\Delta e(t)\cdot(e(t)+\frac{1}{2}\Delta e(t))$$

let $\Delta e(t)=\left[\frac{\partial e(t)}{\partial W(t)}\right]^T\cdot\Delta W(t)$, and

$\because \Delta e(t)=e(t+1)-e(t)$

$\therefore\ e(t+1)=e(t)+\left[\frac{\partial e(t)}{\partial W(t)}\right]^T\cdot\Delta W(t)$

By using the steepest descent algorithm

$$\Delta\vec{W}(t)=-\eta(t)\frac{\partial E(t)}{\partial\vec{W}(t)}$$
$$=\eta(t)\cdot e(t)\cdot\frac{\partial O^2(t)}{\partial\vec{W}(t)}$$
$$\Delta V(t)=\left[\frac{\partial e(t)}{\partial\vec{W}(t)}\right]^T\Delta\vec{W}(t)\cdot\left(e(t)+\frac{1}{2}\left[\frac{\partial e(t)}{\partial\vec{W}(t)}\right]^T\Delta\vec{W}(t)\right)$$
$$=-\eta(t)\cdot e^2(t)\left\|\frac{\partial O^2(t)}{\partial\vec{W}(t)}\right\|^2+\frac{1}{2}\eta^2(t)e^2(t)\left\|\frac{\partial O^2(t)}{\partial\vec{W}(t)}\right\|^4$$

Define

$$g_{\max}=\max_t\|\vec{g}(t)\|=\max_t\left\|\frac{\partial O^2(t)}{\partial\vec{W}(t)}\right\|$$

where $\|\bullet\|$ is the Euclidean norm.

Instead of $\frac{\partial O^2(t)}{\partial\vec{W}(t)}$ by $\vec{g}(t)$

Then

$$\Delta V(t)=-\eta(t)\cdot e^2(t)\cdot\|\vec{g}(t)\|^2+\frac{1}{2}\eta^2(t)\cdot\|\vec{g}(t)\|^4$$
$$=-\frac{1}{2}\|\vec{g}(t)\|^2\eta(t)\left(2-\eta(t)\|\vec{g}(t)\|^2\right)\cdot e^2$$
$$=-\lambda\cdot e^2(t)<0 \qquad (10)$$

For the equation (10), because of $\|\vec{g}(t)\|^2$, e^2 must be greater than zero. And

if $\lambda=\frac{1}{2}\|\vec{g}(t)\|^2\cdot\eta(t)(2-\eta(t)\|\vec{g}(t)\|^2)>0$

Then the system will converge and stabilize, or

if $0<\eta(t)<\frac{2}{g_{\max}^2}$

then the system will be stable

4. Algorithm of RNN in AOI platform

Step 1. Initial weights matrix $W_{ji}^R(t)$, $W_j^O(t)$ and $V_{ji}^R(t)$, $V_j^O(t)$, η_I^O η_I^R, η_c^O and η_c^R.

Step 2. Calculate the inputs $S_i(t)$, $I_j^1(t)$ and outputs of $O_j^1(t)$, $u(t)$ of RNNC.

Step 3. Calculate the output of system plant $y(t)$.

Step 4. Define the input $I_i^1(t)$ of RNNI and calculate each input and output of the layer, $O_j^1(t)$, $I^2(t)$ and $O^2(t)$.

Step 5. using equation (9), (10) to adjust the weights of $W_{ji}^R(t)$, $W_j^O(t)$, $V_{ji}^R(t)$ $V_j^O(t)$.

Step 6. Go back to step 2.

4. Simulation results

According to the practical control conditions used in AOI system the speed simulation were set at 20 mm/s, 10 mm/s and 5 mm/s, individually. The route of moving platform was set 1500 mm, the maximum load could be 120N, and the simulation time for NNC controller to meet the target speed of linear was set in 5 seconds.

The parameters of linear servomotor and NNI are set as follows:

a. J= 6.8×10^{-4}

b. B= 5.15×10^{-4} (viscosity coefficient)

c. $\eta_I^R=0.003$

d. $\eta_I^O=0.003$

e. $\eta_c^R=0.0005$

f. $\eta_c^O=0.0005$

a. Linear moving speed at 400 rpm:

Figure 7 shows the performance of the motor output speed rapidly reach the target speed. Figure 8 shows the performance of motor speed and the NNI learning effect, it can cause NNC to get an adaptive output signal onto the controller.

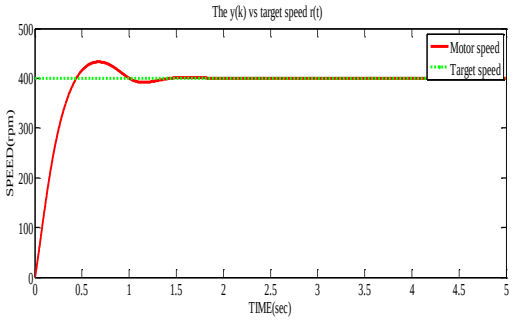


Fig. 7 The performance of motor speed and the target speed

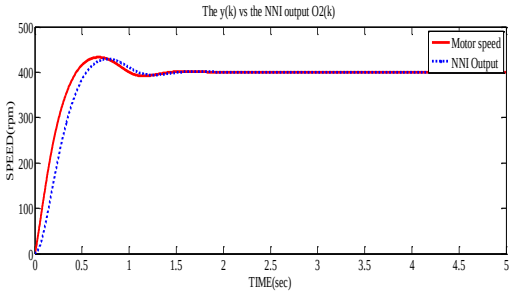


Fig. 8 the performance of motor speed with NNI learning

b. Linear moving speed at 450 rpm:

Figure 9 and 10 show the simulation performances of motor speed vs. target speed and the performance of the NNI traces the motor plant outputs. It can rapidly reach the target value within 1 second.

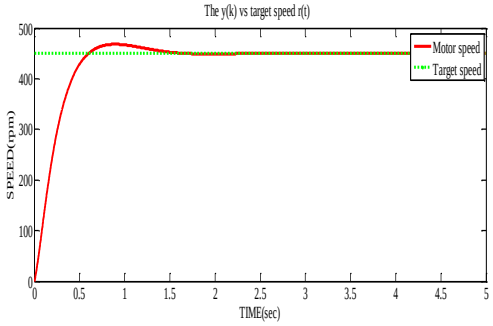


Fig. 9 the performance of the motor speed and the target speed at speed 450 rpm

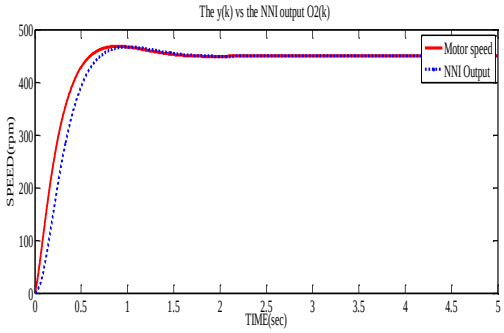
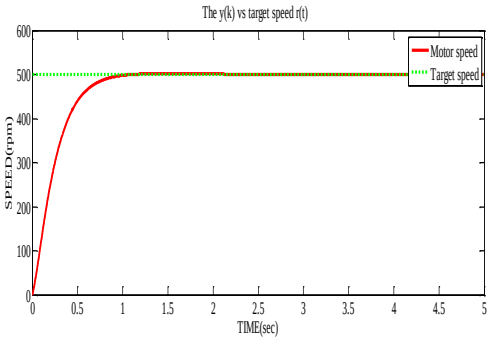


Fig. 10 the performance of motor speed with NNI learning at speed 450 rpm

c. Linear moving speed at 500 rpm:

When the target speed is adjusted to 500 rpm, the NNC controller shows best performance in speed control and NNI learning with the motor plant. No overshoot happens and rapidly reach the target values within 1 second. The simulation results as shown in Fig. 11 and 12, individualy.



the

Fig. 11 performance of the motor speed and the target speed at speed 500 rpm

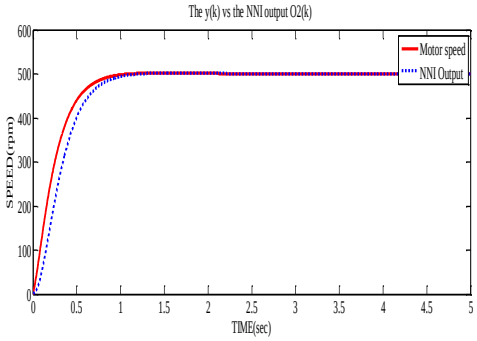


Fig. 12 the performance of motor speed with NNI learning at speed 500 rpm

5. Conclusion

The RNN controller for linear servomotor speed control in AOI system proposed in this paper shows high speed and stable convergence with same learning rate. It differs from the conventional three layer neural network and other controller such as PLC, PID or fuzzy neural controller methods. It also shows better fault tolerance, as well as different speed without learning rates adjustment. They also can avoid the system complexity and effectively control the speed of the linear servomotor.

NOMENCLATURE

E_I	Error function of RNNI.
E_c	Error function of RNNC.
I_i^1	First layer's input of RNNI at i th neuron.
I_{ci}^1	First layer's input of RNNC at i th neuron.
I^2	Second layer's input of RNNI
O_j^1	First layer's output of RNNC at j th neuron
O_j^1	First Layer's output of RNNI at i th neuron.
O^2	Second layer's output of RNNI
u	Output of RNNC

W_{ji}^{R} First layer's weight of RNNI between j th neuron and i th neuron.

W_{j}^{O} Second layer's weight of RNNI at j th neuron.

V_{ji}^{R} First layer's weight of RNNC between j th neuron and i th neuron

V_{j}^{O} Second layer's weight of RNNC between j th neuron.

X_{Ii} Input values of RNNI

X_{i} Input values of RNNC.

y Output of plant.

η_{c}^{O} Learning rate of RNNC at output layer.

η_{c}^{R} Learning rate of RNNC at first layer.

η_{I}^{O} Learning rate of RNNI at output layer.

η_{I}^{R} Learning rate of RNNI at first layer.

Acknowledge: This work is supported by Ministry of Economy Affairs for the Local Scientific Specialist Project under contract 101-EC-17-A-04-S1-220.

REFERENCES

[1] Ziyin LI and Qi Yang "System design for PCB defects detection based on AOI technology,", Image and Signal Processing (CISP), 4th International Congress. Vol. 4, pp. 1988-1991, 2011.

[2] Zeyu Li, Jiangqiang Hu and Xingxing Huo, "PID Control Based on RBF Neural Network for Ship Steering," , Information and Communication Technologies (WICT), World Congress . pp.1076-1080,2012.

[3] ZHENG LI[1], QUN-JING WANG[2]. "NEURAL NETWORK CONTROL SCHEMES FOR PM SPHERICAL STEPPER MOTOR DRIVE". Machine Learning and Cybernetics, 2008 International Conference .Vol. 4, pp. 2042-2047, 2008.

[4] Weifeng Shi. " Multi-Neural Networks Control of Marine Diesel Engine Generator Set" Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference. Vol. 2, pp. 508-512, 2007.

[5] Rong-Jong Wai, *Senior Member, IEEE*, and Chia-Chin Chu, "Motion Control of Linear Induction Motor via Petri Fuzzy Neural Network". IEEE Trans. Vol. 54, pp. 281-295, 2007.

[6] Faa-Jeng Lin, *Senior Member, IEEE*, and Rong-Jong Wai, *Member, IEEE* "Hybrid Control Using Recurrent Fuzzy Neural Network for Linear-Induction Motor Servo Drive", IEEE Trans. Vol. 9, pp. 102-115, 2001.

[7] N. Benamrane, A. Aribi, L. Kraoula, "Fuzzy Neural Networks and Genetic Algorithms for Medical Images Interpretation", Geometric Modeling and Imaging - New Trends. pp. 259-264, 2006.

[8] Quang-Cherng Hsu, Chin-Wen Lin, Jian-Yuan Chen, "Development of an Automatic Optical Inspection System for Defect Detection of Dental Floss Picks" Advanced Intelligent Mechatronics (AIM), IEEE/ASME International Conference. pp. 444-449, 2012.

[9] Rong-Jong Wai, SENIOR MEMBER, IEEE, and Zhi-Wei Yang "Design of Fuzzy-Neural-Network Tracking Control With Only Position Feedback for Robot Manipulator Including Actuator Dynamics" Systems, Man and Cybernetics. SMC 2008. IEEE International Conference. pp.2349-2354, 2008.

[10] Mohammadali Abbasian, Jafar Solt ani, and Ali Salarvand , "Control of High Speed Linear Induction Motor Using Artificial Neural Networks" Human System Interactions, Conference.pp. 415-421, 2008.

[11] Hou Yunhai, Wang Yuhua. "Based on Neuron adaptive Controller for Linear Motor Slip Frequency Vector Control," Intelligent Human-Machine Systems and Cybernetics. IHMSC '09. International Conference.Vol. 2, pp. 94-98, 2009.