Development of Computational Intelligence-based Control System using Backpropagation Neural Network for Wheeled Robot

Conference Paper · October 2018

DOI: 10.1109/ICECOS.2018.8605183

Karlisa Priandana
Bogor Agricultural University
53 PUBLICATIONS **141** CITATIONS
SEE PROFILE

Wulandari Wulandari
Bogor Agricultural University
42 PUBLICATIONS **204** CITATIONS
SEE PROFILE

Sri Wahjuni
Bogor Agricultural University
65 PUBLICATIONS **147** CITATIONS
SEE PROFILE

Agus Buono
Bogor Agricultural University
102 PUBLICATIONS **323** CITATIONS
SEE PROFILE

Development of Backpropagation Neural Network-based Control System for Wheeled Robot

Karlisa Priandana
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
karlisa@apps.ipb.ac.id

Iqbal Abiyoga
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
iqbal_abiyoga@apps.ipb.ac.id

Wulandari
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
wulandari.ilkom@apps.ipb.ac.id

Sri Wahjuni
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
my_juni04@apps.ipb.ac.id

Medria Hardhienata,
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
medria.hardhienata@apps.ipb.ac.id

Agus Buono
Computer Science Department
Bogor Agricultural University
Bogor, Indonesia
agusbuono@apps.ipb.ac.id

***Abstract*—This study aims to develop an optimal autonomous control system for a three-wheeled robot with two motors. The focus of this research is a neural network direct inverse controller system that is trained using backpropagation algorithm. Autonomous control system training is carried out by using the real data of manually controlled wheeled robot. This study analyzed the use of two Backpropagation learning algorithms namely Levenberg Marquardt Backpropagation and Bayesian Regularization Backpropagation, and compares 3 different controller network configurations, *i.e. 13-10-2*, *13-20-2* and *13-26-2*. The simulation results revealed that the best control system network architecture is *13-10-2* which was trained using Bayesian Regularization Backpropagation algorithm. This result serves as early evidence that a neural network-based control system can be used for autonomous wheeled robots.**

***Keywords*—Backpropagation neural network, direct inverse controller, wheeled robot**

I. INTRODUCTION

Wheeled robots are robots that moves on the ground using motorized wheels. These types of robots are widely chosen for various autonomous tasks on flat terrains, *e.g.* land-slides disaster management [1], agricultural applications [2], multi-agent systems and cooperative tasks [3], [4], etc. Generally, wheeled robots are preferred over legged robots which require an additional algorithm for legs coordination[5].

Since its discovery, wheeled robots have been developed utilizing a variety of control systems, both mechanical and mathematical approaches [6], [7]. However, until now, the increasingly complex and non-linear robot environment remains a major problem. Conventional control methods, such as PID and back-stepping controller [8], cannot solve this issue due to its limitation in adapting to the changing environment characteristics [9]. Adaptive control system using mathematical approach can be a solution for this challenge, although its performance is considered not very satisfactory [10]. Neural network can be an option for adaptive control system because it has the ability to learn from its environment by mimicking the learning process of biological neurons [11]. Currently, the Multi-Layer Perceptron (MLP) neural network which is trained by using Backpropagation algorithm is the most widely used neural network architecture due to its simple structure and ease of design, despite its powerful performance. In the learning stage, each layer receives the error feedback from the next layer to iteratively adjust the corresponding weights, producing optimum weights for the network.

The application of neural network as a control system can be done by using the most fundamental control system concept, *i.e.* by using the inverse function of the plant, known as direct inverse control [10]. The basic process is to train and obtain the plant-inverse model by using neural network before applying it as a directly cascaded controller. This concept has been first applied for industrial machinery [12]. Then, following its successful implementation, the neural network direct inverse control system was also utilized as the controller of helicopter [13], hexacopter [9], quadcopter [14] and boat [15]. However, the concept has not been proven for well-known and widely used surface vehicles such as cars and wheeled robots.

This study aims to design and develop the direct inverse controller for a wheeled robot which has 3 wheels: 1 free-wheel and 2 motorized-wheels. Several simulations were conducted to find the optimum inverse controller network configuration in terms of computational training cost as well as control errors. Two Backpropagation learning algorithms namely the Levenberg Marquardt Backpropagation and the Bayesian Regularization Backpropagation are each utilized to train the controller with 3 network configurations, *i.e. 13-10-2*, *13-20-2* and *13-26-2*.

This paper is organized as follows. The developed wheeled robot which is used throughout this study along with its data acquisition for neural network training and testing is described in section 2. Then, the designed inverse controller architecture to be utilized as the robot controller under the direct inverse control scheme is explained in section 3. The results are then compared and analyzed in Section 4. The conclusion of this study is presented in Section 5.

II. The Wheeled Robot and Its Data Acquisition

A. The Wheeled Robot

A three-wheeled robot was developed as the plant that is utilized throughout this study (Fig. 1). The robot has two motorized wheels at the front that should be controlled to produce the desired movement, and one free-wheel at the back to maintain its stability. The robot is equipped with GPS and compass sensor for data acquisition, as well as a receiver to receive the control command when the robot is controlled manually by a remote control.

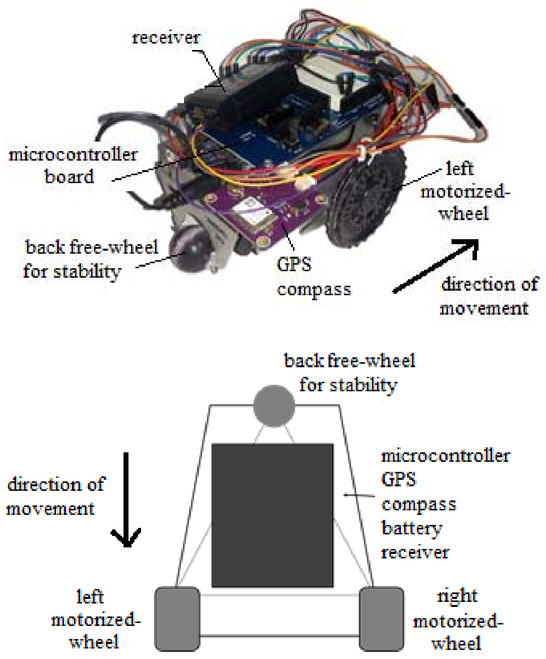

Fig. 1. The developed three-wheeled robot

To move the robot forward, the rotational speed (rpm) of the left and right motorized wheels should be equal. Meanwhile, to control the robot to turn left and right, the rotational speed of the right motorized wheel should be faster than the rotational speed of the left motorized wheel and vice versa. The rotational speed of each motor is described in revolutions per minute (rpm) which is controlled by the Electric Speed Controller (ESC) attached to each motor. The motor rotational speed in rpm is defined through the ESC by giving a corresponding control signal, which in this case, by using a Pulse Width Modulation (PWM) signal. Here, the robot controller generate a PWM signal and send it to the ESC. The ESC will then transform this control signal to a certain current to produce a corresponding motor rotational speed in rpm. Thus, in this application, there are two control signals, namely the left PWM (PWM 1) and the right PWM (PWM 2).

B. Data Acquisition

In this study, the utilized data is the robot motion data which was obtained from a real experiment. Data acquisition was conducted by controlling the robot using a remote control as depicted in Fig. 2. The robot was visually controlled to form a clockwise circular trajectory. There are 2 inputs and 3 outputs in concern:

1. The input $\boldsymbol{x}(k)$ is obtained directly from the remote control data, *i.e.* PWM 1, which is proportional to the rotational speed of the left motor, and PWM 2, which is proportional to the rotational speed of the right motor.
2. The output $\boldsymbol{y}(k)$ is obtained and derived from compass and GPS sensors data:

   (1).Heading: yaw/direction of the robot according to East-North-Up (ENU) coordinate system

   (2). X: position of the robot in earth x-axis

   (3). Y: position of the robot in earth y-axis

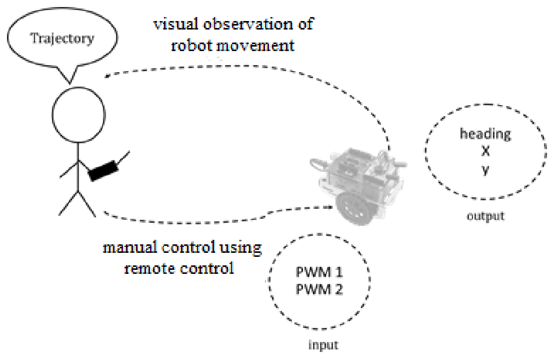
Trajectory
visual observation of robot movement
heading
X
Y
output
manual control using remote control
PWM 1
PWM 2
input

Fig. 2. Robot data acquisition

The control signals given to left and right motorized wheels are depicted in Fig. 3. The straight line on the top shows the sampled values of the left PWM whereas the dashed line on the bottom is the sampled values of the right PWM. The corresponding heading data is depicted in Fig. 4, whereas the robot position data in x and y axis of the earth is shown in Fig. 5.

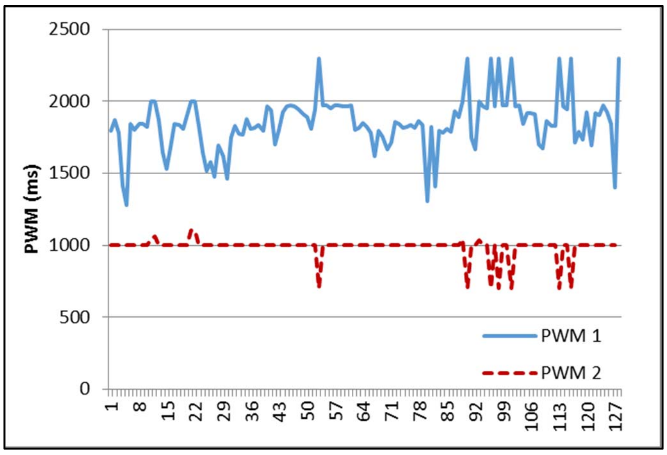

Fig. 3. Control signals

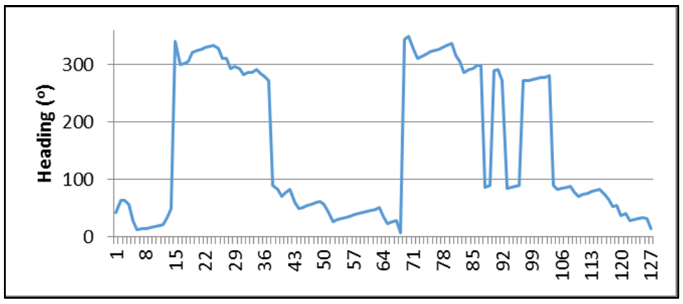

Fig. 4. Heading data

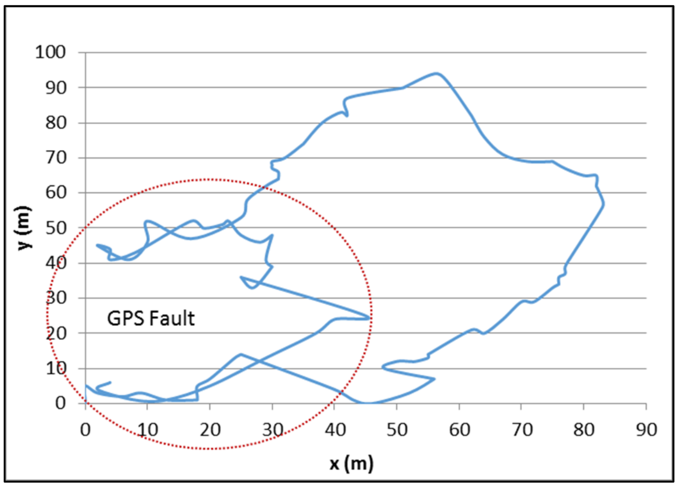

Fig. 5. Position data (trajectory)

Fig. 4 reflects that the robot was forming circular trajectory with some fast-maneuvers, *e.g.* at sample 35 and at sample 85 to 106. Meanwhile, the sudden change of data at sample 68 is due to the range of compass sensor measurement, which is from 0 to 360 degrees. It is depicted in Fig. 5 that there were some GPS fault as circled in red, which is good to test the effect of noisy data / non-linearity to the neural network controller.

III. Neural Network Inverse Controller

The inverse controller works by using the inverse function of the plant to be controlled, which is then directly cascaded with the plant as depicted in Fig. 6. In this scheme, $\boldsymbol{u}(k)$ is the control signals or input of the plant and $\boldsymbol{y}(k)$ is the output signals of the plant. Here, the system is expected to move according to the desired reference signals $\boldsymbol{r}(k)$, where $\boldsymbol{r}(k) \approx \boldsymbol{y}(k+1)$, *i.e.* the reference signal is the expected output of the plant.

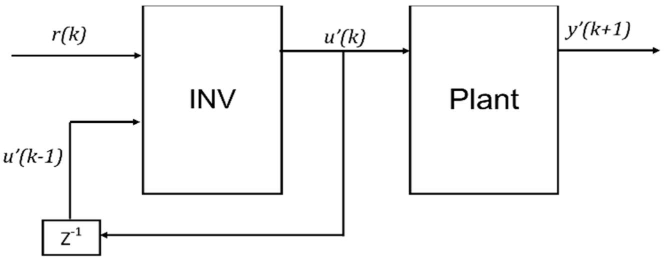

Fig. 6. Direct inverse control scheme

The use of neural networks in the inverse control system is described in [16], where the controller acts as an inverse of the plant, so that the desired plant output can be used directly as the system's input [17]. The utilized control equations are as follows:

$$\boldsymbol{u}[k] = f(\boldsymbol{u}[k-1], \dots, \boldsymbol{u}[k-n_u+1], \boldsymbol{y}[k+1], \boldsymbol{y}[k], \dots, \boldsymbol{y}[k-n_y+1]) \quad (1)$$

where $\boldsymbol{u}[k]$ is the control signal at time k; $\boldsymbol{u}[k-1]$, ..., $\mathbf{u}[k - n_u + 1]$ is the delayed control signal; $\boldsymbol{y}[k+1]$ is the expected output signal which is equal to the reference signal at time k or $\boldsymbol{r}[k]$; $\boldsymbol{y}[k]$ is the output of the plant at time k; whereas $\boldsymbol{y}[k-1]$, ..., $\boldsymbol{y}[k-n_y+1]$ is the delayed output signal. In this case, n_u and n_y is the constants to represent the delay for input and output, respectively, and f is a predictive function of the neural network controller. This equation states that the control signal $\boldsymbol{u}[k]$ is a function of some previous inputs, expected output, current output and some delayed outputs.

The neural network inverse controller is trained by backpropagation learning algorithm using the real plant data as described in section 2, with $n_u = 2$ and $n_y = 2$. The training scheme for inverse controller is shown in Fig. 7.

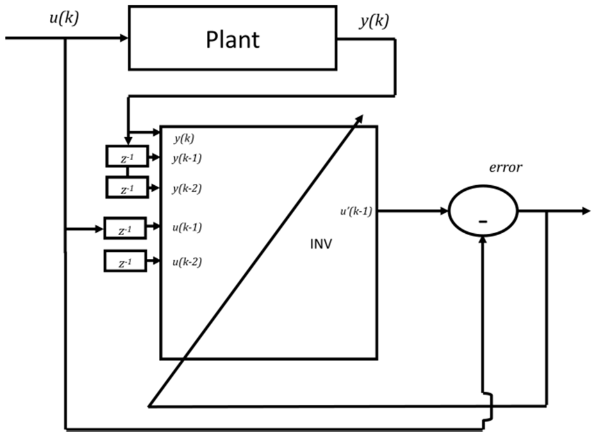

Fig. 7. Inverse controller training scheme

In this study, MLP network with one hidden layer were designed as the required neural network inverse controller. The network configuration is designed by considering the number of input and output of the controller. Here, since there are 2 plant inputs and 3 plant outputs, and $n_u = 2$ and $n_y = 2$, the number of controller input is equal to (2x2) + (3x3) = **13**. Meanwhile, the controller output is the control signal (or input) of the plant, *i.e.* PWM 1 and PWM 2. Therefore, the number of controller output is equal to **2**. Here, 3 different number of hidden neurons were investigated: **10**, **20**, and **26**. Thus, there were 3 investigated network configurations, *i.e. 13-10-2*, *13-20-2* and *13-26-2*.

Two Backpropagation learning algorithms namely the Levenberg Marquardt (LM) Backpropagation and the Bayesian Regularization (BR) Backpropagation were then utilized to train the controller. The two regularization techniques were used with backpropagation learning algorithm to get smaller errors, better response, and to minimize the possibility of overfitting. Both methods are developed based on the Bayesian approach. LM Backpropagation was developed for faster convergence, by using validation data along with the training process. BR Backpropagation has an objective function with several additional parameters to minimize the estimation errors and to obtain better generalization model [18].

IV. Results and Discussions

In the simulation, the obtained experimental data (Fig. 3, Fig. 4 and Fig. 5) is divided into training data, validation data and test data with the proportion of 75%, 5% and 20%, respectively. These proportions were chosen by considering the occurrence of outliers due to the GPS fault as clearly shown in Fig. 5.

A. Levenberg Marquardt Backpropagation Training

The results of Levenberg Marquardt (LM) Backpropagation training for *13-10-2*, *13-20-2* and *13-26-2* controller network configurations are shown in Fig. 8, Fig. 9, and Fig. 10, respectively. Fig. 8 shows that the training for *13-10-2* network configuration converged at epoch 5 and produced training and testing mean-squared errors (MSEs) of *0.0109* and *0.0182*, respectively. Fig. 9 shows that the training process for *13-20-2* network configuration stopped at 10 epochs and produced the best performance on the 4th epoch. The obtained training and testing MSEs for this configuration are *0.0075* and *0.0251*, respectively. Meanwhile, Fig. 10 shows that the training for *13-26-2* network architecture stopped at 2 epochs and produced training and testing MSEs of *0.0182* and *0.0527*, respectively.

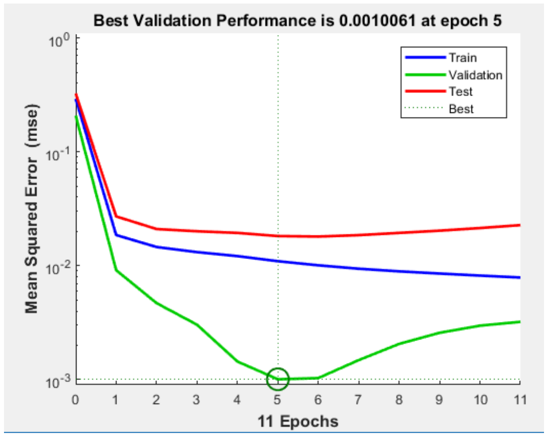

Fig. 8. LM Backpropagation training for *13-10-2 network configuration*

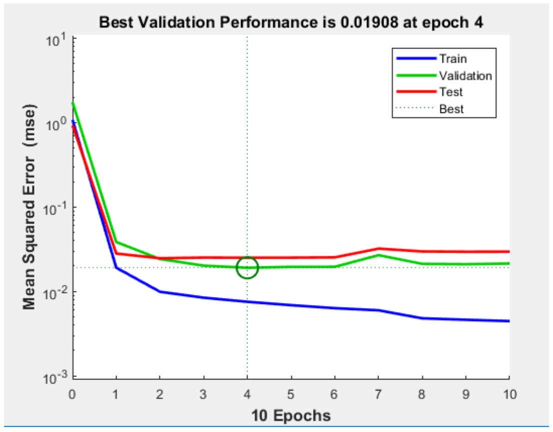

Fig. 9. LM Backpropagation training for *13-20-2 network configuration*

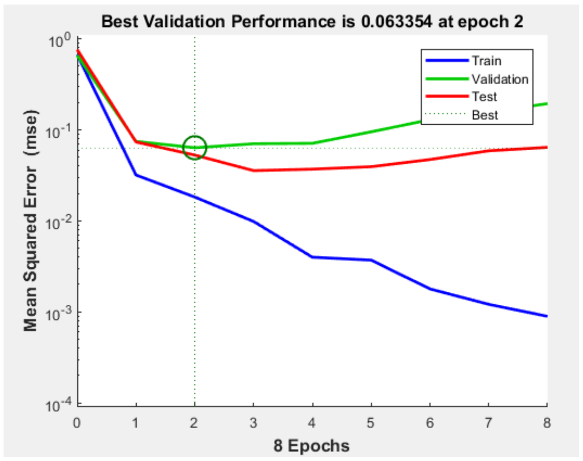

Fig. 10. LM Backpropagation training for *13-26-2 network configuration*

B. *Bayesian Regularization Backpropagation Training*

The results of Bayesian Regularization (BR) Backpropagation training for *13-10-2*, *13-20-2* and *13-26-2* controller network configurations are shown in Fig. 11, Fig. 12, and Fig. 13, respectively. Fig. 11 shows that the training for *13-10-2* network configuration converged at the 178th epochs and produced training and testing MSEs of *0.0173* and *0.0139*, respectively. Fig. 12 shows that the training for *13-20-2* network configuration converged on epoch 484 and the obtained training and testing MSEs for this configuration are *0.0146* and *0.0297*. The best training performance was obtained for *13-26-2* network configuration as depicted in Fig. 13. For this network architecture, training stopped at 1000 epochs and the obtained training and testing MSEs are *0.0064* and *0.0275*, respectively.

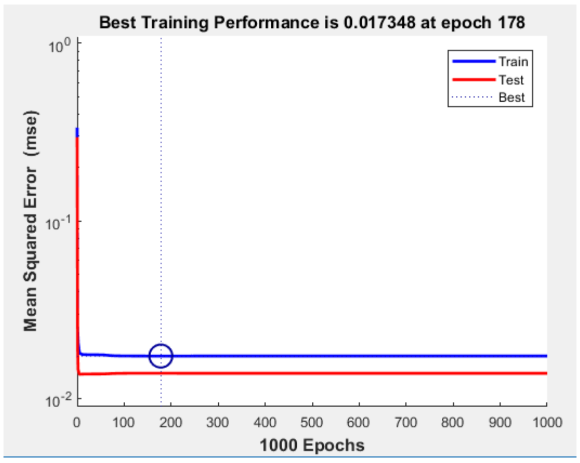

Fig. 11. BR Backpropagation training for *13-10-2 network configuration*

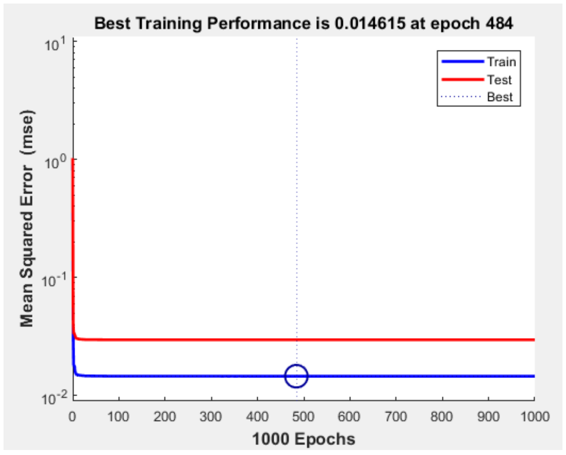

Fig. 12. BR Backpropagation training for *13-20-2 network configuration*

C. *Overall analysis*

The overall comparison of training results are shown in Table 1. From a series of simulations, it was found that the best training MSE was obtained for *13-26-2* system configuration which was trained using Bayesian Regularization Backpropagation algorithm (training MSE = *0.0064*). However, although the training MSE for this configuration is the lowest among all others, the testing MSE (of *0.0275*) is not as small as that for *13-10-2* network architecture (testing MSE = *0.0139*). This is due to training overfitting, where the network is too much trained so that it becomes too fit to the training data and not fit to the testing data. This phenomenon is also reflected in Fig. 13, where the training MSE decreases while the testing MSE increases with the epoch of training.

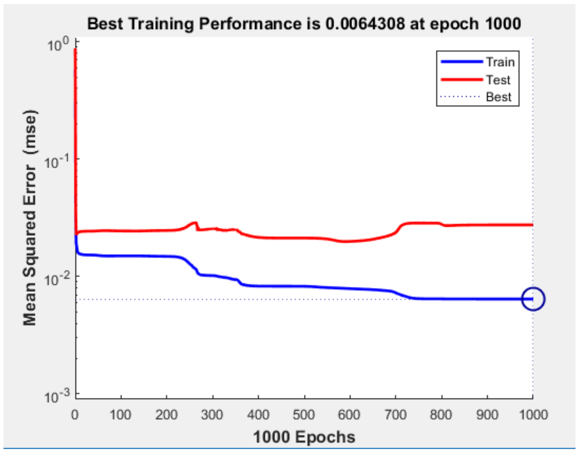

Fig. 13. BR Backpropagation training for *13-26-2 network configuration*

The simulation results revealed that the best controller network configuration was *13-10-2* which was trained using Bayesian Regularization Backpropagation algorithm. The corresponding training and testing MSEs for this configuration are *0.0173* and *0.0139*, respectively, which was obtained at the *178th* training epoch. This result serves as a preliminary evidence that backpropagation neural network-based controller can be applied as the controller of wheeled robots. Further study is required to analyze its real implementation by conducting real experiment.

TABLE I. OVERALL COMPARISON OF SIMULATION RESULTS

Algorithms	Configurations	Training MSE	Training Epoch	Testing MSE
Levenberg Marquardt Backpropagation	13-10-2	0.0109	5	0.0182
	13-20-2	0.0075	4	0.0251
	13-26-2	0.0182	2	0.0527
Bayesian Regularization Backpropagation	13-10-2	0.0173	178	0.0139
	13-20-2	0.0146	484	0.0297
	13-26-2	0.0064	1000	0.0275

One of the drawbacks of neural network-based controller is that it relies on the training data, *i.e.* the controller requires sufficient training data which covers all possible movements of the plant to be controlled. All of these training data should be utilized to train the controller prior to its real implementation, thus, it may require high computational cost in terms of training time. A possible solution to this issue is by using an online-training approach, where the neural network-based controller learns as it is being used to control the plant. However, again, the training cost should be well considered so that it does not disturb the real time behavior of the plant.

V. CONCLUSION

Simulations of neural network inverse controller training by using Levenberg Marquardt Backpropagation and Bayesian Regularization Backpropagation algorithms have been conducted to find the optimum controller network configuration for wheeled robot. The results show that the best performance is obtained by Bayesian Regularization Backpropagation algorithms for *13-10-2* network architecture. This result is an early proof that backpropagation neural network can be used as a controller system for wheeled robots, under the scheme of direct inverse control.

Further research are currently being conducted to analyze other network architectures by varying the numbers of hidden neurons, hidden layers, as well as the number of delays that will affect the number of input neurons.

ACKNOWLEDGMENT

This study is supported by the Ministry of Research and Higher Education of Indonesia and Bogor Agricultural University through *Penelitian Dasar Unggulan Perguruan Tinggi* (PDUPT) Research Grant 2018.

REFERENCES

[1] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," Journal of Intelligent and Robotic Systems: Theory and Applications. 2013.

[2] N. S. Naik, V. V. Shete, and S. R. Danve, "Precision agriculture robot for seeding function," in Proceedings of the International Conference on Inventive Computation Technologies, ICICT 2016, 2017.

[3] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," in Autonomous Robots, 2011.

[4] K. M. Wurm, C. Dornhege, B. Nebel, W. Burgard, and C. Stachniss, "Coordinating heterogeneous teams of robots using temporal symbolic planning," Auton. Robots, 2013.

[5] K. Priandana, A. Buono, and Wulandari, "Hexapod Leg Coordination using Simple Geometrical Tripod-Gait and Inverse Kinematics Approach," in 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2017, vol. 00, pp. 35–40.

[6] J. Gu, X. Ma, F. Liu, Y. Wang, and H. Ren, "Mathematical Modeling and Intelligent Algorithm for Multirobot Path Planning," Mathematical Problems in Engineering, 2017.

[7] K. Naveed and Z. H. Khan, "Adaptive path tracking control design for a wheeled mobile robot," in 2017 3rd IEEE International Conference on Control Science and Systems Engineering, ICCSSE 2017, 2017.

[8] N. G. M. Thao, D. H. Nghia, and N. H. Phuc, "A PID backstepping controller for two-wheeled self-balancing robot," in 2010 International Forum on Strategic Technology, IFOST 2010, 2010.

[9] B. Kusumoputro, H. Suprijono, M. A. Heryanto, and B. Y. Suprapto, "Development of an attitude control system of a heavy-lift hexacopter using Elman recurrent neural networks," in 2016 22nd International Conference on Automation and Computing, ICAC 2016: Tackling the New Challenges in Automation and Computing, 2016.

[10] H. A. F. Almurib, A. A. M. Isa, and H. M. A. A. Al-Assadi, Direct Neural Network Control via Inverse Modelling: Application on Induction Motors, Artificial Neural Networks - Industrial and Control Engineering Applications, Prof. Kenji Suzuki. InTech, 2011.

[11] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," Computer, vol. 29, no. 3. pp. 31–44, 1996.

[12] B. Kusumoputro, K. Priandana, and W. Wahab, "System identification and control of pressure process rig® system using backpropagation neural networks," ARPN J. Eng. Appl. Sci., vol. 10, no. 16, pp. 7190–7195, 2015.

[13] H. Suprijono and B. Kusumoputro, "Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control," J. Telecommun. Electron. Comput. Eng., 2017.

[14] M. A. Heryanto, H. Suprijono, B. Y. Suprapto, and B. Kusumoputro, "Attitude and altitude control of a quadcopter using neural network based direct inverse control scheme," Adv. Sci. Lett., 2017.

[15] K. Priandana, W. Wahab, and B. Kusumoputro, "Comparison of neural networks based direct inverse control systems for a double propeller boat model," in ACM International Conference Proceeding Series, 2016.

[16] P. J. Werbos, “Neural networks for control and system identification,” Proc. 28th IEEE Conf. Decis. Control., pp. 260–265, 1989.

[17] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks.,” IEEE Trans. Neural Netw., vol. 1, no. 1, pp. 4–27, 1990.

[18] M. Kayri, “Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data,” Math. Comput. Appl., 2016.