Neurointerface Using an Online Feedback-Error Learning Based Neural Network for Nonholonomic Mobile Robots

Hyun-Dong Lee*, Keigo Watanabe*, Sang-Ho Jin**, Rafiuddin Syam* and Kiyotaka Izumi*

Department of Advanced Systems Control Eng., Graduate School of Science and Engineering,
Saga University, 1-Hojomachi, Saga 840-8502, Japan
(Tel: +81-952-28-8602; Fax: +81-952-28-8587; E-mail: s_2hd@cnu.ac.kr, {watanabe,izumi}@me.saga-u.ac.jp)
Department of Mechanical Engineering, Doowon Technical College
678, Jangwon-ri, Juksan-myon, Ansung-shi, Kyonggi-do, 456-718 Korea
(Tel: +81-31-670-7134; E-mail: shjin@doowon.ac.kr)

Abstract: In this study, a method of designing a neurointerface using neural network (NN) is proposed for controlling nonholonomic mobile robots. According to the concept of virtual master-slave robots, in particular, a partially stable inverse dynamic model of the master robot is acquired online through the NN by applying a feedback-error learning method, in which the feedback controller is assumed to be based on a PD compensator for such a nonholonomic robot. A tracking control problem is demonstrated by some simulations for a nonholonomic mobile robot with two-independent driving wheels.

Keywords: Neurointerface, Neural networks, Nonholonomic mobile robots, Online feedback-error learning

1. Introduction

Up to now, there have been a lot of efforts to solve a trajectory tracking problem using a geometrical model of mobile robots under nonholonomic condition. Widrow and Lamego [1] proposed a neurointerface approach, whose method is composed mainly of two parts: one is an inverse system realized by neural network (NN) to generate a feedforward control input according to a reference value or the output of a reference model and the other is a feedback mechanism to suppress the effect of disturbance due to the change of initial state, mapping errors of NN, etc.

Mobile robot navigation can be classified into three basic problems: tracking a reference trajectory, following a path, and point stabilization. Fierro and Lewis [2],[3] proposed a control of a nonholonomic mobile robot, in which a control structure that makes the integration of a kinematic controller and an NN computed-torque controller possible is presented for a nonholonomic mobile robot. A combined kinematic/torque control law is developed using backstepping and its stability is guaranteed by Lyapunov theory. That control algorithm can be applied to the above three basic nonholonomic navigation problems. Moreover, the NN controller proposed in their work can deal with unmodeled bounded disturbances and unstructured unmodeled dynamics in the vehicle. Online NN weight tuning algorithm need not require any offline learning, yet guarantee small tracking errors and bounded control signals were utilized.

Syam et al. [4] have already proposed a method for constructing a feedforward part in the framework of neurointerface for a nonholonomic mobile robot by applying a concept of virtual master-slave system. Note however that the method proposed by Syam et al. was impossible to deal with a trajectory tracking for a suddenly changed mass by using an NN trained offline, even though a PD feedback compensator was introduced.

Therefore, in this research, a new method of designing a neurointerface using an NN is proposed for controlling nonholonomic mobile robots. According to the concept of virtual master-slave robots, in particular, a partially stable inverse dynamic model of the master robot is acquired online through the NN by applying a feedback-error learning method, in which the feedback controller is assumed to be based on a PD compensator for such a nonholonomic robot. Topalov et al. [5] studied a similar control system to Fierro and Lewis by using a feedback-error learning method. However, their fuzzy NN was very complicated compared to the present NN.

2. Structure of Mobile Robot

A nonholonomic mobile robot system attaching two-independent driving wheels in the rear and a caster adjustable to any free direction in the front is assumed as Fig. 1. In the figure, C point shows the mass center of the robot, and P point shows the center of two rear wheel axle. The offset distance between P point and C point is denoted by d and the distance between two rear wheels, i.e., tread is given by $2R$, and the radius of the rear wheel is shown by r. A nonholonomic mobile robot can be described by

$$M(\boldsymbol{q})\boldsymbol{q} + V(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = B(\boldsymbol{q})\boldsymbol{\tau} - A^T(\boldsymbol{q})\boldsymbol{\lambda} \tag{1}$$

where $\boldsymbol{q}$ is the generalized coordinates, $\boldsymbol{q} = [x_c \;\; y_c \;\; \theta]^T$, (x_c, y_c) is the robot position of the center of gravity, θ is the azimuth, M is an inertia matrix, V is the centrifugal and Coriolis matrix, B is the input distribution matrix, A is the matrix associated with the constraints, and $\boldsymbol{\lambda}$ is the vector of constraint forces [3],[6].

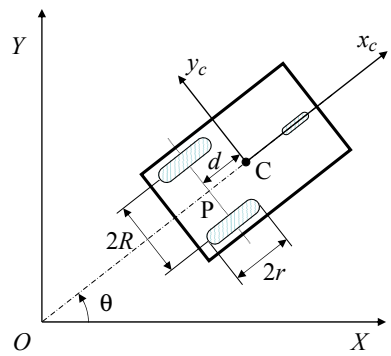

Fig. 1. The schematic diagram of nonholonomic mobile robot.

3. Torque Conversion Model

Generally, Eq. (1) can be shown in steering model of the mobile robot:

$$\begin{aligned}\boldsymbol{\tau}(t) &= \boldsymbol{g}_M(\dot{\boldsymbol{v}}(t)) \\ &= \boldsymbol{g}_M([\boldsymbol{v}(t)-\boldsymbol{v}(t-1)]/\Delta t)\end{aligned} \tag{2}$$

Here, $\boldsymbol{v}(t) = [v(t)\ \ \dot{\theta}(t)]^T$ shows the forward speed of the robot at time t, and $\boldsymbol{g}_M$ is a function that converts the forward acceleration to the input torque $\boldsymbol{\tau}(t)$. Δt is the sampling time. Moreover, $\boldsymbol{v}(t)$ of the robot can be shown in the kinematic relation through Fig. 1:

$$\dot{\boldsymbol{q}}(t) = J(\theta(t))\boldsymbol{v}(t), \quad J(\theta(t)) = \begin{bmatrix}\cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1\end{bmatrix} \tag{3}$$

or

$$\boldsymbol{v}(t) = J^+(\theta(t))\left[\frac{\boldsymbol{q}(t)-\boldsymbol{q}(t-1)}{\Delta t}\right]$$

where, $J^+(\theta(t))$ denotes the pseudo-inverse matrix of $J(\theta(t))$. The steering model can be dominated by a dynamic property according to the forward speed change and the angular acceleration of the robot. If I is the moment of inertia around the P point and m is the mass of the robot, then the relation of the torque of the drive wheels and the acceleration of the robot is given by [6]

$$\begin{bmatrix} m & 0 \\ 0 & I-md^2 \end{bmatrix}\begin{bmatrix}\dot{v} \\ \dot{\theta}\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & 1 \\ R & -R\end{bmatrix}\begin{bmatrix} {}_r \\ {}_l\end{bmatrix} \tag{4}$$

Therefore, Eq. (4) is reduced to

$$\boldsymbol{\tau}(t) = \begin{bmatrix} {}_r \\ {}_l\end{bmatrix} = r\begin{bmatrix}1 & 1 \\ R & -R\end{bmatrix}^{-1}\begin{bmatrix} m & 0 \\ 0 & I-md^2\end{bmatrix}\dot{\boldsymbol{v}}(t) \tag{5}$$

4. Construction of Neurointerface

4.1. PD feedback controller

The PD feedback controller is used so that $\boldsymbol{q}(t) = [x_c\ \ y_c\ \ \theta]^T$ tracks $\boldsymbol{q}_r(t) = [x_r\ \ y_r\ \ \theta_r]^T$, which is the reference trajectory of the robot. If a positional error of the robot is defined as $\boldsymbol{e}(t)$, the input velocity $\dot{\boldsymbol{q}}_{PD}(t)$ in the coordinate system of the slave robot can be given by

$$\dot{\boldsymbol{q}}_{PD}(t) = K_p\boldsymbol{e}(t) + K_d\dot{\boldsymbol{e}}(t) \tag{6}$$

with

$$\boldsymbol{e}(t) = \boldsymbol{q}_r(t) - \boldsymbol{q}(t), \quad \dot{\boldsymbol{e}}(t) = \dot{\boldsymbol{q}}_r(t) - \dot{\boldsymbol{q}}(t)$$

where K_p and K_d are the proportional and derivative gain matrices, respectively.

Using Eq. (3), the input velocity in the coordinate system of the master robot (or steering model) can be obtained by

$$\boldsymbol{v}_{PD}(t) = J^+(\theta_r(t))\dot{\boldsymbol{q}}_{PD}(t) \tag{7}$$

Finally, using Eq. (5) and Eq. (7) yields

$$\boldsymbol{\tau}_{PD}(t) = T_I\dot{\boldsymbol{v}}_{PD}(t) \tag{8}$$

where it is assumed that $\dot{\boldsymbol{v}}_{PD}(t) = [\boldsymbol{v}_{PD}(t) - \boldsymbol{v}_{PD}(t-1)]/\Delta t$ and T_I is a formal torque transformation matrix whose diagonal element represents any fictitious moment of inertia or mass, e.g., $T_I = \mathrm{diag}\,(1,1)$ for the simplicity.

4.2. Online feedback-error learning

The block diagram to which the position and azimuth of a real-time robot can track the reference trajectory is shown in Fig. 2.

The desired torque vector $\boldsymbol{\tau}_r(t)$ can be obtained by using the desired velocity vectors $\boldsymbol{v}_r(t)$ and $\boldsymbol{v}_r(t-1)$, in which the $\boldsymbol{v}_r(t)$ is obtained by using the static transformation $J^+(\theta_r(t))$ and $\dot{\boldsymbol{q}}_r(t)$. Here, if the velocity of Eq. (2) is assumed to be constant, then the value of $\boldsymbol{\tau}(t)$ becomes 0. However, the torque is impossible to become 0 because any disturbance of friction etc. actually exists in the fields, so that it needs to include a disturbance torque $\boldsymbol{\tau}_d(t)$.

The feedforward desired torque $\boldsymbol{\tau}_r(t)$ is now generated by using NN learned by the online feedback-error learning. After calculating the Eq. (8) to generate the value $\boldsymbol{\tau}_{PD}(t)$ provided by the PD controller, the NN should be learned by using $\boldsymbol{\tau}(t)$ value. Ultimately the NN is learned so that the value of $\boldsymbol{\tau}_{PD}(t)$ becomes 0.

The actual coordinate $\boldsymbol{q}(t)$ is simulated by using Eq. (1) so that the input torque $\boldsymbol{\tau}(t)$ to the robot is given by

$$\boldsymbol{\tau}(t) = \boldsymbol{\tau}_r(t) + \boldsymbol{\tau}_{PD}(t) \tag{9}$$

4.3. Design of neural network

If the disturbance torque $\boldsymbol{\tau}_d(t)$, as a non-zero torque, is considered, then Eqs. (2), (4) and (5) are rewritten as follows:

$$\boldsymbol{\tau}_r(t) = \boldsymbol{g}_M([\boldsymbol{v}_r(t) - \boldsymbol{v}_r(t-1)]/\Delta t,\ \boldsymbol{\tau}_d(t)) \tag{10}$$

$$\begin{bmatrix} m & 0 \\ 0 & I-md^2\end{bmatrix}\begin{bmatrix}\dot{v}_r \\ \dot{\theta}_r\end{bmatrix} + \boldsymbol{\tau}_d = \frac{1}{r}\begin{bmatrix}1 & 1 \\ R & -R\end{bmatrix}\begin{bmatrix} {}_r \\ {}_l\end{bmatrix} \tag{11}$$

$$\begin{aligned}\boldsymbol{\tau}_r(t) &= \begin{bmatrix} {}_{rr} \\ {}_{lr}\end{bmatrix} \\ &= r\begin{bmatrix}1 & 1 \\ R & -R\end{bmatrix}^{-1}\left(\begin{bmatrix} m & 0 \\ 0 & I-md^2\end{bmatrix}\dot{\boldsymbol{v}}_r(t) + \boldsymbol{\tau}_d\right) \\ &= \begin{bmatrix} \frac{1}{2}mr\dot{v}_r(t) + \frac{1}{2}r\ {}_{rd} + \frac{r}{2R}(I-md^2)\theta_r(t) + \frac{r}{2R}\ {}_{ld} \\ \frac{1}{2}mr\dot{v}_r(t) + \frac{1}{2}r\ {}_{rd} - \frac{r}{2R}(I-md^2)\theta_r(t) - \frac{r}{2R}\ {}_{ld}\end{bmatrix}\end{aligned} \tag{12}$$

Using Eq. (12), the NN for the online feedback-error learning can be composed as shown in Fig. 3. The input layer consists of six units for the inputs $x_i, i = 1, ..., 6$, the hidden layer consists of two hidden units for the hidden outputs $o_j, j = 1, 2$, and the output layer consists of two units for the outputs, ${}_k, k = 1, 2$.

To train the NN online, we consider the minimization of a squared output error such as,

$$J = \frac{1}{2}\sum_{k=1}^{l(=2)}\varepsilon_k^2(t), \quad \varepsilon_k(t) = {}_k(t) - {}_{kr}(t) \tag{13}$$

where ${}_k$ can be regarded as the teaching signal for the kth output of the NN.

It is further assumed that the weights w_{ij} in general form between the input and hidden layers are unknown, and the weights s_{jk} in general form between the hidden and output layers are known, fixed values. Also it is easy to find that

$${}_{kr}(t) = \sum_{j=1}^{m(=2)} s_{jk}o_j, \quad o_j = \sum_{i=1}^{n(=6)} w_{ij}x_i \tag{14}$$

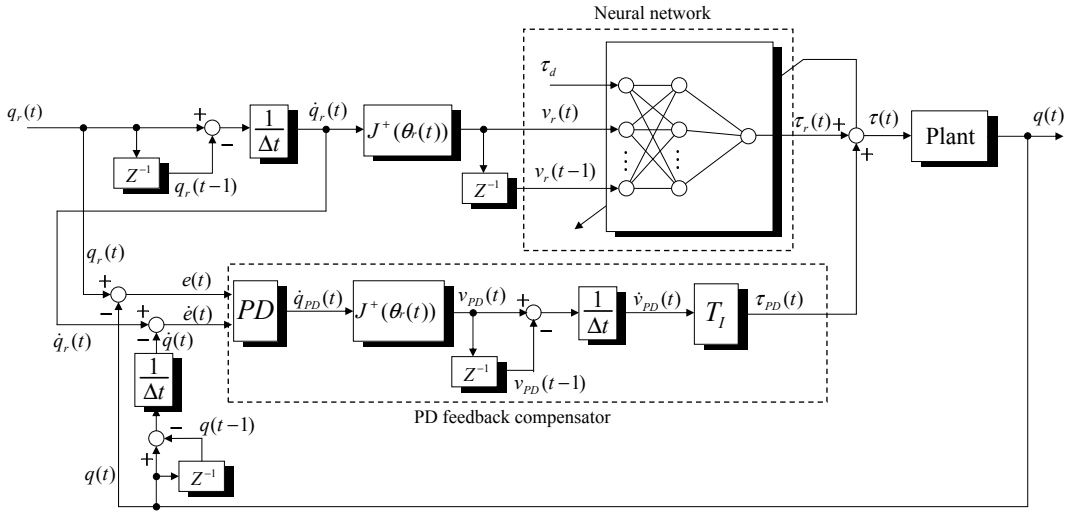

Fig. 2. The neurointerface using online feedback-error learning algorithm.

Then, the gradient of the cost function J with respect to the weight w_{ij} to be learned is derived as

$$\frac{\partial J}{\partial w_{ij}} = \sum_{k=1}^{l} \frac{\partial J}{\partial \varepsilon_k} \frac{\partial \varepsilon_k}{\partial w_{ij}} = \sum_{k=1}^{l} \varepsilon_k \frac{\partial \varepsilon_k}{\partial w_{ij}} \tag{15}$$

On the other hand, we can find that

$$\begin{aligned} \frac{\partial \varepsilon_k}{\partial w_{ij}} &= \frac{\partial \varepsilon_k}{\partial \ _{kr}} \frac{\partial \ _{kr}}{\partial w_{ij}} = -\frac{\partial \ _{kr}}{\partial w_{ij}} \\ &= -\frac{\partial \ _{kr}}{\partial o_j} \frac{\partial o_j}{\partial w_{ij}} = -s_{jk} x_i \end{aligned} \tag{16}$$

Therefore, the incremental value of learning weights can be defined as

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial J}{\partial w_{ij}} = \eta x_i \sum_{k=1}^{l} s_{jk} \varepsilon_k \tag{17}$$

or

$$w_{ij}(t+1) = w_{ij}(t) + \eta x_i \sum_{k=1}^{l} s_{jk} [\ _k(t) - \ _{kr}(t)] \tag{18}$$

where η denotes the small learning rate.

Note here that $w_{11} = w_1$, $w_{22} = w_2$, $w_{31} = w_3$, $w_{42} = w_4$, $w_{51} = w_5$, $w_{62} = w_6$, $s_{11} = s_1$, $s_{12} = s_2$, $s_{21} = s_3$, and $s_{22} = s_4$ to simplify the notations.

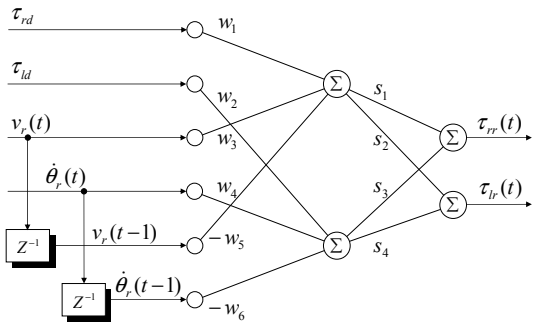

Fig. 3. Neural network structure for the steering model.

5. Simulations

The $\boldsymbol{\tau}(t)$ given by virtual master robot can be provided to the actual (slave) robot:

$$\boldsymbol{q} = M(\boldsymbol{q})^{-1}[B(\boldsymbol{q})\boldsymbol{\tau} - A^T(\boldsymbol{q})\boldsymbol{\lambda} - V(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}] \tag{19}$$

Moreover, the acceleration function obtained by Eq. (19) is substituted for the Runge-Kutta-Gill method to generate the robot position and azimuth, and their rate vector.

To obtain the inverse mapping of the virtual master robot, the neurointerface was executed as shown in Fig. 2. The physical parameters used for the robot simulated here are shown in Table 1, where $K_p = \text{diag}\,(k_p, k_p, k_p)$ and $K_d = \text{diag}\,(k_d, k_d, k_d)$.

When the NN was learned, the initial values of w_i $(i = 1, 2, ..., 6)$ are set as random values. The connection weights w_i $(i = 1, 2, ..., 6)$ connecting the input layer and the hidden layer have their ideal values of $w_1 = r/2$, $w_2 = r/2R$, $w_3 = w_5 = mr/2\Delta t$, $w_4 = w_6 = r(I - md^2)/2R\Delta t$ and the connection weights s_n $(n = 1, 2, ..., 4)$ connecting the hidden layer and the output layer have all 1 except that $s_4 = -1$.

Fig. 4 shows the learning history of connection weights for $\eta = 0.3$, where the corresponding learned connection weights are tabulated in Table 2.

In this simulation, the velocity of the robot was restricted within 5 [m/s], and the torque of the driving wheel was also restricted within 5 [kgm].

Fig. 5 shows the result of the running simulation through a linear trajectory (case 1). The initial $\boldsymbol{q}$ value was set to $[0.3, 0.5, \pi/4]$.

When the value of connection weights appeared as Table 2, the $\boldsymbol{\tau}_{PD}(t)$ was close to 0 such as $\ _{rPD} = 8.14 \times 10^{-6}$ [kgm] and $\ _{lPD} = 5.55 \times 10^{-6}$ [kgm] in the simulation result, which is suitable for the purpose of the algorithm. Moreover, the final trajectory errors of the robot appeared with $e_x (= x_r - x_c) = 5.16 \times 10^{-3}$ [m], $e_y (= y_r - y_c) = 6.53 \times 10^{-3}$ [m], and $e_\theta (= \theta_r - \theta) = 3.94 \times 10^{-3}$ [rad].

Fig. 6 is a result of simulation for a curved trajectory (case 2). The initial $\boldsymbol{q}$ value was set to $[0.5, 0.5, \pi/2]$. In the simulation, the $\boldsymbol{\tau}_{PD}(t)$ was almost close to 0 such as $\ _{rPD}(t) = 8.14 \times 10^{-6}$ [kgm] and $\ _{lPD}(t) = 5.55 \times 10^{-6}$ [kgm] and the connection weights converged as Table 3. The final trajectory errors of the

Table 1. The design and physical parameters for simulations.

Item	NN			Physical parameters						PD	
parameter	η	$_{rd}$ [kgm]	$_{ld}$ [kgm]	m [kg]	I [kgm^2]	R [m]	r [m]	d [m]	s.time [s]	k_p	k_d
value	0.3	0.5	0.5	10	5	0.5	0.05	0.2	0.02	2	0.4

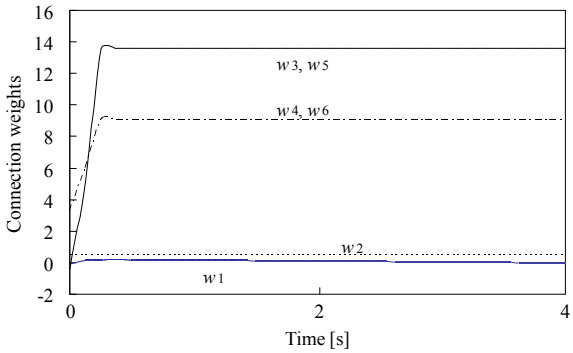

Fig. 4. Learning history of connection weights w_i for $\eta = 0.3$.

Table 2. Connection weights learned for case 1.

	w_1	w_2	w_3, w_5	w_4, w_6
Ideal	0.025	0.05	12.5	11.5
Learned	0.001	0.55	13.62	9.05

robot appeared with $e_x = 6.74 \times 10^{-2}$ [m], $e_y = 5.19 \times 10^{-2}$ [m], and $e_\theta = 7.38 \times 10^{-3}$ [rad].

For the accuracy of the trajectory tracking, the linear trajectory (case 1) is superior to the curved trajectory (case 2).

6. Conclusions

This research has devoted to develop a control algorithm that can track the trajectory of a nonholonomic mobile robot with two-independent driving wheels.

The method of neurointerface based on a virtual master-slave robot was adopted and an online feedback-error learning based neural network was used in this study to reduce the trajectory errors of the robot. The validity of constructing the NN and the performance of the algorithm were verified by simulations.

References

[1] B. Widrow and M. M. Lamego, "Neurointerfaces," *IEEE Trans. on Control Systems Tech.*, vol. 10, no. 2, pp. 221–228, 2002.

[2] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics," *Journal of Robotic Systems*, vol. 14, no. 3, pp. 149–163, 1997.

[3] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot using neural networks," *IEEE Trans. on Neural Networks*, vol. 9, no. 4, pp. 589–600, 1998.

[4] R. Syam, K. Watanabe, and K. Izumi, "Concept of virtual master-slave system and its application to the design of neuro-interface," in *Proc. Intl. SICE Annual Conf.*, Sapporo, Japan, August 4–6, 2004, pp. 1108–1113, 2004.

[5] A. V. Topalov, J. H. Kim, and T. P. Proychev, "Fuzzy-net control of non-holonomic mobile robot using evolutionary feedback-error-learning," *Robotics and Autonomous Systems*, vol. 23, pp. 187–200, 1998.

[6] K. Izumi, R. Syam, and K. Watanabe, "Neural network based disturbance canceller with feedback error learning for non-holonomic mobile robots," in *Proc. of the 4th Int. Symposium on Advanced Intelligent Systems (ISIS 2003)*, pp. 443–446, 2003.

Table 3. Connection weights learned for case 2.

	w_1	w_2	w_3, w_5	w_4, w_6
Ideal	0.025	0.05	12.5	11.5
Learned	0.001	0.54	13.44	10.71

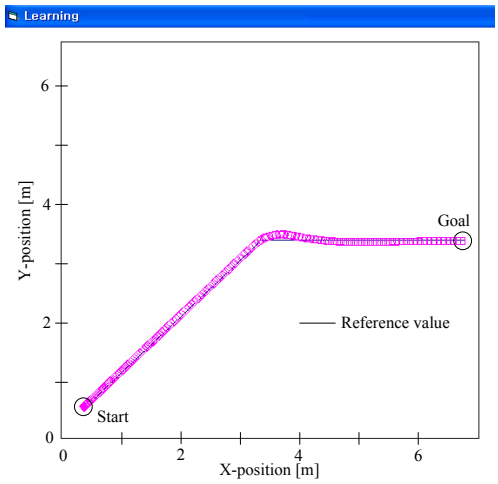

Fig. 5. The simulation result of nonholonomic mobile robot for case 1.

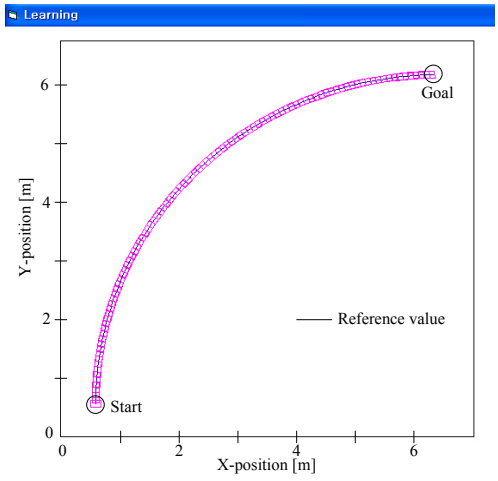

Fig. 6. The simulation result of nonholonomic mobile robot for case 2.