Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 8014019, 19 pages
https://doi.org/10.1155/2018/8014019

Research Article

Robust Optimal Navigation Using Nonlinear Model Predictive Control Method Combined with Recurrent Fuzzy Neural Network

Qidan Zhu, Yu Han, Chengtao Cai, and Yao Xiao

College of Automation, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Yu Han; hanyu@hrbeu.edu.cn

Received 5 February 2018; Revised 5 June 2018; Accepted 18 July 2018; Published 13 August 2018

Academic Editor: Julien Bruchon

This paper presents a novel navigation strategy of robot to achieve reaching target and obstacle avoidance in unknown dynamic environment. Considering possible generation of uncertainty, disturbances brought to system are separated into two parts, i.e., bounded part and unbounded part. A dual-layer closed-loop control system is then designed to deal with two kinds of disturbances, respectively. In order to realize global optimization of navigation, recurrent fuzzy neural network is used to predict optimal motion of robot for its ability of processing nonlinearity and learning. Extended Kalman filter method is used to train RFNN online. Moving horizon technique is used for RFNN motion planner to guarantee optimization in dynamic environment. Then, model predictive control is designed against bounded disturbances to drive robot to track predicted trajectories and limit robot's position in a tube with good robustness. A novel iterative online learning method is also proposed to estimate intrinsic error of system using online data that makes system adaptive. Feasibility and stability of proposed method are analyzed. By examining our navigation method on mobile robot, effectiveness is proved in both simulation and hardware experiments. Robustness and optimization of proposed navigation method can be guaranteed in dynamic environment.

1. Introduction

During past decades, navigation which can fully reflect artificial intelligence and automatic ability of robot has been an attractive topic [1]. Autonomous navigation is realized in kinds of robots like ground mobile vehicles, unmanned underwater vehicle (UUV), unmanned aircraft, etc. [2–5]. These robots with navigation technology have been applied popularly in various fields such as industry, public service places, transportation, and military [6]. Generally, navigation structure is made up of three main components, perception, decision, and control. Different sensors, such as lidar, inertial measurement unit, GPS, and visual camera, have been used for this purpose generating a series of solutions. This is beyond this paper's scope. The main problems that this paper prefers to solve are decision and control under assumption of known perception capability.

Navigation of robot in objective world is complex for the reasons of existing various uncertainty and nonlinearity of system [7, 8]. Disturbance or perturbation caused by uncertainty brings challenge to design navigation system. Many inevitable factors lead to uncertainty like dynamic circumstances, sensor noises, and model error [9]. They are coupled and impact together on system. In order to achieve real-time navigation and guarantee safety of robot taking obstacles into account, nonlinear optimal program of robot motion needs to be carried out [10]. However, to make system stable and robust under this condition, control strategy can be tedious and time-consuming. So, effective method should be designed to deal with optimal robust problem of the nonlinear system.

Fuzzy logic control is well suited to control robot for its accurate calculation capability and inference capability under uncertainty [11]. Many researchers have implemented this method to deal with navigation of robot. Wang [12] proposed a real-time fuzzy logic control in unknown environment to achieve obstacle avoidance. However, fuzzy logic

based method is too sample to deal with complex problem and the structure is less of flexibility. Neural network based approaches have been widely employed in closed-loop control for their strong nonlinear approximation and self-learning capability [13]. Many researches have been done on feedforward multilayer perception neural network for classification, recognition, and function approximation. In [14], recurrent neural network is used to solve optimal navigation problem of multirobot. By constructing formation and using penalty method, navigation becomes convex optimization problem. Recently, fuzzy logic and recurrent neural network structure are combined to form a new structure, i.e., recurrent fuzzy neural network (RFNN). RFNN combines both advantages of neural network and fuzzy logic and earns good nonlinear control capability [15]. In [16], interval type 2 fuzzy neural network is used to control robot. By tuning parameters of structure with genetic algorithm, stable navigation is realized. However, optimization of navigation is not guaranteed and disturbances are not considered in their method.

Model predictive control (MPC) has been successfully applied to process industries during past decades for its online optimization programming ability [17]. At each time-step, MPC obtains a finite sequence of control by solving a constrained, discrete-time, optimal control problem over prediction horizon. Current control variable of the sequence is regarded as control law to be applied to the system. By solving the optimal control problem within one sampling interval, MPC can effectively manage MIMO system and hard constraint system. Robust nonlinear MPC (RNMPC) is an active area of research and can easily handle constraint satisfaction against dynamic uncertainty [18]. If a controlled system is ISS with respect to bounded uncertainties, it has a stability margin and is therefore stable with respect to unmodeled dynamics [19]. Many researches have been published using MPC method realizing optimal navigation [20]. In [21], a dual-layer architecture is proposed using MPC and maximum likelihood estimation method to deal with navigation in uncertainty environment. In contrast to our method, it is a kind of probabilistic framework based method. In [22], MPC is also developed to micro air vehicle system generating real-time optimal trajectory in unknown and low-sunlight environments. In [23], autopilot controlling the nonlinear yaw dynamics of an unmanned surface vehicle is designed using MPC combined with multilayer neural network.

Inspired by methods mentioned above, a dual-layer control system is designed in this paper. Considering nonlinear nonconvex property of navigation in dynamic environment, RFNN based planner is used to program optimal trajectory of robot. Many methods can be used to train RFNN like back-propagation (BP), genetic algorithm, evolutionary strategy, and particle swarm optimization [24]. Extended Kalman filter (EKF) has attracted many researchers for its characteristics of fast convergence, little limitation in training data, and good accuracy in mathematics process [25]. So, in this paper EKF is used to train RFNN to achieve optimal navigation. A kind of nonlinear robust MPC is then designed to generate actual control command of robot to drive robot to track predictions. Considering that there exists intrinsic error between known model and actual model of navigation system of robot, online estimation method needs to be designed. Learning based controller is more adaptive in practical application [26]. By using online data, accurate approximation of the true system model can be constructed. These learning methods mainly consist of Gaussian process and iterative learning methods [27]. In this paper, a specific application based iterative learning method is designed to update system model.

The rest of the paper is structured as follows. Section 2 introduces description of problem settled in this paper. Section 3 describes the working of proposed navigation strategy. Section 4 proves stability and feasibility of proposed method. Section 5 illustrates experiments and results and finally Section 6 concludes the proposed scheme.

2. Problem Description

In this paper, navigation of robot is realized in unknown environment and the system is nonlinear with constraints. Robust and optimal navigation should be achieved considering dynamic environment and disturbances.

Consider the following discrete nonlinear model of robot system:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{w}_k^1 \\ \mathbf{y}_k &= \mathbf{h}\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{w}_k^2, \end{aligned} \tag{1}$$

where $\mathbf{u}_k \in \mathbb{R}^{n_1}$ is control input and $\mathbf{x}_k, \mathbf{x}_{k+1} \in \mathbb{R}^{n_2}$ are system states at time k and $k+1$, respectively. $\mathbf{y}_k \in \mathbb{R}^{n_3}$ is measured output at time k and $\mathbf{w}_k^1 \in \mathbb{R}^{n_2}, \mathbf{w}_k^2 \in \mathbb{R}^{n_3}$ are process and measurement disturbances, respectively. $\mathbf{f}(\bullet)$ and $\mathbf{h}(\bullet)$ are nonlinear functions where $\mathbf{f}(\bullet)$ is model of system and $\mathbf{h}(\bullet)$ is model of observation. Realistic physical plants are strictly proper, so output function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ can be simplified as $\mathbf{h}(\mathbf{x}_k)$.

In (1), $\mathbf{w}^1$ is model error caused disturbance while $\mathbf{w}^2$ mainly contains sensor noise and dynamic environment caused disturbance. Among these uncertainties, dynamic environment changes in random leading to unboundedness of disturbance. On the other hand, model error and sensor noise are bounded and normally are Gaussian that can be estimated. To design useful control method decomposition and combination are used for disturbances as

$$\mathbf{w}^2 = \mathbf{w}'^2 + \mathbf{w}''^2. \tag{2}$$

Feedback controller is usually designed using $\mathbf{y}$ to generate control input $\mathbf{u}$, so (1) changes as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{w}_k^1 + \mathbf{w}_k'''^2 \\ \mathbf{y}_k &= \mathbf{h}\left(\mathbf{x}_k\right) + \mathbf{w}_k'^2, \end{aligned} \tag{3}$$

where $\mathbf{w}''' = \mathbf{f}(\mathbf{C}(\mathbf{w}''))$. $\mathbf{C}(\bullet)$ represents controller. Then, disturbances are separated into two parts, bounded disturbance represented as $\boldsymbol{\Delta}_{1k} = \mathbf{w}_k^1 + \mathbf{w}_k'''^2$ and unbounded disturbance represented as $\boldsymbol{\Delta}_{2k} = \mathbf{w}_k'^2$.

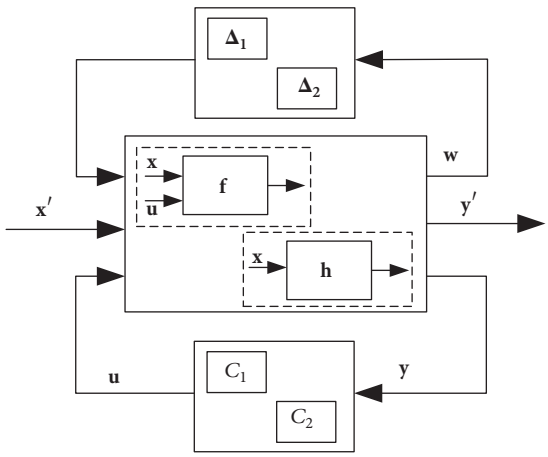


Figure 1: Control system considering disturbances.

To achieve navigation, suitable controller should be designed. Considering disturbances divided into two items, we separate strategy into two phases. Real-time program of optimal trajectory is designed at first and robust control of robot is then designed to track the optimal trajectory. In program phase, recurrent fuzzy neural network with moving horizon is designed to plan optimal motion to get robust performance considering $\mathbf{\Delta}_2$ and can be illustrated as C_1. After $\mathbf{\Delta}_2$ is limited, robust model predictive control method is designed in tracking period to deal with $\mathbf{\Delta}_1$ and can be illustrated as C_2. $\mathbf{\Delta}_1$ is decoupled from $\mathbf{\Delta}_2$, so $\mathbf{\Delta}_2$ is not considered during tracking. Then, a controller with dual-layer structure is formed to deal with two kinds of disturbances, respectively. Structure is shown in Figure 1. By separating controller into two parts, bounded and unbounded disturbances are suitably dealt with so that optimal robust performance is realized. Further introduction of control system is in next section.

3. Control System Design

To realize robust optimal navigation of robot, rational framework of control system is designed in this section. Robust nonlinear model predictive controller and recurrent fuzzy neural network based planner are combined to make up the whole control system. Flow diagram is shown in Figure 3 which is extension of Figure 2. Figure 3 emphasizes each part's function in navigation strategy and is consistent with Figure 2 which mainly explains relationship of uncertainty, controller, and nominal system.

The system is made up of nonlinear planner, robust tracker, and local plant. As mentioned in Section 2, there exist kinds of uncertainty bringing disturbances to robot. In order to improve the performance of robot avoiding obstacles and reaching the target, feedback loops that can be used to construct constraints of system are added. Extended Kalman filter is used to train RFNN online to program optimal motion of plant. Iterative online leaning method is used to make system adaptive to disturbance improving performance of MPC.

3.1. RFNN Based Nonlinear Program. Objective function and constraints should be set to realize motion program of navigation. Considering (3), nominal state-space function of robot is represented as

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathbf{f}\left(\mathbf{z}_k, \mathbf{v}_k\right) + C' \\ \mathbf{y}_k &= \mathbf{h}\left(\mathbf{z}_k\right), \end{aligned} \tag{4}$$

where $\mathbf{\Delta}_1$ discussed later keeps invariant in this department and is represented as C'. Objective function reflecting shortest distance can be used:

$$\min \quad J = \sum_k d\left(\mathbf{z}_k\right), \tag{5}$$

where $d(\bullet)$ is distance operator.

RFNN structure used here contains five layers and is shown in Figure 3. In order to achieve obstacle avoidance and reach target, output of system is chosen as input of RFNN, i.e., $\mathbf{s}' = \mathbf{y} \in \mathbb{R}^{n_3}$, while $\mathbf{v} = \mathbf{s}'' \in \mathbb{R}^{n_1}$ is generated control command to local plant. Subscript k is omitted here for simplification. Each node in membership layer performs a membership function. Membership functions can be defined with Gaussian MF as

$$s^1_{i_1 j_1} = \exp\left[-\frac{\left(s'_{i_1} - c_{i_1 j_1}\right)^2}{\sigma^2_{i_1 j_1}}\right], \tag{6}$$

where $\exp[\bullet]$ is exponential function. $\mathbf{c}$, $\boldsymbol{\sigma}$ are the mean and standard deviation of Gaussian function, respectively. The values of them are initially set via expert experience before the strategy begins.

As shown in Figure 3, each input parameter has two membership values, so there are 2^{n_3} fuzzy rules in fuzzy rule layer. The firing strength of each rule at current step is determined by outputs of membership layer through an AND operator. The result of each rule is calculated as

$$s^2_{i_2} = \prod_{j_1} s^1_{i_1 j_1}. \tag{7}$$

Moreover, a local internal feedback with real-time delay is added to each node of this layer to improve the robustness of the control system. The mathematical form is described by

$$s^3_{i_2}(k) = (1-\gamma)\, s^2_{i_2} + \gamma s^3_{i_2}(k-1), \tag{8}$$

where γ represents the weight of self-feedback loop and is a constant; $\mathbf{s}^3(k)$ is current output of this layer while $\mathbf{s}^3(k-1)$ represents the last step output.

Then, according to the definition of Takagi-Sugeno-Kang (TSK) fuzzy rules, the functions are combined linearly in consequent layer. TSK weight is defined as

$$w_{i_3 i_4} = a^1_{i_3 i_4} z_1 + a^2_{i_3 i_4} z_2 + \cdots + a^{n_3}_{i_3 i_4} z_{n_3}, \tag{9}$$

where $\mathbf{w} \in \mathbb{R}^{2^{n_3} \times n_1}$, $\mathbf{a}^1, \ldots, \mathbf{a}^{n_3} \in \mathbb{R}^{2^{n_3} \times n_1}$. For the purpose of simplifying calculation, it is assumed that $\mathbf{a} = \mathbf{a}^1 = \cdots = \mathbf{a}^n$. The output of this layer can be computed as follows:

$$s^4_{i_4} = \sum_{i_3} w_{i_3 i_4} s^3_{i_2}. \tag{10}$$

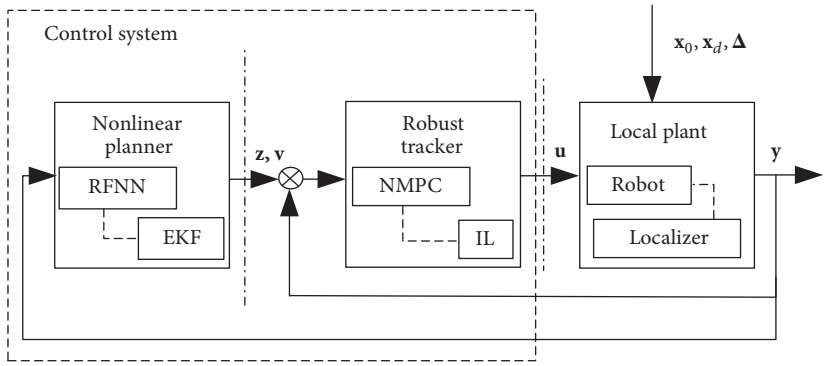


Figure 2: Block diagram of control system.

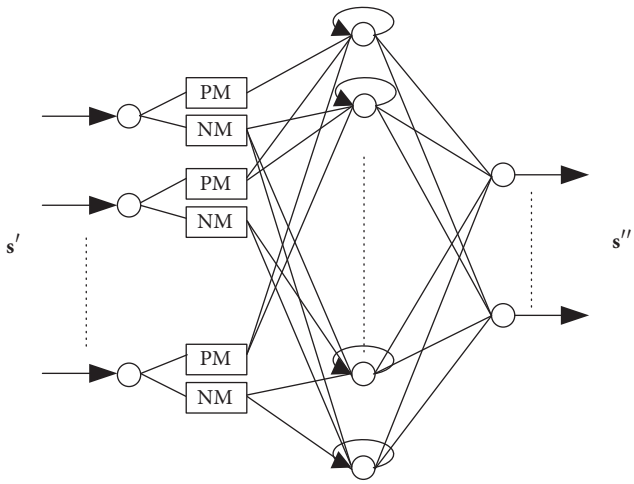


Figure 3: Structure of RFNN.

Activation function is set at each node of output layer to limit output forming output constraint:

$$s''_{i_4} = \frac{s^4_{i_4}}{1 + \eta_{i_4} \left| s^4_{i_4} \right|}, \tag{11}$$

where $\boldsymbol{\eta}$ is constant and its value depends on practical application.

In order to achieve motion program, extended Kalman filter method proposed in [28] is modified to train the weights of RFNN structure online. At step k, the EKF function is expressed as follows:

$$\mathbf{a}_{k+1} = \mathbf{a}_k - \mathbf{K}^{EKF}_k \mathbf{e}^{EKF}_k, \tag{12}$$

$$\mathbf{e}^{EKF}_k = \mathbf{o}_k - \mathbf{o}_{dk}, \tag{13}$$

$$\mathbf{K}^{EKF}_k = \mathbf{P}_k \mathbf{O}_k \left[\mathbf{R}^{EKF}_k + \mathbf{O}^T_k \mathbf{P}_k \mathbf{O}_k \right]^{-1}, \tag{14}$$

$$\mathbf{P}_{k+1} = \mathbf{Q}^{EKF}_k + \left[\mathbf{I} - \mathbf{K}^{EKF}_k \mathbf{O}^T_k \right] \mathbf{P}_k, \tag{15}$$

where $\mathbf{K}^{EKF}$ is Kalman gain matrix. $\mathbf{e}^{EKF}$ is estimation error. $\mathbf{o}_d$ is desired value of $\mathbf{o}$ which is observation vector. $\mathbf{R}^{EKF}$ is covariance matrix of measurement error. $\mathbf{Q}^{EKF}_k$ is constant matrix and is set according to [28], $\mathbf{P}$ is covariance matrix of estimation error, and $\mathbf{I}$ is identity matrix. $\mathbf{O}$ is orderly derivative matrix of observation vector with respect to RFNN weights.

Constraints of obstacles and target during navigation process are achieved by establishing observation vector in (13); Jacobian matrix is then calculated to predict weights of RFNN:

$$\mathbf{O} = \frac{\partial \mathbf{o}^T}{\partial \mathbf{a}}. \tag{16}$$

Target constraint is always prior. Only when obstacles threaten safety of robot, is obstacle constraint considered in (13). As initial weights are generated in random, by repeating calculation suboptimal trajectory can be got satisfying (4). Calculation time is t depending on repetition. To deal with uncertainty caused by dynamic environment, moving horizon is used forming online dynamic program. $Tra_1, \ldots, Tra_n$ are trajectories generated in different horizon. t_Δ is time between contiguous horizons. Time constraint should be satisfied:

$$t < t_\Delta, \tag{17}$$

where t_Δ depends on uncertainty. A new variable is set to judge disturbance using quadratic form as

$$\Delta = \left(\mathbf{o}^p_k - \mathbf{o}^r_k \right)^T \mathbf{Q}_\Delta \left(\mathbf{o}^p_k - \mathbf{o}^r_k \right), \tag{18}$$

where $\mathbf{o}^p_k$ and $\mathbf{o}^r_k$ are observation vectors at each step of program trajectory under system (4) and real trajectory under system (3), respectively. $\mathbf{Q}_\Delta$ is matrix of weights. Upper bound is set to get

$$\Delta < \Delta_{bound}, \tag{19}$$

According to (4)~(19), global motion program is obtained with bounded $\boldsymbol{\Delta}_2$. However, during practical process there exists $\boldsymbol{\Delta}_1$ affecting performance of navigation. So, suitable control and online learning method are needed in tracking procedure to guarantee robustness of system and ensure effective estimation of disturbances.

3.2. Learning Based Robust Tracking

3.2.1. Robust Nonlinear Model Predictive Control. Tube-based robust nonlinear model predictive control makes closed-loop trajectory, in the presence of disturbances, lie in a tube whose center is nominal trajectory. In this paper, nominal trajectory is generated by RFNN motion planner and then a tube-based approach is adopted to tracking system considering uncertainty caused by model and location error.

Considering (3) and (4), robot system to be controlled is the following nonlinear state-space form with bounded disturbances and $\mathbf{\Delta}_{2k}$ is not included:

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{\Delta}_{1k}, \tag{20}$$

where $\mathbf{f}(\bullet)$ is twice continuously differentiable.

If cost function of MPC is Lipschitz continuous; there exist two invariant sublevel sets of $\mathbb{X}$ such that the smaller set is robustly asymptotically stable for controlled system with a region for attracting the larger set [29]. Planned state and control input of navigation system in Section 3.1 are chosen as nominal state and control represented as $\mathbb{Z}$, $\mathbb{V}$ respectively. The controller aims at steering robot with disturbances close to index prediction and keeping actual trajectory of robot in a tube with good robustness. However, nominal state and control provided above are finite and the whole trajectory keeps changing as introduced in (17) ~ (19). To obtain effective and tightened reference sets, the following procedure is used. An object is defined as

$$\mathbb{T} = \left\{\left(Tra_1, t_{\Delta 1}\right), \left(Tra_2, t_{\Delta 2}\right), \ldots, \left(Tra_l, t_{\Delta l}\right), \ldots\right\}, \tag{21}$$

representing trajectories and update interval in different horizon. Only the first t_Δ step of Tra is valid motion. And interpolation is applied to get tightened $\mathbb{Z}$ and $\mathbb{V}$.

Considering deviation of state and control between actual system and nominal system, cost function to be minimized is defined as

$$\begin{aligned} J\left(\mathbf{x}, \mathbf{u}, k, l\right) &= \left(\mathbf{x} - \mathbf{z}_l\right)^T \mathbf{Q}\left(\mathbf{x} - \mathbf{z}_l\right) \\ &\quad + \left(\mathbf{u} - \mathbf{v}_l\right)^T \mathbf{R}\left(\mathbf{u} - \mathbf{v}_l\right) + J_f\left(\mathbf{x}; \mathbf{z}_f, l\right), \end{aligned} \tag{22}$$

where $\mathbf{Q} \in \mathbb{R}^{n_2 \times n_2}$ and $\mathbf{R} \in \mathbb{R}^{n_1 \times n_1}$ are positive definite. $\mathbf{z}_l \in \mathbb{Z}$ and $\mathbf{v}_l \in \mathbb{V}$ are reference trajectory and control action, respectively. $J_f(\bullet)$ is terminal cost function and $\mathbf{z}_f$ is terminal state that is changing due to (21).

Optimal control problem can be illustrated as

$$\min_{\boldsymbol{u}} \quad J\left(\mathbf{x}, \mathbf{u}, k, l\right). \tag{23}$$

The solution of (23) is $\mathbf{u}^* = \{\mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_{N-1}^*\}$ and associated state trajectory is $\mathbf{x}^* = \{\mathbf{x}_0^*, \mathbf{x}_1^*, \ldots, \mathbf{x}_{N-1}^*\}$. $\kappa = \mathbf{u}_0^*$ is applied to uncertainty system, i.e., the first element in the sequence. If $\mathbf{x} = \mathbf{z}_l$, then $\mathbf{u} = \mathbf{v}_l$, so that $J(\mathbf{x}, \mathbf{u}, k, l) = 0$. Minimum value of cost function at each step of trajectory is represented as $J^{\min}(\mathbf{x}, k, l)$. Level set of cost function defines a neighborhood of nominal trajectory. A tube can be defined [29]:

$$S_d = \left\{\mathbf{x} \mid J^{\min}\left(\mathbf{x}, k, l\right) \le d\right\}. \tag{24}$$

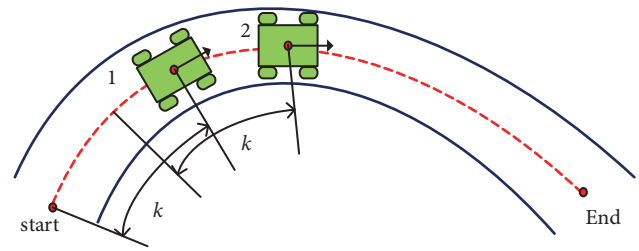


Figure 4: Dynamic iterative route.

According to [30], stable condition exists:

$$\begin{aligned} \min_{\mathbf{u}} \Big\{ & J'_f\left(\mathbf{f}\left(\mathbf{x},\mathbf{u}\right); \mathbf{z}_f, l\right) + \left(\mathbf{x} - \mathbf{z}_f\right)^T \mathbf{Q}\left(\mathbf{x} - \mathbf{z}_f\right) \\ & + \left(\mathbf{u} - \mathbf{v}_f\right)^T \mathbf{R}\left(\mathbf{u} - \mathbf{v}_f\right)\Big\} \le J'_f\left(\mathbf{f}\left(\mathbf{x}, \mathbf{u}\right); \mathbf{z}_f, l\right), \end{aligned} \tag{25}$$

where $J'_f(\mathbf{f}(\mathbf{x}, \mathbf{u}); \mathbf{z}_f, l) = (\mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{z}_f)^T \mathbf{Q}_f (\mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{z}_f)$ and $\mathbf{x} \in \{\mathbf{x} \mid J'_f(\mathbf{x}; \mathbf{z}_f, l) \le \alpha\}$. $\mathbf{Q}_f$ is positive definite. Then $J_f(\bullet)$ is determined as $J_f(\mathbf{x}; \mathbf{z}_f) = dJ'_f(\mathbf{x}; \mathbf{z}_f)/\alpha$. Method in [31] is used to solve the optimization problem.

3.2.2. Iterative Online Learning. In actual situation there are disturbances caused by inaccurate model established a priori and invariant perception error. So, online learning method should be considered to estimate intrinsic part of disturbances online and make the control system effective. As $\mathbf{f}(\bullet)$ is nonlinear in (20), disturbance $\mathbf{\Delta}_1$ is no more Gaussian leading to the fact that GP based estimation method is not suitable here. So, a kind of PD iterative online learning method is designed in this section to estimate disturbances. The process is assumed to occur in an unknown environment and robot moves to goal avoiding obstacles in a predictive routine instead of a known iterative path. So, compared to traditional iterative method dynamic iterative route is used in our iterative learning algorithm. In order to learn compensation with dynamic route, suitable and effective iterative function should be designed as follows.

Kinematic model of mobile robot is used here to introduce the procedure and detailed information is introduced in Section 5.1. Then state-space system in (20) can be illustrated as

$$f\left(\mathbf{x}, \mathbf{u}, \mathbf{w}\right) = \mathbf{x} + \mathbf{A}\left(\mathbf{x}\right)\mathbf{u} + \mathbf{\Delta}_1, \tag{26}$$

where $\mathbf{x}$ is state of system, $\mathbf{u}$ is control input, and $\mathbf{\Delta}_1$ is disturbance need to be learned.

Each iteration has k equations, but start and end points are different. The demonstration of choosing iteration route is illustrated intuitively in Figure 4. There are two trials end at points 1 and 2, respectively, during the process; start points can be got due to same steps k.

In order to use this form of iterative route, a new variable is defined: $\boldsymbol{\sigma} = \mathbf{z} - \mathbf{x}$. According to (26), k equations can be got:

$$\begin{aligned}
\boldsymbol{\sigma}_2 &= \mathbf{z}_2 - \mathbf{x}_2 = \mathbf{z}_2 - \mathbf{x}_1 - \mathbf{A}_1\mathbf{u}_1 - \boldsymbol{\Delta}_1 \\
&= \mathbf{z}_1 - \mathbf{x}_1 + \overline{\mathbf{A}}_1\mathbf{v}_1 - \mathbf{A}_1\mathbf{u}_1 - \boldsymbol{\Delta}_1 \\
&= \boldsymbol{\sigma}_1 + \overline{\mathbf{A}}_1\mathbf{v}_1 - \mathbf{A}_1\mathbf{u}_1 - \boldsymbol{\Delta}_1 \\
\boldsymbol{\sigma}_3 &= \mathbf{z}_3 - \mathbf{x}_3 = \mathbf{z}_2 - \mathbf{x}_2 + \overline{\mathbf{A}}_2\mathbf{v}_2 - \mathbf{A}_2\mathbf{u}_2 - \boldsymbol{\Delta}_2 \\
&= \boldsymbol{\sigma}_2 + \overline{\mathbf{A}}_2\mathbf{v}_2 - \mathbf{A}_2\mathbf{u}_2 - \boldsymbol{\Delta}_2 \\
&= \boldsymbol{\sigma}_1 + \overline{\mathbf{A}}_1\mathbf{v}_1 - \mathbf{A}_1\mathbf{u}_1 + \overline{\mathbf{A}}_2\mathbf{v}_2 - \mathbf{A}_2\mathbf{u}_2 - \boldsymbol{\Delta}_1 - \boldsymbol{\Delta}_2 \\
&\vdots,
\end{aligned} \tag{27}$$

where $\mathbf{z}$, $\mathbf{v}$, and $\overline{\mathbf{A}}$ are variables of references produced by RFNN in Section 3.1. In order to simplify representation, $\boldsymbol{\Delta}_{1k}$ is replaced by $\boldsymbol{\Delta}_k$ above. Then the lifted form can be organized as

$$\boldsymbol{\sigma}_K = \boldsymbol{\sigma}_0 - \mathbf{B}\boldsymbol{\Delta}_K, \tag{28}$$

where K represents iteration and $\mathbf{B}$ is lower triangular matrix. Component of each item is shown as

$$\begin{aligned}
\boldsymbol{\sigma}_K &= \begin{bmatrix} \boldsymbol{\sigma}_2 \\ \boldsymbol{\sigma}_3 \\ \vdots \\ \boldsymbol{\sigma}_{k+1} \end{bmatrix}, \\
\boldsymbol{\sigma}_0 &= \begin{bmatrix} \boldsymbol{\sigma}_1 + \overline{\mathbf{A}}_1\mathbf{v}_1 - \mathbf{A}_1\mathbf{u}_1 \\ \boldsymbol{\sigma}_1 + \overline{\mathbf{A}}_1\mathbf{v}_1 - \mathbf{A}_1\mathbf{u}_1 + \overline{\mathbf{A}}_2\mathbf{v}_2 - \mathbf{A}_2\mathbf{u}_2 \\ \vdots \\ \boldsymbol{\sigma}_1 + \cdots + \overline{\mathbf{A}}_k\mathbf{v}_k - \mathbf{A}_k\mathbf{u}_k \end{bmatrix}, \\
\boldsymbol{\Delta}_K &= \begin{bmatrix} \boldsymbol{\Delta}_1 \\ \boldsymbol{\Delta}_2 \\ \vdots \\ \boldsymbol{\Delta}_k \end{bmatrix}.
\end{aligned} \tag{29}$$

Although the system illustrated by (26) is nonlinear, it is Lipschitz continuous. According to [32], linear method can deal with this kind of problem well. Compared to nominal discrete-time system introduced in [33], the initial conditions in (28) are different at each iteration. Many methods can be used to settle this problem such as initial state learning mechanism or smooth interpolation. As shown above, $\boldsymbol{\sigma}_0$ is made up of discrepancy at all steps in each trajectory. Suppose the first two iterations are convergent, then $\boldsymbol{\sigma}_0$ decreases in later iteration and is upper bounded. So, a simple forgetting factor λ_f is used to discount integration effect of iterative learning. Then a kind of PD-type iterative learner is used:

$$\boldsymbol{\Delta}_{K+1} = \lambda_f\boldsymbol{\Delta}_K + \boldsymbol{\lambda}_p\mathbf{R}^{IL}\mathbf{e}_K + \boldsymbol{\lambda}_d\left(\mathbf{I} - \mathbf{R}^{IL}\right)\mathbf{e}_K, \tag{30}$$

where $\boldsymbol{\lambda}_p$ and $\boldsymbol{\lambda}_d$ are proportional and derivative learning gains, respectively. $\mathbf{R}^{IL}$ is lower triangular Toeplitz matrix derived from column vector $[\mathbf{0}, \mathbf{I}, \mathbf{0}, \ldots, \mathbf{0}]^T$ where 0 is matrix with 0 elements and $\mathbf{I}$ is $n_2 \times n_2$ unit matrix.

For iteration K, error can be defined as

$$\mathbf{e}_K = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}_K, \tag{31}$$

where $\boldsymbol{\sigma}_d$ represents desired error between actual state and optimal predictive state and it is usually 0. Substituting (31) and (28) into (30) yields closed-loop iteration dynamics:

$$\begin{aligned}
\boldsymbol{\Delta}_{K+1} = &\left(\mathbf{I} + \left(\boldsymbol{\lambda}_p\mathbf{R}^{IL} + \boldsymbol{\lambda}_d\left(\mathbf{I} - \mathbf{R}^{IL}\right)\right)\mathbf{B}\right)\boldsymbol{\Delta}_K \\
&+ \left(\boldsymbol{\lambda}_p\mathbf{R}^{IL} + \boldsymbol{\lambda}_d\left(\mathbf{I} - \mathbf{R}^{IL}\right)\right)(\boldsymbol{\sigma}_d - \boldsymbol{\sigma}_0).
\end{aligned} \tag{32}$$

According to [33], iterative learning process is asymptotically stable, if

$$\rho\left(\mathbf{I} + \left(\boldsymbol{\lambda}_p\mathbf{R}^{IL} + \boldsymbol{\lambda}_d\left(\mathbf{I} - \mathbf{R}^{IL}\right)\right)\mathbf{B}\right) < 1, \tag{33}$$

where $\rho(\bullet)$ is spectral radius operator.

4. Stability Analysis

According to Figure 2, the control system is made up of two parts. So, in order to analyze stability of the system, each part's stability should be guaranteed.

4.1. Convergence of Recurrent Fuzzy Neural Network. In this paper, a training method based on EKF is used as shown in Section 3.1. Appropriate parameters should be designed. Then, by iteratively calculating (12) ~ (15) trajectories are generated. By evaluating these trajectories, suboptimal trajectory is chosen.

Observation vector, $\mathbf{o}_k$, is first expanded at final stable weights $\mathbf{a}_d$ as

$$\mathbf{o}_k = \mathcal{O}(\mathbf{a}_d) + (\mathbf{a}_k - \mathbf{a}_d)\frac{\partial\mathbf{o}}{\partial\mathbf{a}} + \boldsymbol{\xi}_k, \tag{34}$$

where $\boldsymbol{\xi}_k$ is first-order approximation residue and $\mathcal{O}(\bullet)$ is mapping function from weights to observation. Error of weights can be defined: $\mathbf{e}_{ak} = \mathbf{a}_k - \mathbf{a}_d$.

Then, Lyapunov function can be chosen as

$$E_k = \mathbf{e}_{ak}^T\mathbf{P}_k^T\mathbf{e}_{ak}. \tag{35}$$

The difference can be calculated as

$$\begin{aligned}
\Delta E(k) &= E(k+1) - E(k) \\
&= \mathbf{e}_{a(k+1)}^T\mathbf{P}_{k+1}^{-1}\mathbf{e}_{a(k+1)} - \mathbf{e}_{ak}^T\mathbf{P}_k^{-1}\mathbf{e}_{ak} \\
&< \left[\mathbf{e}_{a(k+1)} - \mathbf{e}_{ak}\right]^T\mathbf{P}_k^{-1}\mathbf{e}_{ak} \\
&\quad - \mathbf{e}_{a(k+1)}^T\left[\mathbf{P}_{k+1} - \mathbf{Q}_k^{EKF}\right]^{-1}\mathbf{P}_k\mathbf{O}_k^{EKF}\boldsymbol{H}_k^{-1}\boldsymbol{\xi}_k,
\end{aligned} \tag{36}$$

where $\boldsymbol{H}_k = \mathbf{R}_k^{EKF} + \mathbf{O}_k^T\mathbf{P}_k\mathbf{O}_k$. Then following inequation can be got according to [28]

$$\Delta E_k < \frac{3\left\|\boldsymbol{\xi}_k\right\|^2}{r_k} - \frac{\left\|\boldsymbol{e}_k\right\|^2}{(o_k/m + r_k)}, \tag{37}$$

where o_k is the trace of $\mathbf{O}_k^T\mathbf{P}_k\mathbf{O}_k$, $\mathbf{R}_k = r_k\mathbf{I}$. $\mathbf{I}$ is $m \times m$ identity matrix and r_k is a positive real number. For details of inference formula refer to [28].

As mentioned above, $\mathbf{O}$ and $\mathbf{R}$ determine convergence of EKF training process. $\mathbf{O}$ is derived from observation vector $\mathbf{o}$. In order to guarantee effectiveness of recurrent fuzzy neural network structure, $\mathbf{o}$ should be constructed reasonable. According to (6) ~ (16),

$$\mathbf{O} = \frac{\partial \mathbf{o}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{s}''} \frac{\partial \mathbf{s}''}{\partial \mathbf{a}}, \tag{38}$$

where $\partial \mathbf{z}/\partial \mathbf{s}''$ illustrates plant information, $\partial \mathbf{s}''/\partial \mathbf{a}$ illustrates RFNN structure information, and $\partial \mathbf{o}/\partial \mathbf{z}$ illustrates relation between observation and plant's state. Jacobian matrix, $\mathbf{O}$, reflects performance of chosen weights to system. Then $\mathbf{R}$ is designed considering $\mathbf{O}$ to realize convergence of strategy.

In (37), system is convergent if $\Delta E_k < 0$ and then r_k should be set as

$$r_k > \frac{3o_k \left\|\boldsymbol{\xi}_k\right\|^2}{m\left(\left\|\mathbf{e}_k\right\|^2 - 3\left\|\boldsymbol{\xi}_k\right\|^2\right)}, \tag{39}$$

where each element of $\boldsymbol{\xi}_k$ is normal distribution as $\xi_{ik} \sim N(0, r_k)$; then

$$\frac{3o_k}{m} < r_k < \frac{\left\|\mathbf{e}_k\right\|^2}{64m}, \tag{40}$$

with 99.99% $\boldsymbol{\xi}_k$ being bounded.

And $r(k)$ adaption law is got:

$$r_k = \frac{1}{2}\left(\frac{\left\|\mathbf{e}_k\right\|^2}{64m} + \frac{3o_k}{m}\right), \tag{41}$$

which can make control system's error convergent, i.e., bounded $\mathbf{e}_k$.

As mentioned in Section 3.1, initial weights are generated in random leading to predictive trajectories being different with different initial weights. Initial weights are represented as

$$\left\{\mathbf{a}_0^1 \;\; \mathbf{a}_0^2 \;\; \cdots \;\; \mathbf{a}_0^l \;\; \cdots\right\}, \tag{42}$$

where each element corresponds to an independent trajectory produced by EKF training method. Some method can be chosen to learn initial weights, but a simple method of iteration is used to economize computation cost and guarantee real-time program ability. Considering practical condition, evaluation of optimal trajectory is different especially in dynamic navigation problem. With evaluation criteria, $\mathbf{a}_0$ is chosen to obtain suboptimal trajectory.

4.2. Stability and Robustness of Nonlinear Model Predictive Control. By using RFNN based planner, a set of trajectories are obtained in (21) to approximate optimal path in dynamic environment. MPC method is then used to limit states of robot in a tube around the optimal path under disturbance $\boldsymbol{\Delta}_1$.

For each trajectory in $\mathbb{T}$, suppose there is $d' > d$ and bounded area around trajectory is $S_{d'} = \{\mathbf{x} \mid J^{\min}(\mathbf{x}, k, l) \leq d'\}$. Let $\mathbf{x}^* = \mathbf{z}$ denotes generated state of nominal system; i.e., $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{C}$, with input $\mathbf{u}^* = \mathbf{v}$. Since $\mathbf{f}(\bullet)$ is Lipschitz continuous, $\|\mathbf{x}_k^* - \mathbf{z}_{lk}\| \leq L\|\mathbf{x} - \mathbf{z}_{l0}\|$ for all $\mathbf{x} \in S_{d'} + \boldsymbol{\Delta}_1$. According to (22) and (25), there exists d_1 such that

$$J^{\min}(\mathbf{x}, k, l) \leq J\left(\mathbf{x}, \mathbf{u}^*, k, l\right) \leq d_1 \left\|\mathbf{x} - \mathbf{z}_{l0}\right\|^2. \tag{43}$$

Stability condition is established in (25); then

$$\begin{aligned} J^{\min}(\mathbf{x}, k+1, l) \leq\; & J^{\min}(\mathbf{x}, k, l) \\ & - \left(\mathbf{x} - \mathbf{z}_{l0}\right)^T \mathbf{Q}\left(\mathbf{x} - \mathbf{z}_{l0}\right) \\ & - \left(\kappa_k - \mathbf{v}_{l0}\right)^T \mathbf{R}\left(\kappa_k - \mathbf{v}_{l0}\right). \end{aligned} \tag{44}$$

There is also lower bound due to weight matrix $\mathbf{Q}$, so

$$J^{\min}(\mathbf{x}, k, l) \geq d_2 \left\|\mathbf{x} - \mathbf{z}_{l0}\right\|^2. \tag{45}$$

Combine (43) ~ (45), and

$$J^{\min}(\mathbf{x}, k+1, l) \leq \left(1 - \frac{d_2}{d_1}\right) J^{\min}(\mathbf{x}, k, l). \tag{46}$$

In practice, the surroundings where robot needs to achieve navigation can be entirely or partially detected by sensors. Suppose the change of surroundings is continuous and not too fast relative to robot's motion. Then disturbance $\boldsymbol{\Delta}_1$ is bounded. Let $\overline{\Delta} = \max\{\|\boldsymbol{\Delta}\| \mid \boldsymbol{\Delta} \in \boldsymbol{\Delta}_1\}$. According to [29], for all $\mathbf{x} \in S_{d'}$ and $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\Delta}_1$, there is a constant d_3 such that

$$\begin{aligned} J^{\min}(\mathbf{x}, k+1, l) &\leq \left(1 - \frac{d_2}{d_1}\right) J^{\min}(\mathbf{x}, k, l) + d_3 \left\|\boldsymbol{\Delta}_1\right\| \\ &\leq \left(1 - \frac{d_2}{d_1}\right) J^{\min}(\mathbf{x}, k, l) + d_3\overline{\Delta}. \end{aligned} \tag{47}$$

If $(d_2/d_1)J^{\min}(\mathbf{x}, k, l) \geq d_3\overline{\Delta}$, (47) becomes

$$J^{\min}(\mathbf{x}, k+1, l) \leq J^{\min}(\mathbf{x}, k, l) \leq \frac{c_1 c_3 \overline{\Delta}}{c_2}. \tag{48}$$

So, there exists constant $d = c_1 c_3 \overline{\Delta}/c_2$ that limits actual path of robot in a tube of optimal trajectory generated by RFNN.

After analysis and design of the overall system, whole strategy of navigation is illustrated as Algorithm 1.

5. Experiments

5.1. Model of Mobile Robot. In order to approve effectiveness of our control system, a kind of differentially driven mobile robot system model in [9] is used here. The robot model in unknown environment with obstacles and target is illustrated as in Figure 5.

Initialization: $\mathbf{x}_0 = \mathbf{0}$, obtain initial position of target and obstacles, $\mathbf{x}_{target}, \mathbf{x}_{obstacle}$.
set $\mathbf{c}, \boldsymbol{\sigma}, \mathbf{Q}_\Delta, \gamma, \boldsymbol{\eta}, \Delta_{bound}, \mathbf{Q}, \mathbf{R}, (d/\alpha)\mathbf{Q}_f, \lambda_f, \boldsymbol{\lambda}_p, \boldsymbol{\lambda}_d$
while $\|\mathbf{x} - \mathbf{x}_{target}\| > \beta$ **do**
    Prediction: Predict robot motion according to (4) ~ (16) and (33) ~ (39).
Generate trajectory in $\mathbb{T}$ as well as referential states and control commands, i.e. $\mathbf{z}, \mathbf{v}$.
while $\Delta < \Delta_{bound}$ **do**
    Tracking: Generate actual control command, $\mathbf{u}$, according to (21) ~ (30).
    Sensor: Measure distance and angel information among target, obstacles and robot, i.e. $\mathbf{y}$.
    Location: Estimate state of robot, obstacle and target, i.e. $\mathbf{x}, \mathbf{x}_{target}, \mathbf{x}_{obstacle}$.
    Update $\Delta, \mathbf{x}, \mathbf{x}_{target}, \mathbf{x}_{obstacle}$.

Algorithm 1: Navigation strategy.

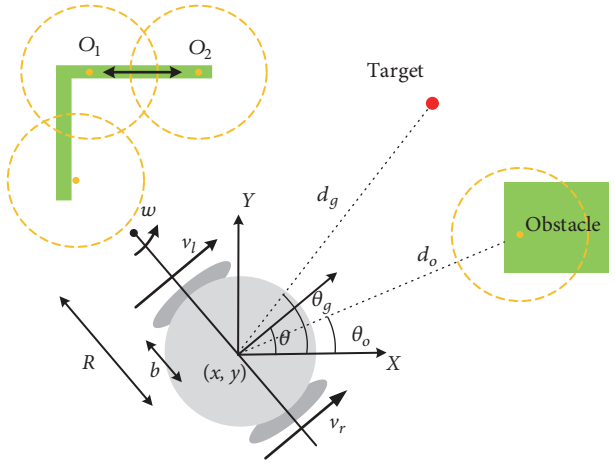


Figure 5: Model of mobile robot in unknown environment.

Robot is motivated by right and left wheels, $[v_r, v_l]^T$. Linear and angular velocities are

$$v = \frac{v_r + v_l}{2}, \tag{49}$$

$$\omega = \frac{v_r - v_l}{2b}, \tag{50}$$

$$R = b\frac{v_r + v_l}{v_r - v_l}, \tag{51}$$

Difference equation is then illustrated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_k, \tag{52}$$

where $\mathbf{u} = (v, \omega)^T$. $\mathbf{x}_k, \mathbf{x}_{k+1}$ are robot's postures at k and $k+1$ steps, respectively, and are represented as $\mathbf{x} = [x, y, \theta]^T$.

Distance and angle of target and nearest obstacle corresponding to robot are chosen as measurement, so output is represented as

$$\mathbf{y}_k = \left[d_{gk}, \theta_{gk}, d_{ok}, \theta_{ok}\right]^T. \tag{53}$$

Table 1: Initialization of parameters.

b	0.2m	Δ_{bound}	0.5
$\mathbf{c}$	$\begin{bmatrix} 0 & 0 & -\frac{\pi}{2} & -\frac{\pi}{2} \\ 10 & 5 & \frac{\pi}{2} & \frac{\pi}{2} \end{bmatrix}$	λ_p	0.1
$\boldsymbol{\sigma}$	$[2.5 \quad 1 \quad 1 \quad 1]$	λ_d	3
$\boldsymbol{\eta}$	$[0.3 \quad 0.3 \quad \cdots]$	λ_f	0.9
γ	0.3	β	0.5

As shown in Figure 5, a sparse representation of obstacles is used to train RFNN. Circles with center on obstacle represent dangerous area. Radius of circle depends on EKF learning method and estimation error. Location of obstacle is usually not accurate leading to error of centers' position shown in figure. Distance between O_1 and O_2 depends on b. Although RFNN structure is convergent, robot will vibrate at terminal position due to feedback control principle. When robot reaches area determined by β, a brake control is used.

5.2. Performance in Static Condition with Bounded Disturbance. Simulation experiment is carried out using Matlab. Activity space is limited in 10 m × 10 m. There exist obstacles and target. Mobile robot needs to reach target and avoid obstacles. Constant parameters to be initialized are listed in Table 1. The weight matrices are set with 25 : 15 : 1 ratio weighting position-tracking error and linear and angular velocity control input in (22). The weight $\mathbf{Q}_\Delta$ is switched between two sets due to position corresponding to target and obstacles.

Proposed method is first examined in static environment. As environment is static, only one prediction of optimal motion of robot is generated and the weights of RFNN are shown in Figure 6. There is no drastic variety underlying smoothness of process. Gaussian noise with mean zero and variance of 0.03 is injected to system named as disturbance 1. And 20 trajectories are generated under random disturbance. Performance is shown as in Figure 7. All trajectories are bounded in a band. Each trajectory of robot can realize reaching adjacent area of target and avoiding obstacles. Optimal and robust navigation is achieved. Position error curves of robot are shown in Figure 8(a), which are bounded in a narrow band. Figures 8(b) and 8(c) represent linear

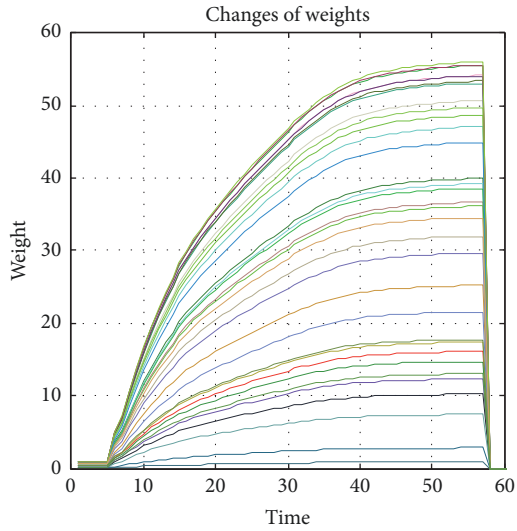


Figure 6: Weights of RFNN.

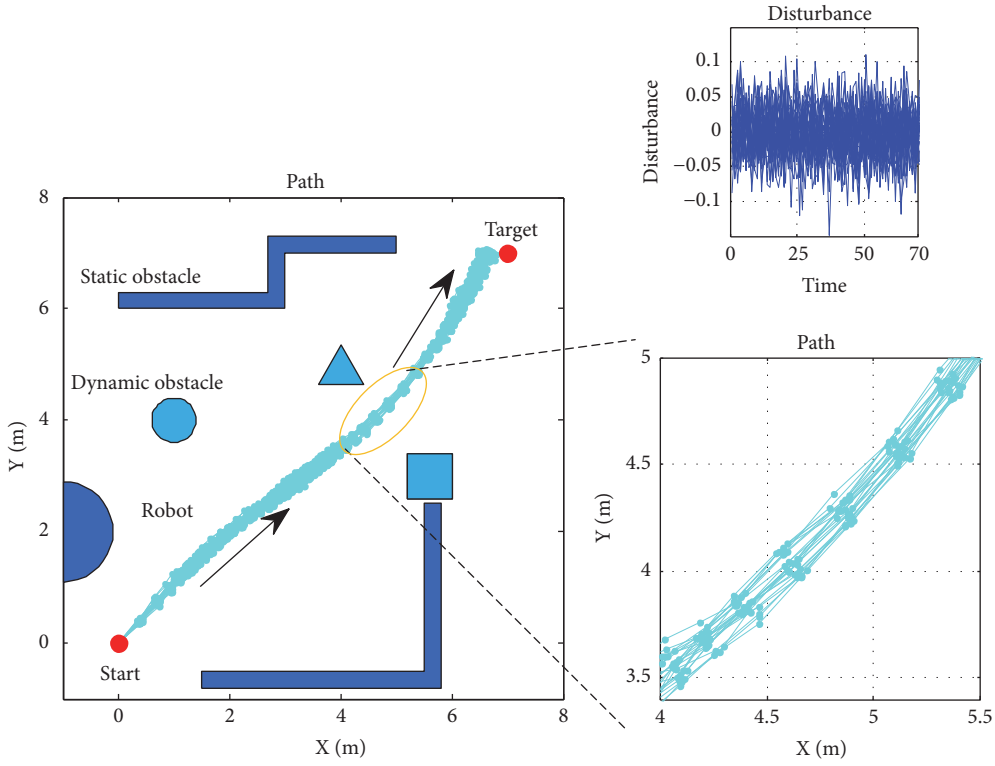
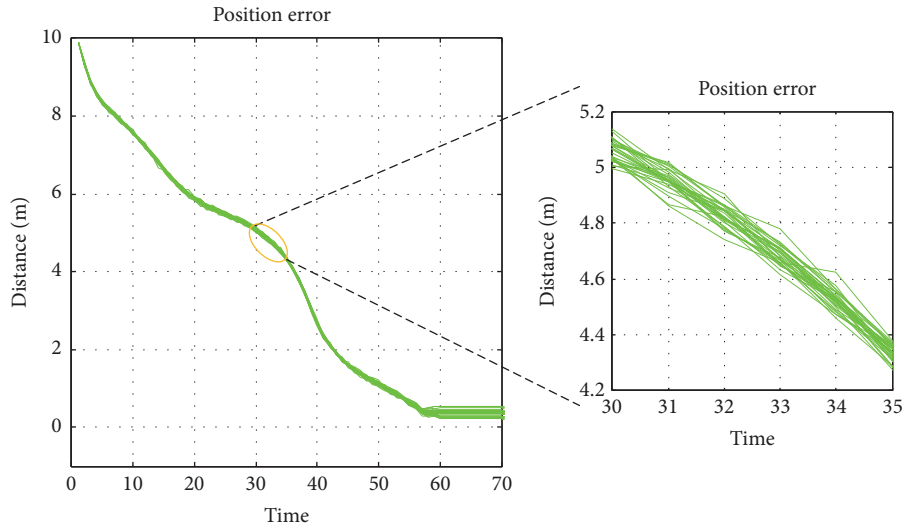


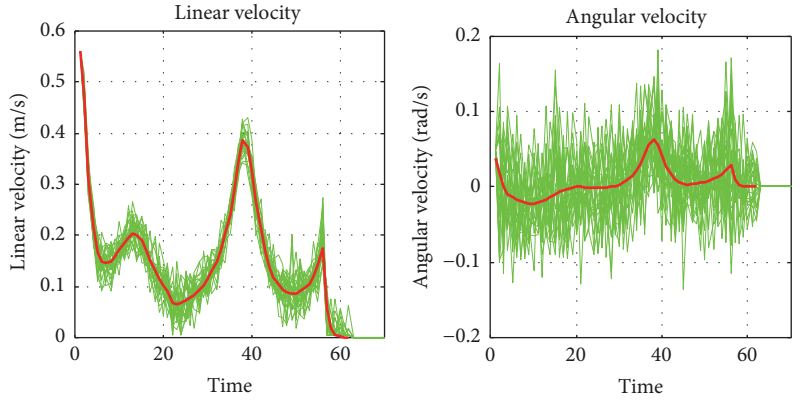
Figure 7: Performance in static environment: 20 trajectories with disturbance 1.

and angular velocities of robot, respectively. They are not narrow contrasted to position error for the reason that input commands should control robot against random disturbance and limit robot's position in a narrow tube. Computation cost of generating optimal control command at each step is shown in Figure 9. Lines in red are average values.

5.3. Performance of Online Iterative Learning Method. In order to examine online iterative learning method proposed, Gaussian noise with mean 0.1 and variance of 0.03 is injected to system named as disturbance 2. 20 trajectories are also generated under this random disturbance. Performance is shown as Figure 10. For neat expression only one trajectory without iterative learning in magenta is shown in figure. This trajectory is dangerous and does not lie in stable tube. Trajectories with iterative leaning are illustrated in cyan and are same as trajectories under disturbance 1 underlying estimation method which is effect. Estimation of disturbance



(a) Distance error of robot



(b) Linear velocity of robot

(c) Angular velocity of robot

Figure 8: Position error and control inputs of robot with 20 trajectories.

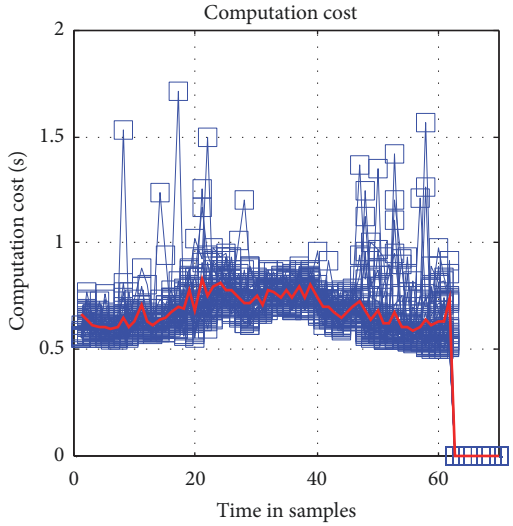


Figure 9: Computation cost at each step of 20 trajectories.

2 is shown in Figure 11. Detailed information is shown in Table 2. There is not much difference of values in each item between two conditions. The largest bias at each point of 20 trajectories under two conditions is shown in Figure 12. The largest bias of trajectories under disturbance 2 is a little higher.

5.4. Comparison with Other Methods. The main novelty of proposed method in this paper is that a motion planning and control method based navigation strategy is proposed to deal with uncertainty and guarantee robust optimization of the process. The dual-layer structure is designed to be feasible and effective by combining RFNN and MPC method. Many motion planning techniques are used to achieve navigation, like potential field method, sampling based planner, interpolation curve method, A∗, neural network, fuzzy logic, genetic algorithm, particle swarm optimization, etc. A∗ method is widely used in autonomous vehicles for its efficiency [34]. Furthermore, OPTI toolbox of Matlab includes many optimal solvers that can be used to design nonlinear planner. In order to introduce our method's effectiveness, A∗ method together with OPTI based numerical optimization method is compared with ours. As control policy is considered in our method, PID method in [35] is used for two planners

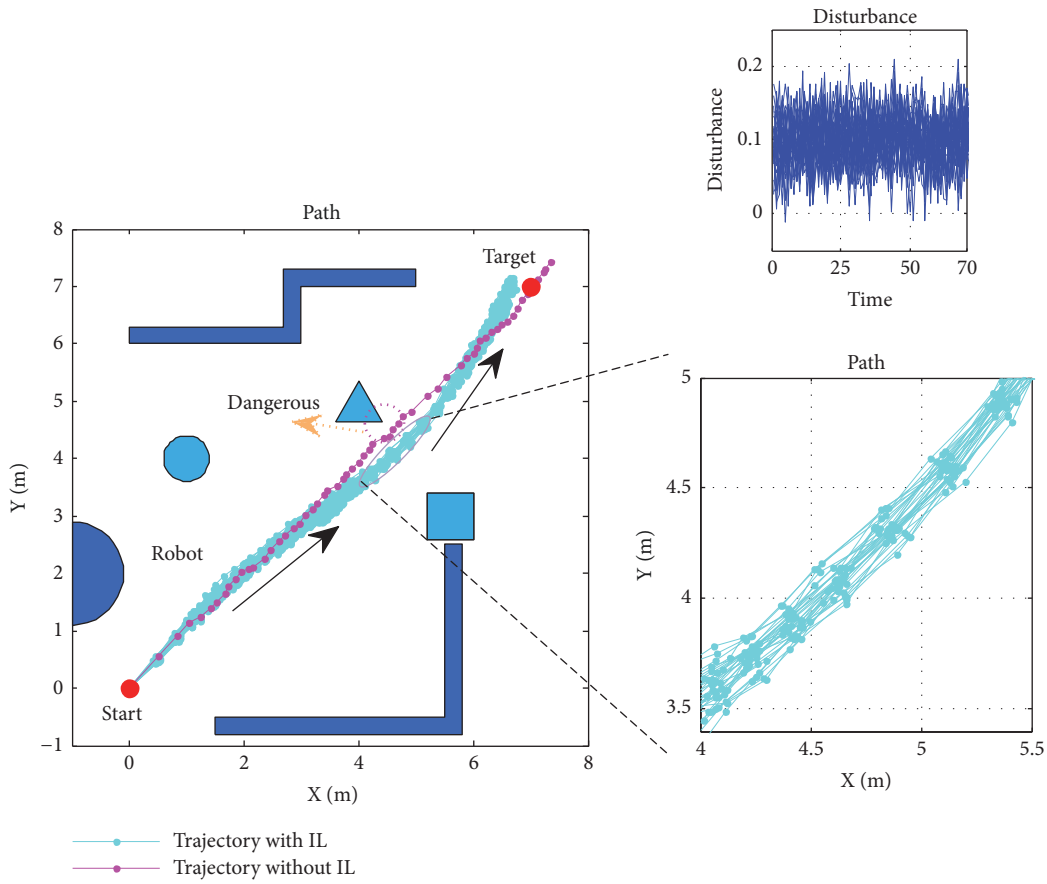


Figure 10: Performance in static environment: 20 trajectories with disturbance 1.

Table 2: Evaluation of trajectories in static environment.

	Prediction cost (s)	Step	Average distance (m)	Average terminal error (m)	Average bias (m)
Trajectories under disturbance 1	4.13	63	9.82	0.36	0.061
Trajectories under disturbance 2	4.13	63	9.86	0.35	0.067

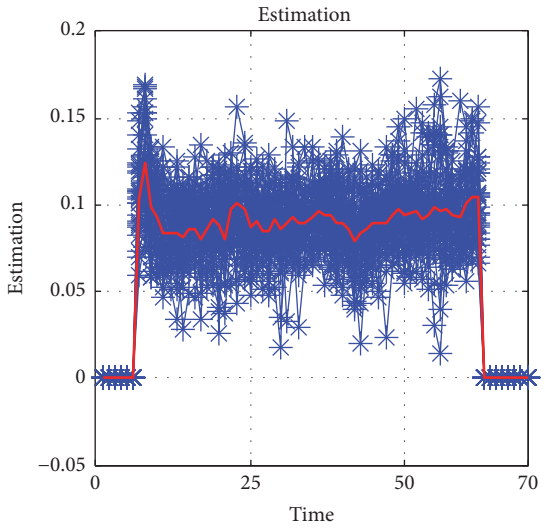


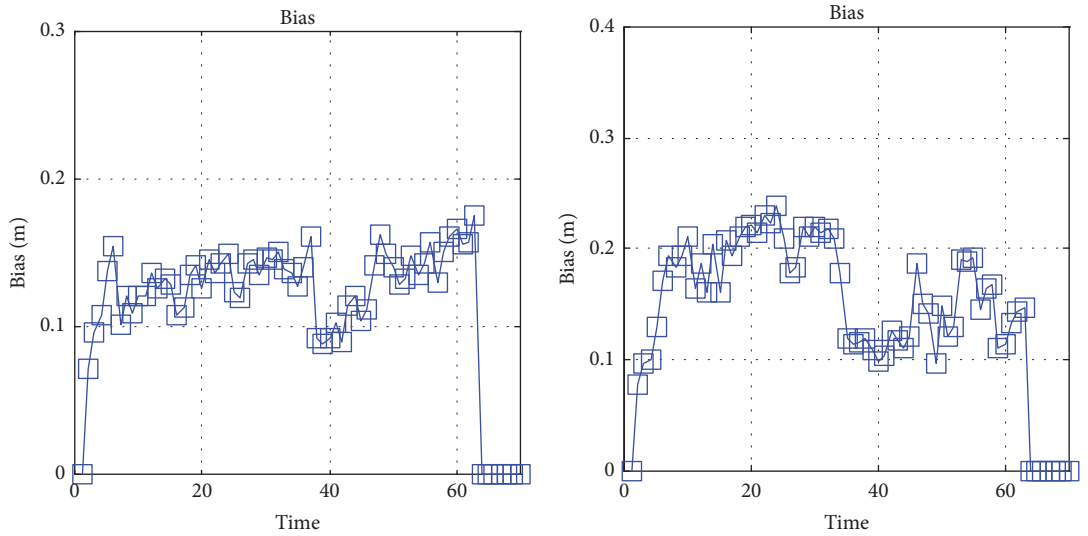
Figure 11: Estimation of disturbance 2.

above forming complete navigation system just like ours. Performance is shown in Figure 13. A∗ method just plans positions of robot, leading to bad tracking performance under disturbance which is reflected by tracking error in Table 3 and the path in red with rectangle marks. The trajectory is not smooth contrasted to our method and OPTI and PID method. PID control method that lacks online learning mechanism fails to drive robot to target when disturbance in Figure 10 acted on system, which is reflected by red path with triangle marks. Compared to A∗ and PID based strategy, OPTI and PID method that plans motion performs better. But OPTI based planner takes up too much computation cost reflected in Table 3 leading to bad real-time planning ability. Detailed information of these methods' performance is illustrated in Table 3. It is proved that our method can achieve good online navigation.

5.5. Performance in Dynamic Condition. Proposed method is then examined in dynamic environment. Performance is

Table 3: Comparison between our method and others.

	Distance (m)	Step	Computation cost (s)	Terminal error (m)	Tracking error (m)
OPTI & PID	10.132	60	37.891	0.289	0.048
A∗ & PID 1	10.389	62	0.512	0.402	0.059
A∗ & PID 2	6.209	37	0.512	3.889	0.231
OURS	10.090	62	3.142	0.251	0.040



(a) Maximal bias under disturbance 1 (b) Maximal bias under disturbance 2

Figure 12: Largest bias at each step under two conditions.

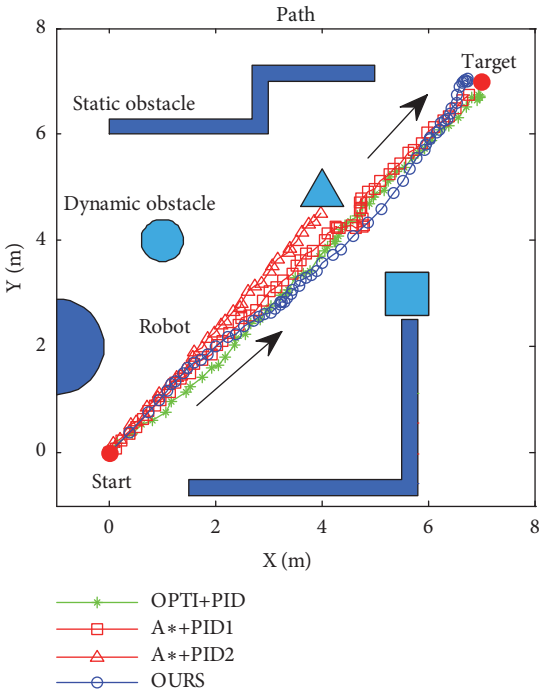


Figure 13: Comparison between our method and others.

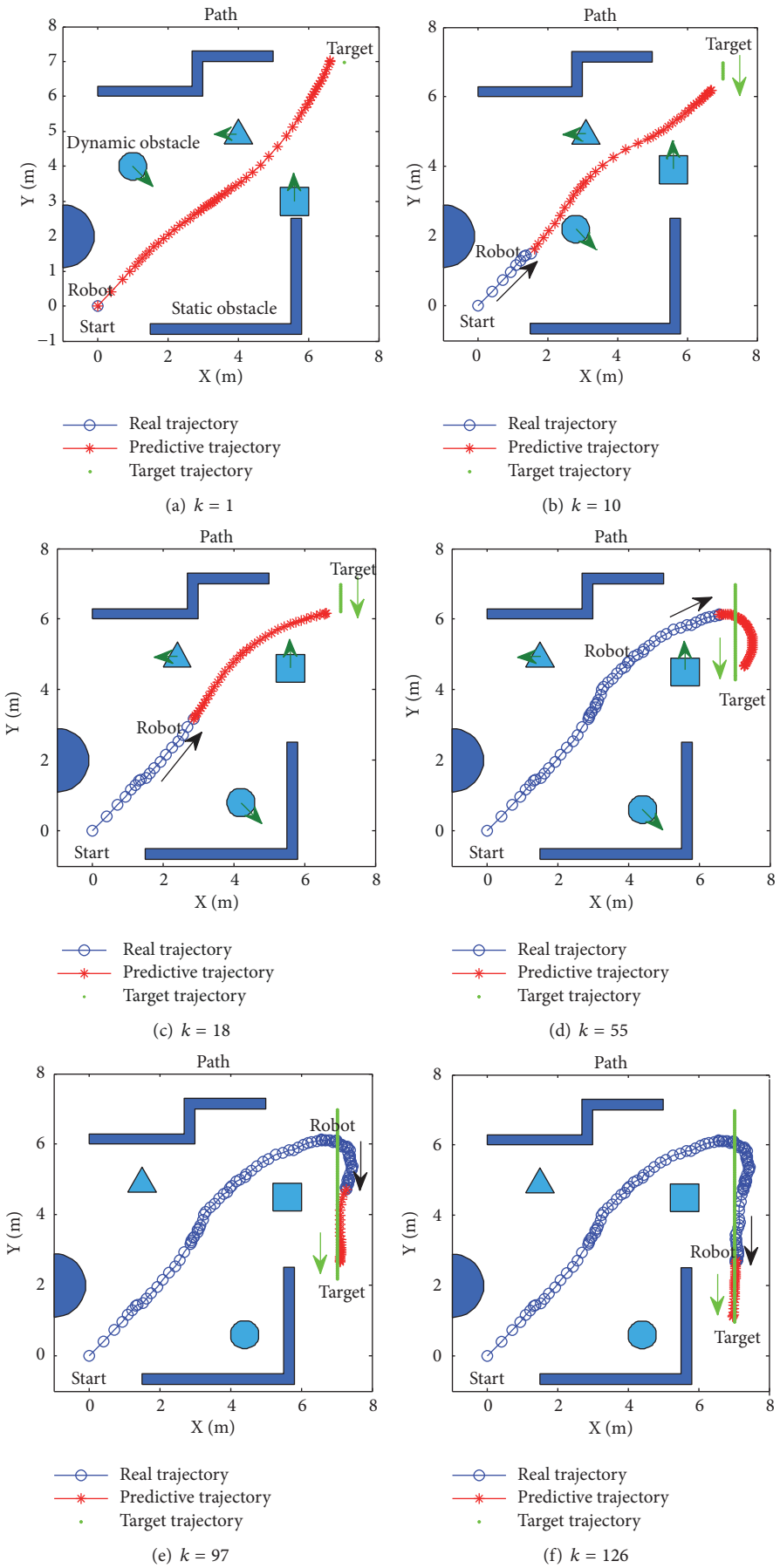
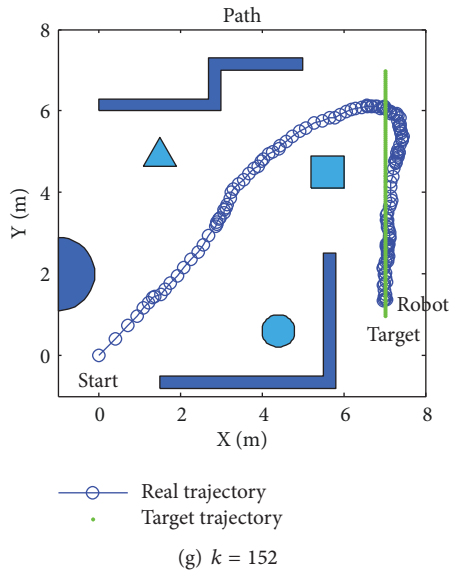


Figure 14: Continued.



(g) $k = 152$

Figure 14: Trajectory of robot in dynamic environment.

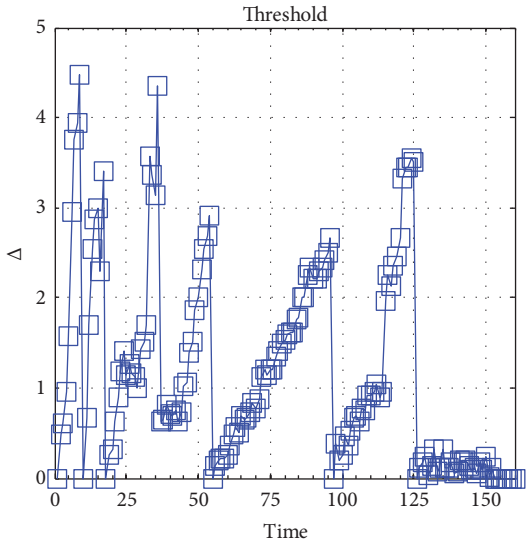


Figure 15: Changes of Δ.

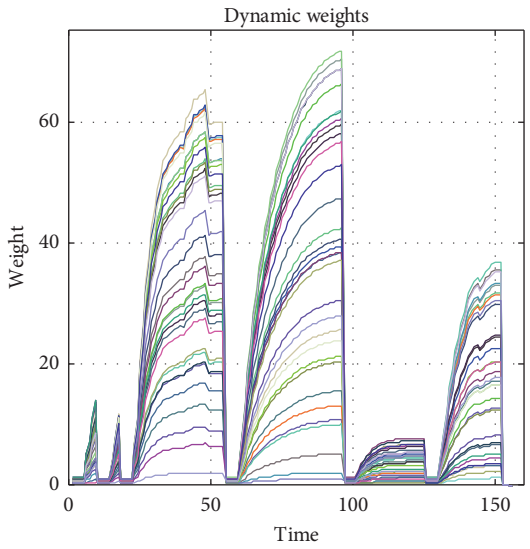
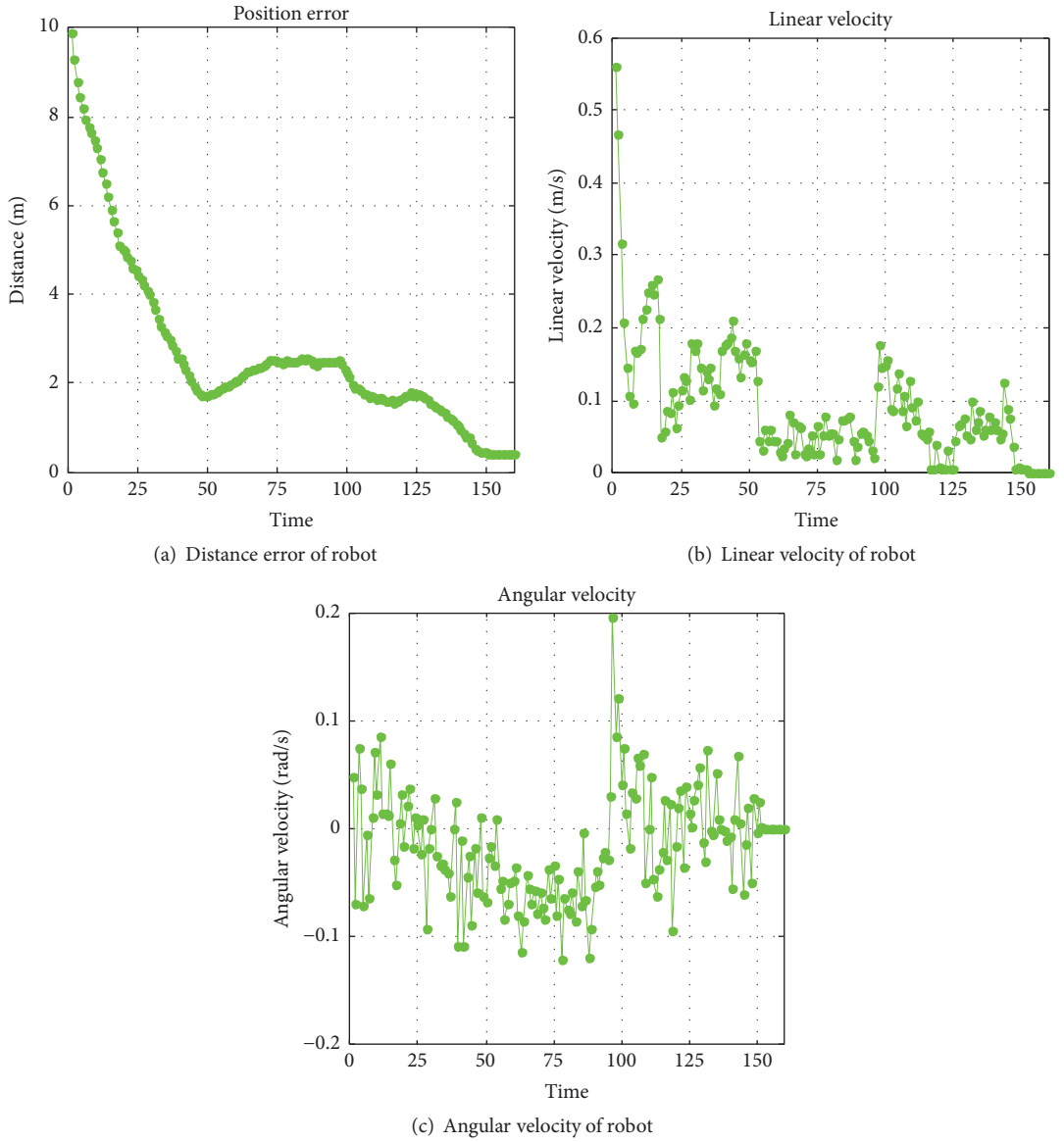


Figure 16: Changes of RFNN weights.

shown in Figure 14. Dynamic obstacles referred to in Figure 7 keep moving during process. Trajectories of obstacles are shown in Figures 14(a)–14(d). Target also keeps moving during the whole process. RFNN needs to update predictions depending on changes of surroundings limited by threshold settled above. And measured Δ varying with time is shown in Figure 15. Each state in Figures 14(a)–14(f) is corresponding to zero point of Δ in Figure 15. Six predictions are generated during the whole process. As seen in Table 4, the first four predictions are generated due to (17) ~ (19) and last two are generated due to runout of steps. Changes of RFNN weights are shown in Figure 16. Detailed information of different phases is shown in Table 4. Position error between robot and target is shown in Figure 17(a). Linear and angular velocity of robot is shown in Figures 17(b) and 17(c). As shown above, optimal and robust navigation is achieved in dynamic environment.

5.6. Hardware Experiments on a Vehicle. In this section, we will validate navigation strategy presented in this paper on a differential driven vehicle. The onboard processor of vehicle is STM322F407 that is responsible for real-time control. All the online computation, including localization, motion planning, and optimal control command generation, is performed on a computer with Intel i5 2.3 GHz processor and 8 GB RAM.



(a) Distance error of robot

(b) Linear velocity of robot

(c) Angular velocity of robot

Figure 17: Position error and control inputs of robot in dynamic environment.

Table 4: Evaluation of trajectories in dynamic environment.

Trajectory	1	2	3	4	5	6
Cost of prediction (s)	4.13	1.47	1.25	0.38	0.02	0.003
Predicted step	63	49	42	51	29	27
Actual step	9	8	37	42	29	27
Predicted distance (m)	9.72	5.11	4.93	2.08	2.06	1.54
Actual distance (m)	2.16	1.87	5.27	2.37	2.30	1.59
Predicted terminal error (m)	0.36	0.41	0.38	0.41	0.42	0.37
Actual terminal error (m)	7.45	5.40	1.87	2.55	1.92	0.37

Commands are sent through a 2.4 GHz Bluetooth transmitter at an update rate of 50 Hz.

The experiment is carried out in indoor environment as shown in Figure 18. There are two moving tracked mobile robots between our robot and target. The robot needs to reach target and avoid collision with them. The software running on the computer is developed based on the ROS middleware. 2D map of surroundings is established using LIDAR carried on the front of robot. Adaptive Monte Carlo localization method is used to localize robot's position. RFNN based motion

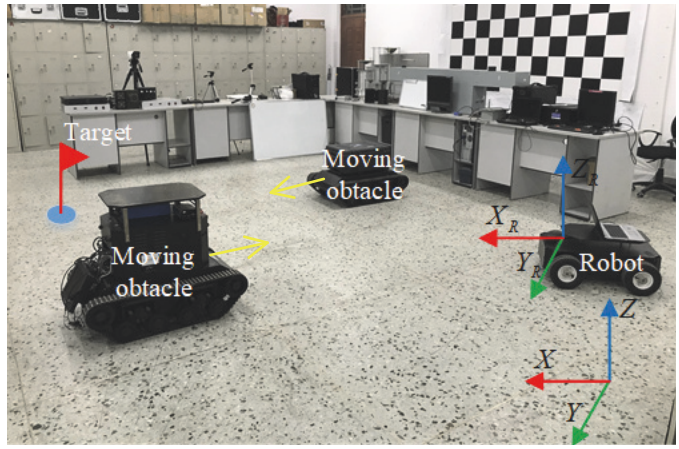
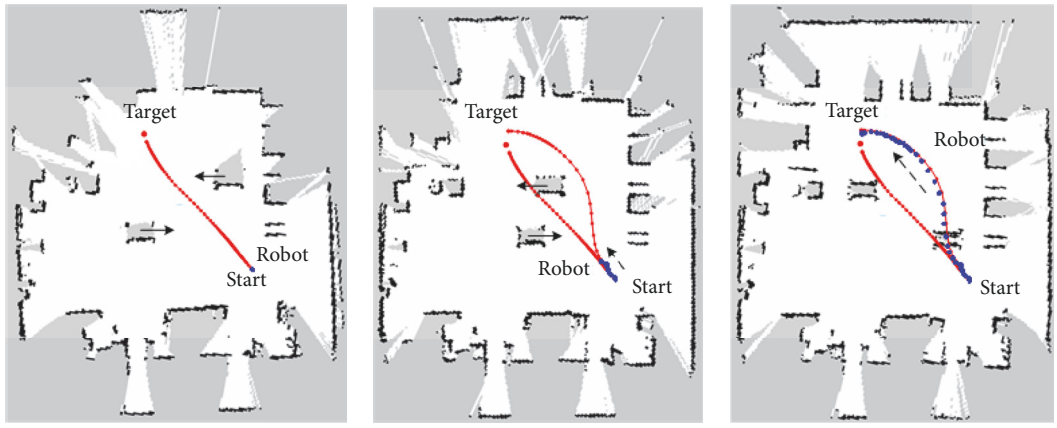


Figure 18: Experimental surroundings.



(a) (b) (c)

Figure 19: Performance of vehicle. Trajectories in red are predictions, and trajectory in blue reflects actual positions of vehicle.

Table 5: Comparison between simulation and hardware experiment.

	Predicted distance (m)	Actual distance (m)	Predicted step	Total prediction cost (times) (s)	Average computation cost (s)	Total time taken (s)	Average tracking error (m)
Simulation	15.186	15.606	153	7.253(6)	0.551	102	0.230
Hardware	7.361	7.577	47	0.727(2)	0.539	65	0.156

planner is used for global optimization programming. NMPC based controller is used for robust optimal control. The performance is shown in Figure 19. The trajectories in red are optimal predictions of robot's motion. Optimal prediction is updated due to moving obstacle. Figure 20 illustrates changes of Δ which is tanglesome contrasted to Figure 15. Trajectory in blue is actual motion of robot. Motion state is reflected in Figure 21. Computation cost and changes of weights are, respectively, shown in Figures 22 and 23. Comparison between hardware and simulation experiment in Section 5.5 is shown in Table 5. As prediction cost depends on distance between vehicle and target, simulation experiment takes up more computation than hardware experiment and prediction time is more. Average computation cost of both experiments at each step is about half a second. Average tracking error of hardware experiment is higher than simulation for the reasons of effective location method in actual situation. According to hardware experiment, our method's effectiveness is proved.

6. Conclusion

In this paper, a combined nonlinear model predictive control and recurrent fuzzy neural network approach is designed to achieve robust and optimal navigation in unknown and dynamic environment. Controller is separated into two parts

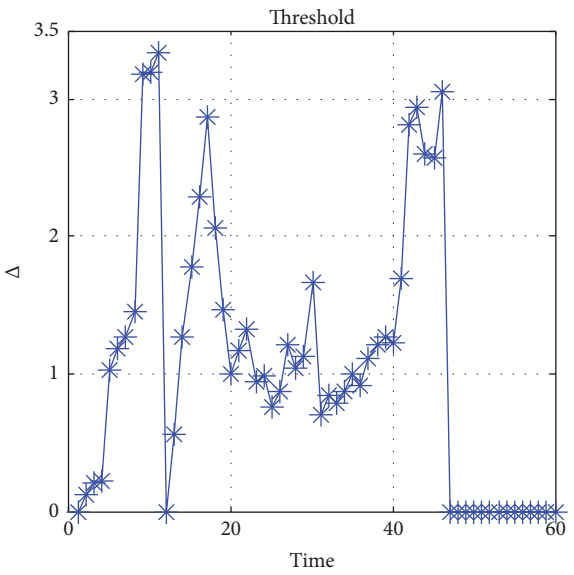


Figure 20: Changes of Δ.

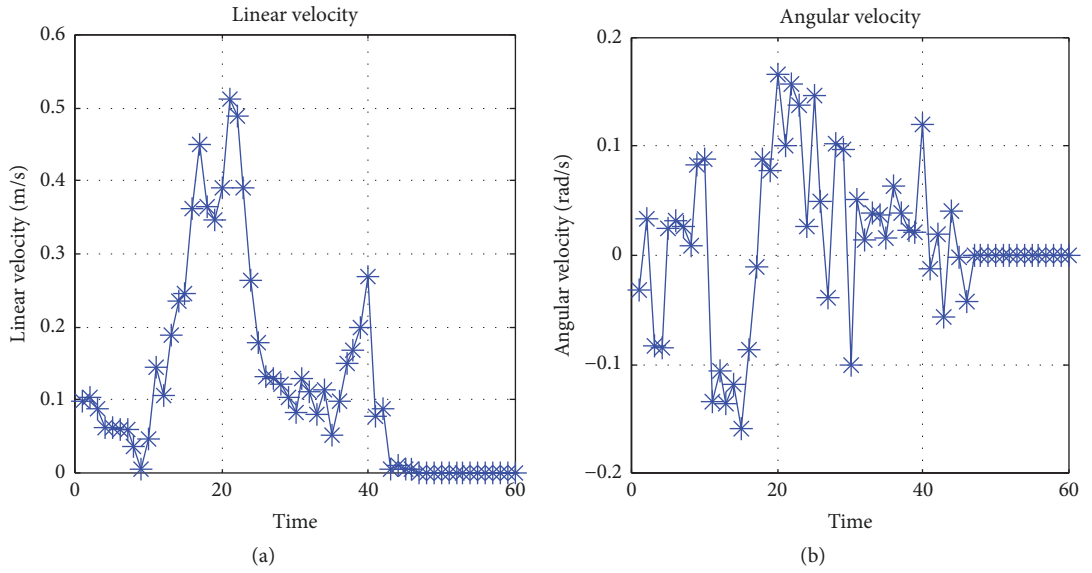


Figure 21: Motion condition of vehicle.

depending on disturbances of system forming dual-layer structure. RFNN based dynamic program is used to predict optimal trajectory in unknown environment. NMPC is used to track optimal trajectories against location and model error. By applying our method to wheeled mobile robot, simulation experiment is carried out. According to analysis of experiment results, robustness and optimization of designed strategy are proved. Although proposed method is used in ground robot system, it is also applicable to UUV, UAV, space aircraft, etc. Because problem described in Section 2 and working in Section 3 are not specific to a certain model, in our future work, perception ability mentioned in Section 1 will be considered in practical navigation system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

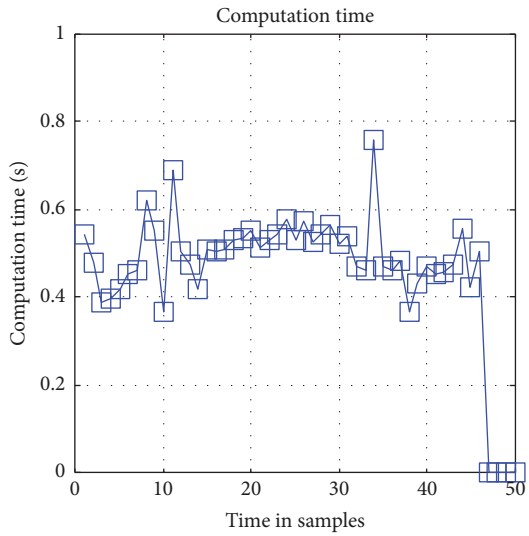


Figure 22: Computation cost.

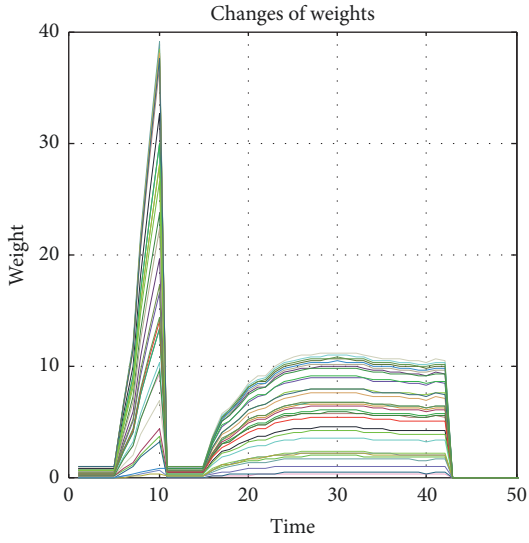


Figure 23: Changes of RFNN weights.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant 61673129, the Natural Science Foundation of Heilongjiang Province of China under Grant F201414, and the Fundamental Research Funds for the Central Universities under Grant HEUCF160418.

Supplementary Materials

There is a video in the supplementary material file. It is simulation experiment result that describes robot's motion in dynamic environment and reflects effectiveness of the navigation method. The video corresponds to Figures 14(a)–14(d) in the manuscript. It is uploaded to show the whole process intuitively for editors. *(Supplementary Materials)*

References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: a survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[2] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[3] H. Wang, G. Fu, J. Li, Z. Yan, and X. Bian, "An adaptive UKF based SLAM method for unmanned underwater vehicle," *Mathematical Problems in Engineering*, vol. 2013, Article ID 605981, 12 pages, 2013.

[4] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: a review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014.

[5] Z. Fang, S. Yang, S. Jain et al., "Robust Autonomous Flight in Constrained and Visually Degraded Shipboard Environments," *Journal of Field Robotics*, vol. 34, no. 1, pp. 25–52, 2017.

[6] B. Kim and J. Pineau, "Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[7] J. Crespo, R. Barber, and O. M. Mozos, "Relational Model for Robotic Semantic Navigation in Indoor Environments," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 3, pp. 1–23, 2017.

[8] C. Wang, A. V. Savkin, and M. Garratt, "A strategy for safe 3D navigation of non-holonomic robots among moving obstacles," *Robotica*, vol. 36, pp. 275–297, 2018.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2006.

[10] P. K. Mohanty and D. R. Parhi, "Optimal path planning for a mobile robot using cuckoo search algorithm," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 35–52, 2016.

[11] H. Ishihara and E. Hashimoto, "Moving obstacle avoidance for the mobile robot using the probabilistic inference," in *Proceedings of the 2013 10th IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2013*, pp. 1771–1776, Japan, August 2013.

[12] M. Wang and J. N. K. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 625–643, 2008.

[13] Bo Shen, Zidong Wang, Jinling Liang, and Yurong Liu, "Recent Advances on Filtering and Control for Nonlinear Stochastic Complex Systems with Incomplete Information: A Survey," *Mathematical Problems in Engineering*, vol. 2012, Article ID 530759, 16 pages, 2012.

[14] Y. Wang, L. Cheng, Z.-G. Hou, J. Yu, and M. Tan, "Optimal formation of multirobot systems based on a recurrent neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 322–333, 2016.

[15] Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 310–321, 2013.

[16] C.-J. Kim and D. Chwa, "Obstacle avoidance method for wheeled mobile robots using interval type-2 fuzzy neural network," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 3, pp. 677–687, 2015.

[17] D. Q. Mayne, "Model predictive control: recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[18] T. Wang, H. Gao, and J. Qiu, "A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control," *IEEE Transactions on Neural Networks and Learning Systems*, no. 99, 2015.

[19] P. Falugi and D. Q. Mayne, "Getting robustness against unstructured uncertainty: a tube-based MPC approach," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1290–1295, 2014.

[20] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[21] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015.

[22] G. Flores, S. T. Zhou, R. Lozano, and P. Castillo, "A vision and GPS-based real-time trajectory planning for a MAV in unknown and low-sunlight environments," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 59–67, 2014.

[23] S. K. Sharma, R. Sutton, A. Motwani, and A. Annamalai, "Nonlinear control algorithms for an unmanned surface vehicle," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 228, no. 2, pp. 146–155, 2014.

[24] R. A. Aliev, B. G. Guirimov, B. Fazlollahi, and R. R. Aliev, "Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: recurrent fuzzy neural networks," *Fuzzy Sets and Systems*, vol. 160, no. 17, pp. 2553–2566, 2009.

[25] J. de Jesús Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm," *Neurocomputing*, vol. 70, no. 13-15, pp. 2460–2466, 2007.

[26] Z. Hou, R. Chi, and H. Gao, "An overview of dynamic-linearization-based data-driven control and applications," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, pp. 4076–4090, 2017.

[27] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking," *International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[28] X. Y. Wang and Y. Huang, "Convergence study in extended Kalman filter-based training of recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 22, no. 4, pp. 588–600, 2011.

[29] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[30] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and Design*, Madison, Wis, USA, 2009.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[32] J. X. Xu, "A survey on iterative learning control for nonlinear systems," *International Journal of Control*, vol. 84, no. 7, pp. 1275–1294, 2011.

[33] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[34] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[35] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.