Neural Comput & Applic (2018) 30:447–462
https://doi.org/10.1007/s00521-016-2643-7

ORIGINAL ARTICLE

Trajectory tracking control of wheeled mobile manipulator based on fuzzy neural network and extended Kalman filtering

Kerui Xia[1] · Haibo Gao[1] · Liang Ding[1] · Guangjun Liu[2] · Zongquan Deng[1] · Zhen Liu[1] · Changyou Ma[1]

Received: 24 November 2015 / Accepted: 21 October 2016 / Published online: 17 November 2016

Abstract For robot trajectory tracking control, it is necessary to model inverse dynamics system sufficiently well to allow high-performance control. However, for complex robots such as wheeled mobile manipulators (WMMs), it is often difficult to model the dynamics system owing to system uncertainties, nonlinearity, and coupling. In this paper, we propose an effective tracking control method based on fuzzy neural network (FNN) and extended Kalman filter (EKF) to achieve WMM followed reference trajectory efficiently. The FNN is trained to generate a feedforward torque. In order to increase the computational efficiency and precision of the training algorithm, the EKF is used to sequentially update both the output weights and centers of the FNN. The effectiveness of the proposed control algorithm is confirmed through system experiments.

Keywords Trajectory tracking · Wheeled mobile manipulator · Fuzzy neural network · Extended Kalman filter

1 Headings

Traditionally, the trajectory tracking problem either in a kinematic or in a dynamic control level is well known that under the assumption of the model is appropriate simplification. However, for complex robots such as wheeled mobile manipulators (WMMs), it is more realistic to consider the tracking problem with system uncertainties, nonlinearity, and coupling to realize high-performance control. In kinematic control level, authors have studied the tracking problem using adaptive controls and robust controls considering system uncertainties and nonlinearity as disturbances [1–5]. However, the dynamics usually cannot be neglected if accurate motion is required. Some adaptive controls and robust controls have been developed to confront the uncertainties and nonlinearity in the dynamics of the systems. In [6, 7], adaptive control techniques were employed to solve the tracking problem of WMM with unknown inertia parameters. In [8, 9], a robust adaptive controller was designed for robot systems with model uncertainties. Dong et al. [10] have studied the motion/-force control for a mobile manipulator. In addition, they have proposed some adaptive robust control strategies. In [11], robust adaptive controllers were proposed for dynamic systems with parametric and nonparametric uncertainties, in which adaptive control techniques were used to compensate for the parametric uncertainties and sliding mode control was used to suppress the bounded disturbances. In [12–16], robust control strategies were presented systematically in the presence of uncertainties and disturbances. However, the major problem of the adaptive robust control approach is that certain functions must satisfy the assumption of "linearity in the parameters," and tedious preliminary computation is required to determine "regression matrices."

In the past decade, great progress has been achieved in the study of using neural networks to control nonlinear systems with uncertain. Extensive works demonstrate that adaptive neural control is particularly suitable for controlling highly uncertain, nonlinear, and complex systems

✉ Haibo Gao
gaohaibo@hit.edu.cn

[1] State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150001, China

[2] Department of Aerospace Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada

[17–20]. In these neuro-adaptive control schemes, the neural network is used to study dynamic system or compensate the effects of system nonlinearity and uncertainties. Therefore, the stability, convergence, and robustness of the control system can be improved. A multilayer perception network-based controller was suggested by Fierro and Lewis [21] to deal with parametric or nonparametric uncertainties for a mobile robot without any prior knowledge of the uncertainties. Other neural networks, such as radial basis function (RBF) neural network [22], wavelet network [23, 24], and fuzzy neural network (FNN), were also adopted for the robust control of mobile manipulators.

The FNN has been widely used for trajectory tracking control of WMMs owing to the advantages in stability, convergence, and robustness. The FNN is essentially a neural network. In this network, the structure is determined using a number of fuzzy rules. The key to evaluate a FNN is to see its generalization ability. The most important effect of the generalization ability is the network structure of the selection. If the number of nodes is not appropriate, it can easily lead to the problem, of over-fitting or training phenomenon, and reduce the generalization ability of the neural network. The traditional learning method mostly uses the back-propagation algorithm, which has low speed, easy to fall into local minimum [25]. Others have used unsupervised procedures (e.g., *k*-means clustering or Kohonen's self-organizing maps [26]) for selecting the centers [27]. Theoretical justification of the suitability of such a strategy is presented in [28]. Nevertheless, in order to acquire optimal performance, the centers training procedure should also include the target data [29], leading to a form of supervised learning that proved to be superior in several applications [30]. Owning to the advantage of quick converge and converge to global minima, extended Kalman filters (EKFs) have been used extensively with neural networks. They have been used to train multilayer perceptron [31–33] and recurrent networks [34, 35]. They have also been used to train RBF networks [36, 37].

In this study, we propose an effective tracking control method based on FNN and EKF to track a reference trajectory (e.g., trajectory for opening a door or grasping an object) in the proposed WMM system. Firstly, we identify the WMM system, FNN, and EKF using a simple description. Secondly, to deal with complex dynamics system, we developed a FNN to approximate the dynamics model. In order to decrease the computational effort of the training algorithm, a pair of parallel EKF is used to sequentially update both the weights and centers, following the line of previous work related to considering the training procedure of a neural network as an estimation problem [38–40]. Thirdly, an effective tracking control law for the WMM system is proposed with system stability analysis. Finally, the proposed tracking control method facilitates precise tracking performance, which was demonstrated using some simulations and experiments.

This paper is organized as follows. The background of the WMM system, FNN, and EKF are introduced in Sect. 2. Section 3 discusses the FNN learning scheme, and a pair of parallel EKF is used to sequentially update both the weights and centers of the network. The effective tracking control strategy for the WMM system is proposed based on the stability analysis in Sect. 4. Some simulations and experiments are presented in Sect. 5. Finally, some conclusions are presented in Sect. 6.

2 Background

2.1 WMM model

In this study, consider the WMM consisting of the wheeled mobile vehicle and 6-link manipulator, as shown in Fig. 1. The dynamics of WMM can be given by the dynamics of wheeled mobile vehicle and 6-link manipulator [41].

$$\begin{aligned}&\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{G}(\boldsymbol{q})+\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{\Gamma}^{-1}\boldsymbol{\tau}_s\\&\quad=\boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau}-\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{\lambda}\end{aligned}\tag{1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ are the joint angles, velocities, and accelerations of the mobile manipulator, respectively. $\boldsymbol{M}(\boldsymbol{q}) \in \Re^{8\times 8}$ is the symmetric bounded positive definite inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \Re^{8\times 8}$ denotes the centripetal and Coriolis torques, $\boldsymbol{G}(\boldsymbol{q}) \in \Re^{8\times 1}$ is gravity matrix, $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \Re^{8\times 1}$ is a vector representing frictional force, $\boldsymbol{\tau}_s \in \Re^{8\times 1}$ denotes the coupling torque, $\boldsymbol{\Gamma} \in \Re^{8\times 8}$ denotes the reduction radio of the speed reducer, $\tau \in \Re^{8\times 1}$ is the actuation input, $\boldsymbol{B}(\boldsymbol{q}) \in \Re^{8\times 8}$ is a full-rank input transformation matrix, which is assumed to be known because it is a function of the fixed geometry of the system, $\boldsymbol{J}^{\boldsymbol{T}}(\boldsymbol{q}) \in \Re^{8\times 8}$ is the Jacobian matrix, and $\boldsymbol{\lambda} \in \Re^{8\times 1}$ denotes the constraining force. From

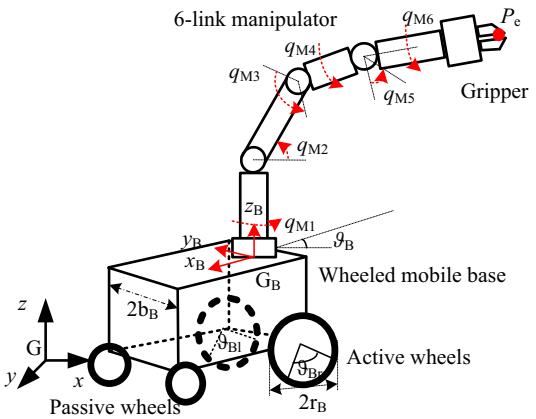

Fig. 1 Coordinate of wheeled mobile manipulator

Fig. 1, we can define $\boldsymbol{q} = [\vartheta_{\mathrm{Bl}} \quad \vartheta_{\mathrm{Br}} \quad q_{M1} \quad q_{M2} \quad q_{M3} \quad q_{M4} q_{M5} q_{M6}]^{\mathrm{T}}$.

By analogy with a similar concept introduced in [41], the dynamics model of WMM considering kinematics can be derived as follows.

$$\tilde{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \tilde{\boldsymbol{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v} + \tilde{\boldsymbol{G}}(\boldsymbol{q}) + \tilde{F}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \tilde{\boldsymbol{\tau}}_s = \tilde{\boldsymbol{\tau}} \tag{2}$$

$$\tilde{\boldsymbol{M}}(\boldsymbol{q}) \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\boldsymbol{M}(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q}_{\mathrm{B}})$$

$$\tilde{\boldsymbol{C}}(\boldsymbol{q},\dot{\boldsymbol{q}}) \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\{\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{S}}(\boldsymbol{q}_{\mathrm{B}}) + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{S}(\boldsymbol{q}_{\mathrm{B}})\}$$

$$\tilde{\boldsymbol{G}}(\boldsymbol{q}) \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\boldsymbol{G}(\boldsymbol{q})$$

$$\tilde{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}}) \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}})$$

$$\tilde{\boldsymbol{\tau}}_s \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\Gamma^{-1}\boldsymbol{\tau}_s$$

$$\tilde{\boldsymbol{\tau}}_s \triangleq \boldsymbol{S}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{B}})\boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau}$$

2.2 Fuzzy neural network

In this paper, a FNN is used, as shown in Fig. 2. Here, r is the number of input variables; x_i ($i = 1, 2, \ldots, r$) is the input linguistic variables; y_i ($i = 1, 2, \ldots, m$) is the output of the system; MF_{ij} is the i-th input variables of the j-th membership function; R_j is the j-th fuzzy rule; w_{mj} is a result parameter of the j-th rule; and u is the rule number of the system.

The following is a detailed description for all the sub-layers of the network.

The first layer is the input layer. Each node represents a variable input language.

The second layer is the membership function layer. Each node represents a membership function. The membership function is the Gauss function

$$\mu_{ij}(x_i) = \exp\left[-\frac{1}{\sigma_{ij}^2}||x_i - c_{ij}||^2\right]; i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, u \tag{3}$$

where r is the number of input variables, u is numbers of membership functions, also represents the total number of rules; μ_{ij} is the j-th Gaussian membership function of x_i; c_{ij} is the j-th Gaussian membership function center of x_i; and σ_{ij} is the j-th Gaussian membership function width of x_i.

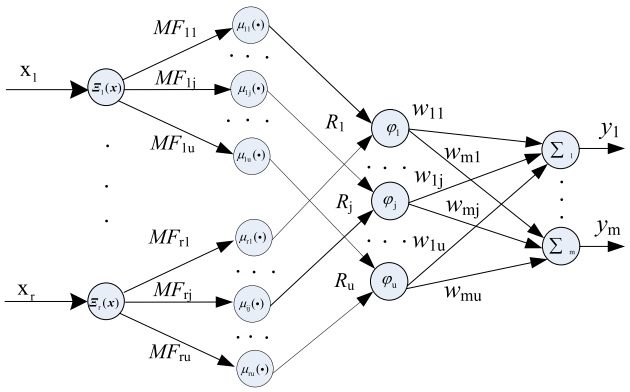

Fig. 2 Structure of a two-layer FNN

The third layer is a T-norm layer. Each node represents the IF part of the possible fuzzy rules, also on behalf of the NN units, and the layer node number reflects the number of fuzzy rules. If the T-norm operator used to compute each rule's firing strength is multiplication, the output of the j-th rule $R_j (j = 1, 2, \ldots, u)$ in the third layer is

$$\varphi_{\mathrm{j}} = \exp\left[-\sum_{i=1}^{r}\frac{1}{\sigma_{ij}^2}\left\|x_i - c_{ij}\right\|^2\right] \quad j = 1, 2, \ldots, u \tag{4}$$

The fourth layer is the output layer. Each node in this layer represents an output variable as the weighted summation of incoming signals. In this study, we consider multi-input and multi-output systems in the following analysis.

$$\mathrm{y}_j = \sum_{i=1,j=1}^{i=p,j=u} w_{ij}\varphi_j \tag{5}$$

where y_j is the value of j-th output variable and w_{ij} is the THEN-part (consequent parameters) or connection weight of the i-th rule.

From the above, the fuzzy rule numbers and the third-layer numbers of the FNN units are same. The first layer is used to realize input variables language; the second layer is equivalent to the fuzzy generator, which is used to calculate the membership functions; the third layer is equivalent to the fuzzy rule base, using the T-norm operator multiplication on the degree of membership of the second layer; and the fourth layer is used to obtain the output. The rule from the input to output is defined as

$$R^j : \mathrm{IF}\, x_1 \,\mathrm{is}\, \mathrm{MF}_{1j} \,\mathrm{and}\ldots\mathrm{and}\, x_r \,\mathrm{is}\, \mathrm{MF}_{rj},\ \mathrm{THEN}\, \mathrm{y}_j \,\mathrm{is}\, w_j$$

where w_j is a real number, a system based on fuzzy S model is used to realize the network, and the output of the S model is

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{j=1}^{u} a_{j0}\exp\left[-\frac{1}{\sigma_j^2}\left\|\boldsymbol{x} - \boldsymbol{C}_j\right\|^2\right] \tag{6}$$

Assume n observation data has produced a u fuzzy rule, the third layer of the FNN units output can be written in matrix form as follows

$$\phi = \begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1n} \\ \vdots & \vdots & \vdots \\ \varphi_{u1} & \cdots & \varphi_{un} \end{bmatrix}$$

where φ_{ij} is the output of the third layer of the i-th neurons when the j-th training data reaches the network.

The output of the system can be obtained using Eq. (5), and hence, Eq. (5) can be rewritten as follows

$$\boldsymbol{W}\phi = \boldsymbol{Y} \tag{7}$$

where $\boldsymbol{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1u} \\ w_{21} & w_{22} & \cdots & w_{2u} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nu} \end{bmatrix}$, $\boldsymbol{Y} = [\mathrm{y}_1\, y_2 \cdots y_n]$.

2.3 Extended Kalman filter

The Kalman filter is usually used to estimate the state variables of a discrete process using a linear stochastic differential equation as $\boldsymbol{x} \in \Re^n$ [42]. However, if the relationship between the process and (or) the observed variables and the observed variables are nonlinear, the direct use of the Kalman filter is not the case. The Kalman filter, which is expected to be linear with the covariance, is known as the EKF, which is used to solve the problem of nonlinear filtering.

In the nonlinear case, the partial derivative of the process and observation equation can be used to estimate the state of the process by using the partial derivative and observation equation. The state vector is assumed as $\boldsymbol{x} \in \Re^n$, the state equation is a nonlinear stochastic differential equation given as:

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{8}$$

The observation variable is given as $\boldsymbol{z} \in \Re^m$ and can be defined as

$$\boldsymbol{z}_k \in \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{9}$$

where the random variables $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process noise and observation noise, respectively. These variables, which are Gauss white noise, are independent of each other. In differential Eq. (8), the nonlinear function $\boldsymbol{f}$ maps the state to the present moment k, and its parameters include driving function $\boldsymbol{u}_{k-1}$ and zero mean noise $\boldsymbol{w}_k$. The nonlinear function $\boldsymbol{h}$ in Eq. (9) is connected with the state variable $\boldsymbol{x}_k$ and the observed variable $\boldsymbol{v}_k$.

Equation (10) is the estimation function of the state vector and Eq. (11) is the estimation function of the observation vector. $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process noise and observation noise, respectively, which are not easy to be estimated by mathematical formula, in practical applications. So, we did not consider them in this step. That is to say, they are regarded as zero.

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{f}(\tilde{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) \tag{10}$$

$$\tilde{\boldsymbol{z}}_k = \boldsymbol{h}(\tilde{\boldsymbol{x}}_k, 0) \tag{11}$$

where the $\tilde{\boldsymbol{x}}_k$ is a posteriori estimation of the state relative to the first time k.

In order to include the process of a nonlinear difference and observation error in practical applications, we can use the new control equation by Eqs. (12) and (13)

$$\boldsymbol{x}_k \approx \tilde{\boldsymbol{x}}_k + \boldsymbol{A}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{W}\boldsymbol{w}_{k-1} \tag{12}$$

$$\boldsymbol{z}_k \approx \tilde{\boldsymbol{z}}_k + \boldsymbol{H}(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k) + \boldsymbol{V}\boldsymbol{v}_{\mathrm{k}} \tag{13}$$

where $\boldsymbol{x}_k$ is the actual value of the state vector. $\boldsymbol{z}_k$ is the actual value of the observation vector. $\tilde{\boldsymbol{x}}_k$ is obtained using Eq. (10) and is the observation value of the state vector. $\tilde{\boldsymbol{z}}_k$ is calculated using Eq. (11), which is the observation value of the observation vector. $\hat{\boldsymbol{x}}_k$ is a posteriori estimation of the state vector of the k moment. The random variables $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process noise and observation noise, in practical applications, which are feedback from sensors. $\boldsymbol{A}$ is the Jacoby matrix of the partial derivative of $\boldsymbol{f}$ to $\boldsymbol{x}$, which can be rewritten as

$$\boldsymbol{A}_{[i,j]} = \frac{\partial \boldsymbol{f}_{[i]}}{\partial \boldsymbol{x}_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) \tag{14}$$

$\boldsymbol{W}$ is the Jacoby matrix of the partial derivative of $\boldsymbol{f}$ to $\boldsymbol{w}$.

$$\boldsymbol{W}_{[i,j]} = \frac{\partial \boldsymbol{f}_{[i]}}{\partial \boldsymbol{w}_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) \tag{15}$$

$\boldsymbol{H}$ is the Jacoby matrix of the partial derivative of $\boldsymbol{h}$ to $\boldsymbol{x}$.

$$\boldsymbol{H}_{[i,j]} = \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{x}_{[j]}}(\tilde{\boldsymbol{x}}_k, 0) \tag{16}$$

$\boldsymbol{V}$ is the Jacoby matrix of the partial derivative of $\boldsymbol{h}$ to $\boldsymbol{v}$.

$$\boldsymbol{V}_{[i,j]} = \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{v}_{[j]}}(\tilde{\boldsymbol{x}}_k, 0) \tag{17}$$

Prediction error is

$$\tilde{\boldsymbol{e}}_{\boldsymbol{X}_k} \equiv \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k \tag{18}$$

The residuals of the observed variables are

$$\tilde{\boldsymbol{e}}_{\boldsymbol{z}_k} \equiv \boldsymbol{z}_k - \tilde{\boldsymbol{z}}_k \tag{19}$$

Furthermore, the expression of the error can be recorded as Eqs. (20) and (21).The covariance matrix is

$$\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k} \approx \boldsymbol{A}(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}) + \boldsymbol{\varepsilon}_k \tag{20}$$

$$\tilde{\boldsymbol{e}}_{\boldsymbol{z}_k} \approx \boldsymbol{H}\,\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k} + \boldsymbol{\eta}_k \tag{21}$$

where $\boldsymbol{\varepsilon}_k$ and $\boldsymbol{\eta}_k$ are zero mean. The covariance matrix is the independent random variables of $\boldsymbol{W}\boldsymbol{Q}\boldsymbol{W}^{\mathrm{T}}$ and $\boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^{\mathrm{T}}$. $\boldsymbol{Q}$ is obtained from $\boldsymbol{p}(\boldsymbol{w}) \sim \boldsymbol{N}(0, \boldsymbol{Q})$, $\boldsymbol{R}$ is obtained from $\boldsymbol{p}(\boldsymbol{v}) \sim \boldsymbol{N}(0, \boldsymbol{R})$. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are Gauss white noise series which are independent of each other and have normal distribution.

By using the observation residuals $\tilde{\boldsymbol{e}}_{\boldsymbol{z}_k}$ in Eq. (19) and the second Kalman filter (hypothesis) estimates $\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k}$ in Eq. (20), the estimation results denoted as $\hat{\boldsymbol{e}}_{\boldsymbol{k}}$, combining

Eq. (18) together to obtain the initial nonlinear process a posteriori state estimation, i.e.,

$$\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k + \hat{\boldsymbol{e}}_k \tag{22}$$

The random variables in Eqs. (20) and (21) are approximate to the probability distribution.

$$\boldsymbol{p}(\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k}) \sim \boldsymbol{N}(0, \boldsymbol{E}[\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k}\tilde{\boldsymbol{e}}_{\boldsymbol{x}_k}^{\mathrm{T}}]) \tag{23}$$

$$\boldsymbol{p}(\boldsymbol{\varepsilon}_k) \sim \boldsymbol{N}(0, \boldsymbol{W}\boldsymbol{Q}\boldsymbol{W}^{\mathrm{T}}) \tag{24}$$

$$\boldsymbol{p}(\boldsymbol{\eta}_k) \sim \boldsymbol{N}(0, \boldsymbol{V}\boldsymbol{R}_{\boldsymbol{k}}\boldsymbol{V}^{\mathrm{T}}) \tag{25}$$

According to the above approximation, the Kalman filter is used to estimate the equation. $\hat{\boldsymbol{e}}_k$ can be written as

$$\hat{\boldsymbol{e}}_k = \boldsymbol{K}_k\tilde{\boldsymbol{e}}_{z_k} \tag{26}$$

Substituting Eqs. (26) and (19) into Eq. (22), we can observe that the second Kalman filter is not actually required

$$\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}_k\tilde{\boldsymbol{e}}_{z_k} = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}_k(z_k - \tilde{z}_k) \tag{27}$$

Equation (27) can be used to extend the observation variables of Kalman filtering, where $\hat{\boldsymbol{x}}_k$ and $\tilde{z}_k$ are derived from Eqs. (10) and (11), respectively.

3 A self-organizing learning algorithm

In general, training a neural network is a challenging nonlinear optimization problem. Various derivative-based methods have been used to train neural networks. However, they typically tend to converge more slowly or tend to converge to local minima. The Kalman filter is a well-known estimation procedure of a vector of parameters from the available measured data. They have been used to train multilayer perceptions and recurrent networks. The need essential of recursive Kalman filtering algorithm for neural network training, except an orderly way to update the weights and centers of the network, is a two-order differential information approximation error variance matrix that also needs to be maintained and updated, which has the advantage of converging quickly and converge to global minima in the training process. In this study, we extend the use of the Kalman filters to train general multi-input multi-output FNN based on using a pair of parallel EKF to sequentially update both the weights and centers of the network known as the self-organizing learning algorithm.

As mentioned in Sect. 2, the third layer of each node represents the IF part of the possible fuzzy rules or the FNN units. If the number of fuzzy rules requires an identification system, we cannot select the pre-structure of the fuzzy neural networks. Therefore, this study presents a new learning algorithm, which can automatically determine the fuzzy rules of the system, to achieve the system of specific performance.

3.1 Production standards of fuzzy rules

In the fuzzy neural network, if the fuzzy rule number is too high, it will not only increase the system complexity and computational burden, but also reduce the generalization ability of the network. If it is too low, the system will not be able to contain input and output state space completely, and will degrade the network performance. Joining the new fuzzy rules or not depends on three important factors: the system error, accommodate boundary, and the error reduction rate.

3.1.1 The system error

The error criterion can be described as follows: for the i-th observation data $(\boldsymbol{x}_{\mathrm{o}i},\ \boldsymbol{y}_{\mathrm{o}i})$, where $\boldsymbol{x}_{\mathrm{o}i}$ is the input vector; $\boldsymbol{y}_{\mathrm{o}i}$ is the desired output. The current structure of the network of all outputs $\boldsymbol{y}_i$ is calculated by using Eq. (3). System error is defined as:

$$\|\boldsymbol{e}_i\| = \|\boldsymbol{y}_{\mathrm{o}i} - \boldsymbol{x}_{\mathrm{o}i}\|;\ i = 1, 2, \ldots, n \tag{28}$$

If

$$\|\boldsymbol{e}_i\| > k_e;\ k_e = \max\left[\boldsymbol{e}_{\max} \times \beta^i,\ \boldsymbol{e}_{\min}\right] \tag{29}$$

then a new rule should be considered; otherwise, new rules should not be produced. The term k_e consists of preselected values according to the expected network precision. $\boldsymbol{e}_{\max}$ is a pre-defined maximum error, $\boldsymbol{e}_{\min}$ is the desired output accuracy, and $\beta(0<\beta<1)$ is the convergence factor.

3.1.2 Accommodate boundary

In a way, the efficient performance of FNN learning is the partition of the input space. The FNN structure and performance and the input membership functions are closely related. In this study, we use the Gauss membership function, and output decreases with increase in the center distance. If a new sample is located in a preexisting Gaussian membership function within the scope of coverage, the new samples can use the existing Gaussian membership function, and they do not need the network to generate new Gaussian units.

Accommodate boundary: For the i-th observation data $(\boldsymbol{x}_i,\ \boldsymbol{y}_i)$, we calculated the i-th input values $\boldsymbol{x}_i$ with the existing FNN units in the center of the $\boldsymbol{c}_i$ between distance $d_i(j)$

$$d_i(j) = \left\|\boldsymbol{x}_i - \boldsymbol{c}_j\right\|;\ i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, u \tag{30}$$

where u is the number of existing fuzzy rules or NN unit.

Define

$$d_{i,\min} = \arg\ \min\left(d_i(j)\right) \tag{31}$$

If

$$d_{i,\min} > k_d, k_d = \max[d_{\max} \times \gamma^i, d_{\min}] \tag{32}$$

then the existing input membership functions cannot effectively partition the input space. It is necessary to add a new fuzzy rule. If not, the observation data will be represented by using the existing nearest FNN units. The term k_d is the effective radius of the accommodate boundary, $d_{\max}$ is the maximum length of the input space, $d_{\min}$ is the minimum length, and $\gamma(0<\gamma<1)$ is the attenuation factor.

3.1.3 The error reduction rate

In traditional learning algorithms, the learning speed will be reduced because of the pruning process, and increase the computational burden [45, 46]. In this study, a new growth criterion is proposed. The algorithm does not need the pruning process; thus, the learning process will speed up.

Given n input/output data $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, $i = 1, 2, \ldots, n$, consider Eq. (3) as a special case of a linear regression model; the linear regression model can be rewritten as

$$\boldsymbol{y}(i) = \sum_{j=1}^{u} \boldsymbol{h}_j(i)\boldsymbol{\theta}_j + \boldsymbol{\varepsilon}(i) \tag{33}$$

Equation (9) can be abbreviated as

$$\boldsymbol{D} = \boldsymbol{H\Theta} + \boldsymbol{E} \tag{34}$$

where $\boldsymbol{D} = \boldsymbol{T}^{\mathrm{T}} \in \Re^n$ is the expected output, $\boldsymbol{H} = \boldsymbol{\Psi}^{\mathrm{T}} \in \Re^{n\times u}$ is a regressor, and $\boldsymbol{\Theta} = \boldsymbol{W}^{\mathrm{T}} \in \Re^u$ is a weighting vector; suppose that $\boldsymbol{E} \in \Re^n$ is not correlated with the regression error vector quantity.

For matrix $\boldsymbol{\Psi}$, if its line number is greater than the number of columns, by the decomposition of QR:

$$\boldsymbol{H} = \boldsymbol{PQ} \tag{35}$$

$\boldsymbol{H}$ transforms into a set of orthogonal basis vectors set $\boldsymbol{P} = [\mathrm{p}_1, \ \mathrm{p}_2, \ \ldots, \ \mathrm{p}_u] \in \Re^{n\times u}$. The dimension is same as that of $\boldsymbol{H}$, the column vector consists of an orthogonal basis, and $\boldsymbol{Q} \in \Re^{\mathrm{u}\times u}$ is an upper triangular matrix. Through this transformation, it is possible to calculate the contribution of each component to the expected energy output from each base vector. Substituting Eq. (11) into Eq. (10), one can obtain

$$\boldsymbol{D} = \boldsymbol{PQ\Theta} + \boldsymbol{E} = \boldsymbol{PG} + \boldsymbol{E} \tag{36}$$

Linear least-squares solution of $\boldsymbol{G}$ is $\boldsymbol{G} = (\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P})^{-1}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{D}$, or

$$\mathrm{g}_k = \frac{p_k^{\mathrm{T}}\boldsymbol{D}}{p_k^{\mathrm{T}}p_k}; \ k = 1, \ 2, \ \ldots, \ u \tag{37}$$

$\boldsymbol{Q}$ and $\boldsymbol{\Theta}$ satisfy the following equation

$$\boldsymbol{Q\Theta} = \boldsymbol{G} \tag{38}$$

When $k \neq l$, p_k and p_l is orthogonal; the quadratic sum of $\boldsymbol{D}$ is given by Eq. (15)

$$\boldsymbol{D^T D} = \sum_{k=1}^{u} g_k^2 p_k^{\mathrm{T}} p_k + \boldsymbol{E^T E} \tag{39}$$

On eliminating the mean, the variance of $\boldsymbol{D}$ is given by Eq. (16)

$$n^{-1}\boldsymbol{D^T D} = n^{-1}\sum_{\mathrm{k}=1}^{u} g_k^2 p_k^{\mathrm{T}} p_k + n^{-1}\boldsymbol{E^T E} \tag{40}$$

From Eq. (16), we can see that $n^{-1}\sum_{\mathrm{k}=1}^{u} g_k^2 p_k^{\mathrm{T}} p_k$ is a part of expected output variance brought about by regressor p_k. Thus, the error rate of decline of p_k can be defined as

$$\mathrm{err}_k = \frac{g_k^2 p_k^{\mathrm{T}} p_k}{\boldsymbol{D^T D}}, \ 1 \le k \le u \tag{41}$$

Substituting Eq. (13) into Eq. (17), one can obtain

$$\mathrm{err}_k = \frac{(p_k^{\mathrm{T}}\boldsymbol{D})^2}{p_k^{\mathrm{T}} p_k \boldsymbol{D^T D}}, \ 1 \le k \le u \tag{42}$$

Equation (18) provides a simple and effective method to return the vales of the quantum set, its significance lies in err_k revealed the similarity of p_k and $\boldsymbol{D}$. The larger the value of err_k and $\boldsymbol{D}$, said p_k similarity is greater, and it has a greater effect on output p_k. Using err_k defining generalized factor (GF), GF can test the generalized ability of the algorithm, and this further simplifies and speeds up the learning process. Definition:

$$\mathrm{GF} = \sum_{k=1}^{u} \mathrm{err}_k \tag{43}$$

Equation (43) can be used to calculate the fuzzy rule numbers. If $\mathrm{GF} < k_{\mathrm{GF}}$, then it is necessary to increase additional new fuzzy rules; otherwise, the additional new rules is not necessary where the value of k_{GF} is a preselected thresholds value.

3.2 Parameter training by extended Kalman filter

Please note that irrespective of newly generated hidden nodes or those that are already present, the network parameters need to be adjusted. In this study, we have used the EKF method to adjust the parameters of the network. A neural network of nonlinear system equations can be written as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k \tag{44}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\theta}_k) \tag{45}$$

where $\boldsymbol{h}(\boldsymbol{\theta}_k)$ is a nonlinear mapping between the input and the output of the network. To achieve a stable EKF algorithm, process noise and observation noise are added to the system Eqs. (44) and (45), which can be rewritten as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{\omega}_k \tag{46}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\theta}_k) + \boldsymbol{v}_k \tag{47}$$

where $\boldsymbol{\omega}_k$ and $\boldsymbol{v}_k$ denote the added noise.

In fact, by using the EFK, one can adjust all the parameters of the network. However, the global method will involve large matrix operations and increase the computational burden. Therefore, we can divide the global problem into a series of sub-problems. Because the center and width of the network are nonlinear, it can be adjusted using the EKF algorithm.

The adjustment of the parameters of the former parts: Because of the nonlinear characteristics of the front end of the network, it can be used as follows: The EKF algorithm is used to update the center and width parameters of the membership function of Gauss.

$$\boldsymbol{K}_i^{\delta} = \boldsymbol{P}_{i-1}^{\delta}\boldsymbol{G}_i^{\mathrm{T}}[\boldsymbol{R}_i + \boldsymbol{G}_i\boldsymbol{P}_{i-1}^{\delta}\boldsymbol{G}_i^{\mathrm{T}}]^{-1} \tag{48}$$

$$\boldsymbol{\delta}_{\mathrm{i}} = \boldsymbol{\delta}_{i-1} + \boldsymbol{K}_i^{\delta}(\boldsymbol{T}_i - \boldsymbol{W}_{i-1}\phi_i) \tag{49}$$

$$\boldsymbol{P}_{\mathrm{i}}^{\delta} = \boldsymbol{P}_{i-1}^{\delta} - \boldsymbol{K}_i^{\delta}\boldsymbol{G}_i\boldsymbol{P}_{i-1}^{\delta} + \boldsymbol{Q}_i \tag{50}$$

where $\boldsymbol{\delta}_{\mathrm{i}}$ is the center or width of the membership function of Gauss, $\boldsymbol{K}_i^{\delta}$ is Kalman gain matrix, and $\boldsymbol{G}_i$ is the partial derivative of the center or width parameter of the network output for the membership function of Gauss $\boldsymbol{\delta}$. $\boldsymbol{P}_i^{\delta}$ is the forecast error variance matrix, $\boldsymbol{Q}_i$ is a process noise covariance matrix, and $\boldsymbol{R}_i$ is the measurement noise covariance matrix.

$$\boldsymbol{G}_i = \left.\frac{\partial \boldsymbol{Y}_{\mathrm{i}}}{\partial \boldsymbol{\delta}_{i-1}}\right|_{\delta=\hat{\delta}_{i-1}} = -2\boldsymbol{\delta}_{i-1}\left\|\boldsymbol{X}_i - \boldsymbol{C}_j\right\|^2 \sum_{j=1}^{u} \boldsymbol{w}_j\varphi_j \tag{51}$$

After adjustment of the parameters: Because the back end of the network has a linear feature, the following Kalman filtering algorithm can be used to update the parameters.

$$\boldsymbol{K}_i^{\mathrm{w}} = \boldsymbol{P}_{i-1}^{w}\phi_i^{\mathrm{T}}[\boldsymbol{R}_i + \phi_i\boldsymbol{P}_{i-1}^{w}\phi_i^{\mathrm{T}}]^{-1} \tag{52}$$

$$\boldsymbol{W}_{\mathrm{i}} = \boldsymbol{W}_{i-1} + \mathbf{K}_i^{\mathrm{w}}(\boldsymbol{T}_i - \boldsymbol{W}_{i-1}\phi_i) \tag{53}$$

$$\boldsymbol{P}_{\mathrm{i}}^{\mathrm{w}} = \boldsymbol{P}_{i-1}^{w} - \boldsymbol{K}_i^{w}\phi_i\boldsymbol{P}_{i-1}^{w} + \boldsymbol{Q}_i \tag{54}$$

where $\boldsymbol{W}_{\mathrm{i}}$ is a post-parameter matrix, $\boldsymbol{K}_i^{\mathrm{w}}$ is Kalman gain matrix, $\boldsymbol{P}_{\mathrm{i}}^{\mathrm{w}}$ is the forecast error variance matrix, $\boldsymbol{Q}_i$ is a process noise covariance matrix, and $\boldsymbol{R}_i$ is the measurement noise covariance matrix.

3.3 The process of adding fuzzy rules

In the algorithm, the process of adding fuzzy rules is as follows.

Initial parameter allocation: When the first observation data $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ is obtained, the network is not yet established, so the data will be selected as the first fuzzy rule: $\boldsymbol{C}_1 = \boldsymbol{x}_1 = [x_{11} \quad x_{21} \quad \ldots \quad x_{r1}]^{\mathrm{T}}$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_0$, $\boldsymbol{w}_1 = \boldsymbol{y}_1$, where $\boldsymbol{\sigma}_0$ is a pre-defined constant.

Growth process: When the i-th observation data $(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{y}_i)$ is obtained, in the third layer, there is a u-th hidden neuron. According to Eqs. (28), (31), and (43), respectively, we can calculate $\boldsymbol{e}_i, d_{i,\min}$, GF.

If

$$\|\boldsymbol{e}_i\| \geq k_e, d_{i,\min} > k_d, \text{and GF} < k_{\mathrm{GF}} \tag{55}$$

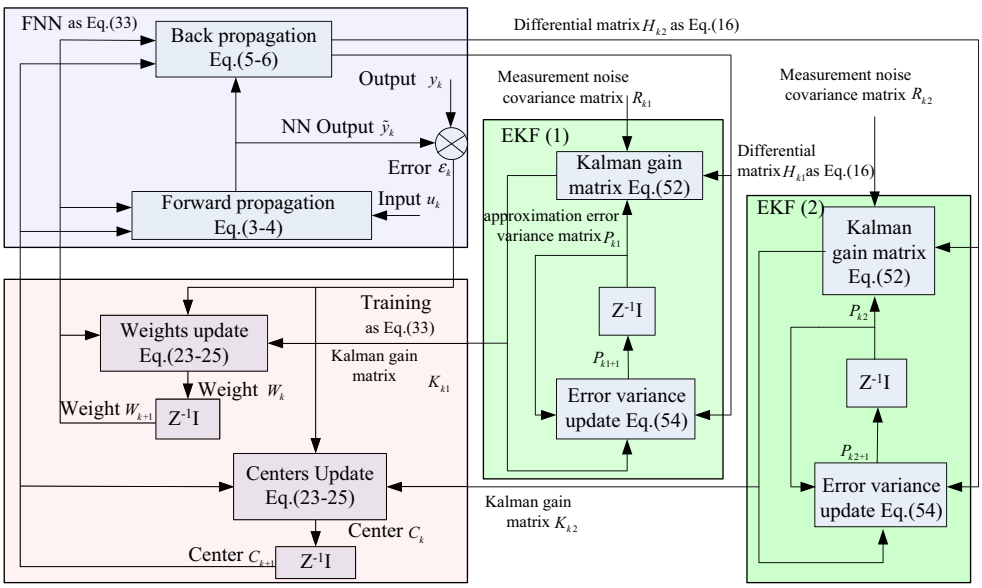

Fig. 3 Signal flow graph of FNN training based on using a pair of parallel EKF

then add a new hidden neuron, where k_e, k_d are given in Eqs. (29) and (32), respectively. The new increase in the center of the hidden neurons and weights assigned to the center as $\boldsymbol{C}_{u+1} = \boldsymbol{x}_i = [x_{1i} \quad x_{2i} \quad \ldots x_{ri}]^{\mathrm{T}}$, $\boldsymbol{\delta}_{u+1} = \boldsymbol{k}_0 d_{i,\min}$, $\boldsymbol{w}_{u+1} = \boldsymbol{e}_i$, where $\boldsymbol{k}_0(\|\boldsymbol{k}_0\| > 1)$ is the overlap factor.

Parameter adjustment: When the new neurons are added, the parameters of all the existing neurons are adjusted by using Eqs. (48–54).

Figure 3 illustrates the application of EKF to the training of general multi-input, multi-output FNN for both weights and centers.

4 Tracking controller design

Define $\boldsymbol{v}_{\mathrm{d}}$ as reference velocity. Take $\boldsymbol{v}_{\mathrm{d}}$ into the dynamics model of WMM (2), which can be rewritten as

$$\tilde{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{v}}_{\mathrm{d}} + \tilde{\boldsymbol{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_d + \tilde{\boldsymbol{G}}(\boldsymbol{q}) + \tilde{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \tilde{\boldsymbol{\tau}}_s = \tilde{\boldsymbol{\tau}} \tag{56}$$

The velocity error vector is defined as

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{v}_d & -\boldsymbol{v} \\ \dot{\boldsymbol{v}}_d & -\dot{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \xi \\ \dot{\xi} \end{bmatrix}. \tag{57}$$

Differentiating Eq. (57), using Eq. (2), and the mobile robot dynamics can be rewritten in terms of the velocity tracking error as

$$\tilde{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{\xi}} = -\tilde{\boldsymbol{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{\theta}) + \tilde{\tau}_s - \tilde{\boldsymbol{\tau}} \tag{58}$$

where the important nonlinear mobile robot function is defined as

$$\boldsymbol{g}(\boldsymbol{\theta}) = \tilde{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{v}}_{\mathrm{d}} + \tilde{\boldsymbol{C}}(\boldsymbol{q},\boldsymbol{v})\boldsymbol{v}_{\mathrm{d}} + \tilde{\boldsymbol{G}}(\boldsymbol{q}) + \tilde{\boldsymbol{F}}(\boldsymbol{q},\boldsymbol{v}) + \tilde{\tau}_s - \tilde{\boldsymbol{\tau}} \tag{59}$$

Here, the vector $\boldsymbol{\theta}$ can be measured by $\boldsymbol{\theta} \equiv \left[\boldsymbol{v}_d^{\mathrm{T}} \quad \dot{\boldsymbol{v}}_d^{\mathrm{T}} \quad \boldsymbol{v}^{\mathrm{T}}\right]^{\mathrm{T}}$.

Function $\boldsymbol{g}(\boldsymbol{\theta})$ contains all the mobile manipulator parameters, such as mass, moments of inertia, and friction coefficients. The system function $\boldsymbol{g}(\boldsymbol{\theta})$ is approximated using the FNN described by Eq. (7). The function $\boldsymbol{g}(\boldsymbol{\theta})$ can be rewritten as

$$\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{\theta}) + \boldsymbol{\varepsilon}(\boldsymbol{\theta}) \tag{60}$$

where $\boldsymbol{w} \in \Re^{(L+1)}$ is the vector of the ideal threshold and their weights. The bounds described by Eq. (60) are modified for $\boldsymbol{w}$ and $\boldsymbol{\varepsilon}(\boldsymbol{\theta})$ and expressed as

$$||\boldsymbol{w}|| \le b_w \text{ and } |\varepsilon(\boldsymbol{\theta})| \le b_\varepsilon \forall \boldsymbol{\theta}. \tag{61}$$

The FNN controller is connected in parallel with the PD controller and robust term to generate a compensated control signal. Theoretically, one can directly control the motor movement by the FNN controller. But in our real test process, we found it easy to diverge in joint tracking control process, since the errors in sensors signal acquisition and data fusion process exist. So we use a PD controller to prevent the divergence which caused by signal errors in the system and a robust term to ensure the robustness. They form the inner closed-loop system that controls the velocity error. The control law is given by:

$$\tilde{\boldsymbol{\tau}} = \tilde{\boldsymbol{g}} + \boldsymbol{KE} + d\mathrm{sgn}(\boldsymbol{\xi}) \tag{62}$$

where $\tilde{\boldsymbol{g}}$ is an estimate of $\boldsymbol{g}$. $\boldsymbol{KE}$ is the torque produced by the PD controller, and $d\mathrm{sgn}(\boldsymbol{\xi})$ is the robust term. An estimate of $\boldsymbol{g}(\boldsymbol{\theta})$ can be given by:

$$\tilde{\boldsymbol{g}}(\boldsymbol{\theta}) = \tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\phi}(\boldsymbol{\theta}) \tag{63}$$

where $\tilde{\boldsymbol{w}} \in \Re$ is the vector of the estimated threshold and weights. There is no simple or standard method of judging which choice is the best; hence, our assumption about $\tilde{\phi} = \phi$ is always feasible. A pair of parallel running extended Kalman filters then can be used to sequentially update both the weights and the centers of the network $\tilde{\boldsymbol{g}}$ as description in Sect. 3.

$\boldsymbol{KE}$ is given by

$$\boldsymbol{KE} = \boldsymbol{K}_{\mathrm{P}}\boldsymbol{\xi} + \boldsymbol{K}_{\mathrm{D}}\dot{\boldsymbol{\xi}} \tag{64}$$

where $\boldsymbol{K} = [\boldsymbol{K}_{\mathrm{P}} \quad \boldsymbol{K}_{\mathrm{D}}]$, $\boldsymbol{K}_P = \mathrm{diag}(k_4, k_5)]$, and $\boldsymbol{K}_{\mathrm{D}} = \mathrm{diag}(k_6, k_7)$ are positive matrix with real numbers.

Using the estimated function $\tilde{\boldsymbol{g}}(\boldsymbol{\theta})$ given by Eq. (63), the system control law (62) becomes:

$$\tilde{\boldsymbol{\tau}} = \tilde{\boldsymbol{w}}^{\mathrm{T}}\phi(\boldsymbol{\theta}) + \boldsymbol{K}_P\boldsymbol{\xi} + \boldsymbol{K}_D\dot{\boldsymbol{\xi}} + d\mathrm{sgn}(\boldsymbol{\xi}/\boldsymbol{\delta}) \tag{65}$$

and the parameter d is defined as:

$$d \ge b_\varepsilon + b_d + \frac{1}{4}\kappa b_w^2 + \in \tag{66}$$

which is related to the bounds described by Eq. (59), the parameter κ in Eq. (65), and a strictly positive constant $\in$.

The structure of the proposed tracking controller scheme is as shown in Fig. 4. The control system is shown in terms of the composition of a compensated controller and a Fuzzy neural network controller.

Next, we perform the system stability analysis of the closed-loop behavior of the proposed control methodology. Substituting the control input (65) into the mobile manipulator dynamics system described by Eq. (56) yields

$$(\tilde{\boldsymbol{M}} + \boldsymbol{K}_D)\dot{\boldsymbol{\xi}} = -(\boldsymbol{K}_P + \tilde{\boldsymbol{C}})\boldsymbol{\xi} + \hat{\boldsymbol{g}} - d\mathrm{sgn}(\boldsymbol{\xi}) \tag{67}$$

where $\hat{\boldsymbol{g}} = \boldsymbol{g} - \tilde{\boldsymbol{g}}$ is the function estimation error.

This estimation error is expressed in terms of Eq. (57) as

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}) = \hat{\boldsymbol{w}}^{\mathrm{T}}\phi(\boldsymbol{\theta}) + \boldsymbol{\varepsilon}(\boldsymbol{\theta}) \tag{68}$$

where $\hat{\boldsymbol{w}}$ is the vector of the threshold and weight estimation errors, defined as

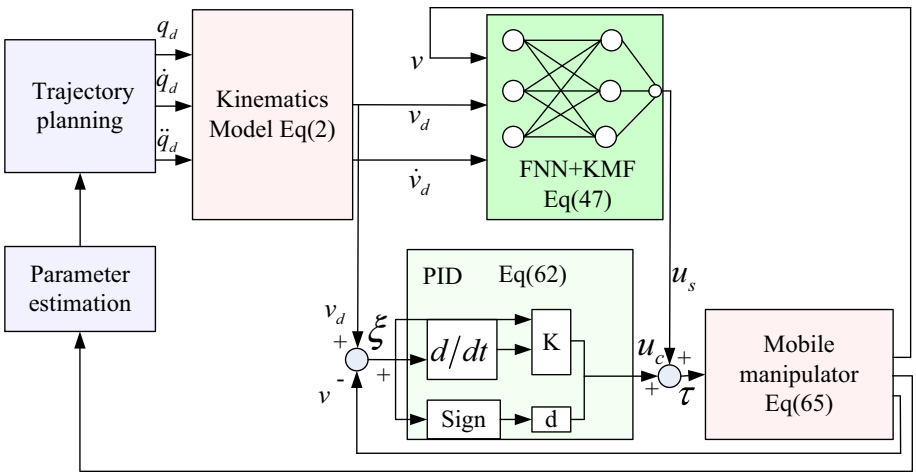

Fig. 4 Structure of the proposed fuzzy neural network-based motion control methodology

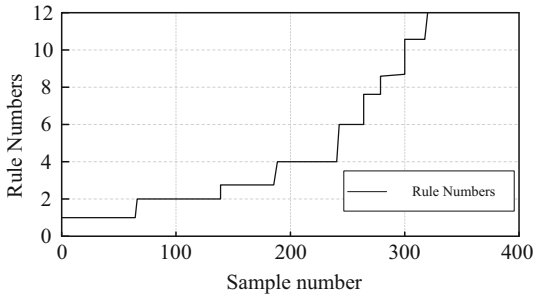

Fig. 5 Fuzzy rule generation

$$\hat{\boldsymbol{w}} = \boldsymbol{w} - \tilde{\boldsymbol{w}} \tag{69}$$

Therefore, Eq. (62) can be written as

$$(\tilde{\boldsymbol{M}} + \boldsymbol{K}_D)\dot{\boldsymbol{\xi}} = -(\boldsymbol{K}_P + \tilde{\boldsymbol{C}})\boldsymbol{\xi} + \hat{\boldsymbol{w}}^{\mathrm{T}}\phi(\boldsymbol{\theta}) + \boldsymbol{\varepsilon}(\boldsymbol{\theta}) - d\mathrm{sgn}(\boldsymbol{\xi}) \tag{70}$$

Consider the following Lyapunov function candidate

$$\mathrm{V}(\boldsymbol{\xi}, t) = \frac{1}{2}\boldsymbol{\xi}^{\mathrm{T}}\tilde{\boldsymbol{M}}\boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{K}_D\boldsymbol{\xi} + \frac{1}{2}\left[(\boldsymbol{w} - \tilde{\boldsymbol{w}})^{\mathrm{T}}(\underline{\ })\right] \tag{71}$$

Differentiation yields

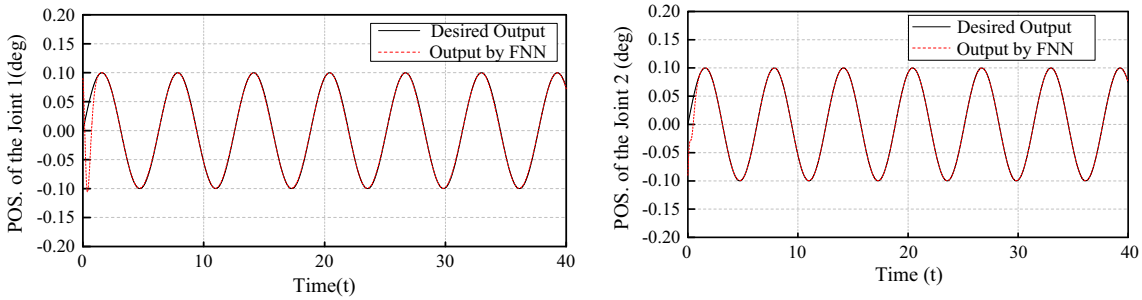

Fig. 6 Desired output and training output

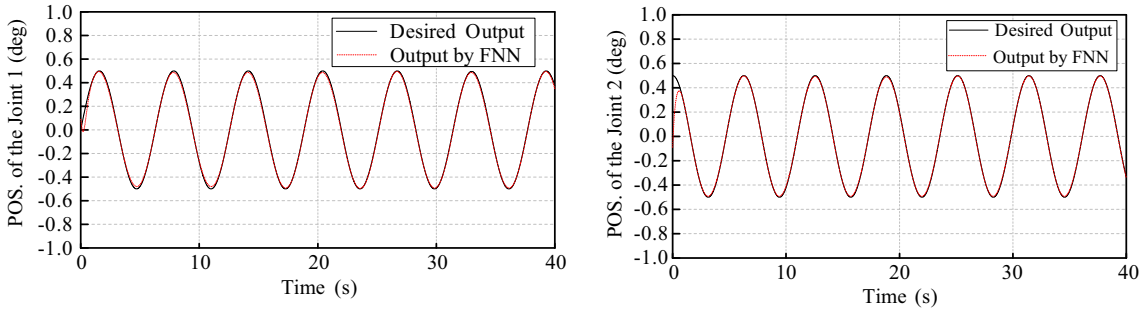

Fig. 7 Desired output and predicted output

$$
\begin{aligned}
\mathrm{V}(\xi,t) &= \xi^{\mathrm{T}}\tilde{\boldsymbol{M}}\dot{\tilde{\xi}} + \frac{1}{2}\xi^{\mathrm{T}}\dot{\tilde{\boldsymbol{M}}}\dot{\tilde{\xi}} + \xi^{\mathrm{T}}\boldsymbol{K}_D\dot{\tilde{\xi}} + \mathrm{tr}\left[(\boldsymbol{w}-\tilde{\boldsymbol{w}})^{\mathrm{T}}(\dot{\tilde{\boldsymbol{w}}})\right] \\
&= \xi^{\mathrm{T}}(-\tilde{C}\xi + \tilde{\boldsymbol{w}}^{\mathrm{T}}\phi - \boldsymbol{K}_D\dot{\tilde{\xi}} - \boldsymbol{K}_P\xi + \boldsymbol{\varepsilon} + \tilde{\tau}_s - d\,\mathrm{sgn}(\xi)) \\
&\quad + \frac{1}{2}\xi^{\mathrm{T}}\dot{\tilde{\boldsymbol{M}}}\dot{\tilde{\xi}} + \xi^{\mathrm{T}}\boldsymbol{K}_D\dot{\tilde{\xi}} - \mathrm{tr}\left[\tilde{\boldsymbol{w}}^{\mathrm{T}}(\phi\xi^{\mathrm{T}} - \kappa\|\xi\|\tilde{\boldsymbol{w}})\right] \\
&= \frac{1}{2}\xi^{\mathrm{T}}(\dot{\tilde{\boldsymbol{M}}} - 2\dot{\tilde{C}})\xi + \xi^{\mathrm{T}}\tilde{\boldsymbol{w}}^{\mathrm{T}}\phi - \xi^{\mathrm{T}}\boldsymbol{K}_P\xi - d\|\xi\| \\
&\quad + \xi^{\mathrm{T}}(\boldsymbol{\varepsilon} + \tilde{\tau}_s) - \mathrm{tr}\left[\tilde{\boldsymbol{w}}^{\mathrm{T}}\phi\xi^{\mathrm{T}} - \kappa\tilde{\boldsymbol{w}}^{\mathrm{T}}\|\xi\|\tilde{\boldsymbol{w}}\right] \\
&= \xi^{\mathrm{T}}\tilde{\boldsymbol{w}}^{\mathrm{T}}\phi - \boldsymbol{K}_P\|\xi\|^2 - d\|\xi\| + \xi^{\mathrm{T}}(\boldsymbol{\varepsilon} + \tilde{\tau}_s) - \mathrm{tr}\left[\xi^{\mathrm{T}}\tilde{\boldsymbol{w}}^{\mathrm{T}}\phi\right] \\
&\quad + \mathrm{tr}\left[\kappa\tilde{\boldsymbol{w}}\|\xi\|(\boldsymbol{w}-\tilde{\boldsymbol{w}})\right]
\end{aligned}
$$

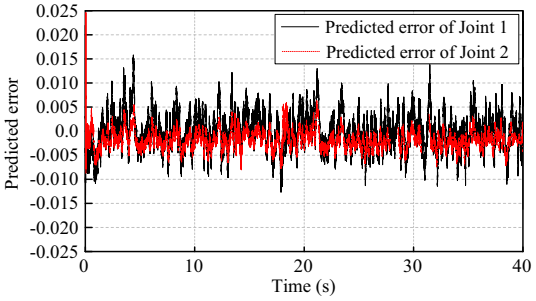

Fig. 8 Predicted error

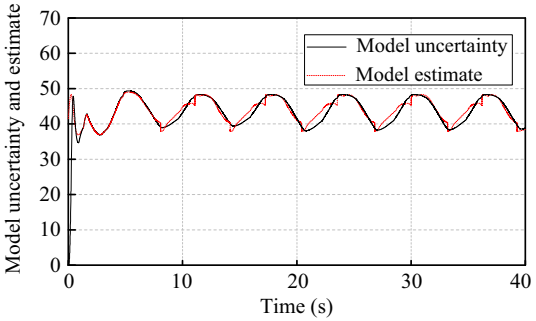

Fig. 9 Model uncertainty and estimate

$$
\begin{aligned}
&\le -\boldsymbol{K}_P\|\xi\|^2 - d\|\xi\| + \|\xi\|(\varepsilon + \tilde{\tau}_s) - \kappa\|\xi\|\|\tilde{\boldsymbol{w}}\|^2 + \kappa\|\xi\|\|\tilde{\boldsymbol{w}}\|b_w \\
&= -\boldsymbol{K}_P\|\xi\|^2 - d\|\xi\| + \|\xi\|(b_e + b_d) - \kappa\|\xi\|(\|\tilde{\boldsymbol{w}}\|^2 - \|\tilde{\boldsymbol{w}}\|b_w) \\
&\quad - \frac{1}{4}\kappa b_w^2\|\xi\| + \frac{1}{4}\kappa b_w^2\|\xi\| \\
&= -\boldsymbol{K}_P\|\xi\|^2 - d\|\xi\| + \|\xi\|(b_e + b_d) \\
&\quad - \kappa\|\xi\|\left(\|\tilde{\boldsymbol{w}}\|^2 - \|\tilde{\boldsymbol{w}}\|b_w + \frac{1}{4}b_w^2\right) + \frac{1}{4}\kappa b_w^2\|\xi\| \\
&= -\boldsymbol{K}_P\|\xi\|^2 - \|\xi\|\left\{d - b_e - b_d + \kappa\left[\left(\|\tilde{\boldsymbol{w}}\| - \frac{1}{2}b_w\right)^2 - \frac{1}{4}b_w^2\right]\right\} \\
&\le -\boldsymbol{K}_P\|\xi\|^2 - \|\xi\| + \kappa\left[\varepsilon + \left(\|\tilde{\boldsymbol{w}}\| - \frac{1}{2}b_w\right)^2\right]
\end{aligned}
\tag{72}
$$

Thus, $\mathrm{V} \ge 0$ and $\dot{V} \le 0$ are guaranteed to be negative, and this shows that $\mathrm{V} \to 0$ and implies $\xi \to 0$, while $\dot{\xi} \to 0$ as $t \to \infty$. Furthermore, Eq. (72) shows $\mathrm{V} = 0$ if and only if $\xi \to 0$. Therefore, $\boldsymbol{v} \to \boldsymbol{v}_d$, as $t \to \infty$.

Remark In practice, the velocity and tracking errors are not exactly equal to zero when using control law (66). The best we can guarantee is that the error converges in the neighborhood of the origin. The discontinuous function "sign" will give rise to control chattering because of imperfect switching in the computer control. This is undesirable because the

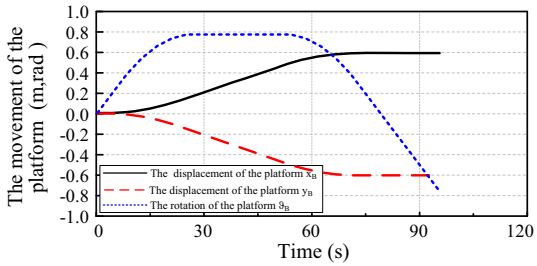

Fig. 11 Reference trajectory of the platform

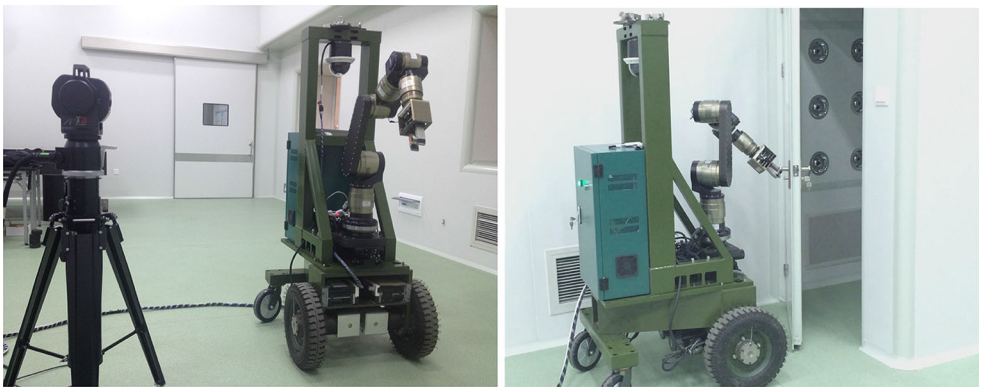

Fig. 10 Mobile manipulator used in this study

Table 1 Trajectory control points of mobile manipulator

Points pose	1	2	3	4	5	6	7
p_x	12.49	31.20	49.90	49.90	49.90	52.95	63.30
p_y	−536.72	−536.39	−533.58	−533.58	−533.58	−533.58	−533.58
p_z	324.80	287.45	259.47	259.47	259.47	233.53	209.41
p_o	280.40	284.50	288.58	288.58	288.58	316.94	331.59
p_a	151.16	155.01	157.87	157.87	157.87	149.67	137.63
p_t	13.90	16.25	18.52	18.52	18.52	46.74	61.25
p_{x_B}	0.00	0.00	100.00	200.00	300.00	300.00	300.00
p_{y_B}	0.00	0.00	−100.00	−200.00	−300.00	−300.00	−300.00
p_{θ_B}	0.00	45.00	45.00	45.00	45.00	0.00	−45.00

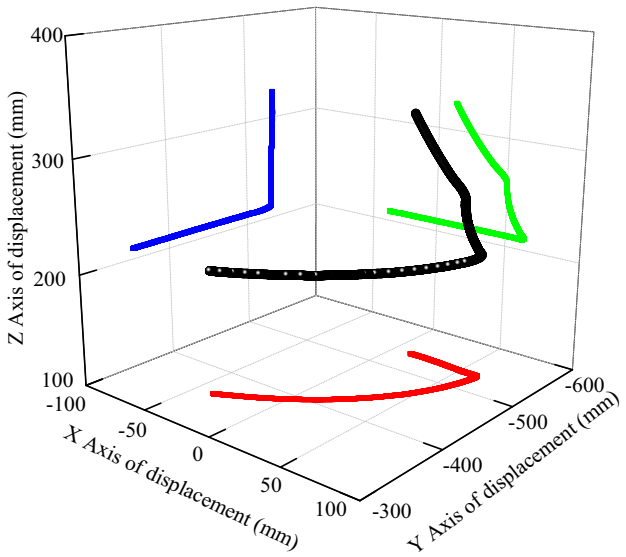

Fig. 12 Reference pose trajectory of the manipulator

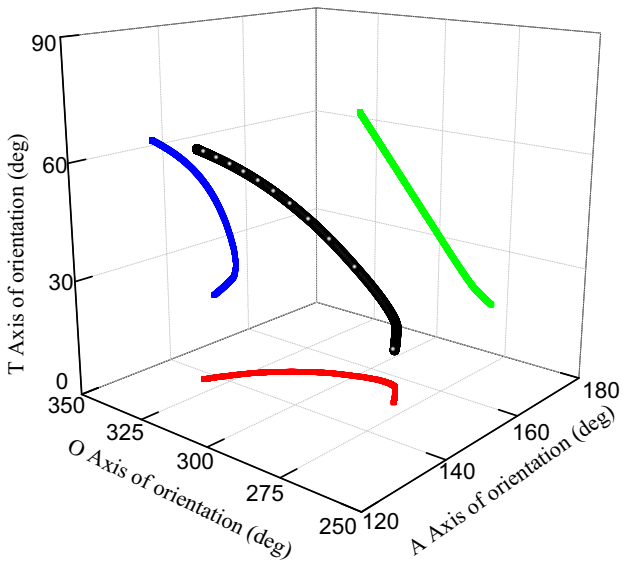

Fig. 13 Reference orientation trajectory of the manipulator

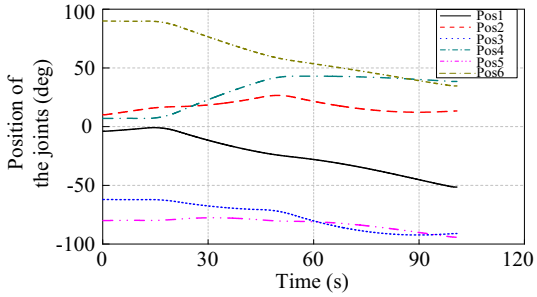

Fig. 14 Reference joints trajectory of the manipulator

unmodeled high-frequency dynamics might become excited. To avoid this, we use the boundary layer technique [43] to smooth the control signal. In a small neighborhood of the velocity error ($\xi = 0$), the discontinuous function "sign" is replaced by a boundary saturation function sat(ξ/δ). Thus, based on dynamics control, the robust neural network motion tracking control law (66) becomes

$$\tilde{\tau} = \tilde{w}^{\mathrm{T}}\phi(\theta) + K_P\xi + K_D\dot{\xi} + \mathrm{dsat}(\xi/\delta) \tag{73}$$

5 Simulation and experimental results

5.1 Trajectory tracking for simulation

In this simulation, we show the process of identification of the dynamic system of a manipulator and predict the future value using proposed control methodology. The dynamic system of the manipulator with two joints is described using Eq. (74), which is defined as:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \tag{74}$$

where $M(q) = \begin{bmatrix} p_1 + p_2 + 2p_3\cos q_2 & p_2 + p_3\cos q_2 \\ p_2 + p_3\cos q_2 & p_2 \end{bmatrix}$,

Table 2 Control parameters

Symbol	Description and initial value
$\tilde{w}$	The vector of the estimated threshold and weights $\begin{bmatrix} 82.6 & 4.5 & 5.5 & 4.7 & 5.0 & 5.2 & 5.8 & 4.9 \\ 12.3 & 94.5 & 3.4 & 4.0 & 3.9 & 0.7 & 1.4 & 3.0 \\ 10.4 & 1.2 & 75.4 & 2.2 & 1.5 & 4.7 & 4.5 & 1.2 \\ 11.2 & 0.6 & 0.7 & 95.4 & 1.5 & 3.8 & 1.4 & 1.3 \\ 10.2 & 0.5 & 0.1 & 2.5 & 93.7 & 3.2 & 0.9 & 4.0 \\ 12.0 & 1.2 & 0.9 & 3.9 & 4.0 & 83.5 & 1.6 & 1.2 \\ 9.8 & 0.3 & 0.4 & 4.0 & 5.9 & 4.7 & 80.0 & 1.5 \\ 7.8 & 0.9 & 0.1 & 4.5 & 3.1 & 4.5 & 0.7 & 94.2 \end{bmatrix}$
C	Gaussian membership function center $\begin{bmatrix} 0.204 & 0.022 & 0.015 & 0.011 & 0.003 & 0.012 & 0.005 & 0.006 \\ 0.020 & 0.221 & 0.006 & 0.005 & 0.001 & 0.009 & 0.005 & 0.006 \\ 0.032 & 0.030 & 0.312 & 0.002 & 0.005 & 0.008 & 0.001 & 0.001 \\ 0.013 & 0.005 & 0.013 & 0.151 & 0.010 & 0.003 & 0.007 & 0.001 \\ 0.025 & 0.015 & 0.005 & 0.001 & 0.321 & 0.003 & 0.011 & 0.012 \\ 0.022 & 0.032 & 0.002 & 0.012 & 0.012 & 0.252 & 0.007 & 0.003 \\ 0.005 & 0.012 & 0.021 & 0.002 & 0.011 & 0.007 & 0.300 & 0.011 \\ 0.001 & 0.005 & 0.030 & 0.006 & 0.007 & 0.003 & 0.005 & 0.250 \end{bmatrix}$
K_D	Differential coefficient matrix in PD loop $\begin{bmatrix} 10.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 12.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 11.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 10.4 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 14.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 12.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 10.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 11.3 \end{bmatrix}$

Symbol	Description and initial value
σ	Gaussian membership function width $\begin{bmatrix} 0.534 & 0.045 & 0.034 & 0.023 & 0.026 & 0.068 & 0.014 & 0.013 \\ 0.042 & 0.443 & 0.012 & 0.014 & 0.019 & 0.043 & 0.013 & 0.012 \\ 0.068 & 0.062 & 0.623 & 0.004 & 0.025 & 0.022 & 0.001 & 0.003 \\ 0.034 & 0.012 & 0.027 & 0.324 & 0.023 & 0.013 & 0.015 & 0.002 \\ 0.056 & 0.034 & 0.012 & 0.005 & 0.655 & 0.015 & 0.029 & 0.027 \\ 0.044 & 0.068 & 0.013 & 0.028 & 0.022 & 0.545 & 0.016 & 0.018 \\ 0.012 & 0.024 & 0.044 & 0.009 & 0.029 & 0.016 & 0.602 & 0.023 \\ 0.003 & 0.015 & 0.062 & 0.013 & 0.015 & 0.008 & 0.014 & 0.543 \end{bmatrix}$
K_P	Proportional coefficient matrix in PD loop $\begin{bmatrix} 10.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 12.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 11.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 10.4 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 14.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 12.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 10.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 11.3 \end{bmatrix}$
d	Robust adjustment coefficient $\begin{bmatrix} 0.032 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.028 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.010 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.009 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.008 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.008 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.009 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.009 \end{bmatrix}$

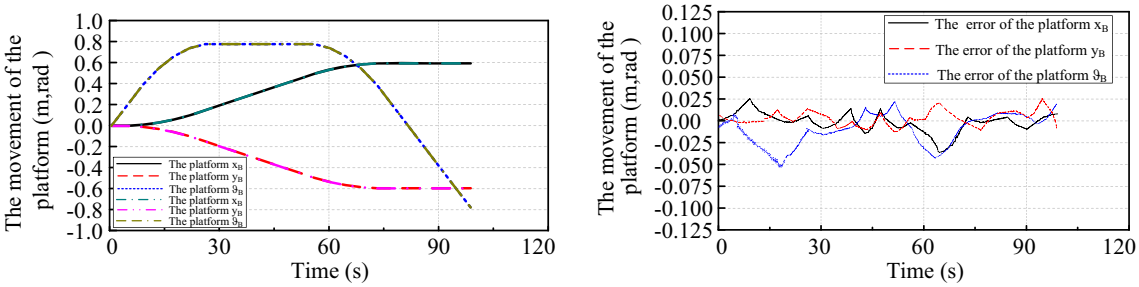

Fig. 15 Reference and real trajectory of the platform

$$\boldsymbol{V}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} -\mathrm{p}_3\dot{q}_2\sin q_2 & -\mathrm{p}_3(\dot{q}_1+\dot{q}_2)\sin q_2 \\ \mathrm{p}_3\dot{q}_1\sin q_2 & 0 \end{bmatrix},$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} \mathrm{p}_4 g\cos q_1 + \mathrm{p}_5 g\cos(q_1+q_2) \\ \mathrm{p}_5 g\cos(q_1+q_2) \end{bmatrix}, \quad \boldsymbol{F}(\dot{\boldsymbol{q}}) = 0.02\mathrm{sgn}(\dot{\boldsymbol{q}}), \quad \boldsymbol{\tau}_\mathrm{d} = [\,20\sin(t) \quad 20\sin(t)\,]^\mathrm{T}.$$

The reference positions for training of the two joints can be given as $\mathrm{q}_{1d} = 0.1\sin(t)$, $\mathrm{q}_{2d} = 0.1\sin(t)$. The reference positions for prediction of the two joints can be given as $\mathrm{q}_{1d} = 0.5\sin(t)$, $\mathrm{q}_{2d} = 0.5\cos(t)$. $\boldsymbol{p} = [p_1,\ p_2,\ p_3,\ p_4,\ p_5] = [\,2.9,\ 0.76,\ 0.87,\ 3.04,\ 0.87\,]$. The parameters of controller (73) are selected as: $\boldsymbol{K}_\mathrm{P} = \mathrm{diag}\{35.26, 38.45\}$, $\boldsymbol{K}_\mathrm{D} = \mathrm{diag}\{25.45,\ 20.15\}$, $\mathrm{b}_\varepsilon = 0.20$, $\mathrm{b}_\mathrm{d} = 0.10$, $\kappa = \mathrm{b}_w = \imath = 0.01$. In order to obtain the time series, using Eq. (74) to generate 7200 data as the input. We use the first 6000 data pairs as training data sets and the last 1200 data pairs to validate the model's predictive performance.

Figure 5 shows that the number of fuzzy rules of the algorithm is 12. The algorithm has good generalization ability at the same time. Figures 6, 7, 8 and 9 show the good performance of the algorithm.

5.2 Trajectory tracking for real MRR

At this stage, a real-life wheeled mobile manipulator is used for the tracking experiment. The robot is named RCAMC-1 (see Fig. 10) [43]. It consists of arm from SCHUNK LWA4/SDH, providing 6 DOF, a four-wheel mobile base with two active wheels (front), and two passive wheels (back). We have written all our software based on VC++ and made use of a variety of open-source packages including KDL (knowledge description language), ROBOOP (an object-oriented toolbox in C++ for robotics simulation), and MATLAB. The API (Automated Precision Inc.) T3 laser tracker system is used for the aided experiment (Fig. 11).

In this study, using the main control points as shown in Table 1, a smooth trajectory for mobile manipulator is proposed to be used with the open-door task. The smooth orientation trajectory is generated by interpolating key orientations with the spherical spline quaternion interpolation method, as shown in Fig. 12, and a Hermite cubic polynomial is used to connect these position points to create a rudimentary position trajectory, as shown in Fig. 13. The key joint positions are solved with the inverse kinematics algorithm according to discrete pose sampled with specific accuracy of the task need, and then smooth joint trajectories with stable start–stop motion and continuous jerk are obtained through interpolating key joint positions with B-spline, as shown in Figs. 11 and 14. Next, Eq. (73) is used as the trajectory tracking control law, to realize the planned trajectory tracking. This controller is realized using C++ programming with a cycle of 20 ms (Table 2).

The response of FNN–EKF-based controller is described under conditions that the dynamic model of the system is not exactly known. The input and output sample data generated from [44]. Figure 15 shows the comparison and errors of reference trajectory and real trajectory for wheeled platform in task space. Figure 16 shows the comparison and errors of reference trajectory and real trajectory for the manipulator in joint space.

6 Conclusion

The new method for mobile manipulator tracking the desired motion trajectory was developed using fuzzy neural network (FNN) and extended Kalman filter (EKF) approach and the robust control algorithm. The FNN is trained to generate a feedforward torque. For parameter tuning of the FNN, an online EKF methodology is derived to sequentially update both the output weights and centers, so that the tracking velocity and converge property can be guaranteed. When external disturbances and unknown system parameters changed, not only the connecting weights but also the centers are adjusted online. This training scheme will increase the learning capability of the FNN. By several simulations and experiments, it is obvious that the tracking performance was advantageous. The method can be extended to more complex robot configurations, such as tracks or legs, although a method is never the perfect solution. Even so, this paper contribution

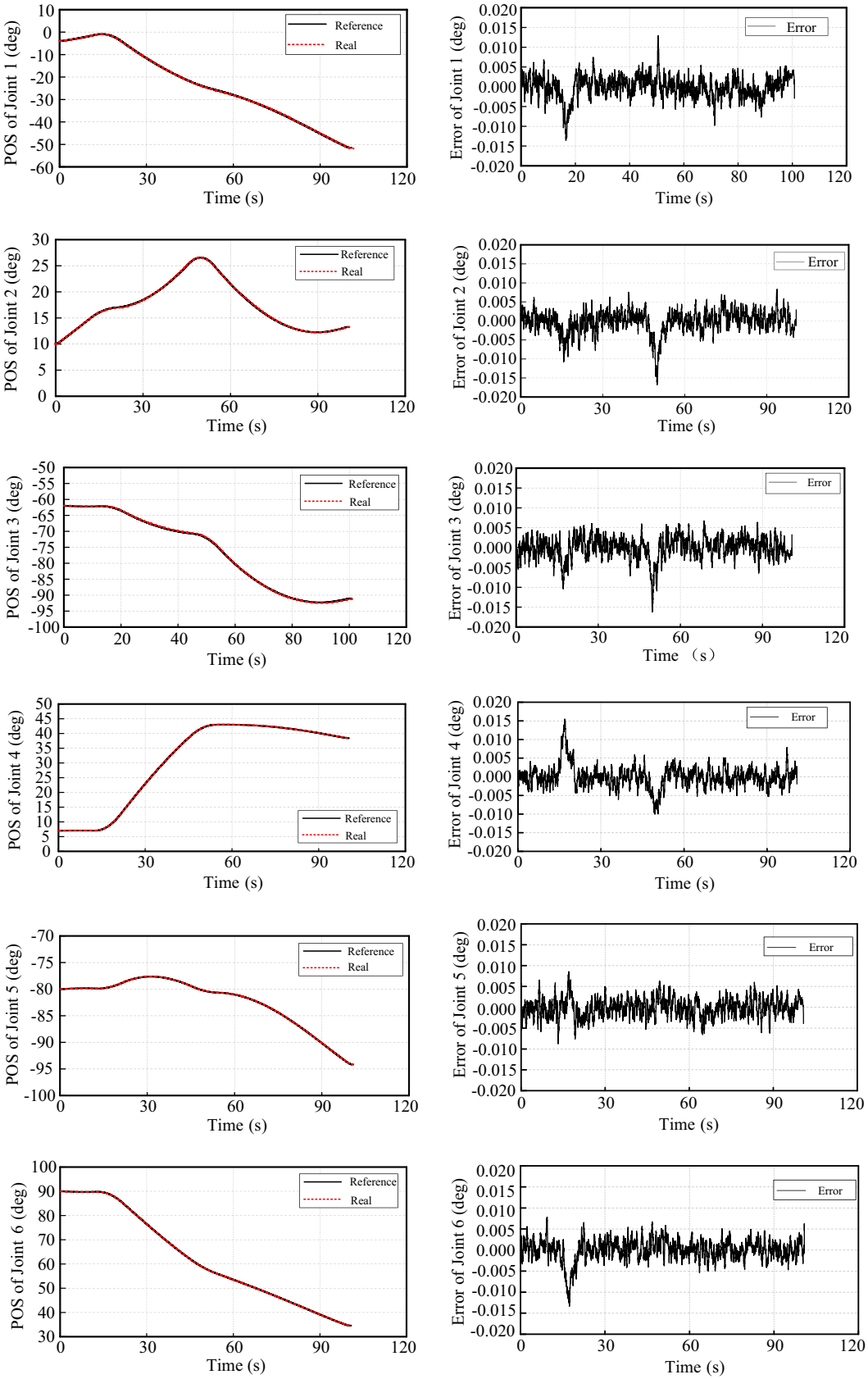

Fig. 16 Reference and real trajectory of the manipulator

focuses on the development of a new method for mobile manipulator tracking the desired motion trajectory that provides a better solution of the high-performance control. Experiments were conducted to contrasting the utility and validity of the proposed method. The method proposed in this study supplies another option for similar robots' process.

Acknowledgements This work was supported by the National Key Basic Research Development Plan Project (973) (2013CB035502), National Natural Science Foundation of China (Grant No. 61370033/51275106), Harbin Talent Program for Distinguished Young Scholars (NO. 2014RFYXJ001). Fundamental Research Funds for the Central Universities (Grant No. HIT.BRETIII.201411), Foundation of Chinese State Key Laboratory of Robotics and Systems (Grant No. SKLRS201401A01), Postdoctoral Youth Talent Foundation of Heilongjiang Province, China (Grant No. LBH-TZ0403), and "111" Project (B07018).

References

1. Atawnih A, Papageorgioua D (2016) Kinematic control of redundant robots with guaranteed joint limit avoidance. Robot Auton Syst 79:122–131
2. Li L, Gruver WA (2001) Kinematic control of redundant robots and the motion optimizability measure. IEEE Trans Syst Man Cybern B Cybern 31(1):155–160
3. Lin H, Chen C (2011) A hybrid control policy of robot arm motion for assistive robots. IEEE Int Proc Inf Autom 33(7):163–168
4. Masoud S, Masoud A (2000) Constrained motion control using vector potential fields. IEEE Trans Syst Man Cybern A 30(2):251–272
5. Falco P, Natale C (2011) On the stability of closed-loop inverse kinematics algorithms for redundant robots. IEEE Trans Robot 27(4):780–784
6. Bloch AM, Reyhanoglu M (1992) Control and stabilization of nonholonomic dynamic systems. IEEE Trans Autom Control 37(11):1746–1757
7. Samson C (1995) Control of chained systems: application to path following and time-varying point-stabilization of mobile robots. IEEE Trans Autom Control 40(1):64–77
8. Sordalen OJ, Egeland O (1995) Exponential stabilization of nonholonomic chained systems. IEEE Trans Autom Control 40(1):35–49
9. Wang ZP, Ge SS, Lee TH (2004) Robust adaptive neural network control of uncertain nonholonomic systems with strong nonlinear drifts. IEEE Trans Syst Man Cybern B Cybern 34(5):2048–2059
10. Dong WJ, Huo W, Tso SK, Xu WL (2000) Tracking control of uncertain dynamic nonholonomic system and its application to wheeled mobile robots. IEEE Trans Robot Autom 16(6):870–874
11. Dong WJ, Xu WL (2001) Adaptive tracking control of uncertain nonholonomic dynamic system. IEEE Trans Autom Control 46(3):450–454
12. Oya M, Su CY, Katoh R (2003) Robust adaptive motion/force tracking control of uncertain nonholonomic mechanical systems. IEEE Trans Robot Autom 19(1):175–181
13. Shojaei K, Shahri AM (2012) Adaptive robust time varying control of uncertain nonholonomic robotic systems. IET Control Theory Appl 6(1):90–102
14. Li Z, Ge SS, Adams M, Wijesoma WS (2008) Robust adaptive control of uncertain force/motion constrained nonholonomic mobile manipulators. Automatica 44(3):776–784
15. Li Z, Yang YP, Li JX (2010) Adaptive motion/force control of mobile under-actuated manipulators with dynamics uncertainties by dynamic coupling and output feedback. IEEE Trans Control Syst Technol 5(18):1068–1079
16. Li Z, Ge SS, Adams M, Wijesoma WS (2008) Adaptive robust output-feedback motion/force control of electrically driven nonholonomic mobile manipulators. IEEE Trans Control Syst Technol 16(6):1308–1315
17. Hua CC, Yang Y, Guan X (2013) Neural network-based adaptive position tracking control for bilateral teleoperation under constant time delay. Neurocomputing 113:204–212
18. Lin CM, Hsu CF (2005) Recurrent neural network based adaptive backstepping control for induction servomotors. IEEE Trans Ind Electron 52(6):1677–1684
19. Lin FJ, Wai RJ (2003) Robust recurrent fuzzy neural network control for linear synchronous motor drive system. Neurocomputing 50:365–390
20. Das T, Kar IN, Chaudhury S (2006) Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics. Neurocomputing 69:2140–2151
21. Fierro R, Lewis FL (1998) Control of a nonholonomic mobile robot using neural networks. IEEE Trans Neural Netw 9(4):589–600
22. Xu D, Zhao D, Yi J, Tan X (2009) Trajectory tracking control of omnidirectional wheeled mobile manipulators: robust neural network-based sliding mode approach. IEEE Trans Syst Man Cybern B Cybern 39(3):788–799
23. de Sousa C, Hemerly EM Jr., Galvao RKH (2002) Adaptive control for mobile robot using wavelet networks. IEEE Trans Syst Man Cybern B Cybern 32(4):493–504
24. Park BS, Yoo SJ, Choi YH (2009) Adaptive neural sliding mode control of nonholonomic wheeled mobile robots with model uncertainty. IEEE Trans Control Syst Technol 17(1):207–214
25. Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355
26. Bezdek J, Keller J, Krishnapuram R, Kuncheva L, Pal H (1999) Will the real Iris data please stand up? IEEE Trans Fuzzy Syst 7:368–369
27. Moody J, Darken C (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1:289–303
28. Chen S, Cowan C, Grant P (1991) Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw 2:302–309
29. Chen S, Wu Y, Luk B (1999) Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Trans Neural Netw 10:1239–1243
30. Duro R, Reyes J (1999) Discrete-time backpropagation for training synaptic delay-based artificial neural networks. IEEE Trans Neural Netw 10:779–789
31. Shah S, Palmieri F, Datum M (1992) Optimal filtering algorithms for fast learning in feedforward neural networks. Neural Netw 5:779–787
32. Sum J, Leung C, Young G, Kan W (1999) On the Kalman Filtering method in neural network training and pruning. IEEE Trans Neural Netw 10:161–166
33. Zhang Y, Li X (1999) A fast U-D factorization-based learning algorithm with applications to nonlinear system modeling and identification. IEEE Trans Neural Netw 10:930–938
34. Obradovic D (1996) On-line training of recurrent neural networks with continuous topology adaptation. IEEE Trans Neural Netw 7:222–228

35. Puskorius G, Feldkamp L (1994) Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. IEEE Trans Neural Netw 5:279–297
36. Birgmeier M (1995) A fully Kalman-trained radial basis function network for nonlinear speech modeling. IEEE international conference on neural networks, Perth, Western Australia, pp 259–264
37. Nabney IT (1996) Practical methods of tracking of non-stationary time series applied to real world problems. In: Rogers SK, Ruck DW (eds) AeroSense'96: applications and science of artificial neural networks II, SPIE Proc. No. 2760, pp 152–163
38. Connor J, Martin R, Atlas L (1994) Recurrent neural networks and robust time series prediction. IEEE Trans Neural Netw 5(2):240–254
39. Puskorious G, Feldkamp L (1994) Neural control of nonlinear dynamic systems with Kalman filter trained recurrent networks. IEEE Trans Neural Netw 5(2):279–297
40. Nelson AT, Wan EA (1997) Neural speech enhancement using dual extended Kalman filtering, ICNN, pp 2171–2175
41. Ding L, Gao H, Xia K, Liu Z, Tao J, Liu Y (2012) Adaptive sliding mode control of mobile manipulators with Markovian switching joints. J Appl Math 10(3):812–836
42. Wu SQ, Er MJ, Gao Y (2001) A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. IEEE Trans Fuzzy Syst 9(4):578–594
43. Xia K, Ding L, Gao H, Deng Z, Liu G, Wu Y (2016) Switch control for operating constrained mechanisms using a rescuing mobile manipulator with multiple working modes. In: Advanced Robotics and Mechatronics (ICARM), International Conference on IEEE, pp 139–146
44. Gao H, Xia K, Ding L, Deng Z, Liu Z, Liu G (2015) Optimized control for longitudinal slip ratio with reduced energy consumption. Acta Astronaut (115):1–17
45. Wu Q, Rete W (2000) Dynamic fuzzy neural networks-a novel approach to function approximation. IEEE Trans Syst Man Part B-Cybern 30(2):358–364
46. Chen S, Cowan CFN, Grant PM (1991) Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw 2(2):302–309