Neurocomputing 338 (2019) 249–261

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Robot trajectory tracking control using learning from demonstration method

Sheng Xu[a], Yongsheng Ou[b,*], Jianghua Duan[a,d], Xinyu Wu[c,e], Wei Feng[a], Ming Liu[f]

[a] Center for Intelligent and Biomimetic Systems, Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS), Shenzhen 518055, PR China
[b] Guangdong Provincial Key Laboratory of Robotics and Intelligent System, SIAT, CAS, Shenzhen 518055, PR China
[c] Key Laboratory of Human-Machine-Intelligence Synergic Systems, SIAT, CAS, Shenzhen 518055, PR China
[d] Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, PR China
[e] Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, SIAT, CAS, Shenzhen 518055, PR China
[f] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 3 April 2018
Revised 3 December 2018
Accepted 16 January 2019
Available online 24 January 2019

Communicated by Dr W Gao

Keywords:
Robot trajectory tracking
Learning from demonstration (LFD)
Extreme learning machines (ELM)
State errors
Stability analysis

ABSTRACT

This paper addresses robot trajectory tracking problem by using the learning from demonstration (LFD) method. Firstly, the trajectory tracking problem is formulated and the related previous works are introduced. Secondly, a trajectory tracking control policy using a three-layer neural network method, i.e., extreme learning machines (ELM), is proposed to minimize the real-time position and velocity errors. In the proposed method, the control algorithms are learnt from demonstrations directly such that the parameter adjusting problem in the traditional model-based methods is avoided. Besides, the trained controller has generalization ability to unseen situations which can be used to track different desired trajectories without any extra re-training. Thirdly, the stability analysis of the proposed control algorithm is provided and the corresponding parameter constraints are derived. Finally, the effectiveness and the generalization ability of the proposed control algorithms are demonstrated and discussed with simulation and experimental examples.

1. Introduction

Recently, as the development of robotic technology, the intelligent robots have been widely used in both military and civilian areas such as target search, attack and rescue missions, industrial applications and home services [1–3]. In both the industrial and service applications such as cutting, welding, assembling, human guiding and assistance, the robots are required to follow a desired trajectory. Especially in the industrial area, the robot arm is required to finish the trajectory tracking mission with high position and velocity accuracy. Besides, as the quality requirement of the cutting and welding, the trajectory tracking should be finished under a given velocity. Therefore, the trajectory tracking not only requires a geometric path-following but also needs the speed-assignment at different waypoints [4,5]. In addition, the desired trajectory should be pre-specified or estimated by some accurate measurement sensors [6]. Thus, the trajectory tracking problem contains two steps, i.e., reference trajectory estimation and trajectory tracking control. In this paper, we only focus on trajectory tracking control and assume the desired trajectory is clearly known.

Many previous works of the close-loop trajectory tracking control strategies have been developed [7–11]. In these literatures, the reference trajectory is known with negligible error and the real-time position/velocity is measured accurately. Then, the position/velocity error-based close-loop servo system is built to steer the robot tracking the reference trace. In the classical close-loop servo system, the robot (or called “plant”) nonlinear dynamic modeling is necessary. With the accurate plant model, the output of an appropriate controller can be transformed to the desired robot state changes. In other words, two steps exists in the classical close-loop control methods, i.e., system dynamic modeling and control policy design. In [12], a weight self-adaptive controller using the neural network method was developed for a robot position servo system. The weights will change automatically according to the changes of the robot arm nonlinear dynamical model. Focused on the robot modeling parametric uncertainties, the authors of [13] developed two different fuzzy logic control algorithms as the compensators for the structured/unstructured uncertainties, e.g., frictions, external disturbances and payload changes, of the robot manipulator. The difference between these two fuzzy controllers is that one considered the pre-known uncer-

* Corresponding author.
E-mail address: ys.ou@siat.ac.cn (Y. Ou).

https://doi.org/10.1016/j.neucom.2019.01.052
0925-2312/

tainties and the properties of robot dynamics whereas the other one did not. In [14], a modified model predictive path tracking controller was designed and the trade-off between robustness and accuracy was solved by selecting a so-called model predictive control cost function. In [7], the extreme learning machines (ELM), a three-layer neural network method with randomly selected input layer weight and hidden layer bias [15], was applied to a nonlinear trajectory tracking problem. Meanwhile, a sliding mode controller was also designed and utilized with the ELM-based close-loop controller to improve the robustness. In [8], the ELM method was employed to identify the uncertainties of the nonlinear robot dynamics and another robust controller was developed to compensate these uncertainties. A stable trajectory tracking controller based on Gaussian radial basis function static neural network method was proposed in [16], and this controller contains two stages using a switch logic to solve the problem caused by system model uncertainties. Furthermore, the detailed robot kinematics were analyzed using a new factorization method of the Coriolis/centripetal matrix for the control algorithm designing. The authors of [10] proposed a predictive control algorithm to solve the constrained path tracking problems for the industrial robot arm. The accurate robot dynamic model was known as the basic condition before the controller designing.

To summary, for these previous works, the robot dynamics are very important to the control policy designing and the performance would decrease once the actual plant model deviates seriously. To avoid these problem, some model identification strategies are required before the controller designing [17,18] and system uncertainties compensation algorithms are also necessary. In [19], an adaptive evolutionary neural controller was developed to solve the external disturbances and dynamic variations for a serial pneumatic artificial muscle (PAM) robot. Furthermore, an adaptive online displacement controller which combined the feed-forward neural networks and the PID control strategies, is proposed for the shape memory alloy (SMA) actuator control in [20]. The methods in these two works solved the plant modelling problem effectively. Another characteristic of these previous studies is that the parameters of the controller should be tuned appropriately based on the plant dynamics. While, to determine these parameters is a tough task especially when the users have limited knowledge about the controller. Consequently, the problem of how to make the robot trajectory tracking controller be simple to implement has not been fully addressed. To simplify the complex process of parameter adjusting in the control policy design, the learning from demonstration (LFD) method, has been proposed and utilized to the robotic areas. In the LFD methods, demonstrations can be provided by an advanced controller or an experienced human worker with no knowledge about control theory, and the controller adjusts its parameters by itself from these demos using machine learning algorithms. Based on the statistical knowledge, different kinds of LFD methods were developed such as using the artificial neural networks, support vector machines (SVM), Gaussian mixture models regression and etc. [21–24]. These learning-based methods can determine the appropriate parameters by statistical knowledge from a large number of repeated demonstrations. The most important advantage of these LFD methods is that the users only need to implement the desired motions repeatedly and the control parameters will be calculated automatically. Consequence, neither the professional knowledge about the controller designing nor the controller parameter adjusting skills is required. Compared with the classical control strategies in the last paragraph, the main advantage of the LFD method is the simplified parameter adjustment. While the kinematic dynamics modeling is still required which can be solved by the LFD strategies as well.

In the tutorial paper [25], the author concluded different Gaussian mixture models (GMM) methods and the extension algorithms of the GMM, which are applied for the service robots to adapt diverse movements in the unconstrained environments. The comprehensive Matlab simulation codings about the GMM and their extensions are provided. Furthermore, some control algorithms using the LFD method have been proposed for the point-to-point tracking problem [26,27]. In [26], the robotic point-to-point motion with preferred control policies was modeled as a nonlinear dynamical system (DS) and a control policy was developed to ensure the states converging to the target point with global asymptotic stability, and the proposed control policy is modeled by the GMM regression method. Furthermore, the parameters of the control policy is determined using a novel method which is called Stable Estimator of Dynamical Systems (SEDS) with the consideration of stability constraints. In addition, the proposed method was also extended to a trajectory tracking problem by an experiment with self-interacting motions. The DS method was also applied by Khansari-Zadeh and Billard [28] to describe the relationship between the desired point-to-point state changes and the control policy, and thus the detailed robot kinematics were unnecessary to be considered separately. By focusing on the control Lyapunov function designing, the global asymptotic stability of the nonlinear DS was guaranteed. In [29], the reach and grasp problem, i.e., a point-to-point motion, became more complex with the consideration of the coupling problem between the hand movements and finger motions. A controller considering the coupled dynamical system was developed and guaranteed the robot fast adaptation for perturbations with zero delay. The authors of [27] also focused on the point-to-point reaching problem in [26], but they proposed a fast, stable and simple LFD method based on the extreme learning machines. Thus, the time-consuming drawback of the SEDS is avoided by using the ELM-based method. To extend the learning-based point-to-point control method into trajectory tracking problem, the reference can be assumed as a set of point-to-point motions and we just need to steer the robot arrive to all the points one-by-one sequentially. However, the tracking performance is not smooth because the zero-velocity constraint was not considered nor required in the previous works [26–28]. In [30], a potential function construction strategy based on learning method was developed for a trajectory following problem. The reference trajectory was perfectly followed by their proposed method and the stability was simple to satisfied. However, if the desired trace is changed, all the controller parameters should be re-determined and the tracking velocity is uncontrollable. Besides, the differences between trajectory following and trajectory tracking problem were introduced in [5], and we focus on the trajectory tracking problem in this paper. Consequence, the previous works cannot be applied directly to provide satisfied performance in the trajectory tracking problem.

Compared with the previous works, two key contributions are made in this paper. Firstly, a learning-based trajectory tracking control policy is derived for different applications using a simple three-layer neural network method, i.e., ELM algorithm. The control algorithm is learnt from demonstrations directly such that the parameter adjusting problem in the existing modeling-based methods [13,18] is avoided. Therefore, the proposed method is simple to implement with the ability of generalization. The trajectory tracking controller is learnt by only using the position errors of the demonstrations. Besides, as the proposed method is to minimize the errors, the control performance is not affected by the geometries of the desired trajectory, e.g. self-interacting trajectory. Secondly, the trajectory tracking problem is solved by the proposed learning-based method which can provide smooth and accurate performance, and it is different from the existing point-to-point control strategies [26,27]. More specifically, compared with [27], in this work the point-to-point problem is extended into accurate trajectory tracking problem. In the previous method, the velocity was not considered at all and the position state was the controller

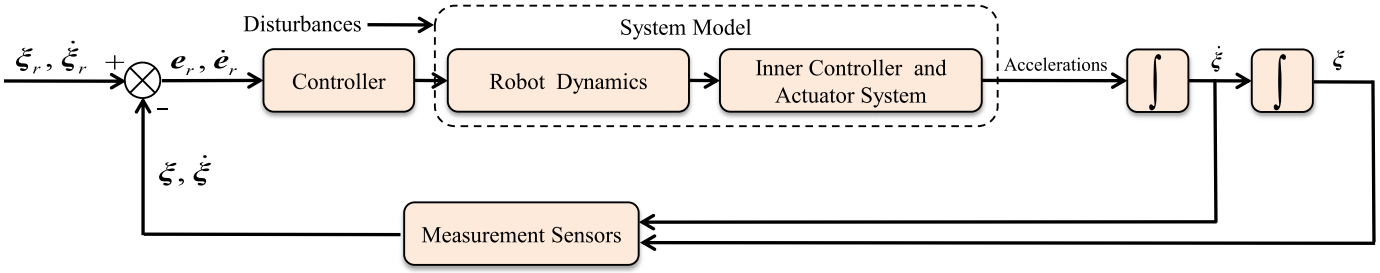

Fig. 1. Diagram of the classical close-loop trajectory tracking system for a robot control problem.

input. In this paper, the velocity is carefully considered both in the control policy design and stability analysis. In addition, the global asymptotic stability analysis and the convergence of position and velocity errors are provided. The corresponding parameter constraints are derived. According to the simulation results, the effectiveness and the characteristics of the proposed method are demonstrated and discussed.

The rest of this paper is structured as follows. Section 2 is the problem formulation. The trajectory tracking control policy based on learning from demonstration method is proposed in Section 3. Section 4 provides the stability analysis of the proposed control algorithm. Simulation and experimental studies are presented in Section 5. Section 6 draws the conclusion and discussion.

2. Problem formulation

In this paper, we consider using a robot, e.g. end effector of industrial robot arm, wheeled mobile robot, quad-rotor flying robot, to imitate a real-time geometric trajectory. The reference trajectory, e.g., in the 3-dimensional space, takes the form

$$\boldsymbol{\xi}_r = [x_r(t), y_r(t), z_r(t)]^T \tag{1}$$

where ^{T} denotes the matrix transposition and the reference state $\boldsymbol{\xi}_r$ is a variable respect to the time t and $\dot{\boldsymbol{\xi}}_r$ is the reference velocity at $\boldsymbol{\xi}_r$. Define the actual position of the robot as

$$\boldsymbol{\xi} = [x(t), y(t), z(t)]^T \tag{2}$$

and the velocity as $\dot{\boldsymbol{\xi}}$. To make the the robot follow the reference trajectory smoothly and perfectly, both the position and velocity need to equal the reference values. Besides, in some practical applications such as cutting and welding, the velocities at the waypoints along the trajectory may impact the production quality significantly. Therefore, it is necessary to realize the trajectory tracking with the reference velocities. In addition, for a pre-given and bounded trajectory tracking problem (or called reappear the reference trajectory) which does not have tracking time requirement, only the position state need to exactly equal the reference values sequentially. However, without considering the velocity will meet the non-smooth problem as appeared in the point-to-point methods. Therefore, the smooth velocity changes are also required but they do not need to precisely equal the reference values and can be defined as the users' requirements. Furthermore, the proposed control policy can also be applied for this degenerated problem. In order to simplify the derivation process in the later sections of this paper, the position, velocity and acceleration errors take the forms

$$\boldsymbol{e} = \boldsymbol{\xi} - \boldsymbol{\xi}_r \quad \dot{\boldsymbol{e}} = \dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_r \quad \ddot{\boldsymbol{e}} = \ddot{\boldsymbol{\xi}} - \ddot{\boldsymbol{\xi}}_r. \tag{3}$$

Therefore, the objective of the reference tracking problem is to design an appropriate control law to minimize the errors at every time instant. For simplification reason, the time index t are ignored in $\boldsymbol{\xi}_r$, $\boldsymbol{\xi}$, $\boldsymbol{e}$ and their derivatives. The diagram of the classical close-loop trajectory tracking system is depicted in Fig. 1.

In order to make the problem analyzable and simple to analyze, two assumptions are proposed.

Assumption 1. The given reference trajectory is reachable which has considered the position constraint of the robot. For example, a 4-DOF robot arm cannot reach all the positions around the robot base. For another example, if the maximum length of the robot arm is 2 m, it can never reach a 10 m away target position without moving its base.

Assumption 2. The robot (or robot arm end effector) is considered as a mass point such that the torques and attitude of the robot are ignored. More specifically, we only focus on trajectory tracking and trying to expect the mass point matching the reference trajectory.

3. Trajectory tracking control policy

In the previous works, the machine learning methods are applied to build and fit the accurate plant model in order to design a specific control law using variant control strategies. In this paper, we develop a control algorithm based on the LFD method and the DS strategy. Thus, in the proposed method, neither the robot dynamic model information nor the separated control law designing is required. In the proposed method we first make several manually demonstrations started from different original positions for the robot to follow up a reference trajectory with the ideal velocity changes. Also, the robot position/velocity information and the errors are recorded. Second, by using the machine learning methods, the control law can be obtained which should be a function respect to the states or state errors. Third, the robot starts from any position and reproduces the trajectory tracking using the trained control policy.

We extend the point-to-point method in [27] into a trajectory tracking problem. The desired trajectory can be perfectly tracked by controlling the real-time robot velocity. Furthermore, we will proof that both the real-time position and velocity states can converge to the reference trajectory using the proposed method.

The control policy is modeled with a first-order DS [27,28] focusing on the state error in the Cartesian coordinate which takes

$$\dot{\boldsymbol{e}} = f(\boldsymbol{e}) \tag{4}$$

Combine (4) and the robot transformation function $\mathcal{S}(\cdot)$, the control policy of the actuator system, defined as $\boldsymbol{u}$ takes

$$\boldsymbol{u} = \mathcal{S}^{-1}(f(\boldsymbol{e})). \tag{5}$$

Note that $\mathcal{S}(\cdot)$ is a function which firstly computes the corrected positions and secondly transforms the corrected positions into the corrected actuator commands. More specifically, once the users acquire the state error and $f(\boldsymbol{e})$, for the robot arm or the mobile robot, the detailed actuator behavior is calculated through $\mathcal{S}(\cdot)$ automatically. In this paper, the identification of $\mathcal{S}(\cdot)$ is not the problem we concerned and many previous methods have been developed already [31]. Thus, $\mathcal{S}(\cdot)$ is assumed as a mapping function

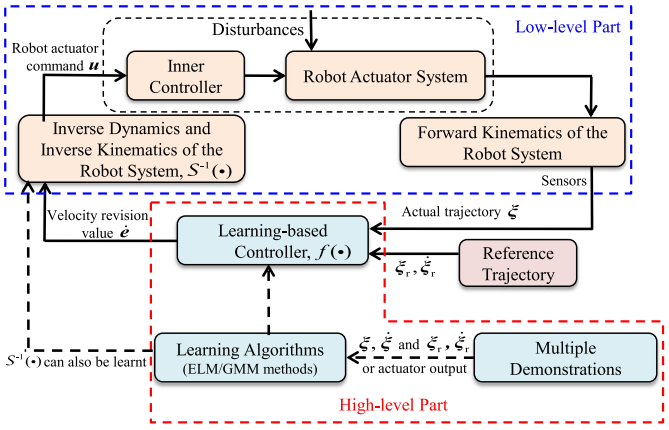

Fig. 2. Diagram of the learning-based trajectory tracking control system.

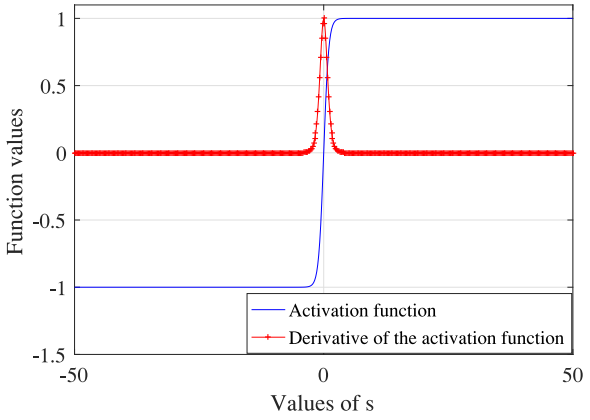

Fig. 3. Activation function $g(s)$ and its derivative $g'(s)$.

which can be different depended on the types of the robots and it will not impact the learning-based control policy design.

As introduced in the literature, different LFD methods such as the artificial neural network (including the ELM), GMM and SVM, can all be utilized to develop the control policy, fitting the accurate expression of $f(\cdot)$. Because the ELM method has the advantages of fast training speed and simple to implement [27], we choose it to train the controller. The diagram of the this learning-based trajectory tracking control system is shown in Fig. 2. More specifically, the designed control system consists of two parts, i.e., high-level and low-level parts. In the high-level part, the controller is trained by multiple desired demonstrations off-line firstly. Then, the revised end-effector command will be calculated by the trained controller with the real-time actual and reference states. Next, the robot inner slave system will transform and execute the revised end-effector command to the corresponding movements of all the motors. Finally, the real-time robot end-effector states are measured by the installed sensors and the close-loop control system is built.

Based on the ELM [22] and the DS [28] methods, the control policy of the robot can be written as

$$\boldsymbol{u} = \mathcal{S}^{-1}(\dot{\boldsymbol{e}}) = \mathcal{S}^{-1}\left(\sum_{i=1}^{N} \boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^T \boldsymbol{e} + b_i\right)\right) \tag{6}$$

where N is the number of the hidden nodes, $\boldsymbol{u}$ is a variable dimensional vector which is determined by the actual actuator mechanical structure. $g(\cdot)$ is the activation function,

$$\boldsymbol{b} = [b_1, b_2, \ldots, b_N]^T_{N\times 1} \tag{7a}$$

$$\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N]_{d\times N} \tag{7b}$$

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_N]_{d\times N} \tag{7c}$$

are the hidden layer bias, the input weight and the output weight, respectively. With the control policy, the final acquired control command can correct the derived position. Note that $(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ are the column vectors and d denotes the state dimension, e.g., $d = 3$ for the 3-dimensional problem. Now, the objective is to determine the parameters $\boldsymbol{b}$, $\boldsymbol{W}$ and $\boldsymbol{\beta}$. Thus, a clear mathematical relationship between the inputs $\boldsymbol{e}$ and the output $\boldsymbol{u}$ will be built. These parameters will be determined by the training process using multiple demonstrations. Note that the proposed control policy is based on the state errors which is very different from the state-based method in [27] for the trajectory tracking. Therefore, the required demonstration data is also different.

Based on [22], $\boldsymbol{b}$ and $\boldsymbol{W}$ in (6) are initialized with invariable random constants. The number of the total data points in the demonstrations is M. Then, the output weight $\boldsymbol{\beta}$ can be calculated by solving the tracking least-square problem

$$\min_{\boldsymbol{\beta}} \left\| \boldsymbol{D}\boldsymbol{\beta}^T - \mathcal{S}(\boldsymbol{u}_o) \right\| \tag{8}$$

where

$$\boldsymbol{D} = \begin{bmatrix} g\left(\boldsymbol{w}_1^T \boldsymbol{e}_1 + b_1\right) & \cdots & g\left(\boldsymbol{w}_N^T \boldsymbol{e}_1 + b_N\right) \\ \vdots & \ddots & \vdots \\ g\left(\boldsymbol{w}_1^T \boldsymbol{e}_M + b_1\right) & \cdots & g\left(\boldsymbol{w}_N^T \boldsymbol{e}_M + b_N\right) \end{bmatrix}_{M\times N} \tag{9}$$

is the hidden layer output matrix. The outputs of the M data points are denoted by $\boldsymbol{u}_o = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M]^T$. As $\boldsymbol{W}$ and $\boldsymbol{b}$ are constants, $\boldsymbol{D}$ is also fixed. Therefore, the solution of the minimization problem in (8) becomes

$$\hat{\boldsymbol{\beta}} = \boldsymbol{D}^{+} \mathcal{S}(\boldsymbol{u}_o) \tag{10}$$

where $\boldsymbol{D}^{+}$ is the Moore–Penrose generalized inverse of $\boldsymbol{D}$ [22]. For the continuous and continuously differentiable activation function $g(\cdot)$, it should satisfy the tracking conditions to guarantee the effectiveness of the ELM

$$g(0) = 0 \tag{11a}$$

$$\dot{g}(s) > 0, \quad s \neq 0. \tag{11b}$$

Consequently, we choose

$$g(s) = \frac{2}{1+\exp(-2s)} - 1 \quad \text{and} \quad g'(s) = \frac{4\exp(-2s)}{[1+\exp(-2s)]^2} \tag{12}$$

as the activation function and its derivative. Fig. 3 shows the functions of $g(s)$ and $g'(s)$. Note that different activation function can be chosen, e.g., sigmoid function, tanh function and smooth-step function.

Based on the data ($\boldsymbol{e}_1$, ..., $\boldsymbol{e}_M$, $\boldsymbol{u}_1$, ..., $\boldsymbol{u}_M$) of the M demonstration data points, (10) and (12), the parameters in (6) can be regressed and determined. Thus, for a new input $\boldsymbol{e}$, the corresponding output $\boldsymbol{u}$ is acquired which can be utilized for robot state correction.

4. Stability analysis

In order to guarantee the trajectory tracking system stable, the Lyapunov stability method is applied. More specifically, to realize the trajectory tracking stable and accurate, both the position and velocity errors $\boldsymbol{c} = [\boldsymbol{e}^T, \dot{\boldsymbol{e}}^T]^T$ are required to converge to zeros. In the Lyapunov stability theorem, the system states $\boldsymbol{c} = [\boldsymbol{e}^T, \dot{\boldsymbol{e}}^T]^T$ are

globally asymptotic stable at the point $\boldsymbol{c}^{\star}=\mathbf{0}$ if a continuous and continuously differentiable Lyapunov-candidate-function (LCF) $V(\boldsymbol{c})$ satisfies [32]

$$V(\boldsymbol{c})>0,\quad \forall \boldsymbol{c}\in\mathbb{R}^{2d}\quad \text{and}\quad \forall \boldsymbol{c}\neq\boldsymbol{c}^{\star} \tag{13a}$$

$$\dot{V}(\boldsymbol{c})<0,\quad \forall \boldsymbol{c}\in\mathbb{R}^{2d}\quad \text{and}\quad \forall \boldsymbol{c}\neq\boldsymbol{c}^{\star} \tag{13b}$$

$$V(\boldsymbol{c}^{\star})=0 \tag{13c}$$

$$\lim_{||\boldsymbol{c}||\to\infty} V(\boldsymbol{c})=\infty. \tag{13d}$$

From (13d), the state may start from a randomly unbounded point and if (13a) and (13b) are satisfied, the state will definitely converge to the global stable point (13c).

To guarantee the system globally asymptotically stable, the requirements in (13) should be always satisfied when the state errors start with any values. Consequence, the trajectory tracking system with any state errors will always converge to the desired trace.

Theorem 1. *When the parameters in the proposed control policy (6) satisfy,* $\forall i\in 1,2,\ldots,N$

$$\boldsymbol{\beta}_i\boldsymbol{w}_i^T\prec\mathbf{0} \tag{14a}$$

$$b_i=0, \tag{14b}$$

where $\prec\mathbf{0}$ *denotes the negative definiteness of a matrix, the trajectory tracking system is globally asymptotically stable (both the position and velocity errors can converge to zeros).*

Proof. We design the LCF $V(\boldsymbol{c})$ including both the position and velocity errors as

$$V(\boldsymbol{c})=\frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}+\frac{1}{2}\dot{\boldsymbol{e}}^T\dot{\boldsymbol{e}}. \tag{15}$$

Thus, $V(\boldsymbol{c})$ obviously satisfies (13a), (13c) and (13d). Consequently, we have

$$\dot{V}(\boldsymbol{c})=\boldsymbol{e}^T\dot{\boldsymbol{e}}+\dot{\boldsymbol{e}}^T\ddot{\boldsymbol{e}}. \tag{16}$$

Eq. (16) is a function including three variables, i.e., $\boldsymbol{e}$, $\dot{\boldsymbol{e}}$ and the unknown acceleration error $\ddot{\boldsymbol{e}}$. Combined with (6), the derivative of $\dot{\boldsymbol{e}}$ is computed as

$$\begin{aligned}\ddot{\boldsymbol{e}}&=\frac{\mathrm{d}\left[\sum_{i=1}^{N}\boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]}{\mathrm{d}t}\\&=\sum_{i=1}^{N}\boldsymbol{\beta}_i\frac{\mathrm{d}\left[g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]}{\mathrm{d}\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)}\boldsymbol{w}_i^T\dot{\boldsymbol{e}}.\end{aligned} \tag{17}$$

Then, all the unknown terms in (16) can be eliminated by replacing $\ddot{\boldsymbol{e}}$ by (17) and then (16) becomes

$$\dot{V}(\boldsymbol{c})=\boldsymbol{e}^T\dot{\boldsymbol{e}}+\dot{\boldsymbol{e}}^T\sum_{i=1}^{N}\boldsymbol{\beta}_i\frac{\mathrm{d}\left[g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]}{\mathrm{d}\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)}\boldsymbol{w}_i^T\dot{\boldsymbol{e}}. \tag{18}$$

Substituting the expression of $\dot{\boldsymbol{e}}$ from (6) yields

$$\dot{V}(\boldsymbol{c})=\boldsymbol{e}^T\sum_{i=1}^{N}\boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)+\dot{\boldsymbol{e}}^T\sum_{i=1}^{N}\boldsymbol{\beta}_i\frac{\mathrm{d}\left[g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]}{\mathrm{d}\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)}\boldsymbol{w}_i^T\dot{\boldsymbol{e}}. \tag{19}$$

For simplification reason, we can rewrite $\frac{\mathrm{d}\left[g\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]}{\mathrm{d}\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)}=g'\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)$ and it is the first-order derivative of the activation function which is always be positive. By using the mean-value theorem [33], (19) becomes

$$\dot{V}(\boldsymbol{c})=\boldsymbol{e}^T\left\{\sum_{i=1}^{N}\boldsymbol{\beta}_i\left[g(0)+g'(s_i)\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\right]\right\}+\dot{\boldsymbol{e}}^T\sum_{i=1}^{N}\boldsymbol{\beta}_i g'\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\boldsymbol{w}_i^T\dot{\boldsymbol{e}} \tag{20}$$

where $g'(s_i)$ is the first-order derivative of $g(s_i)$, and $s_i\in(0,\boldsymbol{w}_i^T\boldsymbol{e}+b_i)$ or $s_i\in(\boldsymbol{w}_i^T\boldsymbol{e}+b_i,0)$. As defined in (11 a), $g(0)=0$, (20) becomes

$$\dot{V}(\boldsymbol{c})=\boldsymbol{e}^T\sum_{i=1}^{N}\boldsymbol{\beta}_i g'(s_i)\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)+\dot{\boldsymbol{e}}^T\sum_{i=1}^{N}g'\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\boldsymbol{\beta}_i\boldsymbol{w}_i^T\dot{\boldsymbol{e}} \tag{21}$$

and equation (21) can be rewritten as

$$\dot{V}(\boldsymbol{c})=\underbrace{\boldsymbol{e}^T\sum_{i=1}^{N}g'(s_i)\boldsymbol{\beta}_i\boldsymbol{w}_i^T\boldsymbol{e}}_{①}+\underbrace{\boldsymbol{e}^T\sum_{i=1}^{N}g'(s_i)\boldsymbol{\beta}_i b_i}_{②}+\underbrace{\dot{\boldsymbol{e}}^T\sum_{i=1}^{N}g'\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)\boldsymbol{\beta}_i\boldsymbol{w}_i^T\dot{\boldsymbol{e}}}_{③}. \tag{22}$$

Furthermore, both the derivatives, i.e., $g'(s_i)$ and $g'\left(\boldsymbol{w}_i^T\boldsymbol{e}+b_i\right)$, of the activation function are always positive.

Combined with the conditions in (14) of the proposed Theorem, $\forall i\in 1,\ 2,\ \ldots,\ N$,

$$\boldsymbol{\beta}_i\boldsymbol{w}_i^T\prec\mathbf{0} \tag{23a}$$

$$b_i=0, \tag{23b}$$

$\dot{V}(\boldsymbol{c})$ is guaranteed be negative definiteness with any values of $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ (except $\boldsymbol{e}=\mathbf{0}$, $\dot{\boldsymbol{e}}=\mathbf{0}$), i.e., ① $\prec 0$, ② $=0$ and ③ $\prec 0$ are satisfied. In other words, $\dot{V}(\boldsymbol{c})$ can be guaranteed negative semi-definiteness and $\dot{V}(\boldsymbol{c})=0$ only when $\boldsymbol{e}=\mathbf{0}$, $\dot{\boldsymbol{e}}=\mathbf{0}$. Therefore, the constraints in Theorem can guarantee the globally asymptotic stability. □

To conclude, the proposed control policy designing with global asymptotic stability is a constraint-optimization problem that

$$\min_{\boldsymbol{\beta}}\left\|\boldsymbol{D}\boldsymbol{\beta}^T-\mathcal{S}(\boldsymbol{u}_o)\right\| \tag{24}$$

subject to, $\forall i\in 1,\ 2,\ \ldots,\ N$,

$$\boldsymbol{\beta}_i\boldsymbol{w}_i^T\prec\mathbf{0} \tag{25a}$$

$$b_i=0. \tag{25b}$$

As $\boldsymbol{w}_i^T$ and b_i are randomly selected, the solution of the optimization problem with the constraints in (25) is available. Last but

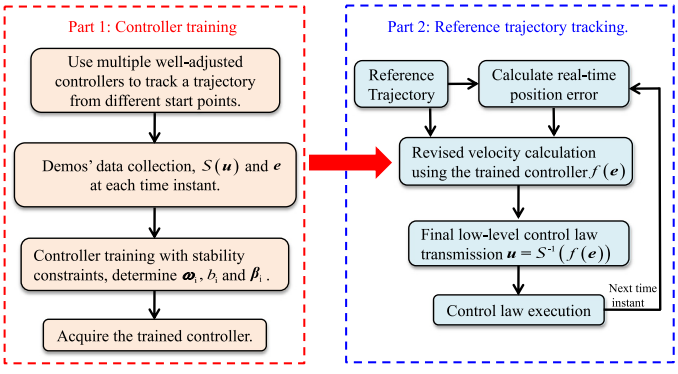

Fig. 4. The work flow diagram of the proposed algorithm.

Table 1
The work flow of the proposed algorithm.

Part 1: Controller training
(1) Demonstration design. We suggest using the well-adjusted typical control strategy to track a nonlinear trajectory multi-times from different start positions with desired performance as the demonstrations.
(2) Demonstration data collection. $\mathcal{S}(\boldsymbol{u})$ (the corrected high-level control commands, e.g., the revised velocity commands), $\boldsymbol{e}$ and the reference trajectory data $\dot{\boldsymbol{\xi}}_r, \boldsymbol{\xi}_r$ at each time instant are recorded.
(3) Controller training with considering the stability constraints. Randomly select $\boldsymbol{w}_i$ and define $b_i = 0$. Substituting the output and input, i.e., $\mathcal{S}(\boldsymbol{u})$ and $\boldsymbol{e}$, into eq. (24) with constraints in eq. (25) yields the final parameters $\boldsymbol{\beta}_i$.
(4) The controller in eq. (6) is acquired.
Part 2: Reference trajectory tracking.
(1) Based on the reference trajectory, calculate the real-time position error.
(2) Corrected velocity law calculation $\dot{\boldsymbol{\xi}}_r + \dot{\boldsymbol{e}}$ using the trained controller $f(\boldsymbol{e})$.
(3) Give the velocity law to the robot. Final control law transmission by the robot embedded system $\mathcal{S}^{-1}(f(\boldsymbol{e})) = \boldsymbol{u}$, i.e., transform the revised velocity $\dot{\boldsymbol{\xi}}_r + \dot{\boldsymbol{e}}$ control laws into robot actuator commands.
(4) Control law execution by the robot and get new position measurements.
(5) Go back to (1) of Part 2 for the next time instant.

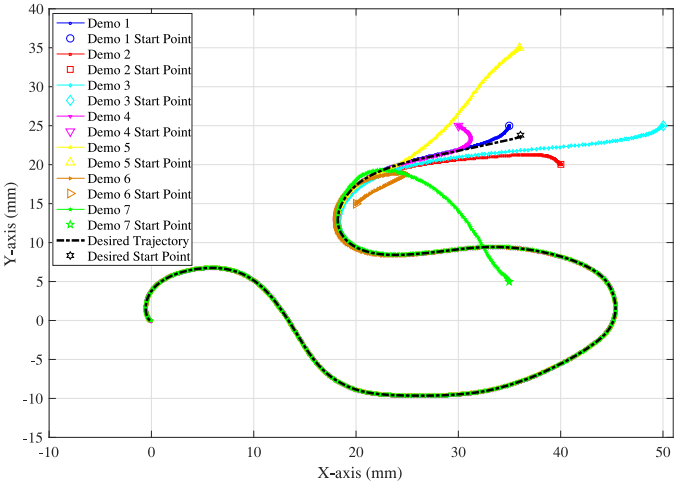

Fig. 5. Trajectories of the 7 demonstrations and the reference in example 1.

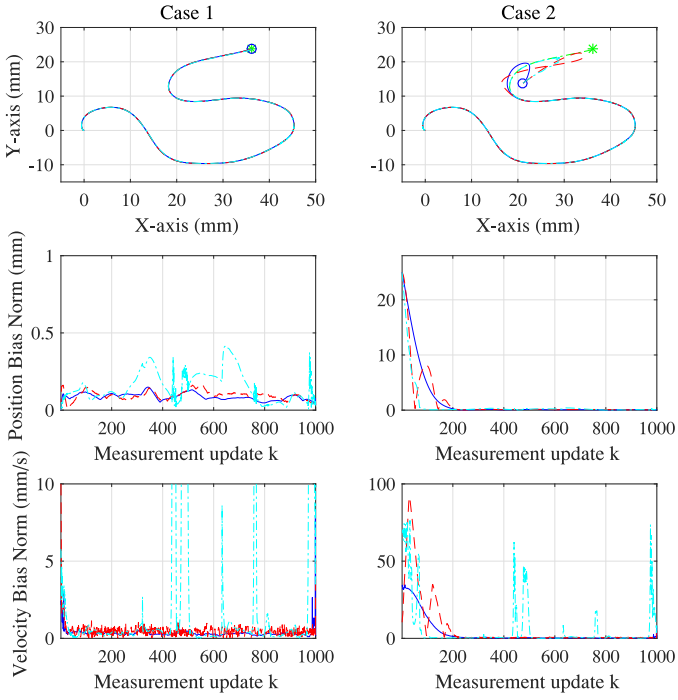

Fig. 6. Simulation results using the proposed algorithms started from different positions. The actual trajectories using method 1, 2 and 3 are depicted by the blue, red and light blue lines. The same start point is indicated with '○'. The desired trajectory is indicated by the green line starting from *.

not least, this constraint-optimization problem is solved by using the data of multiple demonstrations. More specifically, the "fmincon" function in the Matlab using the demonstrations' data with the programmed constraints in (26) can be applied to determine the controller parameters. Finally, as the parameters, i.e., $\boldsymbol{\beta}_i$, $\boldsymbol{w}_i$, b_i, are determined, the control algorithm is obtained.

To clearly understand the proposed method, the work flow of the proposed algorithm is listed in Table 1 and Fig. 4.

5. Simulation and experimental studies

The proposed control policy is evaluated in Matlab via two sets of simulations, i.e., 2D and 3D examples. The first example uses the proposed algorithm to follow different human handwriting trajectories on the 2D plane. Some trajectories in the LASA human handwriting library [26] are referred as the desired target traces. To follow these desired trajectories, i.e., minimizing the position and velocity errors at each time instant, some new demonstrations are manually produced and the corresponding data are recorded for the training. Note that in the practical applications, these new demonstrations can be made by actual human writing demonstrations, or obtained from multiple well-adjusted advanced controllers. Second, a desired nonlinear trajectory in the 3D space is produced in the Matlab and multiple tracking demonstrations starting with different initial positions are built. Based on these demonstrations, the proposed control algorithms are acquired by the training process. Then, the different desired trajectories are tracked from different start points using the trained controller.

The performance of the proposed control policy is compared with the sliding mode control (SMC) [34] method in a trajectory tracking problem. We have two reasons for not using a previous learning-based method as a comparison. First, the point-to-point learning-based methods [26,27] cannot be used for trajectory tracking. If we force to use the methods of [26,27] to the tracking problem, the performance is poor and not comparable. Second, another learning-based method in [30] is also a parameter-dependent strategy which has poor generalization to unseen sit-

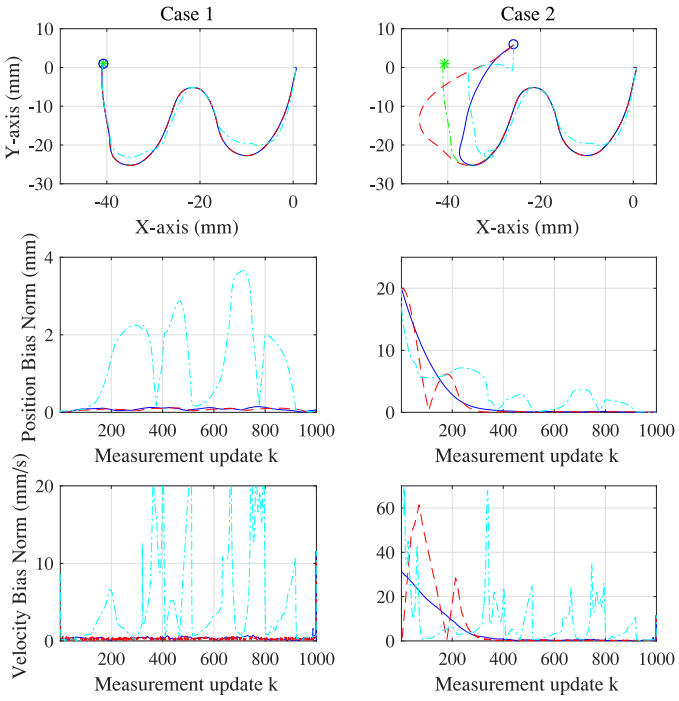

Fig. 7. Simulation comparisons of example 2.

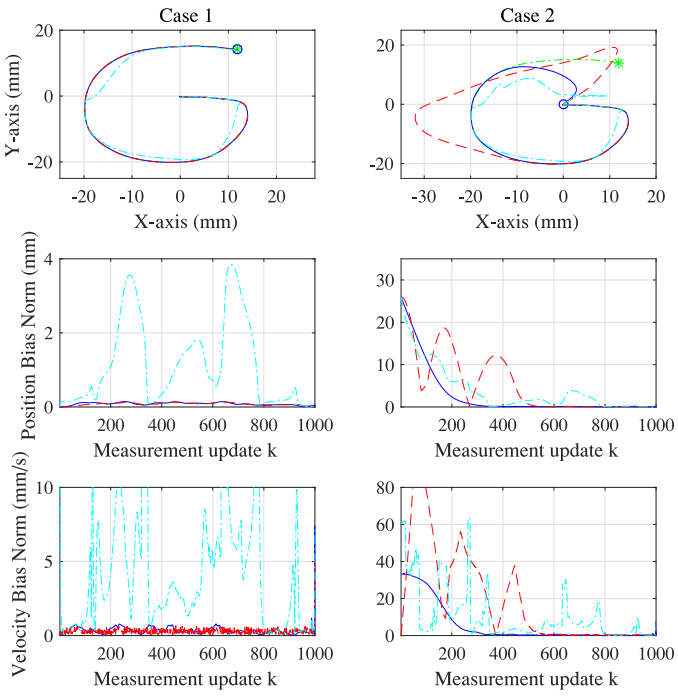

Fig. 8. Simulation comparisons of example 3.

uations. Therefore, we need to chose another nonlinear control method, i.e., SMC.

Furthermore, a typical neural model-based tracking method is also developed as another comparison. Considering the method in [7], the three-layer neural network method is utilized to model the different shape trajectories and then the model-based neural controller is designed. As the different shape reference trajectories have strong nonlinearity, the model-based neural method combined with a SMC controller is applied to design a feedforward control strategy. More details were introduced in [7]. Similar to the SMC method, to track different shape trajectories, the model-based neural controller also need to re-adjust its parameters. Theoretically, both the SMC and the model-based neural controllers have limited generalization to different situations. Note that in the simulations, the low-level robotic mechanical dynamical model $\mathcal{S}(\cdot)$ is ignored and we only consider the high-level DS control. In the proposed algorithms, we set $L = 50$ and each term of $\boldsymbol{w}_i$ is randomly selected in $(-1, 1)$ for the proposed control policy. For simplification reason, we define the proposed strategy as method 1 (M1), the SMC method as method 2 (M2) and the model-based neural method as method 3 (M3).

In the trajectory tracking problem, the geometric similarity [26,27] is not enough to evaluate the performance of the control policy. Refer to [35], the position/velocity bias norm at each measurement time instant and the root-mean-squared-error (RMSE) are utilized which have

$$\text{Bias norm}_k = \|\boldsymbol{e}_k\|_1 \quad \text{or} \quad \|\dot{\boldsymbol{e}}_k\|_1 \tag{26a}$$

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{e}_k\|^2} \quad \text{or} \quad \sqrt{\frac{1}{K}\sum_{k=1}^{K}\|\dot{\boldsymbol{e}}_k\|^2} \tag{26b}$$

where $\|\cdot\|_1$ and $\|\cdot\|$ are the L^1 norm and Euclidean norm. Here K denotes the total number of measurements.

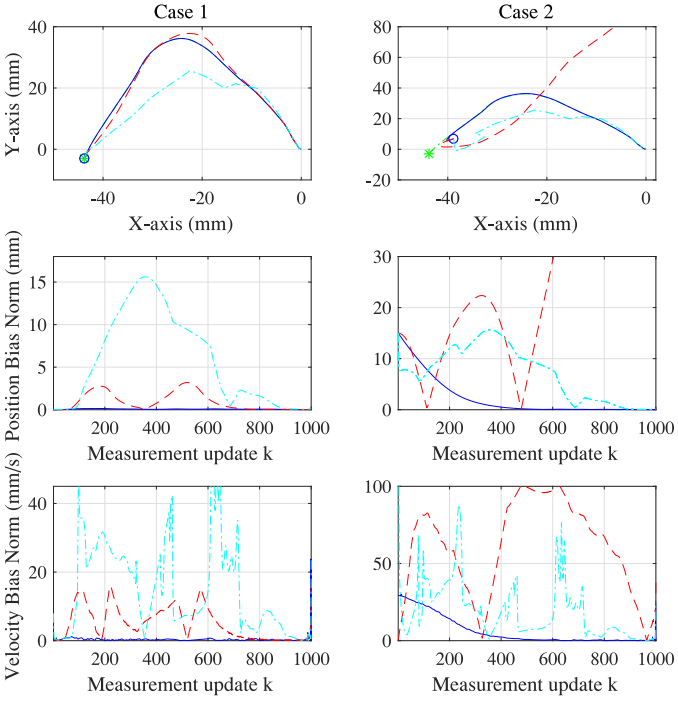

Fig. 9. Simulation comparisons of example 4.

5.1. 2D Trajectory tracking

In the 2D human handwriting simulations, the time intervals between different measurements of writing different letters are variable and each writing example contains 1000 measurements. The measurements of a desired trajectory include the reference position, velocity and acceleration information. Four examples of tracking four different trajectories are provided. In example 1, the proposed method is to follow a “Snake” trajectory; example 2 is to

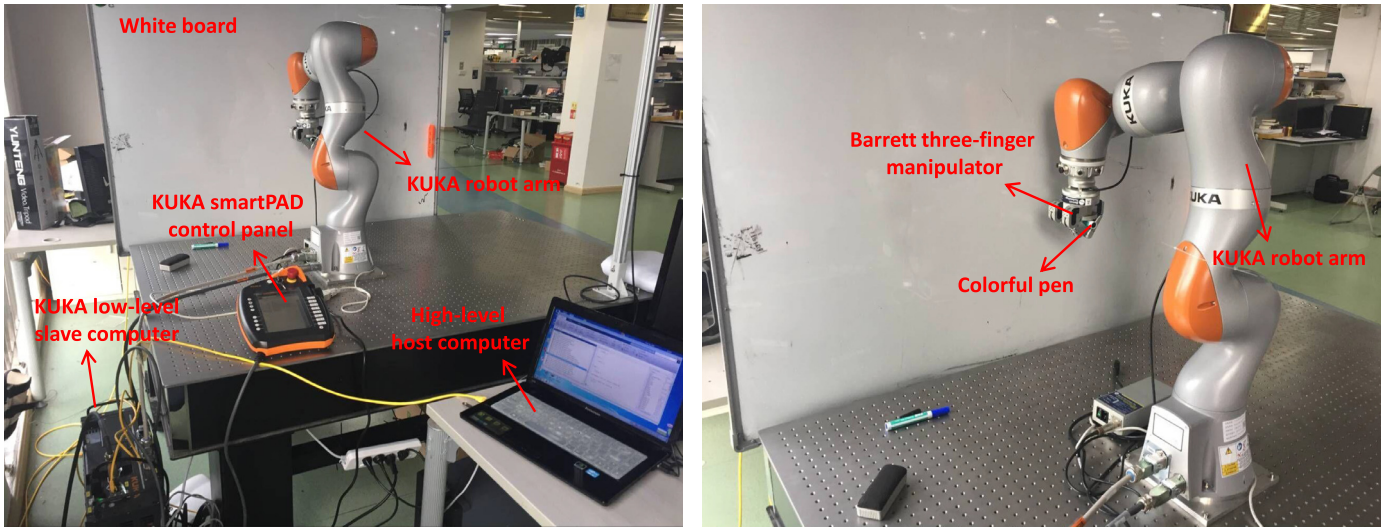

Fig. 10. Experimental setup.

follow a "W" trajectory; example 3 is to follow a "G" trajectory and example 4 is to follow an "Angle" trajectory. The proposed methods are state-error-based and thus it is not necessary to train four controllers for each example specially. The trained controller using the proposed method can be applied for any new reference trajectory tracking because of the generalization to unseen situations. Therefore, the controller trained from the example 1's demonstration data will be applied directly to track different shape trajectories in example 2 to 4.

Based on the reference trajectory of example 1, seven demonstrations with detailed state information are produced from the Simulink and the trajectories are shown in Fig. 5. Note that the performance of the trained controller is impacted by the demonstrations. However, it is hard to find a standard rule to evaluate the quality of the demonstrations. In the simulation, we selected the tracking data of using multiple well-adjusted controllers (performance is shown in Fig. 3) as the demonstrations. In other words, the different demos started from different positions using different groups of controller parameters to converge the reference trajectory fast with small overshoot. In summary, the successfully tracked demonstrations can have better tracking qualities, e.g., higher accuracy or faster reaction speed and they can improve the quality of the learnt controller. After the training process, the parameters in the proposed method are determined and then we use the learned controller of M1 to follow the different desired trajectories in the four examples with different start positions. For the sake of fairness, the proposed method (M1), the SMC algorithm (M2) and the model-based neural algorithm (M3) all use their same controller parameters for the four different examples. Specifically, the M2's controller parameters are adjusted aiming at "Snake" tracking problem. The controller parameters in M3 are trained and adjusted using the previous 7 demos' data of tracking the "Snake" trajectory. In the training, the position errors, corrected velocities and the corresponding velocity errors, i.e., $\boldsymbol{e}$, $\dot{\boldsymbol{\xi}}_{correct}$, $\dot{\boldsymbol{e}} = \dot{\boldsymbol{\xi}}_{correct} - \dot{\boldsymbol{\xi}}_r$, at all the time instants are applied as the inputs and outputs.

For example 1, the results using the different methods are shown in Fig. 6. In the first reproduced "Snake" tracking (Case 1), the actual start point is same as the desired original position. Therefore, the reproduced trajectory using the different controllers of M1, M2 and M3 are very similar to the desired reference trajectory and they look like a perfectly repeating process. Note that the relative poor velocity bias performance of M3 in case 1 of example 1 can be improved by providing more demonstration data. While for fairness reason, we only uses the 7 demonstrations in Fig. 5. M1 and M2 both perform well in examples 2–4 under case 1 which are shown in Figs. 6–9 and Tables 2 and 3. As the initial state errors are very small, both controllers can easily become stable. When the start position and desired trajectory change,

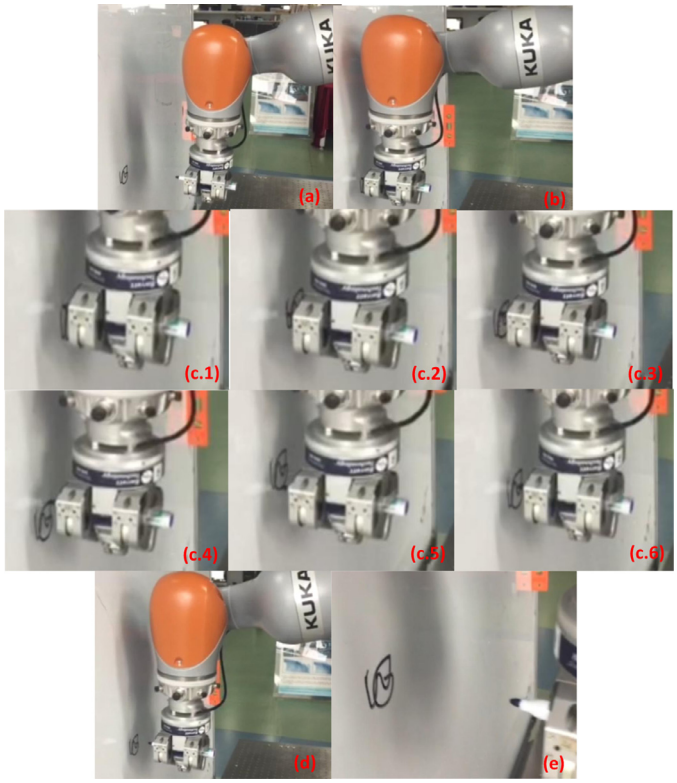

Fig. 11. The experiment process using the proposed algorithm. (a) Begin to work from the initial position, (b) move to the start position, (c) track the reference trajectory, (d) move back to the initial position, (e) the final trajectory.

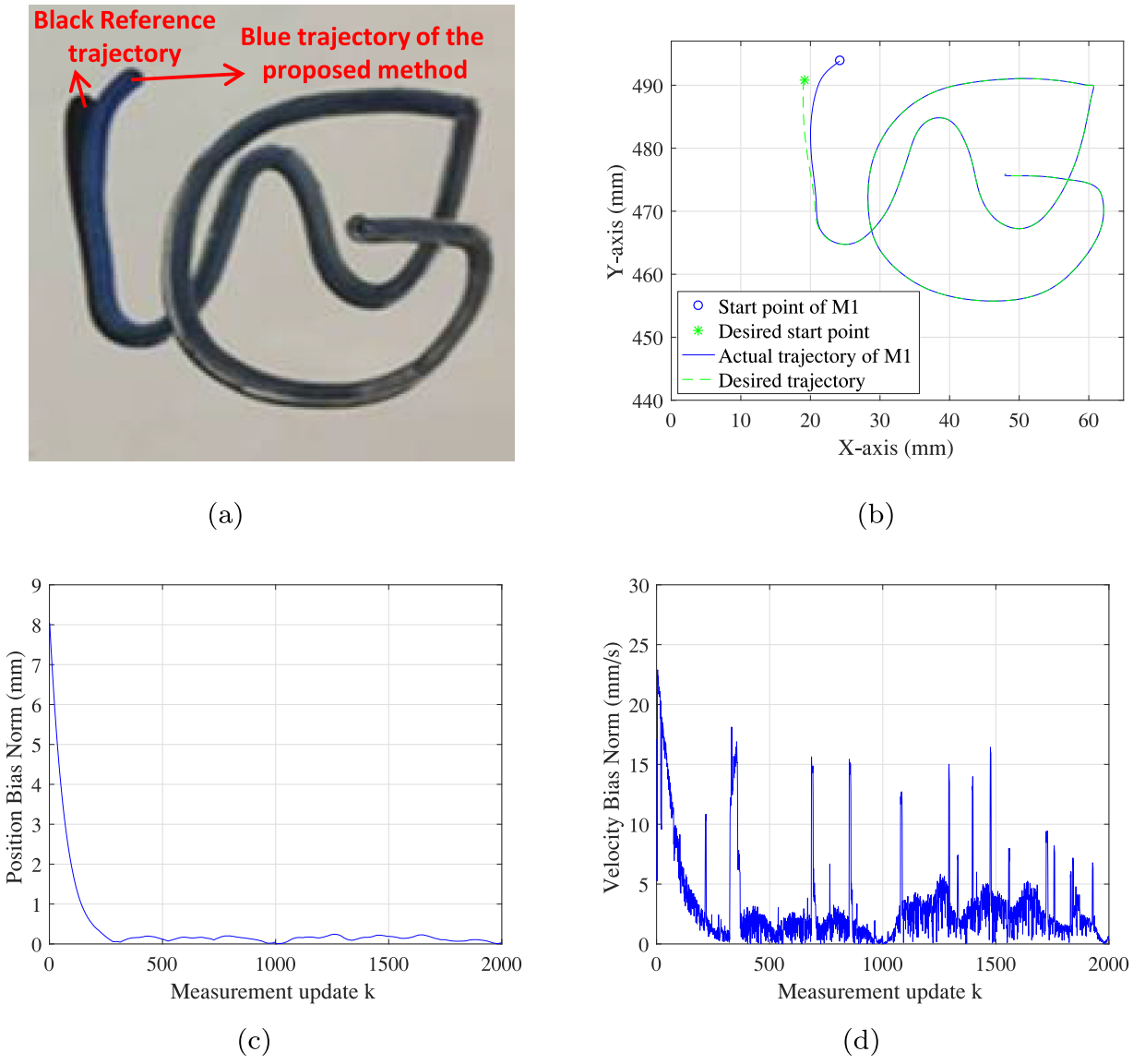

Fig. 12. Experimental results of the proposed algorithm using the KUKA robot arm. The trajectory of the proposed method is depicted by the blue line. The start point is indicated with '○'. The reference trajectory is indicated by the green line starting from *. (a) (b) trajectories, (c) position bias, (d) velocity bias.

Table 2
The position RMSE performance of the 4 examples, unit: mm.

	M1 of Ex.1	M2 of Ex.1	M3 of Ex.1	M1 of Ex.2	M2 of Ex.2	M3 of Ex.2
Case 1	0.0608	0.0798	0.1507	0.0615	0.0730	1.5909
Case 2	3.2434	3.2194	2.3923	3.9781	4.2148	4.0605
	M1 of Ex.3	M2 of Ex.3	M3 of Ex.3	M1 of Ex.4	M2 of Ex.4	M3 of Ex.4
Case 1	0.0730	0.0689	1.4147	0.0718	1.3748	7.7524
Case 2	4.2148	7.1486	5.5540	3.5304	44.7060	8.1550

Table 3
The velocity RMSE performance of the 4 examples, unit mm/s.

	M1 of Ex.1	M2 of Ex.1	M3 of Ex.1	M1 of Ex.2	M2 of Ex.2	M3 of Ex.2
Case 1	0.6352	0.5891	8.8092	0.3545	0.4140	7.6332
Case 2	8.0924	13.4608	13.3317	8.4321	14.4344	49.7285
	M1 of Ex.3	M2 of Ex.3	M3 of Ex.3	M1 of Ex.4	M2 of Ex.4	M3 of Ex.4
Case 1	0.4811	0.2818	6.1428	0.3808	6.7271	19.5985
Case 2	7.2864	24.9895	15.1115	6.3285	65.3632	60.4287

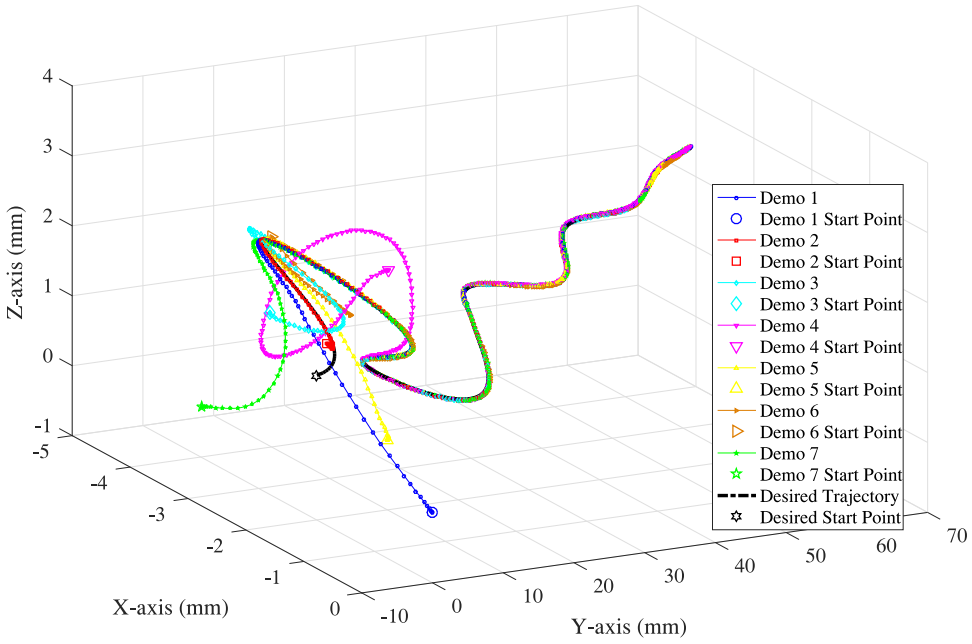

Fig. 13. Trajectories of the 7 demonstrations and the reference in example 6 and 7.

e.g., tracking "Angle", the tracking performance of M2 decreases obviously and even diverges. To improve the performance, the parameters of the SMC algorithm must be re-adjusted respect to the changed trajectories. Furthermore, the model-based method, i.e., M3, shows limited performance in examples 2, 3 and 4. This is because the trained controller of M3 only using "Snake" shape demos cannot adapt the different reference trajectory tracking tasks. Consequence, the controller parameters require the re-adjustment and re-training process. The proposed method can be understood as a unified controller of multiple different-parameterized SMC controllers. The generalization of M1 to unseen situations is obviously validated by the performance of the four examples. In addition, the position error converges faster than the velocity. The velocity errors of the different methods have a similar sudden impulse at the last several time instants because the desired trajectory suddenly stopped at the last point.

Furthermore, we use the proposed method to track a desired reference trajectory including self-interacting behaviours to validate the effectiveness of using error-based strategy. In this more complex circumstance, after the "W" shape tracking, the robot keeps moving along the "G" shape trajectory and thus the cross movements appear. We refer this example as example 5. The KUKA LBR iiwa 7 R800 industrial robot arm is applied to track this mixture trajectory. This type of robot arm has high control accuracy and fast reaction ability which can transform the high-level control laws into accurate movements. The experiment setup is shown in Fig. 10 and the robot arm is online controlled by the Matlab using a KUKA-Matlab communication socket [36]. A three-finger manipulator was installed at the end-effector holding a pen to write the trajectory onto a white board. First, the reference trajectory was written by a black pen on the white board using the reference data. Next, we changed the robot end-effector start position deviated from the desired original state and the new trajectory was written by a blue pen using the proposed algorithm. Note that the LASA trajectory data was shifted to the appropriate space for the robot arm to write the letter onto the white board. The experiment process was shown in Fig. 11. In addition, the corresponding real-time positions and time indexes were recorded by the KUKA robot sensors and the performance of the proposed method is shown in

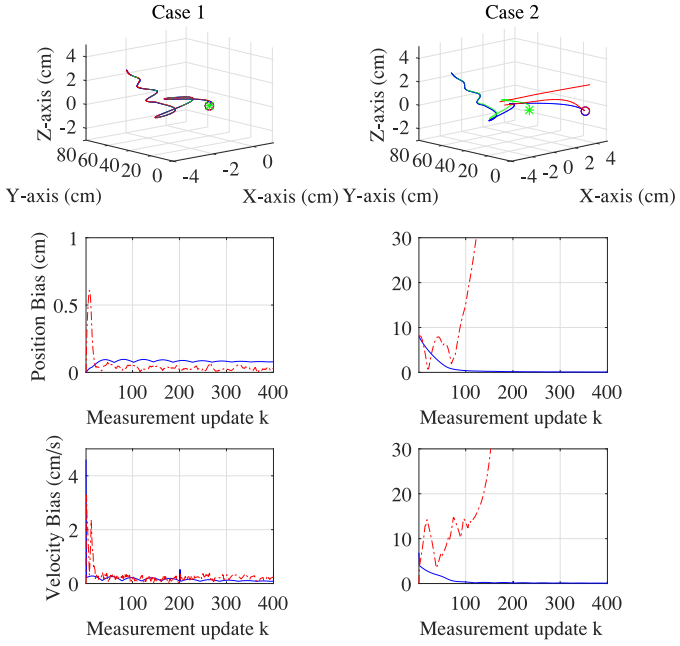

Fig. 14. Simulation results of example 6 using the proposed algorithm started from different positions. The actual trajectory is depicted by blue and red dashed lines using method 1 and 2, and the same start point is indicated with '○'. The desired trajectory is indicated by the green dashed line starting from the red *.

Fig. 12. It can be seen that the position tracking bias converges to small values fast. The velocity tracking bias always exists and this is caused by the slight execution time deviation and noise problems in the real robot system. For example, we find that the execution time for each step of the real robot system has about ± 0.0005s error. Thus, in different steps the robot may spend different times to guarantee the achievements of the given positions. Also, as the velocity is calculated by the position difference and time, the differential noise exists. But, as the measurement index increases, the velocity bias shows a downward trend to zero. The

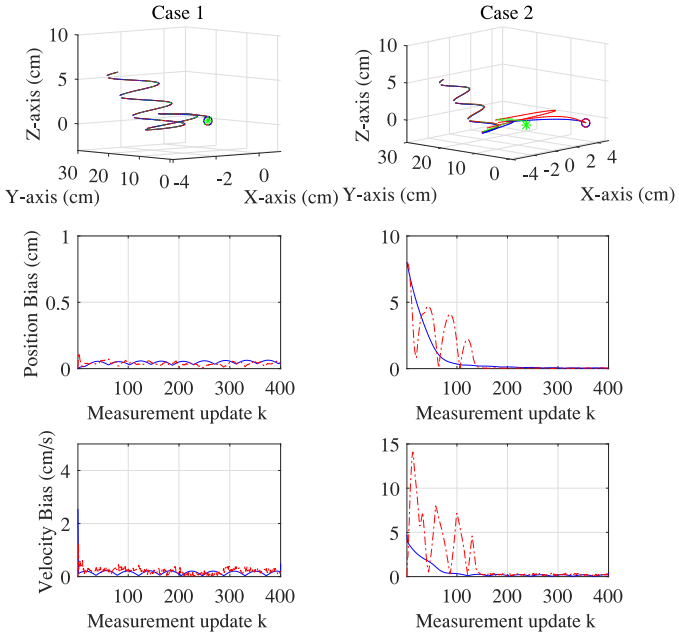

Fig. 15. Simulation results of example 7 using the proposed algorithm started from different positions. The actual trajectory is depicted by blue and red dashed lines using method 1 and 2, and the same start point is indicated with '○'. The new desired trajectory is indicated by the green line starting from *. The original reference trajectory is depicted by the black line.

position and velocity RMSEs of the experiment are 0.8046 mm and 3.7954 mm/s, respectively.

5.2. 3D trajectory tracking

In the 3D trajectory tracking simulation, the desired trajectory is mixed with the sine and straight line movements. The desired movements along the three different axes are independent and we set

$$\begin{aligned} x_r(t) &= \frac{(20-t)\pi}{60}\sin(1.5t+1)-3 \\ y_r(t) &= 3t+1 \\ \text{if} \quad t \le 10s, \quad z_r(t) &= 1+\sin\frac{1}{5}\pi t \\ \text{otherwise} \quad z_r(t) &= 1+5\left(\frac{t}{10}-1\right). \end{aligned} \tag{27}$$

with the corresponding velocities and accelerations. We assume $T = 20$ s and the measurement updates after 0.05s which means $t = 0.05, 0.1, 0.15...$s and $K = T/0.05 = 400$ measurements exist. Seven smoothly and manually produced demonstrations started from different positions to track the same desired trajectory are provided. The demonstrations and the desired trajectory are shown in Fig. 13. Similar to the 2D examples, M1 and M2 are applied in this circumstance and the reproduced tracking results using are indicated as example 6 with two cases and shown in Fig. 14. The effectiveness of the proposed method for a 3D nonlinear trajectory tracking has been validated. Furthermore, in the reproduced simulations, we change the desired trajectory to

$$\begin{aligned} x_r(t) &= \frac{\pi}{3}\sin(1.5t+1)-3 \\ y_r(t) &= t+1 \\ \text{if} \quad t \le 10s, \quad z_r(t) &= 1+\sin\frac{1}{5}\pi t \\ \text{otherwise} \quad z_r(t) &= 1+2\left(\frac{t}{10}-1\right) \end{aligned} \tag{28}$$

Table 4
The position RMSE performance of example 6 and 7.

	M1 of Ex. 6	M2 of Ex. 6	M1 of Ex. 7	M2 of Ex. 7
Case 1 (cm)	0.0726	0.0853	0.0313	80796
Case 2 (cm)	1.2602	0.3318	1.2603	80791

Table 5
The velocity RMSE performance of example 6 and 7.

	M1 of Ex. 6	M2 of Ex. 6	M1 of Ex. 7	M2 of Ex. 7
Case 1 (cm/s)	0.4616	0.0289	0.3530	1.7466
Case 2 (cm/s)	0.8434	0.1586	0.7642	2.7974

and still use the same trained controller to follow the new trace from different initial positions. This example is to validate the generalization ability of the proposed method which is scalable to a different reference trajectory. This simulation is considered as example 7 and the results are shown in Fig. 15. From Fig. 15 we can see that the proposed algorithm has the generalization ability (or strong robustness) to track different desired trajectories. Note that the performance of the trained controller can be improved by providing more high quality demonstrations. The SMC method only uses one group of parameters for the case 1 of example 6. Even it works well in some cases, it performs badly in the other circumstances. Therefore, the controller parameters must be re-adjusted and M2 has limited generalization ability. Besides, the position RMSE performance of example 6 and 7 is listed in Table 4.

6. Conclusion and discussion

We have proposed a control policy based on the LFD method to solve robot trajectory tracking problem. Firstly, the trajectory tracking problem was formulated. Secondly, a trajectory tracking control policy was derived using a simple three-layer neural network method, i.e., ELM, by minimizing the real-time state errors. In the designed controller, the input was only the position error, while the outputs were the desired velocity and the corresponding robot movement command. Furthermore, the parameter adjusting problem in the traditional model-based methods was avoided and the control algorithm was learnt from demonstrations directly. Thirdly, the stability and of the proposed algorithm and the convergence of position and velocity errors were guaranteed, and the corresponding parameter constraints were provided. In addition, it is proven that only using the velocity input of the proposed method can guarantee the position and velocity errors both converge to zeros simultaneously. Finally, from the simulation and experimental examples, the reference trajectories were tracked accurately and fast using the proposed method. In addition, the proposed control algorithm showed good generalization ability to unseen situations. Therefore, the effectiveness and the generalization ability of the proposed control policy were demonstrated.

The decoupled multi-dimensional problem can also be considered as multiple 1-dimensional control problems. Also, using the proposed method with $d = 1$ can simplify the learning process by avoiding dimension problem. Besides, as the proposed algorithm focuses on the tracking error minimizing instead of achieving the specified states, the trained controller for different x-, y- or z-axes can be inter-applied. Thus, the training process is further simplified. With the benefits of the proposed error-based learning strategy, the generalization to unseen situations is satisfactory which has been verified. In other words, the main contribution of this paper is that we proposed a strategy to learn the error con-

vergence behaviors of the desired demonstrations. This gives the proposed method an ability of generalization to unseen situations.

Furthermore, some other learning algorithms, e.g., multi-layer neural networks, support vector machines, Gaussian mixture models regression and reinforce learning, can also be applied for trajectory tracking with possible improvements and they are remained as the future works. In the future, we will focus on extending the off-line learning-based control policy to the online self-adaptive learning algorithms. In addition, if Assumption 1 is not satisfied, the stability analysis considering the motion constraints or the actuator output limitations, i.e., considering the extra constraints from $\mathcal{S}(\cdot)$, will be a new problem.

Acknowledgment

This research is funded by National Natural Science Foundation of China (Grant No. U1613210), Shenzhen Fundamental Research Programs (JCYJ20170413165528221), and China Postdoctoral Science Foundation (2018M643254). The authors would thank Zhiyang Wang and Hao Li for the helps in the experiments using the KUKA robot arm.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2019.01.052.

References

[1] M.A. Goodrich, B.S. Morse, D. Gerhardt, J.L. Cooper, M. Quigley, J.A. Adams, C. Humphrey, Supporting wilderness search and rescue using a camera-equipped mini UAV, J. Field Robot. 25 (1–2) (2008) 89–110.
[2] P. Rocco, Stability of PID control for industrial robot arms, IEEE Trans. Robot. Autom. 12 (4) (1996) 606–614.
[3] K. Morioka, J.H. Lee, H. Hashimoto, Human-following mobile robot in a distributed intelligent sensor network, IEEE Trans. Indus. Electron. 51 (1) (2004) 229–237.
[4] A.P. Aguiar, J.P. Hespanha, P.V. Kokotovic, Path-following for nonminimum phase systems removes performance limitations, IEEE Trans. Autom. Control 50 (2) (2005) 234–239.
[5] M. Bibuli, G. Bruzzone, M. Caccia, L. Lapierre, Path-following algorithms and experiments for an unmanned, J. Field Robot. 26 (8) (2009) 669–688.
[6] T. Faulwasser, Optimization-based Solutions to Constrained Trajectory-tracking and Path-following Problems, Shaker Verlag, Aachen, Germany, 2013.
[7] H.J. Rong, G.S. Zhao, Direct adaptive neural control of nonlinear systems with extreme learning machine, Neural Comput. Appl. 22 (3–4) (2013) 577–586.
[8] L.I. Jun, N. Yong-Qiang, Adaptive tracking control of a rigid arm robot based on extreme learning machine, Electr. Mach. Control 19 (4) (2015) 106–116.
[9] X.L. Wu, X.J. Wu, X.Y. Luo, Q.M. Zhu, X.P. Guan, Neural network-based adaptive tracking control for nonlinearly parameterized systems with unknown input nonlinearities, Neurocomputing 82 (2012) 127–142.
[10] T. Faulwasser, T. Weber, P. Zometa, R. Findeisen, Implementation of nonlinear model predictive path-following control for an industrial robot, IEEE Trans. Control Syst. Technol. 25 (4) (2017) 1505–1511.
[11] Z. Chu, D. Zhu, S.X. Yang, Observer-based adaptive neural network trajectory tracking control for remotely operated vehicle, IEEE Trans. Neural Netw. Learn. Syst. 28 (7) (2017) 1633–1645.
[12] F.L. Lewis, K. Liu, A. Yesildirek, Neural net robot controller with guaranteed tracking performance., IEEE Trans. Neural Netw. 7 (2) (1996). 388–99
[13] B.K. Yoo, W.C. Ham, Adaptive control of robot manipulator using fuzzy compensator, IEEE Trans. Fuzzy Syst. 8 (2) (2000) 186–199.
[14] D. Lam, C. Manzie, M. Good, Model predictive contouring control, in: Proceedings of the IEEE Conference on Decision and Control, 2011, pp. 6137–6142.
[15] M.L. Minsky, S. Papert, Perceptrons : An Introduction to Computational Geometry, The MIT Press, 1969.
[16] J.I. Mulero-Martinez, Robust GRBF static neurocontroller with switch logic for control of robot manipulators, IEEE Trans. Neural Netw. Learn. Syst. 23 (7) (2012) 1053–1064.
[17] L. Sciavicco, B. Siciliano, Modelling and Control of Robot Manipulators, Springer, London, 2000.
[18] D. Kostic, B. De Jager, M. Steinbuch, R. Hensen, Modeling and identification for high-performance robot control: an RRR-robotic arm case study, IEEE Trans. Control Syst. Technol. 12 (6) (2004) 904–919.
[19] H.P.H. Anh, N.N. Son, N.T. Nam, Adaptive evolutionary neural control of perturbed nonlinear serial PAM robot, Neurocomputing 267 (C) (2017) 525–544.
[20] N.N. Son, H.P.H. Anh, Adaptive displacement online control of shape memory alloys actuator based on neural networks and hybrid differential evolution algorithm, Neurocomputing 166 (C) (2015) 464–474.
[21] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks., IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.
[22] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[23] A.M. Andrew, An introduction to support vector machines and other kernel based learning methods, Kybernetes 32 (1) (2001) 1–28.
[24] D. Nguyen-Tuong, M. Seeger, J. Peters, D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, Local gaussian process regression for real time online model learning and control, Adv. Neural Inf. Process. Syst. 22 (2009) (2008) 1193–1200.
[25] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intell. Serv. Robot. 9 (1) (2015) 1–29.
[26] S.M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with gaussian mixture models, IEEE Trans. Robot. 27 (5) (2011) 943–957.
[27] J.H. Duan, Y.S. Ou, J.B. Hu, Z.Y. Wang, S.K. Jin, C. Xu, Fast and stable learning of dynamical systems based on extreme learning machine, IEEE Trans. Syst. Man Cybern. Syst. PP (99) (2017) 1–11.
[28] S.M. Khansari-Zadeh, A. Billard, Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions, Robot. Auton. Syst. 62 (6) (2014) 752–765.
[29] A. Shukla, A. Billard, Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies, Robot. Auton. Syst. 60 (3) (2012) 424–440.
[30] S.M. Khansari-Zadeh, O. Khatib, Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors, Auton. Robots 41 (1) (2017) 45–69.
[31] S.B. Niku, Introduction to Robotics : Analysis, Control, Applications, Prentice Hall, 2001.
[32] H.K. Khalil, Nonlinear Systems, Third Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 2002.
[33] W. Rudin, Principles of Mathematical Analysis, 3rd Edition, McGraw-Hill, 1976.
[34] S. Xu, D. Su, Research of magnetically suspended rotor control in control moment gyroscope based on fuzzy integral sliding mode method, in: Proceedings of the International Conference on Electrical Machines and Systems (ICEMS), Oct. 2013, pp. 1901–1906. Busan, South Korea
[35] A. Lemme, Y. Meirovitch, M. Khansarizadeh, T. Flash, A. Billard, J.J. Steil, Open–source benchmarking for learned reaching motion generation in robotics, Paladyn J. Behav. Robot. 6 (1) (2015) 30–41.
[36] M. Safeea, P. Neto, KUKA Sunrise Toolbox: Interfacing Collaborative Robots with MATLAB, IEEE Robot. Autom. Mag. (Early Access) (2018), doi:10.1109/MRA.2018.2877776.

Sheng Xu received B.S. degree from Shandong University, Shandong, China, in 2011, the M.S. degree from Beihang University, Beijing, China, in 2014, both in Electrical Engineering, and the Ph.D. degree in Telecommunications Engineering from ITR, the University of South Australia, Australia, in 2017. He is currently a postdoctoral fellow with the Center for Intelligent and Biomimetic Systems, Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS), Shenzhen, China. His main research interests include target tracking, nonlinear estimation, statistical signal processing, servo system control and robot control.

Yongsheng Ou received the Ph.D. degree in Department of Mechanical and Automation from the Chinese University of Hong Kong, Hong Kong, China, in 2004. He was a Postdoctoral Researcher in Department of Mechanical Engineering, Lehigh University, USA, for Five years. He is currently a Professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He is the author of more than 150 papers in major journals and international conferences, as well as a coauthor of the monograph on Control of Single Wheel Robots (Springer, 2005). He also serves as Associate Editor of IEEE Robotics & Automation Letters. His research interests include control of complex systems, learning human control from demonstrations, and robot navigation.

Jianghua Duan is currently pursuing the Ph.D. degree in pattern recognition and intelligent system at University of Chinese Academy of Sciences. His research interest includes robot learning from human demonstrations in an unstructured environment and compliant control for safe human robot interaction.

Xinyu Wu is now a professor at Shenzhen Institutes of Advanced Technology, and director of Center for Intelligent Bionic. He received his B.E. and M.E. degrees from the Department of Automation, University of Science and Technology of China in 2001 and 2004, respectively. His Ph.D. degree was awarded at the Chinese University of Hong Kong in 2008. He has published over 180 papers and two monographs. His research interests include computer vision, robotics, and intelligent system.

Wei Feng received the B.S. degree from the School of Materials Science and Engineering in 2001 and the Ph.D. degree in 2006, both from the Huazhong University of Science and Technology, Wuhan, China. He is currently an Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, and the Chinese University of Hong Kong, Hong Kong. His research interests include robotics and intelligent system, computational geometry, and computer graphics, CAD/CAM/CAE.

Ming Liu (S'12M'12SM'18) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical Engineering and Process Engineering, ETH Zurich, Zurich, Switzerland, in 2013. He was a Visiting Scholar with Erlangen Nurnberg University, Erlangen, Germany, and the Fraunhofer Institute IISB, Erlangen. He is currently an Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His current research interests include autonomous mapping, visual navigation, topological mapping, and environment modeling.