Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks

Shuai Li [a], Sanfeng Chen [b], Bo Liu [c,*], Yangming Li [d], Yongsheng Liang [b]

[a] Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[b] Key Lab of Visual Media Processing and Transmission, Shenzhen Institute of Information Technology, Shenzhen, Guangdong 518029, China
[c] Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA
[d] Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, Anhui 07309, China

ARTICLE INFO

ABSTRACT

This paper studies the decentralized kinematic control of multiple redundant manipulators for the cooperative task execution problem. The problem is formulated as a constrained quadratic programming problem and then a recurrent neural network with independent modules is proposed to solve the problem in a distributed manner. Each module in the neural network controls a single manipulator in real time without explicit communication with others and all the modules together collectively solve the common task. The global stability of the proposed neural network and the optimality of the neural solution are proven in theory. Application orientated simulations demonstrate the effectiveness of the proposed method.

1. Introduction

With the development of mechanics, electronics, computer engineering, etc., using a collection of manipulators to perform a common task, such as load transport [1], cooperative assembly [2], dextrous grasping [3], coordinate welding [4], etc., is becoming increasingly popular and has received considerable studies. The solution of executing the task using redundant manipulators, which have more degree-of-freedom (DOF) than required, normally is not unique. The extra design degrees can be exploited for obstacle avoidance, performance optimization and so on to improve the performance.

A fundamental issue in multiple redundant manipulator control is the redundancy resolution problem, which provides feasible solutions in the joint space to a task in the Cartesian space. Conventionally, the general solution of redundancy resolution is obtained by solving a set of redundant time-varying linear [5,6]. However, as pointed in [7,8], this type of method cannot generate a repeatable solution and the drift of joint angles is unavoidable. Authors in [9] formulate the problem of single redundant manipulator kinematic control as a constrained optimization problem

and opened an avenue to study the redundancy resolution problem using optimization based methods [10,11]. In [12], a penalty term is introduced into the objective function to restrict the solution inside a physically feasible range. This method enables direct monitoring and control of the magnitudes of the individual joint torques. However, the solution is merely an approximate solution of the problem and the approximation error strongly depends on the coefficient of the penalty term. To fix this problem, in [13], the optimization problem is studied in the dual space and dual neural networks are developed to solve the problem in real time. Further performance improvements of the dual neural network approach, such as convergence time, architecture complexity, etc., are obtained in the successive studies [14–16]. In [17], a local optimization approach is proposed to resolve the kinematic control problem of redundant manipulators. This approach is applicable to either serial manipulators or parallel manipulators but the result is a locally optimal one instead of globally. In [18], an optimal real-time redundancy resolution scheme is proposed to solve the problem. The optimal control law can be derived in real-time using adaptive critic framework after a period of off-line training. An analytic-iterative solution is presented in [19] to solve the redundancy resolution problem formulated as an inequality constrained convex optimization problem. The optimization based formulation framework also allows us to analyze the multiple redundant manipulator cooperative task execution problem. However, the framework

* Corresponding author.
E-mail addresses: lshuai@stevens.edu (S. Li), chensanf@sziit.com.cn (S. Chen), boliu@cs.umass.edu (B. Liu), ymli@iim.ac.cn (Y. Li), liangys@sziit.com.cn (Y. Liang).

does not provide any information about how to design a decentralized control strategy not requiring explicit information from other manipulators.

In this paper, the cooperative task execution problem is formulated as a separable constrained quadratic programming problem. Benefiting from the separable property of our formulation, this problem can be decoupled into a collection of subproblems, each of which corresponds to a single manipulator. Inspired by the real-time optimization capability of recurrent neural networks [20–27], a dual recurrent neural network, which has separate modules associated with every manipulator in the collection, is designed to solve the problem in real time. To the best of our knowledge, this is the first time that the dual recurrent neural network method [28–30] is extended to the decentralized control of multiple redundant manipulator systems.

In contrast to the great success of recurrent dual neural network based kinematic control of a single redundant manipulator, there has been very little attention to the solution on cooperative control of collaborative redundant manipulators employing dual neural networks. In this background, the dual neural network approach is extended to solve the kinematic control problem of multiple redundant manipulators. Compared to existing dual neural networks [20,22,24,25,28–30], a prominent feature of the proposed neural network lies in its different architecture, which is composed of separate modules associated with every manipulator. This feature enables a completely decentralized control of multiple manipulators.

The remainder of this paper is organized as follows. In Section 2, the multiple manipulator cooperative task execution problem is formulated as a constrained quadratic programming problem. In Section 3, a recurrent neural network with separate modules are proposed to solve the problem in real time. In Section 4, the global stability of the proposed neural network and the optimality of the neural solution are proven in theory. In Section 5, simulations are performed to validate the effectiveness of the proposed method. Section 6 concludes this paper.

2. Problem formulation

In this section, we describe the problem of cooperative task execution with multiple redundant manipulators. Firstly a brief introduction on the redundant manipulator kinematics is given and then, based on this, the multiple manipulator task execution problem is formulated as a constrained quadratic programming problem.

2.1. Redundant manipulator kinematics

For a redundant manipulator, the position and orientation of its end-effector is uniquely determined by its configuration in the joint space, i.e., for a m-DOF redundant manipulator working in a n-dimensional Cartesian space, there exists the following single-valued mapping:

$$r(t) = f(\theta(t)) \tag{1}$$

where $r(t) \in \mathbb{R}^n$ and $\theta(t) \in \mathbb{R}^m$ with $m > n$ are the coordinate of the manipulator in the Cartesian space at time t and the coordinate in the joint space, respectively. The mapping $f(\cdot)$ is a nonlinear function with known parameters for a given manipulator. Calculating time derivative on both sides of (1) yields,

$$\dot{r}(t) = J(\theta(t))\dot{\theta}(t) \tag{2}$$

where $\dot{r}(t) \in \mathbb{R}^n$ and $\dot{\theta}(t) \in \mathbb{R}^m$ are the velocity of the manipulator in the Cartesian space and that in the joint space, respectively. $J(\theta(t)) = \partial f(\theta(t))/\partial \theta(t)$ is the Jacobian matrix. Apparently, $\dot{r}(t)$ is an affine function with respect to $\dot{\theta}(t)$ and is relatively easier to deal with compared with the nonlinear mapping (1).

In robotics, the problem of velocity inverse kinematics is concerned with finding a solution $\dot{\theta}(t)$ for the manipulator model (2) with given desired velocity $\dot{r}(t)$ in Cartesian space and known manipulator model $J(\cdot)$. For redundant manipulators, $n < m$, which means the number of equalities in the velocity inverse kinematic problem is less than the number of dimensions of the decision variable $\dot{\theta}(t)$, and therefore normally the solution to this problem is not unique. This property of redundant manipulators enables us to select the best solution among all the feasible ones according to certain optimum criteria and extra constraints. Possible optimum criteria includes minimum of the Euclidian norm or the infinity norm of the joint velocity vector and possible constraints include joint angle limits, joint speed limits, etc.

2.2. Cooperative task execution using multiple manipulators

Consider the problem of payload transport with a collection of redundant manipulators. The goal is to cooperatively move the payload along a desired reference trajectory with the end-effector of each manipulator holding a different place or a handle on the payload. This task involves two aspects: first, a reference point on the payload, e.g., the center of mass, is expected to track the reference trajectory. Second, the end-effectors are expected to maintain the original formation in space. Note that the second aspect is required to avoid stretching or squeezing of the payload possibly arising from the relative movement between the end-effectors and this requirement can be satisfied by moving all end-effectors with the same velocity as the reference point. By assigning a reference tracking velocity, denoted by $v_d(t)$, along the desired reference trajectory with a given absolute value, the first aspect of the task can be achieved by velocity control and the second aspect can be satisfied by steering all end-effectors with the same velocity in the Cartesian space. In a word, the two aspects of the task can be achieved by steering end-effectors of all manipulators with the same velocity $v_d(t)$. In equation,

$$J_i(\theta_i(t))\dot{\theta}_i(t) = v_d(t) \quad \text{for } i = 1, 2, \ldots, k \tag{3}$$

where $v_d(t) \in \mathbb{R}^n$ denotes the desired velocity of the reference point on the payload at time t, $\theta_i(t) \in \mathbb{R}^m$ and $\dot{\theta}_i(t) \in \mathbb{R}^m$ are the coordinate and the velocity of the i th manipulator in the joint space at time t, respectively, $J_i(\theta_i(t)) \in \mathbb{R}^{n \times m}$ is the Jacobian matrix of the ith manipulator at time t, k denotes the number of manipulators. Without introducing confusions (3) is abbreviated into the following form for easy reading,

$$J_i\dot{\theta}_i = v_d \quad \text{for } i = 1, 2, \ldots, k \tag{4}$$

where v_d, θ_i, $\dot{\theta}_i$ and J_i are short for $v_d(t)$, $\theta_i(t)$, $\dot{\theta}_i(t)$ and $J_i(\theta_i(t))$ in (3), respectively. As the velocity mapping from the joint space to the Cartesian space is affine as shown in (2), using velocity control of the collection of manipulators thoroughly simples the design, compared with the direct position control based on the nonlinear transformation (1). In addition, resulting from the redundancy property of redundant manipulators, the solution satisfying the above mentioned two aspects of the task is not unique, which allows us to take extra optimization criteria and constraints into account. Without loss of generality, we minimize the Euclidean norm squared of the joint velocities, i.e., $\sum_{i=1}^{k} \dot{\theta}_i^T \dot{\theta}_i$ and impose the joint velocity constraints $\eta^- \leq \dot{\theta}_i \leq \eta^+$, to exploit the extra design freedoms. In summary, the cooperative task execution problem can be formulated as follows:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{k} \dot{\theta}_i^T \dot{\theta}_i$$

subject to $\quad J_1\dot{\theta}_1 = v_d, J_2\dot{\theta}_2 = v_d, \ldots J_k\dot{\theta}_k = v_d$

$$\eta^- \le \dot{\theta}_1 \le \eta^+, \eta^- \le \dot{\theta}_2 \le \eta^+, \ldots, \eta^- \le \dot{\theta}_k \le \eta^+ \tag{5}$$

where k denotes the number of manipulators, $v_d \in \mathbb{R}^n$ is the desired velocity of the reference point, $\theta_i \in \mathbb{R}^m$ and $\dot{\theta}_i \in \mathbb{R}^m$ are the coordinate and the velocity of the i th manipulator in the joint space for $i = 1, 2, \ldots, k$, respectively. $J_i \in \mathbb{R}^{n \times m}$ is the Jacobian matrix of the ith manipulator for $i = 1, 2, \ldots, k$. $\eta^+ \in \mathbb{R}^m$ and $\eta^- \in \mathbb{R}^m$ are the upper and lower limit of the allowed velocity in the joint space.

Remark 1. In the problem formulation (5), the kinematic constraints (or more exactly, joint velocity limits and reference velocity constraints) are considered. Constraints, such as joint limits and collision avoidance, however, are not considered explicitly in (5). Actually, as done in [30], the joint limit constraint can be included in the formulation (5) by imposing extra inequality constraints on $\dot{\theta}_i$ as follows:

$$c_0(\theta^- - \theta_i) \le \dot{\theta}_i \le c_0(\theta^+ - \theta_i)$$

where $\theta^+ \in \mathbb{R}^m$ and $\theta^- \in \mathbb{R}^m$ are the upper and lower limit of the allowed angle in the joint space, $c_0 \in \mathbb{R}$, $c_0 > 0$ is a coefficient. Similarly, collision avoidance can be taken into account by following the inequality-based obstacle avoidance QP method proposed in [31], which also imposes linear inequality constraint on $\dot{\theta}$.

Until now, the cooperative task execution problem has been formulated as an optimization problem, a quadratic programming problem to be more accurate, with respect to the decision variables $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_k$. Numerical methods are available to solve this problem point by point at every sampling time. However, this strategy has a total computation time inversely proportional to the sampling period and is very computationally intensive when the sampling period is relatively short, not to mention continuous control, which is equivalent to the case with a zero sampling period. This motivates our study to explore the recurrent neural network method for real-time optimization of the problem.

3. Real-time control via recurrent neural networks

In this section, a recurrent neural network is designed to complete the online optimization of the cooperative task execution problem (5).

According to Karash–Kuhn–Tucker (KKT) conditions [32], the solution to problem (5) satisfies

$$\dot{\theta}_i = -J_i^T \lambda_i - \mu_i \tag{6a}$$

$$J_i\dot{\theta}_i = v_d \tag{6b}$$

$$\begin{cases} \dot{\theta}_i = \eta^+ & \text{if } \mu_i > 0 \\ \eta^- \le \dot{\theta}_i \le \eta^+ & \text{if } \mu_i = 0 \\ \dot{\theta}_i = \eta^- & \text{if } \mu_i < 0 \\ \text{for } i = 1, 2, \ldots, k \end{cases} \tag{6c}$$

Remark 2. Benefiting from the fact that the problem (5) is separable relative to $\theta_1, \theta_2, \ldots, \theta_k$, the decision variables $\theta_1, \theta_2, \ldots, \theta_k$ in (6) are completely decoupled. This means that for all manipulators, greedily minimizing their local cost function under local constraints (for the ith manipulator, the local cost is $\frac{1}{2}\dot{\theta}_i^T\dot{\theta}_i$ and the local constraints are $J_i\dot{\theta} = v_d$ and $\eta^- \le \dot{\theta}_i \le \eta^+$), leads to a collective behavior, which solves the global optimization problem (5). This property enables us to design a recurrent neural network with k separate modules, each of which control a

single manipulator independently but still, all the modules together, collectively solve the cooperative task executive problem (5).

Note that (6c) can be simplified into the following form:

$$\dot{\theta}_i = g(\dot{\theta}_i + \mu_i) \quad \text{for } i = 1, 2, \ldots, k \tag{7}$$

with $g(x) = [g_1(x_1), g_2(x_2), \ldots, g_m(x_m)]^T$ for $x = [x_1, x_2, \ldots, x_m]^T$ and $g_j(x_j)$ to be the following form for $j = 1, 2, \ldots, m$:

$$g_j(x_j) = \begin{cases} \eta_j^+ & \text{if } x_j > \eta_j^+ \\ x_j & \text{if } \eta_j^- \le x_j \le \eta_j^+ \\ \eta_j^- & \text{if } x_j < \eta_j^- \end{cases} \tag{8}$$

where η_j^- and η_j^+ are the j th elements of η^- and η^+, respectively. Substituting (6a) into (7) to cancel out $\dot{\theta}_i$ yields,

$$\mu_i = -g(-J_i^T \lambda_i) - J_i^T \lambda_i \quad \text{for } i = 1, 2, \ldots, k \tag{9}$$

which means μ_i can be explicitly expressed in terms of λ_i for $i = 1, 2, \ldots, k$. $\dot{\theta}_i$ can also be expressed in terms of λ_i by substituting (9) into (6a),

$$\dot{\theta}_i = g(-J_i^T \lambda_i) \quad \text{for } i = 1, 2, \ldots, k \tag{10}$$

Plugging (7) into (6b) yields,

$$J_i g(\dot{\theta}_i + \mu_i) = v_d \quad \text{for } i = 1, 2, \ldots, k \tag{11}$$

Canceling out $\dot{\theta}_i$ by bringing Eq. (6a) into Eq. (11) generates,

$$J_i g(-J_i^T \lambda_i) = v_d \quad \text{for } i = 1, 2, \ldots, k \tag{12}$$

We use a recurrent neural network to solve $\lambda_1, \lambda_2, \ldots, \lambda_k$ in (12) as follows:

$$\epsilon_i \dot{\lambda}_i = J_i g(-J^T \lambda_i) - v_d \quad \text{for } i = 1, 2, \ldots, k \tag{13}$$

where $\epsilon_i > 0$ for $i = 1, 2, \ldots, k$ is a scaling factor. Now, a recurrent neural network is obtained to solve the cooperative task execution problem (5). This neural network is composed of k modules corresponding to the k manipulators, and the state and output equations of the i th module for $i = 1, 2, \ldots, k$ are summarized as follows:

state equation $\epsilon_i \dot{\lambda}_i = J_i g(-J^T \lambda_i) - v_d \tag{14a}$

output equation $\dot{\theta}_i = g(-J_i^T \lambda_i) \tag{14b}$

where $\epsilon_i > 0$ for $i = 1, 2, \ldots, k$ is a scaling factor, k is the number of manipulators, and the function $g(\cdot)$ is defined in (8).

The recurrent neural network for solving the cooperative task execution problem (5) consists of k modules, with k equal to the number of manipulators. Each module is composed of n dynamic neurons, which equals the dimension of the Cartesian space. In a three-dimensional space, $n = 3$ for position control problem and $n = 6$ for the case with both position and orientation control. In each module, there are n dynamic neurons and $n(n-1)/2$ interconnections between neurons. Therefore, for the whole neural network, the total number of neurons is kn and the interconnection complexity is $O(kn^2)$, which is polynomially dependent on the number of manipulators k and the dimension of the Cartesian space n. Besides the capability of solving the cooperative task execution problem in a decentralized manner, which cannot be handled by the conventional recurrent neural network based method for single manipulator motion control [25,33,34], another important distinction from them is that our model has a low architecture complexity and the complexity has no dependence on m, which is the DOF of the redundant manipulator. This property simplifies the hardware design for the analog circuit implementation of the neural network and reduces the cost, especially for redundant manipulators with a large m. Briefly, we have the following remark on this point.

Remark 3. The neural network presented in this paper consists of k modules and the whole neural network has a total number of kn neurons and has an interconnection complexity $O(kn^2)$, which does not depend on the DOF of the redundant manipulator. In addition, The neural network solves the problem in a decentralized manner. This cannot be achieved by the conventional recurrent dual neural networks.

4. Theoretical results

On the stability, solution optimality of the recurrent neural network (14) to the problem (5), we have conclusions stated in the following theorems.

Theorem 1. *The recurrent neural network* (14) *with* $\epsilon_i > 0$ *for* $i = 1, 2, \ldots, k$ *is globally stable in the sense of Lyapunov and converges to an equilibrium point* $[\lambda_1^{*T}, \lambda_2^{*T}, \ldots, \lambda_k^{*T}]^T$.

Proof. To prove the conclusion, the following radially unbounded Lyapunov function is constructed,

$$V = \sum_{i=1}^{k} \frac{\epsilon_i}{2} (J_i g(-J_i^T \lambda_i) - v_d)^T (J_i g(-J_i^T \lambda_i) - v_d) \tag{15}$$

Note that V is a function of the state variables $\lambda_1, \lambda_2, \ldots, \lambda_k$. $V \geq 0$ and the equality holds when $J_i g(-J_i^T \lambda_i) - v_d = 0$ for all $i = 1, 2, \ldots, k$, i.e., the equality holds at the equilibrium point λ_i^* of (14a) for all $i = 1, 2, \ldots, k$. The time derivative of V along the neural network trajectory (14) can be obtained as follows:

$$\dot{V} = -\sum_{i=1}^{k} \epsilon_i (J_i g(-J_i^T \lambda_i) - v_d)^T (J_i D^+ g(-J_i^T \lambda_i) J_i^T \dot{\lambda}_i) \tag{16}$$

where $D^+ g(-J_i^T \lambda_i)$ denotes the upper right dini-derivative of the function $g(-J_i^T \lambda_i)$. According to the definition of $g(\cdot)$ in (8), $D^+ g(-J_i^T \lambda_i)$ is a diagonal matrix of the form $D^+ g(-J_i^T \lambda_i) = \text{diag}(c_{1i}, c_{2i}, \ldots, c_{mi})$ and the j th diagonal element c_{ji} is as follows:

$$c_{ji} = \begin{cases} 1 & \text{if } \eta_j^- \leq d_{ji} < \eta_j^+ \\ 0 & \text{if } d_{ji} < \eta_j^- \text{ or } d_{ji} \geq \eta_j^+ \end{cases} \tag{17}$$

where d_{ji} represents the j th element of the vector $-J_i^T \lambda_i$. From (17), it is obtained that $c_{ji} \geq 0$ for all $j = 1, 2, \ldots, m$. Thus, we can express the diagonal matrix $\text{diag}(c_{1i}, c_{2i}, \ldots, c_{mi}) = \text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \ldots, \sqrt{c_{mi}})$ $\text{diag}^T(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \ldots, \sqrt{c_{mi}})$. Based on this, further conclusion holds,

$$\dot{V} = -\sum_{i=1}^{k} ((J_i g(-J_i^T \lambda_i) - v_d)^T J_i \, \text{diag}(c_{1i}, c_{2i}, \ldots, c_{mi}) J_i^T (J_i g(-J_i^T \lambda_i) - v_d))$$

$$= -\sum_{i=1}^{k} ((J_i g(-J_i^T \lambda_i) - v_d)^T J_i \, \text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \ldots, \sqrt{c_{mi}})$$

$$\times \text{diag}^T(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \ldots, \sqrt{c_{mi}}) J_i^T (J_i g(-J_i^T \lambda_i) - v_d))$$

$$= -\sum_{i=1}^{k} \|\text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \ldots, \sqrt{c_{mi}}) J_i^T (J_i g(-J_i^T \lambda_i) - v_d)\|^2 \leq 0 \tag{18}$$

which means the neural network (14) is globally stable to an equilibrium point $[\lambda_1^{*T}, \lambda_2^{*T}, \ldots, \lambda_k^{*T}]^T$ in the sense of Lyapunov [35]. This completes the proof. □

The following Theorem reveals the relation between the equilibrium point of the neural network and the optimal solution to the cooperative task execution problem (5).

Theorem 2. *Let* $[\lambda_1^{*T}, \lambda_2^{*T}, \ldots, \lambda_k^{*T}]^T$ *be an equilibrium point of the neural network dynamic* (14a). *The output of this neural network at* $[\lambda_1^{*T}, \lambda_2^{*T}, \ldots, \lambda_k^{*T}]^T$, *which is* $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, \ldots, \dot{\theta}_k^{*T}]^T = [g^T(-J_1^T \lambda_1^*),$

$g^T(-J_2^T \lambda_2^*), \ldots, g^T(-J_k^T \lambda_k^*)]^T$, *is the optimal solution to the cooperative task execution problem* (5).

Proof. Since $[\lambda_1^{*T}, \lambda_2^{*T}, \ldots, \lambda_k^{*T}]^T$ is an equilibrium point of the ordinary differential Eq. (14a), and $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, \ldots, \dot{\theta}_k^{*T}]^T$ is the corresponding output, the following holds for all $i = 1, 2, \ldots, k$:

$$J_i g(-J_i^T \lambda_i^*) = v_d$$

$$\dot{\theta}_i^* = g(-J_i^T \lambda_i^*) \tag{19}$$

Defining new variables $\mu_i^* = -g(-J_i^T \lambda_i^*) - J_i^T \lambda_i^*$ for all $i = 1, 2, \ldots, k$, then $(\lambda_i^*, \mu_i^*, \dot{\theta}_i^*)$ is a solution to the equation set composed of Eqs. (9), (10) and (12). Due to the equivalence between the equation set consisting of (9), (10) and (12) and the equation set (6), it is concluded that $(\lambda_i^*, \mu_i^*, \dot{\theta}_i^*)$ is also a solution to the KKT condition (6). The KKT condition (6) gives a sufficient and necessary condition to the constrained quadratic programming problem (5) [32], so $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, \ldots, \dot{\theta}_k^{*T}]^T$ is the optimal solution to the problem (5). This completes the proof. □

5. Simulation results

In this section, the robot arm Puma 560 [36] is used as a testbed for the effectiveness of our method. The Puma 560 is a 6-DOF manipulator. The end-effector of the robot arm can reach any position

Table 1
Summary of the D-H parameters of the Puma 560 manipulator used in the simulation.

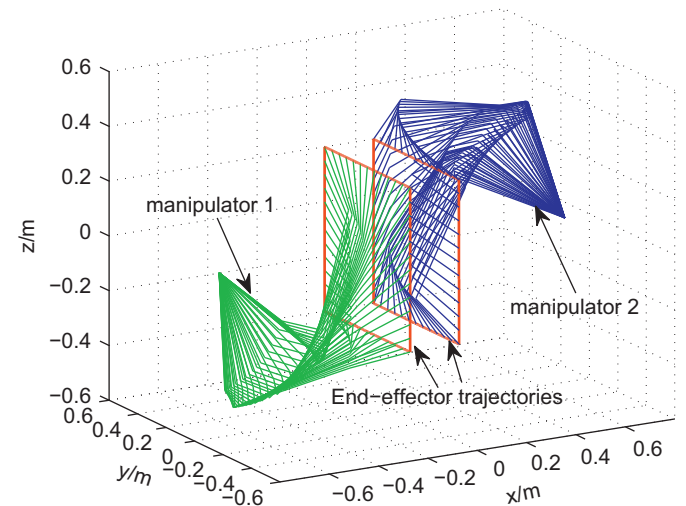| Link	a (m)	α (rad)	d (m)
1	0	$\pi/2$	0
2	0.43180	0	0
3	0.02030	$-\pi/2$	0.15005
4	0	$\pi/2$	0.43180
5	0	$-\pi/2$	0
6	0	0	0.30000

Fig. 1. Moving a payload with two manipulators in Section 5.1. In the figure, the green lines and the blue lines show the links of manipulator 1 and the links of manipulator 2, respectively, in every seconds. The red lines show the trajectory of the end-effectors of manipulator 1 and that of manipulator 2, respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

at a given orientation within its workspace. In the following simulations, only the position control problem in a three-dimensional space is considered, so the Puma 560 arm, which has 6-DOF, is a redundant manipulator to this particular problem. The D-H parameters of the Puma 560 manipulator used in the simulation are summarized in Table 1.
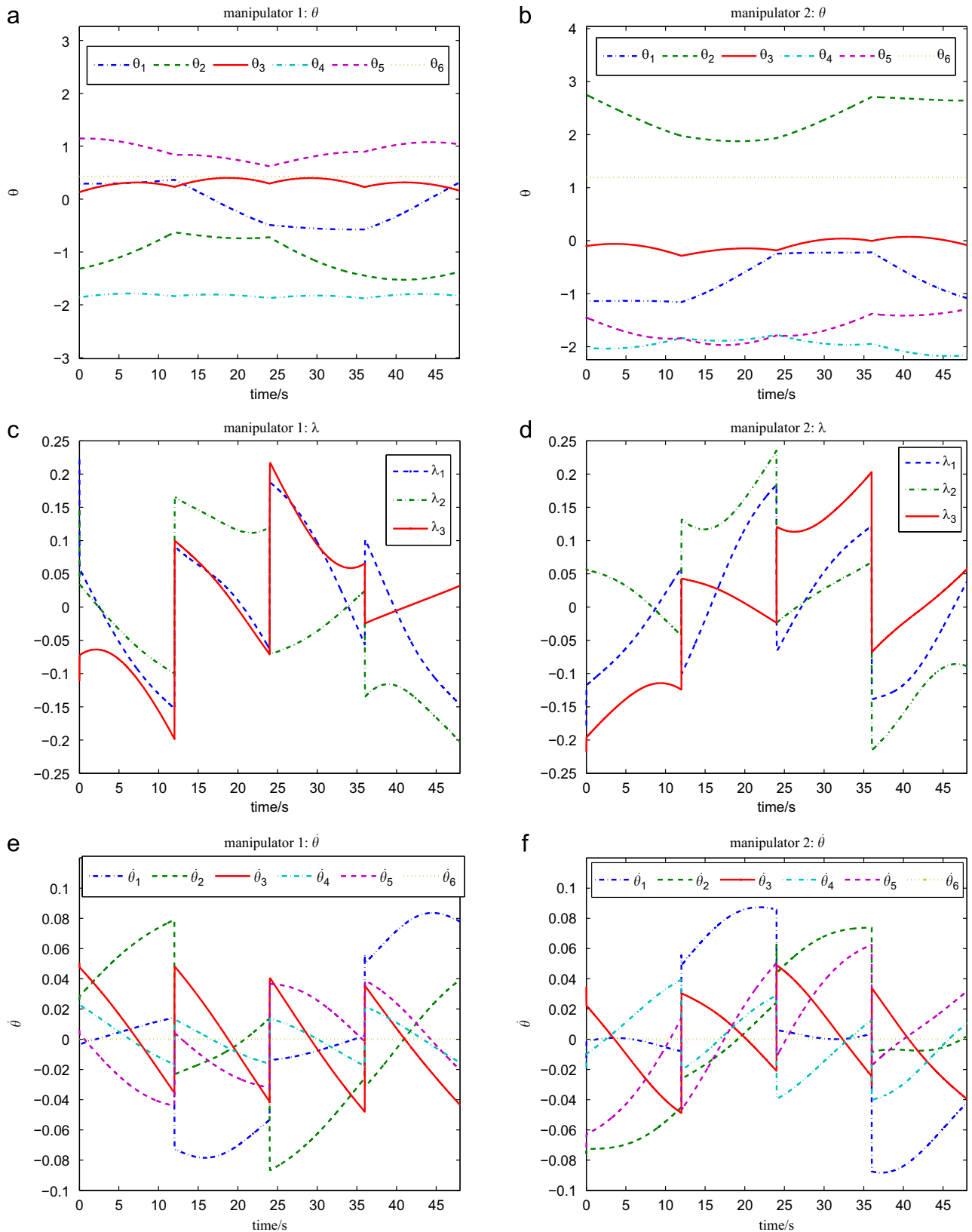

Fig. 2. Evolutions of θ, λ and $\dot{\theta}$ with time for the two manipulators in the simulation example in Section 5.1.

In this section, the proposed recurrent neural network model is applied to the kinematic control of multiple Puma 560 manipulators. Two simulations are performed: one studies cooperative transport with two manipulators and the other one studies cooperative tracking with three manipulators. In the simulations, only positioning of the reference point in three-dimensional space
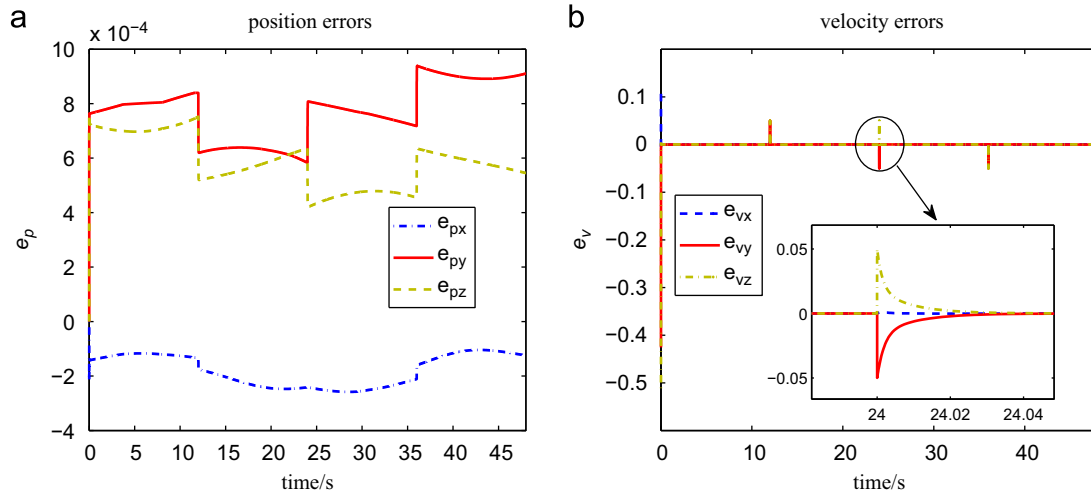

Fig. 3. Evolutions of position errors and velocity errors with time in the simulation example in Section 5.1.
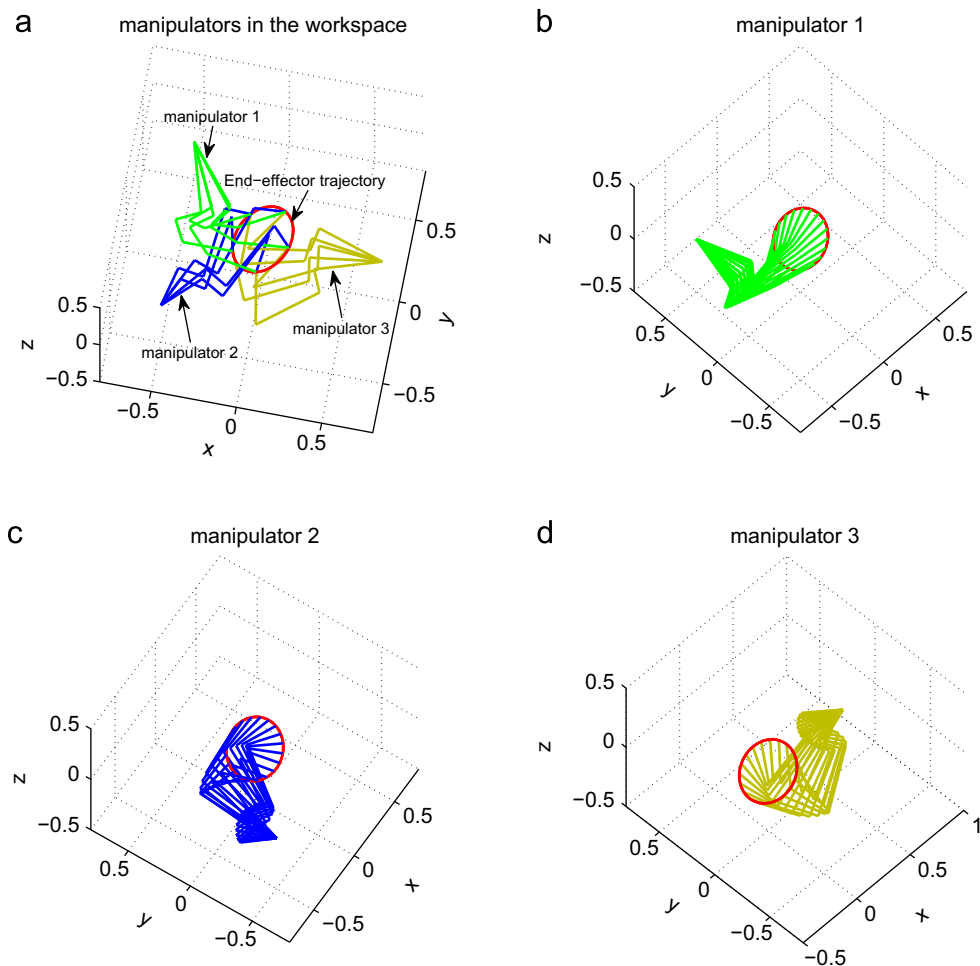

Fig. 4. Cooperative target tracking with three manipulators in Section 5.2 (only five configurations of each manipulator are plotted in (a) for easy recognition). In the figure, the green lines, the blue lines and the yellow lines show the links of manipulator 1, manipulator 2 and manipulator 3, respectively. The red lines show the trajectory of the end-effectors. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

are considered, so $n=3$. Each Puma 560 manipulator has 6 DOF ($m=6$), and therefore for the two simulations, the degree of redundancy are 6 and 9, respectively.

5.1. Payload transport with two manipulators

In this simulation example, we consider the problem of moving a payload along a square trajectory with a pair of manipulators. The D-H parameters of the two manipulators are summarized in Table 1. The bases of manipulator 1 and manipulator 2 locate at $[-0.7, 0, 0]$ m and $[0.7, 0, 0]$ m in the Cartesian space, respectively. The payload, centered at $[0, 0, 0]$ m, has two handles at $[-0.1, 0, 0]$ m and $[0.1, 0, 0]$ m, which will be held by manipulator 1 and manipulator 2 for the movement. The desired motion of the reference point (the center of the payload) is from $[0, 0.3, -0.3]$ m to $[0, 0.3, 0.3]$ m, from $[0, 0.3, 0.3]$ m to $[0, -0.3, 0.3]$ m, from $[0, -0.3, 0.3]$ m to $[0, -0.3, -0.3]$ m, and from $[0, -0.3, -0.3]$ m back to $[0, 0.3, -0.3]$ m consecutively along straight lines with a constant speed 0.05 m/s. As to the neural network parameters, we choose $\epsilon_1 = \epsilon_2 = 10^{-3}$ and the upper and

lower limits of the joint velocity are set to be $\eta^+ = [1, 1, 1, 1, 1, 1]^T$ and $\eta^- = -\eta^+$. Fig. 1 illustrates motion trajectories of the two Puma 560 manipulators. The corresponding profiles for θ, λ and $\dot{\theta}$ are shown in Fig. 2. Note that the desired trajectory is a square, which has a sharp turn at each corner, and hence λ and $\dot{\theta}$ for both manipulators have a sharp change in values at the time 12 s, 24 s and 36 s when the end-effector crosses the corners. To evaluate the control accuracy, the difference between the real position r_r and the desired position r_d of the reference point is used to measure the position error e_p and the difference between the real velocity v_r and the desired velocity v_d of the reference point is used to measure the velocity error e_v. That is, $e_p = r_r - r_d$ and $e_v = v_r - v_d$. According to the problem formulation, the real position of the reference point locates at the midpoint of the two end-effector positions and therefore the real velocity is the average of the two end-effector velocities, i.e., $r_r = (r_1 + r_2)/2$ and $v_r = (v_1 + v_2)/2$, where r_1, r_2, v_1, and v_2 represent the end-effector position of manipulator 1, the end-effector position of manipulator 2, the end-effector velocity of manipulator 1, and the end-effector velocity of manipulator 2, respectively. The evolution of the position error
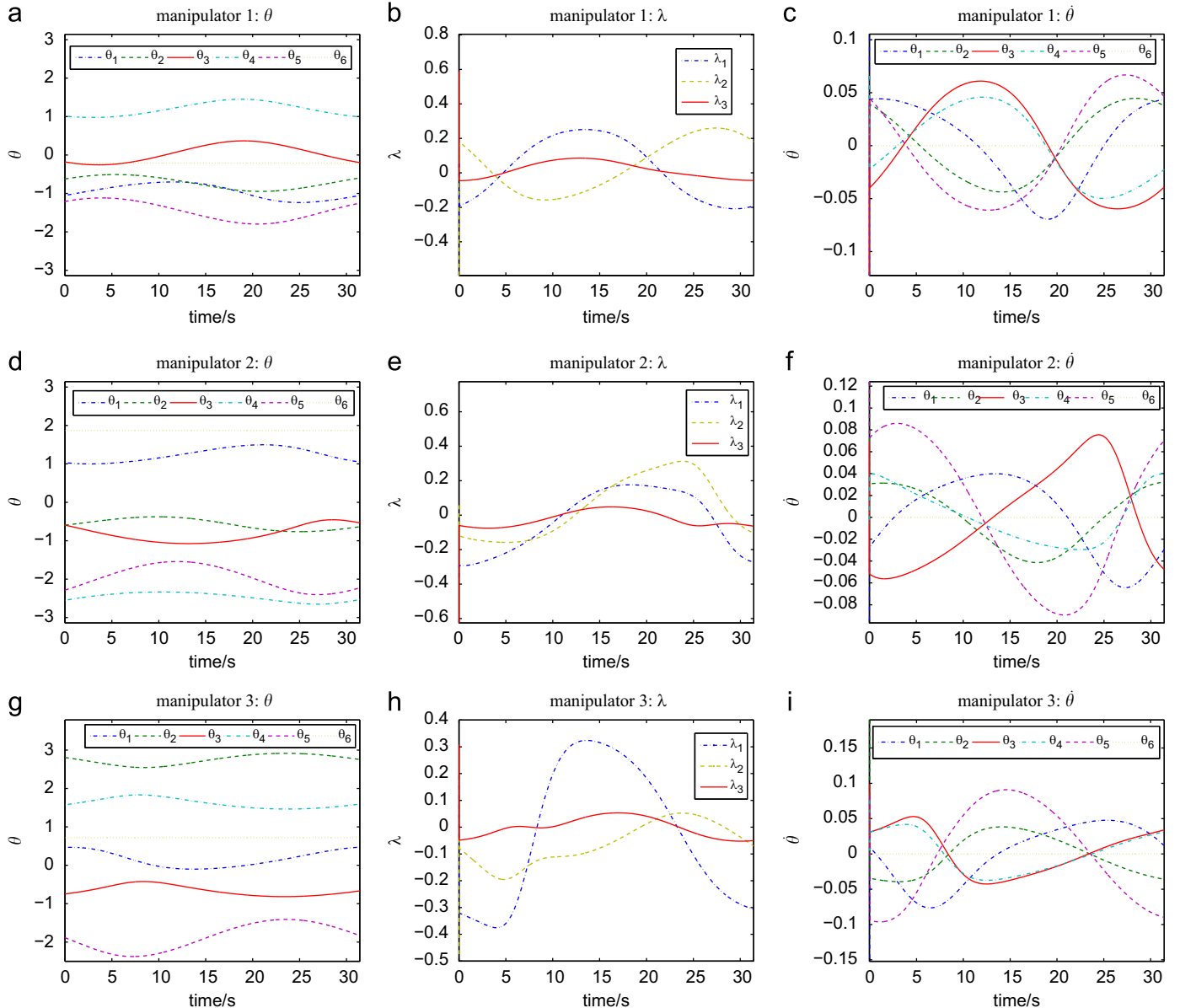

Fig. 5. Evolutions of θ, λ, $\dot{\theta}$ with time for the manipulators in the simulation example in Section 5.2.
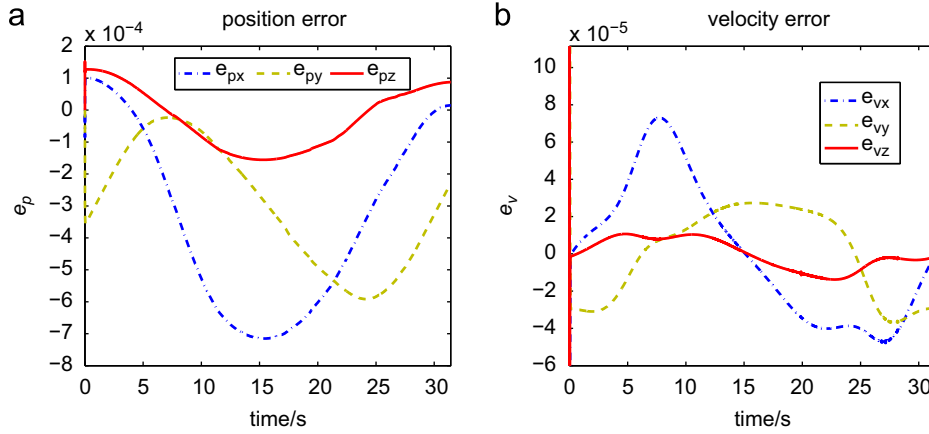
Fig. 6. Position errors and velocity errors for the cooperative target tracking in the simulation example in Section 5.2.

along x-axis (e_{px}), y-axis (e_{py}), z-axis (e_{pz}) and the evolution of the velocity error along x-axis (e_{vx}), y-axis (e_{vy}), z-axis (e_{vz}) are, respectively, plotted in Fig. 3. From the figure, it can be observed that the position error converges to less than 1 mm in the duration of the simulation after a short transient at the very beginning. The velocity error converges very fast every time after the end-effector crosses a corner of the desired square trajectory.

5.2. Cooperative tracking with three manipulators

In this simulation, we consider cooperative target tracking with three identical Puma 560 manipulators. As a particular example of cooperative task execution, potential applications include coordinate welding [4], dextrous grasping [3], etc. Simply speaking, the problem is to control the end-effectors of multiple manipulators to simultaneously track a desired trajectory. In this simulation, the bases of the three Puma 560 manipulators, manipulator 1, manipulator 2 and manipulator 3, locate at $[-0.5, 0.5, 0]$ m $[-0.5, -0.5, 0]$ m and $[0.7, 0, 0]$ m in the Cartesian space, respectively. The desired trajectory is a circle centered at $[0, 0, 0]$ m with diameter 0.4 m and a revolute angle about y-axis for $\pi/6$ rad. The starting position of the trajectory is $[0, 0, 0.2]$ m and the desired tracking speed is 0.04 m/s. We choose $\epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-3}$, $\eta^+ = [1, 1, 1, 1, 1, 1]^T$ and $\eta^- = -\eta^+$ for the neural network. Fig. 4 illustrates motion trajectories of the manipulators. From this figure, it can be observed that the end-effectors of the three manipulators track the desired circular trajectory simultaneously. The evolutions of θ, λ and $\dot{\theta}$ with time for the three manipulators are shown in Fig. 5. Since the desired trajectory is a circle, which is smooth, the value of θ, λ and $\dot{\theta}$ evolves smoothly with time. From Fig. 6, it can be seen that the position error, measured with the position difference between the center point of the three end-effectors and the desired point, and the velocity error, measured with the difference between the velocity of the center point of the three end-effectors and the desired velocity, converge to a small value very fast after a short transient at the very beginning. During the simulation period, both the position error and the velocity error are less than 8×10^{-4} m and 10×10^{-5} m/s, respectively, in the three axial directions.

6. Conclusions

In this paper, the problem of multiple redundant manipulator cooperative task execution is formulated as a quadratic programming problem. A recurrent neural network is proposed to tackle the problem. Conventionally, this problem is solved by calculating the inverse kinematics by pseudo-inverting the Jacobian matrices. However, this treatment, on one hand, is computationally intensive and on the other hand, it generally results in a centralized control law of manipulators. Inspired by the great success of recurrent neural networks in realtime control of single manipulators, the neural network methodology is extended to solve the multiple redundant manipulator cooperative task execution problem. Compared with existing schemes using neural networks for the motion control of manipulators, the proposed neural network is composed of independent modules, each of which controls a single manipulator and all of them together generate an emergent collective behavior to complete the cooperative task. Theoretical analysis proves the global stability of the neural network and the optimality of the solution. Application orientated simulations show that the proposed method is effective and accurate.

Some points are worth to mention about the proposed neural network approach to multiple redundant manipulator cooperative task execution.

- For multiple robot task execution, conventional methods first decompose the task into subtasks and assign each subtask to a robot [37]. Differently, this paper direct considers the problem from a global optimization level but still obtains a decentralized control law.
- As pointed out in Remark 1, the proposed framework admits joint limit constraints and collision avoidance constraints by imposing additional inequality constraints on $\dot{\theta}$.
- The proposed model has a comparably low architecture complexity (the interconnection complexity is $O(kn^2)$) as pointed out in Remark 3. Modules in the neural network have no dependence on each other and thus failure of a particular module has no impact on other modules.
- The proposed neural network does not require any off-line learning procedures and the designed parameters guarantee the optimality of the result to the problem.
- The proposed model is in nature a global optimization method and does not suffer from undesired local minima [17].

Acknowledgements

References

[1] G. Montemayor, J.T. Wen, Decentralized collaborative load transport by multiple robots, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), 2005, pp. 372–377.
[2] J. Fraile, C.J.J. Paredis, C. Wang, P.K. Khosla, Agent-based planning and control of a multi-manipulator assembly system, in: Proceedings of IEEE International Conference on Robotics and Automation, 1999, pp. 1219–1225.
[3] G. Liu, J. Xu, X. Wang, Z. Li, On quality functions for grasp synthesis, fixture planning, and coordinated manipulation, IEEE Trans. Autom. Sci. Eng. 1 (2) (2004) 146–162.
[4] L. Wu, K. Cui, S.B. Chen, Redundancy coordination of multiple robotic devices for welding through genetic algorithm, Robotica 18 (2000) 669–676.
[5] A. Liegeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, IEEE Trans. Syst. Man Cybern. 7 (12) (1977) 868–871.
[6] O. Khatib, A. Bowling, Optimization of the inertial and acceleration characteristics of manipulators, in: Proceedings of IEEE International Conference on Robotics and Automation, 1996, pp. 2883–2889.
[7] C.A. Klein, C.H. Huang, Review of pseudoinverse control for use with kinematically redundant manipulators, IEEE Trans. Syst. Man Cybern. 13 (3) (1983) 245–250.
[8] T. Shamir, Y. Yomdin, Repeatability of redundant manipulators: mathematical solution of the problem, IEEE Trans. Autom. Control 33 (11) (1988) 1004–1009.
[9] F.T. Cheng, R.J. Sheu, T.H. Chen, The improved compact QP method for resolving manipulator redundancy, IEEE Trans. Syst. Man Cybern. 25 (11) (1995) 1521–1530.
[10] R.C.O. Locke, R.V. Patel, Optimal remote center-of-motion location for robotics-assisted minimally-invasive surgery, in: IEEE International Conference on Robotics and Automation, 2007, pp. 1900–1905.
[11] S. Cocuzza, I. Pretto, S. Debei, Novel reaction control techniques for redundant space manipulators: theory and simulated microgravity tests, Acta Astronaut. 68 (11–12) (2011) 1712–1721.
[12] H. Ding, S.K. Tso, A fully neural-network-based planning scheme for torque minimization of redundant manipulators, IEEE Trans. Ind. Electron. 46 (1) (1999) 199–206.
[13] Y. Xia, J. Wang, A dual neural network for kinematic control of redundant robot manipulators, IEEE Trans. Syst. Man Cybern. B: Cybern. 31 (1) (2001) 147–154. (a publication of the IEEE Systems Man and Cybernetics Society).
[14] Y. Zhang, J. Wang, Y. Xia, A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits, IEEE Trans. Neural Networks 14 (3) (2003) 658–667.
[15] Y.S. Xia, G. Feng, J. Wang, A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control, IEEE Trans. Syst. Man Cybern. B: Cybern. 35 (1) (2005) 54–64.
[16] W.S. Tang, J. Wang, A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity, IEEE Trans. Syst. Man Cybern. B: Cybern. 31 (1) (2001) 98–105. (a publication of the IEEE Systems Man and Cybernetics Society).
[17] T.A. Lasky, S.H. Cha, S.A. Velinsky, Kinematic redundancy resolution for serial-parallel manipulators via local optimization including joint constraints, Mech. Based Des. Struct. Mach. 34 (2) (2006) 213–239.
[18] P. Patchaikani, L. Behera, G. Prasad, A single network adaptive critic based redundancy resolution scheme for robot manipulators, IEEE Trans. Ind. Electron. 99 (1) (2011).
[19] H.D. Taghirad, Y.B. Bedoustani, An analytic-iterative redundancy resolution scheme for cable-driven redundant parallel manipulators, IEEE Trans. Robotics 27 (6) (2011) 1137–1143.
[20] Y. Xia, G. Feng, J. Wang, A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations, Neural Networks 17 (7) (2004) 1003–1015.
[21] W. Wu, Neuro-fuzzy and model-based motion control for mobile manipulator among dynamic obstacles, Sci. China Ser. F: Inf. Sci. 46 (1) (2003) 14–30.
[22] X. Hu, Dynamic system methods for solving mixed linear matrix inequalities and linear vector inequalities and equalities, Appl. Math. Comput. 216 (4) (2010) 1181–1193.
[23] B. Liu, H. He, S. Chen, Adaptive dual network design for a class of SIMO systems with nonlinear time-variant uncertainties, Acta Autom. Sin. 36 (2010) 564–572.
[24] Y. Zhang, Y. Shi, K. Chen, C. Wang, Global exponential convergence and stability of gradient-based neural network for online matrix inversion, Appl. Math. Comput. 215 (3) (2009) 1301–1306.
[25] Y. Xia, J. Wang, A dual neural network for kinematic control of redundant robot manipulators, IEEE Trans. Syst. Man Cybern. B 31 (1) (2001) 147–154.
[26] S. Li, M.Q.H. Meng, W. Chen, SP-NN: a novel neural network approach for path planning, in: 2007 IEEE International Conference on Robotics and Biomimetics, 2007, pp. 1355–1360.
[27] B. Liu, H. He, D.W. Repperger, Two-time-scale online actor-critic paradigm driven by POMDP, in: 2010 International Conference on Networking, Sensing and Control (ICNSC), 2010, pp. 243–248.
[28] S. Liu, J. Wang, A simplified dual neural network for quadratic programming with its KWTA application, IEEE Trans. Neural Networks 17 (6) (2006) 1500–1510.
[29] Q. Liu, J. Wang, A one-layer dual recurrent neural network with a heaviside step activation function for linear programming with its linear assignment application, in: Proceedings of the 21st International Conference on Artificial Neural Networks—Volume Part II (ICANN 2011), Springer-Verlag, Berlin/Heidelberg, 2011, pp. 253–260.
[30] Y. Zhang, S.S. Ge, T.H. Lee, A unified quadratic programming based dynamical system approach to joint torque optimization of physically constrained redundant manipulators, IEEE Trans. Syst. Man Cybern. B: Cybern. 34 (5) (2004) 2126–2132.
[31] Y. Zhang, J. Wang, Obstacle avoidance for kinematically redundant manipulators using a dual neural network, IEEE Trans. Syst. Man Cybern. B: Cybern. 34 (1) (2004) 752–759. (a publication of the IEEE Systems Man and Cybernetics Society).
[32] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, USA, 2004.
[33] Y. Zhang, B. Cai, L. Zhang, K. Li, Bi-criteria velocity minimization of robot manipulators using a linear variational inequalities-based primal–dual neural network and Puma 560 example, Adv. Robotics 22 (13–14) (2008) 1479–1496.
[34] Y. Zhang, J. Wang, Y. Xu, A dual neural network for bi-criteria kinematic control of redundant manipulators, IEEE Trans. Robotics Autom. (2002) 923–931.
[35] H. Khalil, Nonlinear Systems, Prentice Hall, 2002.
[36] A.K. Morimoto, Dynamic Modeling of a Puma 560 for Use in Compliant Path Control, University of New Mexico, 1984.
[37] J.A. Shah, P.R. Conrad, B.C. Williams, Fast Distributed Multi-agent Plan Execution with Dynamic Task Assignment and Scheduling, 2009, pp. 289–296.

Shuai Li received the B.E. degree in precision mechanical engineering from Hefei University of Technology, China, and the M.E. degree in automatic control engineering from University of Science and Technology of China, China. Currently, he is pursing the Ph.D. degree in electrical engineering at Stevens Institute of Technology, USA. His research interests include dynamic neural networks, wireless sensor networks, robotic networks, machine learning and other dynamic problems defined on a graph. He is on the editorial board of International Journal of Distributed Sensor Networks.

Sanfeng Chen received the PH.D. degree from the university of science and technology of China, in 2008. She is an assistant professor at Shenzhen Key Lab of Visual Media Processing and Transmission, Shenzhen Institute of Information Technology, Shenzhen, China. Her research interests include artificial intelligence, signal processing and pattern recognition.

Bo Liu received the M.E. degree in computer engineering from Stevens Institute of Technology, USA. Currently, he is pursing the Ph.D. degree in computer science at University of Massachusetts, USA. His research interests include machine learning, data mining, graphical model, kernel approaches and stochastic optimization.

Yangming Li received the B.E. and M.E. degrees from Hefei University of Technology, China, in computer science, and the Ph.D. degree from University of Science and Technology of China, China, in automatic control engineering. He is currently a research associate in the Robot Sensor & Human–Machine Interaction Lab, Institute of Intelligent Machines, Chinese Academy of Sciences, China. His interests center on robotics, including machine perception, tracking, object classification, navigation, planning and map building.

Yongsheng Liang received the Ph.D. degree from Harbin Institute of Technology, Harbin, China, in 1999. He completed the post-doctoral research in Information and Communication Engineering Station of Harbin Institute of Technology, Harbin, China, in 2005. He is a professor at Shenzhen Key Lab of Visual Media Processing and Transmission, Shenzhen Institute of Information Technology, Shenzhen, China. His research interests include computer network and data communication, signal processing and pattern recognition.