

# Decentralized control of collaborative redundant manipulators with partial command coverage via locally connected recurrent neural networks

Shuai Li · Hongzhu Cui · Yangming Li ·  
Bo Liu · Yuesheng Lou

Received: 12 December 2011 / Accepted: 21 June 2012  
© Springer-Verlag London Limited 2012

**Abstract** This paper studies the decentralized control of multiple redundant manipulators for the cooperative task execution problem. Different from existing work with assumptions that all manipulators are accessible to the command signal, we propose in this paper a novel strategy capable of solving the problem even though there exists some manipulators unable to access the command signal directly. The cooperative task execution problem can be formulated as a constrained quadratic programming problem. We start analysis by re-designing the control law proposed in (Li et al. Neurocomputing, 2012), which solves the optimization problem recursively. By replacing the command signal with estimations with neighbor information, the control law becomes to work in the partial command coverage situation. However, the stability and optimality of

the new system are not necessarily the same as the original system. We then prove in theory that the system indeed also globally stabilizes to the optimal solution of the constrained quadratic optimization problem. Simulations demonstrate the effectiveness of the proposed method.

**Keywords** Recurrent neural network · Quadratic programming · Cooperative task execution · Redundant manipulator · Decentralized control · Hierarchical tree

## 1 Introduction

With the development of mechanics, electronics, computer engineering, etc, the costs of manipulators are reducing. This tendency cast lights to use a collection of manipulators to perform a complicated task, which was often performed by special purpose machinery. Beyond the cost consideration, another explicit advantage of using multiple manipulators instead of a single special purpose machinery for task execution lies that the multiple manipulator system is reconfigurable and can also be used for other applications while the special purpose machinery cannot adapt to other applications. Due to this reason, considerable research attentions are paid to multiple manipulator cooperations and find applications in robotics, such as load transport [2], cooperative assembly [3], dextrous grasping [4], coordinate welding [5], etc., are becoming increasingly popular, and have received considerable studies. Among the large variety of manipulators, redundant manipulator, which have more degree of freedom (DOF) than required, can be used for task execution with the exploitation of the extra DOF for optimization of other criteria, such as velocity optimization, obstacle avoidance, etc. and thus has strong potential in dextrous cooperation of multiple manipulators.

---

S. Li (✉)  
Department of Electrical and Computer Engineering,  
Stevens Institute of Technology, Hoboken, NJ 07030, USA  
e-mail: lishuai8@gmail.com

H. Cui  
School of Nursing, Sun Yat-Sen University,  
Guangzhou, Guangdong, China

Y. Li  
Robot Sensor and Human-machine Interaction Lab,  
Institute of Intelligent Machines, Chinese Academy of Sciences,  
Hefei, Anhui, China

B. Liu  
Department of Computer Science,  
University of Massachusetts, Amherst, MA 01003, USA

Y. Lou  
School of Mechatronics and Information,  
Yiwu Industrial and Commercial College,  
Yiwu, Zhejiang, China

In [6], the authors investigated the cooperation problem in the Martha Project with multiple mobile robots. For this method, a central station is used to decompose the task and then the task is assigned to all robots. Therefore, this method essentially is a centralized method and may encounter difficulties when the collection includes a large number of robots since the computation and communication load of the central station will increase exponentially for such a non-scalable method. In [7], an optimization-based framework is proposed for optimal multiple robot cooperation. The problem is formulated as an optimization problem while the solution of this problem may need global information and thus the method is a centralized one. In [8], the multiple impedance control strategy is applied to the coordination of multiple manipulators on a space free-flyer. As the manipulators locate on the same free-flyer, the information exchange between manipulators is not a big issue. A PD-like controller is developed in [9] to control the coordination of multiple under-actuated manipulators. The stability of the control is proven in theory. Most existing works on multiple manipulator control consider non-redundant manipulators. Compared with redundant ones, there is no extra design freedom for optimization purpose. Also, most works assume the communication network is all-to-all connected and each manipulator can use information of all manipulators in the network. This assumption is very restrictive for a large scale manipulator network and the control strategy often results in a huge communication burden in this case and may lead to network congestion or even paralysis.

In this paper, we consider the case with redundant manipulators for multiple manipulator task execution. The use of redundant manipulators left extra design freedom for optimization of the trajectory in certain sense. we formulate the cooperative task execution problem as a separable constrained quadratic programming problem and relax the original control law obtained by solving this optimization problem iteratively to a new one without requiring a full command coverage. The proposed control law is essentially a recurrent neuro-controller and thus has the same real-time performance as other neuro-controllers [10–15]. Also, the proposed strategy is a distributed one: each manipulator in the network only needs to get information from its neighbors in the communication network for control, which makes this method scalable to a network involving a large number of manipulators.

The remainder of this paper is organized as follows. In Sect. 2, some preliminaries on redundant manipulators and graph theory are introduced. In Sect. 3, the multiple manipulator cooperative task execution problem is formulated as a constrained quadratic programming problem and a distributed neural network with partial command coverage is presented for the control. In Sect. 4, the global stability of the proposed neural network and the optimality of

the neural solution are proven in theory. In Sect. 5, simulations are performed to validate the effectiveness of the proposed method. Section 6 concludes this paper.

## 2 Preliminaries

In this section, we describe the problem of cooperative task execution with multiple redundant manipulators. We first give a brief introduction to the redundant manipulator kinematics and then, based on this, we formulate the multiple manipulator task execution problem as a constrained quadratic programming problem.

### 2.1 Redundant manipulator

For a  $m$ -DOF redundant manipulator working in a  $n$ -dimensional Cartesian space ( $m > n$ ), we have the following nonlinear function:

$$r(t) = f(\theta(t)) \quad (1)$$

where  $r(t) \in \mathbb{R}^n$  and  $\theta(t) \in \mathbb{R}^m$  are the coordinate of the manipulator in the Cartesian space at time  $t$  and the coordinate in the joint space, respectively. The velocity of the manipulator in the joint space, which is  $\dot{\theta}(t)$ , and that in the Cartesian space, which is  $\dot{r}(t)$ , have the following relation,

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

where  $J(\theta(t)) = \frac{\partial f(\theta(t))}{\partial \theta(t)}$  is the Jacobian matrix.

### 2.2 Graph theory

A graph  $G(V, E)$  is denoted by  $(V, E)$ , where  $V$  is the set of nodes,  $E$  is the set of edges with  $E \subseteq V \times V$ . A path in a graph is a sequence of vertices such that from each of its vertices, there is an edge to the next vertex in the sequence. A tree is a graph in which any two vertices are connected by exactly one simple path. A rooted tree is a tree in which a special node is singled out. This node is called the “root”. The depth of a node is the length of the path from the root to the node. In a rooted tree, vertex  $v_1$  is the parent of vertex  $v_2$  if  $v_1$  immediately precedes  $v_2$  on the path from the root to  $v_2$ . Vertex  $v_2$  is the child of  $v_1$  if and only if  $v_2$  is a child of  $v_1$ .

## 3 Problem formulation and the solution

### 3.1 Multiple manipulator cooperation

The goal of multiple manipulator cooperation, for example in the application of cooperative payload transportation, is to move the payload along a desired reference trajectory and simultaneously maintain the relative positions of the

end effectors. This task involves two aspects: first, the reference point of the payload tracks the desired trajectory. Second, the end effectors maintain the original formation in space, that is, end effectors have the same velocity as the reference point. Such a multiple manipulator cooperation problem can be formulated as the following constrained optimization problem [1]:

$$\text{minimize} \quad \frac{1}{2} \sum_{i=1}^k \dot{\theta}_i^T \dot{\theta}_i \tag{3a}$$

$$\text{subject to} \quad J_1 \dot{\theta}_1 = v_d, J_2 \dot{\theta}_2 = v_d, \dots, J_k \dot{\theta}_k = v_d \tag{3b}$$

$$\eta^- \leq \dot{\theta}_1 \leq \eta^+, \eta^- \leq \dot{\theta}_2 \leq \eta^+, \dots, \eta^- \leq \dot{\theta}_k \leq \eta^+ \tag{3c}$$

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  $i = 1, 2, \dots, k$ , respectively.  $J_i \in \mathbb{R}^{n \times m}$  is the Jacobian matrix of the  $i$ th manipulator for  $i = 1, 2, \dots, k$ .  $\eta^+ \in \mathbb{R}^m$  and  $\eta^- \in \mathbb{R}^m$  are the upper and lower limit of the allowed velocity in the joint space.

Note that the optimization problem formulation (3) minimizes the Euclidean norm squared of the joint velocities (3a) (corresponds to the minimum kinematic energy) within the joint velocity limits (3c) and with the end effector moving at the velocity  $v_d$  (3b).

Our previous work [1] presents a dynamic neural network with  $k$  independent modules converges to the solution of the optimal problem (3). The  $i$ th module of the dynamic neural network has the following dynamics,

$$\text{state equation} \quad \epsilon \dot{\lambda}_i = J_i g(-J_i^T \lambda_i) - v_d \tag{4a}$$

$$\text{output equation} \quad \dot{\theta}_i = g(-J_i^T \lambda_i) \tag{4b}$$

where  $\epsilon > 0$  is a scaling factor,  $g(x) = [g_1(x_1), g_2(x_2), \dots, g_m(x_m)]^T$  for  $x = [x_1, x_2, \dots, x_m]^T$  and  $g_j(x_j)$  to be of the following form for  $j = 1, 2, \dots, m$ ,

$$g_j(x_j) = \begin{cases} \eta_j^+ & \text{if } x_j > \eta_j^+ \\ x_j & \text{if } \eta_j^- \leq x_j \leq \eta_j^+ \\ \eta_j^- & \text{if } x_j < \eta_j^- \end{cases} \tag{5}$$

where  $\eta_j^-$  and  $\eta_j^+$  are the  $j$ th elements of  $\eta^-$  and  $\eta^+$ , respectively.

The neural network (4) has  $k$  modules. Each module can be implemented in hardware and mounted on the associated manipulator to serve as its motion controller. A prominent advantage of this neural model is that every module is self-autonomous and evolves locally without any dependence on the state of other manipulators, but all the modules all together are still able to collaboratively solve the global optimization problem (3). However, this neural

network requires that all modules are accessible to the desired reference velocity command (i.e., this neural network requires a full command coverage), since  $v_d$  appears in the dynamic Eq. (4) of the  $i$ th module for all possible  $i$ . It is necessary for the command center to broadcast the desired reference velocity to all modules distributed in different places. This neural network is not applicable to applications without direct signal connections from the command center to all manipulators. In this paper, we present a new dynamic neural network model, which is redesigned based on (4) to remove the requirement of broadcasting command to all manipulators.

### 3.2 Inspirations from social behaviors in nature

Let us image a queue of geese in migration. When flying in the V formation, it appears that one goose is the leader, first in the formation and all the other geese follow the leader to maintain the formation. For many geese in such a group with a long distance from the leader goose, the leader goose may be out of the vision of them shaded by other geese in their front. Actually, as observed and verified by simulation, the flock of geese can reach formation and reach the flying speed and direct led by the leader by adapting its flying speed and flying direction based on its observation of its direct neighbors instead of following the leader goose. This phenomena inspires us to redesign the neural network module (4) to a new one with  $v_d$  in the equation replaced by  $J_j g(-J_j^T \lambda_j)$  with  $j \in \mathcal{N}(i)$  for  $i$  outside the coverage of  $v_d$ . This intuition provides a mechanism for the propagation of  $v_d$  from the command center to those manipulators with direct access to  $v_d$  and then from them to their neighbors in communication and from neighbors to neighbor until all manipulators receive the signal. To avoid loops of information flow and to guarantee the stability of the whole network, we then present the dynamic neural network formally and prove the convergence, the optimality of the neural solution rigorously.

### 3.3 Hierarchical topology generation

The proposed neural network is a locally interconnected network in topology and is a dynamic system for each module in the neural network. In this subsection, we present the procedure to construct the neural network topology based on the communication network structure.

For a strict presentation, we first define the communication graph  $\mathcal{G}_1(N_1, E_1)$  consisting of the node set  $N_1$ , which includes all manipulators and the command center, and the edge set  $E_1$ , which includes all communication connections between manipulators and the communication links from the command center to the directly connected

**Algorithm 1** Construction of the tree  $\mathcal{G}_2(N_2, E_2)$

**Require:**

- The communication graph  $\mathcal{G}_1(N_1, E_1)$
- The root node  $n_r$  (the one associated with the command center)

**Ensure:**

The resulting graph  $\mathcal{G}_2(N_2, E_2)$  is a tree rooted at the command center and spanning the graph  $\mathcal{G}_1(N_1, E_1)$ .

$$N_2 \leftarrow \{n_r\}$$

$$E_2 \leftarrow \phi$$

1: **repeat**

2:  $S_1 \leftarrow \mathcal{N}(N_2, N_1) - N_2$  ( $\mathcal{N}(N_2, N_1)$  denotes the neighbor set of the set  $N_2$  in the set  $N_1$ )

3:  $S_2 \leftarrow$  the edges in  $E_1$  bridging the set  $S_1$  and the set  $N_2$

4:  $N_2 \leftarrow N_2 \cup S_1$

5:  $E_2 \leftarrow E_2 \cup S_1$

6: **until**  $N_2 = N_1$

manipulators. For a node in the graph  $\mathcal{G}_1(N_1, E_1)$ , there is no direct parent node nor direct child node defined, and therefore, there is no explicit hierarchy for this manipulator network. In order to form a hierarchical structure, where each node has a unique parent node from whom an estimation of the reference velocity can be obtained, we first define a command-center-rooted spanning tree  $\mathcal{G}_2(N_2, E_2)$  generated from the graph  $\mathcal{G}_1(N_1, E_1)$ . We use a recursive procedure to construct the tree  $\mathcal{G}_2(N_2, E_2)$  as summarized in Algorithm 1 in pseudo code. Hierarchy appears in the

rooted tree  $\mathcal{G}_2(N_2, E_2)$  with each manipulator-associated node having a unique parent. A typical rooted tree structure is depicted in Fig. 1, where the arrows indicate the direction of information flow and the numbers aside the manipulators denote the depth of the node in the tree.

3.4 Neural dynamics

In this section, we present the dynamics of the neural network.

As the neural modules neighboring the command center can directly access the information of  $v_d$ , we simply use the same dynamics as (4) for them. While for those neural modules, which are outside the neighborhood of the command center, the estimation of the desired velocity is used as the reference signal. Concretely, we have the following dynamic equations for the neural network,

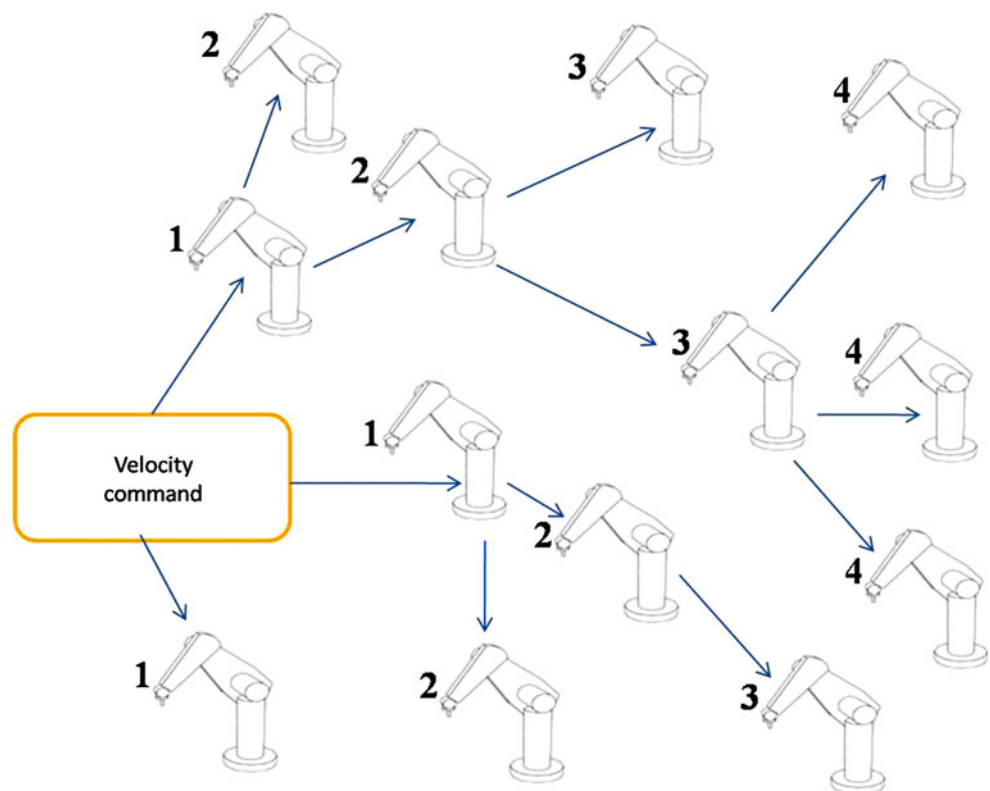
state equation

$$\epsilon_i \dot{\lambda}_i = \begin{cases} J_i g(-J_i^T \lambda_i) - v_d & \text{if } i \in \mathcal{C} \\ J_i g(-J_i^T \lambda_i) - J_{p(i)} g(-J_{p(i)}^T \lambda_{p(i)}) & \text{if } i \notin \mathcal{C} \end{cases} \tag{6a}$$

output equation  $\dot{\theta}_i = g(-J_i^T \lambda_i)$  (6b)

where the function  $p(i)$  is a mapping from  $i$ , which is the label of the  $i$ th manipulator, to the label of its parent,  $\mathcal{C}$  is the command set, whose elements are children of the root in the tree  $\mathcal{G}_2(N_2, E_2)$ .

**Fig. 1** The generated hierarchical topology from the communication network of the multiple redundant manipulator system, where the number aside the manipulators denotes the depth of the node in the rooted tree generated by the communication network



*Remark 1* For manipulators in the command set  $\mathcal{C}$ , the associated neural module dynamic is identical to that of (4). Equation (6) implies that  $J_i g(-J_i^T \lambda_i) = v_d$  at the equilibrium point for  $i \in \mathcal{C}$ . Accordingly, the value of  $J_i g(-J_i^T \lambda_i)$  can be regarded as an estimation of  $v_d$  if the whole neural network is convergent and therefore used by the neighboring manipulators  $j \notin \mathcal{C}$  as the desired reference velocity signal. However, the dynamic interaction between neural modules bring challenges to the convergence analysis, which is crucial for the neural dynamics to solve the optimization problem (3).

#### 4 Theoretical results

In this section, we present theoretical results on the optimality and convergence of the proposed neural network (6).

Before stating the main results, we first introduce the following lemmas, which are useful for the derivation of the main theoretical results.

**Theorem 1** *Let  $[\lambda_1^{*T}, \lambda_2^{*T}, \dots, \lambda_k^{*T}]$  be an equilibrium point of the neural network dynamic (4a). The output of this neural network at  $[\lambda_1^{*T}, \lambda_2^{*T}, \dots, \lambda_k^{*T}]^T$ , which is  $[\hat{\theta}_1^{*T}, \hat{\theta}_2^{*T}, \dots, \hat{\theta}_k^{*T}]^T = [g^T(-J_1^T \lambda_1^*), g^T(-J_2^T \lambda_2^*), \dots, g^T(-J_k^T \lambda_k^*)]^T$ , is the optimal solution to the cooperative task execution problem (3).*

*Proof* The statement of this theorem is exactly the same as the one in reference [1]. It is notably that the neural dynamic of (6) and that in [1] are different and thereby the proofs are not the same. However, noticing that the equilibrium  $[\lambda_1^{*T}, \lambda_2^{*T}, \dots, \lambda_k^{*T}]^T$  of (6) satisfies

$$J_i g(-J_i^T \lambda_i^*) = v_d \tag{7}$$

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

which is exactly the same as the equilibrium points of the neural network investigated in [1]. Using the KKT conditions, we are able to finally prove that this set of equilibrium is the optimal solution to (3) by following a similar argument as in Theorem 2 in [1].

Theorem 1 reveals that the equilibrium point of the recurrent neural network (6) corresponds to the optimal solution to the cooperative task execution problem. In the next step, we are going to show that the neural network is also convergent, which implies that the neural network converges to the optimal solution of (3) with any initialization of the state variables. On this point, we have the following theorem,

**Theorem 2** *The recurrent neural network (6) with  $\epsilon > 0$  is globally asymptotically stable and converges to the*

*optimal solution of the cooperative task execution problem (3).*

*Proof* As we have proved in Theorem 1 that the optimal solution of (3) is identical to the equilibrium point of the neural network (6), we only need to prove the globally asymptotical stability of the neural network to its equilibrium point. We prove that by induction. As the neural network has a hierarchical structure, we analyze the neuron layer by layer in the structure.

We denote  $d(i)$  as the depth of the  $i$ th neuron. In the network, represented by the graph  $\mathcal{G}_2(N_2, E_2)$  constructed in Algorithm 1, the root node corresponds to the command center has  $d = 0$  and all the other nodes have a depth larger than or equal to 1. We first prove that the globally asymptotical stability to the equilibrium point of the neurons in  $\mathcal{G}_2(N_2, E_2)$  with  $d = 1$ .

For the neuron  $i$  with  $d(i) = 1$ , its parent node is the node corresponding to the command center and therefore the neuron  $i$  belongs to the command set  $\mathcal{C}$ . According to the neural dynamic equation (6), its dynamic is governed by

$$\epsilon \dot{\lambda}_i = J_i g(-J_i^T \lambda_i) - v_d \tag{8}$$

This neuron is self-autonomous and has no dependence on others in dynamics. To prove the asymptotically stability of this neuron, we construct the following Lyapunov function,

$$V = -\lambda_i^T J_i g(-J_i^T \lambda_i) - \frac{1}{2} \|g(-J_i^T \lambda_i)\|^2 + v_d^T \lambda_i \tag{9}$$

$$= \frac{1}{2} (\|J_i^T \lambda_i\|^2 - \|-J_i^T \lambda_i - g(-J_i^T \lambda_i)\|^2) + v_d^T \lambda_i$$

Note that  $\|-J_i^T \lambda_i - g(-J_i^T \lambda_i)\|$  can be regarded as the distance from the vector  $-J_i^T \lambda_i$  to the box set bounded by the upper and lower limits of the function  $g(\cdot)$ . Based on the derivative rules for the projection functions [16, 17], we have

$$\nabla_{\lambda_i} \frac{1}{2} \|-J_i^T \lambda_i - g(-J_i^T \lambda_i)\|^2 = J_i (J_i^T \lambda_i + g(-J_i^T \lambda_i)) \tag{10}$$

Therefore, we get

$$\nabla_{\lambda_i} \frac{1}{2} (\|J_i^T \lambda_i\|^2 - \|-J_i^T \lambda_i - g(-J_i^T \lambda_i)\|^2)$$

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

$$= -J_i g(-J_i^T \lambda_i) \tag{11}$$

Accordingly,

$$\nabla_{\lambda_i} V = -J_i g(-J_i^T \lambda_i) + v_d \tag{12}$$

Clearly, the Hessian matrix of  $V$  with respect to  $\lambda_i$  is,

$$\nabla_{\lambda_i}^2 V = J_i g' J_i^T \tag{13}$$

where  $g' = \frac{\partial g(-J_i^T \lambda_i)}{\partial (-J_i^T \lambda_i)}$ . According to the definition of  $g(\cdot)$ , we know that  $g'$  is a diagonal matrix with the diagonal elements either 1 or 0. Thus, we conclude  $g'$  is semi-positive definite and  $\nabla_{\lambda_i}^2 V$  is also semi-positive definite. Therefore, the function  $V(\lambda_i)$  is convex and we have the following inequality according to the property of convex functions,

$$V(\lambda_i) - V(\lambda_i^*) \geq \nabla^T V(\lambda_i^*)(\lambda_i - \lambda_i^*) \tag{14}$$

where  $\lambda_i^*$  denotes the equilibrium point, which satisfies  $J_i g(-J_i^T \lambda_i^*) - v_d = 0$ . Substituting (12) and  $v_d = J_i g(-J_i^T \lambda_i^*)$  into (14) yields,

$$\begin{aligned} V(\lambda_i) &\geq V(\lambda_i^*) - (\lambda_i - \lambda_i^*)^T (J_i g(-J_i^T \lambda_i) - J_i g(-J_i^T \lambda_i^*)) \\ &= V(\lambda_i^*) + ((-J_i^T \lambda_i) - (-J_i^T \lambda_i^*))^T \\ &\quad (g(-J_i^T \lambda_i) - g(-J_i^T \lambda_i^*)) \end{aligned} \tag{15}$$

Notice that the function  $g(\cdot)$  is an increasing function (not strictly increasing as it saturates when the entry goes to infinity); therefore, we have,

$$((-J_i^T \lambda_i) - (-J_i^T \lambda_i^*))^T (g(-J_i^T \lambda_i) - g(-J_i^T \lambda_i^*)) \geq 0 \tag{16}$$

Together with (15), we obtain,

$$V(\lambda_i) \geq V(\lambda_i^*) \tag{17}$$

which means that the function  $V$  is lower bounded and therefore in turn validates that  $V$  is indeed qualified to be a Lyapunov function.

Calculating time derivative of  $V$  along the trajectory of (8) yields,

$$\begin{aligned} \dot{V} &= \dot{\lambda}_i^T (-J_i g(-J_i^T \lambda_i) + v_d) \\ &= -\frac{1}{\epsilon} \|J_i g(-J_i^T \lambda_i) - v_d\|^2 \\ &\leq 0 \end{aligned} \tag{18}$$

Let  $\dot{V} = 0$ , we find  $J_i g(-J_i^T \lambda_i) - v_d = 0$ , which corresponds to the equilibrium point of  $\lambda_i$ . According to Lasalle's invariant set principle [18], the dynamic asymptotically evolves to the equilibrium point.

We have completed the analysis for  $d = 1$ . Now, we assume that the neurons with the depth  $d = l$  in the graph  $\mathcal{G}_2(N_2, E_2)$ , say the  $j$ th one, asymptotically stabilizes to the equilibrium  $J_j g(-J_j^T \lambda_j) - v_d = 0$ . We aim to prove the same result also hold for neurons with the depth  $d = l + 1$ . We consider a particular neuron with the depth of  $d = l + 1$ . We denote this neuron the  $q$ th one and denote  $r = p(q)$ , meaning that the  $r$ th neuron is the parent node of the  $q$ th neuron. As we assumed that neurons with the depth of  $d = l$  are asymptotically stable and the  $r$ th neuron, as the parent node of the  $q$ th one, has the depth  $d = l$ , we

**Table 1** Summary of the D–H parameters of the Puma 560 manipulator used in the simulation

link	$a(m)$	$\alpha(rad)$	$d(m)$
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

conclude that the  $r$ th neuron is asymptotically stable to  $J_r g(-J_r^T \lambda_r) - v_d = 0$ . The dynamic of the  $q$ th neuron for  $q > 1$  writes,

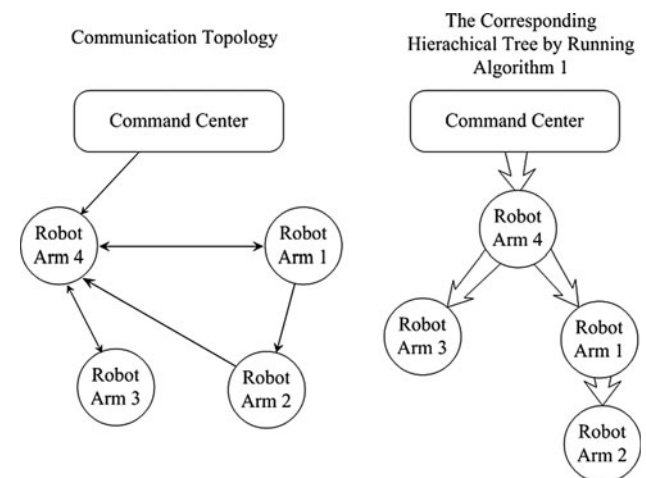
$$\begin{aligned} \epsilon \dot{\lambda}_q &= J_q g(-J_q^T \lambda_q) - J_r g(-J_r^T \lambda_r) \\ &= J_q g(-J_q^T \lambda_q) - v_d + v_d - J_r g(-J_r^T \lambda_r) \end{aligned} \tag{19}$$

Define  $u = v_d - J_r g(-J_r^T \lambda_r)$ . The system (19) can be regarded the following system with input,

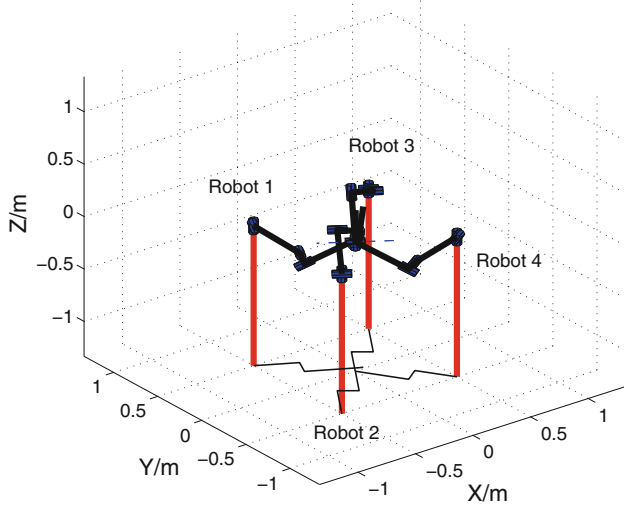
$$\epsilon \dot{\lambda}_q = J_q g(-J_q^T \lambda_q) - v_d + u \tag{20}$$

where  $u$  gets input from the output of the  $r$ th neuron. It attenuates to zero as time elapses. Compared with the dynamic equation (8), the only difference is the presence of the extra input term  $u$  in (20). Following a similar way in the asymptotical stability analysis for  $d = 1$ , we are able to prove that the system (20) is ISS (input-to-state stable). According to the property of serially connected ISS systems [18], we conclude that the  $q$ th neuron asymptotically stabilizes to  $J_q g(-J_q^T \lambda_q) - v_d = 0$ , which is the equilibrium point.

Until now, we have shown that the asymptotical stability to the equilibrium points for the neurons with the depth  $d = l$  implies such a property for neurons with the depth



**Fig. 2** The structure of the communication network and the structure of the generated hierarchical tree by running Algorithm 1 for the four robot arm collaboration example



**Fig. 3** The initial configuration of all robot arms

$d = l + 1$ . Therefore, we conclude that this property holds for all neurons in the neural network. Recalling that the equilibrium point is identical to the optimal solution of (3), we completes the proof.

**5 Simulation results**

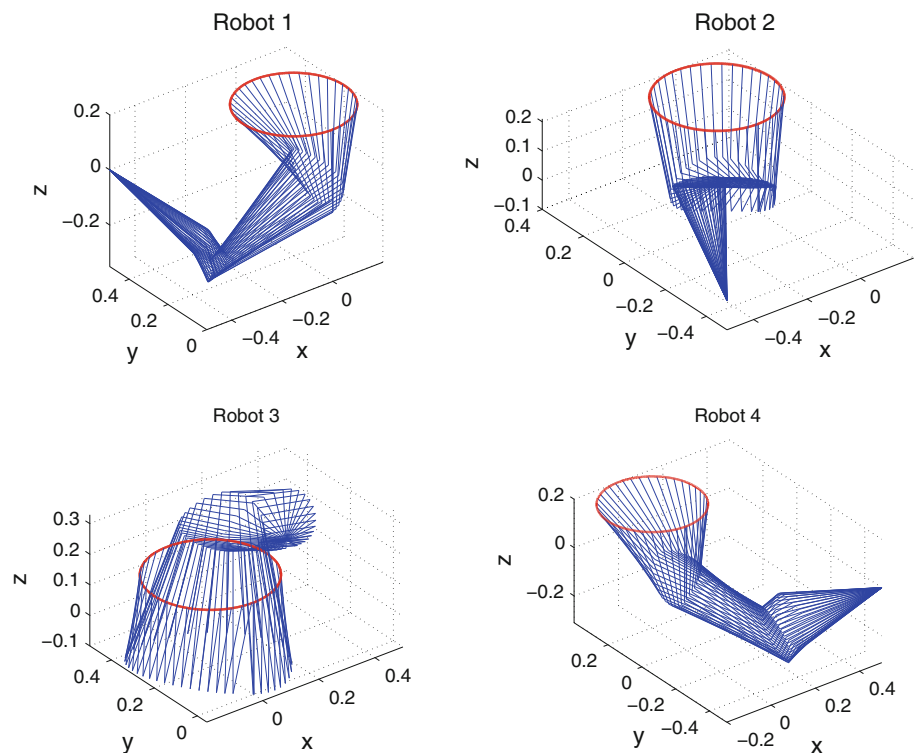
The same as in [1], we use the robot arm Puma 560 as a testbed for the effectiveness of our method. The Puma 560 has 6-DOFs. For the task only considering the end

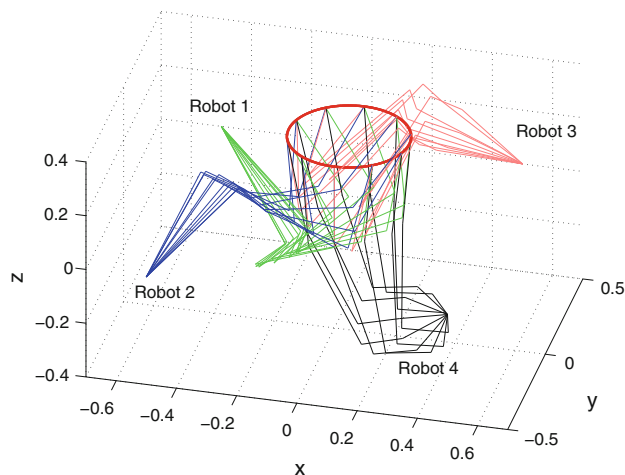
effector’s position in 3-D space, without taking its pose into consideration, the Puma 560 robot arm is a redundant manipulator with redundancy 3. The D-H parameters of the Puma 560 manipulator are summarized in Table 1.

In this section, we apply the proposed recurrent neural network model to the cooperative tracking problem with four Puma 560 manipulators. In the simulations, only positioning of the reference point in three-dimensional space is considered, so  $n = 3$ . Each Puma 560 manipulator has 6 DOF ( $m = 6$ ), and therefore for the simulation, the degree of redundancy are 12 in total for the four manipulators. In this simulation, we consider cooperative target tracking with four identical Puma 560 manipulators. The goal is to control the end effectors of multiple manipulators to simultaneously track a desired trajectory. In this simulation, the bases of the four Puma 560 manipulators, Robot 1, Robot 2, Robot 3, and Robot 4, whose shoulder joints locate at  $[-0.5, 0.5, 0]m$ ,  $[-0.5, -0.5, 0]m$ ,  $[0.5, 0.5, 0]m$ , and  $[0.5, -0.5, 0]$  in the Cartesian space, respectively (as shown in Fig. 3). The desired trajectory is a circle centered at  $[0, 0, 0]m$  with diameter 0.4m 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.

Among all the four manipulators, only the 4th one is directly accessible to the velocity command. As such, the control strategy proposed in [1] is not applicable since the method requires all manipulators accessible to the command signal. For other manipulators in the system, they

**Fig. 4** The trajectories of all manipulators in the workspace (drawn separately)

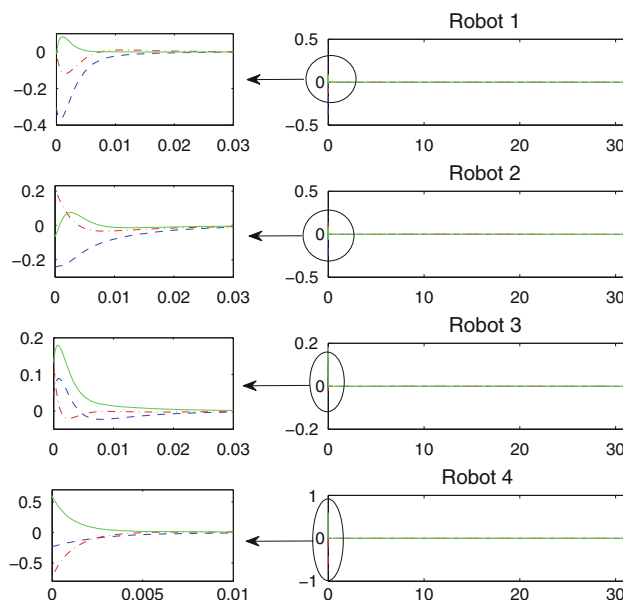
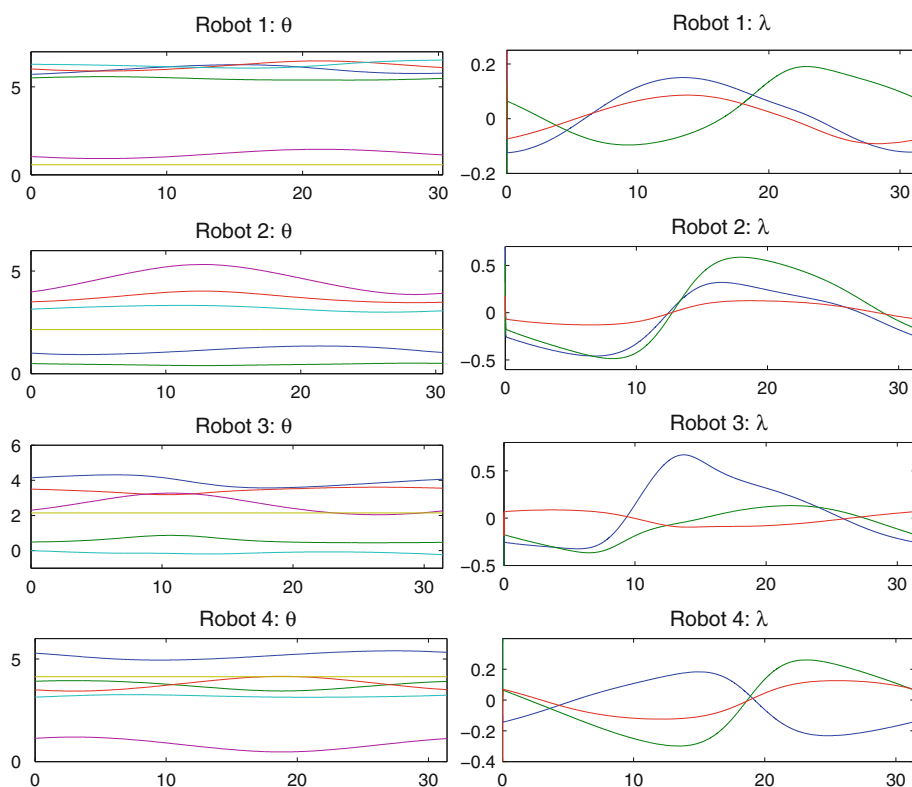




**Fig. 5** The trajectories of all manipulators in the workspace (drawn together)

cannot access the command signal directly but are able to track the desired trajectory by following observations of their neighbors. As shown in Fig. 2, Robot 1 can communicate with Robot 4 bidirectionally and can also send information to Robot 2. Robot 3 can communicate with Robot 4 bidirectionally. Robot 2 can observe information from Robot 1 and pass information to Robot 4. With such a communication topology, we can get the corresponding hierarchical tree by running Algorithm 1 as shown in Fig. 2 (Fig. 3).

**Fig. 6** The time profile of joint angle  $\theta$  and dual parameter  $\lambda$  for all manipulators



**Fig. 7** The time profile of velocity errors of the end effectors in  $x$ -direction (the blue dash curves),  $y$ -direction (the red dash-dot curves),  $z$ -direction (the green solid curves), respectively, for all manipulators (Color figure online)

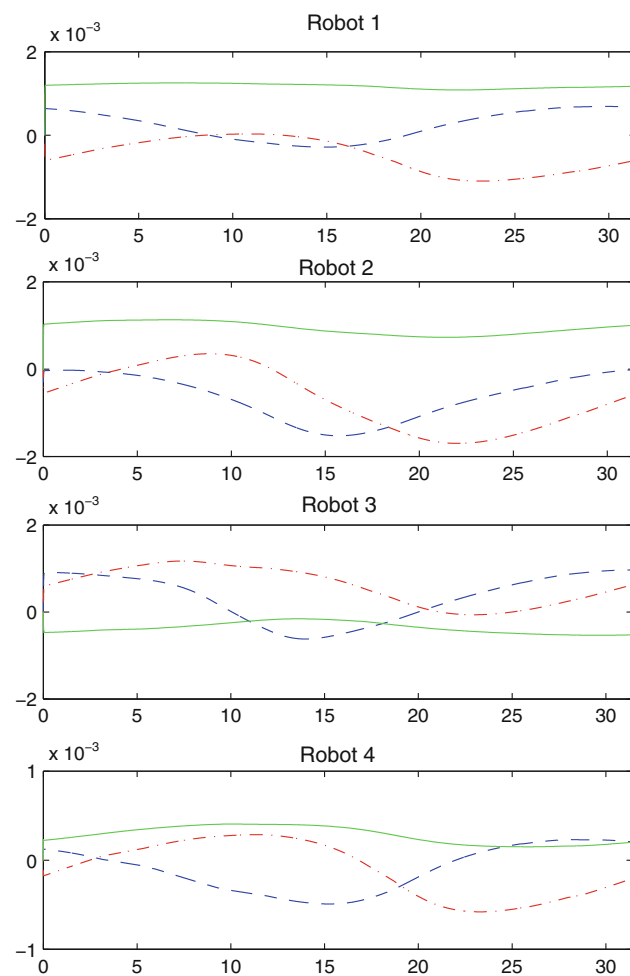
As to the neural dynamics, we choose  $\epsilon = 10^{-3}$ ,  $\eta^+ = [1, 1, 1, 1, 1, 1]^T$  and  $\eta^- = -\eta^+$  for the neural network. Figures 4 and 5 illustrate motion trajectories of the manipulators in the workspace. From this figure, we can observe that



the end effectors of all manipulators track the desired circular trajectory simultaneously. The evolutions of  $\theta$  and  $\lambda$  with time for the manipulators are shown in Fig. 6. We can see from Fig. 7 that the velocity error, measured with the difference between the velocity of the end effector and the desired velocity, converge to a small value very fast after a short transient at the very beginning. Figure 8 plots the time profile of the position error, measured by the position difference between the end effector and the desired reference point. From this figure, we can see that the position errors are controlled within  $2 \times 10^{-3}$  m in the three axial directions, which demonstrates the accuracy of the proposed strategy.

## 6 Conclusions and future work

In this paper, the multiple manipulator task execution problem with only partial command coverage of manipulators is investigated. This problem cannot be solved with



**Fig. 8** The time profile of position errors of the end effectors in  $x$ -direction (the blue dash curves),  $y$ -direction (the red dash-dot curves),  $z$ -direction (the green solid curves), respectively, for all manipulators (Color figure online)

existing distributed neural approach [1], which requires the command signal available to all manipulators. A strategy with hierarchical tree generation on topology and nonlinear neural dynamics as controllers are proposed to tackle the problem. The novelty of this paper lies that most existing work is centralized method, which is not scalable to networks involving a large number of manipulators while the proposed strategy is a distributed one, which only need information from neighbors for each manipulator. On the other hand, most studies on multiple manipulator coordination focus on general manipulators instead of redundant manipulators. Redundant manipulators have extra degree of freedom (DOF) and can be exploited for optimization. However, the extra design degrees also bring new challenge to the control design as not only stability must be guaranteed but also optimality be guaranteed by the design. The global stability of the neural network and the optimality of the solution are both theoretically proven. Simulations show that the proposed method is effective and accurate. As the proposed strategy need to first generate a hierarchical tree from the communication topology and the signal of a children manipulator has dependence on that of its parent manipulator, the proposed method may be fragile to link failure, especially for links close to the command center. For example, if a manipulator loses connection from its parent, then all the descendants of this manipulator will be out of control. This issue will be examined in our future work and information exchanges between all neighbors may help to tackle this problem.

**Acknowledgments** The authors would like to acknowledge the constant motivation and inspiration by the motto stating that “the only limit to our realization of tomorrow will be our doubts of today”. The authors would like to acknowledge the support by the National Science Foundation of China under Grant NSFC: 61105090.

## References

1. Li S, Chen S, Liu B, Li Y, Liang Y (2012) Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks. *Neurocomputing* 91:1–10
2. Montemayor G, Wen JT (2005) Decentralized collaborative load transport by multiple robots. In: *Robotics and automation, 2005. Proceedings of the 2005 IEEE international conference on ICRA 2005*. pp 372–377
3. Fraile J, Paredis CJJ, Wang C, Khosla PK (1999) Agent-based planning and control of a multi-manipulator assembly system. In: *Proceedings of the IEEE international conference on robotics and automation*, pp 1219–1225
4. Liu G, Xu J, Wang X, Li Z (2004) On quality functions for grasp synthesis, fixture planning, and coordinated manipulation. *IEEE Trans Autom Sci Eng* 1(2):146–162
5. Wu L, Cui K, Chen SB (2000) Redundancy coordination of multiple robotic devices for welding through genetic algorithm. *Robotica* 18:669–676

6. Alami R, Fleury S, Herrb M, Ingrand F, Robert F (1998) Multi-robot cooperation in the MARTHA project. *Robot Autom Mag IEEE* 5(1):36–47
7. Peng J (2004) Optimal control of multiple robot systems with friction using mathematical programs with complementarity constraints. In: *Proceedings of the IEEE international conference on robotics and automation*, pp 5224–5231
8. Moosavian SAA, Papadopoulos E (1997) On the control of space free-flyers using multiple impedance control. In: *Proceedings of the 1997 IEEE international conference on robotics and automation*, 1997, vol 1, pp 853–858
9. Liu Y, Xu Y, Bergerman M (1999) Cooperation control of multiple manipulators with passive joints. *IEEE Trans Robot Autom* 15:258–267
10. Zhang Y, Guo D, Cai B, Chen K (2011) Remedy scheme and theoretical analysis of joint-angle drift phenomenon for redundant robot manipulators. *Robot Comput Integr Manuf* 27(4):860–869
11. Li S, Liu B, Chen B, Lou Y (2012) Neural network based mobile phone localization using bluetooth connectivity. *Neural Comput Appl*, Springer-Verlag, 1–9
12. Xia Y, Feng G, Wang J (2004) A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations. *Neural Netw* 17(7):1003–1015
13. Hu X (2010) Dynamic system methods for solving mixed linear matrix inequalities and linear vector inequalities and equalities. *Appl Math Comput* 216(4):1181–1193
14. Li S, Lou Y, Liu B (2012) Bluetooth aided mobile phone localization: a nonlinear neural circuit approach. *ACM Trans Embed Comput Syst* (to appear)
15. Zhang Y, Shi Y, Chen K, Wang C (2009) Global exponential convergence and stability of gradient-based neural network for online matrix inversion. *Appl Math Comput* 215(3):1301–1306
16. Mastroeni G (2005) Gap functions and descent methods for minty variational inequality. In: Qi L, Teo K, Yang X, Pardalos PM, Hearn DW (eds) *Optimization and control with applications*, volume 96 of applied optimization, Springer, Berlin, pp 529–547
17. Fukushima M (1992) Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. *Math Program* 53:99–110
18. Khalil H (2002) *Nonlinear systems*. Prentice Hall, Englewood Cliffs, NJ