Distributed Recurrent Neural Networks for Cooperative Control of Manipulators: A Game-Theoretic Perspective

Shuai Li, Jinbo He, Yangming Li, and Muhammad Usman Rafique

***Abstract*— This paper considers cooperative kinematic control of multiple manipulators using distributed recurrent neural networks and provides a tractable way to extend existing results on individual manipulator control using recurrent neural networks to the scenario with the coordination of multiple manipulators. The problem is formulated as a constrained game, where energy consumptions for each manipulator, saturations of control input, and the topological constraints imposed by the communication graph are considered. An implicit form of the Nash equilibrium for the game is obtained by converting the problem into its dual space. Then, a distributed dynamic controller based on recurrent neural networks is devised to drive the system toward the desired Nash equilibrium to seek the optimal solution of the cooperative control. Global stability and solution optimality of the proposed neural networks are proved in the theory. Simulations demonstrate the effectiveness of the proposed method.**

***Index Terms*— Distributed control, dual neural network, game theory, kinematic resolution, neural network, recurrent neural network, redundant manipulator.**

I. Introduction

AMONG various types of manipulators, redundant manipulators, which have more degree of freedom (DOF) than required, are widely utilized for conducting complicated tasks with extra flexibility. As a powerful tool for real-time parallel processing, recurrent neural networks have demonstrated great advantages in the kinematic redundancy resolution of an individual redundant manipulator. In recent years, using multiple redundant manipulators, to collectively conduct complicated tasks, which require a large amount of efforts, are becoming increasingly popular. Multimanipulator control has attracted intensive research attention in the past years and has successfully been applied to numerous applications [1]–[4]. The extension from individual manipulator control using recurrent neural networks to multimanipulator scenario remains an unsolved problem. In particular, when topological constraints imposed by communication graph are considered, a tractable way to design recurrent neural controllers for distributed control is highly demanded.

Neural network approaches have long been employed for the redundancy resolution of a single manipulator. In [5], an adaptive recurrent neural network with guaranteed asymptotical convergence is applied to the control of a mobile manipulator with unknown dynamics. In [6], the redundancy resolution problem is modeled as quadratic programming where the objective is to minimize kinematic energy consumption. The constraints are formed by a set of linear equations abstracting the affine mapping from joint space to workspace and a set of inequalities corresponding to input bounds. A recurrent neural network is obtained by solving such a quadratic program, and the desired joint torque can thus be obtained in real time. It is found in [7] that the traditional control scheme may fail when applied for circular motion tracking. A new performance index is proposed to conquer this problem. Zhang *et al.* [8] present a minimum-energy redundancy resolution methodology to control manipulators using recurrent neural networks. Zhang and Zhang [9] extend the solution to a more general case with variable joint velocity limits. The minimum infinity norm of joint velocities is employed in [10] as the cost function to establish neural dynamics. Subsequent work shows that the neural model can significantly be simplified in structure under the same problem formulation [11]. For the case with kinematic energy consumption as the cost to minimize, a simplified one-layer dual neural network is obtained in [12] for efficient control of redundant manipulators. In existing approaches of using recurrent neural networks for manipulator control, some consider the problem at velocity level, while the others deal with acceleration. It is shown in [13] that there exists an inherent equivalence for different levels of redundancy resolution. In [14], recurrent neural network approach is extended to address kinematic redundancy resolution of parallel Stewart platforms. In [15], obstacle avoidance is modeled as an inequality constraint for serial manipulator motion control. In principle, recurrent neural networks capable of solving constrained quadratic programming are able to deal with redundancy resolution for a single manipulator [16]–[26].

Although there have been extensive investigations on using recurrent neural networks for the control of a single redundant manipulator, the research on recurrent neural network-based multimanipulator redundancy resolution is far from up-to-date.

Manuscript received September 11, 2015; revised December 7, 2015; accepted January 7, 2016. Date of publication January 21, 2016; date of current version January 17, 2017. This work was supported in part by the Research Grants Council within the Early Career Scheme, Hong Kong, under Grant 25214015, in part by the Departmental General Research Fund through The Hong Kong Polytechnic University under Grant G.UA7L, and in part by the National Natural Science Foundation of China under Grant 61401385.

S. Li, J. He, and M. U. Rafique are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: shuaili@polyu.edu.hk; adriano.he@connect.polyu.hk; csrafique@comp.polyu.edu.hk).

Y. Li is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: ymli81@uw.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2516565

TABLE I
COMPARISONS OF DIFFERENT RECURRENT NEURAL NETWORK ALGORITHMS FOR THE REDUNDANCY RESOLUTION OF MANIPULATORS

Neural networks	Convergence	Optimality	Mathematical formulation	Manipulator numbers	Distributed v.s. centralized	Topology	All connected to command center	Pre-generation of topology
This paper	Yes	Yes	Game theory	Multiple	Distributed	Neighbor-to-neighbor	No	No
Paper [27]	Yes	Yes	Optimization	Two	Centralized	-	Yes	No
Paper [28]	Yes	Yes	Optimization	Multiple	Distributed	Star	Yes	No
Paper [29]	Yes	Yes	Optimization	Multiple	Distributed	Tree	No	Yes
Paper [5]	Yes	No	Adaptive control	Single	-	-	-	-
Papers [6]–[13], [15]	Yes	Yes	Optimization	Single	-	-	-	-

Notes: '-' means the item does not apply to the algorithm proposed in the associated papers.

Actually, the increased complexity and computational burdens in multimanipulator coordination pose more requirements on real-time processing for the redundancy resolution in network level. This also casts lights to recurrent neural networks, thanks to its computational power, for enhanced performance in handling with such a multimanipulator problem. In [27], a recurrent neural network is designed for synchronous manipulation of two redundant robot arms. However, this paper only considers a dual arm system and the generalization to a network of manipulators remains unclear. In [28], a recurrent neural network is proposed to address kinematic control of a class of collaborative redundant manipulators. However, the neural model is derived under the assumption that all manipulators are able to access global command on the desired velocity in workspace. This result is extended to the situation with a hierarchical organization of all manipulators [29]. Comparison between existing recurrent neural network solutions for manipulator redundancy resolution and the proposed solution is summarized in Table I. To the best of our knowledge, there is no systematic solution on recurrent neural network design for multimanipulator coordination under a general topological constraint.

In this paper, we make progress along this direction by presenting a game-theoretic framework to guide the design of recurrent neural networks for distributed coordination of multiple redundant manipulators. In this framework, the multimanipulator coordination problem is formulated as a constrained game. Individual manipulators compete to optimize their own objectives under a set of constraints. A recurrent neural network is derived to solve the Nash equilibrium of this problem dynamically. The obtained neural control law is in compliance with the communication topology by only allowing information exchange between neighboring manipulators. The contributions of this paper are summarized as follows.

1) To the best of our knowledge, this is the first work using recurrent neural networks for multimanipulator control under a general topological constraint.
2) This paper extends the establishment of neural dynamics from an optimization perspective to a game-theoretic perspective.

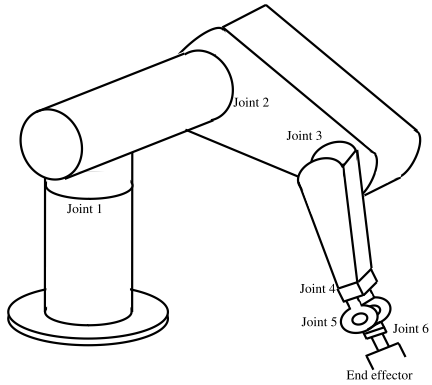


Fig. 1. Schematic of a 6-DOF PUMA 560 manipulator.

3) The presented neural networks only require information exchange from its one-hop neighbors, which is advantageous in comparison with that requiring all-to-all communication between all manipulators.
4) It is proved in this paper that the presented neural controllers are guaranteed to be globally convergent.

The rest of this paper is organized as follows. In Section II, the kinematic control of networked redundant manipulators is formulated as a constrained game. Its Nash equilibrium is expressed in a set of nonlinear equations in Section III. In Section IV, a recurrent neural network with separate modules is proposed to solve the Nash equilibrium in real time. In Section V, the global stability of the proposed neural network and the optimality of the neural solution are proved in the theory. In Section VI, the simulations are performed to validate the effectiveness of the proposed method. Finally, the conclusion is drawn in Section VII.

II. PROBLEM FORMULATION

A. Redundant Manipulator Kinematics

For a manipulator, the position of its end effector is uniquely determined by its configuration in the joint space. For example, the schematic of a 6-DOF programmable universal machine for assembly (PUMA) 560 manipulator is shown in Fig. 1,

six revolutionary joints (joints 1–6 in Fig. 1) connect seven links (the last link is an end effector) in series. As discussed in [30] and [31] in detail, the 3-D workspace position of the end effector at time t, which is $r(t)$, can be expressed as

$$r(t) = T_0T_1T_2T_3T_4T_5T_6r_6$$

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$T_1 = T_1(\theta_1(t)) = \begin{bmatrix} \cos\theta_1(t) & -\sin\theta_1(t) & 0 & 0 \\ \sin\theta_1(t) & \cos\theta_1(t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = T_2(\theta_2(t)) = \begin{bmatrix} \cos\theta_2(t) & -\sin\theta_2(t) & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -\sin\theta_2(t) & -\cos\theta_2(t) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = T_3(\theta_3(t)) = \begin{bmatrix} \cos\theta_3(t) & -\sin\theta_3(t) & 0 & a_2 \\ \sin\theta_3(t) & \cos\theta_3(t) & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = T_4(\theta_4(t)) = \begin{bmatrix} \cos\theta_4(t) & -\sin\theta_4(t) & 0 & a_3 \\ 0 & 0 & -1 & -d_4 \\ \sin\theta_4(t) & \cos\theta_4(t) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5 = T_5(\theta_5(t)) = \begin{bmatrix} \cos\theta_5(t) & -\sin\theta_5(t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_5(t) & -\cos\theta_5(t) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6 = T_6(\theta_6(t)) = \begin{bmatrix} \cos\theta_6(t) & -\sin\theta_6(t) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_6(t) & \cos\theta_6(t) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where d_2, a_2, d_3, a_3, and d_4 are all constants determined by the geometrical properties of the manipulator, $r_6^T = [x_6, y_6, z_6, 1]$ with $[x_6, y_6, z_6]$ being a constant vector denoting the local coordinate of the reference point in the end-effector frame. Clearly, $r(t)$ in the above equation is a nonlinear function of $\theta^T(t) = [\theta_1(t), \theta_2(t), \theta_3(t), \theta_4(t), \theta_5(t), \theta_6(t)]$. In particular, for any m-DOF manipulator working in an n-dimensional Cartesian space, we have 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 coordinates 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 the 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.

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 the known manipulator model $J(\cdot)$. For redundant manipulators, normally, the inverse kinematic solution is not unique, since the number of equalities in this problem is less than the number of dimensions of the decision variable $\dot{\theta}(t)$, i.e., $n < m$. 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 include the minimum of the Euclidian norm or the infinity norm of the joint velocity vector, and possible constraints include joint angle limits, joint speed limits, and so on.

B. Game-Theoretic Formulation of Cooperative Control

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: 1) a reference point on the payload, e.g., the center of mass, is expected to track the reference trajectory and 2) 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 resulting from the relative movement between the end effectors and this requirement can be satisfied by moving all the end effectors at the same velocity as that of 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 the 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 the manipulators with the same velocity $v_d(t)$, i.e., in equation, we have

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

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 ith 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, and k denotes the number of manipulators. Without introducing confusions, we abbreviate (3) 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), compared with the direct position control based on the nonlinear transformation (1), using velocity control of the collection of manipulators thoroughly simplifies the design. Because of redundancy property of the redundant manipulators, the solution satisfying the two aforementioned aspects of the task is not unique. This allows us to consider extra optimization

criteria and constraints. Existing works [28], [29] formulate the problem as an optimization one and solve it in real time using a recurrent neural network. In the peer-to-peer control scenario considered in this paper, the commanding signal is only accessible to its neighboring manipulators, instead of all manipulators [28]. In addition, the pregeneration of the hierarchical topology to propagate the velocity command from the command center [29] is not allowed in the peer-to-peer control investigated in this paper. Thus, it is difficult to reach a distributed control algorithm by following the optimization-based framework. In contrast, the game theory [32], [33] allows us to consider the peer-to-peer control of multiple redundant manipulators in a distributed fashion naturally by confining the opponent players in the neighborhood for each player. Concretely, without loss of generality, we minimize the Euclidean norm squared of the joint velocities, i.e., $U_i = (1/2)\dot{\theta}_i^T\dot{\theta}_i$ for the ith manipulator, under the joint velocity constraint $\eta^- \le \dot{\theta}_i \le \eta^+$ and the desired end-effector velocity constraint, to exploit the extra design freedoms. In summary, the cost function and the constraint for the ith ($i = 1, 2, \ldots, l$) manipulator are as follows:

$$\min\ U_i(\dot{\theta}_i, \dot{\theta}_{-i}) = \frac{1}{2}\dot{\theta}_i^T\dot{\theta}_i \tag{5a}$$

$$\text{s.t. } v_i = \frac{1}{\sum_{k\in\mathcal{N}_i} w_{ik}} \sum_{k\in\mathcal{N}_i} w_{ik}v_k \tag{5b}$$

$$\eta^- \le \dot{\theta}_i \le \eta^+$$

$$v_k = \begin{cases} J_k\dot{\theta}_k & \text{for } k = 1, 2, \ldots, l \\ v_d & \text{for } k = 0 \end{cases} \tag{5c}$$

where $\mathcal{N}_i$ defines the neighbor set of the ith manipulator, $w_{ik} = w_{ki}$ is the connection weight between the ith manipulator and the kth one, where $w_{ik} > 0$ for $k \in \mathcal{N}_i$ and $w_{ik} = 0$ for $k \notin \mathcal{N}_i$. $\theta_i \in \mathbb{R}^m$ and $\dot{\theta}_i \in \mathbb{R}^m$ are the coordinates and the velocity of the ith manipulator in the joint space for the ith manipulator, respectively. In the terminology of game theory, $\dot{\theta}_i$ is the strategy for the ith manipulator. $\dot{\theta}_{-i}$ is defined as the set of strategies chosen by neighboring opponent players in the game, i.e., the set of the neighboring manipulators' joint velocities. In the equation, $\dot{\theta}_{-i} = \{\dot{\theta}_k, \forall k \in \mathcal{N}_i\}$. $\eta^+ \in \mathbb{R}^m$ and $\eta^- \in \mathbb{R}^m$ are the upper bound and lower bound of the allowed velocity in the joint space, respectively. $J_i \in \mathbb{R}^{n\times m}$ is the Jacobian matrix of the ith manipulator. $v_k \in \mathbb{R}^n$ represents the end-effector velocity of the kth manipulator. l is the total number of manipulators, and we regard the command center as a virtual manipulator labeled $k = 0$. The manipulators in the neighborhood of the virtual manipulator have access to the desired end-effector velocity, which is $v_0 = v_d$.

It is noteworthy that the desired end-effector velocity constraint (3) is replaced by the weighted average of its neighboring manipulators' end-effector velocities in (5) due to the unavailability of v_d to the manipulators outside the neighborhood of the command center. Actually, $v_i = (1/\sum_{k\in\mathcal{N}_i} w_{ik})\sum_{k\in\mathcal{N}_i} w_{ik}v_k$ for all $i = 1, 2, \ldots, l$ jointly suffices $v_i = v_d$ for all $i = 1, 2, \ldots, l$ provided that the communication network formed by the manipulators and the command center is connected. This point is summarized in the following lemma.

Lemma 1: The constraint $v_i = \sum_{k\in\mathcal{N}_i} w_{ik}v_k/\sum_{k\in\mathcal{N}_i} w_{ik}$ [line 2 in (5)] for $i = 1, 2, \ldots, l$ with $v_0 = v_d$ is equivalent to $v_i = v_d$ (4) for $i = 1, 2, \ldots, l$ provided that the communication network formed by the manipulators and the command center is connected.

Proof: The constraint $v_i = (1/\sum_{k\in\mathcal{N}_i} w_{ik})\sum_{k\in\mathcal{N}_i} w_{ik}v_k$ for $i = 1, 2, \ldots, l$ can be written in the following compact form using Kronecker algebra:

$$(L \otimes I_m)v = 0 \tag{6}$$

where $\otimes$ is the Kronecker product operator, m denotes the dimension of joint space, I_m is an $m \times m$ identity matrix, $L \in \mathbb{R}^{l\times l}$ is a Laplacian matrix [34], and v is the velocity vector by stacking the joint velocities of all manipulators into a single vector. The expression of L_{ij}, which is the element of L on the ith row and the jth column, and the expression of v are as follows:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \cdots \\ v_l \end{bmatrix}, \quad L_{ij} = \begin{cases} \sum_{k\in\mathcal{N}_i} w_{ik} & \text{if } i = j \\ -w_{ij} & \text{if } i \ne j. \end{cases} \tag{7}$$

Note that the Laplacian matrix L defined by (7) is a symmetric one and so is $L \otimes I_m$ by recalling that $w_{ik} = w_{ki}$ for all possible i and k. The $l \times l$ Laplacian matrix L defined on a connected graph has rank $(l - 1)$ and always has a nullspace spanned by the l dimensional vector $\mathbf{1} = [1, 1, \ldots, 1]^T$ [34]. Accordingly, we conclude that $(L \otimes I_m)v = 0$ is equivalent to $v = \mathbf{1} \otimes v' = [v'^T, v'^T, \ldots, v'^T]^T$, where $v' \in \mathbb{R}^m$. This result indicates that all manipulators have the same joint velocity $v_i = v'$ for $i = 1, 2, \ldots, l$. Then, we need $v' = v_d$ to complete the proof. We now consider a particular manipulator, say the jth one, which is in the neighborhood of the command center. For this manipulator, we have

$$v_j = \frac{1}{\sum_{k\in\mathcal{N}_j} w_{ik}} \sum_{k\in\mathcal{N}_j} w_{jk}v_k \tag{8}$$

that is

$$\begin{aligned} v' &= \frac{1}{\sum_{k\in\mathcal{N}_j} w_{ik}} \left(\sum_{k\in\mathcal{N}_j, k\ne 0} w_{jk}v' + w_{j0}v_0 \right) \\ &= \frac{1}{\sum_{k\in\mathcal{N}_j} w_{ik}} \left(w_{j0}v_0 + v' \sum_{k\in\mathcal{N}_j, k\ne 0} w_{jk} \right) \end{aligned} \tag{9}$$

by substituting $v_i = v'$ for $i = 1, 2, \ldots, l$ into (8). Clearly, the solution of (9) is $v' = v_0$, i.e., $v_i = v_0 = v_d$ for $i = 1, 2, \ldots, l$, which completes the proof. □

So far, we have formulated the peer-to-peer networked redundant manipulator cooperation problem as an l player game. The goal of the ith ($i = 1, 2, \ldots, l$) manipulator is to minimize its own kinematic energy $U_i = \dot{\theta}_i^T\dot{\theta}_i/2$, under the joint velocity constraint $\eta^- \le \dot{\theta}_i \le \eta^+$ and the end-effector velocity constraint $v_i = v_d$, i.e., $v_i = \sum_{k\in\mathcal{N}_i} w_{ik}/\sum_{k\in\mathcal{N}_i} w_{ik}$, as proved in Lemma 1. Note that there is no interaction terms in the cost function U_i and the joint velocity limit constraint, and these items are only dependent on $\dot{\theta}_i$, which is the strategy

of the ith manipulator. The interaction term appears in the end-effector velocity constraint $v_i = \sum_{k\in\mathcal{N}_i} w_{ik}v_k / \sum_{k\in\mathcal{N}_i} w_{ik}$. For this equality constraint, the feasible solution set of $\dot{\theta}_i$ relies on the strategies of neighboring manipulators, which are $\dot{\theta}_k$ for all $k \in \mathcal{N}_i$. We aim to find out the set of joint velocities of all manipulators $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, \ldots, \dot{\theta}_l^{*T}]^T$, at which no manipulator can further reduce their own kinematic energy without violating the constraints by changing its strategy, while the other manipulators keep their strategies unchanged. Or more formally in the language of game theory, the goal is to find the pure-strategy Nash equilibrium for the l player game defined by (5).

III. Nash Equilibrium of the Game for Cooperative Control

In Section II, we have formulated the peer-to-peer networked manipulator cooperation problem as an l player game. In this section, we will explore the solution of this game by equivalently considering problem (5) in its dual space.

According to Karash–Kuhn–Tucker (KKT) conditions [35], the solution to problem (5) satisfies

$$\dot{\theta}_i = -J_i^T\lambda_i - \mu_i \tag{10a}$$

$$v_i = \frac{1}{\sum_{k\in\mathcal{N}_i} w_{ik}} \sum_{k\in\mathcal{N}_i} w_{ik}v_k \tag{10b}$$

$$\begin{cases} \dot{\theta}_i = \eta^+ & \text{if } \mu_i > 0 \\ \eta^- < \dot{\theta}_i < \eta^+ & \text{if } \mu_i = 0 \\ \dot{\theta}_i = \eta^- & \text{if } \mu_i < 0 \end{cases} \tag{10c}$$

where λ_i and μ_i are the dual variables corresponding to the equality constraint and the inequality constraint in (5), respectively. Note that $(\partial v_i/\partial\dot{\theta}_i) = J_i$ and $(\partial v_k/\partial\dot{\theta}_i) = 0$ for $k \neq i$. This is obtained by recalling that $v_k = J_k\dot{\theta}_k$, for $k = 1, 2, \ldots, l$, and it is used in the derivation of (10).

Note that (10c) can be simplified into the following form:

$$\dot{\theta}_i = g(\dot{\theta}_i + \mu_i) \tag{11}$$

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

$$g_j(x_j) = \begin{cases} \eta_j^+ & \text{if } x_j > \eta_j^+ \\ x_j & \text{if } \eta_j^- \le x_j \le \eta_j^+ \\ \eta_j^- & \text{if } x_j < \eta_j^- \end{cases} \tag{12}$$

where η_j^- and η_j^+ are the jth elements of η^- and η^+, respectively. Substituting (10a) into (11) to cancel out $\dot{\theta}_i$ yields

$$\mu_i = -g\big(-J_i^T\lambda_i\big) - J_i^T\lambda_i \tag{13}$$

which means μ_i can be explicitly expressed in terms of λ_i. $\dot{\theta}_i$ can also be expressed in terms of λ_i by substituting (13) into (10a)

$$\dot{\theta}_i = g\big(-J_i^T\lambda_i\big). \tag{14}$$

With the expression of v_i and (11) plugged into (10b), we obtain

$$J_i g(\dot{\theta}_i + \mu_i) = \frac{1}{\sum_{k\in\mathcal{N}_i} w_{ik}} \sum_{k\in\mathcal{N}_i} w_{ik}v_k. \tag{15}$$

Canceling out $\dot{\theta}_i$ by substituting (10a) into (15) yields

$$J_i g\big(-J_i^T\lambda_i\big) = \frac{1}{\sum_{k\in\mathcal{N}_i} w_{ik}} \sum_{k\in\mathcal{N}_i} w_{ik}v_k. \tag{16}$$

Note that the above results hold for all $i = 1, 2, \ldots, l$. Therefore, v_k for $k = 1, 2, \ldots, l$ can be written as follows by substituting (14) for $i = k$:

$$v_k = J_k g\big(-J_k^T\lambda_k\big). \tag{17}$$

With the virtual manipulator $k = 0$ and the real manipulators considered all together, we get

$$v_k = \begin{cases} J_k g\big(-J_k^T\lambda_k\big) & \text{for } k = 1, 2, \ldots, l \\ v_d & \text{for } k = 0. \end{cases} \tag{18}$$

In summary, the pure-strategy Nash equilibrium of problem (5) is the solution of the following equations by combining (14), (16), and (18):

$$J_i g\big(-J_i^T\lambda_i\big) = \frac{1}{\sum_{k\in\mathcal{N}_i} w_{ik}} \sum_{k\in\mathcal{N}_i} w_{ik}v_k \tag{19a}$$

$$\dot{\theta}_i = g\big(-J_i^T\lambda_i\big) \quad \text{for } i = 1, 2, \ldots, l \tag{19b}$$

where

$$v_k = \begin{cases} J_k g\big(-J_k^T\lambda_k\big) & \text{for } k = 1, 2, \ldots, l \\ v_d & \text{for } k = 0. \end{cases} \tag{19c}$$

As problem (5) can be regarded as a constrained quadratic programming problem relative to $\dot{\theta}_i$ for the ith manipulator, the solution obtained above using KKT conditions is an equivalent solution. Thus, the above derivation can be summarized into the following lemma.

Lemma 2: The solution of problem (5) for $i = 1, 2, \ldots, l$ is equivalent to the solution of the equation set (19) for $i = 1, 2, \ldots, l$.

In the equation set (19), $\dot{\theta}_i$ is expressed as a function of λ_i, and λ_i is involved in the nonlinear equations [note that $g(\cdot)$ is a nonlinear mapping] coupling λ_i and λ_k in the neighborhood of the ith manipulator. Therefore, it is difficult to find the closed-form expression of $\dot{\theta}_i$ by solving (19) directly. Instead, as inspired by the great success of recurrent neural networks in online signal processing, optimization, and so on, we will seek to use a recurrent neural network to solve λ_i and $\dot{\theta}_i$ simultaneously for $i = 1, 2, \ldots, l$ in (19) in real time.

Remark 1: The equation set (19) solves the Nash equilibrium of the game (5). However, due to the presence of the nonlinear term $g(\cdot)$ in (19) and the fact that the available information is confined in the neighborhood, (19) cannot be solved analytically. We thus explore to solve (19) in an iterative way.

IV. Online Solution via Recurrent Neural Networks

In this section, we design a recurrent neural network, called game-theoretic neural network (GTNN), to iteratively solve λ_i and $\dot{\theta}_i$ for $i = 1, 2, \ldots, l$ in (19), which is the solution of the Nash equilibrium of the game (5). We will first propose the neural network model and then give a brief discussion.

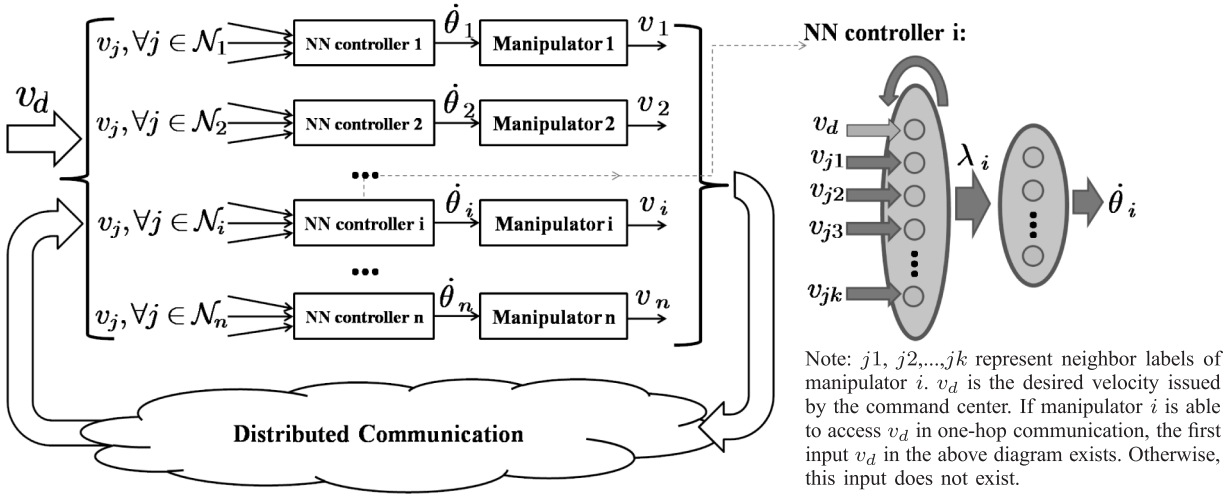


Fig. 2. Architecture of the proposed GTNN to address multimanipulator distributed cooperation problem. Left: overall neural network-based control architecture involving all manipulators. Right: architecture of the ith neural network controller (NN controller) associated with manipulator i.

A. Recurrent Neural Network Model

We use the following dynamic neural model to solve the nonlinear interaction equation $J_i g(-J_i^T \lambda_i) = (1/\sum_{k\in\mathcal{N}_i} w_{ik}) \sum_{k\in\mathcal{N}_i} w_{ik} v_k$ in (19):

$$\epsilon \dot{\lambda}_i = J_i g\big(-J_i^T \lambda_i\big) \sum_{k\in\mathcal{N}_i} w_{ik} - \sum_{k\in\mathcal{N}_i} w_{ik} v_k. \tag{20}$$

Now, the GTNN model can be obtained to solve the networked redundant manipulator cooperation problem (5). The proposed GTNN model is composed of l modules corresponding to the totally l manipulators. Each module is called a neural network controller (referred as NN controller i in Fig. 2 for the ith one). The state and output equations of the ith module, or the ith NN controller, for $i = 1, 2, \ldots, l$ of GTNN are summarized as follows:

State Equation:

$$\epsilon \dot{\lambda}_i = J_i g\big(-J_i^T \lambda_i\big) \sum_{k\in\mathcal{N}_i} w_{ik} - \sum_{k\in\mathcal{N}_i} w_{ik} v_k \tag{21a}$$

$$v_k = \begin{cases} J_k g\big(-J_k^T \lambda_k\big) & \text{for } k = 1, 2, \ldots, l \\ v_d & \text{for } k = 0 \end{cases} \tag{21b}$$

Output Equation:

$$\dot{\theta}_i = g\big(-J_i^T \lambda_i\big) \tag{21c}$$

where $i = 1, 2, \ldots, l$ is the manipulator label, and $\epsilon > 0$ is the scaling factor for the ith manipulator.

About the architecture of the proposed GTNN model (21), we have the following remark.

Remark 2: The proposed GTNN model (21) consists of n modules (n is the number of manipulators), each of which functions as a neural network controller (NN controller in Fig. 2) to steer the movement of the corresponding manipulator. As shown in Fig. 2, each NN controller, say the ith one, receives input v_j from its neighbor $j \in \mathcal{N}_i$. If the command center is also in the neighborhood, v_d is accessible to the ith NN controller; otherwise, it is not accessible. After operations in the module, the ith NN controller outputs $\dot{\theta}_i$ as the control input to the manipulator. Driven by this signal, the ith manipulator generates a movement v_i in its workspace. Overall, the ith NN controller exchanges information with its neighbors defined by the distributed communication and all controllers work together to reach a convergent coordination of the manipulator group. As shown in Fig. 2 (right), the ith NN controller can be viewed as a two-layer structure. The first layer is a recurrent one with λ_i as the state variable and v_j as input for all $j \in \mathcal{N}_i$. Note that v_d is an input to the ith NN controller if the command center is in the neighbor of the ith manipulator; otherwise, v_d is not its input. The output produced by the first layer becomes the input to the second layer, which is a static mapping from λ_i to $\dot{\theta}_i$.

With (21), the neural network dynamics can be written into a compact form by considering all the manipulators together as follows using Kronecker algebra:

$$\begin{aligned} \epsilon \dot{\bar{\lambda}} &= (L \otimes I_m) \bar{J} \bar{g}(-\bar{J}^T \bar{\lambda}) \\ &\quad + (\Lambda \otimes I_m)(\bar{J} \bar{g}(-\bar{J}^T \bar{\lambda}) - \mathbf{1} \otimes v_d) \end{aligned} \tag{22a}$$

$$\dot{\bar{\theta}} = \bar{g}(-\bar{J}^T \bar{\lambda}) \tag{22b}$$

where $\otimes$ is the Kronecker product operator, m denotes the dimension of the joint space, I_m is an $m \times m$ identity matrix, and $\mathbf{1}$ is an l-dimensional vector with all entries 1 and

$$\bar{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \ldots \\ \lambda_l \end{bmatrix}, \ \bar{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \ldots \\ \theta_l \end{bmatrix}, \quad \bar{J} = \begin{bmatrix} J_1 & 0 & 0 & \ldots & 0 \\ 0 & J_2 & 0 & \ldots & 0 \\ 0 & 0 & J_3 & \ldots & 0 \\ & & \ldots & & \\ 0 & 0 & 0 & \ldots & J_l \end{bmatrix} \tag{23}$$

where the function $\bar{g}(\cdot)$ is formed by stacking together $g(\cdot)$ for $i = 1, 2, \ldots, l$, i.e., $\bar{g}(x)$ is defined as follows for $x = [x_1^T, x_2^T, \ldots, x_l^T]^T$ with $x_i \in \mathbb{R}^m$ for $i = 1, 2, \ldots, l$:

$$\bar{g}(x) = \begin{bmatrix} g(x_1) \\ g(x_2) \\ \ldots \\ g(x_l) \end{bmatrix}. \tag{24}$$

Λ is a diagonal matrix defined as

$$\Lambda_{ij} = \begin{cases} w_{0i} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{25}$$

with $\mathcal{N}_0$ denoting the neighbor set of the command center. L is a Laplacian matrix [34], whose element on the ith row and jth column is defined as

$$L_{ij} = \begin{cases} w_{ij} & \text{if } i = j \\ -\sum_{k\in\mathcal{N}_i} w_{ik} & \text{if } i \neq j. \end{cases} \tag{26}$$

Note that the Laplacian matrix L defined by (26) is a symmetric one and so is $L \otimes I_m$ by recalling that $w_{ik} = w_{ki}$ for all possible i and k.

On the distributed property of the GTNN model, we have the following remark.

Remark 3: The proposed GTNN model (21) can be implemented in a distributed fashion as the solution of $\dot{\theta}_i$ and the update of λ_i in (21) only require information from the neighborhood of the ith manipulator and no multihop message passing is required. As a distributed algorithm, GTNN is advantageous in communication burdens and robustness in comparison with centralized and decentralized control schemes. For centralized schemes, the central station is a fragile point, the failure of which may results in the global failure. However, GTNN as a distributed scheme still works well even when some manipulators fail to work. In comparison with decentralized control schemes, which usually require all-to-all communication for information exchange, the presented GTNN as a distributed one only needs neighbor-to-neighbor communication and saves overall communication expenses. This neighbor-to-neighbor communication nature makes GTNN scalable to large-scale networks with a huge number of manipulators involved.

In addition, it is noteworthy that GTNN bears the same module dynamics for all manipulators, i.e., both the state equation and the output equation have the same expression for all i. There is no hierarchy for different manipulators. A manipulator does not need to explicitly differentiate its neighbors. This ensues the peer-to-peer nature of this control algorithm.

B. Model Discussion

As pointed out in Section IV-A, there is no hierarchy among manipulators for the control. This situation is analogous to the scenario when a group of people moves big furniture and none of them knows who is the leader except for the leader himself. It often happens that the line of sight is blocked by the furniture and each person can only see the movement of his/her neighbors. For the person neighboring the leader, as he/she does not know which person is the leader, information from nonleader persons, which may be temporarily misleading, is also considered for decision making. In summary, for such a problem, there are two challenges: 1) no one knows the desired moving direction except the leader and 2) each person can only observe the movement of his/her neighbor. However, even in such a tough situation, such a group of people is still able to complete the task by their collaboration. In addition, the structural property of this problem that only information from neighbors is required for decision making makes this collaboration structure extendable to the collaboration among a large number of agents, as often found in flocking, migration, and so on. The problem of peer-to-peer cooperation of networked manipulators investigated in this paper has similarities by viewing the command center (i.e., the virtual manipulator) as a virtual leader. We model the problem as a game and confine the opponent players in the local neighborhood. The derived dynamic module (21) associated with the ith manipulator only requires motion information from neighboring manipulators in $\mathcal{N}_i$ and no manipulator explicitly knows the desired motion. This model is expected to give insight to researchers to formulate a cooperation problem from a formal mathematical perspective with the property of distributed decision making, which is widely observable in nature [36]–[38].

V. Convergence of the Neurodynamics

The GTNN model (21) provides an iterative way to solve the game (5). In this section, we investigate the convergence of GTNN and the optimality of the neural solution to the original game.

Before stating our main results, we first give the following lemmas which will be used in the derivation of the convergence property.

Lemma 3 [39]: Let $f(t) : \mathbb{R} \to \mathbb{R}$ be differentiable and has a finite limit as $t \to \infty$. If $\dot{f}(t)$ is uniformly continuous, then $\dot{f}(t) \to 0$ as $t \to \infty$.

This lemma is well known as Barbalat's lemma and is widely used in convergence analysis.

The following lemma is also useful in the derivation of the convergence property.

Lemma 4: If solution exists for problem (5), the equation $J_i g(-J_i^T \lambda_i) = v_d$ for $i = 1, 2, \ldots, l$ relative to λ_i or equivalently in a compact form $\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) = \mathbf{1} \otimes v_d$ relative to $\bar{\lambda}$ with $\bar{\lambda}$, $\bar{J}$, and $\bar{g}(\cdot)$ defined in (23) and (24) has at least one feasible solution, provided that the communication network formed by the manipulators and the command center is connected.

Proof: To conclude the result, we only need to prove that the equation $J_i g(-J_i^T \lambda_i) = v_d$ has at least one solution if solution exists for the equation set (19) for $i = 1, 2, \ldots, l$ according to Lemma 2. According to Lemma 1, the constraint $v_i = \sum_{k\in\mathcal{N}_i} w_{ik} v_k / \sum_{k\in\mathcal{N}_i} w_{ik}$ in (5) can be replaced by $v_i = v_d$ without changing the solution. The resulting equations are written as

$$v_i = v_d \tag{27a}$$

$$\dot{\theta}_i = g\big(-J_i^T \lambda_i\big) \tag{27b}$$

$$v_i = J_i g\big(-J_i^T \lambda_i\big) \tag{27c}$$

for $i = 1, 2, \ldots, l$. With (27), we conclude

$$v_d = J_i g\big(-J_i^T \lambda_i\big) \quad \text{for } i = 1, 2, \ldots, l \tag{28}$$

meaning that the solution of (27), i.e., the solution of the game (5), under the condition that the communication network is connected, satisfies (28). This completes the proof. □

Theorem 1: The GTNN model (21) with $\epsilon > 0$ globally converges to the pure-strategy Nash equilibrium of the game (5) provided that the communication network formed by the manipulators and the command center is connected.

Proof: We use Barbalat's Lemma to prove the result based on the compact expression of the indirect neuronal dynamics (22). There are two steps in the proof.

Step 1: We construct a function $V(\bar{\lambda})$ and prove it is lower bounded and monotonic nonincreasing with time.

Step 2: We prove the convergence result based on the steady-state behavior of $V(\bar{\lambda})$.

In Step 1, we construct the following function:

$$V = -(\bar{J}^T\bar{\lambda})^T\bar{g}(-\bar{J}^T\bar{\lambda}) - \frac{1}{2}\bar{g}^T(-\bar{J}^T\bar{\lambda})\bar{g}(-\bar{J}^T\bar{\lambda}) + (\mathbf{1}\otimes v_d)^T\bar{\lambda}. \tag{29}$$

In this step, we first show that V defined above is lower bounded. According to Lemma 4, there exists at least a solution to the equation $\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) = \mathbf{1}\otimes v_d$. Denoting the solution as $\bar{\lambda} = \bar{\lambda}^*$, $V(\bar{\lambda})$ can be rewritten as

$$V = -(\bar{J}^T\bar{\lambda})^T\bar{g}(-\bar{J}^T\bar{\lambda}) - \frac{1}{2}\bar{g}^T(-\bar{J}^T\bar{\lambda})\bar{g}(-\bar{J}^T\bar{\lambda}) + \bar{g}^T(-\bar{J}^T\bar{\lambda}^*)\bar{J}^T\bar{\lambda}. \tag{30}$$

Defining $\bar{x} = -\bar{J}^T\bar{\lambda}$ and $\bar{x}^* = -\bar{J}^T\bar{\lambda}^*$, V becomes

$$V = \bar{x}^T\bar{g}(\bar{x}) - \frac{1}{2}\bar{g}^T(\bar{x})\bar{g}(\bar{x}) - \bar{x}^T\bar{g}(\bar{x}^*). \tag{31}$$

With (31), we further get

$$V = \bar{x}^T(\bar{g}(\bar{x}) - \bar{g}(\bar{x}^*)) - \frac{1}{2}\bar{g}^T(\bar{x})\bar{g}(\bar{x}) = (\bar{x} - \bar{g}(\bar{x}))^T \cdot (\bar{g}(\bar{x}) - \bar{g}(\bar{x}^*)) + \bar{g}(\bar{x})^T(\bar{g}(\bar{x}) - \bar{g}(\bar{x}^*)) - \frac{1}{2}\bar{g}^T(\bar{x})\bar{g}(\bar{x}). \tag{32}$$

Since the saturation operation $\bar{g}(\bar{x})$ can be viewed as a projection of $\bar{x}$ onto the compact convex set Ω^l with Ω being a box constraint $\Omega = \{x, \eta^- \le x \le \eta^+\}$, i.e., $\bar{g}(\bar{x}) = \mathrm{Proj}_{\Omega^l}\bar{x}$, where Proj$(\cdot)$ represents the projection operator. We thus get $(\bar{x} - \bar{g}(\bar{x}))^T(\bar{g}(\bar{x}) - \bar{g}(\bar{x}^*)) = (\bar{x} - \mathrm{Proj}_{\Omega^l}\bar{x})^T(\mathrm{Proj}_{\Omega^l}\bar{x} - \mathrm{Proj}_{\Omega^l}\bar{x}^*) \ge 0$ according to the property of projection to a convex set [40]. Accordingly, we get

$$V \ge \bar{g}(\bar{x})^T(\bar{g}(\bar{x}) - \bar{g}(\bar{x}^*)) - \frac{1}{2}\bar{g}^T(\bar{x})\bar{g}(\bar{x}) = \mathrm{Proj}_{\Omega^l}(\bar{x})^T \cdot (\mathrm{Proj}_{\Omega^l}(\bar{x}) - \mathrm{Proj}_{\Omega^l}(\bar{x}^*)) - \frac{1}{2}\|\mathrm{Proj}_{\Omega^l}(\bar{x})\|^2. \tag{33}$$

Noting that $\mathrm{Proj}_{\Omega^l}(\bar{x}) \in \Omega^l$ and recalling that Ω^l is compact, we conclude that there exists $c_0 > 0$, such that $\|\mathrm{Proj}_{\Omega^l}(\bar{x})\| < c_0$ for any $\bar{x}$. Thus, $\|\mathrm{Proj}_{\Omega^l}(\bar{x}^*)\| < c_0$ also holds. Together with (33), we get

$$\begin{aligned} V &\ge -\|\mathrm{Proj}_{\Omega^l}(\bar{x})\|\|\mathrm{Proj}_{\Omega^l}(\bar{x}) - \mathrm{Proj}_{\Omega^l}(\bar{x}^*))\| - \frac{1}{2}\|\mathrm{Proj}_{\Omega^l}(\bar{x})\|^2 \\ &\ge -\|\mathrm{Proj}_{\Omega^l}(\bar{x})\|(\|\mathrm{Proj}_{\Omega^l}(\bar{x})\| + \|\mathrm{Proj}_{\Omega^l}(\bar{x}^*))\|) - \frac{1}{2}\|\mathrm{Proj}_{\Omega^l}(\bar{x})\|^2 \ge -2c_0^2 - \frac{1}{2}c_0^2 = -\frac{5}{2}c_0^2. \end{aligned} \tag{34}$$

Inequality (34) means that V is indeed lower bounded. Now, we turn to prove that V is monotonic nonincreasing. We first express the gradient of V in terms of $\bar{\lambda}$. After an equivalent transformation, (29) can be written as

$$V = -\frac{\|-\bar{J}^T\bar{\lambda} - \bar{g}(-\bar{J}^T\bar{\lambda})\|^2}{2} + \frac{\|-\bar{J}^T\bar{\lambda}\|^2}{2} + (\mathbf{1}\otimes v_d)^T\bar{\lambda}. \tag{35}$$

The expression $\|-\bar{J}^T\bar{\lambda} - \bar{g}(-\bar{J}^T\bar{\lambda})\|$ can be regarded as the distance between $-\bar{J}^T\bar{\lambda}$ and its projection $\mathrm{Proj}_{\Omega^l}(-\bar{J}^T\bar{\lambda})$, and thus the gradient of $1/2\|-\bar{J}^T\bar{\lambda} - \bar{g}(-\bar{J}^T\bar{\lambda})\|^2$ relative to $-\bar{J}^T\bar{\lambda}$ can be expressed as $\nabla_{-\bar{J}^T\bar{\lambda}}1/2\|-\bar{J}^T\bar{\lambda} - \bar{g}(-\bar{J}^T\bar{\lambda})\|^2 = -\bar{J}^T\bar{\lambda} - \bar{g}(-\bar{J}^T\bar{\lambda})$ according to [41]. Therefore, the gradient of V relative to $\bar{\lambda}$ can be expressed as

$$\begin{aligned} \nabla_{\bar{\lambda}}V &= -\bar{J}\bar{J}^T\bar{\lambda} - \bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) + \bar{J}\bar{J}^T\bar{\lambda} + \mathbf{1}\otimes v_d \\ &= -\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) + \mathbf{1}\otimes v_d. \end{aligned} \tag{36}$$

Computing time derivative of V in (29) along the system trajectory (22a) yields

$$\begin{aligned} \dot{V} &= (\nabla_{\bar{\lambda}}V)^T\dot{\bar{\lambda}} \\ &= \frac{1}{\epsilon}(-\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) + \mathbf{1}\otimes v_d)^T((L\otimes I_m)\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) \\ &\quad + (\Lambda\otimes I_m)(\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d)). \end{aligned} \tag{37}$$

Recall that $L\mathbf{1} = 0$ always holds for a Laplacian matrix L [34] and thus $(L\otimes I_m)(\mathbf{1}\otimes v_d) = (L\mathbf{1})\otimes(I_m v_d) = 0$. Accordingly, we have

$$\begin{aligned} \dot{V} &= \frac{1}{\epsilon}(-\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) + \mathbf{1}\otimes v_d)^T \\ &\quad \cdot ((L\otimes I_m + \Lambda\otimes I_m)\cdot(\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d)) \\ &= -\frac{1}{\epsilon}(\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d)^T(L\otimes I_m + \Lambda\otimes I_m) \\ &\quad \cdot(\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d) \le 0. \end{aligned} \tag{38}$$

Note that the last inequality in (38) holds, since the Laplacian matrix L is positive semidefinite and the diagonal matrix Λ is also positive semidefinite (all its diagonal elements are either positive or zero). With (38), we conclude that V is monotonic nonincreasing.

In Step 2, based on the conclusions drawn in Step 1 that V is lower bounded and monotonic nonincreasing, we conclude that V has a finite limit as $t \to \infty$. In addition, note that $\dot{V}$ in (38) is uniformly continuous. The conditions required by Barbalat's lemma are satisfied and thus we conclude that $\dot{V} \to 0$ as $t \to \infty$ according to Lemma 3. Now, we turn to examine the solution of $\dot{V} = 0$, which solves the steady-state solution of the neural dynamics. Letting $\dot{V} = 0$ in (38) yields

$$\bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d = 0. \tag{39}$$

Since $(L\otimes I_m + \Lambda\otimes I_m)$ is positive definite (this is equivalent to the fact that $L + \Lambda$ is positive definite), which can be verified by the fact that both L and Λ are positive semidefinite and $x^T(L+\Lambda)x = 0$ is equivalent to $x^TLx = 0$ together with $x^T\Lambda x = 0$, whose solution is $x = 0$ only if there exists a nonzero diagonal element in Λ. Therefore, we conclude that the neural network converges to a solution that satisfies

$$\begin{aligned} \bar{J}\bar{g}(-\bar{J}^T\bar{\lambda}) - \mathbf{1}\otimes v_d &= 0 \\ \dot{\bar{\theta}} &= \bar{g}(-\bar{J}^T\bar{\lambda}). \end{aligned} \tag{40}$$

According to Lemmas 1 and 2, the solution of the game (5) is equivalent to the solution of the following equation set:

$$v_i = v_d \tag{41a}$$
$$\dot{\theta}_i = g\big(-J_i^T \lambda_i\big) \tag{41b}$$
$$v_i = J_i g\big(-J_i^T \lambda_i\big) \tag{41c}$$

for $i = 1, 2, \ldots, l$, which is also equivalent to the solution of (40). This result means that the neural network (22) converges to the solution of the game (5). This completes the proof. □

Remark 4: In this paper, the cooperative control of multi-manipulators is modeled as a constrained game, in which each manipulator as a game player minimizes its local costs greedily based on information from its neighbors. One fundamental question naturally arises: is there any global minimum that can be achieved through the selfish gaming among multiple manipulators? Particular for the dynamics of the networked manipulators using the proposed GTNN (21), we give a positive answer to this question in Theorem 1. The proof of Theorem 1 implies the deliberately designed Lyapunov function (29) is indeed such a global cost, which bridges the local selfish dynamics of individual manipulators and the emergence of a global optimality.

VI. Simulations

The GTNN model gives a general framework for kinematic control of multiple redundant manipulators in a distributed fashion. Many multiple manipulator applications can be put into the proposed control framework. For example, applications, such as coordinate welding [4], dextrous grasping [3], and remote surgery with multiple manipulators [42], [43], can be viewed as the cooperative target tracking problem and the goal is to control the end effectors of multiple manipulators to simultaneously track a desired trajectory. In addition, problems, such as the gait control of multiple legged robots, can also be solved with GTNN by regarding each leg as a redundant manipulator and the robot body as a payload to be transported [44]. Moreover, the attitude control of telescope array [45] used for a fine view of stars can also be implemented in the GTNN framework by treating the driven mechanism of each element antenna as a manipulator and setting the focus of this telescope as the desired reference point. In the simulation, a network of manipulators under a randomly generated connected graph for communication is considered to test the effectiveness of the proposed neural networks.

A. Simulation Setup

In this simulation, we consider ten PUMA 560 manipulators for cooperative payload transport. The PUMA 560 is a 6-DOF manipulator (parameters for PUMA 560 manipulator are summarized in Table II). The end effector of the manipulator can reach any position at a given orientation within its workspace and thus is a redundant manipulator when considering to control the 3-D position of the end effector. The desired motion issued by the command center is to track a circular path with radius 0.1 m at an angular speed 0.2 rd/s around its center. The communication topology between the manipulators and the command center is shown in Fig. 3. As observed in Fig. 3, only manipulators 3, 5, 8, and 10 are able to access the desired motion provided by the command center as they are in its neighborhood. Manipulators exchange information by following the communication topology, as shown in Fig. 3. The neural network parameters are chosen as $\epsilon = 1 \times 10^{-4}$ and $\eta^+ = -\eta^- = 0.3$.

TABLE II
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	0.10000

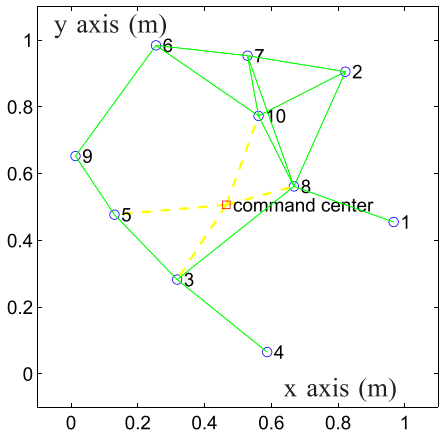


Fig. 3. Information topology for cooperative payload transport by ten manipulators. The circles, square, solid lines, dashed lines, and numbers aside the circles represent manipulators, command center, communication topology, the connection to the command center, and the labels of nodes, respectively.

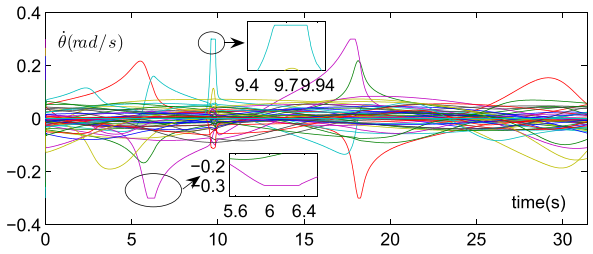


Fig. 4. Time history of the angular velocity $\dot{\theta}$ for all manipulators.

B. Simulation Results

A typical simulation run for one period $T = 2\pi/0.2 = 31.42$ s is conducted by initializing parameters randomly. Fig. 4 shows the control action $\dot{\theta}$ computed by the

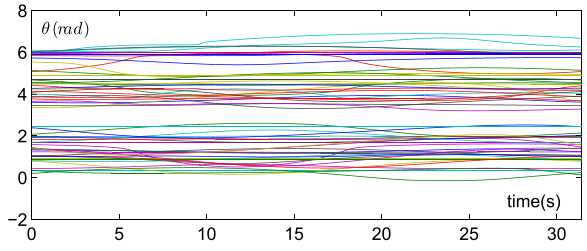


Fig. 5. Time history of the joint angle θ for all manipulators.

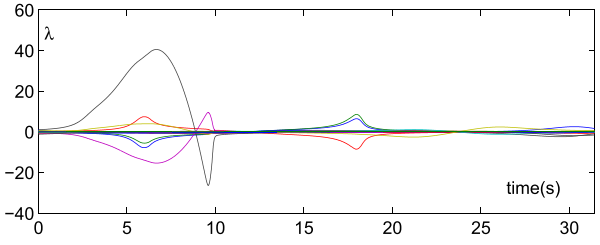


Fig. 6. Time history of the costate variable λ for all manipulators.

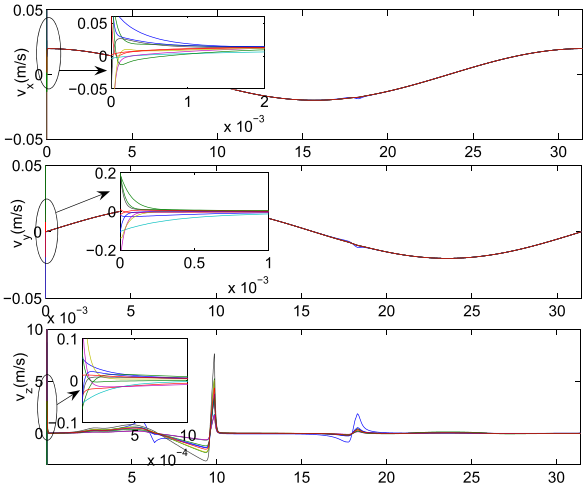


Fig. 7. Time history of the workspace velocity for all manipulators.

neural controller. It is observable in Fig. 4 that the resulting $\dot{\theta}$ is compliant with the constraint $\eta^- \leq \dot{\theta} \leq \eta^+$. When some elements of $\dot{\theta}$ tend to exceed the range $[\eta^-, \eta^+]$ between some time intervals (see small windows in Fig. 4), their values are saturated to ensure the validity of the constraint. The time profiles of the joint angle θ, the costate variable λ, and the workspace velocity for all manipulators are plotted in Figs. 5–7, respectively. As shown in Fig. 7, after a short transition, all manipulators successfully reach the same workspace velocity through neighbor-to-neighbor information exchanging. As further demonstrated in Fig. 8, in spite of different initial configurations for all manipulators, they all achieve the same desired circular trajectory (due to space limitation, only four manipulators are drawn to show their end-effector trajectories and configurations with time). As further shown in Figs. 9 and 10, using GTNN, the velocity tracking error converges to around zero very fast and the position tracking error remains at a tiny value during the period of simulation.

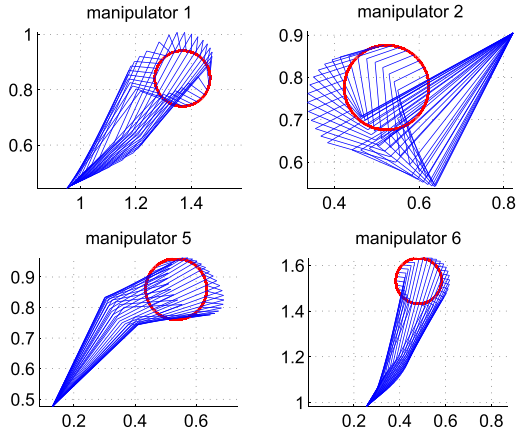


Fig. 8. Trajectories of the end effector (red curve) and the manipulator configuration at different time steps (blue lines).

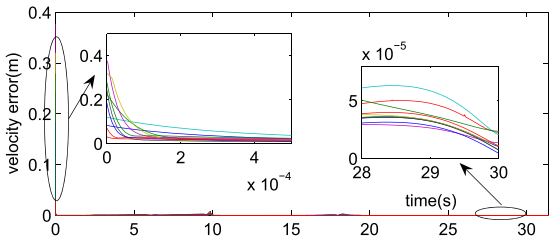


Fig. 9. Time history of the velocity error in workspace.

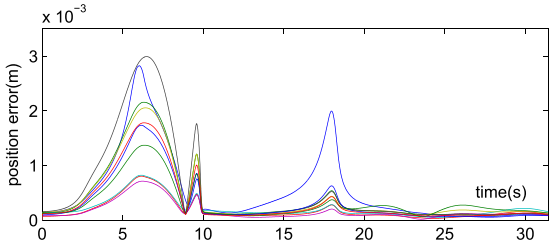


Fig. 10. Time history of the position error in workspace.

VII. Conclusion

In this paper, the problem of multiple redundant manipulator cooperative control is formulated as a constrained game under topological restriction for neighbor information exchanging. A recurrent neural network called GTNN derived from the dual-space analysis is proposed to recursively solve the Nash equilibrium of the formulated game. The global stability of this neural model and its optimality to solve the game are both proved in the theory. A numerical experiment that simulates cooperative payload transport with ten manipulators is conducted. Results from the simulation show that the effectiveness of the proposed method. Before ending this paper, it is worth mentioning that it is the first time to extend dual-space recurrent neural network modeling from optimization perspective to constrained gaming perspective, and to apply the approach to redundancy resolution of multimanipulator coordination with local information exchanging, which constructs a major breakthrough in the research of recurrent neural networks for control.

References

[1] G. Montemayor and J. T. Wen, "Decentralized collaborative load transport by multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 372–377.

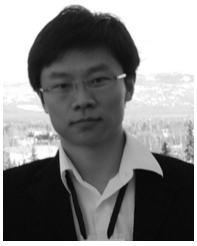
[2] J.-C. Fraile, C. J. J. Paredis, C.-H. Wang, and P. K. Khosla, "Agent-based planning and control of a multi-manipulator assembly system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jun. 1999, pp. 1219–1225.
[3] G. Liu, J. Xu, X. Wang, and Z. Li, "On quality functions for grasp synthesis, fixture planning, and coordinated manipulation," *IEEE Trans. Autom. Sci. Eng.*, vol. 1, no. 2, pp. 146–162, Oct. 2004.
[4] L. Wu, K. Cui, and S. B. Chen, "Redundancy coordination of multiple robotic devices for welding through genetic algorithm," *Robotica*, vol. 18, no. 6, pp. 669–676, 2000.
[5] Z. P. Wang, T. Zhou, Y. Mao, and Q. J. Chen, "Adaptive recurrent neural network control of uncertain constrained nonholonomic mobile manipulators," *Int. J. Syst. Sci.*, vol. 45, no. 2, pp. 133–144, 2014.
[6] Y. Zhang, S. S. Ge, and 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.*, vol. 34, no. 5, pp. 2126–2132, Oct. 2004.
[7] L. Xiao and Y. Zhang, "A new performance index for the repetitive motion of mobile manipulators," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 280–292, Feb. 2014.
[8] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 658–667, May 2003.
[9] Z. Zhang and Y. Zhang, "Variable joint-velocity limits of redundant robot manipulators handled by quadratic programming," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 674–686, Apr. 2013.
[10] H. Ding and J. Wang, "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 3, pp. 269–276, May 1999.
[11] W. S. Tang and 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.*, vol. 31, no. 1, pp. 98–105, Feb. 2001.
[12] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 1, pp. 54–64, Feb. 2005.
[13] B. Cai and Y. Zhang, "Different-level redundancy-resolution and its equivalent relationship analysis for robot manipulators using gradient-descent and Zhang's neural-dynamic methods," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3146–3155, Aug. 2012.
[14] A. M. Mohammed and S. Li, "Dynamic neural networks for kinematic redundancy resolution of parallel Stewart platforms," *IEEE Trans. Cybern.*, vol. 17, no. 3, pp. 1400–1410, Jul. 2015.
[15] D. Guo and Y. Zhang, "Acceleration-level inequality-based man scheme for obstacle avoidance of redundant robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6903–6914, Dec. 2014.
[16] U.-P. Wen, K.-M. Lan, and H.-S. Shih, "A review of Hopfield neural networks for solving mathematical programming problems," *Eur. J. Oper. Res.*, vol. 198, no. 3, pp. 675–687, 2009.
[17] Y. Xia and J. Wang, "A recurrent neural network for solving nonlinear convex programs subject to linear constraints," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 379–386, Mar. 2005.
[18] L. Xiao, "A finite-time convergent neural dynamics for online solution of time-varying linear complex matrix equation," *Neurocomputing*, vol. 167, pp. 254–259, Nov. 2015.
[19] Q. Liu and J. Wang, "A one-layer recurrent neural network for constrained nonsmooth optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1323–1333, Oct. 2011.
[20] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1229–1262, Jul. 2014.
[21] L. Xiao and R. Lu, "Finite-time solution to nonlinear equation using recurrent neural dynamics with a specially-constructed activation function," *Neurocomputing*, vol. 151, pp. 246–251, Mar. 2015.
[22] X. Hu and J. Wang, "An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2022–2031, Dec. 2008.
[23] X. Hu and B. Zhang, "A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem," *IEEE Trans. Neural Netw.*, vol. 20, no. 4, pp. 654–664, Apr. 2009.
[24] S. Li, Y. Li, and Z. Wang, "A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application," *Neural Netw.*, vol. 39, pp. 27–39, Mar. 2013.
[25] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500–1510, Nov. 2006.
[26] Z. Yan and J. Wang, "Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks," *IEEE Trans. Ind. Informat.*, vol. 8, no. 4, pp. 746–756, Nov. 2012.
[27] L. Jin and Y. Zhang, "G2-type SRMPC scheme for synchronous manipulation of two redundant robot arms," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 153–164, Feb. 2015.
[28] S. Li, S. Chen, B. Liu, Y. Li, and Y. Liang, "Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks," *Neurocomputing*, vol. 91, pp. 1–10, Aug. 2012.
[29] S. Li, H. Cui, Y. Li, B. Liu, and Y. Lou, "Decentralized control of collaborative redundant manipulators with partial command coverage via locally connected recurrent neural networks," *Neural Comput. Appl.*, vol. 23, no. 3, pp. 1051–1060, 2012.
[30] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, vol. 3. New York, NY, USA: Wiley, 2006.
[31] B. Armstrong, O. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3. Apr. 1986, pp. 510–518.
[32] G. Szabó and G. Fáth, "Evolutionary games on graphs," *Phys. Rep.*, vol. 446, nos. 4–6, pp. 97–216, Jul. 2007.
[33] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. San Diego, CA, USA: Academic, 1999.
[34] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*. New York, NY, USA: Wiley, 1991, pp. 871–898.
[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[36] R. M. Raafat, N. Chater, and C. Frith, "Herding in humans," *Trends Cognit. Sci.*, vol. 13, no. 10, pp. 420–428, 2009.
[37] J. Toner and Y. Tu, "Flocks, herds, and schools: A quantitative theory of flocking," *Phys. Rev. E*, vol. 58, no. 4, p. 4828, 1998.
[38] E. Karsenti, "Self-organization in cell biology: A brief history," *Nature Rev. Molecular Cell Biol.*, vol. 9, no. 3, pp. 255–262, 2008.
[39] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1991.
[40] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*. Palo Alto, CA, USA: Meboo Publishing, 2006.
[41] G. Mastroeni, "Gap functions and descent methods for minty variational inequality," in *Optimization and Control With Applications* (Applied Optimization), L. Qi, K. Teo, and X. Yang, Eds. New York, NY, USA, 2005, pp. 529–547.
[42] J. Marescaux *et al.*, "Transcontinental robot-assisted remote telesurgery: Feasibility and potential applications," *Ann. Surgery*, vol. 235, no. 4, pp. 487–492, 2002.
[43] A. P. Kypson, L. W. Nifong, and W. R. Chitwood, "Robotic cardiac surgery," *J. Long-Term Effects Med. Implants*, vol. 13, no. 6, pp. 451–464, 2003.
[44] Y. Aiyama, M. Hara, T. Yabuki, J. Ota, and T. Arai, "Cooperative transportation by two four-legged robots with implicit communication," *Robot. Auto. Syst.*, vol. 29, no. 1, pp. 13–19, 1999.
[45] J. Welch *et al.*, "The Allen telescope array: The first widefield, panchromatic, snapshot radio camera for radio astronomy and SETI," *Proc. IEEE*, vol. 97, no. 8, pp. 1438–1447, Aug. 2009.

Shuai Li received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, Hefei, China, in 2005, the M.E. degree in automatic control engineering from the University of Science and Technology of China, Hefei, in 2008, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2014.

He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include control of dynamic networks, distributed estimation and control, and coordination of multiple robots.

Jinbo He received the B.S. degree in internet and multimedia technologies from The Hong Kong Polytechnic University, Hong Kong.

He is currently a Research Assistant with The Hong Kong Polytechnic University, Hong Kong. His current research interests include computer graphics, artificial intelligence, neural networks, and distributed system.



Yangming Li received the B.E. and M.E. degrees in computer science and engineering from the Hefei University of Technology, Hefei, China, and the Ph.D. degree in automatic control engineering from the University of Science and Technology of China, Hefei.

He is currently a Research Associate with the BioRobotics Laboratory, Electrical Engineering Department, University of Washington, Seattle, WA, USA. He also served as an Associate Research Fellow (Associate Professor) with the Robot Sensor and Human–Machine Interaction Laboratory, Institute of Intelligent Machines, Chinese Academy of Sciences, Beijing, China. His current research interests include improve precision, efficiency, and robustness of mobile robots and surgical robot, through probabilistic methods and neural networks.

Muhammad Usman Rafique received the bachelor's and master's degrees in mechatronics engineering from the National University of Sciences and Technology, Karachi, Pakistan, in 2007 and 2010, respectively. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

He was with the National Space Agency of Pakistan (SUPARCO), Karachi, from 2007 to 2008. From 2010 to 2015, he was a Lecturer of Mechatronics Engineering with Air University, Islamabad, Pakistan. His current research interests include mobile robots, manipulation, motion planning, machine learning, computer vision, and control systems.