Chapter 2
Neural Networks for Robot Arm Cooperation with a Star Control Topology

2.1 Introduction

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

A fundamental issue in multiple redundant manipulator control is the redundancy resolution problem, which provides feasible solutions in the joint space to a task in the Cartesian space. Conventionally, the general solution of redundancy resolution is obtained by solving a set of redundant time-varying linear equations [5, 6]. However, as pointed in [7, 8], this type of method cannot generate a repeatable solution and the drift of joint angles is unavoidable. Authors in [9] formulated the problem of single redundant manipulator kinematic control as a constrained optimization problem and opened an avenue to study the redundancy resolution problem using optimization based methods [10, 11]. In [12], a penalty term was introduced into the objective function to restrict the solution inside a physically feasible range. This method enables direct monitoring and control of the magnitudes of the individual joint torques. However, the solution is merely an approximate solution of the problem and the approximation error strongly depends on the coefficient of the penalty term. To fix this problem, in [13], the optimization problem was studied in the dual space and dual neural networks are developed to solve the problem in real time. Further performance improvements of the dual neural network approach, such as convergence time, architecture complexity, etc., were obtained in the successive studies [14–16]. In [17], a local optimization approach was proposed to

S. Li and Y. Zhang, *Neural Networks for Cooperative Control of Multiple Robot Arms*, SpringerBriefs in Computational Intelligence,
https://doi.org/10.1007/978-981-10-7037-2_2

resolve the kinematic control problem of redundant manipulators. This approach is applicable to either serial manipulators or parallel manipulators but the result is a locally optimal one instead of globally. In [18], an optimal real-time redundancy resolution scheme was proposed to solve the problem. The optimal control law can be derived in real-time using adaptive critic framework after a period of off-line training. An analytic-iterative solution was presented in [19] to solve the redundancy resolution problem formulated as an inequality constrained convex optimization problem. The optimization based formulation framework also allows us to analyze the multiple redundant manipulator cooperative task execution problem. However, the framework does not provide any information about how to design a decentralized control strategy not requiring explicit information from other manipulators.

In this chapter, we consider the cooperative task execution problem [20]. The problem is formulated as a separable constrained quadratic programming problem. Benefiting from the separable property of our formulation, this problem can be decoupled into a collection of subproblems, each of which corresponds to a single manipulator. Inspired by the real-time optimization capability of recurrent neural networks [21–28], a dual recurrent neural network, which has separate modules associated with every manipulator in the collection, is designed to solve the problem in real time. To the best of our knowledge, this is the first time that the dual recurrent neural network method [29–31] is extended to the decentralized control of multiple redundant manipulator systems.

In contrast to the great success of recurrent dual neural network based kinematic control of a single redundant manipulator, there has been very little attention to the solution on cooperative control of collaborative redundant manipulators employing dual neural networks. In this background, the dual neural network approach is extended to solve the kinematic control problem of multiple redundant manipulators. Compared to existing dual neural networks [21, 23, 25, 26, 29–36], a prominent feature of the neural network presented in this chapter lies in its different architecture, which is composed of separate modules associated with every manipulator. This feature enables a completely decentralized control of multiple manipulators. The idea in this chapter is also in part inspired by the recently reported results on control, neural networks, and distributed systems [43–71].

2.2 Problem Formulation

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

2.2.1 Redundant Manipulator Kinematics

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

$$r(t) = f(\theta(t)), \tag{2.1}$$

where $r(t) \in \mathbb{R}^n$ and $\theta(t) \in \mathbb{R}^m$ with $m > n$ are the coordinate of the manipulator in the Cartesian space at time t and the coordinate in the joint space, respectively. The mapping $f(\cdot)$ is a nonlinear function with known parameters for a given manipulator. Calculating time derivative on both sides of (2.1) yields,

$$\dot{r}(t) = J(\theta(t))\dot{\theta}(t), \tag{2.2}$$

where $\dot{r}(t) \in \mathbb{R}^n$ and $\dot{\theta}(t) \in \mathbb{R}^m$ are the velocity of the manipulator in the Cartesian space and that in the joint space, respectively. $J(\theta(t)) = (\partial f(\theta(t)))/(\partial\theta(t))$ is the Jacobian matrix. Apparently, $\dot{r}(t)$ is an affine function with respect to $\dot{\theta}(t)$ and is relatively easier to deal with compared with the nonlinear mapping (2.1).

In robotics, the problem of velocity inverse kinematics is concerned with finding a solution $\dot{\theta}(t)$ for the manipulator model (2.2) with given desired velocity $\dot{r}(t)$ in Cartesian space and known manipulator model $J(\cdot)$. For redundant manipulators, $n < m$, which means the number of equalities in the velocity inverse kinematic problem is less than the number of dimensions of the decision variable $\dot{\theta}(t)$, and therefore normally the solution to this problem is not unique. This property of redundant manipulators enables us to select the best solution among all the feasible ones according to certain optimum criteria and extra constraints. Possible optimum criteria includes minimum of the Euclidian norm or the infinity norm of the joint velocity vector and possible constraints include joint angle limits, joint speed limits, etc.

2.2.2 Cooperative Task Execution by Using Multiple Manipulators

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

along the desired reference trajectory with a given absolute value, the first aspect of the task can be achieved by velocity control and the second aspect can be satisfied by steering all end-effectors with the same velocity in the Cartesian space. In a word, the two aspects of the task can be achieved by steering end-effectors of all manipulators with the same velocity $v_d(t)$. In equation,

$$J_i(\theta_i(t))\dot{\theta}_i(t) = v_d(t), \qquad \text{for } i = 1, 2, ..., k, \tag{2.3}$$

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, k denotes the number of manipulators. Without introducing confusions,(2.3) is abbreviated into the following form for easy reading,

$$J_i\dot{\theta}_i = v_d, \qquad \text{for } i = 1, 2, ..., k, \tag{2.4}$$

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 (2.3), respectively. As the velocity mapping from the joint space to the Cartesian space is affine as shown in (2.2), using velocity control of the collection of manipulators thoroughly simples the design, compared with the direct position control based on the nonlinear transformation (2.1). In addition, resulting from the redundancy property of redundant manipulators, the solution satisfying the above mentioned two aspects of the task is not unique, which allows us to take extra optimization criteria and constraints into account. Without loss of generality, we minimize the Euclidean norm squared of the joint velocities, i.e., $\sum_{i=1}^{k} \dot{\theta}_i^T\dot{\theta}_i$ and impose the joint velocity constraints $\eta^- \le \dot{\theta}_i \le \eta^+$, to exploit the extra design freedoms. In summary, the cooperative task execution problem can be formulated as follows,

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\textstyle\sum_{i=1}^{k} \dot{\theta}_i^T\dot{\theta}_i, \\ \text{subject to} \quad & J_1\dot{\theta}_1 = v_d, J_2\dot{\theta}_2 = v_d, ..., J_k\dot{\theta}_k = v_d, \\ & \eta^- \le \dot{\theta}_1 \le \eta^+, \eta^- \le \dot{\theta}_2 \le \eta^+, ..., \eta^- \le \dot{\theta}_k \le \eta^+, \end{aligned} \tag{2.5}$$

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

Remark 2.1 In the problem formulation (2.5), the kinematic constraints (or more exactly, joint velocity limits and reference velocity constraints) are considered. Constraints, such as joint limits and collision avoidance, however, are not considered explicitly in (2.5). Actually, as done in [31], the joint limit constraint can be included in the formulation (2.5) by imposing extra inequality constraints on $\dot{\theta}_i$ as follows,

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

where $\theta^+ \in \mathbb{R}^m$ and $\theta^- \in \mathbb{R}^m$ are the upper and lower limit of the allowed angle in the joint space, $c_0 \in \mathbb{R}$, $c_0 > 0$ is a coefficient. Similarly, collision avoidance can be taken into account by following the inequality-based obstacle avoidance QP method proposed in [37], which also imposes linear inequality constraint on $\dot{\theta}$.

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

2.3 Real-Time Control via Recurrent Neural Networks

In this section, a recurrent neural network is designed to complete the online optimization of the cooperative task execution problem (2.5).

According to Karash-Kuhn-Tucker (KKT) conditions [38], the solution to problem (2.5) satisfies,

$$\dot{\theta}_i = -J_i^T \lambda_i - \mu_i, \tag{2.6a}$$

$$J_i \dot{\theta}_i = v_d, \tag{2.6b}$$

$$\begin{cases} \dot{\theta}_i = \eta^+, & \text{if } \mu_i > 0, \\ \eta^- \leq \dot{\theta}_i \leq \eta^+, & \text{if } \mu_i = 0, \\ \dot{\theta}_i = \eta^-, & \text{if } \mu_i < 0, \end{cases} \tag{2.6c}$$

$$\text{for } i = 1, 2, ..., k.$$

Remark 2.2 Benefiting from the fact that the problem (2.5) is separable relative to $\theta_1, \theta_2, ..., \theta_k$, the decision variables $\theta_1, \theta_2, ..., \theta_k$ in (2.6) are completely decoupled. This means that for all manipulators, greedily minimizing their local cost function under local constraints (for the ith manipulator, the local cost is $\frac{1}{2}\dot{\theta}_i^T \dot{\theta}_i$ and the local constraints are $J_i \dot{\theta} = v_d$ and $\eta^- \leq \dot{\theta}_i \leq \eta^+$), leads to a collective behavior, which solves the global optimization problem (2.5). This property enables us to design a recurrent neural network with k separate modules, each of which control a single manipulator independently but still, all the modules together, collectively solve the cooperative task executive problem (2.5).

Note that (2.6c) can be simplified into the following form,

$$\dot{\theta}_i = g(\dot{\theta}_i + \mu_i), \qquad \text{for } i = 1, 2, ..., k, \tag{2.7}$$

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

$$g_j(x_j) = \begin{cases} \eta_j^+, & \text{if } x_j > \eta_j^+, \\ x_j, & \text{if } \eta_j^- \le x_j \le \eta_j^+, \\ \eta_j^-, & \text{if } x_j < \eta_j^-, \end{cases} \tag{2.8}$$

where η_j^- and η_j^+ are the jth elements of η^- and η^+, respectively. Substituting (2.6a) into (2.7) to cancel out $\dot{\theta}_i$ yields,

$$\mu_i = -g(-J_i^T \lambda_i) - J_i^T \lambda_i, \qquad \text{for } i = 1, 2, ..., k, \tag{2.9}$$

which means μ_i can be explicitly expressed in terms of λ_i for $i = 1, 2, ..., k$. $\dot{\theta}_i$ can also be expressed in terms of λ_i by substituting (2.9) into (2.6a),

$$\dot{\theta}_i = g(-J_i^T \lambda_i), \qquad \text{for } i = 1, 2, ..., k. \tag{2.10}$$

Plugging (2.7) into (2.6b) yields,

$$J_i g(\dot{\theta}_i + \mu_i) = v_d, \qquad \text{for } i = 1, 2, ..., k. \tag{2.11}$$

Canceling out $\dot{\theta}_i$ by bringing Eq. (2.6a) into Eq. (2.11) generates,

$$J_i g(-J_i^T \lambda_i) = v_d, \qquad \text{for } i = 1, 2, ..., k. \tag{2.12}$$

We use a recurrent neural network to solve $\lambda_1, \lambda_2, ..., \lambda_k$ in (2.12) as follows:

$$\epsilon_i \dot{\lambda}_i = J_i g(-J^T \lambda_i) - v_d, \qquad \text{for } i = 1, 2, ..., k, \tag{2.13}$$

where $\epsilon_i > 0$ for $i = 1, 2, ..., k$ is a scaling factor. Now, a recurrent neural network is obtained to solve the cooperative task execution problem (2.5). This neural network is composed of k modules corresponding to the k manipulators, and the state and output equations of the ith module for $i = 1, 2, ..., k$ are summarized as follows:

$$\text{state equation} \quad \epsilon_i \dot{\lambda}_i = J_i g(-J^T \lambda_i) - v_d, \tag{2.14a}$$

$$\text{output equation} \quad \dot{\theta}_i = g(-J_i^T \lambda_i), \tag{2.14b}$$

where $\epsilon_i > 0$ for $i = 1, 2, ..., k$ is a scaling factor, k is the number of manipulators, and the function $g(\cdot)$ is defined in (3.5).

The recurrent neural network for solving the cooperative task execution problem (2.5) consists of k modules, with k equal to the number of manipulators. Each mod-

ule is composed of n dynamic neurons, which equals the dimension of the Cartesian space. In a three-dimensional space, $n = 3$ for position control problem and $n = 6$ for the case with both position and orientation control. In each module, there are n dynamic neurons and $n(n-1)/2$ inter-connections between neurons. Therefore, for the whole neural network, the total number of neurons is kn and the interconnection complexity is $O(kn^2)$, which is polynomially dependent on the number of manipulators k and the dimension of the Cartesian space n. Besides the capability of solving the cooperative task execution problem in a decentralized manner, which cannot be handled by the conventional recurrent neural network based method for single manipulator motion control [26, 39, 40], another important distinction from them is that our model has a low architecture complexity and the complexity has no dependence on m, which is the DOF of the redundant manipulator. This property simplifies the hardware design for the analog circuit implementation of the neural network and reduces the cost, especially for redundant manipulators with a large m. Briefly, we have the following remark on this point.

Remark 2.3 The neural network presented in this chapter consists of k modules and the whole neural network has a total number of kn neurons and has an interconnection complexity $O(kn^2)$, which does not depend on the DOF of the redundant manipulator. In addition, The neural network solves the problem in a decentralized manner. This cannot be achieved by the conventional recurrent dual neural networks.

2.4 Theoretical Results

On the stability, solution optimality of the recurrent neural network (2.14) to the problem (2.5), we have conclusions stated in the following theorems.

Theorem 2.1 *The recurrent neural network (3.6a) with $\epsilon_i > 0$ for $i = 1, 2, ..., k$ is globally stable in the sense of Lyapunov and converges to an equilibrium point $[\lambda_1^{*T}, \lambda_2^{*T}, ..., \lambda_k^{*T}]^T$.*

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

$$V = \sum_{i=1}^{k} \frac{\epsilon_i}{2} \left(J_i g(-J_i^T \lambda_i) - v_d\right)^T \left(J_i g(-J_i^T \lambda_i) - v_d\right). \tag{2.15}$$

Note that V is a function of the state variables $\lambda_1, \lambda_2, ..., \lambda_k$. $V \geq 0$ and the equality holds when $J_i g(-J_i^T \lambda_i) - v_d = 0$ for all $i = 1, 2, ..., k$, i.e., the equality holds at the equilibrium point λ_i^* of (3.6a) for all $i = 1, 2, ..., k$. The time derivative of V along the neural network trajectory (2.14) can be obtained as follows:

$$\dot{V} = -\sum_{i=1}^{k} \epsilon_i \left(J_i g(-J_i^T \lambda_i) - v_d\right)^T \left(J_i D^+ g(-J_i^T \lambda_i) J_i^T \dot{\lambda}_i\right), \tag{2.16}$$

where $D^+ g(-J_i^T \lambda_i)$ denotes the upper right dini-derivative of the function $g(-J_i^T \lambda_i)$. According to the definition of $g(\cdot)$ in (3.5), $D^+ g(-J_i^T \lambda_i)$ is a diagonal matrix of the form $D^+ g(-J_i^T \lambda_i) = \text{diag}(c_{1i}, c_{2i}, ..., c_{mi})$ and the jth diagonal element c_{ji} is as follows,

$$c_{ji} = \begin{cases} 1, & \text{if } \eta_j^- \le d_{ji} < \eta_j^+, \\ 0, & \text{if } d_{ji} < \eta_j^- \text{ or } d_{ji} \ge \eta_j^+. \end{cases} \tag{2.17}$$

where d_{ji} represents the jth element of the vector $-J_i^T \lambda_i$. From (2.17), it is obtained that $c_{ji} \ge 0$ for all $j = 1, 2, ..., m$. Thus, we can express the diagonal matrix $\text{diag}(c_{1i}, c_{2i}, ..., c_{mi})$=$\text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, ..., \sqrt{c_{mi}})$ $\text{diag}^T(\sqrt{c_{1i}}, \sqrt{c_{2i}}, ..., \sqrt{c_{mi}})$. Based on this, further conclusion holds,

$$\begin{aligned} \dot{V} &= -\sum_{i=1}^{k} \Big(\left(J_i g(-J_i^T \lambda_i) - v_d\right)^T J_i \text{diag}(c_{1i}, c_{2i}, ..., c_{mi}) \\ &\quad \cdot J_i^T \left(J_i g(-J_i^T \lambda_i) - v_d\right) \Big) \\ &= -\sum_{i=1}^{k} \Big(\left(J_i g(-J_i^T \lambda_i) - v_d\right)^T J_i \text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, ..., \sqrt{c_{mi}}) \\ &\quad \cdot \text{diag}^T(\sqrt{c_{1i}}, \sqrt{c_{2i}}, ..., \sqrt{c_{mi}}) J_i^T \left(J_i g(-J_i^T \lambda_i) - v_d\right) \Big) \\ &= -\sum_{i=1}^{k} \left\| \text{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, ..., \sqrt{c_{mi}}) J_i^T \left(J_i g(-J_i^T \lambda_i) - v_d\right) \right\|^2 \\ &\le 0, \end{aligned} \tag{2.18}$$

which means the neural network (2.14) is globally stable to an equilibrium point $[\lambda_1^{*T}, \lambda_2^{*T}, ..., \lambda_k^{*T}]^T$ in the sense of Lyapunov [41]. This completes the proof.

The following Theorem reveals the relation between the equilibrium point of the neural network and the optimal solution to the cooperative task execution problem (2.5).

Theorem 2.2 *Let* $[\lambda_1^{*T}, \lambda_2^{*T}, ..., \lambda_k^{*T}]^T$ *be an equilibrium point of the neural network dynamic (2.14a). The output of this neural network at* $[\lambda_1^{*T}, \lambda_2^{*T}, ..., \lambda_k^{*T}]^T$, *which is* $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, ..., \dot{\theta}_k^{*T}]^T = [g^T(-J_1^T \lambda_1^*), g^T(-J_2^T \lambda_2^*), ..., g^T(-J_k^T \lambda_k^*)]^T$, *is the optimal solution to the cooperative task execution problem* (2.5).

Proof Since $[\lambda_1^{*T}, \lambda_2^{*T}, ..., \lambda_k^{*T}]^T$ is an equilibrium point of the ordinary differential equation (3.6a), and $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, ..., \dot{\theta}_k^{*T}]^T$ is the corresponding output, the following

holds for all $i = 1, 2, ..., k$,

$$J_i g(-J_i^T \lambda_i^*) = v_d,$$
$$\dot{\theta}_i^* = g(-J_i^T \lambda_i^*). \tag{2.19}$$

Defining new variables $\mu_i^* = -g(-J_i^T \lambda_i^*) - J_i^T \lambda_i^*$ for all $i = 1, 2, ..., k$, then $(\lambda_i^*, \mu_i^*, \dot{\theta}_i^*)$ is a solution to the equation set composed of Eqs. (2.9), (2.10) and (2.12). Due to the equivalence between the equation set consisting of (2.9), (2.10), (2.12) and the equation set (2.6), it is concluded that $(\lambda_i^*, \mu_i^*, \dot{\theta}_i^*)$ is also a solution to the KKT condition (2.6). The KKT condition (2.6) gives a sufficient and necessary condition to the constrained quadratic programming problem (2.5) [38], so $[\dot{\theta}_1^{*T}, \dot{\theta}_2^{*T}, ..., \dot{\theta}_k^{*T}]^T$ is the optimal solution to the problem (2.5). This completes the proof.

2.5 Simulation Results

In this section, the robot arm Puma 560 [42] is used as a testbed for the effectiveness of our method. The Puma 560 is a 6-DOF manipulator. The end-effector of the robot arm can reach any position at a given orientation within its workspace. In the following simulations, only the position control problem in a three-dimensional space is considered, so the Puma 560 arm, which has 6-DOF, is a redundant manipulator to this particular problem. The D-H parameters of the Puma 560 manipulator used in the simulation are summarized in Table 2.1.

In this section, the presented recurrent neural network model is applied to the kinematic control of multiple Puma 560 manipulators. Two simulations are performed: one studies cooperative transport with two manipulators and the other one studies cooperative tracking with three manipulators. In the simulations, only positioning of the reference point in three-dimensional space are considered, so $n = 3$. Each Puma 560 manipulator has 6 DOF ($m = 6$), and therefore for the two simulations, the degree of redundancy are 6 and 9, respectively.

Table 2.1 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

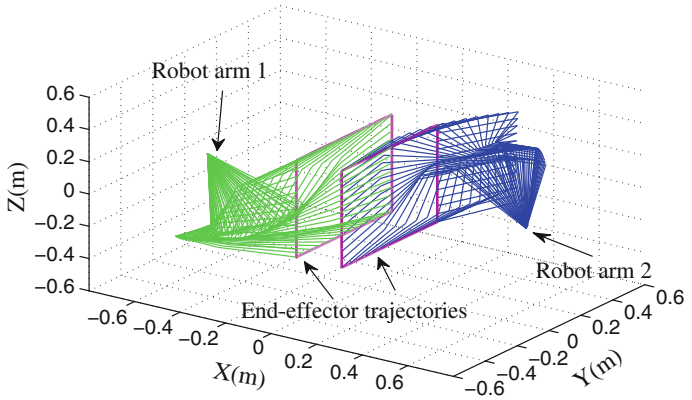
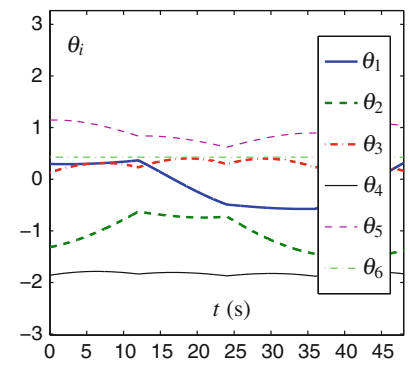


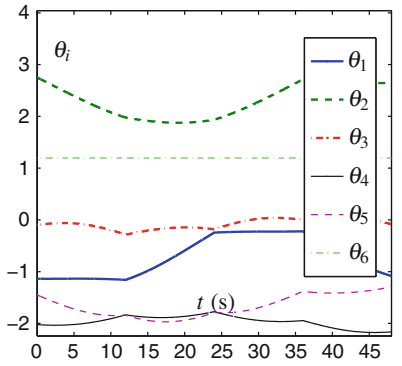
Fig. 2.1 Moving a payload with two manipulators in Sect. 2.5.1. In the figure, the green lines and the blue lines show the links of manipulator 1 and the links of manipulator 2 respectively in every seconds. The red lines show the trajectory of the end-effectors of manipulator 1 and that of manipulator 2, respectively

2.5.1 Payload Transport with Two Manipulators

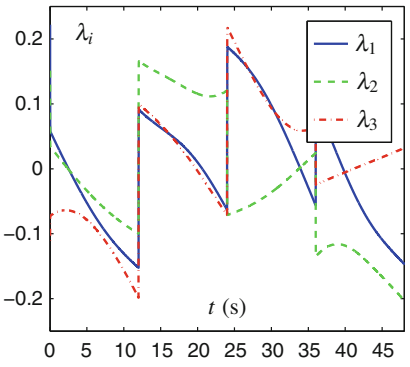
In this simulation example, we consider the problem of moving a payload along a square trajectory with a pair of manipulators. The D-H parameters of the two manipulators are summarized in Table 2.1. The bases of manipulator 1 and manipulator 2 locate at $[-0.7, 0, 0]$m and $[0.7, 0, 0]$m in the Cartesian space, respectively. The payload, centered at $[0, 0, 0]$m, has two handles at $[-0.1, 0, 0]$m and $[0.1, 0, 0]$m, which will be held by manipulator 1 and manipulator 2 for the movement. The desired motion of the reference point (the center of the payload) is from $[0, 0.3, -0.3]$m to $[0, 0.3, 0.3]$m, from $[0, 0.3, 0.3]$m to $[0, -0.3, 0.3]$m, from $[0, -0.3, 0.3]$m to $[0, -0.3, -0.3]$m, and from $[0, -0.3, -0.3]$m back to $[0, 0.3, -0.3]$m consecutively along straight lines with a constant speed 0.05 m/s. As to the neural network parameters, we choose $\epsilon_1 = \epsilon_2 = 10^{-3}$ and the upper and lower limits of the joint velocity are set to be $\eta^+ = [1, 1, 1, 1, 1, 1]^T$ and $\eta^- = -\eta^+$. Figure 2.1 illustrates motion trajectories of the two Puma 560 manipulators. The corresponding profiles for θ, λ and $\dot{\theta}$ are shown in Fig. 2.2. Note that the desired trajectory is a square, which has a sharp turn at each corner, and hence λ and $\dot{\theta}$ for both manipulators have a sharp change in values at the time 12, 24 and 36 s when the end-effector crosses the corners. To evaluate the control accuracy, the difference between the real position r_r and the desired position r_d of the reference point is used to measure the position error e_p and the difference between the real velocity v_r and the desired velocity v_d of the reference point is used to measure the velocity error e_v. That is, $e_p = r_r - r_d$ and $e_v = v_r - v_d$. According to the problem formulation, the real position of the reference point locates at the midpoint of the two end-effector positions and therefore the real velocity is the average of the two end-effector velocities, i.e., $r_r = \frac{r_1+r_2}{2}$ and $v_r = \frac{v_1+v_2}{2}$, where r_1, r_2, v_1, and v_2 represent the end-effector position of manipulator 1, the end-effector



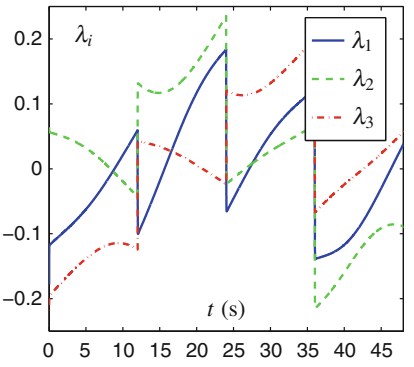
(a) Manipulator 1: θ



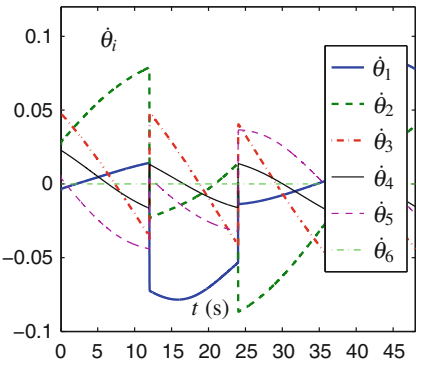
(b) Manipulator 2: θ



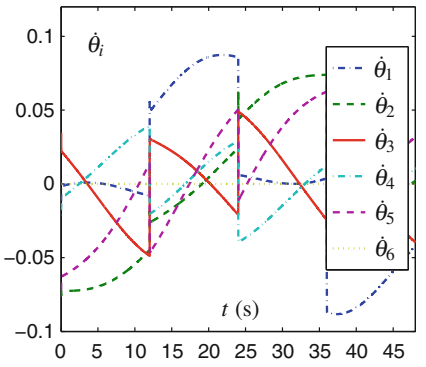
(c) Manipulator 1: λ



(d) Manipulator 2: λ



(e) Manipulator 1: $\dot{\theta}$



(f) Manipulator 2: $\dot{\theta}$

Fig. 2.2 Evolutions of θ, λ and $\dot{\theta}$ with time for the two manipulators in the simulation example in Sect. 2.5.1

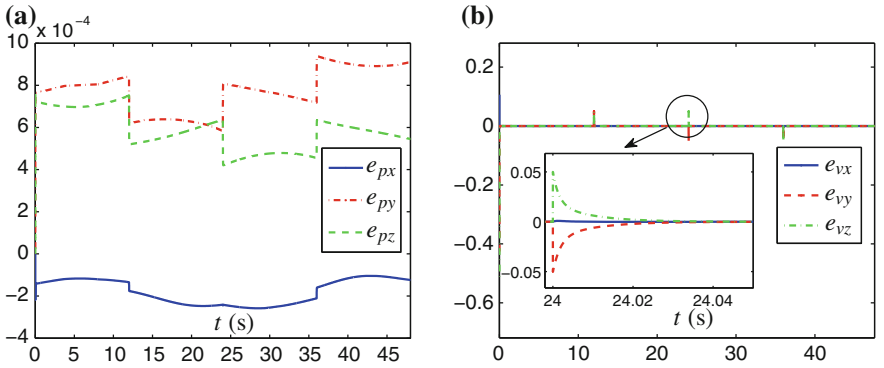


Fig. 2.3 Evolutions of position errors and velocity errors with time in the simulation example in Sect. 2.5.1

position of manipulator 2, the end-effector velocity of manipulator 1, and the end-effector velocity of manipulator 2, respectively. The evolution of the position error along x-axis (e_{px}), y-axis (e_{py}), z-axis (e_{pz}) and the evolution of the velocity error along x-axis (e_{vx}), y-axis (e_{vy}), z-axis (e_{vz}) are respectively plotted in Fig. 2.3. From the figure, it can be observed that the position error converges to less than 1mm in the duration of the simulation after a short transient at the very beginning. The velocity error converges very fast every time after the end-effector crosses a corner of the desired square trajectory.

2.5.2 *Cooperative Tracking with Three Manipulators*

In this simulation, we consider cooperative target tracking with three identical Puma 560 manipulators. As a particular example of cooperative task execution, potential applications include coordinate welding [2], dextrous grasping [1], etc. Simply speaking, the problem is to control the end-effectors of multiple manipulators to simultaneously track a desired trajectory. In this simulation, the bases of the three Puma 560 manipulators, manipulator 1, manipulator 2 and manipulator 3, locate at $[-0.5, 0.5, 0]$m $[-0.5, -0.5, 0]$m and $[0.7, 0, 0]$m in the Cartesian space, respectively. The desired trajectory is a circle centered at $[0, 0, 0]$m with diameter 0.4m 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. We choose $\epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-3}$, $\eta^+ = [1, 1, 1, 1, 1, 1]^T$ and $\eta^- = -\eta^+$ for the neural network. Figure 2.4 illustrates motion trajectories of the manipulators. From this figure, it can be observed that the end-effectors of the three manipulators track the desired circular trajectory simultaneously. The evolutions of θ, λ and $\dot{\theta}$ with time for the three manipulators are shown in Fig. 2.5. Since the desired trajectory is a circle, which is smooth, the value of θ, λ and $\dot{\theta}$ evolves smoothly with time. From Fig. 2.6, it can

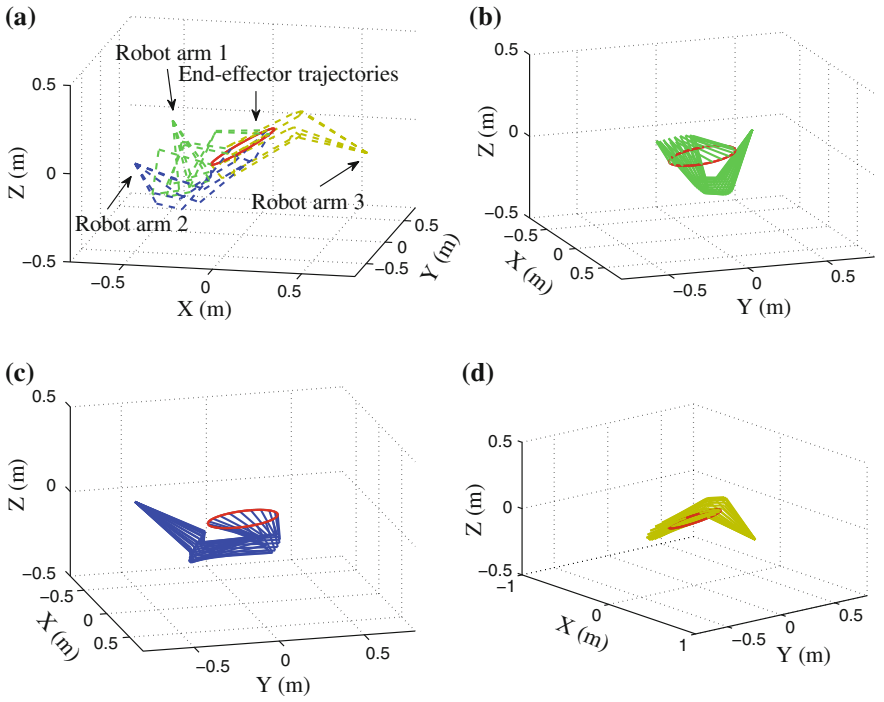
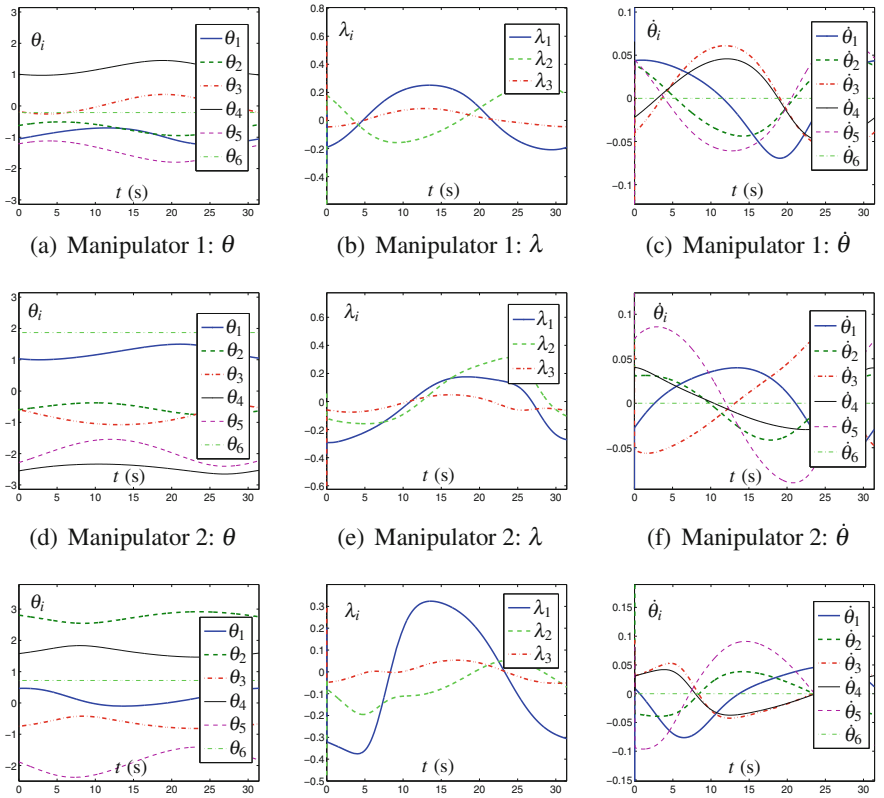


Fig. 2.4 Cooperative target tracking with three manipulators in Sect. 2.5.2 (only 5 configurations of each manipulator are plotted in (**a**) for easy recognition). In the figure, the green lines, the blue lines and the yellow lines show the links of manipulator 1, manipulator 2 and manipulator 3, respectively. The red lines show the trajectory of the end-effectors

be seen that the position error, measured with the position difference between the center point of the three end-effectors and the desired point, and the velocity error, measured with the difference between the velocity of the center point of the three end-effectors and the desired velocity, converge to a small value very fast after a short transient at the very beginning. During the simulation period, both the position error and the velocity error are less than 8×10^{-4}m and 10×10^{-5} m/s respectively in the three axial directions.

2.6 Summary

In this chapter, the problem of multiple redundant manipulator cooperative task execution is formulated as a quadratic programming problem. A recurrent neural network is presented to tackle the problem. Conventionally, this problem is solved by calculating the inverse kinematics by pseudo-inverting the Jacobian matrices. However, this treatment, on one hand, is computationally intensive and on the other hand, it



(a) Manipulator 1: θ (b) Manipulator 1: λ (c) Manipulator 1: $\dot{\theta}$

(d) Manipulator 2: θ (e) Manipulator 2: λ (f) Manipulator 2: $\dot{\theta}$

(g) Manipulator 3: θ (h) Manipulator 3: λ (i) Manipulator 3: $\dot{\theta}$

Fig. 2.5 Evolutions of $\theta, \lambda, \dot{\theta}$ with time for the manipulators in the simulation example in Sect. 2.5.2

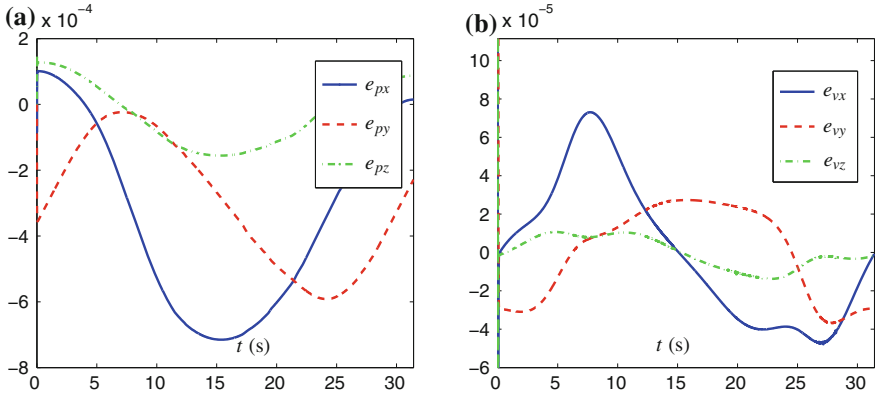


Fig. 2.6 Position errors and velocity errors for the cooperative target tracking in the simulation example in Sect. 2.5.2

generally results in a centralized control law of manipulators. Inspired by the great success of recurrent neural networks in realtime control of single manipulators, the neural network methodology is extended to solve the multiple redundant manipulator cooperative task execution problem. Compared with existing schemes using neural networks for the motion control of manipulators, the presented neural network is composed of independent modules, each of which controls a single manipulator and all of them together generate an emergent collective behavior to complete the cooperative task. Theoretical analysis proves the global stability of the neural network and the optimality of the solution. Application orientated simulations show that the presented method is effective and accurate.

References

1. 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
2. Wu L, Cui K, Chen SB (2000) Redundancy coordination of multiple robotic devices for welding through genetic algorithm. Robotica 18(6):669–676
3. Fraile J, Paredis C.J.J, 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. Montemayor G, Wen JT (2005) Decentralized collaborative load transport by multiple robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 372–377
5. Liegeois A (1977) Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Trans Syst Man Cybern 7(12):868–871
6. Khatib O, Bowling A (1996) Optimization of the inertial and acceleration characteristics of manipulators. In: Proceedings of IEEE International Conference on Robotics and Automation, pp 2883–2889
7. Klein CA, Huang CH (1983) Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Trans Syst Man Cybern 13(3):245–250
8. Shamir T, Yomdin Y (1988) Repeatability of redundant manipulators: mathematical solution of the problem. IEEE Trans Autom Control 33(11):1004–1009
9. Cheng FT, Sheu RJ, Chen TH (1995) The improved compact QP method for resolving manipulator redundancy. IEEE Trans Syst Man Cybern 25(11):1521–1530
10. Locke RCO, Patel RV (2007) Optimal remote center-of-motion location for robotics-assisted minimally-invasive surgery. In: Proceedings of IEEE International Conference on Robotics and Automation, pp 1900–1905
11. Cocuzza S, Pretto I, Debei S (2011) Novel reaction control techniques for redundant space manipulators: Theory and simulated microgravity tests. Acta Astronaut 68(11–12):1712–1721
12. Ding H, Tso SK (1999) A fully neural-network-based planning scheme for torque minimization of redundant manipulators. IEEE Trans Ind Electron 46(1):199–206
13. Xia Y, Wang J (2001) A dual neural network for kinematic control of redundant robot manipulators. IEEE Trans Syst Man Cybern B Cybern 31(1):147–154
14. Zhang Y, Wang J, Xia Y (2003) A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits. IEEE Trans Neural Netw 14(3):658–667
15. Xia YS, Feng G, Wang J (2005) A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control. IEEE Trans Syst Man Cybern B Cybern 35(1):54–64

16. Tang WS, Wang J (2001) A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity. IEEE Trans Syst Man Cybern B Cybern 31(1):98–105
17. Cha SH, Lasky TA, Velinsky SA (2006) Kinematic redundancy resolution for serial-parallel manipulators via local optimization including joint constraints. Mech Based Des Struct Mach 34(2):213–239
18. Patchaikani P, Behera L, Prasad G (2011) A single network adaptive critic based redundancy resolution scheme for robot manipulators. IEEE Trans Ind Electron 59(8):3241–3253
19. Taghirad HD, Bedoustani YB (2011) An analytic-iterative redundancy resolution scheme for cable-driven redundant parallel manipulators. IEEE Trans Robot 27(6):1137–1143
20. 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
21. 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
22. Wu W (2003) Neuro-fuzzy and model-based motion control for mobile manipulator among dynamic obstacles. Sci China Ser F Inf Sci 46(1):14–30
23. 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
24. Liu B, He H, Chen S (2010) Adaptive dual network design for a class of simo systems with nonlinear time-variant uncertainties. Acta Autom Sin 36:564–572
25. 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
26. Xia Y, Wang J (2007) A dual neural network for kinematic control of redundant robot manipulators. IEEE Trans Syst Man Cybern B Cybern 31(1):147–154
27. Li S, Meng MQH, Chen W (2007) SP-NN: A novel neural network approach for path planning. In: Proceedings of IEEE International Conference on Robotics and Biomimetics, pp 1355–1360
28. Liu B, He H, Repperger DW (2010) Two-time-scale online actor-critic paradigm driven by POMDP. In: Proceedings of International Conference on Networking, Sensing and Control, pp 243–248
29. Liu S, Wang J (2006) A simplified dual neural network for quadratic programming with its KWTA application. IEEE Trans Neural Netw 17(6):1500–1510
30. Liu Q, Wang J (2011) A one-layer dual recurrent neural network with a heaviside step activation function for linear programming with its linear assignment application. In: Proceedings of the 21st International Conference on Artificial Neural Networks, pp 253–260
31. Zhang Y, Ge SS, Lee TH (2004) A unified quadratic programming based dynamical system approach to joint torque optimization of physically constrained redundant manipulators. IEEE Trans Syst Man Cybern B Cybern 34(5):2126–2132
32. Jin L, Zhang Y, Li S, Zhang Y (2016) Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. IEEE Trans Ind Electron 63(11):6978–6988
33. Jin L, Zhang Y (2014) G2-Type SRMPC scheme for synchronous manipulation of two redundant robot arms. IEEE Trans Cybern 45(2):153–164
34. Jin L, Zhang Y (2014) Discrete-time Zhang neural network for online time-varying nonlinear optimization with application to manipulator motion generation. IEEE Trans Neural Netw Learn Syst 26(7):1525–1531
35. Jin L, Li S (2017) Nonconvex function activated zeroing neural network models for dynamic quadratic programming subject to equality and inequality constraints. https://doi.org/10.1016/j.neucom.2017.05.017
36. Jin L, Li S, La HM, Luo X (2017) Manipulability optimization of redundant manipulators using dynamic neural networks. 64(6):4710–4720
37. Zhang Y, Wang J (2004) Obstacle avoidance for kinematically redundant manipulators using a dual neural network. IEEE Trans Syst Man Cybern B Cybern 34(1):752–759

38. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
39. Zhang Y, Cai B, Zhang L, Li K (2008) Bi-criteria velocity minimization of robot manipulators using a linear variational inequalities-based primal-dual neural network and PUMA560 example. Adv Robot 22:1479–1496
40. Zhang Y, Wang J, Xu Y (2002) A dual neural network for bi-criteria kinematic control of redundant manipulators. IEEE Trans Robot Autom 18(6):923–931
41. Khalil H (2002) Nonlinear Syst. Prentice-Hall, Englewood Cliffs
42. Morimoto AK (1984) Dynamic modeling of a Puma 560 for use in compliant path control. University of New Mexico
43. Li S, Zhou M, Luo X, You Z (2017) Distributed winner-take-all in dynamic networks. IEEE Trans Autom Control 62(2):577–589
44. Li S, Zhang Y, Jin L (2016) Kinematic control of redundant manipulators using neural networks. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2016.2574363
45. Jin L, Zhang Y, Li S, Zhang Y (2017) Noise-tolerant ZNN models for solving time-varying zero-finding problems: a control-theoretic approach. IEEE Trans Autom Control 62(2):992–997
46. Jin L, Zhang Y, Li S (2016) Integration-enhanced Zhang neural network for real-time varying matrix inversion in the presence of various kinds of noises. IEEE Trans Neural Netw Learn Syst 27(12):2615–2627
47. Mao M, Li J, Jin L, Li S, Zhang Y (2016) Enhanced discrete-time Zhang neural network for time-variant matrix inversion in the presence of bias noises. Neurocomputing 207:220–230
48. Khan M, Lin S, Wang Q, Shao Z (2016) Distributed multi-robot formation and tracking control in cluttered environment. ACM Trans Auton Adapt Syst 11(2):12
49. Wang H, Liu X, Liu P, Li S (2016) Robust adaptive fuzzy fault-tolerant control for a class of non-lower-triangular nonlinear systems with actuator failures. Inf Sci 336:60–74
50. Khan M, Li S, Wang Q, Shao Z (2016) CPS oriented control design for networked surveillance robots with multiple physical constraints. IEEE Trans Comput Aided Des Integr Circuits Syst 35(5):778–791
51. Khan M, Li S, Wang Q, Shao Z (2016) Formation control and tracking for co-operative robots with non-holonomic constraints. J Intell Robot Syst 82(1):163–174
52. Wang H, Yang H, Liu X, Liu L, Li S (2016) Direct adaptive neural control of nonlinear strict-feedback systems with un-modeled dynamics using small-gain approach. Int J Adapt Control Signal Process 30(6):906–927
53. Li S, You Z, Guo H, Luo X, Zhao Z (2015) Inverse-free extreme learning machine with optimal information updating. IEEE Trans Cybern 46(5):1229–1241
54. Wang Z, Liu X, Liu K, Li S, Wang H (2016) Backstepping-based lyapunov function construction using approximate dynamic programming and sum of square techniques. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2016.2574747
55. Muhammad A, Li S (2016) Dynamic neural networks for kinematic redundancy resolution of parallel stewart platforms. IEEE Trans Cybern 46(7):1538–1550
56. Li S, Kong R, Guo Y (2014) Cooperative distributed source seeking by multiple robots: algorithms and experiments. IEEE/ASME Trans Mechatron 19(6):1810–1820
57. Li S, Guo Y (2014) Distributed consensus filtering on directed switching graphs. Int J Robust Nonlinear Control 25:2019–2040
58. Li S, Lou Y, Liu B (2014) Bluetooth aided mobile phone localization: a nonlinear neural circuit approach. ACM Trans Embed Comput Syst 13(4):78
59. Li S, Li Y (2013) Nonlinearly activated neural network for solving time-varying complex Sylvester equation. IEEE Trans Cybern 44(8):1397–1407
60. Li S, Liu B, Li Y (2013) Selective positive-negative feedback produces the winner-take-all competition in recurrent neural networks. IEEE Trans Neural Netw Learn Syst 24(2):301–309
61. Li S, Li Y, Wang Z (2013) A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application. Neural Netw 39:27–39
62. Li S, Guo Y (2013) Dynamic consensus estimation of weighted average on directed graphs. Int J Syst Sci 46(10):1839–1853

63. Li S, Qin F (2013) A dynamic neural network approach for solving nonlinear inequalities defined on a graph and its application to distributed, routing-free, range-free localization of WSNs. Neurocomputing 117:72–80
64. Li S, Guo Y, Fang J, Li H (2013) Average consensus with weighting matrix design for quantized communication on directed switching graphs. Int J Adapt Control Signal Process 27:519–540
65. Li S, Li Y, Liu B (2012) Model-free control of Lorenz chaos using an approximate optimal control strategy. Commun Nonlinear Sci Numer Simul 17:4891–4900
66. Li S, Wang Y, Yu J, Liu B (2013) A nonlinear model to generate the winner-take-all competition. Commun Nonlinear Sci Numer Simul 18:435–442
67. Li S, Wang Z, Li Y (2013) Using Laplacian eigenmap as heuristic information to solve nonlinear constraints defined on a graph and its application in distributed range-free localization of wireless sensor networks. Neural Process Lett 37:411–424
68. Li S, Chen S, Liu B (2013) Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function. Neural Process Lett 37:189–205
69. Li S, Yu J, Pan M, Chen S (2012) Winner-take-all based on discrete-time dynamic feedback. Appl Math Comput 219:1569–1575
70. Li Y, Li S (2013) A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments. Neurocomputing 104:170–179
71. Chen S, Li S, Liu B, Lou Y, Liang Y (2012) Self-learning variable structure control for a class of sensor-actuator systems. Sensors 12:6117–6128

http://www.springer.com/978-981-10-7036-5

Neural Networks for Cooperative Control of Multiple Robot Arms

Li, S.; Zhang, Y.

2018, XV, 74 p. 26 illus., 22 illus. in color., Softcover
ISBN: 978-981-10-7036-5