Decentralized Coordinated Motion Control of Two Hydraulic Actuators Handling a Common Object

Mark Karpenko

Nariman Sepehri[1]
e-mail: nariman@cc.umanitoba.ca

Department of Mechanical and Manufacturing Engineering,
University of Manitoba,
Winnipeg, Manitoba, R3T 5V6, Canada

John Anderson
Department of Computer Science,
University of Manitoba,
Winnipeg, Manitoba, R3T 5V6, Canada

In this paper, reinforcement learning is applied to coordinate, in a decentralized fashion, the motions of a pair of hydraulic actuators whose task is to firmly hold and move an object along a specified trajectory under conventional position control. The learning goal is to reduce the interaction forces acting on the object that arise due to inevitable positioning errors resulting from the imperfect closed-loop actuator dynamics. Each actuator is therefore outfitted with a reinforcement learning neural network that modifies a centrally planned formation constrained position trajectory in response to the locally measured interaction force. It is shown that the actuators, which form a multiagent learning system, can learn decentralized control strategies that reduce the object interaction forces and thus greatly improve their coordination on the manipulation task. However, the problem of credit assignment, a common difficulty in multiagent learning systems, prevents the actuators from learning control strategies where each actuator contributes equally to reducing the interaction force. This problem is resolved in this paper via the periodic communication of limited local state information between the reinforcement learning actuators. Using both simulations and experiments, this paper examines some of the issues pertaining to learning in dynamic multiagent environments and establishes reinforcement learning as a potential technique for coordinating several nonlinear hydraulic manipulators performing a common task. [DOI: 10.1115/1.2764516]

1 Introduction

One aspect of robotics research that continues to receive much attention in the literature is the manipulation of objects using multirobot coordinated frameworks. An obvious benefit of such an approach is the ability to manipulate large or awkward objects that would be difficult for a single robot to handle. Another important advantage of using multimanipulator systems is the possibility of regulating the internal force acting on the object [1]. However, when two or more manipulators are used to move an object, a closed kinematic chain is formed and the motion of one manipulator is translated through the object to affect the motion of the other manipulators [2]. Consequently, it is a challenge to develop control systems for effective coordinated manipulation, although much progress has been made in this area.

The development of suitable techniques for the coordinated control of multiple robots interacting with a common object has been tackled in a number of different ways. Yet, many of the developed techniques seem to fall into one of two basic categories depending upon the feedback signals available for control. The first is the master/slave approach [3,4] where each robot is controlled independently using either position or force feedback. In order to manipulate a common object, the master follows a reference trajectory under position control while the force-controlled slave attempts to regulate the interaction force in response to the motion of the master. One of the main challenges that must be overcome in applying this approach is the design of a slave system that can react quickly to changes in the master position [4].

The second approach to coordinated control of multiple robots is known as hybrid position/force control, which was originally developed for single robot systems [5]. Using this paradigm, both position and force errors are made available to each manipulator and the control design problem is to derive a suitable algorithm that gives reasonable position tracking accuracy and at the same time regulates the interaction forces applied to the object. A number of solutions based on rigid body dynamics of closed-loop kinematic chains appear in the literature. To name a few, Yoshikawa and Zheng [6] derived a single nonlinear state feedback position/force control law that takes both the manipulator and object dynamics into consideration but requires access to position and force information from all manipulators. As such, the controller is an example of a global or centralized coordinated control strategy. Perdereau and Drouin [7] developed model-based local controllers for a two-robot system. In their work, the desired force and position vectors were computed by a central supervisor but no sensory information was exchanged between the two robots. Using this decentralized approach, each actuator is self-controlled [7]. Liu and Arimoto [8] also proposed a decentralized control scheme for a coordinating two robots handling a common object and noted that adopting such a localized control scheme greatly simplifies controller implementation.

In practice, the use of the above cited control methodologies requires knowledge of the exact values of the dynamic model parameters as well as those of the object under manipulation. Consequently, more recent investigations (see Ref. [9] and the references cited therein) have focused on the development of robust and adaptive coordination controls. The problem of system uncertainty has also been addressed using neural networks [10] and iterative learning techniques [11]. Other coordination schemes based on impedance control concepts have also been developed [12] but are often criticized for the difficulty associated with choosing an impedance model that guarantees the desired trajectory tracking performance [8].

One fundamental assumption underlying all of the above work

[1]Corresponding author.

Contributed by the Dynamic Systems, Measurement, and Control Division of ASME for publication in the Journal of Dynamic Systems, Measurement, and Control. Manuscript received June 23, 2006; final manuscript received January 25, 2007. Review conducted by Prabhakar R. Pagilla. Paper presented at the 2006 American Control Conference, Minneapolis, MN.

is that the manipulators are equipped with electric motors that exert controllable torques. However, in many industrial applications, such as manufacturing, forestry, mining, and construction, where large power to weight ratios are required hydraulic actuators are often selected as prime movers. Unlike electric actuators, torque control of hydraulic actuators is not a straightforward problem owing to the nonlinear nature of the hydraulic functions, fluid compressibility, and actuator seal friction [13]. As a result, the literature is not well established in this area and remains limited to only a few examples [13–15]. Mulder and Malladi [14] developed a minimum effort control algorithm for a hydraulically powered cooperative robotic arm and proposed the use of adaptive control element to enable compliant motion of the end effector. Asokan et al. [15] designed an electrohydraulic impedance controller for robotic interaction control. The control system design was, however, carried out and tested using only a single manipulator and the generalization of the results to multimanipulator frameworks was not verified. The development presented in Refs. [13,15] employed a centralized control approach and was validated via simulations in a multimanipulator environment. Limited experimentation was also done in [15] on two single hydraulic actuators. To further contribute to the development of suitable control techniques for coordinating the motion of hydraulic manipulators, this paper investigates the application of reinforcement learning to synchronize a pair of horizontal hydraulic actuators whose task is to rigidly hold and move an object along a specified position trajectory. In contrast to the existing work on coordinated control of hydraulic manipulators, this paper studies the application of an artificial intelligence based decentralized control scheme to a hydraulically actuated system for the first time.

Reinforcement learning encompasses a class of machine learning techniques, whose algorithms use a trial-and-error search to learn a map between situations and actions that maximizes a numerical reward signal [16,17]. The learning agent interacts with its environment by first measuring the environmental state using sensors and then responding to the current stimuli by selecting an appropriate action, which subsequently alters the environment. The reward received by the learning agent depends upon the relevance of the selected actions toward achieving the learning goal. By associating large numerical values of the reward with actions leading to desirable outcomes, the tendency of the learning agent to select the same action when a similar environmental state is sensed in the future is strengthened. On the other hand, when the numerical reward is small, the exploration of alternative actions is encouraged. The main difficulty in reinforcement learning is to design algorithms that guide the evolution of the action-selection policy toward the optimal one via random interactions with the environment. In situations where the reinforcement is delayed, reinforcement learning is often implemented by constructing an internal model of the state-action map that can be used to estimate the long term utilities of various actions selected in sequence. By adjusting the utility estimates, an agent can learn to select actions that tend to maximize the estimated long term reward. Temporal differencing [18], dynamic programming [19], and Q-learning [20] are all examples of such model-based reinforcement learning strategies. In environments where an evaluation signal is always available immediately following an action, it is possible to employ algorithms, such as statistical hill climbing techniques [21], that maximize the immediate reward.

Although some difficult control problems have been solved using reinforcement learning, e.g., Refs. [22–25], many reinforcement learning algorithms assume that only a single learning agent operates on the environment. In situations where multiple learning agents operate autonomously in a shared environment, the environment can be affected by the actions of more than one agent at any time. Hence, it is often difficult to properly assign credit or blame to the behaviors of individual agents, which affect the environment simultaneously. The resulting credit assignment problem is a fundamental issue pertaining to the implementation of reinforcement learning techniques in multiagent learning systems [26]. Consequently, the application of reinforcement learning techniques to decentralized control problems in the presence of multiple learning agents, such as the control problem studied in this paper, continues to receive much attention in the literature.

The primary objective of this paper is to examine how reinforcement learning can improve the coordination of two hydraulic actuators manipulating an object by enabling the discovery of a decentralized control strategy that allows the interaction forces, i.e., the forces between the actuators and the object, to be reduced while the actuators track centrally planned local reference trajectories as closely as possible. The interaction forces arise due to positioning errors caused by the imperfect closed-loop actuator dynamics. By using a specially designed reinforcement learning neural network, which monitors the interaction force, each actuator can learn how to appropriately modulate the measured force by adjusting the local reference trajectory. In effect, the interaction force is reduced not by improving tracking performance but by learning how to eliminate the difference between the tracking errors of each actuator. The interaction force acting on the object can therefore be alleviated without the need to explicitly consider the properties of the object.

In this paper, the stochastic-real-valued (SRV) reinforcement learning algorithm of Guallpalli [27] is applied to solve the multiagent learning problem by maximizing the immediate reward. Implementing a reinforcement learning algorithm that maximizes the immediate reward is possible since an evaluation signal based on the measured value of the interaction force is easy to compute at regular intervals. Consequently, the resulting learning system has the potential for quick learning and adaptation. The SRV algorithm allows continuous real-valued control outputs to be associated with environmental states via the adaptation of the weights of two connectionist neural networks and can be easily extended to multilayer neural network frameworks [24]. While the SRV algorithm has been proven on several single-agent learning tasks [24,27], to the best of the authors' knowledge, its performance has never been tested in a multiagent learning environment. Thus, another objective of this paper is to investigate whether learning can be successful in the absence of any communication between the hydraulic actuators. It is also of interest to examine how simple directed communication [28] can be used to influence the learned behaviors of the hydraulic actuators to further improve the performance and cooperation of the multiagent reinforcement learning system. These questions are investigated in this paper in both simulations and experiments. Simulations are used to determine the ideal configuration of the proposed reinforcement learning neural network controller and to establish the general behavior of the inherently stochastic reinforcement learning system. Experiments are performed to confirm the performance of the reinforcement learning system in practice.

The remainder of this paper is organized as follows. In Sec. 2, the system under investigation is briefly described and a mathematical model of a typical valve controlled hydraulic actuator is derived to facilitate simulation studies. The proposed reinforcement learning control architecture is presented in Sec. 3. Section 4 gives details on the implementation of the reinforcement learning architecture and summarizes the results of several simulation studies that were carried out to ascertain the general behavior of the multiagent learning system. The performance of the multiactuator positioning system is evaluated on a real experimental test rig in Sec. 5 both in the absence and presence of interactuator communication. Concluding remarks follow in Sec. 6.

2 Description of the System Under Investigation

A schematic of a typical servovalve controlled hydraulic actuator for the purpose of mathematical modeling is shown in Fig. 1, along with the relevant nomenclature. By closing the position loop using a simple proportional controller, state equations that de-

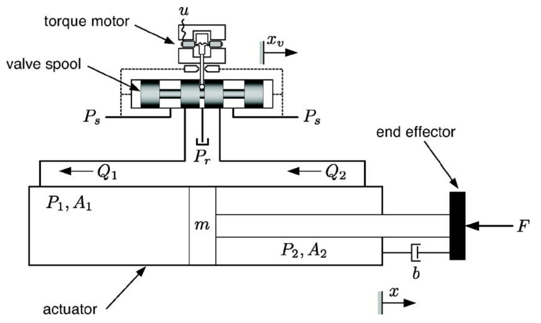


Fig. 1 Schematic of a typical hydraulic actuator for mathematical modeling

scribe the closed-loop hydraulic actuator dynamics between the desired x^d and actual x positions of the actuator can be formed as (readers are referred to Refs. [29,30] for details)

$$\dot{x} = v \tag{1a}$$

$$\dot{v} = \frac{1}{m}(-bv + A_1P_1 - A_2P_2 - F) \tag{1b}$$

$$\dot{P}_1 = \frac{\beta}{A_1x + \bar{V}}(Q_1 - A_1\dot{x}) \tag{1c}$$

$$\dot{P}_2 = \frac{\beta}{A_2(L-x) + \bar{V}}(-Q_2 + A_2\dot{x}) \tag{1d}$$

$$\dot{x}_v = v_v \tag{1e}$$

$$\dot{v}_v = -\omega_v^2 x_v - 2\zeta_v\omega_v v_v + k_v\omega_v^2 u \tag{1f}$$

$$u = K_p(x^d - x) \tag{1g}$$

where

$$Q_1 = C_v w x_v \sqrt{\frac{2(P_s - P_1)}{\rho}}, \quad Q_2 = C_v w x_v \sqrt{\frac{2(P_2 - P_r)}{\rho}} \quad \text{for } x_v \geqslant 0 \tag{2}$$

and

$$Q_1 = C_v w x_v \sqrt{\frac{2(P_1 - P_r)}{\rho}}, \quad Q_2 = C_v w x_v \sqrt{\frac{2(P_s - P_2)}{\rho}} \quad \text{for } x_v < 0 \tag{3}$$

Referring to Eqs. (1*a*)–(1*f*), the system states are actuator position x, actuator velocity v, chamber pressures P_1 and P_2, valve spool displacement x_v, and valve spool velocity v_v. Parameters m and b are the combined mass of the actuator piston and rod, and the equivalent viscous actuator damping, respectively. Area A_1 refers to the area of the piston and area A_2 is the annulus area of the piston on the rod side of the actuator. The effective bulk modulus of the hydraulic fluid is denoted by β, while L denotes the actuator stroke and parameter $\bar{V}$ represents the volume of the connecting lines between the servovalve and the actuator. Load force F is a disturbance input and arises when the actuator interacts with the environment or another actuator.

The valve spool dynamics are expressed as a second-order lag (see Eqs. (1*e*) and (1*f*)) where k_v is the valve spool position gain and parameters ω_v and ζ_v are the servovalve undamped natural frequency and damping ratio, respectively. The gain of the closed-loop proportional controller is denoted by K_p in Eq. (1*g*). In Eqs. (2) and (3), which describe the servovalve control flows, parameter ρ is the mass density of the hydraulic fluid, C_v is the servovalve coefficient of discharge, and w is the slot width of the port through which the fluid flows. P_s and P_r denote the hydraulic supply and return pressures, respectively.

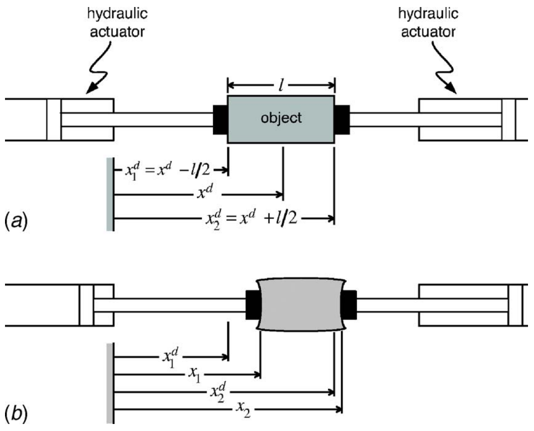


Fig. 2 Schematic of coordinated positioning task: (*a*) definition of reference trajectories x_1^d and x_2^d with respect to desired object position x^d; (*b*) deformation of object due to relative actuator positioning error

Figure 2 shows two hydraulic actuators that are coupled together in order to move an object along a one-dimensional position trajectory x^d, by having each actuator follow the specified local reference trajectory x_1^d and x_2^d. With reference to Fig. 2, (i) the positions of both actuators are considered absolute and positive right, which eliminates the need to consider the local reference frames of each actuator separately. (ii) The object is held rigidly and there is no relative motion between the object and the end effectors. (iii) The reference trajectories for each actuator are functions of the desired object motion trajectory and are centrally planned.

Under ideal circumstances, the two hydraulic actuators will follow their desired trajectories without causing any internal force in the object. However, in practice, positioning errors resulting from the imperfect control implementation are inevitable and cause the actuators to apply an undesirable force to the object. The force is a result of the relative position trajectory error between the two actuators and leads to the deformation of the object (see Fig. 2(*b*)). To improve their performance, each hydraulic actuator is outfitted with a reinforcement learning neural network, described in the next section, that can modify the prescribed position trajectory dynamically in response to the measured force between the object and the actuator. The goal is for the hydraulic actuators to learn, by reinforcement, how to modify their reference trajectories in a decentralized fashion to reduce the interaction force.

3 Proposed Control Architecture

A schematic of the proposed coordinated position control system is shown in Fig. 3. Referring to Fig. 3(*a*), each actuator is self-controlled and uses a local position control law as well as a reinforcement learning neural network (RLNN) to manipulate the object by following a reference trajectory determined by a central trajectory planner. The reinforcement learning control architecture that is implemented on each hydraulic actuator is shown in Fig. 3(*b*). With reference to Fig. 3(*b*), each actuator is positioned using a simple closed-loop proportional control law. The input signal to the control loop $\bar{x}^d$ is the summation of the *a priori* prescribed reference trajectory x^d and a trajectory correction Δx determined by the RLNN. The difference between the prescribed reference trajectory and the actuator position $x^d - x$ is referred to, in this paper, as the position trajectory error while the error seen by the controller that drives the actuator $x^d + \Delta x - x$ is termed the actuator position error. The RLNN consists of two independent multilayer

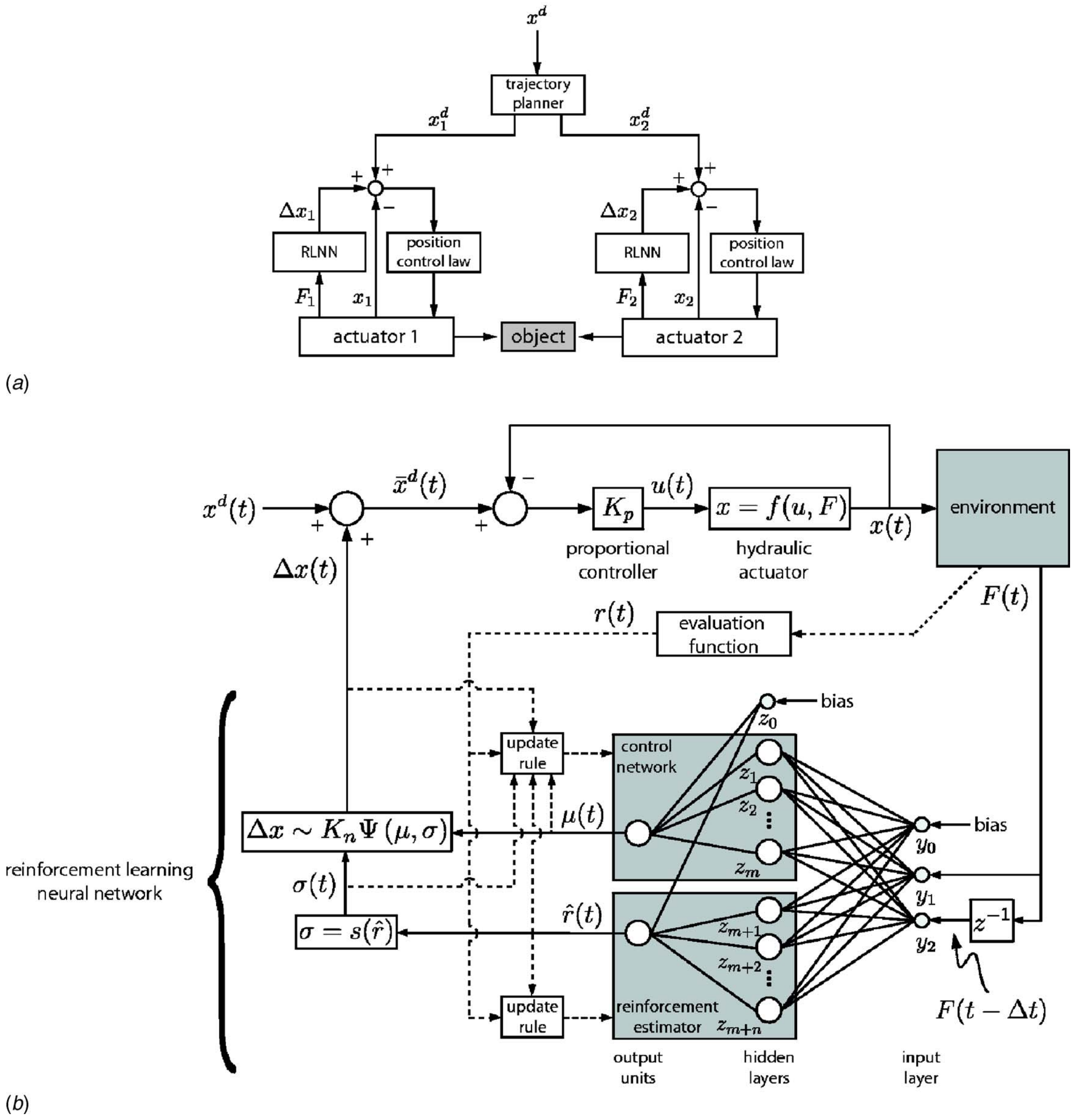


Fig. 3 Schematic of coordinated positioning system: (*a*) decentralized control scheme using localized position and RLNN controllers; (*b*) architecture of local reinforcement learning control system showing signals used for learning as dashed lines

feedforward networks together with a Gaussian random number generator. This neural network architecture represents a SRV reinforcement learning system [24]. Trajectory correction Δx, which is used to modulate the interaction force during the manipulation task, is a normally distributed random variable $\Delta x \sim K_n \Psi(\mu,\sigma)$ whose mean value is controlled by network output μ and scaled by factor K_n. Gain $K_n = 1.0 \times 10^{-3}$ m/mm was used so that the network inputs and outputs have similar orders of magnitude. Function $\sigma = s(\hat{r})$, where $\hat{r} \in [0,1]$ is the estimated reward, controls the trial-and-error search behavior of the reinforcement learning actuator by adjusting the variance of the Gaussian distribution according to

$$\sigma = \lambda(1 - \hat{r}) \tag{4}$$

where λ is a scaling factor. Physically, Eq. (4) limits the variance of the random output to an envelope within $\sim 2\lambda$ units of network output μ, 95% of the time. During learning, the adjustable network weights are adapted so as to maximize the value of the reinforcement signal r received from the environment. If the expected reward $\hat{r}$ can also be accurately estimated, Eq. (4) ensures that $\sigma \to 0$, which makes $\Delta x \approx \mu$. This property enables the actuator to exploit its previously learned behavior.

Referring to Fig. 3(*b*), it is observed that the single input to the RLNN is the measured force F between the implement and the object. Temporal information carried by the force transducer output is implicitly captured by a short-term memory element, which is a first-order tapped delay line represented by unit delay operator z^{-1}. Therefore, each neural network has two inputs, the current value of the force transducer output $F(t)$ and the stored value of the force measurement $F(t-\Delta t)$ from the previous time step. Including the short-term memory element allows the neural network to behave as a one-step ahead predictor [31] and generate an input-output map that considers the dynamics of the input signal. Considering this and the fact that the neural networks have bias terms, the control action tends to resemble a proportional-integral-derivative type control law. It should be noted that additional delay taps could be used to provide supplementary higher-order inputs to the neural network. This way, more complex models relating the measured force to the actuator position can potentially be learned. However, as the number of network inputs increases so does the number of internal parameters that must be tuned during learning. Consequently, increasing the number of network inputs does not guarantee improved performance. In this paper, the neural network structure was kept as simple as possible and, as will be seen in the sequel, the selected configuration of the neural network inputs is adequate to improve actuator performance on the coordinated positioning task. Also in an effort to avoid overly complex neural network structures, each multilayer feedforward network was restricted to have a variable number of hidden neurons arranged in a single layer, as shown in Fig. 3(*b*).

The hidden layers of each neural network generate a nonlinear representation of the network inputs, which is then further processed by the output units. The hidden layer neurons all use the logistic activation function, $\phi(v)=1/(1-e^{-v})$. The output of the first neural network, referred to hereafter as the *control network*, is computed as the linear weighted sum of the outputs of the hidden layer neurons. Hence, the output of the control network is given by the relation

$$\mu=\sum_{j=1}^{m} w_j^{(1)}\phi\left[\sum_{i=0}^{2} w_{ij}^{(0)}y_i\right]+w_0^{(1)}z_0 \tag{5}$$

where y_i refers to input i. The adjustable weight between input i and hidden neuron j is denoted by $w_{ij}^{(0)}$, while $w_j^{(1)}$ is the adjustable weight between hidden neuron j and the output unit. Index m denotes the number of neurons in the hidden layer, and the bias terms are absorbed into Eq. (5) by setting $y_0=1$ and $z_0=1$.

In contrast to the control network, the output unit of the second neural network, referred to hereafter as the *reinforcement estimator*, uses a logistic activation function similar to the hidden layer neurons. This output function was selected because the actual value of the reinforcement signal r generated by the performance evaluation function varies continuously between 0 and 1. Thus, the reinforcement estimate $\hat{r}$ is computed as

$$\hat{r}=\phi\left\{\sum_{j=1}^{n} v_j^{(1)}\phi\left[\sum_{i=0}^{2} v_{ij}^{(0)}y_i\right]+v_0^{(1)}z_0\right\} \tag{6}$$

where $v_{ij}^{(0)}$ and $v_j^{(1)}$ denote the adjustable weights between input i and hidden neuron j, and between hidden neuron j and the output unit, respectively. As before, $y_0=1$ and $z_0=1$ to accommodate the bias terms. Index n denotes the number of neurons in the hidden layer of the reinforcement estimator.

The weight matrices **w** and **v** of each neural network are tuned based on the actual value of the reinforcement signal, which is received continuously from the environment. In this paper, the hidden layer weights of the control network are updated at each time step Δt using a modified version of the update rule proposed by Gullapalli [27]

$$w_j^{(1)}(t+\Delta t)=w_j^{(1)}(t)+0.5\sigma(t)\mathcal{S}[r(t)-\hat{r}(t)][\Delta x(t)-\mu(t)]z_j(t) \tag{7}$$

where the sign function $\mathcal{S}\{r-\hat{r}\}$ that replaces difference $(r-\hat{r})$ in the original algorithm is given by

$$\mathcal{S}\{r-\hat{r}\}=\begin{cases}-1 & r-\hat{r}<0\\ 0 & r-\hat{r}=0\\ 1 & r-\hat{r}>0\end{cases} \tag{8}$$

Referring to Eq. (7), at the end of each learning opportunity, each weight of the control network is incremented by an amount proportional to

$$\Delta w_j^{(1)}=\alpha\mathcal{S}\{r-\hat{r}\}e_j \tag{9}$$

In Eq. (9), $\alpha=0.5\sigma^2$ is assumed to be a constant learning rate parameter, term $\mathcal{S}\{r-\hat{r}\}$ is a reinforcement comparison, and $e_j=(\Delta x-\mu/\sigma)z_j$ is the characteristic eligibility of weight $w_j^{(1)}$ [18]. Together, the reinforcement comparison and the characteristic eligibility determine the direction of the error gradient between the current output Δx and the optimal output for which the maximum reinforcement would be received [24]. To illustrate how the direction of the error gradient is established, Gullapalli [27] finds it useful to view Eq. (9) as the normalized noise that is added to the activation of the control network. If the noise has results in a reinforcement signal r larger than expected $\hat{r}$, then the mean output of the control network μ is adjusted by adapting the network weights to become closer to the current output Δx. If, on the other hand, the noise results in a reinforcement signal smaller than expected, the mean output should be adjusted to be further from the current activation. In this way, the RLNN tends to statistically climb the error gradient [21] so that the reinforcement can be maximized via the association of input states with the optimal control outputs.

Equation (7) is only used to update the hidden layer weights of the control network. The remaining input layer weights of the control network are adapted, in this paper, using standard error backpropagation [31] with a learning rate of 0.5. Also, since reinforcement comparison $(r-\hat{r})$ is known at each time step, the weights of the reinforcement estimator can be easily adapted using simple gradient descent. A learning rate of 0.5 was used.

Table 1 Parameters of hydraulic actuators used in simulations

Parameter	Symbol	Value
Supply pressure	P_s	6.9 MPa (1000 psi)
Return pressure	P_r	0
Total mass of piston and rod	m	10 kg
Viscous damping coefficient	b	350 N s/m
Actuator stroke	L	0.61 m
Piston area	A_1	1.14×10^{-3} m^2
Piston annulus area	A_2	6.33×10^{-4} m^2
Volume of connecting lines	$\bar{V}$	4.15×10^{-5} m^3
Valve coefficient of discharge	C_v	0.6
Valve orifice area gradient	w	0.020 75 m^2/m
Valve spool position gain	k_v	4.06×10^{-5} m/V
Valve natural frequency	ω_v	150 Hz
Valve damping ratio	ζ_v	0.5
Density of hydraulic fluid	ρ	847 kg/m^3
Bulk modulus of hydraulic fluid	β	689 MPa
Controller proportional gain	K_p	250 V/m

4 Simulation Studies

4.1 Implementation. In order to assess the ability of the proposed reinforcement learning control architecture to reduce the magnitudes of the interaction forces applied to the object by the hydraulic actuators during the manipulation task, a set of simulation studies was carried out. The efficacy of learning in the absence of communication between the learning agents was evaluated first in order to establish performance benchmarks to which simulation results pertaining to learning in the presence of directed communication could be later compared.

In the simulations, the object was modeled as a pure stiffness, which resists the motion of the actuators in both tension and compression. With reference to Fig. 2, the net force F applied to the object by the hydraulic actuators is given by the following equation:

$$F=k(x_2-x_1-l) \tag{10}$$

where stiffness k was taken to be 3.0 kN/m and the length l of the object was assumed to be 120 mm. The values of the parameters for the hydraulic actuators used in the simulations are given in Table 1 and it was assumed that each actuator has the same values of the model parameters. However, due to the asymmetry in the

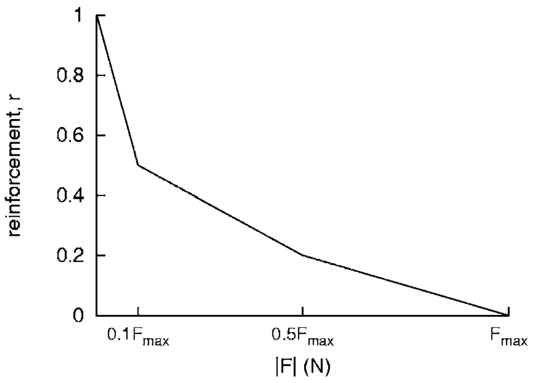


Fig. 4 Reinforcement function for evaluating actuator performance

actuator piston areas, the actuators travel at different speeds depending on the direction of motion. This discrepancy causes differences in the position trajectory errors of each actuator and results in the deformation of the object. In practice, slight differences in the closed-loop position controllers and/or the parameters of each actuator would further influence these errors.

It was expected that the performance of the reinforcement learning actuators, with respect to reducing the interaction force, should continue to improve as the number of repetitions of the manipulation task increases. Thus, to facilitate training, the desired smooth path between the initial and final positions of the manipulated object was approximated by a continuous function so that learning could proceed uninterrupted. In all of the simulations reported here, the following formation constrained reference trajectories for each actuator (see Fig. 2) were used:

$$x_1^d = 200 \sin(0.5\pi t) + 240 \text{ mm} \tag{11}$$

$$x_2^d = 200 \sin(0.5\pi t) + 360 \text{ mm} \tag{12}$$

The performance of each actuator was evaluated based on the absolute value of the measured force F between the actuator and the object, as shown in Fig. 4. As is seen, the evaluation function chosen is piecewise continuous and generates a reinforcement signal $r \in [0,1]$ based on the maximum allowable force $F_{\max}$. The slope of the curve becomes steeper as the origin is approached to increase the sensitivity of the reinforcement signal to small values of the measured force. This was found to improve the performance of the reinforcement learning system by enabling the RLNNs to continue to fine-tune their performance after the force applied to the object had been reduced. In all the simulations and experiments reported here, $F_{\max}=100$ N was assumed. Scaling factor $\lambda=1.0$ mm was selected in Eq. (4) to limit the envelope of stochastic exploration to approximately 2 mm. This value was arrived at by considering the stiffness of the object, and allows each RLNN to modulate the applied interaction forces by about 3 N while they search for the control actions that tend to reduce the values of the interaction forces overall. The numerical value of parameter λ should therefore be selected based upon the anticipated value of the object compliance and should be decreased as the object stiffness increases since smaller changes in actuator position lead to larger changes in the forces applied to the object. Note that setting λ too small may impair the rate of learning by overrestricting the search space.

In addition to measurements of the interaction force, another metric was also defined in order to assess the relative performance of each reinforcement learning hydraulic actuator. The level of cooperation between the actuators was evaluated by examining the ratios of the root-mean-square (rms) values of the trajectory corrections selected by each RLNN to the total corrective effort of the multiactuator team that was necessary to reduce the force. Therefore, the particular notion of cooperation adopted here refers to the ability of the actuators to discover how to equally share the burden of reducing the interaction forces during the coordinated positioning task. The trajectory correction ratio c_1 of actuator 1 is defined as

$$c_1 = \frac{\Delta x_{1,\text{rms}}}{\Delta x_{1,\text{rms}} + \Delta x_{2,\text{rms}}} \tag{13}$$

where $\Delta x_{1,\text{rms}}$ and $\Delta x_{2,\text{rms}}$ are the rms values of trajectory corrections Δx_1 and Δx_2. A similar equation can be written for actuator 2. Referring to Eq. (13), if $c_1 \approx 0.5$ then the level of cooperation achieved between the coordinated actuators is high since $\Delta x_{1,\text{rms}} \approx \Delta x_{2,\text{rms}}$. In other words, both actuators equally share the burden of reducing the interaction force. If, on the other hand, $c_1 \approx 0$ or $c_1 \approx 1.0$, the level of cooperation between the coordinated actuators is low. For example, in the former case, $c_1 \approx 0$ implies $\Delta x_{1,\text{rms}} \approx 0$ so actuator 1 corrects its trajectory very little compared to actuator 2. In the latter case, the neural controller of actuator 2 is relatively inactive and the entire burden of reducing the interaction force becomes the responsibility of actuator 1.

Using simulations, the ideal number of neurons in the hidden layers of the neural networks for each hydraulic actuator was first established by increasing the number of units in the hidden layers until performance improvements diminished. The adjustable network weights were initialized using random numbers on the interval $[-0.5, 0.5]$ and the dynamic equations were simulated to allow the neural networks to learn for 90 s (55 repetitions of the manipulation task). The fourth-order Runge–Kutta integration scheme with a 0.001 s time step was used and learning was assumed to occur at a constant rate of 100 Hz. One hundred 90 s trials were run for hidden layer sizes ranging from 2 to 24 neurons in both the control and reinforcement estimator networks. For each trial, the rms value of the interaction force acting on the object was computed as an indicator of the variance of the force over the entire 90 s learning period. It was observed that networks having ten hidden layer neurons reduced the rms value of the interaction force by at least 65% on average and offered the best performance with respect to the hidden layer size.

4.2 Coordinated Control With Noncommunicating Actuators. To illustrate how the performance of the hydraulic actuators can be improved by the proposed multiagent reinforcement learning system, the time histories of the interaction force as well as the trajectory corrections were monitored before, during, and after the learning process. The RLNN of each actuator had ideal number of hidden neurons in the control network and reinforcement estimator. All of the adjustable network weights were initialized with random numbers on the interval $[-0.5, 0.5]$. The actuators handled the object under proportional control only during the first 10 s of the simulation. This was done to illustrate the base line performance of the multiactuator system. Then, the neural networks were activated and learning was allowed for the next 30 s of simulated time. For the final 10 s of the simulation, no learning was allowed and the hydraulic actuators handled the object by exploiting their previously learned behaviors. Figure 5(*a*) shows that before learning (time interval (1)), when the actuators were operated under proportional control only, the peak value of the interaction force reached nearly 25 N in magnitude. Note that in Fig. 5(*a*), negative values of interaction force represent the compression of the object. After the neural networks were activated and learning began (time interval (2)), the interaction force was reduced almost immediately. Learning continued to improve the performance of the actuators as the time increased. After learning, during time interval (3), it is observed that the peak-to-peak amplitude of the interaction force had been reduced to approximately 3.0 N, an 80% improvement over the base line performance under proportional control only.

The trajectory modifications, shown in Fig. 5(*b*), illustrate the corrective actions applied by the neural networks. At the begin-

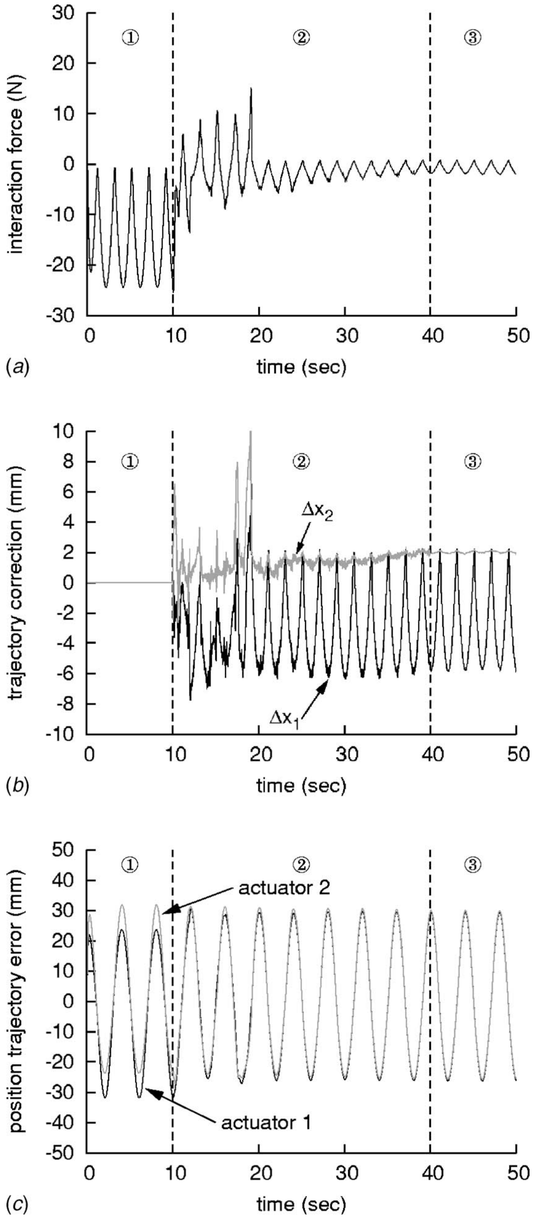


Fig. 5 Typical performance of noncommunicating hydraulic actuators on coordination task: (*a*) interaction force; (*b*) trajectory corrections, Δx_1 and Δx_2; (*c*) position trajectory errors, $x_1^d - x_1$ and $x_2^d - x_2$. Legend: (1) before learning; (2) learning; (3) after learning.

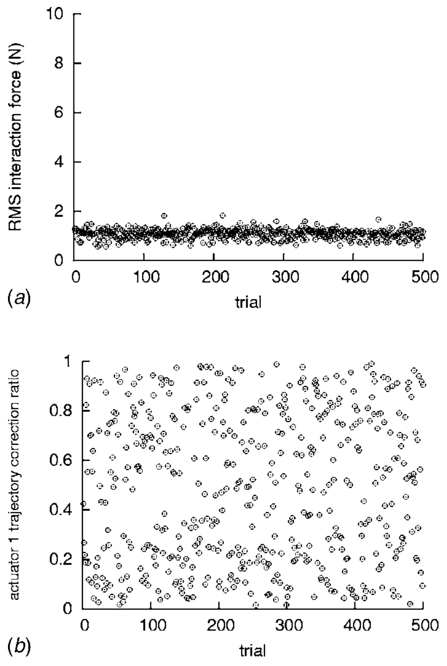


Fig. 6 Performance of noncommunicating hydraulic actuators on coordination task after learning: (*a*) rms interaction force; (*b*) actuator 1 trajectory correction ratio

ning of time interval (2), it is observed that the corrections are noisy and appear to be selected randomly. This is the period of time when the neural networks are exploring the environment most and when the neuronal weights are adapting most rapidly. As the time increases beyond 30 s, the noise associated with the explorative behavior is reduced significantly and the corrections selected by the neural networks become more consistent between repetitions of the manipulation task. The neuronal weights undergo only small refinements during this time as previously learned behaviors can be successfully exploited. A visual comparison of the magnitudes of the trajectory corrections confirms that the hydraulic actuators have not learned how to equally share the burden of reducing the interaction force since on average $|\Delta x_2| \ll |\Delta x_1|$. Figure 5(*c*) shows that the interaction force could be alleviated with negligible effect on the positional accuracy. Thus, the benefit of improved coordination amongst the hydraulic actuators can be obtained without the need to explicitly improve position tracking accuracy. Figure 5(*c*) also reveals that the hydraulic actuators have reduced the interaction force by, in effect, learning how to match their position trajectory errors.

To establish whether the results of Fig. 5 truly reflect the general behavior of the inherently stochastic reinforcement learning system, 500 40 s trials where learning was allowed to occur for the first 30 s of simulated time were carried out. At the beginning of each trial, the adjustable network weights were initialized with different random numbers on the interval $[-0.5, 0.5]$. In addition, the Gaussian random number generators of each neural network were initialized with a different randomly selected seed. For each trial, after the 30 s learning period, the system was run for an additional 10 s where the hydraulic actuators handled the object by exploiting their previously learned behaviors. Over this 10 s time period, the rms values of the interaction forces as well as the trajectory correction ratio of actuator 1 were computed to evaluate the performance after learning had converged. Figure 6 reports the rms interaction force and the corresponding trajectory correction ratio for actuator 1 obtained from the simulations. The data points represent the values of the performance metrics obtained for each trial and are shown in a sequential fashion for the purpose of comparison. Referring to Fig. 6, it is observed that after allowing the neural networks to learn for 30 s, the rms values of the interaction force are about 1.1 N, on average. This is a marked improvement over the base line value of 16.9 N (see Fig. 5(*a*)) that is obtained when the system is operated under proportional control only. These results confirm that, in general, the designed reinforcement learning architecture has the potential to improve the coordination of the hydraulic actuators and allow the interaction force (due to positioning errors) to be greatly reduced.

The results also imply that for a typical trial, the relative magnitudes of the trajectory corrections selected by each RLNN can differ significantly since the values of the actuator 1 trajectory correction ratio were observed to vary widely between 0 and 1 (see Fig. 6(*b*)). Thus, while reinforcement learning improves performance on the manipulation task with respect to reducing the interaction force, a cooperative effort where each actuator selects corrective actions of similar magnitude cannot be guaranteed. This outcome is a direct result of the multiagent credit assignment problem. Since the force sensor of each actuator measures nearly the same value of the interaction force, it is impossible for the RLNNs to infer the relative contribution of each actuator to the measured force. Consequently, when the force is reduced, each of the neural networks receives approximately the same value of reinforcement irrespective of their relative effort. The information contained in the force measurement alone is therefore insufficient to properly reward the corrective actions of each actuator during the learning process.

4.3 Coordinated Control With Communicating Actuators. The motivation behind the use of communication between the hydraulic actuators is to enable the actuators to learn how to effectively cooperate by selecting equal and symmetric neural network outputs as they work to reduce the interaction forces. Recall that an undesirable internal force is produced when position trajectory errors, $e_1=x_1^d-x_1$ and $e_2=x_2^d-x_2$, are different form one another. Consequently, it is necessary to precompensate the original reference trajectories so that $(e_2-e_1)\rightarrow 0$. The role of the RLNNs is, therefore, to alleviate difference (e_2-e_1) by providing this compensating action. In the presence of communication, the multiagent credit assignment problem can be resolved by the design of additional evaluation functions that reward the selection of complimentary actions, which tend to match the relative output levels of each reinforcement learning neural network.

In this paper, the abilities of each neural network to properly compensate the original reference trajectories are evaluated using two locally computed penalties p_1 and p_2. The values of these penalties are then deducted from the value of reinforcement r computed using Fig. 4. Hence, actuator 1 receives evaluation $r_1=r-p_1$, while actuator 2 receives $r_2=r-p_2$. The penalties are computed as follows:

$$p_1=\begin{cases}\gamma|(e_1-e_2)-(\Delta x_1+\Delta x_2)| & \text{if } \gamma|(e_1-e_2)-(\Delta x_1+\Delta x_2)|<0.5\\ 0.5 & \text{otherwise}\end{cases} \qquad (14)$$

$$p_2=\begin{cases}\gamma|(e_2-e_1)-(\Delta x_1+\Delta x_2)| & \text{if } \gamma|(e_2-e_1)-(\Delta x_1+\Delta x_2)|<0.5\\ 0.5 & \text{otherwise}\end{cases} \qquad (15)$$

where $\gamma=0.025$ is a positive constant for scaling.

Referring to Eqs. (14) and (15), when the penalties are nonzero, reinforcement learning with communication is driven by considering both the position trajectory errors and the measured interaction forces. Trajectory corrections Δx_1 and Δx_2 are continuously refined until the trajectory error difference (e_2-e_1) goes to zero, leading to a zero interaction force. However, unlike the case where no communication is used, Eqs. (14) and (15) provide an additional mechanism by which the two actuators can compare their relative efforts and learn how to properly contribute toward reducing the interaction force.

In order to better illustrate how using penalty functions (14) and (15) helps to promote the learning of symmetric compensatory actions, these equations are rewritten in terms of the actuator position errors, $\hat{e}_1=x_1^d+\Delta x_1-x_1$ and $\hat{e}_2=x_2^d+\Delta x_2-x_2$:

$$p_1=\begin{cases}\gamma|-(\hat{e}_2-\hat{e}_1)-2\Delta x_1| & \text{if } \gamma|-(\hat{e}_2-\hat{e}_1)-2\Delta x_1|<0.5\\ 0.5 & \text{otherwise}\end{cases} \qquad (16)$$

$$p_2=\begin{cases}\gamma|(\hat{e}_2-\hat{e}_1)-2\Delta x_2| & \text{if } \gamma|(\hat{e}_2-\hat{e}_1)-2\Delta x_2|<0.5\\ 0.5 & \text{otherwise}\end{cases} \qquad (17)$$

When each learning agent simultaneously discovers an action-selection policy for which it receives the maximum reinforcement, $r_1=r_2=1$, the argument of each penalty function, p_1 and p_2, is zero. Therefore, from Eqs. (16) and (17), it can be seen that

$$\Delta x_1=-\frac{1}{2}(\hat{e}_2-\hat{e}_1) \qquad (18)$$

and

$$\Delta x_2=\frac{1}{2}(\hat{e}_2-\hat{e}_1) \qquad (19)$$

Equations (18) and (19) imply that, under correct operation of the system, the output of each neural network should be half of the actuator position error difference $(\hat{e}_2-\hat{e}_1)$. By expanding and collecting the terms of Eqs. (18) and (19), we obtain

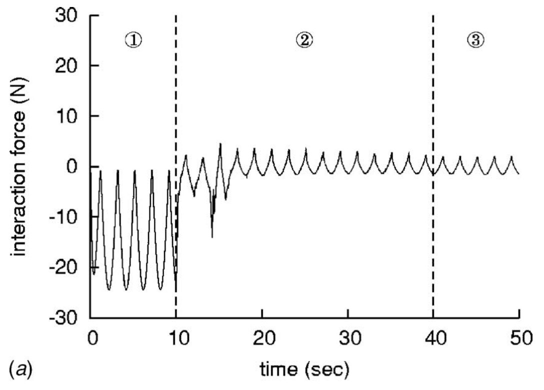
$$-(e_2-e_1)-(\Delta x_1+\Delta x_2)=0 \qquad (20)$$

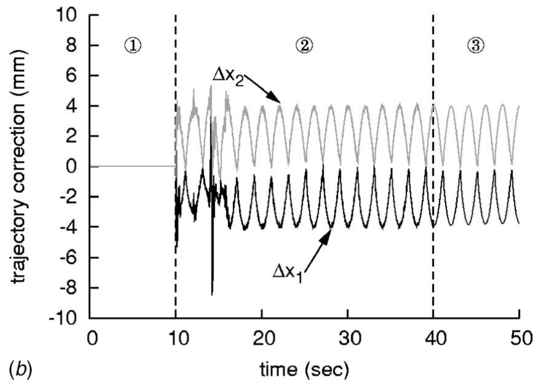
$$(e_2-e_1)-(\Delta x_1+\Delta x_2)=0 \qquad (21)$$

Equations (20) and (21) are the arguments of the original penalty functions (14) and (15), when $p_1=p_2=0$. Simultaneous solution of these equations requires that both $(e_2-e_1)=0$ and $(\Delta x_1+\Delta x_2)=0$. The latter equality implies that $\Delta x_1=-\Delta x_2$. Therefore, in order for the actuators to simultaneously receive the maximal reinforcement, they must learn to make $(e_2-e_1)\rightarrow 0$ by applying equal and opposite trajectory corrections. In other words, the use of the penalty functions, defined by Eqs. (14) and (15), resolves the credit assignment problem and keeps the actuators from receiving their full rewards until their trajectory corrections are adjusted to become symmetric. As will be seen later, in both simulations and experiments, allowing communication between the actuators, in this way, does indeed enable them to learn how to properly select complimentary actions.

Figure 7 reports typical time histories of the interaction force, the trajectory corrections, and the actuator position trajectory errors before, during, and after learning in the presence of communication between the actuators. As is seen, the interaction forces are reduced similar to Fig. 5(*a*), but the use of communication has affected learning such that the actuators learn a highly cooperative positioning strategy where the correction effort of each is similar in magnitude (compare Fig. 7(*b*) to Fig. 5(*b*)). Hence, the hydraulic actuators are better able to work as a team toward reducing the interaction force.

To establish the general behavior of the multiactuator reinforcement learning system in the presence of communication, 500 40 s simulations were carried out. As before, for each trial, learning was allowed for the first 30 s of simulated time followed by a 10 s period in which the performance of the actuators was evaluated as they handled the object using their previously learning behaviors. In addition, the same network initial conditions as in Sec. 4.2 were used in the simulations, and it was assumed that communication occurred at every learning opportunity. It was observed that in the presence of communication, the actuators could learn how to reduce the interaction force similarly to Fig. 6(*a*). The corresponding trajectory correction ratios of actuator 1, using the communication scheme, are shown in Fig. 8. With reference to Fig. 8, it is observed that the correction effort ratios of agent 1 are all clustered around 0.5 with all the data points lying between 0.3 and 0.7. Hence, the rms output levels of each neural controller are similar and the burden of reducing the interaction forces is shared more evenly between the communicating actuators. Consequently, the level of cooperation achieved as a result of reinforcement learning is improved significantly in the presence of interactuator communication.





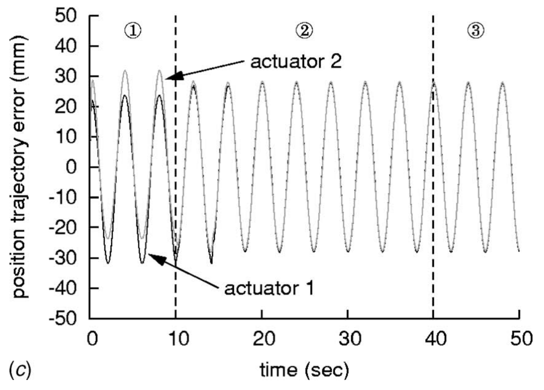


Fig. 7 Typical performance on coordination task with communication: (*a*) interaction force; (*b*) trajectory corrections, Δx_1 and Δx_2; (*c*) position trajectory errors, $x_1^d - x_1$ and $x_2^d - x_2$. Legend: (1) before learning; (2) learning; (3) after learning.

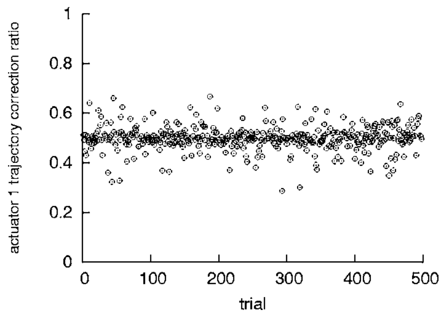


Fig. 8 Actuator 1 trajectory correction ratios after learning using communication scheme

5 Experimentation

5.1 Experimental Setup. In order to test the proposed reinforcement learning architecture in a practical environment, experiments were carried out using real hydraulic actuators. A photograph of the experimental test rig, which consists of two independently controlled hydraulic rams, is shown in Fig. 9. The first actuator is a double rod type having a 610 mm (24 in.) stroke, 38.1 mm (1.5 in.) bore, and 25.4 mm (1 in.) rods. It is controlled by a Moog D765 flow-control servovalve. The second actuator is similar to the first one but has a shorter 200 mm (8 in.) stroke and is controlled by a Moog 31 series servovalve. Both actuators are powered by a common hydraulic supply operating at a nominal pressure of 7.6 MPa (1100 psi).

The position of each actuator is measured using a cable-driven optical rotary encoder and is monitored by the corresponding personal computer (PC) via a Keithley M5312 quadrature incremental encoder card. Each PC is also equipped with a DAS-16F input/output board that is used to send the control signal generated by the software implemented control algorithm to the servovalve. The two PCs have been connected using a dedicated local area network with user datagram protocol (UDP) sockets to establish a channel for experiments involving communication. The network link is also used to synchronize the clocks of each PC at the beginning of each experiment.

The deformable object was simulated in experiments using the fixture shown in Fig. 9 (inset). The fixture, which acts nearly as a pure stiffness, consists of two banks of compression springs arranged in parallel. The particular bank of springs used to generate the resistive force depends upon the sign of the net force applied by the actuators. The nominal object stiffness (5.3 kN/m) can be adjusted by adding or removing pairs of springs. The interaction force was measured using a load cell bolted to one side of the fixture. The load cell output was read by each PC using the DAS-16F board and was conditioned, in software, using a critically damped low pass filters having two real poles located at $s = -25$ before being processed by the neural networks.

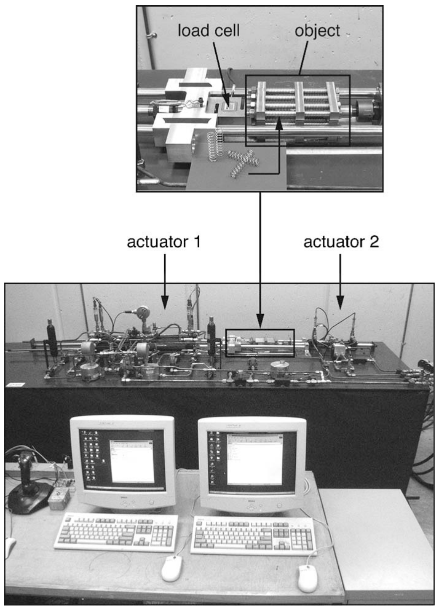


Fig. 9 Photograph of test rig upon which experiments were carried out. Inset: Close-up view of deformable object and load cell.

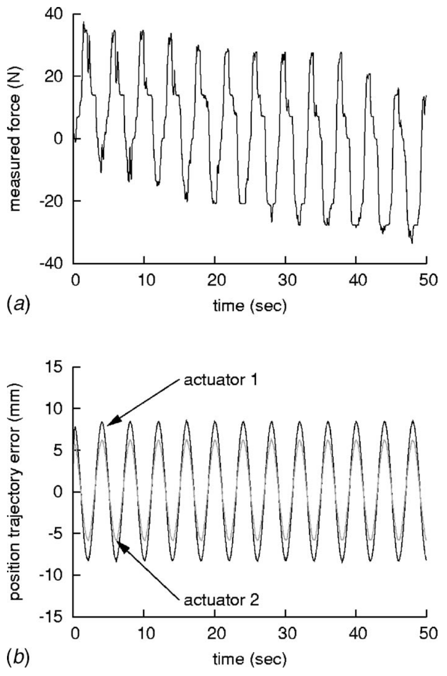


Fig. 10 Benchmark experimental performance on sinusoidal trajectory (22) before learning: (*a*) measured force; (*b*) position trajectory errors, $x_1^d - x_1$ and $x_2^d - x_2$

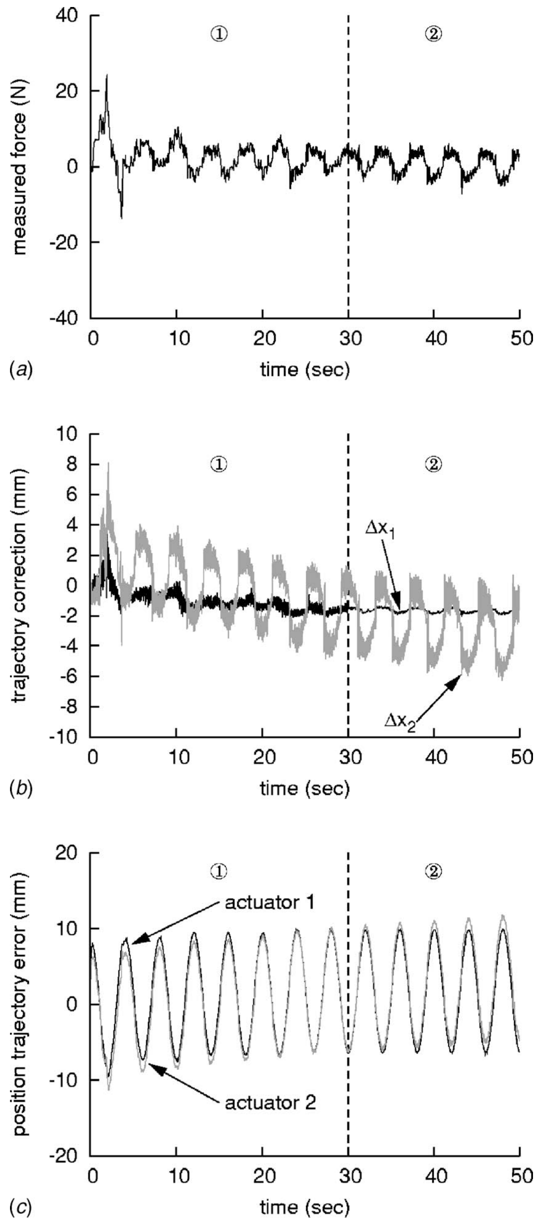


Fig. 11 Typical experimental performance of noncommunicating agents on coordination task: (*a*) measured force; (*b*) trajectory corrections Δx_1 and Δx_2; (*c*) position trajectory errors, $x_1^d - x_1$ and $x_2^d - x_2$. Legend: (1) learning; (2) after learning.

5.2 Results

5.2.1 Benchmark Performance. To implement the learning test in experiments, sinusoidal test signals similar to the ones used in the numerical simulations, but scaled to accommodate the sizes of the experimental actuators, were used:

$$x_1^d = 50 \sin(0.5\pi t) + 485 \text{ mm} \tag{22}$$

$$x_2^d = 50 \sin(0.5\pi t) + 705 \text{ mm} \tag{23}$$

Figure 10 shows the benchmark performance of the experimental system under proportional control only. The proportional controller gain was set to $K_p = 250$ V/m for each actuator. Referring to Fig. 10(*a*), the measured force ranges about 40 N peak to peak. It is also observed that while the peak-to-peak magnitude of the force remains about the same, the mean value of the force decreases over time. This is a result of imperfect position sensing and small differences in the starting times of each actuator, issues which cannot be escaped in a practical setup. The benchmark positioning errors for each actuator are included in Fig. 10(*b*) for reference.

5.2.2 Performance of Noncommunicating Actuators. The experimental performance of noncommunicating reinforcement learning actuators was tested first in order to evaluate the ability of the real-world multiactuator team to reduce the measured force acting on the object. Sinusoidal test signals (22) and (23) were used to facilitate the learning process. As in the numerical simulations, the adjustable network weights were initialized with random numbers on the interval $[-0.5, 0.5]$. Learning was allowed for the first 30 s of the experiment. For the last 20 s of the experiment, no learning was allowed and the hydraulic actuators handled the object using their previously learned behaviors. In the experiments, the nominal learning rate observed was approximately 80 Hz. As a practical consideration, the software algorithm was set up so that the learning process was reinitialized whenever the absolute value of the measured force exceeded 100 N. In practice, such a threshold is needed to prevent damage to the object due to the application of unexpected excessive force. It was observed that the measured force was reduced significantly in all but 1 of the 20 trials conducted. Typical experimental performance, which is in qualitative agreement with the results of the numerical simulations (see Fig. 5), is shown in Fig. 11.

Referring to Fig. 11, it is observed that the measured force is reduced almost immediately from the 40 N peak-to-peak benchmark value (see Fig. 10(*a*)) to about 10 N peak to peak during learning (time interval (1)). This is a marked improvement over the benchmark performance of the system. The corresponding trajectory modifications, which illustrate the corrective actions applied by the neural networks, are shown in Fig. 11(*b*). As in simulations, it is observed that in the absence of communication, the hydraulic actuators do not learn to equally share the burden of reducing the measured force. Comparing Fig. 11(*c*), which reports the position error of each actuator with respect to the desired trajectory, to Fig. 10(*b*), it is observed that the force has been reduced with negligible impact on positional accuracy. Figure 11(*c*) also reveals that in the experiment, the hydraulic actuators have reduced the force by learning how to match their position trajectory errors (see Fig. 11(*c*) for $20 < t < 30$ s). After learning has ceased (time interval (2)), the multiactuator team maintained its performance and kept the measured values of the force less than 10 N peak to peak.

5.2.3 Performance With Communication. The experimental performance of reinforcement learning actuators was tested next in the presence of communication. As before, the actuators learned for the first 30 s of the experiment and handled the objects using their previously learned behaviors for the remaining 20 s of the test. Twenty trials were conducted and the multiactuator team was observed to learn how to significantly reduce the measured value of the force in all but two trials.

Typical experimental results with communication are shown in Fig. 12. Figure 12(*a*) shows that the measured force is reduced very quickly after learning begins (time interval (1)) from 40 N peak to peak down to about 20 N peak to peak. This performance is maintained after learning has ceased during time interval (2). Figure 12(*c*) shows that the learning process had minimal impact on the position error of each actuator with respect to the desired trajectory. More importantly, however, it is observed that the use of communication has enabled the actuators to learn a highly cooperative positioning strategy where the correction effort of each actuator is similar in magnitude (compare time interval (2) of Figs. 11(*b*) and 12(*b*)). The experimental results thus confirm the observation that the hydraulic actuators are better able to learn how to work as a team toward reducing the interaction force when they are permitted to communicate their position errors periodically during learning.

Two final experiments were conducted using communication to further evaluate the practical performance of the reinforcement learning system. In the first experiment, the robustness of the reinforcement learning control architecture to changes in the object properties was assessed. The second experiment tested the generalization of previously learned control actions to more realistic position trajectories that include starting and stopping motions.

In the first experiment, the stiffness of the object was increased by 25% from 5.3 kN/m to 6.6 kN/m. Since the object stiffness has increased, the measured force is more sensitive to the neural network trajectory corrections. Consequently, as discussed previously in Sec. 3, the envelope of stochastic exploration had to be reduced in size by scaling Eq. (4) by a factor of 0.6, i.e., $\sigma = 0.6(1-\hat{r})$, to obtain good learning performance. The neural network weights were initialized with random values and the same learning experiment as in the previous section was conducted. Typical measured forces and actuator trajectory corrections for the object with increased stiffness are shown in Fig. 13(*a*). As expected, the peak-to-peak values of the interaction force have been reduced by approximately 50% from 60 N (using the proportional controller only) to approximately 30 N as a result of reinforcement learning. The performance of the system is also maintained after learning has ceased. Referring to Fig. 13(*b*), the use of communication has again ensured that both hydraulic actuators learn to contribute equally to reducing the measured force.

To test the generalization of previously learned control actions to position trajectories that require the actuators to start and stop, the test trajectory shown in Fig. 14(*a*) was used. The benchmark interaction force under proportional control only for this test trajectory is shown in Fig. 14(*b*). As is seen, the measured force ranges between ±20 N, and the mean value is observed to drift slightly over time.

The experimental performance of the reinforcement learning system for the generalization test is shown in Fig. 15. In this experiment, the neural network weights were first adapted during a 30 s pretraining period using the sinusoidal reference trajectories (22) and (23). Next, the actuators were commanded to move the object according to the desired trajectories shown in Fig. 14(*a*). For the first 15 s of the experiment (time interval (1)), no additional learning was allowed and the actuators handled the object using their previously learned behaviors. Figure 15(*a*) illustrates that the neural controllers can reduce the measured force from 40 N peak to peak to approximately 10 N peak to peak. Hence, the trajectory corrections learned while training on the sinusoidal reference trajectory seem to generalize well to more realistic manipulation tasks where the actuators are required to stop and start. It is also observed that no significant improvement or degradation in performance is obtained with additional learning (time interval (2)).

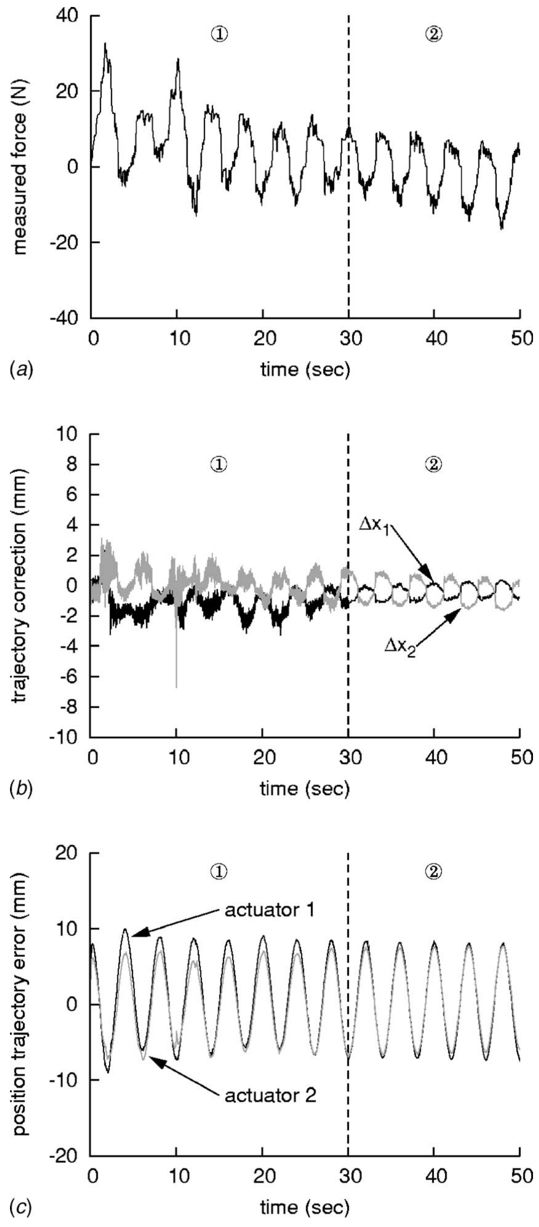


Fig. 12 Typical experimental performance on coordination task with communication: (*a*) measured force; (*b*) trajectory corrections Δx_1 and Δx_2; (*c*) position trajectory errors, $x_1^d - x_1$ and $x_2^d - x_2$. Legend: (1) learning; (2) after learning.

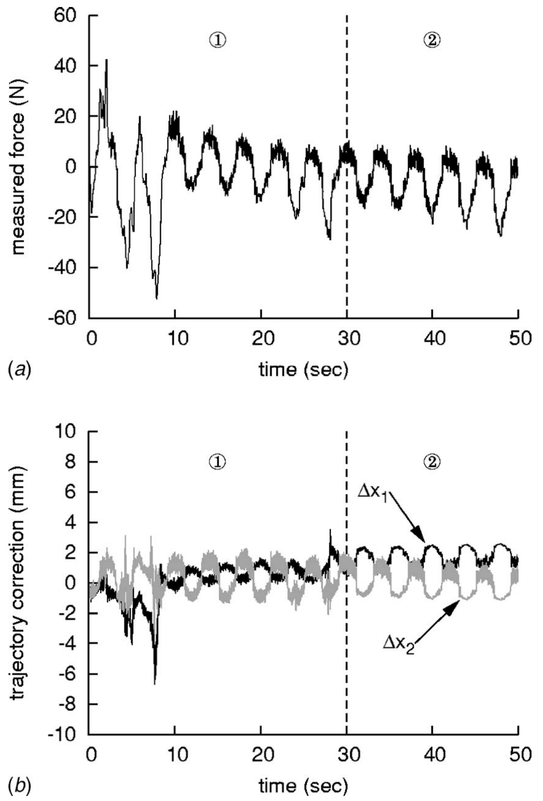


Fig. 13 Typical experimental performance with communication and increased object stiffness: (*a*) measured force; (*b*) trajectory corrections Δx_1 and Δx_2. Legend: (1) learning; (2) after learning.

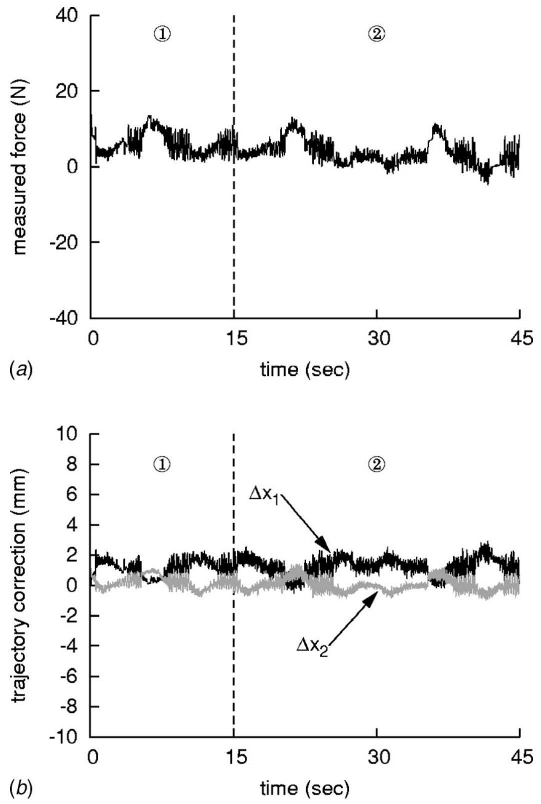


Fig. 15 Typical experimental performance for trajectory following test task with communication: (*a*) measured force; (*b*) trajectory corrections Δx_1 and Δx_2. Legend: (1) after pretraining on sinusoidal trajectory for 30 s; (2) additional learning.

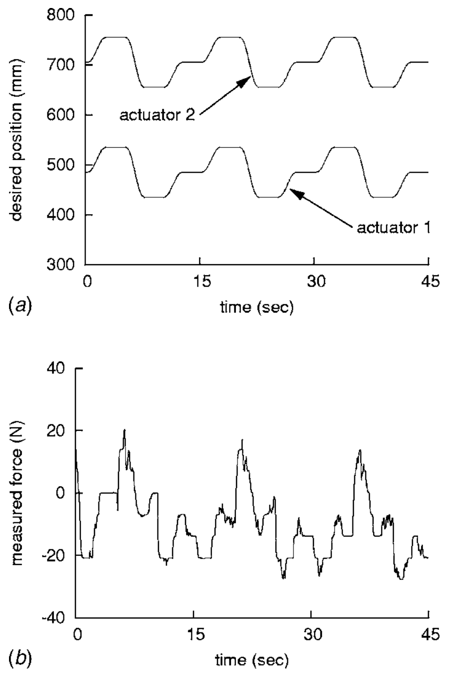


Fig. 14 Benchmark experimental performance on test trajectory with starting and stopping motions: (*a*) desired position; (*b*) measured force

6 Conclusions

In this paper, decentralized multiagent reinforcement learning has been applied to the problem of coordinating the motion of two horizontal hydraulic actuators engaged in moving an object along a specified position trajectory. To reduce the interaction force acting on the object, which arises as a result of imperfect closed-loop position control, each hydraulic actuator was outfitted with a RLNN that can modulate the measured force between the implement and the object by modifying the local prescribed formation constrained reference trajectory. The reinforcement learning goal was to enable each actuator to (i) learn how to select actions that reduce the object interaction force and (ii) to learn a cooperative control strategy where the burden of reducing the interaction force is shared equally amongst the actuators. To achieve these objectives, the weights of each neural network were updated on-line using a modified form of a reinforcement learning algorithm described previously in the literature, which maximizes the immediate reward. The efficacy of the proposed decentralized coordinated motion control scheme was proven using both simulation studies and experiments.

The experimental results showed that the reinforcement learning hydraulic actuators could indeed learn a decentralized control strategy to improve their coordination toward reducing the object interaction force with little effect on positional accuracy and without any knowledge of the other actuator's state. However, the experimental results also showed that without some knowledge of the other's state, the actuators could not learn how to select complimentary control actions that allow them to equally share their efforts. This problem, which arises due to the credit assignment dilemma, was alleviated by allowing the actuators to communicate their local position errors periodically during learning. Experiments showed that the quality of coordinated motion, where each actuator contributes equally to reducing the force, could be improved by permitting interactuator communication. These general characteristics of the inherently stochastic reinforcement learning system were confirmed via simulation studies. This paper has thus established reinforcement learning as a promising technique for synchronizing the motion of several nonlinear hydraulic manipulators in a single degree of freedom. Future research efforts in this area should examine the applicability of the approach toward the coordinated motion control of multiple degree-of-freedom hydraulic robots manipulating objects along several dimensions. The scalability of the approach to more than two manipulators should also be investigated. In addition, the reinforcement learning control technique proposed in this paper may also be relevant in applications involving prime movers other than fluid power actuators.

Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada who provided financial support for this research.

References

[1] Vuckbrotovic, M., and Tuneski, A. I., 1998, “Mathematical Model of Multiple Manipulators: Cooperative Compliant Manipulation on Dynamical Environments,” Mech. Mach. Theory, **33**, pp. 1211–1239.
[2] Braun, B. M., Starr, G. P., Wood, J. E., and Lumia, R., 2004, “A Framework for Implementing Cooperative Motion on Industrial Controllers,” IEEE Trans. Rob. Autom., **20**, pp. 583–589.
[3] Arimoto, S., Miyazaki, F., and Kawamura, S., 1987, “Cooperative Motion Control of Multiple Robot Arms or Fingers,” *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, pp. 1407–1412.
[4] Kopf, C. D., and Yabuta, T., 1988, “Experimental Comparison of Master/Slave and Hybrid Two Arm Position/Force Control,” *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, pp. 1633–1637.
[5] Raibert, M. H., and Craig, J. J., 1981, “Hybrid Position/Force Control of Manipulators,” ASME J. Dyn. Syst., Meas., Control, **102**, pp. 126–133.
[6] Yoshikawa, T., and Zheng, X.-Z., 1993, “Coordinated Dynamic Hybrid Position/Force Control for Multiple Robot Manipulators Handling One Constrained Object,” Int. J. Robot. Res., **12**(3), pp. 219–230.
[7] Perdereau, V., and Drouin, M., 1996, “Hybrid External Contol for Two Robot Coordinated Motion,” Robotica, **14**, pp. 141–153.
[8] Liu, Y.-H., and Arimoto, S., 1996, “Distributively Controlling Two Robots Handling an Object in the Task Space Without any Communicaiton,” IEEE Trans. Autom. Control, **41**(8), pp. 1193–1198.
[9] Uzmay, I., Burkan, R., and Sarikaya, H., 2004, “Application of Robust and Adaptive Control Techniques to Cooperative Manipulation,” Control Eng. Pract., **12**, pp. 139–148.
[10] Woon, L. C., Ge, S. S., Chen, X. Q., and Zhang, C., 1999, “Adaptive Neural Network Control of Coordinated Manipulators,” J. Rob. Syst., **16**(4), pp. 195–211.
[11] Nakayama, T., Arimoto, S., and Naniwa, T., 1995, “Coordinated Learning Control for Multiple Manipulators Holding an Object Rigidly,” *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, **2**, pp. 1529–1534.
[12] Schneider, S. A., and Cannon, R. H., Jr., 1992, “Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results,” IEEE Trans. Rob. Autom., **8**(3), pp. 383–394.
[13] Zeng, H., and Sepehri, N., 2005, “Nonlinear Position Control of Cooperative Hydraulic Manipulators Handling Unknown Payloads,” Int. J. Control, **78**(3), pp. 196–207.
[14] Mulder, M. C., and Malladi, S. R., 1991, “A Minimum Effort Control Algorithm for a Cooperating Sensor Driven Intelligent Multi-Jointed Robotic Arm,” *Proceedings of the 30th IEEE Conference on Decision and Control*, Brighton, UK, **2**, pp. 1573–1578.
[15] Zeng, H., and Sepehri, N., 2007, “On Tracking Control of Cooperative Hydraulic Manipulators,” Int. J. Control, **80**(3), pp. 454–469.
[16] Sutton, R. S., and Barto, A., 1998, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA.
[17] Kaebling, L. P., Littman, M. L., and Moore, A. W., 1996, “Reinforcement Learning: A Survey,” J. Artif. Intell. Res., **4**, pp. 237–285.
[18] Sutton, R. S., 1984, “Temporal Credit Assignment in Reinforcement Learning,” Ph.D. thesis, University of Massachusetts, Amherst.
[19] Russell, S., and Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ.
[20] Watkins, C. J. C. H., 1989, “Learning With Delayed Rewards,” Ph.D. thesis, Cambridge University, Cambridge.
[21] Williams, R. J., 1992, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” Mach. Learn., **8**(3), pp. 229–256.
[22] Barto, A., Sutton, R. S., and Anderson, C. W., 1983, “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems,” IEEE Trans. Syst. Man Cybern., **13**, pp. 834–846.
[23] Anderson, C. W., 1989, “Learning to Control an Inverted Pendulum Using Neural Networks,” IEEE Control Syst. Mag., **9**(3), pp. 31–37.
[24] Gullapalli, V., Franklin, J. A., and Benbrahim, H., 1994, “Acquiring Robot Skills via Reinforcement Learning,” IEEE Control Syst. Mag., **14**, pp. 13–24.
[25] Tzasfestas, S. G., and Rigatos, G. G., 2002, “Fuzzy Reinforcement Learning Control for Compliance Tasks of Robotic Manipulators,” IEEE Trans. Syst., Man, Cybern., Part B: Cybern., **32**, pp. 107–113.
[26] Stone, P., and Veloso, M., 2000, “Multiagent Systems: A Survey From a Machine Learning Perspective,” Auton. Rob., **8**, pp. 345–383.
[27] Gullapalli, V., 1990, “Stochastic Reinforcement Learning Algorithm for Learning Real-Valued Functions,” Neural Networks, **3**(6), pp. 671–692.
[28] Mataric, M. J., 1998, “Using Communication to Reduce Locality in Distributed Multi-Agent Learning,” J. Exp. Theor. Artif. Intell., **10**(3), pp. 357–369.
[29] Merritt, H., 1967, *Hydraulic Control Systems*, Wiley, New York.
[30] Karpenko, M., and Sepehri, N., 2003, “Robust Position Control of an Electrohydraulic Actuator With a Faulty Actuator Piston Seal,” ASME J. Dyn. Syst., Meas., Control, **125**(3), pp. 413–423.
[31] Bishop, C. M., 1995, *Neural Networks for Pattern Recognition*, Oxford University Press, New York.