Human-Like Rewards to Train a Reinforcement Learning Controller for Planar Arm Movement

Kathleen M. Jagodnik, *Member, IEEE*, Philip S. Thomas, Antonie J. van den Bogert, Michael S. Branicky, *Fellow, IEEE*, and Robert F. Kirsch, *Member, IEEE*

***Abstract*—High-level spinal cord injury (SCI) in humans causes paralysis below the neck. Functional electrical stimulation (FES) technology applies electrical current to nerves and muscles to restore movement, and controllers for upper extremity FES neuroprostheses calculate stimulation patterns to produce desired arm movement. However, currently available FES controllers have yet to restore natural movements. Reinforcement learning (RL) is a reward-driven control technique; it can employ user-generated rewards, and human preferences can be used in training. To test this concept with FES, we conducted simulation experiments using computer-generated "pseudo-human" rewards. Rewards with varying properties were used with an actor-critic RL controller for a planar two-degree-of-freedom biomechanical human arm model performing reaching movements. Results demonstrate that sparse, delayed pseudo-human rewards permit stable and effective RL controller learning. The frequency of reward is proportional to learning success, and human-scale sparse rewards permit greater learning than exclusively automated rewards. Diversity of training task sets did not affect learning. Long-term stability of trained controllers was observed. Using human-generated rewards to train RL controllers for upper-extremity FES systems may be useful. Our findings represent progress toward achieving human–machine teaming in control of upper-extremity FES systems for more natural arm movements based on human user preferences and RL algorithm learning capabilities.**

***Index Terms*—Control, functional electrical stimulation (FES), human–machine teaming, modeling, rehabilitation, reinforcement learning (RL), simulation, upper extremity.**

I. Introduction

SPINAL cord injuries (SCIs) can result in paralysis below the level of injury. High-level SCI affects the cervical C1–C4 levels, resulting in partial or total paralysis below the neck. Repair of the spinal cord by regeneration remains an unsolved problem [1], [2]. As an alternative, functional electrical stimulation (FES) technology uses electrodes to apply electrical current to the nerves or muscles to restore function [1], [3]. For individuals with high-level SCI, the restoration of upper-extremity function is important to allow independence in performing daily tasks and the ability to perform work-related functions. Restoring even modest levels of functionality can yield significant gains in quality of life [4].

Presently, a range of FES controllers exists to restore upper-extremity function. Feedforward, or open-loop, control calculates and applies muscular stimulation patterns without the use of sensors to provide feedback information. At present, feedforward control is the predominant method of FES control used clinically [3], [5], due to its ease of implementation. However, feedforward control succeeds only when the actual system being controlled closely matches the modeled system for which the controller has been designed; otherwise, performance of these controllers will be poor [6].

Feedback control uses sensors that allow the correction of deviations from intended movements [5], [6]. As a result, movements produced using feedback controllers tend to be more accurate than those produced using feedforward control [7]. Implantable sensors are becoming available [8], [9] that will facilitate the use of FES feedback control.

While the currently available FES controllers have advanced the range of upper-extremity function available, challenges remain before natural and efficient movements that are tailored to individual users will be achieved. The physiological properties of the arm can never be fully known [10], and the arm is subject to change with time (e.g., spasticity, muscular fatigue, and muscular strengthening as a result of training). Environmental variations will also occur, including objects with substantial mass being held in the hand. An ideal upper-extremity FES controller should be able to adjust to physiological and environmental situations that cannot be taken into account during the controller's design.

Reinforcement learning (RL) algorithms learn from experience through trial and error, and they require a scalar reinforcement (or cost) signal. This is a fitting paradigm for FES control, because although the optimal muscle stimulations are not known, the resulting quality of an arm movement can be captured in a scalar reward signal to produce controller learning.

A feature of RL that is useful for upper-extremity FES control is its ability to incorporate human guidance. A human may influence the learning of an RL controller through the use of

Manuscript received August 13, 2015; revised January 20, 2016 and April 12, 2016; accepted April 23, 2016. This work was supported by the National Institutes of Health fellowship #TRN030167, Veterans Administration Rehabilitation Research and Development predoctoral fellowship "Reinforcement Learning Control for an Upper-Extremity Neuroprosthesis," NIH Training Grant T32 EB004314, and Ardiem Medical Arm Control Device Grant #W81XWH0720044. This paper was recommended by Associate Editor K. Li.

K. M. Jagodnik is with the Fluid Physics and Transport Processes Branch, NASA Glenn Research Center, Cleveland, OH 44135 USA (e-mail: kathleen.jagodnik@nasa.gov).

P. S. Thomas is with the Brunskill Laboratory, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: pthomas@cs.umass.edu).

A. J. van den Bogert is with the Department of Mechanical Engineering, Cleveland State University, Cleveland, OH 44115 USA (e-mail: a.vandenbogert@csuohio.edu).

M. S. Branicky is with the School of Engineering, University of Kansas, Lawrence, KS 66045 USA (e-mail: msb@ku.edu).

R. F. Kirsch is with the Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH 44106 USA (e-mail: rfk3@case.edu).

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

Digital Object Identifier 10.1109/THMS.2016.2558630

rewards. The human can view actions performed by the RL controller, and depending on the quality perceived by the human trainer, he may assign positive or negative rewards to provide feedback to the controller about whether its strategy is acceptable. This technique to incorporate human guidance into RL controller learning is referred to as *shaping*. Shaping has the advantage that the human user needs only observe the actions performed by the controller and then assign rewards, which can be as simple as a binary +1/–1 choice. The human does not need to model the desired action, nor does he need to provide specific guidance about which action should be taken.

We posit that using human rewards to shape RL control for upper-extremity function restoration is a promising technique. Because this shaping process using rewards allows flexible learning, it is possible for the controller to adapt to changing muscular and other physiological properties, and user preferences that may evolve over time. This work expands upon our previous efforts to develop effective RL-based methods of control for upper-extremity FES systems [15], [23], which did not incorporate human-generated rewards. In this study, we investigate whether it is feasible to use human-like rewards to shape RL controller learning for the upper-extremity FES control problem. Because experimenting with control development requires many thousands of training episodes, it is not feasible to use human-generated rewards for all stages of this investigation. Instead, we will use so-called pseudo-human rewards generated by a computer algorithm that simulate the rewards that a human trainer would be likely to give. Similarly, we use a biomechanical planar arm model in place of a real human arm, as performing all experimentation on human subjects implanted with FES systems would be impractical. Performing simulation experiments as a precursor to human experimentation is a useful and necessary step to ensure that the most effective, efficient, and safe control methods possible are developed.

Human rewards have a number of properties that recommend their use to train RL controllers. Most fundamentally, humans can judge qualities of controller-generated movements that can be prohibitively difficult to model using software. Achieving movements in the FES arm that appear natural and effortless is the ultimate goal of our work; humans can easily judge whether movements meet these criteria, while attempting to program these qualities into the action-generating policy of a controller would be extremely challenging. Human rewards also allow individual users to specify their particular preferences for controller behavior, which would not be known in advance by an engineer designing the control algorithm. Shaping RL controller behavior using human-generated rewards may also increase learning speed [11]. Controllers can calculate actions and rewards based solely on the information that they are programmed to access; in contrast, humans can take a long-term view of actions and may give rewards based not only on the current state, but also on implied or imagined future consequences of the current state or action [11].

While human rewards promise benefits for RL controller training, they can also represent a challenging addition to RL. Computers can update state information and generate rewards on a millisecond timescale. In contrast, humans typically have a reaction time of 0.3 to 0.8+ s to press a button (as when assigning a reward) in response to a visual stimulus [12]. Delays in rewards require that the RL controller assess how much of the preceding action the reward should be applied to. This temporal credit assignment problem [13] is a fundamental challenge of RL that is amplified when human-generated rewards, with their significant delays, are used.

In addition, human attention is a finite resource, and humans cannot be expected to train the RL controller by a constant sequence of viewing controller action and immediately assigning a reward; such all-consuming training behavior would not permit useful integration into the FES user's daily life. Instead, RL controllers should ideally be able to use *sparse* rewards from human trainers to effectively learn useful policies. While decreased frequency of rewards slows learning speed [14], [15], various techniques including increasing the actor's learning rate [15] have been found to compensate for slowed learning resulting from sparse rewards.

Human rewards have been used previously to train RL controllers for a variety of systems that are characterized by discrete state and/or action values. Applications have included physical or simulated robots selecting discrete actions in controlled environments [16], [17] and video games [11], [18]. These discrete-time and discrete-action systems provide a useful exploration of the fundamental properties of how human rewards shape RL control, but require an extension to continuous-state and continuous-time characteristics before they can be used in a physiological FES control system. Controllers that are developed using discretized systems may not function properly when applied to continuous systems [19] such as upper-extremity FES systems, and discretization can introduce discontinuities into a system [19].

Previous research suggests that RL may be useful for the continuous-time, continuous-state arm control that our project aims to achieve. Elevator scheduling involves both continuous time and continuous state spaces; RL has been used to improve scheduling for discrete events [20]. Pilarski *et al.* [21] used human rewards to train an RL controller for a simulated two-degree-of-freedom (DOF) robot arm with continuous state and action spaces; the controlled variables were joint angular velocities. RL has also been used to control planar 2-DOF arm models similar to the one used in this study [15], [22], [23], although human(-like) rewards were not used as inputs to shape controller learning.

Preliminary experiments can require thousands of iterations of training or more to address each research question explored, and human time and attention are limited. However, it remains to be determined whether rewards with the sparse, delayed properties similar to those generated by humans will be feasible to train RL controllers for continuous-time, continuous-state systems of a certain complexity. For these reasons, we have elected to create algorithms that generate rewards similar to those that a human would be likely to give; we refer to these computer-generated rewards as "pseudo-human" rewards. These rewards are generated in response to the current arm reaching movement properties of the system, and they are used as inputs to the RL controller. In this study, we will examine whether these

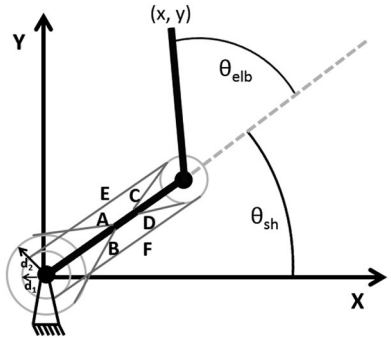

Fig. 1. Top view of the two-joint, six-muscle biomechanical arm model. *Y*-axis is anterior. Movements occur in the sagittal plane with no gravity, as sliding across a frictionless tabletop. Antagonistic muscle pairs are listed as (flexor, extensor): monoarticular shoulder muscles (A: anterior deltoid, B: posterior deltoid); monoarticular elbow muscles (C: brachialis, D: triceps brachii (short head)); and biarticular muscles (E: biceps brachii; F: triceps brachii (long head)). φ_1 and φ_2 are shoulder and elbow joint angles, respectively. Moment arm values: $d_1 = 30$ cm, $d_2 = 50$ cm. Adapted from [24] and [25].

pseudo-human rewards can effectively train an RL controller for a simulated planar arm to perform dynamic goal-oriented reaching movements. Demonstration of the feasibility of this technique will be a necessary precursor to the introduction of rewards generated by humans to train RL controllers for the restoration of voluntary arm movement to individuals with SCI.

II. Methods

This study consisted of a series of experiments applying an RL controller to a simulated planar biomechanical arm performing goal-oriented reaching movements, for the purpose of determining the feasibility of using human-like rewards to train this controller.

A. Musculoskeletal Model

A musculoskeletal model of the human arm was used that constrained movement to a horizontal plane, to approximate an arm moving along a tabletop [25] (see Fig. 1). The model consists of two segments (arm and forearm) and two joints (shoulder and elbow), both with angular ranges of [20°, 90°]. Six muscles are included in the model: four of the muscles are monoarticular, and two are biarticular. Muscles were modeled according to the Hill model [26], [27], which includes a contractile element (CE) characterized by activation dynamics as well as force-length and force-velocity properties, and elements representing the passive effects of connective tissue on the CE-generated force. Muscles were represented by two first-order ordinary differential equations [28]. The mass properties of the segments were derived from [29]. Simulations used a forward Euler approximation with updates to the controller (i.e., timesteps) every 20 ms.

B. Reinforcement Learning Controller

RL control was used for the simulation experiments in this paper. RL controllers attempt to maximize the integral of a numerical reward signal by mapping situations to actions [13] through trial-and-error search [30]. At each timestep, the RL controller calculated six muscle stimulation values (each continuously ranging from 0.0 to 1.0) based upon comparisons of the current joint angle and angular velocity state values with their target (or goal) values. Stimulations were applied to the planar arm model, and state variables were updated to allow the calculation of the next set of stimulation values. A single simulated reaching task corresponded to one episode. Each episode lasted 2 s and consisted of 100 timesteps of 20 ms each.

The environmental state, $s(t) \in S \subseteq \Re^n$, changes as a function of the actions performed, $u(t) \in U = \Re^m$, where t is the time. The transition function deterministically defines the changes to the environment resulting from specific actions: $T(s, u) : S \times U \rightarrow S$ [15]. The state vector, $s \in \Re^6$, is defined as

$$s = \left[\theta(t), \dot{\theta}(t), \theta_{\mathrm{Goal}}(t)\right]^T \tag{1}$$

where $\theta(t)$ is a vector of shoulder and elbow angles, $\dot{\theta}(t)$ is a vector of shoulder and elbow angular velocities, and $\theta_{\mathrm{Goal}}(t)$ is a vector of two goal joint angles that might be provided by a user interface. Target values of joint angular velocities were specified to be 0.

The policy of the RL agent consists of a distribution over actions, for each state. After the actions are applied to the arm model (i.e., the environment), the environment generates a reward. In our instantiation [15], the reward function maps each state to a reward based on the actions performed: $R(s, u) : S \times U \rightarrow \Re$.

A Markov decision process (MDP) mathematically models decision-making situations in which outcomes are partially determined by a decision-generating agent and partially result from random processes. Many RL problems are formulated as MDPs [31], [32]. In MDPs, the actions, rewards, and probabilities of undergoing state-to-state transitions are exclusively dependent upon the current state and action; previous states and actions do not persist [33]. Although the differential equations in the muscle model of our system [28] describe the muscle activation dynamics, some muscle properties, including tendon and muscle fiber lengths, are hidden and cannot be used by the controller. These properties cannot be measured or interpolated from other measurable values. Due to the presence of these nonobservable states, our system is not strictly an MDP, but can more accurately be characterized as a partially observable MDP [32], [34]. However, these hidden internal states are characterized by fast decay times and have minimal history dependence. We, therefore, make the assumption that the observable variables will dominate the performance of our system, and we will treat our system as an MDP, acknowledging that MDP-related convergence guarantees will not apply to these investigations and that this assumption could introduce additional challenges in real-world settings in the presence of muscle fatigue and other nonideal phenomena.

The continuous actor-critic RL algorithm [35] was selected because our system is characterized by continuous time and continuous states. This controller was implemented using C++ [15]. The continuous actor-critic architecture consists of an actor that represents the policy (or algorithm for selecting actions

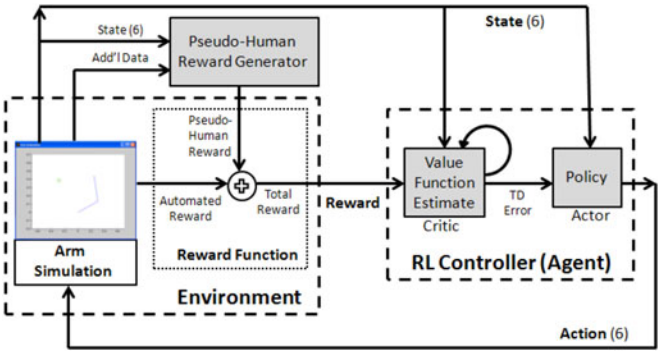

Fig. 2. Block diagram of the RL controller system using pseudo-human rewards.

based on environmental states) and a critic that approximates a value function. The critic is used to evaluate the actions of the actor at a high level, to determine whether the actor should increase or decrease the probability of an action that it selected based on the observed outcome.

The actor generates actions by

$$u(t) = S\left(A(s(t); w^A) + \sigma n(t)\right) \tag{2}$$

where $u(t)$ are a set of six muscle stimulations [represented as Action(6) within the RL controller system (see Fig. 2)], $A(\ )$ is the action-selection function, w^{A} is the vector of actor parameters that encode the policy, σ is a constant that scales exploration, and $n(t)$ defines explorational noise

$$\tau_n \dot{n}(t) = -n(t) + N(t) \tag{3}$$

where τ_n is a time constant, and $N(t)$ is the normal Gaussian noise having the same dimension as the action space. *S* is the monotonically increasing logistic function, given by

$$S(x) = \frac{1}{1 + e^{-x}}. \tag{4}$$

Temporal difference (TD) error is generated by the critic after the actor applies actions to the environment. This TD error is used to update the actor and the critic (see Algorithm 1). The continuous-time TD error [35] is given by

$$\delta_t = R(t) - \frac{1}{\tau}V(t) + \dot{V}(t) \tag{5}$$

where $R(t)$ is the reward function, $V(t\ )$ is the value function, and τ is a hyperparameter specifying a time constant that discounts future rewards. Following preliminary experimentation [15], $\tau = 0.1$ was used for all experiments involving RL control.

The policy of the actor is encoded by a vector w_A, while the critic's estimate of the value function is encoded by a vector w_C. Both vectors encode their respective functions employing fully connected feedforward artificial neural networks using logistic threshold functions.

To help mitigate the temporal and structural credit assignment problems, eligibility traces were used to implement the actor-critic algorithm. These traces keep temporary account of learning-related events such as states that have been visited and actions that have been performed. When a TD error is generated, the eligibility traces restrict learning to only those states and actions that have been recorded as eligible for the assignment of credit or blame [13]. In the absence of eligibility traces, a reward that occurs in the distant future is slowly propagated back to the states it resulted from; in contrast, eligibility traces allow the associated states to be updated immediately, the first time a reward is assigned.

Algorithm 1: The continuous actor-critic reinforcement learning algorithm [15], [35].

1: Initialize ANN actor and ANN critic weight vectors $\vec{w}_a$ and $\vec{w}_e$ via error backpropagation supervised learning to PD controller's actor policy. Then, train with actor's learning rate = 0 to initialize critic's weight vector.
2: Initialize eligibility values to zero: $\vec{e}_c = 0$
3: Repeat for each episode (reaching task)
4: $s \leftarrow$ initial state of the system
5: Repeat for each 20 ms time step within episode
6: Calculate muscle stimulation levels a: $a \leftarrow \pi(s) + n(t)$
7: Apply muscle stimulations to arm model
8: Alow 20 ms to elapse
9: Calculate next state *s'* and reward: $r_{\mathrm{Total}} = r_{\mathrm{Automated}} + r_{\mathrm{Human}}$
10: Compute TD error: $\delta(t) = r(t) + \frac{1}{\Delta t}\left[\left(1 - \frac{\Delta t}{\tau}\right)V(t) - V(t - \Delta t)\right]$
11: Update critic eligibility traces: $\kappa e_i(t) = -e_i(t) + \frac{\delta V(s(t); w)}{\delta w_i}$
12: Update actor: $\dot{w}_i^A = \eta_A \delta(t) n(t) \frac{\delta A(s(t); w^A)}{\delta w_i^A}$
13: Update critic weights: $\dot{w}_i = \eta_C \delta(t) e_i(t)$
14: Until maximum episode length reached
15: Unit maximum rin length reached

Symbols are defined as follows: $\pi(s)$: actor's policy; *n(t)*—explorational noise [defined in (3)]; *r(t)*: reward at timestep *t*; *V(t)*: evaluation of the value function at timestep *t*; κ: constant to scale eligibility traces over time; η_A: actor learning rate; and η_C: critic learning rate.

The RL controller receives rewards from the arm model, referred to as the automated reward, according to [25]

$$r_{\mathrm{Automated}}(t) = W \sum_{i=1}^{6} u_i^2 - \sqrt{(x_{\mathrm{Goal}} - x)^2 + (y_{\mathrm{Goal}} - y)^2} \tag{6}$$

where $W = -0.016$ [15], u is a vector of six muscle stimulations, and (x, y) is the current wrist position calculated

according to

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos(\theta_{\mathrm{sh}}) + L_2 \cos(\theta_{\mathrm{sh}} + \theta_{\mathrm{elb}}) \\ L_1 \sin(\theta_{\mathrm{sh}}) + L_2 \sin(\theta_{\mathrm{sh}} + \theta_{\mathrm{elb}}) \end{bmatrix} \tag{7}$$

where θ_{sh} is shoulder angle, θ_{elb} is elbow angle, L_1 is the length of the upper arm, L_2 is the length of the forearm, and both segments were assumed to have identical lengths. The target wrist position is denoted $(x_{\mathrm{Goal}}, y_{\mathrm{Goal}})$ and is calculated analogously to (7), using the target shoulder and elbow joint angles. The calculated reward punishes the agent proportionally to the distance of the wrist from its target position and the magnitude of muscle stimulation values.

The actor-critic RL control algorithm is given in Algorithm 1. After the actor generates a set of six muscle stimulations (2), these actions are applied to the arm model. Based upon the changes that result from these actions being applied to the environment, the critic generates a TD error, and this error is used to update the actor and the critic (see Algorithm 1 and Fig. 2). During this process, the critic remains on-policy: the value function that it approximates corresponds to the *current* policy of the actor [13].

1) RL Controller Initialization: During preliminary experiments [15], it was determined that the actor-critic RL architecture for planar arm control does not learn well under *tabula rasa* conditions. Therefore, we initialized our RL controller to perform similarly to an optimized proportional-derivative (PD) controller. Parameters for this optimized controller were taken from [25]. This PD-initialized RL controller was used in every experiment presented here.

The three flexor muscles of the model were weakened by 50% in order to present a realistic challenge for controller learning, approximating a human subject with significant muscular atrophy [39], [40]. All simulation experiments used this weakened arm model.

C. Pseudo-Human Rewards for Reinforcement Learning Controller

Because using human rewards for all experiments involving the RL controller would be prohibitively time-consuming during preliminary testing, these simulation experiments instead use "pseudo-human" rewards (see Algorithm 2), which are computer-generated rewards that approximate how a human reward-giver would assign rewards. A variety of such pseudo-human rewards was used, and will be described specifically for each experiment.

Pseudo-human rewards were added to the environmentally generated reward, according to

$$r_{\mathrm{Total}} = r_{\mathrm{Automated}} + \nu r_{\mathrm{Pseudo-Human}} \tag{8}$$

where r_{Total} is the total reward, $r_{\mathrm{Automated}}$ is the reward generated from the arm model, $r_{\mathrm{Pseudo-Human}}$ is pseudo-human-generated reward, and ν is a constant weighting factor.

Algorithm 2: Assign pseudo-human rewards

```
UpperOvershootThreshold = 0.1
LowerOvershootThreshold = 0.2
At final timestep of each movement:
if AtTarget and InDwellState
   and (MaxOvershoot < UpperOvershootThreshold) then
      FinalTimestepReward = 2
else if AtTarget and InDwellState then
      FinalTimestepReward = 1
else if NotAtTarget and ReachedTargetButExited
   and (maxOvershoot < LowerOvershootThreshold) then
      FinalTimestepReward = −1
else if (MaxOvershoot > LowerOvershootThreshold)
   then
      FinalTimestepReward = −2
else FinalTimestepReward = 0
end if
```

D. Performance Metrics

The recorded performance metrics for the performed goal-oriented reaching movements, which were used to evaluate controller performance, were defined as follows:

The size of the target zone was selected to be small enough to require significant accuracy in the controlled reaching movements, while not being so prohibitively small that few episodes would complete successfully. The magnitudes of the cumulative reward (0.1) and the final-timestep reward (1.0) used were selected as the result of preliminary experiments.

1) Dwell-At-Target Success: At the final timestep of each reaching movement episode, wrist position was evaluated relative to the target position. If the wrist was within the target zone and had continuously remained within this target zone for at least 100 ms, the episode was scored as a success. Otherwise, the episode was counted as a failure. Success percentages were calculated over the set of 500 episodes performed per session; success percentages for subsets of 100 episodes were also calculated.

2) Net Learning: For each set of 500 reaching movement episodes, the first 100 and the final 100 episodes were evaluated. The dwell-at-target success percentage for each data subset was recorded, and the net learning over 500 episodes was defined as the difference $(\mathrm{success}_{\mathrm{Final}} - \mathrm{success}_{\mathrm{Initial}})$.

3) Time to Achieve Dwell State: For each set of reaching movement episodes, we recorded the time (in seconds) that each episode required to achieve the dwell state and remain there. Recorded times corresponded to the first timestep following the wrist continuously dwelling in the target zone for 100 ms. If the wrist achieved the dwell state but subsequently exited it, Dwell Success was reset to Dwell Failure. For any episodes in which the wrist did not achieve the dwell state within the allotted 2-s simulation time, that episode was counted as a failure. We did not include failures in statistical calculations for this metric, because failed episodes did not have an associated time required to achieve the dwell state. For the set of successful episodes, the times required to achieve the dwell state were averaged.

4) Speed Change: For each set of 500 reaching movement episodes, the first 100 and the final 100 episodes were evaluated. The mean time to achieve dwell state for the successful episodes within each set of 100 episodes was calculated. The difference $(\mathrm{TimeToAchieveDwell}_{\mathrm{Initial}} - \mathrm{TimeToAchieveDwell}_{\mathrm{Final}})$, in ms, is the speed change over 500 episodes. Positive speed change values indicate that the mean time to achieve dwell state has decreased over the 500-episode period.

E. Actor and Critic Learning Parameter Tuning

Learning parameters (e.g., step sizes, exploration rates, etc.) for the actor and critic components of the RL controller (see Algorithm 1) were explored to determine which values would perform well for a wide range of training conditions. Values of η_{A} and η_{C} were systematically varied. A pair of learning rates that was found to have good performance over all training sets tested, $\eta_{\mathrm{A}} = 150.0$ and $\eta_C = 1.0$, was selected for use for the remaining simulation work, unless otherwise indicated.

F. Simulation Experiments

1) Experiment 1: Effect of Pseudo-Human Reward Type: To examine how the type and spacing of rewards affects RL controller learning, a variety of pseudo-human rewards assigned at different frequencies were investigated. The reward condition that used the most frequently-assigned rewards, *cumulative rewards*, involved a reward being assigned at each timestep that the wrist was in a specified target zone. In contrast, the *final-timestep rewards* condition assigned a pseudo-human reward only once per episode, at the final timestep, based on the status of the wrist position at this point in time. Algorithm 2 describes how the final-timestep pseudo-human rewards were assigned, using a combination of success at remaining within the target zone and extent of target overshoot. The use of these final-timestep rewards alone was investigated, as well as their combination with the automated rewards calculated at each timestep. Experiments using only automated rewards and using the optimized PD controller were also performed. Fig. 2 illustrates the process by which pseudo-human rewards were added to the automated rewards, if present, with the sum (total reward) being used to update the RL controller.

A set of ten interleaved, unique, randomly generated dynamic movement tasks was used to train the RL controller over 500 episodes per run: the RL controller encountered each of the ten unique movements 50 times during each run. Ten 500-episode runs were performed for each reward condition. The average linear distance between the initial hand position and the target hand position of each movement was 31 ± 11 cm. Kruskal–Wallis analysis of variance (ANOVA) analysis with multiple comparison was performed on the dwell-at-target success and time to achieve dwell state metrics.

2) Experiment 2: Effect of Reward Spacing: This experiment was performed to examine the effect of introducing delays to the cumulative pseudo-human rewards. Reward spacing intervals specify the distance, in timesteps, between consecutive cumulative rewards. Reward spacing intervals of 1–10, 15, 20, 25, 30, 35, 50, 75, and 100 timesteps were tested. (Recall that each step is 20 ms, so an interval of 100 timesteps corresponds to a reward every 2 s, i.e., a single reward at the end of each episode). The training task set consisted of 500 episodes of a single large movement that required extensive flexor activation. The linear distance between the initial and target hand positions of the tested movement was 79 cm. Multiple 500-episode runs were performed for reward spacing intervals of 1 and 2, with a single 500-episode run collected for the larger reward spacing intervals. A fixed cumulative pseudo-human reward magnitude of 0.1 was used. Dwell-at-target success and net learning metrics were evaluated, and lines were fit to the resulting datasets.

3) Experiment 3: Effect of Training Task Set Size: This experiment investigated the effect on controller learning of the number of unique tasks in the training task set, for the purpose of informing future experimental design. Learning rates of $\eta_A = 70.0$, $\eta_C = 0.1$ were used for these experiments, as these values were found to permit a wide range of task set sizes to be explored effectively. The number of tasks in the training set was varied from 10 to 500, with set sizes all being factors of 500. Each set of unique tasks was randomly generated, with each set being a subset of the next-larger set. The mean linear distance between initial and target positions for the 500-task set was 31 ± 17 cm. Five 500-episode runs of each case were simulated. Kruskal–Wallis ANOVA analysis was performed for the dwell-at-target success over the full set of 500 episodes and over the final 100 episodes, net learning, and speed change metrics.

4) Experiment 4: Long-Term Learning Using Pseudo-Human Rewards: To investigate whether the RL controller is able to continue learning beyond 500 episodes of training, and to determine the stability of its long-term performance, sets of 4000 episodes were simulated for two conditions: a baseline condition using only automated rewards, and an experimental condition using final-timestep pseudo-human rewards in addition to automated rewards. For both conditions, ten 4000-episode runs were performed on a task set consisting of 50 unique tasks with a mean linear distance between initial and target positions of 32 ± 16 cm. T-test analysis was performed for dwell-at-target success and time to achieve dwell state metrics over the final 100 episodes.

III. Results

A. Experiment 1: Effect of Pseudo-Human Reward Type

Fig. 3 summarizes the relative performance of RL controllers trained using a variety of reward types. Each reward condition was used to train ten RL controllers over a set of 500 reaching movement episodes. Both panels of this figure plot the mean value averaged over each subset of 100 episodes; each plotted point represents the mean value over ten training runs. Error bars show the mean standard deviation averaged over ten training runs (each training run has an associated standard deviation over each subset of 100 runs). Fig. 3(a) shows the dwell-in-target-zone success percentages (out of 500 episodes per run). Fig. 3(b) presents the mean time required to achieve the dwell state.

For the dwell-at-target success metric in Fig. 3(a), all RL controllers visibly outperform the baseline optimized PD

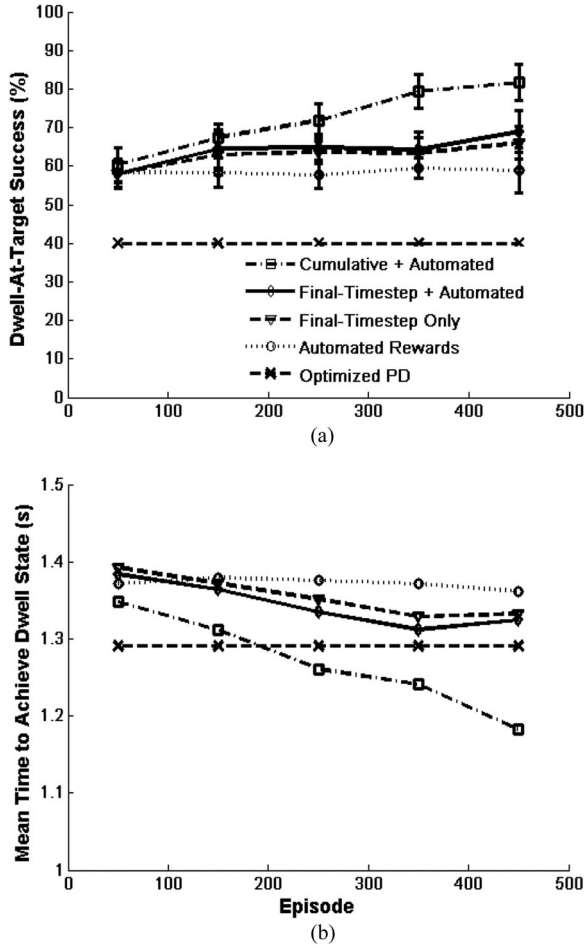

Fig. 3. Performance metrics for controllers trained with various forms of reward on 500 episodes of a training task set containing ten unique tasks. Plotted points show the average over ten runs. (a) Dwell-at-target success percentages. (b) Mean time to achieve the dwell state. Error bars indicate averaged standard deviations over ten runs.

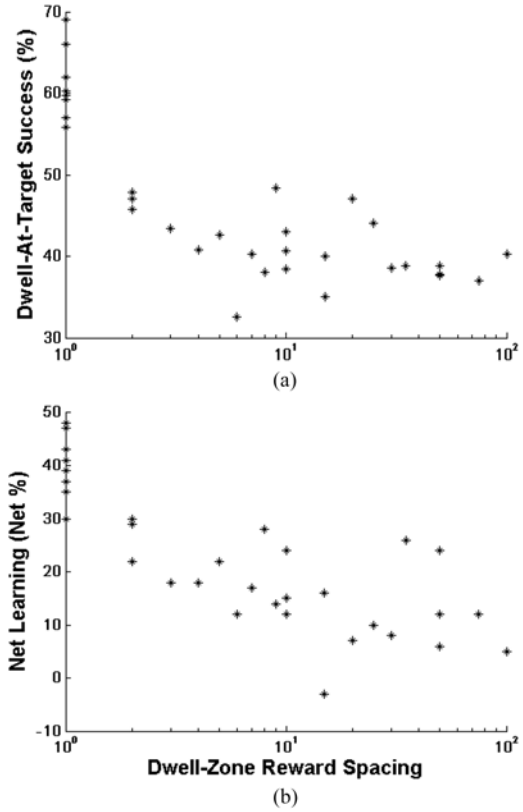

Fig. 4. Effect of reward spacing on actor-critic RL controller learning performing dynamic reaching movements using a planar biomechanical arm model. (a) Dwell-at-target success (%) measures how frequently the hand reaches the target zone and remains there continuously for at least 100 ms; presence in the target zone at the final timestep is also required in order to be counted as success. The model $21.34x^{-1.618} + 39.53$ fits the data with $R^2 = 0.877$, adjusted $R^2 = 0.8691$, SSE = 409.2, and RMSE = 3.633. Note that the *Y*-axis ranges from 0% to 100%. (b) Net learning measures the final – initial dwell-at-target success percentages over each 500-episode training session. The model $30.42x^{-0.9315} + 11.12$ fits the data with $R^2 = 0.7792$, adjusted $R^2 = 0.7649$, SSE = 1465, and RMSE = 6.875.

controller. The automated rewards condition is shown in Fig. 3 as the dotted trendline. In Fig. 3(a), this baseline condition shows no improvement over 500 episodes. Using final-timestep pseudo-human rewards (dashed trendline with triangle symbols) improves upon the baseline automated performance, while combining the final-timestep rewards with automated rewards at each timestep (solid trendline with oval symbols) marginally increases success to reach the target. The controller incorporating cumulative rewards (dashdot trendline with square symbols) shows substantial improvement over the series of 500 episodes.

For the dwell-at-target success metric in Fig. 3(a), significant differences were seen between the following reward condition pairings for data over the final 100 episodes, using Kruskal–Wallis nonparametric ANOVA with multiple comparison ($H = 28.36$, d.f. 3, $p < 0.001$): automated rewards had significantly less success than the combination of pseudo-human and automated rewards ($p = 0.0417$), as well as when compared with cumulative rewards ($p < 0.001$); and final-timestep rewards used alone had significantly less success at achieving the dwell state than cumulative rewards ($p = 0.0017$). All other pairwise comparisons were nonsignificant.

Fig. 3(b) shows the time required to achieve the dwell state. The cumulative rewards case (dashdot trendline with square symbols) was the only one that outperformed (i.e., had smaller values than) the baseline PD case shown as the horizontal dashed trendline with X symbols, for the final set of 100 episodes.

For the performance metric of time required to achieve dwell status in Fig. 3(b), Kruskal–Wallis ANOVA with multiple comparison for the final 100 episodes showed that the cumulative rewards case required significantly less time to achieve the dwell state than the other three reward conditions ($H = 24.56$, d.f. 3, $p < 0.001$; compared with automated rewards, $p < 0.001$; compared with final-timestep-only rewards, $p = 0.0022$; and compared with final-timestep combined with automated rewards, $p = 0.0077$). No other significant comparisons were found.

B. Experiment 2: Effect of Reward Spacing

Fig. 4 displays the effect that the spacing of rewards of constant magnitude has on RL controller learning. Fig. 4(a) shows

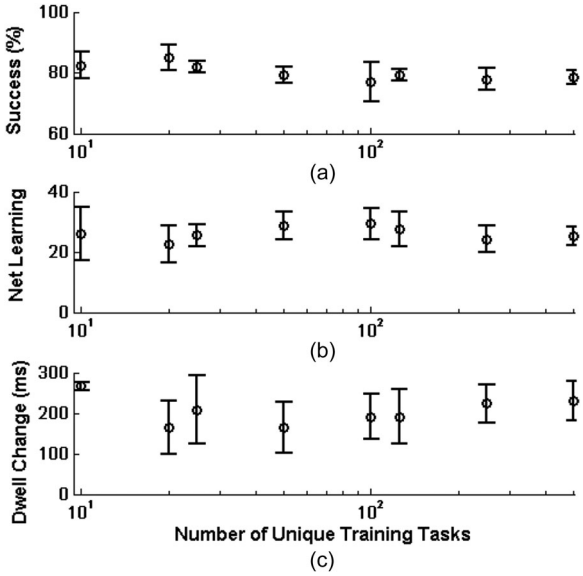

Fig. 5. Learning metrics as a function of the number of unique tasks in each 500-episode training set. Error bars show standard deviations over five runs per condition. (a) Dwell-at-target success (%) measures how often the hand was able to reach the target zone and dwell for at least 100 ms, and presence in the target zone at the final timestep was also required to be counted as success. (b) Net learning is defined as the final–initial dwell-at-target success percentages, and measures how much learning is achieved over each 500-episode training set. (c) Dwell change (in ms) is the difference between the time required for the final 100 episodes of each training set to reach the target, minus the time required for the initial 100 episodes of each training set to reach the target.

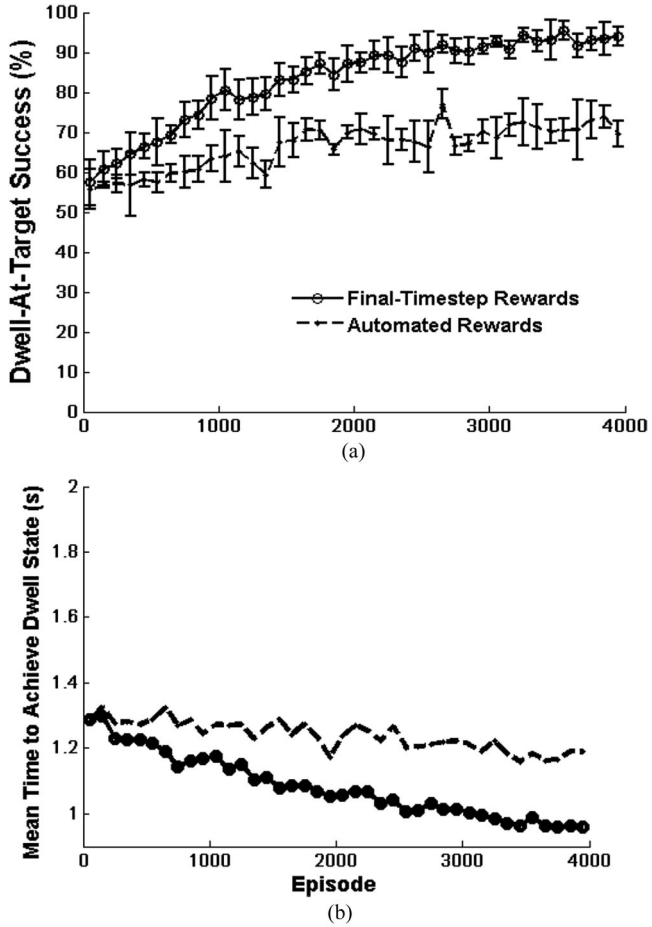

Fig. 6. Long-term performance of RL controllers trained using automated rewards or final-timestep pseudo-human rewards. Mean values are calculated over each set of 100 episodes. Error bars show mean standard deviations over ten runs. (a) Dwell-at-target success percentage. (b) Mean time required to achieve the dwell state.

the percentage of dwell-in-target-zone success over 500 training episodes, and Fig. 4(b) shows the net learning that occurred over each 500-episode run. The highest success and highest rates of learning correspond to the reward spacing of 1, i.e., a computer-generated "cumulative" reward being assigned at every simulation timestep. For the dwell-in-target-zone success metric, behavior is similar (i.e., relatively flat) among all reward spacing values larger than 1. In contrast, the net learning plot shows that learning tends to taper off gradually as the distance between rewards increases.

C. Experiment 3: Effect of Training Task Set Size

We examine the effect that the number of unique training tasks has on RL controller learning in Fig. 5. Here, three different learning metrics are presented as functions of the number of unique tasks in the 500-episode training set. Fig. 5(a) shows the dwell-in-target-zone success percentage, Fig. 5(b) presents the net learning that occurs over each 500-episode run, and Fig. 5(c) displays the amount of change in the mean time required to achieve the dwell state, in ms, over each 500-episode training set. For all three plots, no significant trends are present; varying the number of unique tasks in the training set does not appear to affect learning in a meaningful way.

Fig. 5(c) shows that, for the time-to-achieve-dwell-state metric, the case with the fewest unique tasks (10) has a notably smaller standard deviation than the other cases. However, Kruskal–Wallis ANOVA analysis showed no significant differences ($p > 0.05$) in mean ranks among task sets for dwell-at-target success measured over all 500 episodes ($H = 13.52$, $\mathrm{d.f.}\ 7$) and over the final 100 episodes ($H = 12.3$, $\mathrm{d.f.}\ 7$), for the net learning metric ($H = 7.28$, $\mathrm{d.f.}\ 7$), or for the speed change metric ($H = 11.21$, $\mathrm{d.f.}\ 7$).

D. Experiment 4: Long-Term Learning Using Pseudo-Human Rewards

Fig. 6 presents the long-term learning properties when final-timestep pseudo-human rewards (solid trendline with circle symbols) are used and compares this performance against the use of benchmark automated rewards (dashed trendline). For each training condition, ten 4000-episode RL controller training runs were performed. Each plotted point represents the average value over ten training runs, and subsets of 100 episodes are used. Error bars represent the mean standard deviation calculated over ten runs.

Fig. 6(a) shows the success percentage to achieve the dwell state. Each plotted point represents the mean success percentage averaged over ten training runs, and calculated over each subset of 100 training episodes. Over the final 100 collected episodes, t-test analysis shows that the final-timestep pseudo-human

rewards case $(94.0 \pm 2.31\%)$ results in significantly better performance than the automated rewards case $(69.6 \pm 3.31\%)$; $t(18) = -19.13,\ p < 0.001$.

The time required to achieve the dwell state is presented in Fig. 6(b). T-test analysis of the time-to-dwell values for the final 100 training episodes shows significantly faster movement times when final-timestep pseudo-human rewards are used $(0.96 \pm 0.38\ \text{s})$ compared with automated rewards only $(1.19 \pm 0.40\ \text{s})$; $t(18) = 16.97,\ p < 0.001$.

IV. Discussion

We found that an actor-critic RL controller for human arm movement learned significantly faster when computer-generated sparse, delayed, human-like rewards were given in addition to automated rewards resembling a conventional cost function for optimal control. Furthermore, the type of human-like reward was important. Cumulative rewards, given at every timestep, resulted in significantly faster learning than did final-timestep rewards, which were assigned only once per reaching movement (or episode) (see Algorithm 2). Fig. 4 shows a steep decline in learning as reward frequency is decreased from every-timestep to every-other-timestep, and smaller declines in learning occur as the rewards become increasingly sparse. Nevertheless, even when only a single pseudo-human reward is assigned per episode, using this form of reward in addition to automated rewards presented significant advantages, in comparison with using automated rewards only (see Fig. 6). These results suggest that when human rewards are substituted for pseudo-human rewards in future experiments, the sparse, delayed rewards generated by humans are likely to be useful when training the actor-critic RL controller for planar arm movement.

The number of unique tasks in the training set was varied in Experiment 3 to investigate whether the diversity of tasks influences RL controller learning, in order to determine the appropriate diversity of movements that should be used in future experiments with human subjects. Controllers trained using the ten-unique-task set encountered each task 50 times per 500-episode set, while those trained using the 500-unique-task set trained on only a single instance of each task per training set. For this cumulative rewards training experiment, we found no significant differences among the various training task sets for all dwell-at-target success metrics, including net learning, or for the speed change metric (see Fig. 5). The ability of the actor-critic RL architecture to learn equally well from small or diverse sets of tasks suggests that it is well suited to a real-world setting in which an FES user will need to train his controller on a constantly-changing set of movements.

In Experiment 4 (see Fig. 6), we investigated the long-term properties of RL controller learning when trained using only automated rewards, and when final-timestep pseudo-human rewards were added. Both training conditions showed stable long-term learning properties. The final-timestep rewards case significantly outperformed the automated rewards-only training for both performance metrics, dwell-at-target success and time to achieve dwell state. As the length of training increases, the advantage afforded by the final-timestep rewards becomes increasingly significant. In general, the hand trajectories of controllers trained using the final-timestep pseudo-human rewards were more linear and showed less overshoot than controllers trained using only automated rewards. Hand velocity profiles showed that controllers trained with pseudo-human rewards reached their targets more quickly than controllers trained with automated rewards. Muscle activations for both cases were generally smooth and continuous.

Consistent with our predictions, we found that cumulative rewards resulted in faster learning than final-timestep pseudo-human rewards; the frequency of rewards is recognized to be positively correlated with learning speed [36], [37]. Other researchers have successfully emphasized or relied exclusively upon final-timestep rewards to train RL controllers (e.g., [38]). Extending that earlier final-timestep learning research, it was encouraging to find that final-timestep rewards could result in useful learning for our continuous-time, continuous-state system, because the challenges of RL controller learning using sparse rewards are exacerbated in continuous spaces [36]; there was no *a priori* guarantee that using a single reward per reaching movement episode would result in significant learning on a measurable timescale.

These experiments were simulations performed on a biomechanical model with nonlinear and coupled, yet deterministic and known, properties. In real-world upper-extremity FES systems, each individual's shoulder and arm will be characterized by a number of unknown properties, ranging from musculotendon characteristics to unpredictable episodes of spasticity, as well as muscular fatigue from exertion, and strengthening due to training. The RL controllers trained in these experiments performed well when the arm model's flexor muscles were weakened by a static 50% of their maximal strength, but it is not clear whether the controllers would learn as rapidly or as effectively if the arm properties had varied over the course of training.

The final-timestep pseudo-human rewards were specified by Algorithm 2. While this algorithm was developed to ensure that all five reward levels were represented during RL controller training sessions, we make no claims that human reward-givers would use this or a similar mental model to assign rewards. For this reason, while we can tentatively conclude that using a single reward at the end of each episode can effectively improve RL controller learning, we cannot guarantee that substituting human rewards for the rewards generated by our five-reward-level final-timestep pseudo-human algorithm will exhibit similar controller learning properties.

When implementing an RL controller in a clinical setting with human users, the controller should be pretrained using pseudo-human rewards to achieve a reasonable baseline level of performance upon which the human user could train the controller to improve. This would eliminate the human user experiencing the lengthy preliminary controller training period, as the performance of the controller at this early stage would be impractical for generating useful movements. Additionally, characteristics of movements that are known to be desirable will permit the assignment of computer-generated rewards to augment the human-generated reward signal; this will offload as much effort as possible from the human to the control algorithm,

leaving the more subtle and personalized aspects of movement for the human to rate and shape. The human user could train the controller over the course of daily use of the RL-controlled neuroprosthesis, with the controller's learning accumulating over long periods.

We have demonstrated that sparse, delayed pseudo-human final-timestep rewards can result in significant and consistent RL controller learning gains [see Figs. 3(a) and 6] when compared with the use of automated rewards only. Our findings are informative because, for the planar biomechanical arm model with six muscles being controlled, it was not certain whether sparse, delayed rewards would permit the actor-critic RL controller to learn effectively over a measurable timespan. These results suggest that real human-generated rewards may also be effective for actor-critic RL controller training. The next experimental step will be to introduce real human-generated rewards to train this actor-critic RL controller for planar arm movement and to determine how these rewards perform to train the RL controller, compared with controller learning using pseudo-human rewards.

V. Conclusion

Our results have demonstrated that it is possible to effectively train an actor-critic RL controller to perform goal-oriented reaching movements on a simulated planar human biomechanical arm using sparse, delayed pseudo-human rewards that are assigned only once per reaching task, as a high-level assessment of the quality of the movement. The diversity of the training task set did not significantly impact learning, suggesting that the controller will be able to learn well in a real-world setting involving diverse, continuously changing reaching movements. The long-term stability of controllers trained using final-timestep rewards was observed. Future work will substitute real human-generated rewards for the pseudo-human rewards used in these experiments and will examine the RL controller learning properties under these training conditions, with the ultimate goal of incorporating human-generated input to restore customized, natural arm movements to individuals paralyzed by SCI.

Acknowledgment

The authors would like to acknowledge Dr. S. Sidik for his statistics guidance.

References

[1] K. T. Ragnarsson, "Functional electrical stimulation after spinal cord injury: Current use, therapeutic effects and future directions," *Spinal Cord*, vol. 46, pp. 255–274, 2008.

[2] V. Dietz and K. Fouad, "Restoration of sensorimotor functions after spinal cord injury," *Brain*, vol. 137, no. 3, pp. 654–667, 2014.

[3] P. H. Peckham and J. S. Knutson, "Functional Electrical Stimulation for neuromuscular applications," *Annu. Rev. Biomed. Eng.*, vol. 7, pp. 327–360, 2005.

[4] K. S. Wuolle, A. M. Bryden, P. H. Peckham, P. K. Murray, and M. Keith, "Satisfaction with upper-extremity surgery in individuals with tetraplegia," *Arch. Phys. Med. Rehabil.*, vol. 84, no. 8, pp. 1145–1149, Aug. 2003.

[5] J. J. Abbas and R. J. Triolo, "Experimental evaluation of an adaptive feedforward controller for use in functional neuromuscular stimulation systems," *IEEE Trans. Rehabil. Eng.*, vol. 5, no. 1, pp. 12–22, Mar. 1997.

[6] P. E. Crago, N. Lan, P. H. Veltink, J. J. Abbas, and C. Kantor, "New control strategies for neuroprosthetic systems," *J. Rehabil. Res. Dev.*, vol. 33, no. 2, pp. 158–172, 1996.

[7] D. Blana, "Feedback control of a high level upper extremity neuroprosthesis," Ph.D. dissertation, Dept. Biomed. Eng., Case Western Reserve Univ., Cleveland, OH, USA, 2008.

[8] S. Mularski, T. Picht, B. Kuehn, T. Kombos, M. Brock, and O. Suess, "Real-time tracking of vertebral body movement with implantable reference microsensors," *Comput. Aided Surg.*, vol. 11, no. 3, pp. 137–146, May 2006.

[9] J. M. Lambrecht and R. F. Kirsch, "Miniature low-power inertial sensors: promising technology for implantable motion capture systems," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 6, pp. 1138–1147, Nov. 2014.

[10] B. Bolsterlee, D. H. Veeger, and E. K. Chadwick, "Clinical applications of musculoskeletal modeling for the shoulder and upper limb," *Med. Biol. Eng. Comput.*, vol. 51, no. 9, pp. 953–963, 2013.

[11] W. B. Knox, C. Breazeal, and P. Stone, "Learning from feedback on actions past and intended," presented at the 7th ACM/IEEE Int. Conf. Human-Robot Interaction, Late-Breaking Rep. Session, Boston, MA, USA, 2012.

[12] C. Orosy-Fildes and R. W. Allan, "Psychology of computer use: XII. Videogame play: Human reaction time to visual stimuli," *Perceptual Motor Skills*, vol. 69, pp. 243–247, 1989.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[14] M. J. Mataric, "Reinforcement learning in the multi-robot domain," *Auton. Robots*, vol. 4, pp. 73–83, 1997.

[15] P. S. Thomas, "A reinforcement learning controller for functional electrical stimulation of a human arm," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Case Western Reserve Univ., Cleveland, OH, USA, 2009.

[16] A. L. Thomaz, "Socially guided machine learning," Ph.D. dissertation, School Comput. Sci., Univ. Massachusetts, Amherst, Amherst, MA, USA, 2006.

[17] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Proc. 20th Int. Symp. Robot Human Interactive Commun.*, Jul. 2011, pp. 1–6.

[18] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *Proc. 5th Int. Conf. Knowl. Capture*, 2009, pp. 9–16.

[19] H. van Hasselt, "Insights in reinforcement learning: formal analysis and empirical evaluation of temporal- difference learning algorithms," Ph.D. dissertation, Dutch Res. School Inf. Knowl. Syst., Enschede, Netherlands, 2011.

[20] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 8, pp. 1017–1023, 1996.

[21] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *Proc. IEEE Int. Conf. Rehabil. Robot.*, Zurich, Switzerland, Jun. 2011, pp. 134–140.

[22] J. Izawa, T. Kondo, and K. Ito, "Biological arm motion through reinforcement learning," *Biol. Cybern.*, vol. 91, pp. 10–22, 2004.

[23] P. Thomas, M. Branicky, A. van den Bogert, and K. Jagodnik, "Application of the actor-critic architecture to functional electrical stimulation control of a human arm," in *Proc. 21st Innovative Appl. Artif. Intell. Conf.*, 2009, pp. 165–172.

[24] N. Lan, "Analysis of an optimal control model of multi- joint arm movements," *Biol. Cybern.*, vol. 76, pp. 107–117, 1997.

[25] K. M. Jagodnik and A. J. van den Bogert, "Optimization and evaluation of a proportional derivative controller for planar arm movement," *J. Biomech.*, vol. 43, pp. 1086–1091, 2010.

[26] A. V. Hill, "The heat of shortening and the dynamic constants of muscle," *Proc. Roy. Soc. London Ser. B*, vol. 126, pp. 136–195, 1938.

[27] J. M. Winters, "Hill-based muscle models: A systems engineering perspective," in *Multiple Muscle Systems*. New York, NY, USA: Springer, 1990, pp. 69–93.

[28] S. G. McLean, A. Su, and A. J. van den Bogert, "Development and validation of a 3-D model to predict knee joint loading during dynamic movement," *Trans. ASME.*, vol. 125, pp. 864–874, Dec. 2003.

[29] D. A. Winter *Biomechanics and Motor Control of Human Movement*, 3rd ed. Hoboken, NJ, USA: Wiley, 2005, pp. 59–85.

[30] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[31] S. J. Bradtke and M. O. Duff, "Reinforcement learning methods for continuous-time Markov decision problems," *Adv. Neural Inf. Process Syst.*, vol. 7, no. 7, p. 393, 1995.

[32] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable Markov decision problems," *Adv. Neural Inf. Process. Syst.*, vol. 7, no. 7, pp. 345–52, 1995.
[33] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, vol. 414. Hoboken, NJ, USA: Wiley, 2005.
[34] M. P. Woodward, "Framing human-robot task communication as a partially observable Markov decision process," Ph.D. dissertation, School Eng. Appl. Sci., Harvard Univ., Cambridge, MA, USA, 2012.
[35] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput*., vol. 12, pp. 219–245, 2000.
[36] W. D. Smart, "Making reinforcement learning work on real robots," Ph.D. dissertation, Dept. Comput. Sci., Brown Univ., Providence, RI, USA, 2002.
[37] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proc. 20th Int. Conf. Mach. Learning*, 2003, pp. 792–799.
[38] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 4, pp. 3404–3410.
[39] C. K. Thomas, E. Y. Zaidner, B. Calancie, J. G. Broton, and B. R. Bigland-Ritchie, "Muscle weakness, paralysis, and atrophy after human cervical spinal cord injury," *Exp. Neurol.*, vol. 148, no. 2, pp. 414–423, 1997.
[40] H. Kern, C. Hofer, M. Mödlin, W. Mayr, V. Vindigni, S. Zampieri, S. Boncompagni, F. Protasi, and U. Carraro, "Stable muscle atrophy in long-term paraplegics with complete upper motor neuron lesion from 3-to 20-year SCI," *Spinal Cord*, vol. 46, no. 4, pp. 293–304, 2007.

Kathleen M. Jagodnik (M'06) received the B.S. degree in computer science, chemistry, philosophy, and classics from John Carroll University, University Heights, OH, USA, in 2002, and the M.S. and Ph.D. degrees in biomedical engineering from Case Western Reserve University, Cleveland, OH, in 2007 and 2014, respectively.

She is currently a National Space Biomedical Research Institute Postdoctoral Research Fellow with NASA Glenn Research Center, Cleveland. Her research interests include biomechanics, the effects of microgravity on exercise performance, rehabilitation engineering, and artificial intelligence.

Philip S. Thomas received the B.S. and M.S. degrees in computer science from Case Western Reserve University, Cleveland, OH, USA, in 2008 and 2009, respectively, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, MA, USA, in 2015.

He is currently a Postdoctoral Fellow with the laboratory of Dr. Emma Brunskill in the Computer Science Department (affiliate Machine Learning Department), Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include machine learning, with an emphasis on policy search algorithms.

Antonie J. van den Bogert received the Ph.D. degree from the University of Utrecht, Utrecht, Netherlands.

He is currently the Parker-Hannifin Endowed Chair in Human Motion and Control with Cleveland State University, Cleveland, OH, USA. His research interests include musculoskeletal modeling and simulation, optimal control of human movement, and advanced prosthetics and orthotics.

Michael S. Branicky (M'87–SM'01–F'16) received the Sc.D. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA.

He is currently the Dean of the School Engineering and a Professor with the Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA. His research interests include control systems, robotics, hybrid systems, intelligent control, and learning.

Robert F. Kirsch (M'82) received the Ph.D. degree from Northwestern University, Evanston, IL, USA.

He is currently a Professor and Chair of the Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH, USA, and the Executive Director for the Cleveland VA FES Center. His research interests include restoring movement to disabled individuals using functional electrical stimulation (FES) and controlling FES actions via natural neural commands.