Motion Learning and Rapid Generalization for Musculoskeletal Systems Based on Recurrent Neural Network Modulated by Initial States

Xiaona Wang, Jiahao Chen, *Member, IEEE*, and Hong Qiao, *Fellow, IEEE*

***Abstract*—Musculoskeletal robot with high precision and robustness is a promising direction for the next generation of robots. However, motion learning and rapid generalization of complex musculoskeletal systems are still challenging. Therefore, inspired by the movement preparation mechanism of the motor cortex, this article proposes a motion learning framework based on the recurrent neural network (RNN) modulated by initial states. First, two RNNs are introduced as a preparation network and an execution network to generate initial states of the execution network and time-varying motor commands of movement, respectively. The preparation network is trained by a reward-modulated learning rule, and the execution network is fixed. With the modulation of initial states, initial states can be explicitly expressed as knowledge of movements. By dividing the preparation and execution of movements into two RNNs, the motion learning is accelerated to converge under the application of the node-perturbation method. Second, with the utilization of learned initial states, a rapid generalization method for new movement targets is proposed. Initial states of unlearned movements can be computed by searching for low-dimensional ones in latent space constructed by learned initial states and then transforming them into the whole neural space. The proposed framework is verified in simulation with a musculoskeletal model. The results indicate that the proposed motion learning framework can realize goal-oriented movements of the musculoskeletal system with high precision and significantly improve the generalization efficiency for new movements.**

***Index Terms*—Biologically inspired control, motor cortex, movement preparation, musculoskeletal system, recurrent neural network (RNN).**

Manuscript received 23 August 2021; revised 16 November 2021; accepted 11 December 2021. Date of publication 21 December 2021; date of current version 9 December 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1300203; in part by the National Natural Science Foundation of China under Grant 91648205, Grant 61627808, and Grant 91948303; and in part by the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32050100. *(Corresponding authors: Jiahao Chen; Hong Qiao.)*

Xiaona Wang and Jiahao Chen are with the State Key Laboratory of Management and Control for Complex Systems and Beijing Key Laboratory of Research and Application for Robotic Intelligence of "Hand-Eye-Brain" Interaction, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: jiahao.chen@ia.ac.cn).

Hong Qiao is with the State Key Laboratory of Management and Control for Complex Systems and Beijing Key Laboratory of Research and Application for Robotic Intelligence of "Hand-Eye-Brain" Interaction, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China (e-mail: hong.qiao@ia.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCDS.2021.3136854.

Digital Object Identifier 10.1109/TCDS.2021.3136854

I. Introduction

MANY application scenarios require robots to achieve high-precision and dexterous movements in fluctuating and unstructured environments. The human-like musculoskeletal robot is a promising way to satisfy these requirements and has received extensive attention. Compared with traditional rigid robots, the biological musculoskeletal system has obvious advantages. First, due to the redundancy of joints and muscles, it can achieve movements with high flexibility and robustness, even if a part of the actuators is fatigued or dysfunctional. Second, the stiffness can be modulated by coordinating the activation of agonist and antagonist muscles to adapt to different environments [1], [2]. Therefore, in order to demonstrate similar advantages, many musculoskeletal robots are designed with the imitation of the human musculoskeletal system in terms of muscular arrangement and driving mode [3]–[7]. Typical examples are the robots "Kengoro" built by the University of Tokyo and "ECCEROBOT" funded by the European Union's Human Brain Project, these robots with human-like features, such as compliant, tendon-driven actuators, and complex joints exhibit high anatomical fidelity to the human musculoskeletal structure [7], [8]. In addition, some prosthetic limbs and exoskeleton robots inspired by the musculoskeletal system have also been developed, they have similar design principles and control strategies to musculoskeletal robots [9]–[12]. Dabiri *et al.* [9] have built an artificial prosthetic limb with antagonist artificial muscle structure, and it is driven by one kind of McKibben pneumatic muscle named Festo artificial muscle. Chen *et al.* [10] have implemented a 4-degree-of-freedom upper limb exoskeleton robot, which is actuated by pneumatic muscle actuators via steel cables.

Although the structure and driving mode of musculoskeletal robots offer some benefits, they also bring some challenges to motion control. On the one hand, owing to the interaction between muscles, such as entanglement or friction [13] and the coupling among muscles and joints [14], it is difficult to compute the control signal of each muscle independently. On the other hand, high nonlinearity of muscle dynamics and complex muscular arrangement make it difficult to acquire an explicit mathematic representation of the robot. To solve the motion

control problem of musculoskeletal robots, many model-based methods and model-free methods have been proposed.

In model-based methods, a mathematical relationship between the muscle space and task space is required to be established. Based on this relationship, various controllers are designed to calculate muscle activation or force to achieve the given task. Particularly, static or dynamic optimization [15], computed muscle control [16], inverse-model-based method [14], iterative learning control [17], [18], and adaptive-optimal-critic-based neuro-fuzzy control [19] are applied in the model-based methods. However, for sophisticated robots with intertwined muscular arrangement or highly nonlinear muscles, the requirement of building accurate models is difficult to fulfill, so model-based methods are not suitable for them. These methods are mainly used in the musculoskeletal robots with relatively simple structures or the simulated musculoskeletal systems.

In contrast to model-based methods, the model-free methods do not require establishing the explicit mathematical model. With the development of machine learning technology [20], some model-free methods involving state-of-the-art deep reinforcement learning (DRL) algorithms are proposed. They learn the mapping from motion targets to muscle excitations by training deep neural networks or recurrent neural networks (RNNs) through actor–critic [21], proximal policy optimization [22], and so on [23]–[25]. Specifically, the core idea of them is to evaluate the motion performance under current strategy by a designed reward function, and continuously update the control strategy through trial-and-error learning until the intended motion is achieved. Besides, the model-free methods based on muscle synergy [26] are proposed, in which the activation pattern of each muscle is defined as linear superposition of several synergies, which are modulated independently in amplitude and time [27]–[30]. Chen and Qiao [28] adopted the evolutionary algorithm to optimize the parameter related to synergies based on movement errors, which could generate appropriate time-varying muscle excitations to drive the musculoskeletal system to achieve movements on the sagittal plane. In addition, an iterative learning controller has been designed for modulating muscle synergies to make a musculoskeletal system to realize human-like manipulation [29]. However, these model-free methods need extensive trials to update the control strategy or synergy parameters and take a considerable training time. Additionally, these methods pay little attention to motion generalization for new motion targets. Although the synergy-based method [28] attempts to promote generalization ability for new motion targets, it can only be generalized to similar targets.

The research in neuroscience suggests that motor cortex plays a key role in motion learning and control of biological musculoskeletal systems. First, it is found that motor cortex strongly affects muscle contraction and relaxation with direct projection to interneurons and motorneurons in the spinal cord [31]. Furthermore, Churchland *et al.* [32]–[34] have proposed a dynamical system hypothesis to explain how neural population responses generate and affect the movements. They assume that the neuronal activities in motor cortex follow a general dynamical system, which can produce a sequence of temporal and spatial patterns as motor commands to the downstream circuit. Moreover, neuroscientists have found sufficient evidence to support a process of movement preparation prior to the movement execution process in motor cortex [35]–[37]. An interpretation about movement preparation is the optimal subspace hypothesis proposed by Churchland *et al.* [36], [37]. Specifically, the result of preparatory activity during movement preparation could act as the initial state of a dynamical system, which evolves in the motor cortex during movement execution. To reach each intended movement target accurately, the neurons need to reach a corresponding optimal subspace of initial states when movement preparation is over, which is a subset of appropriate initial neural population firing rates of the movement execution process. In other words, the network dynamics that is responsible for movement execution must be seeded with a suitable initial state before the onset of movement, which determines subsequent evolution of both activities of neurons and actual behavior to a large extent [38]. Multiple RNNs have been proposed to simulate complex cortical dynamics of monkeys or humans during movements [39], [40].

In this article, inspired by the dynamical system hypothesis on the motor cortex and optimal subspace hypothesis on movement preparation, a motion learning framework based on RNN modulated by initial states is proposed to achieve the motion learning and rapid generalization of a sophisticated musculoskeletal robotic system. The main task is to control the musculoskeletal system to reach random targets within the workspace by motion learning, and to reach other new targets in the same environment by generalization. Specific contributions are summarized as follows.

1) First, the proposed framework adopts two RNNs as a preparation network and a execution network to generate initial states of the execution network and time-varying motor commands of movements, respectively. During motion learning, only the weights of the preparation network are optimized by a reward-modulated learning rule and the ones of the execution network are fixed. By dividing the preparation and execution of movements into two RNNs, initial states can be explicitly expressed as knowledge of movements, and the motion learning is accelerated to converge under the application of the node-perturbation method.
2) Second, with the utilization of learned initial states, a rapid generalization method for new movement targets is proposed. Initial states of unlearned movements can be computed by searching for low-dimensional ones in latent space constructed by learned initial states and then transforming them into the whole neural space. Based on the method, the rapid generalization for new movement targets could be performed.

In the experiments, the effectiveness of the proposed framework is verified with a highly redundant and coupled musculoskeletal system. The results indicate that the proposed motion learning framework can realize goal-oriented movements with high precision and significantly improve the generalization efficiency for new movements.

The remainder of this article is organized as follows. Section II introduces the dynamics of musculoskeletal robotic system. Section III illustrates the proposed motion learning framework in detail. Section IV describes the experimental setup and results. Sections V and VI present the discussion and conclusion, respectively.

II. Musculoskeletal Model

In the musculoskeletal system, muscles are attached to skeletons and exert forces to drive the skeletons to achieve motion. As for a single muscle, it is composed of tendon and muscle fiber. The activation dynamics of muscle fiber can be expressed as follows according to the classical Hill-type model [16], [41]:

$$\frac{da}{dt} = \begin{cases} \frac{u-a}{\tau_{\text{act}}(0.5+1.5a)} & u > a \\ \frac{u-a}{\tau_{\text{deact}}/(0.5+1.5a)} & u \le a \end{cases} \tag{1}$$

where scalar u which ranges between 0 and 1 is the input of each muscle fiber and it is called muscle excitation signal, a is the muscle activation signal, and τ_{act} and τ_{deact} are time constants related to activation and deactivation, respectively.

After activation, muscle fiber will contract to produce active and passive force. The active force is related to muscle activation signal a, the length of the muscle fiber l_m, and the contraction speed of the muscle fiber $\dot{l}_m$. The passive force depends on l_m. Thus, the resultant force generated by the muscle fiber can be computed by

$$\begin{aligned} F_M &= F_{AE}\left(a, l_m, \dot{l}_m\right) + F_{PE}(l_m) \\ &= a f_l(l_m) f_v\left(\dot{l}_m\right) + f_{pe}(l_m) \end{aligned} \tag{2}$$

where $f_l(*)$ and $f_{pe}(*)$ are nonlinear force-length functions, $f_v(*)$ is a force-velocity function, and they were defined in detail in the previous work of Thelen [41].

Based on the Hill-type muscle equilibrium model, the relationship between forces of the tendon and muscle fiber for one muscle can be expressed as follows:

$$F_T = F_M \cos(\alpha) \tag{3}$$

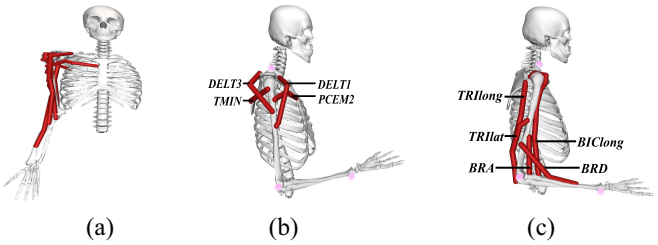
where F_T represents the force generated by the tendon, and α is the pennation angle that indicates the geometric relationship between the tendon and muscle fiber. The tendon connects the muscle fiber and skeleton, which delivers the force to the attached skeleton to generate joint torque. Meanwhile, the same joint is attached by multiple muscles.

Based on the musculoskeletal geometry, the relationship between the muscle contractile velocities and the joint angular velocities can be calculated by the following equation:

$$\dot{\boldsymbol{L}}_{\boldsymbol{mt}} = \mathbf{W}\dot{\boldsymbol{\theta}} \tag{4}$$

where $\boldsymbol{L}_{\boldsymbol{mt}} \in \mathbb{R}^m$ is the muscle length vector, $\mathbf{W} \in \mathbb{R}^{m\times n}$ denotes the Jacobian matrix for muscle space with respect to joint space, and $\boldsymbol{\theta} \in \mathbb{R}^n$ is the joint angular vector. The joint torques generated by tendon forces can be described according to the principle of virtual work as follows:

$$\boldsymbol{T} = \mathbf{W}^T \boldsymbol{F} \tag{5}$$



(a) (b) (c)

Fig. 1. Musculoskeletal model with nine muscles. Nine muscles include DELT1, DELT3, TMIN, PECM2, TRIlong, TRIlat, BIClong, BRA, and BRD. (a) Musculoskeletal model showing all muscles. (b) and (c) Musculoskeletal models showing part of the muscles.

where $\boldsymbol{T} \in \mathbb{R}^n$ is the input torque vector to the joints, and $\boldsymbol{F} \in \mathbb{R}^m$ is a vector composed of the tendon forces of different muscles. The dynamics of the arm musculoskeletal model can be obtained by applying the Lagrange equation of motion as follows:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\dot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{T} \tag{6}$$

where $\mathbf{M}(\boldsymbol{\theta})$, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, and $\mathbf{G}(\boldsymbol{\theta})$ illustrate the mass matrix, Coriolis and centrifugal forces, and the gravitational force, respectively.

In the remainder of this article, a human upper limb musculoskeletal model with two degrees of freedom at the shoulder and elbow joints is adopted, which is obtained by simplifying degrees of freedom and muscles in the previous model [42]. As shown in Fig. 1, it is driven by nine Hill-type muscles attached to skeletons, and it moves in the sagittal plane with the limitation of two degrees of freedom. The models are implemented on the Opensim which is an opensource platform used for simulation of human movement.

III. Method

A. Network Architecture Based on Movement Preparation and Execution

Sufficient observations in neuroscience [32]–[34], [43] support that the motor cortex integrates sensory input or stimuli and exhibits the characteristics of a smooth dynamical system to embed muscle-like commands during movements. In reaching tasks, a population of neurons located in the primary motor cortex and the premotor cortex engages in two processes that fall into movement preparation and execution stage. The same population of neurons acts as two separate circuits with fundamentally distinct properties during two consecutive phases [44].

In this work, RNNs as classical tools of producing rich spatiotemporal patterns [45] are applied to mimic the motor cortical activity. Two RNNs which are referred to as the preparation network and the execution network, respectively, are adopted. At the stage of movement preparation, only the preparation network evolves, the purpose of which is to generate a proper movement-specific initial state of the execution network. When the stage is over, the movement execution stage begins, and the preparation network no longer evolves. During movement execution, only the execution network evolves. The network architecture is shown in Fig. 2(a).

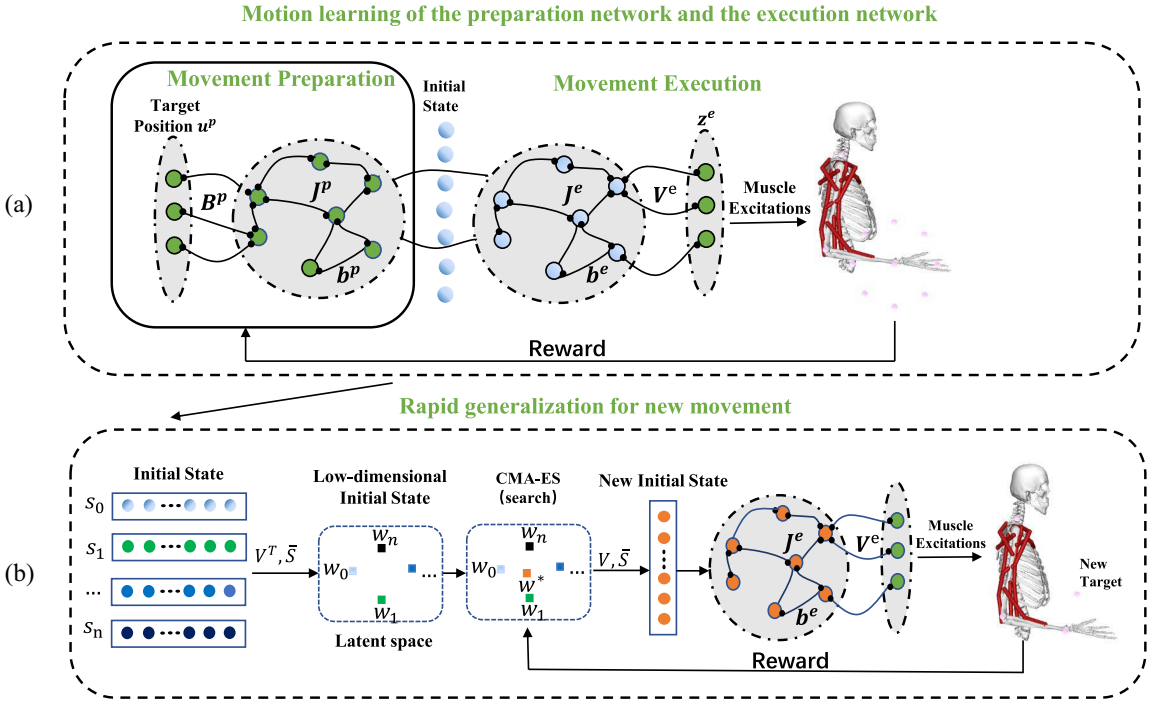


Fig. 2. Schematic of motion learning and rapid generalization for the control of musculoskeletal systems based on RNN modulated by initial states. (a) Motion learning of the preparation network and the execution network. (b) Rapid generalization for a novel movement by searching for corresponding low-dimensional initial state of the execution network in latent space constructed by the learned initial states.

The dynamic equations of the preparation network are consistent with discrete form of the classical RNN equations given by Sussilo [46]

$$\boldsymbol{x}_t^p = \left(1 - \frac{\Delta t}{\tau}\right)\boldsymbol{x}_{t-1}^p + \frac{\Delta t}{\tau}\left(\mathbf{J^p}\boldsymbol{r}_{t-1}^p + \mathbf{B^p}\boldsymbol{u}_t^p + \boldsymbol{b}^p\right) \tag{7}$$

$$\boldsymbol{r}_t^p = \tanh\left(\boldsymbol{x}_t^p\right). \tag{8}$$

The dynamic equations of the execution network are expressed as follows:

$$\boldsymbol{x}_t^e = \left(1 - \frac{\Delta t}{\tau}\right)\boldsymbol{x}_{t-1}^e + \frac{\Delta t}{\tau}\left(\mathbf{J^e}\boldsymbol{r}_{t-1}^e + \boldsymbol{b}^e\right) \tag{9}$$

$$\boldsymbol{r}_t^e = \tanh\left(\boldsymbol{x}_t^e\right) \tag{10}$$

$$\boldsymbol{z}_t^e = \mathbf{V^e}\boldsymbol{r}_t^e \tag{11}$$

where the superscripts "p" and "e" of the variable representations, respectively, indicate whether the variable is related to the preparation network or the execution network, τ is a time constant, the hidden layer of the preparation network and execution network both consist of N neurons, $\boldsymbol{x}^p \in \mathbb{R}^N$and $\boldsymbol{x}^e \in \mathbb{R}^N$are the membrane potential vectors of hidden neurons, $\boldsymbol{r}^p \in \mathbb{R}^N$and $\boldsymbol{r}^e \in \mathbb{R}^N$are the firing rate vectors of hidden neurons, $\boldsymbol{b}^p \in \mathbb{R}^N$and $\boldsymbol{b}^e \in \mathbb{R}^N$are biases, $\mathbf{J^p} \in \mathbb{R}^{N\times N}$and $\mathbf{J^e} \in \mathbb{R}^{N\times N}$are the recurrent weight matrices between hidden neurons of the preparation network or the execution network, respectively, $\mathbf{B^p} \in \mathbb{R}^{N\times M}$is the connection weight matrix of the preparation network from the external input $\boldsymbol{u}^p \in \mathbb{R}^M$ to hidden neurons,$\mathbf{V^e} \in \mathbb{R}^{d\times N}$is the output matrix of the execution network, $\boldsymbol{z}^e \in \mathbb{R}^d$ is the output vector of the execution network, and the value of each element of $\boldsymbol{z}^e$ is clipped to 0 from below and clipped to 1 from above.

In (7)–(11), Δt is a time step interval. The period of time $[0,\ t_1]$ consists of $(t_1/\Delta t) + 1$ time steps, which is called the movement preparation stage. Then, the time period $[t_1,\ t_1 + t_2]$ belongs to movement execution stage.

The execution network directly controls the musculoskeletal system to generate various movements, and its weights will not be changed after being initialized. During the phase of movement execution, the execution network evolves from a specific initial state corresponding to the movement target, and no external input is provided to it. At each time step, the output of the execution network, which can be regarded as muscle excitations, is transmitted to the musculoskeletal system to realize desired movements. The initial state of the execution network prior to each movement is generated by the preparation network, which simulates movement preparation. The preparation network receives the input which represents the position of movement target and transfers the potential of the hidden layer neurons at the last moment of the movement preparation stage, namely the vector $\boldsymbol{x}_{t_1}^p$, as the initial state to the execution network. It is noteworthy that the number of hidden neurons of the preparation network is the same as that of the execution network. Therefore, the transmission of the membrane potential vector between the hidden layers of two networks can be realized. In other words, the execution network receives the initial state delivered from the preparation network as initial potential of its hidden neurons, namely $\boldsymbol{x}_{t_1}^e = \boldsymbol{x}_{t_1}^p$, and it evolves without the external input according to (9)–(11) to generate control signals for the musculoskeletal system from time $t = t_1$ to $t = t_1 + t_2$.

B. Motion Learning of the Preparation Network and the Execution Network

Once parameters related to the execution network including matrices $\mathbf{J^e}$, $\mathbf{V^e}$, and vector $\boldsymbol{b^e}$ are initialized, they remain fixed. As a result, the output of the execution network only depends on its initial state $\boldsymbol{x^e_{t_1}}$. In order to generate the appropriate initial state $\boldsymbol{x^e_{t_1}}$ of the execution network to achieve the desired movement, the weights of the preparation network need to be learned. Therefore, a reward-modulated learning rule based on the node-perturbation method is adopted to modify the weights of the preparation network. The rule is similar to the REINFORCE algorithm that modifies weights along the gradient direction of expected reinforcement. Here, the gradients of the reward function with respect to the weight parameters are estimated based on the node-perturbation method by imposing exploratory perturbations to the neuron activities and computing the fluctuation of the reward function. If the perturbations lead to a better value of the reward function, the parameters are updated along the direction that enables reproducing the perturbations; otherwise, the parameters are updated in the opposite direction.

We apply the Gaussian noise to the potential of hidden neurons of the preparation network at each time step during movement preparation stage to produce exploratory variation in the network as follows:

$$\boldsymbol{x^p_t} = \left(1 - \frac{\Delta t}{\tau}\right)\boldsymbol{x^p_{t-1}} + \frac{\Delta t}{\tau}\left(\mathbf{J^p}\boldsymbol{r^p_{t-1}} + \mathbf{B^p}\boldsymbol{u^p_t} + \boldsymbol{b^p}\right) + \boldsymbol{\gamma_t} \quad (12)$$

where $\boldsymbol{\gamma_t} \in \mathbb{R}^N$, $\boldsymbol{\gamma_t} = [\gamma_t^{(1)}, \gamma_t^{(2)}, \ldots, \gamma_t^{(N)}]$ represents the noise vector applied to hidden neurons of the preparation network, in which $\gamma_t^{(i)}$ is the noise applied to the ith neuron at the tth time step. $\gamma_t^{(i)}$ is subject to a zero-mean normal distribution with variance σ^2. Here, an episode is a movement. According to the node-perturbation algorithm [40], [47], [48], the modification of each element of weight matrix $\mathbf{J^p}$ for each episode could be calculated as follows:

$$\begin{aligned}\Delta J^p_{ij} &= \beta * (R - R_0) * e_{ij} \\ &= \beta * (R - R_0) * \sum_{t=0}^{t_1} \gamma_t^{(i)} \frac{\partial x_t^{p(i)}}{\partial J^p_{ij}} \\ &= \beta * (R - R_0) * \sum_{t=0}^{t_1} \gamma_t^{(i)} \frac{\Delta t}{\tau} r^{p(j)}_{t-1} \\ &= \eta * (R - R_0) * \sum_{t=0}^{t_1} \gamma_t^{(i)} r^{p(j)}_{t-1} \end{aligned} \quad (13)$$

where J^p_{ij} is the weight from hidden neuron j to hidden neuron i of the preparation network, ΔJ^p_{ij} is the increment of the weight J^p_{ij} after an episode, η is the learning rate, R denotes the reward that reflects performance of the initial state generated by the preparation network, and e_{ij} is the eligibility trace of synapse formed by neuron j and neuron i in the hidden layer of the preparation network, which is obtained by accumulating the product of noise and presynaptic input at each moment. R_0 is the expected reward without noise perturbations, which can be approximated by the running average of the rewards within recent episodes. So, in the nth episode

$$R_0(n) = \alpha_{\text{trace}} * R_0(n-1) + (1 - \alpha_{\text{trace}}) * R(n). \quad (14)$$

In (13), a reward prediction error signal in the current episode is reckoned by subtracting R_0 from R, which indicates whether the initial state generated by a noisy preparation network is better or worse than the initial state generated by the noiseless preparation network. ΔJ^p_{ij} can be obtained by multiplying the learning rate η, the fluctuation of the reward $(R - R_0)$ which is caused by applying noise perturbations, and the accumulated eligibility trace e_{ij}. By updating weights according to the above equation, if the accumulated exploratory noise perturbations in one episode are beneficial to final reward, for the same input, the output of preparation network without noise perturbations will be similar to the output before updating the weights when the same accumulated noise perturbations are applied; on the contrary, if the exploratory perturbation reduces the final reward, the effect of updating the weights is equivalent to making the output of preparation network without noise far away from the just-experienced perturbed output. According to (12), the intuitive mechanism of (13) is that $(J^p_{ij} + \gamma_t^{(i)} r^{p(j)}_{t-1}) r^{p(j)}_{t-1}$ makes $x_t^{p(i)}$ move along the direction of $\gamma_t^{(i)}$ for the same $r^{p(j)}_{t-1}$, which reproduces perturbed $x_t^{p(i)}$ to some extent.

Equation (13) can be further expressed as follows:

$$\Delta \mathbf{J^p} = \eta * (R - R_0) * \sum_{t=0}^{t_1} \boldsymbol{\gamma_t} \boldsymbol{r^p_{t-1}}. \quad (15)$$

In the same way, after each episode, the input weight matrix $\mathbf{B^p}$ and the bias vector $\boldsymbol{b^p}$ are updated iteratively by

$$\Delta \mathbf{B^p} = \eta * (R - R_0) * \sum_{t=0}^{t_1} \boldsymbol{\gamma_t} \boldsymbol{u^p_t} \quad (16)$$

$$\Delta \boldsymbol{b^p} = \eta * (R - R_0) * \sum_{t=0}^{t_1} \boldsymbol{\gamma_t}. \quad (17)$$

Based on movement preparation and execution, initial states can be explicitly expressed as knowledge of movements, which is beneficial to motion generalization and will be further described in the next section.

Compared with using only an RNN as motion control network to realize the movement preparation and execution process, by dividing the preparation and execution of movements into two RNNs, the motion learning is accelerated to converge under the application of the node-perturbation method. When only an RNN is used to implement movement preparation and execution process, namely, the recurrent weight matrix $\mathbf{J} = \mathbf{J^p} = \mathbf{J^e}$, in order to update the input weight $\mathbf{B^p}$, perturbations can only be imposed in the time period of $[0, t_1]$ for estimating the gradient of reward function with respect to $\mathbf{B^p}$ and updating $\mathbf{B^p}$. However, the weight $\mathbf{J}$ participates in evolution of the network during the entire time period of $[0, t_1 + t_2]$, so when node-perturbation method is used, the partial gradient of reward function with respect to $\mathbf{J}$ involving the time period of $[t_1, t_1 + t_2]$ cannot be estimated, which is not conducive to convergence of motion learning. For solving

the problem, the preparation and execution of movements are divided into two RNNs as above, and the parameters of the execution network are kept fixed.

C. Rapid Generalization for Novel Movements Based on the Latent Space of the Learned Initial States

By means of controlling the execution network to evolve from different initial states, the downstream musculoskeletal system can realize various movements. Therefore, in order to perform an unlearned motion, it is essential to calculate the specific initial state for the new motion. When motion tasks have similarities, the use of shared knowledge can effectively improve the learning efficiency. In addition, taking into account the optimal subspace hypothesis in neuroscience, redundancy may exist in the high-dimensional initial state to complete a same movement. Inspired by these assumptions, a rapid generalization method based on the latent space of learned initial states for new movement targets is proposed. It no longer requires large-scale training, so that the musculoskeletal system can learn new motion flexibly and quickly.

When given a new movement target, the learned initial states can serve as a kind of motor primitive. For facilitating rapid acquisition of new motion, We first use the learned initial states to construct a latent space by principal component analysis (PCA) [49], [50]. Then, the coordinates of the low-dimensional initial state corresponding to the novel movement target are initialized according to positional relationship between the learned movement targets and the new target in physical space. Subsequently, the optimal low-dimensional initial state for a new movement target in latent space is searched for by combining with the evolutionary algorithm and transformed into the whole neural space, as shown in Fig. 2(b). Hence, the dimensionality of variables that need to be optimized is significantly reduced, so that the rapid generalization of the initial state for a novel movement could be realized.

First, we construct a latent space of initial states via PCA which is a well-known approach to performing dimensionality reduction. Let $\mathbf{S} = [\boldsymbol{s_0}, \boldsymbol{s_1}, \ldots, \boldsymbol{s_n}]$, where $\boldsymbol{s_j} \in \mathbb{R}^N$ represents an initial state which has been learned to achieve a specific movement, that is, the initial membrane potential of hidden neurons of the execution network. The mean vector $\bar{\boldsymbol{s}} \in \mathbb{R}^N$and covariance matrix $\mathbf{C} \in \mathbb{R}^{N\times N}$ of initial states are obtained from the equations

$$\bar{\boldsymbol{s}} = \frac{1}{n+1}\sum_{i=0}^{n} \boldsymbol{s_i} \tag{18}$$

$$\mathbf{C} = \frac{1}{n}\sum_{i=0}^{n} (\boldsymbol{s_i} - \bar{\boldsymbol{s}})(\boldsymbol{s_i} - \bar{\boldsymbol{s}})^T. \tag{19}$$

Let $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ denote an ensemble of eigenvalues of the covariance matrix $\mathbf{C}$, and the elements are arranged in descending order, namely: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. We select ρ orthonormal eigenvectors associated with the largest ρ eigenvalues to form the matrix $\mathbf{V} = [\boldsymbol{e_1}, \ldots, \boldsymbol{e_\rho}]$ which can be regarded as an orthonormal basis for $\mathbb{R}^\rho$, and ρ is the dimensionality of the latent space. Letting $\bar{\mathbf{S}} = [\boldsymbol{s_0} - \bar{\boldsymbol{s}}, \boldsymbol{s_1} - \bar{\boldsymbol{s}}, \ldots, \boldsymbol{s_n} - \bar{\boldsymbol{s}}]$, then the learned high-dimensional initial states can be projected into the latent space which are constructed by ρ eigenvectors

$$\mathbf{W} = \mathbf{V}^T\bar{\mathbf{S}} \tag{20}$$

where $\mathbf{W} = [\boldsymbol{w_0}, \boldsymbol{w_1}, \ldots, \boldsymbol{w_n}]$. Each learned initial state in $\mathbf{S}$ is mapped to a low-dimensional vector in $\mathbf{W}$, which could be regarded as the corresponding coordinate vector in the latent space.

Given a new movement target $\boldsymbol{z}$ expected to be reached, we search for the low-dimensional representation $\boldsymbol{w}$ of the initial state for the novel movement target $\boldsymbol{z}$ based on the covariance matrix adaptation evolutionary strategy (CMA-ES) [51], and transform optimal solution into the whole neural space to reconstruct the high-dimensional initial state, so that the desired movement could be achieved. The process of applying CMA-ES to find the optimal solution vector requires multiple rounds of optimization. In each iteration, there are multiple candidate solution vectors obtained stochastically by sampling the multivariate normal distribution. The performance of each candidate solution is evaluated through the loss function, and parameters of the multivariate normal distribution such as the mean vector would be updated based on the few sampled solution vectors with low loss values, so that the probability of reproducing good candidate solution vectors in the next iteration could be increased. Gradually, we can acquire the appropriate solution with optimal fitness.

Specifically, in each iteration g, multiple parallel low-dimensional initial states are sampled from the multivariate normal distribution, namely

$$\boldsymbol{w_i^g} \sim \mathcal{N}\left(\boldsymbol{\mu^g}, \left(\sigma^g\right)^2\mathbf{\Sigma^g}\right) \quad \text{for } i = 1, \ldots, \lambda \tag{21}$$

where λ represents the number of candidate solution vectors sampled in parallel in an iteration, $\boldsymbol{w_i^g}$ represents the ith candidate solution vector sampled in iteration g, $N(\boldsymbol{\mu^g}, (\sigma^g)^2\mathbf{\Sigma^g})$ represents a multivariate normal distribution with the mean vector $\boldsymbol{\mu^g}$ and covariance matrix $(\sigma^g)^2\mathbf{\Sigma^g}$. The covariance matrix consists of two parts: 1) σ^g denotes the step size and 2) $\mathbf{\Sigma^g}$ is a positive-definite matrix.

Herein, it is a remarkable fact that reasonable initialization of $\boldsymbol{\mu^0}$ can effectively reduce the computational cost of search and improve the speed of search. Therefore, we take advantage of the relation between novel movement target and the learned movement targets in physical space to estimate roughly the low-dimensional initial state for the novel movement target. Specifically, for a novel target, we select k learned movements among all learned motions, whose corresponding movement targets could form a convex polygon which contains new movement target. We calculate the distance from the selected learned movement targets to the new target, and denote the distance by d_m, $m = 1, \ldots, k$. Then, the initial value $\tilde{\boldsymbol{w}}$ of low-dimensional initial state for the new target before optimizing by CMA-ES can be computed as a linear combination of low-dimensional representations of initial states corresponding to selected movements, namely

$$\tilde{\boldsymbol{w}} = \sum_{m=1}^{k} a_m \boldsymbol{w_{(m)}}, \boldsymbol{w_{(m)}} \in \mathbf{W} \tag{22}$$

$$a_m = \frac{d_m^{-1}}{\sum_{m=1}^{k} d_m^{-1}} \quad \text{for } m = 1, \ldots, k \tag{23}$$

where $\boldsymbol{w_{(m)}}$ is the low-dimensional initial state for the mth selected movement. Then, we assign the value of $\tilde{\boldsymbol{w}}$ to the parameter $\boldsymbol{\mu^0}$ related to CMA-ES before sampling candidate vectors in the first iteration. Owing to the use of prior knowledge, learning efficiency can be further improved.

In each iteration, the high-dimensional initial state corresponding to each candidate solution can be calculated as follows:

$$\boldsymbol{s_i^g} = \mathbf{V}\boldsymbol{w_i^g} + \bar{\boldsymbol{s}}. \tag{24}$$

Therefore, these candidate solutions give rise to different movements. We compute the value of loss function $\mathcal{L}(\boldsymbol{w})$ for each candidate solution vector based on the deviation between actual motion and desired motion, and select x candidates with a lower loss among λ candidates to adjust parameters of the multivariate normal distribution. In particular, the mean vector is updated as follows:

$$\boldsymbol{\mu^{g+1}} = \sum_{i=1}^{x} y_i \boldsymbol{w_{i:\lambda}^g} \tag{25}$$

where $\boldsymbol{w_{i:\lambda}^g}$ is the ith best performing candidate solution vector in the iteration, which satisfies: $\mathcal{L}(\boldsymbol{w_{1:\lambda}^g}) \leq \mathcal{L}(\boldsymbol{w_{2:\lambda}^g}) \leq \cdots \leq \mathcal{L}(\boldsymbol{w_{\lambda:\lambda}^g})$, y_i is the weight of $\boldsymbol{w_{i:\lambda}^g}$, and it obeys $\sum_{i=1}^{x} y_i = 1, y_i \geq 0, y_i \propto 1\backslash\mathcal{L}(\boldsymbol{w_{i:\lambda}^g})$ for $i = 1, \ldots, x$. At the same time, in order to sample promising candidate solution vectors with an increased likelihood in the following iterations, Σ and σ also need to be updated iteratively, and they have a corresponding evolution path which represents a summation of a sequence of consecutive steps over a mass of iterations, respectively [52]. The evolution path related to Σ is computed based on the so-called rank-one update for Σ^g as follows:

$$\boldsymbol{p_c^{g+1}} = (1 - c_c)\boldsymbol{p_c^g} + \frac{\sqrt{c_c(2 - c_c)y_{\text{eff}}}}{\sigma_g}\left(\boldsymbol{\mu^{g+1}} - \boldsymbol{\mu^g}\right) \tag{26}$$

where c_c is the cumulation coefficient of the evolution path and y_{eff} is a positive coefficient and it obeys: $y_{\text{eff}} = (\sum_{i=1}^{x}(y_i)^2)^{-1}$. To generate a reliable estimation for the matrix $\boldsymbol{\Sigma}$ in each iteration, it is updated according to the evolution path and several sampled candidate solution vectors with good performance as follows:

$$\boldsymbol{\Sigma_\tau^{g+1}} = \sum_{i=1}^{x} y_i \frac{\left(\boldsymbol{w_{i:\lambda}^g} - \boldsymbol{\mu^g}\right)\left(\boldsymbol{w_{i:\lambda}^g} - \boldsymbol{\mu^g}\right)^T}{(\sigma^g)^2} \tag{27}$$

$$\boldsymbol{\Sigma^{g+1}} = (1 - c_1 - c_c)\boldsymbol{\Sigma^g} + c_1\boldsymbol{p_c^{g+1}}\left(\boldsymbol{p_c^{g+1}}\right)^T + c_\tau\boldsymbol{\Sigma_\tau^{g+1}} \tag{28}$$

where $\boldsymbol{\Sigma_\tau^{g+1}}$denotes the estimation of the matrix $\boldsymbol{\Sigma}$ relying on the selected x candidates in iteration $g + 1$, $\boldsymbol{\Sigma^{g+1}}$ is the estimated matrix $\boldsymbol{\Sigma}$ in the iteration g+1 in practice, and c_1 and c_τ are the learning rates for updating the positive-definite matrix $\boldsymbol{\Sigma}$. The values of c_1, c_τ, and c_1 refer to [53]. The other evolution path is also computed so as to control the step size σ effectively. In the desired situation, successive steps are expected to be independent and uncorrelated. Therefore, the so-called cumulative step length adaptation (CSA) is applied, where the core idea is to promote the actual evolution length of the path to approximate its expected length under random selection as close as possible. The step size σ increases when the actual evolution length is longer than expected, on the contrary, it decreases when the actual evolution length is shorter than expected. Formally, the evolution path related to σ is calculated as follows:

$$\boldsymbol{p_\sigma^{g+1}} = (1 - c_\sigma)\boldsymbol{p_\sigma^g} + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{y_{\text{eff}}}\left(\boldsymbol{\Sigma^g}\right)^{-1/2}\left(\frac{\boldsymbol{\mu^{g+1}} - \boldsymbol{\mu^g}}{\sigma^g}\right) \tag{29}$$

where c_σ is the cumulation coefficient of the evolution path, and $\sqrt{y_{\text{eff}}}(\boldsymbol{\Sigma^g})^{-1/2}([\boldsymbol{\mu^{g+1}} - \boldsymbol{\mu^g}]/\sigma^g) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The update rule of step size can be written as

$$\sigma^{g+1} = \sigma^g \exp\left(\frac{c_\sigma}{d_\sigma}\frac{\left\|\boldsymbol{p^{g+1}}_\sigma\right\|}{E[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]} - \frac{c_\sigma}{d_\sigma}\right) \tag{30}$$

where d_σ denotes the damping parameter for adjusting step size, $E[*]$ computes the expectation of the argument in parentheses. Under random selection, $\boldsymbol{p_\sigma^{g+1}}$ follows multivariate normal distribution with zero mean and unity covariance matrix, so $E[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]$ gives the expected length of $\boldsymbol{p_\sigma^{g+1}}$. The values of c_σ and d_σ can be found in [53].

Gradually, the optimal solution $\boldsymbol{w^*}$, namely, the optimal low-dimensional representation of the initial state for the desired novel movement, is attained via multiple iterations. To achieve the desired movement, the optimal initial state is further given as follows:

$$\boldsymbol{s_{new}} = \mathbf{V}\boldsymbol{w^*} + \bar{\boldsymbol{s}} \tag{31}$$

where $\boldsymbol{s_{new}}$ is the optimal initial state for the novel motion, in other words, $\boldsymbol{s_{new}}$ is the initial membrane potential vectors of hidden neurons of the execution network for the novel movement. Then, the execution network evolves from the initial state $\boldsymbol{s_{new}}$ and produces appropriate muscle excitations, and the musculoskeletal system can complete a novel desired movement accurately. In this way, the dimensionality of variables that need to be optimized is reduced from the number of elements in the initial state, i.e., the number N of hidden neurons of the execution network, to the dimensionality ρ of latent space. The pseudocode of the complete algorithm is shown in Algorithm 1.

IV. Experiments

A. Motion Learning of the Preparation Network and the Execution Network

The experiments to verify the effectiveness of motion learning of the preparation network and execution network are performed, and comparative experiment on dividing movement preparation and execution of movements into one RNN or two RNNs is also performed.

1) Experimental Setup: The center-out reaching task is executed by the musculoskeletal model to evaluate the performance of motion learning based on movement preparation and execution. This task is a classical experiment

Algorithm 1: Rapid Generalization for Novel Movements Based on the Latent Space of the Learned Initial States

Input: loss function $\mathcal{L}(\boldsymbol{w})$, novel movement target z,initial states set $\boldsymbol{S}$; initial distribution parameters $\boldsymbol{\sigma^0}$,and $\boldsymbol{\Sigma^0}$;

Output: initial state $\boldsymbol{s_{new}}$ for the novel movement;

- Compute the mean vector $\bar{s}$ and projection matrix $\mathbf{V}$;
- Set the value of $\boldsymbol{\mu^0}$ with Eq. (22), (23)

for $g = 0 : N_g$ **do**

  for $i = 1 : \lambda$ **do**

    - Obtain a low-dimensional initial state by sampling : $\boldsymbol{w_i^g} \sim \mathcal{N}(\boldsymbol{\mu^g}, (\sigma^g)^2\boldsymbol{\Sigma^g})$;
    - Calculate the new initial state using $\boldsymbol{w_i^g}$: $\boldsymbol{s_i^g} = \mathbf{V}\boldsymbol{w_i^g} + \bar{s}$;
    - The execution network evolves from $\boldsymbol{s_i^g}$, and generate muscle excitations;
    - The musculoskeletal system makes a movement with muscle excitations;
    - Evaluate the solution vector using $\mathcal{L}(\boldsymbol{w_i^g})$;

  - Select x best solution vectors with lower loss;
  - Update $\boldsymbol{\mu^{g+1}}$,$\boldsymbol{\Sigma^{g+1}}$,$\sigma^{g+1}$with Eq. (25)-(30);

  if $g == N_g$ **then** select the best solution vector $\boldsymbol{w^*}$;

- Reconstruct the initial state for the novel movement: $\boldsymbol{s_{new}} = \mathbf{V}\boldsymbol{w^*} + \bar{s}$.

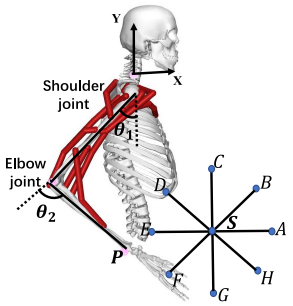


Fig. 3. Experimental Setup of the center-out reaching task. The musculoskeletal model is required to remove its endpoint $\boldsymbol{P}$ from the start position $\boldsymbol{S}$ to eight targets $\boldsymbol{A}$-$\boldsymbol{H}$.

paradigm to investigate motor control of humans in neuroscience, where the participants require to reach from a central location to different targets on a circle surrounding the start position. As depicted in Fig. 3, before each movement, the start position of the endpoint of arm model is $\boldsymbol{S}$. The movement targets are eight equidistant points $\boldsymbol{A}$-$\boldsymbol{H}$ on the circle with a radius of 14 cm around the start position $\boldsymbol{S}$.

First, an RNN is defined to act as the execution network whose output is fed into the musculoskeletal model to activate the muscles. To make the musculoskeletal model move its endpoint from the start position S to eight equidistant movement targets successfully, the other RNN called the preparation network is trained to generate appropriate initial states of the execution network. The 2-D position vector $\boldsymbol{p_d} = (x_d, y_d)$ of eight targets on the circle in the sagittal plane is provided to the preparation network in turn as the input vector in different episodes. The hidden layers of both RNNs are composed

TABLE I
PARAMETERS RELATED TO THE PREPARATION NETWORK AND ITS WEIGHTS MODIFICATION

Symbol	Default Value	Description
τ	10ms	time constant of neuron activation
Δt	1ms	time step
p_0	0.3	probability of weights initially set to zero
N	200	number of neurons in hidden layer
M	2	input dimension
t_1	0.1s	evolving time of the preparation network
–	$U(-0.1, 0.1)$	uniform distribution of initial $\mathbf{B^p}$
–	$U(-0.6, 0.6)$	uniform distribution of initial $\boldsymbol{b^p}$
–	$\mathcal{N}(0, 0.1^2)$	normal distribution of initial $\mathbf{J^p}$
σ	6	standard deviation of node perturbation
α_{trace}	0.33	filter factor of the expected reward

TABLE II
PARAMETERS RELATED TO THE EXECUTION NETWORK

Symbol	Default Value	Description
τ	10ms	time constant of neuron activation
Δt	1ms	time step
p_0	0.3	probability of weights initially set to zero
N	200	number of neurons in hidden layer
d	9	output dimension
t_2	0.27s	evolving time of the execution network
–	$\mathcal{N}(0, 0.1^2)$	normal distribution of $\mathbf{J^e}$
–	$U(-0.8, 0.8)$	uniform distribution of $\boldsymbol{b^e}$
–	$U(-0.2, 0.2)$	uniform distribution of $\mathbf{V^e}$

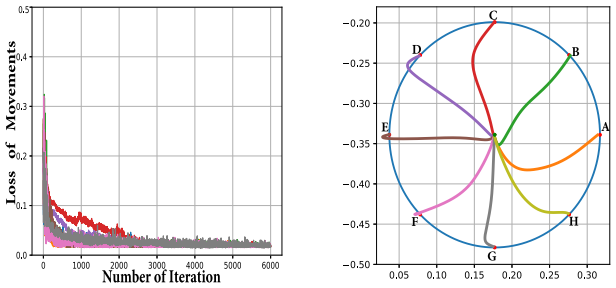
of $N = 200$ neurons. The weights of the preparation network and execution network are initialized according to previous work [39], [47], so that the RNNs can have relative smooth dynamics and enough representation ability. More concretely, each element of $\mathbf{B^p}$, $\mathbf{J^p}$, $\boldsymbol{b^p}$, $\mathbf{J^e}$, $\boldsymbol{b^e}$, and $\mathbf{V^e}$ is initially set to 0 with probability p_0. The nonzero elements of $\mathbf{B^p}$, $\boldsymbol{b^p}$, $\boldsymbol{b^e}$, and $\mathbf{V^e}$ are initialized from uniform distributions over a small range, and the remaining elements of $\mathbf{J^p}$ and $\mathbf{J^e}$ are drawn from a normal distribution. The spectral radius of $\mathbf{J^e}$ is less than 1. The membrane potential of each hidden neuron of the preparation network is set to 0 when each episode begins. The weights related to the execution network are kept fixed after they are initialized. We update the weights of preparation network based on the learning rule as described in Section III-B. Details on parameters related to the preparation network and weights modification are listed in Table I, and details on parameters concerning the execution network are given in Table II.

During training, the performance of each episode is evaluated by adopting the following loss function:

$$L = -R = \gamma_1 \left\| \boldsymbol{p} - \boldsymbol{p^d} \right\|^2 + \gamma_2 \left\| \dot{\boldsymbol{p}} \right\|^2 \tag{32}$$

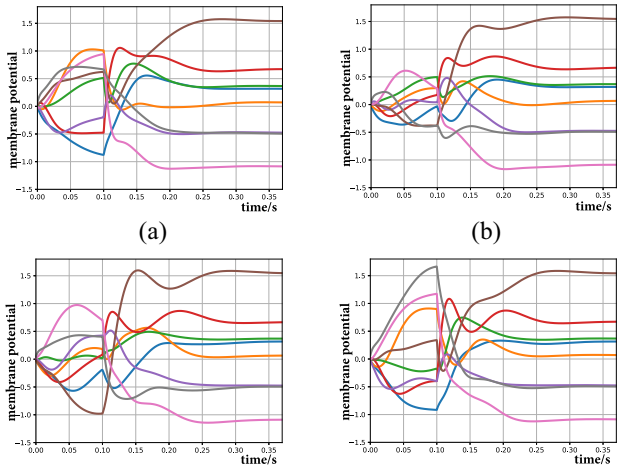
where $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$ represent the final position and velocity of the endpoint of upper limb model, respectively, and $\|\cdot\|^2$ computes the 2-norm. γ_1 and γ_2 are used to set the relative importance of position error and velocity error, and they are tuned to 1 and 0.005, respectively.

2) Motion Learning by Training the Preparation Network: According to the method proposed in Section III-B, the weights of the preparation network can be adjusted iteratively in the direction of producing proper initial states after each episode. After training, the execution network can evolve from the appropriate initial states, which are produced by the trained



(a) (b)

Fig. 4. Performance on motion learning based on the preparation network and the execution network. (a) Value of loss function during training. (b) Trajectories of the endpoint of arm model when training is completed.



(a) (b) (c) (d)

Fig. 5. Membrane potential of partial neurons in hidden layer during movement preparation and execution stage. (a)–(d), respectively, represent the membrane potential of partial neurons in hidden layer when the movement target is point *A*, *C*, *E*, and *G* on the circle.

preparation network. At the same time, it outputs muscle excitation signals to drive the arm model to generate the desired movements with high precision. The curves of loss function associated with the eight target points are shown in Fig. 4(a). It is obvious that the deviation between actual movement and expected movement decreases rapidly as the number of episodes increases. Fig. 4(b) shows the trajectories of the endpoint of the arm model when training is completed in the center-out reaching task. The average error of the endpoint is 1.60 ± 0.83 mm.

The membrane potentials of partial neurons in the hidden layer of the preparation network and execution network during movement preparation and execution stages are shown in Fig. 5. At $t = 0.10$ s, the evolving network is switched from the preparation network to the execution network, and the preparation network transmits the membrane potentials of neurons in the hidden layer at that moment to the execution network as its initial state. Therefore, at $t = 0.10$ s, due to the change of network characteristics and the withdrawal of input vector, the membrane potentials have obvious fluctuations. As shown in Fig. 5, during movement preparation, that is, before $t = 0.1$ s, for the same hidden neurons, the time evolutions of membrane potentials corresponding to different movement targets are significantly different, and membrane

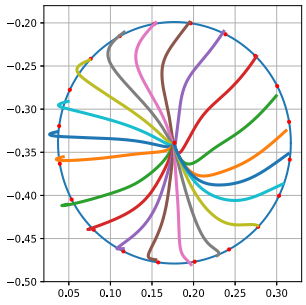
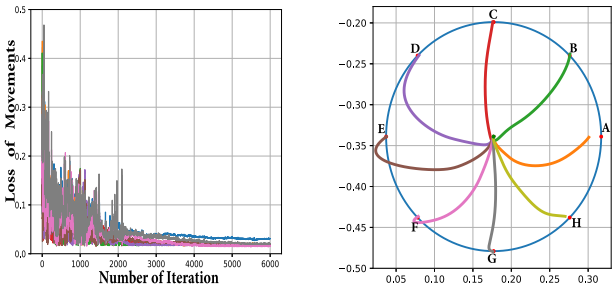


Fig. 6. Performance on motion generalization of the preparation network for the novel movement targets on the circle.



(a) (b)

Fig. 7. Performance on motion learning based on one RNN that goes through movement preparation and execution process. (a) Value of loss function during training. (b) Trajectories of the endpoint of the arm model when training is completed.

potentials have rich patterns at the period, which gives rise to high degree of differentiation between initial states. This is crucial for providing specific initial states to the execution network according to various movement targets.

3) Motion Generalization of the Preparation Network: In addition, after the training, 20 untrained points on the same circle are obtained as movement targets to test the generalization ability of the preparation network, and the trajectories of the endpoint are given in Fig. 6. Through the observation of Fig. 6, the motion learning of the preparation network and execution network can provide the ability of generalization in terms of movement targets with the same distance and different directions.

4) Comparative Experiment on Dividing the Preparation and Execution of Movements Into Two RNNs or One RNN: A comparative experiment is carried out to verify that dividing the preparation and execution of movements into two RNNs rather than one RNN accelerates the motion learning. In the comparative experiment, only one RNN is applied as a motion control network, which also goes through the motion preparation and execution process. The performance of motion learning is depicted in Fig. 7. By comparing it with Fig. 4, when two RNNs are adopted, the loss curve fluctuates less and declines more steadily, the motion learning is faster and can achieve movements with higher precision under the same number of learning episodes.

B. Rapid Generalization for Novel Movements Based on the Latent Space of the Learned Initial States

Through motion learning of the preparation network and execution network, initial states can be explicitly expressed as

knowledge of movements, and some movement-specific initial states have been learned successfully. In the following experiments, the effectiveness of proposed rapid generalization method for new movement targets with the utilization of learned initial states is verified.

1) Rapid Generalization for Novel Movements by Searching for the Corresponding Low-Dimensional Initial States: To prove the validity of the proposed rapid generalization method in Section III-C, we acquired 40 novel movement targets by uniformly sampling points within the same circle, and they are used for reaching tasks. We still adopt the execution network with the same parameters depicted in Table II. Eight initial states, which are generated by the preparation network for the eight equidistant targets on the circle in reaching task, are utilized to construct the latent space according to (18) and (19). The loss function $\mathcal{L}(\boldsymbol{w})$ used to evaluate the low-dimensional initial state coordinates is the same as that in (32). In our experiments, the parameter λ of CMA-ES is set to 10.

Since we applied eight learned 200-D initial states to construct the latent space, the effective projection subspace has a maximum dimension of eight. Under the premise that the dimension of latent space has enough representation ability, the lower the dimensionality of the latent space, the more beneficial it is to improve the efficiency of search for optimal low-dimensional initial state for the desired motion. Herein, the following experiments have indicated the 2-D latent space is sufficient for the rapid generalization of new movement targets within circle. The 2-D latent space also provides convenience for visualization. So the latent spaces in all subsequent experiments are all 2-D, namely, $\rho = 2$.

As mentioned in Section III-C, in an attempt to set the parameter $\boldsymbol{u^0}$ of CMA-ES properly to improve the search efficiency, we initialize the coordinate value of low-dimensional initial state for the novel movement according to (22) and (23) before searching for the optimal solution by CMA-ES and assign the initial value to $\boldsymbol{u^0}$, herein some learned motions need to be selected to calculate the initial value. Specifically, for a novel movement target within the circle, the learned movements which are corresponding to the two nearest targets and the two farthest targets to the new target among the existing eight movement targets on the circle are selected. The reason for this selection is that for almost all points within circle as new movement target, if only initial states for the eight movement targets on circle are provided as prior knowledge, the movement targets corresponding to the selected learned motions could form a convex polygon containing a new movement target.

The initial states of the execution network corresponding to new movements could be learned successfully by searching for proper low-dimension ones in latent space to minimize the loss function and then transforming them into the whole neural space. Fig. 8 shows final trajectories of the endpoint of the musculoskeletal system under ten rounds of iterations of CMA-ES. The average error is 0.75 ± 0.67 mm. It indicates all new movement targets can be reached accurately without large-scale training and confirms the proposed method based on latent space is not only feasible but also can effectively

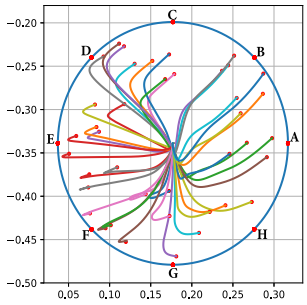


Fig. 8. Performance on rapid generalization for new targets within the circle based on the 2-D latent space of the learned eight initial states.

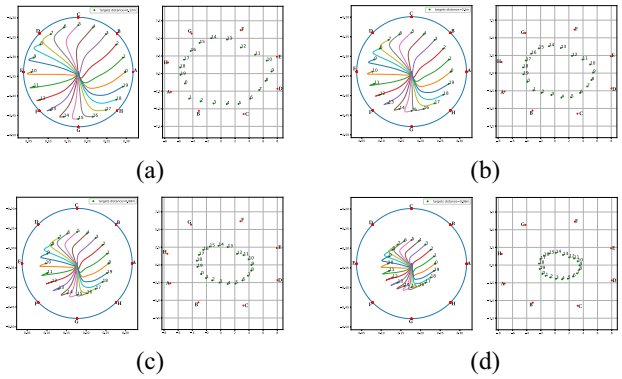


Fig. 9. Performance on rapid generalization for new targets and the distribution of the corresponding low-dimensional initial states. (a)–(d), respectively, represent the performance on rapid generalization for new targets on the circles based on the 2-D latent space of the learned eight initial states (left), and the distribution of optimal low-dimensional initial states in the 2-D latent space (right) when targets are evenly arranged along circles with radii of 12, 10, 8, and 6 cm.

reduce the computational burden for generalization of new movements. This is because the learned initial states are fully utilized as prior knowledge, and the dimensionality of the controlled variable has been greatly reduced.

In addition, to observe the coordinate distribution of different initial states in latent space, four groups of experiments on center-out reaching tasks are designed, in which the same number of movements targets are evenly arranged along circles with radii of 12, 10, 8, and 6 cm, respectively. Fig. 9 illustrates the motion trajectories of the musculoskeletal model after motion generalization. The optimal low-dimensional initial states corresponding to movement targets are also given in Fig 9, which makes it abundantly clear that the initial states of the execution network for movement targets are not disorderly distributed in the latent space. It is observed that the distribution of low-dimensional movement-specific initial states in the latent space has a certain similarity with the arrangement of motion targets in physical space. This provides a basis to estimate low-dimensional initial state roughly by (22) and (23) according to the positional relationship between the learned movement targets and the new target in physical space before searching.

2) Impact of Initializing Low-Dimensional Initial States Based on the Position of Corresponding Targets: We conducted experiments to observe the effect of initializing the

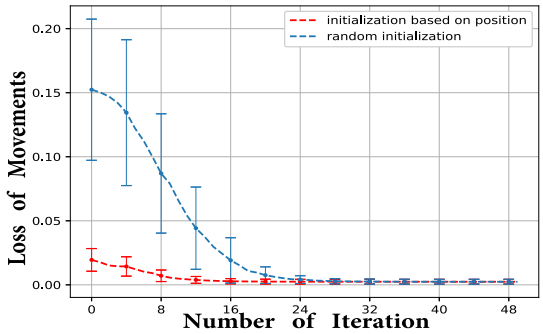


Fig. 10. Results of comparative experiments which show the impact of random initialization of the low-dimensional initial state and initialization of the low-dimensional initial state based on the physical location of the target.

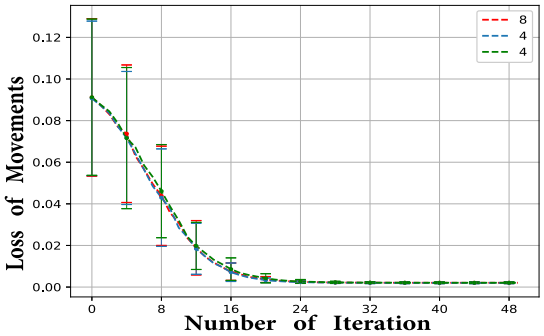


Fig. 11. Results of comparative experiments which show the impact of the number of selected initial states used for constructing the latent space.

low-dimensional initial state for new movement target based on the distribution of targets. Forty uniformly sampled points within the same circle with the radius of 14 cm are obtained as new movement targets, and comparative experiments under random initialization and initialization based on (22) and (23) are carried out to verify the effectiveness of initializing the low-dimensional initial state for new target. The initial value of low-dimensional initial state is assigned to the parameter $\boldsymbol{u}^{\mathbf{0}}$ of CMA-ES. As reflected in Fig. 10, reasonable initialization of the low-dimensional initial state can significantly speed up the process of motion generalization.

3) Impact of the Number of Selected Learned Initial States Used for Constructing the Latent Space: The learned high-dimensional initial states are not only utilized to solve the orthonormal basis of the latent space but also useful for reconstructing the high-dimensional initial state for a novel movement target by means of the mean vector of them, as in (24). To observe whether the number of initial states engaging in producing the latent space has an effect on learning novel movement, we compared the performance of generalization when latent space is based on the learned initial states with different quantity.

We sampled 40 points uniformly within the circle with the radius of 14 cm as novel targets in reaching tasks. In the contrast experiments, the latent spaces are produced through $n = 4, 8$ learned high-dimensional states corresponding to $n = 4, 8$ equidistant targets on the circle, respectively. It is worth noting that there are two possible sets to pick four equidistant targets out of the eight learned targets on the circle with radius of 14 cm to construct latent space. As Fig. 3 shows, one set consists of A, C, E, and G. The other set consists of B, D, F, and H. In order to eliminate the impact of initialization on learning speed in different cases, initial values of low-dimensional initial states for new reaching experiments are the mean of the low-dimensional coordinates of the n learned initial states in each case. Fig. 11 shows the three loss curves of the three cases during searching for the optimal low-dimensional initial states corresponding to new targets. In three cases, after 50 rounds of iterations of CMA-ES, the average errors of endpoint are all less than 1 mm. Since new targets can all be reached with high precision, the initial states with different quantity in three cases contain enough information that are needed to reconstruct the high-dimensional initial states. Furthermore, the three loss curves almost overlap with each other after removing the effects of initialization to the low-dimensional initial states in the three cases. It illustrates that the number of initial states involving in constructing latent space has no significant influence on the search speed of the optimal low-dimensional initial states for new targets and accuracy of new movements, as long as the initial states constructing the latent space contain enough information.

C. Fault Tolerance Analysis of the Initial State

In [38], the hypothesis that "optimal subspace" of an appropriate state for one motion is a high-dimensional ellipsoid is proposed, which indicates the noise occurring in some directions will hardly affect the motion consequences, while noise along some directions will cause the actual motion to deviate significantly from the desired motion. This shows that the optimal initial state corresponding to a desired motion has a certain tolerance space. The hypothesis is also applicable to the initial state of the execution network in this article and is verified in the experiment.

As mentioned before, the dynamic equation of the execution network can be expressed as follows:

$$\tau \dot{\boldsymbol{x}}^{\boldsymbol{e}} = -\boldsymbol{x}^{\boldsymbol{e}} + \mathbf{J}^{\mathbf{e}} \tanh\left(\boldsymbol{x}^{\boldsymbol{e}}\right) + \boldsymbol{b}^{\boldsymbol{e}}. \tag{33}$$

Although $\tanh(\boldsymbol{x}^{\boldsymbol{e}})$ in (33) brings nonlinearity to the execution network, $\tanh(\boldsymbol{x}^{\boldsymbol{e}(i)}) \approx 0.8\boldsymbol{x}^{\boldsymbol{e}(i)}$ when element $\boldsymbol{x}^{\boldsymbol{e}(i)} \in [-1, 1]$, and $\boldsymbol{x}^{\boldsymbol{e}(i)}$ represents the ith element of vector $\boldsymbol{x}^{\boldsymbol{e}}$. Given the membrane potentials of most hidden neurons are in the range $[-1, 1]$ during the movement execution stage, for the convenience of analysis, we ignore the nonlinearity of the execution network dynamics for the time being and regard it as linear, so (33) can be further approximated as follows:

$$\tau \dot{\boldsymbol{x}}^{\boldsymbol{e}} = \left(0.8\mathbf{J}^{\mathbf{e}} - \mathbf{I}\right)\boldsymbol{x}^{\boldsymbol{e}} + \boldsymbol{b}^{\boldsymbol{e}}. \tag{34}$$

Let $\mathbf{A} = 0.8\mathbf{J}^{\mathbf{e}} - \mathbf{I}$, if $\boldsymbol{x}^{\boldsymbol{e}}$ evolves from the initial state $\boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{e}}$ from time $t = 0$ onward, (34) can be solved analytically in mathematics as follows:

$$\boldsymbol{x}^{\boldsymbol{e}}(t) = \exp\left(\frac{\mathbf{A}}{\tau}t\right)\boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{e}} + \int_0^t \exp\left(\frac{\mathbf{A}}{\tau}(t-s)\right)\boldsymbol{b}^{\boldsymbol{e}} ds. \tag{35}$$

The muscle excitations produced by the network can be calculated as follows:

$$z^{e}(t) = \mathbf{V^e}\tanh\left(x^{e}(t)\right) \approx 0.8\mathbf{V^e}\left(\exp\left(\frac{\mathbf{A}}{\tau}t\right)x_{\mathbf{0}}^{e} + f(t)\right) \quad (36)$$

where $f(t) = \int_0^t \exp([\mathbf{A}/\tau](t-s))b^e ds$. Assume that $x_{\mathbf{0}}^{e*}$ is an appropriate initial state which has been learned for target p_d. Similarly, the appropriate muscle excitations which can drive the musculoskeletal system to reach the target p_d can be written as follows:

$$z^{e*}(t) = \mathbf{V^e}\tanh\left(x^{e*}(t)\right) \approx 0.8\mathbf{V^e}\left(\exp\left(\frac{\mathbf{A}}{\tau}t\right)x_{\mathbf{0}}^{e*} + f(t)\right). \quad (37)$$

We define the output error of the execution network as the total squared difference between the actual output and desired output in the following equation. Though in fact, in this experiment, only the time $[0, t_2]$ belongs to the movement execution stage, here we expand the integration domain to an infinite domain to compare the difference between the outputs of the execution network over total time courses under different initial states

$$\begin{aligned} e &= \int_0^{\infty} \left\| z^{e}(t) - z^{e*}(t) \right\|^2 dt \\ &= \int_0^{\infty} \left\| 0.8\mathbf{V^e}\exp\left(\frac{\mathbf{A}}{\tau}t\right)\left(x_{\mathbf{0}}^{e} - x_{\mathbf{0}}^{e*}\right) \right\|^2 dt \\ &= 0.64\left(x_{\mathbf{0}}^{e} - x_{\mathbf{0}}^{e*}\right)^T \mathbf{Q}\left(x_{\mathbf{0}}^{e} - x_{\mathbf{0}}^{e*}\right) \end{aligned} \quad (38)$$

where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is called "observability Gramian" of pair $([\mathbf{A}/\tau], \mathbf{V^e})$, and $\mathbf{Q}$ can be computed by solving a related Lyapunov equation in mathematics, the derivation details of (38) refers to [54].

According to (38), under the assumption that the dynamics of the execution network is approximately linear, if deviations with the same size (Euclidean norm) between $x^e(t)$ and $x^{e*}(t)$ are along the direction of the eigenvectors of $\mathbf{Q}$ corresponding to large eigenvalues of $\mathbf{Q}$, they will cause larger error of muscle excitations. On the other hand, deviations along the direction of the eigenvectors associated with small eigenvalues of $\mathbf{Q}$ have less impact on suitable muscle excitations for the target p_d.

We made the following experiments to verify this conclusion. Noises with the same value of the Euclidean norm, which are along different eigenvector directions, are applied to the appropriate initial states corresponding to multiple targets on the circle. The average position deviations of different targets due to the application of the noise along the same eigenvector direction are calculated. Then, the 200 eigenvectors of $\mathbf{Q}$ are sorted in the descending order of their related eigenvalues, and every eight eigenvectors are divided into one set. The average error related to 8 eigenvectors in each set is further calculated, and 25 sets of corresponding average errors are shown with bars of the same color in Fig. 12. The larger the index of eigenvector sets, the smaller the corresponding eigenvalues of the eigenvectors in this set. As shown in Fig. 12, as the index of eigenvector sets increases, the average position errors decrease. Although the actual network dynamic equations are nonlinear, (38) still accounts for the result to some extent. Specifically, if the actual initial state deviates from the correct initial state in the direction of the eigenvectors with large eigenvalues, it will lead to a large error of muscle excitations, which will further give rise to a large deviation of the final reaching position of the musculoskeletal system. On the contrary, the deviation of the initial state in the direction of eigenvectors with small eigenvalues has less influence. So the perturbations in the potent direction of the eigenvectors with large eigenvalues should be eliminated as possible for the precise control of the musculoskeletal system. The experiments of applying noises with different sizes along the direction of the eigenvectors were also performed, whose results are shown with bars of different colors in Fig. 12. As the size of applied noise gradually increases, the final position deviation of the musculoskeletal system also increases. Although the initial state has great fault tolerance in the directions of eigenvectors with small eigenvalues and some perturbations in these directions are allowed, the perturbation will still cause large movement deviation when the size of perturbation exceeds a certain level.

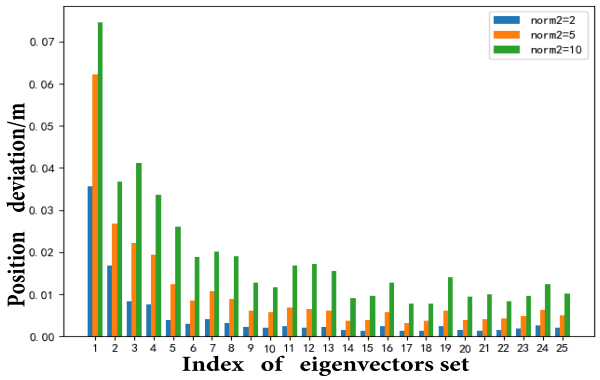


Fig. 12. Results which show position deviation caused by applying noise along different eigenvector directions to the initial state.

V. Discussion

In this section, the work in this article is compared with related work, and the future study is also discussed.

Kao *et al.* [38] aimed to propose an appropriate theoretical scaffold for formulating movement preparation in neuroscience. They adopted optimal anticipatory control to let the RNN automatically evolve from a random state to the desired initial state which is given previously before the reaching motion of a two-link arm. However, the process of getting initial state takes advantage of the expected torque trajectory of the arm which is obtained by backpropagating through the dynamics equations of the two-link arm, but this is difficult for the musculoskeletal system owing to its high redundancy and coupling. In addition, the method in [38] lacks the generalization ability for novel movement with high efficiency. In [55], Sun *et al.* designed an RNN as motor network to control an arm plant which is actuated by six muscles. Although the evolution process of the network is also divided into movement preparation and execution processes, appropriate time-varying motor commands are generated by using Hessian-free optimization to update the weights of the

motor network. Therefore, in [55], initial states are not explicitly modulated and utilized as prior knowledge to speed up subsequent learning. In our article, the initial states can be regarded as a kind of motor primitive, and the execution network is deterministic once it is initialized, so the learned initial states could be reused to generate new initial states for novel movements.

In this article, concentrating on modulating the initial state of the control system based on movement preparation provides a new perspective on the control method of musculoskeletal robotic systems. The framework may also give inspiration to research in some other potential application scenarios, such as upper limb exoskeleton robots and functional electrical stimulation (FES) technology which is applied for the restoration of upper extremity function [56]. In these scenarios, the controlled systems are all strongly redundant, nonlinear, and coupling. However, the proposed framework in this article still have some limitations. First, although the learned initial states could be used to achieve rapid generalization for novel movements, several rounds of optimization are still needed for acquiring the corresponding low-dimensional initial states of the unlearned targets. Second, the framework is used in reaching tasks, the trajectory cannot be as smooth as that of a human due to the lack of trajectory constraints. Third, during motion learning, the reward-modulated learning rule requires plenty of experiments with the musculoskeletal robots, which limits the efficiency in related applications. In the future, we will consider realizing the generalization for new motion without additional optimization. Specifically, an initial state library can be established, and the newly acquired initial state can be added to the library to guide the subsequent movement, so that the efficiency of the algorithm can be improved by incremental learning. The distribution law of initial states in potential space can be further discussed and clearly expressed so as to facilitate generalization. Besides, the existence of optimal subspace reveals which directions are important for precise control, and gives an inspiration that the preparatory errors along these potent directions should first be selectively eliminated during the motor preparation in the future work. In addition, the reaching movement is one form of various robotic manipulations, the musculoskeletal robotic system may be applied to realize more complex tasks such as tracking [57], [58]. More key neural mechanisms on motion control will be incorporated into the control method to reap the potential advantages of the human brain, so that human-like flexible manipulation could be performed.

VI. Conclusion

The main motivation for this article comes from the studies of the neural responses in the motor cortex during animal motion in neuroscience. In a series of motion tasks such as fast ballistic movement, the neural activities in motor cortex can be described by a dynamic system which evolves from a specific initial state in each episode.

Inspired by the research on movement preparation in the motor cortex, a motion learning framework based on RNN modulated by initial states is proposed for a sophisticated musculoskeletal system. First, two RNNs are introduced as the preparation network and the execution network to achieve the motion learning. Dividing the preparation and execution of movements into two RNNs accelerates the convergence of motion learning under the application of the node perturbation method. Based on movement preparation, initial states of RNN can be explicitly utilized as knowledge of movement targets. On that basis, a method that can realize rapid generalization for new movement is further proposed. We proposed to optimize the low-dimensional initial state related to new motion in latent space of the learned initial states and reconstruct the corresponding high-dimensional ones to achieve rapid generalization. A musculoskeletal robotic system with two joints and nine muscles was adopted to verify the effectiveness of the proposed motion learning framework.

References

[1] K. P. Tee, D. W. Franklin, M. Kawato, T. E. Milner, and E. Burdet, "Concurrent adaptation of force and impedance in the redundant muscle system," *Biol. Cybern.*, vol. 102, no. 1, pp. 31–44, 2010.

[2] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 3, pp. 1193–1203, Jun. 2018.

[3] Y. Nakanishi *et al*. "Design approach of biologically-inspired musculoskeletal humanoids," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 4, pp. 216–228, 2013.

[4] H. Qiao, J. Chen, and X. Huang, "A survey of brain-inspired intelligent robots: Integration of vision, decision, motion control, and musculoskeletal systems," *IEEE Trans. Cybern.*, early access, Apr. 28, 2021, doi: 10.1109/TCYB.2021.3071312.

[5] S. Zhong, J. Chen, X. Niu, H. Fu, and H. Qiao, "Reducing redundancy of musculoskeletal robot with convex hull vertexes selection," *IEEE Trans. Cogn. Develop. Syst.*, vol. 12, no. 3, pp. 601–617, Sep. 2020.

[6] Y. Asano *et al*. "Human mimetic musculoskeletal humanoid kengoro toward real world physically interactive actions," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots (Humanoids)*, 2016, pp. 876–883.

[7] Y. Asano, K. Okada, and M. Inaba, "Design principles of a human mimetic humanoid: Humanoid platform to study human intelligence and internal body system," *Sci. Robot.*, vol. 2, no. 13, 2017, Art. no. eaaq0899.

[8] S. Wittmeier *et al*. "Toward anthropomimetic robotics: Development, simulation, and control of a musculoskeletal torso," *Artif. Life*, vol. 19, no. 1, pp. 171–193, 2013.

[9] Y. Dabiri, S. Najarian, M. R. Eslami, S. Zahedi, and D. Moser, "A powered prosthetic knee joint inspired from musculoskeletal system," *Biocybern. Biomed. Eng.*, vol. 33, no. 2, pp. 118–124, 2013.

[10] C.-T. Chen, W.-Y. Lien, C.-T. Chen, and Y.-C. Wu, "Implementation of an upper-limb exoskeleton robot driven by pneumatic muscle actuators for rehabilitation," *Actuators*, vol. 9, no. 4, p. 106, 2020.

[11] E. P. Grabke, K. Masani, and J. Andrysek, "Lower limb assistive device design optimization using musculoskeletal modeling: A review," *J. Med. Devices*, vol. 13, no. 4, p. 13, 2019.

[12] Z. Li, C.-Y. Su, G. Li, and H. Su, "Fuzzy approximation-based adaptive backstepping control of an exoskeleton for human upper limbs," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 3, pp. 555–566, Jun. 2015.

[13] S. Kurumaya, K. Suzumori, H. Nabae, and S. Wakimoto, "Musculoskeletal lower-limb robot driven by multifilament muscles," *Robomech J.*, vol. 3, no. 1, pp. 1–15, 2016.

[14] Y. Wu, J. Chen, and H. Qiao, "Anti-interference analysis of bio-inspired musculoskeletal robotic system," *Neurocomputing*, vol. 436, no. 3, pp. 114–125, 2021.

[15] F. C. Anderson and M. G. Pandy, "Static and dynamic optimization solutions for gait are practically equivalent," *J. Biomech.*, vol. 34, no. 2, pp. 153–161, 2001.

[16] D. G. Thelen, F. C. Anderson, and S. L. Delp, "Generating dynamic simulations of movement using computed muscle control," *J. Biomech.*, vol. 36, no. 3, pp. 321–328, 2003.

[17] K. Tahara and H. Kino, "Reaching movements of a redundant musculoskeletal arm: Acquisition of an adequate internal force by iterative learning and its evaluation through a dynamic damping ellipsoid," *Adv. Robot.*, vol. 24, nos. 5–6, pp. 783–818, 2010.

[18] K. Tahara, Y. Kuboyama, and R. Kurazume, "Iterative learning control for a musculoskeletal arm: Utilizing multiple space variables to improve the robustness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4620–4625.

[19] M. H. Balaghi I., R. Vatankhah, M. Broushaki, and A. Alasty, "Adaptive optimal multi-critic based neuro-fuzzy control of mimo human musculoskeletal arm model," *Neurocomputing*, vol. 173, pp. 1529–1537, Jan. 2016.

[20] W. Qi, H. Su, and A. Aliverti, "A smartphone-based adaptive recognition and real-time monitoring system for human activities," *IEEE Trans. Human-Mach. Syst.*, vol. 50, no. 5, pp. 414–423, Oct. 2020.

[21] H. Kambara, K. Kim, D. Shin, M. Sato, and Y. Koike, "Learning and generation of goal-directed arm reaching from scratch," *Neural Netw.*, vol. 22, no. 4, pp. 348–361, 2009.

[22] J. Weng, E. Hashemi, and A. Arami, "Natural walking with musculoskeletal models using deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4156–4162, Apr. 2021.

[23] Łukasz Kidziński *et al*. "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," in *Proc. NIPS Competition Build. Intell. Syst.*, 2018, pp. 121–153.

[24] Łukasz Kidziński *et al*. "Artificial intelligence for prosthetics: Challenge solutions," in *Proc. NeurIPS Competition*, 2020, pp. 69–128.

[25] A. Diamond and O. E. Holland, "Reaching control of a full-torso, modelled musculoskeletal robot using muscle synergies emergent under reinforcement learning," *Bioinspiration Biomimetics*, vol. 9, no. 1, 2014, Art. no. 016015.

[26] A. d'Avella, P. Saltiel, and E. Bizzi, "Combinations of muscle synergies in the construction of a natural motor behavior," *Nat. Neurosci.*, vol. 6, no. 3, pp. 300–308, 2003.

[27] E. Rückert and A. d'Avella, "Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems," *Front. Comput. Neurosci.*, vol. 7, p. 138, Oct. 2013.

[28] J. Chen and H. Qiao, "Muscle-synergies-based neuromuscular control for motion learning and generalization of a musculoskeletal system," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 6, pp. 3993–4006, Jun. 2021.

[29] J. Chen, S. Zhong, E. Kang, and H. Qiao, "Realizing human-like manipulation with a musculoskeletal system and biologically inspired control scheme," *Neurocomputing*, vol. 339, pp. 116–129, Apr. 2019.

[30] R. S. Razavian, B. Ghannadi, and J. McPhee, "A synergy-based motor control framework for the fast feedback control of musculoskeletal systems," *J. Biomech. Eng.*, vol. 141, no. 3, p. 12, 2019.

[31] J. N. Sanes and J. P. Donoghue, "Plasticity and primary motor cortex," *Annu. Rev. Neurosci.*, vol. 23, no. 1, pp. 393–415, 2000.

[32] M. M. Churchland *et al.*, "Neural population dynamics during reaching," *Nature*, vol. 487, no. 7405, pp. 51–56, 2012.

[33] K. V. Shenoy, M. Sahani, and M. M. Churchland, "Cortical control of arm movements: a dynamical systems perspective," *Annu. Rev. Neurosci.*, vol. 36, pp. 337–359, Jul. 2013.

[34] M. M. Churchland and J. P. Cunningham, "A dynamical basis set for generating reaches," in *Cold Spring Harbor Symposia on Quantitative Biology*, vol. 79. Cold Spring Harbor, NY, USA: Cold Spring Harbor Lab. Press, 2014, pp. 67–80.

[35] A. Afshar, G. Santhanam, M. Y. Byron, S. I. Ryu, M. Sahani, and K. V. Shenoy, "Single-trial neural correlates of arm movement preparation," *Neuron*, vol. 71, no. 3, pp. 555–564, 2011.

[36] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, S. I. Ryu, and K. V. Shenoy, "Cortical preparatory activity: Representation of movement or first cog in a dynamical machine?" *Neuron*, vol. 68, no. 3, pp. 387–400, 2010.

[37] M. M. Churchland, M. Y. Byron, S. I. Ryu, G. Santhanam, and K. V. Shenoy, "Neural variability in premotor cortex provides a signature of motor preparation," *J. Neurosci.*, vol. 26, no. 14, pp. 3697–3712, 2006.

[38] T.-C. Kao, M. S. Sadabadi, and G. Hennequin, "Optimal anticipatory control as a theory of motor preparation: A thalamo-cortical circuit model," *Neuron*, vol. 109, no. 9, pp. 1567–1581, 2021.

[39] G. Hennequin, T. P. Vogels, and W. Gerstner, "Optimal control of transient dynamics in balanced networks supports generation of complex movements," *Neuron*, vol. 82, no. 6, pp. 1394–1406, 2014.

[40] T. Miconi, "Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks," *Elife*, vol. 6, Feb. 2017, Art. no. e20899.

[41] D. G. Thelen, "Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults," *J. Biomech. Eng.*, vol. 125, no. 1, pp. 70–77, 2003.

[42] K. R. Saul *et al.*, "Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model," *Comput. Methods Biomech. Biomed. Eng.*, vol. 18, no. 13, pp. 1445–1458, 2015.

[43] A. A. Russo *et al*. "Motor cortex embeds muscle-like commands in an untangled population response," *Neuron*, vol. 97, no. 4, pp. 953–966, 2018.

[44] G. F. Elsayed, A. H. Lara, M. T. Kaufman, M. M. Churchland, and J. P. Cunningham, "Reorganization between preparatory and movement population responses in motor cortex," *Nat. Commun.*, vol. 7, no. 1, pp. 1–15, 2016.

[45] O. Barak, "Recurrent neural networks as versatile tools of neuroscience research," *Current Opinion Neurobiol.*, vol. 46, pp. 1–6, Oct. 2017.

[46] D. Sussillo, "Neural circuits as computational dynamical systems," *Current Opinion Neurobiol.*, vol. 25, pp. 156–163, Apr. 2014.

[47] J. Chen and H. Qiao, "Motor-cortex-like recurrent neural network and multi-tasks learning for the control of musculoskeletal systems," *IEEE Trans. Cogn. Develop. Syst.*, early access, Dec. 21, 2020, doi: 10.1109/TCDS.2020.3045574.

[48] I. R. Fiete and H. S. Seung, "Gradient learning in spiking neural networks by dynamic perturbation of conductances," *Phys. Rev. Lett.*, vol. 97, no. 4, 2006, Art. no. 048104.

[49] C. Yang, C. Chen, N. Wang, Z. Ju, J. Fu, and M. Wang, "Biologically inspired motion modeling and neural control for robot learning from demonstrations," *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 2, pp. 281–291, Jun. 2019.

[50] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 777–787, Mar. 2019.

[51] N. Hansen, "The CMA evolution strategy: A tutorial," 2016, *arXiv:1604.00772*.

[52] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 312–317.

[53] N. Hansen, "Benchmarking a bi-population cma-es on the bbob-2009 function testbed," in *Proc. 11th Annu. Conf. Companion Genet. Evol. Comput. Late Breaking Papers*, 2009, pp. 2389–2396.

[54] T.-C. Kao and G. Hennequin, "Neuroscience out of control: control-theoretic perspectives on neural circuit dynamics," *Current Opinion Neurobiol.*, vol. 58, pp. 122–129, Oct. 2019.

[55] Y. Sun, H. Shi, and F. Wang, "Learning and encoding motor primitives for limb actions in a brain-like computation approach," *Neurocomputing*, vol. 385, no. 4, pp. 160–168, 2020.

[56] D. C. Crowder, J. Abreu, and R. F. Kirsch, "Hindsight experience replay improves reinforcement learning for control of a mimo musculoskeletal model of the human arm," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1016–1025, May 2021.

[57] H. Su, W. Qi, C. Yang, J. Sandoval, G. Ferrigno, and E. De Momi, "Deep neural network approach in robot tool dynamics identification for bilateral teleoperation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2943–2949, Apr. 2020.

[58] X. Yu, S. Zhang, L. Sun, Y. Wang, C. Xue, and B. Li, "Cooperative control of dual-arm robots in different human-robot collaborative tasks," *Assembly Autom.*, vol. 40, no. 1, pp. 95–104, 2019.